FUP XL (Programming Software)

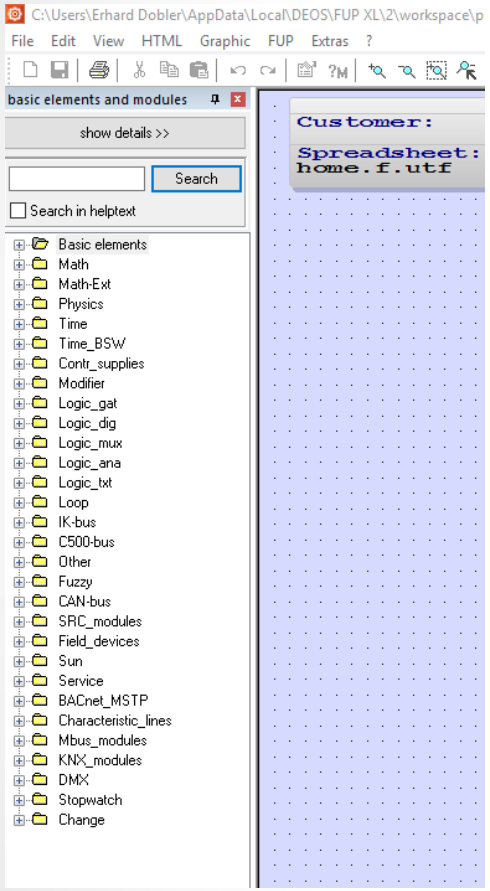
DEOS Controls Training Class

Agenda

Agenda Day #2

1. Introduction to FXL
2. Function Block Groups
3. Function Blocks functionality
4. Create Project
5. Create Program
6. Create Graphics
7. Program Simulation
8. Graphics Simulation

Function Block Groups

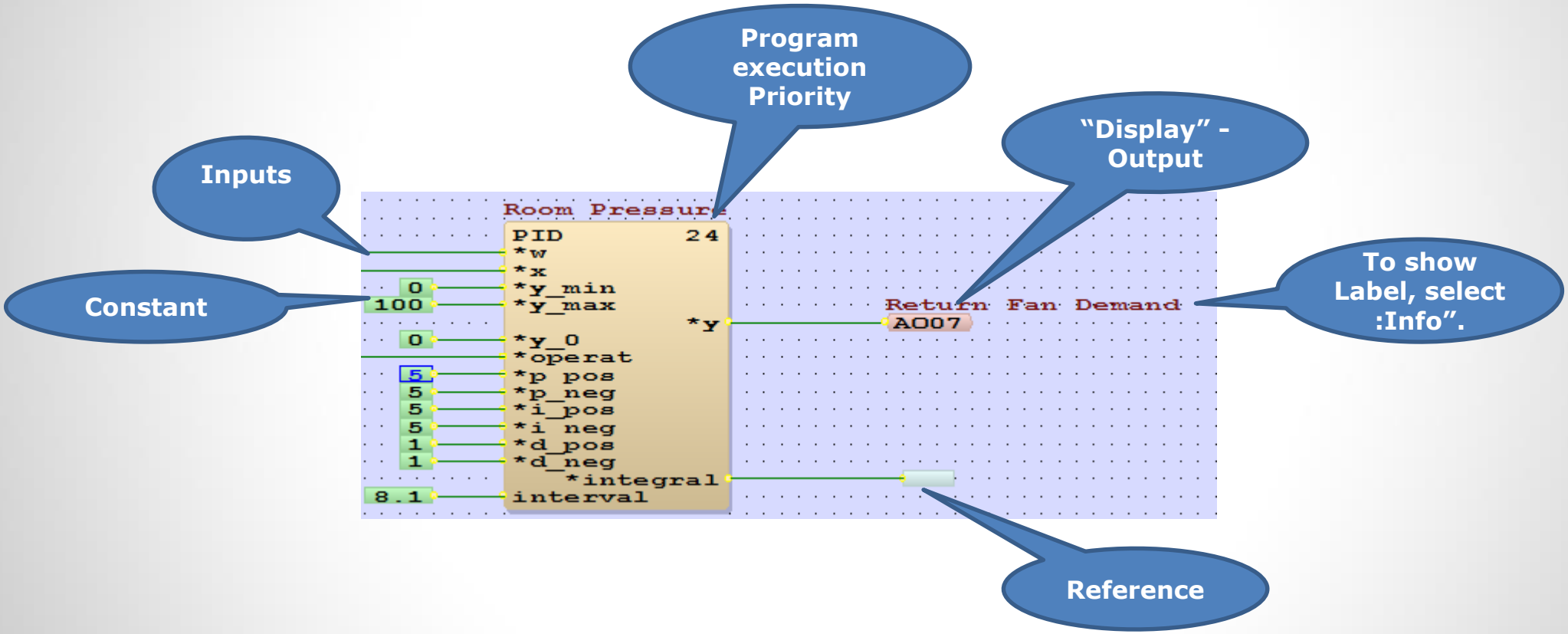


Open “Basic Elements and Modules” at the “View” in the main menu .

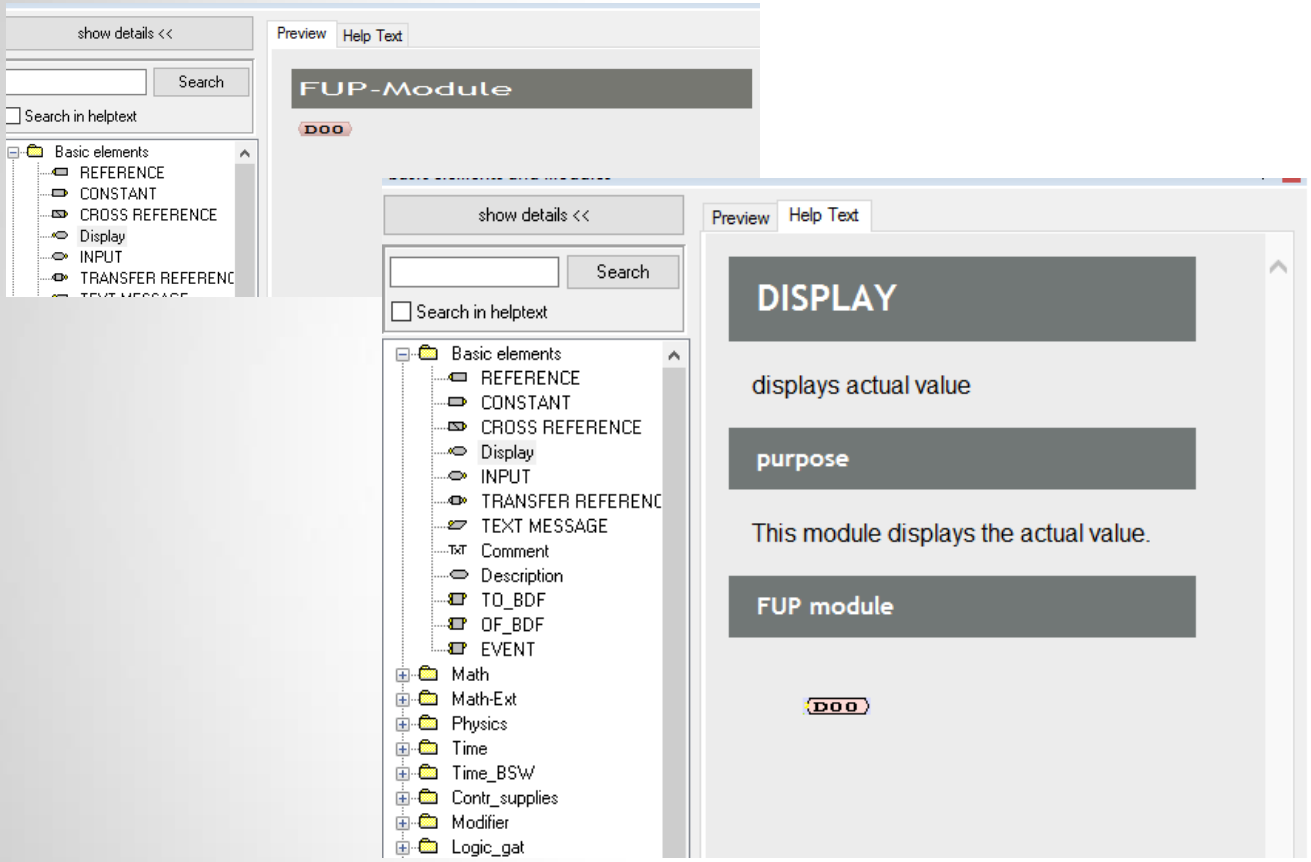
Tip: you can use the hot key “F2”

Each Element Group folder is expandable by pressing (**+**) and collapsible by (**-**)

Function Basic Elements Groups



Function Block Groups

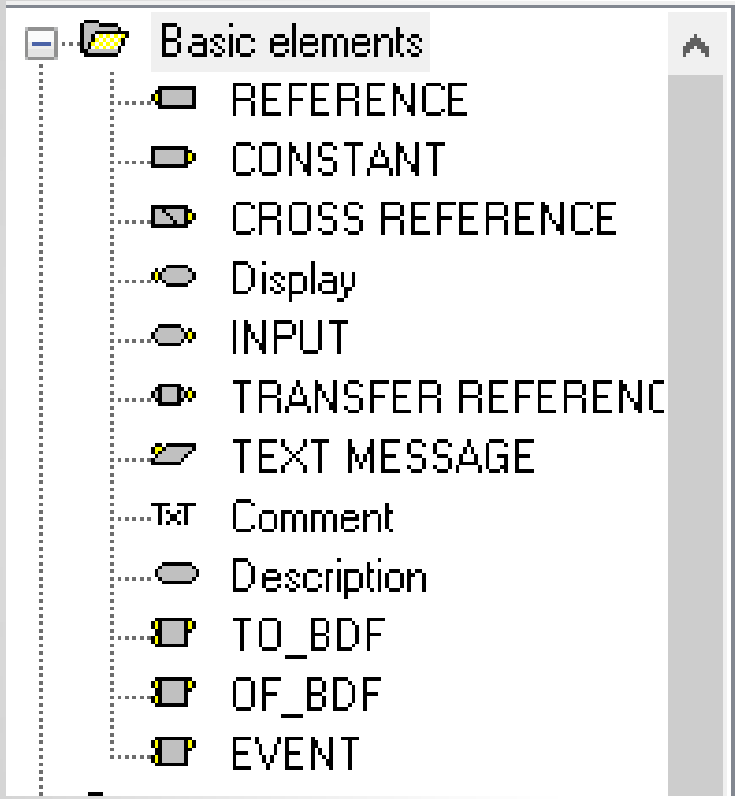


Every Function block has a detailed information, that can be displayed by the “show detail” button.

It opens the “Preview” and “Help Text” tabs, that show labels, inputs, outputs and explain functionality of the function block or element.

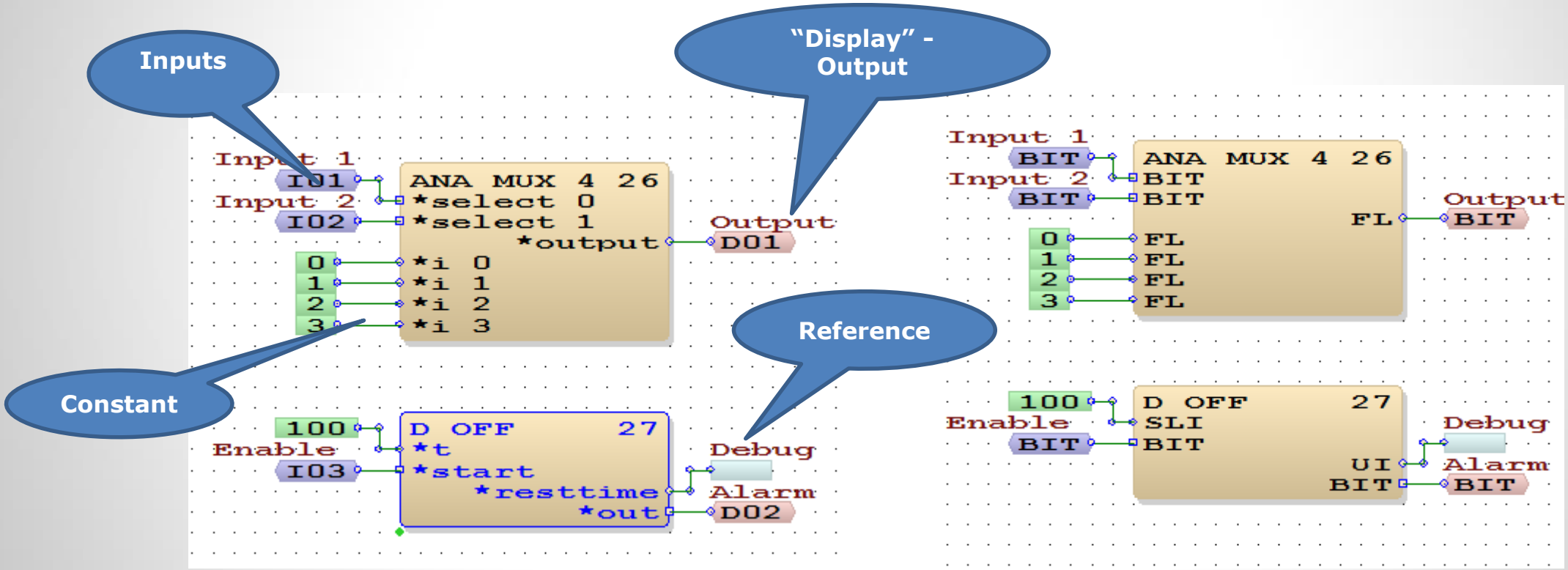
The help file is the part of the programming tool FUP XL and it can assist during development.

Basic Elements Group

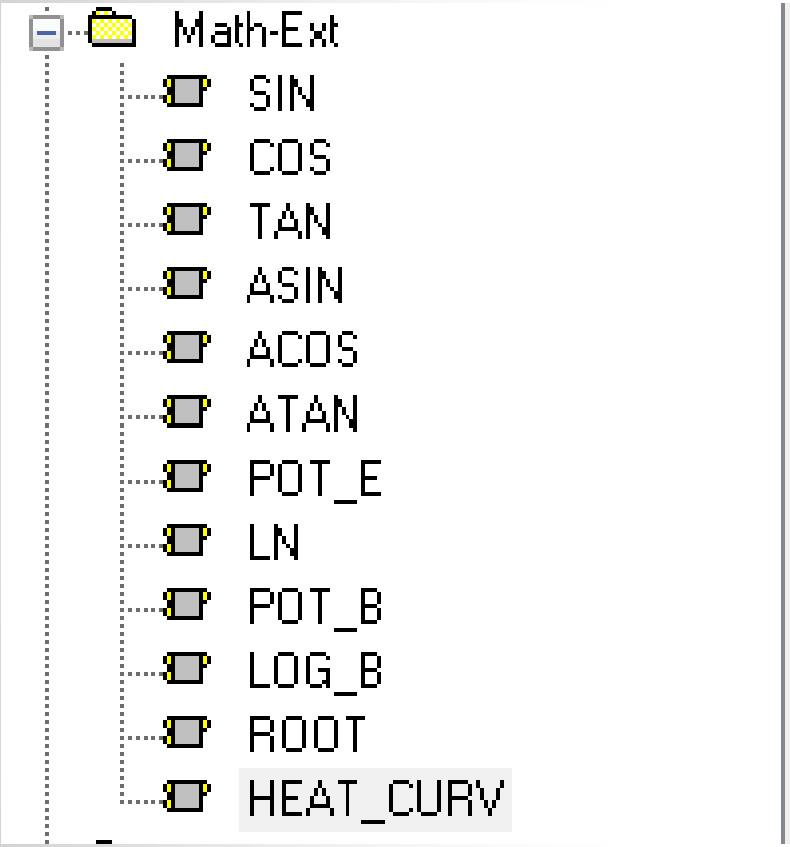


- **Reference** – this value can be shared over multiple program pages and controllers. **Tip: you can use "Reference" on the output side of the function block to avoid error during compilation.**
- **Constant** – this value represents value, that does not change.
- **Cross Reference** – This element returns value of a Reference inside the same controller.
- **Display** – display or output. It is variable, that value can be changed by user or program. **It represents the Output value.**
- **Input** – Represents Input values. It is variable, that value can be changed by user or program
- **Text message** – displays text messages, in the "Events" page in the controller. Can work as alarm and text can be sent via e-Mail. Works as binary value (0 - off, 1 - on)
- **Comments** –text information which you want to place in the program page
- **Description** – to label points or write comments in the program page

Function Basic Elements Groups



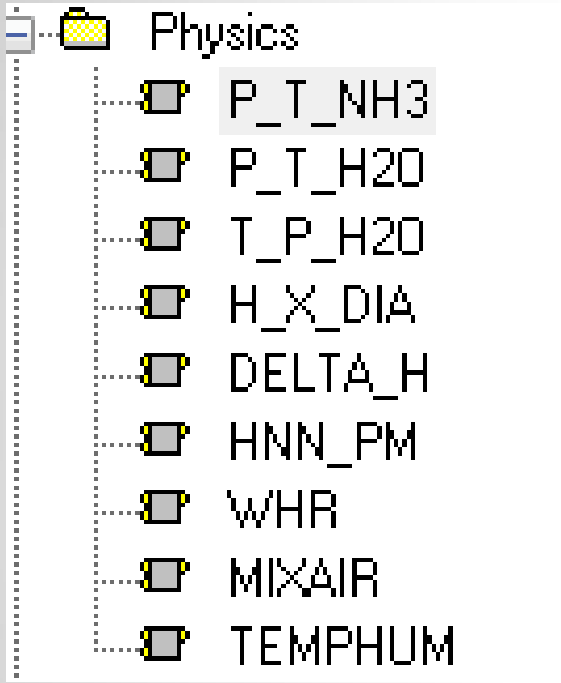
Math Operation (Float Numbers) Group



- **Math Extension are Mathematical calculation which are implemented in the function block.** Typical Math Operations, mostly with float numbers.

