SIEMENS

SIMATIC

PROFINET
Guidelines for the Evaluation Kit ERTEC 200P PN IO V4.0

Programming and Operating Manual

07/2013

A5E03855331-02

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

⚠ DANGER

indicates that death or severe personal injury **will** result if proper precautions are not taken.

⚠ WARNING

indicates that death or severe personal injury **may** result if proper precautions are not taken.

⚠ CAUTION

indicates that minor personal injury can result if proper precautions are not taken.

NOTICE

indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

⚠ WARNING

Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E03855331-02
℗ 07/2013 Technical data subject to change

Preface

Preface

Purpose of the guidelines

These guidelines will quickly familiarize you with the use of the ERTEC 200P evaluation board in combination with the eCos operating system.

Target group of the guidelines

These guidelines are intended for software developers who want to use ERTEC 200P in connection with the eCos operating system for new products. This requires the following basic knowledge:

- Programming experience with C/C++
- Programming techniques such as multi-threading and callback routines
- Experience with PROFINET IO systems
- General knowledge of automation technology
- Basic familiarity with the STEP 7 configuration software

The developer is provided an evaluation kit consisting of an ERTEC 200P evaluation board and a PROFINET IO software package that he can use to test his PROFINET IO device application with the eCos operating system.

Structure of the guidelines

These guidelines describe the ERTEC 200P evaluation board and have the following structure:

- Section 1 Introduction
- Section 2 Scope of Delivery
- Section 3 Overview of the Guideline
- Section 4 Setting up the PROFINET IO Device
- Section 5 Setting up the Automation System
- Section 6 First Steps with PROFINET IO Device Communication
- Section 7 Application Examples
- Section 8 Creating Your Own Device
- Appendix: Abbreviations / Glossary of terms, References

These guidelines include the description of the PROFINET IO stack for the EB 200P evaluation board that is updated when needed. The current version is provided on the Internet (http://www.siemens.com/comdec).

Guide

The manual contains various navigation aids that allow you to find specific information more quickly:

- A complete table of contents as well as a list of all figures and tables are provided at the beginning of the manual.
- In the appendix you will find list of abbreviations and a glossary, which define important technical terms used in this manual.
- References to other documents are indicated by the document reference number enclosed in slashes ("/No./"). The complete title of the document can be obtained from the list of references in the appendix of the manual.

Conventions

Please observe notes labeled as follows:

Note

A note contains important information about the described product, about handling the product or about a specific section of the documentation that requires special consideration.

Additional support

For questions regarding the use of the described module that are not addressed in the documentation, please contact your Siemens partner or local offices.

Please send questions, comments and suggestions regarding this manual in writing to the specified e-mail address.

In addition, you will find general information, current product information, FAQs and downloads that can be useful on the Internet (http://www.siemens.com/comdec).

Technical contact for Germany / worldwide

Siemens AG	Phone: +49 911 750 2080
ComDeC	Phone: +49 911 750 4384
(http://www.siemens.com/comdec)	Phone: +49 911 750 2078
	Fax: +49 911 750 2100
	E-mail: (mailto:ComDeC@siemens.com)
Office address:	Postal address:
Würzburger Str. 121	P.O. Box 2355
90766 Fürth, Germany	90713 Fürth, Germany

Technical contact for the USA

PROFI Interface Center
(http://www.profiinterfacecenter.com)
One Internet Plaza
Johnson City, TN 37604

Phone: +1 (423) 262- 2576
Fax: +1 (678) 297- 7289
E-mail: (mailto:PIC.industry@siemens.com)

Table of contents

Introduction

1

PROFINET is an automation concept for implementing modular, distributed applications. PROFINET allows you to create automation solutions, which are familiar to you from PROFIBUS. PROFINET is implemented by the PROFINET standard for automation devices and by the engineering tool (STEP 7, TIA Portal). This means you have the same application view in engineering regardless of whether you are configuring PROFINET devices or PROFIBUS devices. As a result, programming of the user program is almost identical for PROFINET and PROFIBUS.

Siemens AG provides various development kits based on ERTEC 200P for PROFINET that can be used for quick and inexpensive implementation of PROFINET IO devices. The development kit assists users in developing their hardware and software.

The following circuit diagrams are stored on a CD for developing a PROFINET IO device hardware:

- Complete circuit diagram for the EB 200P
- Minimum configuration for a PROFINET IO device

For developing a PROFINET IO device software, a PROFINET IO stack with a user example is stored on a CD that includes the following functionality:

- Cyclic data exchange RT and IRT with a PROFINET IO controller
- Send and receive diagnostic and process alarms, plug and pull alarms
- Handling I&M0...4 data
- Assignment of IP addresses and device names via PROFINET
- Fast startup functionality
- Shared Device
- Media Redundancy (MRP)

Sound knowledge of PROFINET IO is required for implementing the firmware stack.

Scope of Delivery

2

2.1 Hardware Components

The hardware consists of the following components:

- EB 200P
  - Evaluation board for implementing a PROFINET IO device

2.2 Software Components

The software consists of several CDs with the following content:

- EK-ERTEC 200P
  - PROFINET IO Protocol Stack and application example for implementation of a PROFINET IO device with the eCos operating system
  - eCos V3.0 with Board Support Package for EB 200P
  - GSDML V2.3 example file for integration in STEP 7 HW Config
  - Documentation for the Evaluation Kit
  - Eclipse 3.4.1 including GDB Hardware Debugger Support, e.g., for use of OpenOCD JTAG debugger.

PROFINET IO Stack, application example and eCos are on hand as C-source code for a PROFINET IO device.

2.3 Content and Target Group of the User Description

This document is intended for developers of PROFINET IO devices. It contains:

- Overview of the structure of the EK-ERTEC 200P
- Description and configuration of the required tools
- Description and creation of the user example
- Description of an executable PROFINET overall system

This documentation does **not** include:

- Overview of PROFINET
- Description of the PROFINET protocol
- Detailed description of the PROFINET IO stack structure and processes

2.4 Other Information

The included application example has been tested on an Evaluation Board EB 200P.

2.5 Required Tools and Components

Required tools and components:

- "Wireshark Network Protocol Analyzer" to record Ethernet data packets
- USB mini-B cable for connecting a terminal program on the PC (e.g. TeraTerm, RS232-USB converter is integrated on the EB 200P)
- USB-RS232 driver included on the CD in the directory "\Tools\USB to RS232 driver"
- Ethernet cable
- JTAG debugger: Documentation and script files for an inexpensive OpenOCD-based JTAG debugger can be found after installation of the evaluation kit product CD on your PC in the subdirectory "...\PNIODevkit\OpenOcdDebug" or on the ComDeC website (http://www.siemens.com/comdec).

PROFINET IO controller and engineering tool

- A PROFINET IO controller is not included in the evaluation kit; it must be ordered separately. This could be, for example, a SIMATIC S7 CPU 317-2 PN/DP V3.2 or later. A suitable user example is included on the product CD and can be adapted as needed to other S7 CPUs.

Guideline Overview

3

The figure below shows all the steps that are necessary:

- Install the hardware and software
- Set up the terminal (e.g. TeraTerm) and install the USB-RS232 driver from the CD
- Commission the EB 200P and change the default MAC address, if necessary
- Commission the PROFINET IO controller and a suitable engineering system (not included in the evaluation kit)
- Import the GSD file into the engineering system
- Create the configuration example and load it into the PROFINET controller
- Create an executable PROFINET IO device application and load it onto the evaluation board (Note: An executable application example is already loaded on the EB 200P in the delivery state and is automatically started when power is switched on to the EB).

- Connect the PROFINET IO device to a PROFINET IO controller and test it
- Define your own device application and make adaptations to the GSD file, the configuration, and the device application

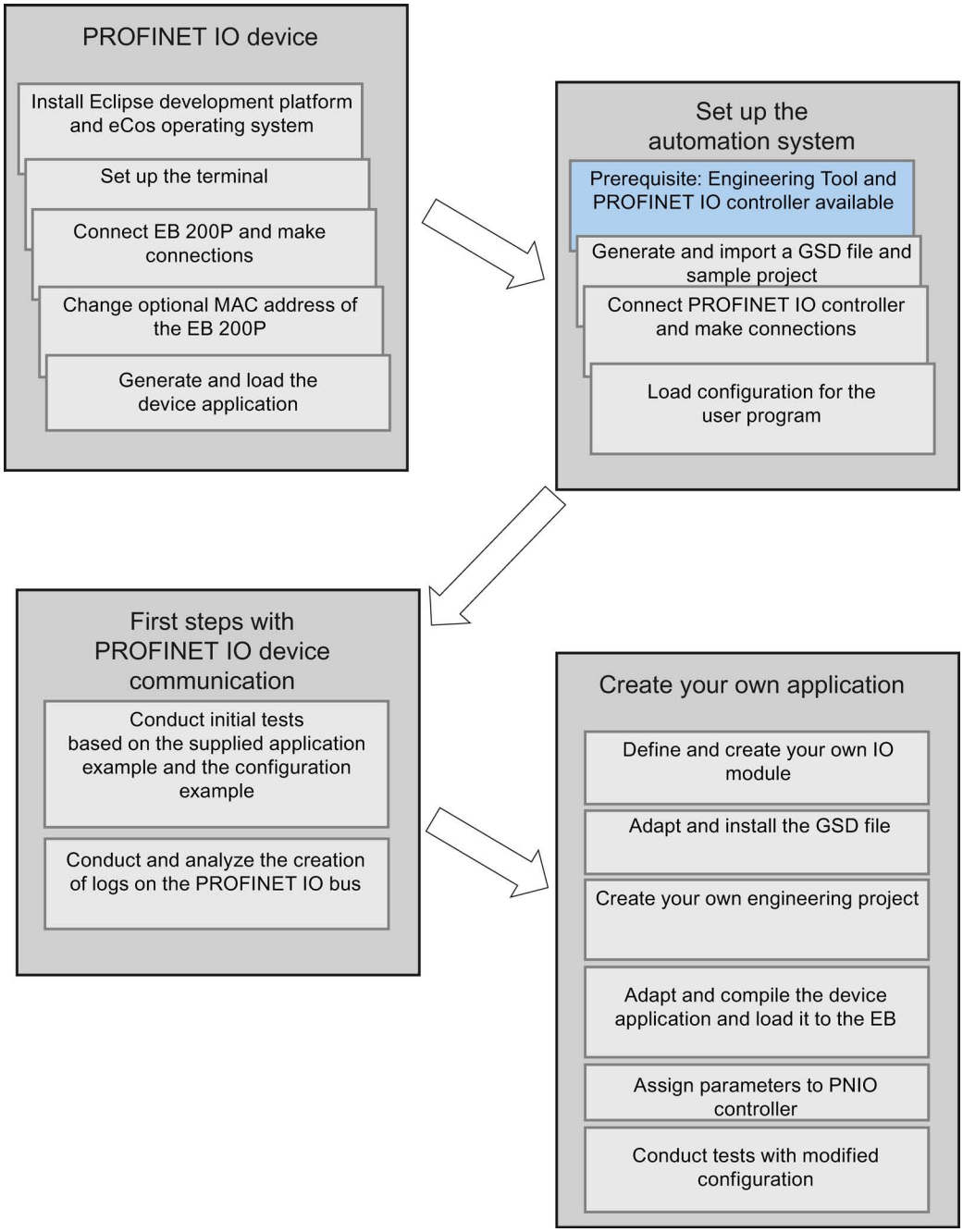


Figure 3-1 Overview: Required steps

Setting up the PROFINET IO Device

4

Before you can create a PROFINET IO device you must first install the eCos operating system with the PROFINET IO device stack and the device application on a Windows PC.

We recommend using the Eclipse development platform, which is also described in these guidelines. If you are using a different development platform, you must perform the necessary adjustments.

Note on installation

Note

All setup and installation instructions in the following sections refer to the EB 200P evaluation board.

4.1 eCos operating system and PROFINET IO Stack

4.1.1 Installation Instructions

The eCos operating system is royalty-free open source software.

Note

eCos has a special exception regarding the copyleft effect, for example, under the GNU General Public License V2 (or later):

"As a special exception, if other files instantiate templates or use macros or inline functions from this file, or you compile this file and link it with other works to produce a work based on this file, this file does not by itself cause the resulting work to be covered by the GNU General Public License. However the source code for this file must still be made available in accordance with section (3) of the GNU General Public License.
This exception does not invalidate any other reasons why a work based on this file might be covered by the GNU General Public License."

The terms of this license and other conditions must be met for use and transmission of eCos. A set of license conditions that are found in eCos are provided in the "Readme_OSS" file on the EvalKit data medium in the section on eCos.

4.1.2 Performing the Installation

1. Put the evaluation kit CD into the CD/DVD drive of your Windows PC.
2. Start the installation by selecting the "Install.bat" batch file in the root directory of the CD. The following license agreements are displayed before the installation:

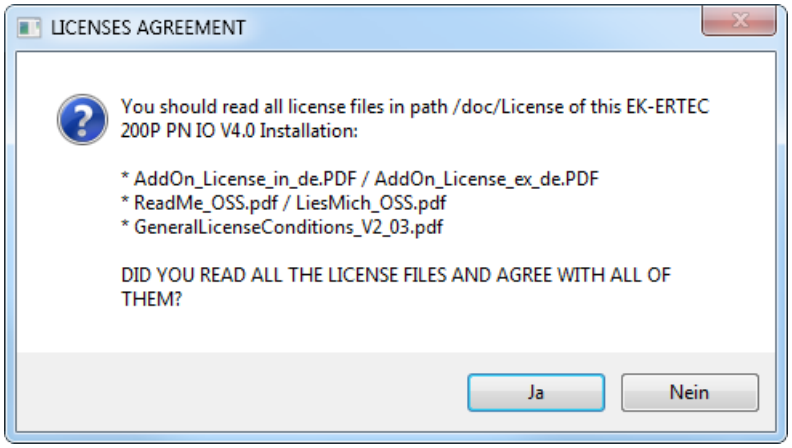


Figure 4-1 License agreements for installation of the eCos operating system

3. Click **Yes** to accept the license agreements. The installation will start.
(Click **No** if you do not accept the license agreements. Installation is aborted.)

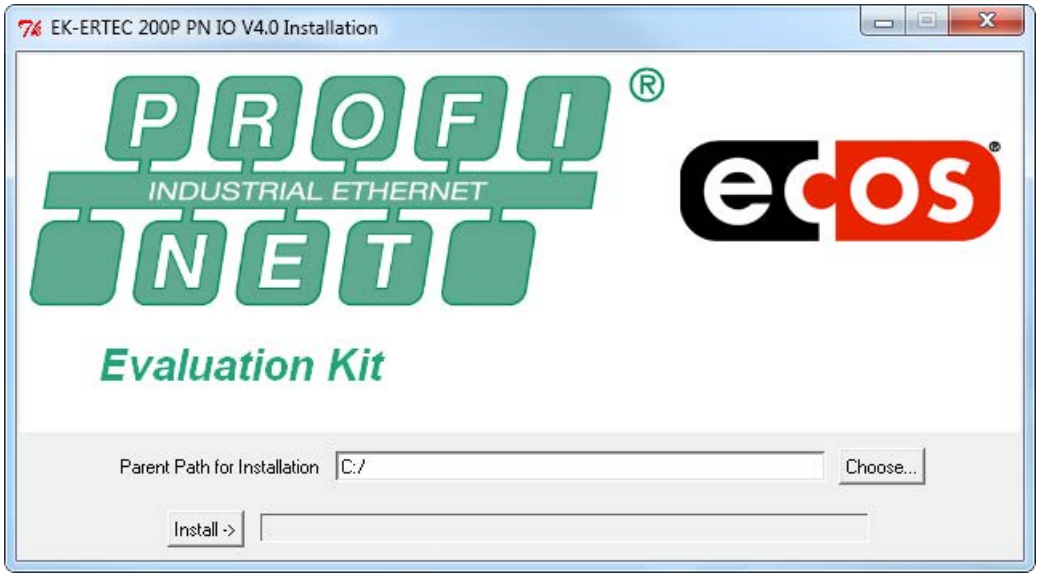


Figure 4-2 Evaluation kit software installation window for the eCos operating system

4. Use the **Choose** button to select the drive where the installation is to be performed. The **default** setting is the **C:/** drive.

