# Welcome

to our

# CODESYS Training

Nadeem Khan

# 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 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?! | cable color |
|---|---|
| A/DI 1 – 4 – Analog / Digital Inputs | brown,green,yellow,grey |
| DO 1 – 3 – Digital Outputs | white,lilac,blue |
| RS232 – serial console of the Linux system | RxD-red-blue,TxD-red,GND-pink |
| CAN1/2 – CAN ports | CAN1-pink(high),blue(low)<br>CAN2-green(high),yellow(low) |
| USB – USB | VCC-red,GND-black,Low-white,High-green |
| Clamp 30 – Battery plus | red |
| Clamp 15 – Ignition (Should be connected into Clamp 30) | red-blue |
| GND / Car GND – Ground (Should be connected if separate clamps)* | GND-black,CarGND-white, black connector(s) |
| Serv_EN – Service Enable | 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.

Boot loader

- boot_opusa3_2.1.0.bin
- boot_opusa3_2.1.0.bin.md5

Linux kernel

- linux_opusa3_2.1.0.bin
- linux_opusa3_2.1.0.bin.md5

Root file system

- rootfs_opusa3_2.1.0.squashfs
- rootfs_opusa3_2.1.0.squashfs.md5

Service image

- service_opusa3_2.1.0.bin
- service_opusa3_2.1.0.bin.md5

Service files to update from OS 1

- boot_egspro_1.1.1.bin
- boot_egspro_1.1.1.bin.md5
- pmic_egspro_1.1.sud
- pmic_egspro_1.1.sud.md5
- rootfs_egspro_1.1.1-we-x11-qt5.tar.gz
- rootfs_egspro_1.1.1-we-x11-qt5.tar.gz.md5

Power management firmware

Service project is an extra application

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

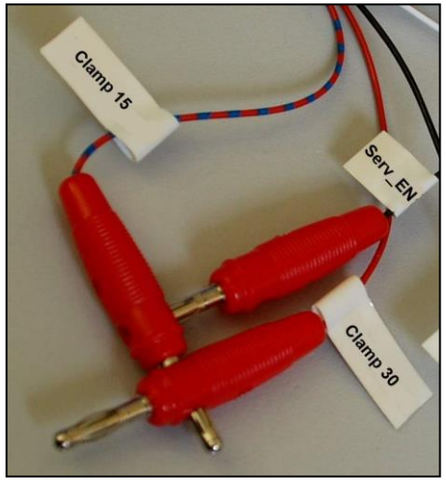
- user_opusa3_3.5.8.6.tar.gz
- user_opusa3_3.5.8.6.tar.gz.md5
- user_egspro_3.5.8.6.tar.gz
- user_egspro_3.5.8.6.tar.gz.md5
- user2_egspro_3.5.8.6_Service.tar.gz
- user2_egspro_3.5.8.6_Service.tar.gz.md5

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

- logo_opusa3_2.x.png
- logo_opusa3_2.x.png.md5
- logo_egspro_a6-te-landscape.bmp
- logo_egspro_a6-te-landscape.bmp.md5
- recovery_egspro_a6-te-landscape.bmp
- recovery_egspro_a6-te-landscape.bmp.md5

# 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

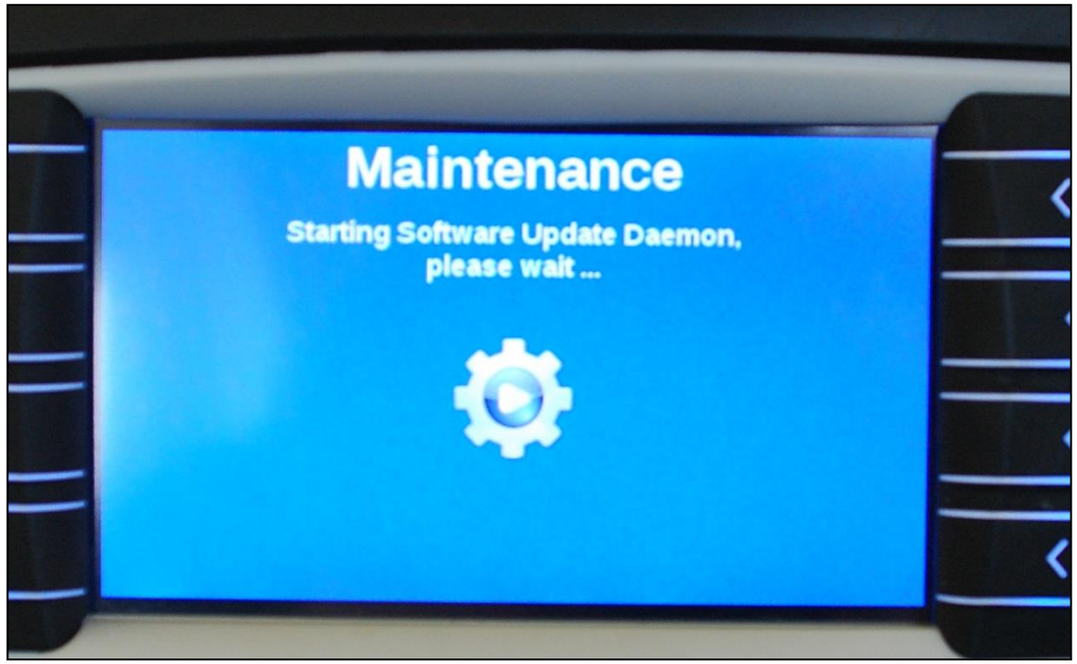
3. Connect the USB stick with the device

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



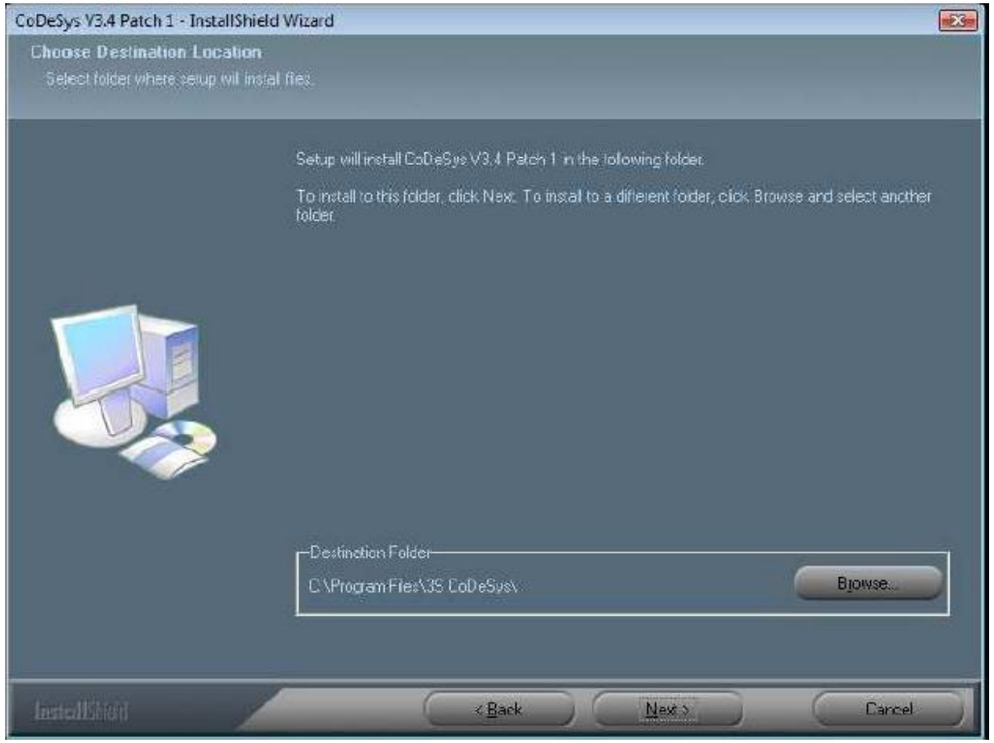

When you see this screen (or similar), let go of the keys or disconnect the clamp service enable

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



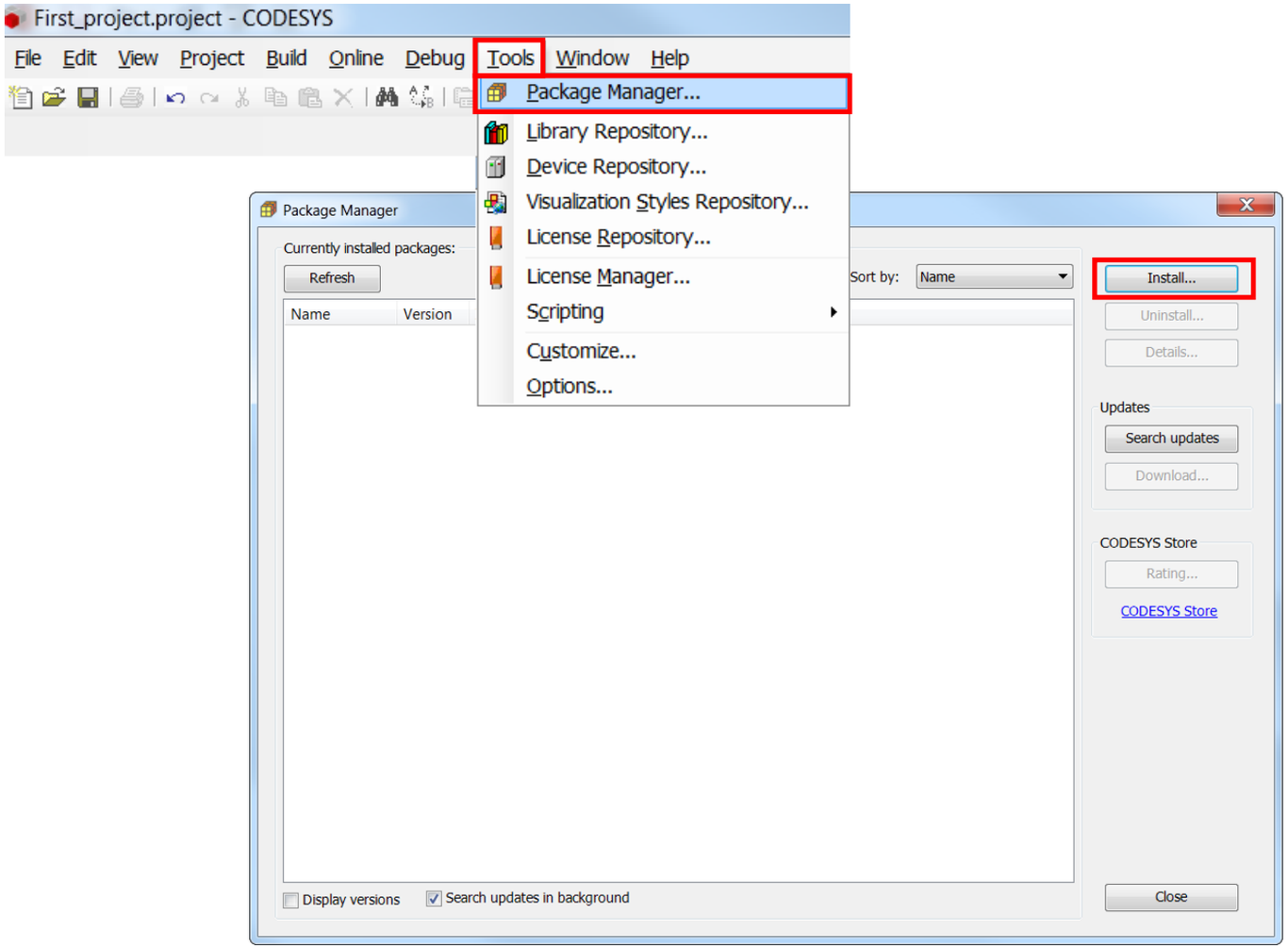

Install the CODESYS IDE with administrator rights

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

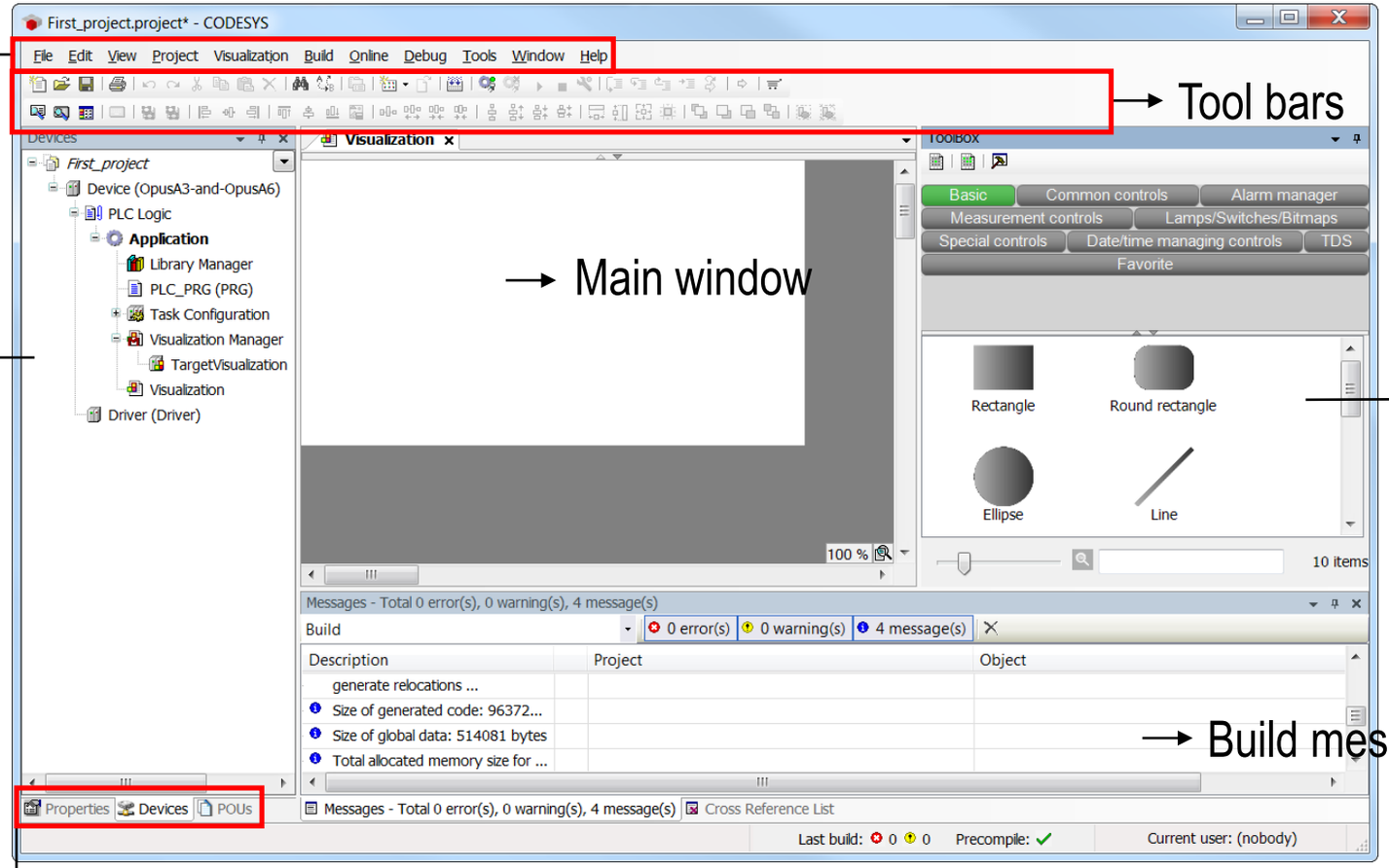



In Codesys, select the menu *Tools -> Package Manager…*

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

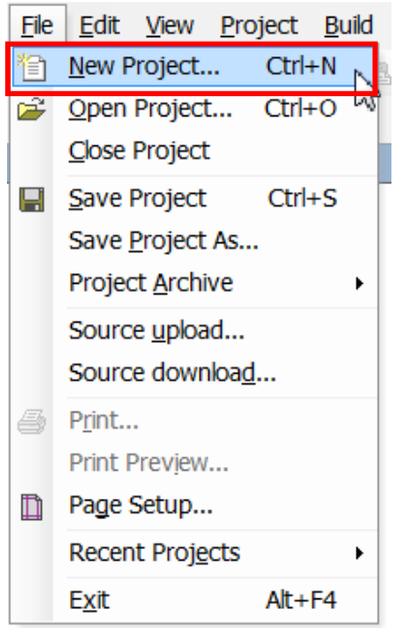
# Agenda

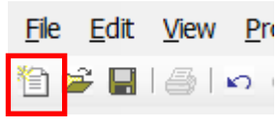
1. Introduction
2. Updating your device
3. Installing the CODESYS IDE
4. **First project**
5. CAN communication
6. Extended agenda
7. FAQ

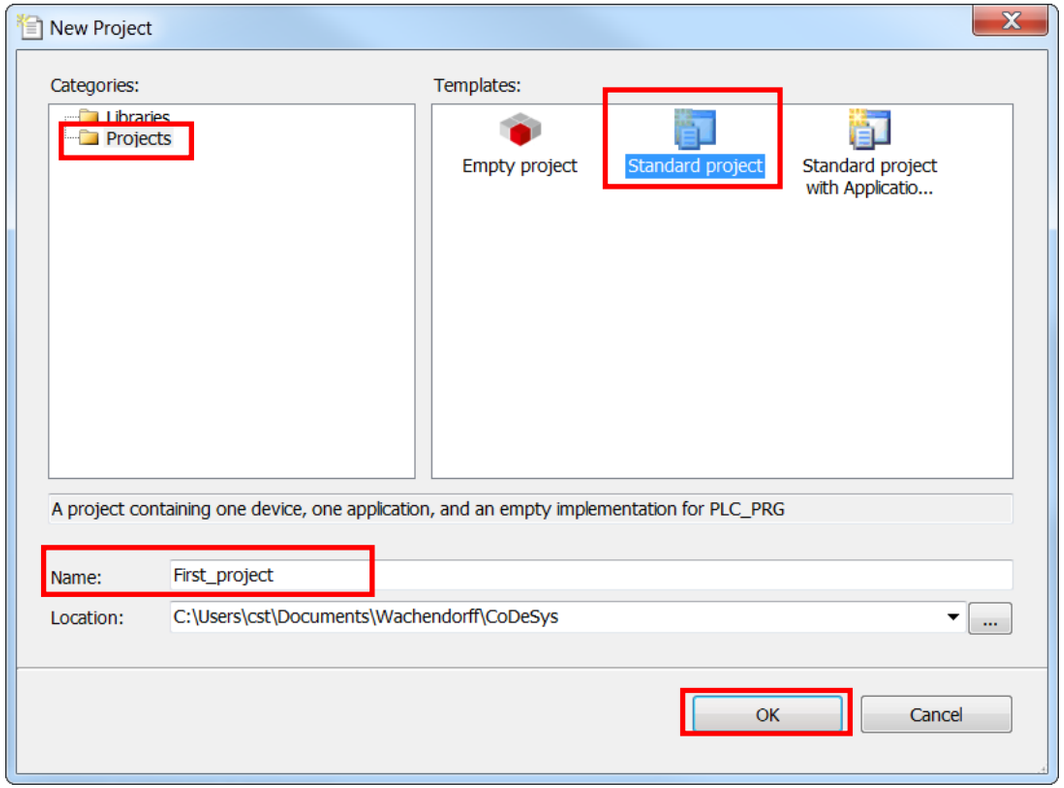
# 4. The first project

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



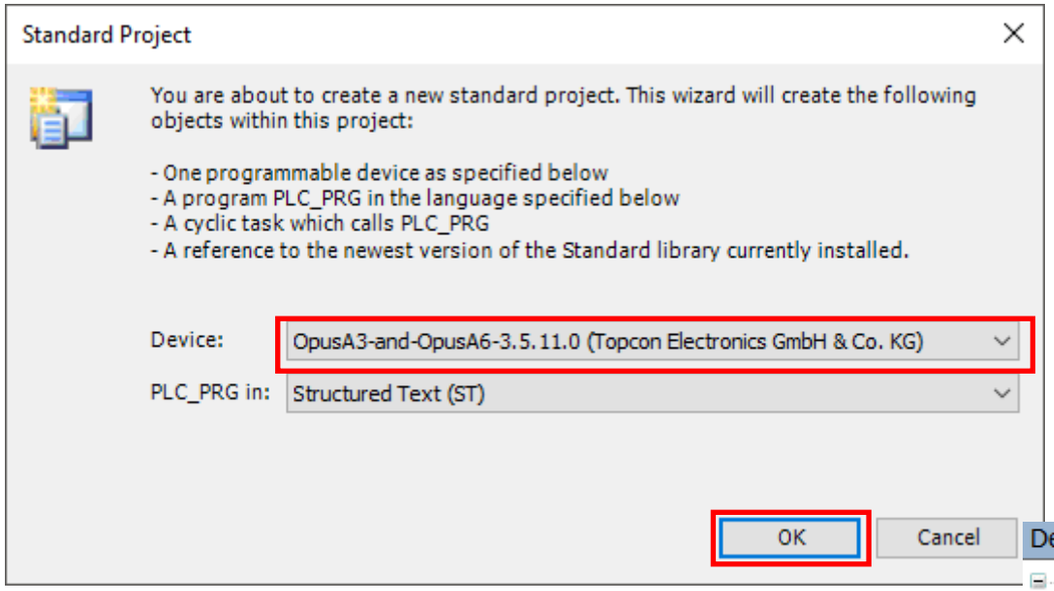



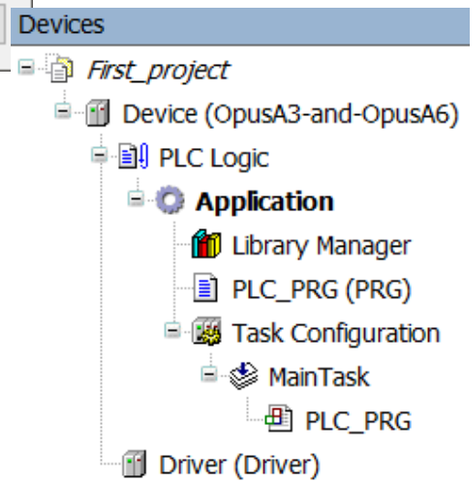




To create a new project,
click on *File -> New Project…*
press the *New Project* icon
or press CTRL + N

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

# 4.1 The first project – Create a new project

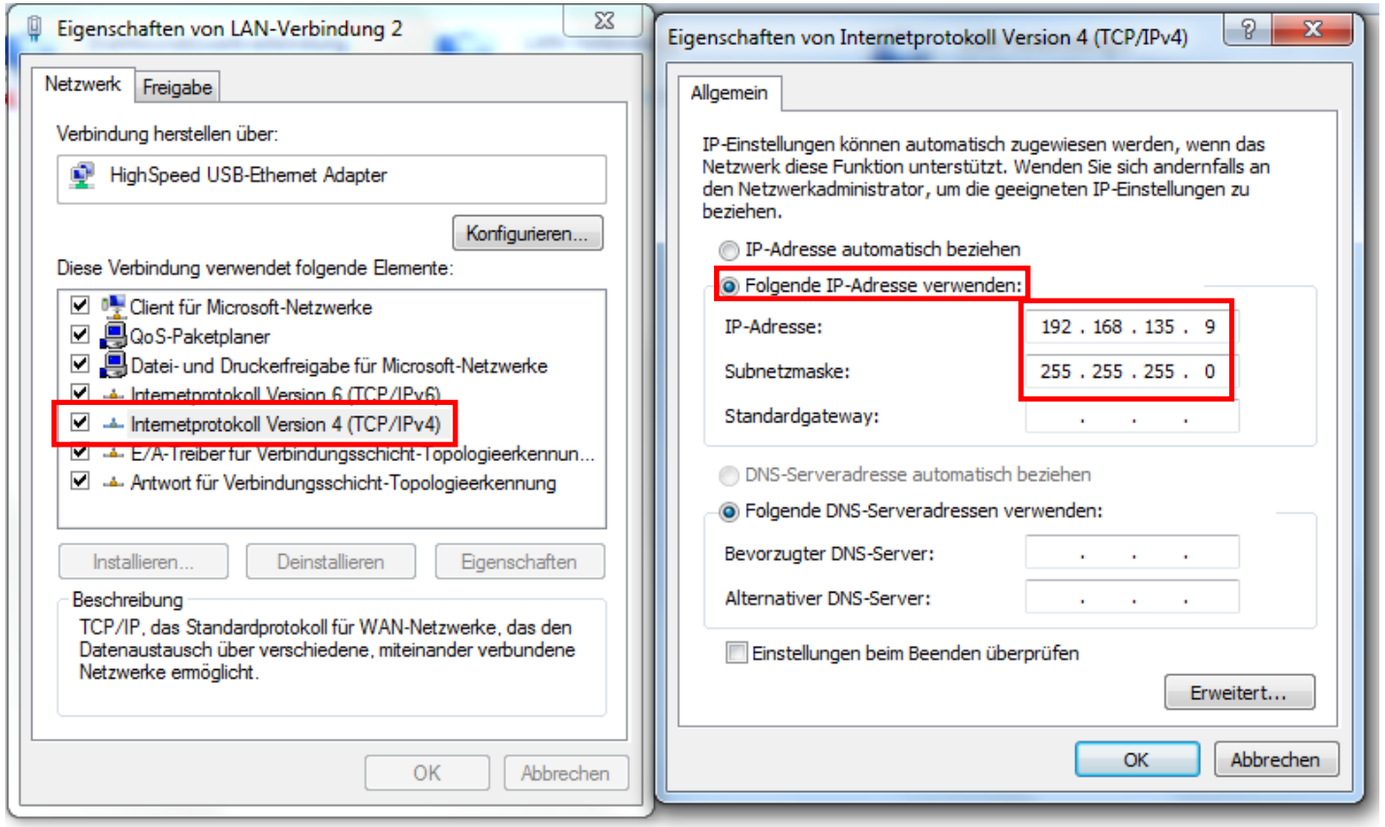



Select the matching device description and choose Structured Text (ST) as the default programming language

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

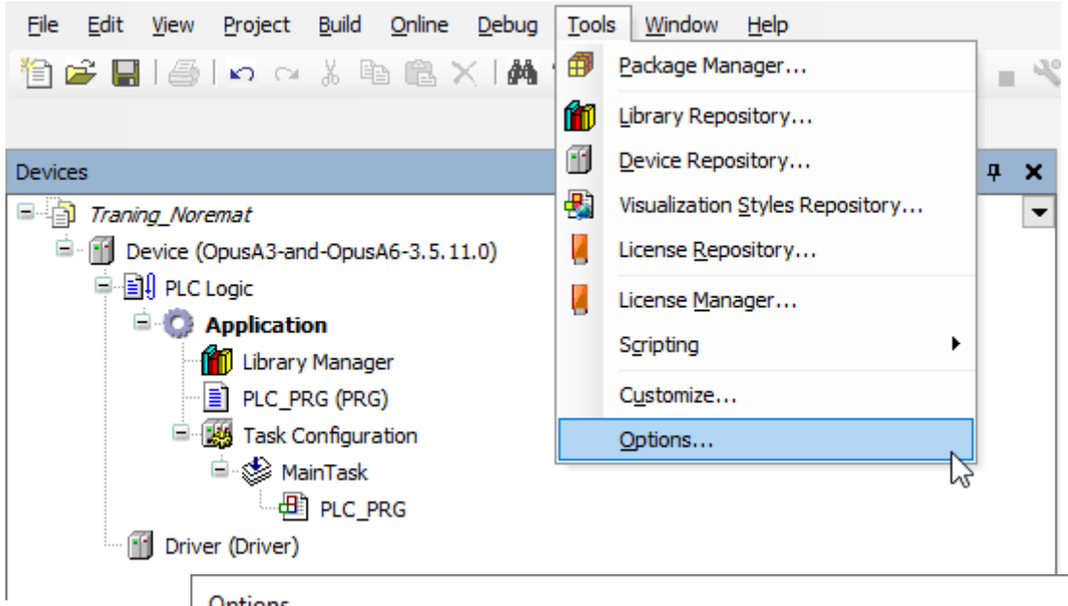
The computer IP has to be in the same subnet as the displays

