POLITECNICO DI TORINO

Master of Science in

Mechatronic Engineering

Supervisor:

Professor Luigi Mazza

Candidate:

Hamid Hosseinzadeh

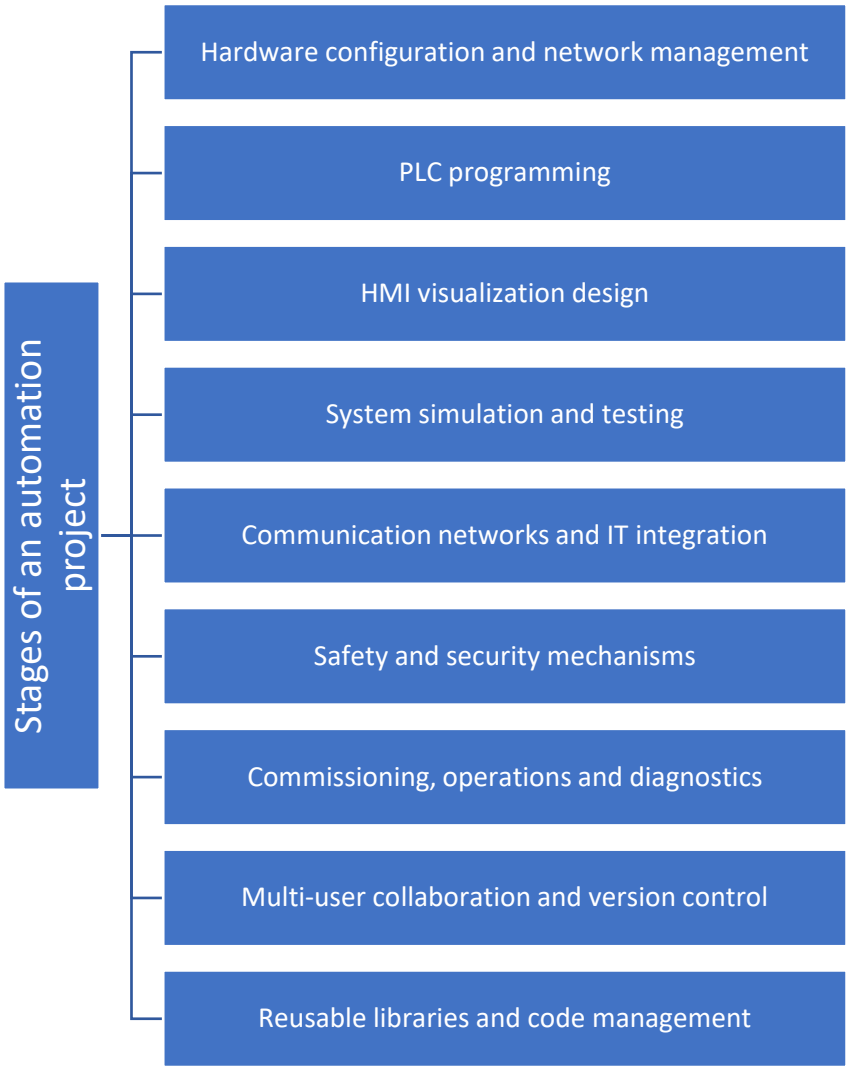
Table of Contents

Chapter 1 - Overview

TIA Portal is Siemens' integrated software environment for managing and programming SIMATIC automation systems across the entire lifecycle of a manufacturing or process industry project.

TIA stands for "Totally Integrated Automation" representing Siemens' vision for complete end-to-end integration of all design, operations and maintenance tasks within one flexible software suite. This aims to optimize efficiency and reduce total cost compared to using separate fragmented tools.

TIA Portal consolidates the following key stages of an automation project into a unified engineering workflow:



It is an integrated product tailored specifically for users of Siemens SIMATIC automation portfolio including controllers, I/O, drives, panels and software. TIA Portal aims to deliver significant engineering efficiency and data consistency advantages.

Key Capabilities and Features

Hardware Configuration and Management

A core capability of TIA Portal is centralized configuration of automation hardware across the Siemens SIMATIC family. It automatically detects PLCs, HMIs, drives, and distributed I/O on PROFINET or PROFIBUS networks.

Engineers can browse the network topology and assign parameters and logical addresses to all components. Diagnostics features monitor online status to quickly isolate faults down to the cable or module level.

TIA Portal simplifies staging, installation, replacement and modification of hardware devices. Large automation systems can be organized hierarchically for manageability across production lines, zones or geographic areas.

PLC Programming Environment

The TIA Portal programming editor streamlines development of control logic to execute on SIMATIC S7 PLCs and WinAC soft PLCs. All IEC 61131-3 languages are supported including Ladder Logic (LAD), Function Block Diagram (FBD), Structured Text (ST), Sequential Function Chart (SFC) and Instruction List (IL). Code blocks can be structured across multiple layers with nested program organization units. Languages can be combined flexibly within a project through segment encapsulation.

Features to boost programming productivity include:

- Reusable libraries of standardized function blocks, equipment phases, and data types
- Import/export of elements to integrate legacy or third-party code
- Symbolic addressing with user-defined identifiers
- Multi-user editing with checkout/check-in mechanisms
- Offline simulation for virtual testing without hardware
- Graphical and textual program comparison
- Search, cross-reference and consistency check tools
- Aliasing of variable data types
- 16-bit integer data types for advanced math

TIA Portal simplifies end-to-end testing through automated download of hardware configurations, PLC programs and HMI visualization to physical devices. Program changes can be downloaded with the machine running for greater uptime.

HMI and Visualization Design

The integrated HMI editor streamlines creation of graphical operator interface panels and visualizations that connect to Siemens controllers. Templates and symbol libraries accelerate building new displays with a consistent look and feel.

Unique HMI design features include support for multi-touch gestures, high-resolution graphics, videos, animations, and adaptive screen layouts that resize responsively across devices. Alarms, trends and data logs allow monitoring key performance indicators. Audit trails track operator actions for regulatory compliance.

TIA Portal enables simulations to test HMI performance prior to physical deployment. HMI screens can be linked dynamically to controller tags for real-time data exchange without external SCADA systems. Faceplates and templates re-apply standardized HMI elements.

Integrated Networking and Communications

With TIA Portal, users can easily configure communication networks and connections between SIMATIC controllers, HMI panels, drives, and devices from other vendors. Automatic network scanning simplifies creating large automation systems with distributed I/O.

Built-in instructions make it easy to program communication over industrial protocols like PROFINET, PROFIBUS, AS-Interface, and OPC UA. TIA Portal also supports IT standards like TCP/IP for enterprise connectivity and remote access.

Safety and Cybersecurity Features

For functional safety applications, TIA Safety Editor provides dedicated tools for programming fail-safe SIMATIC S7 PLCs and ET200 I/O modules in conformance with IEC 61508 / IEC 62061 standards.

Cybersecurity features include password rules, access control mechanisms, HTTPS communication encryption, and antivirus integration to protect intellectual property. Audit logs monitor user actions and configuration changes.

Multi-User Collaboration

Following a server-client architecture, TIA Portal allows multiple engineers to simultaneously work on a project with safeguards against concurrent editing conflicts. Comprehensive version control tracks changes over time while backup mechanisms prevent loss of data.

Libraries and Code Reusability

User-defined libraries in TIA Portal contain standardized logic blocks, HMI screens, equipment phases, and other reusable elements to encourage programming consistency.

Existing legacy code can also be imported into libraries. Read/write access controls prevent unauthorized changes.

TIA Openness API

For integration with third-party tools, TIA Portal offers open programming interfaces for importing/exporting data in various formats. This allows extending the environment with alternative languages, custom apps, or legacy software systems.

Engineering Workflow

By consolidating the suite of engineering tools into a unified environment, TIA Portal aims to optimize the workflow through an automation project lifecycle:

- Hardware configuration: Lay out network topology of controllers, I/O, drives, HMI panels, and other field devices. Assign parameters and addresses.
- PLC programming: Develop control logic with IEC 61131-3 languages for deterministic system behavior. Utilize libraries of reusable functions.
- HMI development: Create graphical operator interfaces with situational visualization screens, symbols, alarms, and historical trending.
- Simulation and testing: Validate control logic and HMI behavior offline through integrated simulations.
- Communication: Configure industrial protocols like PROFINET and IT standards like TCP/IP for connectivity.
- Safety and security: Apply mechanisms like access control and encryption to protect intellectual property.
- Commissioning: Download configurations to physical hardware, connect wiring, calibrate I/O, and tune PID loops.
- Operations: Supervise automation systems via centralized network topology views and diagnostic tools at runtime.
- Maintenance: Quickly isolate and diagnose network, hardware or software faults using integrated diagnostics.
- Version control: Track changes and merge multi-user project edits through version management tools and libraries.

By integrating these previously disparate workflows into a unified environment, TIA Portal aims to reduce manual overhead, risks, and inconsistencies compared to heterogeneous toolchains.

Benefits of TIA Portal Approach

Proponents highlight numerous benefits of the TIA Portal approach compared to a collection of separate engineering tools:

- Reduced engineering time and costs through an integrated toolchain and automation
- Easier network configuration through automatic topology detection
- Faster commissioning and plant modifications enabled by bulk download of hardware configurations
- Improved productivity through unified data models and consistent interfaces
- Enhanced visibility into plant status with integrated diagnostic tools
- Lower training costs due to a common development environment
- Decreased risk of errors or data inconsistency across tools
- Centralized data management for version control, backup and collaboration
- Reusable standardized libraries encourage programming best practices
- Scalability to large automation projects with thousands of hardware devices
- Tighter integration between SIMATIC components without external interfaces

By following rigorous testing methodologies, Siemens aims to deliver high reliability and quality standards for TIA Portal. Frequent updates incorporate customer feedback into ongoing improvement.

However, an integrated toolchain also poses some challenges:

- Hardware ecosystem limited to Siemens SIMATIC products only
- Upgrades may be less flexible compared to composition of best-of-breed point tools
- Large bundled software package can seem heavyweight
- IT infrastructure may need adaptation to new system requirements

Overall, TIA Portal delivers significant engineering efficiency advantages for users committed to the Siemens automation ecosystem. It is best suited for larger enterprises pursuing integrated automation with centralized governance strategies.

TIA Portal Editions

TIA Portal is available in multiple editions scaled for different user needs:

- TIA Portal Basic: For small-scale automation with Basic Panels and small S7-1200 PLCs. Limits hardware configuration sizes.

- TIA Portal Comfort: Supports medium-scale automation with more advanced PLCs, panels, and drives. Expanded library functionality.

- TIA Portal Engineering: Comprehensive engineering edition without hardware limits. Includes S7-1500 controller support.

- TIA Portal Teamcenter Gateway: Adds integrations for Siemens PLM and Teamcenter product lifecycle management tools.

Specific options like TIA Safety for functional safety and TIA Portal Cloud Connector for cloud-based services can be added to editions.

Hardware and Software Requirements

As an integrated engineering framework, TIA Portal places higher demands on client hardware and software infrastructure compared to standalone SCADA tools.

Minimum system requirements for smooth performance include:

- Windows 10 64-bit operating system
- Intel Core i5 or i7 processor
- 16 GB RAM memory
- DirectX 11 graphics adapter
- High-speed SSD storage
- Multiple CPU cores for parallel build

For managing large, complex projects, additional RAM and SSD capacity is recommended. TIA Portal Server runs on Windows Server OS. Databases utilize Microsoft SQL Server.

Licensing and Pricing

TIA Portal follows a proprietary per-seat licensing model tied to the SIMATIC hardware inventory. License costs depend on the edition and options purchased. Volume discounts are available.

Subscription-based options provide flexible short-term licensing. Hardware configuration limits in lower editions incentivize upgrading to higher licenses as automation projects grow.

In summary, TIA Portal delivers an integrated, end-to-end engineering environment for automating with Siemens SIMATIC technology. It aims to maximize engineering efficiency through workflow integration while ensuring consistent data models across design, operations and maintenance.

For Siemens-centric organizations pursuing centralized governance of automation, TIA Portal provides advantages of tighter integration between components, consolidated network management, enhanced standardization, and greater visibility across the lifecycle.

Training and Support Resources

To supplement this guide, Siemens offers extensive training material and resources for maximizing benefits from TIA Portal:

- SITRAIN online and instructor-led training courses
- Series of Getting Started tutorials
- TIA Selection Tool for product selection
- Online forum to connect with other TIA Portal users
- Local account managers and technical support
- TIA Portal help documentation

Engineers should invest time into formal training on TIA Portal to gain proficiency. Hands-on practice with demo projects is also highly valuable for new users. Ongoing learning resources help unlock the platform's full capabilities.

Chapter 2 - Key steps in using TIA Portal

Siemens TIA Portal is an engineering software that allows programmers to develop, program and commission automation projects. It integrates six different software into one common platform, providing a seamless workflow for automation tasks. The key features of TIA Portal include:

- Step 7 Professional for programming PLCs
- WinCC for creating visualizations
- Startdrive for commissioning drives
- SINAMICS Startdrive for controlling drive systems
- SIMATIC Energy Suite for energy management
- SIMATIC Process Device Manager for configuring process instrumentation

This chapter will walk through the key steps in using TIA Portal to develop and program a basic automation project from start to finish. It covers:

- Creating a new project
- Adding and configuring hardware
- Developing a PLC program
- Designing an HMI visualization
- Simulating and testing the program
- Downloading the project to a PLC

Prerequisites

Before starting with TIA Portal, some prerequisites need to be met:

- A Windows-based computer with TIA Portal software installed. The hardware requirements depend on the version of TIA Portal but generally require a relatively modern PC with sufficient RAM and graphics capabilities.
- Access to Siemens automation hardware, such as a SIMATIC S7 PLC and HMI panel. A wide range of PLCs are supported including S7-1200, S7-1500, S7-300, and S7-400 models. For HMI panels, Basic, Comfort, Mobile, and Runtime panels can be used.
- Basic knowledge of PLC programming and industrial automation. The TIA Portal user should understand ladder logic diagrams, data types, tag-based memory addressing, etc. Some training or learning resources may be useful for new users.
- Licenses for any software packages that will be used, such as Step 7 Professional and WinCC Runtime. The licenses allow activating the software tools.
- Documentation or specifications for the particular automation application being developed, including I/O lists, process diagrams, and control narratives. These provide requirements for designing the system.

Having these prerequisites in place helps ensure an efficient and successful start to a new TIA Portal project. The required hardware, software, knowledge, and documentation provide a foundation for productive engineering workflows.

Creating a New Project

When first opening TIA Portal, the Project View portal screen is displayed. This is the starting point for managing projects in the software suite. Here is how to create a new project:

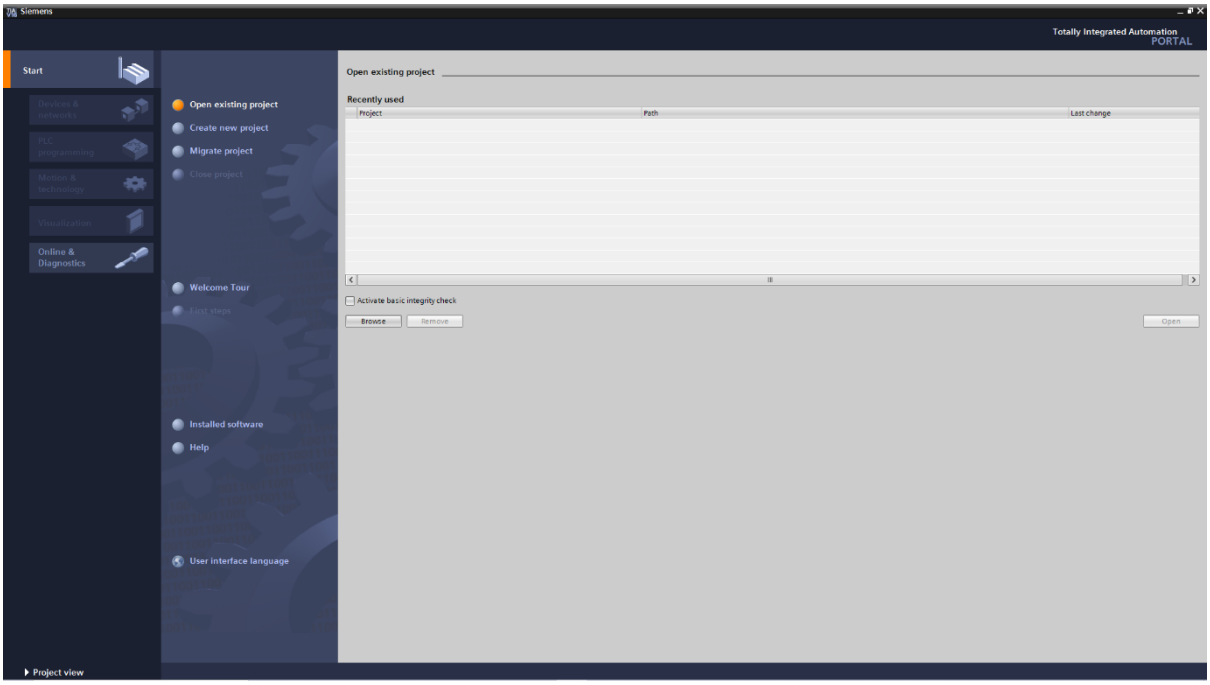


Figure 1 Starting a new Project window in TIA Portal

1. Click "Create new project" on the Start page. This opens a project creation wizard.
2. Select the type of project. Options include Basic Panel, Comfort Panel, RT Advanced, etc. For this guide, select "Basic Panel" for a basic HMI panel project.
3. Enter a project name and location to save project files. Using a descriptive name and organized folder structure is recommended.
4. Select the PLC type for the controller. For this guide, we'll use a S7-1500 PLC.
5. Review the summary of configured devices and settings.
6. Click Create to generate new the TIA Portal project.

The new project is created and the Devices & Networks portal view opens. Additional devices can now be inserted and configured.

The Devices & Networks view is where you configure all the hardware components in your project, including PLCs, HMI panels, drives, and distributed I/O. A network or communication topology can also be defined here.

Adding a PLC and Hardware Configuration

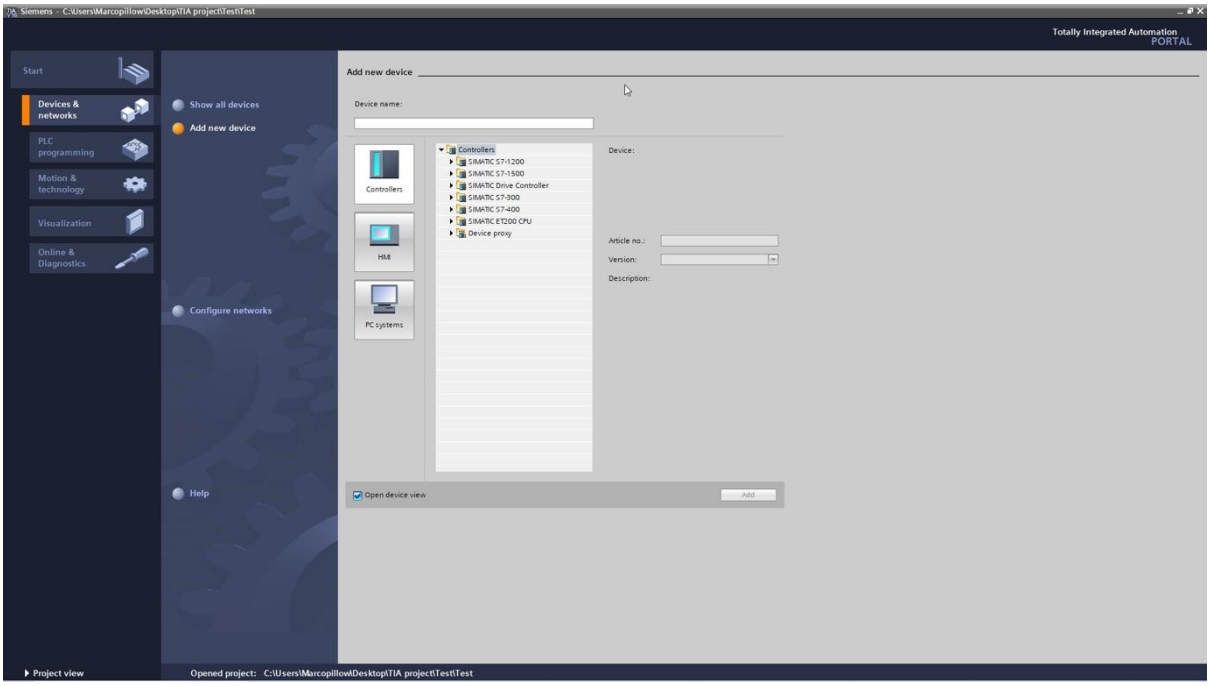


Figure 2 Adding new device window

To add our S7-1500 PLC:

1. In Devices & Networks, right-click the Controller node and select "Add new device." This opens the hardware catalog.
2. Navigate to the SIMATIC S7 family and select the desired CPU model, such as CPU 1514C.
3. Change the PLC name if desired. For example, PLC_1.
4. Review the module details and click OK to add the PLC.

The PLC now appears in the project tree under Controller. We can now start programming the PLC logic.

Developing the PLC Program

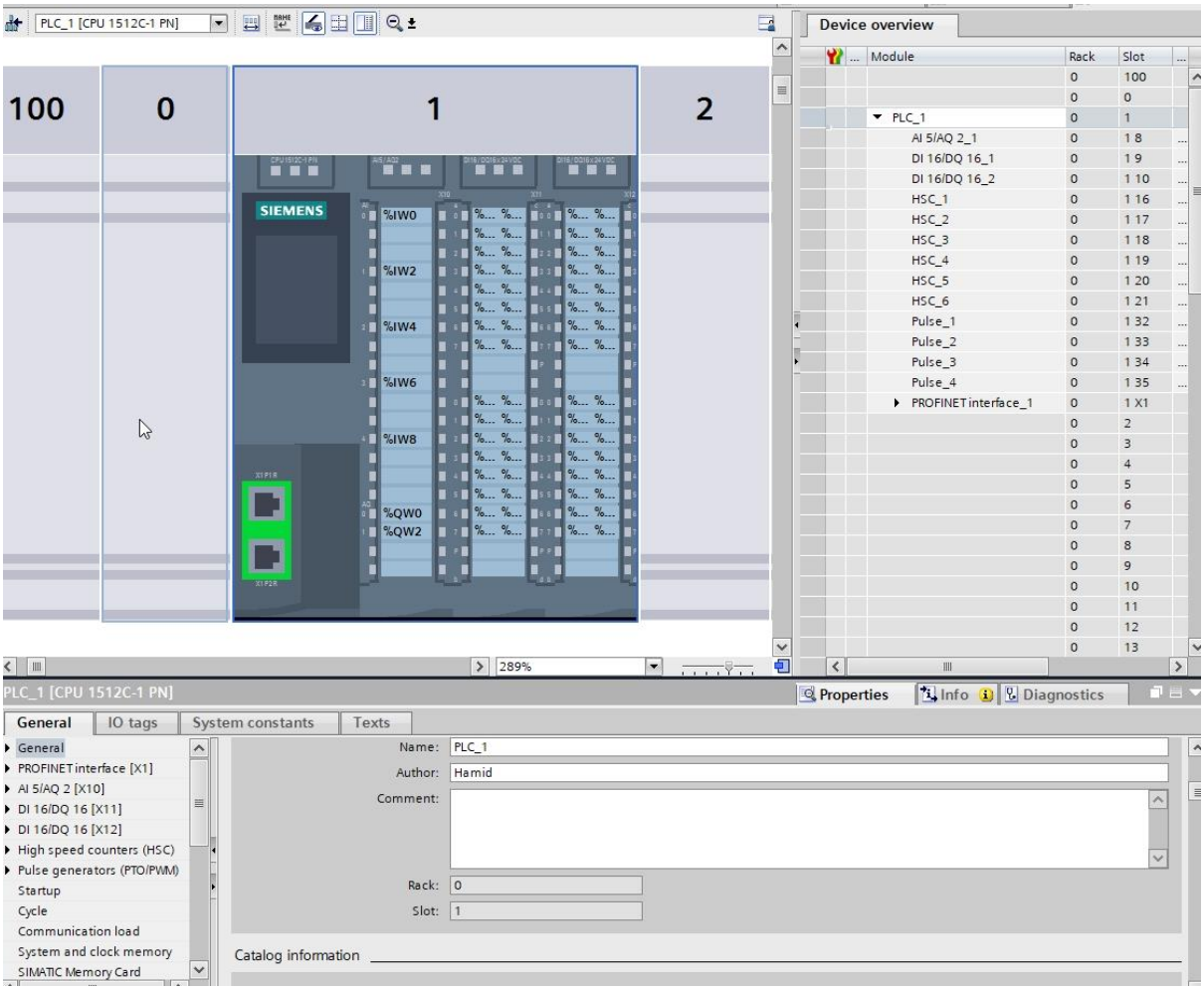


Figure 3 Project View Environment in TIA Portal Software.

TIA Portal includes Step 7 Professional for programming Siemens PLCs with either Ladder Logic (LAD) or Statement List (STL) languages. To start programming:

1. Double-click PLC_1 to open the PLC in the project tree.
2. Right-click Program blocks and select Add new block.
3. Create an Organization Block (OB) to contain the main program logic.
4. Name the OB "Main" and select OB1 for the type.

The PLC programming editor view opens, where you develop the control logic.

For this example, we'll program a simple light control logic in LAD:

1. Double-click the Main OB to open it.
2. From the instructions window on the right, drag and drop two contacts and a coil into the network.
3. Click each contact instruction and configure the address to "Btn_On" and "Btn_Off" buttons.
4. Configure the coil address to "Light_1" output.
5. Right-click and select "Insert network" to add a new rung.

6. Repeat steps 2-4 to add logic for Light_2.

This creates a basic program with two lights controlled by On and Off buttons. More complex logic with timers, counters, math blocks, etc. can be added. Tag names for I/O should follow a systematic naming convention.

The program can now be simulated and tested before downloading to the PLC. Proper syntax and troubleshooting tools help identify errors before runtime.

I/O configuration and addressing for the Siemens S7-1500 PLC in TIA Portal

The S7-1500 PLC works with both digital and analog signals through its onboard or expansion I/O modules. TIA Portal provides tools to configure the I/O and assign meaningful addresses.

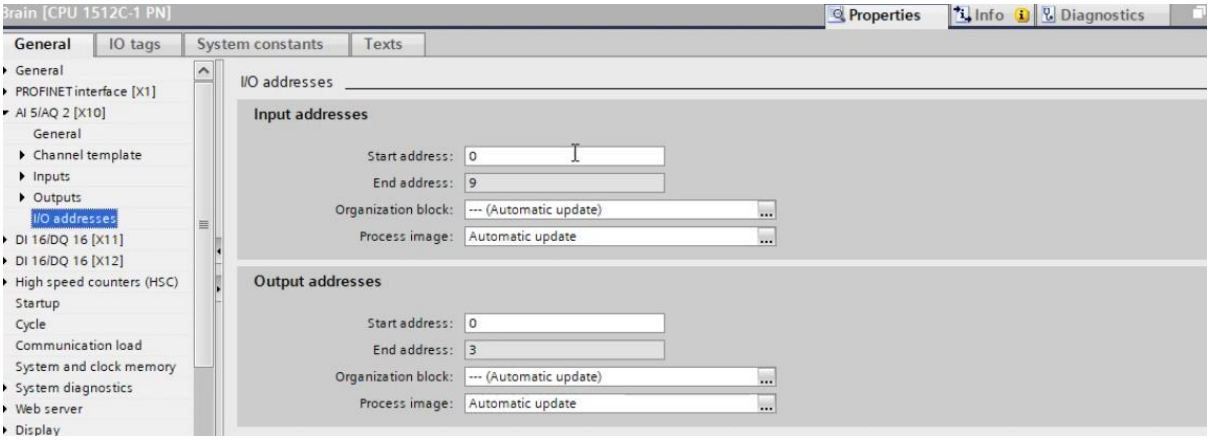


Figure 4 I/O configuration and addressing for the Siemens S7-1500 PLC

Configuring I/O Modules:

- In the Devices & Networks view, double-click the PLC name to view I/O modules
- Select a slot and click "Insert Module" to add an I/O module
- Choose the module type from the catalog (e.g. DI16, DQ16, AI8, AQ4, etc.)
- Set module properties like name, address ranges, and channel parameters

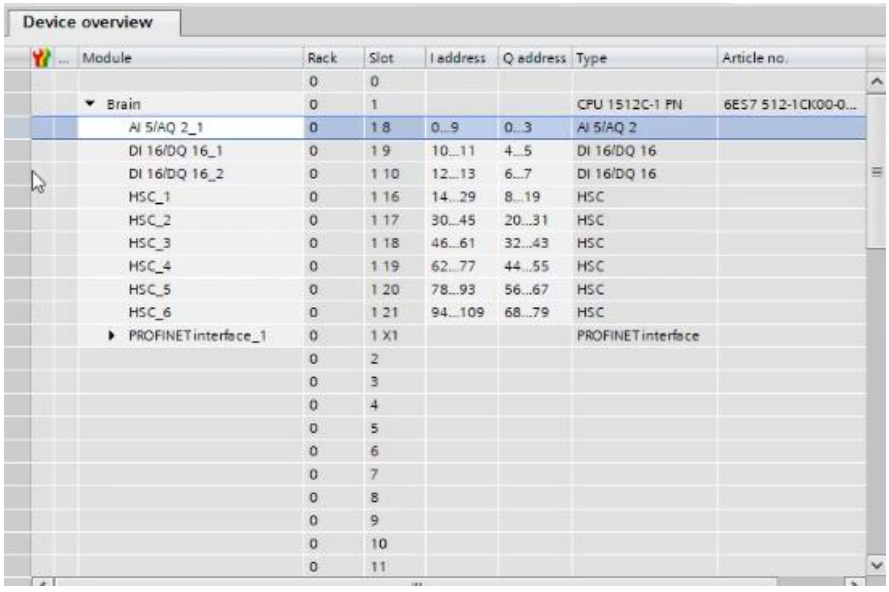


Figure 5 Device Overview of a selected PLC in TIA Portal

PLC tags in TIA Portal

TIA Portal is Siemens' software for programming and configuring Siemens PLCs. In TIA Portal, tags are defined in the PLC tag table. Here are some key things to know about tags in TIA Portal:

- Name - Tag identifier
- Data Type - BOOL, INT, REAL, etc.
- Scope - Local, Global, External
- Usage - I/O, Memory, Alias, System

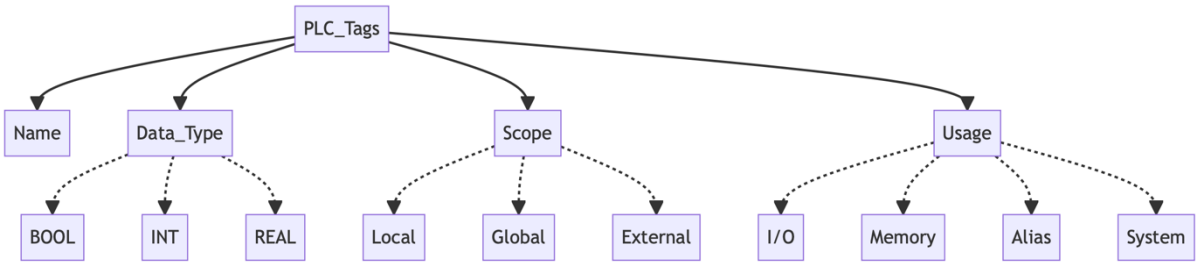


Figure 6 Tag Attributes in TIA Portal

Tags have a name, data type, scope, comment, and other properties. Common data types are BOOL, INT, REAL, etc.

Tag scope determines which parts of the PLC program can access the tag. Scope options are Local (within one POU), Global (controller-wide), and External (accessible outside the controller).

Tags can be tied to physical I/O points, represent internal memory locations, act as alias tags, or have no linkage at all.

Special system tags exist for things like the current scan cycle time or firmware version.

To create a new tag in TIA Portal:

1. Double click PLC Tags in the project tree to open the PLC Tag Table
2. Click the create button
3. Enter the tag Name
4. Select the Data Type from the drop-down list
5. Set any other desired properties like Address, Scope, Comment, etc.

To edit an existing tag, double-click the tag in the PLC Tag Table. This will open the tag properties window where changes can be made.

Tag Characteristics in TIA Portal

- Scope: TIA Portal tags exhibit three scopes:
    - Local: Accessible only within the defining program block.
    - Global: Accessible across the entire PLC program.

    - External: Accessible from other PLC devices or a Human-Machine Interface (HMI).
- Usage: TIA Portal tags serve various purposes, including:
    - I/O: Facilitating data interaction with input and output devices.
    - Memory: Storing data within the PLC program.
    - Alias: Establishing symbolic names for other tags.
    - System: Accessing predefined tags, such as runtime and memory status.

Additional Tag Information:

- Tags may encompass arrays or structs.
- Some tags retain data even during PLC power loss.
- Tags can be annotated with comments for enhanced comprehension and maintenance.
- Tags can be declared within data blocks, function blocks, and function calls.

Data types for Siemens S7-1500 PLC

Data types define the format and behavior of tag values used in a PLC program. Choosing appropriate data types helps utilize PLC memory efficiently.

Standard Data Types

These fundamental PLC data types are available:

- Boolean (BOOL) - Single bit value, 0 or 1
- Integer (INT) - 16-bit whole number
- Double Integer (DINT) - 32-bit whole number
- Real (REAL) - 32-bit floating point number
- String (STRING) - Text string value

Additional integer and real types provide larger sizes. Duration types represent time spans.

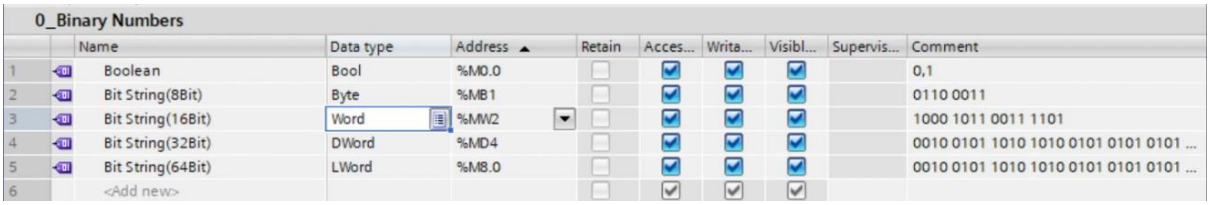


Figure 7 Binary Number Data types

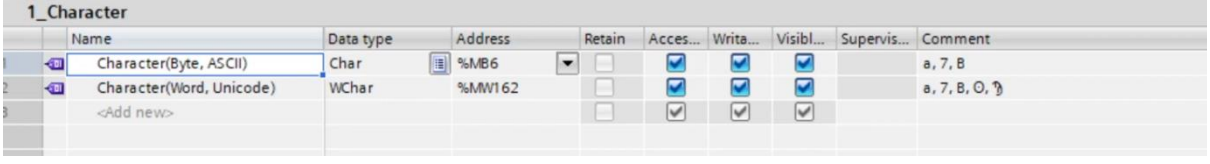


Figure 8 Character Data types

1_Integers

	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	Supervis...	Comment
1	Integer(Word)	Int	%IW54		☑	☑	☑		13501, -23502
2	Integer(Double)	DInt	%ID8		☑	☑	☑		-1648551255, 341
3	Integer(Long)	LInt	%I64.0		☑	☑	☑		-465465126, 6598541549879841651
4	Integer(Byte/Short)	SInt	%IB136		☑	☑	☑		-31, 120
5	Integer(Unsigned Double)	UDInt	%ID146		☑	☑	☑		25, 45461
6	Integer(Unsigned Long)	ULInt	%I152.0		☑	☑	☑		49584984651654198 4798, 498798456...
7	Integer(Unsigned Short)	USInt	%IB160		☑	☑	☑		250, 15
8	Integer(Unsigned Word)	UInt	%IW150		☑	☑	☑		164,4544
9	<Add new>				☑	☑	☑		

Figure 9 Integer Data types

1_Time

	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	Supervis...	Comment
1	Time(Double in ms)	Time	%ID138		☑	☑	☑		T#3m_45_947MS, 2500
2	Time(Long in ms)	LTime	%I80.0		☑	☑	☑		LT#32NS, 7006
3	Time(Word in S5)	S5Time	%IW134		☑	☑	☑		0011 1001 1000 0010
4	<Add new>				☑	☑	☑		

Figure 10 Time Data types

2_Date and Time

	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	Supervis...	Comment
1	Date and Time(Long hours:min...	LTime_Of_D...	%I88.0		☑	☑	☑		LTOD#10:20:30.400_365_215
2	Date and Time(Long year-mont...	LDT	%I56.0		☑	☑	☑		LDT#2020-01-28-08:12:34.567
3	Date and Time(Double hours:...	Time_Of_Day	%ID142		☑	☑	☑		TOD#10:20:30.400
4	Date and Time(Word Year,Mon...	Date	%IW0		☑	☑	☑		D#2020-01-28
5	<Add new>				☑	☑	☑		

Figure 11 Date and Time Data types

2_Parameter

	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	Supervis...	Comment
1	S5 Timer	Timer	%T0		☑	☑	☑		T3
2	S5 Counter	Counter	%C0		☑	☑	☑		C7

Figure 12 Parameters Data types

3_Hardware

	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	Supervis...	C
1	Any Hardware	Hw_Any	%IW28		☑	☑	☑		
2	DP Slave/PROFINET IO device	Hw_Device	%IW30		☑	☑	☑		
3	DP Master	Hw_DpMaster	%IW32		☑	☑	☑		
4	DP Slave	Hw_DpSlave	%IW34		☑	☑	☑		
5	High Speed Counter	Hw_Hsc	%IW36		☑	☑	☑		
6	PROFINET (IO) Port	Hw_IEPort	%IW38		☑	☑	☑		
7	Interface	Hw_Interface	%IW40		☑	☑	☑		
8	ID Number of the CPU/Interface	Hw_Io	%IW42		☑	☑	☑		
9	PN (IO) system or DP Master	Hw_IoSystem	%IW44		☑	☑	☑		
10	Module	Hw_Module	%IW46		☑	☑	☑		
11	Pulse Encoder	Hw_Pto	%IW48		☑	☑	☑		
12	Pulse Width Modification	Hw_Pwm	%IW50		☑	☑	☑		
13	Central Hardware Component	Hw_SubModule	%IW52		☑	☑	☑		
14	Operating Hours Counter	Rtm	%IW132		☑	☑	☑		
15	<Add new>				☑	☑	☑		

Figure 13 Hardware Date types

4_Connection

	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	Supervis...
	Specific Connection	Conn_Any	%MW8	☐	☑	☑	☑	
	Open Communication over PN	Conn_Ouc	%MW10	☐	☑	☑	☑	
	Open Communication over UDP	Conn_Prg	%MW12	☐	☑	☑	☑	
	S7 Communication Block	Conn_R_Id	%MD14	☐	☑	☑	☑	
	Communication Port	Port	%IW126	☐	☑	☑	☑	
	<Add new>			☐	☑	☑	☑	

Figure 14 Connection Data types

4_Data Block

Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	Supervis...
ID of a DB (Name or Number)	DB_ANY	%IW2	☐	☑	☑	☑	
DB Number (Generated by Prog...	DB_DYN	%IW4	☐	☑	☑	☑	
DB Number (Generated for We...	DB_WWW	%IW6	☐	☑	☑	☑	
<Add new>			☐	☑	☑	☑	

Figure 15 Data Block Data types

4_Event

Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	Supervis...
Event ID	Event_Any	%ID16	☐	☑	☑	☑	
Event ID (dynamic for OB)	Event_Att	%ID20	☐	☑	☑	☑	
Event (Hardware Interrupt)	Event_HwInt	%ID24	☐	☑	☑	☑	
Event Identifier	Aom_Ident	%MD0	☐	☑	☑	☑	
<Add new>			☐	☑	☑	☑	

Figure 16 Event Data types

4_Organisation Block

	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	Supervis...
	General OB	OB_Any	%IW104	☐	☑	☑	☑	
	Dynamic OB	OB_Att	%IW106	☐	☑	☑	☑	
	Cyclic OB	OB_Cyclic	%IW108	☐	☑	☑	☑	
	Time-Delay OB	OB_Delay	%IW110	☐	☑	☑	☑	
	Diagnostic OB	OB_DIAG	%IW112	☐	☑	☑	☑	
	Hardware Interrupt OB	OB_HWINT	%IW114	☐	☑	☑	☑	
	Program Cycle OB	OB_PCYCLE	%IW116	☐	☑	☑	☑	
	Startup OB	OB_STARTUP	%IW118	☐	☑	☑	☑	
	Time Error OB	OB_TIMEERROR	%IW120	☐	☑	☑	☑	
	Time of Day OB	OB_Tod	%IW122	☐	☑	☑	☑	
	Connect Synchronous Cycle OB	Pip	%IW124	☐	☑	☑	☑	
	<Add new>			☐	☑	☑	☑	

Figure 17 Organization Block Data Types

Derived Data Types

These data types are based on the standard types:

- Array - Collection of elements with the same data type
- Structure - Collection of different data type members
- Enumeration - User-defined data type restricted to a set of values

Derived data types, such as arrays, structures (also known as user-defined data types or UDTs), and enumerations, are essential components of many programming languages, including those used in programmable logic controllers (PLCs). These data types provide a way to organize and manage data more effectively and make your PLC programs more structured and easier to understand.

Here's an overview of each of these derived data types:

1. **Arrays:**
   - **Definition**: An array is a collection of elements, all of which have the same data type.
   - **Use**: Arrays are useful when you need to work with multiple pieces of data of the same type. You can access individual elements within the array using an index (position), which starts from 0 in most programming languages.
   - Example: In a PLC, you might use an array to store temperature readings from multiple sensors.
2. **Structures (User-Defined Data Types - UDTs):**
   - **Definition:** A structure, also known as a user-defined data type (UDT), is a collection of different data type members bundled together under a single name.
   - **Use:** Structures are handy for organizing related data that belongs together. You can create custom data structures to represent complex objects or machines with multiple attributes.
   - **Example:** In a PLC, you could create a UDT called "Motor" with members like "Speed," "Status," and "Temperature."
3. **Enumerations:**
   - **Definition:** An enumeration, or enum, is a user-defined data type that restricts a variable to a specific set of values, which you define.
   - **Use:** Enums are useful when you want to work with variables that have distinct and predefined states or options. They make your code more readable and help prevent invalid values.
   - **Example:** In a PLC program, you might define an enumeration called "DayOfWeek" with values like "Monday," "Tuesday," "Wednesday," etc.

