

A man in a light blue shirt is shown from the side, holding a tablet computer. He is looking at the screen, which displays a software interface with various charts and data. The background is a blurred industrial factory setting with large machinery and equipment.

SIEMENS

Application Example • 09/2015

User Guide

PROFINET Driver for Controller

PN Driver V1.0

<https://support.industry.siemens.com/cs/ww/en/view/109478514>

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.

Any form of duplication or distribution of these Application Examples or excerpts hereof is prohibited without the expressed consent of Siemens AG.

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks. They are important components in a holistic industrial security concept. With this in mind, Siemens' products and solutions undergo continuous development. Siemens recommends strongly that you regularly check for product updates.

For the secure operation of Siemens products and solutions, it is necessary to take suitable preventive action (e.g. cell protection concept) and integrate each component into a holistic, state-of-the-art industrial security concept. Third-party products that may be in use should also be considered. For more information about industrial security, visit <http://www.siemens.com/industrialsecurity>.

To stay informed about product updates as they occur, sign up for a product-specific newsletter. For more information, visit <http://support.automation.siemens.com>.

Table of Contents

Warranty and Liability	2
Preface	5
1 Task	6
2 Solution	7
2.1 Overview	7
2.2 Hardware and software components	7
2.2.1 Validity	7
2.2.2 Components used	7
2.3 Restrictions	8
3 Basics	9
3.1 Introduction to the PN Driver	9
3.1.1 What is the PN Driver?	9
3.1.2 PN Driver and user guide – complete overview	9
3.1.3 PN Driver and user guide – components	10
3.1.4 Executable compiled PN controller and generated hardware configuration	10
3.1.5 PN Driver functions in the application example	11
4 Hardware Configuration in TIA Portal	13
4.1 Hardware configuration in TIA Portal V12	13
4.2 Interrupt configuration	17
5 Principle of Operation and Structure of a Control Program	19
5.1 Introduction to the PN controller	19
5.2 Interaction options with PN devices	19
5.3 Starting, stopping and shutting down the PN controller	20
5.4 Cyclic program	21
5.5 Hardware interrupts	23
6 Installation and Startup	24
6.1 Installing the hardware	24
6.2 Startup	25
7 Operation of the Application	26
7.1 Overview of the user interface	26
7.2 Main menu	26
7.2.1 ...ReadRange	26
7.2.2 ...EngineKeypad	27
7.2.3 ...ReadWrite	28
7.2.4 ...SetSleeptimer	29
8 Appendix: Implementing your own PN Controller	30
8.1 Steps from the DVD to the executable program	30
8.2 Programming a cyclic program	30
8.3 Tips and tricks	32
8.3.1 The PN device is not accessible	32
8.3.2 The PN device permanently indicates a fault or interrupt	32
8.3.3 The PN device alternately goes online and offline	32
8.3.4 Will the PN controller work on a diagnostic/mirror port of a switch?	33
8.3.5 My network adapter is not detected when I restart the software	33
8.4 Characteristics of the PN Driver	33

Table of Contents

9	Appendix: Introduction to PROFINET.....	34
9.1	Overview	34
9.2	Device types.....	34
9.3	Basic functions	35
10	References.....	36
11	History	36

Preface

About the PN Driver

For simple automation tasks with low communication overheads, it is not always economically efficient to purchase special controller software.

For these cases, Siemens offers the PROFINET Driver for Controller (PN Driver).

It allows you to develop PROFINET applications for standard PCs without special hardware. Communication takes place via a standard PC Ethernet interface.

Using the PN Driver, a created application can even be ported to different operating systems.

With an open XML interface and a cycle time of up to 1 ms, the PN Driver increases flexibility and provides the desired performance at low cost.

About the user guide

To enable you to quickly and conveniently create your own application using the PN Driver, this user guide offers a clearly structured starting point.

The user guide will guide you through the configuration and programming of an application example that solves an automation task with standard use cases.

Within a use case, the console application example deals with the following topics: motor control, HMI, interrupts, distributed I/O and analog and digital inputs and outputs.

In addition, the user guide summarizes expert knowledge regarding all the concepts necessary for the development using the PN Driver.

Structure of this document

Users and **decision-makers** will find information on the functionality of the application example in chapters 3 (basics of the PN Driver), 6 (startup of the example) and 7 (operation of the example).

Even if you do not start up the example, these chapters provide a good overview of the capabilities of the PN Driver.

Developers should go through the entire document. Especially chapters 8 and 9 in the appendix give you an overview of the necessary information and procedure for developing your own PN controller.

1 Task

Introduction

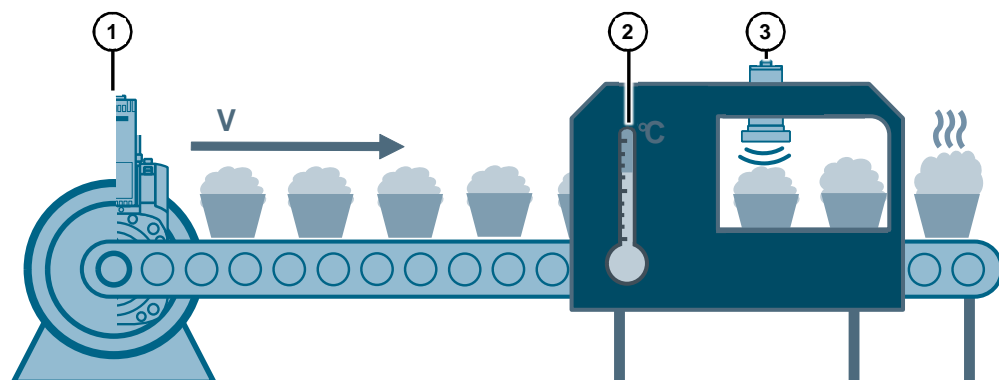
An illustrative use case is used to demonstrate the functionality of the PN Driver. It combines the theory of the sample application with a specific practical application and therefore helps the user better understand the driver.

Use case description

The aim is to control and monitor a continuously running oven.

- A drive moves a cake pan into the oven.
- When the cake pan is in the oven, the oven heats up.
- A Sonar-BERO checks whether a test cake has already fully risen.

Overv



The figure below describes the numbered components.

Table 1-1

No.	Description
1.	Frequency inverter with motor
2.	Passive temperature sensor
3.	Sonar-BERO sensor

A motor-driven conveyor belt transports cakes into an oven.

When a baker sees that a cake is done, he has the next cake transported into the oven. For this purpose, the baker checks the current cake's expansion and temperature visually, using the temperature sensor and the Sonar-BERO.

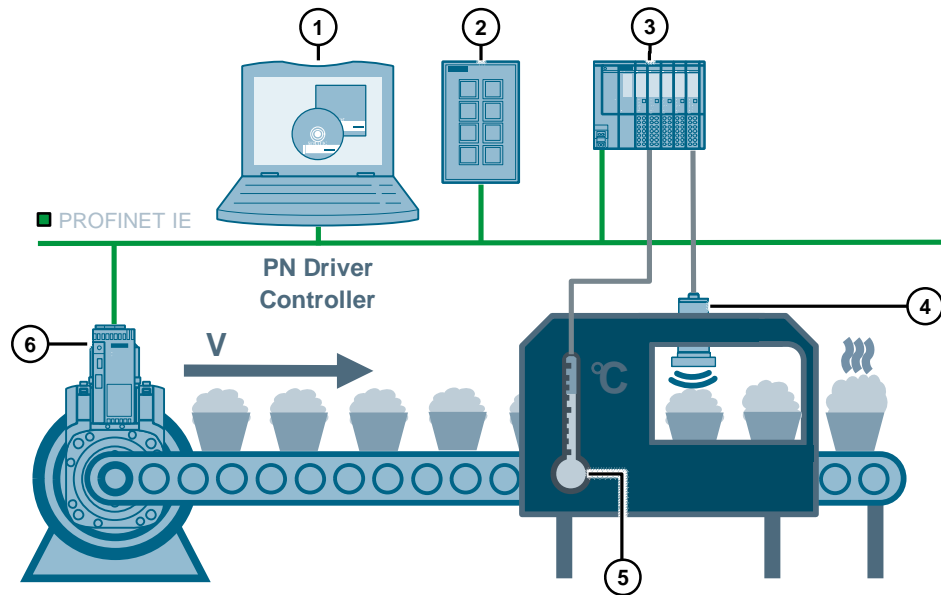
2 Solution

2.1 Overview

Diagrammatic representation

The diagrammatic representation below shows the most important components of the solution:

Figure 2-1



© Siemens AG 2015 All rights reserved

2.2 Hardware and software components

2.2.1 Validity

This application is valid for:

- STEP 7 V12 or higher (TIA Portal V12)
- Visual Studio 2012
- Windows 7
- WinPcap: Developer Resources V4.1.3

2.2.2 Components used

This application was created with the following components:

Hardware components

Table 2-1

No.	Component	Article number	Explanation
1.	PC station with PN Driver		Includes the user program with the PN controller as a console application.
2.	Key Panel (HMI KP8F)	1P 6AV3-3AF37-0AX0	Allows the user to control the conveyor belt of the oven.