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

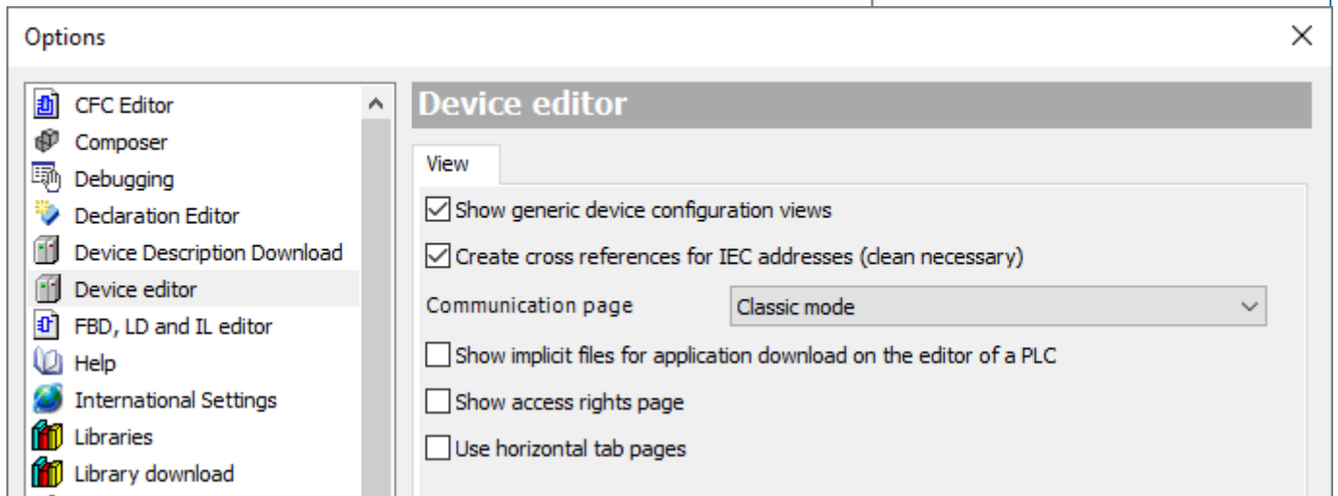
192.168.135.6

Now connect the running device with your PC over Ethernet

# 4.2 The first project – Configuration – Device connection

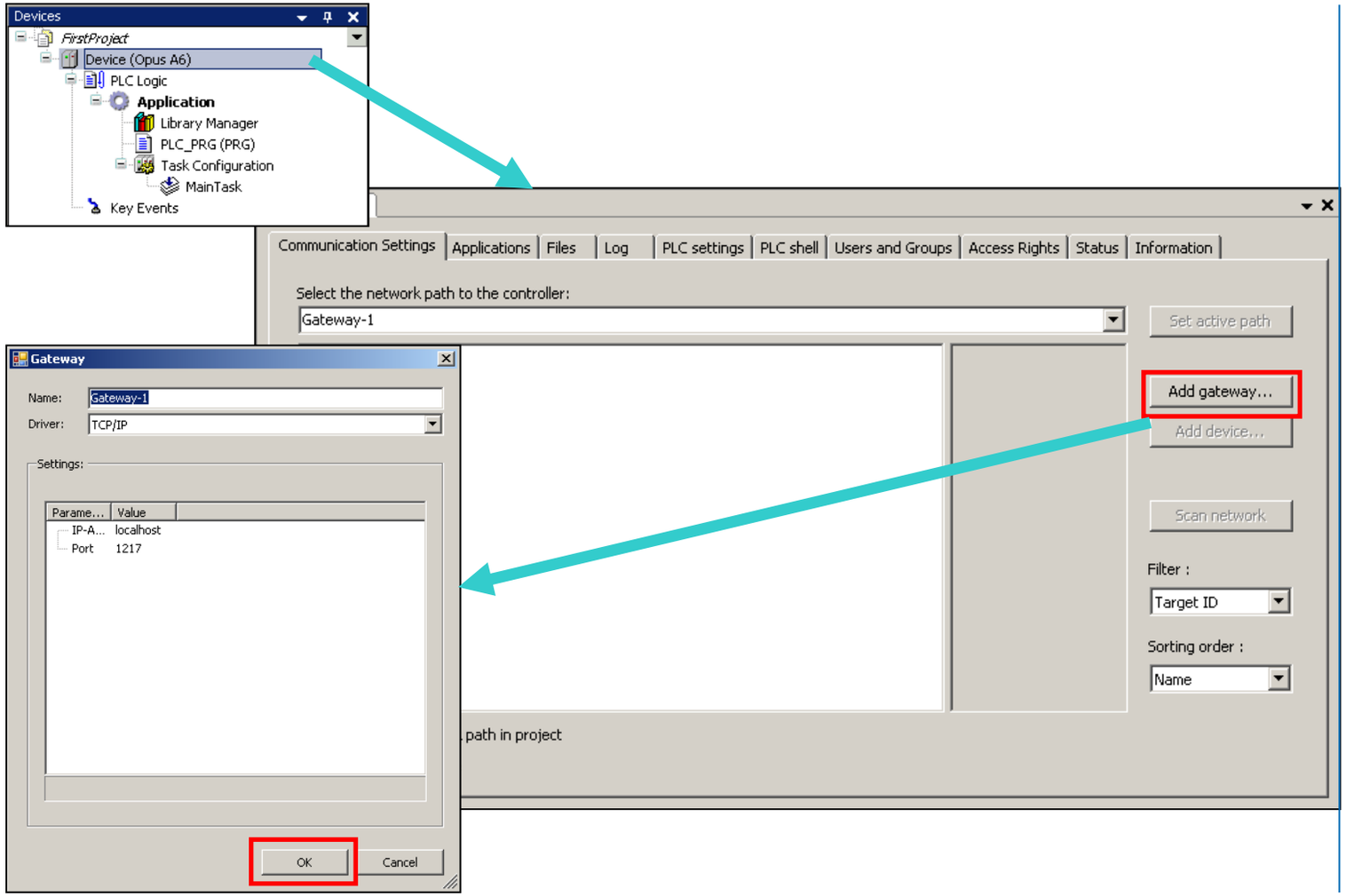



Go to the menu *Tools -> Options…*




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

# 4.2 The first project – Configuration – Device connection



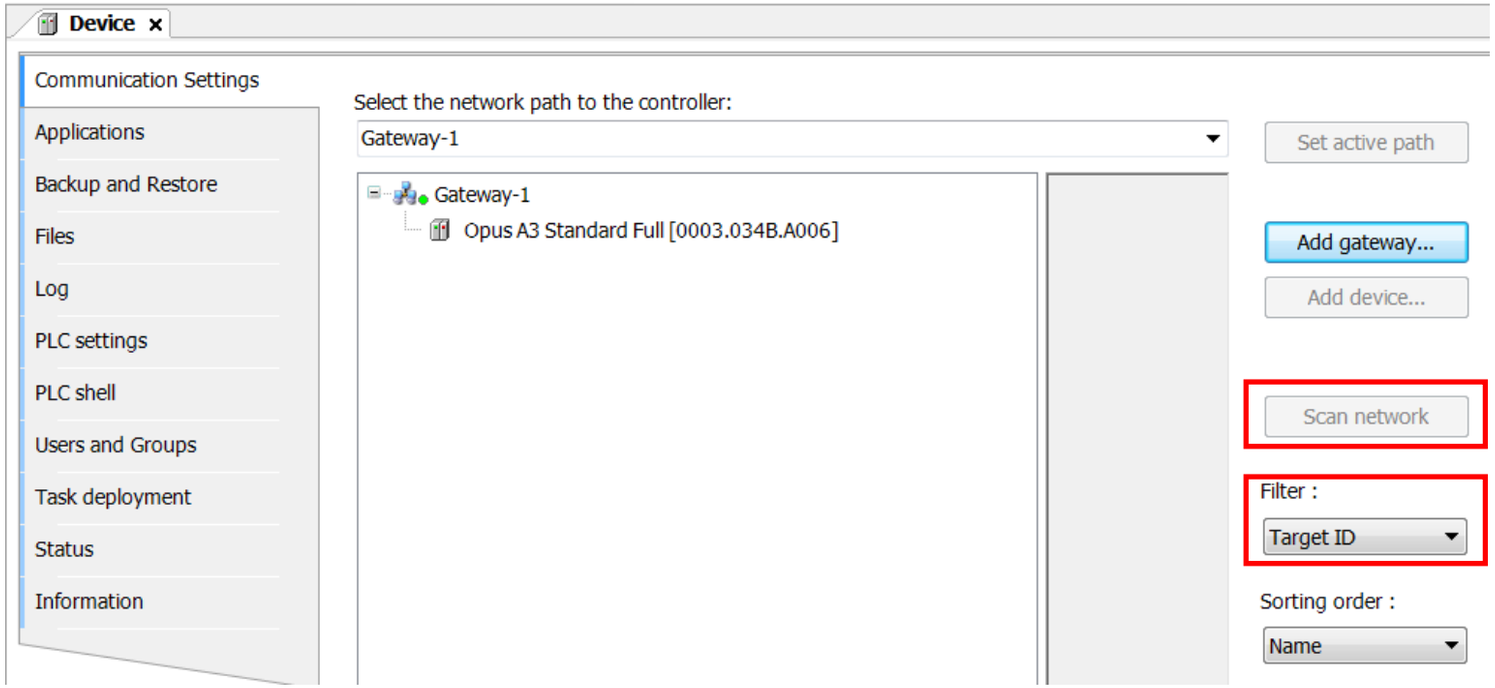

Double click on *Device*

Click on *Add gateway…*

Click *OK*

Or select the existing Gateway

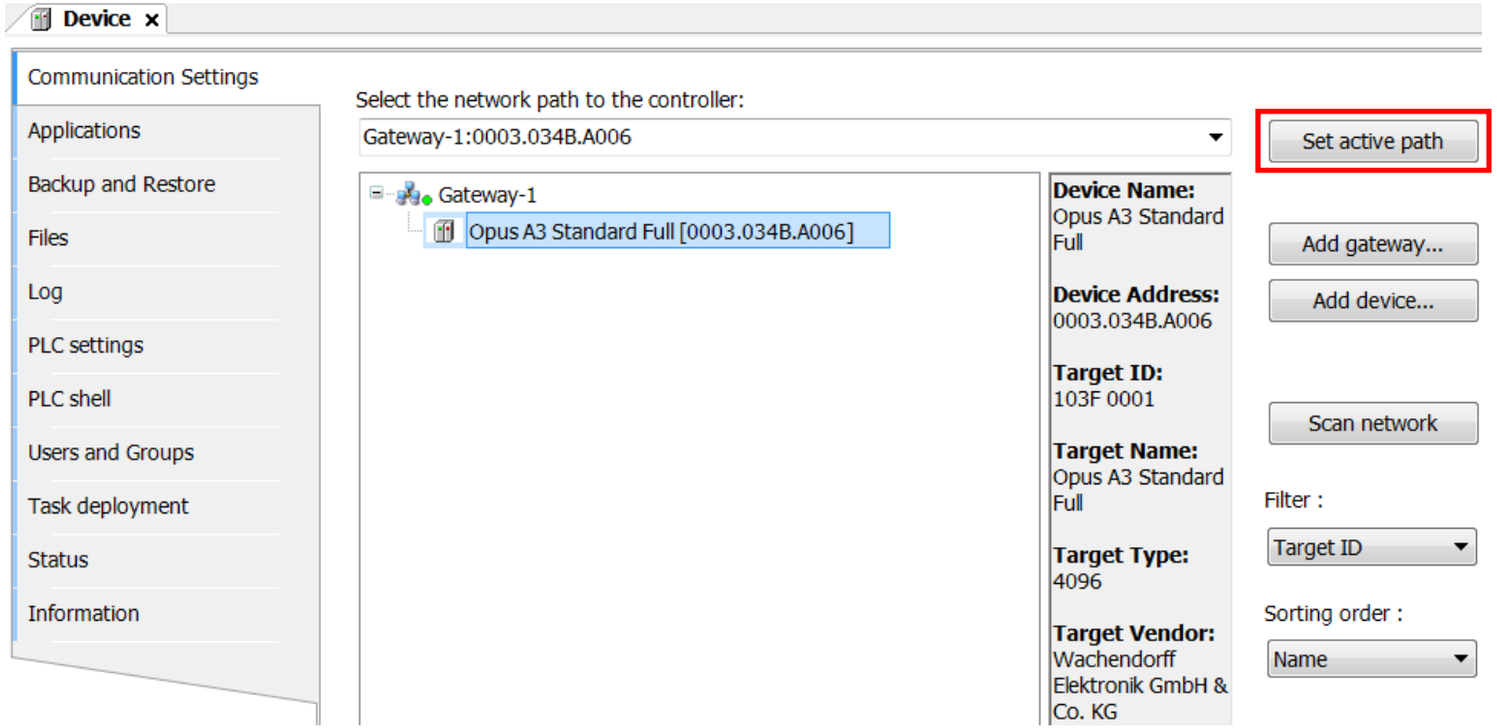
# 4.2 The first project – Configuration – Device connection




Select the gateway and click on *Scan network*

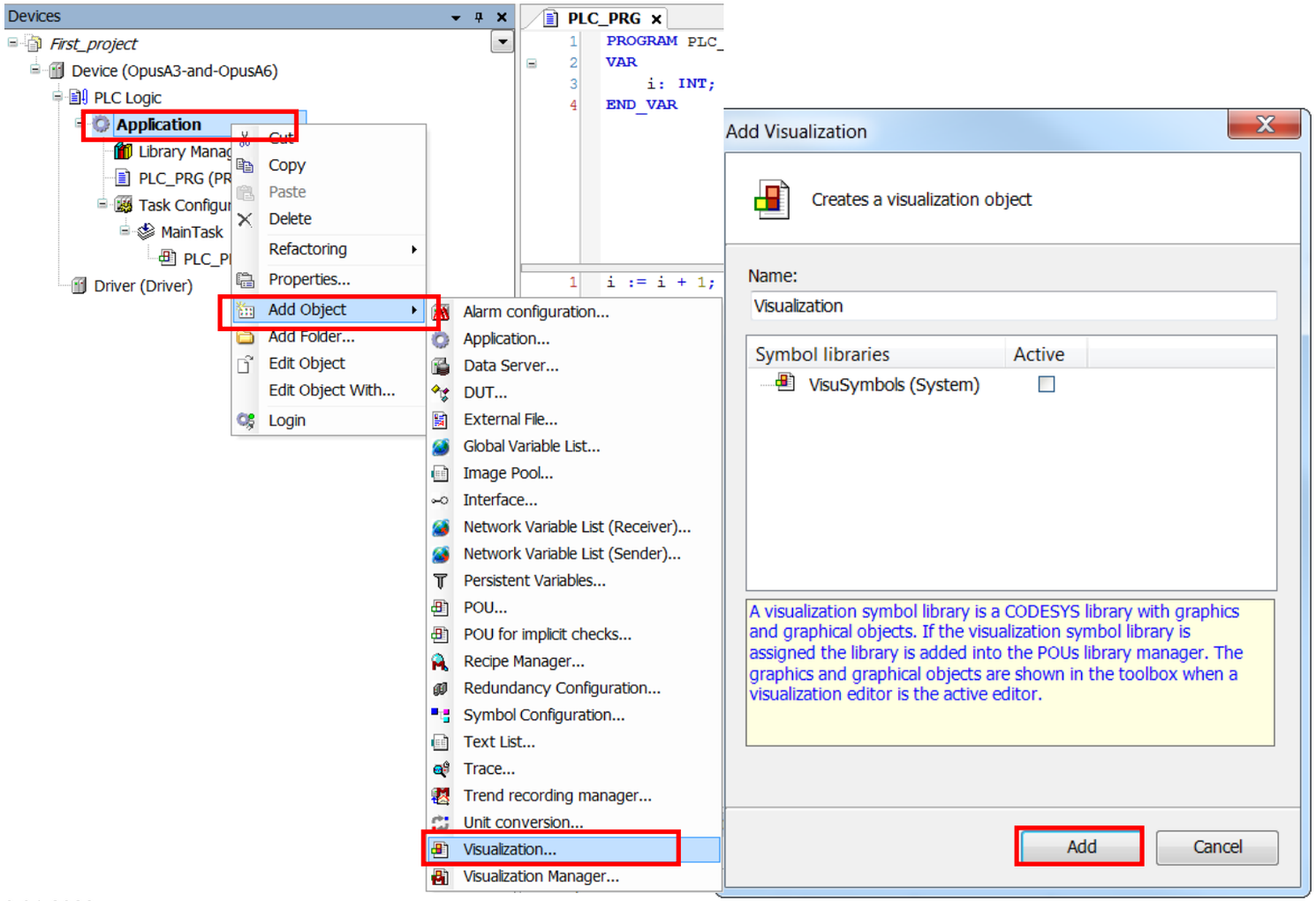
If the device isn't found, set the *Filter* to *None*

# 4.2 The first project – Configuration – Device connection




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

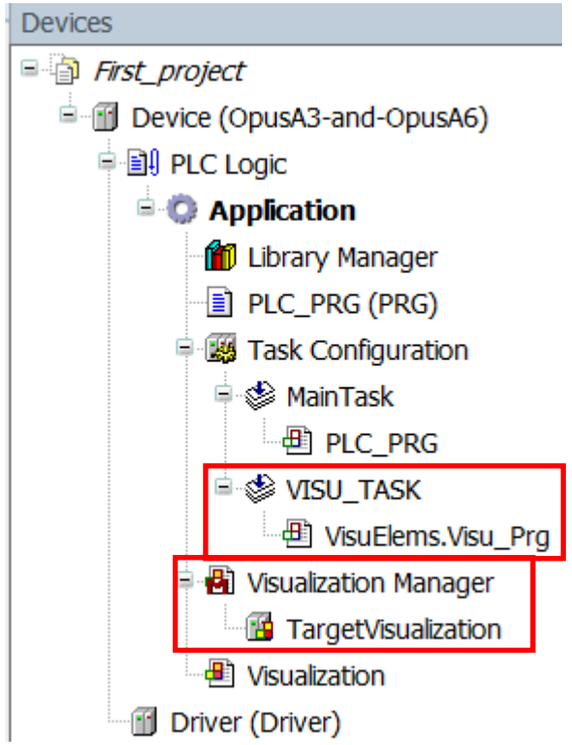
# 4.4 The first project – The first visualization




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

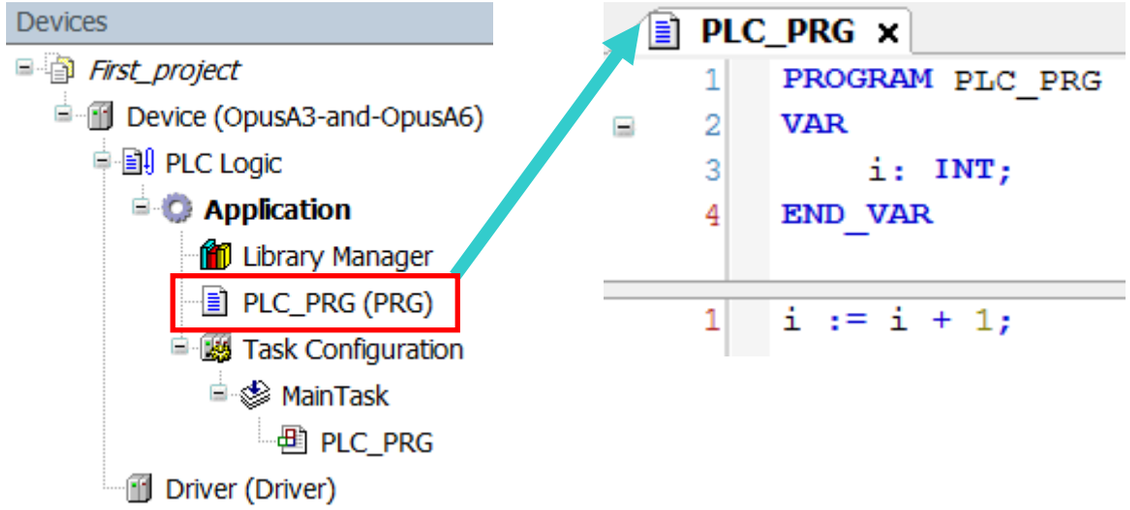
CODESYS adds some components automatically:

- VISU_TASK in the Task Configuration

- Visualization Manager

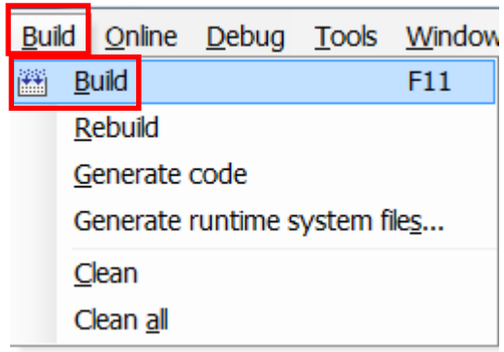
- TargetVisualization in the Visualization Manager

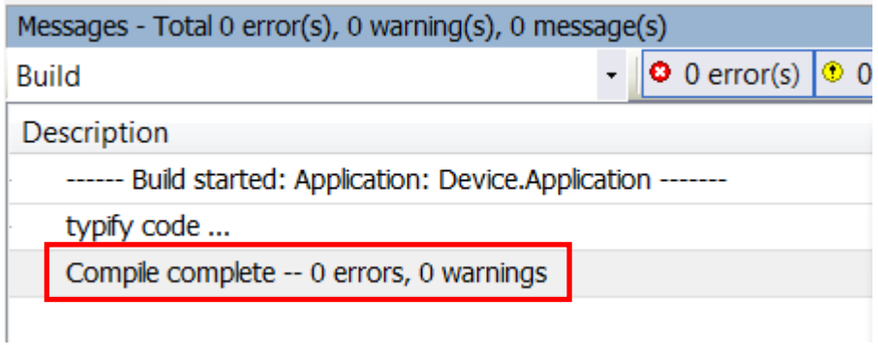
# 4.3 The first project – The first program




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



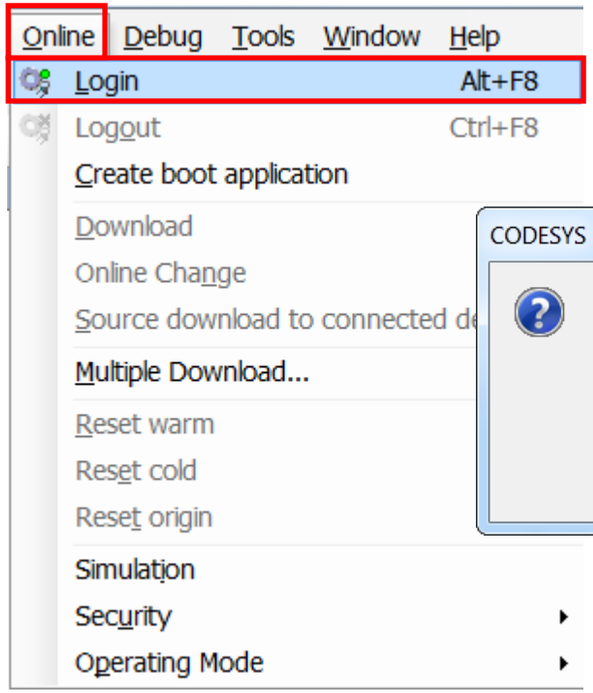



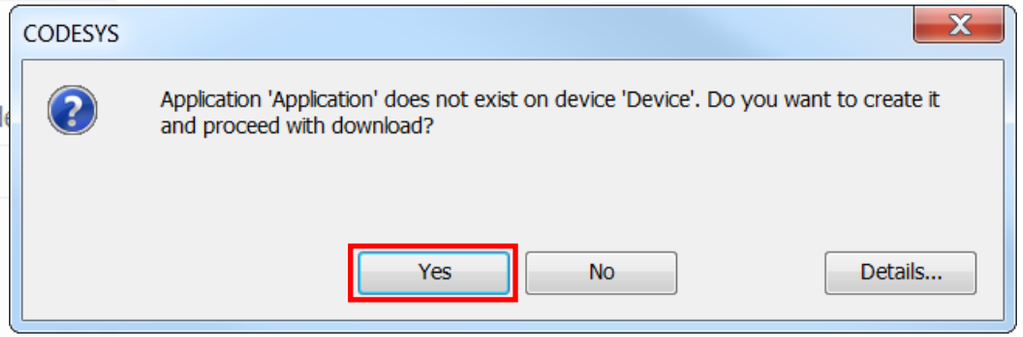

Compile the project by clicking in the menu *Build -> Build* or by pressing *F11*

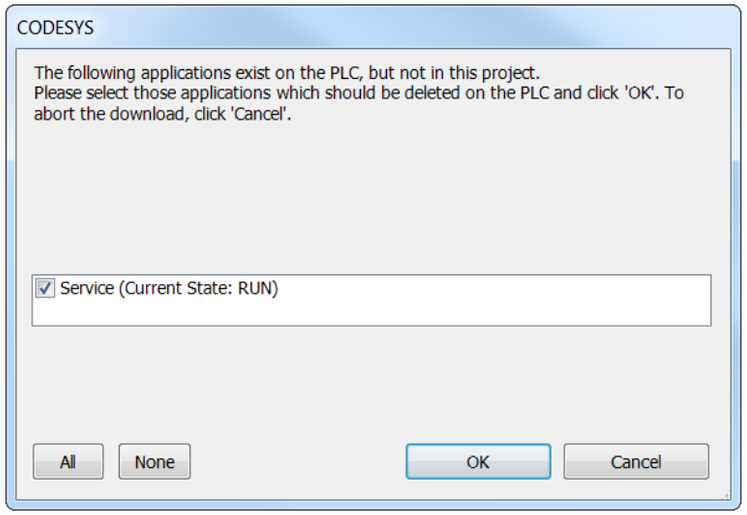
Message box should show the message Compile complete – 0 errors, 0 warnings

Our first program built successfully, big applause

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



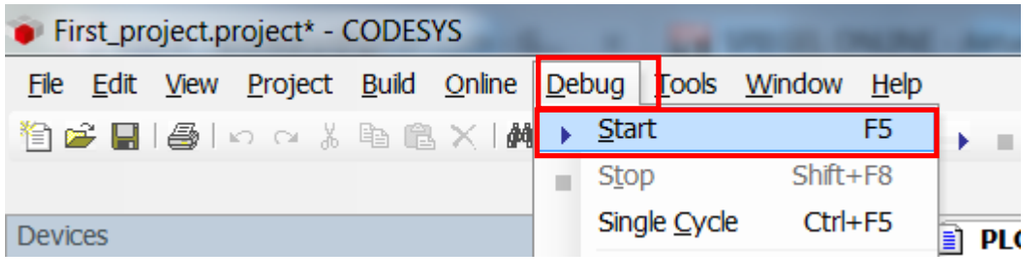




Download the project with *Login* in the menu *Online* (or by pressing Alt+F8) or press the icon in the tool bar

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)

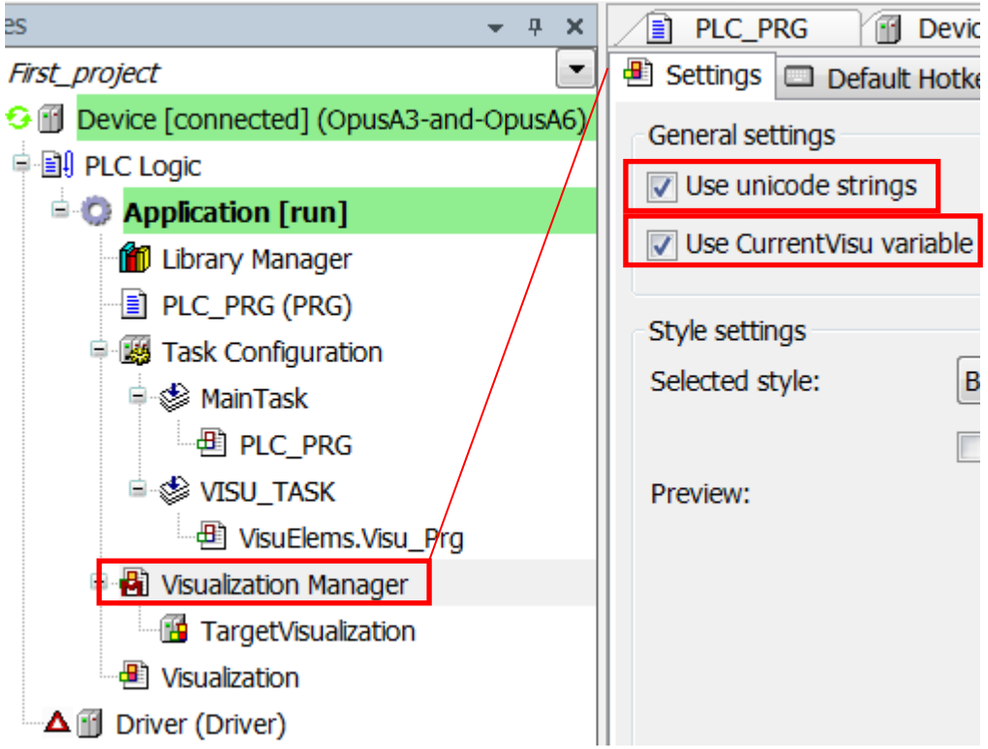
->you can see the counter rising on the PC

-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

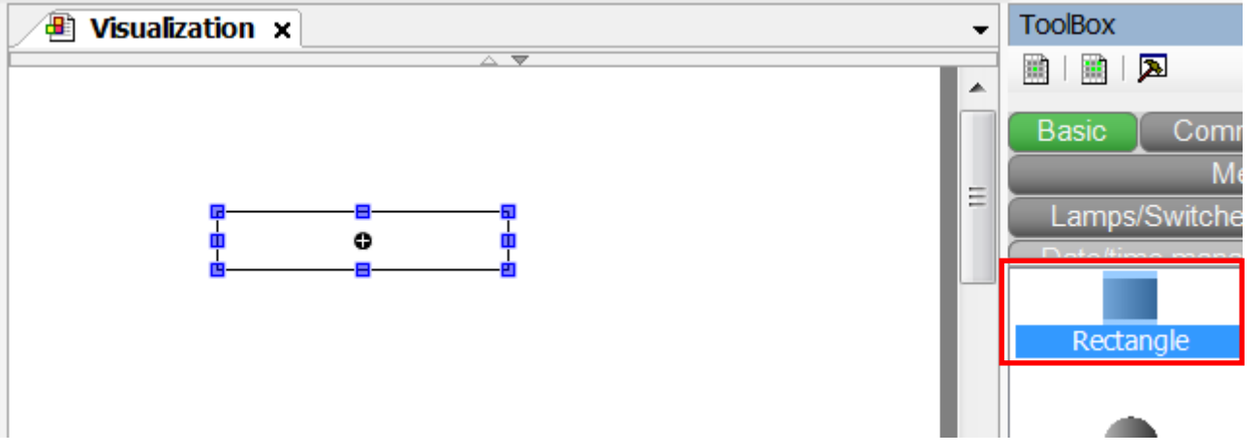


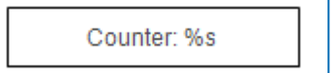

Make the settings according to the screenshots

Visualization Manager:
- Use unicode strings
- Use CurrentVisu variable

These are general recommendations for projects

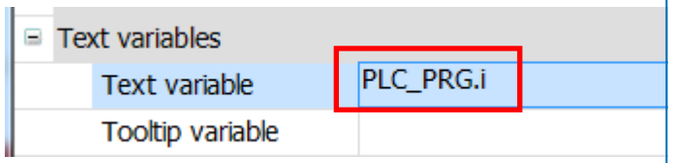
# 4.4 The first project – The first visualization – Rectangle




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*

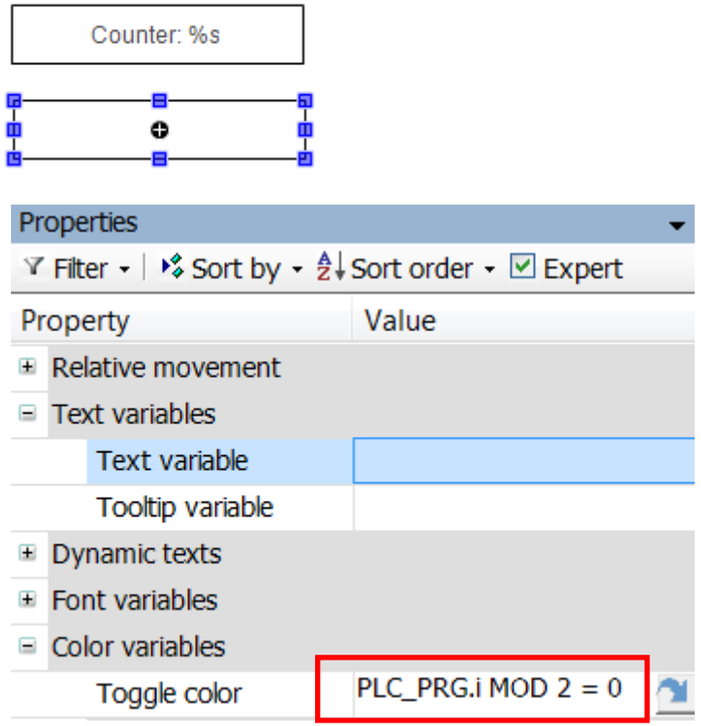
Log in again to update the application on the device, then start the application (F5)
Log out afterwards to continue working

# 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 zerol) |
| 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<br>Syntax: %\|<alignment><minimal width><accuracy>]<br>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



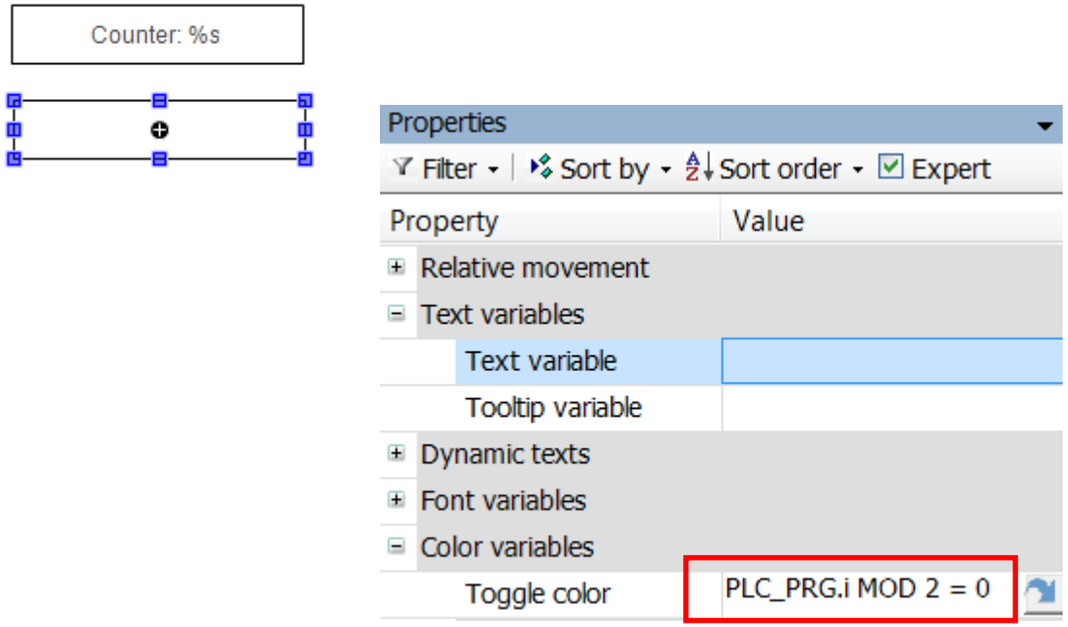

Add a second rectangle

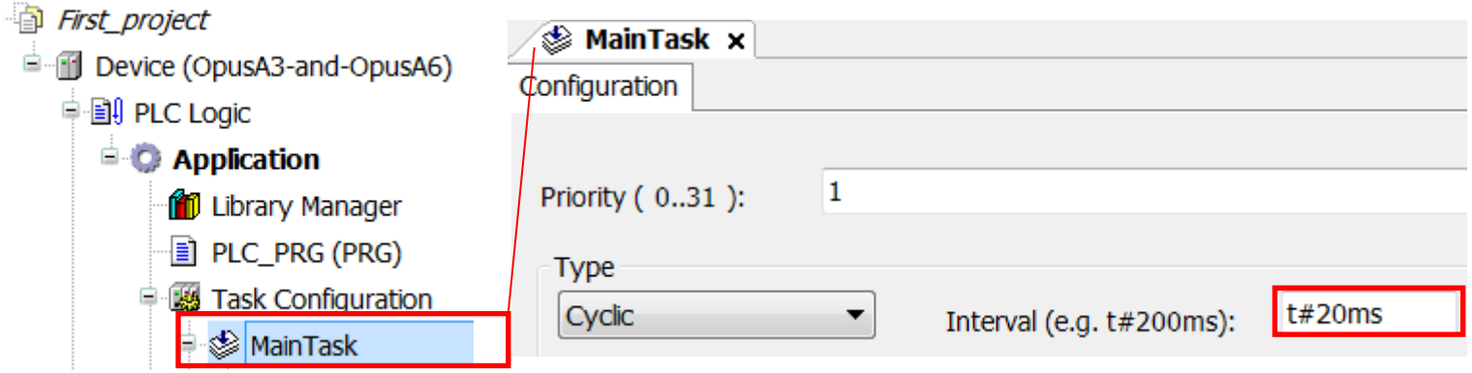
Add the expression
PLC_PRG.i MOD 2 = 0
to the
*Color Variables* -> *Toggle color* property

Download and start the application

Please note the flashing screen.

