CAN-PN/2, CAN-PN/2-FD

Fieldbus Gateway for connecting PROFINET®-IO with CAN CC and CAN FD

CAN-PN/2 (C.2924.02)

CAN-PN/2-FD (C.2924.62)

Manual

to Product	C.2924.02, C.2924.62

esd electronics gmbh
Vahrenwalder Str. 207 • 30165 Hannover • Germany
http://www.esd.eu
Phone: +49 (0) 511 3 72 98-0 • Fax: +49 (0) 511 3 72 98-68

Notes

esd electronics gmbh
Vahrenwalder Str. 207
30165 Hannover
Germany

Tel.:	+49-511-37298-0
Fax:	+49-511-37298-68
E-Mail:	info@esd.eu
Internet:	www.esd.eu

⚠	This manual contains important information and instructions on safe and efficient handling of the CAN-PN/2. Carefully read this manual before commencing any work and follow the instructions. The manual is a product component, please retain it for future use.

Document Information

Document file:	I:\Texte\Doku\MANUALS\CAN\CAN-PN2(-FD)\CAN-PN2_Manual_en_13.docx
Date of print:	2025-07-10
Document-type number:	DOC0800

Hardware version.:	from Rev. 3.0
Software version:	from Rev. 3.0

Document History

The changes in the document listed below affect changes in the hardware as well as changes in the description of the facts, only.

Rev.	Chapter	Changes versus previous version	Date
1.0	-	First English manual of CAN-PN/2 and CAN-PN/2-FD	2023-06-23
1.1	15.2.3	New chapter: Open Source Software Copy	2023-10-19
1.2	-	Safety notice included in Safety Instructions (page 5)	2024-08-02
	-	Use of “CAN CC” or “CAN Classic” according to CiA® Recommendations	
	1.5.3	Error correction in the “Status LEDs” chapter	
	2	Editorial revision of the chapter	
	7	Add list of new features	
	8.4	Note on chapter “CAN Troubleshooting Guide” corrected.	
	15.1	Table revised	
1.3	-	Safety instructions (pages 5 and 101): Specified standard changed to DIN EN 61140	2025-07-10
	1.2	Figure 2 updated	
	1.4	Figure 4 updated	
	1.5	Fixed error in LED description	
	6	Remove hint that CAN FD monitoring is currently unsupported.	
	10.1	Current consumption and power consumption corrected	

Technical details are subject to change without further notice.

Classification of Warning Messages and Safety Instructions

This manual contains noticeable descriptions, warning messages and safety instructions, which you must follow to avoid personal injuries or death and property damage.

This is the safety alert symbol.
It is used to alert you to potential personal injury hazards. Obey all safety messages and instructions that follow this symbol to avoid possible injury or death.

DANGER, WARNING, CAUTION

Depending on the hazard level the signal words DANGER, WARNING or CAUTION are used to highlight safety instructions and warning messages. These messages may also include a warning relating to property damage.

DANGER
Danger statements indicate a hazardous situation which, if not avoided, will result in death or serious injury.

WARNING.
Warning statements indicate a hazardous situation that, if not avoided, could result in death or serious injury.

CAUTION
Caution statements indicate a hazardous situation that, if not avoided, could result in minor or moderate injury.

NOTICE

Notice statements are used to notify people on hazards that could result in things other than personal injury, like property damage.

NOTICE
This NOTICE statement indicates that the device contains components sensitive to electrostatic discharge.

NOTICE
This NOTICE statement contains the general mandatory sign and gives information that must be heeded and complied with for a safe use.

INFORMATION

INFORMATION
Notes to point out something important or useful.

Safety Instructions

- When working with the CAN-PN/2 follow the instructions below and read the manual carefully to protect yourself from injury and the CAN-PN/2 from damage.
- The assembly is classified as open equipment and must therefore be installed in a control cabinet that is designed for the specific environmental. The control cabinet should be made of metal to improve the electromagnetic immunity of the device. It should be equipped with a key locking mechanism to prevent any unauthorized access.
- Do not use damaged or defective cables to connect the CAN-PN/2 and follow the CAN wiring hints in chapter: "Correct Wiring of Galvanically Isolated CAN Networks".
- In case of damages to the device, which might affect safety, appropriate and immediate measures must be taken, that exclude an endangerment of persons and domestic animals and property.
- The galvanic isolation of the CAN-PN/2 has only functional tasks and is not a protection against hazardous electrical voltage.
- The CAN-PN/2 is a device of protection class III according to DIN EN 61140 and may only be operated on supply circuits that offer sufficient protection against dangerous voltages.
- External circuits connected to the physical ports of the CAN-PN/2 must be sufficiently protected against dangerous voltage.
- The user is responsible for compliance with the applicable national safety regulations.
- Do not open the housing of the CAN-PN/2 .
- The CAN-PN/2 must be securely installed before commissioning.
- The permitted operating position is specified as shown (Figure 4). Other operating positions are not allowed.
- Never let liquids get inside CAN-PN/2. Otherwise, electric shocks or short circuits may result.
- Protect the CAN-PN/2 from dust, moisture, and steam.
- Protect the CAN-PN/2 from shocks and vibrations.
- The CAN-PN/2 may become warm during normal use. Always allow adequate ventilation around the CAN-PN/2 and use care when handling
- Do not operate the CAN-PN/2 adjacent to heat sources and do not expose it to unnecessary thermal radiation. Ensure an ambient temperature as specified in the technical data.

NOTICE
Electrostatic discharges may cause damage to electronic components.

→ Take the appropriate precautions for handling electrostatic discharge sensitive devices.

Qualified Personnel

This documentation is directed exclusively towards personnel qualified in control and automation engineering. The installation and commissioning of the product may only be carried out by qualified personnel, which is authorized to put devices, systems, and electric circuits into operation according to the applicable national standards of safety engineering.

Conformity

The CAN-PN/2 is an industrial product and meets the demands of the EU regulations and EMC standards printed in the conformity declaration at the end of this manual.

Warning: In a residential, commercial, or light industrial environment the CAN-PN/2 may cause radio interferences in which case the user may be required to take adequate measures.

Data Safety

This device is equipped with an Ethernet or other interface which is suitable to establish a connection to data networks. Depending on the software used on the device, these interfaces may allow attackers to compromise normal function, get illegal access or cause damage.
esd does not take responsibility for any damage caused by the device if operated at any networks. It is the responsibility of the device's user to take care that necessary safety precautions for the device's network interface are in place.

Intended Use

The intended use of the CAN-PN/2 is the operation fieldbus gateway for connecting PROFINET®-IO with CAN and CAN FD (CAN-PN/2-FD only).
The guarantee given by esd does not cover damages which result from improper use, usage not in accordance with regulations or disregard of safety instructions and warnings.

- The CAN-PN/2 is a built-in unit for installation e.g. in control cabinets.
- The operation of the CAN-PN/2 in hazardous areas, or areas exposed to potentially explosive materials is not permitted.
- The operation of the CAN-PN/2 for medical purposes is prohibited.

Service Note

The CAN-PN/2 does not contain any parts that require maintenance by the user. The CAN-PN/2 does not require any manual configuration of the hardware. Unauthorized intervention in the device voids warranty claims

Disposal

Products marked with a crossed-out dustbin must not be disposed of with household waste. Devices which have become defective in the long run must be disposed in an appropriate way or must be returned to the manufacturer for proper disposal. Please, contribute to environmental protection.

Typographical Conventions

Throughout this manual the following typographical conventions are used to distinguish technical terms.

Convention	Example
File and path names	**`/dev/null`** or **`<stdio.h>`**
Function names	***open()***
Programming constants	`NULL`
Programming data types	`uint32_t`
Variable names	*Count*

Number Representation

All numbers in this document are base 10 unless designated otherwise. Hexadecimal numbers have a prefix of 0x, and binary numbers have a prefix of 0b. For example, 42 is represented as 0x2A in hexadecimal and 0b101010 in binary.

Table of Contents

List of Tables

List of Figures

1 Overview

1.1 About this Manual

In this manual all variants of the fieldbus gateway, the CAN-PN/2 and the CAN-PN/2-FD are described together as CAN-PN/2. The hardware of the two variants is largely identical. They only differ in the design of the CAN interface. While the CAN-PN/2 variant only supports CAN Classic (CAN CC), the CAN-PN/2-FD is capable of CAN FD support.
Differences of the gateway variants are noted accordingly.

1.2 Description of CAN-PN/2

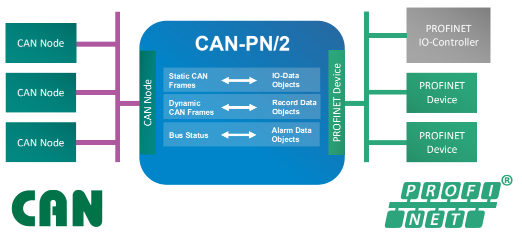
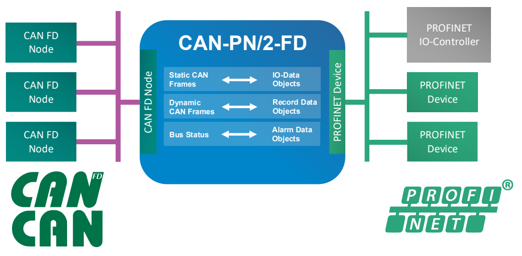
CAN-PN/2 (C.2924.02)	CAN-PN/2-FD (C.2924.62)
	

Figure 1: PROFINET device to CAN CC **Figure 2:** PROFINET device to CAN FD

The CAN-PN/2 gateway offers reliable data exchange between PROFINET IO and CAN. It operates as a PROFINET IO device with a process image of 1440 bytes input data and 1440 bytes output data on the PROFINET bus.
The CAN-PN/2 variant (C.2924.02) supports CAN CC (2.0A/B) with a bit rate of up to 1 Mbit/s. Furthermore, the CAN-PN/2-FD variant (C.2924.62) can connect PROFINET IO with CAN FD. It supports CAN FD with up to 64 bytes in data field and 8 Mbit/s bit rate. Furthermore, it is fully compatible with CAN CC.

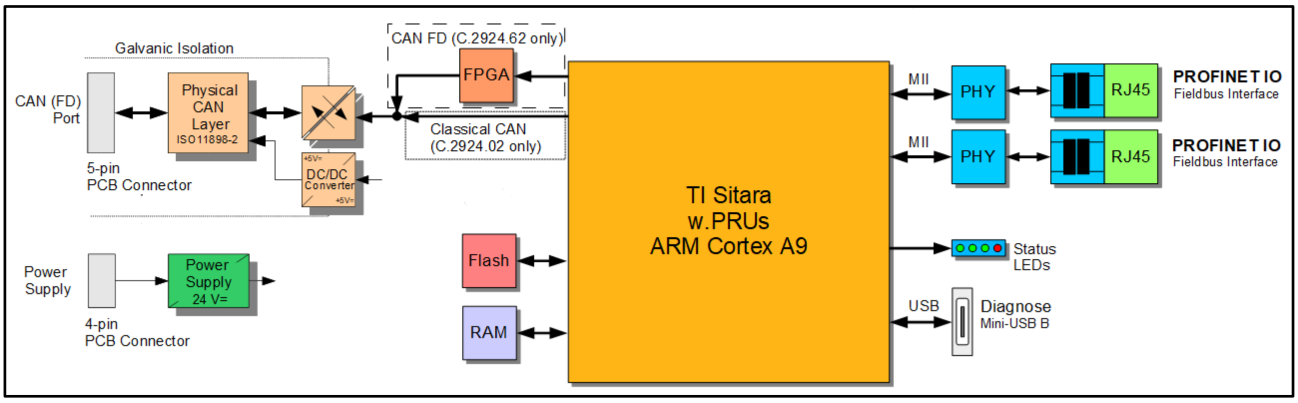


Figure 3: Block circuit diagram

The CAN-PN/2 comes in a compact housing for DIN rail mounting with easily accessible connectors. It is equipped with two Ethernet ports via RJ-45 sockets for PROFINET IO, a CAN port via a connector with spring-cage connection and a Mini-USB-B interface for diagnose and firmware update.

Physical Interfaces

The CAN CC interface and the CAN FD interface (CAN-PN/2-FD only) are ISO 11898-2:2016 compliant.
The 100BASE-TX PROFINET IO interface complies with IEEE802.3 (1) and allows a data transfer rate of 100 Mbit/s.
Both the PROFINET IO and the CAN interface are galvanically isolated from the rest of the circuit.

11bit and 29bit CAN Identifier

The CAN-PN/2 supports 11-bit and 29-bit CAN identifier according to ISO 11898-1:2015 (CAN2.0A/B).

High speed data exchange

The gateway supports the exchange of data between PROFINET IO and CAN interfaces with PROFINET cycle times up to 1ms.

Scheduling data

The CAN-PN/2 supports high precise scheduling of CAN frames in predefined interval.

Alarm Management

The gateway supports an extended alarm management to check the CAN network including the CAN bus status, bus load and the bus statistic.

Configurable for your needs

The CAN-PN/2 is configurable in a simple manner exactly to fit your needs. From simple incoming and outgoing CAN frames to more advanced application, the gateway can be configured to tailor your application.

Monitoring the CAN bus

The gateway provides access to the CAN bus via the Mini-USB using the EtherCAN protocol over RNDIS/TCP, enabling the use of all standard esd tools like CANreal via the NTCAN API.

1.3 Glossary

Abbreviations

Abbreviation	**Term**
API	Application Programming Interface
BSP	Board Support Package
CAL	CAN Application Layer
CAN	Controller Area Network
CAN CC	CAN classic
CAN FD	CAN flexible data rate
CPU	Central Processing Unit
CiA	CAN in Automation
DB	Data Block
DCF	Device Configuration File
EDS	Electronic Data Sheet
GSD	General Station Description
GSDML	General Station Description Markup Language
HW	Hardware
I/O	Input/Output
IO-CS	PROFINET Consumer Status
IO-PS	PROFINET Provider Status
LSB	Least Significant Bit
MSB	Most Significant Bit
n.a.	not applicable
OB	Organization Block
OS	Operating System
PDO	Process Data Object
PRU	Processor Realtime Unit
RTR	Remote Transmission Request
SDK	Software Development Kit
SDO	Service Data Object
USB	Universal Serial Bus
XML	Extensible Markup Language

1.4 View with Connectors

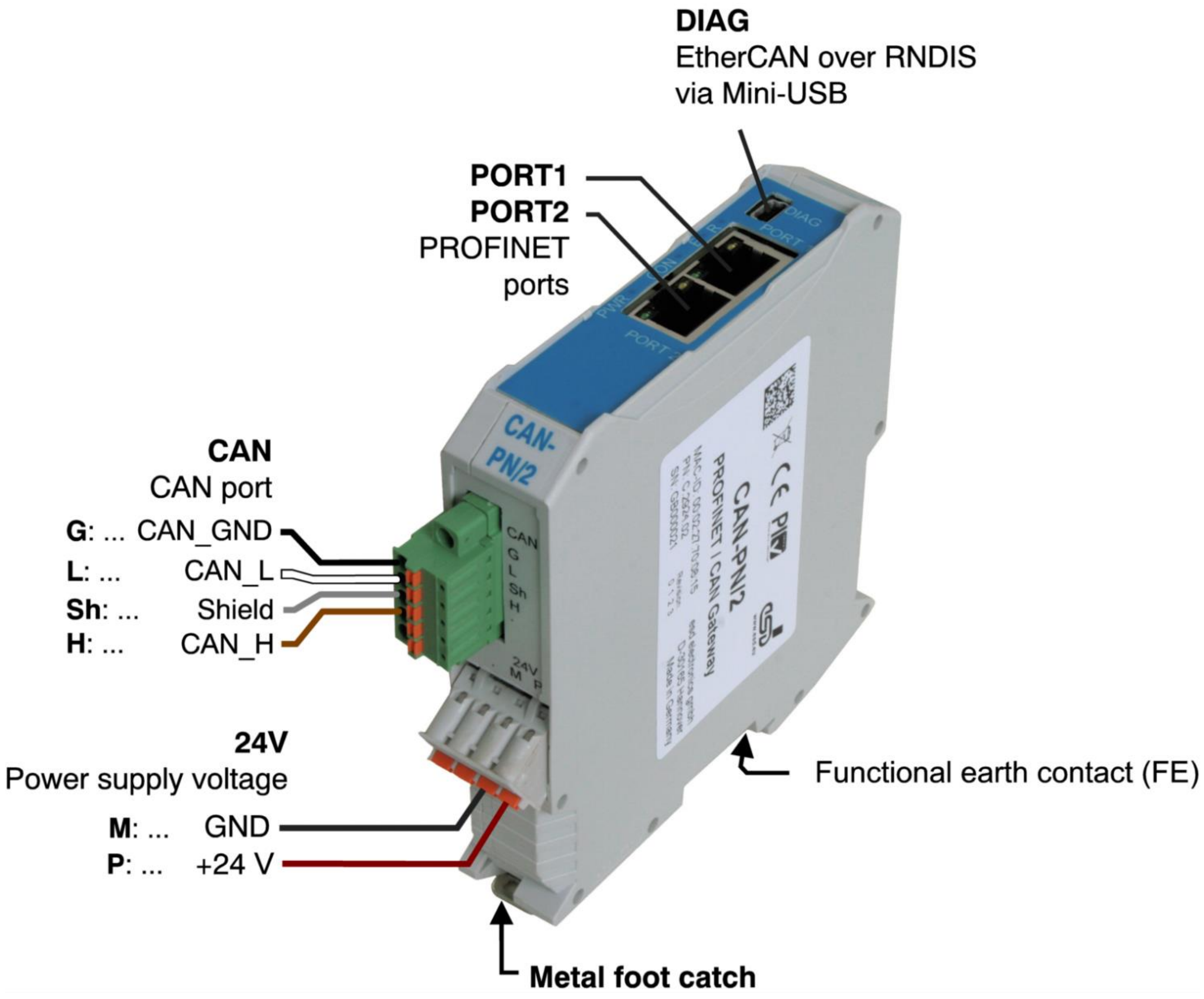


Figure 4: Connecting diagram of CAN-PN/2 (C.2924.02)

The connecting diagram also applies accordingly for the CAN-PN/2-FD.
See also from page 100 for connector assignments and for conductor connection and conductor cross section.

NOTICE
Read chapter " Installing and Uninstalling Hardware" on page 19, before you start with the installation of the hardware!

1.5 LEDs

1.5.1 Position of the LEDs

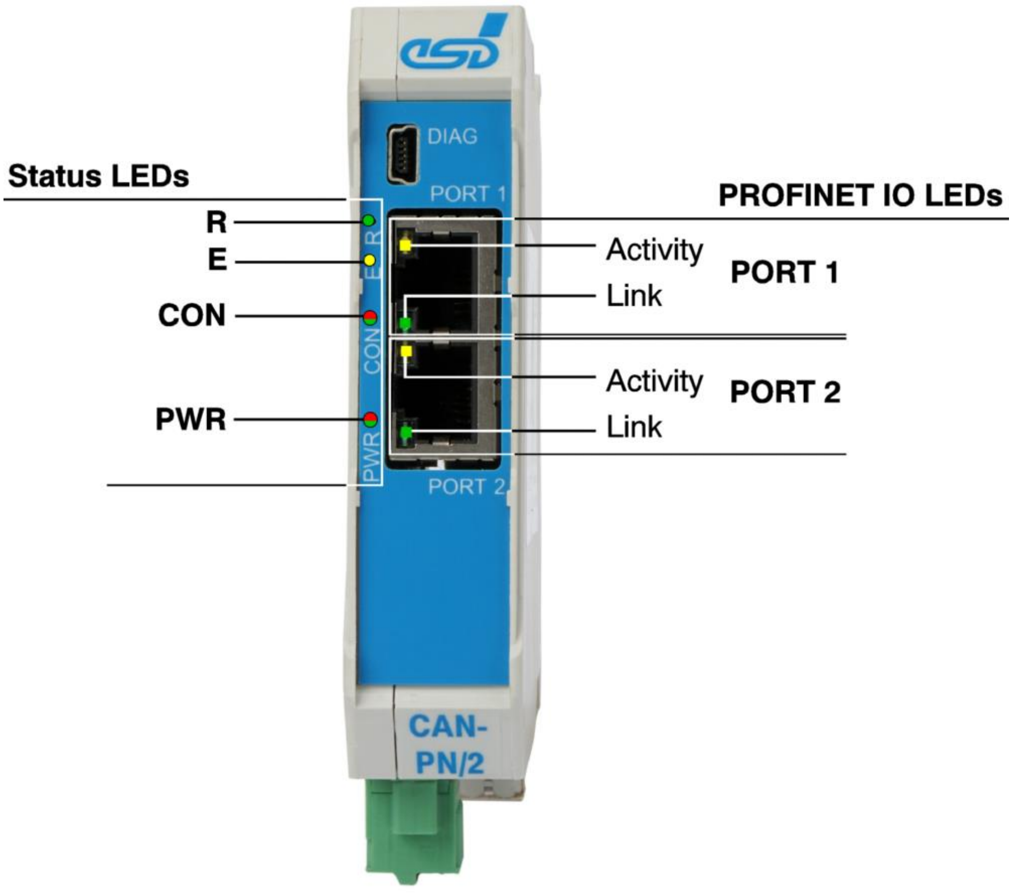


Figure 5: LEDs of CAN-PN/2 (C.2924.02)

The names and positions of the LEDs are identical for the CAN-PN/2-FD.

1.5.2 PRIOFINET IO LEDs

The PROFINET IO LEDs of PORT 1 and PORT 2 are integrated in the RJ-45 sockets. The LEDs indicate the status of the corresponding port.

LED	Colour	Indicator State	Description
Activity	Yellow	Off	No Ethernet connection
		Blinking	Ethernet connection is established; data is transferred
		On	Ethernet connection is established on
Link	Green	Off	No Ethernet connection
		On	Ethernet connection is established

Table 1: Description of PROFINET IO LEDs

1.5.3 Status LEDs

Indicator State	Description
On	LED on
Off	LED off
Blinking	LED blinks with 1 Hz (PROFINET) / 2,5 Hz (CAN)
Single flash	LED 200 ms on, 1000 ms off
Double flash	LED 200 ms on, 200 ms off, 200 ms on, 1000 ms off

Table 2: Indicator states of the Status LEDs

LED	Function	Colour	Indicator State	Description
R	CAN Bus Status	Green	Off	CAN Bus disconnected
			Blinking	*BUS_OFF*
			Single flash	*BUS_ERROR_PASSIVE*
			Double flash	*BUS_WARN*
			On	*BUS OK*
E	Configuration Error	Red	Off	No error
			On	Error in the configuration
CON	PROFINET IO Connect	Green	Off	No valid PROFINET IO link
			Blinking	Request of the PROFINET IO Controller for the identification of the device
			On	Valid PROFINET IO link is established
PWR	Power	Green	Off	No power supply
			On	The gateway has booted successfully

Table 3: Description of Status LEDs

See also chapter 3 on page 21 for a description of the status LEDs during Start-Up.

1.6 Labels

Example

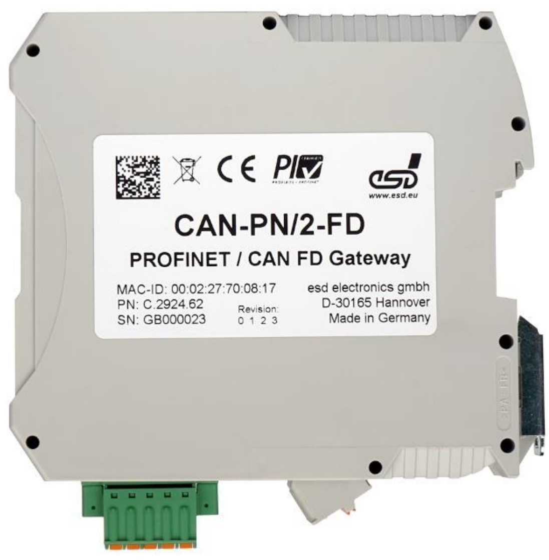


Figure 6: CAN-PN/2-FD with name plate (example)

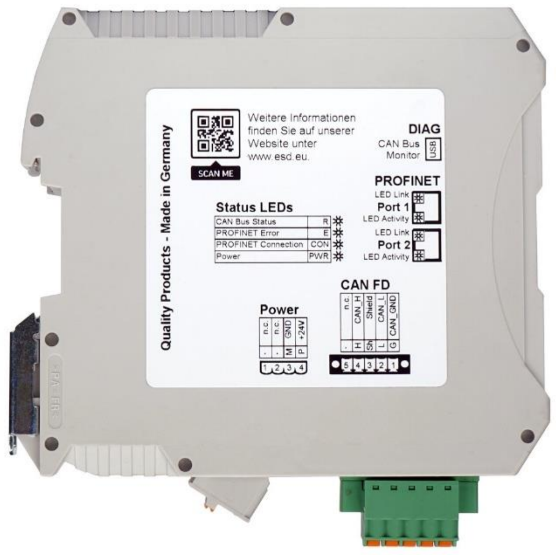


Figure 7: CAN-PN/2-FD with LED/Connector label (example)

The name plate (Figure 5) shows among others the name, MAC-ID, esd order No. (PN) and the serial number (SN).

Name plate	CAN CC variant	CAN FD variant
Name:	CAN-PN/2	CAN-PN/2-FD
MAC-ID:	Individual MAC-ID of the module e.g.: 00:02:27:70:08:15	Individual MAC-ID of the module e.g.: 00:02:27:70:08:17
PN (esd order No.):	C.2924.02	C.2924.62
SN (Serial number):	Individual number of the module e.g. GB000021	Individual number of the module e.g. GB000023

The LED/Connector label (Figure 7) shows short descriptions of the LEDs and connectors and the QR code of esd.

LED/Connector label	CAN CC variant	CAN FD variant
LEDs:	Status LEDs PROFINET LEDs	Status LEDs, PROFINET LEDs
Connectors:	DIAG, PROFINET (Port 1, Port 2), Power, CAN CC	DIAG, PROFINET (Port 1, Port 2), Power, CAN FD

2 Installing and Uninstalling Hardware

To put the CAN-PN/2 into operation, please follow the installation notes.

Step	Procedure	See Page
❗	**NOTICE** **Read the safety instructions at the beginning of this document carefully before you start with the hardware installation!**	5
⚠	**DANGER** Hazardous Voltage - Risk of electric shock due to unintentional contact with uninsulated live parts with high voltages inside of the system into which the CAN-PN/2 is to be integrated. → The CAN-PN/2 is a device of protection class III according to DIN EN IEC 61010-2-201 and may only be operated on supply circuits that offer sufficient protection against dangerous voltages. → External circuits connected to the physical ports of the CAN-PN/2 must be sufficiently protected against dangerous voltages. → Compliance with the applicable national safety regulations is the responsibility of the user. → Ensure the absence of voltage before starting any electrical work.	
To install, continue as described in chapter 2.1‘Installing the Hardware’. To uninstall, continue as described in chapter 2.2 ‘Uninstalling the Hardware’		

2.1 Installing the Hardware

Step	Procedure	See Page
	Follow the safety instructions at the beginning of chapter 2	19
1.	Mount the CAN-PN/2 module and connect the interfaces (power supply voltage, CAN, PROFINET) as described in Figure 4: Connecting diagram of CAN-PN/2	15
	See also chapter 11 for ‘Connector Pin Assignments’.	100
❗	**NOTICE** Incorrect wiring of the 24V power supply voltage can cause damage to the module! → Make absolutely sure to connect the cables correctly to the 24V line connector! → Use only suitable cables for the line plug	101
2.	Please note that the CAN bus must be terminated at both ends! esd offers special T-connectors and termination connectors for external termination. Additionally, the CAN_GND signal must be connected to earth at exactly one point in the CAN network. For details, please read chapter ‘Correct Wiring of Galvanically Isolated CAN Networks’.	105
3.	Switch on the 24 V-power supply voltage of the CAN-PN/2	
4.	Continue with the installation of the software, as described in chapter ‘Software’.	22

2.2 Uninstalling the Hardware

Step	Procedure	See Page
1.	Follow the safety instructions at the beginning of chapter 2	19
2.	Make sure that all connected interfaces and power supply are switched off.	
3.	Disconnect the CAN-PN/2 from the connected interfaces.	
4.	Loosen the fastening of the CAN-PN/2.	
5.	Carefully remove the CAN-PN/2.	

3 Start-Up

After switching on the supply voltage, the CAN-PN/2 starts automatically. During start up the ‘R’ LED (CAN Status) turns on. When the device is started successfully ‘PWR’ LED (Power) turns on and ‘R’ LED (CAN Status) turns off again. This process takes about 10s.

The gateway is now ready to be configured by the PROFINET controller.

When the gateway has established a connection to the PROFINET network, the ‘CON’ LED (PROFINET Connect) turns ON. When the CAN bus is not faulty, the ‘R’ LED (CAN Status) turns on, too.

After the PROFINET controller changes to state RUN, the data exchange is started automatically. When the PLC changes to the state STOP, no more CAN frames are sent.

4 Software

This chapter describes the functionality and installation as well as the configuration of the gateway.

4.1 Functionality

The firmware of the gateway is based on modules, which can be added during the configuration. Each module can be configured based on its type. The gateway supports multiple module types, which can be used for different purposes. For example, there is one module type for incoming and another one for outgoing frames. Each module type comes with its own parameters. All module types are declared in chapter 4.4. The gateway can be configured with up to 512 of these modules.

These modules contain Input or Output data which are directly mapped into the PLC Address Space. For example, an input module would update the data with the reception of a CAN frame. The updated data are transferred via PROFINET to the PLC and updated in the PLC Address Space.

The following figure represents the basic functionality of the firmware for incoming and outgoing frames.

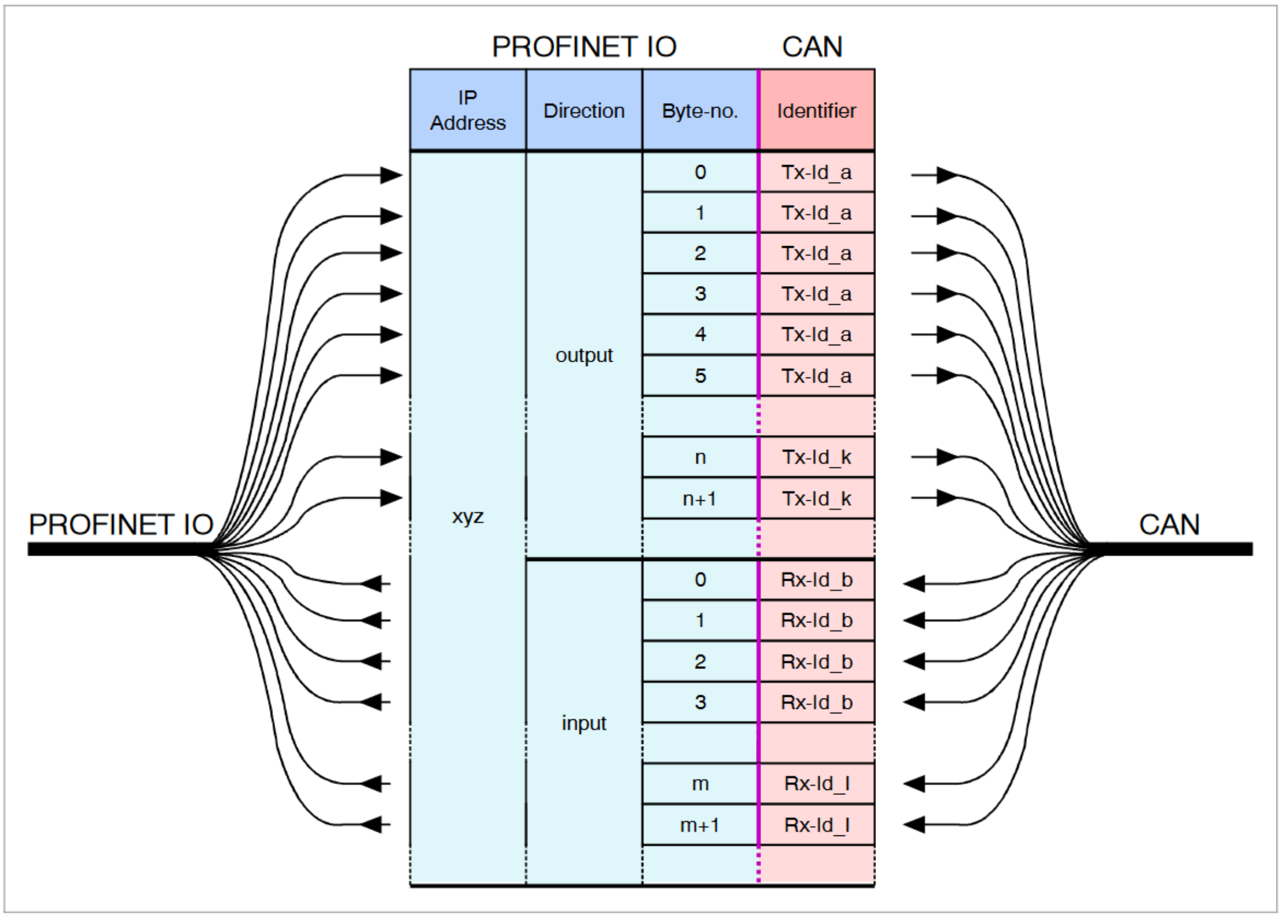


Figure 8: Basic functionality of the firmware

4.2 Installation

The device comes with an installer called `CAN-PN/2(-FD)_X_X_X.exe`. The installation of the installer is mandatory for firmware updates and CAN monitoring. However, to use the base functionality of the device, it is also possible to just download the GSDML file from the esd website.

This installer provides the following packages:

GSDML File	This package includes the GSDML file, which is necessary to configure the gateway.
RNDIS Driver	This package contains the RNDIS driver. It is used to connect the Mini-USB interface to a Windows® computer. The connection is used for firmware updates and CAN monitoring. The driver is installed automatically.
CAN Driver	This package provides the esd CAN-API (NTCAN). It is necessary for the esd CAN-SDK.
esd CAN-SDK	This package contains software for the CAN monitoring and diagnostics, especially the monitoring tool *CANreal*, which can be used to detect and send CAN frames on the bus.
Examples	This package includes a TIA Portal project with some examples.

NOTICE
The CAN driver and the CAN-SDK is not automatically deleted if the CAN-PN software is removed. Therefore use “Software” of the Windows system administration
and remove “EtherCAN [...] Host Driver” and “CAN SDK for Windows”.

