# Learn-/Training Document

Siemens Automation Cooperates with Education (SCE) | As of Version V9 SP1

**PA Module P01-04**
SIMATIC PCS 7 – Individual Drive Functions

siemens.com/sce

## Matching SCE Trainer Packages for this Learn-/Training Document

- **SIMATIC PCS 7 Software Package V9.0 (set of 3)**
  Order No.: 6ES7650-0XX58-0YS5
- **SIMATIC PCS 7 Software Package V9.0 (set of 6)**
  Order No.: 6ES7650-0XX58-2YS5
- **SIMATIC PCS 7 Software Upgrade Packages (set of 3)**
  Order No.: 6ES7650-0XX58-0YE5 (V8.x→ V9.0)
- **SIMIT Simulation Platform with Dongle V10**
  (contains SIMIT S & CTE, FLOWNET, CONTEC libraries) – 2500 simulation tags
  Order No.: 6DL8913-0AK00-0AS5
- **Upgrade SIMIT Simulation Platform V10**
  (contains SIMIT S & CTE, FLOWNET, CONTEC libraries) from V8.x/V9.x
  Order No.: 6DL8913-0AK00-0AS6
- **Demo Version SIMIT Simulation Platform V10**
  Download
- **SIMATIC PCS 7 AS RTX Box (PROFIBUS) only in combination with ET 200M for RTX –**
  Order No.: 6ES7654-0UE23-0XS1
- **ET 200M for RTX Box (PROFIBUS) only in combination with PCS 7 AS RTX Box –**
  Order No.: 6ES7153-2BA10-4AB1

Note that these trainer packages are replaced with successor packages when necessary.
An overview of the currently available SCE packages is available at: siemens.com/sce/tp

## Continued training

For regional Siemens SCE continued training, get in touch with your regional SCE contact
siemens.com/sce/contact

## Additional information regarding SCE

siemens.com/sce

## Information regarding use

We wish to thank the TU Dresden, particularly Prof. Dr.-Ing. Leon Urbas and the Michael Dziallas Engineering Corporation and all other involved persons for their support during the preparation of this Learn-/Training Document.

# Table of contents

# Control module functions

## 1 Goal

After working through this module, the students will be able to define and classify the term 'Control module function' within the scope of object-oriented software structuring. They will understand the concept, structure and functionality of control module functions and will know typical control module functions and their implementation in ***PCS 7***.

## 2 Prerequisite

This chapter builds on chapter 'Plant hierarchy'. To implement this chapter, you can use an existing project from the previous chapter or the archived project 'p01-03-exercise-r1905-en.zip' provided by SCE. The download of the project(s) is stored on the SCE Internet for the respective module.

The file 'p01-04-symbols-r1905-en.dif' (stored in zip file 'p01-04-files-r1905-en.zip') is additionally needed.

The simulation for the SIMIT program can be retrieved from the file 'p01-04-plantsim-v10-r1905-en.simarc'. It can be run in demo mode.

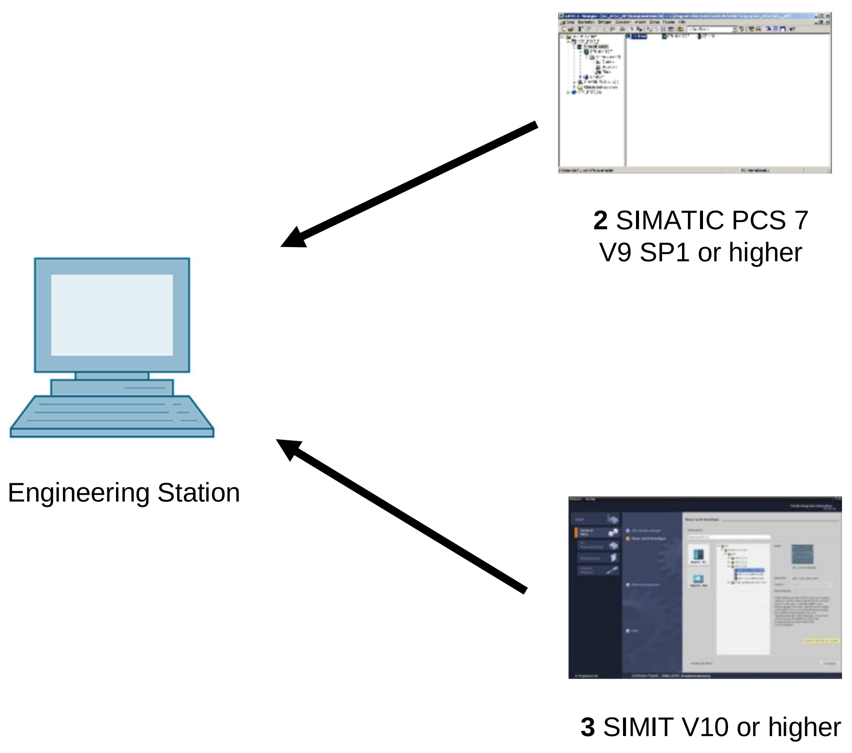
# 3 Required hardware and software

**1** Engineering station: Requirements include hardware and operating system (for further information, see Readme on the PCS 7 installation DVD)

**2** SIMATIC PCS 7 software V9 SP1 or higher

- Installed program packages (contained in SIMATIC PCS 7 Software Trainer Package):
  - *Engineering → PCS 7 Engineering*
  - *Engineering → BATCH Engineering*
  - *Runtime → Single Station → OS Single Station*
  - *Runtime → Single Station → BATCH Single Station*
  - *Options → SIMATIC Logon*
  - *Options → S7-PLCSIM V5.4 SP8*

**3** Demo Version SIMIT Simulation Platform V10


**2** SIMATIC PCS 7 V9 SP1 or higher
**1** Engineering Station
**3** SIMIT V10 or higher

# 4 Theory

## 4.1 Theory in brief

The goal of object-oriented software structuring is to reproduce the structure of the real plant through corresponding modularization of the user software as clearly as possible. To this end, at least one function block is provided for each field device type. This function block in turn provides the entire control logic, the necessary protection and monitoring functions as well as suitable operator control and visualization options. The user program utilizes this block to implement the desired operating behavior of a machine or a process.

Motors and valves are control devices that are not controlled directly, in the sense of object-oriented automation, but rather are initially modeled as function block types. Such function block types are called ***Individual Drive Functions (IDF).*** They enable control, monitoring and operation of the control device by providing corresponding I/Os for actuating and control signals as well as for parameter assignment and monitoring functions. The control is technically implemented by an instance of the function block type and remains hidden from the user.

Figure 1 shows the transition from the real motor – in this case a pump – to a block of the corresponding control module function.

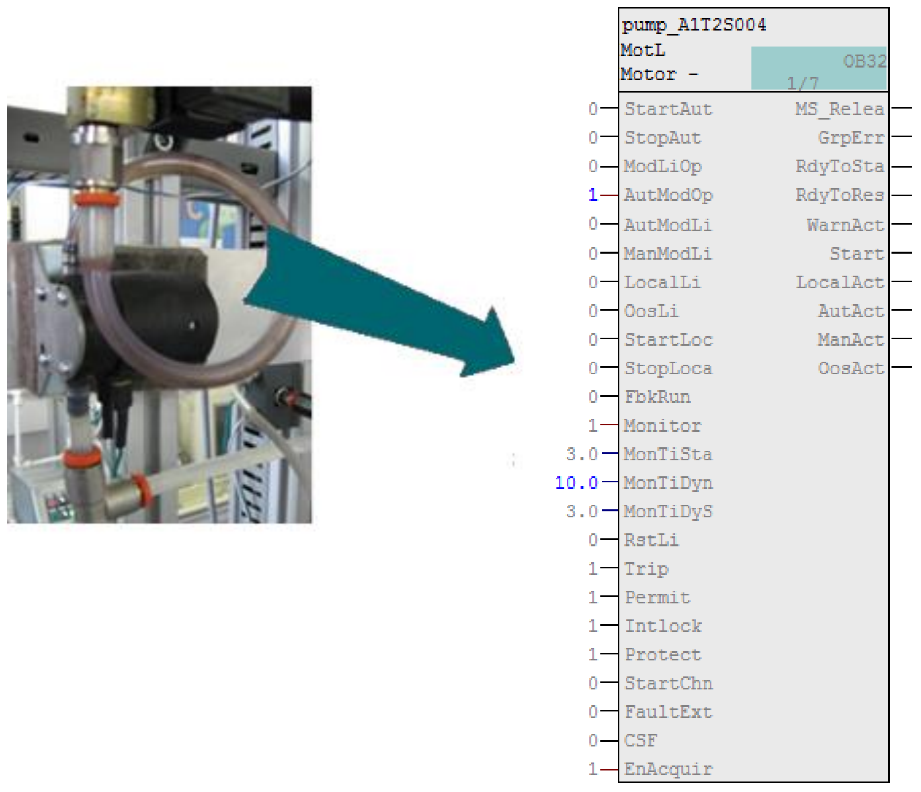


Figure 1: Transition from the real motor to the block of the control module function

In general, control devices can be operated in four different operating modes. The possible operating modes of the device are:

– ***Out of service***

– in ***Manual mode***

– in ***Automatic mode***

– In ***Local mode***

A control module function may only ever adopt one operating mode. The operating modes mentioned can have equal priority or be hierarchically prioritized. In addition, control module functions provide functions for protection against device and process faults. Various interlocks as well as runtime monitoring for the device and for the controlled process are also implemented.

Function block types, referred to as block types in ***PCS 7***, represent pre-assembled program sections for processing recurring functions. They can be inserted in CFCs where they can be parameterized as instances, interconnected and adapted to the project requirements. The block type specifies the characteristics for all instances of this type. ***PCS 7*** already provides many powerful and tested control module functions as block types in the process control libraries. They model one control device each and provide the entire control logic.

In addition, functions are offered for the following:

– ***Operator control*** and ***monitoring*** of the control module function

– ***Controlling*** of signals

– ***Monitoring*** and ***alarming***

– ***Operating state selection***

– ***Interlocks***

***Faceplates*** with different ***views*** allow for seamless integration into a corresponding process control system.

Control module functions enable efficient development of high-performance, high-quality solutions. They modularize and type-define recurring functionalities. These functionalities can be reused and centrally modified, which speeds up the development process considerably.

## 4.2 Object-oriented software structuring

The goal of object-oriented software structuring is to reproduce the structure of the real plant through corresponding modularization of the user software as clearly as possible. To this end, a separate program is created for each field device in the plant. At least one function block is provided for each field device type.

This block implements the entire control logic for this field device type. In addition, it makes necessary protection and monitoring functions available as well as suitable operator control and visualization options. It thus encapsulates the entire functionality that is necessary in connection with the corresponding field device type. The user program utilizes this block to implement a desired control for a machine or process without having to draw on knowledge of the internal data and operations of the function block.

## 4.3 Channel functions (drivers)

In addition to the handling of field devices through dedicated reusable blocks, it is often useful to abstract the I/O interfacing through ***channel blocks*** (***drivers***). Although it is always possible to access the process image directly using symbol names or addresses, the possibly numerous parameters for configuration of the channel must be set elsewhere. This quickly leads to unclear programs. ***PCS 7*** provides a set of channel blocks that evaluate the status signals of the modules and support testing and commissioning of automation programs through simulation modes. In the analog channel blocks according to Table 1, internal digital variables are also mapped to the physical operands and display variables through the VLRANGE and VHRANGE parameters. ***PCS 7*** can generate the necessary drivers automatically when channel blocks are used. Channel blocks are therefore frequently used in the templates of the ***PCS 7*** libraries.

| Channel block | Block | Interconnection, Parameter |
|---|---|---|
| Digital output | PCS7DiOu | PV_Out |
| Digital input | PCS7DiIn | PV_In |
| Analog output | PCS7AnOu | PV_Out, Scale |
| Analog input | PCS7AnIn | PV_In, Scale |

Table 1: Listing of various channel blocks for abstraction of the I/O interfacing

## 4.4 Control module functions

Motors and valves are of crucial importance in factory and process automation as control devices. There are many popular types with specific operating and message behavior. In terms of object-oriented automation, such devices are not controlled directly, but are initially modelled as function block types. Control of these devices always takes place indirectly via an instance of the corresponding function block type. Function blocks for motors and valves are called ***Individual Drive Functions (IDF).*** Control module functions allow control, monitoring and operation of control devices by providing corresponding I/Os for actuating and control signals as well as for parameter assignment and monitoring functions. The technical implementation of the control, such as the startup characteristics, the control of the drive or the device monitoring, is realized through the function block instance and remains hidden from the user. ***PCS 7*** already provides many powerful and tested control module functions as block types in the process control libraries. Table 2 summarizes the control module functions of the ***PCS 7 Advanced Process Library*** [2].

| Control module function | Application | Object name |
|---|---|---|
| MotL | Control of motors with a control signal (on/off) and running feedback | FB 1850 |
| MotRevL | Control of motors with two directions of rotation (counterclockwise, clockwise) and a maximum of two feedbacks. | FB 1851 |
| MotSpedL | Control of motors with two speeds (slow/fast) and a maximum of two feedbacks. | FB 1856 |
| VlvL | Control of control valves with a control signal (open/close) and two position feedback signals (open/closed). | FB 1899 |
| VlvMotL | Control of motor-driven valves with two control signals and position feedback signals (open/closed). | FB 1900 |

Table 2: Control module functions of the ***PCS 7 Advanced Process Library***

## 4.5 Protective measures

Various protective measures must be taken for the control of control devices. First, the devices themselves must be protected from faults. Second, under fault conditions the controlled process must be brought to a safe state and kept there until the fault conditions are eliminated.

***Device faults*** (for example, cable break, axis break) cannot be prevented by the control device. But the effects can be minimized through redundancy concepts. ***Process faults*** (for example, tank overflow, dry run of a pump), on the other hand, must be prevented directly through the control. Appropriate ***interlocks*** are implemented for this. If the control module function detects a dangerous process state based on the current input values, the controlled device is brought to a safe state (refer to the chapter 'Functional Safety'). The device is kept in this state as long as the dangerous process state is present. Interlocks are normally specified using an ***interlock matrix***.

To detect a device fault, a control module function often performs ***runtime monitoring*** By using certain sensor information from end position sensors in valves, for example, the control module function checks whether the output actuating signals also achieve the required effect. If over a certain period of time the measured values contradict the output actuating signals, there is an error. If such a runtime error is detected, the higher-level control system is alerted and the controlled device is deactivated. The device remains inactive until the runtime error is remedied and the alarm has been acknowledged. Simple binary circuit breakers are often used to detect device faults.

## 4.6 Operating modes

Control devices are generally not operated exclusively automatically. From time to time it is necessary to operate the control manually from the control room, or to control the device directly on site; for example, when repairs have to be made. For this reason, we distinguish between four basic operating modes

– ***Out of service***: The device is not active.

– ***Automatic mode***: The control module function is controlled automatically by a higher-level program.

– ***Manual mode***: The operator controls the control module function directly by means of a graphical user interface of the control system.

– ***Local mode***: The operator operates the device locally, for example, using an operator panel.

A control module function may only be in exactly one operating mode at a given time. Various concepts exist for how the related mode changeover can be safely and unambiguously realized. Basically, these concepts can be divided into those in which the operating modes are on a par and those in which an operating mode hierarchy exists. In the latter case, the possible operating modes are additionally explicitly prioritized. In this case, a selected operating mode is changed precisely when the device is not active (operating mode: ***Out of service***) or when the desired new operating mode has a higher priority that the previously selected one.

## 4.7 Function block types in PCS 7

Function block types are referred to as block types in ***PCS 7*** and represent pre-assembled program sections for processing recurring functions. They can be inserted in CFCs where they can be parameterized as instances, interconnected and adapted to the project requirements.

The block type specifies the characteristics for all instances of this type. The utilized block types of a project are stored in the master data library for this. If the block type stored there is changed, the changes are applied immediately to all instances. This concept of type definition supports efficient engineering through reusability and central changeability of frequently recurring functions.

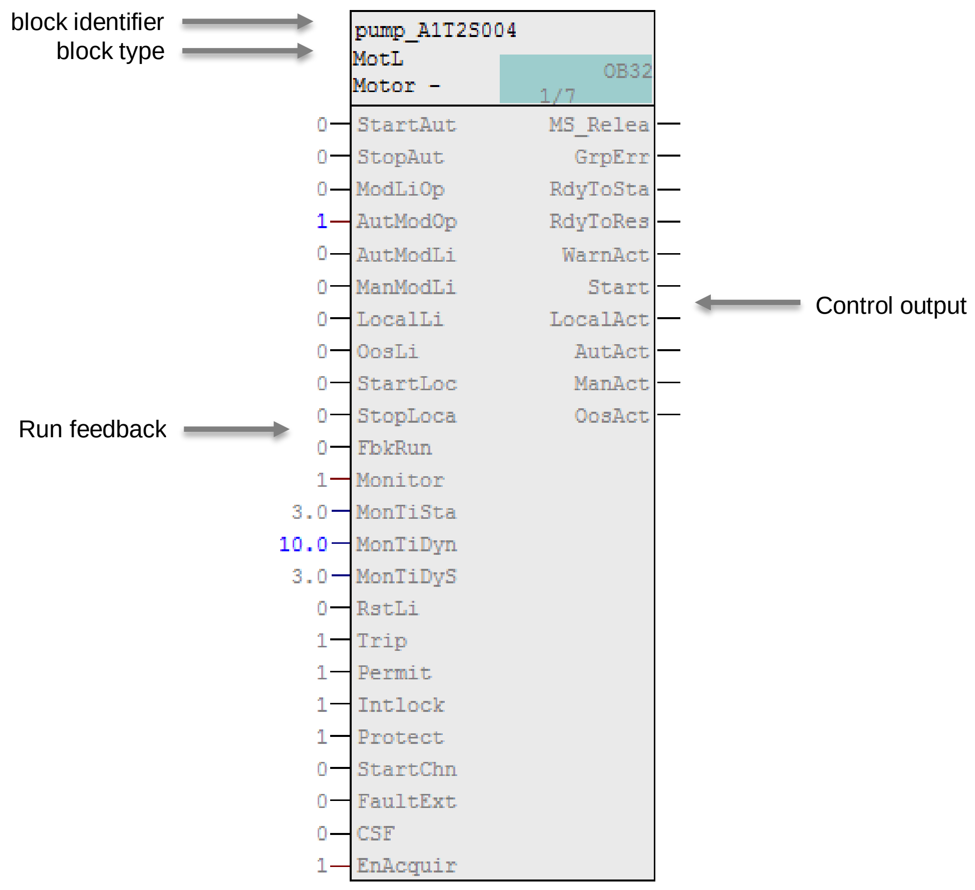


Figure 2: Block of control module function Motor_Lean

A control module function in ***PCS 7*** models a control device and provides the entire control logic. Figure 2 describes the basic structure of the corresponding motor block using the control module function ***Motor_Lean*** as an example.

The block also provides the following functions:

– ***Operator control, monitoring, messaging***

  Operator control and monitoring of process values and setpoints and control of operator authorizations and maintenance releases are possible via a display and operator input area. General and instance-specific messages provide information about the status of the device and the process.

– ***Control of signals***

  Control signals can be output as static or pulsed signals. The signal status, i.e. quality of the actuating signals, is monitored Internal and external setpoints as well as simulated values can be specified, and ramps or dead zones can also be set.

– ***Monitoring***

  The block can monitor limits and generate corresponding warnings or alarms if limits are violated. In addition, feedback from actuating signals can be monitored.

– ***Interlock functions***

  The block allows for a simple activation enable as well as an interlock with and without reset. It implements a motor protection function that can switch off the motor in the event of thermal overload. In addition, rapid stop is available for motors; it has the highest priority in all operating modes and states. In the case of an interlock, the device is automatically brought to a de-energized state and thus to a defined safety position.

– ***Operating state selection***

  The operating modes Local mode, Automatic mode, Manual mode and Out of service presented at the beginning are available for all control module functions in ***PCS 7***. The are prioritized in descending order, i.e. local mode has highest priority and out of service mode has lowest priority with automatic and manual mode having the same priority. In addition, it is possible to put the block into a different operating state using interconnectable input parameters, regardless of the currently pending control (***forcing*** of operating states).

– ***Faceplates with different views***

  Faceplates provide a corresponding block icon and, depending on the application, corresponding views for each block type. Typical faceplates are, for example, the block icon itself, the parameter view of motors and valves and the limit view of motors.

This list of functions clearly shows the complexity and the range of functions of a typical control module function. The number of available inputs and outputs for these block types is corresponding large. For example, the control module function ***Motor_Lean*** has a total of 53 I/Os. To keep the complexity of the program design low nevertheless, it is possible to hide inputs or outputs that are not needed. The control module functions in ***PCS 7*** also use a uniform and integrated scheme for designating the inputs and outputs.

The control module functions in ***PCS 7*** provide a large range of functions and guarantee consistently high quality and reliability of the algorithms used. All block types have been tested extensively and have already been proven in industrial use. This reduces the effort needed to develop high-performance, high quality solutions considerably.

## 4.8 References

[1] Seitz, M. (2008): Speicherprogrammierbare Steuerungen. Hanser Fachbuchverlag

[2] SIEMENS (2018): SIMATIC Process Control System PCS 7 Advanced Process Library (V9.0 SP2). A5E39148920-AC.

support.automation.siemens.com/WW/view/en/109760968)

# 5 Task

A pump motor for draining the liquid from Reactor R001 is to be created as the first program in the Continuous Function Chart (CFC). The pump motor has an output for controlling the pump and a feedback for checking whether the pump is also running.

| Symbol | Address | Data type | Symbol comment |
|---|---|---|---|
| A1.T2.A1T2S003.SO+.O+ | I 1.3 | BOOL | Pump outlet reactor R001 feedback running |
| A1.T2.A1T2S003.SV.C | Q 3.4 | BOOL | Pump outlet reactor R001 control signal |

Table 3: Assignment list

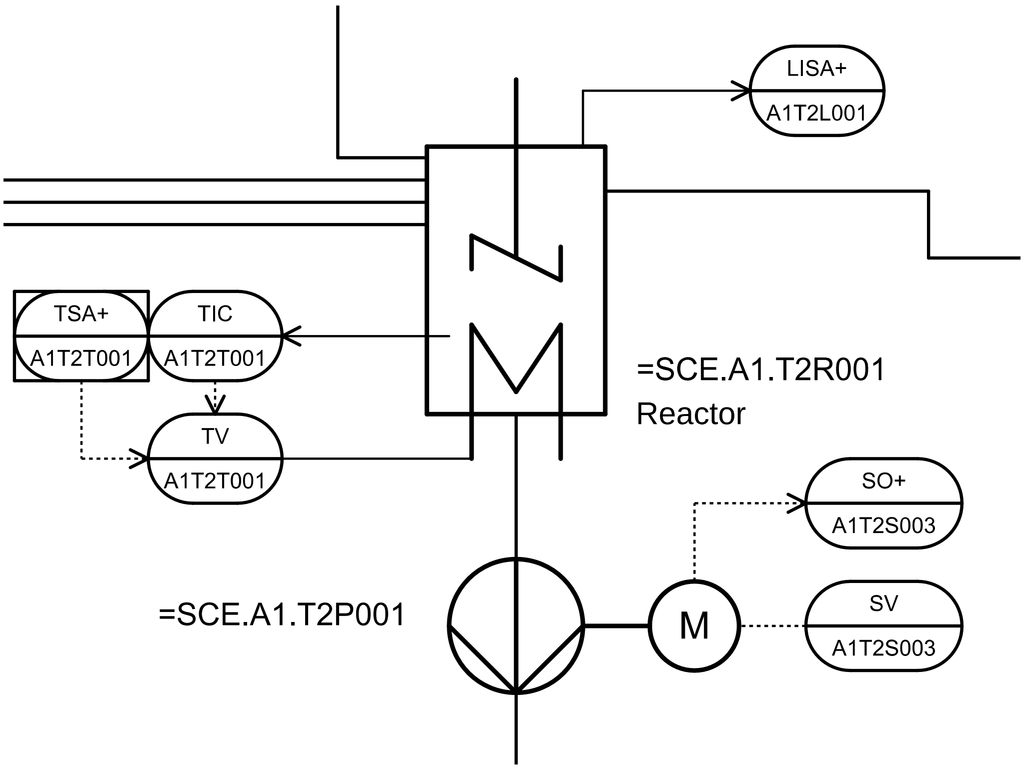


Figure 3: Excerpt from P&ID flowchart

# 6 Planning

Besides the two symbols known from Table 3 of the task, there are many other symbols in the plant. To avoid having to enter all of these by hand, the previously prepared file p01-04-symbols-r1905-en.dif can be imported. It can be generated from the P&ID as the result of an effective plant planning. Because a motor (see Figure 4) is to be realized, a pre-assembled chart 'Motor_Lean" from a ***PCS 7*** library will be used for creating the program. It will be copied to the project's master data library and adapted there. Then the program will be loaded to the PLC simulation and tested.

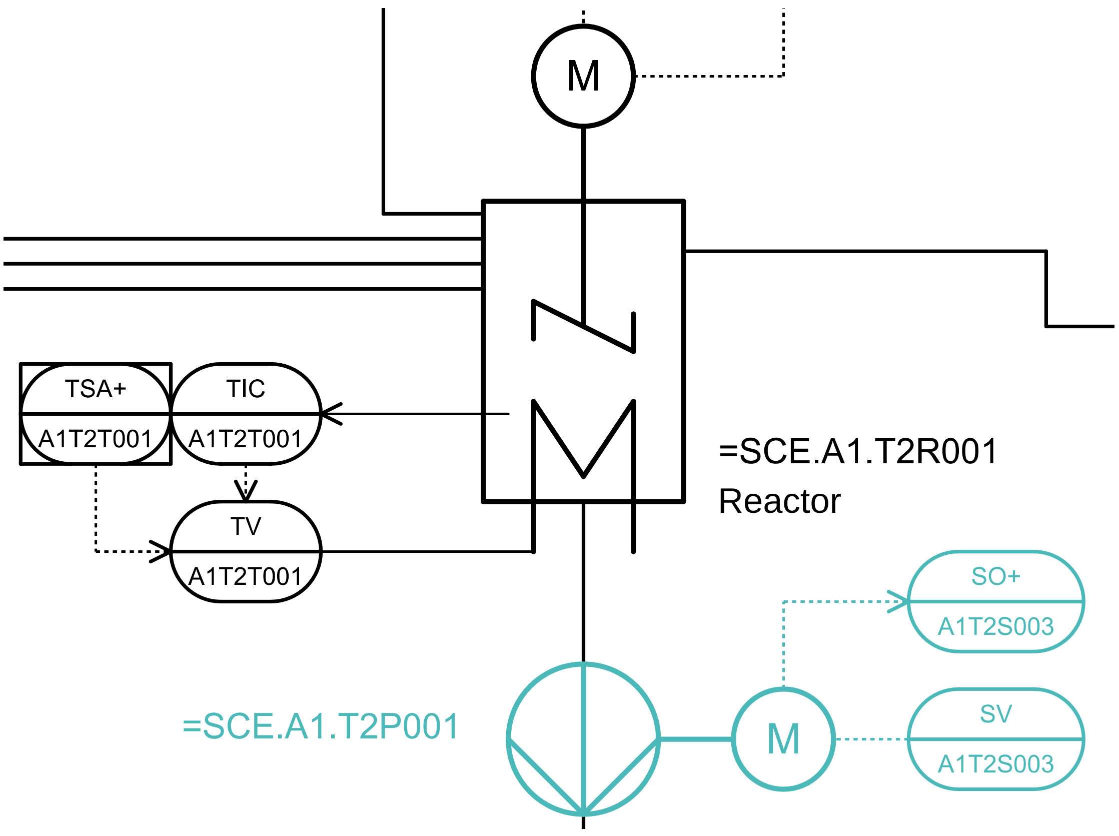


Figure 4: Portion of the P&ID flow chart to be programmed

# 7 Learning objective

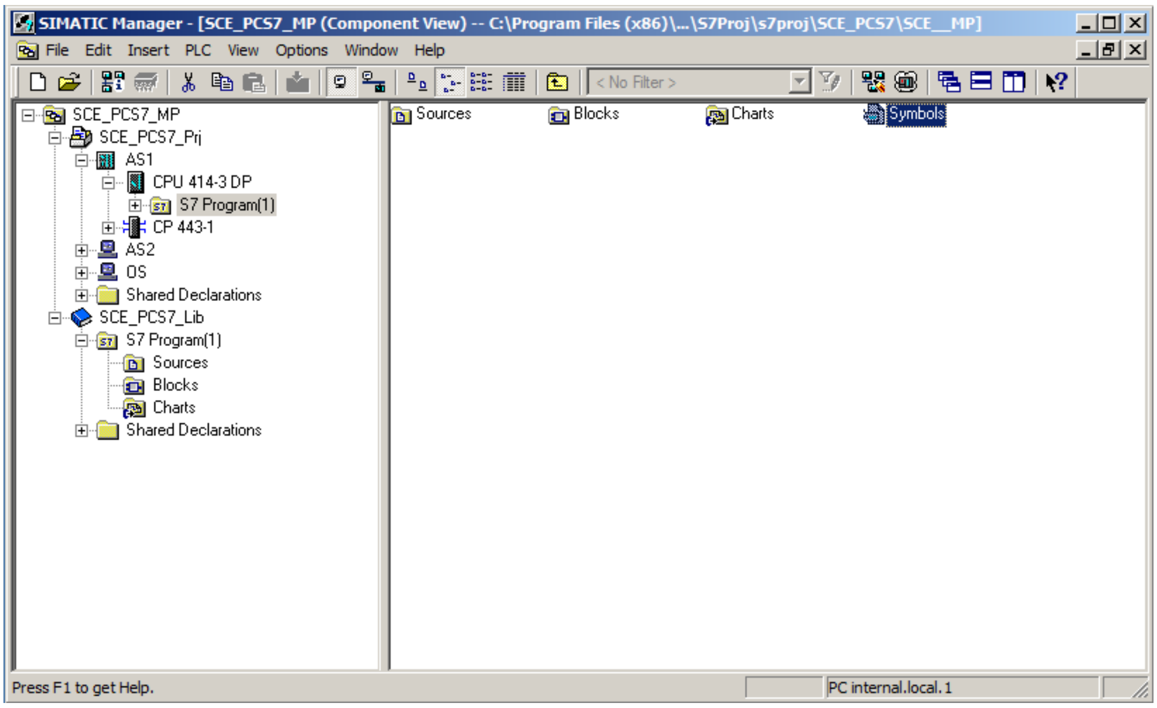
In this chapter, students learn the following:

