SIEMENS

SIMATIC

Process Control System PCS 7 PCS 7 Advanced Process Library V71

Function Manual

03/2009
A5E02102723-01

Legal information

Warning notice system

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

⚠ DANGER

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

⚠ WARNING

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

⚠ CAUTION

with a safety alert symbol, indicates that minor personal injury can result if proper precautions are not taken.

CAUTION

without a safety alert symbol, indicates that property damage can result if proper precautions are not taken.

NOTICE

indicates that an unintended result or situation can occur if the corresponding information is not taken into account.

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

Qualified Personnel

The device/system may only be set up and used in conjunction with this documentation. Commissioning and operation of a device/system may only be performed by **qualified personnel**. Within the context of the safety notes in this documentation qualified persons are defined as persons who are authorized to commission, ground and label devices, systems and circuits in accordance with established safety practices and standards.

Proper use of Siemens products

Note the following:

⚠ WARNING

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

Trademarks

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

Disclaimer of Liability

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

Siemens AG
Industry Sector
Postfach 48 48
90026 NÜRNBERG
GERMANY

A5E02102723-01
℗ 01/2009

Table of contents

1 Basics of APL

1.1 Block modes

1.1.1 Overview of the modes

Overview of the individual modes

The available operating modes are assigned to the block families:

- Motors, valves and dosers
- Controllers
- Blocks without "Manual" and "Automatic" modes

You can find an overview below. Click on one of the operating modes to go directly to the relevant detailed description.

You can find a status diagram for the operating modes (Page 39) at the end of this section.

Operating modes for motors, valves and dosers

The following operating modes are available:

1. Local mode (Page 36)
2. Automatic mode (Page 32)
3. Manual mode (Page 32)
4. Out of service (Page 27)

The mode with the lowest number in the list above has the highest priority. "Manual" and "Automatic" modes have the same priority. General information on the individual modes is available in the following sections. The sections also include block-specific information, for example, non-standard parameter assignment. Refer to the description and function of the relevant blocks.

Operating modes for controllers

The following operating modes are available:

1. Automatic mode (Page 29)
2. Manual mode (Page 29)
3. Program mode for closed-loop controllers (Page 34)
4. Out of service (Page 27)

The mode with the lowest number has highest priority. "Manual" and "Automatic" modes have the same priority. General information on the individual modes is available in the following sections. The sections also include block-specific information, for example, non-standard parameter assignment. Refer to the description and function of the relevant blocks.

Operating modes for blocks without "Manual" and "Automatic" operation

The following operating modes are available:

1. On (Page 27)
2. Out of service (Page 27)

The mode with the lowest number has highest priority. General information on the individual modes is available in the following sections. The sections also include block-specific information, for example, non-standard parameter assignment. Refer to the description and function of the relevant blocks.

Note

Please note that the operating modes are realized differently in the individual block families.

1.1.2 On

"On" operating mode

The "On" operating mode tells you that the block algorithm is being processed (output parameter `OnAct = 1`). This mode is only present in the case of blocks that have faceplates, but not the following, which have operating modes:

- Manual mode or
- Automatic mode or
- Local mode

The "On" mode can only be activated via a control on the faceplate (input parameter `OnOp = 1`). The block must be in the "Out-of-service" operating mode for this to be possible.

1.1.3 Out of service

Using the "out of service" operating mode

The "Out of service" operating mode is available to all blocks that have an operating mode changeover and a direct connection to the process (with a connection to a process tag, for example).

It is intended for maintenance and service purposes (to replace the device, for example), as all messages and functions of the block are switched off. The only function still possible is an operating mode changeover.

All outputs for motors and valves are set to the safe position in this operating mode.

For controllers, the safety manipulated variable (high or low manual manipulated variable) is only used if the `Feature` bit Safety value of the manipulated variable effective at startup (Page 196) is active. Otherwise the manipulated variable remains at the latest value like all the other output parameters.

Refer also to the Safe position for motors, valves and controllers (Page 120) section for more on this.

The last value available is output permanently for all other blocks.