# 4.5 The first project – Tasks and timings




Add a second rectangle

Add the expression
PLC_PRG.i 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

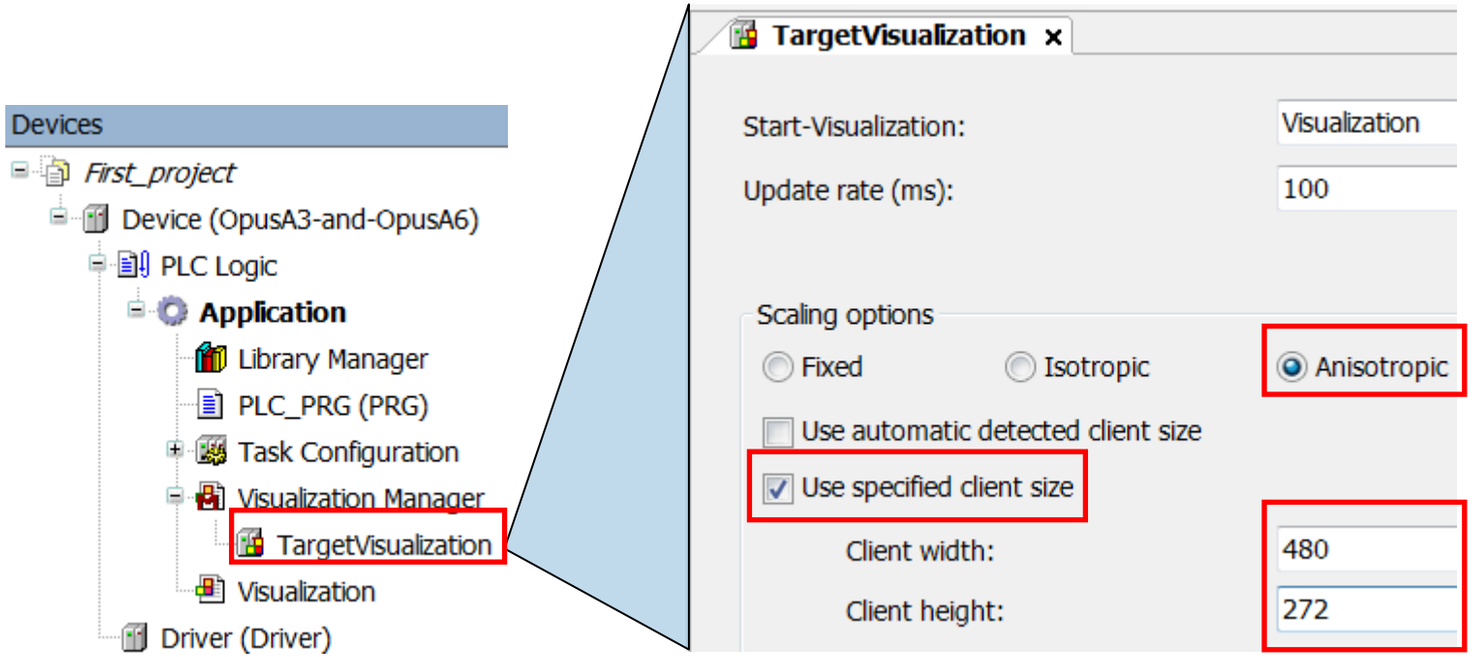
# 4.5 The first project – Tasks and timings

-> 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



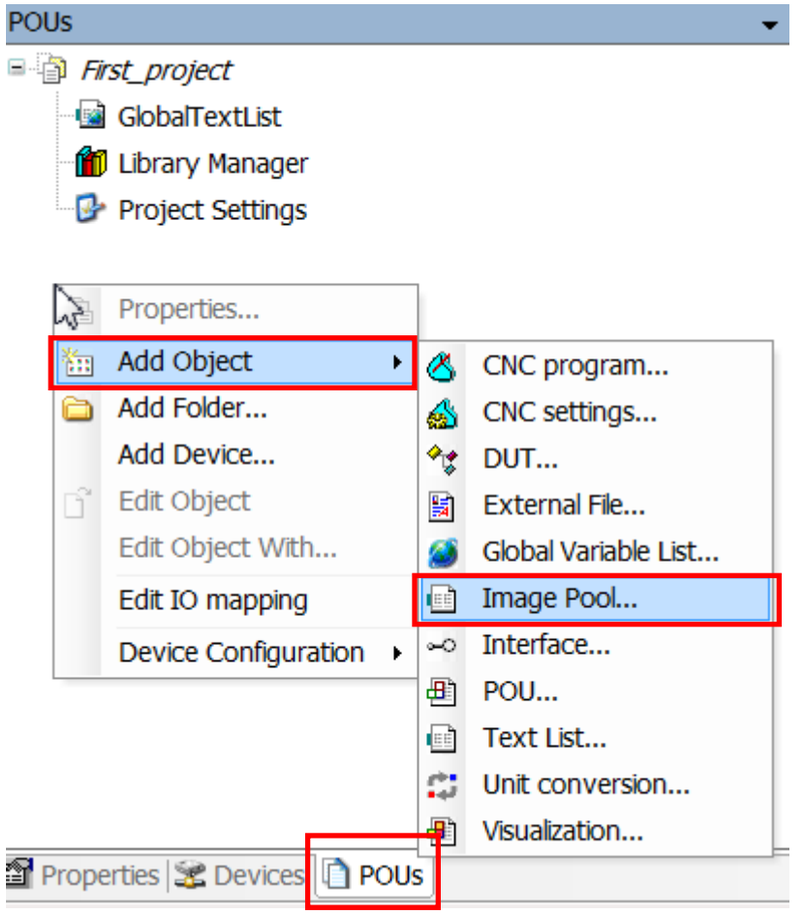

To know how big your screen is in the editor

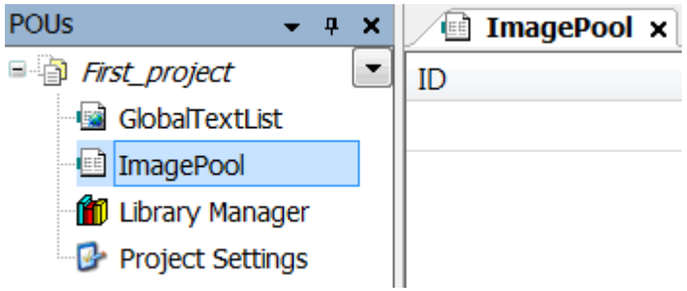
make the settings according to the screenshot

-> 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



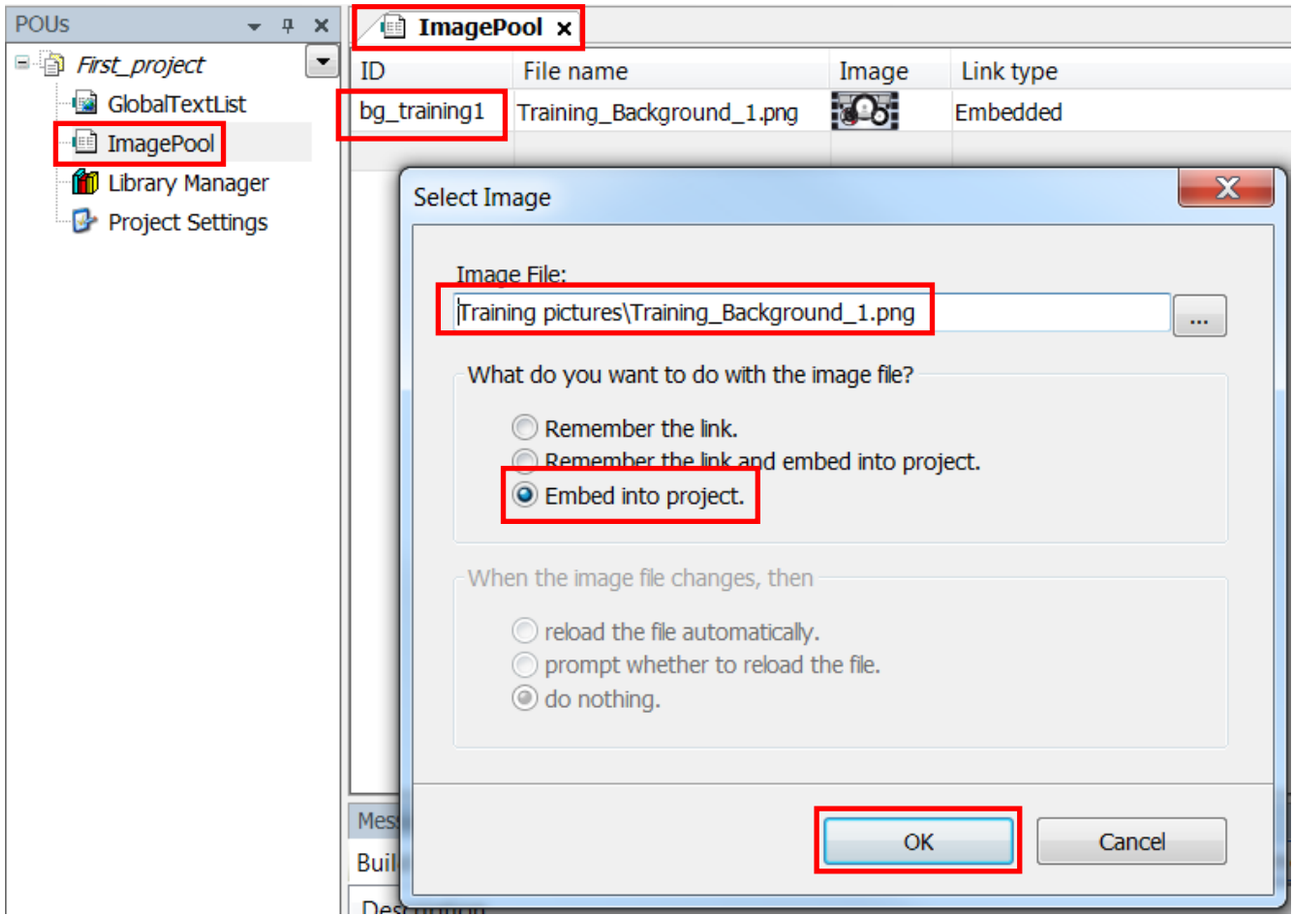




Change to the POU tab

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

Double-click the ImagePool to open it

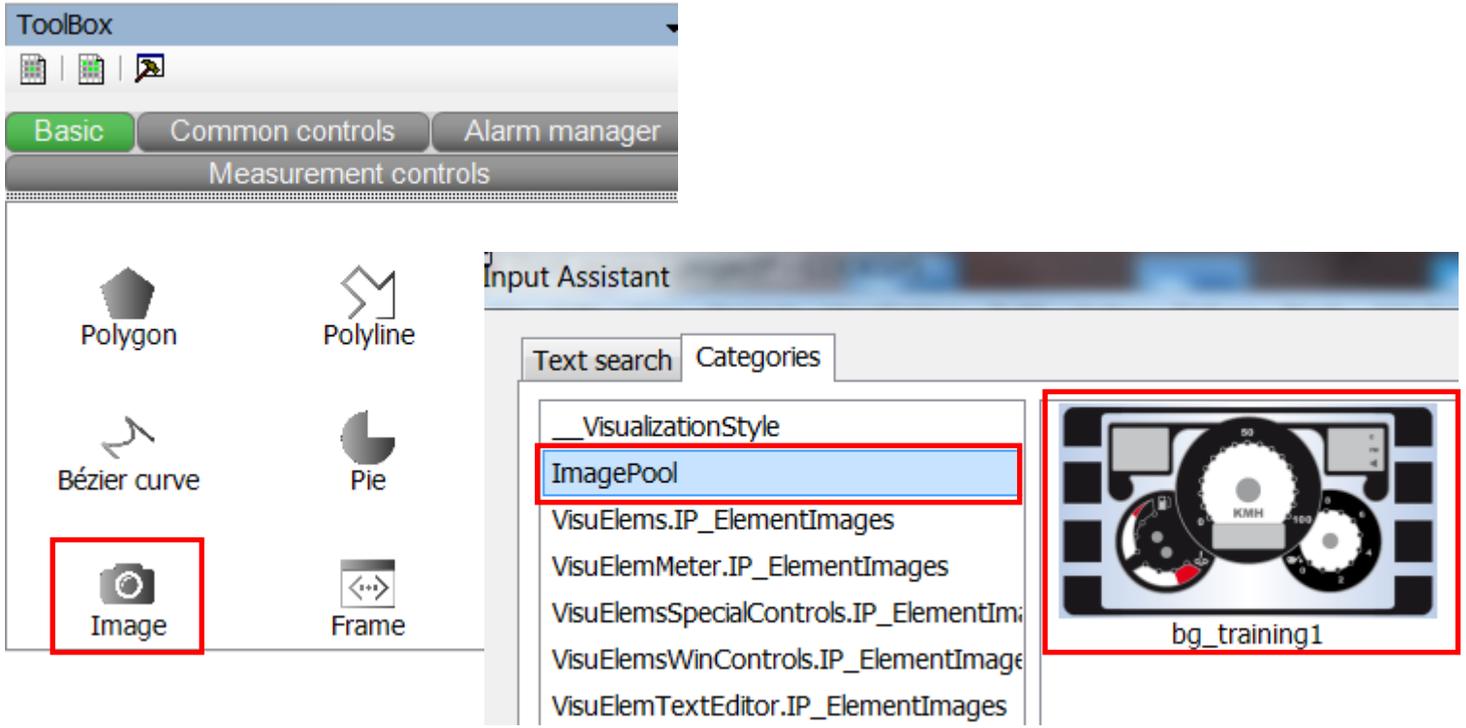
# 4.7 The first project – Images – Static images




Write the ID *bg_training1*

Select the image *Training_Background_1.png* from the training pictures folder

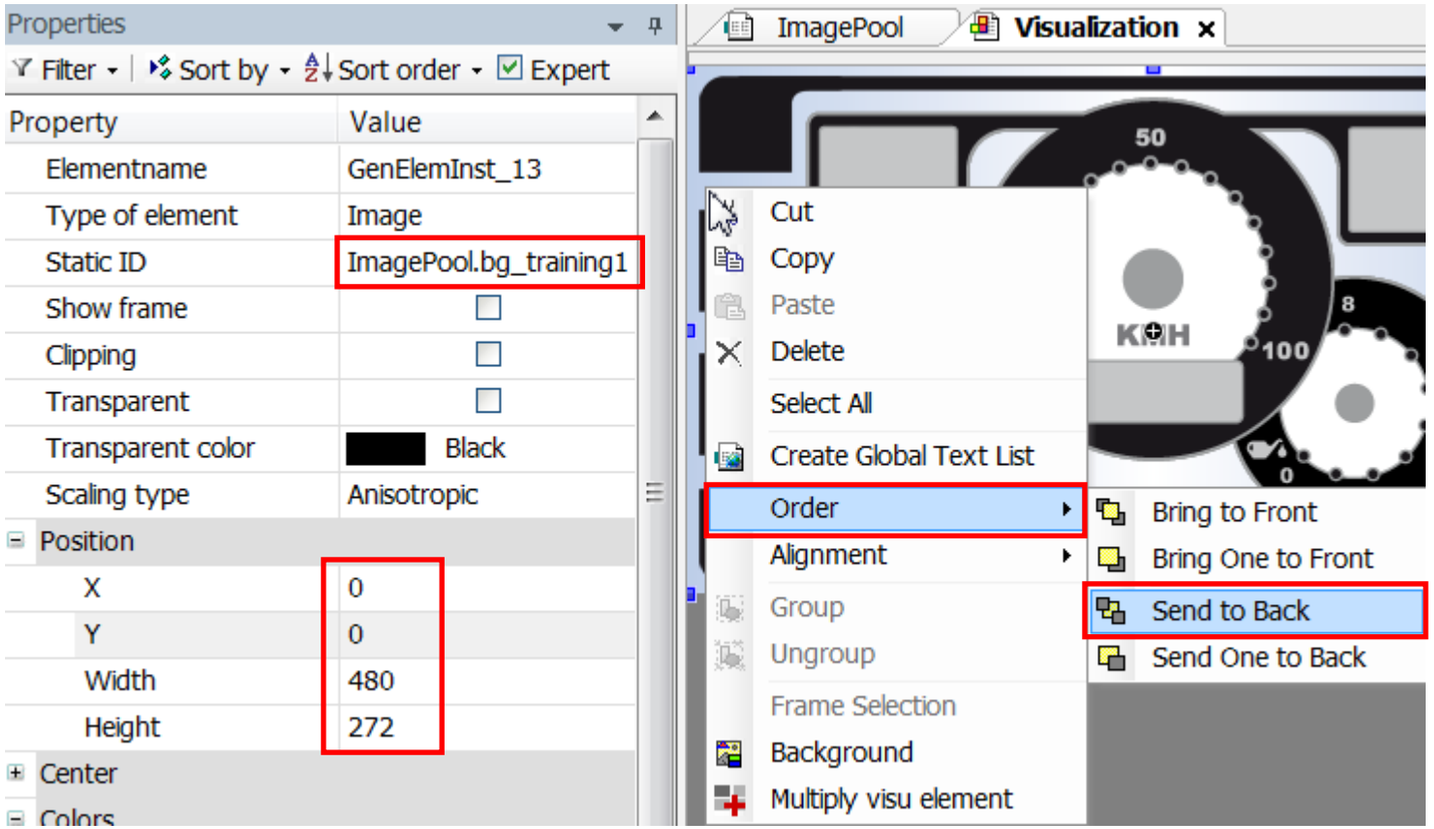
# 4.7 The first project – Images – Static images




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 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

ImagePool × Visualization

| 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

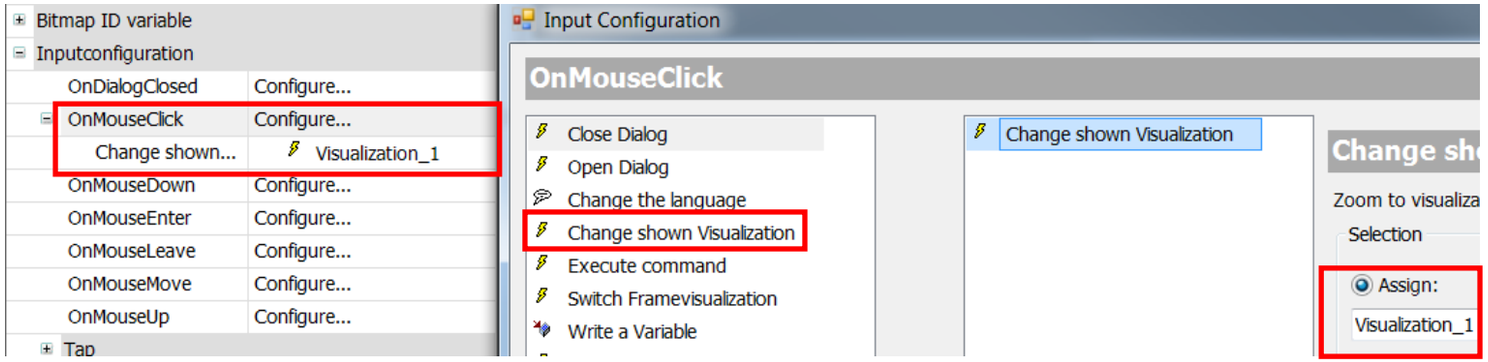
Create another image object
Choose the “plus” image in the dialog

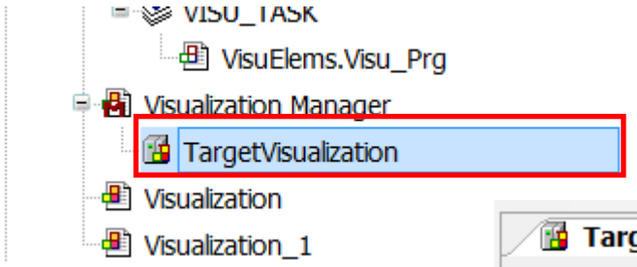
Initialize a string variable in *PLC_PRG*

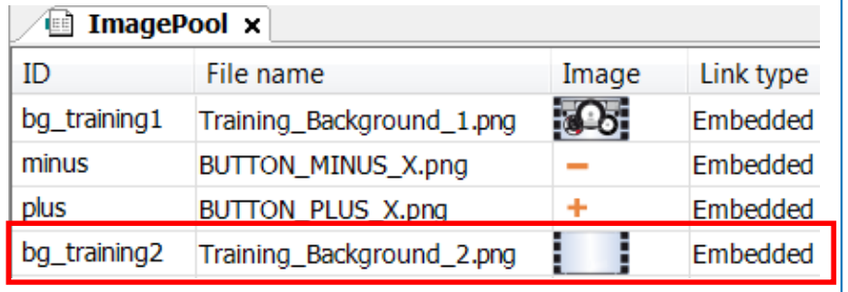
Set the Bitmap ID Variable to this string variable

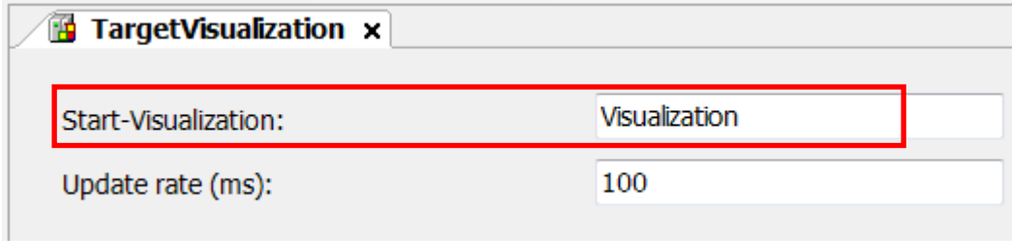
Transfer the project and set the variable on the PC (from ‘plus’ to ‘minus’)

# 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

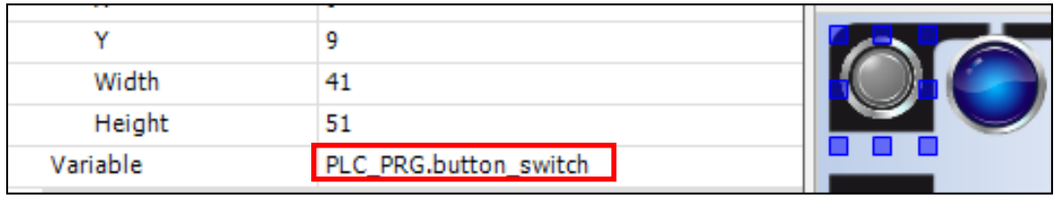
Configure the button to change to this visualization

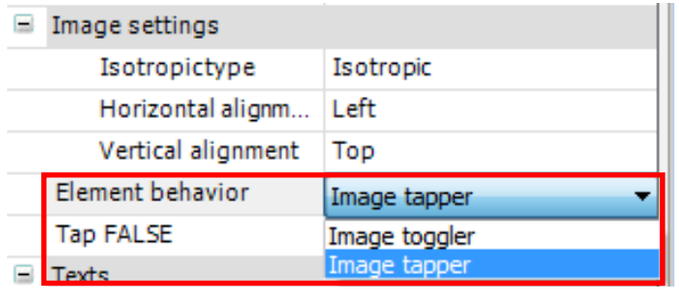
Add the image Training_Background_2.png to the *Global Image Pool*

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

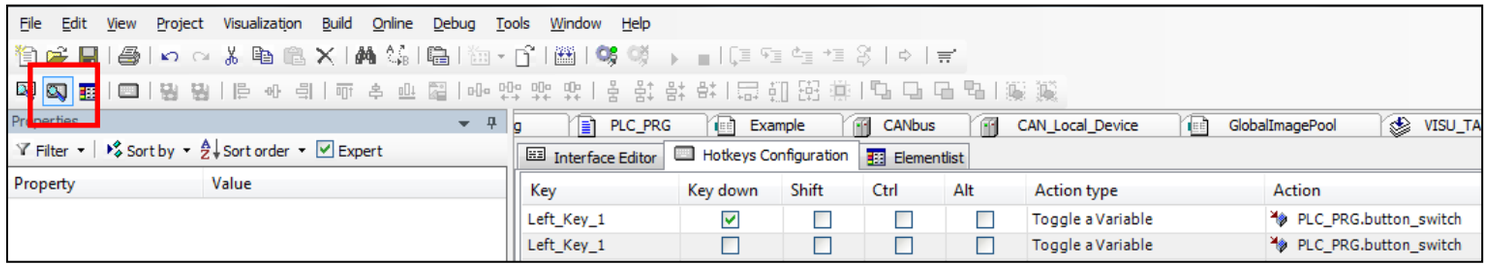
Start visualization? Set it in *TargetVisualization*

