# PENKO Engineering B.V.

Your Partner for Fully Engineered Factory Solutions

Penko EasyWeigher End User manual



### CONTENTS

1. Introduction	3
2. Getting started	3
2.1. Download device descriptions and libraries	3
2.2. Install device descriptions and libraries	4
3. View documentation inside the library	5
4. Set up an example project	5
4.1 Create the project	5
4.2 Example IO mapping	7
4.3 Weigher status struct mapping	8
4.4 Weigher commands (tare, zero, etc)	9
4.5 Reading configuration	11
4.6 Writing configuration – Industrial Mode	13
4.7 Writing configuration – Certified Mode	14
4.8 Calibration	15



### 1. INTRODUCTION

The Penko EasyWeigher solution helps to quickly get started with your Omega. It consists of a CODESYS library for CODESYS V3.5 SP17, and a set of CODESYS device descriptions (\*.devdesc.xml) to talk to the option cards. All files can be downloaded directly from your Omega.

### 2. GETTING STARTED

This chapter will guide you through downloading and installing the necessary files and libraries, and help you setup a very simple starter project. Subsequent chapters will further expand this example with more functionality.

It is expected that you have setup your Omega, and that you can connect to your Omega's web portal from the PC you're working on. See the Omega manual ('7600M1082-<LANGUAGE>-R8 MANUAL OMEGA.pdf') on how to setup your Omega.s

### 2.1. DOWNLOAD DEVICE DESCRIPTIONS AND LIBRARIES

An archive with device descriptions and libraries can be downloaded from two different locations: from the Omega web portal (recommended), and via the CODESYS file manager within the IDE.

### Recommended method: Download from the Omega web portal

Navigate to your Omega's web portal, login and click on "CODESYS Downloads". Follow the steps there to download the libraries and device descriptions as .zip file from that page, and install in CODESYS, see Figure 1.



Figure 1. Navigate to "CODESYS getting started".



### Alternative method: Download via the CODESYS IDE file manager

This step only works if you already have your Omega device description installed in the CODESYS IDE and you have some CODESYS experience. If you haven't, please get the libraries and device descriptions via one of the other methods.

First, find and connect to your Omega, and navigate to the 'Files' tab in the 'Device' window, see Figure 2.



### Figure 2. Navigate to the 'Files' tab

The local file system is visible on the left side. After clicking the refresh icon on the right side, the file system on the Omega becomes visible, see Figure 3.

Devices – 🕈 🗙	Device X						~
ExampleForPS	Communication Sattings	Host Location		• Im X 4	Runtime Location Da /		• 🖂 🗙 🐼
= 1 Device (Penko Omega)		News	Circ	Madera	News	Circ	Madera
Ifi Omega Rack (Omega Rack)	Applications	Name	Size	Modified	Name .	SIZE	Modified 1.
	Backup and Pestore				i trend		
		-			ialarms		
	Files				ac_persistence		
	100				anc		
	**9				Application		
	PLC Settings				sPicLogic\$		
	PI C Shell				STemp\$		
					\$Shared\$		
	Users and Groups				SUSEDevA\$		
	Arress Rights				SChannelPrograms\$		
					\$DeviceDescriptions\$		
	Symbol Rights				CodesysLibraries\$		
	Licensed Software Metrics				Svisu\$		
					Salarms\$		
	Task Deployment				<pre>\$ac_persistence\$</pre>		
	Status				 (iii) \$_cnc\$		
					Sogfiles\$		
	Information				linux.swu	163,68 MB (171.628	24-6-2021 11:14
					_		



All directories CODESYS knows about become visible. The \$DeviceDescriptions\$ directory contains all device descriptions that can be used with the Omega. The \$CODESYSLibraries\$ directory contains CODESYS libraries that can be installed into your IDE to make use of all Omega functionality. Double-click the \$DeviceDescriptions\$ directory, select all device descriptions and use the arrows in the middle to copy them to a location on your hard drive. Similarly, open the \$CODESYSLibraries\$ directory and copy the libraries to your PC.

### 2.2. INSTALL DEVICE DESCRIPTIONS AND LIBRARIES

Follow the standard CODESYS procedure to install the device descriptions and libraries that were extracted from the .zip file.