2 Solution

2.3 Restrictions

No.	Component	Article number	Explanation
3.	Distributed I/O (ET200SP)	6ES7155-6AU00-0CN0	Used to control and read out the two sensors (2 & 3).
4.	Sonar sensor (3RG6-14)	3RG6014-3AH00	Used to digitally determine the cake's degree of expansion.
5.	Temperature sensor (PT100)		Used for analog measurement of the cake's temperature.
6.	Frequency inverter (G120C)	6SL310-1KE18-8AF1 A02	The motor is controlled via the control words of the drive.

Software components

Table 2-2

Component	Article number	Note
TIA Portal V12	6ES7822-1A.02-...	A free trial version is available for download.
Visual Studio 2012		A free version is available for download.
PN Driver for Controller	6ES7195-3AA00-0YA0	
WinPcap: Developer Resources		The free pack is available for download.

Sample files and projects

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

Table 2-3

Component	Note
109478514_PN_Driver_Guide_CODE_v10.zip	This zip file contains the STEP 7 project.
109478514_PN_Driver_Guide_DOC_v10_e.pdf	This document.

2.3 Restrictions

Version 1.0 of the PN Driver dealt with in this document neither supports nor documents the following announced functionalities:

- Use of the PN Driver in Linux
- Porting Windows software to Linux or vice versa
- Creating the hardware configuration without TIA Portal V12
- Isochronous real-time communication via PROFINET

3 Basics

3.1 Introduction to the PN Driver

3.1.1 What is the PN Driver?

The PN Driver is a C/C++ library for programming a PN controller on a standard PC.

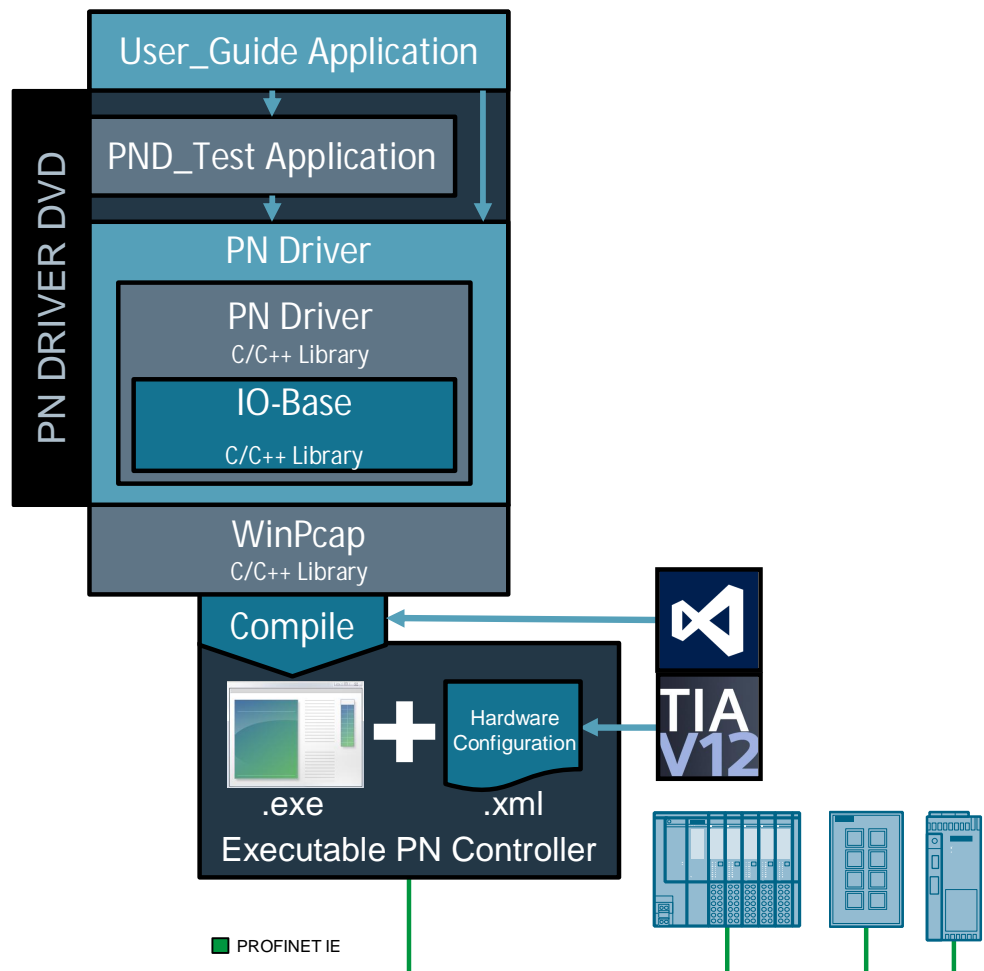
3.1.2 PN Driver and user guide – complete overview

The figure below shows all the components of the PN Driver and the user guide.

Note

Both the user guide and the PN Driver are supplied with documentation. This is not shown in the following figure.

Figure 3-1



The individual components of the figure will be explained in the next section.

3.1.3 PN Driver and user guide – components

The list below follows the order of the items in the above figure.

Table 3-1

Component	Description
User_Guide application	This console application is a realistic application example of the PN Driver. Parts of it are based on the functionality of the PND test application. This example is explicitly the bakery application described above.
PND_Test application	The purpose of this console application is to provide the user with an overview of all the PN Driver functions available. Compared to the user guide application, it does not contain a self-contained automation example but only individual functionalities. (Included in the scope of delivery)
PN Driver	The PN Driver product is based on the following libraries: PN Driver, IO-Base and WinPcap. However, WinPcap is not included in the scope of delivery.
PN Driver C/C++ library	An extension to the IO-Base library for programming a PN controller. This is not a stand-alone library; it is part of the IO-Base library.
IO-Base C/C++ library	This library supports various PROFINET functionalities.
WinPcap C/C++ library	Library for network packet processing. This library provides the PN Driver with access to the network. This library is used by the PN Driver library.

Note

The application example supplied with the PN Driver (pnd_Test) differs from the one of the user guide (user_guide).

3.1.4 Executable compiled PN controller and generated hardware configuration

The following list describes the created results of the above figure.

Table 3-2

XML hardware configuration	The individual PN devices are configured and compiled in TIA Portal V12. The result is an XML hardware configuration. Among other things, it includes information on: <ul style="list-style-type: none"> • IP addresses • Hardware interrupts • Memory addresses per module • Network topology • Module names • Module types
Executable file (.exe)	This file is the result of the compiled C/C++ automation solution in Visual Studio. To be executed, the file requires the XML HW Config.

3.1.5 PN Driver functions in the application example

The following PN Driver functions are used in the application example of this user guide. In addition, we will show you the call path of the function in Visual Studio, starting with the “user_guide.c” file that represents the entire application example. Some of the functions were not used for the implemented standard use case.

Note This table is followed by another table that establishes the link between the functions and the application example.

Paths that include “pnd_test” refer to the PND_Test application supplied with the PN Driver. This test application is not the bakery control system described in this document. However, parts of the bakery control system are based on the PND_Test application.

Table 3-3

PN Driver functions	User guide	Call path
Management functions		
PNIO_controller_open()	X	startPnController/pnd_test_controller_open
PNIO_register_cbf()	X	startPnController/pnd_test_register_devact_cbf
PNIO_controller_close()	X	shutDownPnController/pnd_test_controller_close
Mode functions		
PNIO_set_mode()	X	*/pnd_test_set_mode
PNIO_device_activate()		
Read/write functions		
PNIO_data_read()	X	read_data
PNIO_data_write()	X	write_data
Interface functions		
PNIO_rec_read_req()		
PNIO_rec_write_req()		
Callback events (registered with PNIO_register_cbf)		
PNIO_CBE_REC_READ_CONF		
PNIO_CBE_DEV_ACT_CONF	X	startPnController/pnd_test_register_devact_cbf
PNIO_CBE_ALARM_IND		
PNIO_CBE_MODE_IND	X	startPnController

Note For more information on the functions, please refer to the “PGH_IO-Base_76.pdf” supplied with the PN Driver. Furthermore, they will be described in greater detail in a later chapter.

3.1 Introduction to the PN Driver

The following table establishes the link between the functions explained above and the application example.

Table 3-4

PN Driver functions	Meaning in the application example
PNIO_controller_open()	(For initialization) Registers the PN controller and its interrupts in the IO-Base interface. Note: The "SERV_CP_startup" method must be called first. It configures the PN controller with the information from the hardware configuration.
PNIO_register_cbf()	(For initialization) Registers a callback function for mode changes (Start/Stop, etc.)
PNIO_controller_close()	(For finalization) Unregisters the PN controller in the IO-Base interface.
PNIO_set_mode()	(For initialization) At the beginning, this function sets the current mode of the PN controller to "Operate"; at the end, it sets it to "Offline"
PNIO_data_read()	(For ongoing operation) Reads a byte out of the process image, either out of an input or output address.
PNIO_data_write()	(For ongoing operation) Writes a byte to the process image, either to an input or output address.
PNIO_CBE_DEV_ACT_CONF	(For initialization) Callback event type that triggers the call of any function transferred to PNIO_register_cbf(). This event is triggered when a connect error of the PN controller occurs.
PNIO_CBE_MODE_IND	(For initialization) Callback event type that triggers the call of any function transferred to PNIO_register_cbf(). This event is triggered when the PN controller mode changes, for example, from "Operate" to "Offline".