5. Start the installation with the **Install** button.
   The following message appears on the screen:

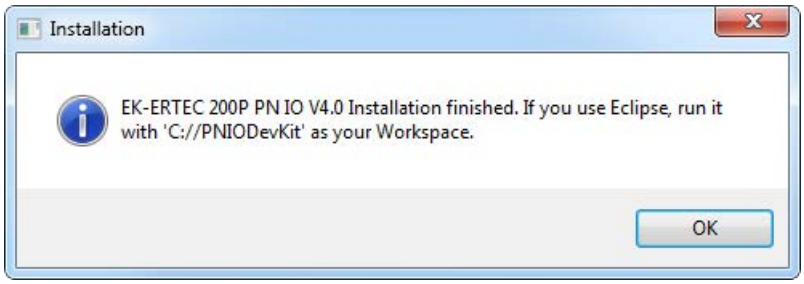


Figure 4-3 Installation window for software requirements

6. Confirm with "OK" to continue the installation. The installation progress is displayed with the progress bar.
7. At the end of the installation, you can choose whether you want to install Eclipse from the CD as well. If you confirm this with **Yes**, a self extracting Zip file, "eclipseInstall.exe", is started from the CD. Otherwise, the Eclipse installation is skipped.
   In Winzip Self-Extractor, select a directory of your choice in which Eclipse is to be installed and press "Extract" to continue installation.
   When installation is finished, exit the WinZip Self-Extractor by pressing the **Close** button.

Result: The installation is performed in the selected path "Drive:\PNIODevKit" as shown in the figure below.

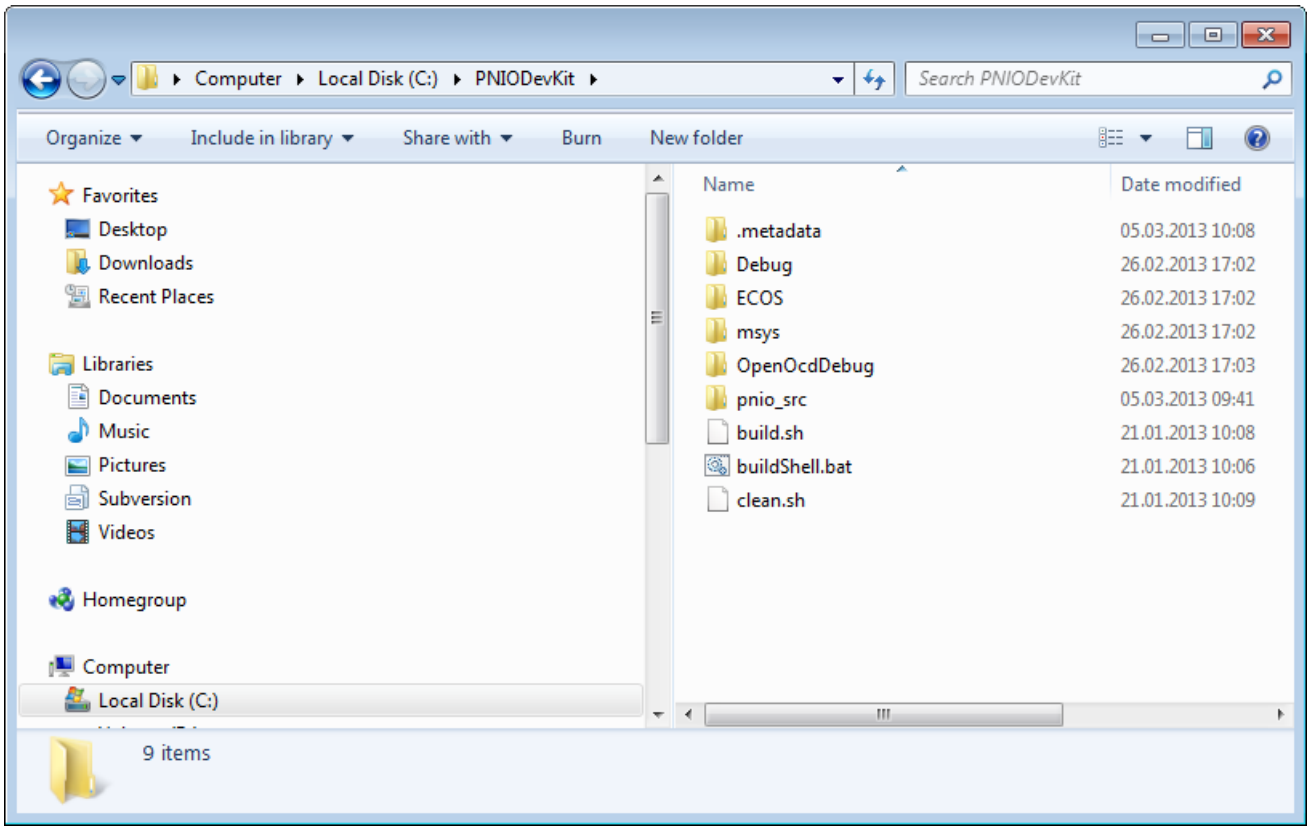


Figure 4-4 eCos installation directory

4.2 eCos with Eclipse development environment

4.2.1 Installing the Eclipsed development platform

We recommend using the Eclipse development environment when developing your own PROFINET IO device with the eCos operating system.

An Eclipse Version 3.4.1 with GDB Hardware Debugger Support (required for use of OpenOCD JTAG debugger) is available on the product CD in the directory "(...)\tools\eclipse".

Eclipse can be installed along with the EvalKit from the CD; the Eclipse installation directory can be freely selected.

Since Eclipse does not make any entries in the Windows registry, the Eclipse installation directory can be subsequently copied or moved as desired.

Eclipse is a royalty-free open source software, which can alternatively be downloaded from the website (http://www.eclipse.org/downloads). Special Open Source license conditions apply to the use of Eclipse.

Note

Users of Windows 7

Significantly longer generating times may result when used in combination with certain antivirus scanners. In such cases, we recommend excluding the PROFINET generating directories from the virus check through corresponding "exclude instructions" in the virus scanner.

Information about reference platforms used for testing Eclipse can be found after the installation of Eclipse in the **Eclipse Release Notes** file "(...)eclipse\readme\readme_eclipse.html".

4.2.2 Setting up the Eclipse development platform

Setting up Eclipse

Note

Requirement for starting Eclipse

In order to start Eclipse IDE on your Windows PC, Java Runtime Environment JRE1.5 or later must be installed. If the software is not installed on your computer, download it from the website (http://www.java.com) and install it. The installation instructions can be found on the same website.

1. Go to your "Eclipse" directory and run "eclipse.exe".
   Eclipse starts with the following window:

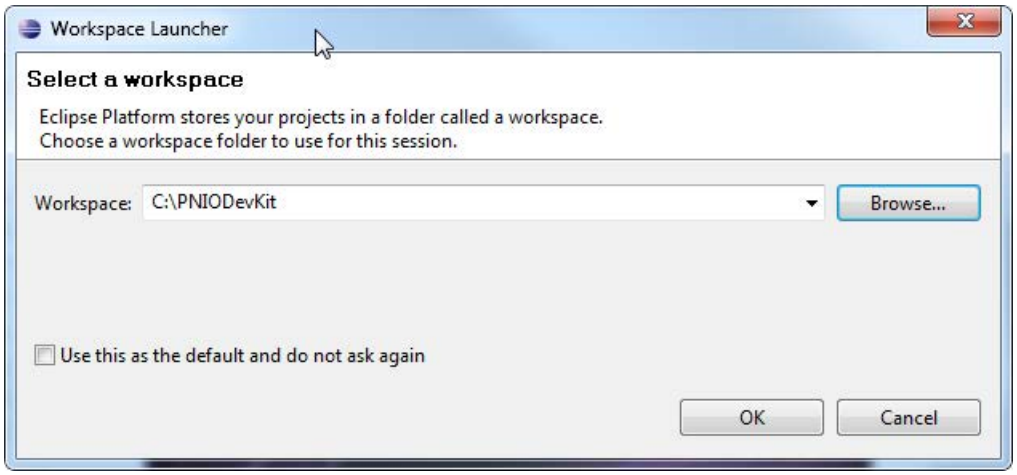


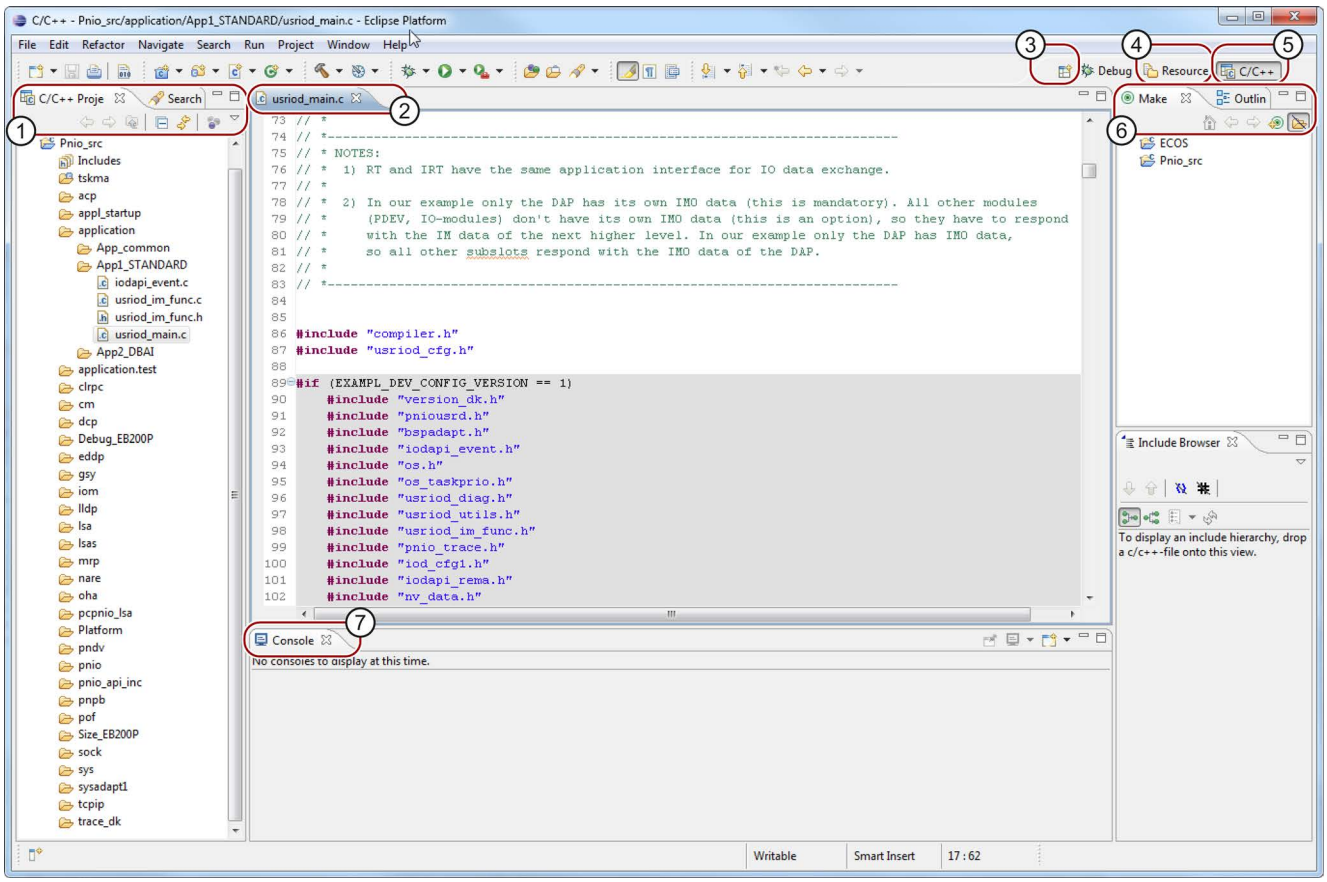
Figure 4-5 Setting up Eclipse

2. In the "Workspace" line, enter the working directory (e.g. "Drive:\PNIODevKit") in which eCos and the PROFINET IO stack were installed. Confirm with **OK**.

Note

This window is opened each time Eclipse is started. If you always want to start with the same settings, select the check box "Use this as the default ...". Then the window will no longer be displayed.

Once the workspace has been set up, Eclipse starts with the following dialog windows:



① Project Explorer with the project workspaces C/C++ Projects, Include Browser
② C/C++ Editor window with all open C and H files, e.g., usriod_main.c
③ Open perspective
④ Perspective: Resource
⑤ Perspective: C/C++ (default)
⑥ Dialog window. The following views are set by default: Make Targets, Outline
⑦ Dialog window. The following view is set by default: Console

Figure 4-6 Eclipse development platform with "C/C++" perspective

You can change the default setting and perspectives at any time or add views and perspectives.

Add views	Select the menu command **Window > Show View > Others**. Select the desired view and click **OK** to confirm. The view will now appear in the dialog window. Now you can move the view as described below.
Move views	Select the view you want to move and drag it into the desired dialog window.
Add perspectives	Click on the **Open Perspective** symbol and select the perspective you want to add. The selected perspective is then displayed as a symbol at the upper right ("Perspectives").

Further steps

After initially installing eCos and Eclipse, perform the following steps **once**:

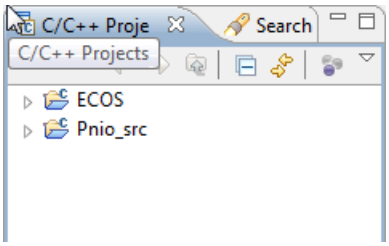
- Generating an eCos kernel for EB 200P (Page 22)
- Setting the build configuration (Page 23)
- Running the build process for the PROFINET IO stack EB 200P (Page 24)

4.2.3 Eclipse: Generating an eCos kernel

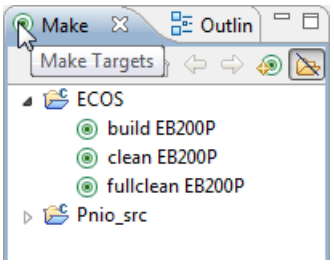
Generating an eCos kernel for EB 200P

1. To do this, select the dialog window "**Make Targets**".
2. Call "**build EB200P**" (see figure below),
   either by double-clicking on "**build EB200P**" or
   by right-clicking on "**build EB200P**" and selecting the **Build Make Target** command from the shortcut menu.

Result: The eCos kernel and the libraries for EB 200P are generated.



"C/C++ Projects" view



"Make Targets" view

With "**clean EB200P**", the object directories in the selected kernel are deleted.

With "**fullclean EB200P**", the build and include directories in the selected kernel are deleted.

Note

You only need to build the eCos kernel again if:

- The eCos kernel is modified
- "clean EB200P" was called
- "fullclean EB200P" was called

4.2.4 Eclipse: Setting the Build Configuration

Setting the build configuration

1. Select the project folder "Pnio_src"and press the right mouse button.
2. Now select a build configuration (e.g. **Build Configurations > Set Active > Size EB200P**)

Result: The build configuration is now set for all subsequent actions.
The active build configuration is shown with a check mark (see figure below).

