Smart People. Expert Solutions.

SIEMENS OPEN LIBRARY

1 - Library Overview and Architecture

JUNE 11, 2019

1. Contents

1. Purpose

The purpose of this document is to provide a general overview of the library and to discuss decisions in program architecture that make best use of the library. Additionally, this document will discuss system compatibility, testing, and how to contribute to growing the Open Library.

2. Intended Use

This document is intended to be the starting point for anyone utilizing the Open Library for PLC and HMI Development.

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	Added information on Supplementary Blocks
1.3	2016-10-11	DMC	No Changes
2.0	2017-11-3	DMC	Added new supported blocks, including WinCC Pro.
2.1	2018-11-29	DMC	Added many new documented blocks and new Alarm Generator application.
3.0	2018-12-05	DMC	Added documentation for many new supported blocks.
4.0	2019-02-15	DMC	Updates for device simulation and V15 release.

4. Open Library License

5. Hardware and Software Compatibility

This library was developed in TIA Portal V13 SP1 and has since migrated to V14, V14 SP1, and V15. It has been tested on the S7-1200 and S7-1500 platforms. The PLC objects can be used with any HMI, however, the configuration of the faceplates is only available using a Comfort Panel or WinCC Advanced, and have been tested on a 7" Comfort Panel.

6. Background

The Siemens Open Library was developed by DMC Inc. over many years. Through a joint collaboration with Siemens Industry, DMC documented the library and released it as an Open Source Library open for anyone to use and distribute. The Library is Open Source and will allow for users to contribute content to help enhance the overall availability of functionality to the greater Siemens User Community.

7. Target Platforms

This section covers the target platforms that apply to the library.

7.1. PLC Target

This library is targeted at S7-1200 and S7-1500 applications. Due to improvements made with the newer processors, TIA portal allows symbolic use of UDTs on the HMI for S7-1200 and S7-1500, which is not available on older PLC platforms. The PLC blocks will still compile and function on S7-300 and S7-400 applications, but the HMI mapping will not be available.

7.2. HMI Target

The HMI used for this library will be a Siemens Comfort Panel or WinCC Advanced RT or WinCC Professional. Due to the limitations of a Basic Panel, the HMI pop-up structures and symbolic addressing will not work on a Basic Panel. The HMI pop-ups can be modified to be used, but will require individual mapping of each point by the end user.

This section covers the testing that was completed for use of the project.

7.3. PLC Testing

The Siemens Open Library was tested on an S7-1500 using TIA Portal V15. Most blocks were tested using simulation, however, some of the blocks, such as G-Series VFD, were tested with actual hardware. DMC makes no guarantee of functionality for any Siemens Open Library objects on a given platform.

7.4. HMI Testing

The Siemens Open Library was tested on a 7'' WinCC Comfort Panel. Faceplates were tested on actual hardware and in simulation mode to verify functionality.

8. Support Resources

This section covers available support resources for use of the Siemens Open Library.

8.1. Technical Support

Since this library was not developed by Siemens, it is not supported by Siemens Technical Support. For questions or complicated implementations, DMC is willing to provide support, however this support is not free. Questions can be directed to the following email address or phone number:

Contribute@OpenPLCLibrary.com
312-255-8757

8.2. Documentation

The following documents have been created to help users implement, customize, and maintain the Siemens Open Library

1. Library Overview and Architecture (This Document)
   Overview of the Library and design considerations.

2. Initial Setup
   Provides information on how to do the initial configuration required for all library objects.

3. Example Object Configuration
   Walks through how to use and configure a single object and uses fbValve_Solenoid as an example.

4. Detailed Library Block Description
   Provides detailed information for the functionality, inputs, outputs, UDTs, and Faceplates for each object.

5. Siemens HMI Alarm Generation
   Provides information for how to generate alarms for use in WinCC Comfort or WinCC RT Advanced using the automated tools.

6. PID Block Configuration
   Walks through how to use and configure the PID Object, as it has special instructions.

7. Customizing Library Objects
   Discusses how to modify the library to change functionality or appearance.

8. SIMANTIC Visual Architect (SiVArc)
   Discusses how to implement SiVArc in projects for the automatic generation of screens and screen objects.

9. Device Simulation
   Provides an overview of the Open Library approach to simulation, including a detailed walkthrough and instruction set.