Note

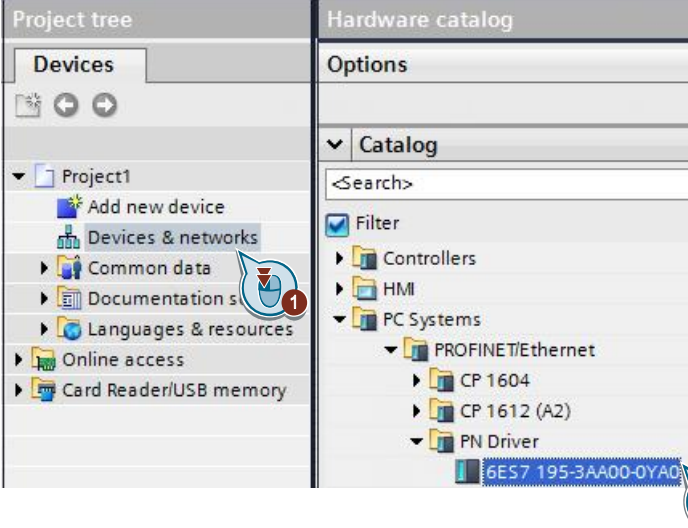
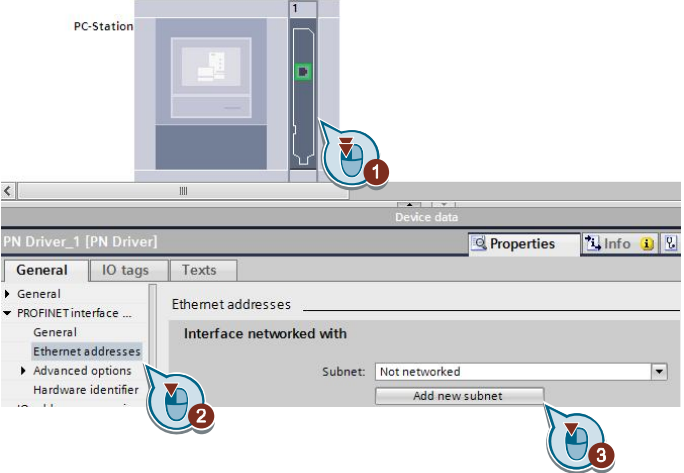
Chapter 5 provides more information on how these functions are used in the application example.

4 Hardware Configuration in TIA Portal

4.1 Hardware configuration in TIA Portal V12

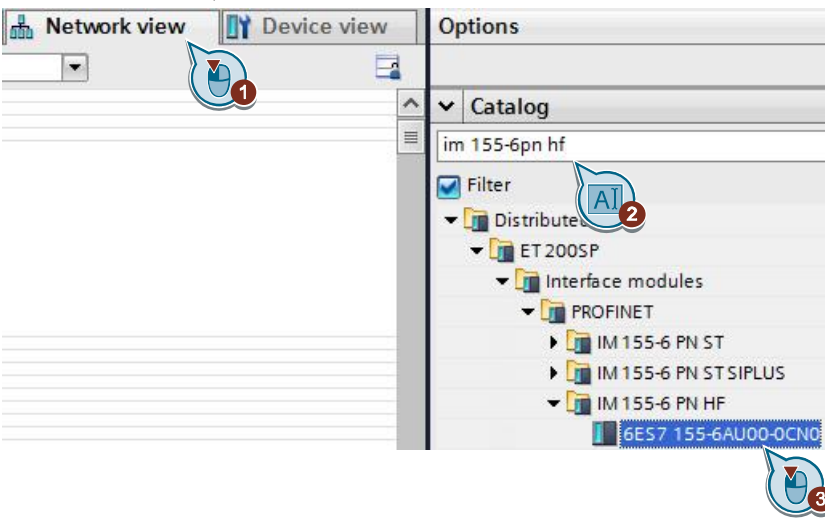
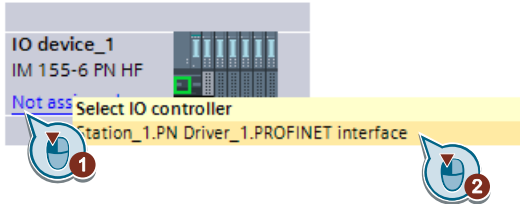
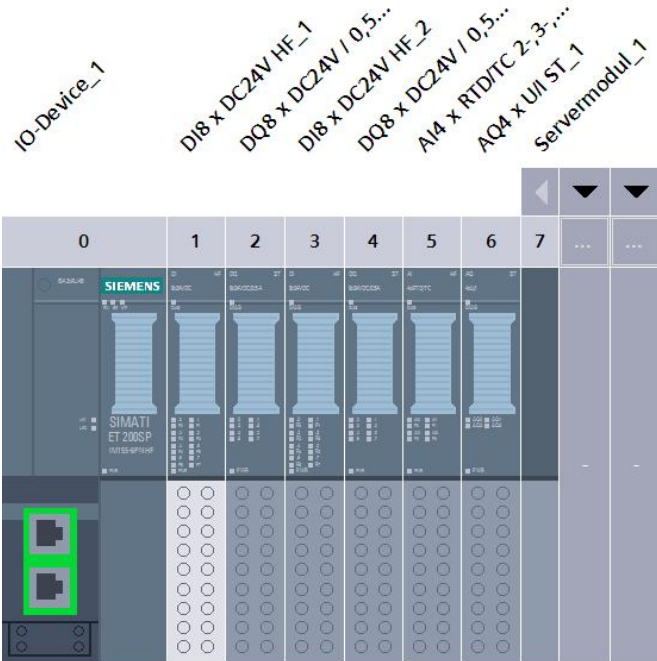
Note For the download links, please refer to the “References” chapter.

Table 4-1

No.	Description
1.	Download TIA Portal V12 and install it.
2.	Download the HSP for TIA Portal V12 and install them.
3.	<p>Add the PN Driver from the hardware catalog.</p> 
4.	<p>Add the subnet.</p>  <p>The PN driver is automatically assigned:</p> <ul style="list-style-type: none"> • PN device name: “pn driver_1” • IP address 192.168.0.1


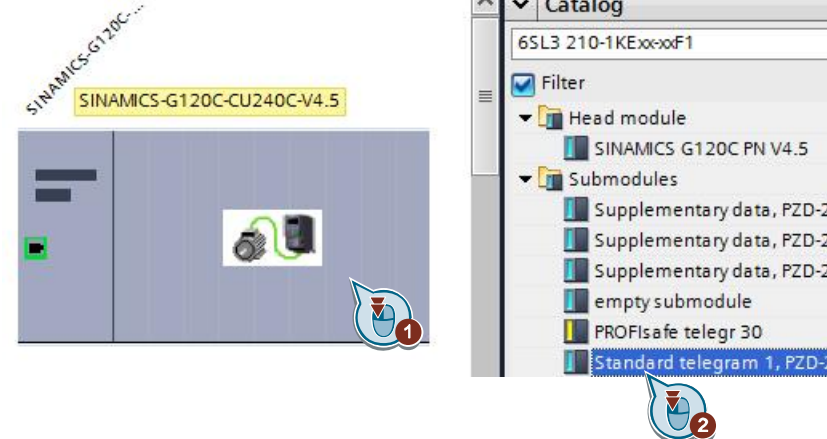
4 Hardware Configuration in TIA Portal

4.1 Hardware configuration in TIA Portal V12

No.	Description
5.	<p>In the Network view, add a distributed I/O module.</p> 
6.	<p>Add the new device to the PN Driver.</p>  <p>The second device is automatically assigned:</p> <ul style="list-style-type: none"> • PN device name: "io device_1" • IP address: 192.168.0.2
7.	<p>Insert the shown modules from the hardware catalog into the distributed I/O module.</p> 

