DMC

Smart People. Expert Solutions.

SIEMENS OPEN LIBRARY

3 – Example Object Configuration

JUNE 11, 2019

Contents

1. Purpose

The purpose of this document is to walk through the steps required to setup an object in the Siemens Open Library. Each object has a different interface, as defined in the detailed block description for that object, however, all of the objects are setup and configured using the methods outlined in this document.

2. Intended Use

This document is intended to be used by anyone utilizing the Open Library for PLC and HMI Development. This document should be used after reviewing the following documents:

1. 1- Siemens Open Library – Library Overview and Architecture
2. 2- Siemens Open Library – Initial Setup

3. Revision History

Version	Date	Author	Comments
1.0	2016-05-23	DMC	Initial Release
1.1	2016-06-20	DMC	No Changes
1.2	2016-08-23	DMC	No Changes
1.3	2016-10-11	DMC	No Changes
2.0	2017-11-03	DMC	Added reference to SiVArc use for HMI Development.
3.0	2018-12-05	DMC	No Changes
4.0	2019-02-15	DMC	No Changes

4. Open Library License

5. Example Setup

This tutorial walks through the process of using one of the blocks from the Open Library. The example object is the Solenoid Valve, but the same method is utilized for any other object block, with special consideration and separate document for use of the PID Compact block. For detailed information for each individual object, see the documentation pertaining to that object in '4- Siemens Open Library – Detailed Block Overview.'

5.1. Initial Setup

Before starting this document please make sure you have set up your project following the steps in '2- Siemens Open Library - Initial Setup.' Each library object block is dependent on global constants and clock memory bits and will not compile without correctly completing the initial setup. The following steps need to be performed:

1. Enable system and clock memory bytes on the CPU.
2. Retrieve the Open Library.
3. Pull the Open Library PLC tags into the project.
4. Setup Mode Control, or understand how Open Library Modes function. Additionally information can be found in '1- Siemens Open Library - Library Overview and Architecture'

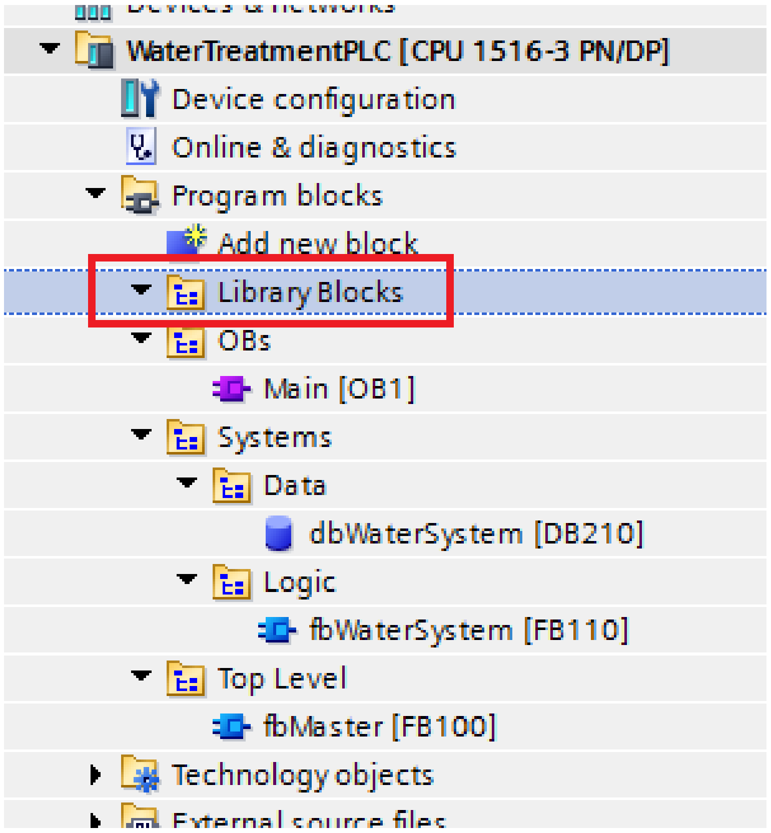
5.2. PLC

This section covers the steps required to configure the PLC project. It utilizes the best practices standard with this library, but different methods can be used to store the Function Block Instance memory and the User Defined Types.

5.2.1. Adding PLC Library Objects to Project

This section walks through the required steps to add an individual library object to a project. The entire PLC library can be added to the project by dragging and dropping the entire folder into the project Program Blocks. The following steps will demonstrate adding fbValve_Solenoid to the project.

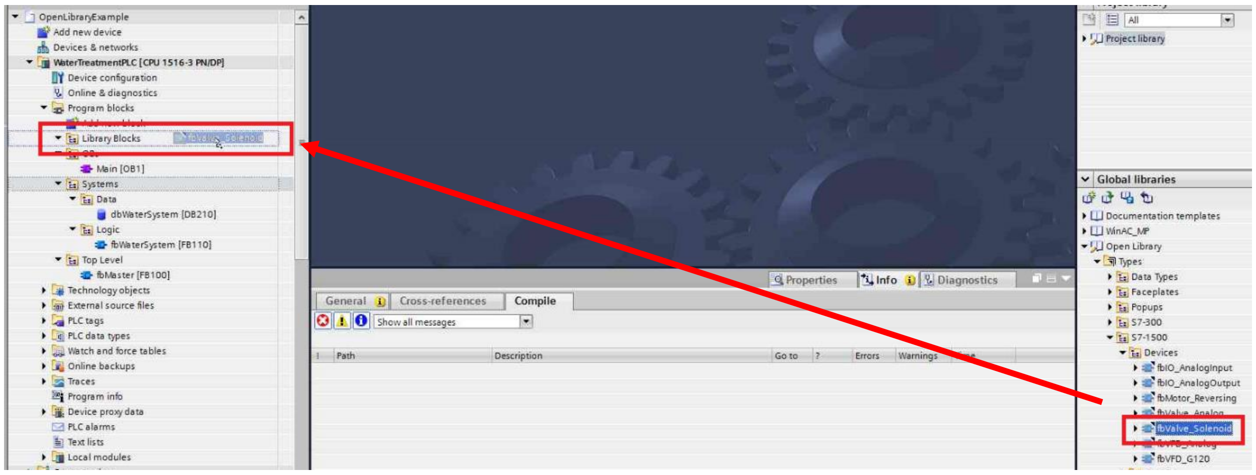
1. In the controller's Program blocks, we have created example groups and blocks used in this tutorial, as seen below. A similar structure is suggested for your project. Notice the Library Blocks group is empty now because this is where the library blocks will be placed.



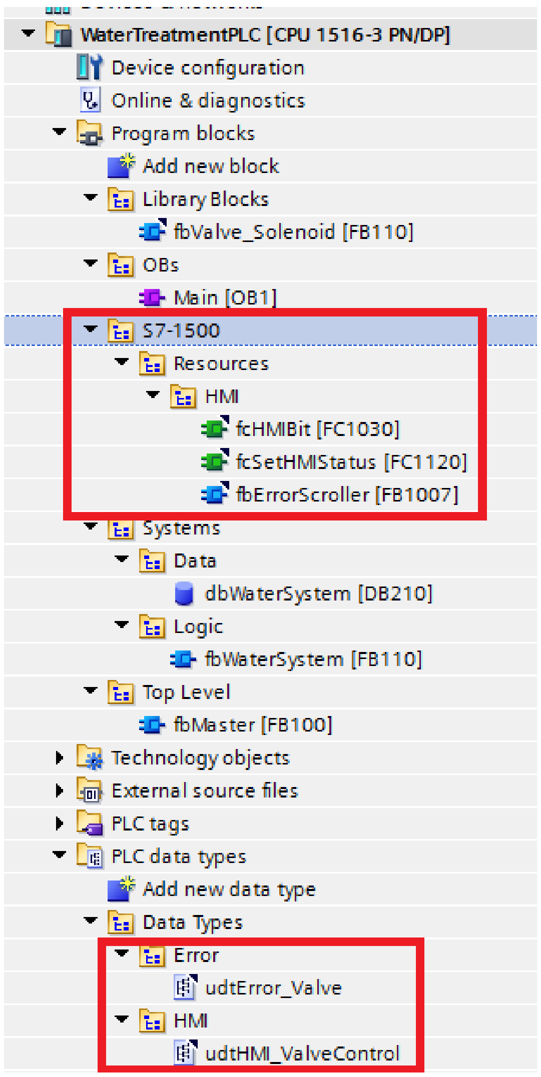
WaterTreatmentPLC [CPU 1516-3 PN/DP]
Device configuration
Online & diagnostics
Program blocks
Add new block
Library Blocks
OBs
Main [OB1]
Systems
Data
dbWaterSystem [DB210]
Logic
fbWaterSystem [FB110]
Top Level
fbMaster [FB100]
Technology objects

2. Copy the desired Function Block from the Open Library Types group of the library into the Library Blocks group created in our project. This is done by dragging and dropping from

the library into the Program Blocks of your PLC. In this case we will demonstrate fbValve_Solenoid. This will need to be done for each PLC in the project that uses this block.