### 3. VIEW DOCUMENTATION INSIDE THE LIBRARY

Extensive documentation of all objects and types can be found in the Penko EasyWeigher library itself from within CODESYS. Select the "Penko EasyWeigher Library" in the Library Manager in your project to find information about the automatically generated objects and how to use them. Here you can also find the list of *enum* definitions that is supported by your library.

See Figure 4.



Figure 4. View Penko Easyweigher library documentation in the CODESYS IDE

### 4. SET UP AN EXAMPLE PROJECT

### Prerequisites

Before setting up an example project, please make sure that your Omega is setup correctly. For more information on how to do this, and for all connection diagrams, refer to the Omega manual (7600M1082-<LANGUAGE>-R8 MANUAL OMEGA.pdf'). Please make sure that:

1. You can navigate to the Omega web portal in your browser via its IP address.

2. The first slot in the rack is occupied by an Omega ADC DIO module. It does not matter what is plugged into the other slots.

3. A load cell or load cell simulator is attached to the channel 1 load cell connector of the Omega ADC DIO module.

### 4.1 CREATE THE PROJECT

When you have confirmed that your Omega is online and its web portal is reachable in your browser, start CODESYS and create a new 'Standard project'.

After clicking "OK", CODESYS asks you to select the device this project is for, and the programming language you'll be working in. Follow Figure 5 to select the "Penko Omega (Penko Engineering BV)" device. Then, select the programming language. In your own future projects you can use what you are most comfortable with, but this example project will use the Function Block Diagram language (Figure 6)



Standard P	roject		×							
	You are about to create a new standard project. This wizard will create the following objects within this project: - One programmable device as specified below - A program PLC_PRG in the language specified below - A cyclic task which calls PLC_PRG - A reference to the newest version of the Standard library currently installed. Device Perko Omega (Penko Engineering BV) MT-CTRL (Weintek Labs., Inc.) CODESYS Control for Raspberry PI NC SL (35 - Smart Software Solutions GmbH) CODESYS Control for Raspberry PI NC SL (35 - Smart Software Solutions GmbH) CODESYS Control RTE V3 V4 (35 - Smart Software Solutions GmbH)									
	Device	Penko Omega (Penko Engineering BV)	$\sim$							
	PLC_PRG in	CMT-CTRL (Weintek Labs., Inc.) CODESYS Control for Raspberry PI MC SL (3S - Smart Software Solutions GN CODESYS Control for Raspberry PI SL (3S - Smart Software Solutions GmbH) CODESYS Control RTE V3 (4S - Smart Software Solutions GmbH) CODESYS Control KITE V3 464 (3S - Smart Software Solutions GmbH) CODESYS Control KITE V3 464 (3S - Smart Software Solutions GmbH) CODESYS Control Win V3 (4S - Smart Software Solutions GmbH) CODESYS Control Win V3 (4S - Smart Software Solutions GmbH) CODESYS Softwort Win V3 (4S - Smart Software Solutions GmbH) CODESYS Softwort Win V3 (4S - Smart Software Solutions GmbH) CODESYS Softwort Win V3 (4S - Smart Software Solutions GmbH) CODESYS Softwort Win V3 (4S - Smart Software Solutions GmbH)	mb⊦ I)							
		CODESYS Softmotion RTE V3 x64 (33 - Smart Software Solutions GmbH) CODESYS Softmotion Win V3 (35 - Smart Software Solutions GmbH) CODESYS SoftMotion Win V3 (35 - Smart Software Solutions GmbH) CODESYS SoftMotion Win V3 x64 (35 - Smart Software Solutions GmbH) Period Omega (Period Engineering BV) Ventore Discussion LL BerSS ventors Later Loc )								

Standard P	roject		×
	You are abou objects within - One program - A program F - A cyclic task - A reference	It to create a new standard project. This wizard will create the following n this project: nmable device as specified below YLC_PRG in the language specified below c which calls PLC_PRG to the newest version of the Standard library currently installed.	
	Device	Penko Omega (Penko Engineering BV)	$\sim$
	PLC_PRG in	Function Block Diagram (FBD)	$\sim$
		Continuous Function Chart (CFC) Continuous Function Chart (CFC) - page-oriented	1
		Function Block Diagram (FBD)	
		Lader Logic Diagram (LD) Sequential Function Chart (SFC) Structured Text (ST)	

Figure 6. Select FBD as programming language

Figure 5. Select Penko Omega as device

When the device and language are configured, click OK. You should now see an empty project for a Penko Omega device, with an "Omega\_Rack" as attached device as in Figure 7.