Custom Data Types

Users can create custom data types by aliasing elementary or derived data types.

For choosing the Data Types, we should consider the type of value, required size, nature of operation, and efficiency when selecting data types in TIA Portal and also we must use the smallest size needed to conserve PLC memory.

The Siemens S7-1500 series PLCs support a variety of data types that are commonly used in industrial automation and control applications. Here's a list of some of the main data types available for the Siemens S7-1500 PLC:

Table 1 Main Data types in TIA Portal

#	**Data Type**	**Description**
1	BOOL	Single bit value, 0 or 1
2	BYTE	8-bit unsigned integer
3	WORD	16-bit unsigned integer
4	DWORD	32-bit unsigned integer
5	INT	16-bit signed integer
6	DINT	32-bit signed integer
7	REAL	32-bit single precision floating point
8	LREAL	64-bit double precision floating point
9	STRING	String of characters
10	TIME	Time value in milliseconds
11	TOD	Time value with date information
12	DATE	Date value
13	COUNTER	Used for counting events
14	TIMER	Used for timing operations
15	S5TIME	S5 format time value
16	S5DATE	S5 format date value
17	S5TOD	S5 format date and time
18	UDT	User-defined data type
19	ENUM	User-defined named values
20	ARRAY	Collection of elements
21	FB	Encapsulates logic and data
22	FC	Encapsulates logic
23	ST	Textual programming language
24	LD	Graphical programming language
25	FBD	Graphical programming language
26	IL	Textual programming language
27	CFC	Graphical programming language
28	ALIAS	Reference to another data type

These data types provide the foundation for creating control programs in Siemens S7-1500 PLCs. Depending on the specific PLC model and firmware version, there may be additional data types and variations available. It's essential to refer to the PLC's documentation and programming environment for the most up-to-date information on data types and their usage.

Using Tags in PLC Programs

In LAD, FBD, and STL programming languages, tags are accessed by their name:

- In LAD/FBD, tags are used in box instructions and wiring
- In STL, tags appear in variable declarations and assignments

Tags can also be monitored and forced to test values in the Tag Monitor window in TIA Portal.

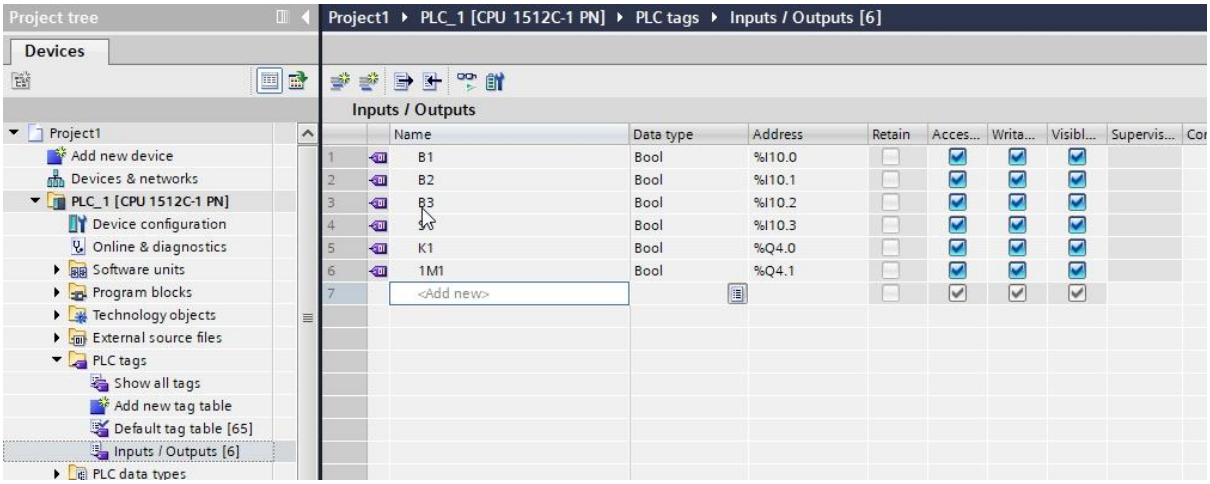


Figure 18 PLC tags (Inputs / Outputs)

Figure 19 Tags and its name are visible on the PLC in TIA portal

Introduction to PLC Programming in TIA Portal

PLCs are programmed to control industrial processes and machines. PLC programming involves developing logic to read inputs, control outputs, monitor variables, communicate with external devices, and perform other control tasks.

TIA Portal includes five IEC 61131-3 standard PLC programming languages:

- Ladder Logic (LAD) - Graphical language that represents logic as rungs of relay contacts and coils.
- Function Block Diagram (FBD) - Graphical language that represents logic through interconnected function blocks.
- Structured Text (STL) - Textual language similar to Pascal with statements and functions.
- Sequential Function Chart (SFC) - Graphical language that represents program flow as step sequences and transitions.
- Continuous Function Chart (CFC) - Graphical language used for analog control systems.

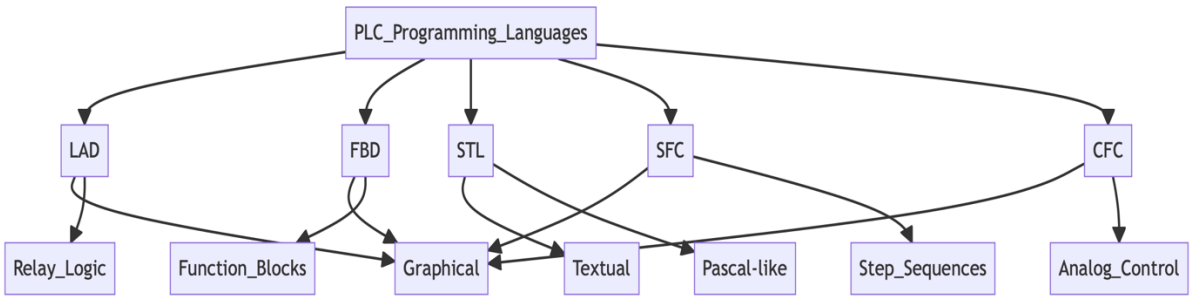


Figure 20 PLC Programming Languages

Creating a PLC Program

The main methods of PLC programming:

- Writing code in editors - For textual languages like STL, write code in the editor window.
- Dragging and linking - Graphical languages use drag-and-drop elements that are linked together.
- Importing function blocks - Add existing FB modules from libraries.
- Tag and variable access - Reference tags and declare variables for code to use.
- Monitoring and testing - Use watch tables and force values to test and debug.

Compiling and Downloading the Program

1. Compile the program to check for errors.

2. Download the compiled program to the connected PLC.
3. Switch the PLC to Run mode to execute the program.

Simulation and Testing

TIA Portal provides simulation capabilities to test and debug PLC programs without requiring actual hardware. Simulation is useful for:

- Validating program logic before downloading to PLC
- Training and learning PLC programming techniques
- Prototyping machine/process behavior
- Reproducing issues and testing fixes

Types of Simulation

In Siemens TIA Portal, the software offers three primary simulation modes, each serving distinct purposes:

1. **PLC Program Simulation:** This mode provides the capability to emulate the behavior of a PLC program independently of any physical hardware connections. Within this virtual environment, users can assess the program's logical functionality, monitor variables, and observe the program's execution.
2. **Hardware Simulation:** Within this mode, users can replicate the interplay between the PLC program and a virtual representation of the physical hardware configuration. This functionality allows for the evaluation of the program's interaction with physical inputs and outputs in a simulated environment, obviating the requirement for actual hardware components.
3. **Online/Offline Simulation:** This mode facilitates a seamless transition between online and offline operational states. The online mode permits users to establish a connection with tangible PLC hardware, thereby enabling real-time testing, debugging, and validation of the program's functionality.

Table 2 key differences between the three simulation modes

Mode	Description
PLC Program Simulation (PLCSIM)	Simulates the behavior of the PLC program without any physical hardware connected.
Hardware Simulation	Simulates the behavior of the PLC program and its interaction with physical hardware.
Online/Offline Simulation	Allows you to seamlessly switch between online and offline operational states.

Setting Up Simulation

To configure simulation:

1. In the Device Configuration, add a simulated device instead of real PLC hardware.
2. In the project tree, select "Set <device name> to Stop" to enable editing.
3. In program editor, open variable tables and force values on tags.
4. Download project to simulated PLC.

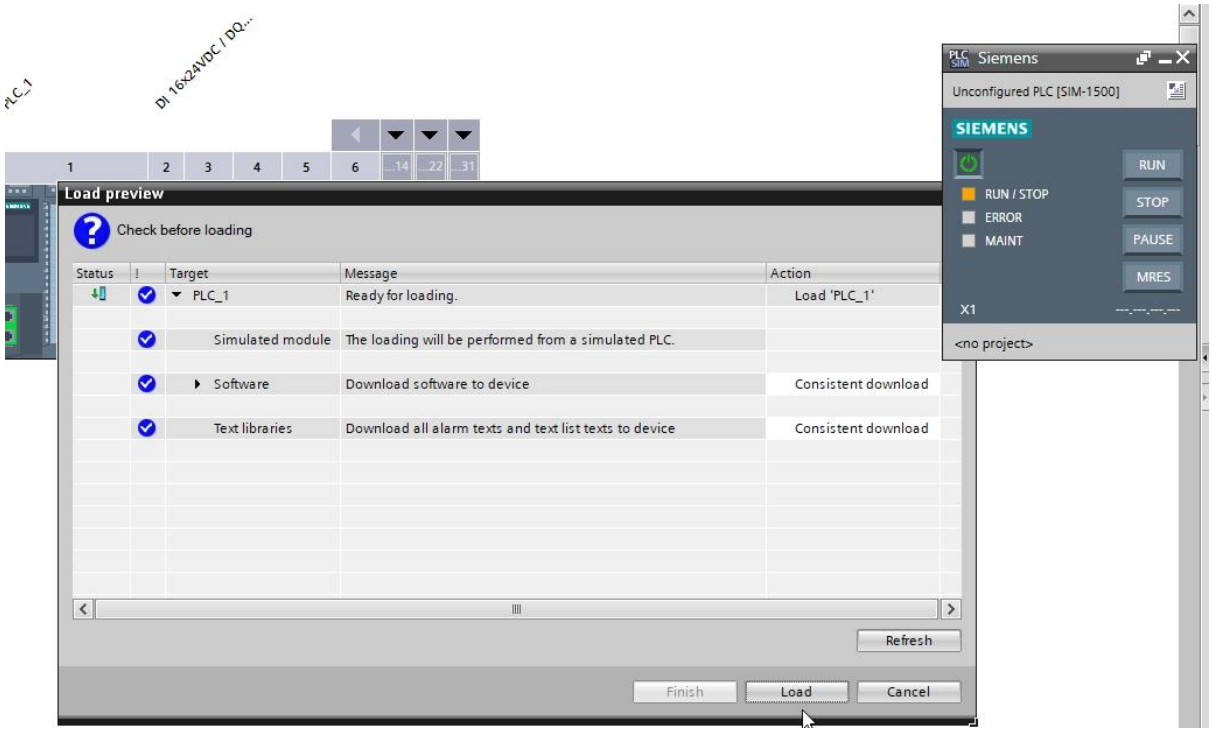


Figure 21 Load preview window for starting the simulation

Running the Simulation

1. Set the simulated PLC to "Run" mode.
2. The program will start executing. Monitor tag values and program behavior.
3. Force-toggle inputs or internal tags to simulate process changes.
4. Watch program responses based on the applied simulation test scenario.
5. Stop the PLC to end test. Analyze program execution in the diagnostic trace.

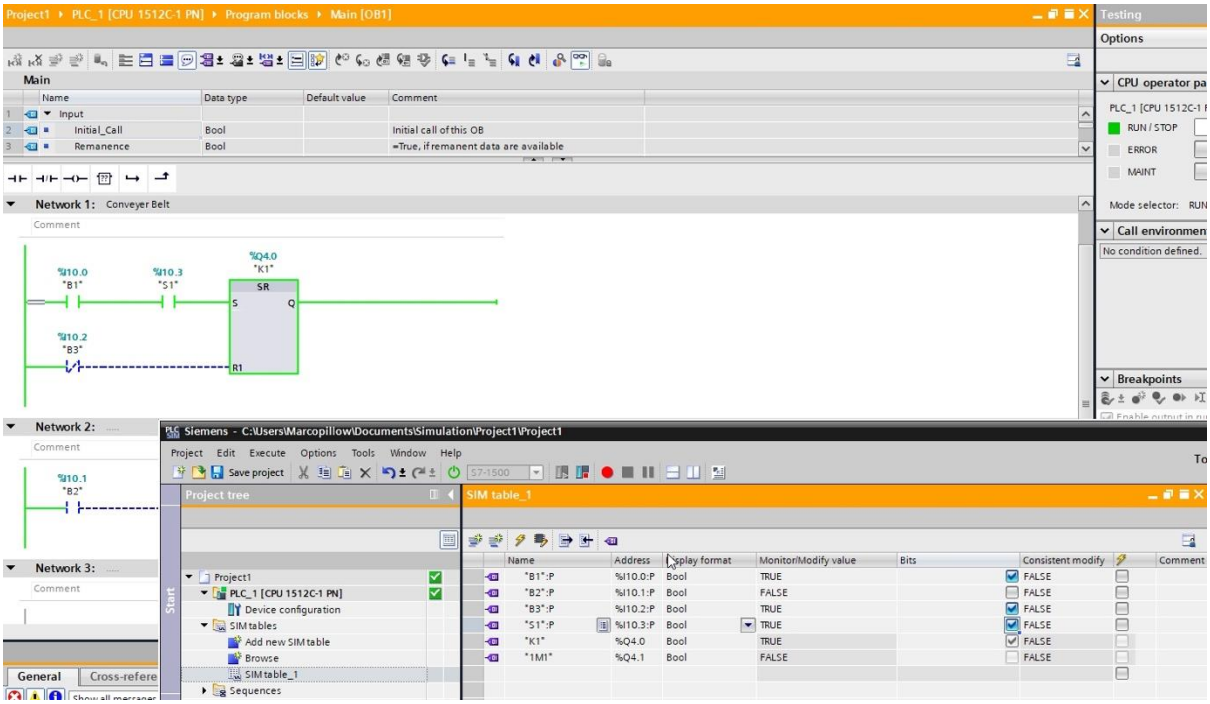


Figure 22 Running the Simulation with PLCSIM in TIA Portal

Simulation allows testing PLC programs without operational risks. Take advantage of TIA Portal simulation tools at all phases of PLC programming.

Chapter 3 – HMI integration in TIA portal

TIA Portal enables robust integrated HMI development by sharing core elements like tags, alarms, and recipes between the PLC and HMI. This helps improve efficiency and reduce errors.

Key aspects of HMI integration include:

Project View - The Devices & Networks portal integrates PLCs, HMI panels, drives, and other devices into one project.

Tag Sharing - Tags defined in the PLC program can be directly accessed by the HMI without separate configuration.

Screens & Display Elements - TIA Portal includes tools to quickly build graphical HMI display screens using libraries of elements like buttons, charts, images etc.

Animations & Events - Screen objects can be linked to PLC tags and animations configured to respond to state changes.

Alarms - Alarm events in the PLC program can be logged and displayed on the HMI. Alarm text, priority, classes, and history can be configured.

Recipes - Recipe data management enabling parameters to be read/written between PLC and HMI.

User Management - Configuring user accounts, credentials, and access permissions at both the PLC and HMI level.

Multi-Language Support - Text libraries allow runtime language switching on the HMI.

Simulation - Virtual testing of HMI graphics and interaction without needing physical hardware.

Adding an HMI Panel

The main HMI panel options available in TIA Portal for Siemens SIMATIC systems are:

- Basic Panels: Entry-level panels for simple HMI applications. Example: KTP400 Basic.
- Comfort Panels: Mid-range panels with better performance and graphics. Example: KP700 Comfort.
- Mobile Panels: Panels designed for mounting on machines/equipment. Example: Mobile Panel 277.
- SIMATIC HMI Siplus panels

To add an HMI panel for visualization and control:

1. In Devices & Networks, right-click HMI connection and select "Add new device."
2. Select the panel model, such as a Basic Panel 2nd Generation.

3. Click OK to insert the default HMI device.
4. Change the HMI name if desired, e.g. HMI_1.

We can now create a graphical interface for our light control logic on this HMI device.

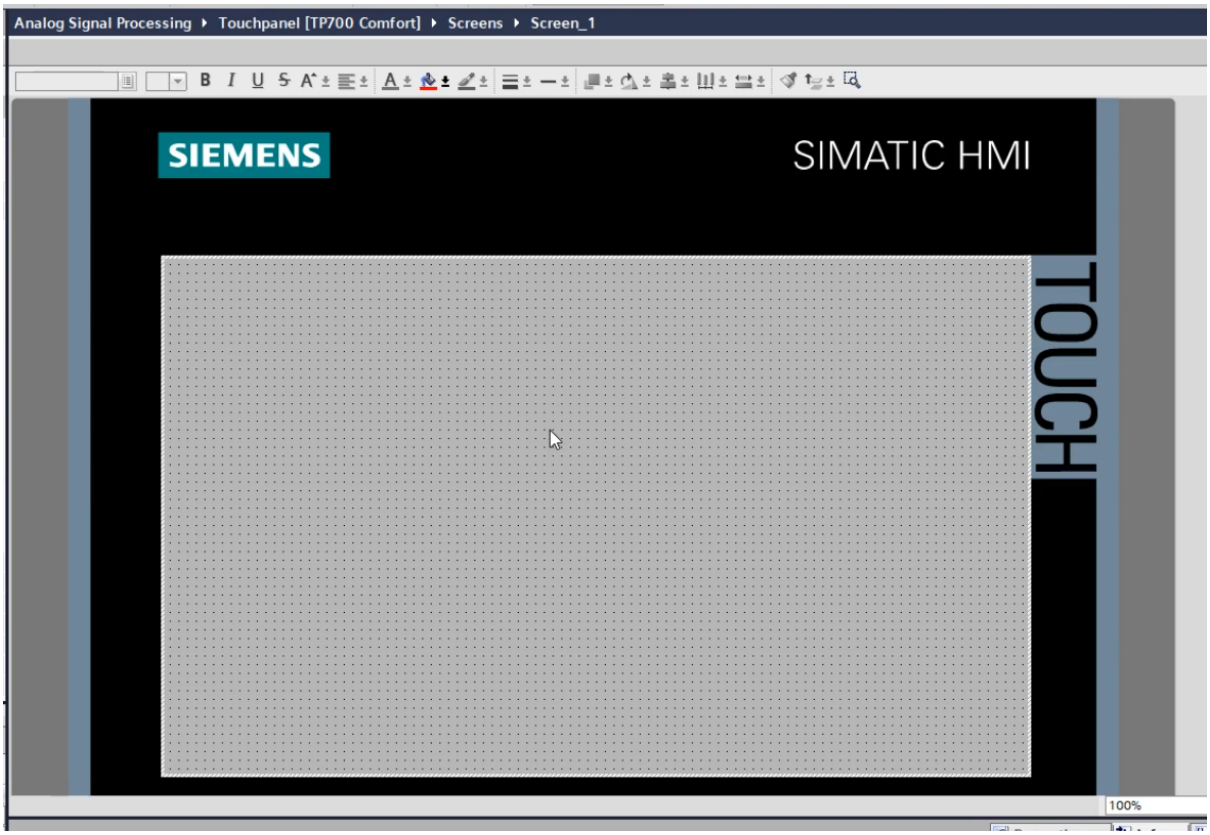


Figure 23 HMI Panel in TIA Portal

Designing the HMI Visualization

TIA Portal uses WinCC to design HMI visualizations with graphical elements linked to PLC tags. To create a basic HMI screen:

1. Double-click HMI_1 to open the device editor.
2. In Screens, right-click and add a new screen. Rename it to something like "Main".
3. From the toolbox, drag two Buttons and two Indicator elements to the screen.
4. Select each element and configure the tag to match the PLC addresses for the buttons and lights.
5. Add static text labels and other design elements.

This creates a basic HMI screen with On and Off buttons to control the lights. More screens can be added for additional interfaces and system views. Advanced controls, symbols, animations, and scripts can be configured for enhanced visualization.

Communication setup between the PLC (controller) and HMI

Here are the basic steps to connect a Siemens HMI to a Siemens PLC in TIA Portal:

1. Create a new TIA Portal project and add both the PLC and HMI devices to the project tree.
2. In the Devices & Networks view, connect the HMI and PLC using a communication link like PROFINET or PROFIBUS.
3. Configure the HMI communication settings to match the PLC network address and protocol.
4. Switch to the HMI screen editor and create a new HMI screen.
5. From the Libraries pane, drag desired objects like buttons, text fields, etc onto the screen.
6. Right-click each object and go to Properties > Events to assign a tag.
7. Insert the PLC tags to be accessed by the HMI screen objects.
8. Go to Connections in the Inspector window and select the PLC tag's data block.
9. Compile and download the project to the HMI and PLC.
10. On the PLC side, the tags are now accessible from the HMI.

The key steps are connecting the devices on the network, assigning matching communication settings, linking HMI objects to PLC tags, and compiling/downloading.

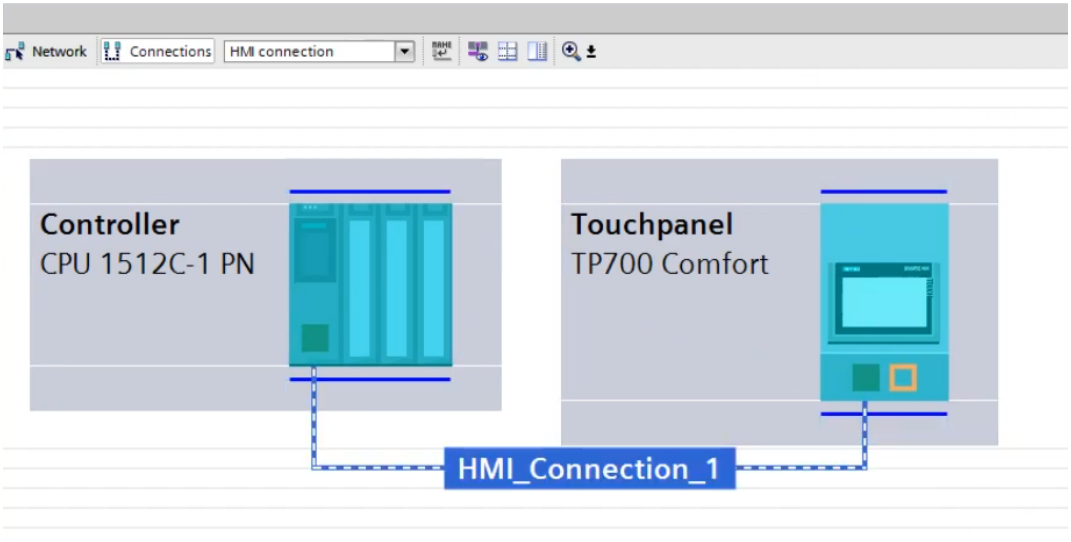


Figure 24 HMI Connection the PLC

HMI screen Navigation and handling

Here is a guide on configuring HMI screen navigation using dropdowns and buttons in TIA Portal:

Using Dropdowns

- Add a dropdown list to the screen from the Controls toolbox.
- Under Properties > Options, define the dropdown choices.
- Under Events > On Selection Changed, choose Screen > Open Screen.
- Select the target screen to open for each dropdown choice.
- The dropdown selection will navigate to the configured screen when changed.

Using Buttons

- Add a button from the Controls toolbox.
- Under Properties > Events > On Click, choose Screen > Open Screen.
- Select the target screen to navigate to when the button is pressed.
- Optional: Add a button animation like Flashing on button click.
- The button press will open the defined screen.

Tips

- Use meaningful names for dropdown options and buttons.
- Arrange elements logically for clear workflow.
- Add additional navigation conditions like tag states if needed.
- Use colors, shapes, and layouts to emphasize navigation controls.
- Test navigation thoroughly on the HMI device.

With dropdowns and buttons, users can intuitively navigate between HMI screens in TIA Portal. Plan and organize screens and controls for optimal workflow.

Configuring user views, administrator controls, and recipe handling in a Siemens HMI

Here is a guide on configuring user views, administrator controls, and recipe handling in a Siemens HMI using TIA Portal:

User Views

- Create user groups and users under User Administration.
- Assign each user group specific authorization levels.
- Develop separate HMI screens/objects for each user group.
- Restrict operator access to critical functions using permissions.

Administrator View

- Make administrator screens only accessible to admin user group.
- Include controls like recipe management, user administration, alarm configuration.
- Allow changing setpoints, modes, system parameters reserved for admins.
- Audit trails can track administrator actions.

Recipes

- Define recipes under Recipes with ingredients, values, and parameters.
- Create recipe screens and tie recipe buttons/objects to PLC tags.
- Use admin view to enable recipe editing and downloading capabilities.
- Operators can view and load recipes based on permissions.

Restricting Operator Input

- Hide, disable, or restrict access to modifying certain objects.
- Make setpoint fields read-only for operators.
- Limit navigation to critical screens using permissions.
- Allow simulating but not acknowledging alarms.

Clearly defined user views, administrative controls, and recipe workflows enhance the operator experience and securely manage HMIs.

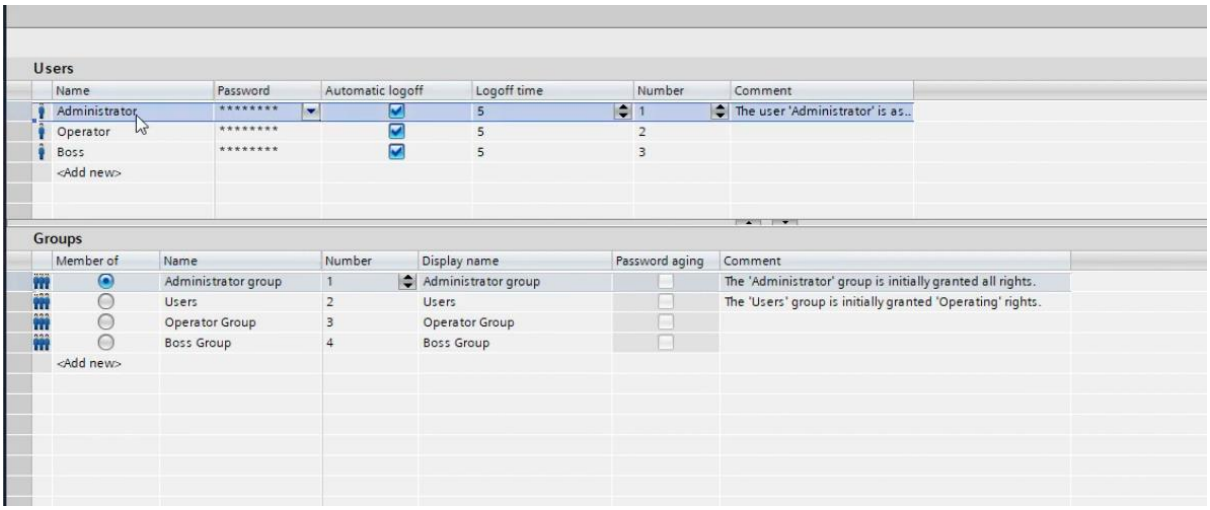


Figure 25 User Views and Administrative Controls

Basic Alarming on HMI

Here is an introduction guide to configuring HMI alarms using TIA Portal:

Introduction to HMI Alarms

- Alarms indicate abnormal conditions or faults to operators.
- TIA Portal allows managing and monitoring PLC alarms from an HMI.
- Alarms have parameters like name, class, priority, tag, message, and status.
- Alarms can be logged in alarm histories for analysis.

Discrete Alarms

- Discrete alarms are triggered by a discrete PLC tag bit, e.g. Motor_Fault bit.
- The bit state change activates the discrete alarm event.

Discrete alarm parameters example:

- Name: Motor_Fault_Alarm
- Priority: High
- Class: Power
- Message: Motor power fault detected

Analog Alarms

- Analog alarms are triggered by an analog tag value.
- High and low limit values define the alarm threshold.

Analog alarm parameters example:

- Name: Tank_Level_Alarm
- Priority: Medium
- Class: Level
- Message: Tank level too high
- Limits: Hi limit = 90%, Low limit = 10%

Alarms can be acknowledged on the HMI based on priority.

Integrating PLC alarms enhances an HMI system by quickly alerting and informing operators. Both discrete and analog alarms are configured easily in TIA Portal.

HMI Events

Human-machine interface (HMI) devices utilize events to trigger actions based on user inputs or changes in component states. Events are configured through the properties of HMI design objects like buttons, switches, and sliders. TIA Portal supports approximately 88 distinct events spanning categories such as click, change, activate, loaded, and more. Proper application of HMI events is key for designing responsive operator interface workflows.

Common HMI Events

Standard HMI components have events including activate, deactivate, click, press, release, change, input finished, loaded, and cleared. Activate and deactivate occur when an element becomes active or inactive. Click comprises press and release on the same object. Change triggers anytime the component value updates. Input finished responds to completed text entry. Loaded and cleared correspond to the HMI screen being opened or closed.

Utilizing HMI Events

HMI events invoke actions like controlling screens, updating tags, managing recipes, and more. For example, a button click event could trigger opening a new HMI display. A slider change event might write the position value to a PLC tag. Correctly leveraging events avoids mis operations like changing values on release instead of submit. Thorough testing on the live HMI is the key.

Events are central to designing reactive, intuitive HMI systems in TIA Portal. While the breadth of possible events can be overwhelming initially, focus on the common events for standard components. With experience, utilizing more advanced events unlocks further HMI workflow flexibility and responsiveness.

- Events allow triggering actions in response to HMI operations like button presses, alarms, etc.
- Configured on HMI screen objects under Properties > Events.
- Wide range of possible events: Press, Release, Click, Change, Select, etc.
- Events execute configured actions like controlling screens, tags, recipes, logs.

Here is a guide on using events in TIA Portal HMI development:

Button Press Events

- Add a Button object to the HMI screen.
- Under Events > Press, select the action, for example:
    - Screen/Open Screen - Navigate to a screen
    - Tag/Write Tag - Set a PLC tag value
    - Alarm/Alarm reporting - Trigger an alarm
- When the button is pressed, the linked event occurs.

Alarm Events

- Under Alarms, configure an alarm and tie it to a PLC tag.
- In Events > On Alarm, choose actions to perform on an active alarm:
    - Screen/Open Screen - Show an alarm screen
    - Log/Log alarm - Record alarm to log
    - E-mail/Send email - Notify over email
- The event responds to the alarm trigger.

Using Events

- Events allow HMI screens to interact with the PLC program.
- Configure events on most HMI objects.
- Link events to associated actions through easy dropdown selections.
- Test event operation thoroughly on the HMI device.

HMI Advanced Design Concepts

Similar to the PLC programming, TIA Portal offers many advanced HMI design features beyond basic graphical configuration:

HMI Templates

Templates allow saving standardized HMI object layouts for reuse across multiple projects. This accelerates development by avoiding reconfiguring common screens.

Global Libraries

Libraries centralize commonly used graphics, objects, scripts, and text for uniformity across all screens and projects. They ensure consistent look and feel.

Alarms and Trends
Robust alarm management and historical trend visualization are critical for operator awareness and responding to abnormal conditions.

User Access Levels
Configuring user access levels and permissions allows securing privileged actions and visibility. This ensures proper operation according to roles.

Audit Trail and Change Logs
Detailed logging of operator actions, alarms, system changes, and errors supports auditing requirements.

Multi-language Text
HMI text can be configured with translations for international sites and languages.
There are many additional HMI advanced design concepts to explore, like screen navigation hierarchy, scripting for dynamic behavior, and animation of graphical objects. Mastering these concepts allows creating customized, informative interfaces that improve visibility into the automation system.

Downloading to the PLC

Once simulation is successful, we can download the complete project to the PLC and HMI:

1. In Project View, select "Download to device."
2. Select the target PLC and HMI panel.
3. Click Load.
4. Confirm the hardware matches the project configuration.
5. Click Download to transfer the PLC program and HMI project.
6. Monitor the download process status.
7. Complete any required hardware configuration to activate the new program.

The PLC and HMI are now running the new program and visualization interface. The system is ready for on-site commissioning and production runtime.

Chapter 4 - Troubleshooting TIA Portal Projects

Despite thorough simulation and testing, issues can still arise when commissioning TIA Portal projects on real hardware. Some potential problems and mitigations include:

PLC Communication Errors

- Verify network cabling, addresses, and communication protocol settings
- Check firewall permissions if connecting through IT networks

HMI Runtime Errors

- Confirm HMI panel model matches project configuration
- Update graphic card drivers and install any OS updates

I/O Address Mismatch

- Cross-check I/O tagging against wiring diagrams
- Monitor I/O forcing to identify mismatched addresses

Logic Execution Issues

- Use debugging tools like breakpoints and watch tables
- Add temporary logging or I/O at key logic points

Simulation Differences

- Confirm simulator configuration matches hardware
- Adjust logic timing as needed

Following standard troubleshooting methodology of breaking down and isolating issues methodically can help resolve most problems. TIA Portal provides diagnostic tools to get to the root cause of errors.

Cross References

Cross references are a powerful feature in Siemens TIA Portal that allow tracking connections between different elements in a project. Cross references show dependencies and relationships between tags, blocks, HMI screens, and other components.

Generating Cross References

The cross reference table is found under the Project tab in TIA Portal. Selecting "Cross References" will analyze the project and produce a table showing the usage of each element. Filters can narrow down cross references by type and scope. The table can also be exported to Excel for additional analysis.

Content of Cross References

Typical cross reference connections displayed include:

- Tag usage - Which blocks, screens, recipes reference a tag
- Block calls - What blocks call another block

- Screen usage - What screens reference a certain HMI screen
- Alarm usage - Where an alarm type is used
- Data block access - What code accesses a data block

Double-clicking a cross reference jumps directly to the location for further inspection.

Leveraging Cross References for Troubleshooting

Cross references enable fast navigation through a project to track down issues. Some examples:

- Find cause of tag issues - Check what logic or screens are accessing a tag
- Identify program dependencies - See what chain of blocks call each other
- Diagnose HMI troubles - Quickly locate screens and objects tied to a screen
- Check alarm configuration - View where an alarm is bound in the HMI
- Verify data access - Check read/writes to a data block across logic

Additionally, cross references support change impact analysis by highlighting changes since the last compile. This reveals modifications that could be causing problems.

Guidelines for Using Cross References

Follow these tips for effectively applying cross references:

- Generate cross references initially as a project overview
- Regenerate cross references after major changes
- Use filters to narrow down specific cross reference types
- Export tables for more flexible analyses in Excel
- Follow call chains by double-clicking cross reference entries
- Compare cross references before and after changes to find impacts

Overall, cross references provide invaluable visibility into the inner connections of TIA Portal projects. Properly utilizing this tool can significantly accelerate troubleshooting, maintenance, and other critical engineering tasks. Cross references are an underused but very powerful capability in TIA Portal.

Traces for Troubleshooting

Traces allow recording diagnostic data during execution of a TIA Portal project. They capture values over time for tags, inputs, outputs, and internal program signals. Traces support debugging logic and analyzing system behavior. Traces are an invaluable tool for debugging logic and analyzing system behavior.

Configuring Traces

To configure a trace, first create a trace recording job under Recording and Traces in the Project tree. Next, define the signals to record, which may include tags, I/O channels, data block registers, etc. It is advisable to focus traces on relevant tags and I/O to reduce data volume. The trace duration and number of recordings can be adjusted as needed. Shorter traces help verify single operations.

To set up a trace:

1. In the Project tree, select Recording and Traces > Configuration.
2. Create a new trace configuration.
3. Select scan cycle or time-based recording.
4. Choose signals to record like tags, I/O channels, DB registers.
5. Select amount of recordings and duration of trace.

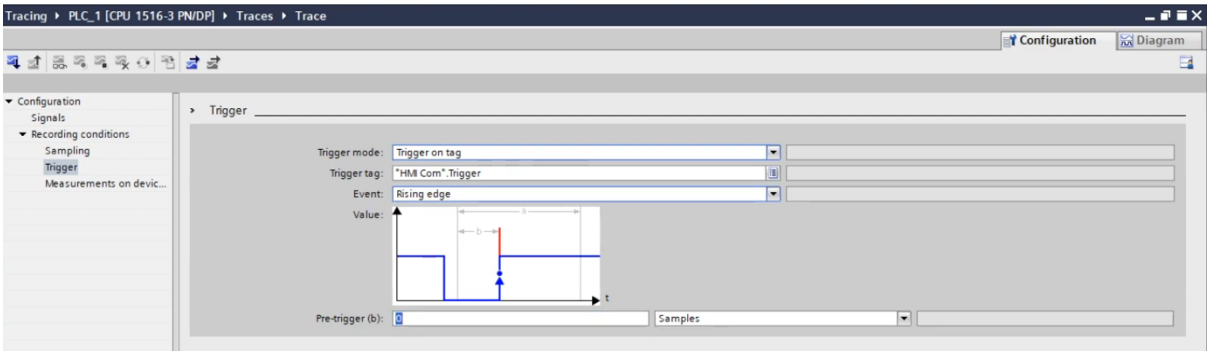


Figure 26 Configuration of Traces for troubleshooting

Executing Traces

Traces can be activated through SIMATIC Manager, PLCSIM, or the physical PLC. The recorded values are then saved to a CSV file for analysis after stopping the trace. Synchronizing trace start/stop with specific operations aids troubleshooting.

Traces are started:

- In SIMATIC Manager by activating the trace
- In PLCSIM by enabling the trace
- On the physical PLC through configuration

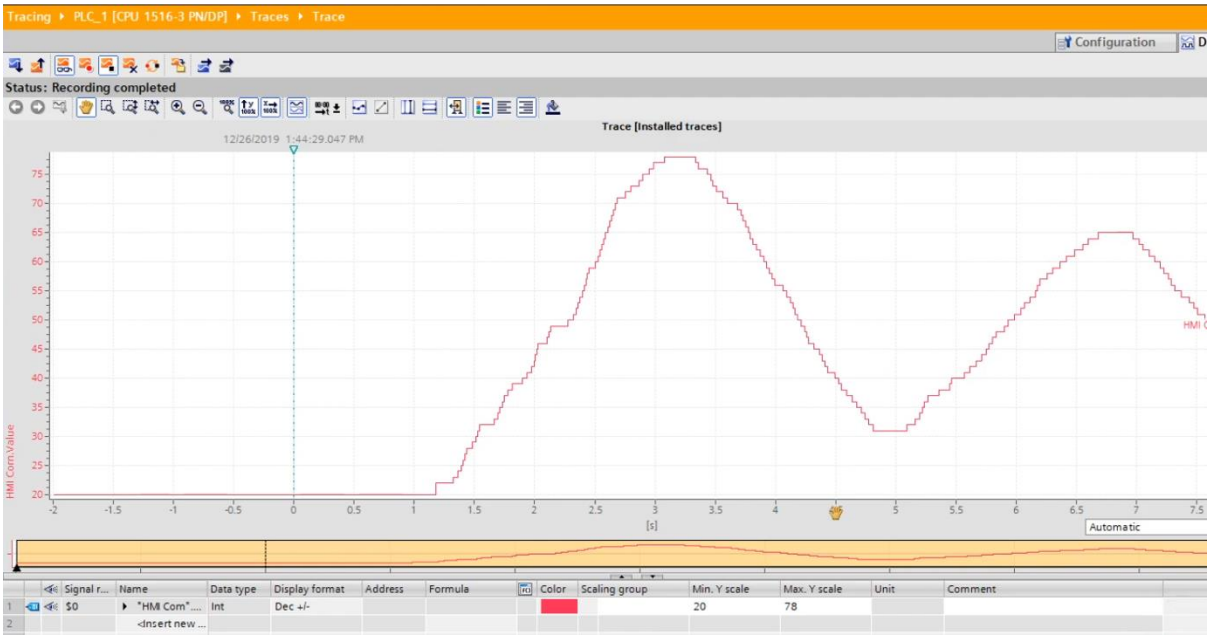


Figure 27 Recording the traces during the simulation

Using Traces to Troubleshoot

Traces help troubleshoot issues like verifying I/O behavior, analyzing calculations, detecting timing problems, monitoring processes, catching variable limits, and profiling usage. They reveal unseen behaviors during normal runtime. Comparing traces before and after correcting an issue validates the fix.

Some examples of using traces for troubleshooting:

- Check I/O signals - Verify if an I/O channel is toggling as expected
- Analyze calculations - Follow an internal register through program execution
- Detect timing issues - Determine if operations meet cycle time requirements
- Monitor processes - Record process values during specific operations
- Catch limits - Identify where variables exceed configured limits
- Profile usage - Evaluate maximum and minimum signal utilization

Traces allow isolating the source of issues by capturing data unseen during normal runtime.

Guidelines for Effective Traces

Follow these guidelines to create useful traces:

- Focus traces - Limit signals to relevant tags and I/O to reduce data volume
- Adjust duration - Use shorter traces for verifying single operations
- Align timing - Synchronize trace to repeating actions by starting/stopping
- Reproduce issues - Execute trace under conditions causing the problem
- Confirm fixes - Run trace again after correcting to validate
- Document results - Save trace files and record conclusions

In summary, traces are invaluable for capturing diagnostic data on program execution. Used properly, they enable understanding of complex logic and help pinpoint transient errors. Traces should become an integral part of any troubleshooting methodology.

S7 Routing and Accessing Devices Through Network for Troubleshooting

Siemens S7 routing is a crucial aspect of industrial automation systems, and it plays a significant role in accessing and troubleshooting devices through a network in the Totally Integrated Automation (TIA) Portal. In this extensive explanation, we will delve into the fundamentals of S7 routing, its importance in industrial automation, and how it is used for troubleshooting purposes, emphasizing its relevance in the modern manufacturing landscape.