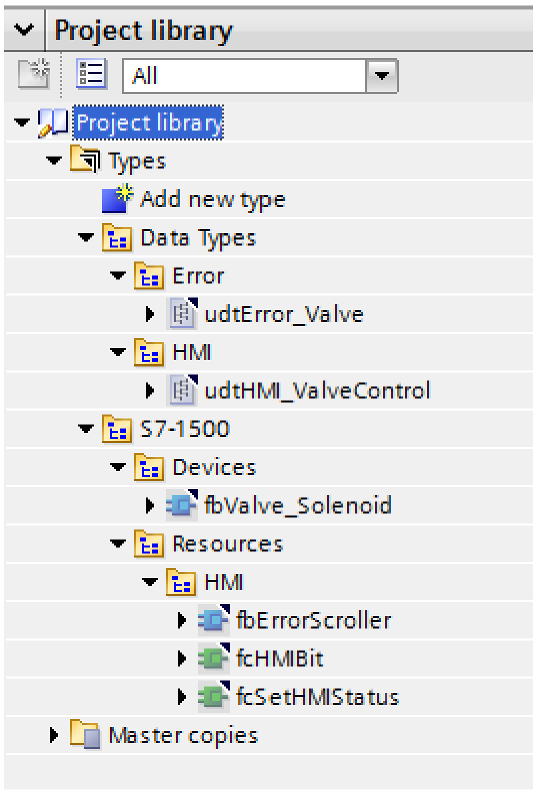


3. Notice that several other function calls and function blocks were added automatically into our project. This happens because fbValve_Solenoid is dependent on fcHMIBit, fcSetHMIStatus, fbErrorScroller, udtError_Valve, and udtHMI_ValveControl. When pulling in a library block, it will automatically pull in all required block dependencies and user defined types. The auxiliary blocks will also be automatically added to the project based on the folder structure of the Open Library.



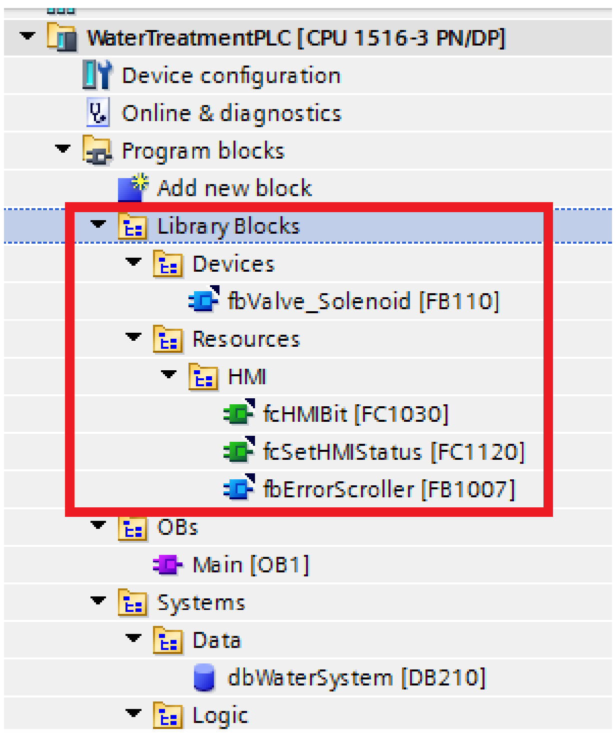
- WaterTreatmentPLC [CPU 1516-3 PN/DP]
  - Device configuration
  - Online & diagnostics
  - Program blocks
    - Add new block
    - Library Blocks
      - fbValve_Solenoid [FB110]
    - OBs
      - Main [OB1]
    - S7-1500
      - Resources
        - HMI
          - fcHMIBit [FC1030]
          - fcSetHMIStatus [FC1120]
          - fbErrorScroller [FB1007]
    - Systems
      - Data
        - dbWaterSystem [DB210]
      - Logic
        - fbWaterSystem [FB110]
    - Top Level
      - fbMaster [FB100]
  - Technology objects
  - External source files
  - PLC tags
  - PLC data types
    - Add new data type
    - Data Types
      - Error
        - udtError_Valve
      - HMI
        - udtHMI_ValveControl

4. The library blocks are added to the Project library. Any library blocks that need to be changed for a particular project can be modified here. Additionally, new versions can be created here, depending on the required application.



```
Project library
All
Project library
  Types
    Add new type
    Data Types
      Error
        udtError_Valve
      HMI
        udtHMI_ValveControl
    S7-1500
      Devices
        fbValve_Solenoid
      Resources
        HMI
          fbErrorScroller
          fcHMIBit
          fcSetHMIStatus
  Master copies
```

5. Reorganize the Program blocks after the library blocks have been automatically added so that navigation is easier. This is not a required step and should be organized based on personal preference or company standards. The organization of the blocks will not affect performance.

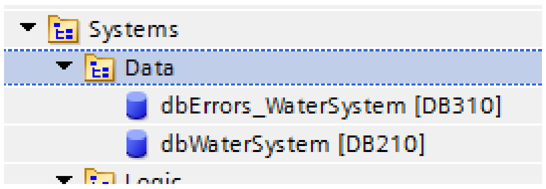


```
WaterTreatmentPLC [CPU 1516-3 PN/DP]
  Device configuration
  Online & diagnostics
  Program blocks
    Add new block
    Library Blocks
      Devices
        fbValve_Solenoid [FB110]
      Resources
        HMI
          fcHMIBit [FC1030]
          fcSetHMIStatus [FC1120]
          fbErrorScroller [FB1007]
    OBs
      Main [OB1]
    Systems
      Data
        dbWaterSystem [DB210]
      Logic
```

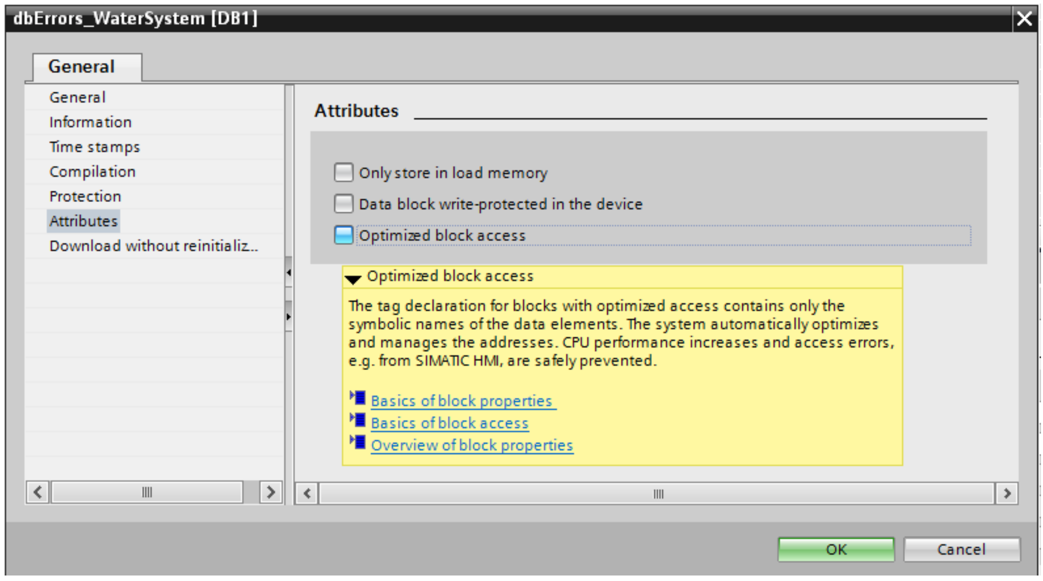
5.2.2. Create Required Data Blocks and User Defined Types

The following section contains the information pertaining to the addition of the required data blocks to contain the User Defined Types in the Siemens Open Library. Blocks can be named or organized in whatever way is best for the individual project, but this section will outline the creation of an HMI Data Block to contain the User Defined Types that map to the Faceplates and an Error Data block to contain the User Defined Types that contain alarms.

1. Create global Data Blocks for housing of the User Defined Types for the HMI and the Errors. In this example, the HMI data will be contained in dbWaterSystem and the error data will be contained in dbErrors_WaterSystem.



