

SIMATIC

PCS 7 Process Control System Compendium Part C - Equipment Phases with SFC Types (V9.1)

Operating Manual

<u>Security information</u>	1
<u>Preface</u>	2
<u>What's new?</u>	3
<u>Introduction</u>	4
<u>Basics</u>	5
<u>SFC type (equipment module)</u>	6
<u>State logic of equipment modules</u>	7
<u>Functionalities and solution paths</u>	8
<u>Notes, recommendations, and guidelines</u>	9
<u>Technological configuration of equipment modules and equipment phases</u>	10
<u>Service and support</u>	11

Valid for PCS 7 V9.1

Legal information

Warning notice system

This manual contains notices you have to observe in order to ensure your personal safety, as well as to prevent damage to property. The notices referring to your personal safety are highlighted in the manual by a safety alert symbol, notices referring only to property damage have no safety alert symbol. These notices shown below are graded according to the degree of danger.

DANGER

indicates that death or severe personal injury **will** result if proper precautions are not taken.

WARNING

indicates that death or severe personal injury **may** result if proper precautions are not taken.

CAUTION

indicates that minor personal injury can result if proper precautions are not taken.

NOTICE

indicates that property damage can result if proper precautions are not taken.

If more than one degree of danger is present, the warning notice representing the highest degree of danger will be used. A notice warning of injury to persons with a safety alert symbol may also include a warning relating to property damage.

Qualified Personnel

The product/system described in this documentation may be operated only by **personnel qualified** for the specific task in accordance with the relevant documentation, in particular its warning notices and safety instructions. Qualified personnel are those who, based on their training and experience, are capable of identifying risks and avoiding potential hazards when working with these products/systems.

Proper use of Siemens products

Note the following:

WARNING

Siemens products may only be used for the applications described in the catalog and in the relevant technical documentation. If products and components from other manufacturers are used, these must be recommended or approved by Siemens. Proper transport, storage, installation, assembly, commissioning, operation and maintenance are required to ensure that the products operate safely and without any problems. The permissible ambient conditions must be complied with. The information in the relevant documentation must be observed.

Trademarks

All names identified by ® are registered trademarks of Siemens AG. The remaining trademarks in this publication may be trademarks whose use by third parties for their own purposes could violate the rights of the owner.

Disclaimer of Liability

We have reviewed the contents of this publication to ensure consistency with the hardware and software described. Since variance cannot be precluded entirely, we cannot guarantee full consistency. However, the information in this publication is reviewed regularly and any necessary corrections are included in subsequent editions.

Table of contents

1	Security information	7
2	Preface	8
3	What's new?	10
4	Introduction	11
5	Basics	13
5.1	Equipment module	13
5.1.1	Separation and shared resources	13
5.1.2	Reducing the number of different types	14
5.1.3	Control strategies (CS)	15
5.1.4	Operating modes and operating states ("Modes and States")	15
5.1.5	Self-terminating EM and non-self-terminating EM	16
5.1.6	Type and instance model	17
5.2	Example: Division in the P&I diagram	17
5.3	Example: Temperature control	18
5.4	Performance specification, requirement specification, test log	18
5.5	Template for creating an SFC type	19
6	SFC type (equipment module)	20
6.1	Characteristics	21
6.1.1	Control strategies (CS)	21
6.1.2	Setpoints	21
6.1.3	Process values	22
6.1.4	Control values	22
6.1.5	Parameters	23
6.1.6	Bit memory	23
6.1.7	Timers	23
6.1.8	Note texts	24
6.1.9	Block contacts	25
6.1.10	Position texts	26
6.2	Messages	27
6.3	Sequencing logic	27

7	State logic of equipment modules	28
7.1	Starting/Run division.....	30
7.2	Completing and Completed	32
7.3	Differences between Holding and Holding (error) branch	33
7.4	Holding, Held, and Resuming	33
7.5	Error, Held (error), and Resuming (error)	33
7.6	Aborting and Stopping.....	34
8	Functionalities and solution paths	35
8.1	State change	35
8.2	Closing and resumption lockouts	36
8.3	Active control strategy change	38
8.4	Setpoint changes during operation	39
8.5	Transmitting messages	40
8.6	Using timers ("Timer").....	43
8.6.1	Example: Using "Timers"	43
8.6.2	Example: Times in Hold.....	45
8.7	Presetting control strategies.....	46
8.8	Instance-specific deselection and selection of control strategies	46
8.9	Multiplexing control modules.....	47
8.10	Control modules in Auto and Manual modes	48
8.11	Optional control modules	49
8.12	Setting position texts ("POSINO")	50
8.13	Self-terminating and non-self-terminating equipment modules	51
8.14	Returns when resuming	52
8.15	Calculations in the SFC.....	55
8.16	Start conditions for sequencers	57
8.17	"Preprocessing"/"Postprocessing" Tab.....	58

9	Notes, recommendations, and guidelines.....	59
9.1	Naming	59
9.2	Combining sequencers.....	62
9.3	Editing within the project.....	63
9.4	Block size of an SFC type	63
9.5	SFC type following CPU STOP/restart	64
9.6	Connecting to SIMATIC BATCH	67
9.7	EPH and EOP	68
9.8	Creating several instances of a type within a unit - multi-instances	69
9.9	Closing lockout for SIMATIC BATCH	69
9.10	"Continuous" functionality	69
9.11	"Extended continuous mode" functionality (SIMATIC BATCH)	71
10	Technological configuration of equipment modules and equipment phases	72
10.1	Overview of EPH and EM	74
10.1.1	EPH and EM in the SIMATIC PCS 7 environment	75
10.1.2	CFC catalog expansion	75
10.2	CMT - Status and commands	76
10.2.1	CMT - Status	77
10.2.2	CMT - Commands	79
10.3	EPH types and EPH instances	80
10.3.1	Creating EPH types.....	81
10.3.2	Technological actions, commands, states and conditional commands.....	83
10.3.3	Internal commands of the SFC	86
10.3.4	Internal states of the SFC	90
10.3.5	Special data types ANYBIT and ANYNUM	90
10.3.6	Use of CMTs.....	92
10.3.7	Further notes on SFC types in connection with EMT / EPHT	96
10.3.7.1	Using a SFC type in multiple EMTs	96
10.3.7.2	Copying an EMT in the master data library.....	96
10.3.7.3	Size of the FB and reserve	97
10.3.8	Instantiation	97
10.3.9	Naming	100
10.3.10	FB numbers of the SFC types	101
10.4	Updates.....	102
10.4.1	Example 1: Modification of a command at the CMT	103
10.4.2	Example 2: Adding blocks to the EPHT / EMT.....	111
10.4.3	Example 3: Explicit change of the "s7_contact" attribute at block I/Os of CMTs	112
10.4.4	Example 4: Implicit change of "s7_contact" attributes at block I/Os of an EMT (adding a setpoint).....	115
10.4.5	Example 5: Changing global commands (e.g. the CmdStatLib).....	116

10.5	Separation between EPH and EM	117
10.5.1	EM types	121
10.5.2	Using EMs in EPH	122
10.5.3	Commands and states for control of lower-level EMs	124
10.5.4	Workflow for control of lower-level EMs	125
10.5.5	Use of a "Shared" EM by several EPHs (MARC)	126
10.5.6	Allocation and allocation strategies	129
10.5.7	Visualization of allocations	131
10.5.8	Special case of only one EPH instance for the EM instance	137
10.5.9	Routing of values	138
10.5.10	Consequences of the assignment logic not visible in the CFC or MARC	143
10.5.11	Name length	145
10.5.12	Additional information about the number of commands and nesting depth in technological actions and technological conditions	145
10.5.13	Supplementing information to commands and status of the CmdStatLib	147
10.6	Central configuration possibility using technological list editors, TLE	149
10.7	Mass or bulk engineering of EPH and EM	151
10.8	Memory requirements	151
10.8.1	Memory requirements for the MARC FC assignment logic	152
10.8.2	Memory requirements by IF-THEN and IF-THEN_ELSE	153
10.9	EM and EPH in the SIMATIC BATCH context	157
10.9.1	Settings and configuration in SIMATIC Manager	157
10.9.2	"Equipment module with EPH" folder	159
10.9.3	"Equipment phase with EPH (derived)" folder	159
10.9.4	Naming of the RPH (SIMATIC BATCH)	162
11	Service and support	164

Security information

Security information

Siemens provides products and solutions with industrial security functions that support the secure operation of plants, solutions, machines, equipment and/or networks.

In order to protect plants, systems, machines and networks against cyber threats, it is necessary to implement – and continuously maintain – a holistic, state-of-the-art industrial security concept. Siemens' products and solutions only form one element of such a concept.

Customer is responsible to prevent unauthorized access to its plants, systems, machines and networks. Systems, machines and components should only be connected to the enterprise network or the internet if and to the extent necessary and with appropriate security measures (e.g. use of firewalls and network segmentation) in place.

Additionally, Siemens' guidance on appropriate security measures should be taken into account. For more information about industrial security, please visit <http://www.siemens.com/industrialsecurity>.

Siemens' products and solutions undergo continuous development to make them more secure. Siemens strongly recommends to apply product updates as soon as available and to always use the latest product versions. Use of product versions that are no longer supported, and failure to apply latest updates may increase customer's exposure to cyber threats.

To stay informed about product updates, subscribe to the Siemens Industrial Security RSS Feed under <http://www.siemens.com/industrialsecurity>.

Preface

Subject of the SIMATIC PCS 7 compendium

As a distinctly open system, SIMATIC PCS 7 can be flexibly adapted to a wide range of customer needs. The system software provides the configuration engineer with a great deal of freedom in terms of project configuration, as well as in the design of the program and visualization.

Experience has shown that subsequent modernization or plant expansion work is made much easier if the project is configured "in conformance with SIMATIC PCS 7" as far as possible right from the start. This means users must adhere to certain basic rules to ensure that the provided system functions will offer optimum usability in the future.

This manual serves as a compendium in addition to the product documentation for SIMATIC PCS 7. The basic tasks for creating and configuring the project are described in the form of instructions with numerous illustrations.

The compendium is divided into the following parts:

- Configuration guidelines including checklist
- Process safety including two checklists
- Equipment phases with SFC types
- Operation and maintenance including checklist
- Hardware installation including checklist
- Industrial Security

You can find the individual parts of the compendium on the PCS 7 overview page (<http://www.siemens.com/industry/online-support/pcs7>) in Industry Online Support.

Subject of Part C - Equipment Phases with SFC Types

Compendium Part C focuses on implementing equipment phases using SFC types.

The description can be used both for equipment phases in continuous processes and for SIMATIC BATCH applications.

Particular attention is paid to the following topics:

- Terms
- State logics
- Functionalities
- Solutions, recommendations
- Connecting to SIMATIC BATCH

Validity

This documentation is valid for the software packages:

- SIMATIC PCS 7 V9.1

SIMATIC PCS 7 in Industry Online Support

An overview of the most important technical information and solutions for SIMATIC PCS 7 is available at <http://www.siemens.com/industry/onlineSupport/pcs7>.

SIMATIC PCS 7 documentation

Full SIMATIC PCS 7 documentation is available free of charge and in multiple languages in PDF format at <http://www.siemens.com/pcs7-documentation>.

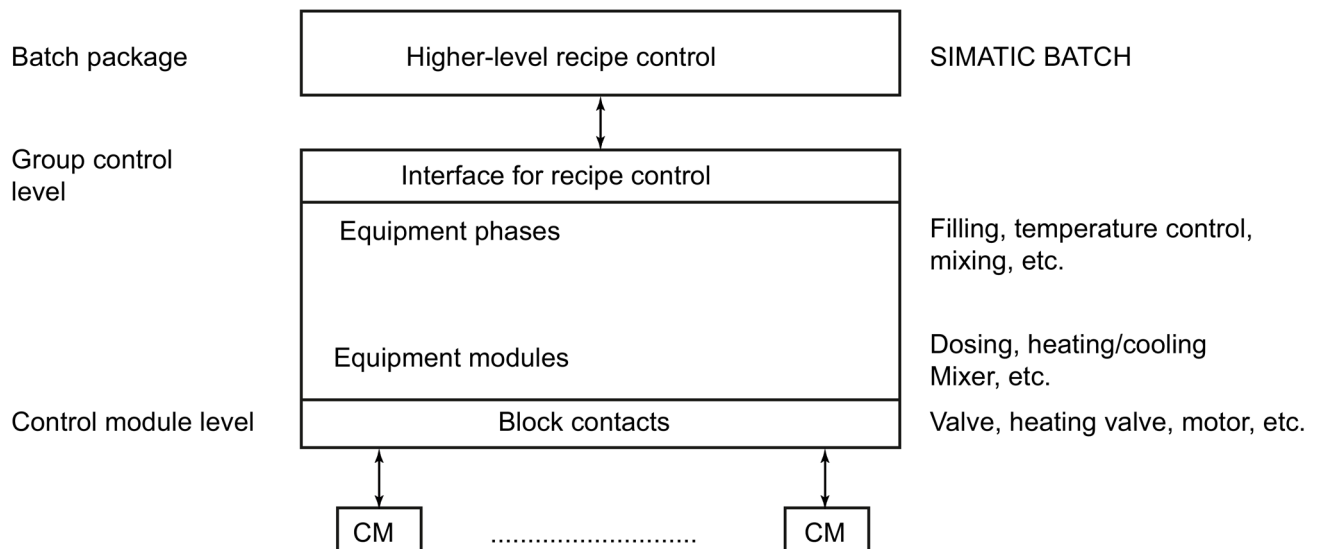
What's new?

The contents of the compendium have been updated to reflect the new functionalities and operator input options of SIMATIC PCS 7 V9.1. The compendium has also been expanded to include the topic "Technological configuration of equipment modules and equipment phases".

Introduction

Hierarchical software structures are usually created during the automation of batch processes. These structures are described in standards (NAMUR NE33, ISA S88.01).

Both higher-level recipe control and the group control level have an important role to play when it comes to hierarchical software structures, with the latter being of particular significance in terms of engineering work. The software blocks used here are known internationally (S88) as equipment modules (EM) and equipment phases (EPH), whereas in German-speaking countries (NAMUR) they are known as "Technische Einrichtungen" ("technical equipment") with "Technische Funktionen" ("technical functions") – GF = "Grundfunktionen" ("basic functions").



Content of this document

This manual describes how equipment modules/phases can be implemented in SIMATIC PCS 7 with the help of SFC types. It represents a supplement to the "SIMATIC Process Control System PCS 7 SFC for SIMATIC S7"

(<https://support.industry.siemens.com/cs/ww/en/view/109748747>) manual.

Various procedures are described at certain points throughout the documentation, although they are not exhaustive. Sometimes a preferred procedure will be described; in other cases, the selection will depend on various supplementary conditions, such as:

- Customer philosophy
- History
- Minimization of implementation work
- Minimization of system load
- Application
- Etc.

A consistent method should be selected and applied within each project.

Definition of terms

Internationally recognized terms are mainly used in this documentation. How these terms relate to German terms previously used and their sources (standards) is shown in the table below.

Term used in this document	German (source)	English (source)
Unit	Teilanlage (NE33)	Unit (NE33 English version and S88.01)
Equipment module (EM) in the physical model (plant model)	Technische Einrichtung (NE33), Grundfunktionsbaustein (GF)	Equipment module (EM) (NE33 English version and S88.01)
Equipment phase (EPH) in the procedural model	Technische Einrichtung, Technische Funktion (NE33)	Equipment phase (EPH) (S88.01)
Setpoint (SP)	Sollwert (= Führungsparameter) (allg. üblich)	Setpoint (customary)
Process value (PV)	Istwert (= Rückmeldeparameter) (allg. üblich)	Actual value, process value (customary)
Mode	Betriebsart (allg. üblich)	(Operation) mode (S88.01)
State	Betriebszustand (allg. üblich)	(Operation) state (S88.01)
Control Module (CM)	Einzelsteuereinheit (allg. üblich) Grundfunktionselement (GFE)	Control module (CM) (S88.01)
Control module	Messstelle	Control module (CM)
Equipment phase (EPH) (smallest element of a procedural control that can execute a process-oriented task.)	Technische Einrichtung, Technische Funktion (NE33)	Equipment phase (EPH) (S88.01)
P&I diagram	Rohrleitungs- und Instrumentenfließschema (R&I Schema)	Piping and Instrumentation Diagram (P&ID)

This section describes the general structure of an equipment module using examples.

5.1 Equipment module

Equipment module (EM)

An equipment module (EM) is a closed process-engineering unit. It is used to implement a task definition on the group control level and, thus, a process-engineering (sub)task.

The scope of equipment modules and the types derived from them can be freely defined. By making the right selections, you will be able to find types which can be used outside of the specific unit class, or even the specific plant, in question. On the other hand, specific process-engineering features are reflected in the equipment module design so that it is rarely possible to create general libraries which are not application-specific (unlike on a control module).

5.1.1 Separation and shared resources

In order to specify the EM in a given plant structure, we recommend that all control modules involved in implementing the same process-engineering function be grouped together in the P&I diagram (example: Heating/cooling system, dosing devices, template, ventilation system, etc.).

The separation between EMs (as components of a unit/S88: unit) and standalone units is not always obvious, as the example of a reservoir shows. You should find the following explanation useful: "As a general rule, a unit can only be involved in the production of one batch at any given time." This means that a unit should contain only one batch at a particular point in time.

If the process operation is configured such that the next batch will be started in the reservoir while the previous batch is still being processed in the main unit, the reservoir must be modeled as a separate unit.

The aim of separating the EM is to assign precisely one EM to each control module. In some cases it may be necessary for two different EMs to share one control module (CM) ("shared resources"). This is not a problem for read-only access, but if actuators are to be activated, the resolution of possible conflicts must be considered (S88: "arbitration"). In the case of split control modules or split technological blocks, for example, the APL block "ShrdResS" or "ShrdResL" can help.

5.1 Equipment module

Since PCS 7 V9.0, there is also a convenient option of splitting entire equipment modules by higher-level equipment modules or phases. You can find additional information on this in the section "Technological configuration of equipment modules and equipment phases".

Shared resources of this type should be avoided, as they generate additional planning and configuration effort and come with the risk of usage conflicts during production. In practice, however, shared resources cannot be avoided 100 % of the time.

5.1.2 Reducing the number of different types

When defining the EM, take reusability into account so that the number of different EM types can be kept as small as possible (reducing engineering work and the AS resources which must be used). This relates to both the sequencing logic and the use of the same instrumentation, as far as possible. If the plant equipment is not specified accordingly, work with "optional CMs". Optional CMs are control modules which may be available in equipment modules.

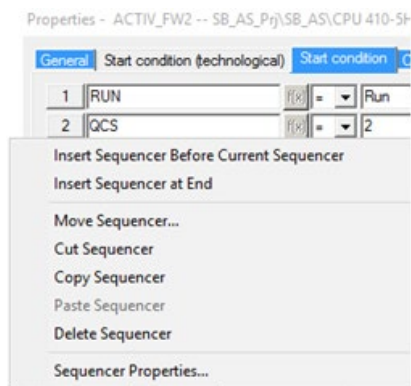
Note

You can find more information about this in the section titled Optional control modules (Page 49)

5.1.3 Control strategies (CS)

Various process-engineering sequences can be implemented using the unit instrumentation belonging to one EM. For example, acid dosing can both provide a certain amount of substance as well as set a pH value. These alternative (optional) operational approaches of an EM are referred to as "control strategies." A control strategy corresponds to a set of setpoints that differ depending on the control strategy.

Different sequencers or alternative branches and the associated setpoint sets can be activated with the selection of the control strategy. The step sequencer is controlled by the start conditions of the step sequencer (e.g. RUN = Run [TRUE]) together with the number of the control strategy (output "QCS" = xx).



The EM returns to its initial state to facilitate a change from one control strategy to another. Depending on the requirements of each specific case, it may be necessary to implement an "active control strategy change",

Note

You can find additional information on this in section Active control strategy change (Page 38)

5.1.4 Operating modes and operating states ("Modes and States")

An EM has different operating modes (S88: modes such as "manual" and "automatic") and different operating states (S88: states; e.g. "Starting", "Run", "Holding", etc.). The states are defined in an operating state logic.

You can find the description of the operating state logic in section "Diagram of the state changes for SFC OSL" of manual "SIMATIC Process Control System PCS 7 SFC for SIMATIC S7" (<https://support.industry.siemens.com/cs/ww/en/view/109748747>).

5.1.5 Self-terminating EM and non-self-terminating EM

Introduction

There are two ways of terminating an EM:

- Self-completing EM
- Non-self-completing EM

In some documentation, the term "self-terminating" may appear instead of "self-completing". Since both the COMPLETING block output and the input for configuration of the behavior are called SELFCOMP, "self-completing" is used in the following.

Both termination methods are supported by the operating state logic. The internal SFC operating state logic controls this by setting the corresponding commands.

Self-completing EM

With a self-completing EM, the SFC operating state logic itself detects that the process-engineering target has been reached and sets the "Completed" operating state.

When "Manual" mode is used, the operating state must be reset manually or the "SELFRESET" property must be activated. A typical example of this is a dosing operation.

When "Auto" mode is being used, the higher-level controller resets the EM to its initial state.

Non-self-completing EM

In the case of the non-self-completing EM, the EM is not automatically completed. Instead, it waits in "Ready to complete" state ("READY_TC") for an external "COMPLETE" command. If "Auto" mode is used, the higher-level controller (e.g. SIMATIC BATCH) detects the state and starts checking the subsequent step enabling conditions in the control recipe. If these are met, the EM receives the "COMPLETE" command. Once the "COMPLETED" state is reached, the EM is set to "IDLE" using the "RESET" command. The higher-level control then activates the next recipe step. If "Manual" mode is being used, the EM must be completed manually.

A typical example of a non-self-completing EM is a mixing procedure that is to be completed by an external event (e.g. the end of a parallel dosing operation):

- The active interim state is achieved when the target mixing speed is reached and is signaled by the status READY_TC: "Ready to complete".
- The completed dosing operation (target quantity reached) switches the mixer off.

5.1.6 Type and instance model

The type/instance model comes into effect when SFC types are used. This means:

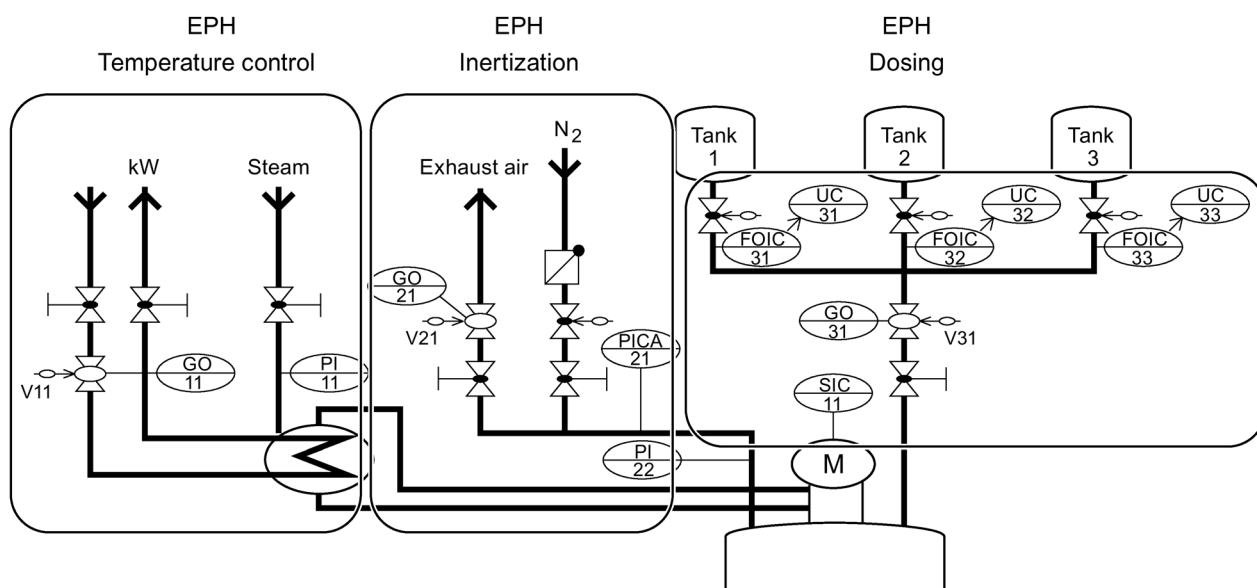
- The block structure (list of input and output parameters, including default values) and sequencing logic are defined in the SFC type. The sequencing logic is only able to access the block inputs and outputs.
- The instance-specific high and low limits can be defined at the SFC instance.
- Control strategies can also be selected at the SFC instance.
- Subsequent changes to the SFC type can be made at a central location. These changes are automatically passed on to all SFC instances.

The type/instance model brings with it significant benefits when it comes to configuring, qualifying, servicing, and maintaining the plant. However, it requires every value to be read or written to be managed via the block inputs/outputs.

5.2 Example: Division in the P&I diagram

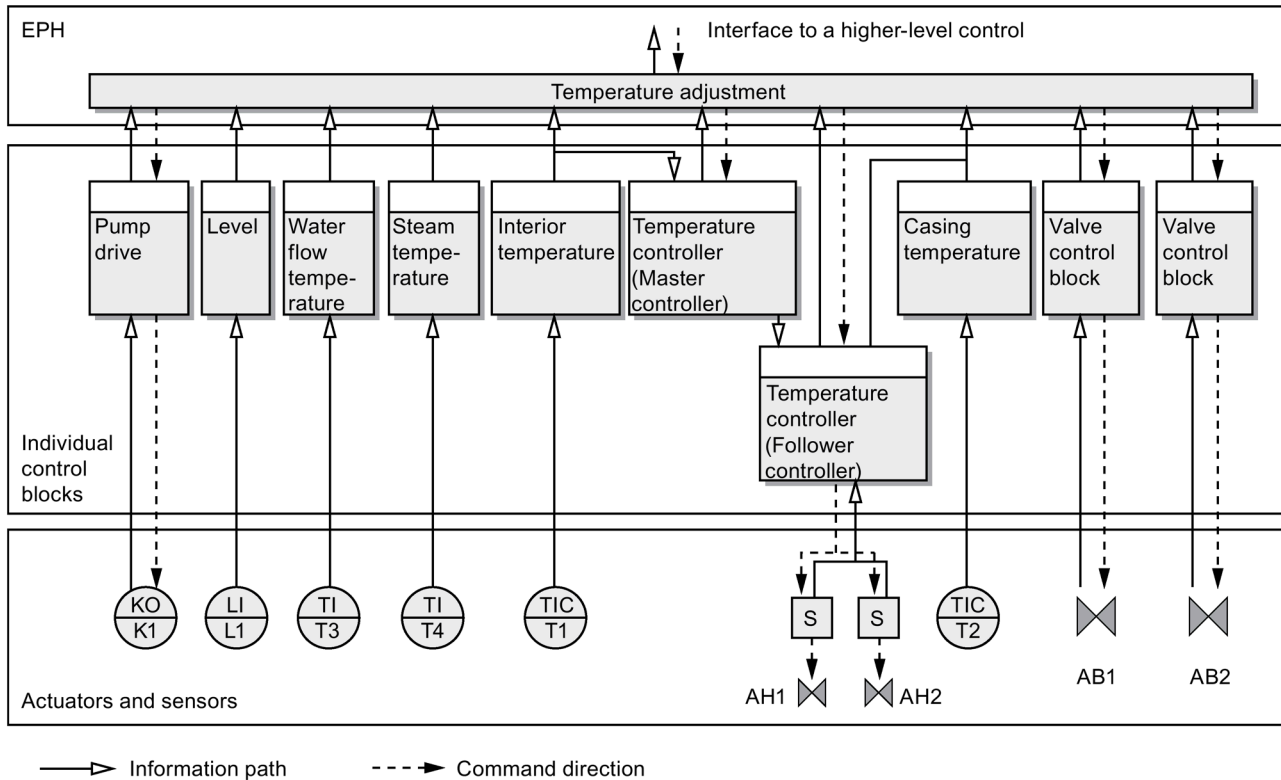
The following extract from a P&I (piping and instrumentation) diagram shows an example of how control modules (CMs) are grouped together to form an equipment phase.

The encircled control modules each belong here to an EM (tempering, inserting, dosing).



5.3 Example: Temperature control

The following figure shows "Temperature control" as an example.



5.4 Performance specification, requirement specification, test log

As a supplement to the compendium, in the document "Work Templates for the Specification of Equipment Phases with SFC Types"

(<https://support.industry.siemens.com/cs/ww/en/view/33412955>) you will find practical examples of performance specifications, requirement specifications, and test logs for the following phases:

- Temperature
Non-self-completing EM
- Discharge
Self-completing EM

5.5 Template for creating an SFC type

To help you create an equipment module you will find the following work templates in the document "Work Templates for the Specification of Equipment Phases with SFC Types" (<https://support.industry.siemens.com/cs/ww/en/view/33412955>) as a supplement to the compendium:

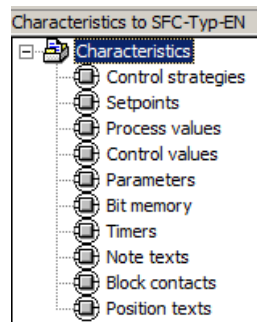
- Creating an SFC type
 - Planning template
 - This template supports the specification of your equipment phases
- Instantiation
 - This template is a project-specific template for instantiation

SFC type (equipment module)

The information provided in this section is required to implement an equipment module and must be specified at the start of EM configuration. This information is needed in order to write a requirement specification. These components are significant in terms of behavior (e.g. control strategy) on the one hand, and for the interface of the SFC types (e.g. setpoints, CMs) on the other.

The significance of these components is described in the SFC Online Help (keyword: Characteristics of the SFC type) and is looked at again here from the point of view of the "Equipment Module".

In the following example, the characteristics for an SFC type with the name SFC-Typ-DE apply.



6.1 Characteristics

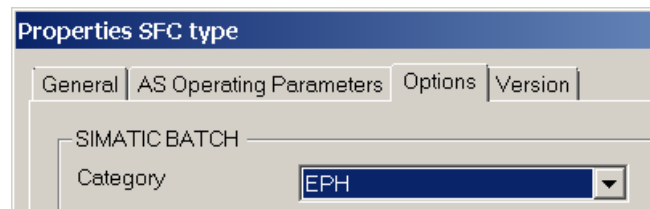
6.1.1 Control strategies (CS)

Control strategies are used for structuring of an SFC type as required for process control. You define control strategies in the characteristics editor in order to use them in configuring sequencers (for heating or cooling, for example). The control strategy can be specified by operator input or by a higher-level controller (for example, SIMATIC BATCH). Control strategies are optional.

The sequence is controlled using the normal start condition of the step sequencer (RUN=TRUE) additionally expanded to include the desired control strategy (CS). The output (QCS) must be queried in order to configure the starting condition.

The initial state (IDLE) of an EM is defined as the "Ready" state.

The control strategies of an EM are relevant in SIMATIC BATCH, for example, and are available for creating recipes. This is the case if a selection was made for the SIMATIC BATCH category in the properties of the SFC type.



6.1.2 Setpoints

Setpoints control the SFC instance or are used by it to control underlying automation functions. Setpoints can be specified by operator input or by a higher-level controller (e.g. SIMATIC BATCH). Not all setpoints are required in every control strategy and may even be misleading or disturbing. Therefore, the setpoints can be specifically assigned to each control strategy.

Characteristics to SFC-Type-EN		Contents Of: 'Characteristics\Setpoints'			
		Name	Display name	Data type	I/O name
[-] Characteristics		SP_Bool	SP_Bool	BOOL	SP_Bool
[-] Control strategies		SP_INT	SP_INT	INT	SP_INT
[-] Setpoints		SP_Real	SP_Real	REAL	SP_Real
[-] Process values		SP_String	SP_String	STRING	SP_String
[-] Control values		SP_PI	SP_PI	PI	SP_PI
[-] Parameters		SP_PO	SP_PO	PO	SP_PO
[-] Bit memory		SP_Mix_Q	SP_Mix_Q	REAL	SP_Mix_Q
[-] Timers					

6.1 Characteristics

When a setpoint is defined, an input is automatically created for the associated actual value.

The setpoints of a (recipe aware) equipment module are BATCH-relevant and are referred to as parameters on the higher-level SIMATIC BATCH control (not be confused with the parameters of the SFC type).

There are special "PI" and "PO" data types for setpoints, which represent a REAL setpoint and are supplemented by the additional attributes "Material" and "Tracking ID". Data types "PI" and "PO" are required, for example, if setpoints with reference to the material are used.

The "SOURCE", "DEST", "VIA" and "TKEY" data types are mainly used for Route Control in connection with SIMATIC BATCH. From the point of view of the SFC, they are integer values, but from the point of view of BATCH they are interpreted for equipment properties of the "Location" type. Corresponding enumerations can be defined and assigned for visualization within the SFC.

Note

The "Tracking ID" setpoint for SFC types are not used in SIMATIC BATCH.

6.1.3 Process values

Process values control the SFC instance based on process signals (for example, the value for the level or the temperature).

Process values are primarily used for step enabling conditions in step sequencers.

Characteristics to SFC-Type-EN		Contents Of: 'Characteristics\Process values'		
		Name	Data type	I/O name
[-] Characteristics				
[-] Control strategies				
[-] Setpoints				
[-] Process values		PV_Mix	REAL	PV_Mix

6.1.4 Control values

Control values are used by the SFC instance for controlling an external logic.

Characteristics to SFC-Type-EN		Contents Of: 'Characteristics\Control values'			
		Name	Data type	I/O name	Comment
[-] Characteristics					
[-] Control strategies					
[-] Setpoints					
[-] Process values					
[-] Control values		QV_rel_manual	BOOL	QV_rel_manual	
		QV_EN_SelfComplete	BOOL	QV_EN_SelfComplete	0= NSC - 1=SC

6.1.5 Parameters

Parameters modify the behavior of the SFC instance. Among other things, parameters are used for selecting alternative branches or for configuring start conditions.

Characteristics to SFC-Typ-EN		Contents Of: 'Characteristics\Parameters'			
 Characteristics		Name	Display name	Data type	I/O name
 Control strategies		 IN_Param1	IN_Param1	BOOL	IN_Param1
 Setpoints		 IN_V01_available	IN_V01_available	BOOL	IN_V01_available
 Process values		 IN_M01_available	IN_M01_available	BOOL	IN_M01_available
 Control values					
 Parameters					

6.1.6 Bit memory

Bit memory is local data memory that is assigned to the SFC instance. Bit memory is used for temporary storage of values and is created as static tags, which enables the data to be retained. The bit memory is not visible in the interface display in the CFC.

Characteristics to SFC-Typ-EN		Contents Of: 'Characteristics\Bit memory'														
 Characteristics		<table><thead><tr><th>Name</th><th>Data type</th><th>I/O name</th></tr></thead><tbody><tr><td> FL_Flag</td><td>REAL</td><td>FL_Flag</td></tr><tr><td> FL_Time</td><td>REAL</td><td>FL_Time</td></tr><tr><td></td><td></td><td></td></tr></tbody></table>	Name	Data type	I/O name	 FL_Flag	REAL	FL_Flag	 FL_Time	REAL	FL_Time					
Name	Data type	I/O name														
 FL_Flag	REAL	FL_Flag														
 FL_Time	REAL	FL_Time														
																
 Control strategies																
 Setpoints																
 Process values																
 Control values																
 Parameters																
 Bit memory																

6.1.7 Timers

Timers are often needed for implementing equipment modules, e.g. a monitoring time or the runtime for a mixer. Timers can be implemented using a standard timer block (TIMER_P), which supports different modes. Possible modes for the timer block:

- Pulse
- Extended pulse
- ON delay
- ON delay with memory
- OFF delay

6.1 Characteristics

When the SFC type is used, the timer block is automatically embedded for processing times.

Characteristics to SFC-Type-EN		Contents Of: 'Characteristics\Timers'			
		Name	I/O name	Comment	Initial value
[-] Characteristics					
[-] Control strategies					
[-] Setpoints					
[-] Process values					
[-] Control values					
[-] Parameters					
[-] Bit memory					
[-] Timers		TI_Mixer	TI_Mixer	Time Mixer	0.0

The "Timers" characteristic is implemented with the pulse-generating timer block "TIMER_P" (object name: FB 5).

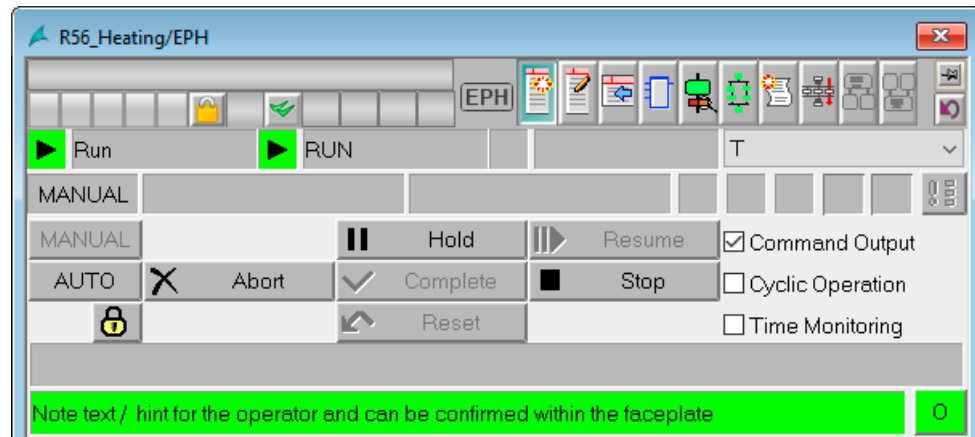
6.1.8 Note texts

Note texts are used for displaying additional notes on the operator station (OS). They can also be used to display additional information in tandem with a message in the event of an error. Configuration work needs to be carried out in order to utilize this function.

The texts, which are predefined in the characteristics dialog, can be displayed simply by setting an output (OPTIPNO) on the interface.

Characteristics to Heating		Contents Of: 'Characteristics\Note texts'		
		Name	Number	Displayed text
[-] Characteristics				
[-] Control strategies		ConfAcquire	1	Please confirm to acquire the lower level EM
[-] Setpoints		ConfRelease	2	Please confirm to release the lower level EM
[-] Process values		NoteExplain	3	Note text / hint for the operator and can be confirmed within the faceplate
[-] Control values				
[-] Parameters				
[-] Bit memory				
[-] Timers				
[-] Note texts				
[-] Block contacts				
[-] Position texts				

These comments can be acknowledged by the operator. Note text is not connected to the alarm logging; it is used for operator prompting and is visible as extra information.



6.1.9 Block contacts

Blocks of the individual control level are integrated into the SFC type via block contacts. Control modules (CM) are controlled by equipment modules. For this, besides the control for the blocks, feedback about the state is also needed. The controls and feedbacks are connected to the EM via block I/Os.

A cohesive group of block I/Os is called a block contact.

For connection of blocks of the basic automation through block contacts, you must specify the I/Os that are relevant for connection to an SFC type at the block types.

6.1 Characteristics

This is achieved by assigning the "S7_contact = true" system attribute to the block I/O. The technological blocks from the PCS 7 Advanced Process Library are prepared accordingly. If required, you can make project-specific modifications to the block types supplied in terms of the relevant I/Os.

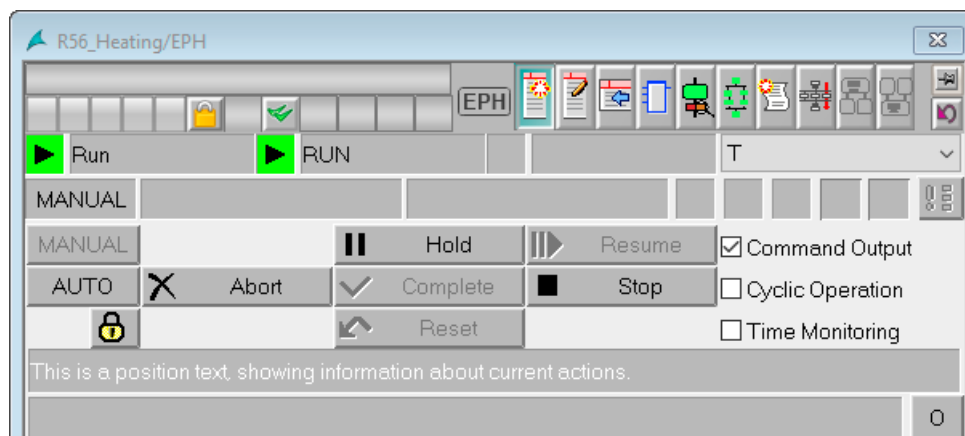
Characteristics to SFC-Typ-EN		Contents Of: 'Characteristics\Block contacts'			
<div> <div>Characteristics</div> <div>Control strategies</div> <div>Setpoints</div> <div>Process values</div> <div>Control values</div> <div>Parameters</div> <div>Bit memory</div> <div>Timers</div> <div>Note texts</div> <div>Block contacts</div> <div>Position texts</div> </div>		Name	Display name	Block	I/O name
		M01	M01	MotSpdL	M01
		V01	V01	VivL	V01

6.1.10 Position texts

Position texts are used to display the current progress in execution or the status of execution of the SFC.

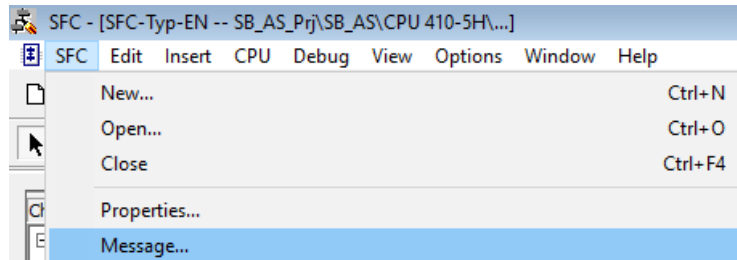
Characteristics to Heating		Contents Of: 'Characteristics\Position texts'		
<div> <div>Characteristics</div> <div>Control strategies</div> <div>Setpoints</div> <div>Process values</div> <div>Control values</div> <div>Parameters</div> <div>Bit memory</div> <div>Timers</div> <div>Note texts</div> <div>Block contacts</div> <div>Position texts</div> </div>		Name	Number	Displayed text
		PosiExplain	1	This is a position text, showing information about current actions.
		PositionText2	2	Another text
		POSINO3	3	Name of Position text is independent from the text shown

A position text is displayed on the OS in the SFC faceplate.



6.2 Messages

Equipment modules transmit messages, which can be set or reset from within the sequence. To do this, the message class and message class must be defined.



You can configure the following number of messages for an SFC type or an SFC instance:

- Seven messages requiring acknowledgment
- Five message that do not require acknowledgment

The SFC type itself requires the remaining available messages (one message for each message type and 10 status messages for SIMATIC BATCH).

6.3 Sequencing logic

The process engineering task itself is implemented in the sequencing logic. What the behavior is to be in the individual states of the EM is defined for each control strategy.

State logic of equipment modules

Core statement

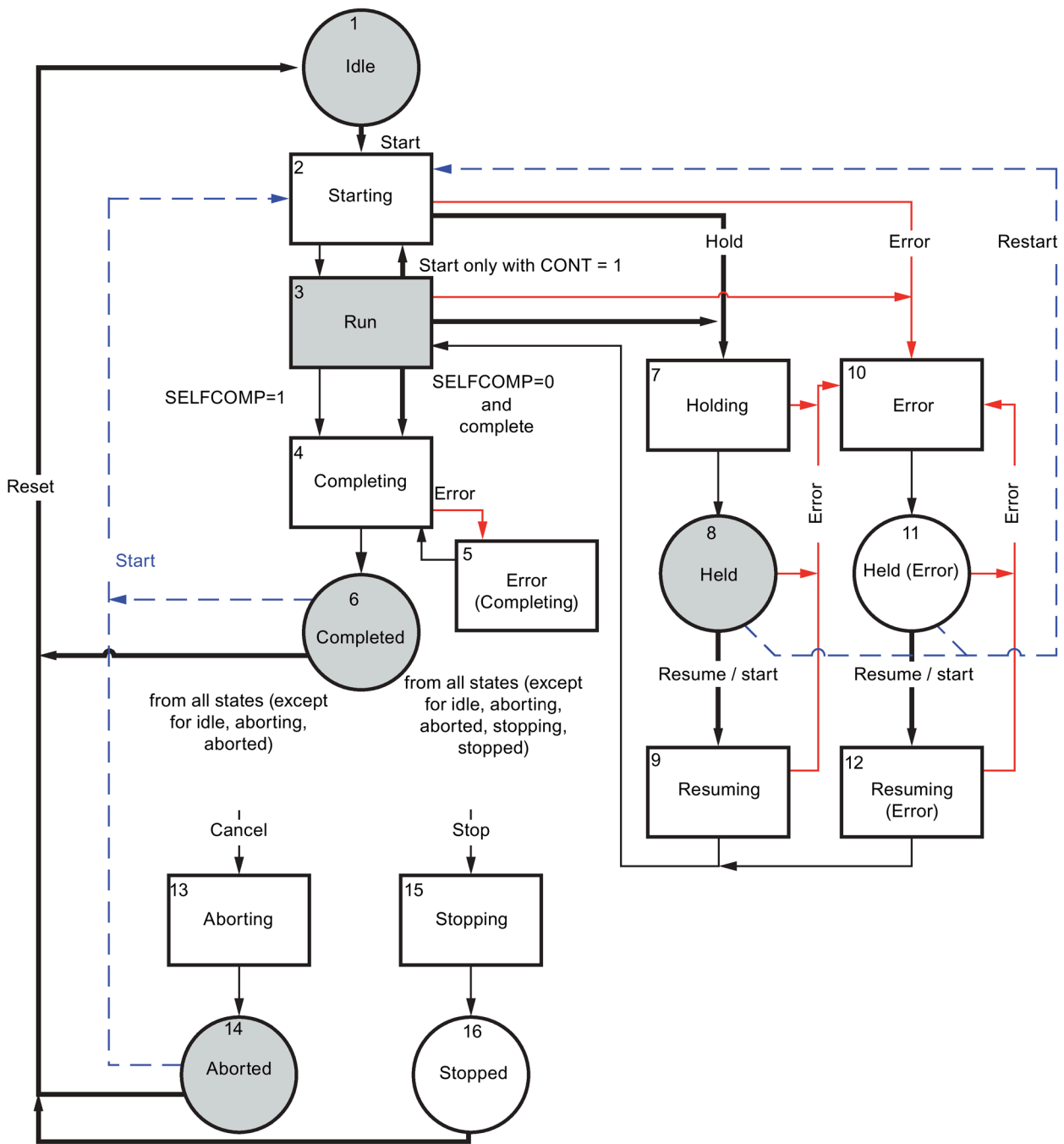
You can find the state logic used for the SFC type in the PCS 7 online help or in the "SIMATIC Process Control System PCS 7 SFC for SIMATIC S7" (<https://support.industry.siemens.com/cs/ww/en/view/109748747>) manual under "Diagram of the state transitions for SFC OSL".

The state logic of the SFC type has 16 different states:

- Ready [1] – Initial state; wait for Start command
- Starting [2] – Start processing after Start command
- Run [3] – Normal processing after ending the Start processing
- Completing [4] – Completion of processing after Complete command or implicit completion
- Error (Completing) [5] – Error processing during Completion processing
- Completed [6] – Completion of processing is finished; waiting for Reset or Start command
- Holding [7] – Hold processing after Hold command
- Held [8] – Hold processing completed; waiting for Resume command
- Resuming [9] – Resume processing after Resume command
- Error [10] – Error processing if errors occur
- Held (error) [11] – Error processing is complete and no more errors are present; waiting for Resume command
- Resuming (error) [12] – Resuming processing after Resume command
- Aborting [13] – Aborting processing after Abort command
- Aborted [14] – Abort processing is complete; waiting for Reset or Start command
- Stopping [15] – Stopping processing after Stop command
- Stopped [16] – Stop processing completed; waiting for Reset command

A sequencer can be stored in the transition states, e.g. "Starting" [2], "Completing" [4], and "Aborting" [13]. Sequencers are not to be inserted in the final states, e.g. "Completed" [6], "Aborted" [14] or "Stopped" [16].

State changes for SFC OSL



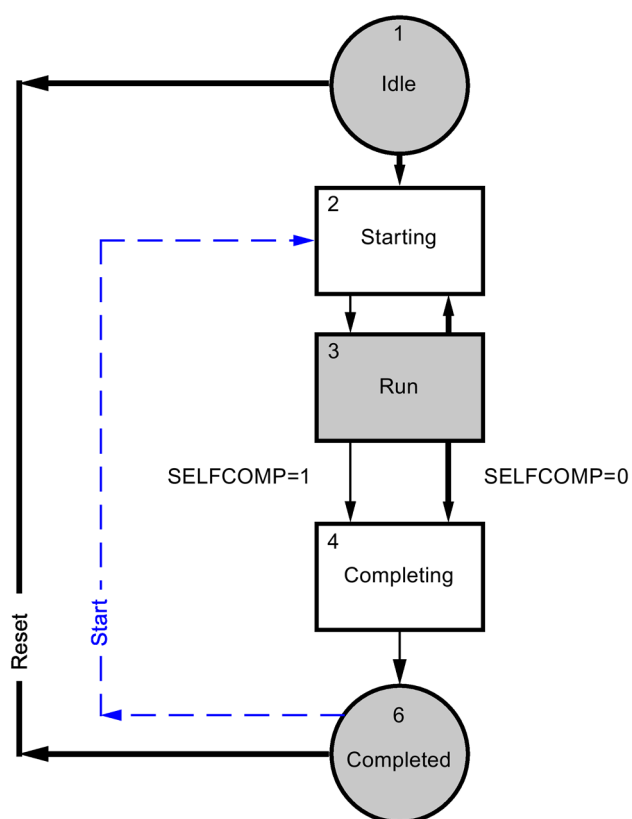
Key

	States that are exited through events
	Transition states that are implicitly exited
	States taken from OSL for SFC V5.x
	Events: Commands / Conditions / External Signals / Internal Signals
	Events taken from OSL for SFC V5.x
	Event: Error
	Implicit transitions that are triggered by the SFC, if the active sequencer has been processed to completion or if there is no sequencer to process.

