

Welcome

to our

**CODESYS** Training



## Preliminary notes

Hardware: OPUS\_A3/A6\_Hardware\_Manual\_V1.0

Software: Operating System: OPUS\_A3\_C\_CPP\_Developer\_Guide  
OPUS update manual  
Projektor-Tool: HTML Help  
CODESYS: Codesys\_Operator\_Manual

Operators must read these installation instructions, particularly the safety information, and must be familiar with the operation of the equipment.

The following topics among others can be found in the above hardware and software documents : Used Instructions Types, Safety instructions, guarantee and liability, Intended Use, Getting Started and Technical Documentation

## Legal notes

All operating manuals and products names cited in the operating Trademarks manual are registered trademarks of Topcon Electronics GmbH & Co KG All rights reserved, including translation.

No part of this operating manual may be reproduced, in any form, (print, photocopy, microfilm, or a different process), nor may it be processed, duplicated, or distributed using electronic systems, without written permission from Topcon Electronics GmbH & Co KG, Geisenheim.

All previous versions are rendered obsolete with publication of New manuals. All information contained herein is subject to correction, manufacturer is not liable for any errors in this presentation

# Agenda

1. **Introduction**
2. Updating your device
3. Installing the CODESYS IDE
4. First project
5. CAN communication
6. Extended agenda
7. FAQ

# 1. Introduction – About me

- Dipl.-Ing. Electrical Engineer
- Over 12 years experience for Wachendorff Elektronik / Topcon Electronics
- Main work areas:
  - Customer support via phone / email
  - Trainings and workshops
  - Creation and maintenance of trade show / example projects
  - Technical documentation

# 1. Introduction – Training targets

Training targets:

- Understanding the project design concept of CODESYS\*
- Learning about our Driver structure to access device functionality
- Being able to configure the CAN protocol you need

\*Controller Development System

# Agenda

1. Introduction
- 2. Updating your device**
3. Installing the CODESYS IDE
4. First project
5. CAN communication
6. Extended agenda
7. FAQ

## 2. Updating your device – The developer cable

What are all those clamps on the developer cable?!

A/DI 1 – 4 – Analog / Digital Inputs

DO 1 – 3 – Digital Outputs

RS232 – serial console of the Linux system

CAN1/2 – CAN ports

USB – USB

Clamp 30 – Battery plus

Clamp 15 – Ignition (Should be connected into Clamp 30)

GND / Car GND – Ground (Should be connected if separate clamps)\*

Serv\_EN – Service Enable

cable color

brown,green,yellow,gre

white,lilac,blue

RxD-red-blue,TxD-red,GND-pink

CAN1-pink(high),blue(low)

CAN2-green(high),yellow(low)

VCC-red,GND-black,Low-white,High-green

red

red-blue

GND-black,CarGND-white, black connector(s)

black

\*Not for B series! There pin4 is WakeUp



## 2. Updating your device

**When you purchase new devices, always perform a software update first!**

All software elements should be on the same level as released from us:

**CODESYS IDE** on your PC

**Operating System (OS)** on the device

**CODESYS runtime** on the device

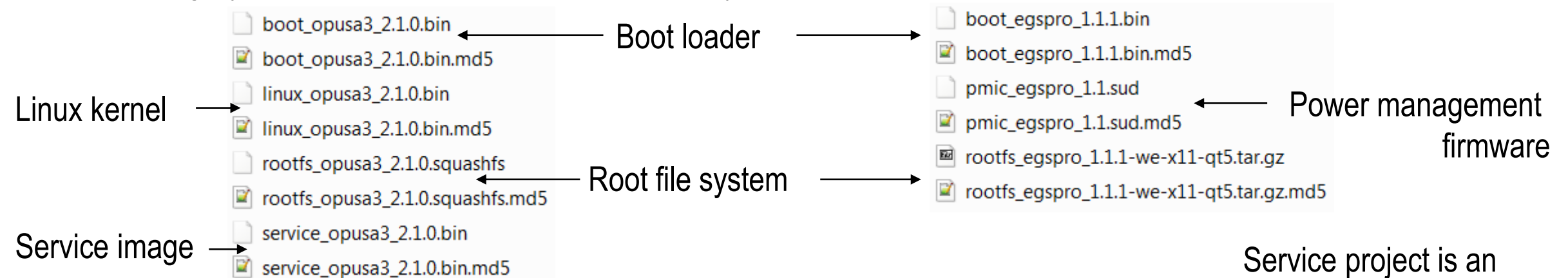
## 2. Updating your device – The update files

The update packet for your device (under \CustomerFiles\Device\OPUSAx\) consists of:

**A3 / A6 G1**

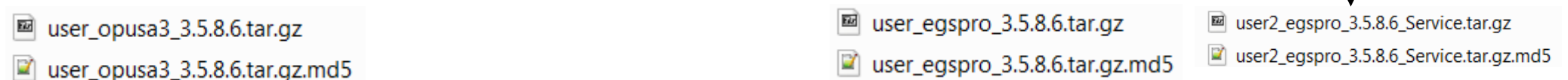
**A6 G2 / A8**

**OS – Operating System.** A customized embedded Linux system.

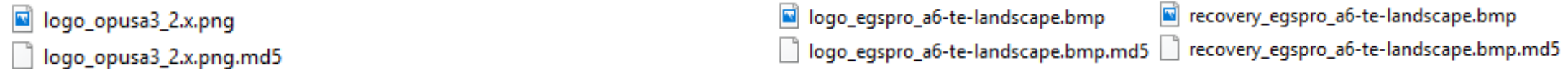


Service files to update from OS 1

**Codesys Runtime –** The application that displays your project on the device.

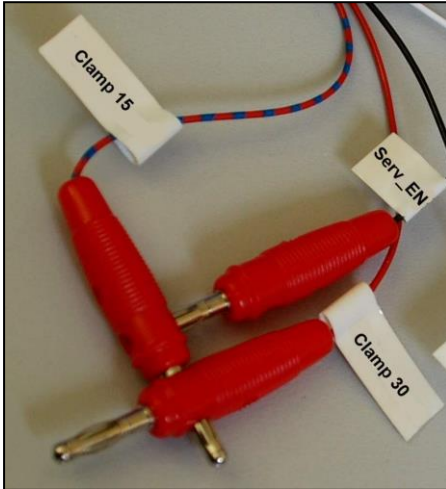


**Boot logo –** The image that is displayed when the device is started -> created by the customer (see chapter 7)



## 2. Updating your device – Preparations

4 a)



4 b)



1. Power off the device

2. Put all 12 update files in the root folder of a **FAT32** formatted USB stick and connect it with the device

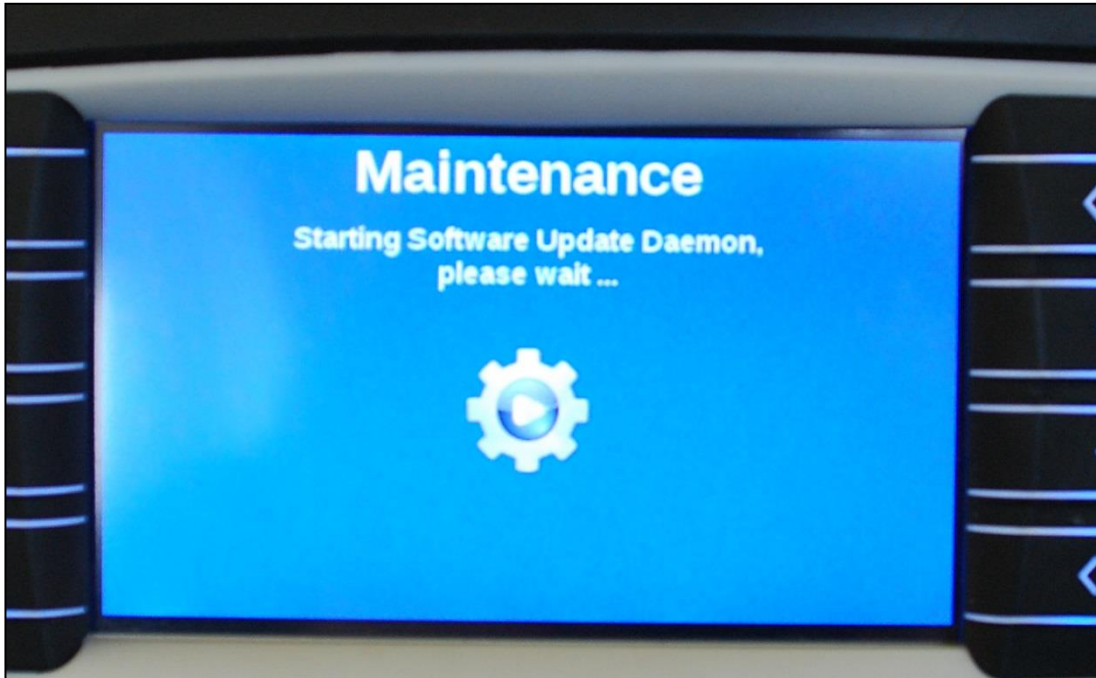
3. Connect the USB stick with the device

4. Power up the device in service mode by

a) Connecting clamp service enable OR

b) Pressing the left keys 1 + 2  
(hold for 3 seconds)  
-> for A8, press Key 1 + Stop

## 2. Updating your device – Almost done



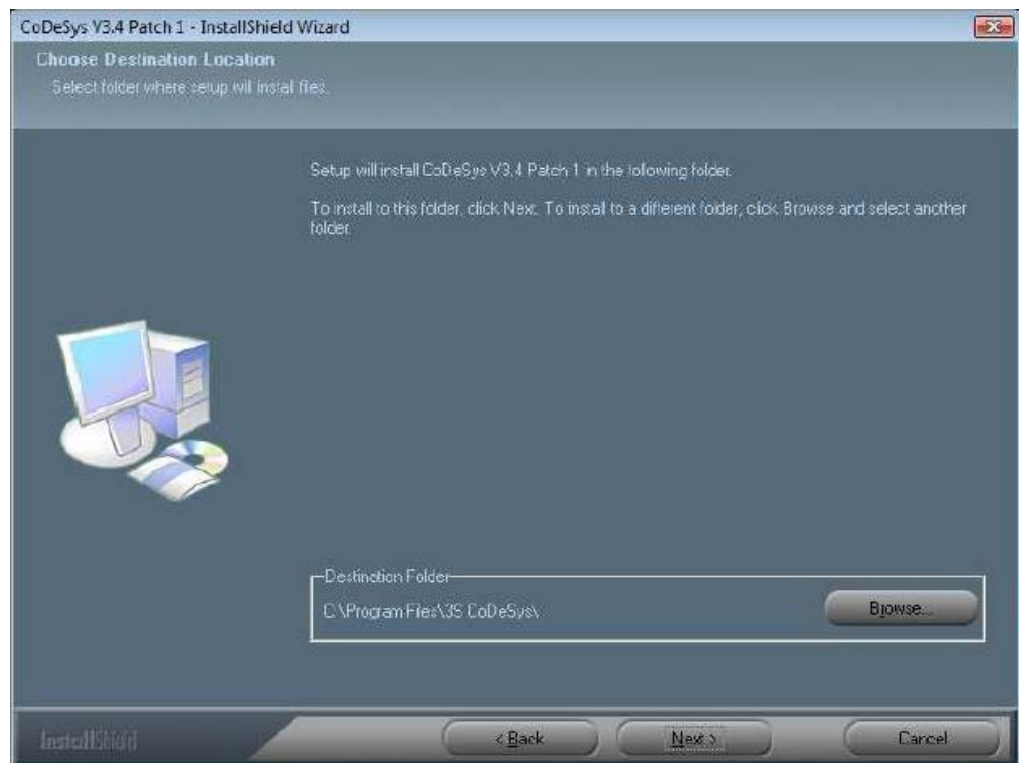
When you see this screen (or similar), let go of the keys or disconnect the clamp service enable

The update takes 3-5 minutes after which the device will restart and, after the touch screen calibration, if you have a touch screen device, show the service project.

# Agenda

1. Introduction
2. Updating your device
- 3. Installing the CODESYS IDE**
4. First project
5. CAN communication
6. Extended agenda
7. FAQ

### 3. Getting to know the CODESYS IDE – Installation



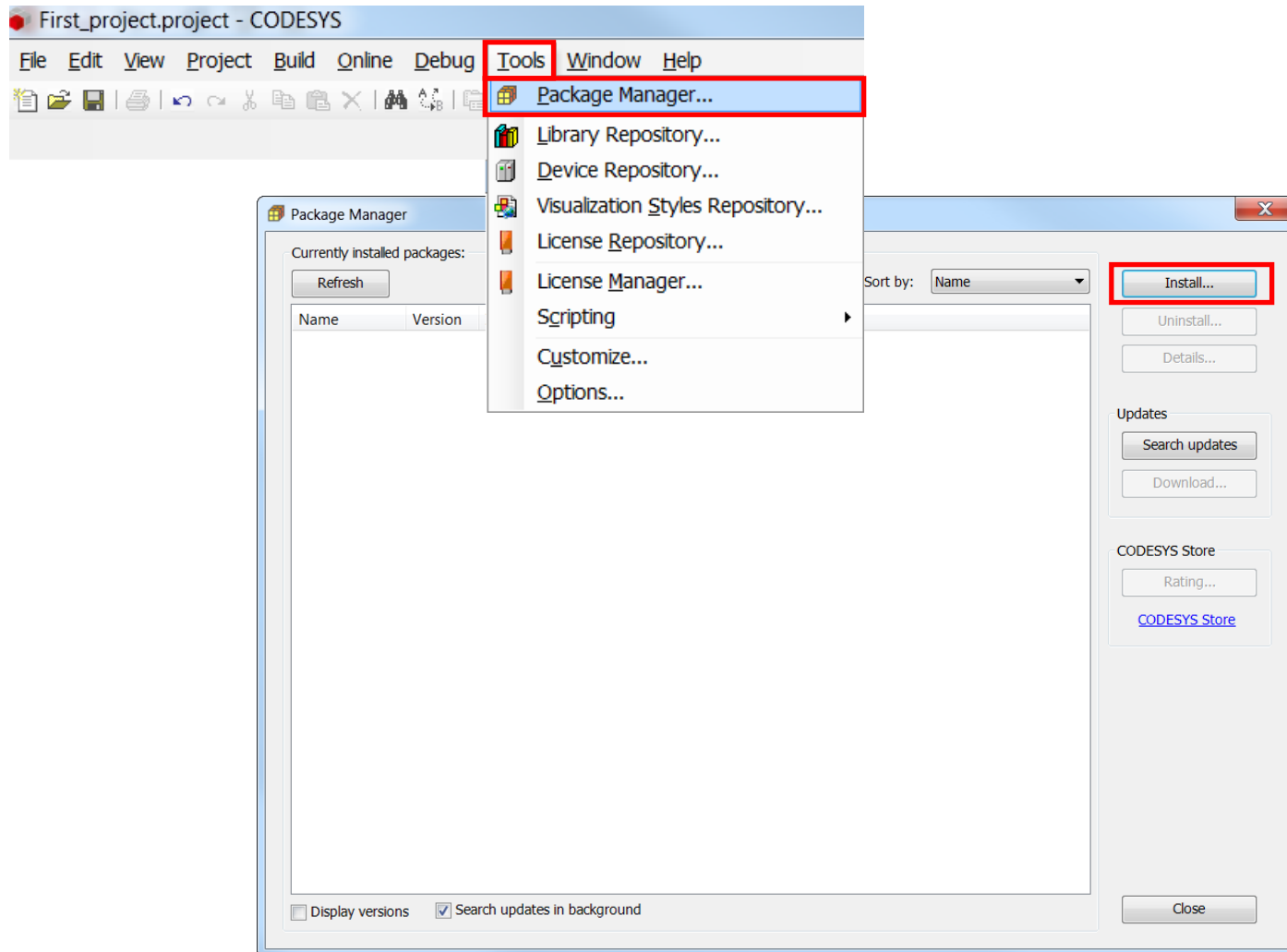
Install the CODESYS IDE with administrator rights

Click through the steps in the installation – usually no settings have to be changed

If you have another Codesys installed, choose a different installation location

Start Codesys after the installation

### 3. Getting to know the CODESYS IDE – Installation



In Codesys, select the menu  
*Tools -> Package Manager...*

In the dialog, click *Install...*  
and browse to `\CustomerFiles\PC`  
in your Codesys package

Install the TE\_\_\_\_\_package

### 3. Getting to know the CODESYS IDE – Program overview



Menu bar

Tool bars

Main window

Device tree

Object Toolbox

Build messages / errors

Object properties / devic tree (application) / POU(program organisation units)

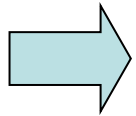


### 3. Getting to know the CODESYS IDE – Program overview



5 programming languages for application programming (IEC 61131-3) are available:

- **IL** (Instruction list) an assembler like programming language
- **ST** (Structured text) similar to programming in PASCAL or C
- **LD** (Ladder diagram) enables the programmer to virtually combine relay contacts and coils
- **FBD** (Function block diagram) enables the user to rapidly program both Boolean and analogue expressions
- **SFC** (Sequential function chart) convenient for programming sequential processes and flows



We will focus on ST code

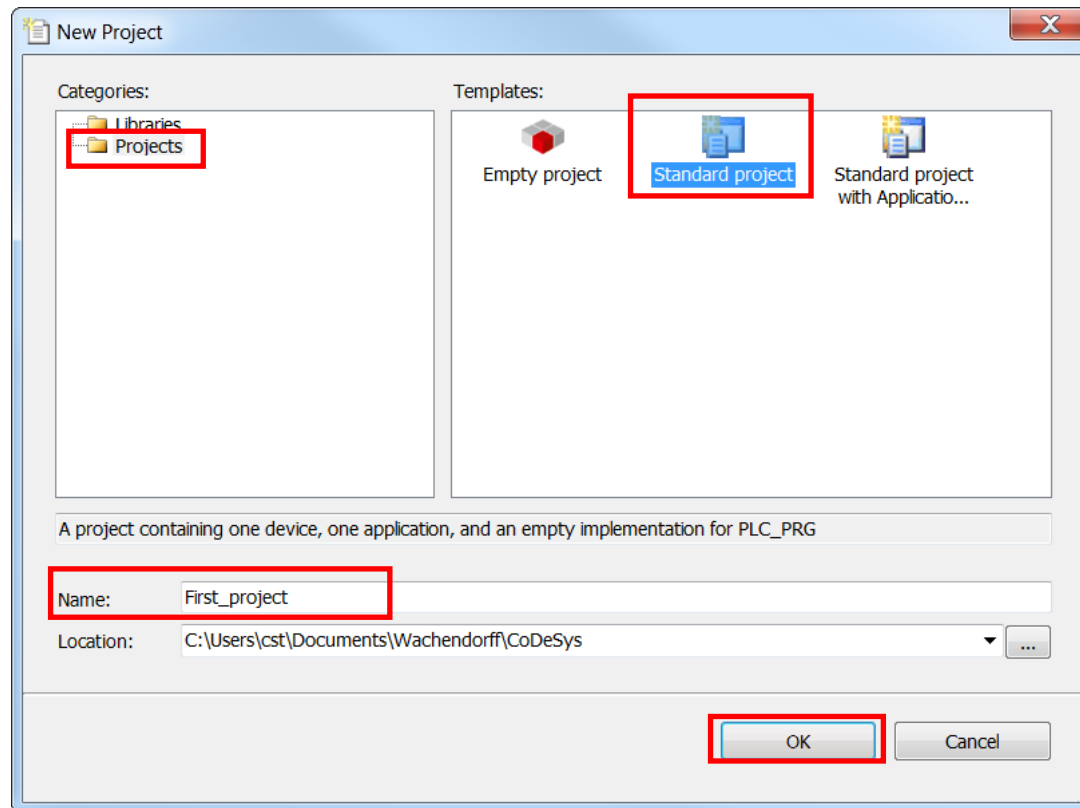
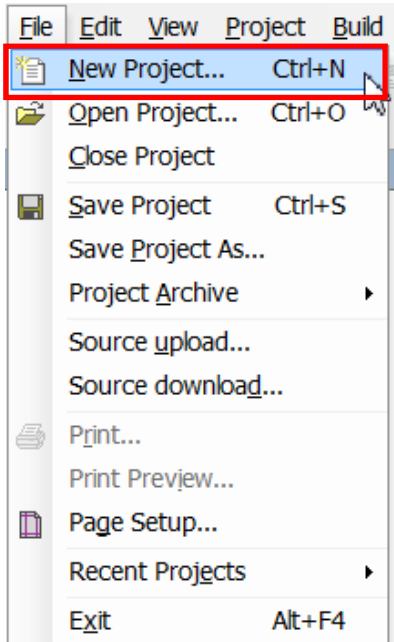
# Agenda

1. Introduction
2. Updating your device
3. Installing the CODESYS IDE
- 4. First project**
5. CAN communication
6. Extended agenda
7. FAQ

## 4. The first project

1. Create a new project
2. Configuration
3. The first program
4. The first visualisation
5. Driver module

## 4.1 The first project – Create a new project



To create a new project, click on *File* -> *New Project...* press the *New Project* icon or press CTRL + N

Select *Standard project*, choose a project name and location click *OK*

## 4.1 The first project – Create a new project

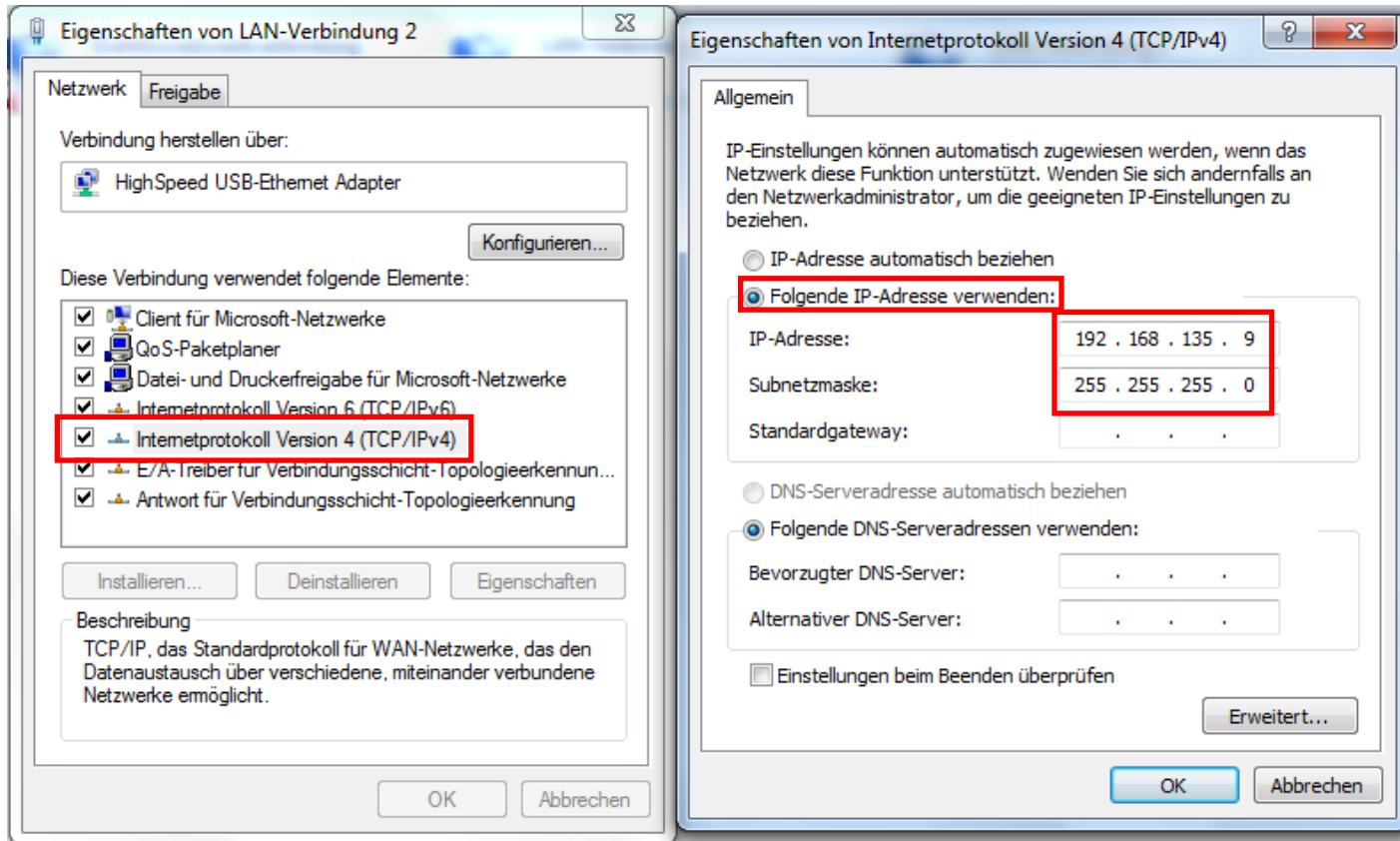
The screenshot shows the 'Standard Project' wizard dialog box. The title bar reads 'Standard Project'. The main text says: 'You are about to create a new standard project. This wizard will create the following objects within this project:'. Below this, a list of objects is shown: '- One programmable device as specified below', '- A program PLC\_PRG in the language specified below', '- A cyclic task which calls PLC\_PRG', and '- A reference to the newest version of the Standard library currently installed.' The 'Device:' dropdown menu is set to 'OpusA3-and-OpusA6-3.5.11.0 (Topcon Electronics GmbH & Co. KG)' and is highlighted with a red box. The 'PLC\_PRG in:' dropdown menu is set to 'Structured Text (ST)'. At the bottom right, the 'OK' button is highlighted with a red box. To the right of the dialog box, a 'Devices' panel shows a project tree structure for 'First\_project'. The tree includes: 'First\_project' (root), 'Device (OpusA3-and-OpusA6)', 'PLC Logic', 'Application' (under PLC Logic), 'Library Manager' (under Application), 'PLC\_PRG (PRG)' (under Application), 'Task Configuration' (under Application), 'MainTask' (under Task Configuration), 'PLC\_PRG' (under MainTask), and 'Driver (Driver)' (under Device).