4.2.1 Manual Installation of the RNDIS Driver

The RNDIS driver is installed automatically with the installer. To check whether the RNDIS driver is installed correctly, connect the Mini-USB interface with the computer. When the installation has been successful, a new network adapter called *RNDIS based ESD Device* should be displayed in the *Device Manager*. If this is not the case, install the driver manually with the following steps:

Step	Action
1.	Connect the gateway with the computer using the Mini-USB interface.
2.	Open the *Device Manager* and search for a new *Serial USB-Device (COMX)* under *Ports (COM & LPT)*.
3.	Right click on the device and select *update driver*. In the following dialog select *Update driver*.
4.	Click *Browse my computer for drivers*.
4.	Select the installation path of the CAN-PN/2 and check the checkbox *Include subfolders*. By default, this installation path is `C:\Program Files (x86)\esd\CAN-PN`.
5.	Press *Next*.
6.	Now there should be a network adapter called *RNDIS based ESD Device*.

Table 4: Manual installation of the RNDIS driver

4.2.2 GSDML File

The GSDML file is installed with the installer or can be downloaded from our website. There is a GSDML file each for the CAN CC and the CAN FD variant. The GSDML file of the predecessor, the CAN-PN (C.2920.02) is still fully supported and compatible with both variants.

NOTICE
It is not possible to use the GSDML file of the CAN FD variant (CAN-PN/2-FD) for the CAN CC variant (CAN-PN/2, C.2924.02).
Although it is possible to configure both CAN-PN/2 variants with the GSDML file of the predecessor CAN-PN, this does not apply vice versa.

4.3 Configuration

This chapter describes the steps which are relevant to configure the CAN-PN/2. The steps are shown with the Siemens TIA Portal as development environment. For further information about your development environment or the TIA Portal, please read the respective documentation.

4.3.1 Quick Start Guide

Step	Action	See from Page
1	Disconnect the online connection in the TIA Portal, because the hardware and software must be compiled in offline mode.	-
2	Change into the project view of the TIA Portal.	-
3	Install the GSDML file as described in chapter 4.3.2.	26
4	Insert the CAN-PN/2 in your project as described in chapter 4.3.3.	26
5	Configure the CAN and PROFINET interfaces as described in chapter 4.3.4, 4.3.5 and 4.3.6.	27
5	Add modules based of the functionality that should be achieved. For further information, see chapter 4.3.7.	30
6	Compile and load the hardware and software as described in chapter 4.3.8.	31
7	Go online as described in chapter 4.3.8.	31

Table 5: Configuration Quick Start Guide

4.3.2 Installation of the GSDML File

To use the GSDML file, it has to be installed into the development environment. To achieve this, switch to the *project view* in the program window of your TIA Portal. Click on *Options* in the taskbar and select *Manage general station description files (GSD)*.

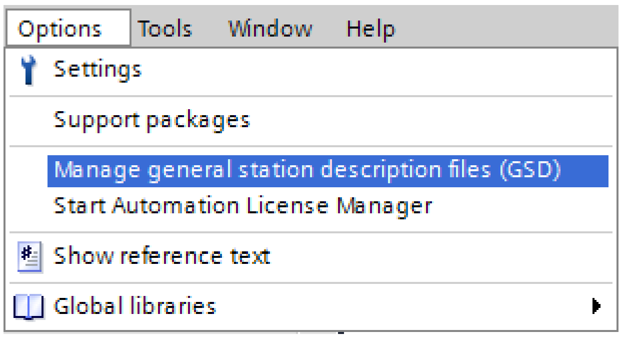
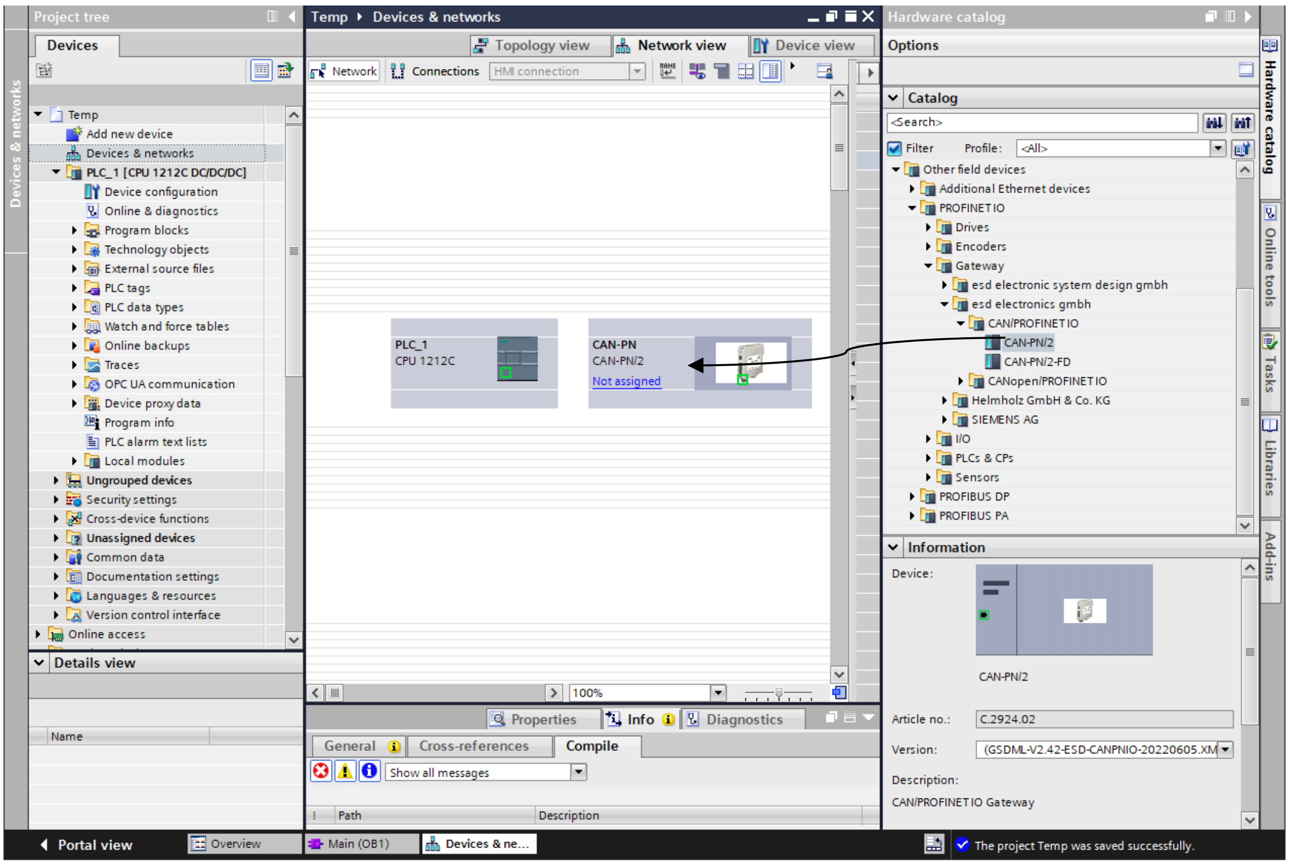


Figure 9: Manage GSDML files

A new dialog appears, in which the path to the directory of the GSDML file must be entered. After installing the installer, the GSDML file is usually located in: `C:\Program Files(x86)\esd\CAN-PN\GSDML`. Select the GSDML file and press *Install*.

4.3.3 Insert the CAN-PN/2

After installing the GSDML file, the PROFINET network can be assembled. Therefore, click under *Project tree → Devices* onto *Devices & networks* as shown in the following figure.
The so-called *Network view* opens. The CAN-PN/2 can now be added from the *Hardware Catalog*. Select your device variant *CAN-PN/2* or *CAN-PN/2-FD* under *Other field devices → Gateway → esd electronics gmbh → CAN/PROFINET-IO*. To insert your device, drag it onto the *Network view*.



Article no.: C.2924.02
Version: (GSDML-V2.42-ESD-CANPNIO-20220605.XM
Description: CAN/PROFINET IO Gateway

The project Temp was saved successfully.

Figure 10: Inserting the CAN-PN/2 (example)

4.3.4 Configuration of the CAN Bus

The CAN bus can be configured with the CAN interface module (*CAN-PN*) in slot 0. To change the module parameters, select the slot, open the *Module parameters,* and set the CAN bitrate to the correct value.

NOTICE
All CAN nodes in the CAN network need to have the same bitrate to work properly.

4.3.5 Assign the PROFINET Network

First the CAN-PN/2 must be assigned to a PROFINET network. To accomplish this, go to the *Network view*, press the button *Not assigned* and click on one of the available PROFINET networks.

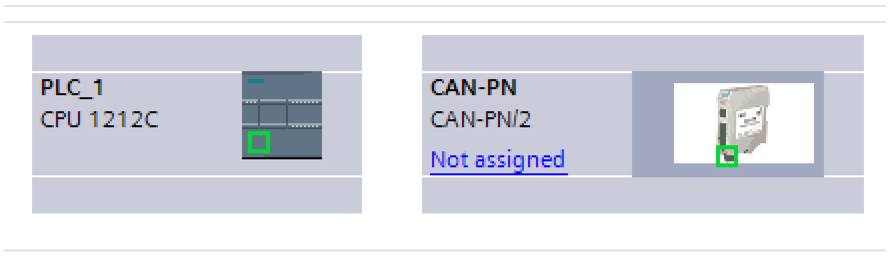


Figure 11: Not assigned CAN-PN/2 (example)

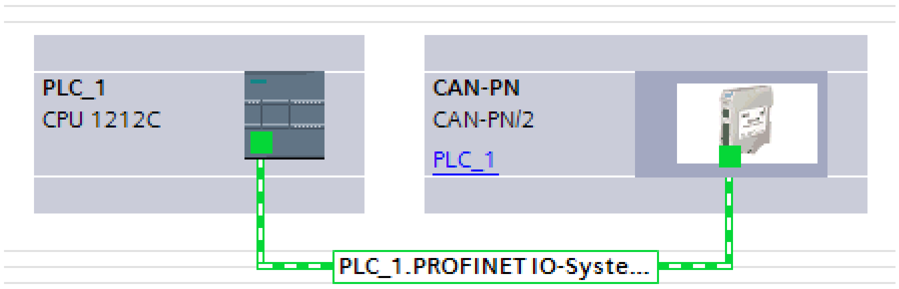


Figure 12: Assigned CAN-PN/2 (example)

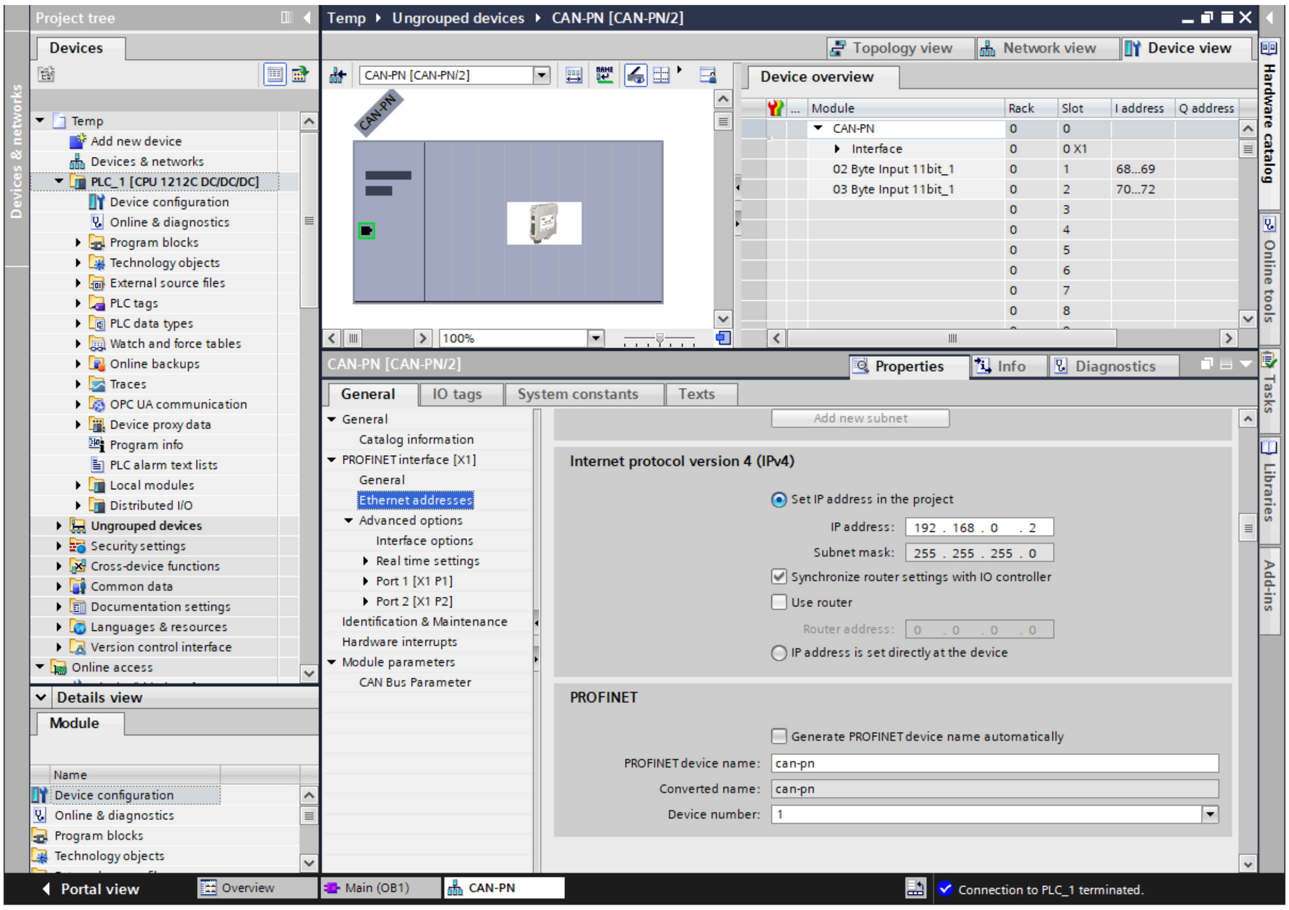
4.3.6 Assign IP Address and PROFINET Device Name

The IP address and the PROFINET device name of the configuration must match those persistently stored in the CAN-PN/2 gateway to work correctly. Both can be configured separately.

NOTICE
Each IP Address and PROFINET device name can only be assigned once per PROFINET network. The IP address normally does not have to be changed manually.

IP Address and Device Name of the Configuration

The IP address and device name of the configuration are generated automatically by default. However, it can be changed manually.
To change it, click on the tab *Device view* of the gateway and select CAN-PN in *Slot 0* of the section *Device overview*. Now open the tabs *Properties* → *General* → *Ethernet addresses* and search for the parameters *IP address* and *PROFINET device name*.



Module	Rack	Slot	I address	Q address
CAN-PN	0	0		
Interface	0	0 X1		
02 Byte Input 11bit_1	0	1	68...69	
03 Byte Input 11bit_1	0	2	70...72	
	0	3		
	0	4		
	0	5		
	0	6		
	0	7		
	0	8		

Internet protocol version 4 (IPv4)

[✓] Set IP address in the project
IP address: 192 . 168 . 0 . 2
Subnet mask: 255 . 255 . 255 . 0
[✓] Synchronize router settings with IO controller
[ ] Use router
Router address: 0 . 0 . 0 . 0
[ ] IP address is set directly at the device

PROFINET

[ ] Generate PROFINET device name automatically
PROFINET device name: can-pn
Converted name: can-pn
Device number: 1

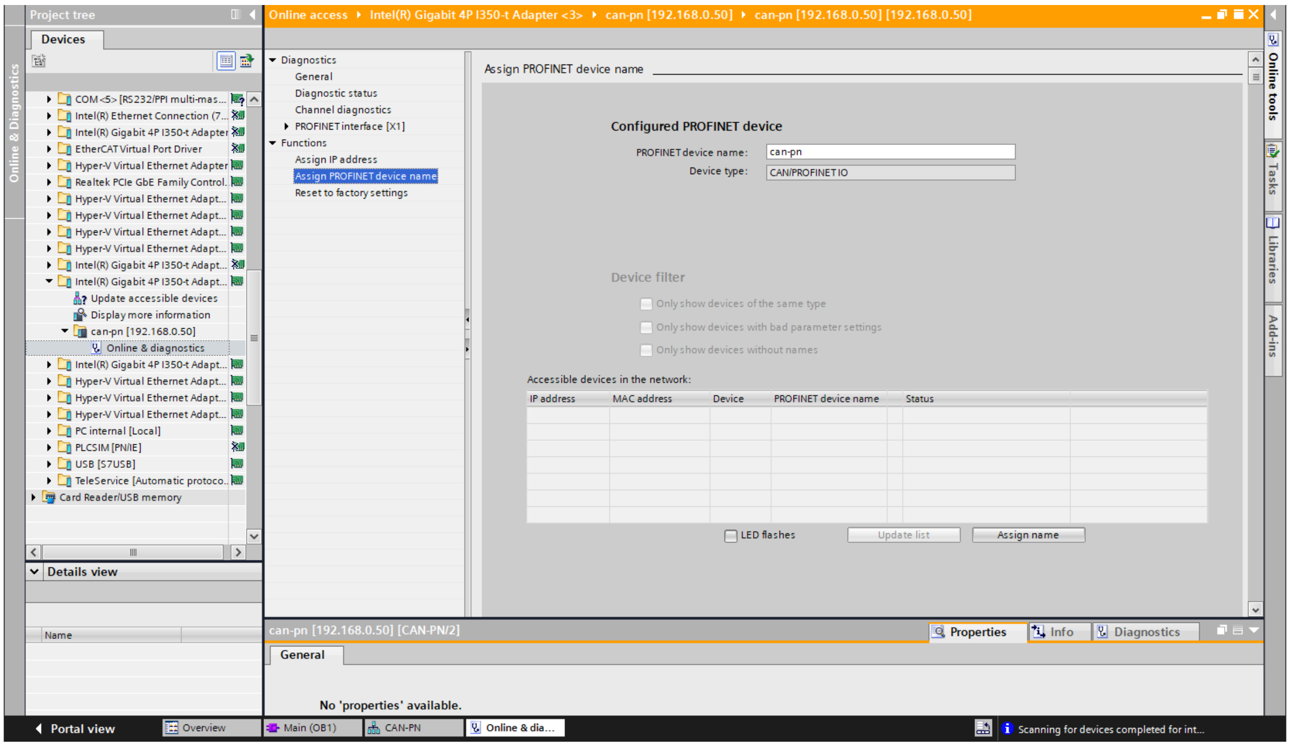
Connection to PLC_1 terminated.

Figure 13: Assign IP address and device name of the configuration

IP Address and PROFINET Device Name of the Gateway

The current IP address and PROFINET device name of the gateway can be found in the *Online Access* section. To display it, click *Update accessible devices* under *Project tree* → *Devices* → *Online Access* → *[Network adapter].* The name as well as the IP address should be displayed.

To change the IP address or the name, expand the device by clicking on the icon and open *Online & diagnostics*. A new dialog will appear as shown in the figure below. Expand *Functions* and select *Assign IP address* or *Assign PROFINET device name*. Insert the new parameter and press the *Assign IP address* or *Assign name* button.


Project tree
Devices
COM <5> [RS232/PPI multi-mas...
Intel(R) Ethernet Connection (7...
Intel(R) Gigabit 4P I350-t Adapter
EtherCAT Virtual Port Driver
Hyper-V Virtual Ethernet Adapter
Realtek PCIe GbE Family Control...
Hyper-V Virtual Ethernet Adapt...
Hyper-V Virtual Ethernet Adapt...
Hyper-V Virtual Ethernet Adapt...
Hyper-V Virtual Ethernet Adapt...
Intel(R) Gigabit 4P I350-t Adapt...
Intel(R) Gigabit 4P I350-t Adapt...
Update accessible devices
Display more information
can-pn [192.168.0.50]
Online & diagnostics
Intel(R) Gigabit 4P I350-t Adapt...
Hyper-V Virtual Ethernet Adapt...
Hyper-V Virtual Ethernet Adapt...
Hyper-V Virtual Ethernet Adapt...
PC internal [Local]
PLCSIM [PN/IE]
USB [S7USB]
TeleService [Automatic protoco...
Card Reader/USB memory
Details view
Name
Online & Diagnostics
Online access ▸ Intel(R) Gigabit 4P I350-t Adapter <3> ▸ can-pn [192.168.0.50] ▸ can-pn [192.168.0.50] [192.168.0.50]
Diagnostics
General
Diagnostic status
Channel diagnostics
PROFINET interface [X1]
Functions
Assign IP address
Assign PROFINET device name
Reset to factory settings
Assign PROFINET device name
Configured PROFINET device
Device filter
Only show devices of the same type
Only show devices with bad parameter settings
Only show devices without names
Accessible devices in the network:
IP address
MAC address
Device
PROFINET device name
Status
LED flashes
Update list
Assign name
Properties
Info
Diagnostics
General
Online tools
Tasks
Libraries
Add-ins
Portal view
Overview
Main (OB1)
CAN-PN
Online & dia...

PROFINET device name: can-pn
Device type: CAN/PROFINET IO

can-pn [192.168.0.50] [CAN-PN/2]

No 'properties' available.

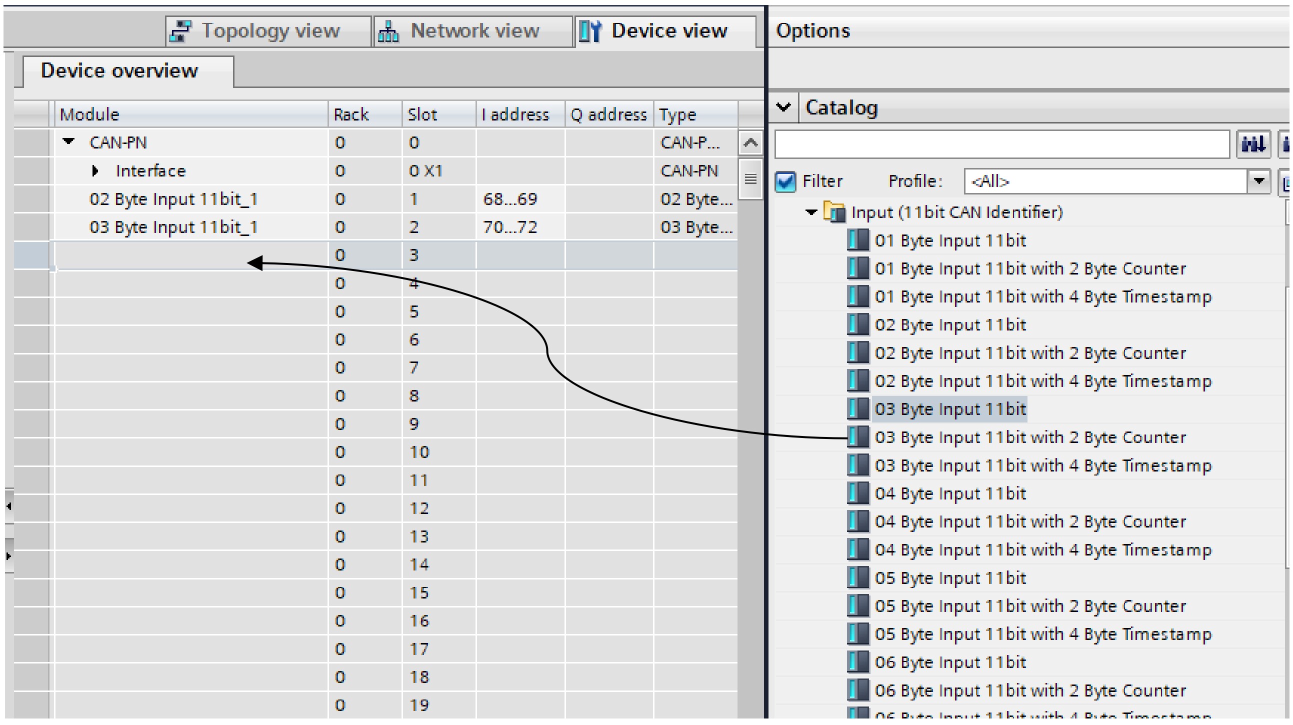
Scanning for devices completed for int...

Figure 14: Assign IP address and device name of the gateway

4.3.7 Add Modules

By default, only the module *CAN-PN* in *Slot 0* is configured. It is of module type *CAN Interface* (see chapter 4.4.1). This module is necessary and cannot be deleted.

To add other modules, go to the *Device view* and open the *Hardware Catalog*. To use a module, just drag and drop it on a slot on the *Device overview*. There are multiple module types which are separated in different folders and are explained in chapter 4.4.



Module	Rack	Slot	I address	Q address	Type
CAN-PN	0	0			CAN-P...
Interface	0	0 X1			CAN-PN
02 Byte Input 11bit_1	0	1	68...69		02 Byte...
03 Byte Input 11bit_1	0	2	70...72		03 Byte...
	0	3			
	0	4			
	0	5			
	0	6			
	0	7			
	0	8			
	0	9			
	0	10			
	0	11			
	0	12			
	0	13			
	0	14			
	0	15			
	0	16			
	0	17			
	0	18			
	0	19			

- Input (11bit CAN Identifier)
  - 01 Byte Input 11bit
  - 01 Byte Input 11bit with 2 Byte Counter
  - 01 Byte Input 11bit with 4 Byte Timestamp
  - 02 Byte Input 11bit
  - 02 Byte Input 11bit with 2 Byte Counter
  - 02 Byte Input 11bit with 4 Byte Timestamp
  - 03 Byte Input 11bit
  - 03 Byte Input 11bit with 2 Byte Counter
  - 03 Byte Input 11bit with 4 Byte Timestamp
  - 04 Byte Input 11bit
  - 04 Byte Input 11bit with 2 Byte Counter
  - 04 Byte Input 11bit with 4 Byte Timestamp
  - 05 Byte Input 11bit
  - 05 Byte Input 11bit with 2 Byte Counter
  - 05 Byte Input 11bit with 4 Byte Timestamp
  - 06 Byte Input 11bit
  - 06 Byte Input 11bit with 2 Byte Counter

Figure 15: Adding modules to the CAN-PN/2

NOTICE
It is allowed to leave gaps between two modules. However, leaving large gaps is not recommended as this may affect the clarity of the configuration.

4.3.8 Compile and Download Hardware and Software

Before the software can be download, it must be compiled.
During this process the TIA Portal must be in offline mode.
To compile the software, select the device (PLC_1 in this case*)* in the field *Project tree → Devices* and click *Compile → Hardware and Software (only changes)* in the pull-down menu.

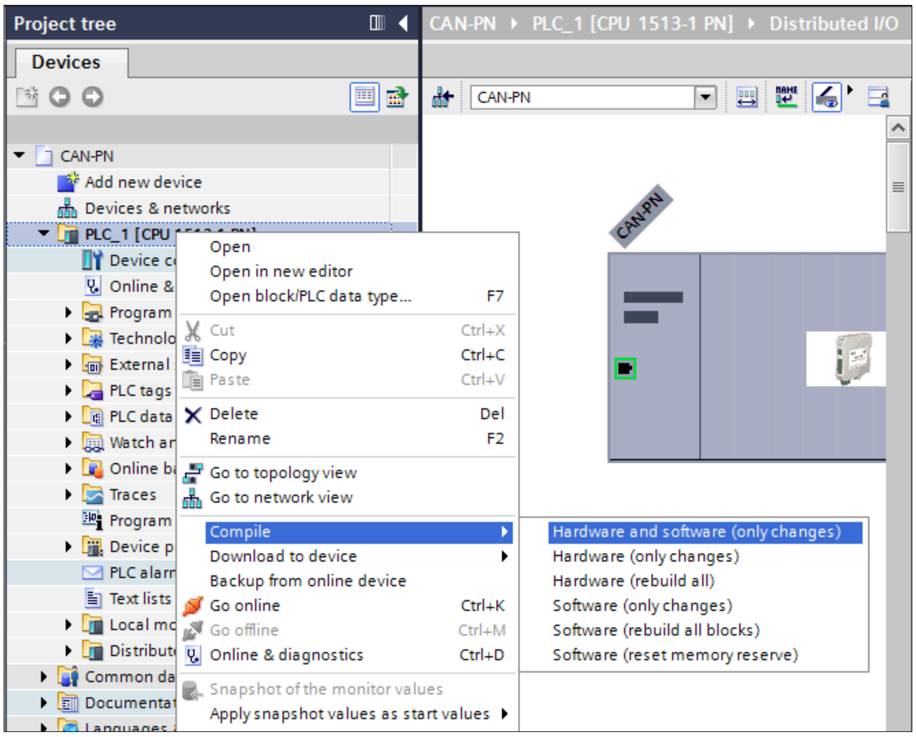


Figure 16: Compile Hardware and Software (detail)

The configuration is compiled. Now the hardware and software can be downloaded to the device. Select your device (PLC_1 in this case*)* again and click *Download → Hardware and Software* in the pull-down menu. A new dialog opens in which the PLC can be chosen.

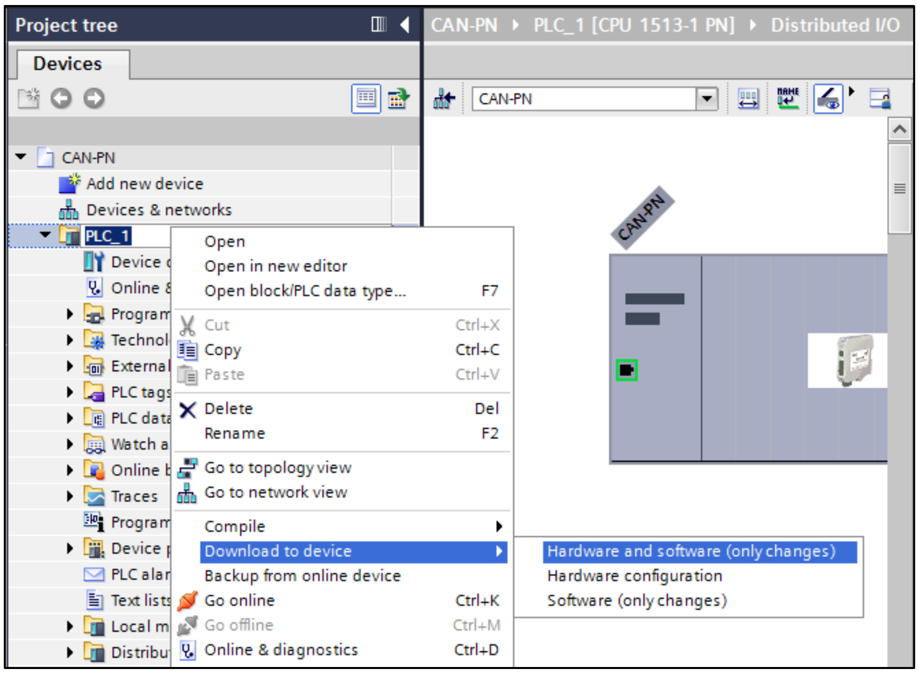


Figure 17: Download hardware and software to device (detail)

The configuration is now successfully passed to the device.

Click on the *Go online* button in the toolbar to go online.

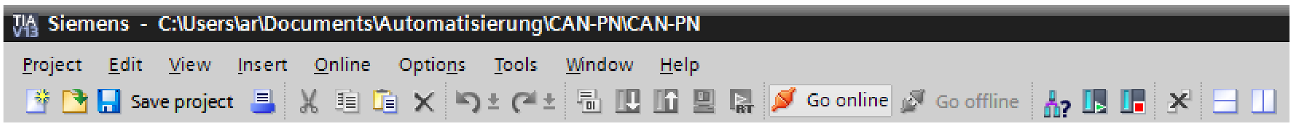
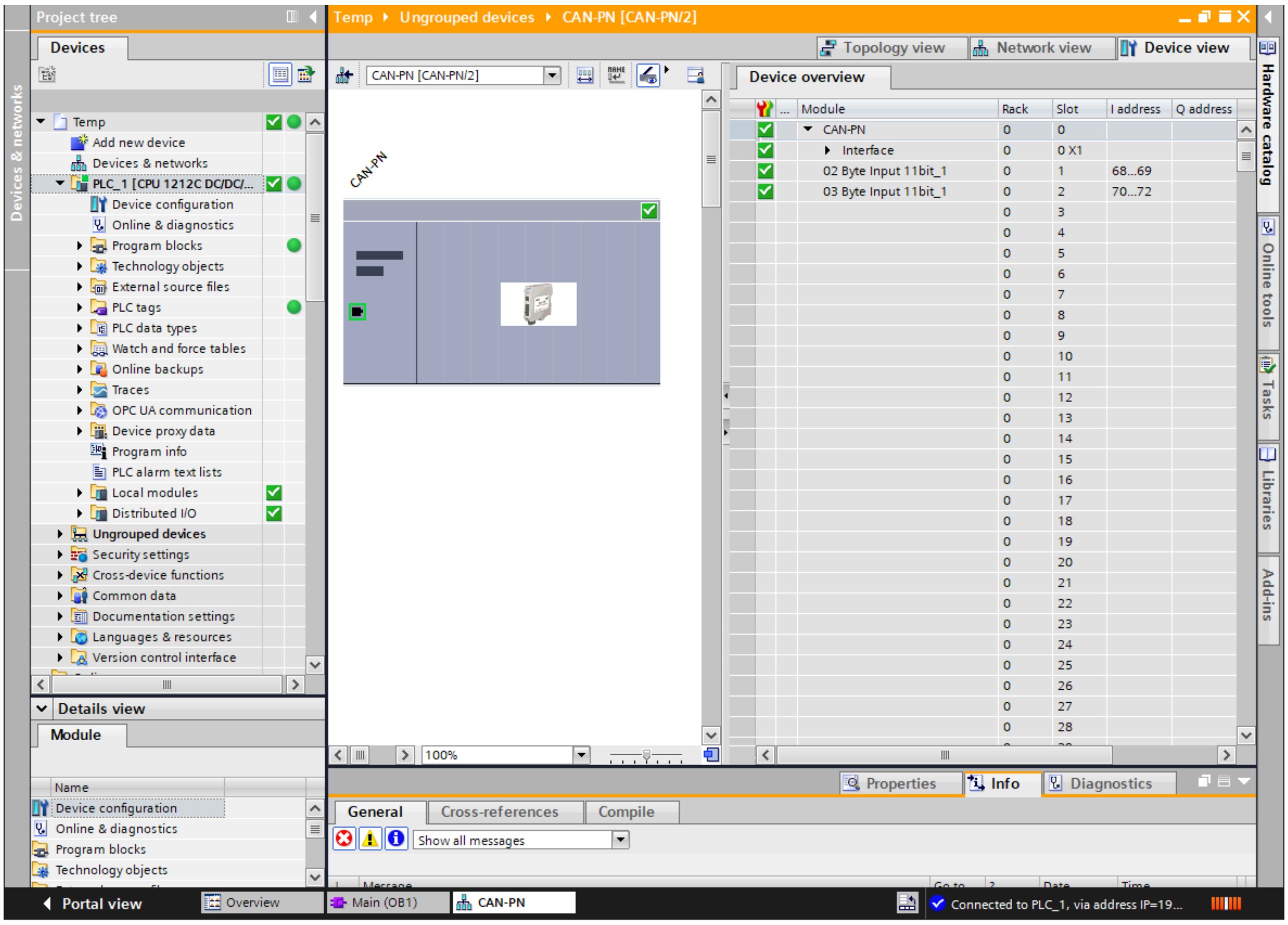


Figure 18: Toolbar with *Go online* button

The online connection is now established.