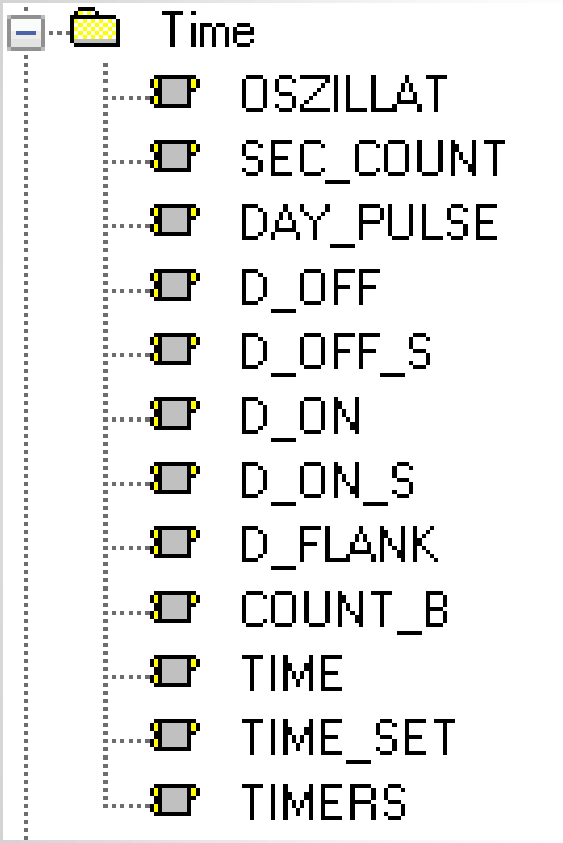
- **Heat_Curv –** This function is used to schedule the water temp. for Radiation and Under floor heating systems. It calculates the Supply Water set point based on OAT temperature.

Physics Group



- **P_T_NH3** - Converts the saturation pressure in the boiling temperature for NH3
- **P_T_H20** - Calculates the dew point temperature for ice and water at given vapor pressure.
- **T_P_H20** - calculates the partial pressure for ice and water by given temperature and humidity.
- **H_X_DIA** - calculates the air situation of temperature "theta", humidity "rh" and barometric pressure "p".
- **Delta H** - calculates the heating and cooling energy which is needed to turn air from one condition to another.
- **HNN_PM** - calculates the average barometric pressure "p_medial" by given location height "h".
- **WHR** – Control of the waste heat recovery with outside and circulating air throttles.
- **MIXAIR** - Calculation of temperature and humidity of mixed air
- **TEMPHUM** - Calculates the humidity, if the temperature of the air is changed.

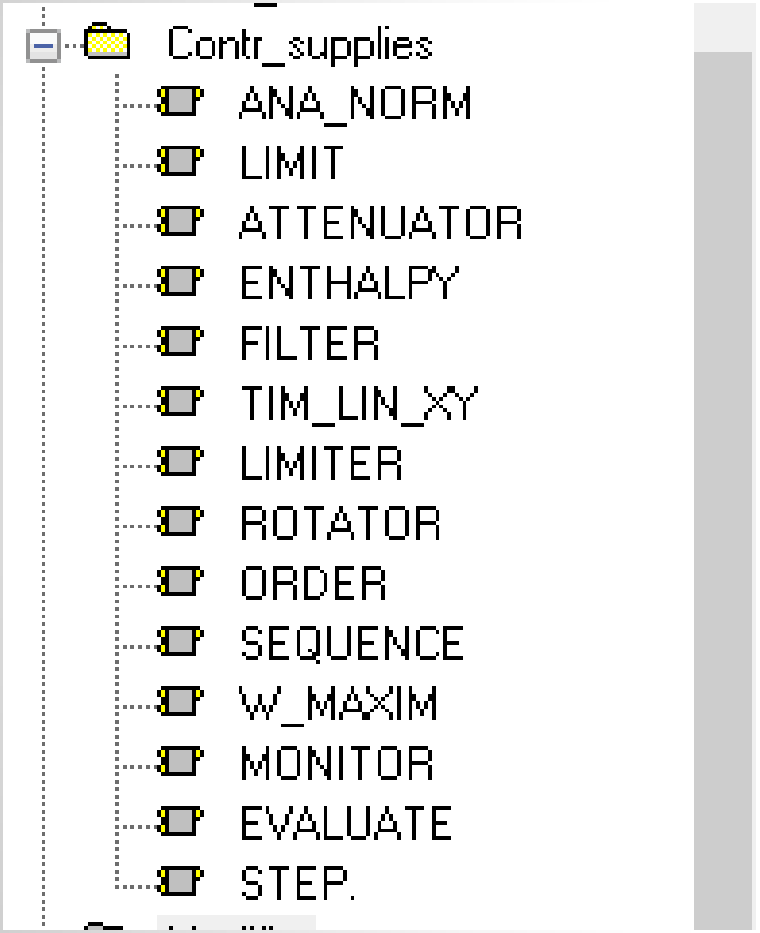
Timers and Time Delay Group



- **OSZILLAT**- Clocks the output "out" when the input "start" is assigned to "1". The pulse and pause times will be set by the variables "pulse" and "pause". While the input "start" is assigned to "0", the output "out" is set to "0“
- **SEC_Count**- Executes the counting of seconds.
- **DAY_PULSE -** Writes out a "1" to the variable "out" for the time which is given by the variable "duration“.
- **DAY_OFF -** Executes a falling delay. Reset time in 1/10 Sec.
- **DAY_OFF_S** - Executes a falling delay. Reset time in Seconds.

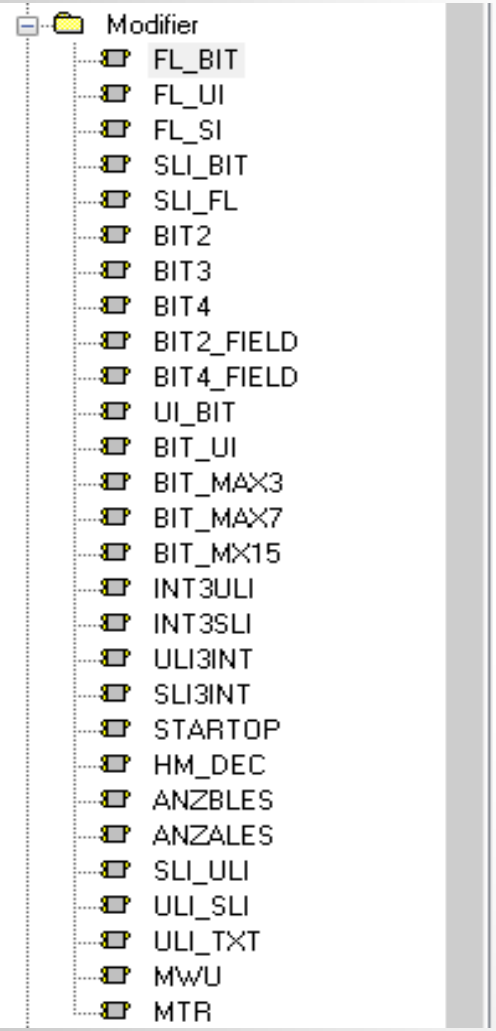
- **DAY_ON -** Executes a Turn on delay. Reset time in 1/10 Sec.
- **DAY_ON_S** - Executes a Turn on delay. Reset time in Seconds.
- **D_FLANK** – Delay module.
- **Count_B** - Operating hours counter (Runtime).
- **TIME** - Used to present the current time.
- **TIME_SET** - Set UST-clock
- **TIMERS** - Measures the time of the input impulses at the inputs "more" and "less".

Control Group



- **Limit**– Limits the OUTPUT by two set values.
- **Filter** – Harmonizes inputs signals
- **Tim_Lin_XY** – Liniear output value which can be timed to avoid cycling of OUTPUT signals
- **Limiter** – Limits output signals to pre-set values. For example 0-100% limit, 20% to 80%.
- **Rotator** – Rotates outputs to equal run times on equipment. For example rotating pumps ext. up to 16 devices per function block

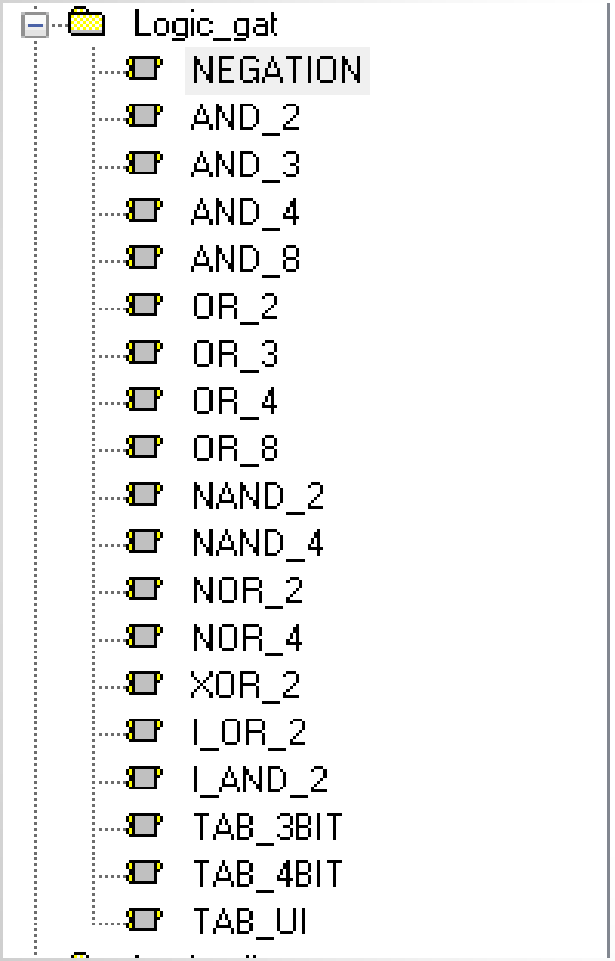
Variable Modifier Group



- **FL_BIT** – Converts a Float value (analog) to a Bit (digital) signal
- **FL_UI** - Changes a float number to an unsigned integer.
- **FL_SI** - Changes a float number to an signed integer.
- **SLI_BIT** - analyzes the value of an integer and operates the output to 1 while the integer value is not equal to 0.
- **SLI_FL** - converts an integer to a float number.
- **BIT2** - analyzes the states of bit-fields of the BDF.
- **BIT3** - analyzes the states of bit-fields of the BDF. A field of 3 bit will be created if there are given 8 ASCII-characters
- **BIT4** - analyzes the states of bit-fields of the BDF. A field of 4 bit will be created if there are given 16 ASCII-characters.
- **BIT2_FIELD and BIT4_FIEL**- This module creates characters at a bit display field of the BDF as a result of the bit pattern in the UST.
- **UI_Bit** - Converts an integer to a bit pattern.
- **BIT_UI** - Converts a bit pattern into a number.
- **BIT_MAX3**/7/15 - sets the number of the highest set bit to the output "out".
- **START_OP** - converts the marker states of the day- \ night- \ clock-choose for the graphic.

NOTE: Compiler does not need BIT (binary) to FL (float) conversion.

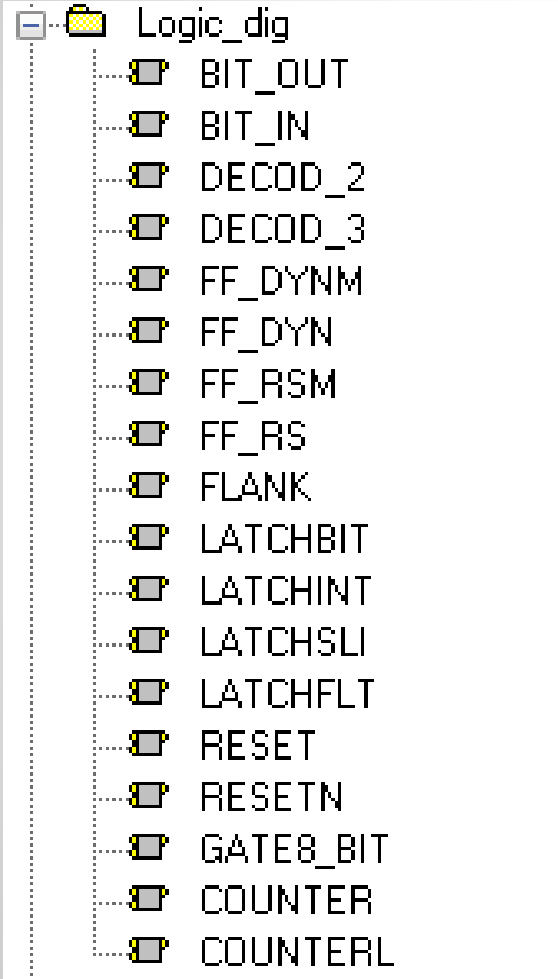
Logic Operations Group



Typical Binary or Boolean Operations.

