

Contents

| | | |
|-----------|--|------------|
| 5. | PLC Tags..... | 5-2 |
| 5.1. | What are Tags and Why do You Need them? | 5-3 |
| 5.1.1. | Addresses of PLC Tags | 5-4 |
| 5.2. | Elementary Data Types Bit and Numeric..... | 5-5 |
| 5.2.1. | Elementary Data Types Date, Time and Character | 5-6 |
| 5.3. | Declaration and Definition of PLC Tags..... | 5-8 |
| 5.3.1. | PLC Tags and PLC Constants..... | 5-9 |
| 5.3.2. | PLC Tags in the Device View | 5-11 |
| 5.3.3. | Finding / Replacing / Sorting PLC Tags..... | 5-12 |
| 5.3.4. | Error Indication in the PLC Tag Table | 5-13 |
| 5.3.5. | Copy & Paste PLC Tags to Excel | 5-14 |
| 5.3.6. | Retentiveness of PLC Tags | 5-15 |
| 5.4. | Details View of PLC Tags | 5-16 |
| 5.5. | Monitoring PLC Tags | 5-17 |
| 5.5.1. | Modifying PLC Tags by means of the Watch Table..... | 5-18 |
| 5.6. | Exercise 1: Copying the PLC Tag Table from the Library | 5-19 |
| 5.6.1. | Exercise 2: Creating the "Conveyor" Tag Table | 5-20 |
| 5.6.2. | Exercise 3: Monitoring the "Conveyor" PLC Tag Table..... | 5-21 |
| 5.6.3. | Exercise 4: Modifying using the Watch Table..... | 5-22 |

5. PLC Tags

At the end of the chapter the participant will ...



- ... be familiar with PLC tags and their memory areas
- ... be able to create PLC tags and address them
- ... know elementary data types
- ... be familiar with global constants and system constants
- ... be able to apply the details view
- ... learn how to monitor and modify PLC tags

5.1. What are Tags and Why do You Need them?

A tag...

- is a memory space on the CPU
- saves values/data
- is described by its name, memory area, address and data type (size, possible value range, use, allowed instructions)

Memory areas of PLC tags:

- Process (image) tags for **I**ntputs or **Q** Outputs
 - Auxiliary tags for saving values in the **M**emory byte area
 - CPU-internal **C**ount area and Time area (**T**imer)
→ not available for S7-1200
-
- Process state
- Process control
- e.g. clock memory (byte)
- Simatic counters and timers for simple but limited count and time functions

The Importance of PLC Tags

Next to commands, tags (variables) are the most important elements of a programming system. Their task is to save values in a program so that they can be further processed at a later time.

Data Types

The data type determines which values are accepted by data and which instructions can be carried out with these values.

Memory Areas of Inputs and Outputs

Within a controller, the data is stored in different memory areas. The input signals of input modules are stored in the process image for inputs where they can be consistently read out throughout a cycle. The control of the process happens via the process image for outputs which is then written to the output modules.

Memory Byte Area

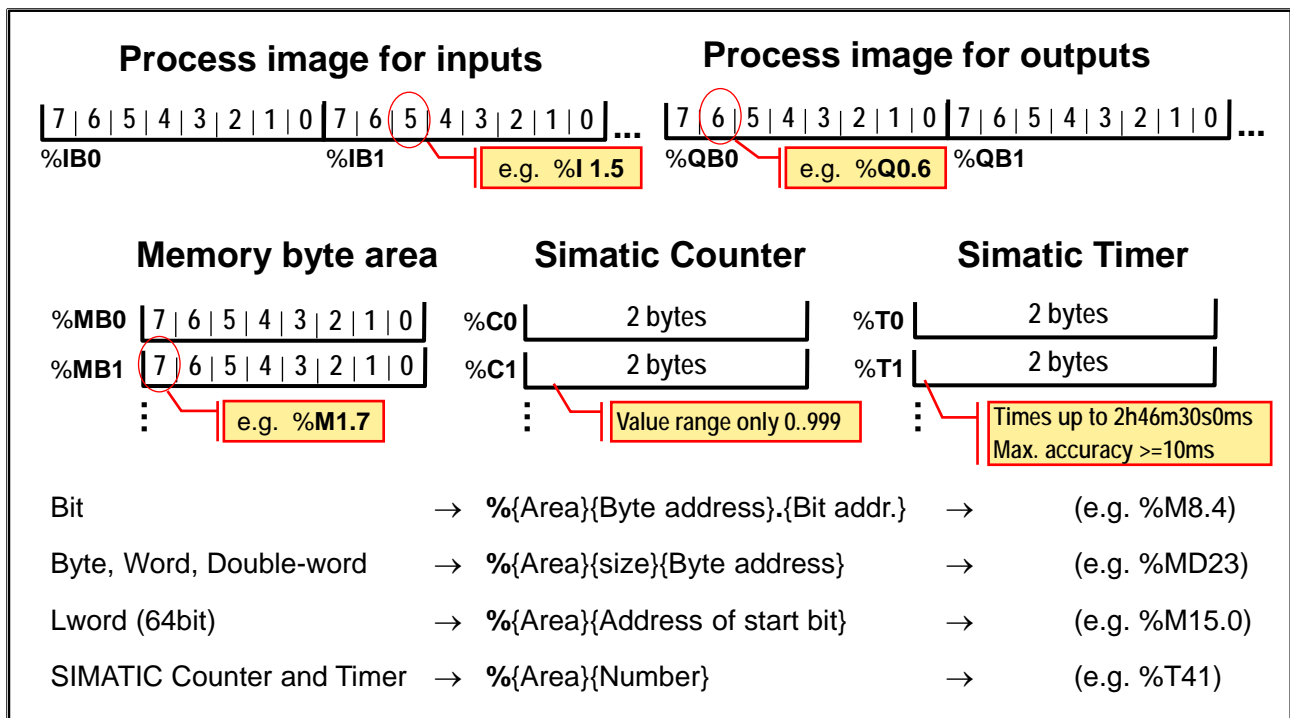
Internally, there is an additional storage area in which values can be saved. Since the clarity and the re-usability of individual tags (variables) of the memory byte area are not fulfilled, there are still other areas which will be discussed in later chapters.

CPU-internal Count and Time Areas (SIMATIC Counter and Timer)

These are a simple way to use count and time functions. SIMATIC counters and timers are, however, very limited. The counters only have a value range of 0 to 999 and the timers can only record times up to 2 hours 46 minutes and 30 seconds with an accuracy of 10 milliseconds and are not available on 1200-series CPUs.

For greater demands on timers and counters there are functions which also will be dealt with in later chapters.

5.1.1. Addresses of PLC Tags



Addressing

An address exactly defines where values are written or read.

They begin at byte address, that is, at the number "0".

The addresses are consecutively numbered, and, within a byte, the bit address 0..7 is numbered from right to left.

Address Style With/Without %

The %-character identifies the presentation of an address. It can be left out during address entry since it is automatically added by the engineering framework.

PLC Tags > 32 Bits:

For tags that are greater than 32 bits (e.g. LWORD), only the start bit is specified during addressing since this can only be accessed symbolically. The required number of bytes that must be reserved for these tags results from the data type of the PLC tag.

