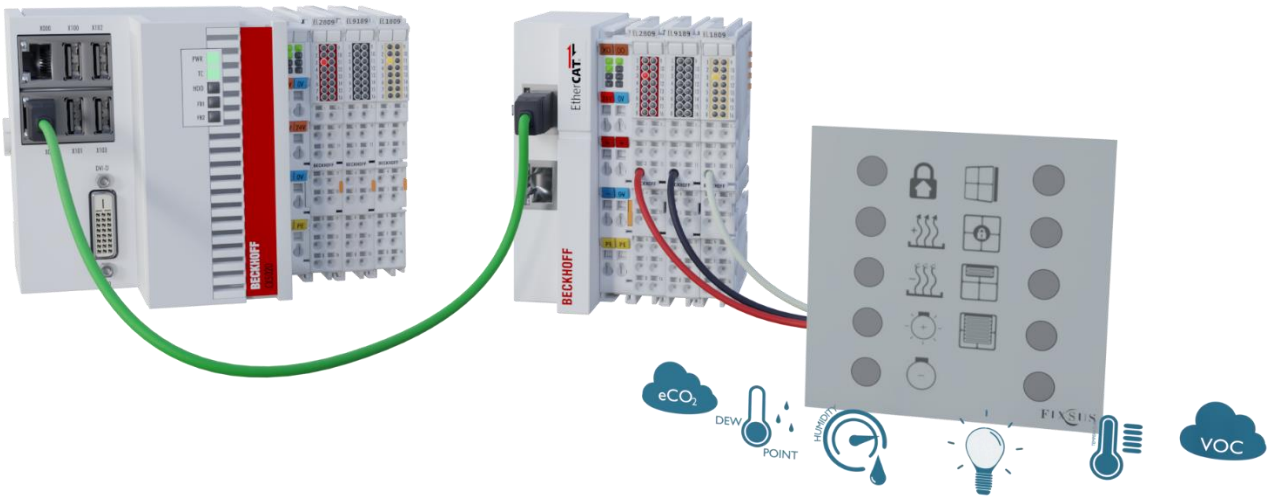
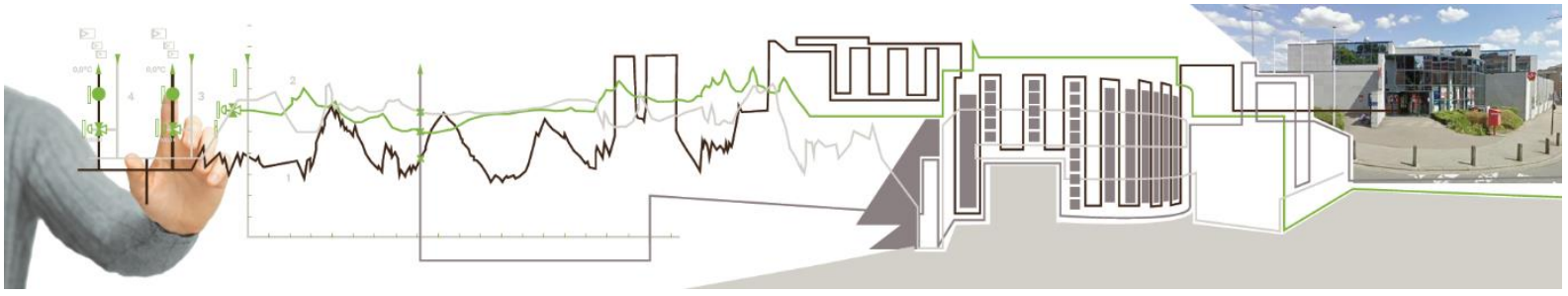
This document explains how to implement DIOC devices into a TwinCAT 2 project.

Manual TP10/RC/DIOC





1. Contents

2. Introduction

This manual is provided to help people implement the TP10 and RC into their own TwinCAT 2 projects. If required, you can visit our site, www.fixsus.be, or our forum, https://forum.fixsus.be, for additional info.

3. Short guide to implementing DIOC into TwinCAT 2

- Step 1: Use E-bus digital input and output terminals
- Step 2: Download the DIOC library 'DIOC_Library' and add it to the project. The latest version of the library can be found on the Fixsus forum, https://forum.fixsus.be
- Step 3: Change the cycle time to 12 ms
    - Method 1: change the standard cycle time to 12 ms and call the instances in MAIN
    - Method 2: make a new task with a cycle time of 12 ms and call the instances in the new task
- Step 4: Implement the new visualizations, if required
    - Implement the 'TP10' or 'TP10 mini' visualization for each TP10.
- Step 5: Change the system manager settings
    - Check if the in- and outputs of the DIOC devices are being called in the correct task
    - Enable the 'I/O at task begin for the linked PLC program

4. Detailed manual to implementing DIOC into TwinCAT 2

Step 1: Use E-bus digital input and output terminals

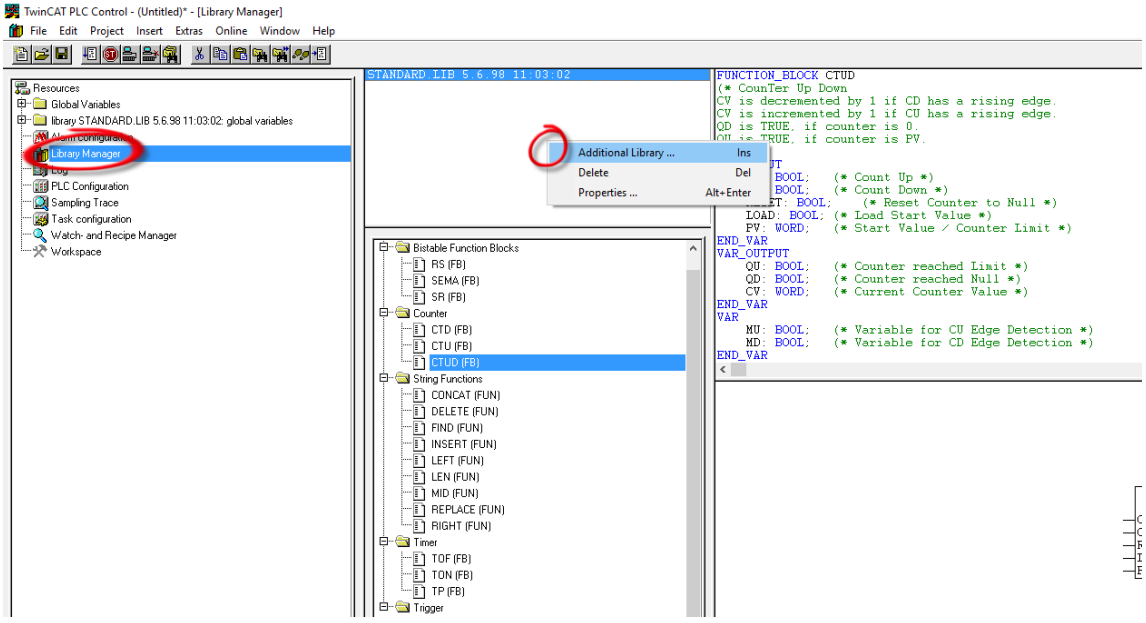
To implement the DIOC protocol, the E-bus must be used. To do this, E-bus digital input and output terminals must be used (e.g. EL1809, EL2809 or EL1859).

The DIOC protocol can not be used on the K-bus. If the amount of inputs/outputs of the K-bus is too large, the I/O cycle time will get an offset causing the DIOC protocol to not function properly. Therefore, **the K-bus is not officially supported**.

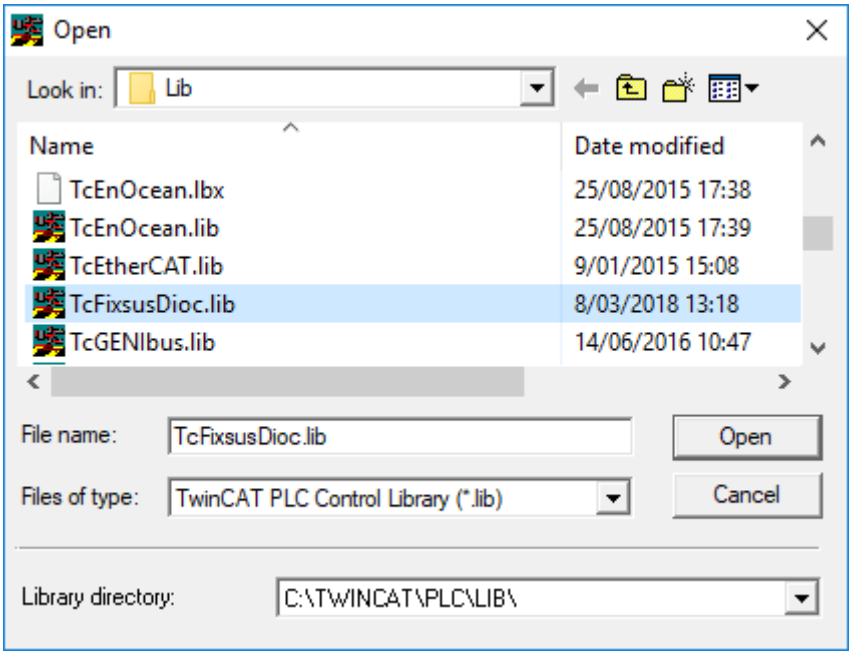
Step 2: implementation of library 'TcFixsusDiocLib'

The first step of the implementation is to load the necessary libraries. To do this you will have to download the library from our forum. (see https://forum.fixsus.be/topics/8-TP10---RA---Technician--Programming)

The library file 'TcFixsusDioc.lib' must be moved to your library directory (usually C:\TwinCAT\Plc\Lib). When the library is in your standard library directory, the library still has to be implemented in your program. This can be done by opening the library manager. This can be found under 'Resources'.