2. The Error Data block needs to be non-optimized to utilize the alarm generator application, or executable to automatically generate alarms. To setup non-optimized access, right click on the data block and select 'Properties.' Under the 'Attribute' tab, verify the 'Optimized block access' is not checked. For additional details see '5- Siemens Open Library - Siemens HMI Alarm Generation.'



dbErrors_WaterSystem [DB1]

General

General
Information
Time stamps
Compilation
Protection
Attributes
Download without reinitializ...

Attributes

☐ Only store in load memory
☐ Data block write-protected in the device
☐ Optimized block access

Optimized block access

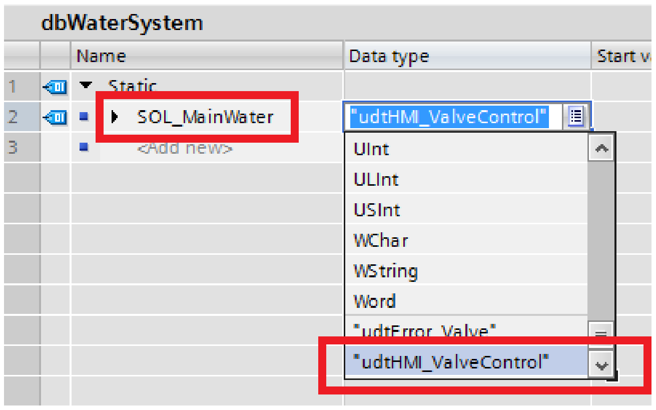
The tag declaration for blocks with optimized access contains only the symbolic names of the data elements. The system automatically optimizes and manages the addresses. CPU performance increases and access errors, e.g. from SIMATIC HMI, are safely prevented.

Basics of block properties
Basics of block access
Overview of block properties

3. Repeat step 2 for the HMI data block if using an S7-300 or S7-400.

4. In the Data Block that will contain the HMI interface, dbWaterSystem in this example, add the HMI control structure for the device by naming the structure and assigning it the

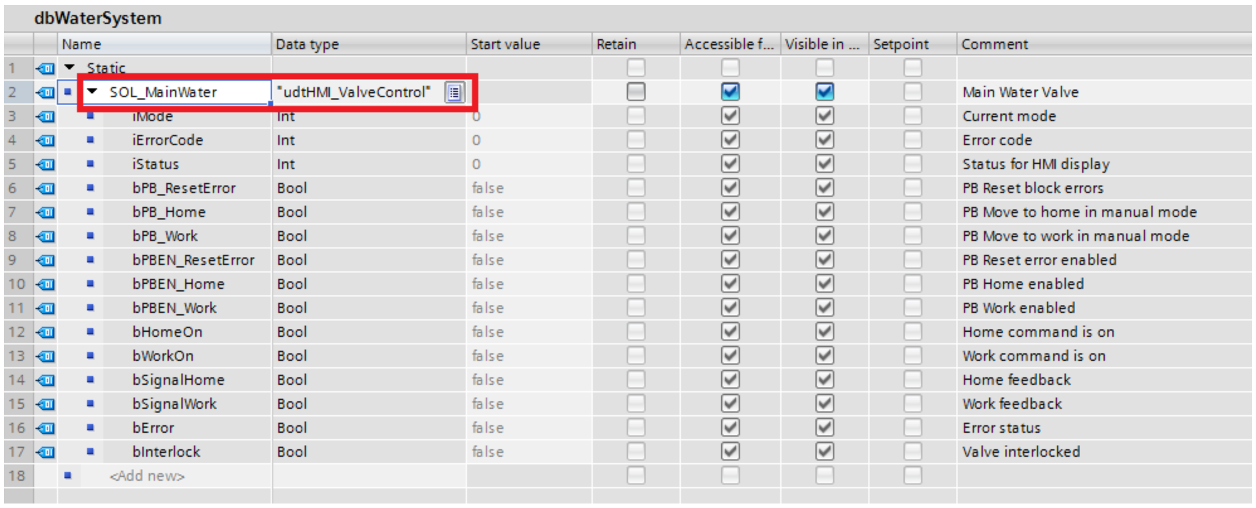
proper Data type. In this example, the UDT is named SOL_MainWater (SOL derives from Solenoid and is a DMC naming standard for Solenoid devices) but any name can be used.



dbWaterSystem

	Name	Data type	Start v
1	Static		
2	SOL_MainWater	"udtHMI_ValveControl"	
3	<Add new>		

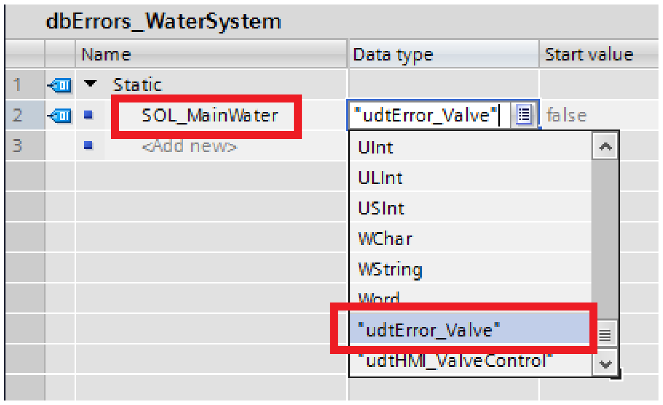
5. The User Defined Type has a large number of variables, with full details provided in the documentation for the specific Library Object that is being used. It can be expanded to see the variables available inside the UDT.



dbWaterSystem

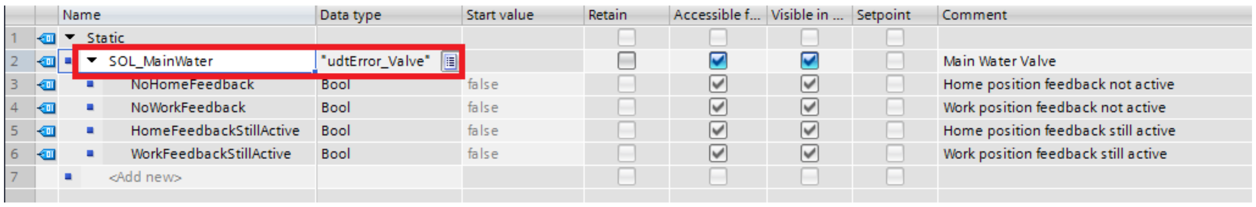
	Name	Data type	Start value	Retain	Accessible f...	Visible in ...	Setpoint	Comment
1	Static			☐	☐	☐	☐	
2	SOL_MainWater	"udtHMI_ValveControl"		☐	✓	✓	☐	Main Water Valve
3	iMode	Int	0	☐	✓	✓	☐	Current mode
4	iErrorCode	Int	0	☐	✓	✓	☐	Error code
5	iStatus	Int	0	☐	✓	✓	☐	Status for HMI display
6	bPB_ResetError	Bool	false	☐	✓	✓	☐	PB Reset block errors
7	bPB_Home	Bool	false	☐	✓	✓	☐	PB Move to home in manual mode
8	bPB_Work	Bool	false	☐	✓	✓	☐	PB Move to work in manual mode
9	bPBEN_ResetError	Bool	false	☐	✓	✓	☐	PB Reset error enabled
10	bPBEN_Home	Bool	false	☐	✓	✓	☐	PB Home enabled
11	bPBEN_Work	Bool	false	☐	✓	✓	☐	PB Work enabled
12	bHomeOn	Bool	false	☐	✓	✓	☐	Home command is on
13	bWorkOn	Bool	false	☐	✓	✓	☐	Work command is on
14	bSignalHome	Bool	false	☐	✓	✓	☐	Home feedback
15	bSignalWork	Bool	false	☐	✓	✓	☐	Work feedback
16	bError	Bool	false	☐	✓	✓	☐	Error status
17	bInterlock	Bool	false	☐	✓	✓	☐	Valve interlocked
18	<Add new>			☐	☐	☐	☐	