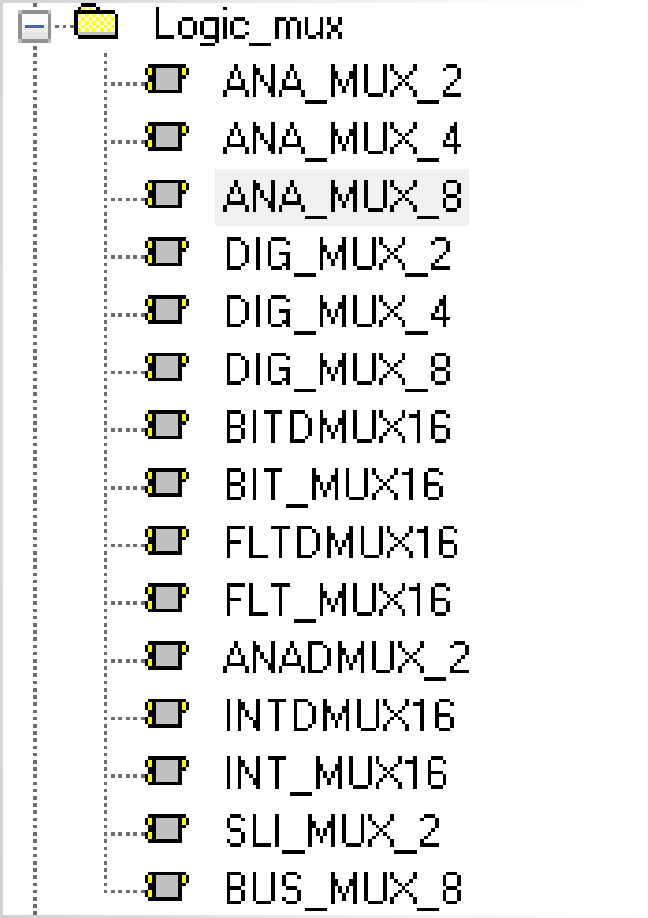
- **NEGATION –** Reversing a Digital Signal
- **AND_2,** 3,4,8 – AND-GATE
- **OR_2,**3,4,8 – OR-GATE
- **NAND_2**,4 – NAND – GATE
- **NOR_2**,4 – NOR – GATE
- **XOR-** Exclusive OR Gate
- **I-AND_2 –** Exclusive AND Gate

Logic Operations Group



- **BIT_OUT –** Output selector, one inputs and 5 outputs.
- **BIT_IN,** Totalizes four Digital inputs and summarizes on the Output
- **DECOD_2** – analysis of 2 digital inputs.
- **DECOD_3** – analysis of 3 digital inputs
- **FF_DYNM** – works like an impulse relay.
- **FF_DYN -** works like an impulse relay.
- **FF_RS –** RS flip flop. While the input "i_0" is set to 1 a 0 is written on the variable "output". If "i_0" is set to 0 and "i_1" to 1 then the output "output" is overwritten with a 1 until "i_0" is set to 1 again.
- **LATCHBIT** - transfers the status of "input" to "output" as long as the input "enabling" = 1. If "enabling" = 0, the status of "input" will not be transmitted to "output" anymore. Then "output" will save the latest transmitted value.
- **RESET** - This module creates a reset impulse of adjustable time.
- "RESET" sets independently the input to null after run of the given time
- **RESETTN** - This module creates a reset impulse of adjustable time.
- "RESETN" sets independently the input to null after run of the given time.

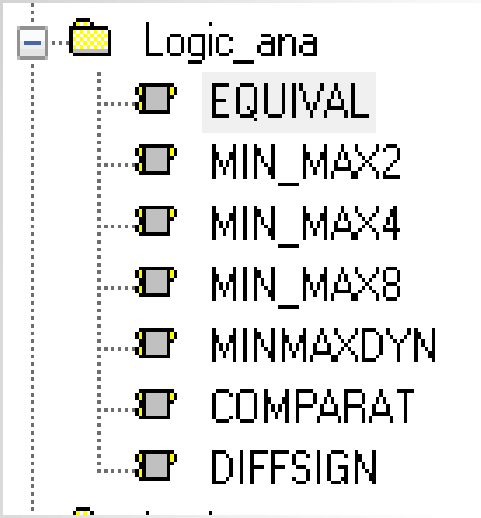
Selectors or Switches and If statements



Mostly Selectors, Switches and If statements.

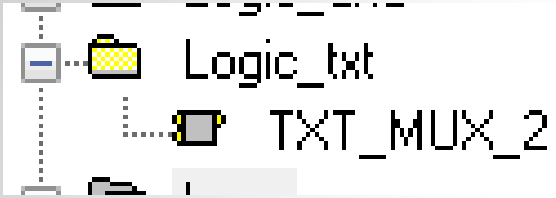
- **ANA_MUX2 –** Analog Multiplexer – 2 dig Inputs
- **ANA_MUX4 –** Analog Multiplexer – 4 dig Inputs
- **ANA_MUX8 –** Analog Multiplexer – 8 dig Inputs
- **DIG_MUX2 –** Digital Multiplexer – 2 dig Inputs
- **DIG_MUX4 –** Digital Multiplexer – 4 dig Inputs
- **DIG_MUX8 –** Digital Multiplexer – 8 dig Inputs
- **BITDMU-16** – De-multiplexer for marker
- **BIT_MUX16** - Multiplexer for marker
- **FLTDMUX** – De-multiplexer for float numbers
- **FLT_MUX16** - multiplexer for float numbers
- **ANADBUX_2**- De-multiplexer for float values
- **INTDMUX_16** – De-multiplexer for long data words. One Input and 16 Outputs
- **INT_MUX16 -** Multiplexer for long data words. 16 Inputs and 1 Output.
- **SLI_MUX_2** - SLI-multiplexer, If input "selector" = 0, then “Output" = "i_0".
- If input "selector" = 1, then “output" = "i_1".
- **BUS_MUX_8 - S**witching of an 8-bit wide bus. If the input "select" is taken by a number higher than "0" the states of the inputs "in1_x" will be set to the outputs "out_x", otherwise the states of the inputs "in0_x" will be set to the outputs "out_x".

Comparisons and Minimum Maximum Group

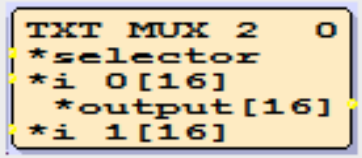


- **EQUIVAL–** Analog Multiplexer – 2 dig Inputs. For example - IF IP1=1 and IP =1 THEN OP1=1, IF IP1=1 and IP2=0 Then OP1 =0.
- **MIN_MAX2** - This module compares the value of the variable "input" with the value of "minimum" and "maximum" and writes out the accordant limited value to "output" and visible for the user to "input".
- **MIN_MAX4** - This module compares the values of the variables "i_1" up to "i_3" and writes out the minimum value to the output "min" and the maximum value to the output "max".
- "MIN_MAX_4" sets up the AND- and OR-relation respectively of 4 elements for FUZZY-controllers.
- **MIN_MAX8** - This module compares the values of the variables "i_1" up to "i_7" and writes out the minimum value to the output "min" and the maximum value to the output "max".
- "MIN_MAX_8" sets up the AND- and OR-relation respectively of 4 elements for FUZZY-controllers.
- **MINMAXDYN** - This module compares the values of the variables "i_1" up to "i_7" and writes out the minimum value to the output "min" and the maximum value to the output "max".
- "MIN_MAX_8" sets up the AND- and OR-relation respectively of 4 elements for FUZZY-controllers.
- **COMPARAT** - This module compares the values of the variables "i_min" and "i_max". While "i_min" < "i_max", a "1" will be set to the variable "output".
- **DIFFSIGN** - This module describes a comparator and compares the variables "i_1" and "i_2".

Text Switch Group

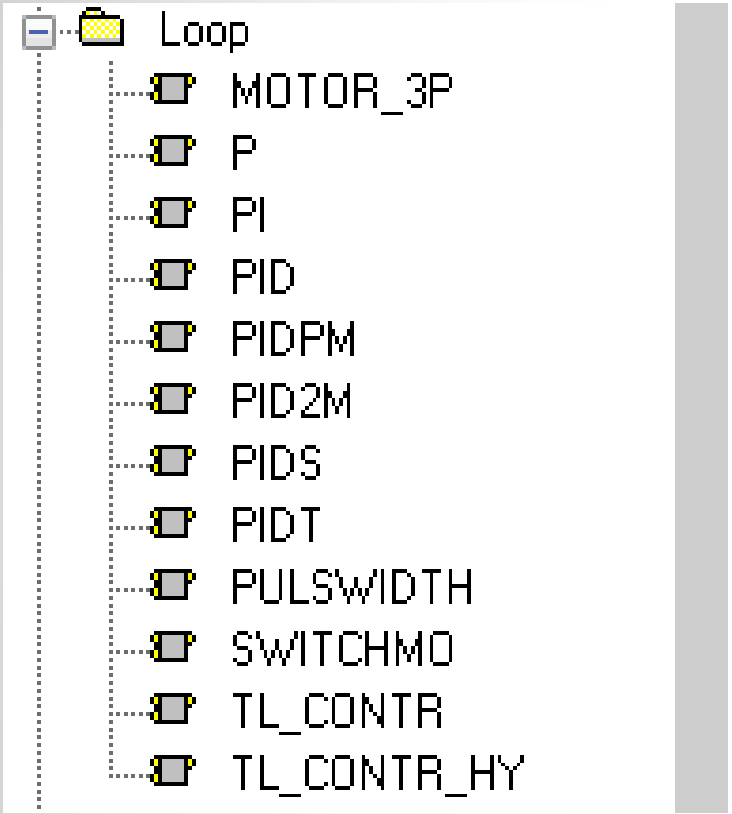


- **TXT_MUX_2**



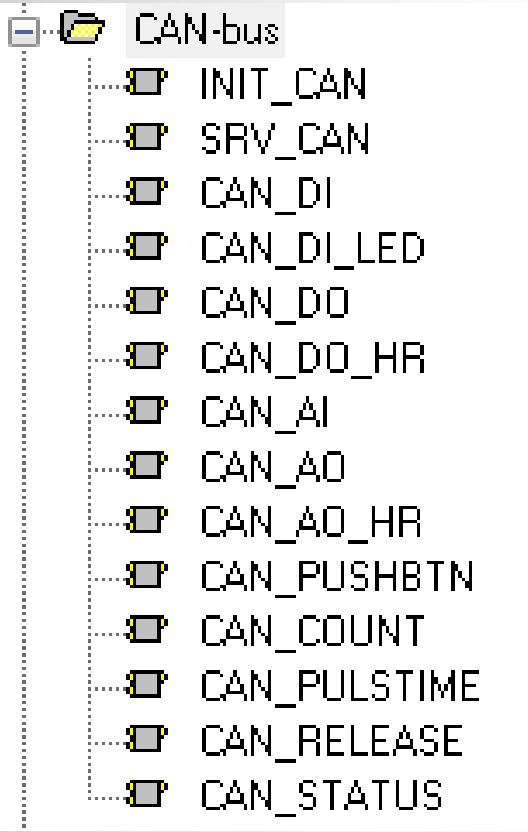
selector	gate input
i_0[16]	text input if gate input = 0
output[16]	text output
i_1[16]	text input if gate input = 1

Loops Group



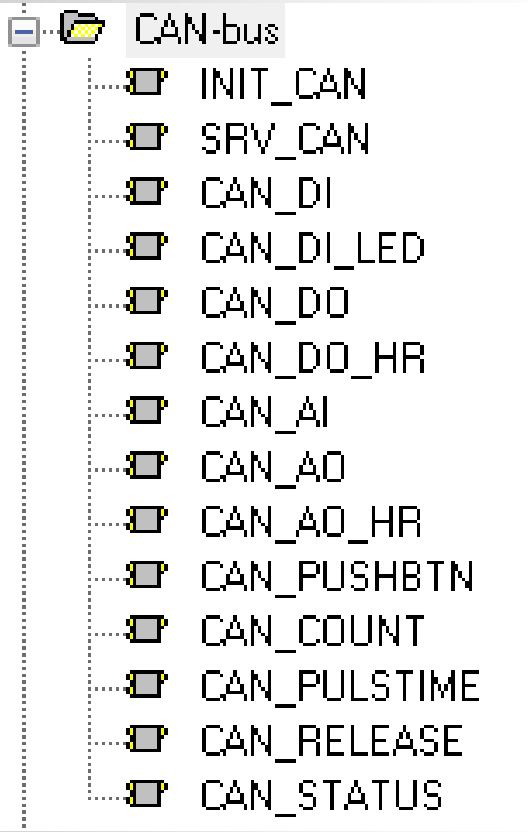
- **MOTOR_3P –** Floating point control with pre-set run-time of motor.
- P – Proportional Control
- PI – Proportional Integral Control
- PID - Proportional–Integral–Derivative Control
- **PIDPM** - Proportional–Integral–Derivative Floating Point Control
- **PID2M** - Proportional–Integral–Derivative Floating Point Control and analog Control output.
- **PIDS** - If "tn" <= 0, the PI-loop will not change the value of "integral" but always keeps the latest given or calculated value.
- If "y_min" > "y_max", the effective direction of the controller will be reversed.
- **PIDT** - If ("i_pos" = 0) and ("i_neg" = 0) then does the PID-loop never change the value of "integral", but always keeps the latest entered value.
- **SWITCHMO** - switching module and switches four digital outputs depending to the value of the variables "w".
- **TL_CONTR** - switching module and switches a digital output in dependance of the value of the variable "x".
- **TL_CONTRL_HY** - two-level controller with adjustable switching hysteresis

CAN-bus Group – Part 1



- **INIT_CAN –** This module is for the initialization of a CAN-bus. Used if I use the Can bus port on the controller.
- **SRV_CAN** – service-module for CAN-bus. Used if I use the Can bus port on the controller.
- **CAN_DI – Can** bus Digitial Input (For DI16) It is not used with the OPRN UI8x8
- **CAN_DI_LED** – This module handles the LED-colors of digital inputs. CAN_DO – Can Bus Module with Digital output
- **CAN_DO_HR** - This module connects the COSMOS IO modules DA8T(H), DA8R(H)
- **CAN_AI** - This module connects the COSMOS IO modules AI8AO4(H).
- Reads the current state of an analog input.
- **CAN_AO** - This module connects the COSMOS IO modules AE8AA4(H).
- Sets the status of an analog output.

CAN-bus Group – Part 2



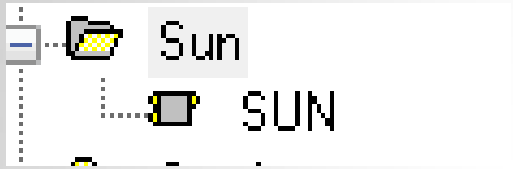
- **CAN_AO_HR** - This module connects the COSMOS IO modules AE8AA4(H).
- Sets the status of an analog output.
- **CAN_PUSHBTN** - Connects the COSMOS IO modules DE16

CAN_COUNT - This module connects the COSMOS IO modules DE16. Output of the counter reading of a digital input.

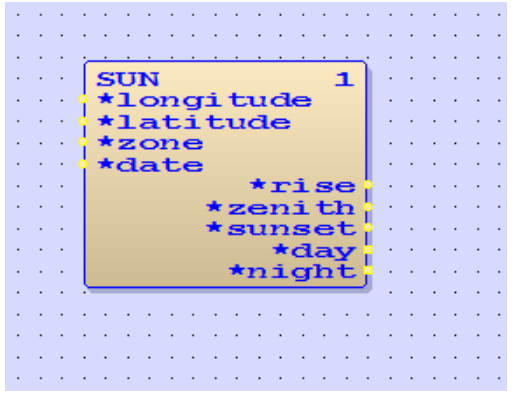
"**CAN_PUSHBTN**" only has to be used once per hardware module and terminal in the project.

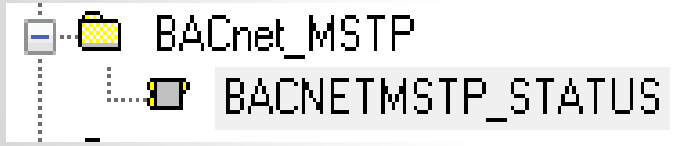
- **CAN_PULSE_TIME** - With this module the time spans between two impulses of an digital input can be measured. It will be measured the time span of the "1" as well as the time span of two impulses.
- **CAN_RELEASE** - Dropping of an output DI8DO8T
- **CAN_STATUS** - Provides runtime information about the CAN data bus.