In the library manager a list can be found which contains all libraries and their content. The DIOC library can be added by right clicking the list of libraries and selecting 'additional library'. Navigate to the location of the library and open the library by selecting it and opening it. The library should be loaded now.


Open
Look in:
Lib
Name
Date modified
TcEnOcean.lbx
25/08/2015 17:38
TcEnOcean.lib
25/08/2015 17:39
TcEtherCAT.lib
9/01/2015 15:08
TcFixsusDioc.lib
8/03/2018 13:18
TcGENIbus.lib
14/06/2016 10:47
File name:
TcFixsusDioc.lib
Open
Files of type:
TwinCAT PLC Control Library (*.lib)
Cancel
Library directory:
C:\TWINCAT\PLC\LIB\

Step 3: Change the cycle time

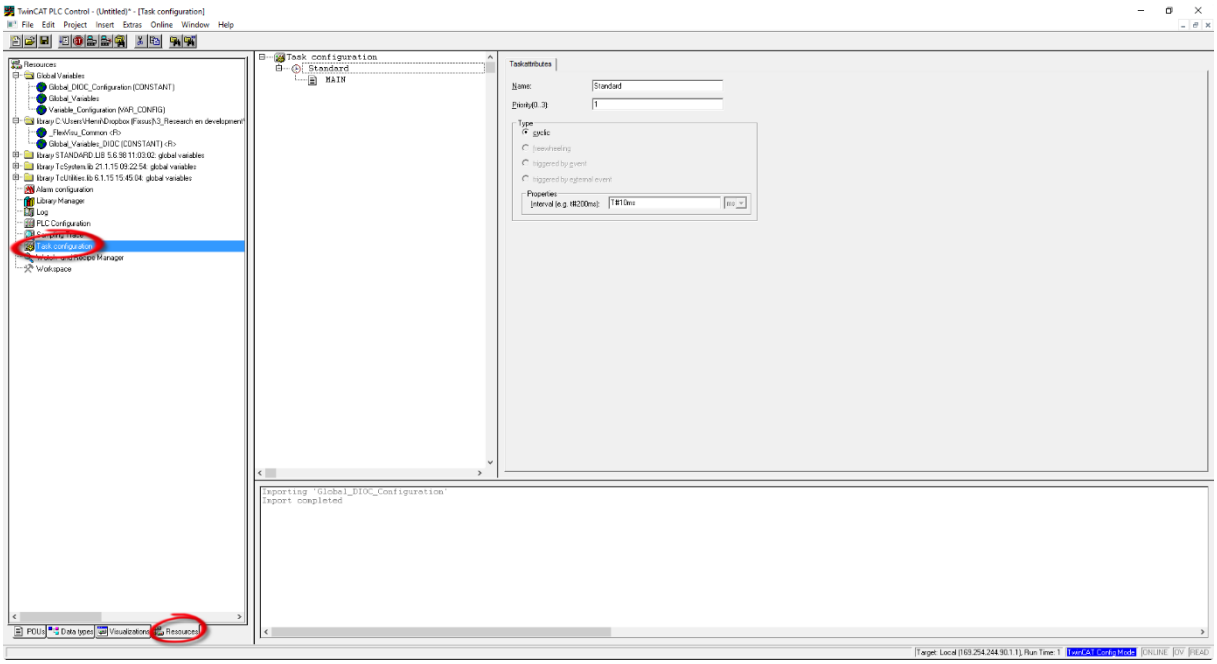
To ensure a good communication with the TP10, RC and other DIOC devices, instances of the TP10, RC and other DIOC function blocks must be called with a fixed cycle time. This cycle time is currently 12 ms. There are two methods to do this:

1. The standard cycle time can be set to 12 ms, the DIOC instances should then be called in the standard program (the MAIN program).
2. A new task can be made with a cycle time of 12 ms in which the DIOC instances can be called.

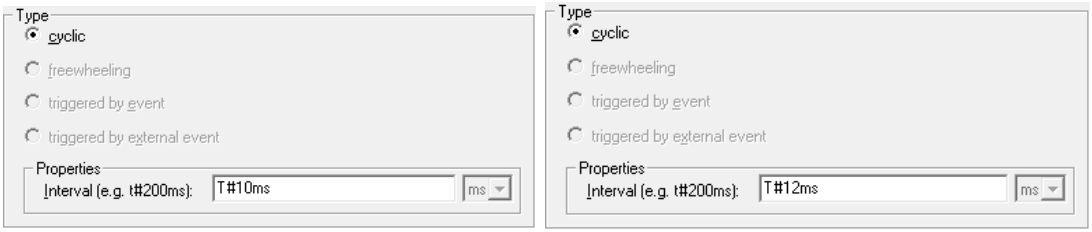
Method 1: Change the standard cycle time

This is the least complicated method. However, when other components of the program need to run on a different cycle time or when the whole program is too large to run on a cycle time of 12 ms, the second method should be used.

The cycle time can be changed under Resources/Task configuration.

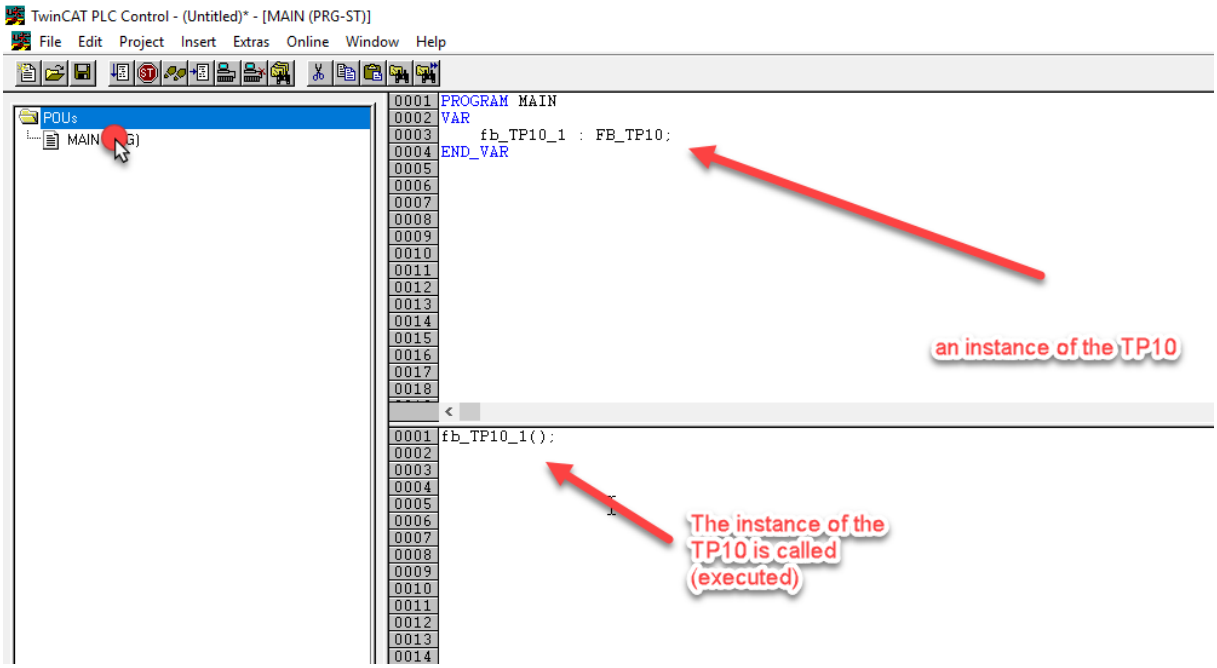


A list of all tasks will be displayed. Then the standard task can be changed to 12 ms:



The next step is to navigate to the MAIN program (or the equivalent if the MAIN program has been renamed) by clicking on POU and MAIN.

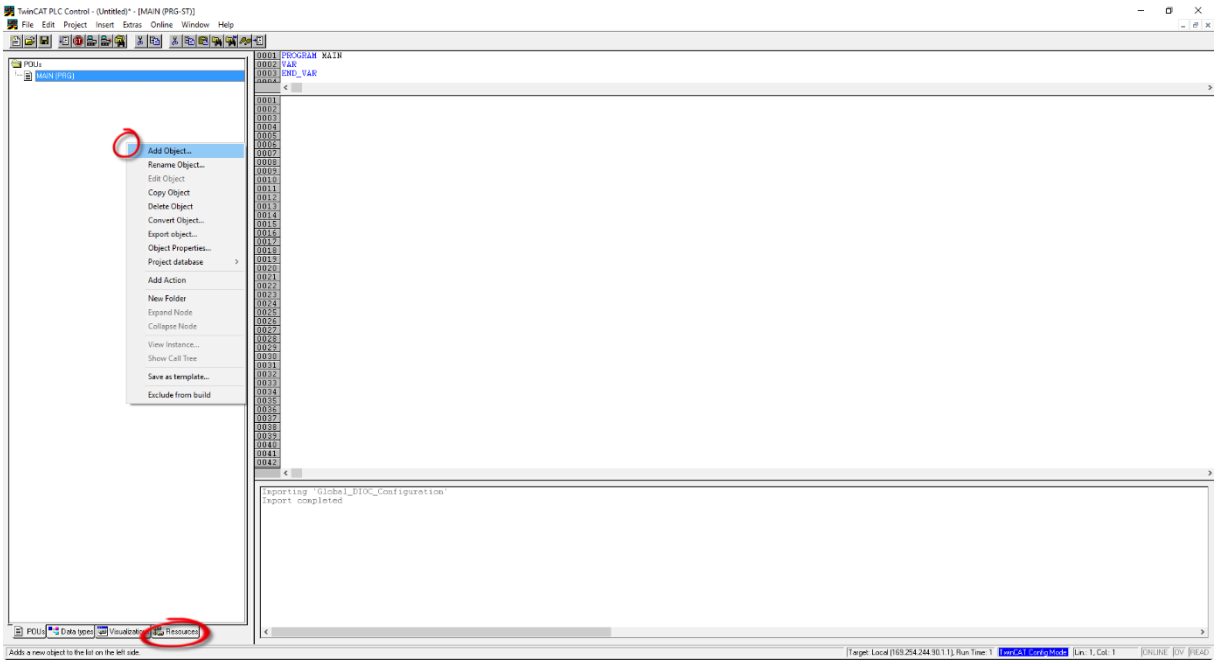
Now the DIOC instances must be called in the MAIN program. Make sure the instance is called every cycle to ensure a good communication.