Requirement for the "out of service" mode

Prerequisite for switching to this operating mode is that the block is in "Manual mode" or "On" mode.

Switching on the out of service operating mode by using the faceplate

The "Out of service" operating mode can only be switched on by using the faceplate when it is in the standard block view (`OosOp = 1` parameter) and even then, only if `ModLiOp = 0`.

To change over the operating mode by using the faceplate, please refer to the descriptions relating to the standard view of the individual blocks.

Switching on the Out of service operating mode by using the interconnection

The "Out of service" operating mode is switched on by using the configurable parameter `OosLi = 1`. This is only possible if the block was previously in manual mode or "on" mode and the `Feature` bit Reaction to the out of service mode (Page 199) was set to 1.

Regardless of operating mode, the parameter view of the faceplate will always display the status of the parameter `OosLi =1` with the symbol for the status "In progress" (see table) next to the Maintenance enable button.

Display	Meaning
	In progress

Refer also to the Maintenance release (Page 47) section for more on this.

Exiting the "Out of service" operating mode

From this operating mode, a block can only be switched by an operator action at the faceplate into the following operating modes:

- "On"
- "Manual mode"

1.1.4 Manual and Automatic mode for control blocks

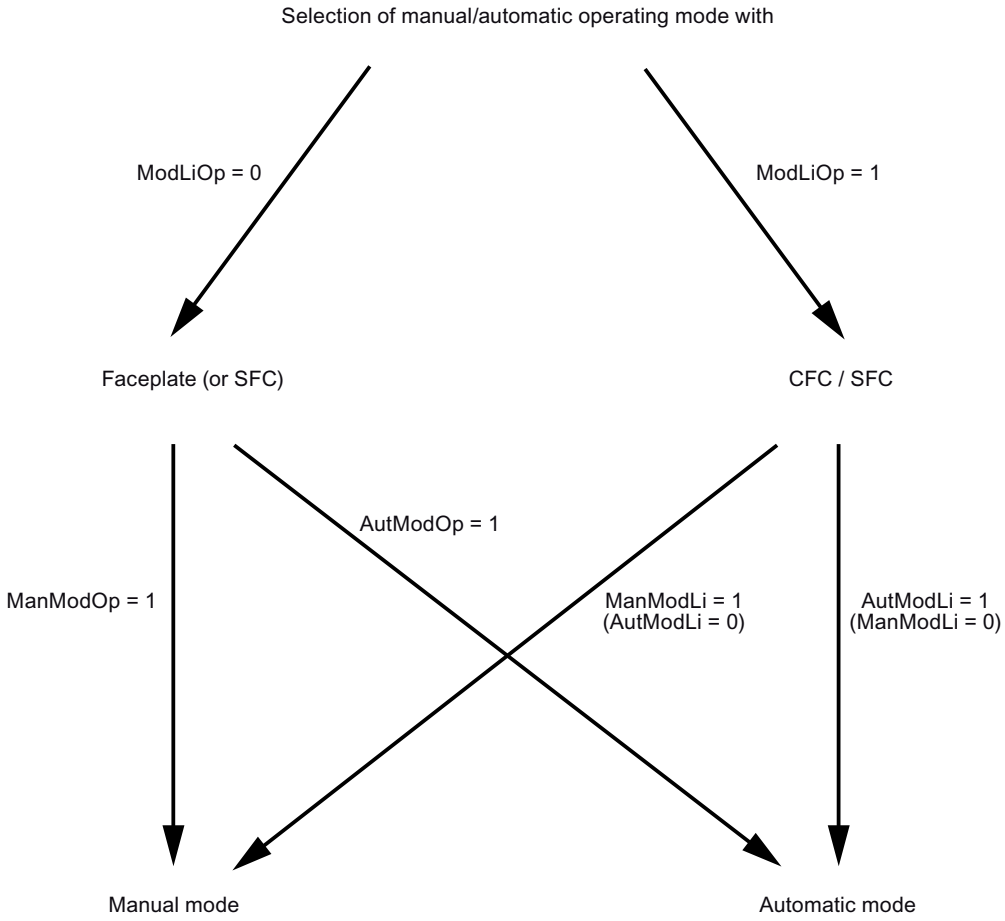
"Manual" and "Automatic" modes for control blocks

In "manual mode", the control settings for the device are made manually by the operator. The operator decides how to change the block's manipulated variable (output signal). The manipulated variable can be analog or binary.