- Creation and import of symbols via a symbol table
- Use of master data libraries
- Creation and editing of CFCs
- Central compilation and download of the project
- Testing the program using the control functions in the CFC

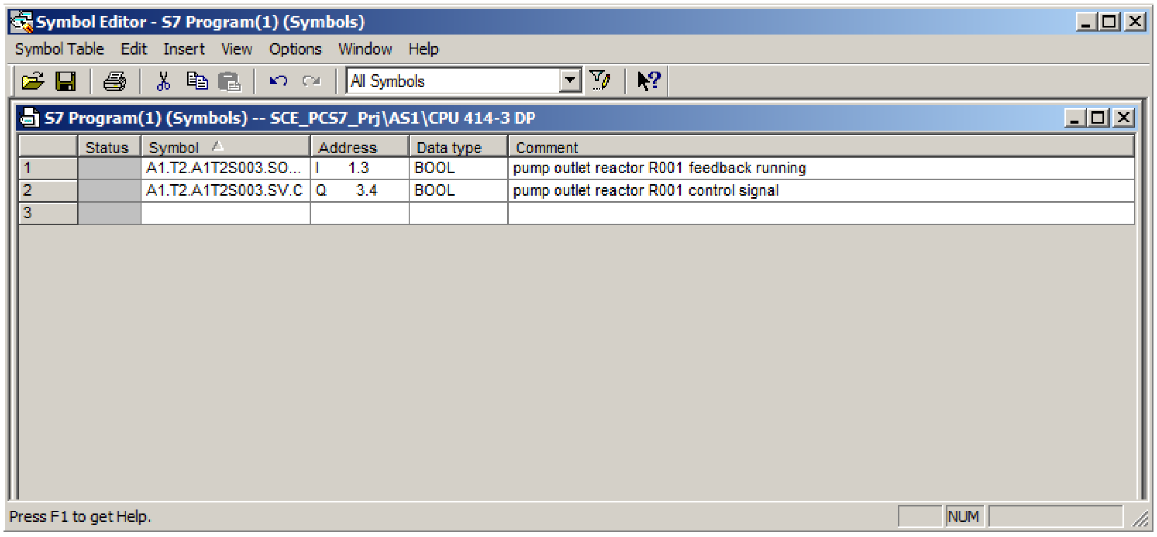
# 8 Structured step-by-step instructions

## 8.1 Editing the symbol table

1. Before starting the programming of the control module function for the pump motor, symbols for the global tags are created. For this, you select the component view in SIMATIC Manager, select the folder 'S7 Program(1)' and double-click Symbols to open the symbol table. (→ SIMATIC Manager → View → Component View → AS1 → CPU 414-3 DP → S7 Program(1) → Symbols )

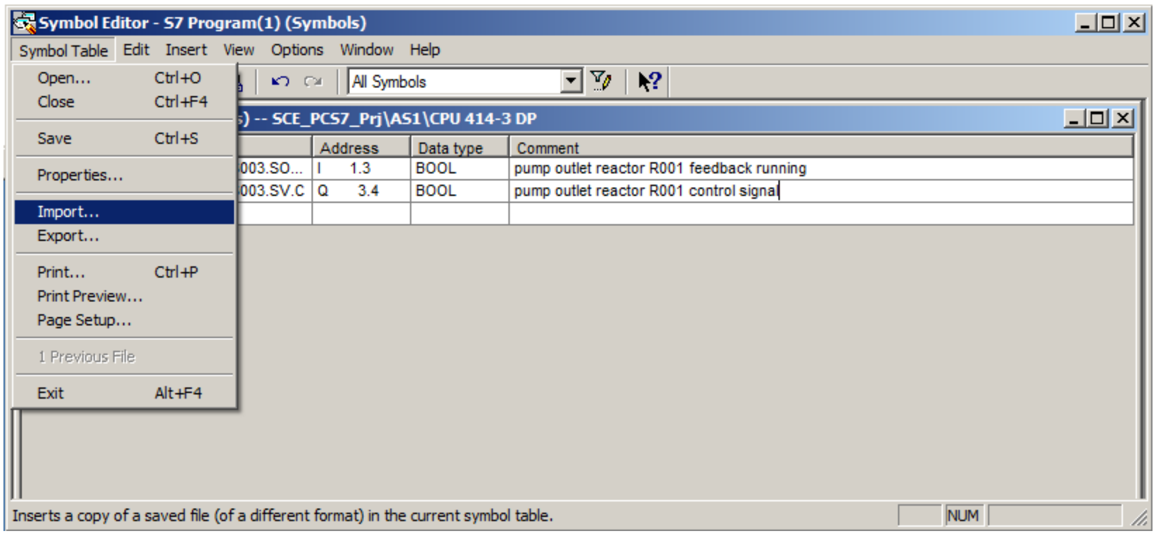


2. The symbol and symbol comment can then be specified for each address in the symbol table.


Symbol Editor - S7 Program(1) (Symbols)
Symbol Table Edit Insert View Options Window Help
All Symbols
S7 Program(1) (Symbols) -- SCE_PCS7_Prj\AS1\CPU 414-3 DP
Press F1 to get Help.
NUM

| | Status | Symbol | Address | Data type | Comment |
|---|---|---|---|---|---|
| 1 | | A1.T2.A1T2S003.SO... | I 1.3 | BOOL | pump outlet reactor R001 feedback running |
| 2 | | A1.T2.A1T2S003.SV.C | Q 3.4 | BOOL | pump outlet reactor R001 control signal |
| 3 | | | | | |

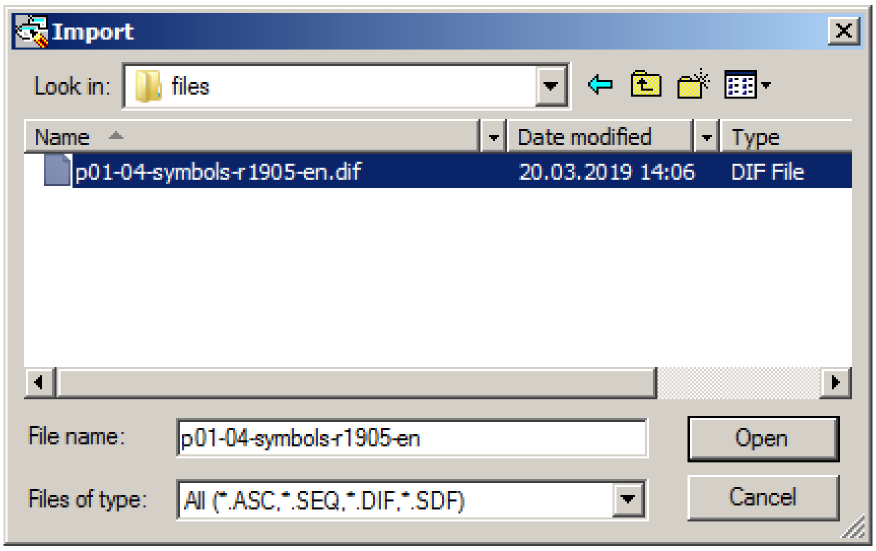
3. If available, the content for the complete symbol table can also imported in *.dif format. In this case, the imported table is integrated into the existing table. (→ Symbol Table → Import)



| | Address | | Data type | Comment |
|---|---|---|---|---|
| 003.SO... | I | 1.3 | BOOL | pump outlet reactor R001 feedback running |
| 003.SV.C | Q | 3.4 | BOOL | pump outlet reactor R001 control signal |

Inserts a copy of a saved file (of a different format) in the current symbol table.

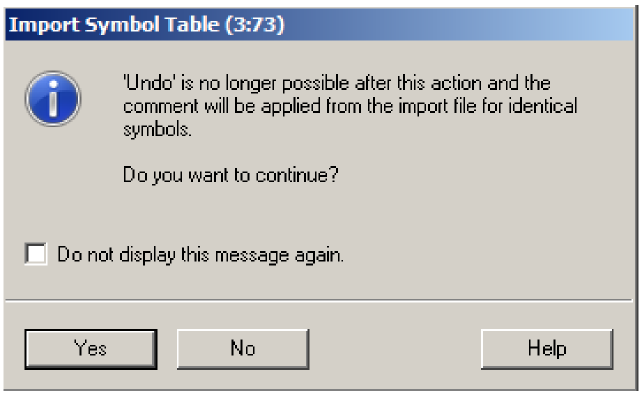
4. The source file in 'Data Interchange Format' (*.DIF) is now selected. (→ p01-04-symbols-r1905-en.dif → Open → Yes → Yes → ☒)



| Name | Date modified | Type |
|---|---|---|
| p01-04-symbols-r1905-en.dif | 20.03.2019 14:06 | DIF File |

File name: p01-04-symbols-r1905-en

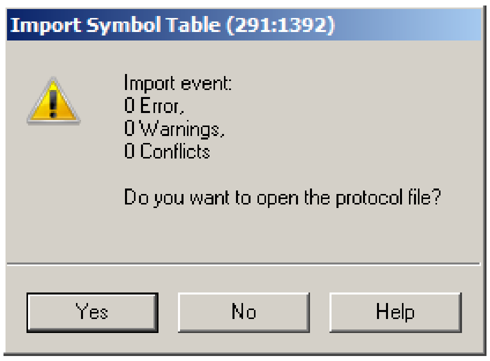
Files of type: All (*.ASC,*.SEQ,*.DIF,*.SDF)



'Undo' is no longer possible after this action and the comment will be applied from the import file for identical symbols.

Do you want to continue?

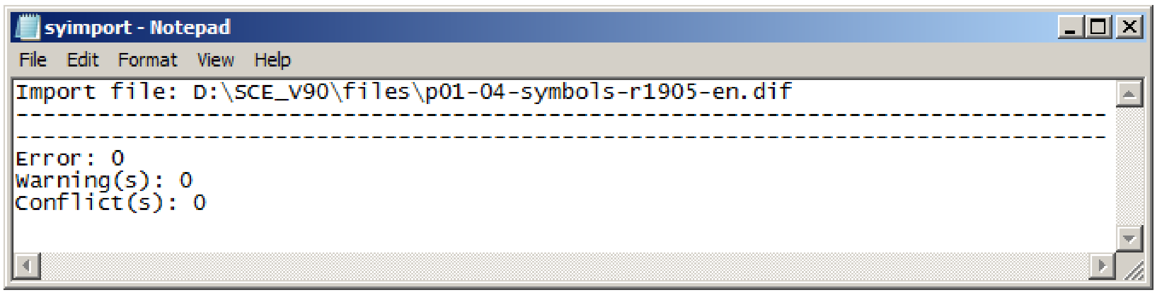
☐ Do not display this message again.



Import event:
0 Error,
0 Warnings,
0 Conflicts

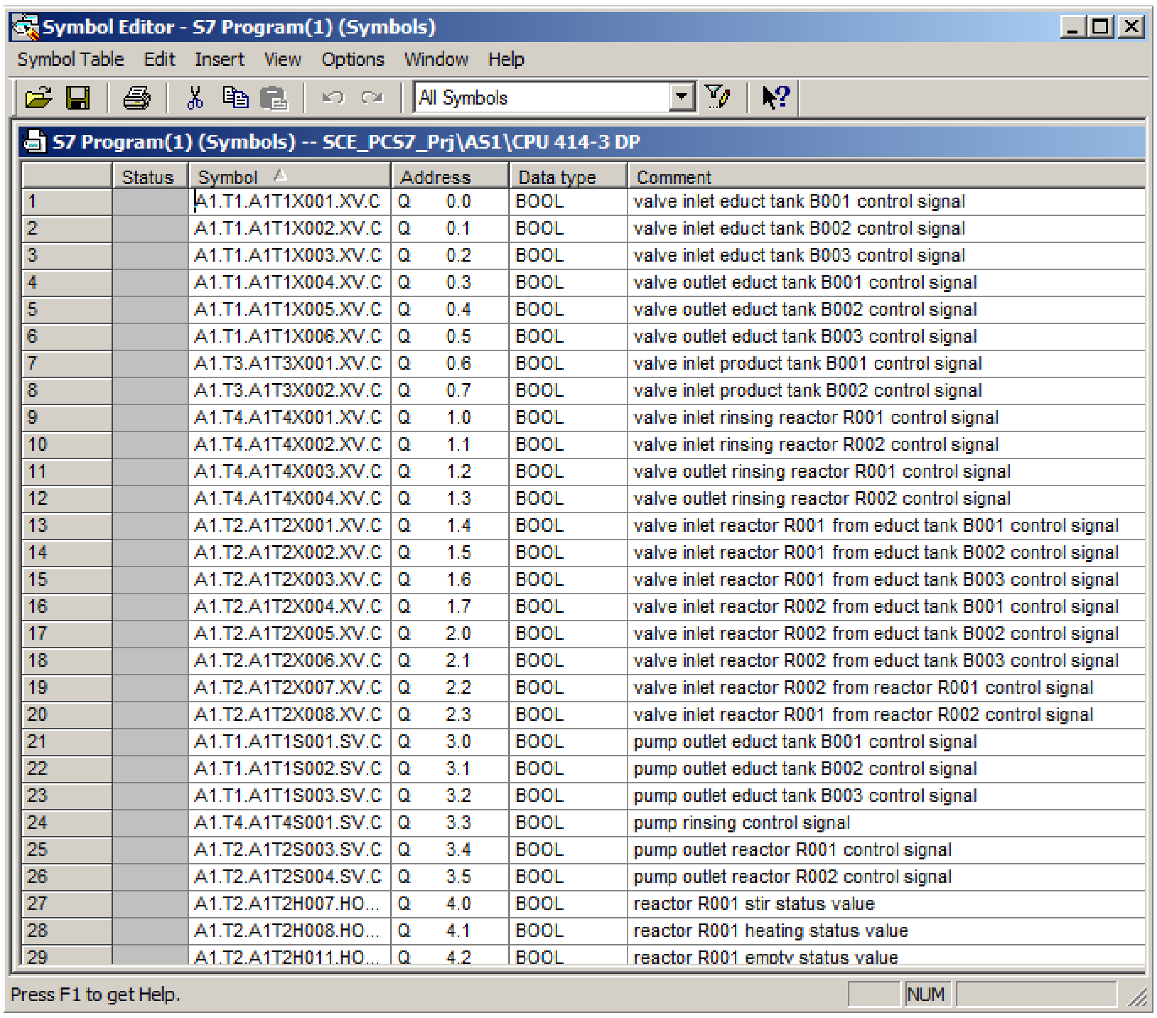
Do you want to open the protocol file?

5. A log is displayed.



```
Import file: D:\SCE_V90\files\p01-04-symbols-r1905-en.dif
--------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------
Error: 0
Warning(s): 0
Conflict(s): 0
```

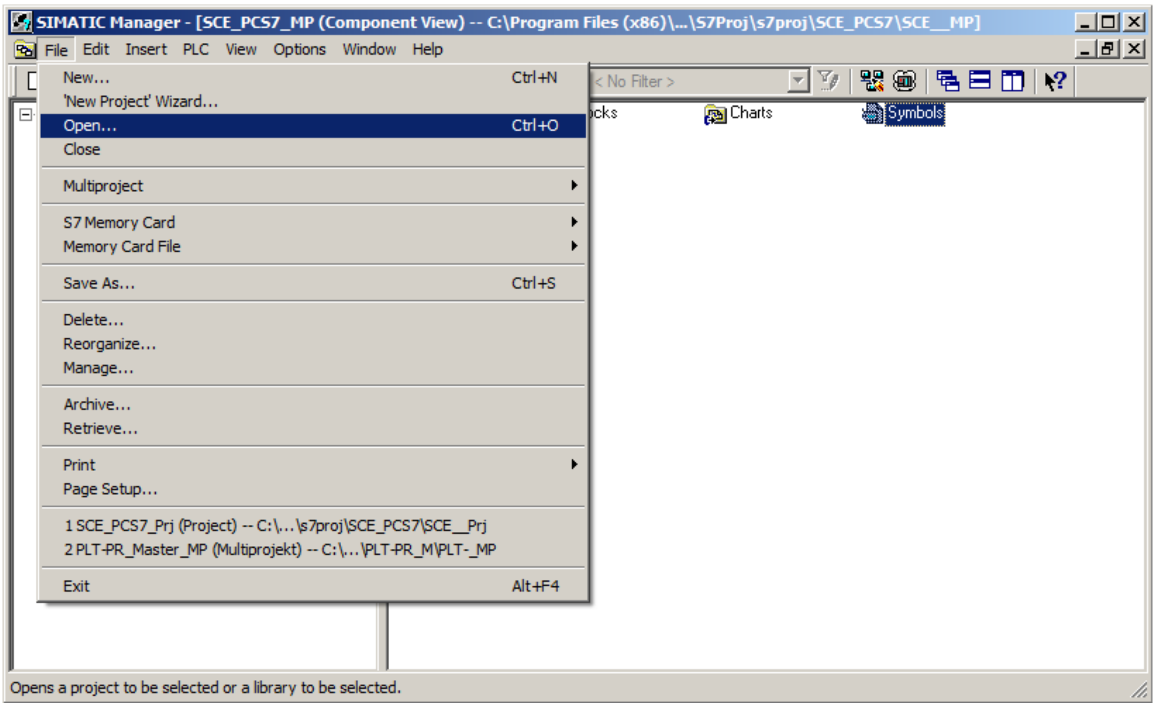
6. The finished symbol table must still be saved before closing. (→ Save → )


Symbol Editor - S7 Program(1) (Symbols)
Symbol Table Edit Insert View Options Window Help
All Symbols
S7 Program(1) (Symbols) -- SCE_PCS7_Prj\AS1\CPU 414-3 DP
Press F1 to get Help.
NUM

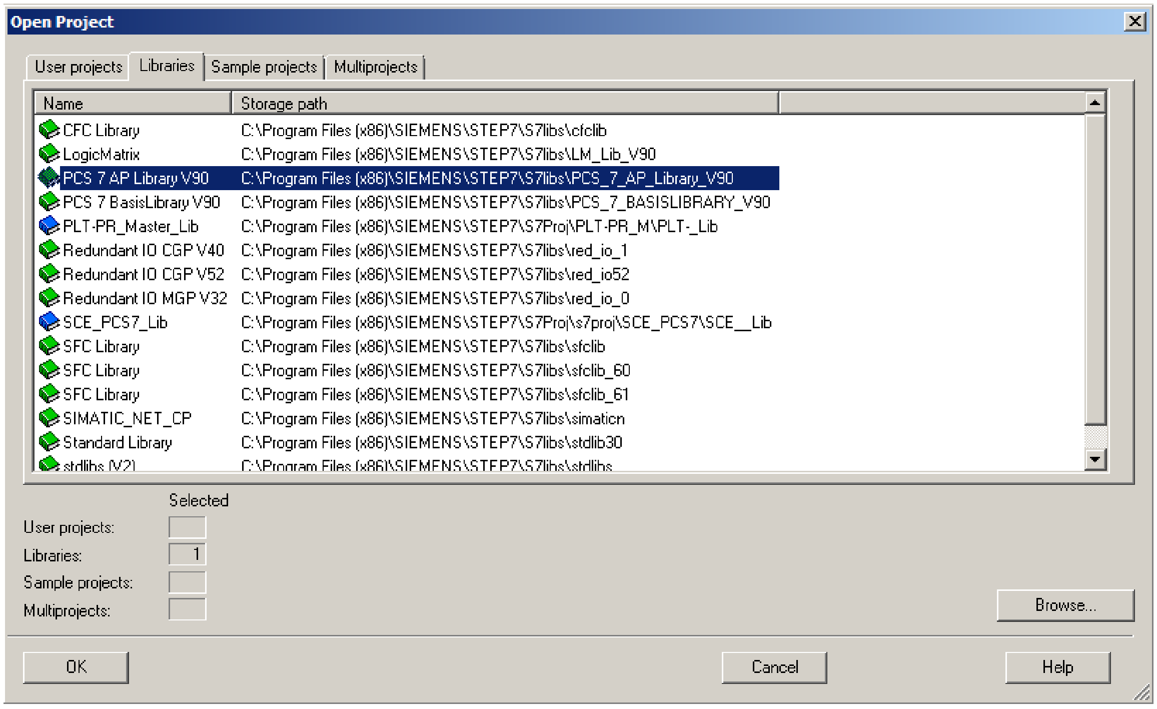
| | Status | Symbol | Address | | Data type | Comment |
|---|---|---|---|---|---|---|
| 1 | | A1.T1.A1T1X001.XV.C | Q | 0.0 | BOOL | valve inlet educt tank B001 control signal |
| 2 | | A1.T1.A1T1X002.XV.C | Q | 0.1 | BOOL | valve inlet educt tank B002 control signal |
| 3 | | A1.T1.A1T1X003.XV.C | Q | 0.2 | BOOL | valve inlet educt tank B003 control signal |
| 4 | | A1.T1.A1T1X004.XV.C | Q | 0.3 | BOOL | valve outlet educt tank B001 control signal |
| 5 | | A1.T1.A1T1X005.XV.C | Q | 0.4 | BOOL | valve outlet educt tank B002 control signal |
| 6 | | A1.T1.A1T1X006.XV.C | Q | 0.5 | BOOL | valve outlet educt tank B003 control signal |
| 7 | | A1.T3.A1T3X001.XV.C | Q | 0.6 | BOOL | valve inlet product tank B001 control signal |
| 8 | | A1.T3.A1T3X002.XV.C | Q | 0.7 | BOOL | valve inlet product tank B002 control signal |
| 9 | | A1.T4.A1T4X001.XV.C | Q | 1.0 | BOOL | valve inlet rinsing reactor R001 control signal |
| 10 | | A1.T4.A1T4X002.XV.C | Q | 1.1 | BOOL | valve inlet rinsing reactor R002 control signal |
| 11 | | A1.T4.A1T4X003.XV.C | Q | 1.2 | BOOL | valve outlet rinsing reactor R001 control signal |
| 12 | | A1.T4.A1T4X004.XV.C | Q | 1.3 | BOOL | valve outlet rinsing reactor R002 control signal |
| 13 | | A1.T2.A1T2X001.XV.C | Q | 1.4 | BOOL | valve inlet reactor R001 from educt tank B001 control signal |
| 14 | | A1.T2.A1T2X002.XV.C | Q | 1.5 | BOOL | valve inlet reactor R001 from educt tank B002 control signal |
| 15 | | A1.T2.A1T2X003.XV.C | Q | 1.6 | BOOL | valve inlet reactor R001 from educt tank B003 control signal |
| 16 | | A1.T2.A1T2X004.XV.C | Q | 1.7 | BOOL | valve inlet reactor R002 from educt tank B001 control signal |
| 17 | | A1.T2.A1T2X005.XV.C | Q | 2.0 | BOOL | valve inlet reactor R002 from educt tank B002 control signal |
| 18 | | A1.T2.A1T2X006.XV.C | Q | 2.1 | BOOL | valve inlet reactor R002 from educt tank B003 control signal |
| 19 | | A1.T2.A1T2X007.XV.C | Q | 2.2 | BOOL | valve inlet reactor R002 from reactor R001 control signal |
| 20 | | A1.T2.A1T2X008.XV.C | Q | 2.3 | BOOL | valve inlet reactor R001 from reactor R002 control signal |
| 21 | | A1.T1.A1T1S001.SV.C | Q | 3.0 | BOOL | pump outlet educt tank B001 control signal |
| 22 | | A1.T1.A1T1S002.SV.C | Q | 3.1 | BOOL | pump outlet educt tank B002 control signal |
| 23 | | A1.T1.A1T1S003.SV.C | Q | 3.2 | BOOL | pump outlet educt tank B003 control signal |
| 24 | | A1.T4.A1T4S001.SV.C | Q | 3.3 | BOOL | pump rinsing control signal |
| 25 | | A1.T2.A1T2S003.SV.C | Q | 3.4 | BOOL | pump outlet reactor R001 control signal |
| 26 | | A1.T2.A1T2S004.SV.C | Q | 3.5 | BOOL | pump outlet reactor R002 control signal |
| 27 | | A1.T2.A1T2H007.HO... | Q | 4.0 | BOOL | reactor R001 stir status value |
| 28 | | A1.T2.A1T2H008.HO... | Q | 4.1 | BOOL | reactor R001 heating status value |
| 29 | | A1.T2.A1T2H011.HO... | Q | 4.2 | BOOL | reactor R001 empty status value |

## 8.2 Inserting the Motor_Lean block from the PCS 7 AP Library V90 in the project library

1. PCS 7 provides extensive libraries containing a large number of prepared blocks as well as pre-assembled charts, called templates. You will use such a template for the pump motor. For this purpose, open the PCS 7 AP Library V9.0. (→ File → Open → Libraries → PCS 7 AP Library V90 → OK)


SIMATIC Manager - [SCE_PCS7_MP (Component View) -- C:\Program Files (x86)\...\S7Proj\s7proj\SCE_PCS7\SCE__MP]

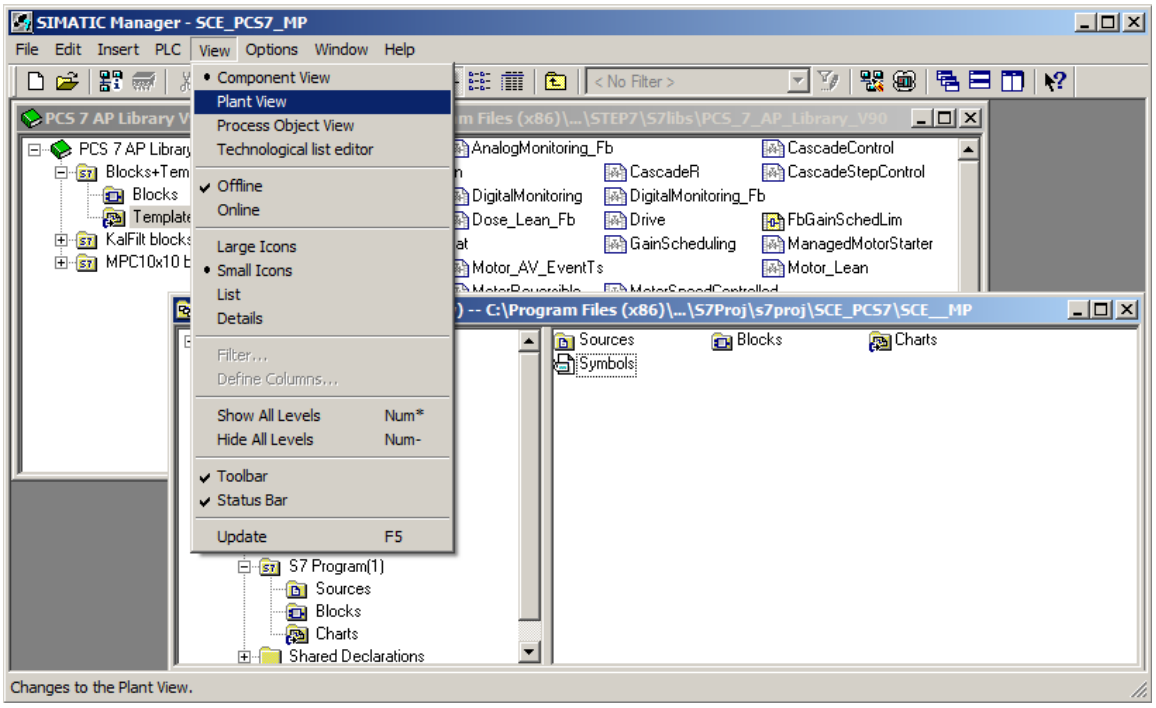
Opens a project to be selected or a library to be selected.