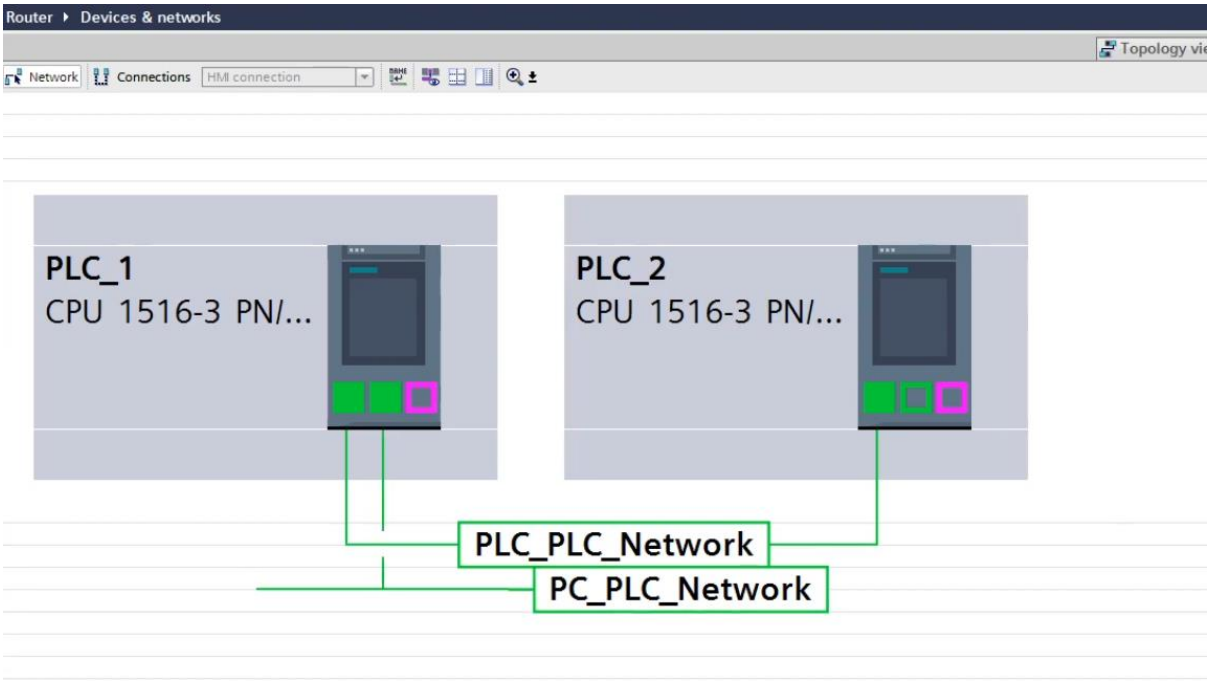


Figure 28 PLC Connections - Network

Understanding S7 Routing

S7 routing refers to the methodology and set of protocols used in Siemens' PLCs to facilitate communication between various devices over a network. The primary objective of S7 routing is to establish a seamless connection between programmable controllers, HMIs, distributed I/O modules, and other industrial devices. It enables data exchange, program uploading and downloading, and remote diagnostics, contributing to the overall efficiency and reliability of industrial processes.

Devices Accessible Through Network

Through S7 routing, various industrial devices can be accessed and controlled via a network. These devices include:

- **PLCs (Programmable Logic Controllers):** PLCs are the central brain of industrial automation systems. With S7 routing, engineers can connect to PLCs remotely, monitor their status, upload or download programs, and troubleshoot issues without physically accessing the PLC hardware.
- **HMI (Human-Machine Interface):** HMIs provide a graphical interface for operators to monitor and control industrial processes. S7 routing allows remote access to HMIs, making it possible to adjust parameters, view process data, and diagnose problems from a central location.
- **Distributed I/O Modules:** These modules are used to extend the reach of digital and analog inputs/outputs within a system. Through S7 routing, technicians can connect to these modules remotely, check sensor values, and reconfigure I/O settings as needed.
- **Remote Maintenance Units (RMUs):** RMUs are specialized devices used for remote diagnostics and troubleshooting. S7 routing enables technicians to access RMUs and diagnose issues across the entire automation network.
- **Industrial Networks:** S7 routing is integral to communication within industrial networks, such as Profinet and Profibus. It ensures seamless data exchange between devices, leading to synchronized operations and real-time monitoring.

Importance of S7 Routing in Troubleshooting

Effective troubleshooting is critical in maintaining uptime and minimizing production losses in industrial settings. S7 routing plays a pivotal role in this process for several reasons:

- **Remote Diagnosis:** One of the primary advantages of S7 routing is the ability to diagnose issues remotely. Technicians can connect to the PLCs, HMIs, or other devices from a central location, reducing the need for on-site visits. This is especially valuable when dealing with geographically dispersed facilities.
- **Real-Time Monitoring:** S7 routing enables real-time monitoring of industrial processes. Engineers can access data, alarms, and system status remotely, allowing them to detect anomalies promptly. This proactive approach minimizes downtime and prevents potential issues from escalating.
- **Program Debugging:** Troubleshooting often involves identifying and rectifying programming errors in PLCs. S7 routing allows engineers to upload and download PLC programs, review code, and make necessary modifications remotely. This capability accelerates the debugging process.

- **Reduced Downtime:** By swiftly identifying and addressing issues through remote troubleshooting, S7 routing significantly reduces downtime. Production lines can continue running smoothly while technicians work on problem resolution.
- **Cost Efficiency:** Traditional troubleshooting methods may involve sending technicians on-site, incurring travel expenses, and causing delays. S7 routing eliminates or reduces these costs, making maintenance and troubleshooting more cost-effective.

Configuring S7 Routing in TIA portal

To use S7 routing for troubleshooting, it's essential to configure it properly in the TIA Portal, Siemens' integrated engineering environment. The following steps outline the process:

- **Network Configuration:** Set up your industrial network with all the relevant devices, including PLCs, HMIs, and I/O modules. Ensure that each device has a unique IP address and is accessible within the network.
- **Routing Configuration:** Configure the routing settings in the TIA Portal. Specify the routing path, which includes the source and destination devices. Define the communication protocols, such as Profinet or Profibus, and ensure that routing tables are correctly populated.
- **Security Measures:** Implement security measures to protect the network from unauthorized access. This may include setting up firewalls, VPNs, and access control policies to safeguard sensitive industrial data.
- **Remote Access Software:** Install remote access software or tools compatible with Siemens' devices. This software allows engineers and technicians to establish secure connections to the devices they need to troubleshoot.
- **Monitoring and Diagnostics:** Once the S7 routing is configured, engineers can monitor device statuses, collect data, and diagnose issues remotely using the TIA Portal or dedicated software solutions.

Advanced Programming Techniques

While the previous section outlined basic PLC programming, TIA Portal offers advanced techniques for optimizing control logic and handling complex applications. Here are some examples:

Reusable Function Blocks

Function blocks encapsulate common logic into reusable modular units. This avoids reprogramming the same code. For example:

- Motor start/stop sequences

- Proportional–integral–derivative (PID) loops
- Timer and counter operations

Function blocks allow drag-and-drop reuse and configuration via inputs and outputs.

Structured Text Programming

Structured Text (ST) is a high-level language commonly used for complex algorithms and math functions. It provides:

- Functions, function blocks, methods
- Loops, branches, returns
- Arrays, structures, pointers

ST complements LAD/STL with text-based code suitable for some tasks.

Online Editing

With an online connection to the PLC, programmers can edit logic and download changes while the PLC is running. This facilitates program optimization.

Cross-Project Navigation

Multiple projects can be grouped and accessed through cross-project navigation. This allows reusing standardized code across different projects.

Version Control Integration

TIA Portal can integrate with version control systems like Git. This enables:

- Saving program revisions
- Comparing logic changes
- Rolling back to previous versions

Version control is crucial for managing programs developed by teams.

In addition to these examples, TIA Portal contains many other advanced programming tools for needs such as trace functionality, safety applications, and support for IEC languages. Leveraging these features is key for scalable and robust PLC logic.

Deploying for Runtime

Once the software is tested and validated, engineers must transfer the project to the target runtime environment. This involves:

- Backing up the engineering project files and database(s).
- Installing any required runtime software to the HMI panel and other devices.
- Transferring PLC, HMI, and drive configuration to their respective hardware.

- Testing process operation with real I/O devices.
- Optimizing PLC logic and HMI operation based on observed runtime behavior.
- Training operators on proper use of the HMI interface.
- Documenting the final commissioned program and hardware setup.

Adhering to configuration management and site acceptance testing procedures is vital for smooth handover to production. Backups and restoration procedures should also be established.

With an organized deployment process, TIA Portal projects can transition reliably from engineering to live plant operation while maintaining revision control and stability.

Documentation and Revision Control

Thorough documentation is essential for maintaining and modifying TIA Portal projects:

- Program printout with annotations
- Version control commit notes
- HMI screen descriptions
- Hardware layout diagrams
- Revision change log

Revision control is also necessary to:

- Track program changes
- Revert to prior versions if needed
- Compare changes between versions

This allows new engineers to understand logic intent and provides a backup in case of software issues.

Following industry best practices for documentation, backups, and version control helps ensure maintainability and continuity for TIA Portal automation projects over their lifecycle.

Chapter 5 – TIA Portal Programming Fundamentals

The key topics covered in the chapter include creating a new project in TIA Portal, using the hardware catalog, understanding the main OB block, ladder logic diagrams, timers, counters, math functions, data blocks, and other foundational PLC programming elements.

Steps to Create a New Project in TIA Portal

Following these steps below enables quickly creating a new TIA Portal project with devices, network configurations, and program logic tailored to the particular automation system. The saved project provides the foundation for ongoing editing and maintenance over the plant lifecycle.

1. Open TIA Portal - Launch the TIA Portal application.
2. Select Project View - Choose "Project View" to access project management tools.
3. Click Create New Project - In the Project View window toolbar, press the "Create new project" icon.
4. Name the Project - Enter a meaningful name for the new project in the dialog box.
5. Select Project Path - Choose the file system location to store project files.
6. Add Devices - Insert desired hardware modules like I/O, HMI, drives into the project.
7. Choose CPU Type - Select the target Siemens PLC type from the product catalog.
8. Configure Device Settings - Parameterize devices with IP addresses, cycle times, and other settings.
9. Create Program Code - Develop the required PLC, HMI, safety programs.
10. Save Project - Save the new project to disk. It is now ready for editing and commissioning.

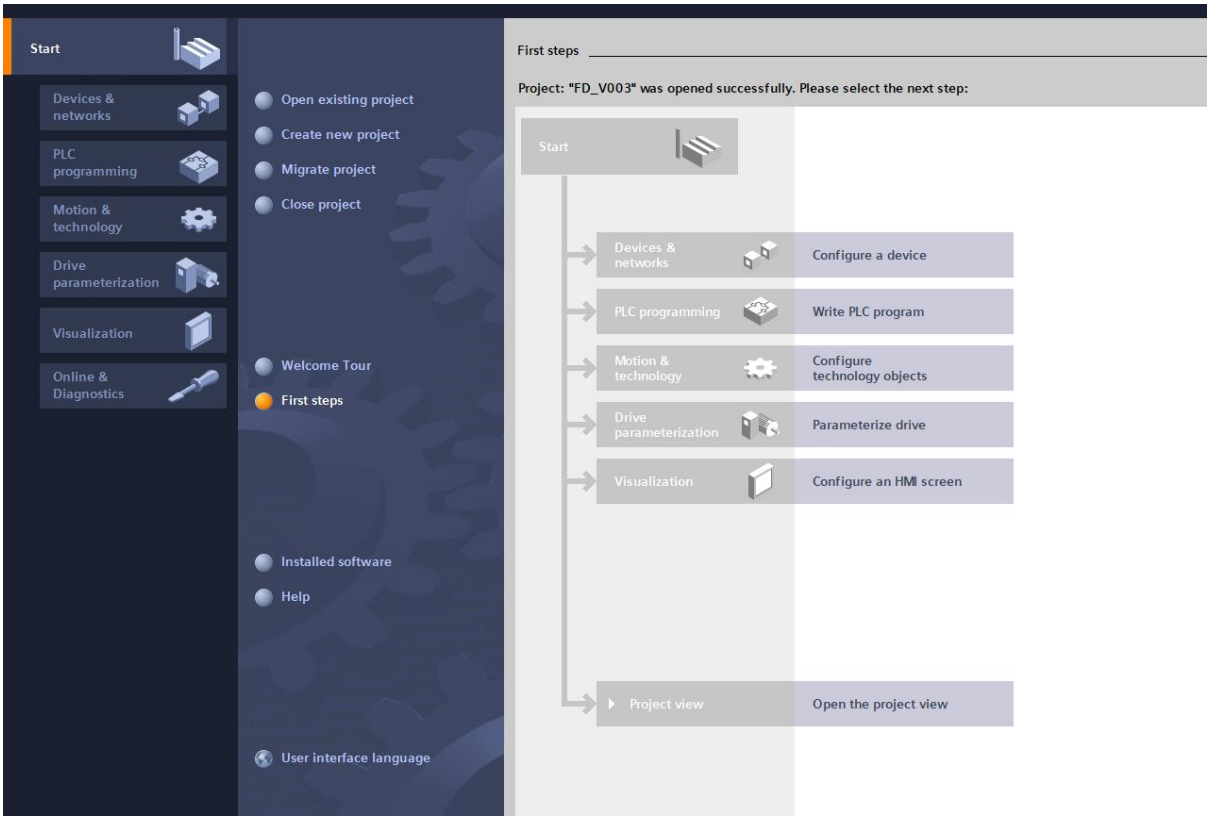


Figure 29 New Project Window in TIA Portal

When creating a project in Siemens TIA Portal, the approach for simulation versus connecting to physical hardware differs. For simulation, virtual PLCs and devices can be added to a project without regard to specific product serial numbers. Any CPU model or hardware module from the catalog can be chosen since no physical connection is being made. This provides flexibility during programming and testing.

However, when connecting to real industrial equipment, the actual serial numbers and product details must match what is inserted into the TIA Portal project. The software cross-references the unique serial number and exact product references when linking to physical controllers and devices. Choosing the precise PLC model and modules to match what is wired up is crucial for proper functionality. Hardware configuration and parameterization depends on accurate selection in TIA Portal based on the real-world components.

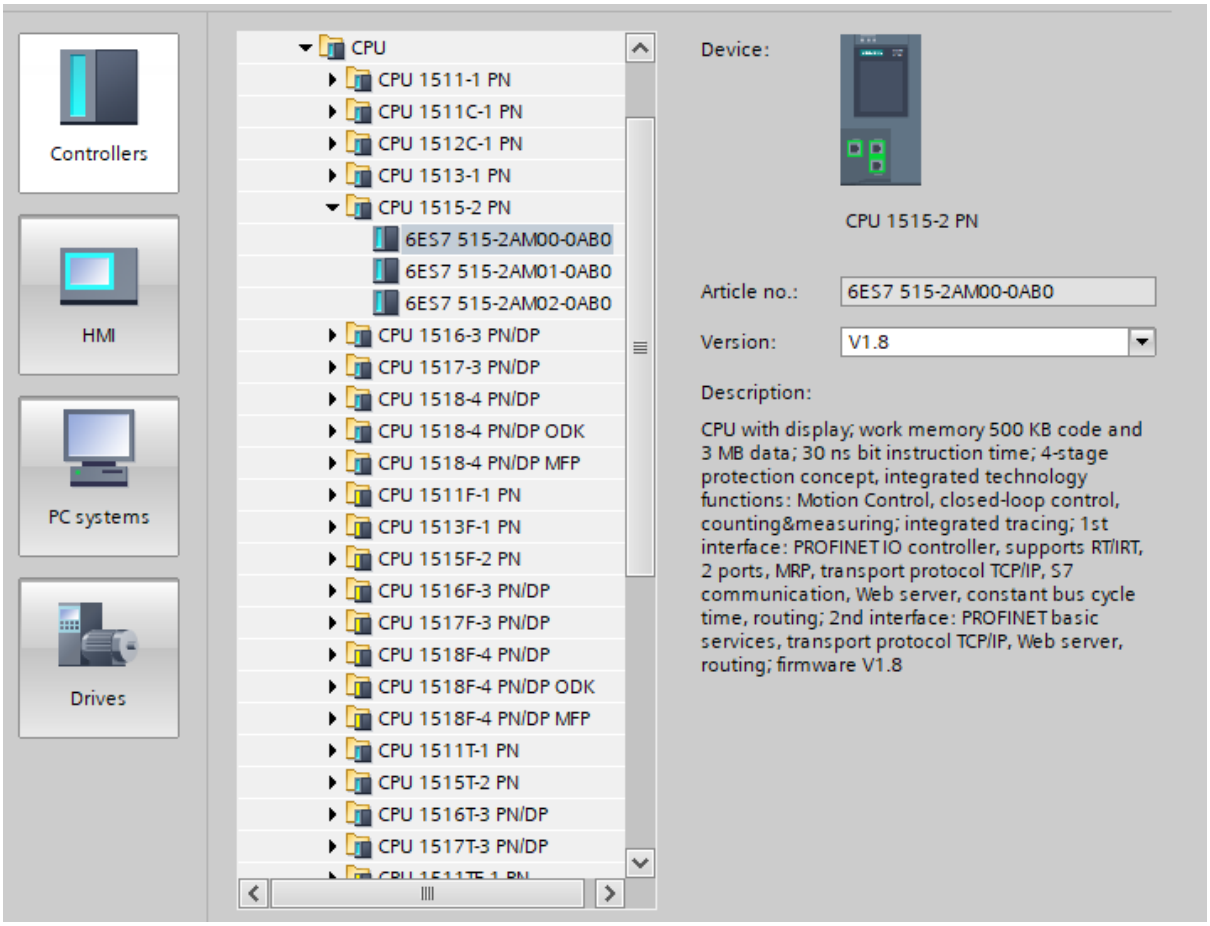


Figure 30 Device Selection (Hardware Catalog) window

Hardware Catalog in TIA Portal

The Device Catalog in Siemens TIA Portal provides access to the library of available hardware components for configuring automation systems. It's essentially a library of all the available hardware modules, devices, and components that can be used to build and configure a PLC (Programmable Logic Controller) or other Siemens control systems. Here's an overview of the types of modules you can find in the hardware catalog of TIA Portal:

The catalog in TIA Portal (Totally Integrated Automation Portal) is a crucial feature that allows engineers and programmers to configure and assemble the hardware components of a Siemens automation system. It's essentially a library of all the available hardware modules, devices, and components that can be used to build and configure a PLC (Programmable Logic Controller) or other Siemens control systems. Here's an overview of the types of modules you can find in the hardware catalog of TIA Portal:

1. CPU (Central Processing Unit): This is the heart of the PLC, responsible for executing the control program. The catalog includes various CPU models with different processing capabilities and features.

2. Signal Boards: Signal boards are used to extend the capabilities of the CPU. They can provide additional communication ports, digital inputs/outputs, analog inputs/outputs, and other functions.

3. Communication Boards: These modules enable the PLC to communicate with other devices and networks. They support various communication protocols such as Ethernet, PROFINET, Profibus, AS-Interface, and more.

4. Battery Board: The battery board provides backup power to the PLC's internal clock and memory to retain data in case of a power failure.

5. Digital Input (DI) Modules: DI modules are used to interface with digital signals from sensors, switches, or other devices. They come in various configurations with different numbers of input channels.

6. Digital Output (DQ) Modules: DQ modules are used to control digital outputs such as relays, solenoids, or indicator lights. They also come in different configurations with varying numbers of output channels.

7. Analog Input (AI) Modules: AI modules are used to interface with analog signals, such as voltage or current signals from sensors or transmitters. They offer different resolutions and input voltage/current ranges.

8. Analog Output (AO) Modules: AO modules are used to generate analog output signals, typically for controlling analog devices like motors or valves. They also offer various voltage/current output ranges.

9. Special Function Modules: These modules serve specific purposes, such as high-speed counters, motion control, position detection, and more. They are used to expand the functionality of the PLC for specialized applications.

10. Power Supplies: Power supply modules provide the necessary electrical power to the PLC and its associated modules. They come in different voltage and current ratings.

11. Rack and Mounting Accessories: These components are used to physically mount and organize the modules within the control cabinet or panel. They include racks, bus connectors, and terminal blocks.

12. Industrial PCs and HMI: TIA Portal also includes hardware components for building industrial PCs and HMI panels that can be integrated into our automation system.

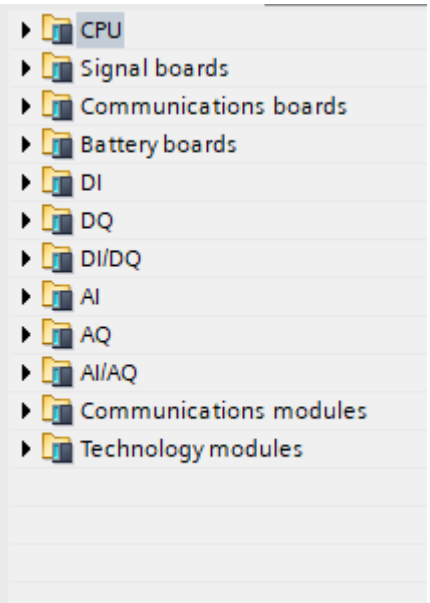


Figure 31 Types of modules available in the hardware catalog of TIA Portal

Overview of Main [OB]:

In TIA Portal software, the "Main [OB1]" is a critical and fundamental part of the program structure, especially in ladder logic programming.

Overview of Main [OB1]

- OB1 stands for "Organization Block 1." In the world of Siemens PLC programming, OB1 is the primary or main organization block.
- The Main [OB1] block is a special and mandatory block in the PLC program. It's always executed cyclically as part of the PLC's scan cycle.
- The scan cycle is the continuous process where the PLC reads inputs, executes the control program, updates outputs, and repeats this cycle over and over to control a machine or process.

Role of Main [OB1]

- Main [OB1] contains the user's control program logic, which defines how the PLC responds to inputs and generates outputs.
- This is where we write the actual control algorithms, decision-making logic, and sequencing steps for our automation process.
- It's common to use ladder logic, function block diagrams, or structured text within Main [OB1] to program the PLC's behavior.

Execution Order

- Main [OB1] is executed sequentially from top to bottom. The logic within it is processed one instruction at a time.
- We can use various programming elements such as contacts, coils, timers, counters, and function blocks to create complex control sequences.

Scanning Process

- When the PLC is in "Run" mode, it scans Main [OB1] repeatedly at a specific scan rate (cycle time) defined in the PLC configuration.
- During each scan, the PLC evaluates the logic in Main [OB1] based on the current state of inputs and internal variables.
- Outputs are updated based on the evaluation of the logic. Any changes to outputs take effect at the end of the scan cycle.

Organization Blocks (OBs):

- In addition to Main [OB1], there are other organization blocks in TIA Portal (e.g., OB2, OB3, etc.), each serving a specific purpose.
- These blocks are typically used for handling specific events, error handling, or time-critical tasks.

Safety Considerations

- It's crucial to design Main [OB1] carefully, as it directly affects the behavior of our automation system.
- Ensuring that our program is safe and reliable is of utmost importance. Proper error handling and fault detection should be incorporated into our Main [OB1] logic.

In summary, Main [OB1] in TIA Portal is the central element of the PLC program where we define the control logic for our automation process. It's executed cyclically, and its proper design and programming are essential for the successful operation of the PLC-controlled system. Careful consideration of safety, efficiency, and reliability is necessary when developing the logic within Main [OB1].

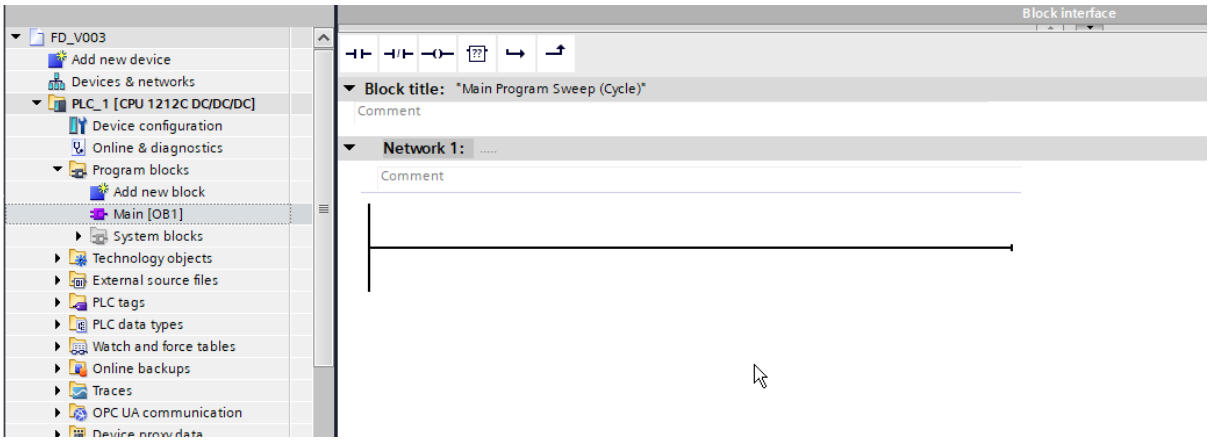


Figure 32 Main [OB1] in TIA Portal - the central element of a PLC program

Basic Instructions in TIA Portal PLC Programming

The Instruction tree in TIA Portal provides the basic Ladder Logic instructions for PLC programming. These predefined instructions execute common logic, bit, timer, counter, math, and data operations.

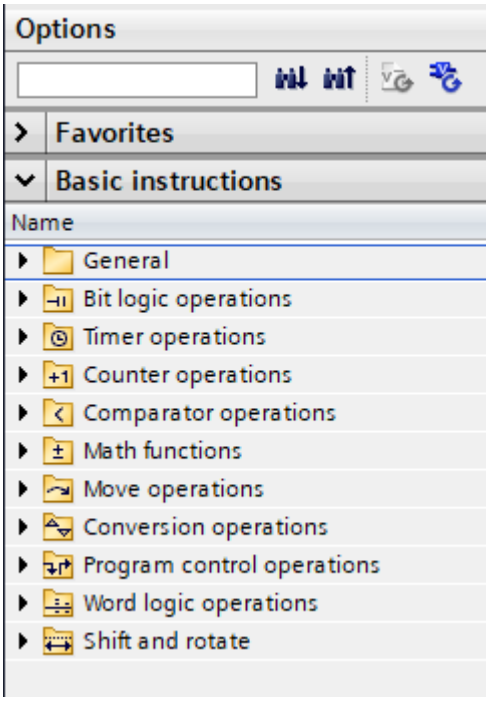


Figure 33 Instruction tree in TIA Portal

For example, in the Bit Logic Operations there are some blocks. These blocks are fundamental in ladder logic programming and are used to create a wide range of control logic for industrial automation and processes. NO and NC contacts, coils, and flip-flops are essential for creating basic control sequences, while P_TRIG and N_TRIG functions add the capability to trigger actions based on edge transitions, which is crucial for precise timing and control.

Table 3 Comprehensive Overview of Essential TIA Portal PLC Instructions

Category	Instruction	Description
Bit Logic Operations	NO	Represents a digital input or a condition that is true when the input is active (e.g., a push button being pressed).
	NC	Represents a digital input or a condition that is true when the input is not active (e.g., a limit switch in its normal state)
	Coil (Output Coil)	Represents a digital output or an action that is activated when the associated control conditions are met (e.g., turning on a motor or solenoid valve).
	Flip-Flop (SR Flip-Flop or RS Flip-Flop):	A flip-flop is a memory element that can be used to store a state (either ON or OFF) until it is explicitly reset or toggled. It is often used for maintaining the state of equipment or processes

Category	Instruction	Description
	P_TRIG (Positive Edge Trigger):	P_TRIG is a timer function that triggers actions or starts timers when an input signal transitions from FALSE to TRUE (rising edge). It is used to initiate actions based on the activation of an input.
	N_TRIG (Negative Edge Trigger):	N_TRIG is a timer function that triggers actions or resets timers when an input signal transitions from TRUE to FALSE (falling edge). It is used to initiate actions based on the deactivation of an input.
Timer Operations	TP (Pulse Timer)	Generates a pulse output when a specified time duration is reached.
	TON (Timer On-Delay)	Activates an output after a specified time delay has elapsed.
	TOF (Timer Off-Delay)	Deactivates an output after a specified time delay has elapsed.
	TONR (Retentive Timer On-Delay)	Activates an output after a specified time delay has elapsed and retains the elapsed time even if the input condition goes false.
Counter Operation	CTU (Counter Up)	Counts up based on input pulses.
	CTD (Counter Down)	Counts down based on input pulses.
	CTUD (Counter Up/Down)	Combines both up and down counting functions.
Comparator Operations	EQU (Equal)	Compares two values for equality.
	NEQ (Not Equal)	Compares two values for inequality.
	GT (Greater Than)	Checks if one value is greater than another.
	GE (Greater Than or Equal)	Checks if one value is greater than or equal to another.
	LT (Less Than)	Checks if one value is less than another.
	LE (Less Than or Equal)	Checks if one value is less than or equal to another.
	IN_Range (Input Range)	Counts input events or conditions within a specified range.
	OUT_Range (Output Range)	Counts input events or conditions outside a specified range.
Math Functions	Calculate	Performs various mathematical calculation based on the user-defined logic.
	ADD	Adds two values.
	SUB (Subtract)	Subtracts one value from another.
	MUL (Multiply)	Multiplies two values.

Category	Instruction	Description
	DIV (Divide)	Divides one value by another.
	MOD (Modulus)	Calculates the remainder when one value is divided by another.
	ABS (Absolute)	Returns the positive value of a number.
	FRAC (Fraction)	Returns the fractional part of a real number.
Move Operations	MOV (Move)	Copies data from one location to another.
Conversion Operations	CONVERT	Converts between data types like INT to REAL
	ROUND	Rounds a real number to nearest integer
	CEIL	Rounds a real number up to next integer
	FLOOR	Rounds a real number down to previous integer
	TRUNC	Truncates the fractional part of a real number
	SCALE_X	It allows you to convert a value from one unit of measurement or range to a different unit of measurement or range
	NORM_X	converts an input value to a standardized range, facilitating consistent data processing and comparison.
Program Control Operations	Jump	Jumps to another network.
	Label	Create a label so the jump will jump to that label in that network.
Word Logic Operation	AND	Performs a logical AND operation.
	OR	Performs a logical OR operation.
	XOR	Performs a logical XOR operation.
	INVERT (INV)	Reverses the state of individual bits within a binary word
Shift and Rotate	SHL (Shift Left)	Shifts bits to the left.
	SHR (Shift Right)	Shifts bits to the right.
	ROL (Rotate Left)	Rotates bits to the left.
	ROR (Rotate Right)	Rotates bits to the right.

Bit Logic Operations

Here we use the most important blocks in the Bit Logic operations.

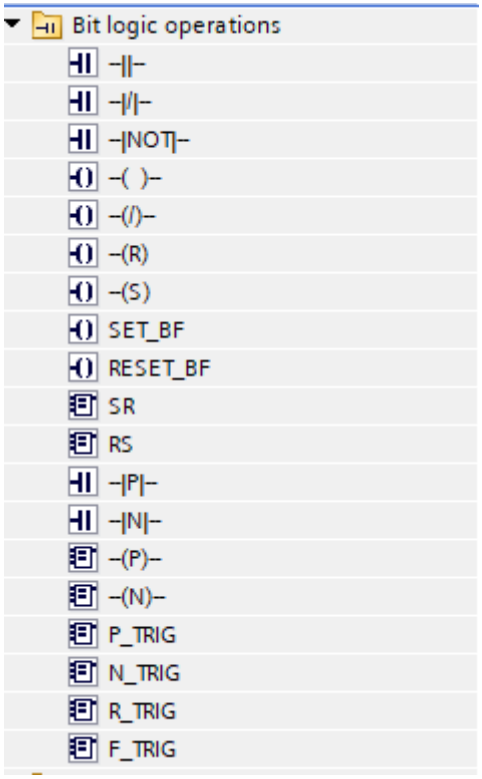


Figure 34 Bit logic Operations blocks

Ladder Logic Diagram for a Single Toggle Push Button

A "Ladder Logic Diagram for a Single Toggle Push Button" represents the control logic for a basic on/off switch using a single push button that toggles the state of an output.

Design the Ladder Logic Diagram

In this ladder diagram, we will use a single push button to toggle an output. Here's a step-by-step breakdown of how to do this:

a. Drag and drop a normally open contact from the toolbox onto the ladder diagram. This represents the push button input. Double-click it to assign an input address, e.g., "I0.0."

b. Connect the normally open contact to a coil by dragging and dropping a coil from the toolbox. This coil represents the output. Double-click it to assign an output address, e.g., "Q0.0."

c. Connect a parallel branch to the same coil by dragging and dropping a normally closed contact and connecting it in parallel with the existing branch. This contact represents the output or Q0.0. So, it keeps it on.

d. Finally, connect the normally closed contact to the same coil.

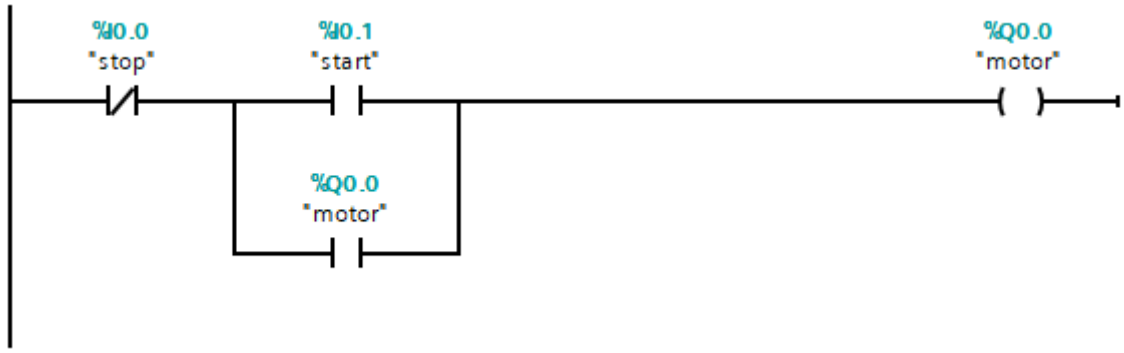


Figure 35 Ladder Logic Diagram for a Single Toggle Push Button – turning on and off a motor

Running the simulation

To run the simulation, click the "Start Simulation" button or press F7.

Then we can monitor the changes in input and output statuses.

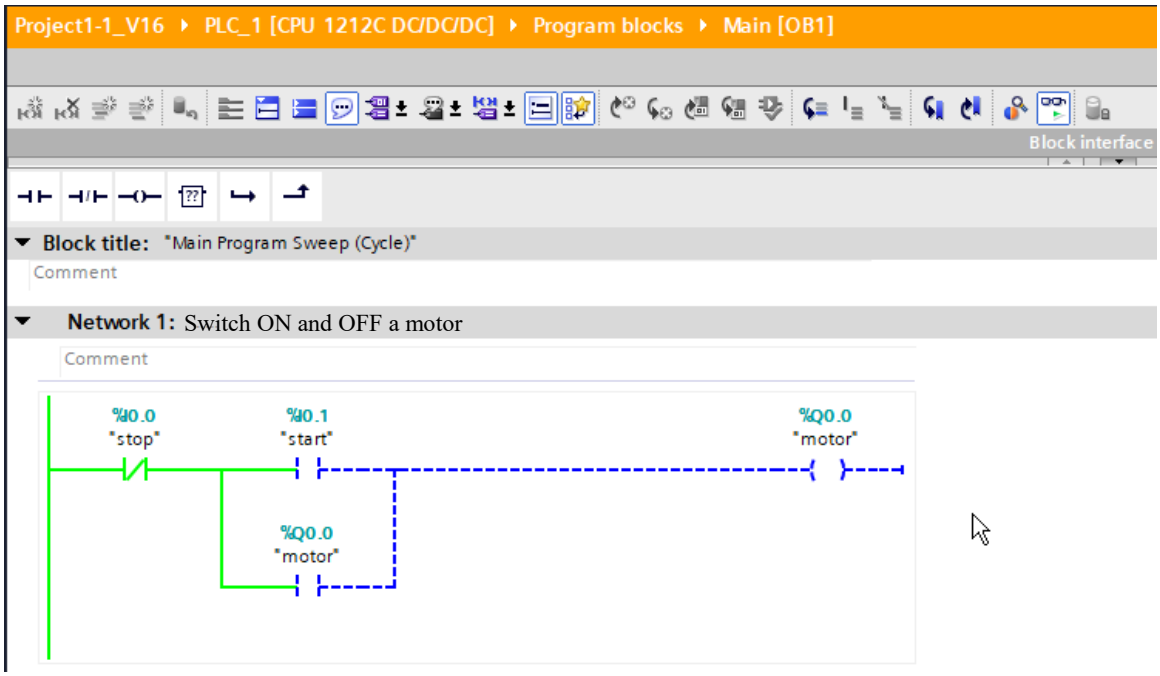


Figure 36 Simulation of the Program

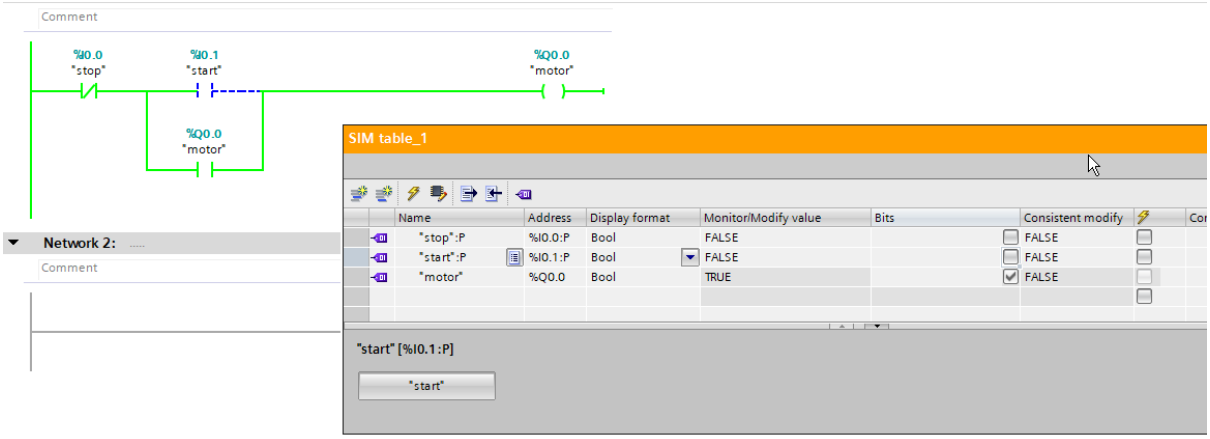


Figure 37 Simulation using the Simulation table for activating or disactivating the blocks

Ladder Logic Diagram using flip flop

A Ladder Logic Diagram using flip-flops, often referred to as a "Latching Circuit" or "Toggle Circuit" in industrial automation, is a common application used to create and control a memory element within a programmable logic controller (PLC). Flip-flops, which are digital logic devices, allow for the creation of stable states or memory within a control system. These circuits are vital for retaining the status of certain operations or processes even after the original triggering conditions have changed.

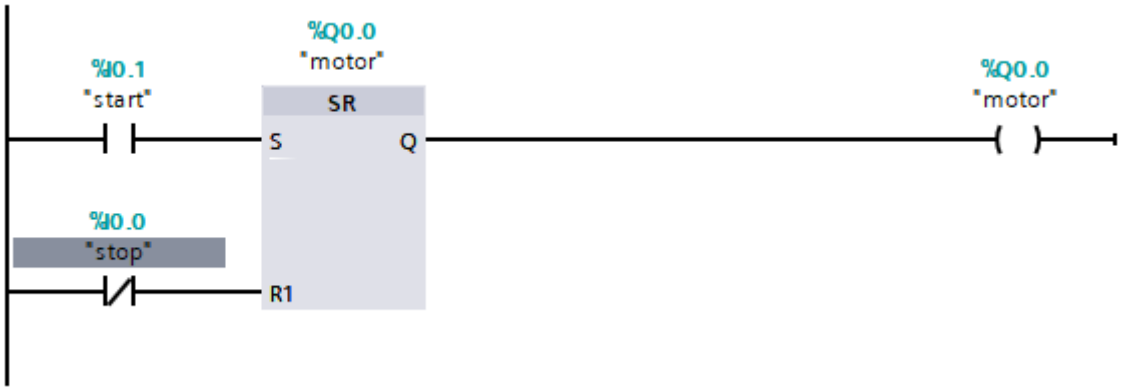


Figure 38 Turing ON and OFF a motor using flip-flop

So, this network is similar to the previous network but instead we are using flip flop. So always for controlling and retaining the status of a certain operation or process flip flops are used.

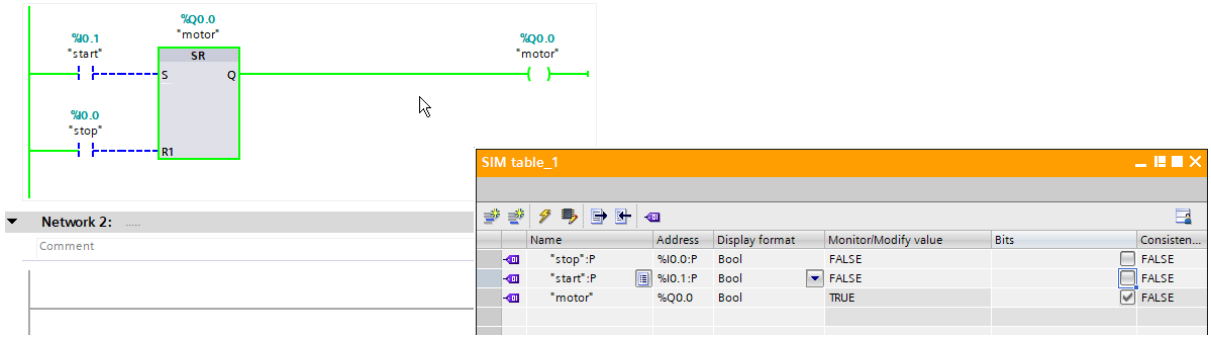


Figure 39 Simulation using the flip-flop with the sim table. Motor is ON here.