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




```
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

# 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... |
| OnMouseClick | 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”

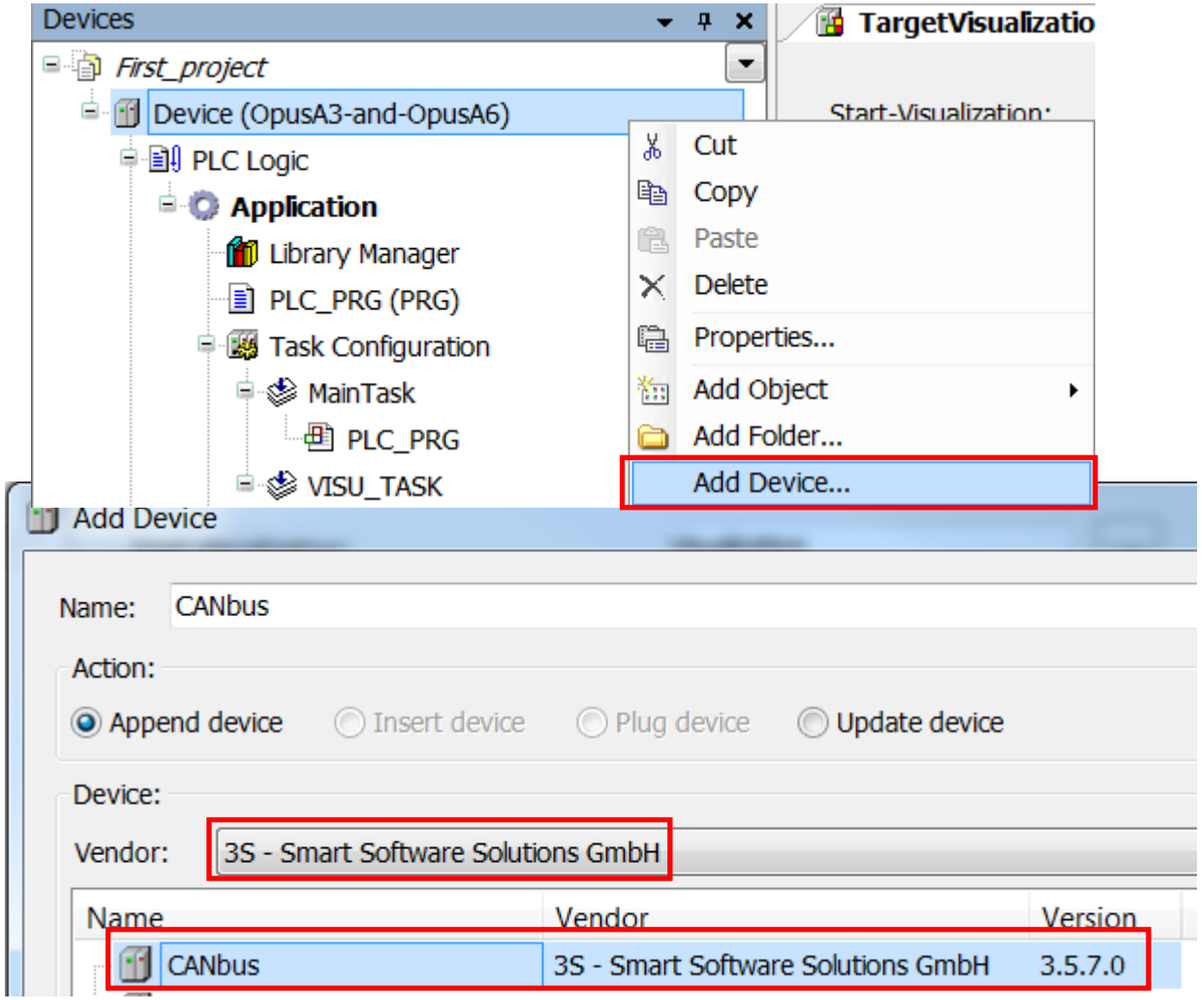
# 5. CAN communication

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

# 5.1a CAN communication

CANopen

# 5.1a CAN communication – Set up the CAN bus

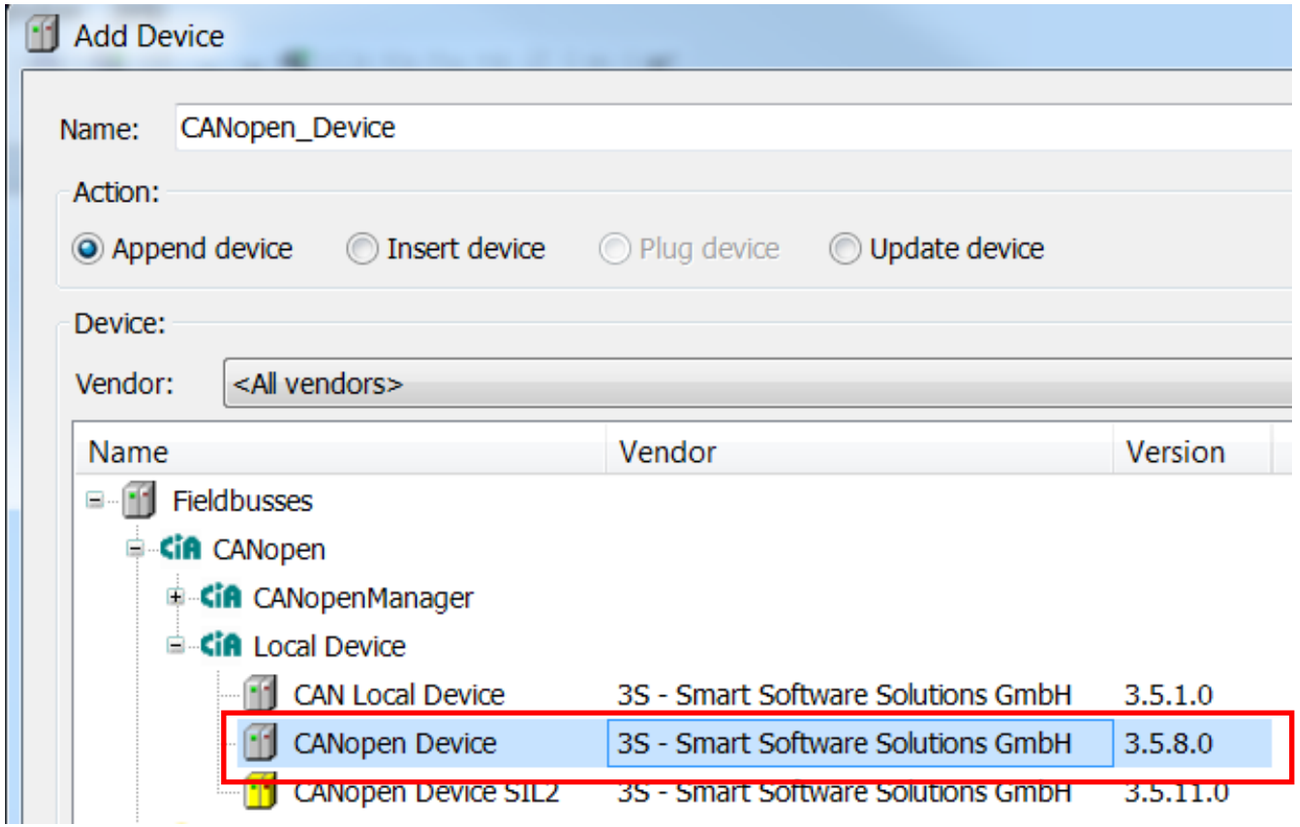



Right click on *Device*, click on *Add Device…*, select 3S company as vendor

Select CANbus, edit the name if wanted

Press on *Add Device*

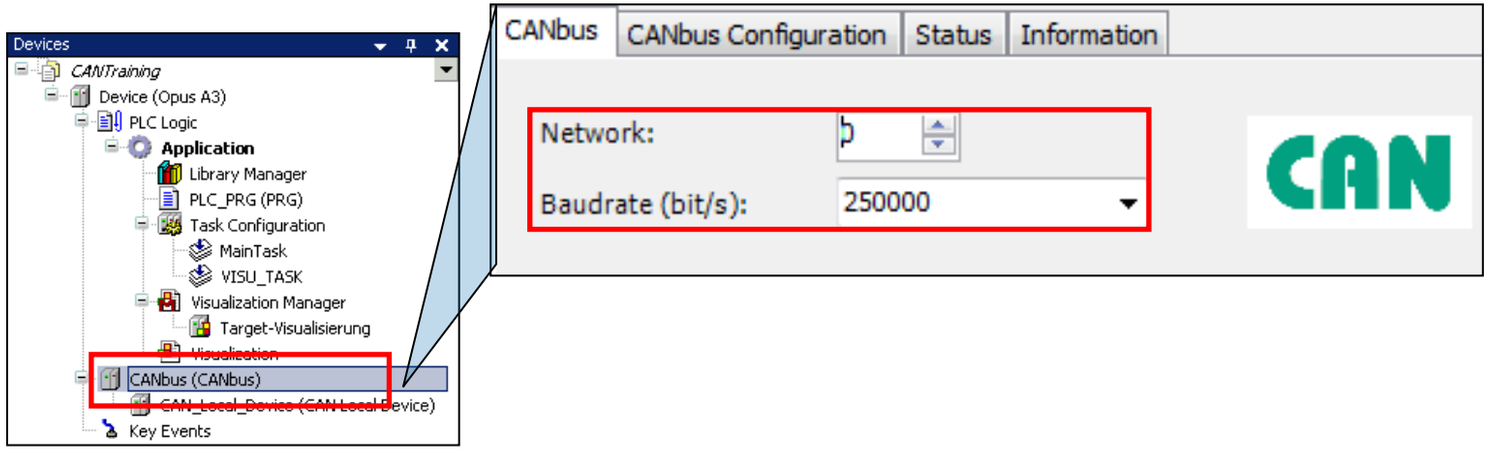
# 5.2a CAN communication – Add local device

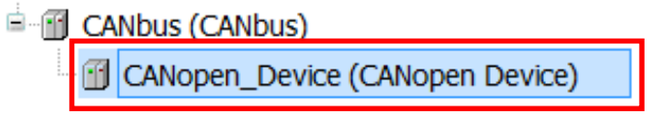


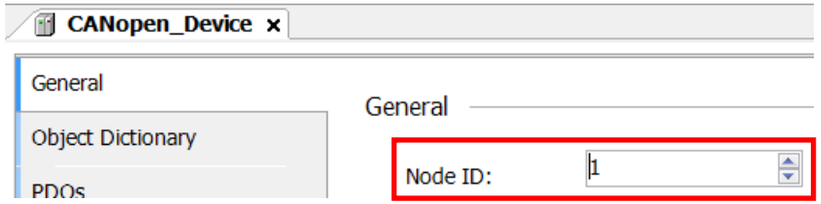

Click on *CANbus* in the device tree, the dialog will change

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

# 5.2a CAN communication – Configuration



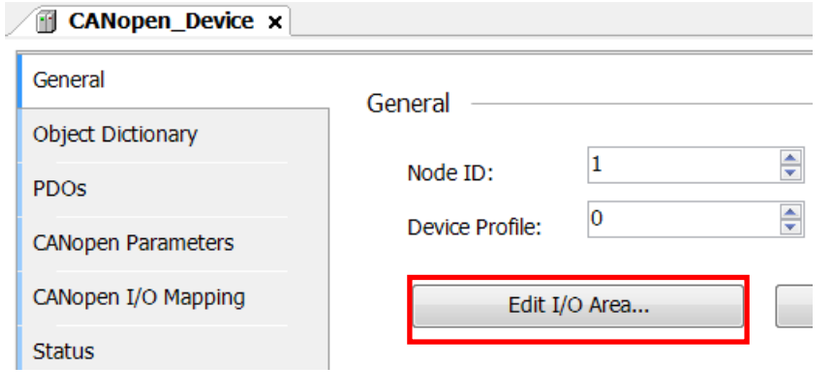



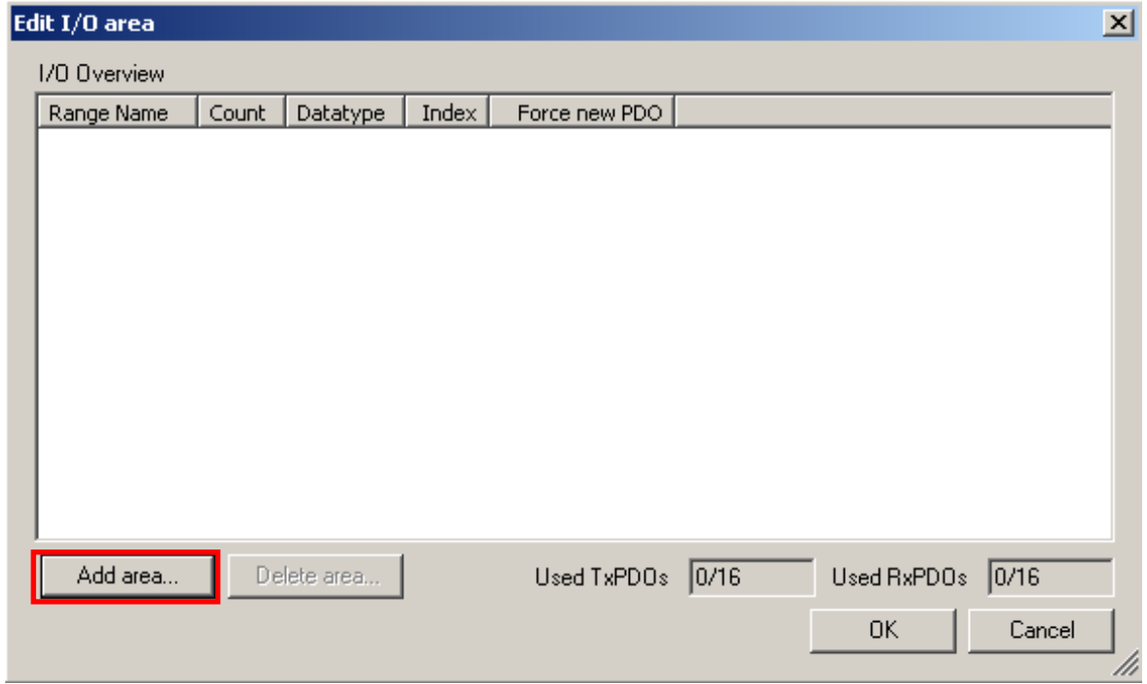

Double click on *CANbus*

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



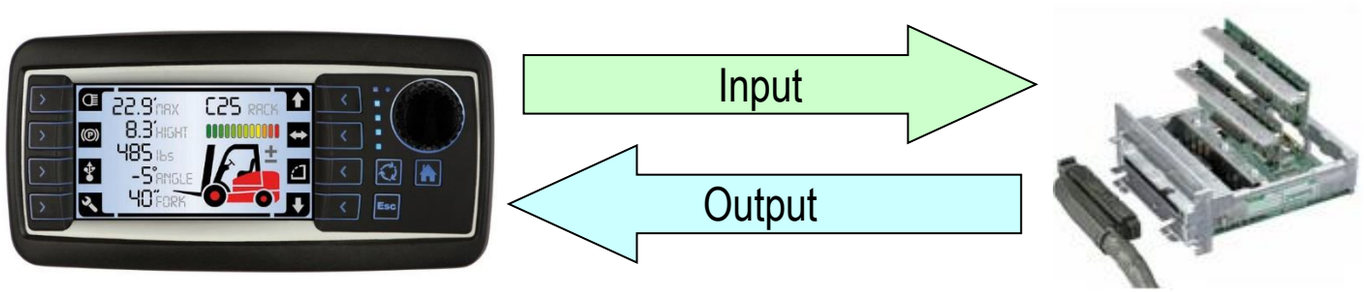




Click on *Edit I/O area…* to add the PDO communication variables

Click on *Add area…*

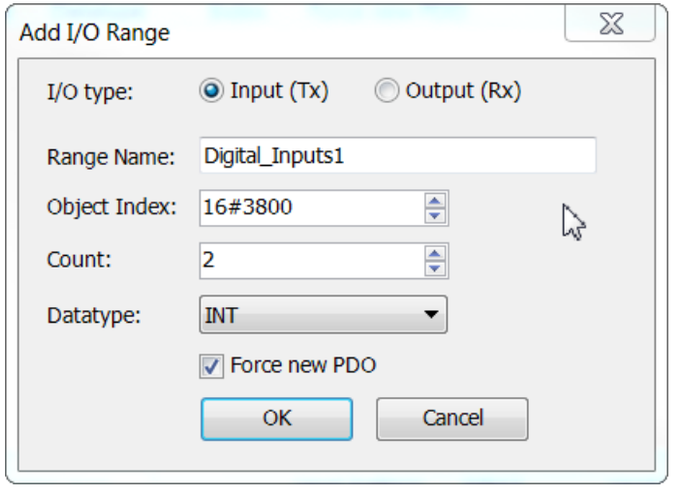
# 5.3a CAN communication – Add I/O areas

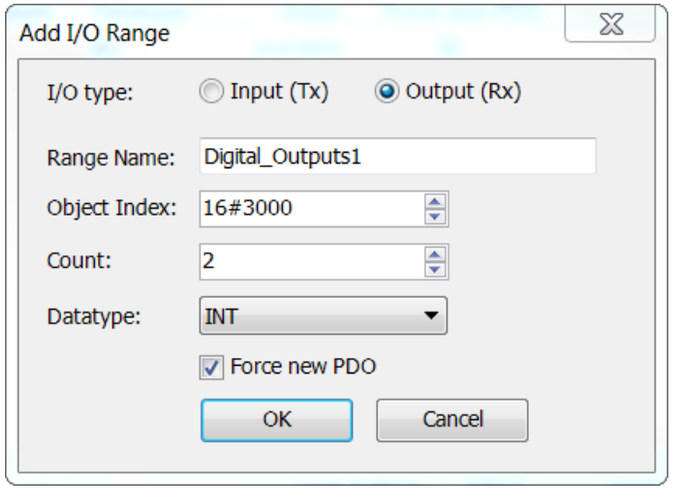
CODESYS always uses the viewpoint of the Master!



# 5.3a CAN communication – Add I/O areas

CANopen



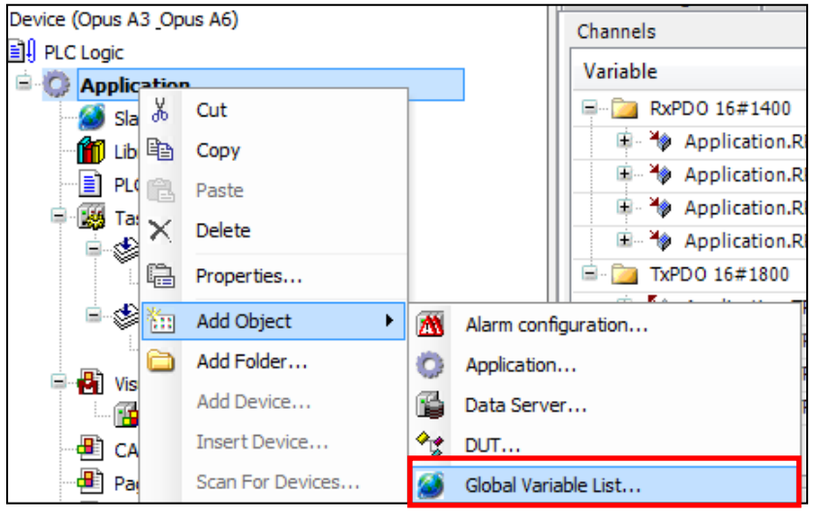




Add 2 INT Input variables for Transmit PDO (the OPUS sends these)

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




```
VAR_GLOBAL
    RPDO1_byte1_4 : UINT;
    RPDO1_byte5_8 : UINT;

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

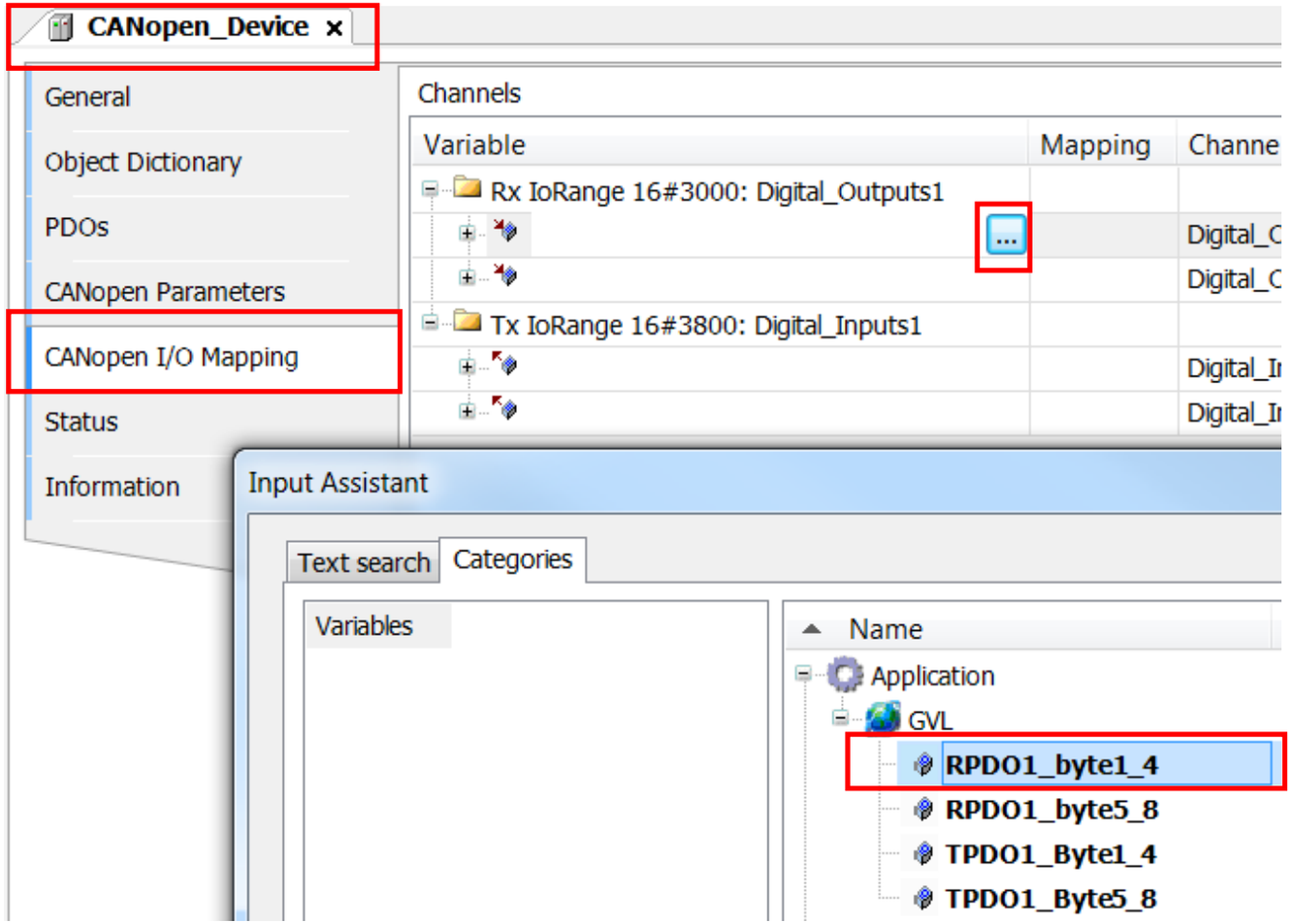
```
1  GVL.TPDO1_Byte1_4 := GVL.TPDO1_Byte1_4 + 10;
2  GVL.TPDO1_Byte5_8 := GVL.TPDO1_Byte5_8;
```

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*

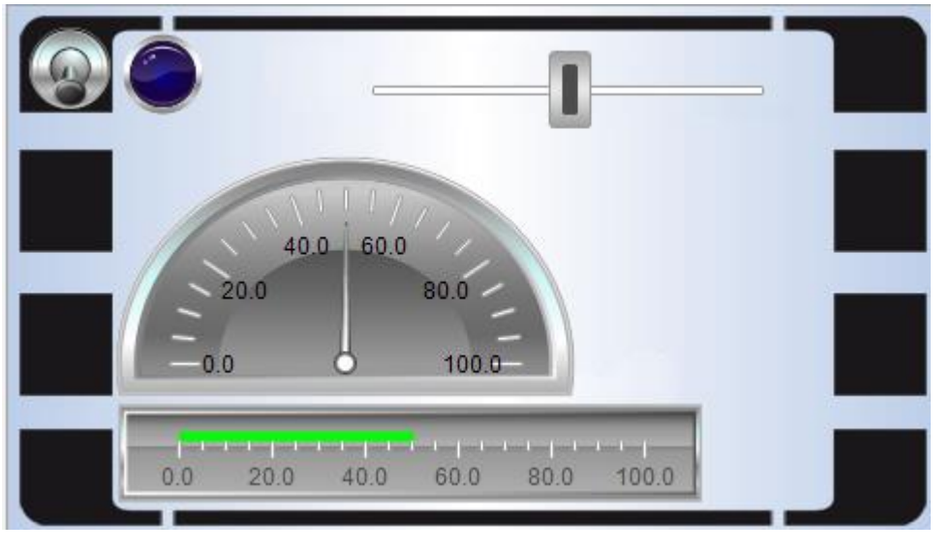
Switch to the tab *CANbus Slave I/O Mapping*

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 |

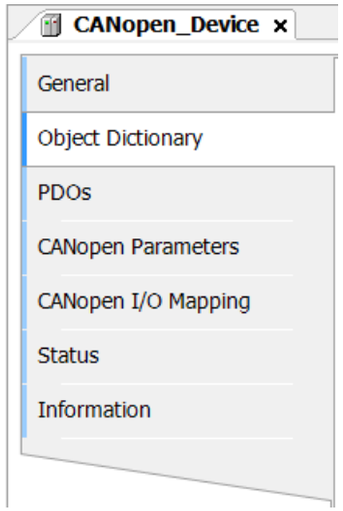
Create a meter, a bargraph and a slider object

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 |

| Value | |
|---|---|
| Default Value | $NODEID+16#180 |
| High Limit | 16#FFFFFFFF |
| Low Limit | 16#00000080 |

| Value | |
|---|---|
| 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

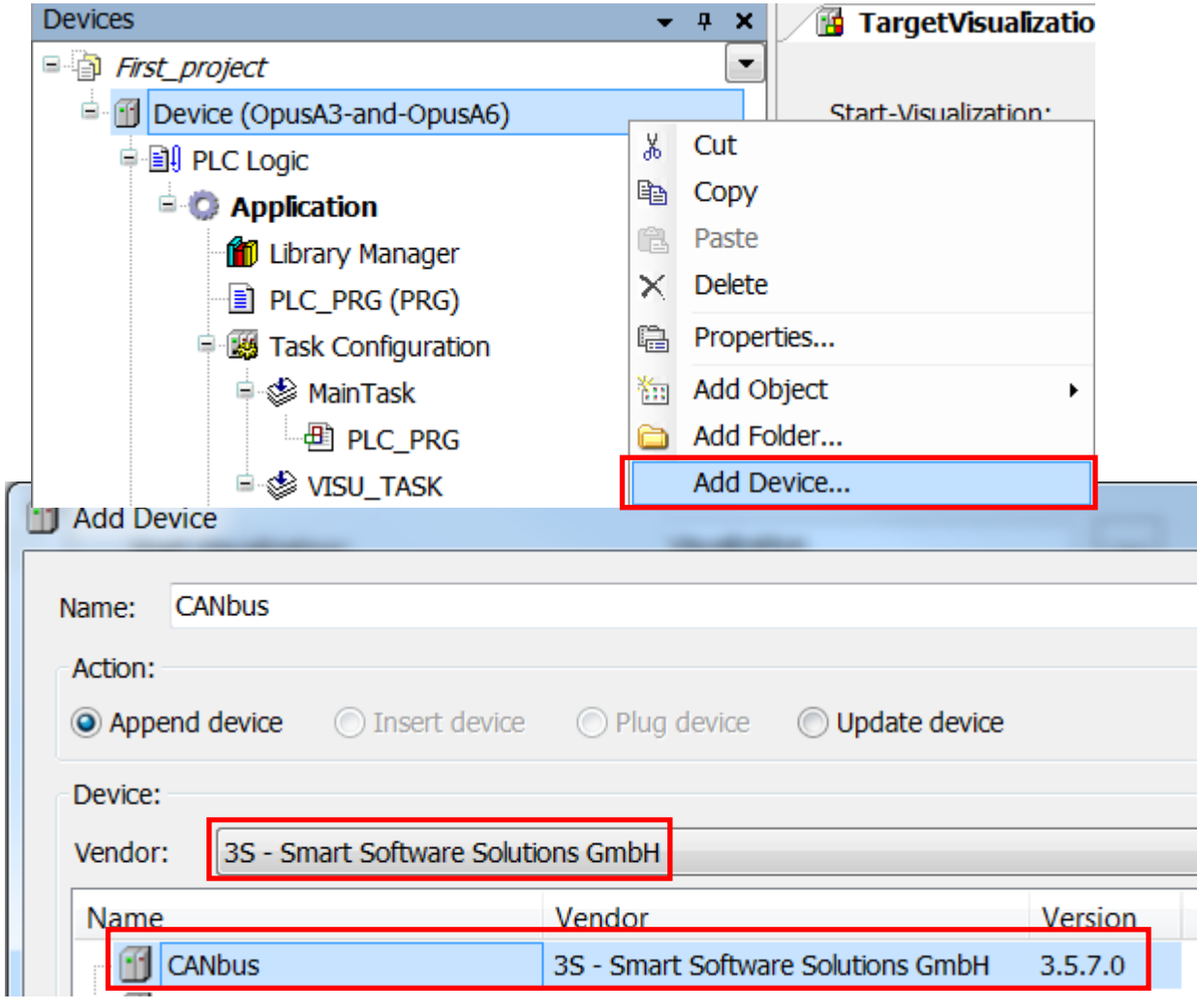
Also you can change the COB ID (not CANopen conform!) by changing the value in *COBID*

Continue with chapter 6

# 5.1b CAN communication

SAE INTERNATIONAL™ J1939

# 5.1b CAN communication – Add CAN interface

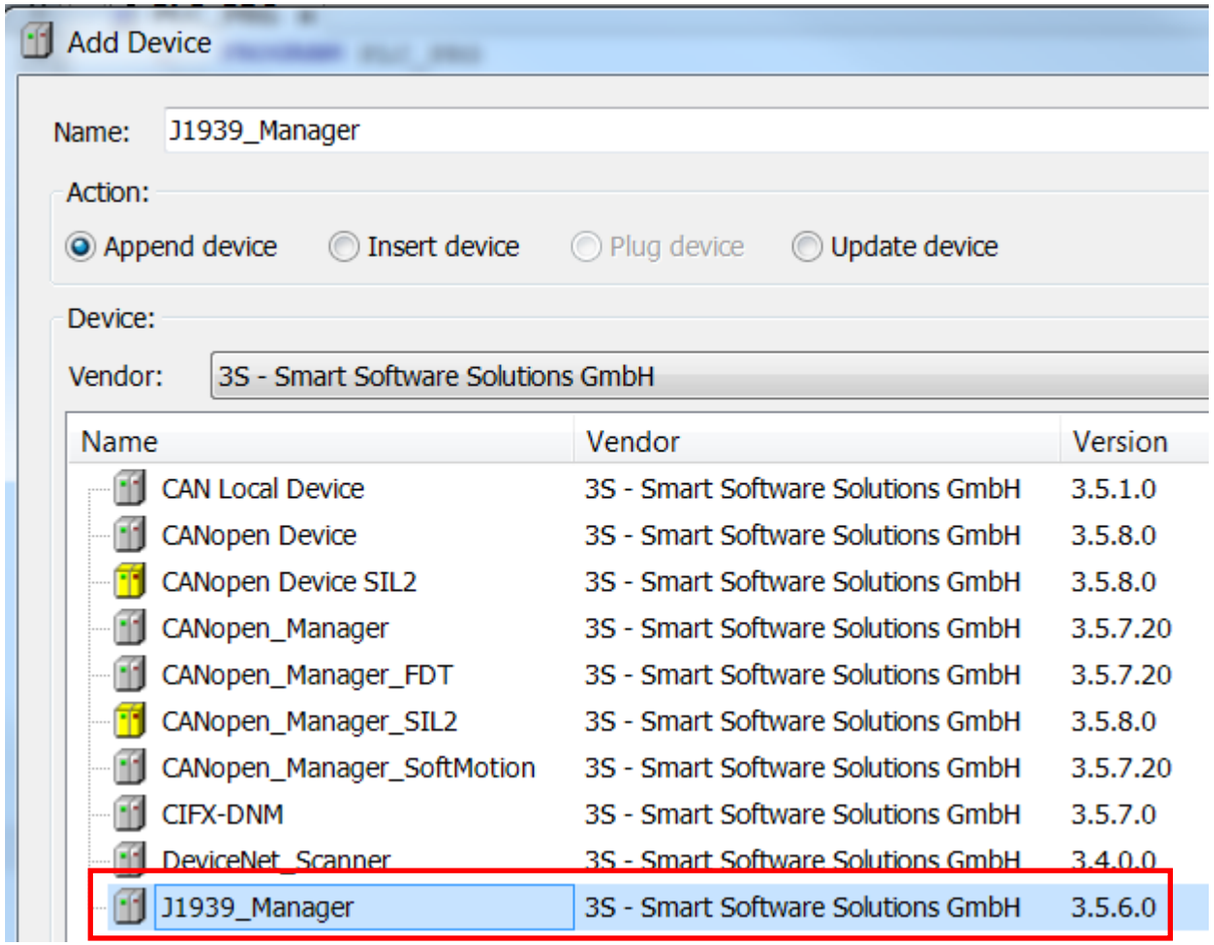



Right click on *Device*, click on *Add Device…*, select 3S company as vendor

Select CANbus in selection, edit the name if wanted

Press on *Add Device*

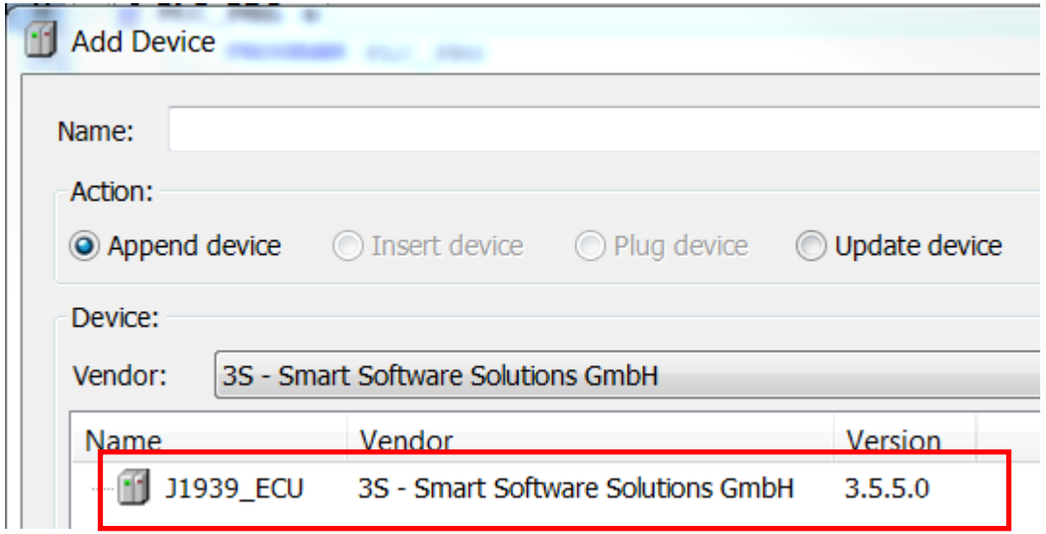
# 5.2b CAN communication – Add J1939 manager




