WinAC CAN Basis driver

COM168 extension board
Microbox IPC427B/C on-board CAN
Nanobox IPC227D on-board CAN

V1.4.0 • July 2012

Applikationen & Tools

Answers for industry.

SIEMENS

Industry Automation and Drives Technologies Service & Support Portal

This article is taken from the Service Portal of Siemens AG, Industry Automation and Drives Technologies. The following link takes you directly to the download page of this document.

http://support.automation.siemens.com/WW/view/en/48281517

If you have any questions concerning this document please e-mail us to the following address:

online-support.automation@siemens.com

applications.aud.koe.nrh.rd@siemens.com

S

SIMATIC
WinAC CAN Basic Driver

Operator Manual

Warranty and Liability

Note

The Application Examples are not binding and do not claim to be complete regarding the circuits shown, equipping and any eventuality. The Application Examples do not represent customer-specific solutions. They are only intended to provide support for typical applications. You are responsible for ensuring that the described products are used correctly. These application examples do not relieve you of the responsibility to use safe practices in application, installation, operation and maintenance. When using these Application Examples, you recognize that we cannot be made liable for any damage/claims beyond the liability clause described. We reserve the right to make changes to these Application Examples at any time without prior notice.
If there are any deviations between the recommendations provided in these application examples and other Siemens publications – e.g. Catalogs – the contents of the other documents have priority.

We do not accept any liability for the information contained in this document.

Any claims against us – based on whatever legal reason – resulting from the use of the examples, information, programs, engineering and performance data etc., described in this Application Example shall be excluded. Such an exclusion shall not apply in the case of mandatory liability, e.g. under the German Product Liability Act ("Produkthaftungsgesetz"), in case of intent, gross negligence, or injury of life, body or health, guarantee for the quality of a product, fraudulent concealment of a deficiency or breach of a condition which goes to the root of the contract ("wesentliche Vertragspflichten"). The damages for a breach of a substantial contractual obligation are, however, limited to the foreseeable damage, typical for the type of contract, except in the event of intent or gross negligence or injury to life, body or health. The above provisions do not imply a change of the burden of proof to your detriment.

Table of Contents

1 Automation Task

1.1 Overview

There are projects with SIMATIC PLCs and CAN peripheral components. When using industrial PCs there are some possibilities to integrate CAN interfaces.

The present driver enables the usage of selected CAN interfaces with the Soft-PLC SIMATIC WinAC RTX.

The CAN interface is implemented as Layer-2 interface (send / receive).

The driver supports the following CAN interfaces:

- COM168 board with 2 CAN interfaces (PCI version)
- On-board CAN interface of Microbox PC427B/C (SJA1000) *1)
- On-board CAN interface of Nanobox IPC227D (Bosch C_CAN) *2)

*1) A CAN on-board interface is an option when ordering the Microbox PC427B/C.

*2) A CAN on-board interface is an option when ordering the Nanobox IPC227D.

1.2 Needed Knowledge

To understand this document the knowledge of the following information is needed:

- Simatic WinAC RTX
- Simatic Manager *or* TIA Portal
- CAN protocol, layer 2

1.3 Required Hardware and Software Components

The application was generated with the following components:

Hardware components

- Simatic Microbox PC 427B (Celeron M, 900 MHz, 512 MB RAM, 2 GB Flash, CAN on-board) with CAN on-board interface
  Windows CP embedded SP2
- COM168 card (PCI104)
- Nanobox IPC227D with CAN on-board interface
  WES7 (Windows Embedded Standard 7)

NOTICE **The older HW revision of COM168 in PC/104+ form is not supported!**

Standard software components

- WinAC RTX 2010 Update 2
- Step7 V5.5

or

- TIA Portal V11 SP2

NOTICE **The driver for the Nanobox IPC227D CAN on-board interface is approved for WinAC RTX 2010 Update 2!**

Sample files and projects

The following list includes all files and projects that are used in this example.

Table 1-1 Files of this driver package

Component	Note
Doc\	This document in English and German
Drivers\	Realtime driver for all supported CAN interface types incl. **Setup_xxx.bat** file for installation on WinAC computer
S7_V5x_Example\	Step7 V5.x example projects
S7_V11_Example\	Step7 V11 (TIA Portal) example projects
Tools\	Helper tools (e.g. COM168-Scan)

2 Automation Solution

2.1 Functional range

The following functions are supported by the WinAC CAN Basis driver:

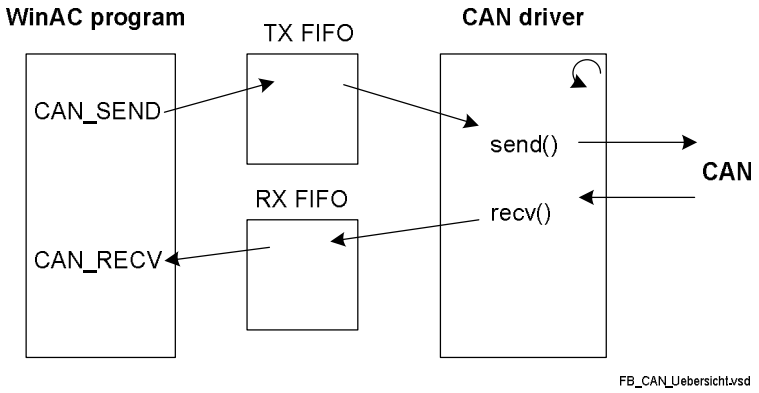
- Initialise CAN interface with given baud rate
- Send CAN messages
- Receive CAN messages
- Get status of CAN interface

The driver supports the pre-defined CANopen baud rates. The COM168 additional supports user-defined baud rates.

The CAN basis driver does not include filter functionality for CAN identifiers. Normally the WinAC should be the CAN controller, so it should receive all CAN messages.

The function blocks in the Step7 project do not access the CAN interface hardware directly. There is a real-time CAN driver working in background. The Step7 function blocks and the real-time driver exchange data via FIFO memory. The CAN driver puts received CAN messages into the Rx-FIFO and picks up CAN messages from the Tx-FIFO and send the messages. The function blocks of the Step7 program only access this both FIFO memories.

Figure 2-1 Overview WinAC CAN Basis driver



This de-coupling is needed because to prevent blocking OB1 when sending many CAN messages at the end of the cycle.

On the other hand there is a drawback of this de-coupling – errors like wire break is not recognized directly. The function blocks are informed about such problems two send commands after.

If the function block CAN_SEND is used very often in a short time, the CAN real-time driver sends the messages out of the buffer one after another. Thus there could be a little delay.

If the CAN real-time driver receives many CAN messages and the CAN_RECV is called rarely there are two problems: the CAN_RECV gets old telegrams and there could be an overflow of the internal RxFIFO memory (with error code).

Attention

The driver supports only one CAN hardware at one time – either the on-board CAN interface of Microbox, the on-board CAN interface of Nanobox or the COM168 card. Usage of more than one type at same time is not supported!

2.2 High level CAN protocols

The driver supports the layer 2 functions (send/receive), only. As part of the application other CAN protocols may be implemented.

It is possible to communicate with CANopen devices, too. In that case the user has to process the received CAN messages and has to compose the CANopen telegrams to be sent.

There is one interesting parameter of the CAN initialisation – the “shutdown telegram”. This telegram is sent automatically when the WinAC changes from RUN to STOP. This “shutdown telegram” could be used for the CANopen telegram “Pre-operational state”. Thus all devices would switch to “pre-operational” state immediately when the WinAC changes to STOP.

The WinAC CAN driver package includes an example with using CANopen digital E/A modules with this CAN layer 2 driver.

2.3 Version of the driver

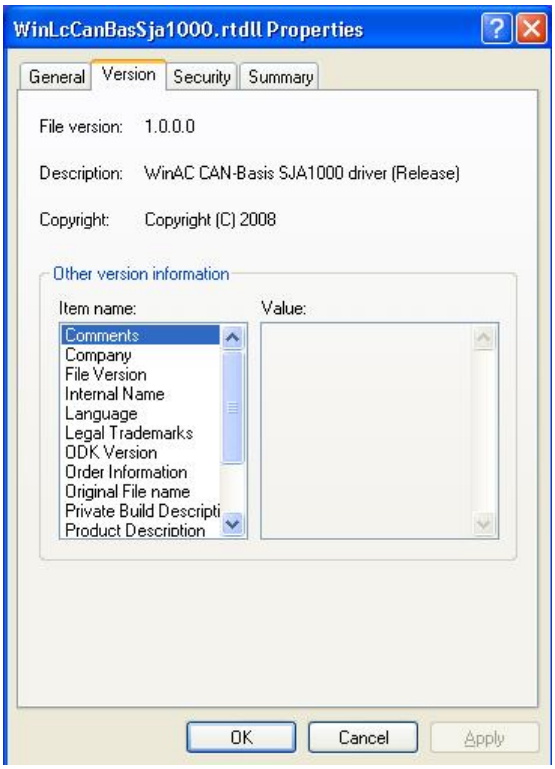
Check driver version with Windows operating system

The registered driver RTDLL is located in the system directory, e.g.

C:\Windows\Rtss\Rtdll

You can identify the version of the driver RTDLL in the file properties (Windows explorer → right click → properties)

Figure 2-2 Version of the driver RTDLL



NOTE

Because of an issue of IntervalZero RTX 9.x the file version of RTDLLs are not shown in the properties dialog under Windows 7 / WES7.

It works well under Windows XP / WES2009.

Check driver version in Step7 program

In the instance data block of CAN_INIT_sss it is possible to read the version of the driver RTDLL and the version of the Step7 driver function blocks.

C_IF.S7_VERSION	Version of Step7 driver function blocks
C_IF.DLL_VERSION	Version of driver RTDLL

3 Installation

3.1 Quickstart

Run-Time system (PC with WinAC)

- Install CAN hardware
- Setup CAN interface as RTX ressource
- Install CAN real-time driver with right install routine:
  Setup_Com168.bat → for COM168
  Setup_SJA1000.bat → for on-board CAN interface of Microbox
  Setup_CCan.bat → for on-board CAN interface of Nanobox

Engineering System (PC with Step7 / TIA Portal)

- Adapt the Step7 or TIA portal example project and use it
- Choose the right INIT-FB matching your CAN hardware in both OB1 and OB100!

NOTICE	**In the provided example applications all three different INIT function blocks are included in OB1 and OB100. You have to utilize the right one fort he used hardware.**

3.2 Installation HW “Microbox CAN on-board interface” Windows XPe

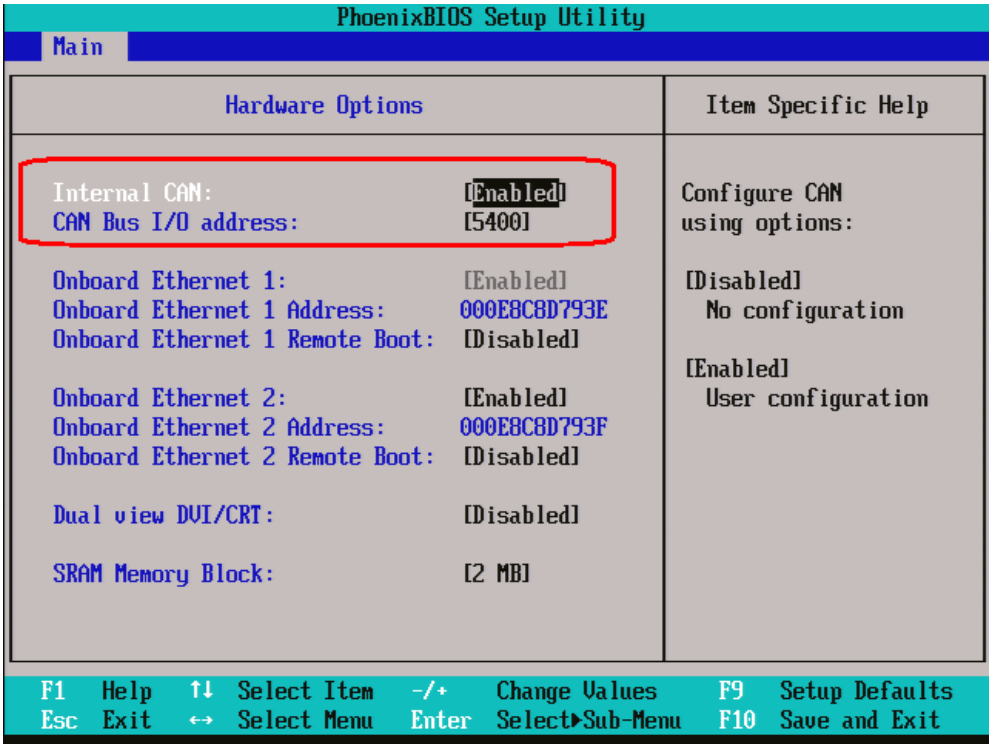
The following screen shots are created under Windows XP embedded. The processing under Windows 7 is comparable.

