PENKO Engineering B.V.

Your Partner for Fully Engineered Factory Solutions

PROFINET®

Protocol description:
PENKO Profinet

PENKO
an ETC Company

Table of Contents

PENKO Profinet protocol

1 Introduction

This manual describes how to install, configure and use the Penko Profinet IO module. The required GSDML file is available for downloading at www.penko.com/Support/Software. The Penko Profinet module contains four submodules:

1. Weigher Input Module
2. Remote Command Module
3. Inputs Outputs Markers Module
4. Diagnostics Module

1.1 Penko Profinet implementation

The Penko devices that supports Profinet are IO device modules. The Penko Profinet devices (PN devices) are not Profinet controllers (PN controller). Figure 1 below shows the schematic overview of the Penko Profinet implementation. In chapter 3 the Penko Profinet device specific information is listed. It is also possible to connect other devices that support Profinet to the Penko devices.

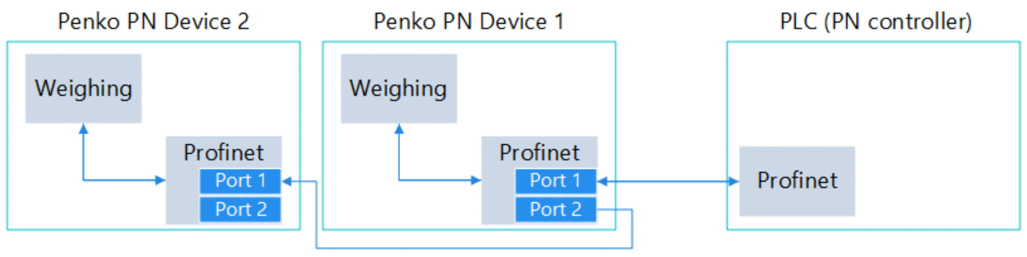


Figure 1 Schematic view Penko Profinet implementation

1.2 Modification history

Version	Topic	Date
1.0	Profinet protocol manual created.	04/2020
1.1	Indicator parameters changed	05/2020

2 Penko Profinet IO module

The Penko Profinet IO module contains four modules. It is not required to use all the modules, only use the modules and channels that are needed. The cycle input data from each module is described in paragraph 2.3.

2.1 Basics

This document describes the General Station Description of the xml file (GSDML) for the SGM760/SGM860 and 1020 indicator/controller.

SGM760/SGM860:	GSDML-V2.35-PENKO-SGM-20200401.xml
Penko 1020 Indicator/controller:	GSDML-V2.35-PENKO-1020-20200401.xml

The SGM760 is always an indicator. The 1020 and SGM860 can be an indicator or a controller with a predefined program for check weigher, belt weigher or mono filler.

The following PENKO devices support Profinet.

Device	Profinet
SGM760	Yes, starting from firmware version V1.7.1.9.0.3
SGM860	Yes, starting from firmware version V1.7.1.9.0.3
1020 Indicator/controller	Yes, starting from firmware version V1.6.1.9.0.3

2.2 Supported communication settings

The following settings are supported.

Setting	Parameter
Send clock (ms)	1
Reduction ratio	8
Watchdog (ms)	48
Conformance class	B

2.3 Submodules

Table 1 below lists all available submodules with corresponding channels. All channels can be individually mapped to a variable in the PLC program.

Table 1 Presentation of provided data in Penko IO modules

Module	Data type	Provided data (channels)
Weigher Input Module	**Cyclic input data**	
	DInt	Net
	DInt	Gross
	DInt	Tare
	DInt	Preset Tare
	Byte	Status 0 = Weight is valid 1 = Stable weight 2 = Net weight 3 = Center of zero 4 = Zero is set 5 = Floatingpoint 6 = Command is ready 7 = Command is in execution mode
	Byte	Decimal point position in non floating point mode
	Byte	Range, active multiple range/multi interval, 0 is none. i.e. 1 = e1, 2 = e2, etc
Remote Command Module	**Cyclic input data**	
	DInt	Result data
	Byte	Command Result Code
	Bool	Status 0 = Weight is valid 1 = Stable weight 2 = Net weight 3 = Center of zero 4 = Zero is set 5 = Floatingpoint 6 = Command is ready 7 = Command is in execution mode
	Cyclic output data	
	DWord	Command
	DWord	Parameter
	DInt	Exchange
Inputs Outputs Markers Module	**Cyclic input data**	
	DWord	Inputs 1 – 3
	DWord	Outputs 1 – 4
	DWord	Markers 401 – 432
	Cyclic output data	
	DWord	Markers 969 – 1000
Diagnostics Module	**Cyclic input data**	
	DInt	Slave sequence counter, integrated Profinet ASIC
	DInt	Master sequence counter, integrated Main CPU

2.4 Submodule parameters

The weigher input and inputs outputs markers module support different parameters. In Table 2 the available parameter for each module are listed. Based on the selected program (indicator, check weigher, monofil or belt weigher) the config parameters represent a different setting. In appendix I, the config parameters each program are described. Chapter 3 describes where to find the module parameters.

See appendix I for a full description of the allowed values from the net weight selector and the config parameters.

Table 2 PENKO IO module parameters

Module	Field name	Data type	Byte offset	Default value	Allowed values
Weigher Input	Weight format	Byte	0	0: fix point	0 1 2 3 4 5 (decimal point position)
	Net weight selector	Byte	0	0: weigher	0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18
Inputs Outputs Markers	Write Configuration	Byte	0	0	
	Config 1	DInt	0	0	
	Config 2	DInt	4	0	
	Config 3	DInt	8	0	
	Config 4	DInt	12	0	
	Config 5	DInt	16	0	
	Config 6	DInt	20	0	
	Config 7	DInt	28	0	
	Config 8	DInt	32	0	
	Config 9	DInt	36	0	
	Config 10	DInt	40	0	
	Config 11	DInt	44	0	
	Config 12	DInt	48	0	
	Config 13	DInt	52	0	
	Config 14	DInt	56	0	
	Config 15	DInt	60	0	
	Config 16	DInt	64	0	
	Config 17	DInt	68	0	
	Config 18	DInt	72	0	
	Config 19	DInt	76	0	
	Config 20	DInt	80	0	
	Config 21	DInt	84	0	
	Config 22	DInt	88	0	
	Config 23	DInt	92	0	
	Config 24	DInt	96	0	
	Config 25	DInt	100	0	
	Config 26	DInt	104	0	
	Config 27	DInt	108	0	
	Config 28	DInt	112	0	
	Config 29	DInt	116	0	

2.5 Supported RPC commands

With the remote command module, it is possible to execute commands. For the RPC_indicator_xx commands it is possible to read and write specific data. The <result_data> holds the reply value. After entering a wrong parameter, the Penko device returns the code RPC_PARAMETER_ERROR.

2.5.1 Commands

The available commands are listed in Table 3.

Table 3 Penko Profinet IO module commands

ID	Command	Description
0	RPC_PROCEDURE_NONE	No operation
1	RPC_INDICATOR_COMMAND	Indicator command, i.e. set or reset tare
2	RPC_INDICATOR_READ	Read indicator parameters
3	RPC_INDICATOR_WRITE	Write indicator parameters
4	RPC_INDICATOR_CALIBRATE	Calibrate indicator
5	RPC_REGISTERS_READ	Read registers
6	RPC_REGISTERS_WRITE	Write registers
7	RPC_INDICATOR_VALUE	Read indicator value
8	RCP_CONFIG_READ	Read Configuration parameters
9	RCP_CONFIG_WRITE	Write Configuration parameters
10	RCP_RECIPE_READ	Read Recipe parameters
11	RCP_RECIPE_WRITE	Write Recipe parameters

2.5.2 Parameters indicator commands

In Table 4 the indicator command parameters are listed.

Table 4 Indicator command parameters

ID	Parameter	Description	Return data
0	RPC_INDICATOR_NONE	No operation	
1	RPC_INDICATOR_ZERORESET	Indicator set zero	Result_code, see paragraph 2.6.2
2	RPC_INDICATOR_ZEROSET	Indicator reset zero	
3	RPC_INDICATOR_TAREOFF	Indicator tare off	
4	RPC_INDICATOR_TAREON	Indicator on	
5	RPC_INDICATOR_PRESETTARE	Indicator set preset tare	Actual tare
6	RPC_INDICATOR_TOTALIZE	Totalize weight	Totalize weight
7	RPC_INDICATOR_RESETPEAK	Reset peak weight	Last peak weight
8	RPC_INDICATOR_RESETVALLEY	Reset valley weight	Last valley weight
9	RPC_INDICATOR_HOLD	Copy actual net weight in hold	Last hold weight

2.5.3 Parameters indicator read

In Table 5 the indicator read functions are listed.

Table 5 Indicator read functions

ID	Parameter	Description
0	RPC_INDICATORREAD_NONE	No operation
1	RPC_INDICATORREAD_TAC	TAC code
2	RPC_INDICATORREAD_NAME	Indicator name in steps of 4 chars
3	RPC_INDICATORREAD_UNIT	Indicator name in steps of 4 chars
4	RPC_INDICATORREAD_MAXLOAD	Maximum load weight
5	RPC_INDICATORREAD_OPERATIONMODE	Operation mode, 0 = OIML, 1= Industrial mode
6	RPC_INDICATORREAD_SAMPLERATE	ADC conversion rate 0=10/s, 1=20/s, 2=25/s, 3=50/s 4=100/s, 5=200/s, 6=400/s, 7=800/s, 8=1600/s
7	RPC_INDICATORREAD_STEPSIZE	Weight step size resolution 0=1, 1=2, 2=5, 3=10, 4=20, 5=50, 6=100, 7=200, 8=500
8	RPC_INDICATORREAD_DECIMALPOINT	Decimal point position 0 = no decimal, 1 = 0.0, 2 = 0.00, 3 = 0.000, 4 = 0.0000, 5 = 0.00000
9	RPC_INDICATORREAD_STABLERANGE	Stable range in weight resolution
10	RPC_INDICATORREAD_STABLETIME	Stable range as time in milliseconds
11	RPC_INDICATORREAD_ZEROTRACKRANGE	Zero tracking range in weight resolution
12	RPC_INDICATORREAD_ZEROTRACKSTEP	Zero tracking step size in weight resolution
13	RPC_INDICATORREAD_ZEROTRACKTIME	Zero tracking time in milliseconds
14	RPC_INDICATORREAD_RANGEPARTS	Multiple-range/Multi-interval number of parts
15	RPC_INDICATORREAD_RANGEMAXSTEP	Multiple-range/Multi-interval maximum step size 0=1, 1=2, 2=5, 3=10, 4=20, 5=50 6=100, 7=200, 8=500
16	RPC_INDICATORREAD_RANGEMODE	Multiple-range/Multi-interval mode 0=Multiple-range, 1= Multi-interval
17	RPC_INDICATORREAD_DIGITALFILTER	Digital filtering: 0= no filter Dynamic application: 1=Dynamic 1Hz, 2=Dynamic 1.4Hz, 3=Dynamic 2.5Hz, 4=Dynamic 5Hz, 5=Dynamic 10Hz 6=Dynamic 20Hz, 7=Dynamic 40Hz Static application: 8=Static 1Hz, 9=Static 1.4 Hz, 10=Static 2.5 Hz, 11=Static 5Hz, 12=Static 10Hz, 13=Static 20Hz, 14=Static 40Hz
18	RPC_INDICATORREAD_FILRTMAVG	Frequency moving Average filer 0=Off, 1=1Hz, 2=2Hz, etc. till 50=50Hz
19	RPC_INDICATORREAD_DISPLAYRATE	Display update rate 0= 1x/s, 1=2x/s, 2=3/s, 3=5x/s 4=10x/s, 5=25x/s, 6=50x/s
20	RPC_INDICATORREAD_DISPLAYMODE	Display indicator selection 0=DISPLAY OIML NET (multiple range/multi interval, approved readout), 1=FAST GROSS (fixed step size),