Select the matching device description and choose Structured Text (ST) as the default programming language

The most basic project

- one device
- one application
- one program (PLC\_PRG)
- one task (MainTask)
- Driver module (for device specific functions)

## 4.2 The first project – Configuration – Network configuration



Go to the network adapter settings of your PC and change the settings for IPv4

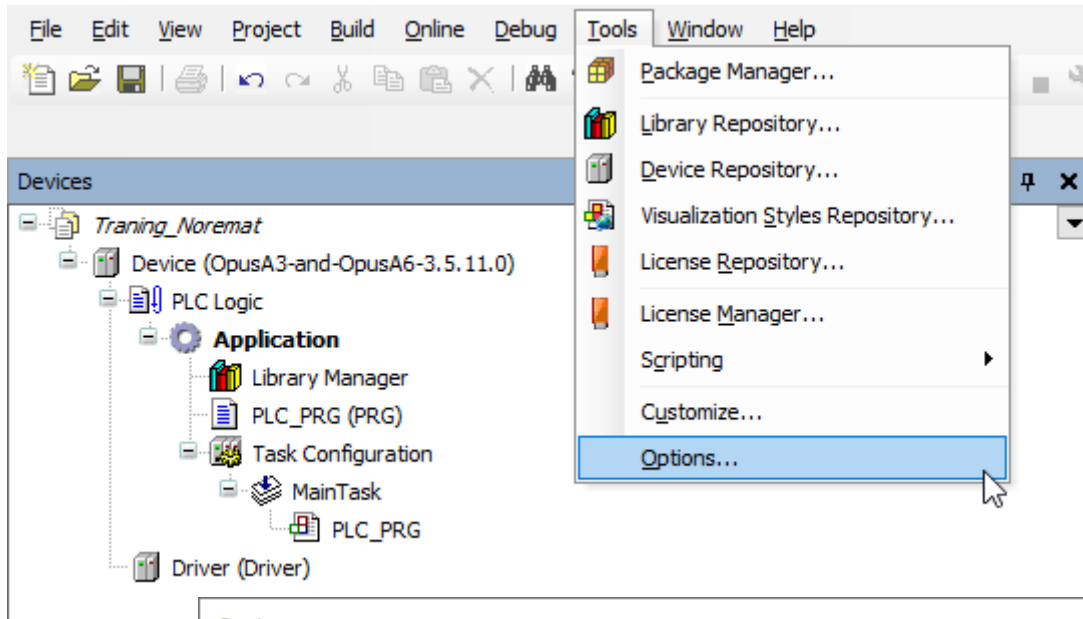
The computer IP has to be in the same subnet as the displays

Standard IP addresses for the displays (can be changed):

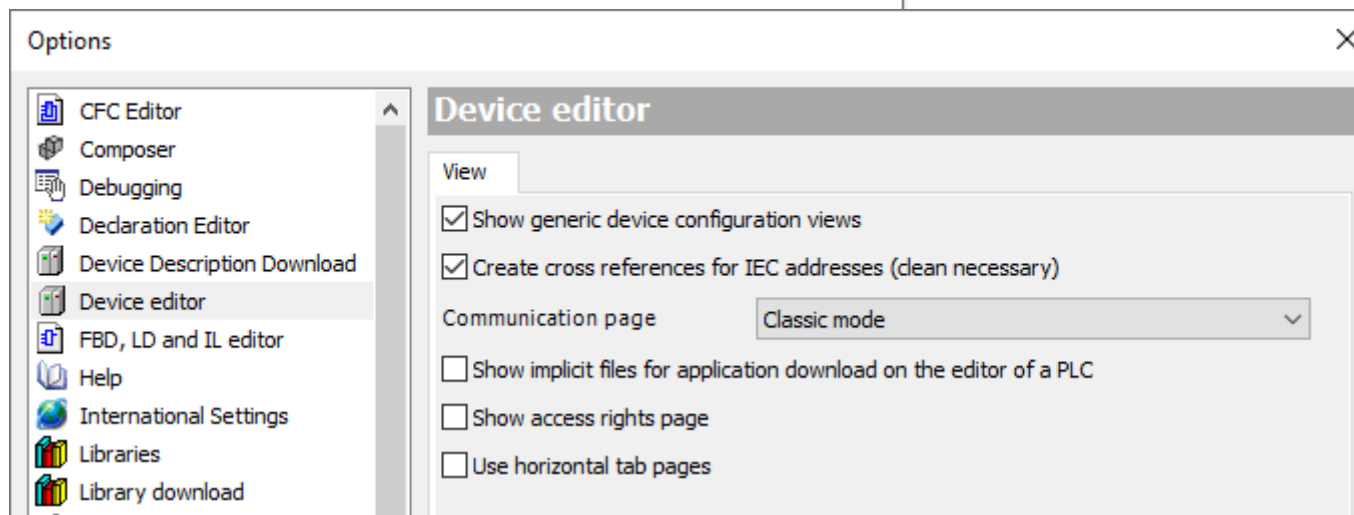
192.168.135.6

Now connect the running device with your PC over Ethernet

## 4.2 The first project – Configuration – Device connection

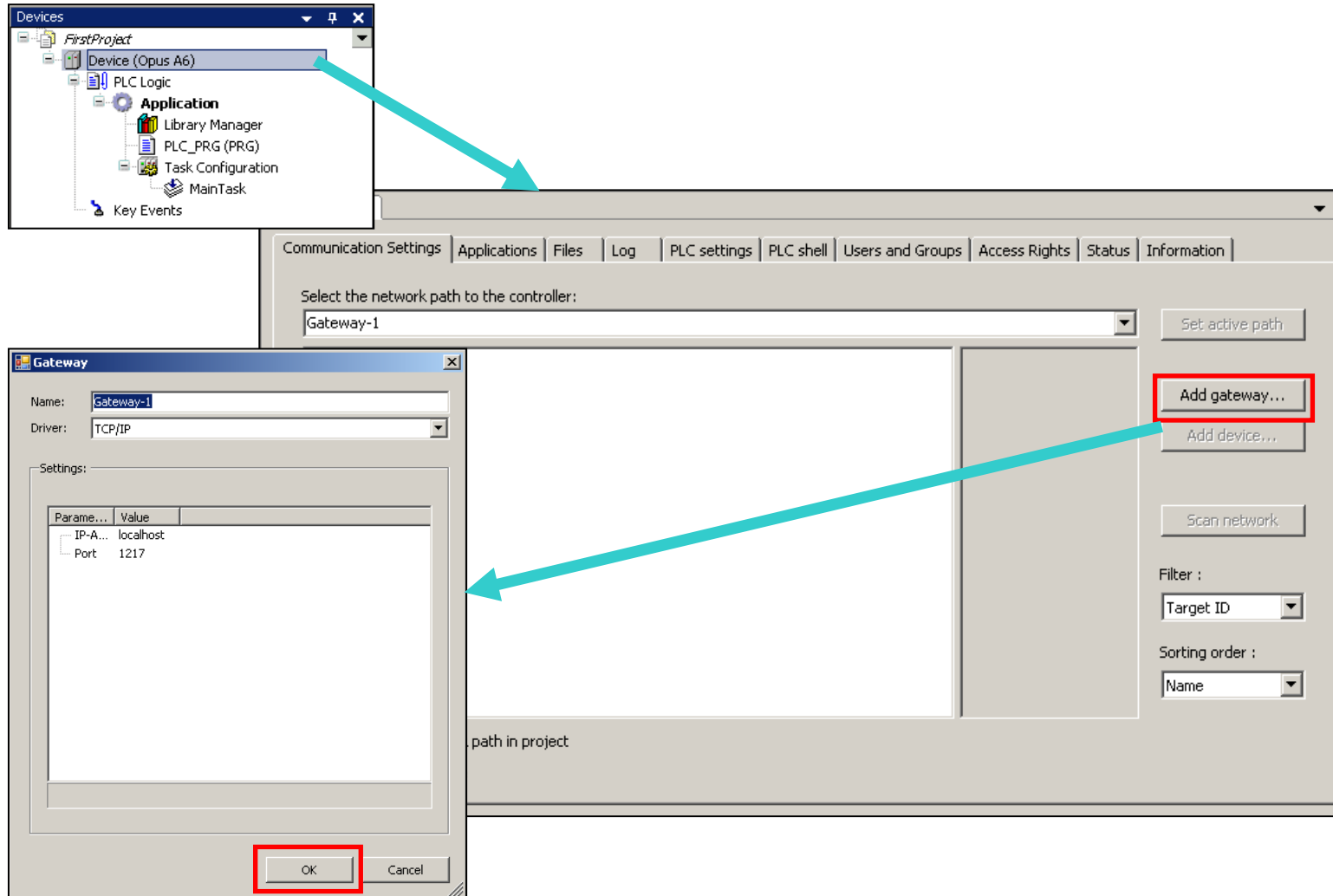


Go to the menu *Tools* -> *Options...*



Go to the category *Device editor*  
Make the settings according to the screenshot

## 4.2 The first project – Configuration – Device connection



Double click on *Device*

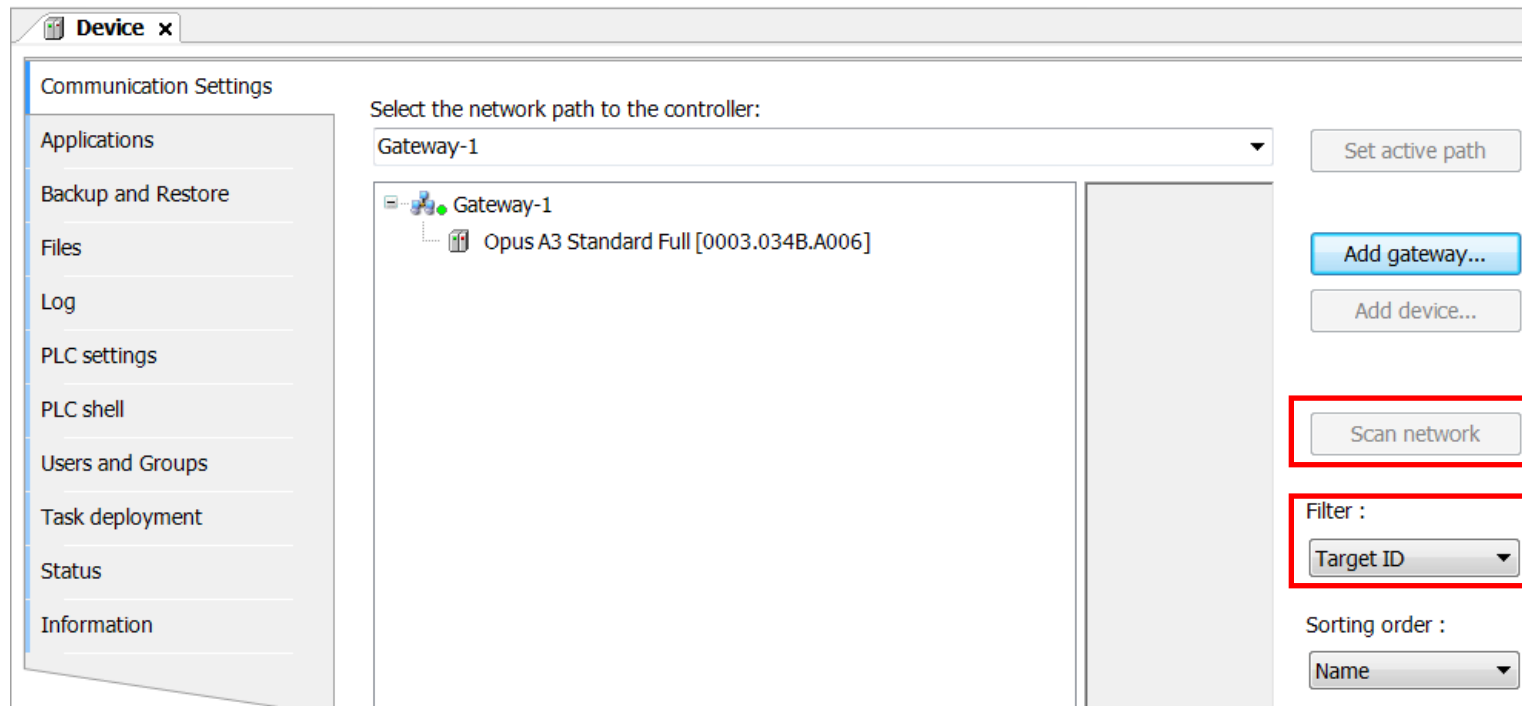
Click on *Add gateway...*

Click *OK*

Or select the existing Gateway



## 4.2 The first project – Configuration – Device connection



Select the gateway and click on *Scan network*

If the device isn't found, set the *Filter* to *None*

## 4.2 The first project – Configuration – Device connection

Device x

Communication Settings

Applications

Backup and Restore

Files

Log

PLC settings

PLC shell

Users and Groups

Task deployment

Status

Information

Select the network path to the controller:

Gateway-1:0003.034B.A006

Gateway-1

Opus A3 Standard Full [0003.034B.A006]

**Device Name:**  
Opus A3 Standard Full

**Device Address:**  
0003.034B.A006

**Target ID:**  
103F 0001

**Target Name:**  
Opus A3 Standard Full

**Target Type:**  
4096

**Target Vendor:**  
Wachendorff  
Elektronik GmbH &  
Co. KG

Set active path

Add gateway...

Add device...

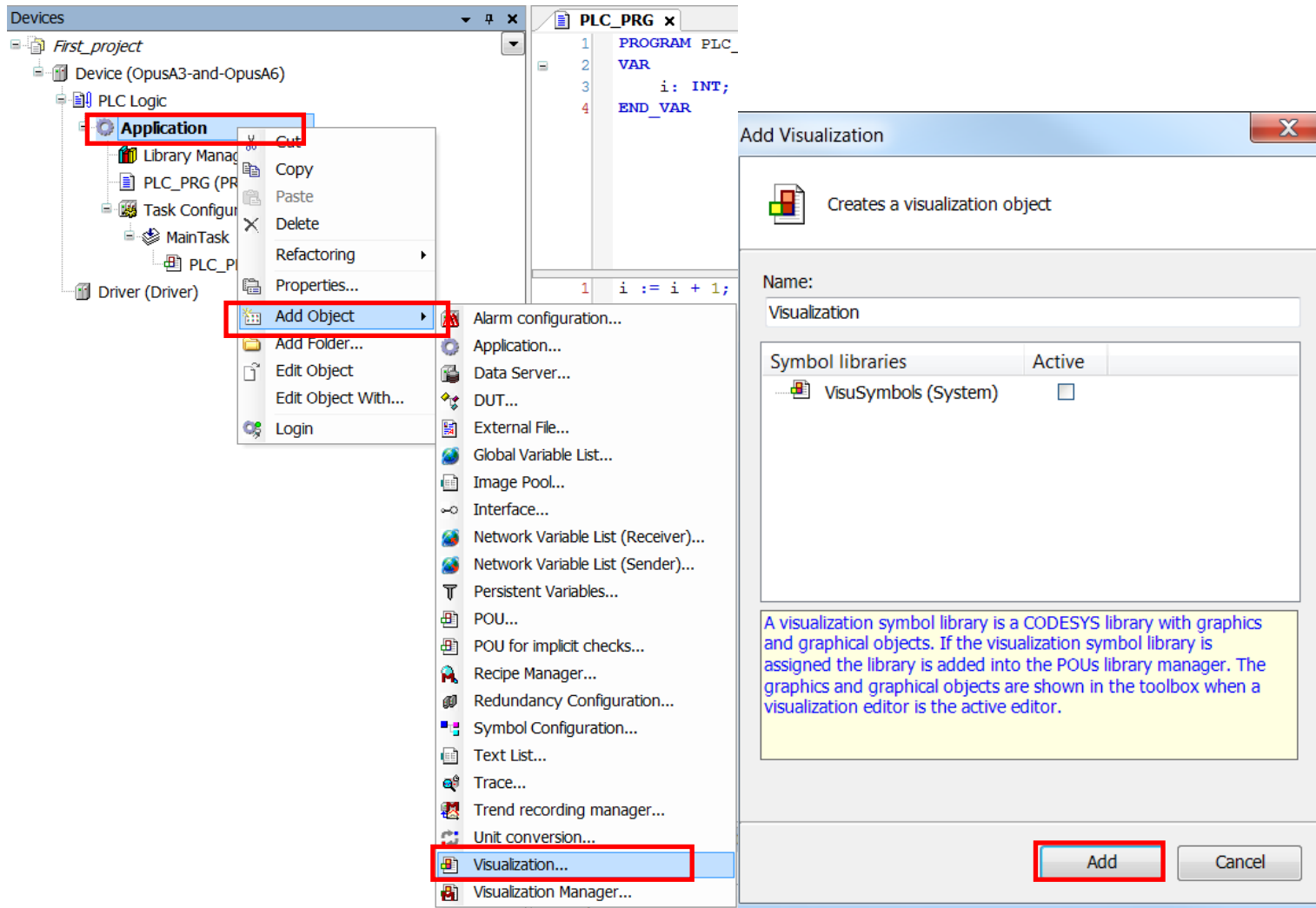
Scan network

Filter :  
Target ID

Sorting order :  
Name

Double click on the device in the list or select it and click on *Set active path* to connect to the device

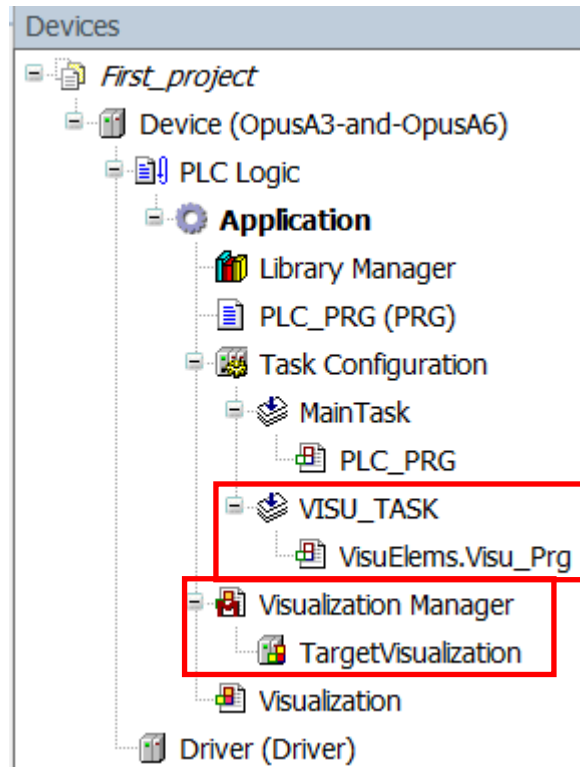
## 4.4 The first project – The first visualization



Right-click on Application and choose Add Object -> Visualization...

Click Add in the dialog (no settings necessary)

## 4.4 The first project – The first visualization

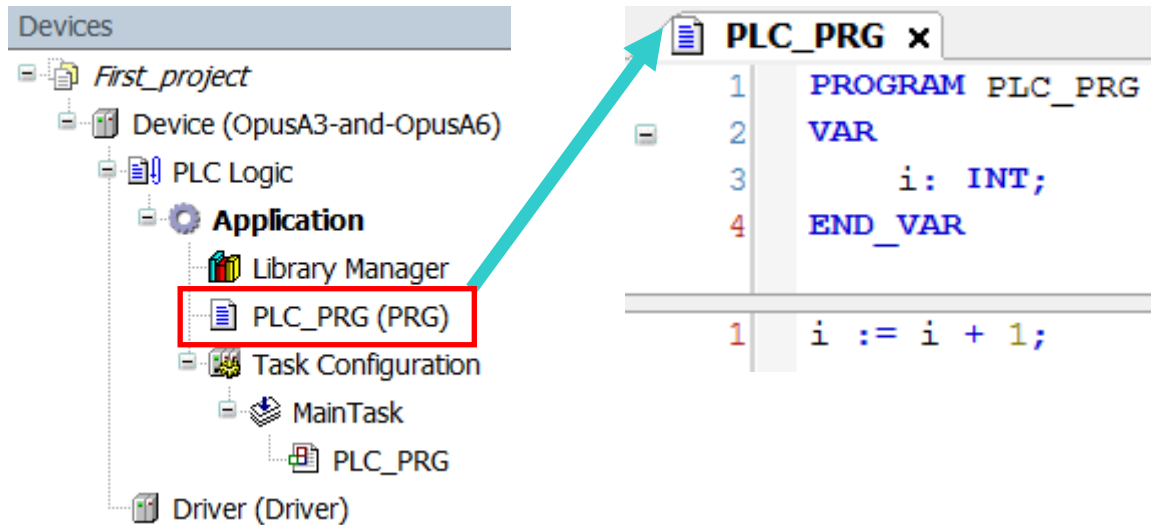


We just added a Visualization element

CODESYS adds some components automatically:

- VISU\_TASK in the Task Configuration
- Visualization Manager
- TargetVisualization in the Visualization Manager

## 4.3 The first project – The first program



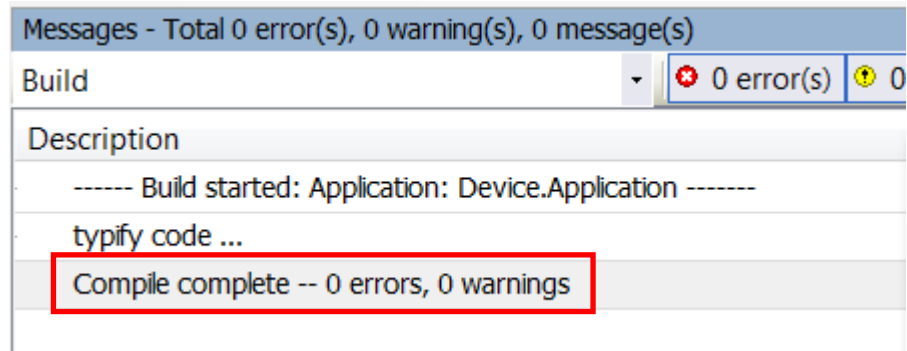
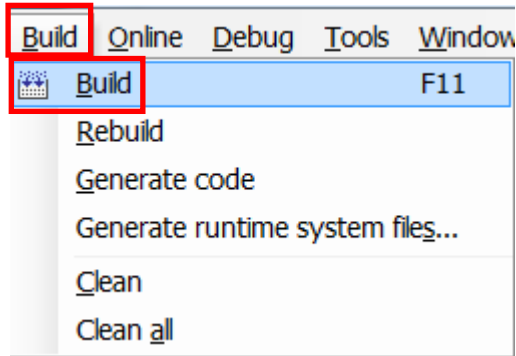
The screenshot displays the software interface for creating a PLC program. On the left, a tree view under 'Devices' shows the project structure: 'First\_project' contains 'Device (OpusA3-and-OpusA6)', which includes 'PLC Logic'. Under 'PLC Logic', there is an 'Application' folder containing 'Library Manager', 'PLC\_PRG (PRG)' (highlighted with a red box), 'Task Configuration', and 'MainTask'. 'MainTask' contains a 'PLC\_PRG' program. A green arrow points from the 'PLC\_PRG (PRG)' item in the tree to the right-hand window. The right-hand window shows the code for the 'PLC\_PRG' program:

```
1 PROGRAM PLC_PRG
2 VAR
3     i: INT;
4 END_VAR

1 i := i + 1;
```

Double click the program PLC\_PRG and write the variable declaration and code according to the screenshot

## 4.3 The first project – The first program – Building the code

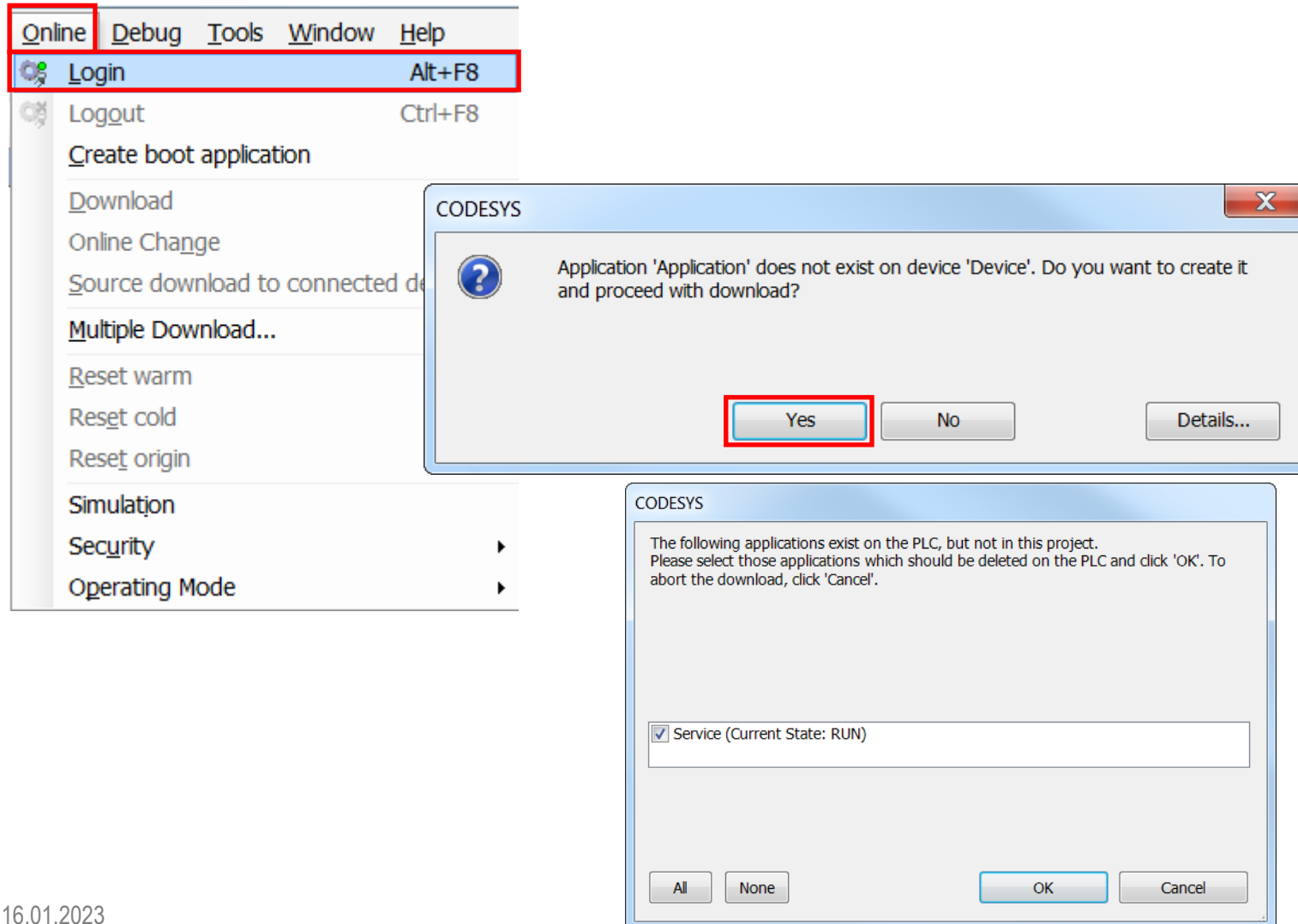



Compile the project by clicking in the menu *Build -> Build* or by pressing *F11*

Message box should show the message  
Compile complete – 0 errors, 0 warnings

Our first program built successfully, big applause

## 4.3 The first project – The first program – The first download

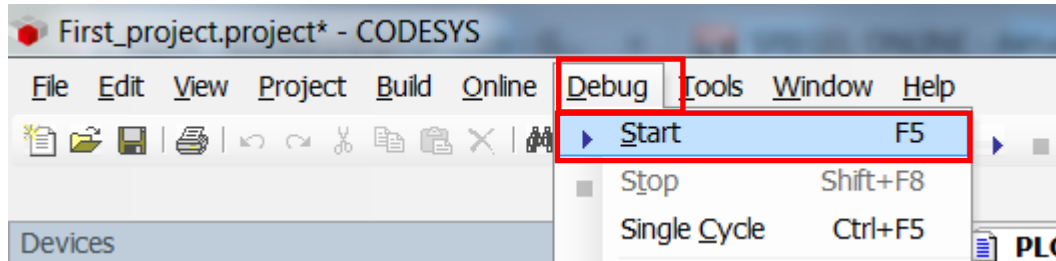



Download the project with *Login* in the menu *Online* (or by pressing Alt+F8) or press the  icon in the tool bar

Press Yes to download the application to the display

Press OK to overwrite the service project

## 4.3 The first project – The first program – The first download




Start the project with *Start* in the menu *Debug* or by pressing *F5* or clicking the  icon in the toolbar

-> empty visualization (white screen)

->you can see the counter rising on the PC

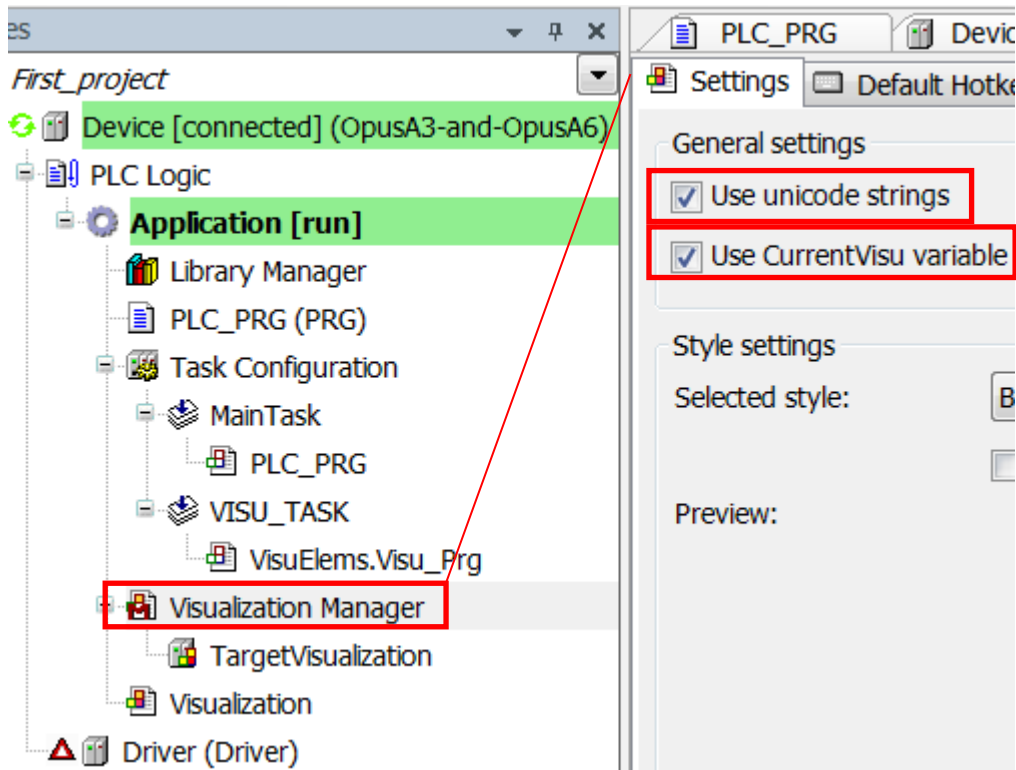
-you can manipulate the value in the column *prepared value*

-send the value with CTRL + F7

-log out of the device with CTRL + F8 or by clicking  in the toolbar to continue working on the project



## 4.4 The first project – The first visualization – Settings



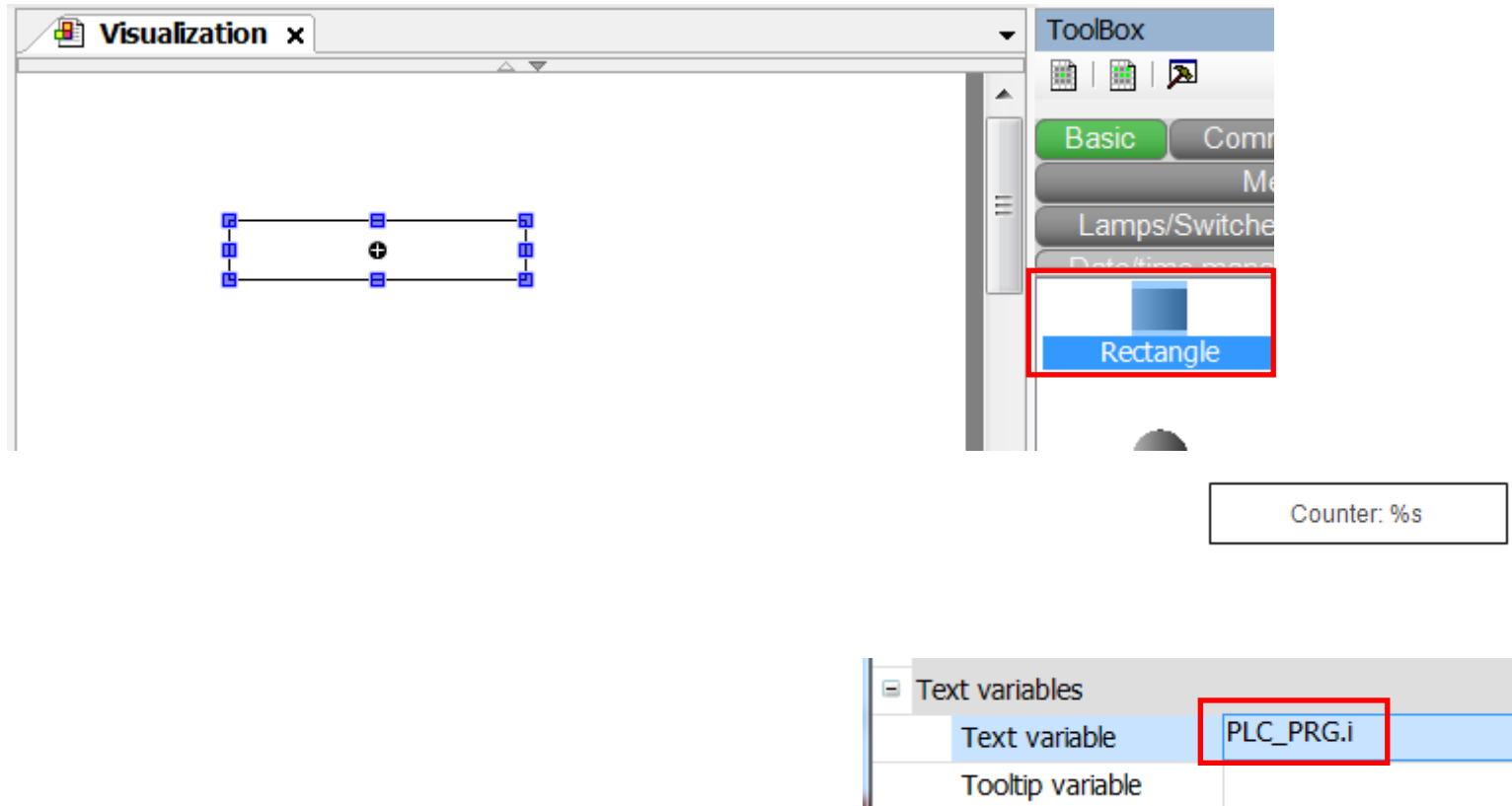
Make the settings according to the screenshots

Visualization Manager:

- Use unicode strings
- Use CurrentVisu variable

These are general recommendations for projects

## 4.4 The first project – The first visualization – Rectangle



Click on *Rectangle* in the toolbox and draw a rectangle in the visualization or drag & drop it from the toolbox into the visualization

Select the rectangle and add *Counter: %s* into *Texts* -> *Text* or by double-clicking the rectangle

Add the variable *PLC\_PRG.i* into *Text variables* -> *Text Variable*

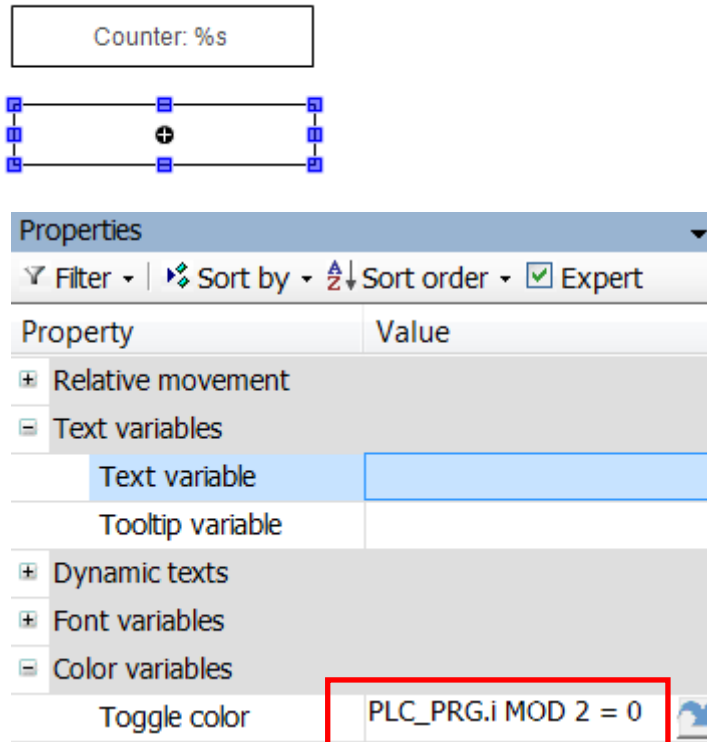
Log in again to update the application on the device, then start the application (F5)  
Log out afterwards to continue working

## 4.4 The first project – The first visualization – Rectangle

### Sample list of strings and arguments

Character after "%"	Argument/Output as
d, i	Decimal number
b	Binary number
o	Unsigned octal number (without leading zero)
x	Unsigned hexadecimal number (without leading 0x)
u	Unsigned decimal number
c	Single character
s	String: this location in online mode will be replaced by the value of the variable which is specified in the 'Text variables' property 'Text variable'.
f	REAL values Syntax: % <alignment><minimal width><accuracy>] The alignment is defined by a minus-sign (left aligned) or a plus-sign (right aligned, default); accuracy defines the number of places behind the comma (default: 6); see example below.

