pbsHMI Concepts

Version: 4.1

Kamjoo Bayat – kb@pbscontrol.com

Update Date: June 2025

1 – Introduction

pbsHMI is a SCADA/HMI platform based on Dot Net X64 bits technology. To use pbsHMI you need to read this document.

pbsHMI supports following functionalities :

- Object oriented graphic pages and symbols with different dynamics and events
- Modbus RTU/TCP Master Driver
- DNP3 Master Driver over TCP and RS232 with IEC62351
- IEC870-5-104 Master Driver with IEC62351
- MQTT Driver
- S7 Driver
- Vestas Wind turbine VMP Protocol.
- Beckhoff ADS Protocol.
- OPC UA Client /Server
- Alarm /Event Management
- C# Scripting
- Scheduling
- SQL Server Data logging
- Offline and Online Trending
- Different type of dynamics for graphic objects
- Reusable graphics symbols
- Web Server function
- Power grid network tools for simulation and estimation
- Machine learning modules for forecasting , anomaly detection , network learning
- Smart Battery management module

pbsHMI is designed for rapid development of HMI/SCADA applications. You can easily create graphical pages and connect them to external signals.

pbsHMI requires the following software components to function properly:

- Microsoft Dot Net Framework 4.7.2 Runtime , 64 Bit
- Microsoft SQL Server for data logging on SQL Server
- Visual C Runtime. you can find it in Utility Directory of pbsHMI
- Microsoft Office for data logging on Access database and generating excel reports

You can download pbsHMI from www.pbscontrol.com

Download and unzip it to any folder you want.

You can find following files in the pbsHMI folder:

- pbsHMIEditor.exe : development environment of pbsHMI . All configurations will handle by pbsHMIEditor application .

- pbsHMIView.exe : runtime version of pbsHMI

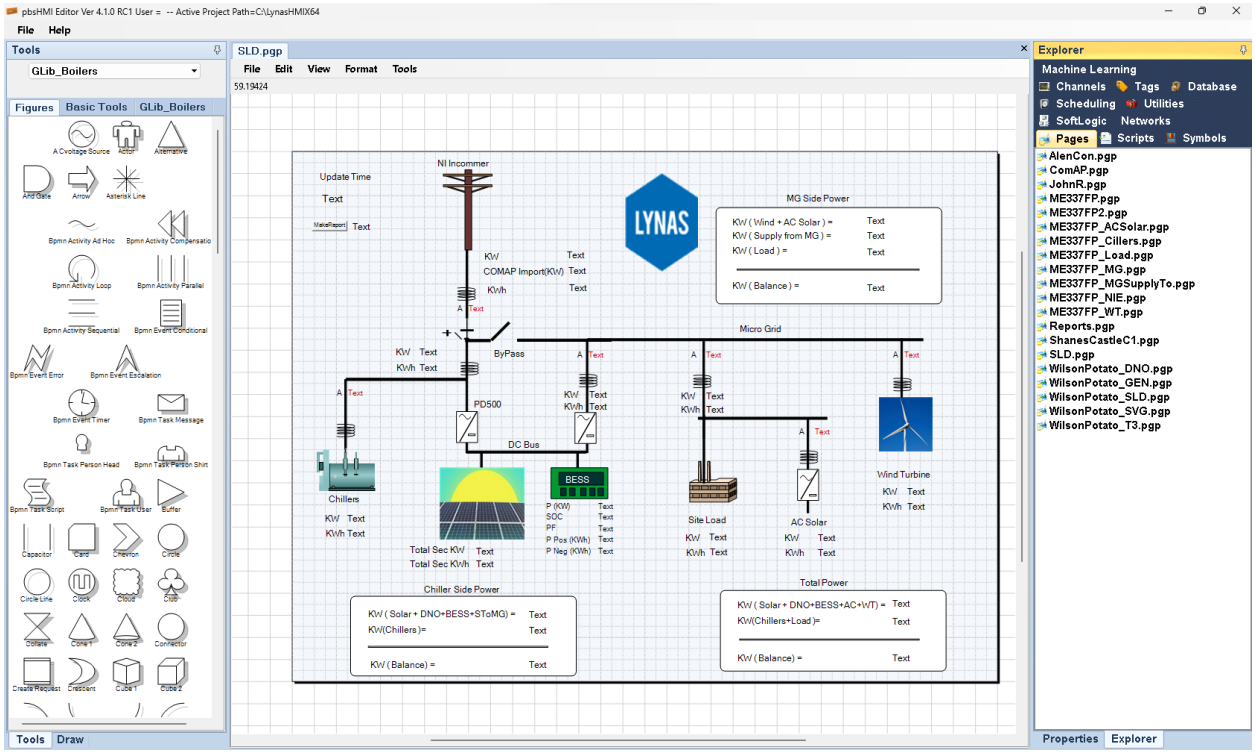
- pbsHMITrend.exe : Historical Trend Utility .
- Option.xml : option file of pbsHMI
- GLIB_x.dll Symbole definition file . pbsHMIEditor and pbsHMIview will load dynamically all GLIB_x.dll file and startup . You can compile new GLIB_X.dll symbol file by symbol Editor but it will not affect on system until you restart pbsHMI View.
- All Other files are system files and must be in pbsHMI Directory.

You can define multiple projects and select one of them as the active project.

You can run multiple instances of the pbsHMI IDE and runtime simultaneously with different active projects.

2 – Basic concepts, Development and Runtime Modules

pbsHMI has a development IDE and a core or runtime viewer. When you run pbsHMIEditor, you can see a screen like the following:



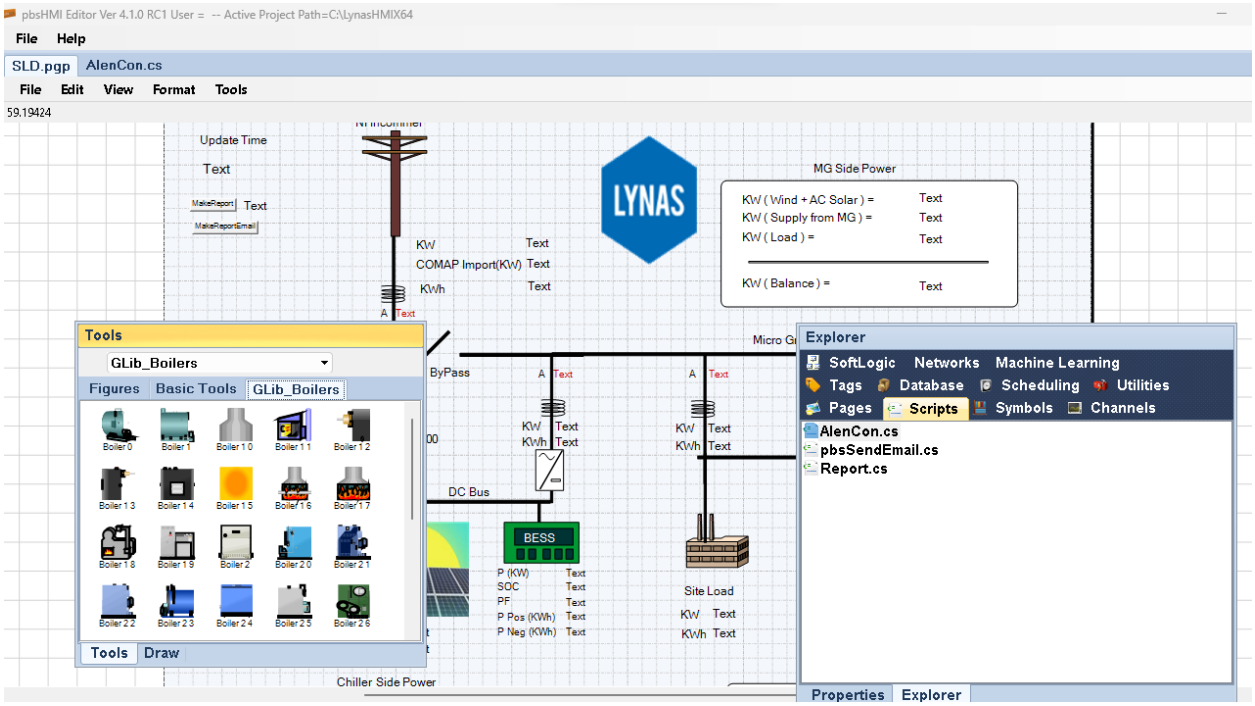
In the right panel, you can see the various pbsHMI modules as follows:

- Channels : pbsHMI Communication Drivers
- Tags : pbsHMI Device , Global , Alarm and event Tags
- Database : Database configuration
- Scheduling : running C# scripts by scheduling
- Utilities : Set Active project , runtime parameters , users
- SoftLogic : Write Function Block Programs
- Networks : define Power grid network , handle Power flow and state estimation function and run ML for network ,GNN
- Pages : Define Graphic pages
- Scripts : Writ C# scripts
- Symbols : Define new Graphic Symbol
- Machine Learning : Define ML Modules for forecasting , anomaly detection, Smart Batter management

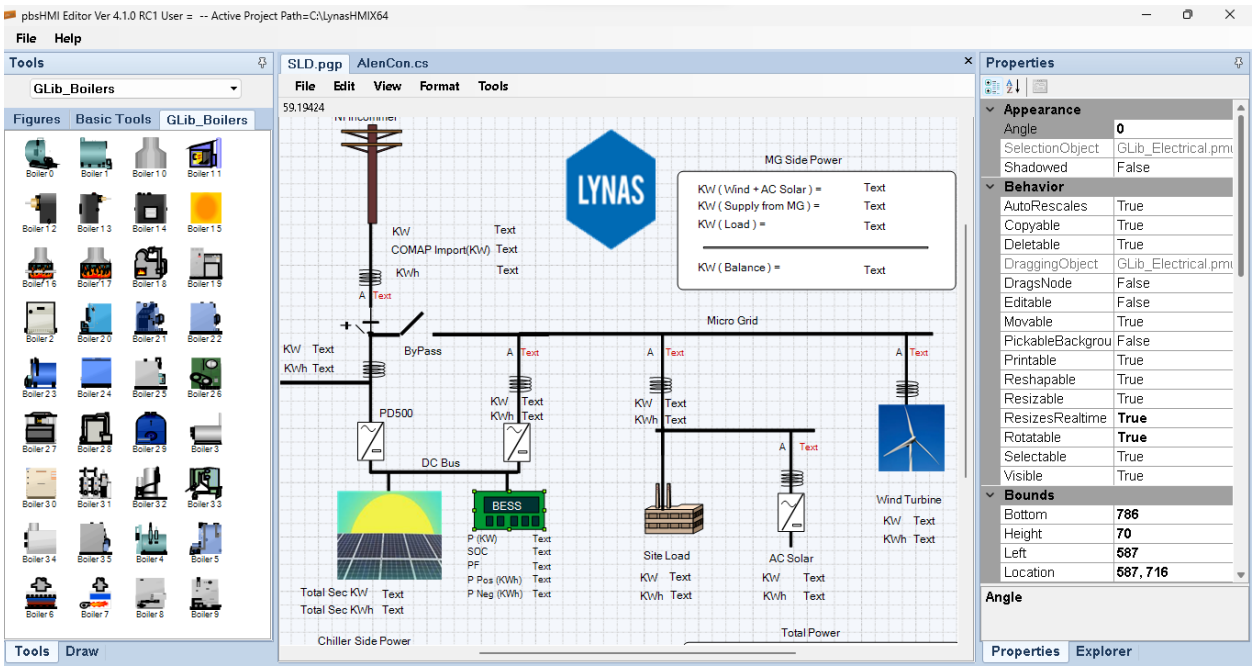
In the left panel you can see different tools like graphic symbols for pages or network elements for network module.

In the middle you can see the content like graphic page.

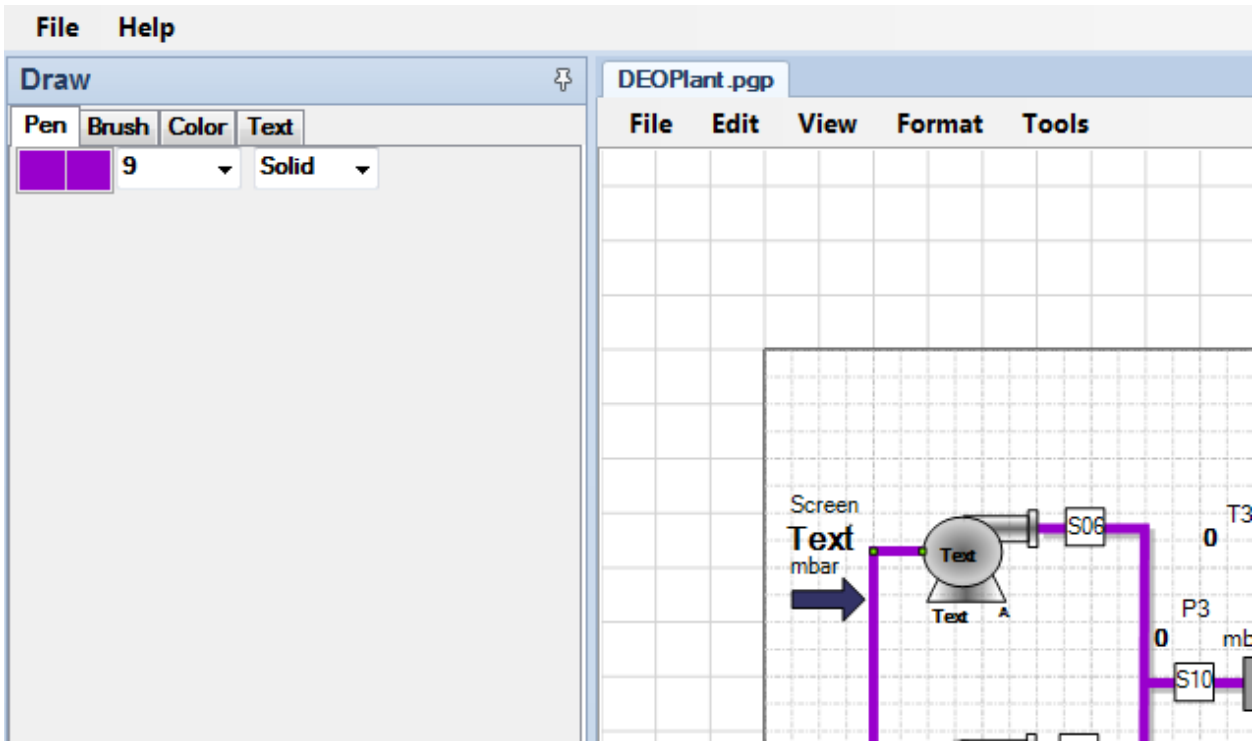
Explorer and tools windows are docking panels and you can set them as auto hide.
You can change Explorer and tools window as float panel. Click on top part of window and drag it.

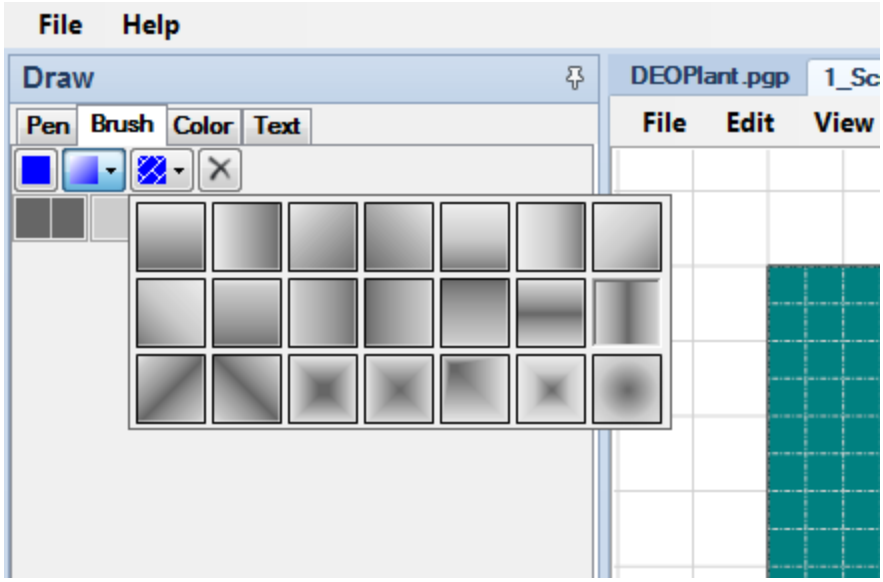


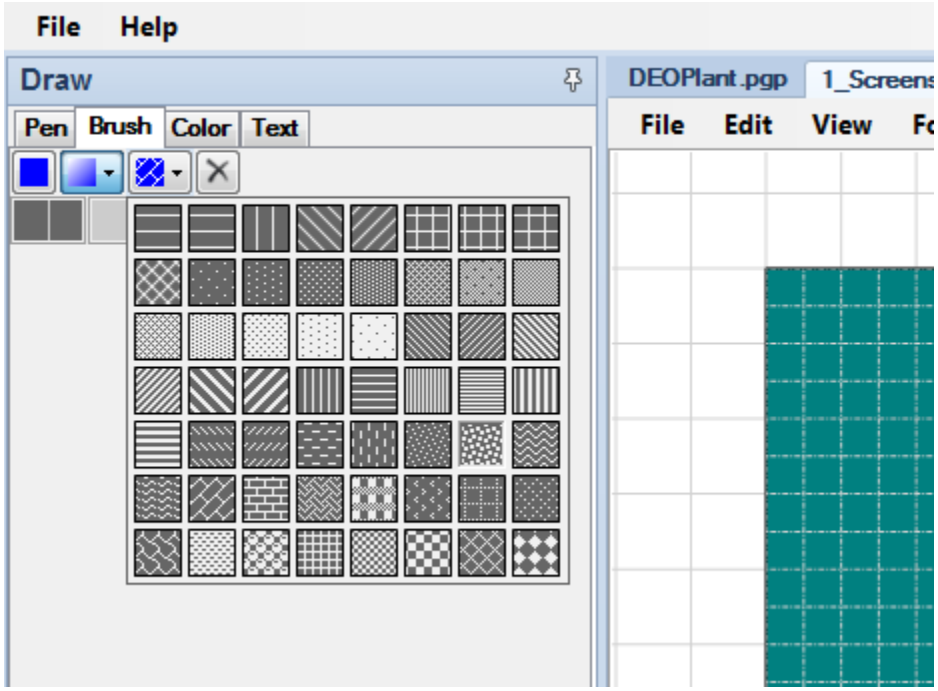
- Menu Item with bold font is default (Double Click) function. For all component default function is open component.
- “Explorer” menu item is for opening component directory. For deleting, renaming and copy/paste component you should use windows explorer.
- “Refresh” menu item will refresh component list.
- You can see Graphic symbols properties in properties tab of explorer window.

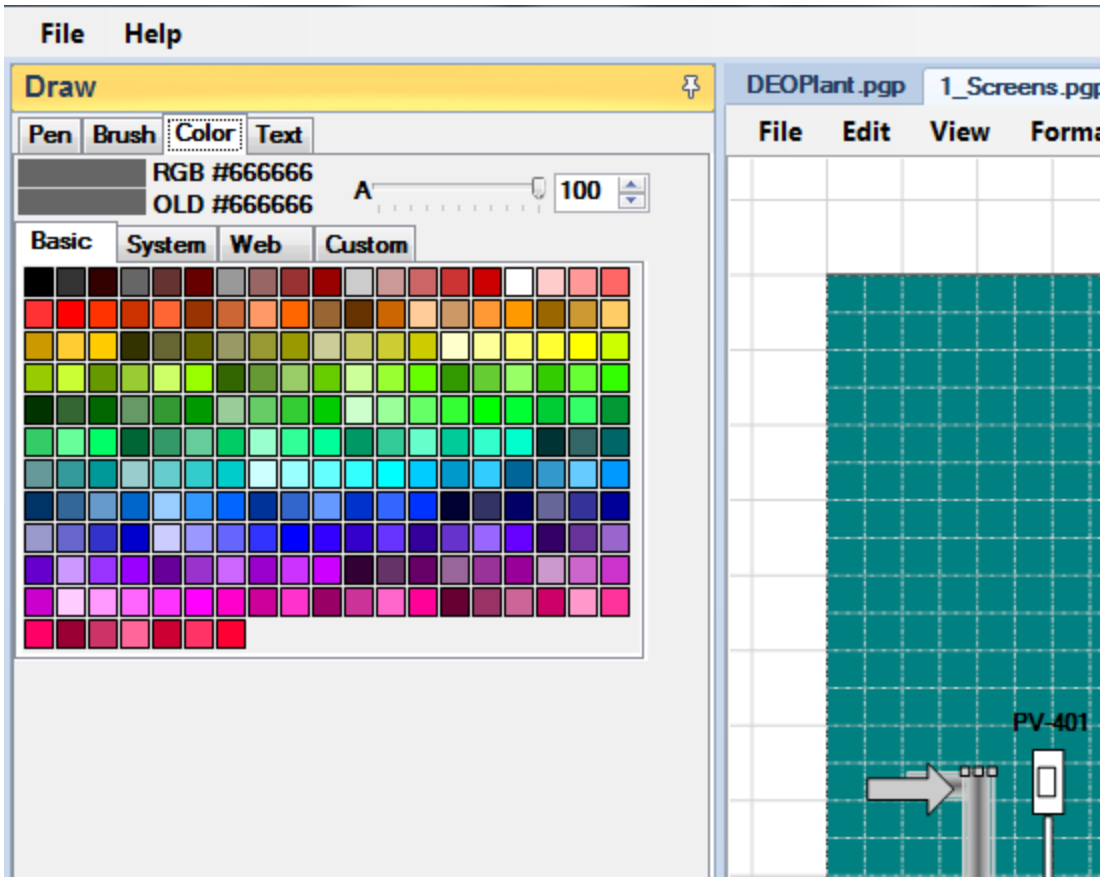


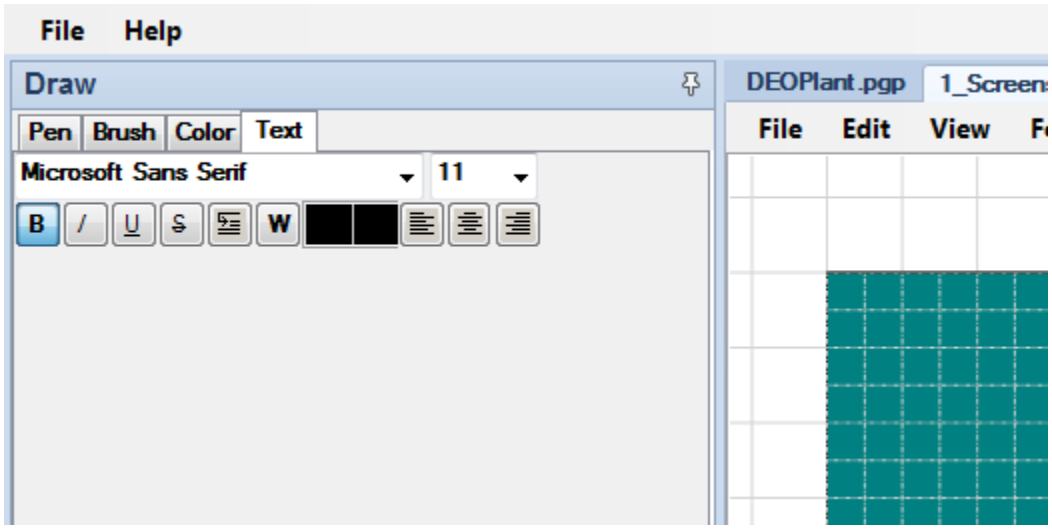
- All graphic dynamics and events are configured by properties window.
- In Draw Tab of Tools window, you can see Pen, Brush, Color and pen Tools for easy configuration of graphics symbols.



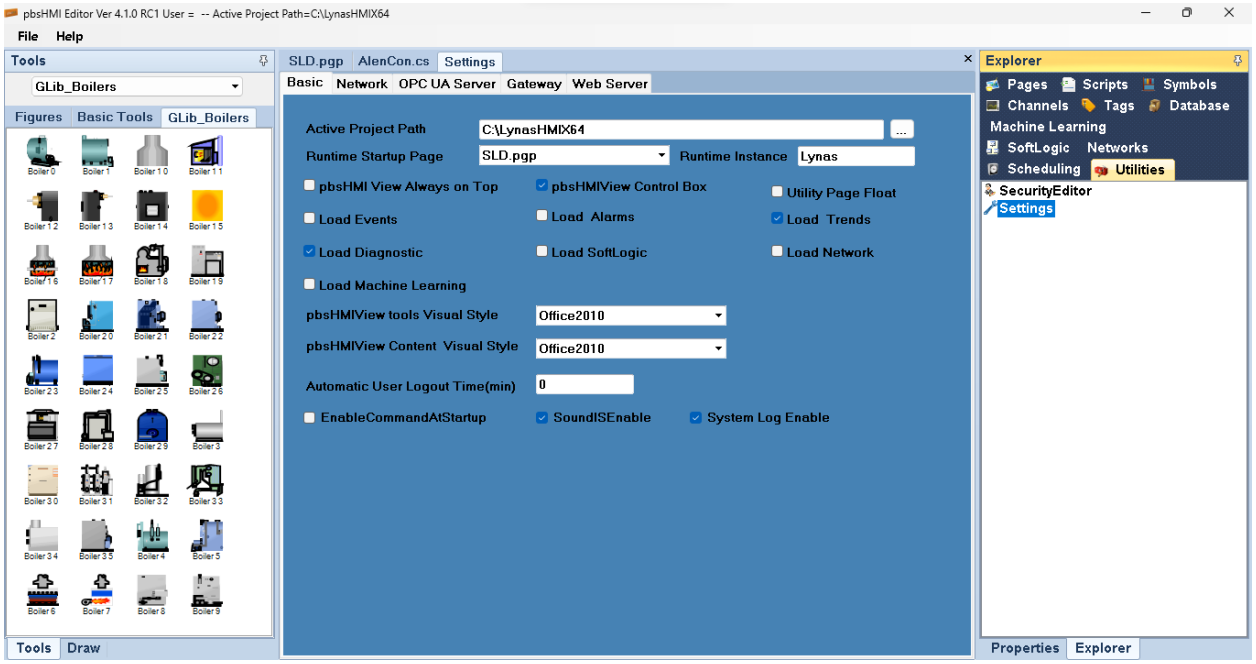

File
Help
Draw
Pen
Brush
Color
Text
DEOPlant.pgp
1_Sc
File
Edit
View


File
Help
Draw
Pen
Brush
Color
Text
DEOPlant.pgp
1_Screens
File
Edit
View

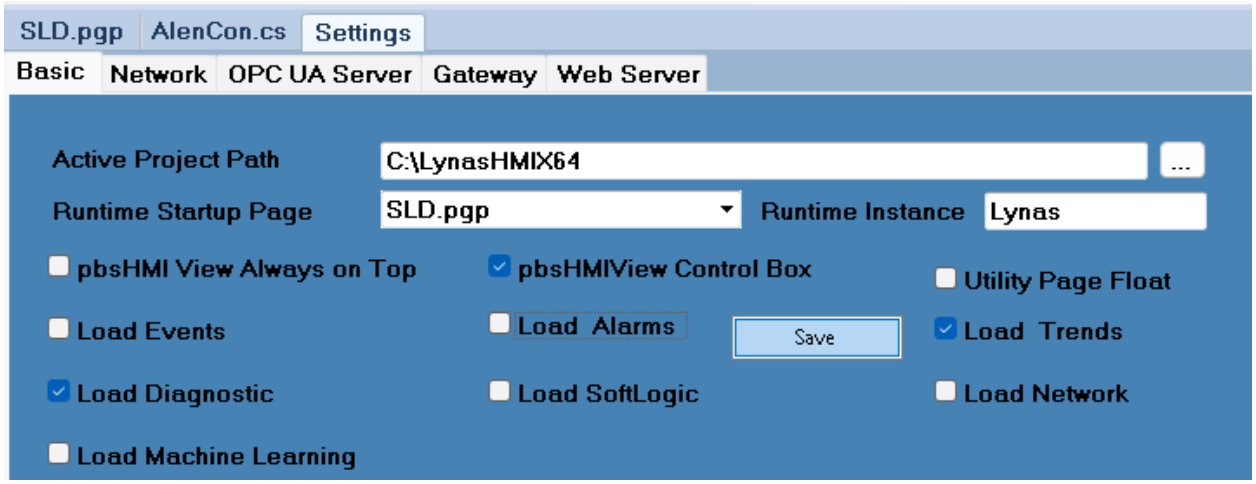

File
Help
Draw
Pen
Brush
Color
Text
RGB #666666
OLD #666666
A
100
Basic
System
Web
Custom
DEOPlant.pgp
1_Screens.pgp
File
Edit
View
PV-401


File
Help
Draw
Pen
Brush
Color
Text
Microsoft Sans Serif
11
B
DEOPlant.pgp
File
Edit
View

You can setup project settings by “Utilities” panel .Double click on “Settings” item, you can see following page:

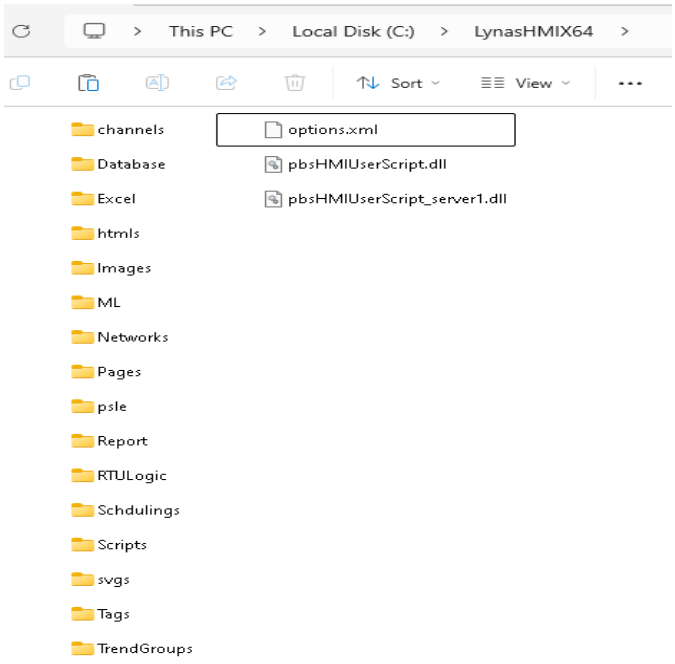


You can set a new or defined project path by “Active Project Path” text. Type a new path and right click and save new configuration.



If you set a new path, you need to exit application and run pbsHMIEditor.exe. At startup if project path is new, pbsHMIEditor create all folders and necessary files at specified project path.

pbsHMI project folder has following structure :



Channels: Communication driver settings

Database: Data Logging Settings

Htmls: exported html files for pbsHMI Web Server.

ML : Machine learning modules and settings

Networks : Power grid configurations and modules

Pages : Graphic pages

Psle : Function Block files . SoftPLC

Scheduling : running C# Scripts by user defined schedules

Scripts : user C# Scripts

Svgs : Exported graph pages to SVG format for Web Server

Tags : Defined device , Global , alarms and event tags

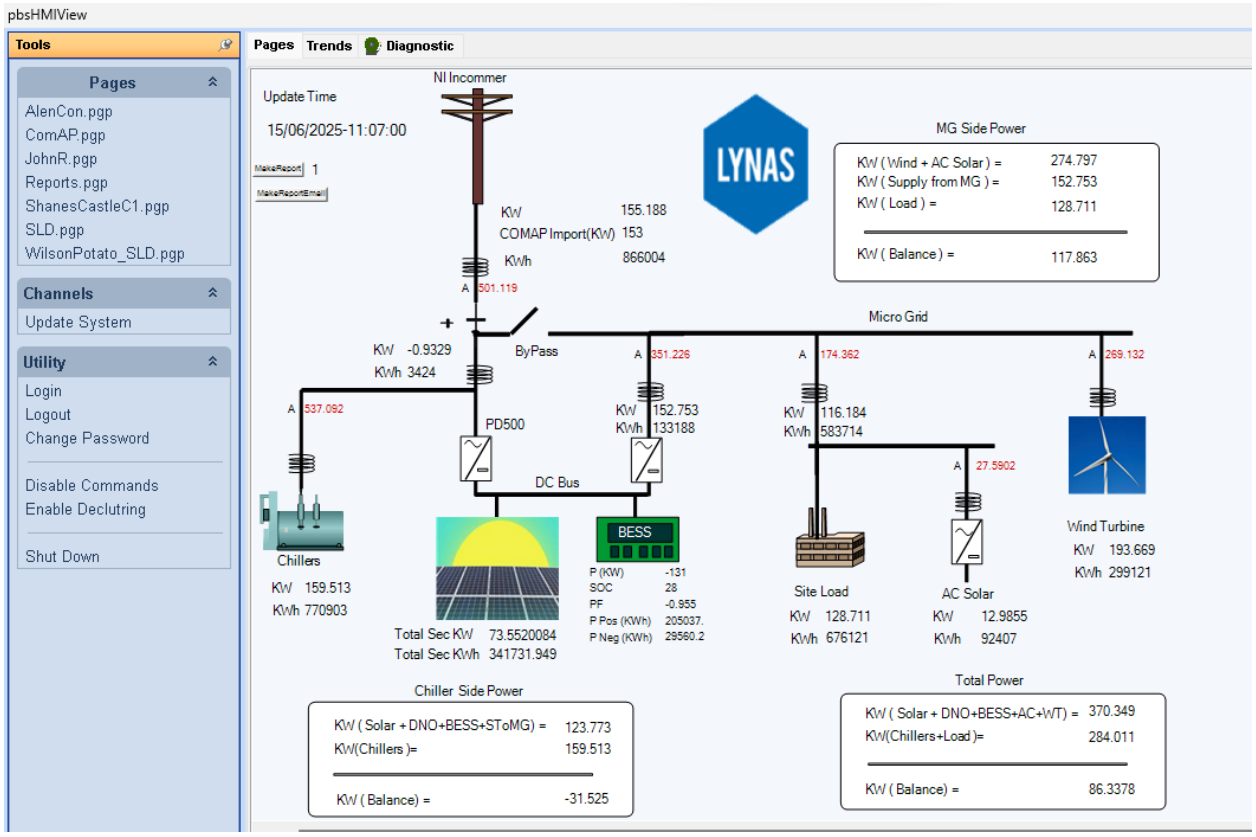
TrendGroups : user defined Trends

Option.xml : project settings and parameters

pbsHMIUserScript.dll : generated by pbsHMIEditor at compile time of C# Scripts

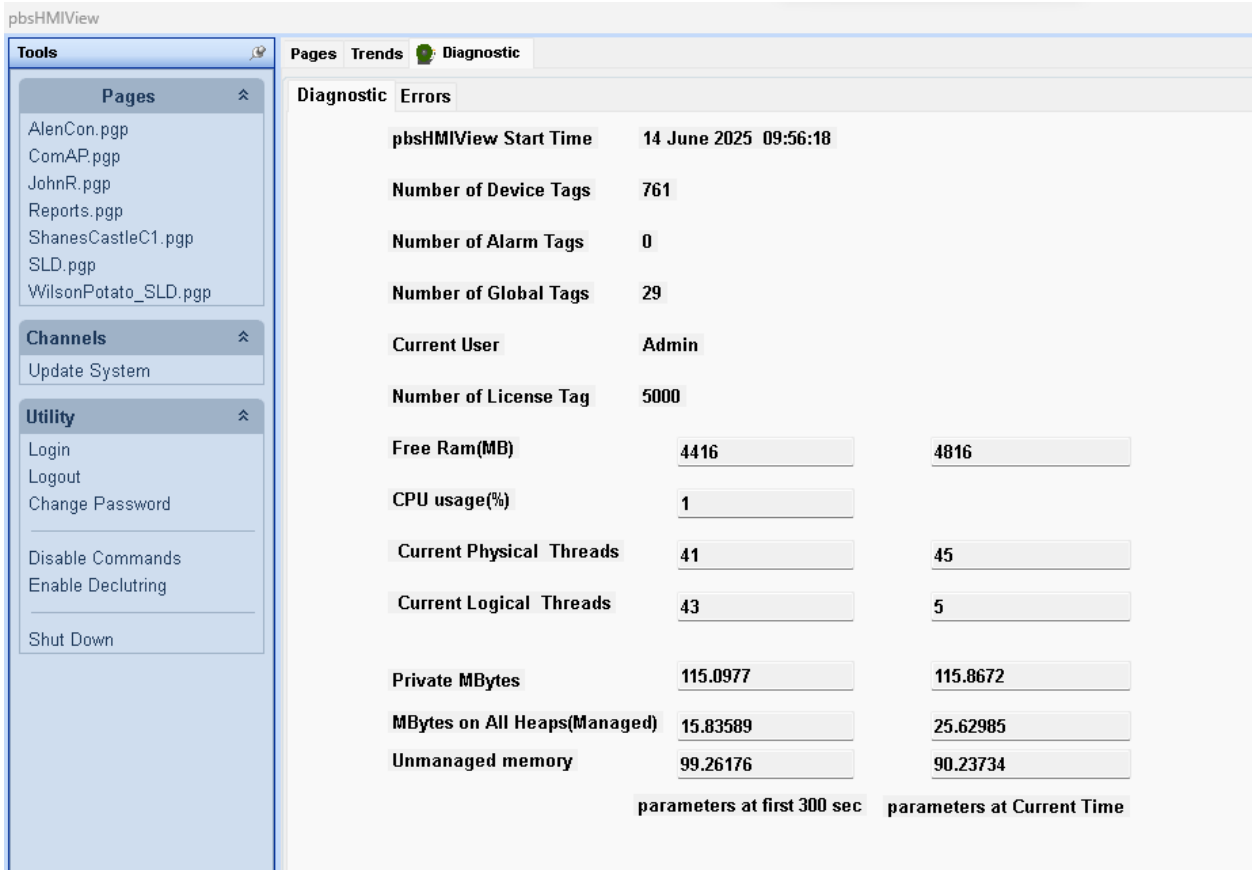
For running pbsHMI Project, you need to run pbsHMIViewer.exe, runtime application. pbsHMI runtime , open active project and load drivers , alarms , events , Database modules and shows startup page .