		2=FAST NET (fixed step size), 3=DISPLAY GROSS (multiple range/multi interval), 4=DISPLAY NET (multiple range/ multi interval), 5=TARE, 6=PEAK, 7=VALLEY, 8=HOLD, 9=FAST NET 10x, 10=FAST GROSS 10X, 11=PRESET TARE, 12=CALCULATED GROSS, 13=CALCULATED TARE, 14=TARE 10X, 15=PEAK 10X, 16=VALLEY 10X, 17=HOLD 10X, 18=SIGNAL in mV

2.5.4 Parameters indicator write

Table 6 shows the indicator write parameters. These are the same as indicator read but then as write table.

Table 6 Indicator write functions

ID	Parameter	Description
0	RPC_INDICATORWRITE_NONE	No operation
1	RPC_INDICATORWRITE_TAC	TAC code
2	RPC_INDICATORWRITE_NAME	Indicator name in steps of 4 chars
3	RPC_INDICATORWRITE_UNIT	Indicator name in steps of 4 chars
4	RPC_INDICATORWRITE_MAXLOAD	Maximum load weight
5	RPC_INDICATORWRITE_OPERATIONMODE	Operation mode, 0 = OIML, 1= Industrial mode
6	RPC_INDICATORWRITE_SAMPLERATE	ADC conversion rate 0=10/s, 1=20/s, 2=25/s, 3=50/s 4=100/s, 5=200/s, 6=400/s, 7=800/s, 8=1600/s
7	RPC_INDICATORWRITE_STEPSIZE	Weight step size resolution 0=1, 1=2, 2=5, 3=10, 4=20, 5=50, 6=100, 7=200, 8=500
8	RPC_INDICATORWRITE_DECIMALPOINT	Decimal point position 0 = no decimal, 1 = 0.0, 2 = 0.00, 3 = 0.000, 4 = 0.0000, 5 = 0.00000
9	RPC_INDICATORWRITE_STABLERANGE	Stable range in weight resolution
10	RPC_INDICATORWRITE_STABLETIME	Stable range as time in milliseconds
11	RPC_INDICATORWRITE_ZEROTRACKRANGE	Zero tracking range in weight resolution
12	RPC_INDICATORWRITE_ZEROTRACKSTEP	Zero tracking step size in weight resolution
13	RPC_INDICATORWRITE_ZEROTRACKTIME	Zero tracking time in milliseconds
14	RPC_INDICATORWRITE_RANGEPARTS	Multiple-range/Multi-interval number of parts
15	RPC_INDICATORWRITE_RANGEMAXSTEP	Multiple-range/Multi-interval maximum step size 0=1, 1=2, 2=5, 3=10, 4=20, 5=50 6=100, 7=200, 8=500
16	RPC_INDICATORWRITE_RANGEMODE	Multiple-range/Multi-interval mode

		0=Multiple-range, 1= Multi-interval
17	RPC_INDICATORWRITE_DIGITALFILTER	Digital filtering: 0= no filter Dynamic application: 1=Dynamic 1Hz, 2=Dynamic 1.4Hz, 3=Dynamic 2.5Hz, 4=Dynamic 5Hz, 5=Dynamic 10Hz 6=Dynamic 20Hz, 7=Dynamic 40Hz Static application: 8=Static 1Hz, 9=Static 1.4 Hz, 10=Static 2.5 Hz, 11=Static 5Hz, 12=Static 10Hz, 13=Static 20Hz, 14=Static 40Hz
18	RPC_INDICATORWRITE_FILRTMAVG	Frequency moving Average filer 0=Off, 1=1Hz, 2=2Hz, etc. till 50=50Hz
19	RPC_INDICATORWRITE_DISPLAYRATE	Display update rate 0= 1x/s, 1=2x/s, 2=3/s, 3=5x/s 4=10x/s, 5=25x/s, 6=50x/s
20	RPC_INDICATORWRITE_DISPLAYMODE	Display indicator selection 0=DISPLAY OIML NET (multiple range/multi interval, approved readout), 1=FAST GROSS (fixed step size), 2=FAST NET (fixed step size), 3=DISPLAY GROSS (multiple range/multi interval), 4=DISPLAY NET (multiple range/ multi interval), 5=TARE, 6=PEAK, 7=VALLEY, 8=HOLD, 9=FAST NET 10x, 10=FAST GROSS 10X, 11=PRESET TARE, 12=CALCULATED GROSS, 13=CALCULATED TARE, 14=TARE 10X, 15=PEAK 10X, 16=VALLEY 10X, 17=HOLD 10X, 18=SIGNAL in mV

2.5.5 Parameters indicator calibrate

Enable (set CAL code) calibration first before start calibrating, otherwise you get the result error code RPC_ACCESSDENIED.

Table 7 Indicator calibration functions

ID	Parameter	Description
0	RPC_INDICATORCALIBRATE_NONE	No operation
1	RPC_INDICATORCALIBRATE_CAL	Get CAL code
2	RPC_INDICATORCALIBRATE_ENABLE	Set CAL code to enable calibration functions
3	RPC_INDICATORCALIBRATE_ZERO	Zero level calibration, no weight on scale
4	RPC_INDICATORCALIBRATE_GAIN	Gain/span calibration, weight on scale
5	RPC_INDICATORCALIBRATE_DEADLOAD	Deadload correction, dead load on scale

2.6 Alarms & Error codes

There are two kinds of error messages; weigher alarms and command errors. The weigher alarms can be found at the Weigher Input Module diagnostics page. These alarms are generated by the weigher software and automatically sent to the PN-Controller. The command errors are returned after entering a wrong command and/or parameter.

2.6.1 Alarms

The Weigher Input Module contains three kinds of alarms.

Number	Alarm code	Description
0	Weigher error 0	Valid weigher data, alarm is cleared
1	Weigher error 1	Weight above Maximum load
2	Weigher error 2	Overload by ADC conversion
3	Weigher error 3	Underload by ADC conversion

2.6.2 Command errors

The following table lists the command response code after executing a command.

Table 8 Command response codes

ID	Code	Description
0	RPC_SUCCES	Command executed success
1	RPC_EXECUTING	Command is executing
2	RPC_UNKNOWN_COMMAND	Unknown Penko Profinet command
3	RPC_UNKNOWN_FUNCTION	Unknown function
4	RPC_NOTIDLE	Busy executing a command
5	RPC_FAILED	Command executing failed
6	RPC_ERROR	Command error
7	RPC_NOT_ALLOWED	Command executing not allowed
8-127	RESERVED	Reserved error codes
128	RPC_PARAMETER_ERROR	Invalid parameter set
129	RPC_NOTSTABLE	Weight not stable
130	RPC_NEGATIVE	Weight negative
131	RPC_NO_TARE	Tare not set
132	RPC_OUTOFRANGE	Weight out of range
134	RPC_NOT_STABLE	Weigher not stable
135	RPC_ABOVE_MAXLOAD	Weight is above maxload
136	RPC_BELOW_ZERO	Weigher below zero
137	RPC_NOT_IN_ZERO_RANGE	Weigher not in zero range
138	RPC_ARITMIC_OVERFLOW	Aritmic overflow
139	RPC_ADC_OVERFLOW	Overload by ADC conversion
140	RPC_ADC_UNDERFLOW	Underload by ADC conversion
141	RPC_GAIN_NEGATIVE	Weight should increase and not decrease
142	RPC_GAIN_OVERFLOW	Weight to low, value between zero and end weight required
143	RPC_ACCESSDENIED	Command executing denied first enter TAC or CAL code

3 Penko Profinet devices

In the Penko product catalog there are two types of devices that support Profinet. Those devices are Profinet IO devices and not Profinet controllers. The device specific information is listed in this chapter. The two Profinet devices are SGM760/860 and Penko 1020 Profinet.

3.1 SGM760/860

The SGM760/860 are the Profinet version of the SGM series. The Profinet implementation is identical in the SGM760 and SGM860. In Figure 2 and Figure 3 the SGM devices are shown.

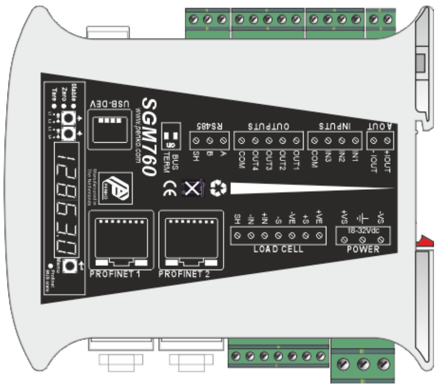


Figure 2 SGM760

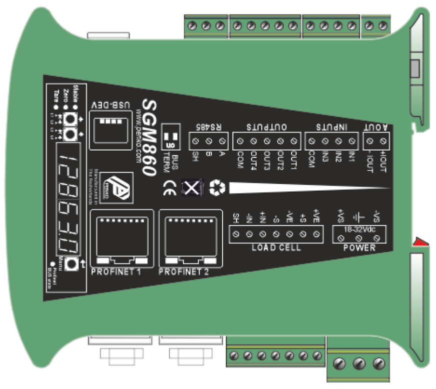


Figure 3 SGM860

In Figure 4 below, the different Profinet status LED’s are described. The DCP_signal LED’s are also used for the weighing part of the device, see the SGM manual. The three LED’s combined are the DCP_signal, the blink frequency is 1 Hz. The Profinet error LED states are described in Table 9.

