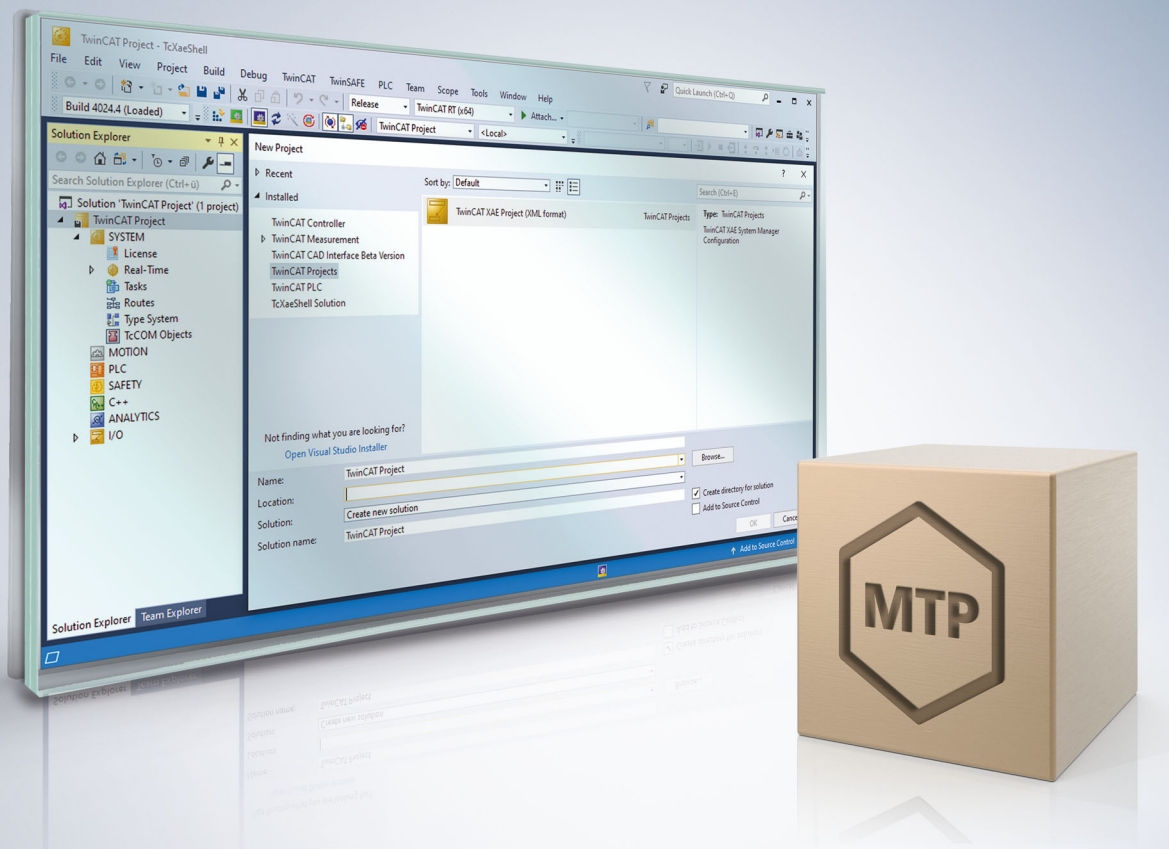
BECKHOFF New Automation Technology

Manual | EN

TF8400

TwinCAT 3 | MTP Runtime



2025-02-12 | Version: 1.1.1

Table of contents

1 Foreword 5
1.1 Notes on the documentation 5
1.2 For your safety 6
1.3 Notes on information security 7
2 Overview 8
3 Installation 9
3.1 Licensing 9
4 Technical introduction 12
5 PLC API 13
5.1 Operation modes 13
5.1.1 Operation Mode 13
5.1.2 Source Mode 15
5.1.3 Service Mode 17
5.2 Properties 18
5.2.1 Name of the object 18
5.2.2 Object description 18
5.2.3 OSLevel 18
5.2.4 WQC 19
5.2.5 Value scaling 19
5.2.6 Unit 19
5.2.7 Value limitation 20
5.2.8 Feedback 20
5.2.9 Read back 20
5.2.10 Handshake procedure 20
5.2.11 Locking 20
5.2.12 Feedback monitoring 21
5.2.13 Limit value monitoring 22
5.2.14 Reset 23
5.3 Function blocks 24
5.3.1 ActiveElements 24
5.3.2 DiagnosticElements 59
5.3.3 IndicatorElements 69
5.3.4 InputElements 76
5.3.5 OutputElements 80
5.3.6 OperationElements 80
5.3.7 ParameterElements 87
5.3.8 PeaElements 101
5.3.9 ServiceElements 102
5.3.10 Report Values 114
5.4 Data types 117
5.4.1 ST_MTP_InputElementConfig 117
6 Support and Service 119

1 Foreword

1.1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.
The documentation and the following notes and explanations must be complied with when installing and commissioning the components.
The trained specialists must always use the current valid documentation.

The trained specialists must ensure that the application and use of the products described is in line with all safety requirements, including all relevant laws, regulations, guidelines, and standards.

Disclaimer

The documentation has been compiled with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without notice.
Claims to modify products that have already been supplied may not be made on the basis of the data, diagrams, and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS®, and XPlanar® are registered and licensed trademarks of Beckhoff Automation GmbH.
If third parties make use of the designations or trademarks contained in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

Third-party trademarks

Trademarks of third parties may be used in this documentation. You can find the trademark notices here: https://www.beckhoff.com/trademarks.

1.2 For your safety

Safety regulations

Read the following explanations for your safety.
Always observe and follow product-specific safety instructions, which you may find at the appropriate places in this document.

Exclusion of liability

All the components are supplied in particular hardware and software configurations which are appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation, and drive technology who are familiar with the applicable national standards.

Signal words

The signal words used in the documentation are classified below. In order to prevent injury and damage to persons and property, read and follow the safety and warning notices.

Personal injury warnings

⚠ DANGER
Hazard with high risk of death or serious injury.

⚠ WARNING
Hazard with medium risk of death or serious injury.

⚠ CAUTION
There is a low-risk hazard that could result in medium or minor injury.

Warning of damage to property or environment

NOTICE
The environment, equipment, or data may be damaged.

Information on handling the product

This information includes, for example:
recommendations for action, assistance or further information on the product.

1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.

2 Overview

The TF8400 TwinCAT 3 MTP Runtime is an IEC 61131 library for implementing the MTP interface types. The MTP interface types are based on the requirements of the MTP guideline.

The following documentation is based on the VDI/VDE/NAMUR 2658 guideline.

3 Installation

System requirements

The following system requirements must be fulfilled for proper functioning of the TF8400 TwinCAT 3 MTP Runtime.

Technical data	Requirement
Target platform	IPC or CX, (x86, x64, Arm®)
Minimum TwinCAT version	3.1.4026
Required TwinCAT setup level	TC1200 TwinCAT 3 PLC

TwinCAT Package Manager: Installation (TwinCAT 3.1 Build 4026)

Detailed instructions on installing products can be found in the chapter Installing workloads in the TwinCAT 3.1 Build 4026 installation instructions.

Install the following package on the command line to be able to use the product:

- TwinCAT.XAE.PLC.Lib.Tc3_MTP

3.1 Licensing

The TwinCAT 3 function can be activated as a full version or as a 7-day test version. Both license types can be activated via the TwinCAT 3 development environment (XAE).

Licensing the full version of a TwinCAT 3 Function

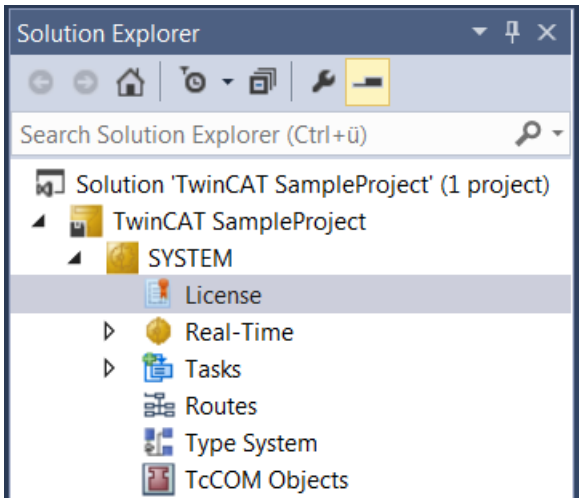
A description of the procedure to license a full version can be found in the Beckhoff Information System in the documentation "TwinCAT 3 Licensing".

Licensing the 7-day test version of a TwinCAT 3 Function

A 7-day test version cannot be enabled for a TwinCAT 3 license dongle.

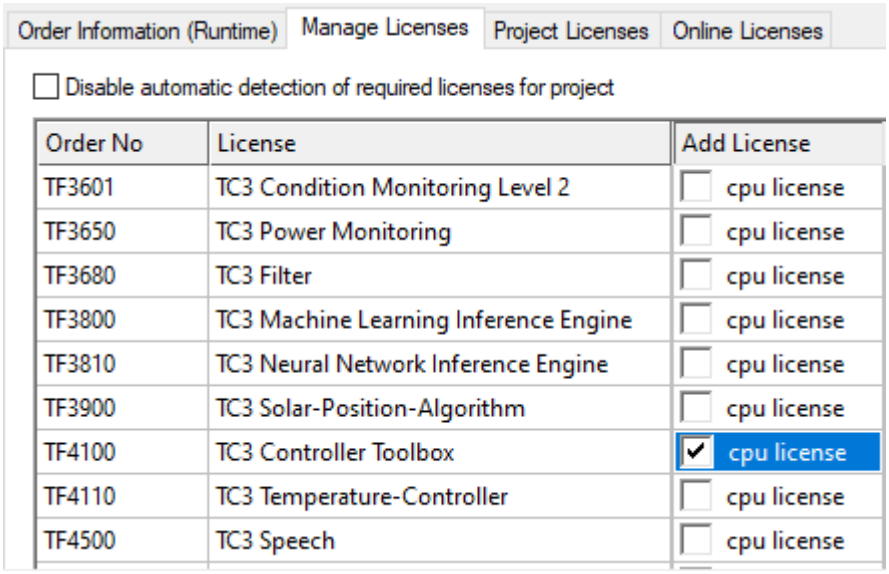
1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.
3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the **Choose Target System** drop-down list in the toolbar.
   ⇨ The licensing settings always refer to the selected target system. When the project is activated on the target system, the corresponding TwinCAT 3 licenses are automatically copied to this system.

4. In the **Solution Explorer**, double-click **License** in the **SYSTEM** subtree.



⇨ The TwinCAT 3 license manager opens.

5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. "TF4100 TC3 Controller Toolbox").

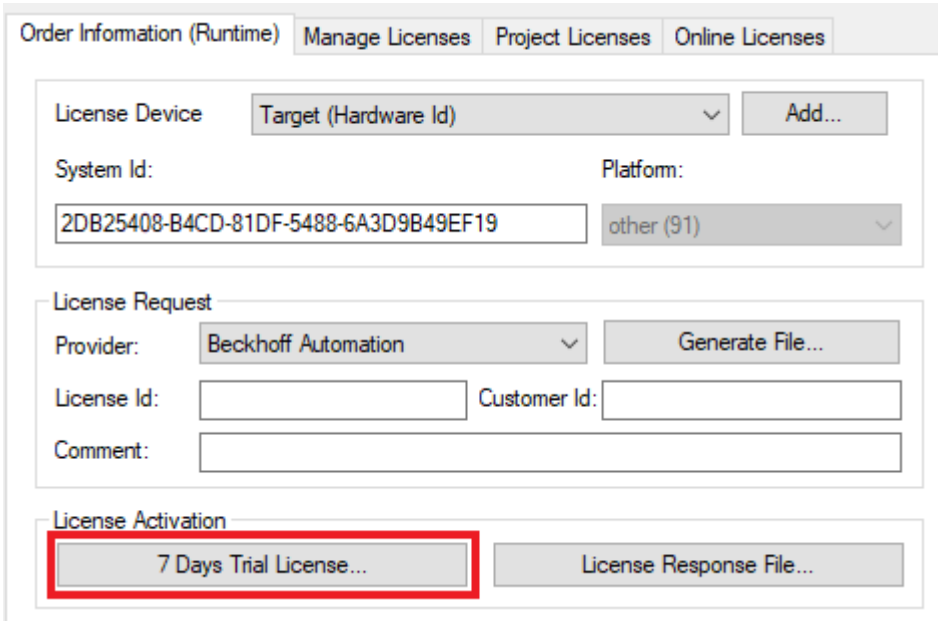


Order No	License	Add License
TF3601	TC3 Condition Monitoring Level 2	cpu license
TF3650	TC3 Power Monitoring	cpu license
TF3680	TC3 Filter	cpu license
TF3800	TC3 Machine Learning Inference Engine	cpu license
TF3810	TC3 Neural Network Inference Engine	cpu license
TF3900	TC3 Solar-Position-Algorithm	cpu license
TF4100	TC3 Controller Toolbox	cpu license
TF4110	TC3 Temperature-Controller	cpu license
TF4500	TC3 Speech	cpu license

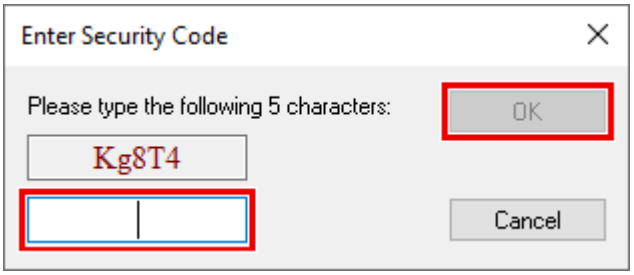
6. Open the **Order Information (Runtime)** tab.

⇨ In the tabular overview of licenses, the previously selected license is displayed with the status “missing”.

7. Click **7-Day Trial License...** to activate the 7-day trial license.



⇨ A dialog box opens, prompting you to enter the security code displayed in the dialog.



8. Enter the code exactly as it is displayed and confirm the entry.
9. Confirm the subsequent dialog, which indicates the successful activation.
   ⇨ In the tabular overview of licenses, the license status now indicates the expiry date of the license.
10. Restart the TwinCAT system.

⇨ The 7-day trial version is enabled.

4 Technical introduction

The PLC library TF8400 TwinCAT 3 MTP Runtime represents objects (function blocks) for controlling and monitoring actuators and sensors in process technology. It is primarily designed for TF8401 TwinCAT 3 MTP Engineering, but can also be used independently.

The objects of the PLC library are divided into the following areas:

- ActiveElements (drives, valves and PID controllers)
- DiagnosticElements (locking indicators)
- IndicatorElements (sensors and displays)
- OperationElements (value specifications)
- InputElements (incoming process value signals)
- OutputElements (outgoing process value signals)
- ServiceElements (services)
- ParameterElements (parameters for services)
- PeaElements (provision of vendor information)

Access to the variables of these objects is controlled via the operation modes [▶ 13] and takes place via:

- Inputs of the function block (for the internal logic)
- External variables (access e.g. via OPC UA)

5 PLC API

5.1 Operation modes

This chapter describes the state machines of the data objects and services.

These are not used alone, but are called within the data objects and services.

5.1.1 Operation Mode

In this library, a distinction is made between `Operator` and `Automatic` as sources for switching requests. The `Operation Mode` is a state machine for managing switching requests from these sources.

Access routes

`Automatic` switching requests are written by the internal PLC logic. The variable names are extended by the suffix `*Aut`.

`Operator` switching requests are made manually (e.g. via OPC UA). They are declared as external variables and cannot be written via the internal PLC logic. `Operator` switching requests add the suffix `*Op` to the variable name.

States

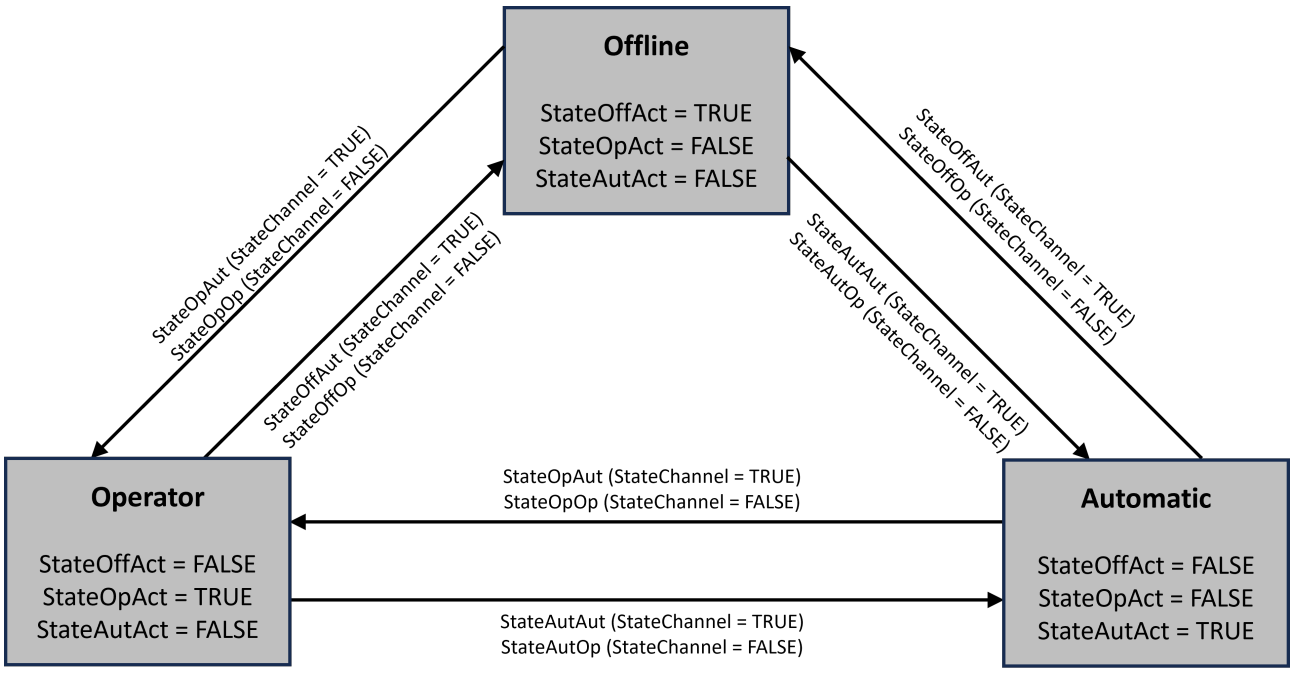
The state machine has three states: `Offline`, `Operator` and `Automatic`. No new switching requests are processed in the `Offline` state. In the `Operator` state, `*Op` switching requests are processed and in the `Automatic` state, `*Aut` switching requests are processed.

The current state is displayed via the outputs `StateOffAct`, `StateOpAct` and `StateAutAct`.

State change

A state change is carried out via switching requests from the internal PLC logic or via an operator (according to the Handshake procedure [▶ 20]). The input `StateChannel` is set via the internal PLC logic and indicates which source can currently trigger a state change.

States are prioritized as follows: `Offline` > `Operator` > `Automatic`. It follows that if all three states are requested at the same time, the state `Offline` is assumed. If `Operator` and `Automatic` are requested at the same time, the state `Operator` is assumed.



A direct transition from the state `Offline` to the state `Automatic` is not provided for in the MTP guideline, but is supported by the state machine.

The `Operation Mode` is part of objects in this library and cannot be used alone.

Inputs

Name	Type	Description	OPC UA access
StateChannel	BOOL	Selection of switching requests for the state machine: 1: `Automatic` switching requests are taken into account. 0: `Operator` switching requests are taken into account.	Read
StateOffAut	BOOL	`Automatic` switching request to transfer the `Operation Mode` to the `Offline` state.	Read
StateOpAut	BOOL	`Automatic` switching request to transfer the `Operation Mode` to the `Operator` state.	Read
StateAutAut	BOOL	`Automatic` switching request to transfer the `Operation Mode` to the `Automatic` state	Read

Outputs

Name	Type	Description	OPC UA access
StateOffAct	BOOL	1: Current state is `Offline`.	Read
StateOpAct	BOOL	1: Current state is `Operator`.	Read
StateAutAct	BOOL	1: Current state is `Automatic`.	Read

External variables

Name	Type	Description	OPC UA access
StateOffOp	BOOL	`Operator` switching request to transfer the `Operation Mode` to the `Offline` state.	Read/write

Name	Type	Description	OPC UA access
		0→1: `Operator` request 1→0: Request has been processed.	
StateOpOp	BOOL	`Operator` switching request to transfer the `Operation Mode` to the `Operator` state. 0→1: `Operator` request 1→0: Request has been processed.	Read/write
StateAutOp	BOOL	`Operator` switching request to transfer the `Operation Mode` to the `Automatic` state. 0→1: `Operator` request 1→0: Request has been processed.	Read/write

Methods

Name	Description
SetOffline	Sets the state machine `OperationMode` of the interface to the state `Offline`.
SetOperator	Sets the state machine `OperationMode` of the interface to the state `Operator`.
SetAutomatic	Sets the state machine `OperationMode` of the interface to the state `Automatic`.

5.1.2 Source Mode

In this library, a distinction is made between `Internal` and `Manual` as sources for value specifications. The `Source Mode` is a state machine for managing value specifications from these sources.

Access routes

`Internal` value specifications are written by the internal PLC logic. The variable names are extended by the suffix `*Int`.

`Manual` value specifications are written by a manual operation (e.g. via OPC UA). They are declared as external variables and cannot be written via the internal PLC logic. `Manual` value specifications add the suffix `*Man` to the variable name.

States

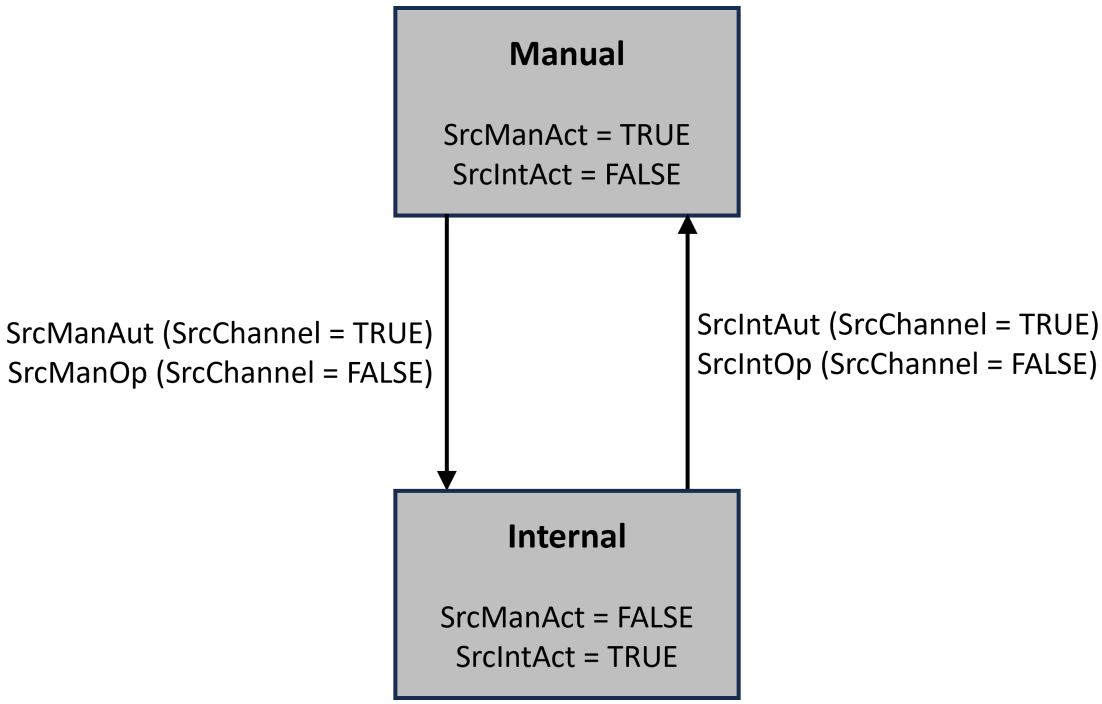
The state machine has two states: `Internal` and `Manual`. In the `Internal` state, `*Int` value specifications are processed and in the `Manual` state, `*Man` value specifications are processed.

The current state is displayed via the outputs `SrcManAct` and `SrcIntAct`.

State change

A state change is carried out via switching requests from the internal PLC logic `*Aut` or via an operator `*Op` (according to the Handshake procedure [▶ 20]). The input `SrcChannel` is set via the internal PLC logic and indicates which source can currently trigger a state change.

`Internal` is prioritized higher than `Manual`. Therefore, if both states are requested at the same time, the state `Internal` is assumed.



The `Source Mode` is part of objects and cannot be used alone.

Inputs

Name	Type	Description	OPC UA access
SrcChannel	BOOL	Selection of switching requests for the state machine: 1: `Automatic` switching requests are taken into account. 0: `Operator` switching requests are taken into account.	Read
SrcIntAut	BOOL	`Automtaic` switching request to transfer the `SourceMode` to the `Internal` state	Read
SrcManAut	BOOL	`Automatic` switching request to transfer the `SourceMode` to the `Manual` state	Read

Outputs

Name	Type	Description	OPC UA access
SrcIntAct	BOOL	`SourceMode` is in the state `Internal`.	Read
SrcManAct	BOOL	`SourceMode` is in the state `Manual`.	Read

External variables

Name	Type	Description	OPC UA access
SrcIntOp	BOOL	`Operator` switching request to transfer the `SourceMode` to the `Internal` state. 0→1: `Operator` request 1→0: Request has been processed.	Read/write
SrcManOp	BOOL	`Operator` switching request to transfer the `SourceMode` to the `Manual` state. 0→1: `Operator` request 1→0: Request has been processed.	Read/write

Methods

Name	Description
SetInternal	Sets the state machine `SourceMode` of the interface to the state `Internal`.
SetManual	Sets the state machine `SourceMode` of the interface to the state `Manual`.

5.1.3 Service Mode

Three sources for switching requests and value specifications are available for the services:

- Manual operation (e.g. via OPC UA): `Operator`
- Internal PLC logic: `Automatic-Internal`
- Control system (e.g. via OPC UA): `Automatic-External`

Operation mode control

The operation mode control allows you to switch between the states `Offline`, `Operator` and `Automatic`.

In the `Offline` state, no new switching requests and value specifications are taken into account. In the `Operator` state, `*OP` switching requests are processed and in the `Automatic` state, `*Aut` switching requests are processed.

The current state is displayed via the outputs `StateOffAct`, `StateOpAct` and `StateAutAct`.

The state change is carried out via the internal PLC logic or via the operator. The input `StateChannel` is set via the internal PLC logic and specifies which source is taken into account during processing.

The states are prioritized as follows: `Offline` > `Operator` > `Automatic`.

This means that if all three states are requested at the same time, the state `Offline` is assumed. If `Operator` and `Automatic` are requested at the same time, the state `Operator` is assumed

Source selection (Automatic)

The source selection is used in the `Automatic` operation mode and allows you to switch between `Automatic-Internal` and `Automatic-External`.

`Automatic-Internal` value specifications are written by the internal PLC logic. The variable names are extended by the suffix `*Int`.

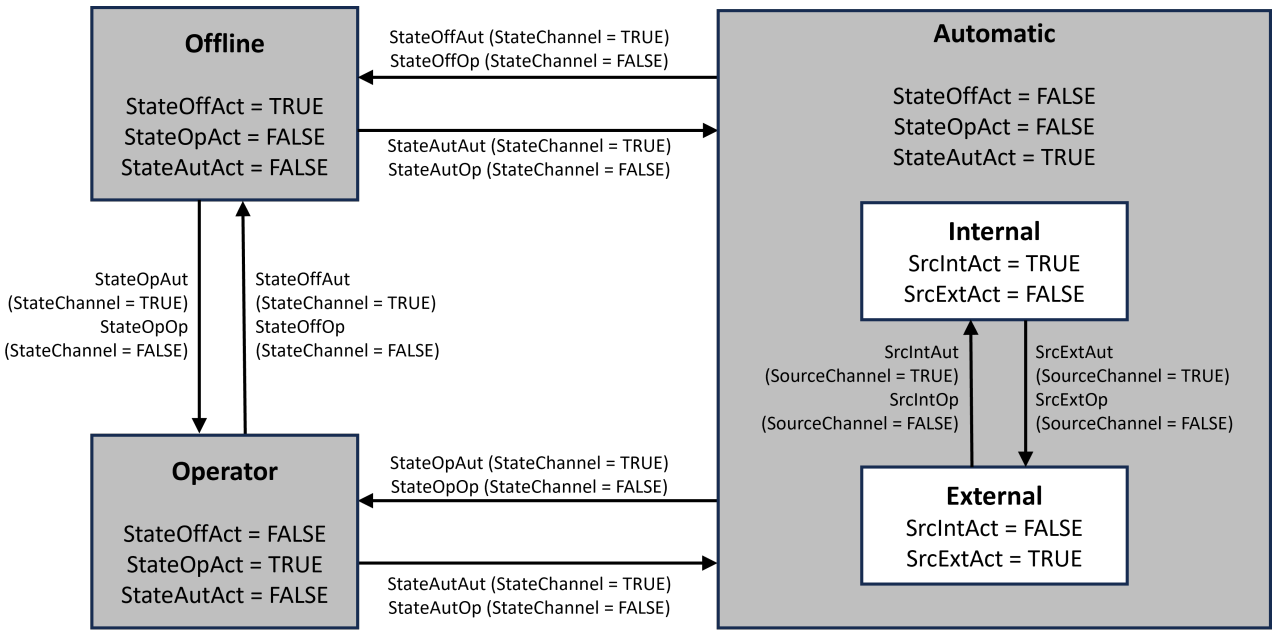
`Automatic-External` value specifications are written manually (e.g. via OPC UA).

`Operator` switching requests are made manually (e.g. via OPC UA). They are declared as external variables and cannot be written via the internal PLC logic. `Operator` switching requests add the suffix `*Op` to the variable name.

`Service Mode` is part of interfaces and cannot be used alone!

Diagram of the state machines:

The state machines work according to the following diagram:



5.2 Properties

This chapter describes the properties of the interfaces.

These are not used alone, but according to the intended use of the interfaces.

5.2.1 Name of the object

The name of the object is described by the variable `TagName`. It is the unique identifier of the object. The length is limited to 64 characters.

Changing the value during runtime is possible, but not allowed according to the MTP guideline!

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-

5.2.2 Object description

The object is described by the variable `TagDescription`.

Changing the value during runtime is possible, but not permitted according to the MTP guideline!

Name	Type	Description	OPC UA access
TagDescription	STRING	Description of the interface	-

5.2.3 OSLevel

The `OSLevel` is used to monitor operator levels. The use describes the following dependencies:

`OSLevel = 0`: Operation only from local level (local HMI).

`OSLevel = 1 – 255`: Operation takes place from the process control level (e.g. via OPC UA).