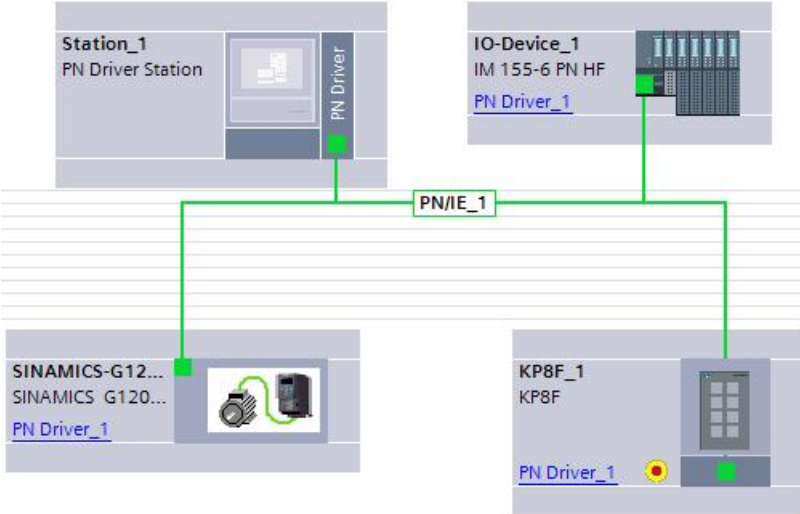
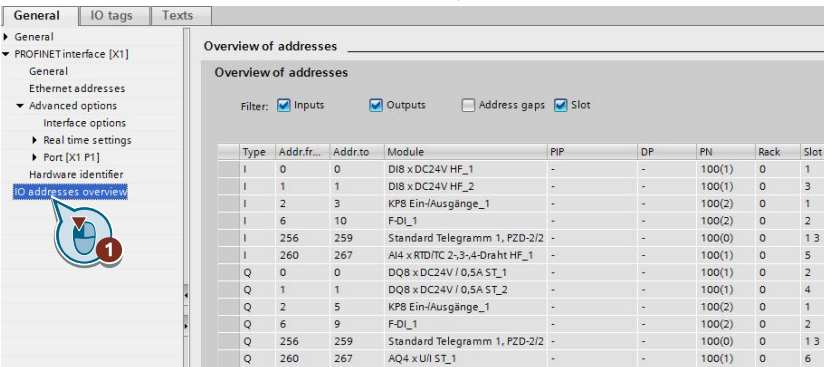
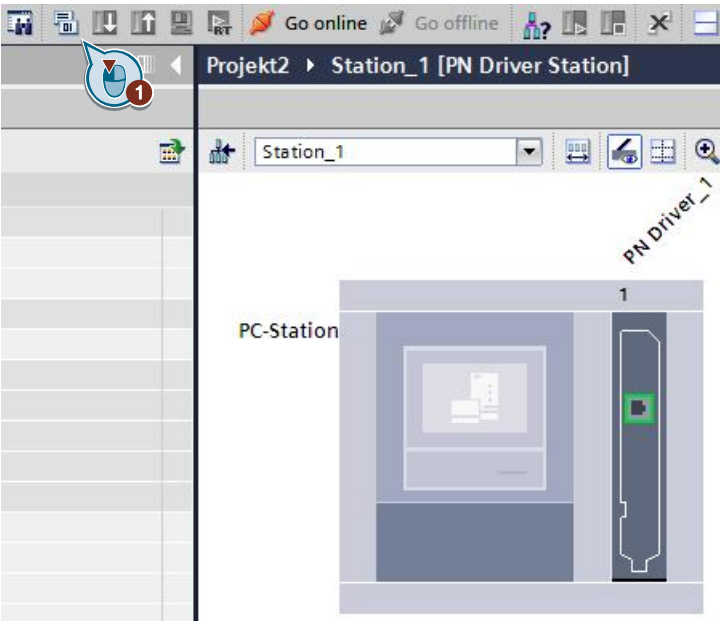
4 Hardware Configuration in TIA Portal

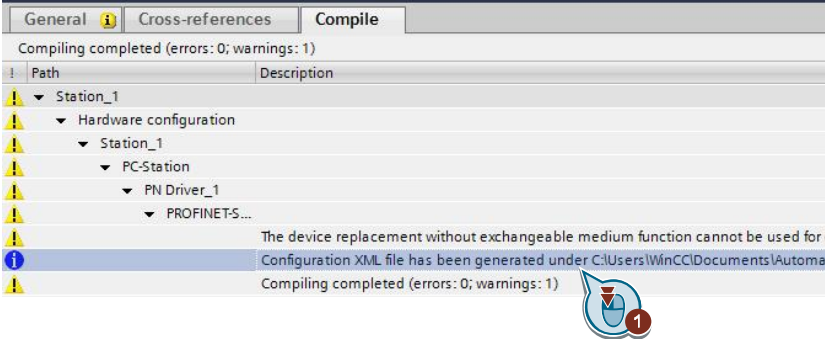
4.1 Hardware configuration in TIA Portal V12

No.	Description
8.	Download the GSDML files for the G120C frequency inverter: https://support.industry.siemens.com/cs/en/en/view/60602080 Extract the .zip file.
9.	Install the GSDML files from the .zip file in TIA Portal: 
10.	One after the other, insert the following modules in the Network view: <ul style="list-style-type: none"> • 6AV3 688-3AF37-0XA0 (HMI/SIMATIC Key Panel/KP8F PN) • 6SL3 210-1KExx-xxF1 (other field devices/Additional Ethernet devices/PROFINET IO/Drives/Siemens AG/SINAMICS/Head module) The third device is automatically assigned: <ul style="list-style-type: none"> • PN device name: "io device_2" IP address 192.168.0.3 The fourth device is automatically assigned: <ul style="list-style-type: none"> • PN device name: "io device_3" IP address 192.168.0.4
11.	In the Device view, click the G120C frequency inverter. In the filtered view of the hardware catalog, double-click Standard telegram 1 to add it to the frequency inverter: 

4 Hardware Configuration in TIA Portal

4.1 Hardware configuration in TIA Portal V12

No.	Description																																																																																																																					
12.	<p>In the Network view, the devices should now look as follows:</p> 																																																																																																																					
13.	<p>Now you can display all address ranges of the project, sorted by IO devices, in the PN Driver.</p> <p>To do this, double-click the PN Driver in any view.</p>  <table border="1" data-bbox="667 1093 1297 1323"> <thead> <tr> <th>Type</th> <th>Addr.fr...</th> <th>Addr.to</th> <th>Module</th> <th>PIP</th> <th>DP</th> <th>PN</th> <th>Rack</th> <th>Slot</th> </tr> </thead> <tbody> <tr><td>I</td><td>0</td><td>0</td><td>DIB x DC24V HF_1</td><td>-</td><td>-</td><td>100(1)</td><td>0</td><td>1</td></tr> <tr><td>I</td><td>1</td><td>1</td><td>DIB x DC24V HF_2</td><td>-</td><td>-</td><td>100(1)</td><td>0</td><td>3</td></tr> <tr><td>I</td><td>2</td><td>3</td><td>KP8 Ein-/Ausgänge_1</td><td>-</td><td>-</td><td>100(2)</td><td>0</td><td>1</td></tr> <tr><td>I</td><td>6</td><td>10</td><td>F-DI_1</td><td>-</td><td>-</td><td>100(2)</td><td>0</td><td>2</td></tr> <tr><td>I</td><td>256</td><td>259</td><td>Standard Telegramm 1, PZD-2/2</td><td>-</td><td>-</td><td>100(0)</td><td>0</td><td>1 3</td></tr> <tr><td>I</td><td>260</td><td>267</td><td>A4 x RTD/TC 2-,3-,4-Draht HF_1</td><td>-</td><td>-</td><td>100(1)</td><td>0</td><td>5</td></tr> <tr><td>Q</td><td>0</td><td>0</td><td>DQ8 x DC24V / 0,5A ST_1</td><td>-</td><td>-</td><td>100(1)</td><td>0</td><td>2</td></tr> <tr><td>Q</td><td>1</td><td>1</td><td>DQ8 x DC24V / 0,5A ST_2</td><td>-</td><td>-</td><td>100(1)</td><td>0</td><td>4</td></tr> <tr><td>Q</td><td>2</td><td>5</td><td>KP8 Ein-/Ausgänge_1</td><td>-</td><td>-</td><td>100(2)</td><td>0</td><td>1</td></tr> <tr><td>Q</td><td>6</td><td>9</td><td>F-DI_1</td><td>-</td><td>-</td><td>100(2)</td><td>0</td><td>2</td></tr> <tr><td>Q</td><td>256</td><td>259</td><td>Standard Telegramm 1, PZD-2/2</td><td>-</td><td>-</td><td>100(0)</td><td>0</td><td>1 3</td></tr> <tr><td>Q</td><td>260</td><td>267</td><td>AQ4 x UI ST_1</td><td>-</td><td>-</td><td>100(1)</td><td>0</td><td>6</td></tr> </tbody> </table>	Type	Addr.fr...	Addr.to	Module	PIP	DP	PN	Rack	Slot	I	0	0	DIB x DC24V HF_1	-	-	100(1)	0	1	I	1	1	DIB x DC24V HF_2	-	-	100(1)	0	3	I	2	3	KP8 Ein-/Ausgänge_1	-	-	100(2)	0	1	I	6	10	F-DI_1	-	-	100(2)	0	2	I	256	259	Standard Telegramm 1, PZD-2/2	-	-	100(0)	0	1 3	I	260	267	A4 x RTD/TC 2-,3-,4-Draht HF_1	-	-	100(1)	0	5	Q	0	0	DQ8 x DC24V / 0,5A ST_1	-	-	100(1)	0	2	Q	1	1	DQ8 x DC24V / 0,5A ST_2	-	-	100(1)	0	4	Q	2	5	KP8 Ein-/Ausgänge_1	-	-	100(2)	0	1	Q	6	9	F-DI_1	-	-	100(2)	0	2	Q	256	259	Standard Telegramm 1, PZD-2/2	-	-	100(0)	0	1 3	Q	260	267	AQ4 x UI ST_1	-	-	100(1)	0	6
Type	Addr.fr...	Addr.to	Module	PIP	DP	PN	Rack	Slot																																																																																																														
I	0	0	DIB x DC24V HF_1	-	-	100(1)	0	1																																																																																																														
I	1	1	DIB x DC24V HF_2	-	-	100(1)	0	3																																																																																																														
I	2	3	KP8 Ein-/Ausgänge_1	-	-	100(2)	0	1																																																																																																														
I	6	10	F-DI_1	-	-	100(2)	0	2																																																																																																														
I	256	259	Standard Telegramm 1, PZD-2/2	-	-	100(0)	0	1 3																																																																																																														
I	260	267	A4 x RTD/TC 2-,3-,4-Draht HF_1	-	-	100(1)	0	5																																																																																																														
Q	0	0	DQ8 x DC24V / 0,5A ST_1	-	-	100(1)	0	2																																																																																																														
Q	1	1	DQ8 x DC24V / 0,5A ST_2	-	-	100(1)	0	4																																																																																																														
Q	2	5	KP8 Ein-/Ausgänge_1	-	-	100(2)	0	1																																																																																																														
Q	6	9	F-DI_1	-	-	100(2)	0	2																																																																																																														
Q	256	259	Standard Telegramm 1, PZD-2/2	-	-	100(0)	0	1 3																																																																																																														
Q	260	267	AQ4 x UI ST_1	-	-	100(1)	0	6																																																																																																														
14.	<p>Click the PN Driver and compile the entire project.</p> 																																																																																																																					