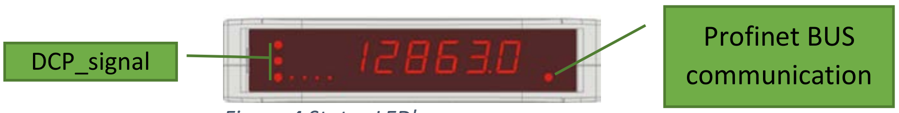


Figure 4 Status LED's

Table 9 SGM Profinet BUS communication state

State BUS FAIL	Description
ON	No link status available
FLASHING	Link status OK; no communication link to a PROFINET-controller.
OFF	The PROFINET-controller has an active communication link to this PROFINET-device.

3.2 Penko 1020 Profinet

The Penko 1020 Profinet is one of the devices in the 1020 series. In Figure 5 the rear view of the device is shown. Right next to the Profinet ethernet ports are two Profinet state LEDs. The different error states for each LED are described in Table 10 and Table 11.

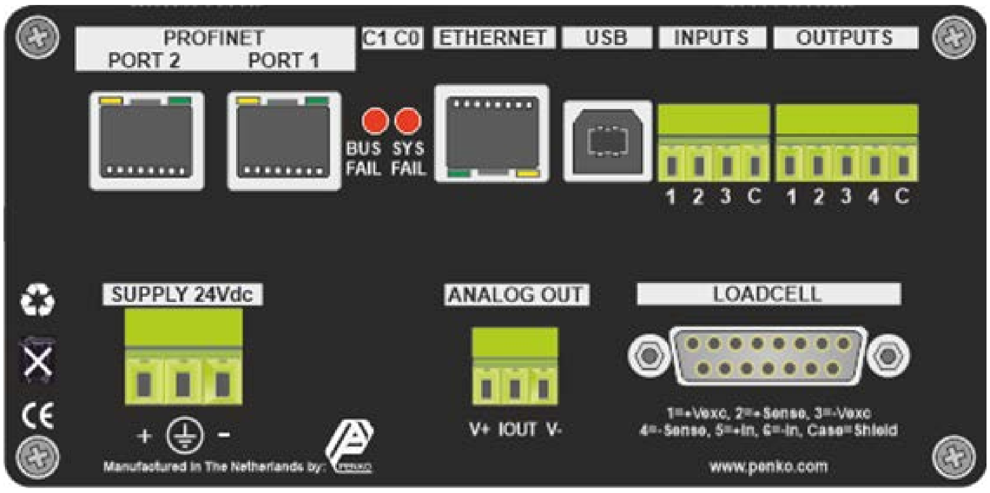


Figure 5 1020 Profinet rear view

Table 10 1020 Profinet BUS communication state

State BUS FAIL	Description
ON	No link status available.
FLASHING	Link status OK; no communication link to a PROFINET-controller.
OFF	The PROFINET-controller has an active communication link to this PROFINET-device.

Table 11 1020 Profinet SYS communication state

State SYS FAIL	Description
ON	PROFINET diagnostic exists.
OFF	No PROFINET diagnostic.

4 Installation & configuration

Install the required software and configure the Penko Profinet IO device module. This manual covers the installation and configuration for the CODESYS development environment. The example projects can be downloaded from the Penko website. The example project is made for the SGM device but can also be used for other Penko devices that supports Profinet.

4.1 CODESYS

4.1.1 CODESYS development environment

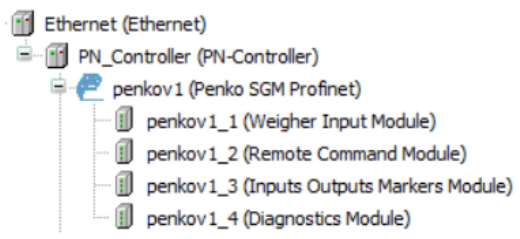
Download and install the newest version from the tool “CODESYS Development System”. The download can be found at store.codesys.com.

When writing this manual, the newest CODESYS version is 3.5.15.20. In this version the required WinPcap driver is missing. Because WinPcap no longer receives updates it is recommended to install the alternative Nmap driver from www.nmap.org/download.html.

4.1.2 Add Penko Profinet IO module

First install the GSDML file in CODESYS. As described in chapter 1, the GSDML file is available for downloading at the Penko website. In CODESYS in menu “Tools “-> “Device Repository” is the option to install the GSDML file.

Connect the Penko Profinet device to the PN-Controller (PLC). Right click on the controller and select “Scan for devices”. The connected Penko Profinet device(s) should appear. Add the devices to the project. In Figure 6 one Penko Profinet device (SGM860) has been added to the project and named as “penkov1”.



```
Ethernet (Ethernet)
  PN_Controller (PN-Controller)
    penkov1 (Penko SGM Profinet)
      penkov1_1 (Weigher Input Module)
      penkov1_2 (Remote Command Module)
      penkov1_3 (Inputs Outputs Markers Module)
      penkov1_4 (Diagnostics Module)
```

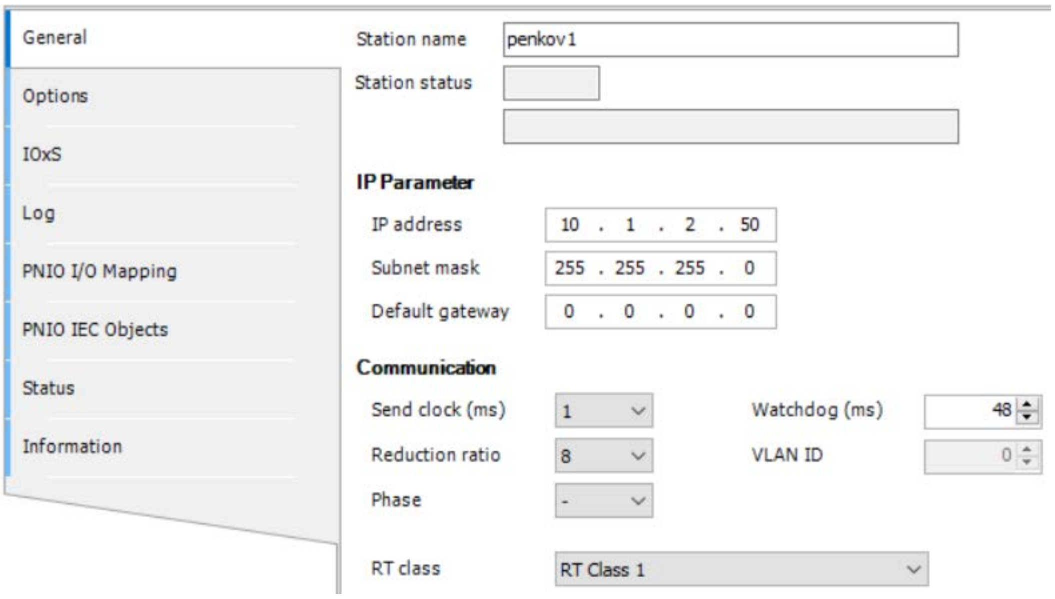
Figure 6 Penko Profinet module added

Ethernet driver
PLC device with PN-controller
Penko SGM Profinet module
Submodule 1 Weigher input Module
Submodule 2 Remote Command Module
Submodule 3 Inputs Outputs Markers Module
Submodule 4 diagnostics Module

4.1.3 Configure Penko Profinet IO module

Configure the Penko Profinet IO module so it can communicate with the PN-Controller. After adding the device in paragraph 4.1.2 go to general settings from the module. Adjust the IP parameters to match the PLC IP range parameters. In this case the IP range is set from 10.1.2.50 to 10.1.1.254. In Figure 7 below the IP address of the connected Penko device is 10.1.2.50 with a subnet mask 255.255.255.0.

For the example program it is not necessary change other settings.



Station name: penkov1
Station status:

IP Parameter
IP address: 10 . 1 . 2 . 50
Subnet mask: 255 . 255 . 255 . 0
Default gateway: 0 . 0 . 0 . 0

Communication
Send clock (ms): 1
Watchdog (ms): 48
Reduction ratio: 8
VLAN ID: 0
Phase: -
RT class: RT Class 1

Figure 7 Penko Profinet module general settings

4.1.4 Submodule parameters

In CODESYS the parameters can be found at the "General" tab of the submodule. See as example the parameters from the weigher input module in Figure 8 and Figure 9 below.

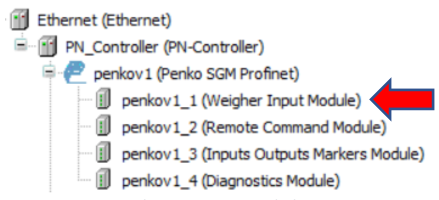
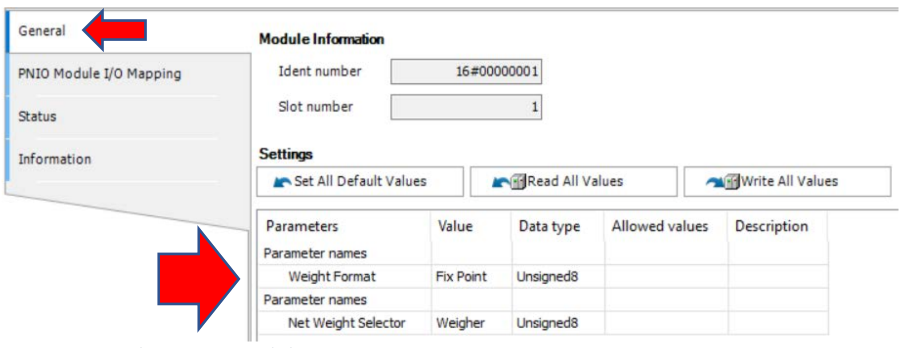

Ethernet (Ethernet)
PN_Controller (PN-Controller)
penkov1 (Penko SGM Profinet)
penkov1_1 (Weigher Input Module)
penkov1_2 (Remote Command Module)
penkov1_3 (Inputs Outputs Markers Module)
penkov1_4 (Diagnostics Module)

Figure 8 Weigher input module



Module Information
Ident number: 16#00000001
Slot number: 1

Settings

Parameters	Value	Data type	Allowed values	Description
Parameter names				
Weight Format	Fix Point	Unsigned8		
Parameter names				
Net Weight Selector	Weigher	Unsigned8		