To check whether the device is working properly, open the *Device View* and see if all check marks are green (see Figure 19). If this is not the case, something is wrong.
Read chapters 4.3.9, 4.5 and 8 for further information.



	...	Module	Rack	Slot	I address	Q address
✓		CAN-PN	0	0		
✓		Interface	0	0 X1		
✓		02 Byte Input 11bit_1	0	1	68...69	
✓		03 Byte Input 11bit_1	0	2	70...72	
			0	3		
			0	4		
			0	5		
			0	6		
			0	7		
			0	8		
			0	9		
			0	10		
			0	11		
			0	12		
			0	13		
			0	14		
			0	15		
			0	16		
			0	17		
			0	18		
			0	19		
			0	20		
			0	21		
			0	22		
			0	23		
			0	24		
			0	25		
			0	26		
			0	27		
			0	28		

Connected to PLC_1, via address IP=19...

Figure 19: Toolbar Online device overview

4.3.9 Configuration Errors

Although many possible configuration errors are intercepted by defining a valid value range for the parameters, there are some configuration errors that can appear. Whenever an online connection is enabled and this symbol is shown in the *Device overview* , there is a configuration error.

The following modules can cause a configuration error:

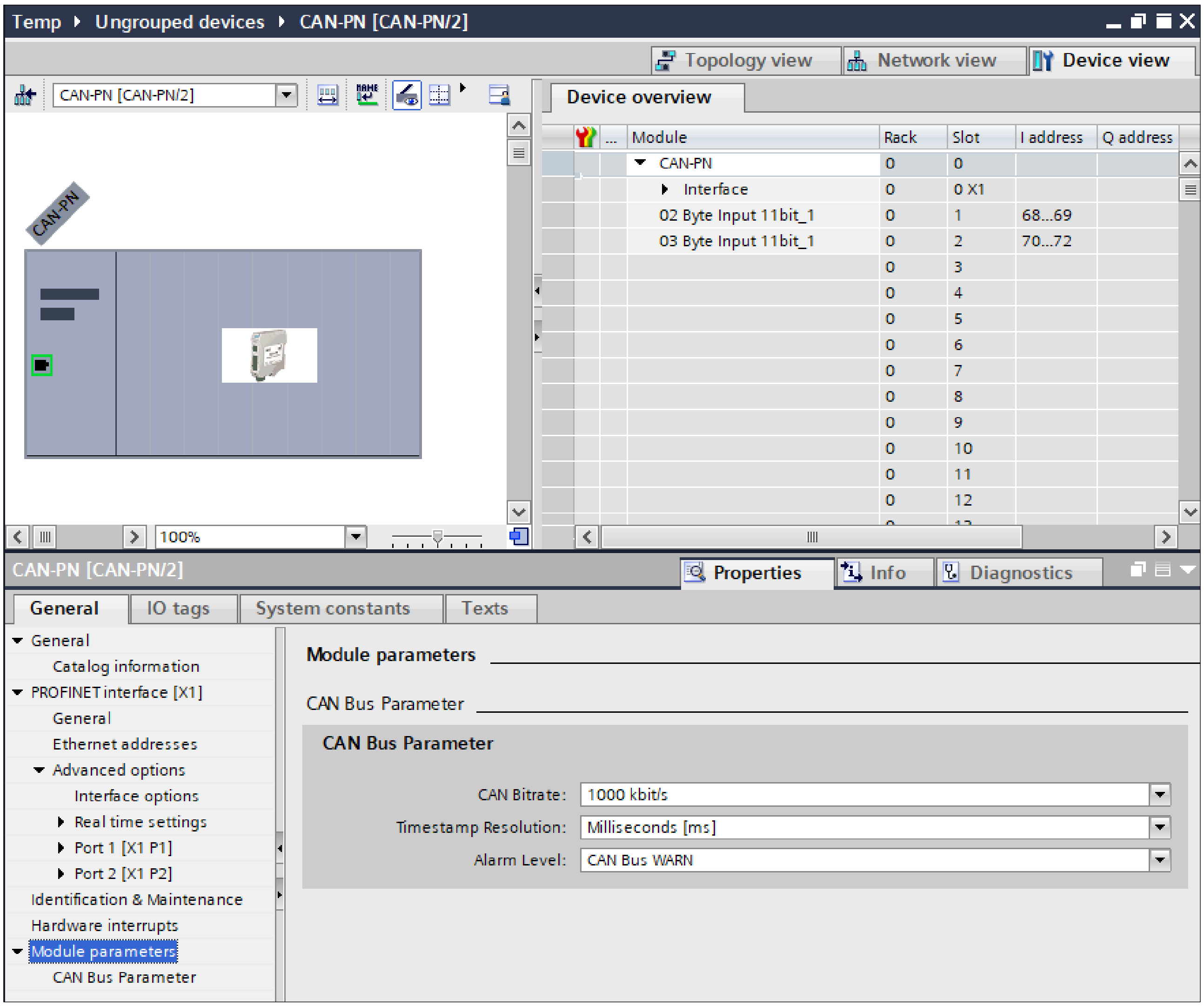
Module	Error	Solution
Bus Load, Bus Status, RX-FIFO, TX-FIFO, Communication Window	This module is unique and cannot be configured multiple times.	Delete all modules of this type except one.
CAN Interface (always Slot 0)	CAN-PN/2-FD only: The CAN Data-Phase bitrate is invalid.	The CAN Data-Phase bitrate must be the same or higher than the CAN bitrate.
All FD modules	CAN-PN/2-FD only: CAN FD interface is not enabled, but a CAN FD module is used.	Enable the CAN FD interface.

Table 6: Configuration errors

NOTICE
It is not allowed to use multiple Input and Output modules with the same CAN identifier in conjunction with cyclic operations. This does not lead to a configuration error, but only the first module is configured correctly. This could lead to unexpected behaviour.

4.4 Modules

The configuration is based on multiple modules, which can be parameterized separately. Each module provides some kind of functionality. The gateway can be configured with up to 512 of these modules. By default, the module *CAN-PN* is configured in *Slot 0* and contains the CAN as well as the PROFINET interface. This module is fixed and cannot be deleted. Each module has its own parameters which can be edited by clicking on the module in *Device overview* and selecting *Properties → General → Module parameters* (see Figure 20).



Module	Rack	Slot	I address	Q address
CAN-PN	0	0		
Interface	0	0 X1		
02 Byte Input 11bit_1	0	1	68...69	
03 Byte Input 11bit_1	0	2	70...72	
	0	3		
	0	4		
	0	5		
	0	6		
	0	7		
	0	8		
	0	9		
	0	10		
	0	11		
	0	12		

- General
  - Catalog information
- PROFINET interface [X1]
  - General
  - Ethernet addresses
  - Advanced options
    - Interface options
    - Real time settings
    - Port 1 [X1 P1]
    - Port 2 [X1 P2]
- Identification & Maintenance
- Hardware interrupts
- Module parameters
  - CAN Bus Parameter

Module parameters

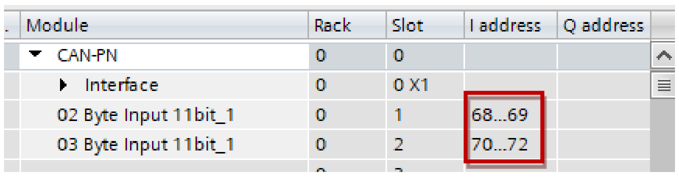
CAN Bus Parameter

CAN Bus Parameter

CAN Bitrate:	1000 kbit/s
Timestamp Resolution:	Milliseconds [ms]
Alarm Level:	CAN Bus WARN

Figure 20: Module Parameters

Most of the modules provide some kind of input and output data, which are directly mapped into the PLC address space. The exact position is shown in the *Device overview* and can be changed if necessary (see Figure 21).



Module	Rack	Slot	I address	Q address
CAN-PN	0	0		
Interface	0	0 X1		
02 Byte Input 11bit_1	0	1	68...69	
03 Byte Input 11bit_1	0	2	70...72	

Figure 21: PLC Address Space of the Modules (detail)

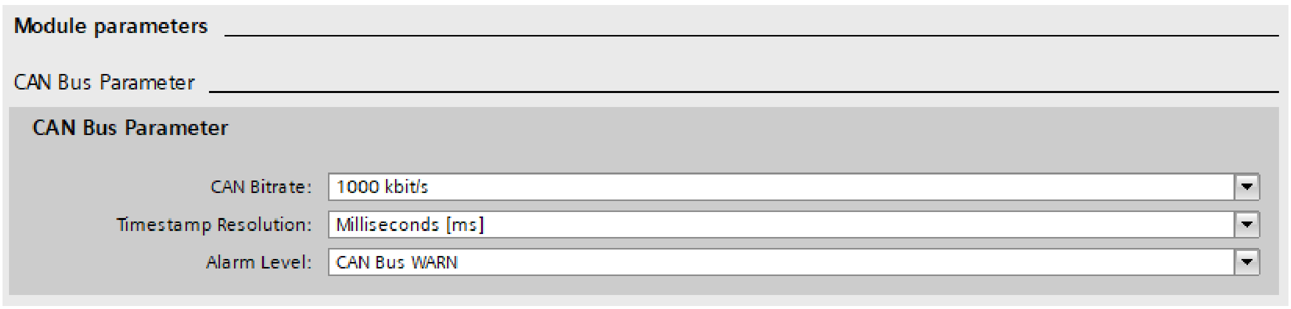
4.4.1 CAN Interface

The CAN interface module is always in Slot 0. It does not provide any input or output data.

The following parameters can be configured:

Parameter	Description
CAN Bitrate	This parameter describes the CAN bitrate in kbits/s. It must be the same on every CAN device on the CAN network.
Timestamp Resolution	This parameter describes the resolution of the timestamps used for *Input Frames* (see chapter 4.4.2).
Alarm Level	This parameter defines at which state of the CAN bus an alarm is sent to the PLC.

Table 7: CAN Interface Parameter


Module parameters

CAN Bus Parameter

CAN Bus Parameter

CAN Bitrate: 1000 kbit/s

Timestamp Resolution: Milliseconds [ms]

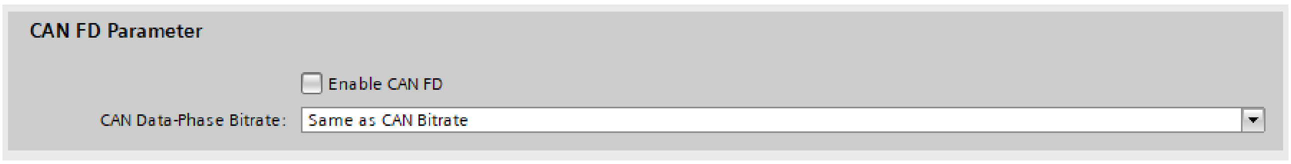
Alarm Level: CAN Bus WARN

Figure 22: CAN Interface Parameter

Additional CAN FD Parameters (CAN-PN/2-FD only):
The CAN-PN/2-FD has some additional parameters, that can be configured as follows:

Parameter	Description
CAN FD	This parameter enables the CAN FD interface and is necessary to use a CAN FD module.
CAN Data-Phase Bitrate	This parameter describes the CAN bitrate during the data-phase. It must be equal or higher than the CAN bitrate. The option *Same as CAN Bitrate* uses the CAN bitrate also for the data-phase bitrate.

Table 8: CAN FD Interface Parameter


CAN FD Parameter

☐ Enable CAN FD

CAN Data-Phase Bitrate: Same as CAN Bitrate

Figure 23: CAN FD Interface Parameter

4.4.2 Input (11-bit and 29-bit CAN Identifier)

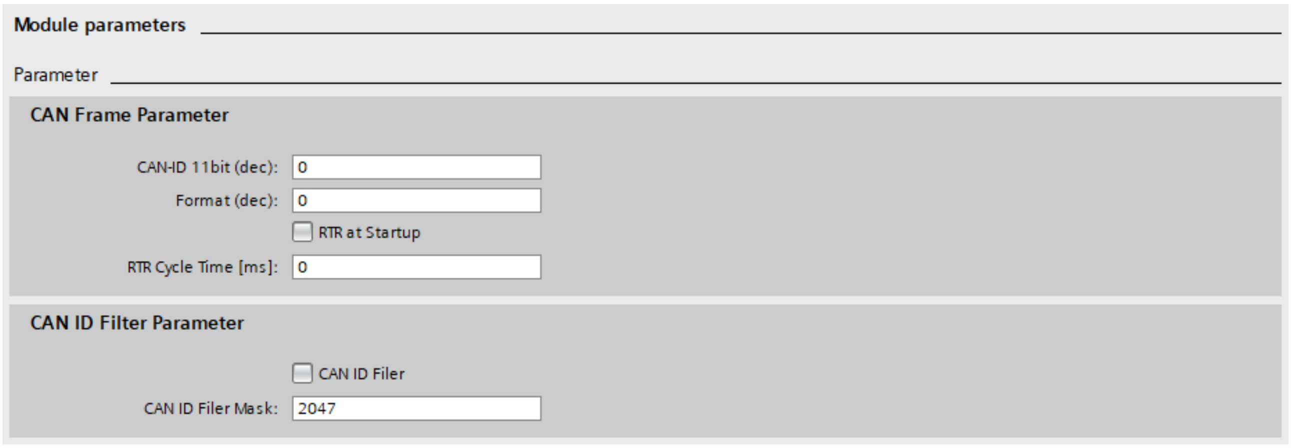
This module category is intended for incoming CAN frames. The module type only has input data depending on the length of the configured CAN frame and no output data. There are modules available for 11-bit and 29-bit CAN identifiers with a length from 1 to 8 byte. Each time a CAN frame with the specified CAN identifier is received, the data is updated.

Additionally, it is possible to select a module with a counter (2 byte) or a timestamp (4 byte).
The counter is incremented each time the configured CAN frame is received. The timestamp is updated each time a new CAN frame is received. Both additional parameters are in big-endian format and are located before the data bytes of the CAN frame. They can be mapped directly to the representative datatype without byte swapping.

Each Input CAN frame is configured with the following parameters:

Parameter	Description
CAN ID 11bit / 29bit	This parameter configures the CAN identifier in a range of 0 to 2047 for 11-bit CAN identifiers and a range of 0 to 536870911 for 29-bit CAN identifiers.
Format	This parameter configures an optional byte-swapping for the data (for details see chapter 4.4.7.4). The parameter has a range of 0 to 255.
RTR at Startup	This parameter defines whether an RTR frame is sent whenever the PLC changes to RUN.
RTR Cycle Time [ms]	This parameter configures a time interval in which a RTR frame is sent (0: Disabled). The supported time interval is between 10ms and 65535ms.
CAN ID Filter	This parameter defines whether a CAN identifier filter mask is applied (for details see chapter 4.4.7.3).
CAN ID Filter Mask	Configures the CAN identifier filter mask in range of 0 to 2047 for 11-bit CAN identifiers and a range of 0 to 536870911 for 29-bit CAN identifiers.

Table 9: Input Parameter


Module parameters

Parameter

CAN Frame Parameter

CAN-ID 11bit (dec): 0

Format (dec): 0

☐ RTR at Startup

RTR Cycle Time [ms]: 0

CAN ID Filter Parameter

☐ CAN ID Filer

CAN ID Filer Mask: 2047

Figure 24: Input Parameter

NOTICE
When the length of the incoming CAN frame differs from the configured length, the frame is still processed. When the configured length exceeds the received length, the unused bytes stay unchanged. When the received length exceeds the configured length, the data is truncated.

NOTICE
It is not allowed to use multiple Input and Output modules with the same CAN identifier in conjunction with cyclic operations.

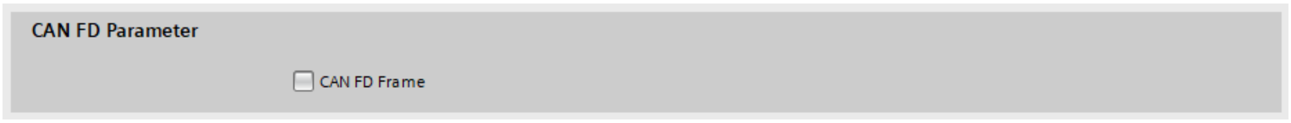
Additional CAN FD Parameters (CAN-PN/2-FD only):

The CAN-PN/2-FD gateway with CAN FD capability also supports CAN frames with the lengths of 12, 16, 20, 24, 32, 48 and 64 bytes. For these modules the range of the parameter *Format* is increased to full 64-bit value.

Parameter	Description
CAN FD Frame	This parameter defines weather the incoming CAN frame is a CAN FD frame. When this function is not enabled, CAN FD frames with the specified CAN identifier are ignored and vice versa.

Table 10: CAN FD Input Parameter

NOTICE
CAN FD frames do NOT support RTR frames. When the incoming CAN frame is configured as CAN FD frame, all RTR parameters are ignored.


CAN FD Parameter

☐ CAN FD Frame

Figure 25: CAN FD Input Parameter

4.4.3 Output (11-bit and 29-bit CAN Identifier)

This module category is intended for outgoing CAN frames. There are modules available for 11-bit and 29-bit CAN identifiers with a length from 1 byte to 8 byte. Each time the data changes, a CAN frame with the specified CAN identifier is sent.

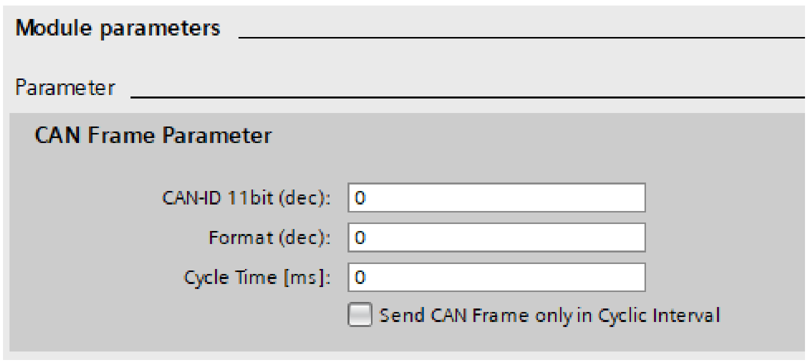
Additionally, there is a controlled version of each module. These modules are not sent when the data changes. Instead, they have an In-Counter and an Out-Counter (1 byte). The Out-Counter is a PLC-Output, and the In-Counter is a PLC-Input which is located before the data bytes of the CAN frame. Whenever the PLC increases the Out-Counter, a CAN frame with the specified CAN identifier is sent and the In-Counter is updated afterwards. This module is intended for CAN frames that are sent with the same data multiple times. To get more information about the In-Counter and Out-Counter see chapter 4.4.7.1.

NOTICE
The In-Counter and Out-Counter do only have a counting range from 0 to 254.
255 is reserved for future use and should not be used.

Each Output CAN frame is configured with the following parameters:

Parameter	Description
CAN ID 11bit / 29bit	This parameter configures the CAN identifier in the range of 0 to 2047 for 11-bit CAN identifiers and the range of 0 to 536870911 for 29-bit CAN identifiers.
Format	This parameter configures an optional byte-swapping for the data (for details see 4.4.7.4). The parameter has a range of 0 to 255.
Cycle Time [ms]	This parameter configures a time interval in which the CAN frame is sent (0: Disabled). The supported time interval is between 10ms and 65535ms.
Send CAN Frame only in Cyclic Interval	This parameter defines whether the CAN frame is only sent in the time interval or also when the data is changed or respectively the Out-Counter is increased.

Table 11: Output Parameter


Module parameters

Parameter

CAN Frame Parameter

CAN-ID 11bit (dec): 0
Format (dec): 0
Cycle Time [ms]: 0
☐ Send CAN Frame only in Cyclic Interval

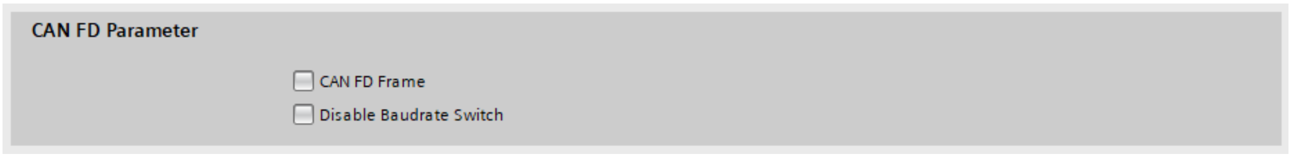
Figure 26: Output Parameter

Additional CAN FD Parameters (CAN-PN/2-FD only):

The CAN-PN/2-FD gateway with CAN FD capability also supports CAN frames with the lengths of 12, 16, 20, 24, 32, 48 and 64 bytes. For these modules the range of the parameter *Format* is increased to full 64-bit value.

Parameter	Description
CAN FD Frame	This parameter defines weather the CAN frame is a CAN FD frame.
Disable Baudrate Switch	By default, each CAN FD frame switches to the configured data-phase bitrate during transmission. This parameter defines whether this function is disabled.

Table 12: CAN FD Output Parameter


CAN FD Parameter

☐ CAN FD Frame

☐ Disable Baudrate Switch

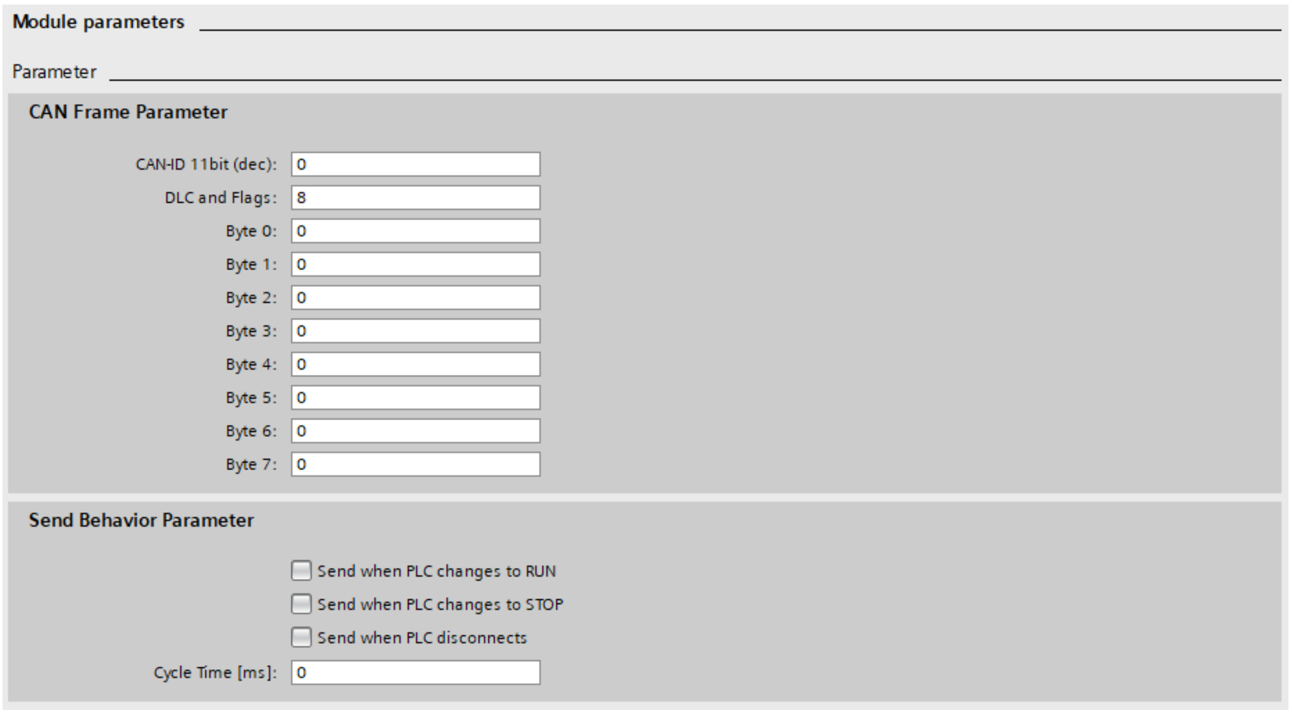
Figure 27: CAN FD Output Parameter

Static Output CAN Frame
This module category contains static outgoing CAN frames which can be used to configure predefined CAN frames and a sending scheduling. Because the CAN data is predefined, the module does not have any IO-Data.

Each Static CAN frame is configured with the following parameters:

Parameter	Description
CAN-ID 11bit / 29bit	This parameter configures the CAN identifier in range of 0 to 2047 for 11-bit CAN identifiers and the range of 0 to 536870911 for 29-bit CAN identifiers.
DLC and Flags	This parameter configures the length as well as additional flags like RTR for the frame (for more detail see chapter 4.4.7.2).
Byte 0...7	These parameters define the CAN data.
Send when PLC changes to RUN	Defines whether the CAN frame is sent when the PLC changes to RUN.
Send when PLC changes to STOP	Defines whether the CAN frame is sent when the PLC changes to STOP.
Send when PLC disconnects	Defines whether the CAN frame is sent when the PLC disconnects.
Cycle Time [ms]	This parameter configures a time interval in which the CAN frame is send (0: Disabled). The supported time interval is between 10 ms and 65535 ms.

Table 13: Static Output Parameter


Module parameters

Parameter

CAN Frame Parameter

CAN-ID 11bit (dec):	0
DLC and Flags:	8
Byte 0:	0
Byte 1:	0
Byte 2:	0
Byte 3:	0
Byte 4:	0
Byte 5:	0
Byte 6:	0
Byte 7:	0

Send Behavior Parameter

☐ Send when PLC changes to RUN
☐ Send when PLC changes to STOP
☐ Send when PLC disconnects
Cycle Time [ms]: 0

Figure 28: Static Output Parameter

Additional CAN FD Parameters (CAN-PN/2-FD only):

The CAN-PN/2-FD gateway with CAN FD capability also supports static outgoing CAN FD frames. For CAN FD frames with a length of up to 8 bytes, the module described above can be used with the corresponding bit in the parameter *DLC and Flags*.
For longer CAN frames with lengths up to 64 bytes, the CAN-PN/2-FD module has to be used.
It defines additional parameters for *Byte 8...Byte 63*.

4.4.4 RX-/TX-FIFO

Both previously defined module categories map one CAN frame to a static PLC address. This module category instead is intended for incoming and outgoing CAN frames with multiple different CAN identifiers. This is ideal for non-time-critical accesses such as writing CANopen SDOs, where a CAN identifier is only used once for the transfer and is ignored afterwards.

Only one RX-FIFO and TX-FIFO each is allowed per configuration. The FIFOs can be configured with 1, 5 or 10 CAN frame exchanges per cycle. Depending on the number of CAN frames the input and output data length may vary. Internally the FIFOs do have a size of 255 CAN frames.

Both FIFOs are controlled by an In-Counter and an Out-Counter (1 byte). The In-Counter is a PLC input, and the Out-Counter is a PLC output. They are located at the first byte of the input and output data. Whenever the PLC increments the Out-Counter the data of the FIFO is processed by the gateway. After the data is processed, the In-Counter is incremented. Setting the Out-Counter to 0xFF clears all FIFO data.

NOTICE
The In-Counter and Out-Counter do only have a counting range from 0 to 254.
255 is reserved for a FIFO reset.

4.4.4.1 CAN Frame Structure

Both FIFO types use the same structure for CAN frames. It consists of 4 bytes for the CAN identifier (big-endian), 1 byte for the *DLC and Flags* field, 1 byte for the length and 8 bytes for the CAN frame data. The *DLC and Flags* field also contains the length of the frame as well as some meta data, however for easy access for incoming frames, the *Length* field can be used for more convenience.

The structure looks like this:

Byte	Content
0	CAN Identifier (Bit 24 … 31 / only 29-bit CAN identifier)
1	CAN Identifier (Bit 16 ... 23 / only 29-bit CAN identifier)
2	CAN Identifier (Bit 8 ... 15)
3	CAN Identifier (Bit 0 ...7)
4	DLC and Flags (see chapter 4.4.7.2)
5	Length (Payload for incoming frames)
6	Data Byte 0
7	Data Byte 1
8	Data Byte 2
9	Data Byte 3
10	Data Byte 4
11	Data Byte 5
12	Data Byte 6
13	Data Byte 7

Table 14: CAN Frame Structure

Additional CAN FD Parameters (CAN-PN/2-FD only):

For CAN FD the CAN frame structure looks similar, but instead of 8 data bytes there are 64 data bytes. To distinguish between CAN CC and CAN FD frames, the *DLC and Flags* parameter should be used.

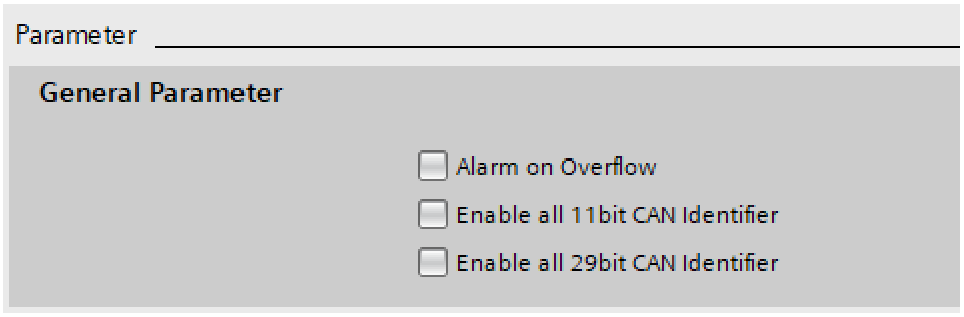
4.4.4.2 RX-FIFO

Whenever the RX-FIFO data is processed, the gateway reads as many frames as possible from the RX FIFO with the configured frames per cycle.
Byte 1 contains the number of CAN frames that are received.
Byte 2 contains the number of CAN frames that are remaining in the RX-FIFO on the gateway.
Byte 3 contains the number of CAN frames missed between two RX-FIFO read cycles.

The following global parameters can be configured:

Parameter	Description
Alarm On Overflow	This parameter defines whether an alarm is sent when a frame is missed.
Enable all 11bit CAN Identifer	This parameter enables all 11-bit CAN identifier.
Enable all 29bit CAN Identifer	This parameter enables all 29-bit CAN identifier.

Table 15: RX-FIFO General Parameter



☐ Alarm on Overflow
☐ Enable all 11bit CAN Identifier
☐ Enable all 29bit CAN Identifier

Figure 29: RX-FIFO General Parameter

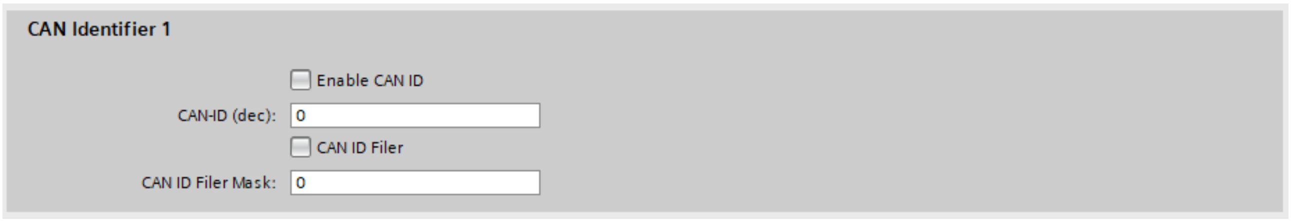
The *General Parameter* only allow to enable all 11-bit and/or 29-bit CAN identifiers. If only specific CAN identifier should be enabled, there is a total of 8 CAN identifiers with CAN ID filter that can be added. Moreover, it is possible to add more CAN identifiers through record (4.6.2.2).

NOTICE
It is not possible to add more than 5000 CAN identifiers to the RX-FIFO. All CAN identifiers that are added after 5000 are ignored. The only exception is when all identifiers of one type are added.

Each of the 8 CAN identifier can be configured as followed:

Parameter	Description
Enable CAN ID	This parameter enables this specific CAN identifier.
CAN ID	This parameter configures the CAN identifier. For 29-bit CAN identifier the 29th bit of the CAN identifier must be set.
CAN ID Filter	Defines whether a CAN identifier filter mask is applied (for details see chapter 4.4.7.3).
CAN ID Filter Mask	This parameter configures the CAN identifier filter mask in the range of 0 to 2047 for 11-bit CAN identifiers and a range of 0 to 536870911 for 29-bit CAN identifiers.

Table 16: RX-FIFO CAN Identifier Parameter



CAN Identifier 1

	☐ Enable CAN ID
CAN-ID (dec):	0
	☐ CAN ID Filer
CAN ID Filer Mask:	0

Figure 30: RX-FIFO CAN Identifier Parameter

The following tables describe the input and output data for CAN CC:

Input Byte	Content
0	In-Counter
1	Number of received frames
2	Number of remaining frames
3	Number of missed frames
4 ... 17	CAN Frame Structure 1 (see chapter 4.4.4.1)
18 ... 31	CAN Frame Structure 2
32 ... 45	CAN Frame Structure 3
46 ... 59	CAN Frame Structure 4
60 ... 73	CAN Frame Structure 5
74 ... 87	CAN Frame Structure 6
88 ... 101	CAN Frame Structure 7
102 ... 115	CAN Frame Structure 8
116 ... 129	CAN Frame Structure 9
130 ... 143	CAN Frame Structure 10

Table 17: RX-FIFO Inputs

Output Byte	Content
0	Out-Counter (0xFF for Reset)

Table 18: RX-FIFO Outputs

Additional CAN FD Parameters (CAN-PN/2-FD only):

For CAN FD the input data are slightly different because CAN FD supports larger CAN frame lengths:

Input Byte	Content
0	In-Counter
1	Number of received frames
2	Number of remaining frames
3	Number of missed frames
4 ... 73	CAN FD Frame Structure 1 (see chapter 4.4.4.1)
74 ... 143	CAN FD Frame Structure 2
144 ... 213	CAN FD Frame Structure 3
214 ... 283	CAN FD Frame Structure 4
284 ... 353	CAN FD Frame Structure 5
354 ... 423	CAN FD Frame Structure 6
424 ... 493	CAN FD Frame Structure 7
494 ... 563	CAN FD Frame Structure 8
564 ... 633	CAN FD Frame Structure 9
634 ... 703	CAN FD Frame Structure 10

Table 19: CAN FD RX-FIFO Inputs

4.4.4.3 TX-FIFO

Whenever the TX-FIFO data is processed, the gateway tries to send the number of frames, which is set in byte 1. CAN frames with an invalid CAN identifier are discarded.
The following parameter can be configured:

Parameter	Description
Alarm On Overflow	This parameter defines whether an alarm is to be sent if the gateway is not able to process the CAN frames within one cycle because the FIFO is full. As long as the FIFO is busy, the In-Counter is not incremented. To unlock a blocked FIFO, set the Out-Counter to 0xFF.