Open Project

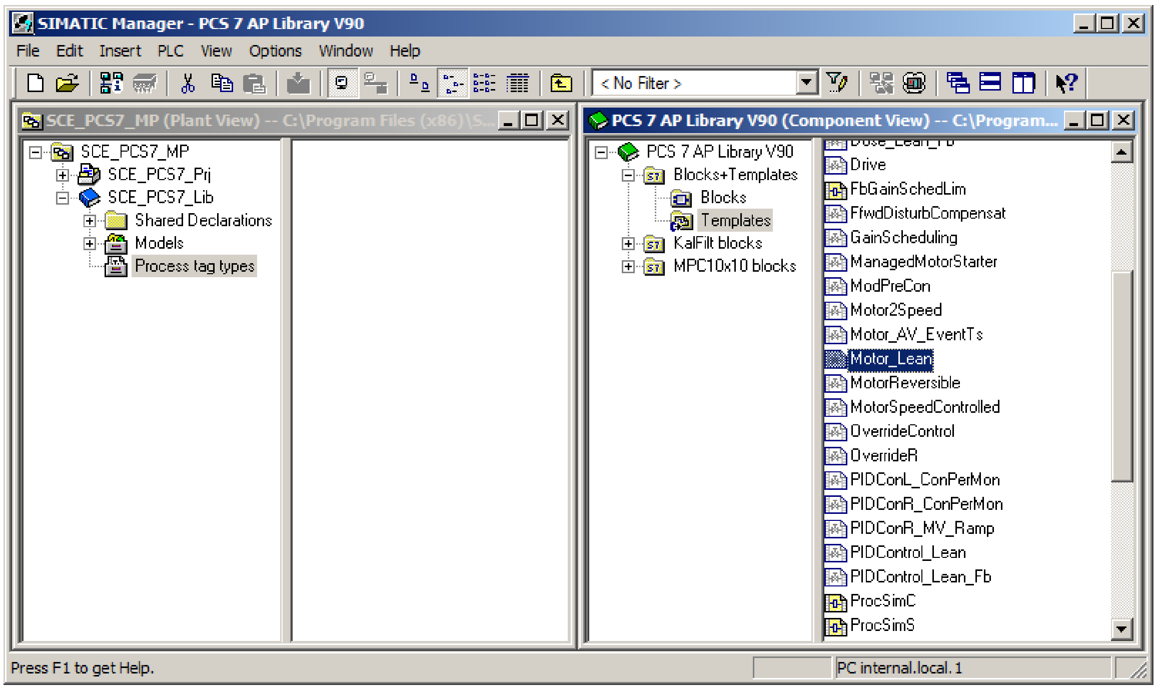
| Name | Storage path |
|---|---|
| CFC Library | C:\Program Files (x86)\SIEMENS\STEP7\S7libs\cfclib |
| LogicMatrix | C:\Program Files (x86)\SIEMENS\STEP7\S7libs\LM_Lib_V90 |
| PCS 7 AP Library V90 | C:\Program Files (x86)\SIEMENS\STEP7\S7libs\PCS_7_AP_Library_V90 |
| PCS 7 BasisLibrary V90 | C:\Program Files (x86)\SIEMENS\STEP7\S7libs\PCS_7_BASISLIBRARY_V90 |
| PLT-PR_Master_Lib | C:\Program Files (x86)\SIEMENS\STEP7\S7Proj\PLT-PR_M\PLT-_Lib |
| Redundant IO CGP V40 | C:\Program Files (x86)\SIEMENS\STEP7\S7libs\red_io_1 |
| Redundant IO CGP V52 | C:\Program Files (x86)\SIEMENS\STEP7\S7libs\red_io52 |
| Redundant IO MGP V32 | C:\Program Files (x86)\SIEMENS\STEP7\S7libs\red_io_0 |
| SCE_PCS7_Lib | C:\Program Files (x86)\SIEMENS\STEP7\S7Proj\s7proj\SCE_PCS7\SCE__Lib |
| SFC Library | C:\Program Files (x86)\SIEMENS\STEP7\S7libs\sfclib |
| SFC Library | C:\Program Files (x86)\SIEMENS\STEP7\S7libs\sfclib_60 |
| SFC Library | C:\Program Files (x86)\SIEMENS\STEP7\S7libs\sfclib_61 |
| SIMATIC_NET_CP | C:\Program Files (x86)\SIEMENS\STEP7\S7libs\simaticn |
| Standard Library | C:\Program Files (x86)\SIEMENS\STEP7\S7libs\stdlib30 |
| stdlibs (V2) | C:\Program Files (x86)\SIEMENS\STEP7\S7libs\stdlibs |

| | Selected |
|---|---|
| User projects: | |
| Libraries: | 1 |
| Sample projects: | |
| Multiprojects: | |

2. In order to move this template from the library to the project using drag-and-drop, go to the plant view of the project. (→ View → Plant View)

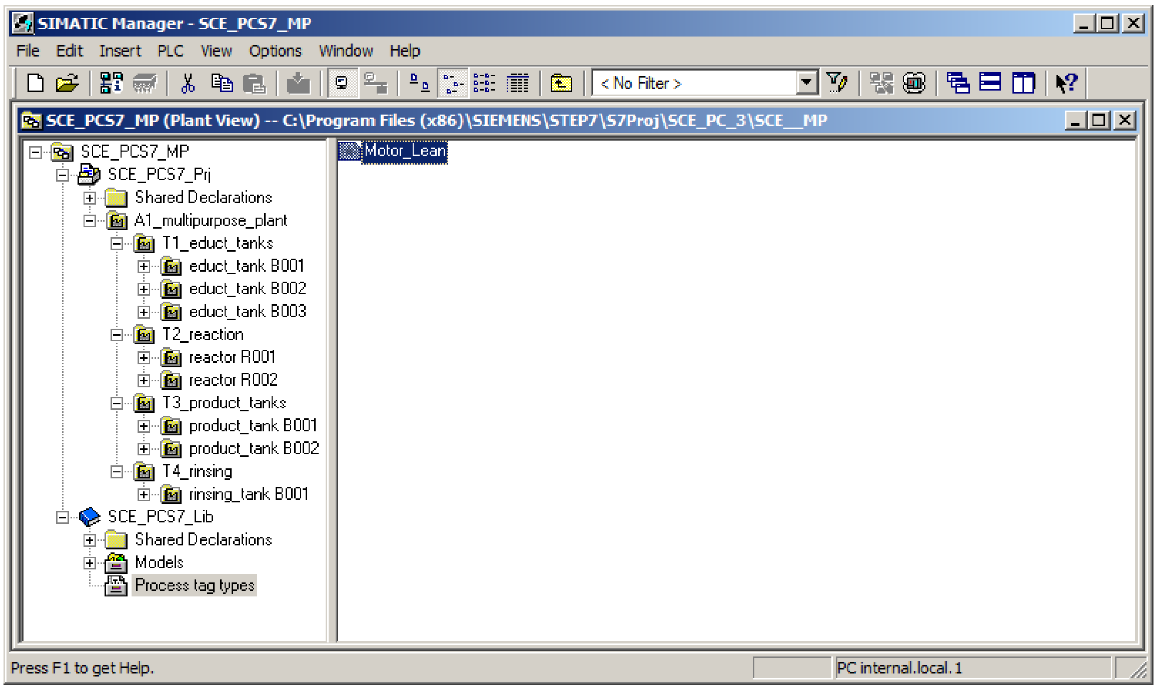


3. Using drag-and-drop, move Motor_Lean (Library, Templates) to Process tag types (master data library). (→ PCS 7 Library → Blocks+Templates → Templates → drag Motor_Lean to plant view → SCE_PCS7_Lib → Process tag types)



## 8.3 Centrally edit Motor_Lean

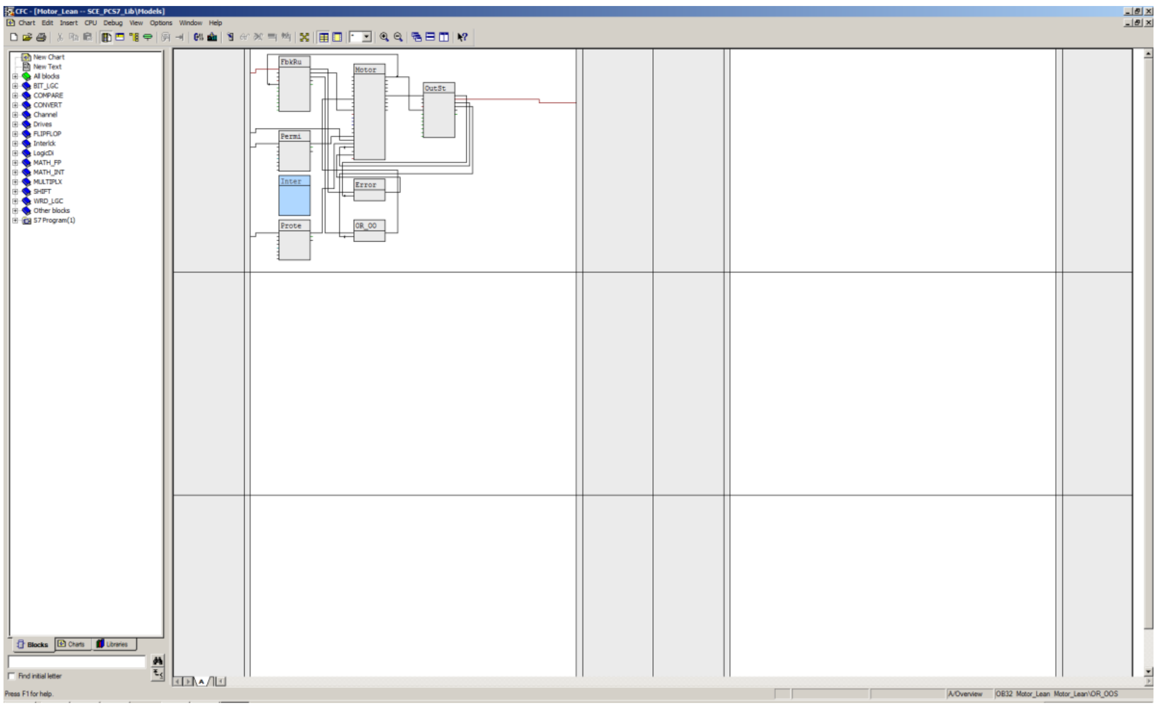
1. A change is now to be made centrally for all process tags of the 'Motor_Lean' type. This is done be opening the CFC 'Motor_Lean' from the master data library with a double click. (→ Motor_Lean)



***Note:***

– *CFC stands for Continuous Function Chart and is a graphic programming language for describing continuous processes. Pre-assembled blocks are placed, parameterized and interconnected in the CFC. This way, the programmer creates an overall software structure for open-loop and closed-loop control of a machine.*

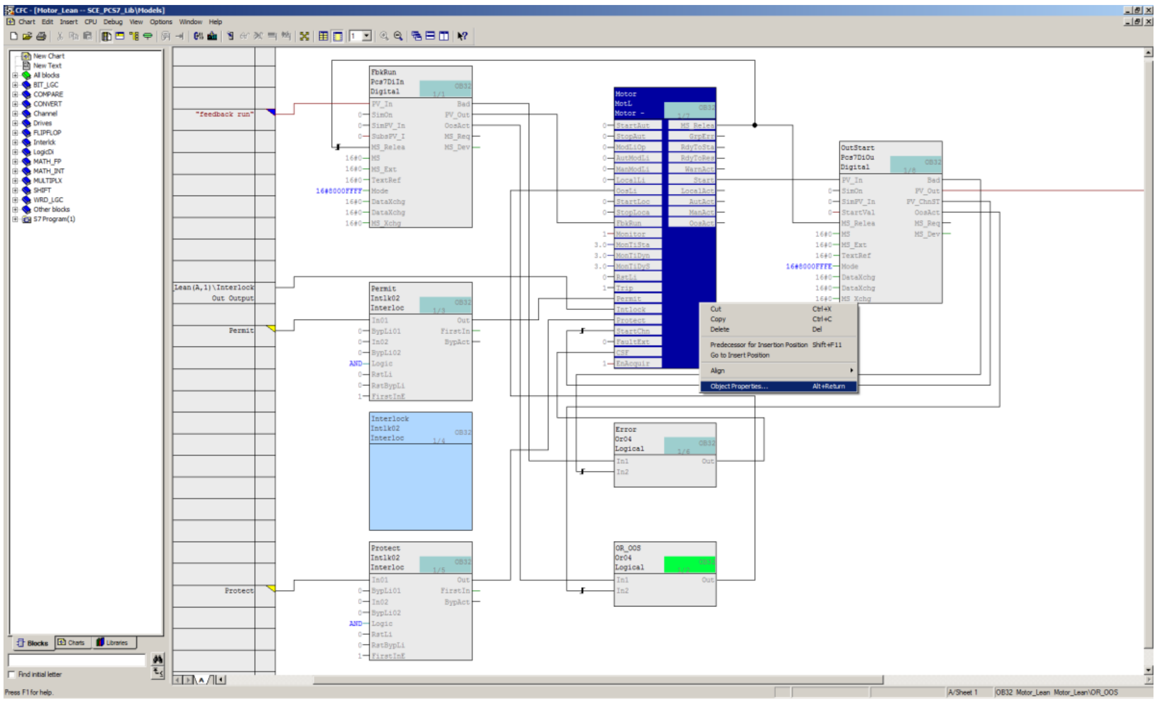
2. A CFC consists of chart partitions with 6 sheets each. In the overview, all six sheets are displayed with their gray sheet bars. The incoming signals are shown on the sheet bars on the left of the sheet and the outgoing signals on the sheet bars on the right. You can change to the sheet view by double-clicking a sheet. (→ Overview → Double-click on first sheet)



***Note:***

– *You can switch between the chart partitions (maximum A to Z) with the tabs in the bottom bar. There is only chart partition A here to start.*

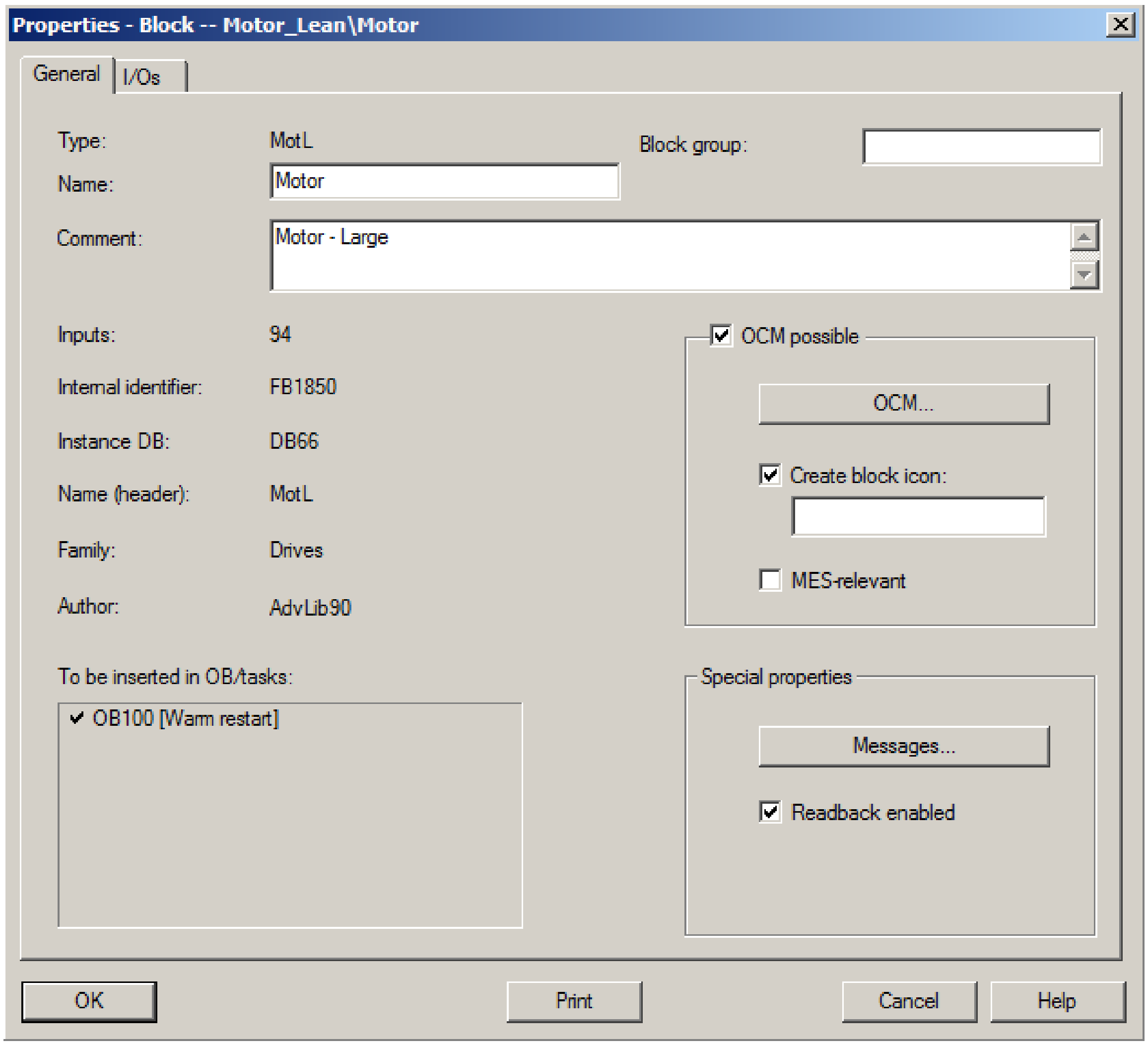
3. You want to make a change to the 'MotL' block in the process tag type 'Motor_Lean'. To do so, open its properties. (→ MotL → right-click → Object Properties)


CFC - [Motor_Lean -- SCE_PCS7_Lib\Models]
Chart Edit Insert CPU Debug View Options Window Help
Cut Ctrl+X
Copy Ctrl+C
Delete Del
Predecessor for Insertion Position Shift+F11
Go to Insert Position
Align
Object Properties... Alt+Return
Motor
MotL
FbkRun
Permit
Interlock
Protect
Error
OR_OOS
OutStart
"feedback run"
Blocks
Charts
Libraries
Press F1 for help.

***Note:***

– *If blocks turn blue and are displayed without I/Os, this means that insufficient space is available for the full display. In this case, the blocks must be shifted in the CFC until no more error messages are output.*

4. You are not interested in changing the general properties, such as whether operator control and monitoring is possible, so you change to the I/Os tab. (→ I/Os)


Properties - Block -- Motor_Lean\Motor

General | I/Os

| | |
|---|---|
| Type: | MotL |
| Name: | Motor |
| Comment: | Motor - Large |
| Block group: | |
| Inputs: | 94 |
| Internal identifier: | FB1850 |
| Instance DB: | DB66 |
| Name (header): | MotL |
| Family: | Drives |
| Author: | AdvLib90 |

[✓] OCM possible

OCM...

[✓] Create block icon:

[ ] MES-relevant

To be inserted in OB/tasks:

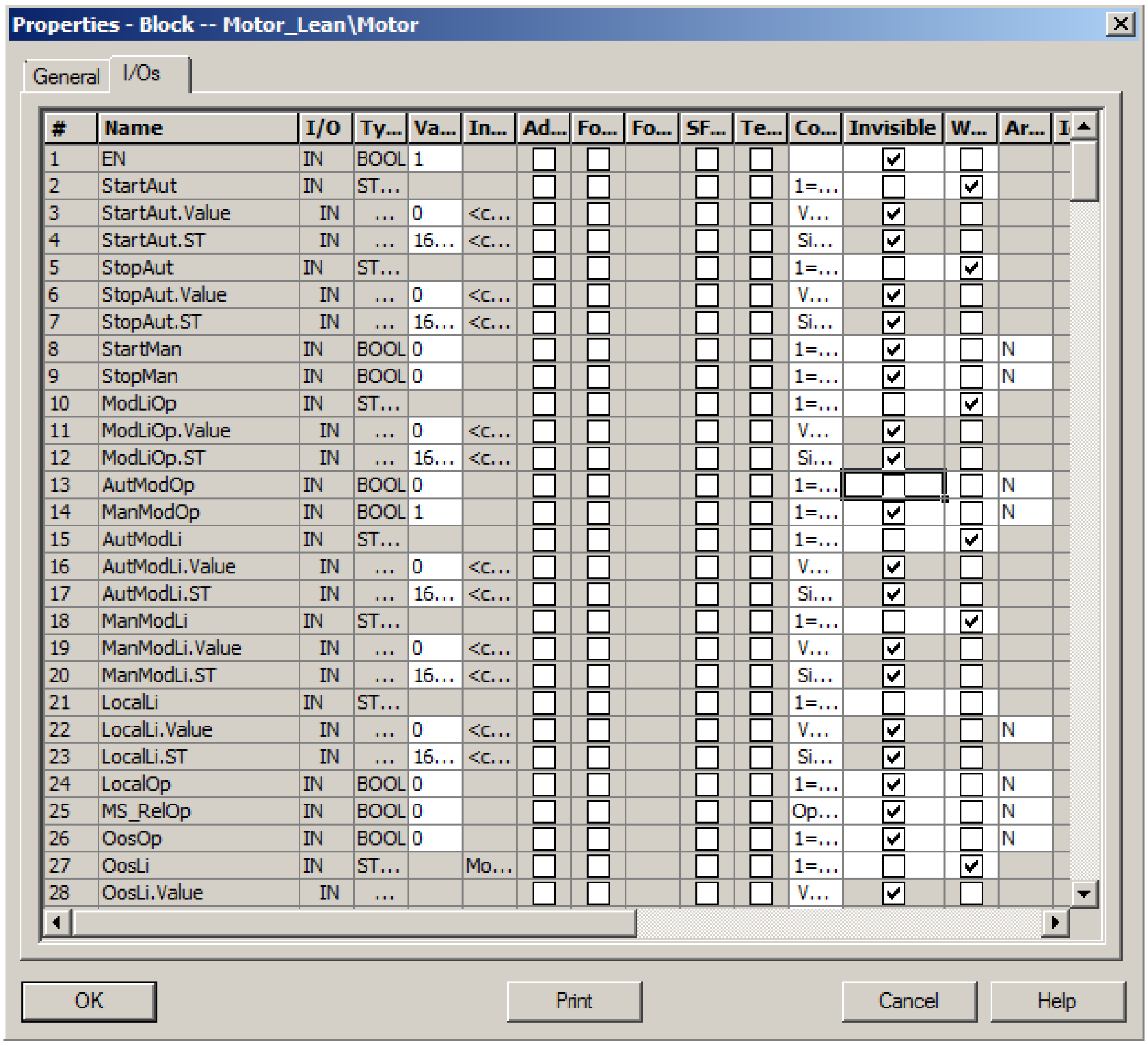
✓ OB100 [Warm restart]

Special properties

Messages...

[✓] Readback enabled

5. The I/Os are displayed in a table together with many configurable properties. You will get to know the most important properties below. For now, you merely want to cancel the invisibility for 'AutModOp' in the 'MotL' block. This will make this I/O visible in the sheet. Exit the properties with 'OK'. (→ AutModOp → Invisible → ☐ → OK)

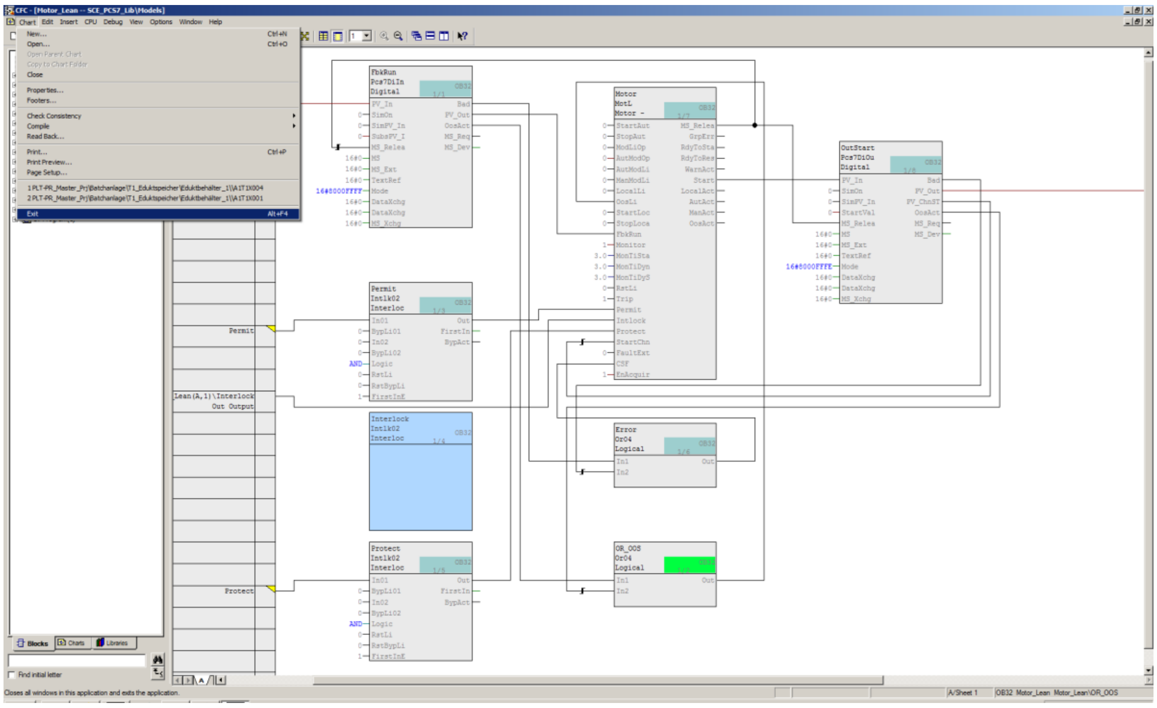


| # | Name | I/O | Ty... | Va... | In... | Ad... | Fo... | Fo... | SF... | Te... | Co... | Invisible | W... | Ar... | I |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | EN | IN | BOOL | 1 | | ☐ | ☐ | | ☐ | ☐ | | ✓ | ☐ | | |
| 2 | StartAut | IN | ST... | | | ☐ | ☐ | | ☐ | ☐ | 1=... | ☐ | ✓ | | |
| 3 | StartAut.Value | IN | ... | 0 | <c... | ☐ | ☐ | | ☐ | ☐ | V... | ✓ | ☐ | | |
| 4 | StartAut.ST | IN | ... | 16... | <c... | ☐ | ☐ | | ☐ | ☐ | Si... | ✓ | ☐ | | |
| 5 | StopAut | IN | ST... | | | ☐ | ☐ | | ☐ | ☐ | 1=... | ☐ | ✓ | | |
| 6 | StopAut.Value | IN | ... | 0 | <c... | ☐ | ☐ | | ☐ | ☐ | V... | ✓ | ☐ | | |
| 7 | StopAut.ST | IN | ... | 16... | <c... | ☐ | ☐ | | ☐ | ☐ | Si... | ✓ | ☐ | | |
| 8 | StartMan | IN | BOOL | 0 | | ☐ | ☐ | | ☐ | ☐ | 1=... | ✓ | ☐ | N | |
| 9 | StopMan | IN | BOOL | 0 | | ☐ | ☐ | | ☐ | ☐ | 1=... | ✓ | ☐ | N | |
| 10 | ModLiOp | IN | ST... | | | ☐ | ☐ | | ☐ | ☐ | 1=... | ☐ | ✓ | | |
| 11 | ModLiOp.Value | IN | ... | 0 | <c... | ☐ | ☐ | | ☐ | ☐ | V... | ✓ | ☐ | | |
| 12 | ModLiOp.ST | IN | ... | 16... | <c... | ☐ | ☐ | | ☐ | ☐ | Si... | ✓ | ☐ | | |
| 13 | AutModOp | IN | BOOL | 0 | | ☐ | ☐ | | ☐ | ☐ | 1=... | ☐ | ☐ | N | |
| 14 | ManModOp | IN | BOOL | 1 | | ☐ | ☐ | | ☐ | ☐ | 1=... | ✓ | ☐ | N | |
| 15 | AutModLi | IN | ST... | | | ☐ | ☐ | | ☐ | ☐ | 1=... | ☐ | ✓ | | |
| 16 | AutModLi.Value | IN | ... | 0 | <c... | ☐ | ☐ | | ☐ | ☐ | V... | ✓ | ☐ | | |
| 17 | AutModLi.ST | IN | ... | 16... | <c... | ☐ | ☐ | | ☐ | ☐ | Si... | ✓ | ☐ | | |
| 18 | ManModLi | IN | ST... | | | ☐ | ☐ | | ☐ | ☐ | 1=... | ☐ | ✓ | | |
| 19 | ManModLi.Value | IN | ... | 0 | <c... | ☐ | ☐ | | ☐ | ☐ | V... | ✓ | ☐ | | |
| 20 | ManModLi.ST | IN | ... | 16... | <c... | ☐ | ☐ | | ☐ | ☐ | Si... | ✓ | ☐ | | |
| 21 | LocalLi | IN | ST... | | | ☐ | ☐ | | ☐ | ☐ | 1=... | ☐ | ☐ | | |
| 22 | LocalLi.Value | IN | ... | 0 | <c... | ☐ | ☐ | | ☐ | ☐ | V... | ✓ | ☐ | N | |
| 23 | LocalLi.ST | IN | ... | 16... | <c... | ☐ | ☐ | | ☐ | ☐ | Si... | ✓ | ☐ | | |
| 24 | LocalOp | IN | BOOL | 0 | | ☐ | ☐ | | ☐ | ☐ | 1=... | ✓ | ☐ | N | |
| 25 | MS_RelOp | IN | BOOL | 0 | | ☐ | ☐ | | ☐ | ☐ | Op... | ✓ | ☐ | N | |
| 26 | OosOp | IN | BOOL | 0 | | ☐ | ☐ | | ☐ | ☐ | 1=... | ✓ | ☐ | N | |
| 27 | OosLi | IN | ST... | | Mo... | ☐ | ☐ | | ☐ | ☐ | 1=... | ☐ | ✓ | | |
| 28 | OosLi.Value | IN | ... | | | ☐ | ☐ | | ☐ | ☐ | V... | ✓ | ☐ | | |

***Note:***

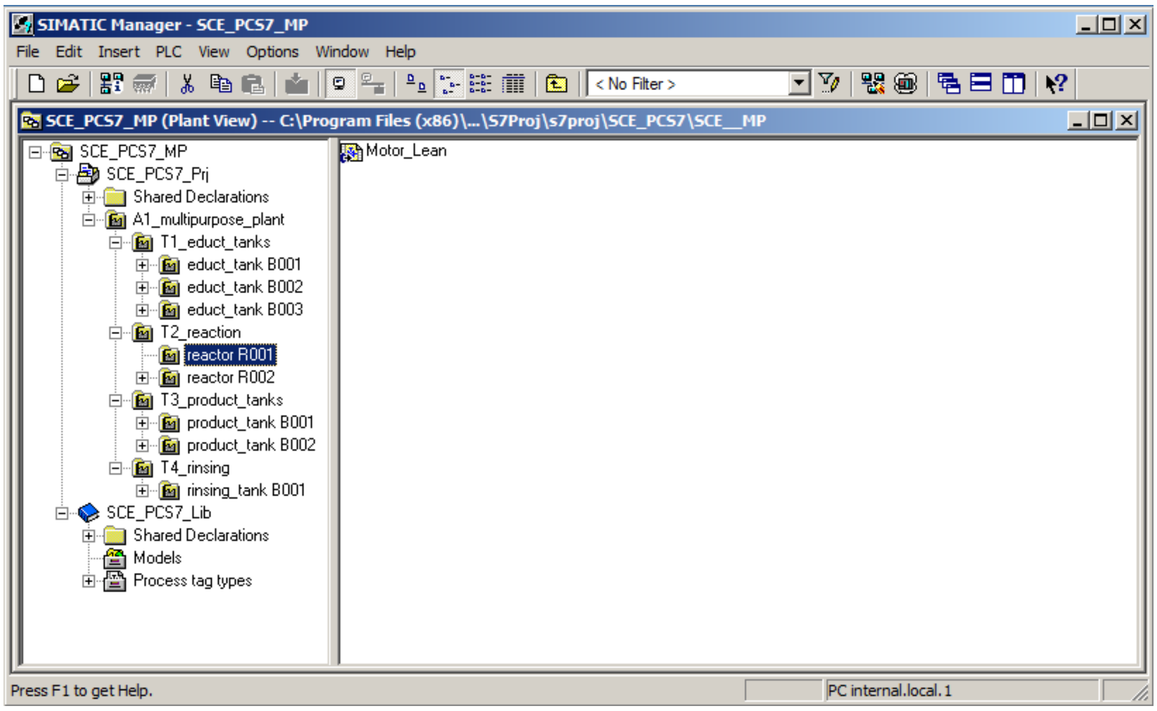
– *If you click on the header of a property, the display can be sorted alphabetically by this column.*