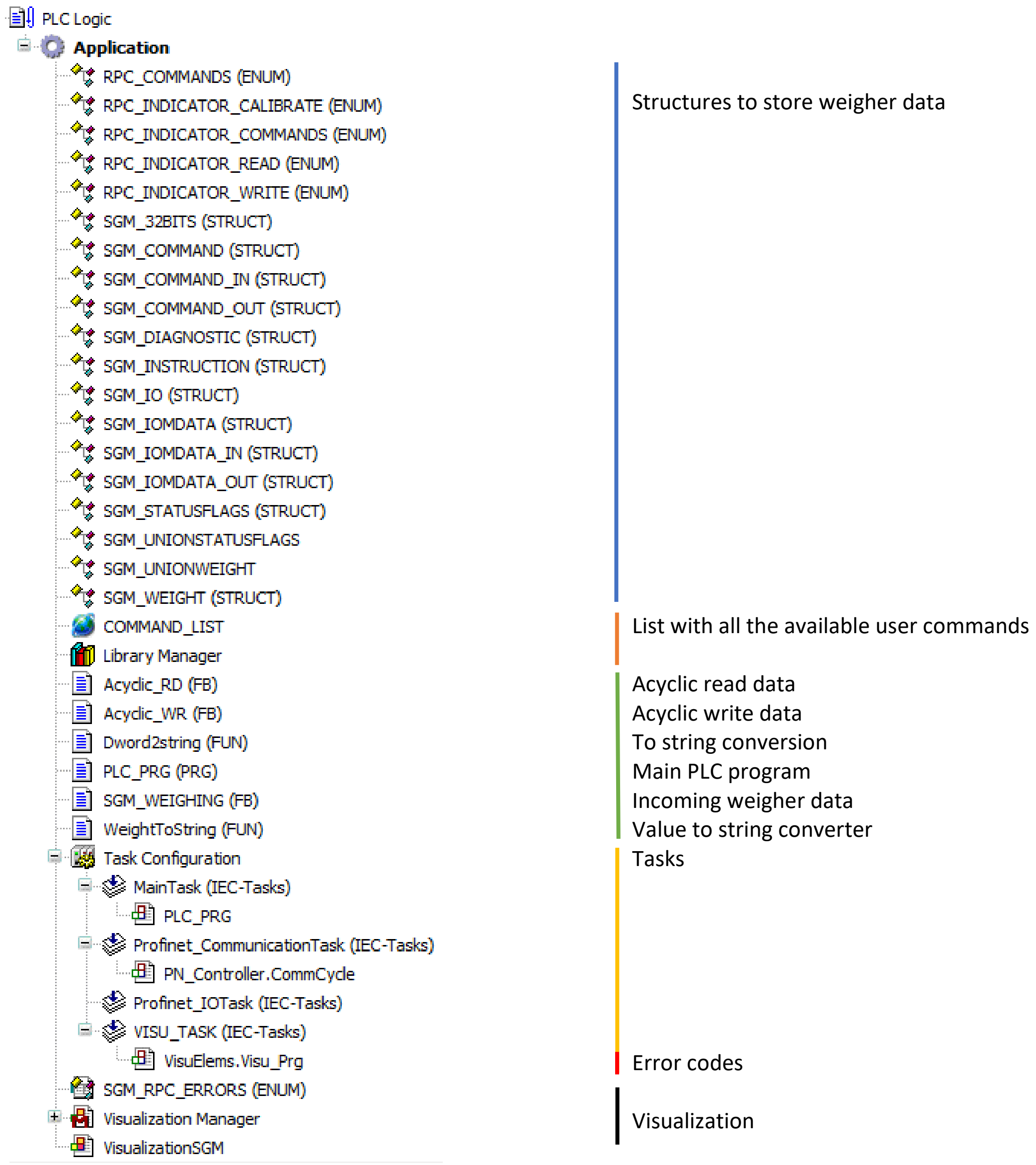
Figure 9 Weigher input module parameters

5 Example 1: Penko Profinet IO module CODESYS

This is an example how to use the Penko Profinet IO module with one Penko SGM device. This example also works at other Penko devices that support Profinet.

5.1 Example application structure

The main file is the PLC_PRG. This file handles the weighing data IO, visualization and user inputs. In SGM_WEIGHING the received data from the Penko Device is processed.



```
PLC Logic
  Application
    RPC_COMMANDS (ENUM)
    RPC_INDICATOR_CALIBRATE (ENUM)
    RPC_INDICATOR_COMMANDS (ENUM)
    RPC_INDICATOR_READ (ENUM)
    RPC_INDICATOR_WRITE (ENUM)
    SGM_32BITS (STRUCT)
    SGM_COMMAND (STRUCT)
    SGM_COMMAND_IN (STRUCT)
    SGM_COMMAND_OUT (STRUCT)
    SGM_DIAGNOSTIC (STRUCT)
    SGM_INSTRUCTION (STRUCT)
    SGM_IO (STRUCT)
    SGM_IOMDATA (STRUCT)
    SGM_IOMDATA_IN (STRUCT)
    SGM_IOMDATA_OUT (STRUCT)
    SGM_STATUSFLAGS (STRUCT)
    SGM_UNIONSTATUSFLAGS
    SGM_UNIONWEIGHT
    SGM_WEIGHT (STRUCT)
    COMMAND_LIST
    Library Manager
    Acyclic_RD (FB)
    Acyclic_WR (FB)
    Dword2string (FUN)
    PLC_PRG (PRG)
    SGM_WEIGHING (FB)
    WeightToString (FUN)
    Task Configuration
      MainTask (IEC-Tasks)
        PLC_PRG
      Profinet_CommunicationTask (IEC-Tasks)
        PN_Controller.CommCycle
      Profinet_IOTask (IEC-Tasks)
      VISU_TASK (IEC-Tasks)
        VisuElems.Visu_Prg
    SGM_RPC_ERRORS (ENUM)
    Visualization Manager
    VisualizationSGM
```

Structures to store weigher data

List with all the available user commands

Acyclic read data
Acyclic write data
To string conversion
Main PLC program
Incoming weigher data
Value to string converter

Tasks

Error codes

Visualization

Figure 10 PLC Penko Profinet CODESYS example application

5.2 Description example code

In this example there is one SGM device initialized, called sgm1. The variable sgm1 is assigned to the struct SGM_IO. The SGM_IO struct contains multiple structs and variables. The flowchart in Figure 11 below is showing the different nested structs. Variables inside the structs are not shown in this flowchart. The term IOM is Input Output Markers.

“In” means send data from PN-Controller to Profinet device. Use this to send data to the Profinet device.
“Out” means receive data from Penko Profinet device. Use this to read data from the Profinet device.

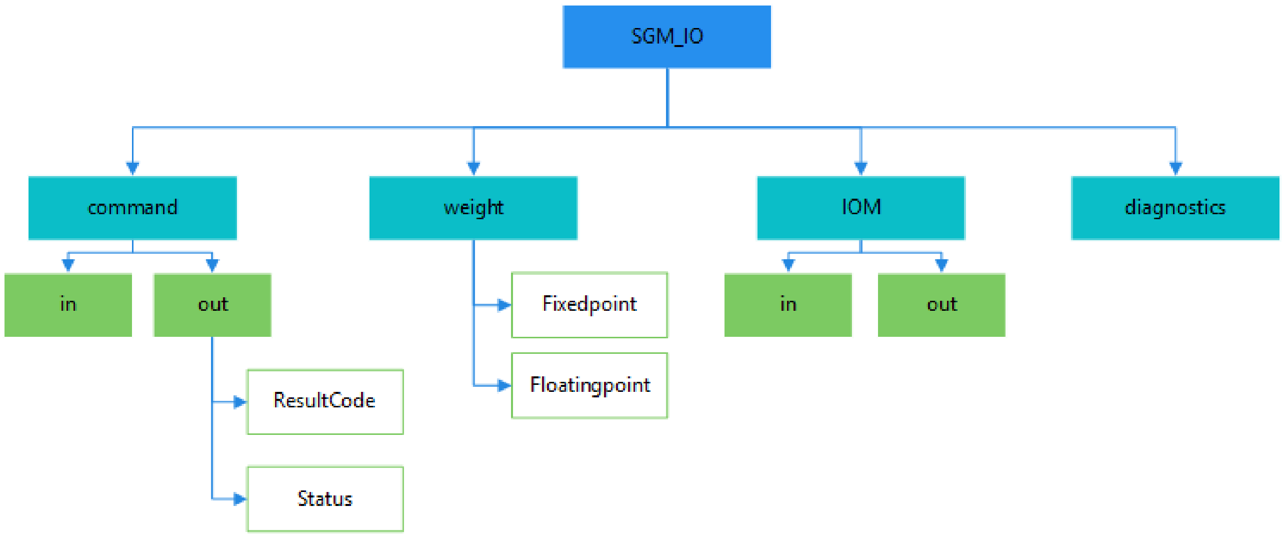


Figure 11 Flowchart SGM_IO struct

First declare sgm1 with the SGM_IO struct assigned to it. $sgm1 : SGM\_IO$;
To access a variable in the sgm1 use: $testvar := sgm1.command.Out.Status.Stable$
In this example the testvar variable contains the stable weight boolean from the status struct, received from the Penko device.

5.2.1 Check valid connection to device

When executing the weigher block, the first step is to check if there is an active connection with the weighing device. Do this by verifying that the data is valid, the application is running and the Profinet module isn't in error mode.