🕡 Gett	ingStarte	d.proje	ect - COD	ESYS										
File	Edit	View	Project	Build	Online	Debu	ug	Tools	Windo	w	Help			
1			CH X	þ í	×М	್ಷ 🐴	<b>^</b> _	1 9	(*)(*)	1	a   🏜 •	· 🖻   🏙	Арр	lication [Device:
Devices					<b>•</b> 1	хų								
- () (	GettingSta	arted				-								
<u>⊨</u> (	👔 Device	e (Penko	o Omega)											
	🖻 🗐 PL	.C Logic												
	<b>⊟</b> -( <b>C</b>	👂 App	lication											
		- 🎁 L	ibrary Mar	nager										
		🗗 P	LC_PRG (	PRG)										
	E	۱ 🌉 🖻	ask Config	guration										
		<u> </u>	🍃 MainTa	sk										
			·····································	C_PRG										
	🖻 - 🗊 o	mega_F	lack (Ome	ga Rack)										
	- <b>K</b>	<em;< td=""><td>oty&gt;</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></em;<>	oty>											
	. <b>с</b>	<em;< td=""><td>oty&gt;</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></em;<>	oty>											
	Ę	<em;< td=""><td>oty&gt;</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></em;<>	oty>											
	Ę	<em;< td=""><td>oty&gt;</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></em;<>	oty>											
	Ę	<em;< td=""><td>oty&gt;</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></em;<>	oty>											
		<em;< td=""><td>oty&gt;</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></em;<>	oty>											
	5	<em;< td=""><td>oty&gt;</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></em;<>	oty>											
	<b>C</b>	<em;< td=""><td>oty&gt;</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></em;<>	oty>											

Figure 7. An Omega starter project.

### Add an Omega ADC DIO module to the Omega Rack

Let CODESYS know about your Omega ADC DIO module by right-clicking the first slot in the Omega Rack, and select "Plug Device" (Figure 8). You should see four options in the popup window (Figure 9), one of which is the Omega ADC DIO Card. Select this and click "Plug Device" in the right-bottom corner. After CODESYS has added the card, click "Close".



ame 🖸	mega_ADC_DIO_Card					
Action Appen	nd device 🔘 Insert device	e 💿 Plug d	evice 🔾 U	Ipdate device		
String for	a full text search		Vendor	<all vendors=""></all>		
Name		Vendor	-	Version	Description	
в- 🗊 м	liscellaneous					
- 1	Omega ADC AIO Card	Penko Eng	ineering BV	3.5.17.0	Dual channel ADC and Analog IO card for the Omega PLC	
-0	Omega ADC DIO Card	Penko Eng	ineering BV	3.5.17.0	Dual channel ADC and Digital IO card for the Omega PLC	
- 0	Omega AIO Card	Penko Eng	ineering BV	3.5.17.0	Dual channel Analog IO card for the Omega PLC	
6	🖞 Omega DIO Card	Penko Eng	ineering BV	3.5.17.0	Dual channel Digital IO card for the Omega PLC	
Group	by category 🔲 Display a	all versions (f	or experts o	nly) 🗌 Displa	ay outdated versions	
i Na	ame: Omega ADC DIO Carc endor: Penko Engineering B ategories: ersion: 3.5.17.0 rder Number: 760084726	l IV DC and Digita	IO card for	the Omega PLC	Ŷ	Ŵ
Ci Vi O D	escription: Dual channel A	be and bigits				
Ca Vi O D lug sele Empty	escription: Dual channel A cted device into the slo	t				
Plug sele <empty:< td=""><td>escription: Dual channel A cted device into the slo &gt; u can select another target</td><td>t node in the</td><td>navigator w</td><td>hile this window</td><td>v is open.)</td><td></td></empty:<>	escription: Dual channel A cted device into the slo > u can select another target	t node in the	navigator w	hile this window	v is open.)	

Figure 8. Plug a device in the first slot.

Figure 9. Select 'Omega ADC DIO Card'

Two new devices have been added of type "Omega ADC DIO easyIND" below the plugged device, one for each channel in the card.

### 4.2 EXAMPLE IO MAPPING

Create a new variable in PLC\_PRG called "rNetWeight" of type "std:REAL":

PROGRAM PLC\_PRG VAR rNetWeight : REAL;

END\_VAR

Then map this variable to the "Net" parameter of the first channel. To to this, double click the first channel, navigate to the "Mapping" tab and open the "Weights" folder. Click the three dots to open a window where you can select the variable to map (Figure 10). Then follow Figure 11 to add a mapping to "rNetWeight".