7.1 Starting/Run division

The normal sequence of events is determined by the following states, if no errors occur:

From "Starting" [2] to "Run" [3]:

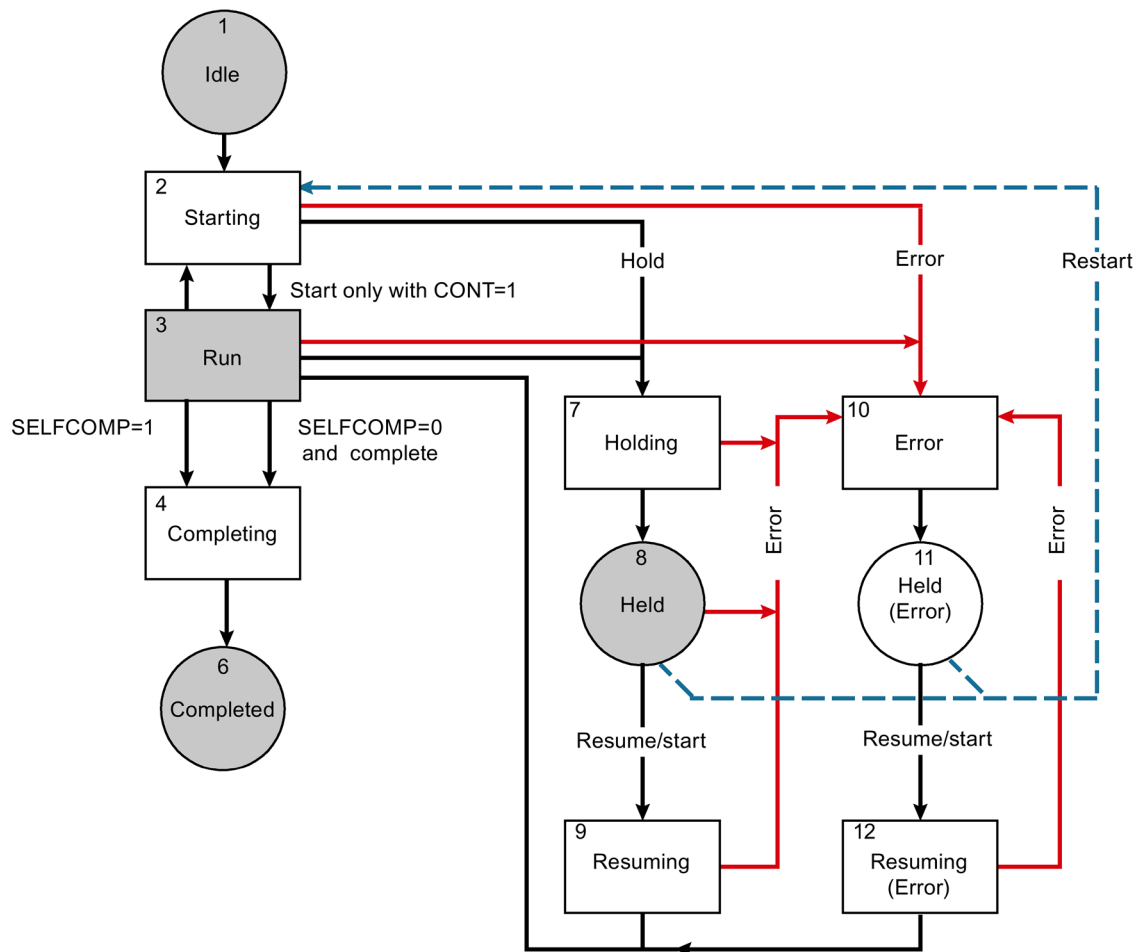


For all states of the SFC operating state logic, so also for Starting [2] and in the Run state [3], one or more sequencers can be configured, which are called depending on the start conditions. A Starting step sequencer [2] is advantageous if several control strategies are used, where the same basic settings are made prior to the Run step sequencer [3]. The operating state logic only changes to the next state when the sequencer called for this state has been processed. If a Starting sequencer is stored, the STARTING status remains on the block until this sequencer has been processed. This increases transparency when monitoring on a higher-level system such as SIMATIC BATCH. The STARTING status is symbolized by an open triangle and shows that this recipe step is still in a preparation phase and that the actual process is not yet running.

There are different philosophies when it comes to dividing sequence steps into the "Starting" [2] and "Run" [3] states in terms of functions:

- The "Starting" state can be used as the state in which switch-on is actually to take place (e.g. mixer on). In the Run state, the actuators are active.
- The "Starting" state can be used for preparation purposes (resetting of bit memories, etc.) and the actual EM sequencing logic becomes active (and, therefore, the actuators are switched on, for example) in the "Run" state.

As a rule, the error case "Held (Error)" [11] or the case "Held" [8] must be carefully considered:



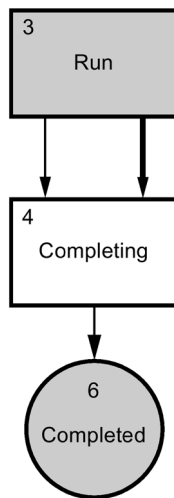
A return from the "Resuming" [9] or "Resuming (Error)" [12] states leads directly to the "Run" [3] state. If a large number of sequences are stored in the "Starting" state, they may also have to be executed in the "Resuming" state.

- A start condition for the sequencer Starting [2]:
STARTING = Starting (TRUE)
- A start condition for the sequencer Run [3]:
RUN = Run (TRUE)

The execution of the sequence (TARGETSEQ) can be started or continued with the step (TARGETSTEP) using the "TARGETSEQ" and "TARGETSTEP" contacts (caution: you must take the sequencer's starting conditions into account).

7.2 Completing and Completed

In the "Completing" state [4], the EM is disabled and switched to the safe state in accordance with the stored sequence.



A start condition for the sequencer Completing [4]:

- COMPLETING = Completing (TRUE)

In many cases, these are the same sequences as for the "Aborting" [13] or "Stopping" [15] states. In this case, the "Completing" [4] sequencer can also be used for "Aborting" [13] and "Stopping" [15]. The start condition for the sequencer "Completing" [4] is one of the following three conditions

- COMPLETING = Completing (TRUE)
- ABORTING = Aborting (TRUE)
- STOPPING = Stopping (TRUE)

In the "Completed" state [6], usually no more actions take place and therefore usually no sequencer is configured for this state. To get to the "Ready" state, the SFC must be reset. Reset can be configured by a parameter ("SELFRESET") in such a way that it occurs automatically in "Manual" mode (without operator input). In "Automatic" mode, the reset is performed by the higher-level recipe control (SIMATIC BATCH).

If you use SELFRESET, note that during operation with SIMATIC BATCH an equipment phase is restarted in the "Ready" state, while the recipe sequence continues in the "Completed" state of the SFC after this recipe step.

The transition to the "Completed" state is also the time when the actual values are read back from SIMATIC BATCH to the setpoints and logged.

7.3 Differences between Holding and Holding (error) branch

The "Held" [7] branch is intended for scheduled/desired holding procedures. This can also be achieved by means of operator input in SIMATIC BATCH, for example.

The "Held (Error)" [10] branch is intended for an "undesired" error case and is not to be activated manually (except for test purposes).

It is frequently the case that the same thing is implemented in both branches.

7.4 Holding, Held, and Resuming

In the "Holding" state [7] the normal sequence is held and switched to the safe state in accordance with the stored sequence. This can also consist of a targeted shutdown in several stages.

- A start condition for the sequencer Holding [7] is:
HOLDING = Holding (TRUE)

In the "Resuming" state [9] the EM currently in the "Held" state [8] is resumed. There are various positions at which the active sequencer may be further processed.

See also

Returns when resuming (Page 52)

7.5 Error, Held (error), and Resuming (error)

In the "Error" state [10] the lower-level blocks are switched to a safe state. This can also consist of a targeted shutdown in several stages.

- A start condition for the sequencer Error [10]:
ERROR = Error (TRUE)

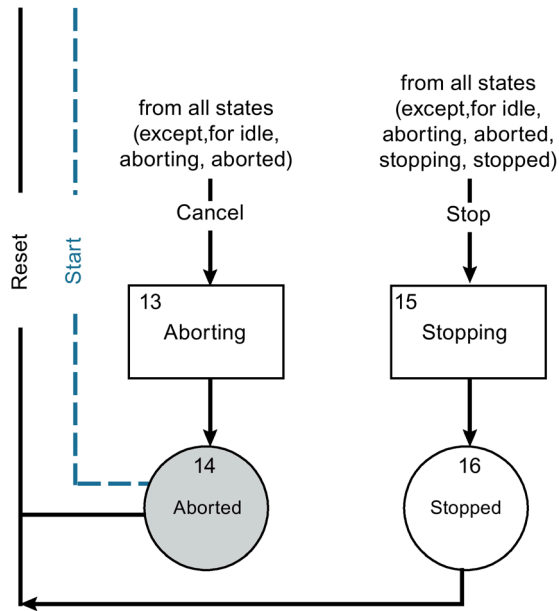
In the "Resume" state [12], the EM currently in the "Held" (error) state [11] is restarted. There are various positions at which the active sequencer may restart.

See also

Returns when resuming (Page 52)

7.6 Aborting and Stopping

In the "Stopping" state [15] the EM is retracted in a controlled manner, similar to with the "Completing" state [4] (e.g. the product is transported out of the worm). In the Aborting branch [13] everything is disabled immediately, without observing an order or any feedback (worm off immediately). For the most part, however, the same sequence is implemented in the "Aborting" [13] and "Stopping" [15] states.



- A start condition for the sequencer Aborting [13]:
ABORTING = Aborting (TRUE)
- A start condition for the sequencer Stopping [15]:
STOPPING = Stopping (TRUE)

The "Aborting" state [13] can be reached from "Stopping" [15], although not vice versa.

Functionalities and solution paths

This section describes the individual task definitions/behaviors of an EM, along with possible solutions.

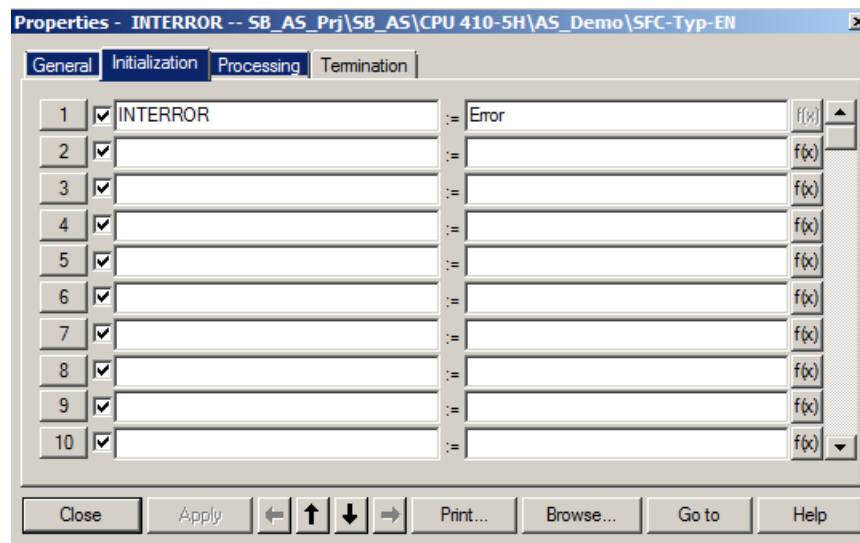
The solutions refer to creation using the SFC type.

8.1 State change

An SFC instance status change (step sequencer change) can be achieved in one of the following ways:

- In "Manual" mode, by means of a manual operation (e.g. via the EM faceplate)
- In "Automatic" mode, by means of the automatic interface
- Via LOCK inputs for interconnections (e.g. LOCKERROR)
- Via INT inputs (e.g. INTHOLD) for a state change from the sequencer

In this context, it is really easy to use INT elements in the sequencer. The INT input is simply activated. The SFC-type state logic evaluates and immediately resets the IN-OUT parameter. The state change will take place if it is permitted within the state diagram. If the command cannot be executed, it will be rejected.

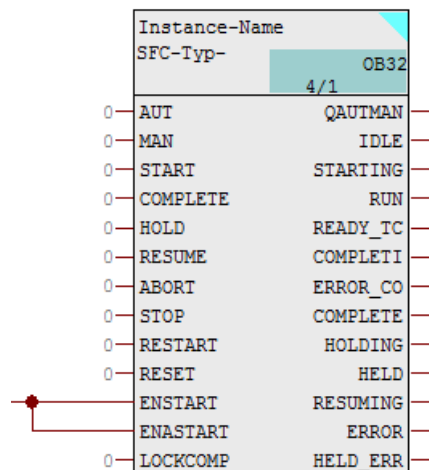


An example of using an INT command is the transition to the Error state (INTERROR) as a function of a condition.

8.2 Closing and resumption lockouts

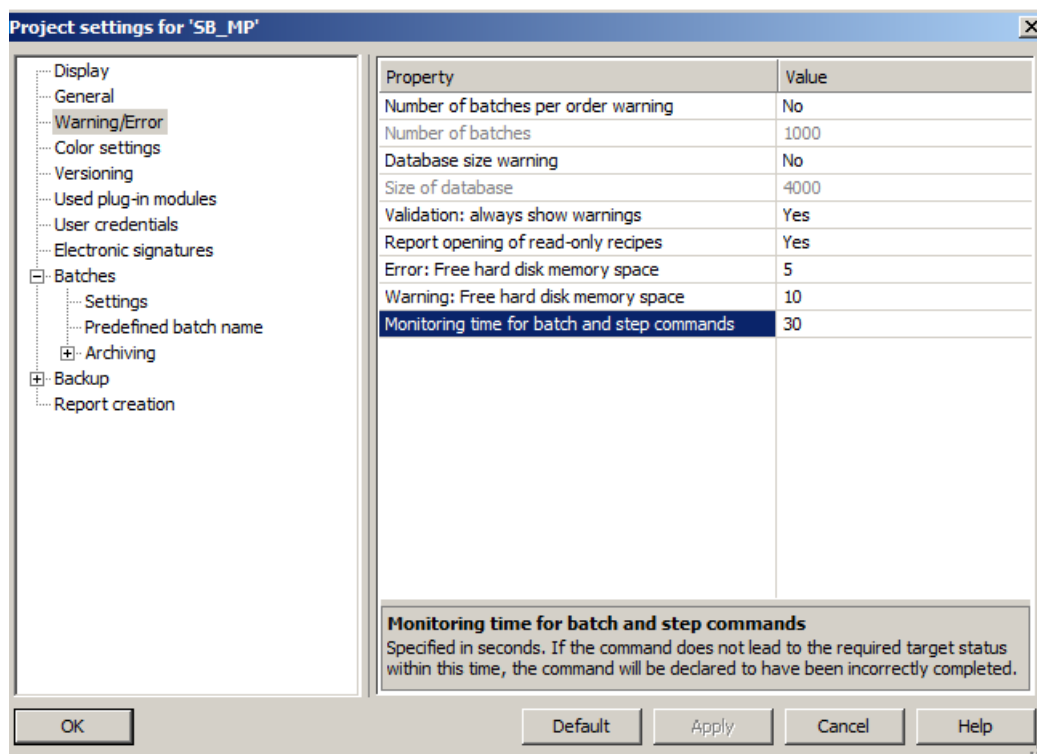
An SFC is started only when the requirements for starting are fulfilled. The start enable must be set (ENSTART = 1) and the SFC must be in a state in which starting is permitted. The closing and resumption lockouts are both instance-specific.

The closing lockout is interconnected via the default input "ENSTART" (for starting from the initial state) or "ENASTART" (for starting from the "Run" state – see also "Active control strategy change"). (Page 38)The interlock can be implemented using an interlock block connected in series. The "ENRESUME" input is used for the resumption lockout.



Using the SFC with SIMATIC BATCH

With a Start command SIMATIC BATCH expects a status change within a preset time. In the BATCH Control Center, this time can be specified under "Options > Settings > Project settings > Warning/error".



If a batch or step command does not lead to the desired target state within the selected time in seconds, then the command is terminated as incorrect and a warning message is issued. The warning displayed as a dialog is confirmed or acknowledged with the "OK" button. The further recipe procedure is not influenced by the warning. The warning is provided for your information. The warning is also listed in the "Extended Batch Information" dialog box in BatchCC.

The default value is 30 seconds. Numbers between 10 and 10000 seconds are allowed.

This is the recommended behavior if the start enable (ENSTART) of the SFC is not set due to an external error (e.g. motor fault).

If, besides function-relevant start conditions, process-relevant conditions (e.g. dosing device available) are also connected to ENSTART, SIMATIC BATCH responds in the same way.

In this case, the batch should continue without acknowledgment once the starting condition occurs.

The following solutions can be implemented to meet this requirement:

1. Query of the process-engineering starting condition in a recipe transition before the start of the recipe phase

In this case, SIMATIC BATCH only issues the starting command once the process-engineering starting condition has been met.

2. The process-engineering starting condition is set not to ENSTART but to a dedicated process value input on the SFC. This value is checked in a transition in the running (/starting) sequence in the SFC.

In this case, the starting command is output immediately by SIMATIC BATCH and is also executed immediately by the SFC/SFC instance (status transition IDLE > (Starting) > RUN). The SFC/SFC instance then waits for the process-engineering starting condition to be in place at the transition.

8.3 Active control strategy change

"Active control strategy change" means that a control strategy can be started while another one is running. This represents a special case that makes use of the possibility to restart an EM in the active state.

An active control strategy change is influenced in the SFC type by the "ENASTART" input (Enable Active Start = Enable command "Start" [in Run]). If the "ENASTART" input is activated, an active control strategy can be aborted when another is started. The same control strategy can also be restarted.

Note

A prepared control strategy is only adopted during SFC start (both in MANUAL and in AUTO).

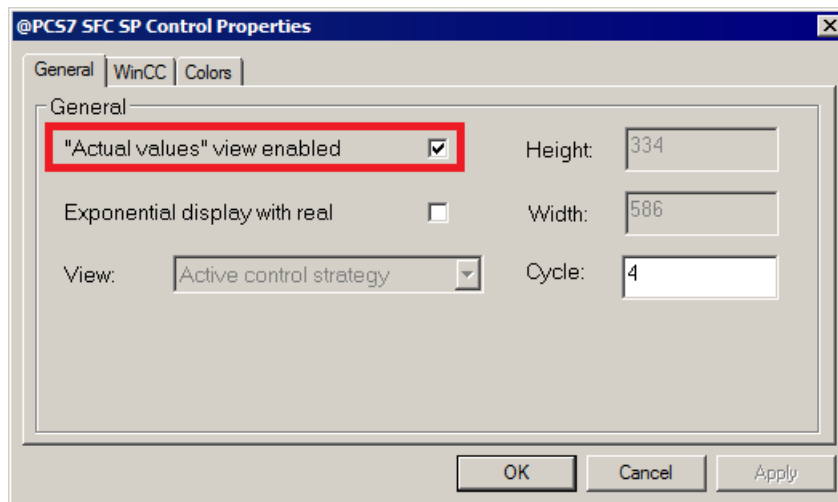
After the manual execution of the "Start" command in manual mode, the prepared control strategy and the prepared setpoints are adopted.

While in manual mode the "Start" command has a direct effect, additional preconditions have to be fulfilled in automatic mode. ENASTART := TRUE and the Continuous flag at the QCONT output must be set, which is only possible in the "Ready_TC" state.

8.4 Setpoint changes during operation

If a setpoint needs to be changed during operation, please observe the following:

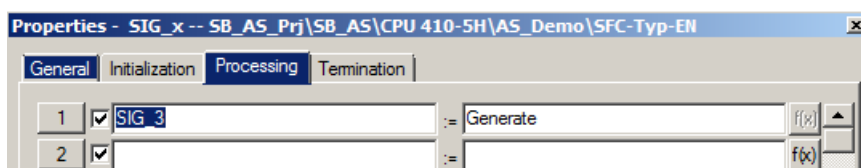
- Pay attention to the setpoint change in the sequential control system of the EM. It may be the case that a return must be performed within the sequence when a setpoint is changed, or that the setpoint must be assigned in each relevant step (e.g. when forwarding to a CM).
Alternatively, sequencer preprocessing or sequencer postprocessing can be used for this purpose. (See also "Preprocessing" / "Postprocessing" tab (Page 58) and "Routing of values" (Page 138))
- The setpoint in the EM can only be changed if that EM is in "Manual" mode. In "Automatic" mode, setpoint operation is blocked for equipment modules. In this case, the setpoint operation must be performed by the higher-level recipe control (SIMATIC BATCH).
- If a setpoint change needs to be made in the active state, it must be enabled on the EM block for every setpoint. The enable is implemented via input "xx_ENOP" (xx: I/O name of the setpoint). If input "xx_ENOP" is activated, the setpoint can be changed. The input can also be set or reset in the sequencing logic (default = Enabled).
- The setpoint change also has to be activated in the faceplate. The screenshot below shows the setting for faceplate in the picture @pg_@sfc_type_actualsp.PDL:



If no setpoint change is required during operation, you can deactivate it for all SFC types in the dialog box too.

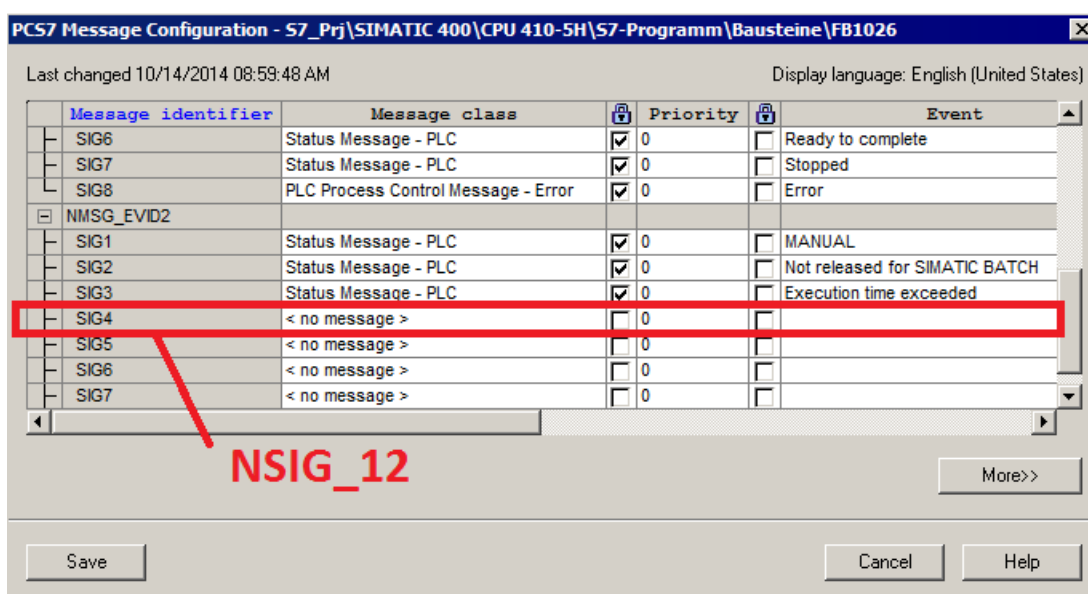
8.5 Transmitting messages

Messages are transmitted in an EM with an SFC type via the inputs "SIG_x" bzw. "NSIG_x".



These elements must be reset again when the reason for the message no longer exists.

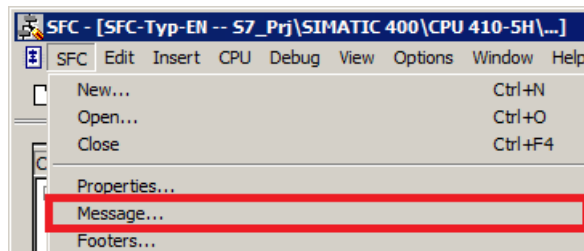
An SFC type contains one ALARM_8P and two NOTIFY_8P blocks, although some messages are allocated by default.



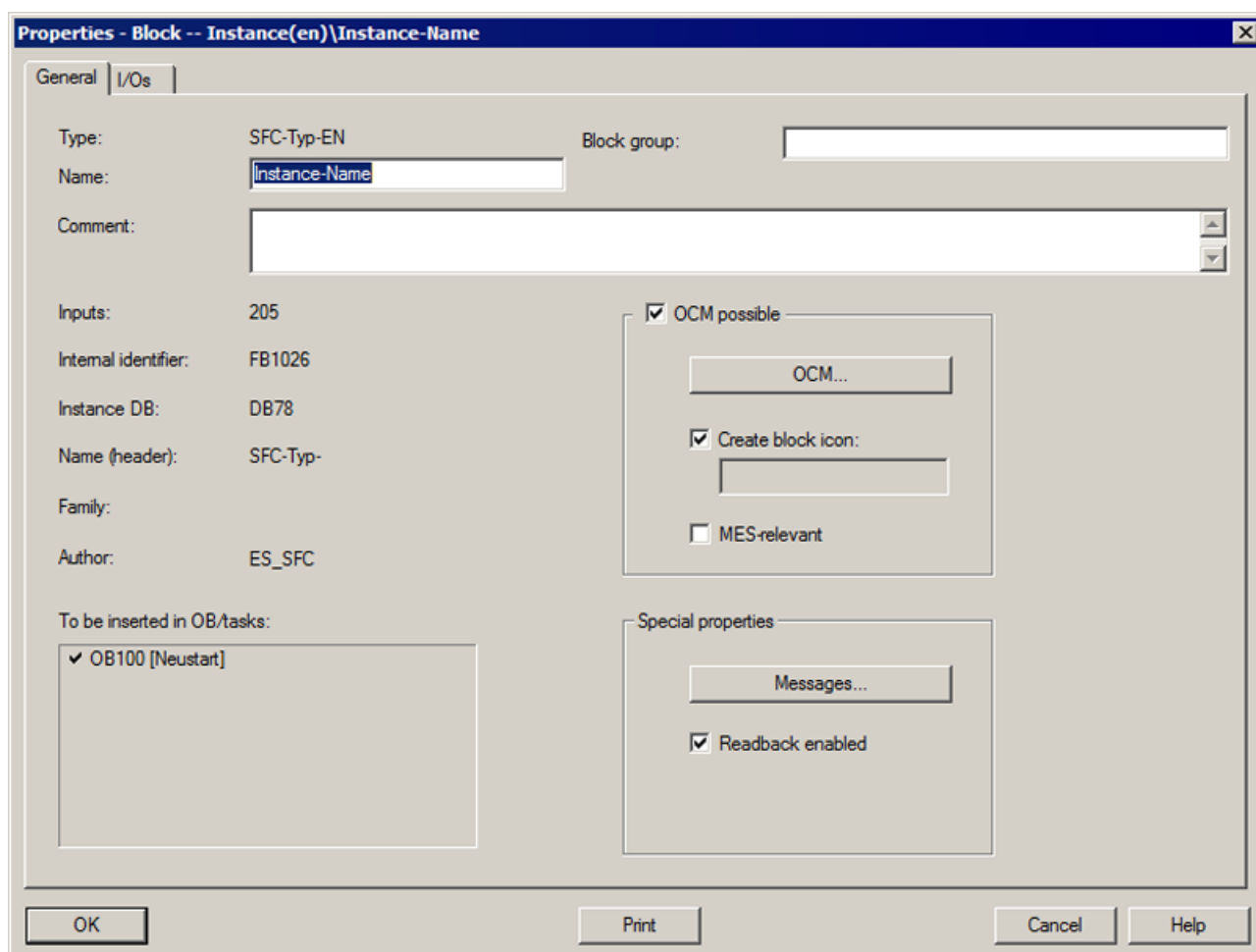
The free messages not used internally can be used and are connected to the inputs "SIG_x" and "NSIG_x". For example, input "NSIG_12" is the fourth message in the second NOTIFY_8P block (SIG4).

I/O name	Message identifier	Message block
SIG_2	SIG2	Alarm_8P
SIG_3	SIG3	Alarm_8P
SIG_4	SIG4	Alarm_8P
SIG_5	SIG5	Alarm_8P
SIG_6	SIG6	Alarm_8P
SIG_7	SIG7	Alarm_8P
SIG_8	SIG8	Alarm_8P
NSIG_12	SIG4	2. Notify_8P
NSIG_13	SIG5	2. Notify_8P
NSIG_14	SIG6	2. Notify_8P
NSIG_15	SIG7	2. Notify_8P
NSIG_16	SIG8	2. Notify_8P

The messages themselves can be edited within the SFC type in the PCS 7 message configuration window using the "SFC > Message ..." command.



Alternatively, you can adapt the messages instance-specifically using the "Messages" button in the block property dialog of the instance.



You can trigger messages via the "SIG_2" to "SIG_8" I/Os through interconnections in the actions of the steps or through direct block interconnections.

Message associated values "AUX_PR04" to "AUX_PR10":

The message associated values "AUX_PR04" to "AUX_PR10" are available as standard interface connections and can be freely used.

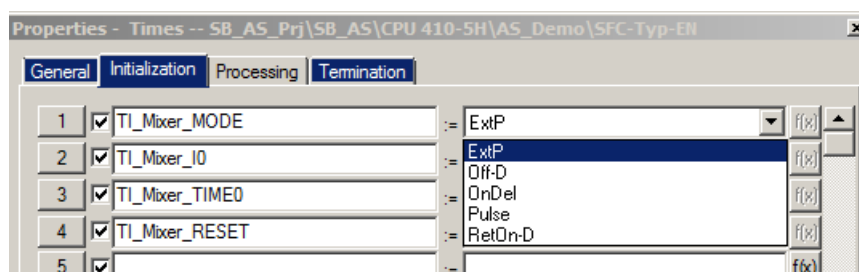
AUX_PR01 to AUX_PR03 are reserved for SIMATIC BATCH and are connected internally. For example, AUX_PR03 contains the respective BATCH_ID of the active batch. AUX_PR01 is reserved for the BATCH name (BA_NA) and AUX_PR02 for the step number (STEP_NO).

8.6 Using timers ("Timer")

8.6.1 Example: Using "Timers"

"Timers" are often used in an EM. The SFC type provides the option of defining the "Timers" in the characteristics dialog. An instance of the BICO block "TIMER_P" is used for each "TIMER". The interface elements of this block are added to the interface of the SFC type.

The time counter's mode can be set via input "xx_MODE" (in the example xx = TI_Mixer).

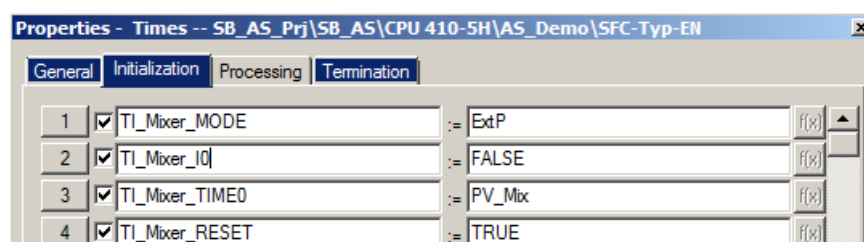


Five "Timer" modes are available:

- ExtP (start timer as extended pulse)
- Off-D (start timer with off delay "Off-D")
- OnDel (start timer with on delay)
- Pulse (start timer as pulse)
- RetOn-D (start timer with retentive on delay)

The exact principle of operation of the "TIMER_P" and its modes are explained in detail in the block help and are only presented here as an example for use in the SFC environment. This should be reset prior to using a "Timer". To do this, the "xx_RESET" input is set to 1 and the input for starting the time is reset (xx_IO = 0).

It is also possible to load the time (xx_TIME0).

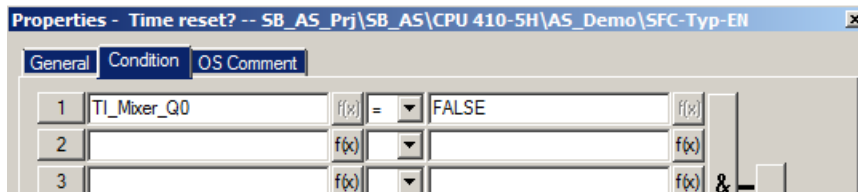


as this is what the SFC type requires. If the setpoint is defined in minutes, then the time must be calculated in seconds.

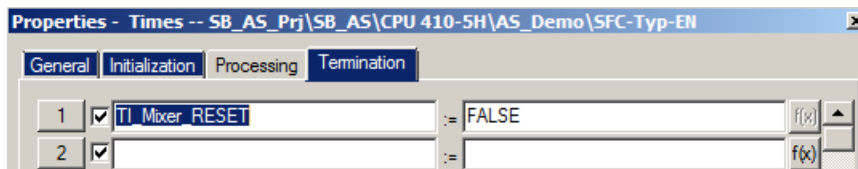
The SFC calculation in steps or transitions can be used for calculation of a "Timer".

8.6 Using timers ("Timer")

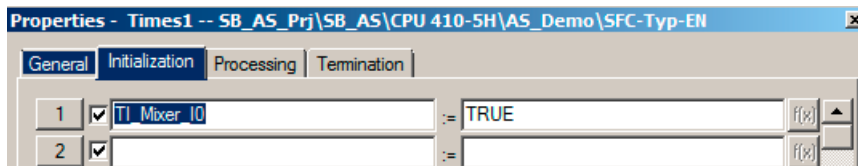
If the "Timer" "xx_Q0" was reset, the sequencer can continue running if this is queried in a transition, for example.



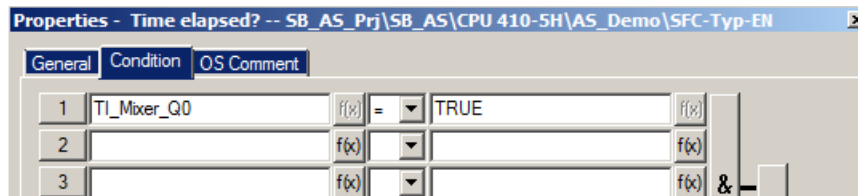
The reset command in the "Termination" tab is also reset (xx_RESET = 0).



If the "Timer" is to be started, the input of the timer (xx_I0) can be set.



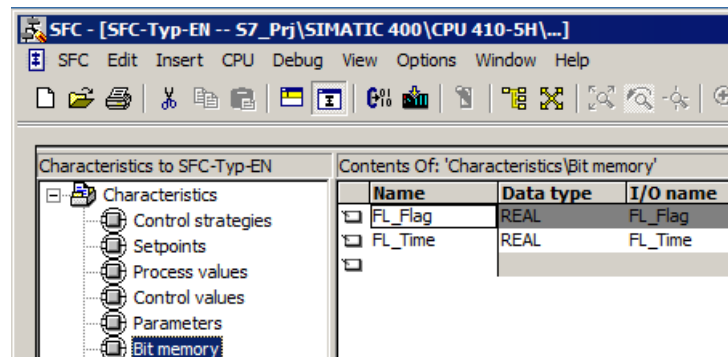
The output (xx_Q0 = 1) is used to query whether the "Timer" has elapsed.



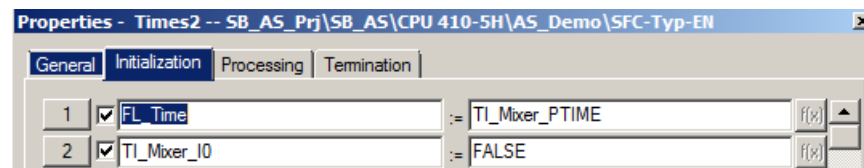
8.6.2 Example: Times in Hold

If a time is used (as described up to now), it continues to run if the EM switches to the "Holding" state. If you do not want this to happen, the time must be stored temporarily.

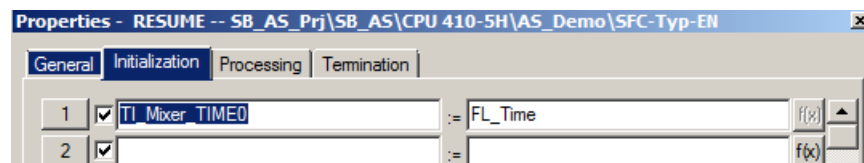
A bit memory is defined for temporary storage in the characteristics dialog (e.g. saved time (I/O name: FL_time)).



The time currently remaining is assigned to the bit memory (xx_PTIME). The input pulse of the time counter (xx_IO) also has to be reset.



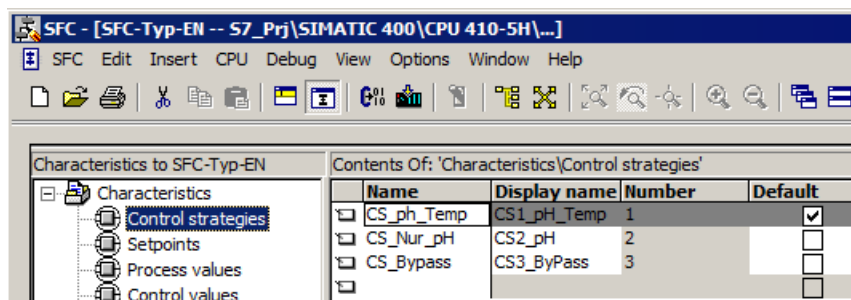
The remaining time can be set again in the "Resuming" state.



8.7 Presetting control strategies

To prevent the most recently executed control strategy from being used as the new control strategy on starting, a control strategy can be defined as a "default".

Once the SFC has been executed, the control strategy selected as the default is automatically offered for the SFC faceplate's "Prepare control strategy" setting.



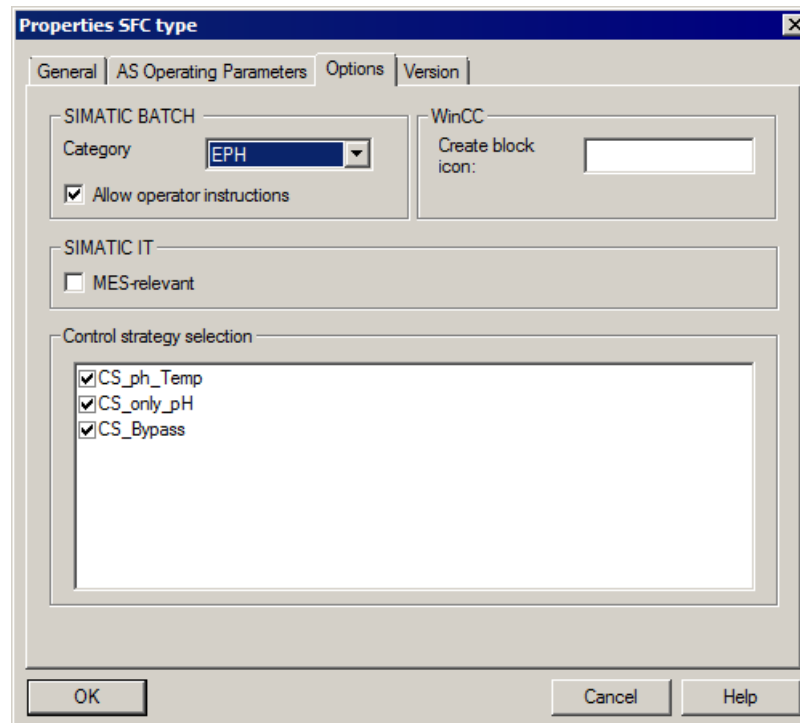
8.8 Instance-specific deselection and selection of control strategies

If the type/instance model is used, it may be necessary to deselect control strategies or sequences on an instance-specific basis in order to limit the number of different types.

For example, for an "Acid-dosing" SFC type the temperature (control strategy 1, standard) can be set, as can a dosing procedure optionally be set purely using the pH value (control strategy 2).

If an equipment module at a unit is only designed for dosing an amount of acid (pH measurement not available), you will need to deselect control strategy 2 at the SFC instance.

If pH dosing is not necessary, the acid supply can be bypassed using control strategy 3 (bypass).



If the "Allow operator instructions" check box is selected, operator instructions are enabled for SIMATIC BATCH. You can find more information on operator instructions in the "SIMATIC Process Control System PCS 7 SFC for SIMATIC S7"

(<https://support.industry.siemens.com/cs/ww/en/view/109748747>) manual.

8.9 Multiplexing control modules

If a control module (CM) must be controlled by several equipment modules, note the following:

The CM cannot be connected to the EM directly. You need a block which will switch the connection elements (CM inputs) between the possible sources. It is also recommended to have this block manage CM assignment as well.

You can find ready-made blocks such as ShrdResL and ShrdResS in the Advanced Process Library for this purpose. These blocks coordinate the access from up to four CFC or SFC step sequencers to a technological block of the families "Drives" or "Dose". These blocks include:

- Motors
- Valves
- Dosers

8.10 Control modules in Auto and Manual modes

Introduction

CMs can be operated in "Manual" (operating personnel) and "Automatic" (operation via the EM). The four possibilities typical in practice with the operating mode in which CMs can be operated are explained below:

- CMs always in Automatic
- CMs in Manual in initial and hold states
- CMs in Automatic on start
- All CMs only to automatic (AUTO) upon activation

All CMs always in Automatic

All CMs are switched to "Automatic" mode in the initial state of the system. The equipment module can only be started when all CMs are in "Automatic" mode (QAUTMAN=AUTO).

All CMs in Manual in Initial and Hold states

The CMs can only be switched to "Manual" mode when in their initial state and when in "Hold". The EM can only be started when all CMs are in "Automatic" mode (ENSTART, ENRESUME). The CMs must be switched to "Automatic" mode prior to starting.

All CMs in Automatic on start

All CMs are switched to "Automatic" mode when started by the EM. If all CMs are not switched to "Automatic" mode after the start, the EM enters the ERROR state.

All CMs only to automatic upon activation

The CMs are only switched to "Automatic" mode when the equipment module performs an activation in the sequencer. The EM can also be started without all CMs being in "Automatic" mode. CMs can be switched to "Manual" mode at any time without the EM developing an error.

Other options and combinations are also possible. The correct one for each particular case will depend on the business and plant operators in question, as well as on how the equipment modules are being used. For example, may a CM be operated manually at all, and what is the plant's level of automation?

The way in which an EM responds to CM operation will have an effect on recipe control. If the equipment module switches to "Hold", SIMATIC BATCH will respond as well.

Note

Rules for automatic operation

For the commands of the Batch Control Server can be processed in automatic mode, the following inputs in the interface block need not remain interconnected and their value must be set to 1:

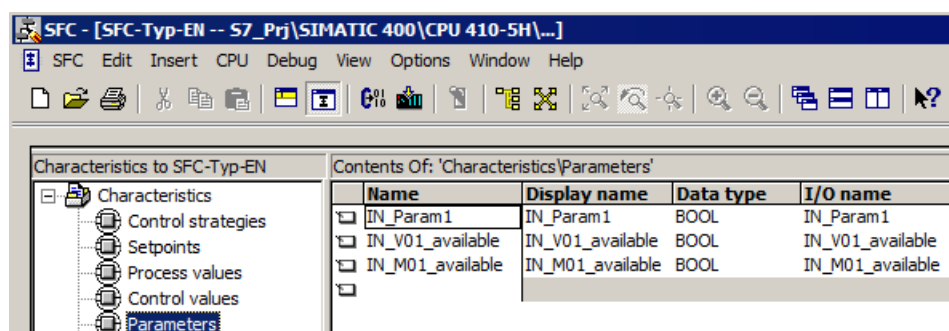
- ENSTART
- ENCOMPLETE
- ENHOLD
- ENRESUME
- ENABORT
- ENSTOP
- ENRESET

8.11 Optional control modules

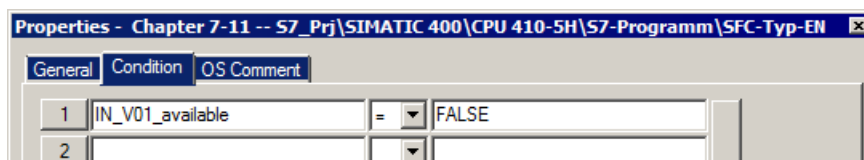
When defining an EM, it makes sense to keep the number of EM types to a minimum. This can be achieved by using optional CMs for the EM types.

An optional CM is a process tag or single control unit, which is not available in every SFC instance. This could mean, for example, that various control strategies cannot be executed in these SFC instances. These control strategies must be blocked in the SFC instance. It is also possible that a control strategy can work with or without optional control modules.

Optional control modules must be detected in the sequencer and queries performed as to their existence. This can be realized using a parameter.



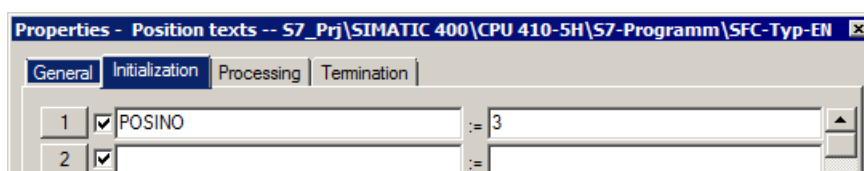
An optional CM can always be activated from the sequencer. However, for the purpose of querying the state of optional control modules, it will be necessary to take account of whether or not the CM is actually present; The previously created parameter is queried for this.



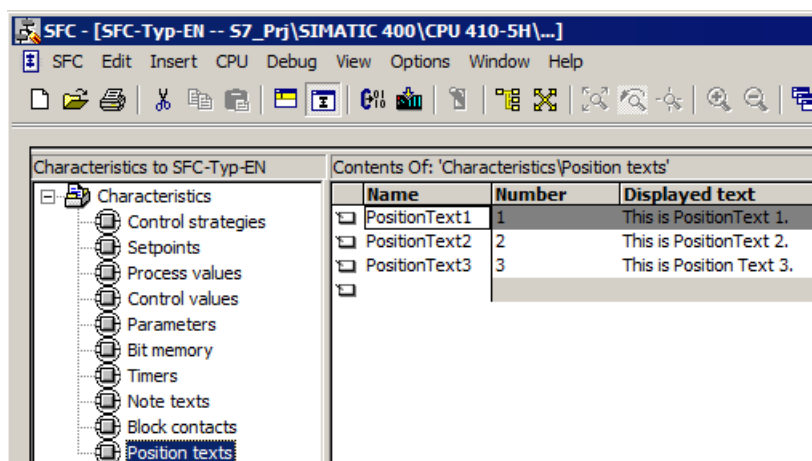
8.12 Setting position texts ("POSINO")

The position is set in the corresponding steps. In each case, the relevant number is assigned to the "POSINO" input. As the position texts are displayed in the EM faceplate, it makes sense to use them in the sequencer. Attention must be paid to the text length.

The same position number can be assigned to associated steps within a sequencer.



These position numbers must be defined in the characteristics in advance.



8.13 Self-terminating and non-self-terminating equipment modules

Introduction

There are two ways of terminating an EM:

- Self-terminating equipment module
- Non-self-terminating equipment module

The SFC type supports both methods. The configuration is set via the SELFCOMP input. The SELFCOMP input changes the termination behavior of the active sequencer in the SFC type.

Self-terminating EM

An example of a self-terminating EM is the dosing procedure. When dosing is complete, the equipment module will be closed automatically.

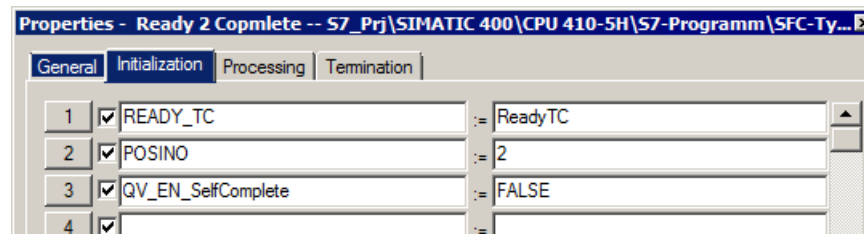
Non-self-terminating EM

An example of a non-self-terminating EM is the mixing procedure.

If the SELFCOMP input is configured with a 0 (FALSE), the SFC remains in the Run state. In this case, the READY_TC (ready to complete) output is set when the active sequencer reaches the end step. The state can also be reached by setting READY_TC to 1 (TRUE or ENUM value ReadyTC) in a step.

Note: The output is called "READY_TC" with underscore, while the value "ReadyTC" is written in upper-lower case and without an underscore.

SIMATIC BATCH (as higher-level controller) can check the following step enabling conditions in the control recipe. If these are met, the EM is completed by SIMATIC batch and set to initial state. The higher-level control then activates the next recipe step.



READY_TC (block I/O) = ReadyTC (ENUM for TRUE)

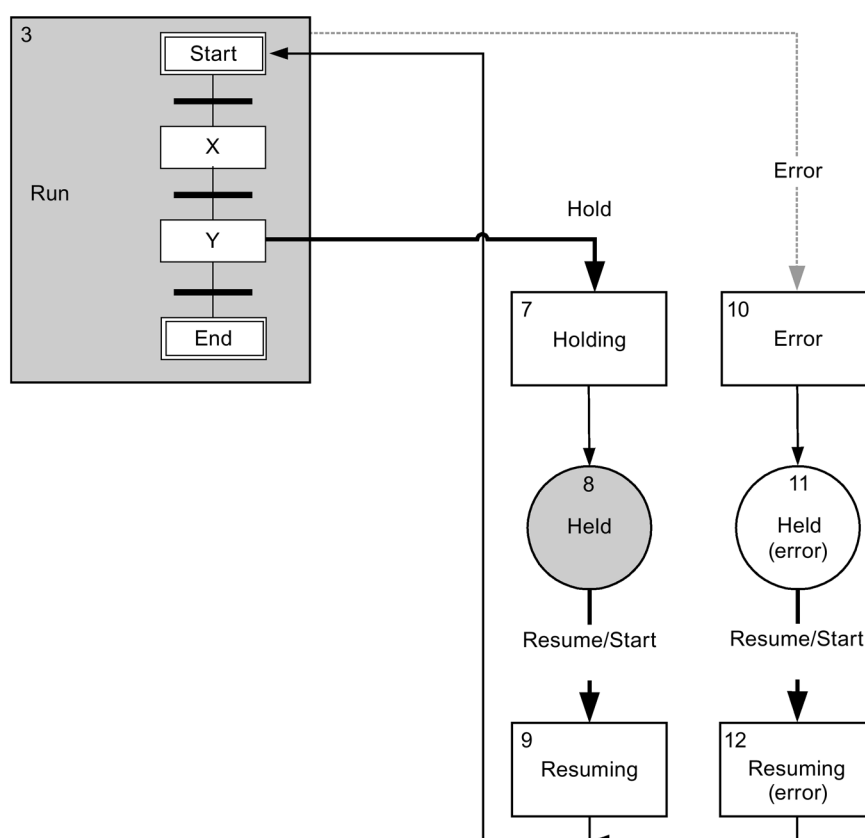
8.14 Returns when resuming

Starting from the beginning

You can implement the active sequencer in such a way that it is restarted from the beginning on resumption (see figure). This keeps the resumption branch [9] relatively small. In this case, flow counters or dosing measurements, for example, may not be reset in the active sequencer; rather, they must be switched to the "Starting" state.

This can be achieved with the RUNHOLD = "Abort" [TRUE] at the instance: When there is a change from "Run" [3] to "Holding" [7], the previous sequencer is **aborted** and the new sequencer is started.

(RUNHOLD = Response of the "Hold" command: FALSE: Hold / TRUE: Abort)



Jump to exited step

The option exists to jump back to the step from which the active sequencer was exited.

This can be achieved via RUNHOLD = "Hold" [FALSE] (default value) at the instance: When there is a change from "Run" [3] to "Holding" [7], the previous sequencer is **held** and the new sequencer is started.