6. The process tag type can now simply be closed. A save operation occurs automatically at each change. (→ Chart → Exit)

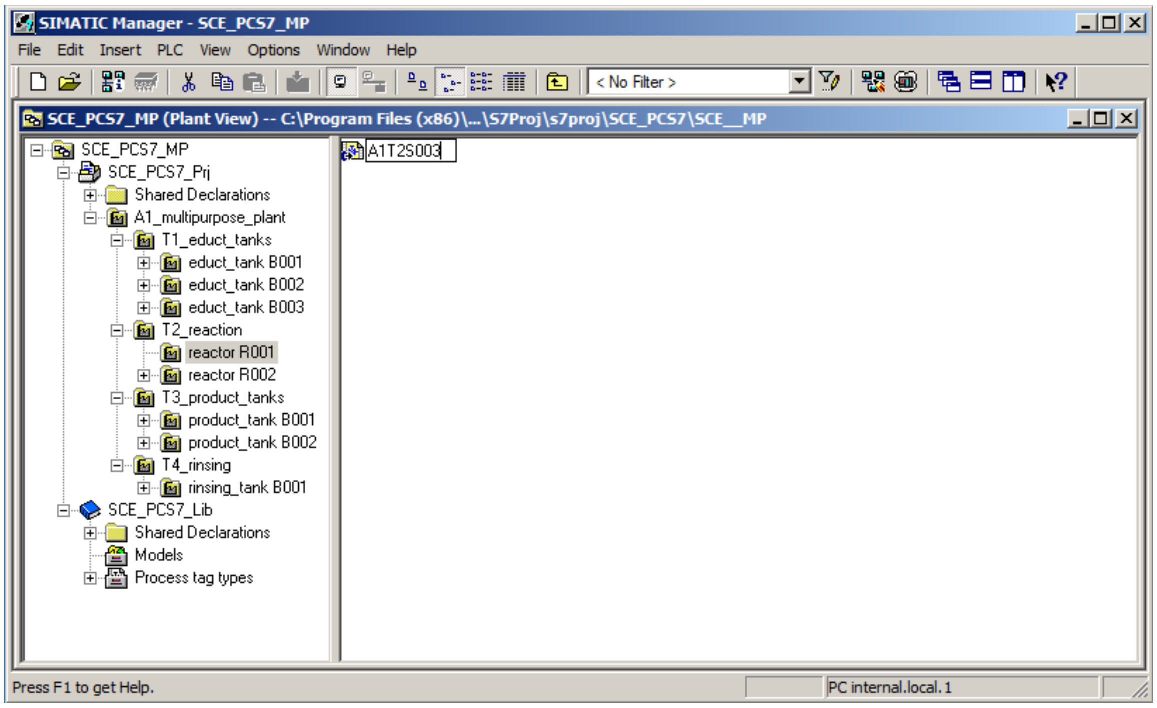


## 8.4 Implementation of an instance of Motor_Lean

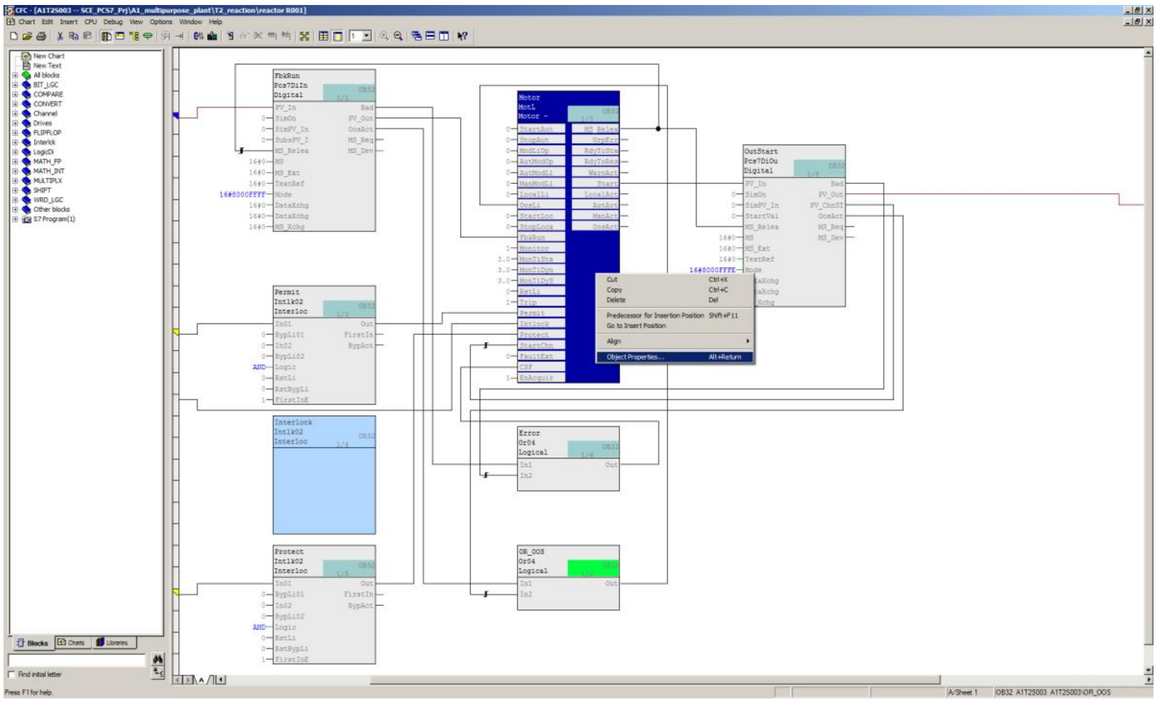
1. To use the 'Motor_Lean' process tag type, it is moved directly to the 'Reactor R001' chart folder using drag-and-drop. (→ Motor_Lean → Reactor R001)



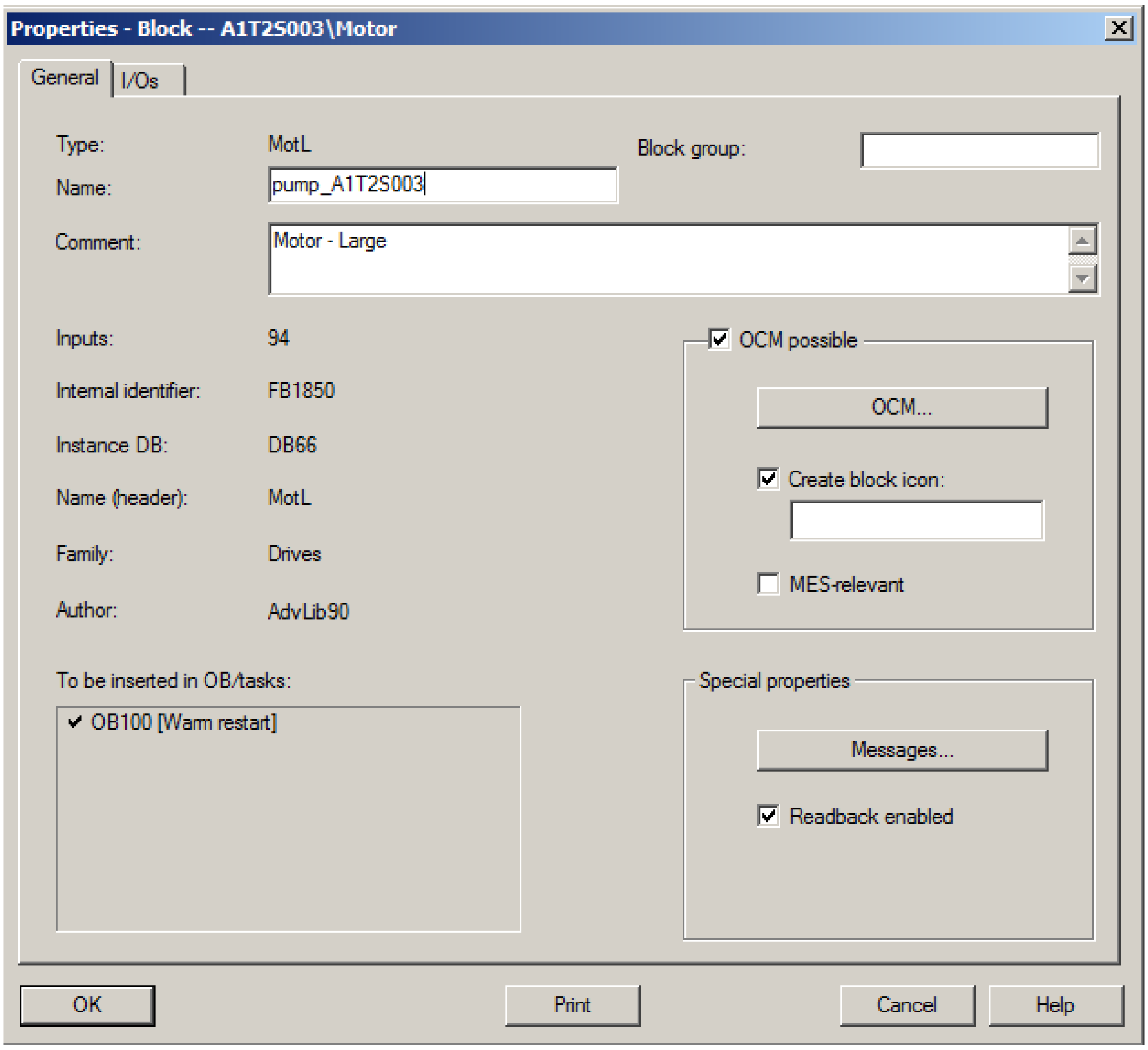
2. Because this chart is to be used to control pump A1T2S003, it is now renamed A1T2S003 and opened with a double-click. (→ Reactor R001 → rename Motor_Lean to A1T2S003 → A1T2S003)



3. The properties of the 'MotL' block are then opened with a right-click. (→ Motor_Lean → Object Properties)



4. The name of the block is changed in the general properties. (→ pump_A1T2S003 → OK)



Properties - Block -- A1T2S003\Motor

Type: MotL

Name: pump_A1T2S003

Block group:

Comment: Motor - Large

Inputs: 94

Internal identifier: FB1850

Instance DB: DB66

Name (header): MotL

Family: Drives

Author: AdvLib90

☑ OCM possible

☑ Create block icon:

☐ MES-relevant

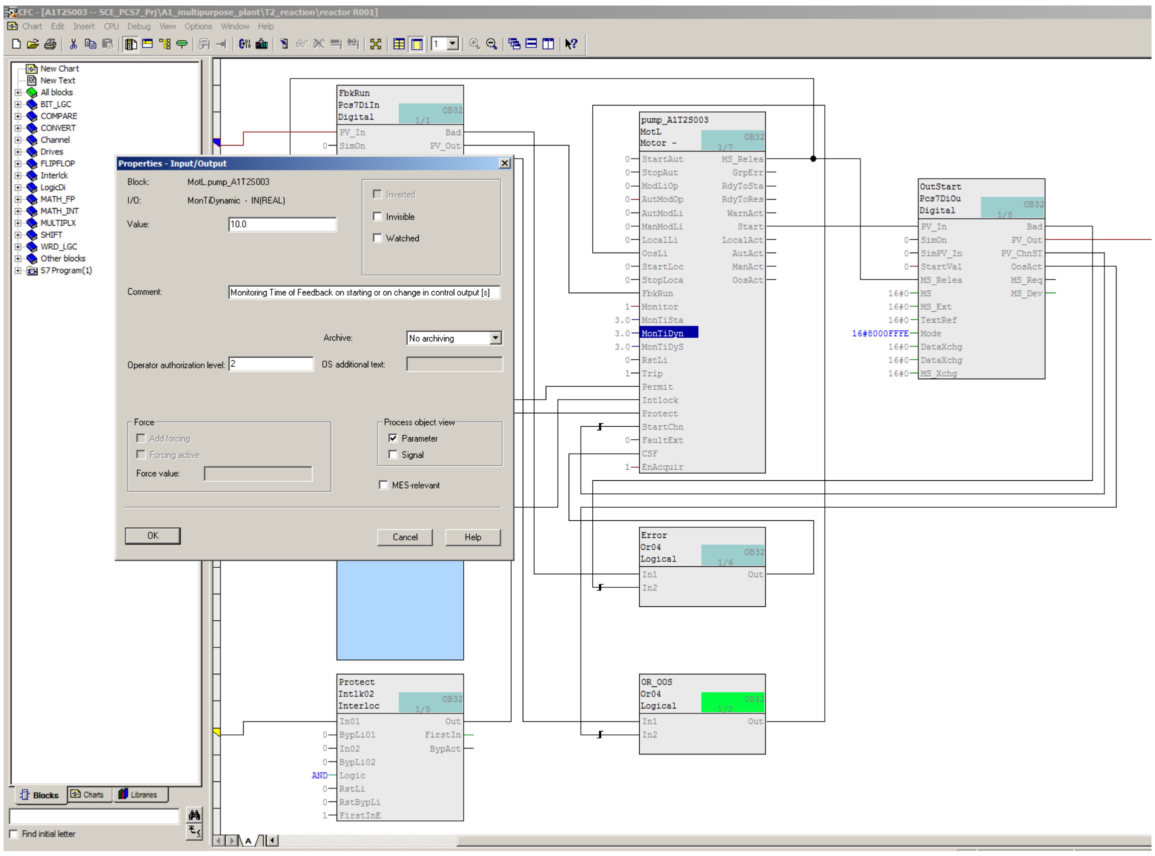
To be inserted in OB/tasks:

✓ OB100 [Warm restart]

Special properties

☑ Readback enabled

5. Now, you want to change the time for the feedback monitoring after successful operation of the motor block to 10.0 seconds. The properties dialog for the 'MonTiDyn' input is opened for this purpose and 10.0 is entered for Value. Exit the dialog again with 'OK'. (→ MonTiDyn → Value: 10.0 → OK)


Properties - Input/Output

Block: MotL.pump_A1T2S003

I/O: MonTiDynamic - IN(REAL)

Value: 10.0

☐ Inverted

☐ Invisible

☐ Watched

Comment: Monitoring Time of Feedback on starting or on change in control output [s]

Archive: No archiving

Operator authorization level: 2

OS additional text:

Force

☐ Add forcing

☐ Forcing active

Force value:

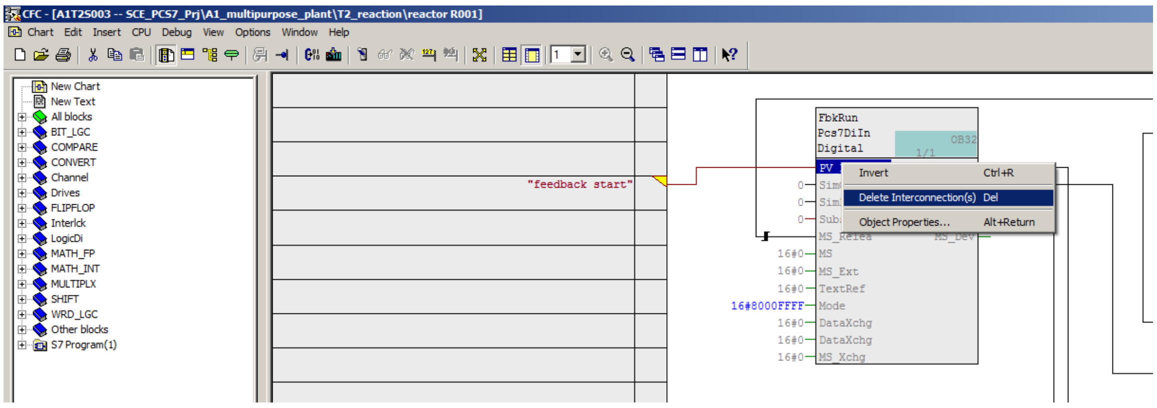
Process object view

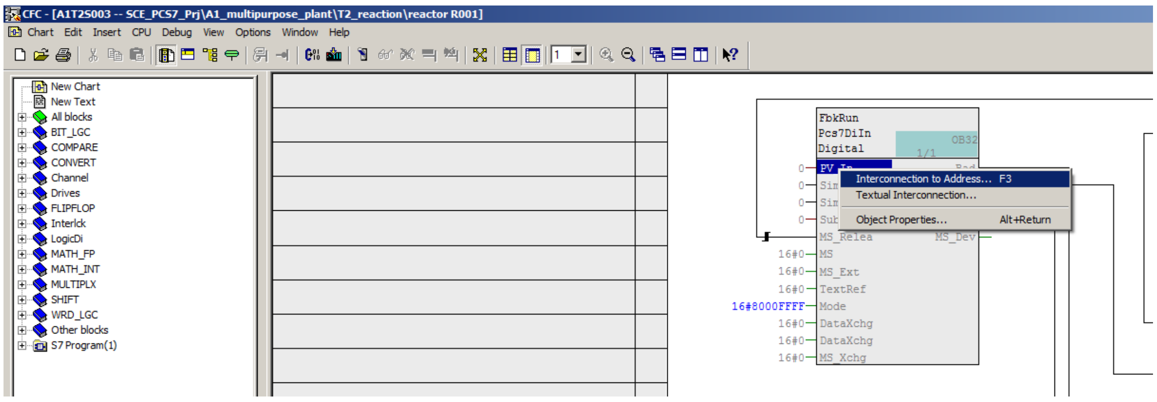
☑ Parameter

☐ Signal

☐ MES-relevant

6. Next, the feedback is interconnected with the input address. This is done by right-clicking the 'PV_In' input on the 'FbkRun' block (PCS7DiIn) and selecting the interconnection to the address. (→ FbkRun → PV_In → Delete Interconnection(s) → Interconnection to Address)



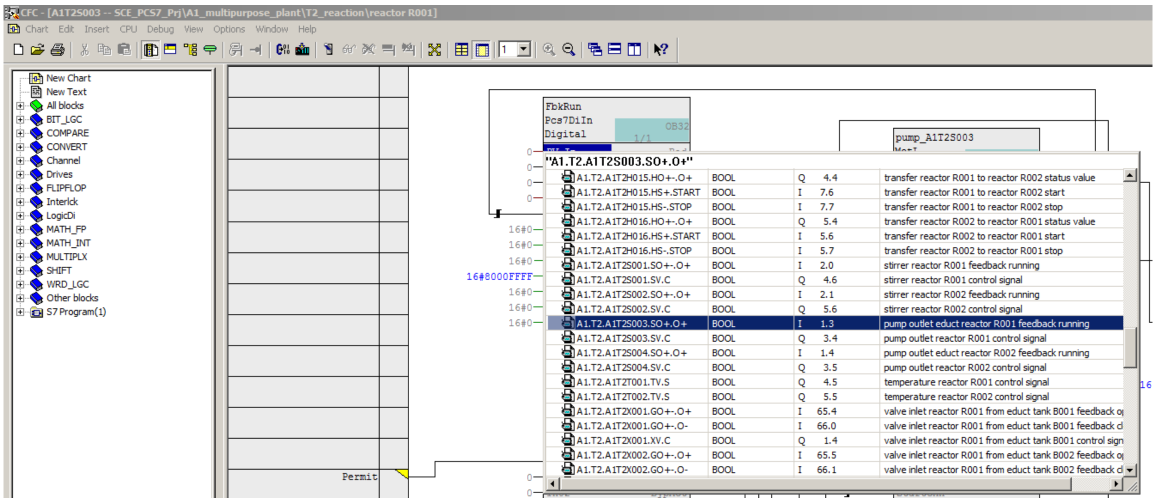

CFC - [A1T25003 -- SCE_PCS7_Prj\A1_multipurpose_plant\T2_reaction\reactor R001]
Chart Edit Insert CPU Debug View Options Window Help
FbkRun
Pcs7DiIn
Digital
OB32
Interconnection to Address... F3
Textual Interconnection...
Object Properties... Alt+Return
16#8000FFFF
Mode
DataXchg
MS_Xchg

***Notes:***

– *Like the PCS7DiOu block, the PCS7DiIn block is a driver block for interfacing with the PLC I/O. If the value 'PV_In' on the input block or the value 'PV_Out' on the output block is interconnected with an address that is also configured in the hardware configuration, the MODE input is automatically supplied with data later during compilation.*

– *For this to happen, 'Generate module drivers' must be selected later for the compilation.*

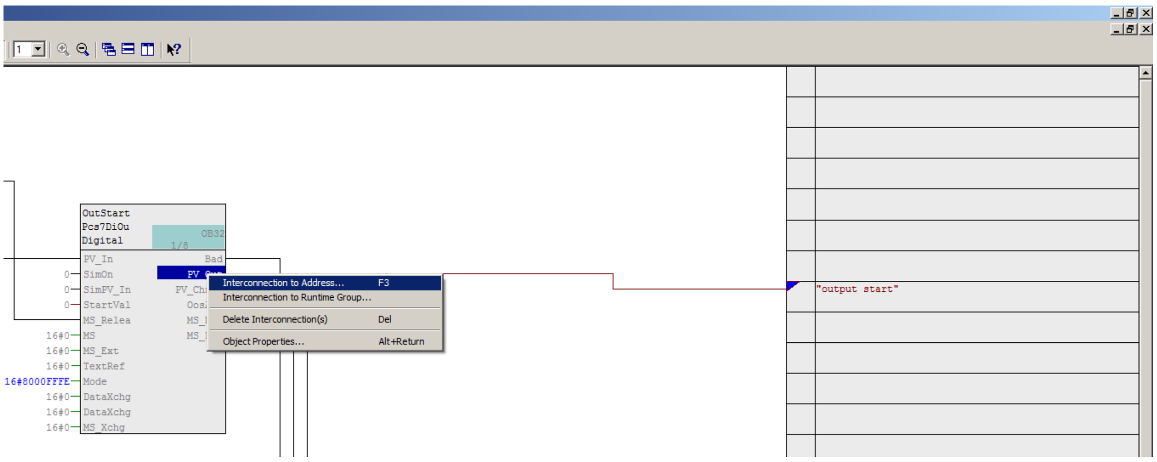
7. The user selects the address that provides feedback on whether the pump is running directly from the symbol table. (→ A1.T2.A1T2S003.S0+.0+)



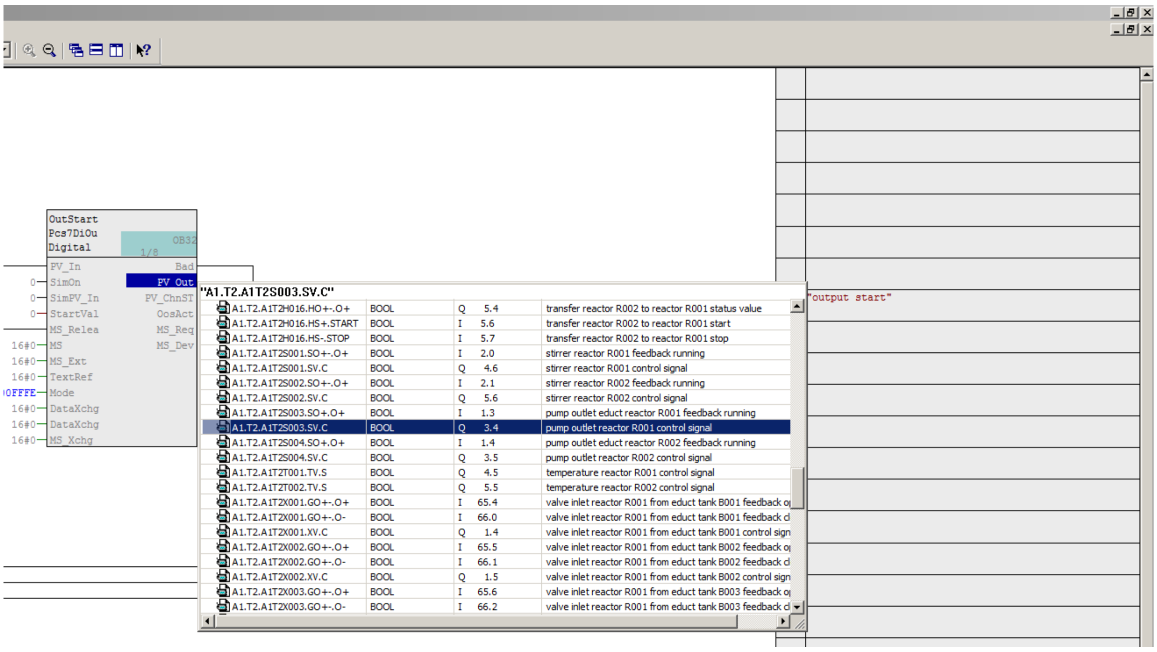
"A1.T2.A1T2S003.SO+.O+"

| | | | | |
|---|---|---|---|---|
| A1.T2.A1T2H015.HO+-.O+ | BOOL | Q | 4.4 | transfer reactor R001 to reactor R002 status value |
| A1.T2.A1T2H015.HS+.START | BOOL | I | 7.6 | transfer reactor R001 to reactor R002 start |
| A1.T2.A1T2H015.HS-.STOP | BOOL | I | 7.7 | transfer reactor R001 to reactor R002 stop |
| A1.T2.A1T2H016.HO+-.O+ | BOOL | Q | 5.4 | transfer reactor R002 to reactor R001 status value |
| A1.T2.A1T2H016.HS+.START | BOOL | I | 5.6 | transfer reactor R002 to reactor R001 start |
| A1.T2.A1T2H016.HS-.STOP | BOOL | I | 5.7 | transfer reactor R002 to reactor R001 stop |
| A1.T2.A1T2S001.SO+-.O+ | BOOL | I | 2.0 | stirrer reactor R001 feedback running |
| A1.T2.A1T2S001.SV.C | BOOL | Q | 4.6 | stirrer reactor R001 control signal |
| A1.T2.A1T2S002.SO+-.O+ | BOOL | I | 2.1 | stirrer reactor R002 feedback running |
| A1.T2.A1T2S002.SV.C | BOOL | Q | 5.6 | stirrer reactor R002 control signal |
| A1.T2.A1T2S003.SO+.O+ | BOOL | I | 1.3 | pump outlet educt reactor R001 feedback running |
| A1.T2.A1T2S003.SV.C | BOOL | Q | 3.4 | pump outlet reactor R001 control signal |
| A1.T2.A1T2S004.SO+.O+ | BOOL | I | 1.4 | pump outlet educt reactor R002 feedback running |
| A1.T2.A1T2S004.SV.C | BOOL | Q | 3.5 | pump outlet reactor R002 control signal |
| A1.T2.A1T2T001.TV.S | BOOL | Q | 4.5 | temperature reactor R001 control signal |
| A1.T2.A1T2T002.TV.S | BOOL | Q | 5.5 | temperature reactor R002 control signal |
| A1.T2.A1T2X001.GO+-.O+ | BOOL | I | 65.4 | valve inlet reactor R001 from educt tank B001 feedback o |
| A1.T2.A1T2X001.GO+-.O- | BOOL | I | 66.0 | valve inlet reactor R001 from educt tank B001 feedback c |
| A1.T2.A1T2X001.XV.C | BOOL | Q | 1.4 | valve inlet reactor R001 from educt tank B001 control sign |
| A1.T2.A1T2X002.GO+-.O+ | BOOL | I | 65.5 | valve inlet reactor R001 from educt tank B002 feedback o |
| A1.T2.A1T2X002.GO+-.O- | BOOL | I | 66.1 | valve inlet reactor R001 from educt tank B002 feedback c |

8. Next, the control of the output address is interconnected. This is done by right-clicking the 'PV_Out' output on the 'OutStart' block (PCS7DiOu) and selecting the interconnection to the address.
   (→ OutStart → PV_Out → Interconnection to Address)



9. The address for control of the pump can now be easily selected directly from the symbol table. (→ A1.T2.A1T2S003.SV.C)



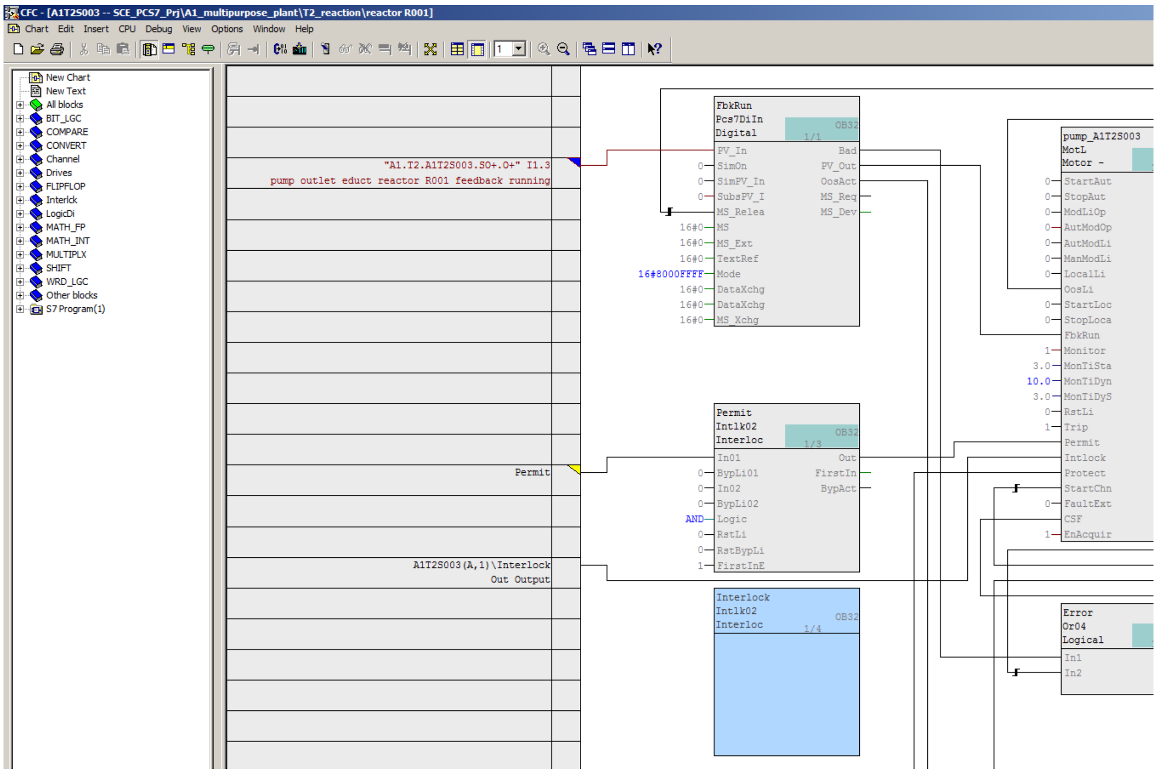
"A1.T2.A1T2S003.SV.C"