You can see following page structure when pbsHMI runtime is loaded:



At the left side you can see list for visible pages, updating runtime and utility menu. At the right side you can see startup page or selected page by user. Also trends, alarms, Events and Diagnostic pages can be shown based on project settings.

If diagnostic is loaded in the setting page, then you can see a Diagnostic tab in the pbsHMI runtime. Like following image:



This is showing when runtime is started, is there license for pbsHMI or not, number of device tags, number of global and alarm tags and information about memory and system resources.

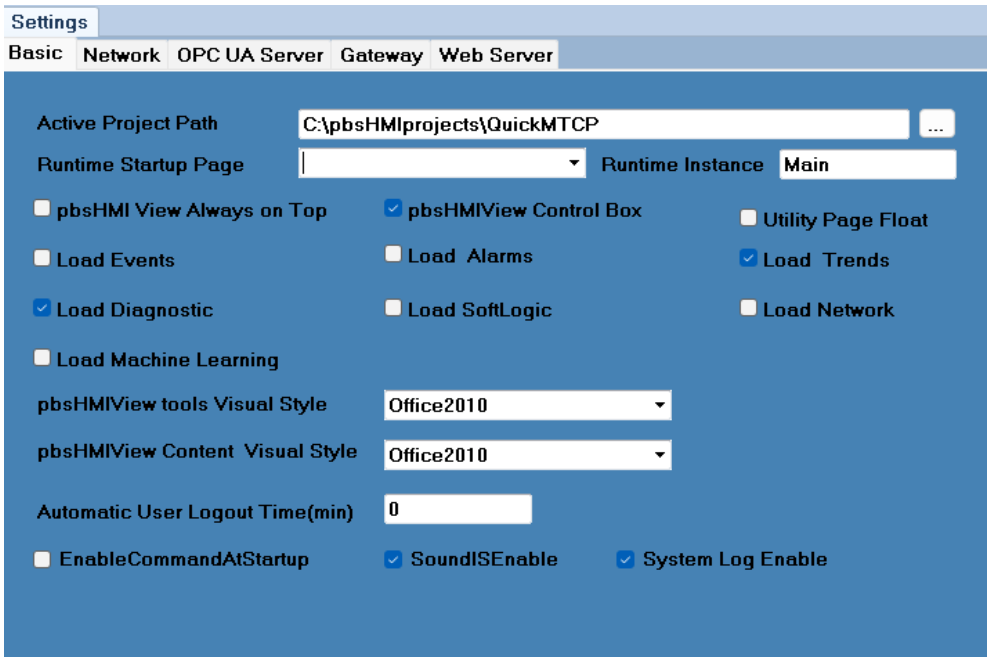
If pbsHMI is running without license you will see -1 for number of license tags and system will stop updating tags after 30 min.

3 – Quick startup project

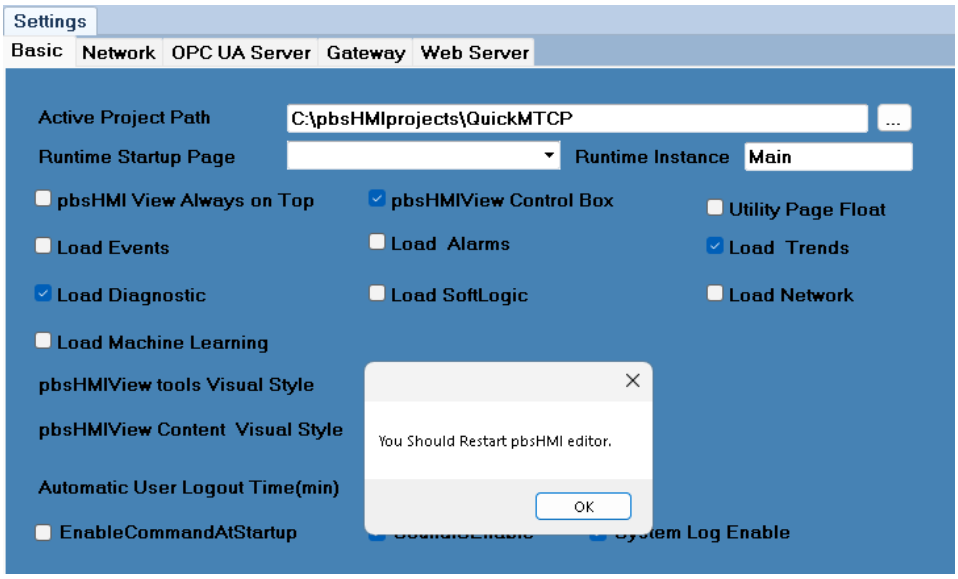
In this section , we will define a simple project based on Modbus TCP protocol and define a graphic page with a few dynamics and events .

Step 1: run pbsHMI Editor, go to Settings panel and double click on settings item.

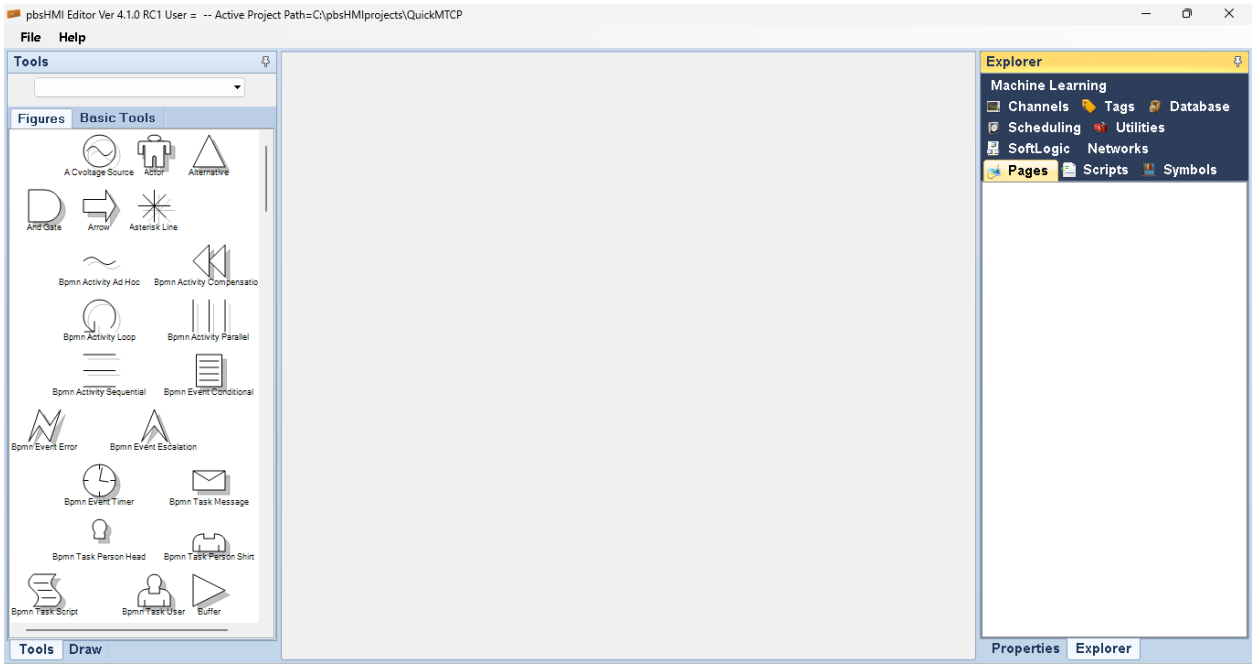
Type new project path for example “C:\pbsHMIprojects\QuickMTCP” as following:



Right click on the page and save setting. Because active project name is changed, you need to restart pbsHMI editor.

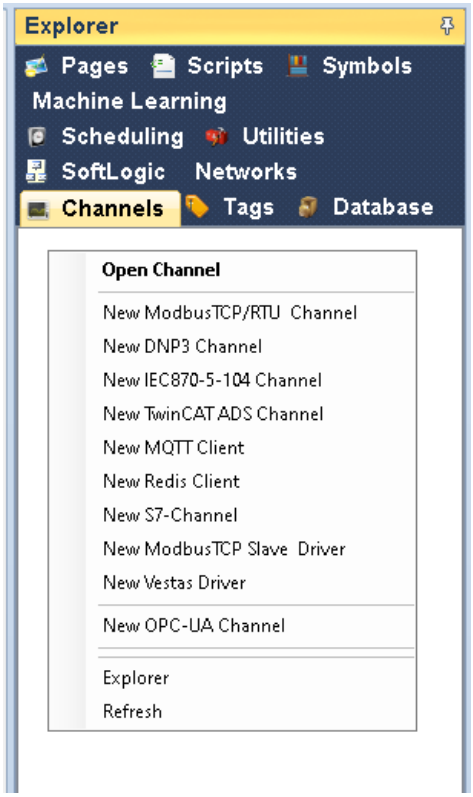


After executing pbsHMI Editor, you can see following screen:

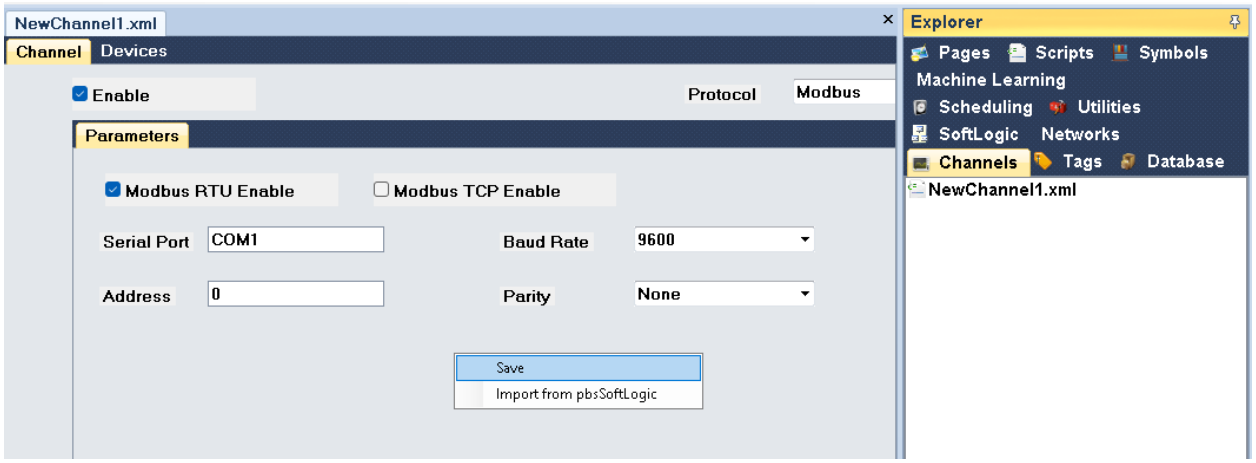


pbsHMI is created necessary folders and setting files at startup .

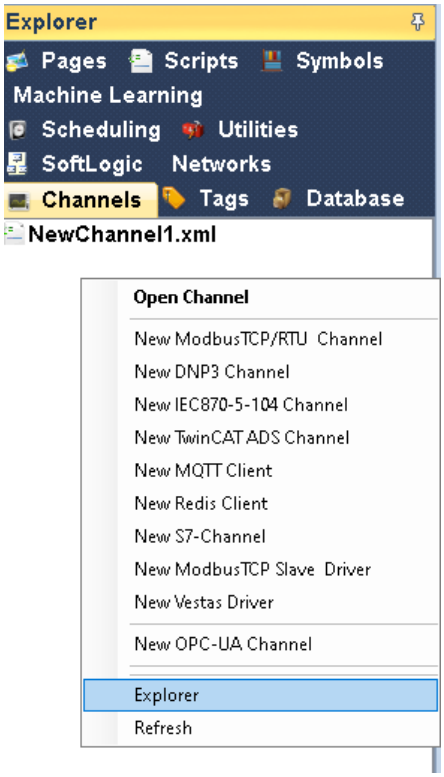
Click on “Channels”; right click on the list area, you can see following items:



Click on “New Modbus TCP Channel” , a new channel with “NewChannel1.xml” name is created in the “channels” panel .

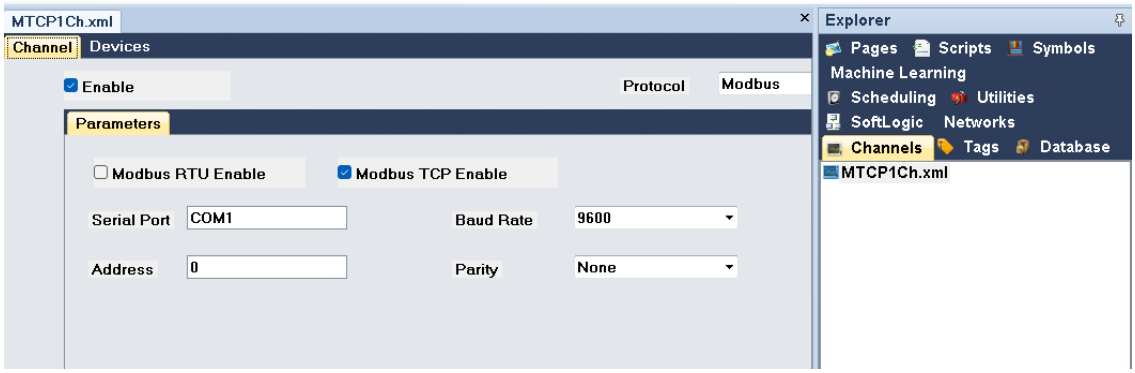


Right click in the channel setting page and save settings. Close “NewChannel1.xml” Setting page and right click on the channel list panel and run “Explorer” item. This will open channels folder in the project folder.

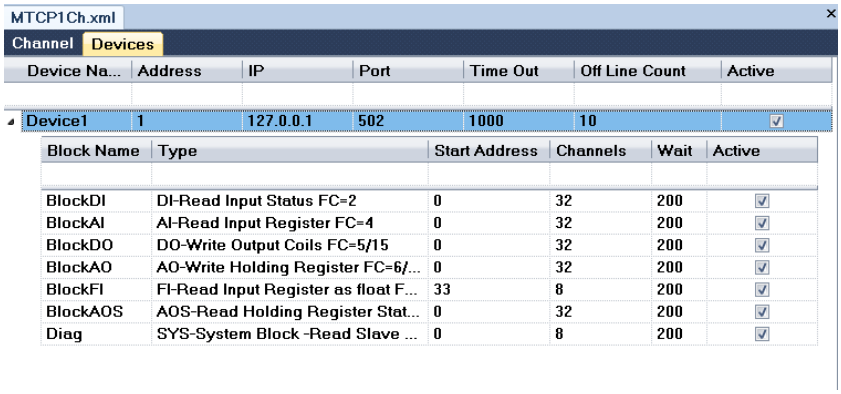


Rename the “NewChannel1.xl” to any name based on your project and refresh channel list. For example we change channel name to “MTCP1Ch.xml”.

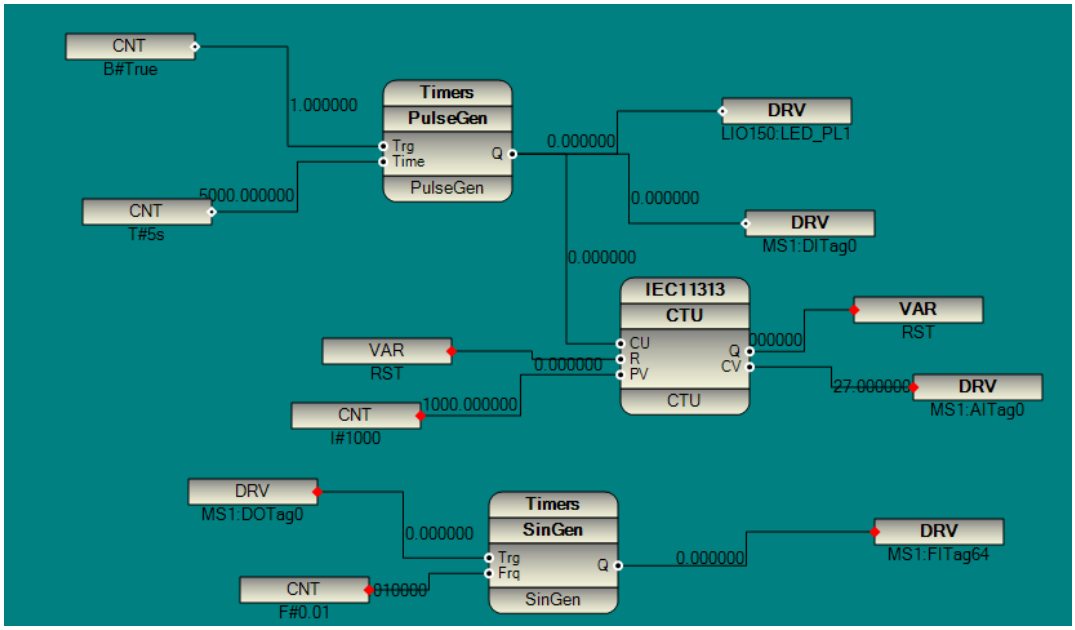
Double click on the “MTCP1Ch.xml” channel and change physical layer to Modbus TCP.



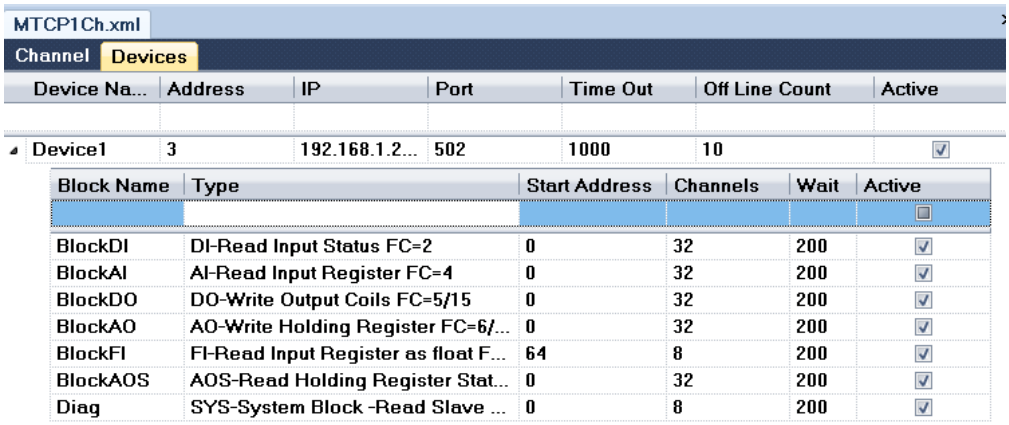
Go to “Devices” tab, you can see default Modbus Blocks that are defined at channel definition time.



I already defined a pbsSoftLogic project for a RTU with IP 192.168.1.225 and modbus TCP Slave Driver with ID = 3 and with following simple logic:



Changing “MTCP1CH.xml” parameters based on slave configuration as following:


MTCP1Ch.xml

Channel | Devices

Device Na...	Address	IP	Port	Time Out	Off Line Count	Active
Device1	3	192.168.1.2...	502	1000	10	☑

Block Name	Type	Start Address	Channels	Wait	Active
BlockDI	DI-Read Input Status FC=2	0	32	200	☑
BlockAI	AI-Read Input Register FC=4	0	32	200	☑
BlockDO	DO-Write Output Coils FC=5/15	0	32	200	☑
BlockAO	AO-Write Holding Register FC=6/...	0	32	200	☑
BlockFI	FI-Read Input Register as float F...	64	8	200	☑
BlockAOS	AOS-Read Holding Register Stat...	0	32	200	☑
Diag	SYS-System Block -Read Slave ...	0	8	200	☑

For Read/Write Modbus tags , you need to define modbus blocks . Set Block Name , Type , Start Address and number of channels .

The first Block that is defined is reading input Status FC = 2 from address 0 for 32 channels. pbsHMI Modbus TCP Driver ,sends proper command to read input status , wait for 200 msec and then expecting for answer from Slave Device . So 200 msec is not timeout, it is time that slave should answer to the master.

Second Block is reading input register from address 0 for 32 channels.

The third block is for writing to the Slave Driver. It is writing from address 0 for 32 channels.

The next block is writing Holding Register from address 0 for 32 holding registers.

The next block is reading floating input from device from address 64 for 8 channels. This command read input register from address 64 and read 16 channels. Then each 2 Registers are considered as one float input channel.

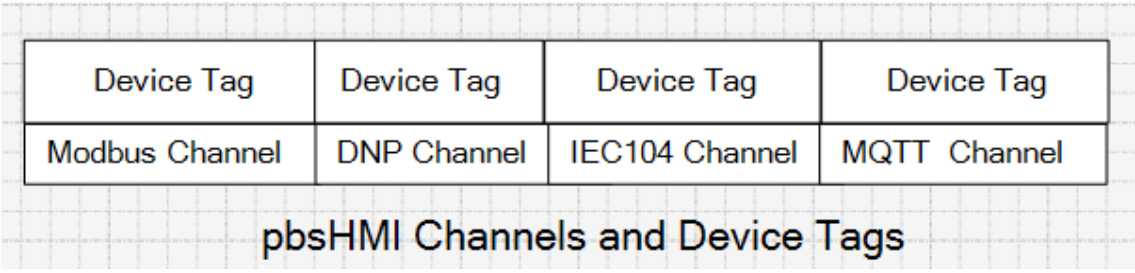
The last Block is for diagnostic, it shows driver is online or offline, how many frames send and how many answer is received from salve device.

You can delete these blocks and define new ones based on slave device address and parameters.

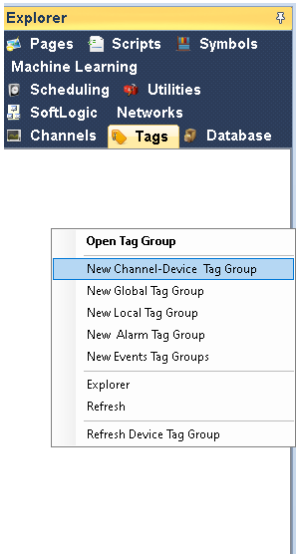
In the sample pbsSoftLogic RTU project, we used FITag64 tag and link it to a sin generator FB. This tag defined by 2 input registers with address 64 and 65, so for reading this tag in pbsHMI , I changed BlockFI start address to 64 .

Save channel by right click on the driver page and use “Save” menu.

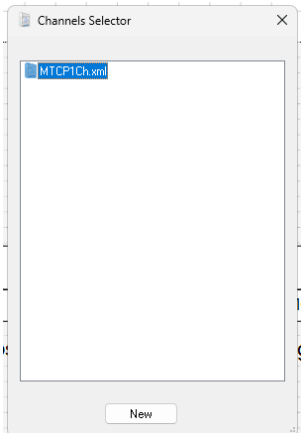
In the following figure, you can see pbsHMI channels and device tags relation. There is one and only one device tag over each channel. pbsHMI channels handle communication with external devices and will pass signals to device tags . For each protocol, there is a Channel with different parameters but Device tags for all protocols have same structure.



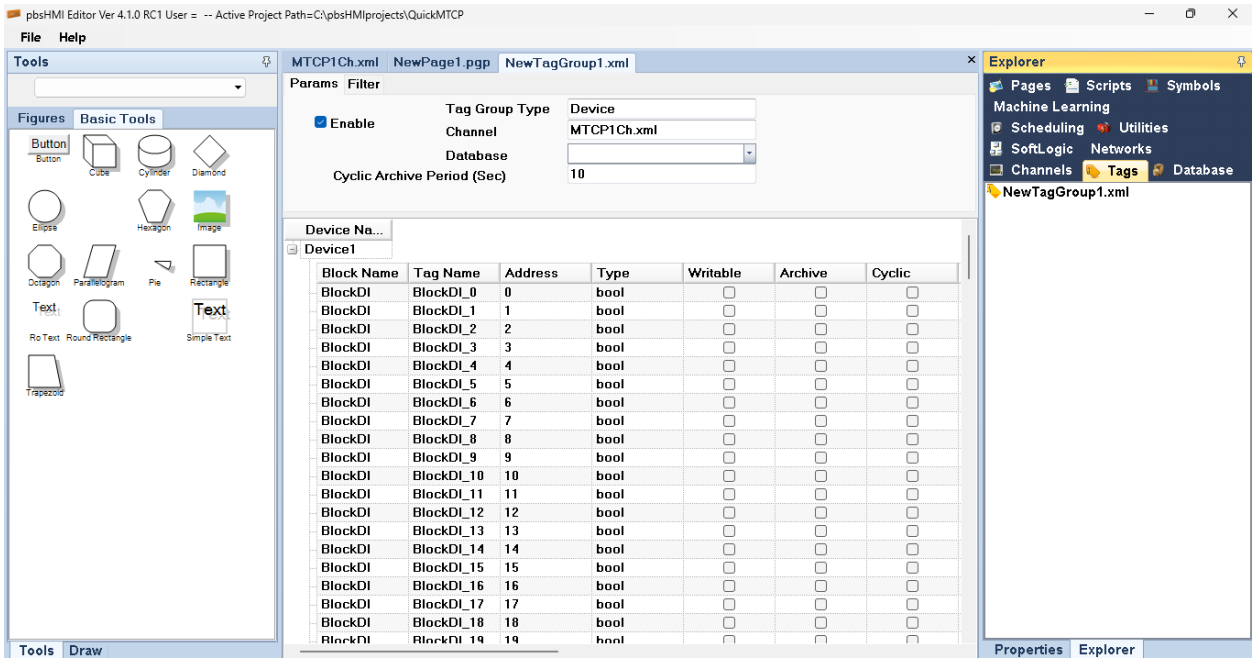
For defining device tag over Modbus Channel, in the “Tags” panel, right click on the list are and define a “New Device Tag”.



Select Channel that you want to define tag from the list. For our example select “MTCP1Ch.xml” channel.



Click on New Button, pbsHMI will generate Tags for selected device. Open Defined Device tag. You can see following image:

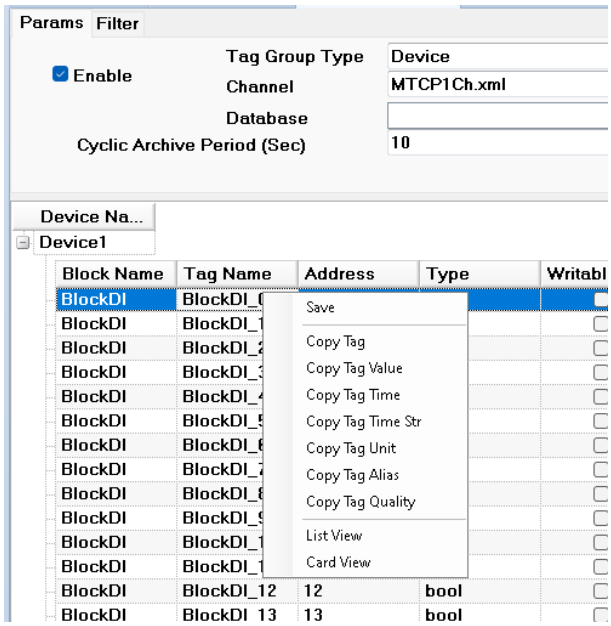


You can change Tag Names based on your Site Tag List, Log Tag by changes or cyclic , scale tag and set unit and Alias for the tag .

Change device tag name to a proper name. Suppose we change it to “MTCP1_Tags.xml”.

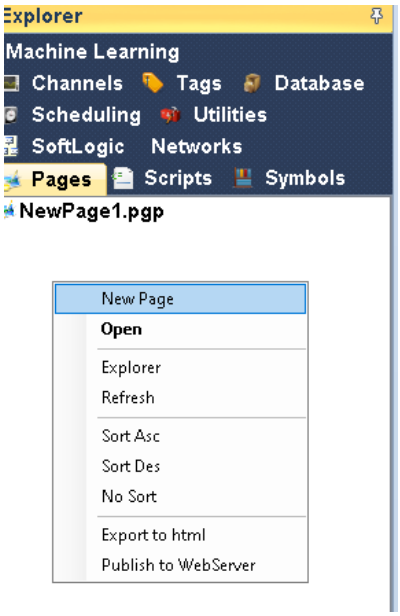
Device Tags will be used in Graphic pages, alarms, events, scripts, ..

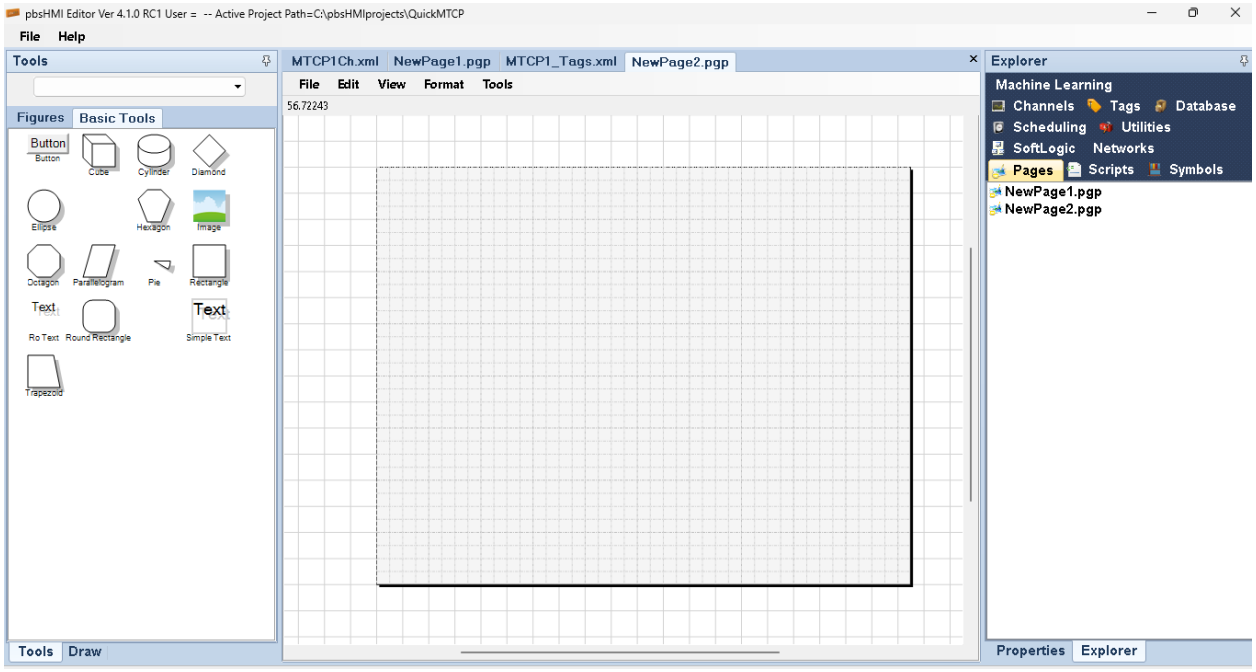
Right Click on the Tag that you want to use. You can copy tag properties:



Suppose we want to monitor value of “BlockDI_0” in a graphic page. Right Click on the “BlockDI_0” tag and select “Copy Tag Value” .

Select “Pages” Tab and create a new page by right click on the page list area:

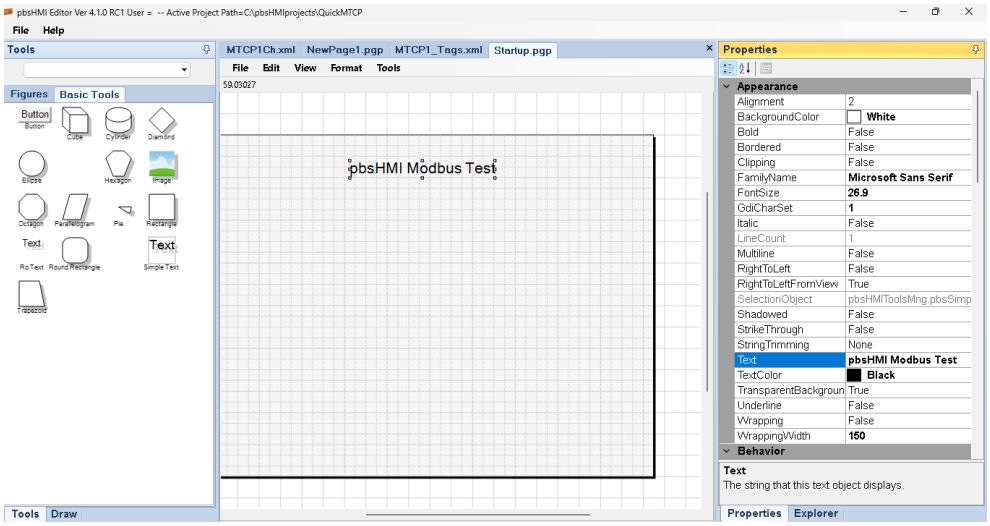




Because a page with name “NewPage1.pgp” is defined before, pbsHMI create a new page with name “NewPage2.pgp”. pbsHMI graphic page extension is “pgp” but it an xml file .

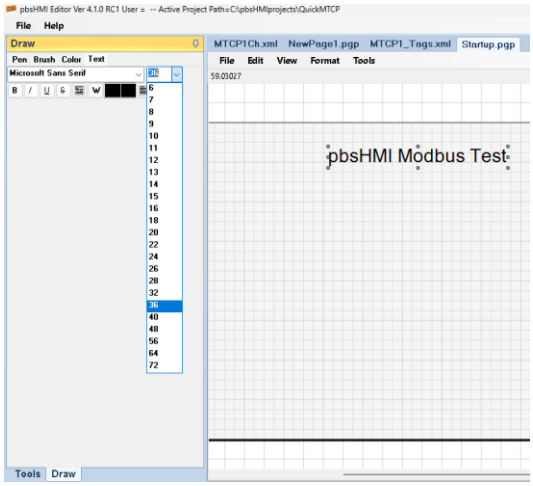
You can change graphic page name by going to the page folder in the project folder and change name and refresh page list same as changing channels name. I changed “NewPage2.pgp” to “Startup.pgp”.

Open “Startup.pgp” page and drag and drop a “Text” object from “Basic Tools “to the “Satrtup.pgp” page.

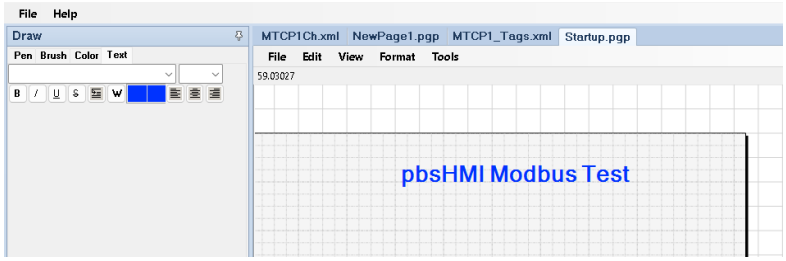


Select “Text” Object and open “Properties” for the selected object. Go to “Text” property and change it to “pbsHMI Modbus Test”.

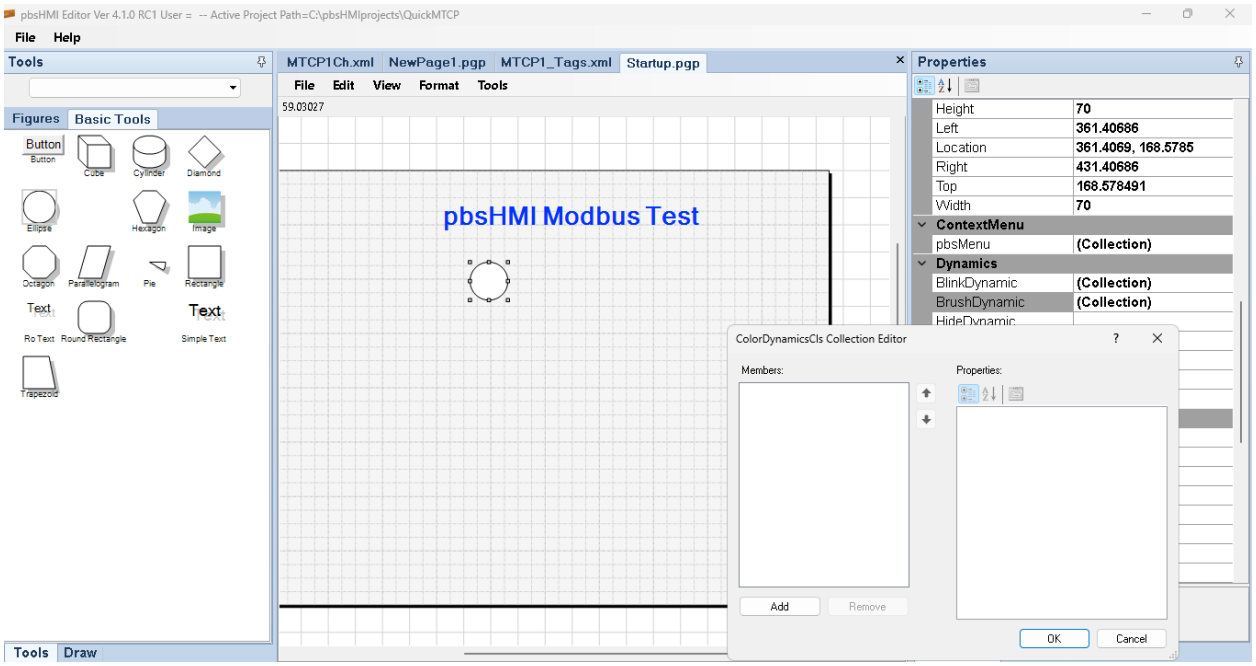
Select “Draw” panel and select “Text”. You can change text size, Color, Format and other properties here.