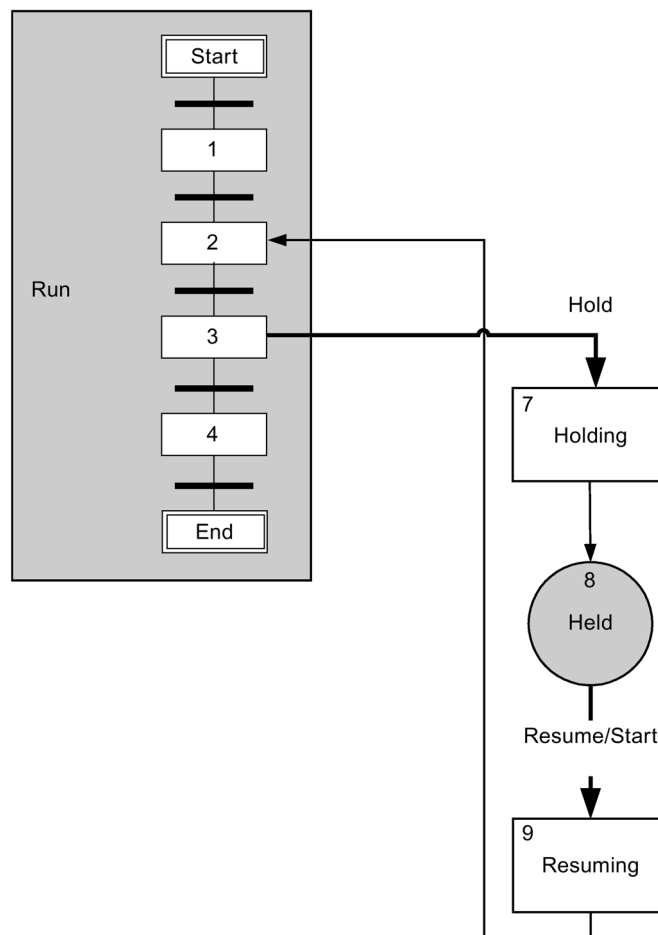
Resuming in a defined step

There is also an option to return from "Resume" to a defined step in the active branch. In this case, the resumption step is more complex, as the valves or motors, for example, have to be appropriately activated again.

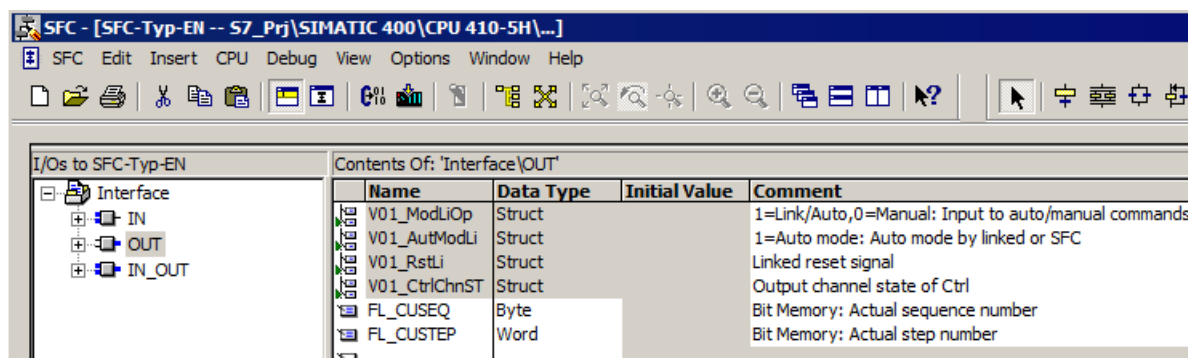
Example: Return to a defined step

To enable a defined step to be returned to during resumption, the step can be defined while the active sequencer is being executed.

In the example, the process is to resume from step 2. The process is held in step 3. On resumption, step 2 must be entered as the resumption step.

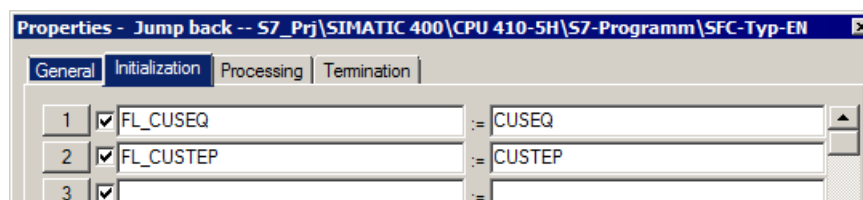


First, step flags (FL_CUSEQ, FL_CUSTEP), in which the current step (CUSEQ, CUSTEP) can be saved, are defined. Data types which are not available in the characteristics are required for this, so the flags must be defined in the I/O view.

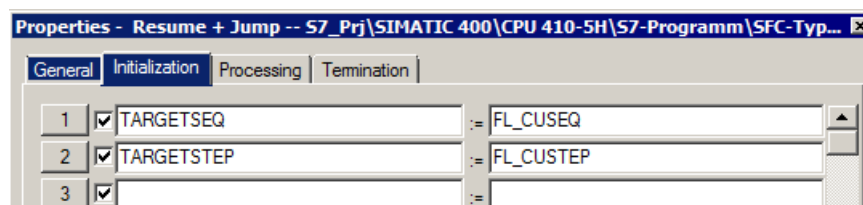


The defined "FL_CUSEQ" and "FL_CUSTEP" step flags are set in step 2.

(Step 2 is the step to be returned to during the resumption procedure.)



If processing is to be resumed in the Held state, the return is entered in the resumption sequencer (via "TARGETSEQ" and "TARGETSTEP").



If the active sequencer starts, a jump will be performed to the resumption step which has been set – this is provided that the sequencer in the "Resume" state and the one in the "Run" state are different sequencers.

Note

Since only one target step can be defined, it may not be in a simultaneous branch, otherwise the sequencer cannot be terminated because not all simultaneous branches would be active.

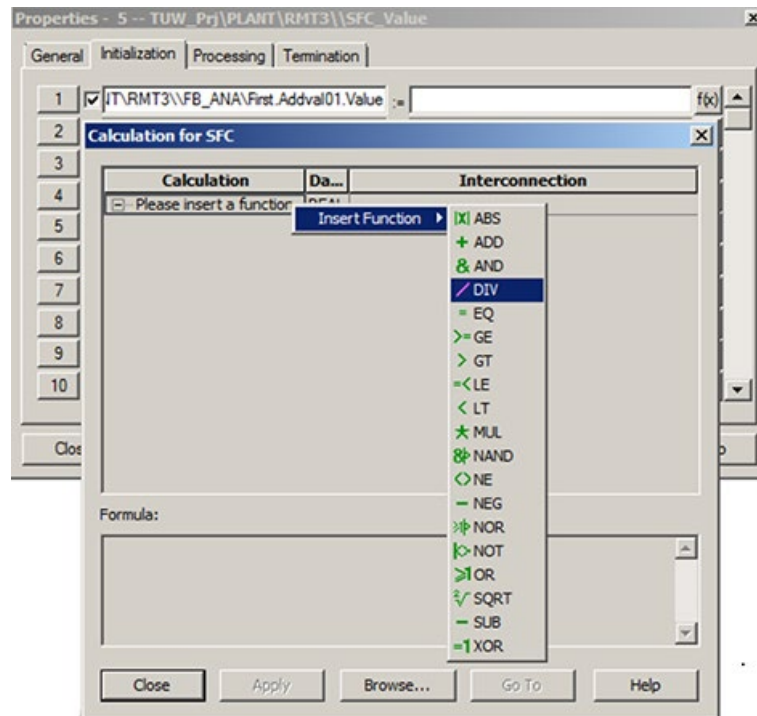
Note

After a RESUME command, the operating state logic always goes to the "Run" state, even if, for example, the "Stopped" state was reached from a "Starting" state via the "Holding" state. (Correspondingly or more frequently "Error" and "Held (error)" in some cases).

8.15 Calculations in the SFC

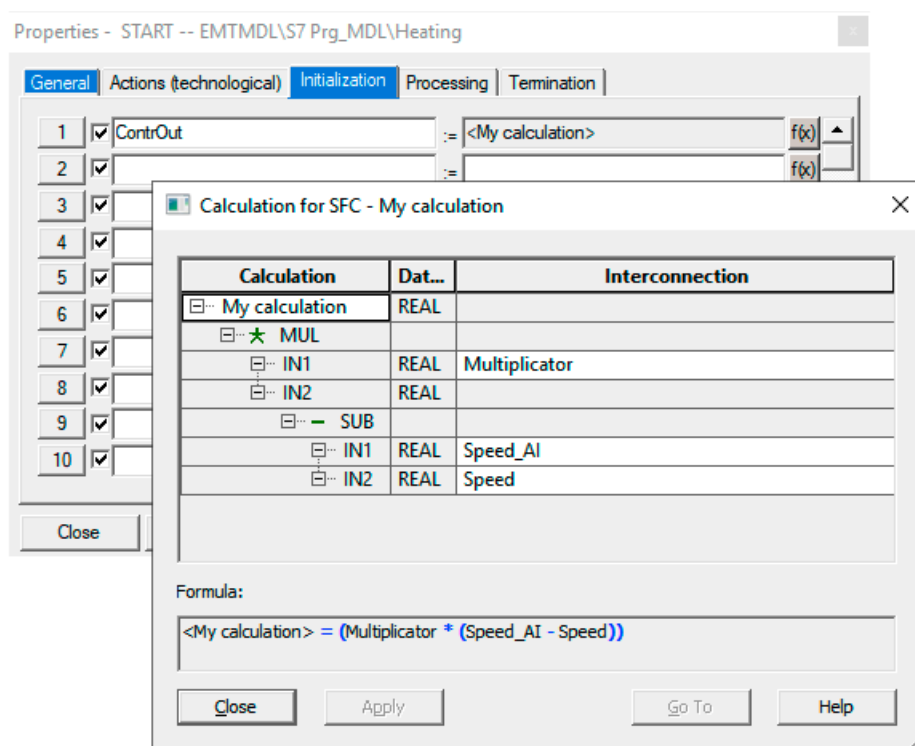
Calculations in the SFC

Starting in SIMATIC PCS 7 V8.2 calculations in steps, transitions or sequencers can be configured in the SFC Editor. Following compilation and download of the program to the AS, the calculations are performed during runtime of the SFC.



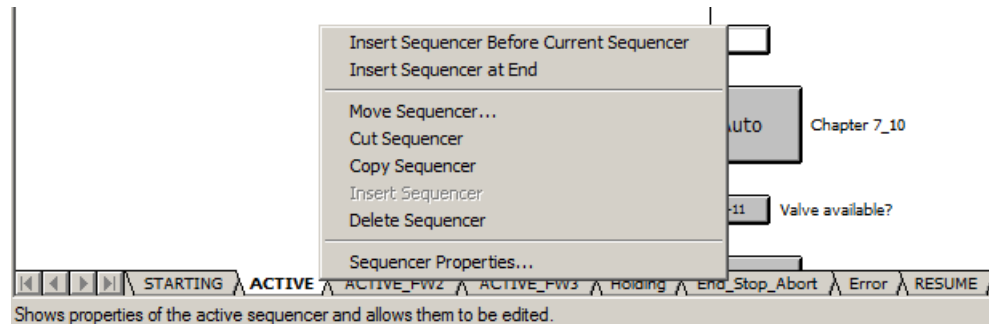
The figure below shows an example of a calculation in the "My calculation" dialog:

ContrOut := MyCalculation = Multiplier * (Speed_AI – Speed)



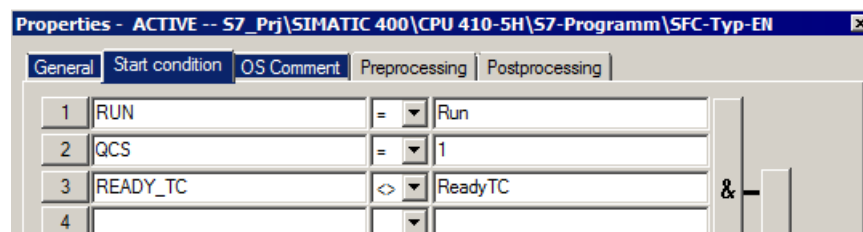
8.16 Start conditions for sequencers

The point at which the sequencer is to be started is defined in the sequencer starting conditions. These conditions form the basis of the relationship between the state logic and the sequencers. The starting conditions for a sequence can be defined via the "Sequencer properties..." command.



The "ACTIVE" sequence is used by way of example to show the starting conditions:

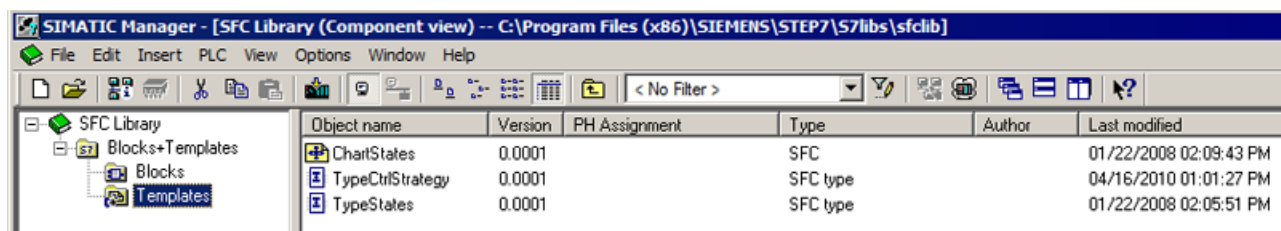
- RUN = Run (TRUE) AND
- QCS = 1 (control strategy 1) AND
- READY_TC <> "ReadyTC" [TRUE]
(Flag READY_TC is not set yet)



In some states, executing the final step of a sequencer results in a state change (implicit), such as "Starting", "Holding", etc., but in other states this is not the case. In these states the starting conditions remain active, so the sequencer is started over again. However, this is usually unwanted and the user must consider this.

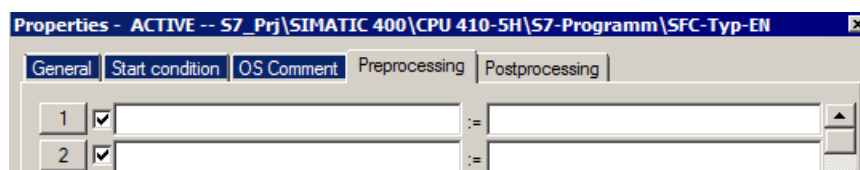
8.17 "Preprocessing"/"Postprocessing" Tab

You will find two examples of SFC types, along with their required transient-state starting conditions, in the SFC Library. These types are "TypeCtrlStrategy" (FB 1026) and "TypeStates" (FB 1025).



8.17 "Preprocessing"/"Postprocessing" Tab

For each sequencer, you configure the start condition and, optionally, the actions for pre-processing and post-processing.



These actions are executed as follows during cyclic execution of the SFC sequencer:

- Preprocessing: Prior to the initialization or processing or termination of a step
- Postprocessing: Following the initialization or processing or termination of a step

Preprocessing includes the actions to be executed in every cycle after the sequencer has started and before the steps and transitions are processed. Postprocessing are the actions to be executed in every cycle after processing the steps and transitions. This, for example, allows you to make presets or to pass on the results of the sequencer execution.

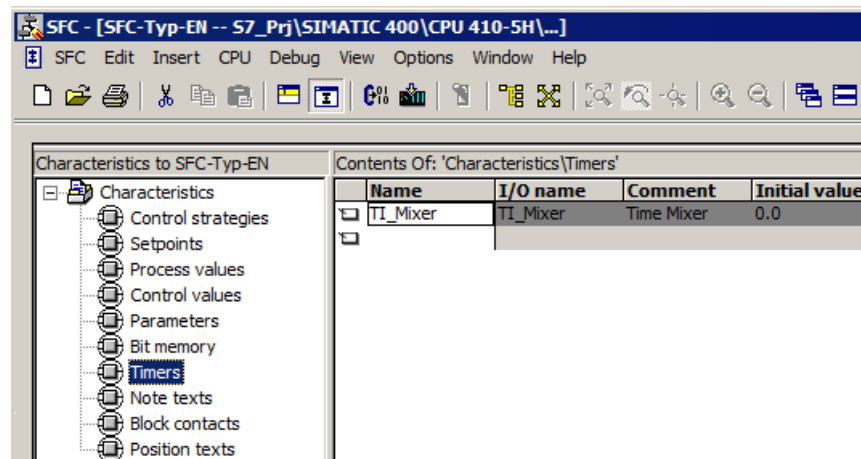
Notes, recommendations, and guidelines

9.1 Naming

Introduction

Adding a prefix to the connection elements when naming I/Os is recommended to make it easier to distinguish between the connection elements of the individual characteristic groups (setpoints, parameters, control values, etc.) of an EM.

A time element, for example, always starts with "TI_". A mixing time would be called "TI_Mixer", for example.



Examples

Characteristic	Prefix	Example
Control strategies	FW_ (CS_)	FW_ph_Temp
Setpoints	SP_	SP_mixing_time
Process values	PV_	PV_container_temperature
Control values	QV_	QV_enable_Manual
Parameters	IN_	IN_temp_hysteresis
Bit memory	FL_	FL_time
Timers	TI_	TI_mixing_time

9.1 Naming

Note

When assigning names, observe the recommendations regarding the number of characters given here.

When names, strings or texts are transferred to the AS or to AS blocks, they are truncated after 16 or 32 characters.

When assigning names, be aware that only the first 8 characters are visible in the CFC block view.

Attributes for characteristics

In section 11.7 of the "Process Control System PCS 7 SFC for SIMATIC S7" programming and operating manual, you can find a detailed description of the possible attributes and a list of the assignment of attributes to characteristics.

Meaning of the attributes

In section 11.7 of the "Process Control System PCS 7 SFC for SIMATIC S7" programming and operating manual, you can find a detailed description of the meaning of the attributes. Only the most important of these are presented below. Refer to the manual for more details.

Name

- No special characters apart from "_"
- No umlauts
- Maximum length: 16 characters (e.g. name of the SFC type)
- ID characteristic (visible designation for batch objects, e.g. for setpoints in master recipe and batch)

Data type

Permitted data types for characteristics are BOOL, INT, DINT, REAL, and STRING. The PI and PO data types can be used for the setpoints. These data types essentially represent a REAL setpoint and additionally contain the "Material" and "Tracking ID" attributes. You can assign enumerations to the data types DEST, SOURCE, VIA and TKEY (data types for Route Control).

Length of I/O name

- Setpoints and times: ≤ 16 characters
- Block contact: ≤ 10 characters
- All other characteristics: ≤ 24 characters

Note

When the interface is generated, suffixes are added to the names of the automatically created I/Os for setpoints, times, and block contacts. When selecting names, remember that only the first eight characters of all contacts can be viewed in CFC.

Long I/O names are only visible in full as tooltip texts.

To ensure that names remain distinguishable, unique and uniform, it is best to define a naming convention at the start of configuration work.

Comment

You can use comments to describe a characteristic in greater detail. The comment can contain a maximum of 80 characters and include any special characters.

Initial value

The initial value corresponds to the value of a characteristic when no actual value is available. The attribute can be changed in the SFC instance.

Unit

For the INT, DINT, REAL, PI and PO data types, a unit can be defined. The units are included as a basic set in the ES data management and can be added to or modified in SIMATIC Manager as "Shared declarations".

Text0 and Text1

- Relevant for the data type: BOOL
- Can be edited at the instance as the "S7_string1" or "S7_string0" system attribute
- Defined in shared declarations (max. 16 characters)
- Is not displayed in SIMATIC BATCH

Enumeration

You can assign an enumeration to the BOOL, INT, DINT, DEST, SOURCE and VIA data types. The enumeration is defined in SIMATIC Manager in the "Shared declarations". You can select the name of the enumeration for the attribute from a drop-down list.

9.2 Combining sequencers

The actions to be realized in each state must be defined for every control strategy of an EM. For example, if there are five control strategies and 12 SFC states, this results in a not inconsiderable number of sequencers (even when you take into account the fact that in some states no sequencers are implemented).

As an SFC type may contain a maximum of 32 sequencers, they must be combined in a sensible way. Sequencers can be combined in the following ways:

- Combination on the control strategy level
- Combination on the state level
- Combination of both methods

Operating Modes	Control Strategy 1	Control Strategy 2	Control Strategy 3	Control Strategy 4	Control Strategy 5
 IDLE					
 RUNNING					
 COMPLETING					
 COMPLETED					
 HOLDING					
 HELD					
 RESUMING					
 ERROR					
 HELD(Error)					
 RESUMING(Error)					
 ABORTING					
 ABORTED					

Grouping of a sequence on OSL phase

Grouping of a sequence on equipment phase

Combination on the control strategy level means that there is a sequencer for every control strategy, with branches to the different states within this sequencer.



A rigid combination according to control strategies or states does have its disadvantages and cannot be sustained if RUNHOLD = FALSE (resumption if a step is held in "RUN"). It would make more sense to use a combination of the two methods.

Combine the states containing sequences specific to a control strategy (e.g. "STARTING", "RUN", etc.) in a control-strategy sequencer. States which contain the same sequence in every control strategy (e.g. "ERROR") can be combined to form a sequencer. The same sequence is often performed in different states (e.g. "Completing" and "Stopping").

9.3 Editing within the project

SFC types can be created in projects as well as in libraries. It is recommended that the SFC types be created in the master data library and distributed from there to the projects.

9.4 Block size of an SFC type

The block size of an SFC type depends on many factors and their combinations, such as the number of setpoints, control values, sequencers, steps, whether many REAL or BOOL setpoints are used, etc.

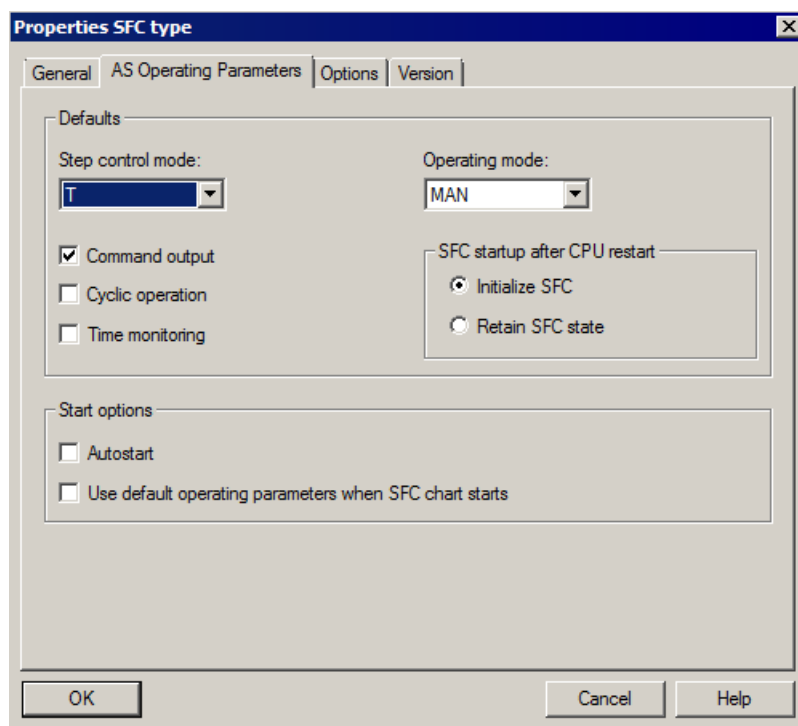
In theory, it is possible to make calculations in advance using a formula contained in the SIMATIC PCS 7 documentation. The system resource requirements can be determined using the Excel worksheet, "SFC-Resources-V8-enEN.xls or SFC-AdditionalResources-V8-enEN.xls". Because the necessary details (e.g. number of steps, setpoints) are often not known at the outset, pre-calculations are often not possible.

These documents can be found under the following path: C:\Program Files (x86)\SIEMENS\STEP7\s7sfc

The maximum block size is 64 KB. The actual block size of an SFC type can be determined using its FCs in the block folder. The compiler optimizes internally and in case the 64-KB limit for action FCs is exceeded, another action FC is concatenated internally. However, this is not possible for instance DBs.

9.5 SFC type following CPU STOP/restart

A setting can be made for every SFC, specifying whether it is initialized following a CPU restart (the approach in SIMATIC PCS 7 versions lower than V7.0) or whether the SFC continues to be processed from the interruption point (the step which was active prior to the CPU STOP).

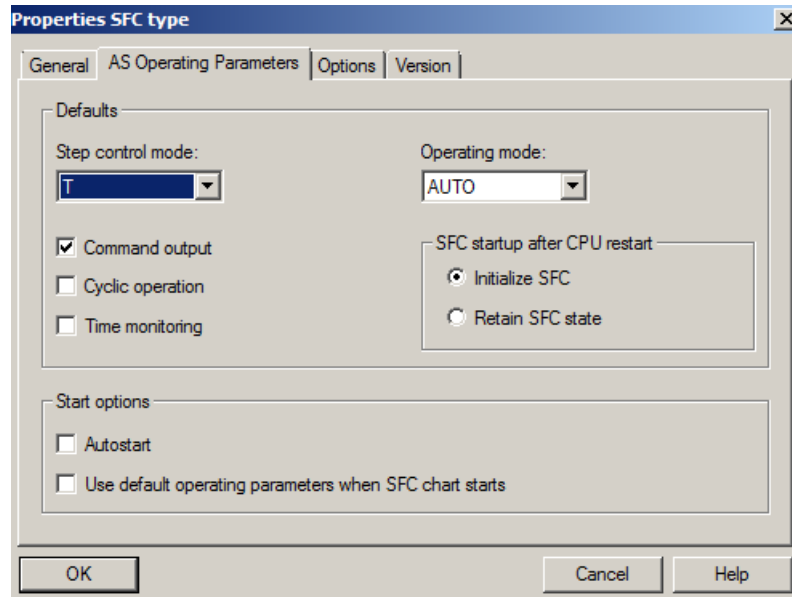


Note

You can find additional information in the FAQ "How can error outputs be reset without starting the productive step sequencer?" (<https://support.industry.siemens.com/cs/ww/en/view/48933314>).

Example: "Initialize SFC" startup characteristics

Scenario: The CPU has been stopped and the startup characteristics of the SFC have been set to "Initialize SFC".



Result: The SFC is initialized following a CPU restart, i.e. the operating state is "Ready" (IDLE) and all steps are initialized. The "AUTO" or "MANUAL" mode remains unchanged.

In the "Ready" operating state, a starting command is possible in the following modes:

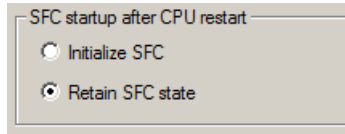
- In MANUAL mode, the "Start" command
- In AUTO mode, the SIMATIC BATCH starting command

SFC visualization operation in the OS: For manual operation, the operating mode must be switched from AUTO to MANUAL. Only the "Start" function is possible. The SFC is executed again.

Operation of the BATCH batch: It is possible to specify when to continue, terminate or stop an interrupted batch following manual operation. The batch can also be resumed during a different step.

"Retain SFC status" startup characteristics

Scenario: The CPU has been stopped and the startup characteristics of the SFC have been set to "Retain SFC status".



SFC visualization operation in the OS: Following the CPU restart, the SFC is visualized in its current state. The last active step before the CPU stop is highlighted. The SFC waits for manual operation by the user. The operating mode must be switched from AUTO to MANUAL. The Resume, Abort or Stop functions are possible. Further procedures can be specified according to your process.

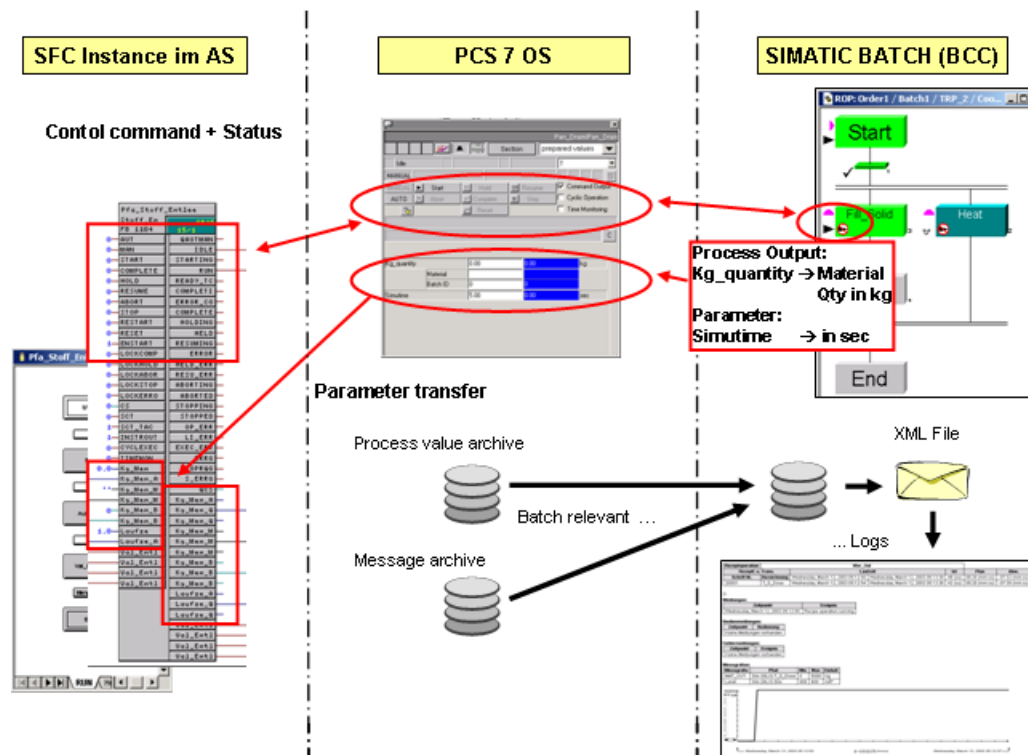
Operation of the BATCH batch: It is possible to specify when to continue, terminate or stop an interrupted batch following manual operation. The batch can also be resumed during a different step.

9.6 Connecting to SIMATIC BATCH

For an SFC type, the connection to SIMATIC BATCH is carried out directly via the definition of the SIMATIC BATCH category (see section "EPH and EOP (Page 68)"). SIMATIC BATCH is able to directly read out the configured batch characteristics (control strategies and setpoints) of an SFC type as well as the associated SFC instances. This means that no additional interface blocks are needed for an SFC type.

Communication/flow of data for the control commands, status and parameter transfer is performed via WinCC variables.

The figure below shows a schematic view of communication or the interaction between the SFC type and SIMATIC BATCH – the so-called PC-based mode.



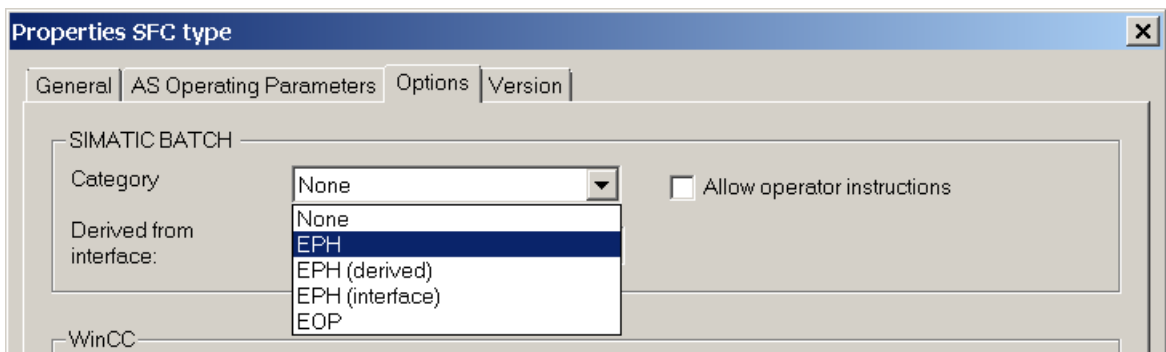
From SIMATIC BATCH version V7.1 SP2 onwards there is also the so-called AS-based mode. In this new mode, SIMATIC BATCH is also able to process the characteristics (control strategies and setpoints) of a type configured in the SFC type and the associated instances.

9.7 EPH and EOP

When creating SFC types, you can set whether this SFC type is used for SIMATIC BATCH (category) under Properties (in the Options tab).

As of SIMATIC PCS 7 V9.0 you can additionally select the following categories for the SFC type (more details also in section "Settings and configuration in SIMATIC Manager" (Page 157)):

- EPH
The name of the BATCH recipe phase is identical to the name of the SFC type
- EPH (derived)
The SFC type is derived from an SFC type with the BATCH category "EPH interface".
- EPH (interface)
An SFC type declared in the BATCH category as an "EPH interface" serves as a base class and thus as a generic interface for the integration of the recipe phase (RPH) of SIMATIC BATCH. An SFC type with the BATCH category "EPH interface" only contains an interface description with setpoints, control strategies, etc. but no implementation. The programming of sequencers with the steps and transitions is no longer offered by the SFC editor in this case.



The difference between EOP and EPH becomes apparent in the SIMATIC BATCH recipe editor. If EPH is selected, a corresponding RPH (recipe phase) is generated in the BATCH system. The recipe phase can be used in the SIMATIC BATCH recipe editor as part of recipe operations.

If EOP is selected, a closed recipe operation which can be used on the operation level in recipes is generated on the recipe level.

If the SFC type should not be used for SIMATIC BATCH, then select the setting "None".

9.8 Creating several instances of a type within a unit - multi-instances

From SIMATIC BATCH version 8.1, multiple instances of the same SFC type can be addressed within a unit. In order to distinguish between several block instances of an SFC type within a unit, the block instance names have to be different.

As soon as there are "multi-instances" somewhere in the project, the project completely switches to multi-instance mode and all instances are affected by the switch.

Since version 9.0, there is additional S88 categorization "EPH-derived". In multi-instance mode, the name of the PH folder is used to derive the name.

More details are shown in the section "EM and EPH in the context of SIMATIC BATCH" (Page 157).

9.9 Closing lockout for SIMATIC BATCH

It is possible to disable or interlock the start of an equipment module from SIMATIC BATCH via input "BA_EN = FALSE". As a result, the control recipe (batch) goes into an error state.

If the EM is then enabled for SIMATIC BATCH (BA_EN = TRUE), the recipe procedure can continue to be executed with "Resume".

9.10 "Continuous" functionality

The Continuous function is intended for recipe creation and batch control in SIMATIC BATCH.

"Continuous" is frequently used for non-self-terminating phases (e.g. mixing).

Non-self-terminating phases are not terminated by conditions (e.g. level reached), but rather by operator input or a higher-level recipe system (see the section Self-terminating and non-self-terminating equipment modules (Page 51)).

In a recipe system, however, it is not always required to switch off a mixer, for example, between two recipe steps; you may instead want to let the mixer continue running at a different speed.

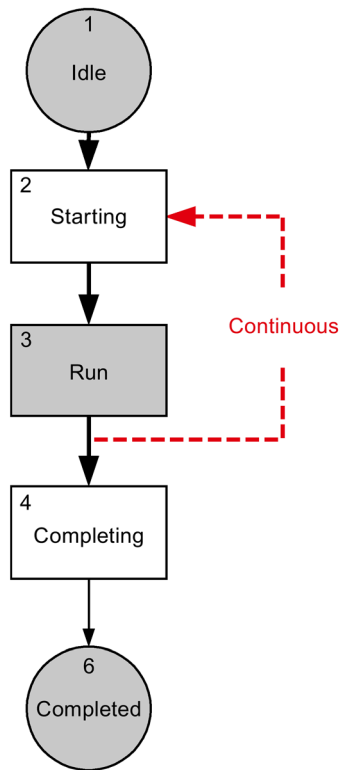
The behavior of the state logic must be taken into account during configuration in order to implement this function.

If the "Continuous" function is selected in the recipe, the next time the phase is started from within the recipe it will not pass through the transition states "Completing" [4], "Completed" [6], etc., until it reaches "Run" [3]; rather it will switch directly from "Run" [3] to "Starting" [2]. You must take note of the transition states or sequencer steps in which the motor for the mixer, for example, is switched off.

The "ENASTART" input on the block must also be set to TRUE so that the SFC can be started again.

9.10 "Continuous" functionality

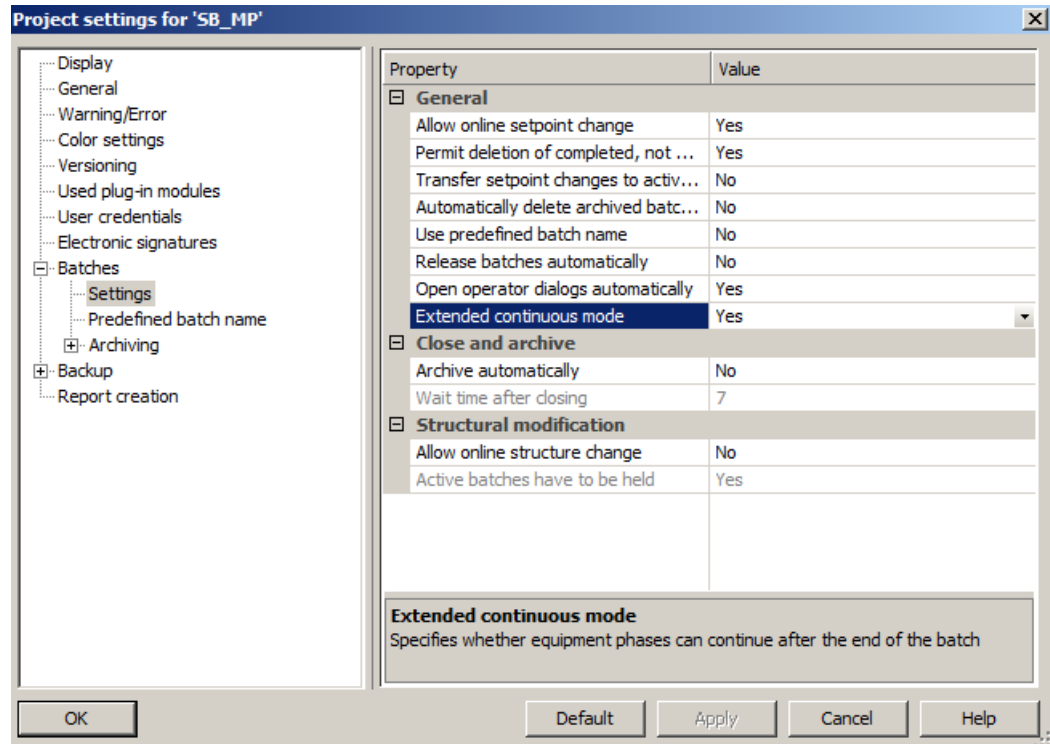
The Continue function is only available for non-self-terminating phases.



9.11 "Extended continuous mode" functionality (SIMATIC BATCH)

The continuous operation of equipment phases has been extended as of version V8.2 of SIMATIC BATCH. In the project settings, you can specify whether continuous or extended continuous operation is used.

Until now, equipment phases have been completed automatically by the end of the batch at the latest in continuous operation and the unit enabled. In extended continuous operation the equipment phases can be assigned to other batches even once the unit has been completed and enabled. The equipment phases retain their "RUN" status.



Introduction

This section deals with the technological configuration of equipment phases (EPHs) and equipment modules (EMs) using control modules.

The technological configuration is a new description language that enables configuration abstracted from the concrete detail implementation, especially for higher-level engineering systems.

From a higher level point of view, control modules are only considered as instances of a CM type, in other words only the instance-specific, i.e. individual, characteristics are externalized and the inner logic is treated like a black box. This also reduces the required description of an instance to the referenced type, individual parameterizations and individual interconnections.

This is done in the same way for equipment modules (EM) and equipment phases (EPH). Instead of interconnecting individual block I/Os, a CM instance fulfills a role from the point of view of the EM and the individual interconnections, which are of course still necessary, are performed automatically in the background.

Similar to an object-oriented language, commands are defined within the CM types, for example, which can then be used in sequential control systems.

A clear and concise structure is achieved through this modularization, typification and abstraction and enables the functionality to be decoupled from the actual implementation. As long as the actual functionality does not change but is implemented in a different way, it is irrelevant for higher-level engineering systems.

Neither equipment phase types (EPHTs) nor equipment module types (EMTs) are to be understood as a substitute for SFC types. Instead, they use the proven SFC type instance concept internally. An EPHT or EMT is like a CMT that contains exactly one instance of an SFC type. Other blocks may also be contained.

The majority of the technological configuration can be completed exclusively in the master data library (MDL). For this reason, the fields displayed in the editors for types within the MDL and the instances in the projects are different.

The most important terms are listed in the following table:

Term (English)	Abbreviation	Term (German)
Control module*	CM	Einzelsteuereinheit
Control module type	CMT	Einzelsteuereinheiten-Typ
Equipment module*	EM	Technische Einrichtung
Equipment module type	EMT	Typ für technische Einrichtungen
Equipment phase*	EPH	Technische Funktion
Equipment phase type	EPHT	Typ für technische Funktionen
Recipe phase*	RPH	Rezeptfunktion
Process action	-	Prozessschritt

* Internationally standardized terms

Technological configuration at a glance

To start, the CMTs are created in the master data library. This also includes the commands and states.

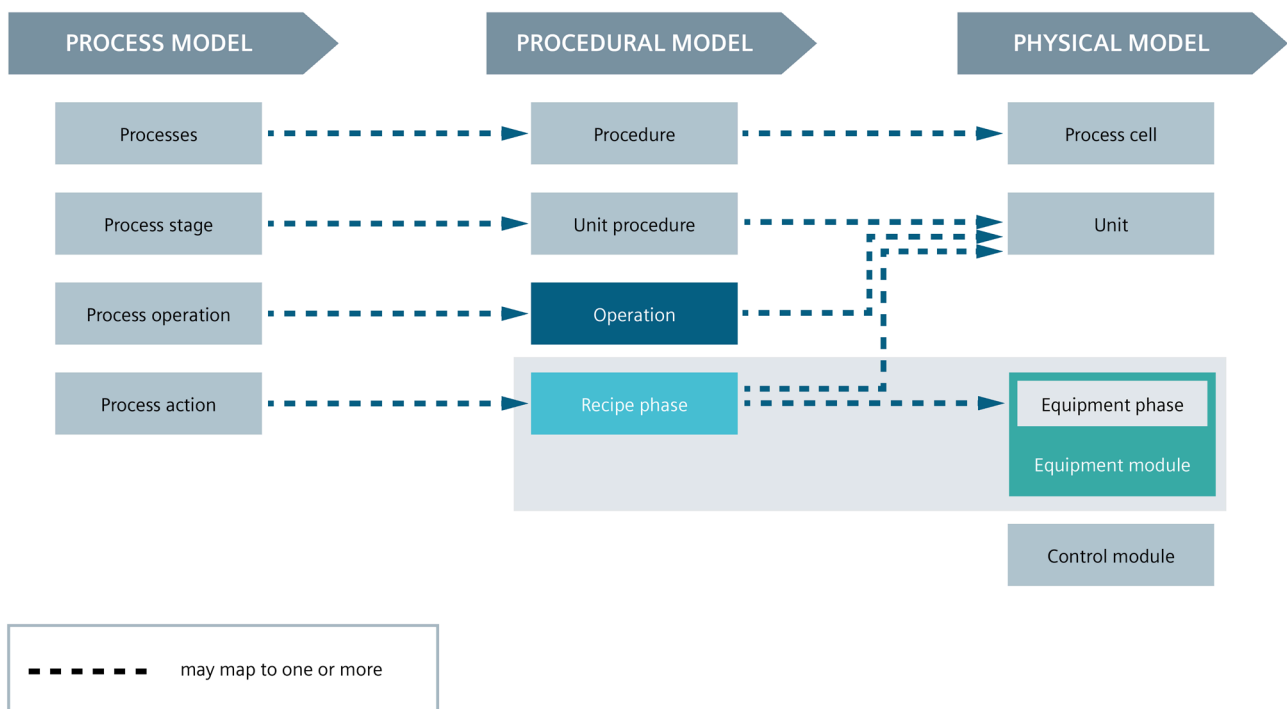
Then, EMTs are created, in which case CMT roles are used as proxies for the CMs to be controlled later. Instead of individual, partially cryptic assignments, these are summarized as readable commands. The commands and status of the CMTs as well as the SFC library can be accessed when configuring the sequencers.

Instances of the CMTs and EMTs are generated in the projects manually or with bool engineering tools.

10.1 Overview of EPH and EM

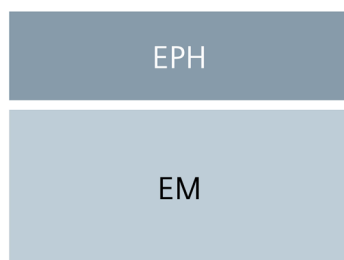
The terms EPH and EM are defined in the ISA-88 standard. An EM describes a functional group of equipment items/parts of a plant that can perform a limited number of smaller process activities. Usually, subordinate control modules are unambiguously assigned to an EM for the most part. The EM then controls the CM assigned to it. An example would be a reactor heater for which valves and pumps are controlled and analog and digital signals are evaluated.

It is assumed in this document that the EM is implemented in the form of an SFC type and the CM is controlled using an SFC type.

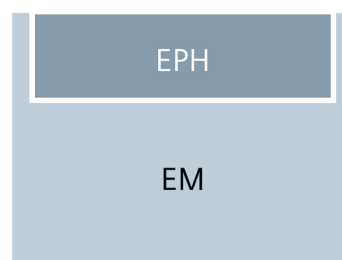


A phase forms the lowest level in the procedural model of ISA-88 and can perform an entire process action or only part of a process action. It is thus a coordinated procedure.

The equipment phase (EPH) that is examined here in more detail forms the counterpart to a recipe phase (RPH). It is controlled directly by the recipe system and must therefore provide a compatible command/status model. This is ensured by the operating state logic of the SFC. The equipment phase controls the EM assigned to it in a coordinated manner.



EPH und EM separat



EPH im EM integriert „recipe aware EM“

If you integrate one or more phases in an equipment module, this is designated in ISA-88 as "recipe aware". The control strategies of an equipment module then correspond to the phases. This has been the approach implemented in SIMATIC BATCH for years.

10.1.1 EPH and EM in the SIMATIC PCS 7 environment

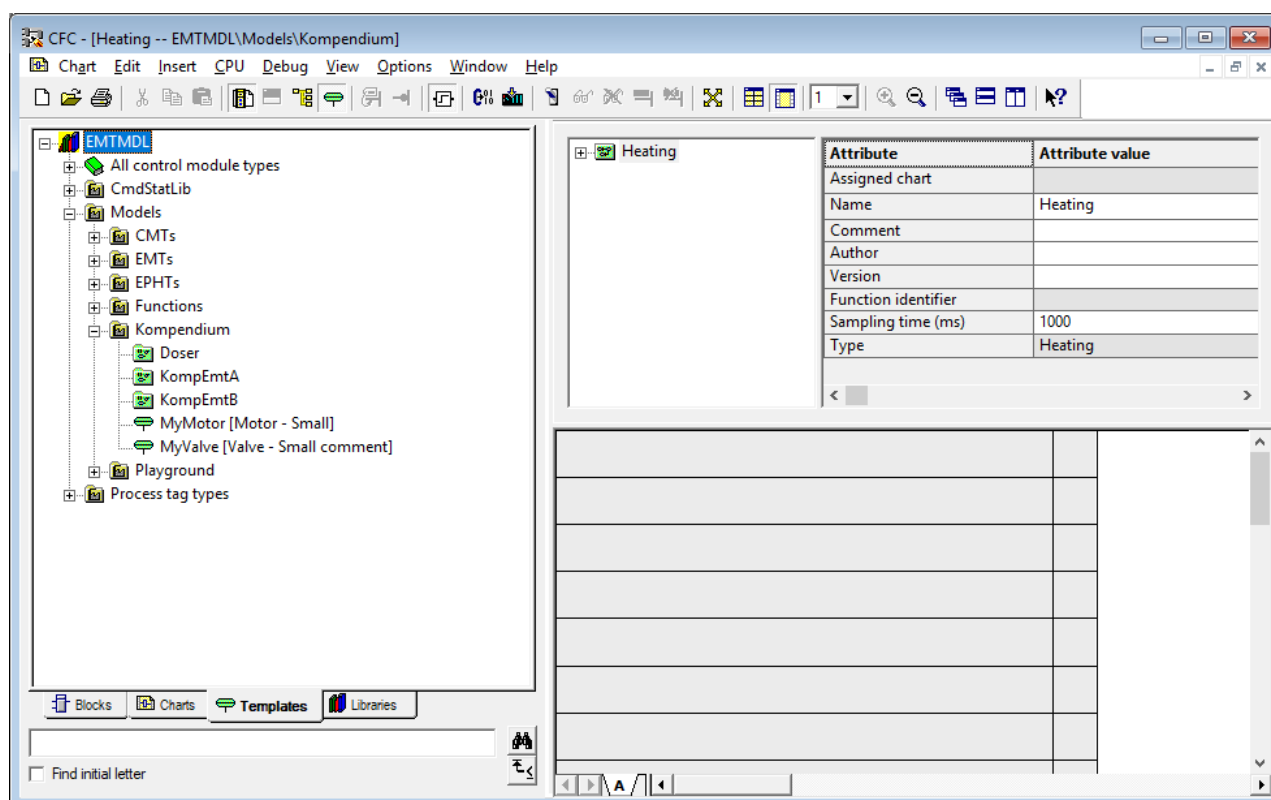
In this document, EMs are always implemented on the basis of the SFC type/instance concept. That is, an equipment module always controls actuators and sensors using a step sequencer even if other ways of doing so would be conceivable.

The object of the equipment phase is also implemented using an SFC. In so doing, the state logic provides the exact states and state transitions that ISA-88 requires. Other advanced features of SIMATIC BATCH are also supported.

10.1.2 CFC catalog expansion

For the technological configuration, a "Templates" tab has been added to the catalog in the CFC editor.

All technological objects such as CMT and EMT are available there.



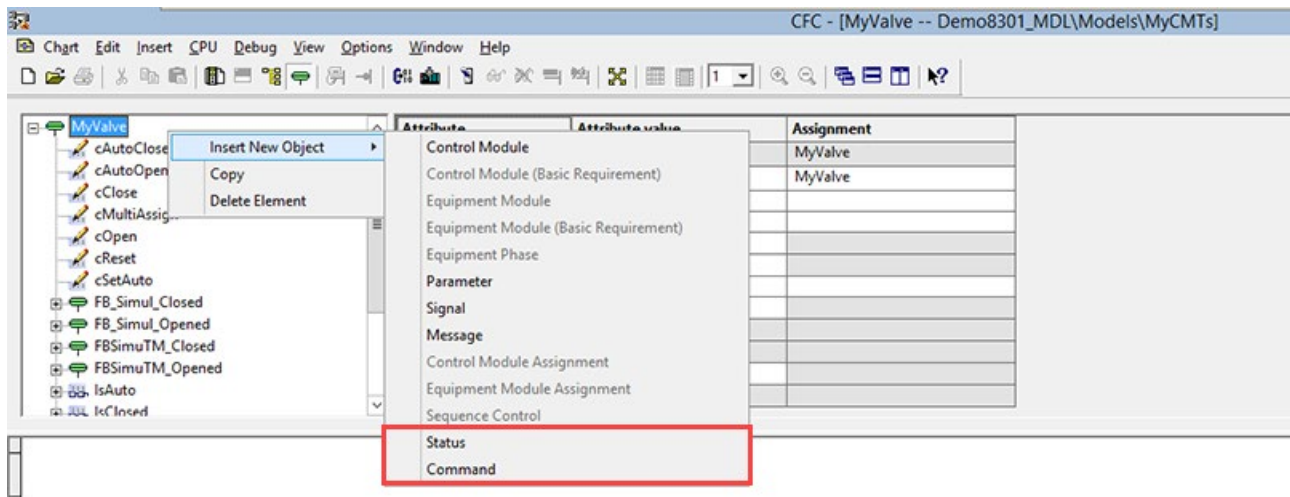
10.2 CMT - Status and commands

Refer to the documentation (e.g. section "Service and Support" (Page 164)) in regard to the configuration of control module types (CMTs). Only the commands and states needed for the EMT or EPHT configuration will be explained here.