GettingStarted.project* - CODESYS								- 🗆 🗙
File Edit View Project Build Online Debug Tools V	Vindow Help Git							<b>T</b> 18
🋍 🚔 🔚 📾 🗠 🕹 🛍 🖄 🖄 🐘 🏦	🐂 🎕 🛱 📷 🕤 🛗 Application	Device: PLC Logic] 🔹 🧐	98 x = 4	[≣ € <u>1</u> * <u>1</u> 8	◆   麗   국   ∛			
Devices 🗸 🖡	X M Omega_ADC_DIO_easyIND	×						-
= i GettingStarted	Ranko:OmegaCardToADCDIOIntef	(m. ).		<b>F</b> 11 <b>e</b> 1 <b>e</b> 1				1
😑 🔟 Device (Penko Omega)	ace IEC Objects	Find		Filter Show all		Add FB for IO Channel	o instance	
E II PLC Logic	2. Omega ADC DIO Channel I/O	3 Variable	Mapping	Channel	Address	Туре	Unit	Description
= 😳 Application	Mapping	📃 🔄 Weights	_					
Library Manager	Omega ADC DIO Channel IEC	4	·	Net	%ID0	REAL		Net weight
PLC_PRG (PRG)	Objects		<u> </u>	Gross	%ID1	REAL		Gross weight
Task Configuration	Status	🍫		Tare	%ID2	REAL		Current tare weig
🗏 💝 MainTask		iii - ¥≱		Status	%IB12	Penko_EasyWeigher_Library.IndicatorStatus		Indicator status
de PLC_PRG	Information	🕸 - 🚞 Digital in						
Omega_Rack (Omega Rack)		🖲 📴 Digital out						
Omega_ADC_DIO_Card (Omega ADC DIO Card)		😟 - 🚞 Debug info						
<ul> <li>M Omega_ADC_DIO_easyIND (Omega ADC DIO easyIND)</li> </ul>								
Omega_ADC_DIO_easyIND_1 (Omega ADC DIO easyINL	"							
<empty></empty>								
r compty >								
<ul> <li>Comply&gt;</li> <li>Emoty&gt;</li> </ul>								
<pre>Scipty&gt; </pre>								
<pre>Scipty&gt; </pre>								
<ul> <li>Sumpty&gt;</li> </ul>		·						>



put Assistant					
Text Search Categories					
Variables	Name     Application     G PLC_PRG     PRG     InConfig_Globals			Type Application PROGRAM REAL VAR_GLOBAL	Α
Structured view	٢		Filter	None	:
ocumentation		☑ Insert with arguments		Insert with namespace	e prefix
NetWeight: REAL(VAR)					
				ОК	Cancel

#### Figure 11. Map rNetWeight to the IO Channel.

The "rNetWeight" variable is now mapped to the "Net" IO channel of Channel 1 of the Omega ADC DIO Card. There is one more thing to do though:

If we would upload this, CODESYS will notice that "rNetWeight" is not used in the code, and optimize its access away. To prevent this, force CODESYS to always update all mapped variables by selecting "Enabled 1" for "Always update variables". See Figure 12 for how to do this.

Devices 💌 🖡	×	Comega_ADC_DIO_Card	PLC_PRG 🕅 Omega_	ADC_DIO_easyIND
	•	Communication Settings Applications Backup and Restore Files	Application for I/O handling PLC Settings Update IO while in stop Behavior for outputs in stop Always update variables But Cycle Ontoos 4	Application
CPLC_PRG     CRG     Omega_Rack (Omega Rack)     GGGGGGGGG	2.	Log PLC Settings PLC Shell Users and Groups	Bus cycle Options Bus cycle task Additional Settings Generate force variables fo	Enabled 2 (Jaways in bus cycle task)
- C <empty> - C <empty> - C <empty> - C <empty> - C <empty> - C <empty></empty></empty></empty></empty></empty></empty>		Access Rights Symbol Rights Licensed Software Metrics		

### Figure 12. Tell CODESYS to always update all variables.

Now we're ready to login and upload. Changing the weight on the load cell or load cell simulator should be visible in the rNetWeight variable.