## 4.5 The first project – Tasks and timings



The screenshot shows a graphical user interface for a counter object. At the top, there is a rectangular box labeled "Counter: %s". Below it is a diagram of a counter mechanism with a central circle containing a plus sign, connected to various points by lines. Below the diagram is a "Properties" panel. The panel has a search bar with "Filter", "Sort by", "Sort order", and "Expert" options. Below this is a table with two columns: "Property" and "Value". The table is expanded to show several categories: "Relative movement", "Text variables", "Dynamic texts", "Font variables", and "Color variables". Under "Text variables", there is a row for "Text variable" with an empty value field. Under "Color variables", there is a row for "Toggle color" with the value "PLC\_PRG.i MOD 2 = 0" entered in the "Value" column. This value is highlighted with a red rectangular box.

Property	Value
Relative movement	
Text variables	
Text variable	
Tooltip variable	
Dynamic texts	
Font variables	
Color variables	
Toggle color	PLC_PRG.i MOD 2 = 0

Add a second rectangle

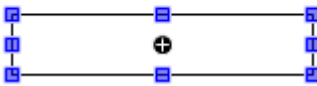
Add the expression  
 $PLC\_PRG.i \text{ MOD } 2 = 0$   
to the  
*Color Variables -> Toggle color* property

Download and start the application

Please note the flashing screen.

## 4.5 The first project – Tasks and timings

Counter: %s



Properties

Filter | Sort by | Sort order | Expert

Property	Value
Relative movement	
Text variables	
Text variable	
Tooltip variable	
Dynamic texts	
Font variables	
Color variables	
Toggle color	PLC_PRG.i MOD 2 = 0

First\_project

- Device (OpusA3-and-OpusA6)
  - PLC Logic
    - Application
      - Library Manager
      - PLC\_PRG (PRG)
      - Task Configuration
        - MainTask

MainTask x

Configuration

Priority ( 0..31 ): 1

Type: Cyclic

Interval (e.g. t#200ms): t#20ms

Add a second rectangle

Add the expression  
 $PLC\_PRG.i \text{ MOD } 2 = 0$   
 to the

*Color Variables* -> *Toggle color* property

Download and start the application

Please note the flashing screen.

Double-click on the *MainTask* and on the *VISU\_TASK*

Change the interval of the *MainTask* from 20 ms to 25 ms and download again

# 4.5 The first project – Tasks and timings

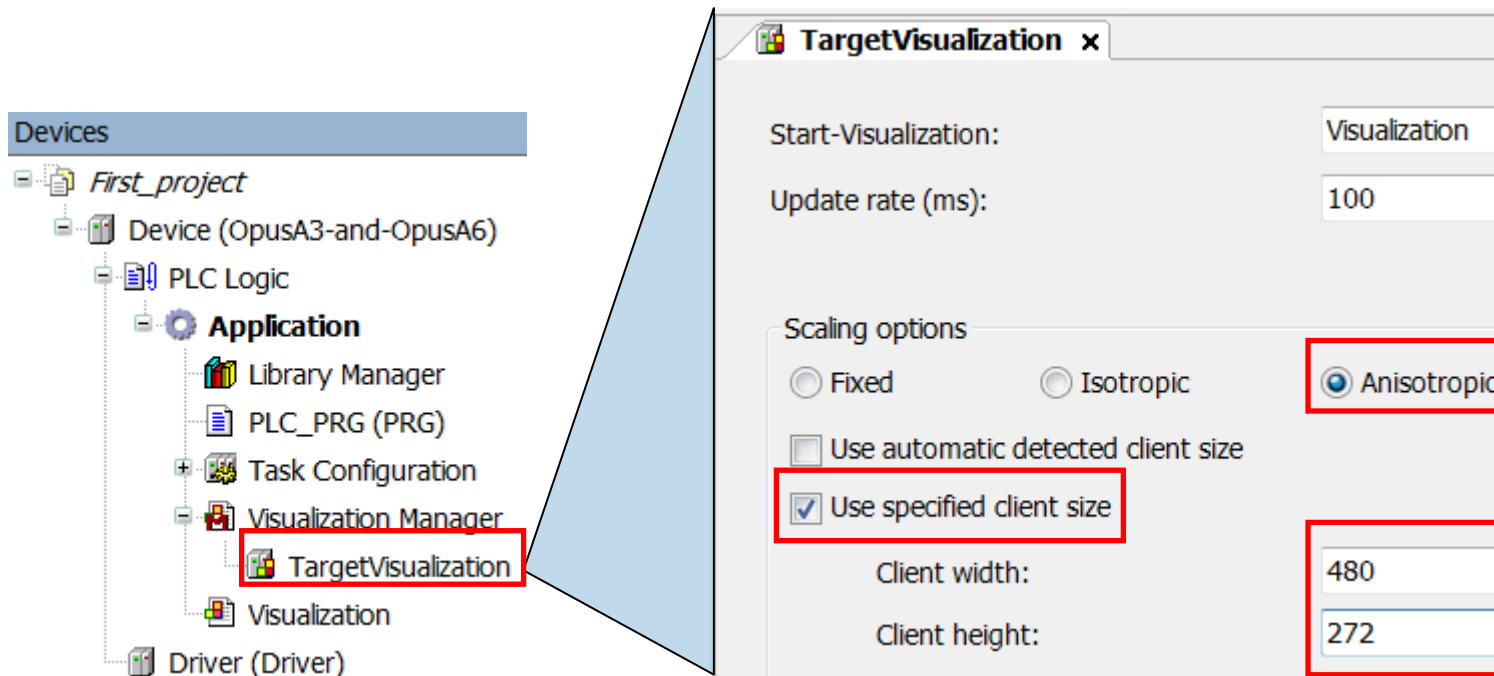


-> Take care of your timings!

Time (ms)	0	20	40	60	80	100	120	140	160	180	200	220	240	260	280	300
MainTask (20 ms)	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
VISU_TASK(100 ms)	0					1					0					1

Time (ms)	0	25	50	75	100	125	150	175	200	225	250	275	300	325	350	375
MainTask (25 ms)	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
VISU_TASK(100 ms)	0				0				0				0			

## 4.6 The first project – Fit the screen



To know how big your screen is in the editor  
make the settings according to the  
screenshot

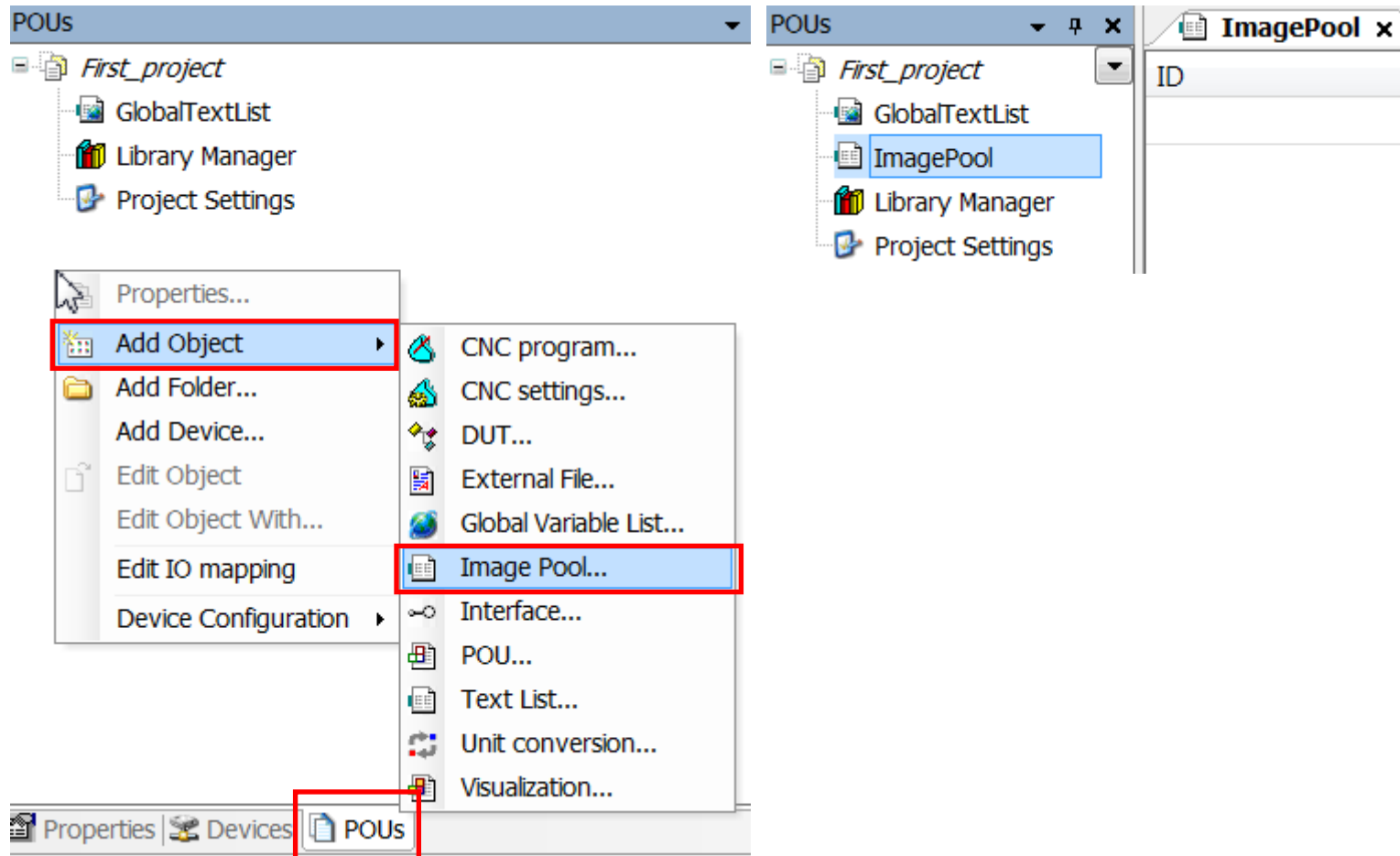
-> Then change the scaling back to *Fixed*

OPUS A3: 480 x 272

OPUS A6 G1 /G2: 800 x 480

OPUS A8: 1280 x 800

## 4.7 The first project – Images – Static images



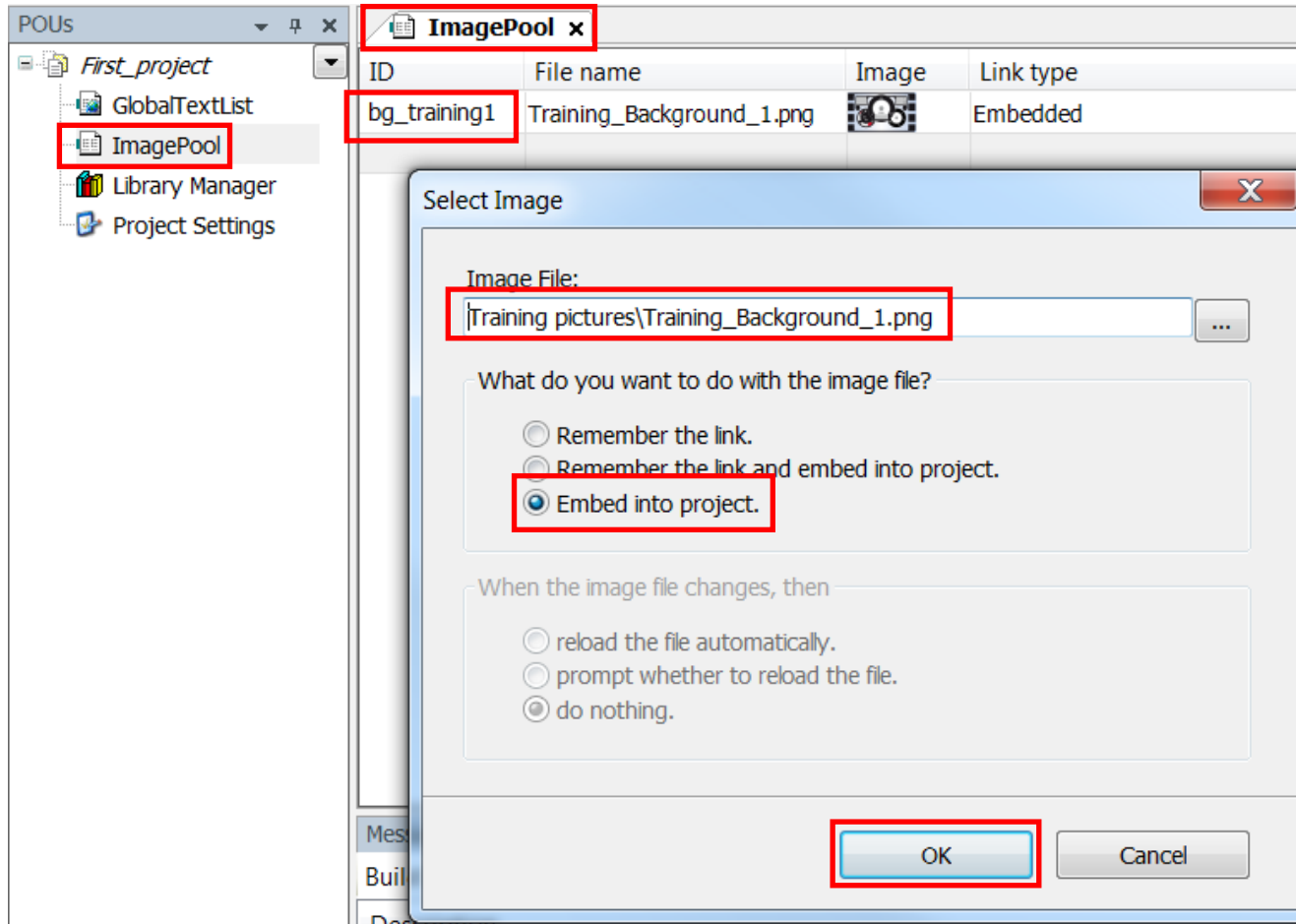
Change to the POU tab

Right-click and select  
*Add Object -> Image Pool...*

Double-click the ImagePool to open it



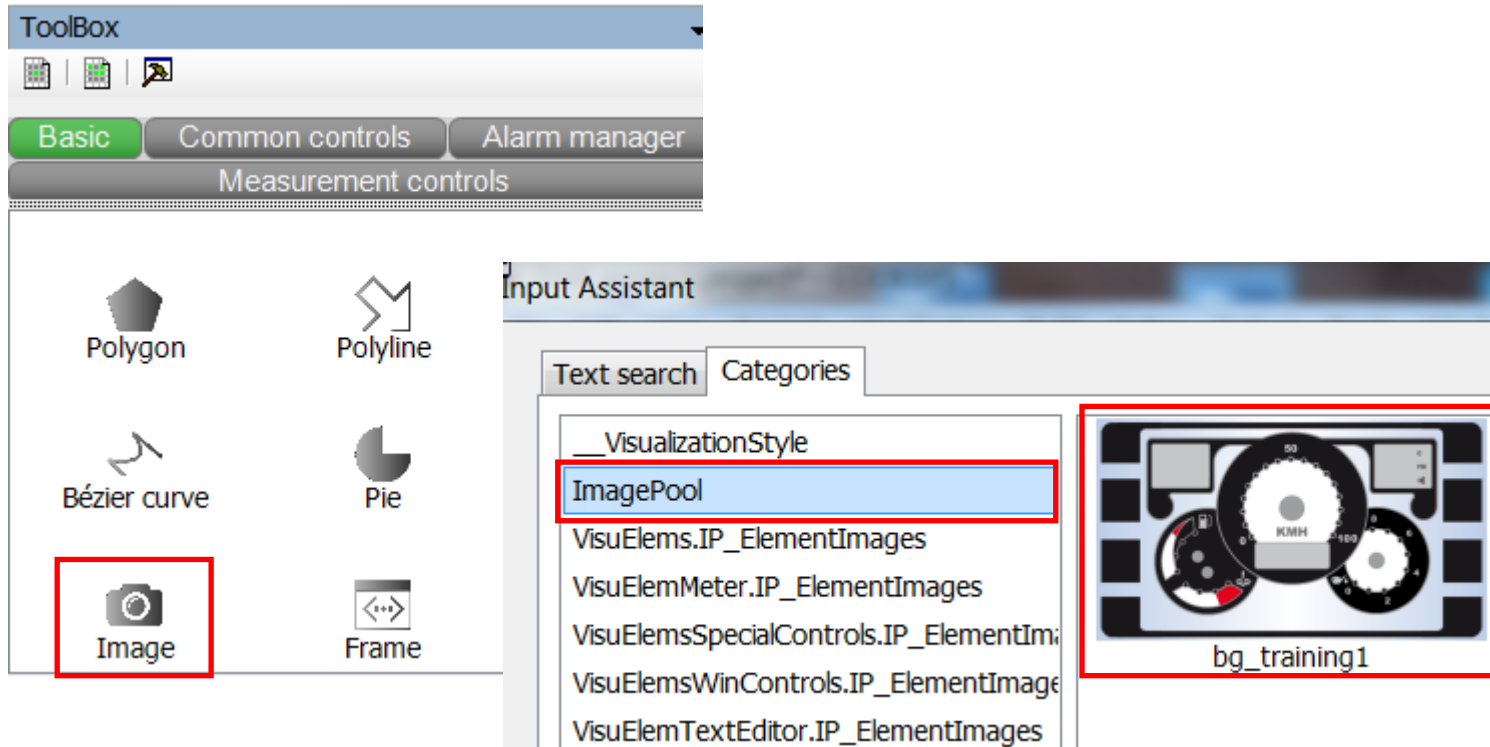
## 4.7 The first project – Images – Static images



Write the ID *bg\_training1*

Select the image  
*Training\_Background\_1.png* from the  
training pictures folder

## 4.7 The first project – Images – Static images



Click on *Image* and draw an image or drag & drop an image into the visu

In the image selection dialog select the image `bg_training1` and click OK

## 4.7 The first project – Images – Static images

The screenshot displays the software interface with the Properties panel on the left and a context menu open over a gear image in the Visualization window on the right.

**Properties Panel:**

Property	Value
Elementname	GenElemInst_13
Type of element	Image
Static ID	ImagePool.bg_training1
Show frame	<input type="checkbox"/>
Clipping	<input type="checkbox"/>
Transparent	<input type="checkbox"/>
Transparent color	Black
Scaling type	Anisotropic
<b>Position</b>	
X	0
Y	0
Width	480
Height	272
<b>Center</b>	
<b>Colors</b>	

**Context Menu:**

- Cut
- Copy
- Paste
- Delete
- Select All
- Create Global Text List
- Order**
  - Bring to Front
  - Bring One to Front
  - Send to Back**
  - Send One to Back
- Alignment
  - Bring One to Front
- Group
- Ungroup
- Frame Selection
- Background
- Multiply visu element




the ID is set in the Static ID property

Set the size and position properties

Right-click the image and select  
*Order -> Send to Back*


Delete the two rectangles

## 4.7 The first project – Images – Dynamic images

ID	File name	Image	Link type
bg_training1	Training_Background_1.png		Embedded
plus	BUTTON_PLUS_X.png		Embedded
minus	BUTTON_MINUS_X.png		Embedded

```

PROGRAM PLC_PRG
VAR
  i:INT;
  plusminus: STRING := 'plus';
END_VAR
    
```

Font color	 Black
Bitmap ID variable	
Bitmap ID	PLC_PRG.plusminus
Absolute movement	

Put the images *BUTTON\_PLUS\_X.png* and *BUTTON\_MINUS\_X.png* in the global image pool

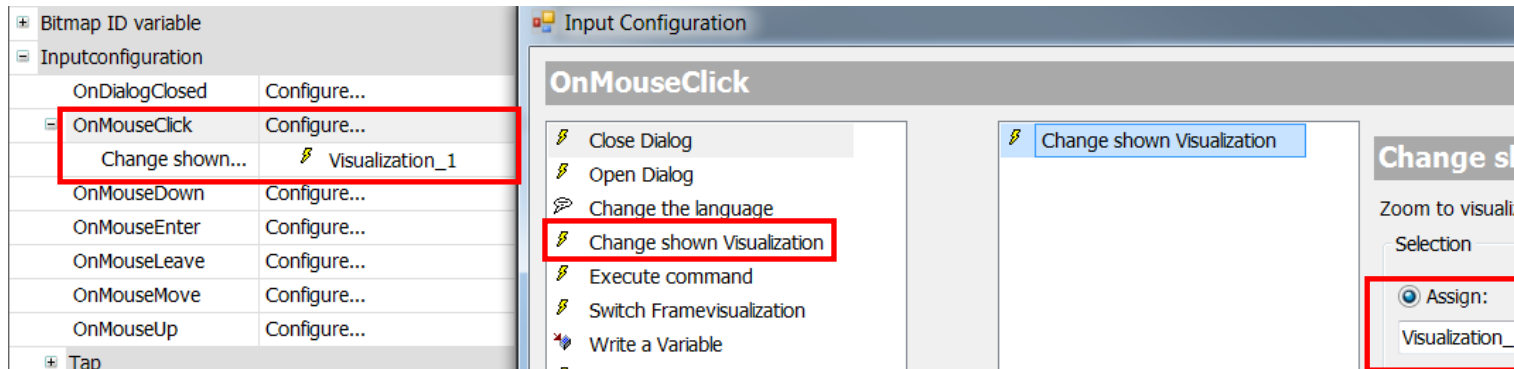
Create another image object  
Choose the “plus” image in the dialog

Initialize a string variable in *PLC\_PRG*

Set the Bitmap ID Variable to this string variable

Transfer the project and set the variable on the PC (from ‘plus’ to ‘minus’)

## 4.8 The first project – Buttons and navigation – A second visualization



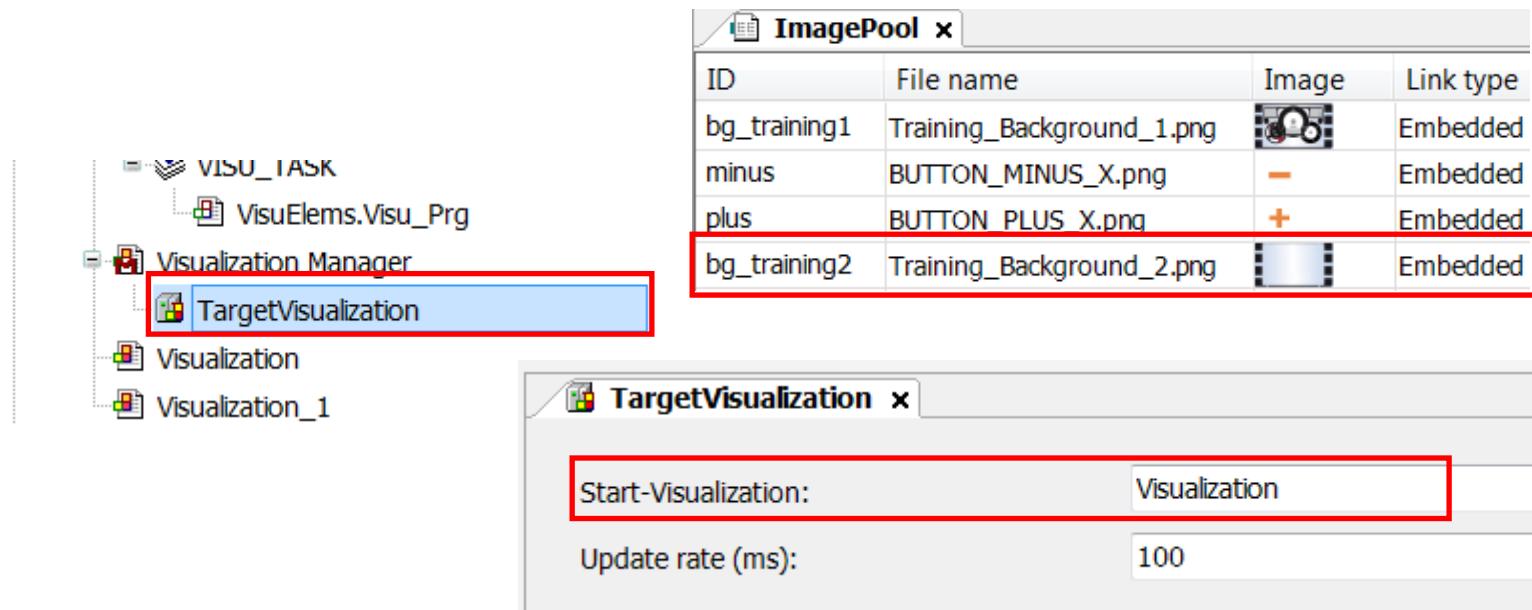
Create another visualization  
(right-click on Application -> Add Object -> Visu)  
Create a button from the Toolbox ->  
Common Controls in the first visualization

Configure the button to change to this  
visualization

Add the image Training\_Background\_2.png  
to the *Global Image Pool*

Create an image in the new visu and use  
the image as a background

Start visualization? Set it in  
*TargetVisualization*



## 4.8 The first project – Buttons and navigation – Push button or toggle

Y	9
Width	41
Height	51
Variable	PLC_PRG.button_switch




Image settings	
Isotropictype	Isotropic
Horizontal alignm...	Left
Vertical alignment	Top
Element behavior	Image tapper
Tap FALSE	Image toggler
Texts	Image tapper

```
PROGRAM PLC_PRG
VAR
    i: INT;
    button_switch: BOOL;
END_VAR
```

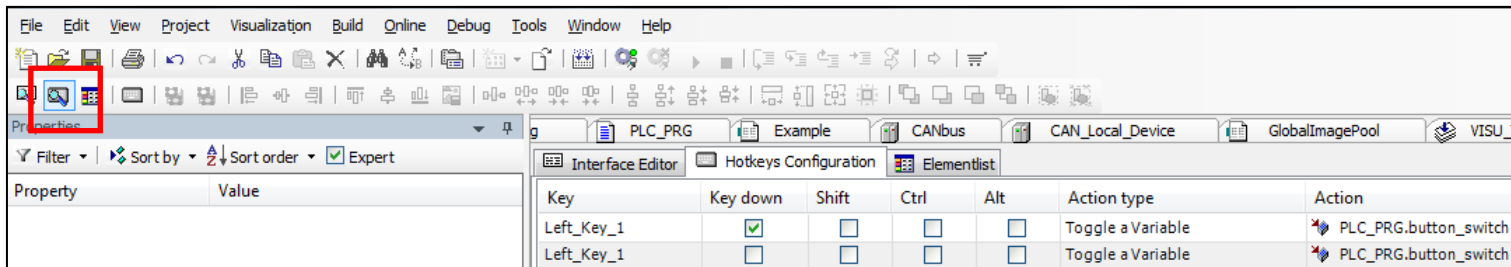
In the new visualization, create a button (push switch, dip switch etc.)