Commands and states are created in the technological I/Os. In so doing, states are symbolized by a pair of glasses and commands by a pencil.

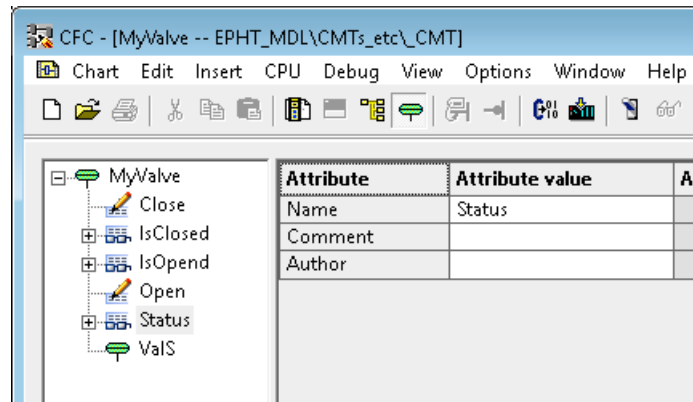


They can be created in the technological I/Os using the shortcut menu of the respective CMT.

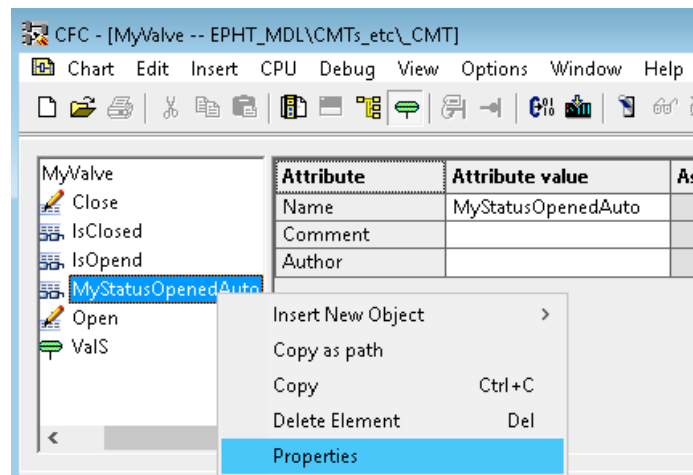


10.2.1 CMT - Status

A status is a logic operation of comparisons and returns a TRUE or FALSE result. All I/Os of the technological block with the "s7_contact" attribute and constants can be used for the comparisons.

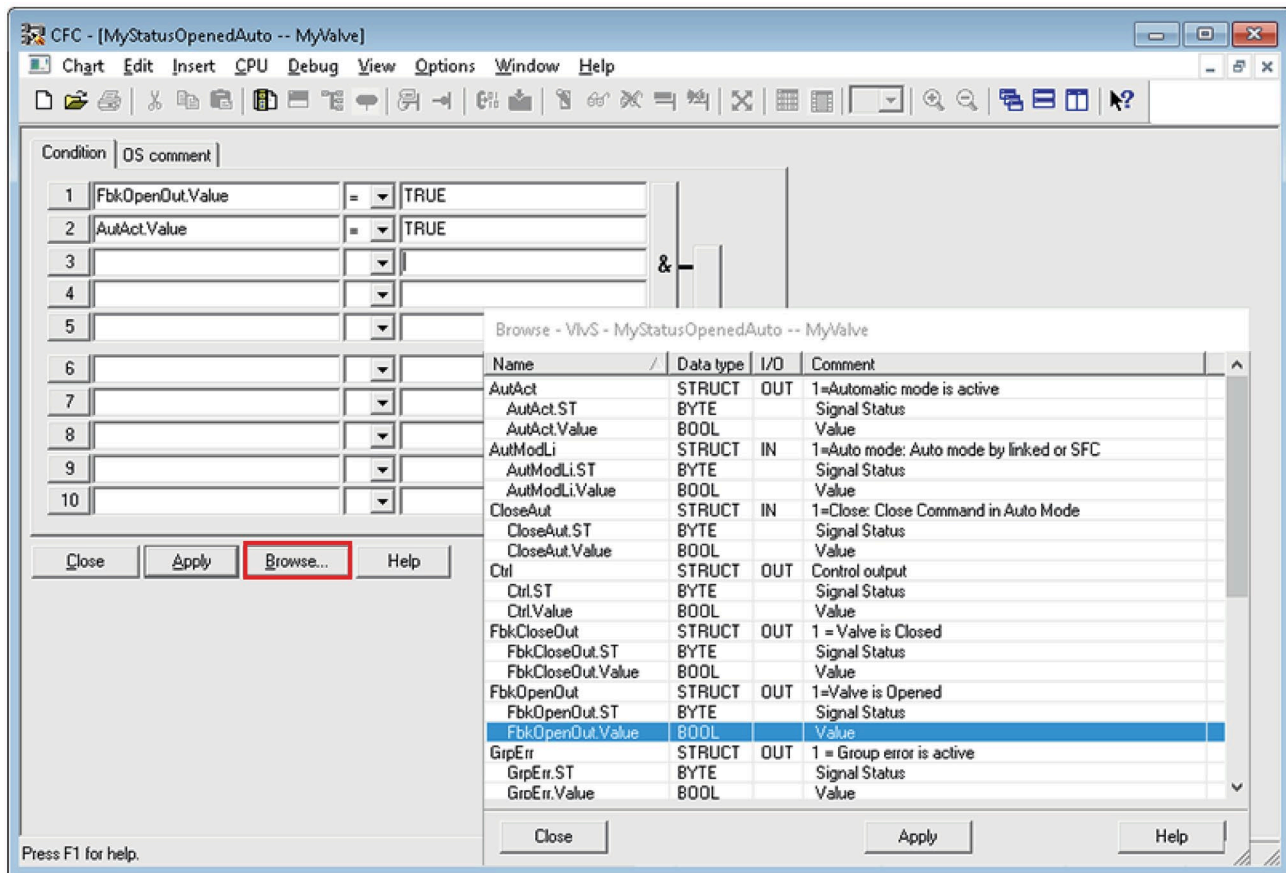


Status and commands are to be descriptively named.



After being named, their properties can be configured. An editor for the link logic is opened via the shortcut menu of the respective status or command.

All I/Os available for the configuration are displayed with the "Browse" button and can be selected by double-clicking or using drag-and-drop.



Note

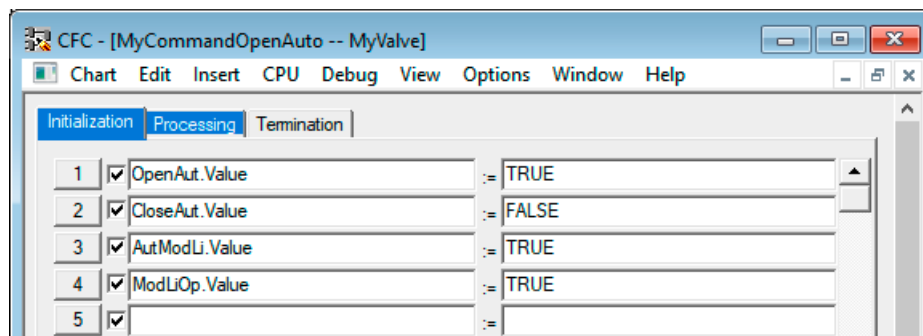
In current versions, a CMT must or may contain only one technological block with the "s7_contact" attribute, provided that technological configuration is to be performed.

If multiple technological blocks with the "s7_contact" attribute are contained, only the first block found is used.

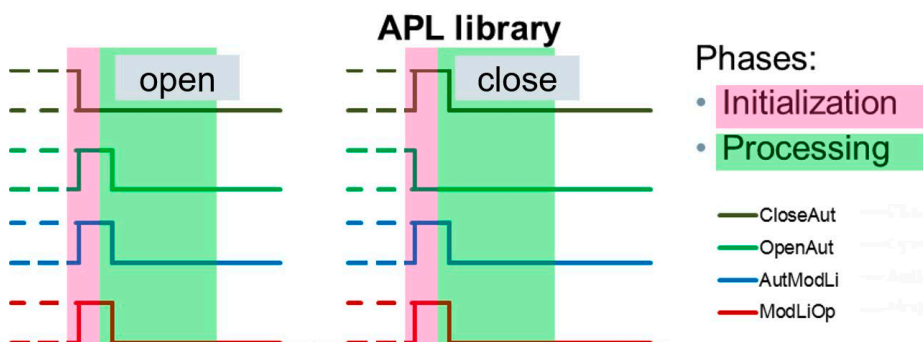
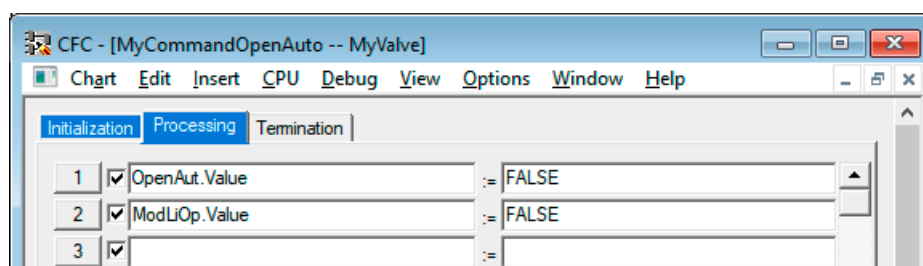
Arithmetic or logic operators and functions are not supported.

10.2.2 CMT - Commands

Numerous value assignments to I/Os of the technological block can be grouped using commands.

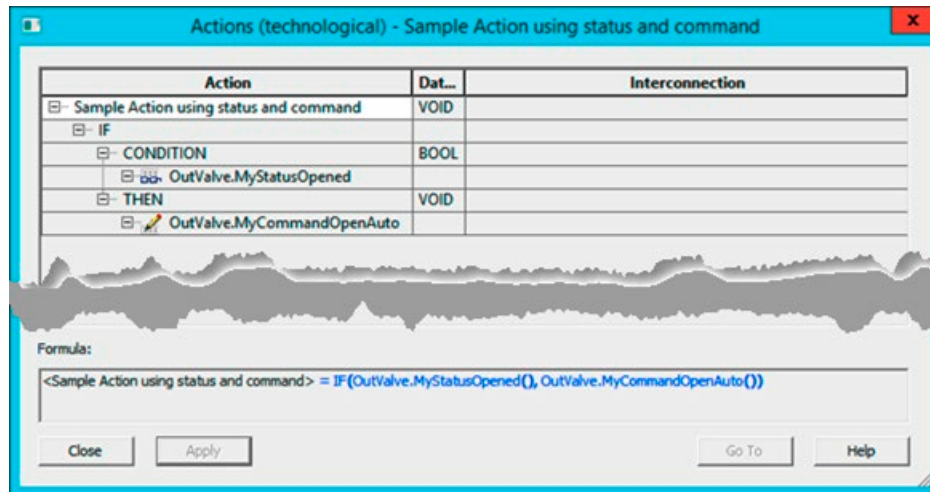


The operating phase of the SFC step (initialization, processing, termination) in which an assignment will be made can also be specified.



10.3 EPH types and EPH instances

These commands and states can be used later in the technological actions of the EPHT and EMT, as the following example shows:



Note that commands are not macros. Each time a command is used, the implementation - i.e. the instructions in the case of commands - is built into the code "flat".

10.3 EPH types and EPH instances

In the following, the equipment phase type (EPHT) is first explained to help in understanding the SFC type in the SIMATIC BATCH environment. The statements also apply in the same way to EMT. Any differences are noted.

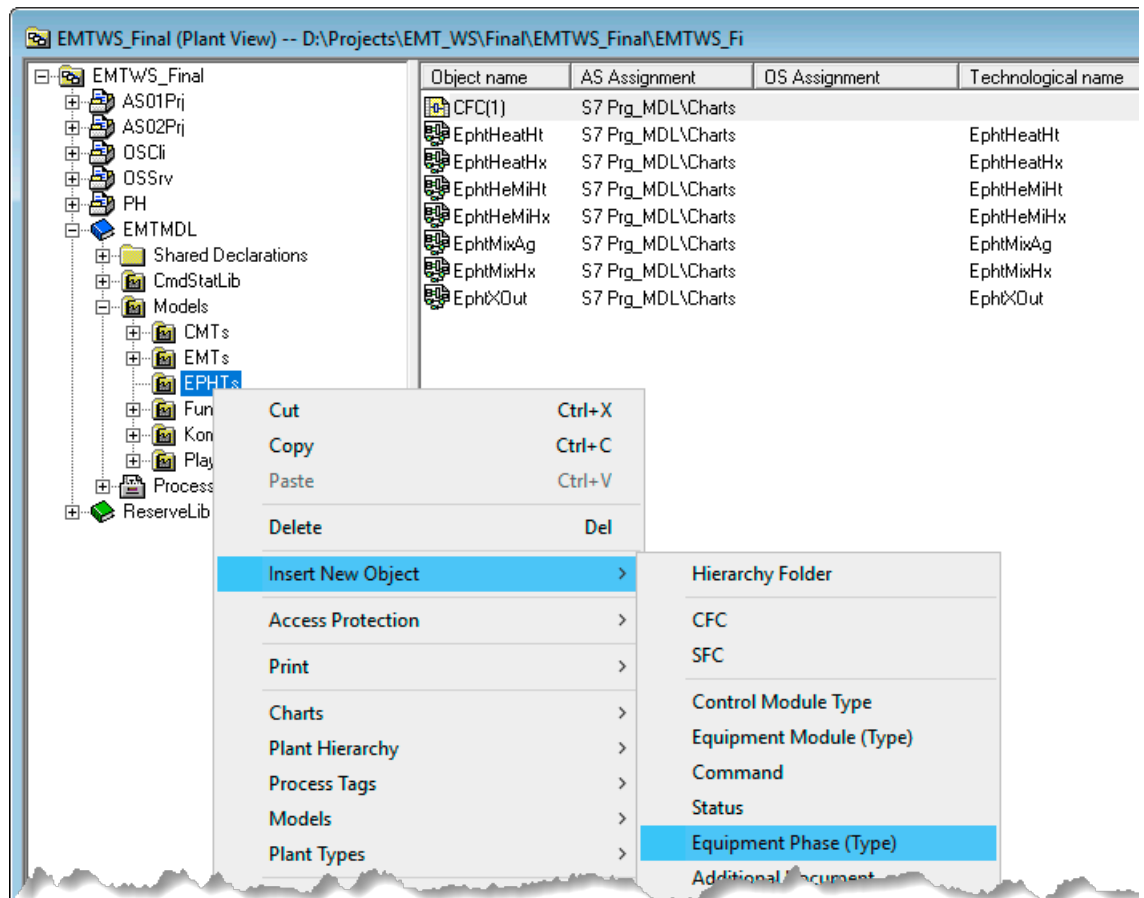
Likewise, EM types can be converted into EPH types and vice versa in the master data library.

The use of EPH types or EM types currently has no influence on the classification for SIMATIC BATCH. This is exclusively determined by the categorization of the SFC type and the folder within the plant hierarchy. Although formally separated, a clear structure here should be aimed to ensure intuitive understanding of the implementation.

10.3.1 Creating EPH types

A new equipment phase type is created within the master data library. For a better overview, creation of separate folders for the respective object types (here "EPHT") is recommended.

A new EPHT can then be created using the shortcut menu of the "EPHT" folder in the plant view. To do so, select the "Equipment Phase (Type)" subcommand in the "Insert New Object" menu ("Insert New Object > Equipment Phase (Type)").

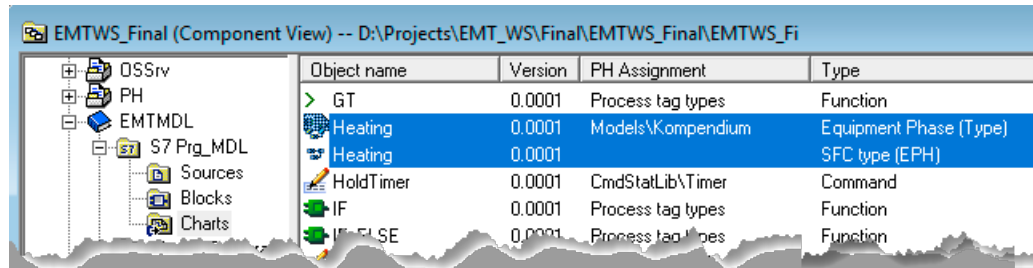


Enter a descriptive name (e.g. "Heating").

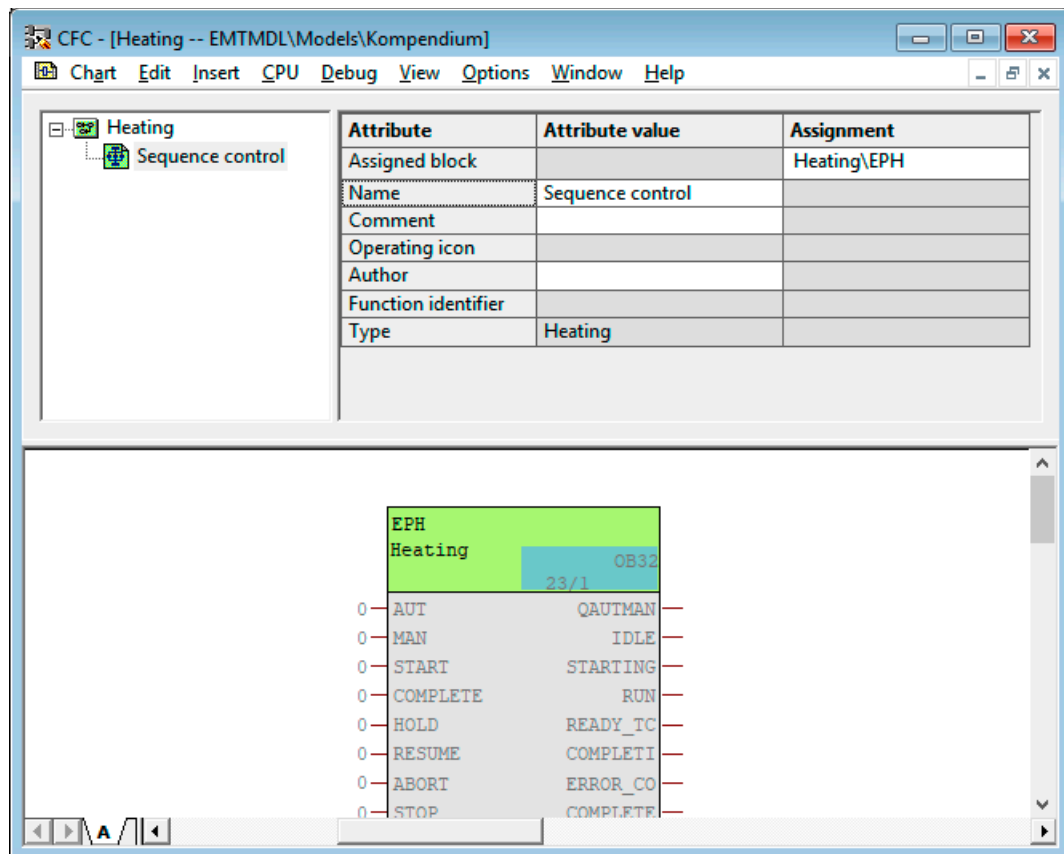
When an EPHT or EMT is created, the associated SFC type is automatically created. Since V9.1, it is automatically given the same name as the EMT or EPHT. If the name is changed, the name of the partner object is immediately adapted, i.e. if the name of the SFC type is changed, the name of the EPHT is also immediately changed.

10.3 EPH types and EPH instances

Duplicate object names are a special case in this regard; they are recognized by the SIMATIC Manager and made unique by a suffix such as "(1)". In these situations, if the EPHT/EMT is renamed at a later time, the name change may not be inherited and must later be changed manually. From then on, there is again a direct relationship between the EPHT and the SFC type. When upgrading projects, the name identity should be verified and corrected if necessary.



The EPHT can be opened by double-clicking it or from the shortcut menu:



The chart belonging to the EPHT can be expanded as desired with additional blocks, which can have the "Optional" identifier as variant.

10.3.2 Technological actions, commands, states and conditional commands

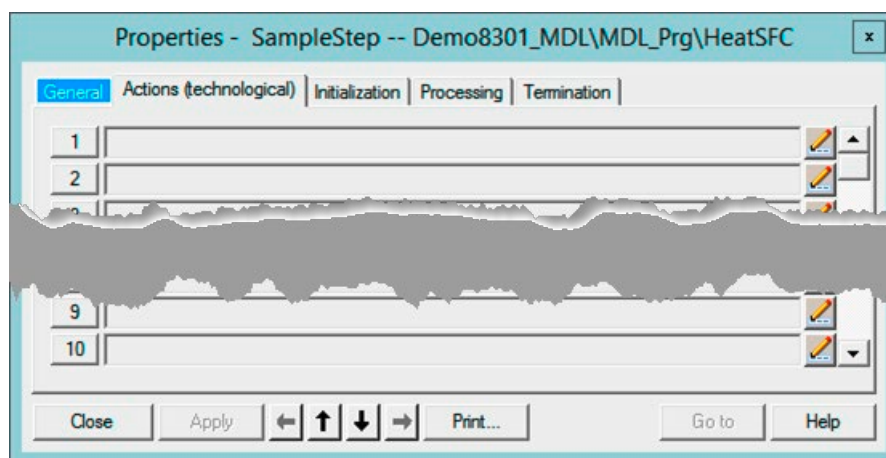
For configuring the SFC steps, the SFC type belonging to the EPHT must be opened in the master data library. Two options have been added to the configuration dialogs of the SFC type for the EPHT and EMT contexts:

- "Actions (technological)"
- "Conditions (technological)"

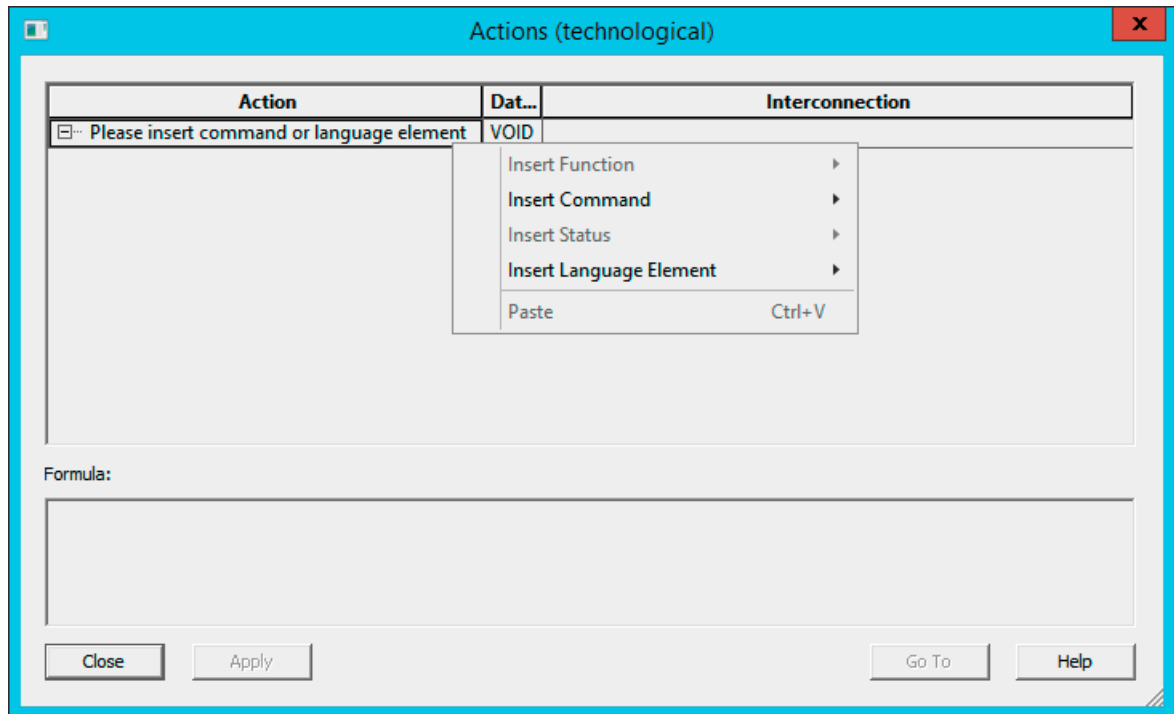
Technological actions are only available in steps and sequencer preprocessing/postprocessing, while technological conditions can be used both in transitions and in IF or IF-ELSE language elements as well as for the start conditions of a sequence.

Technological actions

Up to 50 technological actions can be configured in the "Actions (technological)" tab, whereby multiple commands can also be output within an action.



After you click on the pencil symbol to the right of an action line, the editor for configuring the action opens.



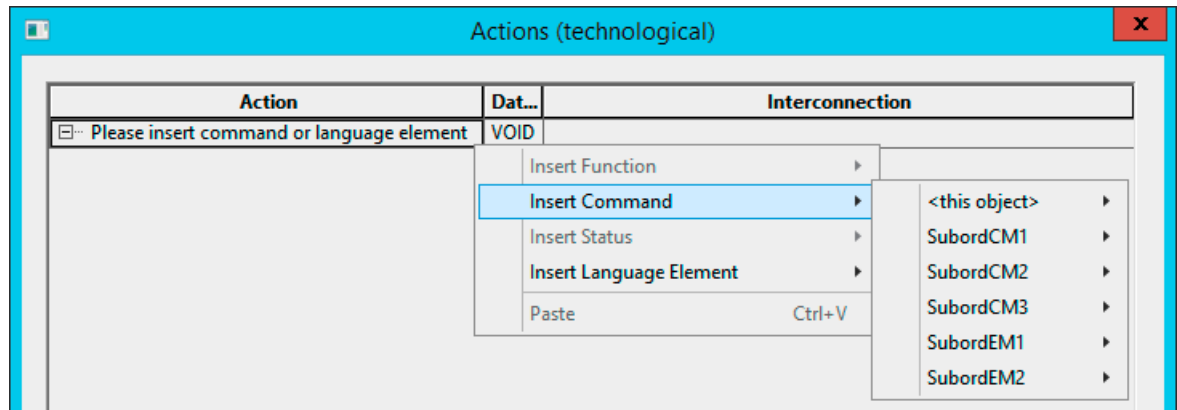
Language elements and direct commands can be inserted using the shortcut menu. The "STATEMENT_LIST" objects and an "IF" or "IF ELSE" branch are available for selection as language elements.

This enables commands to be grouped in one action or commands to be executed under defined conditions. Any combinations and even nestings are possible.

Internal commands for controlling the SFC ("this object" node) as well as object-related commands of lower-level objects are offered as commands. Thus, for example, lower-level control modules such as valves can be controlled, but complete equipment modules can be controlled, as well.

In the following figure, commands are offered for the custom SFC ("this object"), for the three lower-level control modules ("SubordCM1, ..2, ..3") and for the two lower-level equipment modules ("SubordEM1, ...2"), which are displayed after selecting the object.

In the example below, three CMs and two EMS belong to the SFC.



Name of a technological action

A name is generated automatically for the action depending on the command used first. This suggested name can be arbitrarily adapted but should be meaningful to enable the future plant operator to quickly and intuitively understand the given situation.

The name of an action must be unique within the entire sequencer (not the entire SFC). The system corrects non-unique names by adding a suffix.

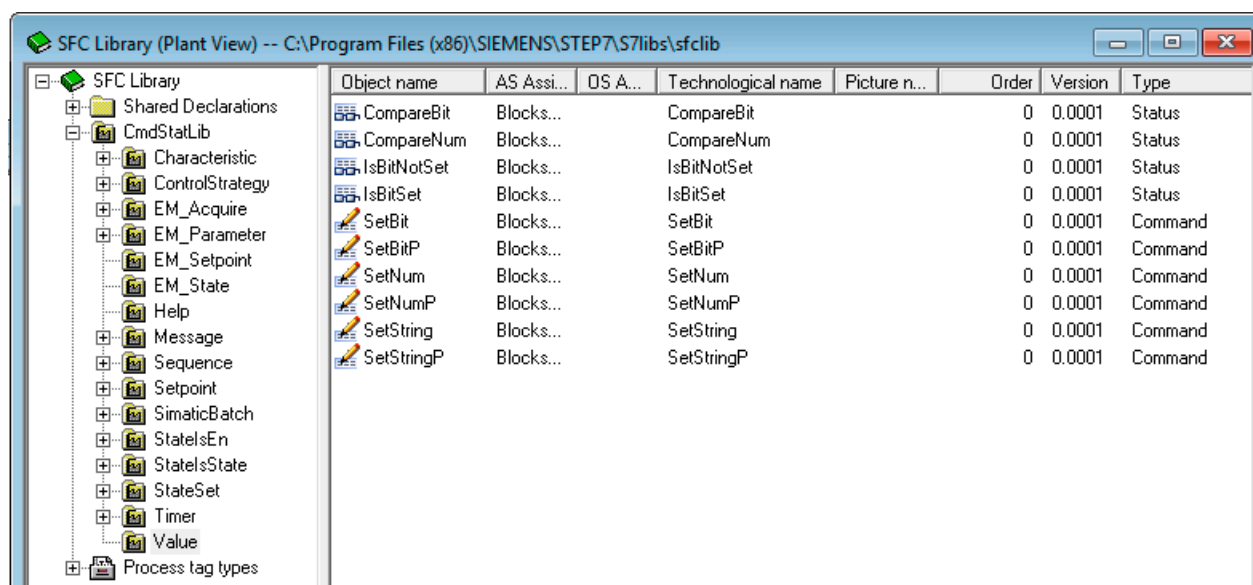
Note

When synchronizing technological types, only the implementation of commands and states in the master data library (MDL) is compared or adapted but not in the projects. Updating of the SFC types in the projects is performed from the master data library using the "Update Block Types" function.

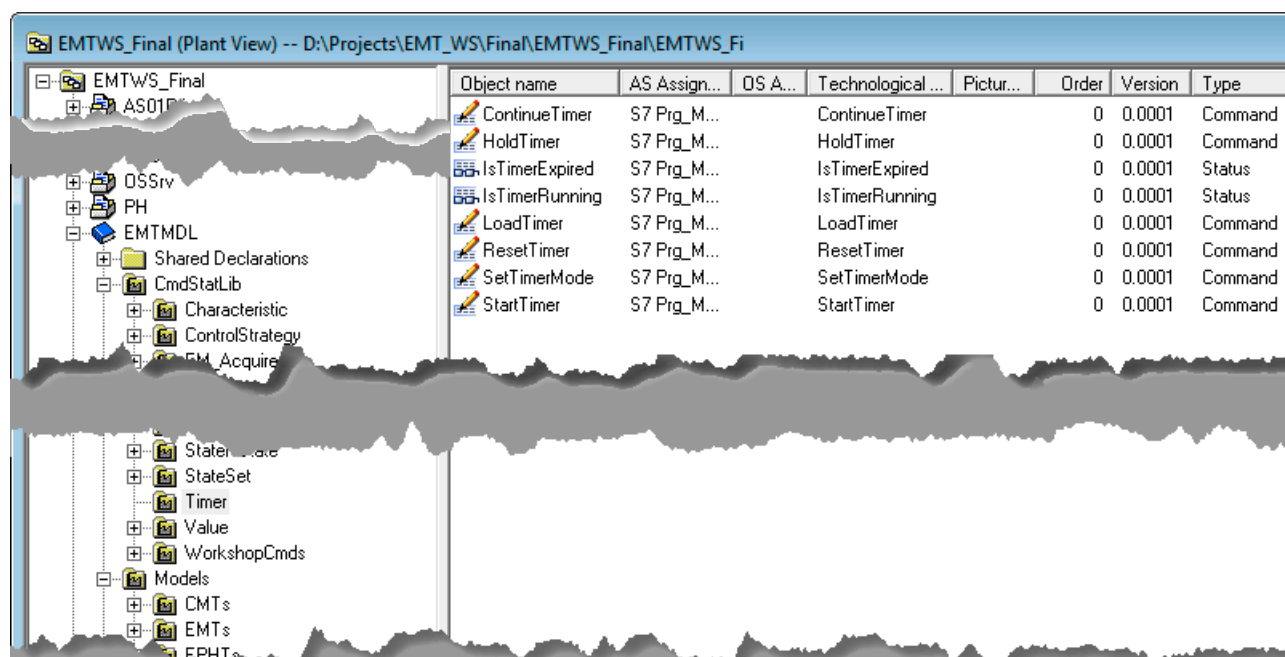
10.3.3 Internal commands of the SFC

Preparations (SFC library)

The internal commands for controlling the SFC are not permanently coded in the program package but rather are implemented as part of the SFC library. In this way, commands can be modified and new commands can even be created.



To do this, open the current SFC library in the plant view in SIMATIC Manager and copy the complete "CmdStatLib" folder (short for Command and Status Library) to the hierarchy folder of the master data library of the project.



All listed commands are then available. The chosen distribution of the commands among subfolders is also echoed later in the configuration and allows more structured working. These are commands and states for the SFC itself ("this object") and also for control and parameter transfer to lower-level EM-SFC. These can be found in PH folders starting with "EM_...", which are then offered when selecting the respective lower-level EM, here in the example "SuborEM1" (detail are available in the section "Commands and states for controlling lower-level EMs (Page 124)").

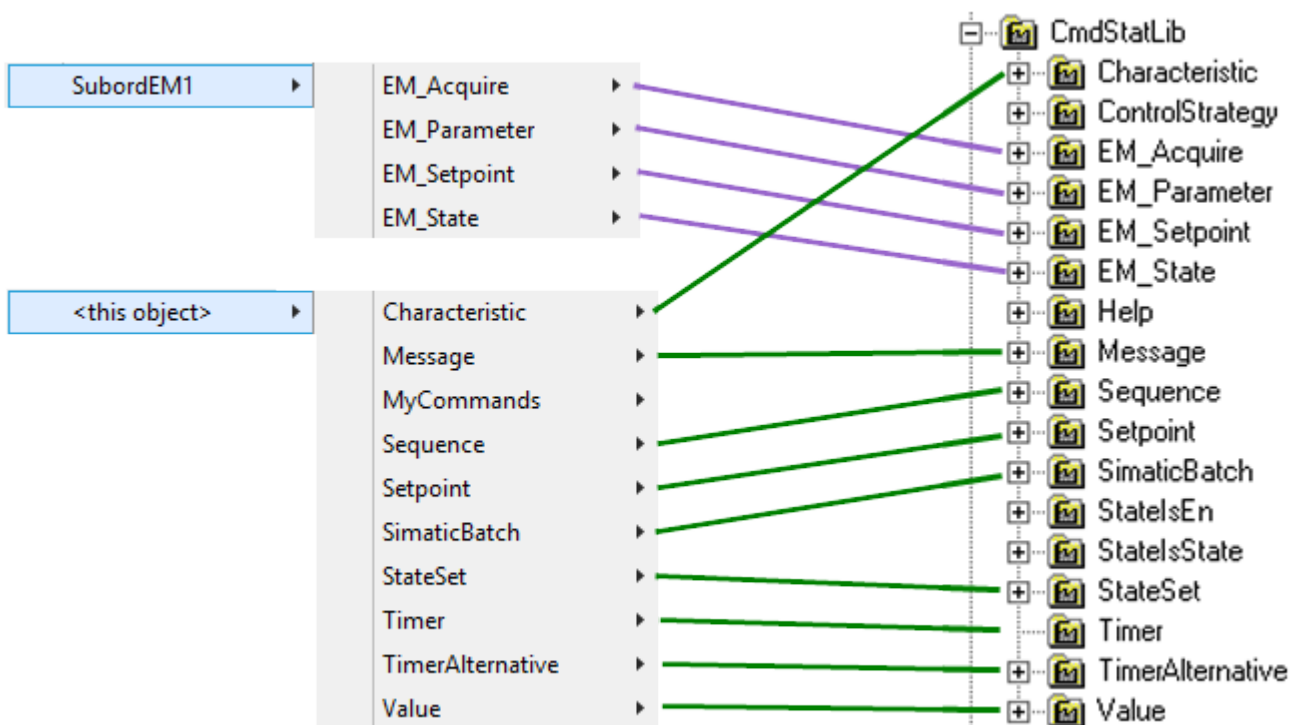
The following enumerations must also be copied from the Global Declarations / Shared Declarations:

Compare_Operator

ES_AcquireStates

ES_AcquireTypes

The values and names must not be changed because these hard-coded values are used internally and the enumeration is only used for a more intuitive and easy to read configuration.



Examples of internal commands are:

- "SetInternalCMDHold", for holding the SFC
- "GenerateMessage4", for outputting the ALARM_8P Message No. 4
- "ShowNoteText", for displaying a comment in the faceplate
- "HoldTimer", for holding the timer

The technological configuration is compared with the existing configuration below using two examples.

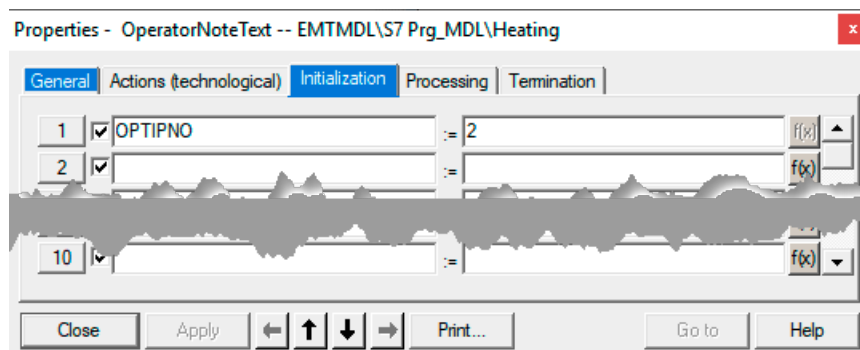
Example 1: Activating a comment

The following comments are configured:

Characteristics to Heating		Contents Of: 'Characteristics\Note texts'		
		Name	Number	Displayed text
<div> <div>Characteristics</div> <div>Control strategies</div> <div>Setpoints</div> <div>Process values</div> <div>Control values</div> <div>Parameters</div> <div>Bit memory</div> <div>Timers</div> <div>Note texts</div> <div>Block contacts</div> <div>Position texts</div> </div>		ConfAcquire	1	Please confirm to acquire the lower level EM
		ConfRelease	2	Please confirm to release the lower level EM

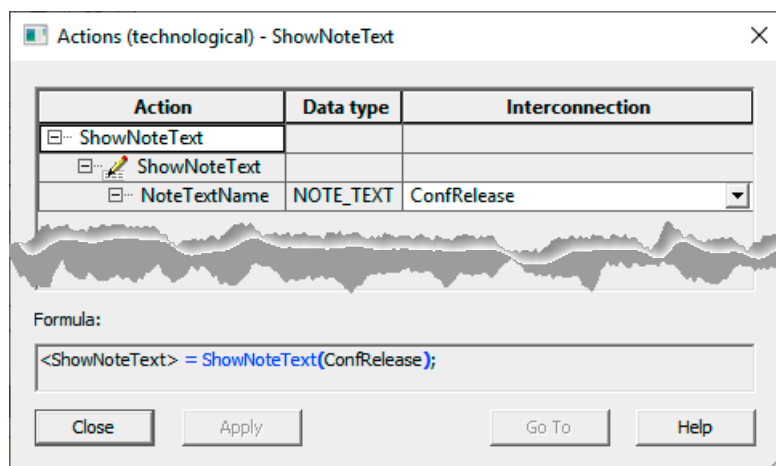
Previous, non-technological configuration

The following figure shows the previous configuration where an integer value is assigned to the OPTIPNO output. This becomes additionally very confusing and thus error-prone with a larger number of comments:



With technological configuration

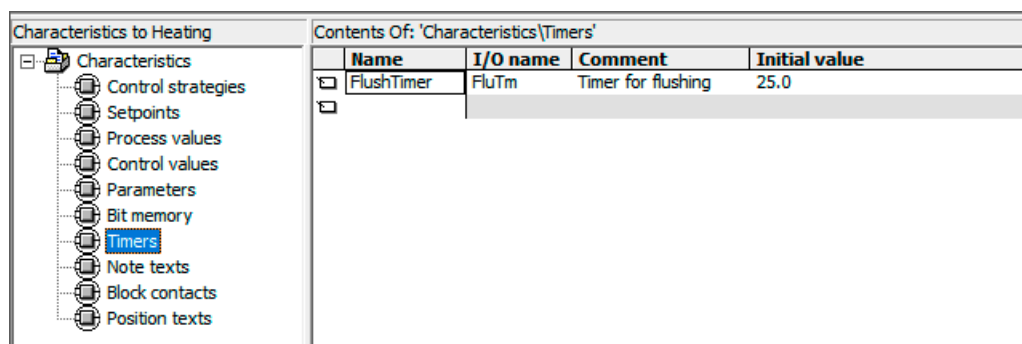
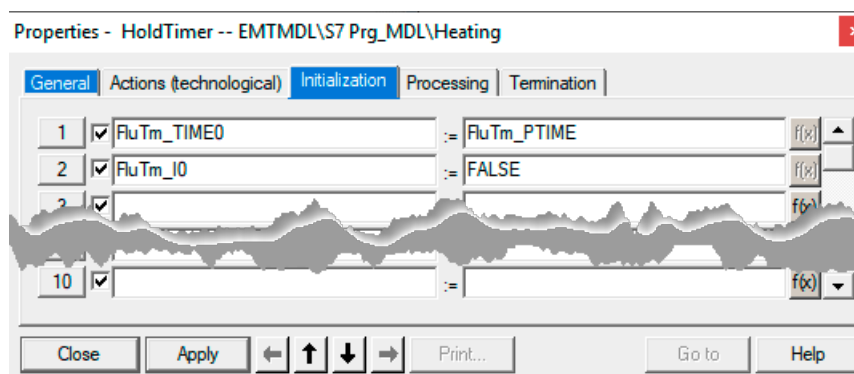
The name of the comment is offered in a drop-down list. It can also be directly recognized that the correct text was displayed.



Example 2: Hold timer

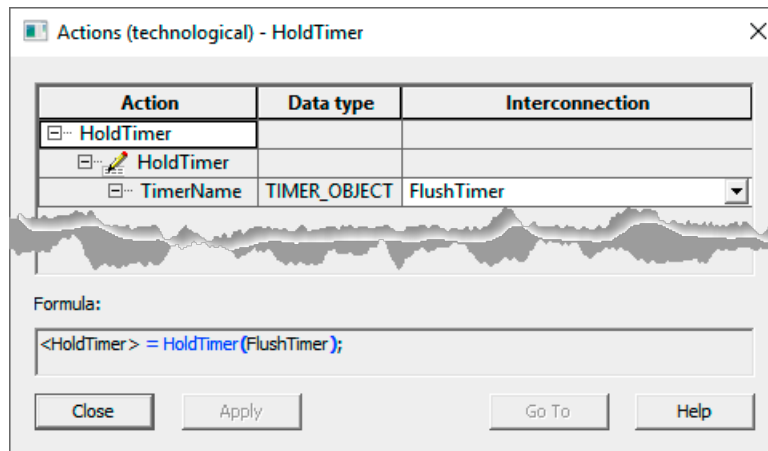
Existing configuration

The following figure shows the timer held at FALSE by the setting of the "xx_IO" (in this example "FluTm_IO") and that the remaining runtime was copied as the new runtime.



Technological configuration

The following figure shows that the "HoldTimer" command acts on the "TimerPassive" object.



10.3.4 Internal states of the SFC

The description of the internal commands of the SFC also applies to the internal states. A set of states is available in the library, as well. Examples of these are the "IsCompleted" status for querying the status of the operating state logic and the "IsActiveCS" status for comparing the current control strategy with a certain control strategy.

10.3.5 Special data types ANYBIT and ANYNUM

Commands for value assignment such as "SetNum" are also provided in the CmdStatLib. To avoid having to manage commands for data type INT, DINT and Real that not your own, these three data types are combined into a more global data type, ANYNUM.

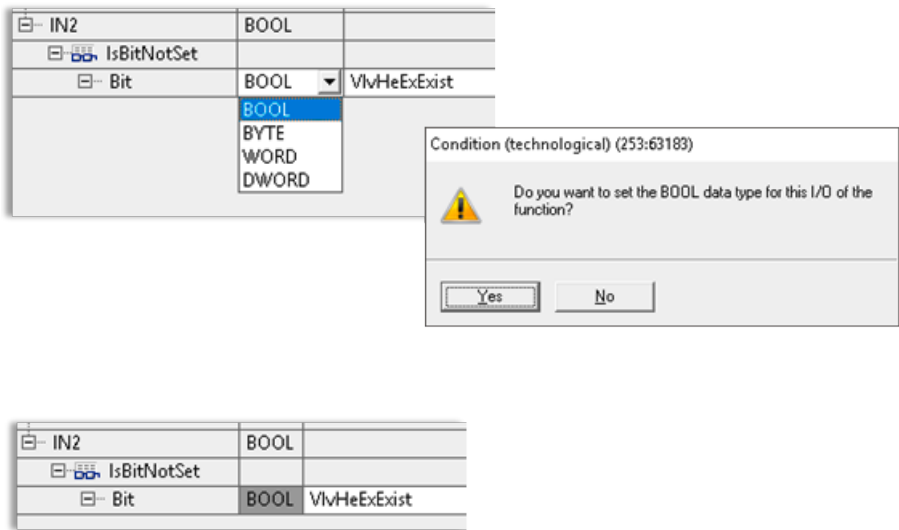
Similarly, this also applies to ANYBIT, which combines BOOL, BYTE, WORD and DWORD.

When using these commands, the concrete data type must still be specified, as shown here with the example of the IsBitNotSet status.

When placing the command, the field for the data type is still white as an indicator that it has not yet been specified. Also, the definition of the variable to be queried does not implicitly make this determination.

IN2	BOOL	
IsBitNotSet		
Bit	BOOL	VlvHeExExist

The specification must be made by the selection field, whereupon the data type field is grayed out.

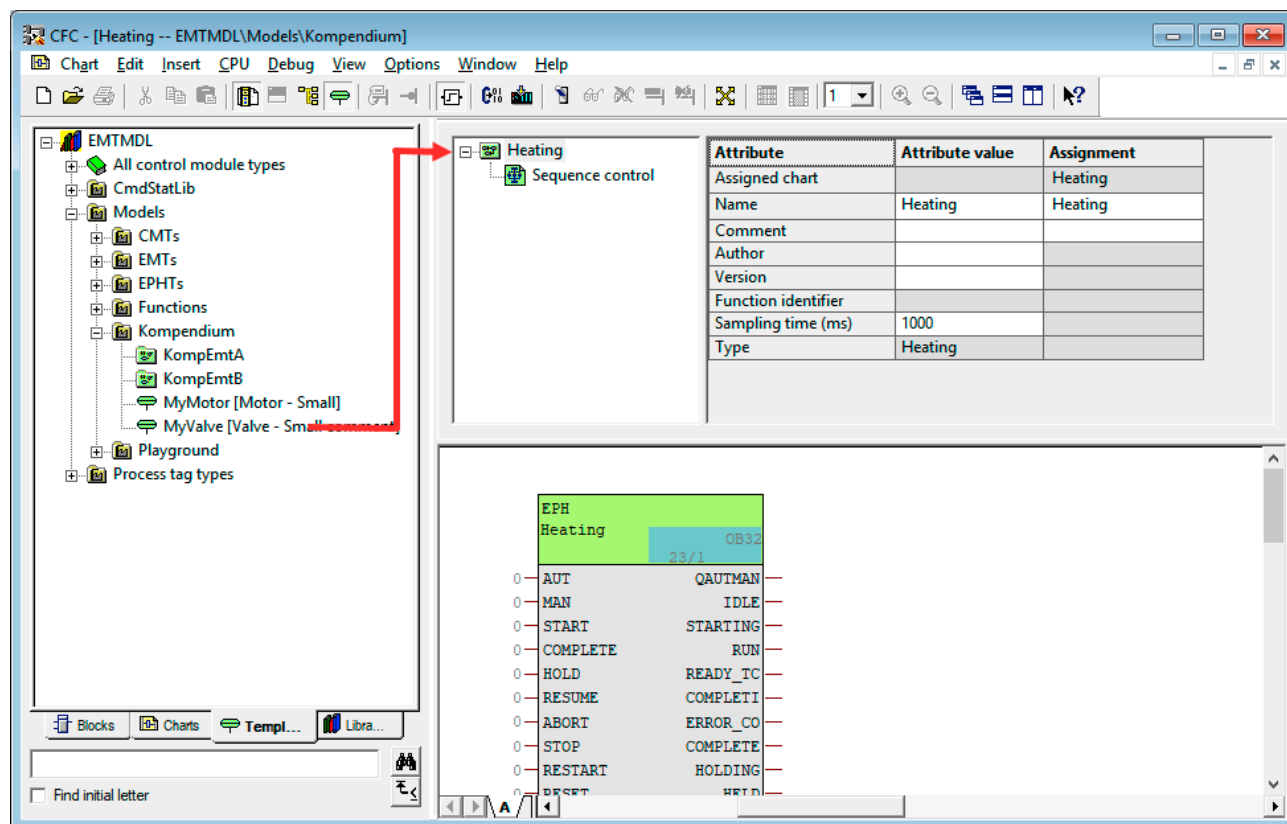


IN2	BOOL	
IsBitNotSet		
Bit	BOOL	VlvHeExExist

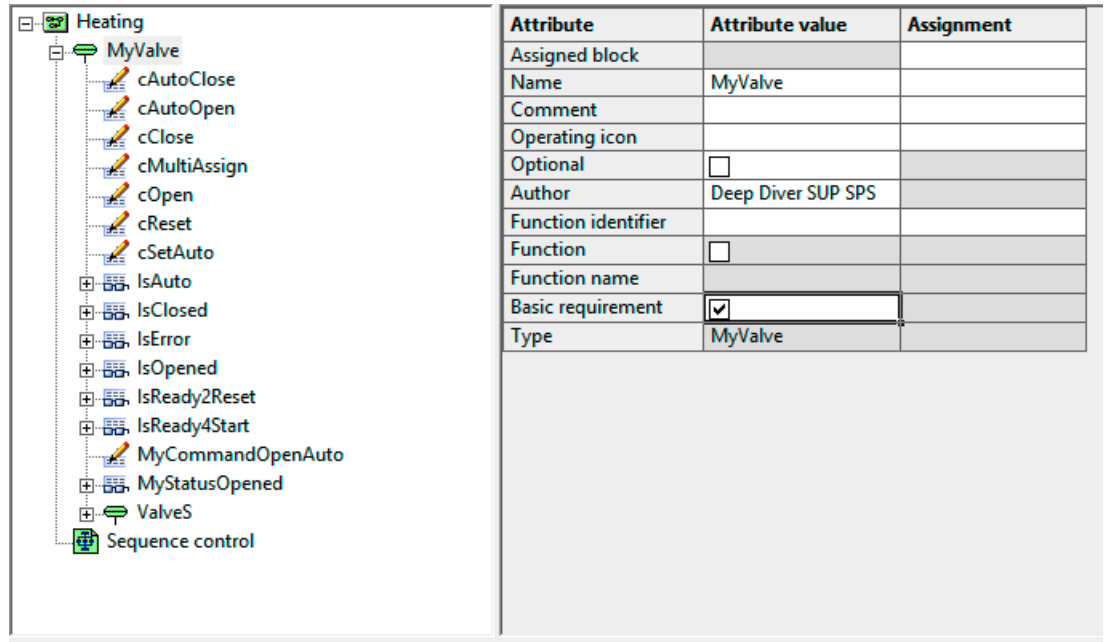
10.3.6 Use of CMTs

Declaration of an CMT in the EPHT

In order to use control modules and their commands in an EPHT/EMT, the CMT must be made known to the EPHT. To do so, the EPHT is opened and the corresponding CMT is then moved from "Catalog > Templates" to the highest node of the EPHT using drag-and-drop. The CMT "MyValve" is assigned to the EPHT "Heating" in the following example.