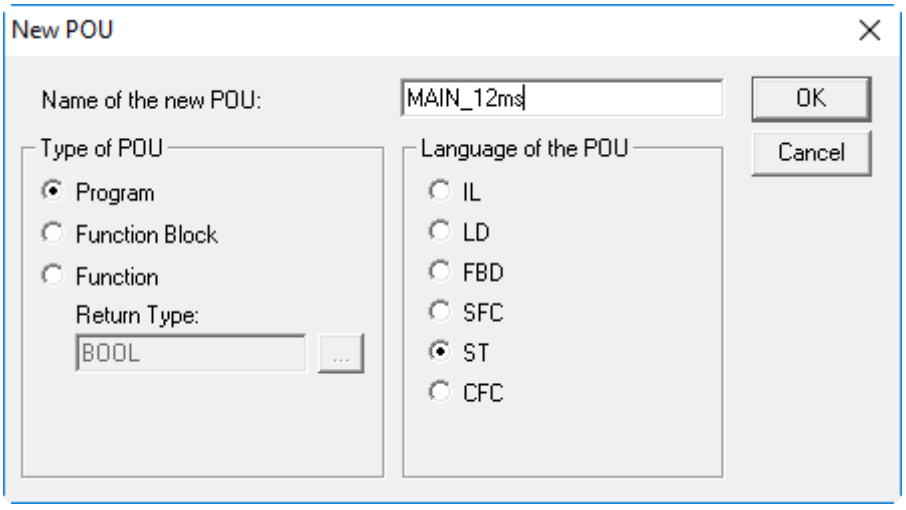

TwinCAT PLC Control - (Untitled)* - [MAIN (PRG-ST)]
File Edit Project Insert Extras Online Window Help
POUs
MAIN
PROGRAM MAIN
VAR
fb_TP10_1 : FB_TP10;
END_VAR
an instance of the TP10
fb_TP10_1();
The instance of the TP10 is called (executed)

Method 2: create a new task with a 12 ms cycle time

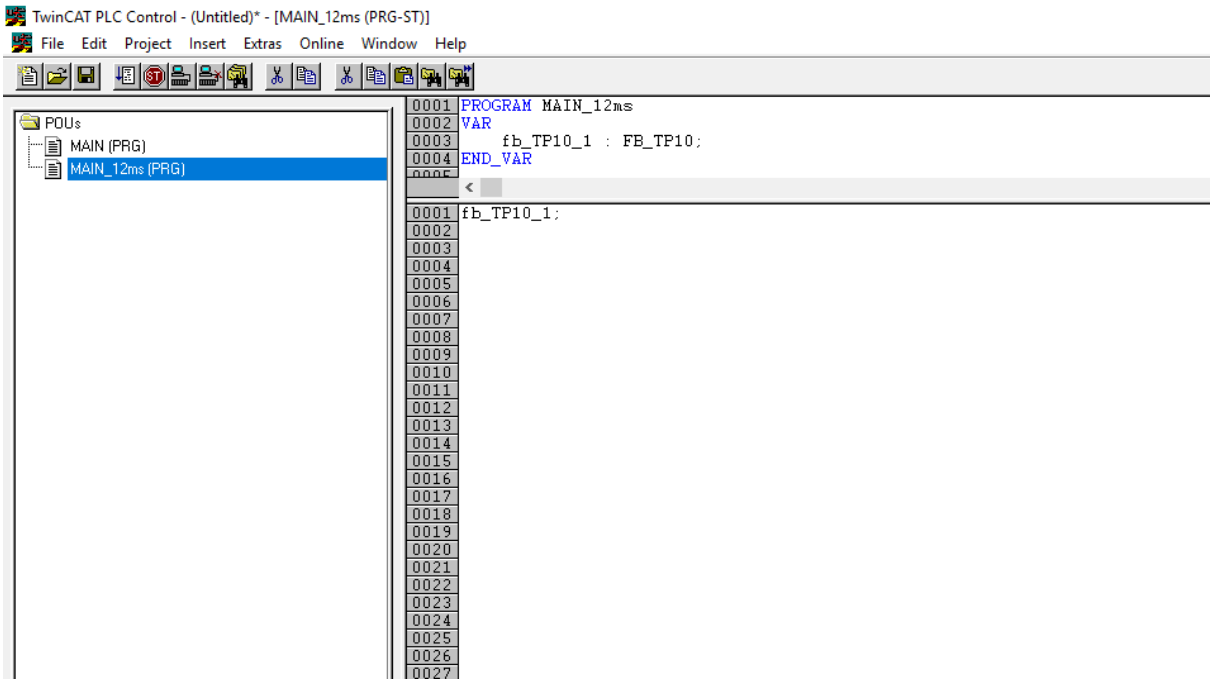
This method can only be used if there are less than 4 tasks in the project. If this is not the case, the cycle time of one of the other tasks needs to be changed to 12 ms and the DIOC program must be called in this task.

The first step is to make a new program that will be executed in the new task. Making a new program can be done by right-clicking under POU and adding a new object. The new program is named MAIN_12ms in the example.

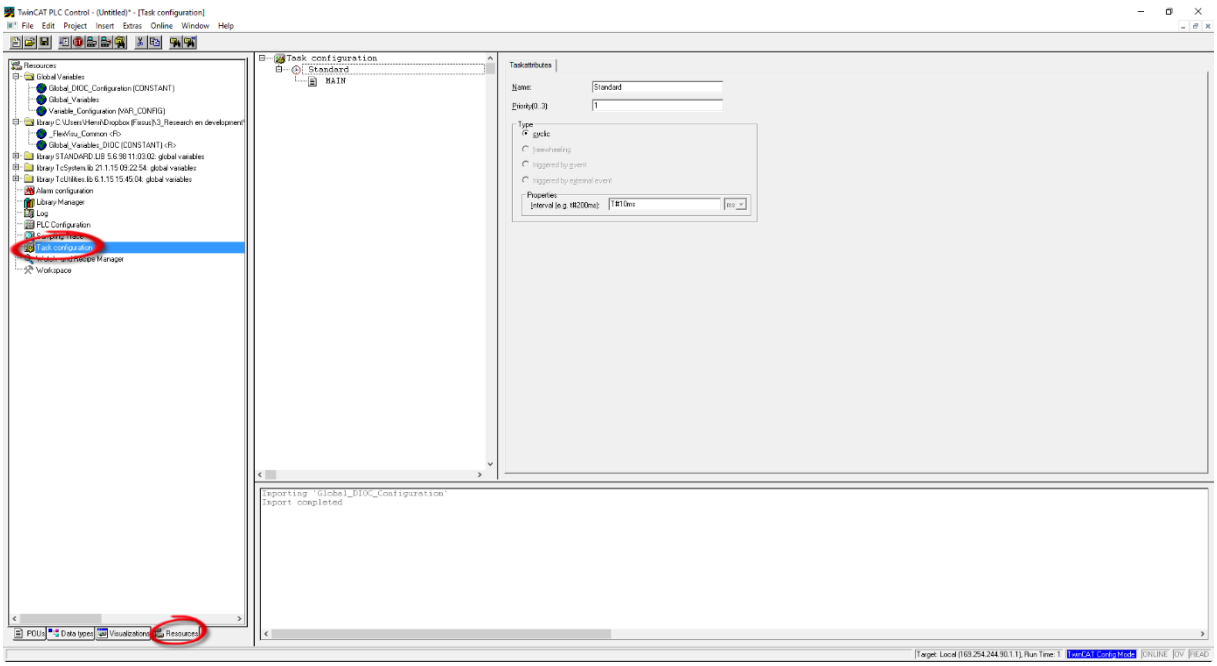




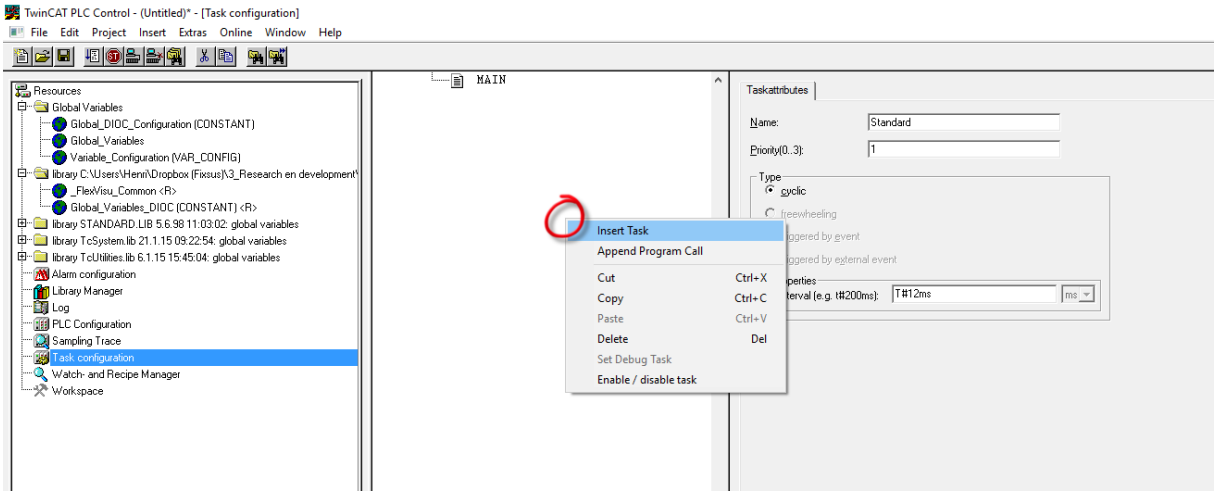
In this new program DIOC instances must be called, this means they will be executed.