6. Add the error structure to error data block. In this example it is added to dbErrors_WaterSystem.



dbErrors_WaterSystem

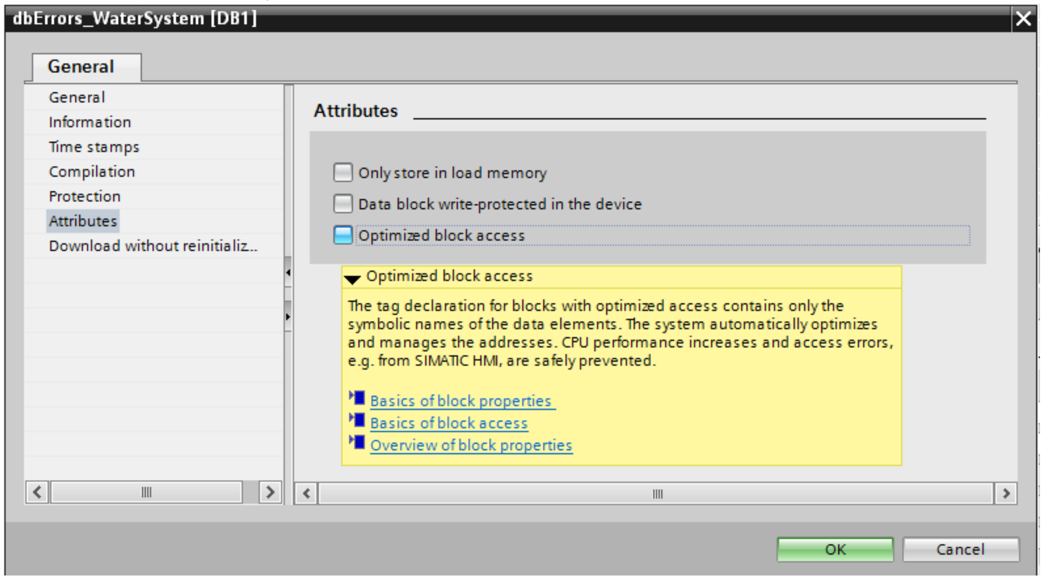
	Name	Data type	Start value
1	Static		
2	SOL_MainWater	"udtError_Valve"	false
3	<Add new>		



	Name	Data type	Start value	Retain	Accessible f...	Visible in ...	Setpoint	Comment
1	Static							
2	SOL_MainWater	"udtError_Valve"			✓	✓		Main Water Valve
3	NoHomeFeedback	Bool	false		✓	✓		Home position feedback not active
4	NoWorkFeedback	Bool	false		✓	✓		Work position feedback not active
5	HomeFeedbackStillActive	Bool	false		✓	✓		Home position feedback still active
6	WorkFeedbackStillActive	Bool	false		✓	✓		Work position feedback still active
7	<Add new>							

The Error Data Block needs to be non-optimized, for which the setting is available via the Data Block Properties by right clicking on the data block. For additional details see '5-

Siemens Open Library - Siemens HMI Alarm Generation.'



dbErrors_WaterSystem [DB1]

General

- General
- Information
- Time stamps
- Compilation
- Protection
- Attributes
- Download without reinitializ...

Attributes

- [ ] Only store in load memory
- [ ] Data block write-protected in the device
- [ ] Optimized block access

Optimized block access

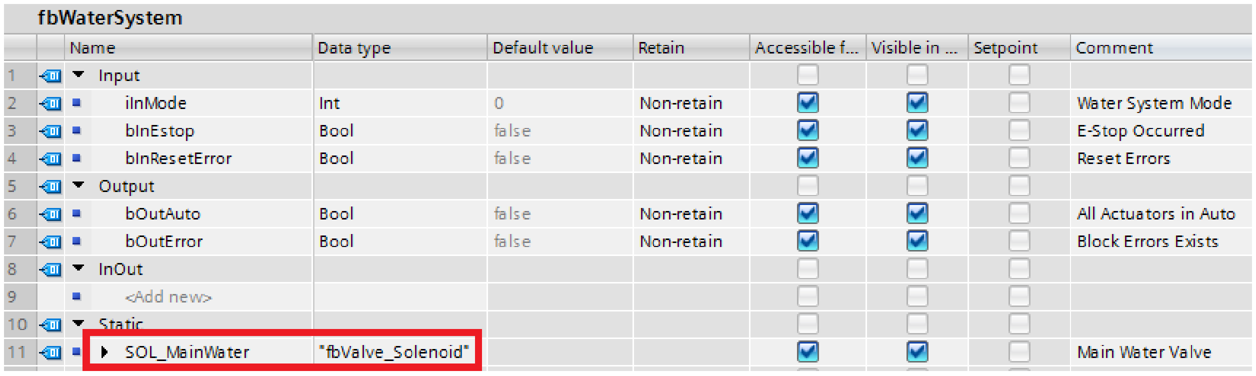
The tag declaration for blocks with optimized access contains only the symbolic names of the data elements. The system automatically optimizes and manages the addresses. CPU performance increases and access errors, e.g. from SIMATIC HMI, are safely prevented.

Basics of block properties
Basics of block access
Overview of block properties

OK Cancel

5.2.3. Create PLC Code

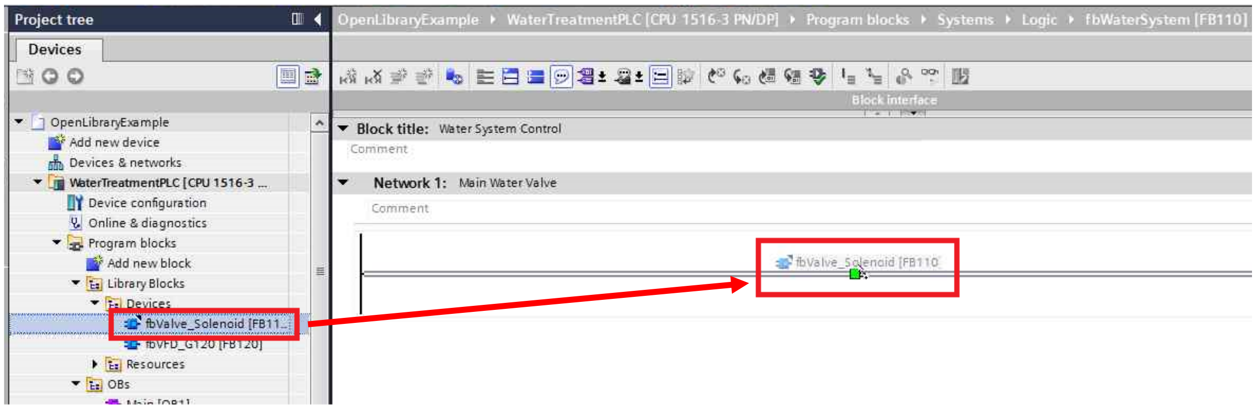
1. Create the function block that will contain the logic for the Solenoid valve. In this case we created a Function Block called fbWaterSystem. Add the inputs and outputs that correspond to the required logic for the water system.
2. Inside the function block that will contain the logic, add a Static Variable to contain the instance memory for the Library Object. In this example, 'SOL_MainWater" is added with data type 'fbValve_Solenoid'.



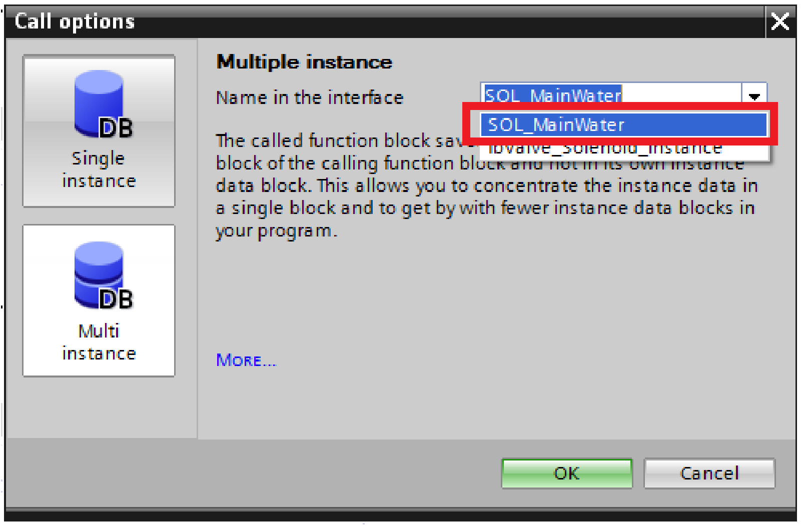
fbWaterSystem

	Name	Data type	Default value	Retain	Accessible f...	Visible in ...	Setpoint	Comment
1	Input				☐	☐	☐	
2	iInMode	Int	0	Non-retain	✓	✓	☐	Water System Mode
3	bInEstop	Bool	false	Non-retain	✓	✓	☐	E-Stop Occurred
4	bInResetError	Bool	false	Non-retain	✓	✓	☐	Reset Errors
5	Output				☐	☐	☐	
6	bOutAuto	Bool	false	Non-retain	✓	✓	☐	All Actuators in Auto
7	bOutError	Bool	false	Non-retain	✓	✓	☐	Block Errors Exists
8	InOut				☐	☐	☐	
9	<Add new>				☐	☐	☐	
10	Static				☐	☐	☐	
11	SOL_MainWater	"fbValve_Solenoid"			✓	✓	☐	Main Water Valve