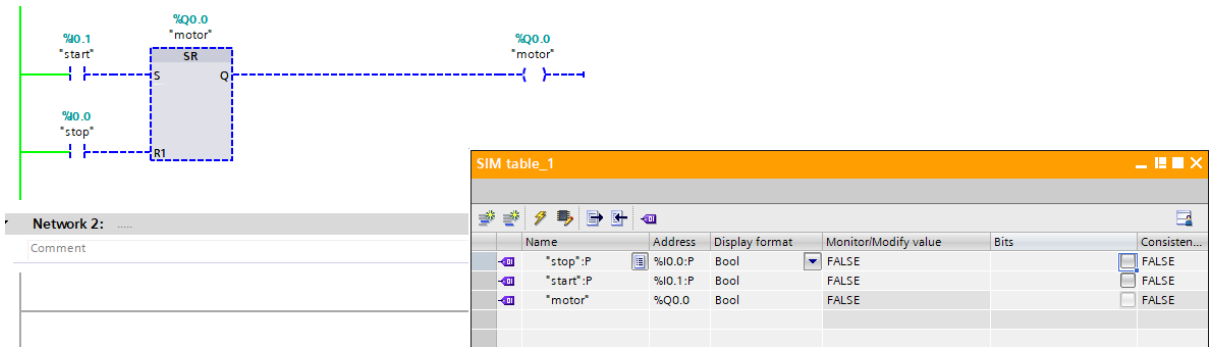


Figure 40 By activation of the stop button the motor would be OFF

Positive Trigger and Negative Trigger

"P_TRIG" and "N_TRIG" are timer functions used to create trigger conditions based on the rising (positive) or falling (negative) edge of a digital input signal. These functions are commonly used to initiate or reset timers, counters, or other control actions in response to specific input events.

Here's an explanation of each:

1. Positive Trigger (P_TRIG):
    - A Positive Trigger, denoted as "P_TRIG" in TIA Portal, is a timer function that activates when the input signal makes a transition from FALSE to TRUE or from 0 to 1 (rising edge).
    - It is used to trigger actions or start timers when an input signal becomes active or transitions from an inactive to an active state.
    - P_TRIG ensures that the associated actions or timers are initiated only when the input condition changes from FALSE to TRUE.
    - For example, you might use a P_TRIG to start a timer when a start button is pressed.
2. Negative Trigger (N_TRIG):
    - A Negative Trigger, denoted as "N_TRIG" in TIA Portal, activates when the input signal makes a transition from TRUE to FALSE or from 1 to 0 (falling edge).
    - It is used to trigger actions or reset timers when an input signal becomes inactive or transitions from an active to an inactive state.
    - N_TRIG ensures that the associated actions or timers are initiated only when the input condition changes from TRUE to FALSE.
    - For example, you might use an N_TRIG to reset a timer when a stop button is pressed.

These trigger functions are essential in ladder logic programming to control the precise timing and sequencing of operations in industrial automation systems. They help ensure that specific actions occur at the correct moments based on the changes in input conditions. By using P_TRIG and N_TRIG functions effectively, we can design control logic that responds accurately to the transitions of digital signals, leading to efficient and reliable automation processes.

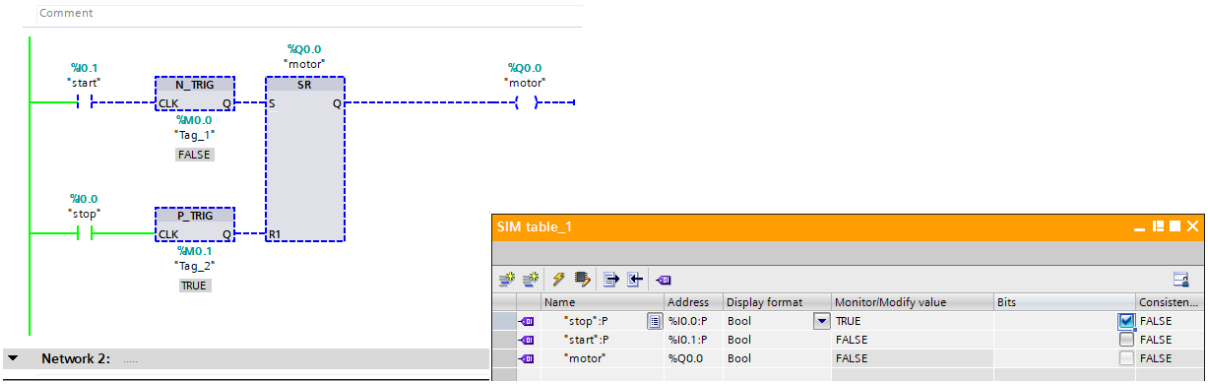


Figure 41 Using N_TRIG and P_TRIG for Turing ON and OFF a motor

Timer operations

Timers are function blocks that provide time delay operations in a PLC program. They are commonly used to insert timed pauses, establish duty cycles, as well as count or measure elapsed time periods.

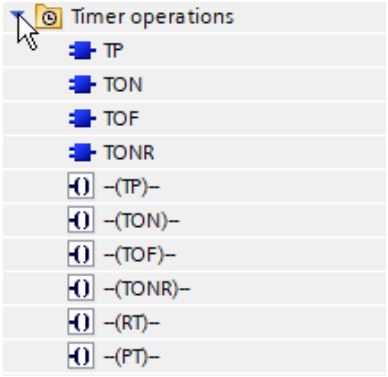


Figure 42 Blocks in Timer Operation

Pulse (TP) Timer

The pulse timer generates a single pulse of preset duration when triggered. Key features:

- Output is 1 for preset time after input rising edge
- Creates a one-shot pulse for a fixed duration
- Used for strobe lights, alarm pulses, single actuations

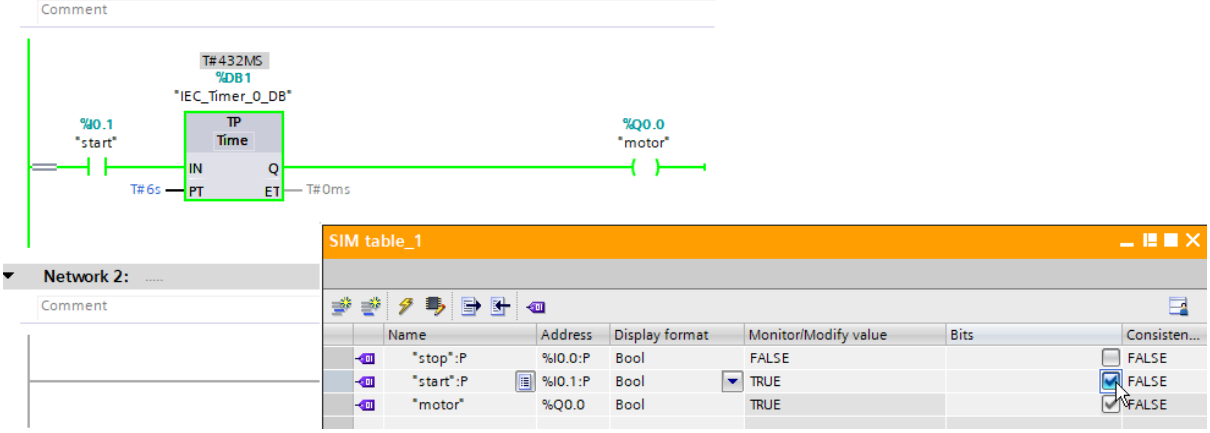


Figure 43 TP is used to generate a time delay after a specific condition or event occurs

On-Delay (TON) Timer

The on-delay timer accumulates time while input is 1. Output sets after the preset time. Key features:

- Accumulates time while input is high
- Output sets after preset time elapsed
- Creates a turn-on delay after input trigger

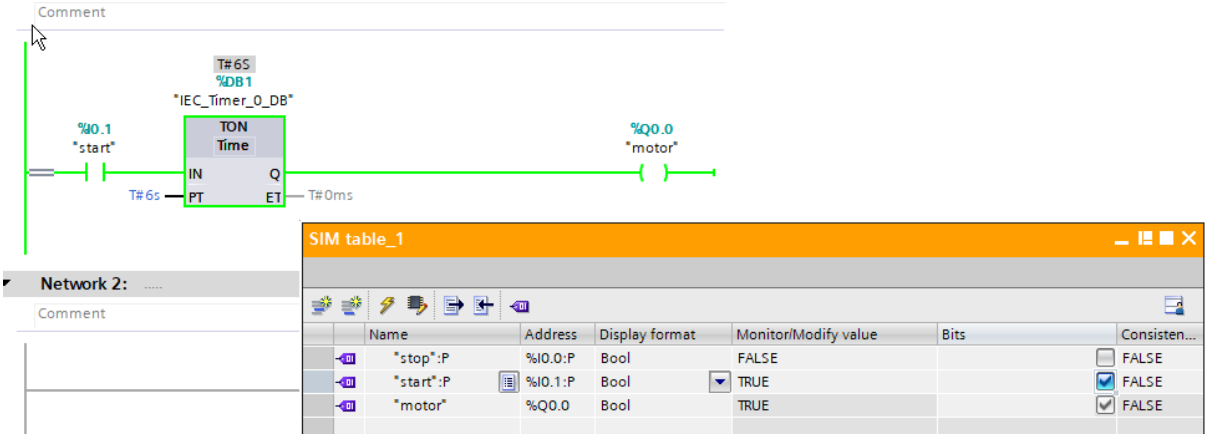


Figure 44 TON (Pulse Timer) responds to the ON condition and generates a brief pulse after a specified time delay when the condition becomes true.

Off-Delay (TOF) Timer

The off-delay timer accumulates time while input is 0 (from 0 to 1). Output resets after preset elapsed. Key features:

- Accumulates time while input is low
- Output resets after preset time elapsed
- Creates a turn-off delay for output deactivation

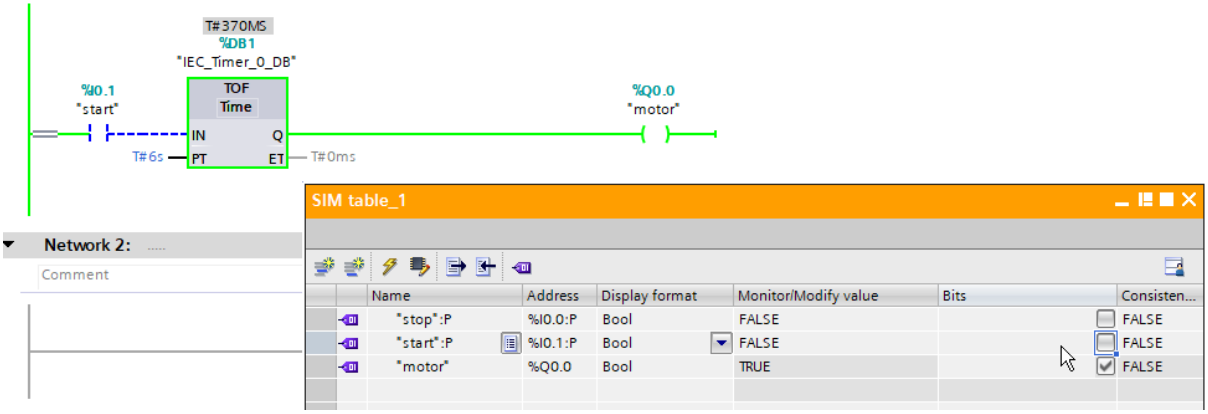


Figure 45 TOF (On-Delay Timer) responds to the OFF condition and generates an output signal after a specified time delay when the condition becomes false.

TONR Timer

The TONR timer is a retentive on-delay timer that continues timing from its accumulated value, even when power is cycled. This prevents the need to restart from zero every time the PLC restarts.

Key behaviors:

- Accumulates time while input is 1
- Output sets after preset time
- Retains elapsed time on power loss
- Resets only when R input is 1

The retentive feature provides continuity of timing through power cycles. This is useful for timers monitoring production durations, maintenance intervals, material cure times, and other operations requiring uninterrupted timing accumulation.

On restart, the TONR timer resumes from the stored elapsed time rather than resetting. The elapsed time value can also be preset on warm restart.

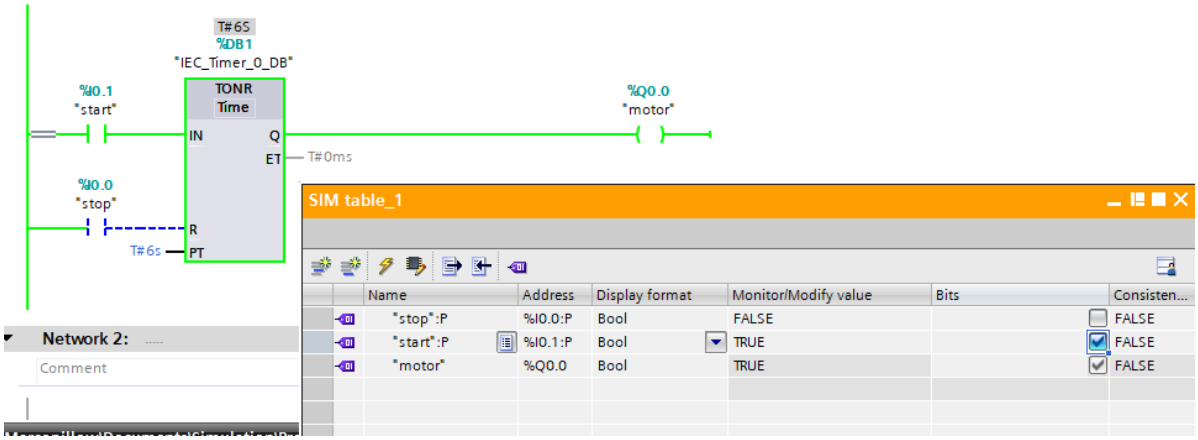


Figure 46 Using TONR can be useful in situations where we need to maintain timing information even when the input condition momentarily de-energizes.

Using a TON Timer to Create a Delayed Output Activation

Here is an example using an SR flip-flop to hold the output state of a TON on-delay timer to control activation of an output:

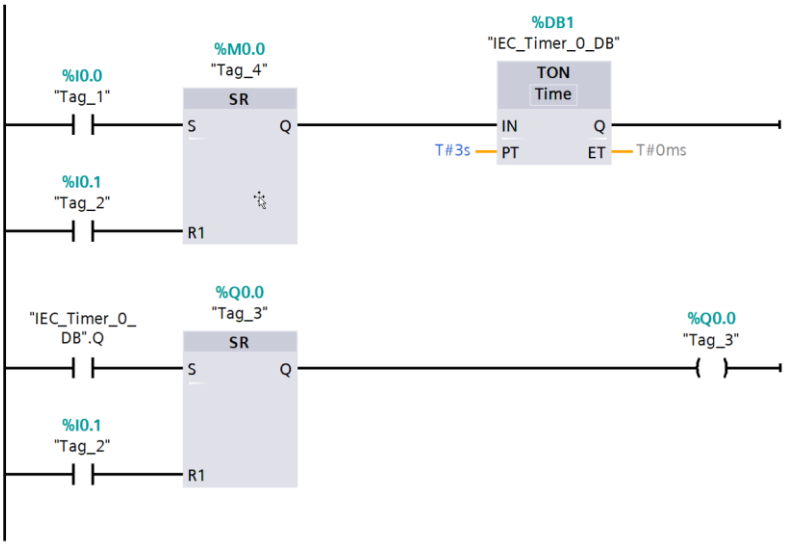


Figure 47 Using an SR flip-flop to hold the output state of a TON on-delay timer to control activation of an output

Counter Operations

Counters are function blocks that allow counting pulses, events, or signals in a PLC program. They are useful for tallying production quantities, operational cycles, detecting the number of sensor activations, and other counting tasks. The built-in counter instructions provide essential counting and tallying capabilities for PLC programs. They can track any type of event or signal in a wide variety of automation and control applications.

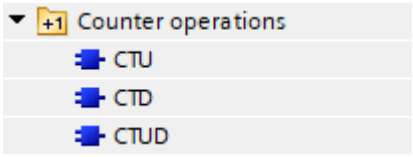


Figure 48 Counter Operation blocks

Up Counter (CTU)

The CTU increments on each positive edge when the input is 1. It counts up to a preset maximum value.

Example: Count the number of parts fabricated by a press machine.

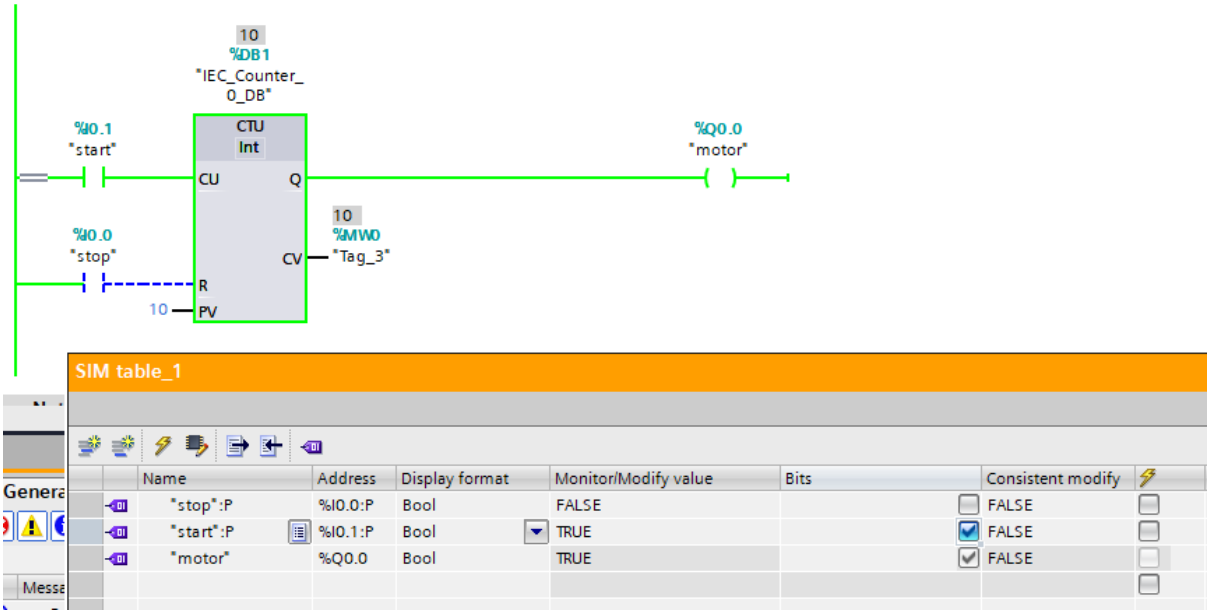


Figure 49 Using a CTU (Up Counter) Block to Count 10 Start Button Activations and Triggering Motor Activation

Down Counter (CTD)

The CTD decrements on each positive edge when the input is 1. It counts down from a preset starting value. So when the count reaches zero, it can trigger an output action.

Example: Count down the number of processing cycles remaining.

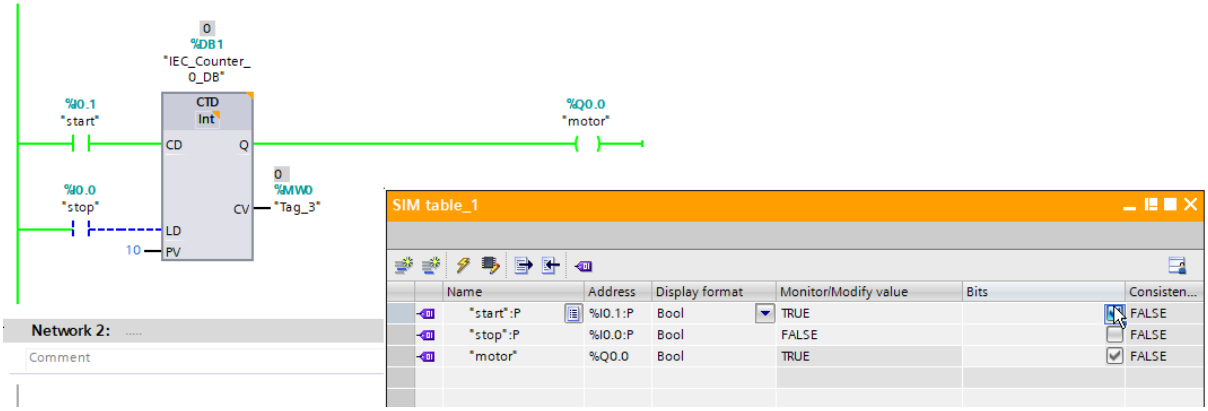


Figure 50 Using CTD to Trigger Motor Activation on Countdown Completion from 10 to 0.

Up/Down Counter (CTUD)

The CTUD acts as both an up and down counter. The direction is controlled by an up/down control input. Example: Counting people entering and exiting a room.

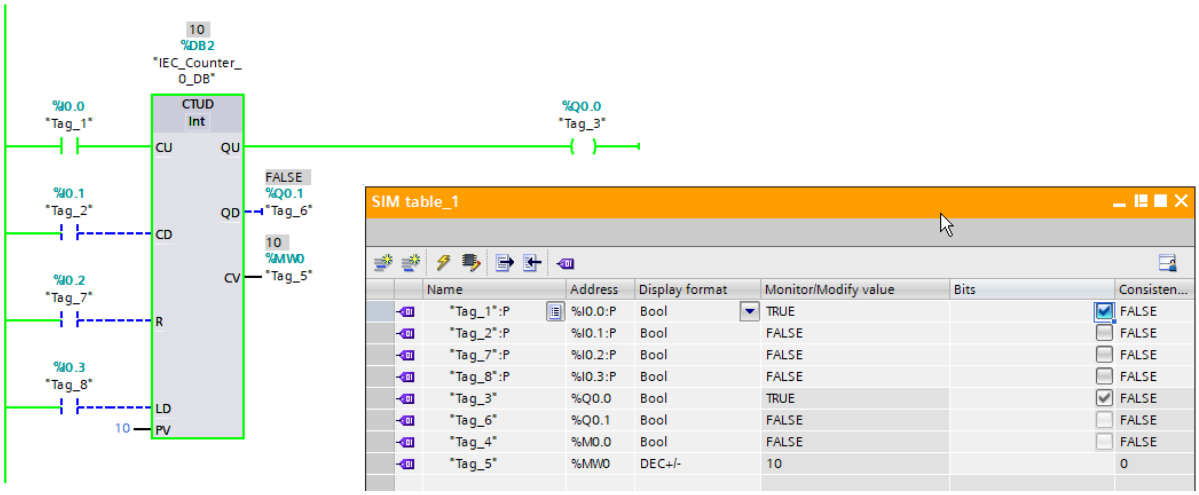


Figure 51 Employing CTUD for Incrementing and Decrementing Counts, Triggering Output after 10 Iterations.

Comparator Operations

Comparators are instructions that compare two numeric values or variables and return a TRUE or FALSE based on the comparison result. They are useful for evaluating analog signals, making control decisions, detecting thresholds, and performing general logic.

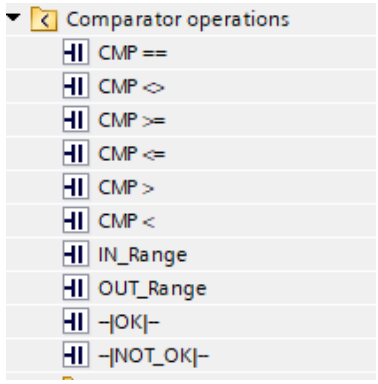


Figure 52 Comparator Operations blocks

Equal (EQU)
EQU returns TRUE if the two values are equal.
Example: Check if a temperature matches the setpoint.

Not Equal (NEQ)
NEQ returns TRUE if the two values are not equal.
Example: Detect an over-temperature condition.

Greater Than (GT)
GT returns TRUE if the first value is greater than the second.
Example: Check if a level exceeds a limit.

Greater Than or Equal (GE)
GE returns TRUE if the first value is greater than or equal to the second.
Example: Monitor if a process value is within acceptable range.

Less Than (LT)
LT returns TRUE if the first value is less than the second.
Example: Detect low power condition.

Less Than or Equal (LE)
LE returns TRUE if the first value is less than or equal to the second.
Example: Check if value is below limit.

IN_Range Counter Operation
The "IN_Range" counter operation is employed to count the occurrences of input events or conditions within a specified range. It allows the programmer to monitor and record the number of times an event falls within the defined range, providing valuable data for process control and analysis.
Example: The IN_Range counter can be used to count the number of bottles filled within a predefined acceptable volume range.

Out_Range Counter Operation
On the other hand, the "OUT_Range" counter operation is utilized to count the occurrences of input events or conditions outside a specified range. It serves as a valuable tool for identifying deviations from expected parameters, allowing for immediate intervention when necessary.
Example: The Out_Range counter can monitor instances when the temperature exceeds safe operating limits.

Math Functions

In the state of industrial automation and PLCs, mathematical operations are fundamental pillars that empower engineers and programmers to shape and control processes with precision. The TIA Portal software is offering a wide array of mathematical functions to

facilitate complex calculation and data manipulations. Among these functions, the most crucial ones encompass the "CALCULATE" function, along with the "ADD," "SUB" (Subtract), "MUL" (Multiply), "DIV" (Divide), "MOD" (Modulus), "ABS" (Absolute), and "FRAC" (Fraction) operations.

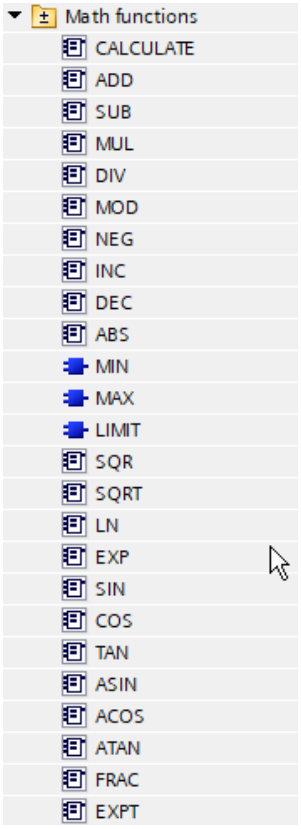


Figure 53 Math functions blocks

CALCULATE Function

The "CALCULATE" function, a versatile asset in the TIA Portal software, empowers programmers to execute diverse mathematical calculations based on user-defined logic.

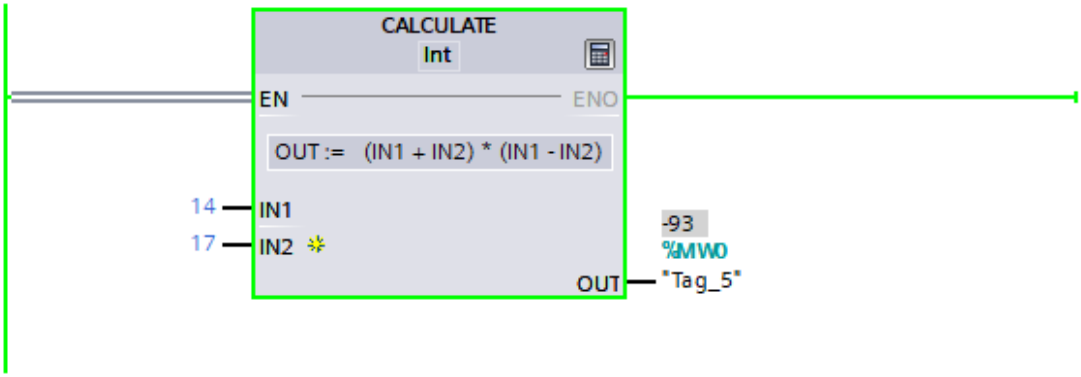


Figure 54 Using CALCULATE for different calculations

ADD, SUB, MUL, and DIV Operations

The quartet of "ADD," "SUB," "MUL," and "DIV" functions represents the foundational arithmetic operations essential for numeric manipulation.

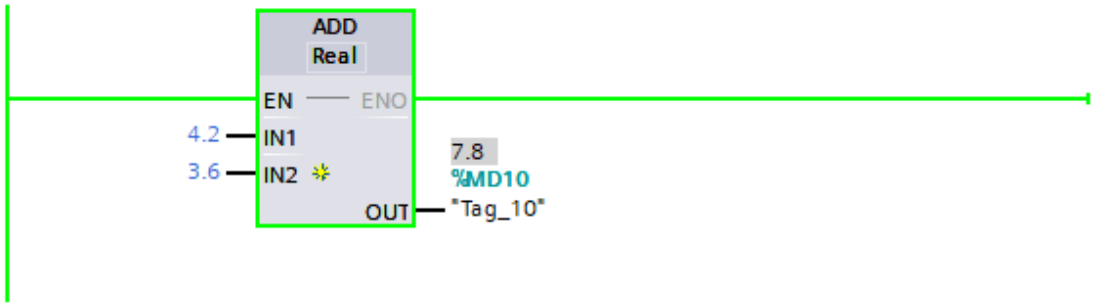


Figure 55 Add function

MOD Function

The "MOD" (Modulus) function, akin to a mathematical compass, calculates the remainder when one value is divided by another.

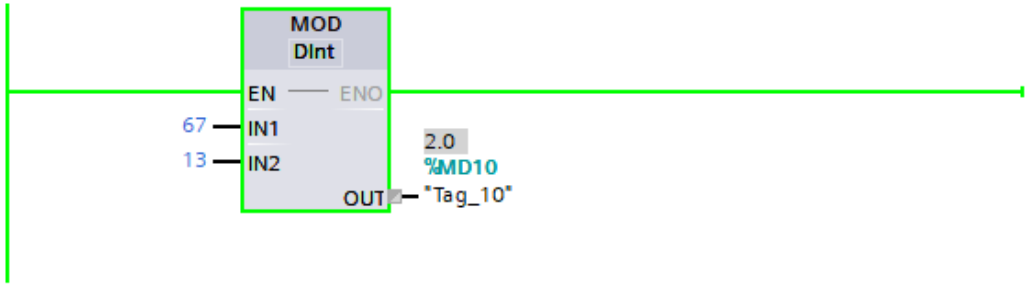


Figure 56 MOD function

ABS Function

In a world where directionality can be inconsequential, the "ABS" (Absolute) function provides clarity by returning the positive magnitude of a number. Often used to normalize values, the "ABS" function is indispensable in scenarios where only the magnitude matters, regardless of direction.

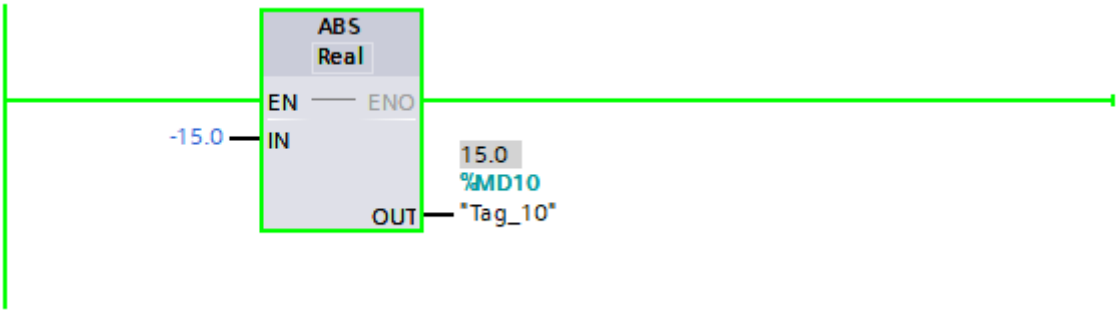


Figure 57 ABS function

FRAC Function

Precision in mathematical representation is a hallmark of the "FRAC" (Fraction) function, which extracts the fractional part of a real number. This operation is invaluable when dealing with analog signals or when precise measurements are required in industrial contexts.

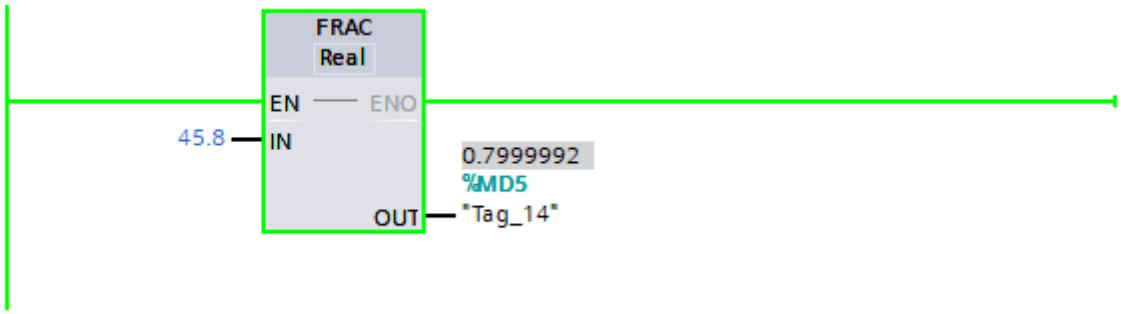


Figure 58 FRAC function

Move operation

The MOVE operation in TIA Portal software allows for the seamless transfer of data within a program. It plays a crucial role in ensuring that information is accurately and efficiently relayed between variables, memory locations, and registers. This fundamental operation is essential for data manipulation, transformation, and the precise control of industrial automation processes.

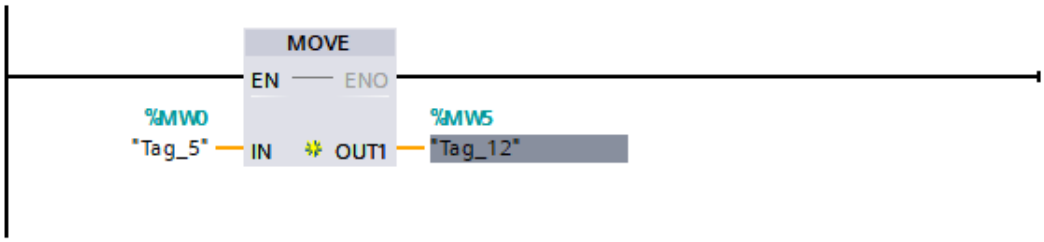


Figure 59 "MOVE" operation is used to transfer data from one location or memory area to another within a PLC.

Conversion operations

Conversion functions allow transforming data between different numeric formats and data types in PLC code. They provide flexibility when dealing with the variety of number representations used in industrial systems.

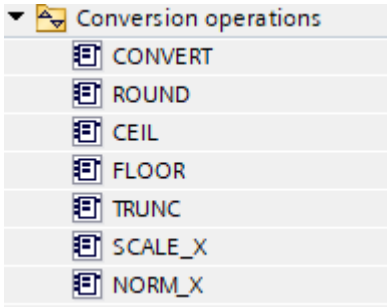


Figure 60 Conversion Operation blocks

Convert Operation

The CONVERT instruction changes data between types like integer, floating point, BCD, strings, etc. This is useful when data needs adaptation between blocks and modules.

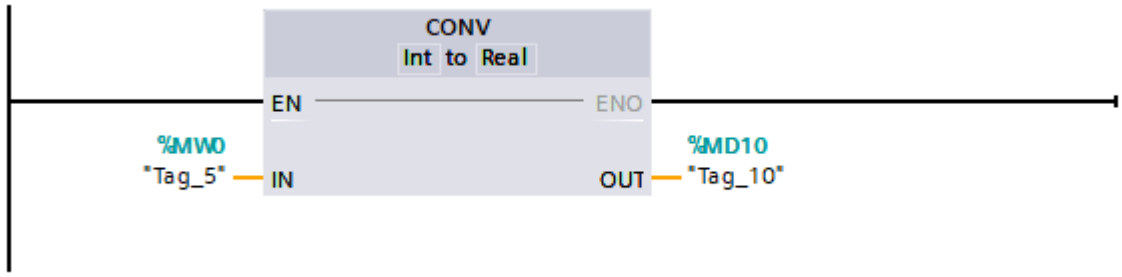


Figure 61 Employing the CONV Function for Integer to Real Data Type Conversion.

Round, Floor, CEIL and Trunc Operation

Rounding functions like ROUND, FLOOR, and CEIL alter the precision of real number values by constraining them to integer or fixed decimal resolutions. Rounding helps reduce noise and normalize sensor readings.

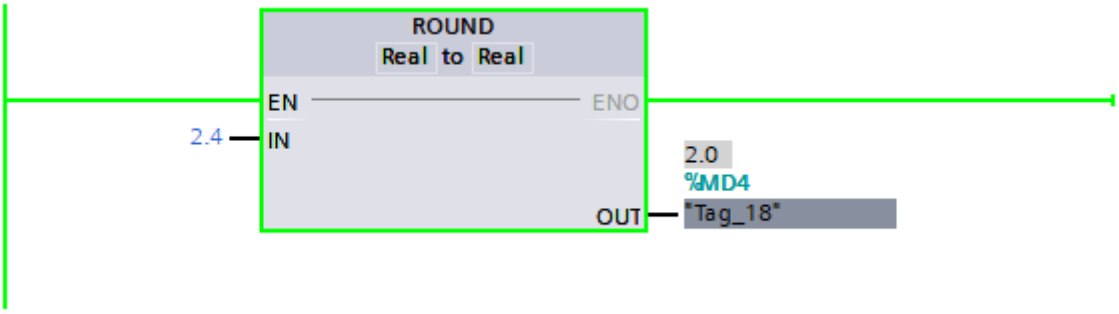


Figure 62 Utilizing the ROUND Function to Achieve Rounding from 2.4 to 2.

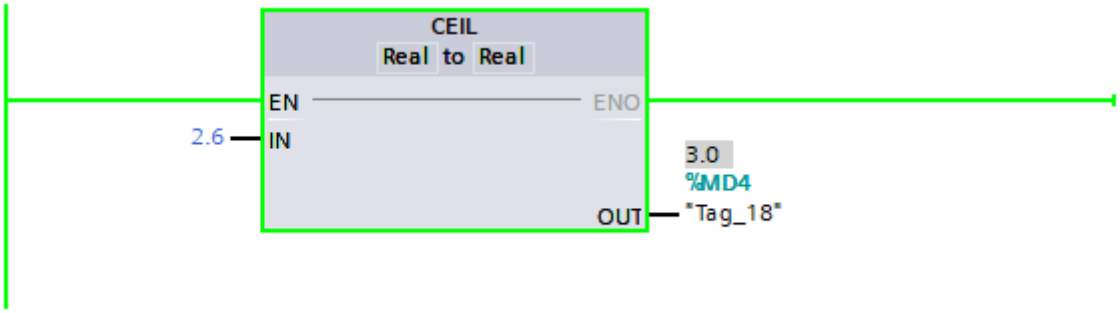


Figure 63 Applying the CEIL Function for the Rounding of 2.6 to 3.

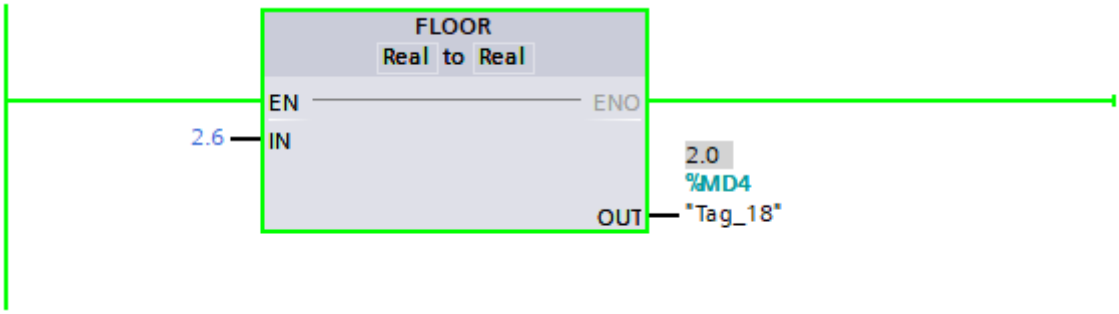


Figure 64 Employing the FLOOR Function for the Purpose of Rounding 2.6 Down to 2.

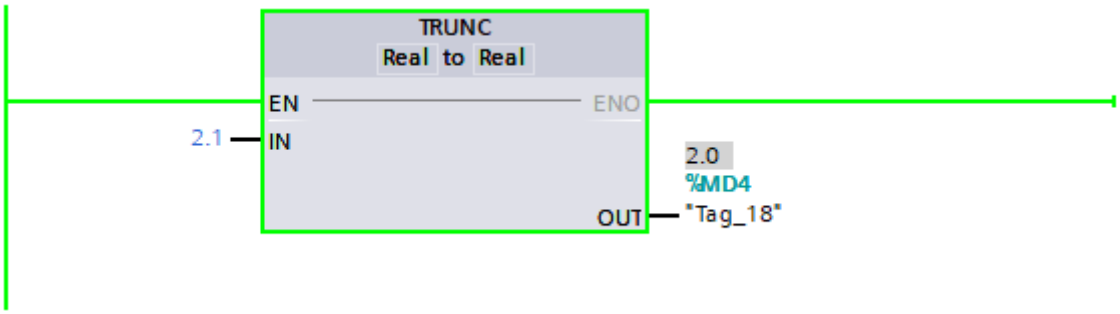


Figure 65 Applying the TRUNC Function to Effectively Round 2.1 Down to 2

Scale_X and Norm_X operations

Scale_X and Norm_X are very important and they can be used for analog inputs. In TIA Portal, the "SCALE_X" and "NORM_X" functions are part of the Conversion Operations, and they are used to scale and normalize values, respectively. These functions are particularly useful when you need to convert data from one range or unit of measurement to another within a PLC program.

Program Control Operations

Here the focus is on two essential components: the "Jump" and "Label" blocks. These elements play a pivotal role in orchestrating the flow of a PLC program, allowing engineers to create flexible and efficient control structures that respond dynamically to real-world conditions and requirements. The "Jump" block allows for conditional and unconditional transitions from one network to another, while the "Label" block serves as a pivotal reference point, marking the network to which the "Jump" operation will navigate.