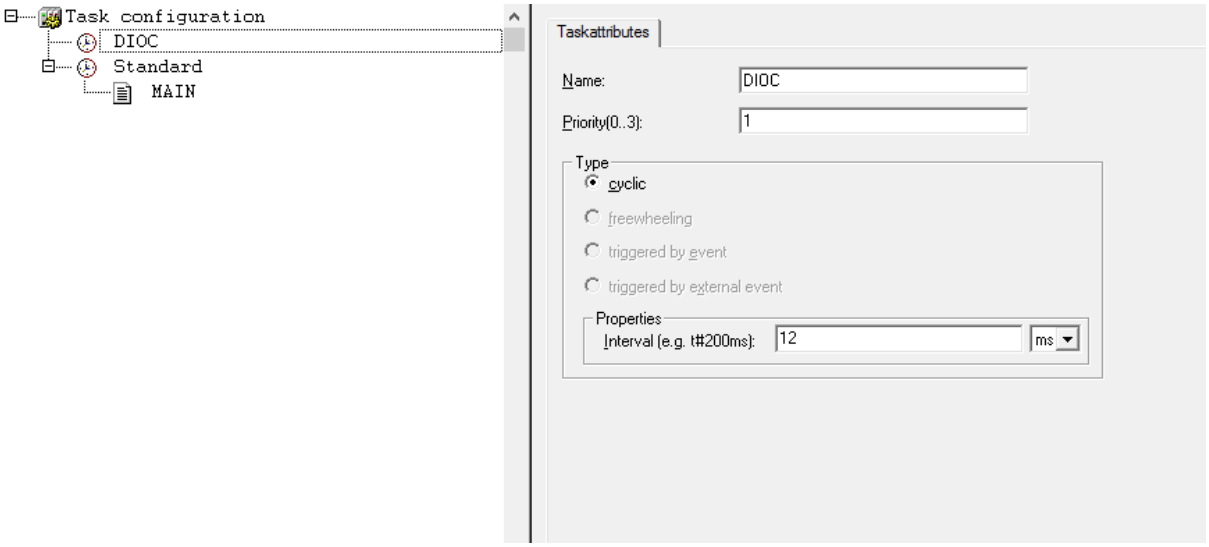
Now a new task will be created. This can be done by navigating to Resources/Task configuration.



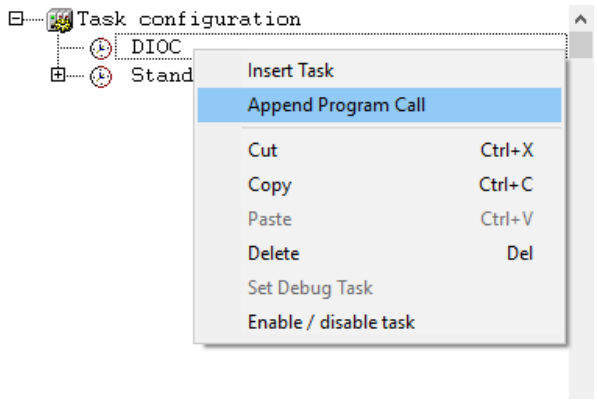
Right-click in the list of tasks and choose ‘insert Task’.



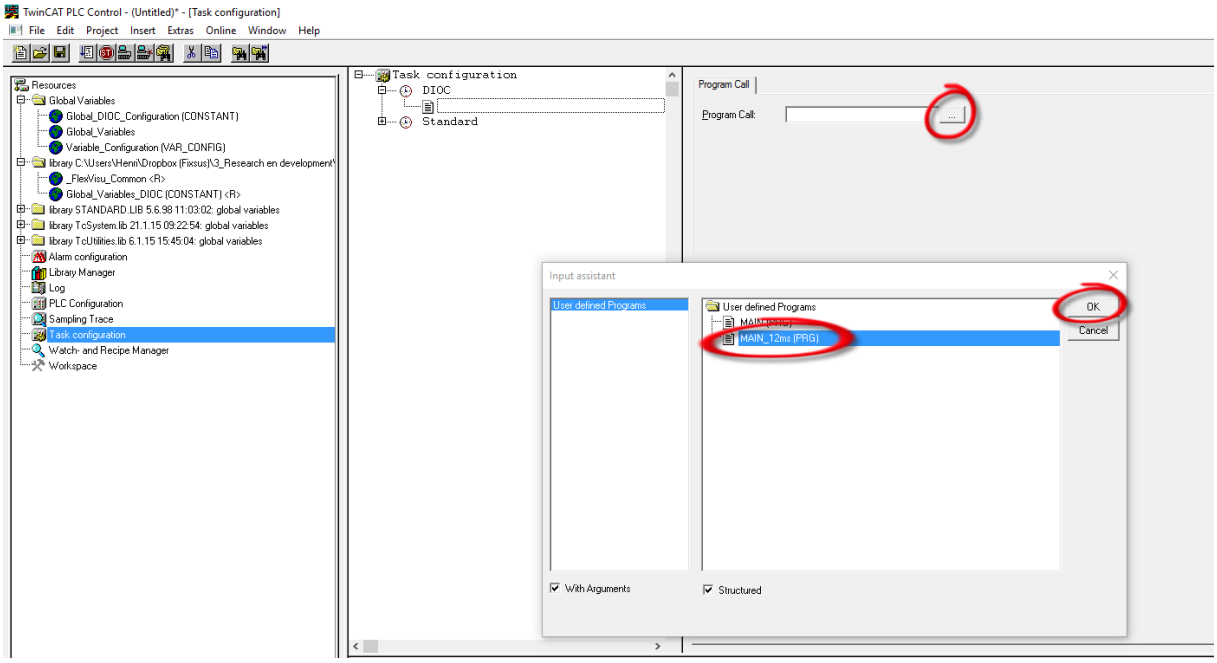
The task can be renamed by double clicking on the name of the new task. In this example the name 'DIOC' will be used. The cycle time of the new task must be changed to 12ms.



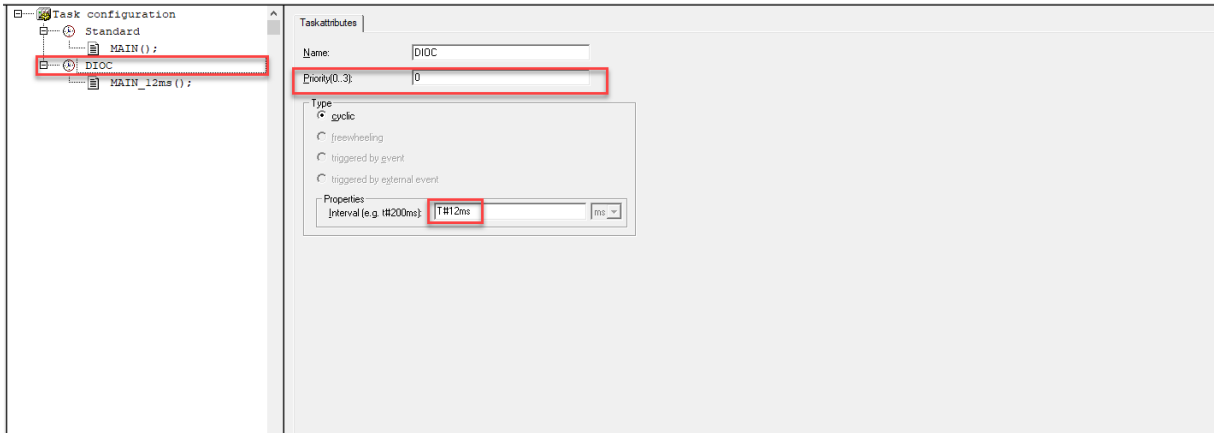
After the new task is made, the task needs to execute the new program. Right-click on the new task and select 'Append Program Call'.