9. Library Overview

This section outlines all objects that the library currently contains.

9.1. Motor Control

9.1.1. G Series VFD Control – fbVFD_GSeries

The G Series VFD Control Function Block is utilized for use of Siemens G Series VFD or Micromaster VFDs using Standard Telegram 1. This block does not use a technology object. This allows for a lower CPU requirement for each drive. It utilizes Standard Telegram 1 and has been tested on a G120, however, it will function with any drive using Standard Telegram 1 on either Profibus or Profinet.

9.1.2. G Series Advanced VFD Control – fbVFD_GseriesAdvanced

This function block builds on top of the regular G Series function block by adding asynchronous current and torque parameter reading as well as ramp up and ramp down parameter writing. Due to the asynchronous communication method, this block can only be instantiated *20 times* on each processor. Any additional instances may cause unreliability of asynchronous parameter reads/writes.

9.1.3. Analog VFD Control – fbVFD_Analog

The fbVFD_Analog Motor Control Function Block is utilized for use of a VFD controlled with digital and analog signals. The error UDT is kept identical to the GSeries error UDT, however, some error outputs are not utilized by this block.

9.1.4. Digital Single Speed Motor – fbMotor_Reversing

The Reversing Motor Control Function Block is utilized for motors started and stopped using digital outputs. This block functions with single direction or reversing motors.

9.1.5. Simocode Single Speed Motor – fbMotor_Simocode

The Simocode Motor Control Function Block is utilized for controlling motors via a Type 2 Simocode Pro V Module. This block can be configured for an Overload Relay, Direct Starter or Reversing Starter control functions.

9.1.6. Softstarter Motor – fbMotor_Softstarter

The Soft Starter Control Function Block is utilized for controlling motors via a SIRIUS 3RW Soft Starter. This block can be configured for the 3RW30 and 3RW40 models.

9.1.7. Softstarter 3RW44 Motor – fbMotor_Softstarter3RW44

The Soft Starter Control Function Block is utilized for controlling motors via a SIRIUS 3RW44 Soft Starter.

9.2. Valve Control

9.2.1. Two State Solenoid Valve – fbValve_Solenoid

This library item is to be utilized with two state solenoid valve applications. It can be used for spring close, spring open, or double acting solenoids.

9.2.2. Analog Valve – fbValve_Analog

This library item is to be utilized with a valve controlled using an analog output. This block will work with any valve that is controlled through an analog output from the PLC.

9.2.3. Hydraulic Valve – fbValve_Hydraulic

This block controls a hydraulic valve and provides easy integration into a pump/valve manifold combination system. It has similar functionality to fbValve_Solenoid with additional control features such as monitoring hydraulic pump status and momentary commands.

9.3. Inputs/Outputs

9.3.1. Analog Input with Scaling and Alarms – fbIO_AnalogInput

This library object scales analog inputs and provides setups for alarms.

9.3.2. Analog Output with Scaling – fbIO_AnalogOutput

This library object scales process values to be used for Analog Outputs.

9.3.3. Digital Input – fbIO_DigitalInput

This library object reads a digital input and allows manual control from the HMI.

9.3.4. Digital Output – fbIO_DigitalOutput

This library object controls a digital output and allows manual control from the HMI.

9.4. General Control

9.4.1. System Control

This library object is a User Defined Type only that is used to provide mode information to other Library Objects, and is controlled by HMI Faceplates.

9.4.2. Run Control - fcSystemRunControl

This library object provides simple start/stop logic from both the PLC and HMI. It also allows for interlocking on certain states and gives feedback for control and display.

9.4.3. Interlock - fbInterlock

This library object is used to provide operator information for multiple values that are interlocking a process. It has inputs for items that may be interlocking a process, and informs the operator of what interlocks are preventing operation via an HMI faceplate.

9.4.1. Permissive – fbPermissive

Similar to Interlock, this block sums permissive conditions and allows for custom display on the HMI.

9.4.2. PID Control – fbPID_Compact

The S7-1200 and S7-1500 provide PID Technology objects that are best practices for use, therefore this would not replace those blocks, but would provide an interface to the technology object on the HMI as well as tuning, mode, and manual control functionality.

9.4.3. Standard Alarm Interface - fbAlarmWarning

The Standard Alarm Interface provides an easy way to setup additional alarms.