5.2. Elementary Data Types

Bit and Numeric

| | Description | Size (Bit) | S7-1200 | S7-1500 | Example |
|--------------------|-------------|------------|---------|---------|-------------------|
| Bit Data Types | BOOL | 1 | | | TRUE |
| | BYTE | 8 | ✓ | ✓ | B#16#F5 |
| | WORD | 16 | | | W#16#F0F0 |
| | DWORD | 32 | | | DW#16#F0F0FF0F |
| | LWORD | 64 | ✗ | ✓ | LW#16#5F52DE8B |
| Numeric Data Types | SINT | 8 | | | 50 |
| | USINT | | | | 20 |
| | INT | 16 | | | -23 |
| | UINT | | | | 64530 |
| | DINT | 32 | ✓ | ✓ | DINT# -2133548520 |
| | UDINT | | | | UDINT#435676 |
| | REAL | | | | 1.0 |
| | LREAL | 64 | | | LREAL#-1.0e-5 |
| | LINT | 64 | ✗ | ✓ | LINT#1543258759 |
| | ULINT | | | | ULINT#154316159 |

BOOL, BYTE, WORD

Variables of the data type BOOL consist of one bit. Variables of the data types BYTE and WORD are bit sequences of 8 or 16 bits. The individual bits are not evaluated in these data types. Special forms of these data types are the BCD numbers and the count value as it is used in conjunction with the count function.

Note:

In the illustration above, **#16#** denotes that the value is represented in Hexadecimal format.

INT, REAL

Variables of these data types represent numbers with which relevant arithmetical calculation operations can be carried out. (INT → integer, REAL → floating point number)

Extensions of INT, REAL and WORD

U - Unsigned

Variables with the extension "U" represent a variable without sign of the relevant data type. **Data types:** USINT, UINT, ULINT, UDINT

S - Short

Variables with the extension "S" represent a variable with a length of 8 bits of the relevant data type. **Data types:** SINT, USINT

D - Double

Variables with the extension "D" represent a variable with a length of 32 bits of the relevant data type. **Data types:** DWORD, DINT, UDINT

L- Long

Variables with the extension "L" represent a variable with a length of 64 bits of the relevant data type. **Data types:** LWORD, LINT, ULINT, LREAL

5.2.1. Elementary Data Types Date, Time and Character

| | Description | Size (Bit) | S7-1200 | S7-1500 | Example |
|-----------------------|----------------------|------------|---------|---------|-----------------------------------|
| Time Types | TIME | 32 | | | T#2h46m30s630ms |
| | DATE | 16 | ✓ | ✓ | D#1994-01-21 |
| | TIME_OF_DAY | 32 | | | TOD#18:15:18:999 |
| | S5TIME | 16 | | | S5T#1h24m10s |
| | LTime | 64 | ✗ | ✓ | LT#11350d20h25m14s830ms652µs315ns |
| | LTIME_OF_DAY | | | | LTOD#10:20:30.400_365_215 |
| | LDT (DATE_AND_LTIME) | | | | LDT#2008-10-25-08:12:34.567 |
| Character Type | CHAR | 8 | ✓ | ✓ | 'R' |
| | WCHAR | 16 | | | WCHAR#'w' |

TIME, LTIME

A variable of the data type TIME (duration in [ms]) occupies a double-word. This variable is used, for example, for specifying time values in IEC timer functions. The contents of the variable are interpreted as a DINT number in milliseconds and can be either positive or negative (for example: T#1s=L#1000, T#24d20h31m23s647ms = L#2147486470).

Just like TIME, LTIME represents a duration whereby the duration is saved with nanosecond resolution in 64 bits. That means, compared with the data type TIME, longer durations with greater resolution can be saved in variables of the data type LTIME.

DATE

A variable of the data type DATE is stored in a word in the form of an unsigned integer. The contents of the variable represent the number of days since 01.01.1990.

TIME_OF_DAY, LTIME_OF_DAY

The data type TOD (TIME_OF_DAY) occupies a double-word and stores the number of milliseconds since the beginning of the day (0:00 o'clock) as an unsigned integer. Variables of the data type LTOD occupy two double-words and state the number of nanoseconds since the beginning of the day.

LDT (Date_AND_LTIME)

The data type LDT (DATE_AND_LTIME) occupies 8 bytes and stores information on date and time in nanoseconds since 01.01.1970 0:00.

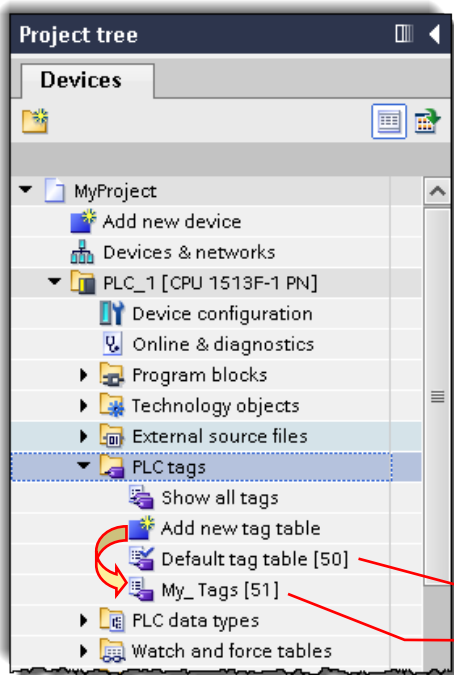
S5TIME

Variables of the data type S5TIME are required for specifying time values in timer functions (S5-timing elements). You specify the time in hours, minutes, seconds or milliseconds. You can enter the timer values with an underline (1h_4m) or without an underline (1h4m).

CHAR, WCHAR

The data type CHAR (8 bits) represents a character in ASCII representation and WCHAR (16 bits) a character in Unicode format.

5.3. Declaration and Definition of PLC Tags



Declaration of a tag:

- create/introduce (any)
- define name and data type

Definition of a tag:

- allocation of a memory area (memory area letter, size and address)

Creating a new PLC tag:

1. Add a new tag table or open an already existing table
2. Declare the PLC tag (name, data type)
3. Define the PLC tag (address)

PLC Tag Tables

In order to achieve a good readability of the CPU program, it makes sense to structure the data storage. For this, there are PLC tag tables for the PLC tags.