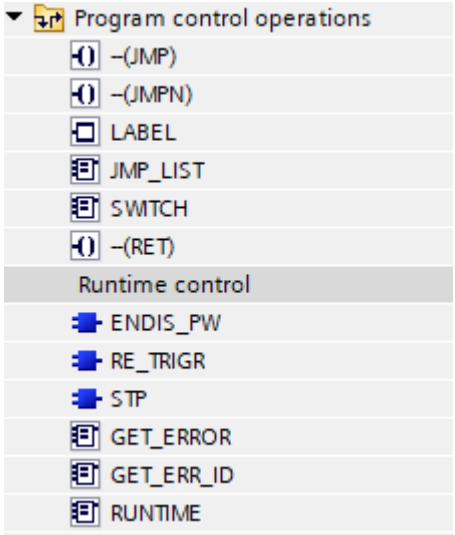


Figure 66 Program Control Operation blocks

Example of using jump and label

An illustrative application of the "Jump" and "Label" blocks entails a scenario featuring three distinct motors, wherein precise control and conditional sequencing are imperative. In this context, should the requirement arise to activate the first motor, with subsequent engagement

of motors 2 and 3 after a defined delay of three seconds, program control operations are crucial. Nevertheless, contingencies may necessitate the independent activation of only the third motor, necessitating the exclusion of motor 2 from activation. The strategic utilization of "Jump" and "Label" blocks within this framework ensures that the operational state of motor 2 remains unaltered, thereby facilitating a nuanced and selective approach to motor control. (So, we want to jump from Network 4)

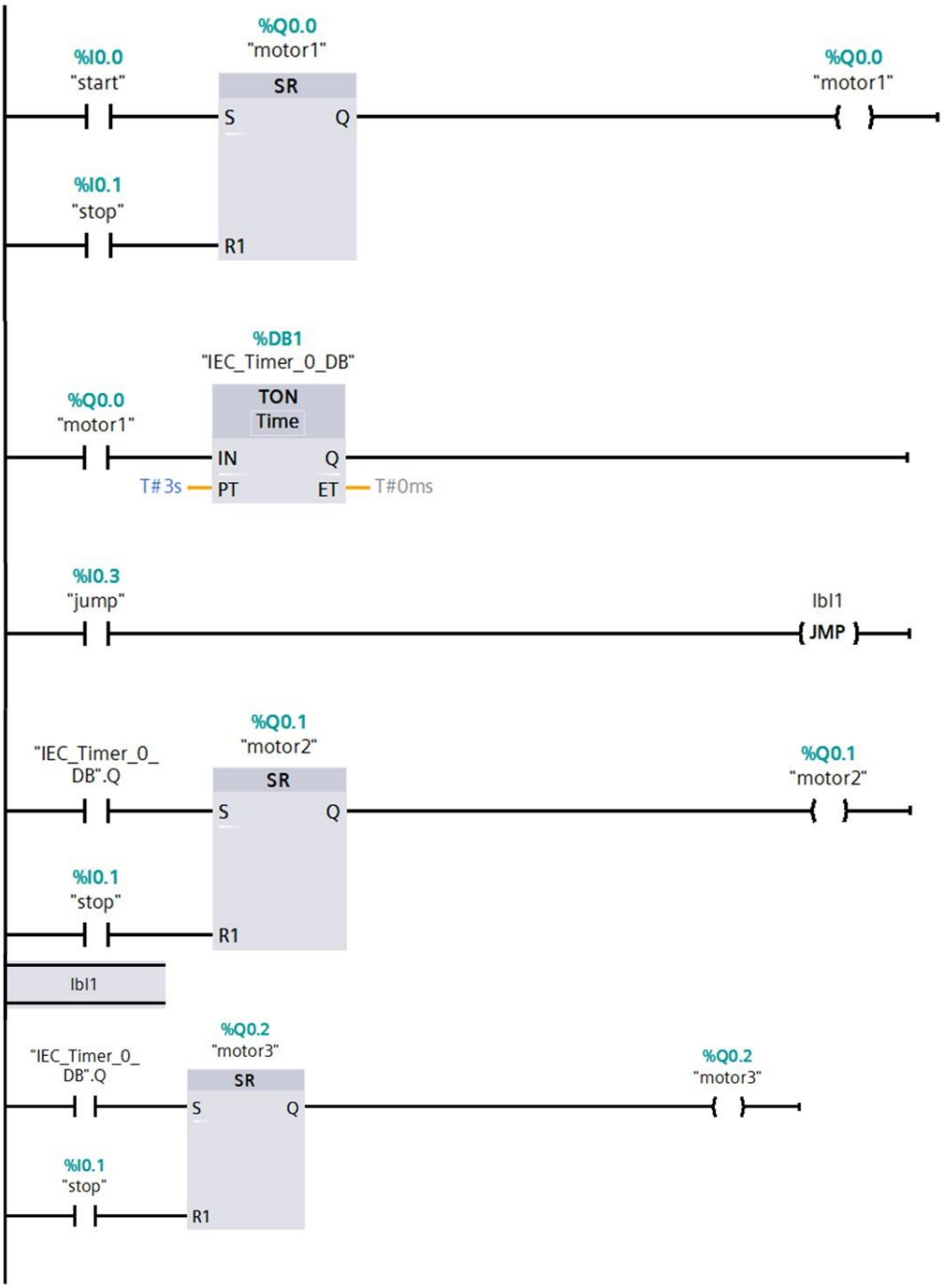


Figure 67 In an illustrative ladder diagram, the 'Jump' and 'Label' blocks play a pivotal role in controlling three distinct motors with precision. This includes activating the first motor and then engaging motors 2 and 3 with a three-second delay if needed. However, there are contingencies where only the third motor needs activation, allowing motor 2 to remain unaffected. The strategic use of 'Jump' and 'Label' blocks in Network 4 enables selective motor control.

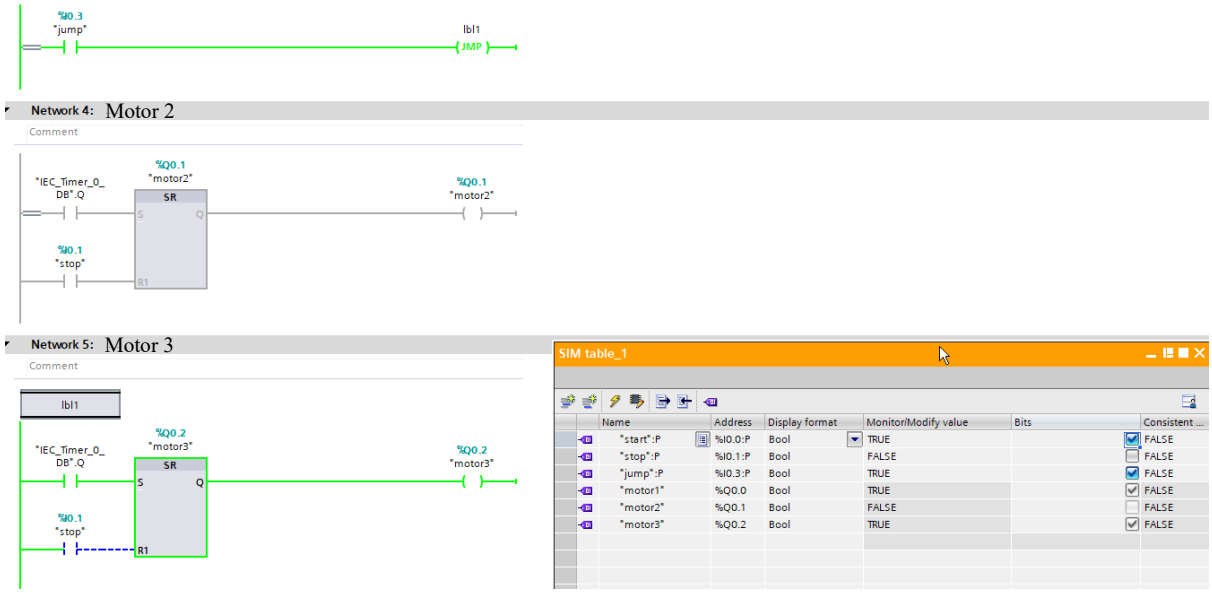


Figure 68 Simulation: Activating 'Jump' for selective motor control for isolating the activation of second motor and jumping to the third motor.

Word logic operations

Word logic instructions allow manipulating integer words and double words in PLC programs. They perform bitwise logical operations between two word-sized values.

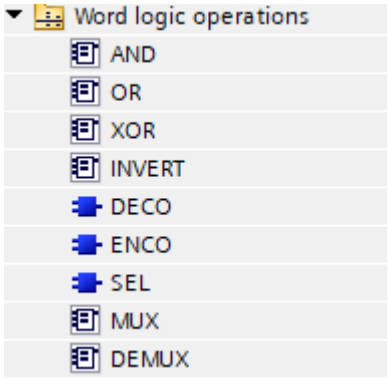


Figure 69 Word logic operation blocks

The AND instruction performs a bitwise Boolean AND, setting the output bit only if both input bits are 1.

The OR instruction performs a bitwise Boolean OR, setting the output bit if either input bit is 1.

The XOR (exclusive OR) instruction sets the output bit only when one corresponding input bit is 1, but not both.

The INVERT instruction flips all the bits of the input word, acting as a bitwise NOT operation.

These word logic blocks enable lower-level manipulation of integer values on a bit-by-bit basis. Operations like masking, toggling, and binary decoding can be implemented more efficiently using word logic.

Shift and Rotate operations

TIA Portal, a leading engineering framework for Siemens automation devices, provides several instructions for bit manipulation, including Shift Left (SHL), Shift Right (SHR), Rotate Left (ROL), and Rotate Right (ROR). These operations are fundamental for handling binary data within PLC programming.

The SHL operation shifts the bits of a binary number to the left by a specified number of places. This operation is often used in creating sequencers in ladder logic. Conversely, the SHR operation shifts the bits of a binary number to the right. There are two types of shifting right: Logical and Arithmetic. Logical fills zeros in the new empty most significant bit (MSB), while Arithmetic fills ones or zeros with respect to what the MSB was.

The ROL and ROR operations involve rotating the bits of a binary number to the left or right, respectively. In a rotate operation, unlike a shift operation, the bit that gets pushed out on one end reappears at the other end.

These operations are particularly useful in PLCs for various applications. For instance, shift registers are commonly used in conveyor systems where products are tracked along various sections. Here, each bit in a shift register can represent a section of the conveyor, and as products move along the conveyor, the corresponding bits are shifted.

In the context of a Shift Left (SHL) operation, when we shift an integer value by a certain number of positions to the left (N positions), it effectively multiplies the integer by 2 raised to the power of N.

In this scenario:

- Input (integer): 12
- N (number of positions to shift): 2

The output can be calculated as follows: Output = Input * 2^N Output = 12 * 2^2 Output = 12 * 4 Output = 48

So, the output of the SHL operation with an input of 12 (integer) and shifting it 2 positions to the left (N = 2) would be 48.

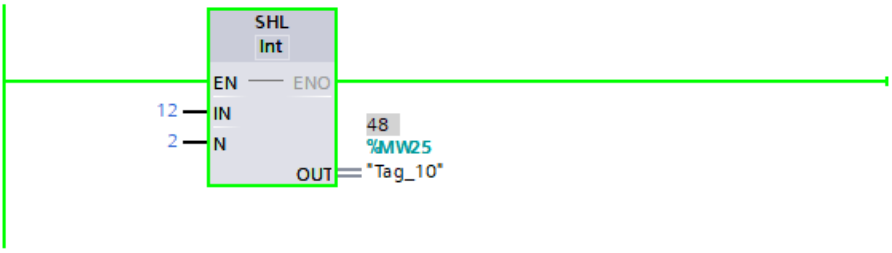


Figure 70 In SHL Operations: Shifting an integer left by N positions multiplies it by 2^N. For instance, with an input of 12 and N = 2, the result is 48.

Difference between shift and rotate

The primary difference between shift and rotate operations lies in how they handle the bit that is “pushed out” of the data word.

In shift operations (like SHL or SHR), the bit that is pushed out — either from the left end (most significant bit) or the right end (least significant bit) — is discarded. The vacant position is then filled with a zero.
On the other hand, in rotate operations (like ROL or ROR), the bit that is pushed out from one end reappears at the other end. This means that no information is lost during a rotate operation, which can be crucial in certain applications.
In summary, while both shift and rotate operations can change the position of bits in a data word, rotate operations preserve all original bits, while shift operations do not.

Understanding Blocks in TIA Portal Software

A key feature of TIA Portal software is its use of blocks, which provide a structured way to organize code and data. There are four main types of blocks in the TIA Portal software: Organization Blocks (OBs), Function Blocks (FBs), Functions (FCs), and Data Blocks (DBs).

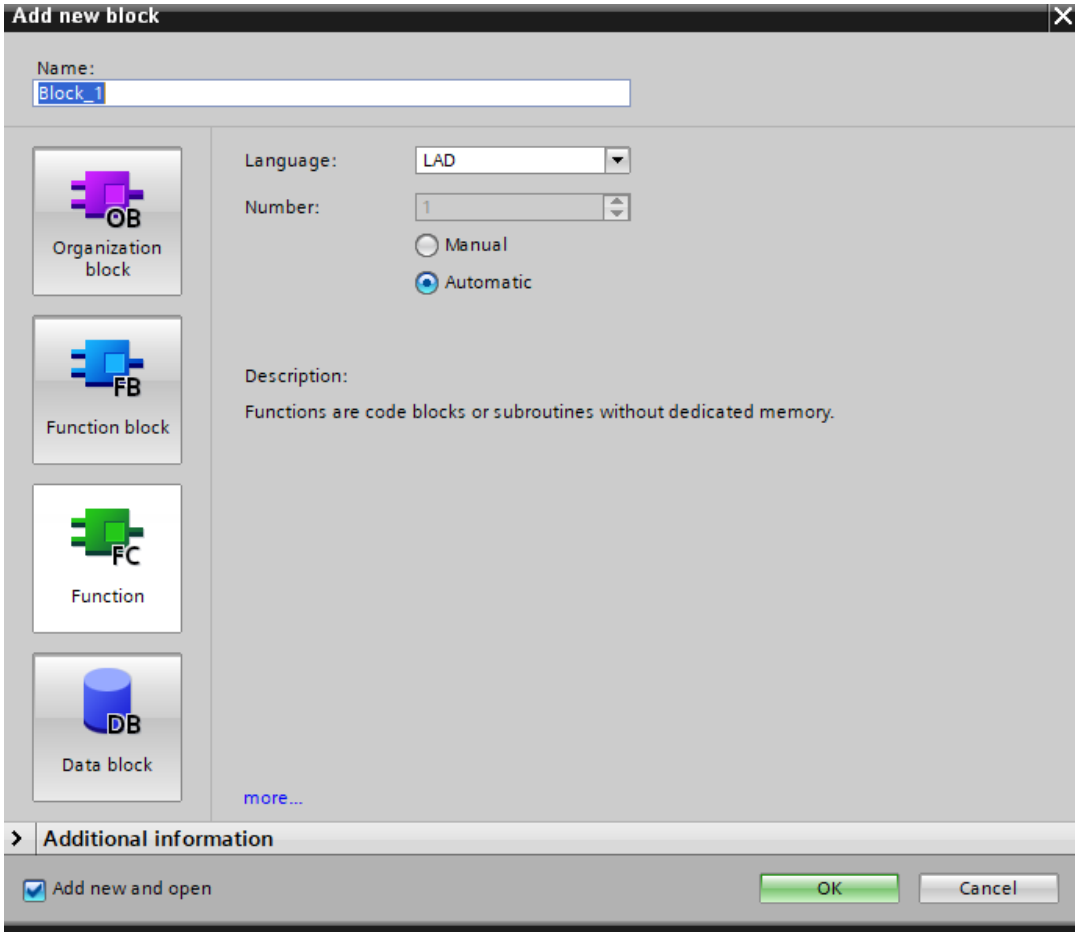


Figure 71 Add blocks window in TIA portal

Organization Blocks (OBs)

This serves as the interface between the operating system of the PLC and the user program. They are triggered by the operating system, either cyclically or when a specific event occurs.

The most commonly used OB is OB1, which is responsible for the cyclic execution of the user program.

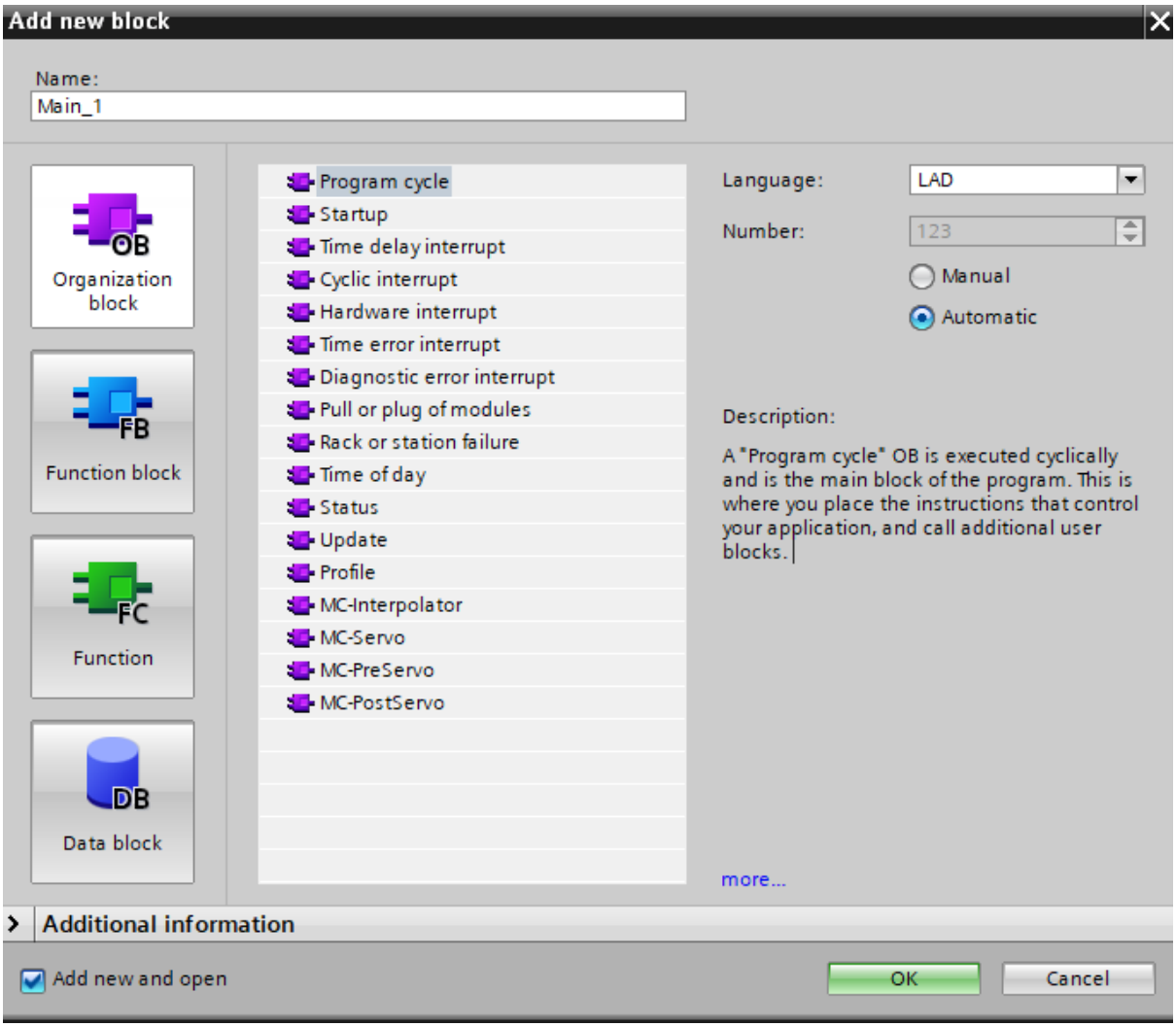


Figure 72 Organization blocks

Function Blocks (FBs)

These blocks are reusable blocks that encapsulate a specific functionality. Unique to FBs is their ability to retain their own memory, allowing them to maintain state information between calls. This makes them ideal for tasks that require persistent data. To create a new FB, one can navigate to the “Program Blocks” folder in the project, double-click the “Add new block” command, and then select “Function block (FB)”.

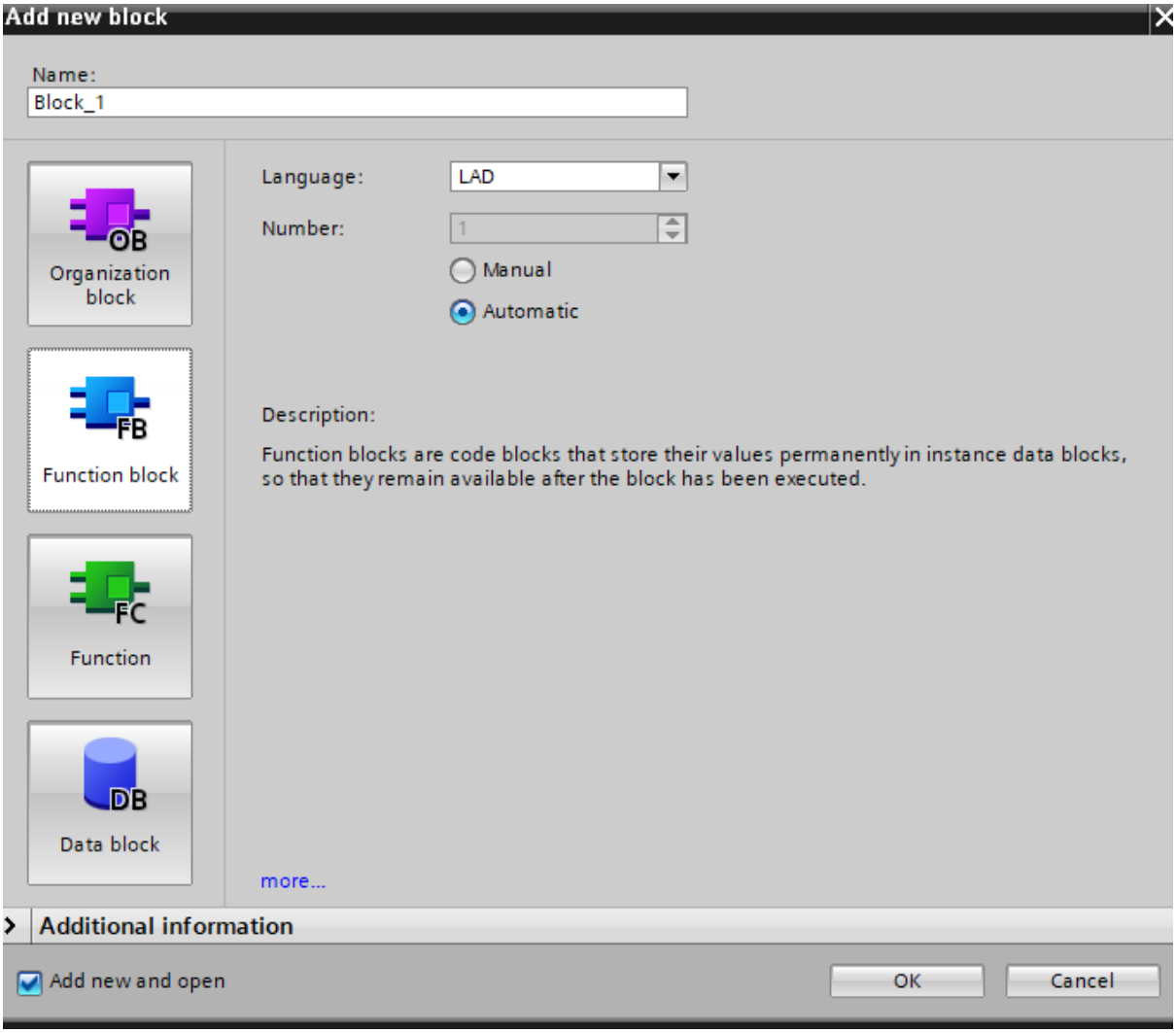


Figure 73 Function block window

Here's how Function Blocks work in TIA Portal:

1. **Definition**: Function Blocks are reusable blocks of code that encapsulate a specific set of functionality. They are defined with input and output parameters, making them versatile and modular.
2. **Parameterization:** FBs can be parameterized, which means we can configure them by setting input parameters to specific values. This allows us to reuse the same FB for various purposes by adjusting its parameters.
3. **Reuse:** Function Blocks promote code reuse. Once we have created an FB for a specific function, we can use it in multiple parts of our program or in different projects, saving development time and reducing the chance of errors.
4. **Encapsulation:** FBs encapsulate functionality, hiding the implementation details from the main program. This enhances program readability and makes it easier to maintain and troubleshoot.
5. **Integration:** TIA Portal allows us to integrate Function Blocks into our ladder logic, structured text, or other programming languages supported by the platform.

6. **Testing and Debugging:** FBs can be individually tested and debugged, making it easier to isolate and fix issues within specific functional blocks without affecting the entire program.

Below, we present an illustrative instance of FB implementation. The function incorporates two motors, a start button, a stop button, and a timer block. Upon activation of the first motor, the second motor initiates operation after a predefined delay.

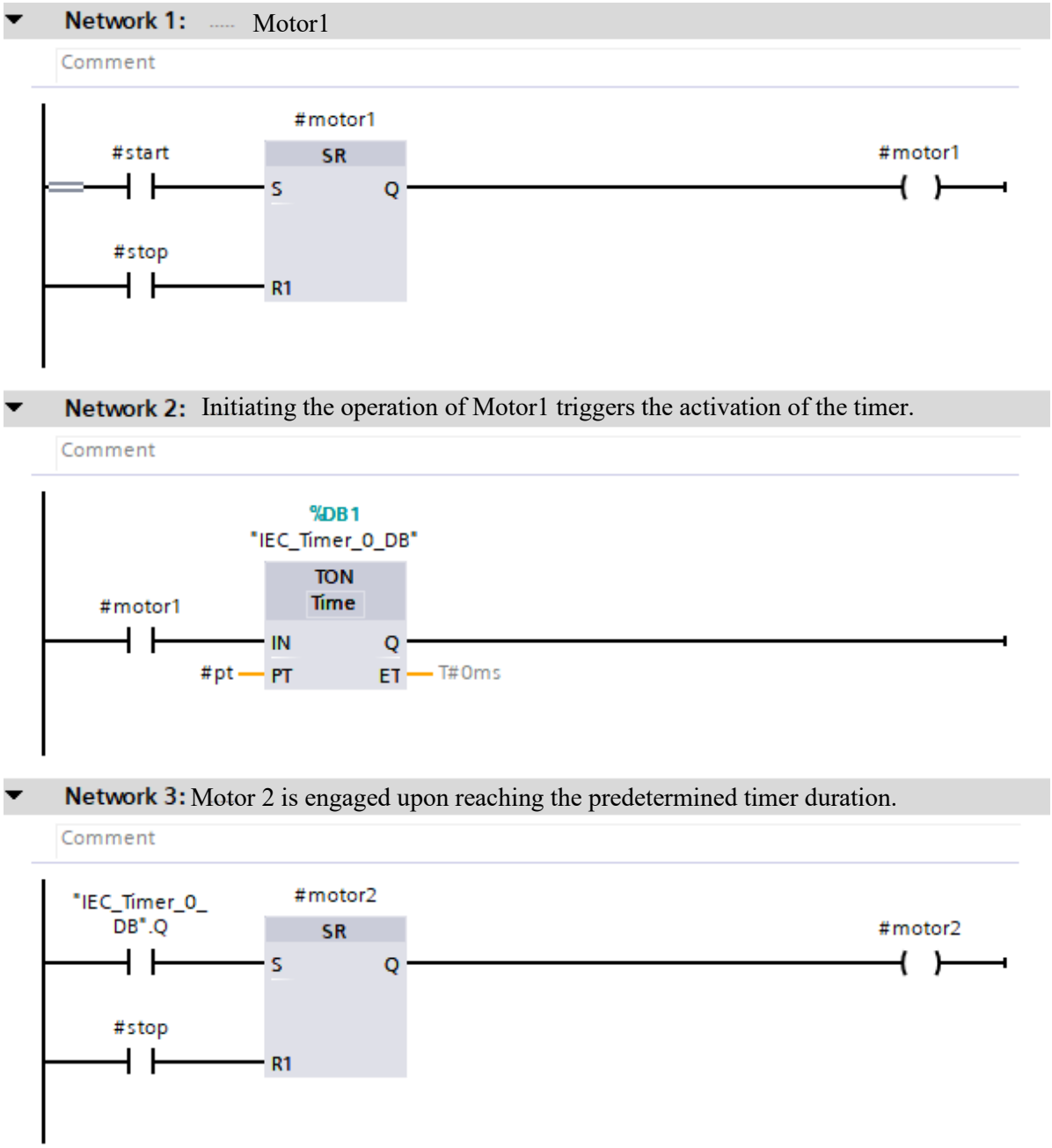


Figure 74 Example of FB Implementation: Utilizing two motors, a start button, a stop button, and a timer block to initiate second motor operation with a predefined delay.

In the table presented below, a comprehensive parameterization is provided, encompassing both input and output parameters. The inputs encompass the start button, stop button, and 'pt' (time), while the outputs include 'Motor1 and 'Motor2

Block_2

	Name	Data type	Default value	Retain	Accessible f...	Writa...	Visible in ...	Setpoint
1	Input				☐	☐	☐	☐
2	start	Bool	false	Non-ret...	☑	☑	☑	☐
3	stop	Bool	false	Non-retain	☑	☑	☑	☐
4	pt	Time	T#0ms	Non-retain	☑	☑	☑	☐
5	Output				☐	☐	☐	☐
6	motor1	Bool	false	Non-retain	☑	☑	☑	☐
7	motor2	Bool	false	Non-retain	☑	☑	☑	☐

Figure 75 Parameterization of FBs enables versatile reuse by adjusting input values. Pt is time which its data type is time.

When we want to add the FB1 in the main because it is FB then we need to make data block as well. So the Tia portal will ask us to create a data block for the function block that we are adding to the main [OB1].

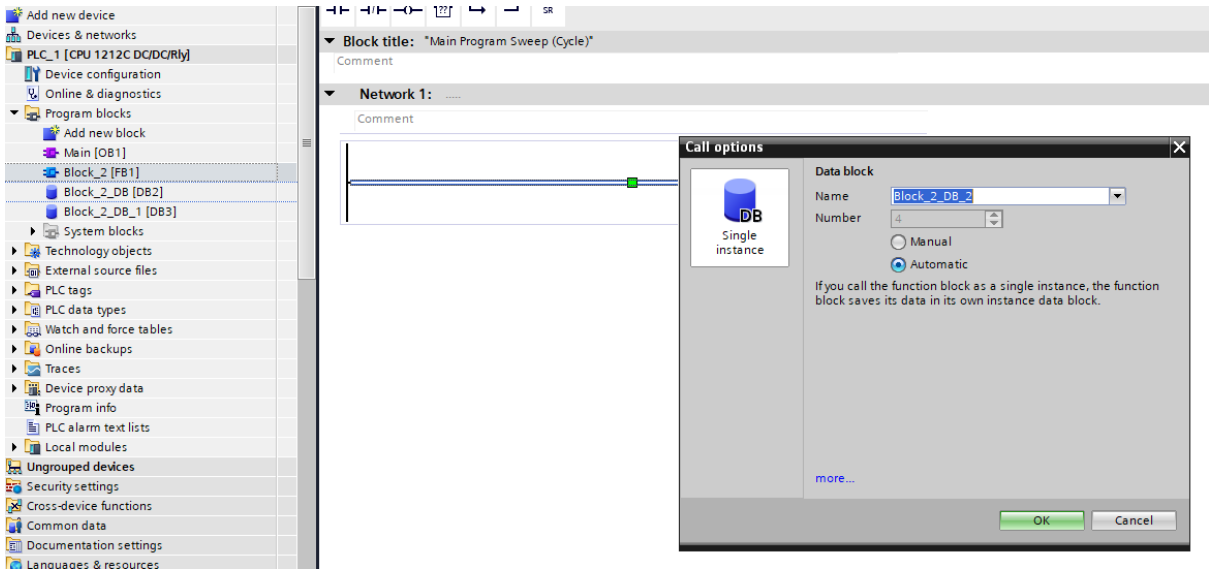


Figure 76 When incorporating FB1 as a function block into the main [OB1], a data block creation request is prompted by TIA Portal.

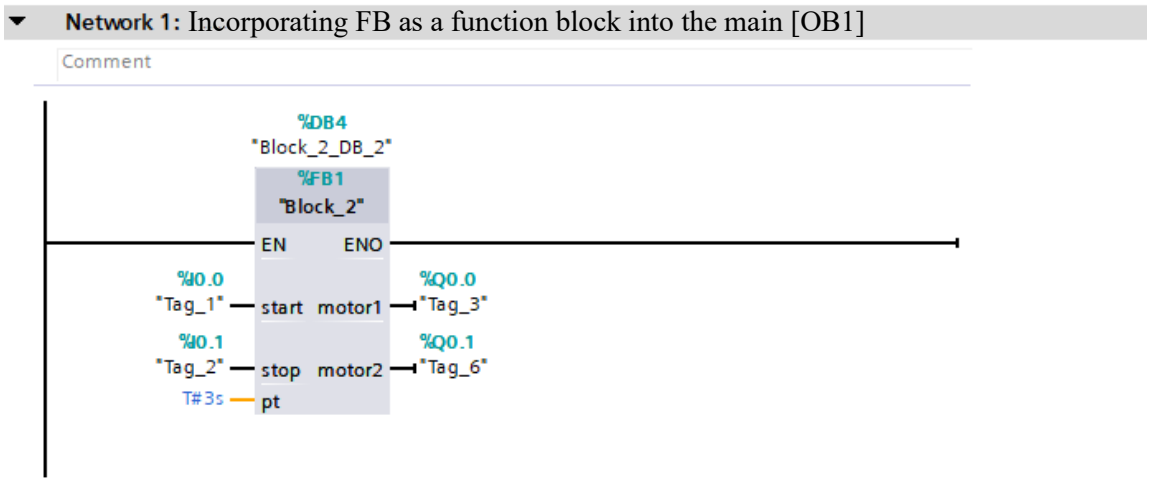


Figure 77 Integrating FB into [OB1] as a function block.

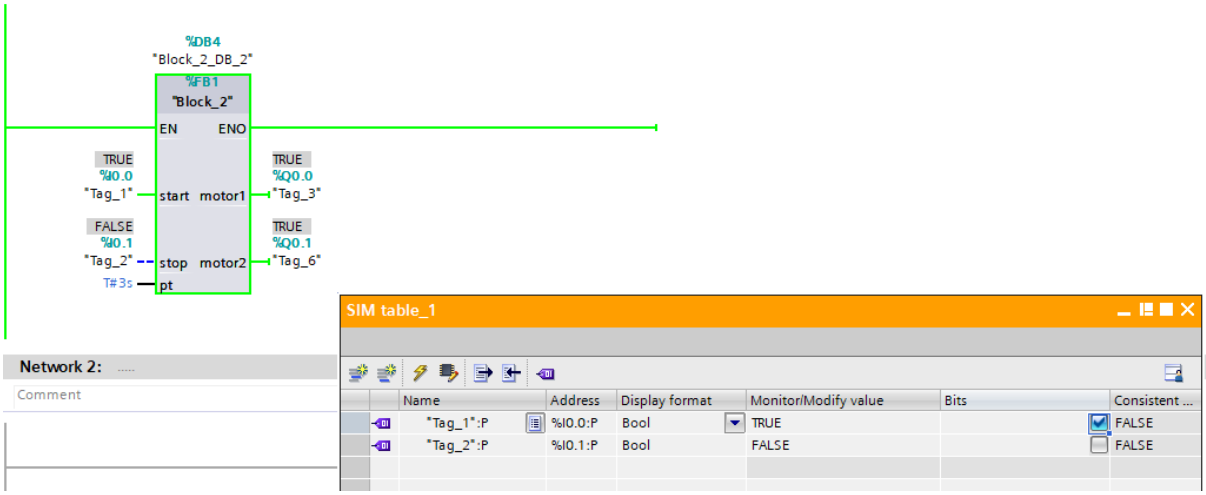


Figure 78 Simulation with Function Block in the main [OB1]

Functions (FCs)

These blocks do not have their own memory and cannot retain values between calls. They are similar to FBs in that they encapsulate a specific functionality. Functions in PLC programming are blocks of code that perform certain operations. They can be used for a variety of tasks, including converting values, adding values, subtracting values, multiplying values, dividing values, and finding the square root of a value. These mathematical functions are fundamental for handling data within PLC programming.

Considering a simple conveyor system where a conveyor belt runs to transfer a product to the end of the line. The conveyor will run when a start signal is available after a presence sensor detects the product on the Belt. The conveyor will start running and continue to run until the product is transferred to the end of the line and a detection sensor detects the product, then the belt will stop. In this example, coding this process will be done easily with a few logic code steps.

However, if we have many different conveyors in our process that need to be controlled in the same manner, it would be best to use functions to make our PLC coding structure easier to read and understand. This also takes advantage of function reusability to reduce the size of our code.

To create a new FC, one can follow the same steps as creating a new FB, but select “Function (FC)” instead.

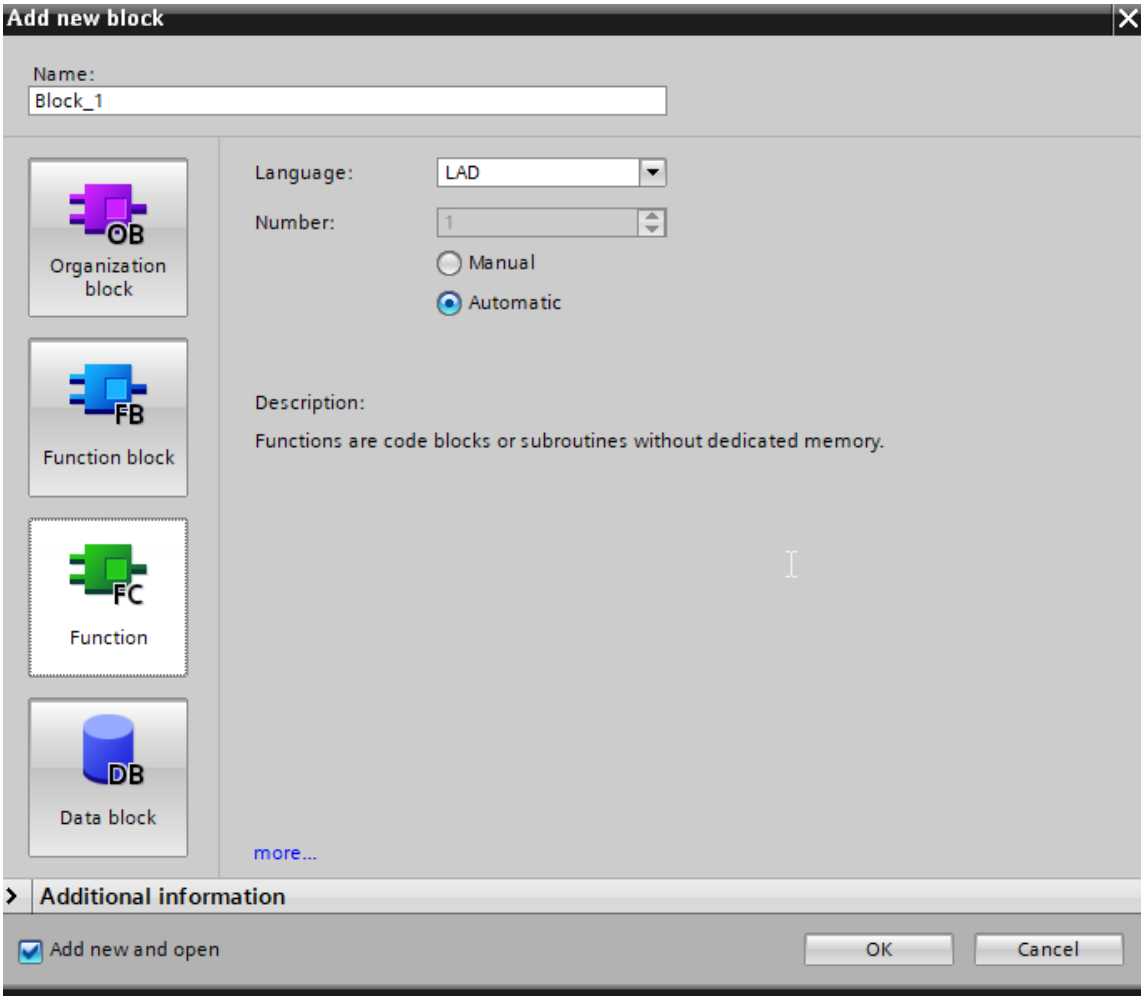


Figure 79 Function window - Functions are code blocks or subroutines without dedicated memory.

Here's how functions work in TIA Portal:

1. **Definition:** Functions are created by defining a set of instructions or calculations within a software module. These instructions can perform specific tasks or return values based on their inputs.
2. **Input and Output:** Functions typically have input parameters (arguments) that we provide when we call the function. These parameters are used as input data for the calculations or actions performed by the function. Functions also have return (output) values, which are the results of their computations.
3. **Reusability:** Once defined, functions can be reused throughout your PLC program. This promotes modular programming and reduces redundancy, as we can use the same function in multiple parts of our program.
4. **Encapsulation:** Functions encapsulate specific functionality, making our code more organized and easier to understand. The details of how a function works are hidden within the function itself, allowing we to focus on the higher-level logic in our program.