9.5. Load Cell Modules

9.5.1. SiwarexU – fbSiwarexU

This block provides a basic weighing interface when using a Siemens SiwarexU load cell module.

9.5.2. SiwarexWP 321 – fbSiwarexWP321

This block provides a basic weighing interface when using a Siemens SiwarexWP 321 load cell module.

9.5.3. SiwarexWP 321 Calibration – fbSiwarexWP321_Calibration

This block works in coordination with the fbSiwarexWP321 block and adds calibration routine functionality on top of standard weighing functions.

9.6. Process Control

9.6.1. First-In-First-Out (FIFO) Queue – fbFIFO

This block implements a standard FIFO queue utilizing variant data types to work dynamically with any array structure. It includes standard queuing commands such as push, pop, and peek.

9.6.2. Flow Totalizer – fbFlowTotalizer

This function takes a raw input from a flow meter and sums the weight total over time. In addition, it has features such as reset, pause, and simulation.

9.6.3. Level Monitoring – fbLevelMonitor

This block monitors the level of a tank or container with an analog level sensor and can alert on preset high and low level set points with a configurable debounce.

9.6.4. Advanced Level Monitoring – fbHopperLevel

This block utilized multiple fbLevelMonitor instances to add more level monitoring features. This includes more alarms (low-low, low, high, and high-high) as well as positive feedback that the container level is within the specified allowable range.

9.6.5. Pulsing Output – fbPulser

This block takes a input command and turns it into a pulsing output with configurable on/off times as well as a maximum count limit.

9.6.6. PWM Control – fbPWM

Similar to fbPulser, this block outputs a rough PWM signal with a configurable period and on time.

9.6.7. Rate Calculation – fbRateCalc

This block calculates the derivate (rate) of any process value over time with configurable sampling period and averaging.

9.6.8. Running Average – fbRunningAverage

This block calculates the running average of a single input over a specified period.

9.6.9. Sequencer – fbStepSequencer

This library object provides an interface for making sequencers and state machines. Each block has inputs for current step, next step, step done and step delay. Multiple blocks can be pointed to the same instance to create a sequence or state machine. Each block's outputs can be mapped to respective steps, and those steps processes.

9.7. Additional Control Devices

9.7.1. Airlock (with motor) – fbAirlock_Motor

This block is a combination of the motor block (fbMotor_Reversing) and a dependent solenoid valve (fbValve_Solenoid) without feedback. The function of the valve is tied to the motor control in automatic mode, however can function independently in manual mode. When the motor is commanded to run, the valve is automatically opened. This functionality is common for a metering airlock.

9.7.2. Airlock (with G-Series VFD) – fbAirlock_VFD

Identical to fbAirlock_Motor with the exception that is controls the motor with a VFD instead of a reversing contactor.

9.7.3. Danfoss VFD – fbVFD_Danfoss

This block controls a Danfoss VFD with a standard telegram. Functionality is identical to fbVFD_GSeries.

9.7.4. Unidrive Servo – fbServo_Unidrive

This block controls a Control Techniques (Emerson) servo drive with speed control and a standard telegram. Functionality is identical to fbVFD_GSeries.

9.7.5. MicroMotion (Modbus) – fbMicroMotion_Modbus

This block controls a MicroMotion Coriolis flow meter using the Modbus protocol. This block provides information from the meter such as flow rate, flow total, and liquid density.

9.7.6. MicroMotion (Profibus) – fbMicroMotion_Profibus

Identical to fbMicroMotion_Modbus with the exception that it uses the Profibus protocol for communication to the meter.

9.8. Supplementary Blocks

Included in the library is a section for both S7-300 and S7-1500 labelled “Supplementary”. These are blocks are included in library for use in auxiliary applications, but are provided as-is and without documentation. As documentation is added for these blocks, they will be merged into the appropriate library location. These blocks will include any user submitted blocks that are submitted without documentation.

10. Library Standards

This section discusses standards that are used in the library.