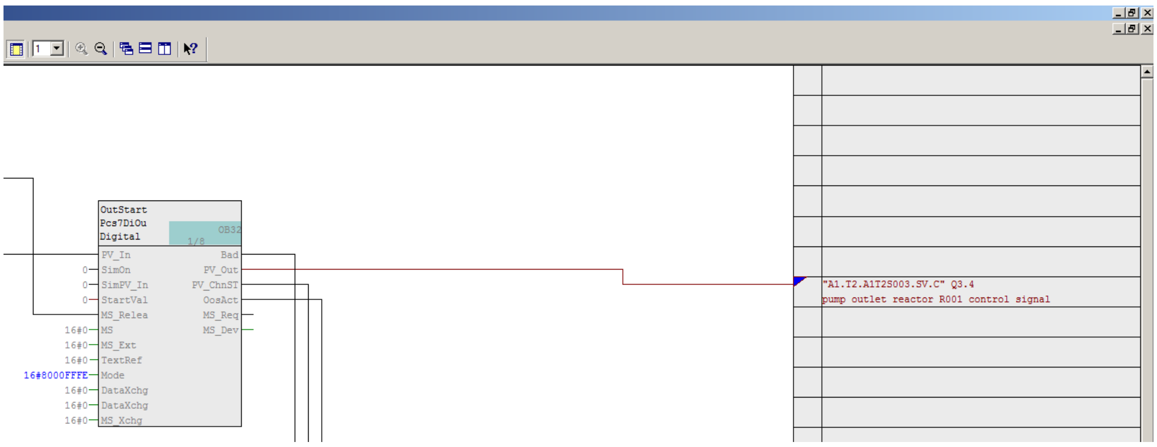
| | | | | |
|---|---|---|---|---|
| A1.T2.A1T2H016.HO+-.O+ | BOOL | Q | 5.4 | transfer reactor R002 to reactor R001 status value |
| A1.T2.A1T2H016.HS+.START | BOOL | I | 5.6 | transfer reactor R002 to reactor R001 start |
| A1.T2.A1T2H016.HS-.STOP | BOOL | I | 5.7 | transfer reactor R002 to reactor R001 stop |
| A1.T2.A1T2S001.SO+-.O+ | BOOL | I | 2.0 | stirrer reactor R001 feedback running |
| A1.T2.A1T2S001.SV.C | BOOL | Q | 4.6 | stirrer reactor R001 control signal |
| A1.T2.A1T2S002.SO+-.O+ | BOOL | I | 2.1 | stirrer reactor R002 feedback running |
| A1.T2.A1T2S002.SV.C | BOOL | Q | 5.6 | stirrer reactor R002 control signal |
| A1.T2.A1T2S003.SO+.O+ | BOOL | I | 1.3 | pump outlet educt reactor R001 feedback running |
| A1.T2.A1T2S003.SV.C | BOOL | Q | 3.4 | pump outlet reactor R001 control signal |
| A1.T2.A1T2S004.SO+.O+ | BOOL | I | 1.4 | pump outlet educt reactor R002 feedback running |
| A1.T2.A1T2S004.SV.C | BOOL | Q | 3.5 | pump outlet reactor R002 control signal |
| A1.T2.A1T2T001.TV.S | BOOL | Q | 4.5 | temperature reactor R001 control signal |
| A1.T2.A1T2T002.TV.S | BOOL | Q | 5.5 | temperature reactor R002 control signal |
| A1.T2.A1T2X001.GO+-.O+ | BOOL | I | 65.4 | valve inlet reactor R001 from educt tank B001 feedback o |
| A1.T2.A1T2X001.GO+-.O- | BOOL | I | 66.0 | valve inlet reactor R001 from educt tank B001 feedback cl |
| A1.T2.A1T2X001.XV.C | BOOL | Q | 1.4 | valve inlet reactor R001 from educt tank B001 control sign |
| A1.T2.A1T2X002.GO+-.O+ | BOOL | I | 65.5 | valve inlet reactor R001 from educt tank B002 feedback o |
| A1.T2.A1T2X002.GO+-.O- | BOOL | I | 66.1 | valve inlet reactor R001 from educt tank B002 feedback cl |
| A1.T2.A1T2X002.XV.C | BOOL | Q | 1.5 | valve inlet reactor R001 from educt tank B002 control sign |
| A1.T2.A1T2X003.GO+-.O+ | BOOL | I | 65.6 | valve inlet reactor R001 from educt tank B003 feedback o |
| A1.T2.A1T2X003.GO+-.O- | BOOL | I | 66.2 | valve inlet reactor R001 from educt tank B003 feedback cl |

***Note:***

– *The placeholder at the output of Pcs7DiOu 'output start' should be deleted; otherwise, there will be a warning during compilation!*

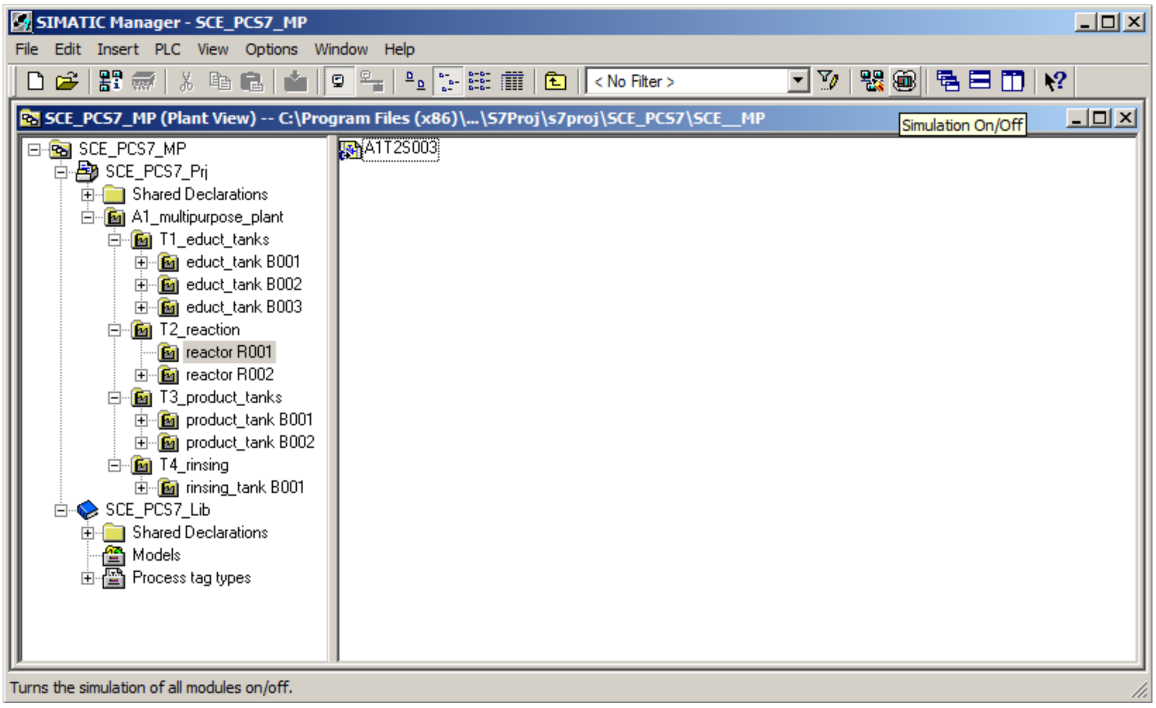
10. The result looks like this:



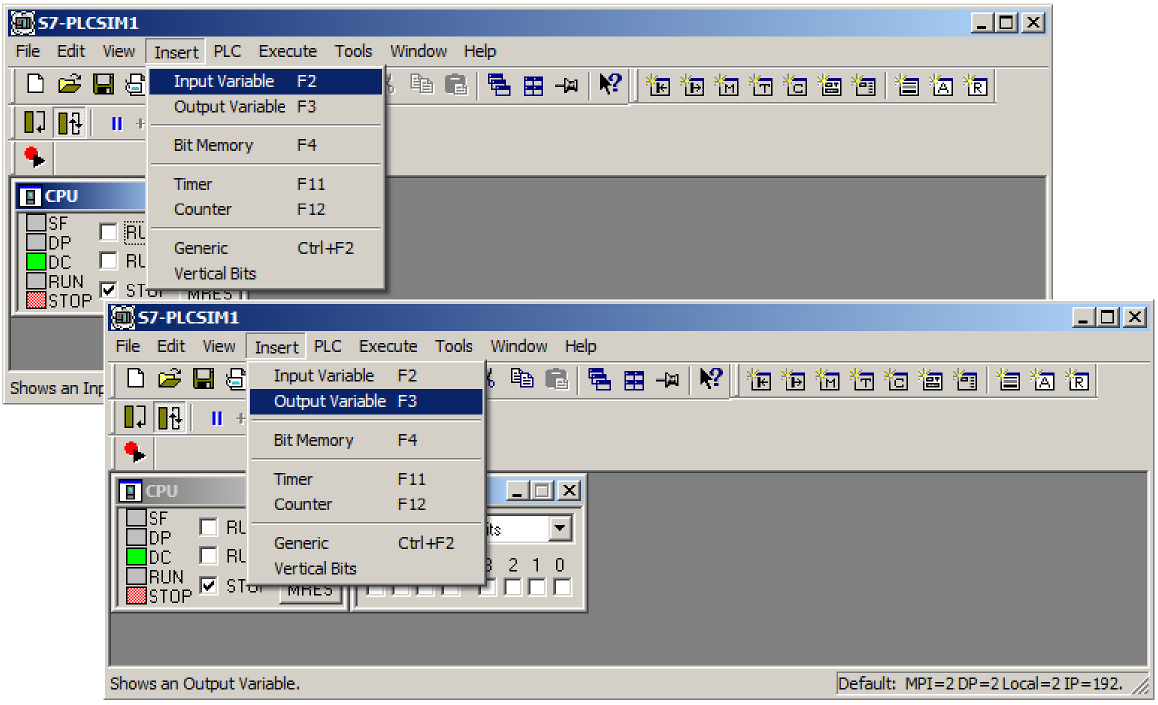

OutStart
Pcs7DiOu
Digital
OB32
1/6
"A1.T2.A1T2S003.SV.C" Q3.4
pump outlet reactor R001 control signal

## 8.5 Preparing a simulation with PLCSIM

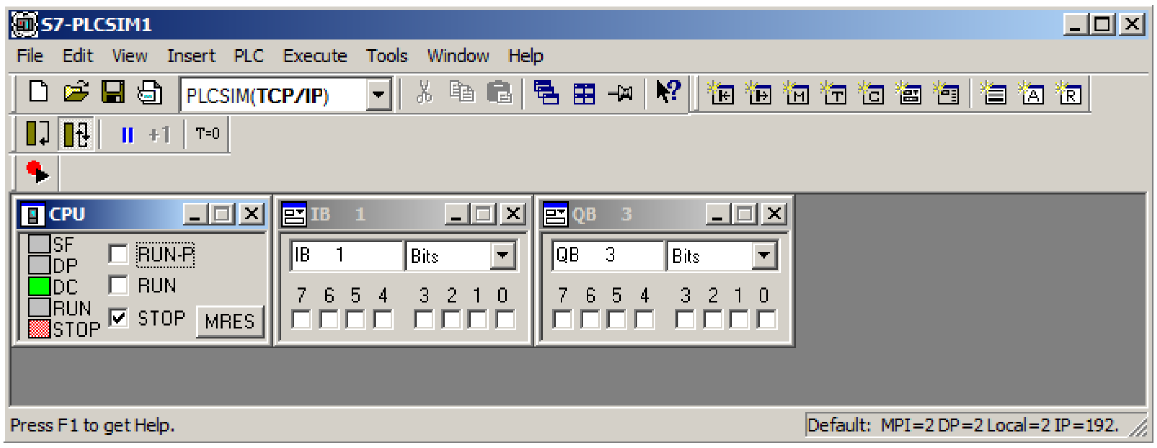
1. Before the program for the pump motor can be compiled and downloaded, the PLC simulation ***S7-PLCSIM*** must be started from SIMATIC Manager. (→ SIMATIC Manager → )



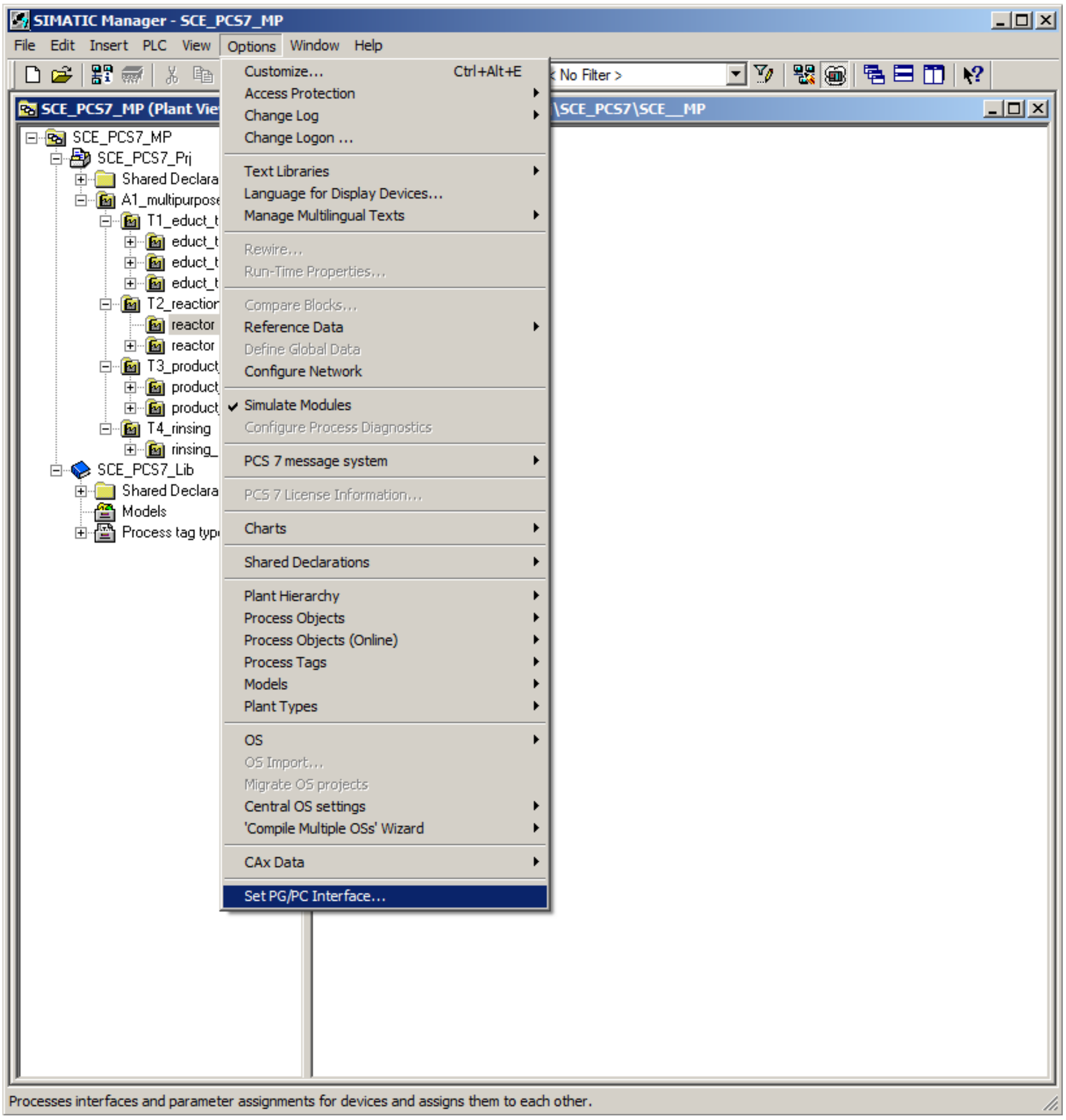
2. The PLC simulation behaves like a real SIMATIC S7 CPU. However, inputs and outputs must be inserted first before they can be monitored and operated. (→ Insert → Input Variable → Insert → Output Variable)



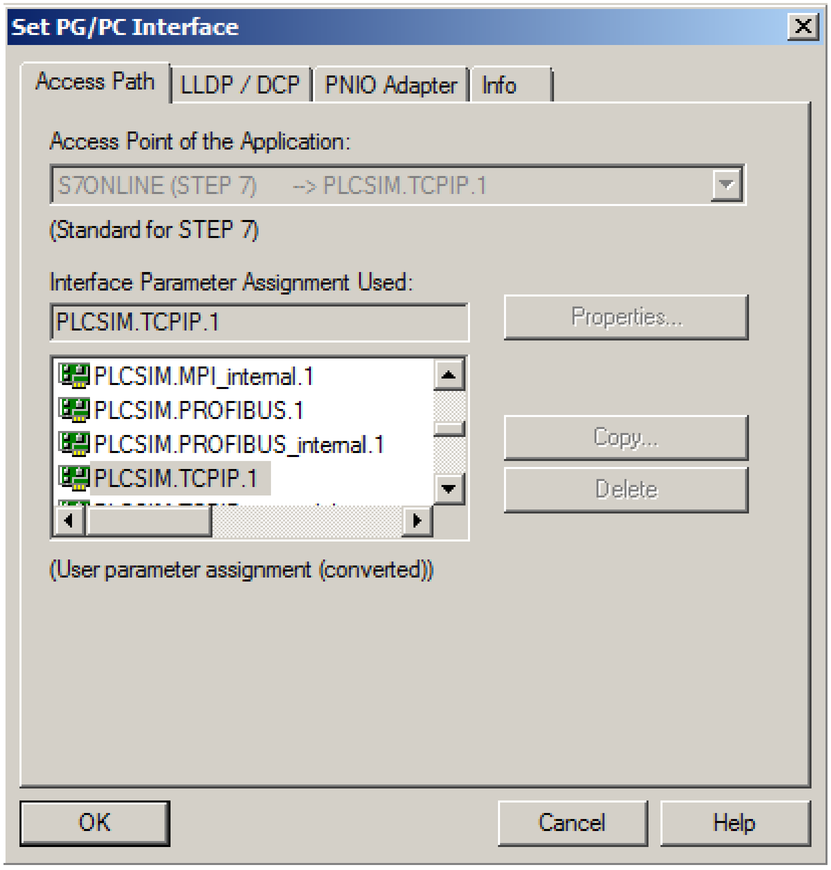
3. The correct byte addresses must now also be entered. (→ IB1 → QB3)



4. So that the program can be downloaded from SIMATIC Manager to ***S7-PLCSIM*** via the correct interface, the PG/PC interface still has to be correctly set. (→ SIMATIC Manager → Options → Set PG/PC Interface)



5. PLCSIM.TCPIP.1 is set here as the interface. (→ PLCSIM.TCPIP.1 → OK)


Set PG/PC Interface

Access Path | LLDP / DCP | PNIO Adapter | Info

Access Point of the Application:
S7ONLINE (STEP 7) --> PLCSIM.TCPIP.1
(Standard for STEP 7)

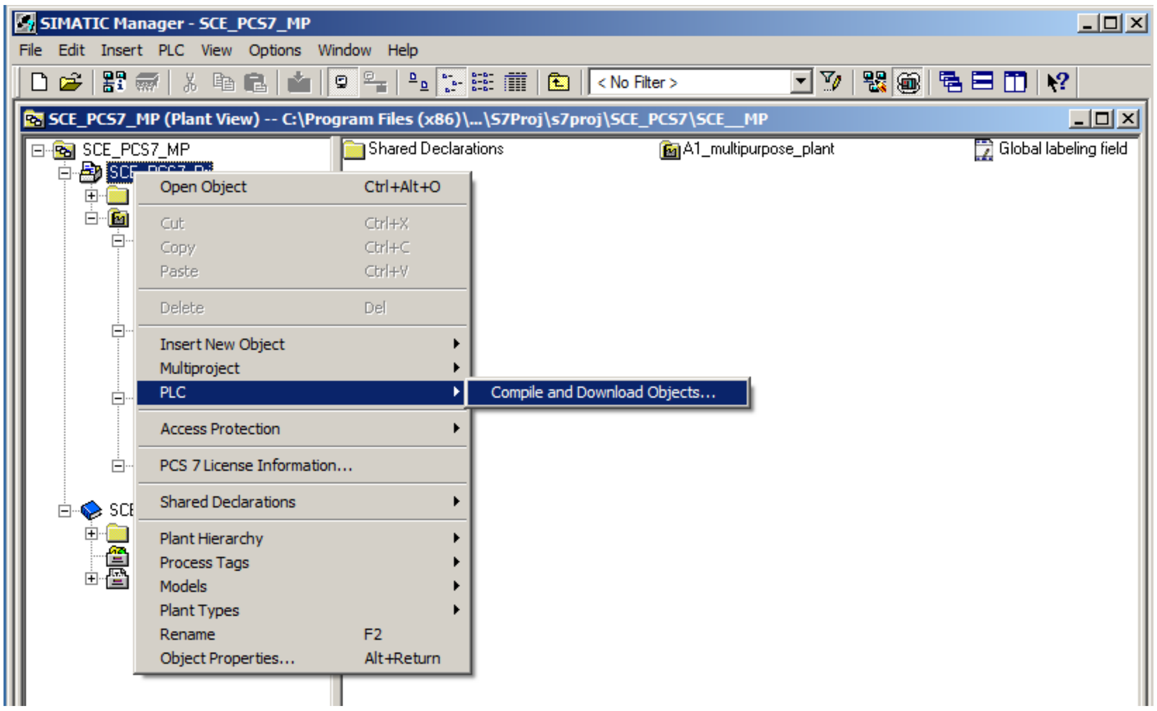
Interface Parameter Assignment Used:
PLCSIM.TCPIP.1

PLCSIM.MPI_internal.1
PLCSIM.PROFIBUS.1
PLCSIM.PROFIBUS_internal.1
PLCSIM.TCPIP.1

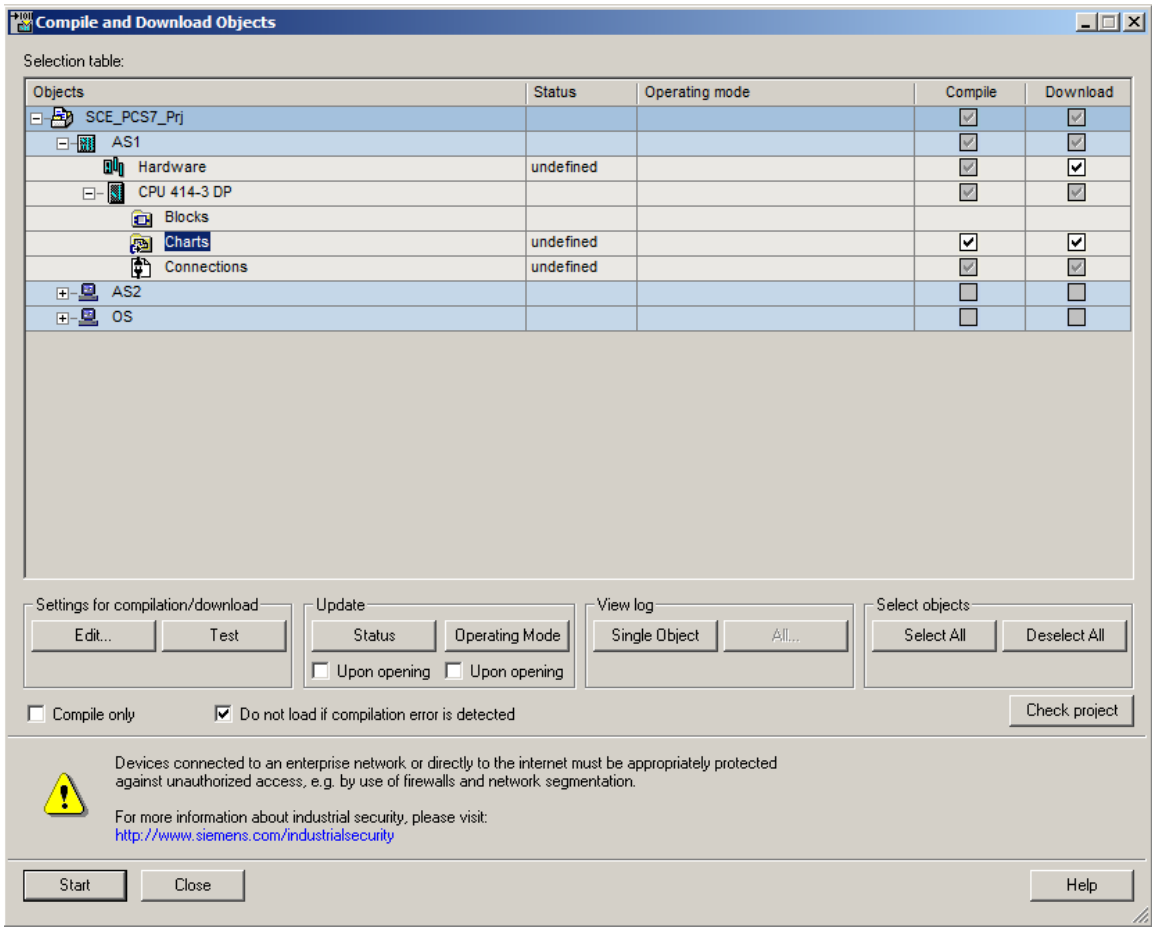
(User parameter assignment (converted))

## 8.6 Compiling and downloading objects

1. Now the project folder can be selected and the objects can be compiled and downloaded. (→ Plant View → SCE_PCS7_Prj → PLC → Compile and Download Objects)



2. In the following selection, 'Compile' and 'Download' are selected for the hardware and the charts of AS1. The 'Charts' folder is then selected and its settings are checked using 'Edit'. (→☑ → ☑ → ☑ → Charts → Edit)


Compile and Download Objects

Selection table:

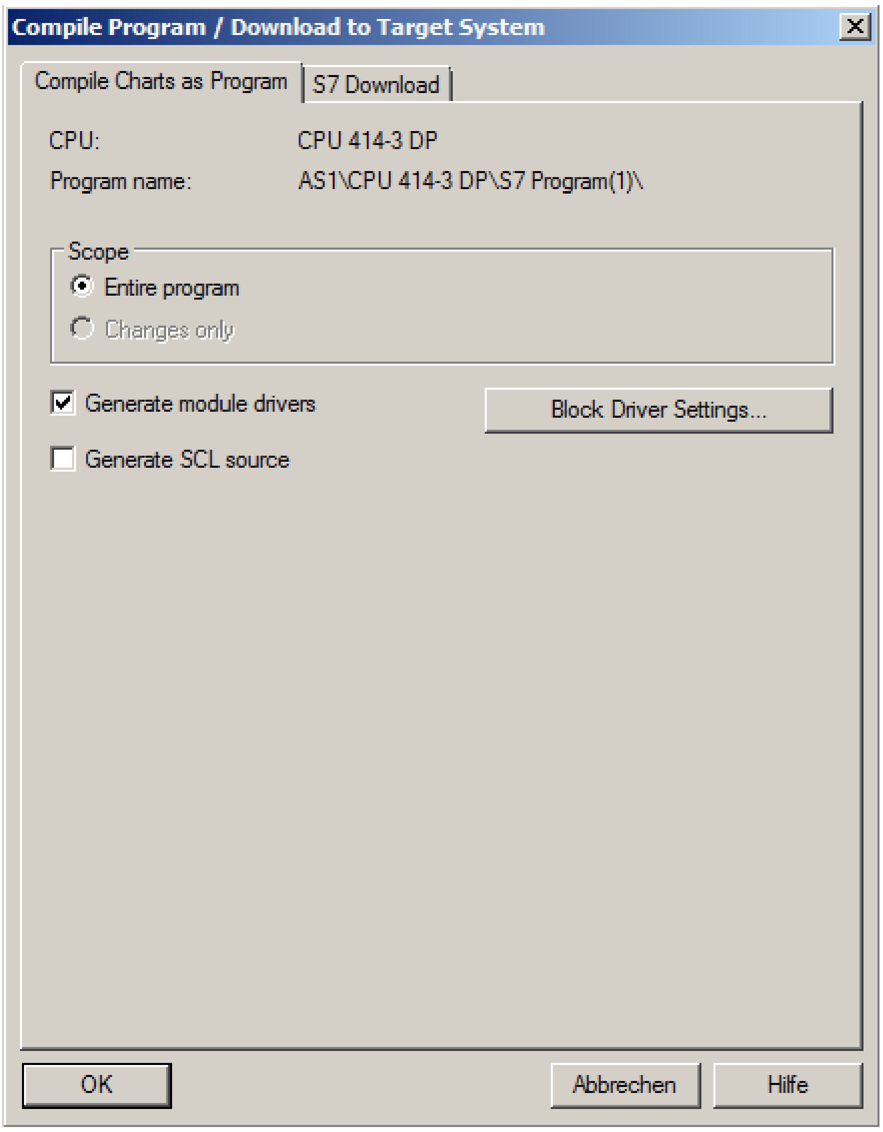
| Objects | Status | Operating mode | Compile | Download |
|---|---|---|---|---|
| SCE_PCS7_Prj | | | ☑ | ☑ |
| AS1 | | | ☑ | ☑ |
| Hardware | undefined | | ☑ | ☑ |
| CPU 414-3 DP | | | ☑ | ☑ |
| Blocks | | | | |
| Charts | undefined | | ☑ | ☑ |
| Connections | undefined | | ☑ | ☑ |
| AS2 | | | ☐ | ☐ |
| OS | | | ☐ | ☐ |

☐ Compile only ☑ Do not load if compilation error is detected

Devices connected to an enterprise network or directly to the internet must be appropriately protected against unauthorized access, e.g. by use of firewalls and network segmentation.