Table 20: TX-FIFO Parameter

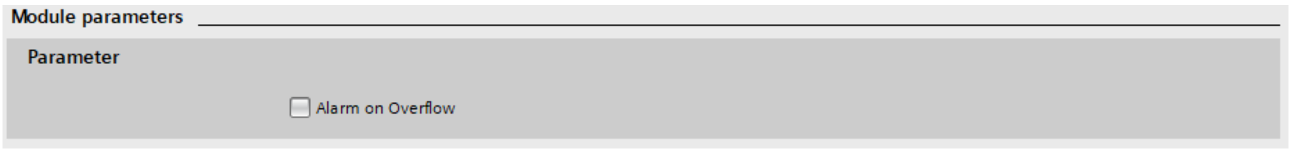

Module parameters
Parameter
☐ Alarm on Overflow

Figure 31: TX-FIFO Parameter

The following tables describe the input and output data:

Input Byte	Content
0	In-Counter

Table 21: TX-FIFO Inputs

Output Byte	Content
0	Out-Counter (0xFF for Reset)
1	Number of frames to be send
2 ... 15	CAN Frame Structure 1 (see chapter 4.4.4.1)
16 ... 29	CAN Frame Structure 2
30 ... 43	CAN Frame Structure 3
44 ... 57	CAN Frame Structure 4
58 ... 71	CAN Frame Structure 5
72 ... 85	CAN Frame Structure 6
86 ... 99	CAN Frame Structure 7
100 ... 113	CAN Frame Structure 8
114 ... 127	CAN Frame Structure 9
128 ... 141	CAN Frame Structure 10

Table 22: TX-FIFO Outputs

Additional CAN FD Parameters (CAN-PN/2-FD only):

For CAN FD the output data are slightly different, because CAN FD supports larger CAN frame lengths:

Output Byte	Content
0	Out-Counter (0xFF for Reset)
1	Number of frames to be send
2 ... 71	CAN FD Frame Structure 1 (see chapter 4.4.4.1)
72 ... 141	CAN FD Frame Structure 2
142 ... 211	CAN FD Frame Structure 3
212 ... 281	CAN FD Frame Structure 4
282 ... 351	CAN FD Frame Structure 5
352 ... 421	CAN FD Frame Structure 6
422 ... 491	CAN FD Frame Structure 7
492 ... 561	CAN FD Frame Structure 8
562 ... 631	CAN FD Frame Structure 9
632 ... 701	CAN FD Frame Structure 10

Table 23: CAN FD TX-FIFO Inputs

4.4.5 Bus Statistic

This module category provides additional data of the CAN bus. Especially if the PLC application does not work as expected, these modules can be used to observe the error.

4.4.5.1 Bus Status

This module provides the current state of the CAN bus in a 1-byte input. It does not need to be configured. The module can have the following values:

Value		Description
Decimal	**Hexadecimal**	
0	0x00	CAN Bus OK
64	0x40	CAN Bus WARN
128	0x80	CAN Bus ERROR PASSIV
192	0xC0	CAN Bus OFF

Table 24: CAN Bus States

NOTICE
This is especially helpful when alarms are disabled. In the CAN interface the alarm threshold can be set to 'Alarm disabled'. By using this module, a simple error routine can be implemented by checking if the bus state has changed.

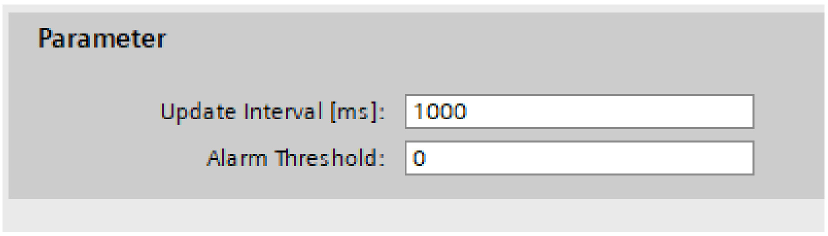
4.4.5.2 Busload

This module provides the current CAN bus load in percentages as a 1-byte input.

The following global parameter can be configured:

Parameter	Description
Update Interval [ms]	This parameter defines in which interval the bus load is calculated.
Alarm Threshold	This parameter defines at which bus load an alarm is sent to the PLC (0: Disabled).

Table 25: Busload Parameter


Parameter

Update Interval [ms]: 1000

Alarm Threshold: 0

Figure 32: Busload Parameter

4.4.5.3 RX-/TX-Counter

These modules provide a counter of the incoming and outgoing CAN frames (4 byte). The data is in the big-endian format. The modules do not need to be configured and have 4-byte input data.

4.4.6 Communication Window

The Communication Window offers a similar functionality as the FIFO modules, but with a lower data exchange rate and a more complex implementation. It was introduced with the predecessor, the CAN-PN (C.2920.02) and is also included in the CAN-PN/2(-FD) for compatibility reasons. It does not have any CAN FD functionality. If possible, use the FIFO modules.

The Communication Window consists of two modules:

- Communication Window Output
- Communication Window Input

Both modules allocate 16 bytes of data. One module as input, the other as output.

NOTICE
The Communication Window does not work if only one of the modules is configured. Both modules are only allowed once per configuration.

The following table describes the input and output data:

Input Byte	Contents
 0 1	As long as no receive data are available ‘0xEEEE’, otherwise: CAN identifier (Bits 10 ... 8) CAN identifier (Bits 7 ... 0)
2 3	For 11-bit CAN Identifier Byte 2 and 3 are always ‘0’ For 29-bit CAN Identifier Byte 2: Identifier bits 28 ... 24 Byte 3: Identifer bits 23 ... 16 **NOTICE** To use a 29-bit identifier, set the 29th bit.
4 5 6 7 8 9 10 11	Data Byte 0 Data Byte 1 Data Byte 2 Data Byte 3 Data Byte 4 Data Byte 5 Data Byte 6 Data Byte 7
12	Number of received data bytes
13	In-Counter (Return value of the Out-Counter which has been transmitted to the gateway via the last PROFINET telegram) (see chapter 4.4.7.1)
14	Number of remaining frames
15	CAN frame received (0x04) or no CAN frame received (0x00)

Table 26: Communication Window Inputs

Output Byte	Content
0 1	CAN identifier (identifier bit 10 ... 8) CAN identifier (identifier bit 7 ... 0)
2 3	For 11-bit CAN identifier Byte 2 and 3 are always ‘0’ For 29-bit CAN identifier Byte 2: Identifier bits 28 ... 24 Byte 3: Identifer bits 23 ... 16 **NOTICE** To use a 29bit identifier, set the 29th bit.
4 5 6 7 8 9 10 11	Data Byte 0 Data Byte 1 Data Byte 2 Data Byte 3 Data Byte 4 Data Byte 5 Data Byte 6 Data Byte 7
12	Data length for transmission jobs (Tx)
13	Out-Counter (has to be incremented in pulse with Organization Block OB1 in order to synchronize the gateway with the OB1 cycle) (see chapter 4.4.7.1)
14	Sub command (always set to ‘0’)
15	Command (defined throughout this chapter)

Table 27: Communication Window Outputs

Commands

The following table shows commands which are currently supported. The sub command is not yet evaluated and should always be set to ‘0’.

Command	Function
0	No Action
1	Transmit a CAN frame
4	Enable CAN identifier for data reception
5	Deactivate CAN identifier for data reception
6	Transmit an RTR frame
7	Execute command 4 and command 6
11	Clear RX-FIFO

Table 28: Communication Window Commands

NOTICE
A command is only processed completely if byte 13 of the module provides the value of the PLC-loop counter which was transferred during the command call.
Before calling the following command, it is therefore advisable to check byte 13 first!

In the following section the commands are described:

Command 0: No Action

No action

Command 1: Transmit a CAN frame

In order to send data via the Communication Window the CAN identifier has to be specified in bytes 0 ... 3.
For 11-bit CAN identifier the bytes 2 and 3 as well as bit 4 to 7 in byte 0 needs to be ‘0’.
Byte 0 → Bits 8 ....15 of the CAN identifier
Byte 1 → Bits 0 ... 7 of the CAN identifier
Byte 2 → Bits 24 ... 31 of the CAN identifier
Byte 3 → Bits 16 … 23 of the CAN identifier

If the 29th bit of the CAN identifier is set, the frame are sent with a 29-bit CAN identifier. The bits 30 and 31 of the identifier should be zero.

The number of bytes to be sent should be specified in byte 12 of the Communication Window. A valid value range is 0 ... 8.

Command 4 as Output: Enable Rx-identifiers for reception

By means of this command the Rx-identifier whose data is to be received has to be enabled. More than one Rx-identifier can be enabled at the same time. For this purpose, the command has to be called a corresponding number of times. For the CAN identifier the same rules apply as for command 1

NOTICE
To enable a 29-bit CAN identifier, the 29th bit must be set.
By enabling a 29-bit CAN identifier, all 29-bit CAN identifiers are enabled.

Command 5: Deactivate reception

After this command has been called no data is received any longer on the specified Rx-identifiers. For the CAN identifier the same rules apply as for command 1.

Command 6: Transmit an RTR frame

By means of this command an RTR frame is transmitted. Prior to the transmission the reception on the CAN identifier has to be enabled by command 4.

Command 7: Execute command 4 and command 6

Combination of command 4 and 6.

Command 11: Clear RX-FIFO

This command clears all CAN frames, stored in the FIFO of the Communication Window. This FIFO contains up to 255 CAN frames.

In-Counter and Out-Counter

The functionality of the counters is described in chapter 4.4.7.1.

4.4.7 Additional Information

4.4.7.1 In-Counter and Out-Counter

To synchronize the data exchange for the controlled outputs (see chapter 4.4.3), the FIFO modules (see chapter 4.4.4) and the Communication Window (see chapter 4.4.6) the gateway uses a mechanism with an In-Counter and an Out-Counter. Both counters have a length of 1 byte.
The In-Counter is a PLC input, and the Out-Counter is a PLC output. The counters are always located in the first input and output byte, except for the Communication Window, where it is located in the byte 13.
The basic concept is that the PLC updates the data of the module first. After this is done, the PLC increments the Out-Counter. This signals the gateway that it should process the data of the module. The PLC must now wait until the In-Counter and the Out-Counter are equal again. It must not change the data as long as this is not the case. During this time the data is processed by the gateway. After the data is processed, the gateway increments the In-Counter to signal the PLC that the current module data have been processed. Now both counters are equal, and the PLC can update the data again and repeat the process. The counting range of the counters is always 0 to 254. The value 255 of the counter is reserved for other functionalities and should not be used as counter.

Example

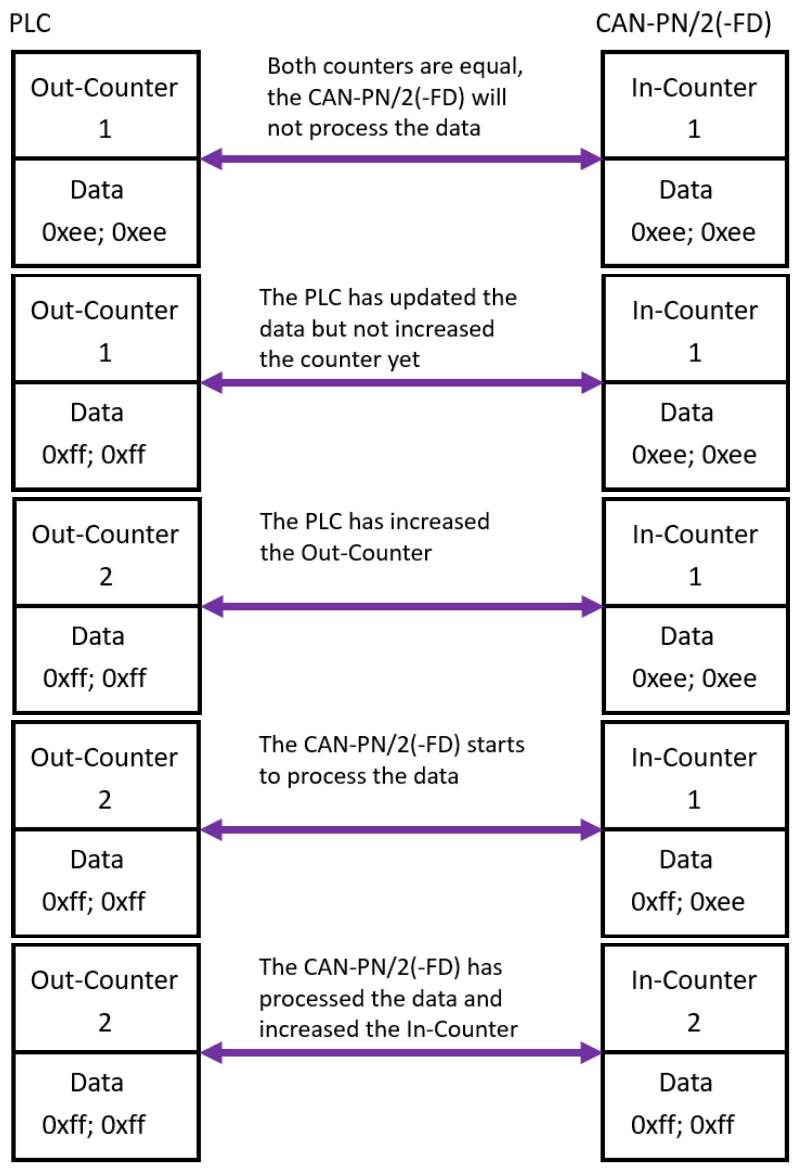


Figure 33: Example In-Counter and Out-Counter

Pseudo Code

Step	**PLC Cycle (Pseudo Code)**
	..
1	Read In-Counter of the module.
2	Check if In-Counter and Out-Counter are equal. If so, continue with step 3, otherwise with step 6.
3	Process the input data of the module (application based).
4	Change the output data of the module (application based).
5	Check if the Out-Counter is less than 254. If this is the case, increment the Out-Counter of the module. Otherwise, set it to ‘0’.
6	Go on with the PLC cycle (next counter comparison in the next PLC cycle)
	..

Table 29: In-Counter and Out-Counter Pseudo Code

4.4.7.2 DLC and Flags

Some of the function use a 1-byte parameter called *DLC and Flags*. This parameter consists of a 4-bit *Data Length Code* (DLC) parameter as well as 4 bits of meta information.

Bit	7	6	5	4	3	2	1	0
Function	Meta information (Flags)				DLC			

Table 30: DLC and Flags

The DLC code is used to decode the payload respectively data length of a CAN frame. This differs for CAN CC and CAN FD.

CAN CC:

DLC	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Frame Length	0	1	2	3	4	5	6	7	8	8	8	8	8	8	8	8

Table 31: DLC CAN CC

CAN FD (CAN-PN/2-FD only):

DLC	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Frame Length	0	1	2	3	4	5	6	7	8	12	16	20	24	32	48	64

Table 32: DLC CAN FD

NOTICE
CAN RTR frames are only defined for CAN CC frames and always have a frame length of 0 independent of the DLC value.

The 4-bit meta information are defined according to the following table.

Bit	Value	
	0	1
4	CAN CC: ➔ Data frame CAN FD: ➔ Data frame with baud rate switch	CAN CC: ➔ RTR frame CAN FD: ➔ Data frame without baud rate switch
5	Reserved	Reserved
6	Reserved	Reserved
7	CAN CC Frame	CAN FD Frame

Table 33: CAN Frame Flags

4.4.7.3 CAN ID Filter

CAN ID filter act as masks on the CAN identifier and can be used to ignore specific bits of the CAN identifier. This is useful for CAN protocols like J1939, where certain bits of the CAN identifier can change depending on the CAN network. With the ID filter, only the bits of the CAN identifier are evaluated where the filter mask is ‘1’. All other bits are then ignored.

Example

The following example with a CAN identifier of 112 and a CAN ID Filter of 2032 should make this more comprehensible:

Parameter	Decimal	Hexadecimal	Binary
CAN Identifer	112	0x07F	0b000001110000
CAN ID Filter	2032	0x7F0	0b011111110000

Table 34: CAN ID Filter

Bit	10	9	8	7	6	5	4	3	2	1	0
CAN Identifer	0	0	0	0	1	1	1	0	0	0	0
CAN ID Filter	1	1	1	1	1	1	1	0	0	0	0
Result	0	0	0	0	1	1	1	X*	X*	X*	X*

* X could be 0 or 1

Table 35: CAN ID Filter Example

The resulting CAN Identifier is 0x07X, where X stands for do not care. That means that for example the CAN identifier 0x071 as well as 0x07F would be applied to this frame.

4.4.7.4 Format

The module types of Input and Output both have the parameter *Format*.
It is used to convert the user data from big -endian (high byte first / Motorola format) into little endian (low byte first / Intel format) to CAN data and vice versa. CAN frames which are longer than one byte, are transmitted on a CAN network in little-endian, while the Siemens PLC operates in big-endian.

Starting with bit 7 of the *Format* byte you can decide whether the following byte is swapped or not. If a ‘1’ is specified for a byte, the following bytes are converted until the next ‘0’ is transmitted. The functionality can be explained best by means of an example.

Example

A CAN telegram has got a date in little endian in the first two bytes, followed by two bytes which are not to be swapped and a long word in the last four bytes which is in little endian again.
Binary the following description for the format byte:

Bit	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Bit of *Format*	1	0	0	0	1	1	1	0
Hexadecimal	0x8				0xE			
Decimal	142							
Action	begin swap	end swap	un-changed	un-changed	begin swap	swap	swap	end swap

Table 36: CAN Format Parameter 1

Data Bytes	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
CAN Frame	2 bytes little-endian		byte 3	byte 4	4 bytes little-endian			
PLC Data	2 bytes big-endian		byte 3	byte 4	4 bytes big-endian			

Table 37: CAN Format Parameter 2

From this the *Format* byte results in 0x8E → 142.
If all eight bytes are to be swapped, for instance, value 0xFE → 254 is specified for the *Format* byte. The lowest bit is generally without significance because the telegram and therefore the formatting have been completed. The bit should always be set to '0'.

NOTICE
The parameter *Format* must be set to ‘0’ if byte swapping is unwanted.

Additional CAN FD Parameters (CAN-PN/2-FD only):

Because CAN FD frames can have a maximum length of 64 byte, a single bit of *Format* is not big enough. Therefore, the parameter is extended to 64bit for such frames. But the functionality is the same.

4.5 Diagnostics

The gateway uses alarms to inform the PROFINET controller about errors on the gateway. The alarms are mostly optional, and the time of appearance can be configured. Whenever this symbol 🟥 is displayed in the *Device overview*, an alarm is pending on the specific module.
Further information about the alarm is described in the diagnostics of the module (Context Menu of *Module* -> *Online & diagnostics* -> *Diagnostics status*.

The following modules can cause alarms:

Module	**Error Type**	**Reason**
CAN Interface (always slot 0)	Line Break	The CAN bus is faulty. The CAN bus state changes to a state worse than configured in the parameter *Alarm Level*. This alarm disappears when the CAN bus state is not faulty anymore. Because the way CAN works, there need to be frames received and send in order to achieve this.
Bus Load	General Error	This alarm appears when the CAN bus load exceeds the % configured in the parameter *Alarm Threshold*. This alarm disappears when the CAN bus load is lower than the configured threshold.
RX-FIFO	General Error	There is an overflow in the RX-FIFO. This alarm occurs when the RX-FIFO lost CAN frames because the internal buffer is full.
TX-FIFO	General Error	Not all data passed by the PLC could be processed in one cycle, because the internal TX buffer is full. This is normally not a problem, because the application tries to send the data with the next cycle.

Table 38: CAN-PN/2 Alarms

The content of the alarm telegram is structured as shown in the table below. However, the PLC will parse the telegram and only show the error type and the address information.

Octet number	Name	Content	
0	reserved	0x00	
1	btype	0x01 (Blocktype:Alarm Notification High)	
2	blen0	0x00	(Blocklength: 30 Byte)
3	blen1	0x1e	
4	verh(1)	0x01 (VersionHigh: 1)	
5	verl(0)	0x00 (VersionLow: 0)	
6	atyp0	0x00	(AlarmType: Diagnosis)
7	atyp1	0x01	
8	ap0	0x00	(API:0)
9	ap1	0x00	
10	ap2	0x00	
11	ap3	0x00	
12	sl0	0x00	SlotNumber
13	sl1	0x00	
14	ssl0	0x00	SubslotNumber
15	ssl1	0x01	
16	mid0	0x10	ModuleID
17	mid1	0x00	
18	mid2	0x00	
19	mid3	0x00	
20	smid0	0x10	SubModuleID
21	smid1	0x00	
22	smid2	0x00	
23	smid3	0x00	
24	a0dmcsss	0xA8	(AR Problem Indicator:1 Diagnosis Exist:1 Manufacturer Specific:0 Channel specific: 1 Sequence number: 0...2047
25	ssssssss	0x00	
26	usi0	0x80	(Use Structure ID: 0x8000 = Channel related diagnosis)
27	usi1	0x00	
28	chn0	0x80	(Channel number: 0x8000)
29	chn1	0x00	
30	dddssmra	0x68 (Direction: 3 = I/O Specifier: 1 = Appears MaintenanceDemanded:0 MaintenanceRequired:0 Accumulative:0)	
31	type	0x00 Type: 0=unspecified	
32	cet0	0x00	(ChannelErrorType: 0x06 = Line Break 0x09 = General Error)
33	cet1	0x06	

After an alarm has been resolved the gateway sends the following alarm telegram. Only the different entries are displayed.

Octet number	Name	Content	
:	:	:	
6	atyp0	0x00	(AlarmType: Diagnosis disappears)
7	atyp1	0x0C	
:	:	:	
24	a0dmcsss	0x00	(AR Problem Indicator:0 Diagnosis Exist:0 Manufacturer Specific:0 Channel specific: 0 Sequence number: 0..2047 (here 1, incremented by one)
25	ssssssss	0x01	
:	:	:	
30	dddssmra	0x10 (Direction: 0 = unspecified Specifier: 2 = Disappears MaintenanceDemanded:0 MaintenanceRequired:0 Accumulative:0	
:	:	:	
32	cet0	0x00	(ChannelErrorType = 0x00 = no error)

4.6 Records

PROFINET records are asynchronous operations that can be used to exchange noncyclic data between the PLC and the gateway. Read records receive data from the gateway while write records send data to the gateway. Therefore, read records need an input buffer in the PLC, in which the gateway can store the data.

The TIA Portal already has preconfigured function blocks called `RDREC` for read records and `WRREC` for write records. The function blocks can be configured and implemented easily. For further information, see chapter 4.6.3 and read the respective documentation of the TIA Portal.

Records are differentiated by their record index. Moreover, the application must clarify how many bytes can be read maximum for read records and how many bytes should be sent for write records. Throughout this chapter the data from a read record is referred to as input data (Gateway → PLC). The data of a write record is referred to as output data (PLC → Gateway). All records are documented in detail throughout this chapter.

To perform some of the CAN operations, the records use an independent CAN handle, which is referred to as *Record Handle*. This means, that for example enabling all CAN identifiers for the RX-FIFO modules does not mean, that all CAN identifiers are enabled for the 'Read CAN Frame' record (0x300).

The following overview shows all supported **read** records.

Index	Record Length [Bytes]	Description	Page
0x30	32	Read CAN statistic	62
0x31	32	Read and clear CAN statistic	62
0x300	1	Read number of remaining frames in the record handle	62
0x301	17	Read CAN frame from the record handle	63
0x302	17 ... 524*	Read multiple CAN frames from the record handle	64
0x303	73	Read CAN FD frame (only supported on the CAN FD variant of the gateway)	65
0x304	73 ... 2764*	Read multiple CAN FD frame (only supported on the CAN FD variant of the gateway)	66
0x305	4	Read current timestamp of the gateway	66

* Record length depends on the number of CAN frames

Table 39: CAN-PN/2 Read Records

The following overview shows all supported **write** records.

Index	Record Length [Bytes]	Description	Page
0x10	-	Reset CAN Statistic	67
0x20	1 ... 9**	Enable CAN identifier for the RX-FIFO module	68
0x21	1 ... 9**	Disable CAN identifier for the RX-FIFO module	69
0x101	13	Transmit CAN frame	70
0x102	14 ... 521*	Transmit multiple CAN frames	70
0x103	69	Transmit CAN FD frame (only supported on the CAN FD variant of the gateway)	71
0x104	70 ... 2761*	Transmit multiple CAN FD frame (only supported on the CAN FD variant of the gateway)	72
0x107	1 ... 9**	Enable CAN identifier for the record handle	72
0x108	1 ... 9**	Disable CAN identifier for the record handle	72
0x109	-	Clear all CAN frames received with the record handle	72

* Record length depends on the number of CAN frames
** Record length depends on the record mode

Table 40: CAN-PN/2 Write Records

4.6.1 Read Records

4.6.1.1 Read CAN Statistic (0x30 / 0x31)

This read record can be used to get additional information about the CAN bus. All return values are in big endian. The record index 0x30 only reads the statistic while 0x31 reads the statistic and resets the statistic afterwards.

The following data will be returned:

Read Record Read CAN Statistic (Index 0x30/0x31 | Record Length 32 Bytes)				
Parameter	**Byte**	**Description**	**Value Range**	**Data Type**
1	0 ... 3	Number of CAN frames that have been received	-	unsigned32
2	4 ... 7	Number of CAN RTR frames that have been received	-	unsigned32
3	8 ... 11	Number of CAN FD frames that have been received	-	unsigned32
4	12 ... 15	Number of CAN frames that have been send	-	unsigned32
5	16 ... 19	Number of CAN RTR frames that have been send	-	unsigned32
6	20 ... 23	Number of CAN FD frames that have been send	-	unsigned32
7	24 ... 27	Number of CAN controller overruns	-	unsigned32
8	28 ... 31	Number of CAN error frames	-	unsigned32

Table 41: Read Record Read CAN Statistic (0x30/0x31)

NOTICE
These values are not reset when the connection to the PROFINET controller is lost. Only a restart or a reset will clear them.

4.6.1.2 Read Number of Remaining Frames in the Record Handle (0x300)

This read record returns the number of CAN frames that are stored in the record handle RX-FIFO.

Read Record Read Number of Remaining Frames in the Record Handle (Index 0x300 | Record Length 1 Byte)				
Parameter	**Byte**	**Description**	**Value Range**	**Data Type**
1	0	Number of remaining frames in the record handle	0x00 ... 0xFF	unsigned8

Table 42: Read Record Read Number of Remaining Frames in the Record Handle (0x300)

4.6.1.3 Read CAN Frame from the Record Handle (0x301)

This read record reads one CAN frame from the record handle.

The required buffer must be at least 17 bytes in size. It uses the same CAN frame structure as described in chapter 4.4.4.1.

Read Record Read CAN Frame from the Record Handle (Index 0x301 | Record Length 17 Bytes)				
Parameter	**Byte**	**Description**	**Value Range**	**Data Type**
1	0	Reserved for future use	-	-
2	1	Number of received CAN frames in this record. When this is ‘0’, no CAN frame was received	0x00 ... 0xFF	Unsigned8
3	2	Number of remaining CAN frames which are stored in the RX-FIFO of the record handle	0x00 ... 0xFF	Unsigned8
4	3	Number of missed CAN frames	0x00 ... 0xFF	Unsigned8
5	4 ... 16	CAN Frame Structure 1 (see chapter 4.4.4.1)	-	CAN Frame Structure

Table 43: Read Record Read CAN Frame from the Record Handle (0x301)

NOTICE
All required CAN Identifiers need be enabled with the record 0x107 (see chapter 4.6.2.8).

4.6.1.4 Read multiple CAN Frames from the Record Handle (0x302)

This read record reads up to 40 CAN frames from the record handle. The number of CAN frames that can be read per records depends on buffer size.

The minimum required buffer size is 17 bytes, which can return one CAN frame, and the maximum buffer size is 524 bytes, which can return up to 40 CAN frames. Like the record 0x302 it uses the same CAN frame structure as described in chapter 4.4.4.1..

Read Record Read multiple CAN Frames from the Record Handle (Index 0x302 | Record Length 17 - 524 Bytes)				
Parameter	**Byte**	**Description**	**Value Range**	**Data Type**
1	0	Reserved for future use	-	-
2	1	Number of received CAN frames in this record. When this is ‘0’, no CAN frame was received	0x00 ... 0xFF	Unsigned8
3	2	Number of remaining CAN frames which are stored in the RX-FIFO of the record handle	0x00 ... 0xFF	Unsigned8
4	3	Number of missed CAN frames	0x00 ... 0xFF	Unsigned8
5	4 ... 16	CAN Frame Structure 1 (see chapter 4.4.4.1)	-	CAN Frame Structure
5	17 ... 510	CAN Frame Structure 2 - 39	-	CAN Frame Structure
6	511 ... 523	CAN Frame Structure 40	-	CAN Frame Structure

Table 44: Read multiple CAN Frames from the Record Handle (0x302)

NOTICE
All required CAN identifiers need to be enabled with the record 0x107 (see chapter 4.6.2.8).

4.6.1.5 Read CAN FD Frame from the Record Handle (0x303)

NOTICE
This record is only supported by CAN-PN/2-FD.

This read record reads one CAN frame from the record handle. The record is similar to 0x301 but has an increased buffer for CAN FD frames. It can also read non-FD CAN frames. To distinguish between CAN CC and CAN FD frames, check the *DLC and Flags* field (see chapter 4.4.7.2).

The required buffer must be at least 73 bytes in size. It uses the same CAN FD frame structure as described in chapter 4.4.4.1.

Read Record Read CAN FD Frame from the Record Handle (Index 0x303 | Record Length 73 Bytes)				
Parameter	**Byte**	**Description**	**Value Range**	**Data Type**
1	0	Reserved for future use	-	-
2	1	Number of received CAN frames in this record. When this is '0', no CAN frame was received	0x00 ... 0xFF	Unsigned8
3	2	Number of remaining CAN frames which are stored in the RX-FIFO of the record handle	0x00 ... 0xFF	Unsigned8
4	3	Number of missed CAN frames	0x00 ... 0xFF	Unsigned8
5	4 ... 72	CAN FD Frame Structure 1 (see chapter 4.4.4.1)	-	CAN FD Frame Structure

Table 45: Read Record Read CAN FD Frame from the Record Handle (0x303)

NOTICE
All required CAN identifiers need to be enabled with the record 0x107 (see chapter 4.6.2.8).

4.6.1.6 Read multiple CAN FD Frames from the Record Handle (0x304)

NOTICE
This record is only supported by CAN-PN/2-FD.

This read record reads up to 40 CAN FD frames from the record handle. The record is similar to 0x302 but has an increased buffer for CAN FD frames. The number of CAN frames that can be read per records depends on the buffer size.

The minimum required buffer size is 73 bytes, which can return one CAN FD frame, and the maximum buffer size is 2764 bytes, which can return up to 40 CAN FD frames.
Like the record 0x303 it uses the same CAN FD frame structure as described in chapter 4.4.4.1.

Read Record Read multiple CAN FD Frames from the Record Handle (Index 0x304 | Record Length 17 - 524 Bytes)				
Parameter	**Byte**	**Description**	**Value Range**	**Data Type**
1	0	Reserved for future use	-	-
2	1	Number of received CAN frames in this record. When this is ‘0’, no CAN frame was received	0x00 … 0xFF	Unsigned8
3	2	Number of remaining CAN frames which are stored in the RX-FIFO of the record handle	0x00 ... 0xFF	Unsigned8
4	3	Number of missed CAN frames	0x00 ... 0xFF	Unsigned8
5	4..72	CAN FD Frame Structure 1 (see chapter 4.4.4.1)	-	CAN FD Frame Structure
5	73 … 2694	CAN FD Frame Structure 2 - 39	-	CAN FD Frame Structure
6	2695 ... 2763	CAN FD Frame Structure 40	-	CAN FD Frame Structure

Table 46: Read Record Read multiple CAN FD Frames from the Record Handle (0x304)

4.6.1.7 Read current timestamp of the gateway (0x300)

This read record returns the number of CAN frames that are stored in the record handle RX-FIFO.

Read Record Read Number of Remaining Frames in the Record Handle (Index 0x300 | Record Length 4 Byte)				
Parameter	**Byte**	**Description**	**Value Range**	**Data Type**
1	0-3	Timestamp of the gateway. Resolution is defined in the CAN interface (see 4.4.1)	0x00 ... 0xFFFFFFFF	Unsigned32

4.6.2 Write Records

4.6.2.1 Reset CAN Statistic (0x10)

This write record resets the CAN statistic. It does not need any input data.

4.6.2.2 Add CAN Identifier to RX-FIFO (0x20)

This record can be used to add additional CAN identifier to the RX-FIFO module. Depended on the data it needs output data of up to 9 bytes.

Write Record Add CAN Identifier to RX-FIFO (Index 0x20 | Record Length 1-9 Bytes)				
Parameter	**Byte**	**Description**	**Value Range**	**Data Type**
1	0	Mode **Function** 0: Add one CAN Identifier (Minimum Input data 5 byte) 1: Add all 11-bit CAN Identifier (Minimum Input data 1 byte) 2: Add all 29-bit CAN Identifier (Minimum Input data 1 byte) 3: Add CAN identifier with Mask (Minimum Input data 9 byte) 4: Add CAN Identifier Region (Minimum Input data 9 byte)	0x00 … 0x04	Unsigned8
2	1 ... 4	**Function** 0: CAN Identifier (for 29-bit Identifier, set the 29th bit) 1: - 2: - 3: CAN Identifier (for 29-bit Identifier, set the 29th bit) 4: First CAN Identifier (for 29bit Identifier, set the 29th bit)	-	Unsigned32
3	5 ... 9	**Function** 0: - 1: - 2: - 3: CAN ID Filter (see chapter 4.4.7.3) 4: Number of CAN Identifiers that should be added, started from the first CAN Identifer (Byte 1-4).	-	Unsigned32

Table 47: Write Record Add CAN Identifier to RX-FIFO (0x20)

NOTICE
It is not possible to add more than 5000 CAN identifiers to the RX-FIFO module. All CAN identifiers that have been added after 5000 will be ignored. The only exception is when all identifiers of one type are added or when a range of CAN identifiers is added.

4.6.2.3 Delete CAN Identifier from RX-FIFO (0x21)

This record can be used to delete CAN identifier from the RX-FIFO module. Depended on the data it needs output data of up to 9 bytes.