Note

The following build configurations are possible:

- Debug_EB200P: "debug" for EB 200P
- Size_EB200P: "optimized size" for EB 200P

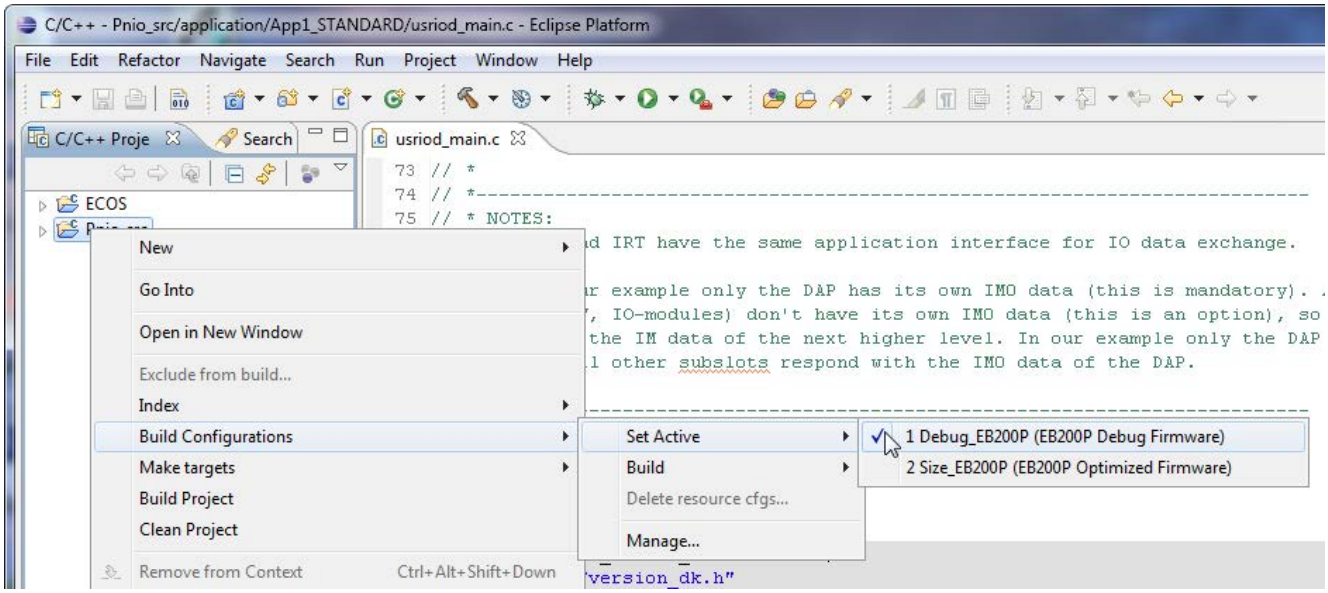


Figure 4-7 Setting the build configuration

4.2.5 Eclipse: Performing the PROFINET IO Application Build

Performing the build process for the PROFINET IO stack including the application for EB 200P

1. Press the right mouse button again in the "Pnio_src" project folder.
2. Select the command **Build Project** ①

Result: The build process is started and a project folder is generated for the corresponding build configuration.

- During the build process, the build progress is shown in the "Progress" window ③ and information is shown in the "Console" window ②.

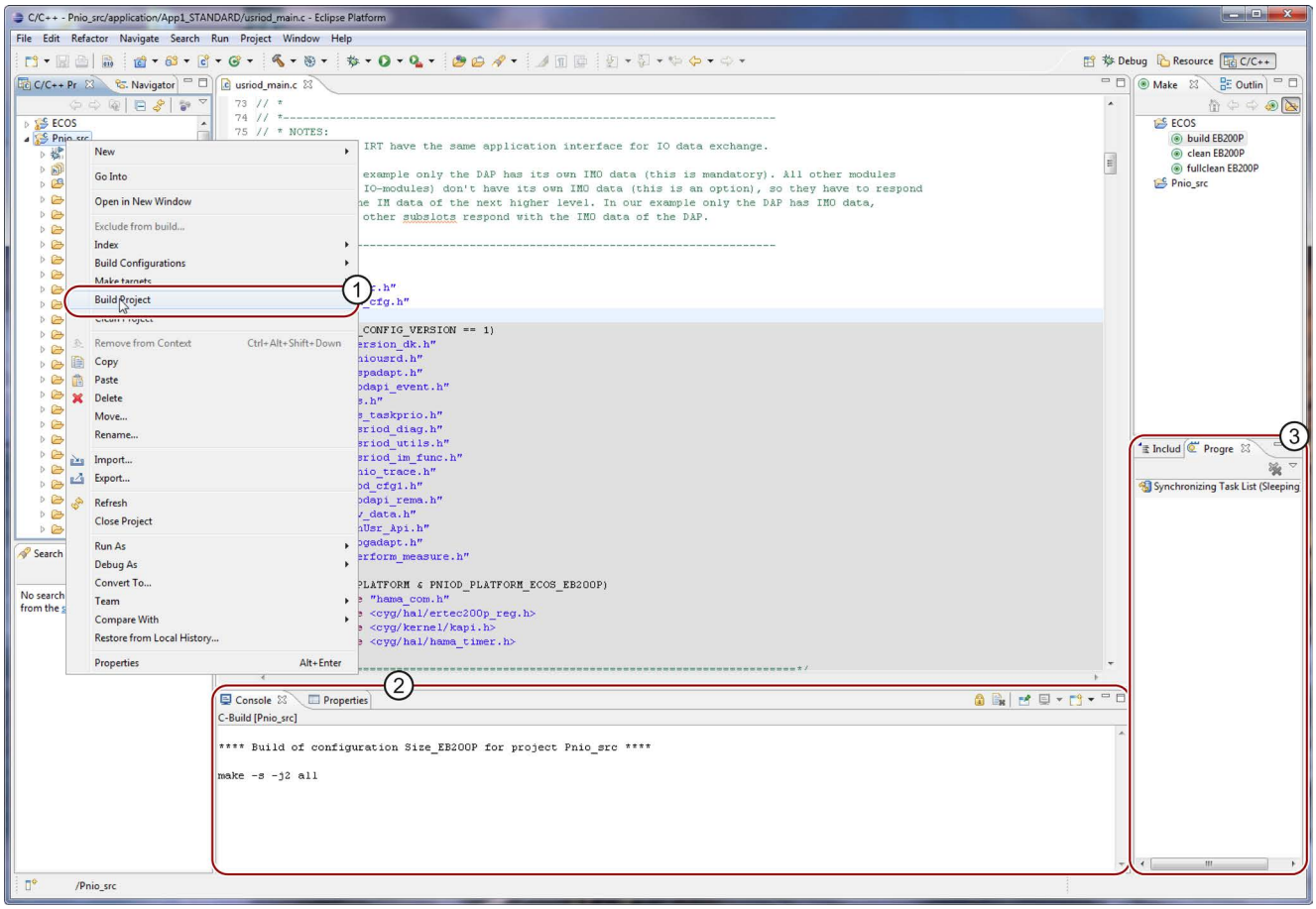


Figure 4-8 Performing the project build

- The build process is complete when the progress bar that is displayed during the build process is closed (see screenshot).

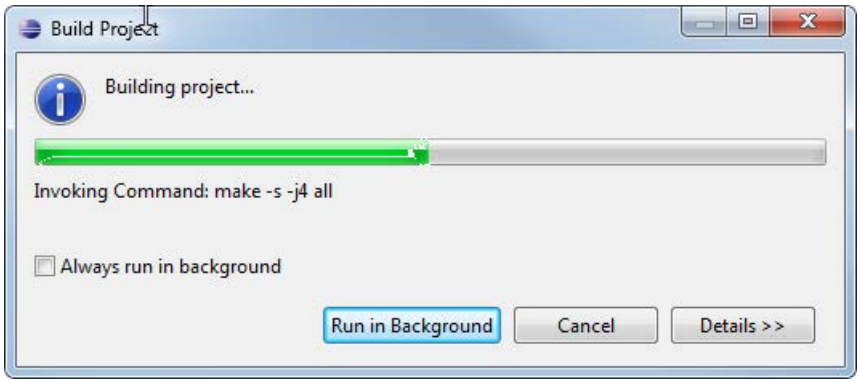


Figure 4-9 Progress bar during the build process

- If the build process is successful, a folder is created in the **"Pnio_src"** project folder, as selected in the build configuration. In our example, this is the folder "Size_EB200P" (see figure below).

  In addition to the object folders, the folder contains the two executable files that were generated:

  – PNIO4ECOS: executable PROFINET IO example (ELF file) for testing with a JTAG debugger (start the firmware from RAM)

  – ecos.bin: binary image file with executable PROFINET IO example for flash programming (BIN file)

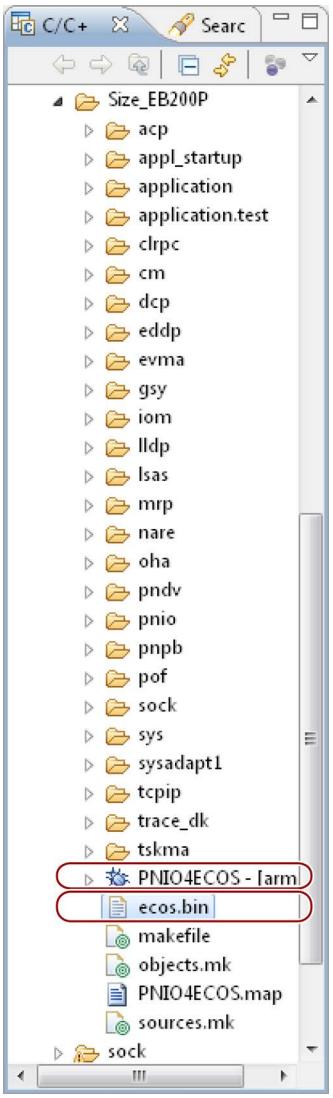


Figure 4-10 Project folder with Bin file and loadable file

4.2.6 Editing the Application Example

The "Pnio_src" folder includes the "application" folder with various examples for a PROFINET IO device. The following PROFINET IO examples can be selected:

Example	Value	Application file
RT Class 1, IRT Class 3 Example for ERTEC 200P	1	Usriod_main.c
DBAI example (Direct Buffer Access Interface)	2	Usriod_main_dbai.c

A sample application is selected in the **"usrapp_cfg.h"** file in the **"application\app_common"** folder.

The RT/IRT example is set by default to:

```
#define EXAMPL_DEV_CONFIG_VERSION 1
```

The RT/IRT example includes the following PROFINET IO device functions:

- Slot 1 → 64 byte input module
- Slot 2 → 64 byte output module
- Various module-related diagnostics
- Process alarm
- I&M0...4 functions

If you make changes to your PROFINET IO device functions in the application folder, you must perform the following steps:

1. Make any necessary changes to the application files and save the changed files.
2. Select the "Pnio_src" project folder and press the right mouse button.
3. Select the command **Build Project**, see Figure 4-8 Performing the project build (Page 24).

Result: The build process is started and all of the changed files are recompiled. After completion of the build process, the loadable PROFINET IO example device ("PNIO4ECOS") and the object file ("ecos.bin") are regenerated , see Figure 4-10 Project folder with Bin file and loadable file (Page 26).

See also

Eclipse: Performing the PROFINET IO Application Build (Page 24)

4.3 Configuring the Terminal Program for Message Outputs

In this example, the "TeraTerm" terminal program is used; it can be downloaded from the Internet as freeware. However, "Microsoft HyperTerminal" can be used as well.

1. Start the terminal program.
2. Perform the setup of the serial interface. Make the following settings for this:

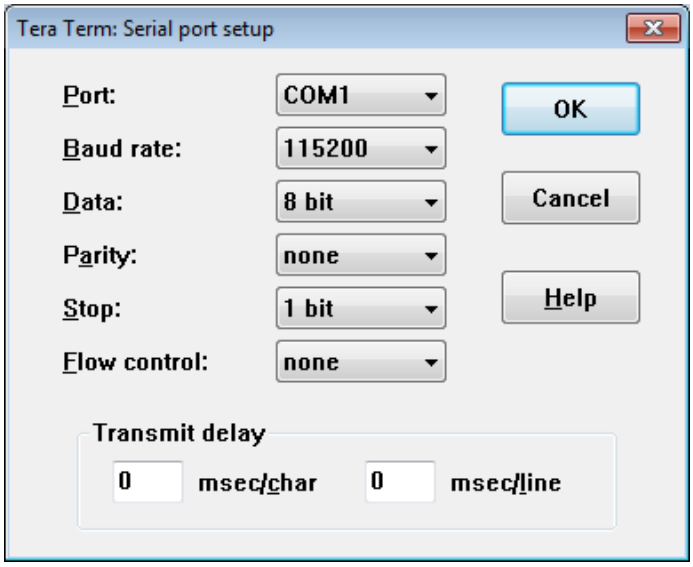


Figure 4-11 Serial port setup parameter for terminal

3. Click **OK** to activate.
4. Save the setup.

4.4 Commissioning ET 200P

4.4.1 Overview of the EB 200P Evaluation Board

Overview of connectors

Note

The EB 200P must be installed in the PC for correct operation.

Note

Connectors (except X1, X2 and X11) may only be plugged/pulled when power is switched off!

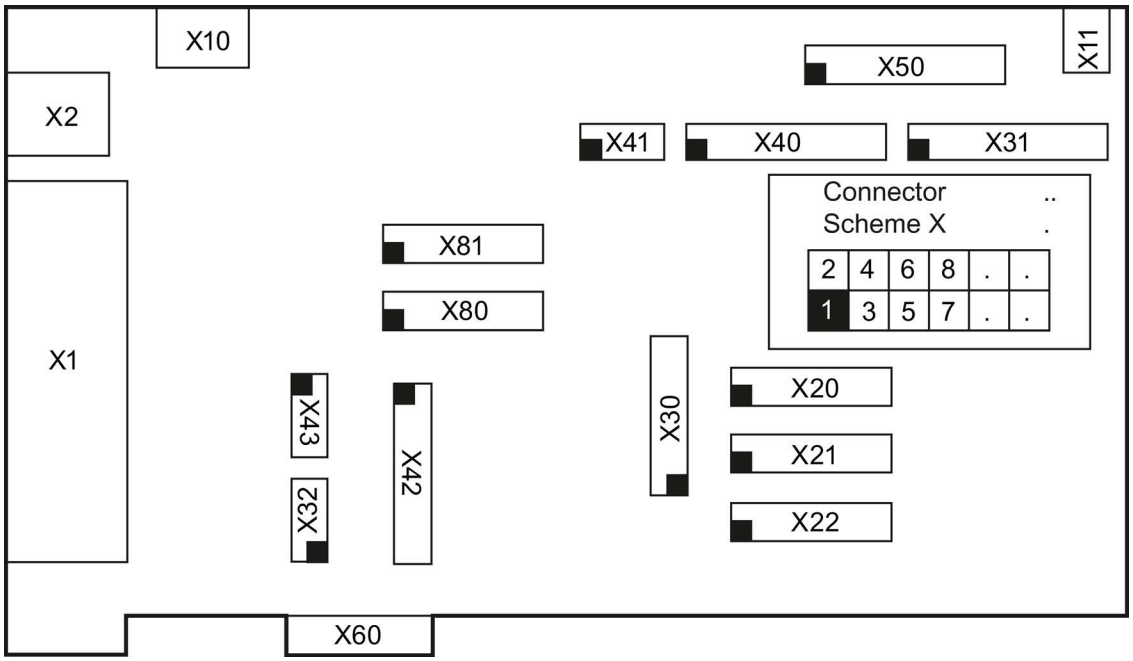


Figure 4-12 Overview of connectors for Evaluation Board EB 200P

Reference

For a detailed description of the Evaluation Board EB 200P, see /2/.