3. Drag and drop the Object from the project tree Program blocks into the ladder logic where it will be used.

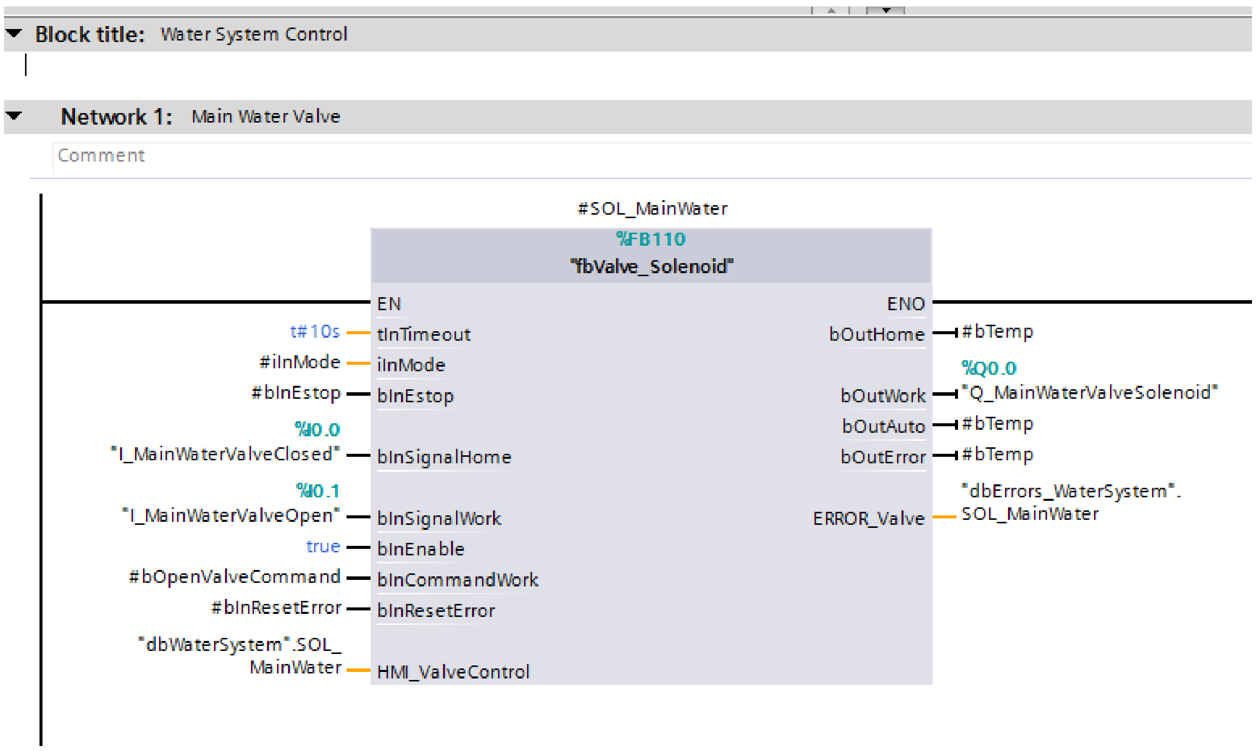


4. Choose to use this block as a multiple instance call and select the instance created in the static memory of fbWaterSystem. Alternatively, you can provide a name for the instance memory and TIA Portal will automatically add it to the static variables section of the block.



The called function block sav... block of the calling function block and not in its own instance data block. This allows you to concentrate the instance data in a single block and to get by with fewer instance data blocks in your program.

5. Assign tags to the inputs and outputs of the object. Detailed information for the meanings of each input and output can be found in the '4- Siemens Open Library - Detailed Block Overview' document.


Block title: Water System Control
Network 1: Main Water Valve
Comment
#SOL_MainWater
%FB110
"fbValve_Solenoid"
EN
ENO
t#10s tInTimeout
#iInMode iInMode
#bInEstop bInEstop
%I0.0 "I_MainWaterValveClosed" bInSignalHome
%I0.1 "I_MainWaterValveOpen" bInSignalWork
true bInEnable
#bOpenValveCommand bInCommandWork
#bInResetError bInResetError
"dbWaterSystem".SOL_MainWater HMI_ValveControl
bOutHome #bTemp
bOutWork %Q0.0 "Q_MainWaterValveSolenoid"
bOutAuto #bTemp
bOutError #bTemp
ERROR_Valve "dbErrors_WaterSystem".SOL_MainWater

5.3. HMI Development

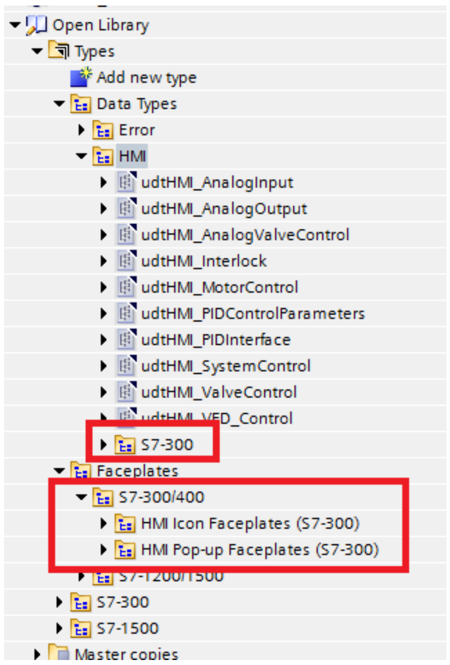
This section covers the use of the HMI Icon and the HMI Pop-up screen. The HMI library can be used as a fully functioning library, or it can be used as a building block for custom HMI Icons and Pop-ups.

5.3.1. Using SIMANTIC Visualization Architect (SiVArc) for HMI Development

One option for HMI development is to use SiVArc, a plug-in for TIA portal that allows for the automatic generation of screens and screen objects based off the controller program for the project. SiVArc is compatible with TIA Portal V14 and later, and is compatible with S7-1200 and S7-1500 PLC's, but is not compatible S7-300 or S7-400 PLC's. To implement SiVArc in a project, refer to "8 – SIMANTIC Visualization Architect" for a description of the process, and "4 – Detailed Block Overview" for the specifics of using it with a particular library object. If not using SiVArc, follow the directions below.

5.3.2. Special considerations for S7-300/S7-400 PLCs

If an S7-300 or S7-400 PLC is used in the project, the faceplates and UDTs created specifically for the S7-300 series PLC need to be used. The below instructions can still be followed, however, the separate faceplates need to used where applicable. Note: V2.1 is the last version of the library to support S7-300 and S7-400 processors.

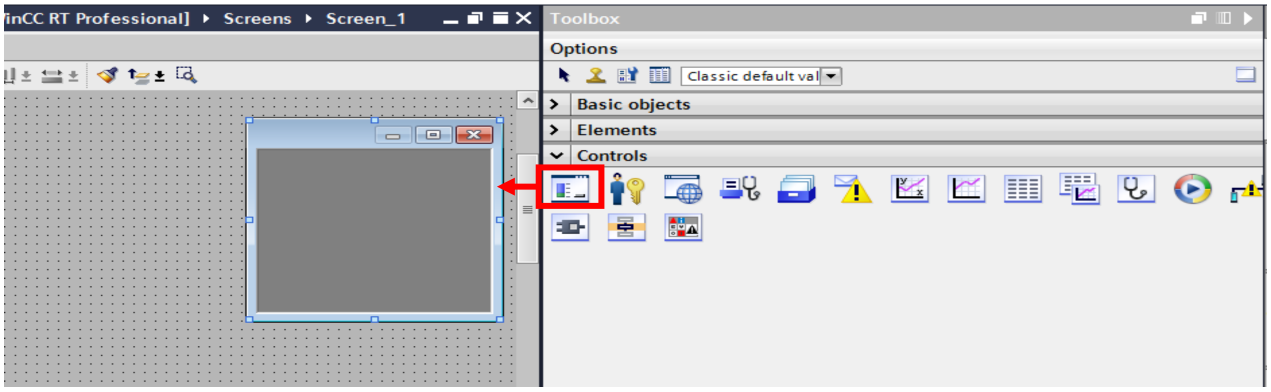


```
Open Library
  Types
    Add new type
    Data Types
      Error
      HMI
        udtHMI_AnalogInput
        udtHMI_AnalogOutput
        udtHMI_AnalogValveControl
        udtHMI_Interlock
        udtHMI_MotorControl
        udtHMI_PIDControlParameters
        udtHMI_PIDInterface
        udtHMI_SystemControl
        udtHMI_ValveControl
        udtHMI_VFD_Control
        S7-300
    Faceplates
      S7-300/400
        HMI Icon Faceplates (S7-300)
        HMI Pop-up Faceplates (S7-300)
      S7-1200/1500
    S7-300
    S7-1500
  Master copies
```