Write Record Delete CAN Identifier to RX-FIFO (Index 0x21 | Record Length 1-9 Bytes)				
Parameter	**Byte**	**Description**	**Value Range**	**Data Type**
1	0	Mode **Function** 0: Delete one CAN Identifier (Minimum Input data 5 byte) 1: Delete all 11-bit CAN Identifier (Minimum Input data 1 byte) 2: Delete all 29-bit CAN Identifier (Minimum Input data 1 byte) 3: Delete CAN Identifier with Mask (Minimum Input data 9 byte) 4: Delete CAN Identifier Region (Minimum Input data 9 byte)	0x00 …0x04	Unsigned8
2	1 ... 4	**Function** 0: CAN Identifier (for 29-bit Identifier, set the 29th bit) 1: - 2: - 3: CAN Identifier (for 29-bit Identifier, set the 29th bit) 4: First CAN Identifier (for 29-bit Identifier, set the 29th bit)	-	Unsigned32
3	5 ... 9	**Function** 0: - 1: - 2: - 3: CAN ID Filter (see chapter 4.4.7.3) 4: Number of CAN Identifiers that should be added, started from the first CAN Identifer (Byte 1-4).	-	Unsigned32

Table 48: Write Record Delete CAN Identifier to RX-FIFO (0x21)

4.6.2.4 Send CAN Frame (0x101)

This write record can be used to send one CAN frame.

The required output data must be at least 14 bytes. It uses the same CAN frame structure as described in chapter 4.4.4.1.

Write Record Send CAN Frame (Index 0x101 | Record Length 14 Bytes)				
Parameter	**Byte**	**Description**	**Value Range**	**Data Type**
1	0 ... 13	CAN Frame Structure 1 (see chapter 4.4.4.1)	-	CAN Frame Structure

Table 49: Write Record Send CAN Frame (0x101)

4.6.2.5 Send multiple CAN Frames (0x102)

This write record can be used to send up to 40 CAN frames. The number of CAN frames that can be send depends on the number of output bytes as well as the number of frames defined in byte 0 of the output data.

The minimum output data size is 15 bytes, which can send one CAN frame, and the maximum buffer size is 560 bytes, which can send up to 40 CAN frames. Like the record 0x101 it uses the same CAN frame structure as described in chapter 4.4.4.1. The length of the outgoing CAN frames is determined by the *DLC and Flags* field in the CAN frame structure and not by the *Length* field, because this allows additional functionality like RTR frames.

Write Record Send multiple CAN Frames (Index 0x102 | Record Length 15 – 561 Bytes)				
Parameter	**Byte**	**Description**	**Value Range**	**Data Type**
1	0	Number of CAN frames to be send	0x00 … 0xFF	Unsigned8
2	1..14	CAN Frame Structure 1 (see chapter 4.4.4.1)	-	CAN Frame Structure
3	15 ... 546	CAN Frame Structure 2-39	-	CAN Frame Structure
4	547..560	CAN Frame Structure 40	-	CAN Frame Structure

Table 50: Write Record Send multiple CAN Frame (0x101)

4.6.2.6 Send CAN FD Frame (0x103)

NOTICE
This record is only supported by CAN-PN/2-FD.

This write record can be used to send one CAN FD frame. The record is similar to 0x101 but has an increased buffer for CAN FD frames. It can also write non-FD CAN frames. To distinguish between CAN CC and CAN FD frames, set the respective bit in the *DLC and Flags* field (see chapter 4.4.7.2).

The required output data must be at least 70 bytes. It uses the same CAN FD frame structure as described in chapter 4.4.4.1.

Write Record Send CAN FD Frame (Index 0x103 | Record Length 70 Bytes)				
Parameter	**Byte**	**Description**	**Value Range**	**Data Type**
1	0 ... 69	CAN Frame Structure 1 (see chapter 4.4.4.1)	-	CAN FD Frame Structure

Table 51: Write Record Send CAN FD Frame (0x103)

4.6.2.7 Send multiple CAN FD Frames (0x104)

NOTICE
This record is only supported by CAN-PN/2-FD.

This write record can be used to send up to 40 CAN FD frames. The number of CAN FD frames that can be send depends on the number of output bytes as well as the number of frames defined in byte 0 of the output data. The record is similar to 0x102 but has an increased buffer for CAN FD frames. It can also write non-FD CAN frames. To distinguish between CAN CC and CAN FD frames, set the according bit in the *DLC and Flags* field (see chapter 4.4.7.2).

The minimum output data size is 71 bytes, which can send one CAN FD frame, and the maximum buffer size is 2761 bytes, which can send up to 40 CAN FD frames. Like the record 0x103 it uses the same CAN FD frame structure as described in chapter 4.4.4.1.

Write Record Send multiple CAN FD Frames (Index 0x104 | Record Length 70 – 2801 Bytes)				
Parameter	**Byte**	**Description**	**Value Range**	**Data Type**
1	0	Number of CAN FD frames to be send	0x00 … 0xFF	Unsigned8
2	1 ... 70	CAN FD Frame Structure 1 (see chapter 4.4.4.1)	-	CAN FD Frame Structure
3	71 ... 2730	CAN FD Frame Structure 2-39	-	CAN FD Frame Structure
4	2731 ... 2800	CAN FD Frame Structure 40	-	CAN FD Frame Structure

Table 52: Write Record Send multiple CAN FD Frame (0x104)

4.6.2.8 Add CAN Identifier to Record Handle (0x107)

This record can be used to add CAN identifier to the record handle. Depending on the data it needs output data of up to 9 bytes. It uses the same data structure as defined in record 0x20 (see chapter 4.6.2.2).

NOTICE
It is not possible to add more than 5000 CAN identifiers to the record handle. All CAN identifiers that have been added after 5000 will be ignored. The only exception is when all identifiers of one type are added or when a range of CAN identifiers is added.

4.6.2.9 Delete CAN Identifier from Record Handle (0x108)

This record can be used to delete CAN identifiers from the record handle. Depended on the data it needs output data of up to 9 bytes. It uses the same data structure as defined in record 0x21 (see chapter 4.6.2.3).

4.6.2.10 Reset Record Handle RX-FIFO (0x109)

This write record will discard all CAN frames, that are pending in the RX-FIFO of the record handle.

4.6.3 PLC Function Blocks

4.6.3.1 Read Records

The function block **RDREC** is used for reading read records asynchronously.

The following parameter needs to be provided:

```
REQ    := BOOL    (Input)
ID     := HW_IO   (Input)
INDEX  := DINT    (Input)
MLEN   := UINT    (Input)
VALID  := BOOL    (Output)
BUSY   := BOOL    (Output)
ERROR  := BOOL    (Output)
STATUS:= DWORD   (Output)
LEN    := UINT    (Output)
RECORD:= VARIANT (I/O)
```

A data block instance of the function block needs to be added. The data block is automatically generated when the function block is called.

Parameter	Description
REQ	Start read operation (always 1)
ID	HW identifier of a module
Index	Record index
MLEN	Minimum length of the bytes to be read. The actually received length of the data is returned in **LEN**.
VALID	Read operation was successful
BUSY	Read operation is in progress
ERROR	Error occurred during read operation, see parameter STATUS for a further information
STATUS	Error description
LEN	Number of received bytes
RECORD	Received data from the gateway. The received length is specified with the parameter **LEN**.

Table 53: Read Record PLC Parameter

NOTICE
The parameter **REQ** is not edge-triggered. If the input is not reset accordingly, the operation will be repeated permanently.

4.6.3.2 Write Records

The function block **`WRREC`** is used for writing write records asynchronously.

The following parameter needs to be provided:

```
REQ    := BOOL    (Input)
ID     := HW_IO   (Input)
INDEX  := DINT    (Input)
LEN    := UINT    (Input)
DONE    := BOOL   (Output)
BUSY   := BOOL    (Output)
ERROR  := BOOL    (Output)
STATUS:= DWORD    (Output)
RECORD:= VARIANT (I/O)
```

A data block instance of the function block needs to be added. The data block is automatically generated when the function block is called.

Parameter	Description
`REQ`	Start write operation (always 1)
`ID`	HW identifier of a module
`Index`	Record index
`LEN`	Length of the bytes to be transferred
`DONE`	Write operation done successfully
`BUSY`	Write operation is in progress
`ERROR`	Error occurred during write operation, see parameter **`STATUS`** for a further information
`STATUS`	Error description
`RECORD`	Data to be transmitted

Table 54: Write Record PLC Parameter

NOTICE
The parameter **`REQ`** is not edge-triggered. If the input is not reset accordingly, the operation will be repeated permanently.

5 Firmware Update

The CAN-PN/2 gateway offers the possibility of firmware updates.

To install a firmware update, the following steps need to be done:

Step	Action
1	Install the installer provided with the CAN-PN/2 (see chapter 4.2) with all packages.
1	Connect the CAN-PN/2 via Mini-USB with a Windows computer.
3	Make sure that CAN-PN/2 is detected correctly, and a network adapter called ‘RNDIS based ESD Device’ shows up in the *Device Manager* (see chapter 4.2).
3	Execute the batch file `update_X_X_X.bat` in the package provided from the esd support.
4	Wait till the gateway restarts (see chapter 3).
5	The update is done.

Table 55: Firmware Update

6 CAN Monitoring

The CAN-PN/2 gateway provides access to the connected CAN bus via the Mini-USB Type B interface. CAN frames that are present on the bus are received via this interface, including those transmitted by the gateway itself as part of its configured operation. This ensures a complete view of the ongoing CAN communication at the physical bus level.

In addition to receiving data, the interface also allows user-defined CAN frames to be transmitted directly onto the CAN bus. These frames are actively injected and are handled by the transceiver in the same way as messages from any other CAN node. The transmission and reception via this interface are independent of the PROFINET IO communication or gateway configuration, making it a valuable tool for diagnostics, commissioning, and application development.

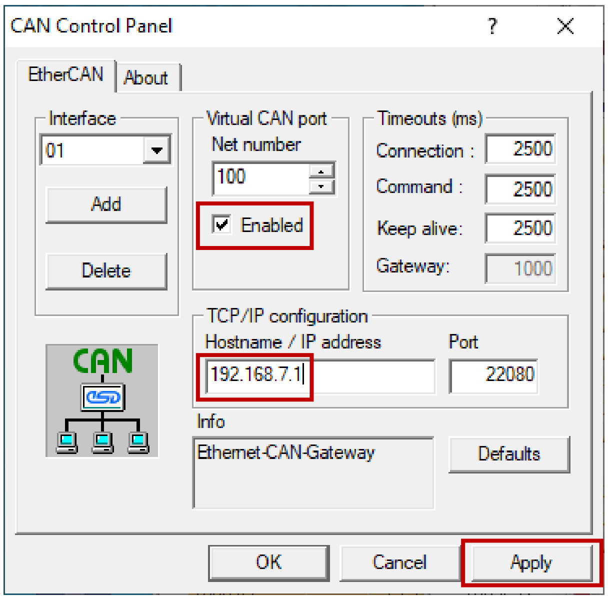
Access to the CAN bus is provided through the EtherCAN protocol, which is compatible with the esd NTCAN API. This allows the use of standard esd tools such as CANreal, CANplot, and CANrepro for monitoring, recording, and replaying CAN traffic. The Mini-USB interface is intended primarily for service and development purposes and does may interfere with the real-time behavior of the gateway.

NOTICE
CAN-PN/2-FD: Works with CAN CC aswell as CAN FD with firmware version V3.0.5 or newer.

To configure the CAN monitoring, the following steps need to be done:

Step	Action
1	An installer is provided with the CAN-PN/2 Install it with all packages (see chapter 4.2).
2	Connect the CAN-PN/2 via Mini-USB with a Windows computer.
3	Make sure that CAN-PN/2 is detected correctly, and a network adapter called ‘RNDIS based ESD Device’ shows up in the *Device Manager* (see chapter 4.2).
4	Start the program *CAN Control Panel* which is installed with the installer. By default, the net number is already set to 100. Set the parameter *Hostname / IP address* to: **192.168.7.1** Make sure that the checkbox *Enabled* is checked. Do not change any other settings (see Figure 34). Press *Apply*.
5	Open the esd tool *CANreal.* Select the net number 100 in the dropdown menu of the input field *Net* on the top. It is not needed to set the baud rate manually because the PROFINET controller already configured it. However, when the gateway is not connected to a PROFINET network, this parameter can also be set manually. Let all other values unchanged. Press *Start*.
6	Now *CANreal* can interact with the CAN interface on the gateway. Some basic functionalities are described in Figure 35. For further information see the CANreal manual (*CANreal* main menu -> *Help* -> *CANreal help*).

Table 56: CAN Monitoring



Interface: 01

Virtual CAN port – Net number: 100

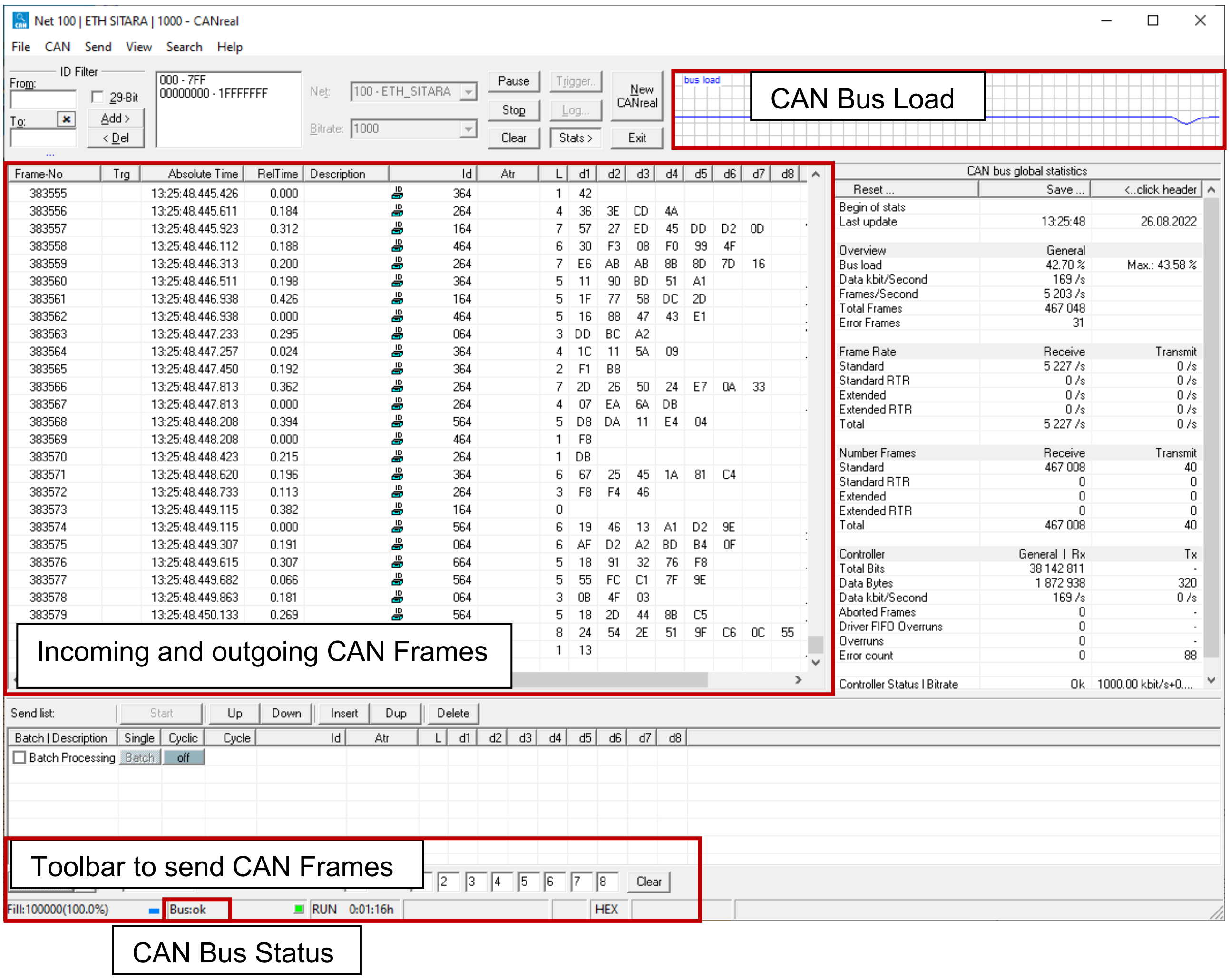
☑ Enabled

Timeouts (ms)
Connection : 2500
Command : 2500
Keep alive: 2500
Gateway: 1000

TCP/IP configuration
Hostname / IP address: 192.168.7.1
Port: 22080

Info: Ethernet-CAN-Gateway

Figure 34: CAN Control Panel



ID Filter – From: / To: ☐ 29-Bit, Add >, < Del
000 - 7FF
00000000 - 1FFFFFFF

Net: 100 - ETH_SITARA
Bitrate: 1000

Frame-No	Trg	Absolute Time	RelTime	Description	Id	Atr	L	d1	d2	d3	d4	d5	d6	d7	d8
383555		13:25:48.445.426	0.000		364		1	42							
383556		13:25:48.445.611	0.184		264		4	36	3E	CD	4A				
383557		13:25:48.445.923	0.312		164		7	57	27	ED	45	DD	D2	0D	
383558		13:25:48.446.112	0.188		464		6	30	F3	08	F0	99	4F		
383559		13:25:48.446.313	0.200		264		7	E6	AB	AB	8B	8D	7D	16	
383560		13:25:48.446.511	0.198		364		5	11	90	BD	51	A1			
383561		13:25:48.446.938	0.426		164		5	1F	77	58	DC	2D			
383562		13:25:48.446.938	0.000		464		5	16	88	47	43	E1			
383563		13:25:48.447.233	0.295		064		3	DD	BC	A2					
383564		13:25:48.447.257	0.024		364		4	1C	11	5A	09				
383565		13:25:48.447.450	0.192		364		2	F1	B8						
383566		13:25:48.447.813	0.362		264		7	2D	26	50	24	E7	0A	33	
383567		13:25:48.447.813	0.000		264		4	07	EA	6A	DB				
383568		13:25:48.448.208	0.394		564		5	D8	DA	11	E4	04			
383569		13:25:48.448.208	0.000		464		1	F8							
383570		13:25:48.448.423	0.215		264		1	DB							
383571		13:25:48.448.620	0.196		364		6	67	25	45	1A	81	C4		
383572		13:25:48.448.733	0.113		264		3	F8	F4	46					
383573		13:25:48.449.115	0.382		164		0								
383574		13:25:48.449.115	0.000		564		6	19	46	13	A1	D2	9E		
383575		13:25:48.449.307	0.191		064		6	AF	D2	A2	BD	B4	0F		
383576		13:25:48.449.615	0.307		664		5	18	91	32	76	F8			
383577		13:25:48.449.682	0.066		564		5	55	FC	C1	7F	9E			
383578		13:25:48.449.863	0.181		064		3	0B	4F	03					
383579		13:25:48.450.133	0.269		564		5	18	2D	44	8B	C5			
							8	24	54	2E	51	9F	C6	0C	55
							1	13							

CAN bus global statistics

Reset ...	Save ...	<..click header
Begin of stats		
Last update	13:25:48	26.08.2022
Overview	General	
Bus load	42.70 %	Max.: 43.58 %
Data kbit/Second	169 /s	
Frames/Second	5 203 /s	
Total Frames	467 048	
Error Frames	31	
Frame Rate	Receive	Transmit
Standard	5 227 /s	0 /s
Standard RTR	0 /s	0 /s
Extended	0 /s	0 /s
Extended RTR	0 /s	0 /s
Total	5 227 /s	0 /s
Number Frames	Receive	Transmit
Standard	467 008	40
Standard RTR	0	0
Extended	0	0
Extended RTR	0	0
Total	467 008	40
Controller	General | Rx	Tx
Total Bits	38 142 811	-
Data Bytes	1 872 938	320
Data kbit/Second	169 /s	0 /s
Aborted Frames	0	-
Driver FIFO Overruns	0	-
Overruns	0	-
Error count	0	88
Controller Status I Bitrate	Ok	1000.00 kbit/s+0....

Batch | Description	Single	Cyclic	Cycle	Id	Atr	L	d1	d2	d3	d4	d5	d6	d7	d8
☐ Batch Processing	Batch	off												

Fill:100000(100.0%) | Bus:ok | RUN 0:01:16h | HEX

Figure 35: Monitoring the CAN Bus with CANreal

NOTICE
CANreal also offers the possibility to save the current CAN frames by clicking *File→Save frames…*. In support cases this is useful to track the issue.
The log file has the extension **`.csplog.`**

7 Compatibility

The CAN-PN/2 (C.2924.02) and the CAN-PN/2-FD (C.2924.62) are successors of the CAN-PN (C.2920.02). As such the new gateways are still compatible with the predecessor. That means, it is possible to use both new gateways as replacement for a CAN-PN (C.2920.02) without any changes to the configuration.
The GSDML file and the features of the predecessor are still fully supported. However, because the new gateways use a different soft- and hardware, minor timing difference may occur.

To use the new features, which are listed below, please use the GSDML file of the CAN-PN/2(-FD).

NOTICE
It is not possible to configure the old CAN-PN (C.2920.02) with the GSDML files of the CAN-PN/2 (C.2924.02) or CAN-PN/2-FD (C.2924.62).

New features of the CAN-PN/2 which are not present on the CAN-PN

- RX- and TX-FIFO modules
- Modules for CAN statistic, CAN state and busload
- Counter and timestamp for CAN frames
- Record-based asynchronous interaction with the CAN bus
- CAN monitoring via USB
- Scheduled static CAN frames

8 Troubleshooting

This chapter shows some common error cases and how to solve them. It is explained on the Siemens TIA Portal as development environment.

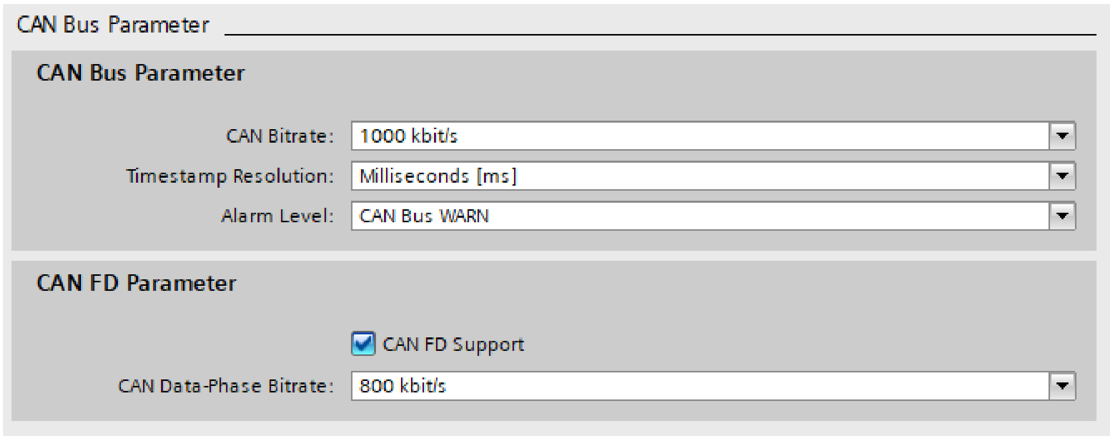
8.1 Faulty PROFINET Connection

How does the error present itself?

- The ‘CON’ LED is not lit.
- The *Device overview* displays multiple missing modules ( ).

How can the error be solved?

- Check if the PROFINET device name of the gateway and the configuration match.
- Check if the PROFINET wiring is correct.
- Check if the correct GSDML file is used. It is not possible to configure the CAN-PN/2 (C.2924.02) with the GSDML file of the CAN-PN/2-FD.
- CAN-PN/2-FD only: Check if the *CAN Data-Phase Bitrate* is equal or higher than the *CAN Bitrate* (see Figure 36).



CAN Bus Parameter

CAN Bus Parameter

CAN Bitrate:	1000 kbit/s
Timestamp Resolution:	Milliseconds [ms]
Alarm Level:	CAN Bus WARN

CAN FD Parameter

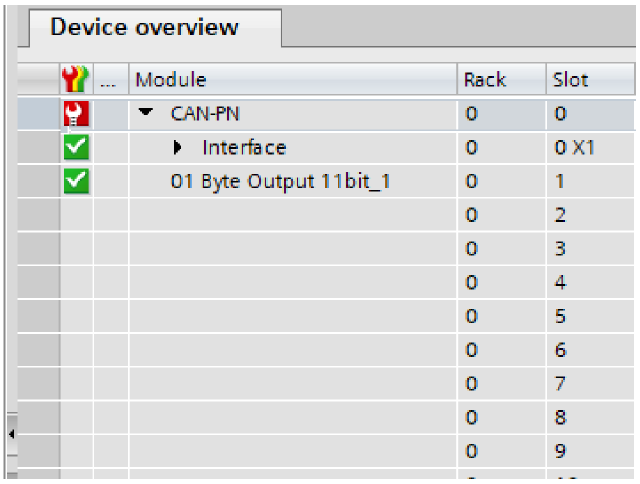
	[✓] CAN FD Support
CAN Data-Phase Bitrate:	800 kbit/s

Figure 36: Invalid CAN Data-Phase Bitrate

8.2 Faulty CAN Bus

How does the error present itself?

- The ‘E’ LED is lit continuously or lit up in single flashes (see chapter 1.5.3).
- The *Device overview* shows an alarm ( ) on the CAN-PN module (see Figure 37).



Device overview

	...	Module	Rack	Slot
🟥		CAN-PN	0	0
✓		Interface	0	0 X1
✓		01 Byte Output 11bit_1	0	1
			0	2
			0	3
			0	4
			0	5
			0	6
			0	7
			0	8
			0	9

Figure 37: Faulty CAN Bus

How can the error be solved?

- Check if all CAN devices have the same baud rate.
- Check whether the CAN bus is terminated.
- Check that the CAN bus wiring is correct (see chapter 12).
- Check the error code of the CANopen Manager module.

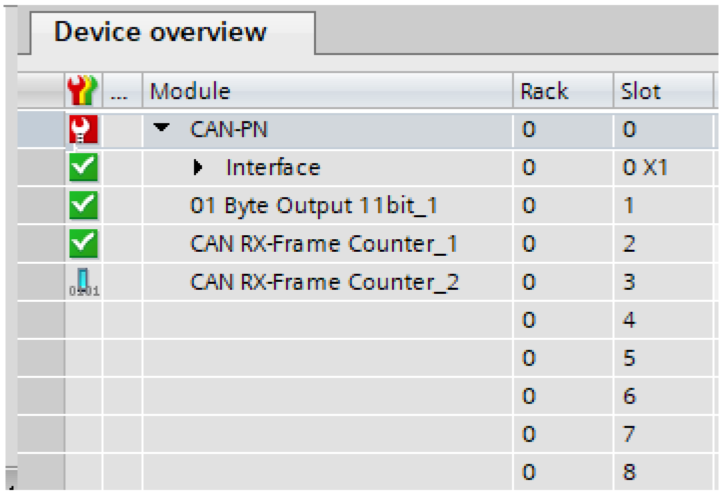
8.3 Configuration Error

How does the error present itself?

- The ‘E’ LED flashes (see chapter 1.5.3).
- One or more modules show the symbol in the *Device overview*.

How does the error can be solved?

- Check if a unique module is configured twice (see Figure 38).
    - RX-/TX-FIFO Modules
    - Bus Statistic Modules

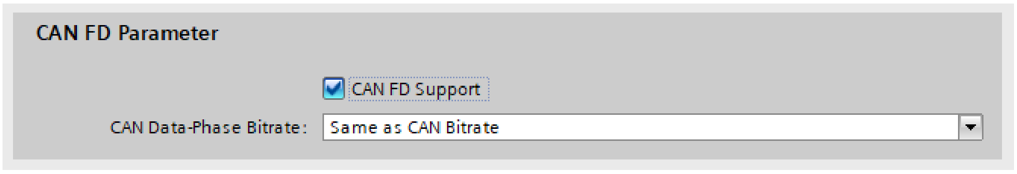


Device overview

	...	Module	Rack	Slot
		CAN-PN	0	0
✓		Interface	0	0 X1
✓		01 Byte Output 11bit_1	0	1
✓		CAN RX-Frame Counter_1	0	2
		CAN RX-Frame Counter_2	0	3
			0	4
			0	5
			0	6
			0	7
			0	8

Figure 38: Duplicates unique Modules

- **CAN-PN/2-FD only:**
  Check if a FIFO module is configured although CAN FD is not enabled on the module CAN-PN/2-FD (see Figure 39: Enable CAN FD).



CAN FD Parameter

[✓] CAN FD Support

CAN Data-Phase Bitrate: Same as CAN Bitrate

Figure 39: Enable CAN FD

8.4 Support by esd

When you have a problem with the CAN-PN/2 please make sure to check the troubleshooting chapter 8 and also chapter “CAN Troubleshooting Guide” first.
If you still cannot find the solution to the problem, don’t hesitate to contact our support team for help. Please contact our support by email to support@esd.eu or by phone +49-511-37298-130.

In order to provide the fastest and best service, please provide the following information if possible:

- Detailed error description
  - How does the error present itself?
  - Are alarms received on the PROFINET controller?
- Serial Number (printed on the device)
- GSDML File (**.xml**)
- Siemens TIA Portal Project or at least a screenshot of the *Device view*
- CAN Monitoring Log (**.csplog**) (see chapter 6)

9 Example

This chapter provides some basic examples of the FIFO and Communication Window. On top of that, it is possible, to install an example project for the CAN-PN/2 with the installer.
There are two projects, one for the CAN-PN/2 and one for the CAN-PN/2-FD. The example shows some more detailed information about the FIFO modules as well as the record functionality of the gateway.

9.1 Example for the RX-FIFO

The basic program flow can be implemented based on the pseudo code defined in chapter 4.4.7.1. In this example, an RX-FIFO module with one CAN frame exchange per cycle is used. This module has 17 bytes of input data and 1 byte of output data.

During configuration of the module, all 11-bit CAN identifier were enabled via the corresponding checkbox. Before you check whether new CAN frames have been received, check if the In-Counter (Byte 0 of the input data) and Out-Counter (Byte 0 of the input data) are equal. When this is the case, increment the Out-Counter to get new data from the gateway. Wait till the In-Counter is equal again. When this is the case, check if byte 1 of the input data is equal to ‘0’. If that is the case, no new data have been received, and the input data does not need to get evaluated further. If byte 1 is unequal to ‘0’, new data is received and needs to get processed as follows:

Input Byte	Content	Data
0	In-Counter	Out-Counter
1	Number of received frames	0x01
2	Number of remaining frames	0x03
3	Number of missed frames	0x00
4	CAN Identifier (Bit 24 ... 31 / only 29-bit CAN identifier)	0x00
5	CAN Identifier (Bit 16 ... 23 / only 29-bit CAN identifier)	0x00
6	CAN Identifier (Bit 8 ... 15)	0x00
7	CAN Identifier (Bit 0 ... 7)	0x14
8	DLC and Flags (see chapter 4.4.7.2)	0x08
9	Length	0x08
10	Data Byte 0	0x01
11	Data Byte 1	0x02
12	Data Byte 2	0x03
13	Data Byte 3	0x04
14	Data Byte 4	0x05
15	Data Byte 5	0x06
16	Data Byte 6	0x07
17	Data Byte 7	0x08

This input data means, that a CAN frame with the identifier 0x14 and the length of 8 bytes with the data bytes 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07 and 0x08 has been received. On top of that, there are 3 remaining frames in the FIFO that are send with the next cycle.

9.2 Example for the TX-FIFO

The basic program flow can be implemented based on the pseudo code defined in chapter 4.4.7.1. In this example a TX-FIFO module with one CAN frame exchange per cycle is used. This module has 1 byte of input data and 14 bytes of output data.

Before changing the data of the module, check if the In-Counter (Byte 0 of the input data) and Out-Counter (Byte 0 of the input data) are equal.
If this is the case, the output data of the module can be changed as follows:

Output Byte	Content	Data
0	Out-Counter	Out-Counter + 1
1	Number of frames to be send	0x01
0	CAN Identifier (Bit 24 ... 31 / only 29bit CAN identifier)	0x20
1	CAN Identifier (Bit 16 ... 23 / only 29bit CAN identifier)	0x00
2	CAN Identifier (Bit 8 ... 15)	0x01
3	CAN Identifier (Bit 0 ... 7)	0x23
4	DLC and Flags (see chapter 4.4.7.2)	0x05
5	Length	0x05
6	Data Byte 0	0x11
7	Data Byte 1	0x22
8	Data Byte 2	0x33
9	Data Byte 3	0x44
10	Data Byte 4	0x55
11	Data Byte 5	0x00
12	Data Byte 6	0x00
13	Data Byte 7	0x00

This change will send a CAN frame on the 29-bit identifier 0x123 and a length of 5 bytes with the data bytes 0x11, 0x22, 0x33, 0x44 and 0x55. When the In-Counter and the Out-Counter are equal again, the CAN frame is processed.

9.3 Example for the Communication Window

9.3.1 Basic Program Flow

The following basic configurations are necessary for all following examples.

In-Counter and Out-Counter

An 8-bit In-Counter and Out-Counter for the data synchronization needs to be established.

The following pseudo code will describe the basic program flow:

Step	PLC Cycle (Pseudo Code)
	..
1	Read Byte 13 (In-Counter) of the Communication Window Input.
2	Check whether Byte 13 of the Communication Window Input and Byte 13 of the Communication Window Output are equal. If they are equal, continue with step 3, otherwise with step 6.
3	Check the data of the Communication Window Input, if for example a CAN frame has been received (application based).
4	Change the data in the Communication Window Output to for example send a CAN frame (application based).
5	Increment the Byte 13 of the Communication Window Output.
6	Go on with the PLC cycle (next counter comparison in the next PLC cycle)
	..

9.3.2 Transmit a CAN Frame

Make sure that the byte 13 of the Communication Window Input and Output are equal. If so, change the data of the Communication Window Output as follows:

Output Byte	Content	Data
0 1	CAN identifier (identifier bit 10 ... 8) CAN identifier (identifier bit 7... 0)	0x00 0x12
2 3	For 11-bit CAN identifier Byte 2 and 3 are always ‘0’ For 29-bit CAN identifier Byte 2: identifier bits 28 ... 24 Byte 3: Identifier bits 23 ...16	0x00 0x00
4 5 6 7 8 9 10 11	Data Byte 0 Data Byte 1 Data Byte 2 Data Byte 3 Data Byte 4 Data Byte 5 Data Byte 6 Data Byte 7	0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07
12	Data length for transmission jobs (Tx)	0x08
13	Out-Counter	Out-Counter + 1
14	Sub command (always set to ‘0’)	0x00
15	Command (Transmit a CAN frame)	0x01

This change will send a CAN frame on the identifier 0x12 with a length of 8 bytes with the data bytes 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07. When the In-Counter and the Out-Counter are equal again, the transmission job is processed.

9.3.3 Receive a CAN Frame

9.3.3.1 Enable CAN identifier for Data Reception

Make sure that the byte 13 of the Communication Window Input and Output are equal. If so, change the data of the Communication Window Output as follows.