For more information about industrial security, please visit:
http://www.siemens.com/industrialsecurity

3. When compiling the charts, it is important to compile the entire program and to have the module drivers generated. (→ Scope: Entire program → Generate module drivers → S7 Download)


Compile Program / Download to Target System

Compile Charts as Program | S7 Download

CPU: CPU 414-3 DP

Program name: AS1\CPU 414-3 DP\S7 Program(1)\

Scope

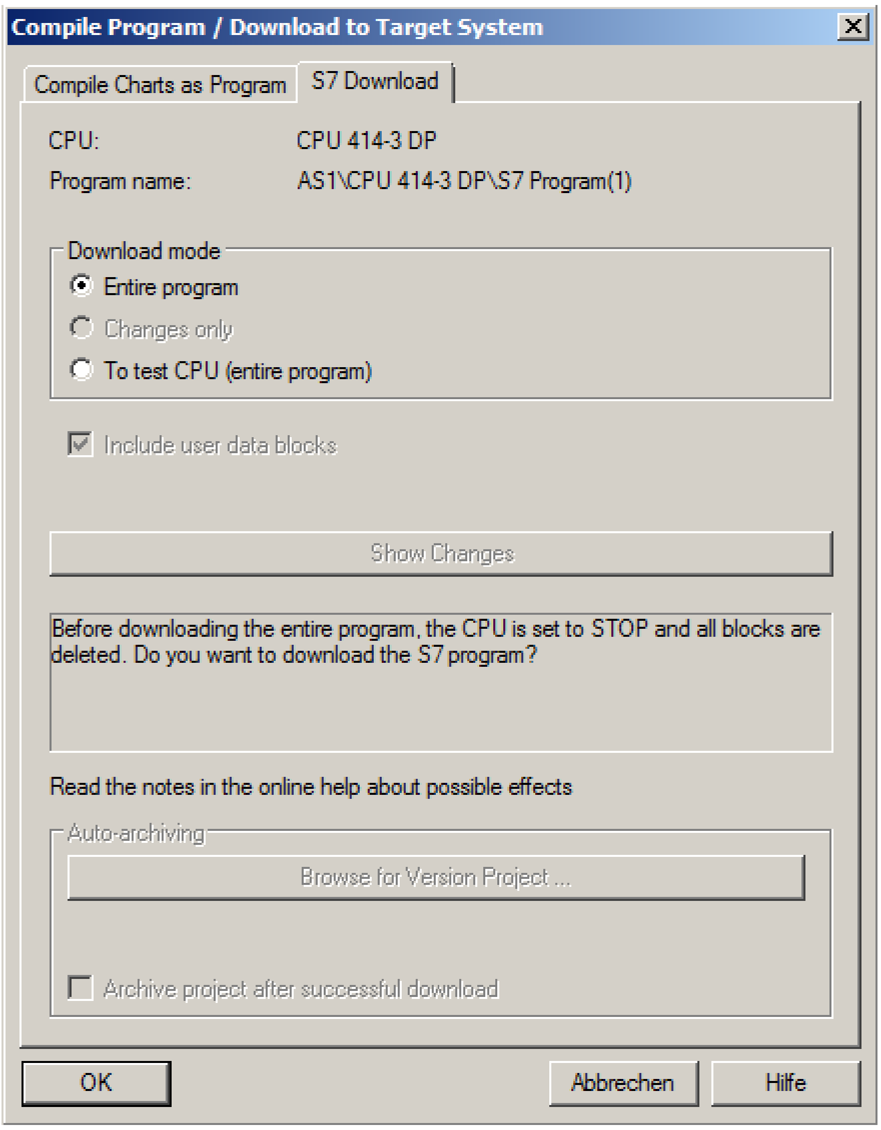
(●) Entire program

( ) Changes only

[✓] Generate module drivers

[ ] Generate SCL source

4. When downloading the charts, it is also important to download the entire program. (→ Download mode: Entire program → OK → OK )


Compile Program / Download to Target System
Compile Charts as Program
S7 Download
Show Changes
Browse for Version Project ...
OK
Abbrechen
Hilfe

CPU: CPU 414-3 DP

Program name: AS1\CPU 414-3 DP\S7 Program(1)

Download mode

[✓ Entire program]

Changes only

To test CPU (entire program)

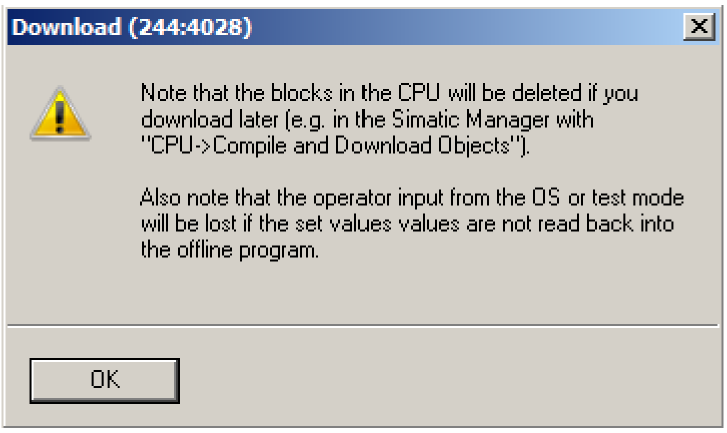
[✓ Include user data blocks]

Before downloading the entire program, the CPU is set to STOP and all blocks are deleted. Do you want to download the S7 program?

Read the notes in the online help about possible effects

Auto-archiving

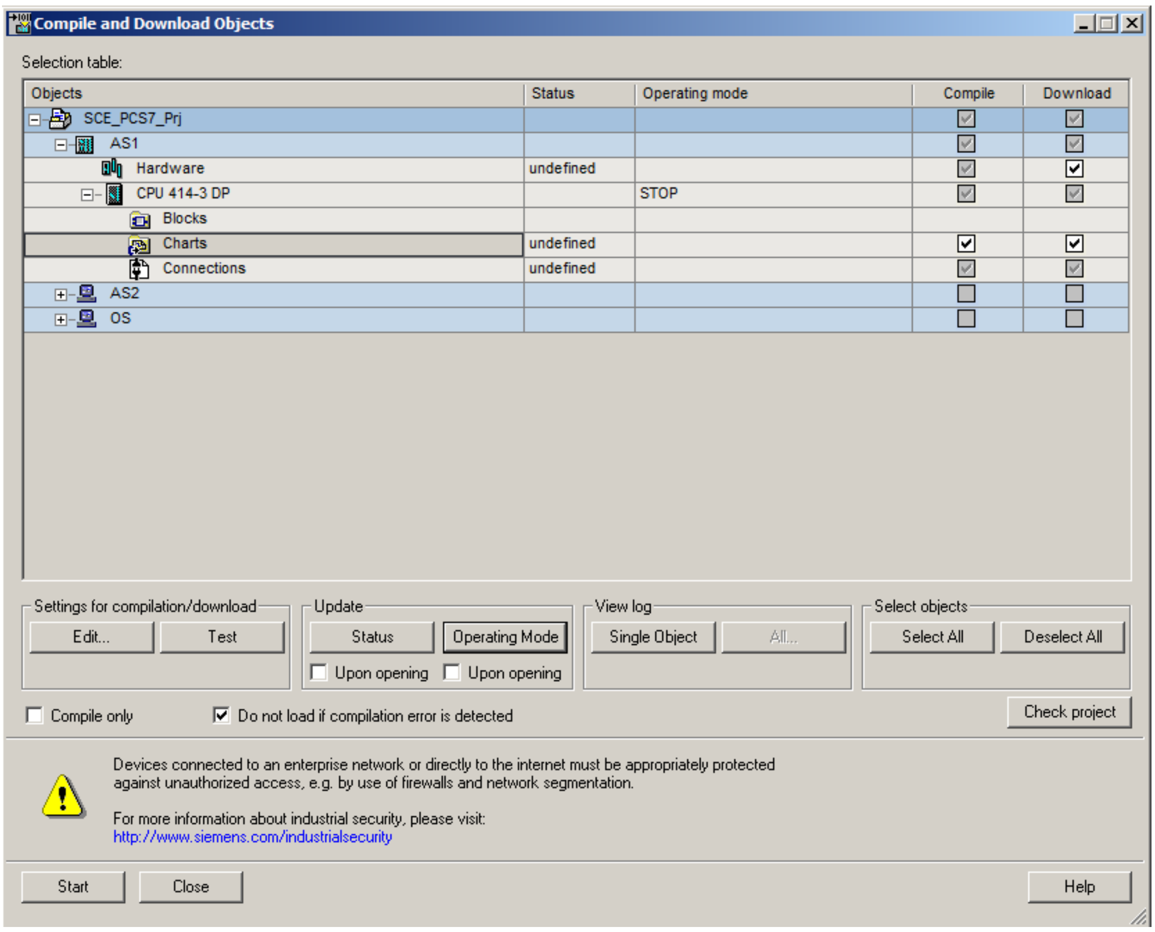
Archive project after successful download



Note that the blocks in the CPU will be deleted if you download later (e.g. in the Simatic Manager with "CPU->Compile and Download Objects").

Also note that the operator input from the OS or test mode will be lost if the set values values are not read back into the offline program.

5. Finally, 'Compile and Download' is started. The warnings and instructions regarding plant safety should be read carefully. The CPU must be switched to 'STOP' before 'Compile and download'. (→ Start → OK → Yes)



Compile and Download Objects

Selection table:

| Objects | Status | Operating mode | Compile | Download |
|---|---|---|---|---|
| SCE_PCS7_Prj | | | ✓ | ✓ |
| AS1 | | | ✓ | ✓ |
| Hardware | undefined | | ✓ | ✓ |
| CPU 414-3 DP | | STOP | ✓ | ✓ |
| Blocks | | | | |
| Charts | undefined | | ✓ | ✓ |
| Connections | undefined | | ✓ | ✓ |
| AS2 | | | ☐ | ☐ |
| OS | | | ☐ | ☐ |

Settings for compilation/download: Edit... Test

Update: Status, Operating Mode; ☐ Upon opening ☐ Upon opening

View log: Single Object, All...

Select objects: Select All, Deselect All

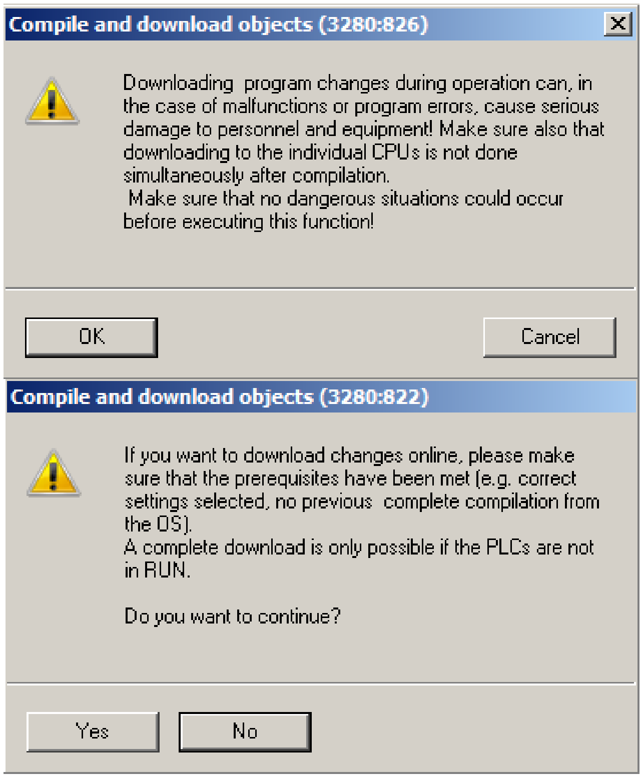
☐ Compile only ☑ Do not load if compilation error is detected

Check project

Devices connected to an enterprise network or directly to the internet must be appropriately protected against unauthorized access, e.g. by use of firewalls and network segmentation.

For more information about industrial security, please visit:
http://www.siemens.com/industrialsecurity

Start Close Help



Compile and download objects (3280:826)

Downloading program changes during operation can, in the case of malfunctions or program errors, cause serious damage to personnel and equipment! Make sure also that downloading to the individual CPUs is not done simultaneously after compilation.
Make sure that no dangerous situations could occur before executing this function!

OK Cancel

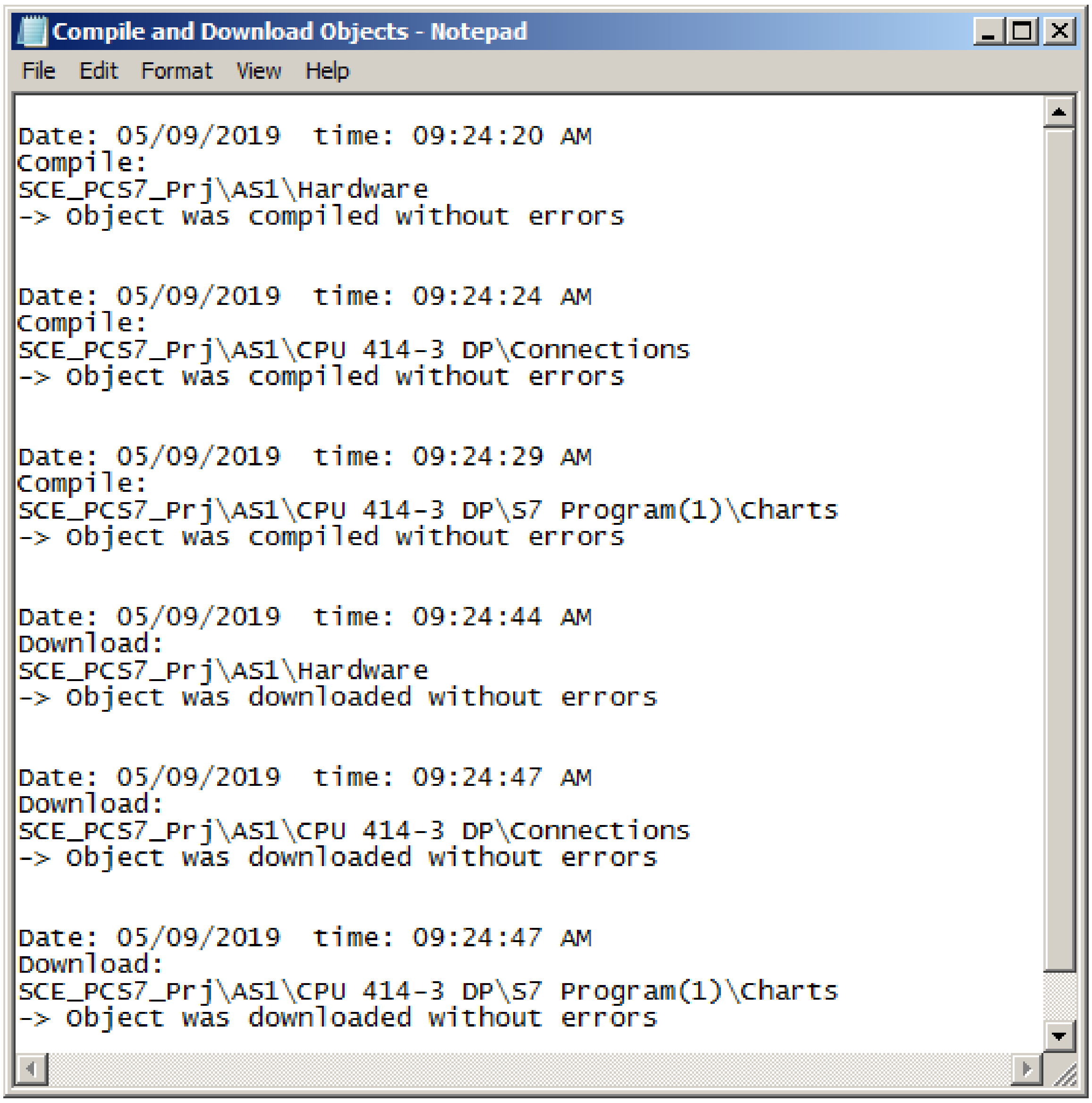
Compile and download objects (3280:822)

If you want to download changes online, please make sure that the prerequisites have been met (e.g. correct settings selected, no previous complete compilation from the OS).
A complete download is only possible if the PLCs are not in RUN.

Do you want to continue?

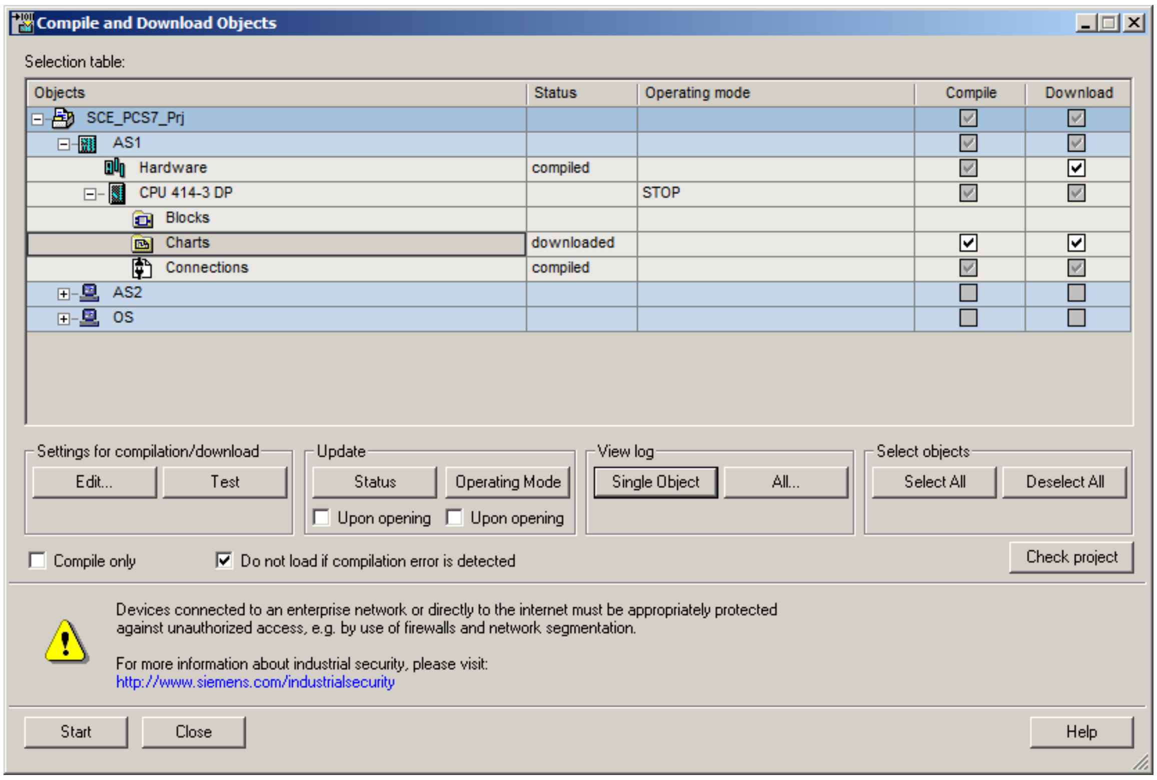
Yes No

6. At the end, errors and warnings are displayed in a log. Close the window. (→ ✕)


Compile and Download Objects - Notepad
File Edit Format View Help

```
Date: 05/09/2019  time: 09:24:20 AM
Compile:
SCE_PCS7_Prj\AS1\Hardware
-> Object was compiled without errors


Date: 05/09/2019  time: 09:24:24 AM
Compile:
SCE_PCS7_Prj\AS1\CPU 414-3 DP\Connections
-> Object was compiled without errors


Date: 05/09/2019  time: 09:24:29 AM
Compile:
SCE_PCS7_Prj\AS1\CPU 414-3 DP\S7 Program(1)\Charts
-> Object was compiled without errors


Date: 05/09/2019  time: 09:24:44 AM
Download:
SCE_PCS7_Prj\AS1\Hardware
-> Object was downloaded without errors


Date: 05/09/2019  time: 09:24:47 AM
Download:
SCE_PCS7_Prj\AS1\CPU 414-3 DP\Connections
-> Object was downloaded without errors


Date: 05/09/2019  time: 09:24:47 AM
Download:
SCE_PCS7_Prj\AS1\CPU 414-3 DP\S7 Program(1)\Charts
-> Object was downloaded without errors
```

7. If you want to view details for the log, you must click on an individual object when displaying the log. (→ Individual object → Close → Close)



Compile and Download Objects

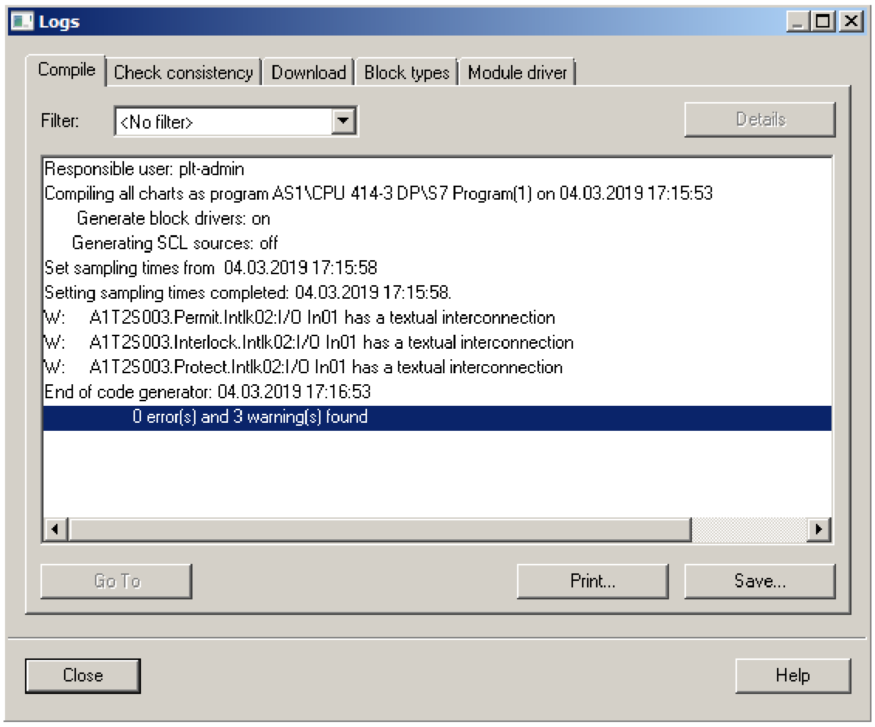
Selection table:

| Objects | Status | Operating mode | Compile | Download |
|---|---|---|---|---|
| SCE_PCS7_Prj | | | ✓ | ✓ |
| AS1 | | | ✓ | ✓ |
| Hardware | compiled | | ✓ | ✓ |
| CPU 414-3 DP | | STOP | ✓ | ✓ |
| Blocks | | | | |
| Charts | downloaded | | ✓ | ✓ |
| Connections | compiled | | ✓ | ✓ |
| AS2 | | | ☐ | ☐ |
| OS | | | ☐ | ☐ |

[✓] Do not load if compilation error is detected

Devices connected to an enterprise network or directly to the internet must be appropriately protected against unauthorized access, e.g. by use of firewalls and network segmentation.

For more information about industrial security, please visit:
http://www.siemens.com/industrialsecurity



Logs

Filter: <No filter>

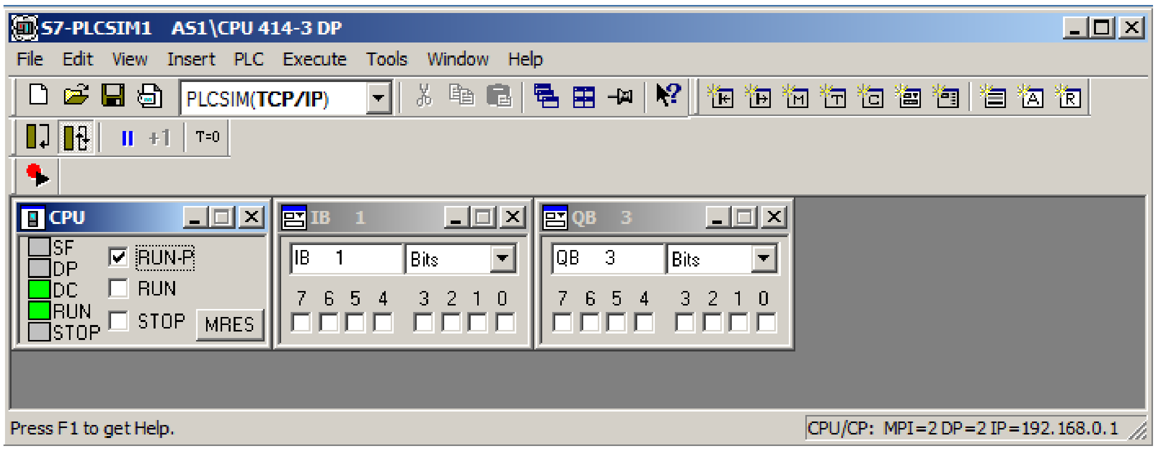
```
Responsible user: plt-admin
Compiling all charts as program AS1\CPU 414-3 DP\S7 Program(1) on 04.03.2019 17:15:53
     Generate block drivers: on
    Generating SCL sources: off
Set sampling times from  04.03.2019 17:15:58
Setting sampling times completed: 04.03.2019 17:15:58.
W:    A1T2S003.Permit.Intlk02:I/O In01 has a textual interconnection
W:    A1T2S003.Interlock.Intlk02:I/O In01 has a textual interconnection
W:    A1T2S003.Protect.Intlk02:I/O In01 has a textual interconnection
End of code generator: 04.03.2019 17:16:53
              0 error(s) and 3 warning(s) found
```

***Note:***

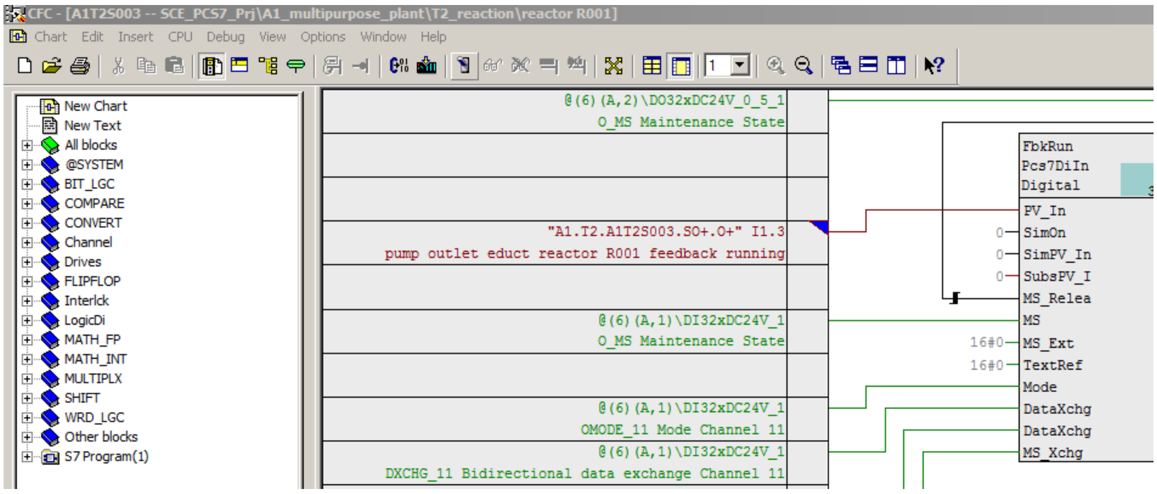
– *Four warnings appear here. Delete the empty OB1 and the presence of textual interconnections. The latter arise due to unconnected I/Os of the template. These I/Os will be connected in chapter P01-05.*

## 8.7 Testing the A1T2S003 block

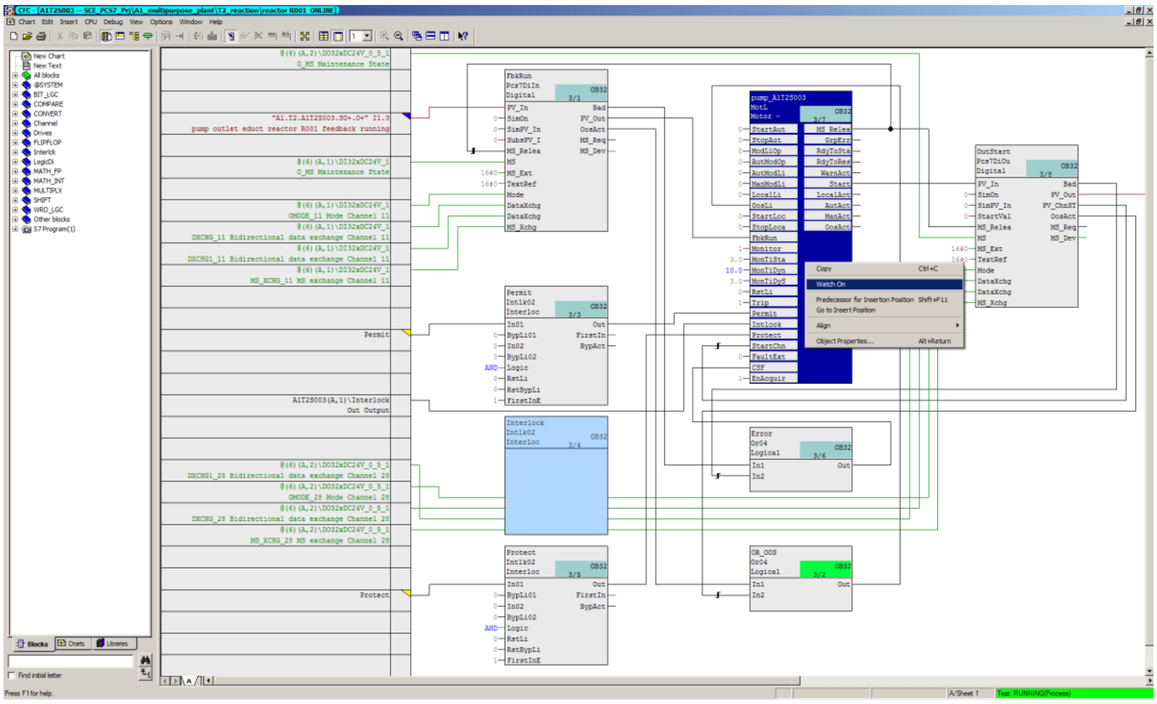
1. To test the program, the CPU in ***S7-PLCSIM*** is switched to 'RUN-P'. (→ S7-PLCSIM → RUN-P)