No.	Description
15.	<p>When the project has been compiled, the hardware configuration is saved in an XML file.</p>  <p>Double-clicking the above line opens the folder with the XML file. This file must later be located in the same folder as the compiled C/C++ PN Driver automation program (.exe).</p>
16.	<p>You can make changes to your hardware configuration at any time and recompile the XML file. Then the PN controller adds the address ranges of the added devices to its process image. Provided that the old devices do not change, adding more devices does not disrupt an already functioning automation program.</p>

4.2 Interrupt configuration

In the application example, all standard interrupts (inserting, removing modules, etc.) are automatically received by the PN Driver and displayed in the console when they are received.

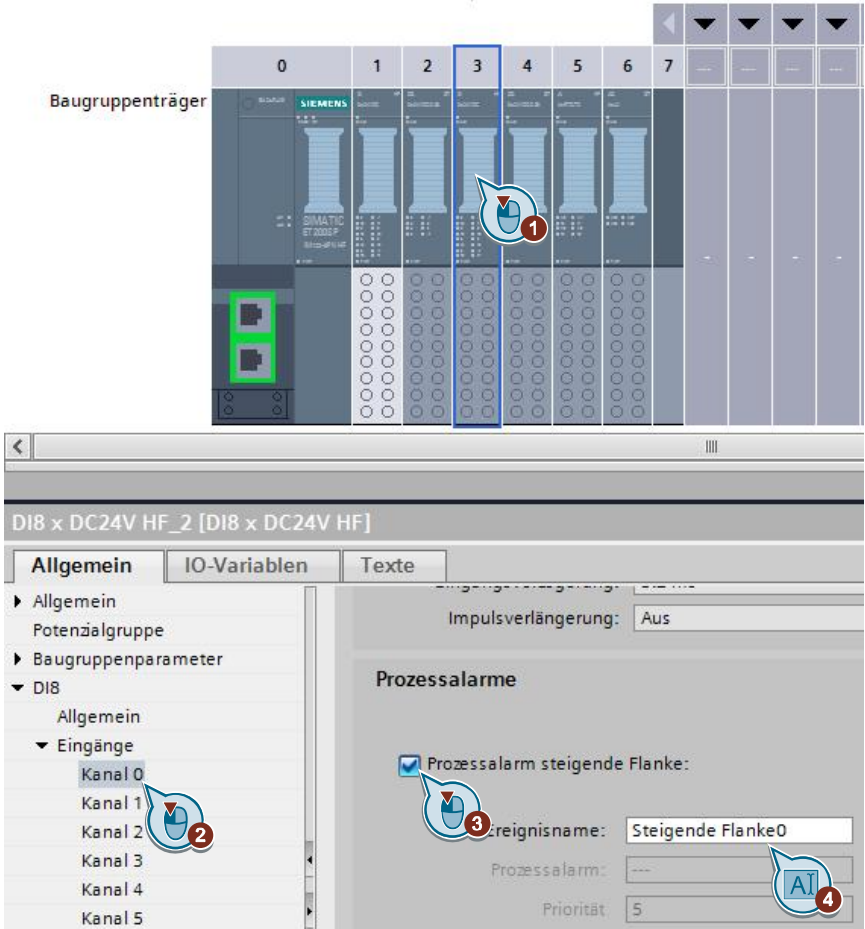
If you want to configure new interrupts (e.g., when there is a positive edge at an addressable bit or a configured interrupt of an analog module), they can be configured in TIA Portal in the usual manner. Using the generated XML hardware configuration, they are automatically processed by the PN controller.

The following table shows how an interrupt can be configured in TIA Portal at a digital input of a distributed I/O module.

4 Hardware Configuration in TIA Portal

4.2 Interrupt configuration

Table 4-2

No.	Step
1.	Navigate to the hardware configuration of the I/O module.
2.	<p data-bbox="496 383 1364 439">In the properties of the DI module, select an input channel and set a hardware interrupt at rising edge for this channel.</p>  <p>The screenshot shows a rack of modules labeled 'Baugruppenträger' with slots 0 through 7. Slot 3 is highlighted in blue. Below the rack, the configuration for 'DI8 x DC24V HF_2 [DI8 x DC24V HF]' is shown. The 'Allgemein' tab is active, and the 'Eingänge' list has 'Kanal 0' selected. In the 'Prozessalarme' section, the checkbox 'Prozessalarm steigende Flanke' is checked, and the event name is 'Steigende Flanke0'. The priority is set to 5. A blue 'AI' button is visible in the bottom right corner.</p>
3.	Now save and compile the hardware configuration.
4.	Copy the generated XML hardware configuration file to the folder where you have saved the executable file of the PN controller.
5.	<p>Now the interrupt will be executed when the interrupt's memory address has been read once.</p> <p>The address can be found in the module properties, HW ID.</p>

5 Principle of Operation and Structure of a Control Program

5.1 Introduction to the PN controller

The PN Driver library allows you to write your own PN controller. It manages a process image (= large memory area that is cyclically updated) with multiple input and output addresses. Some of these addresses are assigned to modules, control words (for frequency inverters, etc.), sensors and actuators.

Input addresses are usually connected to sensors and output addresses are connected to actuators.

Input address 112 corresponds to the 112th byte in the process image.

The status of the first bit of this address (112.0) could signify, for example, the pressing of a button.

If the button is pressed, address 112.0 is assigned the value 1; if it is not pressed, the switch is assigned the value 0.

The PN controller is mainly busy reading values from the process image and writing other values to the process image.

Via the network, all connected and configured PN devices cyclically receive the address values of the process image assigned to them.

5.2 Interaction options with PN devices

The PN controller programmed by you has the following options to access PN devices:

1. Read address values from process image
2. Write address values to process image
3. Respond to (hardware) interrupt of PN device
4. Respond to status changes of PN device (e.g., "has error")

Note

It is explicitly not possible to process diagnostic data (outside of alarms and interrupts) of PN devices. However, it can be viewed in the "Online & diagnostics" function of TIA Portal in "Online access /<adapter>/<PN device>/Online & diagnostics".

5.3 Starting, stopping and shutting down the PN controller

The usual life cycle of a PN controller is enabled using the following methods of the PN Driver library:

Table 5-1

No.	Method	Description
1.	PNIO_controller_open()	Registers and opens the PN controller. This sets the controller to "Stop" mode and transfers parameters for some callback functions that are monitored from this time on.
2.	PNIO_register_cbf()	Directly after PNIO_controller_open, it is recommended to register the callback functions. In the functional example, the "PNIO_CBE_DEV_ACT_CONF" callback function is registered. It has the effect that the PN controller is kept informed of the accessibility of the PN devices.
3.	Load hardware configuration	In the application example, the XML file is loaded using the "loadHardwareConfig" method. It is recommended to use this method as the structure of the XML file is very complex.
4.	PNIO_set_mode()	The "PNIO_set_mode()" method allows you to change the module's mode (Start, Stop, Reset). When switching on, the PN controller should also be set to "Start" mode. This is done with the following command: "PNIO_set_mode(PNIO_MODE_OPERATE)"
5.	PNIO_controller_close()	Before this call, "PNIO_set_mode(PNIO_MODE_OFFLINE);" should be used to set the CP controller status to "Offline". Then the PN controller can be unregistered using the "PNIO_controller_close" command.