4.4.2 Establishing the Connections

The following adaptations need to be made to operate the Evaluation Board EB 200P:

1. Connect 24 V DC power supply to the X10 connector but do not switch it on yet.
2. Install the "RS232 via USB" driver for the PC from the CD. The driver is located in the subdirectory "\Tools\RS232 via USB driver". Now follow the instructions in the installation program.
3. Connect a free USB port of your PC to the USB port (RS232 via USB) of the EB 200P, see terminal connection in Figure 4-12 Overview of connectors for Evaluation Board EB 200P (Page 29).

Note

Use a USB mini-B cable for the serial connection.

4. Connect your JTAG debugger to the JTAG interface (**X31**) of the evaluation board as shown in the figure below.

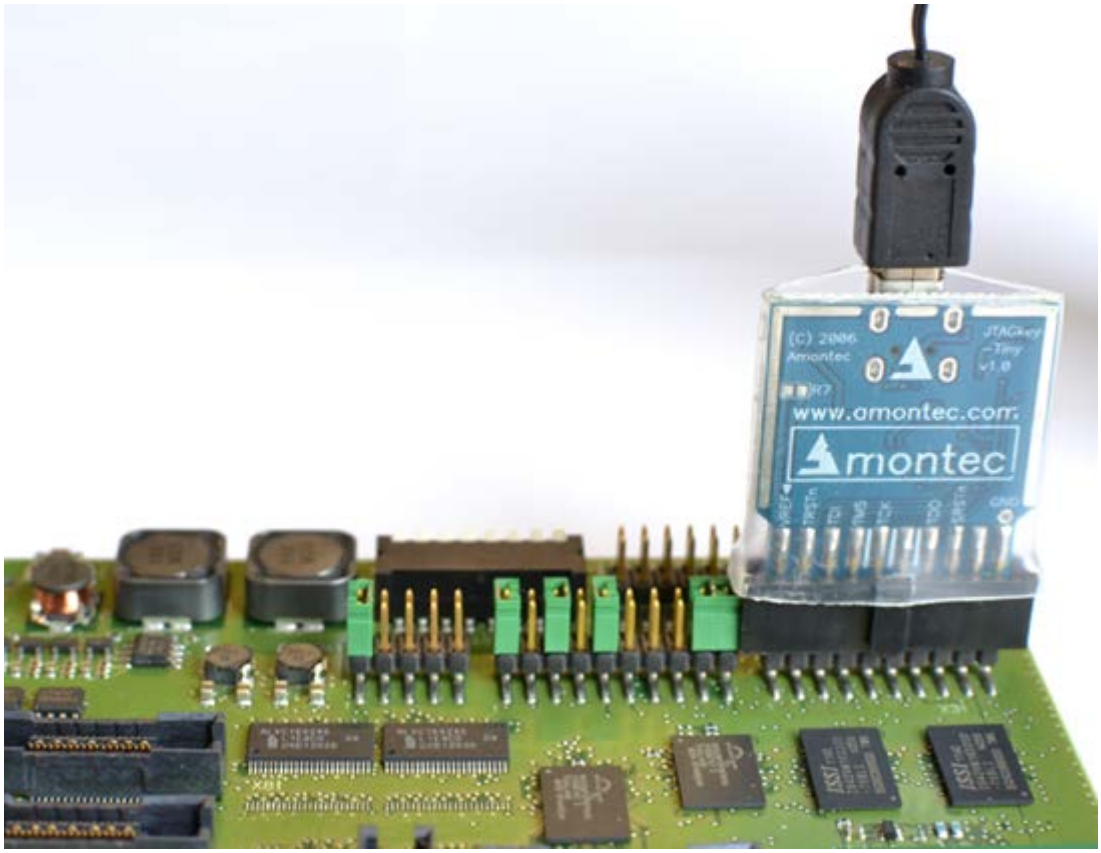


Figure 4-13 Connecting the JTAG debugger to the evaluation board

Note: The jumper settings in the figure above are **not** those of the delivery state.

Note

In the delivery state, the evaluation board already has the example application on the Flash and restarts automatically after power is restored. An update of the flashed application is possible via Ethernet.

Documentation and script files for an OpenOCD-based JTAG debugger can be found after installation of the product CD on your PC in the "Howto_use_Amontec_JTAGkey-Tiny_on_EB200P.pdf" file in the "PNIODevkit\OpenOcdDebug\" subdirectory or on the ComDeC website (http://www.siemens.com/comdec).

5. Connect a free Ethernet port of your PC to a switch port of the EB 200P (PN port 1 or PN port 2).

Note

Only Ethernet cables no longer than 100 m may be used for the LAN connection to/from the evaluation board.

6. Optional: To record Ethernet frames, connect the two TAP ports to a recording system. The EB 200P has an integrated TAP, which can be used to record Ethernet frame traffic. Connect the two TAP ports (port 1-A and port 2-B) each to a free port of the recording system for Ethernet frames. This can be a PC with two free Ethernet ports, for example, or a special hardware system for this purpose (e.g. CP1616 NetPROFI software).

   Note

   Only Ethernet cables no longer than 100 m may be used for the LAN connection to/from the evaluation board.

7. When all connections have been made, switch on the power supply.

EK-ERTEC 200P evaluation kit with PROFINET IO controller

The figure below shows the wiring for the PROFINET IO example.

An Ethernet TAP is integrated in the EB 200P. It is connected in the communication path between PROFINET IO controller and the PROFINET IO device. This TAP is used for passive decoupling of Ethernet frames in a switched network (port 1 only). With the Wireshark Network Protocol Analyzer, the PROFINET IO data packets can be recorded and analyzed.

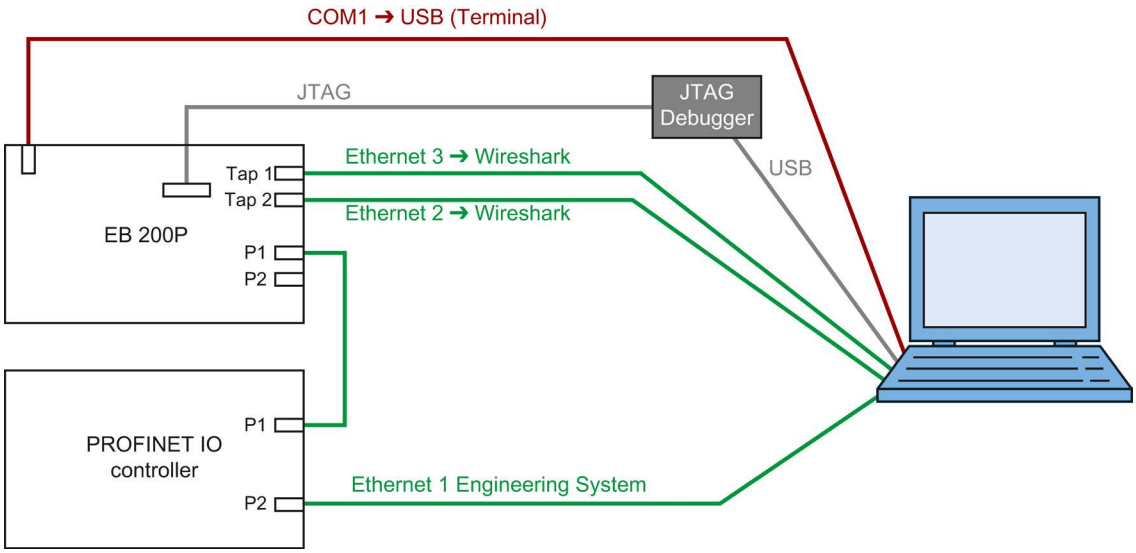


Figure 4-14 Configuration of a PROFINET IO controller/device for EK-ERTEC 200P with any PROFINET IO controller

See also

Overview of the EB 200P Evaluation Board (Page 29)

4.4.3 Setting the MAC Address

MAC address

In the delivery state of the EB 200P, a default MAC address is stored in the Flash. This is the same MAC address for all EB 200P modules. A unique MAC address is attached on the nameplate of the module.

Figure 4-15 MAC Address for Evaluation Board

This unique MAC address must be stored in the Flash on the EB 200P module. Changing the MAC address can be started from the user example.
Perform the following steps.

Changing the MAC address

1. Press the "N" key on the terminal, then <Enter>.
   The following text appears on the terminal:

```
input mac address
***Bsp_nv_data_restore:NvDataType=0***
*** done Bsp_nv_data_restore  ***
Current Ethernet address is: 00:0e:8c:b3:24:24
Modify all 6 bytes (board unique portion) of Ethernet Address
The first 3 bytes are manufacturer's default address block
00 -
```

2. Change the MAC address, as indicated on the nameplate:
   Enter Byte 1 to Byte 6 of the MAC address in succession, as indicated on the nameplate.
   After entering the 6th byte and pressing RETURN, the data are stored in the Flash.
   The following text appears on the terminal:

```
00 - 00
0e - 0e
8c - 8c
b3 - b3
24 - 24
24 - 24
store new Ethernet address  00:0e:8c:b3:24:24
***Bsp_nv_data_store:NvDataType=0***
***Bsp_nv_data_store:NvDataType=0:Data already in Flash***
***Bsp_nv_data_memfree***
*-------------------------------------------------------------*
* to activate the new mac address, perform a system restart *
*-------------------------------------------------------------*
```

Setting up the automation system

5

5.1 Setting up the automation system

The EB 200P device can in principle be operated with any certified PROFINET IO controller. On the CD you will find an example project for a SIMATIC CPU 317-2 PN/DP. It can be used as a template for creating a configuration for another PROFINET IO controller when needed.

If you use a SIMATIC CPU as controller, you must install STEP 7 as an engineering tool. The PROFINET IO controller and the engineering system are not included in the evaluation kit.

This section describes the basic steps for importing the example project for a SIMATIC S7 CPU 317-2 PN/DP included on the CD into the STEP 7 engineering system, installing the GSD file and loading the configuration to the controller.

This documentation can only describe the basic steps. You can find more detailed information about the PROFINET IO controller or the engineering system and how to operate them in the user documentation for the respective product.

5.2 Editing the project example for a PROFINET IO device

Project example

The "SIMATIC_STEP7" directory on the EK-ERTEC 200P PN IO CD contains completed STEP 7 projects suitable for the device application for a variety of application scenarios (Real-time, IRT, MRP, Shared Device, Isochronous IRT Application). They contain:

- The bus configuration with SIMATIC S7 CPU 317-2 PN/DP controller and PNIO device
- Configuration of the inputs and outputs of the PNIO device

Tip: We recommend to use these examples unchanged for the first commissioning.

Editing a project example

1. Importing a project file in STEP 7

   Open STEP 7 and import the file "Cpu317-2_EB200P_RT.zip". To do this, open the **File** menu in STEP 7 and select the **Retrieve** command.

2. Opening and editing an RT project

   In STEP 7, select the menu command **File > Open**.

   In the "Open" window, select the desired project and open it with **OK**.

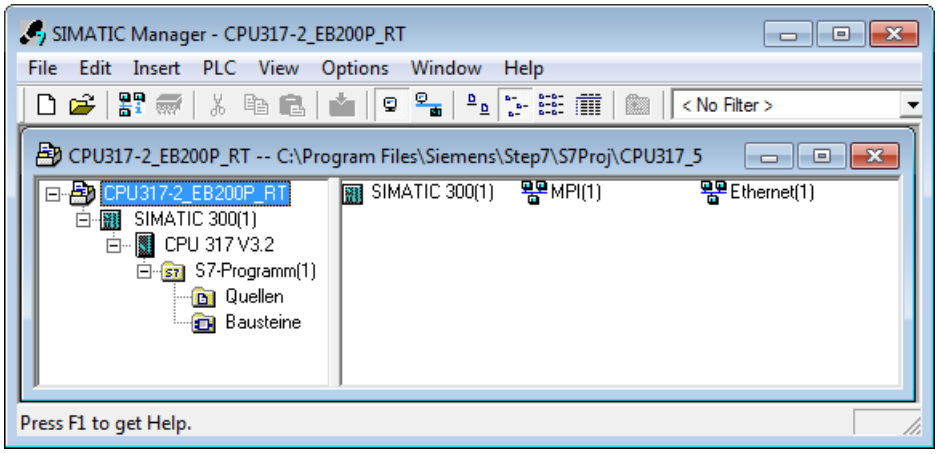


Figure 5-1 Opening and editing an RT project

3. Updating the hardware catalog

   Select "SIMATIC 300(1)" in the opened project, right-click and select the "Open..." command. This opens the hardware configuration tool, HW Config.