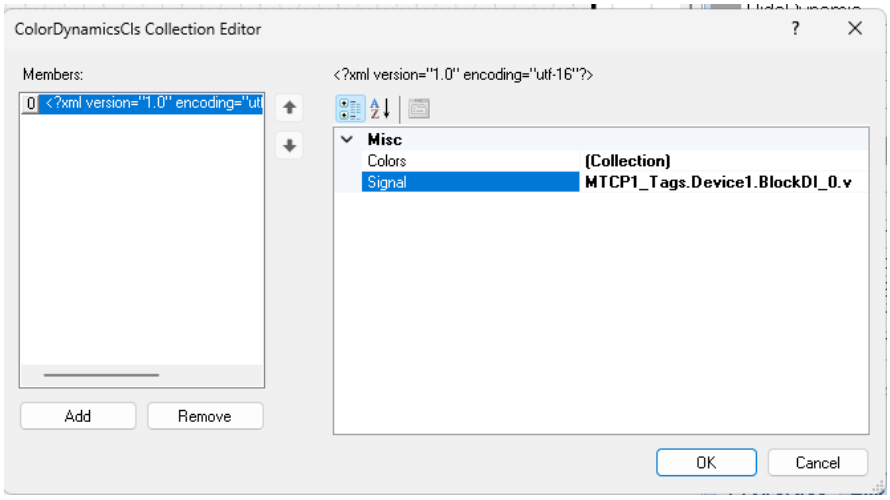
Change Size to 36, and make it bold and text color to blue.



Select a “Circle” object and drop it to the page. Go to “BrushDynamic” , click on “Add” Button .



Paste, copied Tag to Signal part:

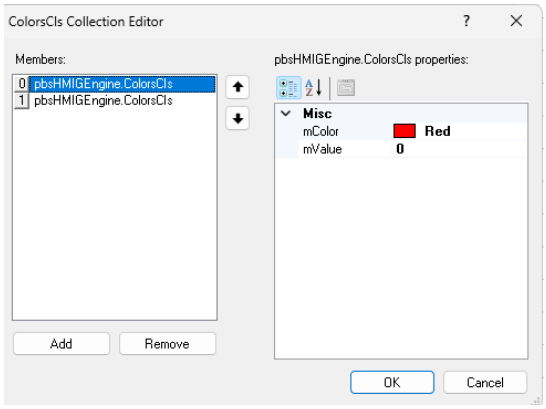


As you can see tag structure is {DeviceTag}.{DeviceName}.{TagName}

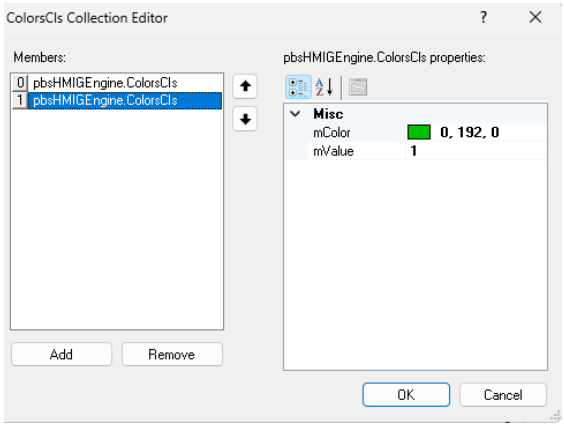
By default tag name has value property. Example: MTCP1_Tags.Device1.BlockDI_0

Tag Value: {DeviceTag}.{DeviceName}.{TagName}.v . Example: MTCP1_Tags.Device1.BlockDI_0.v

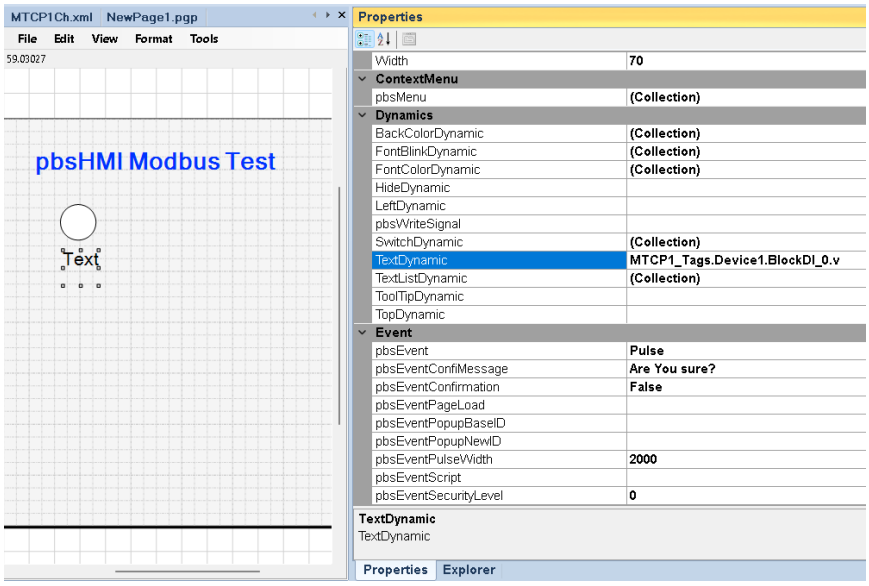
Click on “Colors” and Add color for Tag Value 0 (False).



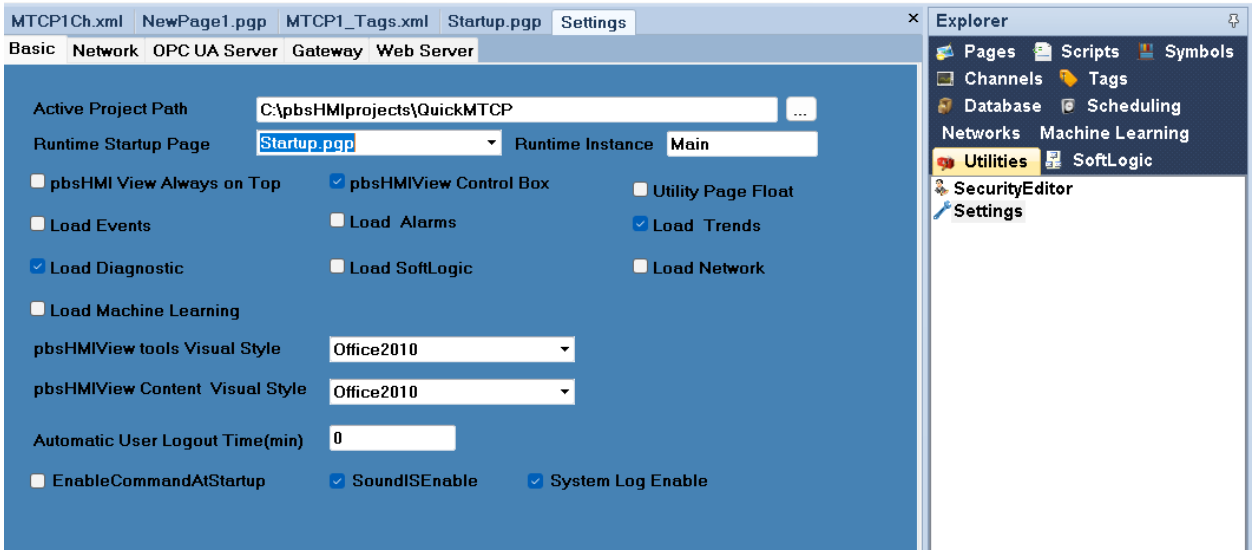
Add another color for value 1(True).



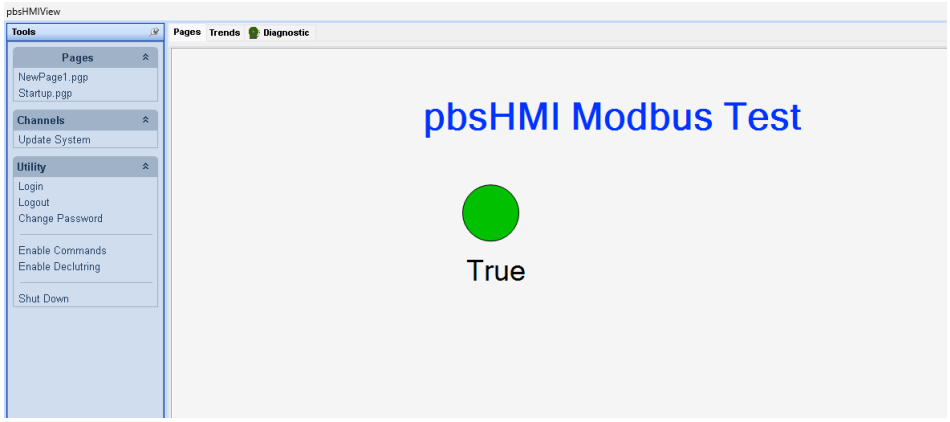
Drag another “Text” object and put it under circle object and paste “MTCP1_Tags.Device1.BlockDI_0.v” to “TextDynamic” property.



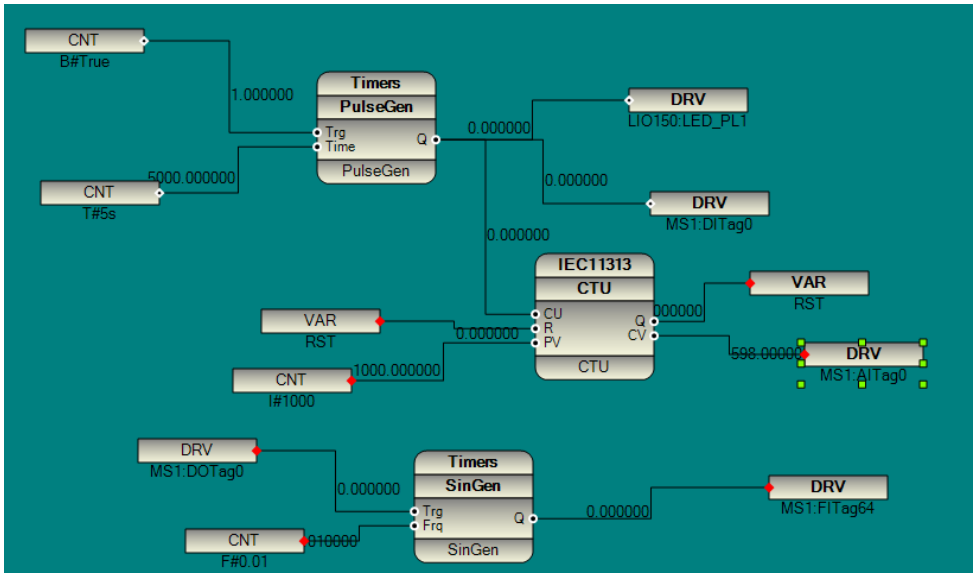
Save Page and open “Settings” page in the “utilities” tools. Select “Startup.pgp” as startup page and save settings.



In this stage, we can run the project by pbsHMIViewer and connect to RTU by Modbus protocol.

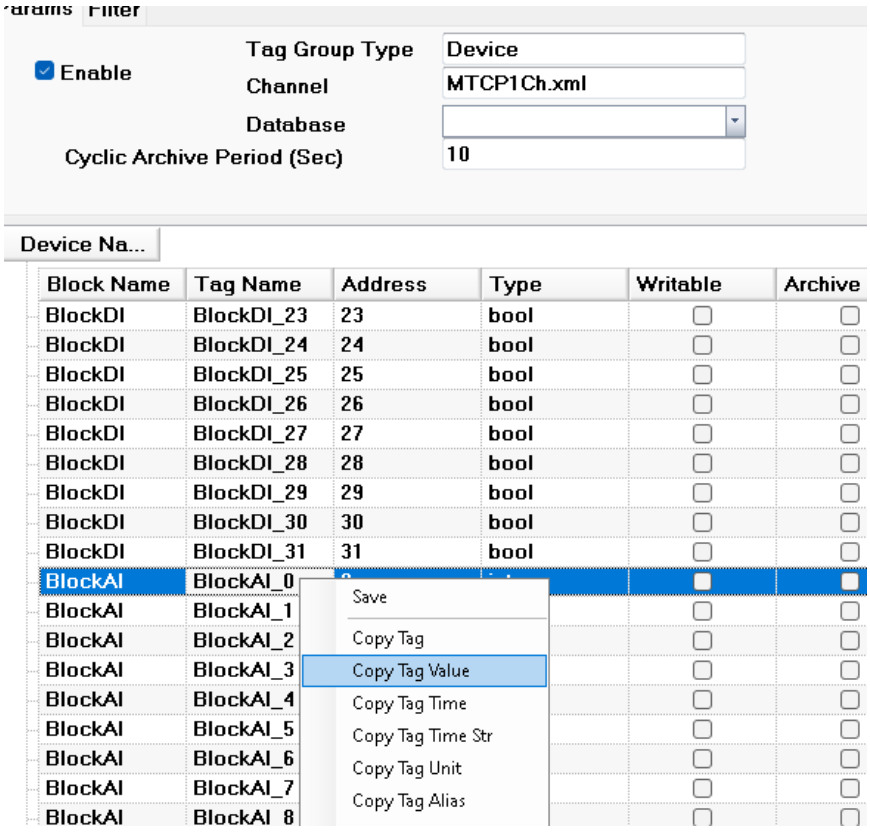


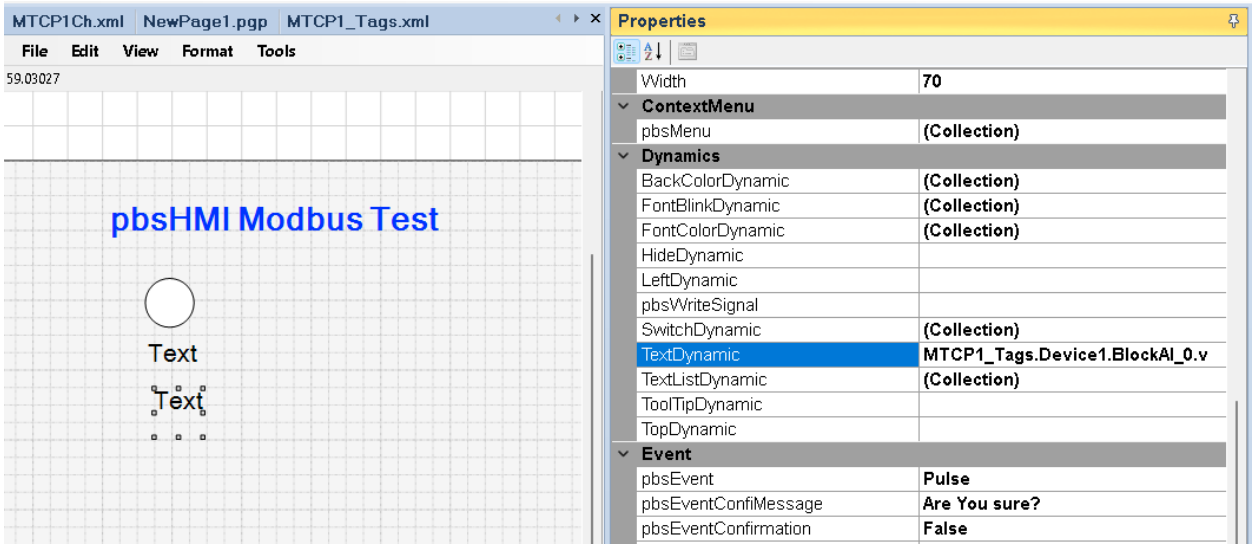
In the RTU logic , “TagAI0” is linked to a Up Counter .



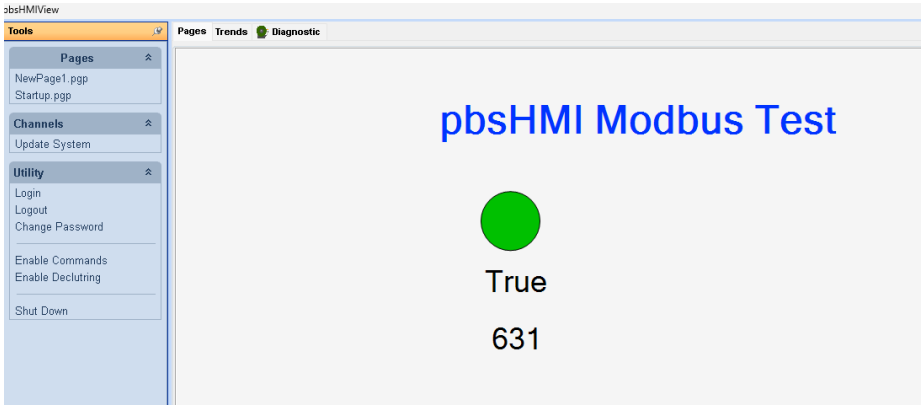
Suppose we want to monitor this tag . Select “TagAI0” from “MTCP1_Tags.xml” and copy tag value.

Drag a new “Text” object to the page and link “MTCP1_Tags.Device1.BlockAI_0.v” to New Text “TextDynamic” property.

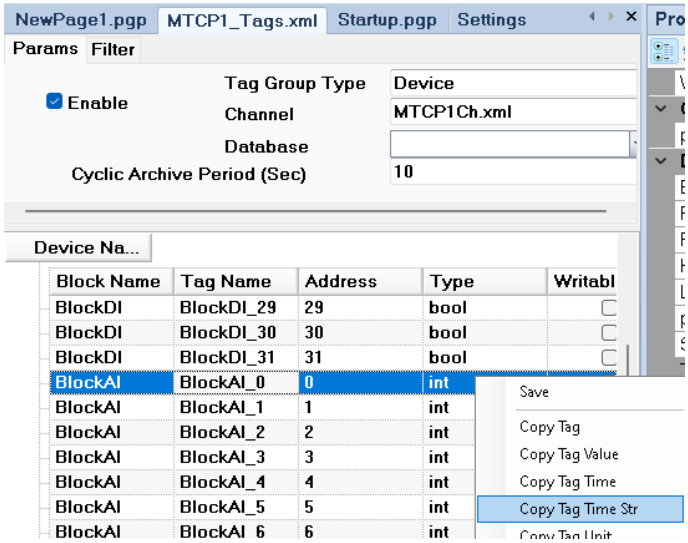


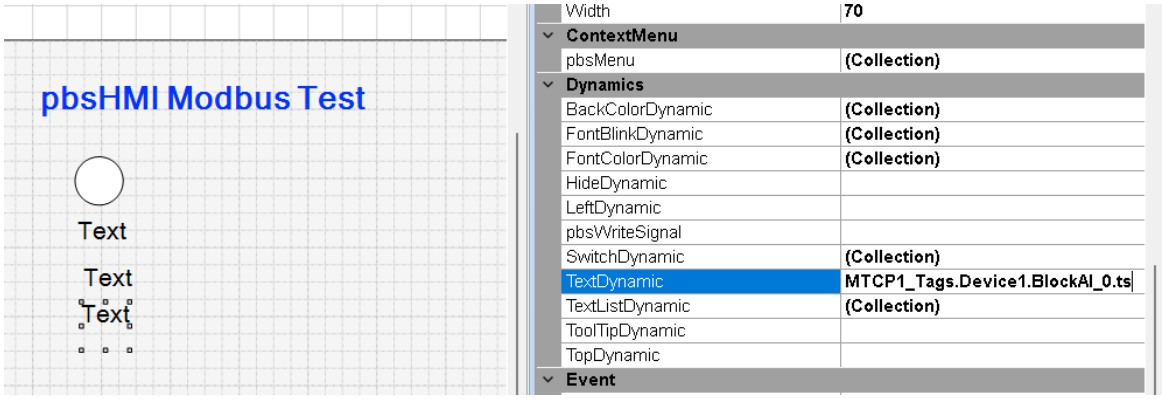


Save page and in the pbsHMIViewer click on “Startup” page. This will update page online.



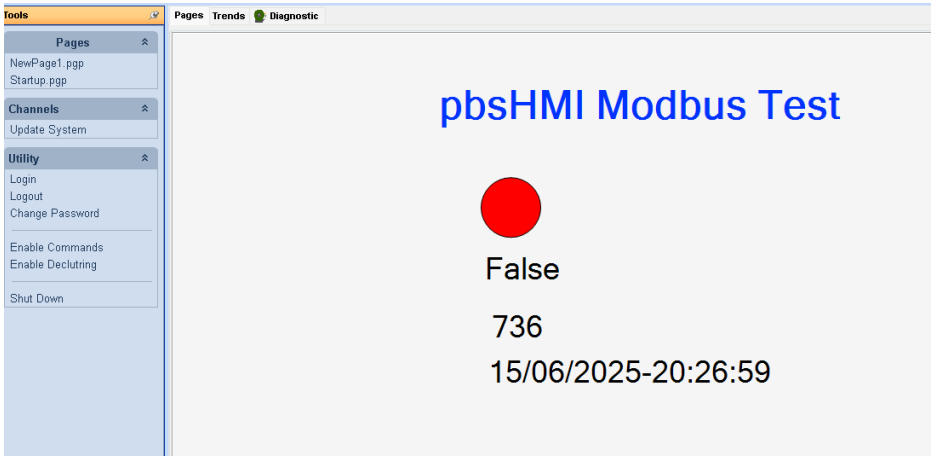
Suppose we want to show time of “BlockAI_0” tag in the page. You can copy “Tag Time Str” and link it to a “TextDynamic” of a new “Text” Object.



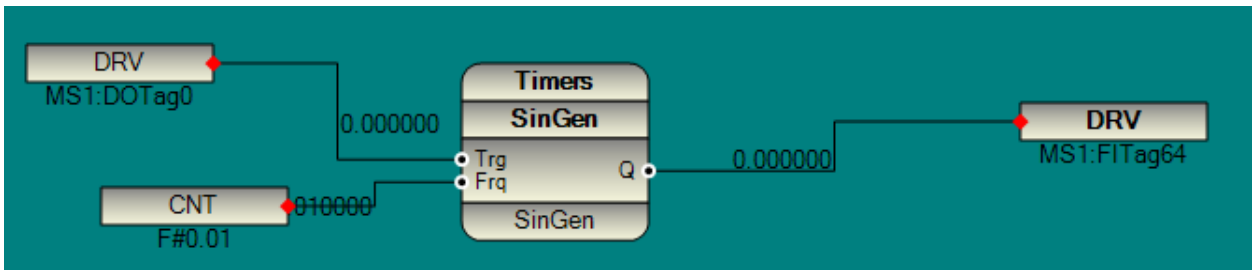


By adding ".ts" to the tag name, it will show tag change time in string format.

Change "pbsCharNum" to 32 to show more characters. Save page and update page at runtime.



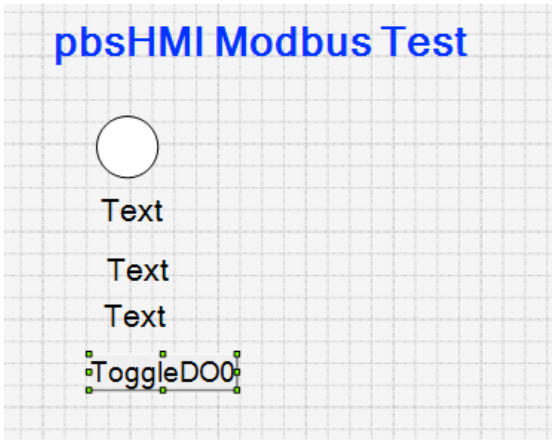
In the RTU logic, "DOTag0" signal is linked to trigger input of Sin generator Function Block and output is linked to FITag64 signal.



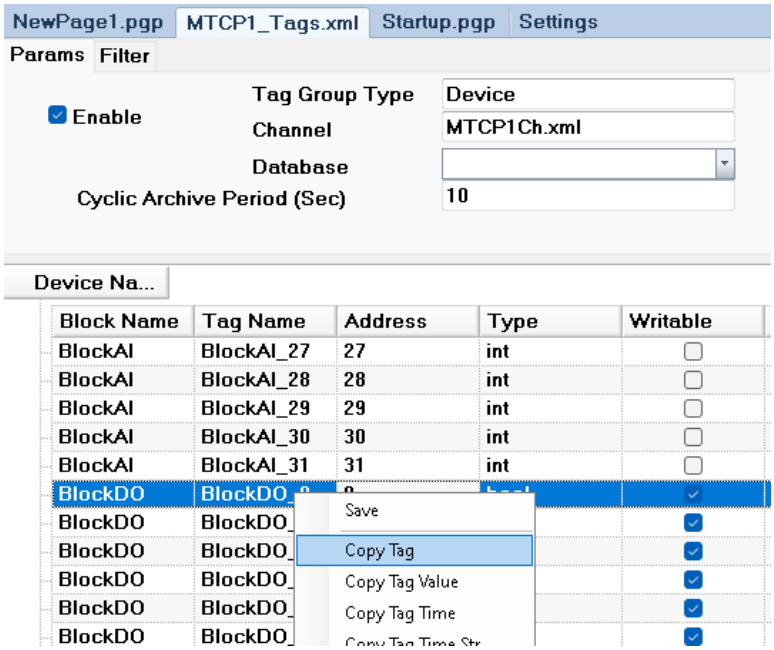
Suppose we want to Write DOTag0 Signal by pbsHMI and monitor value of FITg64 .

For DOTag0 , modbus address is 0 and for FITag64 , modbus address is 64 .

Drag a new "Button" in the page and change "Text" Property to "Toggle DO0". Change Font Size to 20.

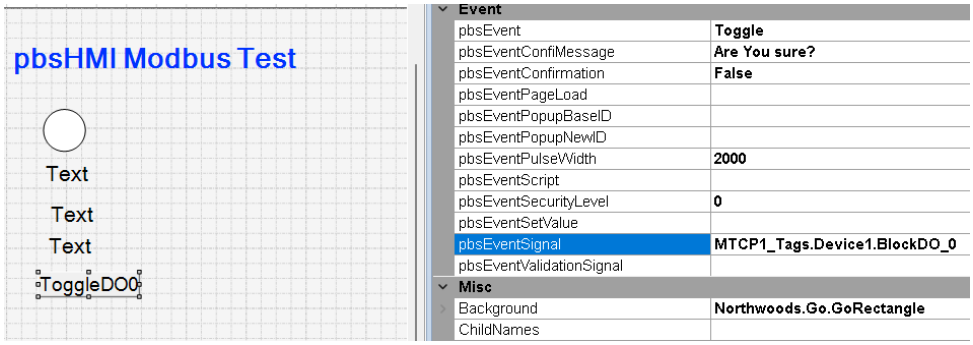


Select “BlockDO_0” tag and copy Tag.

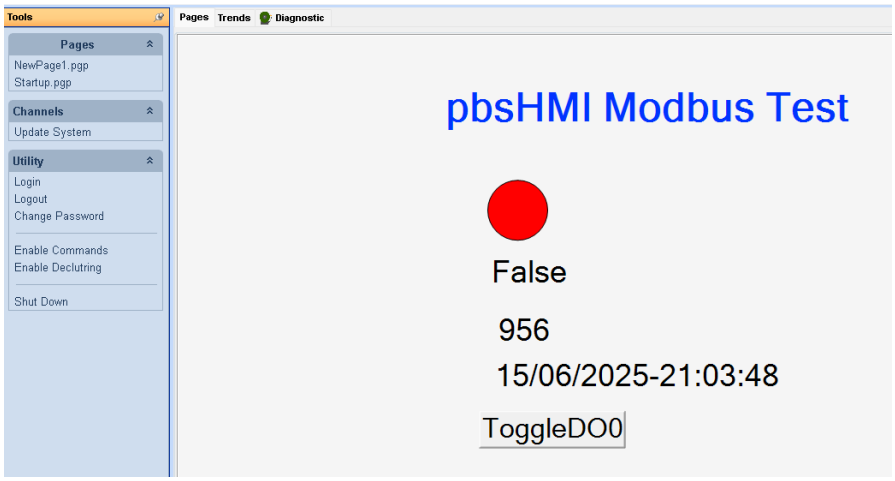


Note : for writing Tags , only tag Signal should be used

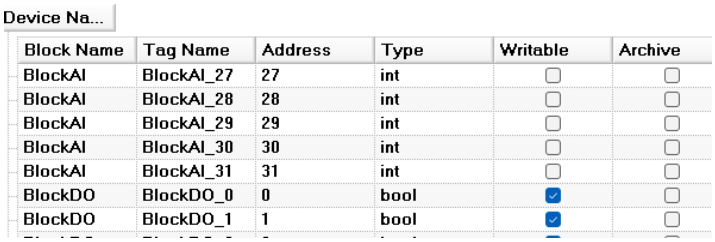
Select Button object in the page and change pbsEvent to “Toggle” and link “MTCP1_Tags.Device1.BlockDO_0” to pbsEventSignal .



Update page at runtime.

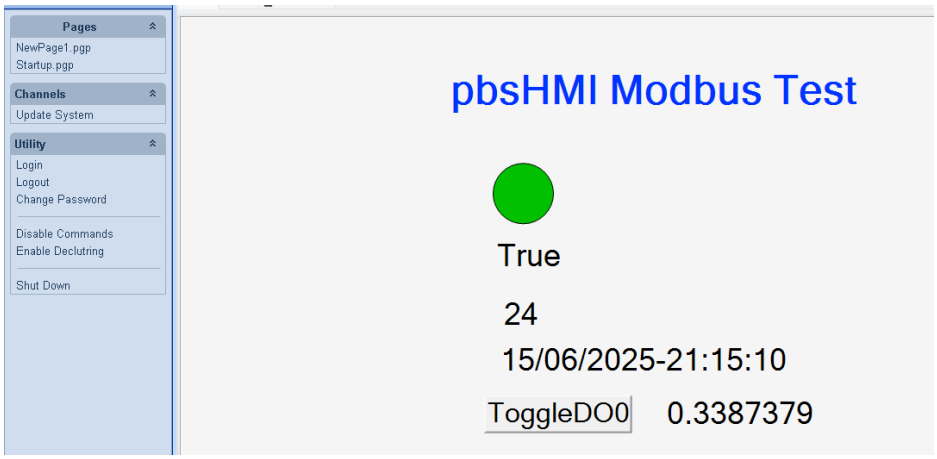


Writing Tag in pbsHMI need two steps. First in the Tag List “Writeable” property should be selected and second in the pbsHMI runtime “Enable commands” should be enabled.



Drag a new Text object to the page and link “TextDynamic” to the “BlockFI_64” Signal value.

Now if you update page and runtime and click on “ToggleDO0” button, that value of “BlockFI_64” is showing on the page.



4 – Modbus Driver Configuration and Global Tags

pbsHMI Supports Modbus Master Driver for communication with different Modbus slave devices .

ModbusRTU and ModbusTCP are supported for communication.

Following Modbus Function codes are supported:

```
FC_ReadCoilStatus = 0x1,
FC_ReadInputStatus = 0x2,
FC_ReadHoldingRegisters = 0x3,
FC_ReadInputRegisters = 0x4,
FC_ForceSingleCoil = 0x5,
FC_PreSetSingleRegister = 0x6,
FC_ReadExceptionStatus = 0x7,
FC_ForceMultiCoils = 0xf,
FC_PreSetMultiRegisters = 0x10
```

pbsHMI Modbus Block types:

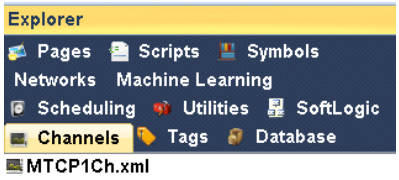
```
BI or DI = 1,// Boolean Input
BO or DO = 2,//Boolean Output(Write)
AI = 3,       //Analog input
AO = 4,       //Analog Output (Write)
BOS or DOS=5,// Boolean Output Status
AOS=6 ,       // Analog output status
FI=7,         // float input
SFI=8,        // swap float input
FO=9,        // float output(Write)
SFO = 10,    // swap float output(Write)
SYS =11,     // System Diagnostic Block
FOS = 12 ,   // Float Output Staus
SFOS = 13    // Swap Float Output Status
```

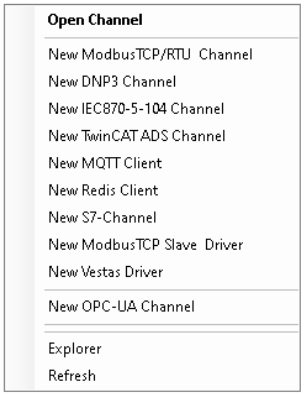
Modbus Channel: pbsHMI driver modeling is based on channel and device Tag configuration.

You need to define a new Modbus channel and define Device Tags based on channel.

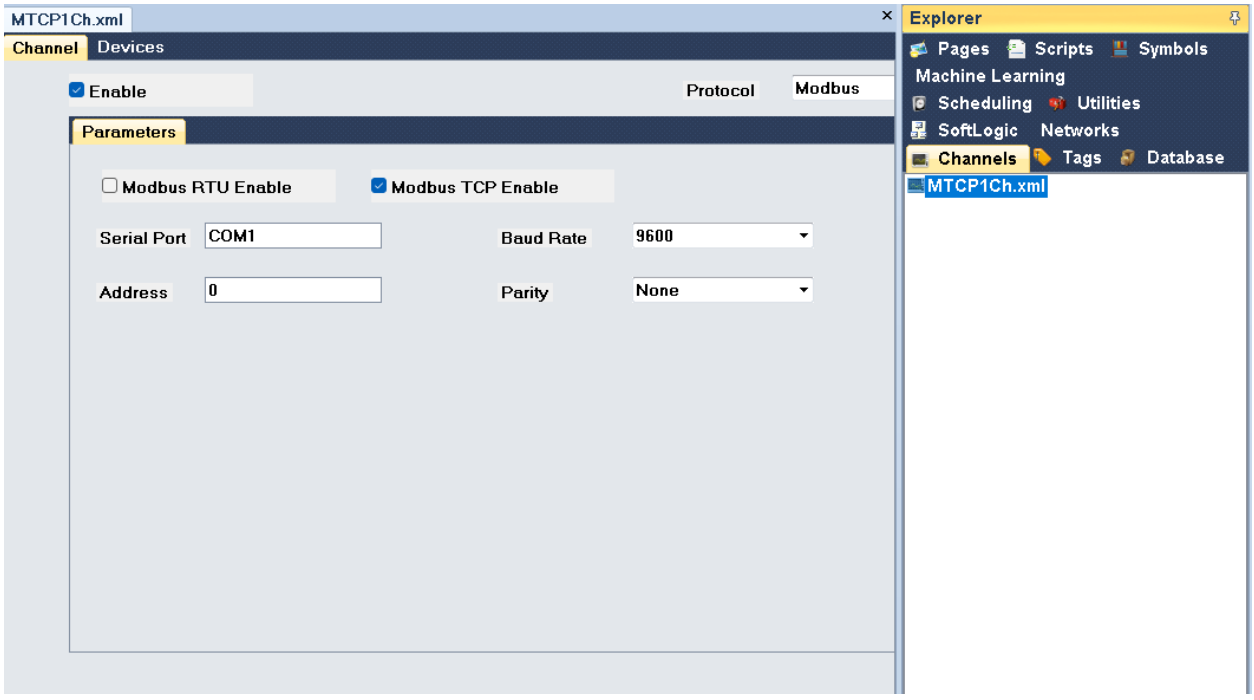
In channels, you will configure communication parameters and in Device Tags, you will define Driver tags.

Defining new MODBUS channel: right click on channels tab and select New MODBUS Channel.





pbsHMI will define a new MODBUS channel as following page :



Channel Tab: Select Serial or Ethernet for communication. By default Serial is Enable. Select COM port and other communication parameters.

Device Tab: Define Modbus Devices and data Blocks.