Misc. Groups

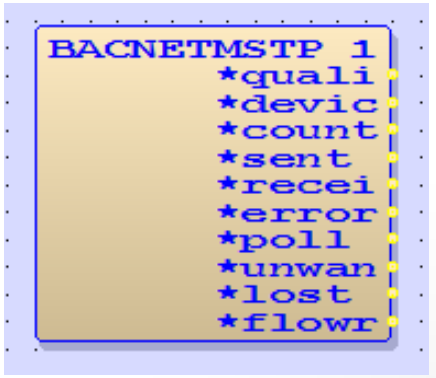


➢**SUN-** SUN allows to calculate the sunrise and the sunset

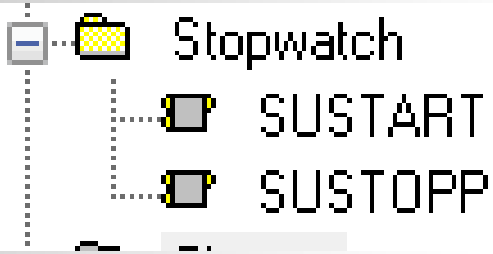




➢**BACNETMSTP** - Providing runtime information about the MS/TP data bus.



Stop Watch



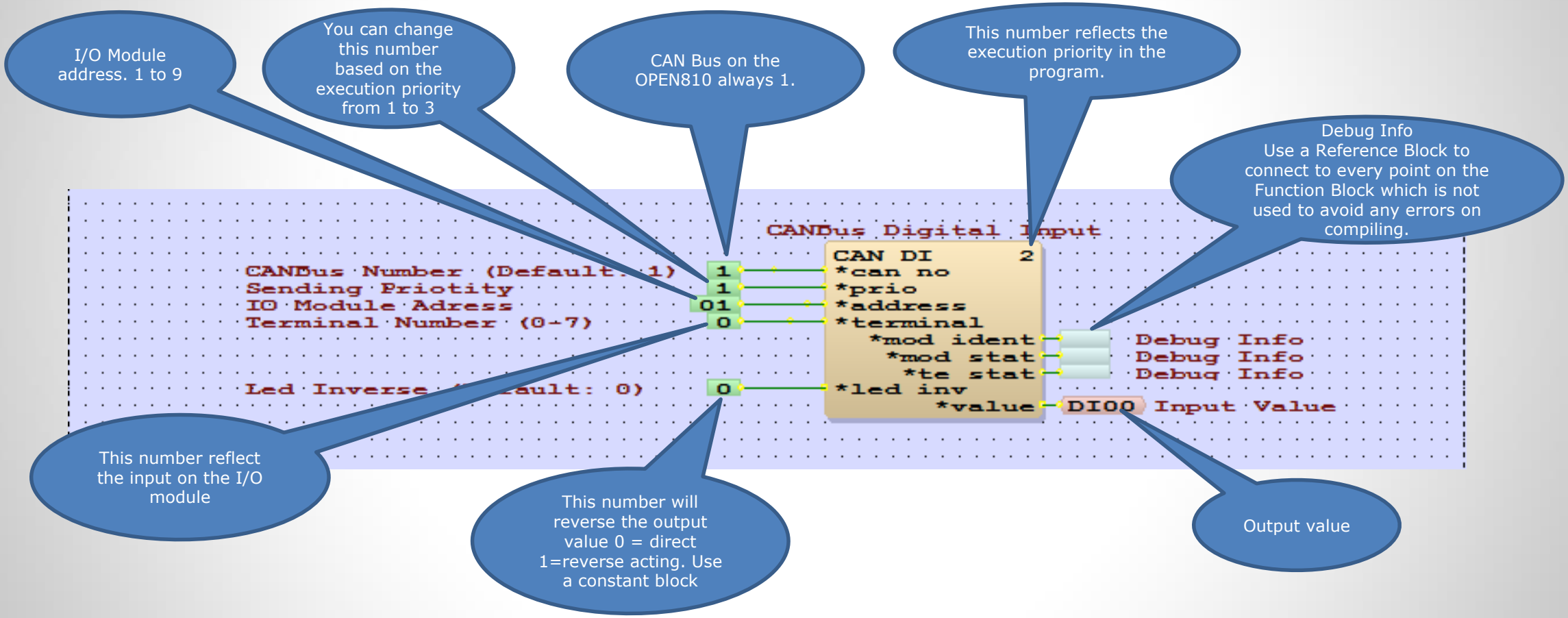
SUSTART – By the stopwatch function it is possible to create multiple stopwatches on a controller which can detect with a high resolution intervals in a runnable controller program.

- These stopwatches can be used for example to optimize controller programs according to reaction timing on changed input values. With this module a stopwatch can be started.

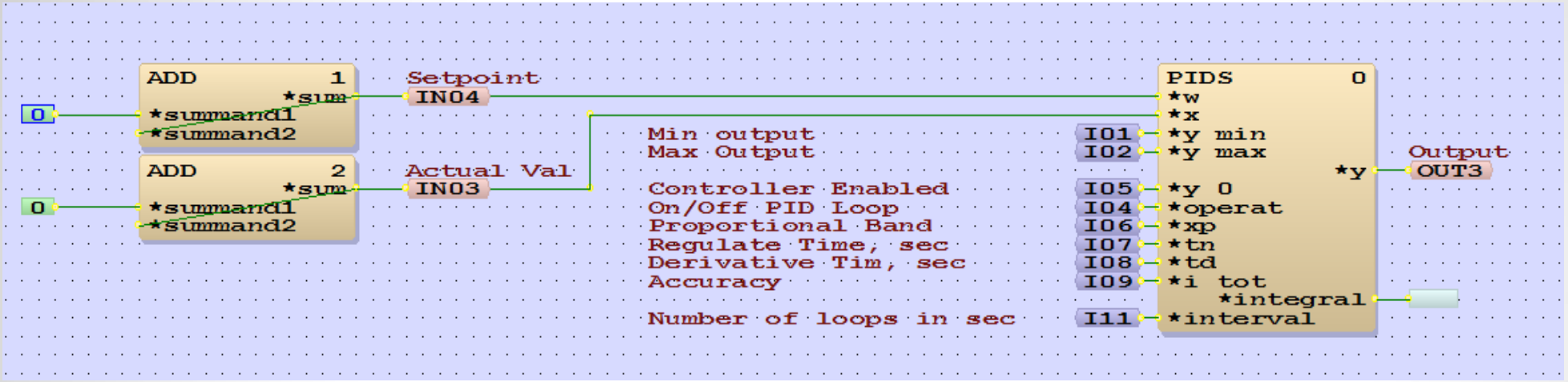
SUSTOP - By the stopwatch function it is possible to create multiple stopwatches on a controller which can detect with a high resolution intervals in a runnable controller program.
These stopwatches can be used for example to optimize controller programs according to reaction timing on changed input values. The interval of a running stopwatch is measured by this module.

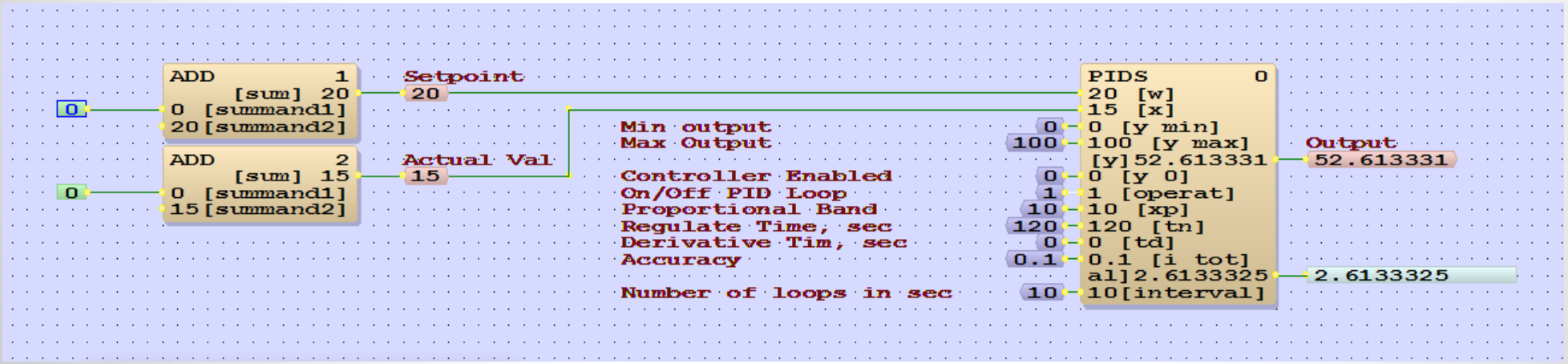
- You can use as many stop times depending on one stopwatch as you whish.

CAN Bus Function Block Setup

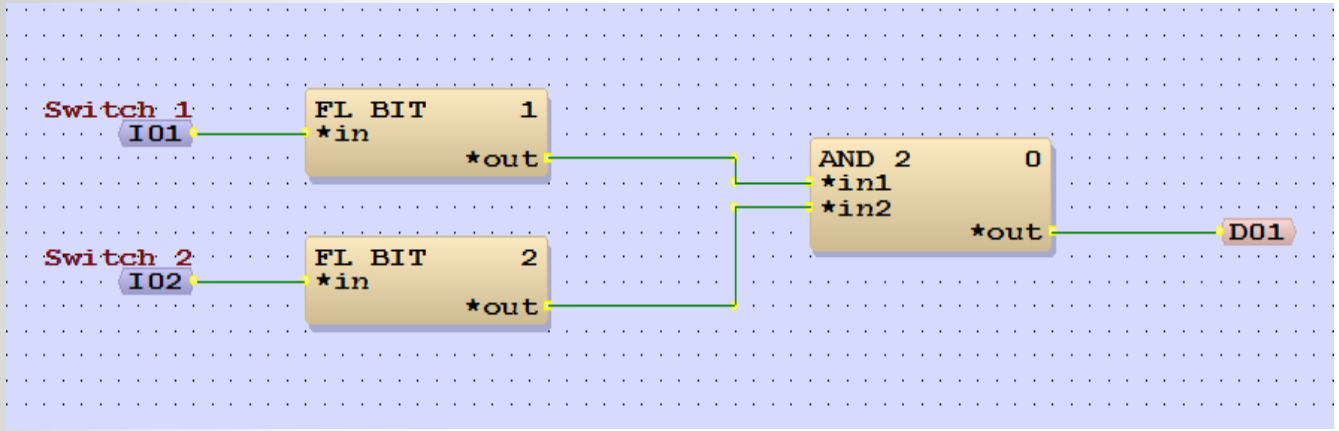


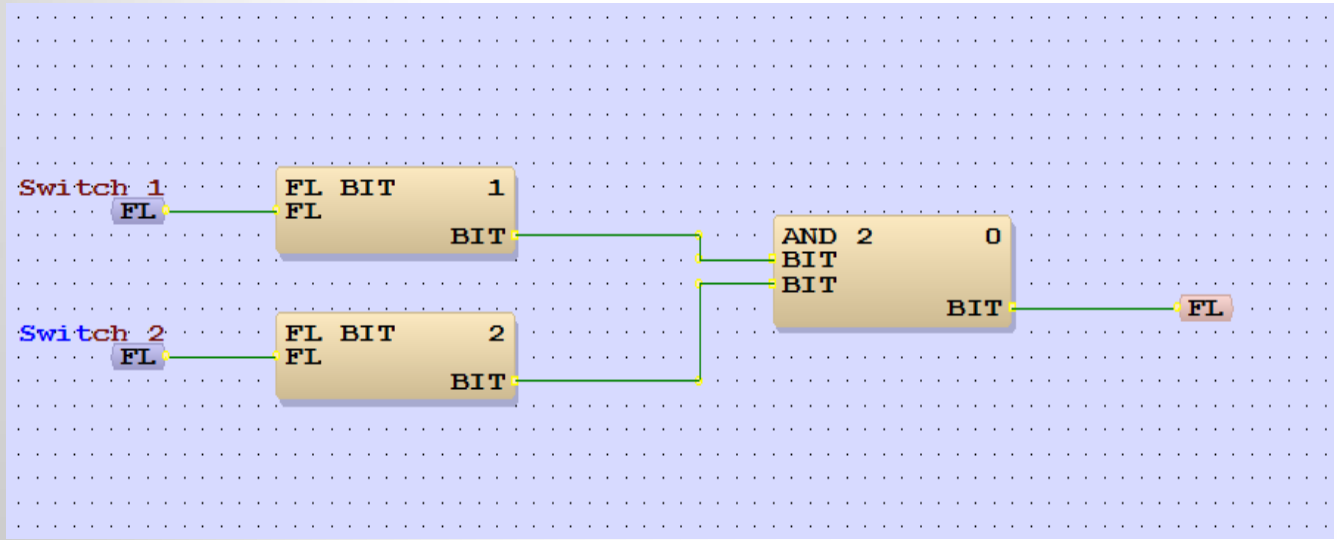
CAN Bus Function Block Setup





Simple Block Programming



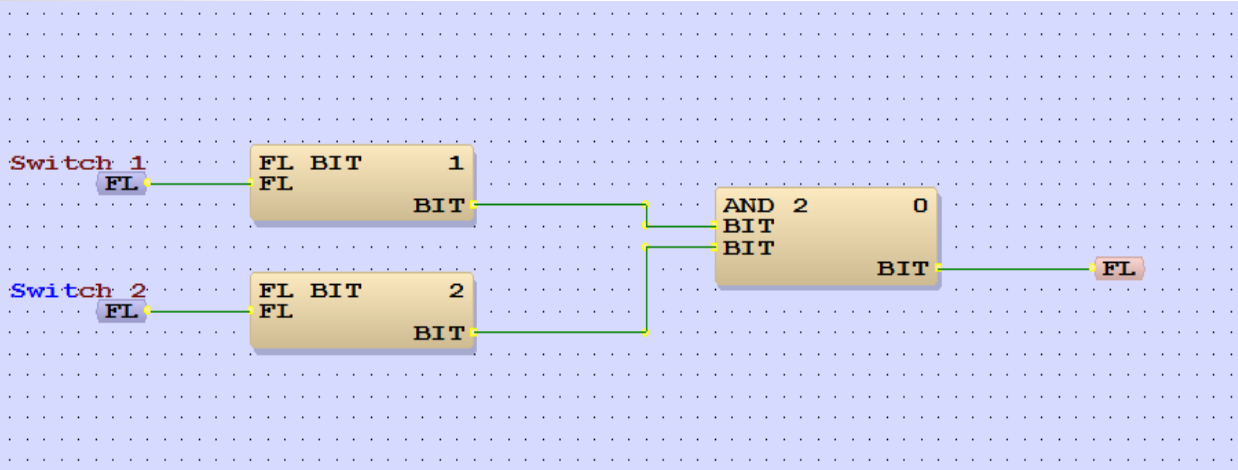


Tip:

To show the Input type, select **VIEW** and select **Show Type** and the I01 will show the Type of the input module you have selected.