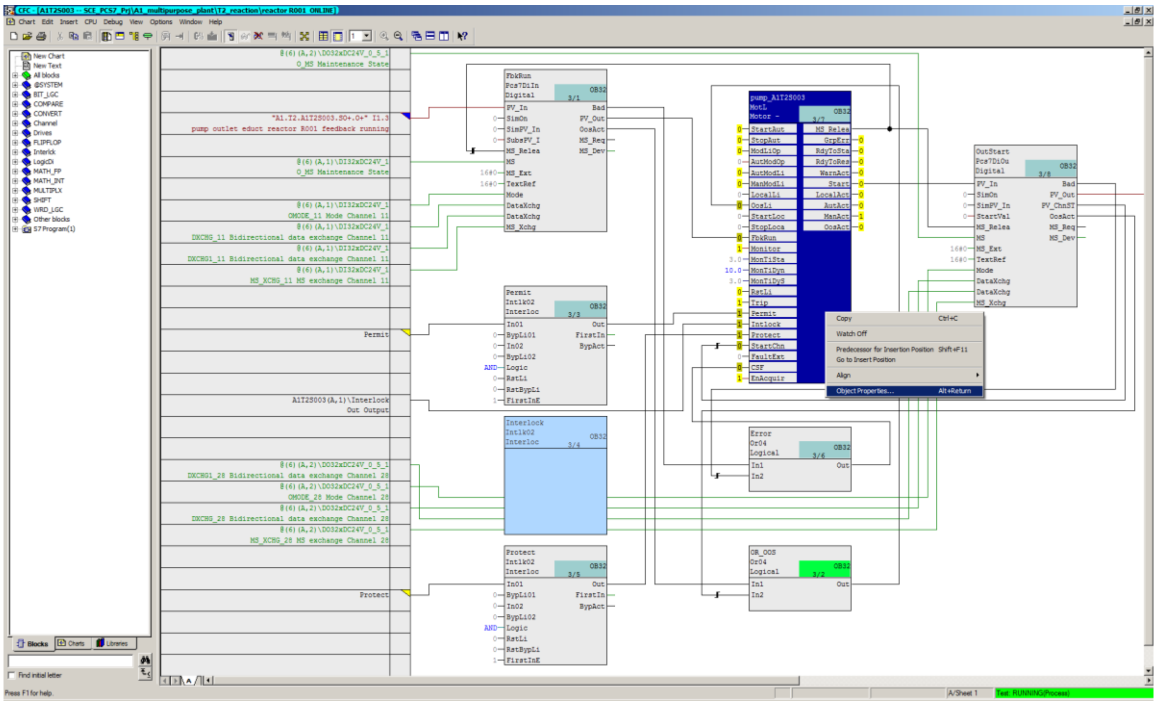
2. Before the individual blocks can be monitored in the CFC, the chart must first be switched to test mode. (→ CFC → )



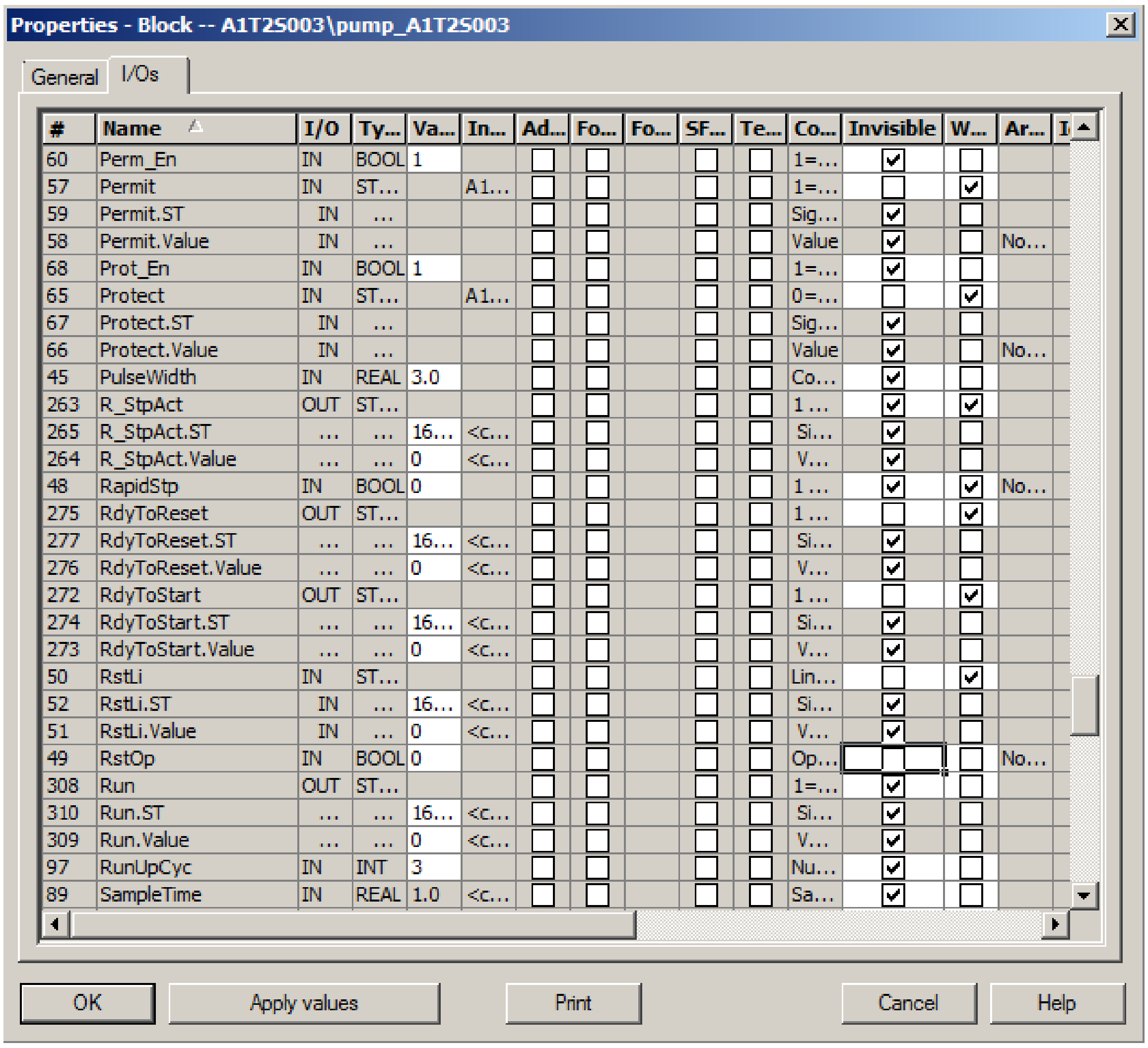
3. The blocks that are to be monitored must now be explicitly activated for monitoring. The same then applies for individual I/Os of the block. (→ pump_A1T2S003 → Watch On)


CFC - [A1T2S003 -- SCE_PCS7_Prj\A1_multipurpose_plant\T2_reaction\reactor R001 ONLINE]
Chart Edit Insert CPU Debug View Options Window Help
pump_A1T2S003
MotL
Motor -
Copy Ctrl+C
Watch On
Predecessor for Insertion Position Shift+F11
Go to Insert Position
Align
Object Properties... Alt+Return
Press F1 for help.
A/Sheet 1
Test: RUNNING(Process)

4. For the next steps, the I/Os for the automatic control 'StartAut' and 'StopAut' of the 'MotL' block must be visible. 'RstOp' and 'MonDynErr' should also be made visible in case errors occur. (→ Object Properties)

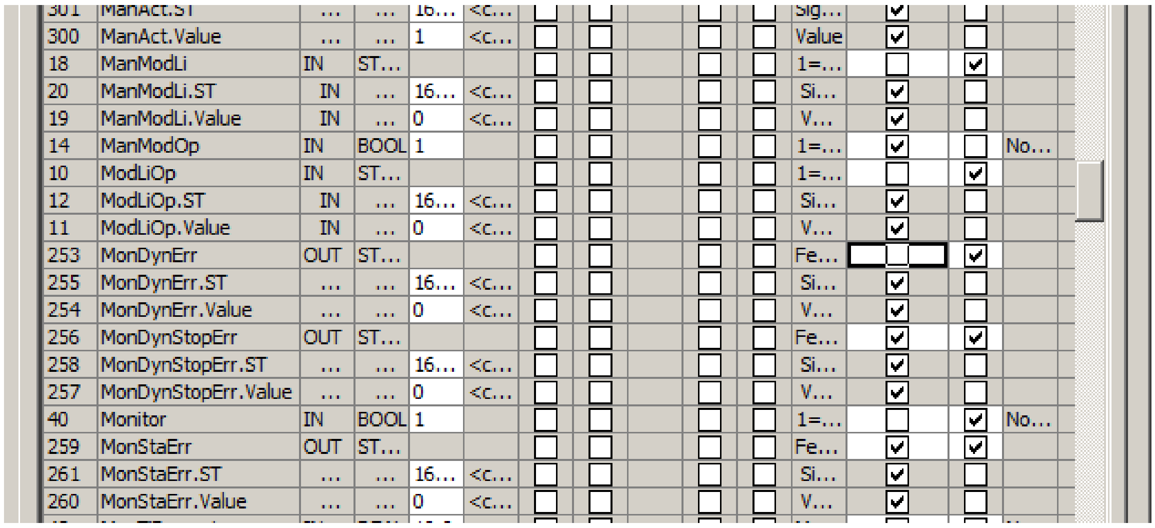

CFC - [A1T2S003 -- SCE_PCS7_Prj\A1_multipurpose_plant\T2_reaction\reactor R001 ONLINE]
Chart Edit Insert CPU Debug View Options Window Help
pump_A1T2S003
MotL
Motor -
Copy Ctrl+C
Watch Off
Predecessor for Insertion Position Shift+F11
Go to Insert Position
Align
Object Properties... Alt+Return
Press F1 for help.
A/Sheet 1
Test: RUNNING(Process)

(→ RstOp → Invisible → ☐)



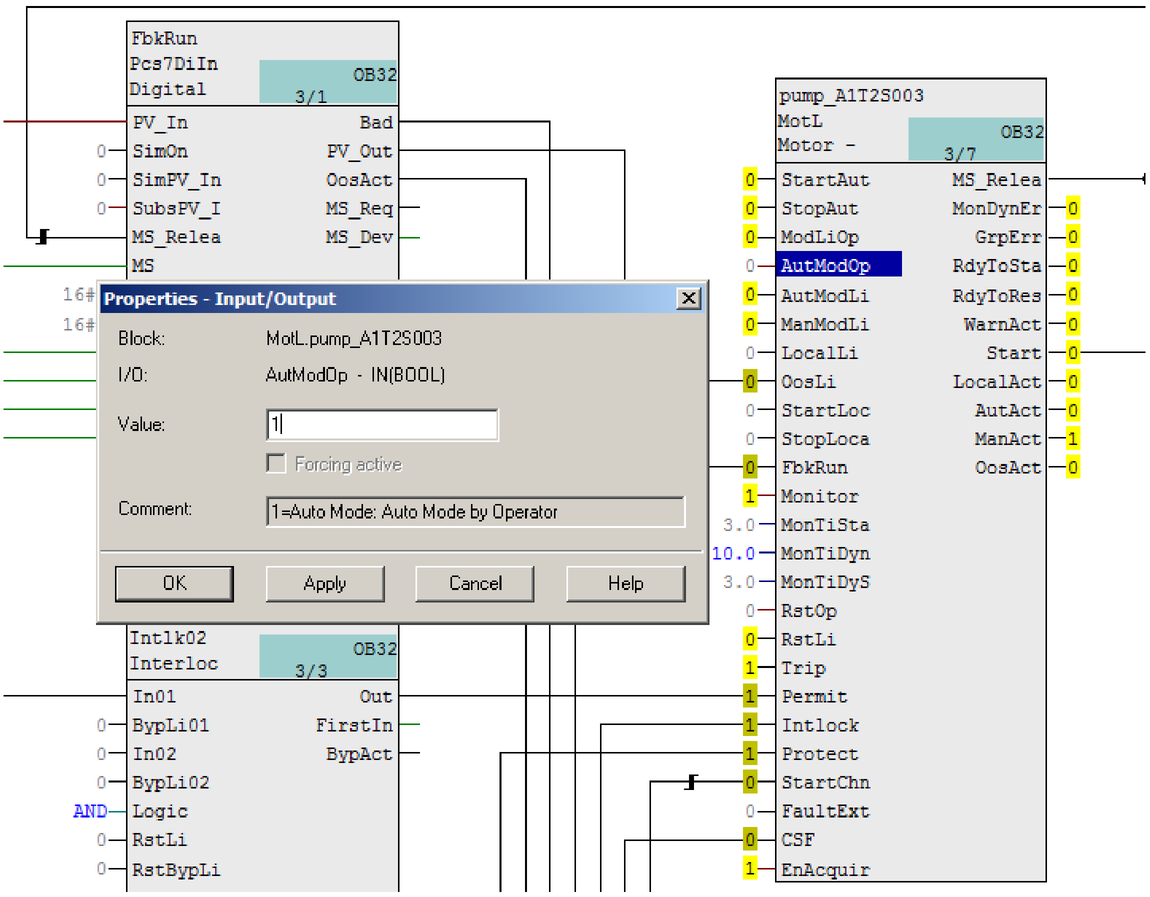
| # | Name | I/O | Ty... | Va... | In... | Ad... | Fo... | Fo... | SF... | Te... | Co... | Invisible | W... | Ar... | I |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 60 | Perm_En | IN | BOOL | 1 | | ☐ | ☐ | | ☐ | ☐ | 1=... | ☑ | ☐ | | |
| 57 | Permit | IN | ST... | | A1... | ☐ | ☐ | | ☐ | ☐ | 1=... | ☐ | ☑ | | |
| 59 | Permit.ST | IN | ... | | | ☐ | ☐ | | ☐ | ☐ | Sig... | ☑ | ☐ | | |
| 58 | Permit.Value | IN | ... | | | ☐ | ☐ | | ☐ | ☐ | Value | ☑ | ☐ | No... | |
| 68 | Prot_En | IN | BOOL | 1 | | ☐ | ☐ | | ☐ | ☐ | 1=... | ☑ | ☐ | | |
| 65 | Protect | IN | ST... | | A1... | ☐ | ☐ | | ☐ | ☐ | 0=... | ☐ | ☑ | | |
| 67 | Protect.ST | IN | ... | | | ☐ | ☐ | | ☐ | ☐ | Sig... | ☑ | ☐ | | |
| 66 | Protect.Value | IN | ... | | | ☐ | ☐ | | ☐ | ☐ | Value | ☑ | ☐ | No... | |
| 45 | PulseWidth | IN | REAL | 3.0 | | ☐ | ☐ | | ☐ | ☐ | Co... | ☑ | ☐ | | |
| 263 | R_StpAct | OUT | ST... | | | ☐ | ☐ | | ☐ | ☐ | 1 ... | ☑ | ☑ | | |
| 265 | R_StpAct.ST | ... | ... | 16... | <c... | ☐ | ☐ | | ☐ | ☐ | Si... | ☑ | ☐ | | |
| 264 | R_StpAct.Value | ... | ... | 0 | <c... | ☐ | ☐ | | ☐ | ☐ | V... | ☑ | ☐ | | |
| 48 | RapidStp | IN | BOOL | 0 | | ☐ | ☐ | | ☐ | ☐ | 1 ... | ☑ | ☑ | No... | |
| 275 | RdyToReset | OUT | ST... | | | ☐ | ☐ | | ☐ | ☐ | 1 ... | ☐ | ☑ | | |
| 277 | RdyToReset.ST | ... | ... | 16... | <c... | ☐ | ☐ | | ☐ | ☐ | Si... | ☑ | ☐ | | |
| 276 | RdyToReset.Value | ... | ... | 0 | <c... | ☐ | ☐ | | ☐ | ☐ | V... | ☑ | ☐ | | |
| 272 | RdyToStart | OUT | ST... | | | ☐ | ☐ | | ☐ | ☐ | 1 ... | ☐ | ☑ | | |
| 274 | RdyToStart.ST | ... | ... | 16... | <c... | ☐ | ☐ | | ☐ | ☐ | Si... | ☑ | ☐ | | |
| 273 | RdyToStart.Value | ... | ... | 0 | <c... | ☐ | ☐ | | ☐ | ☐ | V... | ☑ | ☐ | | |
| 50 | RstLi | IN | ST... | | | ☐ | ☐ | | ☐ | ☐ | Lin... | ☐ | ☑ | | |
| 52 | RstLi.ST | IN | ... | 16... | <c... | ☐ | ☐ | | ☐ | ☐ | Si... | ☑ | ☐ | | |
| 51 | RstLi.Value | IN | ... | 0 | <c... | ☐ | ☐ | | ☐ | ☐ | V... | ☑ | ☐ | | |
| 49 | RstOp | IN | BOOL | 0 | | ☐ | ☐ | | ☐ | ☐ | Op... | ☐ | ☐ | No... | |
| 308 | Run | OUT | ST... | | | ☐ | ☐ | | ☐ | ☐ | 1=... | ☑ | ☐ | | |
| 310 | Run.ST | ... | ... | 16... | <c... | ☐ | ☐ | | ☐ | ☐ | Si... | ☑ | ☐ | | |
| 309 | Run.Value | ... | ... | 0 | <c... | ☐ | ☐ | | ☐ | ☐ | V... | ☑ | ☐ | | |
| 97 | RunUpCyc | IN | INT | 3 | | ☐ | ☐ | | ☐ | ☐ | Nu... | ☑ | ☐ | | |
| 89 | SampleTime | IN | REAL | 1.0 | <c... | ☐ | ☐ | | ☐ | ☐ | Sa... | ☑ | ☐ | | |

(→ MonDynErr → Invisible → ☐ → OK)



| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 301 | ManAct.ST | ... | ... | 16... | <c... | ☐ | ☐ | | ☐ | ☐ | Sig... | ☑ | ☐ | | |
| 300 | ManAct.Value | ... | ... | 1 | <c... | ☐ | ☐ | | ☐ | ☐ | Value | ☑ | ☐ | | |
| 18 | ManModLi | IN | ST... | | | ☐ | ☐ | | ☐ | ☐ | 1=... | ☐ | ☑ | | |
| 20 | ManModLi.ST | IN | ... | 16... | <c... | ☐ | ☐ | | ☐ | ☐ | Si... | ☑ | ☐ | | |
| 19 | ManModLi.Value | IN | ... | 0 | <c... | ☐ | ☐ | | ☐ | ☐ | V... | ☑ | ☐ | | |
| 14 | ManModOp | IN | BOOL | 1 | | ☐ | ☐ | | ☐ | ☐ | 1=... | ☑ | ☐ | No... | |
| 10 | ModLiOp | IN | ST... | | | ☐ | ☐ | | ☐ | ☐ | 1=... | ☐ | ☑ | | |
| 12 | ModLiOp.ST | IN | ... | 16... | <c... | ☐ | ☐ | | ☐ | ☐ | Si... | ☑ | ☐ | | |
| 11 | ModLiOp.Value | IN | ... | 0 | <c... | ☐ | ☐ | | ☐ | ☐ | V... | ☑ | ☐ | | |
| 253 | MonDynErr | OUT | ST... | | | ☐ | ☐ | | ☐ | ☐ | Fe... | ☐ | ☑ | | |
| 255 | MonDynErr.ST | ... | ... | 16... | <c... | ☐ | ☐ | | ☐ | ☐ | Si... | ☑ | ☐ | | |
| 254 | MonDynErr.Value | ... | ... | 0 | <c... | ☐ | ☐ | | ☐ | ☐ | V... | ☑ | ☐ | | |
| 256 | MonDynStopErr | OUT | ST... | | | ☐ | ☐ | | ☐ | ☐ | Fe... | ☑ | ☑ | | |
| 258 | MonDynStopErr.ST | ... | ... | 16... | <c... | ☐ | ☐ | | ☐ | ☐ | Si... | ☑ | ☐ | | |
| 257 | MonDynStopErr.Value | ... | ... | 0 | <c... | ☐ | ☐ | | ☐ | ☐ | V... | ☑ | ☐ | | |
| 40 | Monitor | IN | BOOL | 1 | | ☐ | ☐ | | ☐ | ☐ | 1=... | ☐ | ☑ | No... | |
| 259 | MonStaErr | OUT | ST... | | | ☐ | ☐ | | ☐ | ☐ | Fe... | ☑ | ☑ | | |
| 261 | MonStaErr.ST | ... | ... | 16... | <c... | ☐ | ☐ | | ☐ | ☐ | Si... | ☑ | ☐ | | |
| 260 | MonStaErr.Value | ... | ... | 0 | <c... | ☐ | ☐ | | ☐ | ☐ | V... | ☑ | ☐ | | |

5. Next, the changeover for manual/automatic mode is made to automatic through 'AutModOp' == 1. (→ AutModOp → Properties → Value: 1 → OK)



Properties - Input/Output

Block: MotL.pump_A1T2S003

I/O: AutModOp - IN(BOOL)

Value: 1

Forcing active

Comment: 1=Auto Mode: Auto Mode by Operator

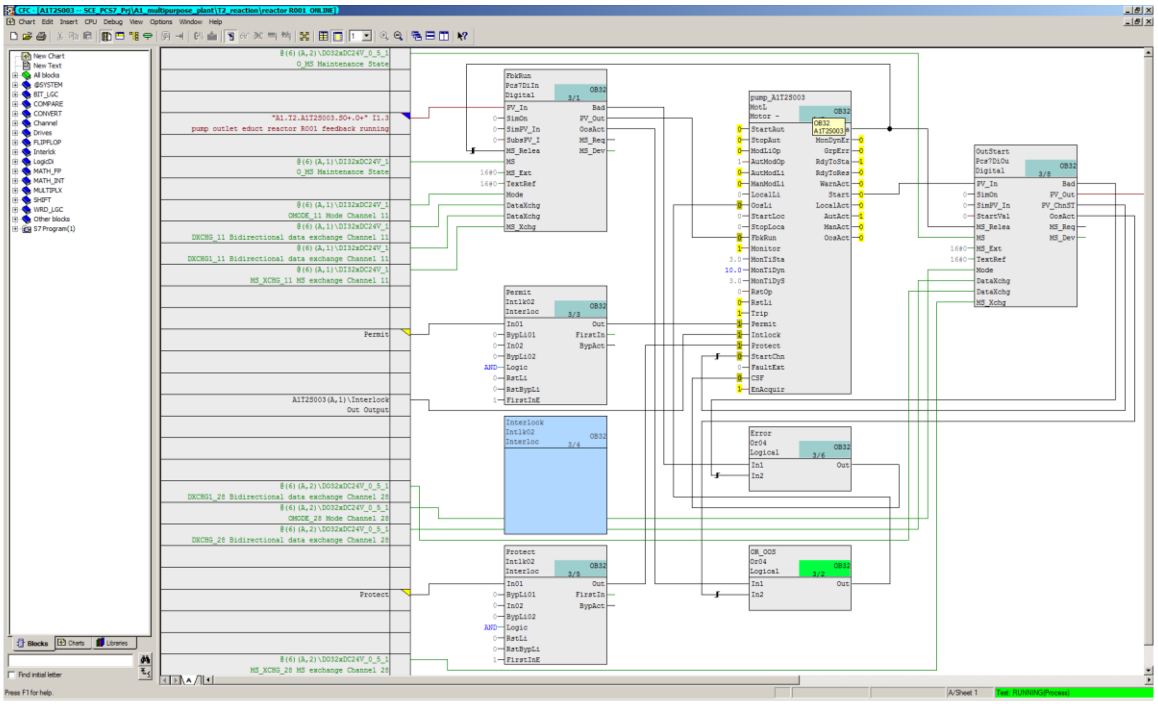
***Notes:***

– *During testing, remember that feedback I 1.3 must be set within 10 seconds after control of output Q 3.4 in* **S7-PLCSIM**. *If this is not done, the Pump_A1T2S003 block will shut down and output an error.*

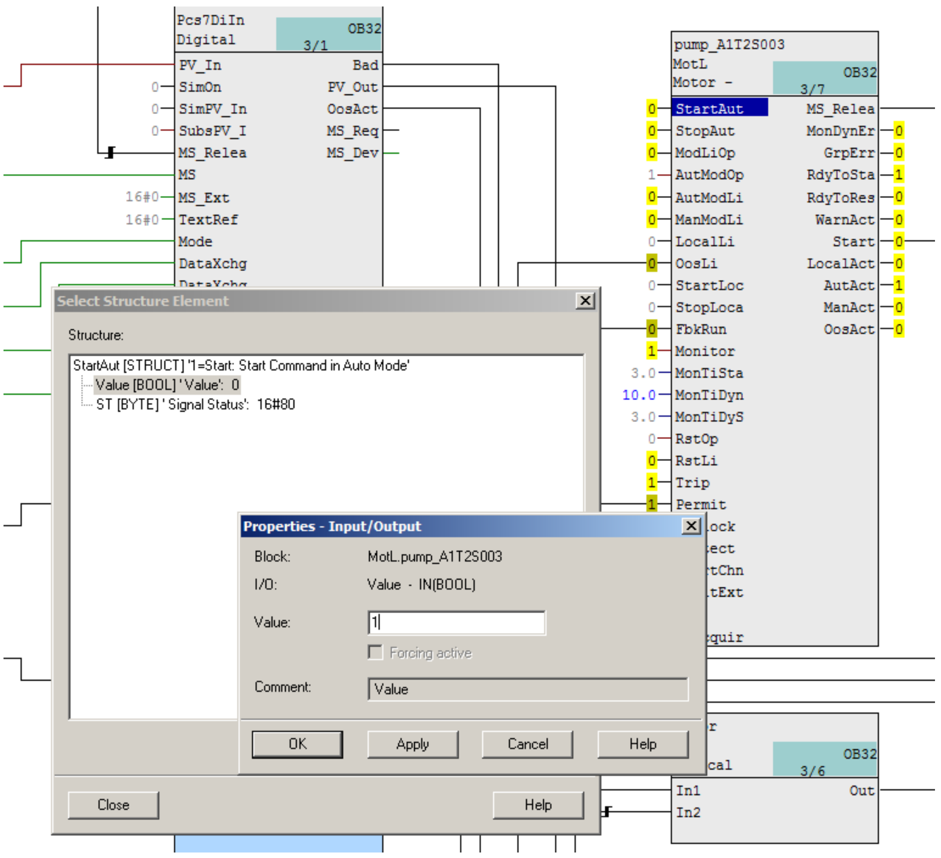
– *If there is no feedback, the pump block not only switches the actuating signal Start to 0 but indicates by setting the 'ModDynErr' output to 1 that the pump running signal of the pump had not arrived on time. The corresponding I/Os must be made visible for this. To prevent damage from repeated switch-on attempts, the pump block must be reset before another start attempt is made. To do this, the 'RstOp' input must be briefly set to 1 and then again to 0!*

*Double-clicking on this input parameter of pump block pump_A1T2S003 opens the above-displayed dialog window. First, a 1 is entered in the Value field. Then, this value is transferred to the control system by clicking the 'Apply' button, and the error outputs are set to 0. To return to the normal mode of functioning, 'RstOp' must be reset to the original state by entering 0 and clicking 'Apply' once more.*

6. In the next step, the pump is started with (StartAut == 1 and StopAut == 0) and can be stopped again with (StartAut == 0 and StopAut == 1).



(→ StartAut → Properties → Value → Value: "1")

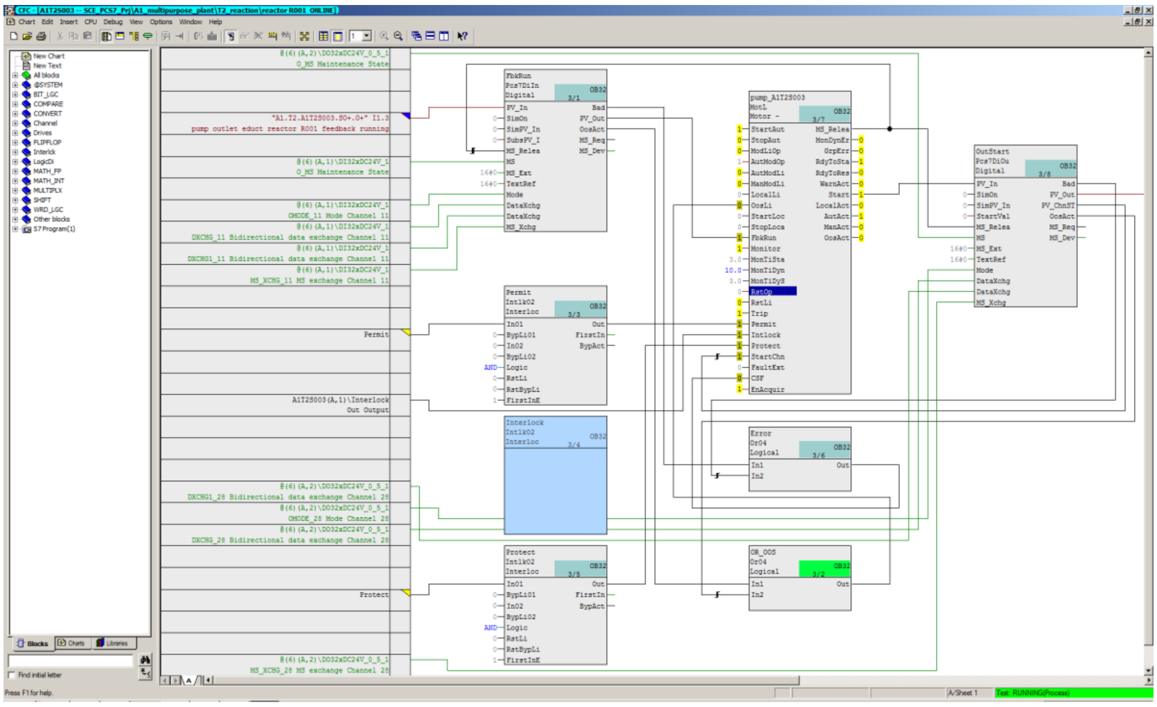


Structure:

StartAut [STRUCT] '1=Start: Start Command in Auto Mode'
- Value [BOOL] 'Value': 0
- ST [BYTE] ' Signal Status': 16#80

| | |
|---|---|
| Block: | MotL.pump_A1T2S003 |
| I/O: | Value - IN(BOOL) |
| Value: | 1 |
| | Forcing active |
| Comment: | Value |

7. If the pump running feedback was not activated in time in the PLCSIM, an error is indicated in the 'MonDynErr' I/O. This can be acknowledged with 'RstOp' == 1.



## 8.8 Checklist – step-by-step instruction

The following checklist helps students to independently check whether all steps of the step-by-step instruction have been carefully completed and enables them to successfully complete the module on their own.

| No. | Description | Checked |
|---|---|---|
| 1 | Symbol table complete | |
| 2 | Motor_Lean present and configured in process tag types (master data library) | |
| 3 | Reactor R001\A1T2S003 created as instance of Motor_Lean and configured | |
| 4 | Objects successfully compiled and downloaded | |
| 5 | Testing of A1T2S003 successful | |
| 6 | Project successfully archived | |

Table 4: Checklist for step-by-step instructions

# 9 Testing the automation logic with the simulation (optional)

The effort needed to manually input process states on the simulated control system is still reasonable when testing minor functions. For more complicated sequences with multiple dynamic process variables, the limits of what is doable are quickly reached. The use of process simulation is recommended here.