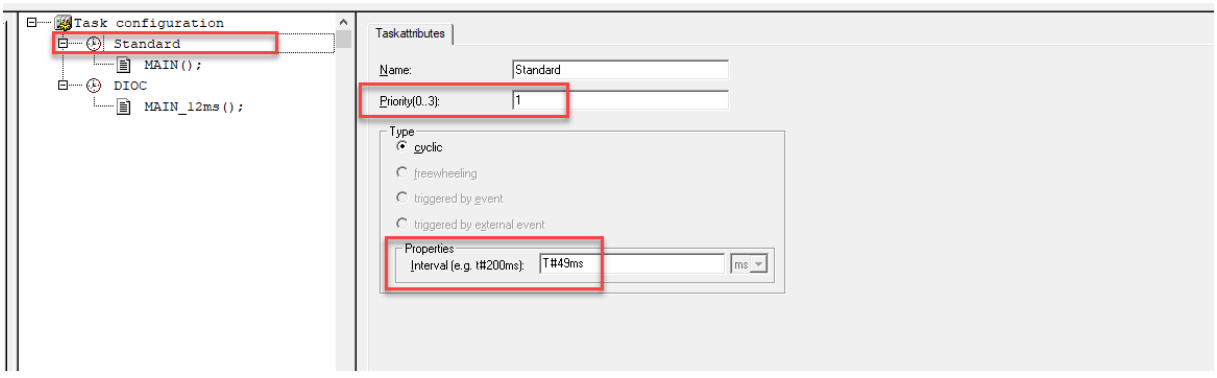


Select the program that must be executed by the task. Use the program that was created earlier which executes the Dioc program.



The priorities of the tasks should also be set in order. The task with the lowest cycle time should always get the lowest priority number (lowest priority number means highest priority).



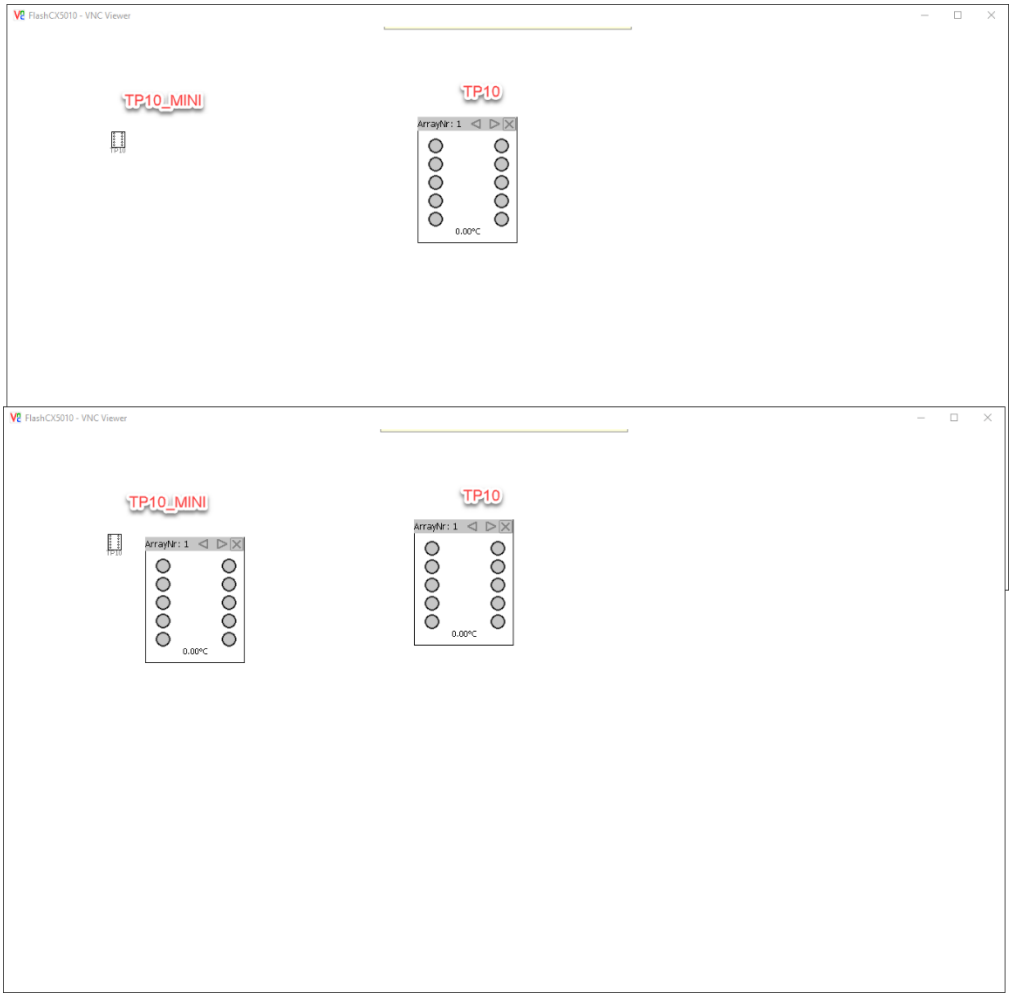


Step 4: Implementation of the visualisation

The DIOC_Library contains two visualisations to use with the TP10 and two visualisations to use with the RC. The visualisation 'TP10' or 'TP10_MINI' can be used for every TP10 separately and 'RC' or 'RC_MINI' can be used for every RC separately.
Below is a description on how to implement the TP10 visualisation. The RC can be visualised in the same way.

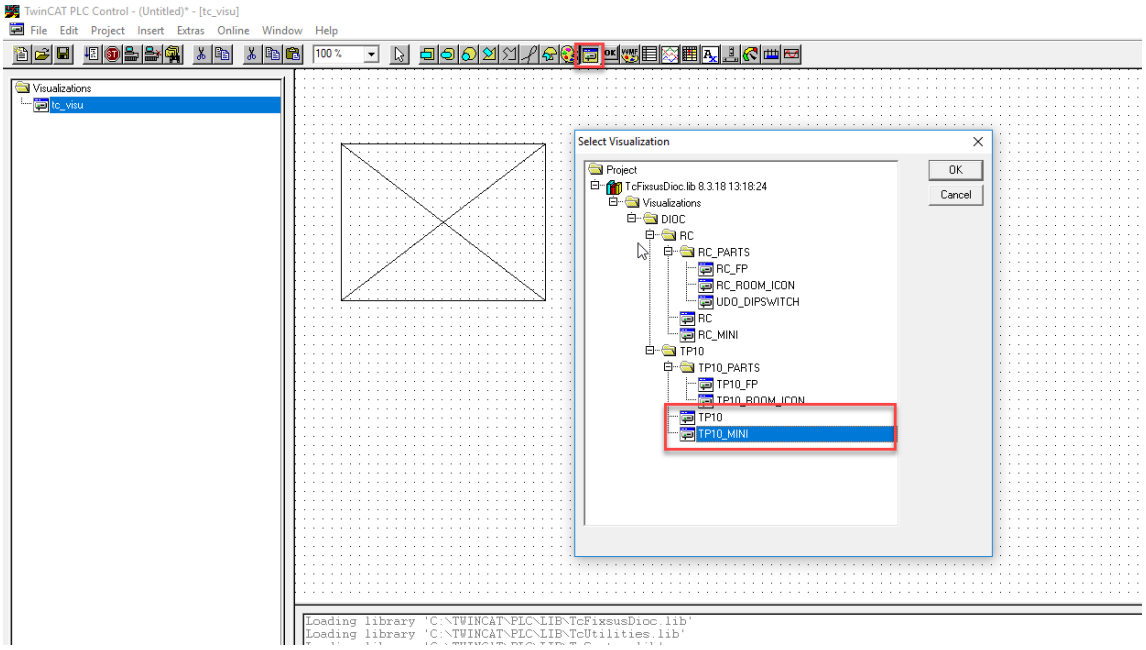
TP10 visualisation

Every TP10 can get its own visualisation where the status of the buttons and the measurements can be read. For a complete explanation of the possibilities, see chapter 'Variables of the TP10'.
There are two possible visualisations that can be used. The 'TP10' visualisation shows the whole TP10, while the 'TP10_MINI' is a small button with which the full visualisation can be opened.



Both visualisations can be added in the same way. As an example, a 'TP10' visualisation is added.
Open the visualisation screen in which the TP10 visualisation will be used. Add a visualisation:

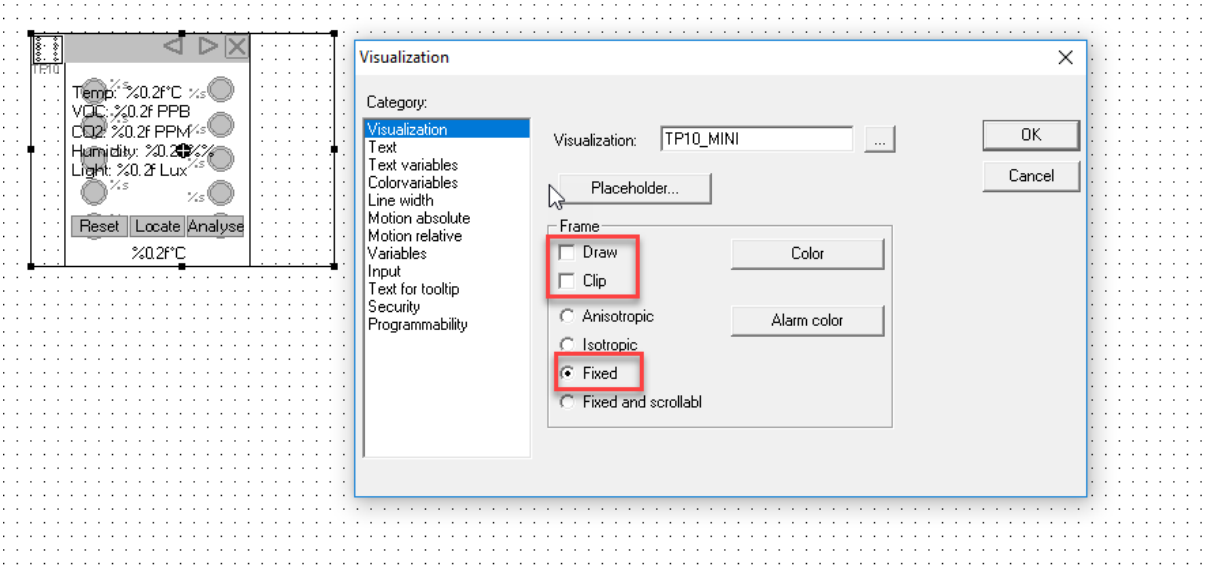
Select ‘TP10’ or ‘TP10_MINI’ and click on ‘OK’.