The on-board CAN interface of the Microbox PC427B/C is connected as ISA device. There are valid settings by default:

- IRQ. 10
- Base address: 0x5400

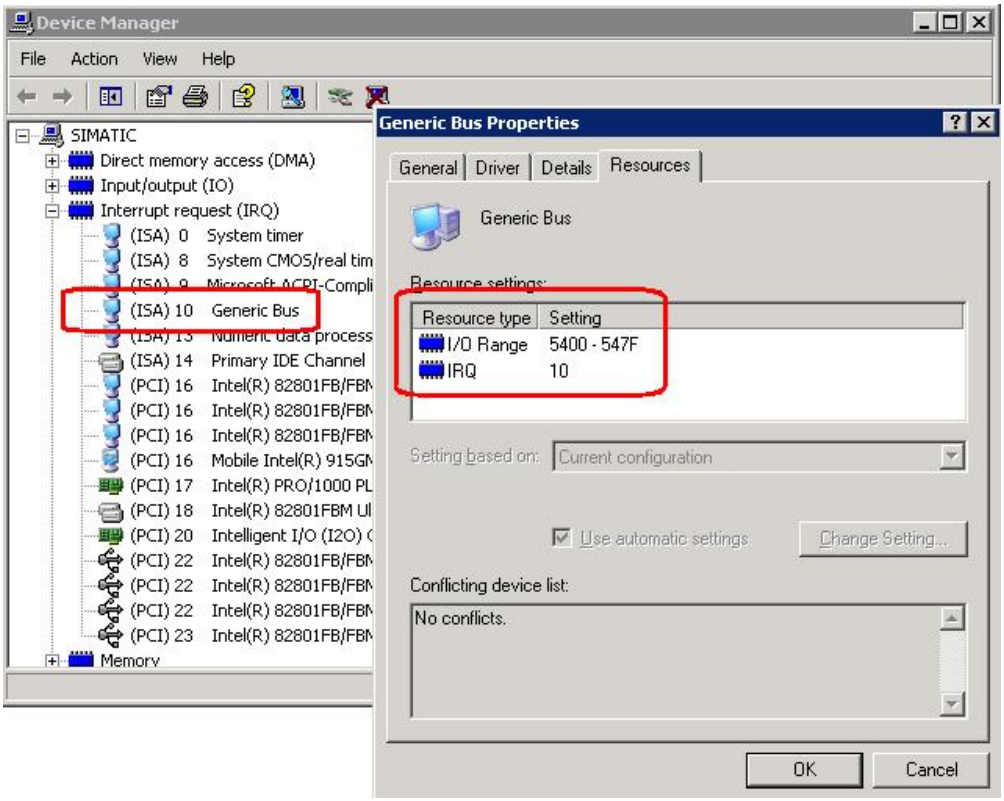
You can activate the CAN interface in the BIOS (Main / Hardware). It is possible to change the base address if needed.

Figure 3-1 BIOS Microbox PC427B



The next step is to check, if the CAN interface got an exclusive interrupt (as in default settings).

Figure 3-2 CAN on-board interface in the Windows Device Manager (Win XP)



These settings (IRQ and base address) are parameters for the driver initialisation (FB CAN_INIT_SJA1000).

3.3 Install HW “COM 168” Windows XP

The following screen shots are created under Windows XP embedded. The processing under Windows 7 is comparable.

Attention

Only the new HW revision in PCI104 is supported!

The old one in PC/104+ is not supported by the driver.

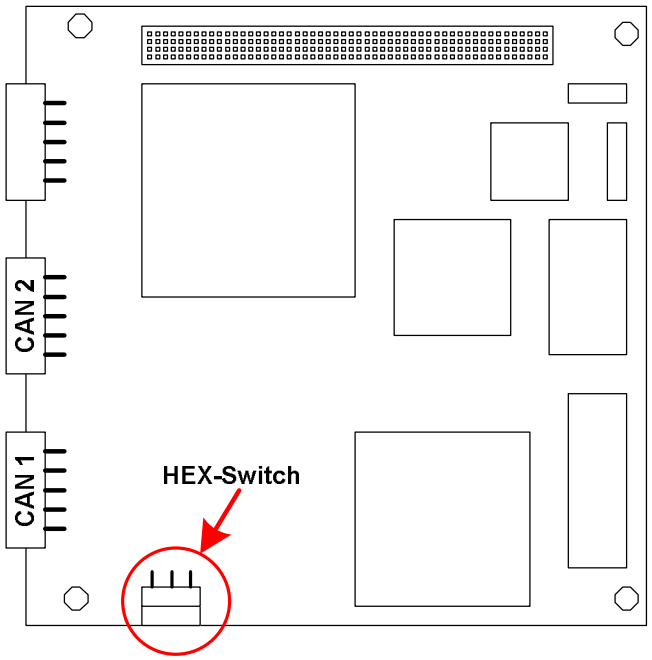
Attention

Before installing the COM168 you have to check the DIP switch (HEX switch) for the right PCI lane!

3.3.1 Configure PCI slot

The COM168 is a PCI104 device. Thus one have to adjust the used PCI slot by HEX switch (see operator manual of COM168 Rev. 2).

Figure 3-3 HEX switch for selecting PCI slot of COM168



3.3.2 Install COM168 as RTX device

The COM168 driver is realized as realtime driver for the Windows realtime extension **IntervalZero RTX** (Realtime eXTension). That’s why the CAN board has to be installed as RTX device.

Cancel Windows Plug and Play manager

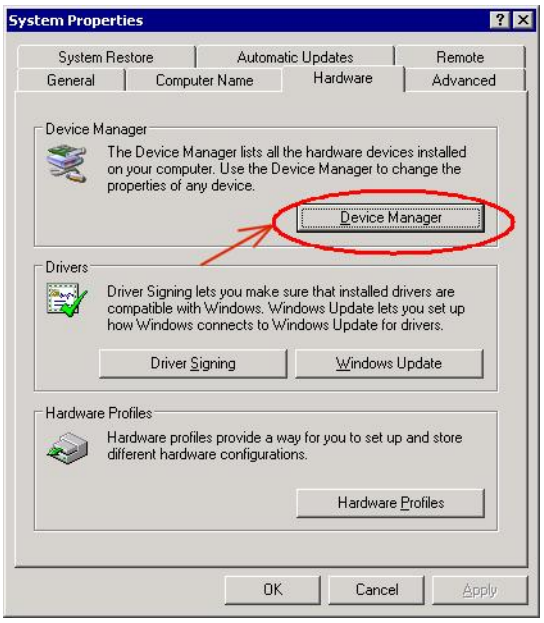
Windows Plug and Play Manager recognized the new hardware and tries to install a new driver. This dialogue has to be **canceled.**

Figure 3-4 Windows XP Plug and Play Manager

Find COM168 in the device manager

The device manager can be started over the system properties.

Figure 3-5 start device manager Win XP



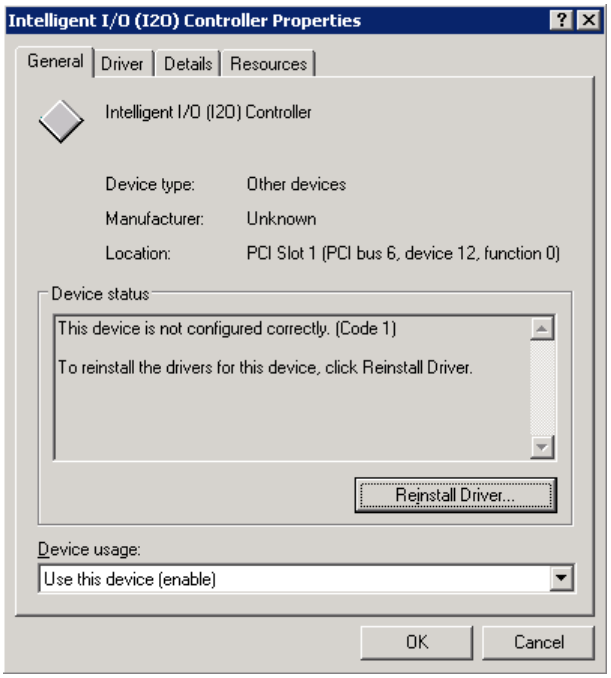
It should exist only one device with a question mark: „Other PCI Bridge Device“. This is the COM168 board.

Figure 3-6 Unknown device „Other PCI Bridge Device“

Characteristic properties of the COM168 board

On the properties page one can see characteristic properties like the slot, bus, device and function or Vendor and Device-ID.

Figure 3-7 Characteristic properties of COM168



Attention

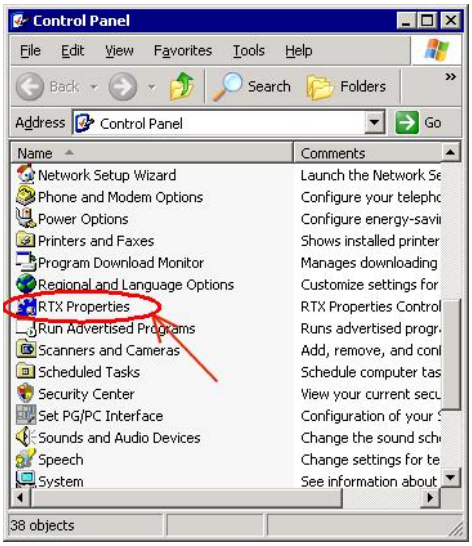
For the Microbox PC427B only PCI Slot 1 and 2 are valid!

There is no exclusive interrupt line for PCI Slot 3. Slot 4 is not allowed.

Settings in RTX Properties (Windows XP)

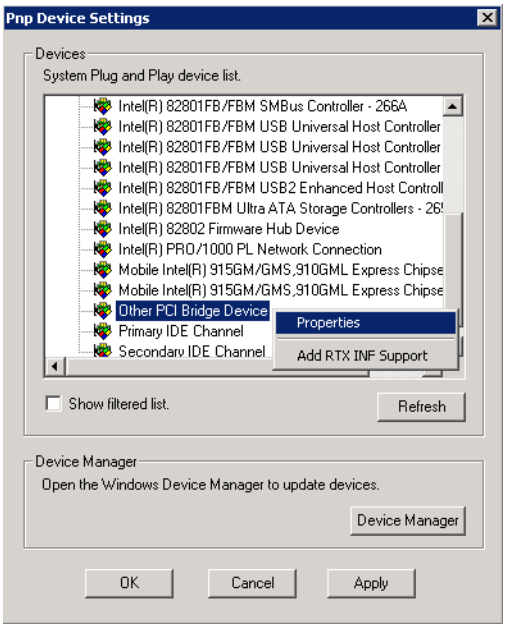
Via control panel of the Windows operating system you can reach the RTX Properties.

Figure 3-8 RTX Properties in control panel



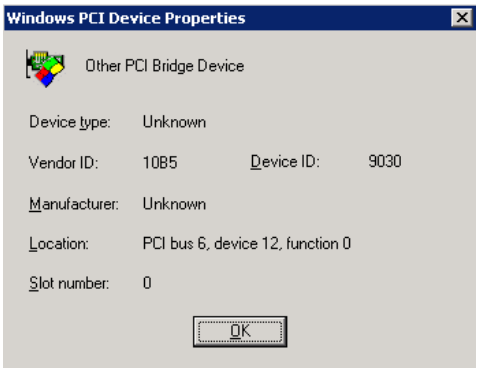
Select the tab “Plug and Play”. You have to remove the check mark **show filtered list**. Then you can see the COM168 (Other PCI Bridge Device).