NewChannel2.xml

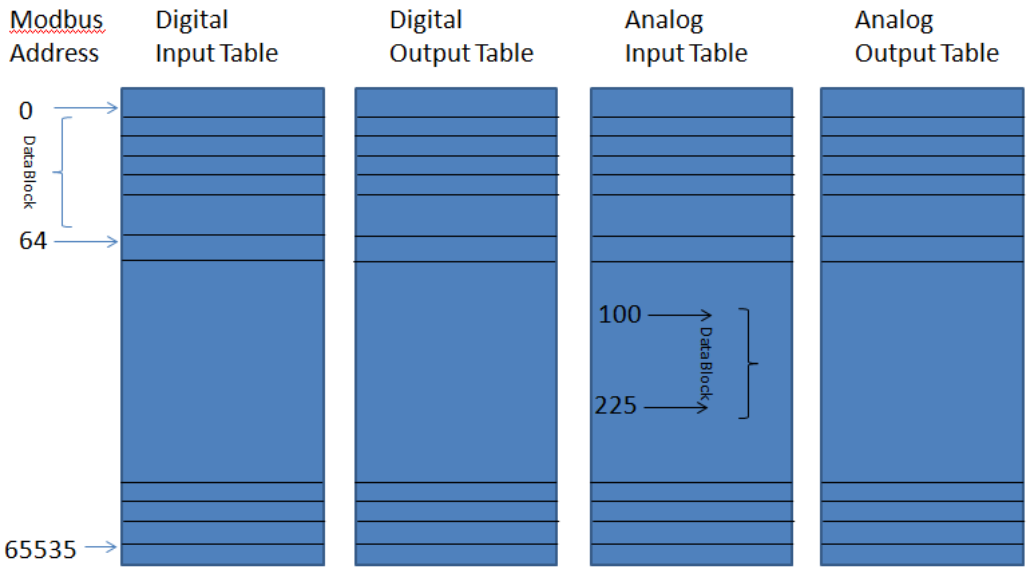
Channel | Devices

Drag a column header here to group by that column.

Device Na...	Address	IP	Port	Time Out	Tell Number	Off Line Count	Active
Device1	1	192.168.1.100	502	1000		10	☑

Device Na...	Block Name	Type	Start Addr...	channels	Wait	Active
Device1	Block1	DI	0	64	200	☑
Device1	Block2	AI	0	64	200	☑
Device1	Block3	DO	0	64	200	☑
Device1	Block4	AO	0	64	200	☑
Device1	Block5	FI	100	63	200	☑
Device1	Diag	SYS	0	8	200	☑

Any Modbus Slave device by default has four tables inside:



Based on MODBUS standard, maximum MODBUS address for each table is 65535, but it depends on Slave implementation. Normally slave devices are not supports whole address range.

pbsHMI Supported function codes for MODBUS protocol:

```
FC_ReadCoilStatus = 0x1 = Read Status of DO Table
FC_ReadInputStatus = 0x2 = Read DI Table
FC_ReadHoldingRegisters = 0x3 = Read Status of AO Table
FC_ReadInputRegisters = 0x4 = Read AI Table
FC_ForceSingleCoil = 0x5=Write  one DO Tag
FC_PreSetSingleRegister = 0x6 = Write one Ao Tag
FC_ForceMultiCoils = 0xf = Write multiple DO Tags
```

FC_PreSetMultiRegisters = 0x10 = Write Multiple AO Tags

MODBUS Data types: MODBUS Supports Digital (0-1) and Analog (unsigned 16 bit integer) Data types.

We can transfer other data types based on analog type. pbsHMI Supports floating point data type for reading and writing floating point values .

FI = Float Input. FI is using AI table. Each FI tag gets two AI addresses. pbsHMI will read FI tags from Slave device.

FO = Float Output. FO is using AO table. Each FO tag gets two AO addresses. pbsHMI will Write FO tags to Slave device.

FOS = Float Output Status. pbsHMI will read Status of FO tags by FOS Data type .

SFI = Swap Float Input. Same FI but AI tags order is reveres.

SFO = Swap Float Output .Same FO but AO tags order is reveres.

SFOS = Swap Float Output Status. Same FOS but AO tags order is reveres.

Any pbsHMI modbus Blocks contains many Modbus tags.

pbsHMI read Modbus input tags (DI , AI , FI , FOS , DOS , SFOS) by sending modbus commands (based on Block definition) and will write outputs (DO , AO , FO , SFO) when it detect changes in Block tags value.

SYS Block contains 3 tags: Online Status, Send Command number and receive Commands number.

Start Address and Number of channels in SYS block are dummy value. So you can keep them as default.

At top of Devices page, you need to set following parameters:

Address: Modbus Slave ID (from 1 to 254)

IP: IP address of Modbus TCP Slave Device

Port: Port Number for ModbusTCP Slave Device. By default its value is 502.

Active: Shows Device is active or not.

Note : When you use ModbusRTU , you can define multiple devices for a channel . But when you use ModbusTCP , just one device is allowed in pbsHMI . You need to define one channel for each ModbusTCP Device.

For defining new Block, type Block name, Block Type, Start Address, Channels, wait time (msec) and checked active box then press Enter at top blank record area.

For saving Block data, right click on blocks area and select save menu.

You can change Blocks by any XML editor. At channels panel, Right click and select explorer menu.

It will open pbsHMI channels directory. Open channel file by any XML editor (Like Notepad++)

```
<?xml version="1.0"?>
<pbsChannelCfg>
    <Version>1.0.0</Version>
    <Protocol>Modbus</Protocol>
    <DllDriver>C:\pbsHMI\pbsHMIModbusDrv.dll</DllDriver>
    <DllType>pbsHMIModbusDrv.pbsHMIModbus</DllType>
    <Enable>True</Enable>
    <Serialport Enable="False" ComPort="COM1" BaudRate="9600" Address="0" Parity="None" Protocol="RTU" />
    <Ethernet Enable="True" LocalIP="192.168.233.129" Address="0" Protocol="TCP" />
    <ModemPool Enable="False" PoolName="Pool1" Protocol="RTU" AlwaysConnect="True" SchdulingEvery="1" SchdulingAt="10" SchdulingFor="5" ReconnectAfter="3" />
    <Devices>
        <Device Name="Device1" Address="1" IP="192.168.233.1" Port="502" Timeout="1000" TellNumber="" OffLineCount="10" Active="True">
            <Blocks>
                <Block BlockName="Block1" Type="DI" StartAddress="0" Channels="64" Wait="200" Active="True" />
                <Block BlockName="Block2" Type="DO" StartAddress="0" Channels="64" Wait="200" Active="True" />
            </Blocks>
        </Device>
    </Devices>
</pbsChannelCfg>
```

You can change or add new Blocks by copy /paste/ modify <Block> tags.

Note: When define Blocks you need to carful about channels:

For AI and AOS maximum channel Is 127. (Modbus frame format limitation)

For FI, FOS, SFI maximum channels is 63. (Modbus frame format limitation)

If number of Analog and Float tags are more than maximum value, then you need to define multiple blocks.

Wait Time is time (msec) that driver will read Modbus Frame after sending command to slave device.

With modifying Wait Time you can change driver operation performance.

Defining Modbus Tags:

Select pbsHMI tags Panel and right click there. Select “New Device Tag Group”. Select channel that you want to made device tag and click on new Button.

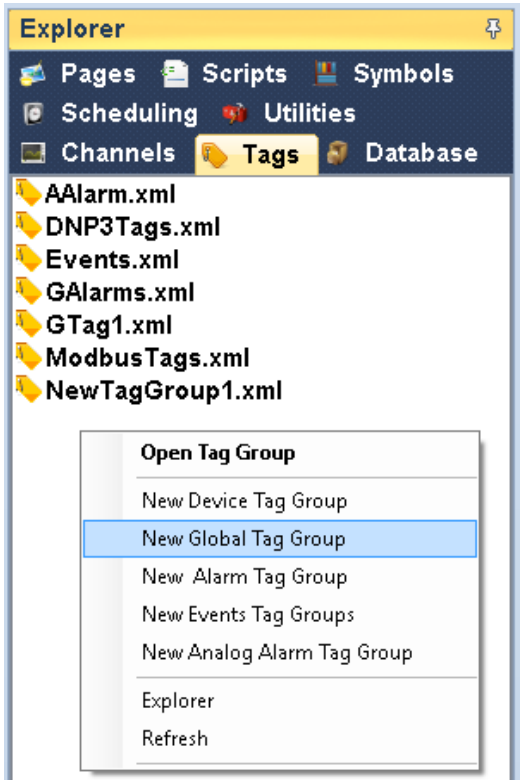
pbsHMI will automatically generate Modbus tags . The new Device tag has NewTagGroupx.xml name format. You can change name to any name you want by windows facility and refresh tags Panel.

Using of device tags is same for all channels and devices. So please refer to chapter 3 for using Device tags.

global tags

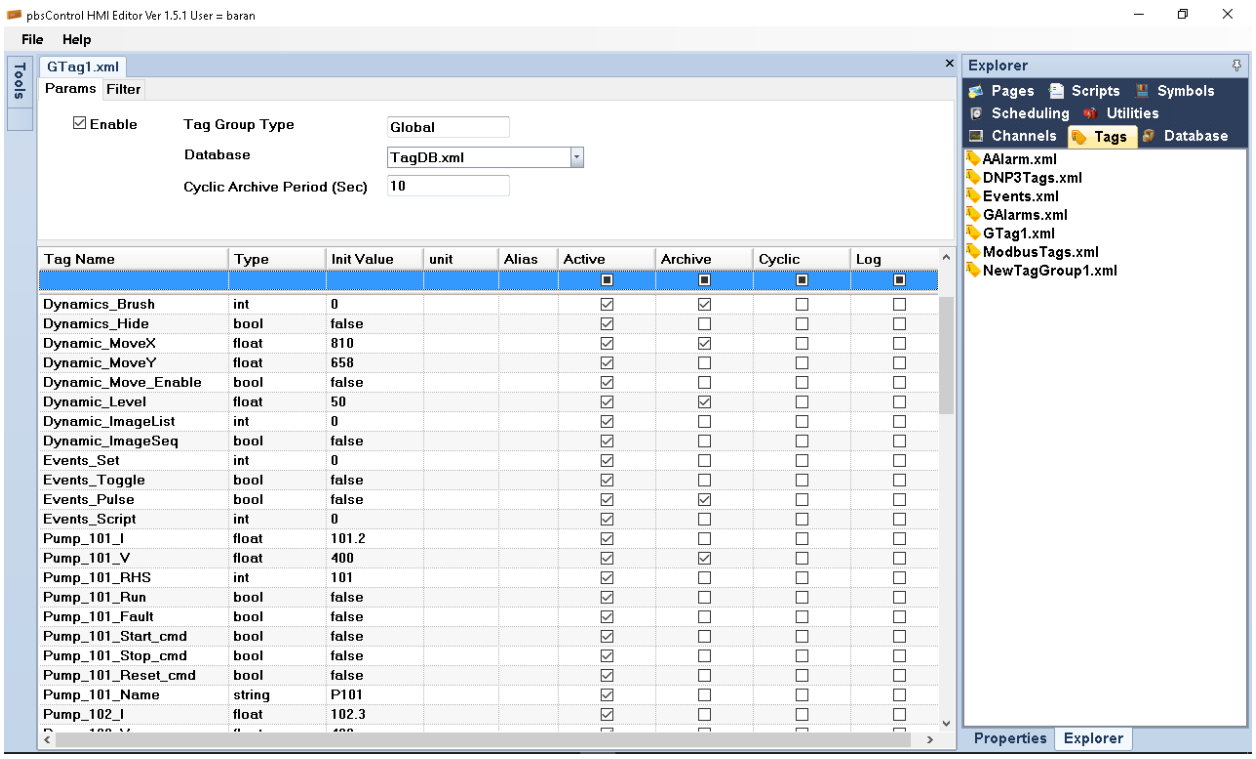
Global tag is internal tag in pbsHMI. You can use global tags in Pages , Scripts and everywhere in pbsHMI .

For defining a new global tag group , right click on “Tags” tab in project explorer . You will see following context menu:



Click on “New Global Tag Group” . You can define unlimited global tag groups and global tags in pbsHMI.

Global tags have following fields:



Tag name : it should be unique in the Tag Group

Type : C# type

int = Signed 32-bit integer , -2,147,483,648 to 2,147,483,647

uint = Unsigned 32-bit integer , 0 to 4,294,967,295

long = Signed 64-bit integer , -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

ulong = Unsigned 64-bit integer, 0 to 18,446,744,073,709,551,615

double = 64-bit signed floating-point value

float= 32-bit signed floating-point value.

bool = one of two possible values, true or false

string = Represents a string of Unicode characters.

Init Value : Initialize value of tag

Unit : unit of tag

Alias : alias of tag

Active : if checked tag is enabled

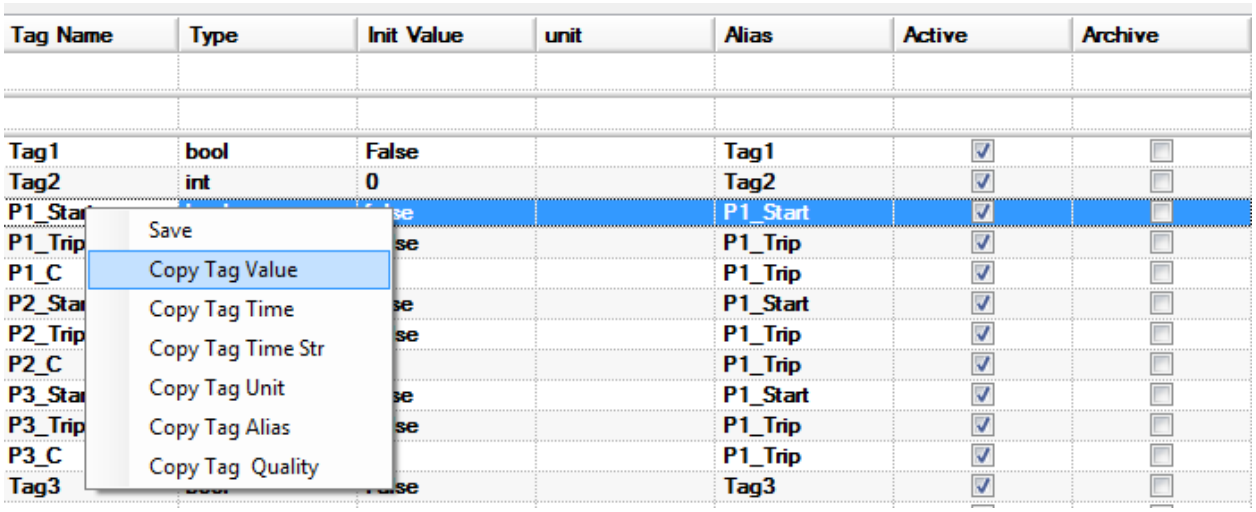
Archive : if checked tag changes will archive in Database which is selected in Tag Group Database field .

Cyclic = When checked, pbsHMI will log signal cyclic . For Cyclic logging, normal logging should be enabled too. Cyclic time is a parameter (cyclic Archive Period (Sec))

Log = When Checked, will make signal persistence. It means pbsHMI will save last value of signal in HDD and it will read signal value when pbsHMI load at startup.

Using tags

For using tags in pbsHMI , right click on tag which you want to use and click on “Copy tag xxx”.



It will copy full tag name in the clipboard for using in pages, scripts, trends,…

“FullTagName” : tag Value

“FullTagName” .v : tag Value

“FullTagName” .V: tag Value

“FullTagName”.t : tag Time in numeric format

“FullTagName”.T : tag Time in numeric format

“FullTagName”.t s : tag Time in string format

“FullTagName”.TS: tag Time in string format

“FullTagName”.q: tag quality

“FullTagName”.Q: tag quality

“FullTagName”.u:tag unit

“FullTagName”.U: tag unit

“FullTagName”.A: tag Alias

“FullTagName”.a: tag Alias

Any global tag and device tag has following attributes :

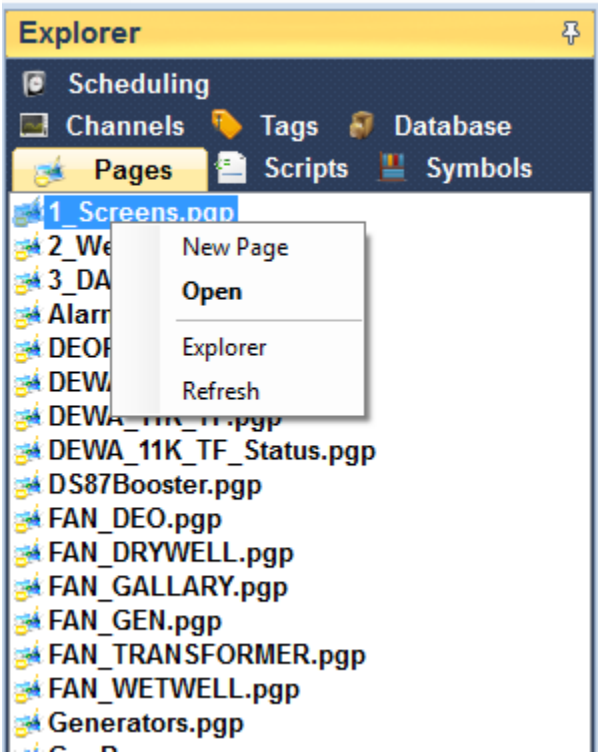
- Tag Value
- Tag Time (Numeric)
- Tag Time (String)
- Tag Quality
- Tag alias
- Tag Unit

You can use tag attributes everywhere in pbsHMI.

5 - Graphics Pages

pbsHMI is using XML format with PGP file extension for saving Graphic page .

In project explorer click on “Pages” tab. you can see list of all graphic pages. Right click in “Pages” tab , you will see following menu :



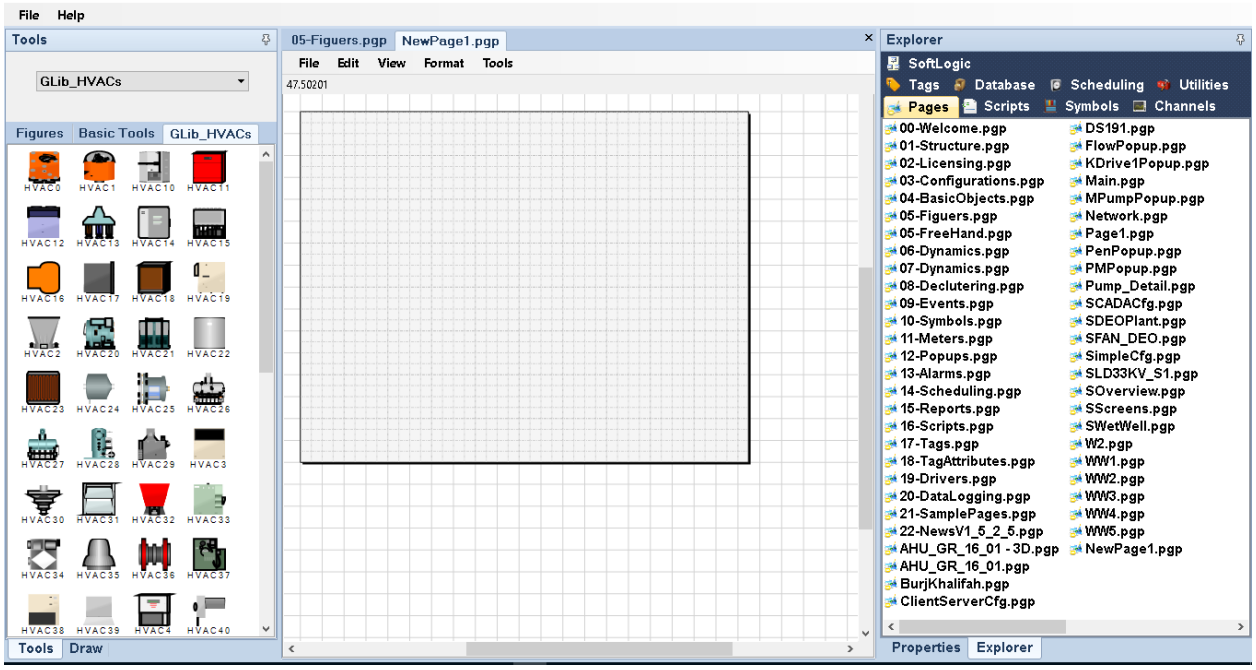
New Page Command: Create a new Page. By default its name is NewPage_x.pgp.

Open Command: open a created page.

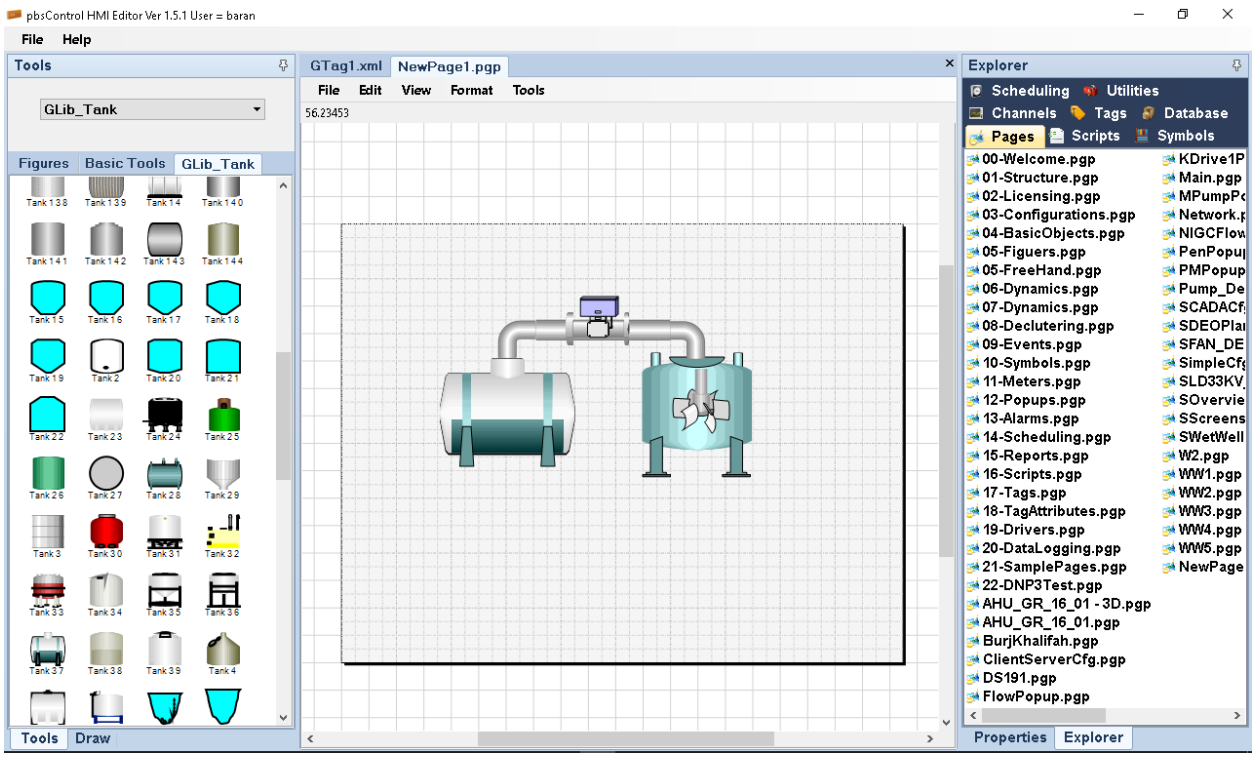
Explorer: Shows Graphic Pages directory for delete, Cut/paste and rename operation.

Refresh: refresh page list with latest Graphic Page Directory changes.

After you make a new page, you will see a blank page with NewPage_x.pgp adds to content area.

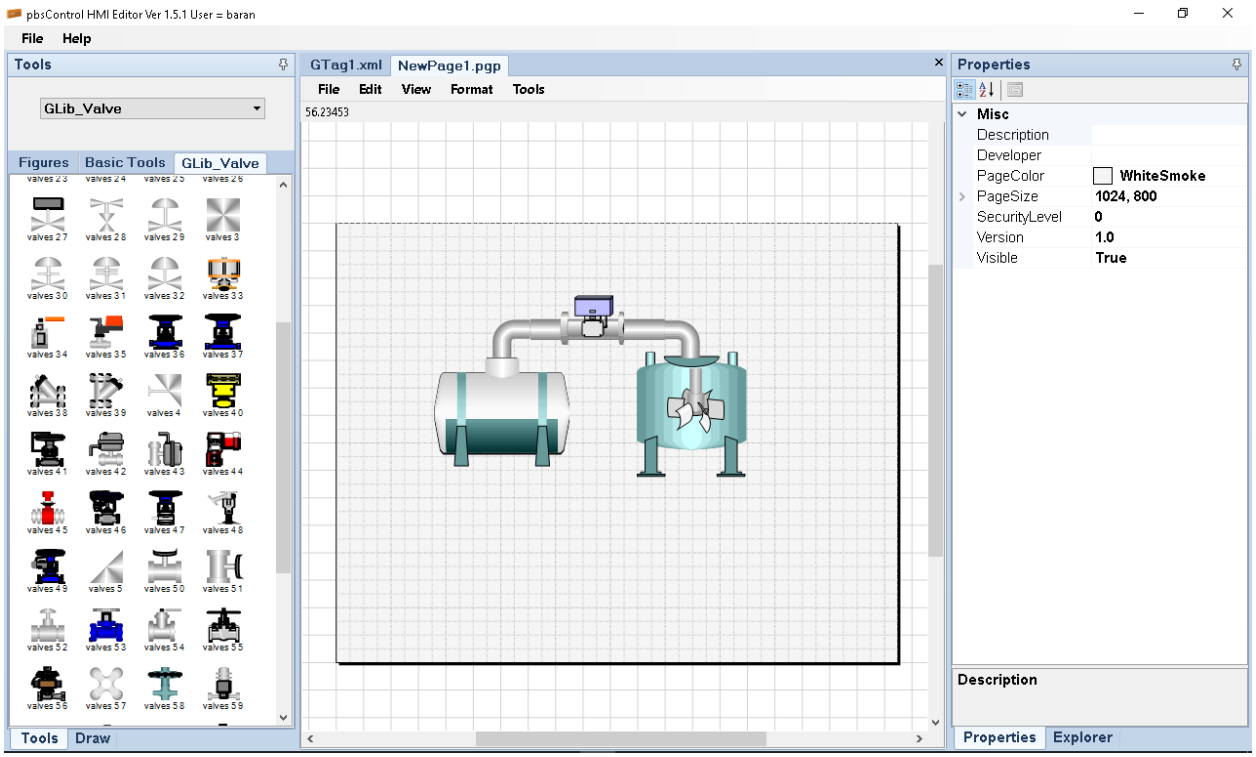


From Tools Panel, select a tool group and click on a component. Drag and Drop component to graphic page.

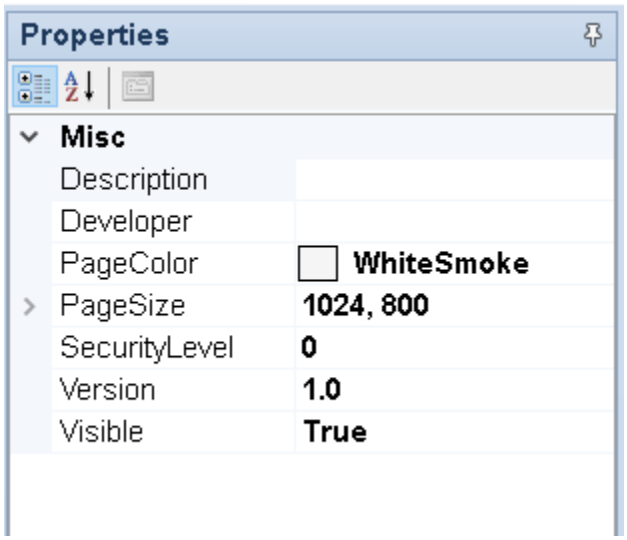


5 -1 Graphics Pages properties

Click on Page area and open properties window:



There are following properties for a page:



Description: Page description.

Developer: Developer Name.

Page Color: Page background color.

Page Size: Page size in pixel.

Version: Page Version.

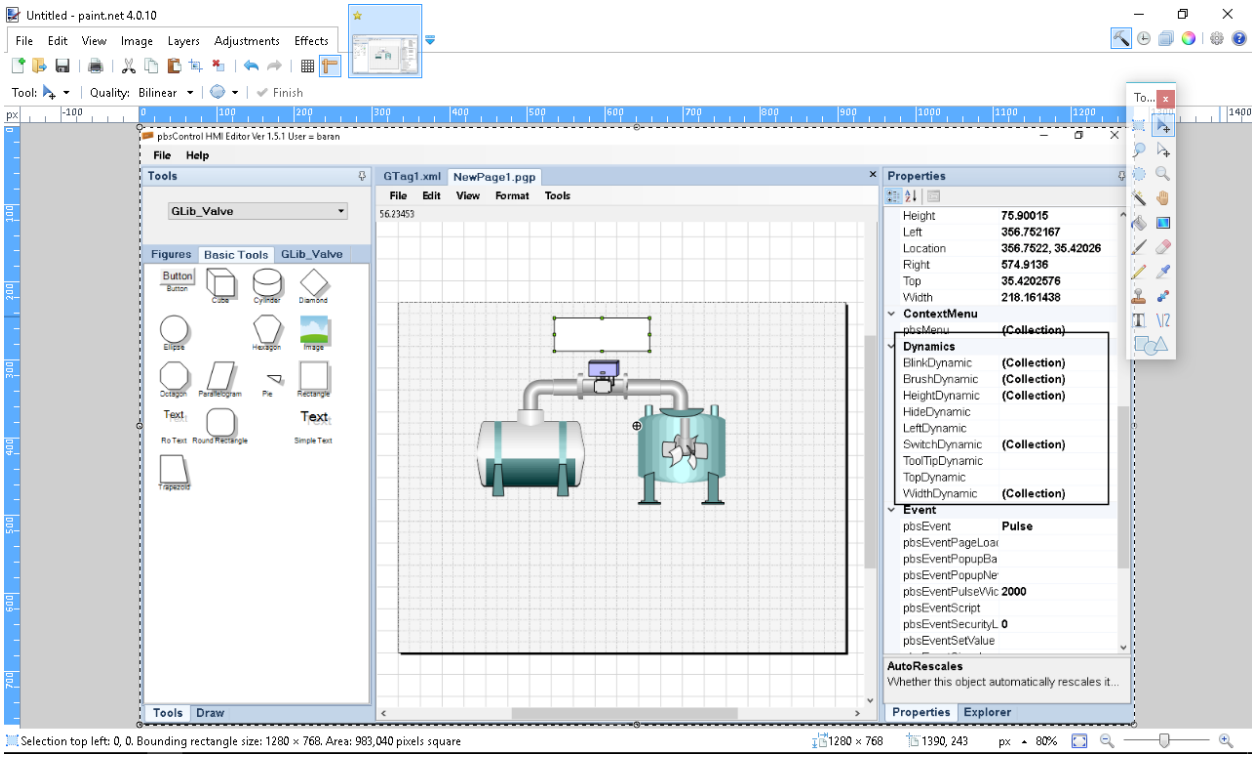
Visible: if True, page will show in Runtime at Page list. For popup pages you should change visibility to false.

SecurityLevel : Value between 0 to 1000 . Users with security level more than this value can see the page.

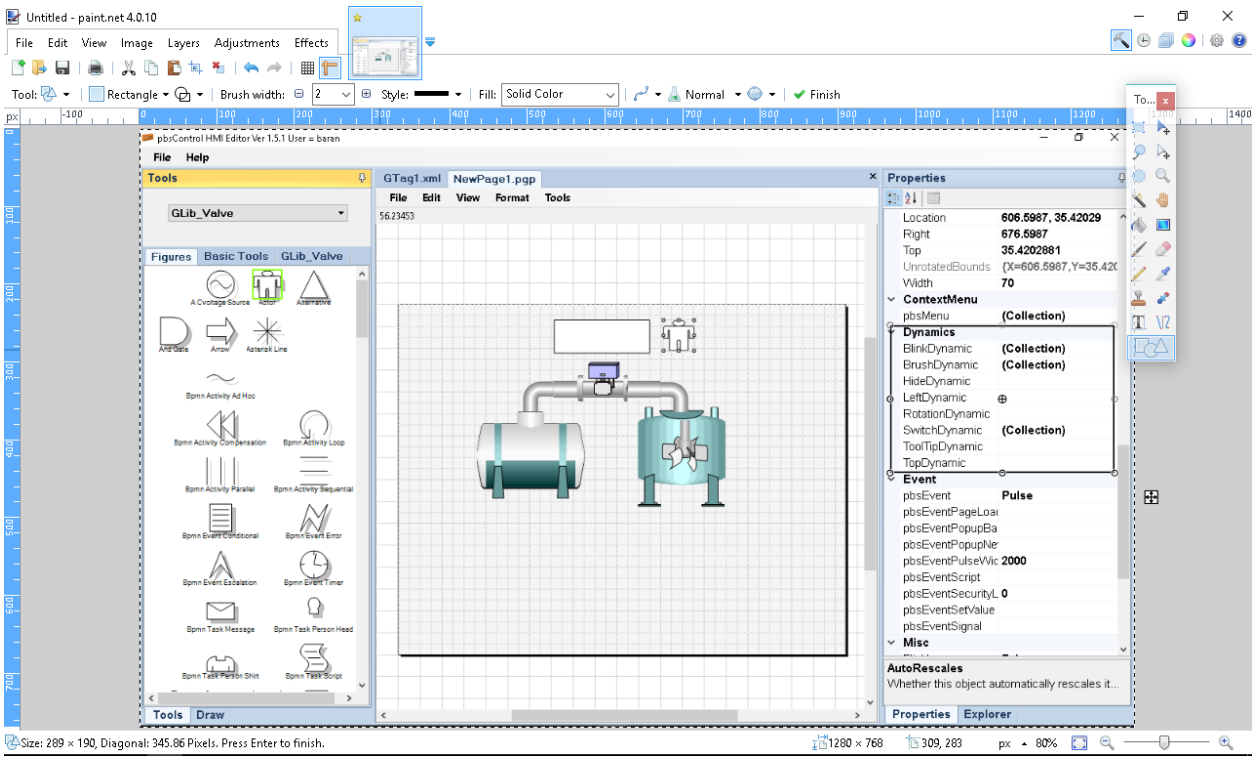
5 -2 Dynamics

For providing dynamic operation (Blinking, Brushing, Movement, Rotation, and Visibility) for graphic components, you can see Dynamic Part in property window. There are different dynamic groups for each component.

Dynamics for a rectangle:

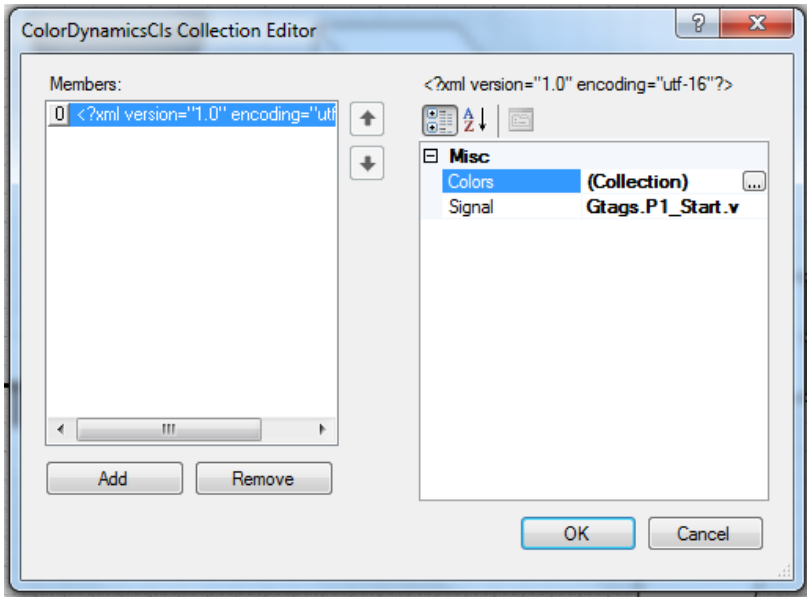


Dynamic for a figure:



5 -2-1 Brush Dynamics

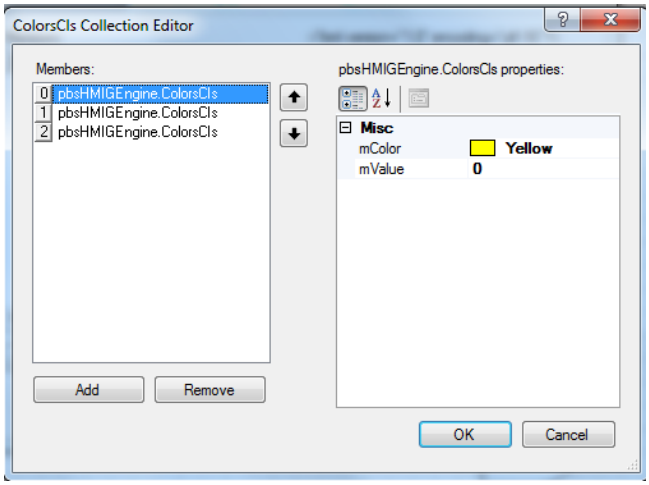
With Brush dynamic you can brush a component with different color at runtime with different value of a signal.