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

Choose *J1939_Manager* and click *Add Device*

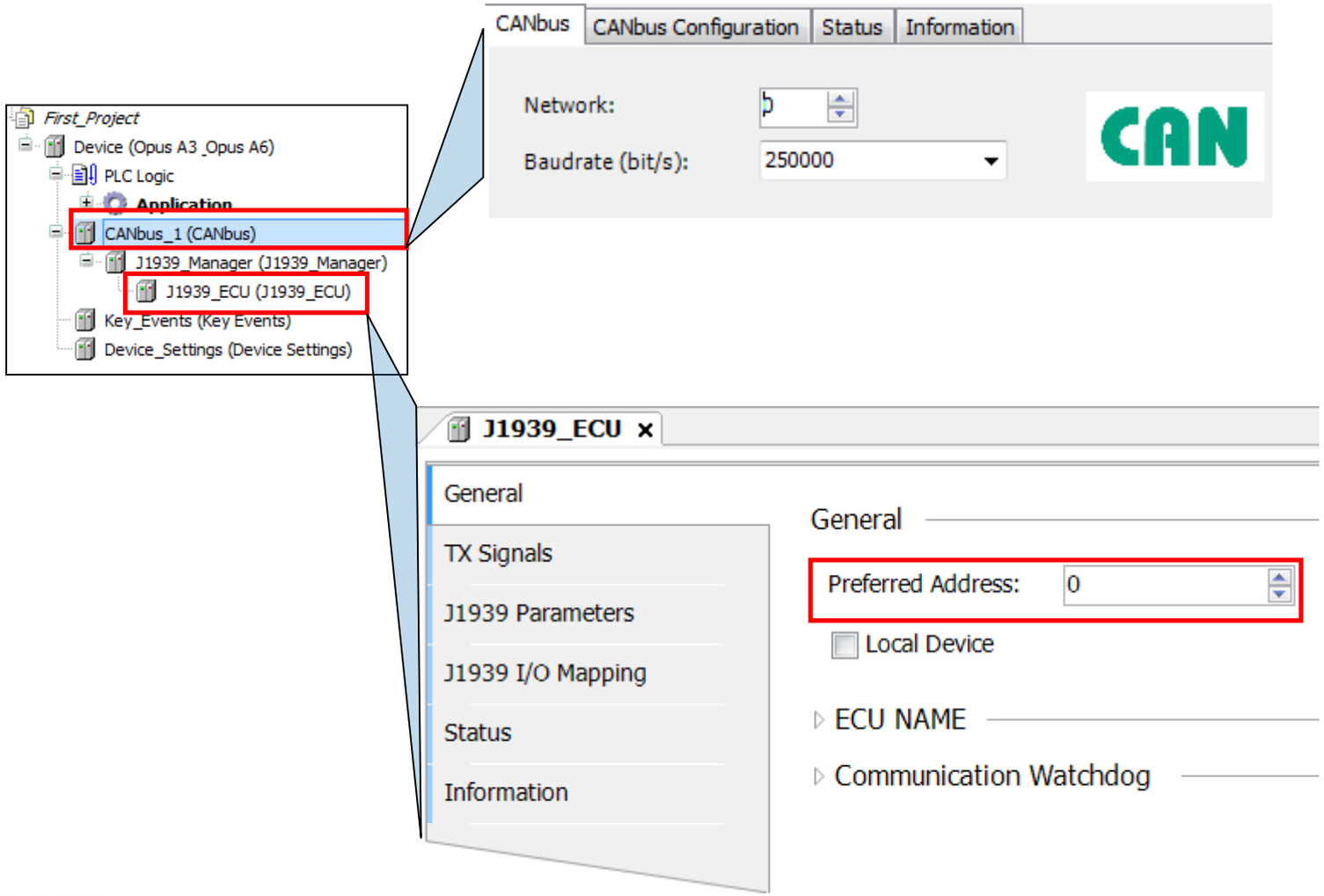
# 5.2b CAN communication – Add J1939 ECU




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



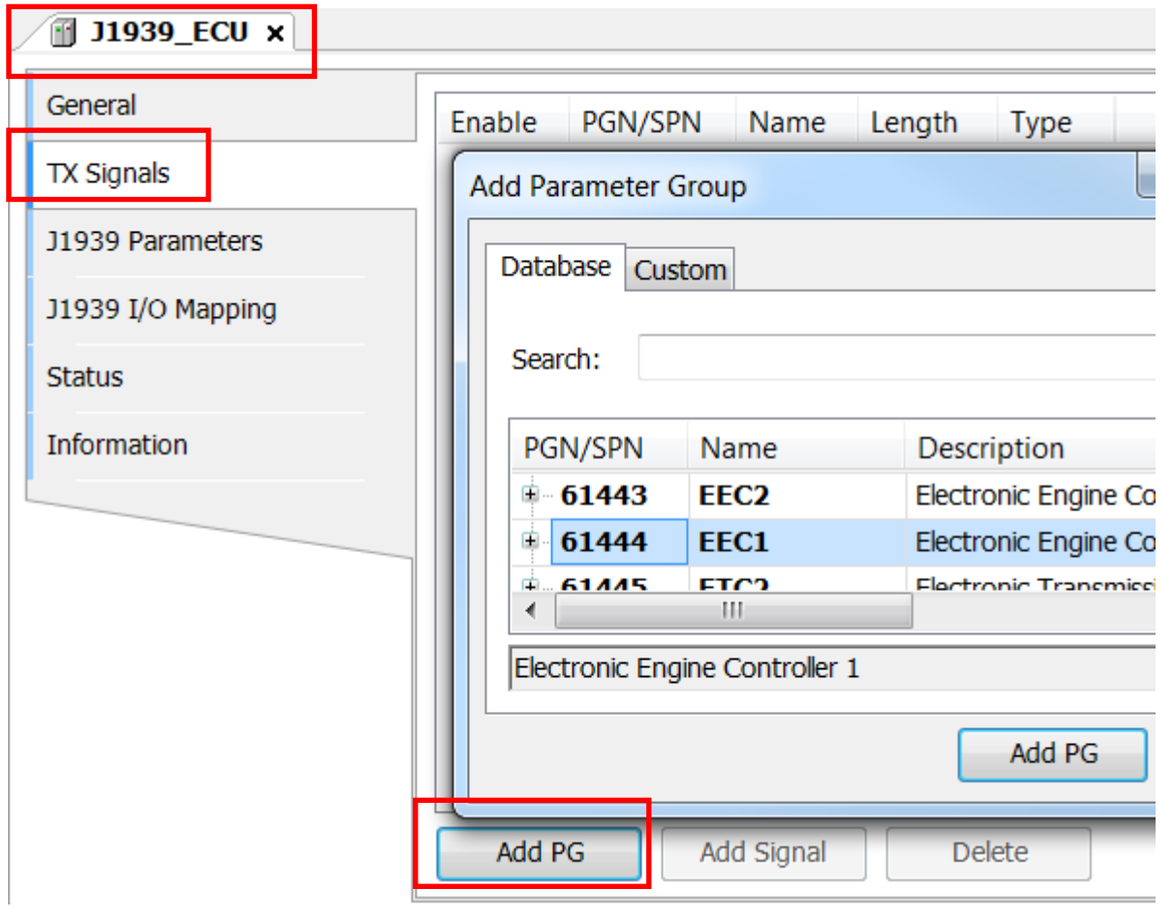

Doubleclick on *CANbus*

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

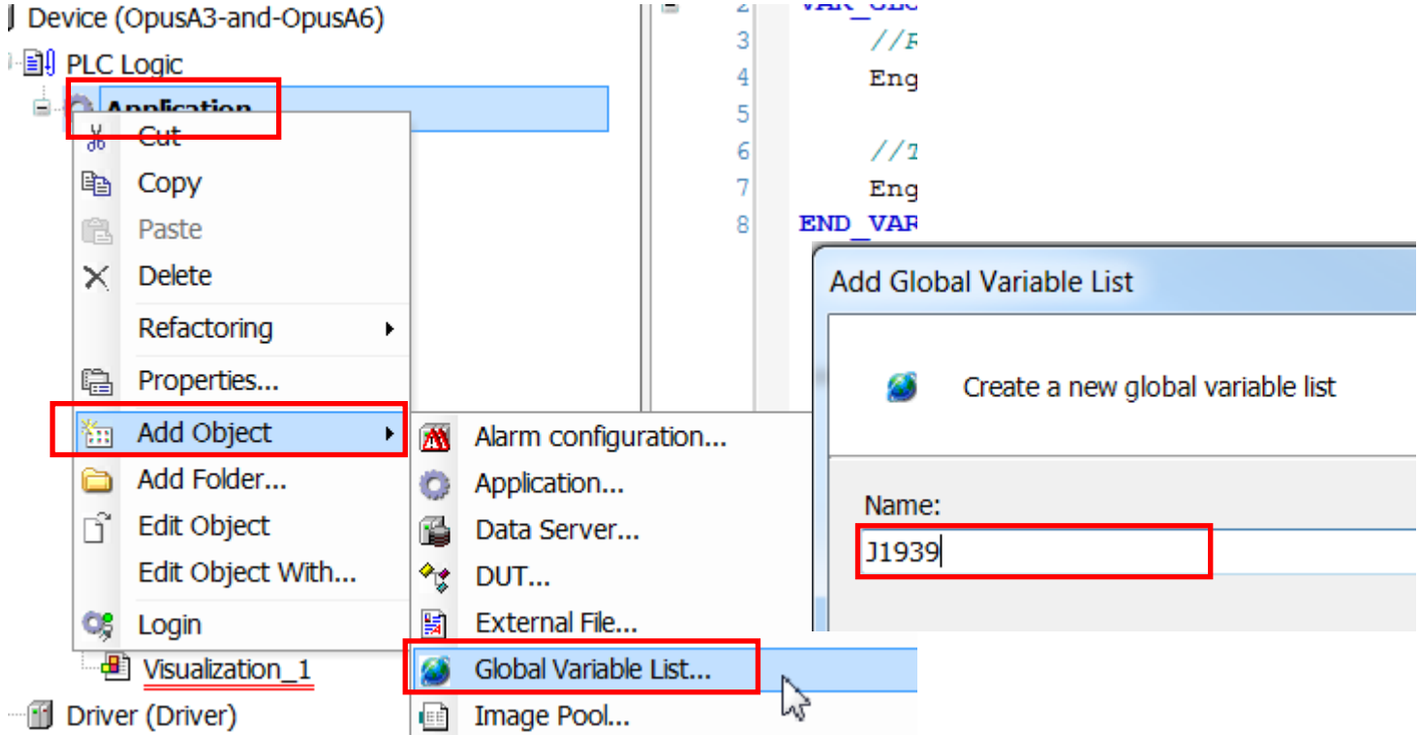



Doubleclick on *J1939_ECU*

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
```

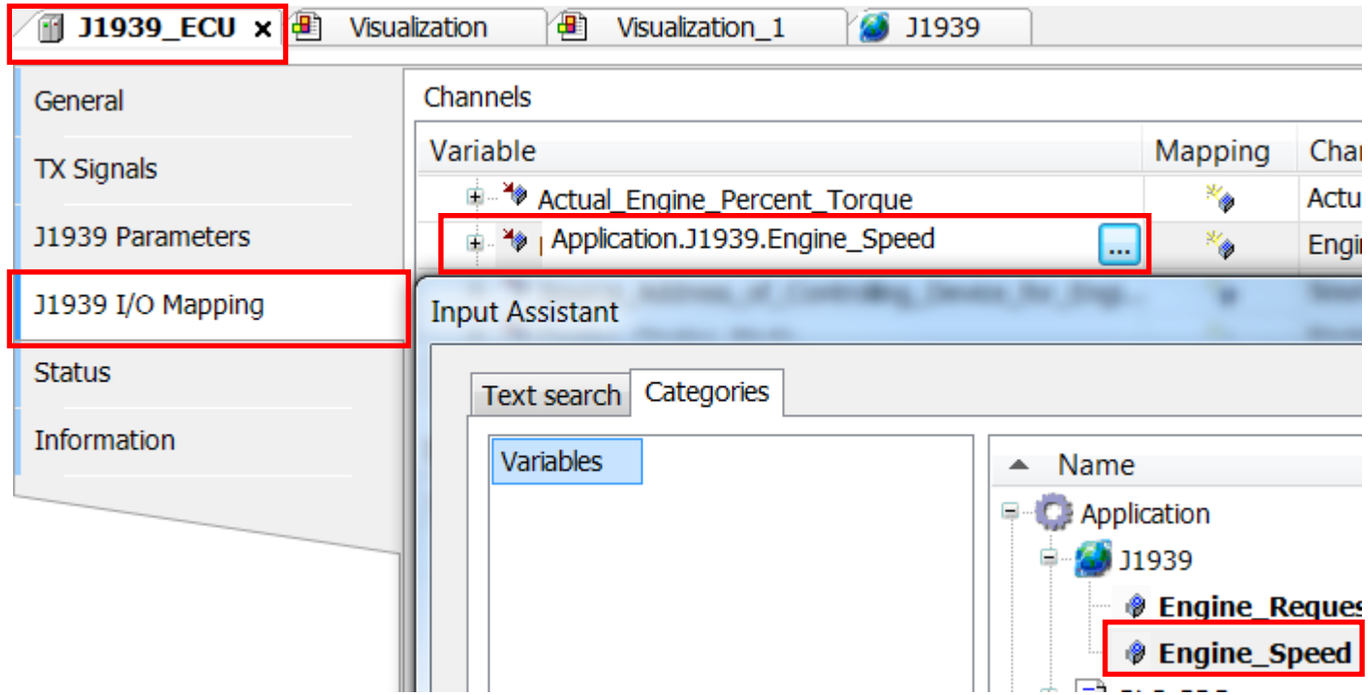
Create a Global Variable List by right-clicking *Application* -> *Add Object* -> *Global Variable List…*

Name it J1939

Click *Add* to close the dialog

Open the GVL J1939 and create these variables

# 5.3b CAN communication – Creating messages

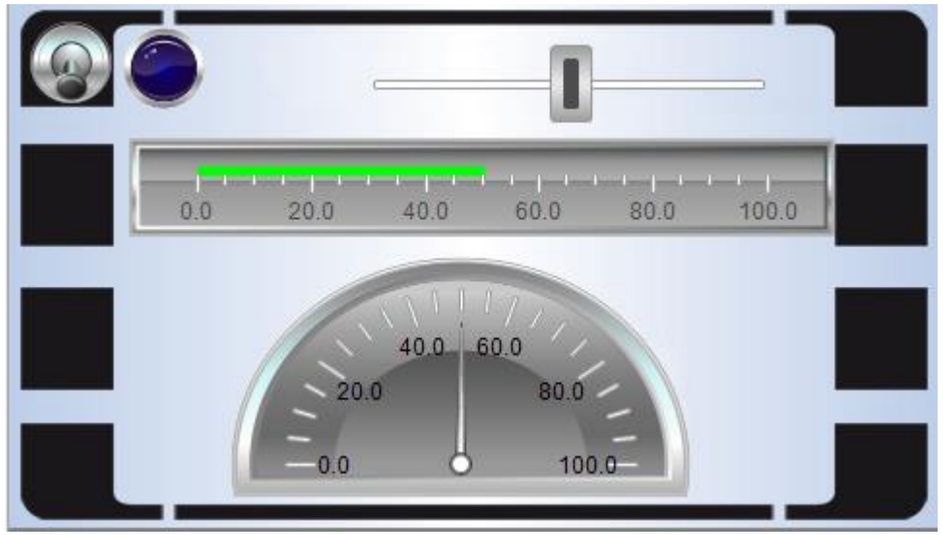