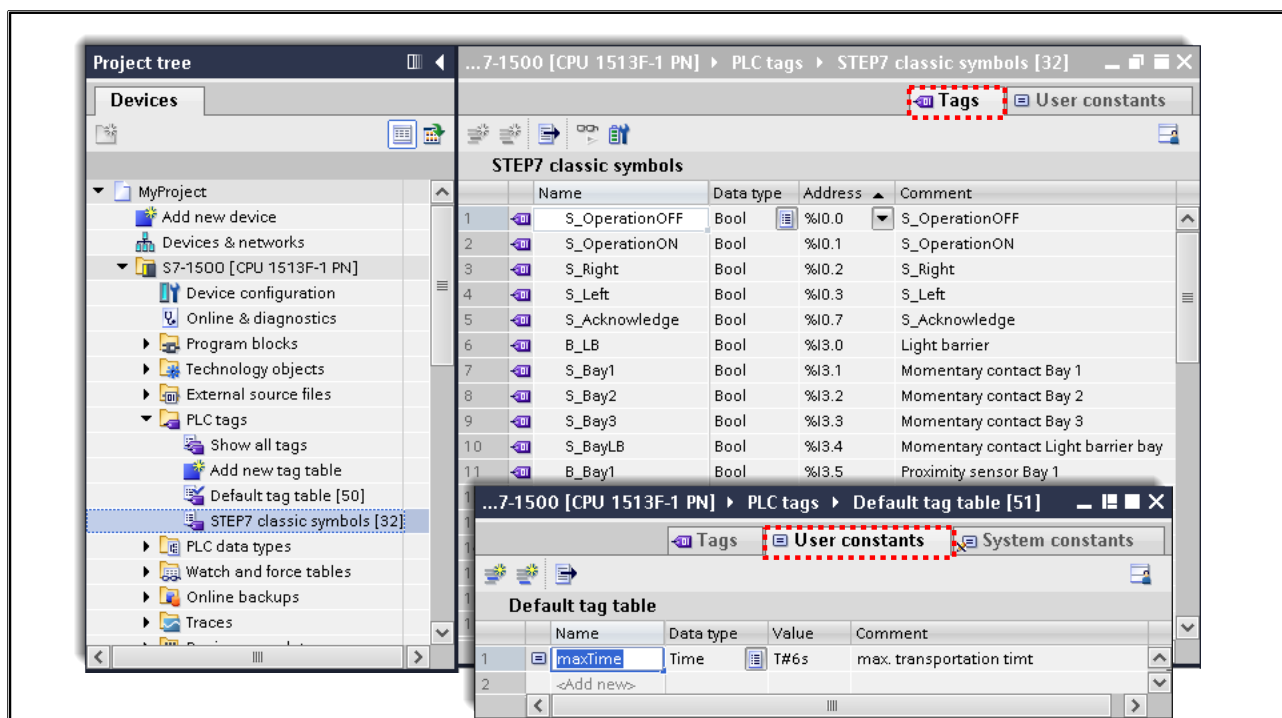
Tag Tables

The default tag table (name can be changed) contains additional CPU information and thus cannot be deleted. It can be used for any PLC tags, but for better clarity, it is recommended that several tag tables be created.

Declaration and Definition of PLC Tags

In the declaration of a tag in the PLC tag table, the symbolic name (for example, "M_Jog_Right"), the data type (for example, Bool) and the absolute address (for example, M16.2) are defined.

5.3.1. PLC Tags and PLC Constants



PLC Tags

The PLC tag table contains the declaration (definition) of CPU-wide valid and thus global tags and constants. For each CPU added in the project, a PLC tag table is automatically created. A PLC tag table contains one tab each for Tags and User constants; the Default tag table also contains a tab for System constants. Tags are operands with changeable content used in the user program.

User Constants

A constant defines an unchangeable data value. During program execution, constants can be read by various program elements, but they cannot be overwritten. Changing the constant value while the program is running is not possible.

In TIA Portal, it is possible to declare symbolic names for constants so as to make static values available in the program under one name. These symbolic constants are valid throughout the CPU. The declaration of the constants is made in the "User constants" tab of the PLC tag table. Constants are operands with unchangeable content, and in addition, constants do not require an absolute address.

Creating Tags and Constants with Group Function

By clicking on the "Fill" symbol in the lower right corner of the cell and then dragging it down, tags and constants are automatically created (comparable to Excel).

It is possible to automatically create tags and constants through the "Name" and "Address" (only for tags) columns. The new tags and constants are created with the name of the current tag/constant appended by a consecutive number. From a tag or constant with the name "T_Station", new tags/constants are then created with the names "T_Station_1", "T_Station_2" etc.

System Constants

System constants are CPU-wide unique, global constants which are required by the system and are automatically created. System constants can, for example, serve the addressing and identification of hardware objects.

Rules

System constants are automatically assigned in the Device view or Network view when components are inserted and are entered in the Default tag table ("System constants" tab). A system constant is created for each module but also for each submodule. In that way, for example, an integrated counter is also given a system constant. System constants consist of a symbolic name as well as a numeric HW-ID and cannot be changed.

System Constant Names

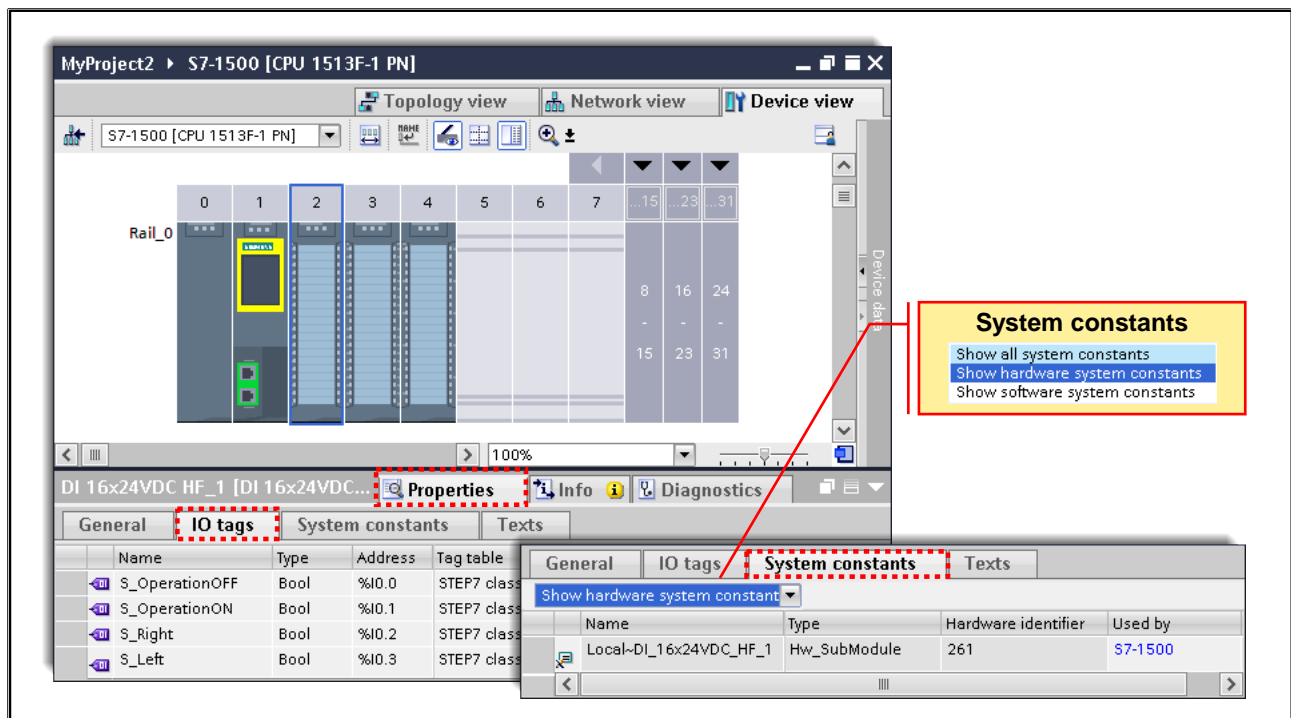
The names of system constants are hierarchically structured. They consist of a maximum of four hierarchy levels which in each case is separated by a tilde "~". In this way, you can recognize the "path" to the relevant hardware module based on the name.

| <div> <div>Tags</div> <div>User constants</div> <div>System constants</div> </div> | | | | | |
|---|-----------------------------------|--------------|-------|---------|--|
| Default tag table | | | | | |
| | Name | Data type | Value | Comment | |
| 39 | Local~Display | Hw_SubModule | 54 | | |
| 40 | Local~Exec | Hw_SubModule | 52 | | |
| 41 | Local | Hw_SubModule | 49 | | |
| 42 | Local~FExec | Hw_SubModule | 55 | | |
| 43 | Local~PROFINET_interface_1 | Hw_Interface | 64 | | |
| 44 | Local~PROFINET_interface_1~Port_1 | Hw_Interface | 65 | | |
| 45 | Local~PROFINET_interface_1~Port_2 | Hw_Interface | 66 | | |

Example

A system constant with the name "Local~PROFINET_interface_1~Port_1" denotes Port 1 of the PROFINET interface 1 of the local CPU.