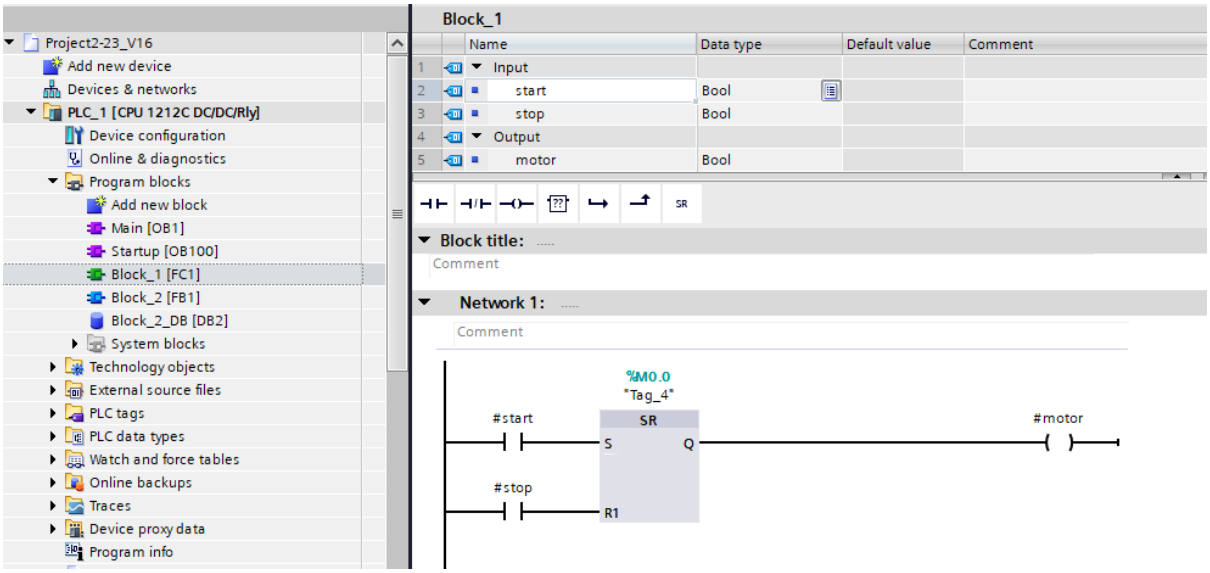


Figure 80 Implementing a Function to Control Motor Activation and Deactivation.

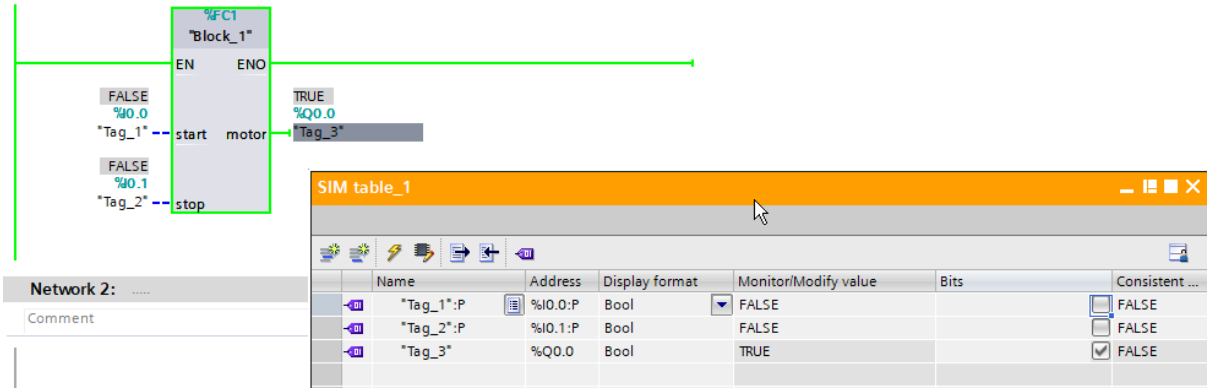


Figure 81 Integrating the Function into [OB1] for Program Use.

Comparing Function Blocks (FBs) and Functions (FCs) in TIA Portal

The primary distinction between a Function Block (FB) and a Function (FC) within TIA Portal software lies in the presence of memory. Function Blocks possess memory, allowing them to store input, output, and in-out parameters in instance data blocks, ensuring their persistence after execution. Conversely, Functions lack instance data memory, necessitating the assignment of actual values to all parameters upon invocation.

Table 4 Key differences between function blocks and functions in TIA Portal

Feature	Function Block	Function
Memory	Yes	No
Instance data block	Yes	No
Parameter assignment	Can be left blank when calling the block	Must be assigned actual values when calling the block
Suitable for	Storing stateful information and implementing complex functions	Performing stateless calculations and simple functions

Data Blocks (DBs)

They are used to store data in a structured manner. They allow for grouping related data together, thereby facilitating easier management and access. For example, for timers and counters the data blocks must be used. To create a new DB, one can navigate to the “Program Blocks” folder in the project, double-click the “Add new block” command, and then select “Data block (DB)”.

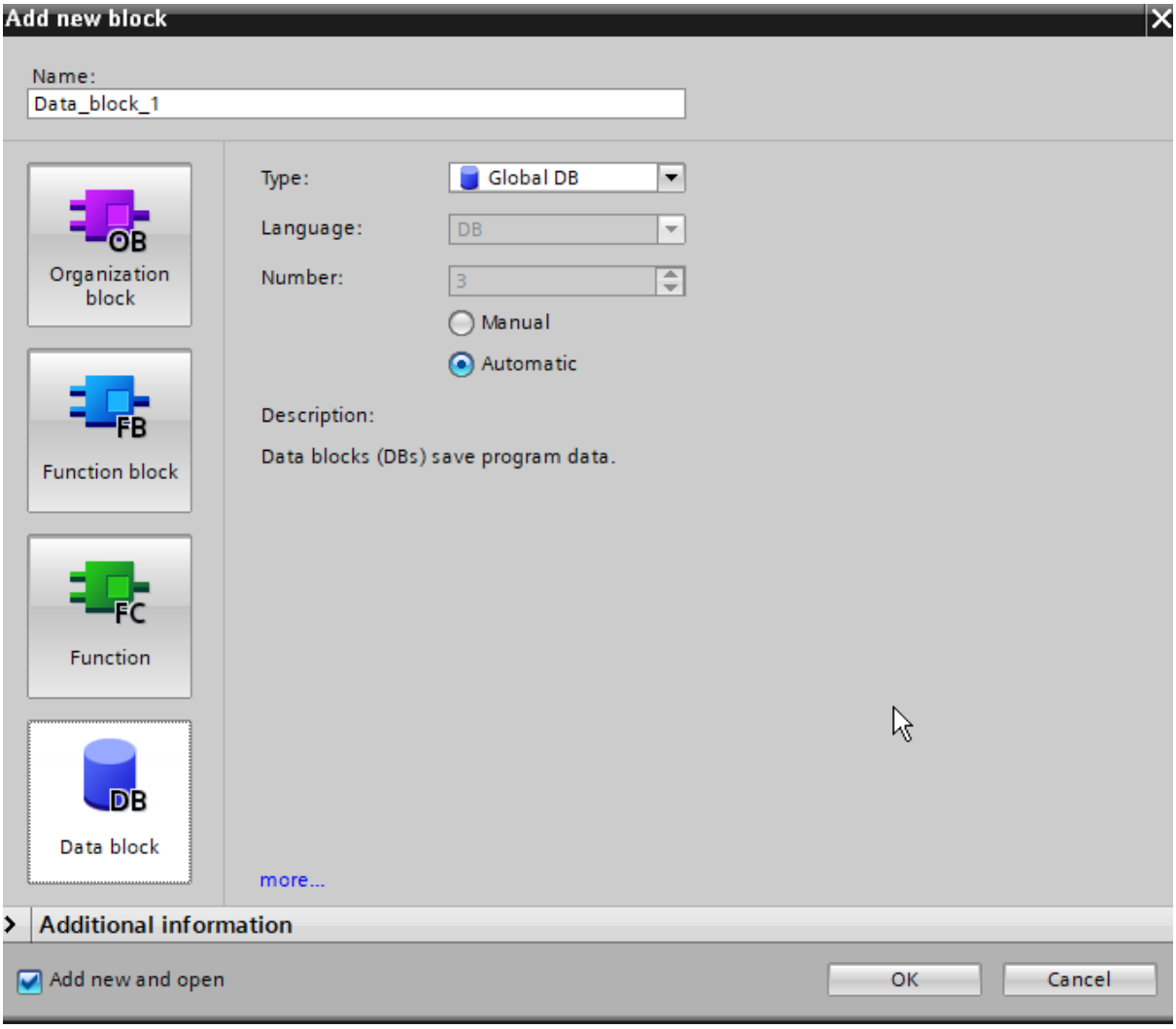


Figure 82 Data block window in TIA portal

Chapter 6 - Practical Applications and Industry Projects

This chapter delves into the practical applications and industry projects that utilize the power of TIA Portal for PLC programming and simulation. We will explore three key examples that demonstrate the versatility and efficiency of TIA Portal in various industrial scenarios.

1. **Sequential Activation of Three-Phase Motors:** This project illustrates how TIA Portal can be used to control the operation of multiple three-phase motors in a sequential manner.
2. **Bidirectional Motor Control with Delay:** Here, we delve into the implementation of bidirectional control (clockwise or counter-clockwise) of a motor with a delay factor, showcasing the precision and flexibility offered by TIA Portal.
3. **Implementing Automated Control in Conveyor Belt Systems:** In this project, we explore how TIA Portal can be used to automate conveyor belt systems, underscoring its potential in streamlining industrial processes.
4. **Utilizing Norm_X and Scale_X Functions:** for Temperature Measurement in a motor. This procedure involves the incorporation of a PT100 sensor or RTD (Resistance Temperature Detector) into a PLC system.
5. **HMI linear Scaling:** This example deals with analog values. so, the challenge arises when translating between abstract concepts like motor speed percentages and PLC integers.
6. **Automated Tank Water Level Control with Siemens S7-1500 PLC and HMI Interface:** With the use of Siemens S7-1500 PLC, automatic control of two pumps based on the water levels in two separate tanks.

Through these examples, we aim to provide a comprehensive understanding of how TIA Portal can be leveraged for diverse industrial applications, thereby equipping readers with practical knowledge that can be applied in real-world scenarios.

Sequential Activation of Three-Phase Motors

In this project, we are dealing with a system that includes two three-phase motors, a start button, and a stop button. The objective is to design a control sequence that operates the motors in a specific pattern upon activation of the start button.

Upon pressing the start button, the first motor is energized and begins operation. Following a delay of 4 seconds, the second motor is then brought into the circuit and commences operation. After an additional delay of 4 seconds, the first motor is de-energized and ceases operation. Subsequently, after yet another delay of 4 seconds, the second motor is also de-energized and stops operating.

Once the second motor ceases operation, the entire sequence is set to repeat from the beginning in an automatic manner. This cyclical process continues indefinitely until interrupted by the activation of the stop button.

Network 1:

The first network consists of a start button (Tag_1) that activates motor 1 (Tag_3) by setting the flip-flop. The reset input of the flip-flop is connected to a stop button (Tag_2). Therefore, network 1 enables the operation of motor 1.

Network 2:

The second network ensures that motor 2 starts after a delay of 4 seconds from the start of motor 1. This is achieved by using a NO contact of motor 1 and connecting it to a timer block (DB2).

Network 3:

The third network uses the output of the timer block (DB2) to set another flip-flop (Tag_6) that controls motor 2. The reset input of this flip-flop is connected to I0.1, which is the common reset address for all the flip-flops in the system. The address of the flip-flop is Q0.1, which corresponds to the address of motor 2.

Network 4:

The fourth network ensures that motor 1 stops after a delay of 4 seconds from the start of motor 2. This is achieved by using a NO contact of motor 2 (Q0.1) and connecting it to another timer block (DB3). The output of this timer block (Q of DB3) is used to reset the flip-flop of motor 1 in network 1, in parallel with the stop button (Tag_2).

Network 5:

The fifth network ensures that motor 2 stops after a delay of 4 seconds from the stop of motor 1. This is achieved by creating a new branch in network 3 that uses the falling edge of motor 1 (Tag_3, Q0.0) to trigger an N-trig block (Tag_4, M0.0). The output of this block is a brief pulse, which is converted into a steady signal by an SR flip-flop (M0.1). The output of this

flip-flop (Q of M0.1) triggers another timer block (DB4) that resets the flip-flop of motor 2 in network 3. The reset input of the SR flip-flop (M0.1) is connected to a NO contact of motor 2 (Q0.1) and an N-trig block (M0.2), which resets the timer block (DB4) when motor 2 stops.

Network 6:

The sixth network ensures that motor 1 starts again after a delay of 4 seconds from the stop of motor 2, creating a cyclic process. This is achieved by creating a new branch in network 1 that uses the falling edge of motor 2 to trigger another N-trig block (M0.3). The output of this block is a brief pulse, which is converted into a steady signal by another SR flip-flop (M0.4). The output of this flip-flop (Q of M0.4) triggers another timer block (DB5) that sets the flip-flop of motor 1 in network 1. The reset input of the SR flip-flop (M0.4) is connected to a NO contact of motor 1 (Q0.0), which resets the timer block (DB5) when motor 1 starts.

Tag	Name	Address
Tag_1	Start Button	I0.0
Tag_2	Stop Button	I0.1
Tag_3	Motor 1	Q0.0
IEC_Timer_0_DB_2	Timer (DB2)	DB3
Tag_6	Flip-Flop for Motor 2	Q0.1
Tag_6	Motor 2	Q0.1
IEC_Timer_0_DB_3	Timer (DB3)	DB4
Tag_24	Flip-Flop (M0.1)	M0.1
Tag_4	N-TRIG for Motor 2	M0.2
IEC_Timer_0_DB_4	Timer (DB4)	DB5
Tag_26	N-TRIG for Motor 1	M0.3
Tag_27	Flip-Flop for Motor 1	M0.4
IEC_Timer_0_DB_5	Timer (DB5)	DB6

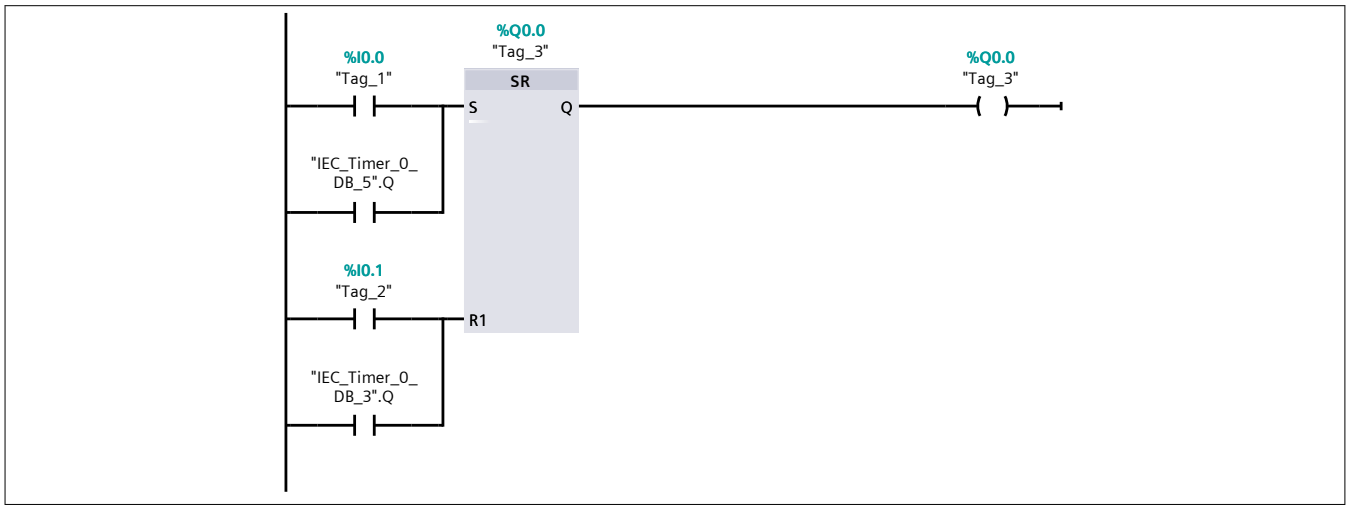
Table 5 List of tags with names and its address of example 1

Sequential Activation of Three-Phase Motors [OB1]

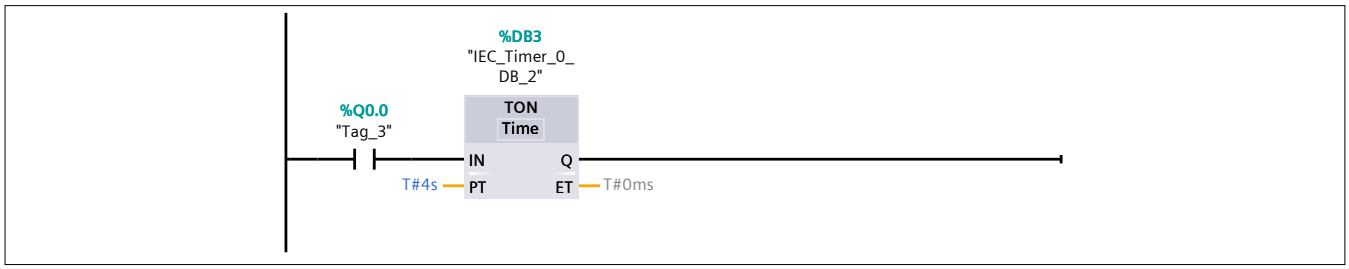
Sequential Activation of Three-Phase Motors Properties					
General					
Name	Sequential Activation of Three-Phase Motors	**Number**	1	**Type**	OB
Language	LAD	**Numbering**	Automatic		
Information					
Title	Sequential Activation of Three-Phase Motors	**Author**	Hamid_Hosseinzadeh	**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Main			
Name	**Data type**	**Default value**	**Comment**
Input			
Initial_Call	Bool		Initial call of this OB
Remanence	Bool		=True, if remanent data are available
Temp			
Constant			

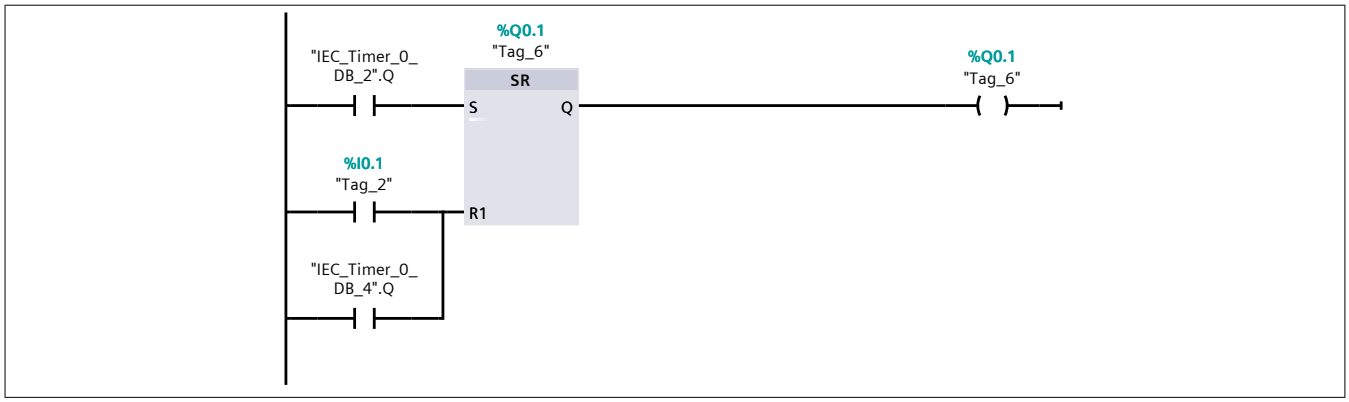
Network 1: Motor 1



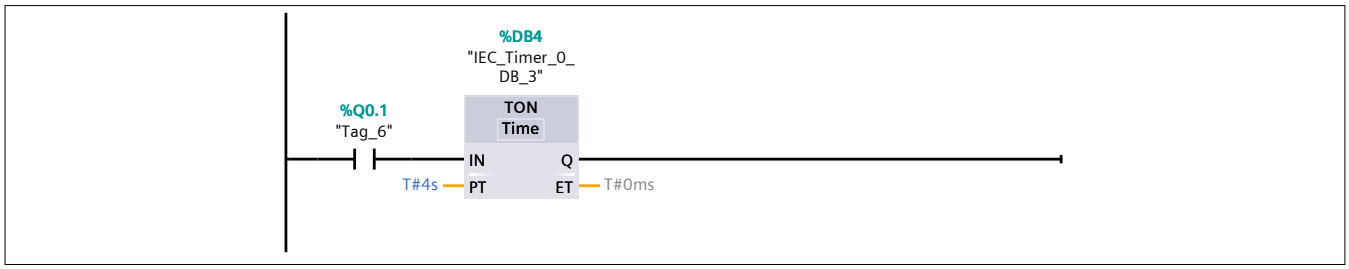
Network 2: Motor 1 is on after 4 seconds



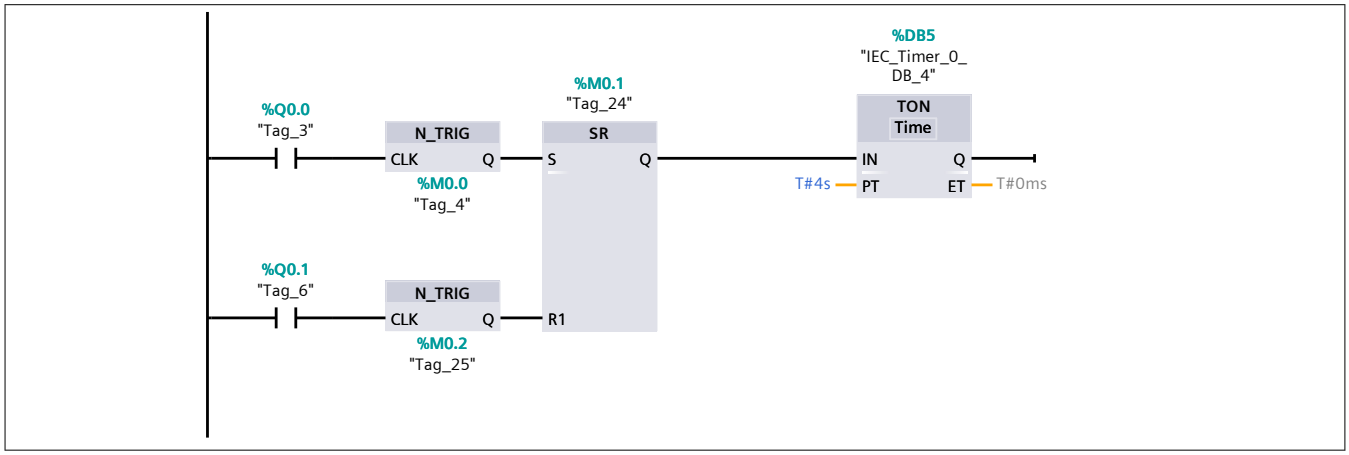
Network 3: after 4 seconds the second moter is ON



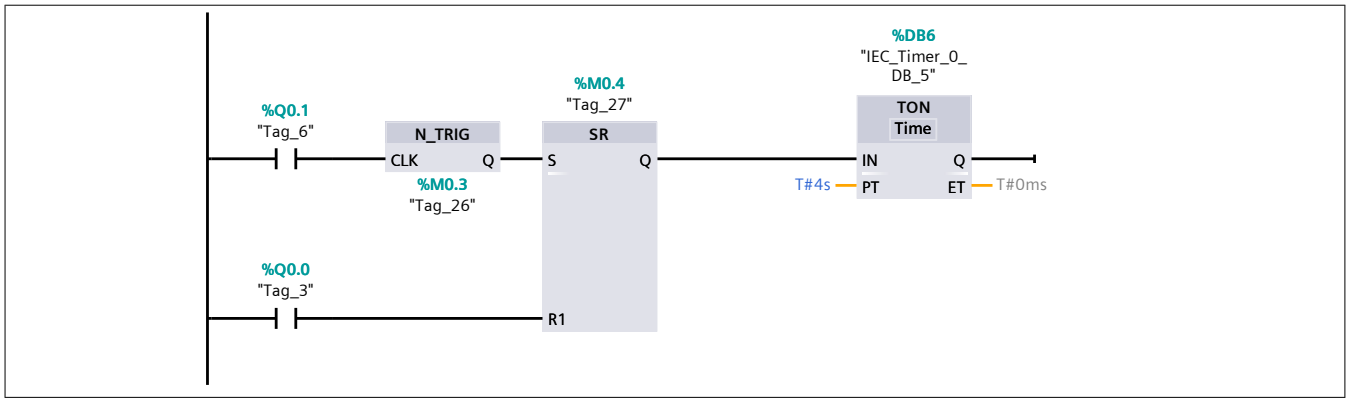
Network 4: when motor 2 is ON after 4 second motor 1 should go OFF



Network 5: when the motor 1 is going to be OFF after 4 sec Motor 2 go OFF too



Network 6: when motor 2 is OFF after 4 sec motor 1 should be ON


%Q0.1
"Tag_6"
N_TRIG
CLK
Q
%M0.3
"Tag_26"
%M0.4
"Tag_27"
SR
S
Q
%DB6
"IEC_Timer_0_
DB_5"
TON
Time
IN
Q
T#4s
PT
ET
T#0ms
%Q0.0
"Tag_3"
R1

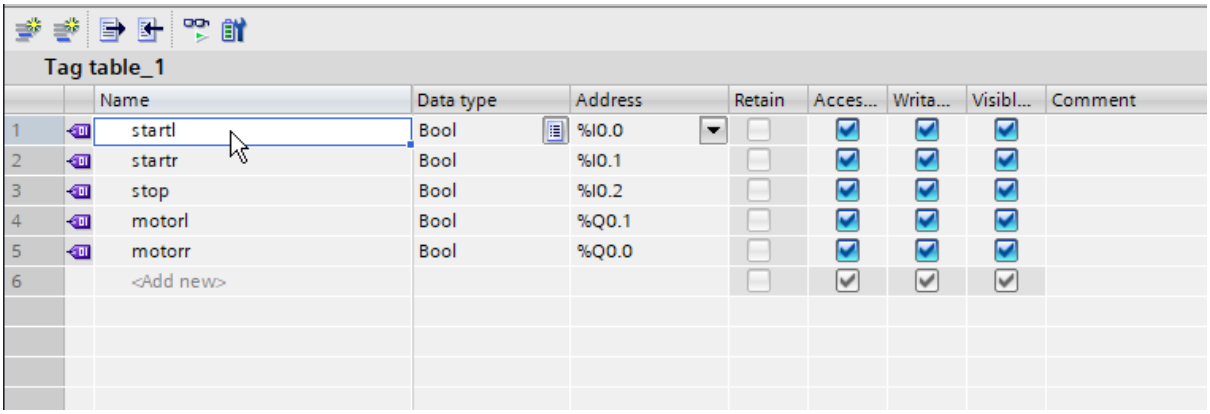
Bidirectional Motor Control (Clockwise or Counter Clockwise) with Delay

In this project, we are presented with a system that includes a single motor, two start buttons (one for left-handed mode and the other for right-handed mode), and one stop button. The objective is to design a control sequence that operates the motor in a specific pattern upon activation of the start buttons.

Upon pressing the left-handed start button, the motor is energized and begins operation in the left-handed direction. If the right-handed start button is pressed, after a delay of 5 seconds, the motor starts moving in the right-handed direction.

Now, if the left-handed start button is pressed again, the motor will commence operation in the left direction after the same delay of 5 seconds.

This project demonstrates an application of bidirectional control in industrial automation, showcasing how precise timing and control sequences can be implemented using Programmable Logic Controllers (PLCs) and TIA Portal software.



Tag table_1

	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	Comment
1	startl	Bool	%I0.0	☐	☑	☑	☑	
2	startr	Bool	%I0.1	☐	☑	☑	☑	
3	stop	Bool	%I0.2	☐	☑	☑	☑	
4	motorl	Bool	%Q0.1	☐	☑	☑	☑	
5	motorr	Bool	%Q0.0	☐	☑	☑	☑	
6	<Add new>			☐	☑	☑	☑	

Figure 83 Tag Table for Bidirectional Motor Control

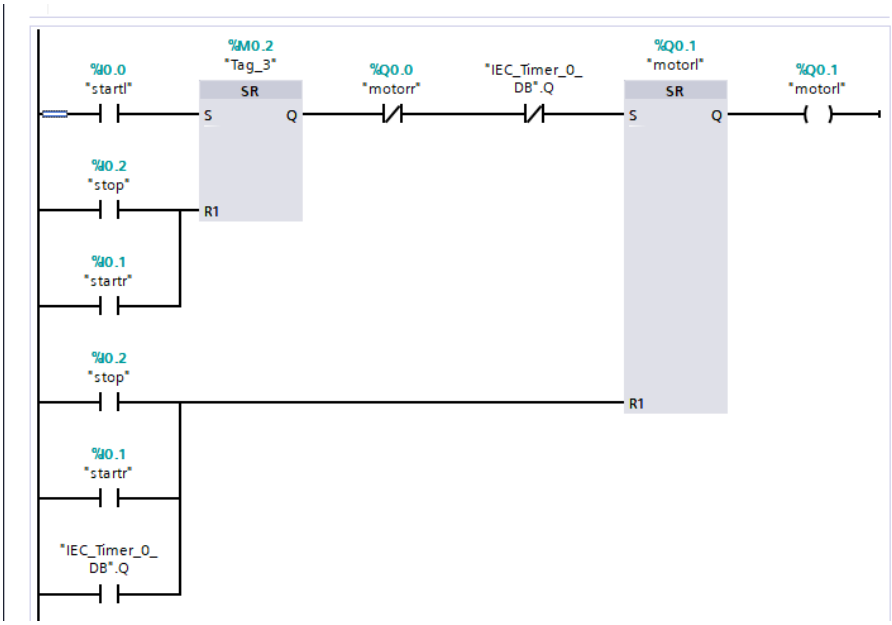


Figure 84 Main window in TIA portal for Bidirectional Motor Control

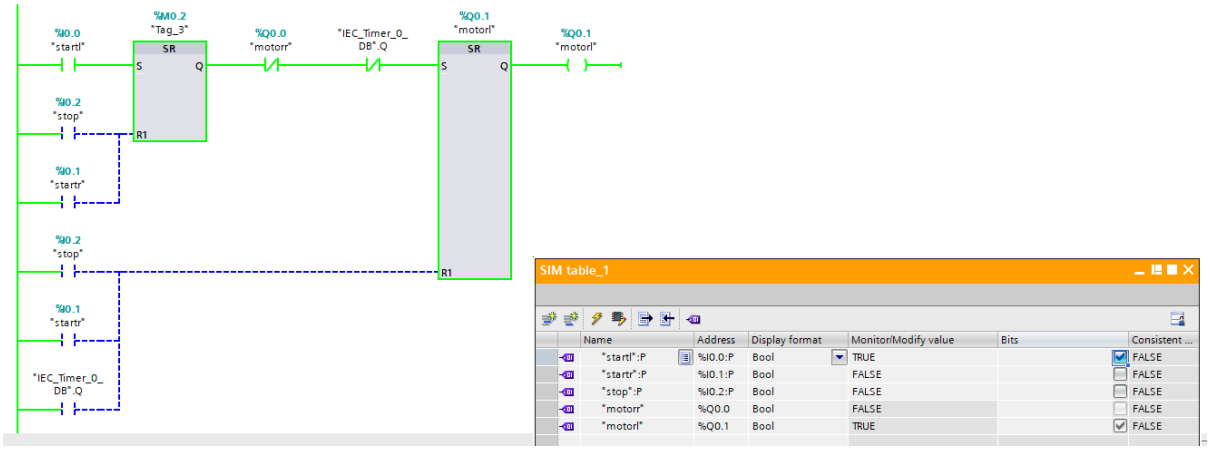


Figure 85 Simulation of Bidirectional Motor Control

Network 1:

This network controls the counterclockwise motor (CCW motor) that rotates to the left. It consists of a flip-flop with the address of motorL. The set input of the flip-flop is startL and the reset input is stop. When the clockwise motor (CW motor) is started, the CCW motor must be turned off. Therefore, the reset input of the flip-flop also includes the start signal of the CW motor (motorR) with the address of I0.1.

The startL button (I0.0) is activated when the motor is off and the timer is off. This sets the flip-flop and starts the motorL (CCW motor). The startL signal is a pulse, so another flip-flop is added after startL to maintain the pulse. (Note: flip-flops are used to convert a quick pulse to a permanent pulse). The address of this flip-flop is m0.2, which is a memory address, since there is no output device (such as a motor) connected to it. The reset input of the flip-flop (m0.2) consists of the stop button, the start signal of the CW motor (motorR), and the timer, connected in parallel.

The second flip-flop has an output to motorL, so the address of the flip-flop is also motorL.

Network 2:

The configuration of network 2 is similar to network 1, except that it controls the CW motor that rotates to the right.

Network 3:

This network implements the timer. The type of timer used here is TP timer. The TP (Pulse) timer is similar to the TON timer, but it operates differently. It produces a pulse at its output when the accumulated time reaches the preset time. The inputs to the timer are motorR and motorL. Two NO contacts for motorR and motorL are connected in parallel. The timer is activated when the motors are turned off, so N-TRIG is used for both of them (M0.0 and M0.1). The preset time is 5 seconds.

Tag	Name	Address
startl	Start Left Button	%I0.0
motorl	Left Motor	%Q0.1
stop	Stop Button	%I0.2
startr	Start Right Button	%I0.1
motorr	Right Motor	%Q0.0
Tag_3	Flip-flop for Left Motor	%M0.2
Tag_4	Flip-flop for Right Motor	%M0.3
IEC_Timer_0_DB	Timer	%DB1
Tag_1	N-TRIG for Right Motor	%M0.0
Tag_2	N-TRIG for Left Motor	%M0.1

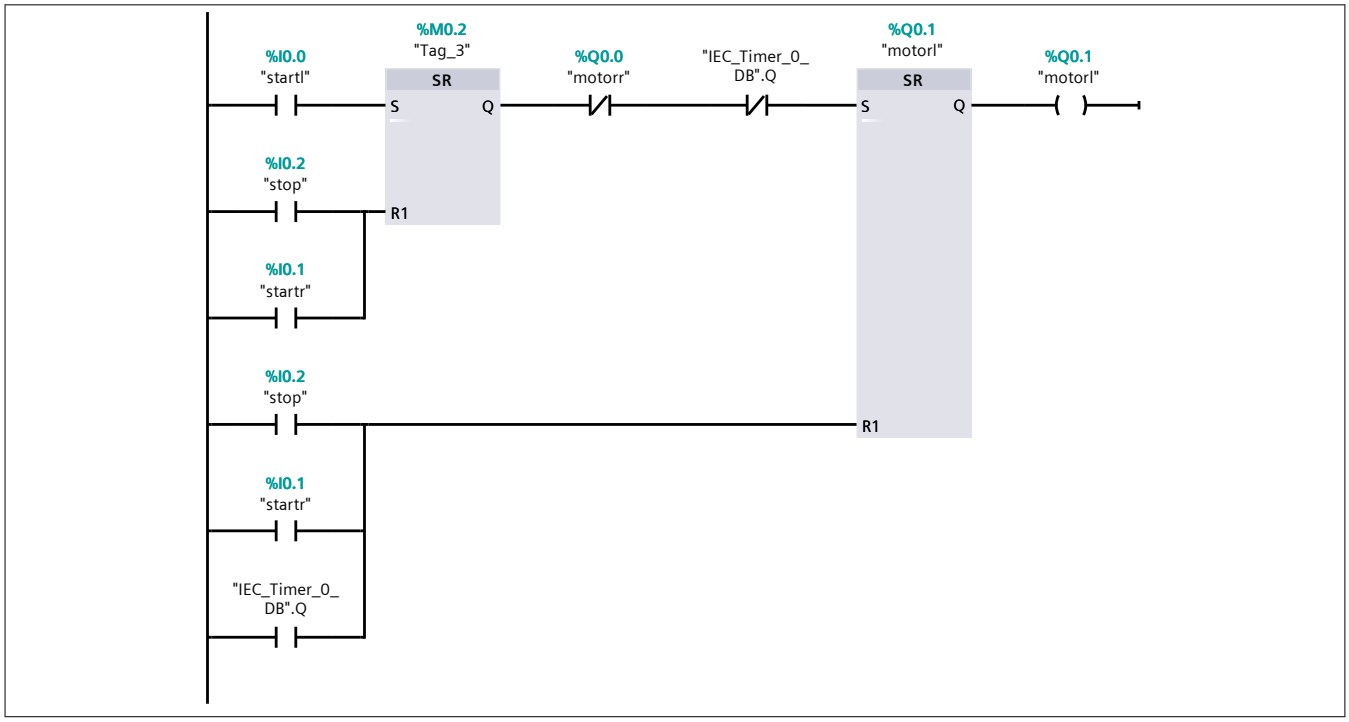
Table 6 List of tags with names and its address of example 2

Main [OB1]

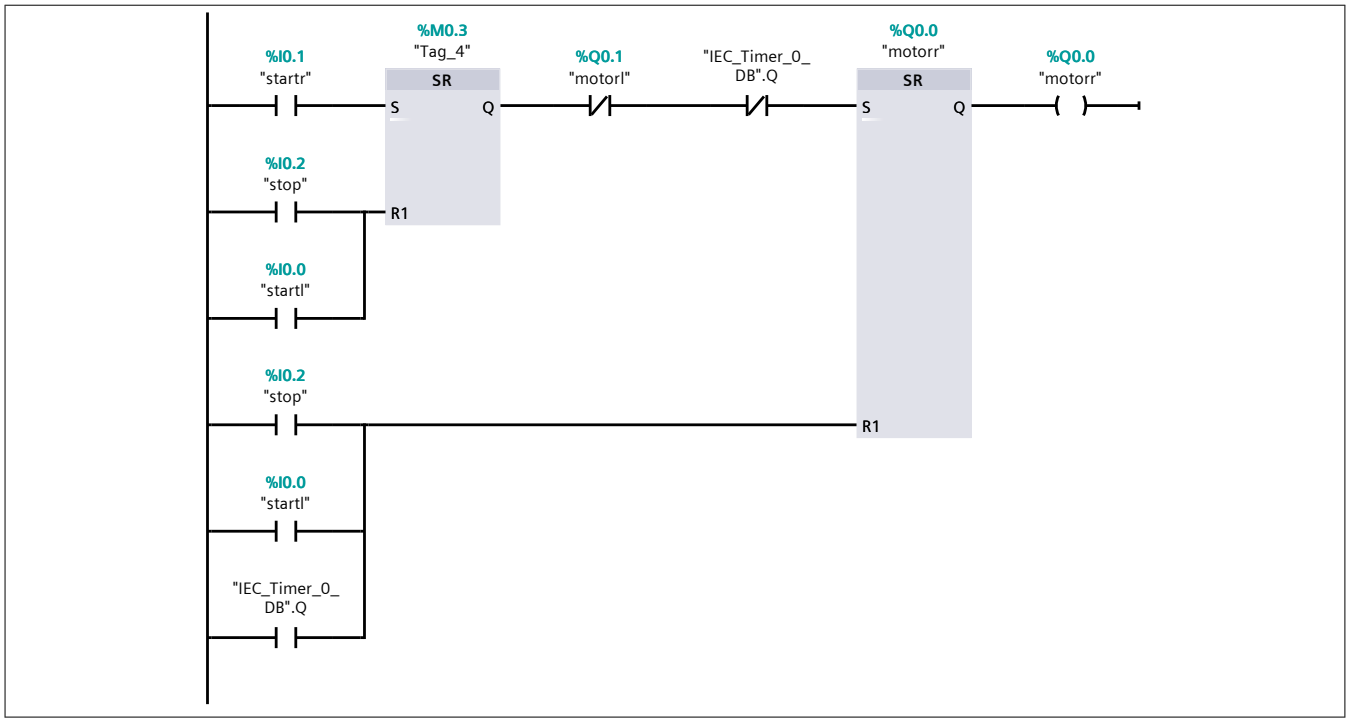
Main Properties					
General					
Name	Main	**Number**	1	**Type**	OB
Language	LAD	**Numbering**	Automatic		
Information					
Title	Bidirectional Motor Control	**Author**	Hamid_Hosseinzadeh	**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Main			
Name	**Data type**	**Default value**	**Comment**
Input			
Initial_Call	Bool		Initial call of this OB
Remanence	Bool		=True, if remanent data are available
Temp			
Constant			

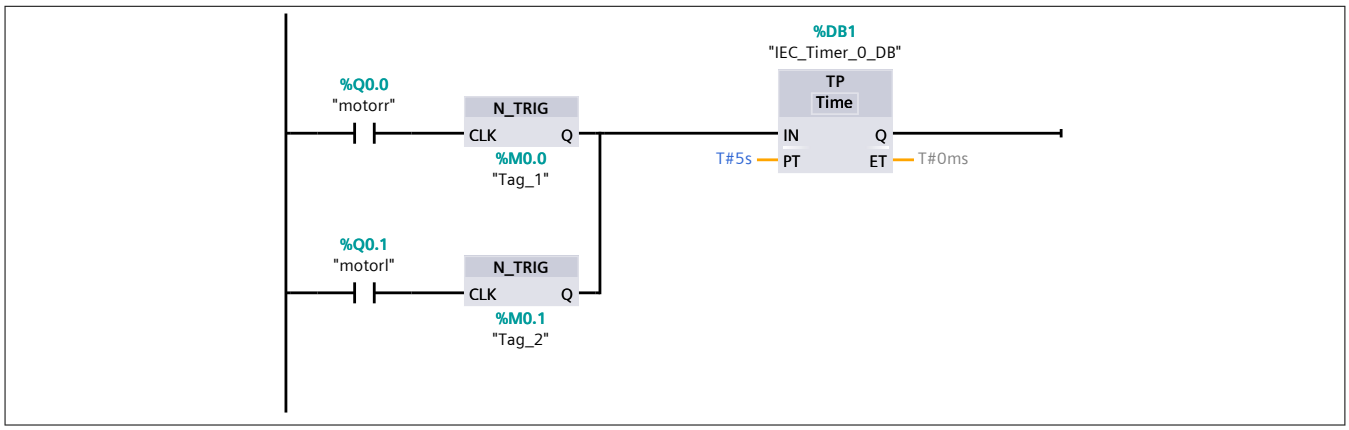
Network 1: Motor Left turn contorl



Network 2: Motor Right turn contorl



Network 3: Timer



Implementing Automated Control in Conveyor Belt Systems Using TIA Portal

In this project, we are dealing with a system that includes two conveyor motors, a start button, a stop button, and two sensors. The objective is to design a control sequence that operates the conveyor motors in a specific pattern upon activation of the start button.