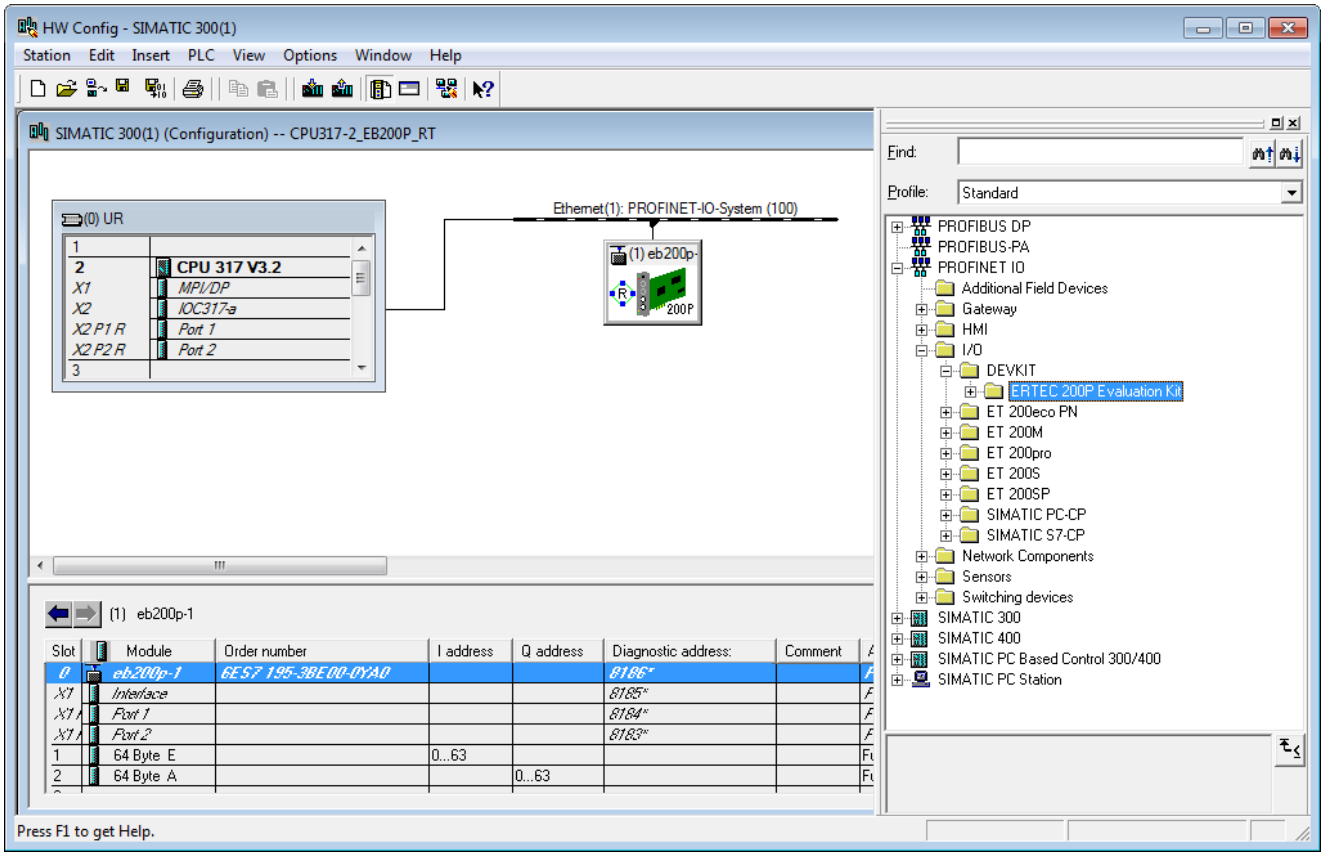


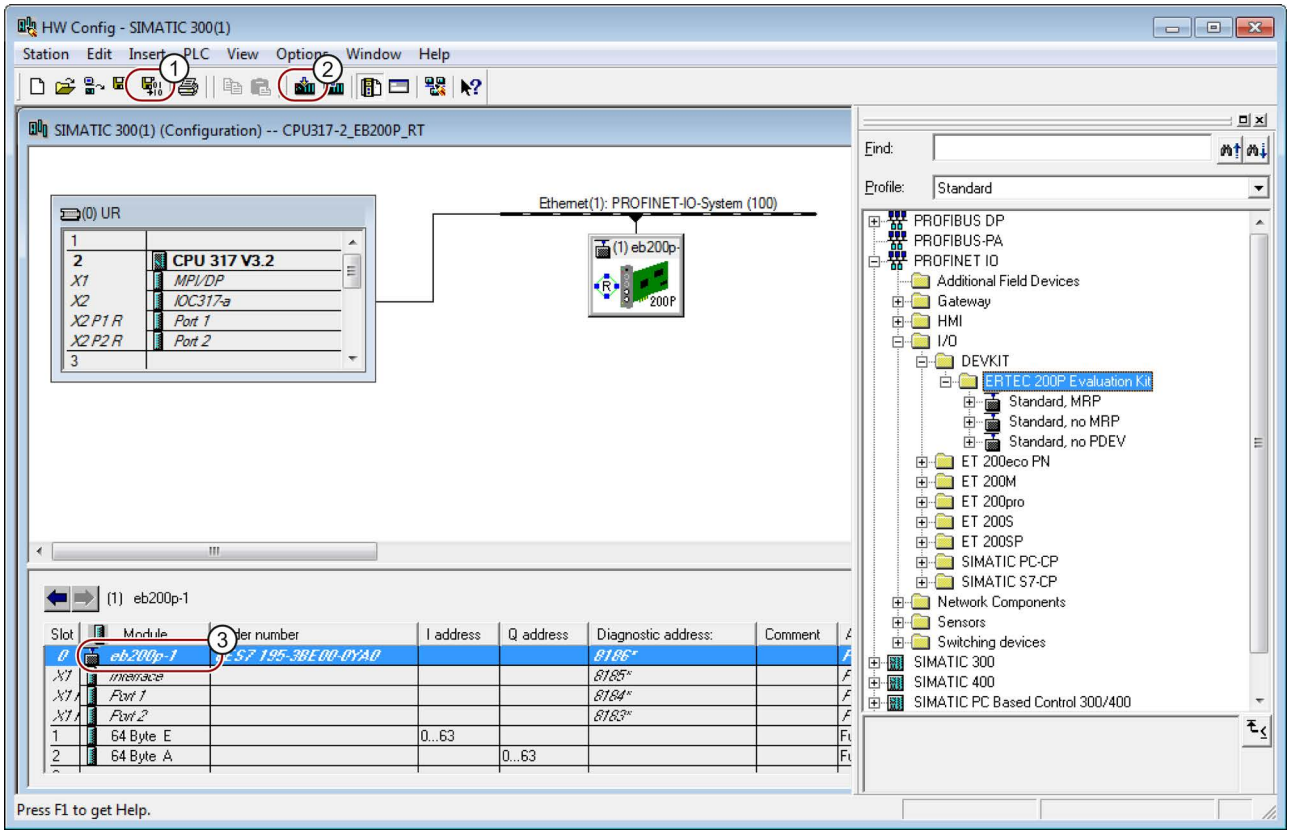
Figure 5-2 Updating the hardware catalog

- Select the menu command **Options > Install GSD Files**.
- In the window "Install GSD Files > Browse", select the directory "\GSDML" on the "EK-ERTEC200P_PNIO" CD and install the displayed GSD file by pressing the **Install** button.

5.3 Editing a PROFINET IO project with STEP 7 and loading it into CPU 317

The HW Config window displays the PROFINET IO bus configuration with the following components:

Component	Device name	Type	Configured parameters
CPU 317 V3.2	IOC317-a	PROFINET IO controller	IP: 192.168.20.171
EB 200P	eb200p-1	PROFINET IO device	IP: 192.168.20.182
		Slot 1 → 64 byte input module Slot 2 → 64 byte output module	Input address 0 - 63 Output address 0 - 63



① Save and compile
② Load configuration into controller
③ Device name "eb200p-1" for the device in this example

Figure 5-3 Configuring a PROFINET device with EB 200P in HW CONFIG

To make changes to the configuration, follow these steps:

1. Make change
   - Select the component you want to change and press the right mouse button. Select "Object Properties".
   - A window opens in which you can make your changes.
   - After all changes have been made, close the configuration in the "Station" menu with the command **Save and compile** (see figure above).
2. Download the configuration to the PROFINET IO controller
   - Connect all components as described in section Establishing the Connections (Page 30).
   - Download the bus configuration to the PROFINET IO controller by selecting the **Download** command in the **PLC** menu.

3. Check the connected PROFINET IO nodes
   - In HW Config, select the menu command **PLC > Edit Ethernet Node**.

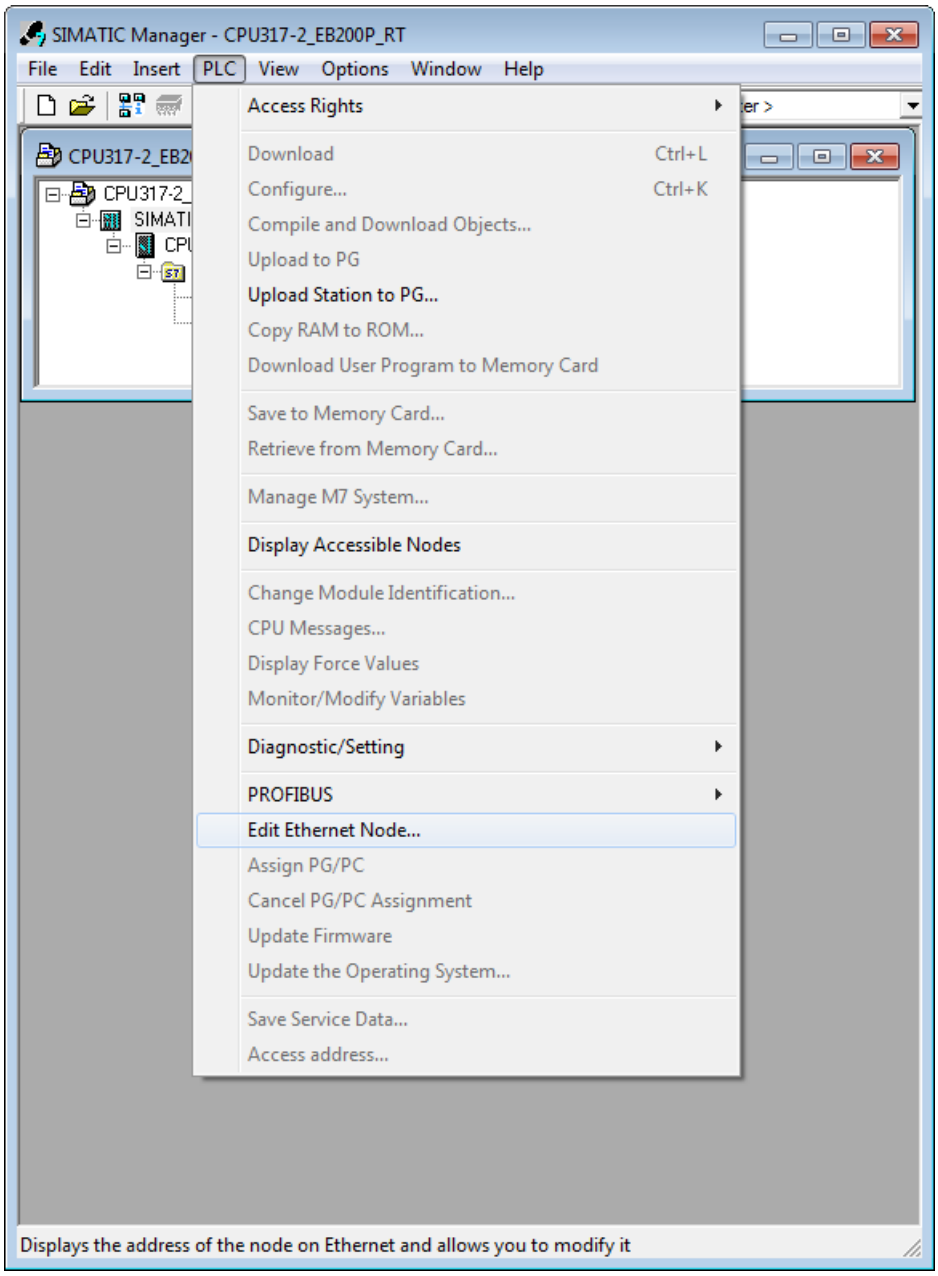


Figure 5-4 Checking all Ethernet nodes

   - Then click on the **Browse...** button. The configuration editor scans the Ethernet bus for a few seconds and shows all PROFINET IO nodes available online with MAC address, IP address (if available), device name and device type in the "Edit Ethernet Nodes" dialog window.

– Use the mouse to select the node you want to edit, and click **OK** to confirm (see figure below).

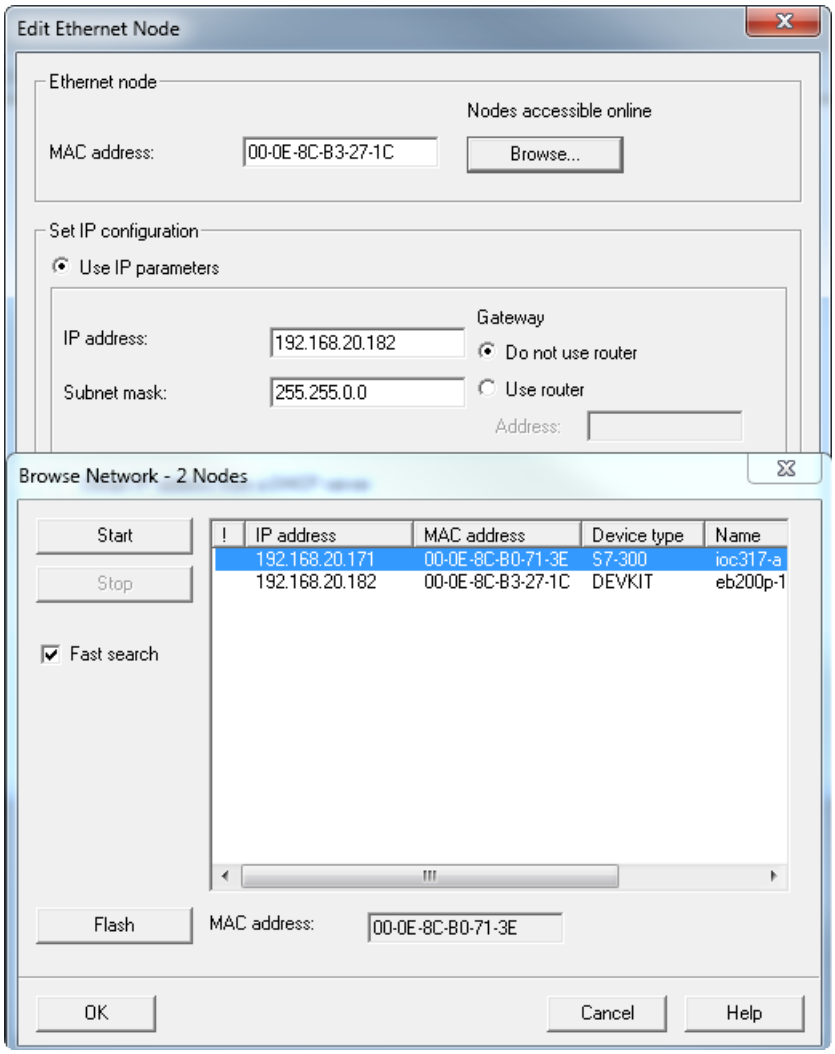


Figure 5-5 Displaying all nodes in the network

4. Assigning a device name / IP address to a selected PROFINET IO node

– In the "Assign device name" box, enter the desired device name and assign it using the **Assign Name** button (see figure below).

Note

If you change the device name, note the following:

- The device name must be unique within the plant.
- The device name on the device must match the one in the HW Config configuration.
- In the user example, the device application is notified on the device when you change the device name, and the new name is stored in the Flash.

– To change the IP address, you need to set the "Use IP parameters" option in the "Set IP configuration" field. Then, the desired IP address and subnet mask can be entered and assigned by pressing the **Assign IP Configuration** (see figure below).

Note

Normally, the controller sets the configured IP address on the device when the device name matches in the configuration and in the device. Therefore, setting the IP address is not required here.

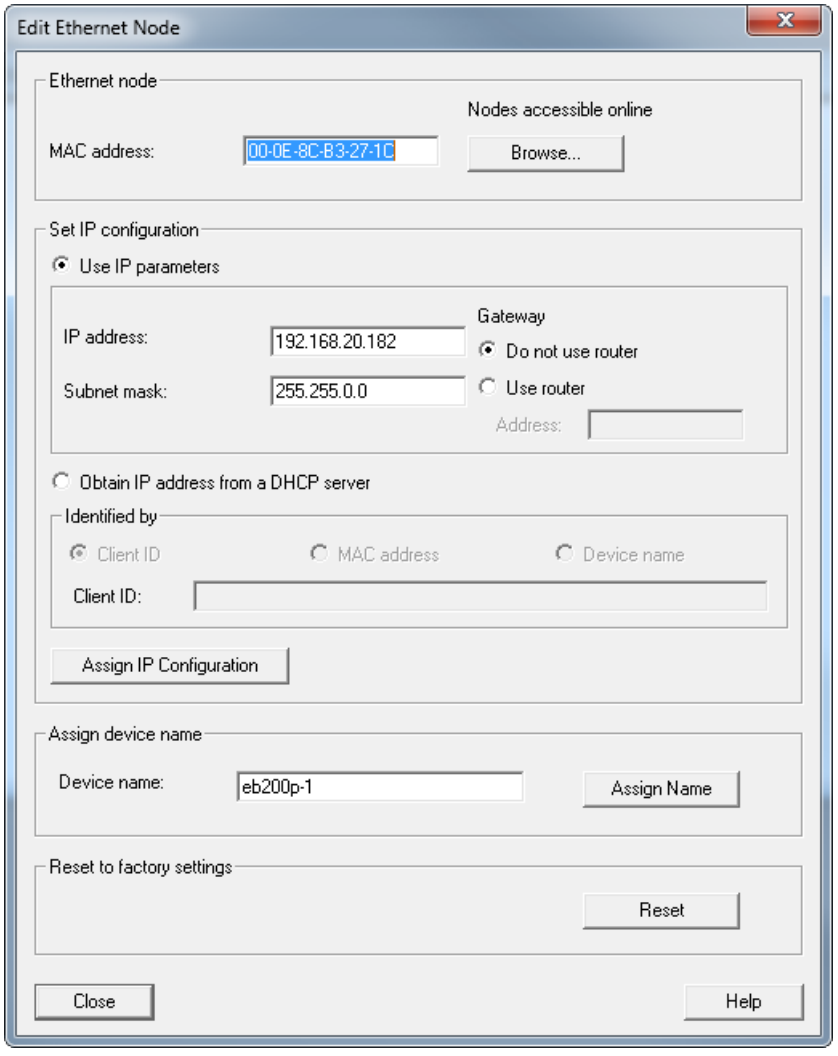


Figure 5-6 Assigning a device name and IP parameters

First steps with PROFINET IO device communication 6

This section describes the first steps toward your own device. We recommend leaving the provided device example and configuration unchanged when conducting initial tests. The example was tested on the EB 200P hardware platform. After you have familiarized yourself with the general PROFINET IO features, you can gradually customize your own device.