Output Byte	Content	Data
0 1	CAN identifier (identifier bit 10 ... 8) CAN identifier (identifier bit 7 ... 0)	0x01 0x23
2 3	For 11bit CAN identifier Byte 2 and 3 are always ‘0’ For 29bit CAN identifier Byte 2: identifier bits 28 ... 24 Byte 3: Identifier bits 23 ... 16	0x00 0x00
4 5 6 7 8 9 10 11	Data Byte 0 Data Byte 1 Data Byte 2 Data Byte 3 Data Byte 4 Data Byte 5 Data Byte 6 Data Byte 7	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
12	Data length for transmission jobs (Tx)	0x00
13	Out-Counter	Out-Counter + 1
14	Sub command (always set to ‘0’)	0x00
15	Command (Enable CAN identifier for data reception)	0x04

This change will enable the CAN identifier 0x123 for reception. When the In-Counter and the Out-Counter are equal again, the job is processed.

9.3.3.2 Reception of an enabled CAN Identifier

The CAN identifier 0x123 is enabled for reception. To receive a CAN frame, the In-Counter and Out-Counter should be checked for equality. Whenever this is the case, the application needs to check if byte 15 is equal to 0x04. If this is not the case, no new data was received. The data of the Communication Window Input does not need to be evaluated further and the Out-Counter can be incremented. When byte 15 is 0x04, a new CAN frame is received, and the input data needs to be processed:

Input Byte	Content	Data
0 1	CAN identifier (identifier bit 10...8) CAN identifier (identifier bit 7...0)	0x01 0x23
2 3	For 11bit CAN identifier Byte 2 and 3 are always '0' For 29bit CAN identifier Byte 2: identifier bits 28...24 Byte 3: Identifier bits 23...16	0x00 0x00
4 5 6 7 8 9 10 11	Data Byte 0 Data Byte 1 Data Byte 2 Data Byte 3 Data Byte 4 Data Byte 5 Data Byte 6 Data Byte 7	0xAA 0xBB 0xCC 0xDD 0xEE 0xFF 0x00 0x00
12	Number of received Data Bytes	0x06
13	In-Counter	Out-Counter
14	Number of remaining frames	0x02
15	CAN frame received	0x04

This input data means, that a CAN frame with identifier 0x123 and a length of 6 bytes with the data bytes 0xAA, 0xBB, 0xCC, 0xDD, 0xEE, 0xFF was received. On top of that, the 0x02 in byte 14 means, that there are two more frames pending in the FIFO of the Communication Window that can be processed in the next cycle.

9.3.3.3 Deactivate CAN Identifier for Data Reception

Make sure that the byte 13 of the Communication Window Input and Output are equal. If so, change the data of the Communication Window Output as follows.

Output Byte	Content	Data
0 1	CAN identifier (identifier bit 10 ... 8) CAN identifier (identifier bit 7 ... 0)	0x01 0x23
2 3	For 11-bit CAN identifier Byte 2 and 3 are always '0' For 29-bit CAN identifier Byte 2: identifier bits 28 ... 24 Byte 3: Identifier bits 23 ... 16	0x00 0x00
4 5 6 7 8 9 10 11	Data Byte 0 Data Byte 1 Data Byte 2 Data Byte 3 Data Byte 4 Data Byte 5 Data Byte 6 Data Byte 7	0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
12	Data length for transmission jobs (Tx)	0x00
13	Out-Counter	Out-Counter + 1
14	Sub command (always set to '0')	0x00
15	Command (Deactivate CAN identifier for data reception)	0x05

This change disables the CAN identifier 0x123 for reception. When the In-Counter and the Out-Counter are equal again, the job is processed. All remaining CAN frames with this identifier, that are already in the RX-FIFO of the Communication Window will still be sent to the PLC.

9.3.4 Example Program

This example uses SCL-Code to show an example implementation.

9.3.4.1 Data Types

The following data types need to be defined:

CAN frames

```
TYPE "CAN_DATA"
VERSION : 0.1
   STRUCT
      id  : UDInt;
      len : UInt;
      rtr : Bool;
      ext : Bool;
      cmd : Byte;
      dat : Array[0..7] of Byte;
   END_STRUCT;

END_TYPE
```

Structure to interact with the Communication Window with a FIFO

```
TYPE "COMM_DATA"
VERSION : 0.1
   STRUCT
      cmd_inptr        : UInt;
      cmd_outptr       : UInt;
      rx_inptr         : UInt;
      rx_outptr        : UInt;
      rx_lost          : UDInt;
      current_command : Byte;
      cmd_data        : Array[0..32] of "CAN_DATA";
      rx_data         : Array[0..32] of "CAN_DATA";
   END_STRUCT;

END_TYPE
```

Communication Windows

```
TYPE "COMM_WINDOW"
VERSION : 0.1
   STRUCT
      id        : Array[0..3] of Byte;
      data      : Array[0..7] of Byte;
      len       : Byte;
      "counter" : Byte;
      sub       : Byte;
      cmd       : Byte;
   END_STRUCT;

END_TYPE
```

IEC Timer

```
DATA_BLOCK "timer0"
{InstructionName := 'IEC_TIMER';
 LibVersion := '1.0';
 S7_Optimized_Access := 'TRUE' }
AUTHOR : Simatic
FAMILY : IEC
NAME : IEC_TMR
VERSION : 1.0
NON_RETAIN
IEC_TIMER

BEGIN

END_DATA_BLOCK
```

9.3.4.2 Data Blocks and Variables

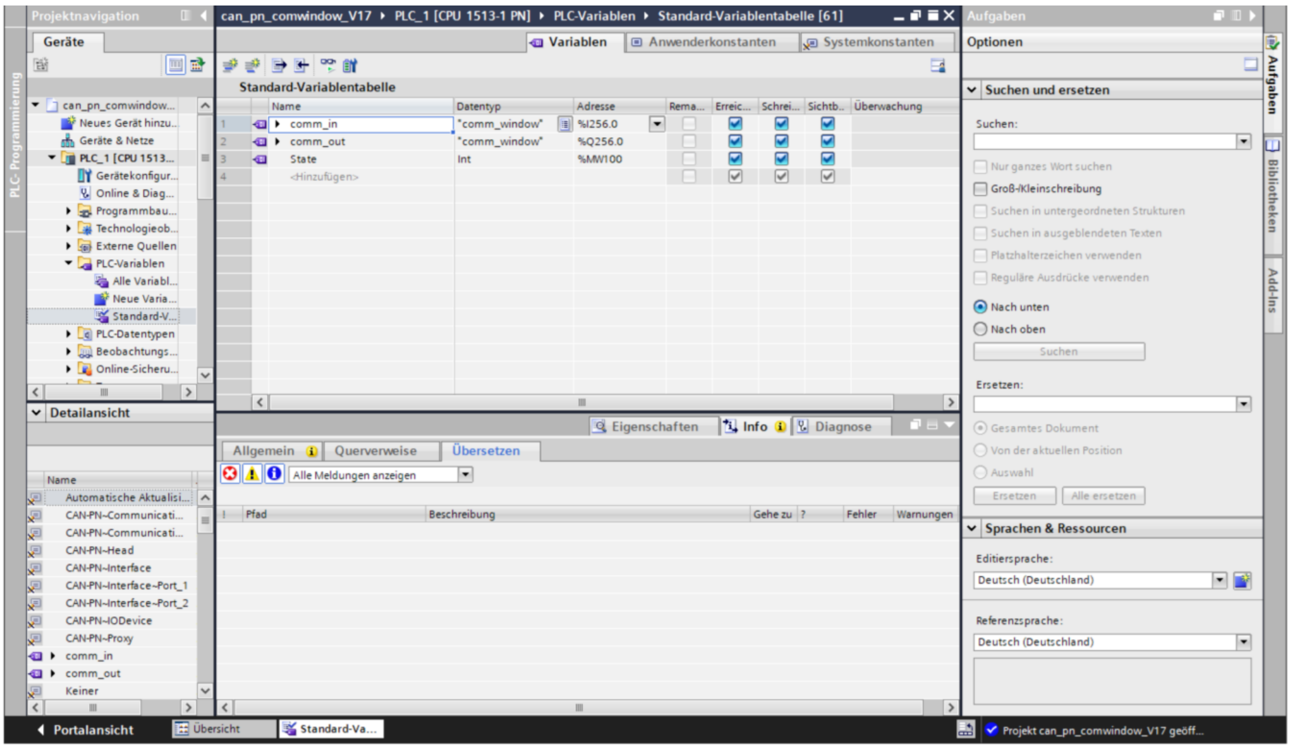
A data block (DB) of type `COMM_DATA` needs to be defined.

```
DATA_BLOCK "my_comm_data"
{ S7_Optimized_Access := 'TRUE' }
VERSION : 0.1
NON_RETAIN
"COMM_DATA"

BEGIN

END_DATA_BLOCK
```

Externally two global variables of the type 'COMM_WINDOW' are needed, which are mapped to the PLC address of the Communication Window Input (*comm_in*) and Communication Window Output (*comm_out*). Moreover, one variable of type integer called *State* is needed. The variables are declared in the *Default tag table* (see Figure 40).



can_pn_comwindow_V17 ▸ PLC_1 [CPU 1513-1 PN] ▸ PLC-Variablen ▸ Standard-Variablentabelle [61]

Standard-Variablentabelle

	Name	Datentyp	Adresse	Rema...	Erreic...	Schrei...	Sichtb...	Überwachung
1	comm_in	"comm_window"	%I256.0		✓	✓	✓	
2	comm_out	"comm_window"	%Q256.0		✓	✓	✓	
3	State	Int	%MW100		✓	✓	✓	
4	<Hinzufügen>				✓	✓	✓	

Figure 40: Default tag table

9.3.4.3 Function for Interaction with the Communication Window

The function to interact with the Communication Window is described in this chapter. It handles the commands that should be sent to the gateway and incoming data from the gateway with two separate FIFOs. Received frames are automatically stored in the FIFO while commands can be sent to the gateway without dealing with the In-Counter and Out-Counter.

Parameter

Name	Data type	Description
init	Bool	Displays the first cycle of the PLC
in_window	"COMM_WINDOW"	The input window in the input memory of the PLC
out_window	"COMM_WINDOW"	The output window in the output memory of the PLC

Example

The function must be called best in organization block OB1 with every cycle.

```
FUNCTION "com_window_handler" : Void
{ S7_Optimized_Access := 'TRUE' }
VERSION : 0.1
   VAR_INPUT
      init       : Bool;
   END_VAR

   VAR_IN_OUT
      in_window  : "COMM_WINDOW";
      out_window : "COMM_WINDOW";
      comm_data  : "COMM_DATA";
   END_VAR

   VAR_TEMP
      l          : Int;
      id_local   : UDInt;
      ptr        : UInt;
   END_VAR


BEGIN
           IF (#init = true) THEN
               #comm_data.cmd_inptr  := 0;
               #comm_data.cmd_outptr := 0;
               #comm_data.rx_inptr   := 0;
               #comm_data.rx_outptr  := 0;
               #comm_data.rx_lost    := 0;;

               #comm_data.current_command := 16#ff;

               #out_window.id[0] := 0;
               #out_window.id[1] := 0;
               #out_window.id[2] := 0;
               #out_window.id[3] := 0;

               FOR #l := 0 TO 7 DO
                   #out_window.data[#l] := 0;
               END_FOR;

               #out_window.len     := 0;
               #out_window.sub     := 0;
               #out_window.cmd     := 0;
               #out_window.counter := #in_window.counter;

           END_IF;

           IF (#comm_data.cmd_inptr <> #comm_data.cmd_outptr) THEN
               IF (#out_window.counter = #in_window.counter) THEN
                   #id_local := #comm_data.cmd_data[#comm_data.cmd_outptr].id;
                   CASE BYTE_TO_INT(#comm_data.cmd_data[#comm_data.cmd_outptr].cmd) OF
                       1:  // canWrite
                           #out_window.cmd := 1;
                           #out_window.sub := 0;
                           #out_window.len := UINT_TO_BYTE(#comm_data.cmd_data[#comm_data.cmd_outptr].len);
                           IF (#comm_data.cmd_data[#comm_data.cmd_outptr].ext) THEN
                               #out_window.id[0] := ULINT_TO_BYTE(SHR(IN := #id_local, N :=  8) AND 16#0ff);
                               #out_window.id[1] := ULINT_TO_BYTE(SHR(IN := #id_local, N :=  0) AND 16#0ff);
                               #out_window.id[2] := ULINT_TO_BYTE(SHR(IN := #id_local, N := 24) AND 16#0ff) OR 16#20;
                               #out_window.id[3] := ULINT_TO_BYTE(SHR(IN := #id_local, N := 16) AND 16#0ff);
                               ;
                           ELSE
                               #out_window.id[0] := ULINT_TO_BYTE(SHR(IN := #id_local, N := 8) AND 16#0ff);
                               #out_window.id[1] := ULINT_TO_BYTE(SHR(IN := #id_local, N := 0) AND 16#0ff);
                               #out_window.id[2] := 0;
                               #out_window.id[3] := 0;
                           END_IF;
                           FOR #l := 0 TO 7 BY 1 DO
                               #out_window.data[#l] := #comm_data.cmd_data[#comm_data.cmd_outptr].dat[#l];
                           END_FOR;
                           #comm_data.current_command := "comm_out".cmd;
                           #out_window.counter := #out_window.counter + 1;
                       4:  // canIidAdd
                           #out_window.cmd := 4;
                           #out_window.sub := 0;
                           #out_window.len := 0;
                           IF (#comm_data.cmd_data[#comm_data.cmd_outptr].ext) THEN
                               #out_window.id[0] := ULINT_TO_BYTE(SHR(IN := #id_local, N := 8) AND 16#0ff);
                               #out_window.id[1] := ULINT_TO_BYTE(SHR(IN := #id_local, N := 0) AND 16#0ff);
                               #out_window.id[2] := ULINT_TO_BYTE(SHR(IN := #id_local, N := 24) AND 16#0ff) OR 16#20;
                               #out_window.id[3] := ULINT_TO_BYTE(SHR(IN := #id_local, N := 16) AND 16#0ff);
                           ELSE
                               #out_window.id[0] := ULINT_TO_BYTE(SHR(IN := #id_local, N := 8) AND 16#0ff);
                               #out_window.id[1] := ULINT_TO_BYTE(SHR(IN := #id_local, N := 0) AND 16#0ff);
                               #out_window.id[2] := 0;
                               #out_window.id[3] := 0;
                           END_IF;
                           #comm_data.current_command := "comm_out".cmd;
                           #out_window.counter := #out_window.counter + 1;
                           ;
                       5:  // canIdDelete
                           #out_window.cmd := 5;
                           #out_window.sub := 0;
                           #out_window.len := 0;
                           IF (#comm_data.cmd_data[#comm_data.cmd_outptr].ext) THEN
                               #out_window.id[0] := ULINT_TO_BYTE(SHR(IN := #id_local, N := 8) AND 16#0ff);
                               #out_window.id[1] := ULINT_TO_BYTE(SHR(IN := #id_local, N := 0) AND 16#0ff);
                               #out_window.id[2] := ULINT_TO_BYTE(SHR(IN := #id_local, N := 24) AND 16#0ff) OR 16#20;
                               #out_window.id[3] := ULINT_TO_BYTE(SHR(IN := #id_local, N := 16) AND 16#0ff);
                           ELSE
                               #out_window.id[0] := ULINT_TO_BYTE(SHR(IN := #id_local, N := 8) AND 16#0ff);
                               #out_window.id[1] := ULINT_TO_BYTE(SHR(IN := #id_local, N := 0) AND 16#0ff);
                               #out_window.id[2] := 0;
                               #out_window.id[3] := 0;
                           END_IF;
                           #comm_data.current_command := "comm_out".cmd;
                           #out_window.counter := #out_window.counter + 1;
                           ;
                       ELSE  // Statement section ELSE
                           ;
                   END_CASE;
                   #comm_data.cmd_outptr := (#comm_data.cmd_outptr + 1) AND 31;
               END_IF;
           ELSE
               IF ("comm_in".counter = "comm_out".counter) THEN
                   #out_window.cmd := 0;
                   #comm_data.current_command := "comm_out".cmd;
                   #out_window.counter := #out_window.counter + 1;
                   IF ("comm_in".cmd = 4) THEN
                       #ptr := #comm_data.rx_inptr;
                       IF (((#ptr + 1) AND 31) <> #comm_data.rx_outptr) THEN
                           #id_local := #in_window.id[2];
                           #id_local := SHL_UDINT(IN := #id_local, N := 8);
                           #id_local := #id_local OR #in_window.id[3];
                           #id_local := SHL_UDINT(IN := #id_local, N := 8);
                           #id_local := #id_local OR #in_window.id[0];
                           #id_local := SHL_UDINT(IN := #id_local, N := 8);
                           #id_local := #id_local OR #in_window.id[1];
                           IF ((#id_local AND 16#20000000) <> 0) THEN
                               #comm_data.rx_data[#ptr].id := #id_local AND 16#1fffffff;
                               #comm_data.rx_data[#ptr].ext := true;
                           ELSE
                               #comm_data.rx_data[#ptr].id := #id_local AND 16#000007ff;
                               #comm_data.rx_data[#ptr].ext := FALSE;
                           END_IF;
                           #comm_data.rx_data[#ptr].len := BYTE_TO_UINT(#in_window.len);
                           FOR #l := 0 TO 7 DO
                               IF (#l < #comm_data.rx_data[#ptr].len) THEN
                                   #comm_data.rx_data[#ptr].dat[#l] := #in_window.data[#l];
                               ELSE
                                   #comm_data.rx_data[#ptr].dat[#l] := 0;
                               END_IF;
                           END_FOR;
                           #comm_data.rx_data[#ptr].rtr := false;
                           #comm_data.rx_inptr := (#comm_data.rx_inptr + 1) AND 31;
                       ELSE
                           #comm_data.rx_lost := #comm_data.rx_lost + 1;
                       END_IF;
                   END_IF;
               END_IF;
           END_IF;
END_FUNCTION
```

9.3.4.4 Function to add CAN Identifier

The following function can be used to enable a CAN identifier for reception.

```
FUNCTION "canIdAdd" : Bool
{ S7_Optimized_Access := 'TRUE' }
VERSION : 0.1
   VAR_INPUT
      id        : UDInt;
      ext       : Bool;
   END_VAR

   VAR_IN_OUT
      comm_data : "COMM_DATA";
   END_VAR

   VAR_TEMP
      ptr       : UInt;
      id_local  : UDInt;
   END_VAR


BEGIN
         #ptr := #comm_data.cmd_inptr;

         IF (((#ptr + 1) AND 31) = #comm_data.cmd_outptr) THEN
             #canIdAdd := false;
             RETURN;
         END_IF;

         IF (#ext = false) THEN
             #id_local := #id AND 16#000007ff;
         ELSE
             #id_local := #id AND 16#1fffffff;
         END_IF;

         #comm_data.cmd_data[#ptr].ext     := #ext;
         #comm_data.cmd_data[#ptr].id      := #id_local;
         #comm_data.cmd_data[#ptr].len     := 0;
         #comm_data.cmd_data[#ptr].rtr     := false;
         #comm_data.cmd_data[#ptr].dat[0]  := 0;
         #comm_data.cmd_data[#ptr].dat[1]  := 0;
         #comm_data.cmd_data[#ptr].dat[2]  := 0;
         #comm_data.cmd_data[#ptr].dat[3]  := 0;
         #comm_data.cmd_data[#ptr].dat[4]  := 0;
         #comm_data.cmd_data[#ptr].dat[5]  := 0;
         #comm_data.cmd_data[#ptr].dat[6]  := 0;
         #comm_data.cmd_data[#ptr].dat[7]  := 0;

         #comm_data.cmd_data[#ptr].cmd     := 4;
         #comm_data.cmd_inptr              := ((#ptr + 1) AND 31);

         #canIdAdd                         := true;
         RETURN ;

END_FUNCTION
```

9.3.4.5 Function to transmit CAN Frame

The following function can be used to transmit a CAN frame:

```
FUNCTION "canTx" : Bool
{ S7_Optimized_Access := 'TRUE' }
VERSION : 0.1
   VAR_INPUT
      id : UDInt;
      len : UInt;
      ext : Bool;
      rtr : Bool;
      data : Array[0..7] of Byte;
   END_VAR

   VAR_IN_OUT
      comm_data : "COMM_DATA";
   END_VAR

   VAR_TEMP
      ptr      : UInt;
      id_local : UDInt;
      l        : Int;
   END_VAR


BEGIN
         #ptr := #comm_data.cmd_inptr;

         IF (((#ptr + 1) AND 31) = #comm_data.cmd_outptr) THEN
             #canTx := false;
             RETURN;
         END_IF;

         IF (#ext = false) THEN
             #id_local := #id AND 16#000007ff;
         ELSE
             #id_local := #id AND 16#1fffffff;
         END_IF;

         #comm_data.cmd_data[#ptr].ext := #ext;
         #comm_data.cmd_data[#ptr].id  := #id_local;
         #comm_data.cmd_data[#ptr].len := #len;
         #comm_data.cmd_data[#ptr].rtr := #rtr;

         FOR #l := 0 TO 7 DO
             IF ((#l < #len) AND (#rtr = false)) THEN
                 #comm_data.cmd_data[#ptr].dat[#l] := #data[#l];
             ELSE
                 #comm_data.cmd_data[#ptr].dat[#l] := 0;
             END_IF;
         END_FOR;

         #comm_data.cmd_data[#ptr].cmd := 1;
         #comm_data.cmd_inptr := ((#ptr + 1) AND 31);

         #canTx := true;
         RETURN;
END_FUNCTION
```

9.3.4.6 Organization Block OB1

The following Organization Block OB1 uses a simple state machine to enable the CAN identifier 0x0, 0x80, 0x100, 0x102, 0x701 and 0x702 for reception.
After that CAN frames are sent. One CAN frame has the 11-bit identifier 0x00 and a length of 2 with the data bytes 0x01 and 0xFF. The other one has the 29-bit identifier 0x11223344 and a length of 8 with the data bytes 0x12, 0x34, 0x56, 0x78, 0x9A, 0xBC, 0xDE, 0xF0.

The source code of the program is provided on request.

```
ORGANIZATION_BLOCK "Main"
TITLE = "Main Program Sweep (Cycle)"
{ S7_Optimized_Access := 'TRUE' }
VERSION : 0.1
   VAR_TEMP
      status : Bool;
      can_tx_data : Array[0..7] of Byte;
   END_VAR


BEGIN
           "com_window_handler"(init := #Initial_Call,
                                in_window := "comm_in",
                                out_window := "comm_out",
                                comm_data := "my_comm_data");

           CASE "State" OF
               0:  #status := true;
                   "timer0".TON(IN := true, PT := T#1s, Q => #status);
               1:  #status := "canIdAdd"(id := 16#0,   ext := false, comm_data := "my_comm_data");
               2:  #status := "canIdAdd"(id := 16#80,  ext := false, comm_data := "my_comm_data");
               3:  #status := "canIdAdd"(id := 16#101, ext := false, comm_data := "my_comm_data");
               4:  #status := "canIdAdd"(id := 16#102, ext := false, comm_data := "my_comm_data");
               5:  #status := "canIdAdd"(id := 16#701, ext := false, comm_data := "my_comm_data");
               6:  #status := "canIdAdd"(id := 16#702, ext := false, comm_data := "my_comm_data");
               7:  #status := "canIdAdd"(id := 16#0  , ext := true,  comm_data := "my_comm_data");
               8:
                   #can_tx_data[0] := 16#01;
                   #can_tx_data[1] := 16#ff;
                   #status := "canTx"(id := 16#702, len := 2, rtr := false, data:=#can_tx_data, ext := false,
comm_data := "my_comm_data");
               9:
                   #can_tx_data[0] := 16#12;
                   #can_tx_data[1] := 16#34;
                   #can_tx_data[2] := 16#56;
                   #can_tx_data[3] := 16#78;
                   #can_tx_data[4] := 16#9a;
                   #can_tx_data[5] := 16#bc;
                   #can_tx_data[6] := 16#de;
                   #can_tx_data[7] := 16#f0;
                   #status := "canTx"(id := 16#11223344, len := 7, rtr := false, data := #can_tx_data, ext :=
true, comm_data := "my_comm_data");

               ELSE
                   #status := false;
           END_CASE;

           IF (#status = true) THEN
               "State" := "State" + 1;
           END_IF;
END_ORGANIZATION_BLOCK
```

10 Technical Data

10.1 General Technical Data

Power supply voltage	Nominal voltage 18 V/DC … 32 V/DC) Current consumption (24 V, 20 °C): $I_{TYPICAL}$ = 125 mA, $I_{MAXIMUM}$ = 208 mA
Power consumption	Typical: 3 W (FW 50% CPU Load and 24 V power supply) Maximum: 5 W
Protective circuits	Reverse voltage protection Protection against transient overvoltages (triggering from 26 V)
Temperature range	0 °C... +50 °C ambient temperature
Humidity	Max. 90%, non-condensing
Protection class	IP20
Pollution degree	Maximum permissible according to DIN EN 61131-2: Pollution Degree 2
Housing	Plastic housing for carrier rail mounting NS35/7,5 DIN EN 60715
Form factor / Dimensions	Width: 22.5 mm, height: 99 mm, depth: 114.5 mm (Without connectors)
Weight	130 g

Table 57: General Data of the module

10.2 CPU and Memory

CPU	ARM Cortex A9, 1 GHz, AM4377, 32-bit
SDRAM	1 Gbyte
EEPROM	256 kBit
NOR Flash	512 Mbit

Table 58: CPU and Memory

10.3 Connectors accessible from Outside

Name	Function, Port	Type
CAN	CAN	5-pos. Phoenix Contact PCB header MC 1,5/5-GF-3,81 with PCB connector FK-MCP 1,5/5-STF-3,81
PORT1	PROFINET Port 1 (EtherCAT IN)	Dual port RJ45 socket with integrated transformer and LEDs
PORT2	PROFINET Port 2 (EtherCAT OUT)	
DIAG	USB-Device	Mini-USB socket, type B
24V	24V-power supply	4-pos. Phoenix Contact PCB header MSTBO 2,5/ 4-G1L KMGY with PCB connector FKCT 2,5/4-ST

Table 59: Connectors, accessible from outside

10.4 PROFINET IO

Number of PROFINET ports	2 ports
Standard	IEEE 802.3, 100BASE-TX,
Bit rate	10/100 Mbit/s
Connection	Twisted Pair (compatible with IEEE 802.3), 100BASE-TX
Controller	Integrated in CPU
Galvanic isolation	Via transformer, integrated in RJ-45 socket
Connector	Dual port RJ-45 socket in the front panel with integrated LEDs (Link- and Activity)

Table 60: Data of the PROFINET IO interface

10.5 DIAG

Number	1
Standard	USB Specification Rev. 2.0
Bit rate	Max. 480 Mbit/s (Hi-speed)
Controller	Integrated in CPU
Connector	Mini-USB socket type B

Table 61: Data of the USB device interface

10.6 CAN/ CAN FD

Number of CAN ports	CAN-PN/2 (C.2924.02): 1x CAN CC CAN-PN/2-FD (C.2924.62): 1x CAN FD
CAN controller	According to ISO 11898-1 (CAN 2.0 A/B) CAN-PN/2 (C.2924.02): integrated in CPU CAN-PN/2-FD (C.2924.62): esdACC in Intel® MAX® 10 FPGA
CAN protocol	According to ISO 11898-1:2015:
Physical CAN Layer	High-speed CAN interface according to ISO 11898-2:2016, CAN-PN/2 (C.2924.02): Bit rate up to 1 Mbit/s CAN-PN/2-FD (C.2924.62): Bit rate up to 8 Mbit/s
Galvanic isolation	Separation by means of optocouplers and DC/DC-converters voltage over CAN isolation (CAN to slot bracket/EARTH; CAN to Host/System Ground; CAN to CAN): 1kV DC @ 1s (I < 1 mA)
Bus termination	Terminating resistor must be set externally, if required
Connector	5-pin PCB connector

Table 62: Data of the CAN interfaces

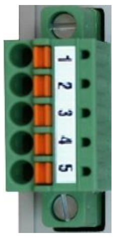
11 Connector Pin Assignments

11.1 CAN

Device connector:	Phoenix Contact PCB header MC 1,5/5-GF-3,81
Cable plug:	Phoenix Contact PCB connector FK-MCP 1,5/5-STF-3,81, Push-in spring connection, 3,81 mm pitch Phoenix Contact Order No.: 1851261, included in delivery For conductor connection and conductor cross section [1] see page 104.

Pin Position:

(Cable plug)



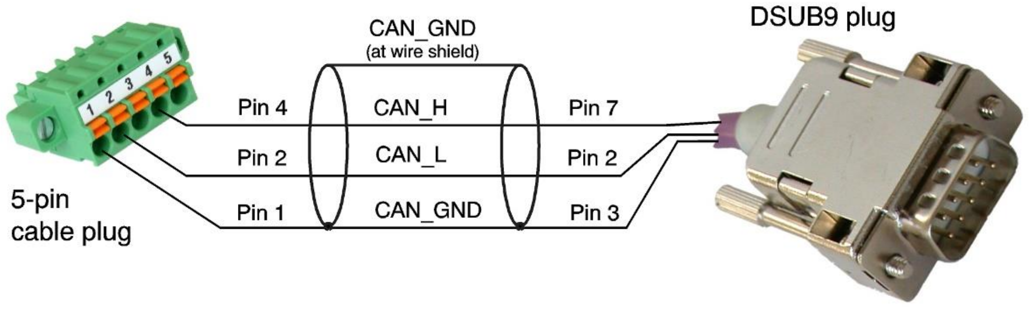
Pin Assignment:

Imprint	Signal	Pin
G	CAN_GND	1
L	CAN_L	2
Sh	Shield	3
H	CAN_H	4
●	-	5

Signal Description:

CAN_L, CAN_H …	CAN signal lines
CAN_GND …	Reference potential of the local CAN physical layer
Shield …	Pin for line shield connection (using hat rail mounting direct contact to the mounting rail potential, if it is connected)
- …	Reserved, do not connect!

Recommendation of an adapter cable from 5-pin cable plug (here Phoenix Contact FK-MCP1,5/5-STF_3,81 with spring-cage-connection) to 9-pin DSUB:



The assignment of the 9-pin DSUB-connector and the cable plug is designed according to CiA 106.

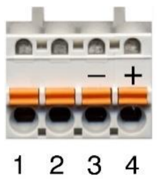
1) For further technical data see Phoenix Contact website, PCB Connectors, Product list PCB connectors

11.2 24 V Power Supply Voltage

DANGER
The CAN-PN/2 is a device of protection class III according to DIN EN 61140 and may only be operated on supply circuits that offer sufficient protection against dangerous voltages.

Device socket:	Phoenix Contact PCB header MSTBO 2,5/4-G1L-KMGY
Cable plug:	Phoenix Contact PCB connector FKCT 2,5/4-ST, 5.0 mm pitch, Push-in spring connection, included in the scope of delivery (Phoenix Contact order No.: 19 21 90 0) For conductor connection and conductor cross section 2) see page 104.

Pin Position on cable plug:



Pin Assignment:

			24V	
Device housing label	.	.	M	P
Connector label	(none)	(none)	-	+

Pin	1	2	3	4
Signal	**Do not connect !**	**Do not connect !**	M24 (GND)	P24 (+ 24 V)

Please refer to the connecting diagram page 15.

NOTICE
Feeding through the +24V power supply voltage can cause damage on the modules.
It is not permitted to feed through the power supply voltage through this connector and to supply the power supply voltage to another CAN module station!

→ Make absolutely sure to connect the cables correctly to the cable plug!
→ Use only suitable cables for the line plug.

Signal Description:

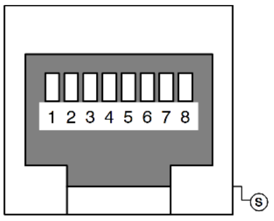
P24...	Power supply voltage (18 V … 32 V)
M24...	Reference potential

2) For further technical data see Phoenix Contact website, PCB Connectors, Product list PCB connectors

11.3 PROFINET IO

Device Connector:	RJ45 socket, 8-pin According to IEEE 802.3-2015, “Table 25–2—Twisted-pair MDI contact assignments”

Pin Position:



Pin Assignment:

Pin	Signal	Meaning
1	Tx0+ (TxD+)	Transmit Data +
2	Tx0- (TxD-)	Transmit Data -
3	Rx0+ (RxD+)	Receive Data +
4	-	-
5	-	-
6	Rx0- (RxD-)	Receive Data -
7	-	-
8	-	-

S	Shield	

Signal Description:

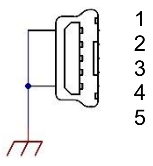
Tx0+/-, Rx0+/- ...	Ethernet data lines
- ...	reserved for future applications, do not connect!
Shield...	case shield, connected with the front panel of the CAN-PN/2

NOTICE
Cables of category CAT5 or higher must be used to grant the function in networks with 100 Mbit/s. esd grants the EU conformity of the product if the wiring is carried out with shielded twisted pair cables.

11.4 DIAG

Device connector: USB 2.0 Mini-B receptacle, standard pinning

Pin Position:



Pin Assignment:

Pin	Signal
1	V_{BUS}
2	D-
3	D+
4	-
5	GND

Signal Description:

VBUS ...	+5 V power supply voltage
D+, D-...	Data USB 2.0, differential pair +/-
-	Reserved (ID for USB-type). Do not connect!
GND...	Reference potential

11.5 Conductor Connection/Conductor Cross Section

The following table contains an extract of the technical data of the cable plugs.

Characteristics	***Connector Type*[3]**	
	Power Supply Voltage 24 V	CAN-Connector
Connector type plug component	FKCT 2,5/..-ST KMGY	FK-MCP 1,5/5-STF-3,81
Connection method	Push-in spring connection	Push-in spring connection
Stripping length	10 mm	9 mm
Conductor cross section rigid.	0.2 mm² … 2.5 mm²	0.14 mm² … 1.5 mm²
Conductor cross section flexible	0.2 mm² … 2.5 mm²	0.14 mm² … 1.5 mm²
Conductor cross section AWG	24 … 12	26 … 16
Conductor cross section flexible, with ferrule without plastic sleeve	0.25 mm² … 2.5 mm²	0.25 mm² … 1.5 mm²
Conductor cross section flexible, with ferrule with plastic sleeve	0.25 mm² … 2.5 mm²	0.25 mm² … 0.75 mm²
2 conductors with same cross section, stranded, TWIN ferrules with plastic sleeve, min./max.	0.5 mm² … 1.5 mm²	not allowed

[3] Technical Data from Phoenix Contact website, printed circuit board connector, plug component

12 Correct Wiring of Galvanically Isolated CAN Networks

NOTICE
This chapter applies to CAN networks with bit rates up to 1 Mbit/s.
If you work with higher bit rates, as for example used for CAN FD, the information given in this chapter must be examined for applicability in each individual case.
For further information refer to the CiA® CAN FD guidelines and recommendations (https://www.can-cia.org/).