Upon pressing the start button, the first conveyor motor is energized and begins operation. After the first conveyor sensor counts 7 boxes, the first conveyor motor is de-energized and ceases operation. Simultaneously, the second conveyor motor is energized and begins operation, directing the boxes to the warehouse.

Once the second sensor has counted these boxes and they have been directed to the warehouse, the first motor is immediately re-energized and resumes operation, directing additional boxes to the warehouse.

This project demonstrates an application of sequential control in industrial automation, showcasing how precise timing and control sequences can be implemented using Programmable Logic Controllers (PLCs) and TIA Portal software.

Tag table_1

	Name	Data type	Address	Retain	Acces...	Writa...	Visibl...	Comment
1	start	Bool	%I0.0	☐	☑	☑	☑	
2	stop	Bool	%I0.1	☐	☑	☑	☑	
3	sensor1	Bool	%I0.2	☐	☑	☑	☑	
4	motor2	Bool	%Q0.1	☐	☑	☑	☑	
5	motor1	Bool	%Q0.0	☐	☑	☑	☑	
6	sensor2	Bool	%I0.3	☐	☑	☑	☑	
7	<Add new>			☐	☑	☑	☑	

Figure 86 Tag table for Automated Control in Conveyor Belt Systems

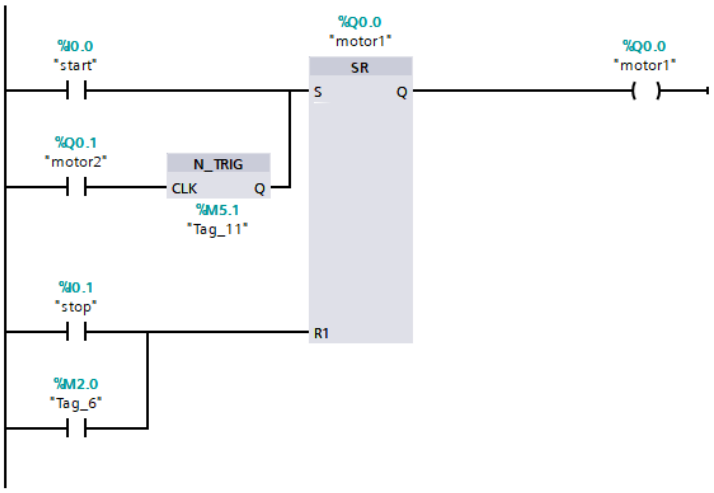


Figure 87 Main [OB1] for Automated Control in Conveyor Belt Systems

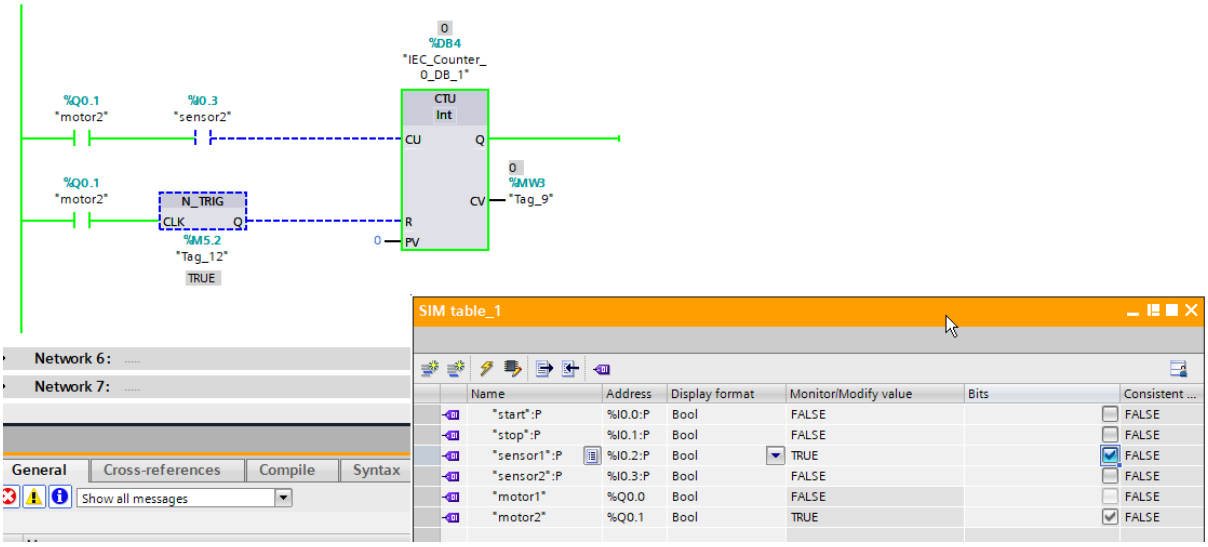


Figure 88 Simulation for Automated Control in Conveyor Belt Systems

Network 1:

This network consists of a start button, a flip-flop SR, and a coil for the motor. The set input of the flip-flop is connected to the start button and the output of the second motor's N_TRIG (M5.1), which ensures that the cycle continues when the second motor is turned off. The reset input of the flip-flop is connected to the stop button and M2.0, which is defined in Network 3. When the counter reaches 7, the first motor is reset.

Network 2:

This network is activated when the first motor is on, as indicated by a NO contact of motor 1 (Q0.0) in series with sensor 1 (I0.2). A CTU (Count Up) operation is used to count the number of cycles. The output of CV is a MWord, so the address is MW0. The reset input of the counter is connected to the output of the first motor and the N_TRIG (M2.1), which resets the counter when the first motor is turned off. The Q and PV outputs of the counter are not used, so PV is set to 0.

Network 3:

This network uses a CMP Equal operation to compare MW0 with 7. If they are equal, the output is stored in a coil with the address M2.0. This address is chosen because M0 and M1 are already occupied by MW0, which is a word of 16 bits (2 bytes). M2.0 is used to reset the first motor in Network 1 when the counter reaches 7.

Network 4:

This network controls the second motor, which is turned on when the first motor is turned off. This is achieved by connecting the output of the first motor and the N_TRIG to the set input of a flip-flop, whose output is the second motor. The reset input of the flip-flop is

connected to the stop button and M5.0 (Tag_10), which is defined in Network 6. When the counter in Network 5 reaches 7, the second motor is reset.

Network 5:
This network counts the number of cycles of the second motor and sensor 2 using a CTU operation. The output of CV is a MWord with the address MW3. PV is set to 0. The reset input of the counter is connected to the output of the second motor and the N_TRIG, which resets the counter when the second motor is turned off.

Network 6:
This network uses a CMP Equal operation to compare MW3 with 7. If they are equal, the output is stored in a coil with the address M5.0. This address is used to reset the second motor in Network 4 when the counter reaches 7.

Tag	Name	Address
start	Start Button	%I0.0
stop	Stop Button	%I0.1
motor1	Motor 1	%Q0.0
sensor1	Sensor 1	%I0.2
IEC_Counter_0_DB	Counter for Motor 1	%DB1
Tag_5	Counter Value for Motor 1	%MW0
Tag_7	N-TRIG for Motor 1	%M2.1
Tag_6	Flip-flop for Motor 2	%M2.0
Tag_8	Flip-flop for Motor 2	%M2.2
motor2	Motor 2	%Q0.1
sensor2	Sensor 2	%I0.3
IEC_Counter_0_DB_1	Counter for Motor 2	%DB4
Tag_9	Counter Value for Motor 2	%MW3
Tag_12	N-TRIG for Motor 2	%M5.2
Tag_10	Flip-flop for Motor 1	%M5.0
Tag_11	Flip-flop for Motor 1	%M5.1

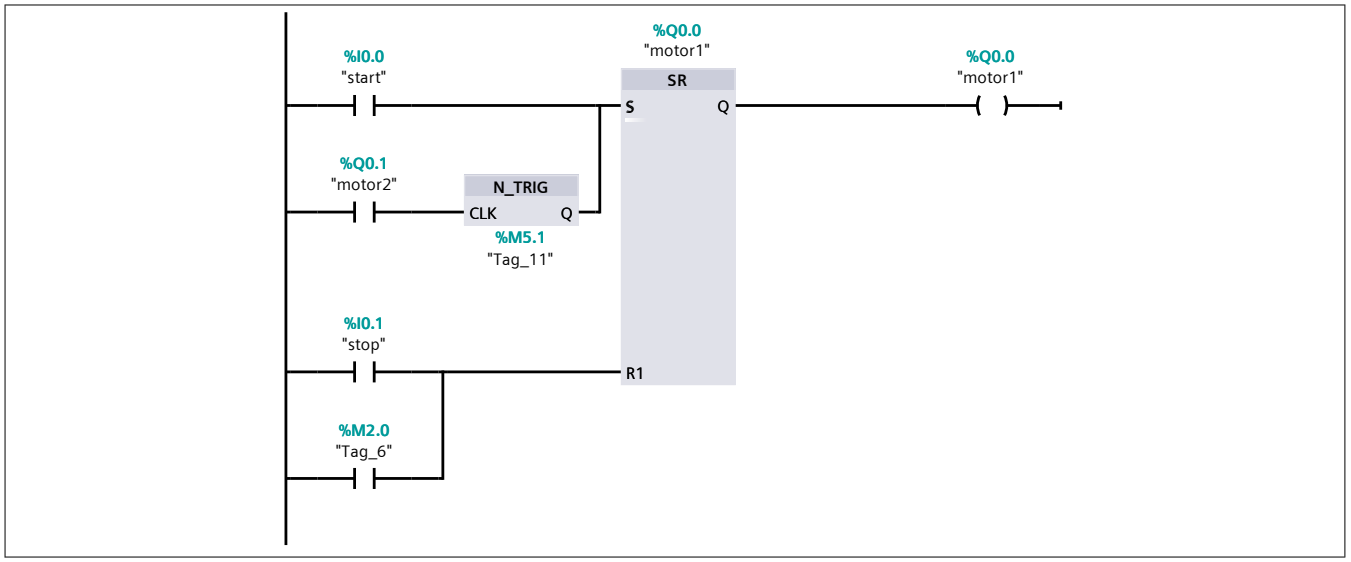
Table 7 List of tags with names and its address of example 3

Main [OB1]

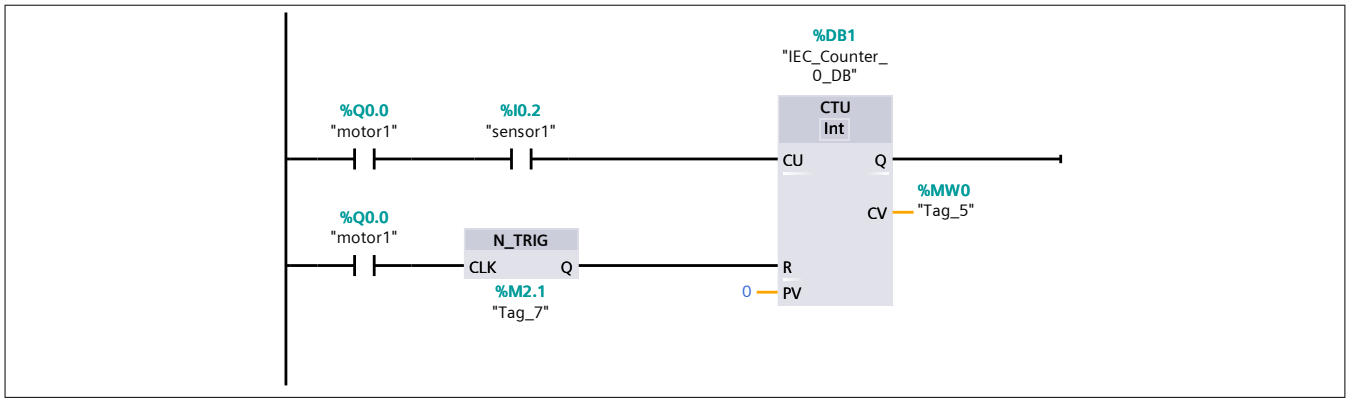
Main Properties					
General					
Name	Main	**Number**	1	**Type**	OB
Language	LAD	**Numbering**	Automatic		
Information					
Title	Conveyor Belt Systems	**Author**	Hamid_Hosseinzadeh	**Comment**	
Family		**Version**	0.1	**User-defined ID**	

Main			
Name	**Data type**	**Default value**	**Comment**
Input			
Initial_Call	Bool		Initial call of this OB
Remanence	Bool		=True, if remanent data are available
Temp			
Constant			

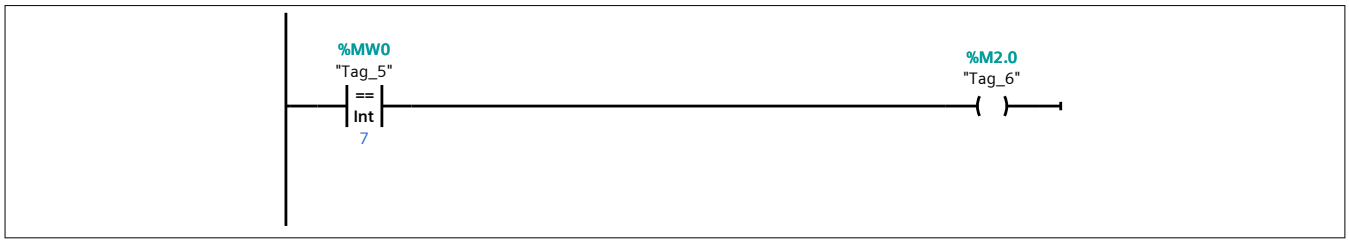
Network 1: Motor 1



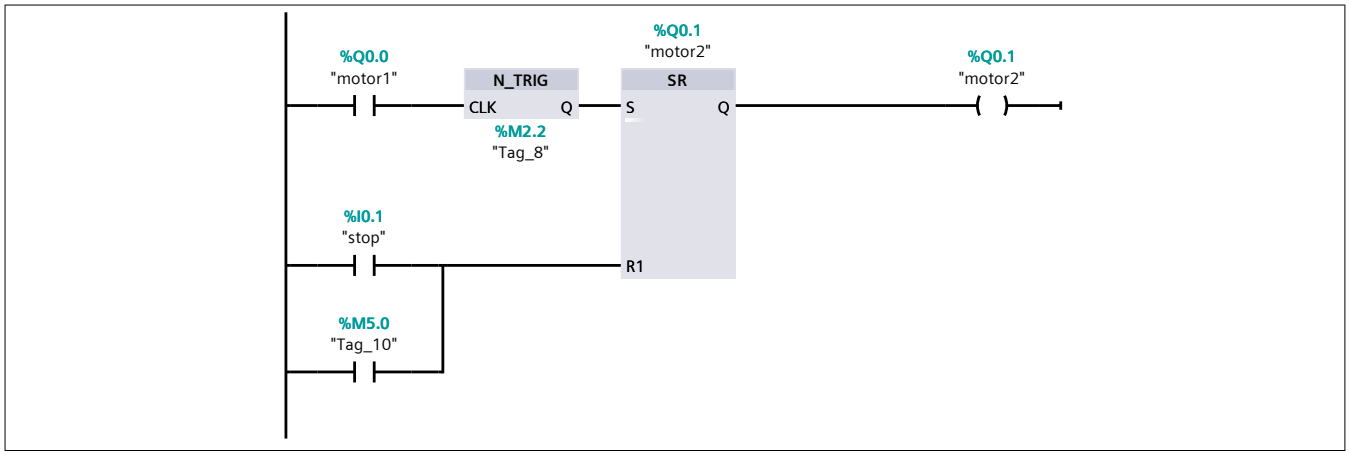
Network 2: Motor 1 and sensor 1 with Counter



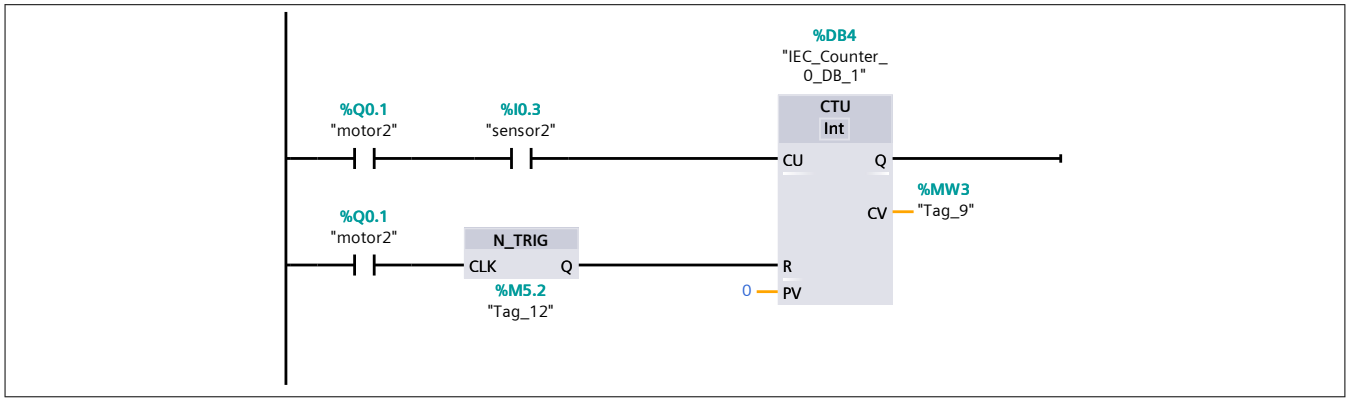
Network 3: when 7 boxes passed motor 1 should reset



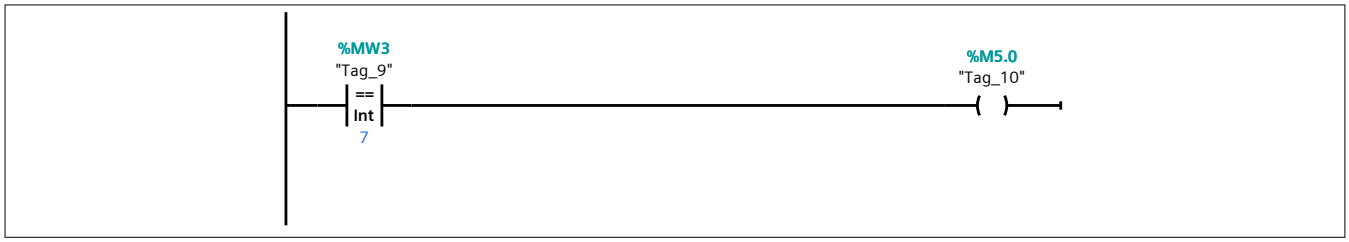
Network 4: Motor 2 is on when motor 1 is off



Network 5: when 7 boxes passed motor 2 should reset



Network 6: when 7 boxes passed motor 2 should reset

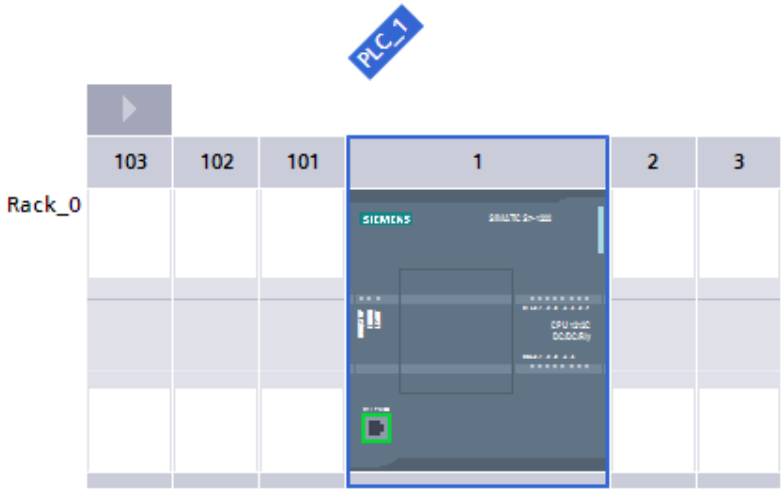


Utilizing Norm_X and Scale_X Functions for Temperature Measurement in TIA Portal

This process necessitates the integration of a PT100 sensor or RTD (Resistance Temperature Detector) with a PLC, allowing for the conversion of analog temperature data into a format suitable for further processing.

PLC Analog Inputs Configuration

Begin by configuring the analog inputs of the PLC, a fundamental step in the measurement process. In this context, the objective is to measure the temperature of a motor utilizing a PT100 sensor. The PLC under consideration provides two analog inputs, denoted as AI 2, accessible through specific addresses. The address of the first analog channel (Channel 0) is IW64, while the address of the second channel (Channel 1) is IW66.



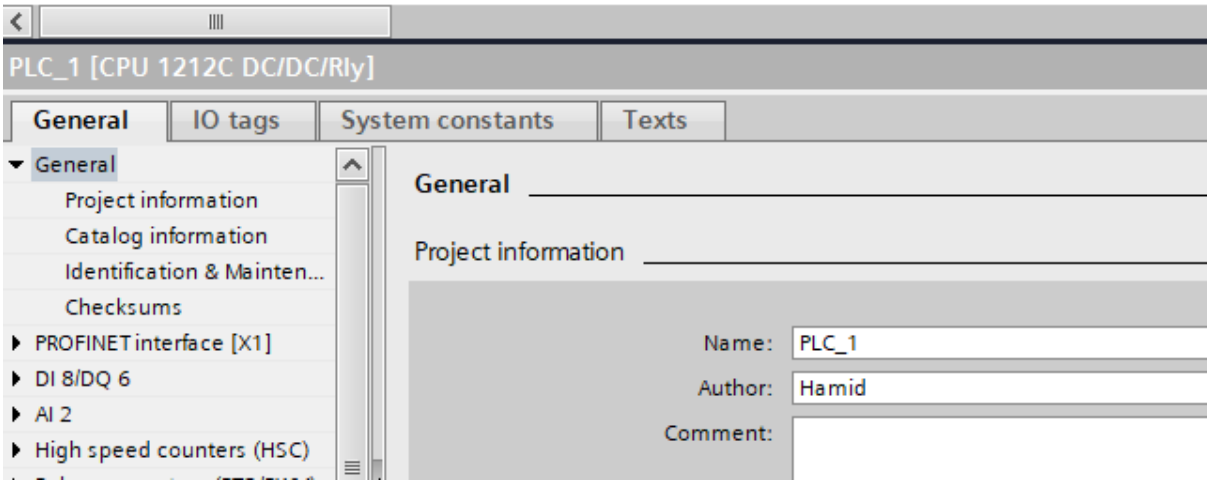


Figure 89 The PLC General Tab: Configuration settings for the PLC.

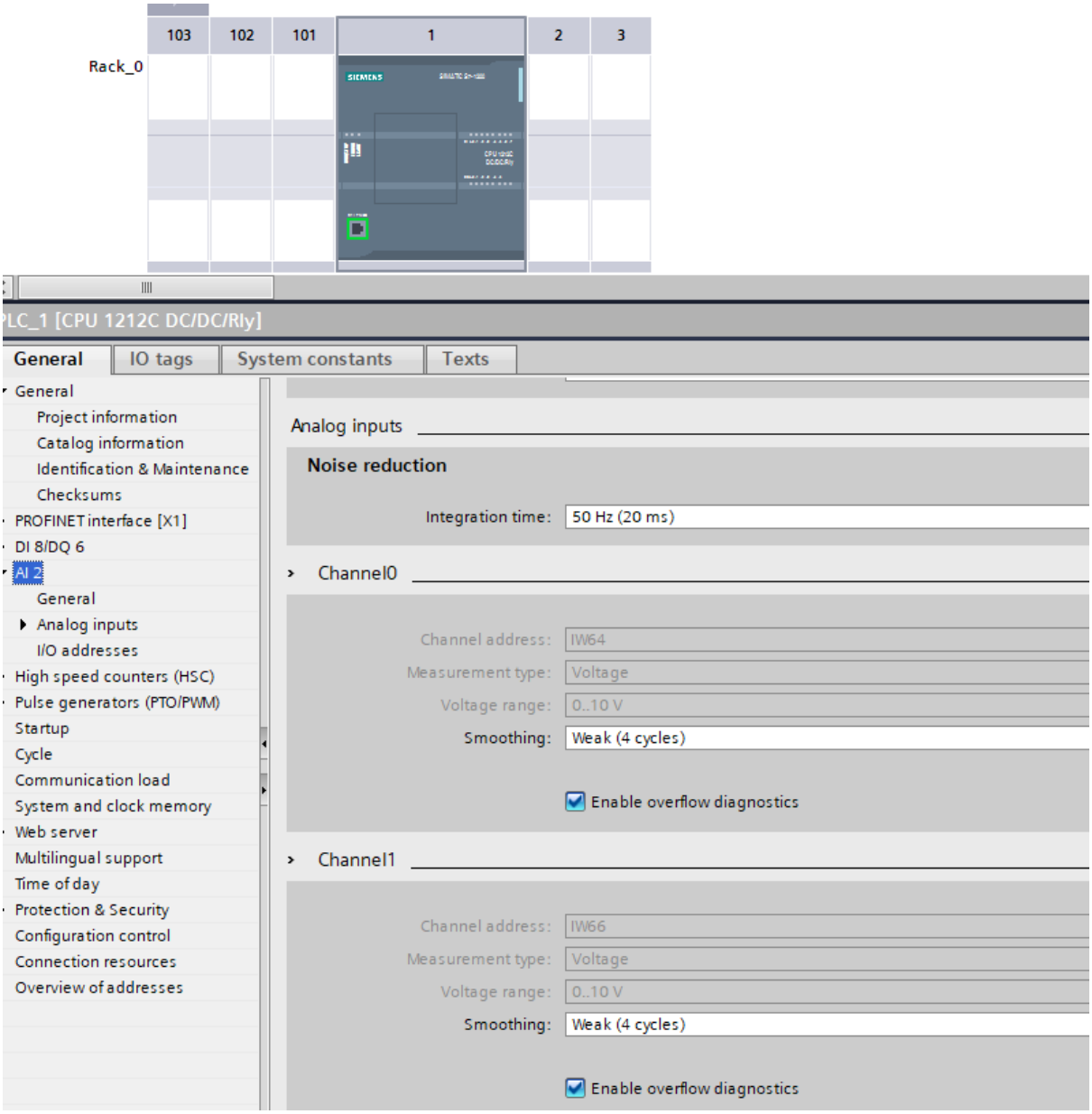


Figure 90 Analog Inputs in PLC: Dual Channels, Channel 0 and Channel 1

Data Conversion with Norm_X:

The Norm_X function plays a pivotal role in converting the acquired analog data into a format compatible with further processing. Configuration involves specifying the minimum and maximum values, which are intrinsically tied to the PLC's capabilities. In this case, the minimum pulse achievable by the PLC is 0, and the maximum is 27,648. The input value is drawn from IW64, the analog input address representing the sensor's data. The objective here is to transform the integer data into a real number format, and consequently, the output is directed to MD0.

Scaling Temperature Values with Scale_X:

Following the conversion facilitated by Norm_X, the Scale_X function comes into play. Its purpose is to scale the real number temperature data into a user-defined range. For the

motor's temperature measurement, we consider a minimum value of 0 and a maximum value of 80 degrees Celsius. The input data for Scale_X is derived from MD0, representing the real number output of the Norm_X function. In this instance, the goal is to convert real data into another real number format, ultimately directing the scaled temperature value to MD4.

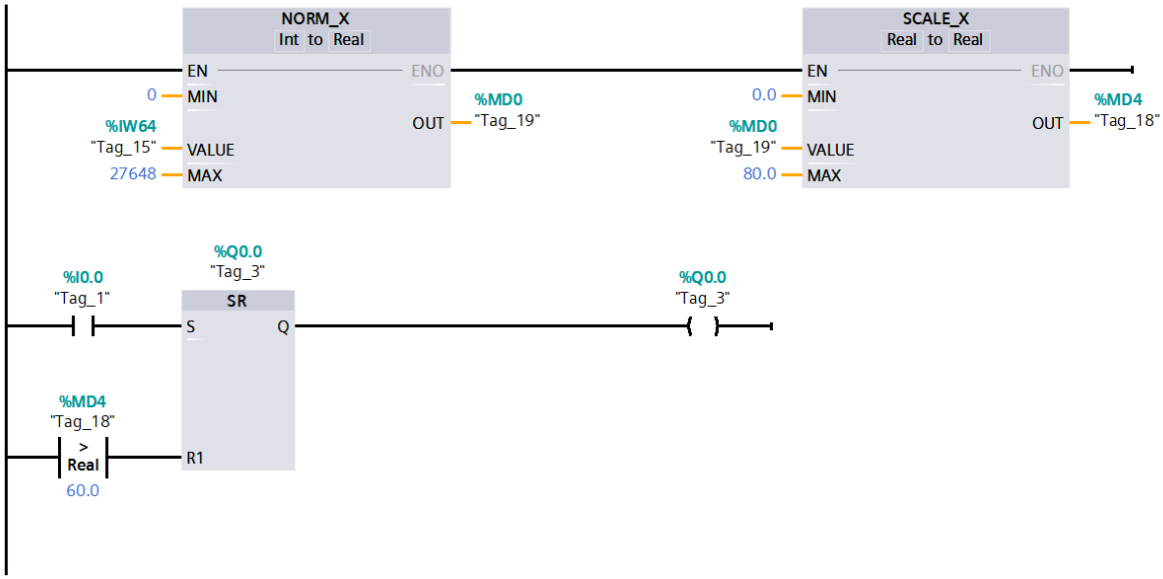


Figure 91 Application of Norm_X and Scale_X functions in Temperature Measurement for Motor Monitoring.

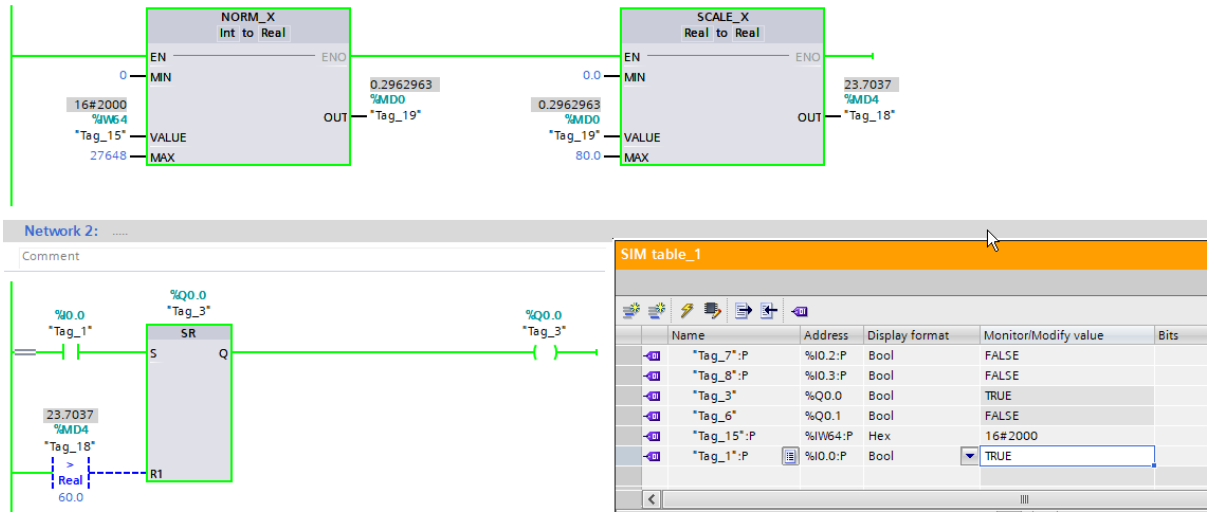
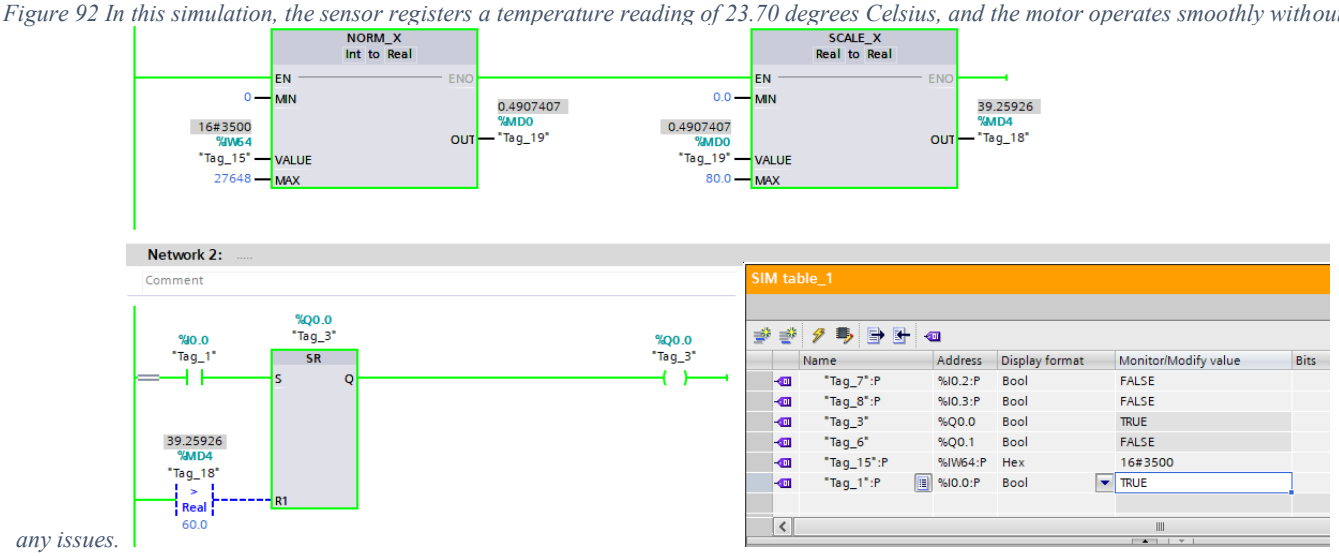


Figure 92 In this simulation, the sensor registers a temperature reading of 23.70 degrees Celsius, and the motor operates smoothly without



any issues.

Figure 93 During the second phase, the sensor records a temperature reading of 39.25 degrees Celsius, and the motor continues to operate effectively.

In the illustrative case presented below, an additional layer of sophistication can be introduced through the incorporation of a conditional logic block, specifically a "greater than" comparison. This logical construct enables us to formulate a precise control statement: "If the temperature, represented by the value stored in MD4, exceeds the critical threshold of 60 degrees Celsius, then execute the action of turning off the motor."

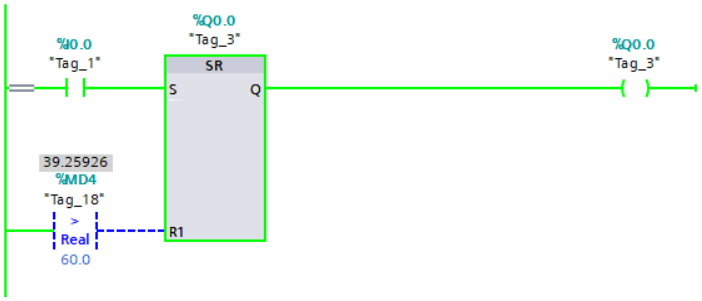


Figure 94 In this scenario, adding complexity involves utilizing a conditional logic block, particularly 'greater than' comparison. This allows for a precise control statement: 'If MD4's temperature value surpasses 60°C, turn off the motor.

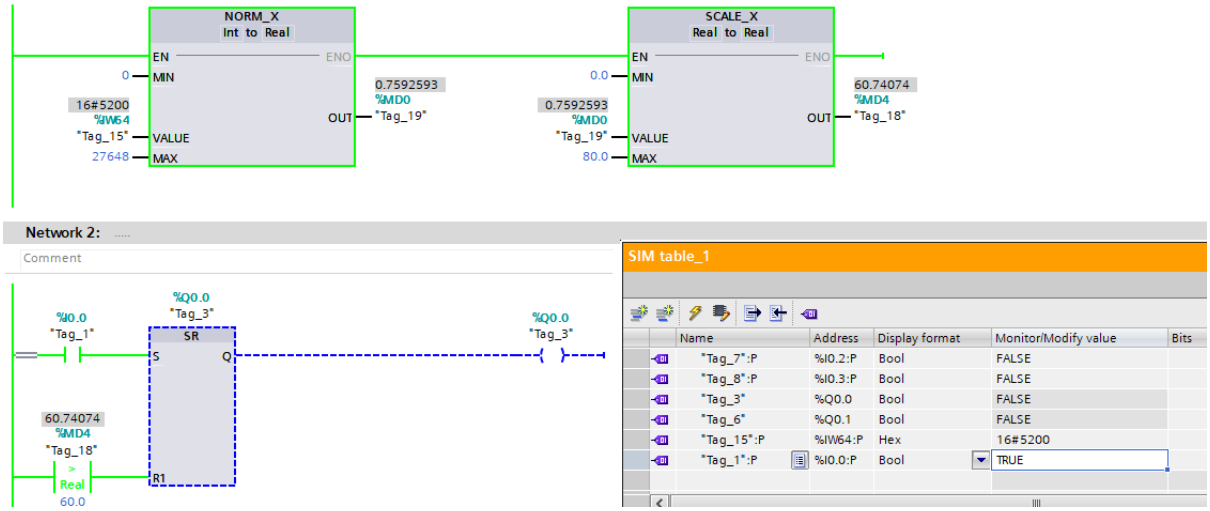


Figure 95 In this simulation, the temperature sensor records a reading of 60.74°C, surpassing the critical threshold of 60°C, thereby triggering the reset action and resulting in the motor's deactivation.

HMI linear scaling

Human-machine interfaces (HMIs) often require scaling analog values from PLCs to meaningful units for operators. One common example is controlling motor speed based on a PLC output value. This example demonstrates utilizing the built-in linear scaling function in Siemens TIA Portal HMI configuration to map PLC values to real-world units.

When dealing with analog values in the context of Siemens HMI, we often encounter the need for scaling. Consider a scenario where we aim to control the speed of a motor using an HMI interface. To achieve this, we typically begin by defining a variable within our data block, often of integer type. The challenge lies in correctly scaling this variable, ensuring that 0 corresponds to the motor being off, while 32,767 represents the motor running at 100% speed.

Now, the question arises: should we input values such as 30 mm or 100% or 16,200 or 50%? Ideally, we would prefer to work with percentages or RPM values, as it simplifies the interface and aligns with our understanding. Fortunately, we can accomplish this scaling procedure for both RPM and percentage values. To illustrate this process, we will demonstrate how to set up a slider on the HMI interface. The slider will range from 0 to 100, representing the desired motor speed percentage. However, a common misconception is that 0% corresponds to a PLC value of 0, and 100% equates to 32,767. In reality, this is not the case. In the context of an analog output, the maximum value reaches 32,767. Therefore, to achieve 100% motor speed, we must set the slider to approximately 32,700. However, this approach is impractical and unintuitive for users. To address this issue, we seek a scaling solution where 0% corresponds to 0 and 100% corresponds to 32,767.

Table 8 Mapping Percentage Values to PLC Integers and Corresponding Scaled Values

Motor Speed %	PLC Value	Slider Value
100%	32,767	100
90%	29,490	90
80%	26,213	80
70%	22,936	70
60%	19,659	60
50%	16,383	50
40%	13,106	40
30%	9,829	30
20%	6,552	20
10%	3,276	10
0%	0	0

Traditionally, this would involve writing a program to perform the necessary mathematical transformations. However, within the HMI, there is an automatic scaling feature that

simplifies this process. By selecting the appropriate tag and configuring the linear scaling properties, we can ensure that the HMI interface displays scaled values that align with our desired outcomes. This eliminates the need for manual programming and streamlines the scaling process.

Figure 96 Linear Scaling within HMI

In conclusion, linear scaling in Siemens HMI systems offers a convenient and efficient way to represent analog values in a user-friendly manner. By leveraging this feature, we can avoid complex programming and ensure that our HMI interfaces align with our expectations.

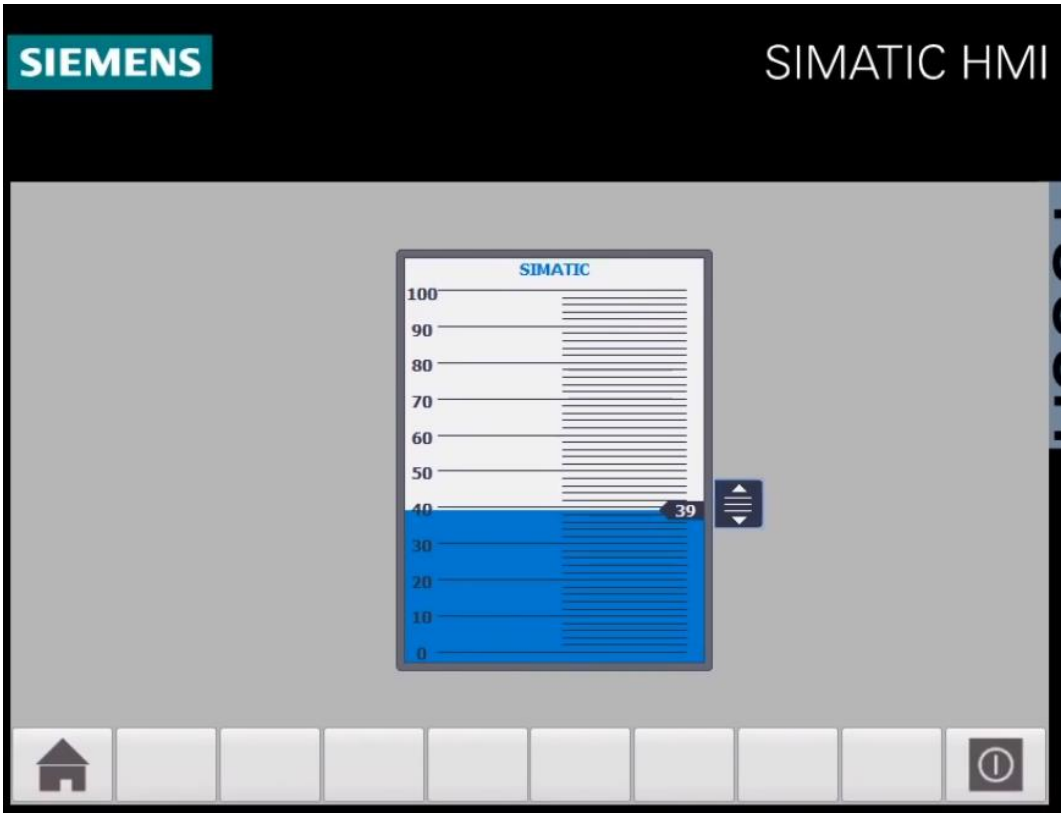


Figure 97 HMI interface in TIA portal using a scale bar

Automated Tank Water Level Control with S7-1500 PLC and HMI Interface

This example showcases the seamless integration of the Siemens S7-1500 PLC with an HMI interface to automatically control two pumps based on water levels in two tanks. The ladder diagram programming ensures precise and efficient control, while the HMI provides a user-friendly interface for monitoring and manual intervention when necessary, making it a robust and versatile solution for managing water levels in industrial applications.