If the interface is to be used in accordance with the MTP guideline, the `OSLevel` may only be set from the control level. The internal logic only allows this value to be manually set to 0 if the connection to the control level is lost. The implementation of the operator levels is carried out by the creator of the control level and is not part of this documentation.

Name	Type	Description	OPC UA access
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write

5.2.4 WQC

The `WQC` (Worst Quality Code) describes the conditions under which the value of the interface was recorded and how trustworthy this value is. It is usually formed by the sensors or modules.

The values are not defined in more detail in the current version of the MTP guideline. However, they can be derived from the NE 184, for example.

Name	Type	Description	OPC UA access
WQC	BYTE	Worst Quality Code	Read

5.2.5 Value scaling

Values are displayed using value scaling. Variables with the suffixes `*SclMin` and `*SclMax` are available for this purpose.

According to the MTP guideline, these limit values cannot be changed at runtime, but are supported by the object.

Inputs

Name	Type	Description	OPC UA access
*SclMin	REAL/ DINT	Lower limit of the value representation	Read
*SclMax	REAL/ DINT	Upper limit of the value representation	Read

5.2.6 Unit

The unit of a value is described by the Unit variable with the suffix `*Unit` . The possible units are listed in the file type `E_MTP_Unit`.

Inputs

Name	Type	Description	OPC UA access
*Unit	INT	Enumeration value of the unit list	Read

5.2.7 Value limitation

Value specifications for objects in this library are limited by an upper and lower limit. The limit values for value specifications are described by variables with the suffixes `*Min` and `*Max`. Value specifications outside the limits are replaced accordingly by the lower or upper limit.

Inputs

Name	Type	Description	OPC UA access
*Min	REAL/ DINT	Lower limit of the value specification	Read
*Max	REAL/ DINT	Upper limit of the value specification	Read

5.2.8 Feedback

Feedback for switching requests or value specifications can be provided via the input variables with the suffix `*Fbk`. These feedback signals can come from field devices or be simulated.

Examples:

1. Feedback contact of a contactor
2. Current speed of a motor from the encoder

Inputs

Name	Type	Description	OPC UA access
*Fbk	REAL/ DINT/ BOOL	Return value	Read

5.2.9 Read back

The default values are read back via the variables with the suffix `*Rbk`. The current, unmodified value specification is returned by these variables.

5.2.10 Handshake procedure

The handshake procedure is used for some variables to transport information. A value, e.g. logical "1" (switching request), is written to a variable by the manual operation. This value is set back to a logical "0" after processing by the function block:

0 → 1: Switching request by manual operation

1 → 0: Switching request was read

The reset only describes that the value has been read. It does not mean that this value has also had an influence. The state-describing variable is used to show whether a value has influenced an output of the function block.

5.2.11 Locking

The locking system consists of a three-stage system that includes the states `Permit`, `Interlock` and `Protect`.

`Permit` is a switching release for switching requests. If `Permit = FALSE`, the object can still be operated. If the object is transferred to a state that corresponds to the safety position, further switching requests are no longer taken into account - the switching release is withdrawn.

`Interlock` is an interlock that withdraws the switching release and sets the object to the safety position.

`Protect` has the same functionality as `Interlock`. `Protect` does not reset automatically, but must be acknowledged via Reset [▶ 23].

The respective locking options can be activated using Enable variables `*En`.

Inputs

Name	Type	Description	OPC UA access
PermEn	BOOL	Enable `Permit`: 1: Enabled 0: Disabled	Read
Permit	BOOL	`Permit` signal 1: `Permit` inactive 0: `Permit` active	Read
IntlEn	BOOL	Enable `Interlock`: 1: Enabled 0: Disabled	Read
Interlock	BOOL	`Interlock` signal 1: `Interlock` inactive 0: `Interlock` active	Read
ProtEn	BOOL	Enable `Protect`: 1: Enabled 0: Disabled	Read
Protect	BOOL	`Protect` signal 1: `Protect` inactive 0: `Protect` active, `Reset` required	Read

5.2.12 Feedback monitoring

Feedback monitoring compares the states of the switching request `*Ctrl` with the respective feedback contacts `*Fbk`. A distinction is made between static and dynamic errors during monitoring. The monitoring function is enabled by the variable `MonEn` by an operator (e. g. via OPC UA) or initially in the module.

A static error `MonStatErr` is triggered by a change in the feedback contact if the switching request remains unchanged after the monitoring time `MonStatTi` has elapsed. An example of this is a motor where the supply voltage fails.

A dynamic fault `MonDynErr` is triggered by a change in the switching request without a resulting change in the feedback contact after the monitoring time `MonDynTi` has elapsed. This can be triggered, for example, by a contactor that remains "stuck".

The monitoring times in which the respective faults are triggered can be set by the internal PLC logic. The variable `MonSafePos` is used to describe the behavior of the interface when an error occurs.

Inputs

Name	Type	Description	OPC UA access
MonStatTi	REAL	Monitoring time for static errors [s]	Read
MonDynTi	REAL	Monitoring time for dynamic errors [s]	Read
MonSafePos	BOOL	Behavior of the interface after an error occurs: 1: Safe position should be approached.	Read

Name	Type	Description	OPC UA access
		0: The current state is retained.	

Outputs

Name	Type	Description	OPC UA access
MonStatErr	REAL	Static error: 1: active 0: inactive	Read
MonDynErr	REAL	Dynamic error: 1: active 0: inactive	Read

External variables

Name	Type	Description	OPC UA access
MonEn	BOOL	Enable `FeedbackMonitoring`: 1: `FeeedbackMonitoring` active 0: `FeedbackMonitoring` inactive	Read/write

5.2.13 Limit value monitoring

Limit value monitoring adds up to six monitorable limits to an interface. The limit value to be monitored is set via the `*Lim` variable by an operator (e. g. via OPC UA). Monitoring is enabled via the respective `*En` variable. If the upper limit values are exceeded or the lower limit values are not reached, the respective output `*Act` is set to `TRUE`.

Inputs

Name	Type	Description	OPC UA access
*AHEn	BOOL	Monitor limit `Alarm High`: 1: monitoring active 0: no monitoring	Read
*ALEn	BOOL	Monitor limit `Alarm Low`: 1: monitoring active 0: no monitoring	Read
*WHEn	BOOL	Monitor limit `Warning High`: 1: monitoring active 0: no monitoring	Read
*WLEn	BOOL	Monitor limit `Warning Low`: 1: monitoring active 0: no monitoring	Read
*THEn	BOOL	Monitor limit `Tolerance High`: 1: monitoring active 0: no monitoring	Read
*TLEn	BOOL	Monitor limit `Tolerance Low`: 1: monitoring active 0: no monitoring	Read

Outputs

Name	Type	Description	OPC UA access
*AHAct	BOOL	`Alarm High` limit exceeded.	Read
*ALAct	BOOL	`Alarm Low` limit not reached.	Read
*WHAct	BOOL	`Warning High` limit exceeded.	Read
*WLAct	BOOL	`Warning Low` limit not reached.	Read
*THAct	BOOL	`Tolerance High` limit exceeded.	Read
*TLAct	BOOL	`Tolerance Low` limit not reached.	Read

External variables

Name	Type	Description	OPC UA access
*AHLim	REAL / DINT	`Alarm High` limit	Read/write
*ALLim	REAL / DINT	`Alarm Low` limit	Read/write
*WHLim	REAL / DINT	`Warning High` limit	Read/write
*WLLim	REAL / DINT	`Warning Low` limit	Read/write
*THLim	REAL / DINT	`Tolerance High` limit	Read/write
*TLLim	REAL / DINT	`Tolerance Low` limit	Read/write

5.2.14 Reset

`Reset` resets pending `*Err` messages and the `Protect-Verriegelung`. `Reset` takes place via two equally prioritized inputs: `ResetOp` and `ResetAut`. The reset by manual operation (e. g. via OPC UA) is carried out via a Handshake procedure [▶ 20].

Inputs

Name	Type	Description	OPC UA access
ResetAut	BOOL	`Automatic` reset switching request: 1: Reset requested. 0: No reset requested.	Read

External variables

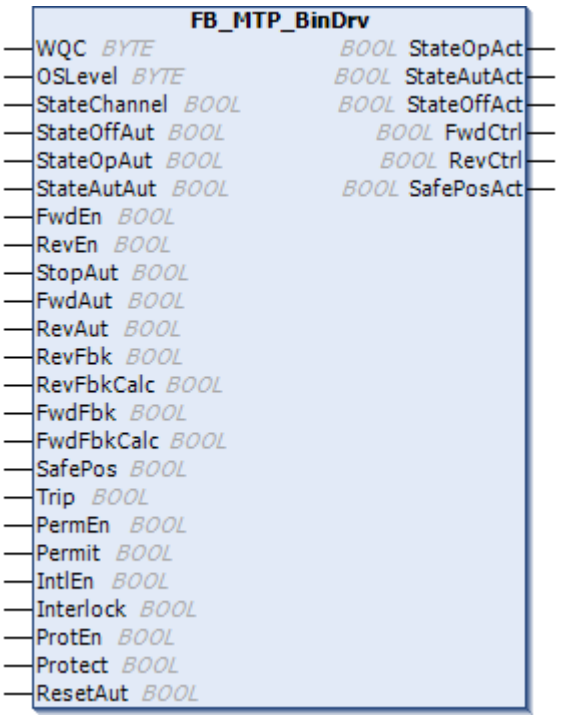
Name	Type	Description	OPC UA access
ResetOp	BOOL	`Operator` reset switching request: 0→1: `Operator` request 1→0: Request has been processed.	Read/write

5.3 Function blocks

5.3.1 ActiveElements

`ActiveElements` are objects that allow access to actuators from different sources. The sources can be the internal PLC logic or manual operation. The actuators include valves, electric drives or PID controllers. Access is controlled via state machines.

5.3.1.1 FB_MTP_BinDrv



The function block `FB_MTP_BinDrv` is an object for controlling a binary drive from different sources: internal PLC logic or manual operation (e. g. via OPC UA). Switching requests are managed via the state machine of Operation Mode [▶ 13]. The OPC UA access rights are described in the variable tables.

Controlling the drive

The direction of rotation of the drive is specified via the switching requests `Fwd*`, `Rev*` and `Stop*`. The state machine of the Operation Mode [▶ 13] manages whether and from which source new switching requests are processed. With simultaneous requests from `Fwd*` and `Rev*`, both are prioritized equally and the drive remains in its state. `Stop*` is prioritized highest at `SafePos = FALSE` with simultaneous requests from `Fwd*`, `Rev*` and `Stop*`. If `SafePos = TRUE`, the switching requests `Fwd*`, `Rev*` and `Stop*` are prioritized equally and the drive remains in its current state in the event of simultaneous requests.

When Locking [▶ 20] is active, it is no longer possible to control the drive, depending on the type of interlock.

Safety position

The safety position is defined via the variables `SafePos`, `SafePosDirEn` and `SafePosDir`.

The variable `SafePos` is used to specify whether the safety position of the drive is the de-energized state or an active state.

> The variables `SafePosDirEn` and `SafePosDir` are not part of the MTP guideline. They have also been implemented and can be used to define a safety position in the active state. These variables are not listed in the MTP file and are not made available via OPC UA.

Safety position: de-energized state

In the de-energized state, the safety position of the drive is: `FwdCtrl = FALSE` and `RevCtrl = FALSE`.

Safety position: active state

For the active state, the default direction of rotation can be enabled (`SafePosDirEn = TRUE`) and the direction of rotation can be specified (`SafePosDir`).

If the specification of the direction of rotation is disabled `SafePosDirEn = FALSE`, the current movement state (forward, reverse or stop) is the safety position.

If the default setting is enabled, the safety position `SafePosDirEn = TRUE` forward (`SafePosDir` = `FALSE`) or reverse (`SafePosDir` = `TRUE`).

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

OSLevel [▶ 18]

Feedback [▶ 20]

Locking [▶ 20]

Reset [▶ 23]

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
State*		See Operation Mode [▶ 13]	
FwdEn	BOOL	Enable forward movement: 1: Enabled 0: Disabled	Read
RevEn	BOOL	Enable reverse movement: 1: Enabled 0: Disabled	Read
StopAut	BOOL	`Automatic` switching request to set the drive to stop: (Relevant if `StateAutAct`) 1: Execute stop. 0: No stop	Read
FwdAut	BOOL	`Automatic` switching request to set the drive in forward motion: (Relevant if `StateAutAct` & `FwdEn`) 1: Execute forward movement. 0: Do not execute any forward movement.	Read
RevAut	BOOL	`Automatic` switching request to set the drive in forward motion: (Relevant if `StateAutAct` & `RevEn`)	Read

Name	Type	Description	OPC UA access
		1: Execute reverse movement. 0: Do not execute any reverse movement.	
FwdFbkCalc	BOOL	Source of the feedback signal Forward movement: 1: Calculated 0: Sensor	Read
FwdFbk	BOOL	Feedback signal Forward movement: 1: Forward movement 0: No forward movement	Read
RevFbkCalc	BOOL	Source of the feedback signal Reverse movement: 1: Calculated 0: Sensor	Read
RevFbk	BOOL	Feedback signal Reverse movement: 1: Reverse movement 0: No reverse movement	Read
Perm* Int* Prot*		See Locking [▶ 20]	
Trip	BOOL	Signaling contact for motor protection: 1: No error 0: Motor protection tripped.	Read
SafePos	BOOL	1: Retain current state. 0: Stop	Read
SafePosDirEn	BOOL	1: Rotation direction specification for `SafePos = TRUE` via the variable `SafePosDir` 0: No rotation direction specified	–
SafePosDir	BOOL	Rotation direction specification for `SafePos = TRUE` and `SafePosDirRpmEn = TRUE` 1: reverse 0: forward	–
ResetAut	BOOL	`Automatic` reset switching request: 1: Reset requested. 0: No reset requested.	Read

Outputs

Name	Type	Description	OPC UA access
State*		See Operation Mode [▶ 13]	
FwdCtrl	BOOL	Forward movement: 1: active 0: inactive	Read
RevCtrl	BOOL	Reverse movement: 1: active 0: inactive	Read
SafePosAct	BOOL	Safety position enabled: 1: Safety position is active. 0: Safety position is not active.	Read

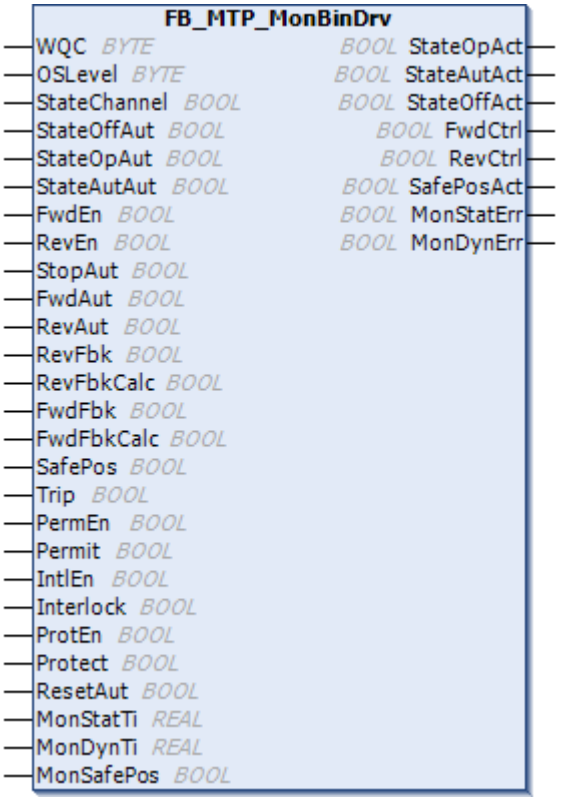
External variables

Name	Type	Description	OPC UA access
State*		See Operation Mode [▶ 13]	
StopOp	BOOL	`Operator` switching request to set the drive to stop: (Relevant if `StateOpAct`) 1: Execute stop. 0: Do not execute a stop.	Read/write
FwdOp	BOOL	`Operator` switching request to set the drive in forward movement: (Relevant if `StateOpAct` & `FwdEn`) 1: Execute forward movement. 0: Do not execute any forward movement.	Read/write
RevOp	BOOL	`Operator` switching request to set the drive in forward movement: (Relevant if `StateOpAct` & `RevEn`) 1: Execute reverse movement. 0: Do not execute any reverse movement.	Read/write
ResetOp	BOOL	`Operator` reset switching request: 0→1: `Operator` request 1→0: Request has been processed.	Read/write

Methods

Name	Description
SetOffline SetOperator SetAutomatic	See Operation Mode [▶ 13]

5.3.1.2 FB_MTP_MonBinDrv



The function block `FB_MTP_MonBinDrv` is an object for controlling a binary drive from different sources: internal PLC logic or manual operation (e. g. via OPC UA). Switching requests are managed via the state machine of Operation Mode [▶ 13]. It also includes the option of monitoring the output values with the respective feedback signals. The OPC UA access rights are described in the variable tables.

Controlling the drive

The direction of rotation of the drive is specified via the switching requests `Fwd*`, `Rev*` and `Stop*`. The state machine of the Operation Mode [▶ 13] manages whether and from which source new switching requests are processed. With simultaneous requests from `Fwd*` and `Rev*`, both are prioritized equally and the drive remains in its state. `Stop*` is prioritized highest at `SafePos = FALSE` with simultaneous requests from `Fwd*`, `Rev*` and `Stop*`. If `SafePos = TRUE`, the switching requests `Fwd*`, `Rev*` and `Stop*` are prioritized equally and the drive remains in its current state in the event of simultaneous requests.

When Locking [▶ 20] is active, it is no longer possible to control the drive, depending on the type of interlock.

Safety position

The safety position is defined via the variables `SafePos`, `SafePosDirEn` and `SafePosDir`.

The variable `SafePos` is used to specify whether the safety position of the drive is the de-energized state or an active state.

> The variables `SafePosDirEn` and `SafePosDir` are not part of the MTP guideline. They have also been implemented and can be used to define a safety position in the active state. These variables are not listed in the MTP file and are not made available via OPC UA.

Safety position: de-energized state

In the de-energized state, the safety position of the drive is: `FwdCtrl = FALSE` and `RevCtrl = FALSE`.

Safety position: active state

For the active state, the default direction of rotation can be enabled (`SafePosDirEn = TRUE`) and the direction of rotation can be specified (`SafePosDir`).

If the specification of the direction of rotation is disabled `SafePosDirEn = FALSE`, the current movement state (forward, reverse or stop) is the safety position.

If the default setting is enabled, the safety position `SafePosDirEn = TRUE` forward (`SafePosDir = FALSE`) or reverse (`SafePosDir = TRUE`).

Monitoring

The control and the respective Feedback [▶ 20] can be monitored via Feedback monitoring [▶ 21].

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

OSLevel [▶ 18]

Locking [▶ 20]

Reset [▶ 23]

Inheritance hierarchy

FB_MTP_BinDrv

  FB_MTP_MonBinDrv

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
State*		See Operation Mode [▶ 13]	
FwdEn	BOOL	Enable forward movement: 1: Enabled 0: Disabled	Read
RevEn	BOOL	Enable reverse movement: 1: Enabled 0: Disabled	Read
StopAut	BOOL	`Automatic` switching request to set the drive to stop: (Relevant if `StateAutAct`) 1: Execute stop. 0: No stop	Read
FwdAut	BOOL	`Automatic` switching request to set the drive in forward motion: (Relevant if `StateAutAct` & `FwdEn`) 1: Execute forward movement. 0: Do not execute any forward movement.	Read
RevAut	BOOL	`Automatic` switching request to set the drive in forward motion:	Read

Name	Type	Description	OPC UA access
		(Relevant if `StateAutAct & RevEn`) 1: Execute reverse movement. 0: Do not execute any reverse movement.	
FwdFbkCalc	BOOL	Source of the feedback signal Forward movement: 1: Calculated 0: Sensor	Read
FwdFbk	BOOL	Feedback signal Forward movement: 1: Forward movement 0: No forward movement	Read
RevFbkCalc	BOOL	Source of the feedback signal Reverse movement: 1: Calculated 0: Sensor	Read
RevFbk	BOOL	Feedback signal Reverse movement: 1: Reverse movement 0: No reverse movement	Read
Perm* Int* Prot*		See Locking [▶ 20]	
Trip	BOOL	Signaling contact for motor protection: 1: No error 0: Motor protection tripped.	Read
SafePos	BOOL	1: Retain current state. 0: Stop	Read
SafePosDirEn	BOOL	1: Rotation direction specification for `SafePos = TRUE` via the variable `SafePosDir` 0: No rotation direction specified	–
SafePosDir	BOOL	Rotation direction specification for `SafePos = TRUE` and `SafePosDirRpmEn = TRUE` 1: reverse 0: forward	–
MonStatTi	REAL	Monitoring time for static errors [s]	Read
MonDynTi	REAL	Monitoring time for dynamic errors [s]	Read
MonSafePos	BOOL	Behavior of the interface after an error occurs: 1: Safe position should be approached. 0: The current state is retained.	Read
ResetAut	BOOL	`Automatic` reset switching request: 1: Reset requested. 0: No reset requested.	Read

Outputs

Name	Type	Description	OPC UA access
State*		See Operation Mode [▶ 13]	
FwdCtrl	BOOL	Forward movement: 1: active 0: inactive	Read
RevCtrl	BOOL	Reverse movement:	Read

Name	Type	Description	OPC UA access
		1: active 0: inactive	
SafePosAct	BOOL	Safety position enabled: 1: Safety position is active. 0: Safety position is not active.	Read
MonStatErr	REAL	Static error: 1: active 0: inactive	Read
MonDynErr	REAL	Dynamic error: 1: active 0: inactive	Read

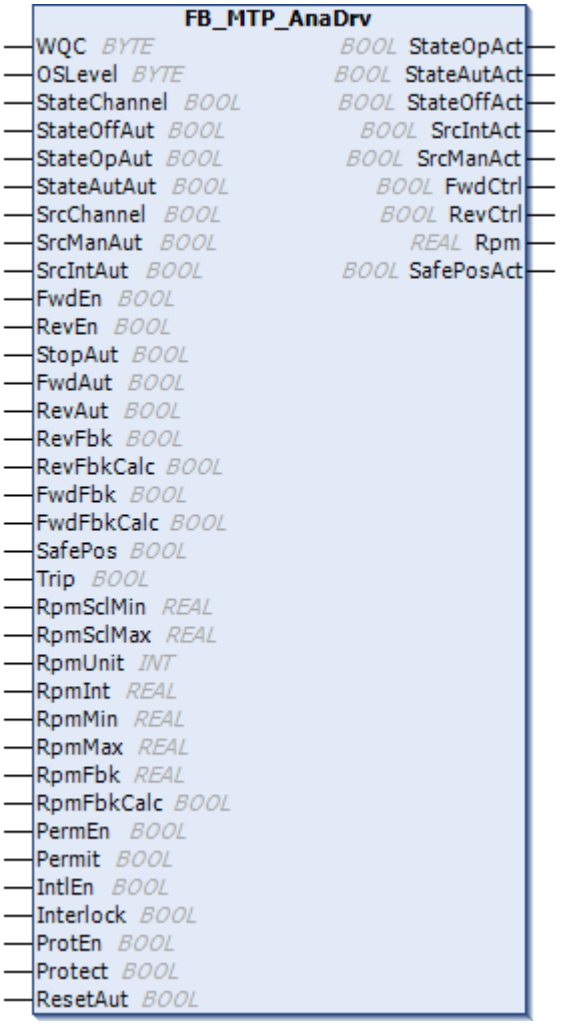
External variables

Name	Type	Description	OPC UA access
State*		See Operation Mode [▶ 13]	
StopOp	BOOL	`Operator` switching request to set the drive to stop: (Relevant if `StateOpAct`) 1: Execute stop. 0: Do not execute a stop.	Read/write
FwdOp	BOOL	`Operator` switching request to set the drive in forward movement: (Relevant if `StateOpAct` & `FwdEn`) 1: Execute forward movement. 0: Do not execute any forward movement.	Read/write
RevOp	BOOL	`Operator` switching request to set the drive in forward movement: (Relevant if `StateOpAct` & `RevEn`) 1: Execute reverse movement. 0: Do not execute any reverse movement.	Read/write
MonEn	BOOL	Enable `FeedbackMonitoring`: 1: `FeeedbackMonitoring` active 0: `FeedbackMonitoring` inactive	Read/write
ResetOp	BOOL	`Operator` reset switching request: 0→1: `Operator` request 1→0: Request has been processed.	Read/write

Methods

Name	Description
SetOffline SetOperator SetAutomatic	See Operation Mode [▶ 13]

5.3.1.3 FB_MTP_AnaDrv



The function block `FB_MTP_AnaDrv` is an object for controlling a drive with variable speed from different sources: internal PLC logic or manual operation (e. g. via OPC UA). Switching requests and value specifications are managed independently of each other via the state machines of Operation Mode [▶ 13] and Source Mode [▶ 15]. The OPC UA access rights are described in the variable tables.

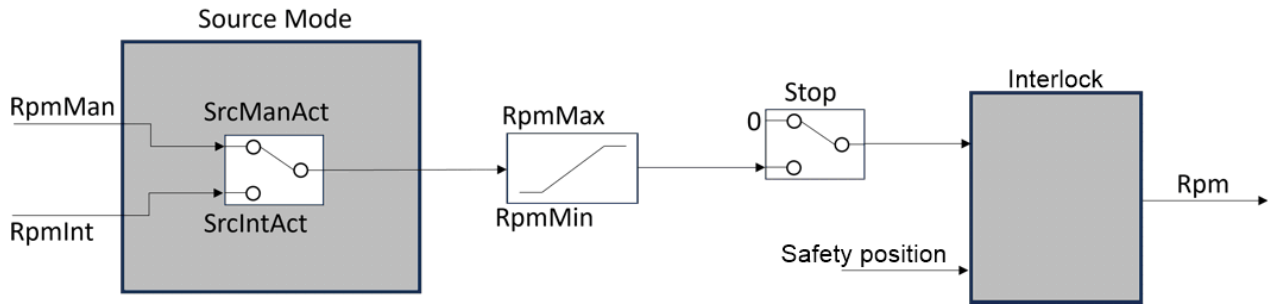
Controlling the drive

The direction of rotation of the drive is specified via the switching requests `Fwd*`, `Rev*` and `Stop*`. The state machine of the Operation Mode [▶ 13] manages whether and from which source new switching requests are processed. With simultaneous requests from `Fwd*` and `Rev*`, both are prioritized equally and the drive remains in its state. `Stop*` is prioritized highest at `SafePos = FALSE` with simultaneous requests from `Fwd*`, `Rev*` and `Stop*`. If `SafePos = TRUE`, the switching requests `Fwd*`, `Rev*` and `Stop*` are prioritized equally and the drive remains in its current state in the event of simultaneous requests.

When Locking [▶ 20] is active, it is no longer possible to control the drive, depending on the type of interlock.

Setpoint specification

The speed is set via `Rpm*` variables. The speed specification is managed by the state machine Source Mode [▶ 15] and is output according to the diagram below:



Safety position

The safety position is defined via the variables `SafePos`, `SafePosDirRpmEn`, `SafePosDir` and `SafePosRpm`.

The variable `SafePos` is used to specify whether the safety position of the drive is the de-energized state or an active state.

> The variables `SafePosDirRpmEn`, `SafePosDir` and `SafePosRpm` are not part of the MTP guideline. They have also been implemented and can be used to define a safety position in the active state. These variables are not listed in the MTP file and are not made available via OPC UA.

Safety position: de-energized state (`SafePos = FALSE`)

In the de-energized state, the drive is stopped (`FwdCtrl` = `FALSE`, `RevCtrl` = `FALSE` and `Rpm = 0`)

Safety position: active state (`SafePos = TRUE`)

If the state is active, the default direction of rotation and speed can be activated (`SafePosDirRpmEn = TRUE`) and the direction of rotation (`SafePosDir`) or speed (`SafePosRpm`) can be specified.

If `SafePosDirRpmEn = FALSE`, the current movement state (forward, reverse or stop) is the safety position.

If the default setting (`SafePosDirRpmEn = TRUE`) is activated, the safety position is forward (`SafePosDir = FALSE`) or reverse (`SafePosDir = TRUE`) with the defined safety speed (`SafePosRpm`).