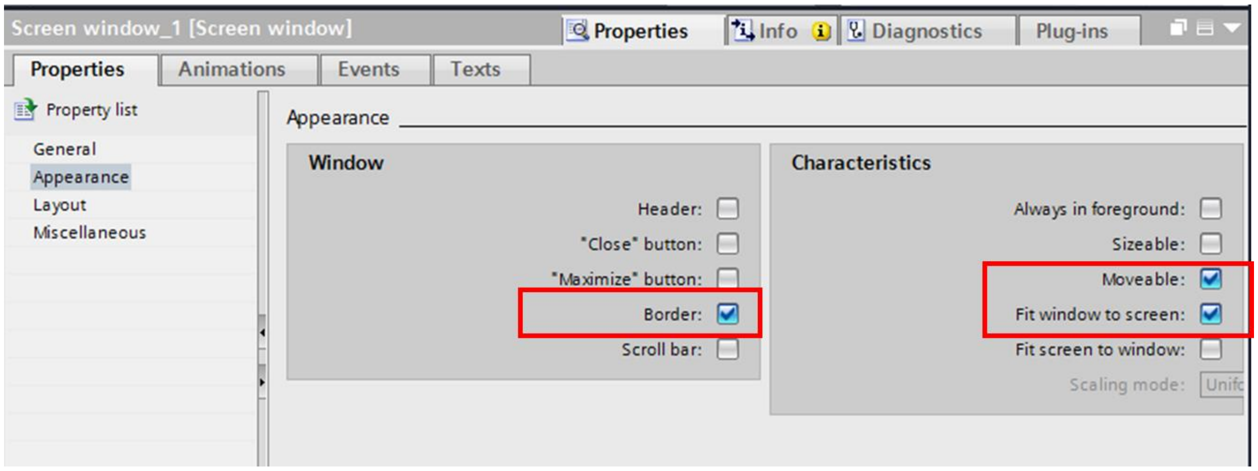
5.3.3. HMI Pop-up Creation and Configuration for WinCC Professional (no SiVArc)

This section covers how to use the library object screens associated with the WinCC pro implementation of this library to create pop-ups. In this method, one screen is created per type, and tag prefixes are used to determine the instance loaded.

1. Create a screen window. This is where the screens from the library will be loaded. Do this by dragging a screen window in from the toolbox, under the controls drop-down.



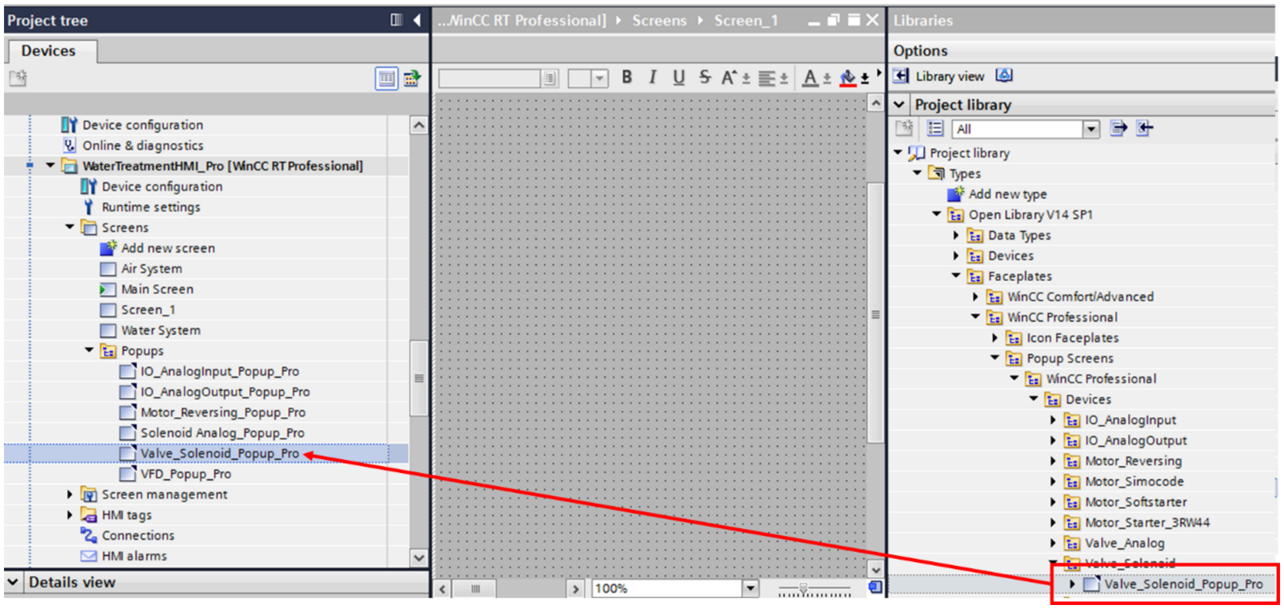
2. Change the appearance of the screen window. With the screen window selected, go to Properties>Appearance, and check only "Border" under window, and only "Movable" and "Fit screen to window" under Characteristics. This ensures the pop-up will display correctly, and can be moved on the screen.



Window:
- Header: ☐
- "Close" button: ☐
- "Maximize" button: ☐
- Border: ✓
- Scroll bar: ☐

Characteristics:
- Always in foreground: ☐
- Sizeable: ☐
- Moveable: ✓
- Fit window to screen: ✓
- Fit screen to window: ☐
- Scaling mode: Unifc

3. Repeat steps 1 and 2 on any screen that will have pop-ups.
4. Drag the desired pop-up screen(s) into the screens folder. It is recommended to make a "Popups" folder or similar for organization, but it is not required.

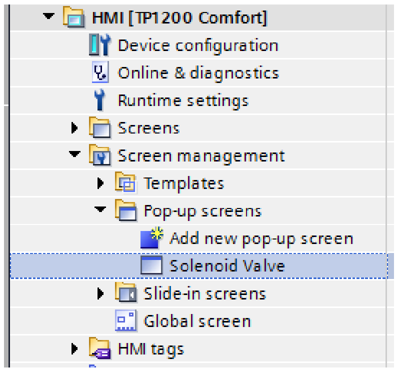


5. No interface is defined. Instead, the tag prefix for the instance is loaded when the screen is launched. This is defined on the corresponding HMI Icon.

5.3.4. HMI Pop-up Creation and Configuration for WinCC Comfort/Advanced (no SiVArc)

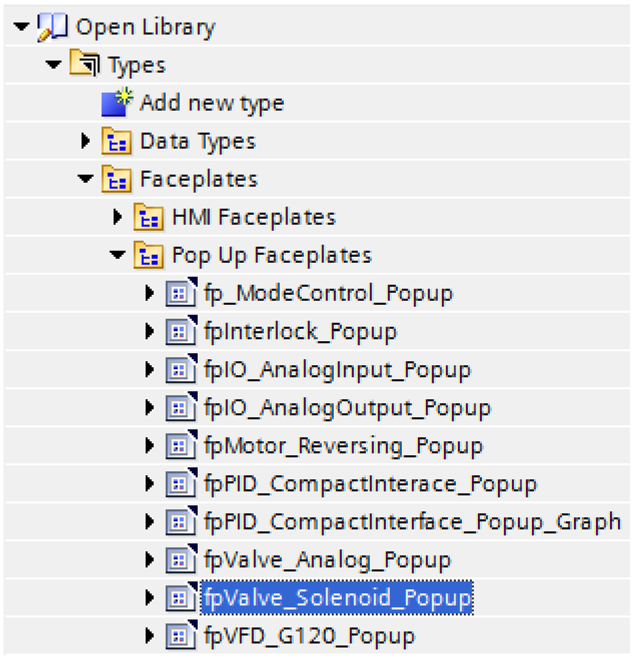
This section covers how to use the Pop-up faceplates on a pop-up screen. Each object will have a separate pop-up screen.