Figure 3-9 Plug and Play of the RTX Properties



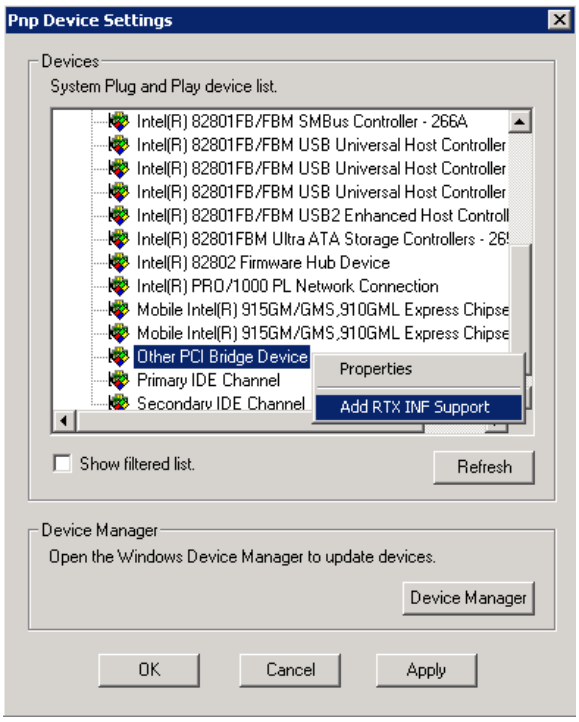
Via the device properties you can check whether it’s the right device (slot, bus, ...).

Figure 3-10 Device Properties in dialogue RTX Properties



Next you choose the device with the right mouse bottom and click on **Add RTX INF support.**

Abbildung 3-11 Adding RTX INF support to the COM168

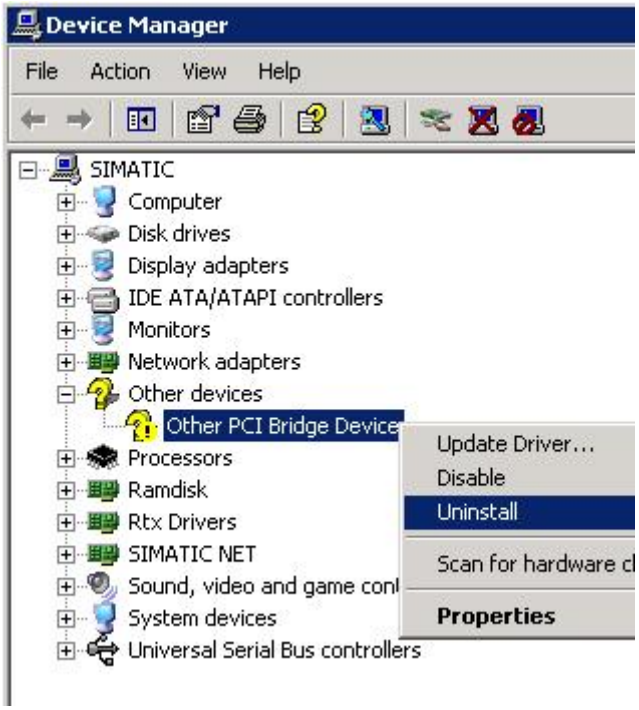


NOTICE **You must press the „Apply“ button to activate the „RTX INF Support“!**

Uninstall the device in the Windows device manager

Next you go again to the **Windows device manager** in order to uninstall the COM168 („Other PCI Bridge Device“ with question mark).

Figure 3-12 (Windows) device Uninstallation (Win XP)



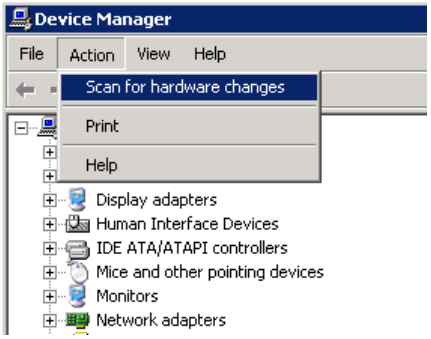
The device disappears firstly from the device manager.

Installation of device as RTX device

Afterwards you choose at the menu **Action → Scan for new Hardware** (This works only, if you have clicked into the main window of the device manager before).

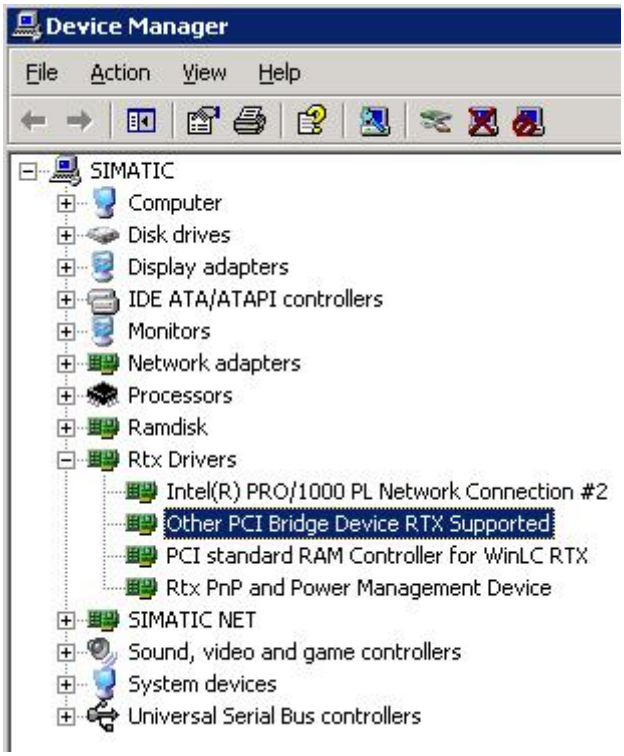
Now the COM168 will be registered as RTX device automatically.

Figure 3-13 Scan for hardware changes (Win XP)



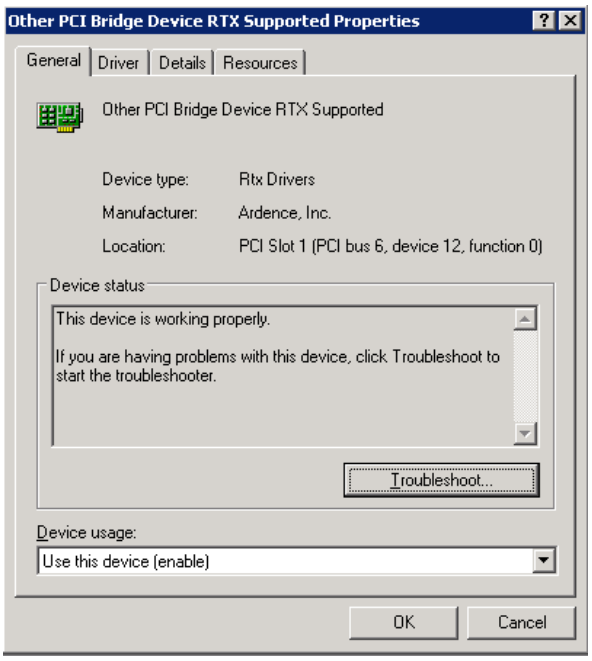
Now the COM168 is automatically recognized as RTX device.

Figure 3-14 COM168 as RTX device (Win XP)



To be safe you should check with the device properties whether it's the right device.

Figure 3-15 Properties as PCI device (Win XP)



3.3.3 Check COM168 installation with “Com168Scan.rtss”

The WinAC CAN basis driver includes a tool to check the installed COM168 board. The tool **Com168Scan.rtss** is located in the **\tools** directory. It is a RTX application. On a computer with installed Ardence RTX you can start this application with double click. This is the case on all computers with WinAC RTX.

Figure 3-16 Output of Com168Scan.rtss

```
<Com168Scan> <Release-Version>
START

COM168 found:
bus <6>  slot/dev <12>  func. <0>   INT <20>
vendor <0x10B5>  dev <0x9030>  subvendor <0x0000>  subsystem
<0x3413>

COM168 Info:
Software-Version:       <0x00920000>
Hardware-Version:       <0x01010000>
Flashbootloaderversion: <0x01060000>
FPGA-Version:           <0x01050000>
END
```

3.4 Install HW “Nanobox CAN on-board Interface” (WES 7)

The following screen shots are created under Windows 7. The processing under Windows XP embedded is comparable.

The CAN interface of the Nanobox IPC227D is connected to the system as PCIe device. It supports MSI (Message-Signaled Interrupts). The real-time driver realizes the interrupt processing with MSI.

3.4.1 Install CAN interface as RTX device

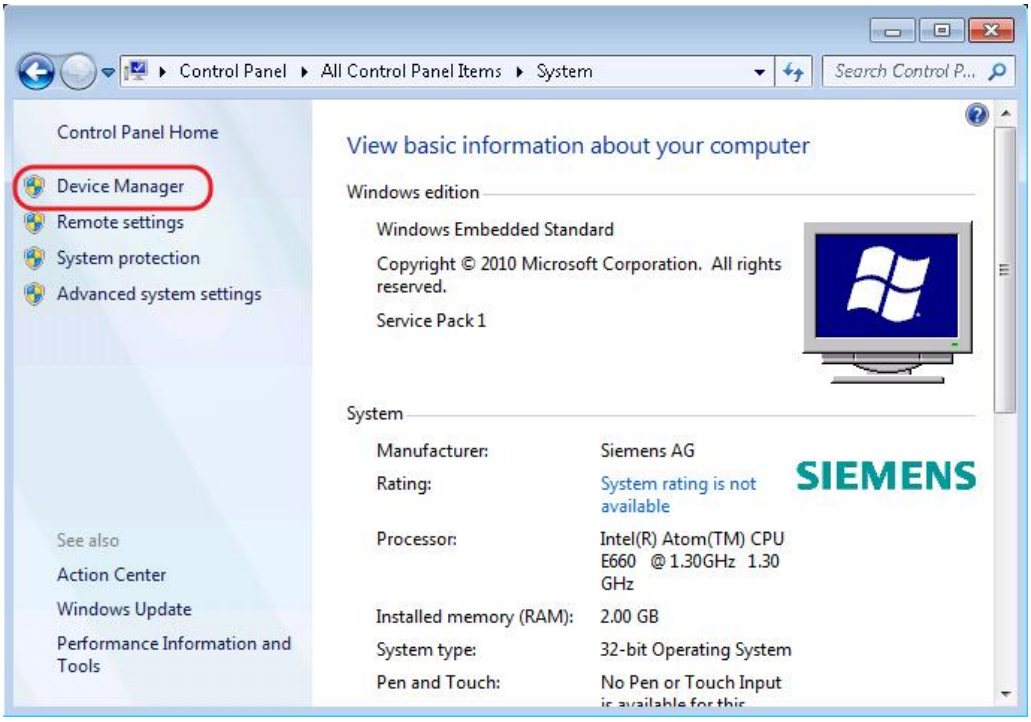
The driver for the CAN interface of the Nanobox IPC227D is realized as real-time driver for the Windows real-time extension **IntervalZero RTX** (Realtime eXTension). That’s why the CAN interface has to be installed as RTX device.

NOTICE	**If the CAN interface is not setup as RTX device, the starting of the RTX driver will cause a program failure.**

Find CAN interface in the device manager

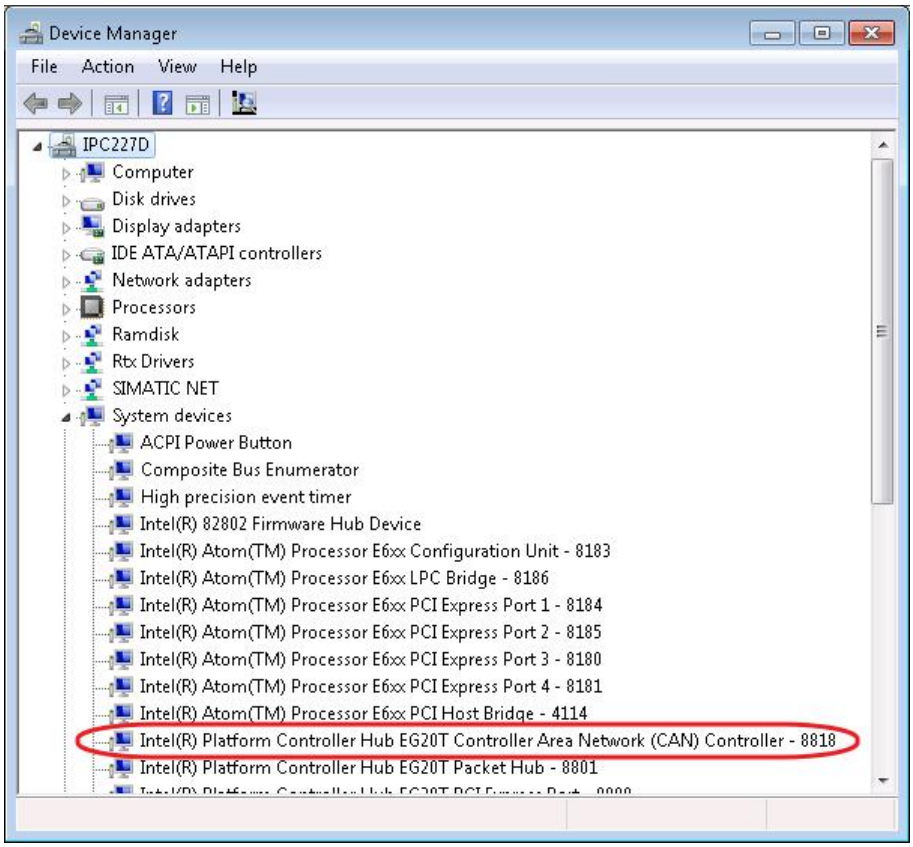
The device manager can be started over the system properties.