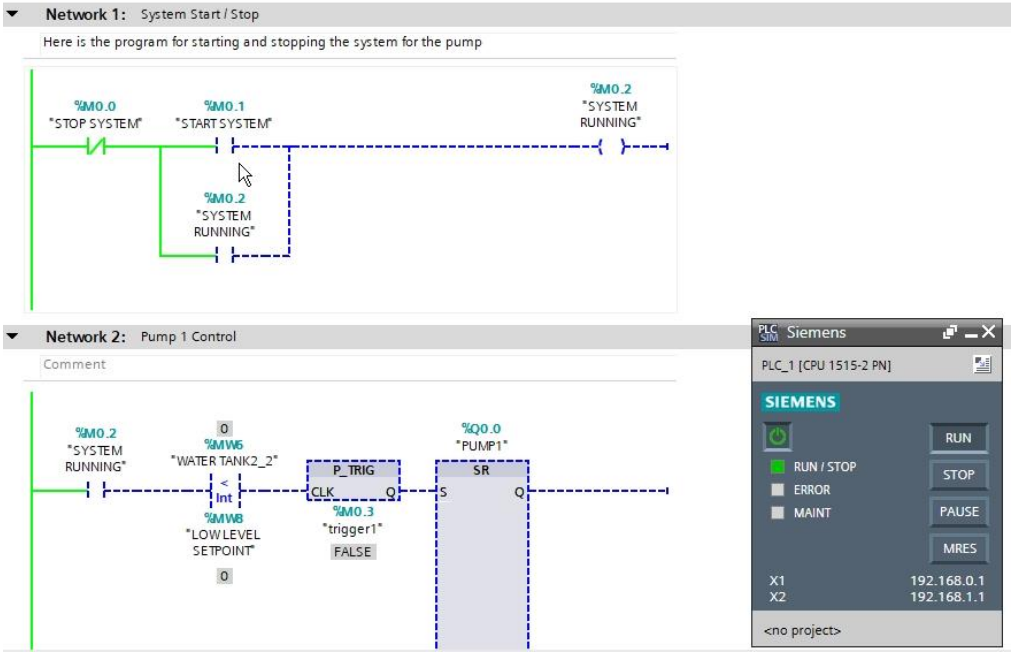


Figure 98 Main [OB1] for the water pumps automatic system

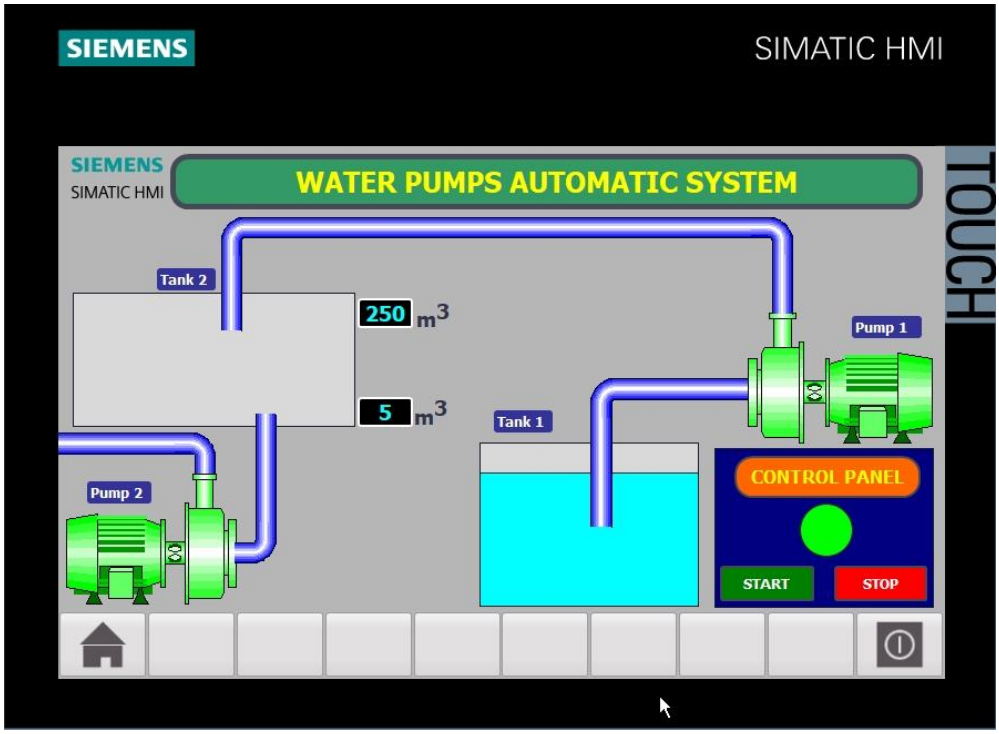


Figure 99 Development of a Human-Machine Interface (HMI) for an Automated Water Pump Control System

Network 1:
This network controls system operation on a top level using "START SYSTEM" and "STOP SYSTEM" buttons mapped to M memory bits. A third bit "SYSTEM RUNNING" indicates overall system status. Using M bits allows Boolean on/off control from HMI pushbuttons. The buttons set and reset the status bit using flip-flop logic for latching control.

Network 2:
This ladder logic controls Pump 1 operation based on tank water levels. "SYSTEM RUNNING" is a permissive condition to allow this logic to execute. Two compare blocks check if tank level MW2 reaches low or high setpoints stored in MW memory. On transition past each setpoint, P_TRIG (positive transition trigger) temporary contacts close to activate start and stop sequences for Pump 1. P_TRIGs are used since we only want to react to specific upward level crossing events. The pump Q0.0 output toggles on and off using SR flip-flops, whose outputs are latched based on S or R inputs. MW memory holds key process values for flexible single word access.

Network 3:
This network creates a simulated water fill/drain animation by continuously incrementing and decrementing counter DB1 while Pump 1 runs. The counter value is mapped to tank level MW2 for operator display. Count up and down rates match the cycle time of two different clock inputs for varied speed animation. Counter control via CTU and CTD increments/decrements upon clock rising edges while enabled.

Network 4:
Fundamentally the same implementation as Network 2, but controlling Pump 2 based on the levels in a second coupled tank. Pump 2 activates at high level in Tank 2 and stops at low level in Tank 2_2 to demonstrate cascade logic.

Network 5:
Creates animation for Tank 2_2 level by decreasing counter DB2 value while Pump 2 runs, and vice versa, using 10Hz clock rate. CTD and CTU counter blocks handle direction control and clock actions.

Tag	Name	Address
STOP SYSTEM	Stop System Button	%M0.0
START SYSTEM	Start System Button	%M0.1
SYSTEM RUNNING	System Running Status	%M0.2
WATER TANK2	Water Level Tank 2	%MW2
HIGH LEVEL SETPOINT	High Level Setpoint	%MW4
WATER TANK2_2	Water Level Tank 2_2	%MW6
LOW LEVEL SETPOINT	Low Level Setpoint	%MW8
trigger1	Trigger for Pump 1 On	%M0.3
PUMP1	Pump 1	%Q0.0
trigger2	Trigger for Pump 1 Off	%M0.4
Clock_5Hz	5Hz Clock	%M2001.1
C0	Counter for Animation	%DB1
trigger6	Trigger for Animation	%M1.0
trigger3	Trigger for Pump 2 On	%M0.5
PUMP2	Pump 2	%Q0.1
trigger5	Trigger for Pump 2 Off	%M0.7
Clock_10Hz	10Hz Clock	%M2001.0
C1	Counter for Animation	%DB2
trigger4	Trigger for Animation	%M0.6

Table 9 List of tags with names and its address of example 6

WaterPumps / PLC_1 [CPU 1515-2 PN] / Program blocks

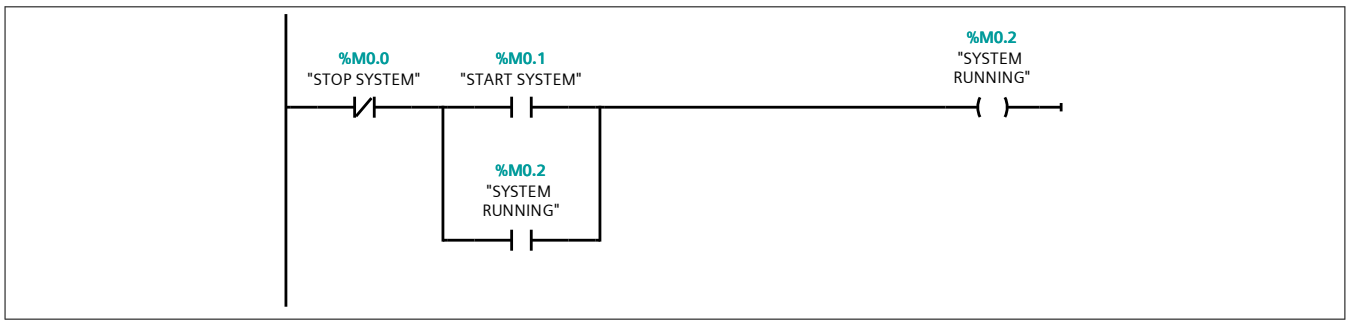
Main [OB1]

Main Properties					
General					
Name	Main	**Number**	1	**Type**	OB
Language	LAD	**Numbering**	Automatic		
Information					
Title	"Main Program Sweep (Cycle)"	**Author**		**Comment**	
Family		**Version**	0.1	**User-defined ID**	

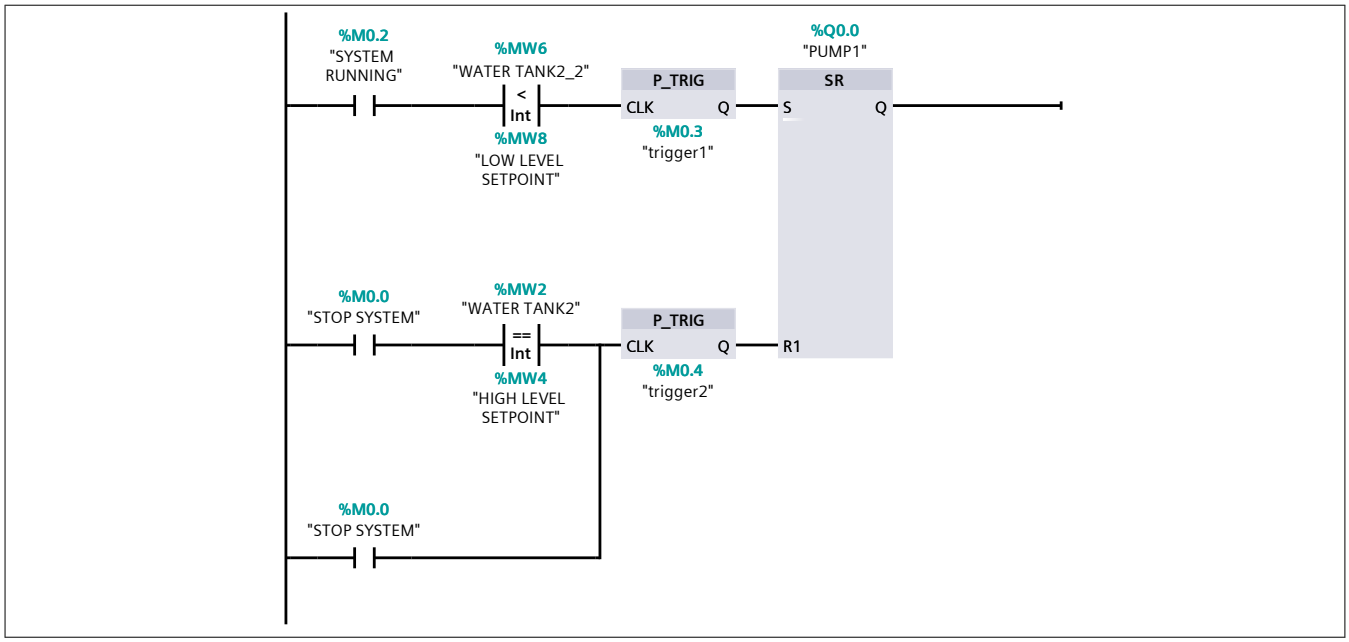
Main			
Name	**Data type**	**Default value**	**Comment**
▼ Input			
Initial_Call	Bool		Initial call of this OB
Remanence	Bool		=True, if remanent data are available
Temp			
Constant			

Network 1: System Start / Stop

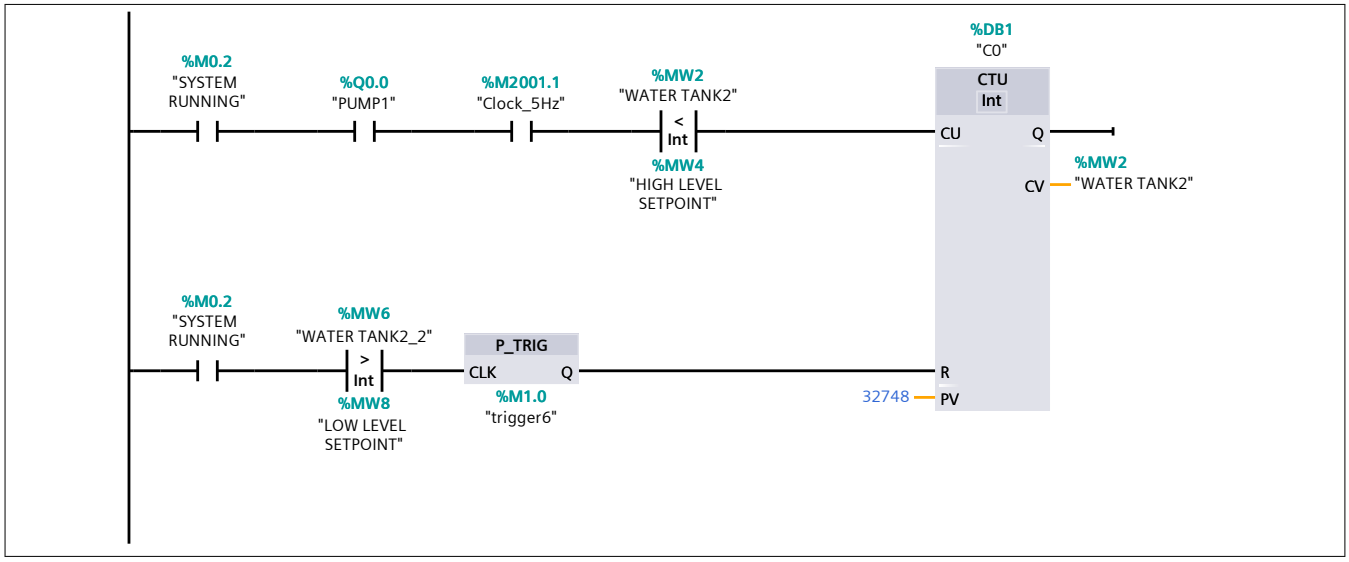
Here is the program for starting and stopping the system for the pump



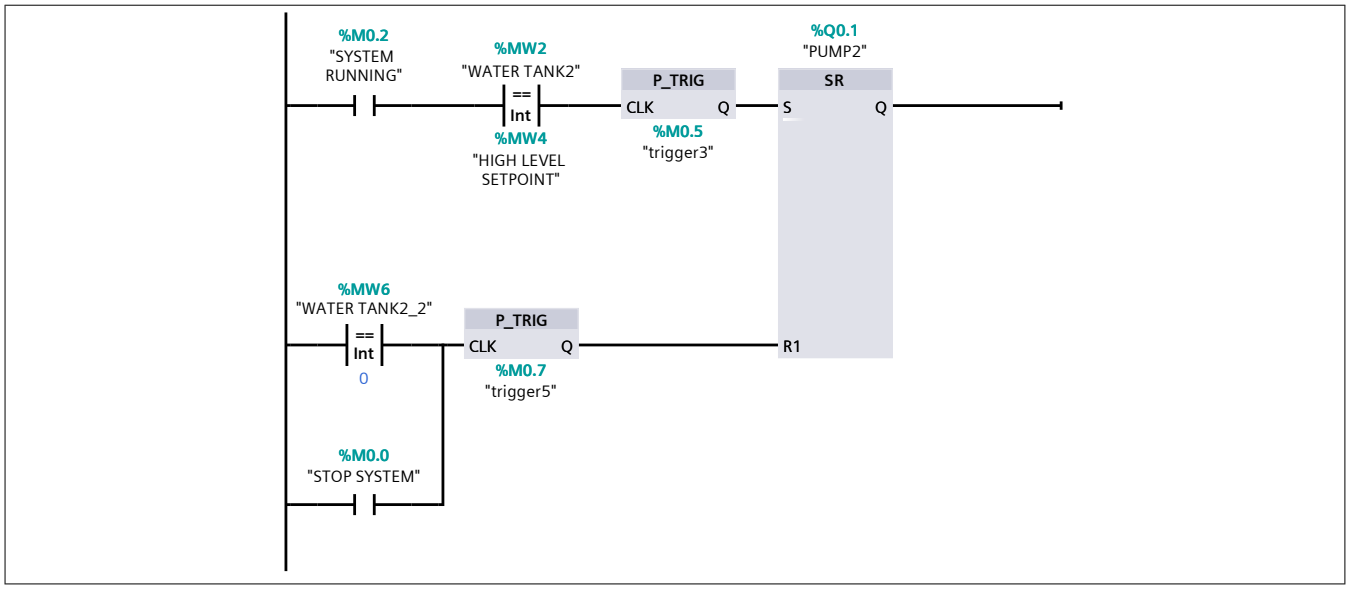
Network 2: Pump 1 Control



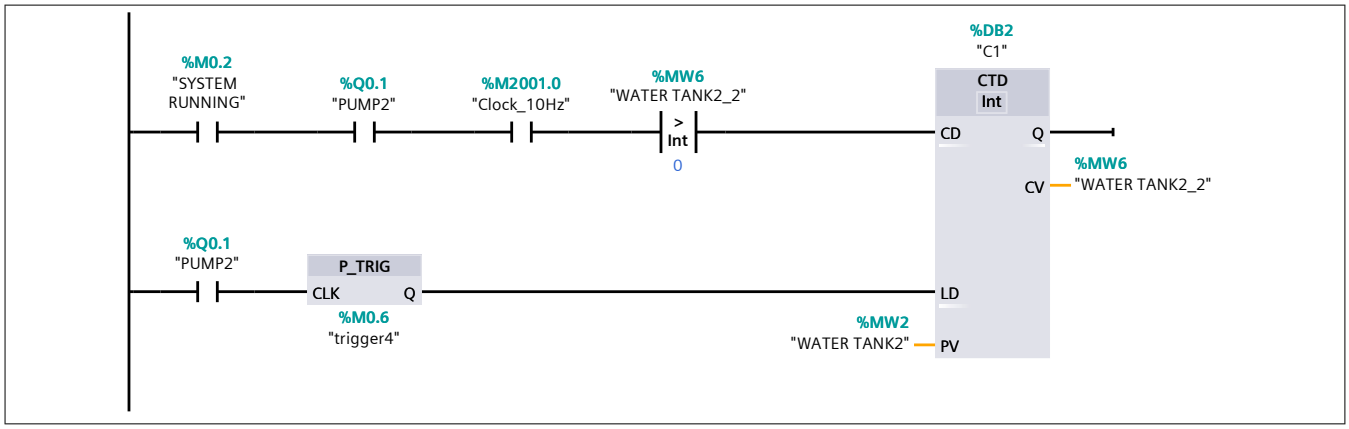
Network 3: Level water animation



Network 4: Pump2 contorl



Network 5: Level Water Animation



Annex - Integration of Festo FluidSim and Siemens TIA Portal through EzOPC for Virtual PLC Simulation

EzOPC is an Open Platform Communication (OPC) server and FluidSim is a powerful simulation tool for pneumatic and hydraulic systems, while TIA Portal is a comprehensive engineering framework for automating industrial processes. By establishing a connection between these two platforms through EzOPC, a software-based industrial control system, we will demonstrate the potential for seamless communication and control between virtual simulations and real-world automation systems.

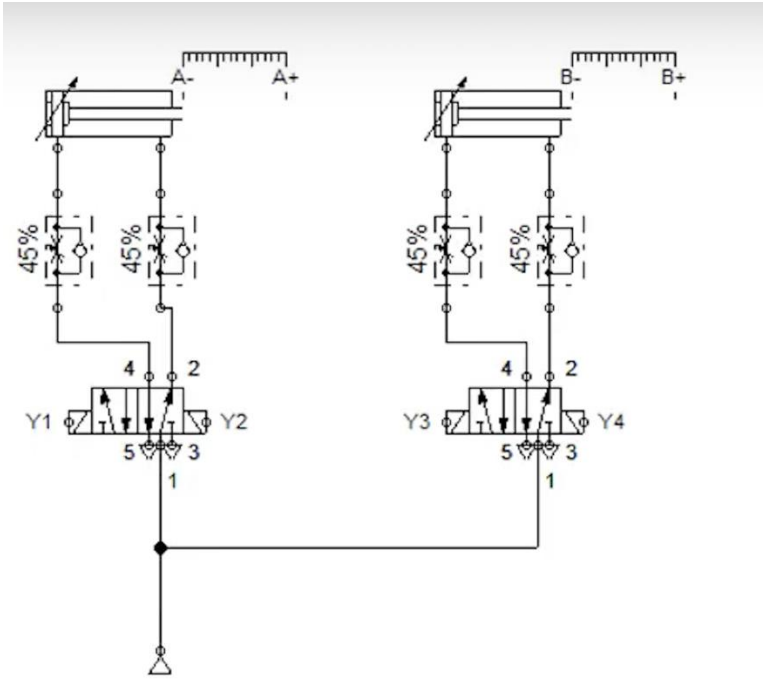
The steps below will involve configuring the EzOPC software to act as an intermediary between FluidSim and TIA Portal, enabling bidirectional data exchange. This integration will facilitate the transfer of simulation data from FluidSim to TIA Portal, allowing for the control and monitoring of simulated components within the TIA Portal environment. Conversely, control commands generated in TIA Portal can be transmitted back to FluidSim, enabling real-time visualization and validation of the simulated system's behavior.

As a practical application, the example will focus on controlling a simulated pneumatic cylinder in FluidSim using the TIA Portal programming environment. The following steps will be covered:
Steps for Controlling a Cylinder for Opening and Closing:

1. Set up the FluidSim environment by creating a pneumatic circuit with a cylinder and necessary components (e.g., valve, compressor, tubing). It's important to note that FluidSim provides a wide range of components and configuration options, allowing us to create complex pneumatic circuits and simulate various scenarios. The specific steps and components used may vary depending on the complexity of our system and the desired level of detail in the simulation. Additionally, we can incorporate advanced features like programmable logic controllers (PLCs) or other control systems within FluidSim to automate the process sequence and integrate with other software platforms, such as TIA Portal or EzOPC, as described in this thesis.

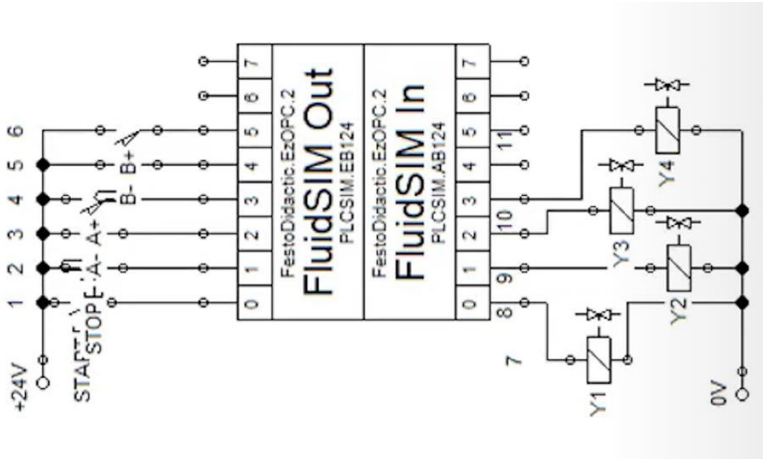
To set up the FluidSim software environment with two cylinders (Cylinder A and Cylinder B) and simulate the process sequence of A+B+B-A-, follow these steps:

i. Launch the Festo FluidSim software and create a new circuit diagram.
ii. From the component library, drag and drop a pneumatic compressor onto the circuit diagram. This will serve as the air supply for the system.
iii. Add two double-acting cylinders (Cylinder A and Cylinder B) from the component library to the circuit diagram. Position them in a suitable location.
iv. Connect the compressor to the inlet ports of both cylinders using pneumatic tubing components from the library.
v. Add two 5/2-way solenoid valves (one for each cylinder) to control the direction of airflow and, consequently, the movement of the cylinders.
vi. Connect the solenoid valves to the respective cylinder ports using appropriate tubing components.
vii. Connect the exhaust ports of the solenoid valves to the atmosphere using silencers or exhaust components from the library.
viii. Add any necessary auxiliary components, such as air filters, pressure regulators, or flow control valves, as per our requirements.
ix. Connect the electrical control signals for the solenoid valves. This can be done by adding electrical components (e.g., switches, sensors) and wiring them to the respective solenoid valve coils.
x. Configure the electrical control components to simulate the desired process sequence of A+B+B-A-, where: A+: Extend Cylinder A, B+: Extend Cylinder B, B-: Retract Cylinder B, A-: Retract Cylinder A
xi. Set up the necessary sensors or switches to trigger the appropriate valve actions for each step of the sequence.
xii. Adjust the cylinder stroke lengths, piston diameters, and other parameters as needed to match our requirements.
xiii. Connect the necessary pressure gauges, flow meters, or other instrumentation components to monitor the system's behavior.
xiv. Once the circuit is complete, we can run simulations to observe the behavior of the two cylinders and validate the process sequence.



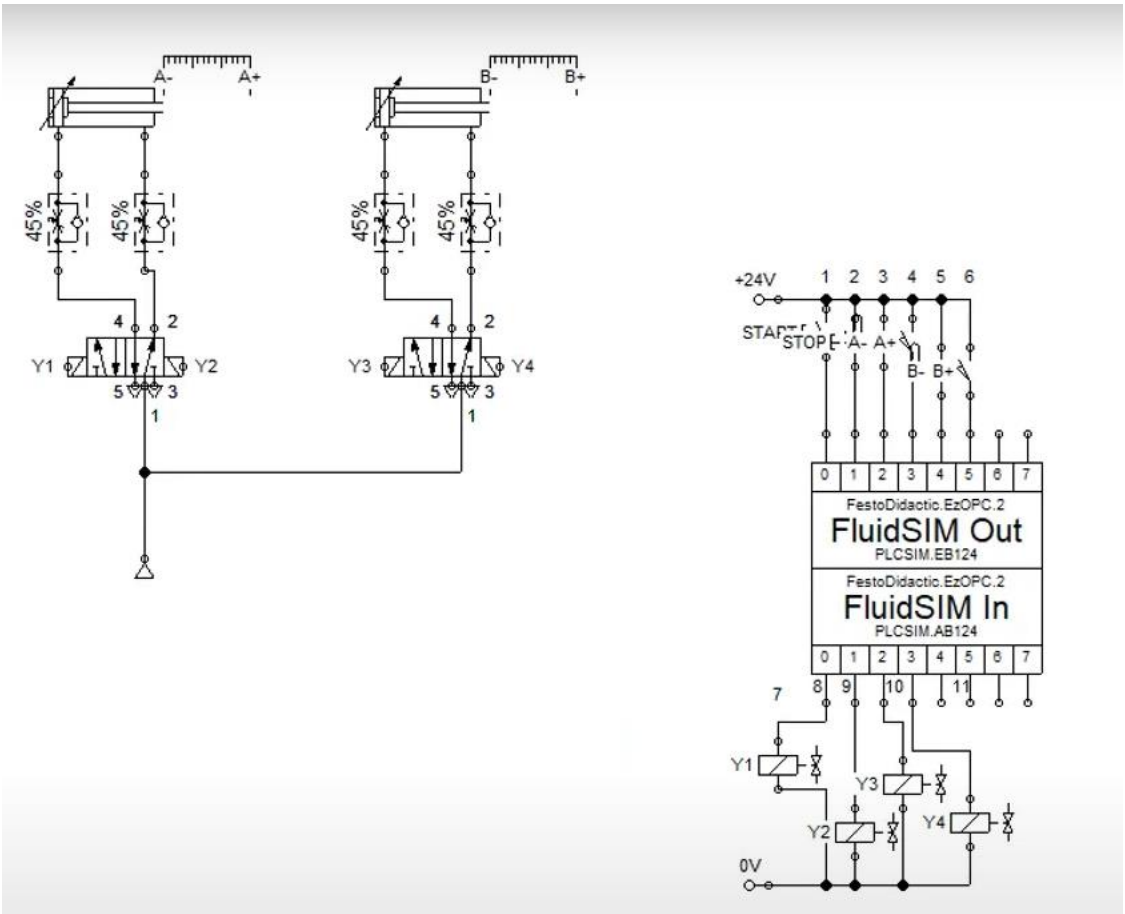
FluidSim provides virtual binary inputs (FluidSim-out) and binary outputs (FluidSim-in) in its component library. These interfaces represent the transmission of binary and analog signals to and from the simulated system, respectively. To enable communication between FluidSim and TIA Portal (or any other external system like EzOPC), we need to include the FluidSim-Out and FluidSim-In components in the pneumatic circuit. Here's how we can incorporate these components:

a. From the FluidSim component library, locate and add the " FluidSim Out" component to your circuit diagram.
b. Connect the " FluidSim Out" component to the appropriate control signals or sensors that you want to monitor or exchange data with the external system (TIA Portal or EzOPC).
c. Similarly, add the " FluidSim In" component to the circuit diagram.
d. Connect the " FluidSim In" component to the appropriate actuators or control elements (e.g., solenoid valves) that you want to control from the external system.
e. Configure the properties of the "FluidSim Out" and " FluidSim In" components to specify the data exchange parameters, such as IP addresses, ports, and communication protocols.
f. In the external system (TIA Portal or EzOPC), set up the corresponding communication interface to establish a connection with FluidSim using the configured parameters.
g. Map the input and output signals from FluidSim to the corresponding addresses or tags in the external system, allowing for bidirectional data exchange.



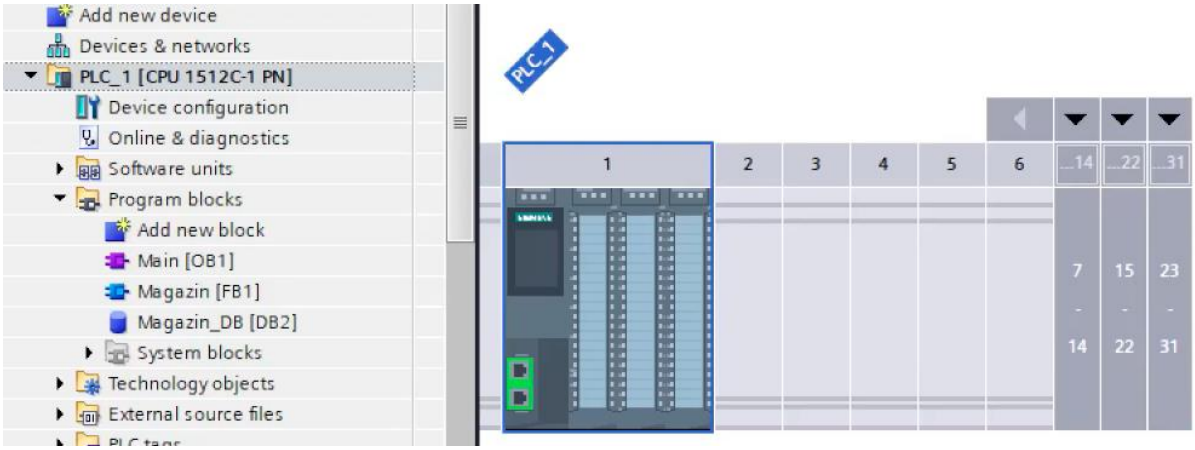
With the "FluidSim Out" component, you can send data from FluidSim to the external system, such as sensor readings, cylinder positions, or any other relevant information. Conversely, the "FluidSim In" component allows you to receive control commands or setpoints from the external system and apply them to the actuators or control elements within the FluidSim simulation.

This integration of the FluidSim Out and FluidSim In components is crucial for enabling communication between FluidSim and TIA Portal. EzOPC establishes a bridge between the virtual simulation environment and the real-world automation system, allowing for seamless data exchange and control capabilities.

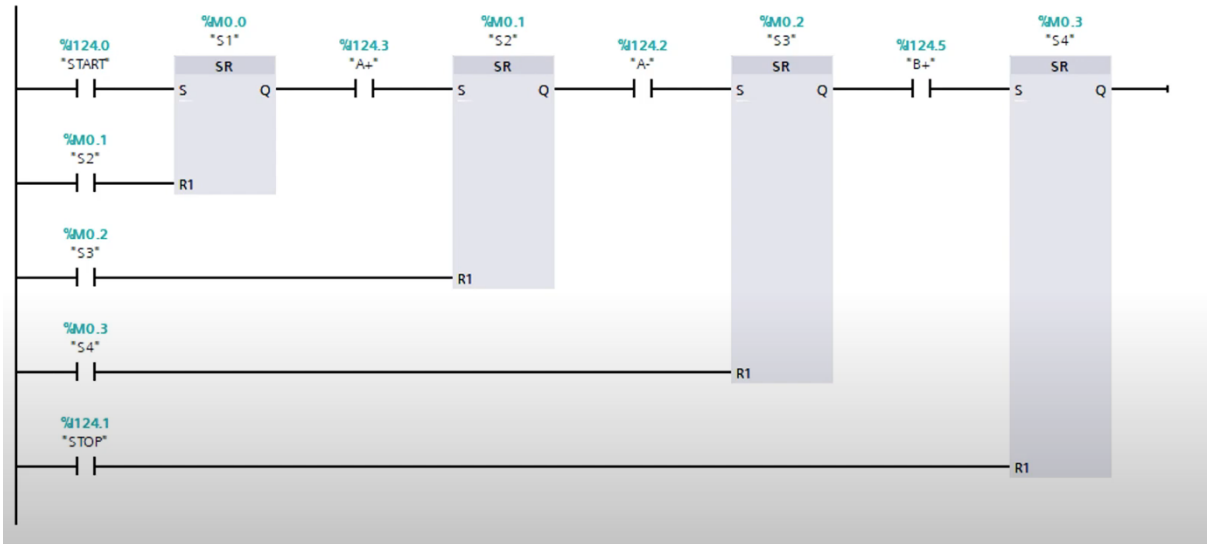


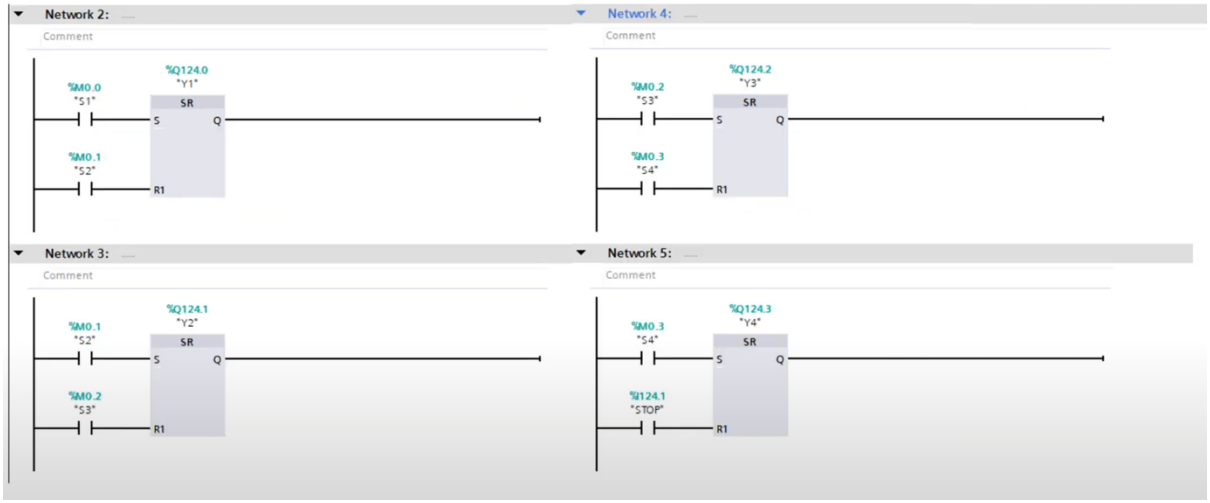
2. Configure the EzOPC software to establish communication with FluidSim and TIA Portal. EzOPC serves as the OPC server, facilitating data exchange between FluidSim and PLCSIM Advanced or PLCSIM, which act as OPC clients. The configuration involves setting the data flow between the two programs, defining the I/O areas of the simulated controller, and specifying the start addresses and number of bytes/words for binary and analog signals.

3. In TIA Portal, create a new project and add the necessary hardware components (e.g., PLC, I/O modules) to the configuration. Within TIA Portal, a supported controller is selected in the device configuration, and the address ranges for binary and analog I/O are set. The PLC program is developed based on the desired control logic, utilizing the signals from FluidSim as inputs and sending outputs to control the simulated actuators.



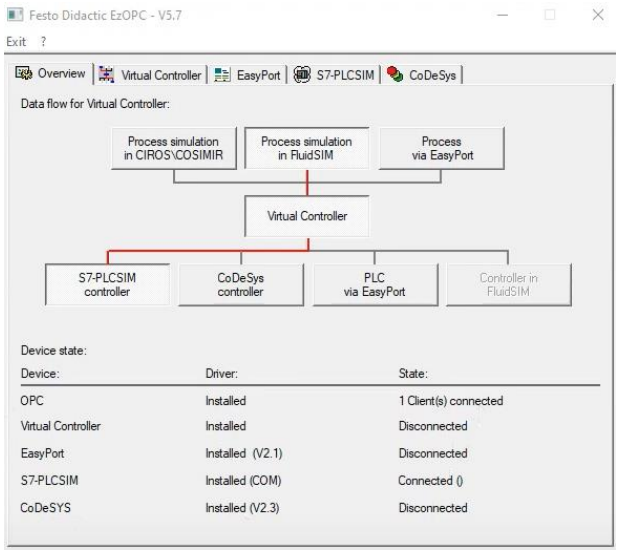
4. Program the PLC logic in TIA Portal to control the simulated cylinder's movement. This can be done using ladder logic, function block diagrams, or structured text.



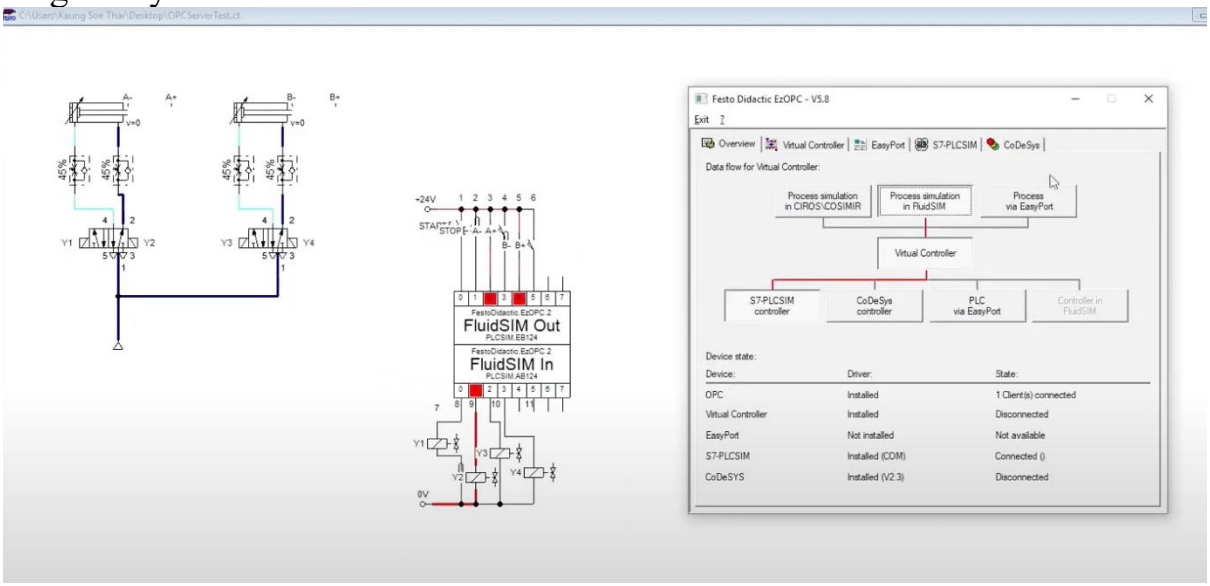


5. Map the input and output signals from the simulated cylinder in FluidSim to the corresponding I/O addresses in TIA Portal through the EzOPC software.

6. Establish the communication between TIA Portal and EzOPC, enabling the exchange of control signals and feedback data.



7. Test and validate the cylinder control logic by executing the program in TIA Portal and observing the cylinder's behavior in the FluidSim simulation.



8. Analyzing and optimizing the control algorithm based on the simulation results, considering factors such as response time, accuracy, and potential issues like cylinder bouncing or sticking.

9. Documenting the findings, highlighting the benefits and challenges of integrating FluidSim simulations with real-world automation systems using TIA Portal and EzOPC.