```
J1939.Engine_Speed := J1939.Engine_Speed;
```

In J1939_ECU Common J1939 I/O Mapping, map the variable *Application.J1939.Engine_Speed* to Engine_Speed

In the *PLC_PRG*, write this line

# 5.3b CAN communication – Creating messages




Create a meter, a bargraph and a slider object

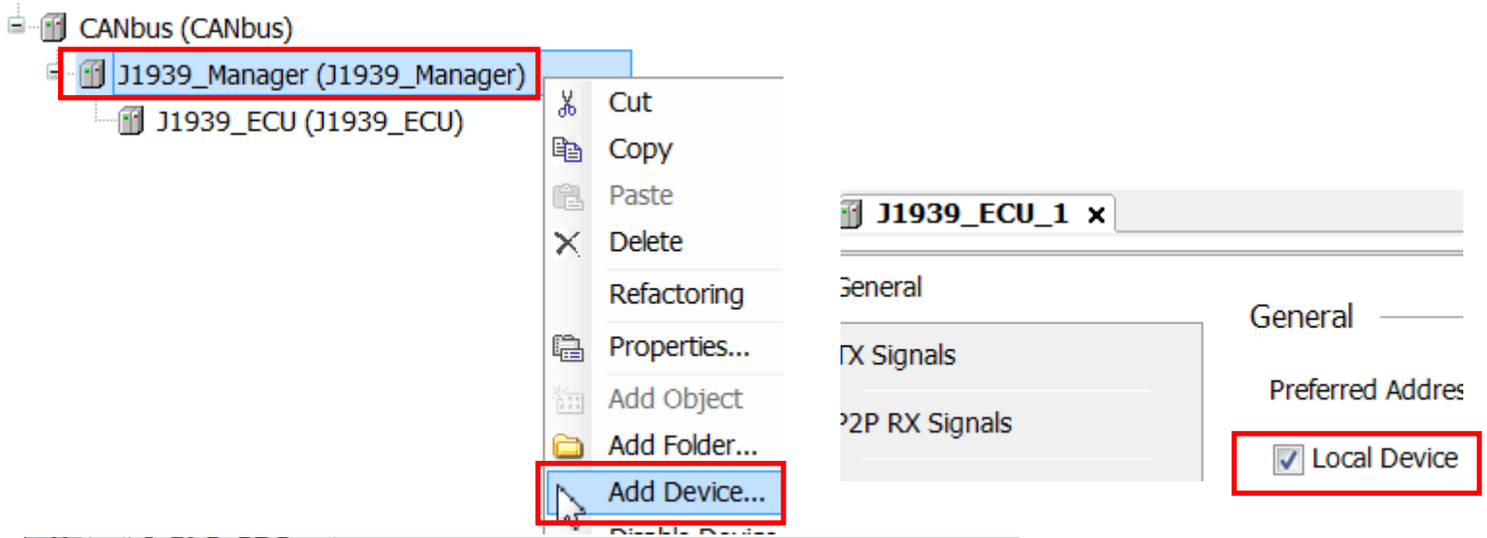
| operty | Value |
|---|---|
| Elementname | GenElemInst_2 |
| Type of element | Meter 180° |
| Value | J1939.Engine_Speed |

| | |
|---|---|
| Elementname | GenElemInst_3 |
| Type of element | Bar display |
| 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 | ☑ |

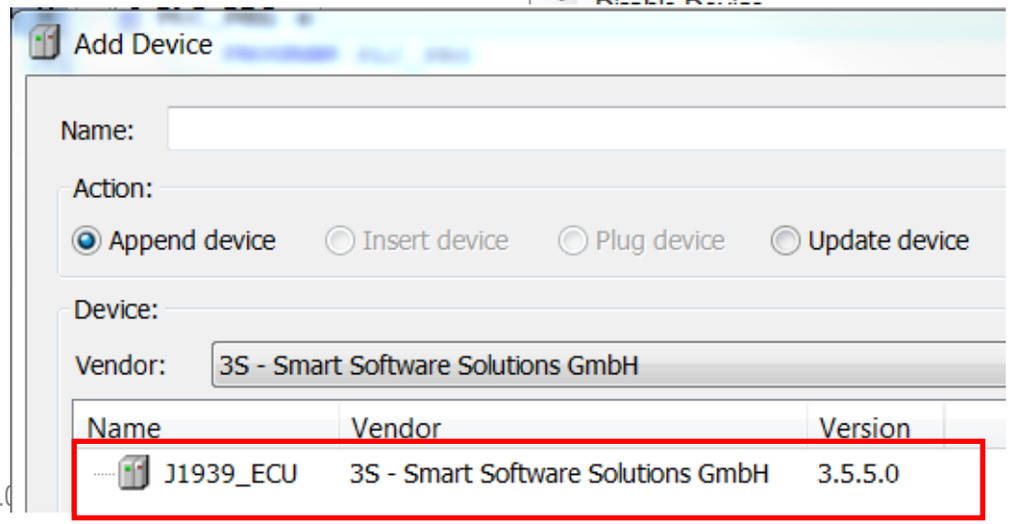
Connect the objects with the variables as shown

# 5.3b CAN communication – Sending PGNs







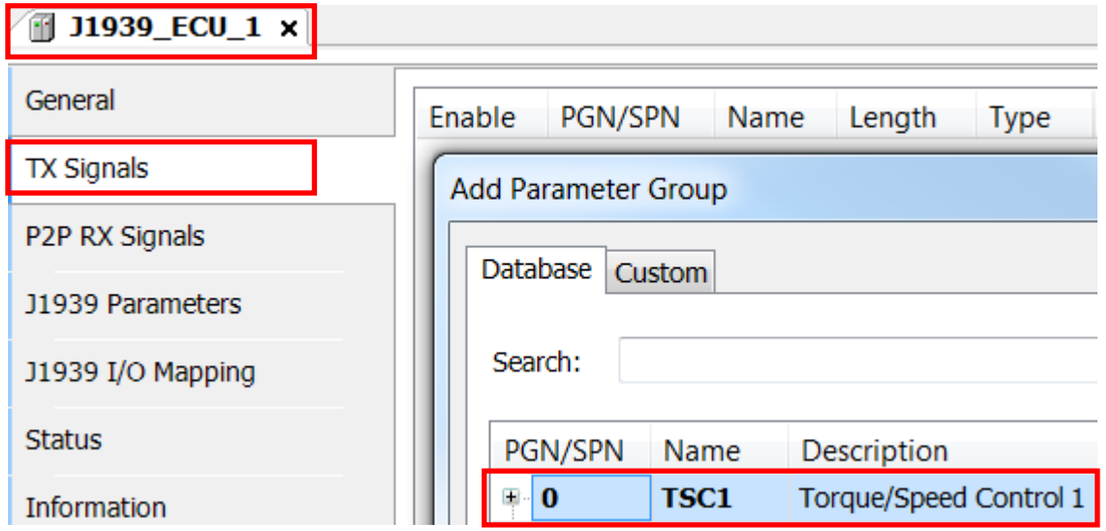



The previous J1939 configuration is only for receiving

To send, add another *J1939_ECU*, double-click it and check *Local Device*

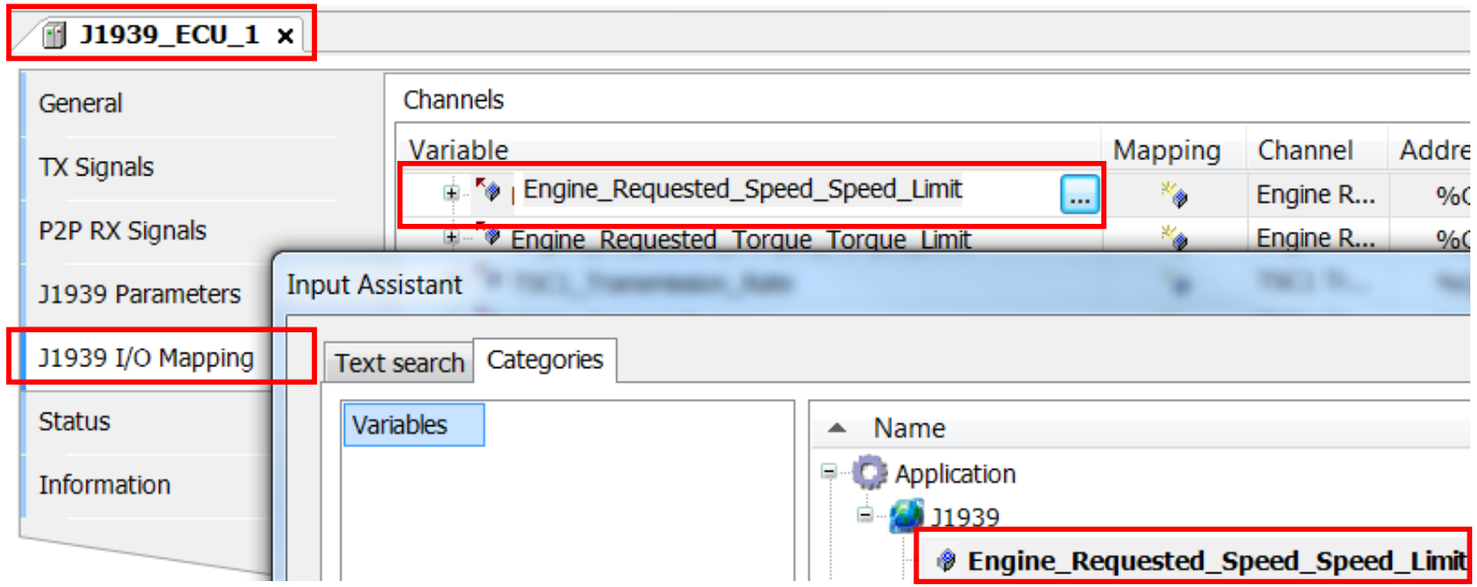
-> This device is only for sending!

# 5.3b CAN communication – Sending PGNs




Add a PG and use PG 0

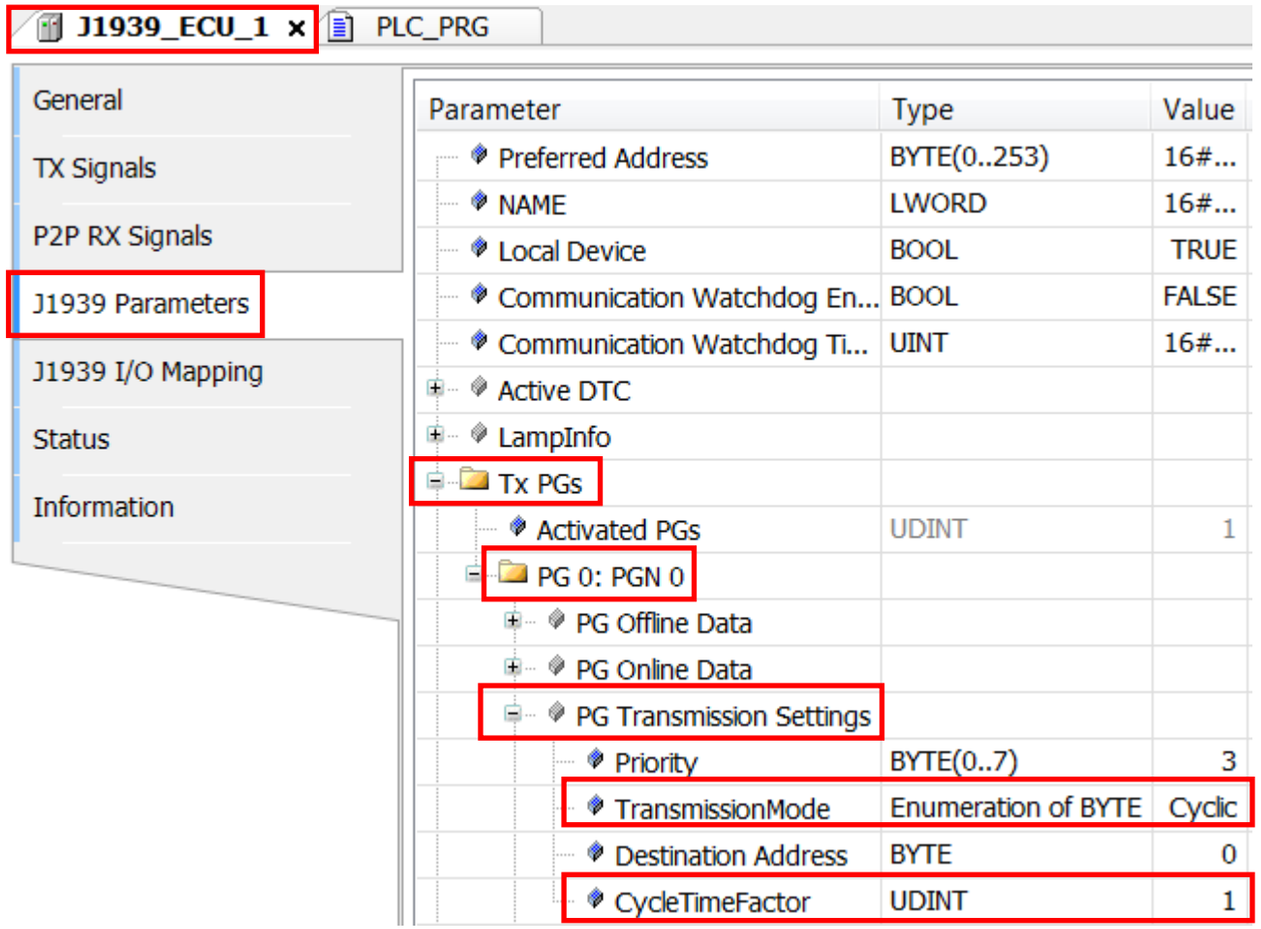
# 5.3b CAN communication – Sending PGNs




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



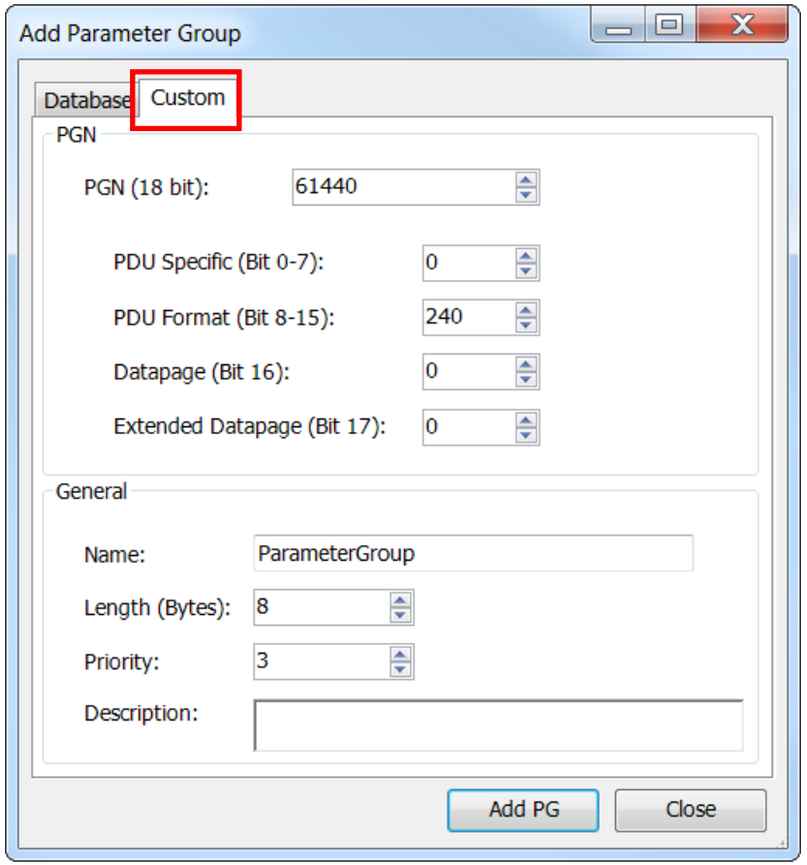

Make the settings according to the screenshot

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




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

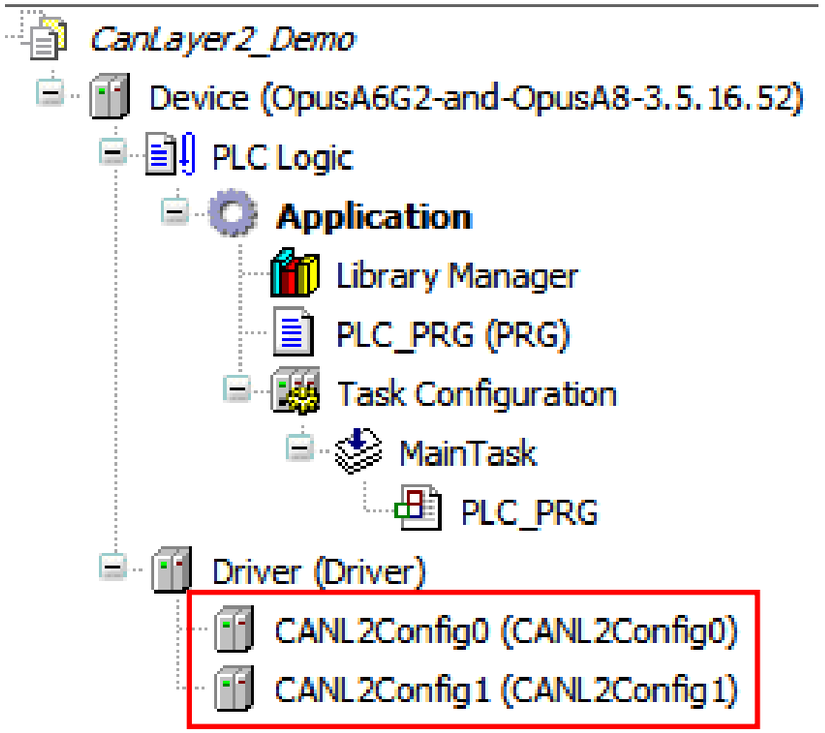
Continue with chapter 6

# 5.1c CAN communication

## CAN Layer 2

# 5.1c CAN communication – Setup

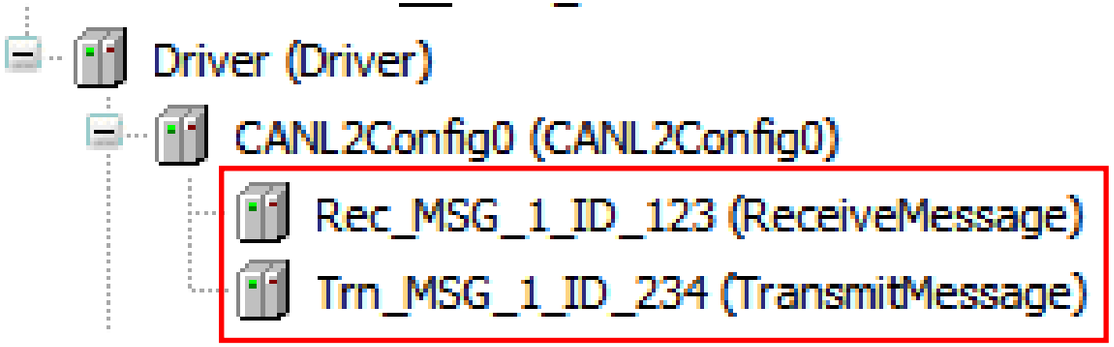
CAN Layer 2




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

CAN Layer 2

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

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 catched messages |
| Diagnosis Message: | STRING | " | " | | |
| DiagAcknowledge | BOOL | 0 | 0 | | Extended diagnostic information acknowledge |

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

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

# 5.1c CAN communication – Receive messages - The data

CAN Layer 2

The data for a receive mapping can be processed like this

Declaration

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

Code

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

# 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 catched messages |
| Diagnosis Message: | STRING | " | " | | |
| DiagAcknowledge | BOOL | 0 | 0 | | Extended diagnostic information acknowledge |

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

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

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 catched 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 – onValueChange (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

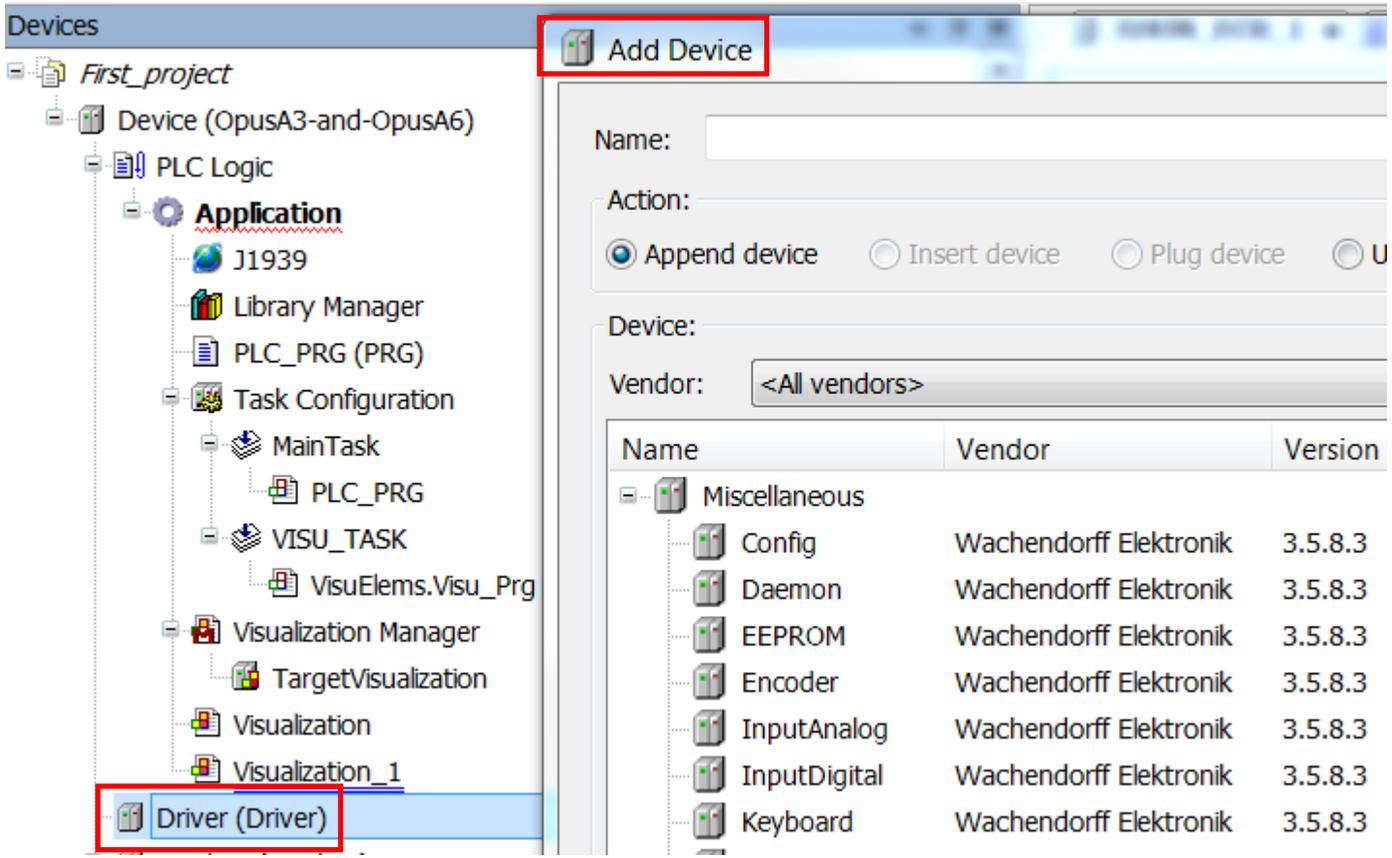
# Agenda

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

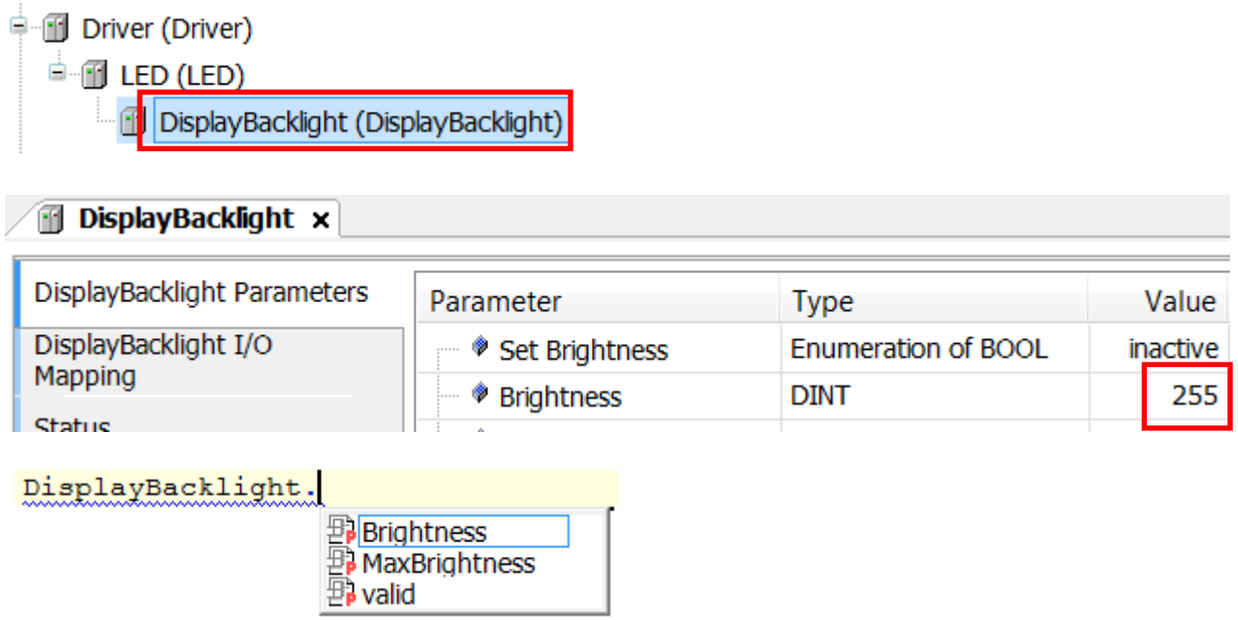
# 6.1. Extended agenda – Wachendorff ARM library




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

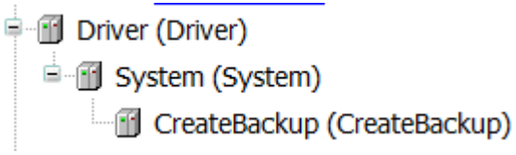


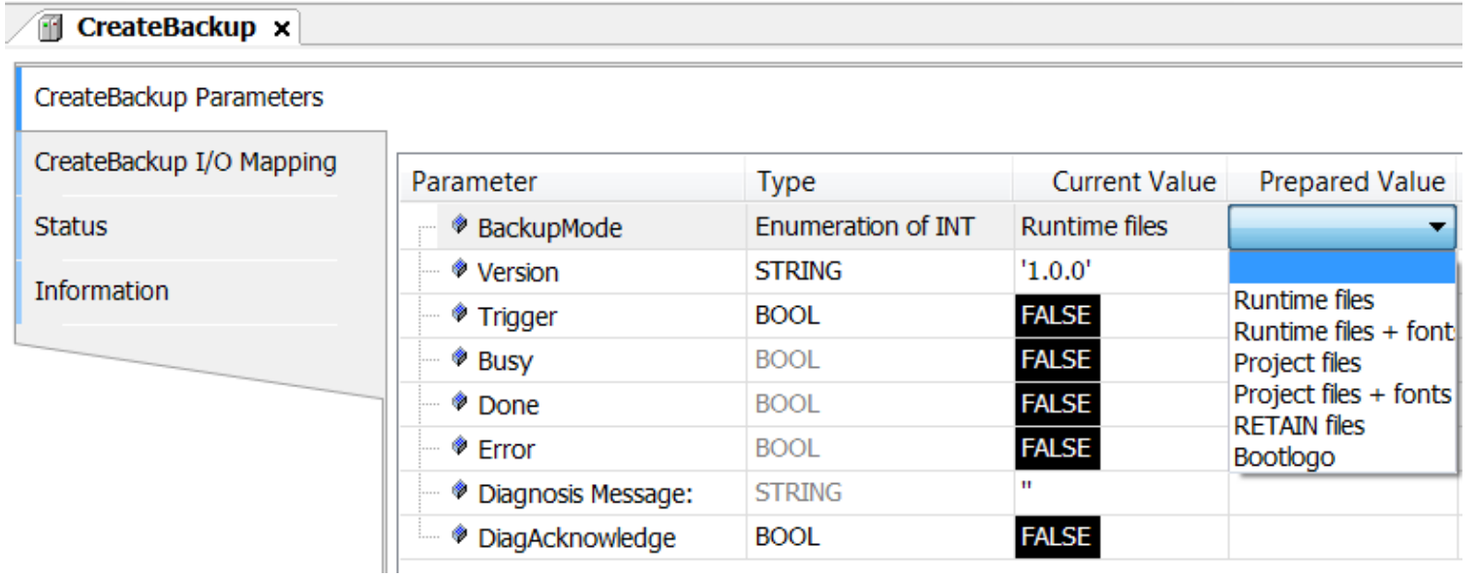

Example: Display Backlight

Usage in code:

DisplayBacklight.Brightness := 255;

# 6.2. Backup / USB project loading




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

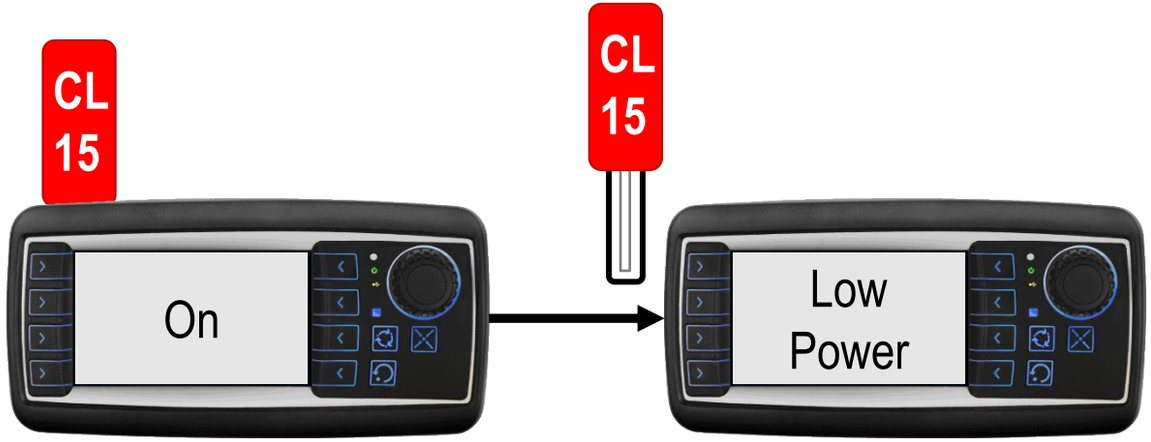
Runtime files -> user____.tar.gz
Project files -> Update at runtime with Update Daemon

# 6.4. Extended agenda – Power management

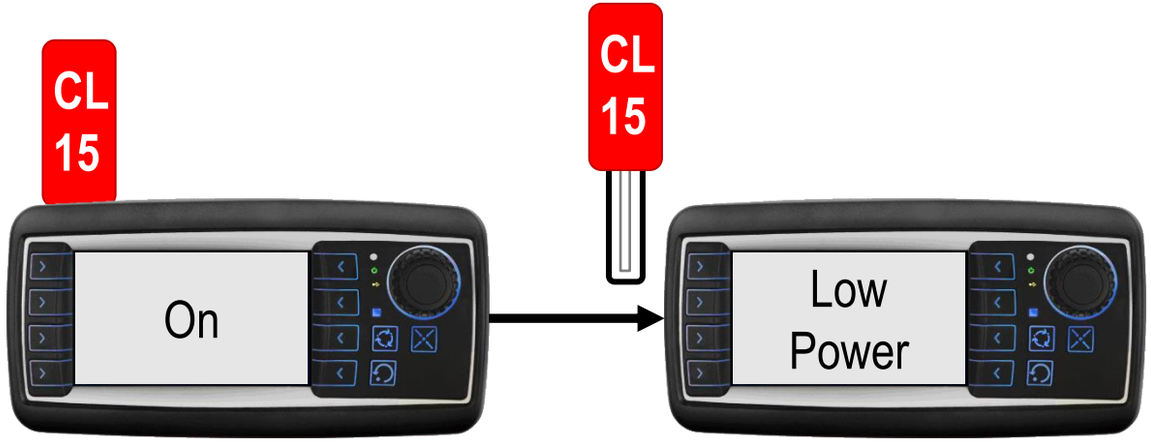

mA @13.5 V

| | A3 | A6 | A6G2 | A8 |
|---|---|---|---|---|
| On | 430 | 900 | 1000 | 1600 |

# 6.4. Extended agenda – Power management



# 6.4. Extended agenda – Power management



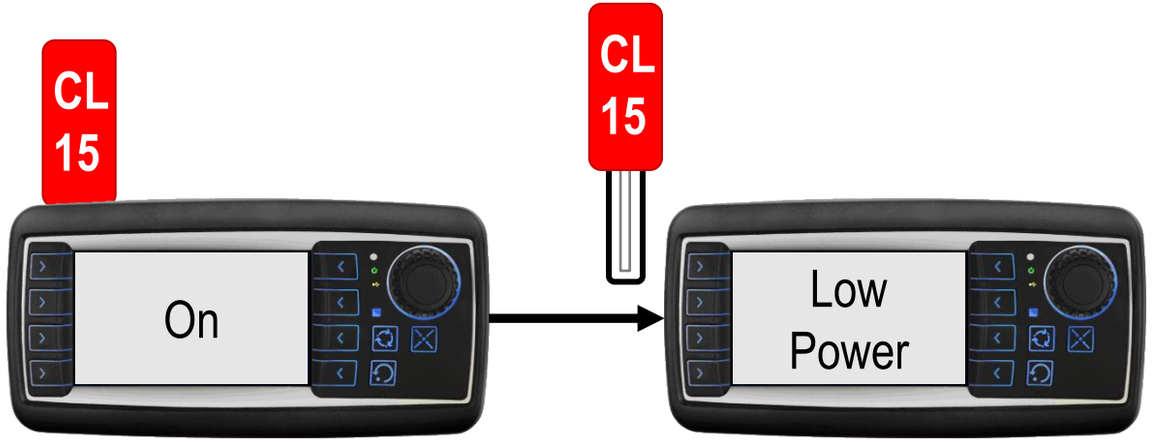

Codesys, CAN are working

☐/☑ Display function
☐/☑ Display backlight
☐/☑ Key backlight
☐/☑ Key function
☐/☑ Encoder function
☐/☑ Outputs

☐/☑ Multicolor LED
☐/☑ Beeper / Speaker
☐/☑ Touchscreen
☐/☑ Console (RS232)
☐/☑ Video processing
☐/☑ 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 |

*With minimal configuration

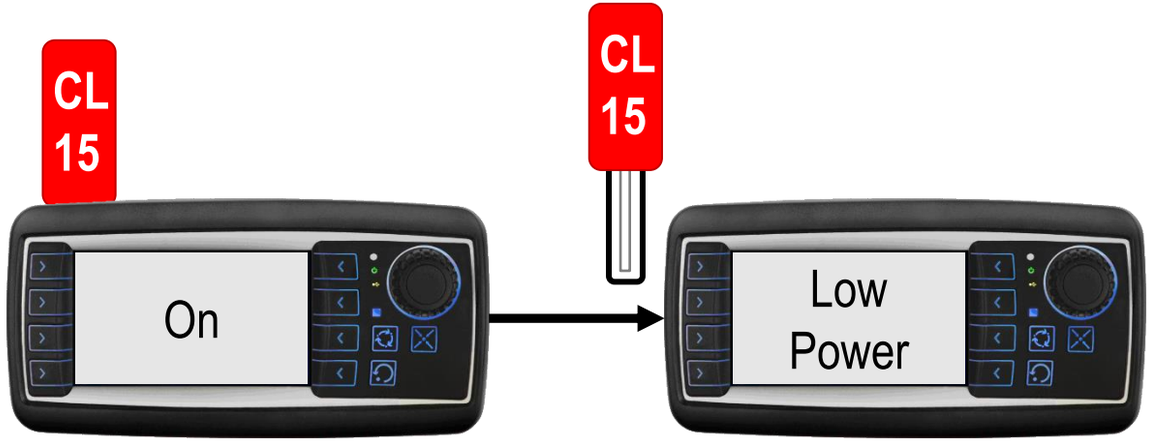
Codesys, CAN are working

☐/☑ Display function
☐/☑ Display backlight
☐/☑ Key backlight
☐/☑ Key function
☐/☑ Encoder function
☐/☑ Outputs

☐/☑ Multicolor LED
☐/☑ Beeper / Speaker
☐/☑ Touchscreen
☐/☑ Console (RS232)
☐/☑ Video processing
☐/☑ Camera output

Driver->Daemon->PDM->_____)

# 6.4. Extended agenda – Power management




0 – 2332800 seconds
PMD.LowPowerTimeout)

# 6.4. Extended agenda – Power management

# 6.4. Extended agenda – Power management


Nothing is working, no functionality!

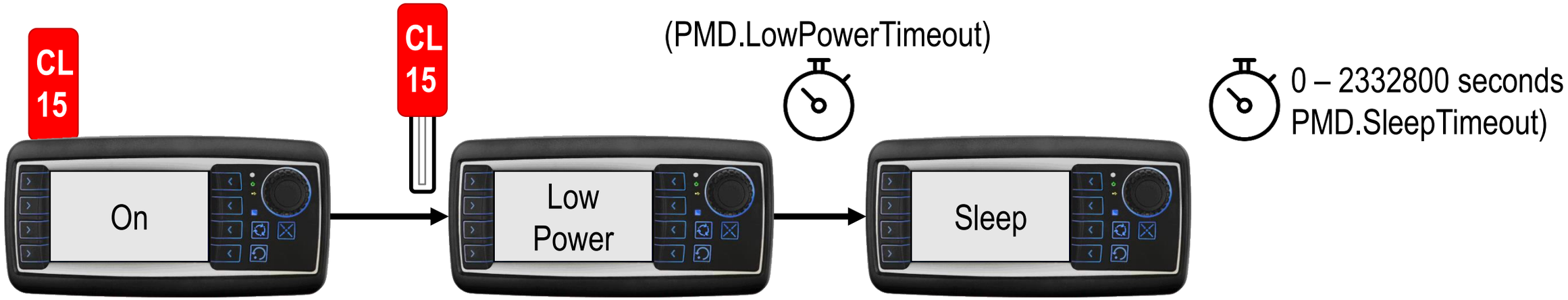
# 6.4. Extended agenda – Power management


mA @13.5 V

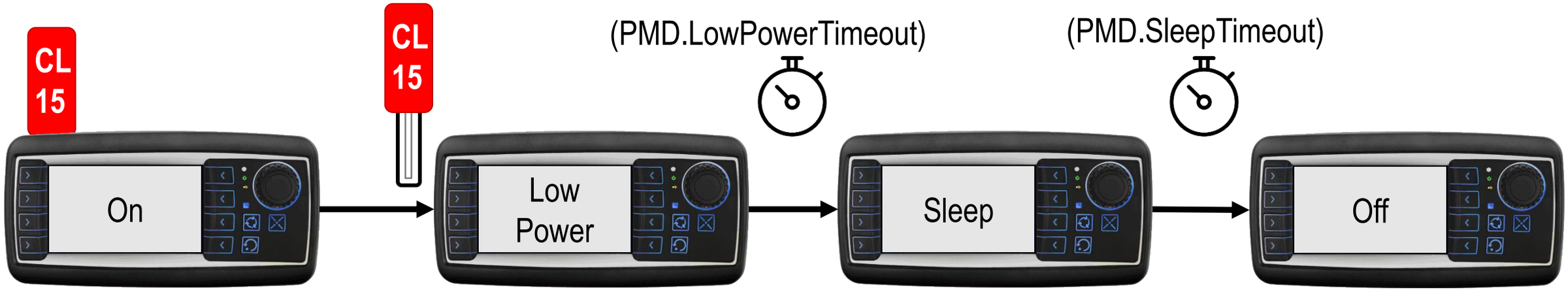
| | A3 | A6 | A6G2 | A8 |
|---|---|---|---|---|
| On | 430 | 900 | 1000 | 1600 |
| Low Power | 160 | 200 | 200 | 300 |
| Sleep | ≤ 90 | ≤ 100 | ≤ 100 | ≤ 200 |

Nothing is working, no functionality!

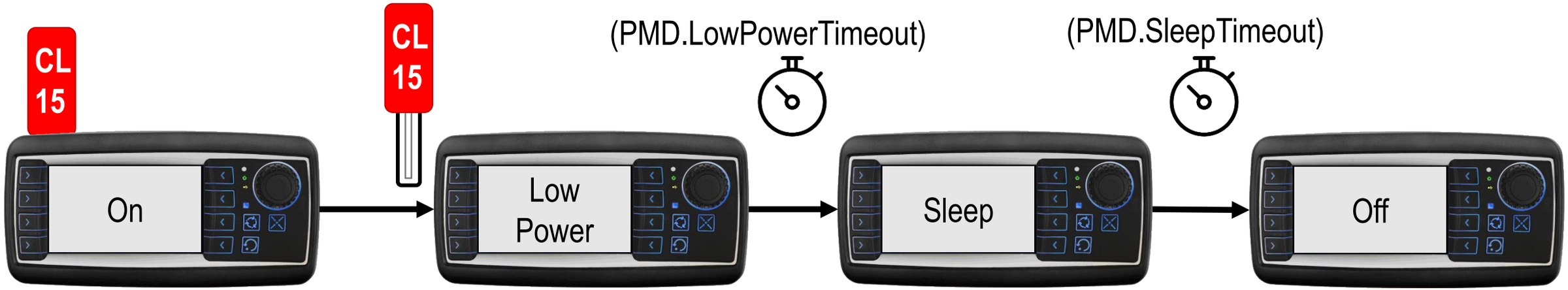
# 6.4. Extended agenda – Power management



# 6.4. Extended agenda – Power management



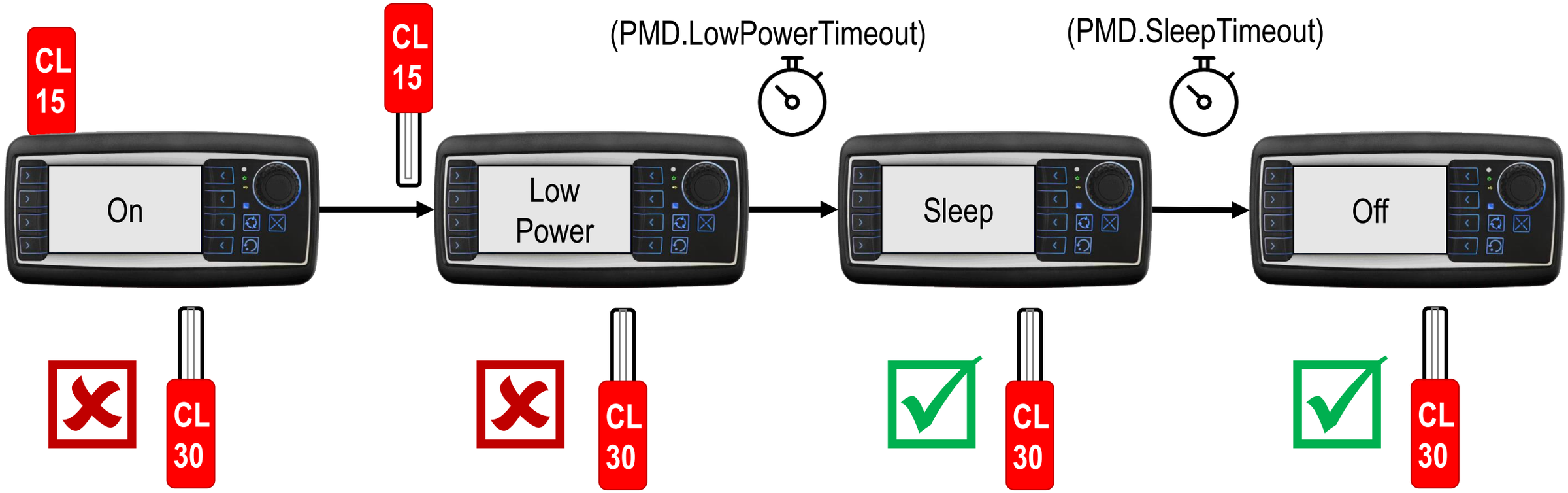
# 6.4. Extended agenda – Power management



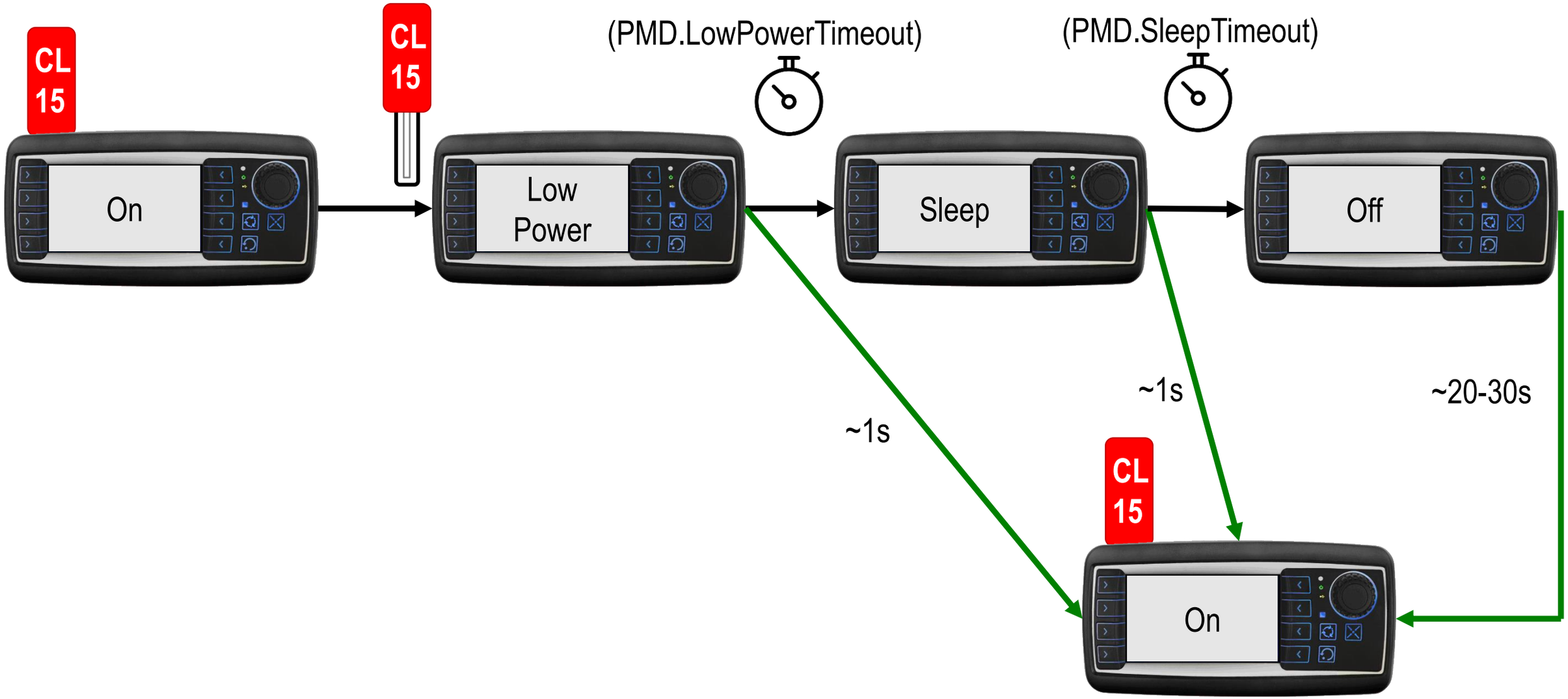

mA @13.5 V

| | 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 |

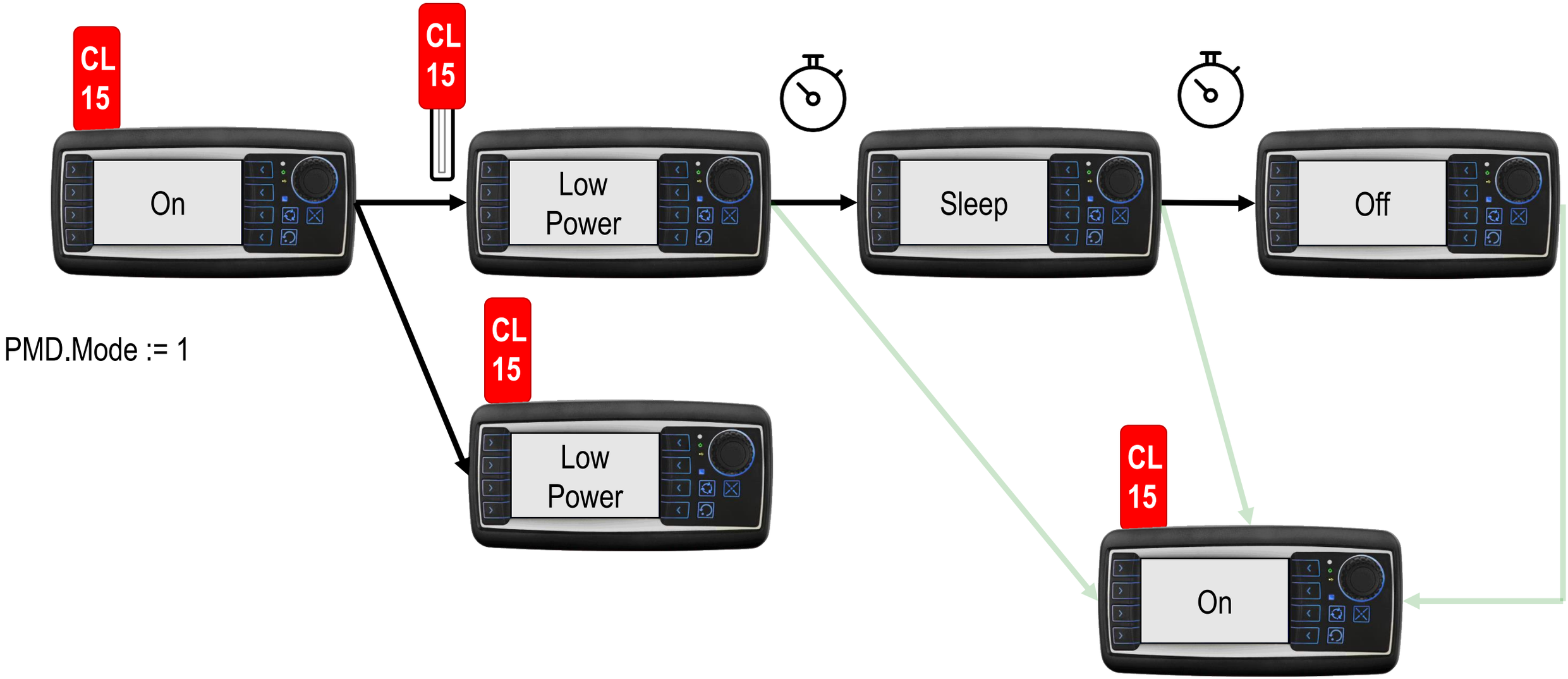
# 6.4. Extended agenda – Power management



# 6.4. Extended agenda – Power management

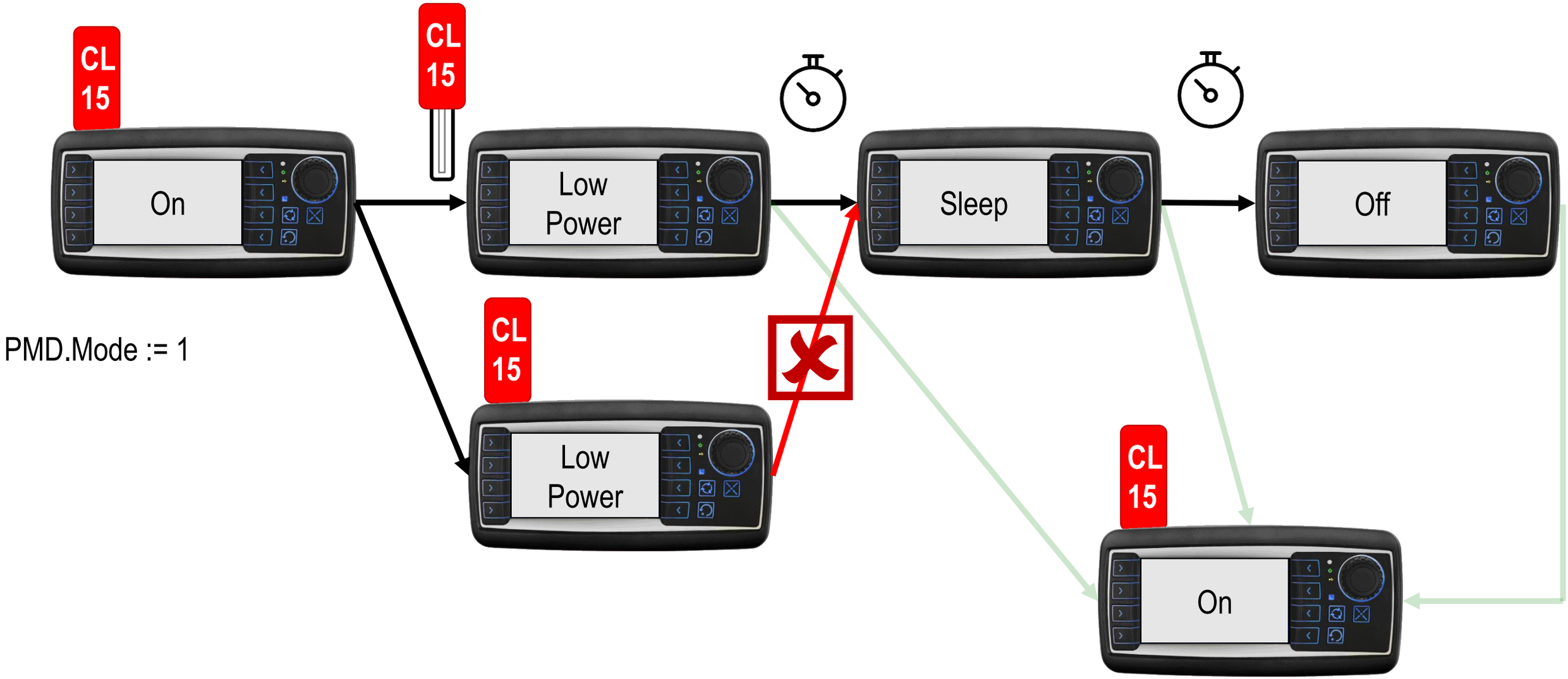


# 6.4. Extended agenda – Power management



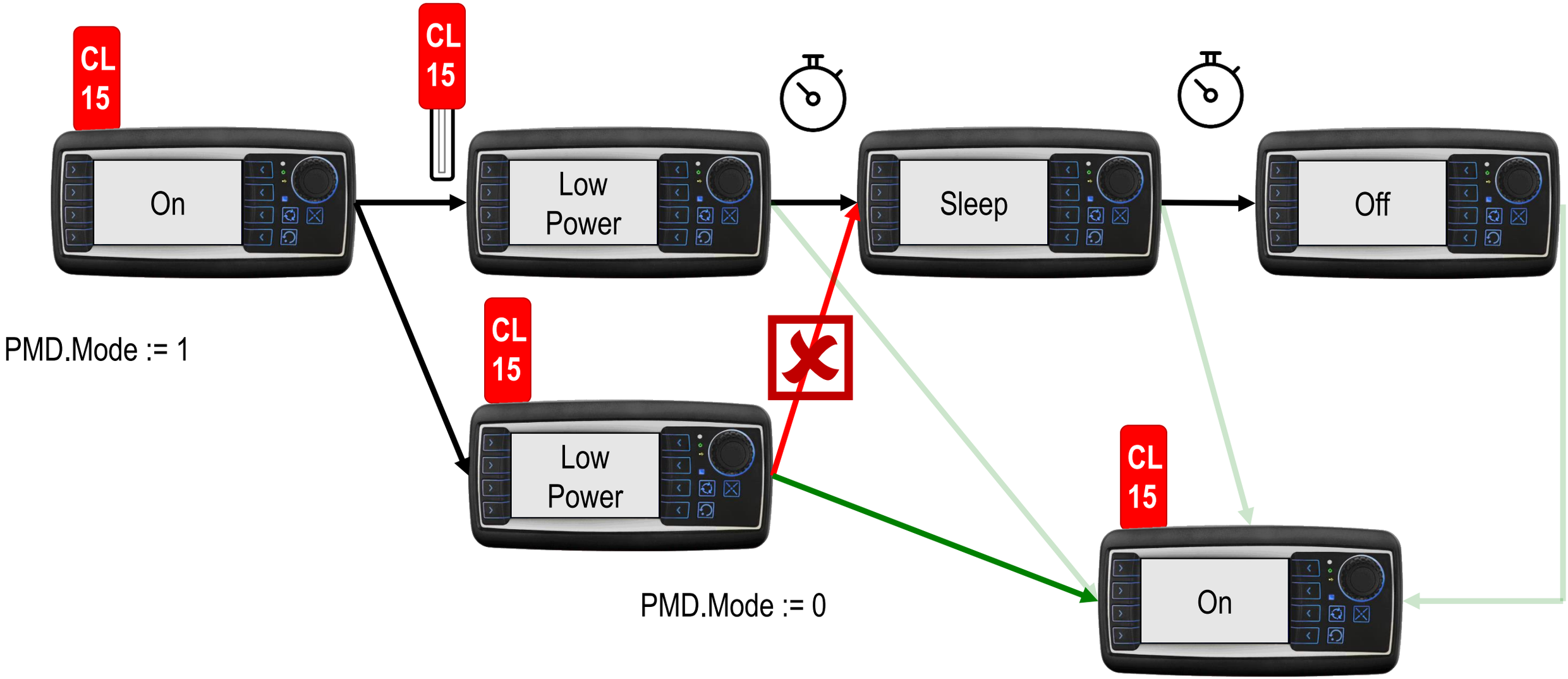

PMD.Mode := 1

# 6.4. Extended agenda – Power management

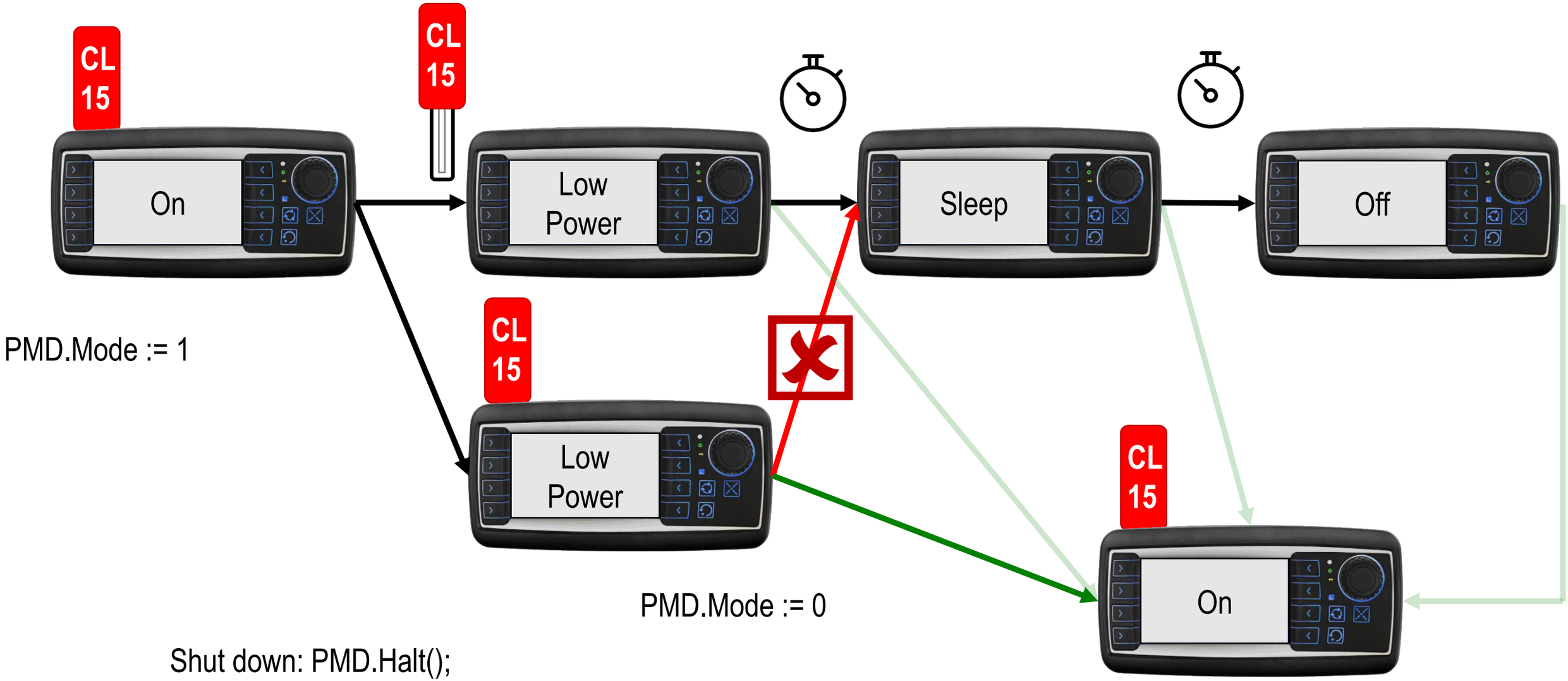



PMD.Mode := 1

# 6.4. Extended agenda – Power management

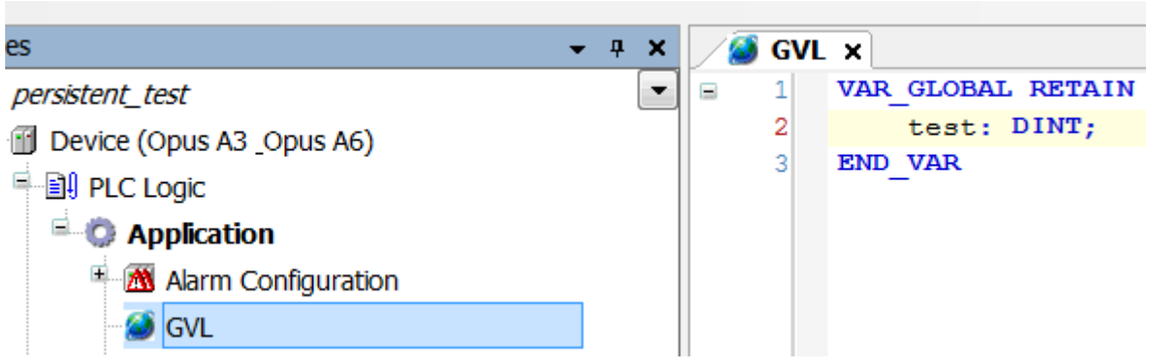


# 6.4. Extended agenda – Power management




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

# 6.5. Retain variables

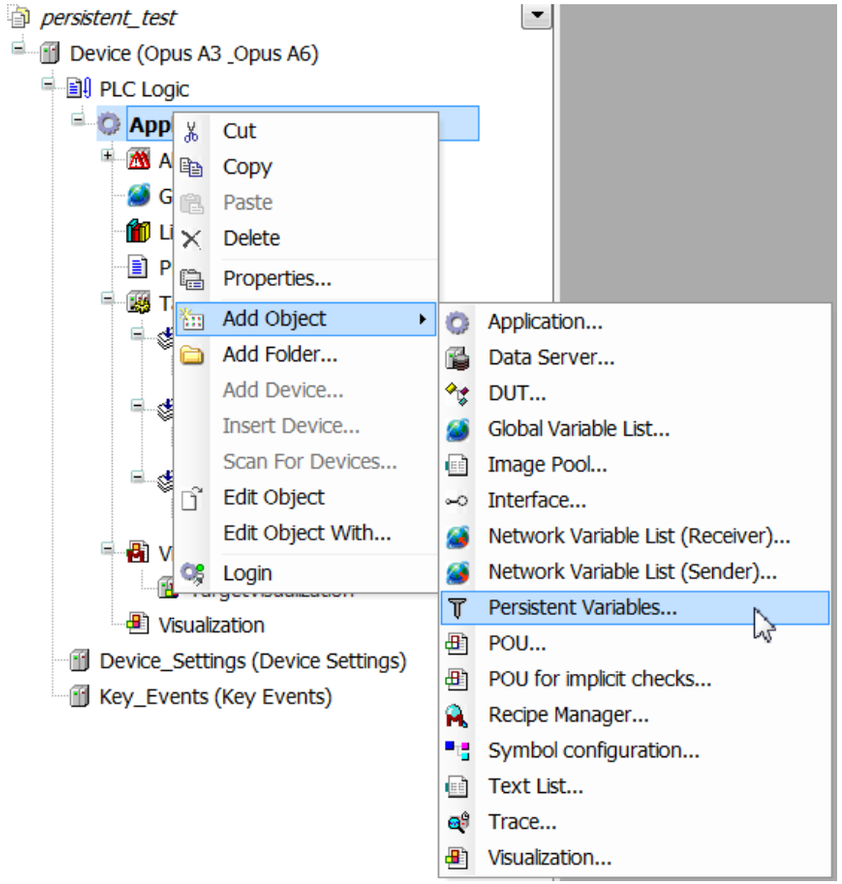



x = Value gets maintained  - = Value gets initialized

| 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

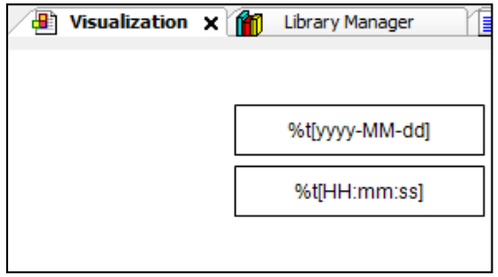
```
VAR
        result:RTS_IEC_RESULT;
        retname:STRING;
        pApp:POINTER TO cmpapp.APPLICATION;