This is a great tool to avoid a mixing type of constant and variables during the programming

Simple Block Programming

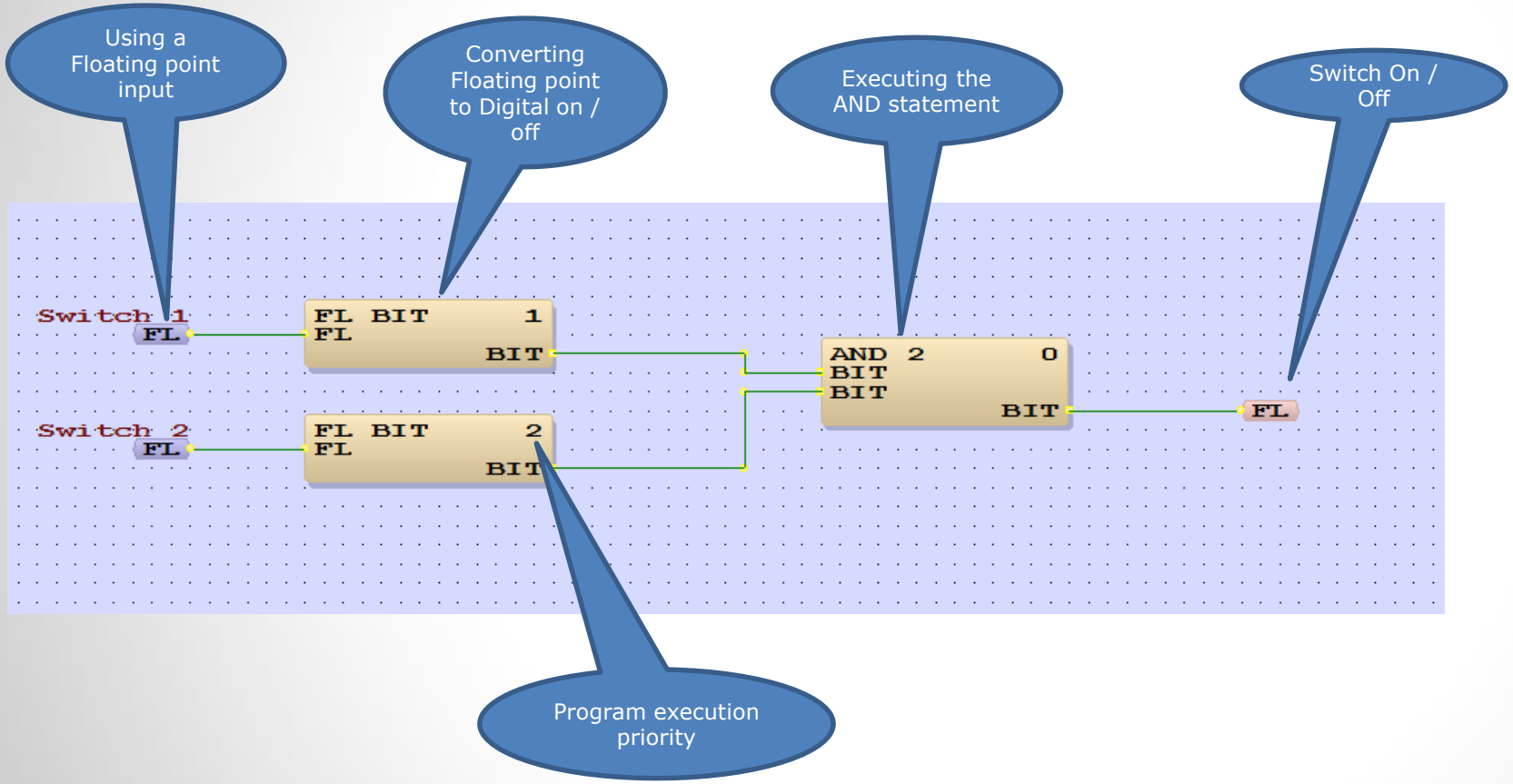


Tip:

To show the Input type, select VIEW and Select show and the I01 will show the Type of the input module you have selected.

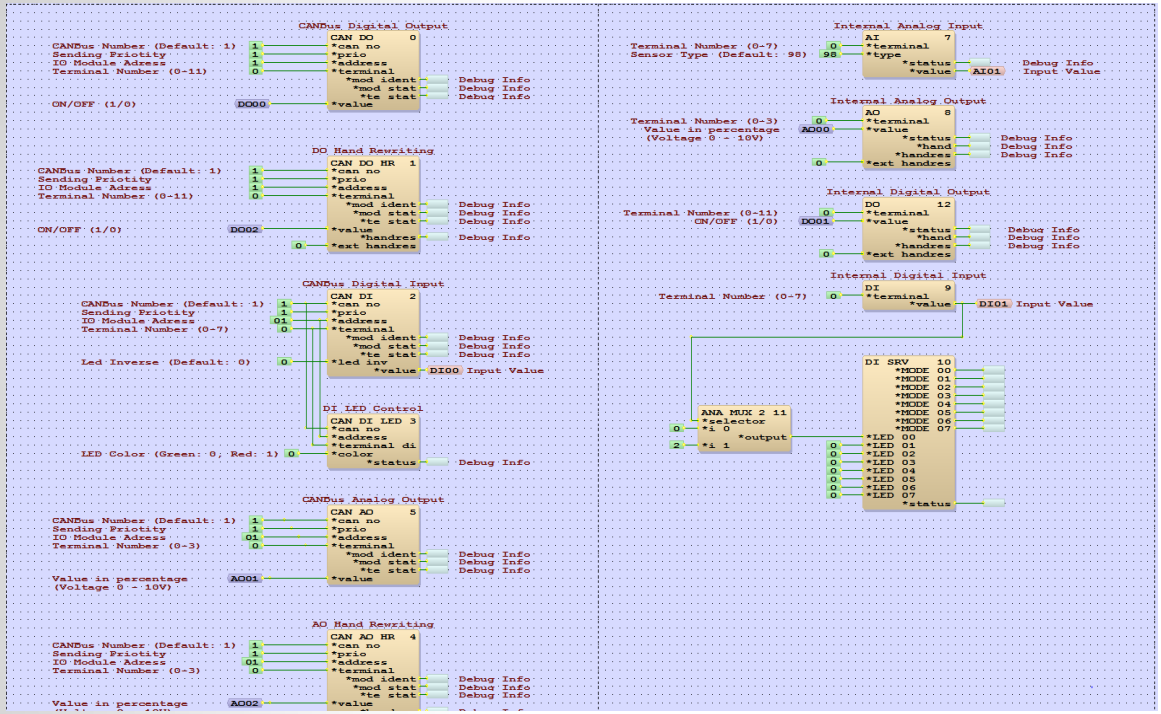
This is a great tool to avoid a using the wrong type of constant of variables during the programmin

Simple Block Programming



Tip:
Don’t mix Floating function blocks with Digital function blocks. You always have to match Digital to Digital and convert with function block Floating to Bit.

Exercise #2



Create Pre-set I/O & PID Blocks

Create your own Library of pre-set
Can-bus – I/O blocks
Pre-set PID loops
UI 8x8
DI 16
DO8

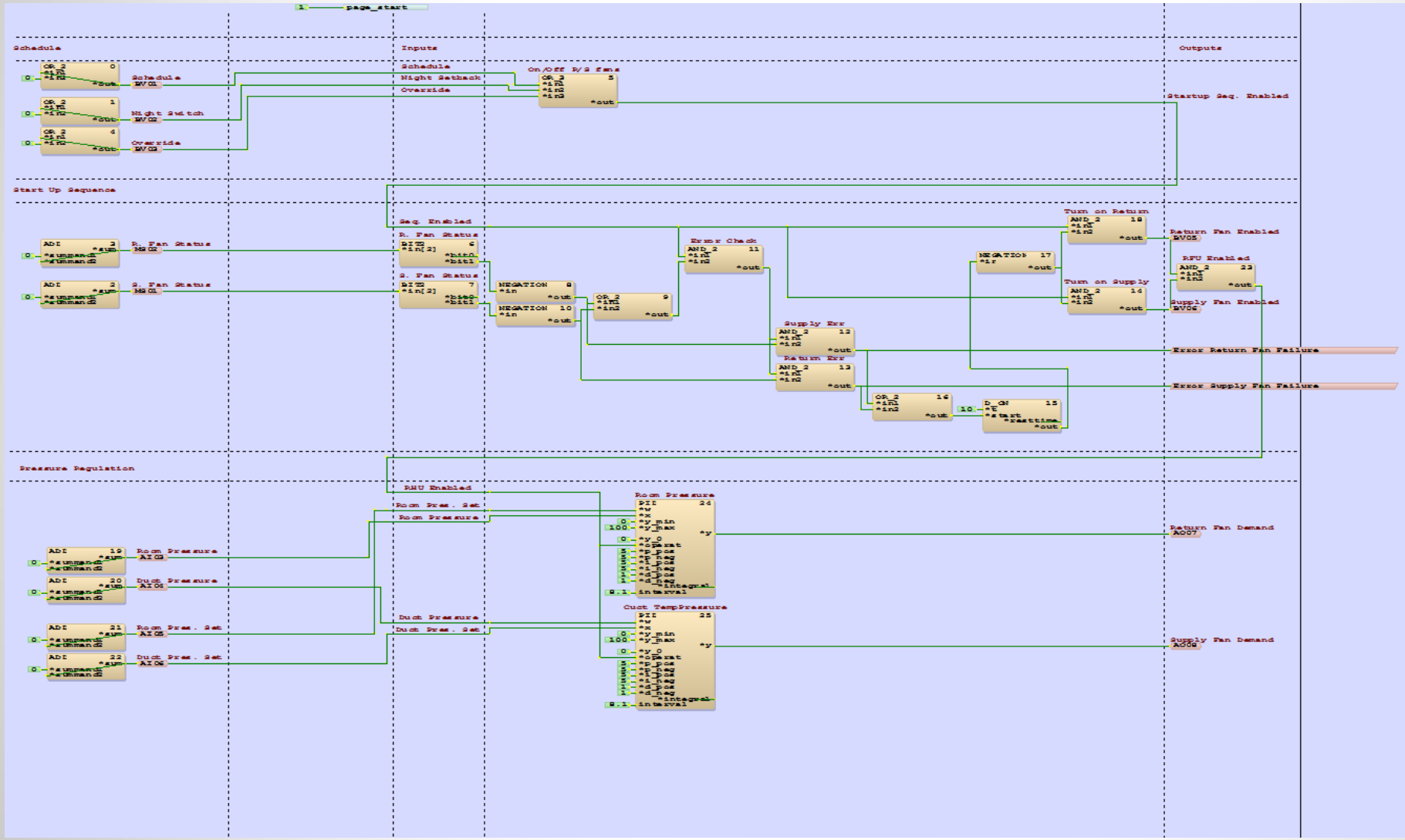
The exercise is to build up your library with module which we have testing in class.

Exercise #3

Create a AHU Program

Inputs			Outputs		
1	Night Set Back		1	RF-enable	
2	Override		2	SF-Enable	
3	RF-Status		3	RF-Speed	
4	SF-Status		4	SF-Speed	
5	Room Press.		5		
6	Room Press. SP		6		
7	Supply Duct Press.		7		
8	Supply Duct Press. SP		8		

Exercise #3

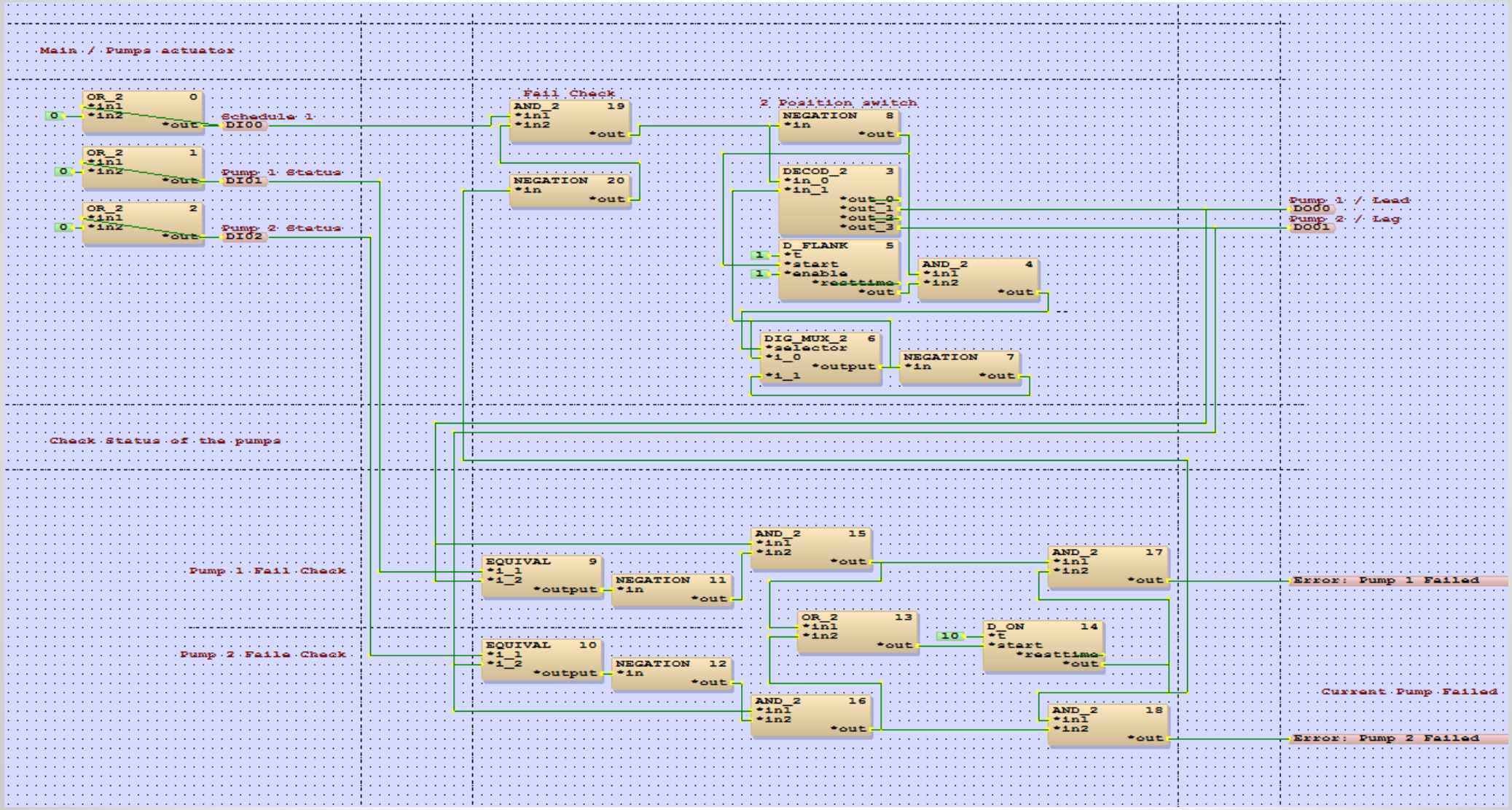

page_start
Schedule
Inputs
Outputs
Schedule
Night Setback
Override
On/Off R/S Fans
Startup Seq. Enabled
Start Up Sequence
Seq. Enabled
R. Fan Status
S. Fan Status
Error Check
Turn on Return
Turn on Supply
Return Fan Enabled
RFU Enabled
Supply Fan Enabled
Supply Err
Return Err
Error Return Fan Failure
Error Supply Fan Failure
Pressure Regulation
RHU Enabled
Room Pressure
Room Pressure
Duct Pressure
Duct Pressure
Return Fan Demand
Supply Fan Demand

Exercise #4

Create a Lead Lag Program

Inputs			Outputs		
1	Pump 1 Status		1	Pump 1 Enable	
2	Pumpt 2 Status		2	Pump 2 enable	
3	Pump 1 Fail		3		
4	Pump 2 Fail		4		
5	Supply Water Temp		5	Heating Valve Control	
6	Return Water Temp.		6		
7	Supply Water Setpint		7		
8			8		

Exercise #4


Main / Pumps actuator
OR_2 0
*in1
*in2
*out
Schedule 1
DI00
OR_2 1
Pump 1 Status
DI01
OR_2 2
Pump 2 Status
DI02
Fail Check
AND_2 19
NEGATION 20
2 Position switch
NEGATION 8
DECOD_2 3
*in_0
*in_1
*out_0
*out_1
*out_2
*out_3
D_FLANK 5
*t
*start
*enable
*resttime
AND_2 4
DIG_MUX_2 6
*selector
*i_0
*i_1
*output
NEGATION 7
Pump 1 / Lead
DO00
Pump 2 / Lag
DO01
Check Status of the pumps
Pump 1 Fail Check
EQUIVAL 9
NEGATION 11
AND_2 15
AND_2 17
Error: Pump 1 Failed
Pump 2 Fail Check
EQUIVAL 10
NEGATION 12
OR_2 13
D_ON 14
AND_2 16
AND_2 18
Current Pump Failed
Error: Pump 2 Failed

