Application note

# Redundant Profinet Ring with an AC500 PLC and PNQ22.0 UMC100.3

2CDC135044D0201

In this Application Note will be described how to set up and create a redundant ring with two Profinet devices PNQ22.0 and four connected UMC100.3 to each device. In addition to that it will be described how to get the diagnostic for each slave.

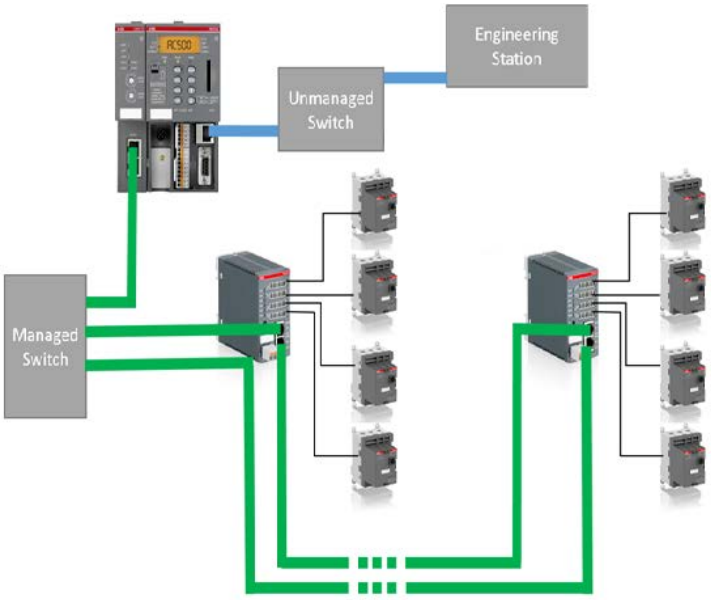
In the attachment to this application note is a complete example project included. This has to be opened and the CPU and Profinet master have to be adapted to the real used hardware. For an existing project the application can be imported into the project. For this use the attached "REDUNDANT_PROFINET_RING_WITH_AC500.EXP" file.

## Hardware setup

For this Application Note following devices are used:

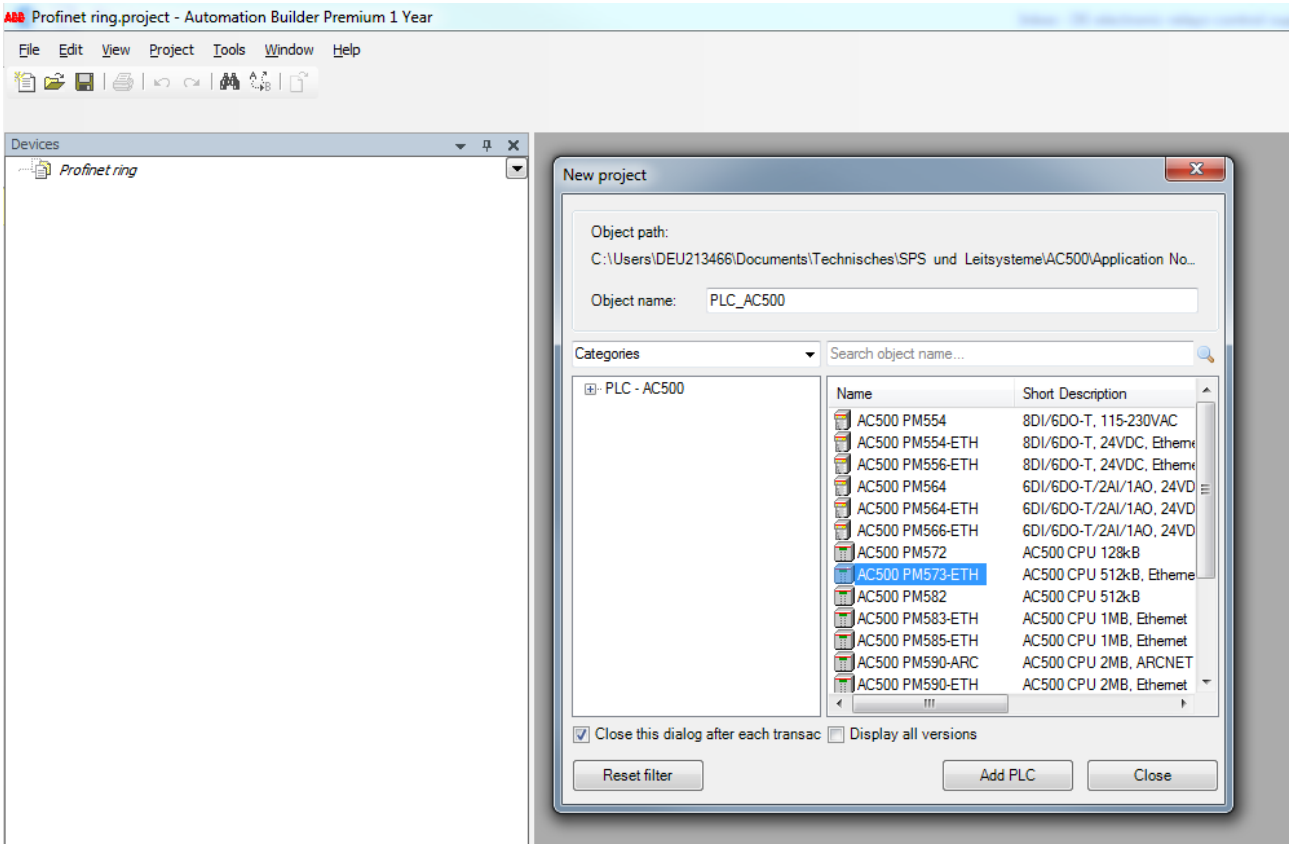
1. AC500
   a. CPU: PM573-ETH (1SAP130300R0271 A7, firmware V2.5.1, Build: 15409)
   b. Profinet master: CM579-PNIO (1SAP170901R0001, firmware 2.8.1 (2))
2. PNQ22.0 Profinet communication module
3. UMC100.3 Motorcontroller
4. Hirschmann switch (RS20, managed switch)
5. Hirschmann switch (RS2, unmanaged switch)

All mentioned devices will be connected like shown on the picture:

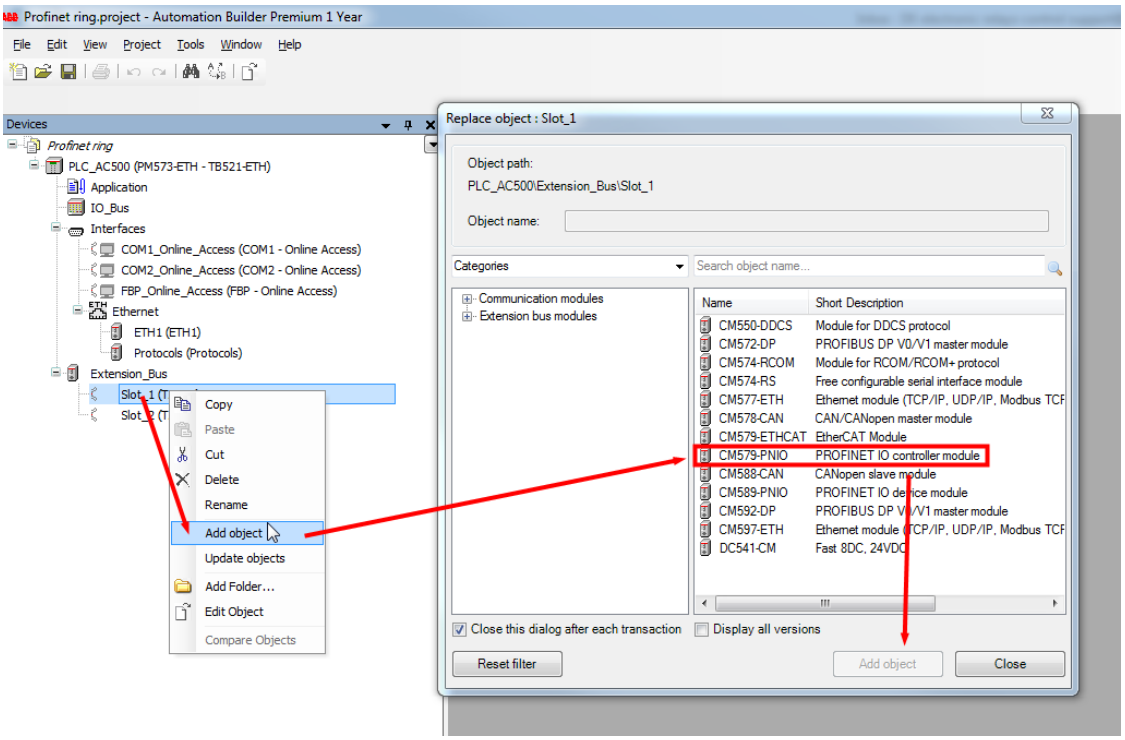


## Creating the hardware configuration for the project in Automation Builder

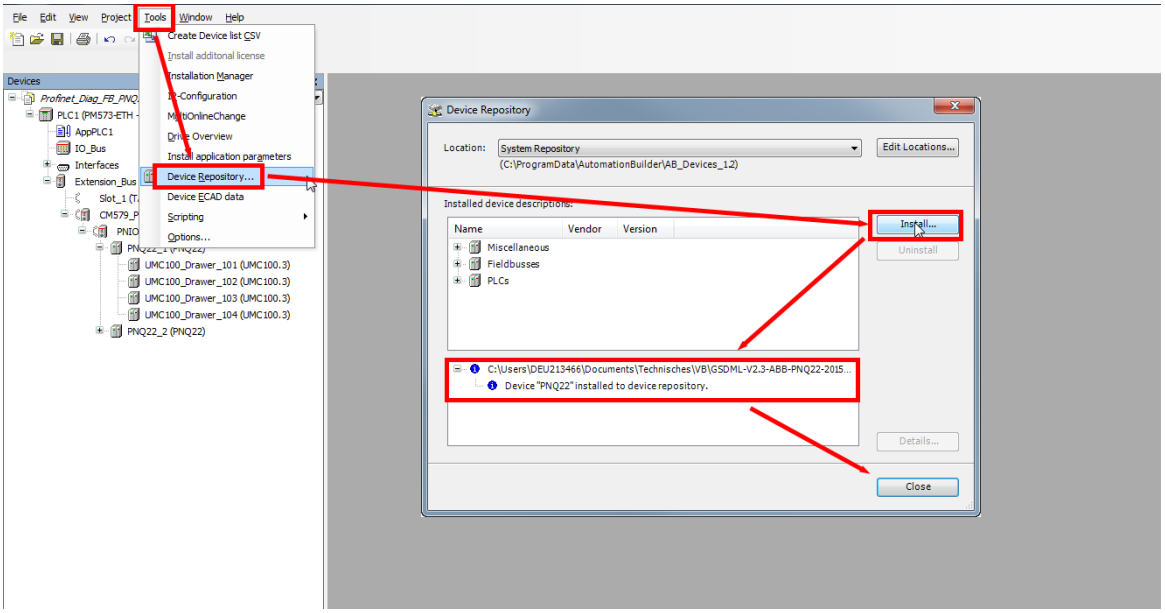
1. Create a project.
2. Add the used PLC (in this example PM573-ETH)