In "automatic mode", the control settings for the controller are made automatically as calculated by the block algorithm.

Changing between operating modes

The switchover between manual and automatic modes takes place as shown in the following schematic:



Switchover initiated in the faceplate (`ModLiOp` = 0)**:** The changeover between operating modes is carried out in the standard view of the faceplate. In the function block, the parameters `ManModOp` for "manual mode" and `AutModOp` for "automatic mode" are used.

If both signals (`ManModOp =1, AutModOp = 1`) are set, `ManModOp = 1` has priority.

Switchover per interconnection (CFC or SFC instance) (`ModLiOp = 1`)**:** The switchover between the operating modes is carried out with an interconnection on the function block. The parameters `ManModLi` for "manual mode" and `AutModLi` for "automatic mode" are used in pushbutton operation. In switch mode (requirement: `Feature Bit 4 = 1`, see Setting switch or button mode (Page 195)) connection `ManModLi` is used exclusively.

If both signals (`ManModLi =1, AutModLi = 1`) are set, `ManModLi = 1` has priority.

Note

You can access the variable parameters `AutModOp` and `ManModOp` from a normal SFC (in contrast to the instance of an SFC type). The SFC can thus change the operating mode without revoking the access rights of the operator (i.e. without setting `ModLiOp = 1`).

Switchover from automatic mode to manual mode

When changing over from "automatic mode" to "manual mode", the last valid control settings (**M**anipiulated **V**alue `MV`) for the controller set in "automatic mode" remain valid until you change the control settings manually.

Switchover from automatic mode to manual mode

The switchover from manual to automatic mode can take place with or without the internal setpoint tracking the process value. You specify this behavior on the SP_TrkPV I/O, which can also be operated from the faceplate in the parameter view (Option "SP = PV"). For the blocks PIDConL and PIDStepL you can also change the behavior for the switchover via the parameter Feature bit Disabling bumpless changeover to automatic mode for controllers (Page 197):

- **Switchover with internal setpoint tracking process variable** (SP_TrkPV = 1) means that in "Manual" mode the setpoint (SP) tracks the process variable (PV) (bumpless switchover). After switching back to "Automatic" mode, the manipulated variable remains constant until the setpoint value (SP) is changed or the process value (PV) changes.
- **Switchover without internal setpoint tracking process variable** (SP_TrkPV = 0) means that the block immediately recalculates the value of the manipulated variable based on the setpoint and process value (PV) when the mode is changed. The Feature parameter is used to choose between the two variants:
  - **Switchover without P step** (standard setting, Feature bit = 0):

    During switchover, the I action of the controller is set in such a way that the switchover is carried out without a P step (virtually bumpless referring to the manipulated variable). A control deviation is only regulated via the I action.
  - **Switchover with P step** (Feature bit = 1):

    During switchover, the I action of the controller is set in such a way that the switchover is carried out with a P step (not bumpless referring to the manipulated variable). A control deviation is regulated via the P and the I action.

Note

Points to note about switchovers with a P step change:

- The P action must be active for the setting "Switchover with P step" (PropSel = 1)
- If the P action is in the feedback (PropFacSP = 0), the "Switchover with P step" setting has no effect.
- If the switchover function with the internal setpoint tracking the process variable is active (SP_TrkPV = 1), the "Switchover with P step" setting has no effect.

Reaction of signals when operating mode is changed

Using the Feature bit Resetting the commands for changing the mode (Page 193), you can choose whether the block automatically resets the signal for changing the operating mode.

Switch on program mode

A few control blocks allow you to operate in program mode. Please consult the section on functions of the individual control blocks to find out whether a control block allows program mode.