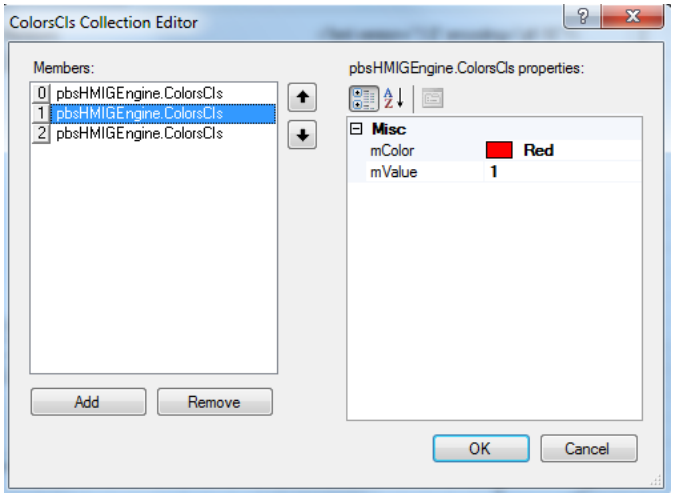


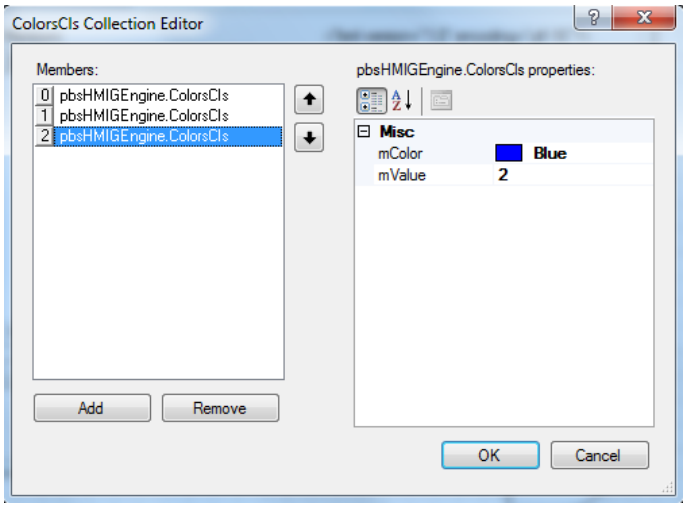
- At Dynamic part in properties window click on BrushDynamic
- Click on “Add” Button to add a new Brush Dynamic

- Write signal name .(Copy Signal Value from Device Tag or Global tag and paste in signal field)
- Click on “colors” Collection
- Click on “Add” Button to add a new Color .Select color and value of signal. Suppose you want to have following brush Dynamics :
    - If Signal Value == 0 then Brush color =Yellow
    - If Signal Value == 1 then Brush color =Red
    - If Signal Value == 2 then Brush color =Blue

Then you should add 3 colors as following:





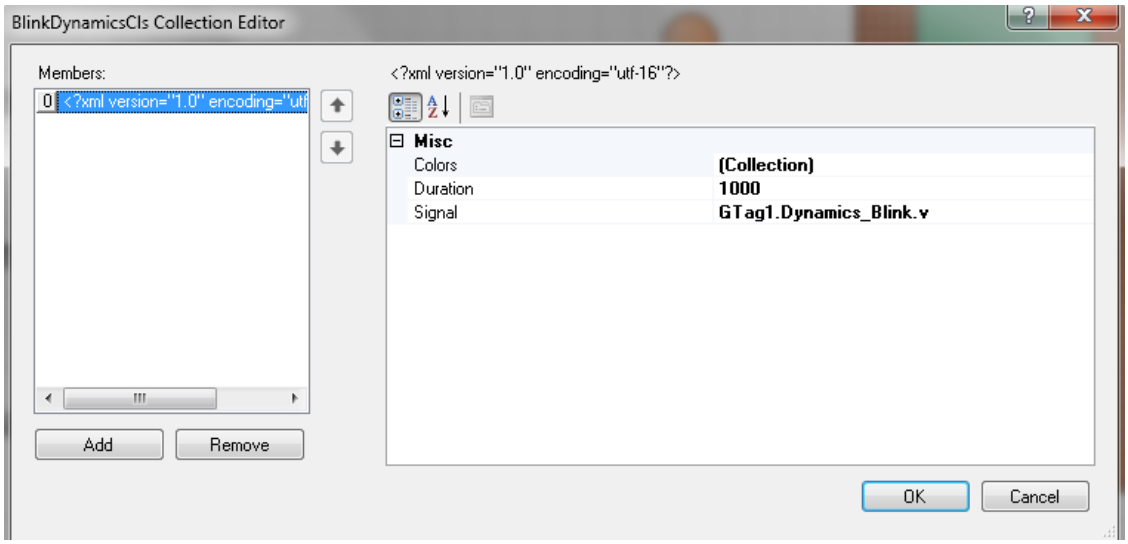


Click on “Ok” button and save page.

If Signal Type is digital (Boolean) , true is equal to 1 and false is equal to 0.

5 -2-2 Blink Dynamics

With Blink Dynamic you can make a blink on an object when signal is activated.

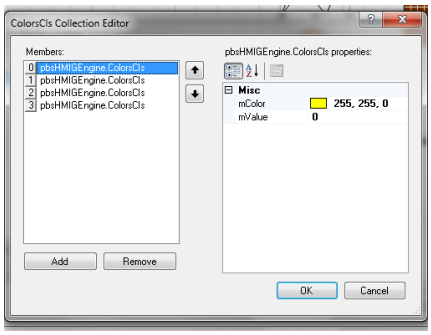


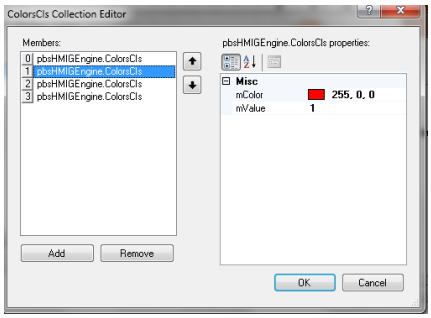
Signal : Device or Global tag for activating Blink

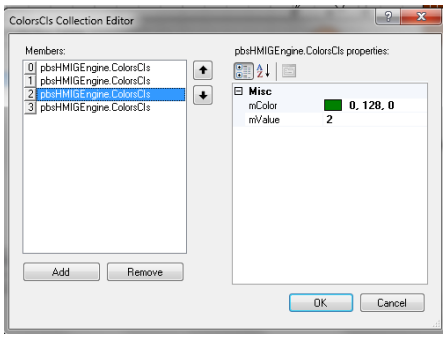
Duration (msec) : Blink Duration

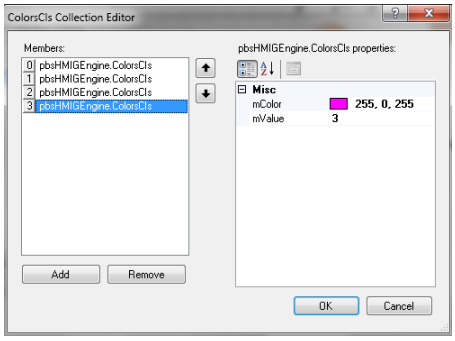
Colors : Sequence of colors that object with blink at Duration time

In following example object will change its color by Yellow, Red, Green and purple.









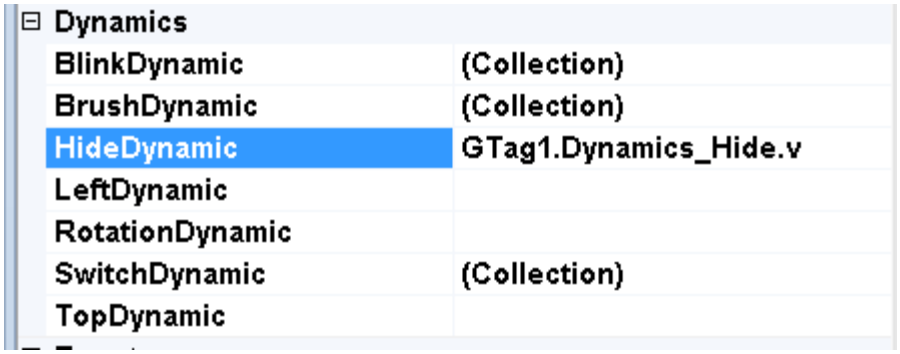
Signal Value :

Bool (True) , int (not equal to 0) : start to blink

Bool (False) , int (0) : stop blink , set brush to default brush of element .

5 -2-3 Hide Dynamics

With hide dynamics, you can hide an object by signal value.



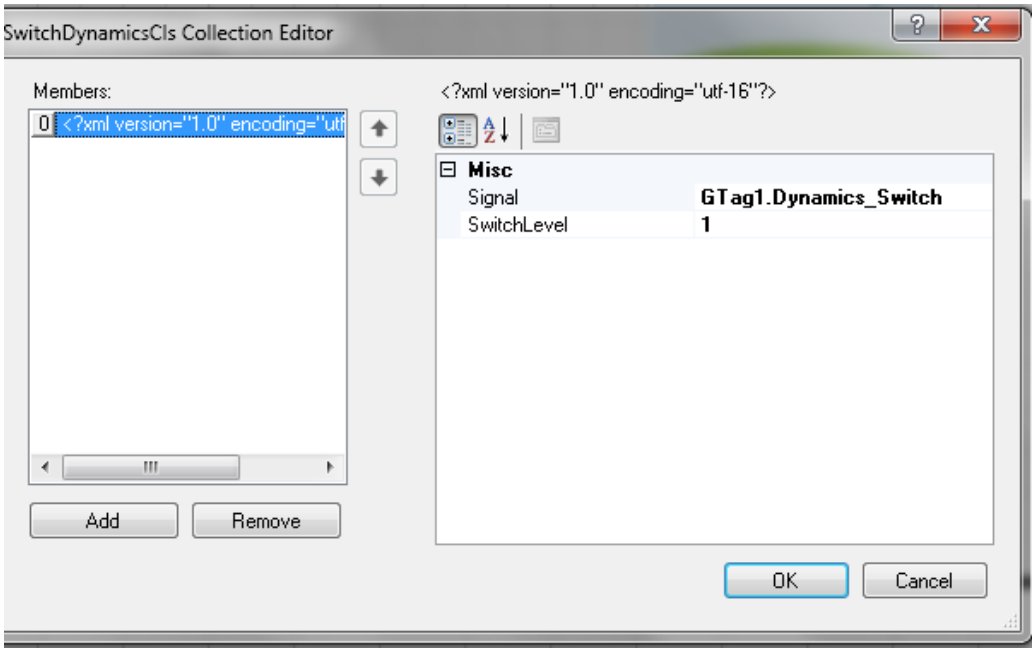
Signal Value :

Bool (True) , int (not equal to 0) : Object Hide

Bool (False) , int (0) : Object Visible

5 -2-4 Switch Dynamics

With switch dynamic, you can show one object and hide other objects with one signal. Switch Dynamic is operating on multiple objects.

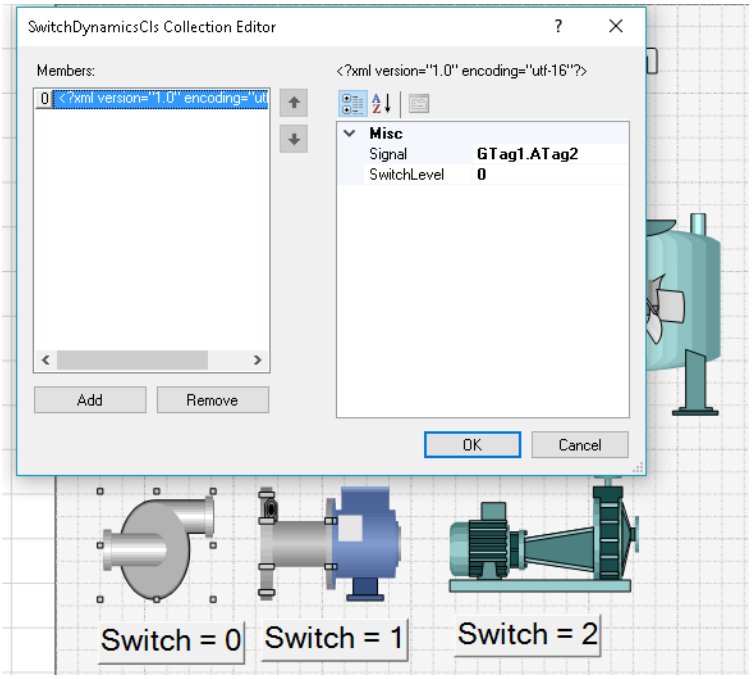


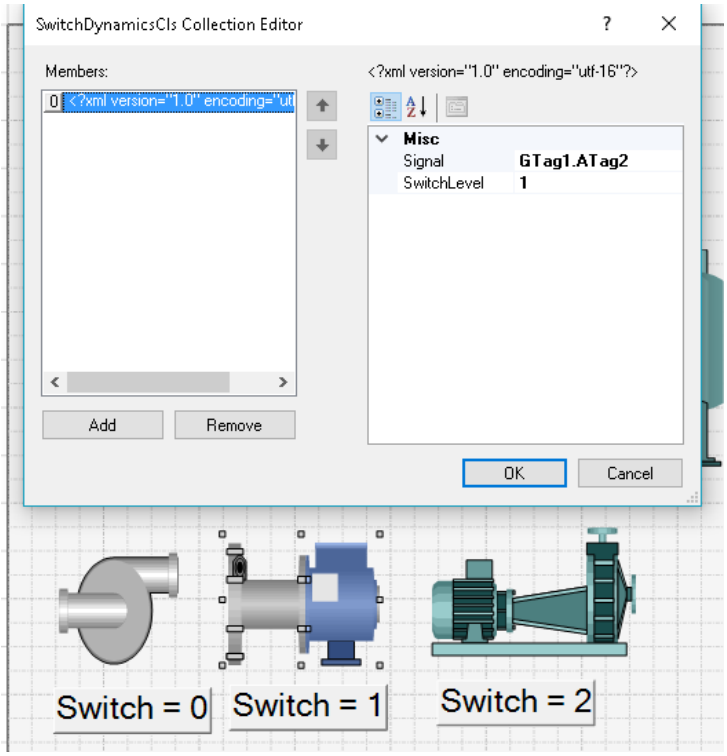
Switch dynamic is very powerful tool for page configuration.

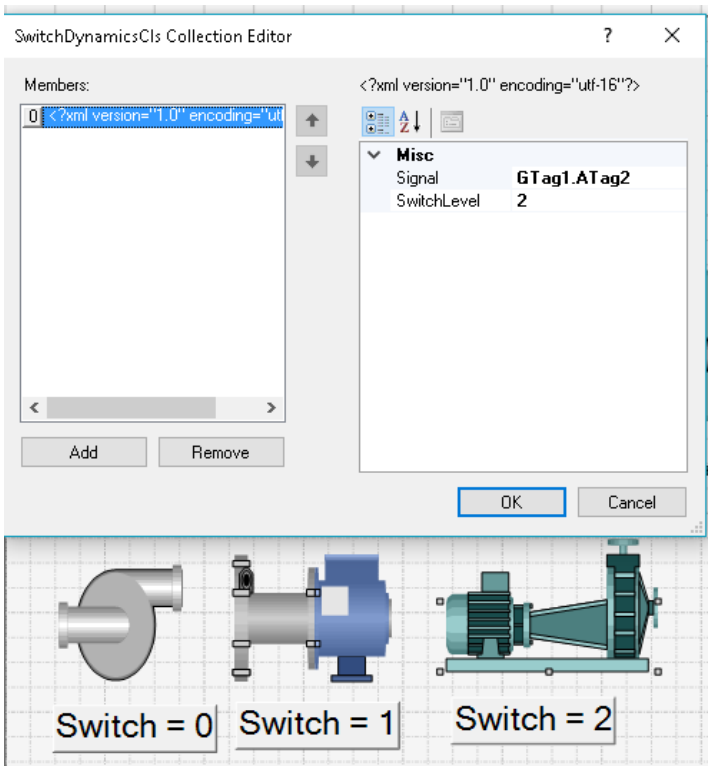
Switch Level : Value of Signal that Object is visible ,

Signal : int signal with different value . Object will show when Value of Signal and Signal level is identical otherwise will hide.

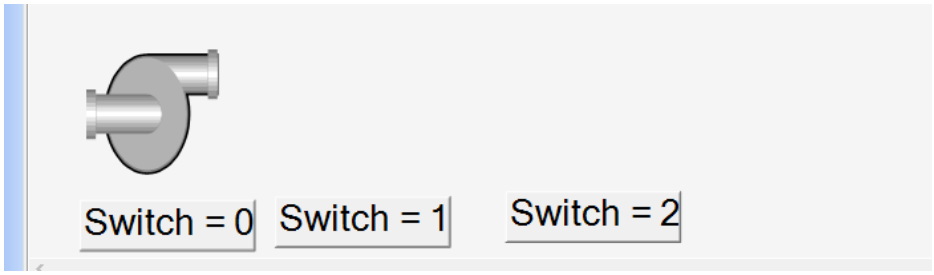
In following sample , three objects has same Switch dynamic signal with different Switch Value .



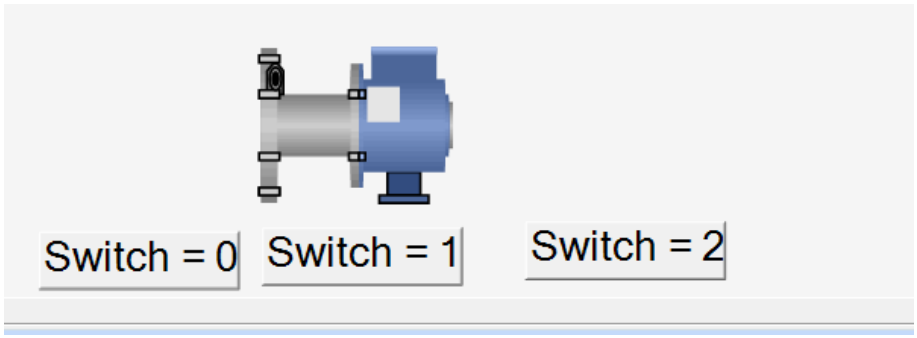




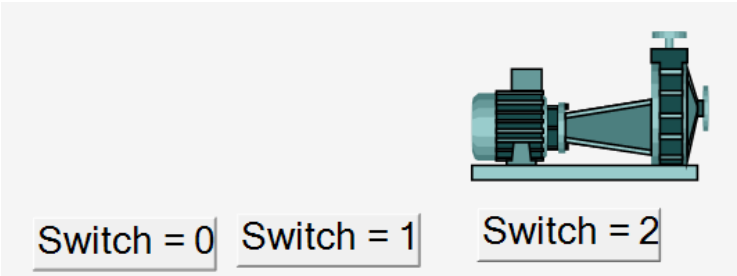
At runtime when value of GTag1.Tag2 is 0 , only object with Switch signal equal to 0 is visible and two other objects are hide .



At runtime when value of GTag1.Tag2 is 1 , only object with Switch signal equal to 1 is visible and two other objects are hide .



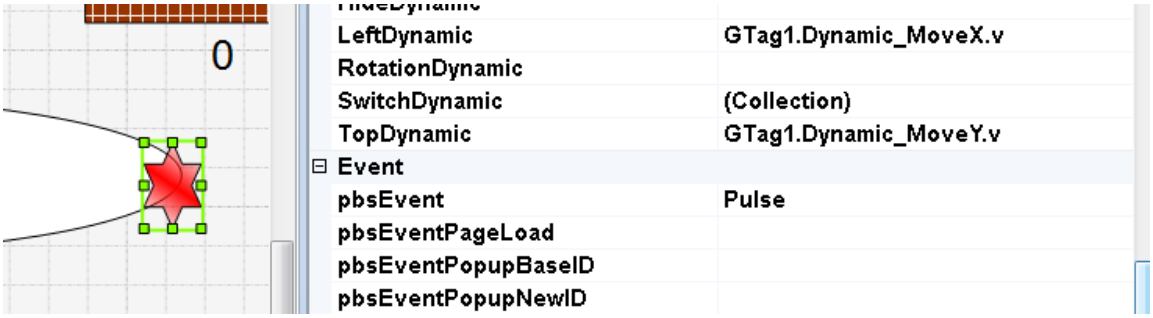
At runtime when value of GTag1.Tag2 is 2 , only object with Switch signal equal to 2 is visible and two other objects are hide .



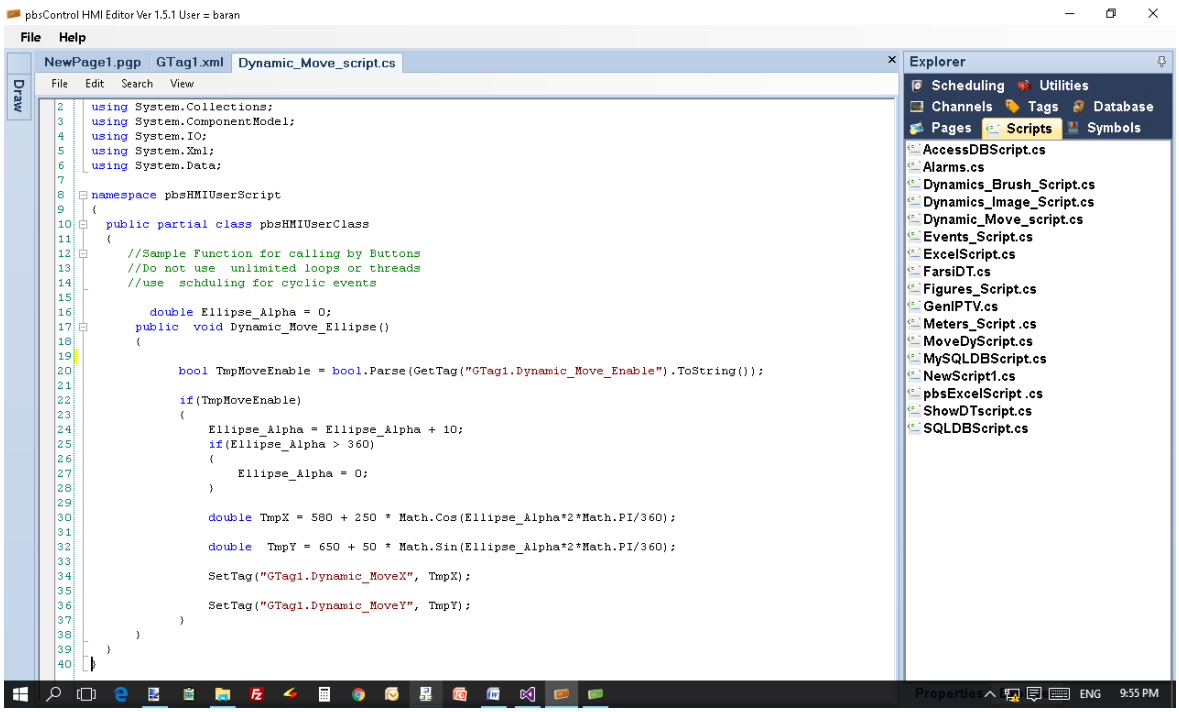
When GTag1.Tag2 is not equal to 0 or 1 or 2 all three objects will be hide .

5 -2-5 Left /Top Dynamics

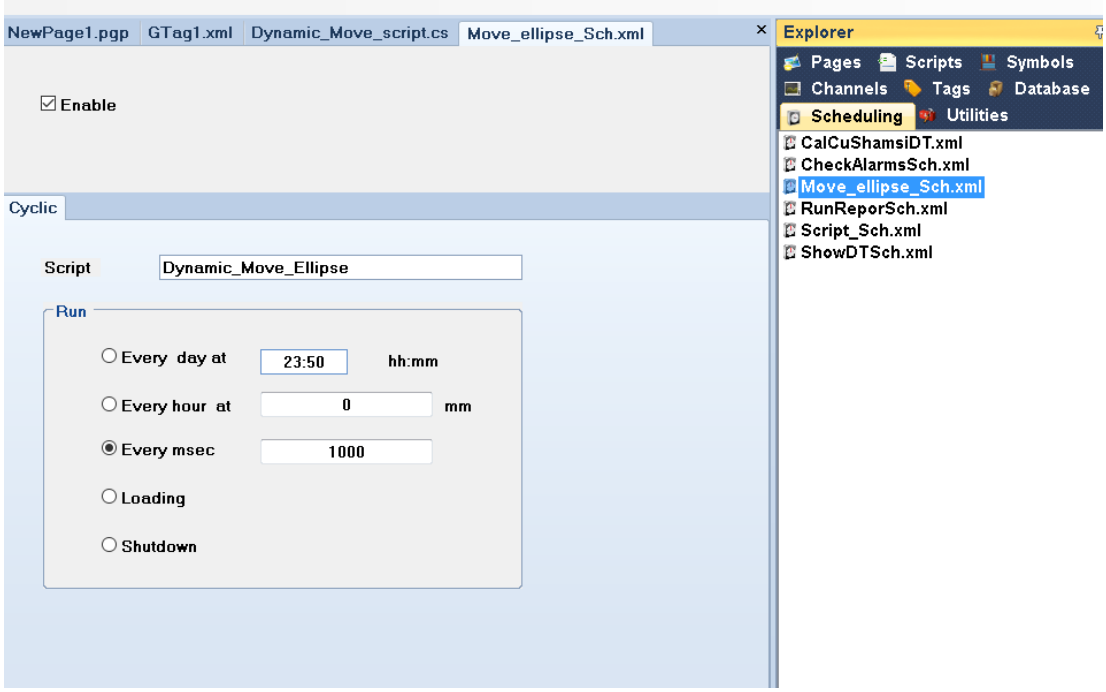
You can use Left and Top Dynamics for moving an object in the page. Unit Is pixel .



Following script is used to move above start in an ellipse shape.



above script is run every sec with following schedule object .



This schedule object is calling Dynamic_Move_Ellipse Function every sec .

In Dynamic_Move_Ellipse , because Ellipse_Alpha is defined as public variable , so its value will keep every time Dynamic_Move_Ellipse is calling by schedule object .

At each execution of Dynamic_Move_Ellipse , Ellipse_Alpha will increase by 10 and calculate TmpX and TmpY in pixel based on ellipse formula .

SetTag Function will set value of a pbsHMI Tags .

SetTag("GTag1.Dynamic_MoveX", TmpX);

SetTag("GTag1.Dynamic_MoveY", TmpY);

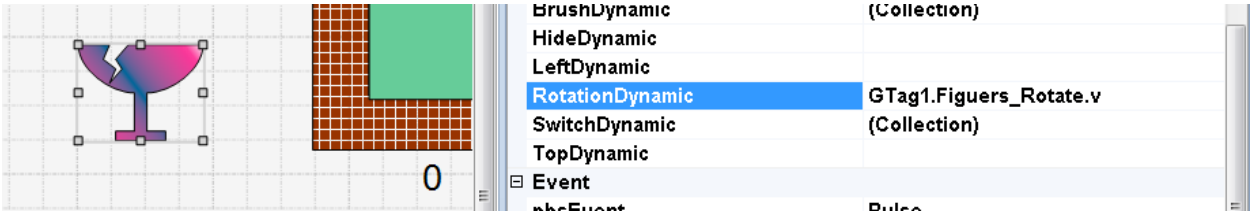
SetTag (string pTagName , object pValue) ;

Note : In pbsHMI you don't have direct access to Graphic page objects properties . Suppose you want to move an object , then you should do following :

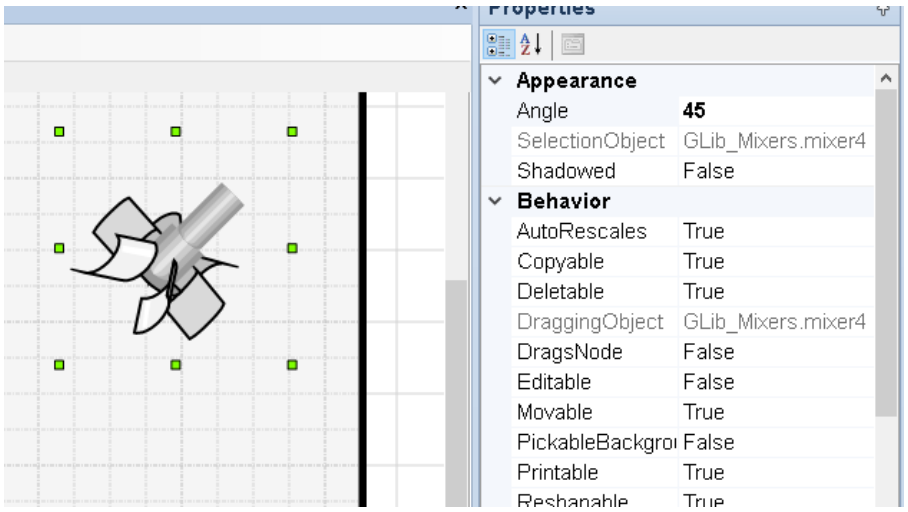
- Define a global Tag to show current Location of Object (Tag1_X)
- Write a script to change value of Tag1_X based on your movement
- Link Tag1_X to Left Dynamic of Object

5 -2-6 Rotate Dynamics

For all pbsHMI graphic objects you can use RotationDynamics for rotating object at runtime by a signal value.



Any Object has Angle property for static rotation of Object in pbsHMI Editor.

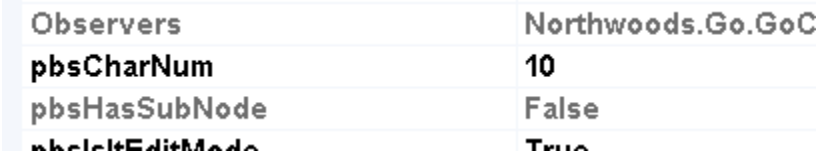


Rotation direction is clockwise when Angle Value is Positive.

5 -2-6 Text Dynamics

With text dynamic you can show numerical or string values on the page.

pbsCharNum : by default is 10 . It shows number of characters at runtime.

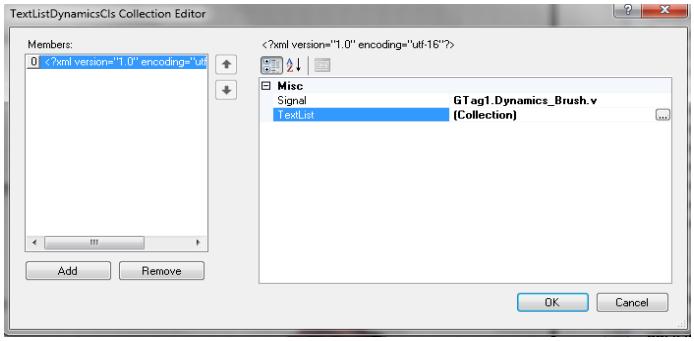


5 -2-7 Text List Dynamics

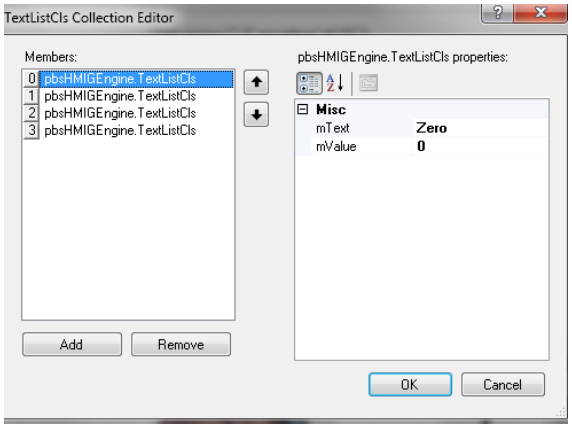
You can show strings which is mapped to signal value.

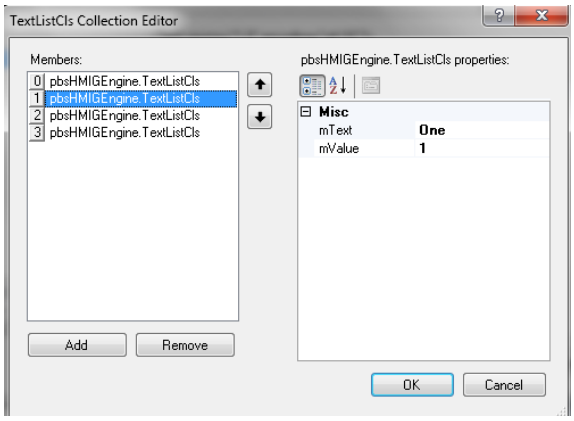
Suppose if value of signal = 1 it will show “One”

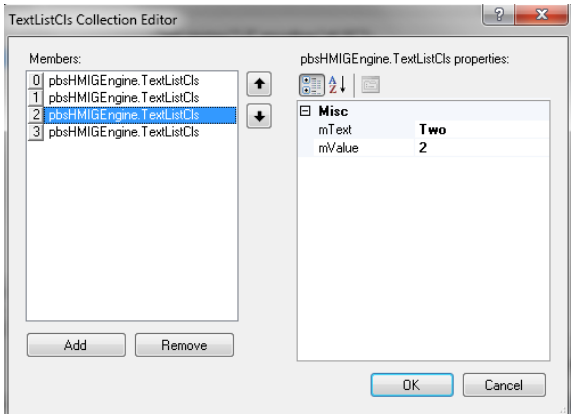
If Value of Signal is 2 it will show “Two”



Then you need following TextList :

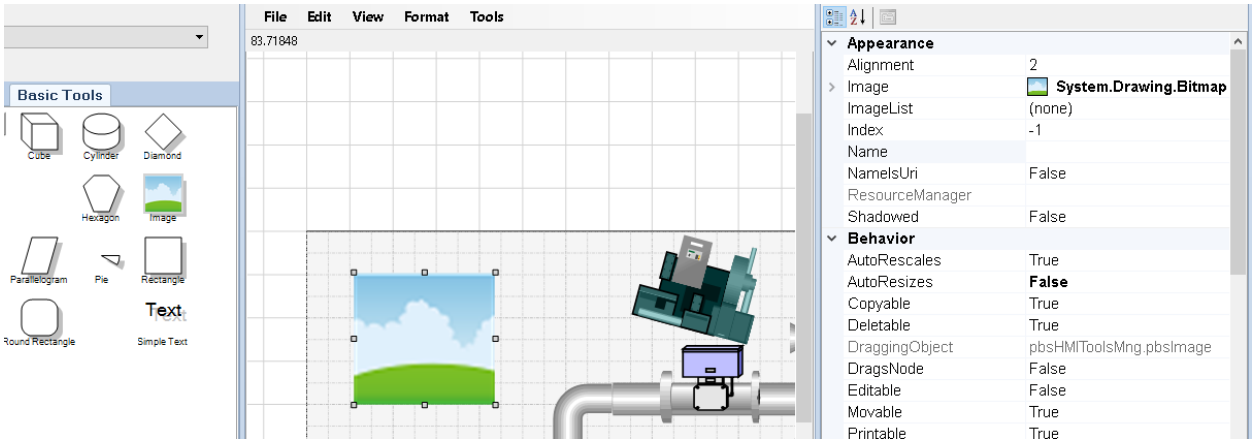




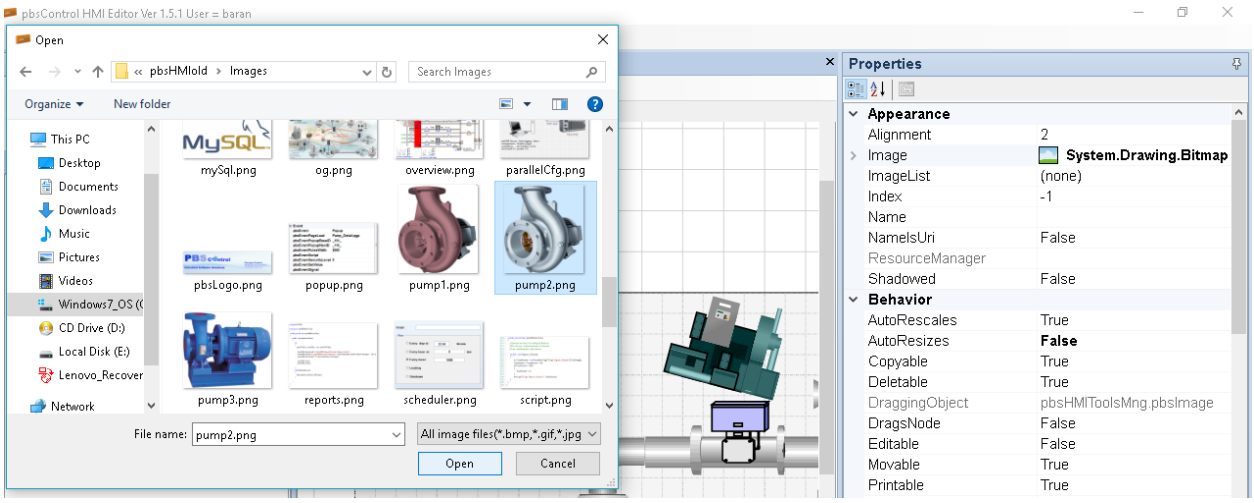


5 -2-8 Image List Dynamics

With Image List dynamic you can show different Images with different signal value. For defining Image List Dynamics , Drag and Drop Image Object from Basic Tools and link a static image to it by Image property .