The following table shows the operating options for the drive (`Fwd*`, `Rev*`, `Stop*` and `Rpm*`) when `Permit` is active, depending on the initial state of the drive (stop drive, set drive to forward movement and set drive to reverse movement) and the defined safety position:

SafePos	SafePos DirRpmEn	SafePos-Dir	Drive set to Stop	Drive set to Forward	Drive set to Reverse
FALSE	-	-	SafePosAct = TRUE Rev* + Fwd*: x Rpm*: x	SafePosAct = FALSE Rev* + Stop*: ✓ Rpm*: ✓	SafePosAct = FALSE Fwd* + Stop*: ✓ Rpm*: ✓
TRUE	FALSE	-	SafePosAct = TRUE Rev* + Fwd*: x Rpm*: x	SafePosAct = TRUE Rev* + Stop*: x Rpm*: x	SafePosAct = TRUE Fwd* + Stop*: x Rpm*: x
TRUE	TRUE	FALSE (Fwd)	SafePosAct = FALSE Rev* + Fwd*: ✓ Rpm*: ✓	SafePosAct = TRUE Rev* + Stop*: x Rpm*: x (SafePosRpm)	SafePosAct = FALSE Fwd* + Stop*: ✓ Rpm*: ✓

SafePos	SafePos DirRpmEn	SafePos-Dir	Drive set to Stop	Drive set to Forward	Drive set to Reverse
TRUE	TRUE	TRUE (Rev)	SafePosAct = FALSE Rev* + Fwd*: ✓ Rpm*: ✓	SafePosAct = FALSE Rev* + Stop*: ✓ Rpm*: ✓	SafePosAct = TRUE Fwd* + Stop*: x Rpm*: x (SafePosRpm)

x: Switching requests and/or value specifications are no longer taken into account

✓: Switching requests and/or value specifications are taken into account

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

OSLevel [▶ 18]

Unit [▶ 19]

Feedback [▶ 20]

Locking [▶ 20]

Reset [▶ 23]

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
Src*		See Source Mode [▶ 15]	
RpmInt	REAL	Internal Speed setpoint specification	Read
RpmUnit	INT	Speed Unit	Read
RpmMin	REAL	Upper limit of the speed setpoint	Read
RpmMax	REAL	Lower limit of the speed setpoint	Read
RpmFbkCalc	BOOL	Source of the actual speed value: 1: Calculated 0: Sensor	Read
RpmFbk	REAL	Actual speed value	Read
RpmSclMin	REAL	Speed scale start	Read
RpmSclMax	REAL	Speed scale end	Read
Perm* Int* Prot*		See Locking [▶ 20]	
Trip	BOOL	Signaling contact for motor protection: 1: No error 0: Motor protection tripped.	Read

Name	Type	Description	OPC UA access
SafePos	BOOL	1: Retain current state. 0: Stop	Read
SafePosDirRpmEn	BOOL	1: Rotation direction specification for `SafePos = TRUE` via the variable `SafePosDir` 0: No rotation direction specified	–
SafePosDir	BOOL	Rotation direction specification for `SafePos = TRUE` and `SafePosDirRpmEn = TRUE` 1: reverse 0: forward	–
SafePosRpm	REAL	Speed specification for safety position, if `SafePos = TRUE` and `SafePosDirRpmEn = TRUE`	–
ResetAut	BOOL	`Automatic` reset switching request: 1: Reset requested. 0: No reset requested.	Read

Outputs

Name	Type	Description	OPC UA access
State*		See Operation Mode [▶ 13]	
FwdCtrl	BOOL	Forward movement: 1: active 0: inactive	Read
RevCtrl	BOOL	Reverse movement: 1: active 0: inactive	Read
Src*		See Source Mode [▶ 15]	
Rpm	REAL	Speed setpoint on drive	Read
SafePosAct	BOOL	Safety position enabled: 1: Safety position is active. 0: Safety position is not active.	Read

External variables

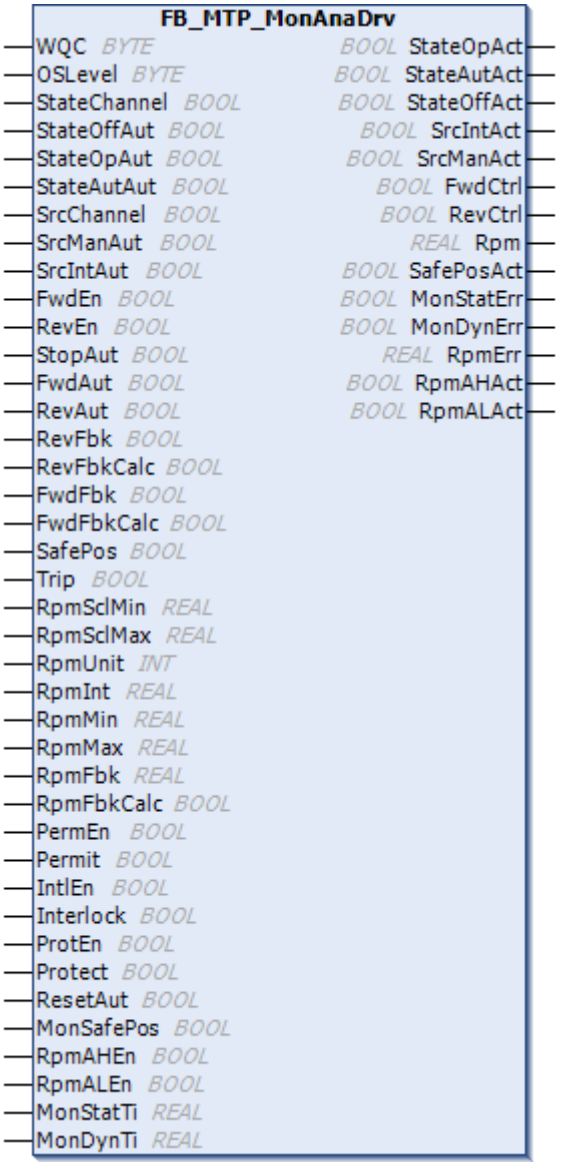
Name	Type	Description	OPC UA access
State*		See Operation Mode [▶ 13]	
StopOp	BOOL	`Operator` switching request to set the drive to stop: (Relevant if `StateOpAct`) 1: Execute stop. 0: Do not execute a stop.	Read/write
FwdOp	BOOL	`Operator` switching request to set the drive in forward movement: (Relevant if `StateOpAct & FwdEn`) 1: Execute forward movement. 0: Do not execute any forward movement.	Read/write
RevOp	BOOL	`Operator` switching request to set the drive in forward movement: (Relevant if `StateOpAct & RevEn`) 1: Execute reverse movement.	Read/write

Name	Type	Description	OPC UA access
		0: Do not execute any reverse movement.	
Src*		See Source Mode [▶ 15]	
RpmMan	REAL	`Manual` setpoint specification for the speed	Read/write
RpmRbk	REAL	Unprocessed value of the `Operator` speed value specification	Read
ResetOp	BOOL	`Operator` reset switching request: 0→1: `Operator` request 1→0: Request has been processed.	Read/write

Methods

Name	Description
SetOffline SetOperator SetAutomatic	See Operation Mode [▶ 13]
SetInternal SetManual	See Source Mode [▶ 15]

5.3.1.4 FB_MTP_MonAnaDrv



The function block `FB_MTP_MonAnaDrv` is an object for controlling a drive with variable speed from different sources: internal PLC logic or manual operation (e. g. via OPC UA). Switching requests and value specifications are managed independently of each other via the state machines of Operation Mode [▶ 13] and Source Mode [▶ 15] . It also includes the option of monitoring the output values with the respective feedback signals. The OPC UA access rights are described in the variable tables.

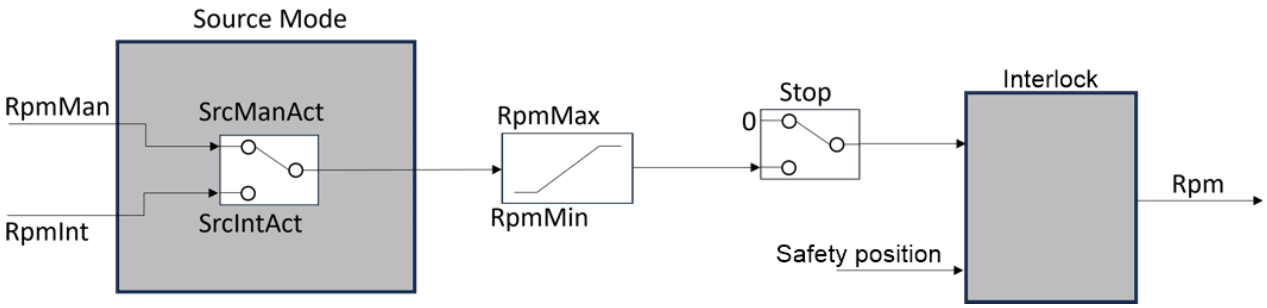
Controlling the drive

The direction of rotation of the drive is specified via the switching requests `Fwd*`, `Rev*` and `Stop*`. The state machine of the Operation Mode [▶ 13] manages whether and from which source new switching requests are processed. With simultaneous requests from `Fwd*` and `Rev*`, both are prioritized equally and the drive remains in its state. `Stop*` is prioritized highest at `SafePos = FALSE` with simultaneous requests from `Fwd*`, `Rev*` and `Stop*`. If `SafePos = TRUE`, the switching requests `Fwd*`, `Rev*` and `Stop*` are prioritized equally and the drive remains in its current state in the event of simultaneous requests.

When Locking [▶ 20] is active, it is no longer possible to control the drive, depending on the type of interlock.

Setpoint specification

The speed is set via `Rpm*` variables. The speed specification is managed by the state machine Source Mode [▶ 15] and is output according to the diagram below:



Safety position

The safety position is defined via the variables `SafePos`, `SafePosDirRpmEn`, `SafePosDir` and `SafePosRpm`.

The variable `SafePos` is used to specify whether the safety position of the drive is the de-energized state or an active state.

> The variables `SafePosDirRpmEn`, `SafePosDir` and `SafePosRpm` are not part of the MTP guideline. They have also been implemented and can be used to define a safety position in the active state. These variables are not listed in the MTP file and are not made available via OPC UA.

Safety position: de-energized state (`SafePos = FALSE`)

In the de-energized state, the drive is stopped (`FwdCtrl` = `FALSE`, `RevCtrl` = `FALSE` and `Rpm = 0`)

Safety position: active state (`SafePos = TRUE`)

If the state is active, the default direction of rotation and speed can be activated (`SafePosDirRpmEn = TRUE`) and the direction of rotation (`SafePosDir`) or speed (`SafePosRpm`) can be specified.

If `SafePosDirRpmEn = FALSE`, the current movement state (forward, reverse or stop) is the safety position.

If the default setting (`SafePosDirRpmEn = TRUE`) is activated, the safety position is forward (`SafePosDir = FALSE`) or reverse (`SafePosDir = TRUE`) with the defined safety speed (`SafePosRpm`).

The following table shows the operating options for the drive (`Fwd*`, `Rev*`, `Stop*` and `Rpm*`) when `Permit` is active, depending on the initial state of the drive (stop drive, set drive to forward movement and set drive to reverse movement) and the defined safety position:

SafePos	SafePos DirRpmEn	SafePos-Dir	Drive set to Stop	Drive set to Forward	Drive set to Reverse
FALSE	-	-	SafePosAct = TRUE Rev* + Fwd*: x Rpm*: x	SafePosAct = FALSE Rev* + Stop*: ✓ Rpm*: ✓	SafePosAct = FALSE Fwd* + Stop*: ✓ Rpm*: ✓
TRUE	FALSE	-	SafePosAct = TRUE Rev* + Fwd*: x Rpm*: x	SafePosAct = TRUE Rev* + Stop*: x Rpm*: x	SafePosAct = TRUE Fwd* + Stop*: x Rpm*: x
TRUE	TRUE	FALSE (Fwd)	SafePosAct = FALSE Rev* + Fwd*: ✓ Rpm*: ✓	SafePosAct = TRUE Rev* + Stop*: x Rpm*: x (SafePosRpm)	SafePosAct = FALSE Fwd* + Stop*: ✓ Rpm*: ✓

SafePos	SafePos DirRpmEn	SafePos-Dir	Drive set to Stop	Drive set to Forward	Drive set to Reverse
TRUE	TRUE	TRUE (Rev)	SafePosAct = FALSE Rev* + Fwd*: ✓ Rpm*: ✓	SafePosAct = FALSE Rev* + Stop*: ✓ Rpm*: ✓	SafePosAct = TRUE Fwd* + Stop*: x Rpm*: x (SafePosRpm)

x: Switching requests and/or value specifications are no longer taken into account

✓: Switching requests and/or value specifications are taken into account

Monitoring

The control and the respective Feedback [▶ 20] can be monitored via Feedback monitoring [▶ 21].

The output RpmErr indicates the current speed deviation RpmErr = Rpm – RpmFbk. The speed deviation can be monitored for an upper and lower limit (Alarm High and Alarm Low) using the Limit value monitoring [▶ 22].

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

OSLevel [▶ 18]

Value scaling [▶ 19]

Value limitation [▶ 20]

Locking [▶ 20]

Reset [▶ 23]

Inheritance hierarchy

FB_MTP_AnaDrv

  FB_MTP_MonAnaDrv

Inputs

Name	Type	Description	OPC UA access
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
State*		See Operation Mode [▶ 13]	
FwdEn	BOOL	Enable forward movement: 1: Enabled 0: Disabled	Read
RevEn	BOOL	Enable reverse movement: 1: Enabled 0: Disabled	Read
StopAut	BOOL	Automatic switching request to set the drive to stop:	Read

Name	Type	Description	OPC UA access
		(Relevant if `StateAutAct`) 1: Execute stop. 0: No stop	
FwdAut	BOOL	`Automatic` switching request to set the drive in forward motion: (Relevant if `StateAutAct` & `FwdEn`) 1: Execute forward movement. 0: Do not execute any forward movement.	Read
RevAut	BOOL	`Automatic` switching request to set the drive in forward motion: (Relevant if `StateAutAct` & `RevEn`) 1: Execute reverse movement. 0: Do not execute any reverse movement.	Read
FwdFbkCalc	BOOL	Source of the feedback signal Forward movement: 1: Calculated 0: Sensor	Read
FwdFbk	BOOL	Feedback signal Forward movement: 1: Forward movement 0: No forward movement	Read
RevFbkCalc	BOOL	Source of the feedback signal Reverse movement: 1: Calculated 0: Sensor	Read
RevFbk	BOOL	Feedback signal Reverse movement: 1: Reverse movement 0: No reverse movement	Read
Src*		See Source Mode [▶ 15]	
RpmInt	REAL	`Internal` Speed setpoint specification	Read
RpmUnit	INT	Speed Unit	Read
RpmMin	REAL	Upper limit of the speed setpoint	Read
RpmMax	REAL	Lower limit of the speed setpoint	Read
RpmFbkCalc	BOOL	Source of the actual speed value: 1: Calculated 0: Sensor	Read
RpmFbk	REAL	Actual speed value	Read
RpmSclMin	REAL	Speed scale start	Read
RpmSclMax	REAL	Speed scale end	Read
Perm* Int* Prot*		See Locking [▶ 20]	
Trip	BOOL	Signaling contact for motor protection: 1: No error 0: Motor protection tripped.	Read
SafePos	BOOL	1: Retain current state. 0: Stop	Read
SafePosDirRpmEn	BOOL	1: Rotation direction specification for `SafePos = TRUE` via the variable `SafePosDir` 0: No rotation direction specified	–

Name	Type	Description	OPC UA access
SafePosDir	BOOL	Rotation direction specification for `SafePos = TRUE` and `SafePosDirRpmEn = TRUE` 1: reverse 0: forward	–
SafePosRpm	REAL	Speed specification for safety position, if `SafePos = TRUE` and `SafePosDirRpmEn = TRUE`	–
MonStatTi	REAL	Monitoring time for static errors [s]	Read
MonDynTi	REAL	Monitoring time for dynamic errors [s]	Read
MonSafePos	BOOL	Behavior of the interface after an error occurs: 1: Safe position should be approached. 0: The current state is retained.	Read
RpmAHEn	BOOL	Monitor limit `Alarm High`: 1: monitoring active 0: No monitoring	Read
RpmALEn	BOOL	Monitor limit `Alarm Low`: 1: monitoring active 0: No monitoring	Read
ResetAut	BOOL	`Automatic` reset switching request: 1: Reset requested. 0: No reset requested.	Read

Outputs

Name	Type	Description	OPC UA access
State*		See Operation Mode [▶ 13]	
FwdCtrl	BOOL	Forward movement: 1: active 0: inactive	Read
RevCtrl	BOOL	Reverse movement: 1: active 0: inactive	Read
Src*		See Source Mode [▶ 15]	
Rpm	REAL	Speed setpoint on drive	Read
MonStatErr	REAL	Static error: 1: active 0: inactive	Read
MonDynErr	REAL	Dynamic error: 1: active 0: inactive	Read
RpmErr	REAL	Speed deviation: `RpmErr = Rpm - RpmFbk`	
RpmAHAct	BOOL	`Alarm High` limit exceeded.	Read
RpmALAct	BOOL	`Alarm Low` limit not reached.	Read

External variables

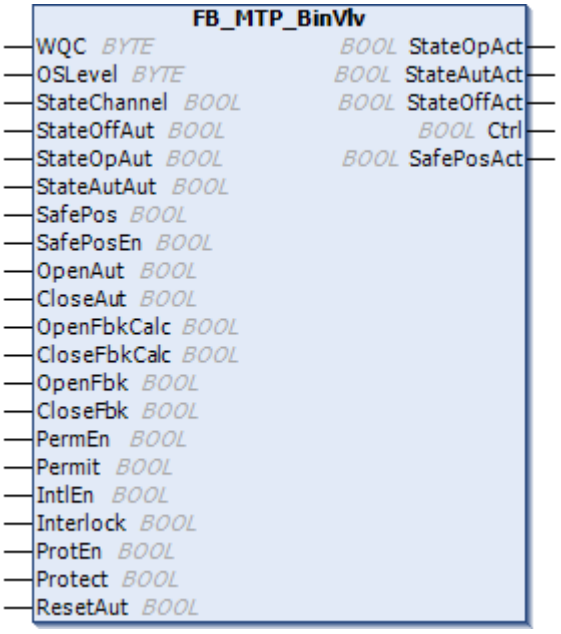
Name	Type	Description	OPC UA access
State*		See Operation Mode [▶ 13]	

Name	Type	Description	OPC UA access
StopOp	BOOL	`Operator` switching request to set the drive to stop: (Relevant if `StateOpAct`) 1: Execute stop. 0: Do not execute a stop.	Read/write
FwdOp	BOOL	`Operator` switching request to set the drive in forward movement: (Relevant if `StateOpAct` & `FwdEn`) 1: Execute forward movement. 0: Do not execute any forward movement.	Read/write
RevOp	BOOL	`Operator` switching request to set the drive in forward movement: (Relevant if `StateOpAct` & `RevEn`) 1: Execute reverse movement. 0: Do not execute any reverse movement.	Read/write
Src*		See Source Mode [▶ 15]	
RpmMan	REAL	`Manual` setpoint specification for the speed	Read/write
RpmRbk	REAL	Unprocessed value of the `Operator` speed value specification	Read
MonEn	BOOL	Enable `FeedbackMonitoring`: 1: `FeeedbackMonitoring` active 0: `FeedbackMonitoring` inactive	Read/write
RpmAHLim	REAL	`Alarm High` limit (>0)	Read/write
RpmALLim	REAL	`Alarm Low` limit (<0)	Read/write

Methods

Name	Description
SetOffline SetOperator SetAutomatic	See Operation Mode [▶ 13]
SetInternal SetManual	See Source Mode [▶ 15]

5.3.1.5 FB_MTP_BinVlv



The function block `FB_MTP_BinVlv` is an object for controlling a binary valve from different sources: internal PLC logic or manual operation (e. g. via OPC UA). Switching requests are managed via the state machine Operation Mode [▶ 13]. The OPC UA access rights are described in the variable tables.

Controlling the valve

The specification for opening and closing the valve is made via the switching requests `Open*` and `Close*`. The state machine of the Operation Mode [▶ 13] manages whether and from which source new switching requests are processed. For simultaneous requests from `Open*` and `Close*`, `Close*` has the highest priority. If Locking [▶ 20] is active, control of the outputs via the switching requests is prevented or the safety position is assumed.

Safety position

The safety position is defined via the variables `SafePosEn` and `SafePos` .

The variable `SafePosEn` is used to describe whether the valve has a fixed safety position (`SafePosEn = TRUE`) or whether the current position should be maintained (`SafePosEn = FALSE`).

The safety position is described with `SafePos`:

`SafePos = FALSE`: Safety position of the valve: closed

`SafePos = TRUE`: Safety position of the valve: open

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

OSLevel [▶ 18]

Feedback [▶ 20]

Locking [▶ 20]

Reset [▶ 23]

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
State*		See Operation Mode [▶ 13]	
OpenAut	BOOL	`Automatic` switching request to move to the selected position of the valve.	Read
CloseAut	BOOL	`Automatic` switching request to close the valve.	Read
OpenFbkCalc	BOOL	Source of the feedback signal Limit switch Valve open: 1: Calculated 0: Sensor	Read
OpenFbk	BOOL	Feedback signal Limit switch Valve open	Read
CloseFbkCalc	BOOL	Source of the feedback signal Limit switch Valve closed: 1: Calculated 0: Sensor	Read
CloseFbk	BOOL	Feedback signal Limit switch Valve closed.	Read
Perm* Int* Prot*		See Locking [▶ 20]	
SafePosEn	BOOL	Use the safe position of the valve: 1: Use safe position. 0: Do not use safe position.	Read
SafePos	BOOL	Safe position of the valve: 1: Open 0: Closed	Read
ResetAut	BOOL	`Automatic` reset switching request: 1: Reset requested. 0: No reset requested.	Read

Outputs

Name	Type	Description	OPC UA access
State*		See Operation Mode [▶ 13]	
Ctrl	BOOL	Valve switching command: 1: Open 0: Close	Read
SafePosAct	BOOL	Safe position: 1: Move to a safe position. 0: Normal operation	Read

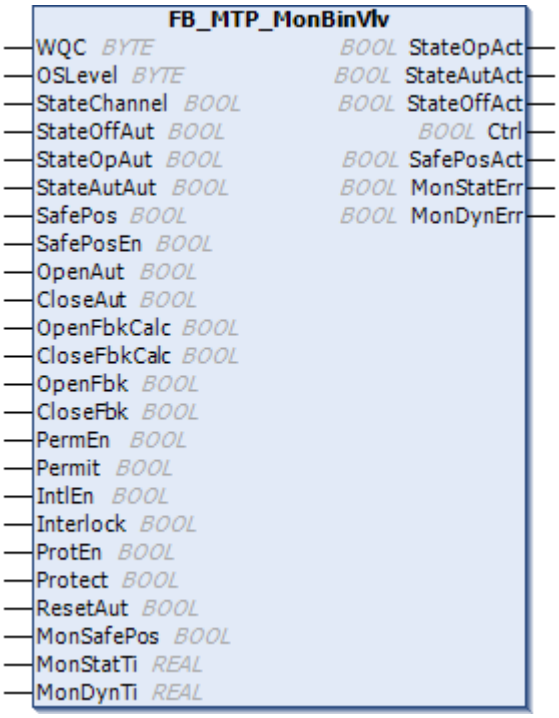
External variables

Name	Type	Description	OPC UA access
State*		See Operation Mode [▶ 13]	
OpenOp	BOOL	`Operator` switching request to open the valve.	Read/write
CloseOp	BOOL	`Operator` switching request to close the valve.	Read/write
ResetOp	BOOL	`Operator` reset switching request: 0→1: `Operator` request 1→0: Request has been processed.	Read/write

Methods

Name	Description
SetOffline SetOperator SetAutomatic	See Operation Mode [▶ 13]

5.3.1.6 FB_MTP_MonBinVlv



The function block `FB_MTP_MonBinVlv` is an object for controlling a binary valve from different sources: internal PLC logic or manual operation (e. g. via OPC UA). Switching requests are managed via the state machine Operation Mode [▶ 13]. It also includes the option of monitoring the output value with the respective feedback. The OPC UA access rights are described in the variable tables.

Controlling the valve

The specification for opening and closing the valve is made via the switching requests `Open*` and `Close*`. The state machine of the Operation Mode [▶ 13] manages whether and from which source new switching requests are processed. For simultaneous requests from `Open*` and `Close*`, `Close*` has the highest priority. If Locking [▶ 20] is active, control of the outputs via the switching requests is prevented or the safety position is assumed.

Safety position

The safety position is defined via the variables `SafePosEn` and `SafePos` .

The variable `SafePosEn` is used to describe whether the valve has a fixed safety position (`SafePosEn = TRUE`) or whether the current position should be maintained (`SafePosEn = FALSE`).

The safety position is described with `SafePos`:

`SafePos = FALSE`: Safety position of the valve: closed

`SafePos = TRUE`: Safety position of the valve: open

Monitoring

The control and the respective Feedback [▶ 20] can be monitored with the Feedback monitoring [▶ 21].

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

OSLevel [▶ 18]

Locking [▶ 20]

Reset [▶ 23]

Inheritance hierarchy

FB_MTP_BinVlv

  FB_MTP_MonBinVlv

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
State*		See Operation Mode [▶ 13]	
OpenAut	BOOL	`Automatic` switching request to move to the selected position of the valve.	Read
CloseAut	BOOL	`Automatic` switching request to close the valve.	Read
OpenFbkCalc	BOOL	Source of the feedback signal Limit switch Valve open: 1: Calculated 0: Sensor	Read
OpenFbk	BOOL	Feedback signal Limit switch Valve open	Read
CloseFbkCalc	BOOL	Source of the feedback signal Limit switch Valve closed: 1: Calculated 0: Sensor	Read

Name	Type	Description	OPC UA access
CloseFbk	BOOL	Feedback signal Limit switch Valve closed.	Read
Perm* Int* Prot*		See Locking [▶ 20]	
SafePosEn	BOOL	Use the safe position of the valve: 1: Use safe position. 0: Do not use safe position.	Read
SafePos	BOOL	Safe position of the valve: 1: Open 0: Closed	Read
ResetAut	BOOL	`Automatic` reset switching request: 1: Reset requested. 0: No reset requested.	Read
MonStatTi	REAL	Monitoring time for static errors [s]	Read
MonDynTi	REAL	Monitoring time for dynamic errors [s]	Read
MonSafePos	BOOL	Behavior of the interface after an error occurs: 1: Safe position should be approached. 0: The current state is retained.	Read

Outputs

Name	Type	Description	OPC UA access
State*		See Operation Mode [▶ 13]	
Ctrl	BOOL	Valve switching command: 1: Open 0: Close	Read
SafePosAct	BOOL	Safe position: 1: Move to a safe position. 0: Normal operation	Read

External variables

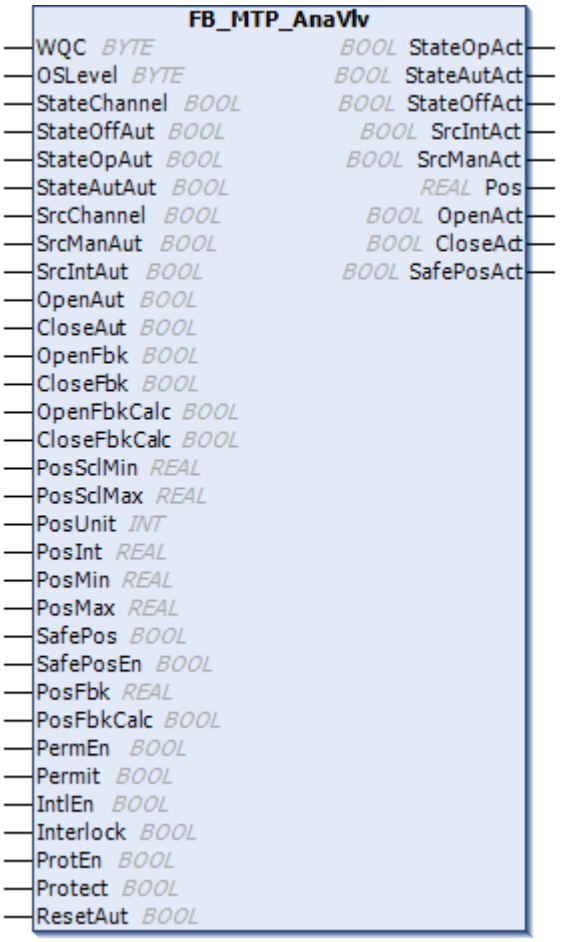
Name	Type	Description	OPC UA access
State*		See Operation Mode [▶ 13]	
OpenOp	BOOL	`Operator` switching request to open the valve.	Read/write
CloseOp	BOOL	`Operator` switching request to close the valve.	Read/write
ResetOp	BOOL	`Operator` reset switching request: 0→1: `Operator` request 1→0: Request has been processed.	Read/write
MonEn	BOOL	Enable `FeedbackMonitoring`: 1: `FeeedbackMonitoring` active 0: `FeedbackMonitoring` inactive	Read/write

Methods

Name	Description
SetOffline	See Operation Mode [▶ 13]

Name	Description
SetOperator SetAutomatic	

5.3.1.7 FB_MTP_AnaVlv



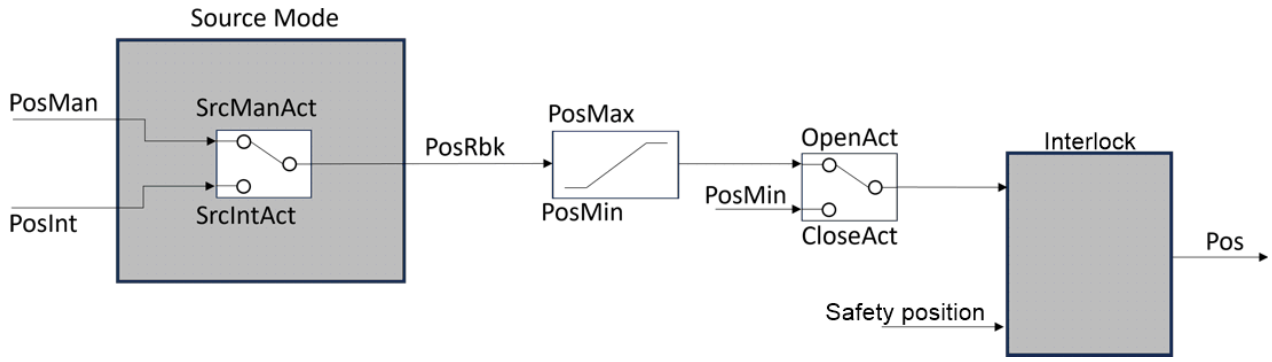
The function block `FB_MTP_AnaVlv` serves as an interface for controlling an analog valve from different sources: internal PLC logic or manual operation (e. g. via OPC UA). Switching requests and value specifications are managed independently of each other via the state machines of Operation Mode [▶ 13] and Source Mode [▶ 15]. The OPC UA access rights are described in the variable tables.

Controlling the valve

The specification for opening and closing the valve is made via the switching requests `Open*` and `Close*`. The state machine of the Operation Mode [▶ 13] manages whether and from which source new switching requests are processed. For simultaneous requests from `Open*` and `Close*`, `Close*` has the highest priority. If Locking [▶ 20] is active, control of the outputs via the switching requests is prevented or the safety position is assumed.

Setpoint specification

The position is specified via `Pos*` variables. The position specification is managed by the state machine of the Source Mode [▶ 15] and output according to the diagram below:



Safety position

The safety position is defined via the variables `SafePosEn` and `SafePos` .