Figure 3-17 start device manager (WES 7)



The CAN interface is located in the “System devices” section.

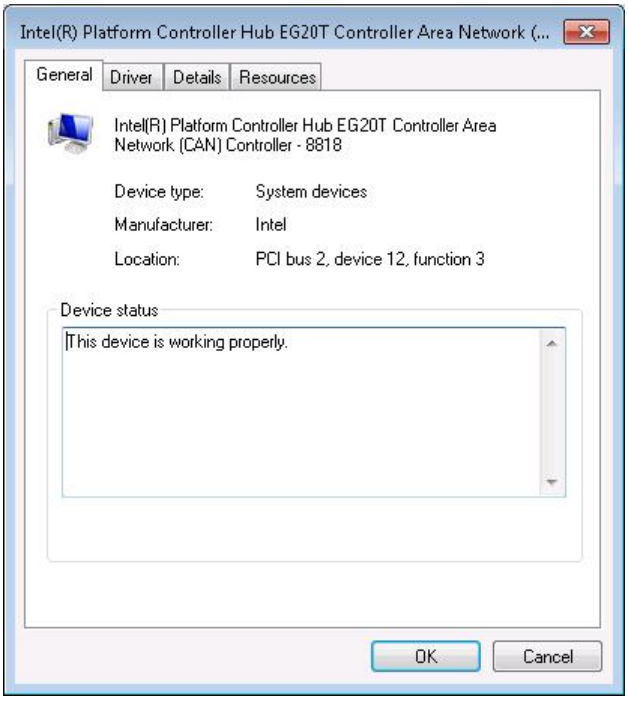
Figure 3-18 CAN interface in the Device Manager (WES 7)



Characteristic properties of the CAN interface

On the properties page one can find characteristic properties of the device, like the slot, bus, device and function or the device and vendor ID.

Figure 3-19 Characteristic device properties



Settings in the RTX Properties

Via control panel of the Windows operating system you can reach the RTX Properties.

Figure 3-20 RTX Properties in control panel (WES 7)

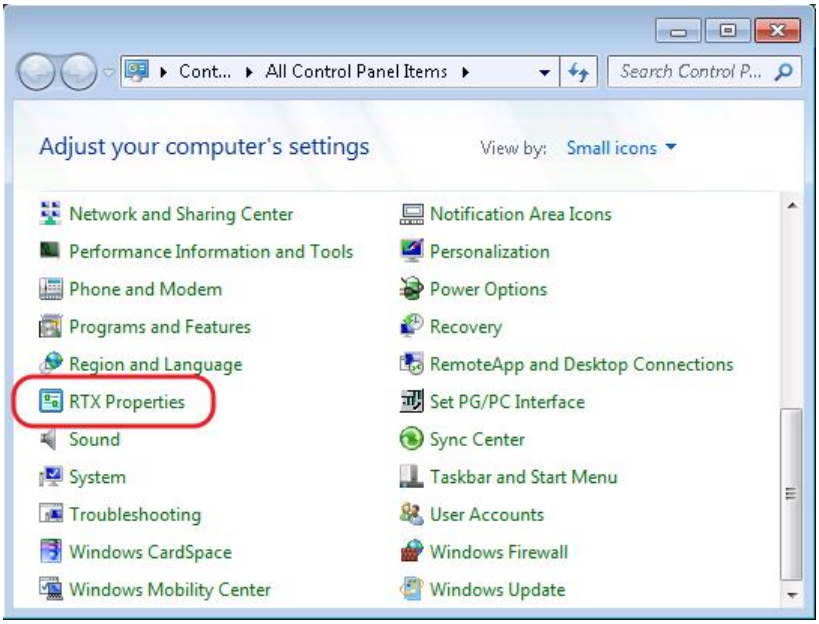
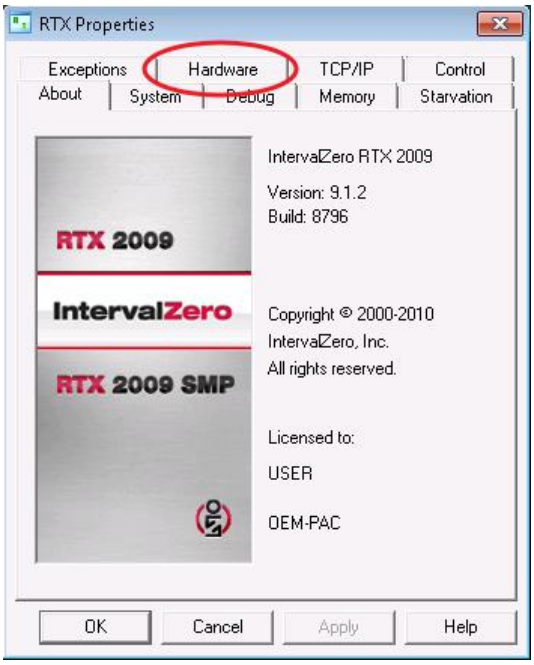
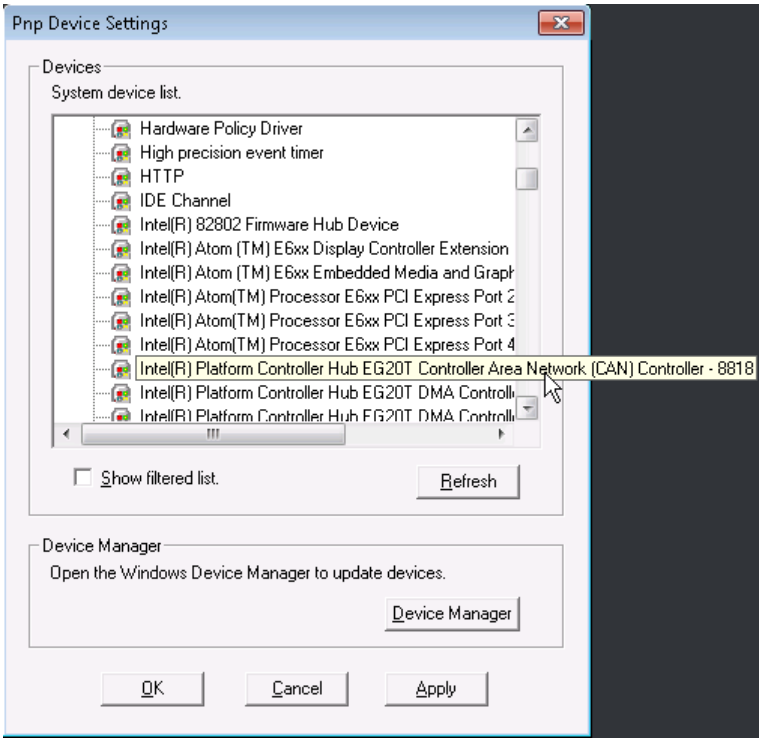


Figure 3-21 RTX Properties, tab “Hardware”



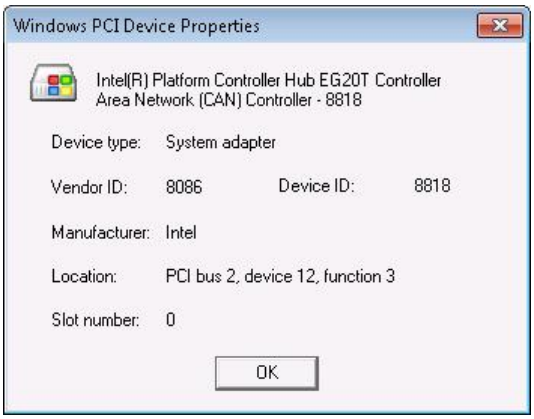
In the tab “Hardware” select the button “Settings …” under the “Devices”. You have to remove the check mark **show filtered list**. Then you can see the CAN interface.

Figure 3-22 Plug and Play settings of the RTX Properties (WES 7)



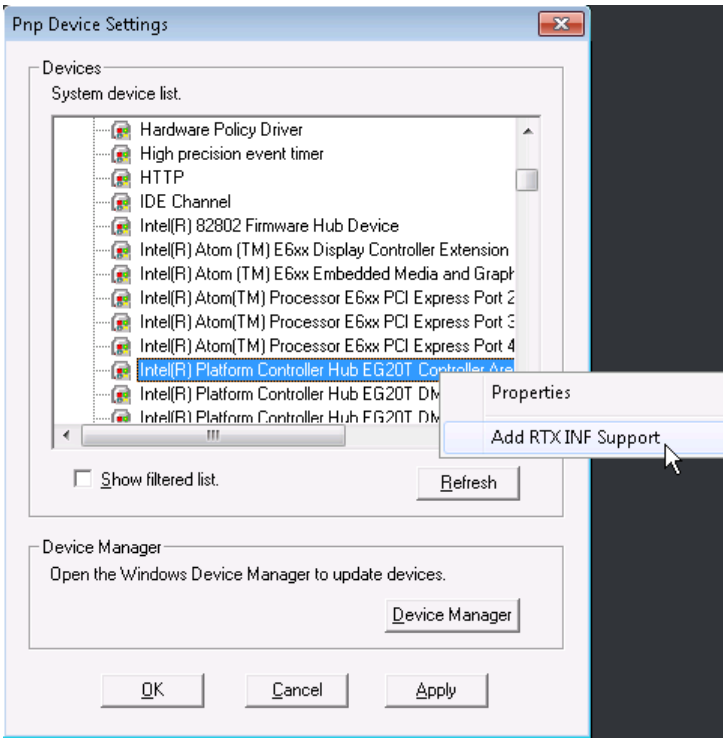
Via the device properties you can verify whether it’s the right device (slot, bus, ...).

Figure 3-23 Device Properties in dialogue RTX Properties



Next you choose the device with the right mouse bottom and click on **Add RTX INF support.**

Abbildung 3-24 Adding RTX INF support to the COM168

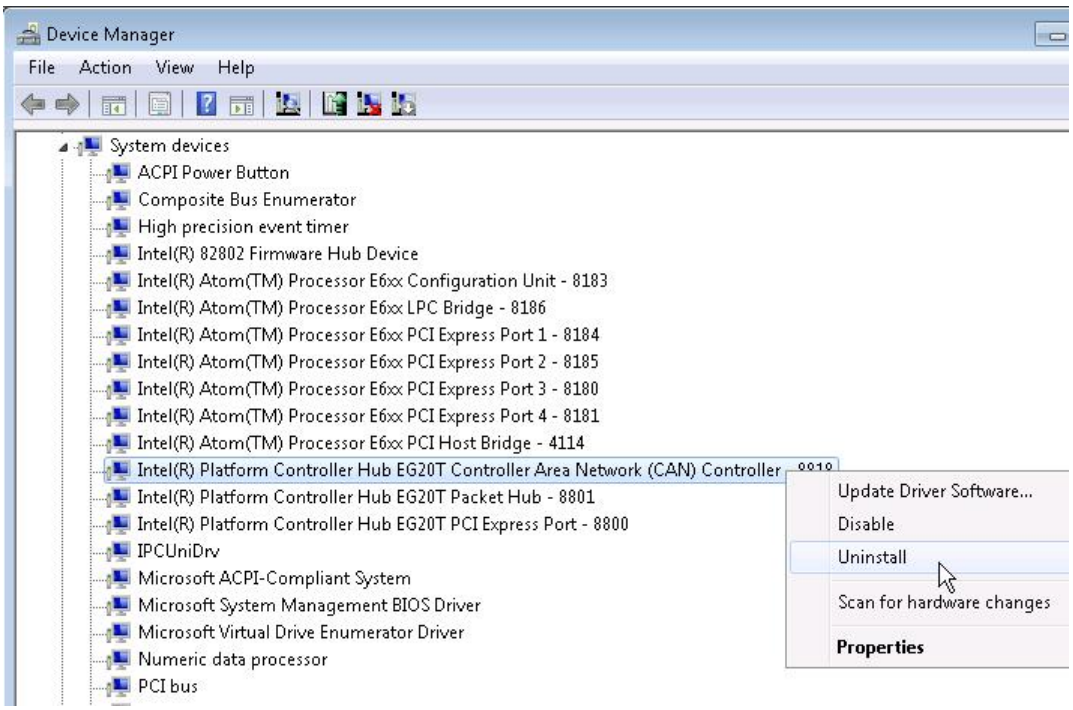


NOTICE **You must press the „Apply“ button to activate the „RTX INF Support“!**

Uninstall the device in the Windows device manager

Next you go again to the **Windows device manager** in order to uninstall the COM168 („Other PCI Bridge Device“ with question mark).