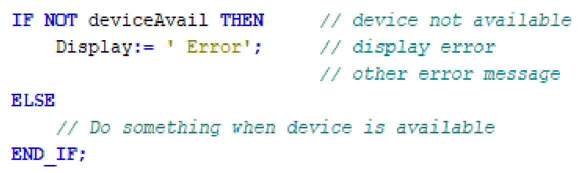
Get valid data bool:

$$DataValid := (statusIOPS\ AND\ GOOD) = GOOD\ AND\ (statusIOfloat\ AND\ GOOD) = GOOD\ AND\ (statusIOdiagnostic\ AND\ GOOD) = GOOD;$$

Then check if device is available:

$$Weighing.deviceAvail := Penko\_Profinet\_Module\_V1\_0.xRunning\ AND\ NOT\ Penko\_Profinet\_Module\_V1\_0.xError\ AND\ DataValid;$$

In the SGM_WEIGHER code, use this to check if the device is available:

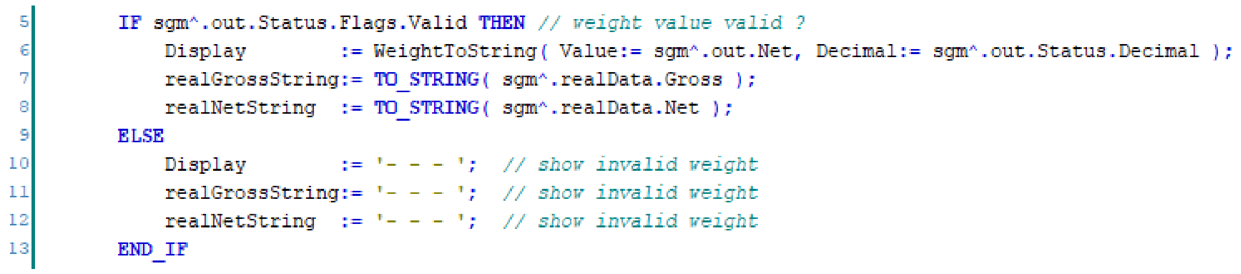


```
IF NOT deviceAvail THEN      // device not available
    Display:= ' Error';      // display error
                             // other error message
ELSE
    // Do something when device is available
END_IF;
```

Figure 12 Check if device is available

5.2.2 Using valid weight data

To make sure the application is always using valid weight data, use the "Weight is valid" channel from the Penko device input. This can be found in the file: "SGM_WEIGHING (FB)".



```
    IF sgm^.out.Status.Flags.Valid THEN // weight value valid ?
        Display         := WeightToString( Value:= sgm^.out.Net, Decimal:= sgm^.out.Status.Decimal );
        realGrossString:= TO_STRING( sgm^.realData.Gross );
        realNetString   := TO_STRING( sgm^.realData.Net );
    ELSE
        Display         := '- - - ';  // show invalid weight
        realGrossString:= '- - - ';  // show invalid weight
        realNetString   := '- - - ';  // show invalid weight
    END_IF
```

Figure 13 Using valid weight data

5.2.3 User input

In this example there are two types of user input. This is to show how to connect an external input to the Penko Profinet implementation. The button and/or drop-down list can be replaced by another input from the PLC application.

1. Buttons
2. Drop-down list

The commands are executed in the SGM_WEIGHING code and set in the PLC_RPG code.

Buttons

Figure 14 below shows the dataflow of the button command. The file names below the buttons indicates in which file the code is executed.

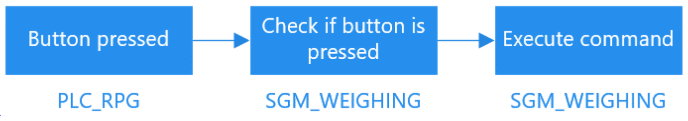
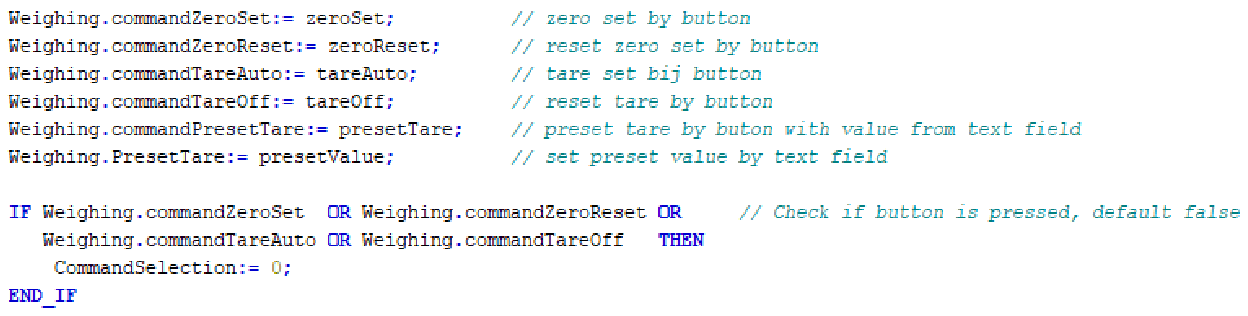


Figure 14 Flowchart button commands

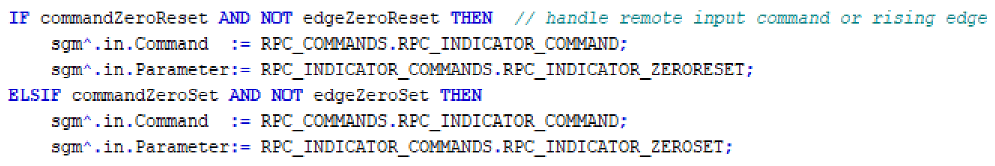
With the code in Figure 15 below it is possible to connect a button to a command. The values assigned to the command are Booleans. In the SGM_WEIGHING code there is a check if a Boolean is set to true. If true, the command is executed. Figure 16 below shows the two commands.



```
Weighing.commandZeroSet:= zeroSet;          // zero set by button
Weighing.commandZeroReset:= zeroReset;      // reset zero set by button
Weighing.commandTareAuto:= tareAuto;        // tare set bij button
Weighing.commandTareOff:= tareOff;          // reset tare by button
Weighing.commandPresetTare:= presetTare;    // preset tare by buton with value from text field
Weighing.PresetTare:= presetValue;          // set preset value by text field

IF Weighing.commandZeroSet  OR Weighing.commandZeroReset OR     // Check if button is pressed, default false
   Weighing.commandTareAuto OR Weighing.commandTareOff   THEN
    CommandSelection:= 0;
END_IF
```

Figure 15 Connect buttons to commands



```
IF commandZeroReset AND NOT edgeZeroReset THEN  // handle remote input command or rising edge
    sgm^.in.Command  := RPC_COMMANDS.RPC_INDICATOR_COMMAND;
    sgm^.in.Parameter:= RPC_INDICATOR_COMMANDS.RPC_INDICATOR_ZERORESET;
ELSIF commandZeroSet AND NOT edgeZeroSet THEN
    sgm^.in.Command  := RPC_COMMANDS.RPC_INDICATOR_COMMAND;
    sgm^.in.Parameter:= RPC_INDICATOR_COMMANDS.RPC_INDICATOR_ZEROSET;
```

Figure 16 Execute button commands

Drop-down list

The drop-down list in this example contains all available Profinet commands. Figure 17 below shows the dataflow of the drop-down list. The file names below the buttons indicate in which file the code is executed.

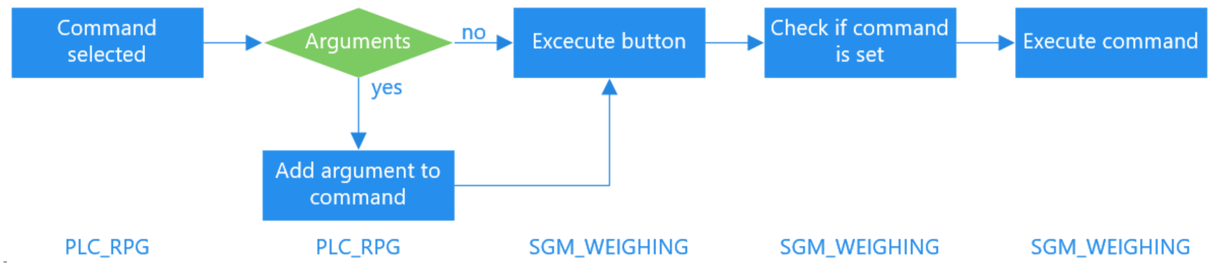
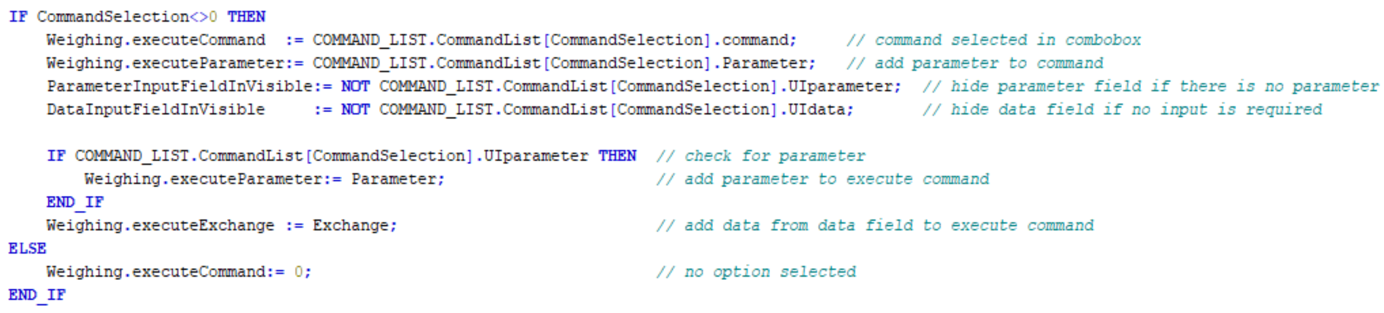


Figure 17 Flowchart drop-down list commands

The CommandSelection parameter is connected to the drop-down list in the visualization. The code in Figure 18 below detects if there is an option selected from the dropdown list. When required the data and/or parameter field are visible.

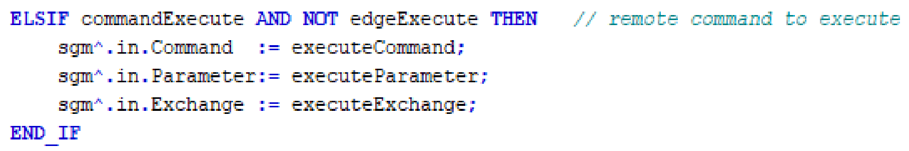


```
IF CommandSelection<>0 THEN
    Weighing.executeCommand  := COMMAND_LIST.CommandList[CommandSelection].command;     // command selected in combobox
    Weighing.executeParameter:= COMMAND_LIST.CommandList[CommandSelection].Parameter;   // add parameter to command
    ParameterInputFieldInVisible:= NOT COMMAND_LIST.CommandList[CommandSelection].UIparameter;  // hide parameter field if there is no parameter
    DataInputFieldInVisible     := NOT COMMAND_LIST.CommandList[CommandSelection].UIdata;       // hide data field if no input is required

    IF COMMAND_LIST.CommandList[CommandSelection].UIparameter THEN  // check for parameter
        Weighing.executeParameter:= Parameter;                      // add parameter to execute command
    END_IF
    Weighing.executeExchange := Exchange;                           // add data from data field to execute command
ELSE
    Weighing.executeCommand:= 0;                                    // no option selected
END_IF
```

Figure 18 Detect selection from drop-down list

The commandExecute parameter is connected to the execute button in the visualization. Until this button isn't pressed the command won't be executed. In Figure 19 below the command is in an elsif. This is because the command is listed below the execute button command. When the drop-down list is the only command to be executed change the elsif to if.