The variable `SafePosEn` is used to describe whether the valve has a fixed safety position (`SafePosEn = TRUE`) or whether the current position should be maintained (`SafePosEn = FALSE`)

The safety position is described with `SafePos`:

`SafePos = FALSE`: Safety position of the valve: `PosMin`

`SafePos = TRUE`: Safety position of the valve: `PosMax`

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

OSLevel [▶ 18]

Operation Mode [▶ 13]

Source Mode [▶ 15]

Feedback [▶ 20]

Unit [▶ 19]

Value scaling [▶ 19]

Value limitation [▶ 20]

Locking [▶ 20]

Reset [▶ 23]

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
State*		See Operation Mode [▶ 13]	

Name	Type	Description	OPC UA access
OpenAut	BOOL	`Automatic` switching request to move to the selected position of the valve.	Read
CloseAut	BOOL	`Automatic` switching request to close the valve.	Read
OpenFbkCalc	BOOL	Source of the feedback signal Limit switch Valve open: 1: Calculated 0: Sensor	Read
OpenFbk	BOOL	Feedback signal Limit switch Valve open	Read
CloseFbkCalc	BOOL	Source of the feedback signal Limit switch Valve closed: 1: Calculated 0: Sensor	Read
CloseFbk	BOOL	Feedback signal Limit switch Valve closed.	Read
Src*		See Source Mode [▶ 15]	
PosInt	REAL	`Internal` value specification for the position	Read
PosUnit	INT	Unit of the position	Read
PosMin	REAL	Lower limit of the value specification for the valve position	Read
PosMax	REAL	Upper limit of the value specification for the valve position	Read
PosFbkCalc	BOOL	Source of the position feedback value: 1: Calculated 0: Sensor	Read
PosFbk	REAL	Feedback value of the position	Read
PosSclMin	REAL	Scale start of the position display	Read
PosSclMax	REAL	Scale end of the position display	Read
Perm* Int* Prot*		See Locking [▶ 20]	
SafePosEn	BOOL	Use the safe position of the valve: 1: Use safe position. 0: Do not use safe position (Hold position).	Read
SafePos	BOOL	Safe position of the valve: 1: `PosMax` 0: `PosMin`	Read
ResetAut	BOOL	`Automatic` reset switching request: 1: Reset requested. 0: No reset requested.	Read

Outputs

Name	Type	Description	OPC UA access
State*		See Operation Mode [▶ 13]	
OpenAct	BOOL	Open valve switching command.	Read
CloseAd	BOOL	Close valve switching command.	Read
Src*		See Source Mode [▶ 15]	
Pos	REAL	Position setpoint on valve	Read
SafePosAct	BOOL	Safe position: 1: Move to a safe position 0: Normal operation	Read

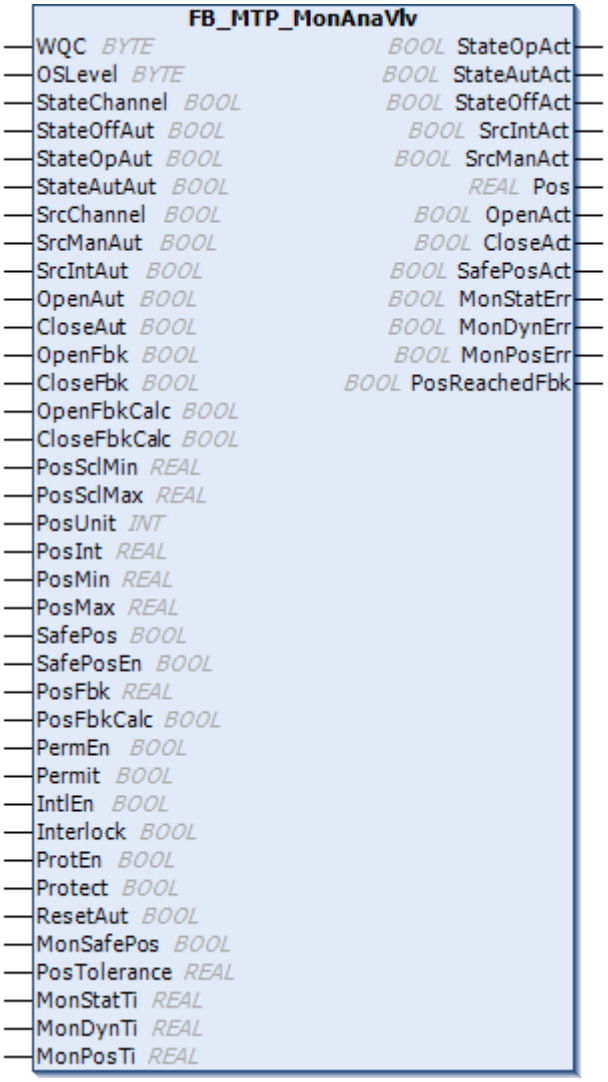
External variables

Name	Type	Description	OPC UA access
State*		See Operation Mode [▶ 13]	
OpenOp	BOOL	`Operator` switching request to move to the selected position of the valve.	Read/write
CloseOp	BOOL	`Operator` switching request to close the valve.	Read/write
Src*		See Source Mode [▶ 15]	
PosMan	BOOL	`Manual` value specification for the position	Read/write
PosRbk	REAL	Unprocessed value of the `Operator` Value specification for the position	Read/write
ResetOp	BOOL	`Operator` reset switching request: 0→1: `Operator` request 1→0: Request has been processed.	Read/write

Methods

Name	Description
SetOffline SetOperator SetAutomatic	See Operation Mode [▶ 13]
SetInternal SetManual	See Source Mode [▶ 15]

5.3.1.8 FB_MTP_MonAnaVlv



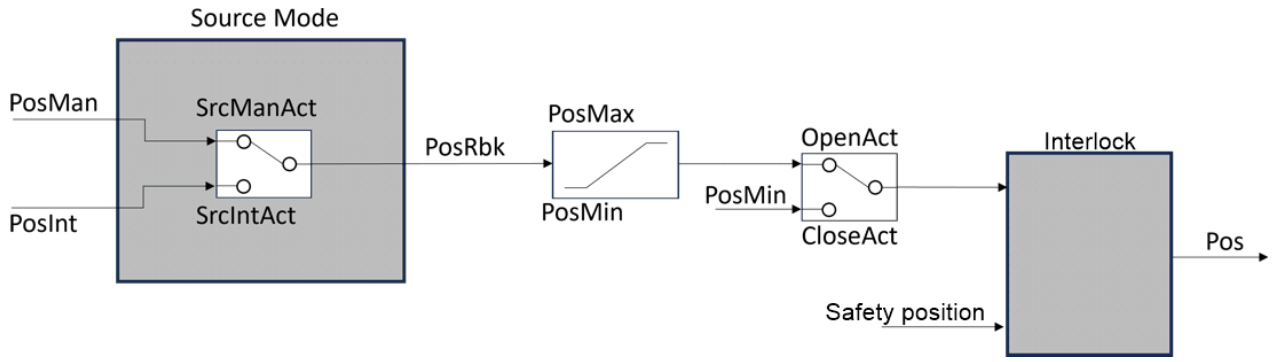
The function block `FB_MTP_MonAnaVlv` is an object for controlling an analog valve from different sources: internal PLC logic or manual operation (access via OPC UA). Switching requests and value specifications are managed independently of each other via the state machines Operation Mode [▶ 13] and Source Mode [▶ 15]. It also includes the option of monitoring the output values with the respective feedback signals. The OPC UA access rights are described in the variable tables.

Controlling the valve

The specification for opening and closing the valve is made via the switching requests `Open*` and `Close*`. The state machine of the Operation Mode [▶ 13] manages whether and from which source new switching requests are processed. For simultaneous requests from `Open*` and `Close*`, `Close*` has the highest priority. If Locking [▶ 20] is active, control of the outputs via the switching requests is prevented or the safety position is assumed.

Setpoint specification

The position is specified via `Pos*` variables. The position specification is managed by the state machine of the Source Mode [▶ 15] and output according to the diagram below:



Safety position

The safety position is defined via the variables `SafePosEn` and `SafePos` .

The variable `SafePosEn` is used to describe whether the valve has a fixed safety position (`SafePosEn = TRUE`) or whether the current position should be maintained (`SafePosEn = FALSE`)

The safety position is described with `SafePos`:

`SafePos = FALSE`: Safety position of the valve: `PosMin`

`SafePos = TRUE`: Safety position of the valve: `PosMax`

Monitoring

The control and the respective Feedback [▶ 20] can be monitored via Feedback monitoring [▶ 21].

The interface also has a position monitor, which extends the Feedback monitoring [▶ 21] with two additional outputs:

The output `PosReachedFbk` indicates whether the set position has been reached:

```
PosReachedFbk = (PosFbk - Pos) <= PosTolerance
```

The time to reach the position is specified via the input `MonPosTi`. If the time is exceeded, the output `MonPosErr` is set to `TRUE`.

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

OSLevel [▶ 18]

Value scaling [▶ 19]

Value limitation [▶ 20]

Locking [▶ 20]

Reset [▶ 23]

Inheritance hierarchy

FB_MTP_AnaVlv

  FB_MTP_MonAnaVlv

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
State*		See Operation Mode [▶ 13]	
OpenAut	BOOL	`Automatic` switching request to move to the selected position of the valve.	Read
CloseAut	BOOL	`Automatic` switching request to close the valve.	Read
OpenFbkCalc	BOOL	Source of the feedback signal Limit switch Valve open: 1: Calculated 0: Sensor	Read
OpenFbk	BOOL	Feedback signal Limit switch Valve open	Read
CloseFbkCalc	BOOL	Source of the feedback signal Limit switch Valve closed: 1: Calculated 0: Sensor	Read
CloseFbk	BOOL	Feedback signal Limit switch Valve closed.	Read
Src*		See Source Mode [▶ 15]	
PosInt	REAL	`Internal` value specification for the position	Read
PosUnit	INT	Unit of the position	Read
PosMin	REAL	Lower limit of the value specification for the valve position	Read
PosMax	REAL	Upper limit of the value specification for the valve position	Read
PosFbkCalc	BOOL	Source of the position feedback value: 1: Calculated 0: Sensor	Read
PosFbk	REAL	Feedback value of the position	Read
PosSclMin	REAL	Scale start of the position display	Read
PosSclMax	REAL	Scale end of the position display	Read
Perm* Int* Prot*		See Locking [▶ 20]	
SafePosEn	BOOL	Use the safe position of the valve: 1: Use safe position. 0: Do not use safe position.	Read
SafePos	BOOL	Safe position of the valve: 1: Open 0: Closed	Read
ResetAut	BOOL	`Automatic` reset switching request: 1: Reset requested. 0: No reset requested.	Read

Outputs

Name	Type	Description	OPC UA access
State*		See Operation Mode [▶ 13]	
OpenAct	BOOL	Open valve switching command.	Read
CloseAd	BOOL	Close valve switching command.	Read
Src*		See Source Mode [▶ 15]	
Pos	REAL	Position setpoint on valve	Read
SafePosAct	BOOL	Safe position: 1: Move to a safe position 0: Normal operation	Read

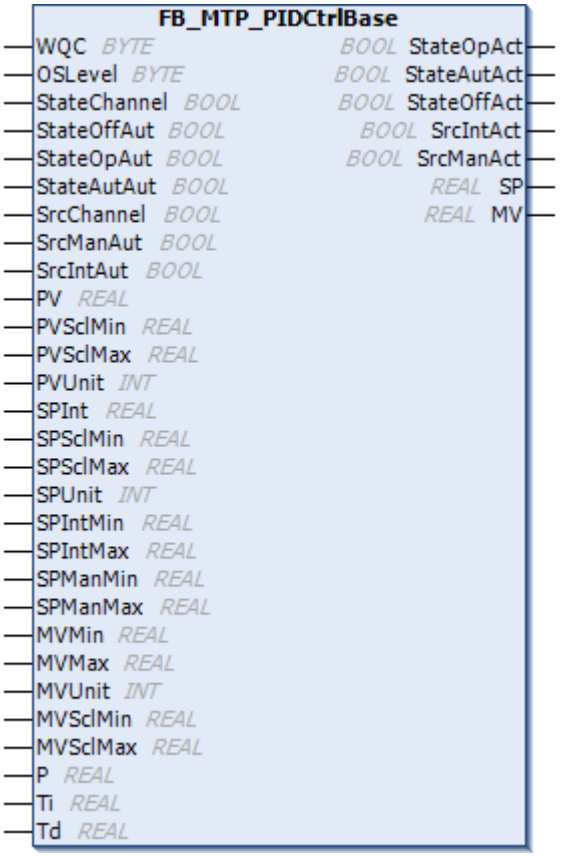
External variables

Name	Type	Description	OPC UA access
State*		See Operation Mode [▶ 13]	
OpenOp	BOOL	`Operator` switching request to move to the selected position of the valve.	Read/write
CloseOp	BOOL	`Operator` switching request to close the valve.	Read/write
Src*		See Source Mode [▶ 15]	
PosMan	BOOL	`Manual` value specification for the position	Read/write
PosRbk	REAL	Unprocessed value of the `Operator` Value specification for the position	Read/write
ResetOp	BOOL	`Operator` reset switching request: 0→1: `Operator` request 1→0: Request has been processed.	Read/write
MonEn	BOOL	Enable `FeedbackMonitoring`: 1: `FeeedbackMonitoring` active 0: `FeedbackMonitoring` inactive	Read/write

Methods

Name	Description
SetOffline SetOperator SetAutomatic	See Operation Mode [▶ 13]
SetInternal SetManual	See Source Mode [▶ 15]

5.3.1.9 FB_MTP_PIDCtrlBase



The function block `FB_MTP_PIDCtrlBase` serves as an interface for using a PID controller from different sources: internal PLC logic or manual operation (e. g. via OPC UA). Value specifications are managed independently of each other by the state machines of Operation Mode [▶ 13] and Source Mode [▶ 15]. The OPC UA access rights are described in the variable tables.

The `FB_MTP_PIDCtrlBase` is an abstract function block. This cannot be instantiated, but must be derived (see syntax)!

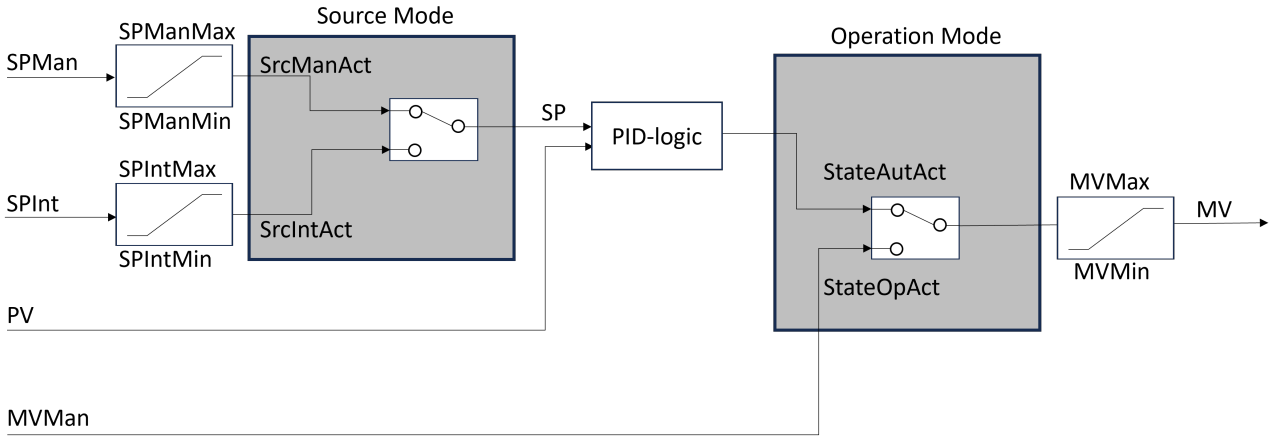
State machines

The source of the setpoint `SP*` is managed via the state machine of the Source Mode [▶ 15]. The source of the output manipulated variable `MV*` is managed via the state machine of the Operation Mode [▶ 13]: Output value of the PID logic or specification by manual operation `MVMan`.

PID controller

The PID logic (`PIDLogic()`) can be freely selected. The PID logic is called via the method `PIDLogic()`. The parameters for the controller are described via the inputs `P`, `Ti` and `Td`. The output `MV` describes the current manipulated variable depending on the current states of the state machines. The output `SP` shows the currently used setpoint of the PID logic.

The function block works according to the following scheme:



Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

OSLevel [▶ 18]

Value scaling [▶ 19]

Value limitation [▶ 20]

Unit [▶ 19]

Syntax

```
FUNCTION_BLOCK FB_MTP_PIDCtrl EXTENDS FB_MTP_PIDCtrlBase
VAR
    ///Controller Toolbox (delete if you want to use own PID Logic)
    CTRL_PID: FB_CTRL_PID;
    PIDParams: ST_Ctrl_PID_PARAMS;
END_VAR
```

```
SUPER^();
```

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
State*		See Operation Mode [▶ 13]	
MVUnit	INT	Unit of manipulated variable	Read
MVSclMin	REAL	Minimum value of the manipulated variable display	Read
MVSclMax	REAL	Maximum value of the manipulated variable display	Read
MVMin	REAL	Minimum value of the manipulated variable	Read
MVMax	REAL	Maximum value of the manipulated variable	Read
Src*		See Source Mode [▶ 15]	
SPInt	REAL	`internal` setpoint specification	Read
SPUnit	INT	Unit of the setpoint	Read
SPIntMin	REAL	Lower limit value of the `internal` setpoint specification	Read

Name	Type	Description	OPC UA access
SPIntMax	REAL	Upper limit value of the PEA internal setpoint specification	Read
SPManMin	REAL	Lower limit value of the manual setpoint specification	Read
SPManMax	REAL	Upper limit value of the manual setpoint specification	Read
SPSclMin	REAL	Minimum value for the display of the setpoint specification	Read
SPSclMax	REAL	Maximum value for the display of the setpoint specification	Read
PV	REAL	Process value	Read
PVUnit	INT	Unit of the process value	Read
PVSclMin	REAL	Minimum value for displaying the process value	Read
PVSclMax	REAL	Maximum value for displaying the process value	Read
P	REAL	Proportional gain of the controller	Read
Ti	REAL	Integral action time of the controller [s]	Read
Td	REAL	Derivative action time of the controller [s]	Read

Outputs

Name	Type	Description	OPC UA access
State*		See Operation Mode [▶ 13]	
MV	REAL	Current manipulated variable of the controller	
Src*		See Source Mode [▶ 15]	
SP	REAL	Setpoint currently used by the controller.	

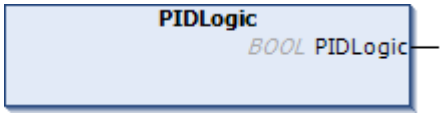
External variables

Name	Type	Description	OPC UA access
State*		See Operation Mode [▶ 13]	
MVMan	REAL	`Manual` specification of the manipulated variable	Read/write
Src*		See Source Mode [▶ 15]	
SPMan	REAL	`Manual` setpoint specification	Read/write

Methods

Name	Description
PIDLogic	Implementation of the logic for calling the PID controller
ResetPID	Resetting the PID controller

5.3.1.9.1 PIDLogic



This method enables the implementation of the logic of the PID controller.

Syntax

```
METHOD PIDLogic : BOOL
```

Call Beckhoff PID controller (`Tc2_ControllerToolbox` is required)

```
//Set PID Params
PIDParams.tCtrlCycleTime   := LREAL_TO_TIME(fCycleTime * 1000);
PIDParams.tTaskCycleTime   := LREAL_TO_TIME(fCycleTime * 1000);
```

```
PIDParams.fKp               := P;
PIDParams.tTn               := REAL_TO_TIME(TI * 1000);
PIDParams.tTv               := REAL_TO_TIME(TD * 1000);
PIDParams.fOutMaxLimit      := MVMax;
PIDParams.fOutMinLimit      := MVMin;
PIDParams.bARWOnIPartOnly   := TRUE;

//Call Controller Toolbox PID function block
CTRL_PID(
   fSetpointValue := SP,
   fActualValue   := PV,
   eMode          := eCTRL_MODE_ACTIVE,
   stParams       := PIDParams);

//Write resulting Manipulated Value to MTP function block
MV:= LREAL_TO_REAL(CTRL_PID.fOut);
```

Return value

Name	Type	Description
PIDLogic	BOOL	

5.3.1.9.2 ResetPID

This method allows the implementation of a logic for resetting the PID logic.

Syntax

```
METHOD ResetPID : BOOL
```

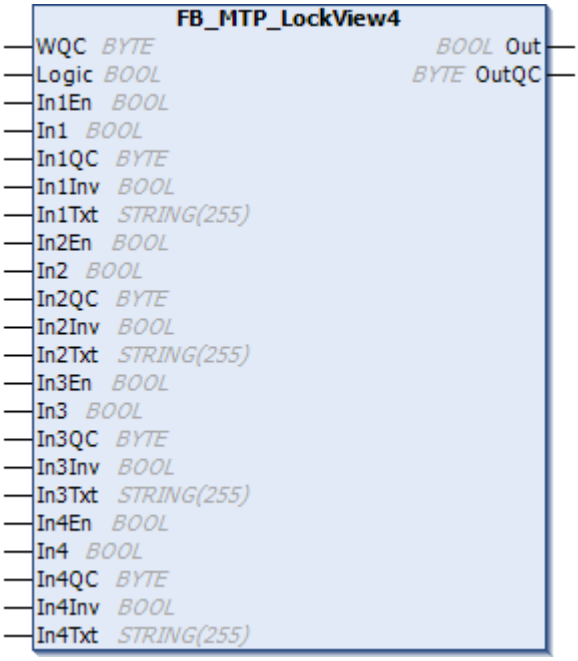
```
CTRL_PID(
   eMode := eCTRL_MODE_Reset,
   stParams:= PIDParams
);

//Write resulting Manipulated Value to MTP function block
MV:= LREAL_TO_REAL(CTRL_PID.fOut);
```

5.3.2 DiagnosticElements

Locking views visualize the current state of a basic logical operation `AND/OR`.

5.3.2.1 FB_MTP_LockView4



The function block `FB_MTP_LockView4` is an object for visualizing a basic logical operation `AND/OR` with four inputs. This is made available via OPC UA.

Each input can be enabled `*En`, inverted `*Inv` and described with a text `*Txt`. Each input also contains a quality code `*QC`. It describes the conditions under which the value of the interface was recorded and how trustworthy this value is.

The inputs can be evaluated as `AND`-logic (`Logic = TRUE`) or as `OR`-logic (`Logic = FALSE`). The linking result from the inputs is output at `Out` . Only enabled inputs are taken into account for this. The result of the `QC` inputs is output at `OutQC`.

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

Inputs

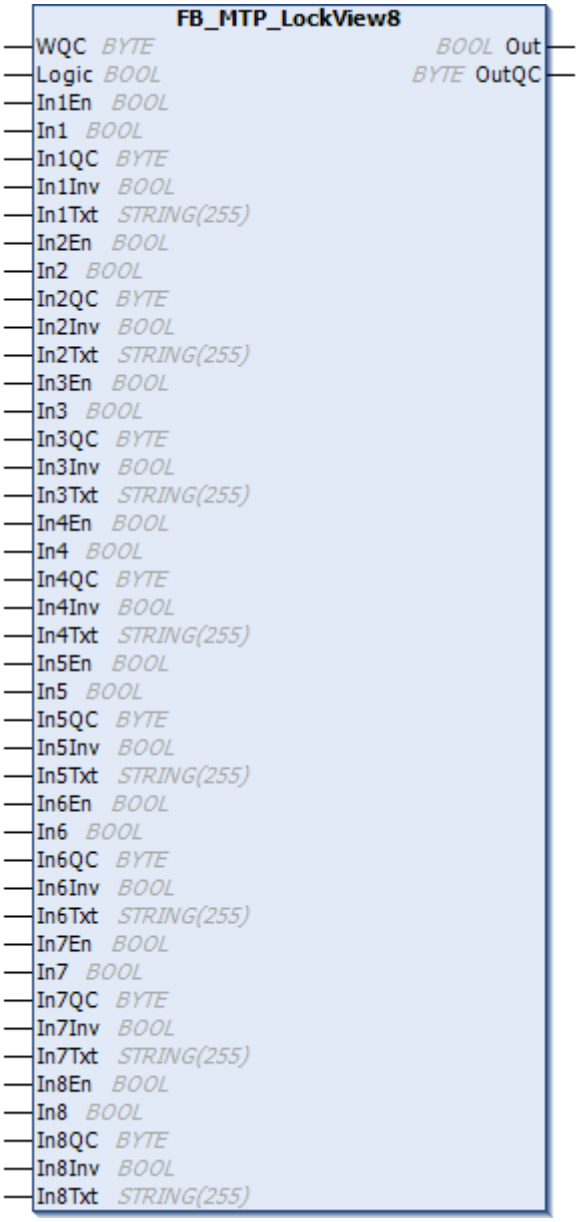
Name	Type	Description	OPC UA access
WQC	BYTE	Worst Quality Code	Read
Logic	BOOL	Logical operation: 0: `OR` 1: `AND`	Read
In1En	BOOL	Enable input 1: 0: Input not enabled. 1: Input enabled	Read
In1	BOOL	Input 1 - value	Read
In1QC	BYTE	Input 1 – `Qualitiy Code`	Read
In1Inv	BOOL	Invert input 1: 0: Not inverted 1: Inverted	Read

Name	Type	Description	OPC UA access
In1Txt	STRING	Input 1 - description text	Read
In2En	BOOL	Enable input 2: 0: Input not enabled. 1: Input enabled.	Read
In2	BOOL	Input 2 - value	Read
In2QC	BYTE	Input 2 – `Qualitiy Code`	Read
In2Inv	BOOL	Invert input 2: 0: Not inverted 1: Inverted	Read
In2Txt	STRING	Input 2 - description text	Read
In3En	BOOL	Enable input 3: 0: Input not enabled. 1: Input enabled.	Read
In3	BOOL	Input 3 - value	Read
In3QC	BYTE	Input 3 – `Qualitiy Code`	Read
In3Inv	BOOL	Invert input 3: 0: Not inverted 1: Inverted	Read
In3Txt	STRING	Input 3 - description text	Read
In4En	BOOL	Enable input 4: 0: Input not enabled. 1: Input enabled.	Read
In4	BOOL	Input 4 - value	Read
In4QC	BYTE	Input 4 – `Qualitiy Code`	Read
In4Inv	BOOL	Invert input 4: 0: Not inverted 1: Inverted	Read
In4Txt	STRING	Input 4 - description text	Read

Outputs

Name	Type	Description	OPC UA access
Out	BOOL	Output of the link result	Read
OutQC	BOOL	QC of the link result	Read

5.3.2.2 FB_MTP_LockView8



The function block `FB_MTP_LockView8` is an object for visualizing a basic logical operation `AND/OR` with eight inputs. This is made available via OPC UA.

Each input can be enabled `*En`, inverted `*Inv` and described with a text `*Txt`. Each input also contains a quality code `*QC`. It describes the conditions under which the value of the interface was recorded and how trustworthy this value is.

The inputs can be evaluated as `AND`-logic (`Logic = TRUE`) or as `OR`-logic (`Logic = FALSE`). The linking result from the inputs is output at `Out` . Only enabled inputs are taken into account for this. The result of the `QC` inputs is output at `OutQC`.

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

Inputs

Name	Type	Description	OPC UA access
WQC	BYTE	Worst Quality Code	Read
Logic	BOOL	Logical operation: 0: OR 1: AND	Read
In1En	BOOL	Enable input 1: 0: Input not enabled. 1: Input enabled	Read
In1	BOOL	Input 1 - value	Read
In1QC	BYTE	Input 1 – Qualitiy Code	Read
In1Inv	BOOL	Invert input 1: 0: Not inverted 1: Inverted	Read
In1Txt	STRING	Input 1 - description text	Read
In2En	BOOL	Enable input 2: 0: Input not enabled. 1: Input enabled.	Read
In2	BOOL	Input 2 - value	Read
In2QC	BYTE	Input 2 – Qualitiy Code	Read
In2Inv	BOOL	Invert input 2: 0: Not inverted 1: Inverted	Read
In2Txt	STRING	Input 2 - description text	Read
In3En	BOOL	Enable input 3: 0: Input not enabled. 1: Input enabled.	Read
In3	BOOL	Input 3 - value	Read
In3QC	BYTE	Input 3 – Qualitiy Code	Read
In3Inv	BOOL	Invert input 3: 0: Not inverted 1: Inverted	Read
In3Txt	STRING	Input 3 - description text	Read
In4En	BOOL	Enable input 4: 0: Input not enabled. 1: Input enabled.	Read
In4	BOOL	Input 4 - value	Read
In4QC	BYTE	Input 4 – Qualitiy Code	Read
In4Inv	BOOL	Invert input 4: 0: Not inverted 1: Inverted	Read
In4Txt	STRING	Input 4 - description text	Read
In5En	BOOL	Enable input 5: 0: Input not enabled. 1: Input enabled.	Read
In5	BOOL	Input 5 - value	Read
In5QC	BYTE	Input 5 – Qualitiy Code	Read

Name	Type	Description	OPC UA access
In5Inv	BOOL	Invert input 5: 0: Not inverted 1: Inverted	Read
In5Txt	STRING	Input 5 - description text	Read
In6En	BOOL	Enable input 6: 0: Input not enabled. 1: Input enabled.	Read
In6	BOOL	Input 6 - value	Read
In6QC	BYTE	Input 6 – `Qualitiy Code`	Read
In6Inv	BOOL	Invert input 6: 0: Not inverted 1: Inverted	Read
In6Txt	STRING	Input 6 - description text	Read
In7En	BOOL	Enable input 7: 0: Input not enabled. 1: Input enabled.	Read
In7	BOOL	Input 7 - value	Read
In7QC	BYTE	Input 7 – `Qualitiy Code`	Read
In7Inv	BOOL	Invert input 7: 0: Not inverted 1: Inverted	Read
In7Txt	STRING	Input 7 - description text	Read
In8En	BOOL	Enable input 8: 0: Input not enabled. 1: Input enabled.	Read
In8	BOOL	Input 8 - value	Read
In8QC	BYTE	Input 8 – `Qualitiy Code`	Read
In8Inv	BOOL	Invert input 8: 0: Not inverted 1: Inverted	Read
In8Txt	STRING	Input 8 - description text	Read

Outputs

Name	Type	Description	OPC UA access
Out	BOOL	Output of the link result	Read
OutQC	BOOL	QC of the link result	Read

5.3.2.3 FB_MTP_LockView16

The function block `FB_MTP_LockView16` is an object for visualizing a basic logical operation `AND/OR` with sixteen inputs. This is made available via OPC UA.

Each input can be enabled `*En`, inverted `*Inv` and described with a text `*Txt`. Each input also contains a quality code `*QC`. It describes the conditions under which the value of the interface was recorded and how trustworthy this value is.

The inputs can be evaluated as `AND`-logic (`Logic = TRUE`) or as `OR`-logic (`Logic = FALSE`). The linking result from the inputs is output at `Out` . Only enabled inputs are taken into account for this. The result of the `QC` inputs is output at `OutQC`.