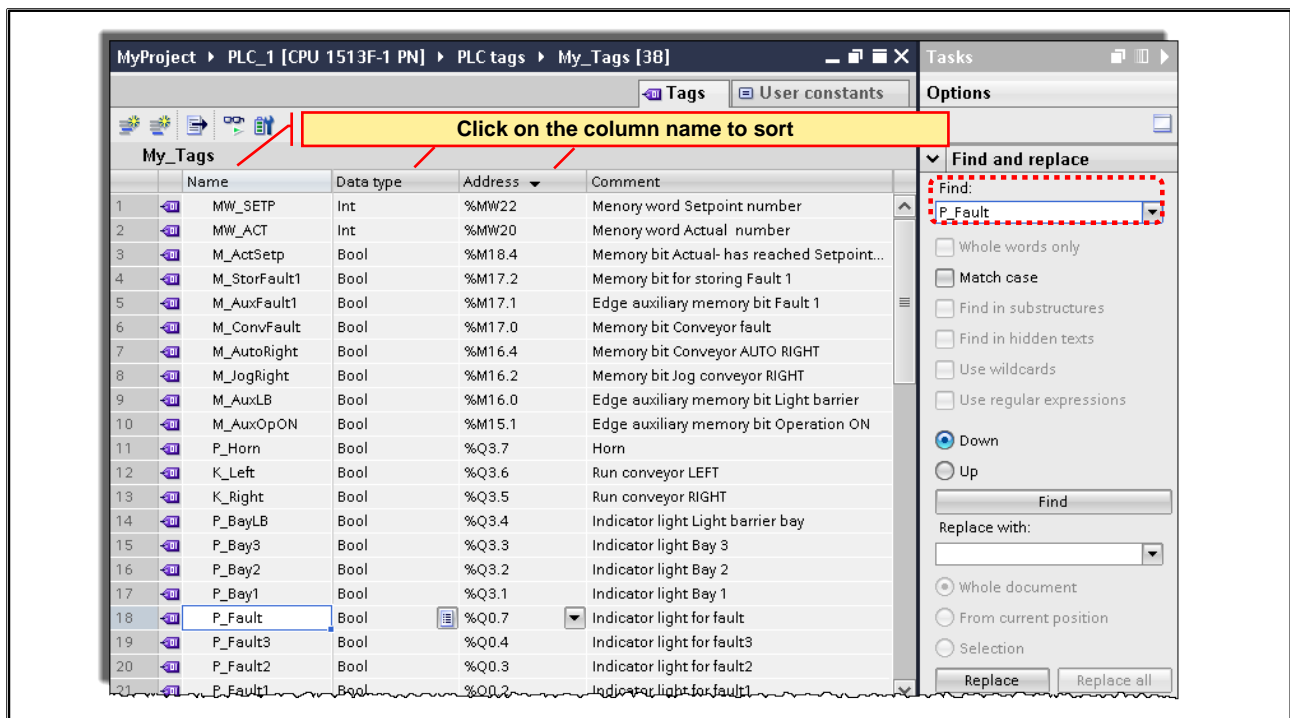
5.3.2. PLC Tags in the Device View



The PLC tags of inputs and outputs can also be declared and changed in the Device view. In the Properties, in the Inspector window you open the IO tags tab for this.

In addition, the system constants of the selected hardware are displayed in the "System constants" tab.

5.3.3. Finding / Replacing / Sorting PLC Tags



Sorting

By clicking on one of the column names "Name", "Data type" or "Address", the tags are sorted alphabetically or according to address (ascending or descending) depending on the column.

Finding / Replacing

In the PLC tag table, tags can be found and replaced via the "Tasks" Task Card. Dummies can also be used (? for one character, * for several characters).

Example of "Find and replace":

Assign byte address 4. to all outputs with byte address 8.:

Find: Q 8. and Replace with: Q 4.

5.3.4. Error Indication in the PLC Tag Table

The screenshot displays the 'STEP7 classic symbols' window in SIMATIC TIA Portal. The window shows a table of PLC tags with columns for Name, Data type, Address, Retain, and various accessibility and visibility options. Several errors are highlighted with red and yellow boxes and arrows:

- Red dashed box:** Encloses rows 3 and 4, highlighting the tag 'B_LB' (address %I3.0) and 'B_LB(1)' (address %I3.6). A red arrow points from the text 'If the name (symbol) already exists in the table, a "(1)" is automatically added' to the '(1)' in the name.
- Red box with '!!!':** Points to the address '%I3.6' for tag 'B_LB(1)'. A red arrow points from the text 'The absolute address is not compatible with the data type of the tag!' to this address.
- Yellow box:** Points to the address '%I80' for tag 'S_Left'. A red arrow points from the text 'Absolute address is used twice!' to this address.
- Red box with '!!!':** Points to the address '%I3.3' for tag 'P_Bay3'. A red arrow points from the text 'Absolute address is used twice!' to this address.

| | Name | Data type | Address | Retain | Accessible from HMI/OPC UA | Writable fr... | Visible in HMI engineering |
|----|-----------------------|-----------|---------|-------------------------------------|-------------------------------------|-------------------------------------|-------------------------------------|
| 1 | B_Bay1 | Bool | %I3.5 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| 2 | B_Bay3 | Bool | %I3.7 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| 3 | B_LB | Bool | %I3.0 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| 4 | B_LB(1) | Bool | %I3.6 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| 5 | K_Left | Bool | %Q3.6 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| 6 | K_Right | Bool | %Q3.5 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| 7 | S_Left | Bool | %I80 | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| 8 | M_2Hz | Bool | %M10.3 | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| 9 | M_AutoRight | Bool | %M16.4 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| 10 | M_AuxOperationON | Bool | %M15.1 | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| 11 | M_AuxStop | Bool | %M16.1 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| 12 | M_ConveyorFault | Bool | %M17.0 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| 13 | M_JogRight | Bool | %M16.2 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| 14 | OD_ConveyorMonitoring | Timer | %T17 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| 15 | P_Bay1 | Bool | %Q3.1 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| 16 | P_Bay2 | Bool | %Q3.2 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| 17 | P_Bay3 | Bool | %I3.3 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| 18 | P_BayLB | Bool | %I3.3 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |
| 19 | P_Fault | Bool | %Q0.7 | | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> | <input checked="" type="checkbox"/> |

Syntax Check

With every entry, there is a syntax check in which existing errors are displayed in RED, or for warnings in YELLOW. A still faulty PLC tag table can be saved but as long as it is still faulty, the program cannot be compiled and downloaded into the CPU.