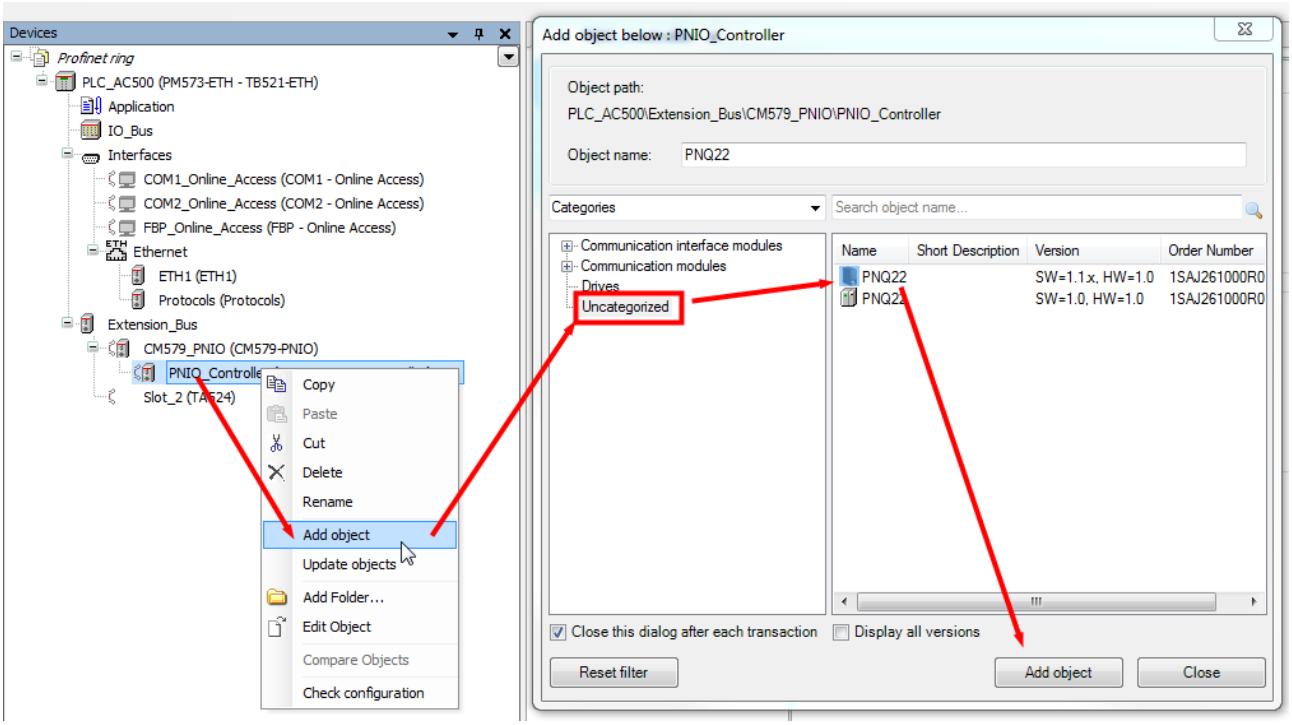

3. After selecting the CPU, a tree opens below the project name. Under “Extension bus” right click on one empty slot and click on “Add object”. Choose the Profinet master module CM579-PNIO and add this to the tree.



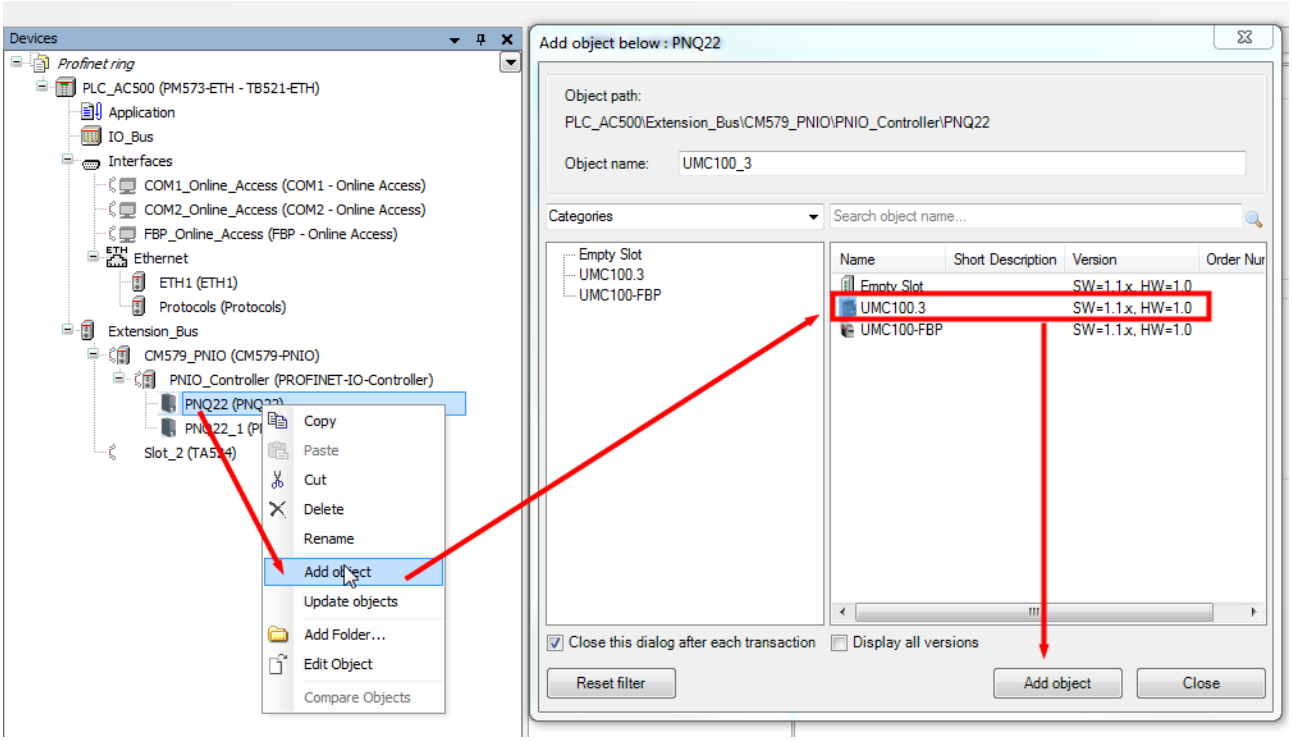

4. For adding the GSDML file open the “Tools” ribbon and click on “Device Repository…”.
5. Click on “Install” and open the GSDML file.
6. When the message from below picture appears, the installation was successful.



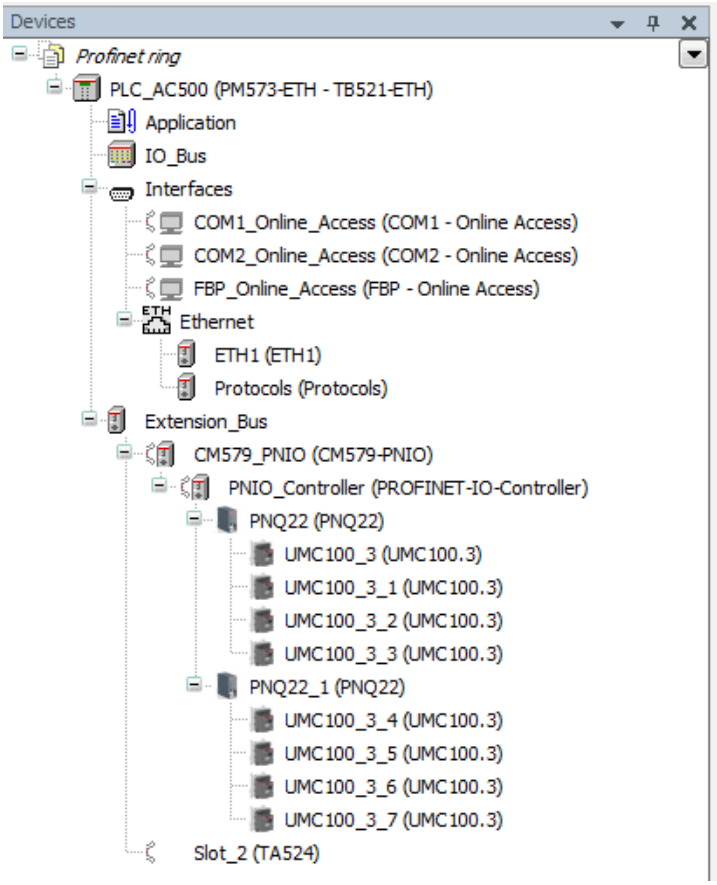

7. Right click on “PNIO_Controller (PROFINET-IO-Controller)” and select “Add object”. In “Uncategorized” the PNQ22 can be found. Select it and click on “Add object”.




8. Repeat Step 7 to get a second PNQ22.0 (repeat step 7 until you have the amount of used PNQs)
9. Right click on “PNQ22 (PNQ22)” and select “Add object”. Select the used motor controller or “Empty Slot” if no device is connected to the port of PNQ22.0.




10. Repeat Step 9 until all four ports of each PNQ are configured.

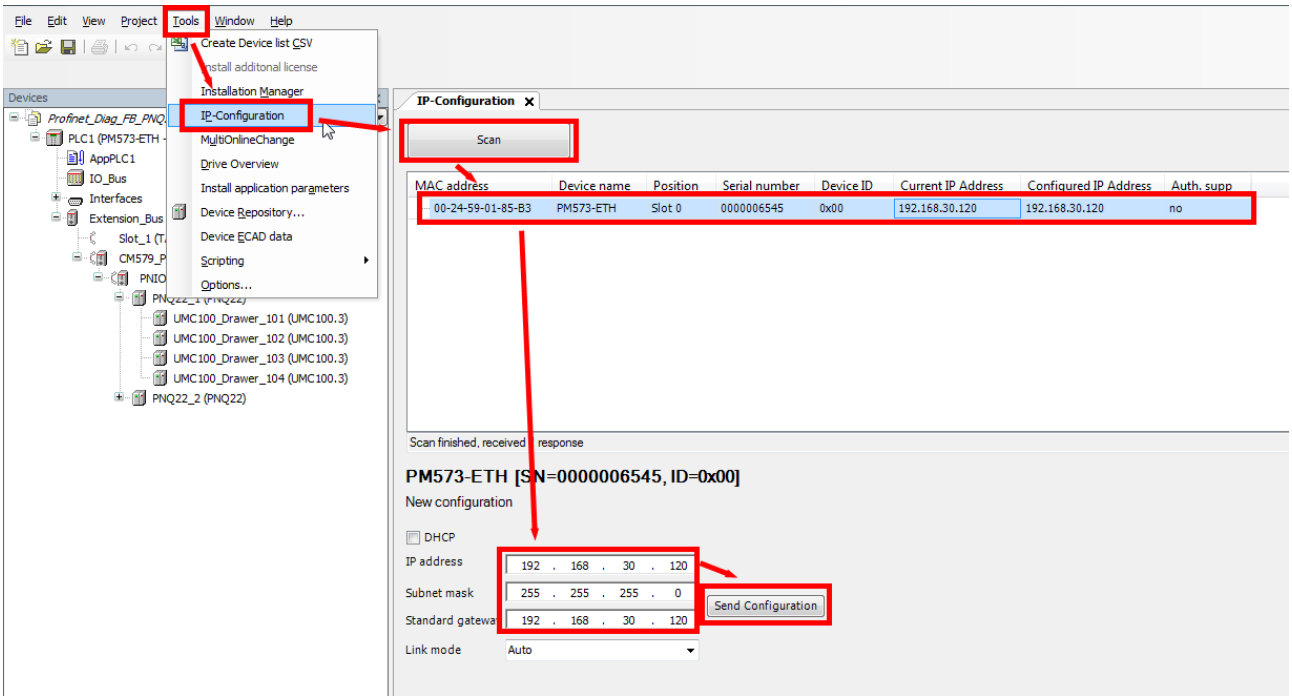



## Configuring the components

To have a working program/configuration there are some additional steps to do.