Exercise #5

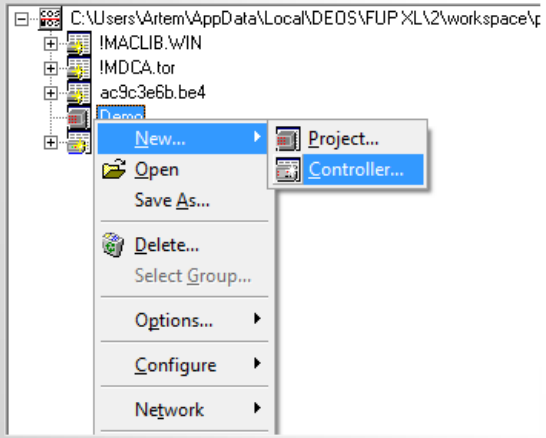
Create a functional Library with applications for your own library

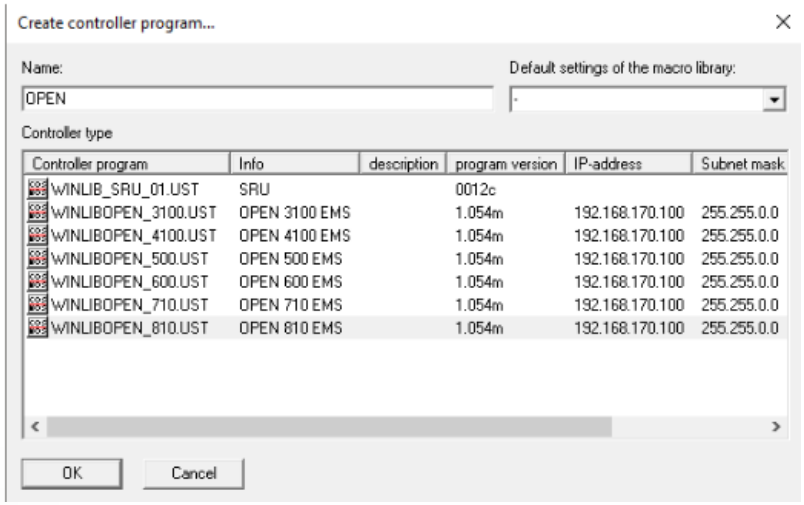
Simple Controls Loops		
Lead Lag Program		
Start up Sequnce		
Damper Controls		
Heating & Cooling valve controls		
Boiler Control (with 5 open flame boilers)		
Chiller Start up and Control sequence		

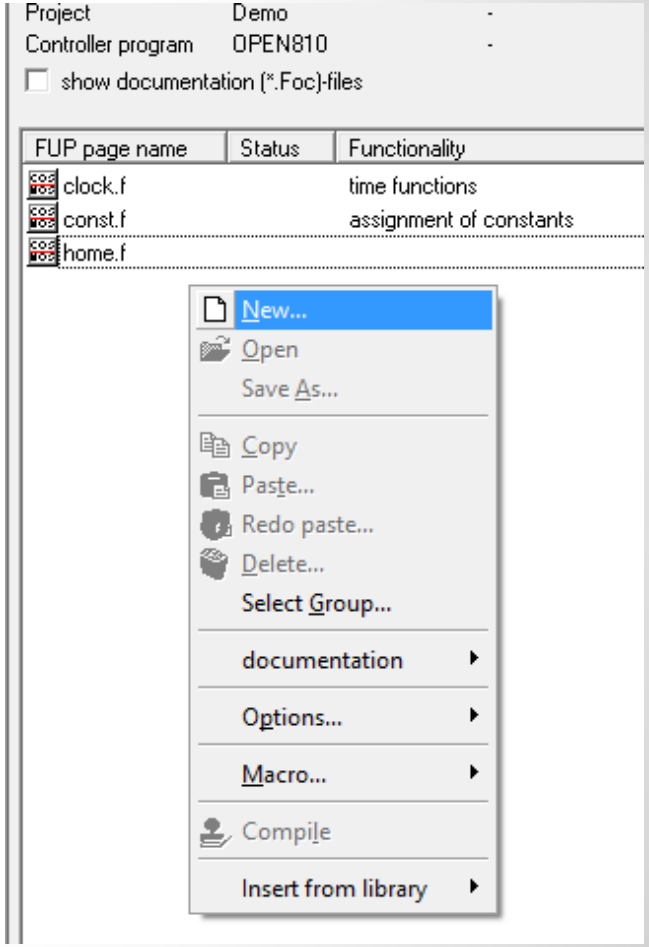
FXL Graphics

Create New Project and Controller Add new programming Page.

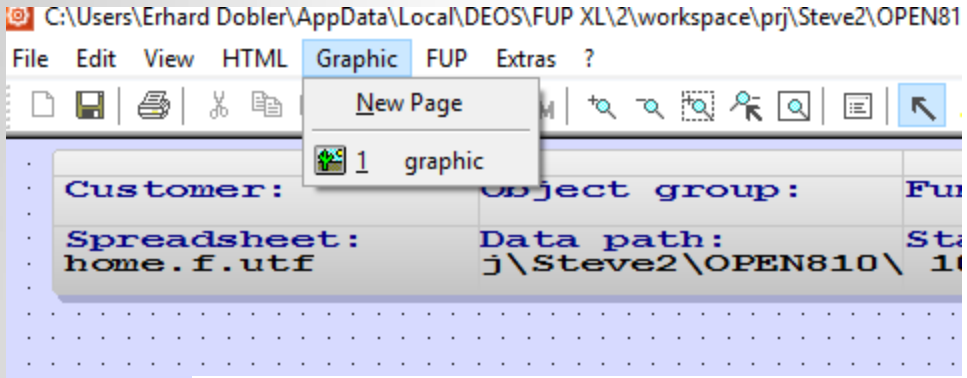
Tip – You can import any other pictures or Visio (Auto Cad) floor plan Graphics.

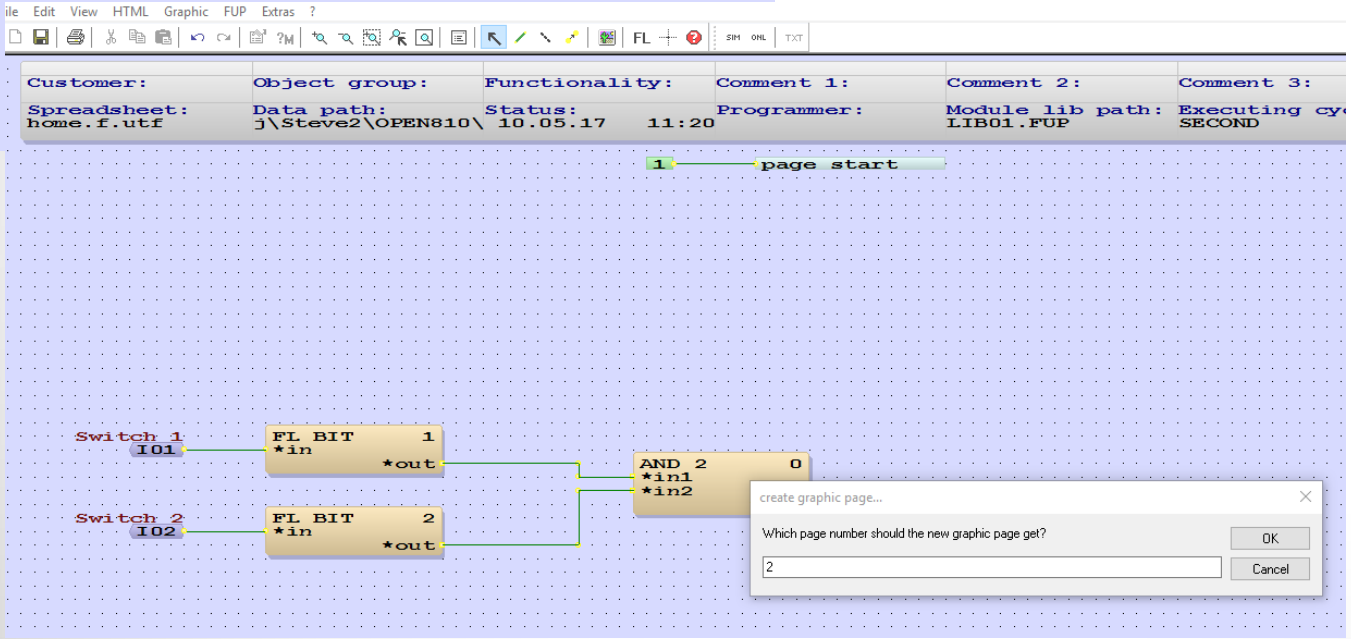






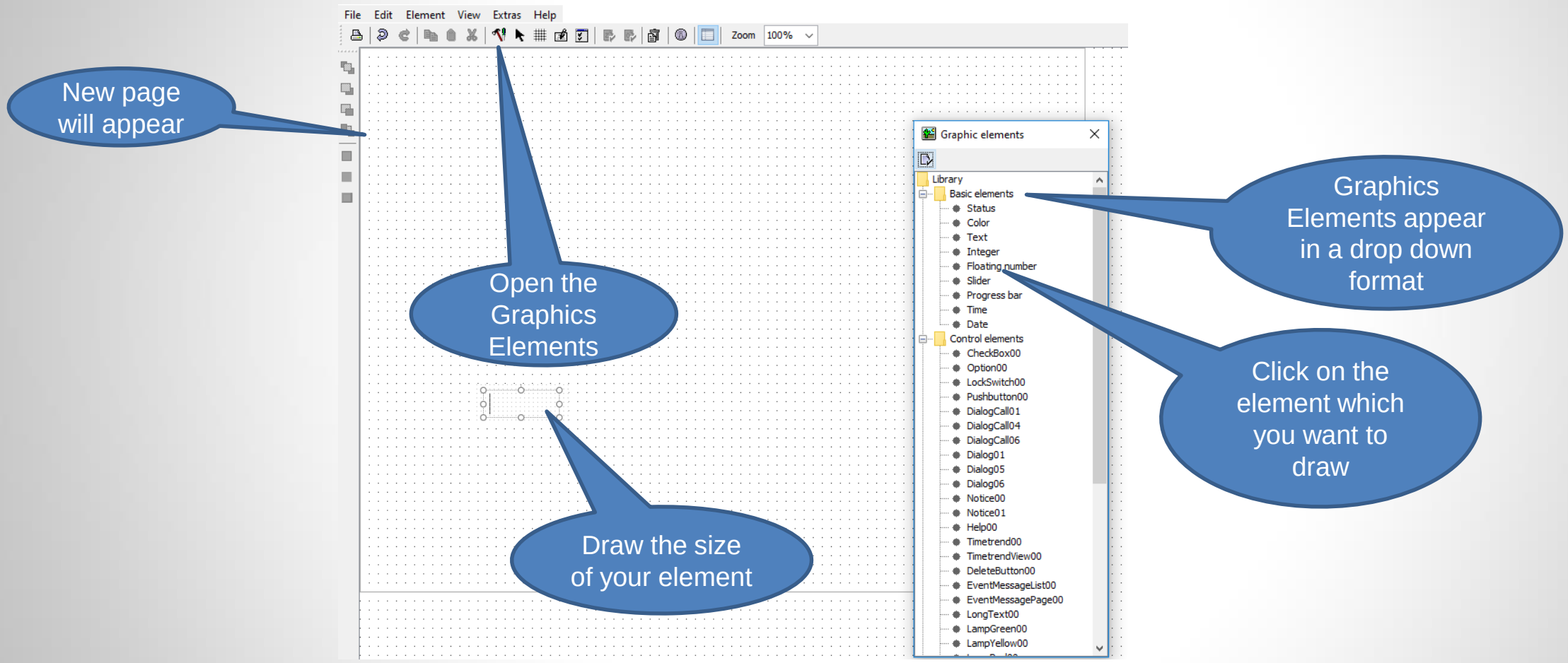
Creating a new Graphic Page



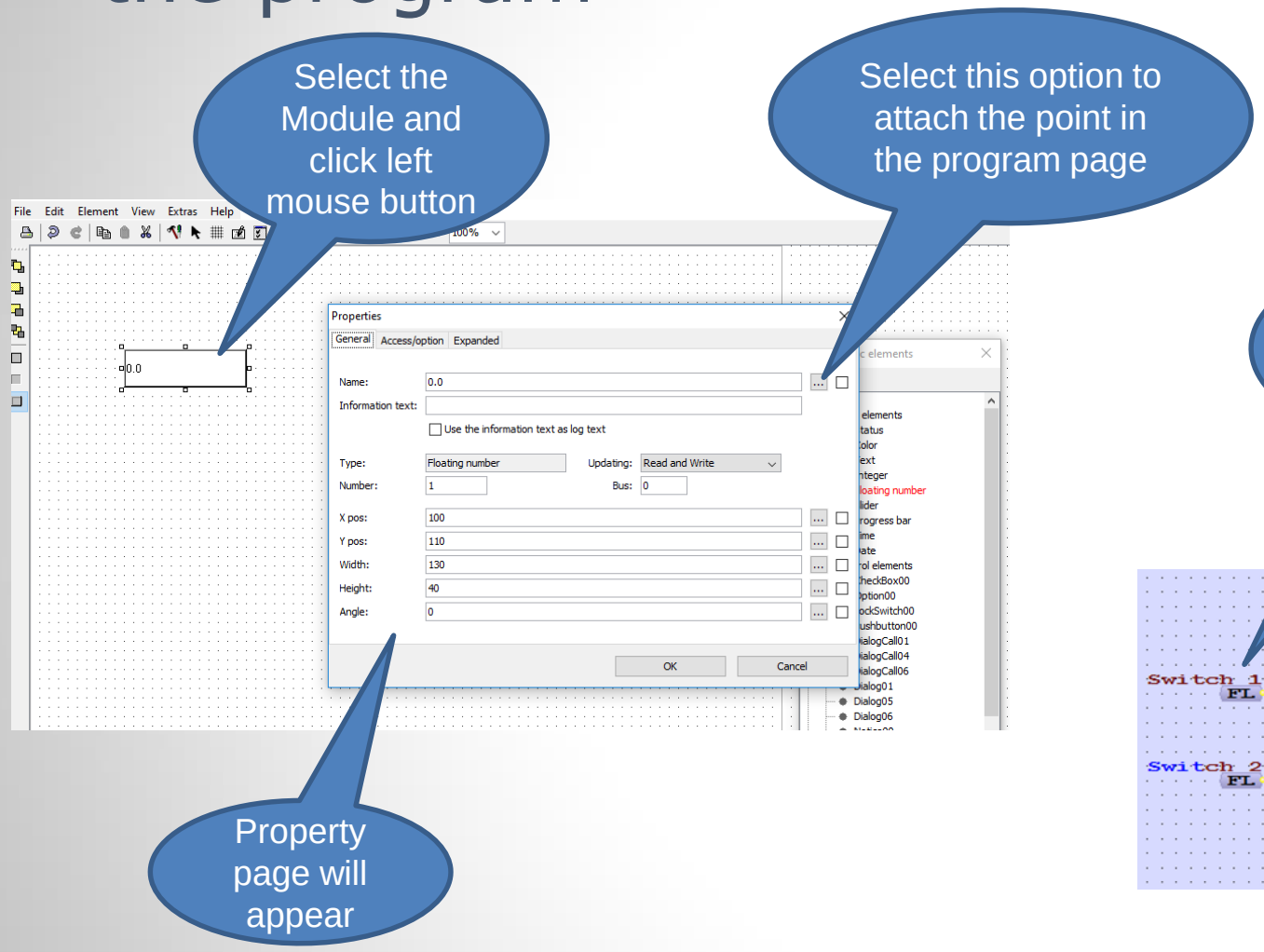


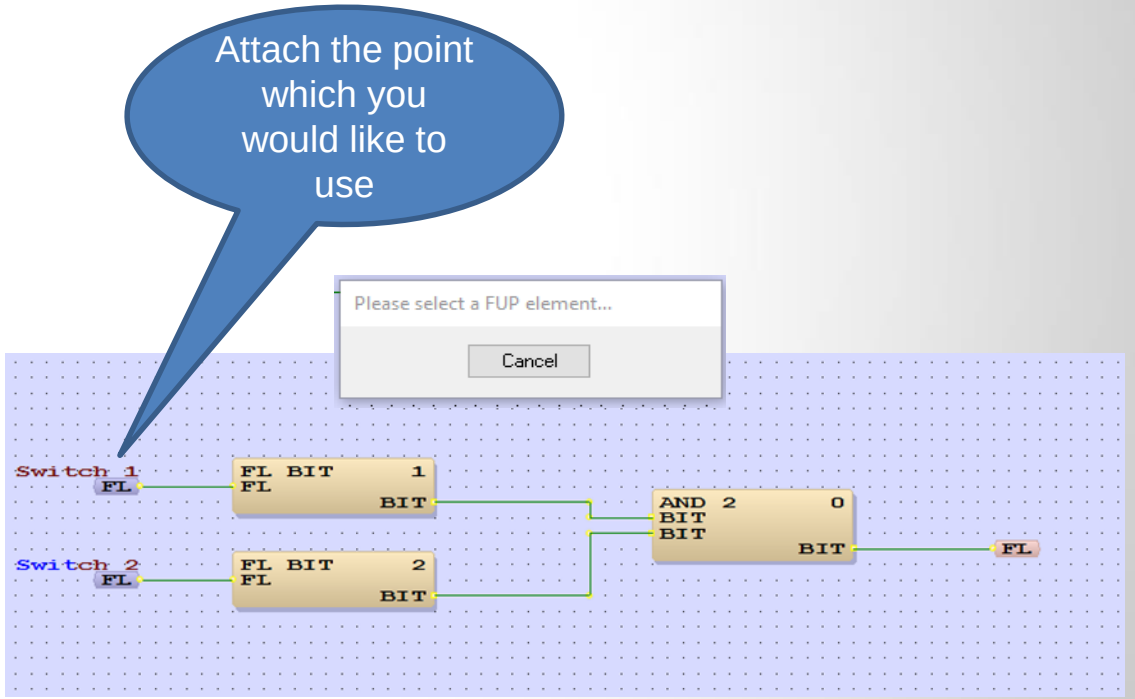
1. Open program page
2. Select Graphics in the Main Menu
3. New graphical page will be created