### 4.3 WEIGHER STATUS STRUCT MAPPING

The weigher status is a struct consisting of a number of bits that describe the current status of the weigher. For example, it tells you whether or not the weight is stable, if the weight is higher than what the hardware can manage, or if (preset) tare is active. The mapping is found in the 'Weights' folder of the IO mapping, with the name 'Status'.



To use this status in your program, create a new variable with the correct struct type in your program: PROGRAM PLC\_PRG VAR

```
rNetWeight : REAL;
stStatus : Penko_EasyWeigher_Library.IndicatorStatus;
END_VAR
```

The CODESYS autocompletion is your friend in the struct type: start typing 'Penko\_', press "CTRL+Space", and CODESYS will automatically fill the rest of the EasyWeigher name. After the '.', CODESYS will show 'IndicatorStatus' as one of the options. Map stStatus to the Status channel in the IO mapping, similar to how rNet was mapped in the previous section. After this, your IO mapping should look like Figure 13 below.



Figure 13. Map the stStatus struct to the Status Channel

### 4.4 WEIGHER COMMANDS (TARE, ZERO, ETC)

Each channel in the rack automatically generates four objects that can be found in the "IEC Objects" tab of the respective channel. The name of the object is prefixed with the name of the channel it belongs to. In the example that is "Omega\_ADC\_DIO\_easyIND", then an underscore, and then the type of object.

If the name of the channel changes, then the prefix changes as well.

Basic CODESYS skills are assumed in this section, such as adding a POU and creating a Visualization with a textfield and a button.

This section will focus on the Command object. We will extend the example program with visualization that shows the current weight, and a button to perform a tare action.

### Show the current weight in a Visualization.

First add a Visualization Manager to your project. Then add a textfield with the text "%f" and our "rNetWeight" variable as "Text variable".

Upload your program and verify that the visualization shows the current weight.

### Insert the Command function block for the first channel.

Insert an empty Box in the first network of the PLC\_PRG, and make it the "\_Command" object that starts with the same name as your first channel. If you did not rename the first channel, the object will be called "Omega\_ADC\_DIO\_easyIND\_Command". See Figure 14. If all went well, the first network now contains the Omega\_ADC\_DIO\_easyIND\_Command object as shown in Figure 15.





Figure 14. Use an Omega\_ADC\_DIO\_easyIND\_Command function block.



Figure 15. The Omega\_ADC\_DIO\_easyIND\_Command function block has been added.

The Command object has three inputs:

- 1. xExecute: a Boolean that indicates whether the function block should be doing its command.
- 2. eCommandType: an Enum of type CommandType. The Enum is defined in the EasyWeigher library, and defines for example COMMAND\_TARE\_SET. See the documentation inside the EasyWeigher library for more commands (follow Chapter 3).
- 3. rOptionalDataIn: Some commands require an input argument, such as COMMAND\_PTARE\_SET to set the preset tare to a predefined weight. Other commands such as COMMAND\_TARE\_SET do not require an input argument. In those cases, this field can be left empty.

The Command also has three outputs:

- 1. xDone: a Boolean that is TRUE when the function block is done, and FALSE when the function block is not executing (xExecute is FALSE) or not yet done executing and thus needs another execution cycle.
- 2. xError: a Boolean that is TRUE when something went wrong during execution.
- 3. eError: an enum of type ErrorType, which indicates what went wrong in case of an error. When there is no error, this will be ERROR\_SUCCESS. All values of this enum are defined and documented in the EasyWeigher library.

### Use the Command function block for the first channel.

We will set and reset a tare as a test. First, add two booleans to your PLC\_PRG that indicates whether the Command functions block will be run, one to set the tare, and one to reset the tare: PROGRAM PLC\_PRG

```
VAR
rNetWeight : REAL;
xExecuteTareSet : BOOL;
xExecuteTareReset : BOOL;
FND VAR
```

END\_VAR

Then, tie xExecuteTareSet to the xExecute input of the Command function block, and set the eCommandType input to COMMAND\_TARE\_SET. The easiest way to do this is start typing "COMMAND\_" in the eCommandType field, and press CTRL+SPACE to let CODESYS do the autocompletion. This will give a list of COMMAND\_ values that can be filled in. COMMAND\_TARE\_SET can be selected from the list by double clicking the entry. CODESYS will automatically add the namespace, which is 'Penko EasyWeigher Library.CommandType' in this example.