Also create a Lamp object and in *PLC\_PRG* a variable *button\_switch* (bool) and connect it with the button and the LED

Test the behavior

You can change from *Toggle* to *Tap* in the property *Element behavior* of the button object

Add the same functionality for a soft key



Key	Key down	Shift	Ctrl	Alt	Action type	Action
Left_Key_1	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Toggle a Variable	PLC_PRG.button_switch
Left_Key_1	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Toggle a Variable	PLC_PRG.button_switch

## 4.8 The first project – Buttons and navigation – Button with 2 images

Position	
X	0
Y	0
Width	68
Height	68

back_1	C:\Users\cst\Documents\Training pictures\Codesys\BUTTON_BACK.p...	→
back_2	C:\Users\cst\Documents\Training pictures\Codesys\BUTTON_BACK_...	→

```

VAR
  i:INT;
  plusminus: STRING := 'plus';
  meterdata: INT;
  button_back: STRING := 'back_1';
END VAR
    
```

Bitmap ID variable	
Bitmap ID	PLC_PRG.button_back
Inputconfiguration	
OnDialogClosed	Configure...
OnMouseClicked	Configure...
Execute ST-Code	⚡ PLC_PRG.button_back := 'back_1';
Change shown Visualization	⚡ Visualization
OnMouseDown	Configure...
Execute ST-Code	⚡ PLC_PRG.button_back := 'back_2';

On the new visualization, create an image with these settings

Put the images *BUTTON\_BACK.png* and *BUTTON\_BACK\_X.png* in the global image pool

In the *PLC\_PRG*, create a variable *button\_back*

Configure the events for the “button”

## 5. CAN communication

1. Basic settings
2. Terminal and Owner ECU configuration
3. Creating messages
4. Background information





## 5.1a CAN communication

**CAN**open

## 5.1a CAN communication – Set up the CAN bus

# CANopen

The screenshot shows the 'Add Device' dialog box in the software. The 'Name' field is set to 'CANbus'. The 'Action' section has 'Append device' selected. The 'Device' section has 'Vendor' set to '3S - Smart Software Solutions GmbH'. A table at the bottom shows the selected device:

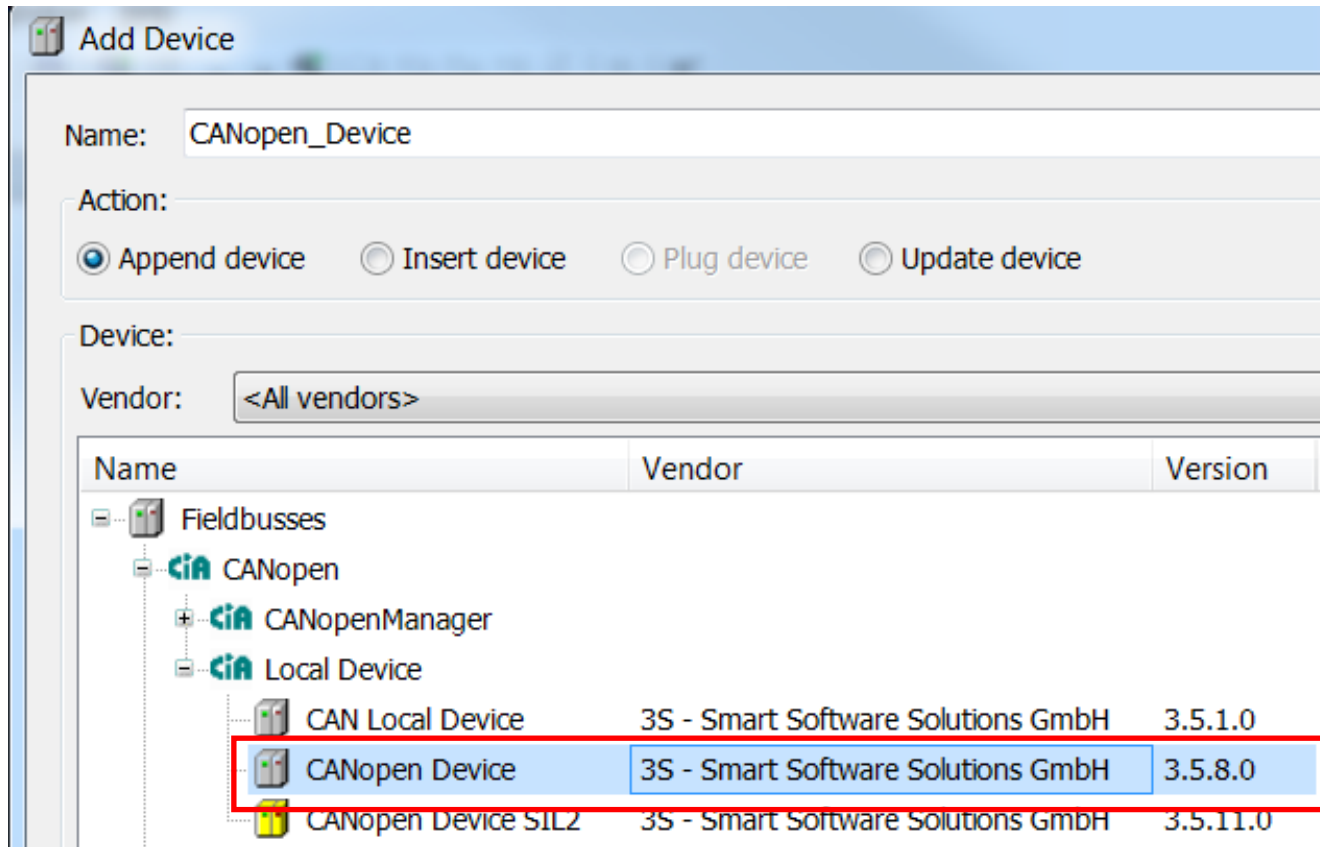
Name	Vendor	Version
CANbus	3S - Smart Software Solutions GmbH	3.5.7.0

Right click on *Device*, click on *Add Device...*, select 3S company as vendor

Select CANbus, edit the name if wanted

Press on *Add Device*

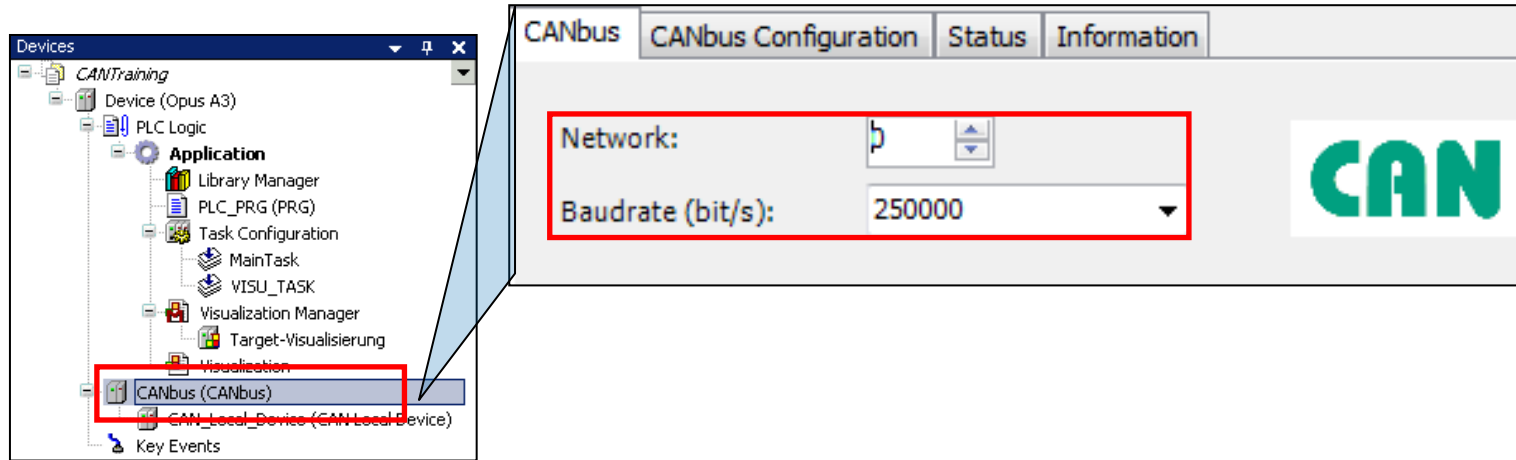
## 5.2a CAN communication – Add local device



Click on *CANbus* in the device tree, the dialog will change

Choose *CANopen Device*, version 3.5.8.0, click *Add Device* and close the dialog

# 5.2a CAN communication – Configuration



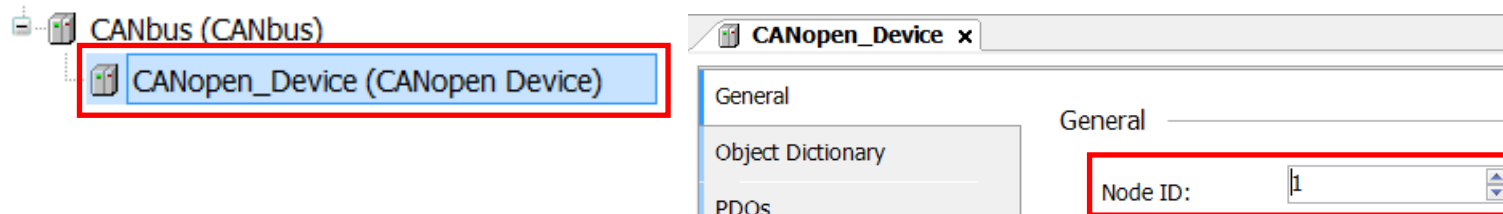
Double click on *CANbus*

Choose the network:

0 - CAN1

1 - CAN2

Choose the desired baudrate

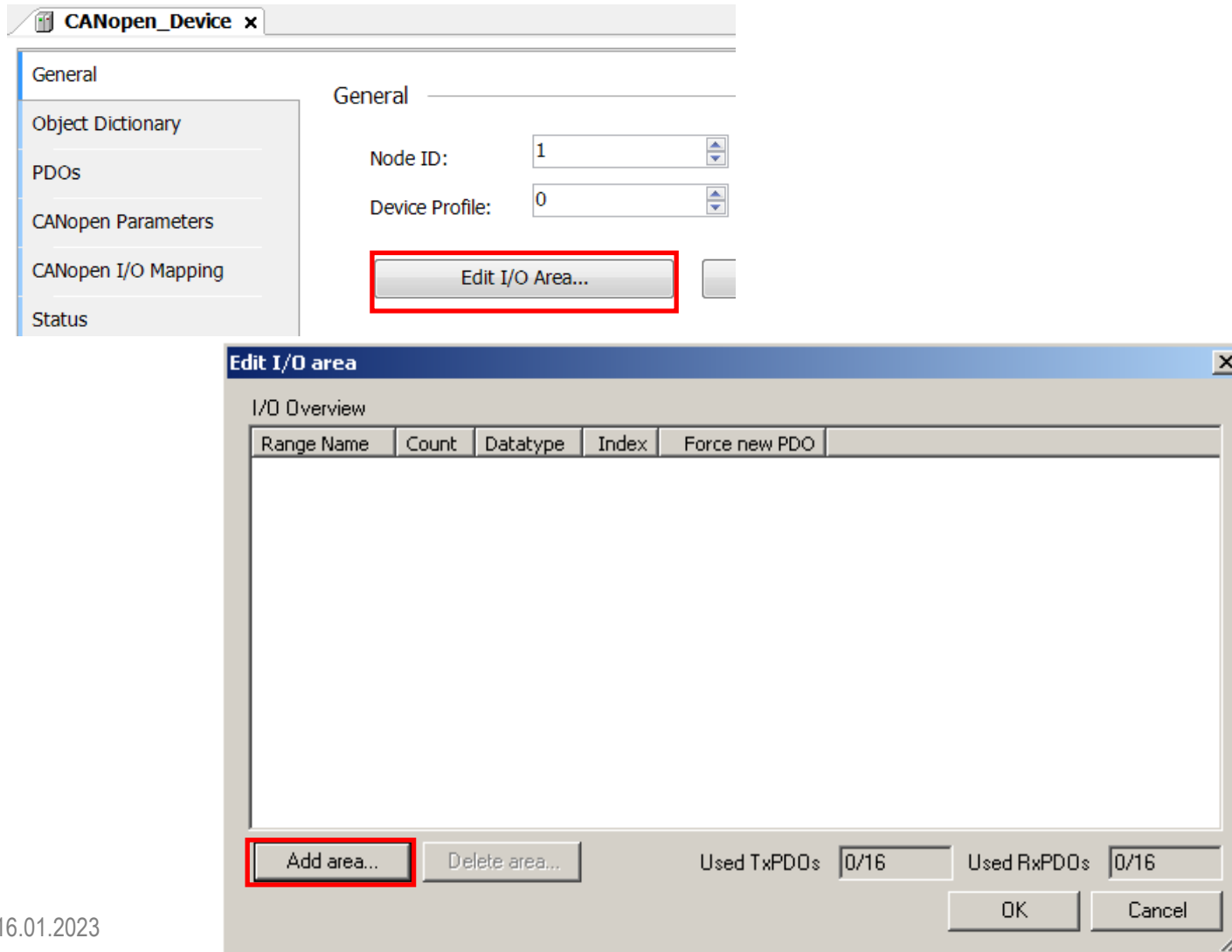


Double click on *CAN\_Local\_Device*

Choose the node ID

## 5.3a CAN communication – Add I/O areas

# CANopen



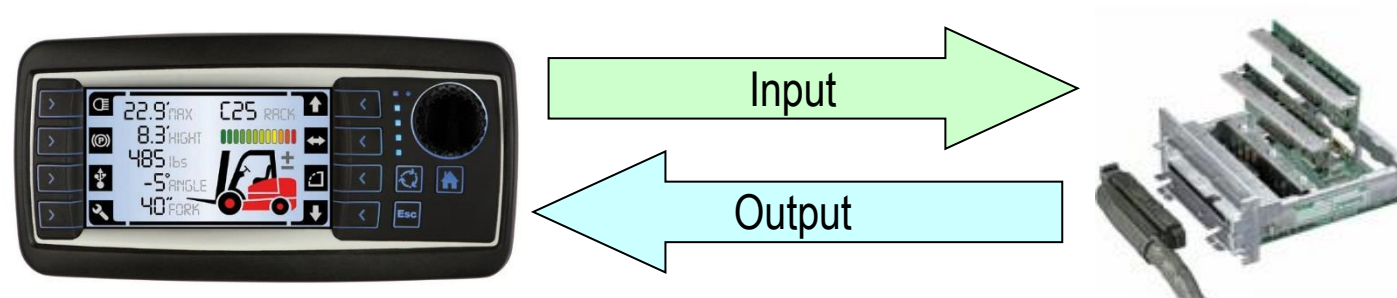
Click on *Edit I/O area...* to add the PDO communication variables

Click on *Add area...*

## 5.3a CAN communication – Add I/O areas

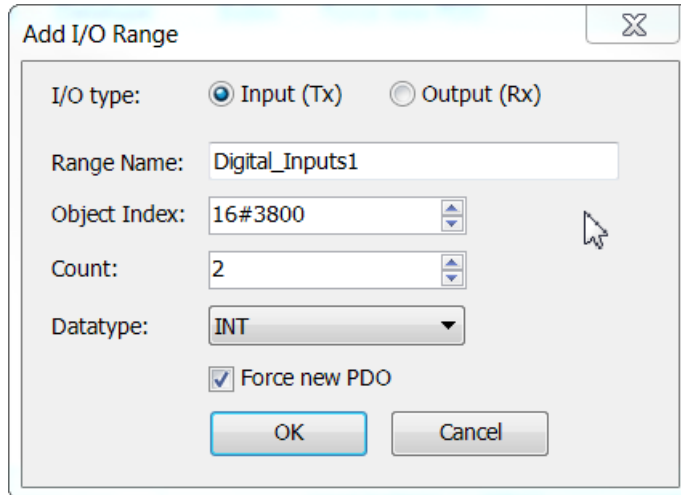


CODESYS always uses the viewpoint of the Master!



## 5.3a CAN communication – Add I/O areas

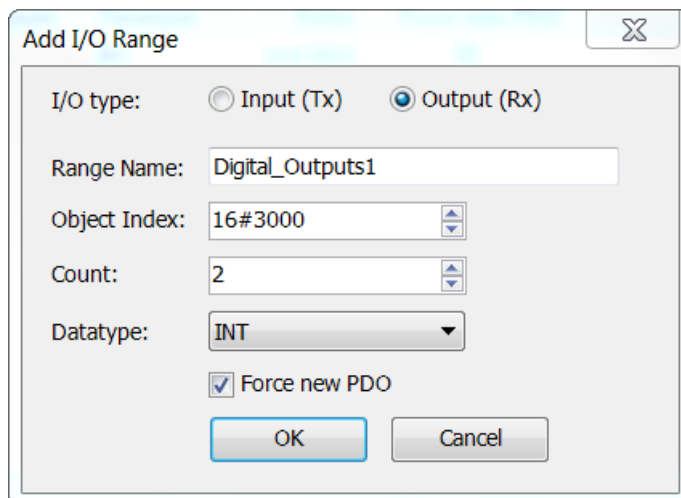
# CANopen



The screenshot shows the 'Add I/O Range' dialog box with the following settings:

- I/O type:  Input (Tx)  Output (Rx)
- Range Name: Digital\_Inputs1
- Object Index: 16#3800
- Count: 2
- Datatype: INT
- Force new PDO

Buttons: OK, Cancel



The screenshot shows the 'Add I/O Range' dialog box with the following settings:

- I/O type:  Input (Tx)  Output (Rx)
- Range Name: Digital\_Outputs1
- Object Index: 16#3000
- Count: 2
- Datatype: INT
- Force new PDO

Buttons: OK, Cancel

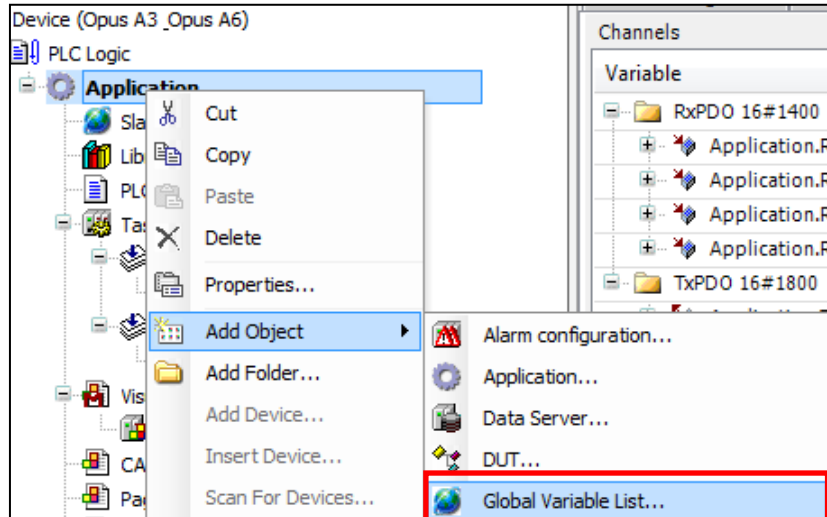
Add 2 INT Input variables for Transmit PDO  
(the OPUS sends these)

Click again and add 2 INT variables as  
output (the OPUS receives these)

Click on OK to close the dialog

## 5.3a CAN communication – Add variables

# CANopen



```

VAR_GLOBAL
  RPDO1_byte1_4 : UINT;
  RPDO1_byte5_8 : UINT;

  TPDO1_Byte1_4: INT := 10;
  TPDO1_Byte5_8: INT := 20;
END_VAR

```

```

1 GVL.TPDO1_Byte1_4 := GVL.TPDO1_Byte1_4 + 10;
2 GVL.TPDO1_Byte5_8 := GVL.TPDO1_Byte5_8;

```

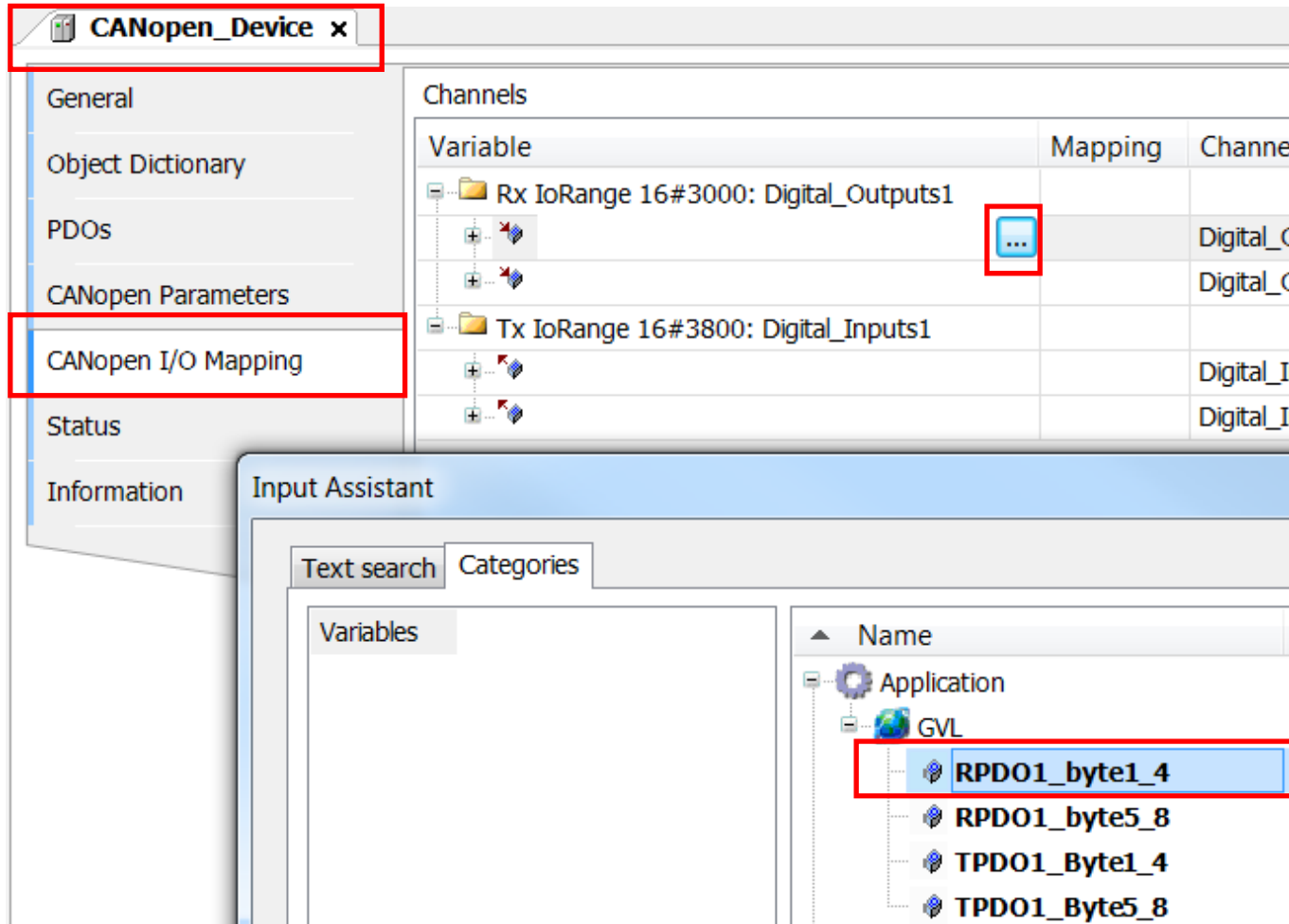
Create a global variable list (leave the name GVL)

Add the following variable declarations

Write the following program into the *PLC\_PRG*



## 5.3a CAN communication – Add variables



Double click on *CAN\_Local\_Device*

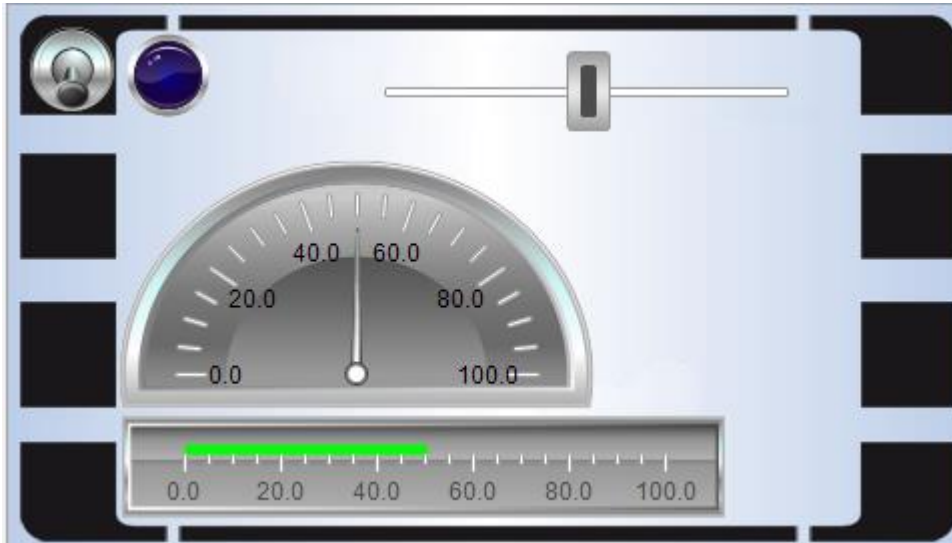
Switch to the tab *CANbus Slave I/O Mapping*

Map the declared variables to the CAN Input & Output:

Digital\_Outputs1:  
RPDO1 variables

Digital\_Inputs1:  
TPDO1 variables

## 5.3a CAN communication – Create Visu



Type of element	Slider
Position	
X	162
Y	23
Width	253
Height	40
Variable	GVL.TPDO1_Byte5_8

Receive / Transmit		Trace	PCAN-USB			
Receive	Message	DLC	Data	Cycle Time	Count	
	181h	4	CC FA 64 00	20	21917	
Transmit	Message	DLC	Data	Cyc...	Co...	Trigger
	000h	2	01 00	Wait	5	Manual
	201h	8	12 12 00 00 12 12 00	Wait	11	Manual

Create a meter, a bargraph and a slider object

Set the variable / value property to:

- TPDO1\_Byte5\_8 for the slider
- RPDO1\_Byte1\_4 for the meter
- RPDO1\_Byte5\_8 for the bargraph

Test the communication

-> no communication?

Send a message with

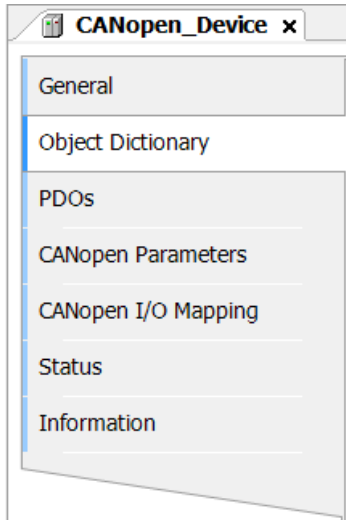
ID 0x000

DLC 2

Data 01 00

first to set the OPUS to operational mode

## 5.3a CAN communication – Sending cyclically



16#1800	TPDO communication parameter
16#1800:16#00	Highest sub-index supported
16#1800:16#01	COB-ID used by TPDO
16#1800:16#02	Transmission type
16#1800:16#03	Inhibit time
16#1800:16#04	compatibility entry
16#1800:16#05	Event timer

<b>Value</b>	
Default Value	\$NODEID+16#180
High Limit	16#FFFFFFFF
Low Limit	16#00000080

<b>Value</b>	
Default Value	16#0
High Limit	
Low Limit	

Set the *Event Timer* to a value > 0 to enable cyclic sending

-> Value has to be a multiple of the main task

Also you can change the COB ID (not CANopen conform!) by changing the value in *COBID*

Continue with chapter 6

## 5.1b CAN communication

 SAE J1939  
INTERNATIONAL™

## 5.1b CAN communication – Add CAN interface

The screenshot shows the 'Add Device' dialog box in the software. The 'Name' field is set to 'CANbus'. The 'Action' section has 'Append device' selected. The 'Device' section has 'Vendor' set to '3S - Smart Software Solutions GmbH'. Below the dialog, a table lists the available devices:

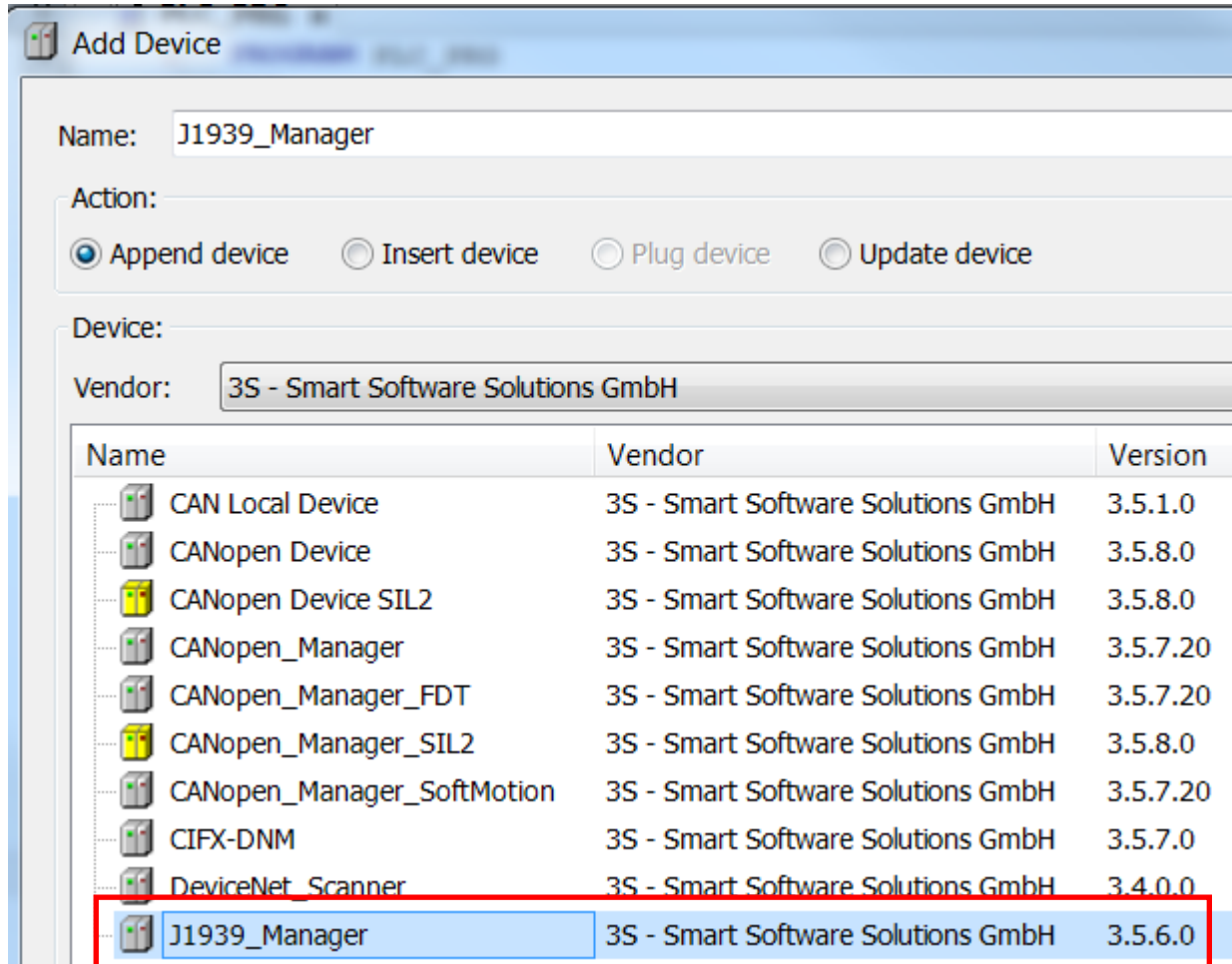
Name	Vendor	Version
CANbus	3S - Smart Software Solutions GmbH	3.5.7.0

Right click on *Device*, click on *Add Device...*, select 3S company as vendor

Select CANbus in selection, edit the name if wanted

Press on *Add Device*

## 5.2b CAN communication – Add J1939 manager



Click on *CANbus* in the device tree, the dialog will change

Choose *J1939\_Manager* and click *Add Device*

## 5.2b CAN communication – Add J1939 ECU



Name	Vendor	Version
J1939_ECU	3S - Smart Software Solutions GmbH	3.5.5.0

Click on *J1939\_Manager* in the device tree, the dialog will change

Choose *J1939\_ECU*, click on *Add Device* and then *Close*

## 5.3b CAN communication – Configuration

The image shows a software interface for configuring CAN communication. On the left, a project tree for 'First\_Project' is visible, with 'CANbus\_1 (CANbus)' and 'J1939\_ECU (J1939\_ECU)' highlighted in red. The main window is divided into two panes. The top pane, titled 'CANbus Configuration', shows 'Network:' set to '0' and 'Baudrate (bit/s):' set to '250000'. The bottom pane, titled 'J1939\_ECU x', shows the 'General' tab with 'Preferred Address:' set to '0' and 'Local Device' unchecked. Other tabs in the bottom pane include 'TX Signals', 'J1939 Parameters', 'J1939 I/O Mapping', 'Status', and 'Information'.