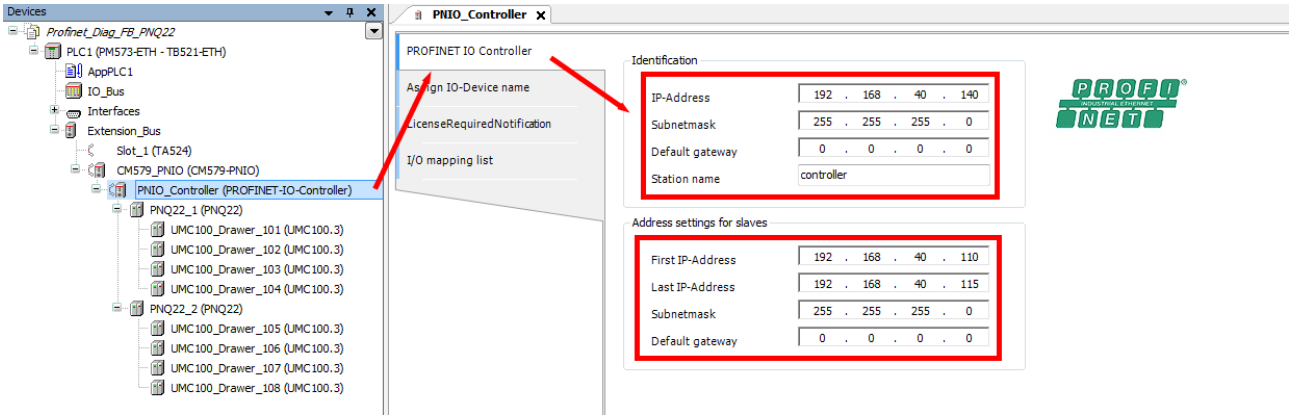
## PM573-ETH

1. Click on “Tools”.
2. Open the “IP-Configuration”.
3. After connecting and powering on the PLC, click on “Scan”.
4. Select the used PLC.
5. Below the window of found devices, the IP-parameters can be set.
6. Save the changes inside the PLC by clicking “Send Configuration”.




## CM579-PNIO

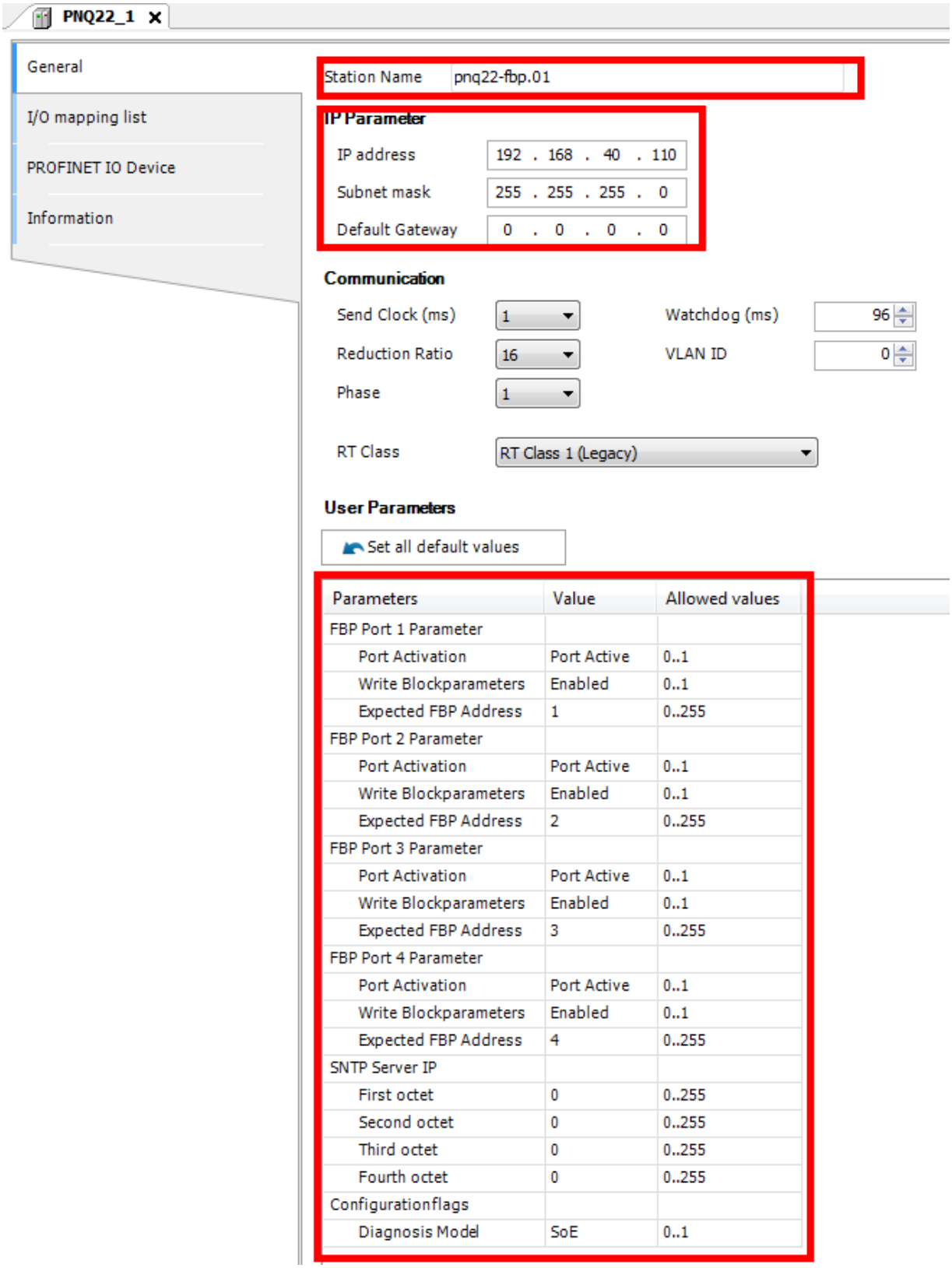
1. Double click on “PNIO_Controller (PROFINET-IO-Controller)”.
2. Open the “PROFINET IO Controller” ribbon
3. Set the master IP-address and the station name of the Profinet controller
4. Set the IP-range for the slaves.




## PNQ22.0

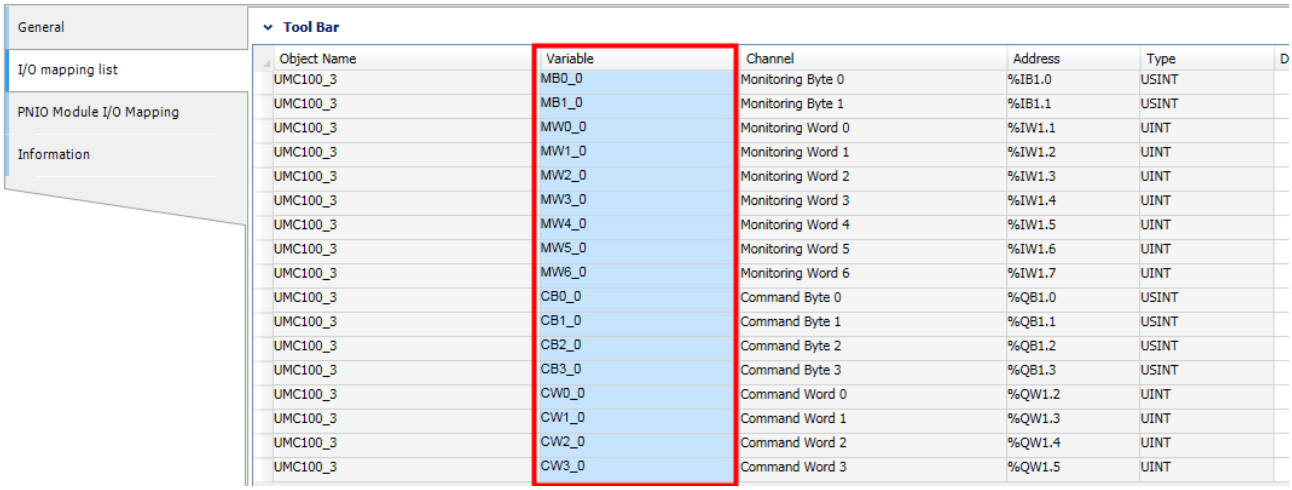
1. Double click on the PNQ22 in the list.
2. Open the “General” ribbon.
3. Set the station name and the IP parameter of the connected PNQ22.0.
4. Enable the ports where UMCs are connected and set the port (FBP-) address of the connected device.
5. Change the diagnosis model to “SoE” (Sequence of Events, This can be found on the bottom of the picture on the next page).
6. Save the changes.