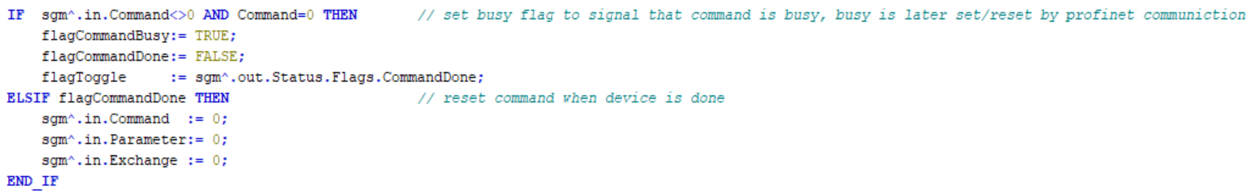


```
ELSIF commandExecute AND NOT edgeExecute THEN   // remote command to execute
    sgm^.in.Command  := executeCommand;
    sgm^.in.Parameter:= executeParameter;
    sgm^.in.Exchange := executeExchange;
END_IF
```

Figure 19 Execute selected command

5.2.4 Check if command is busy

It is possible that the device is still busy with the execution of a command. In this case the new command isn’t executed by the device and the busy indicator in the visualization turns orange. In the example flags are used to detect the device (busy) state. In Figure 20 below the code shows how to detect if the device is busy with executing a (previous) command.



```
IF  sgm^.in.Command<>0 AND Command=0 THEN       // set busy flag to signal that command is busy, busy is later set/reset by profinet communiction
    flagCommandBusy:= TRUE;
    flagCommandDone:= FALSE;
    flagToggle     := sgm^.out.Status.Flags.CommandDone;
ELSIF flagCommandDone THEN                      // reset command when device is done
    sgm^.in.Command  := 0;
    sgm^.in.Parameter:= 0;
    sgm^.in.Exchange := 0;
END_IF
```

Figure 20 Detect device is busy with a command

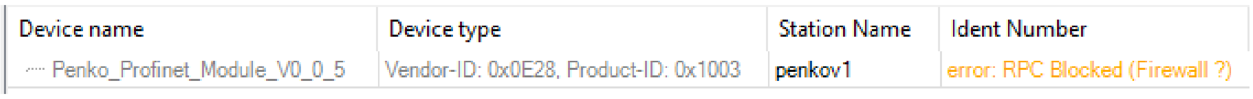
6 Problem solving

General problem-solving info while configuring the development environment.

6.1 CODESYS

After adding the Penko Profinet module it is possible that there is an error. In this paragraph some common errors are described and how to solve them.

6.1.1 Error: RPC Blocked (Firewall?)



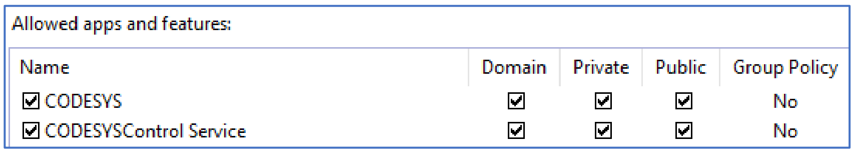
Device name	Device type	Station Name	Ident Number
Penko_Profinet_Module_V0_0_5	Vendor-ID: 0x0E28, Product-ID: 0x1003	penkov1	error: RPC Blocked (Firewall ?)

Figure 21 Error RPC blocked (firewall?)

The error in Figure 21 indicates that the CODESYS application isn't configured properly in the Windows Defender firewall settings. To change this, you need administrator rights.

Go to the control panel -> Windows Defender Firewall -> Allow an app or feature

Enable the options to change the settings. Scroll down in the application list to CODESYS. If you don't see this application add it to the list. Make sure that all the checkboxes are checked (Domain, Private and public). The restart CODESYS to apply the changes.

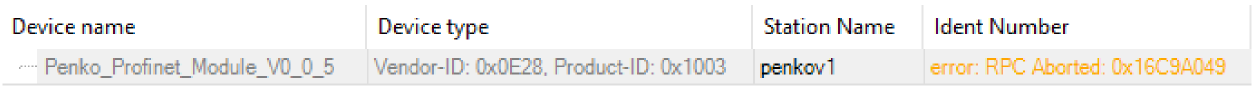


Allowed apps and features:

Name	Domain	Private	Public	Group Policy
☑ CODESYS	☑	☑	☑	No
☑ CODESYSControl Service	☑	☑	☑	No

Figure 22 Windows Firewall settings

6.1.2 Error RPC Aborted 0x16C9A049

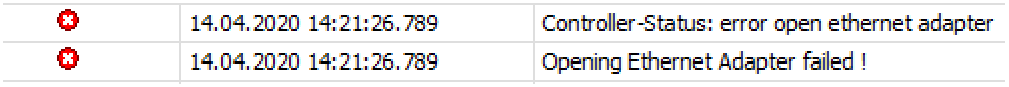


Device name	Device type	Station Name	Ident Number
Penko_Profinet_Module_V0_0_5	Vendor-ID: 0x0E28, Product-ID: 0x1003	penkov1	error: RPC Aborted: 0x16C9A049

Figure 23 Error RPC aborted

Turn OFF and ON the power supply to solve this problem.

6.1.3 Opening ethernet adapter failed



✖	14.04.2020 14:21:26.789	Controller-Status: error open ethernet adapter
✖	14.04.2020 14:21:26.789	Opening Ethernet Adapter failed !

In this case the adapter configuration could be wrong or there is something with the Nmap driver. Try another ethernet port and/of try reinstalling the Nmap driver as described in paragraph 4.1.1.

List of figures

List of tables

Appendix

List with all the appendix.

Appendix I. Module config parameters

For every program there are different configuration parameters available. Not all programs are using all the 30 config fields, but it is possible to read and write all config parameters. The indicator (IND) program isn't using any of the config parameters. All the parameters with the unit s (time in seconds) are using the format 0.00s.

Net weight selector

Display indicator selection

Value	Description
0	DISPLAY OIML NET (multiple range/multi interval, approved readout)
1	FAST GROSS (fixed step size)
2	FAST NET (fixed step size)
3	DISPLAY GROSS (multiple range/multi interval
4	DISPLAY NET (multiple range/ multi interval)
5	TARE
6	PEAK
7	VALLEY
8	HOLD
9	WEIGHER 10X
10	FASTGROSS 10X
11	FASTNET 10X
12	GROSS 10X
13	NET 10X
14	TARE 10X
15	PEAK 10X
16	VALLEY 10X
17	HOLD 10X
18	SIGNAL in mV

Program: Check weigher (CHK)

Config	Field name	Allowed values	Unit	Description
Config 1	Mode	0 1		0 = Static 1 = Dynamic
Config 2	Stability			0 = Off 1 = Stable 2 = H-Time 3 = H-Time+Stable 4 = H-Time/Stable 5 = Stable+H-Time
Config 3	H-Time		s	
Config 4	Display Hold		s	
Config 5	Reject Mode	0 1		0 = Time 1 = Photocell
Config 6	Fixed Speed	0 1		0 = No 1 = Yes
Config 7	Min Speed		%	Value between 0.00 and 100.00

Config 8	Max Speed		%	Value between 0.00 and 100.00
Config 9	Recipe	0 1		0 = Local 1 = remote
Config 10	Online Ticket	0 1		0 = No 1 = Yes
Config 11	Use Alibi Memory	0 1		0 = No 1 = Yes

Program: Monofil (MFL)

Config	Field name	Allowed values	Unit	Description
Config 1	Dosing	0 1		0 = In 1 = Out
Config 2	Weighing	0 1		0 = Net 1 = Gross
Config 3	Stability	0 1 2 3 4		0 = Stable 1 = H-Time+Stable 2 = H-Time 3 = H-Time/Stable 4 = Stable+H-Time
Config 4	H-Time		s	
Config 5	K.E.B. Time		s	
Config 6	Inflight		%	
Config 7	Max Inflight Corr		kg	
Config 8	Turnover Correlation		%	
Config 9	Fine Time		s	
Config 10	Tolerance	0 1		0 = No 1 = Yes
Config 11	Tolerance Interval		s	
Config 12	Display Hold		s	
Config 13	Release Valve	0 1		0 = No 1 = Yes
Config 14	Empty Level		kg	
Config 15	Empty Time		s	
Config 16	Recipe	0 1		0 = Local 1 = Remote
Config 17	Online Ticket	0 1		0 = No 1 = Yes
Config 18	Use Alibi Memory	0 1		0 = No 1 = Yes
Config 19	Coarse Delay		s	
Config 20	Fine Delay		s	
Config 24	Start Delay		s	
Config 25	Start Level	0 1		0 = No 1 = Yes
Config 26	Auto Start	0 1		0 = No 1 = Yes

Program Belt weigher (BLT)

Config	Field name	Allowed values	Unit	Description
Config 1	Flow Point Pos.	0 1 2 3 4 5		0 = 000000 1 = 00000.0 2 = 0000.00 3 = 000.000 4 = 00.0000 5 = 0.00000
Config 2	Totals Point Pos.	0 1 2 3 4 5		0 = 000000 1 = 00000.0 2 = 0000.00 3 = 000.000 4 = 00.0000 5 = 0.00000
Config 3	Max Flow		kg/h	
Config 4	Dynamic Tare Band		%	
Config 5	Dynamic Tare Time		s	
Config 6	Zero Suppress		kg/h	
Config 7	Filter Time		s	
Config 8	Weight per Pulse		kg	
Config 9	Correction		*	
Config 10	Pulses per Meter			
Config 11	Measurement Method	0 1		0 = Beltweigher 1 = Impact Flow Meter
Config 12	Analogue Use	0 1		0 = Flow Measurement 1 = Flow Regulation
Config 13	Control Correction		$	