5.4 Cyclic program

Flowchart

The following figure gives you a rough overview of the contents of the application example's main method.

Figure 5-1

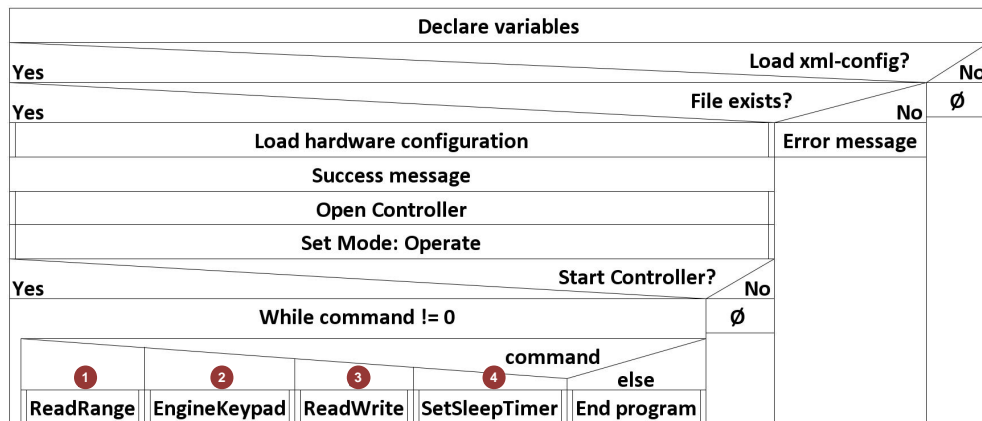


Table 5-2

No.	Name	Explanation
1.	ReadRange	Read and display a range from the process image from a start to an end address.
2.	EngineKeypad	Starts baking mode. The example is controlled by keypad. This routine can be compared to the cyclic operation of the OB1 block of a Siemens PLC.
3.	ReadWrite	Starts a routine included in the PN Driver to manually read or write any process image values. This function is mainly used to debug the application.
4.	SetSleepTimer	After each read or write in the application example, a wait time of 20 milliseconds is granted by default. When debugging the application, this ensures that no interrupts are lost (if the program works correctly, no interrupts will be lost; this function is for debugging only). Using this function, any value can be entered instead of 20 milliseconds. When 0 is entered, no wait time is granted and real-time mode starts.

NOTE

If you want to trigger an interrupt manually, first read out the interrupt's memory address.

Normally, the value should be read once at the very beginning in the control program to activate the interrupt.

Overview of the structure

The following screenshot shows the top level of the code structure of the bakery application.

Figure 5-2

```

/*****
/*
/* Description :
/*
/* User Guide Application for PNDriver
/*
/*
/*****

+ Includes
+ Global Definitions
+ Global Declarations
+ Support Functions (Read/Write/Callbacks/Start/Stop/Shutdown)
+ Cyclic PN-Controller Bakery Application (The Main Program)
+ Entry Point (MAIN)
    
```

Table 5-3

No.	Region	Description
1.	Includes	Includes all libraries and header files used. Worth mentioning: <ul style="list-style-type: none"> • pniobase.h (=IO-Base library) • pnd_test.h (=PND_Test application) • eps_*.h (=PN Driver library) Note: The “eps_*.h” includes contain a reference to the WinPcap library. This means that all libraries from the chart are available.
2.	Global Definitions	Global #define definitions.
3.	Global Declarations	Includes the declarations of the message structures used by the callback functions for display in the console application.
4.	Support Functions	This area contains all functions that allow a cyclic automation program. Callback Notification Functions <ul style="list-style-type: none"> • Triggered by the registered callbacks of the PN devices • Use the structures from Global Declarations to display messages in the console. Callback Functions (of PNIO_Open_Controller) <ul style="list-style-type: none"> • Interrupt callbacks are processed here • Read/write callbacks are processed here Callback Functions (of StartPNController) <ul style="list-style-type: none"> • Sample callback function that can be filled as desired. Triggered when the PN controller changes its status (Start/Stop/Offline)

No.	Region	Description
		<p>Flush Process Image Buffer</p> <ul style="list-style-type: none"> Function that clears the process image buffer to start the PN controller on a data consistent basis. <p>Read/Write Operations</p> <ul style="list-style-type: none"> These two functions allow read/write operations from/to the process image. These functions are very important for the main program. <p>Load HW-Configuration, Start/Shut Down PN-Controller</p> <ul style="list-style-type: none"> Contains the functions for loading the hardware configuration and for starting and shutting down the PN controller.
5.	Cyclic PN-Controller Bakery Application	Cyclic main program that controls the application example. Will be described in greater detail in the next section.
6.	Entry Point	<p>The “main” method is the entry point of this application. In “main”, ...</p> <ul style="list-style-type: none"> ... the hardware configuration is loaded. ... the PN controller is made ready to run. ... the main menu is displayed. ... the PN controller is closed when the user exits the program.

5.5 Hardware interrupts

In the PN controller code, alarms and interrupts are registered and processed as callback functions.

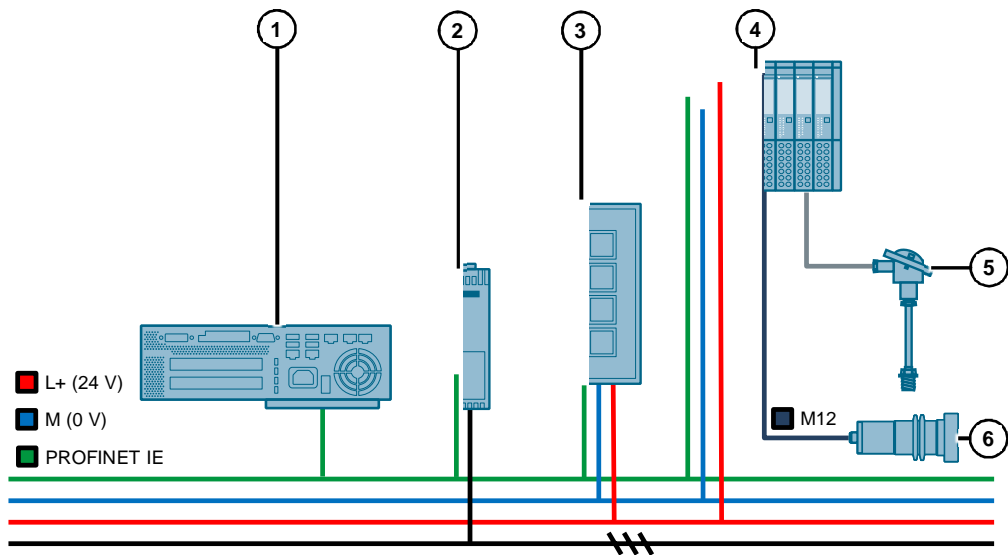
They can be registered either when starting the controller with the “PNIO_controller_open” command as (up to 3) parameters or at any time in the program code using “PNIO_register_cbf()”.

The implemented callback function for interrupts has all interrupts activated. Currently, they are displayed as notifications in the console. The function that handles the interrupts is `callback_for_alarm_ind()`.

Note

Interrupts that are not standard interrupts (such as inserting and removing modules or cables) must be configured in TIA Portal in the hardware configuration and registered in the PN controller.

The standard interrupts only need to be registered.



The following table describes the components of the above figure.

Table 6-1

No.	Description
1.	SIMATIC IPC627C (Industrial PC)
2.	SINAMICS G120C (frequency inverter)
3.	SIMATIC HMI KP8F (keypad)
4.	SIMATIC ET 200SP (distributed I/O module)
5.	PT 100 (temperature sensor)
6.	SONAR BERO (sonar)

The following table describes how to connect the modules.

Table 6-2

	Action
1.	Attach the individual modules to a suitable rack.
2.	Connect the power supply to the power grid (230V AC). Connect the frequency inverter and the motor to the appropriate three-phase supply.
3.	Connect the power supply to the PN devices. Ensure that the polarity is correct.
4.	Connect the PROFINET interfaces of the PG/PC to the ones of the PN device.

6.2 Startup

This chapter describes the steps to start up the sample code.

These steps require that the hardware configuration be configured and that the hardware be installed correctly.

1. Open the CD and copy the contents
2. Navigate to the `*\source\pnedriver\vs` folder and
3. open the Visual Studio project (`pnedriver_test_w32.sln`)
4. In the project structure, navigate to the `pnedriver_test_w32/pnd/test` folder
5. and insert the “`user_guide.c`” file from the user guide .zip file
6. Open the “`pnd_test.c`” file
7. Press CTRL+A to select all lines
8. Press CTRL+K+C to comment out all lines (CTRL+K+U undoes this operation). This overwrites the main method and makes the main method from the “`user_guide.c`” file the current method.
9. Compile the code either as “release” (more efficient code) or “debug” (debug options)
10. Now save the XML hardware configuration to the source folder. (For release compilation, this is `\source\pnedriver\gen\release`; for debug compilation, `\source\pnedriver\gen\debug` is the source folder)
11. Now run the `pnedriver_test_w32.exe` file

7 Operation of the Application

7.1 Overview of the user interface

The following figure shows the console user interface of the bakery program.