7. Repeat the steps for each connected PNQ22.0.



## UMC100.3

1. Double click on the UMC below the PNQ22.0.
2. Open the “General” ribbon.
3. Set all parameter according the application.
4. Open the “I/O mapping list” ribbon.
5. Set unique names for the command and monitoring bytes/words, e.g.:



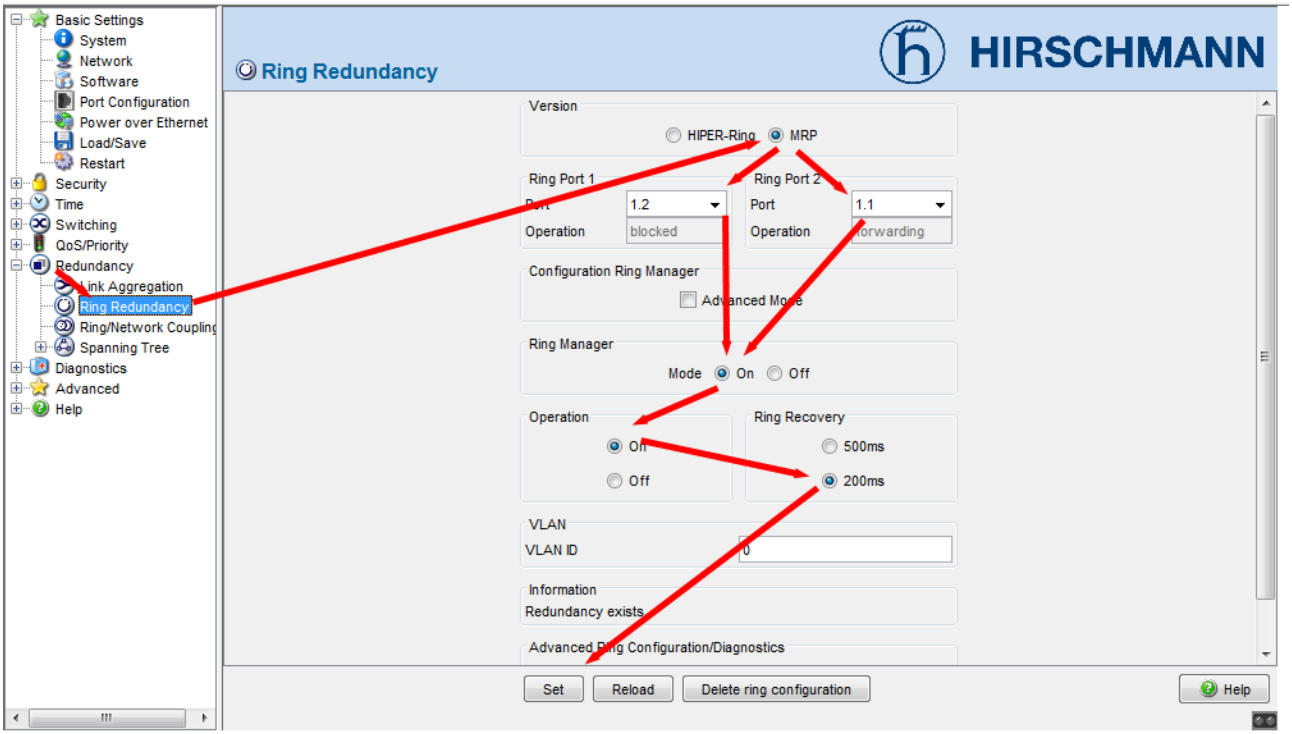

| Object Name | Variable | Channel | Address | Type |
|---|---|---|---|---|
| UMC100_3 | MB0_0 | Monitoring Byte 0 | %IB1.0 | USINT |
| UMC100_3 | MB1_0 | Monitoring Byte 1 | %IB1.1 | USINT |
| UMC100_3 | MW0_0 | Monitoring Word 0 | %IW1.1 | UINT |
| UMC100_3 | MW1_0 | Monitoring Word 1 | %IW1.2 | UINT |
| UMC100_3 | MW2_0 | Monitoring Word 2 | %IW1.3 | UINT |
| UMC100_3 | MW3_0 | Monitoring Word 3 | %IW1.4 | UINT |
| UMC100_3 | MW4_0 | Monitoring Word 4 | %IW1.5 | UINT |
| UMC100_3 | MW5_0 | Monitoring Word 5 | %IW1.6 | UINT |
| UMC100_3 | MW6_0 | Monitoring Word 6 | %IW1.7 | UINT |
| UMC100_3 | CB0_0 | Command Byte 0 | %QB1.0 | USINT |
| UMC100_3 | CB1_0 | Command Byte 1 | %QB1.1 | USINT |
| UMC100_3 | CB2_0 | Command Byte 2 | %QB1.2 | USINT |
| UMC100_3 | CB3_0 | Command Byte 3 | %QB1.3 | USINT |
| UMC100_3 | CW0_0 | Command Word 0 | %QW1.2 | UINT |
| UMC100_3 | CW1_0 | Command Word 1 | %QW1.3 | UINT |
| UMC100_3 | CW2_0 | Command Word 2 | %QW1.4 | UINT |
| UMC100_3 | CW3_0 | Command Word 3 | %QW1.5 | UINT |

6. Repeat these steps for each UMC (use different variables for each UMC).
7. Save all changes.

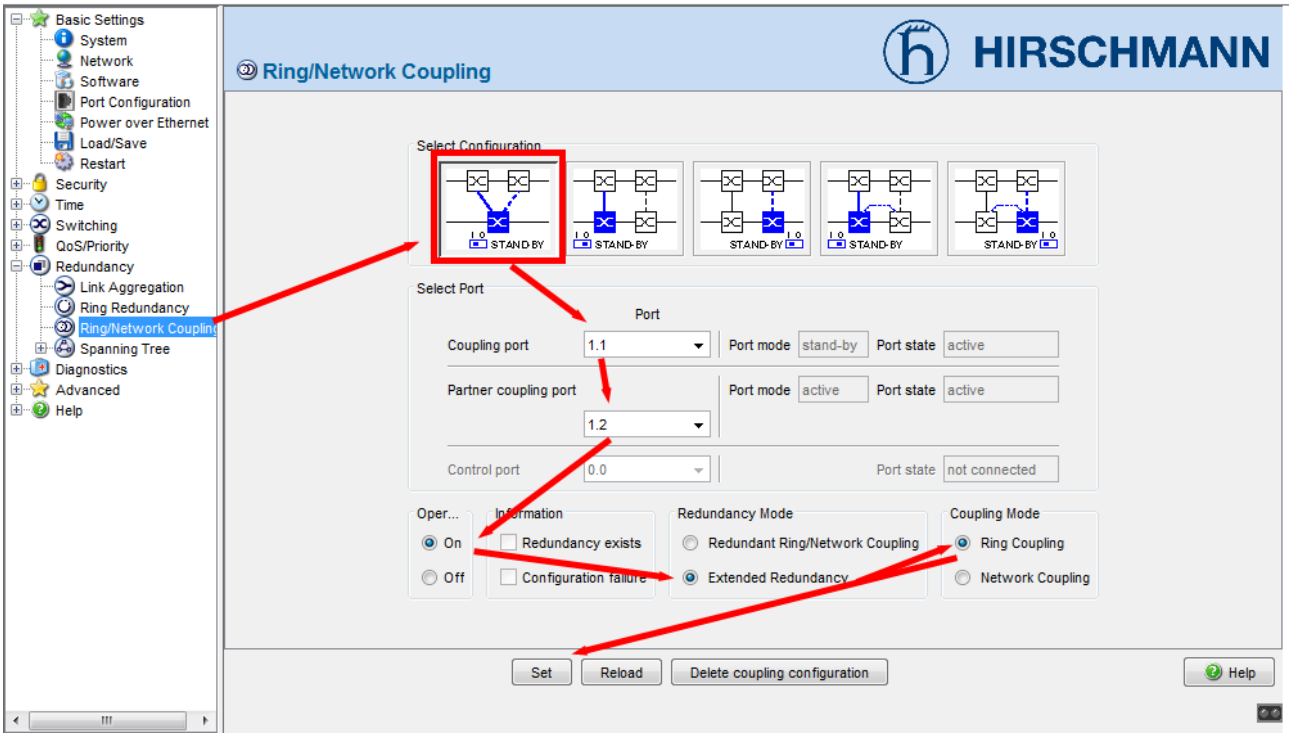
## Hirschmann Switch RS20

To get a Profinet ring the Switch has to be configured. The PNQ22.0 expects a MRP ring for proper operation.

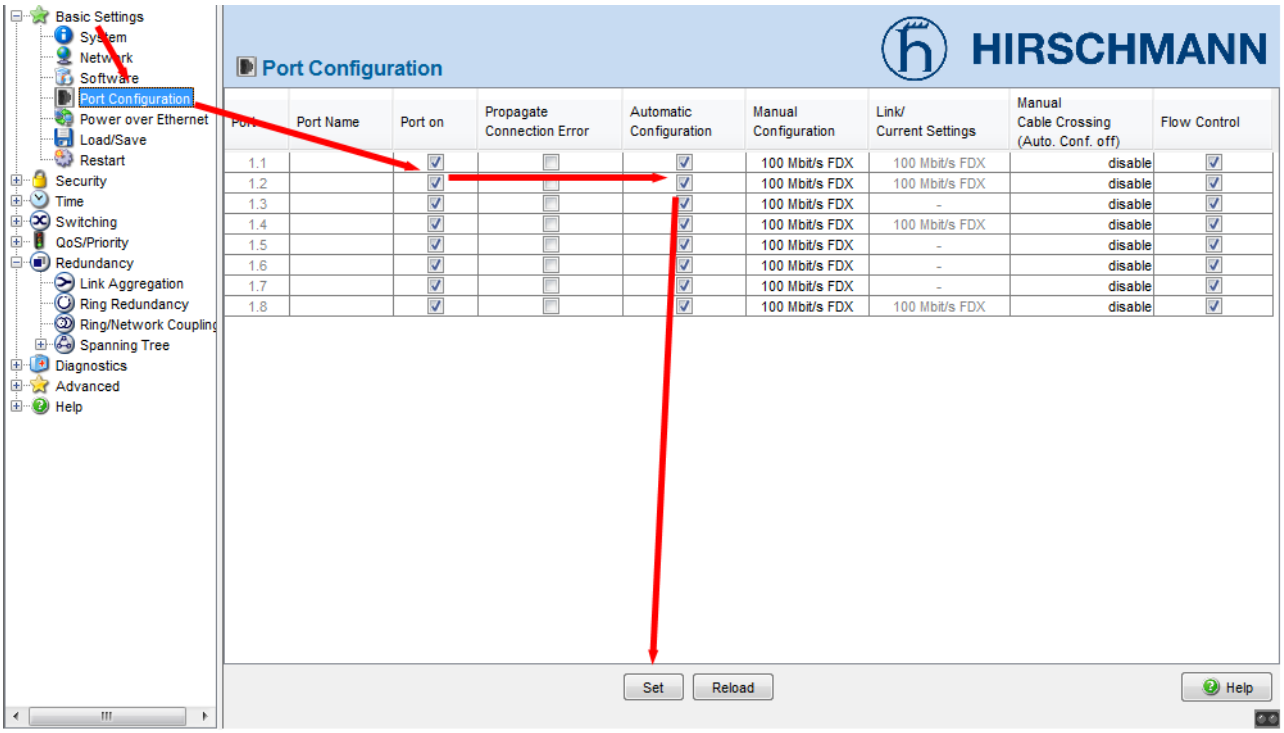
1. Connect the computer directly to the RS20 switch by using an Ethernet cable.
2. Open a Browser and type in the IP-address.
3. Login as admin (default password is “private”)
4. Open “Redundancy” → “Ring Redundancy”.
5. Select “MRP”, insert the ports where the ring is connected (in this case port 1 and 2)
6. Turn “Ring Manager” and “Operation” on and select a “Ring Recovery” of 200 ms.
7. Apply by clicking “Set”.