Figure 3-25 Device Uninstallation (WES 7)

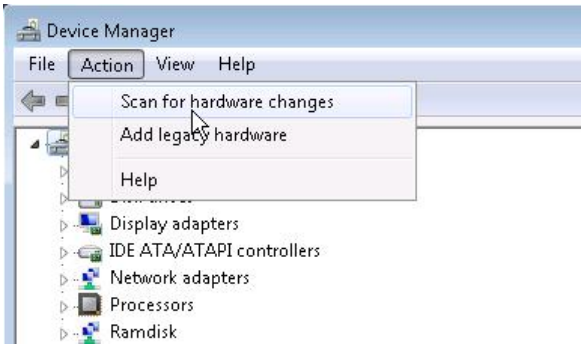


The device disappears firstly from the device manager.

Installation of device as RTX device

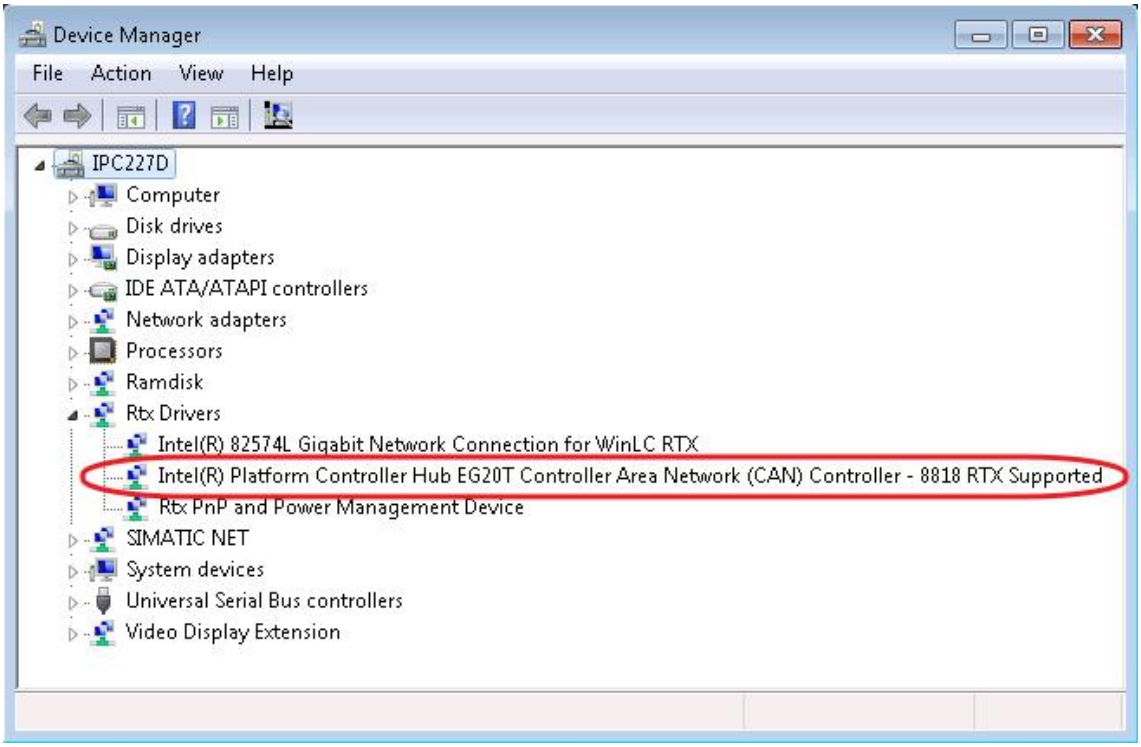
Afterwards you choose at the menu **Action → Scan for new Hardware**.
Now the CAN interface will be registered as RTX device automatically.

Figure 3-26 Scan for hardware changes (WES 7)



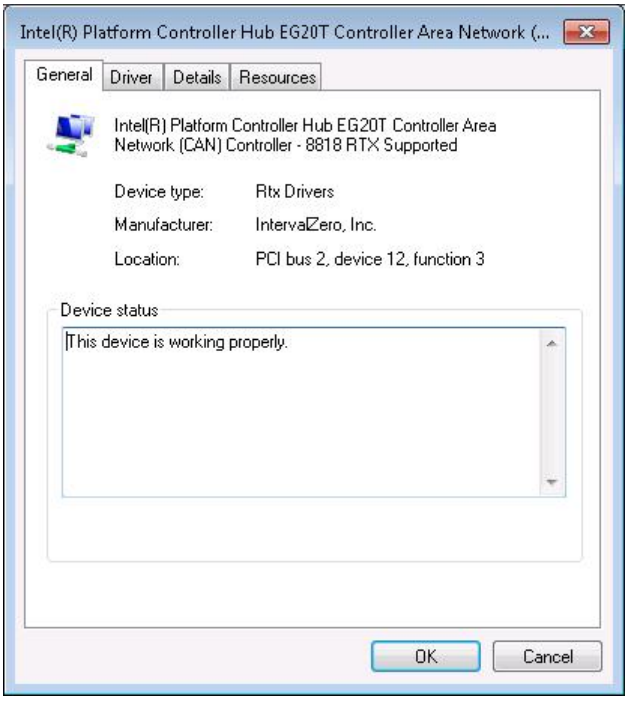
Now the CAN interface is automatically recognized as RTX device.

Figure 3-27 CAN interface as RTX device (WES 7)



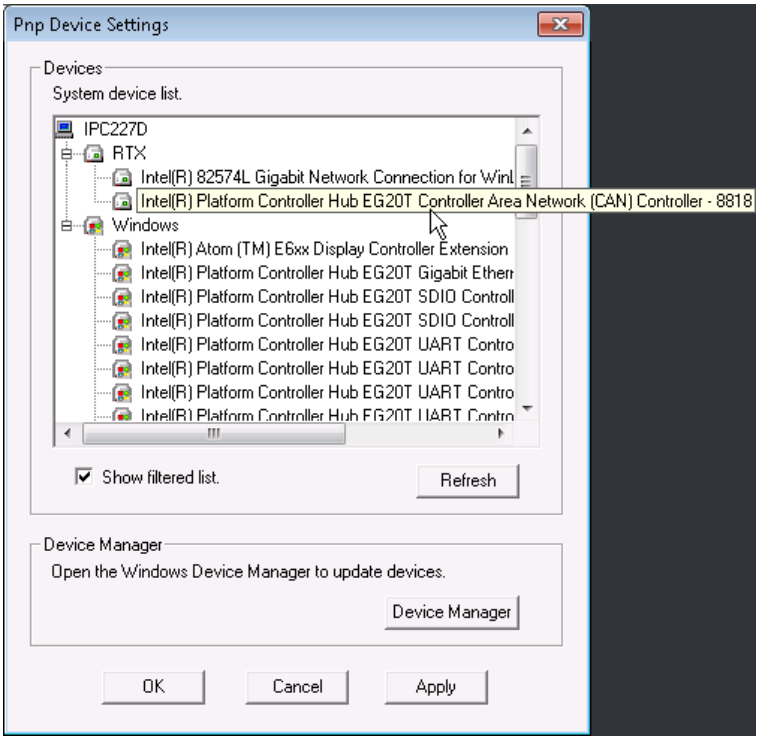
To be safe you should check with the device properties whether it's the right device.

Figure 3-28 Properties as PCI device (WES 7)



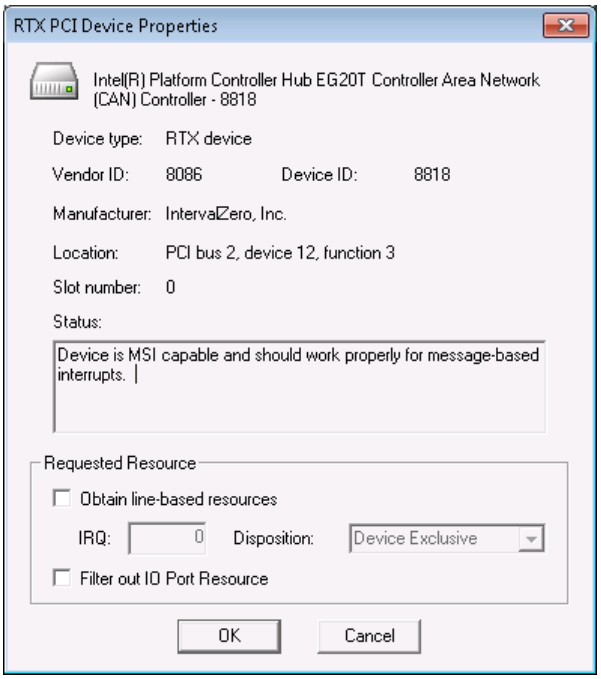
The “Pnp Device Settings” of the “RTX properties” now show the CAN interface in the “RTX” section.

Figure 3-29 CAN interface as RTX device in the RTX devices dialog



The device properties (right mouse click / properties) show the MSI capability of this device.

Figure 3-30 MSI capability in the RTX device properties



3.4.2 Check installation with “CCanScan.rtss”

The WinAC CAN basis driver includes a tool to check the installed CAN interface. The tool **CCanScan.rtss** is located in the **\tools** directory. It is a RTX application. On a computer with installed Ardence RTX you can start this application with double click. This is the case on all computers with WinAC RTX.

Figure 3-31 Output of CCanScan.rtss

```
==================================
Start  <CCanScan> V1.4.0.0

(1) Searching for C_CAN interface ...
PCI information of C_CAN interface:
  Vendor ID <0x8086> Device ID <0x8818>
  Bus<2> Dev<12> Func<3>
(2) Initialize C_CAN interface ...
(3) Configure CAN parameter: <Loopback @ 1 MBit> ...
(4) Sending CAN telegramm (Loopback mode) ...
(5) Receiving CAN telegramm (Loopback mode) ...

Scan and test of C_CAN interface:
   <SUCCESS>

(6) Clean-Up C_CAN interface ...
End   <CCanScan>
==================================
```

The shown PCI information should fit the information of the Windows device manager (or RTX properties).

NOTICE **Before starting CCanScan.rtss the first time, the CAN interface must be set-up as RTX device! (see chapter 3.4.1)**

3.5 Installation WinAC driver on runtime system

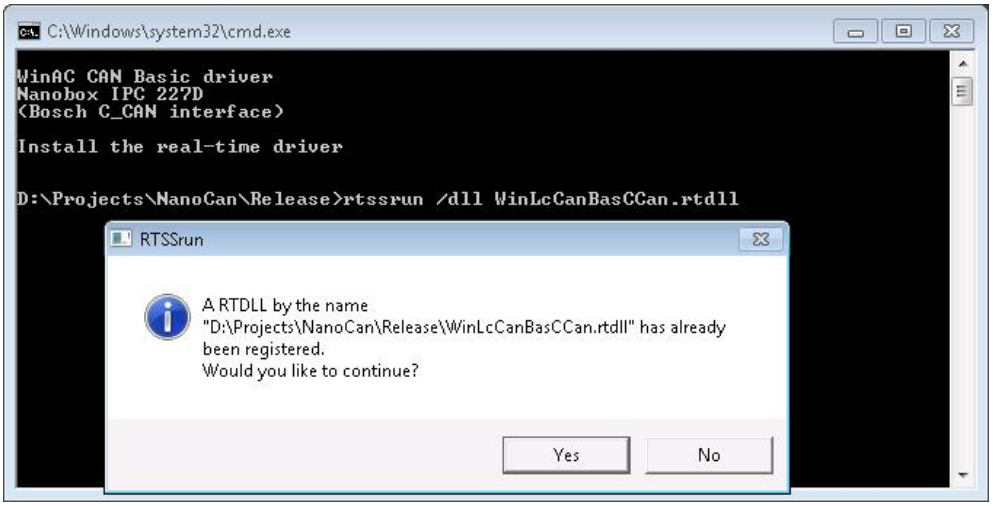
The installation of the WinAC CAN driver is limited to the registration of the driver realtime DLL. One has to use the setup routine corresponding to the available type of CAN interface.