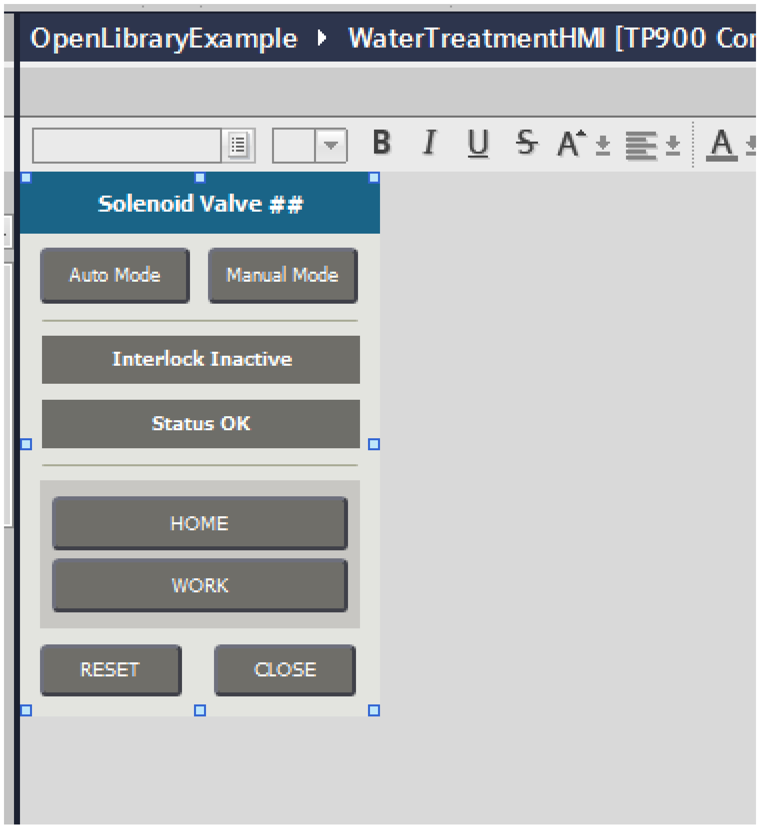
1. Create a new Pop-up screen for the Solenoid Valve. This is done by clicking the 'Add new pop-up screen' button in the Pop-up screens section. Give a name that identifies the specific object being used.



2. Drag the Pop-up Faceplate, fpValve_Solenoid_Popup in this example, from the library onto the new Pop-up screen. The faceplate can be resized to best fit the size of the HMI that is being used, and text size can be defined in the interface. Be sure that the Pop-up screen size is equivalent to the size of the faceplate. If the screen size is smaller than the faceplate,

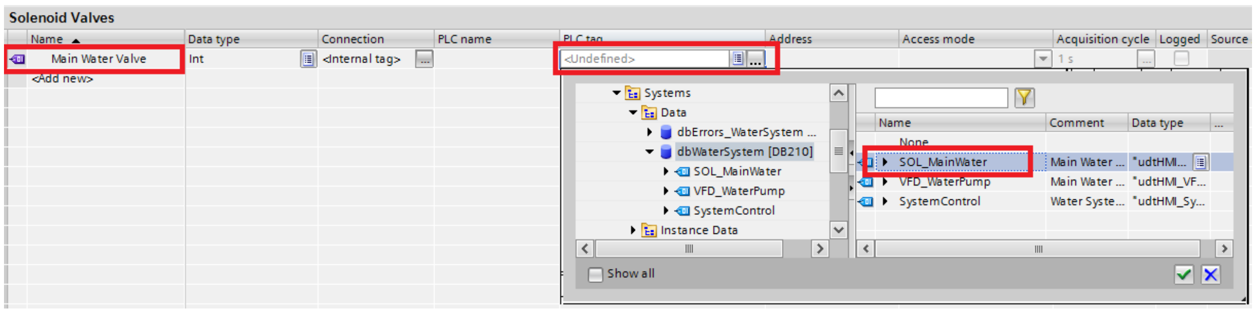
it will be cut off. If the screen size is larger than the faceplate, there will be a white boarder around the Pop-up.


Open Library
Types
Add new type
Data Types
Faceplates
HMI Faceplates
Pop Up Faceplates
fp_ModeControl_Popup
fpInterlock_Popup
fpIO_AnalogInput_Popup
fpIO_AnalogOutput_Popup
fpMotor_Reversing_Popup
fpPID_CompactInterace_Popup
fpPID_CompactInterface_Popup_Graph
fpValve_Analog_Popup
fpValve_Solenoid_Popup
fpVFD_G120_Popup



3. Create an HMI tag for the object in the HMI Tag Table and map it to the HMI UDT that was created on the PLC, in this case: “dbWaterSystem”.SOL_MainWater. As an additional option,

the tag can be automatically created by dragging and dropping the UDT directly from the PLC into the HMI Interface.
Below shows the method for creating the object tag from the HMI Tag Table:



Solenoid Valves

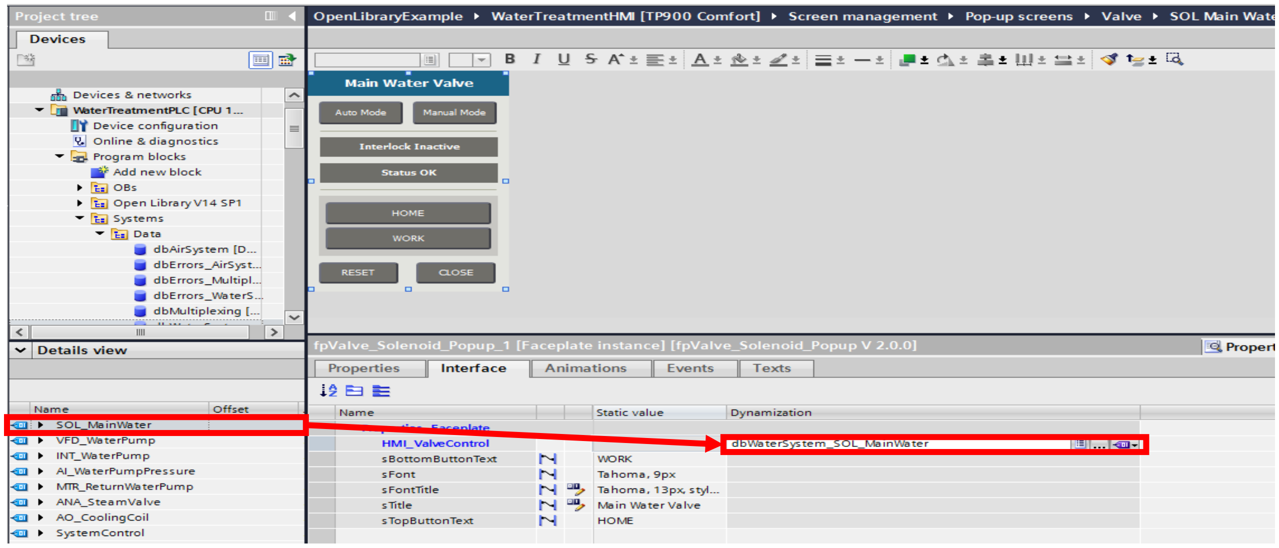
Name	Data type	Connection	PLC name	PLC tag	Address	Access mode	Acquisition cycle	Logged	Source
Main Water Valve	Int	<Internal tag>		<Undefined>			1 s		
<Add new>									

Systems
Data
dbErrors_WaterSystem ...
dbWaterSystem [DB210]
SOL_MainWater
VFD_WaterPump
SystemControl
Instance Data

Name	Comment	Data type
None		
SOL_MainWater	Main Water ...	"udtHMI_...
VFD_WaterPump	Main Water ...	"udtHMI_VF...
SystemControl	Water Syste...	"udtHMI_Sy...

Show all

And below shows how to have Portal automatically have the tags created for you, by dragging and dropping the UDT directly into the interface:

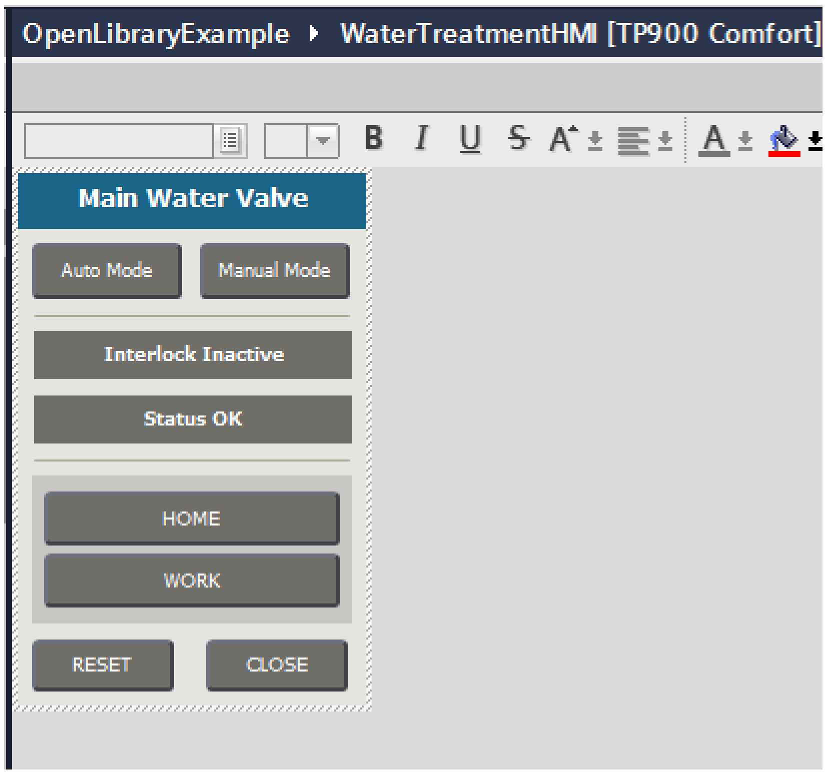


Name	Offset
SOL_MainWater	
VFD_WaterPump	
INT_WaterPump	
AI_WaterPumpPressure	
MTR_ReturnWaterPump	
ANA_SteamValve	
AO_CoolingCoil	
SystemControl	