Double click on the new visualisation and configure the visualisation under ‘visualization’ with the configurations below:

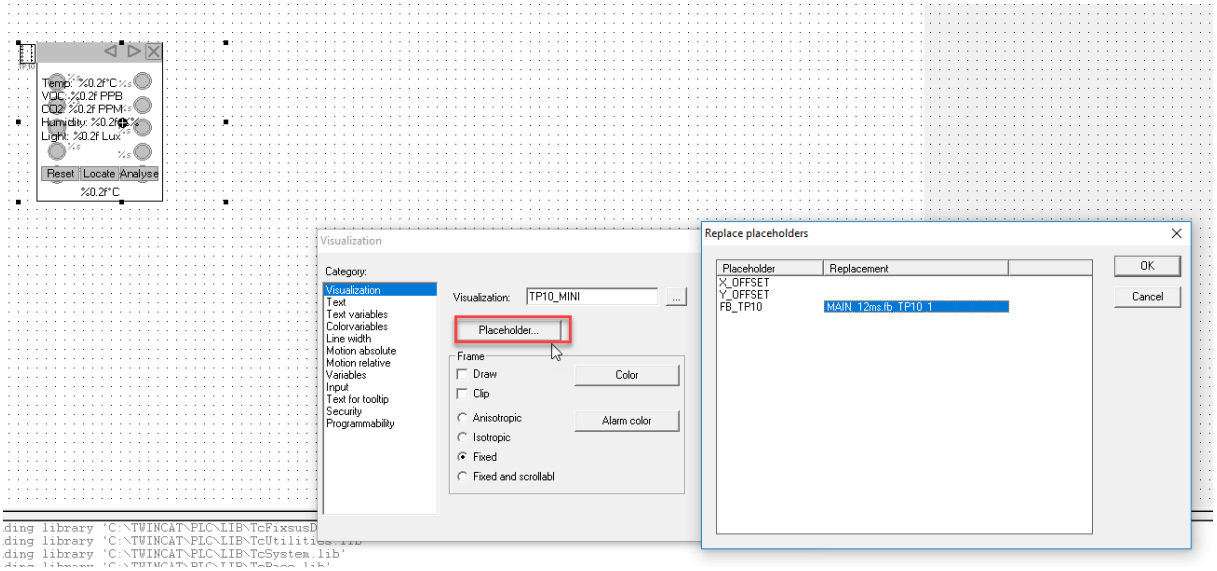
- ‘Draw’ and ‘Clip’ should be off.
- ‘Fixed’ should be on

This configuration makes sure the size and ratios are correct.



To make sure the TP10 visualisation works as expected, the correct links must be made. This can be done in the menu of the visualisation. Select ‘Placeholder’ in the ‘Visualization’ menu. In this menu the next configurations can be done:

- FB_TP10 : The location of the instance of the TP10 in the program.
- X_OFFSET and Y_OFFSET: only used in the TP10_MINI. With these placeholders the TP10 can be moved relative to the button to open the TP10 visualisation.

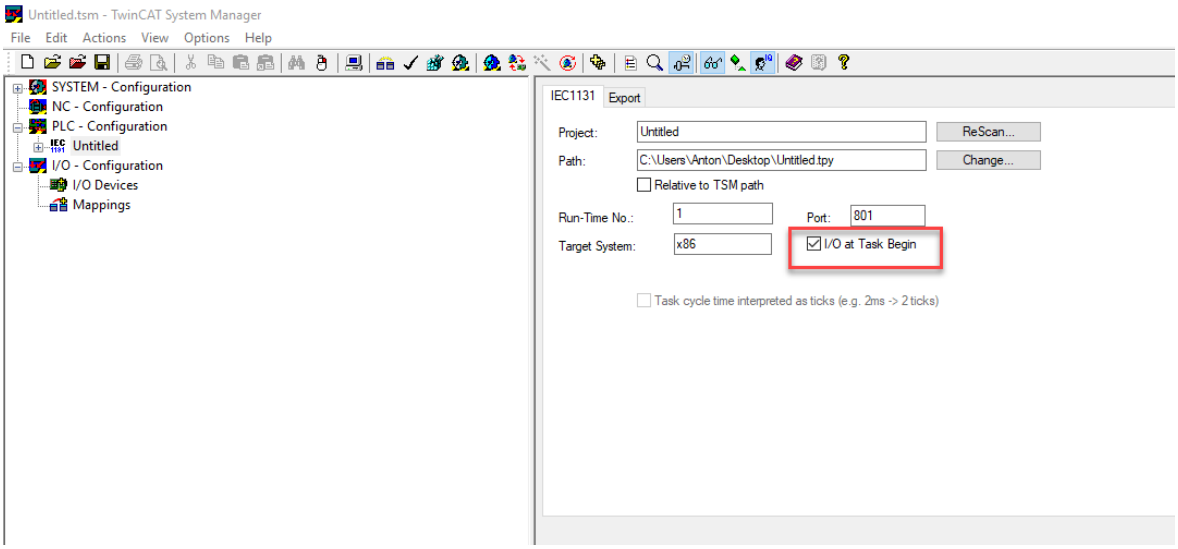

Temp: %0.2f°C
VOC: %0.2f PPB
CO2: %0.2f PPM
Humidity: %0.2f%
Light: %0.2f Lux
Reset
Locate
Analyse
%0.2f°C
Visualization
Category:
Visualization
Text
Text variables
Colorvariables
Line width
Motion absolute
Motion relative
Variables
Input
Text for tooltip
Security
Programmability
Visualization:
TP10_MINI
Placeholder...
Frame
Draw
Clip
Color
Anisotropic
Isotropic
Fixed
Fixed and scrollabl
Alarm color
Replace placeholders
Placeholder
Replacement
X_OFFSET
Y_OFFSET
FB_TP10
MAIN_12ms.fb_TP10_1
OK
Cancel

Step 5: Changing the configurations on the system manager

When using the TwinCAT system manager there are two things that have to be checked to implement the TP10, RC and other DIOC devices:

I/O at task begin

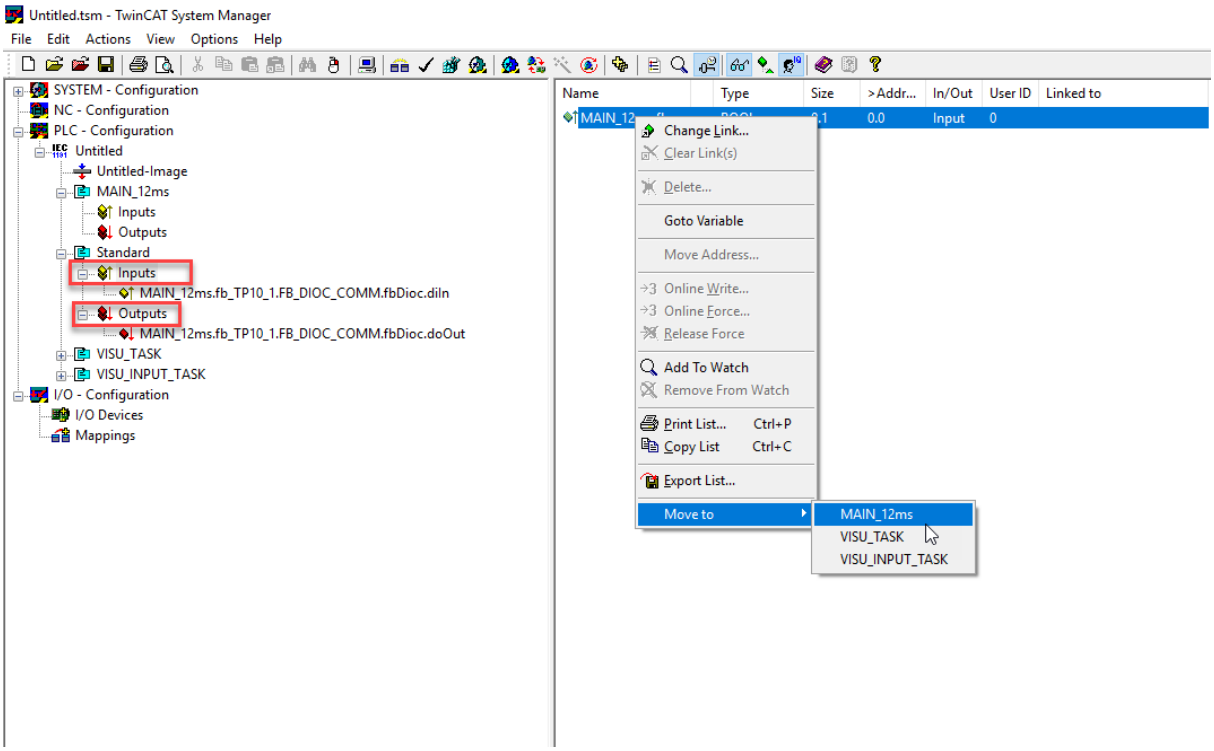
Under 'PLC-Configuration/PLC program' the option 'I/0 at task begin' needs to be checked to ensure a good communication with the DIOC device.