Table 3-1 Setup depending on the type of CAN interface

CAN interface	Setup file
Microbox IPC 427B/C on-board	Setup_SJA1000.bat
COM168 PCI Steckkarte	Setup_Com168.bat
Nanobox IPC 227D on-board	Setup_CCan.bat

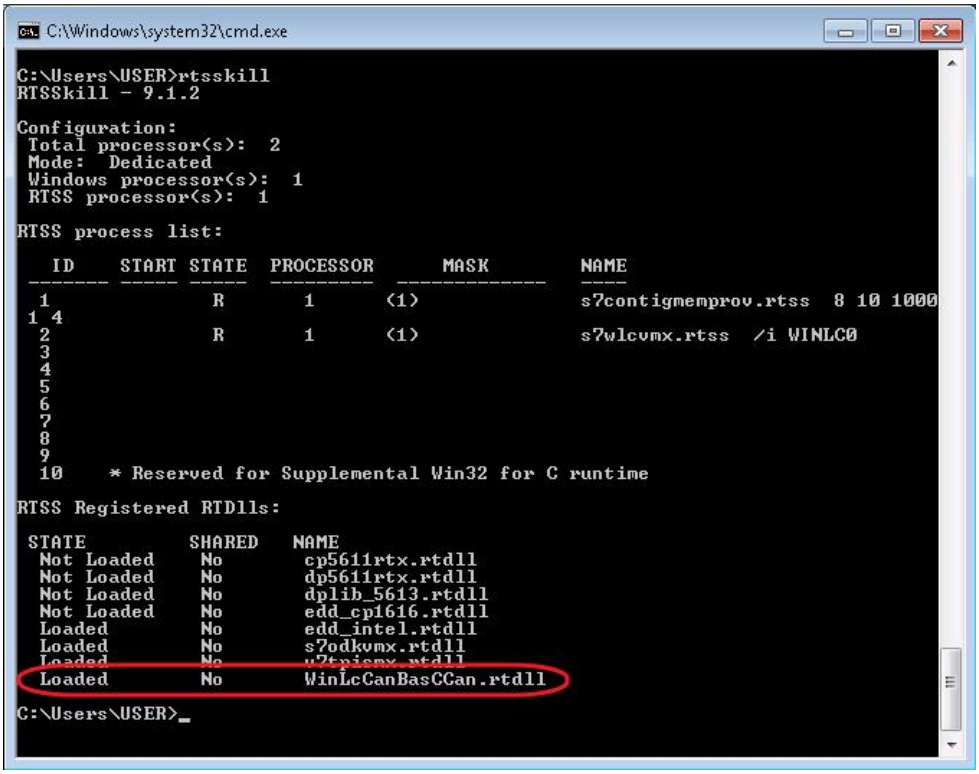
If a version of the driver is just installed, a message will be shown. Please confirm the “Yes” button.

Figure 3-32 Information, if a driver is just installed (Nanobox / WES7)



You can check the installation with the command **rtsskill**. At the registered DLLs you must see the WinLcCanBasCom168.rtdll, WinLcCanBasSja1000.rtdll or WinLcCanBasCCan.rtdll.

Figure 3-33 Installed real-time drivers (Nanobox / WES7)



3.6 Installation WinAC driver on engineering system

On the engineering system only this documentation and the Step7 or TIA portal example program is needed. You can copy the needed function blocks and user defined types from this Step7 / TIA portal example to you own application.

There is no need to execute the batch file setup_xxx.bat on the engineering system!

NOTICE **In the provided example applications all three different INIT function blocks are included in OB1 and OB100. You have to utilize the right one fort he used hardware.**

4 Simatic projecting for CAN driver

4.1 Component Configurator on runtime system

The CAN interface is NOT part of the station configuration of the station configuration of the runtime system.

All settings are done with DBs/FBs.

4.2 Simatic Manager HW Config

The CAN interface is NOT part of the station hardware configuration of the Simatic Manager project.

All settings are done with DBs/FBs.

5 The Step7 user interface

To use the CAN functionality from the WinAC program there are some function blocks and user defined types in the example project. Of course you can change the numbers of the FBs, DBs and UDTs.

FB61000 – CAN_INIT_COM168

FB61001 – CAN_INIT_SJA1000

FB61002 – CAN_SEND

FB61003 – CAN_RECV

FB61004 – CAN_GET_STATUS

FB61005 – CAN_INIT_CCAN

Only the initialisation function block depends on the used hardware platform Microbox (SJA1000), Nanobox (C_CAN) or COM168.

5.1 Multi instance FBs

The driver FBs are not multi instancable!

Explanation:

The WinAC driver is realised with the WinAC ODK (Open Develeopment Kit). All driver FBs need the reference to the driver RTDLL (ODK handle). The Init-FB distributes this ODK handle to all the instance DBs of the driver FBs and does some initialisation within the instance DBs.

5.2 Initialisation of WinAC CAN Basis driver

The initialisation function block has to be called before any other driver FB call.

This FB loads the driver RTDLL. It distributes the information about the ODK handle to the instanced DBs of all driver FBs. The FB reads the information about the installed COM interfaces (on-board or COM168). This data is stored in the instance DB of this FB.

Additional this function block initialises the CAN interface(s).

The RTX driver checks the version of the Step7 FBs. Only for a matching version the FB call is done without error.

There are three different function blocks for initialisation, depending on the used hardware (SJA1000 / C_CAN / COM168). You have to use the matching function block.

Attention

The driver supports <u>either</u> the on-board CAN interface Microbox (SJA1000), Nanobox (C_CAN) <u>or</u> the COM168 card. Thus you may call only one of the three initialisation function blocks only!

Coding of the RTDLL version

The version of the RTDLL is coded hexadecimal. The last digit of the DWORD is used to label the Debug or Release version.

D – Debug-Version

A – Release-Version

Figure 5-1 Examples for RTDLL version in instance DB of CAN_INIT_sss

```
"iDB_CAN_INIT_COM168".C_IF.DLL_VERSION   HEX   DW#16#000 1000 D
                                                        \  /|
                                                         \/ +- Debug
                                                         +---- V 1.0.0.0


"iDB_CAN_INIT_SJA1000".C_IF.DLL_VERSION   HEX   DW#16#000 1100 A
                                                         \  /|
                                                          \/ +- Release
                                                          +---- V 1.1.0.0
```

Note The data of the instance DB (e.g. driver version) is valid after an error-free call of CAN_INIT_xxx only!

5.2.1 Initialisation with CAN_INIT_SJA1000

If you want to use the CAN on-board interface of the Microbox PC427B you have to call this function block.

Attention

The function block CAN_INIT_SJA1000 may not be used at same time as INIT_CAN_COM168 because the driver does support only one CAN hardware at one time.

Table 5-1 Parameter des FBs CAN_INIT_SJA1000

Parameter	In/ Out	Type	Description
STARTUP	In	Bool	Flag for first call
DB_CAN_SEND	In	Block_DB	Instance DB of FB CAN_SEND *1)
DB_CAN_RECV	In	Block_DB	Instance DB of FB CAN_RECV *1)
DB_CAN_GET_STATUS	In	Block_DB	Instance DB of FB CAN_GET_STATUS *1)
REQ	In	Bool	Request for INIT CAN (positive edge)
USE_SHUTDOWN	In	Bool	Use shutdown message
CAN_PARAM	In	UDT_CAN_CFG_SJA1000	Parameters of CAN interface SJA1000
SHUTDOWN_MSG	In	UDT_CAN_MSG *2)	CAN message sent in transition RUN → STOP
BUSY	Out	BOOL	Set; if the FB is still busy
DONE	Out	BOOL	Is set, if the FB has finished (with error or without error)
ERROR	Out	BOOL	If set, an error occurred. STATUS gives detailed information about the error.
STATUS	Out	WORD	Error information

*1) The numbers of the instance DBs have to be given with the first call (STARTUP = TRUE).

*2) See Table 5-12 “UDT for CAN telegram UDT_CAN_MSG”, pg. 47

Table 5-2 Definition of UDT UDT_CAN_CFG_SJA1000

Parameter	In/ Out	Type	Description
SJA1000_BASE_ADDR	In	Word	Base address of SJA1000 (see BIOS settings)
SJA1000_IRQ_NO	In	Int	Interrupt number of SJA1000 (see Windows device manager)
SJA1000_BAUD_CAN	In	Int	Baud rate for CAN *1)

*1) The baud rate for the CAN interface is defined by the following values:

0 - 1 MBit/s
1 - 800 KBit/s (not supported by SJA1000)
2 - 500 KBit/s
3 - 250 KBit/s
4 - 125 KBit/s
5 - 100 KBit/s
6 - 50 KBit/s
7 - 20 KBit/s (not supported by SJA1000)

5.2.2 Additional information in the instance DB of CAN_INIT_SJA1000

The user can obtain additional information in the instance DB of the INIT function block:

Table 5-3 Informationen of instance DB of CAN_INIT_SJA1000

Name	In/Out	Bemerkung
C_IF.FB_VERSION	Out	Version of Step7-FBs of the driver
C_IF.DLL_VERSION	In	Version driver RTDLL

5.2.3 Initialisation with CAN_INIT_COM168

If you want to use the COM168 card you have to call this function block.

Attention

The function block CAN_INIT_COM168 may not be used at same time as INIT_CAN_SJA1000 because the driver does support only one CAN hardware at one time.

Table 5-4 Parameter of FB CAN_INIT_COM168

Parameter	In/ Out	Typ	Beschreibung
STARTUP	In	Bool	Flag for first call
DB_CAN_SEND	In	Block_DB	Instance DB of FB CAN_SEND *1)
DB_CAN_RECV	In	Block_DB	Instance DB of FB CAN_RECV *1)
DB_CAN_GET_STATUS	In	Block_DB	Instance DB of FB CAN_GET_STATUS *1)
REQ	In	Bool	Request for INIT CAN (positive edge)
USE_SHUTDOWN_0	In	Bool	Use shutdown message #0
USE_SHUTDOWN_1	In	Bool	Use shutdown message #1
CAN_PARAM	In	UDT_CAN_CFG_COM168	Parameter of the two CAN interfaces of COM168
SHUTDOWN_MSG0	In	UDT_CAN_MSG	CAN message sent in transition RUN → STOP via CAN #0
SHUTDOWN_MSG1	In	UDT_CAN_MSG *2)	CAN message sent in transition RUN → STOP via CAN #1
BUSY	Out	BOOL	Set; if the FB is still busy
DONE	Out	BOOL	Is set, if the FB has finished (with error or without error)
ERROR	Out	BOOL	If set, an error occurred. STATUS gives detailed information about the error.
STATUS	Out	WORD	Error information

*1) The numbers of the instance DBs have to be given with the first call (STARTUP = TRUE).