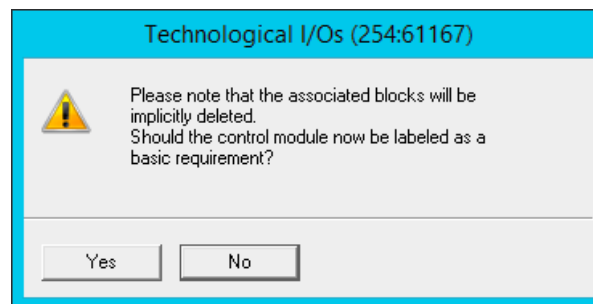


When the CMT is drawn into the EPHT a copy of the CFC of the CMT is stored within the chart of the EPHT. In most cases, this is unwanted and the CMT must be declared as a "Basic requirement". Only in a few special cases a CMT instance is placed within an EPHT/EMT.



Attribute	Attribute value	Assignment
Assigned block		
Name	MyValve	
Comment		
Operating icon		
Optional	☐	
Author	Deep Diver SUP SPS	
Function identifier		
Function	☐	
Function name		
Basic requirement	☒	
Type	MyValve	

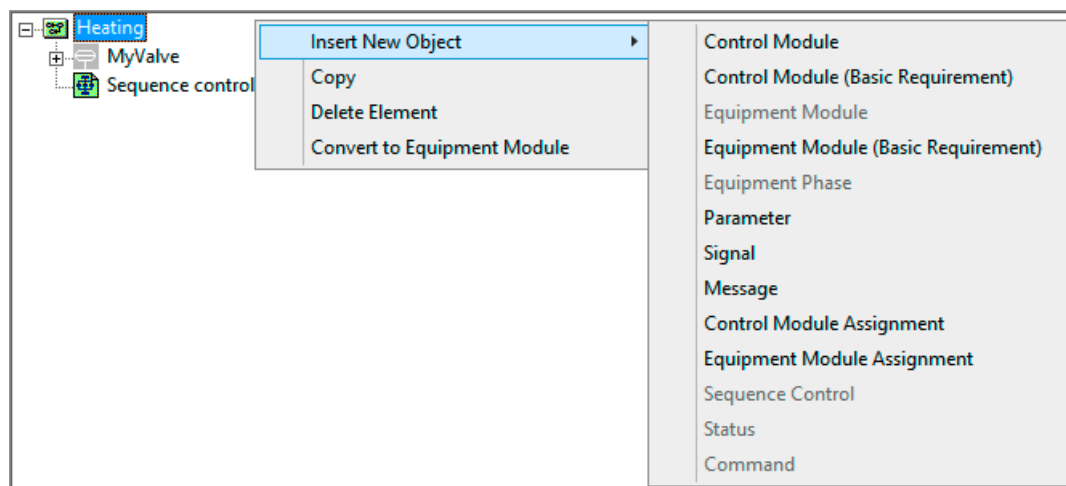
To do so, you set the "Basic requirement" property of the CMT, which triggers the following note:



After this message is confirmed and, if necessary, the view is refreshed using the F5 key, the CFC blocks are removed from the chart of the EPHT.

Declaration of the roles

The CMs used later fulfill a task or role from the perspective of the EPHT. This role is also defined at the EPHT. For each new role, a new object ("Control Module Assignment") is to be inserted in the highest node of the EPHT using the shortcut menu and provided with a descriptive name.

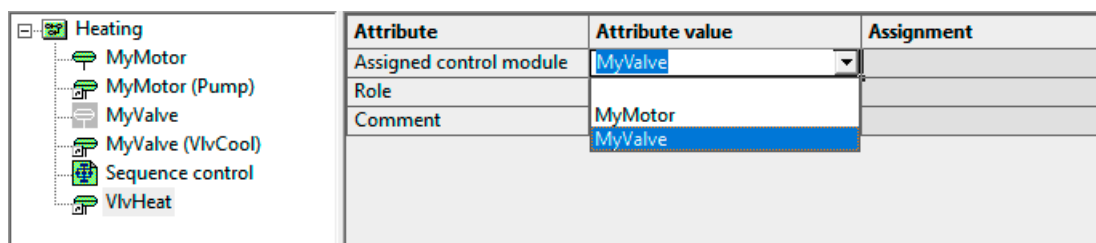


The name must not exceed 12 characters because a block contact will be generated from it later. The comment field is used for a more detailed description.

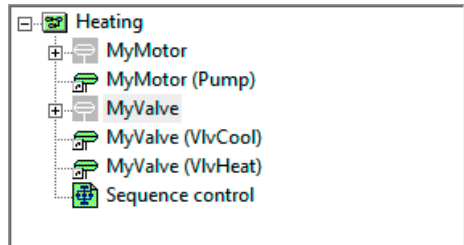
As an example, the sequential control system is to control two valves of the same type. A "VlvCool" or "VlvHeat" role is created for each valve for this.

The pull-down menu in the "Assigned Control Module" field, which offers the available referenced CMTs, defines which CM type is to be later used for each role.

In the following example, the "VlvCool" role was assigned the CM type "MyValve". The "VlvHeat" role is not yet assigned to a CMT.



For other roles, e.g. "Pump", that are based on the CMT "MyMotor", the procedure is similar:

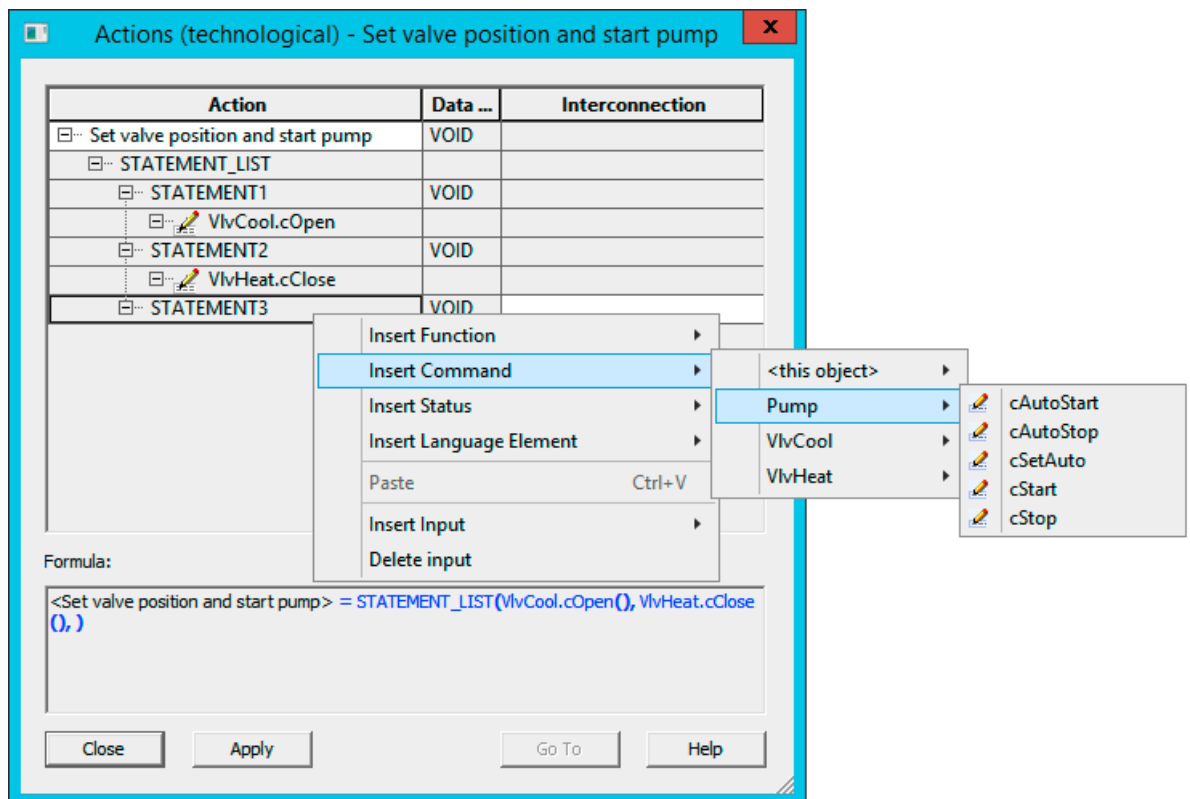


Using CMT commands

As part of configuring the CMT, "commands" have been defined for the CMT. Once the actual preparations have been completed in the EPHT, the commands assigned to the respective role are available within the technological actions.

In the following example, the "Set valve position and start pump" action is created in a step. The following commands are used in this example:

- "cOpen" for the role "VlvCool" based on the CMT "MyValve"
- "cOpen" for the role "VlvHeat" based on the CMT "MyValve"
- "cStart" for the role "Pump" based on the CMT "MyMotor"



Automatically generated block contacts in the characteristics dialog

When a role is used for the first time, the system automatically creates a block contact for the technological block used in the CMT. The following three block I/Os (block contacts) were created for the example: "Pump" with block "MotS", "VlvHeat" with block "VlvS" and "VlvCool" with Block "VlvS".

Characteristics to HeatSFC		Contents Of: 'Characteristics\Block contacts'				
<div> <div>Characteristics</div> <div>Control strategies</div> <div>Setpoints</div> <div>Process values</div> <div>Control values</div> <div>Parameters</div> <div>Bit memory</div> <div>Timers</div> <div>Note texts</div> <div>Block contacts</div> <div>Position texts</div> </div>		Name	Display name	Block	I/O name	Comment
	Pump	Pump	Pump	MotS	Pump	Pump for circulation of the media
	VlvHeat	VlvHeat	VlvHeat	VlvS	VlvHeat	Valve for circulation of the heating source
	VlvCool	VlvCool	VlvCool	VlvS	VlvCool	text is independent from comment inside role of EPHT/EMT

In some cases, it can be convenient to delete the generated block I/Os and have them be generated again by using a command/status.

Note

The name of the block I/O (block contact) and the entry in the "I/O name" column must not be changed.

10.3.7 Further notes on SFC types in connection with EMT / EPHT

10.3.7.1 Using a SFC type in multiple EMTs

There is a direct relationship between EMT and SFC type. Therefore, an SFC type can only be used in one EMT.

10.3.7.2 Copying an EMT in the master data library

If it is absolutely necessary to copy an EMT, the FB numbers must be adapted, because in doing so they obtain an FB number starting with FB1 instead of the usual FB number range >FB1024.

Since V9.1, there is a "FB number" column for this in the SIMATIC Manager in the chart folder, which allows a simple overview and management.

10.3.7.3 Size of the FB and reserve

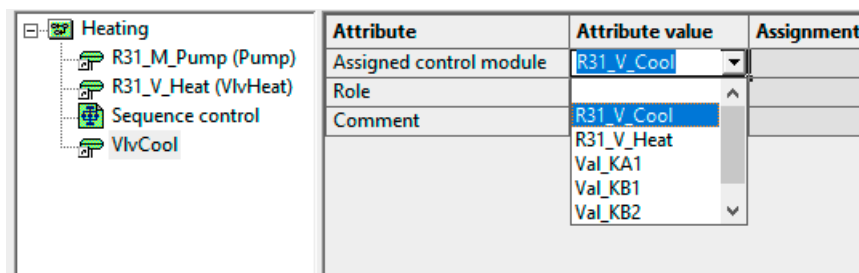
When creating an SFC type, an FB function block and FC functions are created implicitly. The limit of approx. 64,000 bytes continues to apply as an absolute upper limit for the FBs. In order to be prepared for future innovations, this limit must not be exhausted, otherwise, for example, errors during compiling could prevent loading. Therefore, make sure that you have a reserve, for example, of 20%, i.e. an FB size of less than 50 KB.

Irrespective of this, the recommendation to keep the utilization below 70% still applies for the entire AS, so that delta downloads can run safely, especially with CiR.

10.3.8 Instantiation

EPHTs and EMTs are instantiated in the same way as CMTs. For this, the EPH type is moved from the master data library to the desired location in the plant hierarchy of the target project using drag-and-drop. In so doing, an EPH is generated from an EPH type or an EM is generated from an EM type.

If lower-level control modules or equipment modules were also configured as roles in the EPHT/EMT, they must also be assigned to specific CM instances at the EPH in the project. For this purpose, all available instances that are based on the referenced CM type are offered for selection for each role. After selection, all I/Os are automatically connected.



If there is an erroneous interconnection, the empty field must be selected during the assignment. All connections are then deleted again.

Note

EPH and EM instances should not be copied in the project. Instead, they should always be dragged from the master data library to the appropriate position in the project.

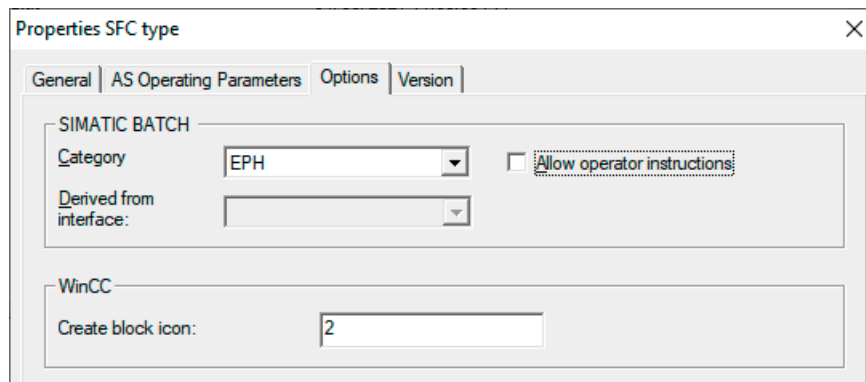
This applies in particular to whole hierarchy paths in which objects are interconnected (technologically) because it is possible that the references are not updated or created again.

Visualization of the CM in the OS/SFC visualization

So that all available possibilities can be utilized in the visualization, the current block icons must be generated during the OS compilation.

To do this, the number "2" must be entered in the "Create block icon" field under WinCC in the "Options" tab of the properties of the respective SFC type.

This is no longer necessary with newer PCS 7 versions, i.e. the innovated block icon is now used for newly created SFC types by default.

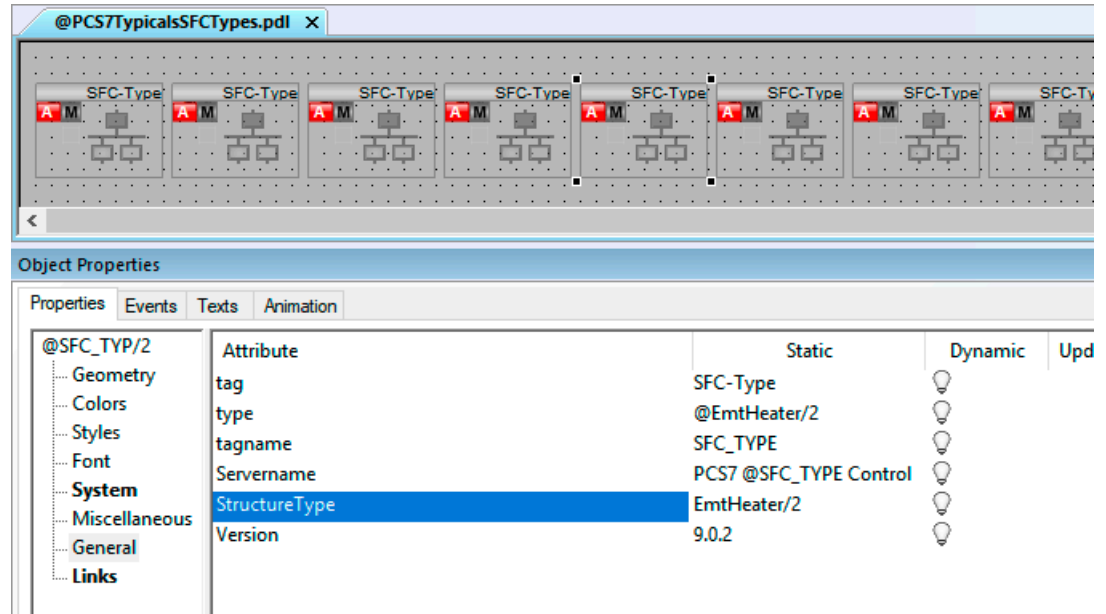


How are block icons for SFC type instances generated internally?

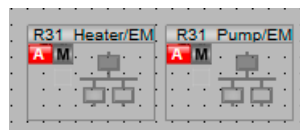
Block icons are created and placed in two steps.

First, SFC type-specific templates (in the figure @PCS7TypicalsSFCTypes.pdl) are generated for each SFC type from a general template. Later, an adapted copy is placed in the picture for each instance of the respective SFC type.

The following figure shows a section of the @PCS7TypicalsSFCTypes.pdl picture generated during OS compilation (a separate template is created for each SFC type, recognizable by the StructureType property).



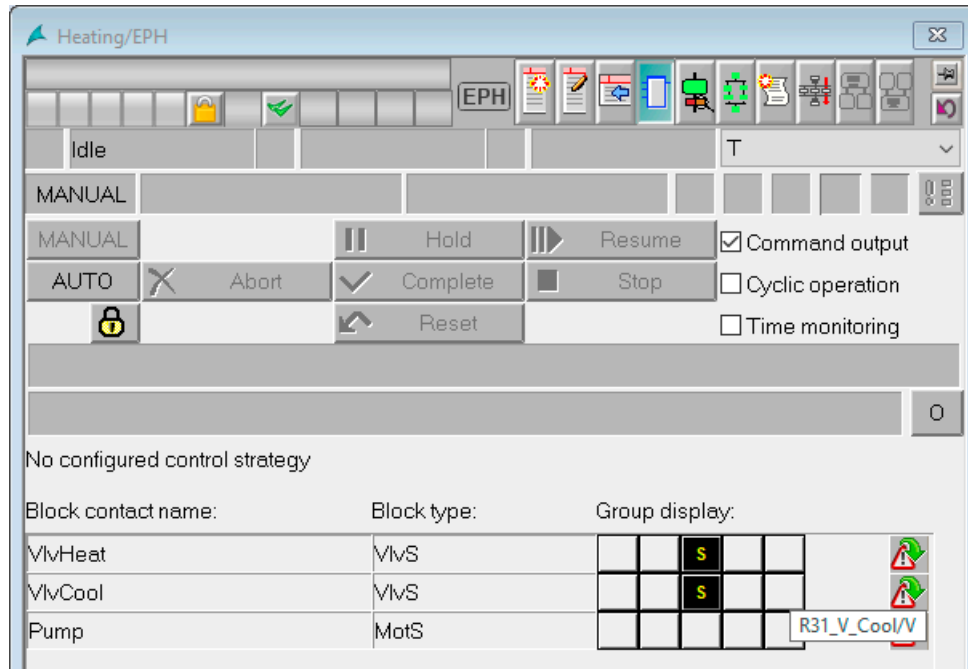
The next figure shows the instance of a block icon.



10.3 EPH types and EPH instances

The faceplates are also implemented in the current APL style:

Block I/Os, e.g. valves and motors, have their own subordinate picture in the faceplate for the SFC instance in which they are visualized in the OS.



It displays the status of the block instance belonging to the role or block I/O. The associated instance can be determined from the tooltip text on the button for the loop-in. When the button is pressed, a jump to the corresponding picture is initiated and the instance is marked.

Block contact name:	Block type:	Group display:
VlvHeat	VlvS	
VlvCool	VlvS	
Pump	MotS	

10.3.9 Naming

Control strategies, setpoints and block contacts must have unique names.

Because a block contact is created implicitly during creation of a technological action (with the role name), a collision can occur. Therefore, the name of the role cannot be the identical to the name of a control strategy, even if these different objects are independent of one another.

10.3.10 FB numbers of the SFC types

Basically, FB numbers have to be unique only within a (sub-)project and this is ensured by the SIMATIC Manager.

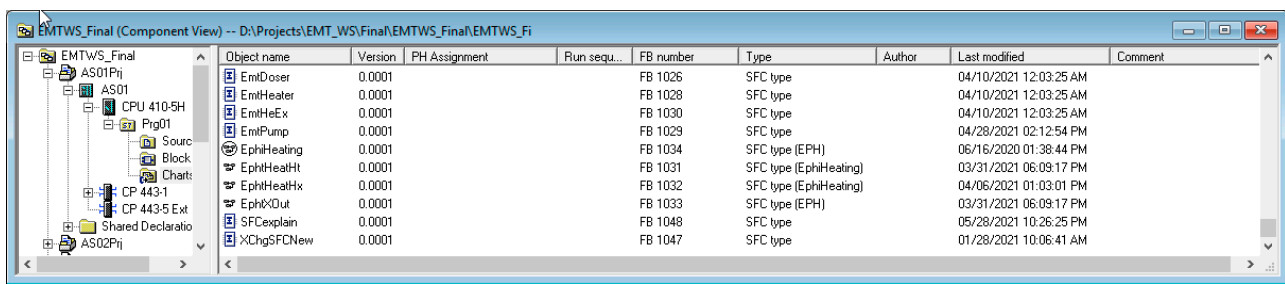
This principle should also apply to the entire multiproject to ensure that potential conflicts are avoided from the outset. The master data library, MDL, should always be decisive.

For SFC types, the block number of the FB can be specified by the user in the properties dialog. This should start with FB1025.

In current versions, when an EMT or EPHT is copied, usually the first free FB number is assigned, starting with FB1. This should be immediately corrected to a suitable FB number.

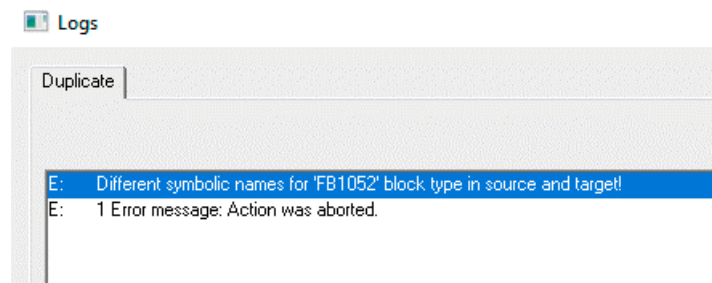
In this way, integrity between FB name, FB number and its function can be ensured over the entire multiproject.

Since PCS 7 V9.1, this is facilitated by the additional "FB number" column in the SIMATIC Manager.



Object name	Version	PH Assignment	Run sequ...	FB number	Type	Author	Last modified	Comment
EmtDoser	0.0001			FB 1026	SFC type		04/10/2021 12:03:25 AM	
EmtHeater	0.0001			FB 1028	SFC type		04/10/2021 12:03:25 AM	
EmtHeEx	0.0001			FB 1030	SFC type		04/10/2021 12:03:25 AM	
EmtPump	0.0001			FB 1029	SFC type		04/28/2021 02:12:54 PM	
EphiHeating	0.0001			FB 1034	SFC type (EPH)		06/16/2020 01:38:44 PM	
EphiHeatHt	0.0001			FB 1031	SFC type (EphiHeating)		03/31/2021 06:09:17 PM	
EphiHeatHx	0.0001			FB 1032	SFC type (EphiHeating)		04/06/2021 01:03:01 PM	
EphiXOut	0.0001			FB 1033	SFC type (EPH)		03/31/2021 06:09:17 PM	
SFCexplain	0.0001			FB 1048	SFC type		05/28/2021 10:26:25 PM	
XChgSFCNew	0.0001			FB 1047	SFC type		01/28/2021 10:06:41 AM	

If a message like the following is output when instantiating an EM, the SFC type used should first be copied from the master data library into the target project.



Then, if necessary, the FB number in the MDL must also be adapted to the FB number from the project. Only then can the EMT be copied / instantiated in the target project.

10.4 Updates

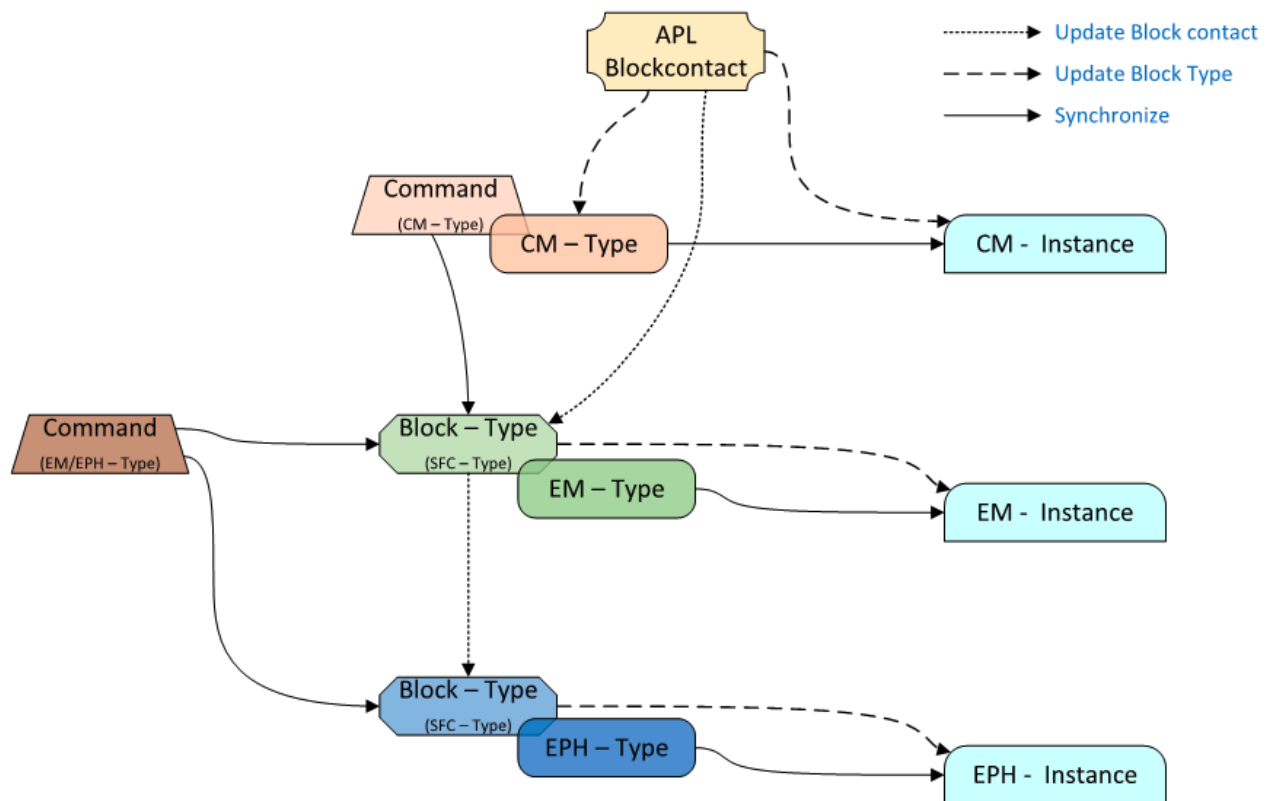
Handling changes is an important aspect. Depending on what changes were made, appropriate actions must be performed so that they take effect.

Typical actions are:

- Synchronization of the commands or EPH instances
- Update block types of the SFC types
- Block contacts within the SFC type
- Close textual interconnections (also already in the MDL)

These actions are illustrated by some examples, because by keeping to a necessary sequence and limiting the actions to those that are actually necessary, considerable time can be saved and the desired result can be achieved.

First, a general overview of the objects and the actions that may be required:



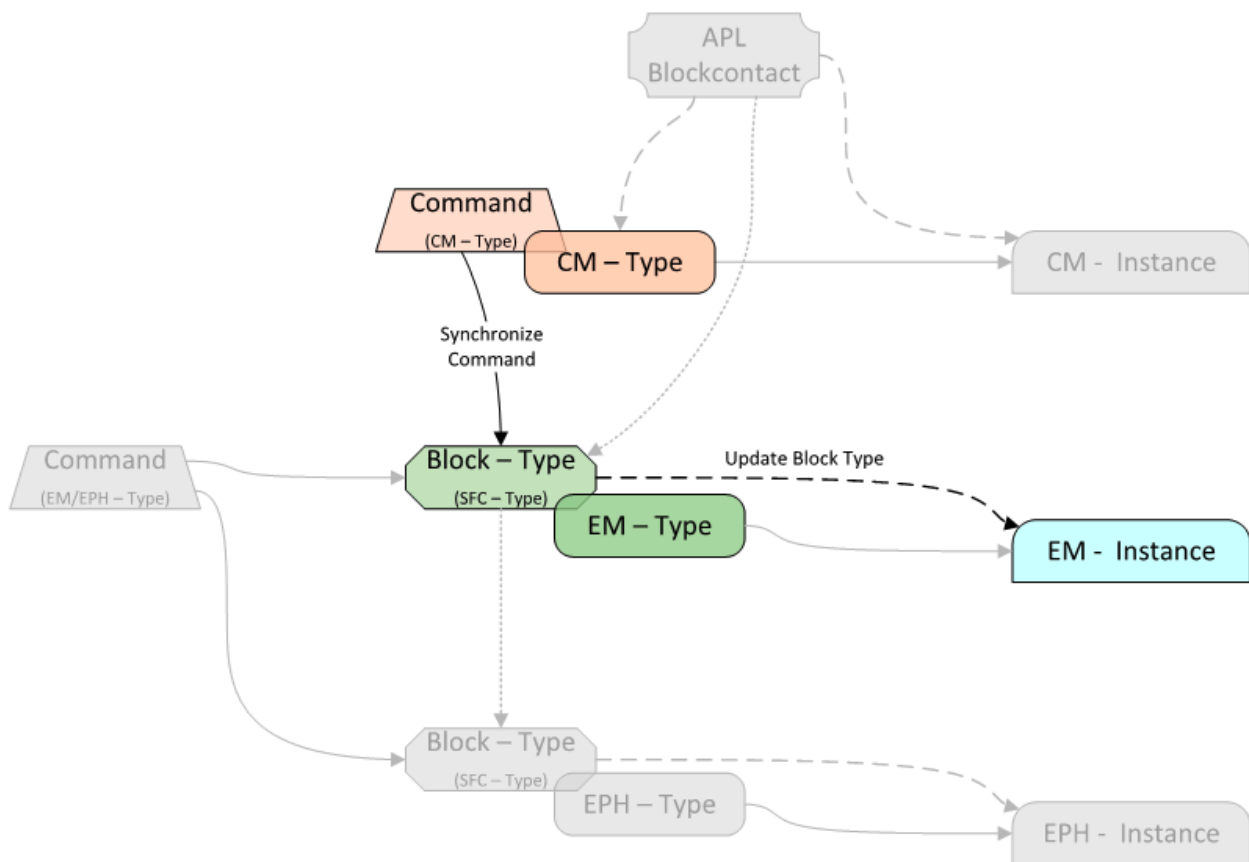
The objects CM(T), EM(T) and EPH(T) are shown here deliberately divided. For example, for EM(T) the SFC type has been separated from the rest of the EM(T). The EMT technological object can be synchronized, while the SFC type belonging to it is brought into the project via a device type update. Global commands and states as provided by the CmdStatLib are shown on the left as a separate object.

In most cases, only a few actions are required.

10.4.1 Example 1: Modification of a command at the CMT

It is possible that the commands and states of the CMT were formally defined but not configured with detailed value assignments for the CMT. These commands can be used within the technological SFC configuration even if they have no effect.

In this example, the content of a previously used command is configured or changed. This change must take effect at the locations of use within the SFC types and within the master data library. Accordingly, this change must be added to the master data library through a synchronization.

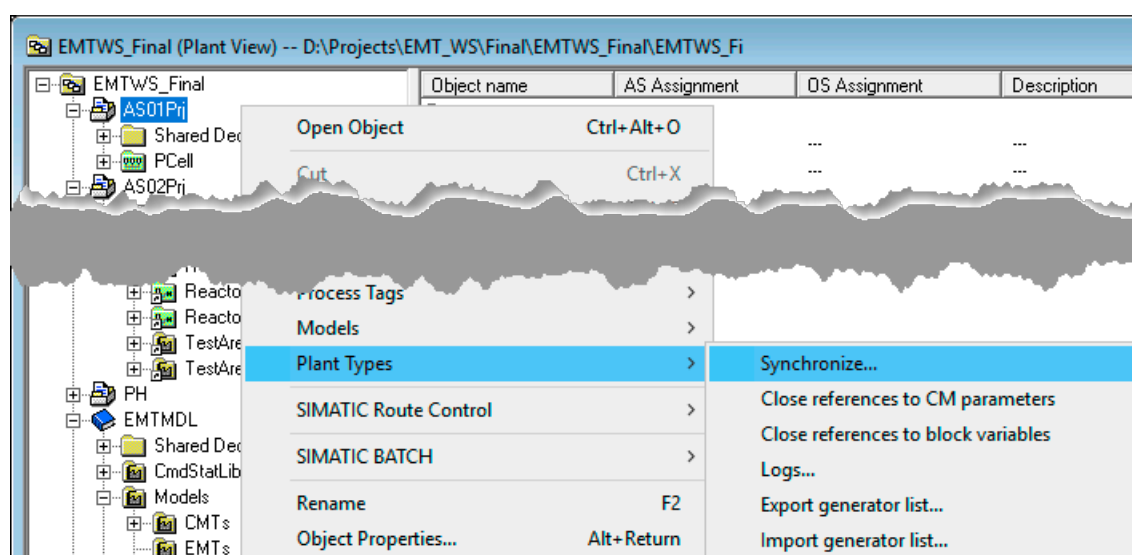


When changing a command of a CMT, the actual CMT is not affected. Locations of use for commands are exclusively in EMT or EPHT. To have the changed command implementation take effect there, the changed commands (and status) must be **synchronized**. The object of synchronization is therefore the command.

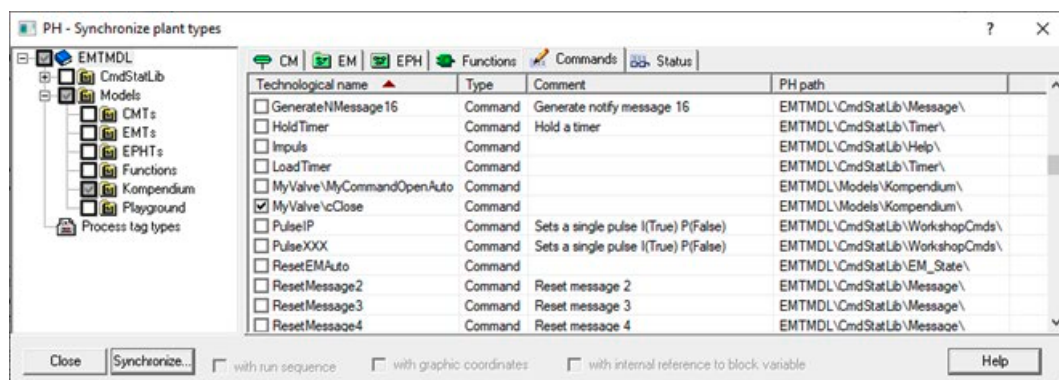
As a result of the synchronization, the SFC has changed. This change must be transferred to the projects via "**Update Block Types**".

Synchronization of CMT, EMT or EPHT is not required and would not be successful.

A project ("Technological types > Synchronize") is selected as the target of the synchronization.



Only the modified command "cOpen" command of the CMT "MyValve" is selected in this example. For this, the "Commands" tab is selected and the commands to be synchronized are selected.



When the synchronization is initiated with the "Synchronize" button, the objects that must be synchronized are determined and displayed. The button actually has the Compare function, because no change has been made yet, and it can therefore be used without risk.

As a result of the comparison, another window opens. In the hierarchy tree, two EPHT and one EMT are listed, in which there are different uses of the "Close" command. There are several locations of use within the "Doser" EMT, and you can recognize at the lowest level that the assignment in the "Processing" tab was not present beforehand ("Only in B" status) and was deleted in the "Termination" tab.

Data transfer - Synchronize templates

A: Project - [AS02Prj -- D:\Projects\EMT_WS\Final\EMTWS_Final\AS02Prj]

B: Command: MyValve\cClose

Attribute	A	B	Status
1 OS-relevant		true	Only in B

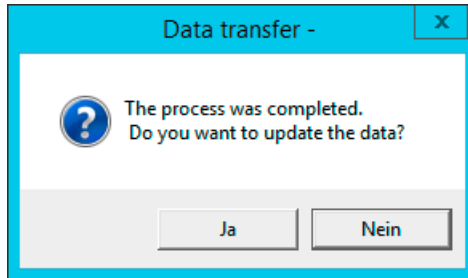
Lower-level object	Object name	Status
1 Primary expression	Variable access	Only in B
2 Primary expression	Constant	Only in B

Object type Object name Object path Message

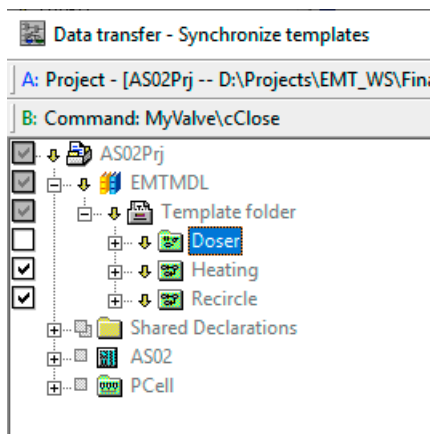
Close [Navigation icons] Synchronize templates Help

A decision can be made now as to whether all uses will be synchronized or whether individual branches will be excluded. By clearing the check box, you exclude the point of use and hierarchy levels below it from the synchronization.

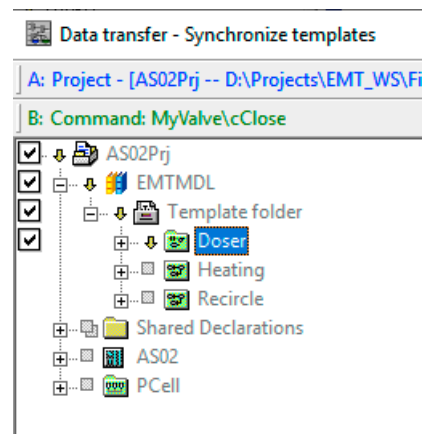
The selected instances are only synchronized after pressing the "Synchronize templates" button. After that, a message regarding the data update is displayed and the changes can be checked.



In this example, the "Doser" hierarchy path was first deselected, which resulted in the following after the first synchronization:



Before



Intermediate state

After the first synchronization, the locations of use in the "Heating" and "Recircle" hierarchy paths already correspond to the current configuration. The "Doser" folder does not yet correspond to the current configuration.

After selection of "Doser" and a further synchronization, all locations of use correspond to the "target" configuration.

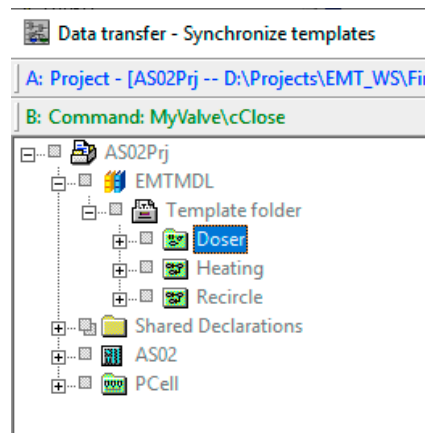


Figure 10-1 After

The SFC types within the MDL have been modified due to this synchronization process. These changes to the SFC types must be transferred from the master data to the target project via "Update Block Types".

Note

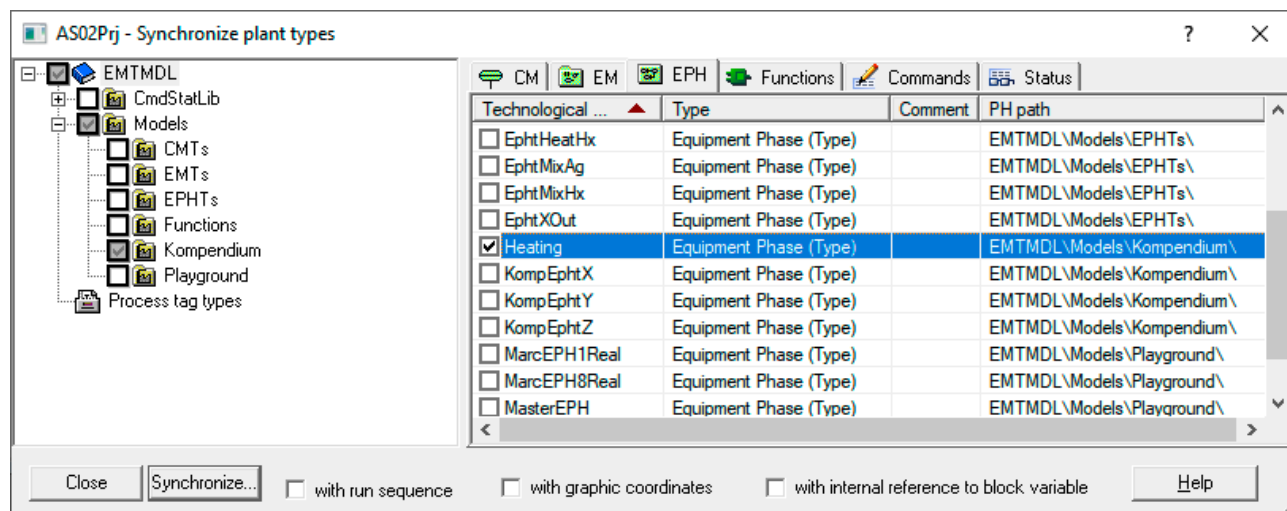
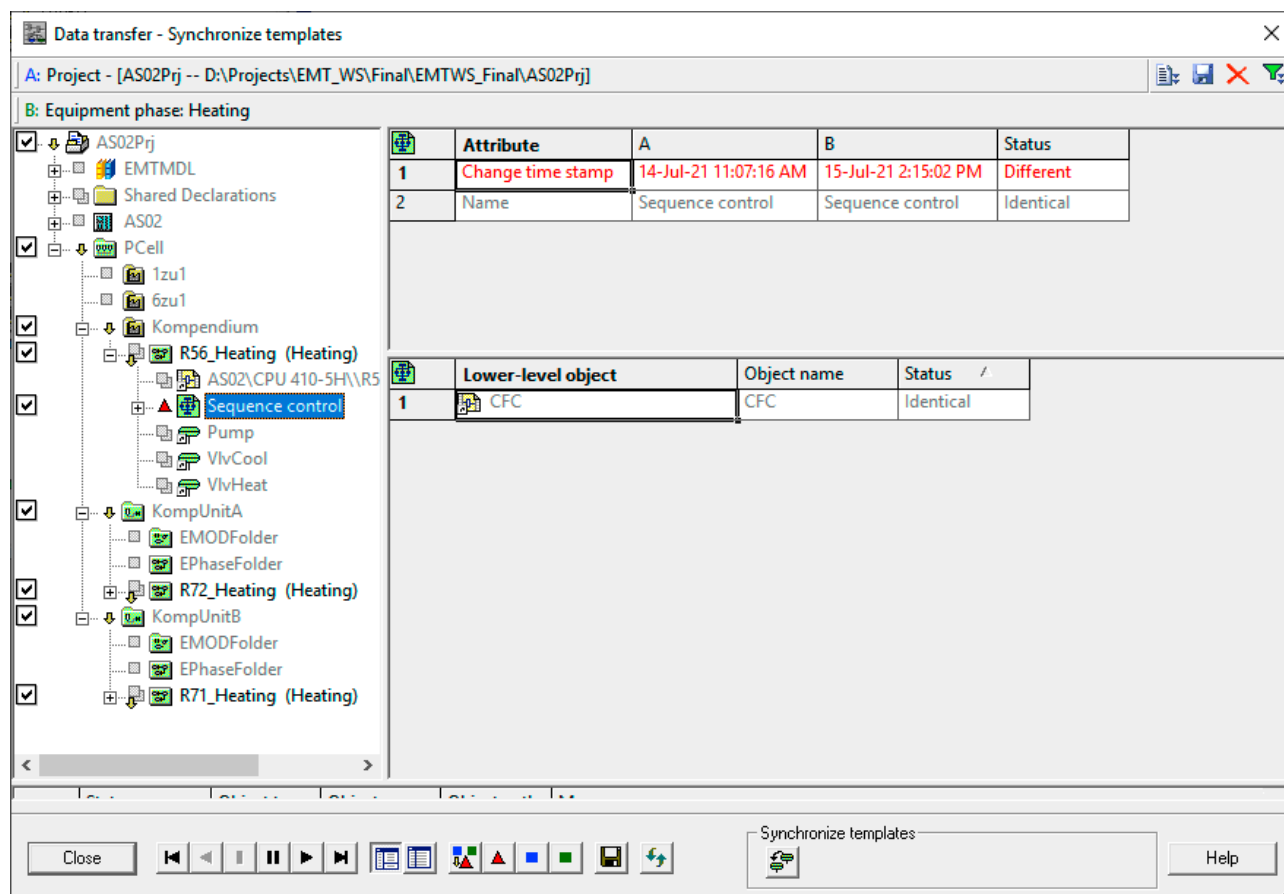
The SFC types from the MDL are updated in the project through the "Update Block Types" command. This means that the MDL is the starting node in the Component view. The block types are updated using menu command "Options > Charts > Update Block Types", and the project is synchronized by a synchronization. This means that, in this case, the project in the Plant view is the starting node and the function is also available in the shortcut menu.

Note that after opening the dialog, all (sub-)projects and also all libraries are selected and updated by default. If, for example, a library is being used as a backup, it is essential to deselect it beforehand.

Synchronization operations are always performed only for the selected tab. It is not possible, for example, to synchronize commands, CMTs and EPHTs in one operation, because there are often dependencies.

10.4 Updates

If the "Heating" EPHT is synchronized in the next step, the result remains the same after synchronization. The synchronization mechanism contains **no** update of the block types ("Update Block Types"). Consequently, the SFC types in the master data library and project are also different.



The update of the block types must be performed separately in the meantime.

Update block types

Select the S7 programs to be checked. 1(3)

	Download in R...	Program	Resources	Status
☒	☐	AS01Prj\AS01\CPU 410-5H\Prg01		☐
☒	☐	AS02Prj\AS02\CPU 410-5H\Prg02		☐
☒		EMTMDL\S7 Prg_MDL		☐
☐		ReserveLib\S7 Prg_DumLib		☐

Log file: Other file...

Back Next Cancel Help

After you have selected the S7 programs to be checked, the block types that will be updated are displayed in the next dialog. Confirm the update by clicking on "Finish".

Update block types

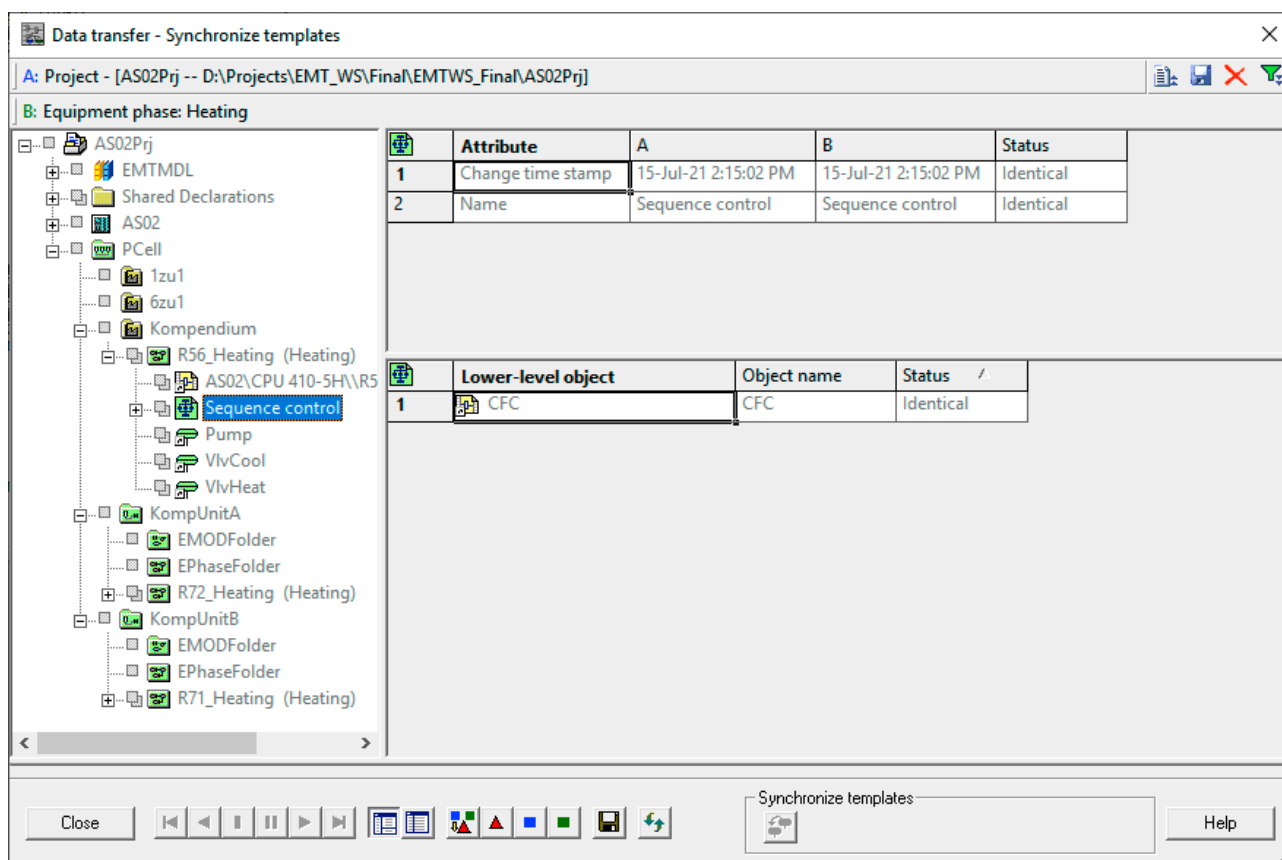
Select the block types to be updated. 2(3)

	Block type	Program	Consequence --->
☒	Heating	AS02Prj\AS02\CPU 410-5H\Prg02	 The OS needs to be recompile

Back Finish Show differences Cancel Help

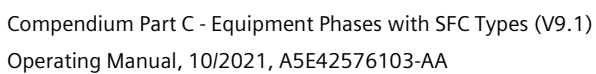
10.4 Updates

If another synchronization is then performed for the EMT "MyEMTIn", there are no more differences.