For this course, the essential interrelationships of the process to be automated have been modeled with the ***SIMIT*** simulation software. The model reproduces the dynamic behavior of the pumps, valves, tanks and reactors as well as the local operator panel with main switch, EMERGENCY OFF, switchover to local operation and the corresponding operator controls. The dynamic processes are accelerated by a factor 5 to 50 compared to reality to keep waiting times short.

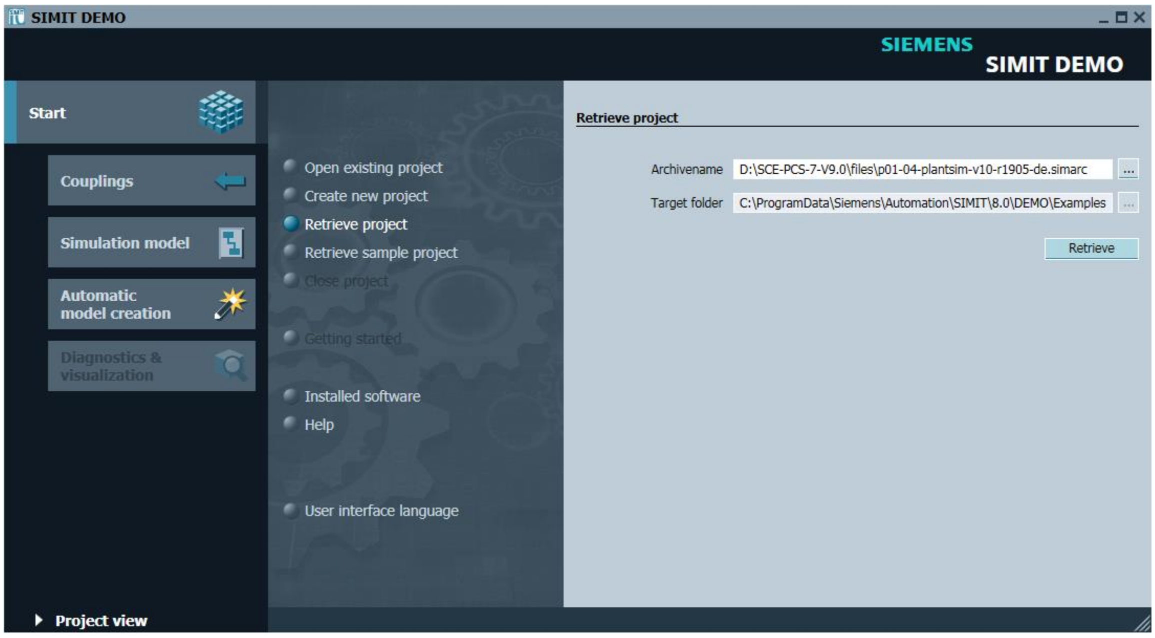
The operator interface of the simulator is shown in Figure 5. It shows the process scheme as well as the signal levels of controlled and measured variables on the left side. The ochre-shaded local operator panel is shown at the top right. Below that a series of control elements is provided for the simulation.

Using the simulator is simple – all that has to be ensured is that the assignment of the inputs and outputs has not been changed.

The pump control can now be checked very easily with the simulator:

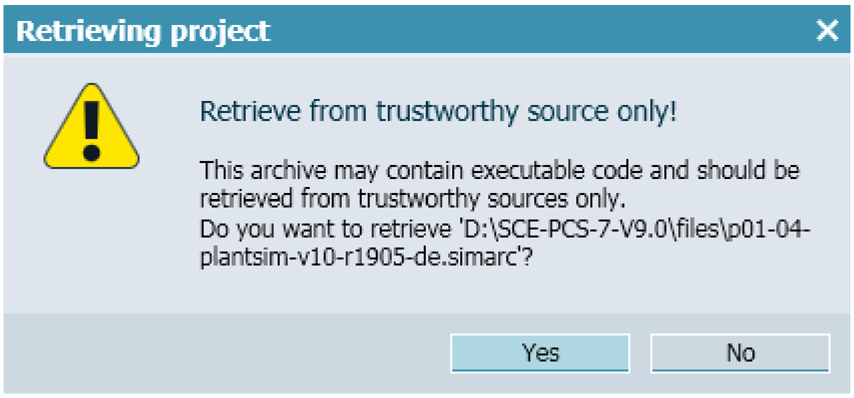
1. After S7-PLCSIM the simulation program is started.

2. You retrieve and open the simulation program, if you have not already done so. Use the supplied file 'p01-04-plantsim-v10-r1905-en.simarc' for this.



Archivename D:\SCE-PCS-7-V9.0\files\p01-04-plantsim-v10-r1905-de.simarc

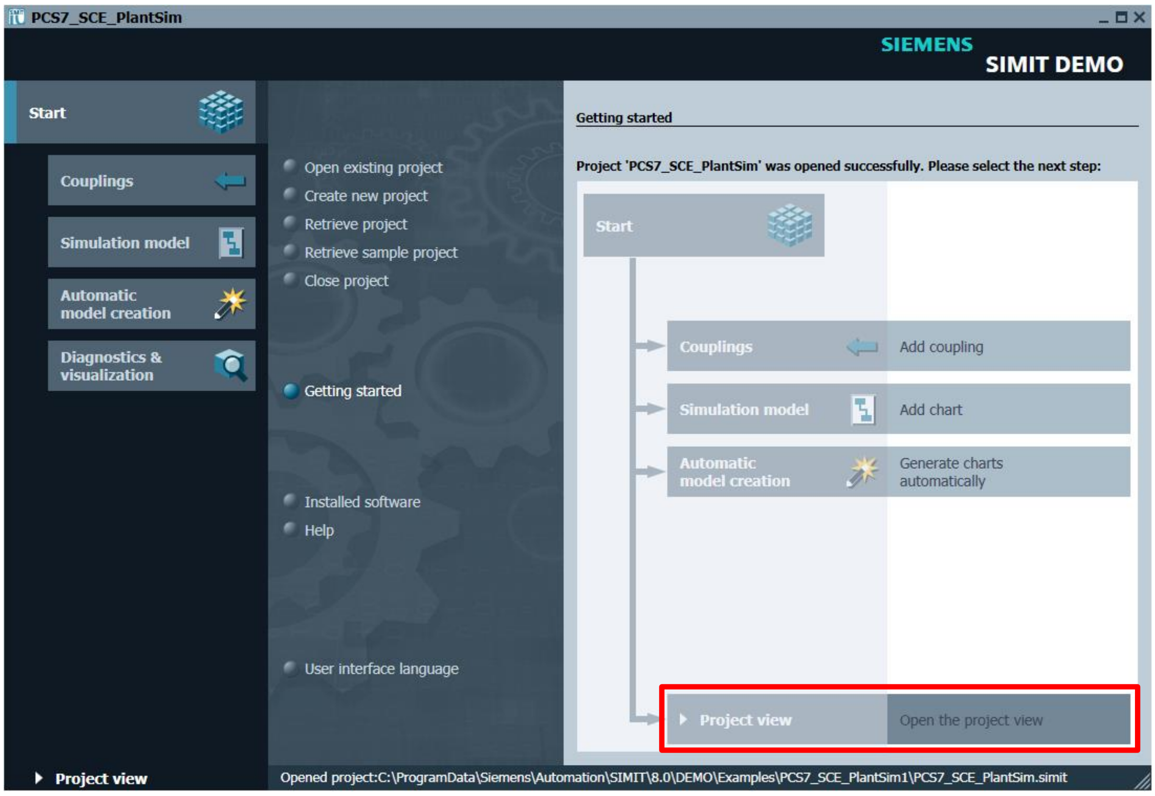
Target folder C:\ProgramData\Siemens\Automation\SIMIT\8.0\DEMO\Examples

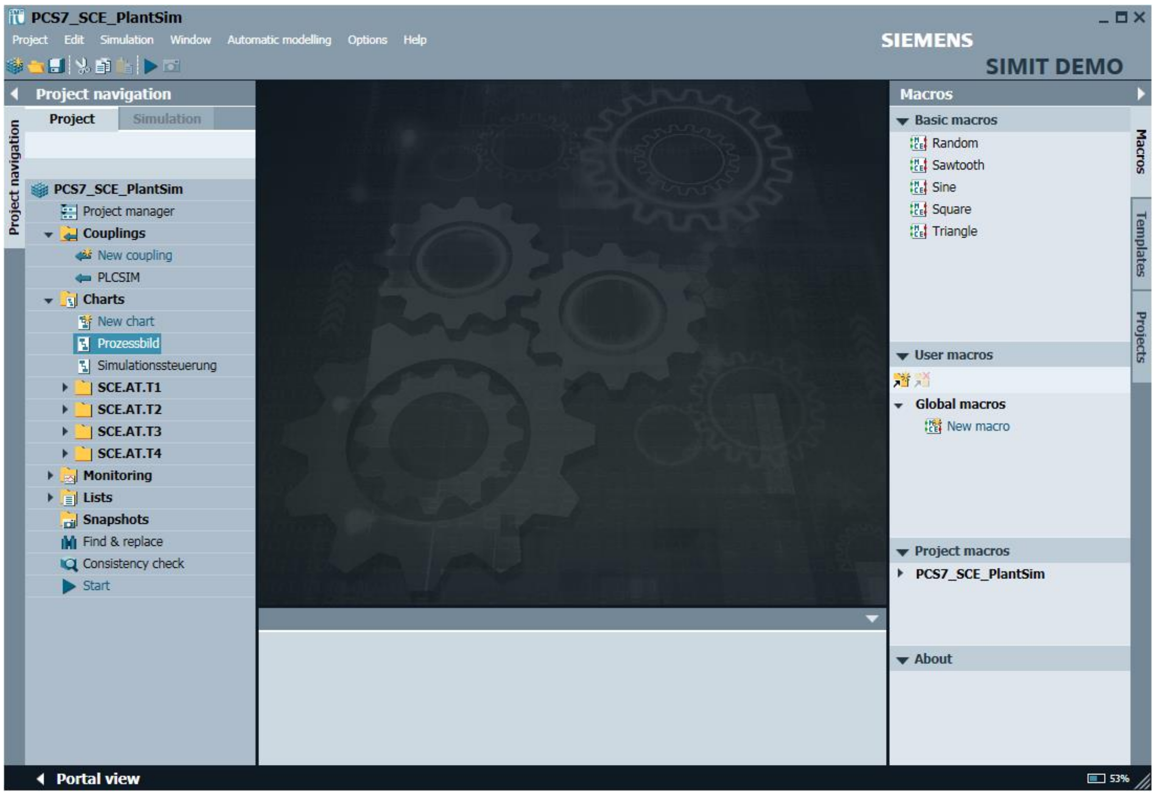


Retrieve from trustworthy source only!

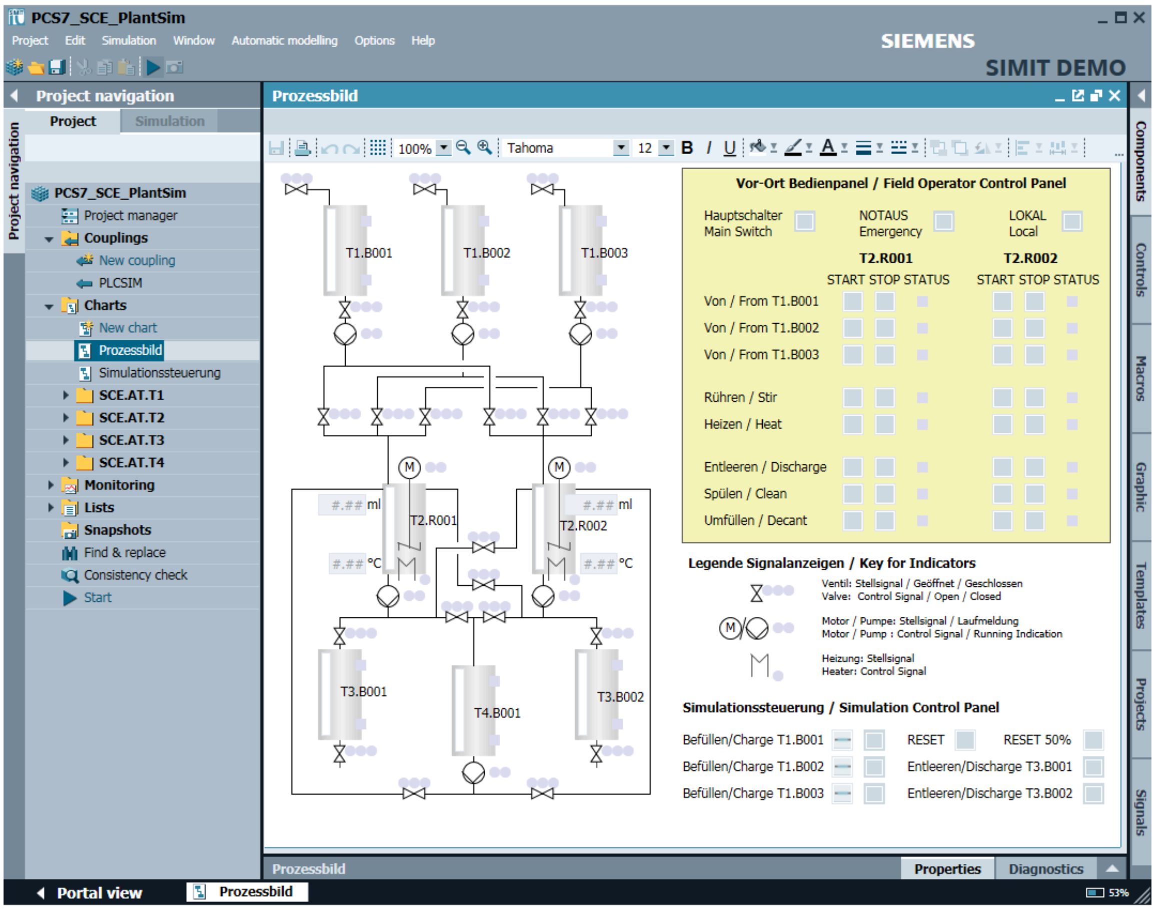
This archive may contain executable code and should be retrieved from trustworthy sources only.
Do you want to retrieve 'D:\SCE-PCS-7-V9.0\files\p01-04-plantsim-v10-r1905-de.simarc'?

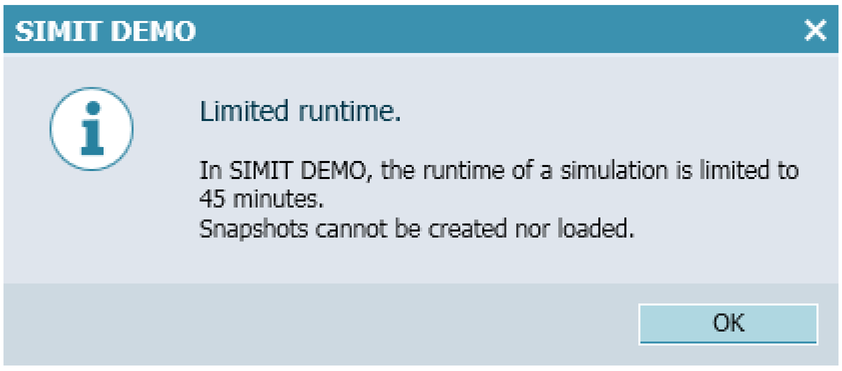
3. Change to the project view and open the process picture.





4. Start the simulation of the plant in SIMIT with the ▶ button and confirm the message regarding limited runtime with 'OK'.




SIMIT DEMO

Limited runtime.

In SIMIT DEMO, the runtime of a simulation is limited to 45 minutes.
Snapshots cannot be created nor loaded.

OK

5. You start the simulation of the PLC in S7-PLCSIM by switching over from STOP to RUN-P.

6. As shown in Figure 5, the simulation starts with educt tanks filled 75%. Everything else is empty. This state can be restored in the simulation control at any time with the 'RESET' option. Die 'RESET 50%' option fills all tanks 50%.

7. For testing, the motor block is controlled as described in the previous section 0. In the simulation, the actuating signal of the pump is illuminated green.

8. The simulated motor startup takes about 2 seconds. Then, the motor's running indicator lights up green in the simulation, and the signal level for binary input I 1.3 of the control system is set.

9. To open the delivery path, the valve for product tank T3.B001 must be opened as well. This valve will be operated using a suitable control module function in the subsequent exercise. When the pump is switched on and the valve is open, it is possible to monitor how the content of reactor T2.R001 is pumped into product tank T3.B001 in the simulation.

10. The simulation controls can be used to empty the product tanks and fill the educt tanks. When the simulation control is used for the filling, the simulation first has to be disconnected from the control system. To do so, click the button with the horizontal line (▬) once. The valve can then be opened with the button to the right of it. The actuating signal lights up green. After approximately one second, the signal of the end-position switch for the open position follows. After an additional 5 to 10 seconds, the first level changes are visible.

11. With 'RESET' and 'RESET 50%', the simulation can be returned to a defined state.

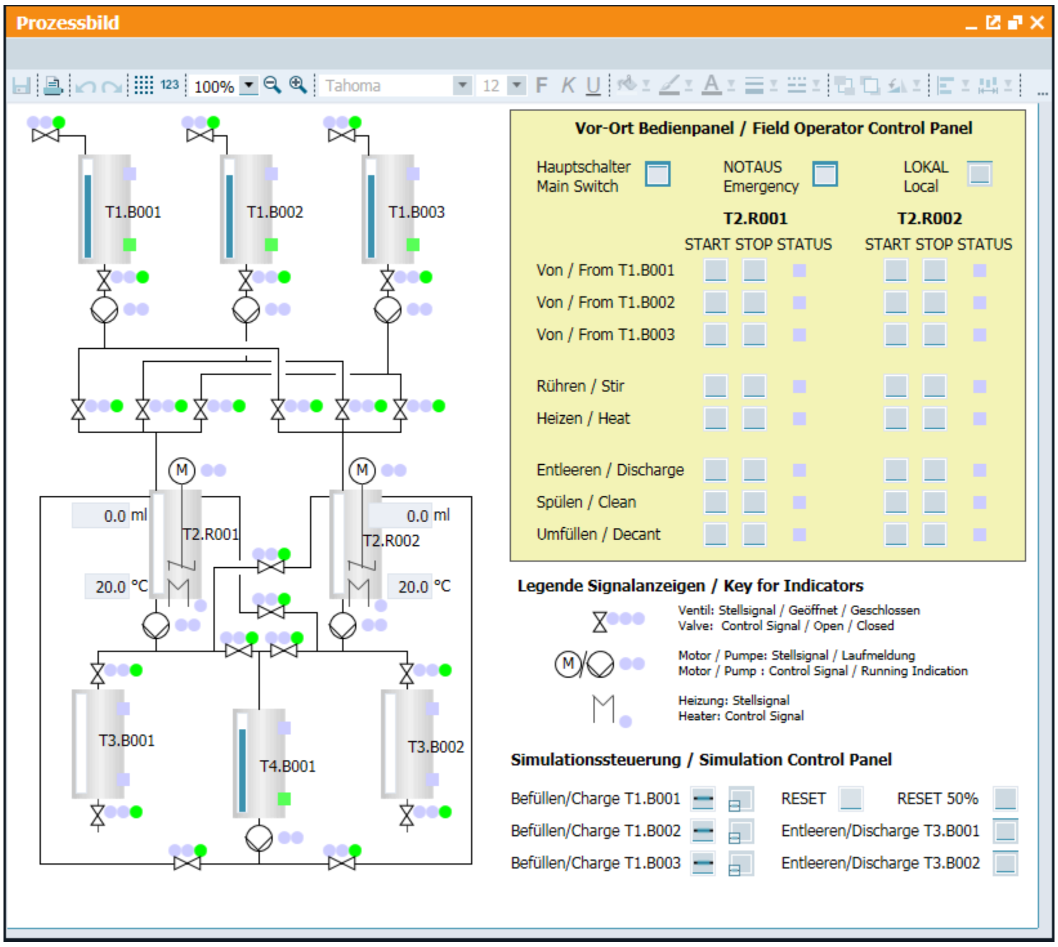


Figure 5: Operator interface of the process simulation

# 10 Exercises

In the exercises, you apply what you learned in the theory section and in the step-by-step instructions. The existing multiproject from the step-by-step instructions (p01-04-project-r1905-en.zip) is to be used and expanded for this. The download of the project is stored as zip file "Projects" on the SCE Internet for the respective module.

The objective of this exercise is to create a CFC that can be used for controlling the valves of the plant. The exercise is based on the knowledge acquired through the step-by-step instructions where a similar CFC for controlling the motor was created.

In addition, a CFC is created for scaling the level, i.e. an analog input value, from the digitalized value to the physical value. This task adds to your knowledge because the CFC is created without a template but still uses blocks from the library. This CFC will be needed for the Functional Safety chapter.

## 10.1 Tasks

The following tasks are geared to the step-by-step instructions. The corresponding steps of the instructions can be used to assist with each task.

1. Insert the 'ValveLean' template in the process tag types (similar to 'Motor_Lean'). This template is used to implement the valves.
2. An object instance of the valve template is now to be inserted in the chart folder 'product_tank B001' of unit T3_product_tanks and renamed to A1T3X001. Open the CFC and also adapt the name of the 'VlvL' block. Now, close the feedback and control signals (see Figure 7 and Table 5).
3. Download and test your implementation with the SIMIT model. The following I/Os should be visible for this: 'ModLiOp', 'AutModOp', 'ManModOp', 'OpenAut', 'CloseAut', 'MonDynErr' and 'RstOp'.
4. To integrate the analog level sensor A1T2L001 (see Figure 7), create a new CFC in chart folder 'reactor R001'. Name this CFC A1T2L001 and open it. Drag the block 'Pcs7AnIn' (FB1869) from the catalog to the CFC. To do this, select the Libraries tab in the left frame and then use either the search function at the very bottom or open PCS 7 AP Library V90/Blocks+Templates\Blocks/Channel. As soon as the block has been inserted, rename it Level_A1T2L001.
5. Now assign parameters to the block 'Pcs7AnIn' by setting the input value 'Scale' to High = 0.0 and Low = 1158.0, and input value 'PV_InUni' to 1040 (for the unit ml). Connect the 'PV_In' input of the 'Pcs7AnIn' block to the symbol for the actual level value (see Table 5) of Reactor R001.

6. Now implement the high and low level sensor of product tank B001. For this, create a CFC in the chart folder 'product_tank B001' and name it A1T3L001. Open the chart and add the 'Pcs7DiIn' block from the catalog twice (similar to exercise 5). Name one block A1T3L001_LSA+ and the other A1T3L001_LSA-. Interconnect 'PV_In' with the sensor signals in each case (see Table 4).
7. Next, create CFCs for the main switch, the EMERGENCY OFF switch and the switch for local operation. For this, one CFC each is created for A1H001, A1H002 and A1H003 like in task 6 in the chart folder 'Multipurpose plant' (see also Figure 6). The 'Pcs7DiIn' block is added, named, and interconnected with the respective address to 'PV_In'.

| Task | | Symbol | Address | Data type | Comment |
|---|---|---|---|---|---|
| | 2 | A1.T3.A1T3X001.XV.C | Q 0.6 | BOOL | Open/close valve product tank B001 control signal |
| | | A1.T3.A1T3X001.GO+-.O+ | I 67.4 | BOOL | Open/close valve product tank B001 feedback open |
| | | A1.T3.A1T3X001.GO+-.O- | I 67.5 | BOOL | Open/close valve product tank B001 feedback closed |
| | 5 | A1.T2.A1T2L001.LISA+.M | IW 72 | WORD | Actual level reactor R001 |
| | 6 | A1.T3.A1T3L001.LSA+.SA+ | I 70.6 | BOOL | Level monitoring product tank B001 operating point H |
| | | A1.T3.A1T3L001.LSA-.SA- | I 70.7 | BOOL | Level monitoring product tank B001 operating point L |
| | 7 | A1.A1H001.HS+-.START | I 0.0 | BOOL | Switch on multipurpose plant |
| | | A1.A1H002.HS+-.OFF | I 0.1 | BOOL | Activate emergency off (NO contact) |
| | | A1.A1H003.HS+-.LOC | I 0.2 | BOOL | Activate local operation |

Table 5: Symbols for implementing the valve control and level sensor

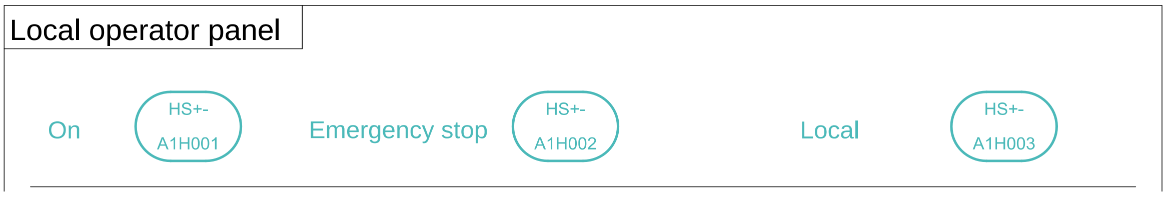


Figure 6: Excerpt from the local operator station

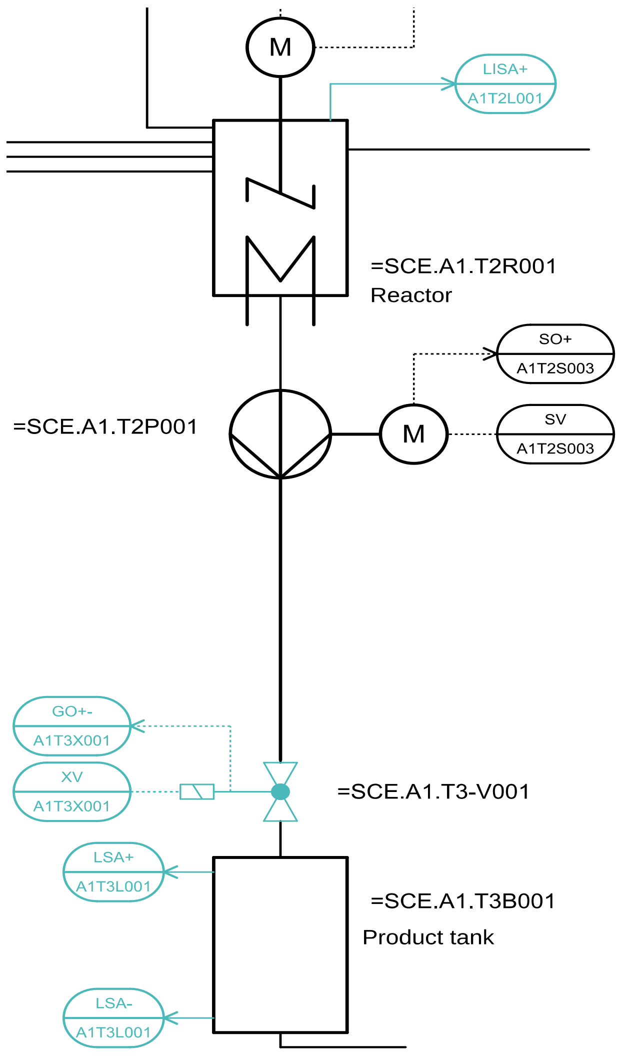


Figure 7: Excerpt from P&ID flowchart

## 10.2 Checklist – exercise

The following checklist helps students to independently check whether all steps of the exercise have been carefully completed and enables them to successfully complete the module on their own.

| No. | Description | Checked |
|---|---|---|
| 1 | Valve_Lean in process tag types | |
| 2 | Product tank B001\ A1T3X001 created, configured and tested | |
| 3 | Reactor R001\ A1T2L001 created and configured | |
| 4 | Product tank B001\A1T3L001 created and configured | |
| 5 | A1H001, A1H002 and A1H003 created and configured | |
| 6 | Project successfully archived | |

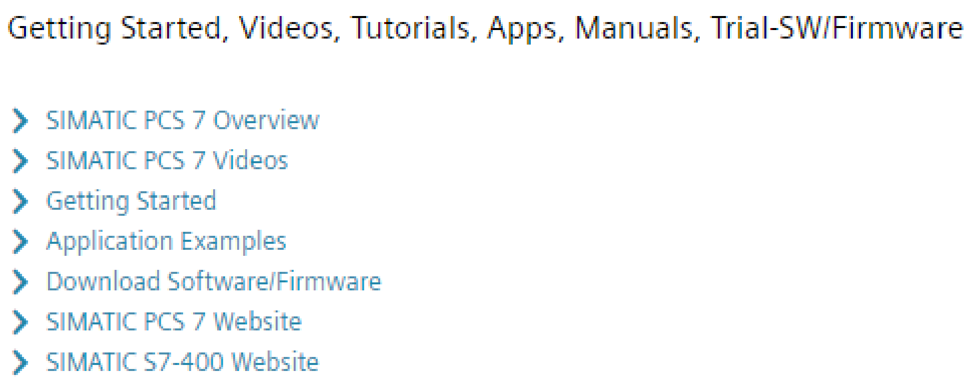
Table 6: Checklist for exercises

# 11 Additional information

More information for further practice and consolidation is available as orientation, for example: Getting Started, videos, tutorials, apps, manuals, programming guidelines and trial software/ firmware, under the following link:

siemens.com/sce/pcs7

**Preview "Additional information"**



Getting Started, Videos, Tutorials, Apps, Manuals, Trial-SW/Firmware

- SIMATIC PCS 7 Overview
- SIMATIC PCS 7 Videos
- Getting Started
- Application Examples
- Download Software/Firmware
- SIMATIC PCS 7 Website
- SIMATIC S7-400 Website

## Further Information

Siemens Automation Cooperates with Education
**siemens.com/sce**

Siemens SIMATIC PCS 7
**siemens.com/pcs7**

SCE Learn-/Training Documents
**siemens.com/sce/documents**

SCE Trainer Packages
**siemens.com/sce/tp**

SCE Contact Partners
**siemens.com/sce/contact**

Digital Enterprise
**siemens.com/digital-enterprise**

Industrie 4.0
**siemens.com/future-of-manufacturing**

Totally Integrated Automation (TIA)
**siemens.com/tia**

TIA Portal
**siemens.com/tia-portal**

SIMATIC Controller
**siemens.com/controller**

SIMATIC Technical Documentation
**siemens.com/simatic-docu**

Industry Online Support
**support.industry.siemens.com**

Product catalogue and online ordering system Industry Mall
**mall.industry.siemens.com**

Siemens
Digital Industries, FA
P.O. Box 4848
90026 Nuremberg
Germany

**siemens.com/sce**