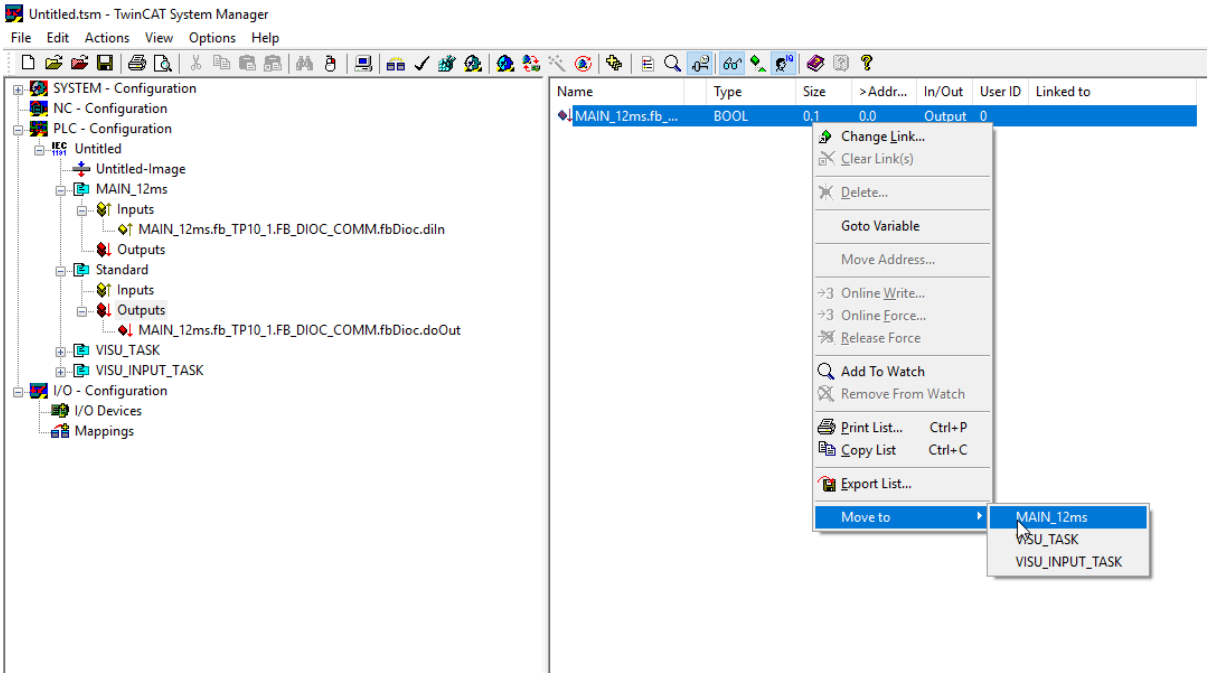
Calling I/O in the right task

The in- and outputs of the PLC will be called in a certain cycle time. The in- and outputs of the DIOC devices must be called in the same cycle time as the DIOC program. This can be done by calling the in- and outputs in the same task as the DIOC program. When the standard task has a 12 ms cycle time, this step can be ignored.

To call the in- and outputs in the right task, navigate to 'PLC-configuration/ PLC program / standard task / inputs' and select all DIOC inputs. Right-click on the selected inputs and go to 'move to'. Choose the task in which the DIOC instances are executed. In this example this is 'Main_12ms'.



Do the same for the outputs in 'PLC-Configuration/PLC program/ standard task/ outputs'.



Assigning the in- and outputs of the TP10 and RC

Outputs of the TP10 and RC have the name shown below

(=instance.FB_DIOC_COMM.fbDioc.doOut):

Inputs of the TP10 have the name shown below:

(= instance.FB_DIOC_COMM.fbDioc.diIn)

Sync unit assignment

For bigger projects, it might be a good idea to assign sync units to your I/Os. Without sync units the TP10's and RC's might not work if another I/O is missing or malfunctioning. Typically, a different sync unit should be assigned to every EtherCAT Coupler in your project. For more information on the sync units visit the Beckhoff information site.

https://infosys.beckhoff.com/english.php?content=../content/1033/tcsystemmanager/reference/ethercat/html/ethercat_syncunitassignment.htm&id=

5. Inputs and outputs of the TP10 block

Description usage of the inputs and outputs of the TP10

The TP10 block has a lot of inputs that can change the behaviour of the TP10.

As an example below the RGB leds of the TP10 are set to red. To do this, predefined colors can be used.

```
fb_TP10_1.dwRgb :=RGB_RED;
```

The other variables of the TP10 can also be addressed this way. The table below shows a list of all the inputs, outputs and configuration variables the TP10 has.

Other colour constants available in the DIOC library are listed below in the description of the input.

For a better understanding of a full implementation of a TP10, a very simple example is implemented in the sample project.

Inputs:

Name	Type	Description
bRoomAnalyser	BOOL	This boolean must be true of the connected device is a Room Analyser. If this boolean is true, all buttons are disabled, except button 10. Button 10 still be used to make the Room Analyser flash green. This can be used to test the DIOC communication with the PLC.
arr_bLeds	ARRAY [0..10] OF BOOL	Every button of the TP10 has it's own led. These can be controlled by changing the values in this array. True will make the led go on, false will make the led go off. arr_bLeds [1] = led 1, arr_bLeds [10] = led 10
iIntensityLeds	INT	Value between 0 and 100 that changes the intensity of the buttonleds.
arr_bMasks	ARRAY [0..10] OF BOOL	Every button of the TP10 can be turned off, this can be done by changing the values in this array. False means the button is enabled, true means the button is disabled. arr_bMasks [1] = button 1, arr_bMasks [10] = button 10, bRoomAnalyser overrules these.
arr_sButtonComments	ARRAY [0..10] OF STRING(8)	Every button of the TP10 has a short description (maximum 8 characters) that will be displayed on the visualisation. arr_sButtonComments [1] = comment button 1, arr_sButtonComments [10] = comment button 10
bReset	BOOL	When the TP10 has to be reset, this boolean should be set to true briefly. Once bReset is false again, the reset time will count to restart the TP10.

iIntervalCO	INT	Interval for the CO measurement in seconds. This determines how fast the measurements of the CO sensor must be checked. This is standard 7 (seconds). This variable must be changed before the start of the program. This value will not be sent to the TP10 once the TP10 program is running. After a restart or reset, this value will be sent again.
iIntervalCO2	INT	Interval for the CO2 measurements in seconds. This determines how fast the measurement of the CO2 sensor must be checked. This is standard 8 (seconds). The same conditions apply as iIntervalCO.
iIntervalIllumination	INT	Interval for the illumination measurement in seconds. This determines how fast the measurement of the illumination sensor must be checked. This is standard 13 (seconds). The same conditions apply as iIntervalCO.
iIntervalRoomHumidity	INT	Interval for the humidity measurement in seconds. This determines how fast the measurement of the humidity sensor must be checked. This is standard 11 (seconds). The same conditions apply as iIntervalCO.
iIntervalRoomtemp	INT	Interval for the roomtemperature measurement in seconds. This determines how fast the measurement of the roomtemperature sensor must be checked. This is standard 3 (seconds). The same conditions apply as iIntervalCO.
iIntervalVOC	INT	Interval for the VOC measurement in seconds. This determines how fast the measurement of the VOC sensor must be checked. This is standard 5 (seconds) The same conditions apply as iIntervalCO.
dwRgb	DWORD	The TP10 has a few RGB leds that can be used to light up the TP10. This value

		determines the intensity of each led. Predefined colors can be used for this input: RGB_BLACK , RGB_NAVY , RGB BLUE , RGB_GREEN , RGB_TEAL , RGB_LIME , RGB_AQUA , RGB_MAROON , RGB_PURPLE , RGB_OLIVE , RGB_GREY , RGB_ORANGE , RGB_FUCHSIA , RGB_YELLOW , RGB_WHITE You may also create your own color. To do this a DWORD has to be made. (eg. 16#1E8FE03F) In the example 1E is a hexadecimal value for the intensity, 8F is the red value, E0 is the green value and 3F is the blue value.
bEn	BOOL	Enable bit.
bLocate	BOOL	IF TRUE: makes the TP10 flash green 3 times to know which one you are currently using.
bWallSurface	BOOL	Boolean that lowers the sensitivity of the buttons. Enable this boolean for a surface mount TP10.