For testing it would be nice to automatically set xExecuteTareSet back to FALSE when the command block is done: to achieve this, xExecuteTareSet is set to FALSE when xDone is TRUE, and is unchanged while xDone is FALSE. See Figure 16.

1	Omega ADC_DIO_easyIND_Command
	Penko EasyWeigher Library.Command NOT AND
	xExecuteTareSet xExecute xDone 2 xExecuteTareSet
	Penko_EasyWeigher_Library.CommandType.COMMAND_TARE_SETeCommandType xError C
	-rOptionalDataIn eError
	xExecuteTareSet

Figure 16. A simple network to activate a tare.

In a similar way, we can create a network to reset the tare below that. Note that we added two new Boolean values: xExecuteTareReset and xDoneTareReset, and that the eCommandType input has changed to COMMAND\_TARE\_RESET.

2	Omega_ADC_DIO_easyIND_Command	
	Penko_EasyWeigher_Library.Command NOT AND	
	xExecuteTareReset xExecute xDone 2 xExecuteTareR	eset
	Penko_EasyWeigher_Library.CommandType.COMMAND_TARE_RESET — eCommandType xError X	
	-rOptionalDataIn eError	
	vFveriteTareBeset	

Figure 17. A simple network to deactivate the tare.

Set the xExecuteTareSet and xExecuteTareReset variables using a button in the Visualization or via the debugging functionality, and check that setting a tare sets the net weight to zero, and resetting the tare sets the weight back.

### 4.5 READING CONFIGURATION

The function block to read configuration is named \$(deviceName)\_ConfigRead. If the first channel of the first Omega card was not renamed, the function block named Omega\_ADC\_DIO\_easyIND\_ConfigRead can be used to read configurations of that channel.

### Insert the ConfigRead function block for the first channel.

To start, add a third network, insert an empty box in that network and make it the \_ConfigRead function block, similar to how the \_Command function block was inserted in Figure 14. Your network should look like Figure 18 below:





Figure 18. The Omega\_ADC\_DIO\_easyIND\_ConfigRead block is added to a network.

All inputs and outputs of this ConfigRead block are explained in detail in the EasyWeigher library documentation within CODESYS, (see Chapter 3 on how to get there. The Config block works following the same principles as the Command block: it executes while input xExecute is high, and input eConfigType defines what configuration parameter is being read. It is important to note that there are two data outputs: udiDataOut and rDataOut. Some configuration parameters are weights. As before, all weights are of type REAL, and these parameters thus need to be read from rDataOut.

For example, when eConfigType is set to CONFIG\_MAXLOAD, the block will read the MaxLoad parameter from the weigher configuration, and this is a weight. Therefore, the correct output appears at rDataOut.

Similarly, when eConfigType is set to CONFIG\_DECIMALPOINT, the block will read out the number of decimals the weigher uses internally. This is inherently an integer number. Therefore, the correct output appears at udiDataOut.

All possible enum values for eConfigType can be found in the EasyWeigher library documentation within CODESYS (see Chapter 3), and each value documents which output must be used.

### Use the ConfigRead function block for the first channel.

Let's use this ConfigRead block to read out the decimal precision that is used in the weight. First, add some variables to the PLC\_PROG, one to execute the block and one to store its result:

The number of decimals that the weigher uses internally is read by setting input eConfigType to PENKO\_EASYWEIGHER.ConfigType.CONFIG\_DECIMALPOINT, adding the udiDecimalsRead variable to the udiDataOut output, and tieing the xExecuteConfigReadDecimals variable to the xExecute input like we did before. For easier testing, the NOT-AND network is added after xDone to automatically reset xExecuteConfigReadDecimals. See Figure 19 below.



#### Figure 19. Variables are tied to the ConfigRead block.



Use the CODESYS debugging functionality to set xExecuteConfigReadDecimals to TRUE, and observe that the number of decimals that are used in the weighing card are put into udiDecimalsRead. In the example below, the weight used three decimals precision (Figure 20).