For the CAN wiring all applicable rules and regulations (EU, DIN), such as regarding electromagnetic compatibility, security distances, cable cross-section or material, must be observed.

12.1 CAN Wiring Standards

The flexibility in CAN network design is a major strength of the various extensions based on the original CAN standard ISO 11898-2, such as CANopen®, ARINC825, DeviceNet® and NMEA2000. However, taking advantage of this flexibility absolutely requires a network design that considers the interactions of all network parameters.

In some cases, the CAN organizations have adapted the scope of CAN in their specifications to enable applications outside the ISO 11898 standard. They have imposed system-level restrictions on data rate, line length and parasitic bus loads.

However, when designing CAN networks, a margin must always be planned for signal losses over the entire system and cabling, parasitic loads, network imbalances, potential differences against earth potential, and signal integrities. **Therefore, the maximum achievable number of nodes, bus lengths and stub lengths may differ from the theoretically possible number!**

esd has limited its recommendations for CAN wiring to the specifications of ISO 11898-2.
A description of the special features of the derived specifications CANopen, ARINC825, DeviceNet, and NMEA2000 is omitted here

The consistent compliance with the ISO 11898-2 standard offers significant advantages:

- Reliable operation due to proven design specifications
- Minimization of error sources due to sufficient distance to the physical limits.
- Easy maintenance because there are no "special cases" to consider for future network modifications and troubleshooting.

Of course, reliable networks can be designed according to the specifications of CANopen, ARINC825, DeviceNet and NMEA2000, **however it must be observed that it is strictly not recommended to mix the wiring guidelines of the various specifications!**

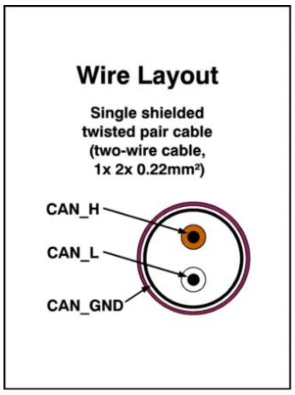
12.2 Light Industrial Environment (*Single* Twisted Pair Cable)

12.2.1 General Rules

NOTICE
esd grants the EU Conformity of the product if the CAN wiring is carried out with at least single shielded **single** twisted pair cables that match the requirements of ISO 11898-2. Single shielded *double* twisted pair cable wiring as described in chapter 12.3 ensures the EU Conformity as well.

The following **general rules** for CAN wiring with single shielded *single* twisted pair cable should be followed:

1	A suitable cable type with a wave impedance of about 120 Ω ±10% with an adequate conductor cross-section (≥ 0.22 mm²) must be used. The voltage drop over the wire must be considered.
2	For light industrial environment use at least a two-wire CAN cable, the wires of which must be assigned as follows: • Two twisted wires must be assigned to the data signals (CAN_H, CAN_L). • The cable shield must be connected to the reference potential (CAN_GND).
3	The reference potential CAN_GND must be connected to the functional earth (FE) at exactly **one** point.
4	A CAN bus line must not branch (exception: short cable stubs) and must be terminated with the characteristic impedance of the line (generally 120 Ω ±10%) at both ends (between the signals CAN_L and CAN_H and **not** at CAN_GND).
5	Keep cable stubs as short as possible (l < 0.3 m).
6	Select a working combination of bit rate and cable length.
7	Keep away cables from disturbing sources. If this cannot be avoided, double shielded wires are recommended.


Wire Layout
Single shielded twisted pair cable (two-wire cable, 1x 2x 0.22mm²)
CAN_H
CAN_L
CAN_GND

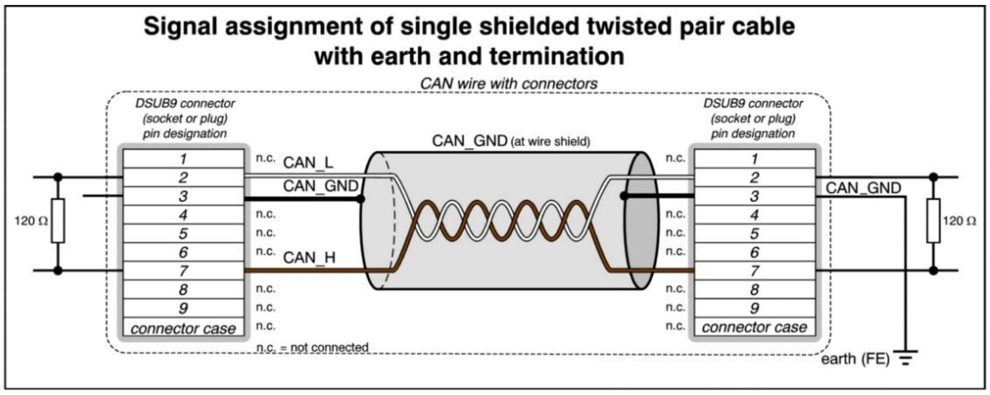

Signal assignment of single shielded twisted pair cable with earth and termination
CAN wire with connectors
DSUB9 connector (socket or plug) pin designation
CAN_GND (at wire shield)
n.c.
CAN_L
CAN_GND
CAN_H
1
2
3
4
5
6
7
8
9
connector case
120 Ω
n.c. = not connected
earth (FE)

Figure 41: CAN wiring for light industrial environment

12.2.2 Cabling

- To connect CAN devices with just one CAN connector per net use a short stub (< 0.3 m) and a T-connector (available as accessory). If these devices are located at the end of the CAN network, the CAN terminator “CAN-Termination-DSUB9” can be used.

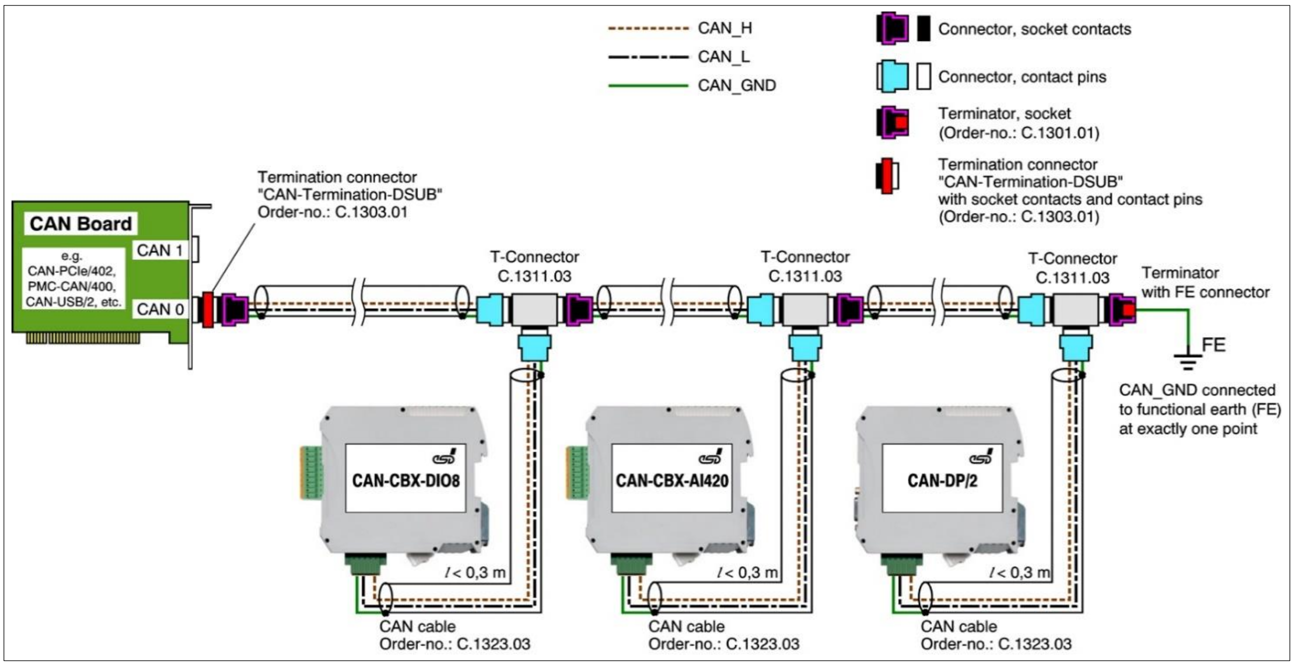

CAN_H
CAN_L
CAN_GND
Connector, socket contacts
Connector, contact pins
Terminator, socket (Order-no.: C.1301.01)
Termination connector "CAN-Termination-DSUB" with socket contacts and contact pins (Order-no.: C.1303.01)
Termination connector "CAN-Termination-DSUB" Order-no.: C.1303.01
CAN Board
e.g. CAN-PCIe/402, PMC-CAN/400, CAN-USB/2, etc.
CAN 1
CAN 0
T-Connector C.1311.03
T-Connector C.1311.03
T-Connector C.1311.03
Terminator with FE connector
FE
CAN_GND connected to functional earth (FE) at exactly one point
CAN-CBX-DIO8
CAN-CBX-AI420
CAN-DP/2
l < 0,3 m
l < 0,3 m
l < 0,3 m
CAN cable Order-no.: C.1323.03
CAN cable Order-no.: C.1323.03
CAN cable Order-no.: C.1323.03

Figure 42: Example for proper wiring with single shielded single twisted pair wires

12.2.3 Branching

- In principle the CAN bus must be realized in a line. The nodes are connected to the main CAN bus line via short cable stubs. This is normally realised by so called T-connectors. esd offers the CAN-T-Connector (Order No.: C.1311.03)
- If a mixed application of single twisted and double twisted cables cannot be avoided, ensure that the CAN_GND line is not interrupted!
- Deviations from the bus structure can be realized by using repeaters.

12.2.4 Termination Resistor

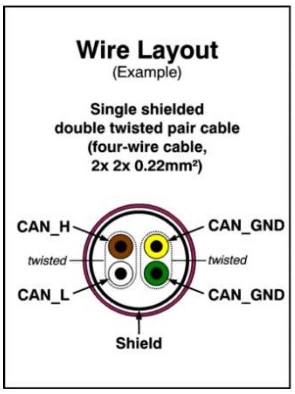
- A termination resistor must be connected at both ends of the CAN bus.
  If an integrated CAN termination resistor is connected to the CAN interface at the end of the CAN bus, this integrated termination must be used instead of an external CAN termination resistor.
- 9-pole DSUB-termination connectors with integrated termination resistor and pin contacts and socket contacts are available from esd (order no. C.1303.01).
- For termination of the CAN bus and grounding of the CAN_GND, DSUB terminators with pin contacts (order no. C.1302.01) or socket contacts (order no. C.1301.01) and with additional functional earth contact are available.

12.3 Heavy Industrial Environment (Double Twisted Pair Cable)

12.3.1 General Rules

The following **general rules** for the CAN wiring with single shielded *double* twisted pair cable should be followed:

1	A suitable cable type with a wave impedance of about 120 Ω ±10% with an adequate conductor cross-section (≥ 0.22 mm²) must be used. The voltage drop over the wire must be considered.
2	For heavy industrial environment use a four-wire CAN cable, the wires of which must be assigned as follows: • Two twisted wires must be assigned to the data signals (CAN_H, CAN_L) and • The other two twisted wires must be assigned to the reference potential (CAN_GND). • The cable shield must be connected to functional earth (FE) at least at one point.
3	The reference potential CAN_GND must be connected to the functional earth (FE) at exactly **one** point.
4	A CAN bus line must not branch (exception: short cable stubs) and must be terminated with the characteristic impedance of the line (generally 120 Ω ±10%) at both ends (between the signals CAN_L and CAN_H and **not** to CAN_GND).
5	Keep cable stubs as short as possible (l < 0.3 m).
6	Select a working combination of bit rate and cable length.
7	Keep away CAN cables from disturbing sources. If this cannot be avoided, double shielded cables are recommended.


Wire Layout
(Example)
Single shielded double twisted pair cable (four-wire cable, 2x 2x 0.22mm²)
CAN_H
CAN_GND
twisted
twisted
CAN_L
CAN_GND
Shield

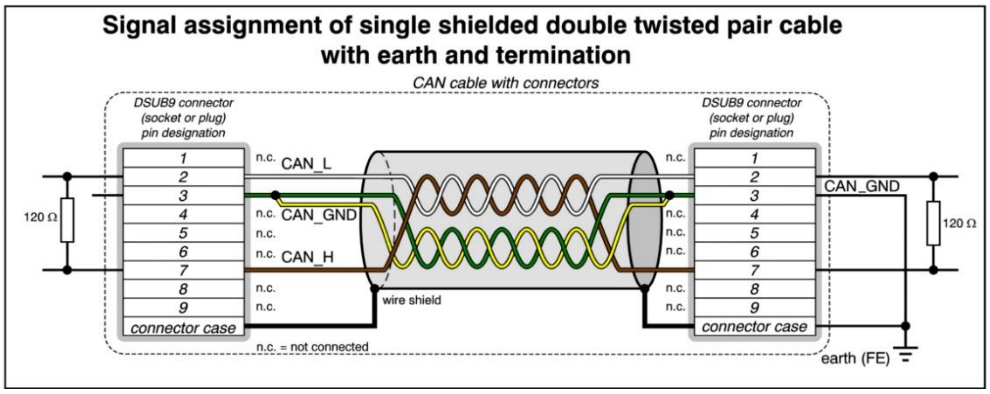
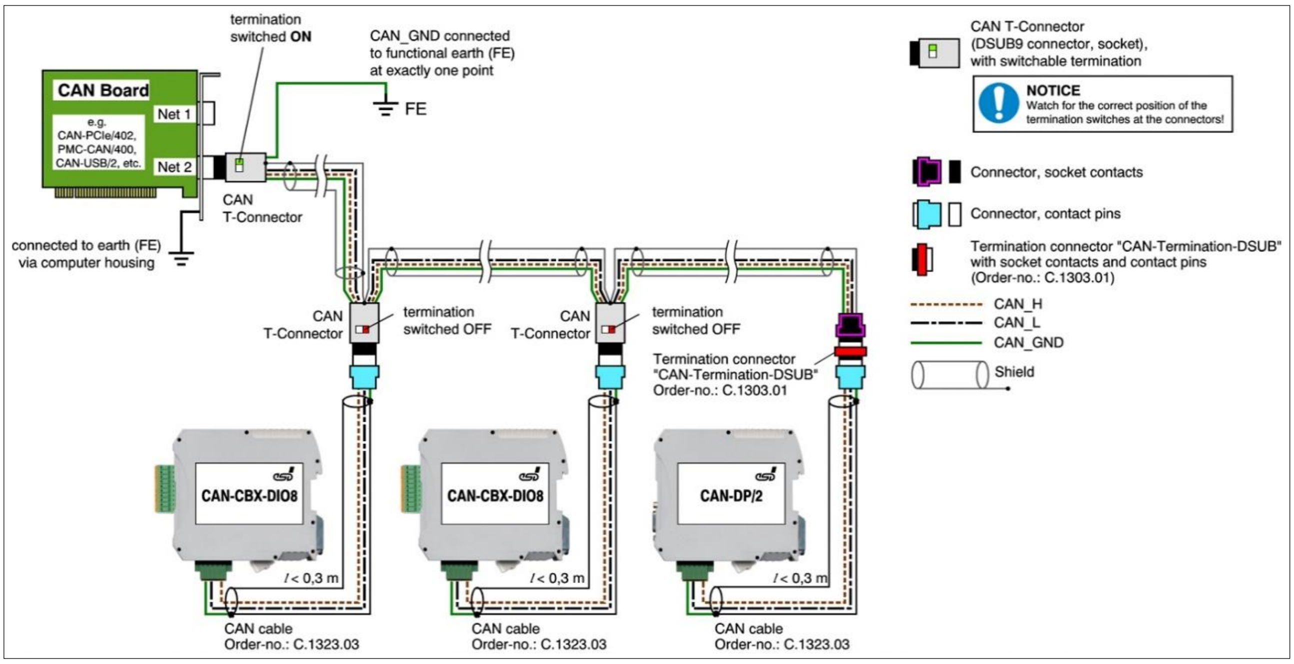

Signal assignment of single shielded double twisted pair cable with earth and termination
CAN cable with connectors
DSUB9 connector (socket or plug) pin designation
DSUB9 connector (socket or plug) pin designation
1
2
3
4
5
6
7
8
9
connector case
n.c.
CAN_L
CAN_GND
CAN_H
120 Ω
wire shield
n.c. = not connected
earth (FE)

Figure 43: CAN wiring for heavy industrial environment

12.3.2 Device Cabling



termination switched ON
CAN_GND connected to functional earth (FE) at exactly one point
FE
CAN Board
e.g. CAN-PCIe/402, PMC-CAN/400, CAN-USB/2, etc.
Net 1
Net 2
CAN T-Connector
connected to earth (FE) via computer housing
CAN T-Connector
termination switched OFF
CAN T-Connector
termination switched OFF
Termination connector "CAN-Termination-DSUB" Order-no.: C.1303.01
CAN-CBX-DIO8
CAN-CBX-DIO8
CAN-DP/2
l < 0,3 m
CAN cable Order-no.: C.1323.03
CAN T-Connector (DSUB9 connector, socket), with switchable termination
NOTICE Watch for the correct position of the termination switches at the connectors!
Connector, socket contacts
Connector, contact pins
Termination connector "CAN-Termination-DSUB" with socket contacts and contact pins (Order-no.: C.1303.01)
CAN_H
CAN_L
CAN_GND
Shield

Figure 44: Example of proper wiring with single shielded double twisted pair cables

12.3.3 Branching

- In principle, the CAN bus must be realized in a line. The nodes are connected to the main CAN bus line via short cable stubs. This is usually realised via so called T-connectors.
  When using esd's CAN-T-Connector (order no.: C.1311.03) in heavy industrial environment and with four-wire twisted cables, it must be noted that the shield potential of the conductive DSUB housing is not looped through this type of T-connector. This interrupts the shielding. Therefore, you must take appropriate measures to connect the shield potentials, as described in the manual of the CAN-T-Connector. For further information on this, please refer to the CAN-T-Connector Manual (order no.: C.1311.21).
  Alternatively, a T-connector can be used, in which the shield potential is looped through, for example the DSUB9 connector from ERNI (ERBIC CAN BUS MAX, order no.:154039).
- If a mixed application of single twisted and double twisted cables cannot be avoided, ensure that the CAN_GND line is not interrupted!
- Deviations from the bus structure can be realized by using repeaters.

12.3.4 Termination Resistor

- A termination resistor must be connected at both ends of the CAN bus.
  If an integrated CAN termination resistor is connected to the CAN interface at the end of the CAN bus, this integrated termination must be used instead of an external CAN termination resistor.
- 9-pole DSUB-termination connectors with integrated termination resistor and pin contacts and socket contacts are available from esd (order no. C.1303.01).
- 9-pole DSUB-connectors with integrated switchable termination resistor can be ordered for example from ERNI (ERBIC CAN BUS MAX, socket contacts, order no.:154039).

12.4 Electrical Grounding

- For CAN devices with electrical isolation the CAN_GND must be connected between the CAN devices.
- CAN_GND should be connected to the earth potential (FE) at **exactly one** point of the network.
- Each *CAN interface with electrical connection to earth potential* acts as a grounding point. For this reason, it is recommended not to connect more than one *CAN device with electrical connection to earth potential.*
- Grounding can be done for example at a termination connector (e.g. order no. C.1302.01 or C.1301.01).

12.5 Bus Length

The bus length of a CAN network must be adapted to the set bit rate. The maximum values result from the fact that the time required for a bit to be transmitted in the bus system is shorter the higher the transmission rate is. However, as the line length increases, so does the time it takes for a bit to reach the other end of the bus. It should be noted that the signal is not only transmitted, but the receiver must also respond to the transmitter within a certain time. The transmitter, in turn, must detect any change in bus level from the receiver(s). Delay times on the line, the transceiver, the controller, oscillator tolerances and the set sampling time must be considered.
In the following table you will find guide values for the achievable bus lengths at certain bit rates.

Bit Rate [kbit/s]	Theoretical values of reachable wire length **with esd interface** l_{max} [m]	**CiA recommendations** (07/95) for reachable wire lengths l_{min} [m]	Standard values of the cross-section according to CiA 303-1 [mm²]
1000	37	25	0.25 to 0.34
800	59	50	
$666.\overline{6}$	80	-	
500	130	100	0.34 to 0.6
$333.\overline{3}$	180	-	
250	270	250	
166	420	-	0.5 to 0.6
125	570	500	
100	710	650	
$83.\overline{3}$	850	-	0.75 to 0.8
$66.\overline{6}$	1000	-	
50	1400	1000	
$33.\overline{3}$	2000	-	
20	3600	2500	not defined in CiA 303-1
12.5	5400	-	
10	7300	5000	

Table 63: Recommended cable lengths at typical bit rates (with esd-CAN interfaces)

Optical couplers are delaying the CAN signals. esd modules typically achieve a wire length of 37 m at 1 Mbit/s within a proper terminated CAN network without impedance disturbances, such as those caused by cable stubs > 0.3 m.

NOTICE
Please note that the cables, connectors, and termination resistors used in CANopen networks shall meet the requirements defined in ISO 11898-2.
In addition, further recommendations of the CiA, like standard values of the cross section, depending on the cable length, are described in the CiA recommendation CiA 303-1 (see CiA 303 CANopen Recommendation - Part 1: “Cabling and connector pin assignment,” Version 1.9.0, Table 2). Recommendations for pin-assignment of the connectors are described in CiA 106: “Connector pin-assignment recommendations ”.

12.6 Examples for CAN Cables

esd recommends the following two-wire and four-wire cable types for CAN network design. These cable types are used by esd for ready-made CAN cables, too.

12.6.1 Cable for Light Industrial Environment Applications (Two-Wire)

Manufacturer	Cable Type	
U.I. LAPP GmbH Schulze-Delitzsch-Straße 25 70565 Stuttgart Germany www.lappkabel.com	e.g. UNITRONIC ®-BUS CAN UL/CSA (1x 2x 0.22) (UL/CSA approved) UNITRONIC ®-BUS-FD P CAN UL/CSA (1x 2x 0.25) (UL/CSA approved)	 Part No.: 2170260 Part No.: 2170272
ConCab GmbH Äußerer Eichwald 74535 Mainhardt Germany www.concab.de	e. g. BUS-PVC-C (1x 2x 0.22 mm²) BUS-Schleppflex-PUR-C (1x 2x 0.25 mm²)	 Order No.: 93 022 016 (UL appr.) Order No.: 94 025 016 (UL appr.)

12.6.2 Cable for Heavy Industrial Environment Applications (Four-Wire)

Manufacturer	Cable Type	
U.I. LAPP GmbH Schulze-Delitzsch-Straße 25 70565 Stuttgart Germany www.lappkabel.com	e.g. UNITRONIC ®-BUS CAN UL/CSA (2x 2x 0.22) (UL/CSA approved) UNITRONIC ®-BUS-FD P CAN UL/CSA (2x 2x 0.25) (UL/CSA approved)	 Part No.: 2170261 Part No.: 2170273
ConCab GmbH Äußerer Eichwald 74535 Mainhardt Germany www.concab.de	e. g. BUS-PVC-C (2x 2x 0.22 mm²) BUS-Schleppflex-PUR-C (2x 2x 0.25 mm²)	 Order No.: 93 022 026 (UL appr.) Order No.: 94 025 026 (UL appr.)

INFORMATION
Ready-made CAN cables with standard or custom length can be ordered from **esd**.

13 CAN Troubleshooting Guide

The CAN Troubleshooting Guide is a guide to finding and eliminating the most common problems and errors when setting up CAN bus networks and CAN-based systems.

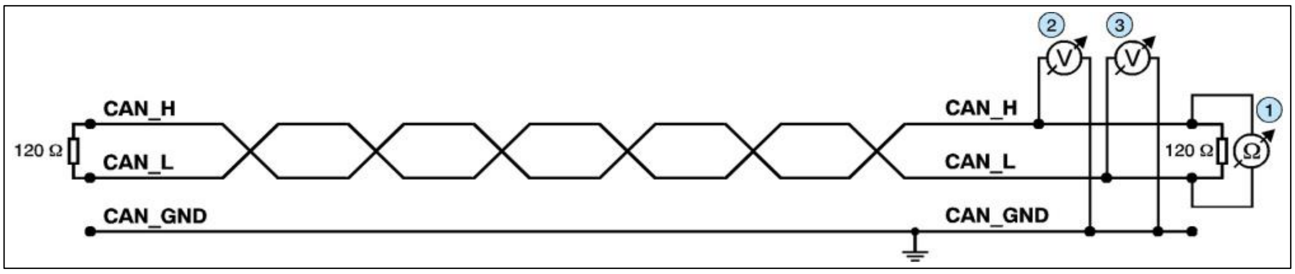


Figure 45: Simplified diagram of a CAN network

Termination
The bus termination is used to match impedance of a node to the impedance of the bus line used. If the impedance is mismatched, the transmitted signal is not completely absorbed by the load and will be partially reflected back into the transmission line.
If the impedances of the sources, transmission lines and loads are equal, the reflections are avoided. This test measures the total resistance of the two CAN data lines and the connected terminating resistors.

To **test this, please proceed as follows:**

1. Switch off the supply voltages of all connected CAN nodes.
2. Measure the DC resistance between CAN_H and CAN_L at one end of the network, measuring point ① (see figure above).

Expected result:
The measured value should be between 50 Ω and 70 Ω.

Possible causes of error:
▪ If the determined value is below 50 Ω, please make sure that:
- There is no **short circuit** between CAN_H and CAN_L wiring.
- **No more than two** terminating resistors are connected.
- The transceivers of the individual nodes are not defective.

▪ If the determined value is higher than 70 Ω, please make sure that:
- All CAN_H and CAN_L lines are correctly connected.
- Two terminating resistors of 120 Ω each are connected to your CAN network (one at each end).

13.1 Electrical Grounding

The CAN_GND of the CAN network should be connected to the functional earth potential (FE) at only **one** point. This test indicates whether the CAN_GND is grounded at one or more points.

Please note that this test can only be performed with electrically isolated CAN nodes.

To test this, please proceed as follows:

1. Disconnect the CAN_GND from the earth potential (FE).

2. Measure the DC resistance between CAN_GND and earth potential (see figure on the right).

Do not forget to reconnect CAN_GND to earth potential after the test!

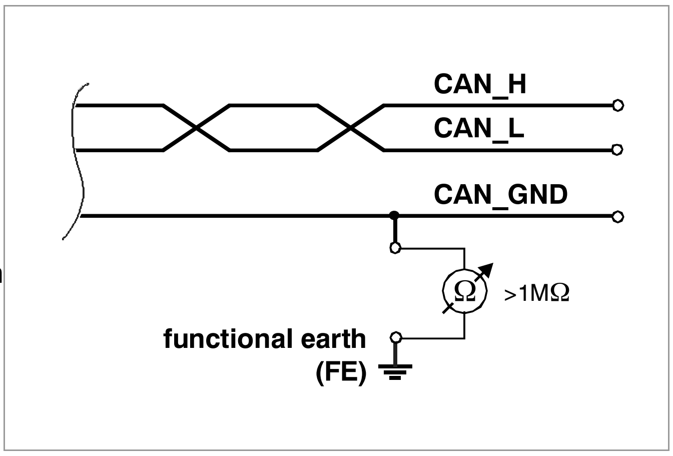


Figure 46: Simplified schematic diagram of ground test measurement

Expected result:
The measured resistance should be greater than 1 MΩ. If it is smaller, please search for additional grounding of the CAN_GND wires.

13.2 Short Circuit in CAN Wiring

A CAN bus might possibly still be able to transmit data even if CAN_GND and CAN_L are short-circuited. However, this will usually cause the error rate to rise sharply.
Ensure that there is no short circuit between CAN_GND and CAN_L!

13.3 Correct Voltage Levels on CAN_H and CAN_L

Each node contains a CAN transceiver that outputs differential signals. When the network communication is idle the CAN_H and CAN_L voltages are approximately 2.5 V measured to CAN_GND. Defective transceivers can cause the idle voltages to vary and disrupt network communication.

To test for defective transceivers, please proceed as follows:

1. Switch on all supply voltages.
2. Terminate all network communication.
3. Measure the DC voltage between CAN_H and CAN_GND, measuring point ②.
   (See “Simplified diagram of a CAN network” on previous page).
4. Measure the DC voltage between CAN_L and CAN_GND, measuring point ③.
   (See “Simplified diagram of a CAN network” on previous page).

Expected result:
The measured voltage should be between 2.0 V and 3.0 V.

Possible causes of error:

- If the voltage is lower than 2.0 V or higher than 3.0 V, it is possible that one or more nodes have defective transceivers.
  - If the voltage is lower than 2.0 V, please check the connections of the CAN_H and CAN_L lines.
- To find a node with a defective transceiver within a network, please check individually the resistances of the CAN transceivers of the nodes (see next section).

13.4 CAN Transceiver Resistance Test

CAN transceivers have circuits that control CAN_H and CAN_L. Experience shows that electrical damage can increase the leakage current in these circuits.

To measure the current leakage through the CAN circuits, please use an ohmmeter and proceed as follows:

1. Switch **off** the node ④ and **disconnect** it from the CAN network.
   (See figure below.)
2. Measure the DC resistance between CAN_H and CAN_GND, measuring point ⑤
   (See figure below.)
3. Measure the DC resistance between CAN_L and CAN_GND, measuring point ⑥
   (See figure below.)

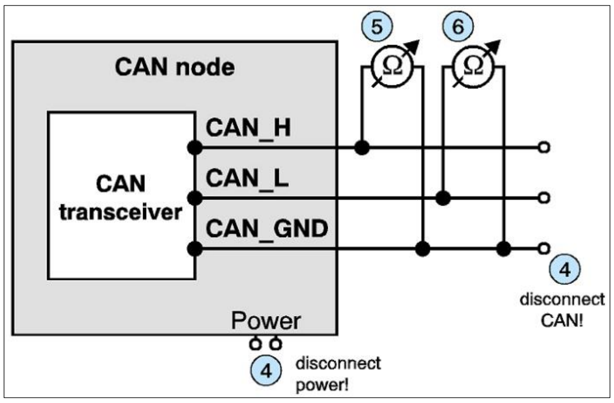


Figure 47: Measuring the internal resistance of CAN transceivers

Expected result:
The measured resistance should be greater than 10 kΩ for each measurement.

Possible causes of error:

- If the resistance is significantly lower, the CAN transceiver may be defective.
- Another indication of a defective CAN transceiver is a very high deviation of the two measured input resistances (>> 200 %).

13.5 Support by esd

If you have followed the troubleshooting steps in this troubleshooting guide and still cannot find a solution to your problem, our support team can help.
Please contact our support by email to support@esd.eu or by phone **+49-511-37298-130.**

14 References

(1) IEEE Standard for Ethernet, IEEE Std 802.3™-2015, IEEE Standards Association, New York, USA,

15 Software Licenses

NOTICE
The software used for the CAN-PN/2 from esd and from third parties is subject to licenses. You must read and accept these license conditions before the installation!

The license terms of esd (esd electronics License Conditions) and of 3rd parties (3rd Party Licenses) are displayed and installed on your system during installation via the installation program (CAN-PN/2(-FD)_X_X_X.exe (see chapter 4.2).
You can also download the licenses from our website, see the following chapters.