Doubleclick on *CANbus*

Choose the network:

- 0 - CAN1
- 1 - CAN2

Choose the baudrate (250 kbit/s is standard for J1939)

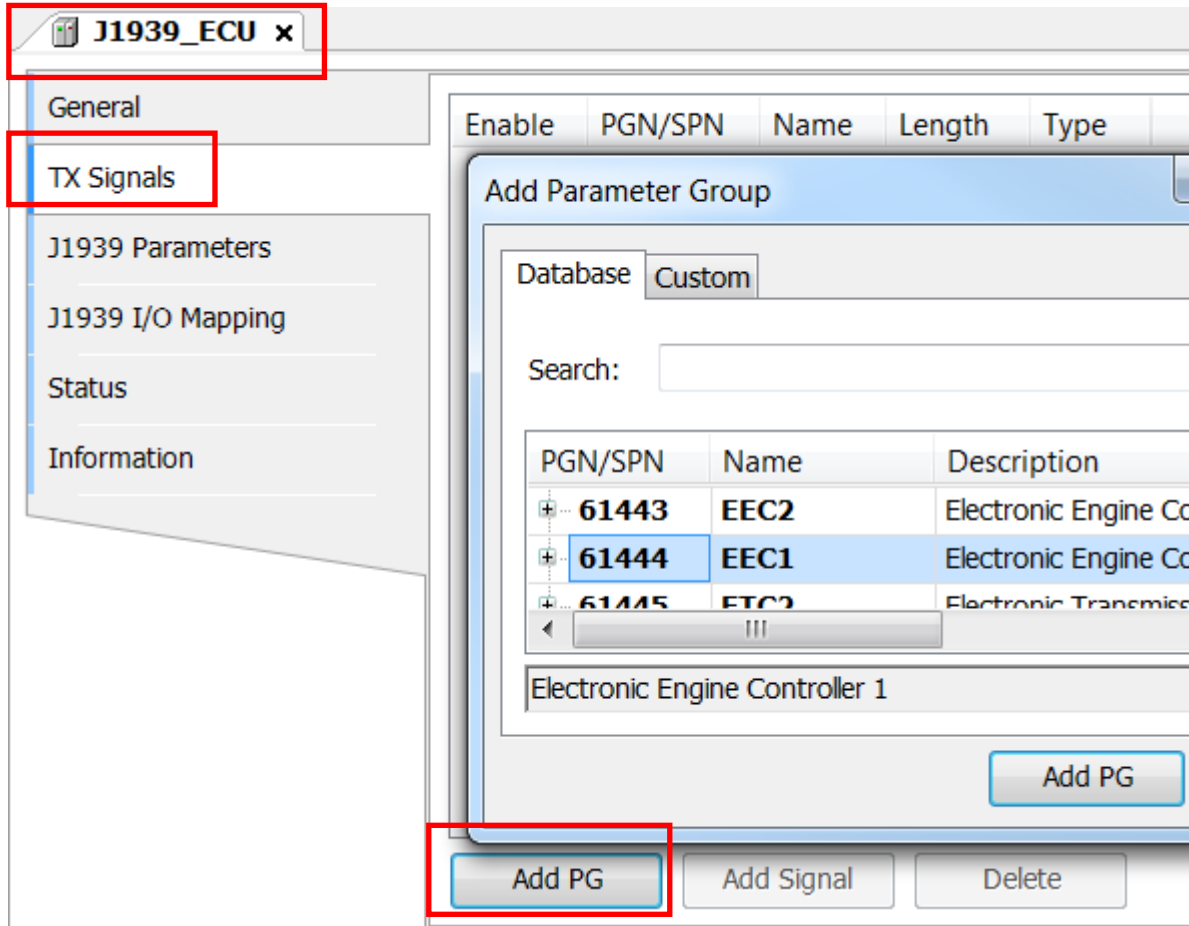
Doubleclick on J1939 ECU

Choose the preferred address

-> These are the settings for the J1939 ECU



## 5.3b CAN communication – Creating messages



Doubleclick on *J1939\_ECU*

Go to the tab *TX Signals*, click on *Add PG*

Choose PGN 61444 and click *Add PG* to close the dialog

## 5.3b CAN communication – Creating messages

```
VAR_GLOBAL
//Receive
Engine_Speed : UINT;

//Transmit
Engine_Requested_Speed_Speed_Limit : UINT;
END_VAR
```

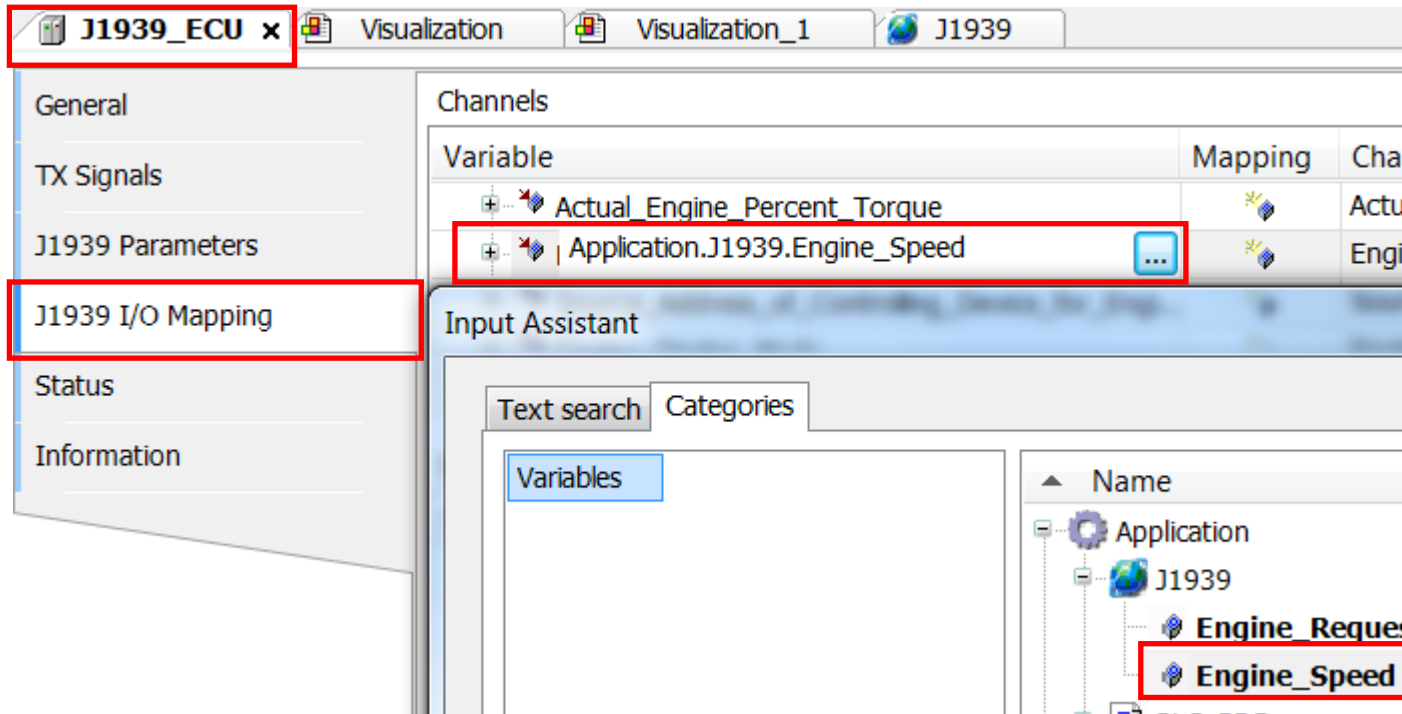
Create a Global Variable List by right-clicking *Application* -> *Add Object* -> *Global Variable List...*

Name it J1939

Click *Add* to close the dialog

Open the GVL J1939 and create these variables

## 5.3b CAN communication – Creating messages

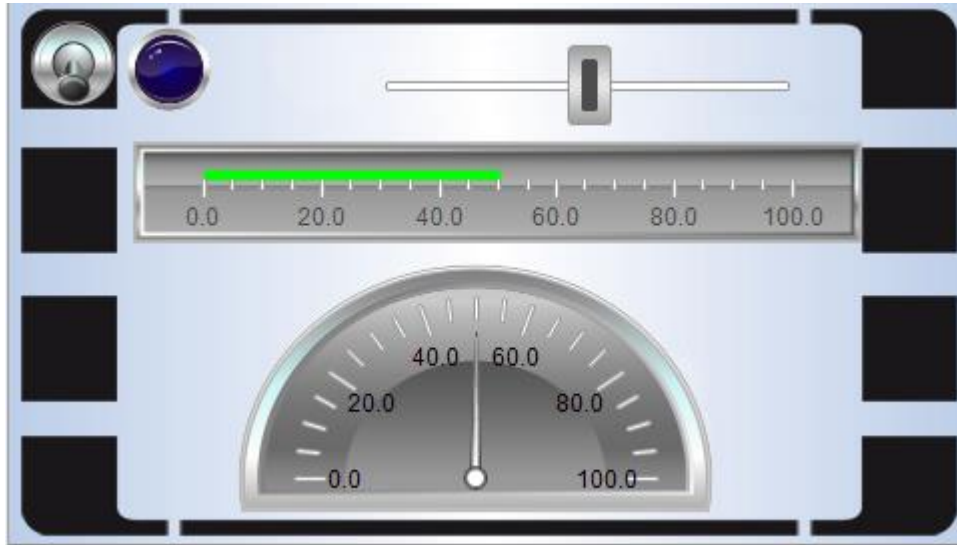


In J1939\_ECU Common J1939 I/O Mapping, map the variable *Application.J1939.Engine\_Speed* to Engine\_Speed

```
1 J1939.Engine_Speed := J1939.Engine_Speed;
```

In the *PLC\_PRG*, write this line

## 5.3b CAN communication – Creating messages

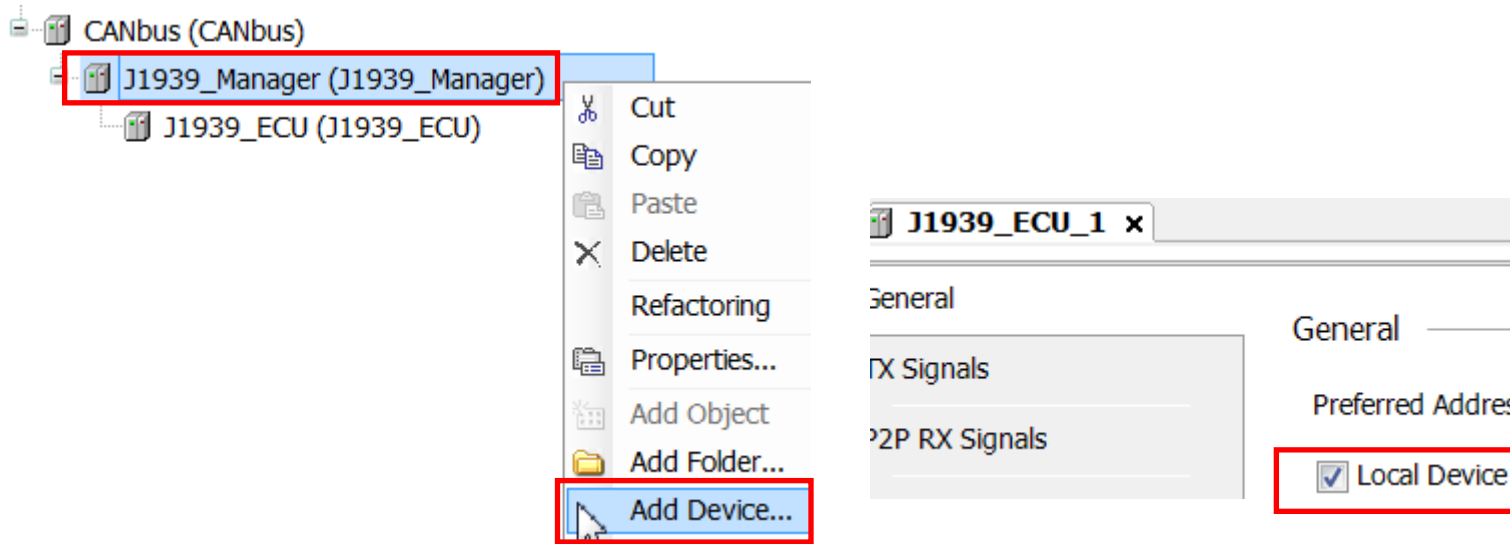


Create a meter, a bargraph and a slider object

Property	Value	Property	Value
Elementname	GenElemInst_2	Elementname	GenElemInst_3
Type of element	Meter 180°	Type of element	Bar display
Value	J1939.Engine_Speed	Value	J1939.Engine_Requested_Speed_Speed_Limit
		Elementname	GenElemInst_7
		Type of element	Slider
		Position	
		Variable	J1939.Engine_Requested_Speed_Speed_Limit
		Move to click	<input checked="" type="checkbox"/>

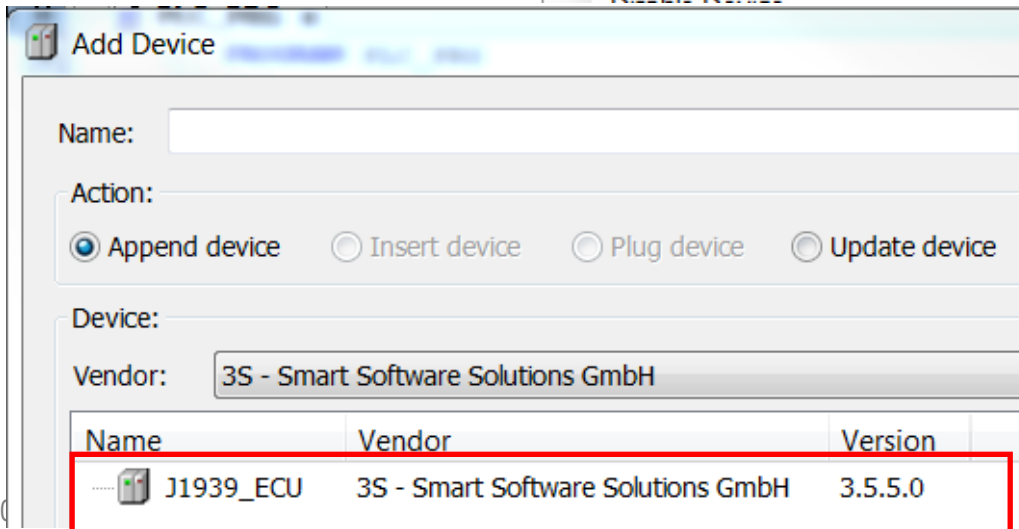
Connect the objects with the variables as shown

## 5.3b CAN communication – Sending PGNs



The previous J1939 configuration is only for receiving

To send, add another *J1939\_ECU*, double-click it and check *Local Device*



-> This device is only for sending!

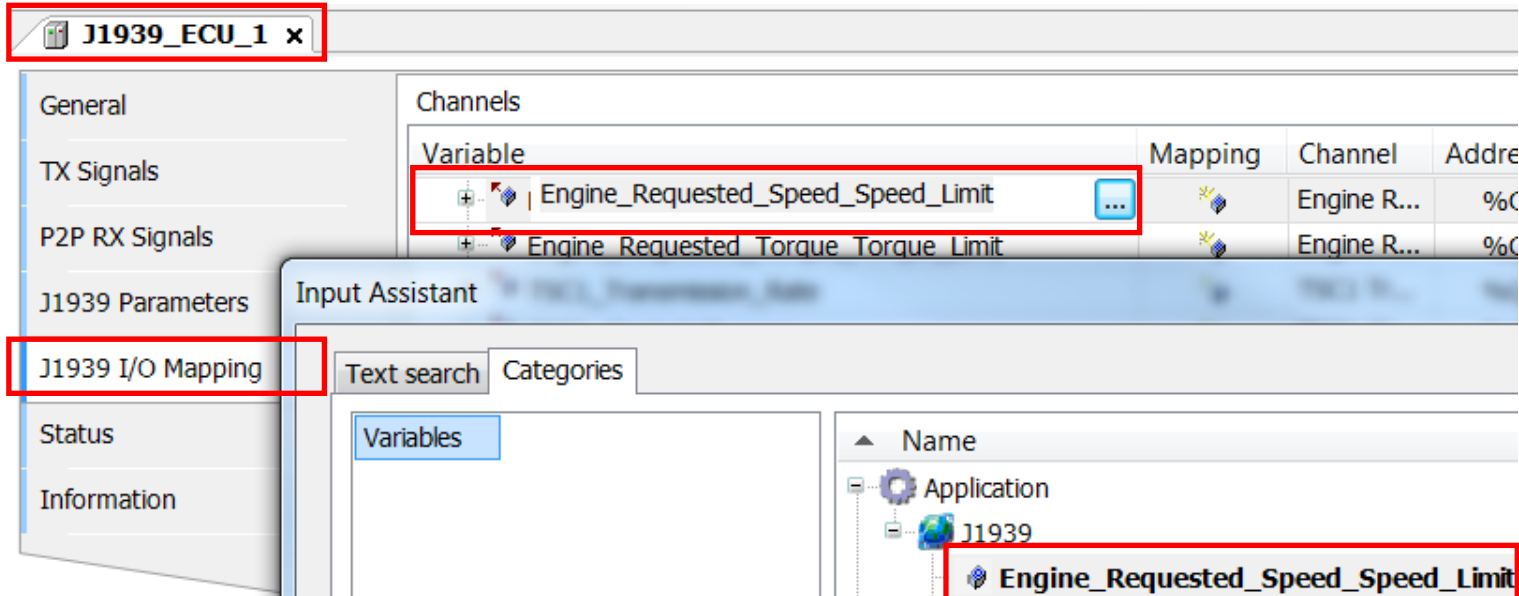
## 5.3b CAN communication – Sending PGNs

The screenshot shows the configuration interface for J1939\_ECU\_1. The 'TX Signals' tab is selected. The 'Add Parameter Group' dialog is open, showing a search for 'TSC1' in the 'Custom' database. The search results table is as follows:

PGN/SPN	Name	Description
0	TSC1	Torque/Speed Control 1

Add a PG and use PG 0

## 5.3b CAN communication – Sending PGNs



Map the variable  
Application.J1939.  
Engine\_Requested\_Speed\_Speed\_Limit

Download and test  
Change the value with the slider

## 5.3b CAN communication – Sending cyclically



J1939\_ECU\_1 x PLC\_PRG

General  
TX Signals  
P2P RX Signals  
**J1939 Parameters**  
J1939 I/O Mapping  
Status  
Information

Parameter	Type	Value
Preferred Address	BYTE(0..253)	16#...
NAME	LWORD	16#...
Local Device	BOOL	TRUE
Communication Watchdog En...	BOOL	FALSE
Communication Watchdog Ti...	UINT	16#...
Active DTC		
LampInfo		
<b>Tx PGs</b>		
Activated PGs	UDINT	1
<b>PG 0: PGN 0</b>		
PG Offline Data		
PG Online Data		
<b>PG Transmission Settings</b>		
Priority	BYTE(0..7)	3
<b>TransmissionMode</b>	Enumeration of BYTE	<b>Cyclic</b>
Destination Address	BYTE	0
<b>CycleTimeFactor</b>	UDINT	<b>1</b>

Make the settings according to the screenshot

The default sending frequency is derived from the task with the lowest interval

Change it by setting the CycleTimeFactor

Example:

Main task runs with 50 ms;

CycleTimeFactor 3

-> PGN is sent every 150 ms



## 5.3b CAN communication – Custom PGNs

Add Parameter Group

Database: **Custom**

PGN

PGN (18 bit): 61440

PDU Specific (Bit 0-7): 0

PDU Format (Bit 8-15): 240

Datapage (Bit 16): 0

Extended Datapage (Bit 17): 0

General

Name: ParameterGroup

Length (Bytes): 8

Priority: 3

Description:

Add PG Close

If a custom PGN message is needed, add a PG and switch to the Custom tab to make the necessary settings

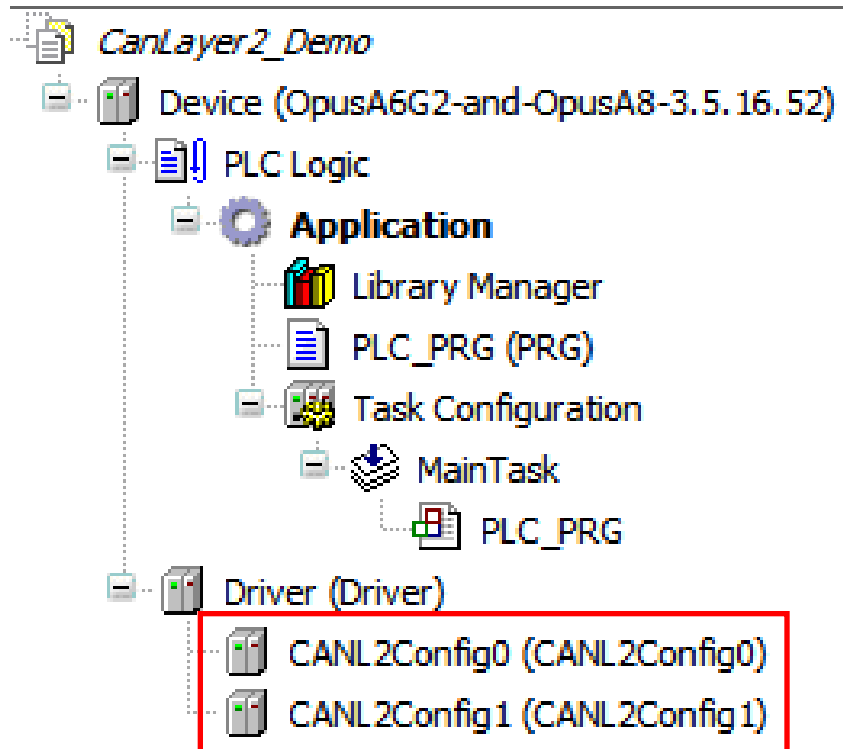
Continue with chapter 6

## 5.1c CAN communication

# CAN Layer 2

## 5.1c CAN communication – Setup

## CAN Layer 2

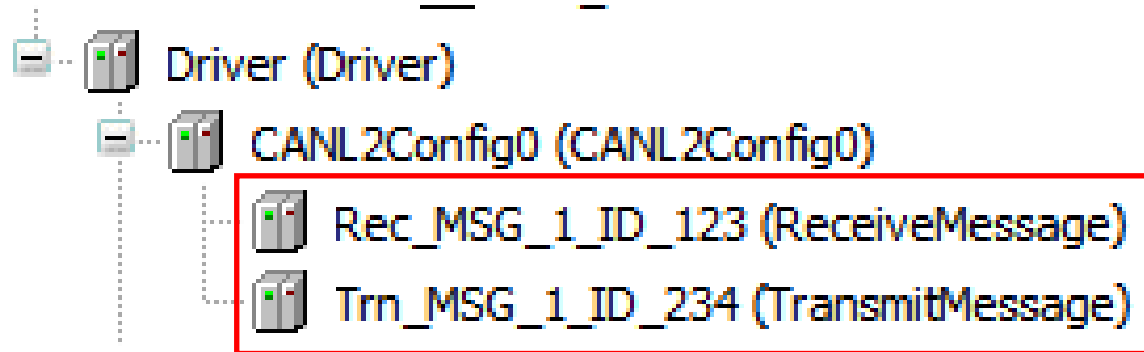


First, we need one or both of these modules in the Driver Tree

Config0 -> First CAN port

Config1 -> Second CAN port

## 5.1c CAN communication – Setup



## CAN Layer 2

Add ReceiveMessage modules and TransmitMessage modules as needed for each CAN port used

One Module for each Mapping / CAN ID

## 5.1c CAN communication – Setup

```
1 | CANL2Config0.Active := TRUE;
```

## CAN Layer 2

The CAN Layer 2 module needs to be activated at the start of the project, so put this code in a startup sequence

## 5.1c CAN communication – Receive messages

## CAN Layer 2

Parameter	Type	Value	Default Value	Unit	Description
Set COBID	Enumeration of BOOL	inactive	inactive		Set CONID on startup
COBID	UDINT				COBID of the message
Set DLC	Enumeration of BOOL	inactive	inactive		Set DLC on startup
DLC	USINT				DLC of the message
Set x29Bit	Enumeration of BOOL	inactive	inactive		Set x29Bit on startup
x29Bit	Enumeration of BOOL	inactive	inactive		29 bit identifier message
xRTR	Enumeration of BOOL	inactive	inactive		Remote request
Timestamp	UDINT				Timestamp of the message
Counter	ULINT				Number of caught messages
Diagnosis Message:	STRING	"	"		
DiagAcknowledge	BOOL	0	0		Extended diagnostic information acknowledge

The setup of a message can be done in the table (Double-click on the message in the tree) or in code

Important settings:

- COBID – The CAN ID of the message (in the example: 0x123)
- DLC – The length of the data (0 to 8 byte)
- x29bit – True if 29bit ID's are used, False if 11bit ID's are used

The same settings in code

```

Rec_MSG_1_ID_123.COBID := 16#123;
Rec_MSG_1_ID_123.DLC := 8;
Rec_MSG_1_ID_123.x29Bit := false;

```

## 5.1c CAN communication – Receive messages - The data

## CAN Layer 2

The data for a receive mapping can be processed like this

```
values : ARRAY [0..7] OF USINT;
```

```
values := Rec_MSG_1_ID_123.Data;  /// read all 8 bytes  
values[0] := Rec_MSG_1_ID_123.Byte0;  /// read the first byte
```

Declaration

Code

## 5.1c CAN communication – Receive messages - Important

## CAN Layer 2

What's important to know:

There is no automatic processing of the messages, getting the data must be done in code in a timely manner



# 5.1c CAN communication – Transmit messages

## CAN Layer 2

Parameter	Type	Value	Default Value	Unit	Description
Set COBID	Enumeration of BOOL	inactive	inactive		Set CONID on startup
COBID	UDINT				COBID of the message
Set DLC	Enumeration of BOOL	inactive	inactive		Set DLC on startup
DLC	USINT				DLC of the message
Set Transmit Trigger	Enumeration of BOOL	inactive	inactive		Set transmit trigger on startup
Transmit Trigger	Enumeration of DINT				Transmit trigger for the message
Set Timeout	Enumeration of BOOL	inactive	inactive		Set timeout on startup
Timeout	time	t#100ms	t#100ms		Timeout for the timer based trigger
Set x29Bit	Enumeration of BOOL	inactive	inactive		Set x29Bit on startup
x29Bit	Enumeration of BOOL	inactive	inactive		29 bit identifier message
Set xRTR	Enumeration of BOOL	inactive	inactive		Set xRTR on startup
xRTR	Enumeration of BOOL	inactive	inactive		Remote request
Trigger	Enumeration of BOOL	inactive	inactive		Send the message
Counter	ULINT				Number of caught messages
Diagnosis Message:	STRING	"	"		
DiagAcknowledge	BOOL	0	0		Extended diagnostic information acknowledge

The setup of a message can be done in the table (Double-click on the message in the tree) or in code

Important settings:

- COBID – The CAN ID of the message (in the example: 0x123)
- DLC – The length of the data (0 to 8 byte)
- Transmit Trigger – When/how should the message be sent (see next slide)
- x29bit – True if 29bit ID's are used, False if 11bit ID's are used

```
Trn_MSG_1_ID_234.COBID := 16#234;
Trn_MSG_1_ID_234.DLC := 8;
Trn_MSG_1_ID_234.x29Bit := FALSE;
```

The same settings in code

## 5.1c CAN communication – Transmit trigger

## CAN Layer 2

Parameter	Type	Value	Default Value	Unit	Description
Set COBID	Enumeration of BOOL	inactive	inactive		Set CONID on startup
COBID	UDINT				COBID of the message
Set DLC	Enumeration of BOOL	inactive	inactive		Set DLC on startup
DLC	USINT				DLC of the message
Set Transmit Trigger	Enumeration of BOOL	inactive	inactive		Set transmit trigger on startup
Transmit Trigger	Enumeration of DINT				Transmit trigger for the message
Set Timeout	Enumeration of BOOL	inactive	inactive		Set timeout on startup
Timeout	time	t#100ms	t#100ms		Timeout for the timer based trigger
Set x29Bit	Enumeration of BOOL	inactive	inactive		Set x29Bit on startup
x29Bit	Enumeration of BOOL	inactive	inactive		29 bit identifier message
Set xRTR	Enumeration of BOOL	inactive	inactive		Set xRTR on startup
xRTR	Enumeration of BOOL	inactive	inactive		Remote request
Trigger	Enumeration of BOOL	inactive	inactive		Send the message
Counter	ULINT				Number of caught messages
Diagnosis Message:	STRING	"	"		
DiagAcknowledge	BOOL	0	0		Extended diagnostic information acknowledge

Transmit trigger can be:

- 0 – none (mapping inactive)
- 1 – timerBased (Timeout needs to be set)
- 2 – onValueChanged (any value in the mapping changes)
- 3 – a combination of 1 and 2

```
Trn_MSG_1_ID_234.COBID := 16#234;
Trn_MSG_1_ID_234.DLC := 8;
Trn_MSG_1_ID_234.x29Bit := FALSE;
Trn_MSG_1_ID_234.TransmitTrigger := 1;
Trn_MSG_1_ID_234.Timeout := Time#100ms;
```

Example in code for a cyclic transmit message every 100 ms

# Agenda

1. Introduction
2. Updating your device
3. Getting to know the CODESYS IDE
4. First project
5. CAN communication
6. Extended agenda
7. FAQ

## 6. Extended agenda

1. ARM library
2. Backup / USB project loading
3. PDF Reader
4. Power management
5. Retain variables
6. Date and time
7. Languages
8. Data logging / USB file transfer
9. Alarms

## 6.1. Extended agenda – Wachendorff ARM library

The screenshot shows the OPUS software interface. On the left, a project tree is visible under 'First\_project'. The 'Driver (Driver)' element is highlighted with a red box. On the right, the 'Add Device' dialog box is open, also with a red box around its title bar. The dialog has fields for 'Name', 'Action', 'Device', and 'Vendor'. The 'Action' section has radio buttons for 'Append device', 'Insert device', 'Plug device', and 'U'. The 'Vendor' dropdown is set to '<All vendors>'. Below these fields is a table listing various device components from the Wachendorff library.