PLC_PRG 🗙 📆 Device 🙀 Omega_A	ADC_DIO_easyIND						
Device.Application.PLC_PRG							
Expression	Type Value	Prepared value	Address	Comment			
🔷 rNetWeight	REAL 0						
xExecuteTareSet	BOOL FALSE						
xExecuteTareReset	BOOL FALSE			2. RIGHT	MO	USE BUTTON	
xExecuteConfigReadDecimals	BOOL FALSE	TRUE			- V	Cut	
udiDecimalsRead	UDINT 3				Ba	Conv	
	· · · · · · · · · · · · · · · · · · ·					Deste	
	4. 🖲					Paste	
			<del>, Auxarbaran</del>	SELLOL	$\sim$	Delete	
						Select All	
				xExecu	r	Browse	•
					63	Add to Watchlist	
2			Omega_ADC_D	IO_easyIND_Command		Advanced	•
		Denne internet	Penko_EasyWei	gher_Library.Command		Innut Anistant	
Penko FasyWeigher Library Cor	MEADD TARE RES	FT C 2	KEXecute CommandTupe	xDone	12 B	Input Assistant	
			rOptionalDataIr	eFrror ERR		Keractoring	•
			•		10	New Breakpoint	
				xExect		Toggle Breakpoint	
					→≡	Run to Cursor	
			Omogen ADC D	IO apavIND ConfigDood	8	Set Next Statement	
3		Г	Danka RageWai	abon Library ConfigRead	2		
	xExecuteConfigReadI	ecimals FALSE <tru< td=""><td><pre>penko_basywei &gt; ecute</pre></td><td>xDone</td><td>э.</td><td>Write All Values of 'Device.Application'</td><td></td></tru<>	<pre>penko_basywei &gt; ecute</pre>	xDone	э.	Write All Values of 'Device.Application'	
Penko_EasyWeigher_Library.Com	nfigType.CONFIG_DECIMALPOI	NT C 8	eConfigType	xError -		Force All Values of 'Device.Application'	
				eError —	E	Unforce All Values of 'Device.Application'	
				udiDataOut - u	10	Display Mode	+

Figure 20. Use the CODESYS debugger to toggle reading the number of decimals.

### 4.6 WRITING CONFIGURATION - INDUSTRIAL MODE

Writing a configuration in industrial mode is very similar to reading a configuration. The only difference is that a different Function Block must be used, which has inputs udiDataIn and rDataIn, instead of two outputs udiDataOut and rDataOut.

### Insert the ConfigWrite function block for the first channel.

Add a fourth network, insert an empty box in that network and make it the \_ConfigWrite function block, similar to how the \_Command function block was inserted in Figure 14.

Your network should look like Figure 19 below.

4 Omega_ADC_DIO_easyIND_ConfigWrite				
	Penko_EasyWeigher_Library.ConfigWrite			
	xExecute	xDone		
	222 — eConfigType	xError - ?	??	
	222 — udiDataIn	eError - ?	??	
	222 — rDataIn			

### Figure 21. The Omega\_ADC\_DIO\_easyIND\_ConfigWrite block is added to a network.

All inputs and outputs are documented extensively in the EasyWeigher library documentation within CODESYS, and work very similar to the earlier Command and ConfigRead blocks.

This ConfigWrite block has two data inputs, udiDataIn and rDataIn, to allow two types of data to be written to the indicator configuration. rDataIn is used to write weights of type REAL such as the MaxLoad parameter, udiDataIn is used to write integer values such as the number of decimals that is used internally.

**IMPORTANT**: Whenever one input is used, the other input MUST BE ZERO.



### Use the ConfigRead function block for the first channel.

Add two new variables xExecuteConfigWriteDecimals and udiDecimalsWrite to your PLC\_PRG, and complete the network similar to the ConfigRead network from the previous section. In the end, your network should look like Figure 22 below:



Figure 22. Variables are tied to the ConfigWrite block.

Note that, because udiDataIn must be used and not rDataIn, rDataIn has been set to zero.

Use the CODESYS debugging functionality to prepare a value for udiDecimalsWrite and xExecuteConfigWriteDecimals, similar to the previous section and Figure 20. If the number of decimals was e.g. 3, set udiDecimalsWrite to 2. After writing the new udiDecimalsWrite value (don't forget to set xExecuteConfigWriteDecimals high to do the write action), the weight in rNetWeight should have changed accordingly (e.g. by changing a factor 10).

### 4.7 WRITING CONFIGURATION - CERTIFIED MODE

When the Omega weighing module is configured to be in certified mode, the configuration is protected by a TAC value. The TAC value is used to prevent the configuration from being changed accidentally. Trying to write a config value without unlocking the configuration will result in an eError=ERRORTYPE\_ACCESSDENIED.