Name	Static value	Dynamization
Properties Faceplate		
HMI_ValveControl		dbWaterSystem_SOL_MainWater
sBottomButtonText	WORK	
sFont	Tahoma, 9px	
sFontTitle	Tahoma, 13px, styl...	
sTitle	Main Water Valve	
sTopButtonText	HOME	

4. Configure the required variables in the Interface of the faceplate. If the Interface tab is not visible, right click on the faceplate, select properties, and the window should appear on the bottom of the screen. Each library object has a slightly different interface. The details of each can be seen in the documentation for the specific object. In this case, the interface contains the following variables: HMI_ValveControl, sBottomButtonText, sFont, sFontTitle, sTopButtonText, and sTitle. Configure the HMI_ValveControl using the HMI tag created in

step 3, or by dragging and dropping the tag from the PLC. Configure any strings required by typing the fixed text into the interface.

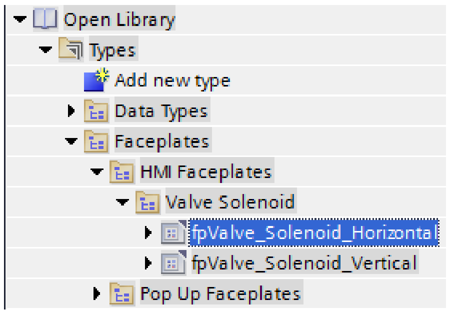


5.3.5. HMI Icon Creation and Configuration (no SiVArc)

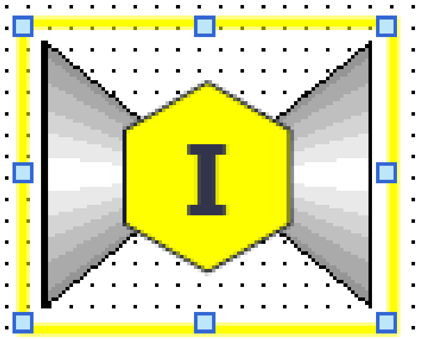
This section covers how to configure the HMI Icon. The library contains several predefined HMI Icons. Many more can be created, or existing icons can be modified using the Symbol Object Library. For details on customizing objects, see '7- Siemens Open Library – Library Customization.'

Details regarding the functionality, colors, and interface to each HMI Icon can be found in the documentation for that object.

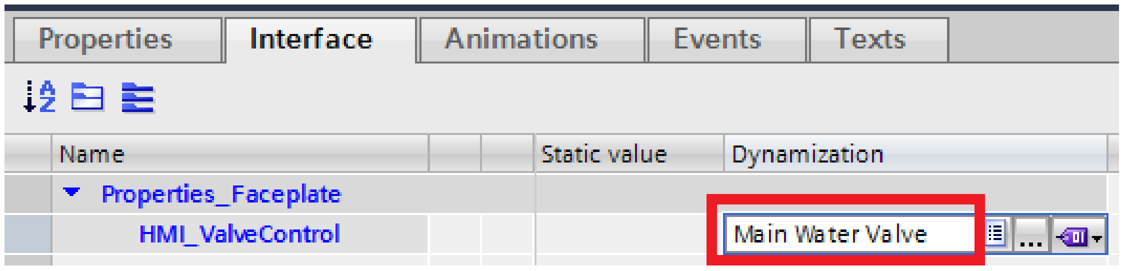
1. Navigate to the correct HMI Icon Faceplate in the Open Library. In this case we will use fpValve_Solenoid_Horizontal



2. Pull the Icon Faceplate into the desired screen. If the device is rotated, the 'I' will be at the angle of the device. There is a Horizontal and Vertical version of all devices to allow for vertical alignment with a correctly oriented interlock symbol.



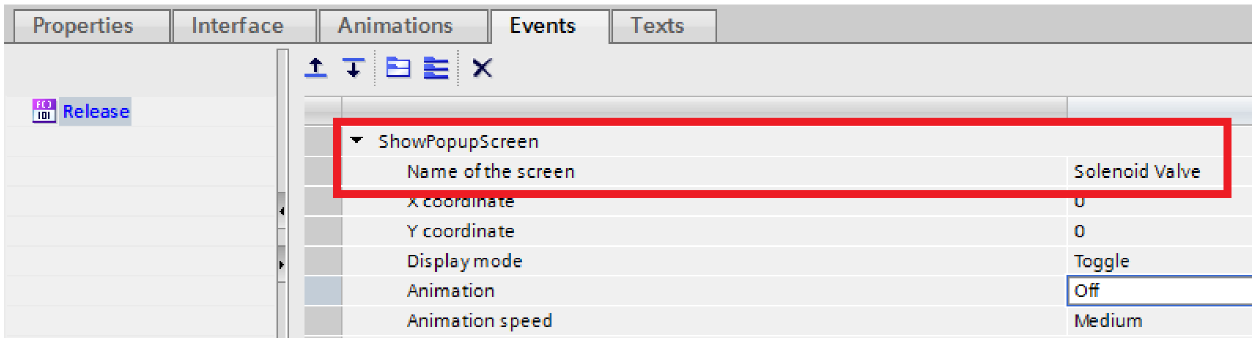
3. Configure the Interface to the Icon Faceplate using the tag configured in the HMI Pop-up, or by dragging and dropping the UDT from the PLC.



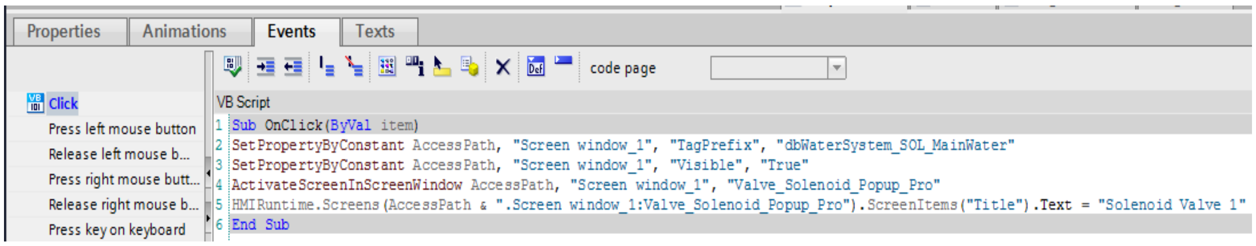
Name	Static value	Dynamization
Properties_Faceplate		
HMI_ValveControl		Main Water Valve

4. Create an event that will open the Pop-up window in runtime when the valve on the HMI screen is clicked. For WinCC Comfort/Advanced, this event can be a "ShowPopupScreen" event, as shown in the first image below. For WinCC Pro, this event needs to be a visual

basic script event, as shown in the second image below, so that the screen gets loaded into the screen window with the correct tag prefixes.



ShowPopupScreen	
Name of the screen	Solenoid Valve
X coordinate	0
Y coordinate	0
Display mode	Toggle
Animation	Off
Animation speed	Medium



```
Sub OnClick(ByVal item)
SetPropertyByConstant AccessPath, "Screen window_1", "TagPrefix", "dbWaterSystem_SOL_MainWater"
SetPropertyByConstant AccessPath, "Screen window_1", "Visible", "True"
ActivateScreenInScreenWindow AccessPath, "Screen window_1", "Valve_Solenoid_Popup_Pro"
HMIRuntime.Screens(AccessPath & ".Screen window_1:Valve_Solenoid_Popup_Pro").ScreenItems("Title").Text = "Solenoid Valve 1"
End Sub
```

The first line in the script sets the tag prefix for the screen window. The second line makes the screen window visible. The third line loads in the desired pop-up screen. The fourth line defines the title on the pop-up.