Appendix II. CODESYS Penko example code

POU: PLC_PRG variables

```
PROGRAM PLC_PRG
VAR CONSTANT
GOOD : BYTE := 128 ;
END_VAR
VAR
Weighing : SGM_WEIGHING ;
MyAcyclic : Acyclic_RD ;
MyAcyclicWr : Acyclic_Wr ;
sgm1 : SGM_IO ; // penkov1 (SGM760)

display : STRING ;
zeroSet : BOOL := FALSE ;
zeroReset : BOOL := FALSE ;
tareAuto : BOOL := FALSE ;
tareOff : BOOL := FALSE ;
presetTare : BOOL := FALSE ;
presetValue : DINT := 0 ;

IOInvalid : BOOL ;
CommandInvalid : BOOL ;
DiagnosticsInvalid : BOOL ;
WeigherInvalid : BOOL ;

frameMaster : DWORD := 0 ;
frameSlave : DWORD := 0 ;
onces : BOOL := FALSE ;
commandExecute : BOOL := FALSE ;

CommandSelection : WORD := 0 ;
EdgeCommandSelection : WORD ;
Parameter : WORD := 0 ;
Exchange : DINT := 0 ;
ResultData : String ;
ParameterInputFieldInVisible : BOOL ;
DataInputFieldInVisible : BOOL ;
ResultFieldInvisible : BOOL ;
Interval : DINT := 0 ;
END_VAR

VAR CONSTANT
LOCKED : BOOL := FALSE ;
END_VAR

```

POU: PLC_PRG code

```
IF onces THEN // dummy statements to force updates of Profinet variables, Codesys bug?
onces := FALSE ;
frameMaster := frameSlave := 0 ;
END_IF

Weighing . sgm := ADR ( sgm1 ) ; // set base ponter of an profibus device
Weighing . deviceAvail := penkov1 . xRunning AND NOT penkov1 . xError AND
( Weighing . sgm ^ . weight . isValid AND GOOD ) = GOOD ; // detect communication errors or invalid data

Weighing . commandZeroSet := zeroSet ; // commands set by buttons
Weighing . commandZeroReset := zeroReset ;
Weighing . commandTareAuto := tareAuto ;
Weighing . commandTareOff := tareOff ;
Weighing . commandPresetTare := presetTare ;
Weighing . PresetTare := presetValue ; // set by text field

IF Weighing . commandZeroSet OR Weighing . commandZeroReset OR
Weighing . commandTareAuto OR Weighing . commandTareOff THEN
CommandSelection := 0 ;
END_IF ;

IOInvalid := NOT penkov1 . xRunning OR penkov1 . xError OR sgm1 . iom . InIsValid <> GOOD OR sgm1 . iom . OutIsValid <> GOOD ;
CommandInvalid := NOT penkov1 . xRunning OR penkov1 . xError OR sgm1 . command . InIsValid <> GOOD OR sgm1 . command . OutIsValid <> GOOD ;
DiagnosticsInvalid := NOT penkov1 . xRunning OR penkov1 . xError OR sgm1 . diagnostic . IsValid <> GOOD ;
WeigherInvalid := NOT penkov1 . xRunning OR penkov1 . xError OR sgm1 . weight . isValid <> GOOD ;

IF CommandSelection <> 0 THEN
Weighing . executeCommand := COMMAND_LIST . CommandList [ CommandSelection ] . command ; // set by combobox
Weighing . executeParameter := COMMAND_LIST . CommandList [ CommandSelection ] . Parameter ;
ParameterInputFieldInVisible := NOT COMMAND_LIST . CommandList [ CommandSelection ] . UIparameter ;
DataInputFieldInVisible := NOT COMMAND_LIST . CommandList [ CommandSelection ] . UIdata ;

IF COMMAND_LIST . CommandList [ CommandSelection ] . UIparameter THEN // Use UI parameter ?
Weighing . executeParameter := Parameter ;
END_IF
Weighing . executeExchange := Exchange ;
ELSE
Weighing . executeCommand := 0 ;
END_IF
Weighing . commandExecute := commandExecute ;
ResultFieldInvisible := NOT PLC_PRG . Weighing . flagcommandDone ;

Weighing . resetCommand := CommandSelection <> EdgeCommandSelection ;
EdgeCommandSelection := CommandSelection ;

// Interval:= Interval+1;
// MyAcyclic.Request:= Interval=1;
// MyAcyclic();
//
// IF MyAcyclic.valid THEN
// MyAcyclicWr.buffer[0]:= MyAcyclic.buffer[0] +1;
// MyAcyclicWr.buffer[1]:= MyAcyclic.buffer[1] +1;
// MyAcyclicWr.buffer[2]:= MyAcyclic.buffer[2] +1;
// MyAcyclicWr.Request:= TRUE;
// ELSE
// MyAcyclicWr.Request:= FALSE;
// END_IF
// MyAcyclicWr();
//
// //IF MyAcyclicWr.valid OR Interval>50 THEN
// IF Interval>50 THEN
// Interval:= 0;
// END_IF

Weighing ( ) ; // execute Penko weighing block

display := weighing . Display ; // copy display data
IF COMMAND_LIST . CommandList [ CommandSelection ] . UIChar THEN
ResultData := DWord2String ( TO_DWORD ( Weighing . executeResultData ) ) ;
ELSE
ResultData := TO_STRING ( Weighing . executeResultData ) ;
END_IF

```

POU: SGM_WEIGHING variables

```
FUNCTION_BLOCK SGM_WEIGHING
VAR_INPUT
sgm : POINTER TO SGM_IO ;
deviceAvail : BOOL := FALSE ;

PresetTare : DINT := 0 ;

commandTareAuto : BOOL := FALSE ;
commandTareOff : BOOL := FALSE ;
commandPresetTare : BOOL := FALSE ;
commandZeroSet : BOOL := FALSE ;
commandZeroReset : BOOL := FALSE ;
commandExecute : BOOL := FALSE ;

executeCommand : WORD ;
executeParameter : WORD ;
executeExchange : DINT ;
executeResultData : DINT ;

resetCommand : Bool ;
END_VAR
VAR_OUTPUT
flagNet : BOOL ;
flagValid : BOOL ;
flagStable : BOOL ;
flagZeroSet : BOOL ;
flagZeroCenter : BOOL ;
flagCommandBusy : BOOL ;
flagCommandDone : BOOL ;

Decimal : BYTE ;
Range : BYTE ;
RangeInValid : BOOL ;
Display : STRING ;
executeResult : STRING ;
END_VAR
VAR
Command : DWORD ;
flagToggle : BOOL ;

edgeTareAuto : BOOL := FALSE ;
edgeTareOff : BOOL := FALSE ;
edgePresetTare : BOOL := FALSE ;
edgeZeroSet : BOOL := FALSE ;
edgeZeroReset : BOOL := FALSE ;
edgeExecute : BOOL := FALSE ;
END_VAR

```

POU: SGM_WEIGHING code

```
IF NOT deviceAvail THEN // device avail ?
Display := ' Error' ;
ELSE
IF sgm ^ . weight . Flags . Valid THEN // weight value valid ?
Display := WeightToString ( Weight := sgm ^ . weight . display , Decimal := sgm ^ . weight . Decimal , Floatingpoint :=
sgm ^ . weight . Flags . Floatingpoint ) ;
ELSE
Display := '- - - ' ; // show invalid weight
END_IF

flagValid := sgm ^ . weight . Flags . Valid ; // copy status flags
flagNet := sgm ^ . weight . Flags . Net ;
flagStable := sgm ^ . weight . Flags . Stable ;
flagZeroSet := sgm ^ . weight . Flags . ZeroSet ;
flagZeroCenter := sgm ^ . weight . Flags . ZeroCenter ;
flagCommandBusy := sgm ^ . weight . Flags . CommandBusy ;
IF Resetcommand THEN
flagToggle := sgm ^ . weight . Flags . CommandDone ;
END_IF
flagCommandDone := sgm ^ . weight . Flags . CommandDone XOR flagToggle ;

executeResultData := sgm ^ . command . Out . ResultData ;
executeResult := TO_STRING ( sgm ^ . command . Out . ResultCode ) ;
Decimal := sgm ^ . weight . Decimal ; // get decimal point position
Range := sgm ^ . weight . Range ; // get multiple range of multi interval
RangeInValid := Range = 0 ; // is ranging active ?

IF commandZeroReset AND NOT edgeZeroReset THEN // handle remote input command or rising edge
sgm ^ . command . In . Command := RPC_COMMANDS . RPC_INDICATOR_COMMAND ;
sgm ^ . command . In . Parameter := RPC_INDICATOR_COMMANDS . RPC_INDICATOR_ZERORESET ;
ELSIF commandZeroSet AND NOT edgeZeroSet THEN
sgm ^ . command . In . Command := RPC_COMMANDS . RPC_INDICATOR_COMMAND ;
sgm ^ . command . In . Parameter := RPC_INDICATOR_COMMANDS . RPC_INDICATOR_ZEROSET ;
ELSIF commandTareAuto AND NOT edgeTareAuto THEN
sgm ^ . command . In . Command := RPC_COMMANDS . RPC_INDICATOR_COMMAND ;
sgm ^ . command . In . Parameter := RPC_INDICATOR_COMMANDS . RPC_INDICATOR_TAREON ;
ELSIF commandTareOff AND NOT edgeTareOff THEN
sgm ^ . command . In . Command := RPC_COMMANDS . RPC_INDICATOR_COMMAND ;
sgm ^ . command . In . Parameter := RPC_INDICATOR_COMMANDS . RPC_INDICATOR_TAREOFF ;
ELSIF commandPresetTare AND NOT edgePresetTare THEN
sgm ^ . command . In . Command := RPC_COMMANDS . RPC_INDICATOR_COMMAND ;
sgm ^ . command . In . Parameter := RPC_INDICATOR_COMMANDS . RPC_INDICATOR_PRESETTARE ;
sgm ^ . command . In . Exchange := PresetTare ;
ELSIF commandExecute AND NOT edgeExecute THEN // remote command to execute ?
sgm ^ . command . In . Command := executeCommand ;
sgm ^ . command . In . Parameter := executeParameter ;
sgm ^ . command . In . Exchange := executeExchange ;
END_IF

IF sgm ^ . command . In . Command <> 0 AND Command = 0 THEN // set busy flag to signal that command is busy, busy is
later set/reset by profinet communiction
flagCommandBusy := TRUE ;
flagCommandDone := FALSE ;
flagToggle := sgm ^ . weight . Flags . CommandDone ;
ELSIF flagCommandDone THEN // reset command when done
sgm ^ . command . In . Command := 0 ;
sgm ^ . command . In . Parameter := 0 ;
sgm ^ . command . In . Exchange := 0 ;
END_IF

Command := sgm ^ . command . In . Command ; // save changes for rising edge detection
edgeTareAuto := commandTareAuto ;
edgeTareOff := commandTareOff ;
edgePresetTare := commandPresetTare ;
```

```
edgeZeroSet := commandZeroSet ;
edgeZeroReset := commandZeroReset ;
edgeExecute := commandExecute ;

END_IF