Name	Vendor	Version
Miscellaneous		
Config	Wachendorff Elektronik	3.5.8.3
Daemon	Wachendorff Elektronik	3.5.8.3
EEPROM	Wachendorff Elektronik	3.5.8.3
Encoder	Wachendorff Elektronik	3.5.8.3
InputAnalog	Wachendorff Elektronik	3.5.8.3
InputDigital	Wachendorff Elektronik	3.5.8.3
Keyboard	Wachendorff Elektronik	3.5.8.3

Device structure for everything specific to OPUS devices:

- Inputs/Outputs
- Backlight
- Beeper
- Power management
- EEPROM
- Device configurations

The elements are explained in our help file

## 6.1. Extended agenda – Wachendorff ARM library

The screenshot shows a software development environment. In the top-left project tree, 'DisplayBacklight (DisplayBacklight)' is selected and highlighted with a red box. Below this, a window titled 'DisplayBacklight x' displays a table of parameters:

DisplayBacklight Parameters	Parameter	Type	Value
DisplayBacklight I/O Mapping	Set Brightness	Enumeration of BOOL	inactive
Status	Brightness	DINT	255

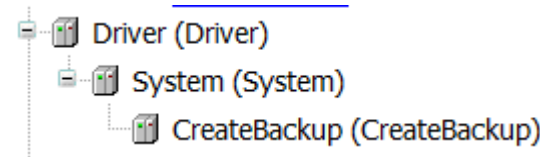
Below the table, a code editor shows the text 'DisplayBacklight.' with a dropdown menu open, listing 'Brightness', 'MaxBrightness', and 'valid'. The 'Brightness' option is selected.

Example: Display Backlight

Usage in code:

```
DisplayBacklight.Brightness := 255;
```

## 6.2. Backup / USB project loading



CreateBackup x

CreateBackup Parameters

CreateBackup I/O Mapping

Status

Information

Parameter	Type	Current Value	Prepared Value
BackupMode	Enumeration of INT	Runtime files	▼
Version	STRING	'1.0.0'	
Trigger	BOOL	FALSE	Runtime files
Busy	BOOL	FALSE	Runtime files + font
Done	BOOL	FALSE	Project files
Error	BOOL	FALSE	Project files + fonts
Diagnosis Message:	STRING	"	RETAIN files
DiagAcknowledge	BOOL	FALSE	Bootlogo

To install the project to devices with a USB stick, a backup from a connected device is needed

Install these devices

First set the mode of the backup

Then set the *Trigger* to *true*

USB stick needs to be inserted

Runtime files -> user\_\_\_\_.tar.gz

Project files -> Update at runtime with Update Daemon

## 6.4. Extended agenda – Power management

CL  
15

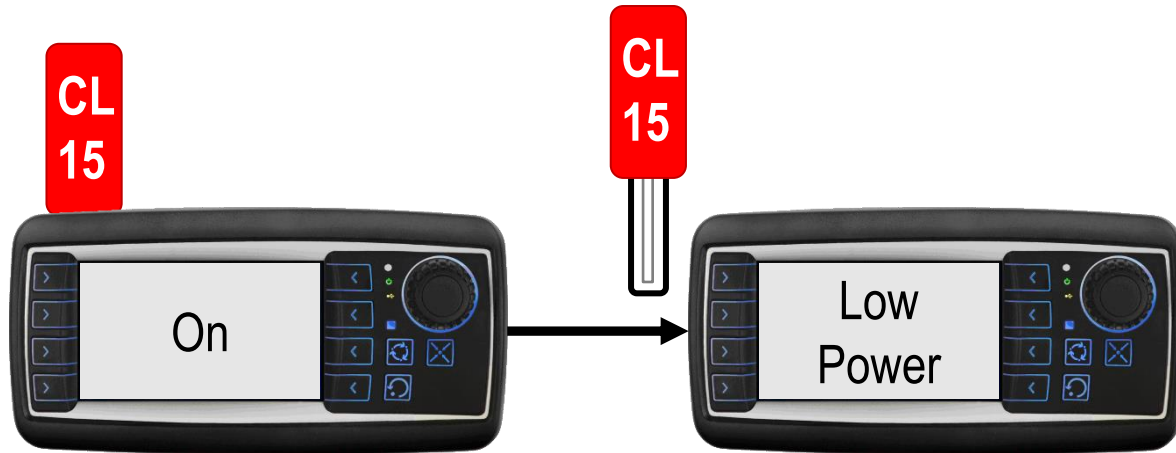


mA @13.5 V

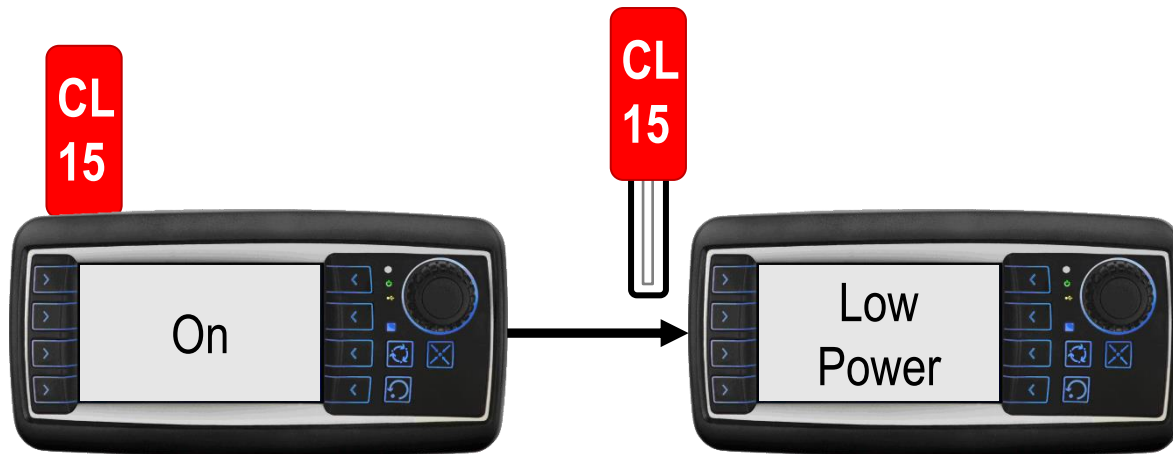
	A3	A6	A6G2	A8
On	430	900	1000	1600



## 6.4. Extended agenda – Power management



## 6.4. Extended agenda – Power management

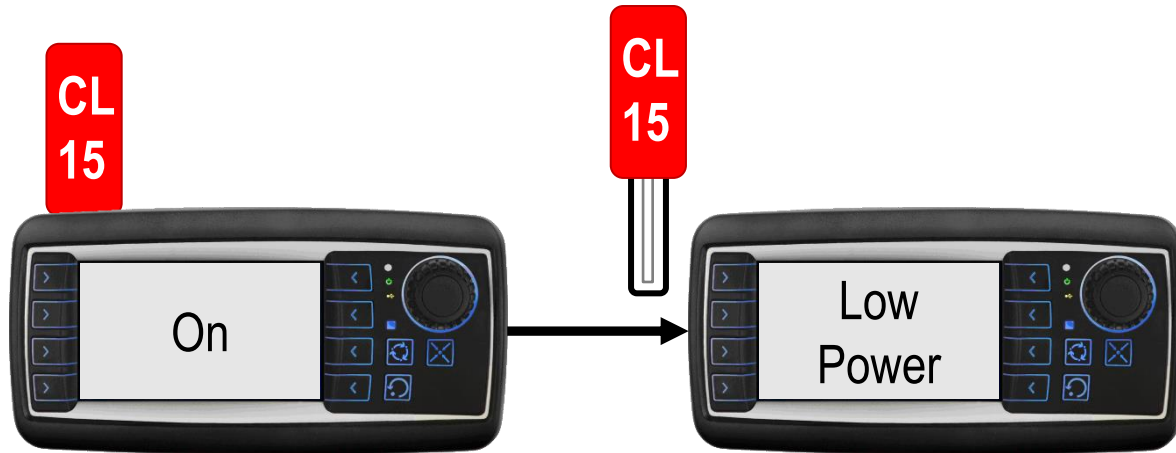


Codesys, CAN are working

- |   |  |
|---|--|
| <input type="checkbox"/> /✓ Display function  | <input type="checkbox"/> /✓ Multicolor LED   |
| <input type="checkbox"/> /✓ Display backlight | <input type="checkbox"/> /✓ Beeper / Speaker |
| <input type="checkbox"/> /✓ Key backlight     | <input type="checkbox"/> /✓ Touchscreen      |
| <input type="checkbox"/> /✓ Key function      | <input type="checkbox"/> /✓ Console (RS232)  |
| <input type="checkbox"/> /✓ Encoder function  | <input type="checkbox"/> /✓ Video processing |
| <input type="checkbox"/> /✓ Outputs           | <input type="checkbox"/> /✓ Camera output    |

Driver->Daemon->PMD->\_\_\_\_\_)

# 6.4. Extended agenda – Power management



mA @13.5 V

	A3	A6	A6G2	A8
On	430	900	1000	1600
Low Power*	160	200	200	300

Codesys, CAN are working

- /  Display function
- /  Display backlight
- /  Key backlight
- /  Key function
- /  Encoder function
- /  Outputs
- /  Multicolor LED
- /  Beeper / Speaker
- /  Touchscreen
- /  Console (RS232)
- /  Video processing
- /  Camera output

\*With minimal configuration

Driver->Daemon->PDM->\_\_\_\_\_)

## 6.4. Extended agenda – Power management



## 6.4. Extended agenda – Power management



## 6.4. Extended agenda – Power management



Nothing is working, no functionality!

# 6.4. Extended agenda – Power management

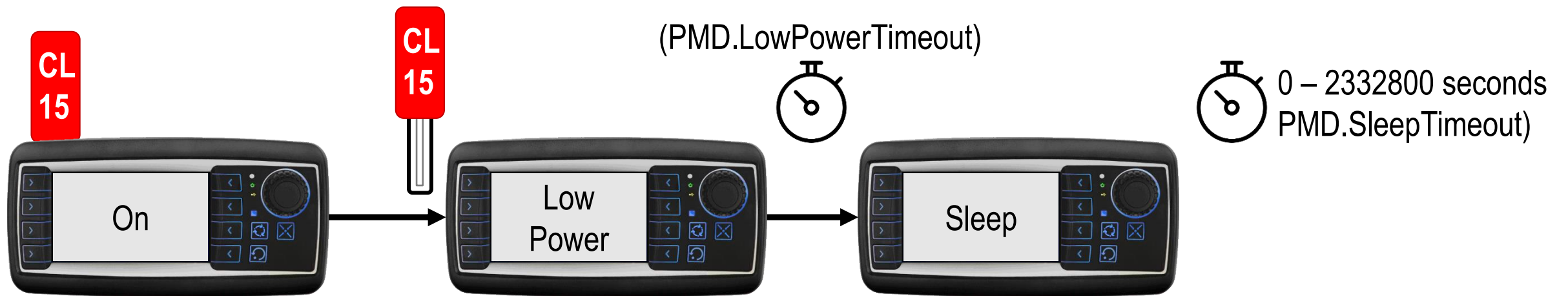


mA @13.5 V

	A3	A6	A6G2	A8
On	430	900	1000	1600
Low Power	160	200	200	300
Sleep	≤ 90	≤ 100	≤ 100	≤ 200

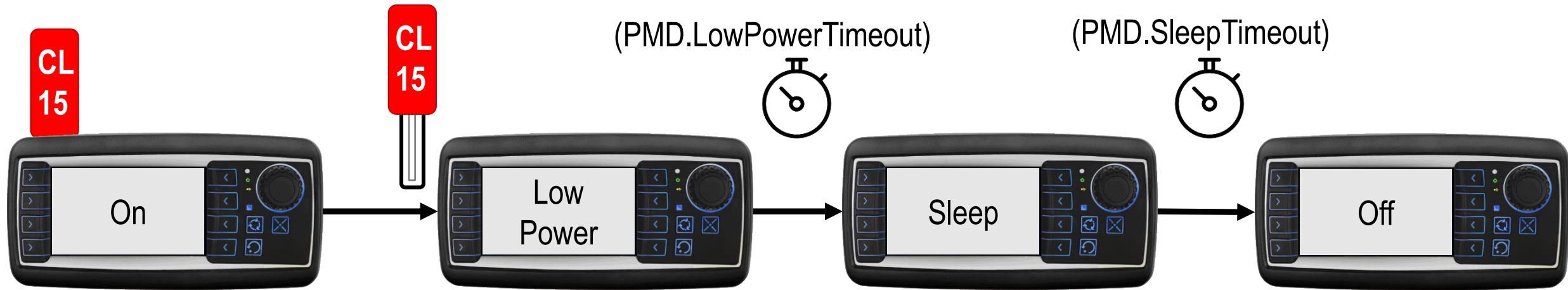
Nothing is working, no functionality!

# 6.4. Extended agenda – Power management

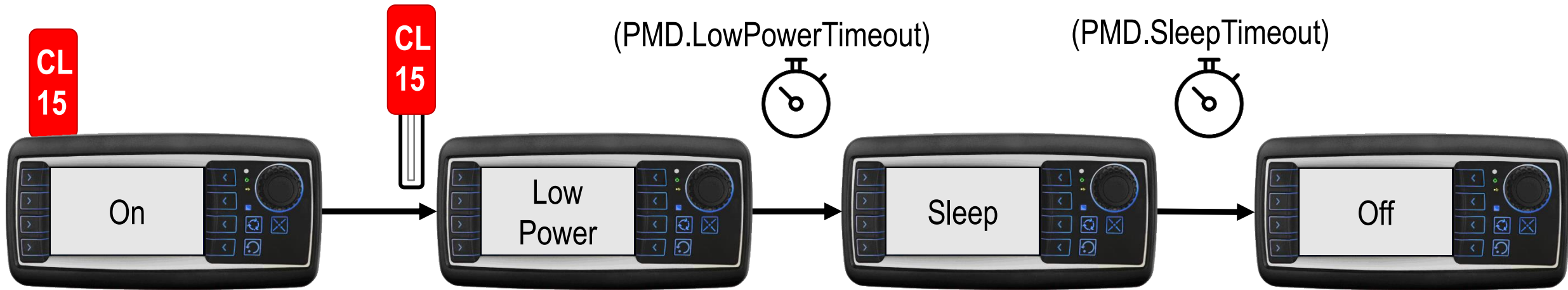




# 6.4. Extended agenda – Power management



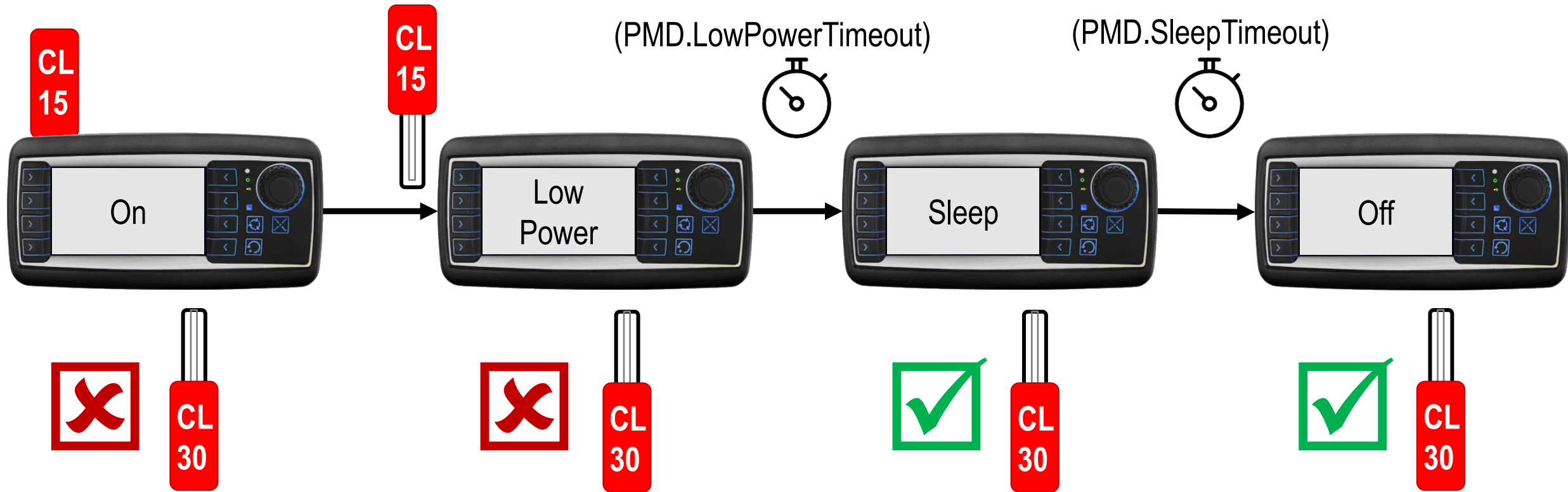
# 6.4. Extended agenda – Power management



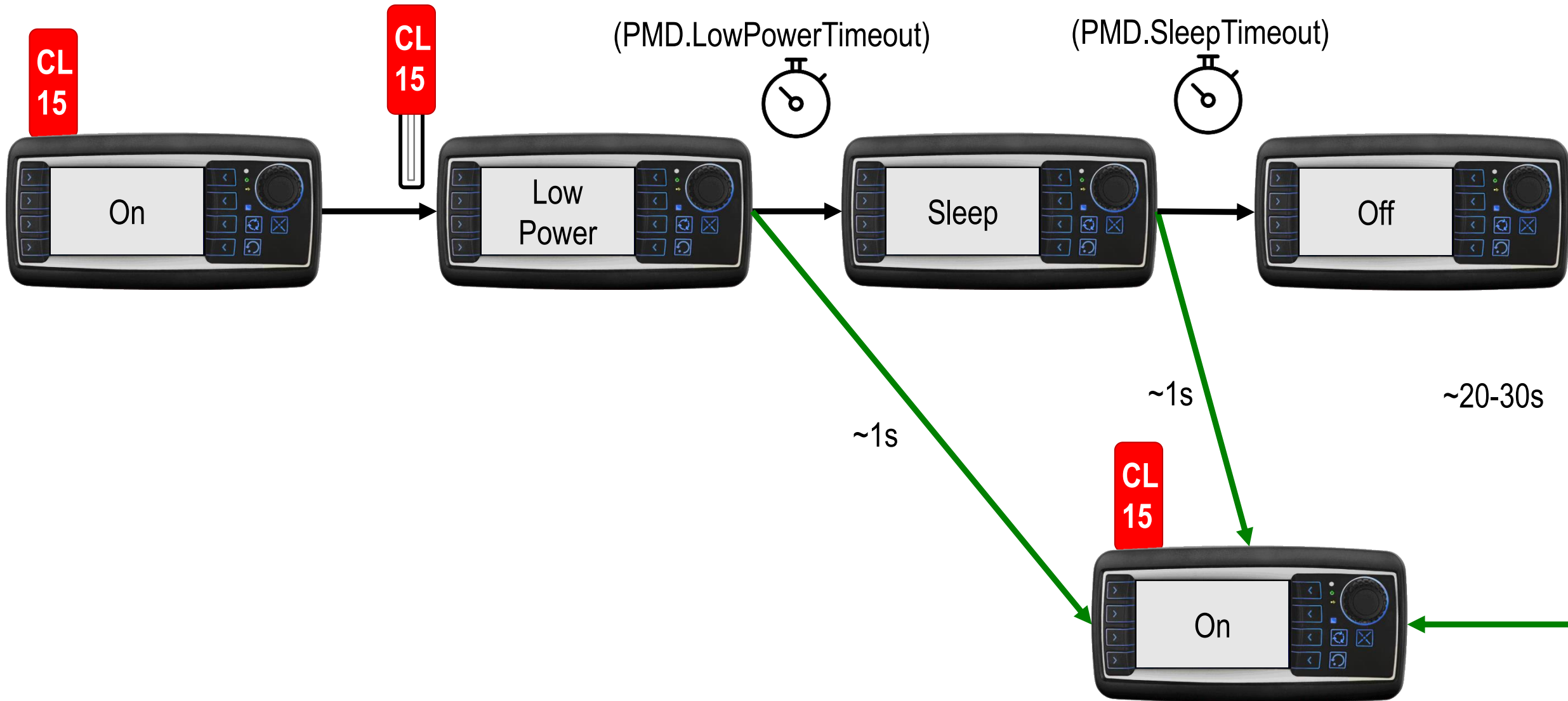
mA @13.5 V

	A3	A6	A6G2	A8
On	430	900	1000	1600
Low Power	160	200	200	300
Sleep	≤ 90	≤ 100	≤ 100	≤ 200
Off	<3	<3	<3	<3

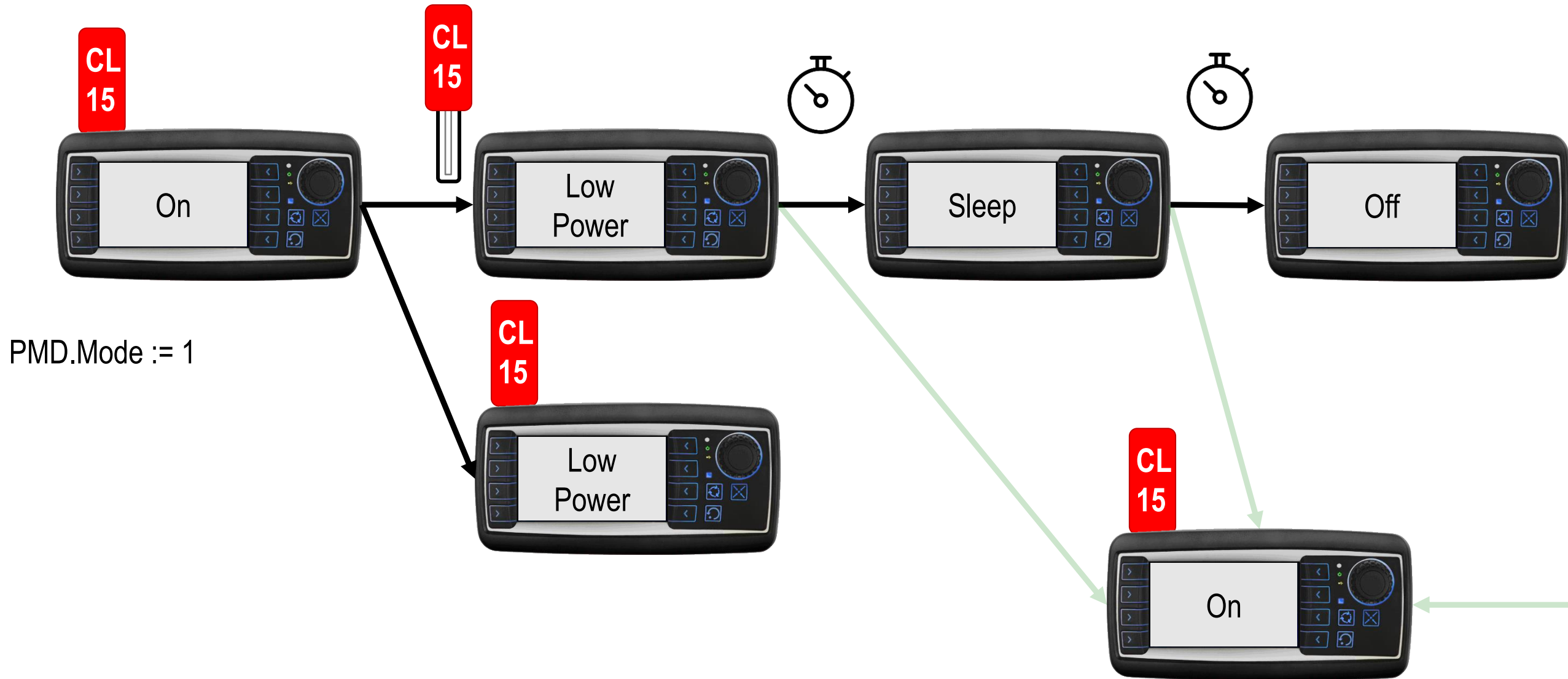
# 6.4. Extended agenda – Power management



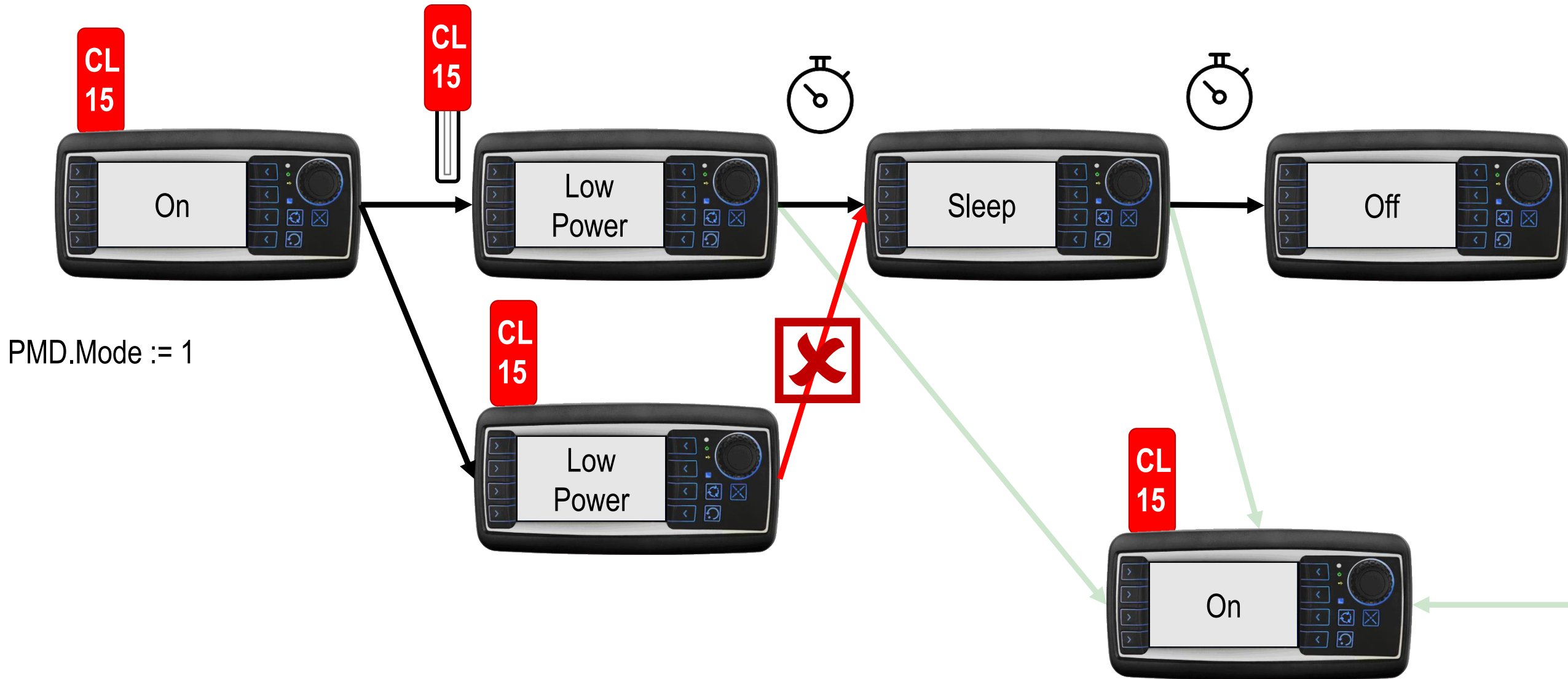
# 6.4. Extended agenda – Power management