6.1 Starting the PROFINET IO device communication

The following requirements must be met:

- Cables have been installed and the power supply for the components is switched-on, see section Establishing the Connections (Page 30).
- A terminal program (e.g. TeraTerm) is started on the PC and adapted to the appropriate RS-232 interface (COM port) (see settings Configuring the Terminal Program for Message Outputs (Page 28)).
- The controller configuration has been loaded and the controller example application has been started.

The following lines of text have been output on the terminal, providing information about the process, for example:

- Restore data from non-volatile memory
- Initialization completed and application is started

```
***Bsp_nv_data_restore:NvDataType=2***
*** done Bsp_nv_data_restore  ***
***Bsp_nv_data_restore:NvDataType=0***
*** done Bsp_nv_data_restore  ***
***Bsp_nv_data_memfree***
***Bsp_nv_data_restore:NvDataType=0***
*** done Bsp_nv_data_restore  ***
***Bsp_nv_data_memfree***
***Bsp_nv_data_restore:NvDataType=0***
*** done Bsp_nv_data_restore  ***
***Bsp_nv_data_memfree***
***Bsp_nv_data_restore:NvDataType=3***
*** done Bsp_nv_data_restore  ***
##REMA DATA STORE total Bufsize=700
SubPlugResponse(0) = OK
SubPlugResponse(1) = OK
SubPlugResponse(2) = OK
SubPlugResponse(3) = OK
SubPlugResponse(4) = OK
SubPlugResponse(5) = OK
SubPlugResponse(6) = OK
SubPlugResponse(7) = OK
***Bsp_nv_data_memfree***
***Bsp_nv_data_memfree***
AutoFlash Task started...
TCP interface initialized
TCP interface wait on connection...
PNIO DEVKIT appl.-example no.9, V4.0.100.1, 2012-12-21  10:57
SYSTEM STARTUP FINISHED, OK
```

At this time, the startup of the PROFINET IO stack completed and the communication between controller and device starts automatically.

The following information is displayed on the terminal when a connection is established, for example:

- Number of modules and module information
- End of the configuration (for each submodule)
- First valid data or IOxS from PNIO controller received (for each submodule)

```
##CONNECT_IND HostIp = 0x0 AR = 0 AR type = 16
##OWNERSHIP_IND AR = 1 number of submodules = 5
  Api=0 Slot=0 Sub=1 ModID=3 SubID=1 OwnSessKey=1025 isWrong=0
  Api=0 Slot=0 Sub=32768 ModID=3 SubID=2 OwnSessKey=1025 isWrong=0
  Api=0 Slot=0 Sub=32769 ModID=3 SubID=3 OwnSessKey=1025 isWrong=0
  Api=0 Slot=0 Sub=32770 ModID=3 SubID=3 OwnSessKey=1025 isWrong=0
  Api=0 Slot=2 Sub=1 ModID=49 SubID=1 OwnSessKey=1025 isWrong=0
***Bsp_nv_data_store:NvDataType=2:Data already in Flash***
##PARAM END for Api=0, Slot=0, Sub=1, ArHndl=1, Session=1025
##PARAM END for Api=0, Slot=0, Sub=32768, ArHndl=1, Session=1025
##PARAM END for Api=0, Slot=0, Sub=32769, ArHndl=1, Session=1025
##PARAM END for Api=0, Slot=0, Sub=32770, ArHndl=1, Session=1025
##PARAM END for Api=0, Slot=2, Sub=1, ArHndl=1, Session=1025
##REMA SHADOW MEM STORE total memsize=700
***Bsp_nv_data_store:NvDataType=3***
##AR IN-Data event indication received, ArHndl = 0h, Session = 0h
new IO controller provider status = 0x80 in slot 2, subslot 1
***Bsp_nv_data_store:NvDataType=3:Data already in Flash***
##CONNECT_IND HostIp = 0x0 AR = 1 AR type = 1
##OWNERSHIP_IND AR = 2 number of submodules = 1
  Api=0 Slot=1 Sub=1 ModID=48 SubID=1 OwnSessKey=1026 isWrong=0
new IO controller provider status = 0x80 in slot 2, subslot 1
##PARAM END for Api=0, Slot=1, Sub=1, ArHndl=2, Session=1026
##AR IN-Data event indication received, ArHndl = 1h, Session = 0h
new IO controller consumer status = 0x80 in slot 1, subslot 1
```

A PROFINET IO bus recording was carried out for startup. This file can be opened with the Wireshark Network Protocol Analyzer and analyzed.

Bus recording, see file attachment "Start_Up_RT.pcap".

6.2 Testing the PROFINET IO device communication

Various PROFINET IO features can be activated/deactivated with the keyboard inputs. To display the Help menu, enter **?** on the terminal.

The following list appears with the following information (for example):

- Start Help menu
- Send PN IO process alarm
- Activate/deactivate PN IO diagnostics
- Display IO data
- Switch LEDs
- Set MAC Address
- Plug and pull modules/submodules

```
COMMAND LIST    PNIO DEVKIT appl.-example no.9, V4.0.100.1, 2012-12-21  10:57

CONTROL:
'a'      send process alarm on slot1, subslot1
'B / b'  send channel diag-alarm 'line break' appear/disapp. on slot1, subsl.1
'C / c'  send generic diag-alarm appear/disapp. on slot2, subslot1
'E / e'  send extended channel diag-alarm appear/disapp. on slot2, subsl.1
'Q / q'  increment/decrement input data values
'f'      download firmware via TCP and store into flash
'G'      send upload alarm (backup) to ipar-server
'g'      send retrieval alarm (restore) to ipar-server
'j'      print cyclic io data
'K'      set RS232 output trace level
'k'      set MEM output trace level
'L'      set LED (1..N, 0:all) on
'l'      set LED (1..N, 0:all) off
'M'      set MAC address and store in flash
'P'      enable user printf output on serial console
'p'      disable user printf output
'S'      plug a submodule
's'      pull a submodule
'T'      set trace level for a package
't'      set trace level for a single subsystem
'z'      measure system performance
'0'      Perform System Reboot
```

This list can be expanded by the user. See files in the folder "application":

- "\App1_Standard\usriod_main.c "
- "\App2_DBAI\usriod_main_dbai.c"

Familiarize yourself with the PROFINET IO protocol:

1. Activate a selection of commands in the command list and record them with the Wireshark Network Protocol Analyzer.
2. Then analyze the recorded commands.

Application Examples

7

The evaluation kit includes various application examples for establishing a PROFINET IO communication between a controller and a device (EB 200P). The examples are compatible with each other ("ready to use") and affect the following functionalities:

- RT communication
- IRT communication
- IRT communication with isochronous application
- Media Redundancy (MRP)
- Shared Device, i.e., access of two controllers to one device
- TCI
- PROFIenergy

A suitable device application example (see Examples for Device Applications (Page 48)), a GSD file (see GSD files (Page 59)) and a configuration (see Controller application examples (Page 60)) are always included on the CD with regard to these functions from the user perspective (use cases). The combination of suitable device user examples and configuration (e.g. for SIMATIC CPU 3xx) is documented in the respective use cases.

7.1 Examples for Device Applications

Three different application examples are implemented in the source code in the "Application" directory; they differ mainly in the way they access the IO data.

- The example code for access via the standard interface (SI) is stored in the "App1_Standard" subdirectory.
- The code for the IO data access via the Direct Buffer Access Interface (DBAI), on the other hand, is stored in the "App2_DBAI" directory.
- "App3_IsoApp" contains a user example for IRT with isochronous application, including the T_IO_Input and T_IO_Output handling.

All common files that are shared by the user examples are contained in the"App_common" directory.

All functions, such as RT/IRT mode, PROFIenergy, redundancy, etc. are integrated in these examples. If necessary, certain compiler switches in header files, which are either stored in the application directory itself or in the system adaptation "(...)\sysadapt1\cfg," will have to be changed.

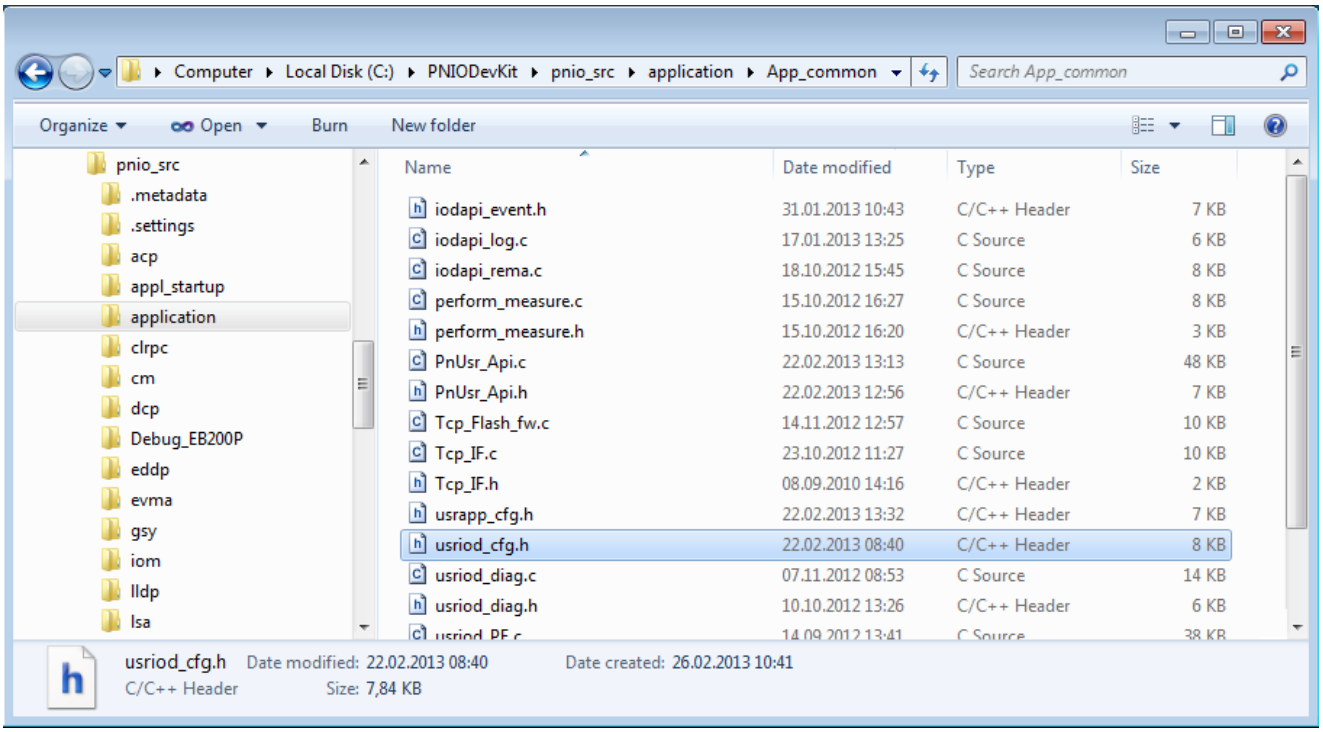


Figure 7-1 Configuration file "usriod_app.h"

Other examples "App3_xxx", "App4_yyy", etc. can be integrated. The selection of the user example is made with the compiler switch, for example

```
#define EXAMPL_DEV_CONFIG_VERSION
```

in the "\App_common\usrapp_cfg.h" file.

The user examples are operated with a terminal program, which is connected via USB to the UART of the ERTEC 200P (terminal X11). The interface parameters of the UART are:

"115 kBaud, 8bit data, no parity, 1 stopbit, no flow control"

The basic structure of the samples and the control console are largely the same. They each consist of two tasks "mainAppl()" and "Task_CycleIO," which are each contained in the file "usriod_main.c" (SI) or "usriod_main_dbai.c" (DBAI). The functions of these tasks are to:

- "mainAppl()":
  - Initialize and start of the PNIO stack
  - Transfer the actual module configuration to the PNIO stack
  - Create and start "Task_CycleIO"
  - Operate the console interface in a continuous loop
- "Task_CycleIO()":
  - Reads the output data from the PROFINET IO controller and transfers it to the physical outputs, returns the IOCS value to the stack (the IOCS indicates whether the output data could be processed correctly).
  - Reads the physical input data and transfers it together with the Provider State IOPS to the PROFINET IO stack (the IOPS indicates whether the read data is valid).
  - Wait until the next cycle

Note

It is important that all tasks which communicate with the PROFINET stack via the user interface are created with the function "OsCreateThread()", and that they are assigned a message queue with "OsCreateMsgQueue()".

7.1.1 Device application example for the standard interface (SI)

This application example is stored on the Flash of the evaluation board in the delivery state and starts automatically when the power is switched on. In addition, an executable image is stored in the subdirectory "\BootableBinary".

The "SI" is based on a callback concept, which means a separate callback function is executed for each submodule after activation of an IO data communication (read or write). The handling is very easy for the application because it does not have to know the structure of the IOCR or have to manage the individual Application Relations (ARs). The relationship with the data from the perspective the application is data-submodule based.

However, the "DBAI" has an AR and IOCR based application perspective regarding the data. The application must manage each of ARs and carry out a separate exchange of data for each AR. For this purpose, the application keeps the pointer on the input and output IOCR and can read and write the IO data or the IOPS/IOCS directly contained therein. This requires, however, that the application recognizes the individual ARs as well as the structure of the data IOCRs contained in these. The application receives these via the connect message frame.

Additional information about SI and DBAI can be found in the interface description /3/.

A manual for this example can also be found in section First steps with PROFINET IO device communication (Page 43).