5.3.5. Copy & Paste PLC Tags to Excel

Import from Excel only in "Show all tags" tag table

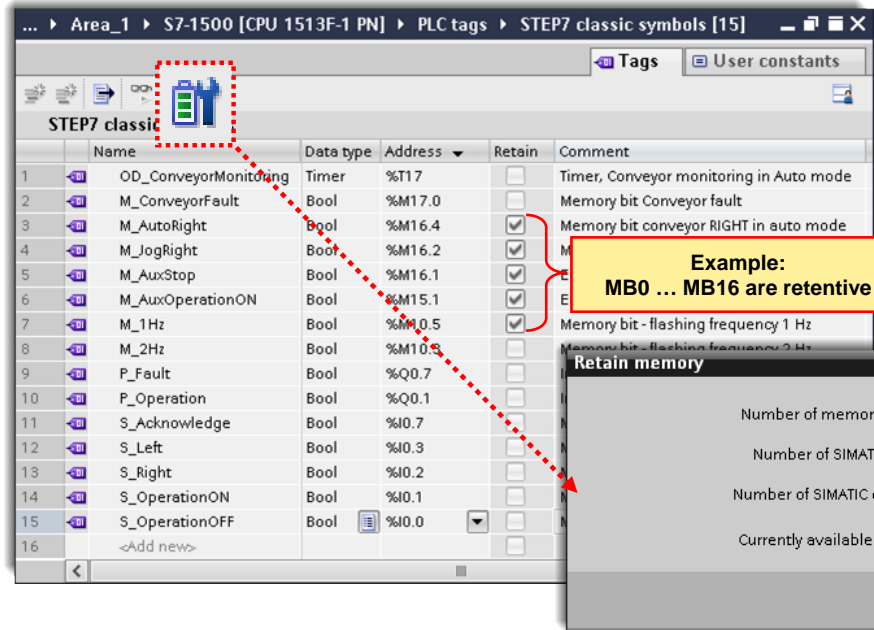
Export from all tag tables to Excel

| | Name | Tag table | Data type |
|----|----------------|-----------------------|-----------|
| 1 | S_OperationOFF | STEP7 classic symbols | Bool |
| 2 | S_OperationON | STEP7 classic symbols | Bool |
| 3 | S_Right | STEP7 classic symbols | Bool |
| 4 | S_Left | STEP7 classic symbols | Bool |
| 5 | S_Acknowledge | STEP7 classic symbols | Bool |
| 6 | B_LB | STEP7 classic symbols | Bool |
| 7 | S_Bay1 | STEP7 classic symbols | Bool |
| 8 | S_Bay2 | STEP7 classic symbols | Bool |
| 9 | S_Bay3 | STEP7 classic symbols | Bool |
| 10 | S_BayLB | STEP7 classic symbols | Bool |
| 11 | B_Bay1 | STEP7 classic symbols | Bool |
| 12 | B_Bay2 | STEP7 classic symbols | Bool |
| 13 | B_Bay3 | STEP7 classic symbols | Bool |

Copy & Paste from and to Excel

The Windows Copy & Paste function as well as the Import / Export function can be used to easily copy individual or several tags from a PLC tag table or a data block to Excel to further process it/them there and then to copy it/them back from Excel to the PLC tag table or the data block.

5.3.6. Retentiveness of PLC Tags



S7-1500:

- Retentiveness (retain) can be set for memory bytes, timers and counters

S7-1200:

- Retentiveness (retain) can only be set for memory bytes

Example:
MB0 ... MB16 are retentive

Retain memory dialog:

- Number of memory bytes starting at MB0: 17
- Number of SIMATIC timers starting at T0: 0
- Number of SIMATIC counters starting at C0: 8
- Currently available retain memory (bytes): 90679

Retentive (Retain) Memory

The S7-1500 CPUs have a retentive memory for storing retentive data when the power is switched OFF. The size of the retentive memory is documented in the technical data of the CPU.

The utilization of the retentive memory of the configured CPU is shown offline in the Project tree under "Program info > Resources of..." or online in the Project tree under "Online & diagnostics > Diagnostics > Memory".

When you define data as retentive, their contents are retained after a power failure, during CPU start-up and during loading of a modified program.

You can define the following data or objects as retentive:

- Memory bytes, timers, counters
- Tags of global data blocks
- Tags of instance data blocks of a function block

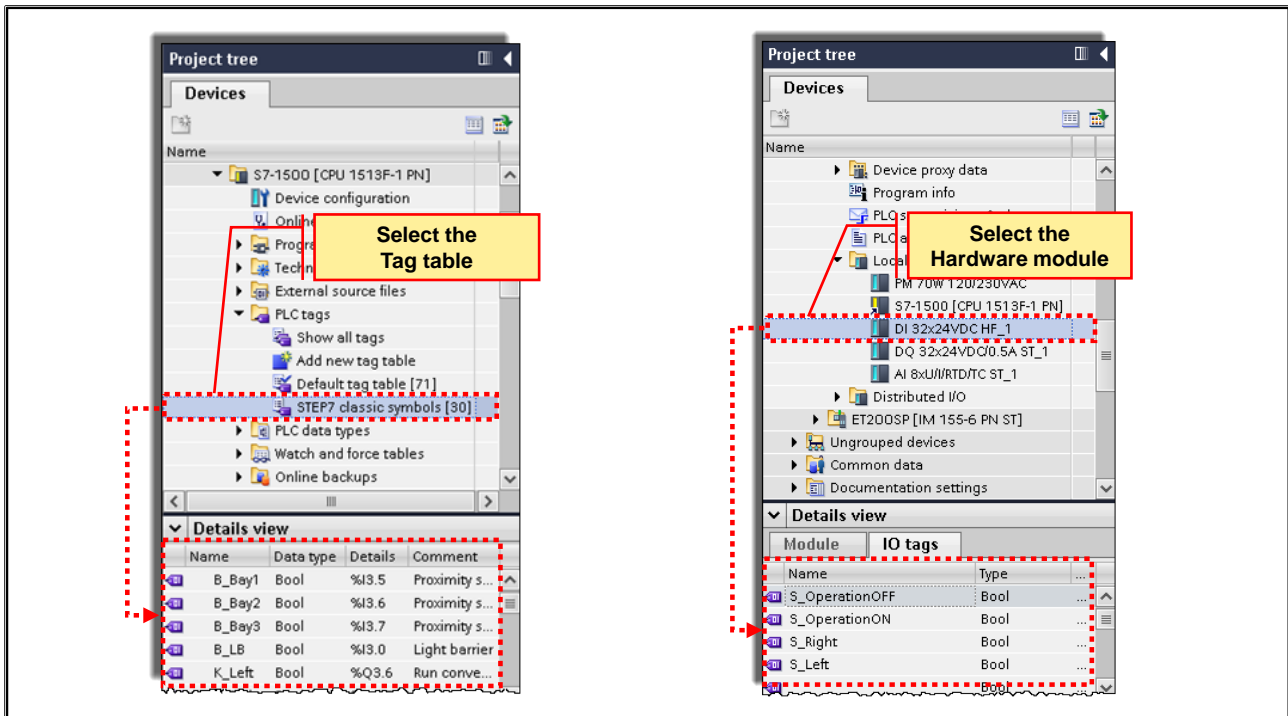
Certain tags of technology objects are always retentive, for example, adjustment values of absolute value encoders.

Memory Bytes, Timers, Counters

For the S7-1500, the number of retentive memory bytes, timers and counters can be defined in the PLC tag table via the "Retain" button.

For the S7-1200, only the number of retentive memory bytes can be defined in the PLC tag table via the "Retain" button.