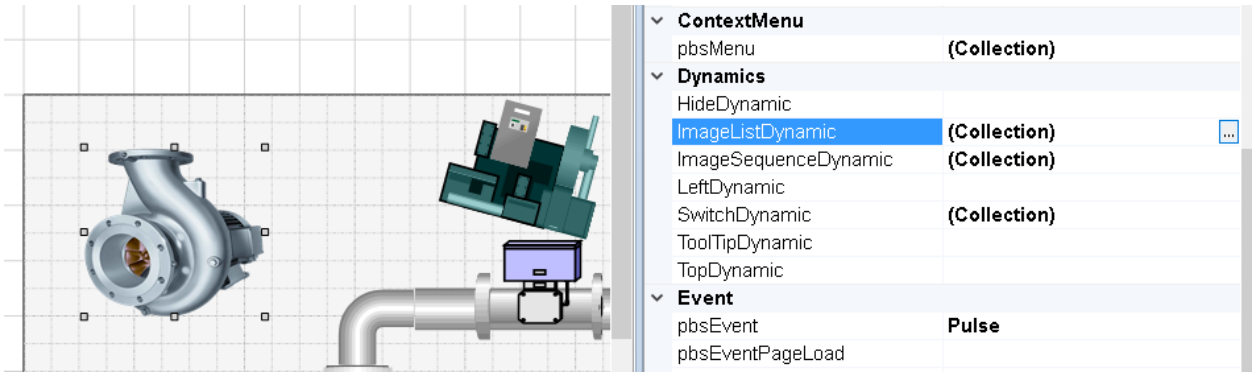


Click on Image property and select image. pbsHMI will integrate Image in page and no need to copy original image in target computer .

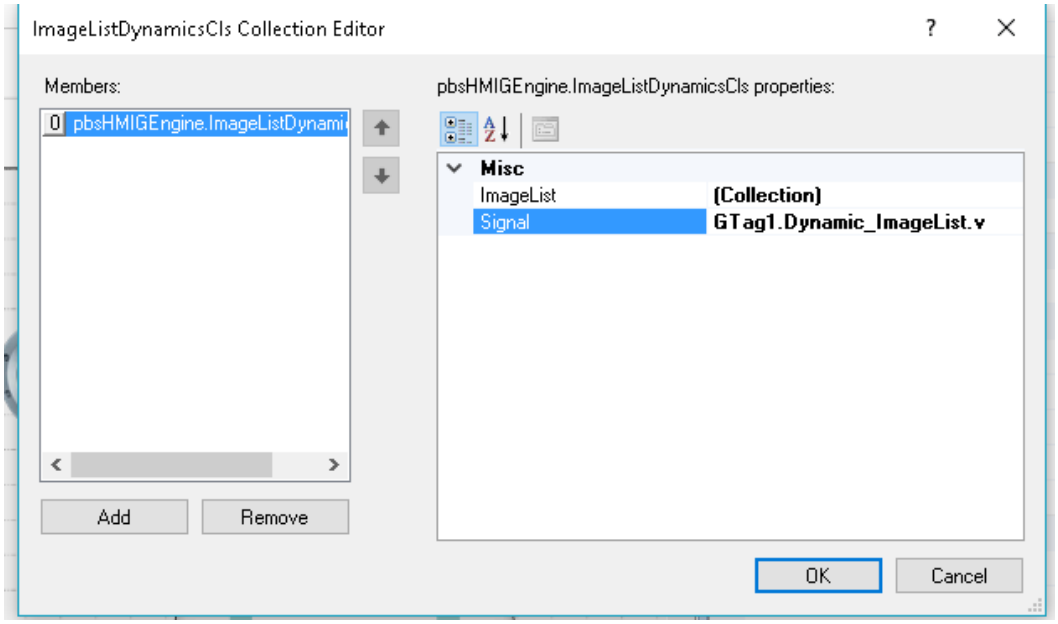


Select image and click on Open Button.

Select ImageListDynamic and click on … Button .



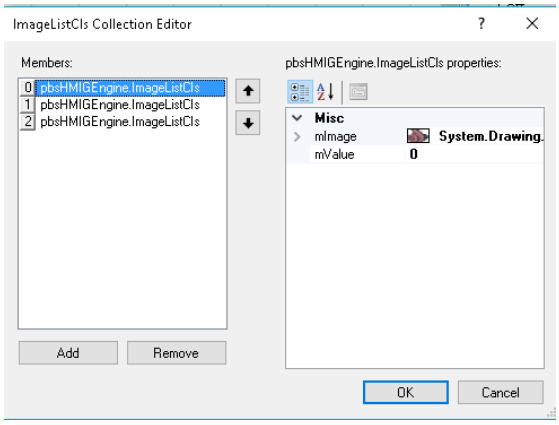
Click on add Button :

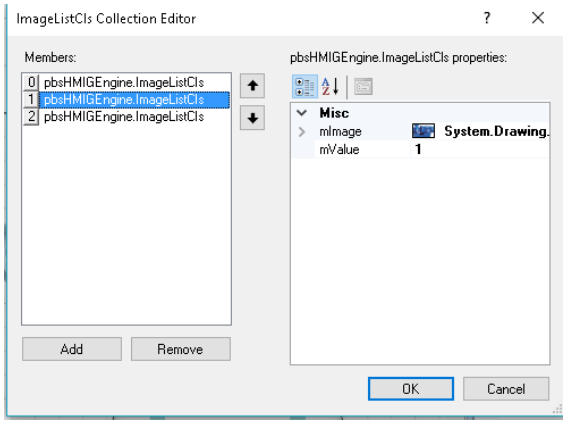


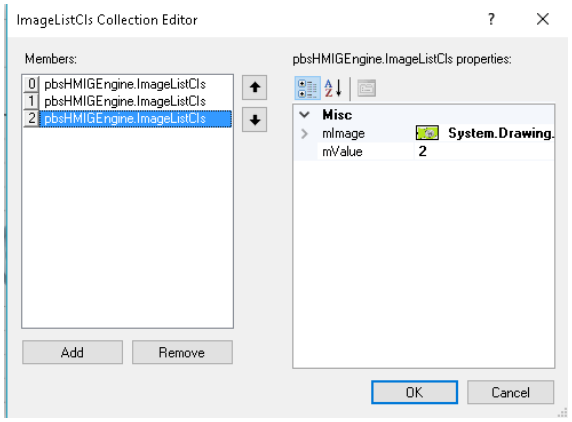
Signal: pbsHMI Global Signal or Device signal for changing Image . Based on Value of Signal , Image will changed .

Click on ImageList Collection and add One by One all images that you want to change by Signal Value.

For each image you need to set one Value. When Value of Image is equal to Signal Value, image will show.





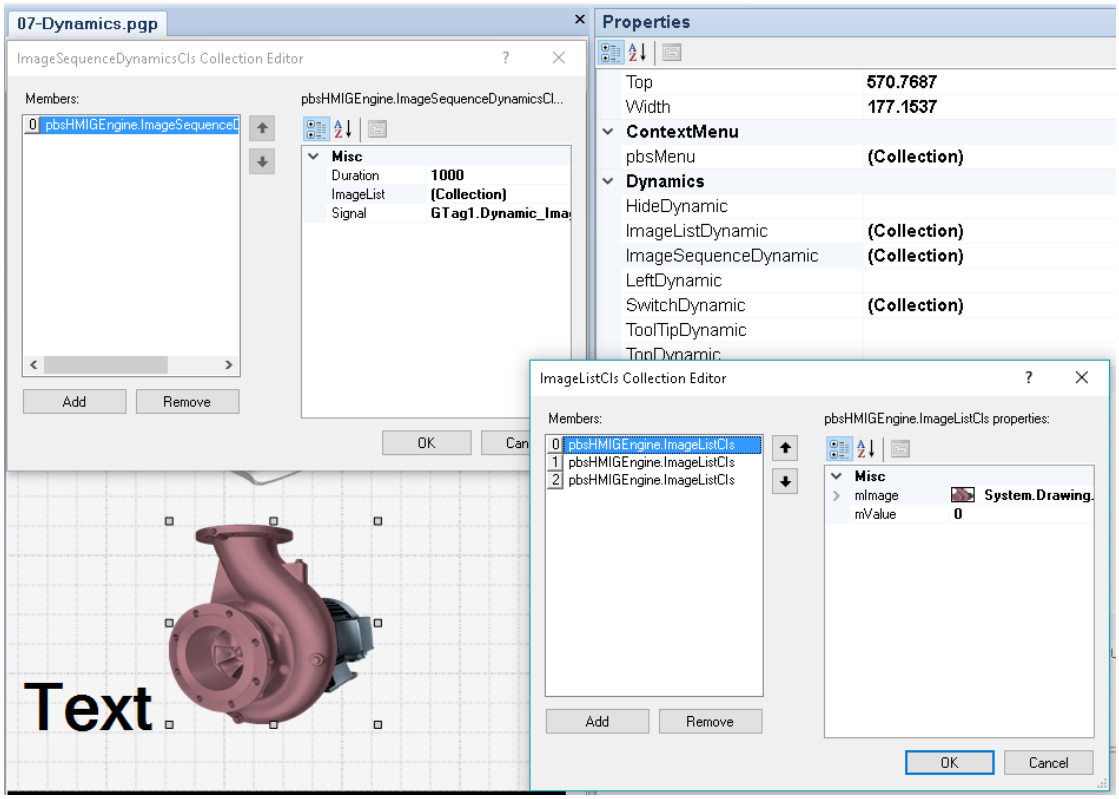


5 -2-9 Image Sequence Dynamics

With Image Sequence Dynamic, you can show sequence of images when signal is active. Signal can be int or bool .

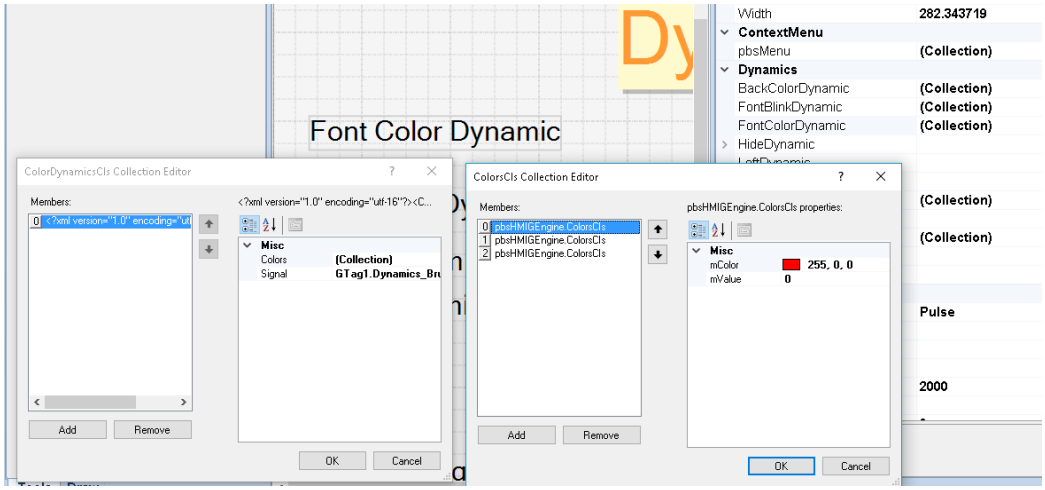
Value of image is not valid for image sequence dynamic.

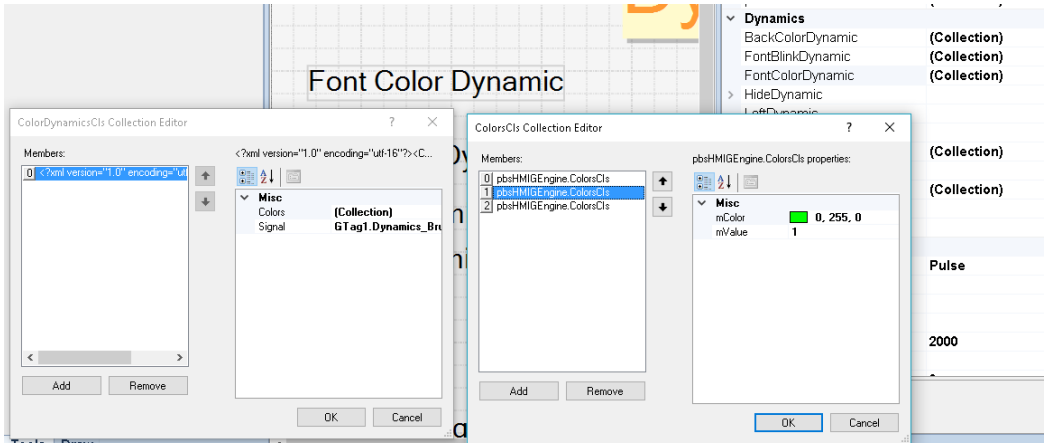
Configuration of ImageSequnceDynamics is same as ImageListDynamics only different is when value of Signal is True or not equel to 0 , then it is start to show Image Sequence based on Image sequence that you defined in configuration .

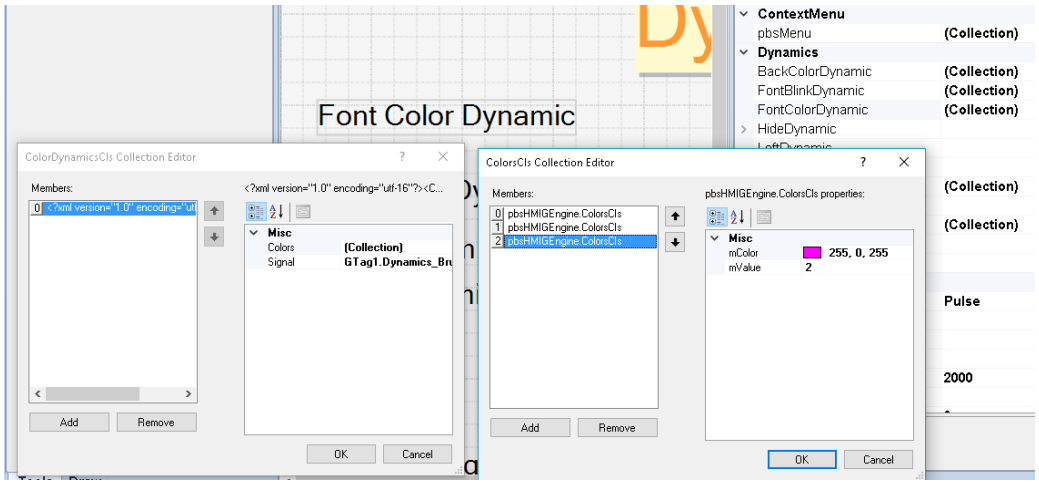


5 -2-10 Font Color Dynamics

With Font Color Dynamic, you can change Text font color by value of signal.

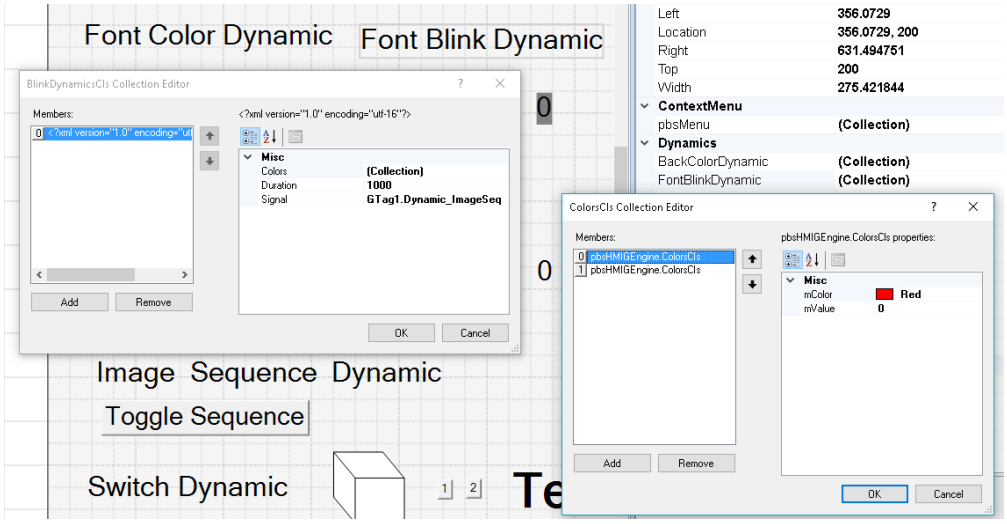


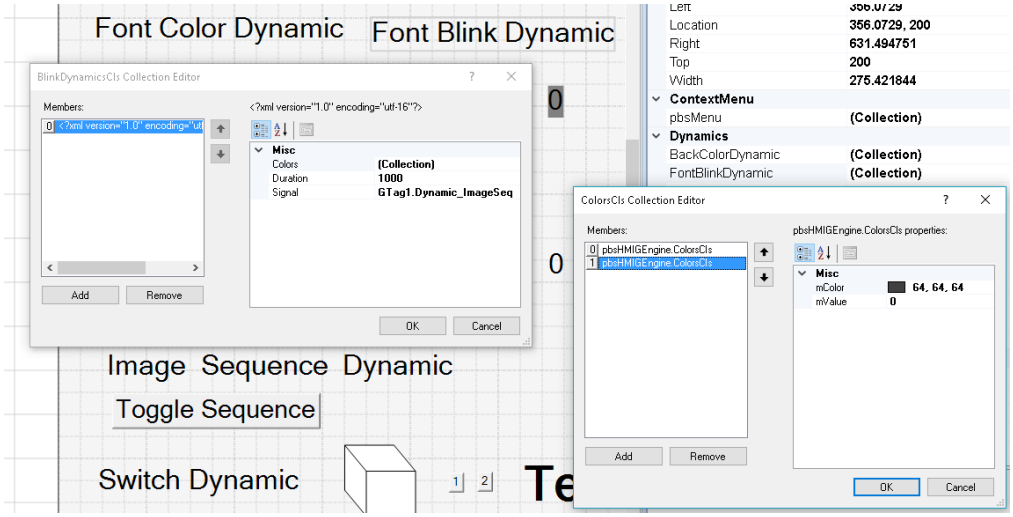




5 -2-11 Font Blink Dynamics

With Font Blink Dynamic, you can blink a text when Signal is activate.





5 -2-12 Font Back Color Dynamics

With Font Back Color Dynamic, you can change back color of a text with signal value.

TransparentBackground should be false.

5 -3 Events

In pbsHMI , each graphic object has Events and context menu . Events will activate by left click and Context Menu Is right click menu.

There are following Events in pbHMI :

- Set Event : Set Event Signal to predetermine value
- Toggle Event : Toggle value of signal
- Pulse Event : make a pulse
- Script Event : run a script
- Load Event :load a graphic page
- Popup Event : load a popup page

You can use same events for context menu.

There are Event and Context Menu section in properties of a graphic symbol.

⊟ **ContextMenu**	
pbsMenu	**(Collection)**
⊟ **Dynamics**	
BlinkDynamic	**(Collection)**
BrushDynamic	**(Collection)**
HideDynamic	
LeftDynamic	
RotationDynamic	
SwitchDynamic	**(Collection)**
TopDynamic	
⊟ **Event**	
pbsEvent	**Set**
pbsEventPageLoad	
pbsEventPopupBaseID	
pbsEventPopupNewID	
pbsEventPulseWidth	**2000**
pbsEventScript	
pbsEventSecurityLevel	**0**
pbsEventSetValue	**10**
pbsEventSignal	**GTag1.Events_Set**

pbsEvent : Select one of pbsHMI events .

pbsEventPageLoad : name of page which will load by Load event .

pbsEventPopupBaseID : refer to popup pages

pbsEventpopupNewID : refer to popup pages

pbsEventpulseWidth : pulse width for Pulse Event In msec .

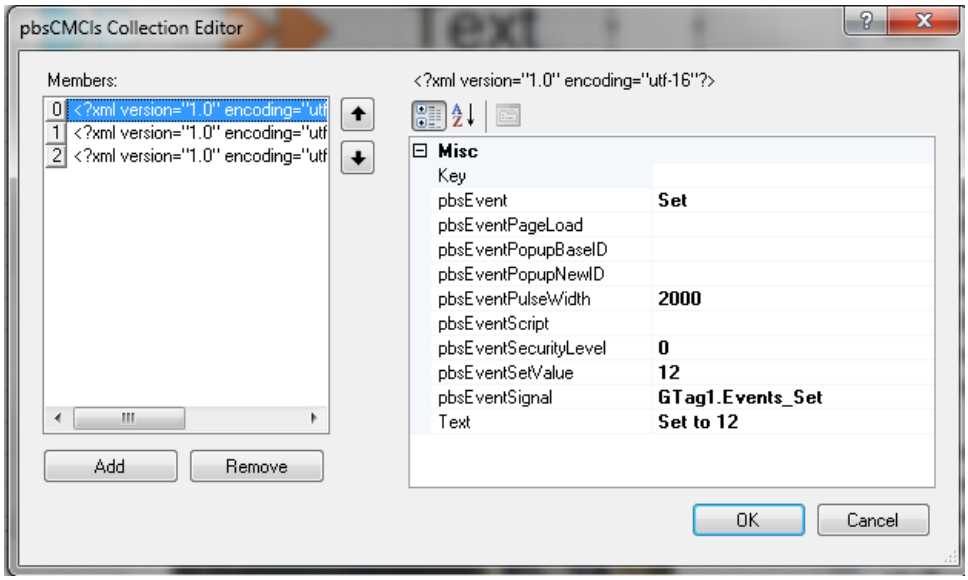
pbsEventScript : name of c# function (without ()) for running by script event .

pbsSecurity Level: a number between 0 to 1000 . this event will be activated for users with higher security level .

pbsEventSetValue : fix value for Set Event .

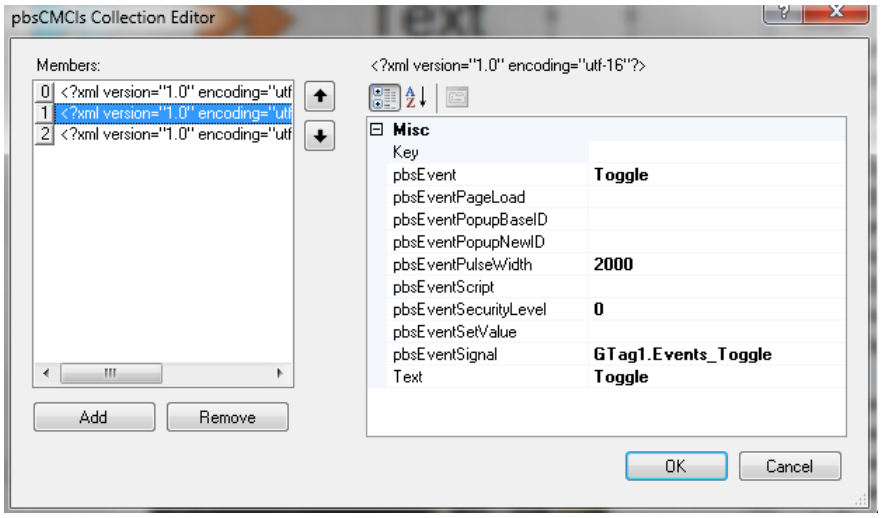
pbsEventSignal : name of signal which will link to Event for Set , Toggle and Pulse Events .

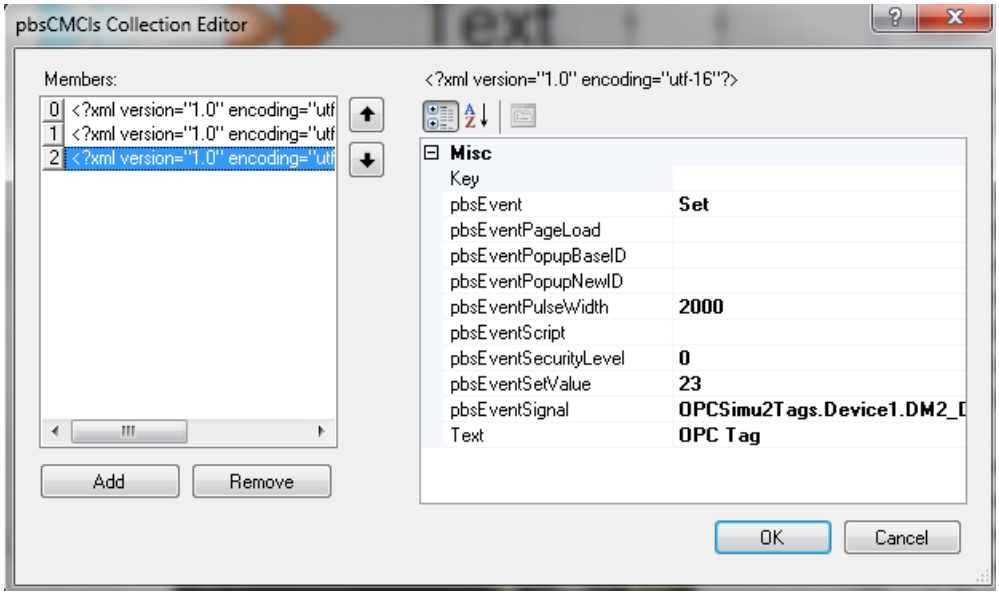
Context menu: in Contextmenu Groups open pbsMenu and add any number of menu item as following :



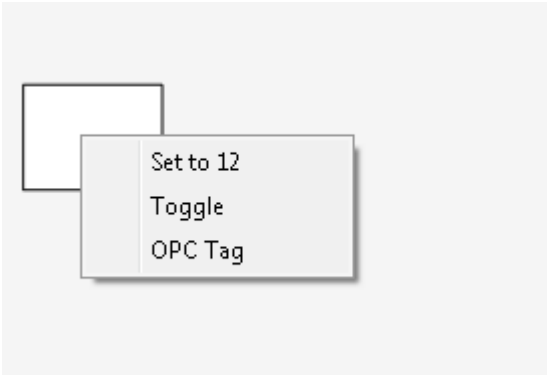
Key : unique ID for changing menu properties by Script in symbols . like Enable/Disable and checked menu . This functionality is just valid in symbols .

Text: text which will show on Context menu.





When you run pbsHMIView , for a graphic object with above context menu configuration , when you right click on object , you can see following menu :



When you click on any context menu, configured Event will be activating.

For changing Context menu properties, please refer to User Designed Symbol section.

User Designed Symbols

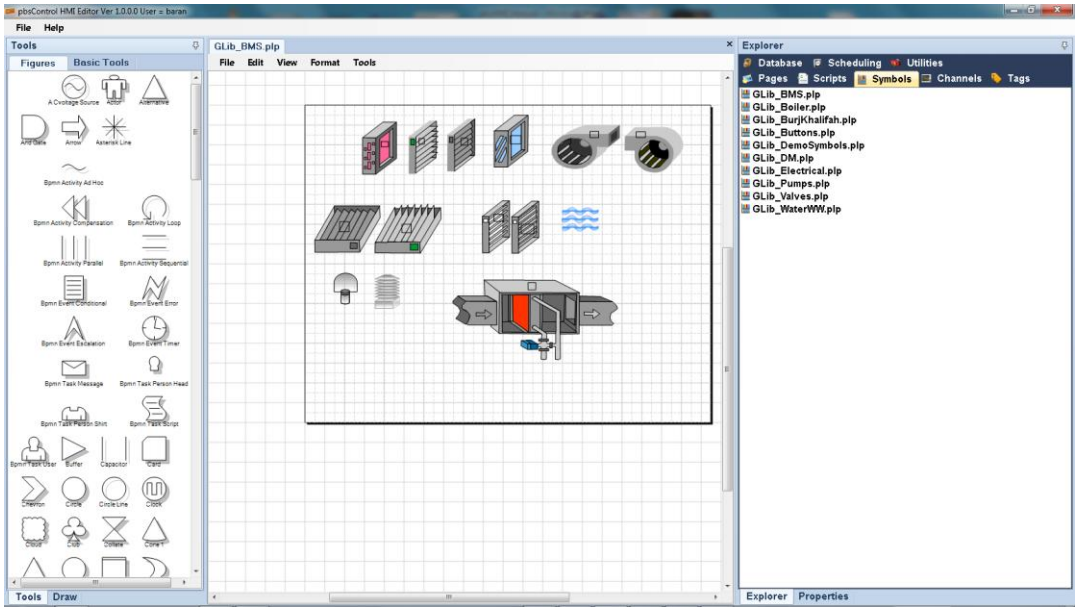
pbsHMI supports symbols for fast development of Graphic pages . Symbols have following specifications:

- Reusable graphic objects
- Symbols will add to whole platform and it is not for a specific project
- Symbol is an object with graphic, Script and dynamics.
- You can define Public and Private Properties for Symbols.
- You can link signals to Public Properties and Private properties are local for Symbol.
- You can define Context menu for Symbols and change menu properties by Script

pbsHMI includes Symbol editor internally for making new symbols by user .

Symbol group: you can define many symbols inside a symbol group.

In pbsHMIEditor , click on Symbol tab and right click for making a new Symbol . You can open existing Symbol libraries. In following figure, BMS Library is opened:

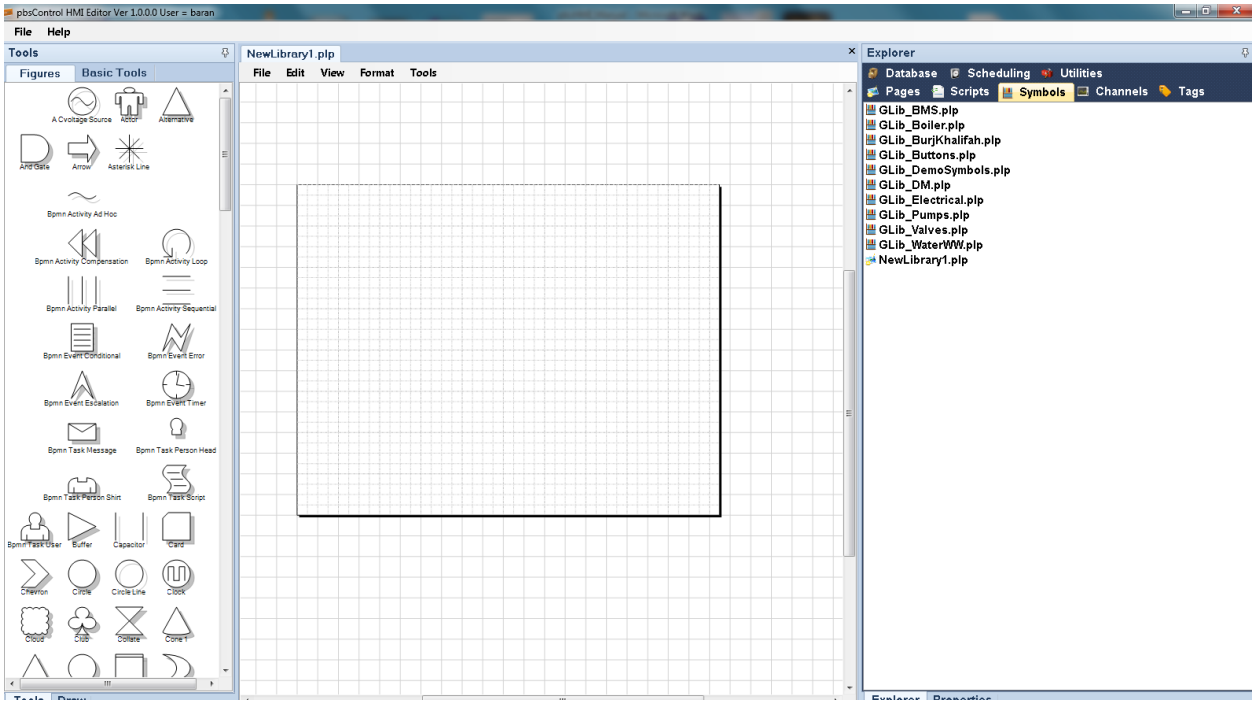


You can change functionality of each symbol and compile symbol again.

Final output of Symbol Editor is a Dot Net DLL which will load automatically by pbsHMI at next booting.

Limitation: Symbol elements are basic pbsHMI elements and it is not possible to use symbol inside symbol. You can use basic elements, figures, polygon, strok and free hand tools for designing symbols.

Making new symbol group: in pbsHMI Editor, select Symbol Tab, right click and select make new library.



pbsHMI Editor will make a blank library with name newLibrary{n}.plp

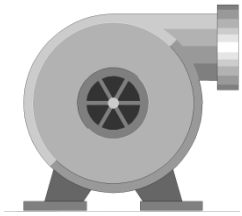
Save library and close it. Rename Library to any name you want. Based on pbsHMI standard, it is better to start library by “Glib_” prefix.

Change new library to “GLib_Cooling.plp“and refresh Library list.

Open Glib_Cooling.plp by double click on Glib_Cooling.plp name.

Symbol #1: Cool Pump.

Suppose we want to make following symbol as Cool Pump.



Cool Pump has following properties:

State : Public , type = int , Init Value = 0

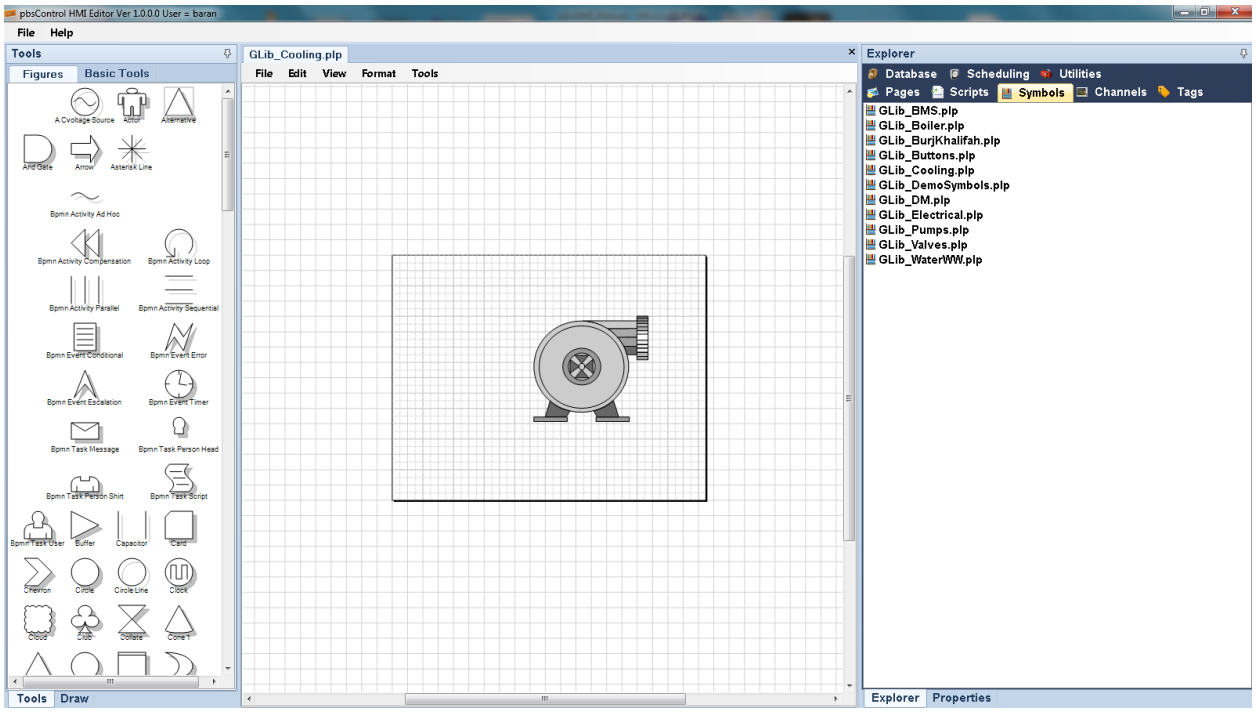
When state=0 , pump is off , color = gray

When state=1 , pump is ON , color = green

When state = 2 , pump fault , Color = Red , blink

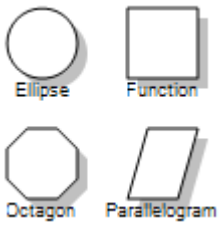
Use basic elements and strok tool for making above symbol.

Step 1 : with help of circle , rectangle and strok tool you can draw above figure like following shape :

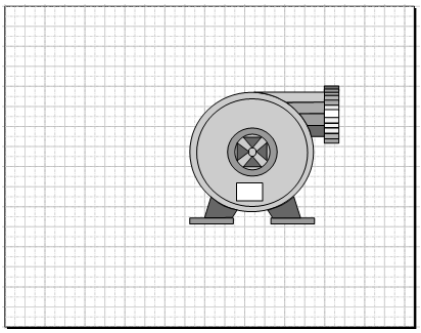


Step 2 : In Basic tool , use Function tool and drag it inside the pump shape .