10.1. Naming Conventions

The Siemens Open Library uses a specific set of variable naming prefixes. These prefixes increase readability and speed up the coding process once they are understood. Each item will contain two prefixes and then a meaningful name. The first prefix is variable type. The second prefix corresponds to the data flow (input, output, local variable) and can sometimes be blank. The standard for Siemens S7 PLC and HMI programs is as follows:

10.1.1. Variable Prefix

TYPE	PREFIX	EXAMPLE
BOOL	b	bInAuto
BYTE	by	byOutStatus
INT	i	iInMode
REAL	r	rActualWeight
WORD	w	wInModbusAddress
DWORD	dw	dwInFlow
TIME	t	tInUpdateTimeout
STRING	s	sTitle
USER DEFINED TYPE	None	HMI_VFD_Control
MULTI INSTANCE FUNCTION BLOCK	None	ParameterReadWrite

10.1.2. Data Flow Prefix

TYPE	PREFIX	EXAMPLE
FB/FC Input	In	bInAuto
FB/FC Output	Out	byOutStatus
Internal Static Variable	<blank>	rActualWeight
Internal Temporary Variable	Temp	tTempElapsedTime

10.2. Using Constants

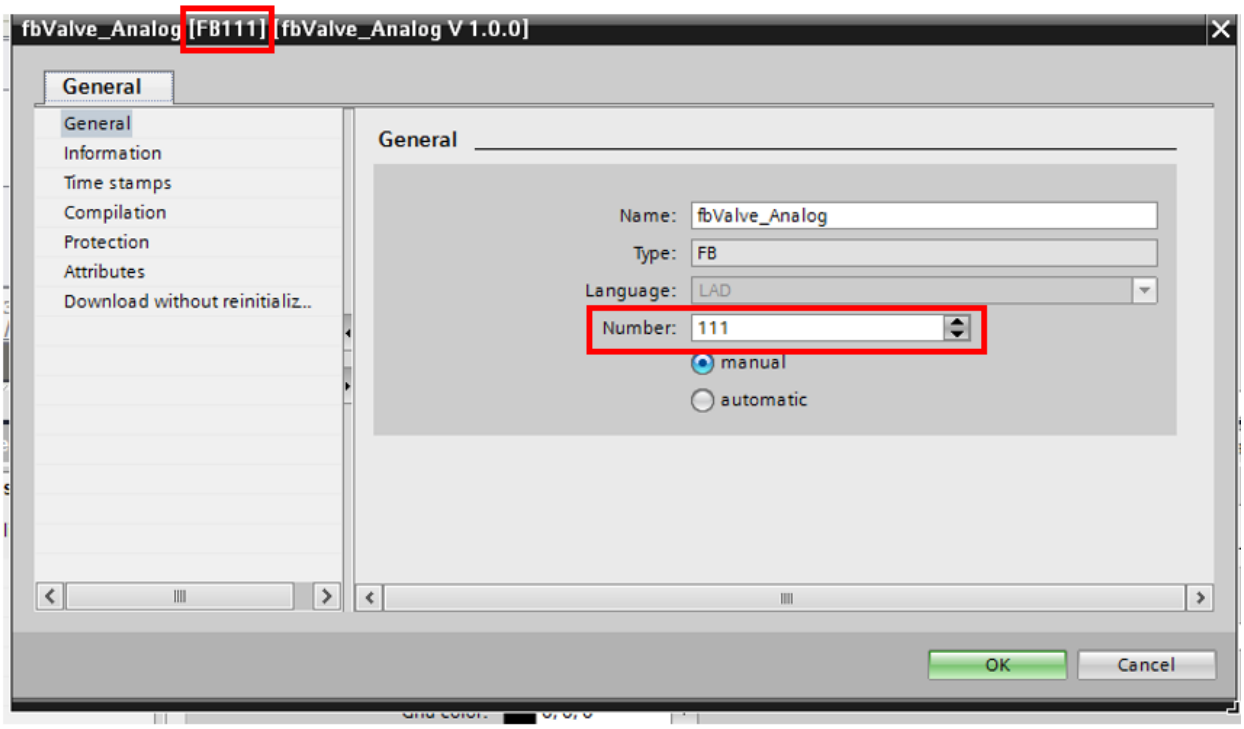
The Siemens Open Library takes advantage of the User Constants feature of PLC tag tables. These constants should be used wherever possible instead of hardcoding values in function blocks. The Library constants are located in the 'Open Library' tag table in the Master Copies section. Organization of the constants is done using dot (period) notation. The leftmost parts of constant names are general categories; categories or names get more specific toward the end. Below is a short description of the larger categories.