5.4. Details View of PLC Tags



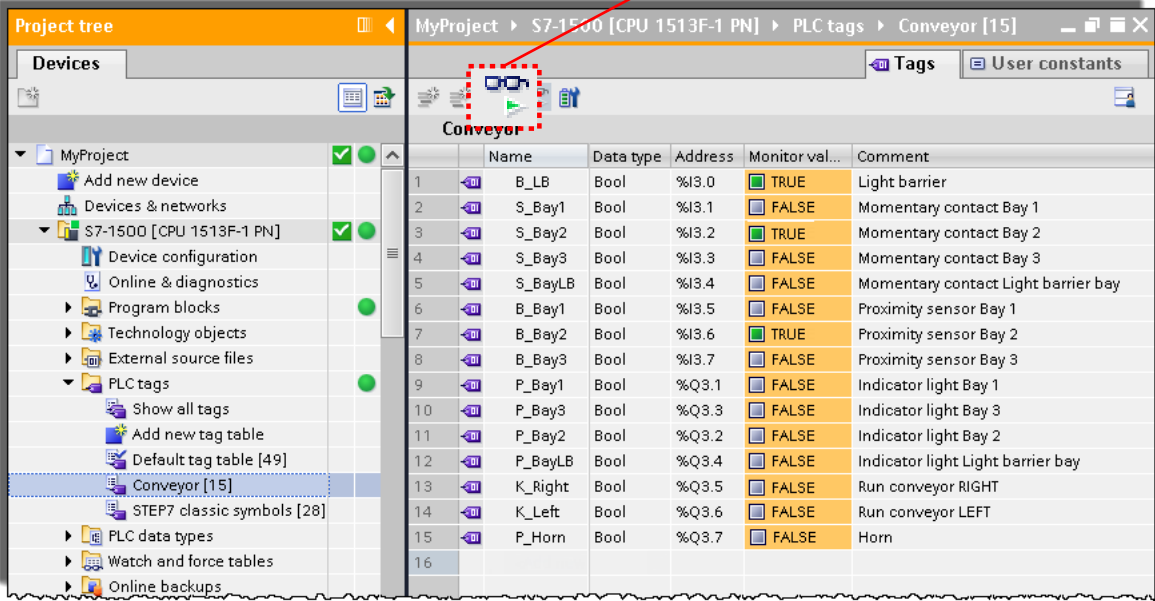
The Details view shows the tags of the object selected (highlighted) in the Project tree:

For example:

- ...tags of the selected tag table
- ...channels of the selected local modules and their tags

That way it is easy for the user, using drag & drop, to integrate tags in the user program, for example.

5.5. Monitoring PLC Tags



Monitor all On / Off

| | Name | Data type | Address | Monitor val... | Comment |
|----|---------|-----------|---------|----------------|-------------------------------------|
| 1 | B_LB | Bool | %I3.0 | TRUE | Light barrier |
| 2 | S_Bay1 | Bool | %I3.1 | FALSE | Momentary contact Bay 1 |
| 3 | S_Bay2 | Bool | %I3.2 | TRUE | Momentary contact Bay 2 |
| 4 | S_Bay3 | Bool | %I3.3 | FALSE | Momentary contact Bay 3 |
| 5 | S_BayLB | Bool | %I3.4 | FALSE | Momentary contact Light barrier bay |
| 6 | B_Bay1 | Bool | %I3.5 | FALSE | Proximity sensor Bay 1 |
| 7 | B_Bay2 | Bool | %I3.6 | TRUE | Proximity sensor Bay 2 |
| 8 | B_Bay3 | Bool | %I3.7 | FALSE | Proximity sensor Bay 3 |
| 9 | P_Bay1 | Bool | %Q3.1 | FALSE | Indicator light Bay 1 |
| 10 | P_Bay3 | Bool | %Q3.3 | FALSE | Indicator light Bay 3 |
| 11 | P_Bay2 | Bool | %Q3.2 | FALSE | Indicator light Bay 2 |
| 12 | P_BayLB | Bool | %Q3.4 | FALSE | Indicator light Light barrier bay |
| 13 | K_Right | Bool | %Q3.5 | FALSE | Run conveyor RIGHT |
| 14 | K_Left | Bool | %Q3.6 | FALSE | Run conveyor LEFT |
| 15 | P_Horn | Bool | %Q3.7 | FALSE | Horn |
| 16 | | | | | |

Monitoring PLC Tags

PLC tags can be monitored directly through the PLC tag table. In so doing, the "Monitor value" shows the current value of the tags in the CPU.

5.5.1. Modifying PLC Tags by means of the Watch Table

Modify values once

Monitor all On / Off

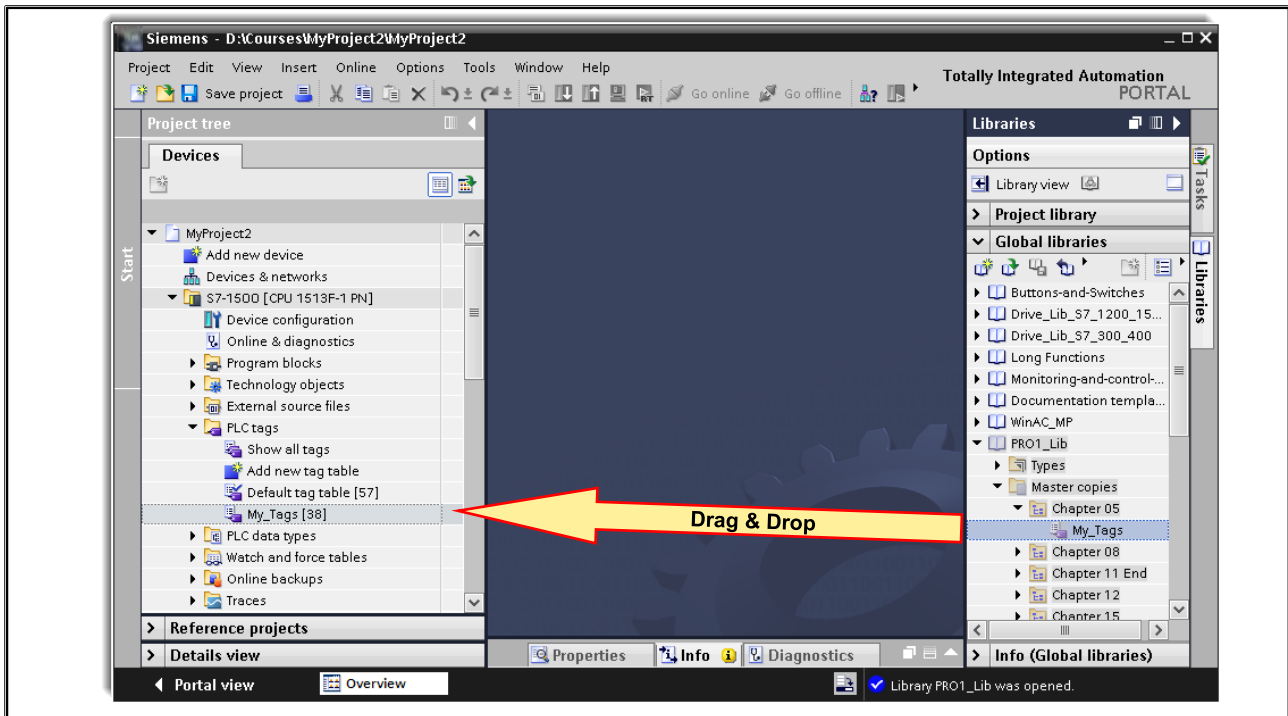
| | Name | Address | Display format | Monitor value | Modify value |
|----|-----------|---------|----------------|--|--------------|
| 1 | "B_Bay1" | %I3.5 | Bool | <input type="checkbox"/> FALSE | |
| 2 | "B_Bay2" | %I3.6 | Bool | <input type="checkbox"/> FALSE | |
| 3 | "B_Bay3" | %I3.7 | Bool | <input type="checkbox"/> FALSE | |
| 4 | "B_LB" | %I3.0 | Bool | <input checked="" type="checkbox"/> TRUE | |
| 5 | "K_Left" | %Q3.6 | Bool | <input type="checkbox"/> FALSE | |
| 6 | "K_Right" | %Q3.5 | Bool | <input checked="" type="checkbox"/> TRUE | TRUE |
| 7 | "P_Bay1" | %Q3.1 | Bool | <input type="checkbox"/> FALSE | |
| 8 | "P_Bay2" | %Q3.2 | Bool | <input type="checkbox"/> FALSE | |
| 9 | "P_Bay3" | %Q3.3 | Bool | <input type="checkbox"/> FALSE | |
| 10 | "P_BayLB" | %Q3.4 | Bool | <input type="checkbox"/> FALSE | |
| 11 | "P_Horn" | %Q3.7 | Bool | <input checked="" type="checkbox"/> TRUE | TRUE |
| 12 | "S_Bay1" | %I3.1 | Bool | <input type="checkbox"/> FALSE | |
| 13 | "S_Bay2" | %I3.2 | Bool | <input type="checkbox"/> FALSE | |
| 14 | "S_Bay3" | %I3.3 | Bool | <input type="checkbox"/> FALSE | |
| 15 | "S_BayLB" | %I3.4 | Bool | <input type="checkbox"/> FALSE | |
| 16 | <Add new | | | | |