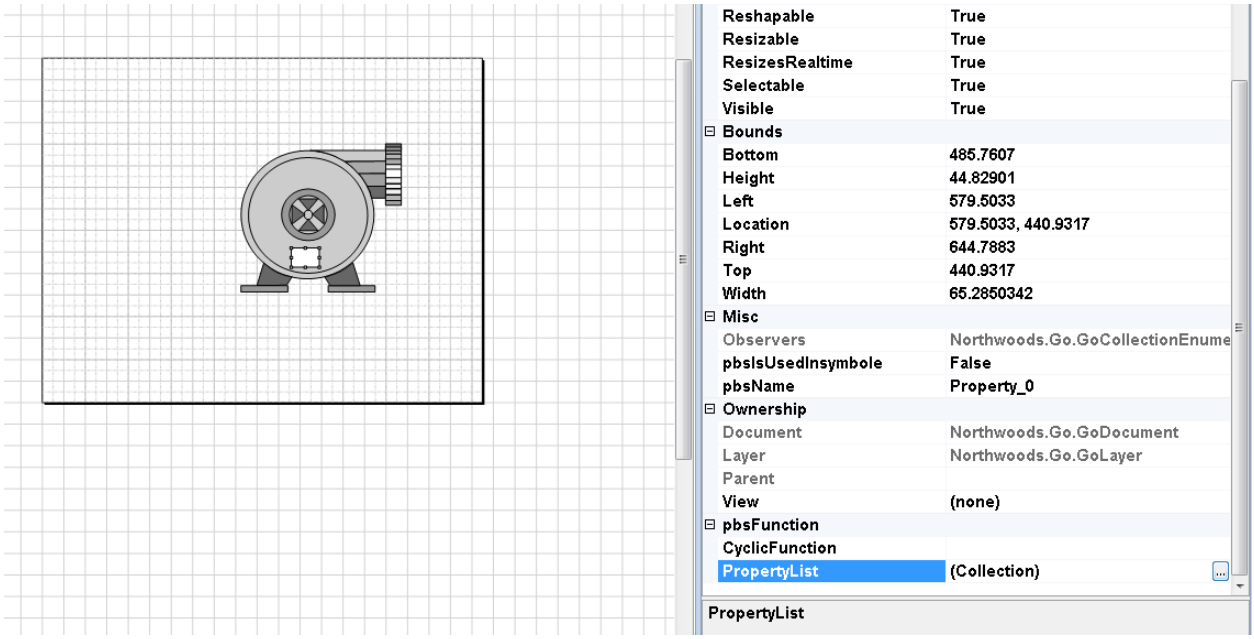
With Function element, you can define properties and write script for symbol.



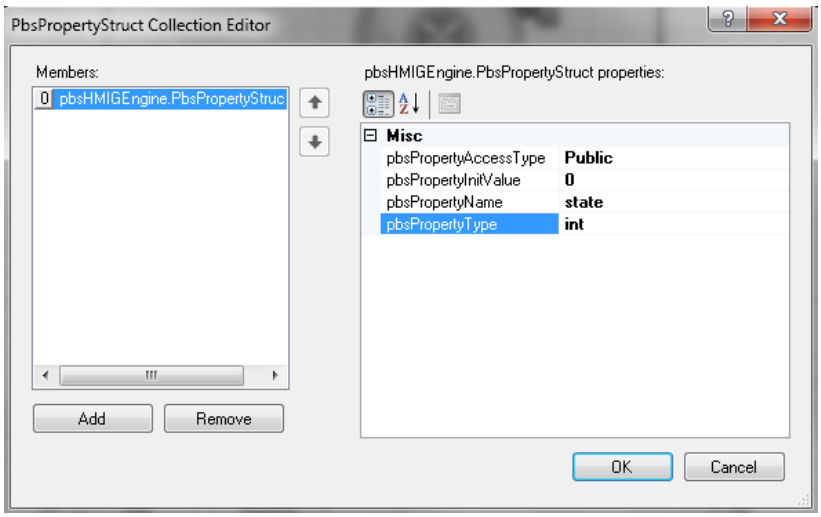
You can put Function element everywhere in pump symbol because it will be hiding at runtime.



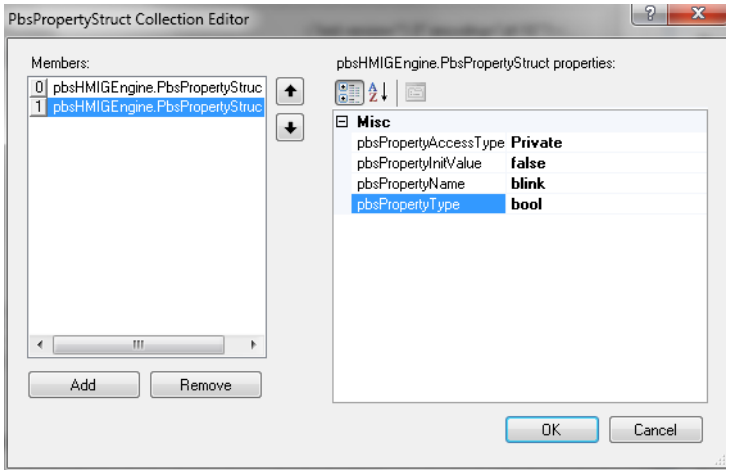
Step 3 : Click on Function Element and define symbol properties at Property List :



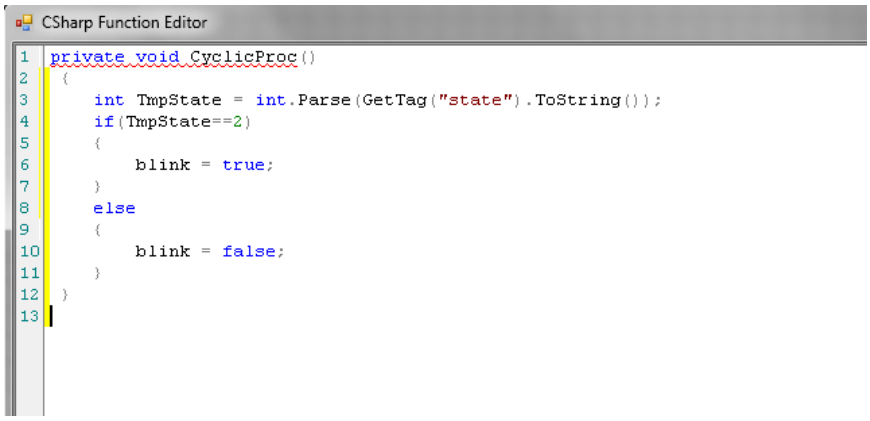
Open PropertyList and define state property as following L



Step4 : define another private property as following :



Step5 : open CyclicFunction property of Function element and write following C# Code there :

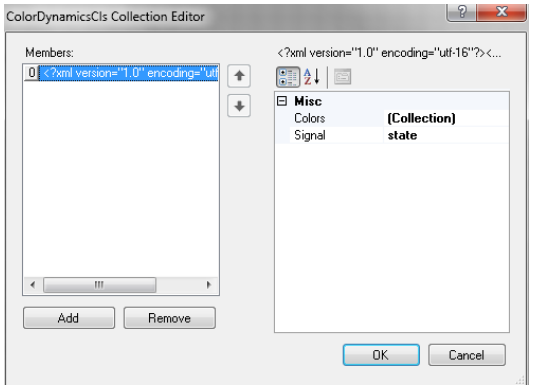


Object GetTag(string pNameofPublicProperty)

With GetTag Function you can read value of a Public property. Return value is object.

Now we can use state and blink in pump symbol dynamic.

Step 6 : click on main circle of pump and define following brush dynamic :



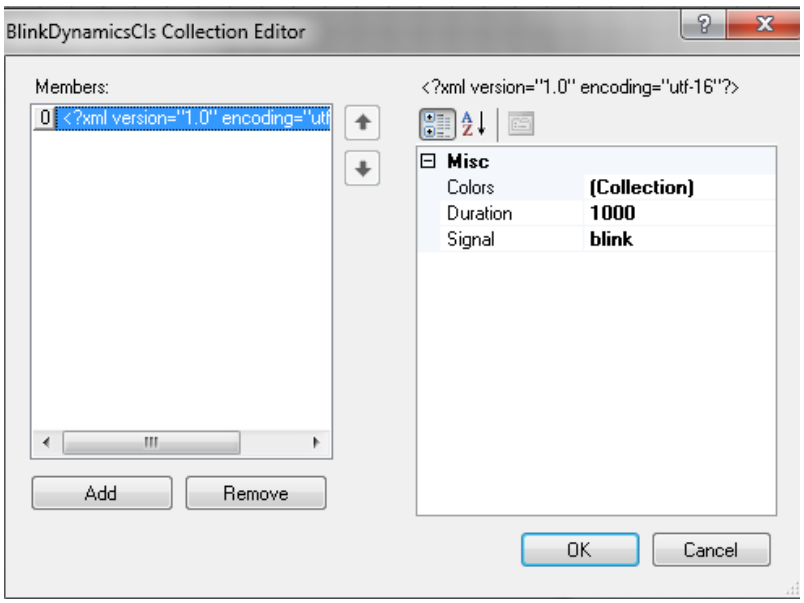
When state = 0 then brush color = gray

When state = 1 then brush color = green

When state = 2 then brush color = red

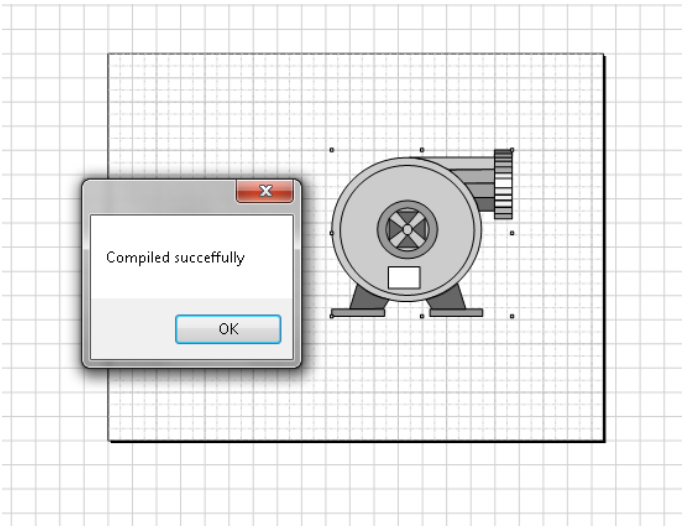
Step7 : for the same element define Blink dynamic as following :

When blink is true, start to change color of circle with this sequence Gray → red → Yellow → Gray



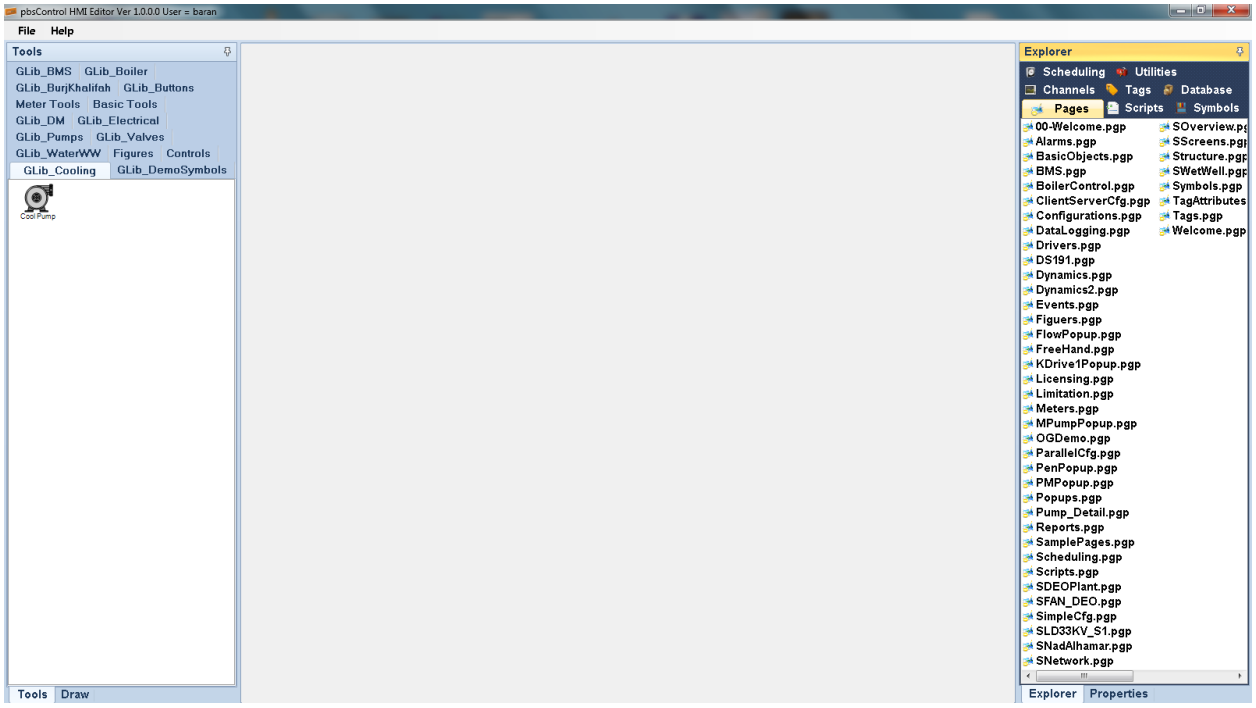
Step 8:Select all pump elements and make them group . In Group properties , change pbsName to “CoolPump”

Step 9: Save library and from File menu compile library.

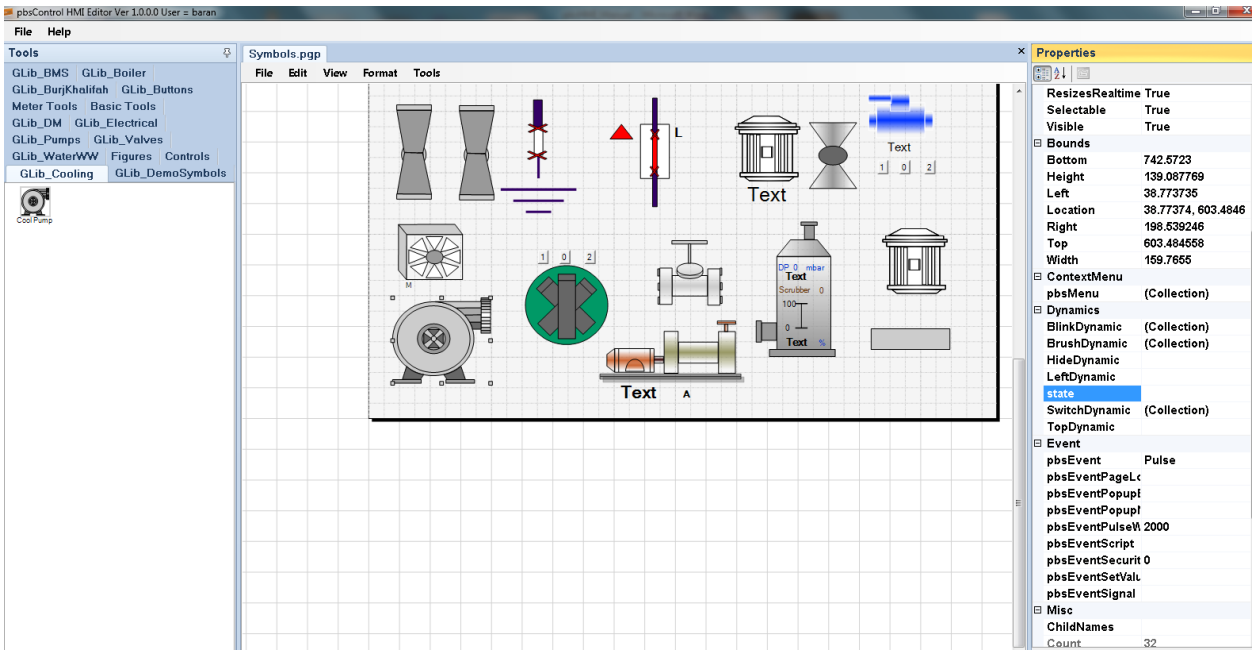


Step 10: you can see Glib_Cooling.DLL in c:\pbsHMI Directory.

Step 11 : Close pbsHMI and run it again . Now Glib_Cooling is loaded to pbsHMI Editor .



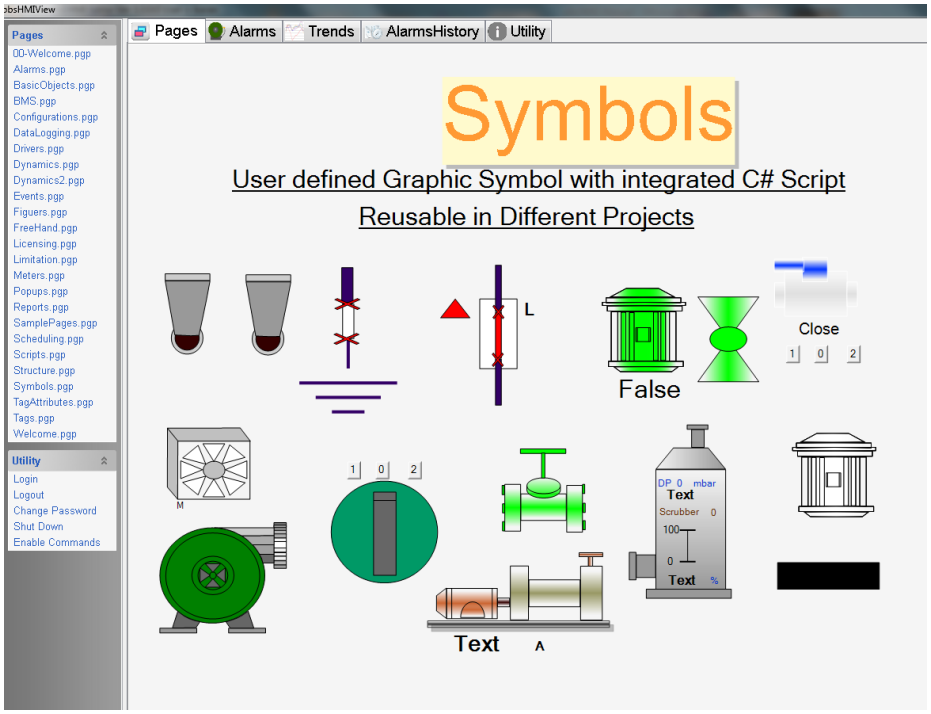
Step 12 : open a page and f=drag and drop CoolPump symbol to page .



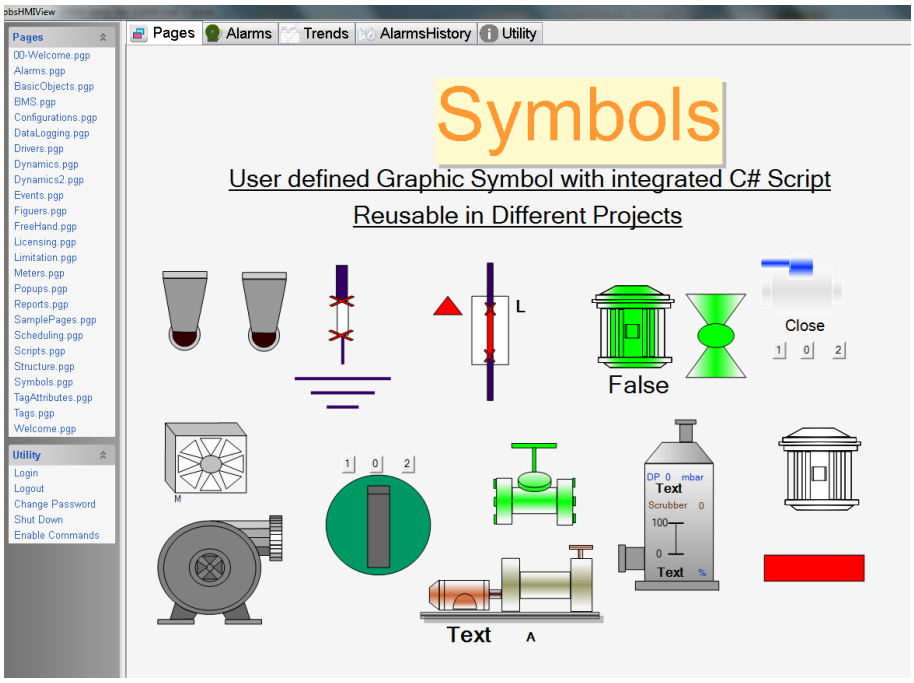
Step 13 : open properties of CoolPump Symbol , you can see “state” is added to Dynamics group and you need to link signal to it .

Step14: Link @I#1 to state signal . @I#1 is constant value , with type int and value 1 .

Step15 :run pbsHMI View and see status of CoolPump .



Step 16 : change state signal value to @I#0 and check status of CoolPump .



Step 17: change value of state to @I#2 and check status of CoolPump . It will start to blink.

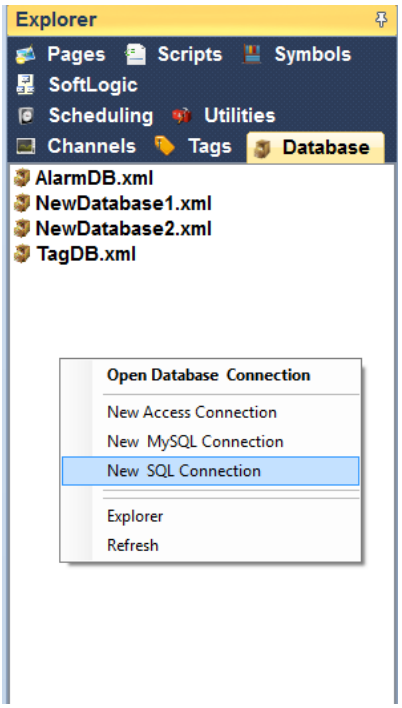
6 – SQL Server and data logging

pbsHMI has built in facilities for logging data in Microsoft MS Access , SQL Server and My SQL databases . In this section we will see how you can configure and archive data in databases.

6 – 1 SQL Server Configuration and data logging

pbsHMI is using MS SQL Server for data , event and alarm archiving .

Defining MS SQL Server connection in pbsHMI : Click on Database tab and right click on Database area .



Then select “New SQL Connection”.

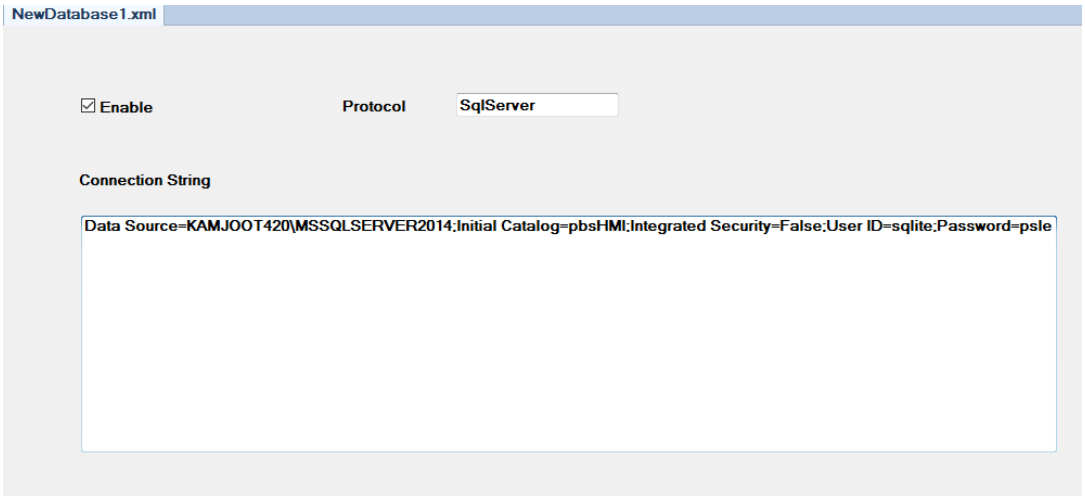
pbHMI will make a Database configuration file in Database directory .

Name of Database connection is NewDatabase{x}.xml.

You need to Open database directory by click on explorer and change NewDatabase{x}.xml to any proper name based on your project.

You will use Database configuration file name in Device Tags, Global Tags, Alarms and Event configuration for setting data logging.

Double click on SQL Server configuration file you can see following page:



You should set SQL Connection String for connecting to MS SQL Server.

Typical SQL Connection strings:

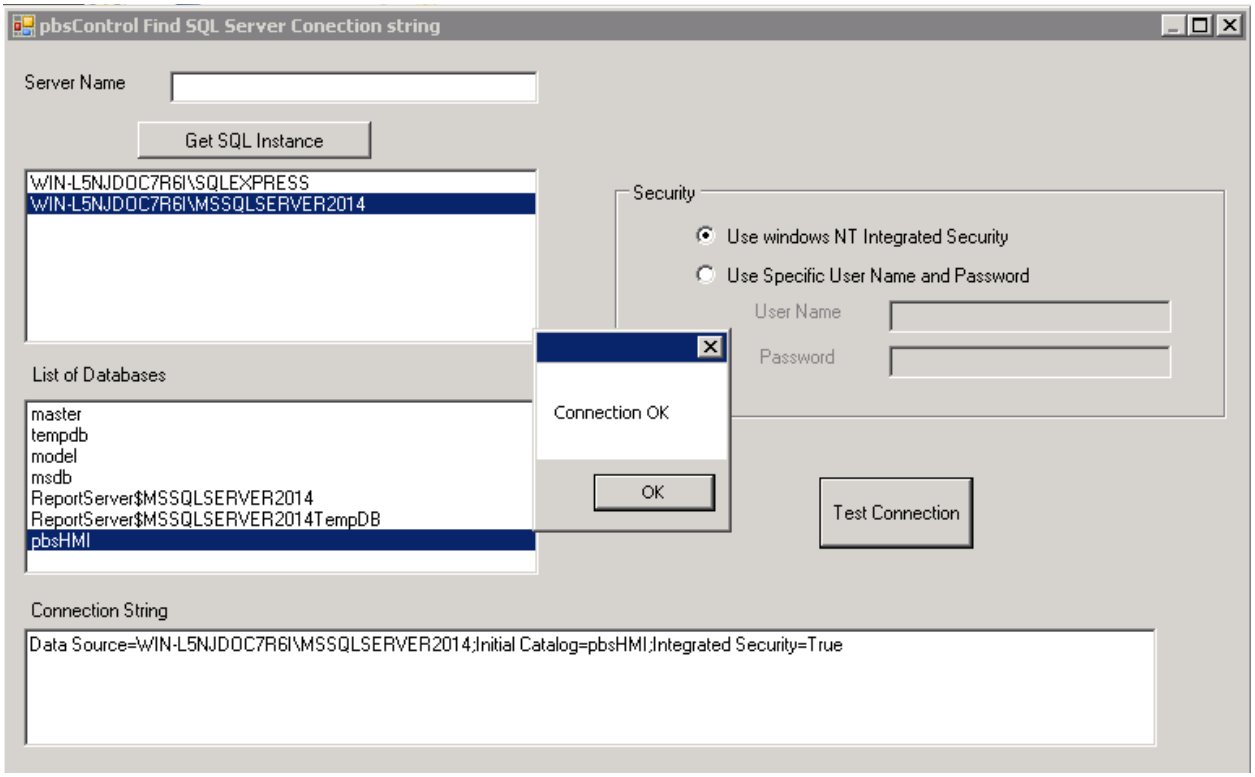
When using windows authentication:

Data Source=WIN-L5NJDOC7R6I\MSSQLSERVER2014;Initial Catalog=pbsHMI;Integrated Security=True

When using SQL Server user name and password:

Data Source=WIN-L5NJDOC7R6I\MSSQLSERVER2014;Initial Catalog=pbsHMI;Integrated Security=False;User ID=sqlite;Password=xyziiii

There is a simple application (GetSQLConnStr.exe) in utility directory to find your server Connection String. Copy GetSQLConnStr.exe to your SQL Server Computer and run it.



Click on Get SQL Instance , you can see list of installed SQQL Server .Select your SQL Server Instance , you can see name of databases In this instance .Select pbsHMI Database and select security .you can select windows Integrated Security or SQL Server User . At final step , click on Test Connection . Security is correct you can see Connection String with Connection OK Message . Copy Connection string and paste it in pbsHMI SQL Server Database configuration file in connection String part .

In database directory of pbsHMI , you can see six sql script files to make pbsHMI Tables in SQL Server .

Make a new database in SQL Server and name it pbsHMI .

You should make two tables for Alarm Logging by following sql scripts in pbsHMI Database:

SQLServerAlarmIndex.sql

SQLServerAlarmData.sql

For Global and Device Tags archiving you need to use following scripts to make two tables:

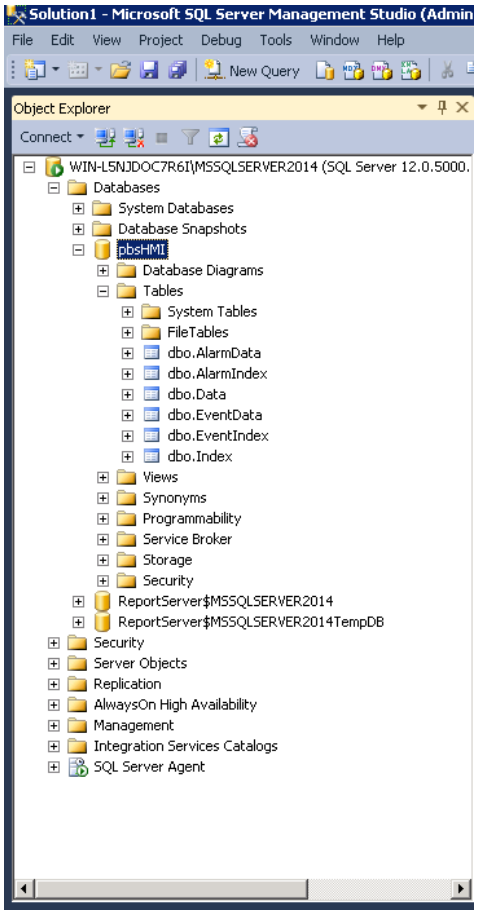
SQLServerIndex.sql

SQLServerData.sql

And you need to use following scripts to make two tables for event logging:

SQLServerEventIndex.sql

SQLServerEventData.sql

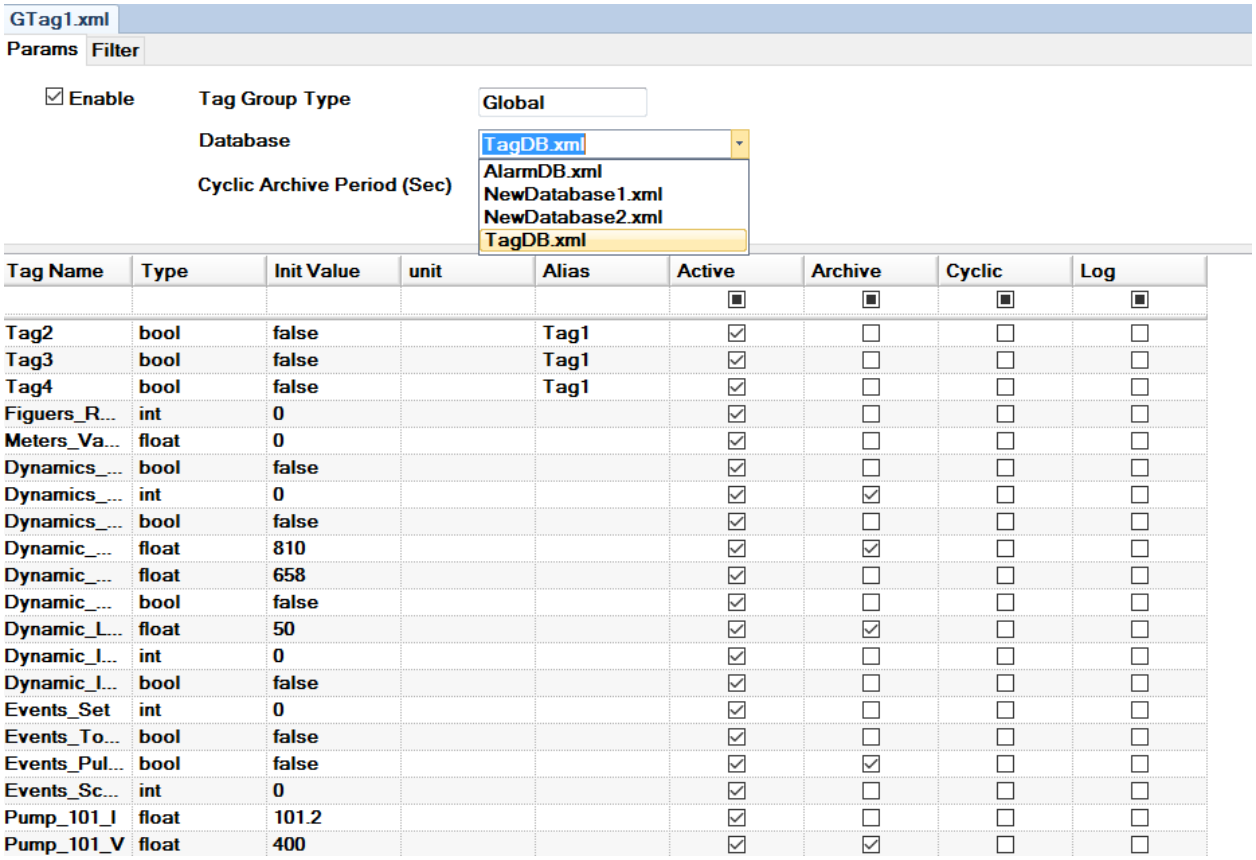


dbo.Index and dbo.Data tables are used for global tags and device tags logging.

dbo.AlarmIndex and dbo.AlarmData tables are used for Alarms data logging

dbo.EventIndex and dbo.EventData Tables are used for logging Events

Using Database configuration file in General and device tags:



Tag Name	Type	Init Value	unit	Alias	Active	Archive	Cyclic	Log
Tag2	bool	false		Tag1	☑	☐	☐	☐
Tag3	bool	false		Tag1	☑	☐	☐	☐
Tag4	bool	false		Tag1	☑	☐	☐	☐
Figuers_R...	int	0			☑	☐	☐	☐
Meters_Va...	float	0			☑	☐	☐	☐
Dynamics_...	bool	false			☑	☐	☐	☐
Dynamics_...	int	0			☑	☑	☐	☐
Dynamics_...	bool	false			☑	☐	☐	☐
Dynamic_...	float	810			☑	☑	☐	☐
Dynamic_...	float	658			☑	☐	☐	☐
Dynamic_...	bool	false			☑	☐	☐	☐
Dynamic_L...	float	50			☑	☑	☐	☐
Dynamic_I...	int	0			☑	☐	☐	☐
Dynamic_I...	bool	false			☑	☐	☐	☐
Events_Set	int	0			☑	☐	☐	☐
Events_To...	bool	false			☑	☐	☐	☐
Events_Pul...	bool	false			☑	☑	☐	☐
Events_Sc...	int	0			☑	☐	☐	☐
Pump_101_I	float	101.2			☑	☐	☐	☐
Pump_101_V	float	400			☑	☑	☐	☐

In global and device tag files you can see database combo box to select proper database connection. you can use one database configuration file for different resources .

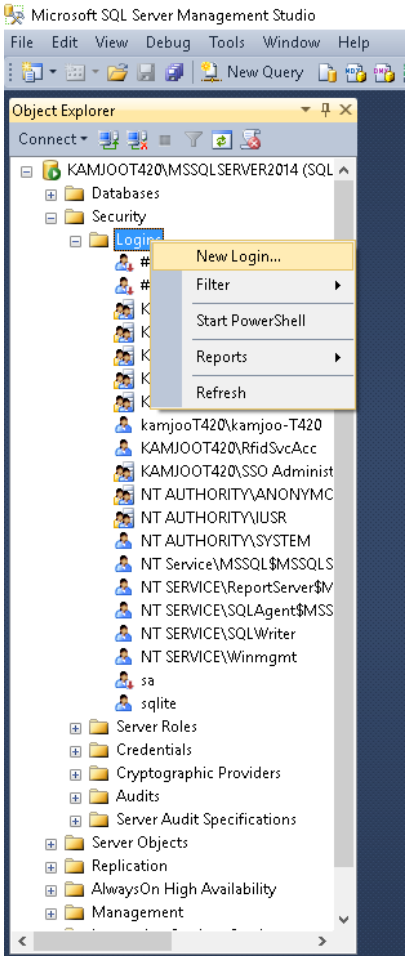
Logging Data by changes: if you select Archive option for a tag , pbsHMI will archive all tag value changes in database .

Logging Data Cyclic : If you select Cyclic option for a tag , pbsHMI will log value of tag every “Cyclic Archive Period “ in database .

For Alarm and Events you couldn’t select specific Alarm or Events and pbsHMI will log all Events and Alarms.