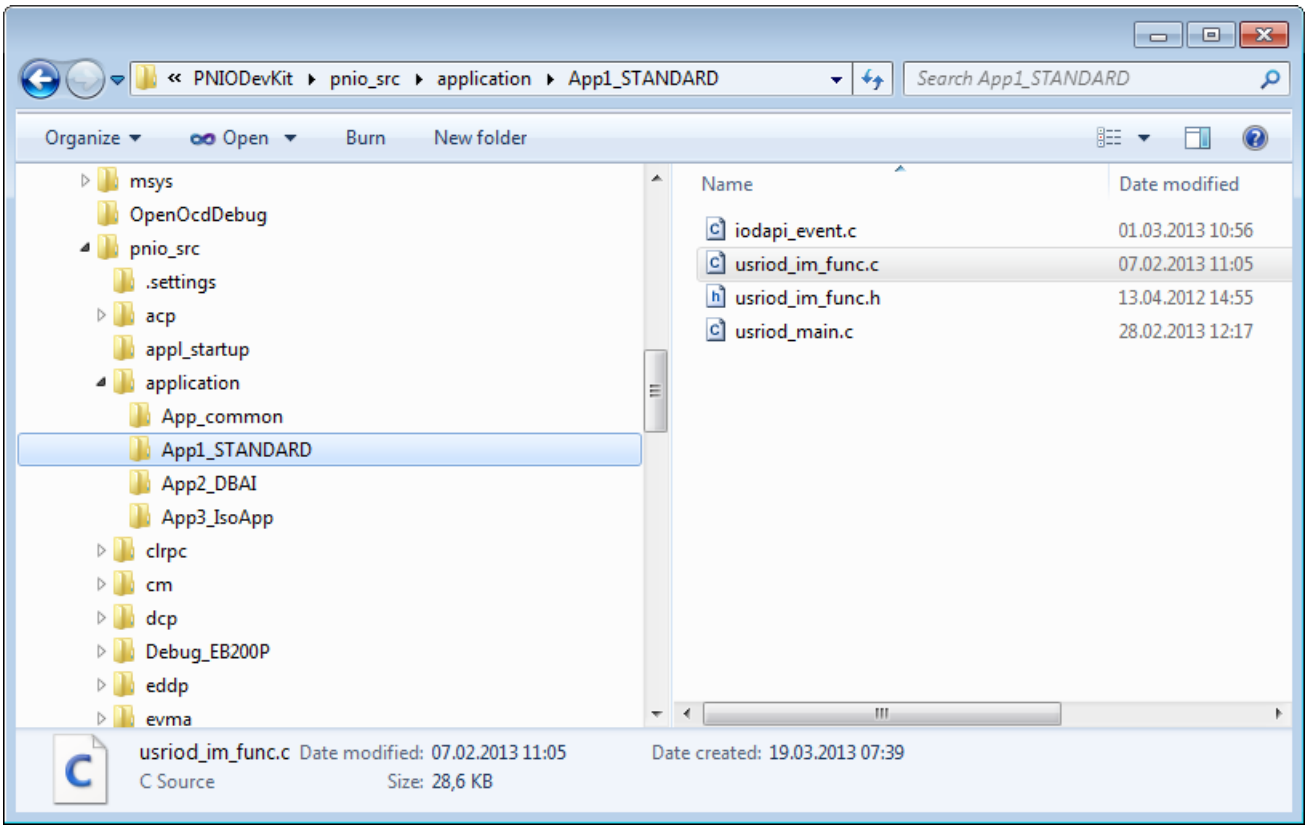


Figure 7-2 Files of the Standard Interface Example (SI)

This application example is included, if

```
# define EXAMPL_DEV_CONFIG_VERSION 1
```

is set in the "\Application\App_Common\usrapp_cfg.h" header file.

7.1.2 Device application example for the Direct Buffer Access Interface (DBAI)

The specific source code for this application example can be found in the directory "(...)\Application\App2_DBAI".

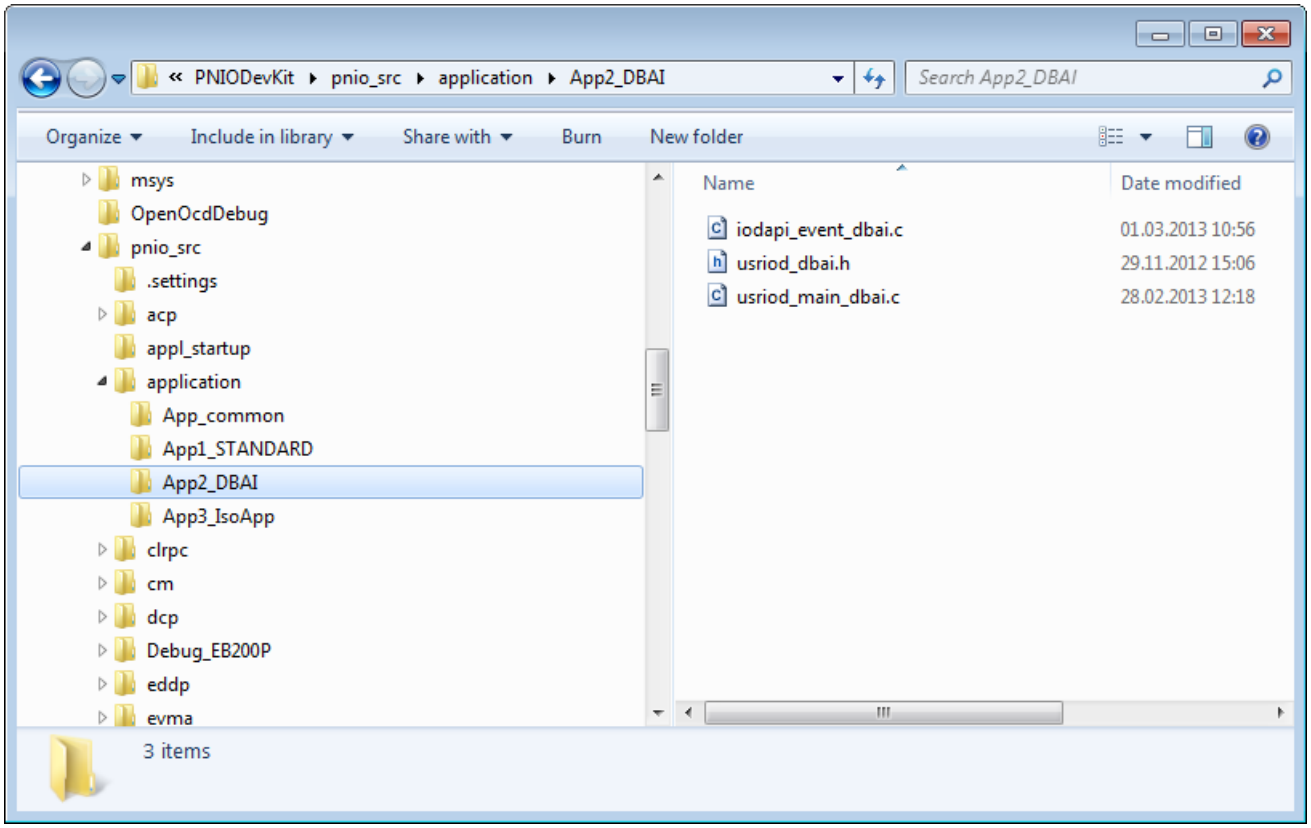


Figure 7-3 Files of the DBAI example

This application example is included, if

```
# define EXAMPL_DEV_CONFIG_VERSION 2
```

is set in the "\Application\App_Common\usrapp_cfg.h" header file.

7.1.3 Device application example for an isochronous application with IRT

The specific source code for this application example can be found in the directory "(...)\Application\App3_IsoApp".

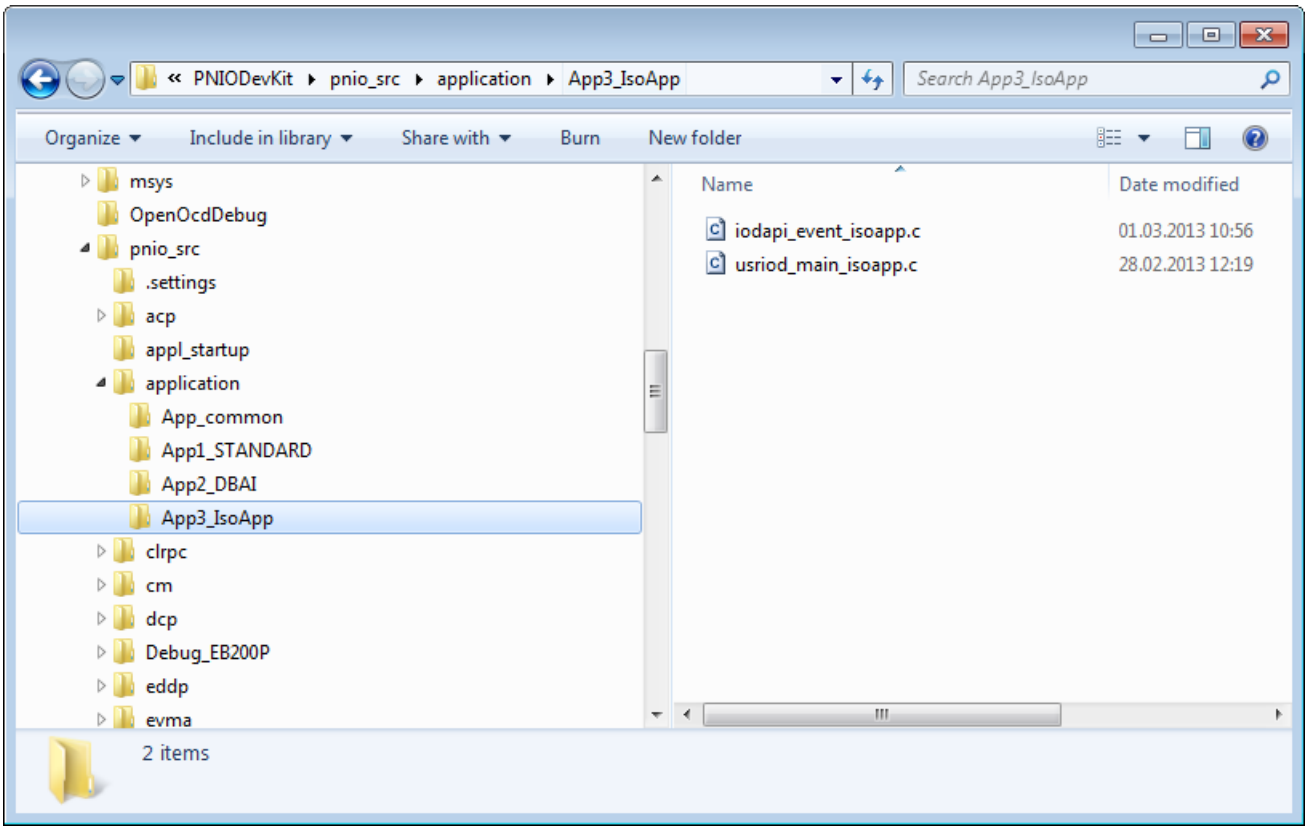


Figure 7-4 Files of the IsoApp example

This application example is included, if

```
# define EXAMPL_DEV_CONFIG_VERSION 3
```

is set in the "\Application\App_Common\usrapp_cfg.h" header file.

7.2 Functionality from the user perspective (use cases)

7.2.1 Use Case RT

The standard interface example "App1_Standard" and the example for direct buffer access "App2_DBAI" can be used unchanged for RT.

For the configuration, use the "xxx_RT.zip" project file that is compatible with your controller, see section Controller application examples (Page 60).

7.2.2 Use case IRT (without isochronous application)

The standard interface example "App1_Standard" can be used for RT as well as for IRT. In principle, the configured standard IO modules of this example application can also be used for both RT and IRT.

For the configuration, use the "CPU317-2_EB200P_IRT.zip" project file or a project file that is compatible with your controller, see section Controller application examples (Page 60).

If you also need an isochronous application for IRT that correctly handles the T_IO_Input and T_IO_Output values configured in the engineering, there is a special application example for this in the evaluation kit, see section Use case IRT with isochronous application. (Page 54).

Note

The display of console messages as well as a connected JTAG debugger can affect the sequence timing of the software. This may cause problems in IRT mode with short cycle times (e.g. 250 µs). In this case, deactivate the console messages or remove the JTAG debugger.

7.2.3 Use case IRT with isochronous application

You can synchronize the application to the bus clock with high precision when using an isochronous PROFINET application. In this case, the so-called T_IO_Input and T_IO_Output values are set by the application using a special startup record (IsochronousModeData, record index 0x8030), which is configured by the engineering system. It is then possible to sample input signals across all devices at a precise, predetermined point in time (T_IO_Input) and to activate output signals at a precise, predetermined point in time (T_IO_Output). The two parameters "T_IO_Input" and "T_IO_Output" are delay parameters that specify the delay at the start of the cycle (NewCycle).

The device example application "App3_IsoApp" is provided for this application. It uses special input-output modules from the GSD file, which are only permitted in IRT mode and which work isochronously.

For the implementation of T_IO_Input or T_IO_Output time constraints, it is possible to use various GPIOs (1 to 7) which trigger an output pulse with a programmable delay at the start of the cycle (Newcycle). In similar fashion, it is also possible to install interrupt routines that call a user-specified callback routine with a programmable delay at the start of the cycle.

You can find more information about the technological relationships in the PROFINET IO Specification /5/ and in the interface description /3/, section "Isochronous application with IRT, T_Input and T_Output evaluation".

For the configuration, use the "CPU317-2_EB200P_IRTIsoApp.zip" project file or a project file that is compatible with your controller, see section Controller application examples (Page 60).

Note

The display of console messages as well as a connected JTAG debugger can affect the sequence timing of the software. This may cause problems in IRT mode with short cycle times (e.g. 250 µs). In this case, deactivate the console messages or remove the JTAG debugger.

7.2.4 Use Case Tool Calling Interface (TCI)

Controller application and CPD tool

An executable application example for TCI can be found on the CD in the directory "(...)\tools\TCI". It includes a CPD example tool that establishes a connection with the PROFINET device EB 200P via the STEP 7 communication server and then writes or reads a 4 byte example record (record index 50) to/from the device. The contents of the 4 byte record can be changed on the CPD Tool screen.

The CPD tool was created with Microsoft Visual Studio 2008; a corresponding project "S7TCITest.dsp" as well as the source code is located in the directory "\tools\TCI\CpdTool\MSVC".

A guide for CPD tool is available on the CD in the directory "tools\TCI\CpdTool\doc".

STEP 7 must be installed with communication server functionality and started on the PC on which the CPD tool runs.

Device application

The TCI is included in the standard application example "App1_Standard", which means that it must be loaded and started on the device.

The TCI example works in the following way: When a write record request on record index 50 is received, an example record with a length of 4 bytes is stored and can be read back when a corresponding read record request is made on the same index.

A message is displayed on the console interface for each write/read of the record.

Note

STEP 7 V5.5 SP3 or later is required to communicate with the device via the communication server and the device tool.

See also

Controller application examples (Page 60)

7.2.5 Use Case PROFIenergy

Device application

The PROFIenergy functionality is included in the application examples standard interface ("App1_Standard") and DBA interface ("App2_DBAI"). There, the PROFIenergy services are mapped in accordance with specification for record read and write requests which are based on the record index 0x80a0. The sequence is generally the following:

1. A record write request is sent to the PROFINET device with index 0x80a0.
2. The response is polled by one or more record read requests with index 0x80a0 until the command is terminated with or without error feedback.

The "PROFIenergy_RequestHandler()" function is therefore selected for the record write request to index 0x80a0 in the application example (standard interface and DBA Interface). The data of the request is temporarily stored in a data structure, since it is required on the same index for the next record read.

The response to the request is configured in the "PROFIenergy_ResponseHandler()" which is executed for a record read request with index 0x80a0.

A message is displayed in the terminal program for the individual PROFIenergy services as an example. Since the example illustrates only the principles of PROFIenergy, "request completed" is returned as the first response. A process simulation with a timer is not included here to keep the example simple.

For the configuration, use the "xxx_RT.zip" project file that is compatible with your controller, see section Controller application examples (Page 60).