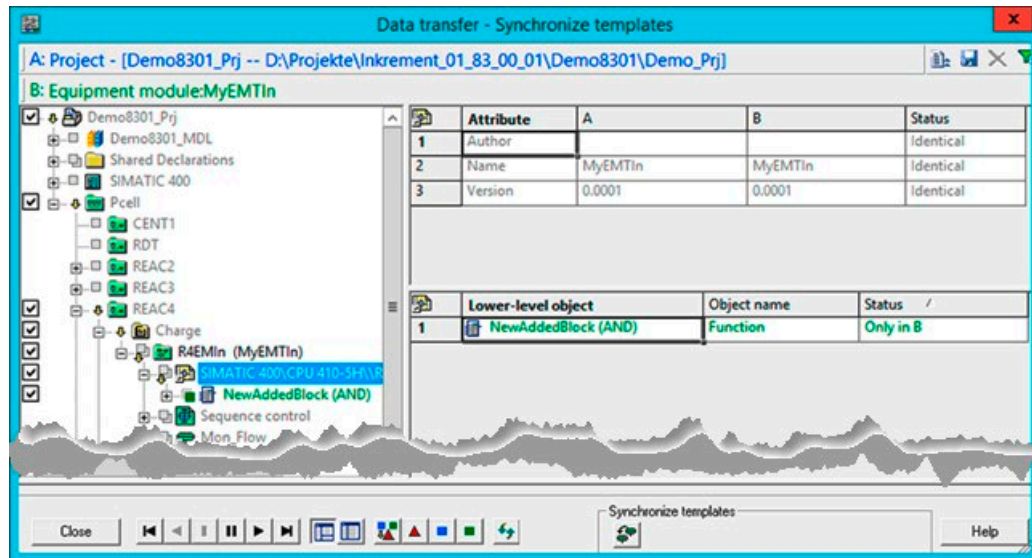
**Note**

Object types (e.g. commands, CMTs, EMTs and EPHTs) can only be synchronized separately.

In this example, an AND gate named "NewAddedBlock (AND)" was added in the chart for the EMT "MyEMTIn" (without interconnections, etc.). In this case, a synchronization will cause all instances to have the new block.



In this case, only the EMT was changed, but not the corresponding SFC type. Similar to a change to a CMT, this change must be **synchronized** to the EM instances. The object of the synchronization is therefore the EMT.



10.4.3 Example 3: Explicit change of the "s7_contact" attribute at block I/Os of CMTs

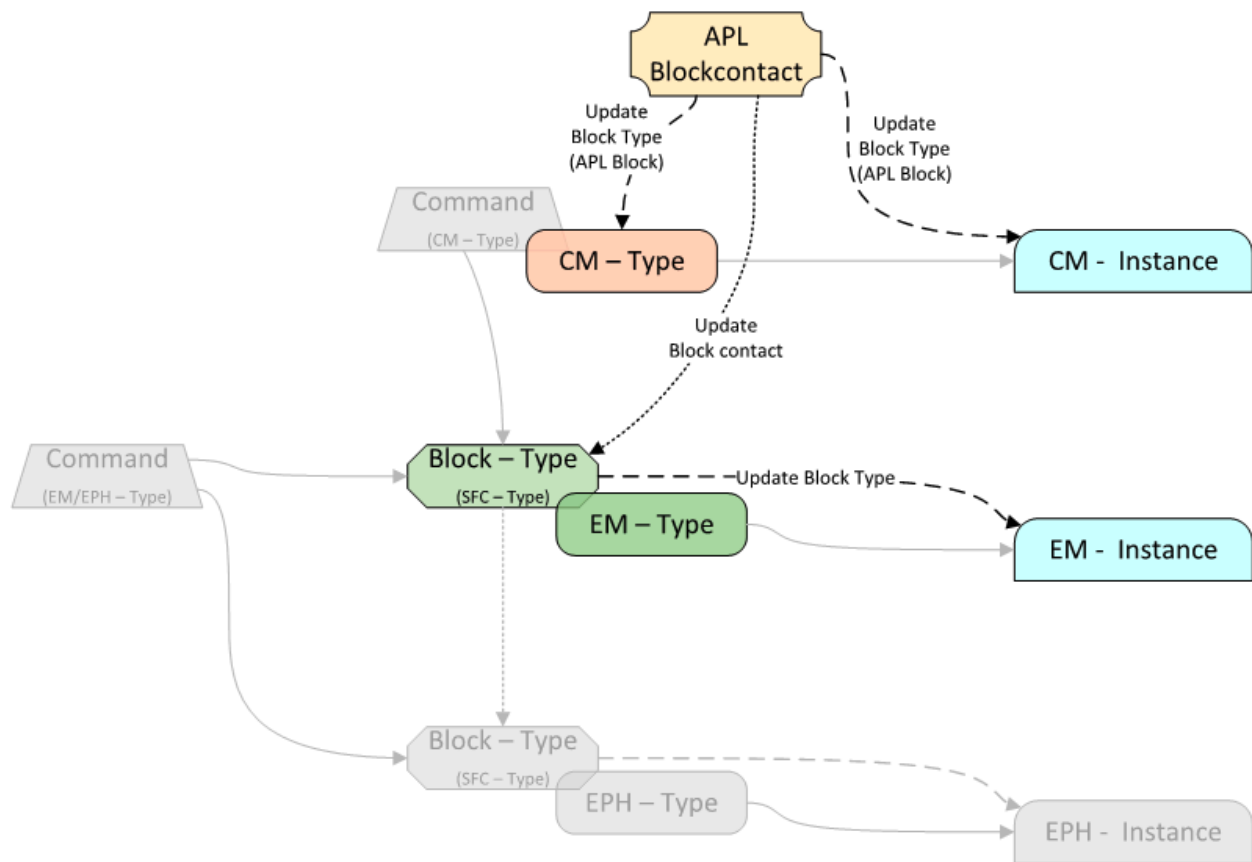
If only control modules with technological blocks from the APL were used when configuring the SFC, the "s7_contact" attribute of the block contacts generally does not change or changes only with a new library version.

When using the technological configuration, however, it is more often necessary to adapt the attributes of the used blocks to the requirements.

The technological block MonAnS is mentioned here as an example. It has no "s7_contact" attributes in the factory state. If you want to use the actual value or query the limit monitoring, you have to adapt the attributes of the block type and explicitly set them in this case.

To do this, a mirrored interface is created at the SFC type when the feature block contact is created, but it is deliberately not adapted automatically. The menu bar contains the "Options > Block Contacts..." item for updating the mirrored interface.

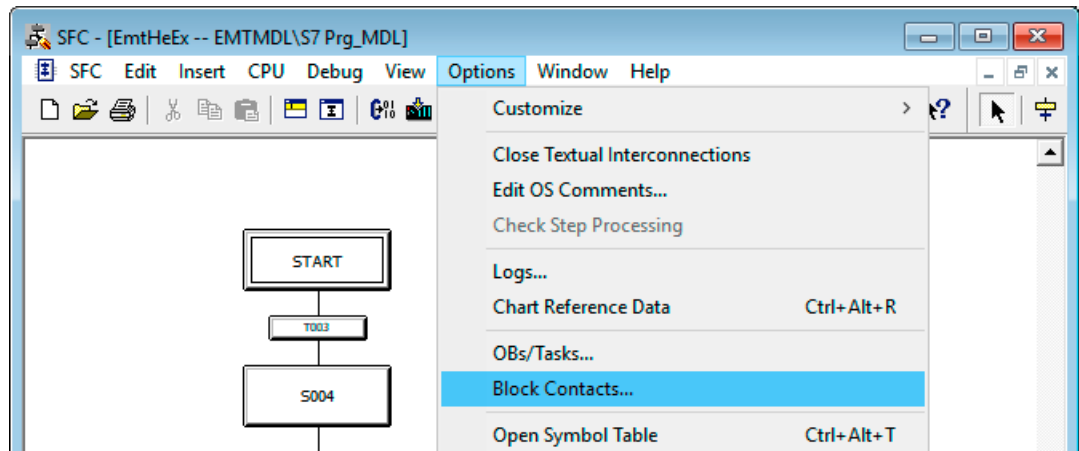
The following overview shows the usecase of attribute change at the APL block:



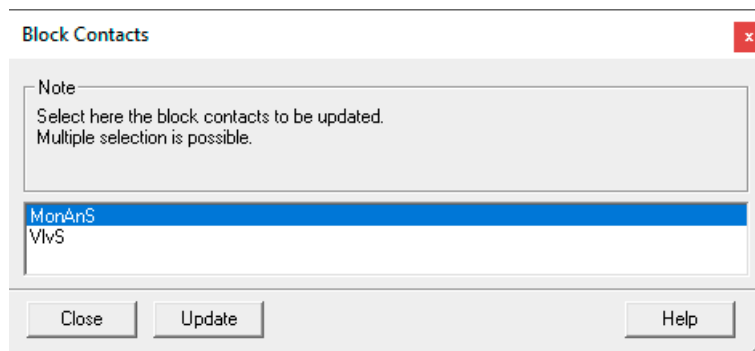
After changing the attributes of the (APL) block in the master data library, this change is simultaneously inherited in the CFC of the master data library and in the module order as well as CFC of the projects using **"Update Block Types"**. The object of the first block type update is the function module.

To adapt the mirrored interface of the SFC types in the MDL, these must be opened in the SFC editor and **"Options > Block Contacts..."** must be executed.

Since the SFC type has changed as a result, this change must be transferred into the projects with "**Update Block Types**" for the affected SFC types. The object of this second block type update is the SFC type.



The lower-level objects must then be selected and updated in the window that follows. For each block contact with "s7_contact" attribute of the lower-level block or EMT, the corresponding counterpart is created at its own SFC.



10.4.4 Example 4: Implicit change of "s7_contact" attributes at block I/Os of an EMT (adding a setpoint)

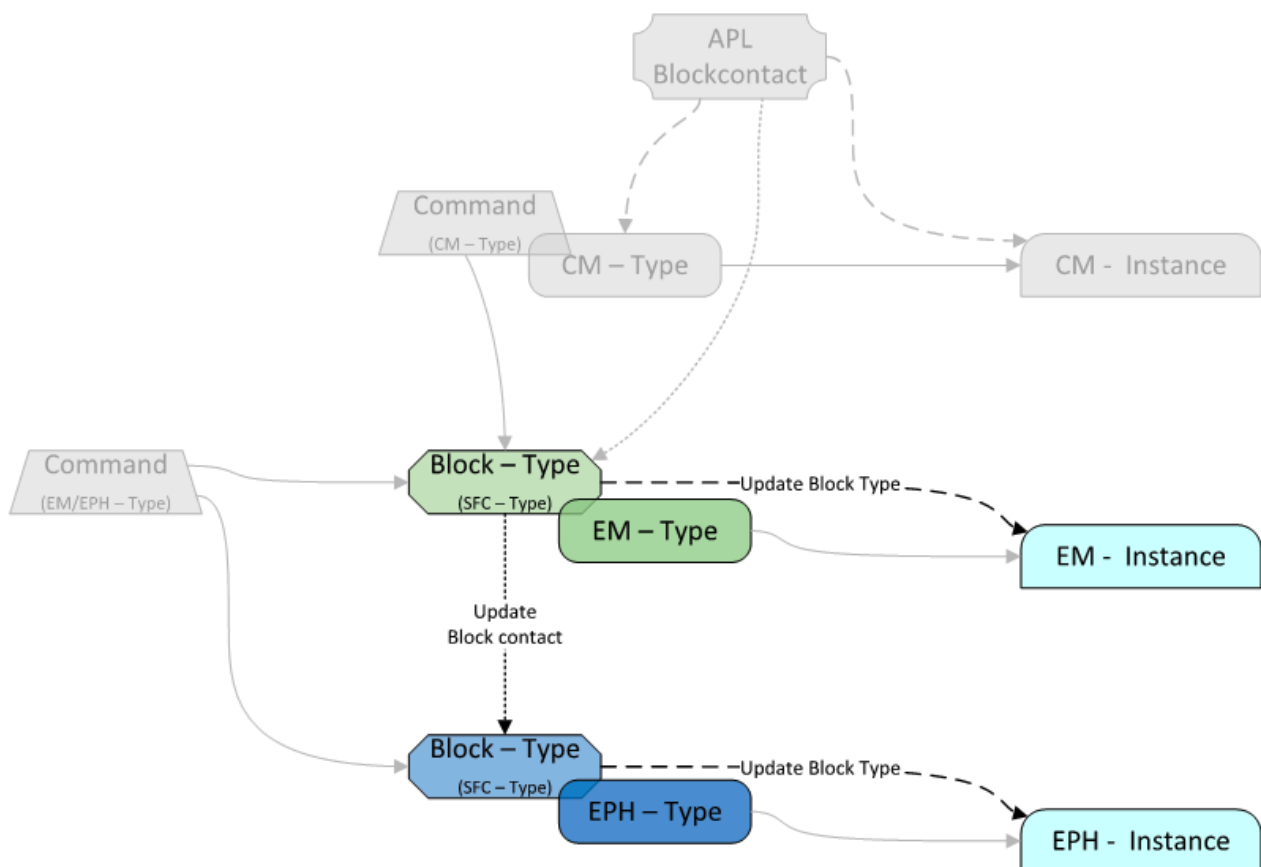
The "s7_contact" attributes are assigned explicitly for technological blocks as used in CMTs. In contrast, this is done automatically or implicitly for EMTs, so that the mirrored interface for the EM is created at the EPH without any effort. In addition to the always present inputs and outputs for control and assignment, type-specific connections for setpoints or parameters are also created. These must also be usable for control by a higher-level EPH or EM. The attributes required for this are assigned by the system without additional configuration.

(In justified exceptional cases, the user can give the attribute to additional connections.)

The mirrored interface of the higher-level EPHT/EMT is created when it is first used, but it is deliberately not adapted on its own.

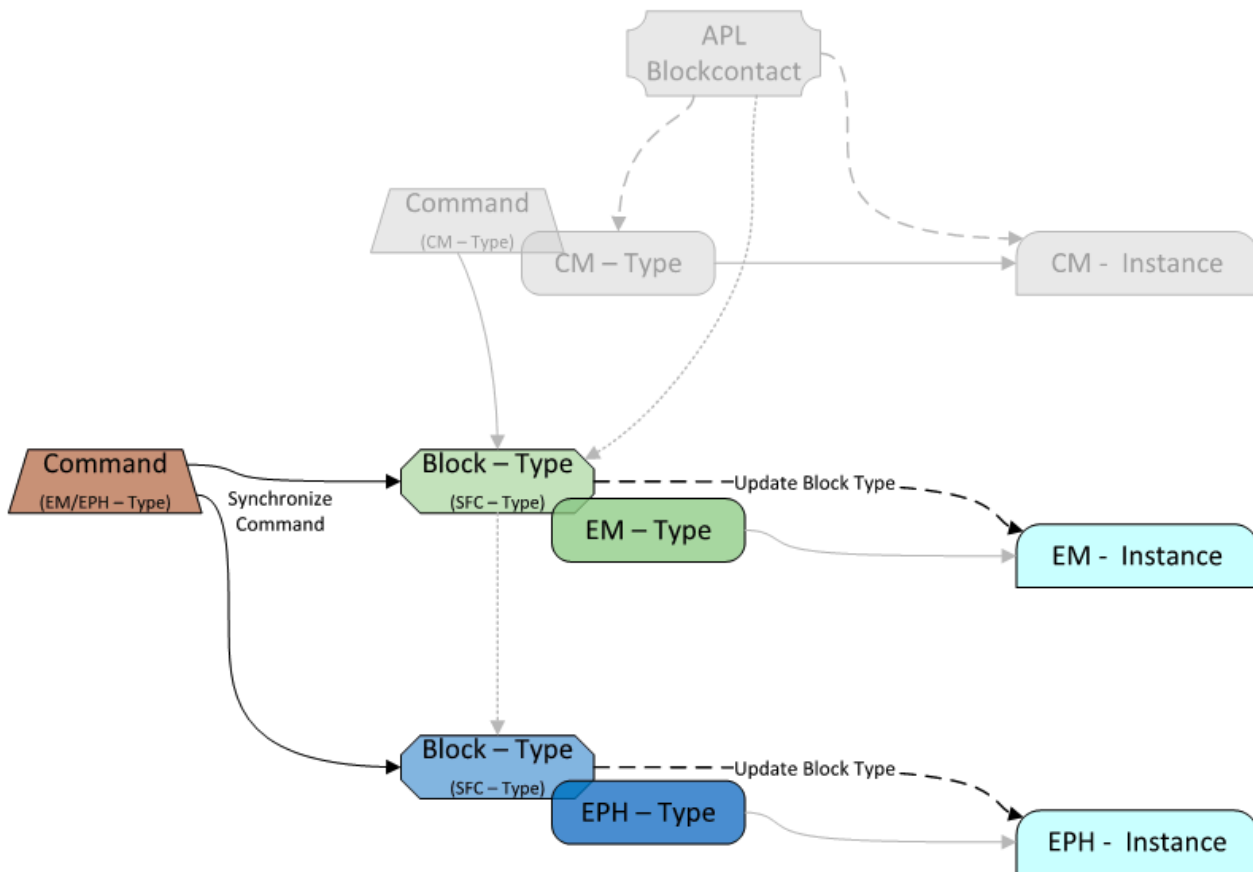
If a new setpoint or parameter is added to the lower-level EM, this new setpoint must also be made known to the higher-level EPHT or EMT so that the block interface is also updated here. For the lower-level EM, the setpoint input and actual value output automatically get the "s7_contact" attribute.

However, this step also changes the block interface of the SFC type. This is why the "Update and synchronize block types" (in the SFC type of the EPHT) and "Block contacts" steps of the charts involved are necessary.



10.4.5 Example 5: Changing global commands (e.g. the CmdStatLib)

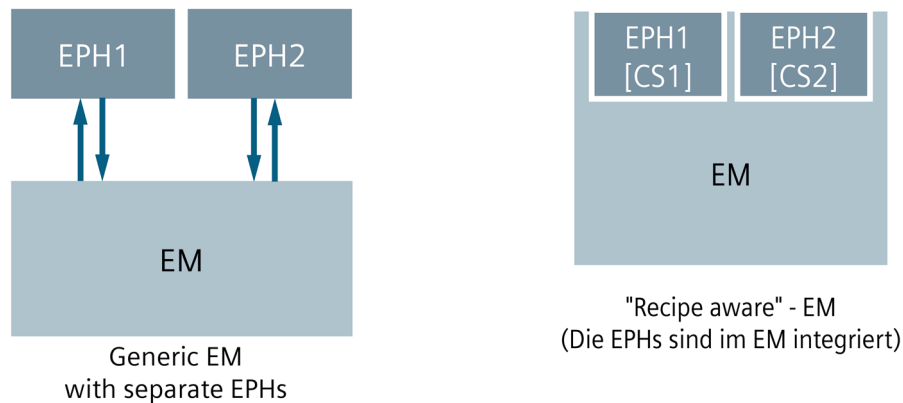
If the implementation of a global command or status defined in the SIMATIC Manager is changed, it must be synchronized to the locations of use. Then the changed SFC types must be inherited by the projects via "Update Block Types". These can be SFC types from both EPHT and EMT.



10.5 Separation between EPH and EM

ISA-88 provides for two basic ways of implementing equipment modules and equipment phases. In the previously described "Recipe Aware" model, the phase logic is already contained in the EM. An EM can also contain more than one EPH. Within SIMATIC PCS 7 / SIMATIC BATCH, the various equipment phases of an EM are implemented by the control strategies.

Alternatively, the equipment phase (EPH) can be implemented separately from the equipment module (EM). This is then referred to as a generic EM. In this case, the EPH is the counterpart to the recipe phase and provides a corresponding state model and the interface for the phase parameters (setpoints).



However, the separation of the equipment phase from the EM offers other advantages:

1. As a result of the separation, the phase is tied to a unit but the EM is not tied to a phase. A central functionality can thus be implemented in an EM and be used from the unit context as a phase.

Several EPHs can use the same EM. In so doing, it is irrelevant whether these EPHs are assigned to the same or different units.

The write accesses can come only from one EPH, but the EM can be monitored from all EPHs assigned to it (e.g. state, current actual values).

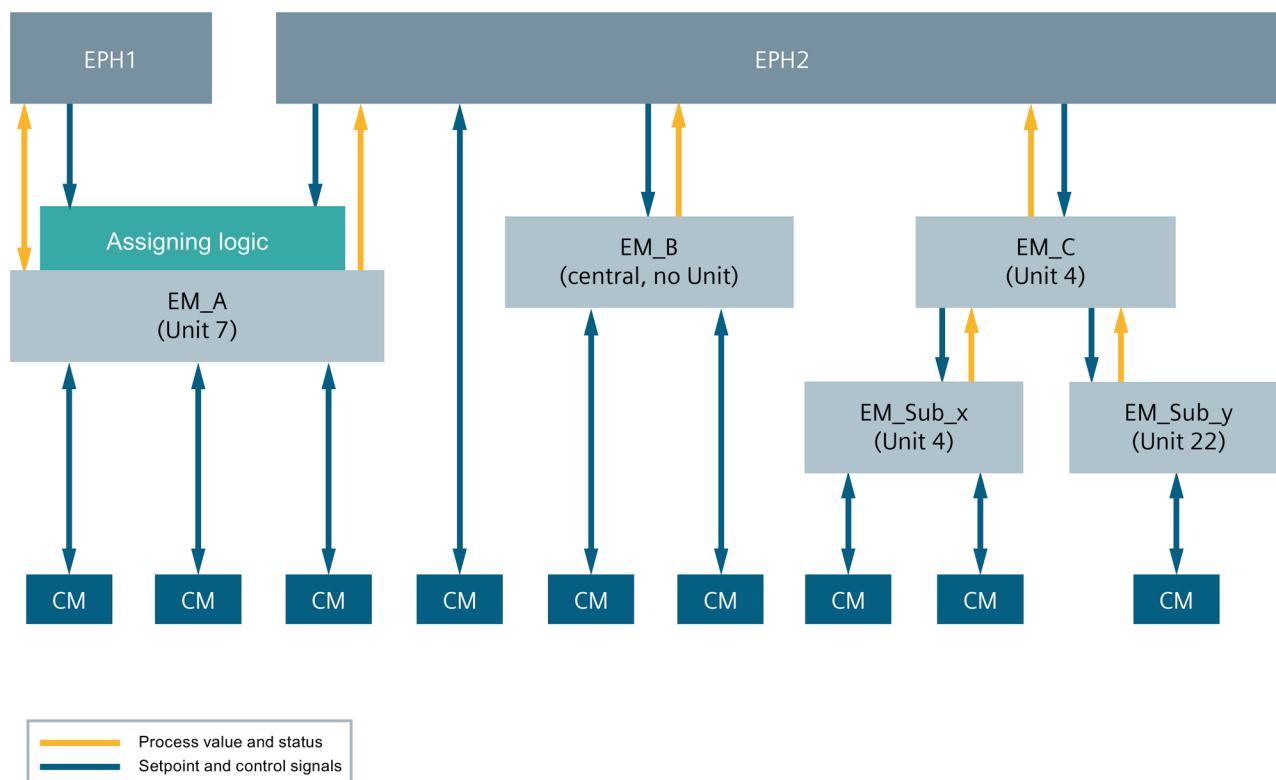
The system thereby automatically creates a high-performance assignment logic including visualization at runtime.

In many cases, the problem of Shared CMs can be circumvented in an elegant manner through Shared EMs.

2. An EPH can also use more than one EM. As a result of this, it is possible to implement individual processes / sequences with clean separation and to run them in a coordinated manner in the phase context without the need for a relatively complex coordination on the recipe level.
3. If desired, EMs can also be cascaded. That is, an EM controls other lower-level EMs.

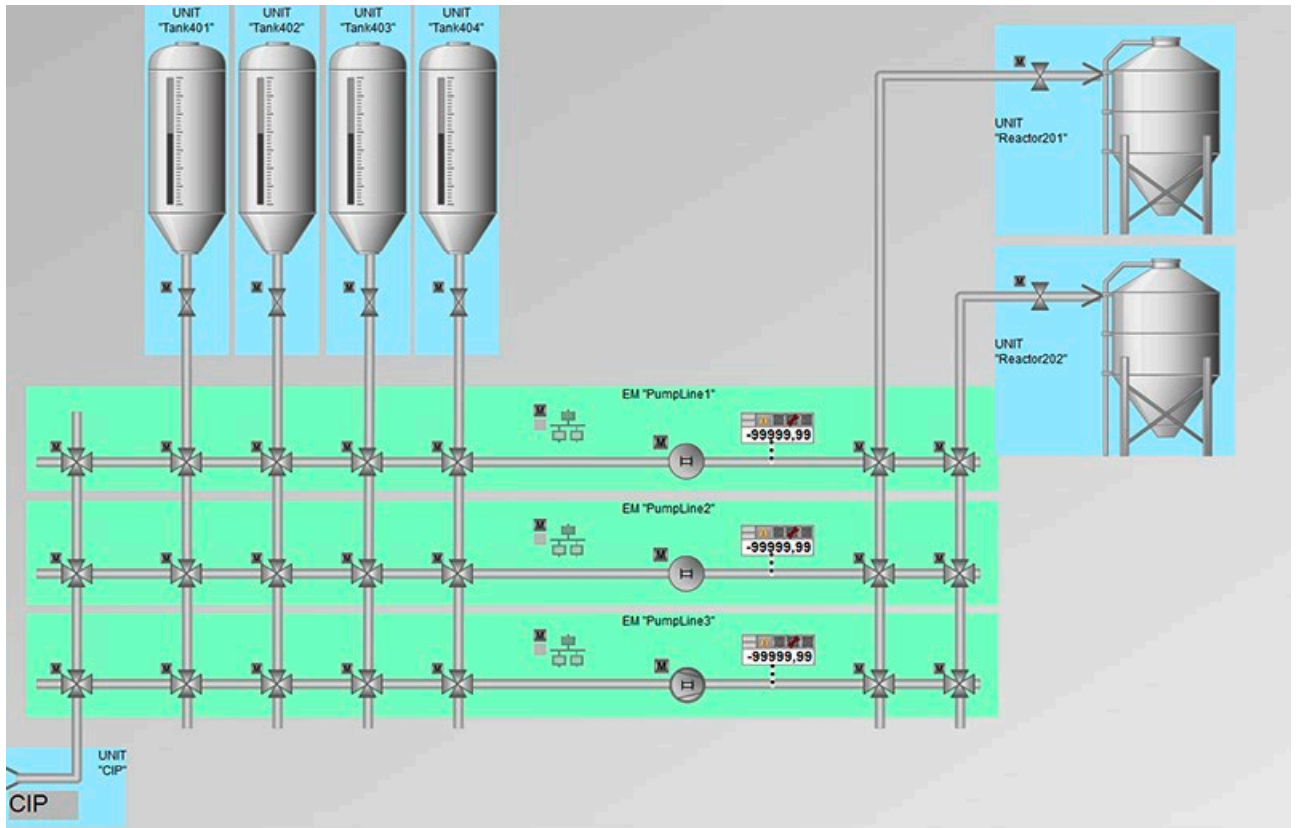
10.5 Separation between EPH and EM

4. All combinations of these three points are possible. As shown in the example in the following figure, EPH2 directly controls a CM and would thus be a "Recipe aware" phase. A shared EM "EM_A" and the cascaded EMs "EM_C" with "EM_Sub_x" and "EM_Sub_y" are controlled as well.



Despite all the advantages of such modularization, it must not be overlooked that coordination can become complex and require significant effort.

The following figure shows an application example in which items 1 to 4 are illustrated.



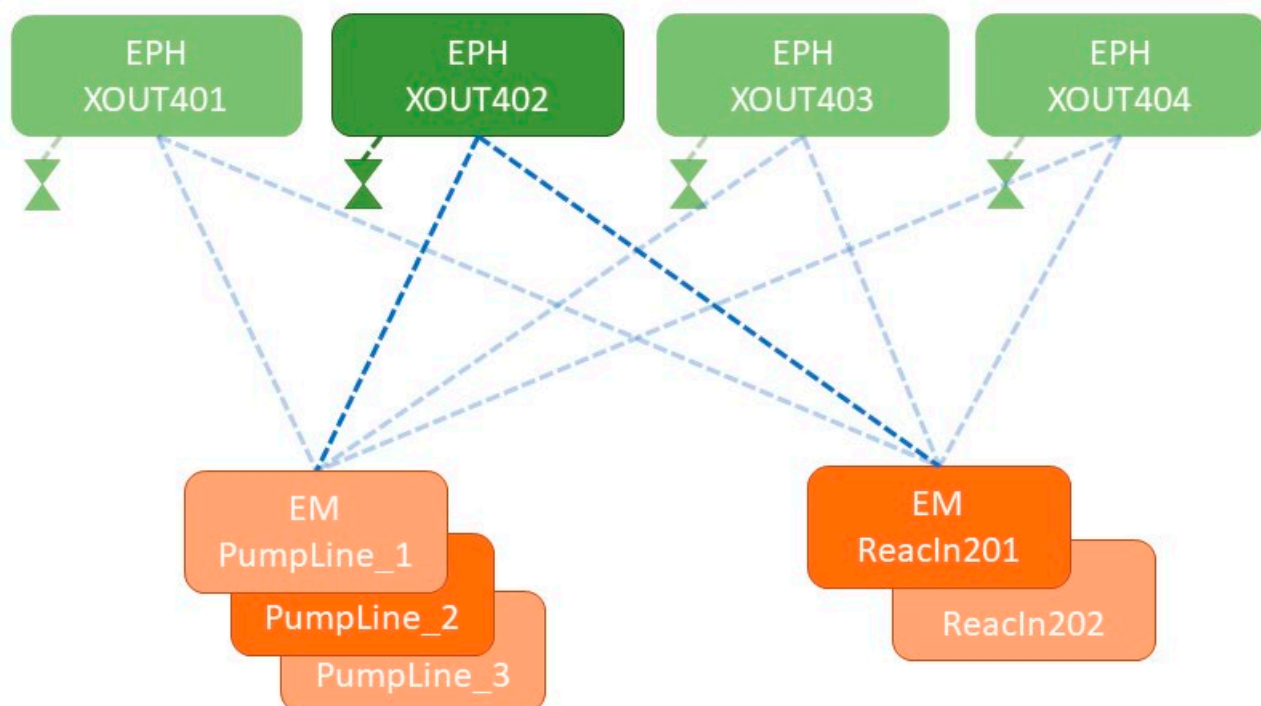
A section of a tank farm is shown in the figure above. The outlet and inlet valves are assigned to the respective unit "Tank_x" or "Reactor_y".

All elements such as valves, pumps and flow meters of a horizontal line are grouped as EM "PumpLine_z". This allows a type with three instances to be developed very effectively due to the same structure.

Because the elements in the green areas are not assigned to a (BATCH) unit, they are also available for everything. The request of such an EM then comes about through an equipment phase assigned to the unit.

For a transport from "Tank402" to Reactor 201, a "Transfer_Out" phase is then started at the "Tank402" unit. For one thing, this controls the outlet valve directly or indirectly in an EM. For another, a search is made for a free pump line ("PumpLine_2" in this example), which is then requested for the transport of the medium to the reactor. If a pump line is available, it is allocated for this operation and is no longer available for another transport.

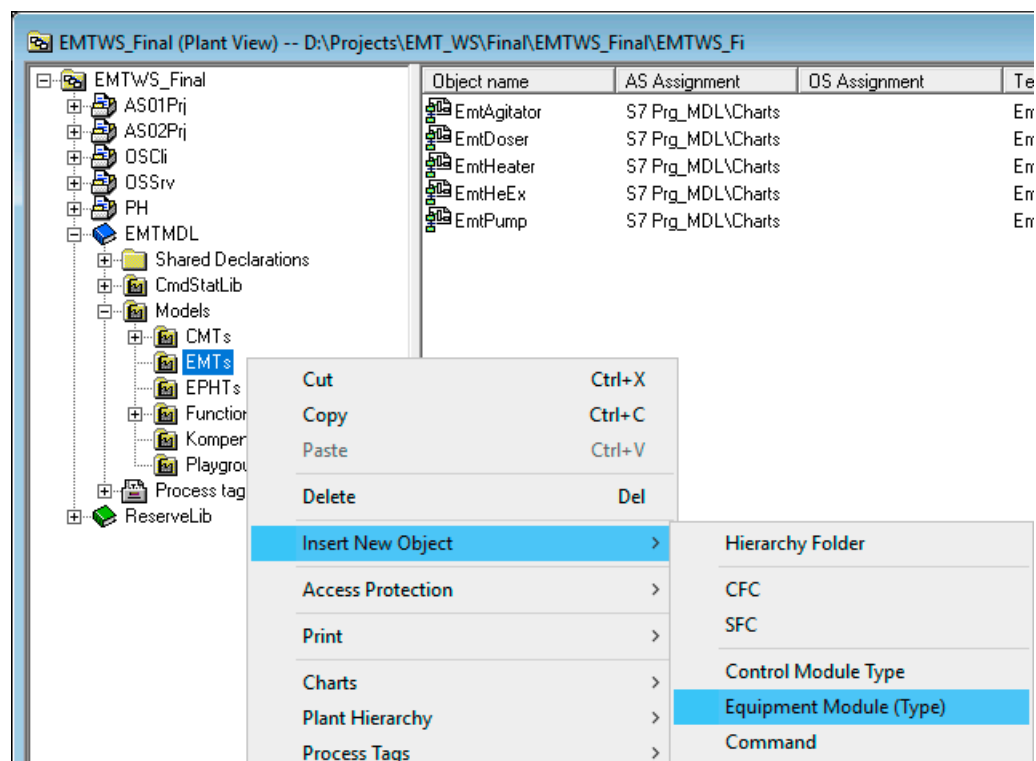
The EM for the reactor inlet ReacIn201 is assigned and controlled in the same way.



10.5.1 EM types

The basis for the separation between EPH and EM is the use of EPH types and EM types. In this case, EM types are created and configured in the same way as described previously for the EPH types.

To create a new EMT, a shortcut menu in the plant hierarchy has the entry "Insert New Object > Equipment Module Type". Afterwards, CMTs and roles are defined in the EMT, and the contained SFC and its step sequencers are configured.



10.5.2 Using EMs in EPH

Once an EMT is created, it is also available in the catalog under "Templates".

You follow the same procedure as for use of CMTs in EMTs or EPHTs to enable control of an EM later by an EPH.

Specifically, you open an EPHT, for example, and use drag-and-drop to move the desired EMT into the highest node in the technological I/Os of the EPHT.

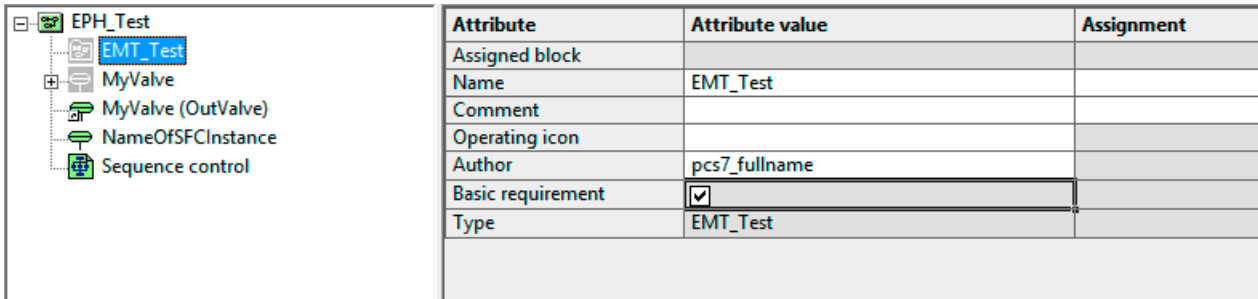
The screenshot displays the CFC software interface for configuring an EPH_Test module. The left pane shows a tree view of the Demo8301_MDL project, with EMT_Test selected under MyEMTs. The right pane shows the EPH_Test module's configuration table and a list of I/Os.

Attribute	Attribute value	Assignment
Assigned block		
Name	NameOfSFCInstance	EPH_Test\NameOfS
Comment		
Operating icon		
Optional	☐	
Author	pcs7_fullname	
Function identifier		
Function	☐	
Function name		
Basic requirement	☐	
Type	EPH_Test	

The I/O list for EPH_Test is as follows:

Input	Output
0- AUT	QAUTMAN
0- MAN	IDLE
0- START	STARTING
0- COMPLETE	RUN
0- HOLD	READY_TC
0- RESUME	COMPLETI
0- ABORT	ERROR_CO
0- STOP	COMPLETE
0- RESTART	HOLDING
0- RESET	HELD
En Start	ENSTART
0- LOCKCOMP	ERROR
0- LOCKHOLD	HELD_ERR
0- LOCKABOR	RESU_ERR
0- LOCKSTOP	ABORTING
0- LOCKERRO	ABORTED
0- CS	STOPPING
T- SCT	STOPPED
0- SCT_IAC	OP_ERR
InstrOut	INSTROUT
0- CYCLEXEC	EXEC_ERR
0- TIMEMON	ERRG
EnAcquir	ENACQUIR
	I_OPRQG
	S_ERRG
	QCS
	OutValve

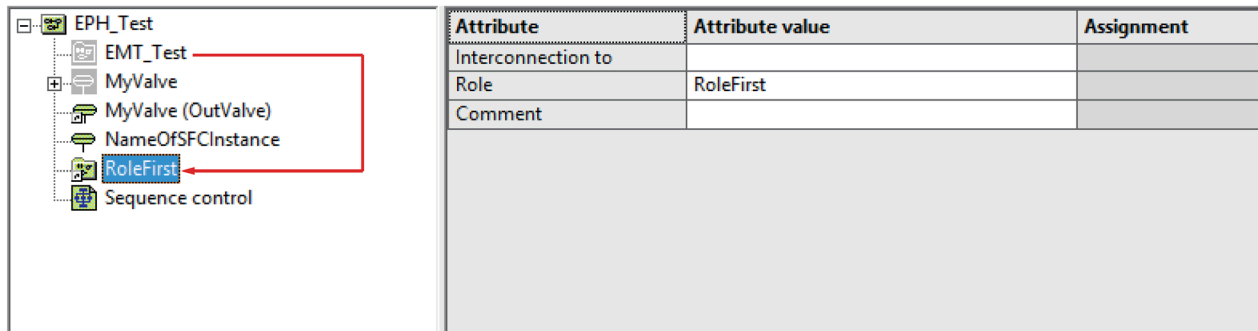
In contrast to using CMTs, an EMT is defined as a "Basic requirement" by default, and this property cannot be changed.



The screenshot shows a tree view on the left with the following items: EPH_Test, EMT_Test (highlighted), MyValve, MyValve (OutValve), NameOfSFCInstance, and Sequence control. To the right is a table with the following data:

Attribute	Attribute value	Assignment
Assigned block		
Name	EMT_Test	
Comment		
Operating icon		
Author	pcs7_fullname	
Basic requirement	☒	
Type	EMT_Test	

In the next step, an "Equipment Module Assignment" object corresponding to the CMT use is inserted, which specifies the role for the EMT.

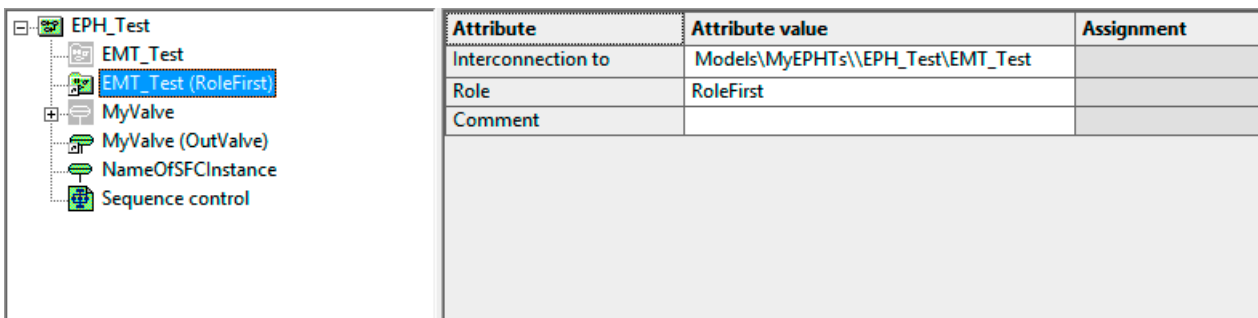


The screenshot shows the tree view with an additional item, RoleFirst, added below NameOfSFCInstance. A red box highlights the RoleFirst item and its corresponding row in the table. The table data is as follows:

Attribute	Attribute value	Assignment
Interconnection to		
Role	RoleFirst	
Comment		

Similar to the control module assignment, the referenced object, the EM type in this case, is also assigned to the role via a pull-down menu in the equipment module assignment.

This is also possible graphically by dragging the EMT onto the role with the mouse.

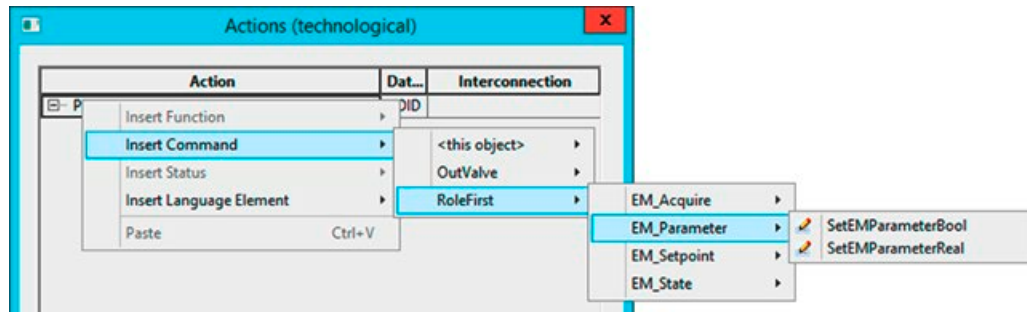


The screenshot shows the tree view with a new item, EMT_Test (RoleFirst), added below the original EMT_Test. The table data is as follows:

Attribute	Attribute value	Assignment
Interconnection to	Models\MyEPHTs\EPH_Test\EMT_Test	
Role	RoleFirst	
Comment		

Several roles could also be assigned to the same EMT. An example would be several independent dosing devices that are controlled in a coordinated manner by the same phase or the three pump lines in the example above.

CmdStatLib provides commands and states for accessing a lower-level EM.



10.5.3 Commands and states for control of lower-level EMs

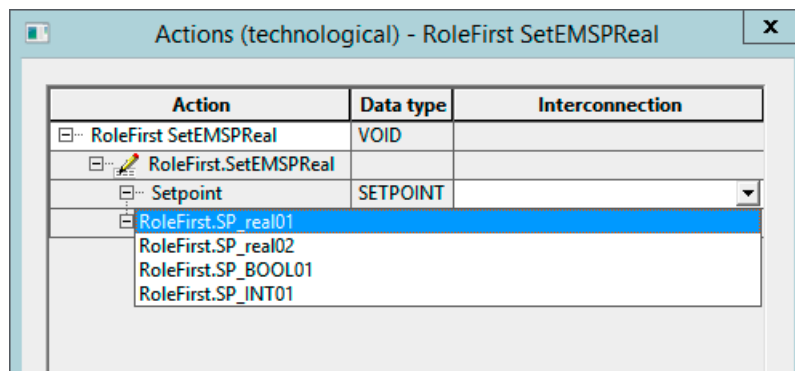
EMs, commands and states are provided with the SFC library for control of lower-level. If these commands and states are used in the project, they must be copied from the SFC library to the master data library of the multiproject. Specifically, the commands have the following function:

- Allocation of the EM ("EM_Acquire" group)
- Handling of parameters ("EM_Parameter" group)
- Handling of setpoints ("EM_Setpoint" group)
- Handling of state logic ("EM_State" group)

Using the "SetEMSPReal" command, for example, a setpoint of REAL data type is supplied with a value.

After selecting the command, all setpoints of the lower-level EM are displayed in a drop-down list and the desired setpoint can be selected.

By selecting from a list you avoid errors when typing or transferring the setpoint.



10.5.4 Workflow for control of lower-level EMs

The lower-level EMs can be controlled as follows:

1. Optional: Query of allocation status of the desired EM
2. Allocation request and allocation check
3. Supply of the EM with setpoints and parameters
4. Start of the sequence logic and reaction to status change
5. Readback of actual values, if necessary

In automatic mode, setpoints are not directly applied at the automatic input. Instead, they must be checked for limit violations and applied as valid setpoints. Depending on the desired behavior, the following commands can be used for this:

- DisableEMSPCheck
- EnableEMSPCheck
- EnableEMSPCheckInRun

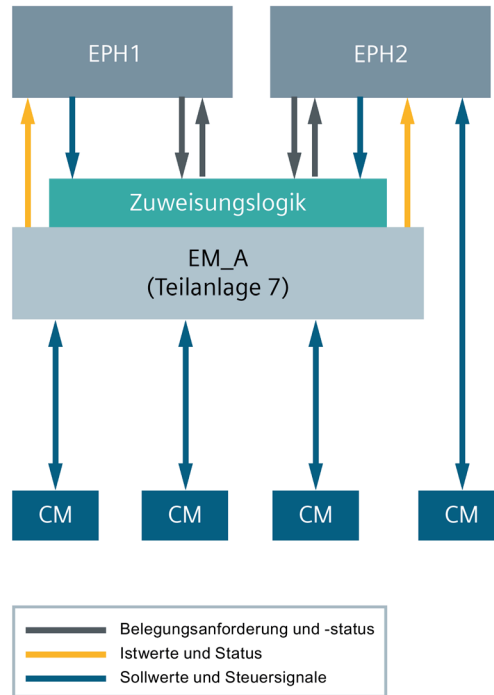
These commands act on the familiar SFC inputs PARAM and TAKESP. You can find more details on this in the SFC documentation.

Accordingly, the preconditions should first be checked before applying a command to the sequencing logic. This pertains, in particular, to the start disable (output QDIS_START of the SFC), which can be queried using the "IsEMDisstart" status.

10.5.5 Use of a "Shared" EM by several EPHs (MARC)

If more than one EPH can control an EM, the source of the control signals and, in particular, setpoints and parameter assignments must be explicitly specified.

This requires an assignment logic, which is examined in more detail in the following. Actual values and states can be read (passively) by all EPHs.



If you wanted to implement this assignment logic in a conventional way, individual charts or project-specific blocks would be created separately for each (destination) EM because the interfaces differ for each EM, especially with respect to setpoints. The effort needed to create these and, in particular, to maintain them, would be enormous.

This functionality is therefore provided by the system without the need for any actions by the configuration engineer.

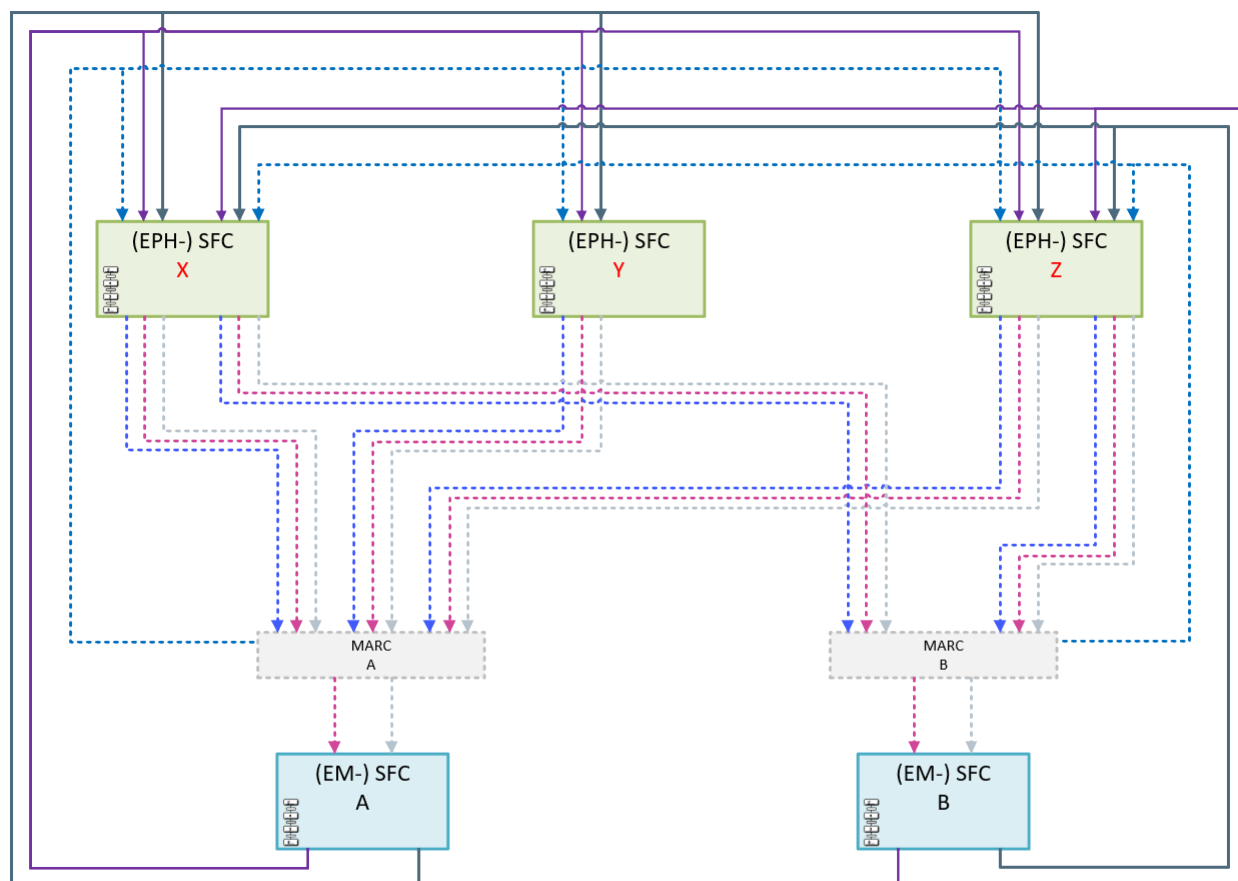
When generating the code, a "MARC FC" is created for each EM instance to be controlled, which forms the link between EPH instances and EM instances.

MARC stands for Multiplexing, Acquisition, Release and Control, which also describes all its tasks. The underlying EM is allocated or released and the control signals and data are multiplexed. The MARC FCs and their interconnections are not visible in the CFC. They can only be found in the block folder.

The size of the MARC FC depends on the number of potential EPH instances controlling the EM and the number of setpoints and parameters to be multiplexed.

The following figure shows a constellation where EM "A" can be controlled by the three phases "X", "Y" and "Z", while EM "B" can only be controlled by "X" or "Z". From this, it can also be seen that for each EM instance an individual assignment logic (MARC) must be generated by the code generator and internally interconnected.

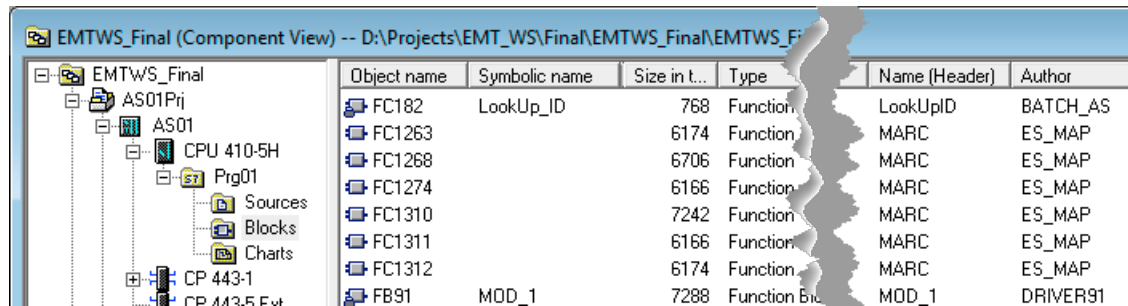
Only the interconnections with solid lines are visible in the CFC, the dashed lines are not.