Figure 7-1

```

Load Hardware Configuration File? [y/n]
y
Loading PNDriverConfiguration.xml

WinPcap found network adapters:
-----
0 ... 00:22:75:d6:cf:46 - USB2.0 to Gigabit Ethernet Adapter
-----

Enter the number of the network adapter: 0
SERU_CP_startup(<) Result: 0

Loading PNDriverConfiguration.xml complete

Start PLC Program? [y/n]
y

-----
1... ReadRange
2... EngineKeypad
3... ReadWrite
4... SetSleepTimer
0... Exit Program

```

The top operation follows the structured chart of the previous chapter.

The system first loads the hardware configuration, then selects the network adapter that has access to the PN devices and finally displays the program's main menu with its five options.

7.2 Main menu

The following section explains the functionalities of the five options:

7.2.1 ...ReadRange

Figure 7-2

```

Please enter the beginning of the range
0
Please enter the end of the range
5
Address: 0x0 Value: 0x0
Address: 0x1 Value: 0x0
Address: 0x2 Value: 0x0
Address: 0x3 Value: 0x0
Address: 0x4 Value: 0x0
Address: 0x5 Value: 0x0

```

7.2 Main menu

After entering the start and end address, ReadRange allows the user to read a number of input addresses out of the process image.

This functionality was added to the example for debugging purposes and for activating configured hardware interrupts.

7.2.2 ...EngineKeypad

This option starts the bakery application. It is controlled via an illuminated keypad and via console messages.

Console view

Figure 7-3

```
#####
#####
##### Motor Control Word: 0xeb 0x40
##### Motor Target Value: 0x0 0x0
##### Motor RPM: 0
##### Motor Status: Connection Established
##### SONAR IN REACH: False
##### Temperature: 25.3°C
##### CycleTime: 114 milliseconds
#####
#####
```

Note

The cycle time of 114 milliseconds was measured with a read/write delay of 20 milliseconds for both read/write in debug mode. This value does not reflect the achievable speed of a PN Driver application (1 millisecond).

The following table describes the individual values:

Table 7-1

No.	Value	Description
5.	Motor Control Word	The 2-byte control word for the motor's frequency inverter. Among other things, it specifies whether and in which direction the motor should turn.
6.	Motor Control Value	The 2-byte control value for the motor's frequency inverter. It allows you to adjust the motor speed.
7.	Motor RPM	The motor's rotations per minute. This value is calculated from the current control value of the frequency inverter.
8.	Motor Status	Status of the motor. This value is implied from the inverter's status word.
9.	SONAR IN REACH	Once the desired preset degree of expansion of a cake is reached, this status is set to "True".
10.	Temperature	Displays the temperature of the current cake.
11.	CycleTime	Displays the time of a completed cycle of the bakery program.

Keypad

The following table explains the display and operation of the keypad.

When the application is not in the “EngineKeypad” function, all buttons of the keypad are displayed in white.

Figure 7-4

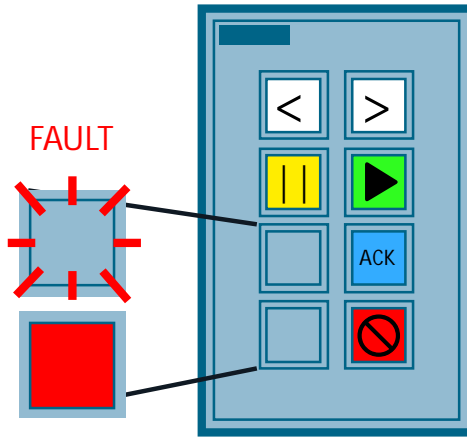


Table 7-2

Button	Meaning
Red	Exit bakery application and re-enable for console application.
Blue	Acknowledge frequency inverter fault (indicated by the flashing of one of the two bottom left buttons on the keypad).
Green	Start motor and run it at 10 rpm in counterclockwise direction. The cakes are moved into or out of the oven.
Yellow	Stop motor. Now the motor has 0 rpm. The cakes are in the oven.
White “<”	Counterclockwise rotation of motor at 1500 rpm.
White “>”	Clockwise rotation of motor at 1500 rpm.

© Siemens AG 2015 All rights reserved

7.2.3 ...ReadWrite

This option starts a program of the PND_Test application for manually reading from and writing to the process image. It is controlled via console inputs.

Figure 7-5

```

1...PNIO_data_write
2...PNIO_data_read
0...return to main menu
2
-----
I Address:
0
PNIO_data_write() ApplHandle: 0, Addr: 0, readwritedata: 0
I Data: 0x0 remState: 0 (0=GOOD, 1=BAD)
PNIO_data_read() response: 0x0
    
```

The figure shows the read operation from an output address; in this case, the read value is “0x0” (response).

Read operations are automatically performed from output addresses, write operations are automatically performed to the process image input addresses.

This program mainly serves the purpose of developing a custom application. The program allows you to test an idea step by step before hard-coding it in a cyclic program.

7.2.4 ...SetSleeptimer

This option starts a program for setting a wait time after each read and write operation of the User_Guide application in milliseconds.

This wait time exists to ensure correct display of all interrupts while debugging a cyclic application.

Figure 7-6

```
Please enter the wait time <ms> after reads/writes for alarm detection
10
SLEEPTIMER changed to 10 milliseconds
This amounts to an average cycle time of about 43.773398 milliseconds
```

For a running cyclic program, a wait time of 0 milliseconds should always be set.

The displayed average cycle time is based on observed values from the User_Guide application. For your custom program, the specified time would no longer be accurate.

It is controlled via console inputs.

8 Appendix: Implementing your own PN Controller

This section provides a stand-alone description (independently of the application example) of how to write your own PN controller.

8.1 Steps from the DVD to the executable program

The following table shows the specific steps for implementing a PN controller application in a Windows operating system.

Table 8-1

No.	Description
1.	Download the free WinPcap Developer's Pack and install it.
2.	Copy the contents of the PN Driver DVD to your Windows file system.
3.	Download TIA Portal 2012 and install it.
4.	Download Visual Studio 2012 and install it.
5.	Download the current HSP for TIA Portal V12 and install them.
6.	Configure the hardware configuration of your PN controller in TIA Portal as described in the previous chapter.
7.	Open the Visual Studio project from your PN Driver DVD in Visual Studio. To do this, open the following file: <code>\source\pn_driver\vs\pn_driver_test_w32.sln</code>
8.	If you want to write a complete custom application, create a new .c file in the <code>pn_driver_test_w32/pnd/test</code> folder. To use the ready-made functions, copy the <code>user_guide.c</code> file to this folder.
9.	Compile your written program and the hardware configuration in TIA Portal.
10.	Save both the compiled .exe file and the generated xml file to one folder.
11.	Run the program using the .exe file.

8.2 Programming a cyclic program

Requirements for creating the program

Note

We recommend that you base your application on the functions of the `user_guide.c` file. This is the bakery application example. We explicitly recommend that you directly overwrite the `EngineKeypad` loop with your application. In this way, you retain the full debugging capabilities of the application and have a certain guarantee that the basic aspects of the program should work from the start.

When you have completed the structure for loading the hardware configuration, starting and stopping the PN controller and registering the callback functions (or applied it from our example), you can create your cyclic program.