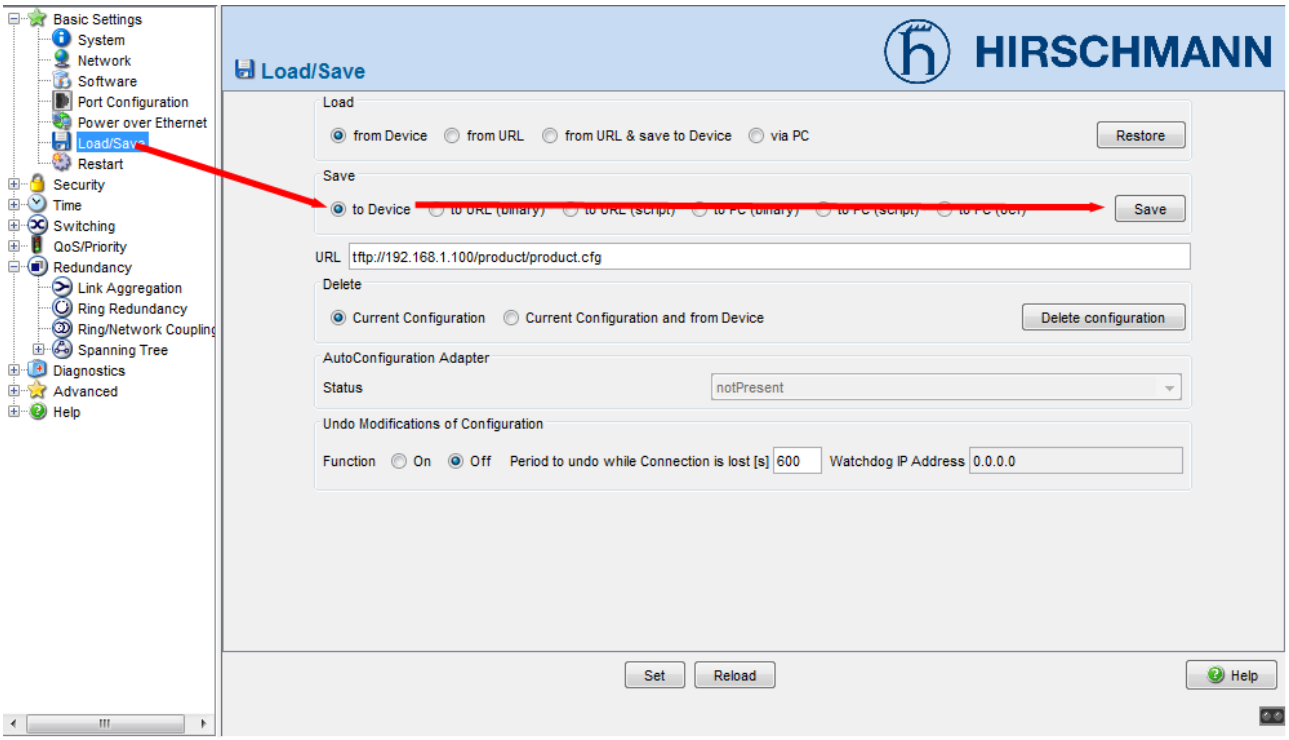

8. Open “Ring/Network coupling”
9. Select the first configuration and change the settings to those on the screenshot below.




10. Apply the settings by clicking “Set”.
11. Open “Basic Settings” → “Port configuration” and check if “Port on” and “Automatic Configuration” is checked for both ring ports. If changes are done, apply again with “Set”.




12. Go on “Load/Save” and mark the point “to Device” under “Save” and then click on the “Save” button to download all settings to the device.



## Hirschmann Switch RS2

No settings have to be done. The RS2 switch is an unmanaged switch.

## Configuration of cycle times, etc.

In this chapter is described how to configure timing of each component.

The following timings and factors have to be adapted:

| Name | Description |
|---|---|
| PLC CPU (PM573-ETH) | |
| PLC cycle | The time after that the main task is called again. |
| Min update time | This is the update time between the Profinet master and the CPU. After the set time the actual data will be sent to the CPU for further evaluation. |
| Profinet slave (PNQ22.0) | |
| Send clock | Parameter Send clock determines the SendCycle.<br>SendCycle = Send clock x Reducation ratio <= 512ms x |
| Reduction Ratio | The Reduction ratio determines the factor for calculating the cycle time.<br>Cycle time = Send clock x Reducation ratio |
| Phase | Defines the part of the SendCycle at which an IO frame is sent. |
| Watchdog | The watchdog supervises if the connection to the master is lost. If the time between two telegrams is longer than the watchdog time, the PNQ will indicate the lost master connection. |
| Datahold factor | If the Watchdog time has been expired, the Datahold time will be checked. If the Datahold time also has been expired, the substitution values of the IOs will be used. For RT_Class_1 communication Datahold time and the Watchdog time are usually configured with the same value. |
| Managed switch Hirschmann RS20 | |
| Ring Recovery | The switch is sending test packets for checking if the ring is closed. If the telegram didn't reach the receiving port in the set time the package is lost and the switch routes also through the second ring port. |

PLC cycle: There are several possibilities for setting the cycle time. We will only use the cyclic mode: In this mode the next cycle starts after the set interval is over (e.g. 20ms)

Ring Recovery: In the switch configuration of the RS20 switch are two recovery times possible:

1. 200ms
2. 500ms

The switch sends cyclically test packages. If after the selected time the test package didn't came back on the other ring port, it's lost and the switch routes the data to both ring ports to communicate with all slaves.

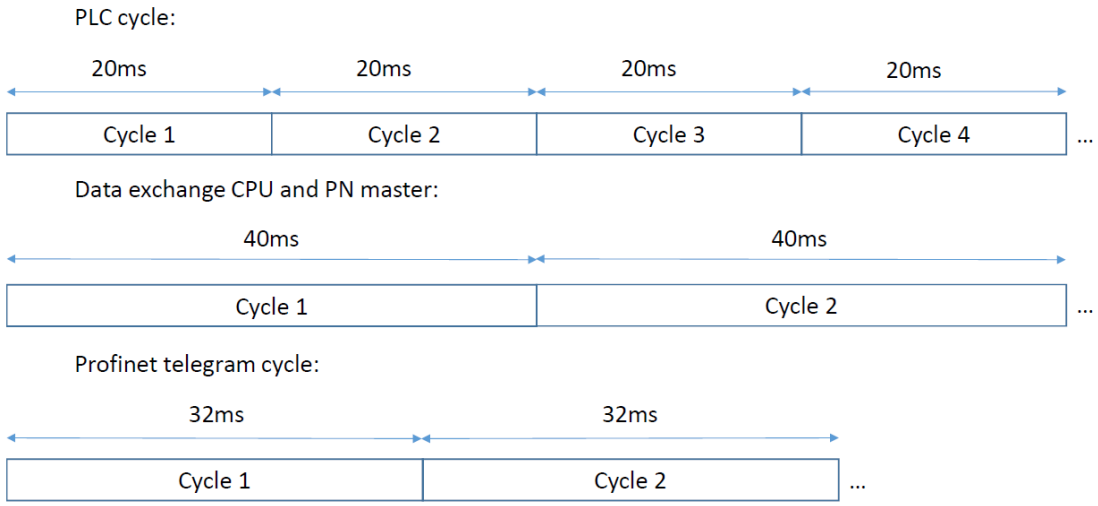
For a ring all times have to be balanced:

1. For a sufficient communication with Profinet slaves the combination of
   a. Send clock
   b. Reduction Ratio
   c. Phase

   has to be smaller than the PLC Task cycle time. Otherwise the data won't be send in one PLC task cycle.
2. The Min update time has to be smaller/equal than the half of the PLC cycle time (e.g. 20 ms PLC cycle → Min update time has to be smaller or equal to 10 ms).
3. The Watchdog time and the Datahold factor have to have the same value.
4. The value of watchdog and data hold factor have to be higher than the ring recovery of the managed switch (higher than 200 or 500 ms) to not get a communication fault in case of broken ring.

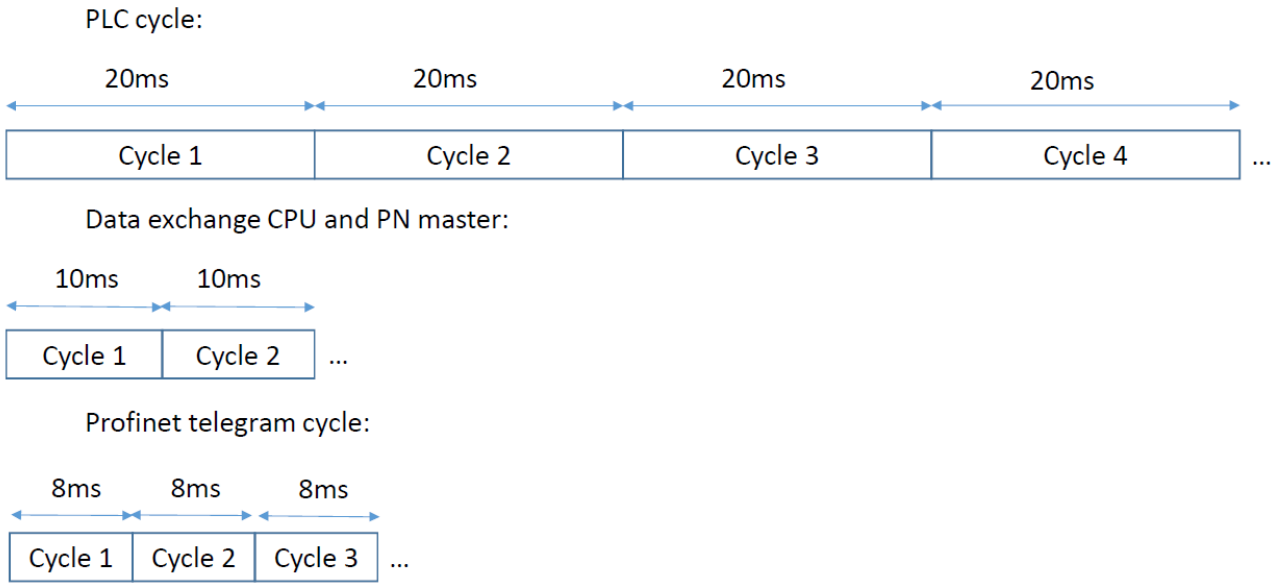
The following examples shall demonstrate how to set the values for PLC cycle time, send clock, reduction ratio and phase:

Example 1 (**WRONG!**): The PLC cycle time shall be cyclically 20ms. Send clock is set to 1ms and Reduction ratio is set to 32. An IO telegram will be sent every 32ms.




The PLC task is faster than the Profinet cycle time. There won't be new Profinet data available in the next PLC task.

Example 2(**CORRECT!**): The PLC cycle time shall be cyclically 20ms. Send clock is set to 1ms and Reduction ratio is set to 8. An IO telegram will be sent every 8ms.




The Profinet data will be always new when a new PLC cycle starts. Inside the task always the newest information will be available.

## Examples for setting a combination of ring recovery, data hold factor and watchdog

Example 1(**WRONG!**): Ring recovery is set to 200ms, data hold factor and watchdog are set to 32ms.