Erläuterungen / Legende

- Anfragen an die Multiplexer-Logik / den MARC (AcquireType)
- Feedback vom MARC (AcquireState)
- SFC-Befehle auf die SFC-Zustandslogik (z.B. Start, Complete)
- Feedback der SFC-Zustandslogik (z.B. Run, Completing)
- Sollwerte und Parameter zum EM
- Istwerte zu Sollwerten

Even if the MARC FCs are not placed in any CFC, they are included in the code and also occupy memory, as can also be seen in the following screenshot:

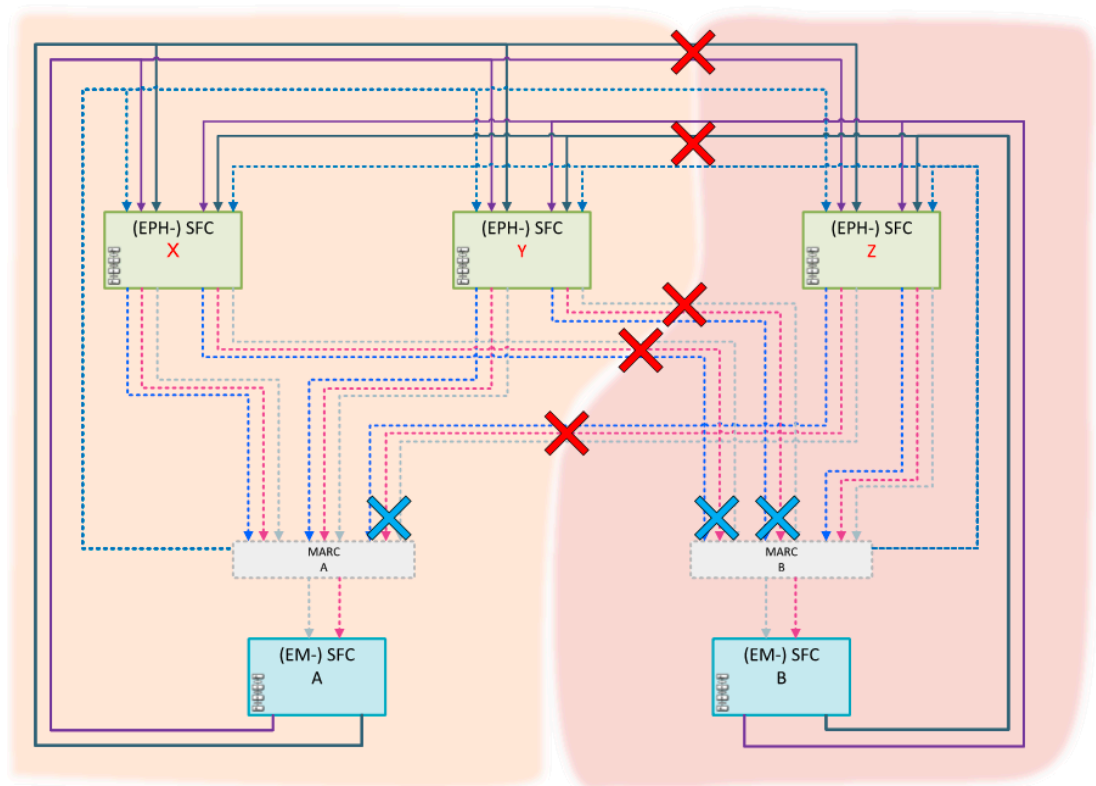


Object name	Symbolic name	Size in t...	Type	Name (Header)	Author
FC182	LookUp_ID	768	Function	LookUpID	BATCH_AS
FC1263		6174	Function	MARC	ES_MAP
FC1268		6706	Function	MARC	ES_MAP
FC1274		6166	Function	MARC	ES_MAP
FC1310		7242	Function	MARC	ES_MAP
FC1311		6166	Function	MARC	ES_MAP
FC1312		6174	Function	MARC	ES_MAP
FB91	MOD_1	7288	Function	MOD_1	DRIVER91

It must also be noted that block contacts are created by each role and this is reflected in the memory requirements for the instance data modules. The number of interconnections in the SCL code also increases.

Convenient visualization and operator input capability / interaction at runtime in the OS is also provided (section "Visualization of allocations (Page 131)").

However, in current versions, EPH and EM can only be separated within an AS, i.e. all EM that are controlled by an EPH must be located in the same AS.



Red cross: Cross-AS connection cannot be created.

Blue cross: Inputs of the MARC FC cannot be created because the cross-AS or even cross-project information is not even available or configurable.

10.5.6 Allocation and allocation strategies

If you want to make use of an EM by an EPH, the EM must be allocated beforehand. To do so, you send an allocation request to the EM and receive an allocation status in return. In some cases, it is also recommended that the status be queried before the allocation request.

This is especially true if an EM can potentially be controlled by only one EPH. For convenience, in this 1:1 relationship, the EM is allocated by default after an AS complete download, which is also referred to as "Acquired by default".

The following commands from the SFC library are available for the allocation:

EM_Acquire	▶	✎	EMAQAcquire
EM_Parameter	▶	✎	EMAQAcquireId
EM_Setpoint	▶	✎	EMAQAcquireL
EM_State	▶	✎	EMRelease
		✎	SetEMAQAcquireId
		✎	SetEMAQPriority
		✎	SetEMAQWaitTmMax

States are also available for the technological configuration:

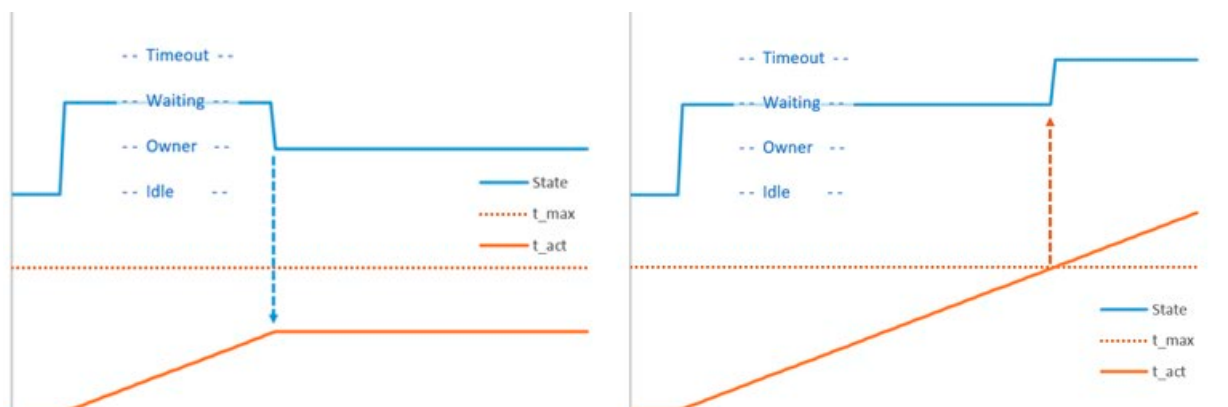
EM_Acquire	▶	📄	IsEMAQAvailable
EM_Setpoint	▶	📄	IsEMAQConfigError
EM_State	▶	📄	IsEMAQDeactivated
		📄	IsEMAQNotAvailable
		📄	IsEMAQOwner
		📄	IsEMAQSnatched
		📄	IsEMAQTimeout
		📄	IsEMAQWaiting

Accordingly, these commands and states are also available in the SFC visualization in the OS.

The simplest type of allocation would be to use the "EMAQAcquire" command and allocation type "Now". Immediately thereafter you receive the feedback "Not Available" or "Owner", if it was allocated by the requester.

For the allocation type "Wait", the requester attempts to allocate the EM for the configurable time "WaitTmMax". As long as the maximum time has not expired, the "Waiting" status is returned to the requesting EPH. As soon as the EM can be allocated, the "Owner" status is returned. After expiration of the maximum time, the "Timeout" status is returned. You need to keep in mind here that even the "Timeout" feedback is only a status and does not trigger further reaction. If a reaction is desired, a corresponding exception handling must be configured (e.g. error message via "GenerateMessageY" or holding of the sequencer by "SetInternalCMDHold").

The behavior is shown again below as a timing diagram.

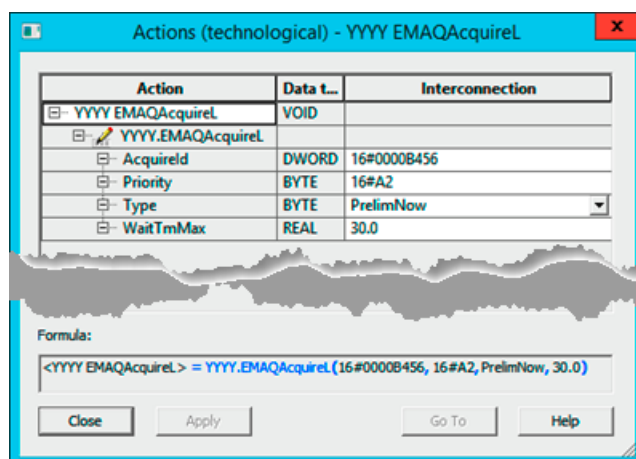


Another type of allocation is adoption of an EM by another EPH using the "Takeover" allocation type. The allocating EPH up to that point then receives the allocation status "snatched" because the EM was snatched from it.

In addition, a safe status can also be specified for the allocation. This safe status can be defined from a combination of operating state and UserStatus of the EM.

Priorities and "friendships" can also be defined. In this case, a requesting EPH whose AcquireID is the same as the currently allocating EPH has more rights than if its Acquire ID were different.

The combination of these possibilities is presented in the next figure using the "EMAQAcquireL" command:



Enumerations for allocation type and allocation status

In the code, allocation strategies and allocation status are implemented as a data type byte. For intuitive operation, the corresponding enumerations ("ES_AcquireTypes" or "ES_AcquireStates") are stored in the CmdStatLib and are also displayed in the commands as easy-to-read text.

ES_AcquireTypes

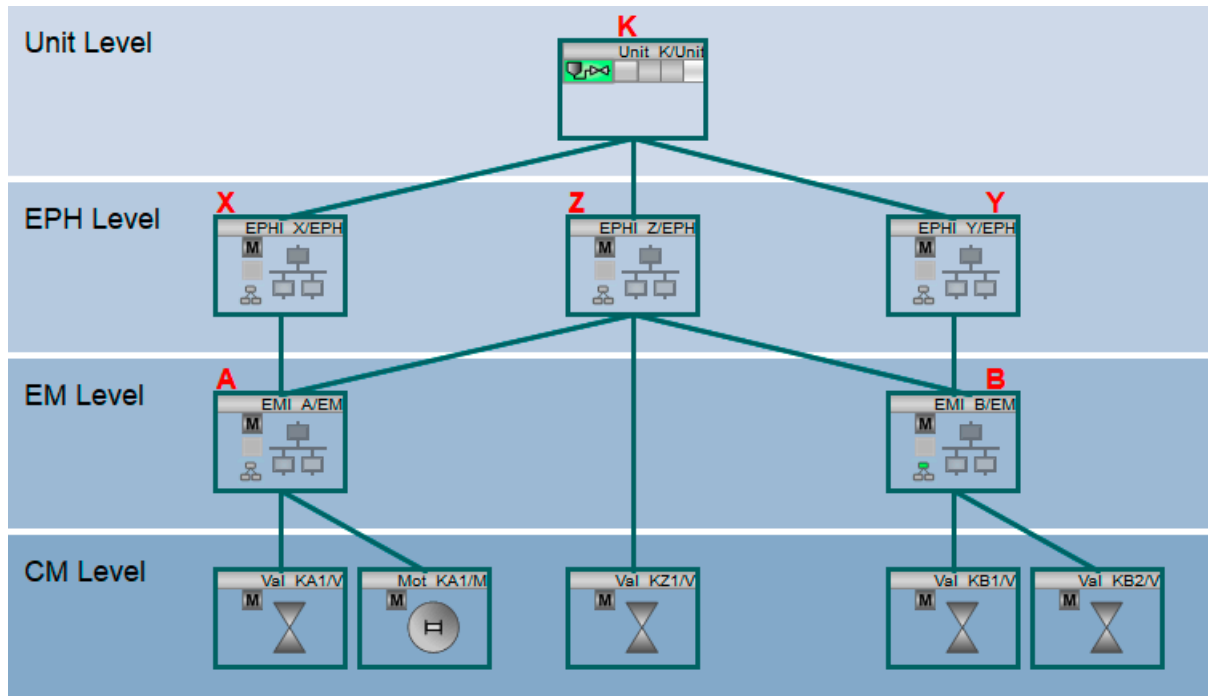
- [0] No Request
- [1] PrelimWait
- [2] PrelimNow
- [3] Wait
- [4] Now
- [5] Takeover
- [6] ForceSafe
- [7] ForceNow

ES_AcquireStates

- [0] Idle
- [1] Owner
- [2] Waiting
- [3] NotAvailable
- [4] Deactivated
- [5] Timeout
- [6] Snatched
- [7] ConfigError

10.5.7 Visualization of allocations

The following schematic model is used for the visualization:



10.5 Separation between EPH and EM

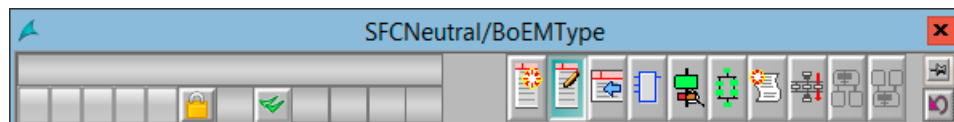
Below Unit K are the Equipment Phases X, Y and Z, which are implemented by the EPH instances "EPHI_X", "EPHI_Y" and "EPHI_Z". Below that are the Equipment Modules A and B, which are implemented by the EM instances "EMI_A" and "EMI_B".

The following relations exist:

- EPH X can control EM A
- EPH Z can control EM A, EM B and CM Z1.
- EPH Y can control EM B.
- EM A can be controlled by EPH X or EPH Z.
- EM B can be controlled by EPH Z or EPH Y.

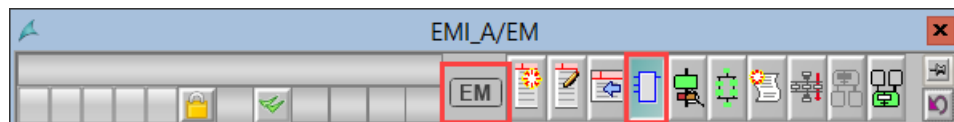
The upper part of the faceplate for the SFC indicates whether the SFC belongs to an "EM" or "EPH" (based on EMT or EPHT) or is "neutral".

Neutral:



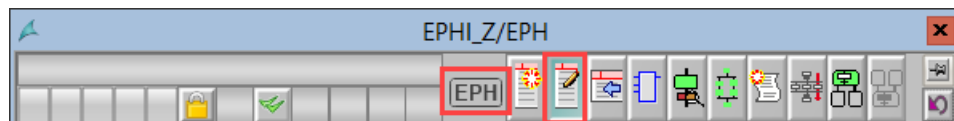
EM:

Additional tabs are available for both cases. There is a tab for the higher-level EPH for an EM:



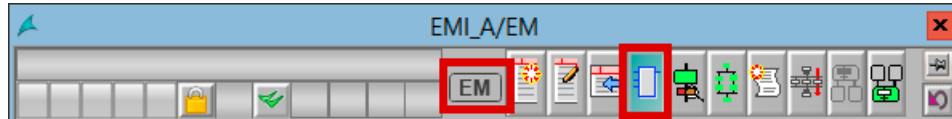
EPH:

There is a tab for the lower-level EMs for the EPH:

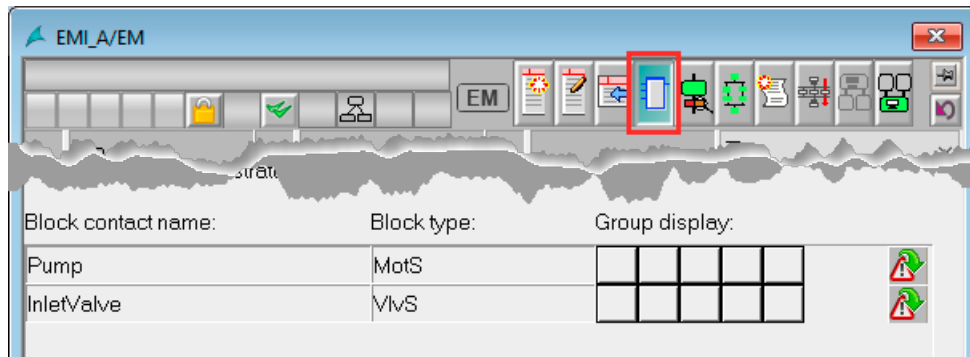


For EM

The "Block contacts" tab is available on the faceplate of the EMs.

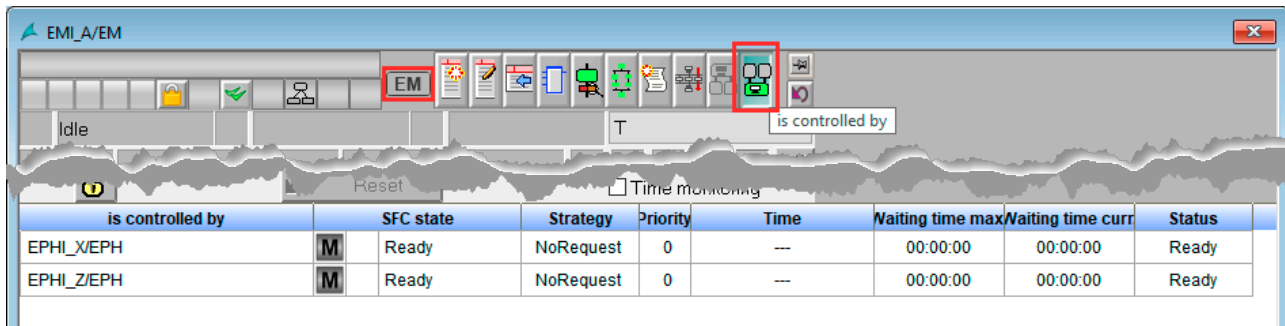


In the block contacts, the control modules are displayed and can be jumped to from there via the loop-in, in which case the faceplate is closed, the picture containing the control module is opened and the block icon of the CM is marked.



The EM tab "is controlled by" is only available for EMs.

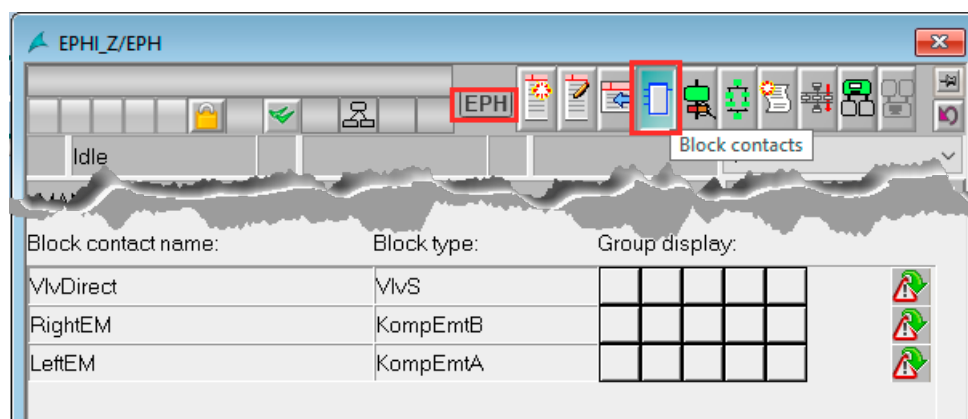
All EPHs that can control the EM are displayed here.



For EPH

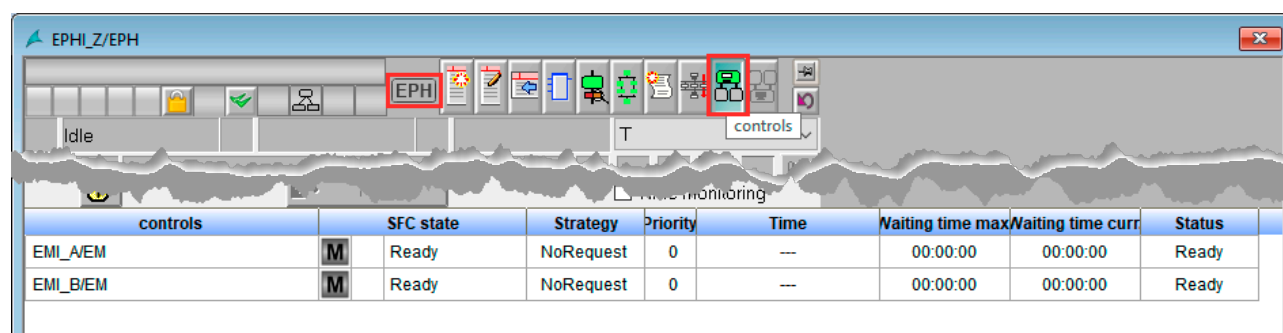
The "Block contacts" EPH tab is shown in the following.

In the block contacts, the control modules and equipment modules are displayed and can be jumped to from there via the loop-in, in which case the faceplate is closed, the picture containing the control module or equipment module is opened and the "Block icon" of the CM is marked.



The EPH tab "controls" is available for EPHs.

All EMs that can control the EPH are displayed.



For EM and EPH: Allocation logic

The row for an EM or EPH is not only used to display the current states but also allows operator control.

controls		SFC state	Strategy	Priority	Time	Waiting time max	Waiting time curr	Status
EMI_A/EM	M	Ready	NoRequest	0	---	00:00:00	00:00:00	Ready
EMI_B/EM	M	Ready	NoRequest	0	---	00:00:00	00:00:00	Ready

Pressing the area highlighted in green of a line opens the faceplate of the object.

An additional dialog window for editing is opened in the right pane highlighted in red, which must always be interpreted from the perspective of the higher-level object:

Change allocation data

EPHI_Z/EPH

Strategy: NoRequest

Priority: 0 (0-255)

Max. waiting time: 00:00:00

Additional allocation information

Allocation ID: 0x00000000 (0,0)

Safe state: Show details

OK Cancel

The current allocation command and type can be changed using a drop-down list. This allows the EM to be manually enabled by selecting the "NoRequest" strategy.

Change allocation data

EPHI_Z/EPH

Strategy: PrelimNow

Priority: NoRequest

Max. waiting time: PrelimWait

Allocation ID: Wait

Safe state: Now

Takeover

ForceSafe

ForceNow

Show details

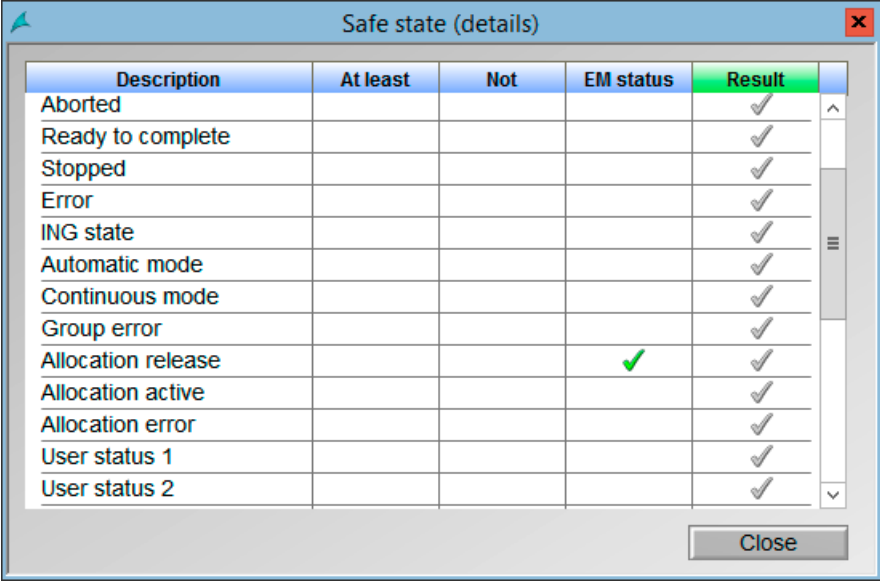
OK Cancel

The status of the EM that is defined as the safe state and which status is the current status is also displayed.

10.5 Separation between EPH and EM

The "At least" column for the EM block instance is thereby determined by the input DWORD "AC_SafeOrMsk" and the "Not" column by the input DWORD "AC_SafeNotMsk".

The "VSTATUS" status is used internally for comparison. Details regarding "VSTATUS" can be obtained from the SFC documentation.



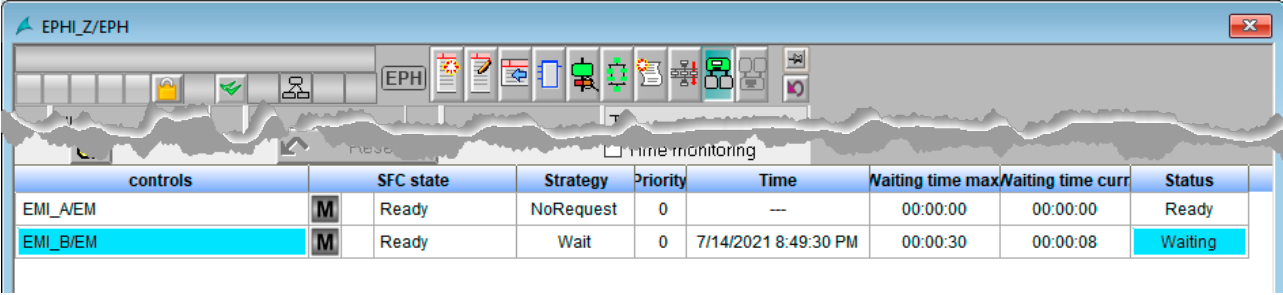
Description	At least	Not	EM status	Result
Aborted				✓
Ready to complete				✓
Stopped				✓
Error				✓
ING state				✓
Automatic mode				✓
Continuous mode				✓
Group error				✓
Allocation release			✓	✓
Allocation active				✓
Allocation error				✓
User status 1				✓
User status 2				✓

Example: Time response of the allocation

The time response for the allocation is illustrated in the following example.

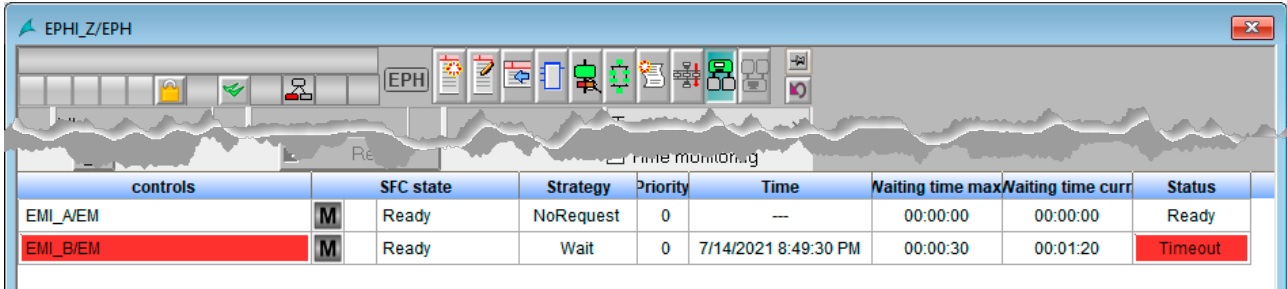
The phase attempts to allocate the still allocated EM with the strategy/allocation type "Wait".

During this time, the "Waiting" status is displayed along with the maximum time and the already elapsed waiting time:



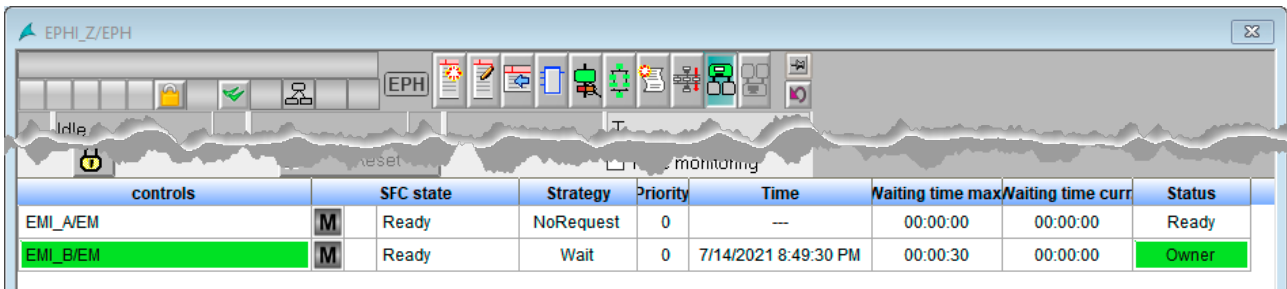
controls	SFC state	Strategy	Priority	Time	Waiting time max/Waiting time curr	Status
EMI_A/EM	M Ready	NoRequest	0	---	00:00:00 00:00:00	Ready
EMI_B/EM	M Ready	Wait	0	7/14/2021 8:49:30 PM	00:00:30 00:00:08	Waiting

After expiration of the maximum time, the request changes to "Timeout" status, while the time elapsed since the request continues running: The elapsed time is also available at the EPH and can be evaluated.



controls	SFC state	Strategy	Priority	Time	Waiting time max	Waiting time curr	Status
EMI_A/EM	M	Ready	NoRequest	0	---	00:00:00	Ready
EMI_B/EM	M	Ready	Wait	0	7/14/2021 8:49:30 PM	00:01:20	Timeout

If the EM can be allocated within the time interval, the EPH receives the "Owner" status.



controls	SFC state	Strategy	Priority	Time	Waiting time max	Waiting time curr	Status
EMI_A/EM	M	Ready	NoRequest	0	---	00:00:00	Ready
EMI_B/EM	M	Ready	Wait	0	7/14/2021 8:49:30 PM	00:00:00	Owner

10.5.8 Special case of only one EPH instance for the EM instance

Even if the lower-level EM can be controlled by only one phase / EPH (1:1 relationship), a MARC is still built in.

However, the phase occupies the EM after a complete download of the AS to make it easier for the user. However, if the allocation is deliberately removed, it must be requested again. This should be taken into account in the code of the EPH SFC.

Since the allocation logic reacts to changes of the AcquireType, a Command EMRelease may not be executed because the AcquireType already has the value "0 (No Request)."

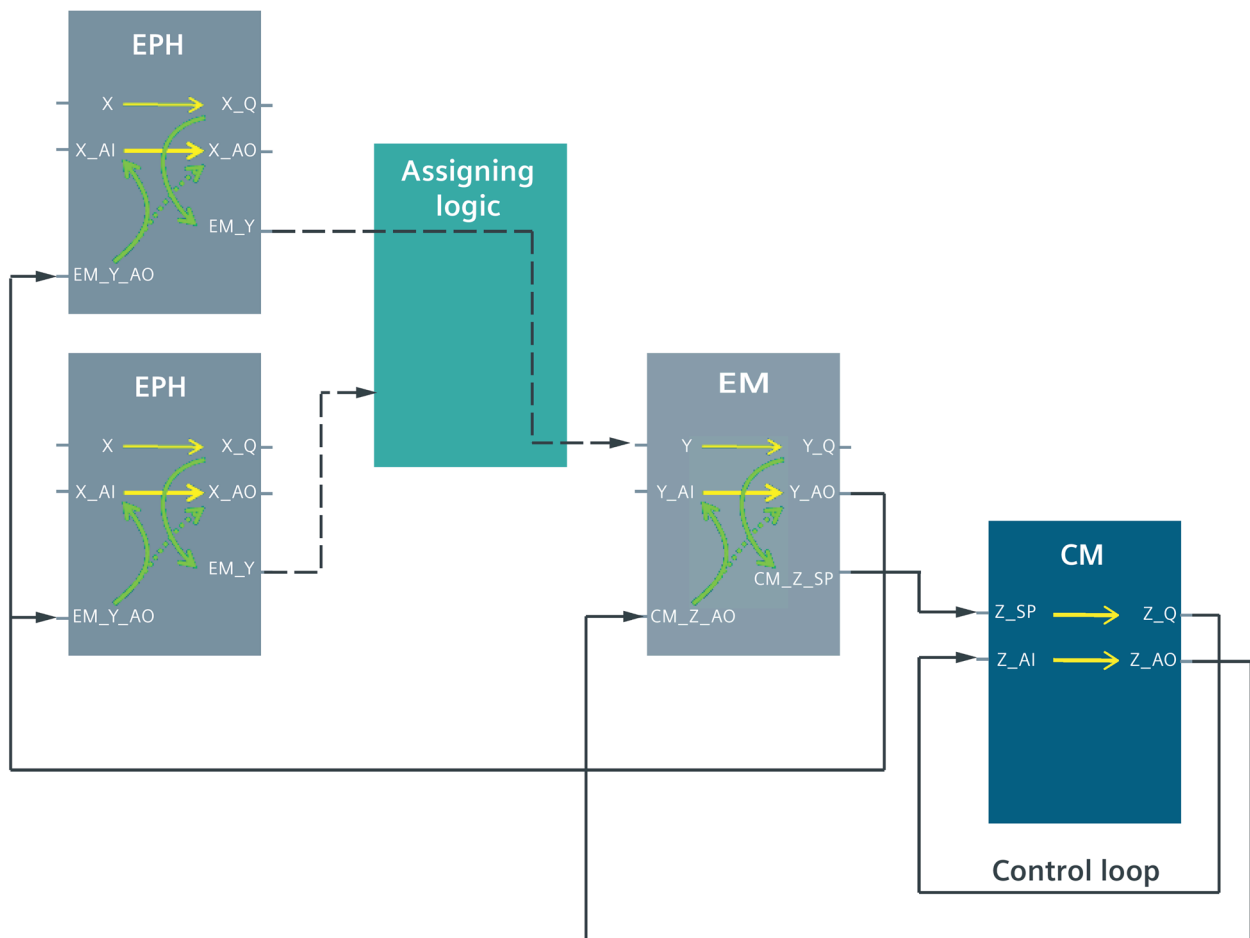
10.5.9 Routing of values

In many cases, setpoints and actual values are to be forwarded directly without a specific reference to a step. An example would be the temperature of a reactor, the actual value of which is to always be read at the phase.

Possibly, there should be the capability to change the setpoint at any time. This is necessary if the setpoint is to be actively changed while the recipe phase is running in BATCH mode. Direct interconnection would be possible only with limitations.

In the following example, an EPH setpoint X is to be sent to the controller setpoint Z_SP as quickly as possible via the lower-level EM (also as setpoint Y) and the actual value Z_AI is to be read back.

For simplification, the controller setpoint was directly fed back to the controller actual value as the controlled system.

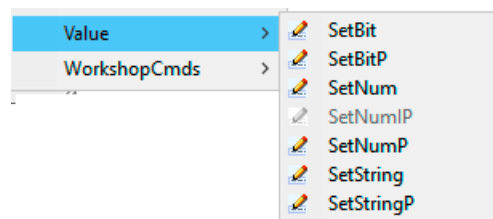


The setpoint "X" at the input of the EPH is transferred to the setpoint output "X_Q" after an internal validity check, depending on the SFC settings. The actual value "X_AI" is transferred to the "X_AO" output. These assignments (yellow lines) take place before the sequencers are called.

For the SFC setpoint to be routed to the lower-level object (EM or CM), the valid setpoint must be passed to the corresponding output for the lower-level object.

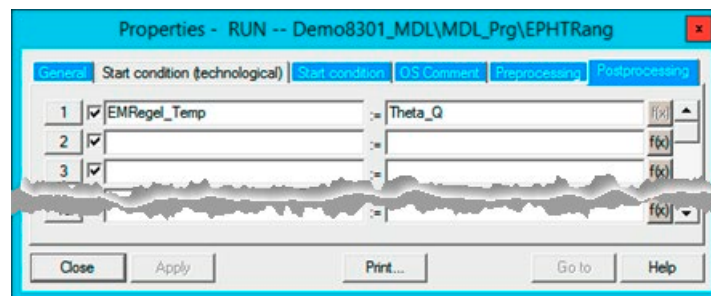
Because the values are to be forwarded at each call of the SFC (in each OB cycle), the sequencer pre-processing or sequencer post-processing lends itself to this.

Since PCS 7 V9.1, sequencer preprocessing can also be technologically configured. Either Initialization or Processing may be configured in the command definition to ensure clear recognition of the assignments to be executed. Termination is always ignored. To avoid accidental misuse, commands that cannot be used are displayed in gray (SetNumIP).



As a project-specific convention, set values in the sequencer postprocessing or also "Post-processing (technological)", for example, are directed to the lower-level object:

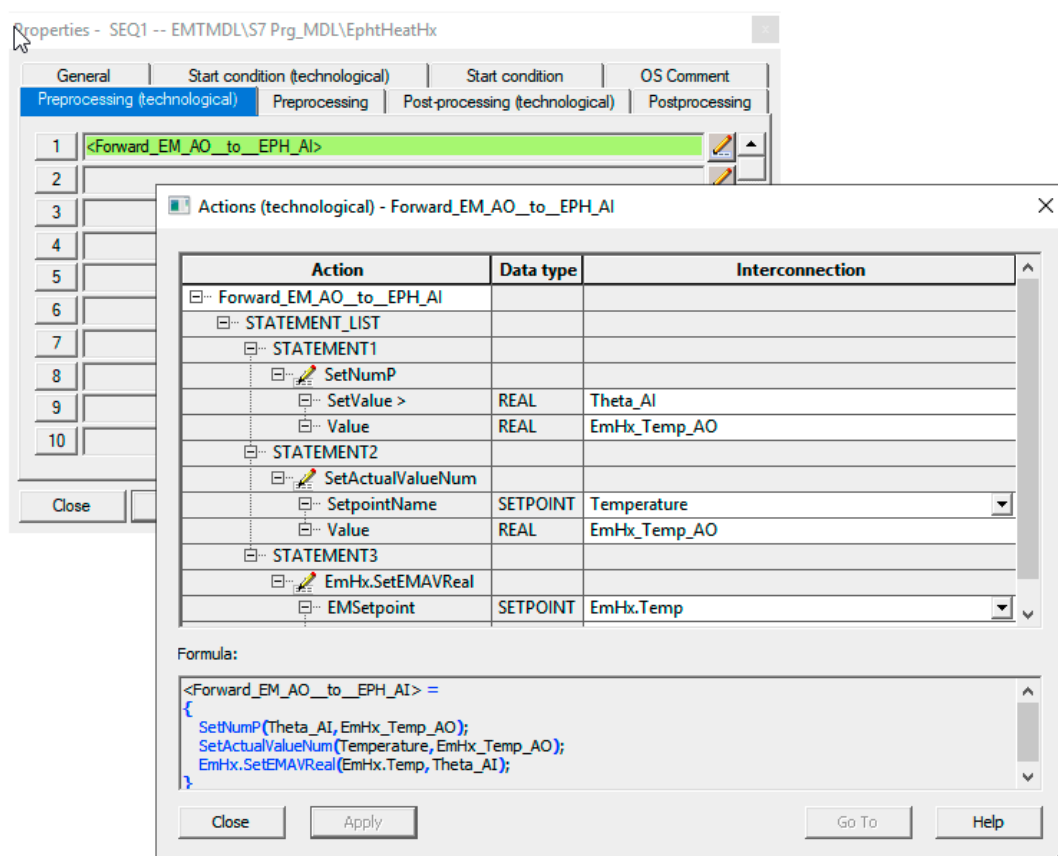
- EPH: The valid "Theta_Q" setpoint of the EPH is written to the "Temp" setpoint input of the lower-level EM "EMController".
- EM: The writing of the valid setpoint of the EM to the CM takes place in the same way.
- For a better distinction, the routing of the setpoint is shown non-technologically, as it was still required at the beginning of the technological configuration, because the "Postprocessing(technological)" tab did not exist yet.



Actual values are processed, for example, in the "Pre-processing (technological)" sequencer:

- EPH: The actual values of the EM must now be written to the actual values of the EPH.
- EM: The CM actual values are routed to the EM actual values in the same way.
- In the screenshot, three alternatives are shown in one action to save space:
 - "SetNum" is used to assign from one block I/O to another.
 - With the "SetActualValueNum" command, the setpoint name of the EPH is selected from a pull-down menu and displayed instead of the IO name.
 - With the "SetEMAVReal" command, writing occurs to a block I/O from the point of view of the setpoint of the lower-level EM.

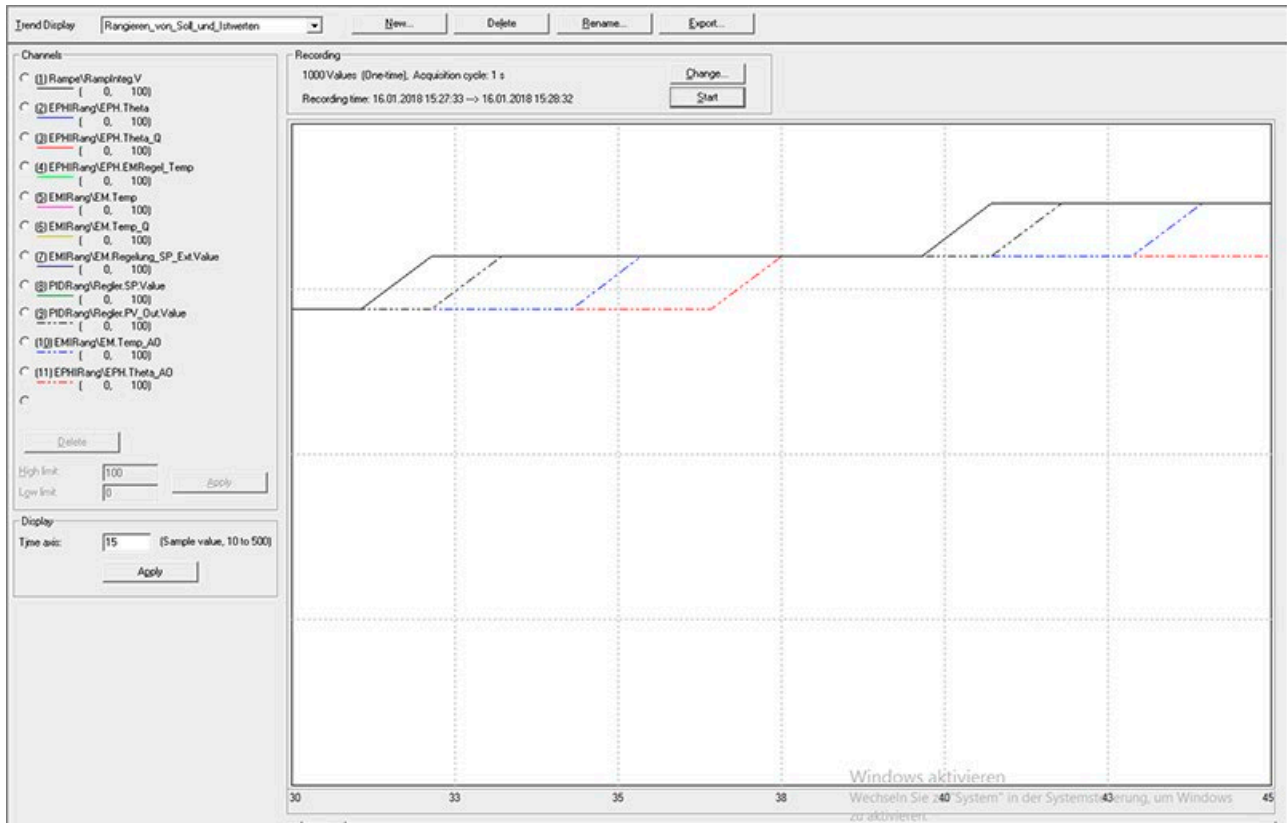
Of course, only one of the statements would be needed.



Result

The results are produced assuming the following run sequence EPH - EM - CM.

Thus, the setpoints are written from the source to the CM via EPH and EM without delay in the same cycle. This situation is displayed graphically by the identical curve shape of the "Channels" 1-8.

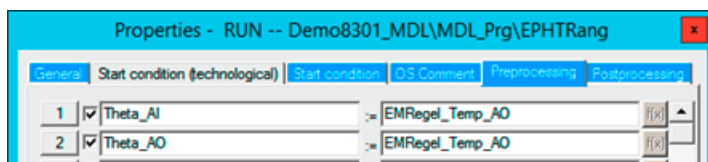


The black dashed line shows the actual value of the process. You can see that this actual value is available as the EM actual value only 2 OB cycles later and as the EPH actual value only 2 additional OB cycles later, thus with a total delay of 4 OB cycles.

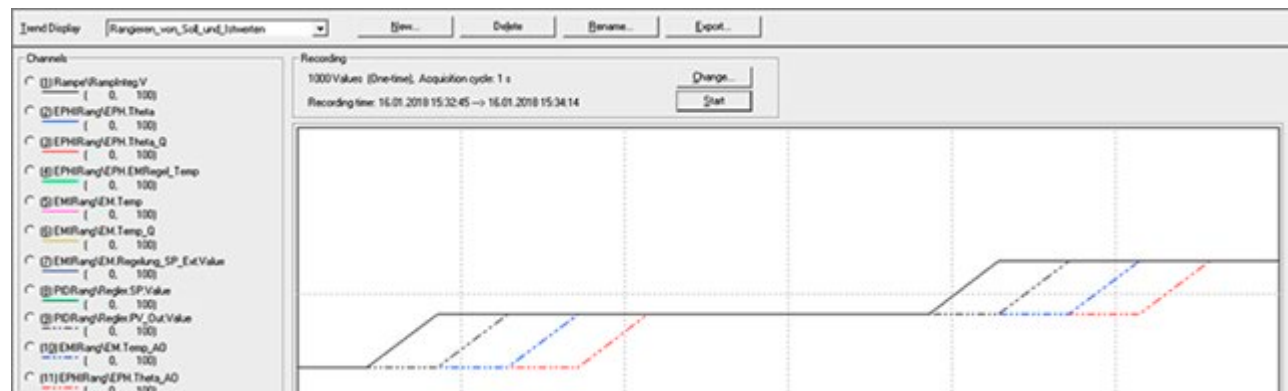
The reason for this is that after the SFC call, the "Setpoint_AI" input is first written to the "Setpoint_AO" output before the sequencer processing is started.

10.5 Separation between EPH and EM

To save the OB cycle for the routing of "Setpoint_AI" to "Setpoint_AO", the actual value of the lower-level object is additionally written to the "Setpoint_AO", as shown in the EPH example:



The actual value is then already available at the phase after 2 OB cycles.

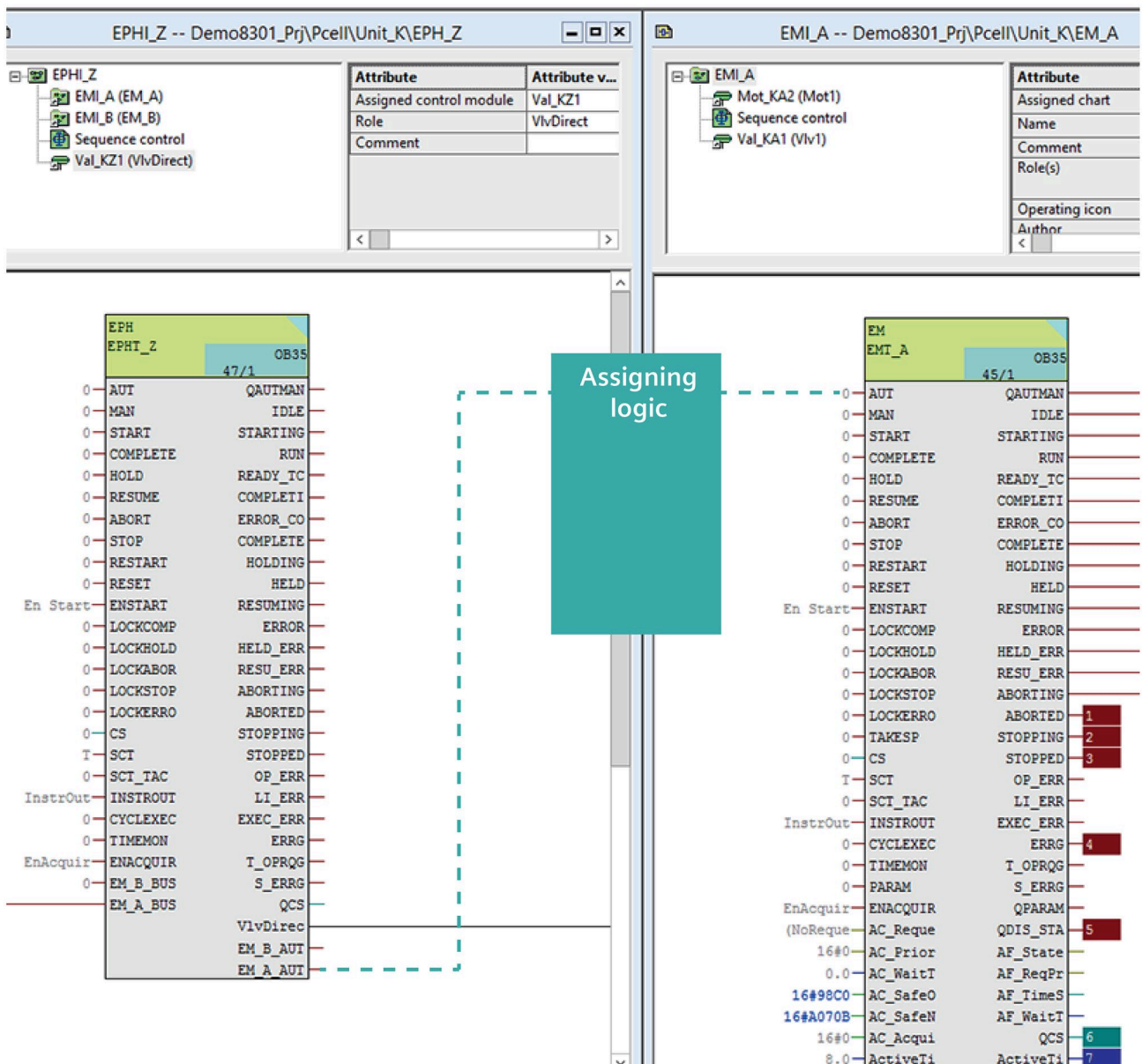


10.5.10 Consequences of the assignment logic not visible in the CFC or MARC

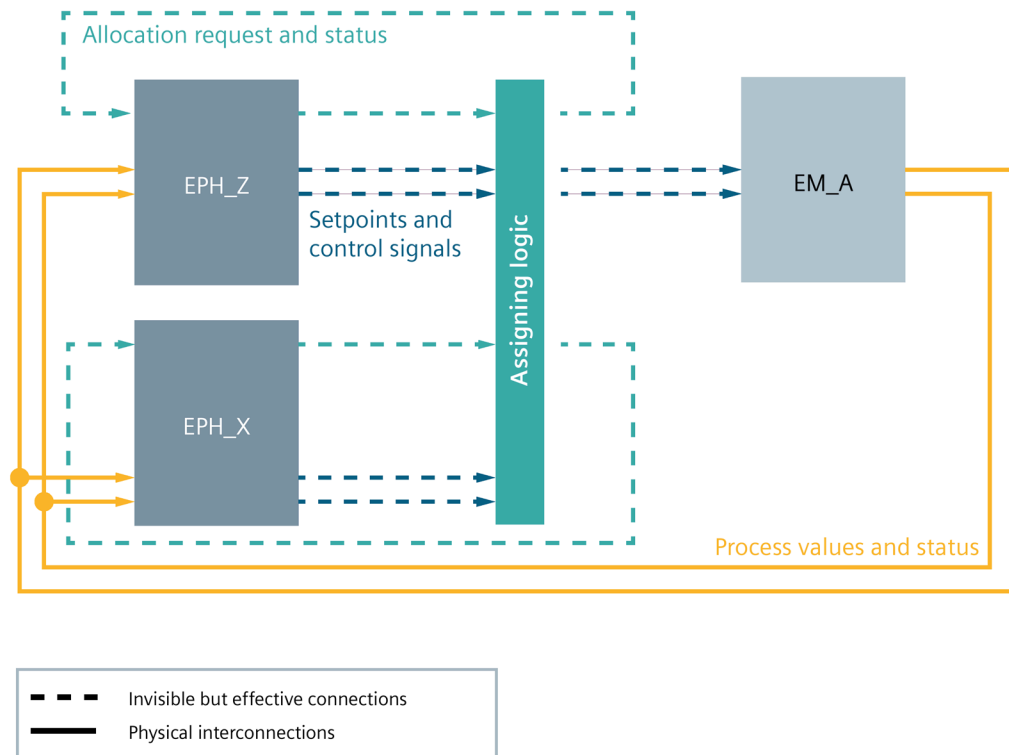
During monitoring and manipulation within the online CFC, it must be kept in mind that there are no visible connections from the higher-level EPH to the lower-level EM. Only feedback messages from the EM to the EPHs are recognizable as interconnections.