```

POU: WeightToString variables

```
FUNCTION WeightToString : String
VAR_INPUT
Weight : SGM_UNIONWEIGHT ;
Decimal : INT ;
Floatingpoint : BOOL ;
END_VAR
VAR
Value : DINT ;
Result : STRING ;
ValueStr : String ;
Sign : BOOL ;
Position : INT ;
N : INT ;
Length : INT ;
END_VAR

IF Floatingpoint THEN // no decimalpoint so simple string conversion
Result := REAL_TO_STRING ( Weight . Floatingpoint ) ;
ELSIF Decimal = 0 THEN // no decimalpoint so simple string conversion
Result := DINT_TO_STRING ( Weight . Fixedpoint ) ;
ELSE
Value := Weight . Fixedpoint ;
Sign := Value < 0 ; // save sign and make value positive
IF Sign THEN
Value := - Value ;
END_IF

ValueStr := DINT_TO_STRING ( Value ) ; // convert to string
Result := '' ;

WHILE Len ( ValueStr ) <= Decimal DO // expand string when string is shorter as decimal point
ValueStr := Concat ( '0' , ValueStr ) ;
END_WHILE

Length := Len ( ValueStr ) ;
Position := Length - Decimal ;
FOR N := 1 TO Length DO // insert decimal point
Result := Concat ( Result , Mid ( ValueStr , 1 , N ) ) ;
IF N = Position THEN
Result := Concat ( Result , '.' ) ;
END_IF
END_FOR

IF Sign THEN // add sign symbol
Result := Concat ( '-' , Result ) ;
END_IF
END_IF

WeightToString := Result ; // done
RETURN ;

```

POU: Dword2String variables

```
FUNCTION Dword2string : STRING
VAR_INPUT
Value : DWORD ;
END_VAR
VAR
P : POINTER TO BYTE ;
Str : STRING ;
Ch : BYTE ;

END_VAR

```

POU: Dword2String code

```
Str := '' ;
P := ADR ( Str ) ; // Get string address
WHILE ( Value <> 0 ) DO
Ch := TO_BYTE ( Value AND 255 ) ; // Get char out of dword
Value := SHR ( Value , 8 ) ; // Value>> = 8
P ^ := Ch ; // Assign char to string
P := P + 1 ; // move pointer in string
END_WHILE
P ^ := 0 ; // Add nul terminator
Dword2string := Str ;

```

POU: Acyclic_WR variables

```
FUNCTION_BLOCK Acyclic_WR
VAR_INPUT
Request : BOOL ;
buffer : ARRAY [ 0 .. 31 ] OF BYTE ;
END_VAR
VAR_OUTPUT
valid : BOOL ;
END_VAR
VAR
wrRec : CommFB . WRREC ;
writeError : DWORD ;
writeState : BOOL ;
END_VAR
VAR CONSTANT
WEIGHING_AND_DOSING : WORD := 16#5C00 ;
END_VAR

```

POU: Acyclic_WR code

```
// Write Acyclic Data

IF Request THEN
wrRec . REQ := TRUE ;
valid := FALSE ;
wrRec . ID := penkov1 . GetID ( API := 0 , SLOT := 1 , SUBSLOT := 1 ) ; // get ID of 1. module; // "pnDevice" is a Profinet slave in the device tree
wrRec . INDEX := UINT_TO_INT ( WEIGHING_AND_DOSING ) ;
wrRec . LEN := TO_INT ( SIZEOF ( buffer ) ) ;
wrRec . RECORD := ADR ( buffer ) ;
writeState := TRUE ;
writeError := 0 ;
END_IF

wrRec ( ) ;
IF ( wrRec . DONE AND NOT wrRec . BUSY ) THEN
valid := writeState ;
writeState := FALSE ;
ELSIF ( wrRec . ERROR ) THEN
writeError := wrRec . STATUS ; // read service failed
END_IF
wrRec . REQ := wrRec . BUSY ; // only write once


```

POU: Acyclic_WR variables

```
FUNCTION_BLOCK Acyclic_RD
VAR_INPUT
Request : BOOL ;
END_VAR
VAR_OUTPUT
valid : BOOL ;
buffer : ARRAY [ 0 .. 31 ] OF BYTE ;
END_VAR
VAR
rdrec : CommFB . RDREC ;
readError : DWORD ;
RequestState : BOOL ;
END_VAR
VAR CONSTANT
IM0 : WORD := 16#AFF0 ;
WEIGHING_AND_DOSING : WORD := 16#5C00 ;
END_VAR


```

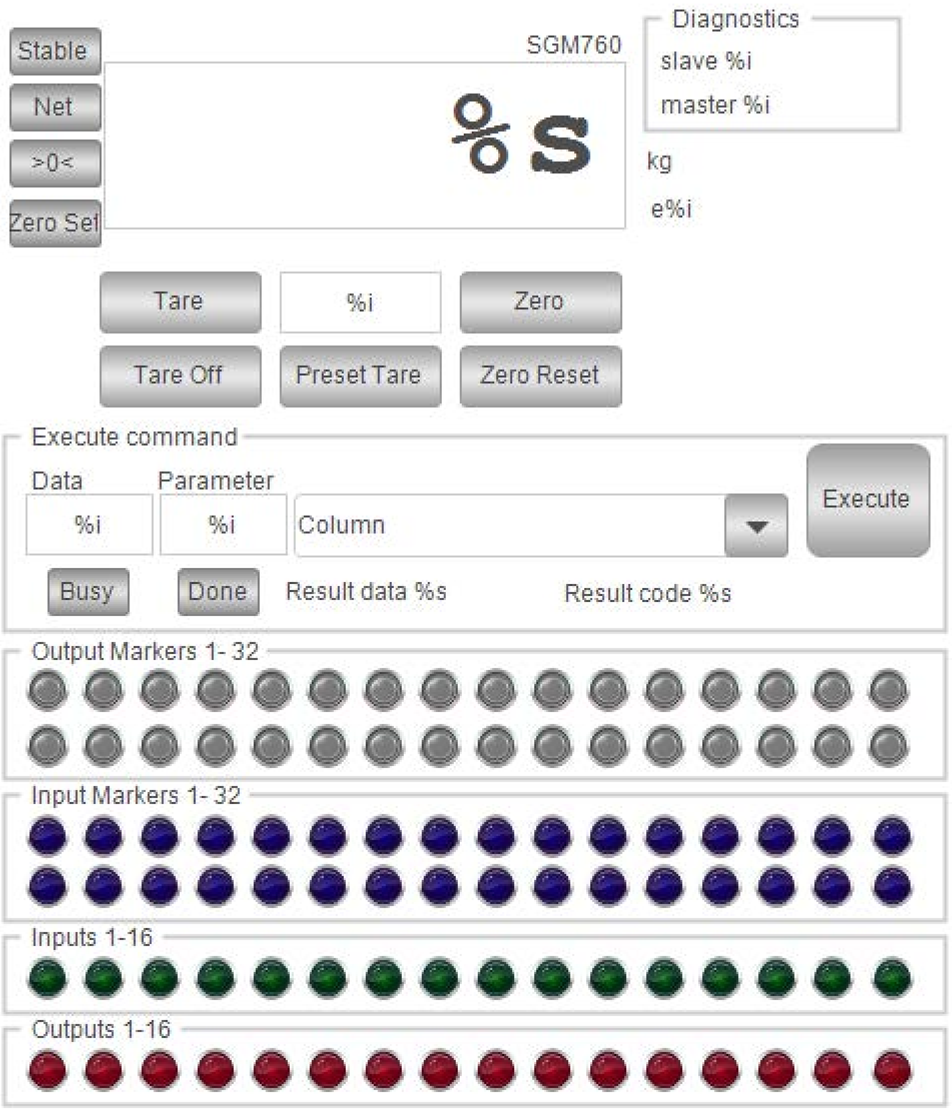
POU: Acyclic_WR code

```
IF Request THEN
rdrec . REQ := TRUE ;
RequestState := TRUE ;
valid := FALSE ;
END_IF

// Read Cyclic
rdrec . ID := penkov1 . GetID ( API := 0 , SLOT := 1 , SUBSLOT := 1 ) ; // get ID of 1. module; // "pnDevice" is a Profinet slave in the device tree
rdrec . INDEX := UINT_TO_INT ( WEIGHING_AND_DOSING ) ;
rdrec . MLEN := TO_INT ( SIZEOF ( buffer ) ) ;
rdrec . RECORD := ADR ( buffer ) ;
rdrec ( ) ;
IF ( rdrec . VALID AND NOT rdrec . BUSY ) THEN
valid := RequestState ;
RequestState := FALSE ;
Request := FALSE ;
ELSIF ( rdrec . ERROR ) THEN
readError := rdrec . STATUS ; // read service failed
END_IF
rdrec . REQ := rdrec . BUSY ; // only read once


```

Visualization



About PENKO

At PENKO Engineering we specialize in weighing. Weighing is inherently chemically correct, independent of consistency, type or temperature of the raw material. This means that weighing any kind of material guaranties consistency and thus, it is essential to sustainable revenue generation in any industry. As a well-established and proven solution provider, we strive for the ultimate satisfaction of custom design and/or standard applications, increasing your efficiencies and saving you time, saving you money.

Whether we are weighing raw materials, components in batching, ingredients for mixing or dosing processes, - or weighing of static containers and silos, or - in-motion weighing of railway wagons or trucks, by whatever means required during a process, we are essentially forming vital linkages between processes and businesses, anywhere at any time. We design, develop and manufacture state of the art technologically advanced systems in accordance with your strategy and vision. From the initial design brief, we take a fresh approach and a holistic view of every project, managing, supporting and/or implementing your system every step of the way. Curious to know how we do it? www.penko.com

Certifications
PENKO sets high standards for its products and product performance which are tested, certified and approved by independent expert and government organizations to ensure they meet – and even – exceed metrology industry guidelines. A library of testing certificates is available for reference on:
www.penko.com/nl/publications_certificates.html

PENKO Professional Services
PENKO is committed to ensuring every system is installed, tested, programmed, commissioned and operational to client specifications. Our engineers, at our weighing center in Ede, Netherlands, as well as our distributors around the world, strive to solve most weighing-system issues within the same day. On a monthly basis PENKO offers free training classes to anyone interested in exploring modern, high-speed weighing instruments and solutions. Training sessions on request: www.penko.com/training

PENKO Alliances
PENKO's worldwide network: Australia, Brazil, China, Denmark, Germany, Egypt, Finland, France, India, Italy, Netherlands, Norway, Poland, Portugal, Slovakia, Spain, Syria, Turkey, United Kingdom, South Africa, Slovakia Sweden and Switzerland, Singapore.
A complete overview you will find on: www.penko.com/distributor