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

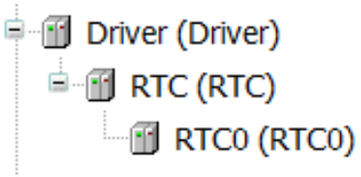
To save at runtime, include this code in your project (e.g. when leaving a settings page)

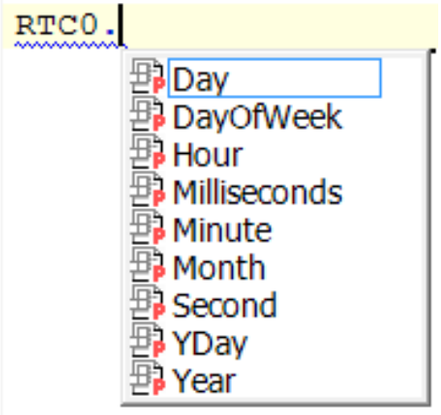
The processing takes a second, so don’t do it too often (not cyclic)

# 6.6. Date and time




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



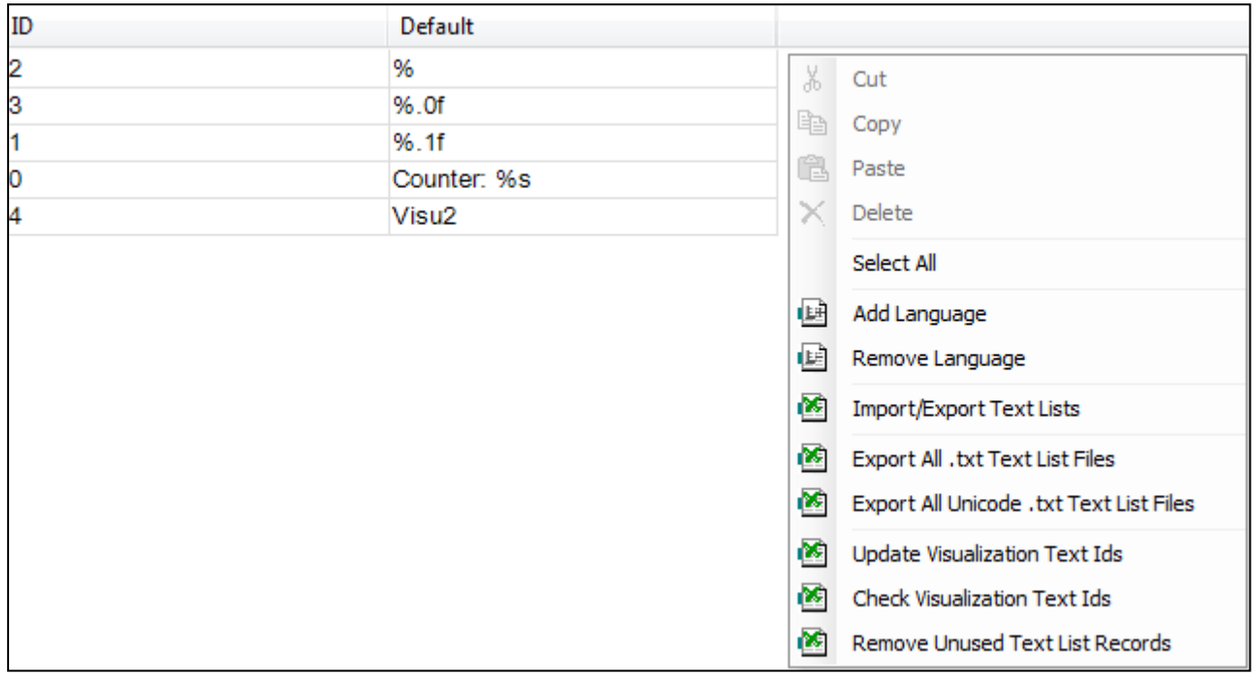




Showing the date and time from the RTC is very easy (in 24h format)

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

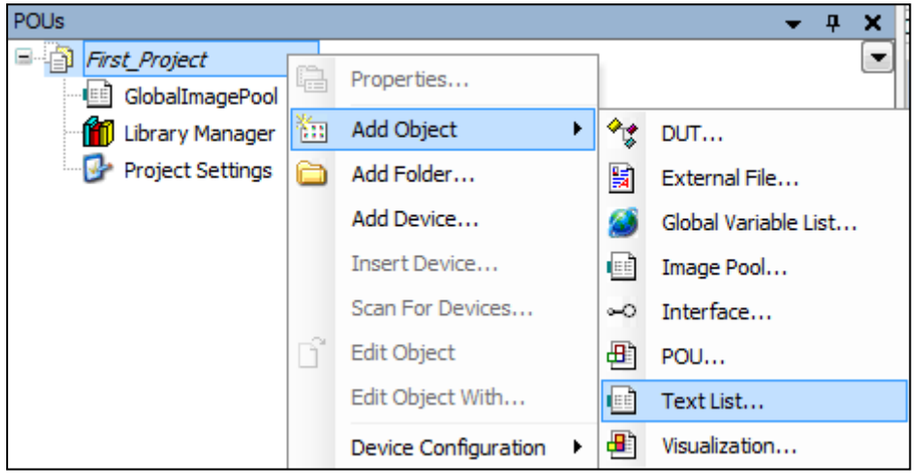



Every String used in the project can be found in the GlobalTextList

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

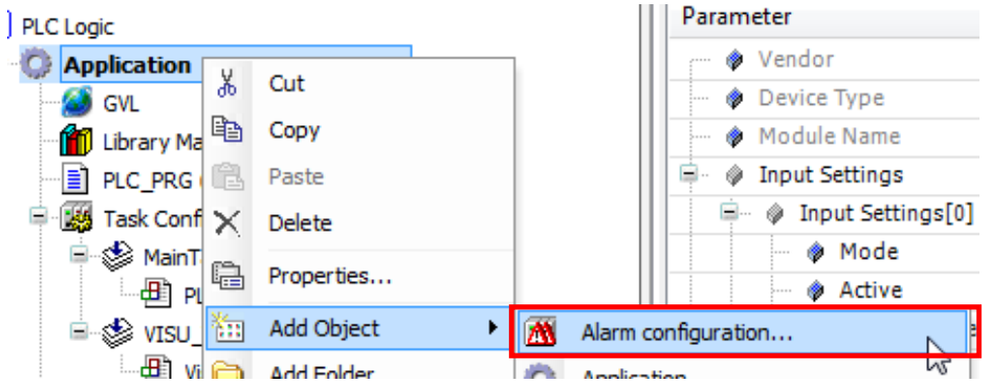
Connect the object with the text list

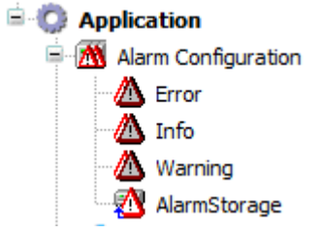
Switch languages with the variable CurrentLanguage (library VisuElemBase)

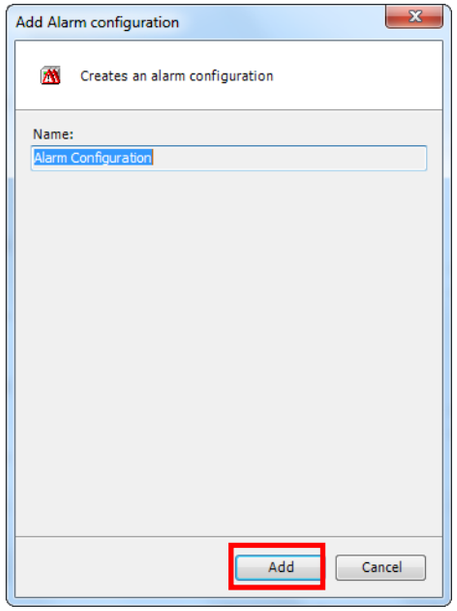
# 6.8. – Data logging / USB file transfer

Please check the example project USB_File_Operations

# 6.9. – Alarms



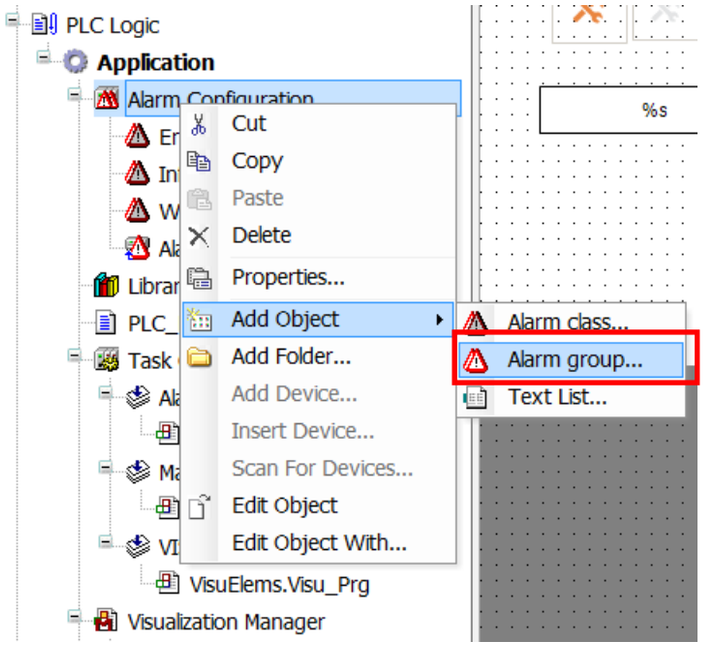

To create alarms, first add an *Alarm Configuration*

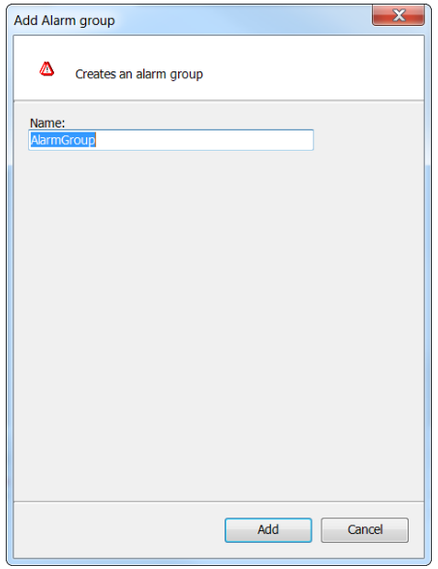
Rename if needed and click *Add*

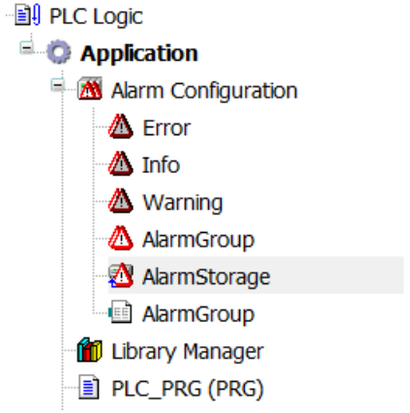
A structure with 3 alarm levels is created

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

# 6.9. – Alarms



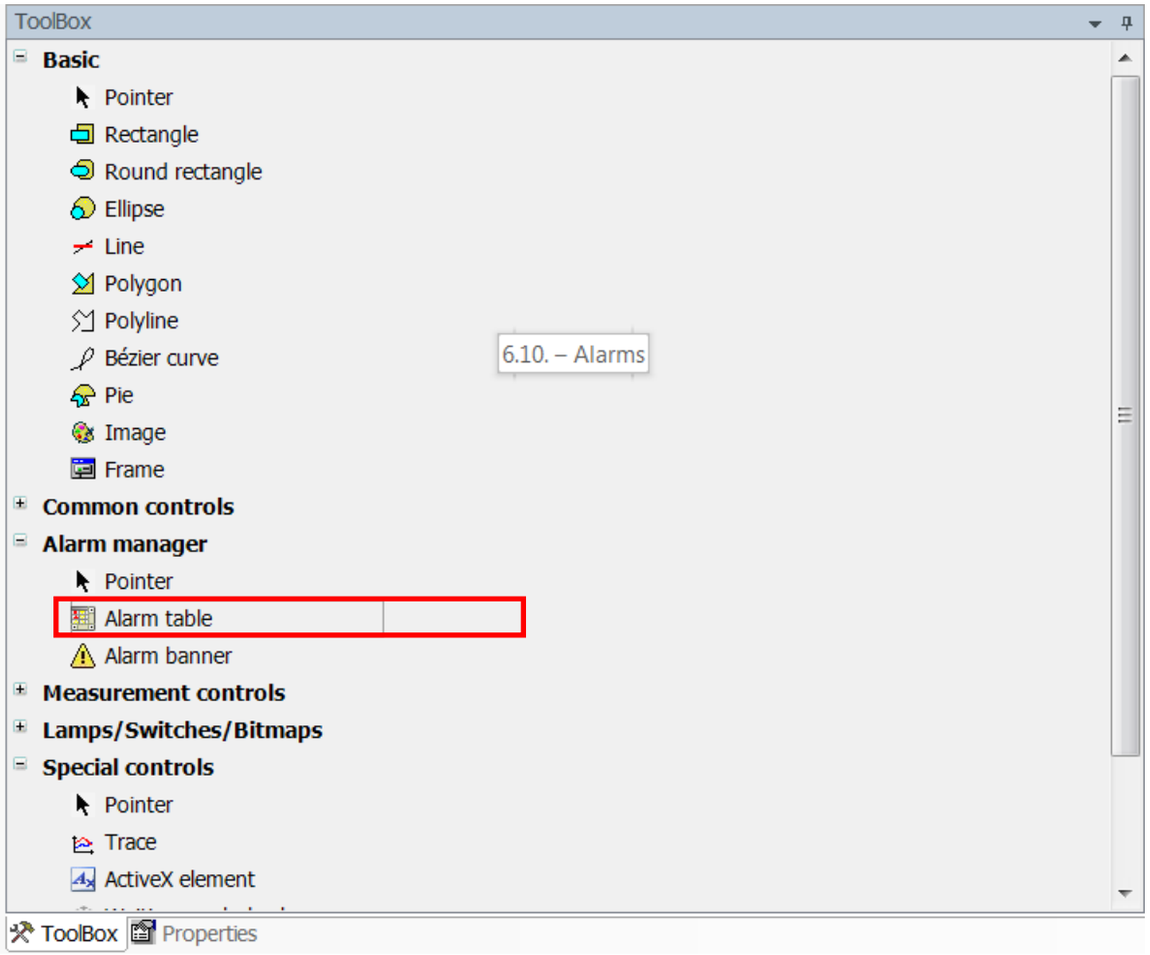




Add an alarm group and give it a name of your choice

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

```
PROGRAM PLC_PRG
VAR
    counter :INT := 10;
    up, down :BOOL;
    ack: bool;

END_VAR
```