10.2.1. Constant Categories

CATEGORY NAME	DESCRIPTION
Modes	Top level system modes. See Section 11.1.2 for definitions.
Modes.PID	PID_Compact modes used in fbPID_CompactInterface block.
HMI	Any constants used for HMI display or logic.
HMI.Status	Controls color animations for HMI objects.
IoScaling	Used to scale any value read in from IO.
IoScaling.S7Analog	Contains constants for all S7 series AI cards and all sensor types.
Estop	Defines the action taken for an output when an E-Stop occurs.
DataTypes	Contains minimum and maximum limits for data types.

10.3. Data and Function Block Numbering

Each data and function block requires a unique identifying number. This number is typically set automatically by Portal, but it can also be manually modified. In order to avoid conflicts with this numbering system, it is suggested that the Siemens Open Library be imported before any custom data or function blocks are created. This insures that each block has a unique identifying number. If two or more blocks are using the same number, it is possible to resolve the conflict by changing the assigned number. This can be done by right clicking the block and selecting properties. Under the general tab, select a unique identifying number for that block.


fbValve_Analog [FB111] [fbValve_Analog V 1.0.0]
General
General
Information
Time stamps
Compilation
Protection
Attributes
Download without reinitializ...
General
Name: fbValve_Analog
Type: FB
Language: LAD
Number: 111
manual
automatic
OK
Cancel

11. Program Architecture

This section covers overall architecture and general design decisions. While these design outlines do not need to be adhered to for the blocks to function, they demonstrate why some design decisions were made with the blocks.

11.1. Modes and System Control

Mode is an integer that will be an input into all library objects that allow for HMI control. This section describes how to best use Modes inside your project.

11.1.1. Overview

The Library Utilizes a Mode architecture that is used to control a given block in Off, Auto, or Manual modes. A single global mode is typically used for small systems, and multiple system modes are typically used for larger systems. For example, if the system is a standalone skid, then a single mode can apply to all objects. If the system is an entire manufacturing line with multiple sections, then it is often best to create a mode for each subsystem, however, this is not required.

11.1.2. Mode Definitions

A typical project will have a mode for each major subsystem, so that all equipment in a given area carries an identical mode. The library blocks will have the following modes:

Mode	Value	Description
Stop	0	All Outputs disabled. Nothing in the system can run.
Auto	1	All blocks are in Auto. Each block will use the PLC configured values and logic for control.
Manual	2	All blocks are in Manual. Each block will use the HMI for control.
Independent	10	Each block mode will be set via the HMI. This allows for some devices to be in automatic and other to be in manual based on HMI mode selection. Most current industry systems maintain a global mode of Independent, which allows for switching of devices to an individual mode independent of the mode of other devices.

11.1.3. Using Mode for a Small System

For a small system a single global mode will need to be created. This mode will be passed into each Library Object that is created, and will be controlled by the HMI. There is no Function Block for controlling the mode, as that will be done from the HMI Pop-up faceplate.

11.1.4. Using Mode for a Medium or Large System

For a medium or large system, there will usually be multiple modes configured, where each mode corresponds to a subsystem of the application. This mode will be passed into each Library Object that is part of that subsystem, and will be controlled by the HMI. There is no Function Block for controlling the mode, as that will be done from the HMI Pop-up faceplate, which will need to be added to the HMI for each subsystem.

The values for Mode should be passed into the function block that corresponds to that subsystem and controlled from the customer created function block.

11.1.5. Changing Global Mode

If Global Mode Changes To:	Then Function Block Mode Changes To:
Auto	All function blocks change to Auto mode
Manual	All function blocks change to Manual mode
Independent	All function blocks maintain their previous mode, however, each block mode can now be changed independently via the HMI interface.
Stop	All function blocks change to Stop mode

11.1.6. Disable Modes

For some systems, the HMI should always be in control of the mode for each object. This can be accomplished by using the constant 'Modes.Independent' as the input for each library block, which will allow the HMI pop-up to control the mode for a given object.

11.2. Memory Management

This Section will cover information pertaining to standard storage of memory required for the Library Objects.

11.2.1. Function Block Memory

All Function Blocks require instance memory in order to be utilized, as they contain many variables that need to be retained across multiple scans. Function Block memory can be allocated as Multi-instance or Single Instance, however, Siemens best practices would recommend the use of a Multi-instance. The library will utilize a single instance for the PID interface, which is done to preserve all of the built in functionality for PID_Compact.