15.1 3rd Party Software License Terms

License Name	**Identifier** (from SPDX License List)
License Conditions for Siemens Profinet Stack	n.a.
Apache License 2.0	Apache-2.0
BSD 2-Clause "Simplified" License	BSD-2-Clause
BSD 3-Clause "New" or "Revised" License	BSD-3-Clause
BSD 4-Clause "Original" or "Old" License	BSD-4-Clause
bzip2 and libbzip2 License v1.0.4	n.a.
GNU General Public License v2.0 only	GPL-2.0-only
GNU General Public License v3.0 only	GPL-3.0-only
GCC Runtime Library exception 3.1	(GPL-3.0 with) GCC-exception-3.1
ISC License	ISC
GNU Library General Public License v2 only	LGPL-2.0
GNU Lesser General Public License v2.1 only	LGPL-2.1
GNU Lesser General Public License v3.0 only	LGPL-3.0
MIT License	MIT
Spencer License 94	Spencer-94
Texas Instruments Text File License	(TI-TFL)
Texas Instruments Technology and Software Publicly Available Software License	(TI-TSPA)
Unicode License Agreement - Data Files and Software (2016)	Unicode-DFS-2016

15.2 Licence Conditions of the Software Modules

15.2.1 Yocto-Linux License Modules

PACKAGE NAME: amx3-cm3
PACKAGE VERSION: 1.9.2
RECIPE NAME: amx3-cm3
LICENSE: TI-TSPA

PACKAGE NAME: base-files
PACKAGE VERSION: 3.0.14
RECIPE NAME: base-files
LICENSE: GPL-2.0-only

PACKAGE NAME: base-passwd
PACKAGE VERSION: 3.5.29
RECIPE NAME: base-passwd
LICENSE: GPL-2.0-only

PACKAGE NAME: bash
PACKAGE VERSION: 5.1.16
RECIPE NAME: bash
LICENSE: GPL-3.0-or-later
PACKAGE NAME: busybox
PACKAGE VERSION: 1.35.0
RECIPE NAME: busybox
LICENSE: GPL-2.0-only & bzip2-1.0.4

PACKAGE NAME: busybox-syslog
PACKAGE VERSION: 1.35.0
RECIPE NAME: busybox
LICENSE: GPL-2.0-only & bzip2-1.0.4

PACKAGE NAME: busybox-udhcpc
PACKAGE VERSION: 1.35.0
RECIPE NAME: busybox
LICENSE: GPL-2.0-only & bzip2-1.0.4
PACKAGE NAME: busybox-udhcpd
PACKAGE VERSION: 1.35.0
RECIPE NAME: busybox
LICENSE: GPL-2.0-only & bzip2-1.0.4

PACKAGE NAME: coreutils
PACKAGE VERSION: 9.0
RECIPE NAME: coreutils
LICENSE: GPL-3.0-or-later

PACKAGE NAME: coreutils-stdbuf
PACKAGE VERSION: 9.0
RECIPE NAME: coreutils
LICENSE: GPL-3.0-or-later
PACKAGE NAME: eudev
PACKAGE VERSION: 3.2.10
RECIPE NAME: eudev
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later

PACKAGE NAME: glibc
PACKAGE VERSION: 2.35
RECIPE NAME: glibc
LICENSE: GPL-2.0-only & LGPL-2.1-only

PACKAGE NAME: gmp
PACKAGE VERSION: 6.2.1
RECIPE NAME: gmp
LICENSE: GPL-2.0-or-later | LGPL-3.0-or-later

PACKAGE NAME: gnupg
PACKAGE VERSION: 2.3.4
RECIPE NAME: gnupg
LICENSE: GPL-3.0-only & LGPL-3.0-only

PACKAGE NAME: gnupg-gpg
PACKAGE VERSION: 2.3.4
RECIPE NAME: gnupg
LICENSE: GPL-3.0-only & LGPL-3.0-only

PACKAGE NAME: gnutls
PACKAGE VERSION: 3.7.4
RECIPE NAME: gnutls
LICENSE: LGPL-2.1-or-later

PACKAGE NAME: init-ifupdown
PACKAGE VERSION: 1.0
RECIPE NAME: init-ifupdown
LICENSE: GPL-2.0-only

PACKAGE NAME: init-system-helpers-service
PACKAGE VERSION: 1.62
RECIPE NAME: init-system-helpers
LICENSE: BSD-3-Clause & GPL-2.0-only

PACKAGE NAME: initscripts
PACKAGE VERSION: 1.0
RECIPE NAME: initscripts
LICENSE: GPL-2.0-only

PACKAGE NAME: initscripts-functions
PACKAGE VERSION: 1.0
RECIPE NAME: initscripts
LICENSE: GPL-2.0-only

PACKAGE NAME: inotify-tools
PACKAGE VERSION: 3.22.1.0
RECIPE NAME: inotify-tools
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-base
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-devicetree
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-image
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-image-fitimage
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-auth-rpcgss-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-cdc-acm-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-dwc3-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-dwc3-omap-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-ehci-hcd-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-ehci-omap-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-ehci-platform-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-g-ether-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-g-mass-storage-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-irq-pruss-intc-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-libcomposite-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-mq-deadline-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-nfsv2-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-nfsv3-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-nfsv4-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-ohci-hcd-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-oid-registry-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-phy-generic-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-pru-rproc-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-prueth-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-pruss-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-roles-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-scsi-mod-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-sd-mod-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-tun-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-u-ether-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-uas-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-udc-core-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-uio-module-drv-5.4.28-rt19+
PACKAGE VERSION: 2.3.1.0+gitAUTOINC+78c535afe8
RECIPE NAME: uio-module-drv
LICENSE: BSD-3-Clause

PACKAGE NAME: kernel-module-uio-pruss-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-usb-common-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-usb-f-ecm-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-usb-f-ecm-subset-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-usb-f-eem-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-usb-f-mass-storage-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-usb-f-rndis-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-usb-otg-fsm-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-usb-storage-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-usbcore-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-xhci-hcd-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-module-xhci-plat-hcd-5.4.28-rt19+
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kernel-modules
PACKAGE VERSION: 5.4.28+gitAUTOINC+c3dd64420d
RECIPE NAME: linux-ti-staging-rt
LICENSE: GPL-2.0-only

PACKAGE NAME: kmod
PACKAGE VERSION: 29
RECIPE NAME: kmod
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later

PACKAGE NAME: ldconfig
PACKAGE VERSION: 2.35
RECIPE NAME: glibc
LICENSE: GPL-2.0-only & LGPL-2.1-only

PACKAGE NAME: ldd
PACKAGE VERSION: 2.35
RECIPE NAME: glibc
LICENSE: GPL-2.0-only & LGPL-2.1-only

PACKAGE NAME: libarchive
PACKAGE VERSION: 3.6.1
RECIPE NAME: libarchive
LICENSE: BSD-2-Clause

PACKAGE NAME: libassuan
PACKAGE VERSION: 2.5.5
RECIPE NAME: libassuan
LICENSE: LGPL-2.1-or-later

PACKAGE NAME: libattr
PACKAGE VERSION: 2.5.1
RECIPE NAME: attr
LICENSE: LGPL-2.1-or-later

PACKAGE NAME: libcap
PACKAGE VERSION: 2.63
RECIPE NAME: libcap
LICENSE: BSD-3-Clause | GPL-2.0-only

PACKAGE NAME: libcap-ng
PACKAGE VERSION: 0.8.2
RECIPE NAME: libcap-ng
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later

PACKAGE NAME: libcrypto
PACKAGE VERSION: 3.0.3
RECIPE NAME: openssl
LICENSE: Apache-2.0

PACKAGE NAME: libgcc
PACKAGE VERSION: 11.2.0
RECIPE NAME: libgcc
LICENSE: GPL-3.0-with-GCC-exception

PACKAGE NAME: libgcrypt
PACKAGE VERSION: 1.9.4
RECIPE NAME: libgcrypt
LICENSE: LGPL-2.1-or-later

PACKAGE NAME: libgpg-error
PACKAGE VERSION: 1.44
RECIPE NAME: libgpg-error
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later

PACKAGE NAME: libidn2
PACKAGE VERSION: 2.3.2
RECIPE NAME: libidn2
LICENSE: (GPL-2.0-or-later | LGPL-3.0-only) & Unicode-DFS-2016

PACKAGE NAME: libinotifytools
PACKAGE VERSION: 3.22.1.0
RECIPE NAME: inotify-tools
LICENSE: GPL-2.0-only

PACKAGE NAME: libkmod
PACKAGE VERSION: 29
RECIPE NAME: kmod
LICENSE: LGPL-2.1-or-later

PACKAGE NAME: libksba
PACKAGE VERSION: 1.6.0
RECIPE NAME: libksba
LICENSE: GPL-2.0-or-later | LGPL-3.0-or-later

PACKAGE NAME: libopkg
PACKAGE VERSION: 0.5.0
RECIPE NAME: opkg
LICENSE: GPL-2.0-or-later

PACKAGE NAME: libsolv
PACKAGE VERSION: 0.7.22
RECIPE NAME: libsolv
LICENSE: BSD-3-Clause

PACKAGE NAME: libstdc++
PACKAGE VERSION: 11.2.0
RECIPE NAME: gcc-runtime
LICENSE: GPL-3.0-with-GCC-exception

PACKAGE NAME: libubootenv
PACKAGE VERSION: 0.3.2
RECIPE NAME: libubootenv
LICENSE: LGPL-2.1-only

PACKAGE NAME: libubootenv-bin
PACKAGE VERSION: 0.3.2
RECIPE NAME: libubootenv
LICENSE: LGPL-2.1-only

PACKAGE NAME: libunistring
PACKAGE VERSION: 1.0
RECIPE NAME: libunistring
LICENSE: LGPL-3.0-or-later | GPL-2.0-or-later

PACKAGE NAME: libxcrypt
PACKAGE VERSION: 4.4.28
RECIPE NAME: libxcrypt
LICENSE: LGPL-2.1-only

PACKAGE NAME: libzstd
PACKAGE VERSION: 1.5.2
RECIPE NAME: zstd
LICENSE: BSD-3-Clause & GPL-2.0-only

PACKAGE NAME: lmsensors-config-libsensors
PACKAGE VERSION: 1.0
RECIPE NAME: lmsensors-config
LICENSE: MIT

PACKAGE NAME: lmsensors-libsensors
PACKAGE VERSION: 3.6.0
RECIPE NAME: lmsensors
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later

PACKAGE NAME: lmsensors-sensors
PACKAGE VERSION: 3.6.0
RECIPE NAME: lmsensors
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later

PACKAGE NAME: lsof
PACKAGE VERSION: 4.94.0
RECIPE NAME: lsof
LICENSE: Spencer-94

PACKAGE NAME: lzo
PACKAGE VERSION: 2.10
RECIPE NAME: lzo
LICENSE: GPL-2.0-or-later

PACKAGE NAME: memtool
PACKAGE VERSION: 2018.03.0
RECIPE NAME: memtool
LICENSE: GPLv2

PACKAGE NAME: modutils-initscripts
PACKAGE VERSION: 1.0
RECIPE NAME: modutils-initscripts
LICENSE: MIT

PACKAGE NAME: mtd-utils
PACKAGE VERSION: 2.1.4
RECIPE NAME: mtd-utils
LICENSE: GPL-2.0-or-later

PACKAGE NAME: mtd-utils-ubifs
PACKAGE VERSION: 2.1.4
RECIPE NAME: mtd-utils
LICENSE: GPL-2.0-or-later

PACKAGE NAME: ncurses-libncurses
PACKAGE VERSION: 6.3
RECIPE NAME: ncurses
LICENSE: MIT

PACKAGE NAME: ncurses-libncursesw
PACKAGE VERSION: 6.3
RECIPE NAME: ncurses
LICENSE: MIT

PACKAGE NAME: ncurses-libtinfo
PACKAGE VERSION: 6.3
RECIPE NAME: ncurses
LICENSE: MIT

PACKAGE NAME: ncurses-terminfo-base
PACKAGE VERSION: 6.3
RECIPE NAME: ncurses
LICENSE: MIT

PACKAGE NAME: netbase
PACKAGE VERSION: 6.3
RECIPE NAME: netbase
LICENSE: GPL-2.0-only

PACKAGE NAME: nettle
PACKAGE VERSION: 3.7.3
RECIPE NAME: nettle
LICENSE: LGPL-3.0-or-later | GPL-2.0-or-later

PACKAGE NAME: npth
PACKAGE VERSION: 1.6
RECIPE NAME: npth
LICENSE: LGPL-2.0-or-later

PACKAGE NAME: openssh
PACKAGE VERSION: 8.9p1
RECIPE NAME: openssh
LICENSE: BSD-2-Clause & BSD-3-Clause & ISC & MIT

PACKAGE NAME: openssh-keygen
PACKAGE VERSION: 8.9p1
RECIPE NAME: openssh
LICENSE: BSD-2-Clause & BSD-3-Clause & ISC & MIT

PACKAGE NAME: openssh-scp
PACKAGE VERSION: 8.9p1
RECIPE NAME: openssh
LICENSE: BSD-2-Clause & BSD-3-Clause & ISC & MIT

PACKAGE NAME: openssh-ssh
PACKAGE VERSION: 8.9p1
RECIPE NAME: openssh
LICENSE: BSD-2-Clause & BSD-3-Clause & ISC & MIT

PACKAGE NAME: openssh-sshd
PACKAGE VERSION: 8.9p1
RECIPE NAME: openssh
LICENSE: BSD-2-Clause & BSD-3-Clause & ISC & MIT

PACKAGE NAME: openssl-conf
PACKAGE VERSION: 3.0.3
RECIPE NAME: openssl
LICENSE: Apache-2.0

PACKAGE NAME: openssl-ossl-module-legacy
PACKAGE VERSION: 3.0.3
RECIPE NAME: openssl
LICENSE: Apache-2.0

PACKAGE NAME: opkg
PACKAGE VERSION: 0.5.0
RECIPE NAME: opkg
LICENSE: GPL-2.0-or-later

PACKAGE NAME: opkg-arch-config
PACKAGE VERSION: 1.0
RECIPE NAME: opkg-arch-config
LICENSE: MIT

PACKAGE NAME: os-release
PACKAGE VERSION: 1.0
RECIPE NAME: os-release
LICENSE: MIT

PACKAGE NAME: packagegroup-core-boot
PACKAGE VERSION: 1.0
RECIPE NAME: packagegroup-core-boot
LICENSE: MIT

PACKAGE NAME: packagegroup-core-ssh-openssh
PACKAGE VERSION: 1.0
RECIPE NAME: packagegroup-core-ssh-openssh
LICENSE: MIT

PACKAGE NAME: pinentry
PACKAGE VERSION: 1.2.0
RECIPE NAME: pinentry
LICENSE: GPL-2.0-only

PACKAGE NAME: procps
PACKAGE VERSION: 3.3.17
RECIPE NAME: procps
LICENSE: GPL-2.0-or-later & LGPL-2.0-or-later
PACKAGE NAME: procps-lib
PACKAGE VERSION: 3.3.17
RECIPE NAME: procps
LICENSE: GPL-2.0-or-later & LGPL-2.0-or-later

PACKAGE NAME: procps-ps
PACKAGE VERSION: 3.3.17
RECIPE NAME: procps
LICENSE: GPL-2.0-or-later & LGPL-2.0-or-later

PACKAGE NAME: procps-sysctl
PACKAGE VERSION: 3.3.17
RECIPE NAME: procps
LICENSE: GPL-2.0-or-later & LGPL-2.0-or-later

PACKAGE NAME: prueth-fw
PACKAGE VERSION: 2022.01
RECIPE NAME: prueth-fw
LICENSE: TI-TFL

PACKAGE NAME: pruhsr-fw
PACKAGE VERSION: 2022.01
RECIPE NAME: pruhsr-fw
LICENSE: TI-TFL

PACKAGE NAME: pruprp-fw
PACKAGE VERSION: 2022.01
RECIPE NAME: pruprp-fw
LICENSE: TI-TFL

PACKAGE NAME: readline
PACKAGE VERSION: 8.1.2
RECIPE NAME: readline
LICENSE: GPL-3.0-or-later

PACKAGE NAME: run-postinsts
PACKAGE VERSION: 1.0
RECIPE NAME: run-postinsts
LICENSE: MIT

PACKAGE NAME: shadow
PACKAGE VERSION: 4.11.1
RECIPE NAME: shadow
LICENSE: BSD-3-Clause

PACKAGE NAME: shadow-base
PACKAGE VERSION: 4.11.1
RECIPE NAME: shadow
LICENSE: BSD-3-Clause

PACKAGE NAME: shadow-securetty
PACKAGE VERSION: 4.6
RECIPE NAME: shadow-securetty
LICENSE: MIT

PACKAGE NAME: sysvinit
PACKAGE VERSION: 3.01
RECIPE NAME: sysvinit
LICENSE: GPL-2.0-or-later

PACKAGE NAME: sysvinit-inittab
PACKAGE VERSION: 2.88dsf
RECIPE NAME: sysvinit-inittab
LICENSE: GPL-2.0-only

PACKAGE NAME: sysvinit-pidof
PACKAGE VERSION: 3.01
RECIPE NAME: sysvinit
LICENSE: GPL-2.0-or-later

PACKAGE NAME: tar
PACKAGE VERSION: 1.34
RECIPE NAME: tar
LICENSE: GPL-3.0-only

PACKAGE NAME: uio-module-drv
PACKAGE VERSION: 2.3.1.0+gitAUTOINC+78c535afe8
RECIPE NAME: uio-module-drv
LICENSE: BSD-3-Clause

PACKAGE NAME: update-alternatives-opkg
PACKAGE VERSION: 0.5.0
RECIPE NAME: opkg-utils
LICENSE: GPL-2.0-or-later

PACKAGE NAME: update-rc.d
PACKAGE VERSION: 0.8
RECIPE NAME: update-rc.d
LICENSE: GPL-2.0-or-later

PACKAGE NAME: util-linux
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-addpart
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-agetty
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-blkdiscard
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-blkid
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-blkzone
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-blockdev
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-cal
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-cfdisk
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-chcpu
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-chmem
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-choom
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-chrt
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-col
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-colcrt
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-colrm
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-column
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-ctrlaltdel
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-delpart
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-dmesg
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-eject
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-fallocate
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-fdisk
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-fincore
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-findfs
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-findmnt
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-flock
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-fsck
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-fsck.cramfs
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-fsfreeze
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-fstrim
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-getopt
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-hardlink
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-hexdump
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-hwclock
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-ionice
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-ipcmk
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-ipcrm
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-ipcs
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-irqtop
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-isosize
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-kill
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-last
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-ldattach
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-libblkid
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: LGPL-2.1-or-later

PACKAGE NAME: util-linux-libfdisk
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: LGPL-2.1-or-later

PACKAGE NAME: util-linux-libmount
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: LGPL-2.1-or-later

PACKAGE NAME: util-linux-libsmartcols
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: LGPL-2.1-or-later

PACKAGE NAME: util-linux-libuuid
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux-libuuid
LICENSE: BSD-3-Clause

PACKAGE NAME: util-linux-logger
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-look
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-losetup
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-lsblk
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-lscpu
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-lsipc
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-lsirq
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-lslocks
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-lslogins
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-lsmem
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-lsns
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-mcookie
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-mesg
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-mkfs
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-mkfs.cramfs
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-mkswap
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-more
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-mount
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-mountpoint
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-namei
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-nologin
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-nsenter
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-partx
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-pivot-root
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-prlimit
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-readprofile
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-rename
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-renice
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-resizepart
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-rev
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-rfkill
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-rtcwake
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-script
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-scriptlive
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-scriptreplay
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-setarch
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-setpriv
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-setsid
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-setterm
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-sfdisk
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-sulogin
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-swaplabel
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-swapoff
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-swapon
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-switch-root
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-taskset
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-uclampset
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-ul
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-umount
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-unshare
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-utmpdump
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-uuidd
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-uuidgen
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-uuidparse
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-wall
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-wdctl
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-whereis
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-wipefs
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-write
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: util-linux-zramctl
PACKAGE VERSION: 2.37.4
RECIPE NAME: util-linux
LICENSE: GPL-2.0-or-later & LGPL-2.1-or-later & BSD-3-Clause & BSD-4-Clause

PACKAGE NAME: zlog
PACKAGE VERSION: 1.2.15
RECIPE NAME: zlog
LICENSE: LGPL-2.1-only

15.2.2 Others

NAME: U-Boot
VERSION: v2020.01
LICENSE: GPL-2.0-or-later

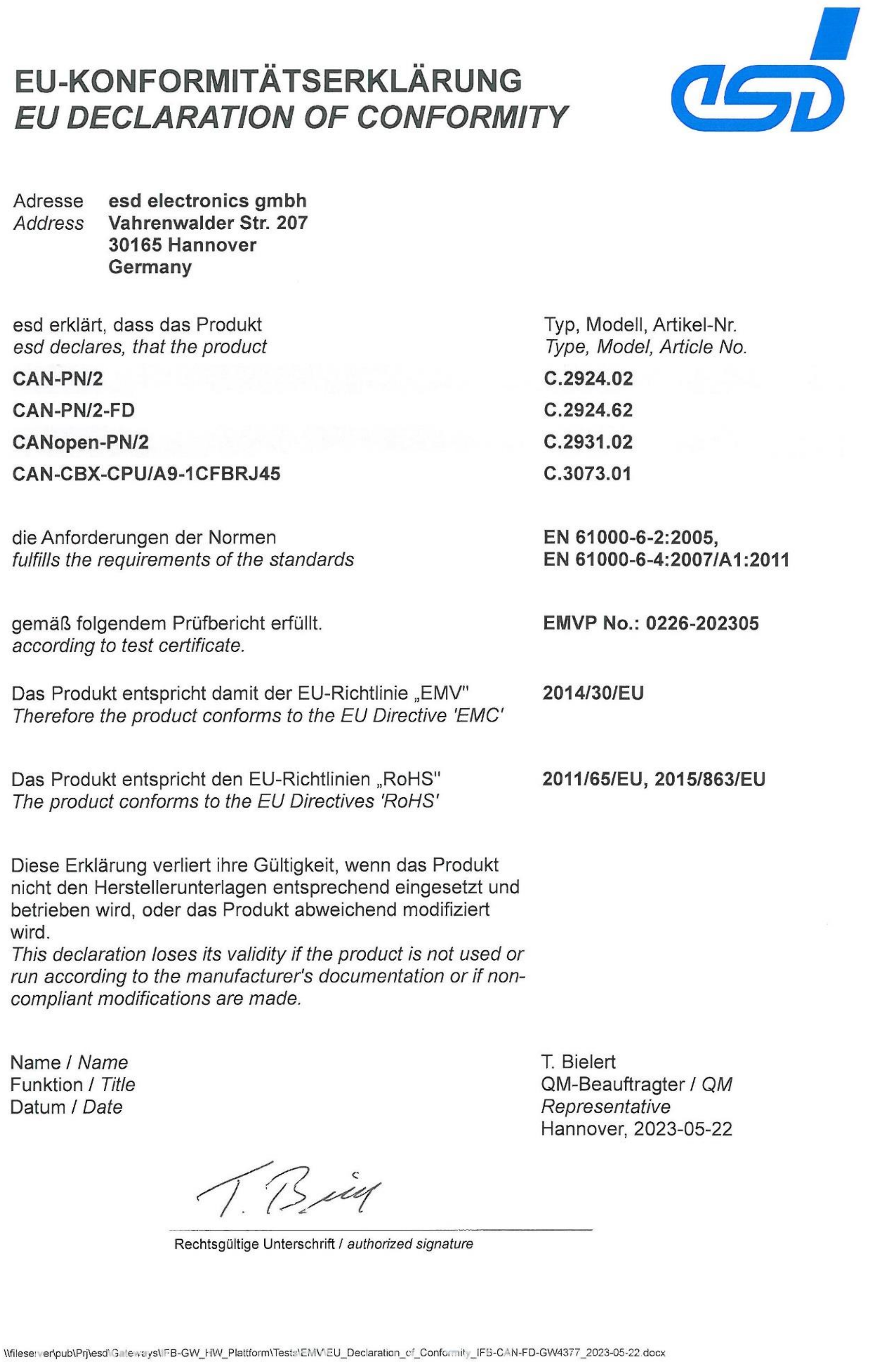
NAME: PROFINET Stack-Lizenzbedingungen
VERSION: 2011-08-01
LICENSE: License Conditions for Siemens Profinet Stack

15.2.3 Open Source Software Copy

You may obtain a copy of the source code, if and as required under the license by sending a mail to oss-compliance@esd.eu

You may also obtain a copy of the source code, if and as required under the license, by sending a check or money of EUR 25.00 to:
esd electronics gmbh
Vahrenwalder Str. 207
30165 Hannover, Germany

16 Declaration of Conformity



EU-KONFORMITÄTSERKLÄRUNG
EU DECLARATION OF CONFORMITY

Adresse *Address*	**esd electronics gmbh** **Vahrenwalder Str. 207** **30165 Hannover** **Germany**

esd erklärt, dass das Produkt *esd declares, that the product*	Typ, Modell, Artikel-Nr. *Type, Model, Article No.*
CAN-PN/2	**C.2924.02**
CAN-PN/2-FD	**C.2924.62**
CANopen-PN/2	**C.2931.02**
CAN-CBX-CPU/A9-1CFBRJ45	**C.3073.01**

die Anforderungen der Normen *fulfills the requirements of the standards*	**EN 61000-6-2:2005,** **EN 61000-6-4:2007/A1:2011**
gemäß folgendem Prüfbericht erfüllt. *according to test certificate.*	**EMVP No.: 0226-202305**
Das Produkt entspricht damit der EU-Richtlinie „EMV" *Therefore the product conforms to the EU Directive 'EMC'*	**2014/30/EU**
Das Produkt entspricht den EU-Richtlinien „RoHS" *The product conforms to the EU Directives 'RoHS'*	**2011/65/EU, 2015/863/EU**

Diese Erklärung verliert ihre Gültigkeit, wenn das Produkt nicht den Herstellerunterlagen entsprechend eingesetzt und betrieben wird, oder das Produkt abweichend modifiziert wird.
This declaration loses its validity if the product is not used or run according to the manufacturer's documentation or if non-compliant modifications are made.

Name / *Name*	T. Bielert
Funktion / *Title*	QM-Beauftragter / *QM Representative*
Datum / *Date*	Hannover, 2023-05-22

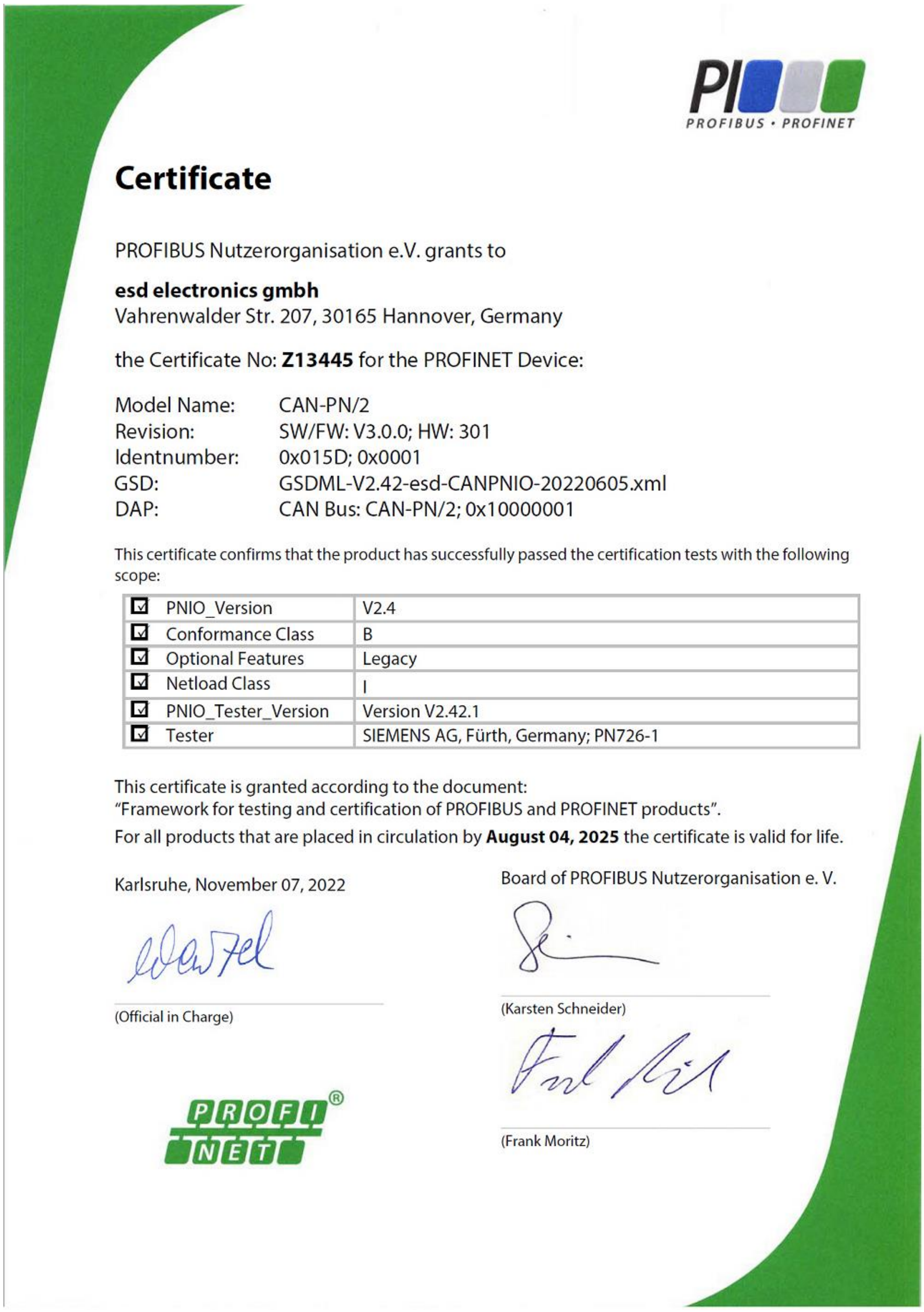
[illegible]

Rechtsgültige Unterschrift / *authorized signature*

\\fileserver\pub\Prj\esd\Gateways\IFB-GW_HW_Plattform\Tests\EMV\EU_Declaration_of_Conformity_IFB-CAN-FD-GW4377_2023-05-22.docx

17 PNO Certificates

17.1 CAN-PN/2 (C.2924.02)



PI PROFIBUS • PROFINET

Certificate

PROFIBUS Nutzerorganisation e.V. grants to

esd electronics gmbh
Vahrenwalder Str. 207, 30165 Hannover, Germany

the Certificate No: **Z13445** for the PROFINET Device:

Model Name:	CAN-PN/2
Revision:	SW/FW: V3.0.0; HW: 301
Identnumber:	0x015D; 0x0001
GSD:	GSDML-V2.42-esd-CANPNIO-20220605.xml
DAP:	CAN Bus: CAN-PN/2; 0x10000001

This certificate confirms that the product has successfully passed the certification tests with the following scope:

☑	PNIO_Version	V2.4
☑	Conformance Class	B
☑	Optional Features	Legacy
☑	Netload Class	I
☑	PNIO_Tester_Version	Version V2.42.1
☑	Tester	SIEMENS AG, Fürth, Germany; PN726-1

This certificate is granted according to the document:
"Framework for testing and certification of PROFIBUS and PROFINET products".
For all products that are placed in circulation by **August 04, 2025** the certificate is valid for life.

Karlsruhe, November 07, 2022

[illegible]

(Official in Charge)

Board of PROFIBUS Nutzerorganisation e. V.

[illegible]

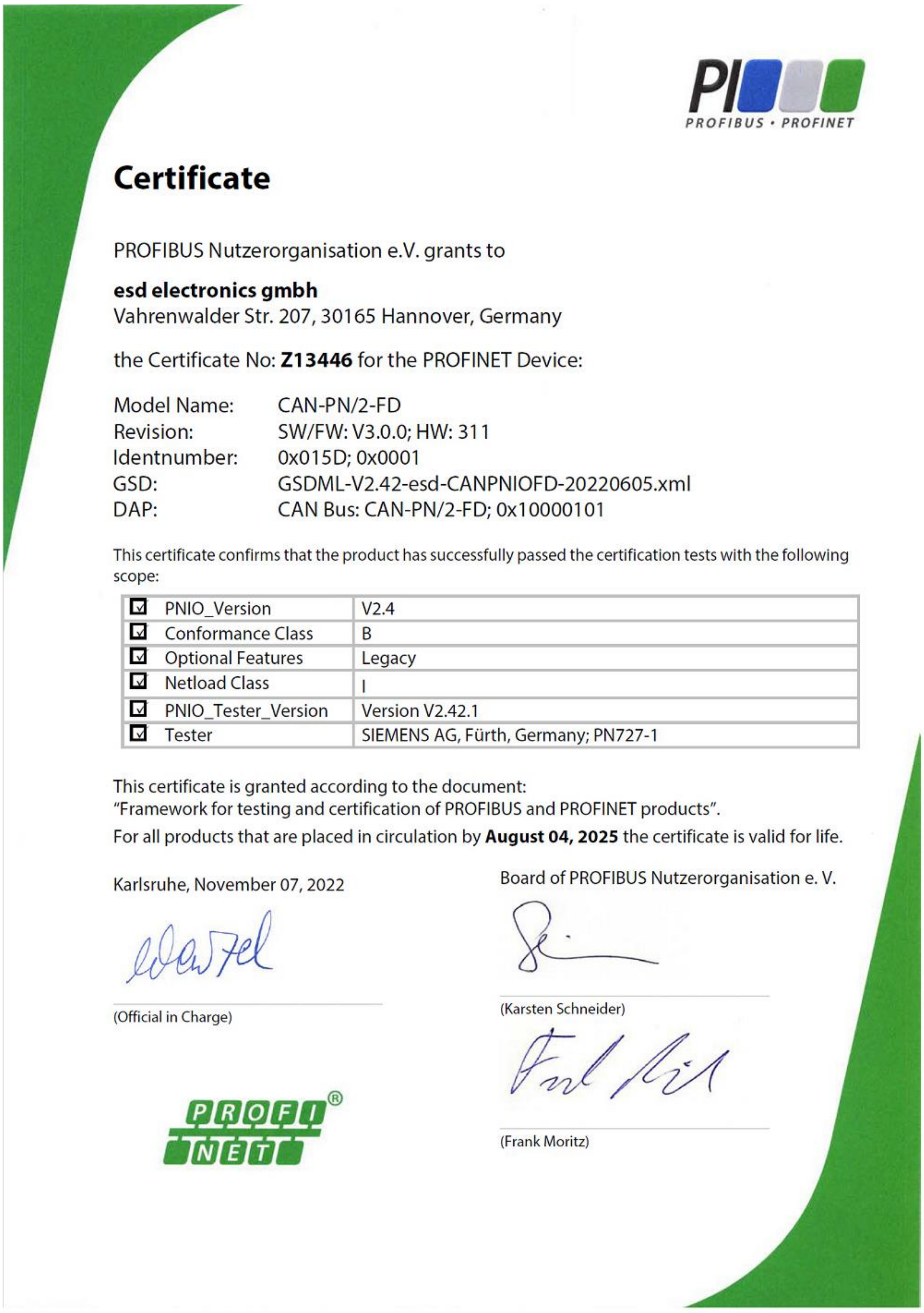
(Karsten Schneider)

[illegible]

(Frank Moritz)

PROFI NET®

17.2 CAN-PN/2-FD (C.2924.62)



PROFIBUS • PROFINET

Certificate

PROFIBUS Nutzerorganisation e.V. grants to

esd electronics gmbh
Vahrenwalder Str. 207, 30165 Hannover, Germany

the Certificate No: **Z13446** for the PROFINET Device:

Model Name:	CAN-PN/2-FD
Revision:	SW/FW: V3.0.0; HW: 311
Identnumber:	0x015D; 0x0001
GSD:	GSDML-V2.42-esd-CANPNIOFD-20220605.xml
DAP:	CAN Bus: CAN-PN/2-FD; 0x10000101

This certificate confirms that the product has successfully passed the certification tests with the following scope:

☑	PNIO_Version	V2.4
☑	Conformance Class	B
☑	Optional Features	Legacy
☑	Netload Class	I
☑	PNIO_Tester_Version	Version V2.42.1
☑	Tester	SIEMENS AG, Fürth, Germany; PN727-1

This certificate is granted according to the document:
"Framework for testing and certification of PROFIBUS and PROFINET products".
For all products that are placed in circulation by **August 04, 2025** the certificate is valid for life.

Karlsruhe, November 07, 2022

[illegible]

(Official in Charge)

Board of PROFIBUS Nutzerorganisation e. V.

[illegible]

(Karsten Schneider)

[illegible]

(Frank Moritz)

PROFI NET®

18 Order Information

18.1 Hardware

Type	Properties	Order No.
CAN-PN/2	High-performance **CAN CC** to PROFINET-IO device gateway with wide range of configuration options. CAN CC interface according to ISO-11898 with galvanic isolation, PROFINET physical layer 100BASE-TX according to IEEE802.3 with integrated switch for DIN rail mounting. Comprehensive debugging options with CAN diagnostic software CANreal via USB interface.	C.2924.02
CAN-PN/2-FD	High-performance **CAN FD** to PROFINET-IO device gateway with a wide range of configuration options. CAN FD interface according to ISO-11898 with galvanic isolation, PROFINET physical layer 100BASE-TX according to IEEE802.3 with integrated switch for DIN rail mounting. Extensive debugging possibilities with CAN diagnostic software CANreal via USB interface.	C.2924.62

Table 64: Order information hardware

18.2 Manuals

PDF Manuals
For the availability of the manuals see the table below.
Please download the manuals as PDF documents from our esd website https://www.esd.eu for free.

Manuals		Order No.
CAN-PN/2-ME	Hardware and software manual for CAN-PN/2 and CAN-PN/2-FD in English	C.2924.21
CAN-API-ME	NTCAN-API, Part 1: Application Developers Manual NTCAN-API, Part 2: Driver Installation Guide	C.2001.21

Table 65: Available Manuals

Printed Manuals
If you need a printout of the manual additionally, please contact our sales team (sales@esd.eu) for a quotation. Printed manuals may be ordered for a fee.