Activate/Deactivate values to be modified

In order to be able to modify tags (variables) you require a watch table which you can create in the 'Watch and force tables' folder.

Any tags (values) can be monitored and modified in a Watch table. To modify a tag, a modify value is specified, the tag to be modified is activated (is automatically activated during creation) and by means of the button "Modify all selected values once and now" the values of the activated tag are loaded into the controller.

5.6. Exercise 1: Copying the PLC Tag Table from the Library



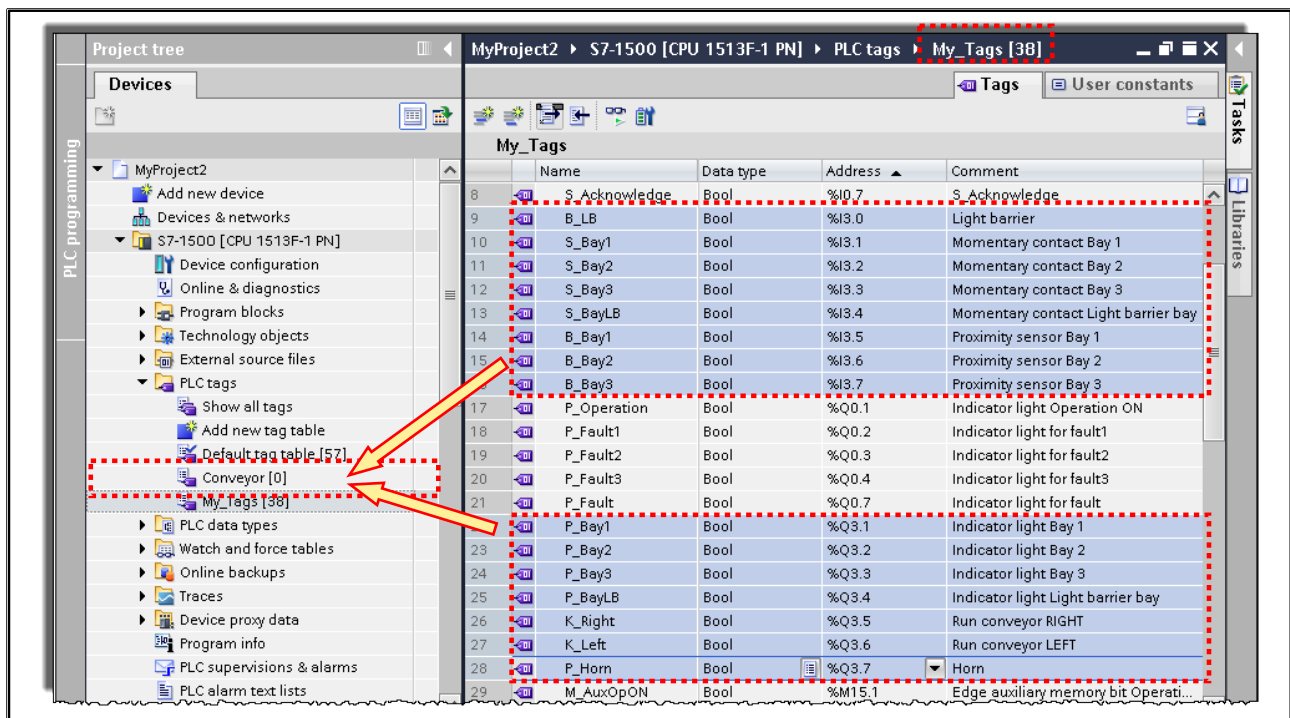
Task

You are to copy the prepared PLC tag table "My_Tags" from the "PRO1_Lib" library into your own project.

What to Do

1. In the Task Card "Libraries > Global libraries", open the library "PRO1_Lib" which is located in the folder C:\02_Archives\TIA_Portal\TIA-PRO1 of your programming device.
2. Using drag & drop, copy the PLC tag table "My_Tags" from the Libraries' folder Master copies\Chapter 05 into your project's folder PLC tags.
3. Save your project.

5.6.1. Exercise 2: Creating the "Conveyor" Tag Table



Task

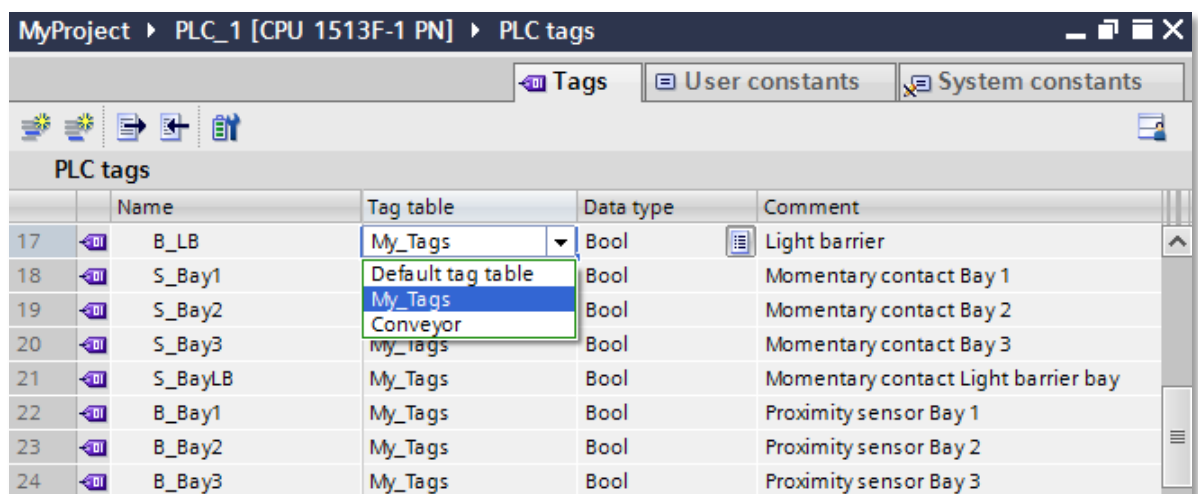
You are to create a separate PLC tag table "Conveyor" for the inputs and outputs to which the sensors and actuators of the conveyor model are connected.