*2) See Table 5-12 “UDT for CAN telegram UDT_CAN_MSG”, pg. 47

Table 5-5 Definition of UDT UDT_CAN_CFG_COM168

Parameter	In/ Out	Typ	Beschreibung
BAUD_CAN_0	In	Int	Baud rate for CAN channel #0 *1)
BAUD_CAN_1	In	Int	Baud rate for CAN channel #1 *1)
BTRON_CAN_0	In	Bool	CAN channel 0: 0 – use pre-defined baud rate (see above) 1 – use bit timing register for user defined-

Parameter	In/ Out	Typ	Beschreibung
			baud rate
BTRON_CAN_1	In	Bool	CAN channel 1: 0 – use pre-defined baud rate (see above) 1 – use bit timing register for user defined-baud rate
BTR	In	Array[0..1]	Bit timing register for user defined baud rate *2)
PRESDIV	In	Byte	Prescaler division factor
RJW	In	Byte	Resyncronization jump width: 0..3
PSEG1	In	Byte	Phase buffer segment 1: 0..7
PSEG2	In	Byte	Phase buffer segment 2: 0..7
PROPSEG	In	Byte	Propagation segment: 0..7
SETVAL_ ERRFRAMCTR	In	Word	Set Value Error Frame Counter

*1) Pre-defined baud rates

*1) The baud rate for the CAN interface is defined by the following values:

0 - 1 MBit/s
1 - 800 KBit/s
2 - 500 KBit/s
3 - 250 KBit/s
4 - 125 KBit/s
5 - 100 KBit/s
6 - 50 KBit/s

Baud rate 20 and 10 Kbit/s are **not** supported by the COM168 card!

*2) Bit Timing Register (BTR) for user defined baud rates

The parameters of the Bit Timing Register (BTR) define the timing behavior of the CAN signals and (indirect) the baud rate.

For 200 kBit/s the following settings are possible:

Table 5-6 Alternative settings of Bit Timing Register for 200 kBit/s

PRESDIV	PROPSEG	PSEG1	PSEG2	RJW	→ SP
0x13	0x7	0x7	0x7	2	68%
0x18	0x7	0x7	0x2	2	85%
0x31	0x6	0x0	0x0	2	90%

The **Set Value Error Frame Counter** can be computed by the choosen baud rate:

SetVal = (5.000.000 * bit-time * 10) / baud rate

For 200 kBit/s and a bit-time of 5.4 the right value is:

SetVal = (5.000.000 * 54) / 200.000

SetVal = 1.350 dez = 546 hex

5.2.4 Additional information in the instance DB of CAN_INIT_COM168

The user can obtain additional information in the instance DB of the INIT function block:

Table 5-7 Information in the instance DB of CAN_INIT_COM168

Name	In/ Out	Comment
C_IF.FB_VERSION	Out	Version of Step7-FBs
C_IF.DLL_VERSION	In	Version of driver RTDLL
C_IF.COM168_SW_VER	In	SW version COM168
C_IF.COM168_HW_VER	In	SW version des COM168
C_IF.COM168_FBL_VER	In	Firmware bootloader version COM168
C_IF.COM168_FPGA_VER	In	FPGA version COM168

5.2.5 Initialisation with CAN_INIT_C_CAN

If you want to use the CAN on-board interface of the Nanobox IPC227D you have to call this function block.

Attention

The function block CAN_INIT_C_CAN may not be used at same time as INIT_CAN_COM168 or CAN_INIT_SJA1000 because the driver does support only one type of CAN hardware at one time.

Table 5-8 Parameter des FBs CAN_INIT_C_CAN

Parameter	In/ Out	Type	Description
STARTUP	In	Bool	Flag for first call
DB_CAN_SEND	In	Block_DB	Instance DB of FB CAN_SEND *1)
DB_CAN_RECV	In	Block_DB	Instance DB of FB CAN_RECV *1)
DB_CAN_GET_STATUS	In	Block_DB	Instance DB of FB CAN_GET_STATUS *1)
REQ	In	Bool	Request for INIT CAN (positive edge)
USE_SHUTDOWN	In	Bool	Use shutdown message
CAN_PARAM	In	UDT_CAN_CFG_C_CAN	Parameters of CAN interface C_CAN
SHUTDOWN_MSG	In	UDT_CAN_MSG *2)	CAN message sent in transition RUN → STOP
BUSY	Out	BOOL	Set; if the FB is still busy
DONE	Out	BOOL	Is set, if the FB has finished (with error or without error)
ERROR	Out	BOOL	If set, an error occurred. STATUS gives detailed information about the error.
STATUS	Out	WORD	Error information

*1) The numbers of the instance DBs have to be given with the first call (STARTUP = TRUE).

*2) See Table 5-12 “UDT for CAN telegram UDT_CAN_MSG”, pg. 47

Table 5-9 Definition of UDT UDT_CAN_CFG_C_CAN

Parameter	In/ Out	Type	Description
BAUD_CAN	In	Int	Baud rate for CAN *1)

*1) The baud rate for the CAN interface is defined by the following values:

0 - 1 MBit/s

1 - 800 KBit/s

2 - 500 KBit/s

3 - 250 KBit/s
4 - 125 KBit/s
5 - 100 KBit/s
6 - 50 KBit/s
7 - 20 KBit/s

5.2.6 Additional information in the instance DB of CAN_INIT_C_CAN

The user can obtain additional information in the instance DB of the INIT function block:

Table 5-10 Information of instance DB of CAN_INIT_C_CAN

Name	In/Out	Bemerkung
C_IF.FB_VERSION	Out	Version of Step7-FBs of the driver
C_IF.DLL_VERSION	In	Version driver RTDLL

5.3 Send CAN telegrams with CAN_SEND

The function block **CAN_SEND** sends one CAN telegram over the chosen channel (only COM168). The telegram is not send directly but is copied into the Tx-FIFO. The real-time CAN driver transfers the message from the Tx-FIFO to the CAN chip (see chapter 2.1 “Functional range”, pg. 8).

Table 5-11 Parameters of FBs CAN_SEND

Parameter	In/ Out	Type	Description
CHANNEL	In	Byte	Channel-No SJA1000, C_CAN: 0 COM168: 0, 1
CANMSG	In	UDT_CAN_MSG	The CAN telegram to send.
STATUS	Out	WORD	Error information
ERROR	Out	BOOL	If set, an error occurred. STATUS gives detailed information about the error.

Table 5-12 UDT for CAN telegram UDT_CAN_MSG

Parameter	Typ	Beschreibung
IDENT	Dword	Identifier (11 / 29 bit)
DATA_LEN	Byte	Data length
REMOTE	Bool	Remote bit (TRUE = active)
EXTENDED	Bool	Extendet identifier (TRUE = 29 bit)
DATA	Array[1..] of Byte	Telegram data

5.4 Receive CAN telegrams with CAN_RECV

The real-time CAN driver stores all received CAN telegrams in the internal Rx-FIFO. The function block **CAN_RECV** reads one telegram from the Rx-FIFO (see chapter 2.1 “Functional range”, pg. 8).

If the WinAC program reads so slow the Rx-FIFO will overflow. The function block CAN_RECV will return a corresponding error code. Both bits ERROR and NDR are set in this case.

Note A wire break is not recognized by CAN_RECV!

Attention

The COM168 can not receive remote frames, e.g. CAN telegrams with remote bit set.

Table 5-13 Parameters of FBs CAN_RECV

Parameter	In/ Out	Type	Description
CHANNEL	In	Byte	Channel-No SJA1000, C_CAN: 0 COM168: 0, 1
CANMSG	Out	UDT_CAN_MSG *1)	The received CAN telegram.
STATUS	Out	WORD	Error information
ERROR	Out	BOOL	If set, an error occurred. STATUS gives detailed information about the error.
NDR	Out	BOOL	New Data Received – if set, a new telegram was received.

*1) See Table 5-12 “UDT for CAN telegram UDT_CAN_MSG”, pg. 47

5.5 Get CAN status with CAN_GET_STATUS

With this function block you will get a number of status information of the CAN interface.

Table 5-14 Parameters of FB CAN_GET_STATUS

Parameter	In/Out	Type	Description
STATUS	Out	WORD	Error information
ERROR	Out	BOOL	If set, an error occurred. STATUS gives detailed information about the error.

The status information of the CAN channels is stored in the instance DB of the function block. For the SJA1000 only the first is used, for COM168 both structures are used.

Table 5-15 Status information of one CAN channel UDT_CAN_STATUS

Parameter	In/ Out	Type	Description
STATUS.RUN	Out	Bool	CAN is running
STATUS.CAN_ERROR	Out	Bool	CAN error occurred
STATUS.BUS_OFF	Out	Bool	CAN ‚bus off'
STATUS.PASSIVE	Out	Bool	CAN ‚passive error'
STATUS.RX_OVERFLOW	Out	Bool	CAN ‚receive buffer Overflow'
STATUS.TX_ERROR	Out	Bool	CAN ‚transmit error'
RX_MSG_COUNT	Out	DWord	Received messages total
RX_BYTE_COUNT	Out	DWord	Received bytes total
TX_MSG_COUNT	Out	DWord	Transmitted messages total
TX_BYTE_COUNT	Out	DWord	Transmitted bytes total
REM_TX_COUNT	Out	DWord	Remote Frames total
ERR_FRAME_COUNT	Out	DWord	Error frame count

Attention

Note on SJA1000: if the bit "error occurred" is set by a wire break, this bit can not be resented by a CAN_INIT_SJA1000. This bit is only cleared by a hardware reset (property of SJA1000).

6 Examples for applications

The driver package includes two different examples.

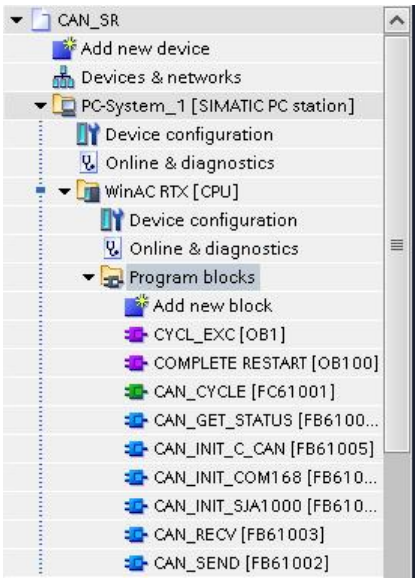
The first example shows how to send / receive CAN telegrams.

In the second example shows a simple implementation of CANopen as part of the application. The send/receive function blocks are used to automatically update the process image. The whole CAN functionality is encapsulated in one function block (FB61010 CAN_UPDATE_PI). In this example two different CANopen DI/DO modules are used.

Both examples are available for Step7 V5.x and for Step7 V11 (TIA Portal). Both versions have the same functionality. Because of the differences of both engineering systems, the examples look a little bit different in details.

Figure 6-1 WinAC CAN driver function blocks in Step7 V5.x

Figure 6-2 WinAC CAN driver function blocks in TIA Portal V11 SP2



6.1 Example 1 – CAN send/receive

In this example you can send / receive single telegrams with triggers (see variable table). Communication partners are CANopen devices.

Matching your hardware (on-board / COM168) you have to call the right INIT FB in OB1 and OB100!

OB100 Complete Restart

In the OB100 the CAN driver is initialized with STARTUP bit is set (First run). The numbers of the instance DBs are giben.

ATTENTION

You have to use the right INIT function block matching your hardware!

OB1 CYCL_EXEC

In the first network the CAN interface is initialised. This is done by a rising edge of the REQ bit. The CANopen telegram for “Pre-operational state” is used as shutdown telegram. When the WinAC changes from RUN to STOP this telegram is sent automatically – all CANopen devices switch to the “pre-operational state” immediately.

ATTENTION

You have to use the right INIT function block matching your hardware!

The processing of the following networks takes place after CAN_READY is set.

In the following networks different prepared CAN telegrams can be sent by trigger bits (see variable table).

In the last networks you can receive CAN telegrams and ask for the CAN status (also done by trigger bits).

DB10000 DB_CAN

This data block contains a number of variables used by various parts of the program. Thus the complete example can work without usage of flags.

DB10001 DB_MSG_SEND

In this data block some CAN telegrams are prepared for sending. The UDT_CAN_MSG is used for the right structure of the messages. In the example four telegrams are prepared for communication with CANopen devices.

- Command “Operational state”
- Command “Pre-operational state”
- Command “Set digital outputs” (CANopen device with Dos)
- Free user command

DB10002 DB_MSG_RECV

This data block contains one telegram for receiving.

UDT61000 UDT_CAN_CFG_COM168

Configuration of both CAN channels of the COM168 card. It includes the baud rate for both channels.

UDT61001 UDT_CAN_CFG_SJA1000

Configuration for the on-board interface of the Microbox IPC427B/C (SJA1000 chipset). It includes the base address, IRQ number and baud rate.

UDT61002 UDT_CAN_MSG

Definition of the structure of on CAN telegram.

UDT61003 UDT_CAN_STATUS

Definition of the structure for the status of one CAN channel.