11.2.2. Using Instance Memory

The Siemens Open Library was designed to provide all required information as part of the HMI or Error UDTs or outputs from the Function Block. No memory contained inside of the instance memory of the Function Block should be used. This is to ensure proper operation of the block.

11.2.3. Multi-instance Memory Model

All documentation will utilize Multi-instance usage of Function Blocks. This is done to reduce overall number or instance data blocks that will need to be created, and to improve overall layout of the project structure.

11.2.4. Single Instance Memory Model

All blocks can be utilized as part of a single instance call of the block. This increases the overall number of instance data blocks, but will not impair the functionality of any of the Library Blocks.

11.3. HMI Display Status and Errors

Most Library function blocks take advantage of standard HMI display dynamics. The two commonly used in the Library are for actuator status and error displays.

11.3.1. Status

Status is indicated on the HMI by changing the color of the actuator or device. HMI indication is done through an integer in the HMI control structure named 'iStatus'. Status is not used for device control popups. The 'iStatus' integer is meant for HMI indication only and is therefore not recommended to be used in PLC logic.

11.3.2. Error Codes

Error codes are used to display dynamic text error messages in the device popup faceplates. The Library function blocks utilize 'fbErrorScroller' to scan through active alarms. Scrolling is accomplished using the 'iErrorCode' integer in the HMI structure. Because this value is scrolling, 'iErrorCode' is for HMI display only and should not be used in PLC logic. Boolean status bits should be used for logic instead.

12. Device Simulation

Starting in version 4.0, released for TIA Portal V15, many of the device function blocks in the Open Library include a built-in software simulation feature. Open Library simulation allows device blocks to respond in a no-error, running state without controlling physical outputs or requiring physical input feedback. Simulation is not controllable via the HMI interface and must be toggled using the block interface on the PLC. See Document 9 (Device Simulation) for more details.

13. Alarm Usage

Siemens Comfort Panels and WinCC Advanced Runtime use bits out of words to trigger alarms, and there are no methods to configure Boolean alarms. This library, however, utilizes bits for all alarms. To automate this process, the Siemens Open Library includes both a Microsoft Excel macro and stand alone executable that take a data block of Booleans and generates alarms in a WinCC format. For detailed information, please see '5- Siemens Open Library- Siemens Alarm Generation.'

13.1. General Alarms

Document 5 Siemens HMI Alarm Generation 9, as well as other supporting documents, contains instructions on adding error structures to error data blocks for each device. This same process should be used for creating individual alarms that are not covered within the scope of a device (e.g. high temperature, path blocked, door open, etc.). Single error bits should be placed within 'dbErrors' data blocks and given descriptive comments. These error bits may be placed inside structures for organization or just within the data block top level. Individual error bits may be alarms or warnings. These bits should then be written to in system-level logic using fbAlarmWarning. It is important to note that all auto-generating alarm tools will mark any error bit as an 'Error' in the HMI. Any warning bits will need to be manually changed to 'Warning' after import.

13.2. Alarm Generator

Included in the zip of this library is a Windows application written in C# .NET to allow for quick generation of WinCC Comfort, Advanced, or Professional alarms. This application takes in one or many ".udt" and/or ".db" files exported from TIA Portal and generates a ".xlsx" file that can then be inported into the WinCC alarm configuration section.

13.3. Alarm Special Considerations

The alarm generator has the following special considerations in order for the system to function:

1. The Data Block must be non-optimized (applies to 1200/1500 only and is accessible via the properties of the Data Block).
2. The User Defined Type, Struct, and/or individual Boolean comments will be utilized for the alarm text, so it is important to put in meaningful comments on each alarm.

14. Custom Modifications

Once blocks are pulled into the Project Library they can be modified to add or remove functionality or change appearance on the HMI. Additionally, it is possible to create a new global library, or add to the Siemens Open Library any block modifications that need to be used across multiple projects. There are special considerations for modifying UDTs that are used with S7-300 or S7-400 processors, and details for how to modify the library can be found in 'Siemens Open Library – Custom Modifications' document.

15. Contributing to the Library

The Siemens Open Library is an open source library, and contributions, improvements, and changes are welcome from anyone who has improved or added to the library. Please submit any custom objects to DMC by emailing them to Contribute@OpenPLCLibrary.com.

Library objects may not be added immediately but will be added as additional versions of the library are released.