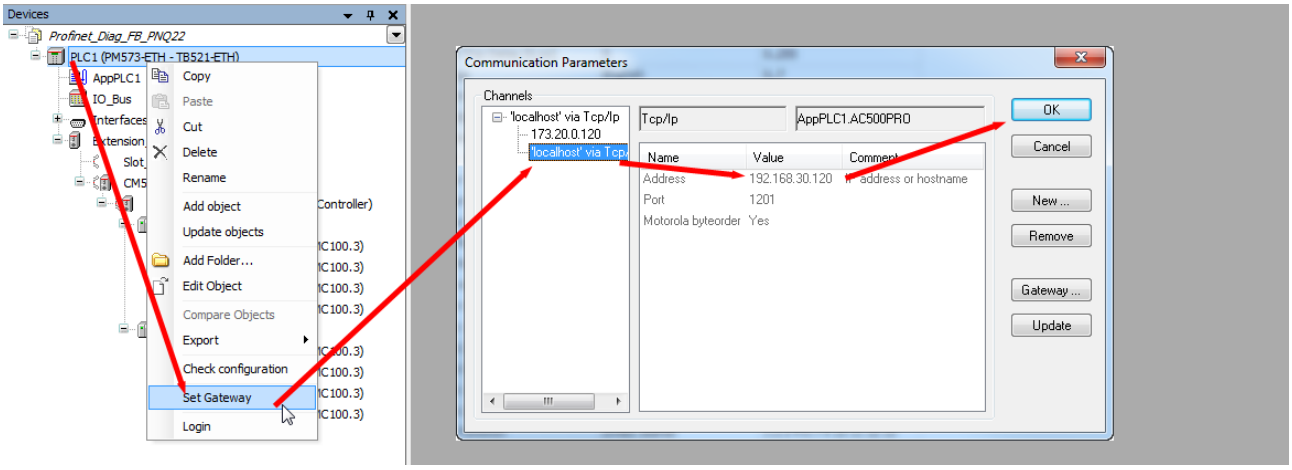
In this case the PNQ22.0 will indicate a missing master after the connection between two PNQs and/or between the forwarding port and the first PNQ will be lost. The test package of the switch hadn't recognized that the package is lost and the second ring port hasn't opened yet. After at least 200ms the switch will recognize that there is a broken wire in the ring and starts communicating through the second ring port. After that time the PNQ has lost its data and the connection between master and slave has to be established from new.

Example 2 (**CORRECT!**): Ring recovery is set to 200ms, data hold factor and watchdog are set to 208ms.

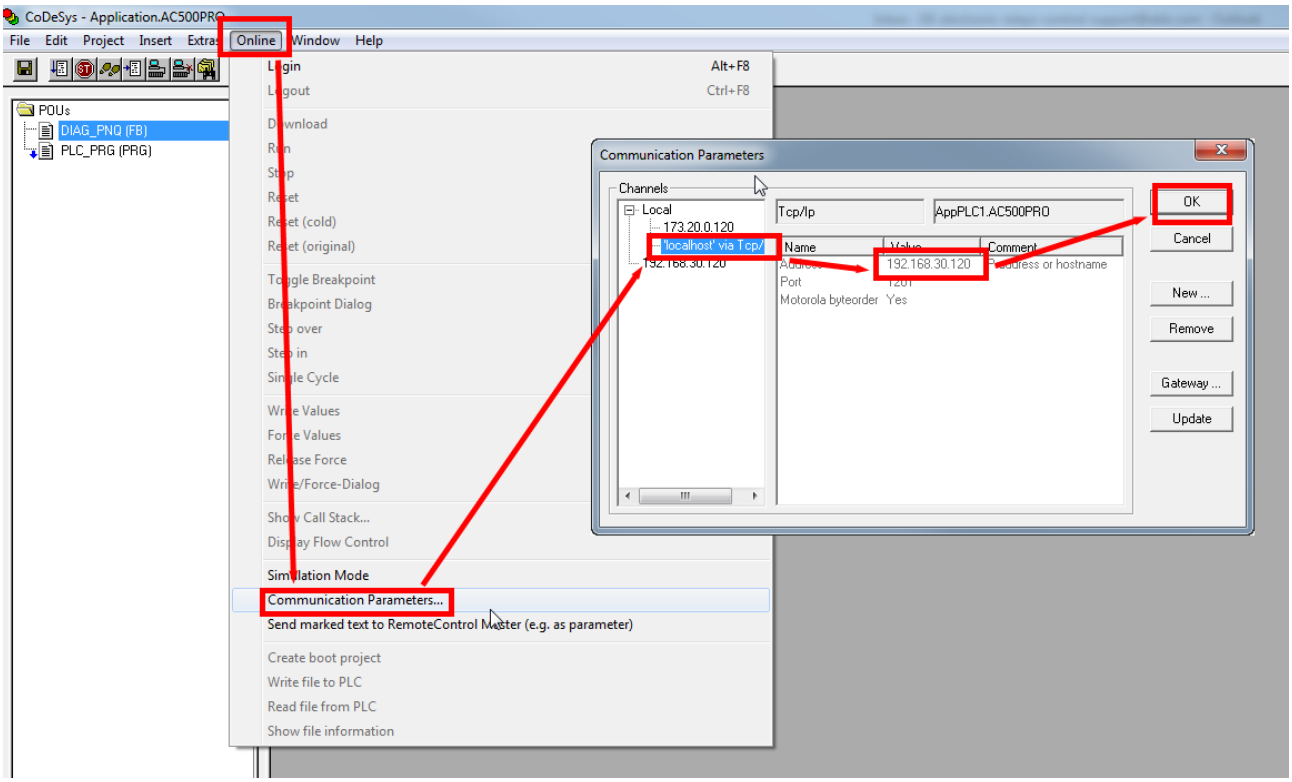
When with these settings the ring will be interrupted the test package will be handled as a lost package after 200ms. The data hold and watchdog times are higher that this value, so the PNQ22.0 won't lose the connection till the moment the second ring port starts communicating. The PNQ will act as nothing happened.

## Programming the application

1. Right-Click on the “PLC-AC500” in the project tree and choose “Set Gateway”
2. Select or add a channel.
3. Insert the IP address of the Ethernet port and apply with the “OK” button.




4. Double click on “Application” in the project tree. The editor will be opened.
5. Inside the editor go on “Online” → “Communication Parameters” and set the address of the PLC.




Inside this application all functions of the PLC can be programmed, for example the UMC100.3 control, PNQ22 Diagnostic and also the whole evaluation of the received data.

In the following chapter is described how to implement a ready-made function block “PNQ_ALARM_APP”, that uses PNIO_DEV_ALARM of the AC500 to get UMC diagnostics in the “Sequence of Events” (SoE) diagnostic format.

## PNQ22 Diagnostic with PNQ_ALARM_APP function block

Usually the diagnostic of Profinet slaves of an AC500 PLC are done with the special made PNIO_DEV_DIAG function block. For the ABB specific diagnosis model "Sequence of Events" (SoE) another function block, PNIO_DEV_ALARM, has to be used when the PNQ22.0 Profinet slave is connected to a Profinet interface of the AC500.

The block "PNQ_ALARM_APP" is specially made for Universal Motor Controllers (UMC) connected to the PNQ22.0 as a Subslave.

For a better overview directly in the application, all active alarms will be stored in arrays, one for each PNQ22 port. Each connected UMC has its own alarm list which will be updated every time a new alarm comes in.

For each connected PNQ22.0 one instance of this block has to be created in the AC500 Application.

Attached to this document there is an export file which can be simply imported to an existing AC500 project and also a whole project file as example. Please be aware that the variable declaration from chapter "UMC100.3" step 5 is used in the visualization!

## Variable declaration

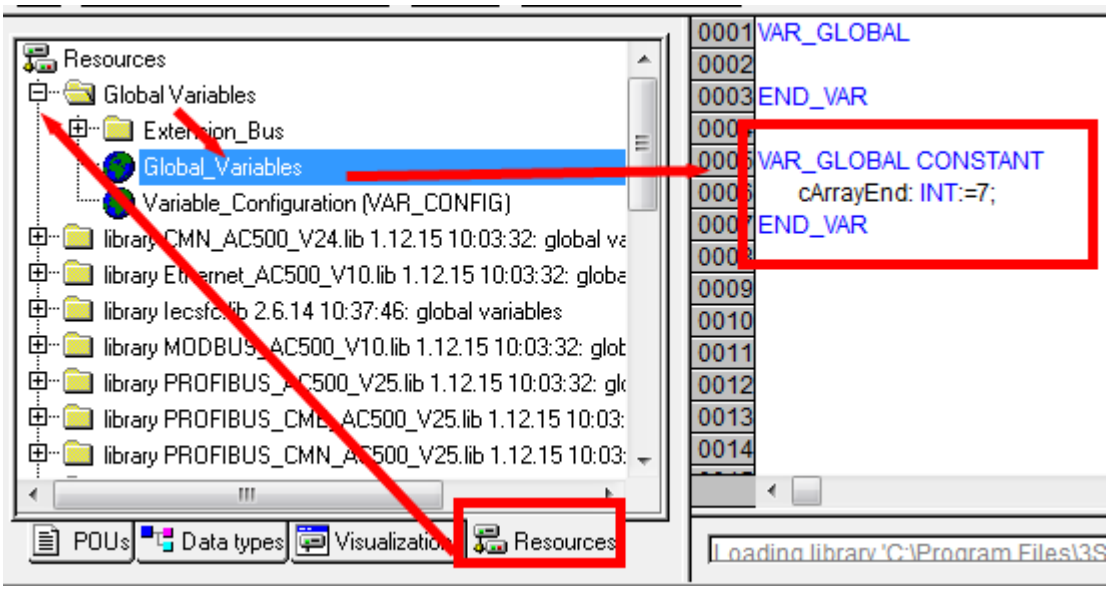
To use the "PNQ_ALARM_APP" block it' necessary to create at least the following 5 variables where you want to call the function block:

1. Instance "fbPNQ22_1" of the function block
2. An Array of string "asActiveAlarmPNQ1UMC1" for port 1 of PNQ22 with upper bound "cArrayEnd"
3. An Array of string "asActiveAlarmPNQ1UMC2" for port 2 of PNQ22 with upper bound "cArrayEnd"
4. An Array of string "asActiveAlarmPNQ1UMC3" for port 3 of PNQ22 with upper bound "cArrayEnd"
5. An Array of string "asActiveAlarmPNQ1UMC4" for port 4 of PNQ22 with upper bound "cArrayEnd"

```
0001 PROGRAM PLC_PRG
0002 VAR
0003     fbPNQ22_1: PNQ_ALARM_APP;                          (*Instance  of PNQ_ALARM_APP*)
0004     asActiveAlarmPNQ1UMC1: ARRAY[1..cArrayEnd] OF STRING;   (*Array where the alarms of PNQ22 port 1 shall be stored.*)
0005     asActiveAlarmPNQ1UMC2: ARRAY[1..cArrayEnd] OF STRING;   (*Array where the alarms of PNQ22 port 2 shall be stored.*)
0006     asActiveAlarmPNQ1UMC3: ARRAY[1..cArrayEnd] OF STRING;   (*Array where the alarms of PNQ22 port 3 shall be stored.*)
0007     asActiveAlarmPNQ1UMC4: ARRAY[1..cArrayEnd] OF STRING;   (*Array where the alarms of PNQ22 port 4 shall be stored.*)
0008
```

The constant "cArrayEnd" in the declaration of the arrays, has to be defined in the global variable list:

1. Open the "Resources" ribbon below the project tree.
2. Expand the folder "Global Variables"
3. If there is no special list for constants (e.g. "Global_Variables (CONSTANT)") open another, existing "Global_Variables" list and insert the following lines:



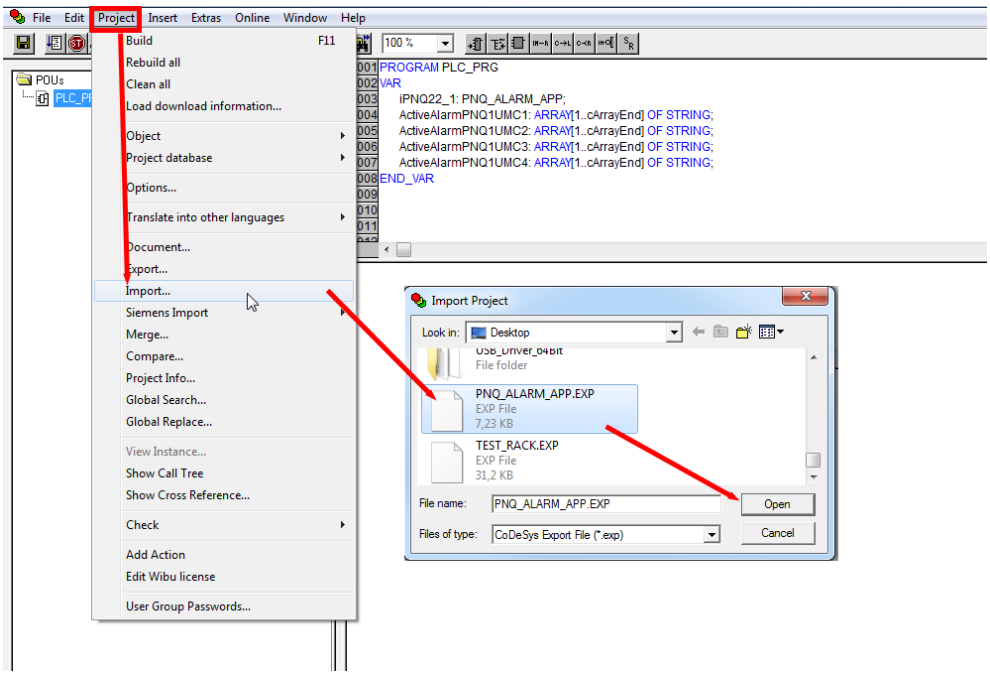

The variable “cArrayEnd” has to be declared as a constant, otherwise it won’t be possible to use it as upper bound for an array. The constant is set to the value “7” as standard which should be enough to save all active alarms, but it can be defined as necessary.

ATTENTION!
The constant “cArrayEnd” will be used in the code inside the function block!
It’s important to define it exactly like shown above.
If it’s necessary to get an array with more than 7 entries just change the constant to the wished number.

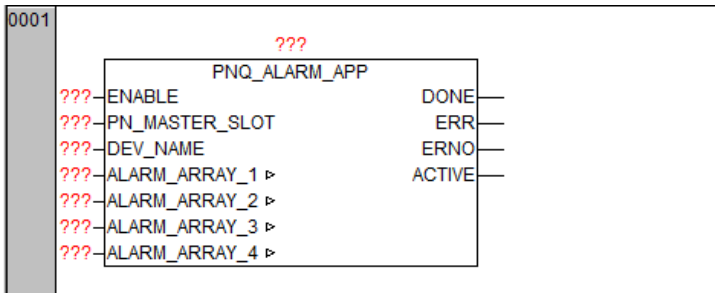
## Insert the function block in the program

Before using the function block it has to be imported to the project.

1. Open the ribbon “Project” and click on “Import…”.
2. Look for the “PNQ_ALARM_APP.EXP” file and open it.

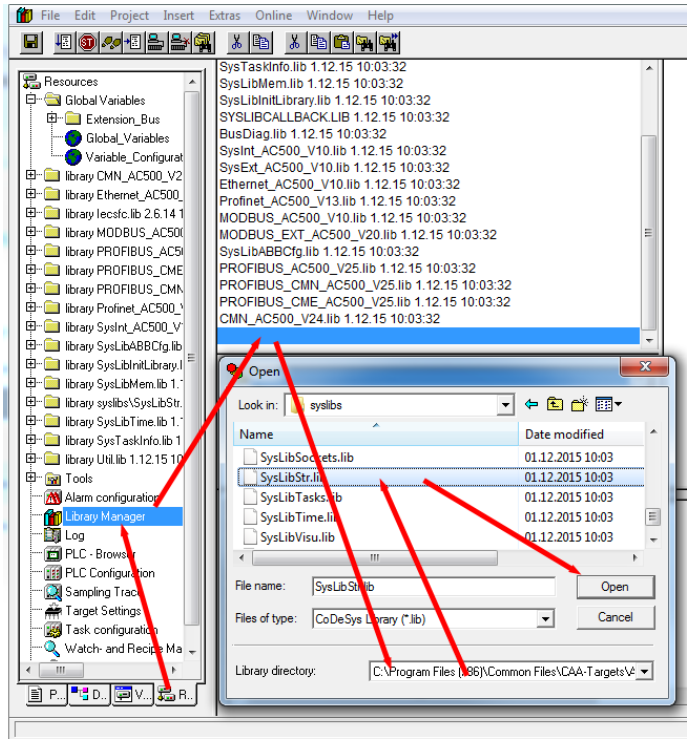


3. Go back to the program / function block where you want to insert the function block.
4. Insert a box in a network and type in “PNQ_ALARM_APP”.




## Add “SysLibStr.lib” to the library

1. Open the “Resources” ribbon below the POUs.
2. Double click on “Library manager”.
3. Right click in the part of the window, where the used libraries are shown and click on “Additional Libraries …”.
4. Go to the directory “C:\Program Files (x86)\Common Files\CAA-Targets\ABB_AC500\AC500_V12\library\syslibs” and look for “SysLibStr.lib” and open it.

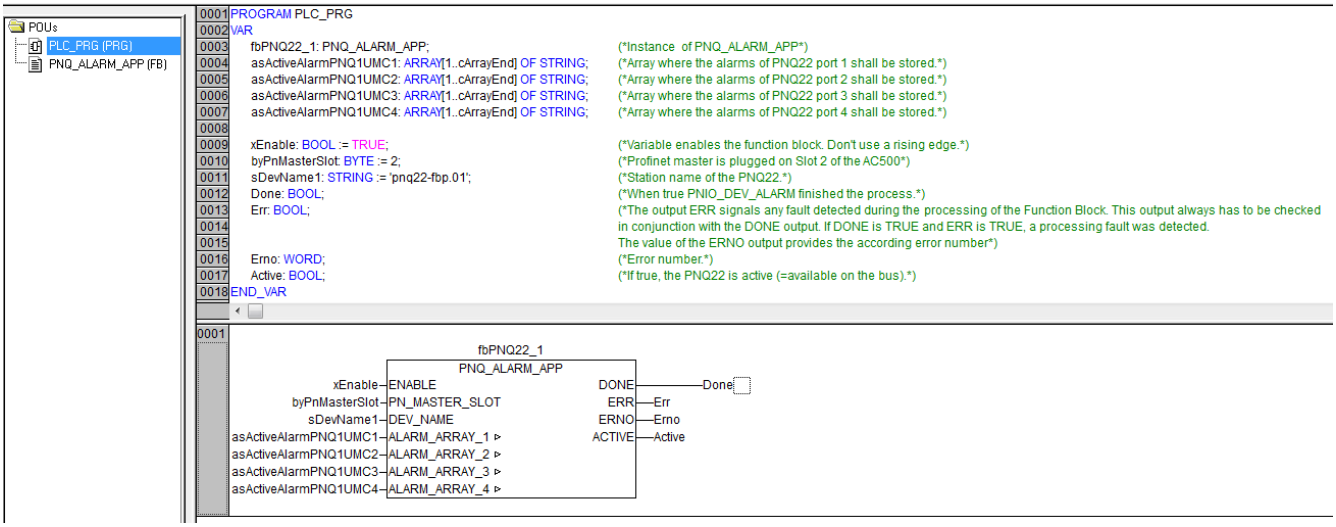


### In- and outputs of “PNQ_ALARM_APP”

Mark the “???” above the block and insert the instance you created before. A description of the in- and outputs can be found in the following table:

| Instance | | Data type | Description |
|---|---|---|---|
| ENABLE | Input | BOOL | When this input is enabled the function block is working. |
| PN_MASTER_SLOT | Input | BYTE | Input SLOT selects the Communication Module serving the PROFINET® IO device described by DEV_NAME. Valid values are 1...4, counting from right to left, starting with 1 as the first Communication Module left to the CPU. For PM595 Processor Modules, internal PROFINET Communication Module with connector ETH3 is slot 5, ETH4 is slot 6. |
| DEV_NAME | Input | STRING | Enter here the station name from the PNQ22. |
| ALARM_ARRAY_1 | Input | Array of STRING | Insert here the ALARM_ARRAY_1 |
| ALARM_ARRAY _2 | Input | Array of STRING | Insert here the ALARM_ARRAY _2. |
| ALARM_ARRAY _3 | Input | Array of STRING | Insert here the ALARM_ARRAY _3. |
| ALARM_ARRAY _4 | Input | Array of STRING | Insert here the ALARM_ARRAY _4. |
| DONE | Output | BOOL | When true PNIO_DEV_ALARM finished the process. |
| ERR | Output | BOOL | The output ERR signals any fault detected during the processing of the Function Block. This output always has to be checked in conjunction with the DONE output. If DONE is TRUE and ERR is TRUE, a processing fault was detected. The value of the ERNO output provides the according error number |
| ERNO | Output | WORD | Error number. |
| ACTIVE | Output | BOOL | PNQ22 is active (=available on the bus). |

After connecting all in- and outputs (and also declaring additional variables) the window should look like this:



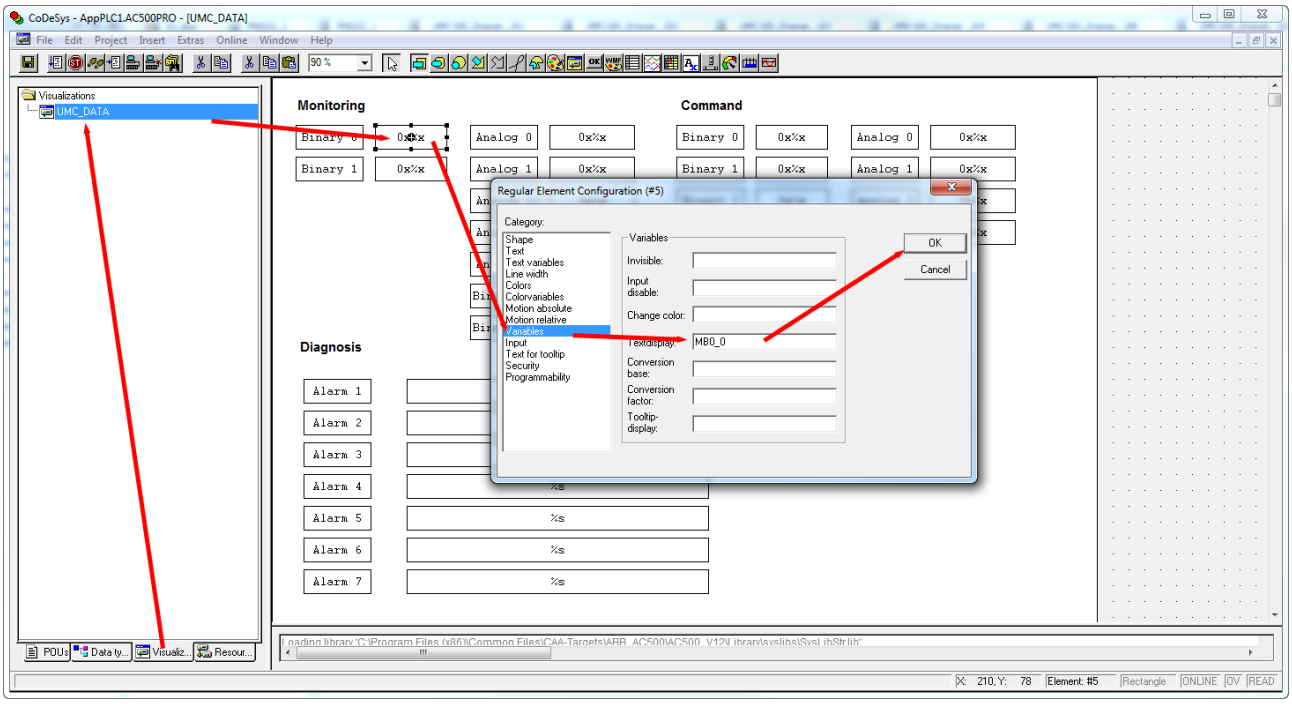

After fulfilling all above mentioned steps the block is ready to use.