DEOS Controls – Products

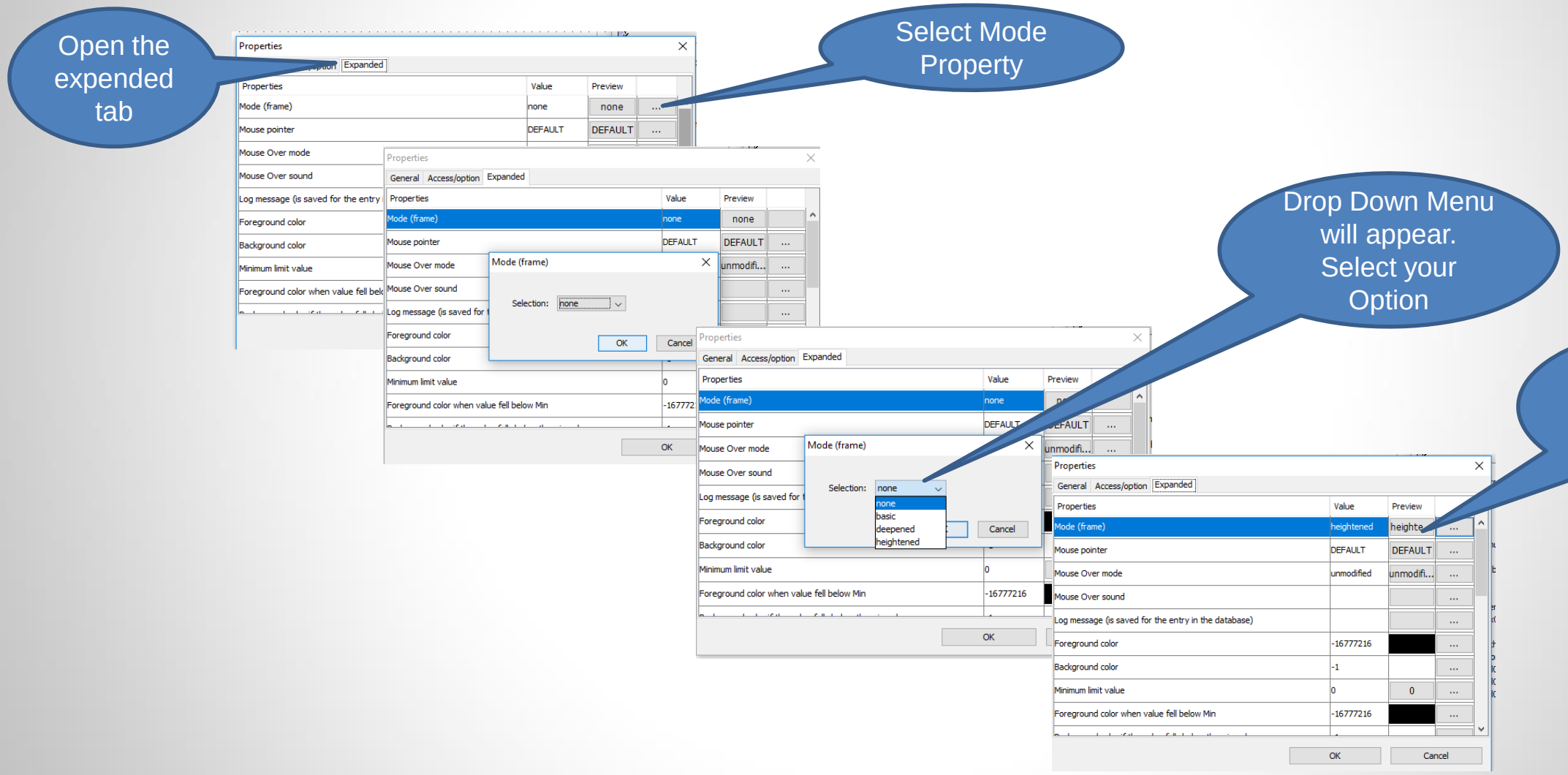


Binding Graphic elements to Points in the program



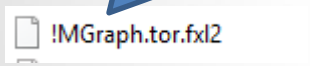


Configure Graphic elements visual effect

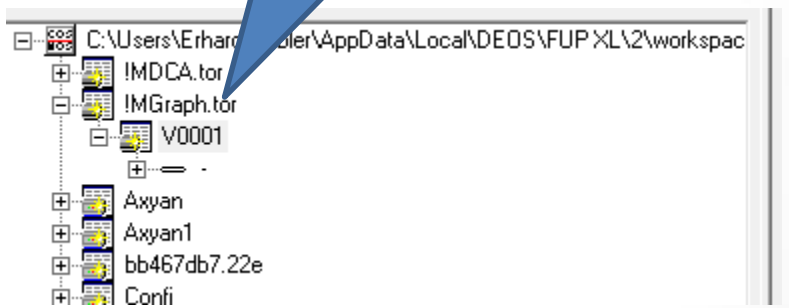


Working with Graphics Library

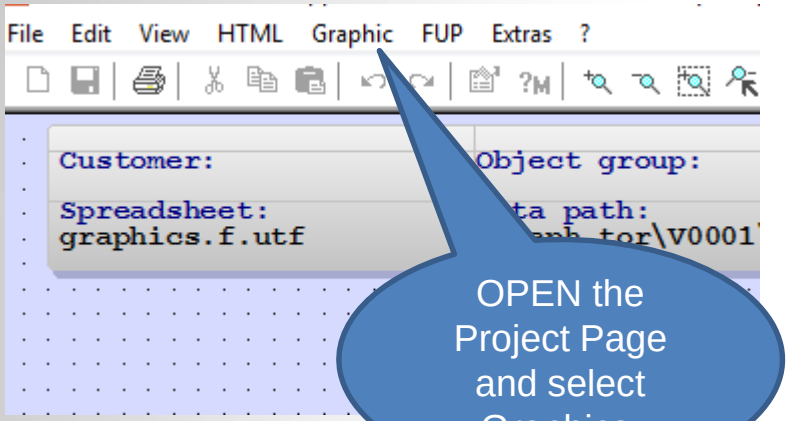
Download the Master Graph Library from the DEOS Controls Partner Section.



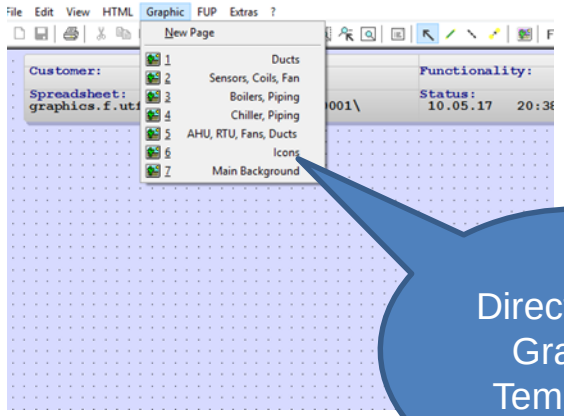
Import into your Project Directory



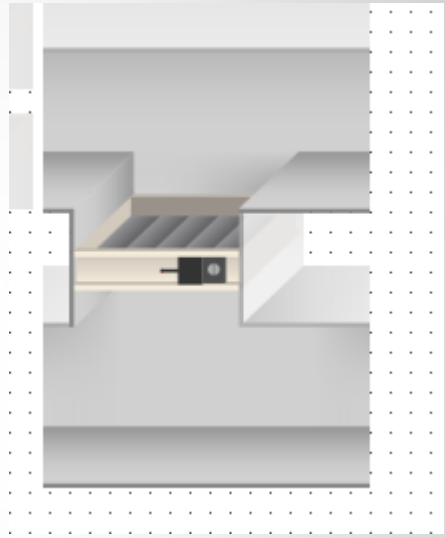
Working with Graphics Library

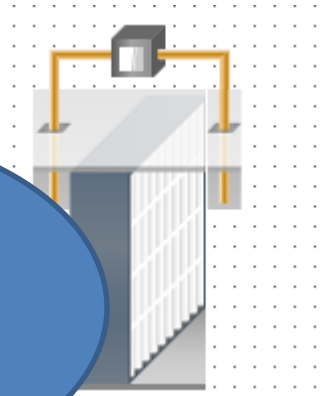


OPEN the Project Page and select Graphics.

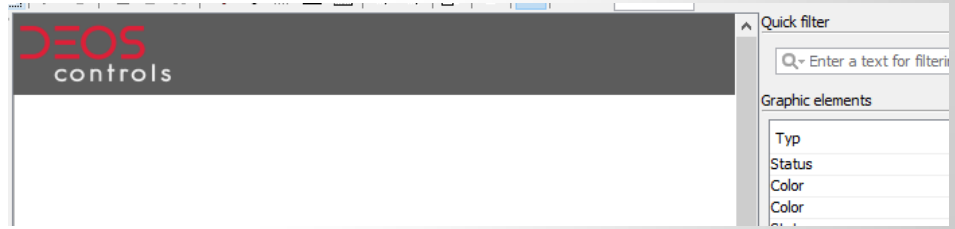


Directory for Graphic Templates





You can copy and past the templates your project and bind the points to your program

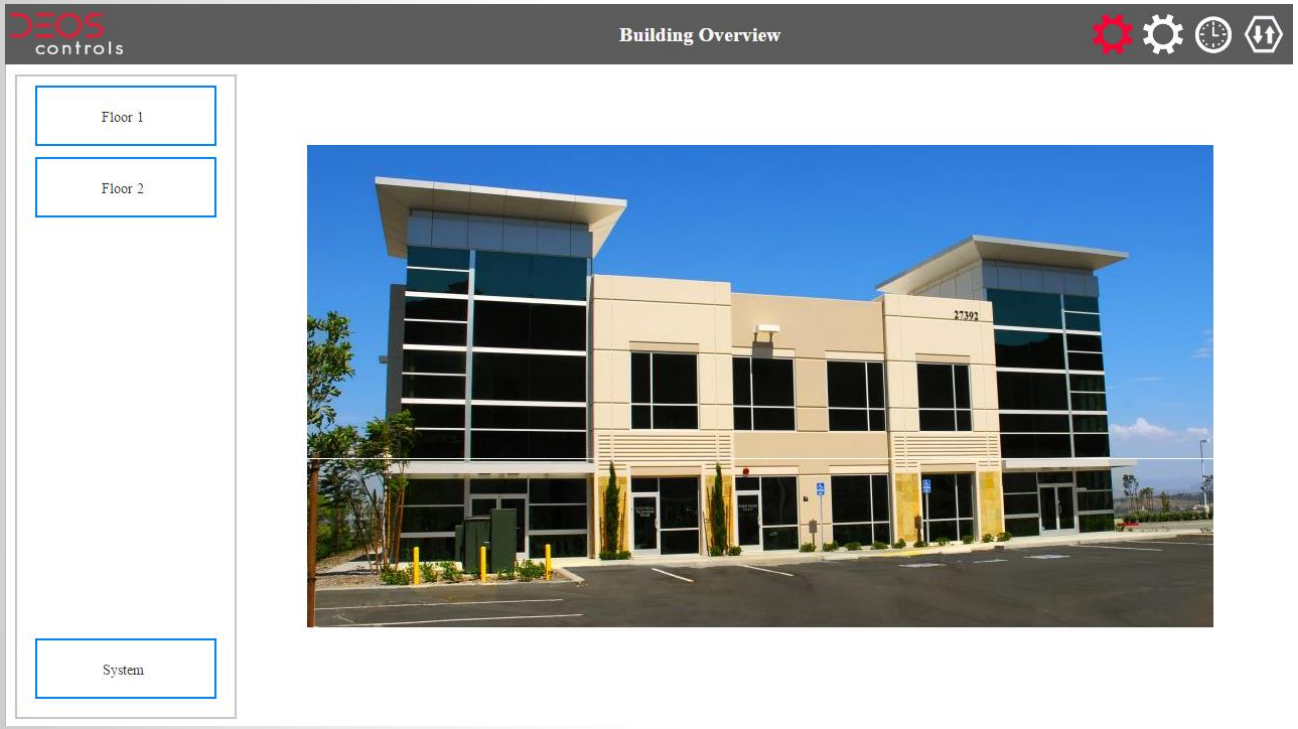


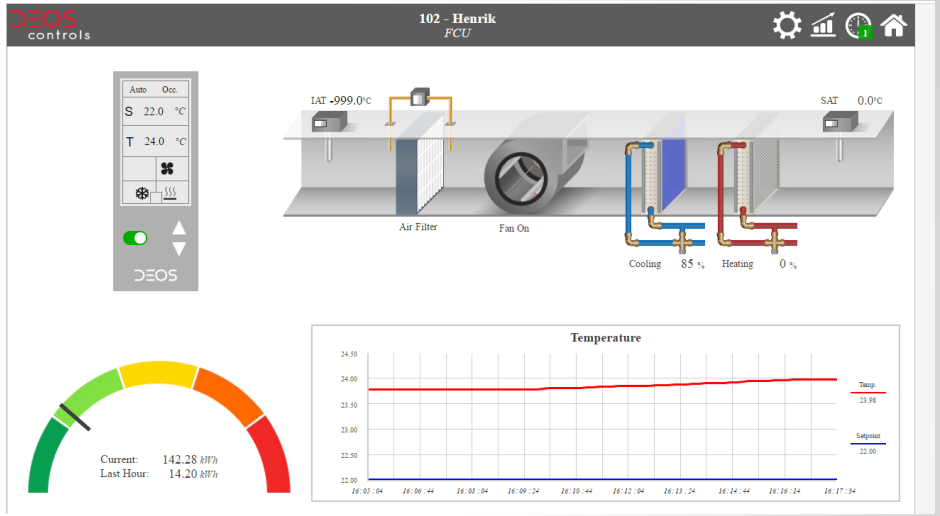
Exercise #6

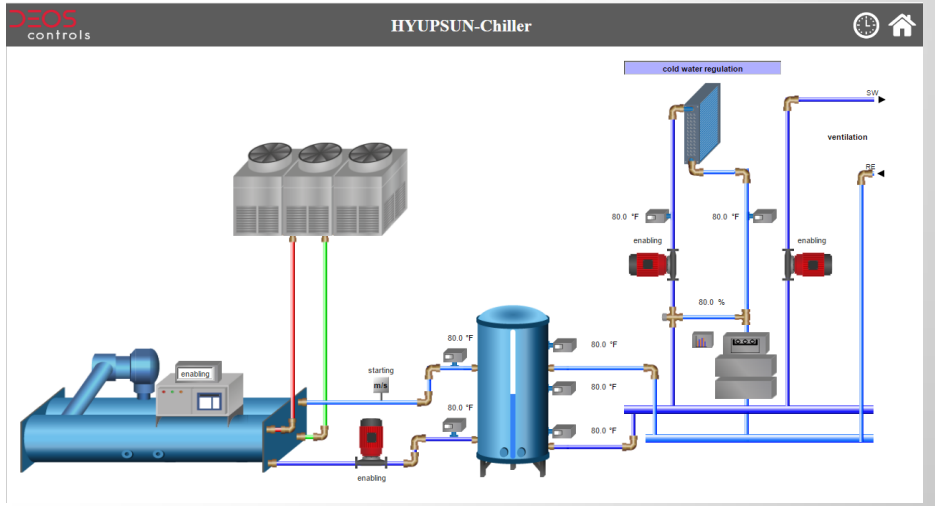
Create Graphics for our Trainings Project