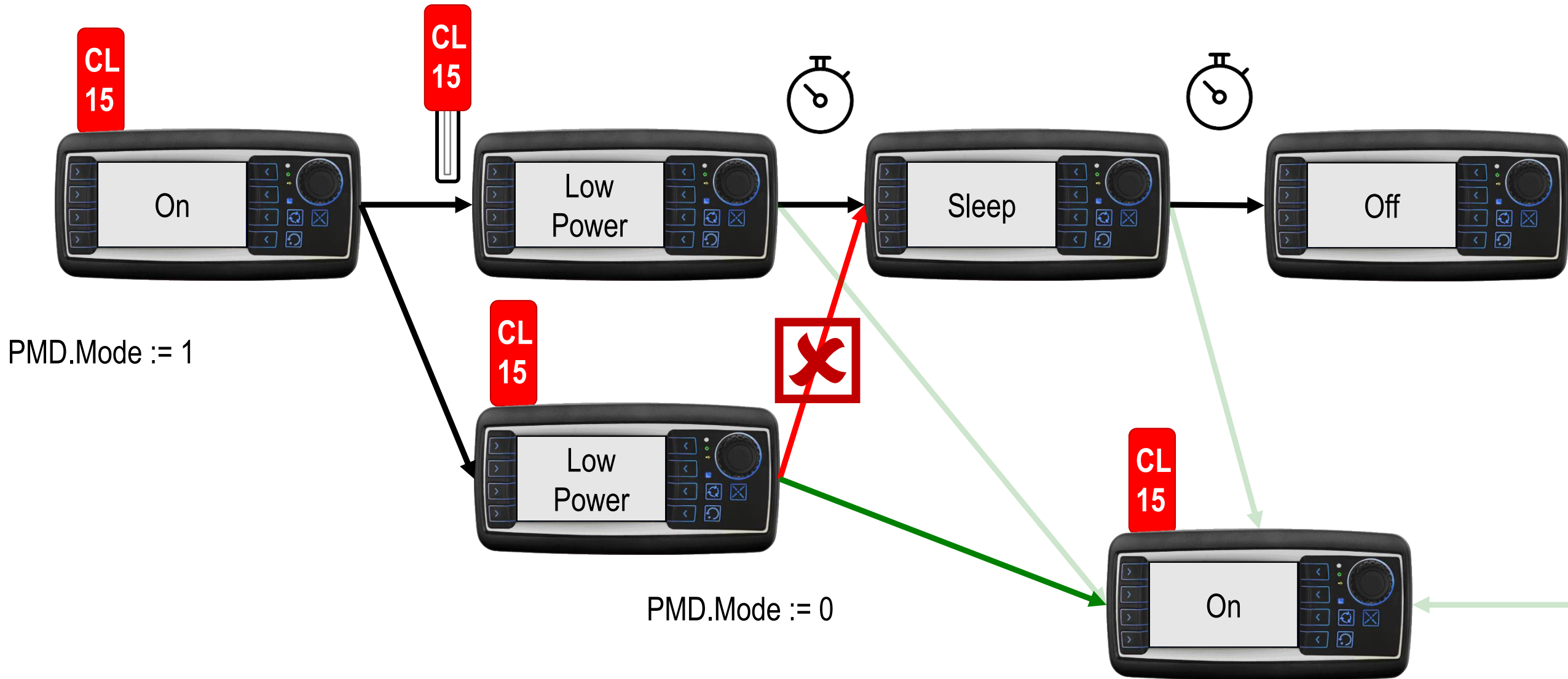
# 6.4. Extended agenda – Power management



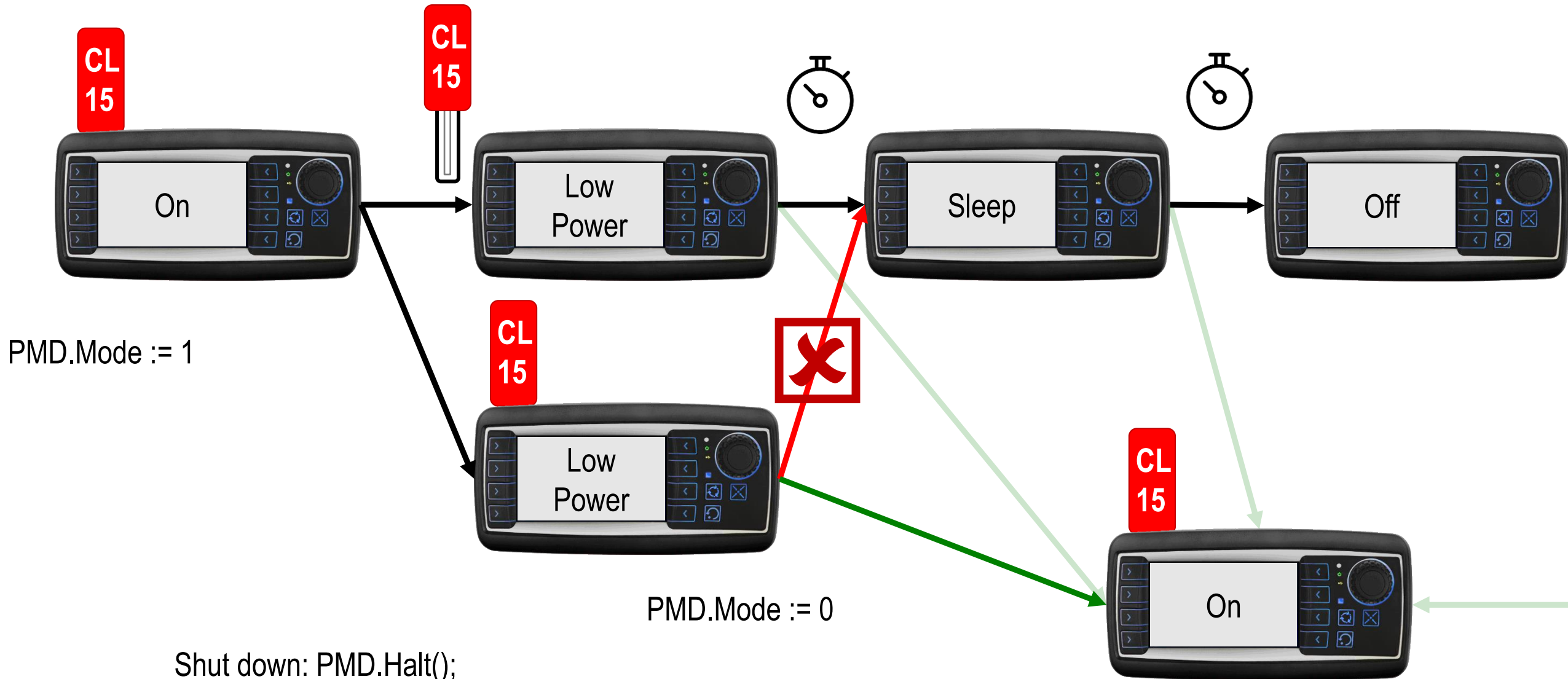
# 6.4. Extended agenda – Power management



# 6.4. Extended agenda – Power management



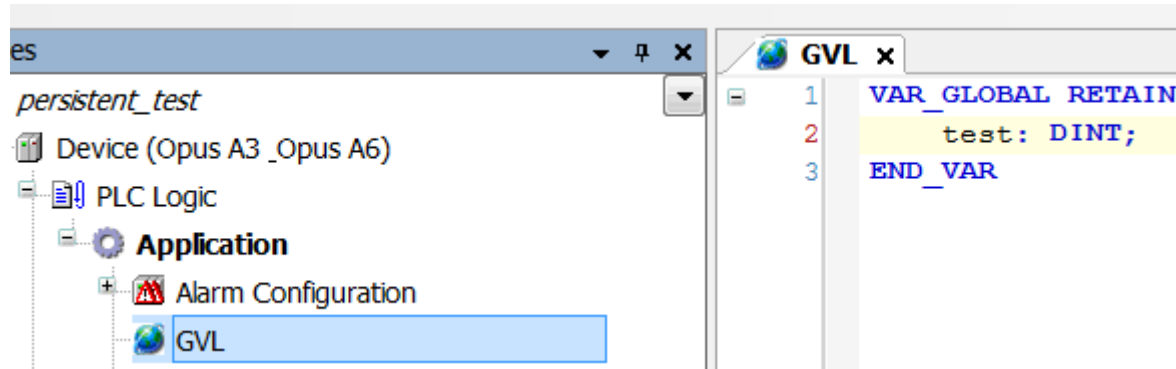
# 6.4. Extended agenda – Power management



Shut down: PMD.Halt();  
 Reboot: PMD.Reboot();



## 6.5. Retain variables

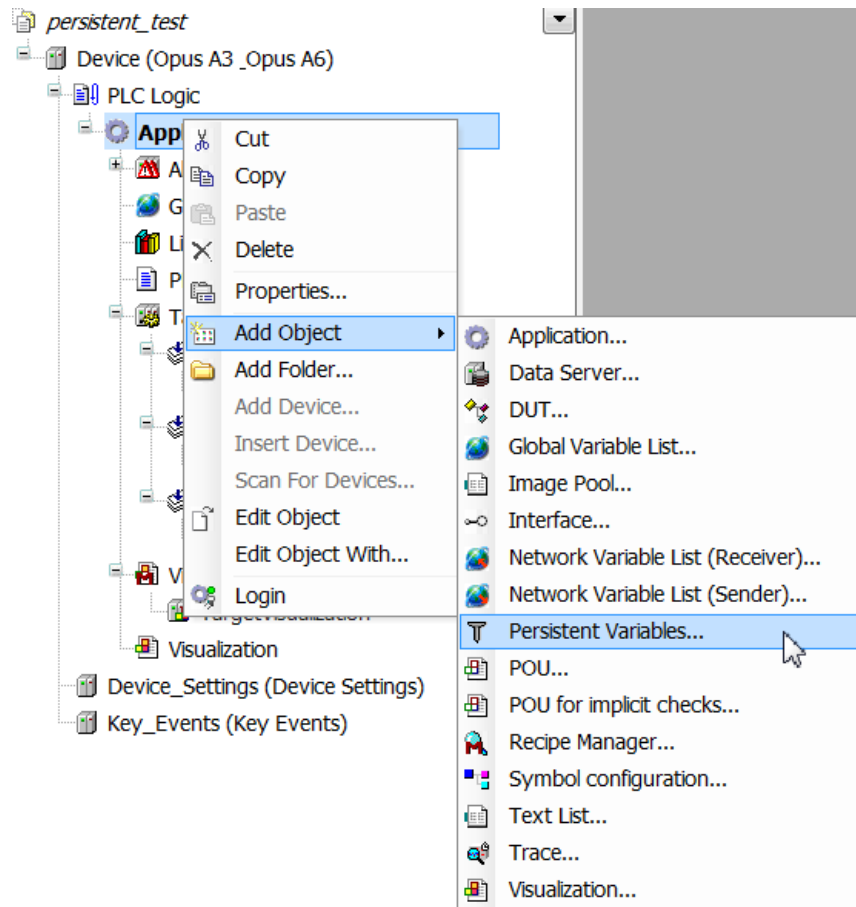


x = Value gets maintained - = Value gets initialized

after Online command	VAR	VAR RETAIN	VAR RETAIN PERSISTENT VAR PERSISTENT RETAIN
Reset warm	-	x	x
Reset cold	-	-	x
Reset origin	-	-	-
Download	-	-	x
Online Change	x	x	x
Reboot	-	x	x

If variables should survive a shutdown or restart, declare them as retain in a Global Variable List.

## 6.5. Retain variables



For persistent variables you need to create a specific persistent variable list.

## 6.5. Retain variables

VAR

```
result:RTS_IEC_RESULT;  
retname:STRING;  
pApp:POINTER TO cmpapp.APPLICATION;
```

END\_VAR

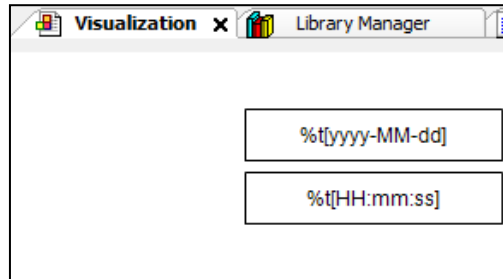
```
pApp := cmpapp.AppGetCurrent(ADR(result));  
retname := concat(pApp^.szName, '.ret');  
result := cmpapp.AppStoreRetainsInFile(pApp, retname);  
TDS.SystemCall('sync');
```

Retain and persistent retain variables are only saved when the device is shutting down

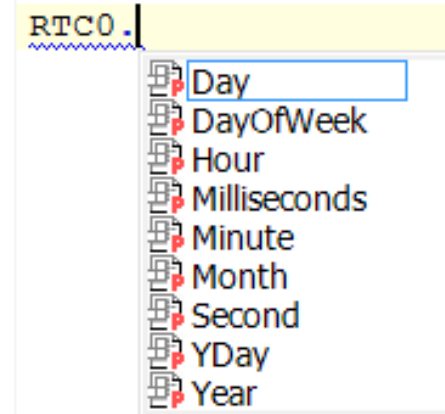
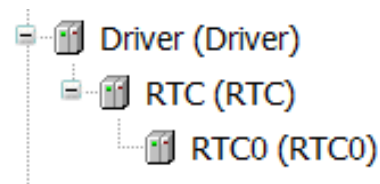
To save at runtime, include this code in your project (e.g. when leaving a settings page)

The processing takes a second, so don't do it too often (not cyclic)

## 6.6. Date and time



-> AM/PM? See chapter "Text and Language in Visualization" in the online help



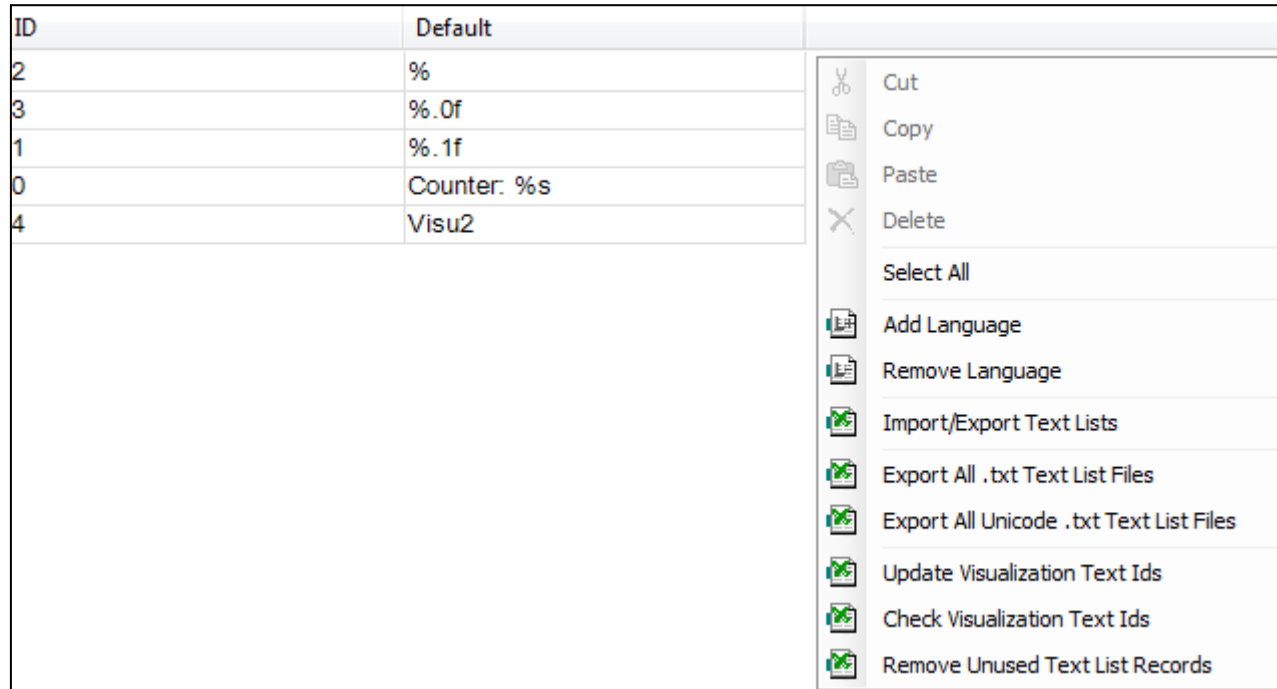
Showing the date and time from the RTC is very easy (in 24h format)

If you need the values (e.g. for writing), add these devices

Values can be read and written in code (not in Visus directly)

## 6.7. Languages – static

ID	Default
2	%
3	%.0f
1	%.1f
0	Counter: %s
4	Visu2



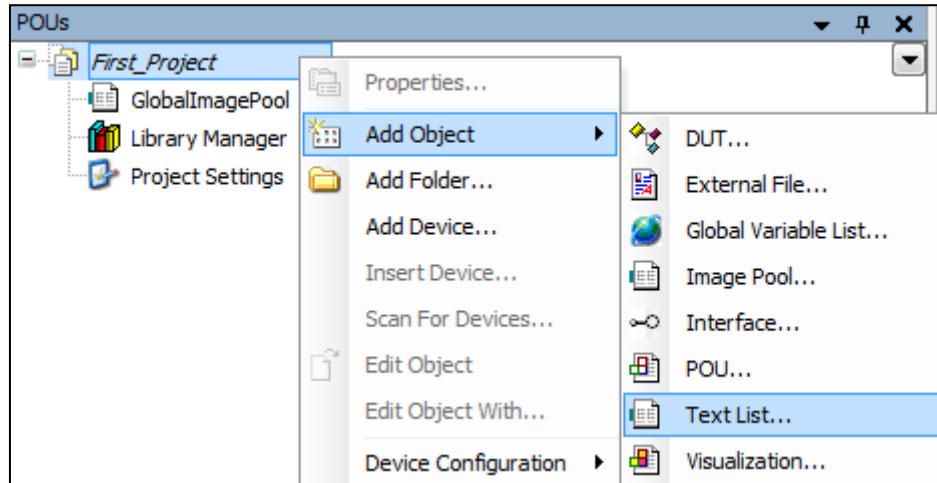
The screenshot shows a table with two columns: 'ID' and 'Default'. The table contains five rows of data. To the right of the table, a context menu is open, listing various actions such as Cut, Copy, Paste, Delete, Select All, Add Language, Remove Language, Import/Export Text Lists, Export All .txt Text List Files, Export All Unicode .txt Text List Files, Update Visualization Text Ids, Check Visualization Text Ids, and Remove Unused Text List Records.

Every String used in the project can be found in the GlobalTextList

There you can add a new language and fill the texts for each language

Export/Import is also possible (for translators)

## 6.7. Languages – dynamic



ID	Default	german
123	english Text	deutscher Text

Dynamic texts	
Text list	'Example'
Text index	123

For dynamic texts you need to create a text list

Add languages just like for static text

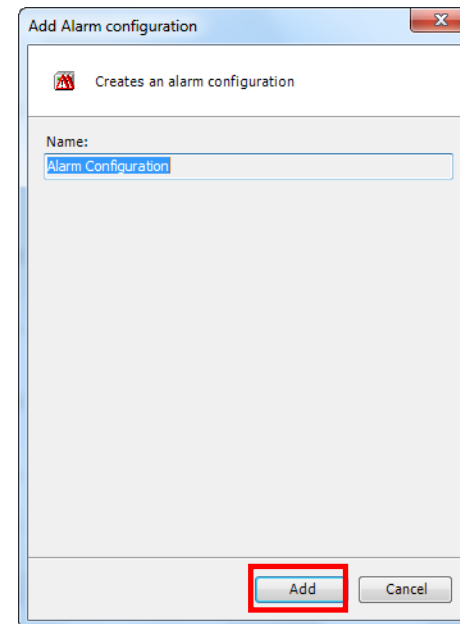
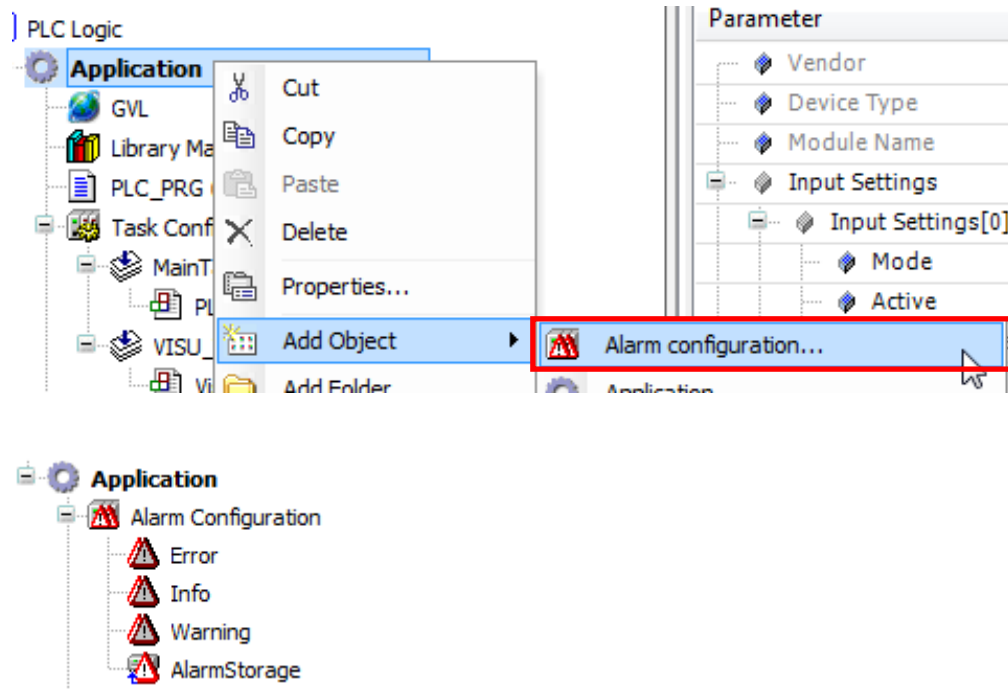
Connect the object with the text list

Switch languages with the variable CurrentLanguage (library VisuElemBase)

## 6.8. – Data logging / USB file transfer

Please check the example project  
USB\_File\_Operations

## 6.9. – Alarms



To create alarms, first add an *Alarm Configuration*

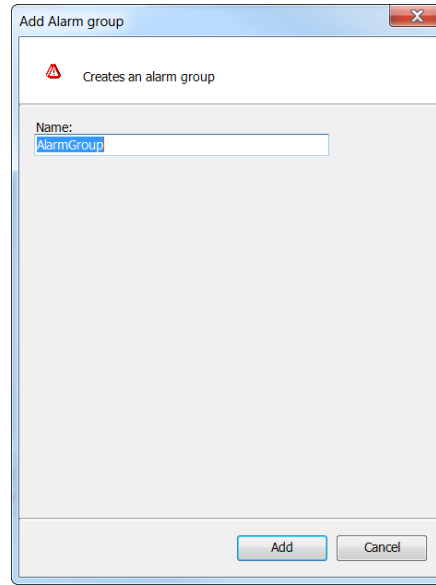
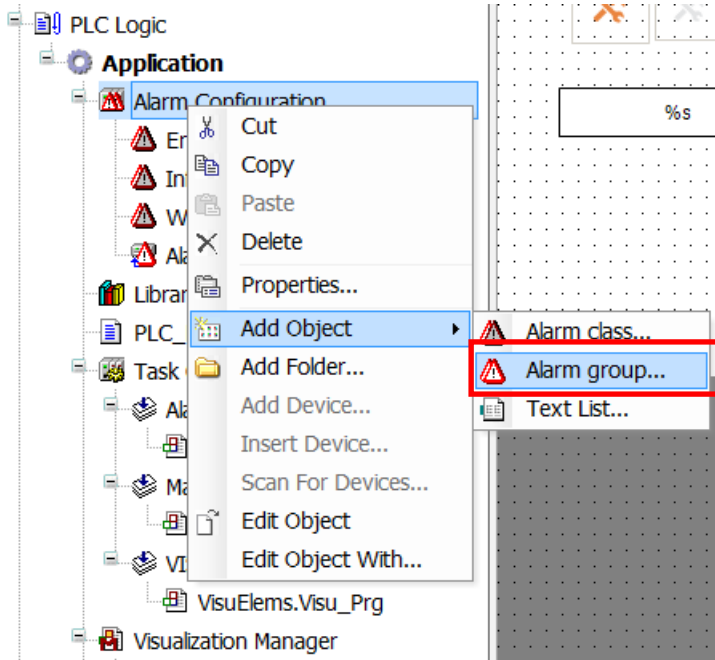
Rename if needed and click *Add*

A structure with 3 alarm levels is created

The AlarmStorage is for archiving the alarms that occur(ed) on the device



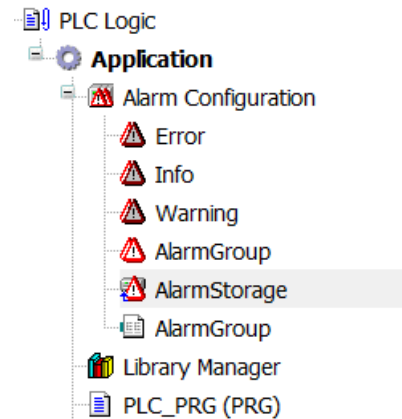
# 6.9. – Alarms



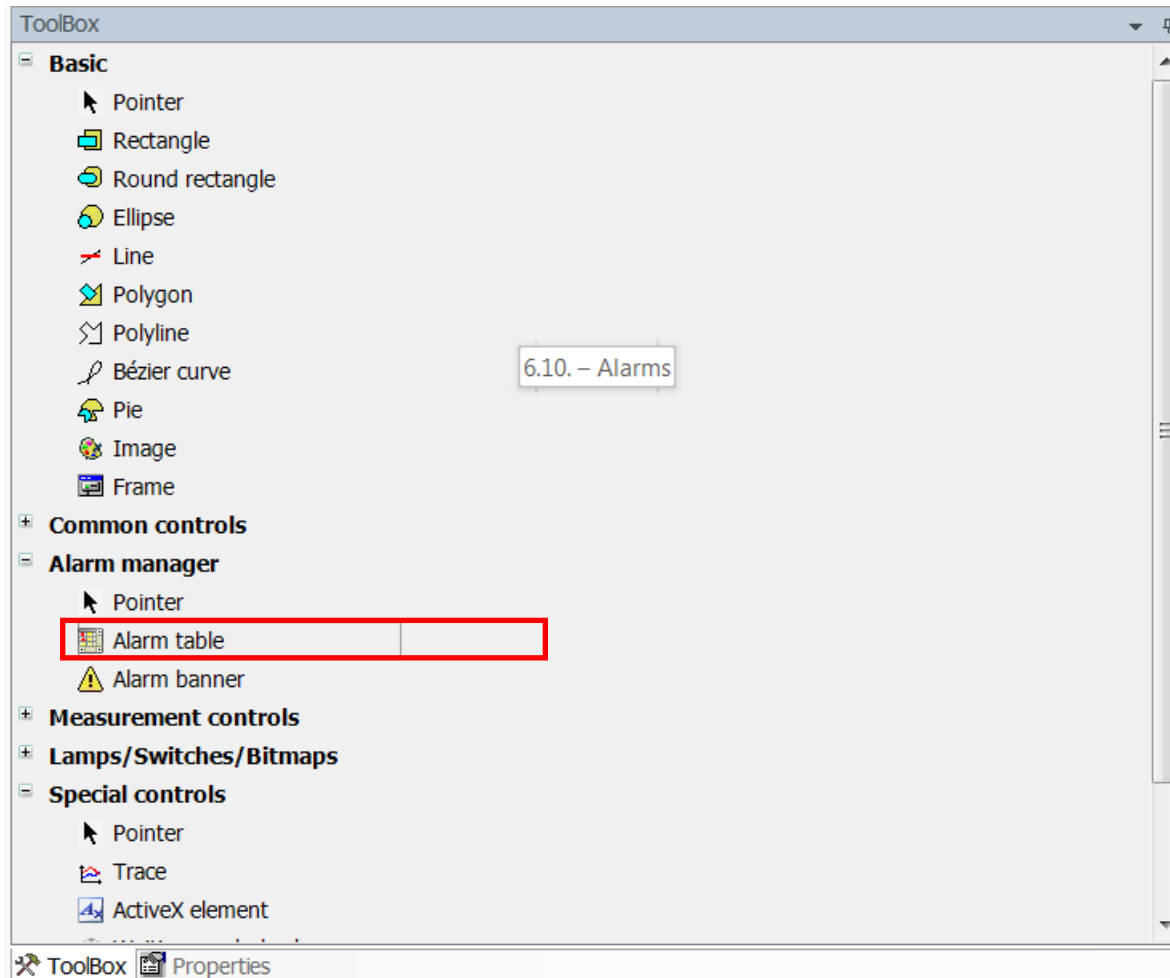
Add an alarm group and give it a name of your choice

In the alarm group the actual alarm events (observations) will be configured

The event in the alarm group will trigger the action defined in the alarm level



## 6.9. – Alarms



Create an alarm table on a visu

The alarm table will display a history of the occurred alarms

You can also create an Alarm banner to quickly see the last / most important alarm

## 6.9. – Alarms

```
1 PROGRAM PLC_PRG
2 VAR
3   counter :INT := 10;
4   up, down :BOOL;
5   ack: bool;
6
7 END_VAR
8
9
10 IF up THEN
11   counter := counter +1;
12 END_IF
13
14 IF down THEN
15   counter := counter -1;
16 END_IF
```

Control variables	
Acknowledge selected	
Acknowledge all visi...	PLC_PRG.ack

Create these variables and the code below

Use ack in the  
Acknowledge all property of the table

## 6.9. – Alarms

The screenshot displays the 'AlarmGroup' configuration window. At the top, the title bar shows 'PLC\_PRG' and 'AlarmGroup x'. Below the title bar, there are fields for 'Textlist: AlarmGroup', 'Archiving: (none)', and 'Deactivation:'. The main area contains a table with the following data:

ID	Observation type	Details	Deactivation	Class	Message	Min. pend. time	Latch var 1	Latch var 2	Higher prio. alarm
0	Lower limit	PLC_PRG.counter < 10		Warning	too low		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
1	Upper limit	PLC_PRG.counter > 20		Error	too high		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
2	Inside range	10 < PLC_PRG.counter AND PLC_PRG.counter < 20		Info	OK		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>

Below the table, there is a configuration panel for the 'Lower limit' observation. It includes a text field for 'Expression' containing 'PLC\_PRG.counter', a dropdown menu set to '<', and a numeric field set to '10'. There are also fields for 'Hysteresis in'.

In the Alarm Group, create 3 observations

# 6.9. – Alarms

Priority 10

Acknowledgement method REP\_ACK

Notification actions

Action	activate	deactivate	confirm	Details	Dea
Click here to ad...				Click here to add a new notifi...	