What to Do

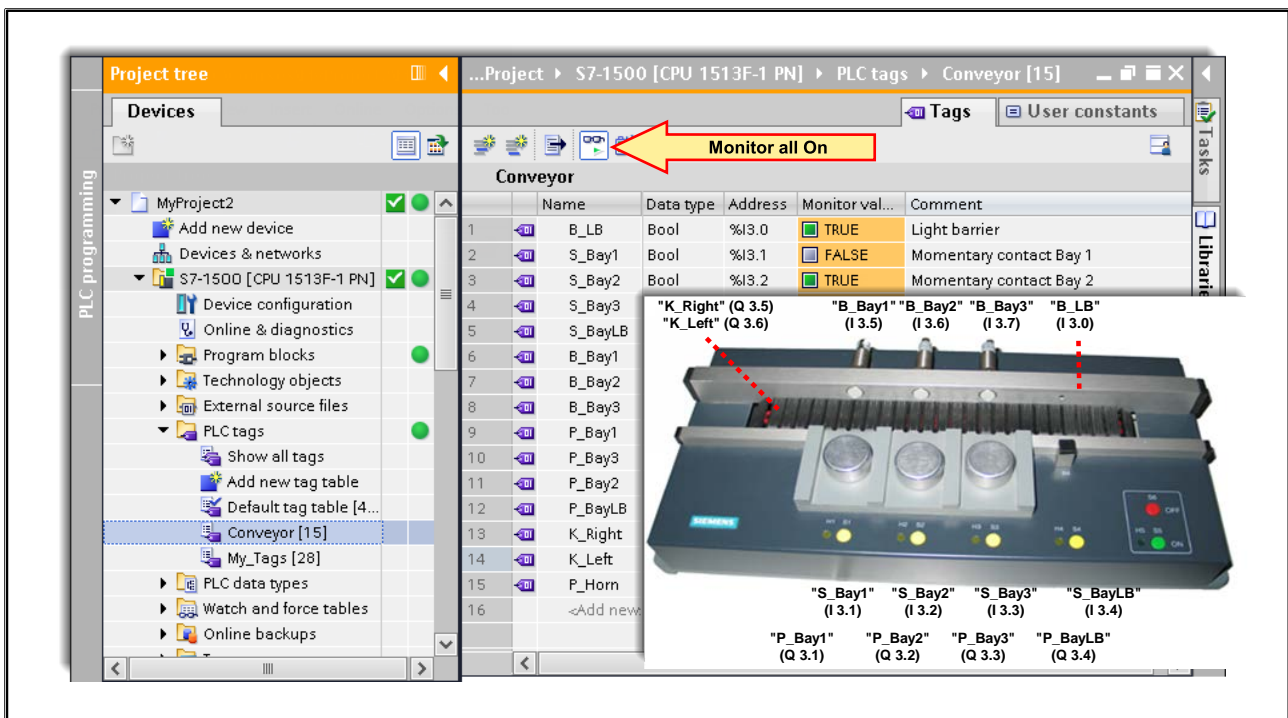
1. Using "Add new tag table", create the new PLC tag table "Conveyor".
2. Open the PLC tag table "My_Tags" and move the inputs and outputs into the newly created PLC tag table "Conveyor" using drag & drop (mouse pointer on tag icons, see picture).
3. Save your project.

Note:

As an alternative, for every tag you can define in which tag table it is to be saved. You do this in the tag table "Show all tags" via the column "Tag table".



5.6.2. Exercise 3: Monitoring the "Conveyor" PLC Tag Table



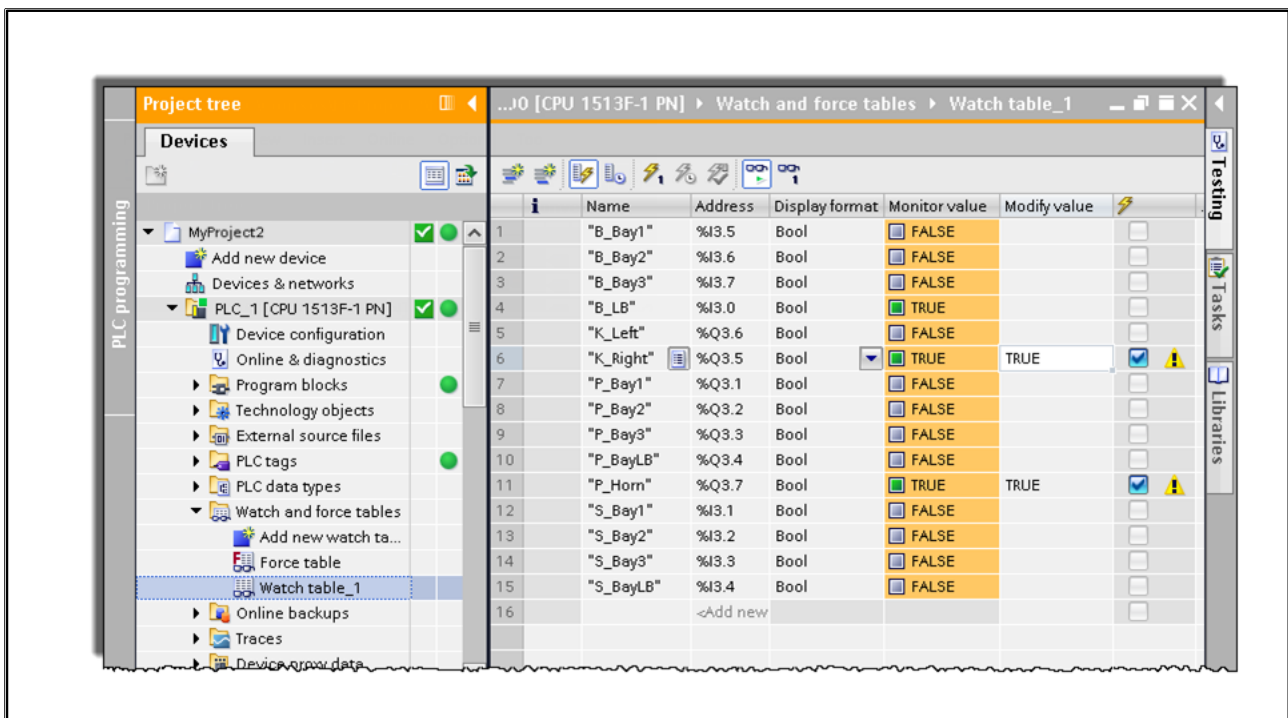
Task

You are to monitor the inputs in the tag table.

What to Do

1. In order to adopt the changes and so that the CPU can switch to RUN, compile and download the software.
2. Check whether the broadband cable of your conveyor model is connected to the "S7-1500 DI/DO" socket on the back of your training case.
3. Switch your conveyor model on.
4. Open the newly created PLC tag table "Conveyor" and activate the function "Monitoring".
5. On the conveyor model, press the Bay pushbuttons and check whether Status '1' or "TRUE" is displayed at the corresponding inputs.
6. Save your project.

5.6.3. Exercise 4: Modifying using the Watch Table



Task

You are to modify the outputs with the help of a watch table.

What to Do

1. In the Watch and force tables folder, add a new watch table.
2. Open the newly created watch table.
3. In the Project tree, select the PLC tag table "Conveyor".
4. From the Detail view, copy the PLC tags of the "Conveyor" PLC tag table into your watch table using drag & drop.
5. For some of the outputs, specify the Modify value TRUE, that is, 1.
6. Check whether the relevant tags are activated for modifying.
7. Activate the function "Monitor all" and modify the outputs.
8. Check whether the relevant outputs are activated.
9. For the outputs, specify the Modify value FALSE, that is, 0 in order to switch them off again.
10. Modify the outputs once again.
11. Save your project.