For further PNQs several instances and also separate data arrays have to be created.

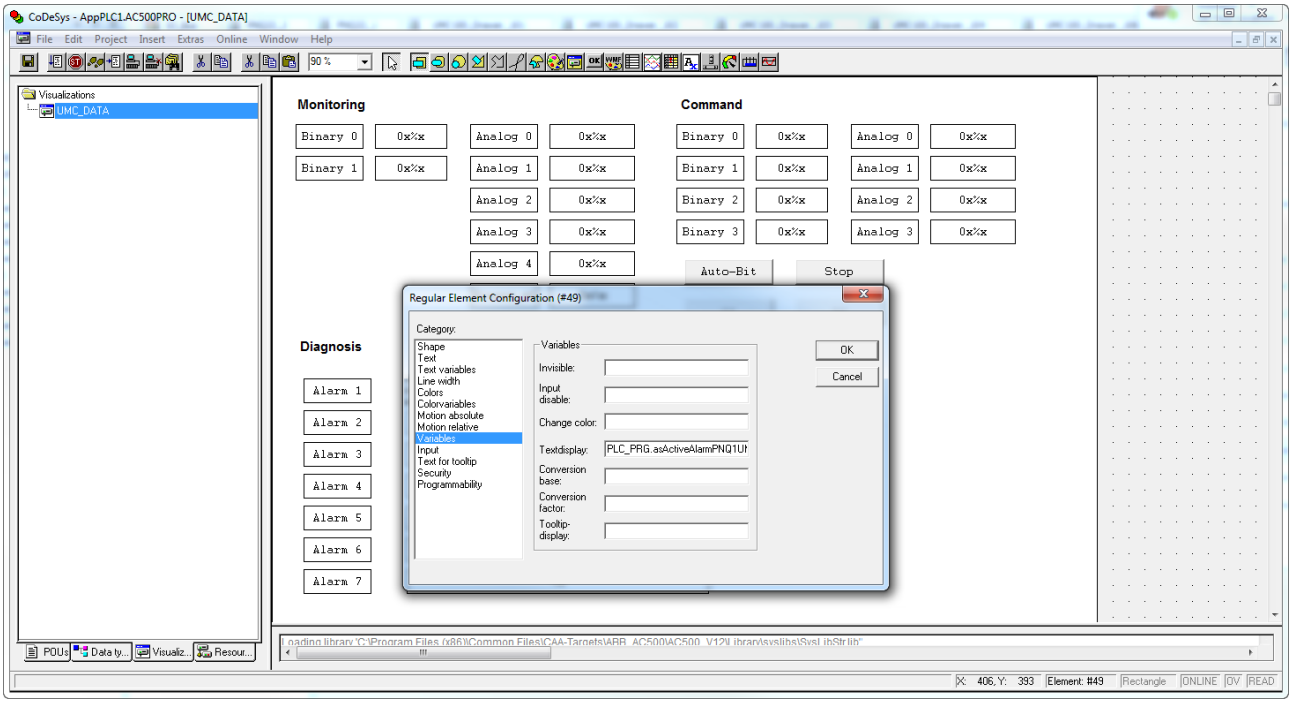
## Visualization for in- and output data

This chapter describes an example how to display all in- and output data and alarms for one UMC on a faceplate. This only shall demonstrate how to realize a visualization with the integrated functions. In the attached files a template for the first connected UMC with the variable names from chapter “UMC100.3” step 5 can be found. For changing the variables please follow these steps:

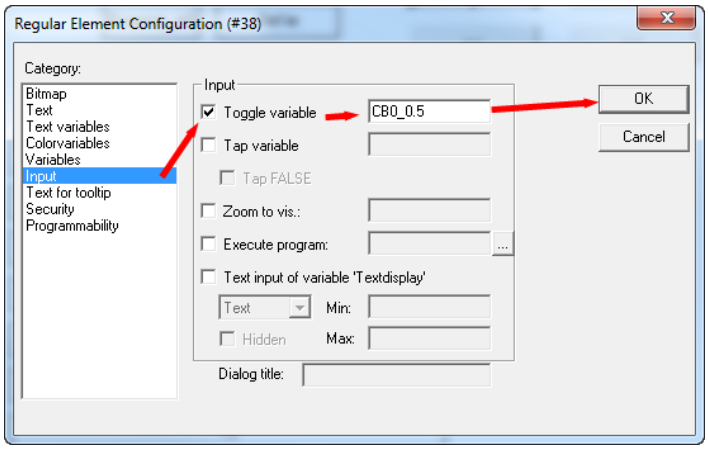
1. Open the “Visualization” ribbon and double click on “UMC_DATA” to open the faceplate.
2. Double click on a field saying “0x%x” (“0x” will be shown after going online as a prefix for hex values, “%x” will be replaced with the value of the variable).
3. To connect a variable to the field open “Variables” in the list and insert the global defined variable to “Textdisplay”.



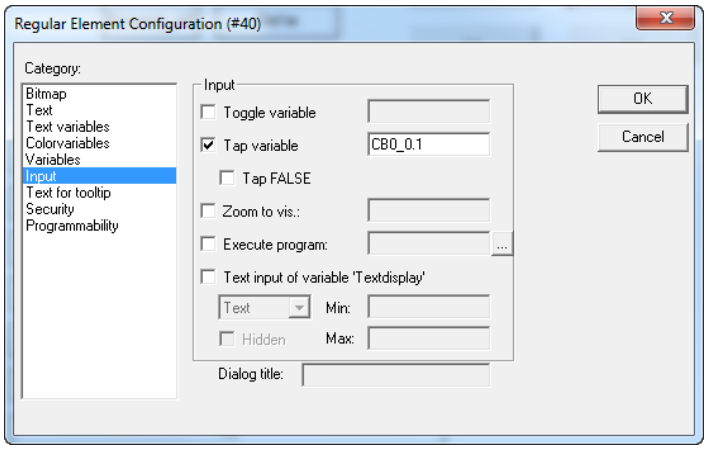

4. For changing the fields showing “%s” (“s” stands for string), double click on the field. In comparison to the “%x” fields the local variables from the PLC_PRG shall be used for showing the actual alarm list.
5. For getting the content of the array insert “PLC_PRG.” Before the actual array field (e.g. “PLC_PRG.asActiveAlarmPNQ1UMC1[1]”).



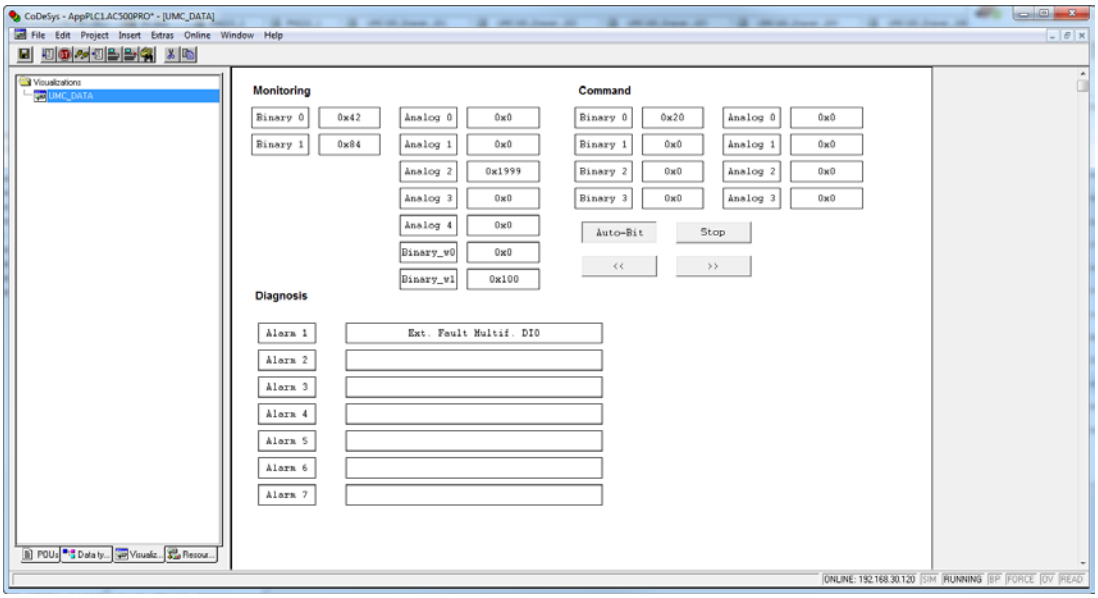

6. For controlling the UMC there are 4 pushbuttons on the template (reverse running only works when a reversing starter is configured. To connect the buttons with the command telegrams double click on each button.
    a. “Auto-Bit” should stay in the state after pushing once. Therefore bit 5 of command byte 0 shall be a toggle variable.




b. “Stop”, “<<” and “>>” are only pulses which will be send over the bus, therefore they are configured as “Tap variable”. “Stop” is bit 1 of command byte 0, “<<” is bit 0 of command byte 0 and “>>” is bit 2 of command byte 0.




7. After saving the project go online. Instead of “%x” and %s% there will be shown the actual values from the UMC and the UMC can be controlled by the pushbuttons of the template.



## Contact us

ABB STOTZ-KONTAKT GmbH
P. O. Box 10 16 80
69006 Heidelberg, Germany
Phone: +49 (0) 6221 7 01-0
Fax: +49 (0) 6221 7 01-13 25
E-mail: info.desto@de.abb.com

You can find the address of your local sales organization on the ABB home page
http://www.abb.com/contacts
-> Low Voltage Products and Systems

2CDC135043D0201