Outputs:

Name	Type	Description
qarr_bButtons	ARRAY [0..12] OF BOOL	Every button of the TP10 can be read. This can be done by reading the values from this array. False means the button is operated, true means the button is unoperated. qarr_bButtons [1] = button 1, qarr_bButtons [10] = button 10.
qfCO2	REAL	Value of the CO2 sensor in PPM (parts per million).
qfHumidity	REAL	Value of the humidity in percent.
qfLux	REAL	Value of the illumination sensor in lux.
qfRoomTemperature	REAL	Value of the temperature measurement in °C.
qfVOC	REAL	Value of the VOC sensor in PPB (parts per billion)

Systeminfo:

Name	Type	Description
qbDeviceActive	BOOL	Boolean that indicates if the TP10 is active. True = TP10 active False = TP10 not active
qdtVersionHw	DATE	Date of the hardware version of the TP10.
qdtVersionSw	DATE	Date of the software version of the TP10.
qdtVersionReg	DATE	Date of the register version of the TP10.
qrVoltageLevelA	REAL	Voltage level of the A line in Volt.
qrVoltageLevelB	REAL	Voltage level of the B line in Volt.
qsUniqueId	STRING	Unique ID of the TP10

Sample program listing TP10

```
PROGRAM P_Samples_TP10
VAR
	(* TP10 Parameters *)
	bEn			: BOOL := TRUE;
	bReset		: BOOL;
	iIntensityLeds : INT := 50; (* Change the led intensity *)

	(* Internal vars *)
	PFButton1	: R_TRIG;	(* a rising edge trigger *)

END_VAR


(* ------- CONTROL SAMPLES *)
(* Activate the device *)
fb_TestTP10_1.bEn			:= bEn;		(* OPTIONAL Device is standard enabled
internally *)

(* Reset the device *)
fb_TestTP10_1.bReset			:= bReset;

(* Toggle led when button 1 is pressed*)
PFButton1(CLK:= fb_TestTP10_1.qarr_bButtons[1]);
IF PFButton1.q THEN
	fb_TestTP10_1.arr_bLeds[1]:= NOT fb_TestTP10_1.arr_bLeds[1];
END_IF

(* No color when button 2 is pressed*)
IF fb_TestTP10_1.qarr_bButtons[2] THEN
	fb_TestTP10_1.dwRgb		:= RGB_BLACK;
END_IF

(* Disable button 3,4 and 5
	IF arr_bMasks[1] .. arr_bMasks[10] == TRUE then these buttons are disabled *)
fb_TestTP10_1.arr_bMasks[3]	:= TRUE;
fb_TestTP10_1.arr_bMasks[4]	:= TRUE;
fb_TestTP10_1.arr_bMasks[5]	:= TRUE;

(* Red color when button 6 is pressed*)
IF fb_TestTP10_1.qarr_bButtons[6] THEN
	fb_TestTP10_1.dwRgb		:= RGB_RED;
END_IF

(* (light) Red color when button 7 is pressed*)
IF fb_TestTP10_1.qarr_bButtons[7] THEN
	(* Non default colour with max (100%) intensity *)
	fb_TestTP10_1.dwRgb		:= 16#64_64_0A_0A;
END_IF

(* (light) Red color when button 8 is pressed*)
IF fb_TestTP10_1.qarr_bButtons[8] THEN
	(* Non default colour with standard (30%) intensity *)
	fb_TestTP10_1.dwRgb		:= 16#1E_64_19_19;
END_IF
```

```
(* (light) Blue color when button 9 is pressed*)
IF fb_TestTP10_1.qarr_bButtons[9] THEN
	(* Non default colour with medium (65%) intensity *)
	fb_TestTP10_1.dwRgb	:= 16#41_19_19_64;
END_IF

(* Blue color when button 10 is pressed*)
IF fb_TestTP10_1.qarr_bButtons[10] THEN
	fb_TestTP10_1.dwRgb	:= RGB_BLUE;
END_IF

(* Change the led intensity *)
fb_TestTP10_1.iIntensityLeds	:= iIntensityLeds;

(* Interval update times for the sensors *)
fb_TestTP10_1.iIntervalRoomtemp;		(* Default := 3 *)
fb_TestTP10_1.iIntervalRoomHumidity;	(* Default := 11 *)
fb_TestTP10_1.iIntervalCO2;			(* Default := 8 *)
fb_TestTP10_1.iIntervalCO;			(* Default := 7 *)
fb_TestTP10_1.iIntervalVOC;			(* Default := 5 *)
fb_TestTP10_1.iIntervalIllumination;		(* Default := 13 *)
fb_TestTP10_1.iIntervalIR;			(* Default := 17 *)

(* ------- FEEDBACK SAMPLES,
	* these are the measured values a TP10 can return
	* be aware that it might take along time before the data is available after starting
	*)
fb_TestTP10_1.qfRoomTemperature;
fb_TestTP10_1.qfVOC;
fb_TestTP10_1.qfCO2;
fb_TestTP10_1.qfHumidity;
fb_TestTP10_1.qfLux;
fb_TestTP10_1.qfInfrared;

(*Analyse data*)
fb_TestTP10_1.qdtVersionHw;
fb_TestTP10_1.qdtVersionSw;
fb_TestTP10_1.qdtVersionReg;
fb_TestTP10_1.qsUniqueId;
fb_TestTP10_1.qrVoltageLevelA;
fb_TestTP10_1.qrVoltageLevelB;
fb_TestTP10_1.qbDeviceActive;
```

6. Inputs and outputs of the RC block

Description usage of the inputs and outputs of the RC

The RC block has a lot of inputs that can change the behaviour of the RC.

The table below shows a list of all the inputs, outputs and configuration variables the RC has.

Inputs:

Name	Type	Description
bEn	BOOL	This boolean must be true of the connected device is a Room Analyser. If this boolean is true, all buttons are disabled, except button 10. Button 10 still be used to make the Room Analyser flash green. This can be used to test the DIOC communication with the PLC.
bEnableFan	BOOL	enable fan bit (relay pin 41-42)
bHeating_3P_plus	BOOL	Heating plus signal (output pin 7)
bHeating_3P_min	BOOL	Heating min signal (output pin 8)
bCooling_3P_plus	BOOL	Cooling plus signal (output pin 20)
bCooling_3P_min	BOOL	Cooling min signal (output pin 21)
bFireDamper_OPN	BOOL	open signal fire damper (output pin 33)
bFireDamper_CLS	BOOL	close signal fire damper (output pin 34)
bRelais_45	BOOL	relay pin 45 (DO3)
bRelais_46	BOOL	relay pin 46 (DO2)
bRelais_47	BOOL	relay pin 47 (DO1)
iIntervalACVoltage	UDINT	retrieval time ac voltage (in seconds)
iIntervalTempHeatingWater	UDINT	retrieval time temperature heating water (in seconds)
iIntervalTempICEWater	UDINT	retrieval time temperature ice water (in seconds)
iInterval_FB_Pulsion	UDINT	retrieval time feedback pulsion (in seconds)
iIntervalTempAirPulsion	UDINT	retrieval time temperature pulsion air (in seconds)
iInterval_FB_Extraction	UDINT	retrieval time feedback extraction (in seconds)
iIntervalTempAirExtraction	UDINT	retrieval time temperature extraction (in seconds)
iIntervalDipswitches	UDINT	retrieval time dipswitches (in seconds)
iIntervalFBFiredamper	UDINT	retrieval time feedback firedamper (in seconds)
iIntervalFanFaultStatus	UDINT	retrieval time fan fault (in seconds)

iSendIntervalSpHeating	INT	send interval time for the heating set point in seconds*
iSendIntervalSpCooling	INT	send interval time for the cooling set point in seconds
iSendIntervalSpPulsion	INT	send interval time for the pulsion set point in seconds
iSendIntervalSpExtraction	INT	send interval time for the extraction set point in seconds
iSendIntervalSpFan	INT	send interval time for the fan set point in seconds
bForceSendSpHeating	BOOL	Set this to true to send the heating set point immediatly
bForceSendSpCooling	BOOL	Set this to true to send the cooling set point immediatly
bForceSendSpPulsion	BOOL	Set this to true to send the pulsion set point immediatly
bForceSendSpExtraction	BOOL	Set this to true to send the extraction set point immediatly
bForceSendSpFan	BOOL	Set this to true to send the fan set point immediatly
iSpHeating	INT	heating setpoint in % 0% = 0V, 100% = 10V
iSpCooling	INT	cooling setpoint in % 0% = 0V, 100% = 10V
iSpPulsion	INT	pulsion setpoint in % 0% = 0V, 100% = 10V
iSpExtraction	INT	extraction setpoint in % 0% = 0V, 100% = 10V
iSpFan	INT	fan setpoint in % 0% = 0V, 100% = 10V
bReset	BOOL	if true: resets the Room Controller

arr_sConnectionComments	ARRAY [1..41] OF STRING(8)	comments for every connection that is visible on the visualisation

Outputs:

Name	Type	Description
qbFiredamperFB_OPN	BOOL	feedback firedamper open (input pin 37)
qbFiredamperFB_CLS	BOOL	feedback firedamper closed (input pin 36)
qbFanFault	BOOL	fan fault (input pin 43)
qarr_bDipSwitches	ARRAY[1..12] OF BOOL	status dipswitches
qfACVoltageLevel	REAL	measured ac voltage
qfTempHeatingWater	REAL	temperature heating water in °C (PT1000 pin 12-13)
qfTempIceWater	REAL	temperature ice water in °C (PT 1000 pin 25-26)
qfPulsionFB	REAL	pulsion vav feedback in % (pin 17) 0% = 0V, 100% = 10V
qfTempAirPulsion	REAL	temperature pulsion in °C (PT1000 pin 18-19)
qfExtractionFB	REAL	extraction vav feedback in % (pin 30) 0% = 0V, 100% = 10V
qfTempAirExtraction	REAL	temperature extraction in °C (PT1000 pin 31-32)

Systeminfo:

Name	Type	Description
qbDeviceActive	BOOL	Boolean that indicates if the RC is active. True = RC active False = RC not active
qdtVersionHw	DATE	Date of the hardware version of the RC.
qdtVersionSw	DATE	Date of the software version of the RC.
qdtVersionReg	DATE	Date of the register version of the RC.
qrVoltageLevelA	REAL	Voltage level of the A line in Volt.
qrVoltageLevelB	REAL	Voltage level of the B line in Volt.
qsUniqueId	STRING	Unique ID of the RC