UDT61004 UDT_CAN_CFG_C_CAN

Configuration for the on-board CAN interface of the Nanobox IPC227D (C_CAN chipset): baud rate of the CAN bus.

6.2 Example 2 – process image with CANopen devices

This example shows how to use two CANopen devices (digital I/O). The function block FB_UPDATE_PI determines the current process image with sent / received CANopen telegrams.

After initialisation of CAN automatically the CANopen telegram for “Operational state” is sent. If you want to adjust some CANopen parameters (like guarding time) you have to send the corresponding telegrams before changing to “Operational state”.

This updating of the process image is encapsulated in one FB (FB61010 CAN_UPDATE_PI).

Attention

Matching your hardware (on-board / COM168) you have to call the right INIT FB in FB61010 CAN_UPDATE_PI!

It is called at <u>two</u> places in this FB!

Note

In this example in every cycle <u>all</u> process output data is written. If you use a small cycle time you will get a high CAN bus load!

To prevent this you only have to send changes of the process output data or to move the process image update in a slow cyclic OB.

OB100 Complete Restart

In the OB100 the FB UPDATE_PI is called with STARTUP bit set.

OB1 CYCL_EXEC

In the beginning the process image is updated by caling FB UPDATE_PI. This FB works with the process image on DB1001 DB_CAN_PI_IN and DB1002 DB_CAN_PI_OUT.

After that the data is used for some manipulation.

FB61010 UPDATE_PI

In this function block the whole CAN processing is encapsulated.

In network 1 some internal variables are initialised during startup.

The following program is realised as step sequence:

- Step 1 – load driver
  In this sep the CAN driver is loaded. The instance DBs of the CAN FBs are given.
  ATTENTION
  You have to use the right INIT function block matching your hardware!

- Step 2 – Init the CAN interface
  In this step the CAN channel(s) are initialised. The CAN configuration is given. The CANopen telegram for “Pre-operational state” is used as shutdown telegram. When the WinAC changes from RUN to STOP this telegram is sent automatically – all CANopen devices switch to the “pre-operational state” immediately.
  ATTENTION
  You have to use the right INIT function block matching your hardware!
- Step 3 – CANopen “Operational state”
  This step sends the CANopen telegram “Operational state” to switch all the devices to this state.
- Step 4 – Update the process image
  In this step all output values are sent to the CANopen devices with the corresponding CANopen telegrams.
  Additional all received telegrams are processed to update the input values of the process image.

This function block has to input parameters – the datablocks containing the current process image.

An important element of the function block is the temporally telegram MSG (UDT_CAN_MSG). It is used for sending and receiving CAN telegrams.

To update the outputs of the process image the corresponding CANopen telegrams are sent to the CANopen devices.

First the telegram is initialised. After that the CANopen telegram is composed for the CAN device, e.g.

- Ident = $22A_h$ (200_h + Node ID $2A_h$)
- Length = 8 Bytes
- Data DO0, DO1

After composing this telegram it is sent by CAN_SEND.

The update of the process image inputs is done by processing all telegram. In the loop all telegrams are read from the RxFIFO by CAN_RECV. These telegrams are analysed and the new input values are copied to the datablock for the input data:

First the temporally telegram MSG is initialised. Then one telegram is picked up (CAN_RECV). If NDR bit is set, the identifier is separated to module address and object number.

In the following networks object number and module address are processed. The data is copied to the matching address in the process image input data block.

This loop is processed until no CAN telegram is delivered by CAN_RECV.

At least the status of the CAN interface is updated.

DB1001 DB_CAN_PI_IN / DB1002 DB_CAN_PI_OUT

These data blocks contain the current process image provided by the CANopen devices.

UDT61000 UDT_CAN_CFG_COM168

Configuration of both CAN channels of the COM168 card. It includes the baud rate for both channels.

UDT61001 UDT_CAN_CFG_SJA1000

Configuration for the on-board interface of the Microbox IPC427B/C (SJA1000 chipset). It includes the base address, IRQ number and baud rate.

UDT61002 UDT_CAN_MSG

Definition of the structure of on CAN telegram.

UDT61003 UDT_CAN_STATUS

Definition of the structure for the status of one CAN channel.

UDT61004 UDT_CAN_CFG_C_CAN

Configuration for the on-board CAN interface of the Nanobox IPC227D (C_CAN chipset): baud rate of the CAN bus.

7 Error Codes

The WinAC CAN-Basis driver can provide different classes of error messages:

- Code in the FB-output **STATUS** according to WinAC-ODK (see chapter 8.1 in this document)
- Special error codes of the CAN-Basis driver (see chapter 8.2 on page 56 in this document)

7.1 Error codes of WinAC ODK

The WinAC CAN-Basis driver had been developed with the WinAC ODK (Open Development Kit). The ODK can also generate error codes, which are returned from the **STATUS** of the FBs.

Table 7-1 WinAC ODK error messages

ODK Code (HEX)	Description
0	Success
8001	An exception occurred.
8002	Input: the ANY pointer is invalid.
8003	Input: the ANY pointer range is invalid.
8004	Output: the ANY pointer is invalid.
8005	Output: the ANY pointer range is invalid.
8006	More bytes were written into the output buffer by the extension object than were allocated.
8007	ODK system has not been initialized: no previous call to SFB65001 (CREA_COM).
8008	The supplied handle value does not correspond to a valid extension object.
8009	More bytes were written into the input buffer by the extension object than were allocated.
807F	An internal error occurred.
80C3	Maximum number (32) of parallel jobs/instances exceeded.
8102	The call to CLSIDFromProgID failed.
8103	The call to CoInitializeEx failed.
8104	The call to CoCreateInstance failed.
8105	The library failed to load.
8106	A Windows response timeout occurred.
8107	Controller is in an invalid state for scheduling an OB.
8108	Schedule information for OB is invalid.
8109	Instance ID for SFB65001 call is invalid.
810A	Controller could not load proxy DLL.
810B	The WinAC controller could not create or initialize shared memory area.
810C	Attempt to access unavailable option occurred.
8201	The Execute command index could not be found

ODK Code (HEX)	Description
8250	No more available positions in the job list
8252	The count is invalid
8253	A data type of an item in the list is invalid
8254	The count specified is invalid
8255	A memory area of an item in the list is invalid
8256	A DB number of an item in the list is invalid
8257	A bit number of an item in the list is invalid
8258	A pBuff of an item in the list is invalid
8259	A data quantity is invalid
825A	The area offset parameter is invalid for this type
825B	The frequency value is invalid
825C	The callback pointer is invalid
825D	The job ID pointer is invalid
825E	The job ID is invalid
825F	Job could not be completed because address is incorrect
8260	Job could not be completed because of protection level
8261	Job could not be completed because of hardware issues
8301	Invalid Thread Execution Priority
8401	Invalid Asynchronous Event
8402	Asynchronous Processor Queue is empty
8403	Asynchronous Processor Queue is full

7.2 Special error codes of the CAN Basis driver

Among the general error bit of the driver FBs there is a special error code in the value of **STATUS** to describe the reason of the problem.

Table 7-2 Error codes of CAN-Basis driver

0 - no error
errors of Sja1000 driver
0x8500 - error creating receive event
0x8501 - error creating transmit trigger event
0x8502 - error creating transmit polling thread
0x8503 - error setting transmit trigger event
0x8504 - error creating receive ISR/polling thread
0x8505 - receive event not defined
0x8510 - no SJA 1000found
0x8511 - more than one SJA1000 found
0x8512 - given base address differs from found base address
0x8513 - error reaching Reset mode during searching SJ1000
0x8514 - wrong SJA1000 mode (only BasicCAN and PeliCAN)
0x8515 - SJA1000 controller does not enter Reset Mode
0x8516 - SJA1000 controller does not leave Reset Mode
0x8517 - error register interrupt service routine
0x8518 - base address of SJA100 not defined
0x8519 - undefined code for baud rate
0x851A - baud rate unsupported
0x851B - SJA1000 transmit buffer not empty
0x851C - CAN telegram to long
0x851D - no receive data in buffer
0x851E - no receive data in FIFO
0x851F - no receive data in FIFO
0x8520 - undefined pointer for sending data
0x8521 - undefined pointer for receiving data
0x8522 - problem while transmitting data occured
0x8530 - CAN is not active (not initialized without error)
0x8551 - receive FIFO overflow (RxFIFO)
0x8552 - send FIFO overflow (TxFIFO)
0x8561 - Error RtGetClockTime() for start time
0x8562 - Error RtGetClockTime() for end time
0x8563 - internal table for time stamps is full
0x8564 - undefined ID for time stamp table
0x8570 - internal error #1

0x8571 - internal error #2
0x8572 - internal error #3
0x8573 - internal error #4
0x8574 - internal error #5

errors of COM168 driver

0x8801 - more than one COM168 found
0x8802 - RtTranslateBusAddress failed
0x8803 - Failure on RtMapMemory
0x8804 - RtAttachInterruptVectorEx failed
0x8805 - undefined CAN channel number
0x8810 - Cypress arbitration: got no access in waiting time
0x8811 - timeout setting Tx delay
0x8812 - undefined baudrate value
0x8813 - timeout setting baudrate
0x8814 - timeout initiating SW reset of COM168
0x8815 - no receive data in buffer of COM168
0x8816 - transmit buffer overflow COM168
0x8817 - got no abritation from COM168 for transmit
0x8818 - got no abritation from COM168 for receive
0x8819 - got no abritation from COM168 for status
0x8820 - undefined object for status request

0x8830 - no CAN config object given
0x8831 - bit rate parameter RJW out of range
0x8832 - bit rate parameter PSEG1 out of range
0x8833 - bit rate parameter PSEG2 out of range
0x8834 - bit rate parameter PROPSEG out of range

errors with WinAC Handling (ODK part)

0x8901 - error using ODK_Read.. function
0x8902 - error using ODK_Write.. function
0x8911 - Step7 driver version does not match
0x8913 - false number for WinAC IRQ OB (only 52-54 allowed)
0x8921 - error creating event for signaling IRQ to WinAC
0x8923 - error creating event for signaling 'DeActivate ready'
0x8922 - multiple creation of IRQ monitor (perhaps multiple CAN_INI

errors of asynchronous initialization

0x8A00 - error creating ODK object 'async. command'
0x8A01 - error creating internal init event

0x8A02 - error creating init thread
0x8A03 - error: wait for init event abandoned
0x8A04 - error: wait for init event failed
0x8A05 - error: wait for init stopped undefined
0x8A06 - error: timeout while waiting for init event
0x8A07 - error killing init thread
0x8A08 - error closing thread handle
0x8A09 - error signaling the init event
0x8A0A - 'first call' bit not set in first call of initialization
0x8A0B - 'first call' bit is set in follow call of initialization

errors of C_CAN driver

0x8b01 - no Bosch C_CAN interface found
0x8b02 - more than one Bosch C_CAN interface found
0x8b03 - wrong CAN mode (only Basic CAN and Extended CAN)
0x8b04 - wrong bit timing register values
0x8b05 - error calculating bit rate
0x8b06 - no MSI support signaled
0x8b07 - no valid data provided
0x8b08 - unsupported CANCLKSEL value
0x8b09 - unsupported CANDIV value
0x8b0a - send in bus-off called
0x8b0b - timeout when put object in CAN chip

8 Related Literature

8.1 Bibliography

This list is not complete and only represents a selection of relevant literature.

Table 8-1

	Subject	Title
/1/	STEP7	Automation with STEP7 in STL and SCL Hans Berger Publisher: Vch Pub ISBN-10 3895783412 ISBN-13 9783895783418
/2/		

8.2 Internet Link Specifications

This list is not complete and only represents a selection of relevant information.

Table 8-2

	Subject	Title
\1\	Reference to the entry	http://support.automation.siemens.com/WW/view/en/EntryID
\2\	Siemens I IA/DT Customer Support	http://support.automation.siemens.com
\3\		

9 History

Table 9-1 Document history

Version	Datum		
V 1.1	17.06.09	AB	Changes for new HW revision COM168
V 1.0	16.03.09	AB	First version for delivery
V1.2	04.01.10	SC	- tested with WinAC RTX 2009
V1.3	02.12.10	AB	Improvement COM168 driver: pre-defined and user-defined baud rates are possible now
V1.4.0	20.07.12	AB	New document layout Added picture for HEX switch of COM168 board Nanobox IPC227D with on-board CAN interface additional supported (new function block, description of installation, etc.) TIA Portal example added Support for Windows 7