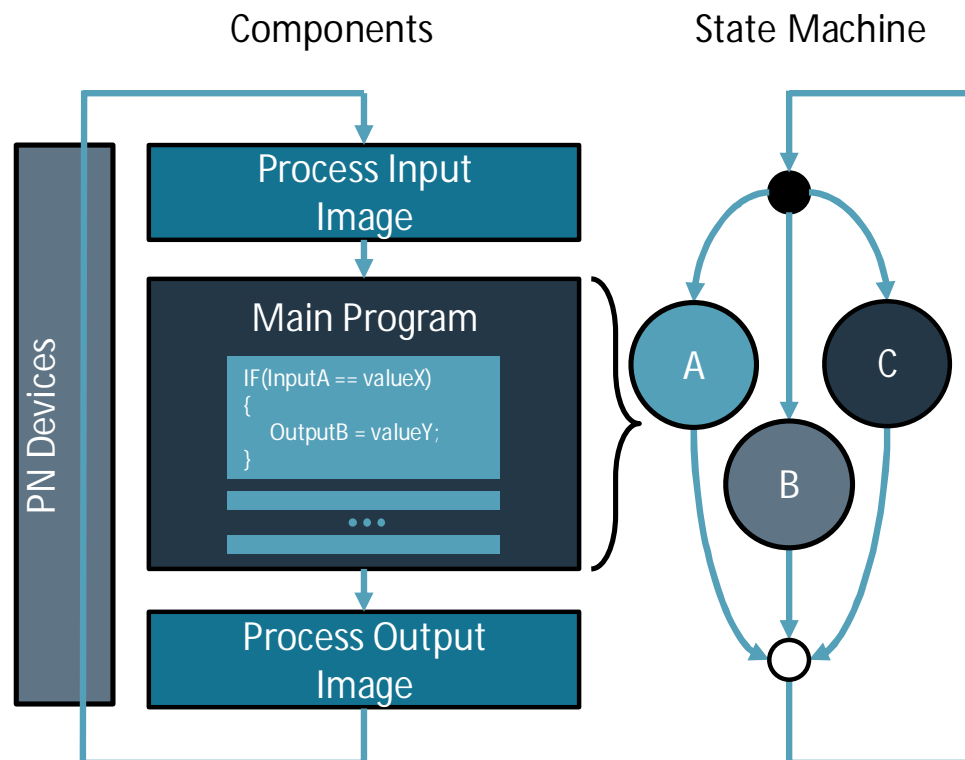
Rules for a cyclic program

A cyclic program should follow the rules below:

1. (Large) loop calls within the main cycle should be avoided as otherwise an acceptable cycle time can no longer be ensured.
2. To be able to terminate the application, the main cycle should have an end condition that depends on the change of a single output address bit (protective switch). This provides both an emergency stop option and a prerequisite for forced troubleshooting in the event of a crash. In addition, stopping the main cycle would be difficult in other cases as you would have to wait until a condition is met that the user cannot influence.
3. At the end of the cycle, the main cycle is to move its current state to a desired state using conditions (if-else/case). The aim is to avoid processing a state over multiple cycles – as otherwise the real-time functionality could no longer be ensured.

Typical structure of a cyclic program

Figure 8-1



The left part of the figure shows that the process image must be stable at the start and end of each cycle to ensure that the PN devices do not make errors.

The right part shows that a number of conditions must be met within a cycle and queries without a specific result must no longer be present.

The blue main cycle (similar to the OB1 cycle) on the left side corresponds to a loop. The loop itself specifies the clock of the main program. This means that the PN controller is always provided with the current process image in real time – therefore, the main cycle must be fast enough to detect the fastest relevant PN device change.

8.3 Tips and tricks

In this section, you will find help regarding frequent problems and general helpful notes on working with the PN Driver.

8.3.1 The PN device is not accessible

How does the fault manifest itself?

- The PN device flashes. This indicates that there is no connection.
- Bad reads or writes.

Solution

1. Make sure that the PN controller is in “Operate” mode.

This is done with the following command:

```
pnd_test_set_mode(PNI 0_MODE_OPERATE);
```

2. Make sure that you can see the device in the network with TIA Portal and that no IP address conflict has occurred.

It is also possible that you have configured a module name in TIA Portal that has not yet been assigned to the module.

3. To assign a name, go to the “Online & diagnostics” view of the module in TIA Portal and go online. The “Assign name” menu item allows you to assign a PN name.

8.3.2 The PN device permanently indicates a fault or interrupt

How does the fault manifest itself?

- PN device indicates a fault
- PN device indicates an interrupt

Solution

Make sure that you have configured the correct modules with the correct firmware. Furthermore, you can read out the diagnostic buffer in the “Online & diagnostics” view to obtain more information.

In the case of interrupts, make sure that the interrupt is not triggered correctly.

8.3.3 The PN device alternately goes online and offline

How does the fault manifest itself?

- The PN device alternately goes online and offline

Solution

Check the configured cycle time on the device in the PN Driver’s hardware properties in TIA Portal. It is possible that the cycle time of your program exceeds the configured cycle time. In this case, you can configure a higher cycle time in the PN Driver and move the generated hardware configuration to the PN-Controller.exe folder.

8.3.4 Will the PN controller work on a diagnostic/mirror port of a switch?

No. This configuration generates duplicated packets for the PN controller and can cause communications problems.

Disable all mirror and monitor ports of your switches to ensure the correct functionality of the PN controller.

8.3.5 My network adapter is not detected when I restart the software

How does the fault manifest itself?

In the console application, no – or a wrong – adapter is displayed in adapter selection.

Solution

Try to restart the network packet filter (npf) service or check if it is running. This service is required in order to process the PN packets. To do this, open a command prompt with administrator rights, type “net stop npf” and then type “net start npf”.

If the adapter is still not available, it might not be compatible or is not supported. In this case, you can try to update the adapter drivers and restart the PN.

8.4 Characteristics of the PN Driver

Configured interrupts are only registered after reading their memory address

The interrupt addresses configured in the hardware configuration must first be read by the PN controller before they can be triggered. This is a characteristic of PN Driver version 1.0 and should be fixed with the next version.

PN Driver is only available in the hardware catalog

The project wizard cannot select the PN Driver as the S7 controller. It must be selected separately in the hardware catalog in the TIA hardware configuration. This is not an error. Regarding its device type, the PN Driver is a PC station. PC stations are usually only available in the hardware catalog.

Some PN devices must be added as a GSD file in TIA Portal

In the case of some devices that are difficult to configure, for example motors, it is possible that they can only be added as a GSD file. If available, they have to be downloaded from Siemens Industry Online Support (SIOS) for each module.

Note

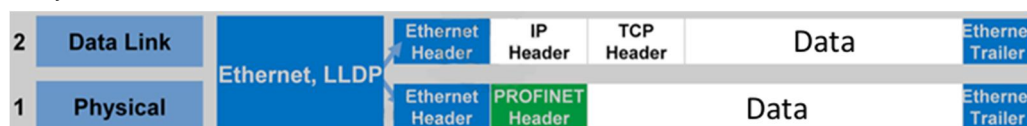
Siemens Industry Online Support can be found here:
<https://support.industry.siemens.com/cs/start?lc=en-DE>

9 Appendix: Introduction to PROFINET

9.1 Overview

PROFINET is based on Ethernet. Actually, the only difference is that within an Ethernet PDU, a PROFINET header is added at ISO/OSI layer 1. Therefore, the entire functionality of Ethernet and the protocols based on it are also ensured by PROFINET.

The following figure shows the visual assignment of PROFINET to the ISO/OSI 7-layer model:



Due to their importance, PDUs with IO data are always given priority in PROFINET in the send cycle of the Ethernet port. This means that – provided that PROFINET PDUs exist – they are prioritized over all other PDUs with less critical contents (e.g., email, HTTP, IP, etc.) and forwarded by switches.

As is common at the two lower ISO/OSI layers, PROFINET communication is based on the MAC address of devices.

Note: The fact that the configuration of PN devices with programming units is performed via IP addresses is useful only to the user. Communication between the devices is based (if not specified in this way with special blocks) on MAC addresses.

Note: If you want to monitor your communication using network protocol analysis tools such as Wireshark: “PROFINET Real-Time” PDUs can be reliably identified by the “Type” Ethernet PDU field with the “0x8892” ID.

9.2 Device types

Furthermore, the PROFINET header distinguishes between the following device types:

Table 9-1

Device type	Header type
IO controller	An automation controller, usually a PLC. In the case of the PROFINET IO Driver, the used computer with the driver becomes the IO controller.
IO device	A field device controlled by the IO controller. It consists of modules and submodules which in turn have inputs and outputs (for example, I0.0 or O10.5). In fact, these are all modules that are not the controller itself.
IO supervisor	A programming unit that parameterizes and diagnoses IO devices Furthermore, it usually loads the automation program to the IO controller.

9.3 Basic functions

The basic functions that can be accessed via the PROFINET header at layer 1 are the following ones:

Table 9-2

Function	Name	Description	Protocol	Addressing
Cyclic I/O	IO Data CR	The actual IO data at the inputs and outputs. In each cycle, the current data is exchanged between the IO device and the IO controller.	Ethernet	MAC address
Parameters	Record Data CR	For acyclic parameterization, diagnostics and configuration. Designed for rare one-time communication that is not an emergency.	UDP/IP	IP address
Alarms	Alarm CR	Acyclic, time-critical emergency notifications. Transmitted in real time, reception must be confirmed; otherwise, the notifications will be resent.	Ethernet	MAC address

10 References

Table 10-1

	Subject	Title
\1\	Siemens Industry Online Support	http://support.automation.siemens.com
\2\	Download page of the entry	https://support.industry.siemens.com/cs/ww/en/view/109478514
\3\	WinPcap Developer Resources download page	https://www.winpcap.org/devel.htm
\4\	TIA Portal V12 download page	https://support.industry.siemens.com/cs/ww/en/view/67639464
\5\	Visual Studio 2012 Express download page	https://www.microsoft.com/en-us/download/details.aspx?id=34673
\6\	Download page for the HSP for TIA Portal that include the PN Driver	https://support.industry.siemens.com/cs/en/de/view/72341852

11 History

Table 11-1

Version	Date	Modifications
V1.0	09/2015	First version