"Start Page" with Building Picture and Labels for the Floor Levels
Single "Floor Level" with Room's and Labels for Temperature Sensors and VAV links to VAV Graphic
Create "Navigation" function so you can navigate though your project

Project Example

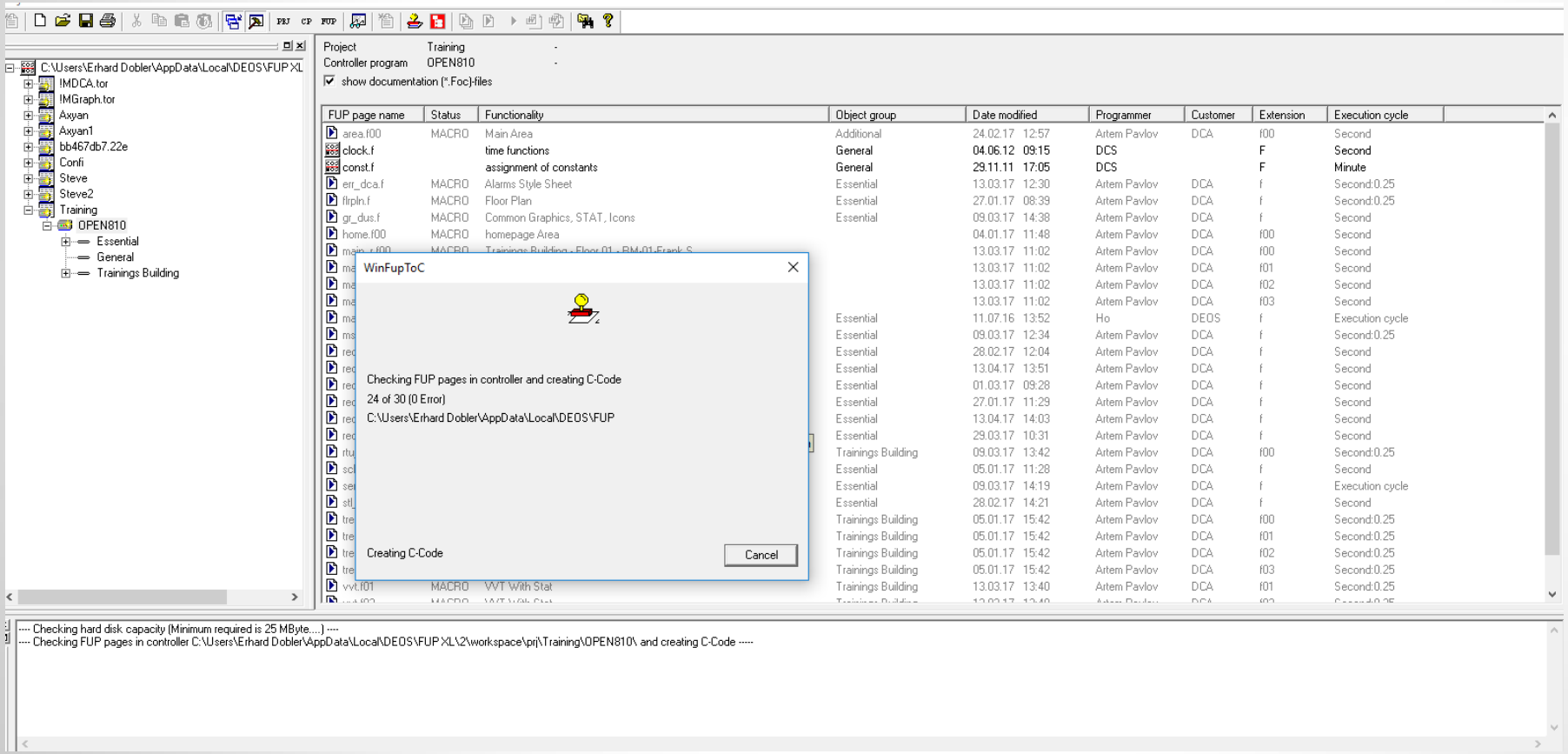




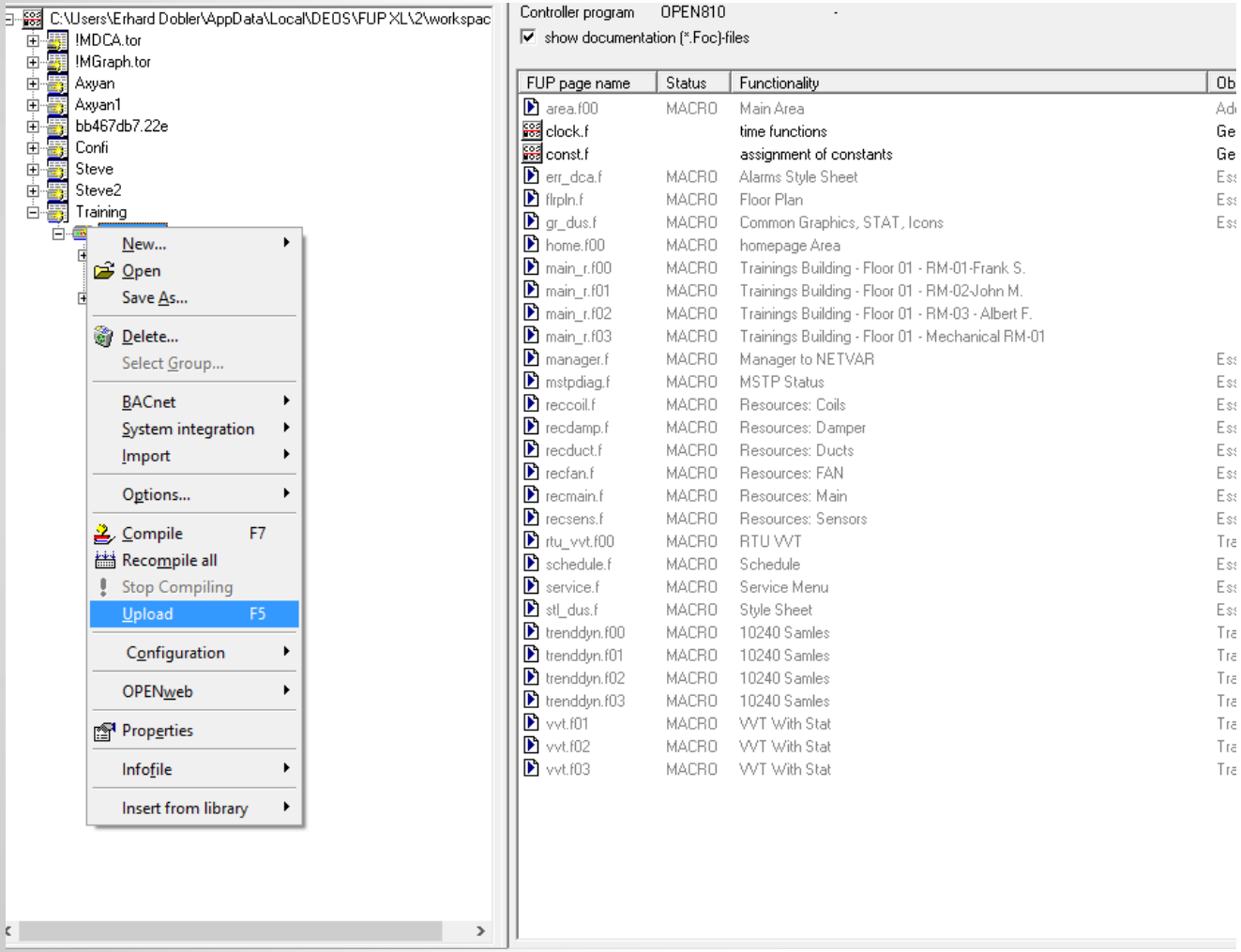


Compile & upload a Project

- Right Click on your project in the tree and choose “Compile”

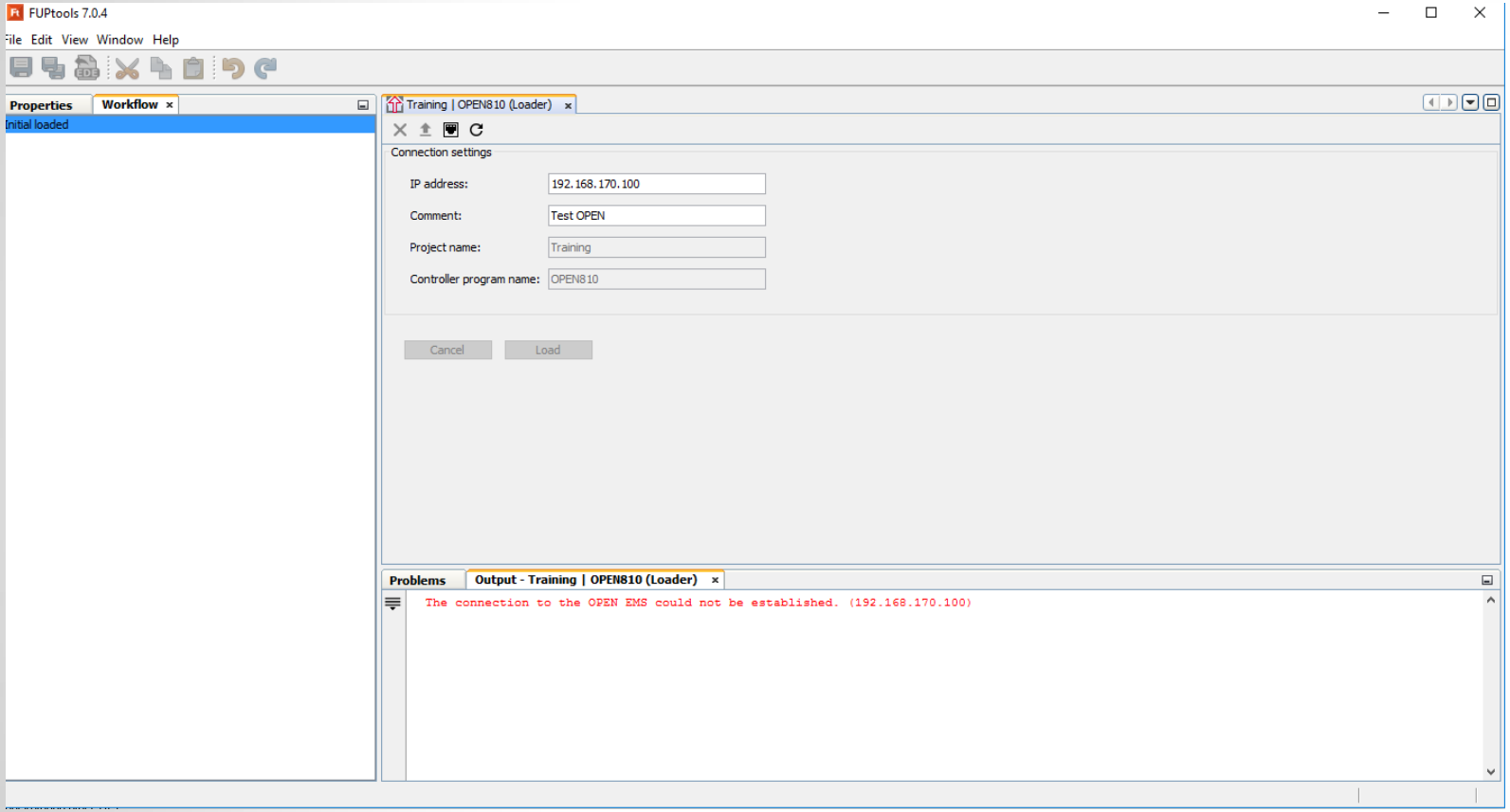


Compile & upload a Project



- Right Click on your project in the tree and choose “Upload”
- Make sure, your computer is connected to the OPEN controller you with to load

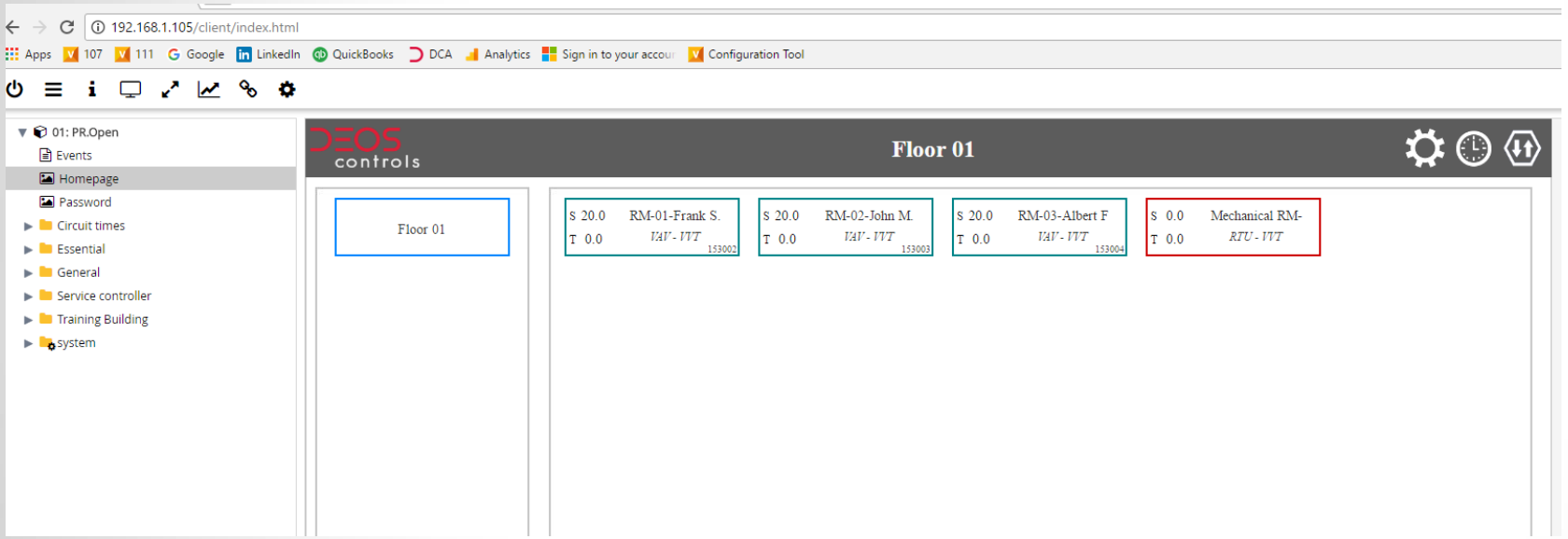
Compile & upload a Project



- Enter the IP address of the connected OPEN controller
- Click „Load“

Compile & upload a Project

- Enter the controller‘s IP address in the address bar of your browser and navigate through the graphical user interface



- Questions