Details

Display options for alarm table/alarm banner

State	Font	Background color
Active	■ Arial Narrow; 7,8pt	■ 255; 0; 0
Waiting for confirmation		

Priority 100

Acknowledgement method REP\_ACK

Notification actions

Action	activate	deactivate	confirm	Details	Dea
Click here to ad...				Click here to add a ne	

Details

Display options for alarm table/alarm banner

State	Font	Ba
Active	■ Microsoft Sans Serif; 7,8pt	■
Waiting for confirmation		

Configure the alarm levels

## 6.9. – Alarms

PLC\_PRG AlarmGroup Error Info **Warning x**

Priority 30

Archiving

Acknowledgement  
 Acknowledgement method REP\_ACK  
 acknowledge separately

Notification actions

Action	activate	deactivate	confirm	Details
Variable	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	CURRENTVISU := 'HomePage'
Click here to ad...				Click here to add a new notifica

Details

Display options for alarm table/alarm banner

State	Font	Background
Active	■ Arial; 8,25pt	■ 255

Configure the alarm levels

Then test the alarms by changing the counter variable

# Agenda

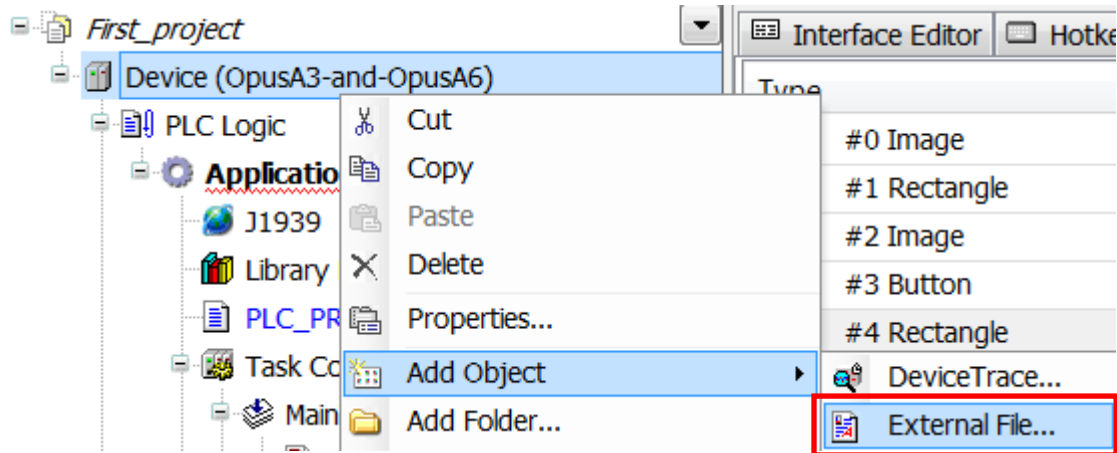
1. Introduction
2. Updating your device
3. Getting to know the CODESYS IDE
4. First project
5. CAN communication
6. Extended agenda
7. FAQ

## 7 – FAQ

1. How can I use different fonts in my project?
2. How can I change the boot logo?
3. How can I display a camera image
4. How can I use the Codesys Web Visu?



## 7.1 – FAQ – How can I use different fonts in my project?

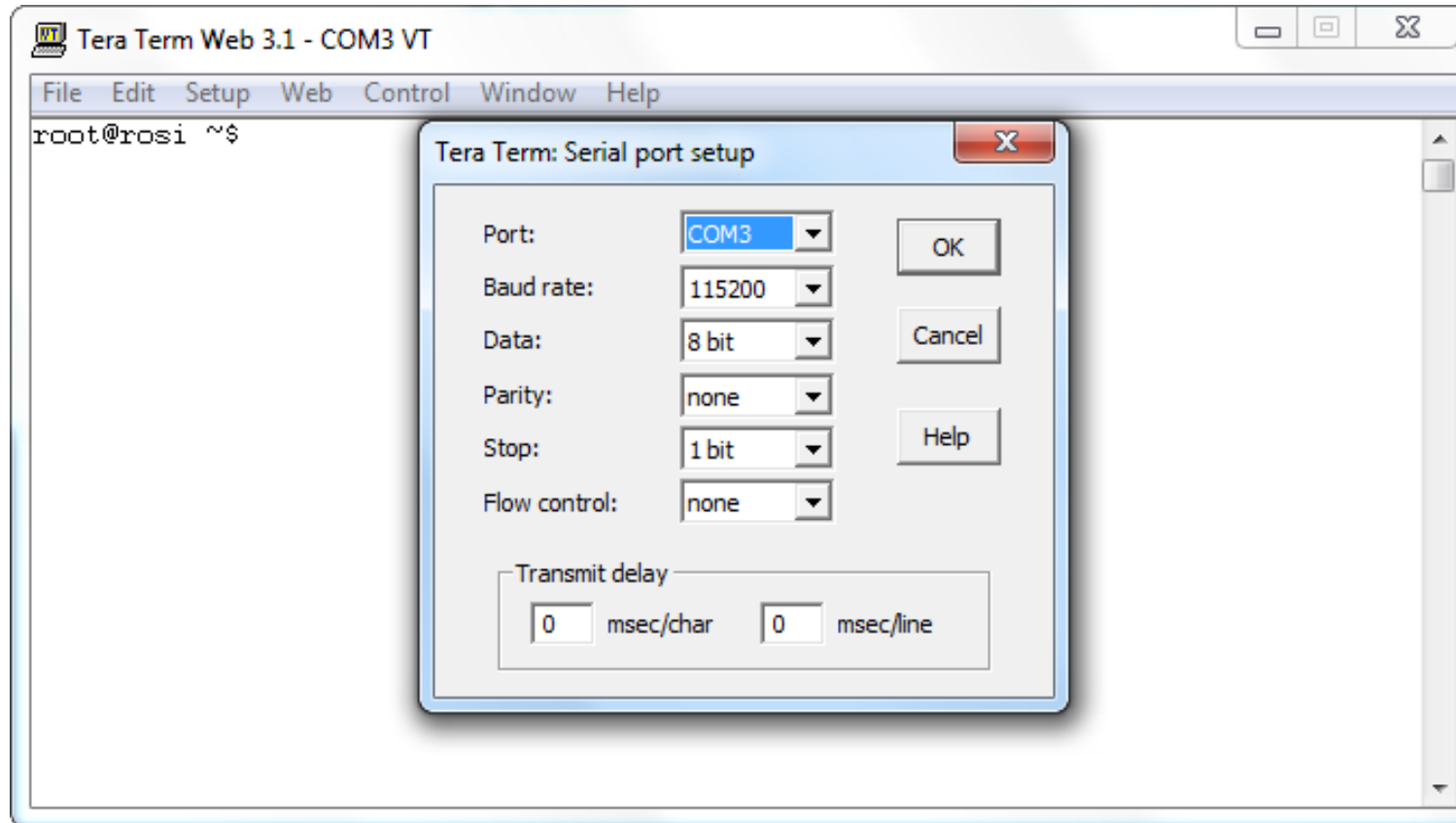


Fonts need to be installed in Windows so they can be chosen in Codesys

But they also need to be on the device.

Load them into the project with *Device* -> *Add Object* -> *External File...*

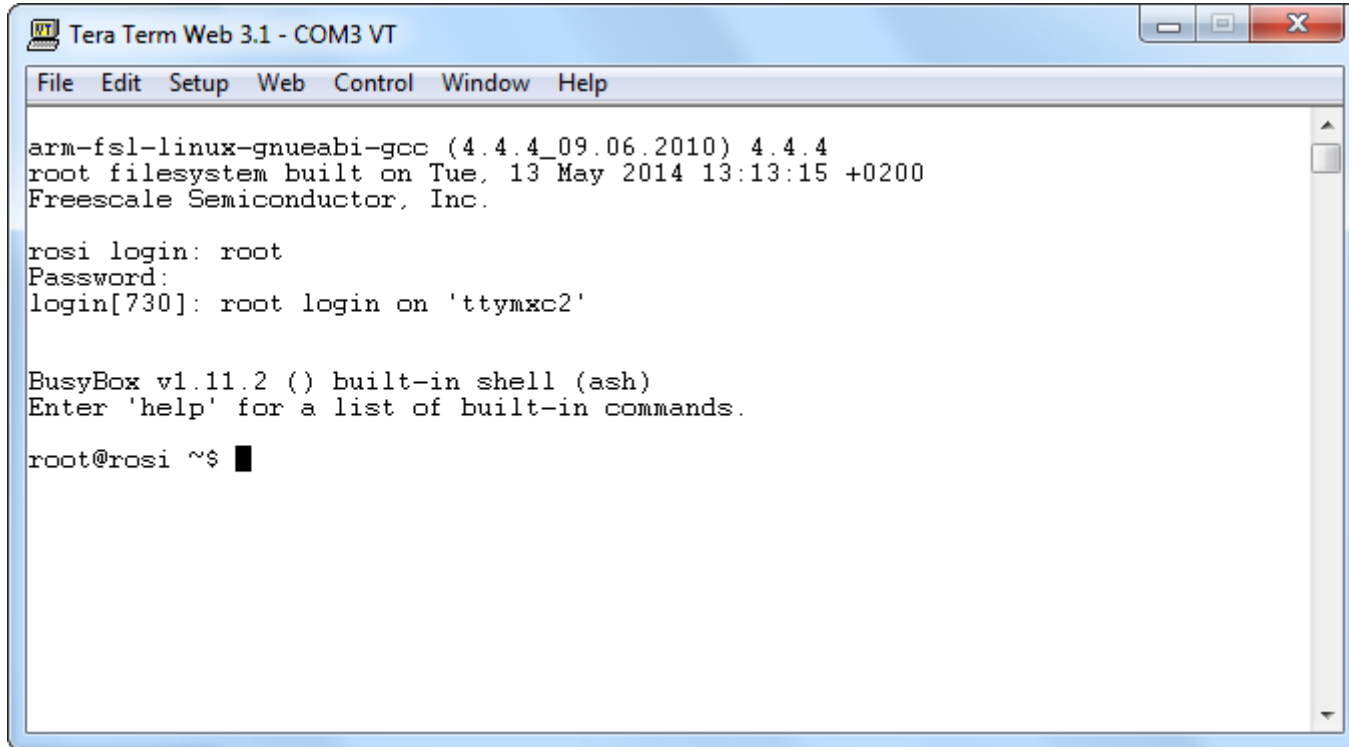
## 7.2 – FAQ – Connect to the serial console



To connect to the serial console of the device, use a program like HyperTerminal, PuTTY or TeraTerm.

Use the following settings to connect (COM port can differ depending on your configuration)

## 7.2 – FAQ – Connect to the serial console



```
Tera Term Web 3.1 - COM3 VT
File Edit Setup Web Control Window Help
arm-fsl-linux-gnueabi-gcc (4.4.4_09.06.2010) 4.4.4
root filesystem built on Tue, 13 May 2014 13:13:15 +0200
Freescale Semiconductor, Inc.

rosi login: root
Password:
login[730]: root login on 'ttyMXC2'

BusyBox v1.11.2 () built-in shell (ash)
Enter 'help' for a list of built-in commands.

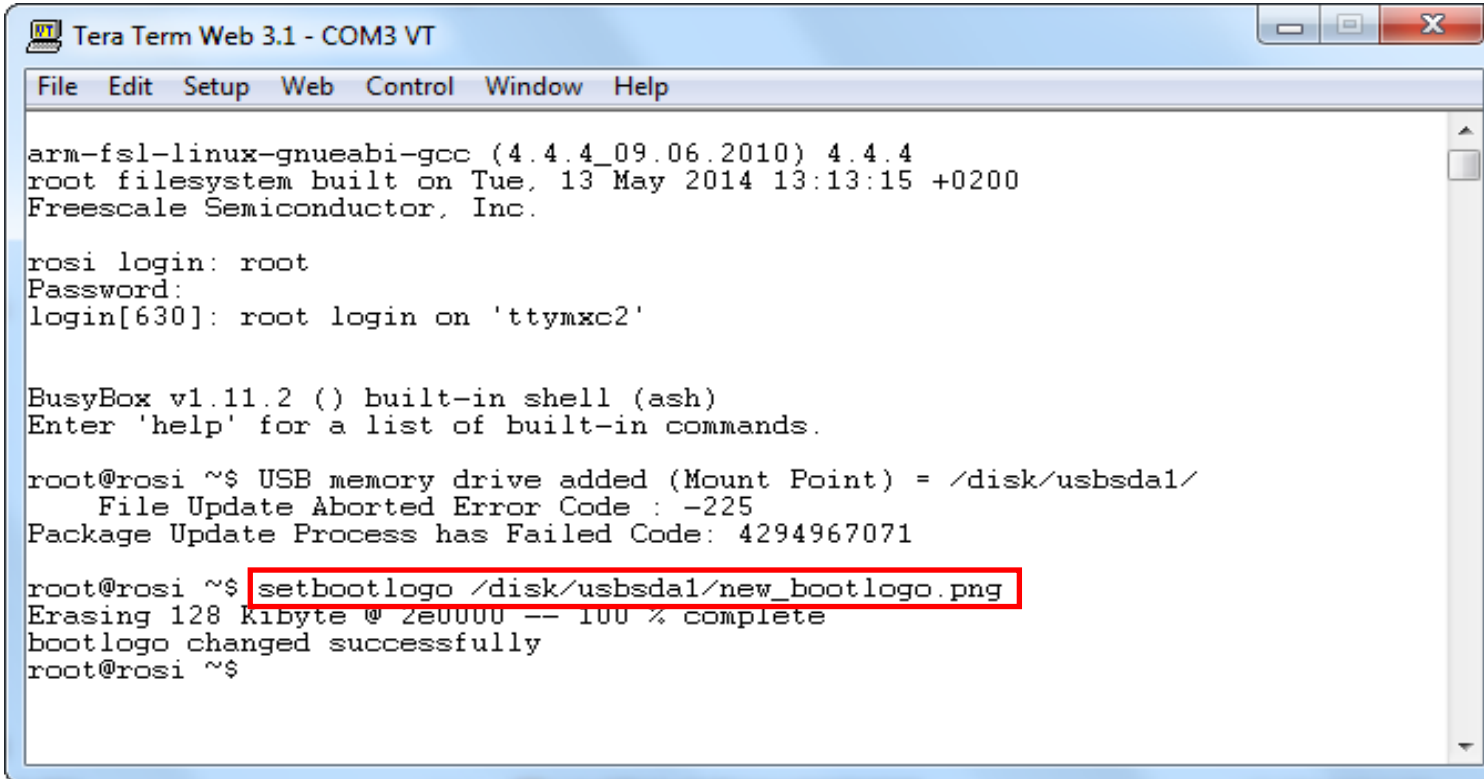
root@rosi ~$ █
```

To log in, use

username **root**

password “**opusa3**” or “**opusa6**”

## 7.2 – FAQ – Change the boot logo



```
Tera Term Web 3.1 - COM3 VT
File Edit Setup Web Control Window Help
arm-fsl-linux-gnueabi-gcc (4.4.4_09.06.2010) 4.4.4
root filesystem built on Tue, 13 May 2014 13:13:15 +0200
Freescale Semiconductor, Inc.

rosi login: root
Password:
login[630]: root login on 'ttymxc2'

BusyBox v1.11.2 () built-in shell (ash)
Enter 'help' for a list of built-in commands.

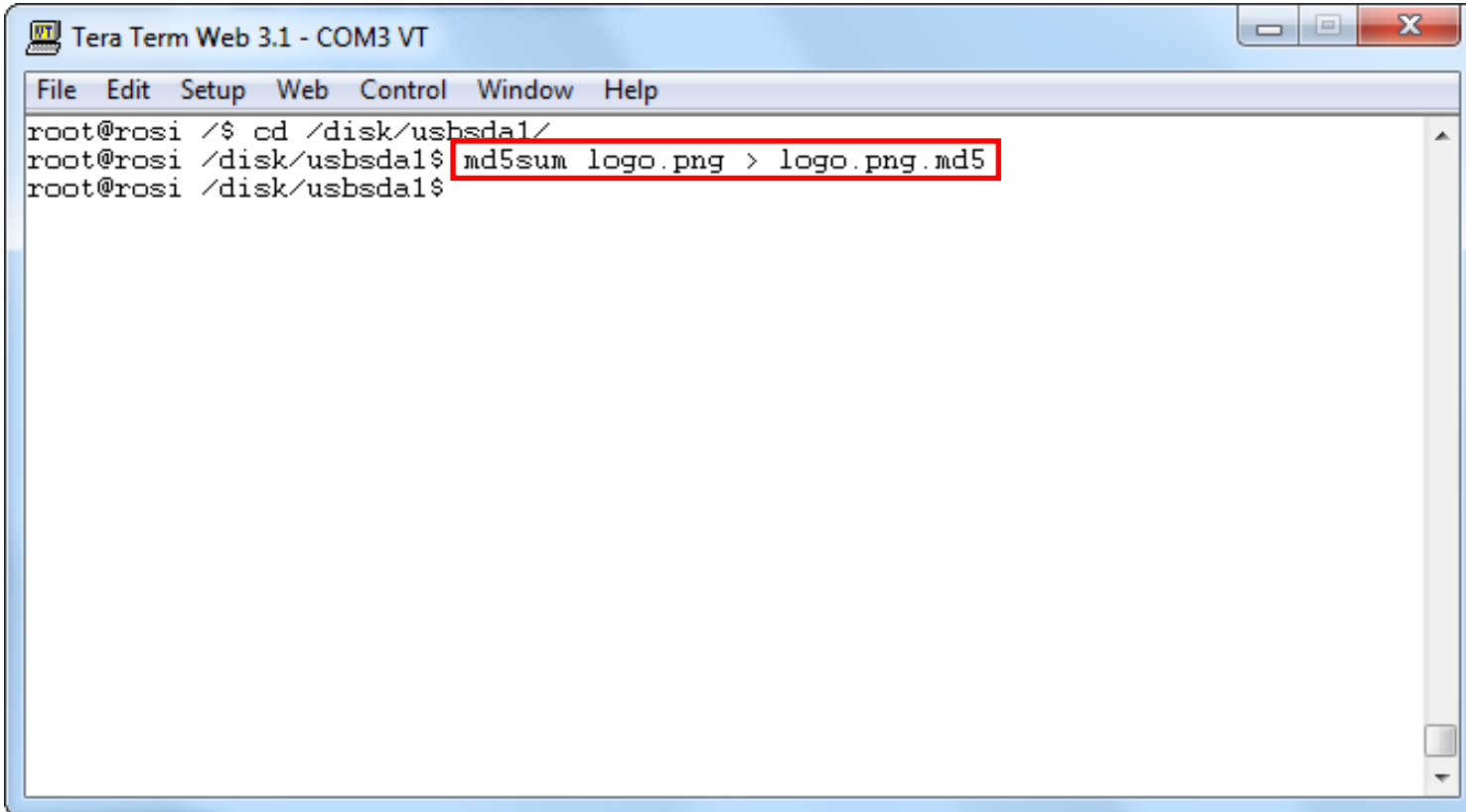
root@rosi ~$ USB memory drive added (Mount Point) = /disk/usbsda1/
File Update Aborted Error Code : -225
Package Update Process has Failed Code: 4294967071

root@rosi ~$ setbootlogo /disk/usbsda1/new_bootlogo.png
Erasing 128 kbyte @ ze0000 -- 100 % complete
bootlogo changed successfully
root@rosi ~$
```

Put the image (png file, resolution 480x272 for A3, 800 x 480 for A6) on a USB stick and connect it to the device.

Use the command  
“**setbootlogo** *imagepath*”  
to change the boot logo of the device

## 7.2 – FAQ – Change the update boot logo

A screenshot of a terminal window titled "Tera Term Web 3.1 - COM3 VT". The terminal shows a sequence of commands: "root@rosi /\$ cd /disk/usbsda1/", "root@rosi /disk/usbsda1\$ md5sum logo.png > logo.png.md5", and "root@rosi /disk/usbsda1\$". The command "md5sum logo.png > logo.png.md5" is highlighted with a red rectangular box. The terminal window has a menu bar with "File", "Edit", "Setup", "Web", "Control", "Window", and "Help".

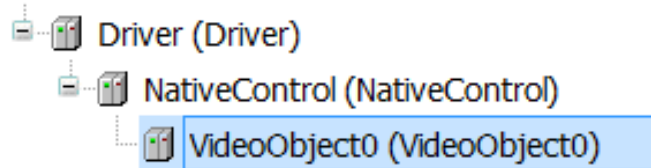
```
root@rosi /$ cd /disk/usbsda1/
root@rosi /disk/usbsda1$ md5sum logo.png > logo.png.md5
root@rosi /disk/usbsda1$
```

To change the boot logo for the update process, you need to create a checksum file.

Rename the image to “logo.png”, put it on a USB stick and connect it to the device.

Use the command “**md5sum** logo.png > logo.png.md5” to create the md5 checksum file.

## 7.3 – FAQ – How can I display a camera image?



VideoObject0 x

Parameter	Type	Value	Default Value	Unit	Description
Set Channel	Enumeration of BOOL	inactive	inactive		Set channel on startup
Channel	INT	1			Video channel of the device
Set Rotate	Enumeration of BOOL	inactive	inactive		Set rotate on startup
Rotate	UINT				Rotation value, must be a multiple of 90
Set FlipH	Enumeration of BOOL	inactive	inactive		Set flip horizontal on startup
FlipH	Enumeration of BOOL				Flip horizontal
Set FlipV	Enumeration of BOOL	inactive	inactive		Set flip vertical on startup
FlipV	Enumeration of BOOL				Flip vertical
Set Brightn...	Enumeration of BOOL	inactive	inactive		Set brightness on startup
Brightness	SINT				Brightness value
Set Contrast	Enumeration of BOOL	inactive	inactive		Set contrast on startup

First you need to add a VideoObject to the Driver module

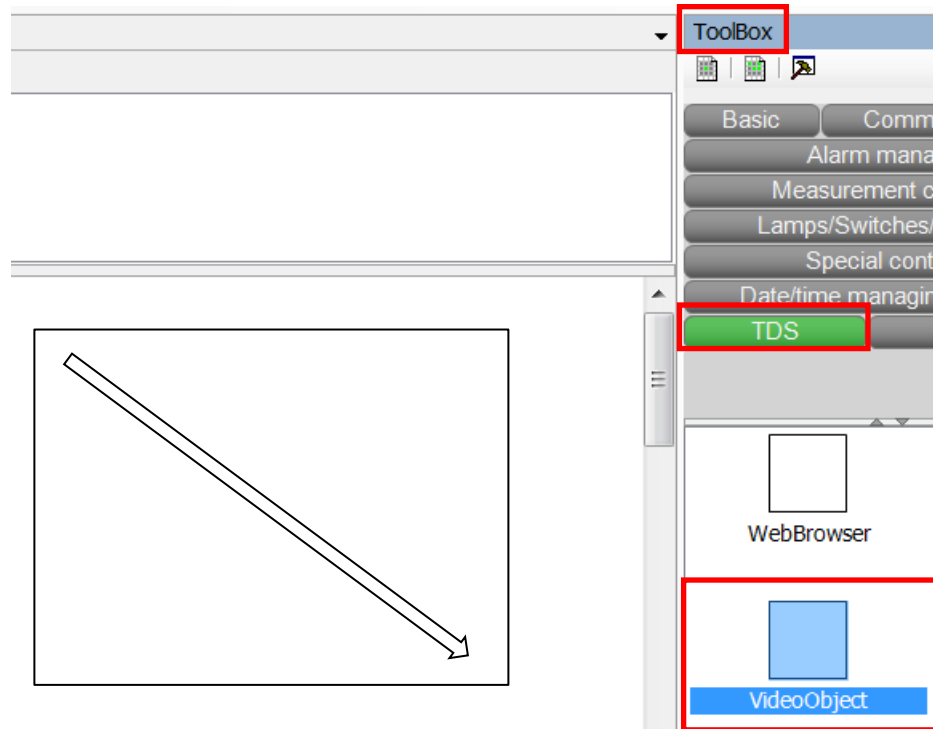
Double click it and make the necessary settings (especially the Channel)

VideoObject0 is for camera inputs 1 and 3

VideoObject1 is for camera input 2

- VideoObject0 Channel 0 -> camera input 1
- VideoObject0 Channel 1 -> camera input 3
- VideoObject1 Channel 0 -> camera input 2

## 7.3 – FAQ – How can I display a camera image?



Now select the VideoObject in the ToolBox, in the category TDS and draw a rectangle in the visu window

The object is a frame of the type video object

## 7.3 – FAQ – How can I display a camera image?

Elementname	GenElemInst_3
Type of element	Frame
Clipping	<input type="checkbox"/>
Show frame	No frame
Scaling type	Anisotropic
Deactivate the backgr...	<input type="checkbox"/>
References	Configure...
TargetContainer.TD...	
data	VideoObject0
Position	
X	1
Y	2
Width	360
Height	272
Center	
Color	

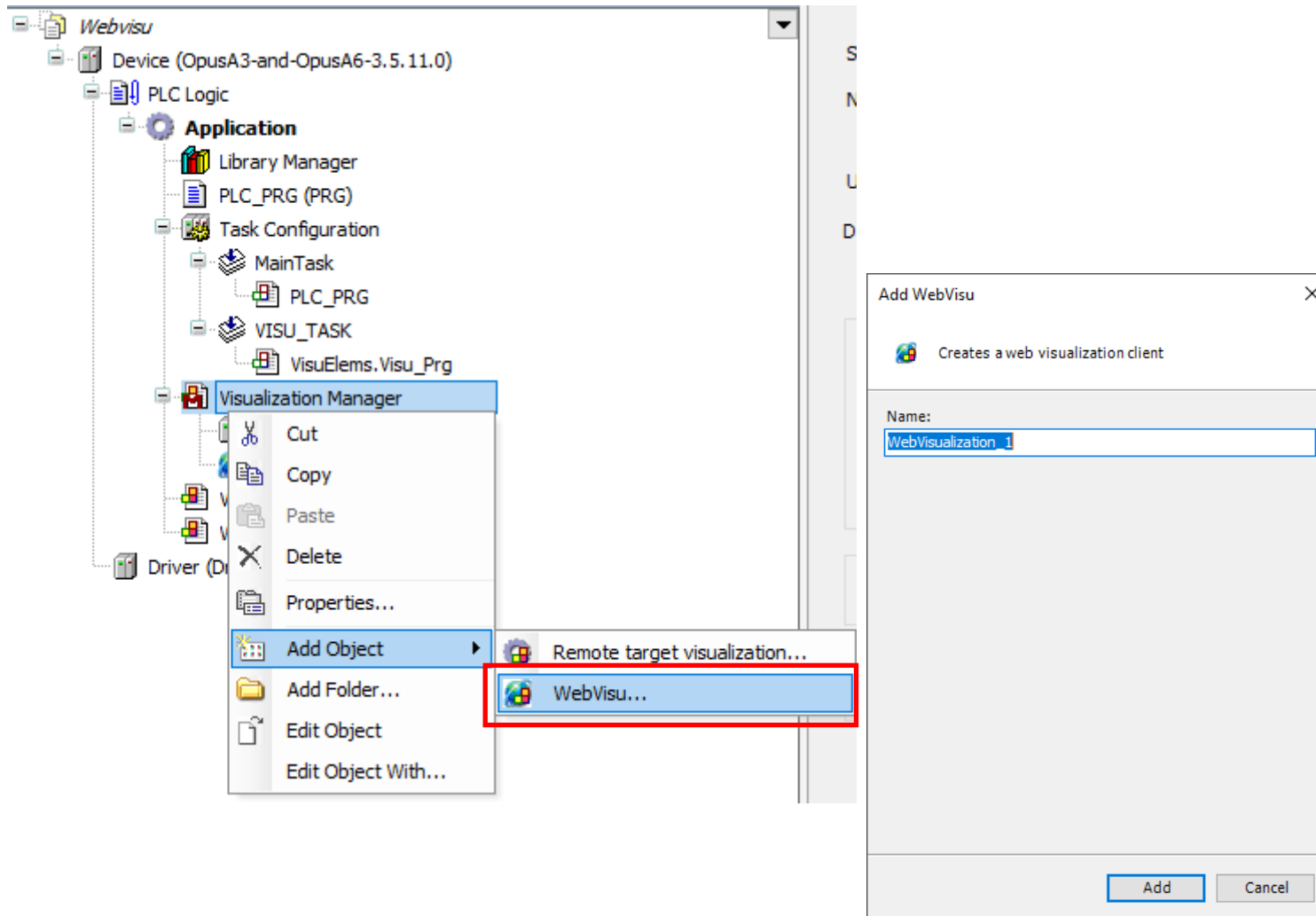
In the properties of the VideoObject frame, set References -> TargetContainer -> data to the name of the VideoObject you created

Resize the frame to a 4:3 size ratio for a correct video image





## 7.4 – FAQ – How can I use the Codesys Web Visu?



To add a Web Visu, right-click on the Visualization Manager, then Add Object, then WebVisu...

Give it a name of your choice and click Add in the dialog

## 7.4 – FAQ – How can I use the Codesys Web Visu?

WebVisualization x

Start Visualization: Visualization

Name of .htm file: webvisu

Use as default page

Update rate (ms): 200

Default communication buffer size: 50000

[Show used visualizations](#)

Scaling options

Fixed  Isotropic  Anisotropic

Use scaling options for dialogs

Client width: 1280

Client height: 1024

Presentation options

Antialiased drawing

Default text input

Input with: Touchscreen

Name of the visu that is shown as the first web visu

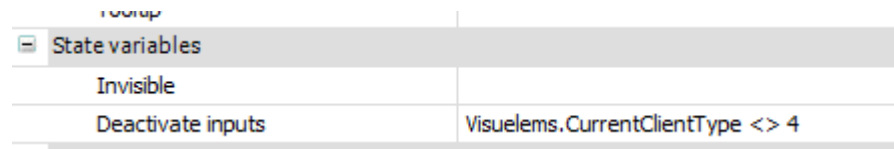
Name of the web visu html file

Open the WebVisualization to configure the web visu

For a mirror functionality use the same visu as in the target visualization

If you want to have a different visu for the web visu, enter a specific visualization in “Start Visualization.”

## 7.4 – FAQ – How can I use the Codesys Web Visu?



Group	
State variables	
Invisible	
Deactivate inputs	Visuelems.CurrentClientType <> 4

To keep objects from being used in the web visu, use the property Deactivate inputs for each input object

Visuelems.CurrentClientType determines where the visu is running:

- Device (target): Visuelems.CurrentClientType=4
- PC Codesys: Visuelems.CurrentClientType=1
- Webvisu: Visuelems.CurrentClientType=8

## 7.4 – FAQ – How can I use the Codesys Web Visu?

To access the web visu, you need to be in the same network as the device, or create a VPN. The call in the browser has to be done like this:

<IP\_of\_device>:8080/<web visu name>

e.g.

192.168.135.6:8080/webvisu.htm