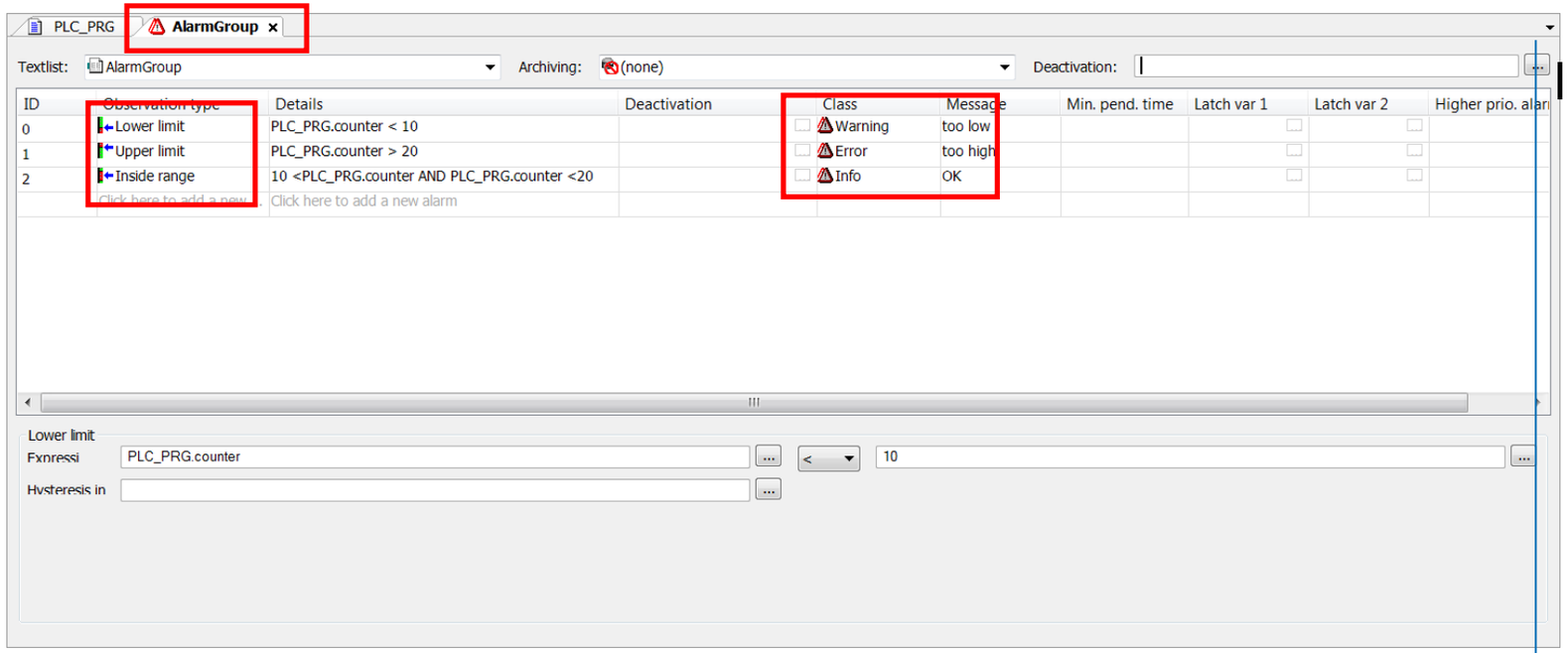
```
IF up THEN
    counter := counter +1;
END_IF

IF down THEN
    counter := counter -1;
END_IF
```

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

Create these variables and the code below

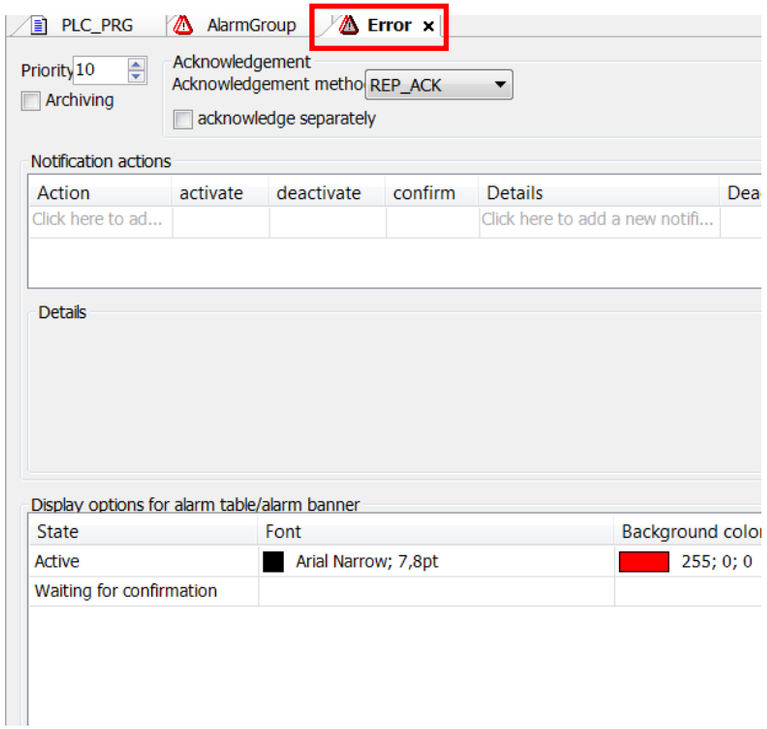
Use ack in the
Acknowledge all property of the table

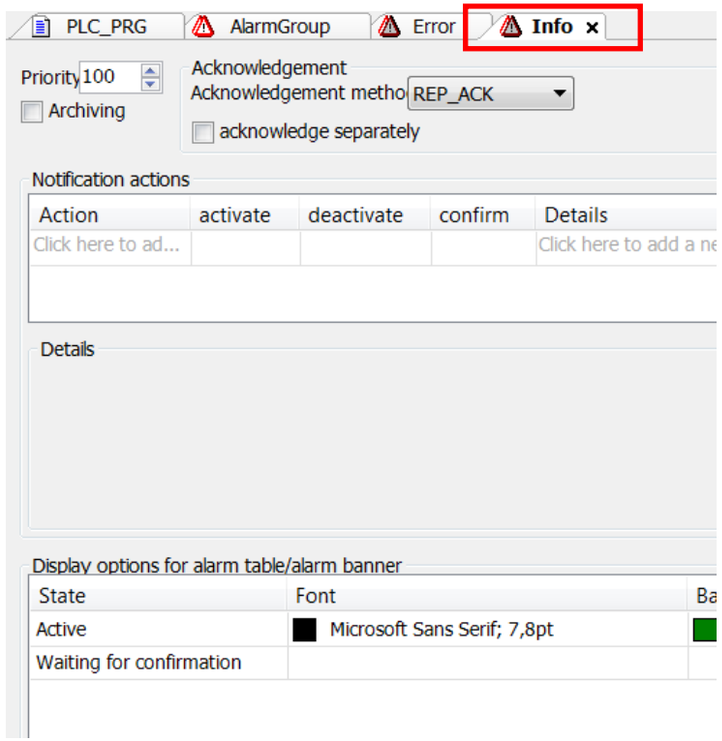
# 6.9. – Alarms




In the Alarm Group, create 3 observations

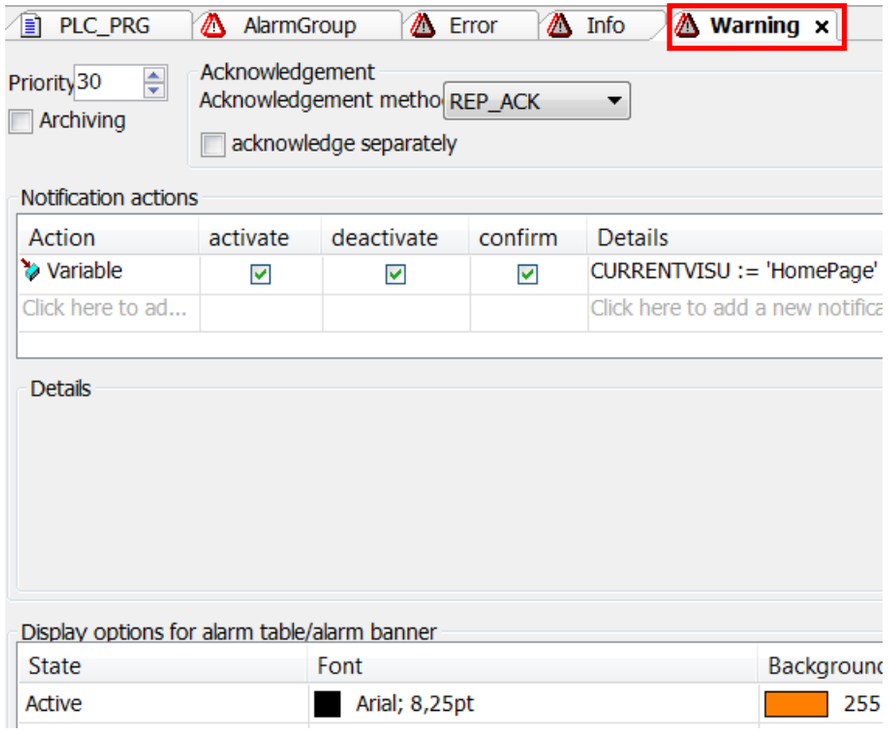
# 6.9. – Alarms




Configure the alarm levels

# 6.9. – Alarms




Configure the alarm levels

Then test the alarms by changing the counter variable

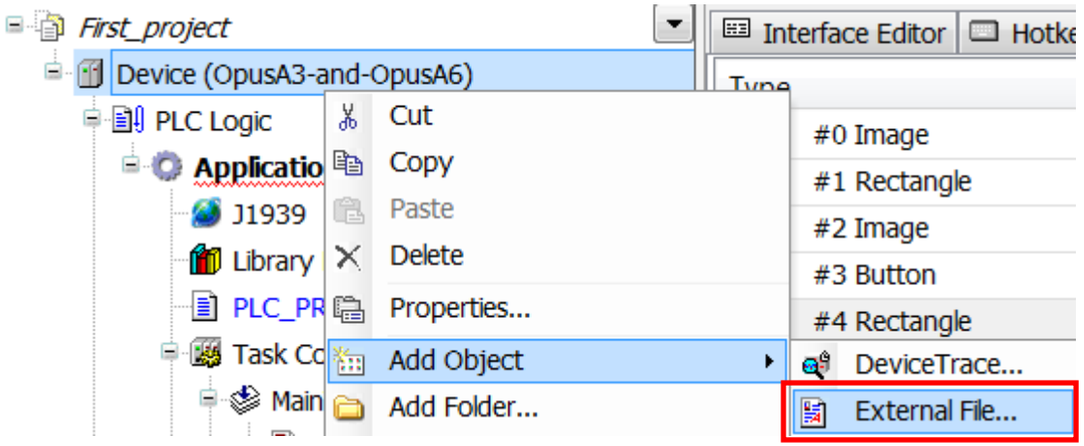
# Agenda

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?

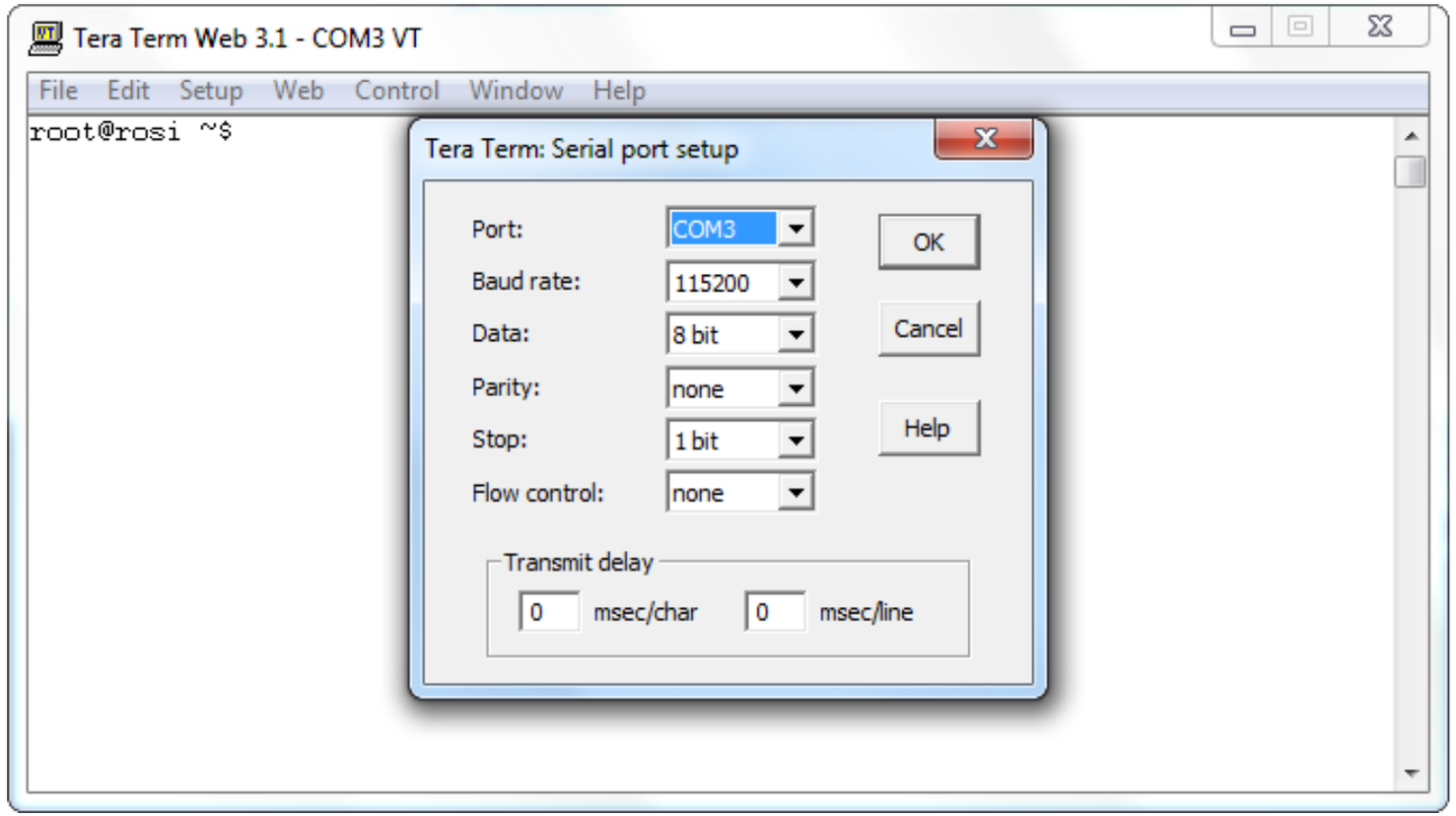



Fonts need to be installed in Windows so they can be chosen in Codesys

But the also need to be on the device.

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

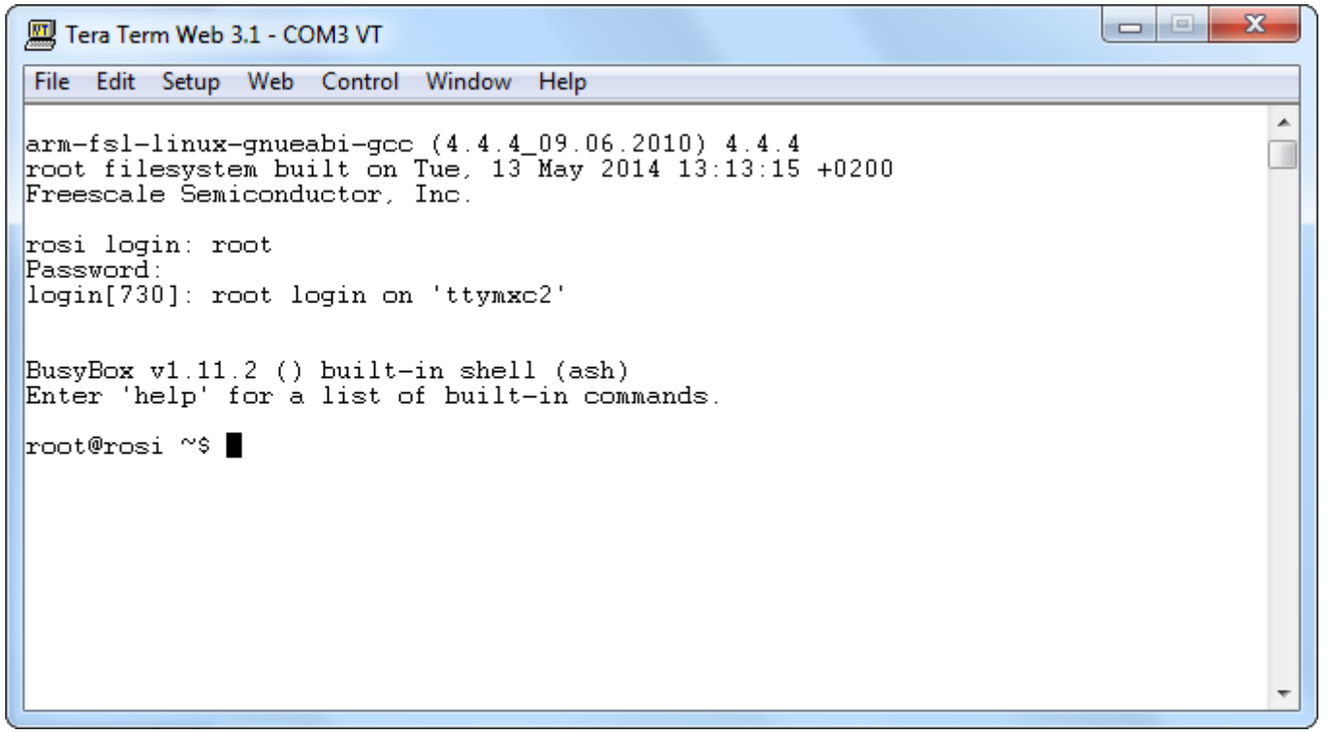
# 7.2 – FAQ – Connect to the serial console




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




To log in, use

username **root**
password “**opusa3**” or “**opusa6**”

# 7.2 – FAQ – Change the boot logo

Tera Term Web 3.1 - COM3 VT

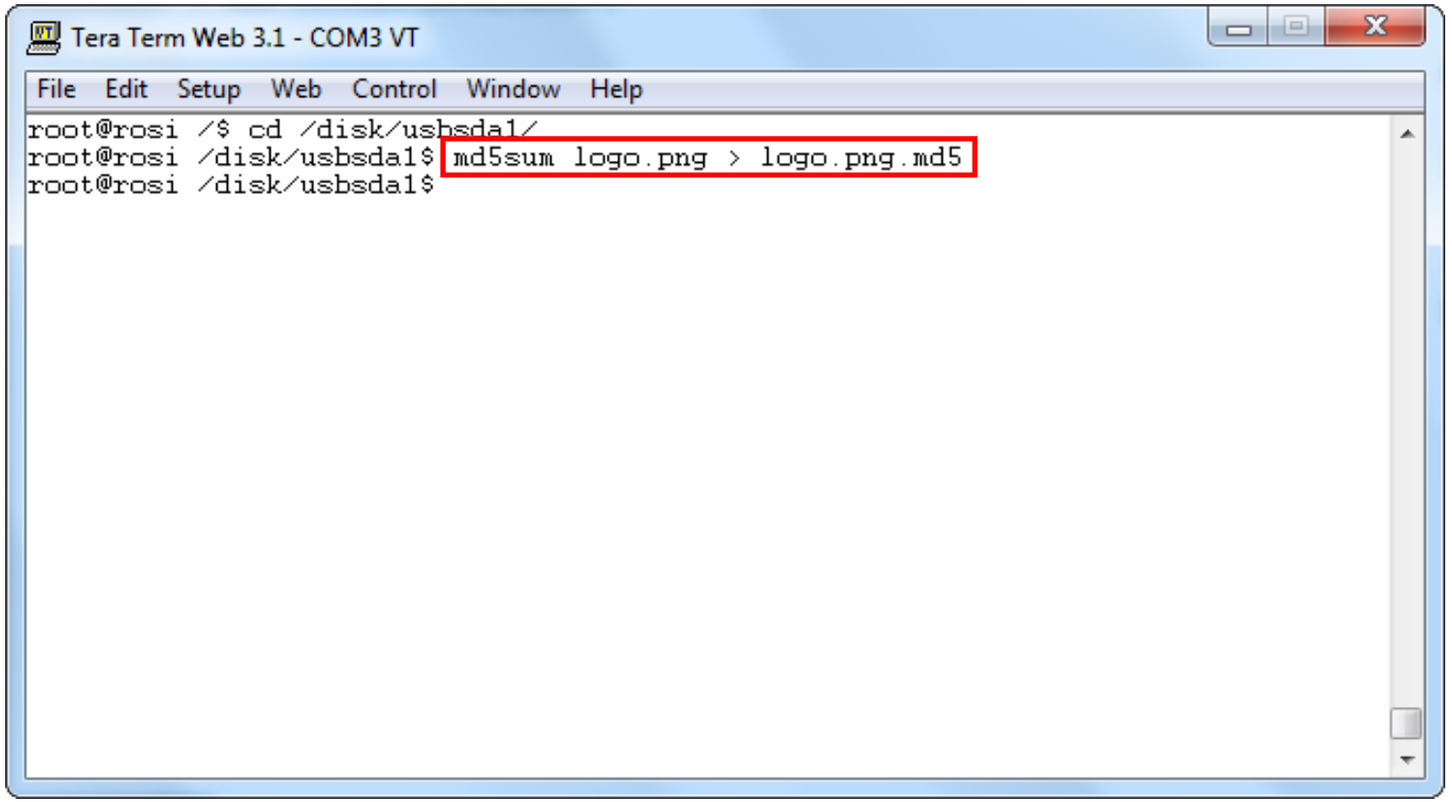
File Edit Setup Web Control Window Help

```
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 Kibyte @ 2e0000 -- 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

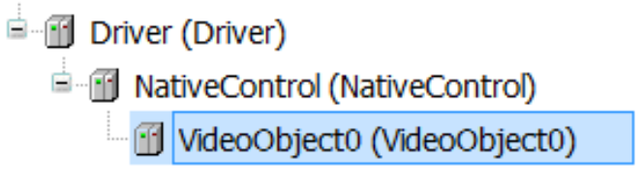



To change the boot logo for the update process, you need to create a checksum file.

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 ×

VideoObject0 Parameters
VideoObject0 I/O Mapping
Status
Information

| 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 |

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

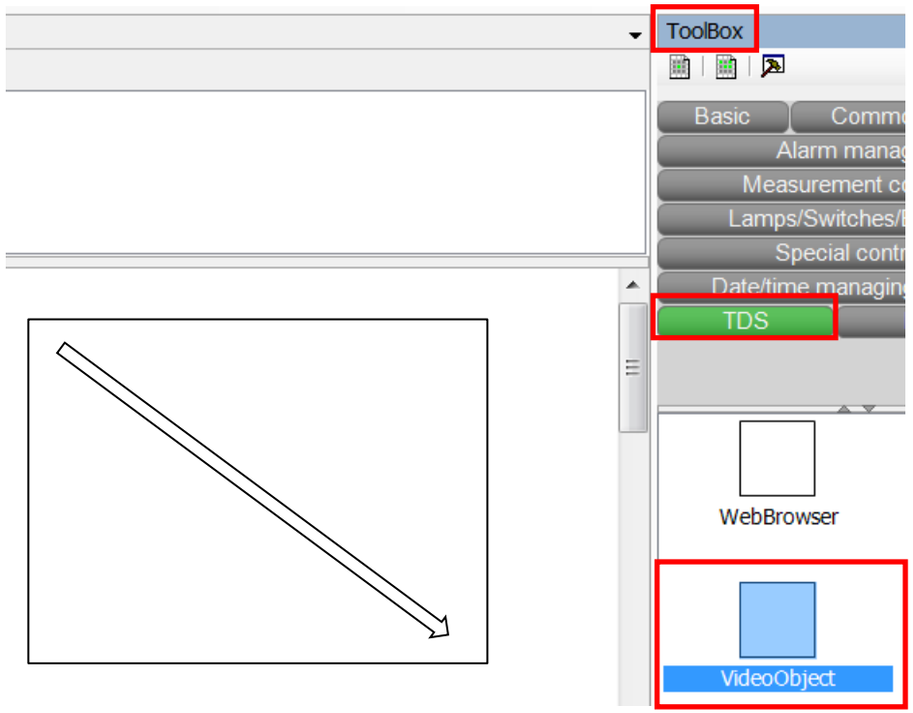
First you need to add a VideoObject to the Driver module

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

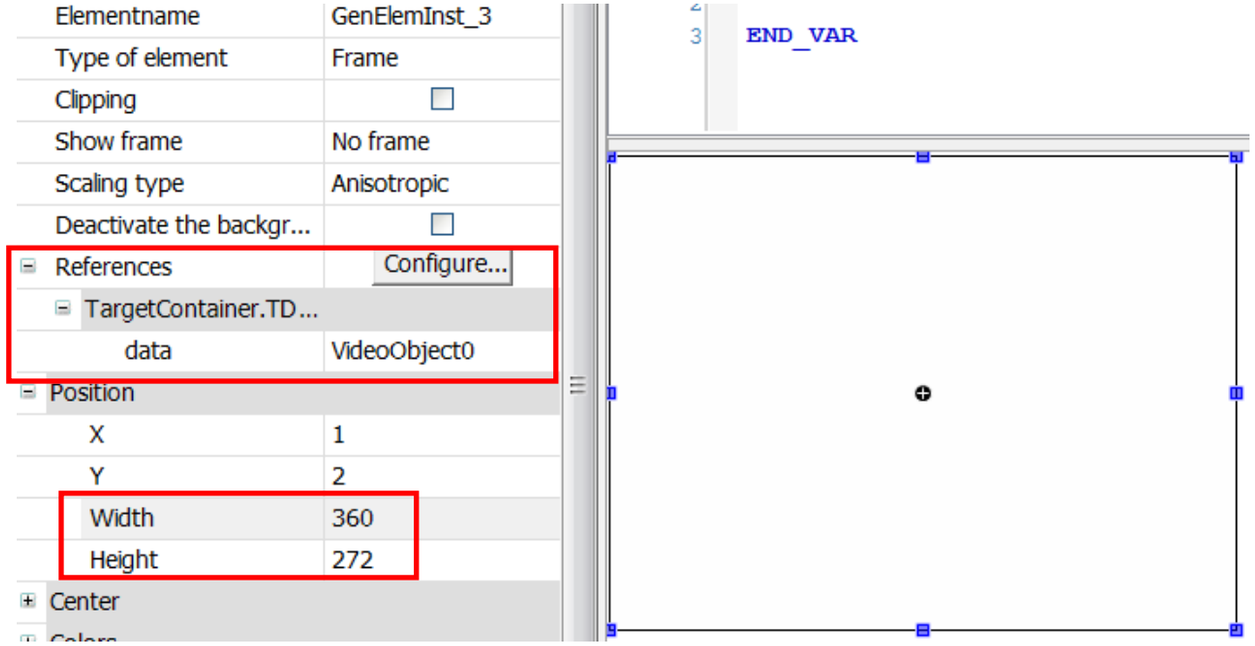
# 7.3 – FAQ – How can I display a camera image?




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?



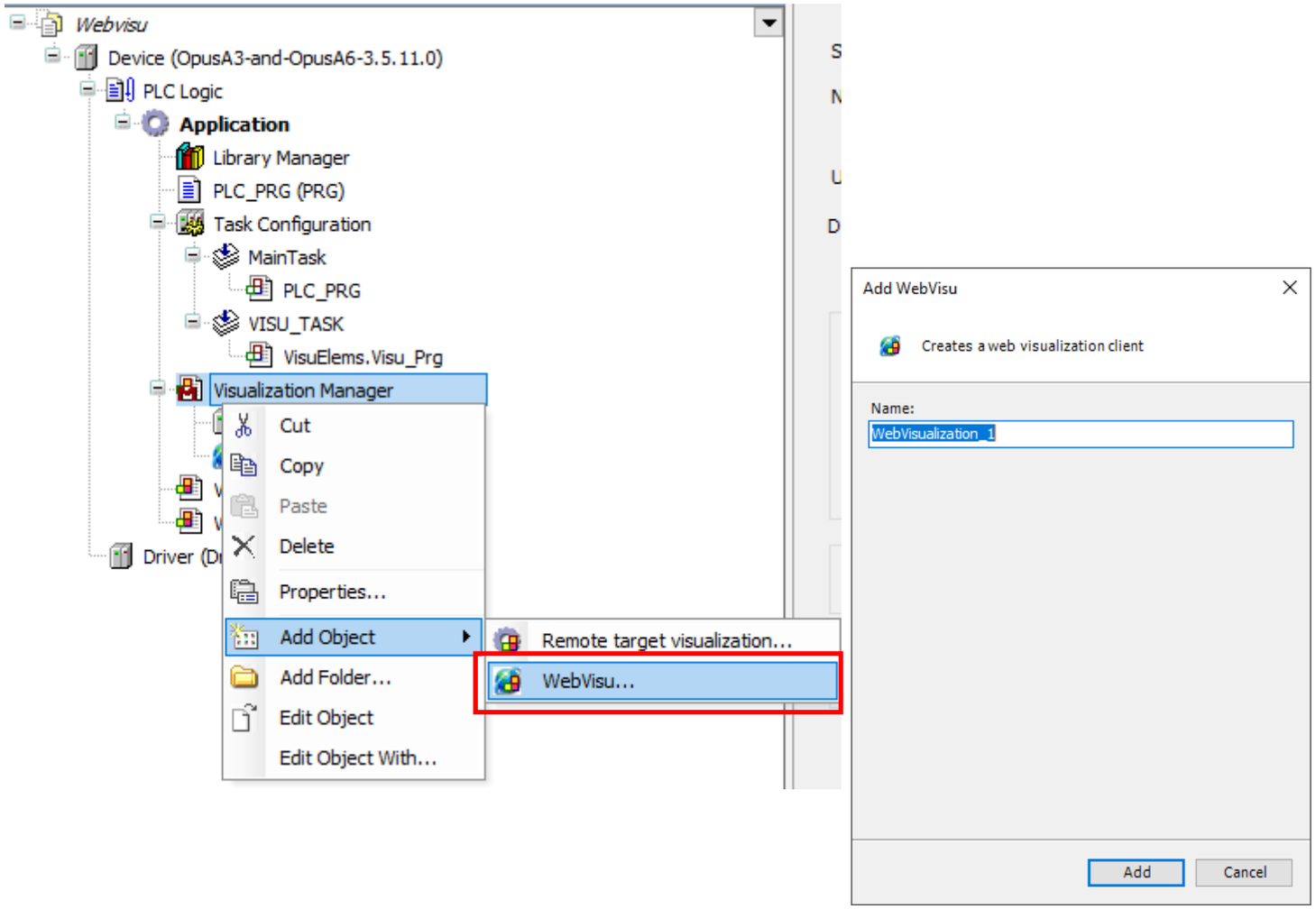

3 4

9 16

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

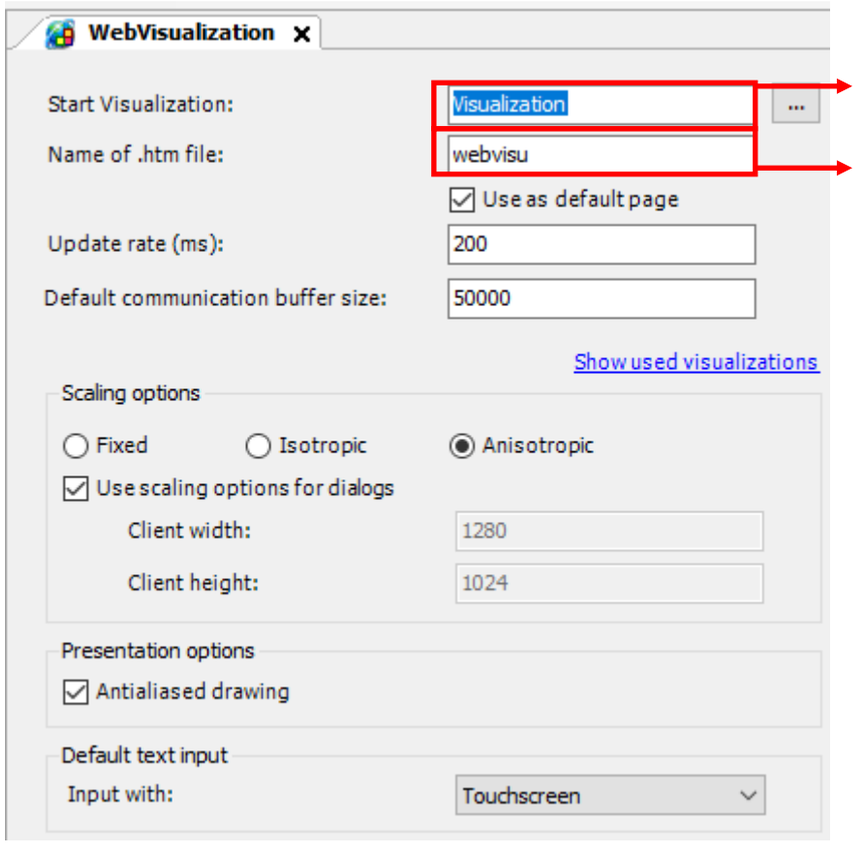
# 7.4 – FAQ – How can I use the Codesys Web Visu?




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?




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?

| | |
|---|---|
| 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