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

Inputs

Name	Type	Description	OPC UA access
WQC	BYTE	Worst Quality Code	Read
Logic	BOOL	Logical operation: 0: `OR` 1: `AND`	Read
In1En	BOOL	Enable input 1: 0: Input not enabled. 1: Input enabled	Read
In1	BOOL	Input 1 - value	Read
In1QC	BYTE	Input 1 – `Qualitiy Code`	Read
In1Inv	BOOL	Invert input 1: 0: Not inverted 1: Inverted	Read
In1Txt	STRING	Input 1 - description text	Read
In2En	BOOL	Enable input 2: 0: Input not enabled. 1: Input enabled.	Read
In2	BOOL	Input 2 - value	Read
In2QC	BYTE	Input 2 – `Qualitiy Code`	Read
In2Inv	BOOL	Invert input 2: 0: Not inverted 1: Inverted	Read
In2Txt	STRING	Input 2 - description text	Read
In3En	BOOL	Enable input 3: 0: Input not enabled. 1: Input enabled.	Read
In3	BOOL	Input 3 - value	Read
In3QC	BYTE	Input 3 – `Qualitiy Code`	Read
In3Inv	BOOL	Invert input 3: 0: Not inverted 1: Inverted	Read
In3Txt	STRING	Input 3 - description text	Read
In4En	BOOL	Enable input 4: 0: Input not enabled. 1: Input enabled.	Read
In4	BOOL	Input 4 - value	Read
In4QC	BYTE	Input 4 – `Qualitiy Code`	Read
In4Inv	BOOL	Invert input 4: 0: Not inverted 1: Inverted	Read

Name	Type	Description	OPC UA access
In4Txt	STRING	Input 4 - description text	Read
In5En	BOOL	Enable input 5: 0: Input not enabled. 1: Input enabled.	Read
In5	BOOL	Input 5 - value	Read
In5QC	BYTE	Input 5 – `Qualitiy Code`	Read
In5Inv	BOOL	Invert input 5: 0: Not inverted 1: Inverted	Read
In5Txt	STRING	Input 5 - description text	Read
In6En	BOOL	Enable input 6: 0: Input not enabled. 1: Input enabled.	Read
In6	BOOL	Input 6 - value	Read
In6QC	BYTE	Input 6 – `Qualitiy Code`	Read
In6Inv	BOOL	Invert input 6: 0: Not inverted 1: Inverted	Read
In6Txt	STRING	Input 6 - description text	Read
In7En	BOOL	Enable input 7: 0: Input not enabled. 1: Input enabled.	Read
In7	BOOL	Input 7 - value	Read
In7QC	BYTE	Input 7 – `Qualitiy Code`	Read
In7Inv	BOOL	Invert input 7: 0: Not inverted 1: Inverted	Read
In7Txt	STRING	Input 7 - description text	Read
In8En	BOOL	Enable input 8: 0: Input not enabled. 1: Input enabled.	Read
In8	BOOL	Input 8 - value	Read
In8QC	BYTE	Input 8 – `Qualitiy Code`	Read
In8Inv	BOOL	Invert input 8: 0: Not inverted 1: Inverted	Read
In8Txt	STRING	Input 8 - description text	Read
In9En	BOOL	Enable input 9: 0: Input not enabled. 1: Input enabled.	Read
In9	BOOL	Input 9 - value	Read
In9QC	BYTE	Input 9 – `Qualitiy Code`	Read
In9Inv	BOOL	Invert input 9: 0: Not inverted 1: Inverted	Read
In9Txt	STRING	Input 9 - description text	Read
In10En	BOOL	Enable input 10:	Read

Name	Type	Description	OPC UA access
		0: Input not enabled. 1: Input enabled.	
In10	BOOL	Input 10 - value	Read
In10QC	BYTE	Input 10 – `Qualitiy Code`	Read
In10Inv	BOOL	Invert input 10: 0: Not inverted 1: Inverted	Read
In10Txt	STRING	Input 10 - description text	Read
In11En	BOOL	Enable input 11: 0: Input not enabled. 1: Input enabled.	Read
In11	BOOL	Input 11 - value	Read
In11QC	BYTE	Input 11 – `Qualitiy Code`	Read
In11Inv	BOOL	Invert input 11: 0: Not inverted 1: Inverted	Read
In11Txt	STRING	Input 11 - description text	Read
In12En	BOOL	Enable input 12: 0: Input not enabled. 1: Input enabled.	Read
In12	BOOL	Input 12 - value	Read
In12QC	BYTE	Input 12 – `Qualitiy Code`	Read
In12Inv	BOOL	Invert input 12: 0: Not inverted 1: Inverted	Read
In12Txt	STRING	Input 12 - description text	Read
In13En	BOOL	Enable input 13: 0: Input not enabled. 1: Input enabled.	Read
In13	BOOL	Input 13 - value	Read
In13QC	BYTE	Input 13 – `Qualitiy Code`	Read
In13Inv	BOOL	Invert input 13: 0: Not inverted 1: Inverted	Read
In13Txt	STRING	Input 13 - description text	Read
In14En	BOOL	Enable input 14: 0: Input not enabled. 1: Input enabled.	Read
In14	BOOL	Input 14 - value	Read
In14QC	BYTE	Input 14 – `Qualitiy Code`	Read
In14Inv	BOOL	Invert input 14: 0: Not inverted 1: Inverted	Read
In14Txt	STRING	Input 14 - description text	Read
In15En	BOOL	Enable input 15: 0: Input not enabled. 1: Input enabled.	Read

Name	Type	Description	OPC UA access
In15	BOOL	Input 15 - value	Read
In15QC	BYTE	Input 15 – `Qualitiy Code`	Read
In15Inv	BOOL	Invert input 15: 0: Not inverted 1: Inverted	Read
In15Txt	STRING	Input 15 - description text	Read
In16En	BOOL	Enable input 16: 0: Input not enabled. 1: Input enabled.	Read
In16	BOOL	Input 16 - value	Read
In16QC	BYTE	Input 16 – `Qualitiy Code`	Read
In16Inv	BOOL	Invert input 16: 0: Not inverted 1: Inverted	Read
In16Txt	STRING	Input 16 - description text	Read

Outputs

Name	Type	Description	OPC UA access
Out	BOOL	Output of the link result	Read
OutQC	BOOL	QC of the link result	Read

5.3.3 IndicatorElements

5.3.3.1 FB_MTP_AnaView

The function block `FB_MTP_AnaView` provides the analog input value `V` together with scale values and unit via OPC UA.

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

Value scaling [▶ 19]

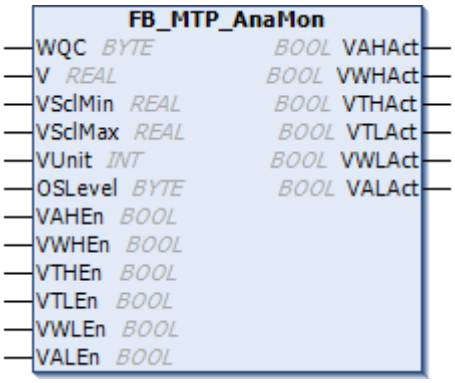
Unit [▶ 19]

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-

Name	Type	Description	OPC UA access
WQC	BYTE	Worst Quality Code	Read
V	REAL	Value	Read
VUnit	INT	Unit of value	Read
VSclMin	REAL	Scale start for value display	Read
VSclMax	REAL	Scale end for value display	Read

5.3.3.2 FB_MTP_AnaMon



The function block `FB_MTP_AnaMon` makes the analog input value `V` available for manual operation (e.g. via OPC UA). This value `V` can be monitored for up to six limits.

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

OSLevel [▶ 18]

Value scaling [▶ 19]

Unit [▶ 19]

Limit value monitoring [▶ 22]

Inheritance hierarchy

FB_MTP_AnaView

FB_MTP_AnaMon

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
V	REAL	Value	Read
VUnit	INT	Unit of value	Read

Name	Type	Description	OPC UA access
VSclMin	REAL	Scale start for value display	Read
VSclMax	REAL	Scale end for value display	Read
VAHEn	BOOL	Monitor limit `Alarm High`: 1: Monitoring active 0: No monitoring	Read
VALEn	BOOL	Monitor limit `Alarm Low`: 1: Monitoring active 0: No monitoring	Read
VWHEn	BOOL	Monitor limit `Warning High`: 1: Monitoring active 0: No monitoring	Read
VWLEn	BOOL	Monitor limit `Warning Low`: 1: Monitoring active 0: No monitoring	Read
VTHEn	BOOL	Monitor limit `Tolerance High`: 1: Monitoring active 0: No monitoring	Read
VTLEn	BOOL	Monitor limit `Tolerance Low`: 1: Monitoring active 0: No monitoring	Read

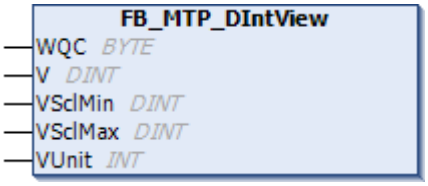
Outputs

Name	Type	Description	OPC UA access
VAHAct	BOOL	`Alarm High` limit exceeded.	Read
VALAct	BOOL	`Alarm Low` limit not reached.	Read
VWHAct	BOOL	`Warning High` limit exceeded.	Read
VWLAct	BOOL	`Warning Low` limit not reached.	Read
VTHAct	BOOL	`Tolerance High` limit exceeded.	Read
VTLAct	BOOL	`Tolerance Low` limit not reached.	Read

External variables

Name	Type	Description	OPC UA access
VAHLim	REAL	`Alarm High` limit	Read/write
VALLim	REAL	`Alarm Low` limit	Read/write
VWHLim	REAL	`Warning High` limit	Read/write
VWLLim	REAL	`Warning Low` limit	Read/write
VTHLim	REAL	`Tolerance High` limit	Read/write
VTLLim	REAL	`Tolerance Low` limit	Read/write

5.3.3.3 FB_MTP_DIntView



The function block FB_MTP_DIntView provides the integer input value V via OPC UA.

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

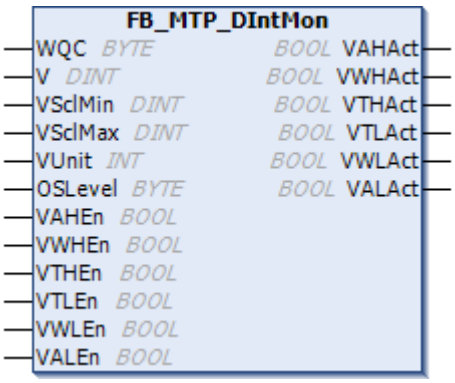
Value scaling [▶ 19]

Unit [▶ 19]

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
V	DINT	Value	Read
VUnit	INT	Unit of value	Read
VSclMin	DINT	Scale start for value display	Read
VSclMax	DINT	Scale end for value display	Read

5.3.3.4 FB_MTP_DIntMon



The function block FB_MTP_DIntMon provides the integer input value V for manual operation (e. g. via OPC UA). This value V can be monitored for up to six limits.

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

OSLevel [▶ 18]

Value scaling [▶ 19]

Limit value monitoring [▶ 22]

Inheritance hierarchy

FB_MTP_DIntView

FB_MTP_DIntMon

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
V	DINT	Value	Read
VUnit	INT	Unit of value	Read
VSclMin	DINT	Scale start for value display	Read
VSclMax	DINT	Scale end for value display	Read
VAHEn	BOOL	Monitor limit `Alarm High`: 1: Monitoring active 0: No monitoring	Read
VALEn	BOOL	Monitor limit `Alarm Low`: 1: Monitoring active 0: No monitoring	Read
VWHEn	BOOL	Monitor limit `Warning High`: 1: Monitoring active 0: No monitoring	Read
VWLEn	BOOL	Monitor limit `Warning Low`: 1: Monitoring active 0: No monitoring	Read
VTHEn	BOOL	Monitor limit `Tolerance High`: 1: Monitoring active 0: No monitoring	Read
VTLEn	BOOL	Monitor limit `Tolerance Low`: 1: Monitoring active 0: No monitoring	Read

Outputs

Name	Type	Description	OPC UA access
VAHAct	BOOL	`Alarm High` limit exceeded.	Read
VALAct	BOOL	`Alarm Low` limit not reached.	Read
VWHAct	BOOL	`Warning High` limit exceeded.	Read
VWLAct	BOOL	`Warning Low` limit not reached.	Read
VTHAct	BOOL	`Tolerance High` limit exceeded.	Read
VTLAct	BOOL	`Tolerance Low` limit not reached.	Read

External variables

Name	Type	Description	OPC UA access
VAHLim	DINT	`Alarm High` limit	Read/write
VALLim	DINT	`Alarm Low` limit	Read/write
VWHLim	DINT	`Warning High` limit	Read/write

Name	Type	Description	OPC UA access
VWLLim	DINT	`Warning Low` limit	Read/write
VTHLim	DINT	`Tolerance High` limit	Read/write
VTLLim	DINT	`Tolerance Low` limit	Read/write

5.3.3.5 FB_MTP_BinView

The function block `FB_MTP_BinView` provides the binary input value `V` via OPC UA. A text can be assigned to the states `TRUE` and `FALSE` of the input value via the inputs `VState0` and `VState1`.

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

Value scaling [▶ 19]

Unit [▶ 19]

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
V	BOOL	Value	Read
VState0	STRING	Text for value `FALSE`	Read
VState1	STRING	Text for value `TRUE`	Read

5.3.3.6 FB_MTP_BinMon

The function block `FB_MTP_BinMon` provides the binary input value `V` for manual operation (e. g. via OPC UA). A text can be assigned to the states `TRUE` and `FALSE` of the input value via the inputs `VState0` and `VState1`.

Monitoring

The binary input value `V` can be monitored for flutter (change of input signal). The output `VFlutAct` goes `TRUE` if the change in the input signal per time interval `VFlutTi` exceeds the maximum value `VFlutCnt` .

The output signal `VOut` corresponds to the debounced input value `V` .`VOut` and follows a change in the input signal `V` after `VFlutLen`. The input signal `V` must retain its value for this time.

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

OSLevel [▶ 18]

Inheritance hierarchy

FB_MTP_BinView

FB_MTP_BinMon

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
V	BOOL	Value	Read
VState0	STRING	Text for value `FALSE`	Read
VState1	STRING	Text for value `TRUE`	Read
VFlutEn	BOOL	Enable flutter monitoring: 1: Enabled 0: Disabled	Read
VFlutLen	REAL	Bounce time	Read

Outputs

Name	Type	Description	OPC UA access
VOut	BOOL	Output value	Read
VFlutAct	BOOL	Input value flutters	Read
Vout	BOOL	Debounced input value	Read

External variables

Name	Type	Description	OPC UA access
VFlutTi	REAL	Duration of the time interval	Read/write
VFlutCnt	INT	Maximum input signal changes per time interval	Read/write

5.3.3.7 FB_MTP_StringView

The function block `FB_MTP_StringView` makes the input variable `Text` available via OPC UA.

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

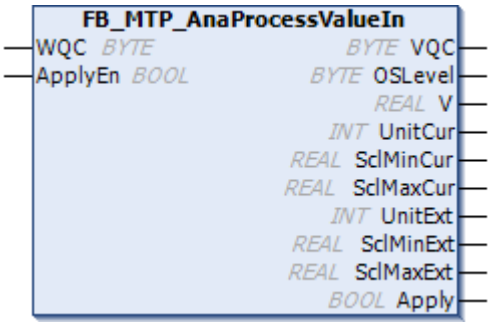
Inputs

Name	Type	Description	OPC UA access
WQC	BYTE	Worst Quality Code	Read
Text	STRING	Dynamic text	Read

5.3.4 InputElements

`InputElements` are process-related input values that can be written as input values via OPC UA, for example. It is possible to adjust the value range (unit with value scaling) to ensure that the values are displayed correctly.

5.3.4.1 FB_MTP_AnaProcessValueIn



The function block `FB_MTP_AnaProcessValueIn` provides an analog process value `V` with the associated `ValueQualityCode VQC` from a higher-level system (e.g. via OPC UA).

The variables `UnitCur`, `SclMinCur` and `SclMaxCur` are used to display the current unit and the current value scaling.

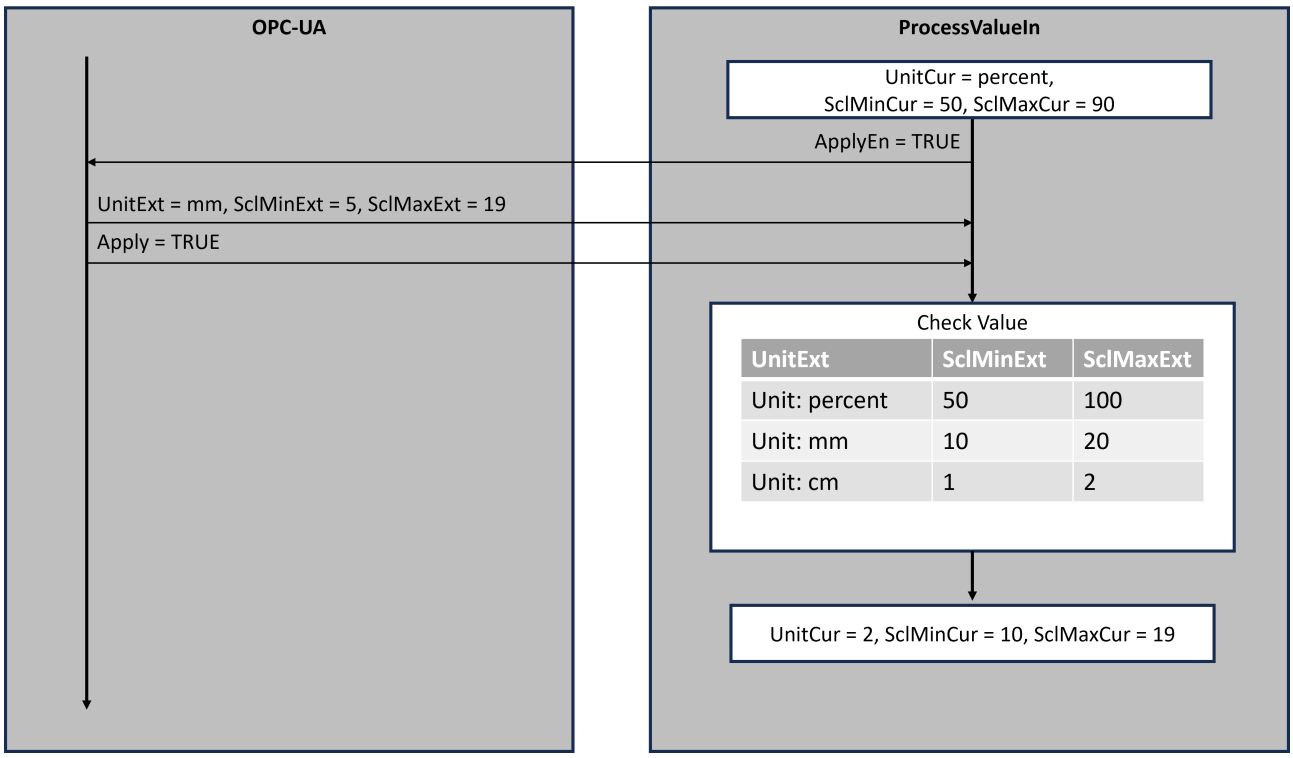
Applying a new configuration

A new configuration is specified via the variables `UnitExt`, `SclMinExt` and `SclMaxExt`.

To change the configuration (unit and value scaling), an array of the type ST_MTP_InputElementConfig [▶ 117] with a list of supported configurations is transferred on the PLC side via the input variable `Configs`. The value scaling for the respective units is stored in this array.

Enabling the application of the configuration is signaled on the PLC side by the variable `ApplyEn = TRUE`. The application can then be requested via the variable `Apply = TRUE`.

During the application, `UnitExt` is compared with the stored possible limit values in `Configs`. If the limit is exceeded or not reached, the stored upper or lower limit value is used.



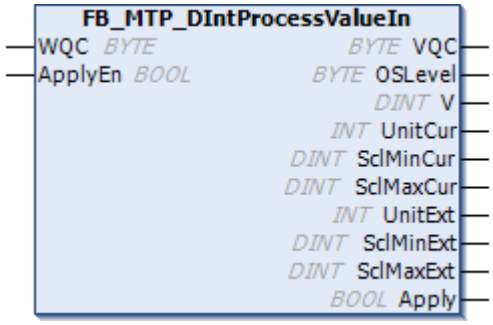
Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
Configs	ARRAY [*] OF ST_MTP_InputElementsConfig	Possible configurations	-
ApplyEn	BOOL	Enable for request: Apply configuration.	Read

Outputs

Name	Type	Description	OPC UA access
VQC	BYTE	Quality Code for the value	Read/write
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
V	REAL	Incoming process value via OPC UA	Read/write
UnitCur	INT	Current unit	Read
SclMinCur	REAL	Current scale start	Read
SclMaxCur	REAL	Current scale end	Read
UnitExt	INT	Configuration input for the unit	Read/write
SclMinExt	REAL	Configuration input for the scale start	Read/write
SclMaxExt	REAL	Configuration input for the scale end	Read/write
Apply	BOOL	Requirement: Apply configuration (Unit* and Scl*).	Read/write

5.3.4.2 FB_MTP_DIntProcessValueIn



The function block `FB_MTP_DIntProcessValueIn` provides an integer process value `V` with the associated `ValueQualityCode VQC` from a higher-level system (e.g. via OPC UA).

The variables `UnitCur`, `SclMinCur` and `SclMaxCur` are used to display the current unit and the current value scaling.

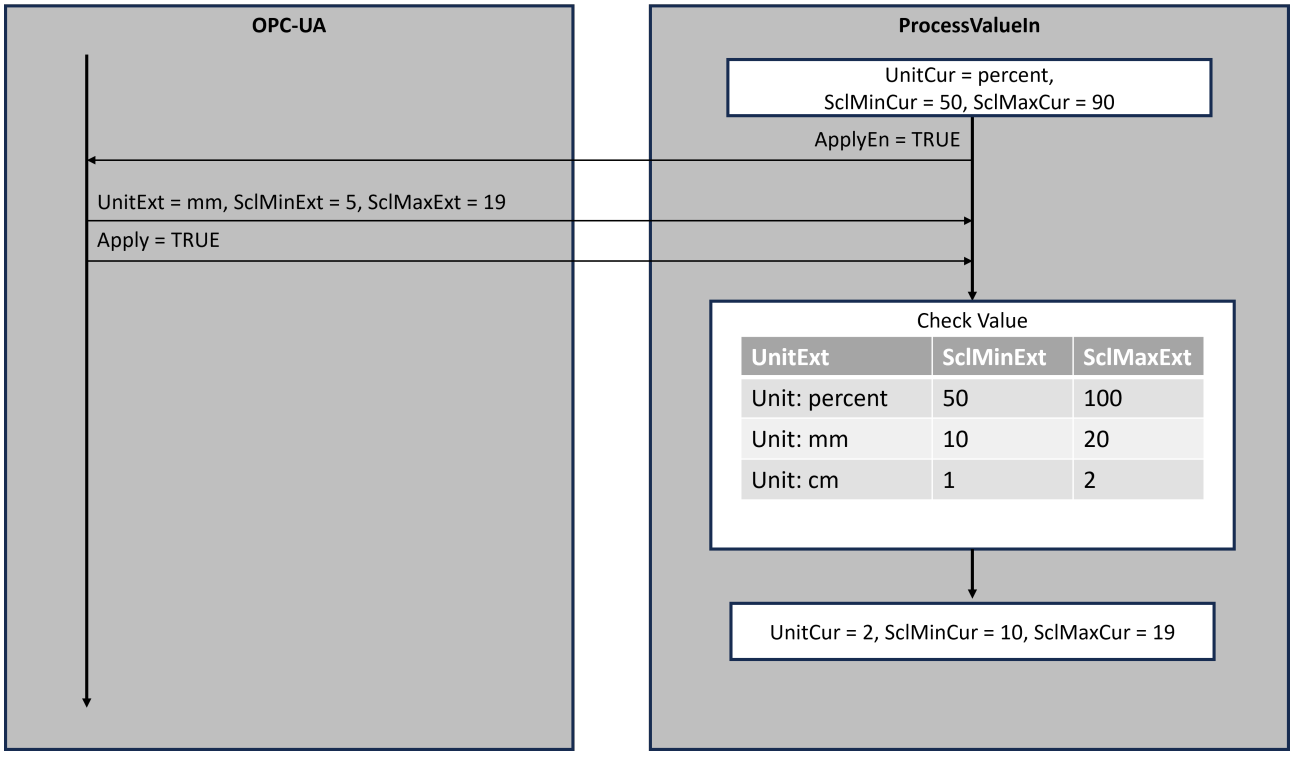
Applying a new configuration

A new configuration is specified via the variables `UnitExt`, `SclMinExt` and `SclMaxExt`.

To change the configuration (unit and value scaling), an array of the type ST_MTP_InputElementConfig [▶ 117] with a list of supported configurations is transferred on the PLC side via the input variable `Configs`. The value scaling for the respective units is stored in this array.

Enabling the application of the configuration is signaled on the PLC side by the variable `ApplyEn = TRUE`. The application can then be requested via the variable `Apply = TRUE`.

During the application, `UnitExt` is compared with the stored possible limit values in `Configs`. If the limit is exceeded or not reached, the stored upper or lower limit value is used.



Inputs

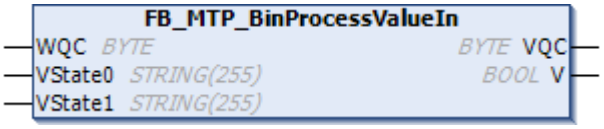
Name	Type	Description	OPC UA access
WQC	BYTE	Worst Quality Code	Read

Name	Type	Description	OPC UA access
Configs	ARRAY [*] OF ST_MTP_InputElementsConfig	Possible configurations	-
ApplyEn	BOOL	Enable for request: Apply configuration.	Read

Outputs

Name	Type	Description	OPC UA access
VQC	BYTE	Quality Code for the value	Read/write
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
V	DINT	Incoming process value via OPC UA	Read/write
UnitCur	INT	Current unit of the process value	Read
SclMinCur	DINT	Current scale start	Read
SclMaxCur	DINT	Current scale end	Read
UnitExt	INT	Configuration input for the unit	Read/write
SclMinExt	DINT	Configuration input for the scale start	Read/write
SclMaxExt	DINT	Configuration input for the scale end	Read/write
Apply	BOOL	Requirement: Apply configuration (Scl* and Unit*).	Read/write

5.3.4.3 FB_MTP_BinProcessValueIn



The function block FB_MTP_BinProcessValueIn provides the binary value V from a higher-level system (e.g. via OPC UA). A text can be assigned to the TRUE and FALSE states of the value via the inputs VState0 and VState1.

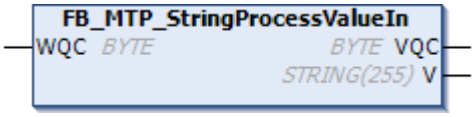
Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
VState0	String	Text for value = FALSE	Read
VState1	String	Text for value = TRUE	Read

Outputs

Name	Type	Description	OPC UA access
VQC	BYTE	Quality Code for the value	Read/write
V	BOOL	Value	Read/write

5.3.4.4 FB_MTP_StringProcessValueIn



The function block `FB_MTP_StringProcessValueIn` provides a process value `V` from a higher-level system (e.g. via OPC UA).

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read

Outputs

Name	Type	Description	OPC UA access
VQC	BYTE	Quality Code for the value	Read/write
V	STRING	Incoming process value via OPC UA	Read/write

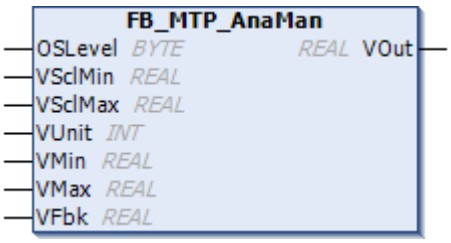
5.3.5 OutputElements

`OutputElements` are instances of IndicatorElements [▶ 69], which are defined in the MTP file as `OutputElements`. They can contain information for other plant areas and are made available to a control system via OPC UA, for example.

To use OutputElements, MTP Engineering and the MTP export contained in it need to be implemented!

5.3.6 OperationElements

5.3.6.1 FB_MTP_AnaMan



The function block `FB_MTP_AnaMan` is an object for an analog value specification of a manual operation `VMan` (e. g. via OPC UA). The output value `VOut` corresponds to the value limited by Value limitation [▶ 20].

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

OSLevel [▶ 18]

Feedback [▶ 20]

Value scaling [▶ 19]

Unit [▶ 19]

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
VUnit	INT	Unit of value	Read
VMin	REAL	Upper limit of the value specification	Read
VMax	REAL	Lower limit of the value specification	Read
VFbk	REAL	Feedback value	Read
VSclMin	REAL	Scale start for value display	Read
VSclMax	REAL	Scale end for value display	Read

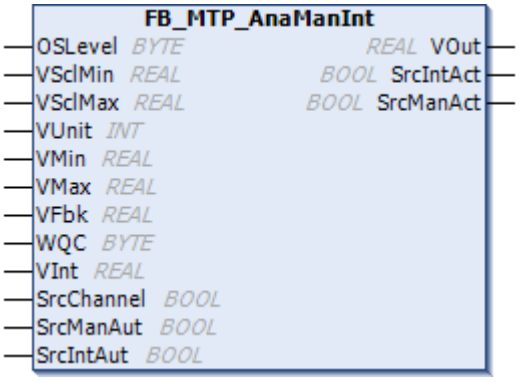
Outputs

Name	Type	Description	OPC UA access
VOut	REAL	Output value of the current value specification taking into account the limit values	Read

External variables

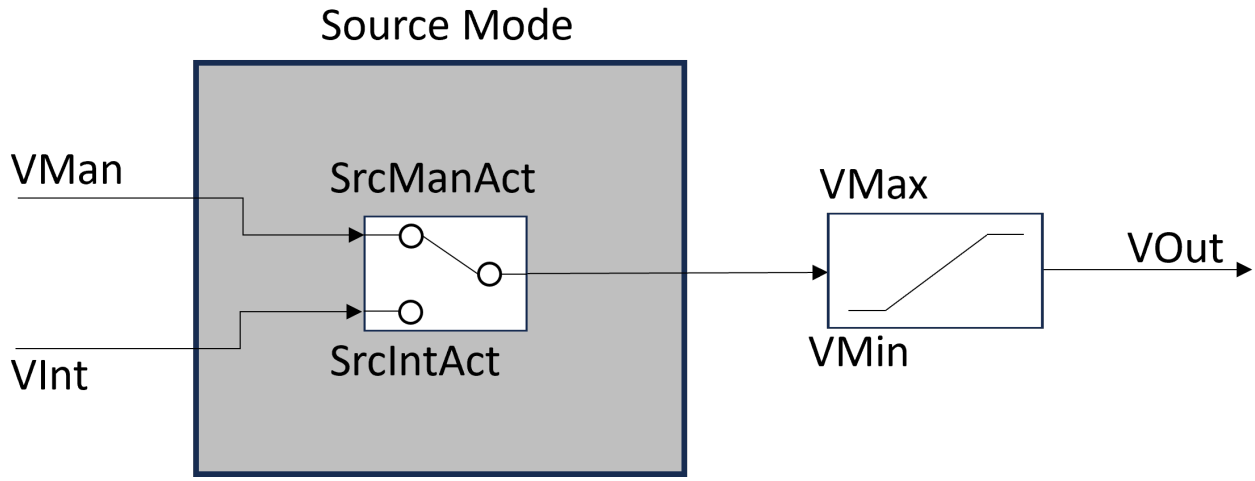
Name	Type	Description	OPC UA access
VMan	BOOL	`Manual` value specification	Read/write

5.3.6.2 FB_MTP_AnaManInt



The function block `FB_MTP_AnaManInt` is an object for an analog value specification from different sources: internal PLC logic `VInt` or manual operation `VMan` (e. g. via OPC UA). The values are managed via the state machine of the Source Mode [▶ 15].

The output value `VOut` corresponds to the value limited by Value limitation [▶ 20].



Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

OSLevel [▶ 18]

Feedback [▶ 20]

Value scaling [▶ 19]

Unit [▶ 19]

Inheritance hierarchy

FB_MTP_AnaMan

FB_MTP_AnaManInt

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
Src*		See Source Mode [▶ 15]	
VInt	REAL	`Internal` value specification. (Relevant if `SrcIntAct = TRUE`)	Read
VUnit	INT	Unit of value	Read
VMin	REAL	Upper limit of the value specification	Read
VMax	REAL	Lower limit of the value specification	Read
VFbk	REAL	Feedback value	Read
VSclMin	REAL	Scale start for value display	Read
VSclMax	REAL	Scale end for value display	Read

Outputs

Name	Type	Description	OPC UA access
Src*		See Source Mode [▶ 15]	
VOut	REAL	Output value of the current value specification taking into account the limit values	Read

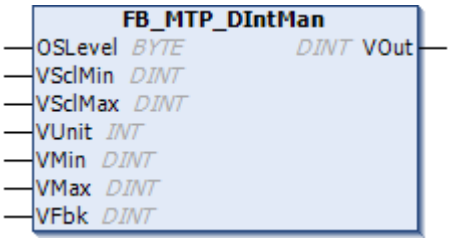
External variables

Name	Type	Description	OPC UA access
VMan	REAL	`Manual` value specification (Relevant if `SrcManAct = TRUE`)	Read
Src*		See Source Mode [▶ 15]	

Methods

Name	Description
SetInternal SetManual	See Source Mode [▶ 15]

5.3.6.3 FB_MTP_DIntMan



The function block `FB_MTP_DIntMan` is an object for an integer value specification of a manual operation `VMan` (e. g. via OPC UA). The output value `VOut` corresponds to the value limited by Value limitation [▶ 20].

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

OSLevel [▶ 18]

Feedback [▶ 20]

Value scaling [▶ 19]

Unit [▶ 19]

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
VUnit	INT	Unit of value	Read

Name	Type	Description	OPC UA access
VMin	DINT	Upper limit of the value specification	Read
VMax	DINT	Lower limit of the value specification	Read
VFbk	DINT	Feedback value	Read
VSclMin	DINT	Scale start for value display	Read
VSclMax	DINT	Scale end for value display	Read

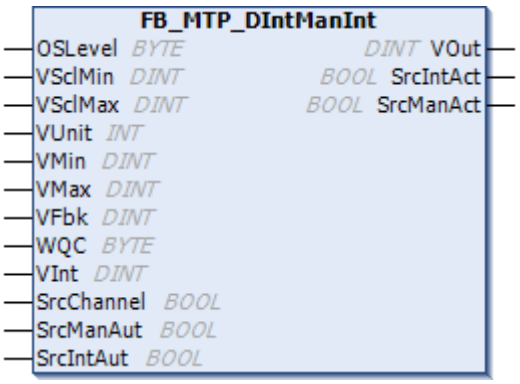
Outputs

Name	Type	Description	OPC UA access
VOut	DINT	Output value of the current value specification taking into account the limit values	Read

External variables

Name	Type	Description	OPC UA access
VMan	REAL	`Manual` value specification	Read/write

5.3.6.4 FB_MTP_DIntManInt



The function block `FB_MTP_DIntManInt` is an object for an integer value specification from different sources: internal PLC logic `VInt` or manual operation `VMan` (e. g. via OPC UA). The values are managed via the state machine of the Source Mode [▶ 15].

The output value `VOut` corresponds to the value limited by Value limitation [▶ 20].

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

OSLevel [▶ 18]

Feedback [▶ 20]

Value scaling [▶ 19]

Unit [▶ 19]

Inheritance hierarchy

FB_MTP_DIntMan

  FB_MTP_DIntManInt

Inputs

Name	Type	Description	OPC UA access
WQC	BYTE	Worst Quality Code	Read
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
Src*		See Source Mode [▶ 15]	
VInt	DINt	`Internal` value specification. (Relevant if `SrcIntAct = TRUE`)	Read
VUnit	INT	Unit of value	Read
VMin	DINT	Upper limit of the value specification	Read
VMax	DINT	Lower limit of the value specification	Read
VFbk	DINT	Feedback value	Read
VSclMin	DINT	Scale start for value display	Read
VSclMax	DINT	Scale end for value display	Read

Outputs

Name	Type	Description	OPC UA access
Src*		See Source Mode [▶ 15]	
VOut	DINT	Output value of the current value specification taking into account the limit values	Read

External variables

Name	Type	Description	OPC UA access
Src*		See Source Mode [▶ 15]	
VMan	REAL	`Manual` value specification (Relevant if `SrcManAct = TRUE`)	Read

Methods

Name	Description
SetInternal SetManual	See Source Mode [▶ 15]

5.3.6.5 FB_MTP_BinMan

The function block `FB_MTP_BinMan` is an object for a binary value specification of a manual operation `VMan` (e. g. via OPC UA). The states `TRUE` and `FALSE` of the value specification are assigned state texts via the input variables `VState0` and `VState1`.

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

OSLevel [▶ 18]

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
VState0	STRING	Text for value `VOut = FALSE`	Read
VState1	STRING	Text for value `VOut = TRUE`	Read
VFbk	BOOL	Feedback value	Read

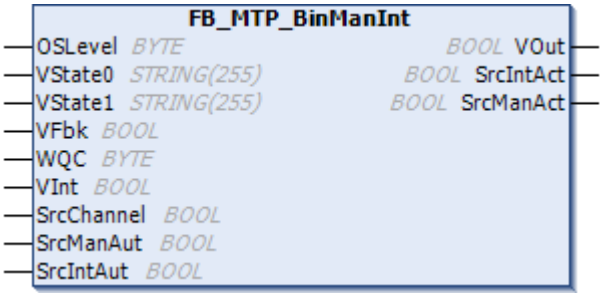
Outputs

Name	Type	Description	OPC UA access
VOut	REAL	Output value of the current value specification	Read

External variables

Name	Type	Description	OPC UA access
VMan	BOOL	`Manual` value specification	Read

5.3.6.6 FB_MTP_BinManInt



The function block `FB_MTP_BinManInt` is an object for a binary value specification from different sources: internal PLC logic `VInt` or manual operation `VMan` (e. g. via OPC UA). The values are managed via the state machine of the Source Mode [▶ 15]. The states `TRUE` and `FALSE` of the value specification are assigned state texts via the input variables `VState0` and `VState1`.

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

OSLevel [▶ 18]

Inheritance hierarchy

FB_MTP_BinMan

  FB_MTP_BinManInt

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
VUnit	INT	Unit of value	Read
VMin	DINT	Upper limit of the value specification	Read
VMax	DINT	Lower limit of the value specification	Read
VFbk	DINT	Feedback value	Read
VSclMin	DINT	Scale start for value display	Read
VSclMax	DINT	Scale end for value display	Read
Src*		See Source Mode [▶ 15]	
VInt	BOOL	`Internal` value specification (Relevant if `SrcIntAct = True`)	Read

Outputs

Name	Type	Description	OPC UA access
Src*		See Source Mode [▶ 15]	
VOut	DINT	Output value of the current value specification taking into account the limit values	Read

External variables

Name	Type	Description	OPC UA access
Src*		See Source Mode [▶ 15]	
VMan	BOOL	`Manual` value specification (Relevant if `SrcManAct = True`)	Read

Methods

Name	Description
SetInternal SetManual	See Source Mode [▶ 15]

5.3.7 ParameterElements

`ParameterElements` are objects for value specifications by parameters from different sources: internal PLC logic and manual operation (e.g. via OPC UA). The values are managed via the state machine Service Mode [▶ 17].

The parameters can be divided into configuration parameters and procedure parameters. Configuration parameters are assigned to a service and can be used by all procedures. Procedure parameters are assigned to one or more procedures within a service.

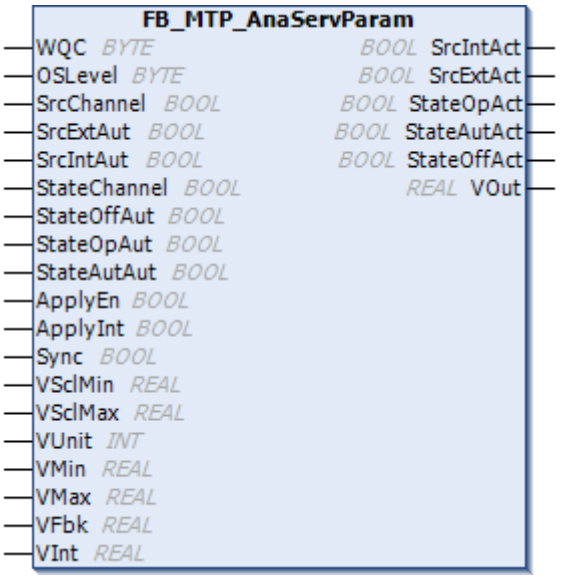
Parameter instances that are transferred to the input `ConfParameters` of a service are considered configuration parameters.

Parameter instances that are transferred to one or more procedures of the service at the input `ProcParameters` are considered procedure parameters.

The sum of all procedure parameters of the procedures of a service must also be transferred collectively at the input `ProcParameters` of the higher-level service.

For the use of parameters, the use of MTP engineering and its automatic code generation is recommended!

5.3.7.1 FB_MTP_AnaServParam



The function block `FB_MTP_AnaServParam` is an object for analog parameter value specifications from different sources: internal PLC logic or manual operation (e.g. via OPC UA). The values are managed via the state machine Service Mode [▶ 17].

The output value `VOut` corresponds to the current value specification limited to limit values and can be used within a service (see ParameterElements [▶ 87]).

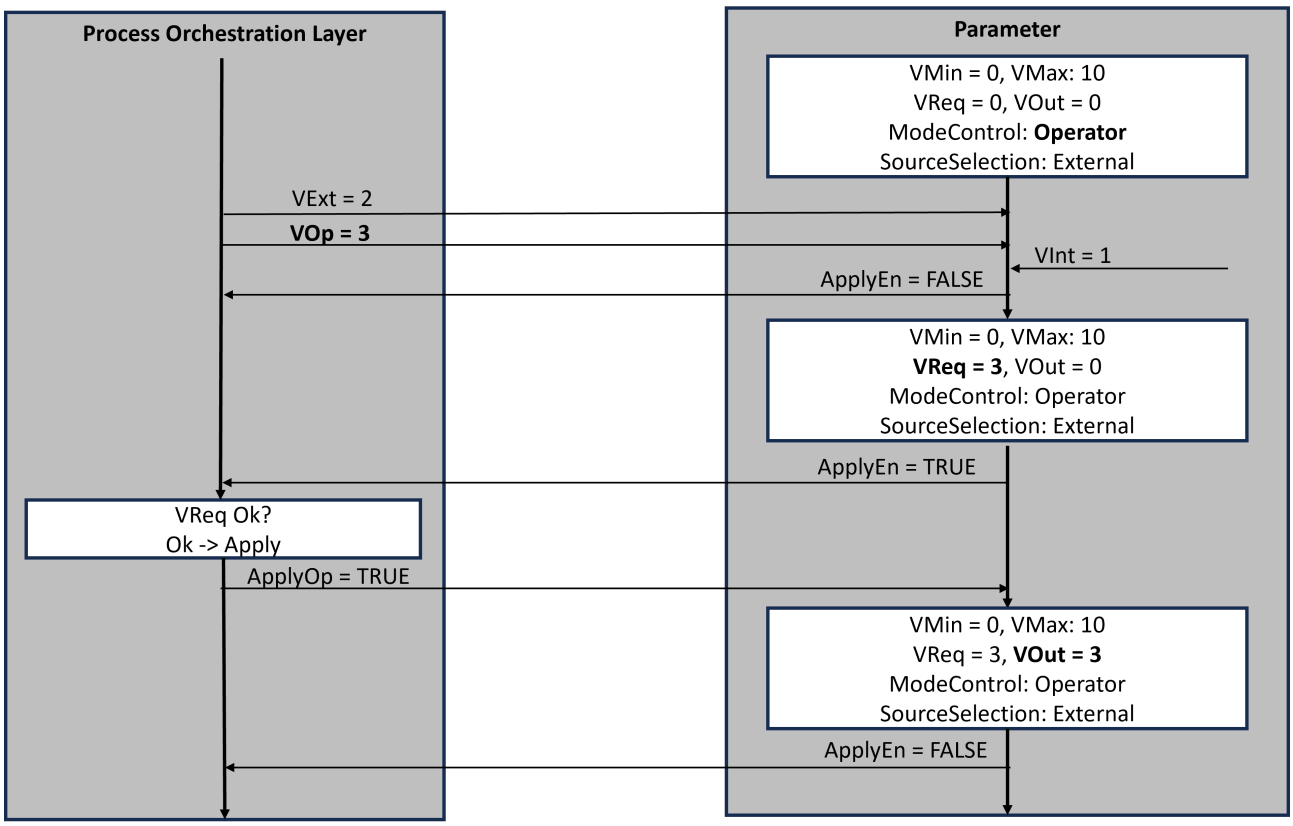
Applying values

Each parameter follows the same procedure when applying values. The values are applied in two steps:

1. Preparing the value
2. Applying of the value

The preparation of the value begins with the value specification via the variables `VOp`, `VInt`, `VExt`. Depending on the state of the Service Mode [▶ 17], a value is selected and limited via the value limit. The resulting requested value is displayed via the variable `VReq`.

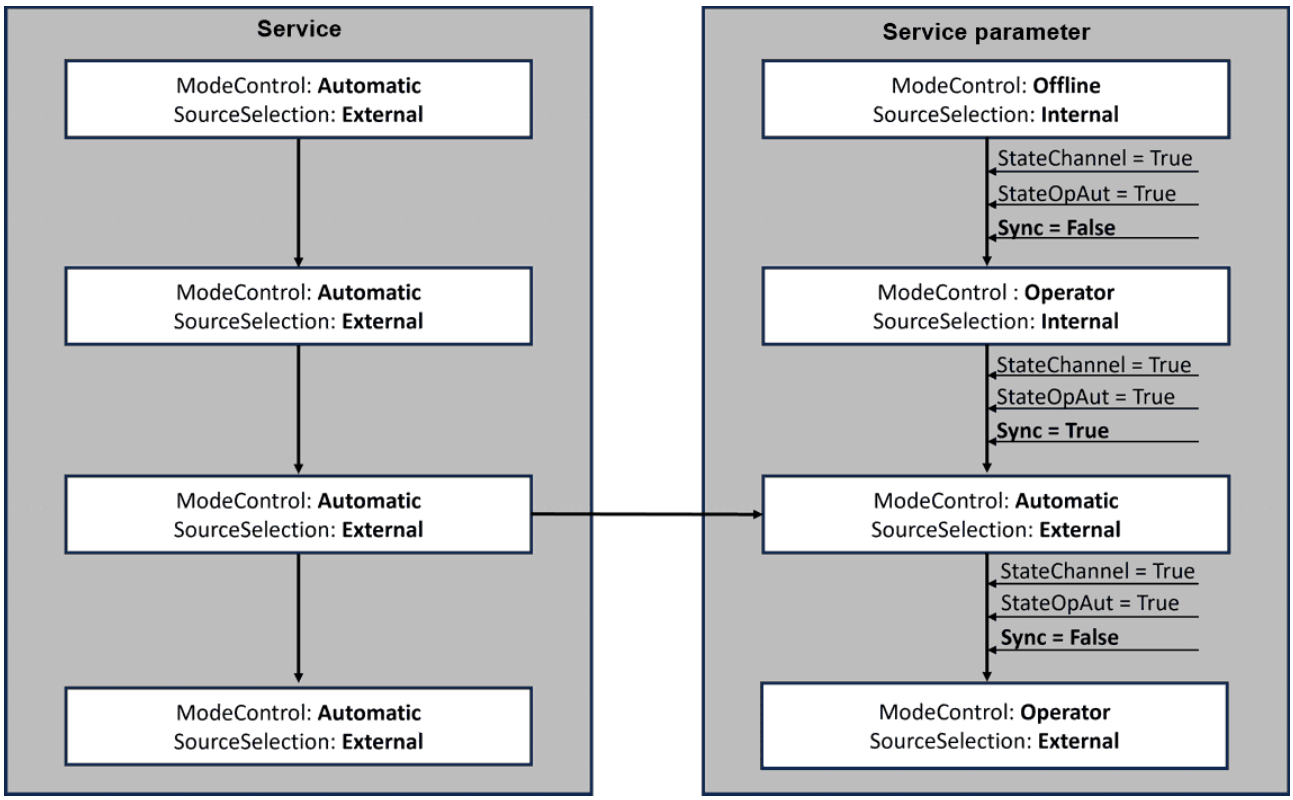
To apply the value, the enabling is signaled on the PLC side via the variable `ApplyEn = TRUE`. The application can then be confirmed via variables `ApplyOp`, `ApplyInt` or `ApplyExt`, depending on the state of Service Mode [▶ 17]. After successful application, the value of the variable `VOut` corresponds to the value of `VReq`.



Values can also be applied at service level using the same procedure. See the Parameters section at FB_MTP_ServiceControl [▶ 106].

Operation mode synchronization with higher-level service

The `Service Mode` of the parameter can be synchronized with the higher-level service via the variable `Sync`. The input variables for controlling the state machine are ignored and the state machine of the parameter automatically assumes the state of the Service Mode [▶ 17] of the service.



Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

OSLevel [▶ 18]

Service Mode [▶ 17]

Value scaling [▶ 19]

Unit [▶ 19]

Value limitation [▶ 20]

Read back [▶ 20]

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
State* Src*		See Service Mode [▶ 17]	
ApplyEn	BOOL	Enable Apply parameter: 1: Parameters can be applied. 0: Parameters cannot be applied.	Read
ApplyInt	BOOL	`Internal` switching request to apply requested parameters (relevant if `StateAutAct = TRUE` and `SrcIntAct = TRUE`): 1: Apply value. 0: Do not apply value	Read
sync	BOOL	Operation mode synchronization: 1: Synchronize with service. 0: Independent operation	Read/write
VInt	REAL	`Internal` value specification (relevant if `StateAutAct = TRUE` and `SrcIntAct = TRUE`)	Read
VUnit	INT	Unit of value	Read
VMin	REAL	Upper limit of the value specification	Read
VMax	REAL	Lower limit of the value specification	Read
VFbk	REAL	Feedback value	Read
VSclMin	REAL	Scale start for value display	Read
VSclMax	REAL	Scale end for value display	Read

Outputs

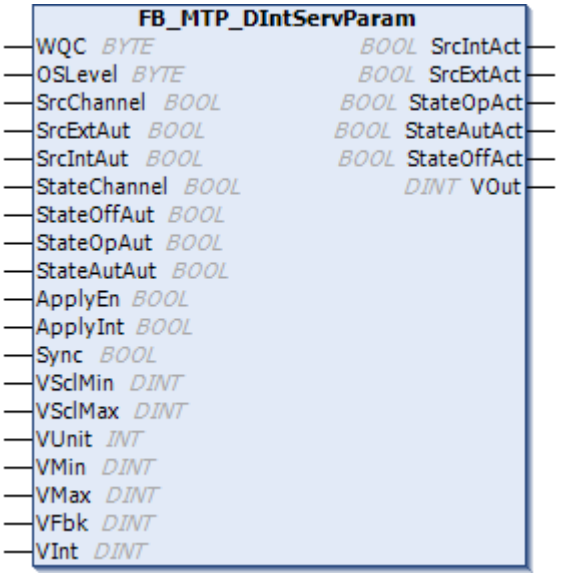
Name	Type	Description	OPC UA access
State*		See Service Mode [▶ 17]	

Name	Type	Description	OPC UA access
Src*			
VOut	REAL	Current parameter value	Read

External variables

Name	Type	Description	OPC UA access
State* Src*		See Service Mode [▶ 17]	
ApplyOp	BOOL	`Operator` switching request to apply currently requested parameters (relevant if `StateOpAct = TRUE`): 1: Apply value. 0: Do not apply value	Read/write
ApplyExt	BOOL	`External` switching request to apply currently requested parameters (relevant if `SrcExtAct = TRUE` and `StateAutAct = TRUE`): 1: Apply value. 0: Do not apply value	Read/write
VOp	REAL	`Operator` value specification (relevant if `StateOpAct = TRUE`)	Read/write
VExt	REAL	`External` value specification (relevant if `SrcExtAct = TRUE` and `StateAutAct = TRUE`)	Read/write

5.3.7.2 FB_MTP_DIntServParam



The function block `FB_MTP_DIntServParam` is an object for integer parameter value specifications from different sources: internal PLC logic and manual operation (e.g. via OPC UA). The values are managed via the state machine Service Mode [▶ 17].

The output value `VOut` corresponds to the current value specification limited to limit values and can be used within a service (see ParameterElements [▶ 87]).

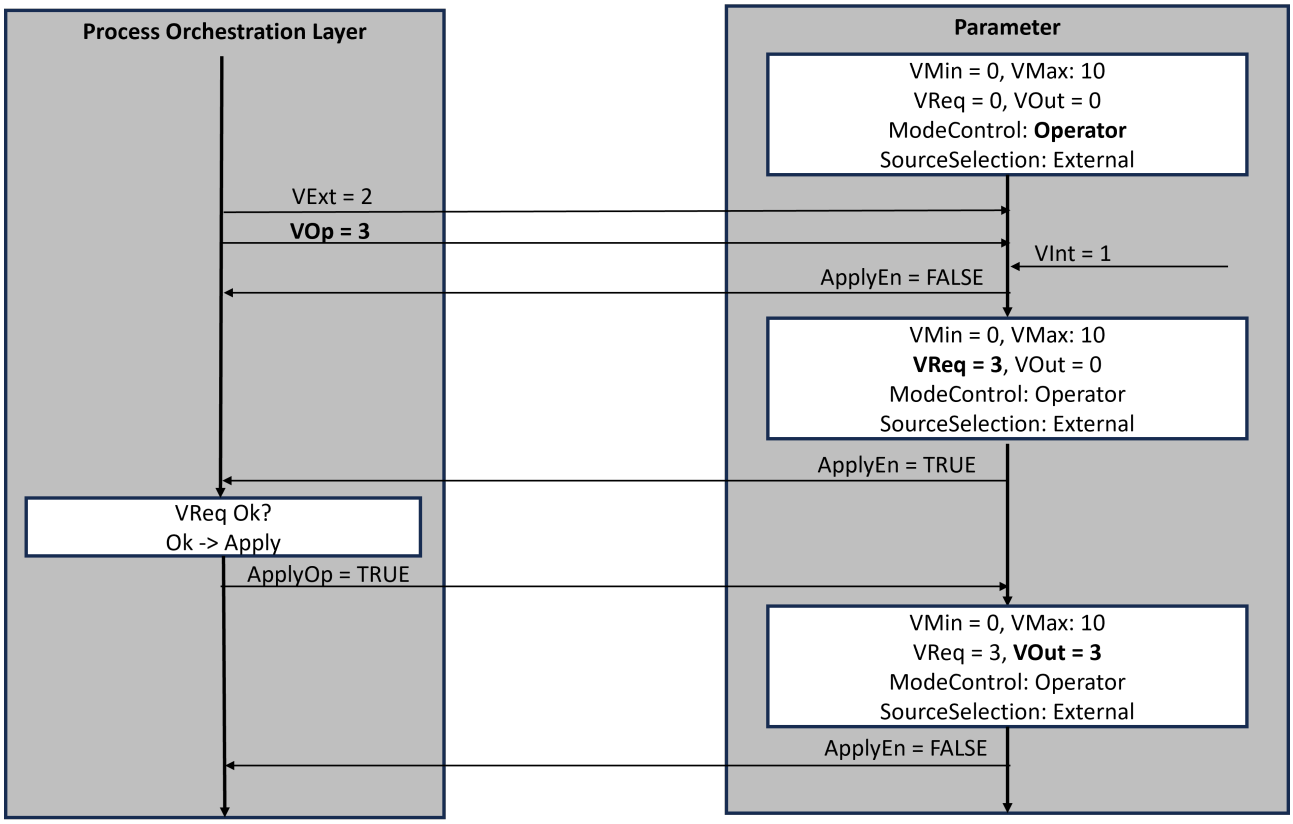
Applying values

Each parameter follows the same procedure when applying values. The values are applied in two steps:

1. Preparing the value
2. Applying of the value

The preparation of the value begins with the value specification via the variables `VOp`, `VInt`, `VExt`. Depending on the state of the Service Mode [▶ 17], a value is selected and limited via the value limit. The resulting requested value is displayed via the variable `VReq`.

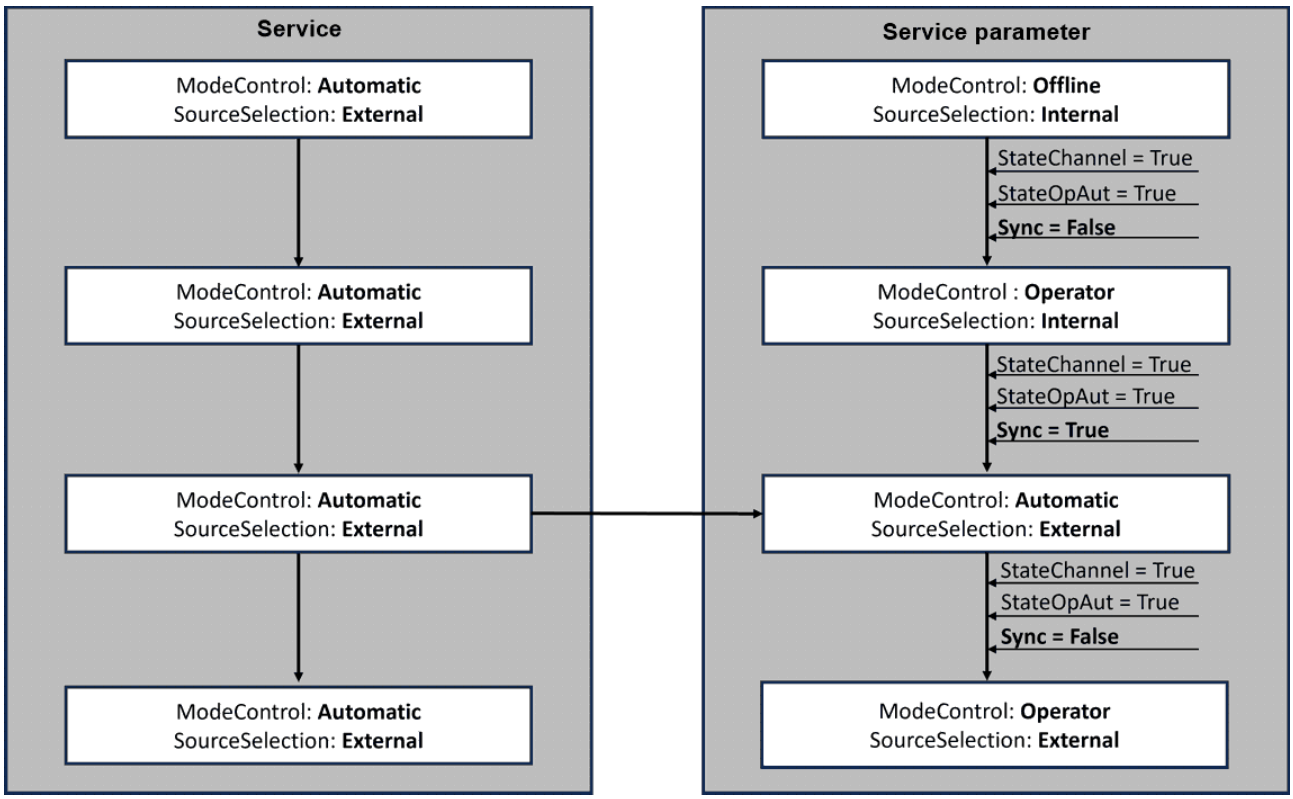
To apply the value, the enabling is signaled on the PLC side via the variable `ApplyEn = TRUE`. The application can then be confirmed via variables `ApplyOp`, `ApplyInt` or `ApplyExt`, depending on the state of Service Mode [▶ 17]. After successful application, the value of the variable `VOut` corresponds to the value of `VReq`.



Values can also be applied at service level using the same procedure. See the Parameters section at FB_MTP_ServiceControl [▶ 106].

Operation mode synchronization with higher-level service

The `Service Mode` of the parameter can be synchronized with the higher-level service via the variable `Sync`. The input variables for controlling the state machine are ignored and the state machine of the parameter automatically assumes the state of the Service Mode [▶ 17] of the service.



Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

OSLevel [▶ 18]

Service Mode [▶ 17]

Value scaling [▶ 19]

Unit [▶ 19]

Value limitation [▶ 20]

Read back [▶ 20]

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
State* Src*		See Service Mode [▶ 17]	
ApplyEn	BOOL	Enable Apply parameter: 1: Parameters can be applied. 0: Parameters cannot be applied.	Read

Name	Type	Description	OPC UA access
ApplyInt	BOOL	`Internal` switching request to apply requested parameters (relevant if `StateAutAct = TRUE` and `SrcIntAct = TRUE`): 1: Apply value. 0: Do not apply value	Read
sync	BOOL	Operation mode synchronization: 1: Synchronize with service. 0: Independent operation	Read/write
VInt	DINT	`Internal` value specification (relevant if `StateAutAct = TRUE` and `SrcIntAct = TRUE`)	Read
VUnit	INT	Unit of value	Read
VMin	DINT	Upper limit of the value specification	Read
VMax	DINT	Lower limit of the value specification	Read
VFbk	DINT	Feedback value	Read
VSclMin	DINT	Scale start for value display	Read
VSclMax	DINT	Scale end for value display	Read

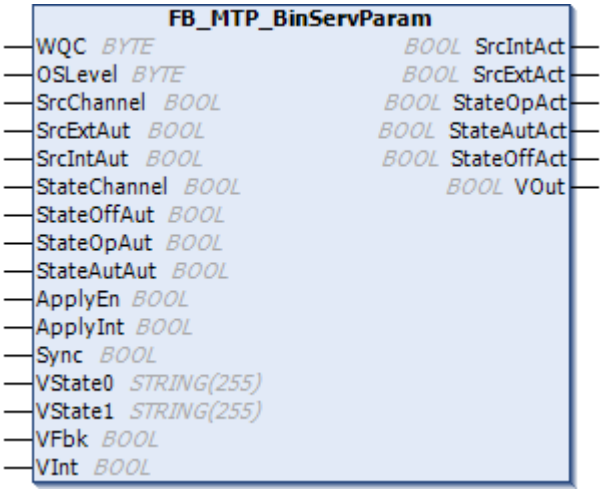
Outputs

Name	Type	Description	OPC UA access
State* Src*		See Service Mode [▶ 17]	
Vout	DINT	Current parameter value	Read

External variables

Name	Type	Description	OPC UA access
State* Src*		See Service Mode [▶ 17]	
ApplyOp	BOOL	`Operator` switching request to apply currently requested parameters (relevant if `StateOpAct = TRUE`): 1: Apply value. 0: Do not apply value	Read/write
ApplyExt	BOOL	`External` switching request to apply currently requested parameters (relevant if `SrcExtAct = TRUE` and `StateAutAct = TRUE`): 1: Apply value. 0: Do not apply value	Read/write
VOp	DINT	`Operator` value specification (relevant if `StateOpAct = TRUE`)	Read/write
VExt	DINT	`External` value specification (relevant if `SrcExtAct = TRUE` and `StateAutAct = TRUE`)	Read/write

5.3.7.3 FB_MTP_BinServParam



The function block `FB_MTP_BinServParam` is an object for binary parameter value specifications from different sources: internal PLC logic and manual operation (e.g. via OPC UA). The values are managed via the state machine Service Mode [▶ 17].