The following figure shows that there is no interconnection from the "EM_A_AUT" output of the EPH to the "AUT" input of EM instance "EMI_A", but this is supplied nevertheless.

If you attempt to write the "AUT" input in the online CFC, this change will have no effect because it will be overwritten in the interim.



The assignment logic is presented schematically in the following:



BATCH-related associated values and messages

As soon as a lower-level EM is allocated by a phase, the values of the "QBA_NA", "QBA_ID" and "QSTEP_NO" outputs are automatically interconnected to the EM.

These are normally reserved for operation with SIMATIC BATCH. (If SIMATIC BATCH is not in use, this functionality can be used otherwise.)

In return, however, this means that the values are deleted again when the EM is released. For continuous acquisition and logging, this may not be desirable. In this case, you must take suitable measures. At this point, for example, a phase for allocating all EMs can be called at the start of a "Recipe Unit Procedure" - RUP. This is done with the EM assignment strategy "Wait (temporary)". Then any other phase can take over the EM, use it and after the use release it again (release and expiry of the wait time) the EM is taken over again by the original EPH.

Background

SIMATIC BATCH recognizes the batch reference from messages based on associated values to which the batch ID "QBA_ID", among other things, is written. With allocation of a unit, all EPHs belonging to it are supplied with a batch name and batch ID, but not the EMs, because from the BATCH point of view they are not recognized or included. The EMs and thus also the CMs connected to them must be supplied by allocation.

10.5.11 Name length

The names of the roles are also used for generating the I/O names. An I/O name is made up of the role name and the name of the I/O of the lower-level object, separated by "_". The length of a name for an I/O to the SFC is limited to 24 characters.

For the name of the role of an EMT on an EPH, a maximum of 7 characters is allowed, because the I/O with the longest name and "s7_contact" attribute on the EMT is "ERROR_COMPLETING". This already has 16 letters. This leaves 7 letters ($24 - 16 - 1 = 7$) for the role name.

For the same reason, the I/O name of a setpoint of the lower-level EM must not be longer than $24 - 7 \{\text{MaxRoleName}\} - 4 \{"_ERR"\} - 1 = 12$ characters.

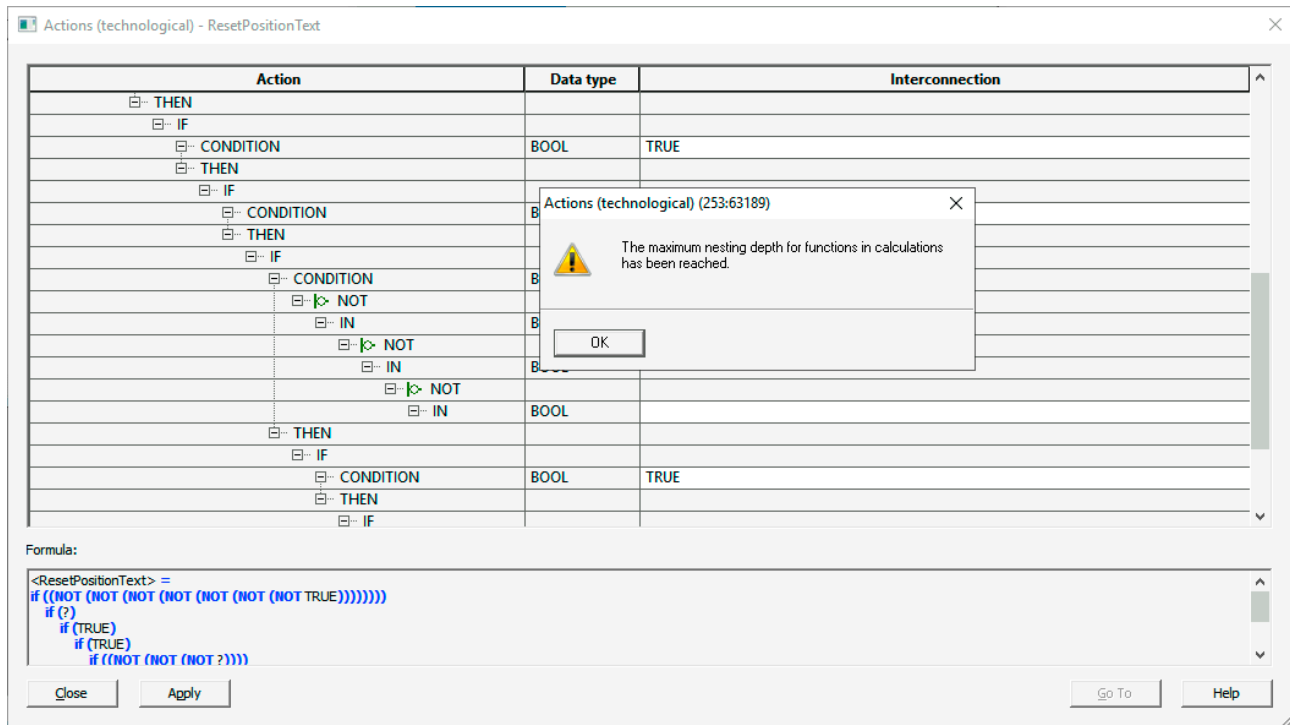
For the roles of the CMT on the EMT, this cannot be said generally because it is unknown which I/O names are used on the technological block of the CMT. In general (e.g. APL library), these are much shorter so this restriction comes into play less often here.

10.5.12 Additional information about the number of commands and nesting depth in technological actions and technological conditions

The use of functions (technological conditions or IF) and language elements (STATEMENT_LIST, IF_THEN) results in nesting, which is also displayed graphically in the editor.

The maximum nesting depth is 8 and is independent of how it is created. Nesting is also displayed graphically in the editor and a message is output when the maximum depth is exceeded.

The following example shows a mixture of nested IF with multilayer condition:



Regarding the number of commands as they arise within a technological action by STATEMENT_LIST, there is no fixed limit and is also rather irrelevant from the current point of view.

Assignments in SFC steps are realized in an "Action FC" for numerous steps. If the 64-KB limit specified by the system is exceeded, another FC is appended and managed internally.

A restriction could theoretically arise if a technological action required so much code that it could not be implemented in an FC.

10.5.13 Supplementing information to commands and status of the CmdStatLib

As previously mentioned, the CmdStatLib in the SFC library serves as a template. The commands can thus be adopted but also modified and adapted to the respective needs.

Some commands are fully implemented, but would not work immediately if used, because the block I/Os to be used for them lack the "s7_contact" attribute. This decision was made consciously from the consideration over the interface size and thus also the memory requirement for the assignment logic (see MARC).

It becomes more understandable with the example, SetEMSPPO:

Action	Data type	Interconnection
Role_SetEMSPPO		
Role.SetEMSPPO		
SetpointPO	SETPOINT	Role.SpPO_1
Value	REAL	12.34
ValueB	DINT	6789
ValueM	STRING	'PurifiedWater'

Formula:

```
<Role_SetEMSPPO> = Role.SetEMSPPO(Role.SpPO_1, 12.34, 6789, 'PurifiedWater');
```

Close Apply Go To Help

The following warning comes when compiling in the master data library:

*W: MarcEPH1Real\RUN END <Role_SetEMSPPO> Role.SetEMSPPO.I.002 TREF:
Role_SetpointPO_B: the parameter has an open textual interconnection*

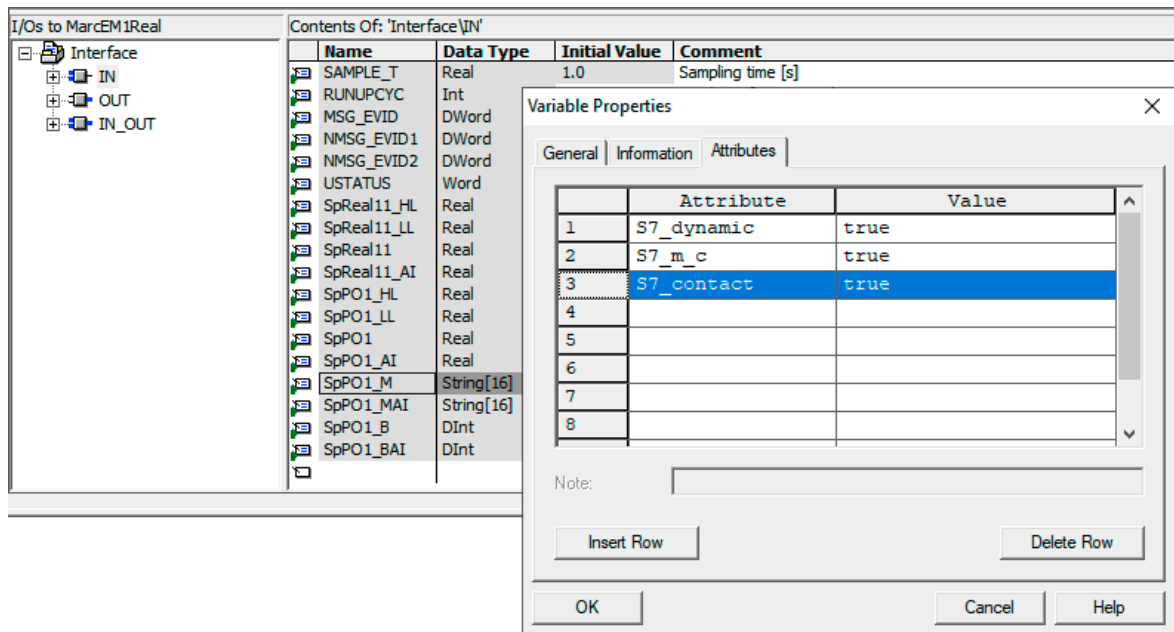
If one would try to compile existing instances in the project, additional errors of the respective instances would be added and compiling is not possible:

E: No block contact I/Os available: 'Role_SpPO1_M' (SFC type 'MarcEPH1Real'). Please update block contacts.

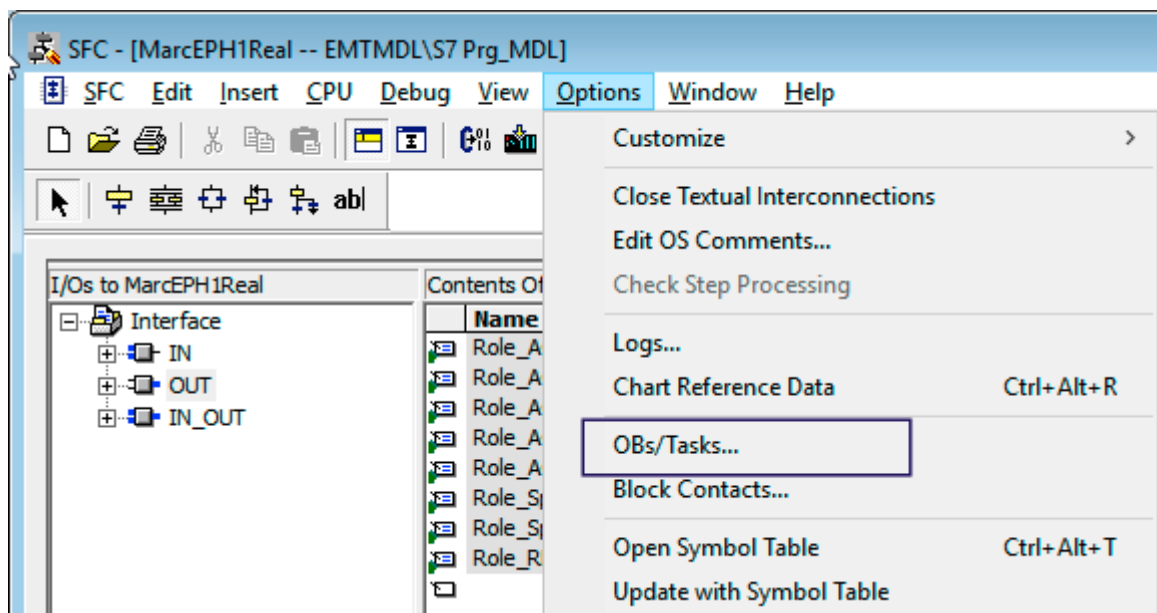
Cause and remedy:

For the command to work properly, the two contacts (outputs) Role_SetpointPO_B and Role_SetpointPO_M must actually exist on the EPH.

For the automatic creation of these contacts, the inputs SetpointPO_B and SetpointPO_M of the EM(T) to be controlled must be given the "s7_contact" attribute. This is done in the block contact dialog of the SFC type within the master data library.



For this correction to also take effect in the higher-level EPH and the mirrored interface to be updated, the "Block Contacts" option must be used at the EPH.



Afterwards, the textual interconnections can be successfully completed.

10.6 Central configuration possibility using technological list editors, TLE

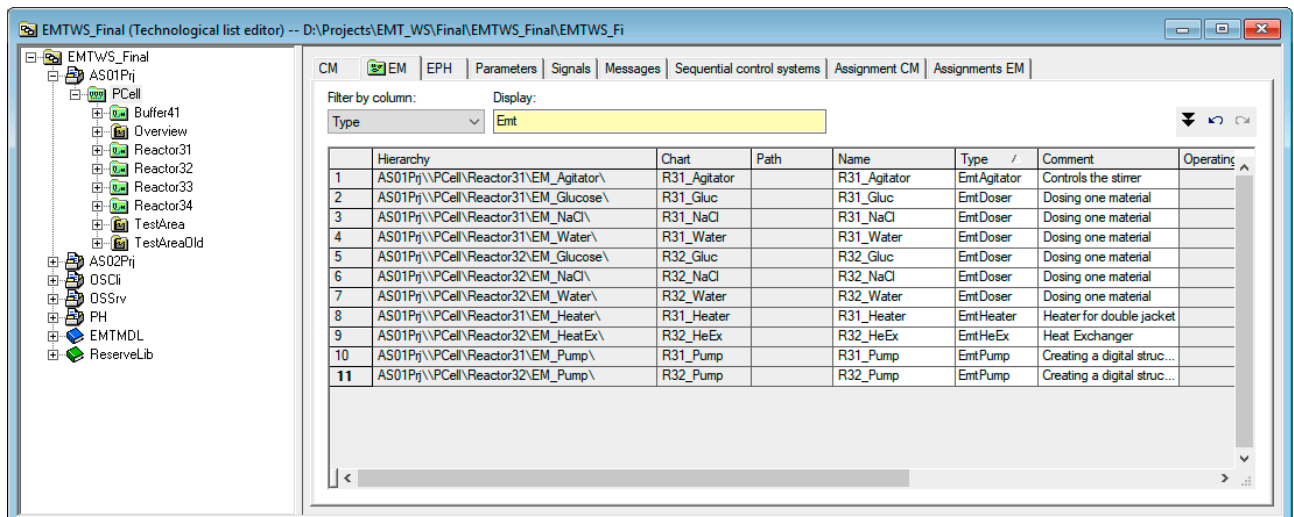
With the technological engineering, a description language was created, which makes it possible to exchange data from higher-level engineering systems as for example COMOS or PAA over an export and import, to create objects, to configure and to interconnect. But even without these tools, further possibilities and simplification of configuration have been created.

For this, in addition to the component view, plant view and process object view, the SIMATIC Manager has been extended with a new view called the technological list editor, or TLE for short.

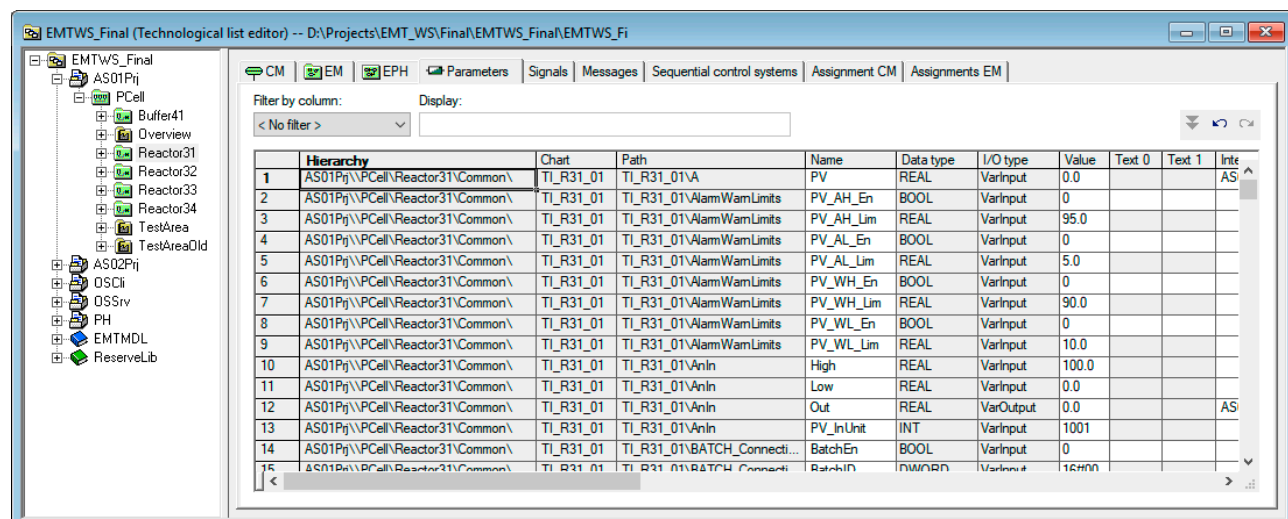
The respective display is filtered according to the branch of the PH or CM, EM or EPH selected in the tree view on the left.

There is a separate register each for the objects CM, EM and EPH, which shows the objects of this category and their nodes (partly also called Sub-CM). Names and cycle times can be adapted from a central location here.

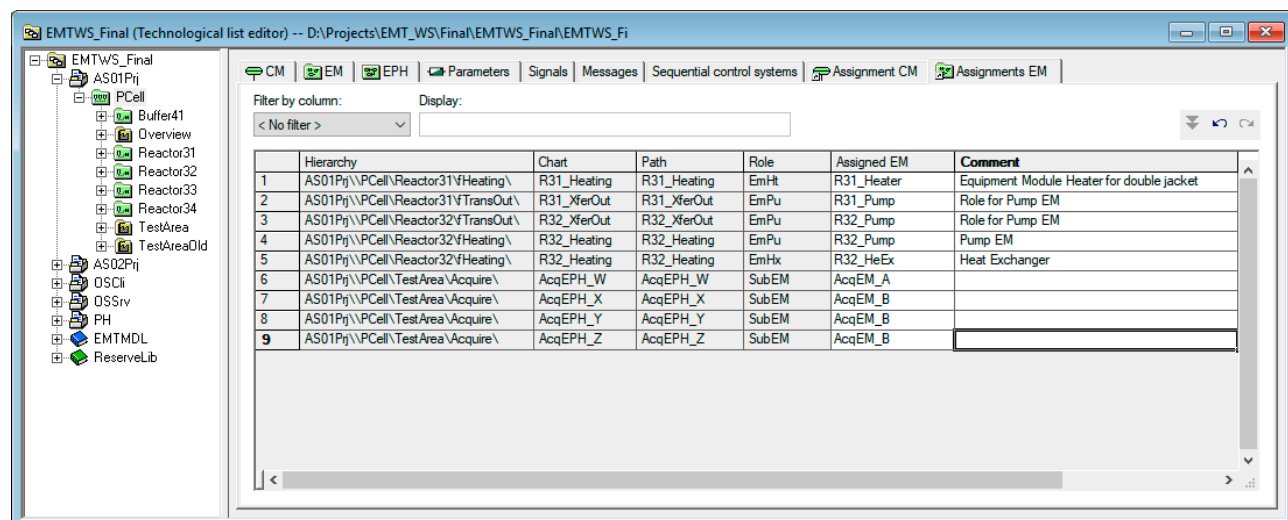
The figure below shows all EM instances with a type name starting with "Emt". There are therefore 6 instances of the "EmtDoser" type here.



In the Parameters and Signals tabs, all parameters and signals of all objects (CM, EM or EPH) are displayed and can be edited there. Filtering in the hierarchy tree and possibly the columns is especially recommended for these lists, because update times can be reduced considerably.



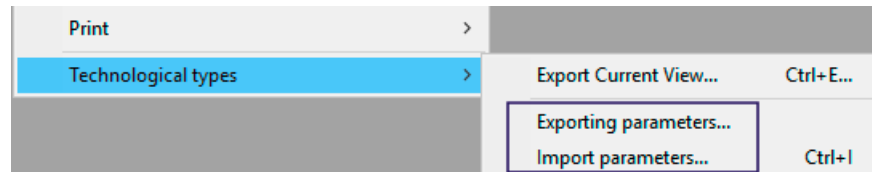
For assigning roles, i.e. which CM role is to be fulfilled by which CM instance or which EM role is to be fulfilled by which EM instance, there are two additional tabs, so that this configuration can also be made centrally without having to open the respective CFC.



10.7 Mass or bulk engineering of EPH and EM

The texts and values can be adapted not only in the TLE. In addition, the tables can also be exported and imported as a whole or filtered.

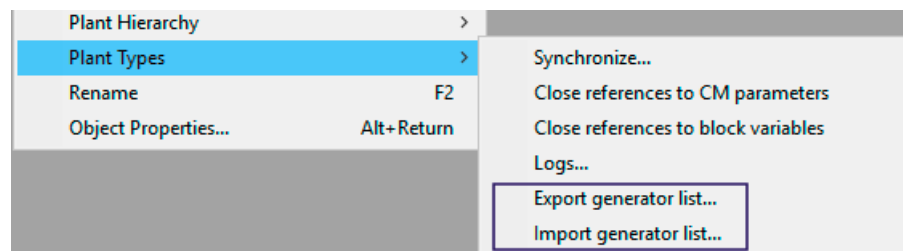
The "Technological types" subcommand in the shortcut menu is used for this purpose.



The export takes place in a CSV file. This file can then be edited externally, either manually or with the help of a tool, and imported again.

It should be noted that no new lines are created in the process, because all these are parameterizations in objects already existing.

For creation of new CM, EM or EPH and the addition or deactivation of variants, the "Plant Generator" is used, which is called from the shortcut menu of the hierarchy tree.



Export and import can be performed as CSV as well as XML.

10.8 Memory requirements

The simplified configuration options, the ease of maintenance, the greater transparency is achieved by functionality in the code. The EXCEL calculation tool mentioned in the readme cannot reflect this.

For a better estimation, empirically determined values for the new functionalities like commands, language elements like IF-THEN and also the assignment logic MARC are specified below.

10.8.1 Memory requirements for the MARC FC assignment logic

As described in detail above, an MARC assignment logic is created for each EM instance that is controlled by a higher-level EPH (or also EM).

The required memory is mainly determined by the number and type of setpoints and parameters and, additionally, by the number of EPHs potentially using the EM.

Three different scenarios were selected for this purpose:

1. Setpoint of the Real type
2. Real setpoints and 2 PI/PO setpoints. *[Currently as 10*Real]
3. 10 Real setpoints and 2 String[32] setpoints,

In each case, the memory requirement is given for a 1:1 relationship, i.e., 1 EPH controls the EM and in the second case 6:1, 6 EPHs share the EM.

Empirically determined values based on CFC 9.0.5.3:

Size MARC FC	1 * Real	8 * Real + 2 * PO *	10 * Real + 2 * Str[32]
1 EPH : 1 EM	6,174 bytes	6,238 bytes	6,382 bytes
6 EPH : 1 EM	8,898 bytes	9,182 bytes	10,146 bytes

The following formula can be used for rough estimation:

Memory requirement MARC for an EM instance

$$\text{Mem} := 5.634 + \#EPH * (528 + \#Real * 8 + \#Str[32] * 72)$$

, where #Real is the number of Real setpoints, #Str is the number of String[32] setpoints, and #EPH is the number of EPHs potentially controlling this EM.

10.8.2 Memory requirements by IF-THEN and IF-THEN_ELSE

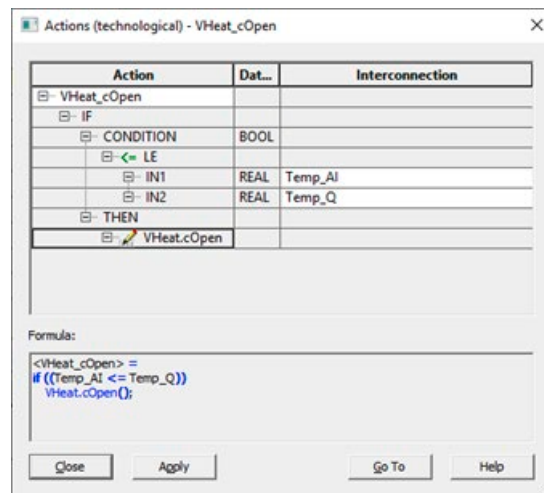
Each instruction has its own Initialization, Processing and Termination processing phases (in short: I, P, and T), which, in the case of conditional instructions, must be handled independently of the phases of the step.

This complex processing and especially, the possibility of online visualization in the OS (WinCC) is also reflected in the memory requirements.

A simple calculation as with the MARC is not possible here and therefore some examples should help with the estimation.

Simple IF:

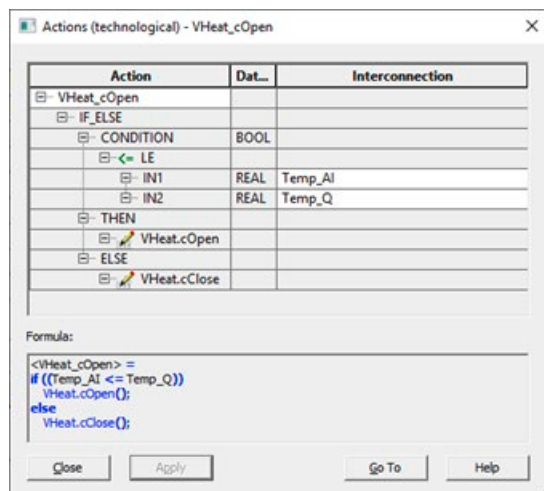
The valve is to be opened when the actual temperature falls below the setpoint temperature.



Additional memory requirement in the Action FC: 776 bytes.

IF-ELSE:

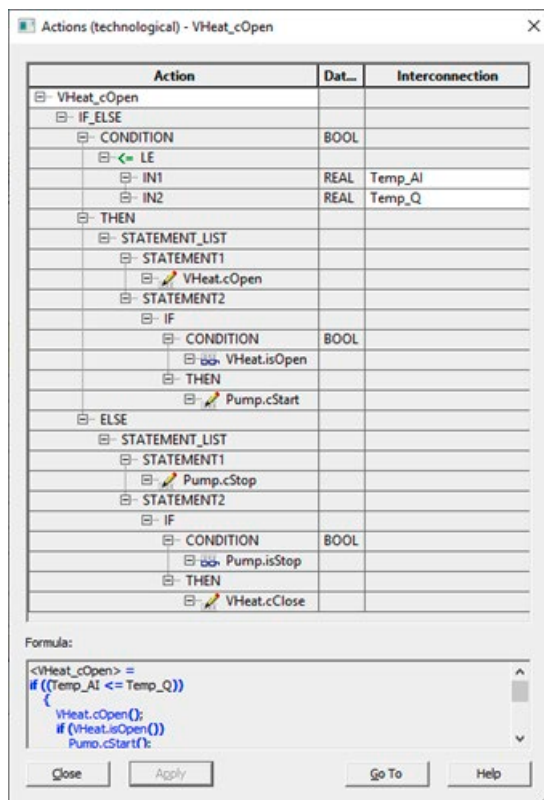
For a two-point characteristic, the valve is to be opened when it falls below the limit and otherwise closed.



Additional memory requirement in the Action FC: 1,526 bytes,

Nested IF:

Similar to IF THEN, however the pump should not start until the valve is open, but the pump should stop first before closing the valve.

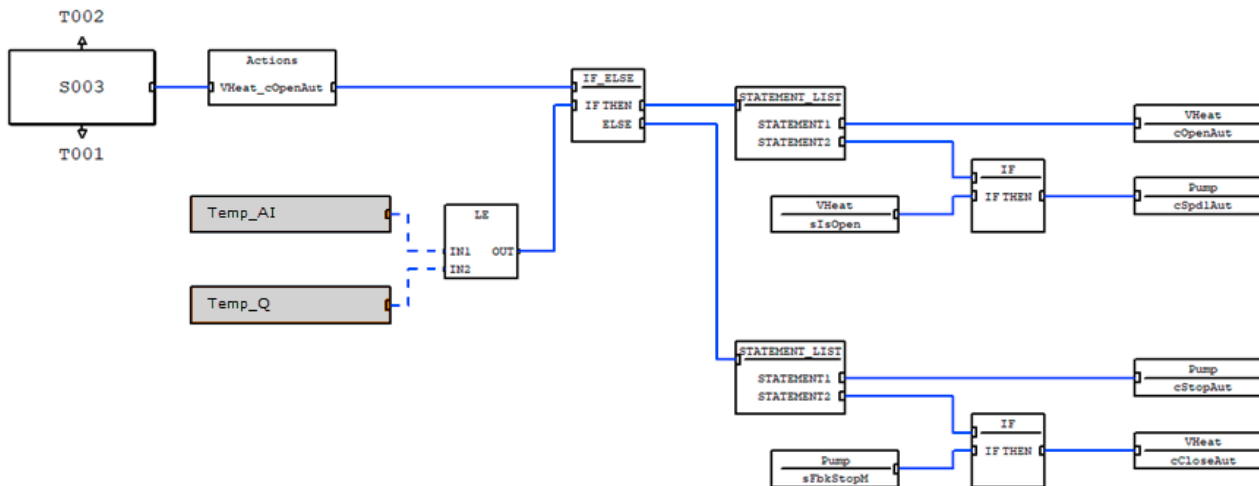


Additional memory requirement in the Action FC: 3,206 bytes,
Values were determined based on CFC 9.0.5.3.

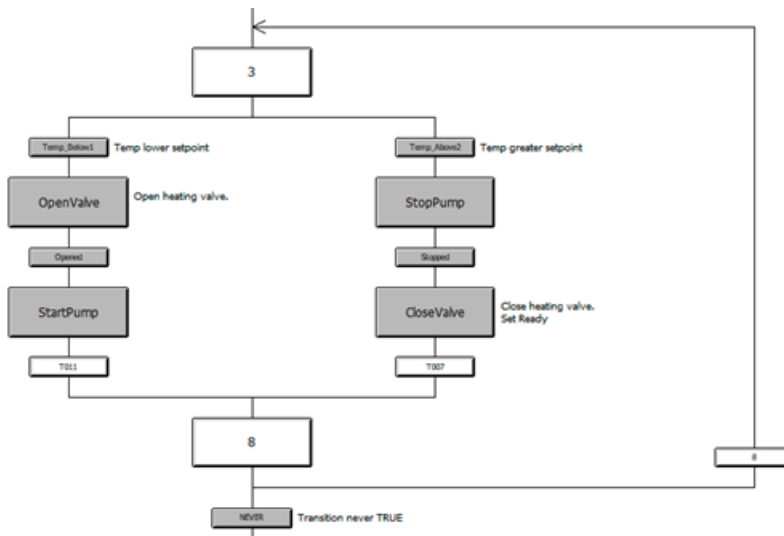
Valve commands: Ini: 4 assignments ; proc: 2 assignments
Valve status: 1 comparison
Motor commands: Ini: 3 assignments ; proc: 2 assignments
Motor status: 1 comparison

Representation in the PIQ:

In a PIQ, the current representation would look like this: This shows the consistency of the technological configuration to be able to configure SFC belonging to an EM/EPH as well as in the higher-level engineering system.



For comparison, the Nested IF scenario was realized conventionally (similar but not identical behavior).

**Fully conventional:**

Additional memory requirement in the Action FC: 590 bytes,

Partially conventional:

This means technological commands and status but foregoing of IF structures:

Additional memory requirement in the Action FC: 796 bytes.

10.9 EM and EPH in the SIMATIC BATCH context

All the previous statements also apply to use in conjunction with SIMATIC BATCH. In addition, attention must be paid to the classification of the SFC types, the classification of the hierarchy folders, the object EPH interface and the resulting effects in SIMATIC BATCH. From the perspective of SIMATIC BATCH, only the level of the phases is to be taken into consideration.

10.9.1 Settings and configuration in SIMATIC Manager

The classification of the hierarchy folder in the S88 Type tab has the following values:



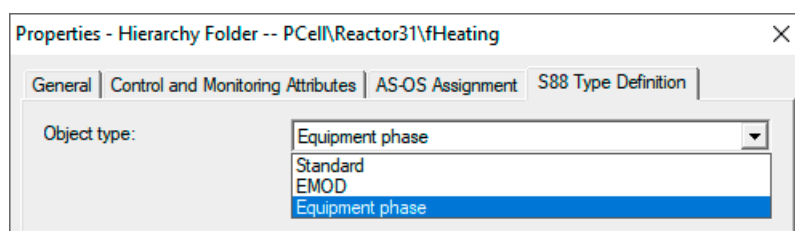
<neutral> ("Standard"), i.e. no BATCH relevance



Equipment module ("EMOD")



"Equipment Phase" - new with PCS 7 V9.0



The following values exist for the classification of the SFC types:



"None" for SFC without BATCH context and EM



"EPH"



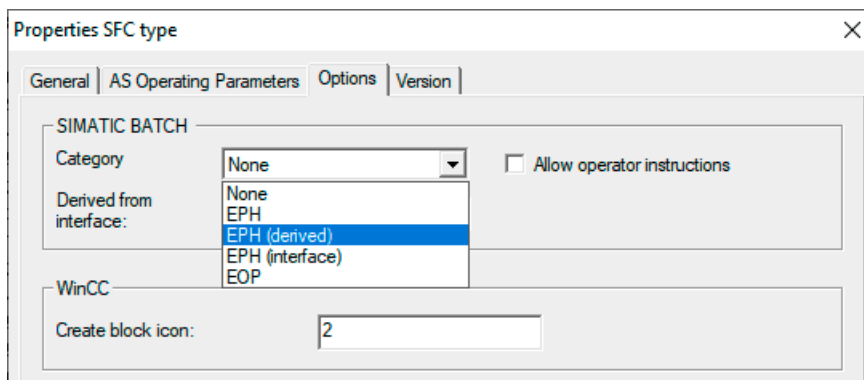
"EPH (derived)" - new with PCS 7 V9.0.



"EPH (interface)" - new with PCS 7 V9.0



"EOP"



There is a dependency between the S88 type of the hierarchy folder and the values of the SFC types used. This means only the following combinations can be configured: Other combinations are interpreted by the BATCH dialog when generating the data as an error with a corresponding message and consequently not compiled.

S88 type Hierarchy folder	BATCH category: SFC type	Effectiveness
<neutral>	None	Use as equipment module (EM) without direct BATCH reference or usual SFC instance
Equipment module	EPH	Interpretation as "Recipe aware" EM, whereby the name of the recipe phase is determined from the name of the SFC type.
Equipment phase	EPH (derived)	Interpretation as EPH with an interface for BATCH whereby the name of the recipe phase is determined from the name of the EPH interface.

The "EPH" value of the SFC type is allowed in the folder of type "EMOD".

Instances of SFC types of the "EPH (derived)" value may be placed in a folder of the "Equipment module" type. An "EPH (interface)" also has to be available for these types.

The respective values are also visualized in SIMATIC Manager with different icons:



CFC



CM instance (created from CMT)



EM instance (created from EMT) - new with PCS 7 V9.0



EM instance (created from EPHT) - new with PCS 7 V9.0

10.9.2 "Equipment module with EPH" folder

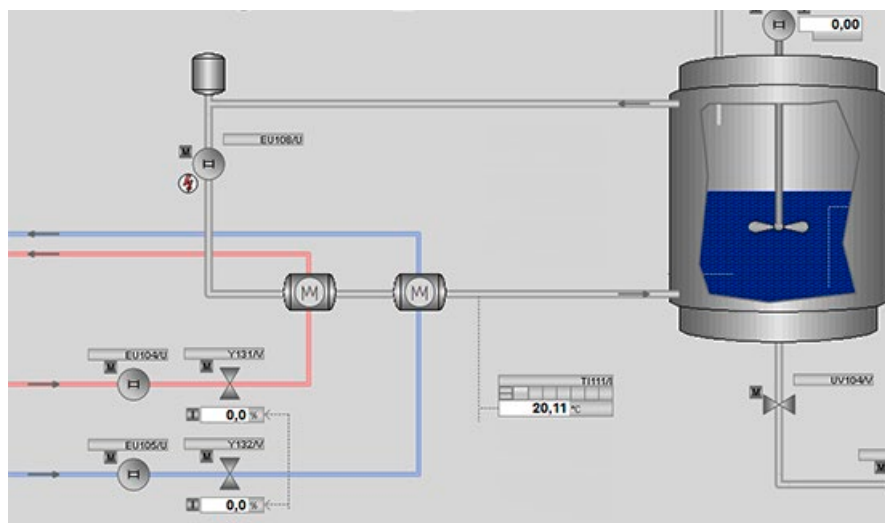
Starting with SIMATIC PCS 7 V9.0, the phases can also be created based on EPHTs using technological configuration, thereby utilizing their benefits.

10.9.3 "Equipment phase with EPH (derived)" folder

Starting with PCS 7 V9.0, an additional configuring option has been implemented. On the one hand, this makes it possible to combine different realizations of a technical function under the same recipe phase, and, on the other hand, it allows an abstraction of the names in the recipe from the names in the basic automation.

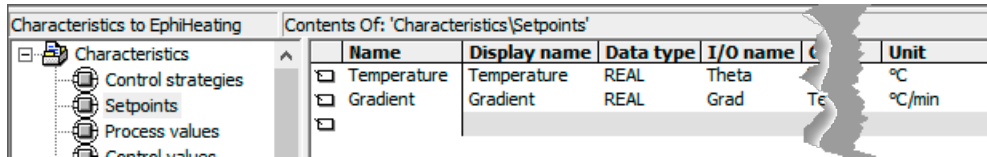
A temperature control, for example, can be implemented differently on an instance-specific basis. The various SFC types can be called in SIMATIC BATCH under the same name.

- This takes place in the small 2 liter reactor through an electrically actuated heating sleeve.
- In the large 3000 liter reactor, this takes place through a double wall of the reactor. A liquid heating medium is circulated through it, and a heat exchanger is connected for heating and cooling. These have valves for process steam and cooling water, which must be controlled as shown in the following figure:



From the perspective of the technologist, only the following criteria are important for the recipe-control.

- Tempering, here "EphiHeating" (= name of the phase)
- Target temperature, here "Temperature" in °C
- Temperature gradient, here "Gradient" in °C/min

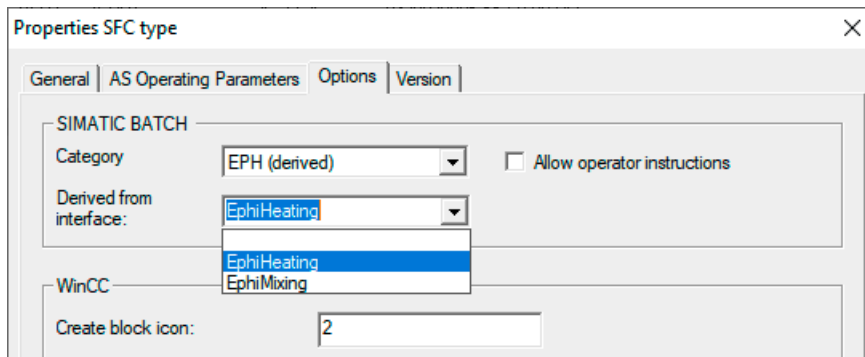


Name	Display name	Data type	I/O name	Unit
Temperature	Temperature	REAL	Theta	°C
Gradient	Gradient	REAL	Grad	°C/min

The name of the later phase (in this example "EphiHeating") and the parameters relevant for the recipe ("Temperature" and "Gradient") are defined through the EPH interface.

For this, a formal SFC type must be created in the Component view of SIMATIC Manager and categorized as "EPH (interface)" in the object property. Formally, this is a SFC type, so the same editor can be used. However, this does not have any sequencers, is not loaded into an AS and therefore does not require any memory or computing time.

This functionality is implemented in one or more EPH types. Once an EPH interface has been created beforehand, the actual EPH types are categorized as "EPH (derived)", and the appropriate EPH (interface) is selected in the "Derived from interface" field that becomes visible.



Properties SFC type

General | AS Operating Parameters | Options | Version

SIMATIC BATCH

Category: EPH (derived) ☐ Allow operator instructions

Derived from interface: EphiHeating

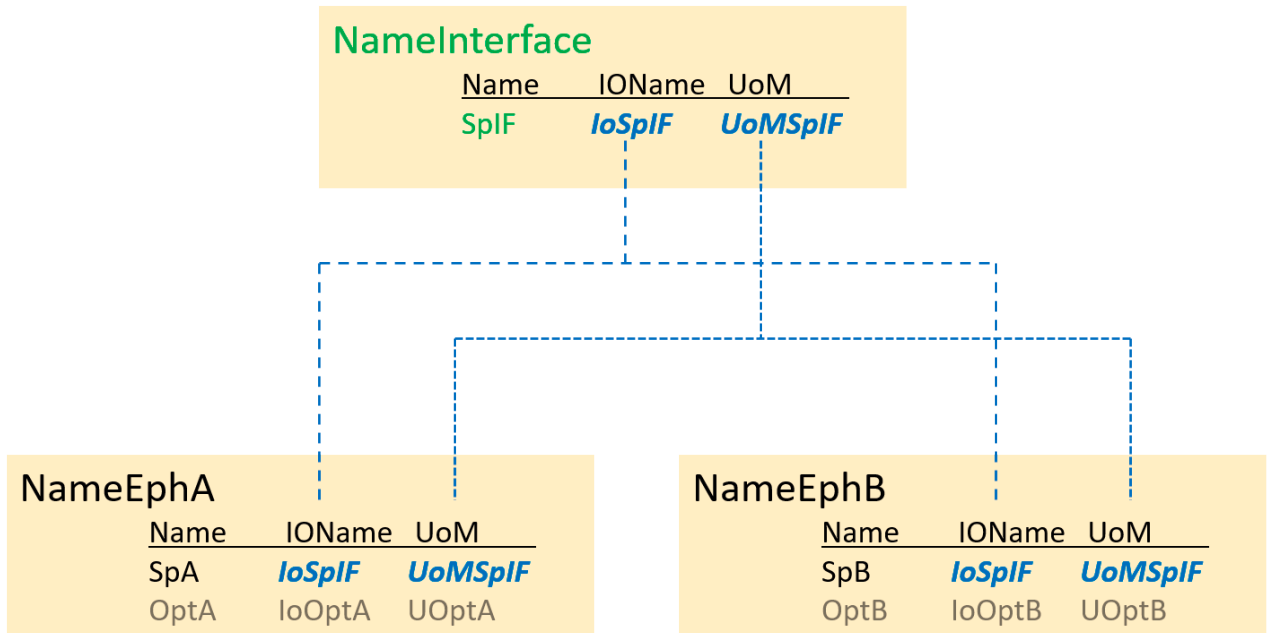
WinCC

Create block icon: 2

This relationship is also visible in the detail view in SIMATIC Manager:

EphiHeating	SFC type (EPH)
EphiMixing	SFC type (EPH)
EPHT401SP	SFC type (EPH)
EPHT410SP	SFC type (EPH)
EphHeatHt	SFC type (EphiHeating)
EphHeatHx	SFC type (EphiHeating)

The connection between EPH interface, i.e. the BATCH phase, and the SFC types used is made via the I/O name and the unit of engineering ("UoM").



The name of the interface and the setpoints may differ. Likewise, the setpoint names of the assigned SFC types may differ. In there are discrepancies, corresponding error messages are generated by the BATCH dialog.

NOTICE

In the BATCH dialog, neither the control strategies nor the assignment of the setpoint to the control strategy are checked. The settings on the interface are decisive for SIMATIC BATCH. For the respective faceplate in the PCS 7 OS, it is the assignment at the SFC type. The project engineer must ensure that the corresponding functionality (behavior for a specific CS value) is provided correctly.

The high and low limits are instance-specific and have no significance for the EPH interface.

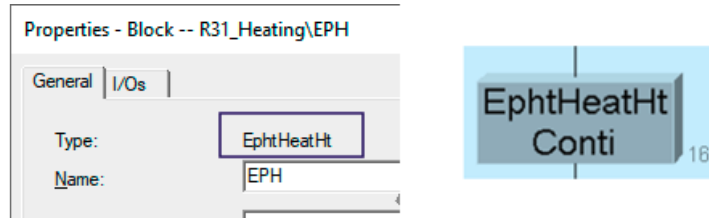
10.9.4 Naming of the RPH (SIMATIC BATCH)

The name of a recipe phases in the recipe editor / BatchCC depends on the type of configuration. There are a total of eight cases, of which only four are supported or permissible.

The differences lie in the classification of the hierarchy folder, whether or not the project is in multi-instance mode:

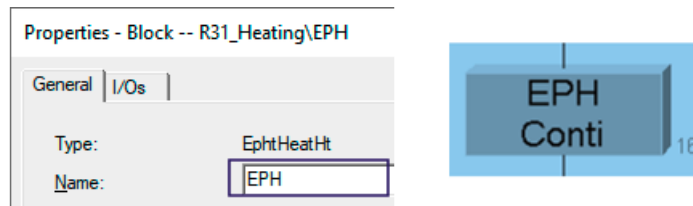
1. EMOD without multi-instance:

The name of the recipe phase results from the name of the SFC type.



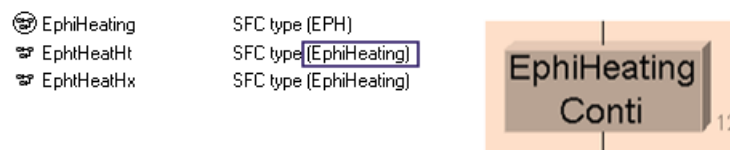
2. EMOD with multi-instance:

The name of the recipe phase results from the block name of the instance of the SFC type.



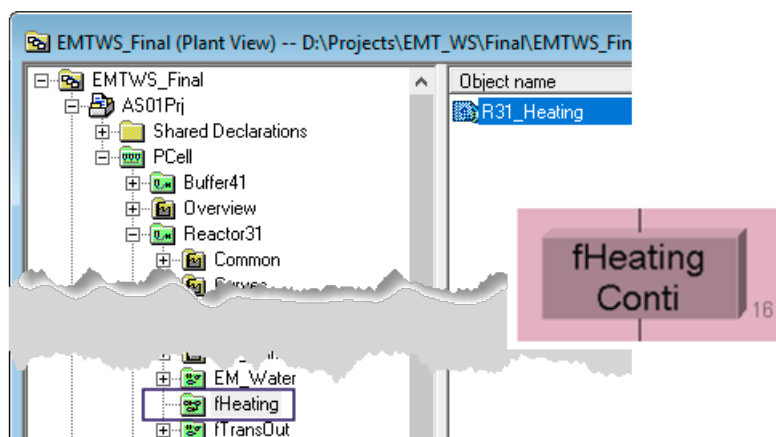
3. Equipment phase without multi-instance:

The name of the recipe phase results from the name of the EPH interface.



4. Equipment phase with multi-instance:

The name of the recipe phase results from the name of the hierarchy folder containing the EPH.



This is summarized again briefly in the following figure:

	Projekt nicht im Multiinstanzmodus	Projekt im Multiinstanzmodus
Ordertyp in TH „EPH“	Name des SFC-type 	Name der SFC-instance 
Ordertyp in TH „EPH (derived)“	Name des Interface 	Name des Ordners in der TH 

Service and support

Industry Online Support

Do you have questions or need assistance?

Using the Industry Online Support, you have round-the-clock access to expertise spanning the entire range of service and support, as well as to our services.

Industry Online Support is the central address for information about our products, solutions and services.

Product information, manuals, downloads, FAQs and application examples – all information can be accessed with just a few mouse clicks: <https://support.industry.siemens.com/>.

Industry Online Support App

The "Siemens Industry Online Support" app provides you with optimal support even when you are on the go. The app is available for Apple iOS, Android and Windows Phone:
<https://support.industry.siemens.com/cs/ww/en/sc/2067>.

Technical Forum

Exchange your experience and know-how about our products or systems or benefit from the knowledge of others.

Have discussions on special products or general topics, discover new ideas and inspiration and help yourself and others on the Technical Forum
(<http://www.siemens.com/automation/forum>) – free of charge, outside office hours and at the weekend.

Technical Support

The Siemens Industry Technical Support offers you fast and competent support for any technical queries you may have with a number of tailor-made solutions – ranging from basic support to individual support contracts.

Send your queries to Technical Support using the following web form:
www.siemens.com/industry/supportrequest.

Range of services

Our range of services includes the following:

- Product training courses
- Plant data services
- Spare parts services
- Repair services
- On-site and maintenance services
- Retrofitting and modernization services
- Service programs and contracts

You can find detailed information on our range of services in the service catalog:

<https://support.industry.siemens.com/cs/sc>.

Contact partner

If you have any questions or need support, please contact your local representative, who will put you in contact with the responsible service center. You can find your contact partner in the contact database: www.siemens.com/yourcontact.