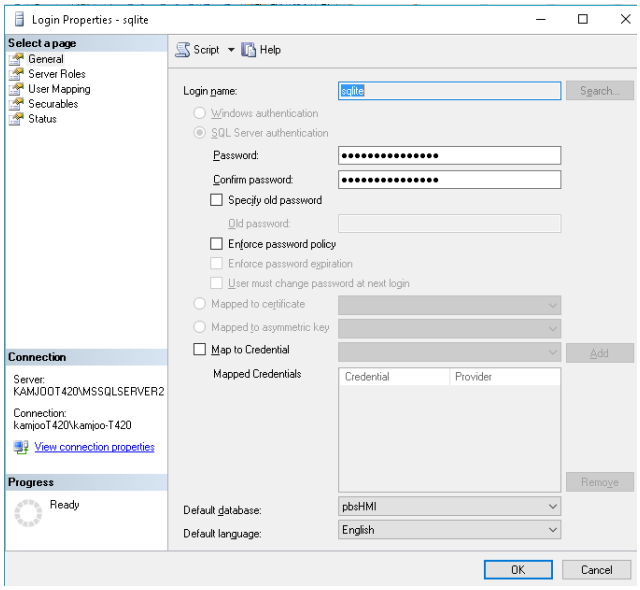
MS SQL Server setting : you need to define pbsHMI Database in SQL Server at control room and define one User with password to use in Client/server Structure inpbsHMI . Please do following steps :

1- Make a new database in SQL Server ,name it pbsHMI. (You did it before)
2- Use SQL Script that is in SQLite Directory of pbsSoftLogic or use Database directory of pbsHMI (You did it before)
3- Run SQLServerData.sql and SQLServerIndex.sql to make data and index tables in pbsHMI Databse (You did it before)
4- In SQl Server Management studio , Use “Security” item and open “Login” segment .
5- Right click in Login and select New Login

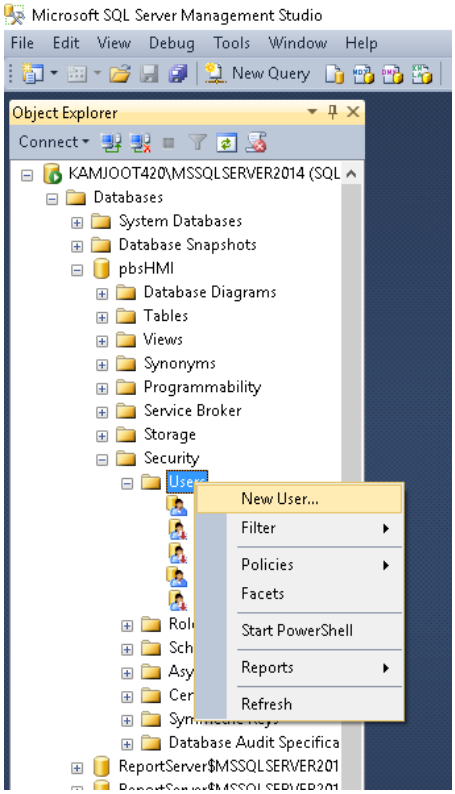


6 – In Login Properties page type user name for example “sqlite” and set password .Remove pass policy , expiration and user should change password in next login . you will use this user and password in SQLite driver options .

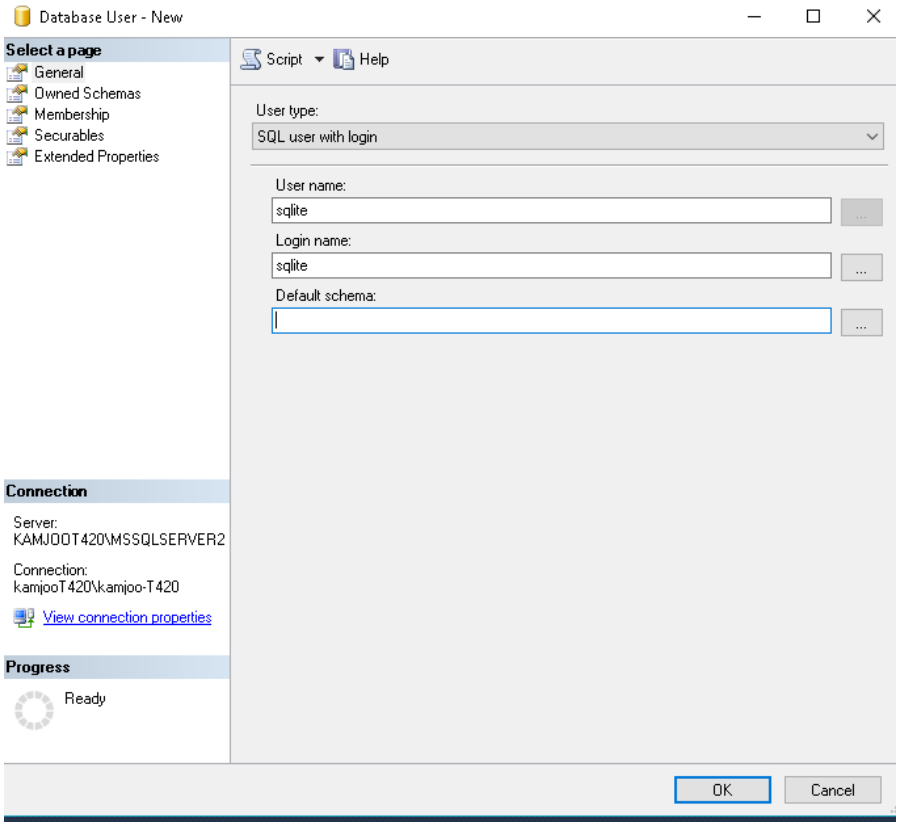
7 – select pbsHMI as default database .



8 – select pbsHMI Database , Select Security and open Users and right click on user to make new user.

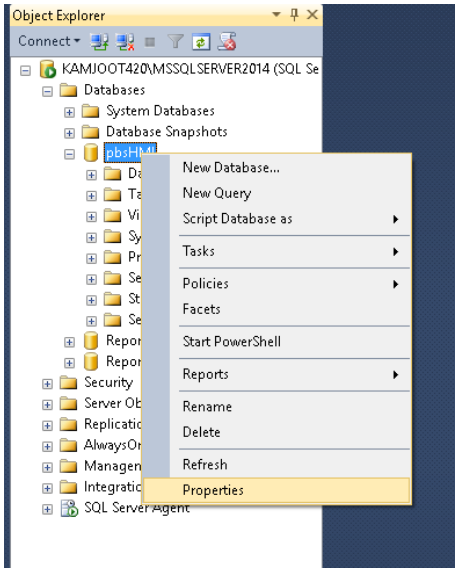


9 –use same user that you make for SQL server “sqlite” and redefine it here.

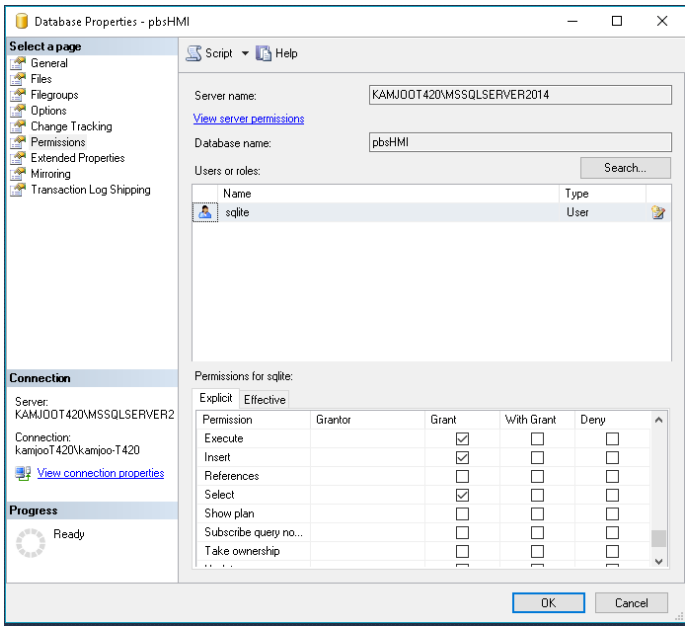


10 – Click on ok to define user .

11 – Select properties of pbsHMI Database

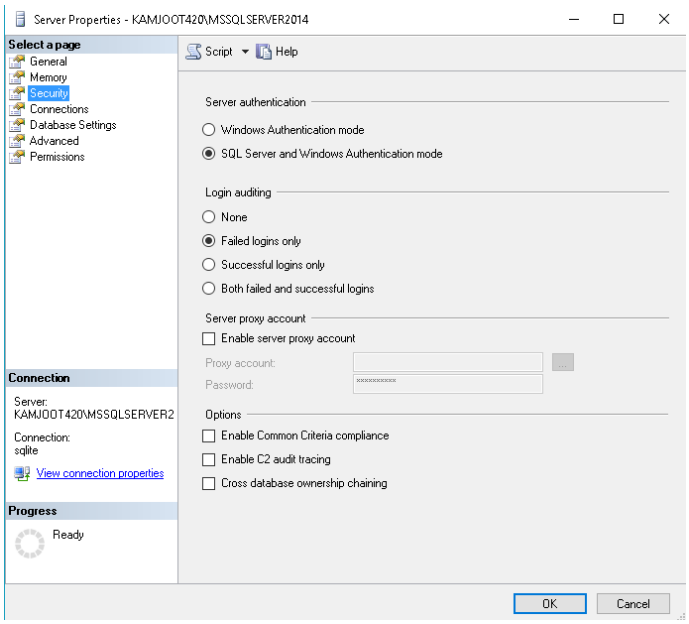


12 – select permissions Page and Grant Connect , Execute , Insert , Update , Delete and select functionality to sqlite user . In this stage SQL Server is ready for proper operation Client/server Structure.



13 – Check pbsHMI Database collation is not Arabic , Persian , … and only Latin like SQL_Latin1_General_CP1_CI_AS will communicate with RTU

14 – Check SQL Server Authentication is SQL Server and Windows

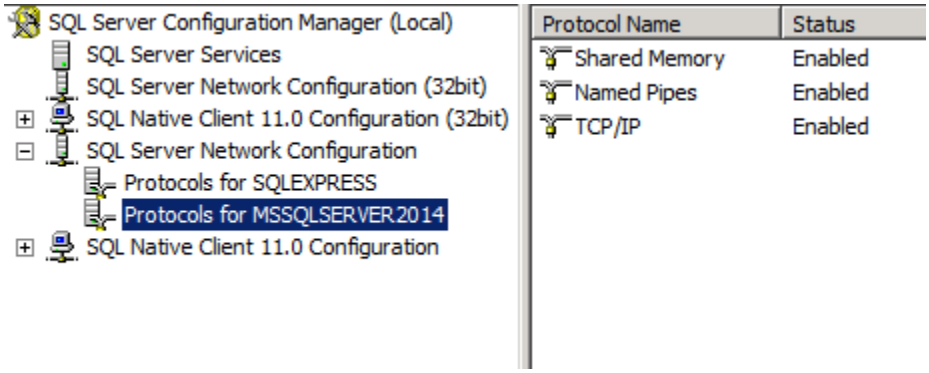


15 – Check in Windows Firewall. TCP Port 1433 should be allowed in both inbound and outbound rules.

16 – Check in Windows Firewall. SQLBrowser.exe utility should be in Allowed programs List .you can find SQLBrowser.exe Path from C:\Program Files (x86)\Microsoft SQL Server\90\Shared\sqlbrowser.exe

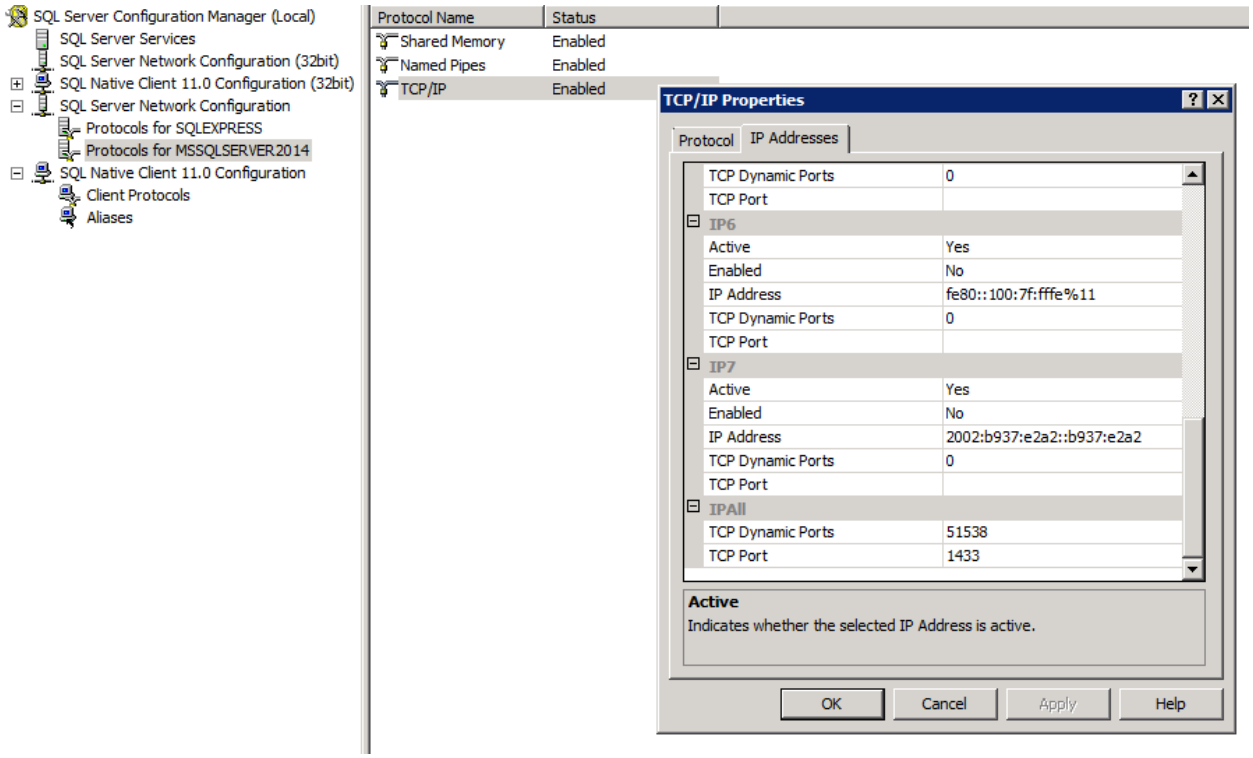
Check SQL Server and SQL Server Browser Services are started properly.

17 – Open SQL Server Configuration Utility and open SQLServer Network Configuration.



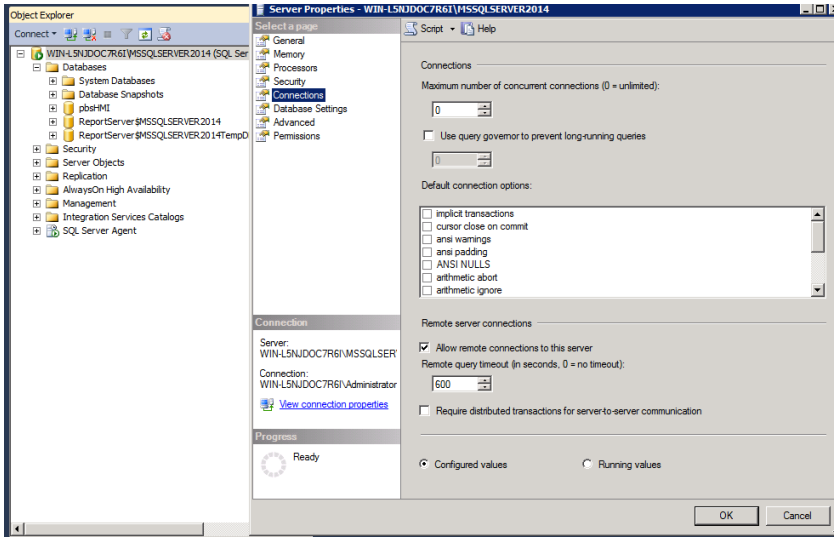
Check Named Pipes and TCP/IP protocols are enabled. You can Make Them Enable by Right Click on each Item and select Enable Option.

Double click on TCP/IP Protocol and select IP Tab.



Scroll Down to IPALL and write TCP Port 1433.

18 – Open property page of main SQL server instance and select Connections.



Check Allow Remote Connections to this server is checked.

19 – Restart SQL Server Service.

If you are using SQL Server Express database, there is a 10 GB limitation for maximum database fie size.

When you reach to this limit, pbshMI cannot archive data.

For fixing this problem, you need to define a new pbsHMI2 database and generate all tables for this database and change in “pbsHMI” Database name in pbsHMI configuration files to “pbsHMI2”.

7 – C# Scripting

pbsHMI supports C# Scripting for data manipulations . You can communicate with Database systems, Microsoft office, … by C# Scripting .

All C# codes that is developing by user is compiled in one Dll . pbsHMIUserScript.dll

pbsHMIUserScript.dll has its own thread and it is communicating with pbsHMI Runtime kernel Ram Based RDBMS.

There are two functions for reading and writing to all pbsHMI Tags in C# Scripting

SetTag(string pName , object pValue)

SetTag writes to pbsHMI Tag .

pName = Full name of a pbsHMI Tag

pValue = Value of Tag in object format

Object GetTag(string pName)

Read pbsHMI Tag

pName = Full name of a pbsHMI Tag

All pbsHMI script files are located in c:\pbsHMI\scripts directory.

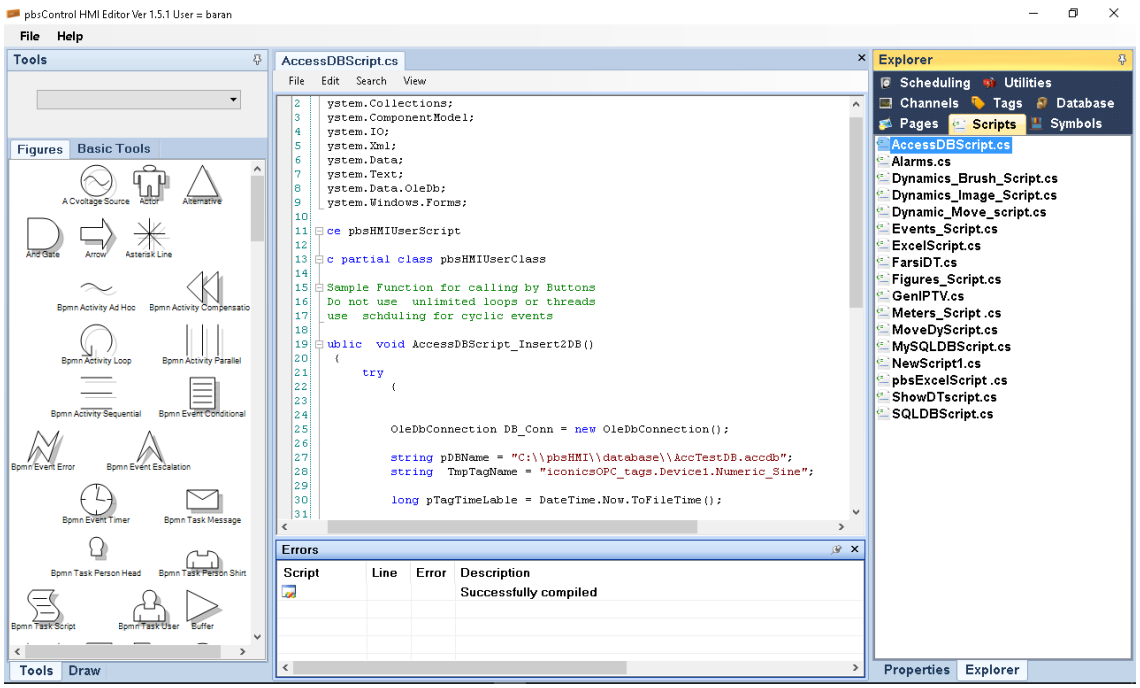
There is a system script file in scripts directory. UserScripts.cs Please do not delete or do any modification in this file.

For developing C# Scripts you need to know C# programming language. There are many samples in c:\pbsHMI\Scripts directory for your quick reference.

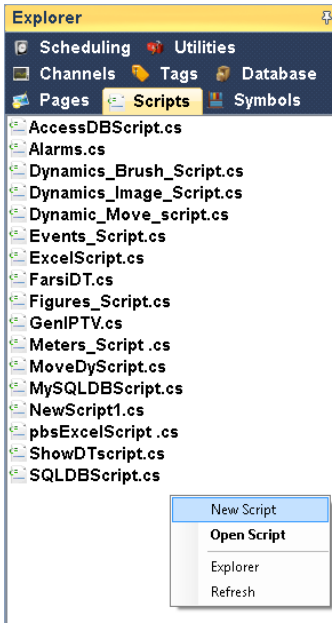
You can define public tag in scripts. pbsHMI will keep value of signal In each execution of script .

pbsHMI Scripts can be run by pbsHMI Scheduling or by events .

You can see all scripts in Scripts tab of pbsHMI Explorer .



For defining new script, right click on Scripts tab area and select new script.



New script has following format:

```
using System;
using System.Collections;
using System.ComponentModel;
using System.IO;
using System.Xml;
using System.Data;
using System.Windows.Forms;

namespace pbsHMIUserScript
{
  public partial class pbsHMIUserClass
  {

     //Sample Function for calling by Buttons
     //Do not use  unlimited loops or threads
     //use  schduling for cyclic events
      public  void NewScript1()
      {

      }
  }
}
```

Please do not change namespace and class name. You can only change name of Function.

Suppose you want to increase value of a pbsHMI Tag at each runtime of script .

```
using System;
using System.Collections;
using System.ComponentModel;
using System.IO;
using System.Xml;
using System.Data;

namespace pbsHMIUserScript
{
  public partial class pbsHMIUserClass
  {
     //Sample Function for calling by Buttons
     //Do not use  unlimited loops or threads
     //use  schduling for cyclic events

      public  void Dynamic_Brush_Change()
      {
            int TmpBrushValue = int.Parse(GetTag("GTag1.Dynamics_Brush").ToString());
            TmpBrushValue++;
            if(TmpBrushValue > 3)
            {
                TmpBrushValue = 0;
            }
            SetTag("GTag1.Dynamics_Brush", TmpBrushValue);
      }
  }
}
```

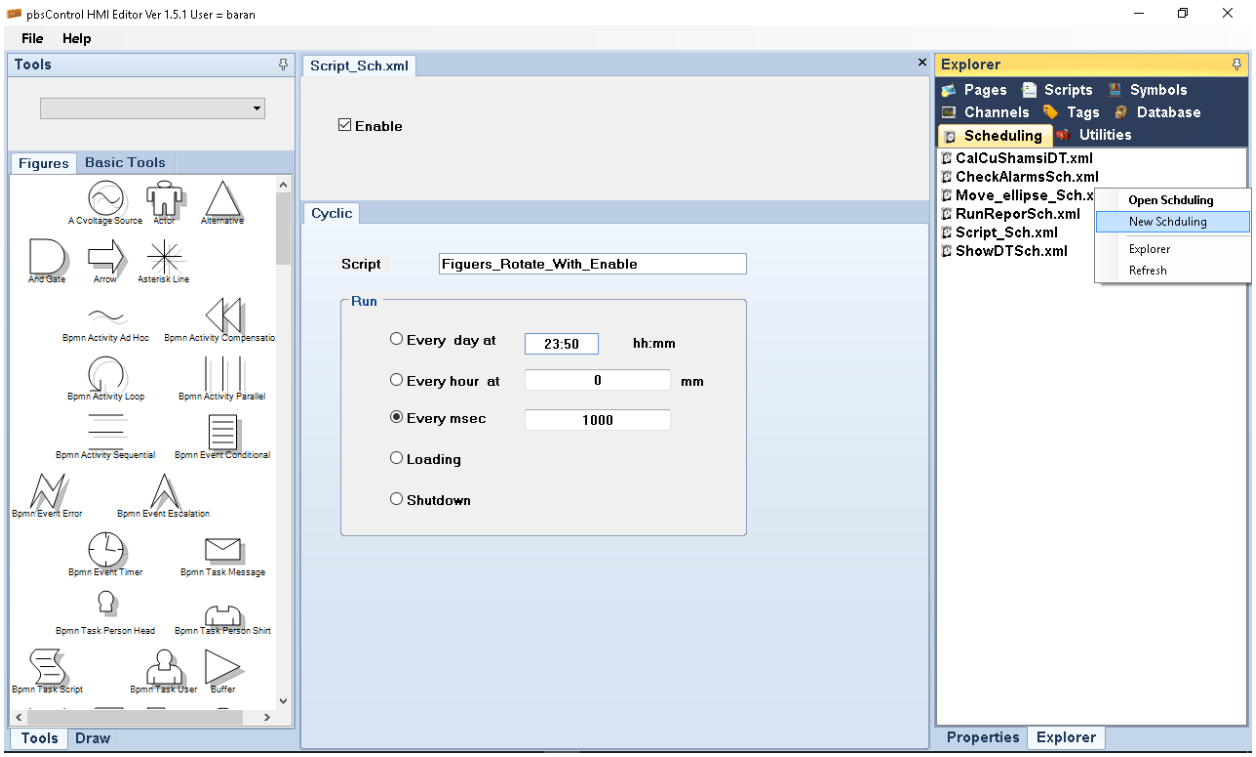
pbsHMI will load all functions dynamically at startup . pbsHMI will call functions by scheduling or by user events .

You couldn't change any graphic object properties by scripting directly. For this scenario you need to do following steps:

1- define one global tag
2- Link global tag to graphic object dynamics
3- Change value of global tag in script.

8 – Scheduling

pbsHMI supports different types of Scheduling for doing automatic tasks .You can define new schedule in scheduling tab of pbsHMI explorer .



Right click on scheduling area and define a new schedule.

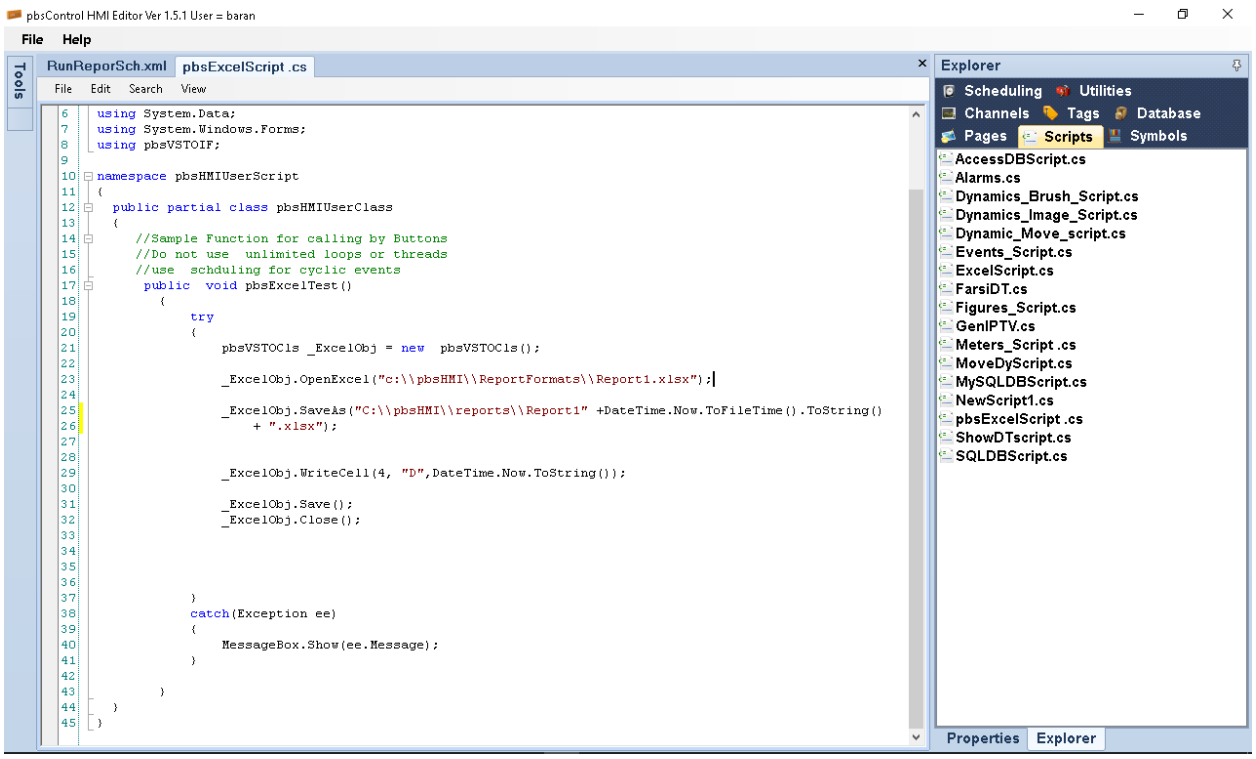
Script = Name of script function for execution in schedule

Following scheduling is defined in pbsHMI :

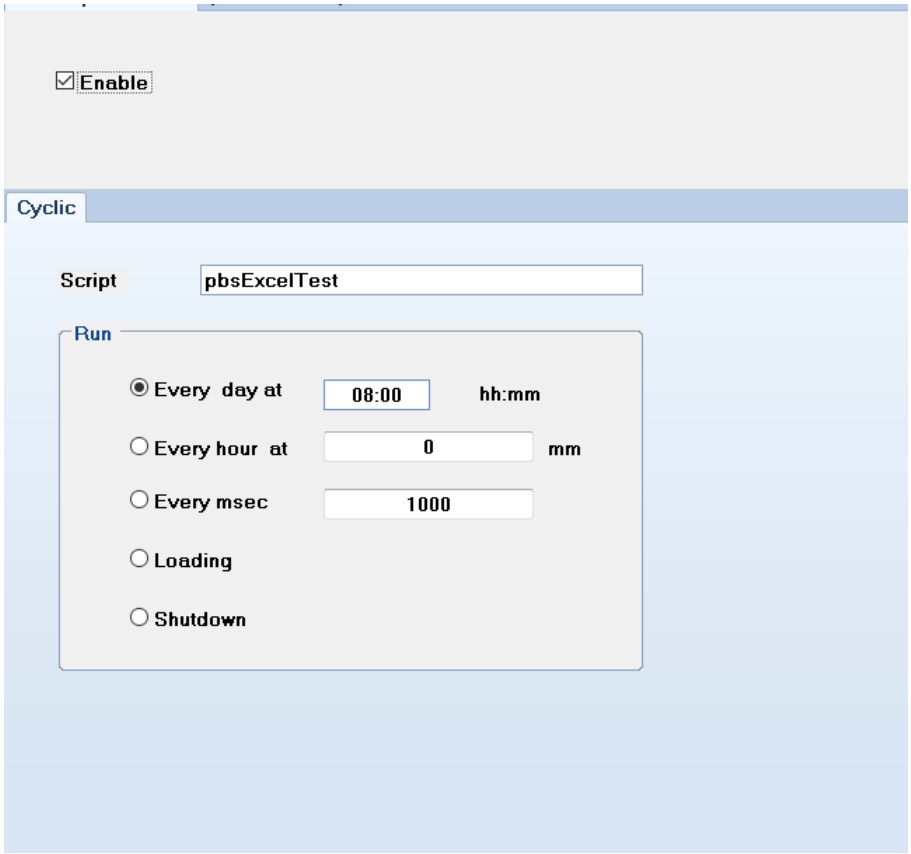
- Every Day at hh:mm = It will run Script for one time every day at hh:mm
- Every Hour at mm = It will run script every hour at mm min.if mm is 10 as an example , then it will run script every hour at 10 min .
- Every msec : will run script every msec . you can use this schedule for executing script in cyclic mode .
- Loading : run script at load time
- Shutdown : Load script at pbsHMI shutdown

Schedules are global in pbsHMI. Schedules are not running in clients applications. Schedules are running only in pbsHMI Server application. When value of a pbsHMI tag is changed in script , then all changes will publish to clients automatically and no need to run schedule in clients .

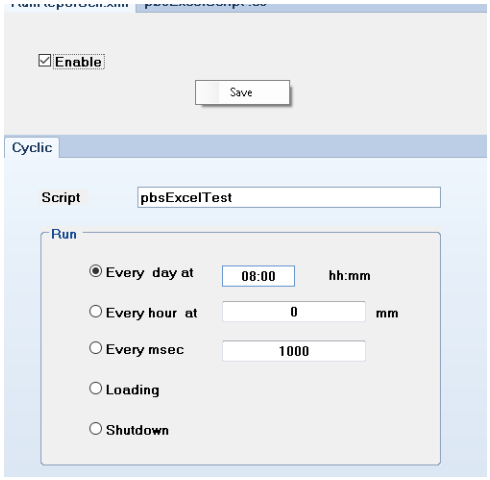
Suppose you want to run an excel report every day at 8 am . At first stage you need to define a new script to provide execl file. Look at following script as an example:



Define a new schedule and run pbsExcelTest function every morning at 08:00



Save schedule by right click on schedule header form and select save.



pbsHMI Runtime will execute pbsExcelTest function every day at 08:00 am .

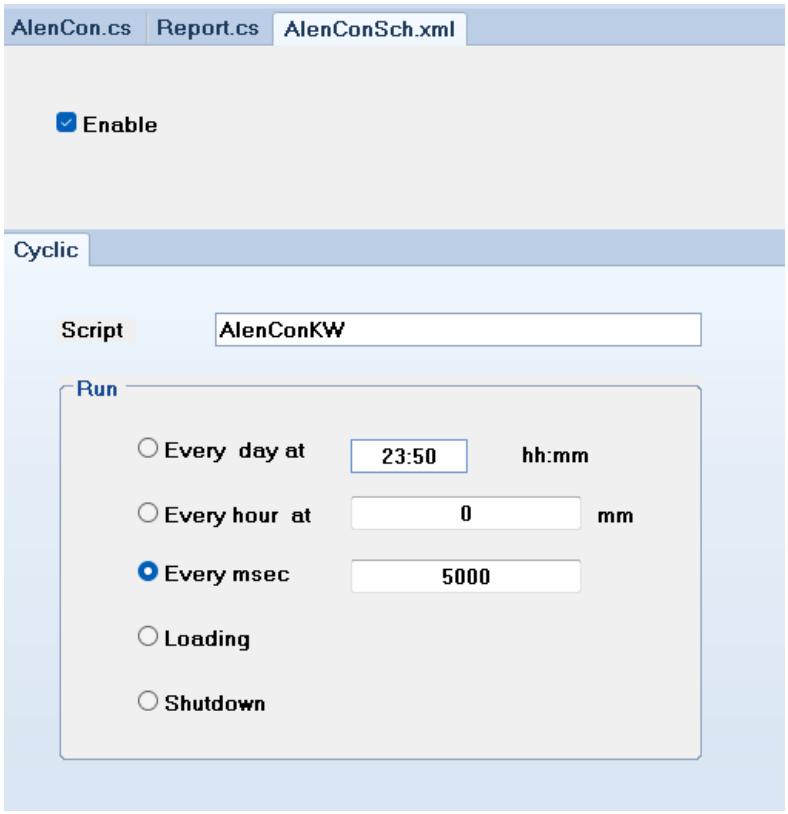
in the following code we read current and voltage from different Solar panels and calculate Active Power and set value on a global tag .

AlenCon.cs Report.cs

File Edit Search View

```
using System.IO;
using System.Xml;
using System.Data;
using System.Windows.Forms;
namespace pbsHMIUserScript
{
  public partial class pbsHMIUserClass
  {
      public  void AlenConKW()
      {
            double TmpAlenCon11_PV = double.Parse(GetTag("MQTTTags.Devcie1.11_Primary_Voltage").ToString());
            double TmpAlenCon12_PV = double.Parse(GetTag("MQTTTags.Devcie1.12_Primary_Voltage").ToString());
            double TmpAlenCon13_PV = double.Parse(GetTag("MQTTTags.Devcie1.13_Primary_Voltage").ToString());
            double TmpAlenCon11_PI = double.Parse(GetTag("MQTTTags.Devcie1.11_Primary_Current").ToString());
            double TmpAlenCon12_PI = double.Parse(GetTag("MQTTTags.Devcie1.12_Primary_Current").ToString());
            double TmpAlenCon13_PI = double.Parse(GetTag("MQTTTags.Devcie1.13_Primary_Current").ToString());
            double AlenCon11_PKW = (TmpAlenCon11_PV * TmpAlenCon11_PI) / 1000;
            double AlenCon12_PKW = (TmpAlenCon12_PV * TmpAlenCon12_PI) / 1000;
            double AlenCon13_PKW = (TmpAlenCon13_PV * TmpAlenCon13_PI) / 1000;
            SetTag("gTags.AlenCon11_Kw", AlenCon11_PKW);
            SetTag("gTags.AlenCon12_Kw", AlenCon12_PKW);
            SetTag("gTags.AlenCon13_Kw", AlenCon13_PKW);
            double TmpAlenCon11_SV = double.Parse(GetTag("MQTTTags.Devcie1.11_Secondary_Voltage").ToString());
            double TmpAlenCon12_SV = double.Parse(GetTag("MQTTTags.Devcie1.12_Secondary_Voltage").ToString());
            double TmpAlenCon13_SV = double.Parse(GetTag("MQTTTags.Devcie1.13_Secondary_Voltage").ToString());
            double TmpAlenCon11_SI = double.Parse(GetTag("MQTTTags.Devcie1.11_Secondary_Current").ToString());
            double TmpAlenCon12_SI = double.Parse(GetTag("MQTTTags.Devcie1.12_Secondary_Current").ToString());
            double TmpAlenCon13_SI = double.Parse(GetTag("MQTTTags.Devcie1.13_Secondary_Current").ToString());
            double AlenCon11_SKW = (TmpAlenCon11_SV * TmpAlenCon11_SI) / 1000;
            double AlenCon12_SKW = (TmpAlenCon12_SV * TmpAlenCon12_SI) / 1000;
            double AlenCon13_SKW = (TmpAlenCon13_SV * TmpAlenCon13_SI) / 1000;
            SetTag("gTags.AlenCon11_KwS", AlenCon11_SKW);
            SetTag("gTags.AlenCon12_KwS", AlenCon12_SKW);
            SetTag("gTags.AlenCon13_KwS", AlenCon13_SKW);
      }
  }
```

We want to run this script every 5 sec, so we need to define following scheduling:



You can define multiple scheduling for a project and all will execute independent from each other.