The output value `VOut` corresponds to the current value specification limited to limit values and can be used within a service (see ParameterElements [▶ 87]).

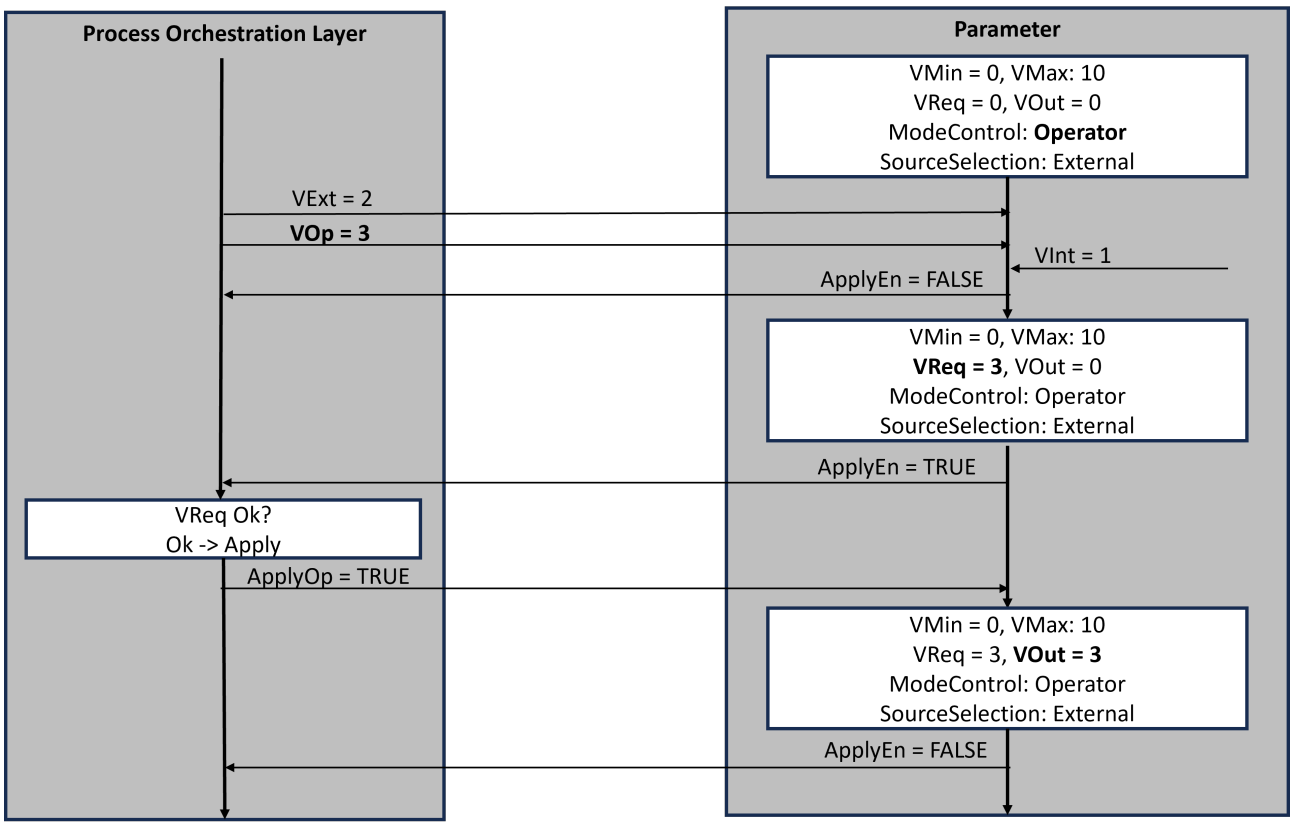
Applying values

Each parameter follows the same procedure when applying values. The values are applied in two steps:

1. Preparing the value
2. Applying of the value

The preparation of the value begins with the value specification via the variables `VOp`, `VInt`, `VExt`. Depending on the state of the Service Mode [▶ 17], a value is selected and limited via the value limit. The resulting requested value is displayed via the variable `VReq`.

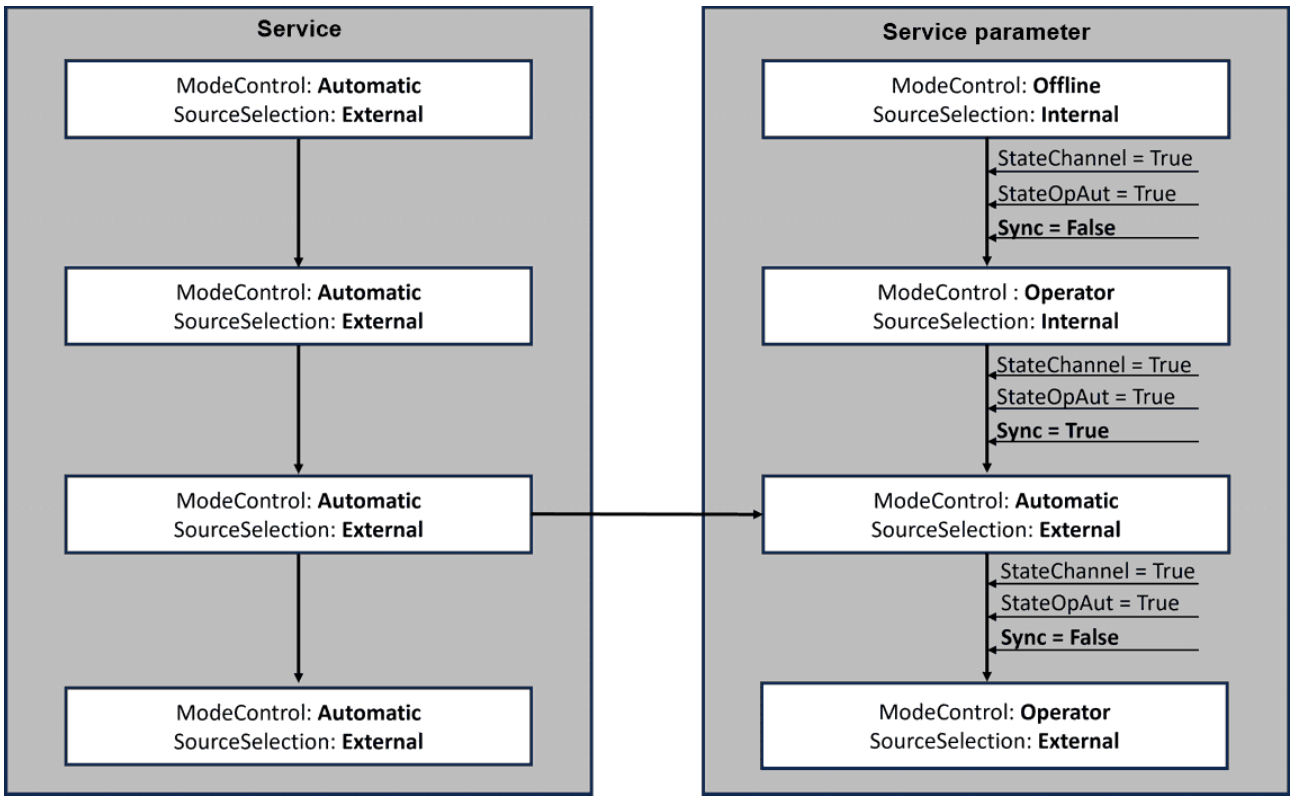
To apply the value, the enabling is signaled on the PLC side via the variable `ApplyEn = TRUE`. The application can then be confirmed via variables `ApplyOp`, `ApplyInt` or `ApplyExt`, depending on the state of Service Mode [▶ 17]. After successful application, the value of the variable `VOut` corresponds to the value of `VReq`.



Values can also be applied at service level using the same procedure. See the Parameters section at FB_MTP_ServiceControl [▶ 106].

Operation mode synchronization with higher-level service

The `Service Mode` of the parameter can be synchronized with the higher-level service via the variable `Sync`. The input variables for controlling the state machine are ignored and the state machine of the parameter automatically assumes the state of the Service Mode [▶ 17] of the service.



Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

OSLevel [▶ 18]

Service Mode [▶ 17]

Read back [▶ 20]

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
State* Src*		See Service Mode [▶ 17]	
ApplyEn	BOOL	Enable Apply parameter: 1: Parameters can be applied. 0: Parameters cannot be applied.	Read
ApplyInt	BOOL	`Internal` switching request to apply requested parameters (relevant if `StateAutAct = TRUE` and `SrcIntAct = TRUE`): 1: Apply value. 0: Do not apply value	Read
sync	BOOL	Operation mode synchronization: 1: Synchronize with service. 0: Independent operation	Read/write
VInt	BOOL	`Internal` value specification (relevant if `StateAutAct = TRUE` and `SrcIntAct = TRUE`)	Read
VState0	STRING	Text for value `VOut = FALSE`	Read
VState1	STRING	Text for value `VOut = TRUE`	Read
VFbk	BOOL	Feedback value	Read

Outputs

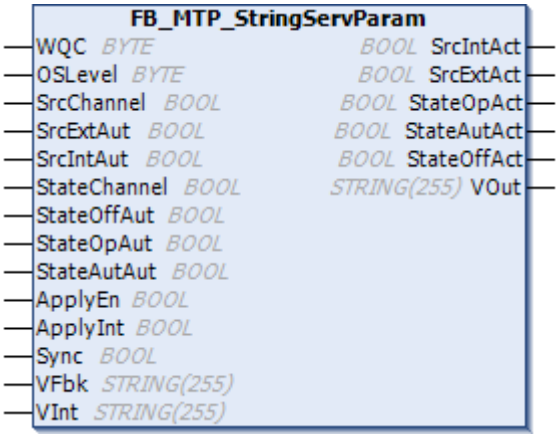
Name	Type	Description	OPC UA access
State* Src*		See	
Vout	BOOL	Current parameter value	Read

External variables

Name	Type	Description	OPC UA access
State* Src*		See Service Mode [▶ 17]	

Name	Type	Description	OPC UA access
ApplyOp	BOOL	`Operator` switching request to apply currently requested parameters (relevant if `StateOpAct = TRUE`): 1: Apply value. 0: Do not apply value	Read/write
ApplyExt	BOOL	`External` switching request to apply currently requested parameters (relevant if `SrcExtAct = TRUE` and `StateAutAct = TRUE`): 1: Apply value. 0: Do not apply value	Read/write
VOp	REAL	`Operator` value specification (relevant if `StateOpAct = TRUE`)	Read/write
VExt	REAL	`External` value specification (relevant if `SrcExtAct = TRUE` and `StateAutAct = TRUE`)	Read/write

5.3.7.4 FB_MTP_StringServParam



The function block `FB_MTP_StringServParam` is an object for string parameter value specifications from different sources: internal PLC logic and manual operation (e.g. via OPC UA). The values are managed via the state machines Service Mode [▶ 17].

The output value `VOut` corresponds to the current value specification limited to limit values and can be used within a service (see ParameterElements [▶ 87]).

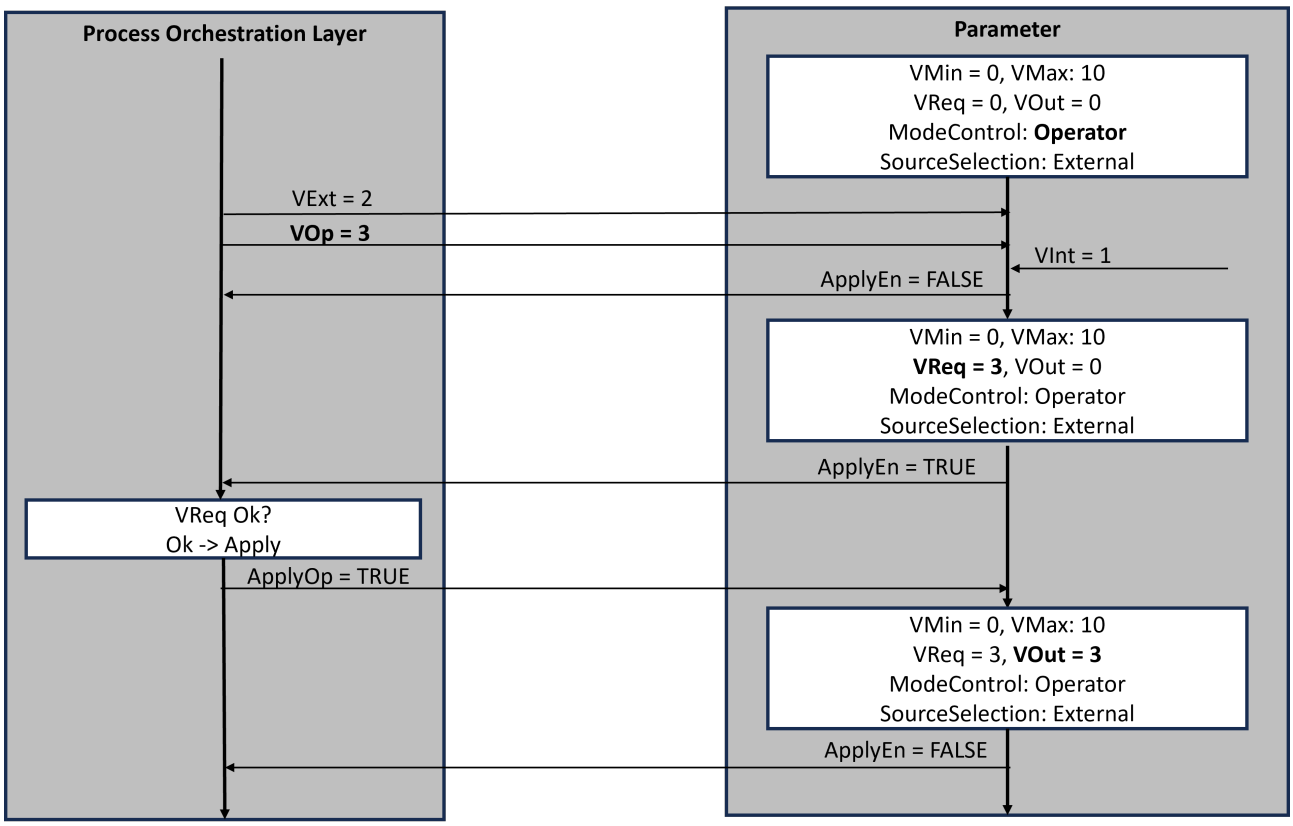
Applying values

Each parameter follows the same procedure when applying values. The values are applied in two steps:

1. Preparing the value
2. Applying of the value

The preparation of the value begins with the value specification via the variables `VOp`, `VInt`, `VExt`. Depending on the state of the Service Mode [▶ 17], a value is selected and limited via the value limit. The resulting requested value is displayed via the variable `VReq`.

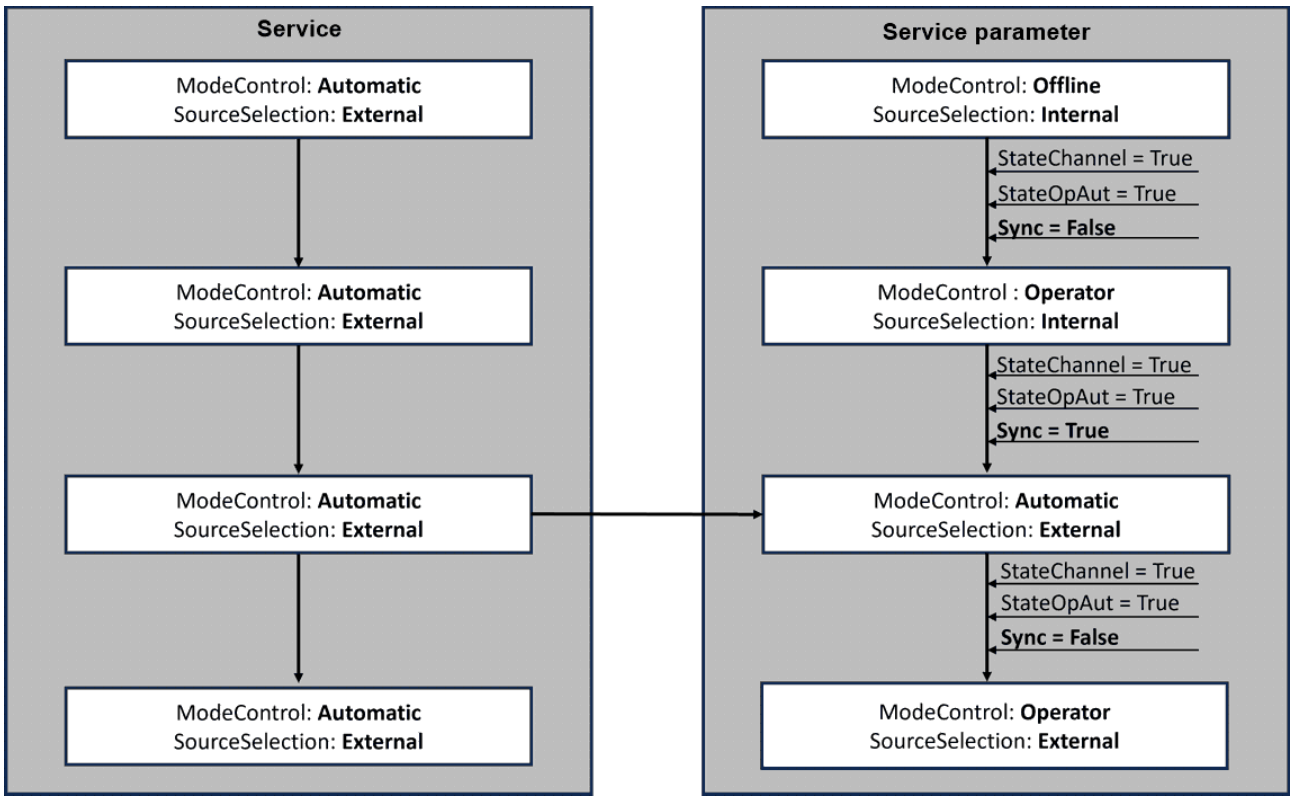
To apply the value, the enabling is signaled on the PLC side via the variable `ApplyEn = TRUE`. The application can then be confirmed via variables `ApplyOp`, `ApplyInt` or `ApplyExt`, depending on the state of Service Mode [▶ 17]. After successful application, the value of the variable `VOut` corresponds to the value of `VReq`.



Values can also be applied at service level using the same procedure. See the Parameters section at FB_MTP_ServiceControl [▶ 106].

Operation mode synchronization with higher-level service

The `Service Mode` of the parameter can be synchronized with the higher-level service via the variable `Sync`. The input variables for controlling the state machine are ignored and the state machine of the parameter automatically assumes the state of the Service Mode [▶ 17] of the service.



Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

OSLevel [▶ 18]

Service Mode [▶ 17]

Read back [▶ 20]

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
VInt	STRING	`Internal` value specification (relevant if `StateAutAct = TRUE` and `SrcIntAct = TRUE`)	Read
VUnit	INT	Unit of value	Read
VMin	REAL	Upper limit of the value specification	Read
VMax	REAL	Lower limit of the value specification	Read
VFbk	REAL	Feedback value	Read
VSclMin	REAL	Scale start for value display	Read
VSclMax	REAL	Scale end for value display	Read

Outputs

Name	Type	Description	OPC UA access
State* Src*		See Service Mode [▶ 17]	
Vout	STRING	Current parameter value	Read

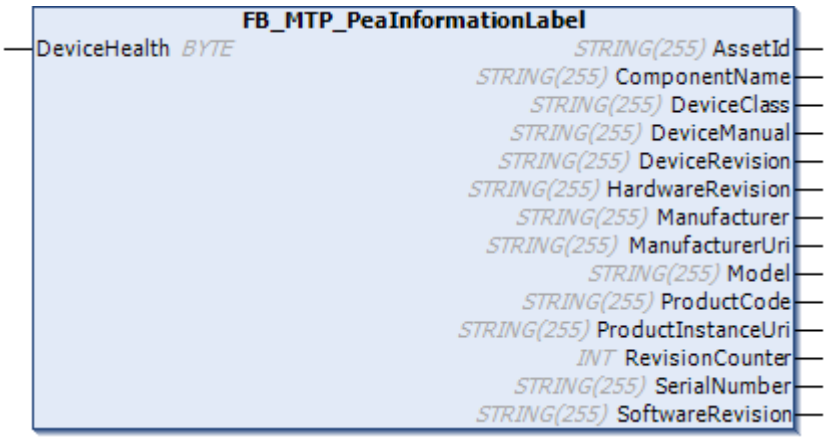
External variables

Name	Type	Description	OPC UA access
State* Src*		See Service Mode [▶ 17]	
ApplyOp	BOOL	`Operator` switching request to apply currently requested parameters (relevant if `StateOpAct = TRUE`): 1: Apply value. 0: Do not apply value	Read/write
ApplyExt	BOOL	`External` switching request to apply currently requested parameters (relevant if `SrcExtAct = TRUE` and `StateAutAct = TRUE`): 1: Apply value. 0: Do not apply value	Read/write
VOp	STRING	`Operator` value specification (relevant if `StateOpAct = TRUE`)	Read/write

Name	Type	Description	OPC UA access
VExt	STRING	`External` value specification (relevant if `SrcExtAct = TRUE` and `StateAutAct = TRUE`)	Read/write

5.3.8 PeaElements

5.3.8.1 FB_MTP_PeaInformationLabel



The function block `FB_MTP_PeaInformationLabel` is an object that provides information about a `PEA` during runtime (e.g. via OPC UA). `PEA` stands for `Process Equipment Assembly` and is a process technology module. One instance of this object is created per module. The information at the output of this FB is initialized when the FB is instantiated. These values can be read and changed during runtime by a manual operation (e.g. via OPC UA) (see access rights in the variable tables).

The variables `AssetId`, `ComponentName` and `DeviceClass` are declared as persistent.

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
DeviceHealth	BYTE	Summarized WQC at module level (see also WQC [▶ 19])	Read

Outputs

Name	Type	Description	OPC UA access
AssetId	STRING	Unique ID within a system. This can be specified by the system operator.	Read/write
ComponentName	STRING	Module name within a system. This can be specified by the system operator.	Read/write
DeviceClass	STRING	The device class can be predefined by the PEA manufacturer. This value can be overwritten by the system operator.	Read/write
DeviceManual	STRING	Reference to module-related documents (e.g. web pages)	Read
DeviceRevision	STRING	Version information of the interface. This is executed as a Major.Minor.Patch notation (e.g. 1.2.3).	Read
HardwareRevision	STRING	Version information of the structure of a PEA. This is executed as a Major.Minor.Patch notation (e.g. 1.2.3).	Read
Manufacturer	STRING	Name of the module manufacturer	Read
ManufacturerUri	STRING	Address of the manufacturer's web pages (e.g. www.Beckhoff.de)	Read

Name	Type	Description	OPC UA access
Model	STRING	Name of the PEA. This value is identical to the name of the MTP.	Read
ProductCode	STRING	Unique product name of the PEA	Read
ProductInstanceUri	STRING	Unique identifier (URI) of the PEA (according to DIN SPEC 91406)	Read
RevisionCounter	STRING	Revision counter	Read
SerialNumber	STRING	Serial number of the PEA (Assigned by the PEA manufacturer.)	Read
SoftwareRevision	STRING	Version information of the PLC software. This is executed as a Major.Minor.Patch notation (e.g. 1.2.3).	Read

5.3.8.2 FB_MTP_WebServerUrlInfo

The function block `FB_MTP_WebServerUrlInfo` is an object for providing an endpoint of an existing web server.

If a web server is to be specified, the endpoint of the web server can be instantiated via the variable `Url` when the object is initialized.

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-

Outputs

Name	Type	Description	OPC UA access
Url	STRING	Endpoint of the web server	Read

5.3.9 ServiceElements

System functions can be executed in encapsulated services FB_MTP_ServiceControl [▶ 106]. The service bundles various procedures, parameters and report values. The execution takes place in procedures. Procedures are different execution variants of a service.

One example would be filling a tank: the service could be called `Filling`. The procedures of this service could be called `Volume_Filling` and `Duration_Filling` with different parameters to fill a certain volume or for a certain time.

Services describe the function and execute it in procedures, so a service must always have at least one procedure FB_MTP_Procedure [▶ 103].

The following parameters can be transferred to the procedures for execution:

Configuration parameters

Configuration parameters are assigned to all procedures of a service. They are typically set during commissioning.

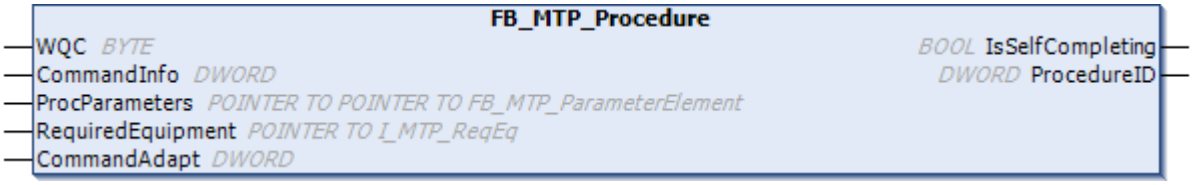
Procedure parameters

Unlike configuration parameters, procedure parameters can be assigned to one or more procedures of a service. They are typically set before a service is started.

Report values

`ReportValues` are assigned to one or more procedures of a service. They can be used, for example, to document important process values.

5.3.9.1 FB_MTP_Procedure



The function block `FB_MTP_Procedure` contains all attributes and methods for describing a procedure. A procedure is an execution variant of a service and can therefore only be assigned to one service. All procedures receive a unique `ProcedureID` in the service. This must be described by a natural number, whereby the number 0 may not be used! The variable `IsSelfCompleting` is used to specify whether the process function(s) of the procedure complete(s) themselves (e.g. after a time has elapsed).

The outputs `IsSelfCompleting` and `ProcedureID` are initialized when this function block is instantiated.

`CommandInfo` is coded according to and is updated separately for each procedure that is in the state `Automatic` or `Operator`, regardless of the state of the service. It indicates which control commands are currently enabled.

The variable `CommandAdapt` is coded according to . It can be used to further restrict the enabling of control commands in `CommandInfo`.

The variable `CommandAdapt` is not part of the MTP guideline. It has been implemented additionally. This variable is not listed in the MTP file and is not made available via OPC UA.

The `FB_MTP_Procedure` is an abstract function block. This cannot be instantiated, but must be derived (see Syntax section)!

For the use of procedures, the use of MTP engineering and its automatic code generation is recommended!

Required parameters and equipment

The function block receives the process parameters `ProcParameters` required for execution as `Array[*] OF POINTER TO FB_MTP_ParameterElement`. The required equipment `RequiredEquipment` is transferred as `Array[*] OF POINTER TO I_MTP_ReqEq`.

If no `ProcParameter` or `RequiredEquipment` is required, you must transfer an empty array!

Services state machine

The states of the service state machine are executed in the methods of the same name in this function block. The methods are called by the service state machine.

Calling the procedure

The procedures can be called by the higher-level service. A possible call is described in the function block FB_MTP_ServiceControl [▶ 106].

Further characteristics

WQC [▶ 19]

Syntax

```
FUNCTION_BLOCK FB_MTP_Procedure_1_Service EXTENDS FB_MTP_Procedure
```

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
CommandInfo	DWORD	Currently enabled control commands	Read
ProcParameters	ARRAY [*] OF POINTER TO POINTER TO FB_MTP_ParameterElement	Required procedure parameters	-
RequiredEquipment	ARRAY [*] OF POINTER TO I_MTP_ReqEq	Required equipment	-
CommandAdapt	DWORD	Variable for blocking state transitions	-

Outputs

Name	Type	Description	OPC UA access
IsSelfCompleting	BOOL	1: Procedure is self-completing. 0: Procedure is completed via the control command.	Read
ProcedureID	DWORD	Unique identification of the procedure within the service	Read

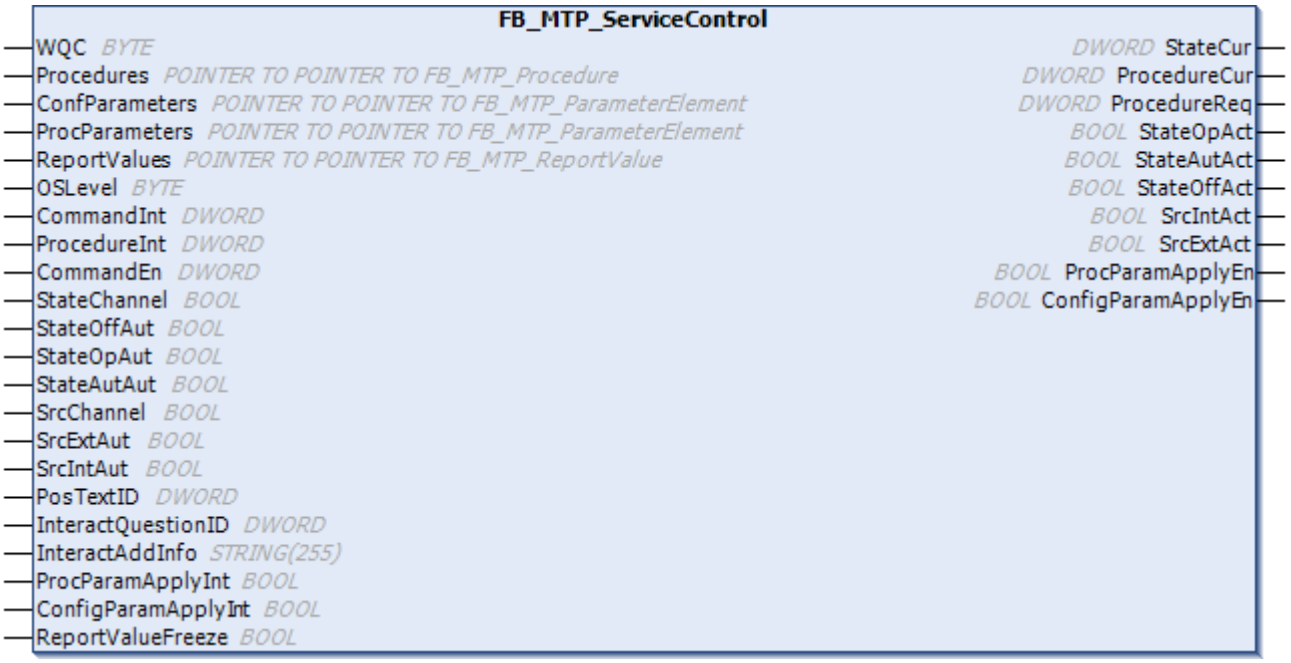
Methods

Name	Type	Description
Aborted	BOOL	Implementation of the logic for the `Aborted` state of the service state machine.
Aborting	BOOL	Implementation of the logic for the `Aborting` state of the service state machine.
Completed	BOOL	Implementation of the logic for the `Completed` state of the service state machine.