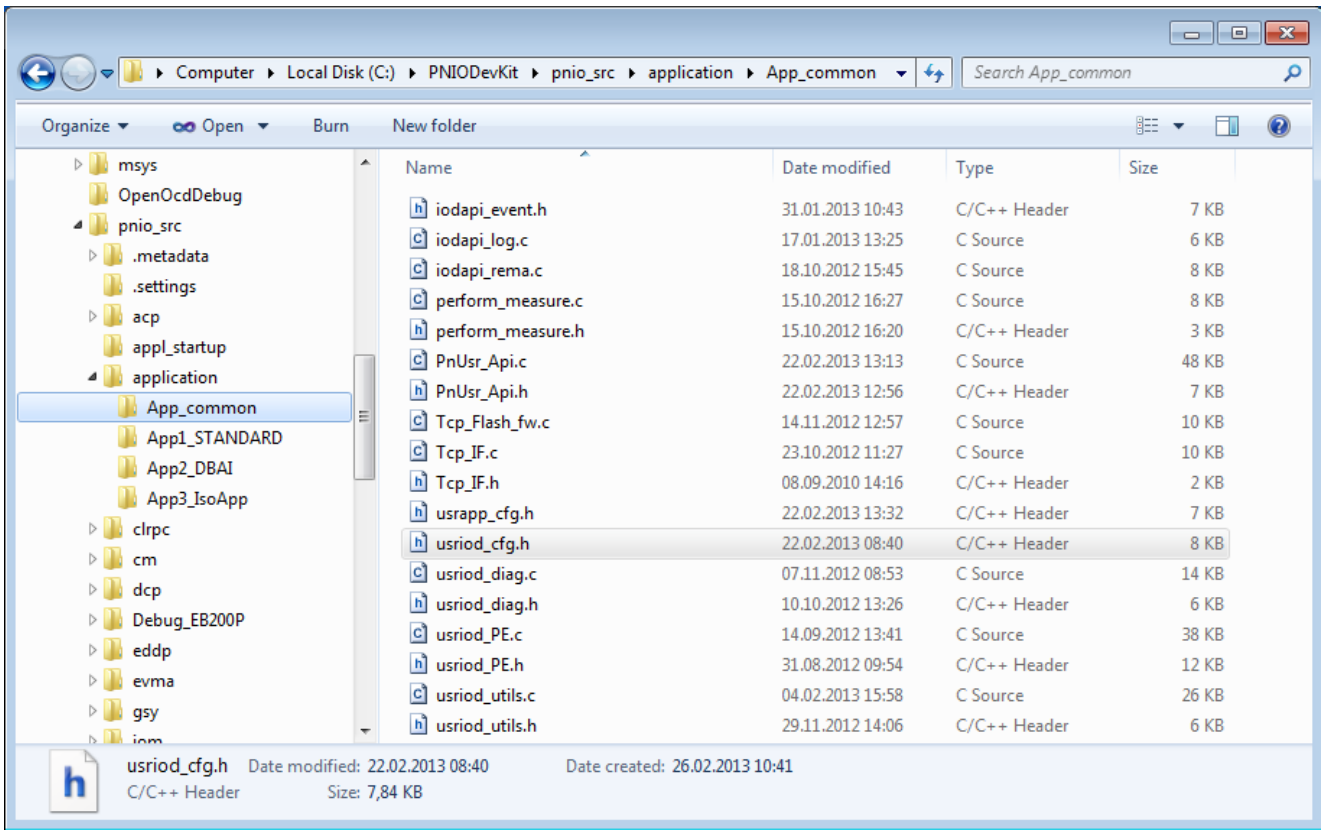


Figure 7-5 Application file for the PROFIenergy example

The actual PROFIenergy functionality is implemented in the module "(...)\application\usriod_pe.c". Data structures of the PROFIenergy services are implemented in the header file "(...)\application\usriod_pe.h".

Controller application

For the configuration, use the "CPU317-2_EB200P_RT.zip" project file or a project file that is compatible with your controller, see section Controller application examples (Page 60).

The function blocks FB815 (Start_Pause, End_Pause) and FB816 (open PE interface for PE_QUERY_MODES, PEM_STATUS, PE_IDENTIFY, etc.) are included in the STEP 7 user program for the SIMATIC S7-300. These are selected in OB1 and can be started from a memory bit:

FB815	START_PAUSE	memory bit 210.0
FB815	END_PAUSE	memory bit 210.1
FB816	OPEN INTERFACE	memory bit 220.0

7.2.6 Use case Media Redundancy (MRP)

Device application

The standard interface example "App1_Standard" must be used in this case. The two ports of the EB 200P can be configured in the configuration as ring ports.

MRP capability is set by the a transfer parameter in the function of the user interface "PNIO_device_open()". The activation of the MRP functionality takes place via the engineering in the configuration tool.

If MRP capability is the default setting, ports 1 and 2 are defined as ring ports so that the device can be configured as a node of the MRP ring in the engineering system.

In addition, an MRP-capable DAP (DAP3) must be plugged into slot 0 in the application. It includes the MRP client functionality, but not the MRP manager functionality.

The MRP capability is automatically set in the device application based on the example DAP (2 or 3) that is loaded. The DAP is stored in the Flash (REMA data) if REMA is activated so that the software automatically starts up with the correct DAP at the next startup (and all thereafter).

For the configuration, use the "CPU317-2_EB200P_MRP.zip" project file or a project file that is compatible with your controller, see section Controller application examples (Page 60).

A SIMATIC CPU 317-2 PN/DP with 2 Ethernet ports is used as the controller; it is connected to an EB 200P as a ring.

Note

A CPU firmware version of 3.2 or higher is required for MRP functionality.

Note

In order to safely avoid a possible ring closure during startup of a MRP device, an additional waiting period of 1.65 s is implemented in the PROFINET stack for the second ring port of the EvalKit before link-up takes place on this ring port.

Note

Media redundancy and Fast StartUp

Fast StartUp (FSU) is not possible if media redundancy is activated. In addition, if the MRP capable DAP3 is plugged at the same time FSU is to be activated, then the PROFINET IO controller must send a startup record at connect to switch off MRP. Otherwise FSU will not function. If the PROFINET IO controller cannot send this "MRP off" record, the MRP capability (see function "PNIO_device_open()") may not be activated on the device, which means DAP2 must be plugged for this example.
For this use case, DAP2 and DAP3 must have the same properties except for MRP capability (switched on in DAP3 and switched off in DAP2).

If the MRP capability is configured, but there is still no REMA data for the PDEV available on the device, then it will always start up with activated MRP as the default setting. This is necessary in order to avoid a logical ring (circulating frames) for unconfigured nodes during initial startup.

7.2.7 Use Case Shared Device

Shared Device operation is possible in principle in RT and IRT mode. But two IRT connections cannot be activated at the same time, which means one of the two must be configured as an RT connection in this case.

The standard interface example "App1_Standard" can be used unchanged for RT or IRT with a shared device. The DBAI example cannot be used, however, because it does not include the management of multiple ARs. This can be upgraded by the user accordingly if necessary.

For the configuration, use the "CPU317-2_EB200P_Shared.zip" project file or a project file that is compatible with your configuration, see section Controller application examples (Page 60).

Two SIMATIC CPUs 317-2 PN/DP with two Ethernet ports each are used as controllers.

Note

A CPU firmware version of 3.2 or higher is required for Shared Device functionality.

Note

If the sync master of a shared device (which is operated with IRT with "high performance") fails, it may cause the shared device to briefly fail when other IO controllers try to access it.

7.3 GSD files

A GSD file is stored on the CD in the directory "(...)\GSDML" for the EB 200P Evaluation Board.

The GSD file contains the following DAPs (Device Access Points):

DAP1 only for outdated PROFINET IO controllers without PDEV

DAP2 Standard DAP without MRP, without Shared Device

DAP3 Standard DAP + MRP + Shared Device

In addition, special IRT modules are configured in the GSD file for highly dynamic motion control applications in which all inputs and outputs of the devices of an IRT domain can be read or written simultaneously by assigning specific T-input and T-output times. These can be configured exclusively for IRT mode (RT Class3). The other modules can be operated for RT and IRT (Class1, Class3).

Note

We recommend upgrading the module ID of the affected DAP if enhancements are made to the functionality of a device. This ensures that a configuration that includes such functions cannot be inadmissibly loaded on to an old device.

7.4 Controller application examples

7.4.1 Controller application examples for SIMATIC S7-300 CPUs

The "\SIMATIC_STEP7" directory on the Evaluation Kit CD contains examples for the EB 200P for RT and IRT (RT Class3) mode.

The following table shows the device applications that can be used to operate the controller examples.

Table 7- 1 Controller examples with device application

File name	Content	Device example application
CPU317-2_EB200P_RT.zip	RT, CPU 317-2 PN/DP + EB 200P	App1_Standard, App2_DBAI
CPU317-2_EB200P_IRT.zip	IRT (high performance), CPU 317-2 PN/DP + EB 200P	App1_Standard, App2_DBAI
CPU317-2_EB200P_Shared	RT, 2 x CPU 317-2 PN/DP (firmware ≥ V3.2) + EB 200P	App1_Standard
CPU317-2_EB200P_MRP	RT, CPU 317-2 PN/DP + EB 200P in redundant mode	App1_Standard
CPU317-2_EB200P_IsoApp	IRT, isochronous device application	App3_IsoApp

Other examples may be added in future versions.

7.5 Loading Application Firmware in the Flash

A firmware update can be performed from the application since the firmware is loaded into RAM when Flash is booted and started there. A corresponding function is included on the device in all application examples for this purpose. For this purpose a PC program ("TcpFwLoader.exe") is used to initially load firmware onto the device via TCP from where it is programmed into by the application.

An executable firmware image, the PC program required for the download and compact operating instructions ("Readme.txt") are available on the CD in the subdirectory "(...)\Bootable Binary".

Creating Your Own Devices

8

Now that you are acquainted with the ERTEC 200P Evaluation Kit, this section briefly describes the steps necessary to generate and test your own PROFINET IO device application.

1. **Define the device-specific data**
   Depending on the function of your device, define the following data:
   - Required DAPs (Direct Access Point)
   - Required modules and submodules (with or without startup parameters)
   - User-specific diagnostics/alarms
   - User-specific acyclic data (read/write records)
   - I&M records
2. **Adapting the GSD file (see also /4/)**
   - Adapt general data:
     Vendor ID
     Vendor Name
     Device ID
     Main Family/Product Family, etc.
   - Adapt device-specific data (as defined in Step 1):
     - DAPs
     - Modules and submodules
     - Diagnostics/alarms
     - Acyclic records
3. **Check the modified GSD file**
   - Check the modified GSD file with the PROFINET XML viewer.

   Note

   A PROFINET XML viewer with integrated GSD checker is provided by the PROFIBUS User Organization and can be downloaded. It is free for members.

4. **Install the modified GSD file in the configuration tool**
   - Install your GSD file as described in the section Editing the project example for a PROFINET IO device (Page 36).
5. **Create and load your own configuration**
   - Adapt and compile the configuration in the configuration tool
   - Load the modified configuration into the PROFINET IO controller

6. **Adapt the device application**
   - Adapt general data:
     - Vendor ID
     - Vendor name
     - Device ID, etc.
   - Adapt device specific data:
     - Defined DAPs
     - Defined modules and submodules
     - User specific diagnostics/alarms
     - User specific records
     - I&M records
   - Adapt test command list
     - If necessary, adapt test commands to your defined test functions.
7. **Compile the device application**
   - Eclipse (see section Eclipse: Performing the PROFINET IO Application Build (Page 24))
8. **Load the device application onto the EB 200P and test it**
   - For flashing the firmware, see section Loading Application Firmware in the Flash (Page 60).
   - For debugging, see the document "Howto_use_Amontec_JTAGkey-Tiny_on_EB200P.pdf" in the installed subdirectory "(...)\PNIODevkit\OpenOcdDebug".

Appendix A

A.1 Abbreviations / Glossary of terms

ACP	**A**cyclic **C**ommunication **P**rotocol, refers to one of the basic software packages of the IO stack
API	**A**pplication **P**rocess **I**dentifier
AR	**A**pplication **R**elationship
BSP	**B**oard **S**upport **P**ackage
CLRPC	**C**onnection**l**ess **R**emote **P**rocedure **C**all, refers to one of the basic software packages of the IO stack
CM	**C**ontext **M**anagement, refers to one of the basic software packages of the IO stack
DAP	**D**evice **A**ccess **P**oint, specific entry in the GSD file
DBAI	**D**irect **B**uffer **A**ccess **I**nterface
DCP	**D**iscovery and basic **C**onfiguration **P**rotocol, refers to one of the basic software packages of the IO stack
DK_SW	**D**evelopment **K**it **S**oft**w**are (development kit for platforms based on standard Ethernet controllers)
EB 200P	**E**valuation **B**oard for ERTEC 200P (component of the Evaluation Kit ERTEC 200P)
EDDI	**E**thernet **D**evice **D**river for **I**RTE switch in the ERTEC 200/400 (earlier versions: EDD_ERTEC)
EDDP	**E**thernet **D**evice **D**river for **P**N switch in the ERTEC 200P
EDDS	**E**thernet **D**evice **D**river for **S**tandard Ethernet controller (earlier versions: EDD_soft)
EDD	**E**thernet **D**evice **D**river, general term for EDDI, EDDP, EDDS
ELOG	**E**rror **Log**ging for debugging purposes (level 1 = only errors are output)
GSD	**G**eneric **S**tation **D**escription
GSDML	**GSD M**arkup **L**anguage
GSY	**G**eneric **Sy**nc module, refers to one of the basic software packages of the IO stack
IOCR	**I**nput/**O**utput **C**ommunication **R**elationship
IOCS	**I**nput/**O**utput Object **C**onsumer **S**tatus
IOPS	**I**nput/**O**utputObject **P**rovider **S**tatus
IRT	**I**sochronous **R**eal**T**ime, Class 2 (IRT Class 2) or Class 3 (IRT Class 3)
LLDP	**L**ink **L**ayer **D**iscovery **P**rotocol (IEEE 802.1AB, allows stations to exchange chassis and port information)
LSA	**L**ayer **S**tructure **A**rchitecture
LW	Drive

MIB	**M**anagement **I**nformation **B**ase. Database for SNMP services
MRP	**M**edia **R**edundancy **P**rotocol
NARE	**N**ame **A**ddress **Re**solution
NRT	**N**on **R**eal**t**ime is a generic term for all non-real-time frames (not type 0x8892)
NV	**N**on-**v**olatile
OHA	**O**bject **Ha**ndler
OS	**O**perating **S**ystem, refers here to the abstraction layer for any operating system to which the IO stack is to be ported.
PDEV	**P**hysical **Dev**ice
PNIO	**P**ROFI**N**ET **IO**
PNO	**P**ROFIBUS **N**utzer **O**rganisation (user organization)
PCF	**P**olymeric **C**ladded **F**iber (optical communication medium)
POF	**P**olymeric **O**ptical **F**iber (optical communication medium)
RT	**R**eal **T**ime is a generic term for acyclic and cyclic real-time
RT Class	**R**eal**T**ime class according to the PROFINET IO specification
SI	**S**tandard **I**nterface
SNMP	**S**imple **N**etwork **M**anagement **P**rotocol
SOCK	UDP Socket Interface for PROFINET IO, refers to one of the basic software packages of the IO stack
TAP	**T**est **A**ccess **P**ort
UDP	**U**ser **D**atagram **P**rotocol
UUID	**U**niversal **U**nique **Id**entifier

A.2 References

/1/
GSDML Specification for PROFINET IO Version 2.3 or later

PROFIBUS User Organization e.V.

/2/
EB 200P Manual

(Manual_EB200P_V1.0.pdf)

/3/
Interface Description Evaluation Kit ERTEC 200P PN IO V4.0

(Interface_Description_EvalKit_ERTEC200P_V4.0.pdf)

/4/
GSDML Getting Started Made Easy V1.2

(GSDML-GettingStarted.pdf)

/5/
PROFINET IO Application Layer Service Definition and
PROFINET IO Application Layer Protocol Specification

(Can be downloaded from the PNO website (http://www.profinet.com))