### Managing the TAC value

Unlocking the configuration is not difficult: first read the current required TAC value, and then write that same TAC value back. This will unlock the configuration for approximately 100 seconds.

In a certifiable system, the CODESYS programmer must create a bit of visualization for the end user, who has to enter the current TAC manually to write back his configuration. This way, writing configuration values becomes a conscious effort for the end user which reduces chances of accidentally changing the configuration and thereby breaking the certification. While it is *possible* for the CODESYS programmer to programmatically read out the TAC and write it back automatically, it takes away the conscious effort for the end user, and thus takes away the safety mechanism. If your product needs to be certified, this is very probably not allowed.

The most user friendly way to write the configuration from for example a visualization is to let your user fill in the configuration they want, and then write it to the Omega weighing module all at once. That way, your user does not have to enter the TAC code each time they write a configuration value, but only the one time that all configuration values are written.

### **Reading the TAC value**

The TAC value is part of the configuration block, and can be read just like any configuration using the \_ConfigRead block of your module, with input eConfigType set to PENK0\_EASYWEIGHER.ConfigType.CONFIG\_TAC. See Figure 23 below. If you want to have xExecuteConfigReadTAC automatically reset when the block is done, use the same NOT+AND boxes after xDone as we did in the previous examples.



6	Omega_ADC_DIO_easyIND_ConfigRead		
	Penko EasyWeigher Library.ConfigRead		
	xExecuteConfigReadTac	xExecute xDone	
	Penko_EasyWeigher_Library.ConfigType.CONFIG_TAC	eConfigType xError-	
		eError —	
		udiDataOut — udiTacRead	
		rDataOut —	

Figure 23. Read the TAC value using the ConfigRead block.

### Writing the TAC value

The TAC is written in the same way as any configuration value, using the \_ConfigWrite block. Again, the eConfigType must be set to PENK0\_EASYWEIGHER.ConfigType.CONFIG\_TAC, as in Figure 24.

7	Omega_ADC_DIO_easyIND_ConfigWrite		
	Penko EasyWeigher Library.ConfigWrite		.ConfigWrite
	xExecuteConfigWriteTac —	xExecute	xDone -
	<pre>Penko_EasyWeigher_Library.ConfigType.CONFIG_TAC —</pre>	eConfigType	xError -
	udiTacWrite —	udiDataIn	eError -
	0 —	rDataIn	

Figure 24. Write the TAC value using the ConfigWrite block.

### 4.8 CALIBRATION

All calibration actions are done using the \$(deviceName)\_Calibrate function block, as shown in Figure 25.

3	Omega_ADC_DIO_easyIND_Calibrate	
	Penko_EasyWeigher_Library.Calibrate	
	-xExecute xDone	
	-eCalibrateType xError	-
	-udiOptionalDataIn eError	-
	-rOptionalDataIn udiOptionalDataOut	-
	rOptionalDataOut	-

The \_Calibrate block operates in the same way as the \_Command, \_ConfigRead and \_ConfigWrite blocks. Depending on the calibration action you choose for input eCalibrateType you may have to provide extra input data to udiOptionalDataIn or rOptionalDataIn. Also depending on the calibration action is whether it function block generates an output in udiOptionalDataOut or rOptionalDataOut. Which actions there are, and what inputs they need and what outputs they generate is documented in the EasyWeigher library documentation within CODESYS.

Calibration is very similar to writing configurations in certified mode. The same way that the TAC code acts as a protection against accidental changes in the configuration, the calibration is protected by a CAL code. The CAL code needs to be managed in the same way as the TAC code in certified mode.



Figure 25. The \_Calibrate block.

### Managing the CAL code

The CAL code must be managed in the same way as the TAC code, see Section 4.6 above. In short, when a calibration action returns ERRORTYPE\_ACCESSDENIED, you will have to let your user enter the CAL code to unlock the calibration.

### Reading and writing the CAL code

To start, add the following variable definitions to your PLC\_PRG:

Reading and writing the CAL code is done in a similar way to reading and writing the TAC code. However, this time the \_Calibrate block is used, instead of the \_ConfigRead or \_ConfigWrite block. See Figure 26. Reading and writing the CAL code..



Figure 26. Reading and writing the CAL code.





#### 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? <u>www.penko.com</u>

#### 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** Distributor

A complete overview you will find on: <u>www.penko.com/Find-A-Dealer</u>