Name	Type	Description
Completing	BOOL	Implementation of the logic for the `Completing` state of the service state machine.
Execute	BOOL	Implementation of the logic for the `Execute` state of the service state machine.
Held	BOOL	Implementation of the logic for the `Held` state of the service state machine.
Holding	BOOL	Implementation of the logic for the `Holding` state of the service state machine.
Paused	BOOL	Implementation of the logic for the `Paused` state of the service state machine.
Pausing	BOOL	Implementation of the logic for the `Pausing` state of the service state machine
Resetting	BOOL	Implementation of the logic for the `Resetting` state of the service state machine.
Resuming	BOOL	Implementation of the logic for the `Resuming` state of the service state machine.
Starting	BOOL	Implementation of the logic for the `Starting` state of the service state machine.
Stopping	BOOL	Implementation of the logic for the `Stopping` state of the service state machine.
Stopped	BOOL	Implementation of the logic for the `Stopped` state of the service state machine.
Unholding	BOOL	Implementation of the logic for the `Unholding` state of the service state machine.
IsActuatorsInternal	BOOL	1: All actuators assigned to the procedure are in the state `Internal`.
IsActuatorsManual	BOOL	1: All actuators assigned to the procedure are in the state `Manual`.
IsActuatorsOffline	BOOL	1: All actuators assigned to the procedure are in the state `Offline`.
IsActuatorsOperator	BOOL	1: All actuators assigned to the procedure are in the state `Operator`.
IsActuatorsAutomatic	BOOL	1: All actuators assigned to the procedure are in the state `Automatic`.
IsReqEqInternal	BOOL	1: PID logic and all actuators assigned to the procedure are in the state `Internal`
IsReqEqManual	BOOL	1: PID logic and all actuators assigned to the procedure are in the state `Manual`
IsReqEqOffline	BOOL	1: PID logic and all actuators assigned to the procedure are in the state `Offline`
IsReqEqOperator	BOOL	1: PID logic and all actuators assigned to the procedure are in the state `Operator`
IsReqEqAutomatic	BOOL	1: PID logic and all actuators assigned to the procedure are in the state `Automatic`
SetActuatorsInternal	BOOL	All actuators assigned to the procedure are transferred to the state `Internal`.
SetActuatorsManual	BOOL	All actuators assigned to the procedure are transferred to the state `Manual`.
SetActuatorsOffline	BOOL	All actuators assigned to the procedure are transferred to the state `Offline`.
SetActuatorsOperator	BOOL	All actuators assigned to the procedure are transferred to the state `Operator`.
SetActuatorsAutomatic	BOOL	All actuators assigned to the procedure are transferred to the state `Automatic`.
SetReqEqInternal	BOOL	PID logic and all actuators assigned to the procedure are transferred to the state `Internal`.
SetReqEqManual	BOOL	PID logic and all actuators assigned to the procedure are transferred to the state `Manual`.

Name	Type	Description
SetReqEq Offline	BOOL	PID logic and all actuators assigned to the procedure are transferred to the state `Offline`.
SetReqEq Operator	BOOL	PID logic and all actuators assigned to the procedure are transferred to the state `Operator`.
SetReqEqA utomatic	BOOL	PID logic and all actuators assigned to the procedure are transferred to the state `Automatic`.

5.3.9.2 FB_MTP_ServiceControl



The function block `FB_MTP_ServiceControl` enables the execution of procedures from different sources: internal PLC logic, manual operation (e.g. via OPC UA) or control system (e.g. via OPC UA). Switching requests and value specifications are managed via Service Mode [▶ 17].

The `FB_MTP_ServiceControl` is an abstract function block. This cannot be instantiated, but must be derived (see Syntax section)!

The use of MTP engineering and its automatic code generation is recommended for the use of services!

Procedures

All procedures associated with the service are transferred as `ARRAY[*] OF POINTER TO FB_MTP_Procedure` at the input `Procedures`. A service requires at least one procedure.

A procedure is selected via the variables `ProcedureInt, ProcedureOp, ProcedurExt` depending on the state of the Service Mode [▶ 17] of the service. The currently preselected procedure ID, which is used when the service is started, is displayed via `ProcedureReq`. If the procedure ID is invalid, a 0 is output here. `ProcedureCur` shows the procedure currently started. Only one procedure can be executed at a time.

Parameter

At the input `ConfParameters`, all configuration parameters associated with the service are transferred as `ARRAY[*] OF POINTER TO FB_MTP_ParameterElement`.

At the input `ProcParameter`, all procedure parameters associated with the service are transferred as `ARRAY[*] OF POINTER TO FB_MTP_ParameterElement`.

The service-wide transfer of new values for the parameters follows the same principle as described in the sub-chapters of ParameterElements [▶ 87]. The values of all parameters assigned to the service are applied at once. The values to be applied can be checked in advance using the variable `VReq` of the parameters.

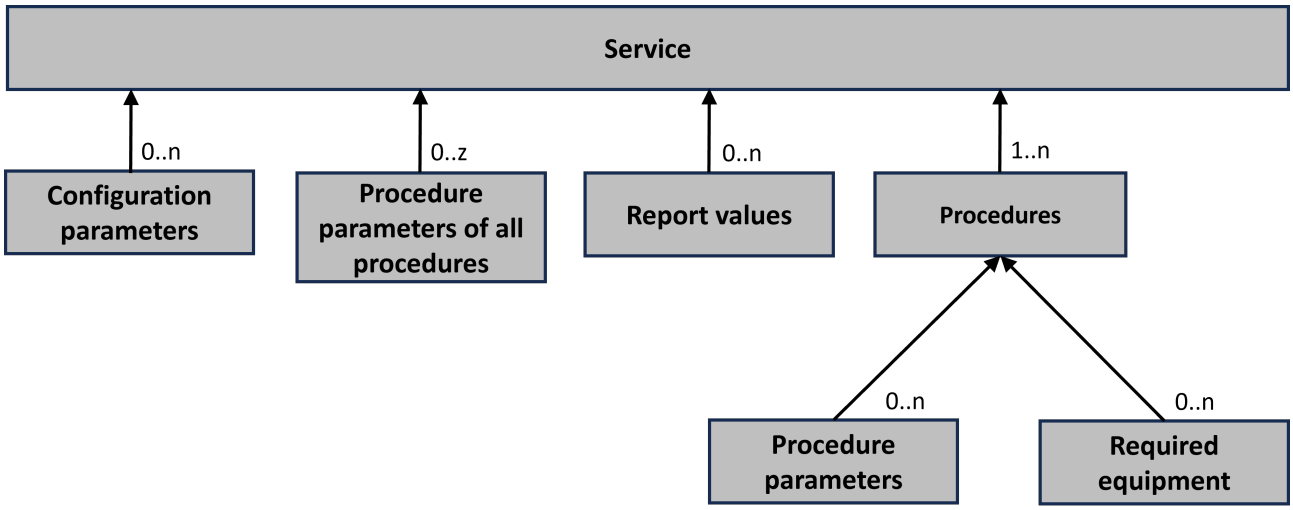
The application is enabled via the variable `ProcParameterApplyEn` for procedure parameters or via the variable `ConfigParameterApplyEn` for configuration parameters. A service-wide enable is only possible if all procedure parameters or configuration parameters are enabled `ApplyEn = TRUE`.

Depending on the state of `Service Mode`, the values are applied service-wide via the variables `ProcParameterApplyInt`, `ProcParameterApplyOp` or `ProcParameterApplyExt` for procedure parameters or via the variables `ConfigParameterApplyInt`, `ConfigParameterApplyOp` or `ConfigParameterApplyExt` for the configuration parameters if `Apply = TRUE` is set.

Report values

All report values associated with the service are applied at the input `ReportValues` as `ARRAY[*] OF POINTER TO FB_MTP_ReportValue`. All report values `V` of the service can be frozen via the input `ReportValueFreeze`. If the value of the input value `VIn` changes more than once during this time, this is displayed at the output `MissedValue = TRUE` of the parameter.

The figure below shows an overview of the arrays to be applied:

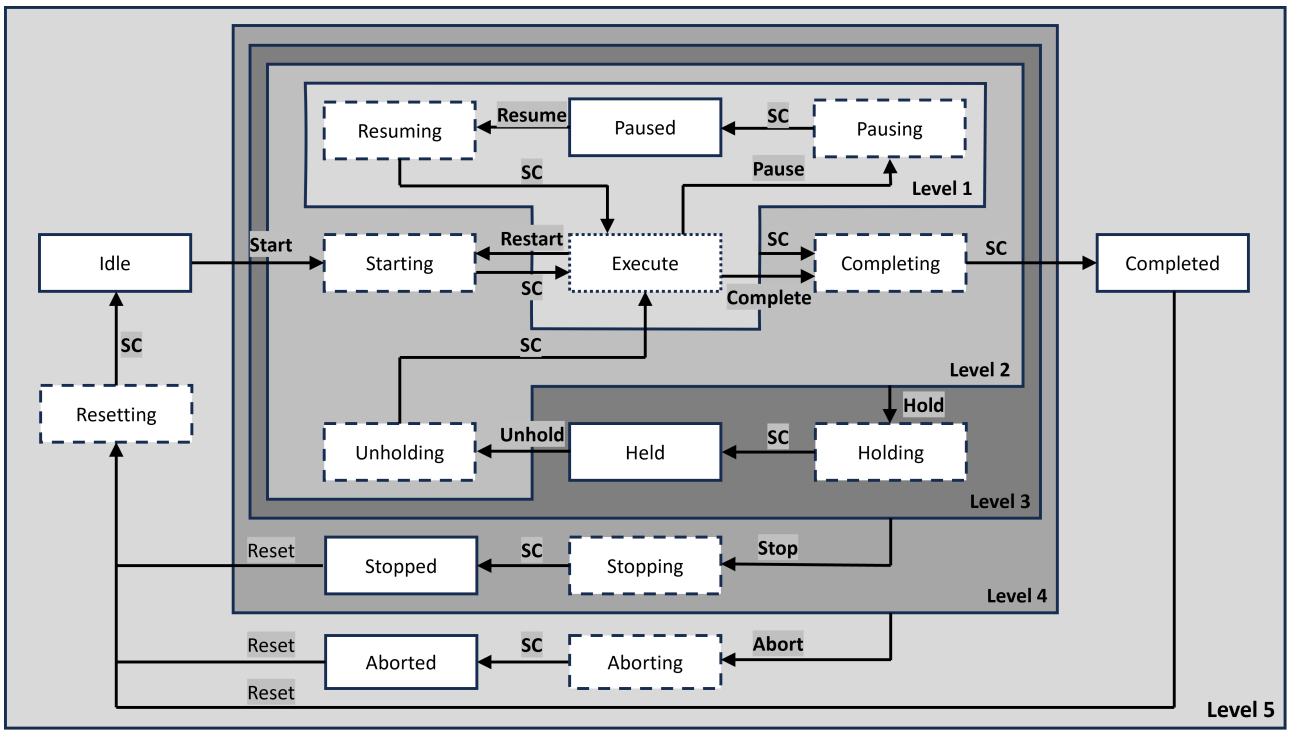


If no `ConfParameter`, `ProcParameter` or `RequiredEquipment` are required, apply an empty array.

Service state machine

The service state machine has 16 states on five levels. The states are divided into transient and non-transient states. The transient states are reached via control commands of the control word. Non-transient states are reached after the transient states (`SC`) have been successfully processed. One exception is the state `Execute`. This can be both transient and non-transient. This is defined via the variable `IsSelfCompleting` in the procedure [▶ 103].

The service state machine is structured according to the scheme below:



The service state machine can be viewed in different loops:

Main loop

The main loop for executing the main process function(s) of the machine contains the following states:

`Idle→Starting→Execute→Completing→Completed→Resetting→Idle`

The following loops can also be considered:

Pause loop

The pause loop can only be started from the state `Execute`. It is terminated by the command `Resume` or by a command that leads to a higher-level state.

Sample:

`Execute→Pausing→Paused→Resuming→Execute`

Hold loop

The hold loop can be started from the states from level 2 or lower. It is terminated by the command `Unhold` or by a command that leads to a higher-level state.

Sample:

`Execute→Holding→Held→Unholding→Execute`

In the MTP guideline, the hold loop is used for the implementation of the logic to be executed in the event of an error.

Stop loop

The stop loop can be started from the states from level 3 or lower. It is terminated with the command `Reset` or `Abort`.

`Execute→Stopping→Stopped→Resetting→Idle`

In the MTP guideline, the stop loop is used for the implementation of the logic to stop the service.

Abort loop

The abort loop can be started from the states from level 4 or lower. It is terminated with the command `Reset`.

`Execute→Aborting→Aborted→Resetting→Idle`

In the MTP guideline, the abort loop is used for the implementation of the logic to stop the service quickly.

Control of the service state machine

The variable `CommandEn` enables the individual control commands of the control word. The variables `CommandInt`, `CommandOp` and `CommandExt` can be used to specify the control word (see) for the service state machine depending on the state of the Service Mode [▶ 17].

All states can be transferred to a higher level state with a control command, provided this is enabled: e.g. all states of levels 1 and 2 can be transferred to the state `Holding` with the command `Hold`.

Execution of procedures

If the command `Start` is issued, the selected procedure is transferred to the state `Starting`. All other procedures assigned to the service are blocked.

The states, with the exception of `Idle`, are executed in the methods of the selected procedure with the same name. When executing transient states, the state transition (`SC`) occurs when the respective method returns `TRUE`. After successful execution, they enter their subsequent state.

Another procedure can only be selected and started once the service state machine is back in the state `Idle`. The current state of the service state machine is displayed via the variable `StateCur` (see).

Dependencies with Service Mode

If the `Service Mode` is transferred from the `Offline` state to one of the other two states `Operator` or `Automatic*`, the service state machine is always in the `Idle` state. It is only possible to return the `Service Mode` to the `Offline` state if the service state machine is in the basic state `Idle`. Switching between `Operator` and `Automatic*` is possible regardless of the state of the service state machine and does not change its current state.

Position description

A text list can be created to describe the current state of the service in more detail. This can be used to describe the individual work steps in more detail. The variable `PosTextID` can be used to refer to the individual work steps in the text list.

The use of MTP engineering and its MTP export is recommended for the position description, as the text list is not made available via OPC UA, but is described in the MTP file!

Service operator interaction

Interaction between the service and the operator can take place during runtime. Text lists with questions and corresponding answers are created for this purpose. The service uses the variable `QuestionID` to refer to a question in the text list. You will be shown possible answers to the question. When a response is selected, it is written to the service via the variable `InteractQuestionID`. The variable `AdditionalInfo` makes it possible to provide further information in text form.

The use of MTP engineering and its MTP export is recommended for service operator interaction, as the text lists are not made available via OPC UA, but are described in the MTP file!

In the sample below, the procedures, configuration parameters and report values are declared in the declaration part of a derivation of `FB_MTP_ServiceControl` and summarized in arrays. These are then called up in the body of the `FB_MTP_ServiceControl`. This means that all service-relevant procedures, parameters and report values are combined in one place.

Only one procedure can be executed at a time. A procedure can only be started if the service is in the `Idle` state.

Syntax

```
FUNCTION_BLOCK FB_MTP_Service EXTENDS FB_MTP_ServiceControl
VAR
///Procedures - Parameters and Required Equipment
///Procedure 1
   <ProcParametersArray1>    : ARRAY[0..<n>] OF POINTER TO FB_MTP_ParameterElement :=
[ADR(<ProcParam_0>), ADR(<ProcParam_1>)];
   <RequiredEquipmentArray1> : ARRAY[0..<n>] OF I_MTP_ReqEq := [ADR(<ReqEq_0>), ADR(<ReqEq_1>)];

///Procedure 2
   <ProcParametersArray2>    : ARRAY[0..<n>] OF POINTER TO FB_MTP_ParameterElement := [ADR(<ProcPara
m_1>), ADR(<ProcParam_2>)];

   <RequiredEquipmentArray2> : ARRAY[0..<n>] OF I_MTP_ReqEq := [ADR(<ReqEq_0>), ADR(<ReqEq_1>)];

///Procedures
   ProcedureArray
         : ARRAY[1..<n>] OF POINTER TO FB_MTP_Procedure := [ADR(<Procedure_1>), ADR(<Procedure_1>)];

///ConfigurationParameters
   ConfParameterArray      : ARRAY[0..<n>] OF POINTER TO FB_MTP_ParameterElement := [ADR(<ConfParame
ter_0>)]

///ProcedureParameters
   ProcParameterArray      : ARRAY[0..<z>] OF POINTER TO FB_MTP_ParameterElement := [ADR(<ProcParame
ter_0>), ADR(<ProcParameter_1>), ADR(<ProcParameter_2>)];

///ReportValues
   ReportValueArray        : ARRAY[0..<n>] OF POINTER TO FB_MTP_ReportValue [ADR(<ReportValue_0>)]
END_VAR

VAR_INPUT
///Procedures
   <Procedure_1>             : FB_Procedure := (<initial values>);
   <Procedure_2>             : FB_Procedure := (<initial values>);

///ConfigurationParameters
   < ConfParameter_0>        : FB_MTP_AnaServParam := (<initial values>);

///ProcedureParameters
   (<ProcParameter_0>        : FB_MTP_AnaServParam := (<initial values>);
   (<ProcParameter_1>        : FB_MTP_AnaServParam := (<initial values>);
   (<ProcParameter_2>        : FB_MTP_AnaServParam := (<initial values>);

///ReportValues
   (<ReportValue_0>: FB_MTP_AnaReportValue := ((<initial values>);
END_VAR
```

```
<Procedure_1>(
   ProcParameters    := <ProcParametersArray_1>,
   RequiredEquipment := <RequiredEquipmentArray_1>);

<Procedure_2>(
   ProcParameters    := <ProcParametersArray_2>,
   RequiredEquipment := <RequiredEquipmentArray_2>);

SUPER^(
   Procedures     := ProcedureArray,
   ConfParameters := ConfParameterArray,
   ProcParameters := ProcParameterArray,
   ReportValues   := ReportValueArray);
```

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

OSLevel [▶ 18]

Service Mode [▶ 17]

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
OSLevel	BYTE	Level must be defined for system. Value 0 is reserved for local operation.	Read/write
Procedures	ARRAY [*] OF POINTER TO FB_MTP_Procedure	Array with all procedures belonging to the service	-
ConfParameters	ARRAY [*] OF POINTER TO FB_MTP_ParameterElement	Array with all configuration parameters belonging to the service	-
ProcParameters	ARRAY [*] OF POINTER TO FB_MTP_ParameterElement	Array with all procedure parameters belonging to the service	-
ReportValues	ARRAY [*] OF POINTER TO FB_MTP_ReportValue	Array with all `ReportValues` belonging to the service	-
CommandInt	DWORD	`Internal` control word (relevant if `StateAutAct = TRUE` and `SrcIntAct = TRUE`)	Read
ProcedureInt	DWORD	`Internal` procedure preselection (relevant if `StateAutAct = TRUE` and `SrcIntAct = TRUE`)	Read
CommandEn	DWORD	Enabling control commands of the control word	Read
State* Src*		See Service Mode [▶ 17]	
PosTextID	DWORD	ID variable for the position text	Read
InteractQuestionID	DWORD	ID variable for the question text	Read
InteractAddInfo	STRING	Additional information on the current InteractQuestionID	Read
ProcParamApplyInt	BOOL	Apply all `Internal` process parameter values `VInt` belonging to the service. (Relevant, if `StateAutAct = TRUE` and `SrcIntAct = TRUE`)	Read

Name	Type	Description	OPC UA access
ConfigParamApplyInt	BOOL	Apply all `Internal` configuration parameter values `VInt` belonging to the service. (Relevant, if `StateAutAct = TRUE` and `SrcIntAct = TRUE`)	Read
ReportValueFreeze	BOOL	1: Freeze all report values of the service 0: Do not freeze the report values of the service	Read/write

Outputs

Name	Type	Description	OPC UA access
StateCur	DWORD	Current state of the service state machine	Read
ProcedureCur	DWORD	Procedure currently used	Read
ProcedureReq	DWORD	Currently preselected procedure, which is executed at start.	Read
State* Src*		See Service Mode [▶ 17]	
ProcParamApplyEn	BOOL	Enable Apply procedure parameters: 1: Parameters can be applied. 0: Parameters cannot be applied.	Read
ConfigParamApplyEn	BOOL	Enable Apply configuration parameters: 1: Parameters can be applied. 0: Parameters cannot be applied.	Read

External variables

Name	Type	Description	OPC UA access
State* Src*	BOOL	See Service Mode [▶ 17]	
CommandOp	DWORD	`Operator` control word (relevant if `StateOpAct = TRUE`)	Read/write
CommandExt	DWORD	`External` control word (relevant if `StateAutAct = TRUE` and `SrcExtAct = TRUE`)	Read/write
ProcedurOp	DWORD	`Operator` procedure preselection (relevant if `StateOpAct = TRUE`)	Read/write
ProcedureExt	DWORD	`External` procedure preselection (relevant if `StateAutAct = TRUE` and `SrcExtAct = TRUE`)	Read/write
InteractAnswerID	DWORD	ID variable for the answer text	Read/write
ProcParamApplyOp	BOOL	Apply all `Operator` process parameter values `VOp` belonging to the service. (Relevant if `StateOpAct = TRUE`)	Read/write
ProcParamApplyExt	BOOL	Apply all `External` process parameter values `VExt` belonging to the service. (Relevant, if `StateAutAct = TRUE` and `SrcExtAct = TRUE`)	Read/write
ConfigParamApplyOp	BOOL	Apply all `Operator` configuration parameter values `VOp` belonging to the service. (Relevant if `StateOpAct = TRUE`)	Read/write

Name	Type	Description	OPC UA access
ConfigParamApplyExt	BOOL	Apply all `External` configuration parameter values `VExt` belonging to the service. (Relevant, if `StateAutAct = TRUE` and `SrcExtAct = TRUE`)	Read/write

Methods

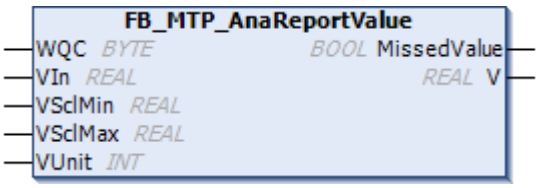
Name	Type	Description
Idle		Non-transient initial state. No process function is executed.
IsActuatorsInternal	BOOL	1: All actuators assigned to the service are in the state `Internal`.
IsActuatorsManual	BOOL	1: All actuators assigned to the service are in the state `Manual`.
IsActuatorsOffline	BOOL	1: All actuators assigned to the service are in the state `Offline`.
IsActuatorsOperator	BOOL	1: All actuators assigned to the service are in the state `Operator`.
IsActuatorsAutomatic	BOOL	1: All actuators assigned to the service are in the state `Automatic`.
IsReqEqInternal	BOOL	1: PID logic and all actuators assigned to the service are in the state `Internal`
IsReqEqManual	BOOL	1: PID logic and all actuators assigned to the service are in the state `Manual`
IsReqEqOffline	BOOL	1: PID logic and all actuators assigned to the service are in the state `Offline`
IsReqEqOperator	BOOL	1: PID logic and all actuators assigned to the service are in the state `Operator`
IsReqEqAutomatic	BOOL	1: PID logic and all actuators assigned to the service are in the state `Automatic`
SetActuatorsInternal	BOOL	All actuators assigned to the service are transferred to the state `Internal`.
SetActuatorsManual	BOOL	All actuators assigned to the service are transferred to the state `Manual`.
SetActuatorsOffline	BOOL	All actuators assigned to the service are transferred to the state `Offline`.
SetActuatorsOperator	BOOL	All actuators assigned to the service are transferred to the state `Operator`.
SetActuatorsAutomatic	BOOL	All actuators assigned to the service are transferred to the state `Automatic`.
SetReqEqInternal	BOOL	PID logic and all actuators assigned to the service are transferred to the state `Internal`.
SetReqEqManual	BOOL	PID logic and all actuators assigned to the service are transferred to the state `Manual`.
SetReqEqOffline	BOOL	PID logic and all actuators assigned to the service are transferred to the state `Offline`.
SetReqEqOperator	BOOL	PID logic and all actuators assigned to the service are transferred to the state `Operator`.
SetReqEqAutomatic	BOOL	PID logic and all actuators assigned to the service are transferred to the state `Automatic`.
SetOffline	BOOL	Transfer the service to the state `Offline`.
SetOperator	BOOL	Transfer the service to the state `Operator`.
SetAutomaticIntern	BOOL	Transfer the service to the state `AutomaticIntern`.

Name	Type	Description
SetAutomaticExtern	BOOL	Transfer the service to the state `AutomaticExtern`.
SetProcParamApplyEn	BOOL	Value specification (input variable of the method `ApplyEn`) for input variable `ApplyEn` of the procedure parameters: All service parameters assigned to the service (if `SyncOnly = FALSE`) All service parameters assigned to the service with input variable `Sync = TRUE` (if input variable of the method `SyncOnly = TRUE`)
SetConfigParamApplyEn	BOOL	Value specification (input variable method `ApplyEn`) for configuration parameters: All service parameters assigned to the service (if `SyncOnly = FALSE`) All service parameters assigned to the service with input variable `Sync = TRUE` (if input variable of the method `SyncOnly = TRUE`)
ToOnline	BOOL	Implementation of logic when leaving the Offline state.
ToOffline	BOOL	Implementation of logic when entering the Offline state.

5.3.10 Report Values

Report values are assigned to a service. To update the report values, new values must be transferred to the function block and the function block must be called manually.

5.3.10.1 FB_MTP_AnaReportValue



The function block `FB_MTP_AnaReportValue` provides an analog input value `VIn` of a manual operation (e.g. via OPC UA) for the documentation (output `V`). This value is assigned to exactly one service. Within this service, it can be assigned to one or more procedures.

Freezing the value

The value `V` can be frozen via the higher-level service (see FB_MTP_ServiceControl [▶ 106]). If the input value `VIn` changes more than once during this time, it is signaled at output `MissedValue = TRUE`.

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

Value scaling [▶ 19]

Unit [▶ 19]

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read

Name	Type	Description	OPC UA access
VIn	REAL	Input value	-
VSclMin	REAL	Scale start for value display	Read
VSclMax	REAL	Scale end for value display	Read
VUnit	INT	Unit of value	Read

Outputs

Name	Type	Description	OPC UA access
MissedValue	BOOL	Change of the input value `VIn` during the freezing of `V`: 1: The input value `VIn` has changed at least twice since freezing. 0: The input value `VIn` has not changed more than once since freezing.	Read
V	REAL	Value that is made available via OPC UA.	Read

5.3.10.2 FB_MTP_DIntReportValue

The function block `FB_MTP_DIntReportValue` provides an integer input value `VIn` of a manual operation (e.g. via OPC UA) for the documentation (output `V`). This value is assigned to exactly one service. Within this service, it can be assigned to one or more procedures.

Freezing the value

The value `V` can be frozen via the higher-level service (see FB_MTP_ServiceControl [▶ 106]). If the input value `VIn` changes more than once during this time, it is signaled at output `MissedValue = TRUE`.

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

Value scaling [▶ 19]

Unit [▶ 19]

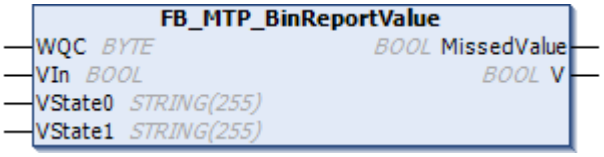
Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
VIn	DINT	Input value	-
VSclMin	DINT	Scale start for value display	Read
VSclMax	DINT	Scale end for value display	Read
VUnit	INT	Unit of value	Read

Outputs

Name	Type	Description	OPC UA access
MissedValue	BOOL	Change of the input value `VIn` during the freezing of `V`: 1: The input value `VIn` has changed at least twice since freezing. 0: The input value `VIn` has not changed more than once since freezing.	Read
V	DINT	Value that is made available via OPC UA.	Read

5.3.10.3 FB_MTP_BinReportValue



The function block `FB_MTP_BinReportValue` provides a binary input value `VIn` of a manual operation (e.g. via OPC UA) for the documentation (output `V`). This value is assigned to exactly one service. Within this service, it can be assigned to one or more procedures.

Freezing the value

The value `V` can be frozen via the higher-level service (see FB_MTP_ServiceControl [▶ 106]). If the input value `VIn` changes more than once during this time, it is signaled at output `MissedValue = TRUE`.

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
VIn	BOOL	Input value	-
VState0	STRING	Text for `VIn = FALSE`	Read
VState1	STRING	Text for `VIn = TRUE`	Read

Outputs

Name	Type	Description	OPC UA access
MissedValue	BOOL	Change of the input value `VIn` during the freezing of `V`: 1: The input value `VIn` has changed at least twice since freezing. 0: The input value `VIn` has not changed more than once since freezing.	Read
V	BOOL	Value that is made available via OPC UA.	Read

5.3.10.4 FB_MTP_StringReportValue

The function block `FB_MTP_StringReportValue` makes the input variable `TextIn` of a manual operation (e.g. via OPC UA) available for documentation (output `Text`). This value is assigned to exactly one service. Within this service, it can be assigned to one or more procedures.

Freezing the value

The value `Text` can be frozen via the higher-level service (see FB_MTP_ServiceControl [▶ 106]). If the input value `TextIn` changes more than once during this time, it is signaled at output `MissedValue = TRUE`.

Further characteristics

Name of the object [▶ 18]

Object description [▶ 18]

WQC [▶ 19]

Inputs

Name	Type	Description	OPC UA access
TagName	STRING	Name of the interface	-
TagDescription	STRING	Description of the interface	-
WQC	BYTE	Worst Quality Code	Read
TextIn	STRING	Input text	-

Outputs

Name	Type	Description	OPC UA access
MissedValue	BOOL	Change of the input value `TextIn` during the freezing of `Text`: 1: The input value `TextIn` has changed at least twice since freezing. 0: The input value `TextIn` has not changed more than once since freezing.	Read
Text	STRING	Value that is made available via OPC UA.	Read

5.4 Data types

5.4.1 ST_MTP_InputElementConfig

`ST_MTP_InputElementConfig` is a configuration for `InputElements`. It contains units with associated value representations and is transferred as an array. When transferring to `DIntProcessValueIn`, the variables `SclMinExt` and `SclMaxExt` are converted into the data types `DInt`.

Values

Name	Type	Description
UnitExt	E_MTP_Unit	Unit

Name	Type	Description
SclMinExt	REAL	Lower limit of the value representation
SclMaxExt	REAL	Upper limit of the value representation

6 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Download finder

Our download finder contains all the files that we offer you for downloading. You will find application reports, technical documentation, technical drawings, configuration files and much more.

The downloads are available in various formats.

Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on our internet page: www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline:	+49 5246 963-157
e-mail:	support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline:	+49 5246 963-460
e-mail:	service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone:	+49 5246 963-0
e-mail:	info@beckhoff.com
web:	www.beckhoff.com

Trademark statements

More Information:
www.beckhoff.com/tf8400

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com