Please also refer to the section Program mode for closed-loop controllers (Page 34) for information on program mode.

1.1.5 Manual and automatic mode for motors, valves and dosers

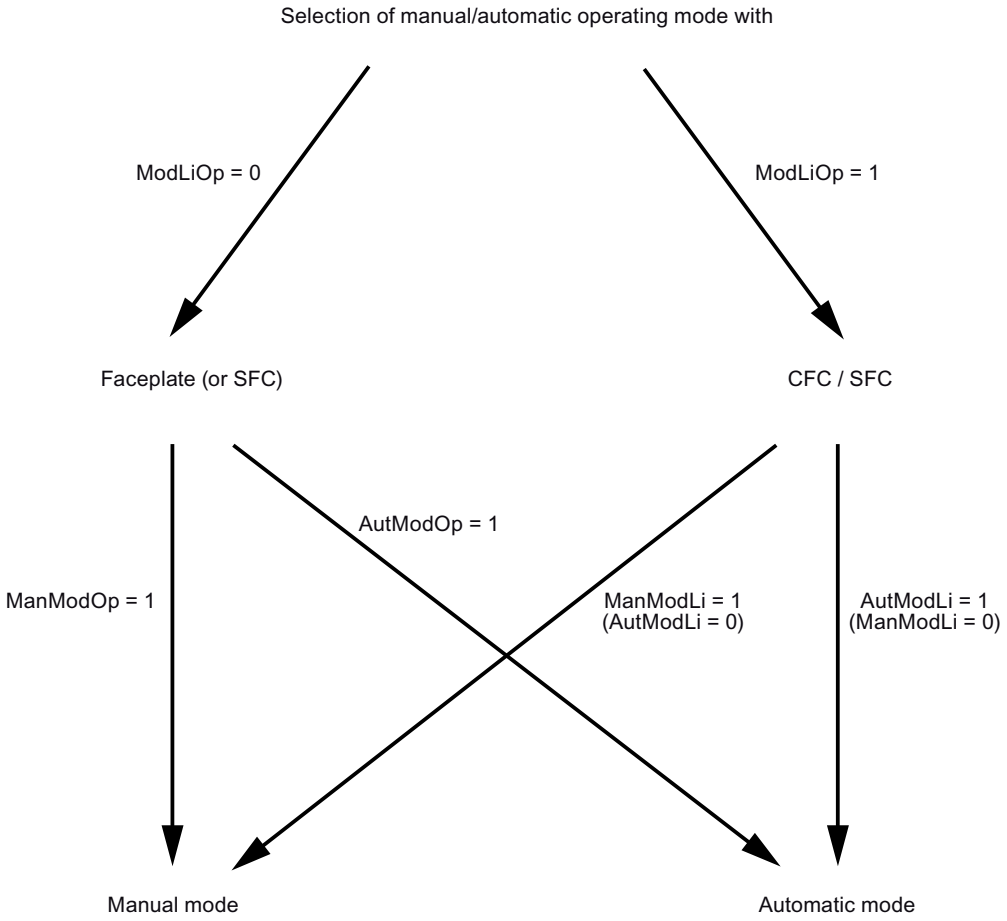
Manual and automatic mode for motors, valves and dosers

In "manual mode", the control settings for the device are made manually by the operator. The operator decides how to change the block's manipulated variable (output signal). The manipulated variable can be analog or binary in accordance with the function block.

In "automatic mode", the control settings for the device are made by the block algorithm via interconnected inputs or inputs controlled by SFC.

Changing between operating modes

The switchover between "manual and automatic mode" takes place as shown in the following schematic:



Switchover using faceplates (`ModLiOp = 0`)**:** The changeover between operating modes is carried out in the standard view of the faceplate. In the function block, the parameters `ManModOp` for "manual mode" and `AutModOp` for "automatic mode" are used.

Switchover per interconnection (CFC or SFC instance) (`ModLiOp = 1`)**:** The switchover between the operating modes is carried out with an interconnection on the function block. The parameters `ManModLi` for "manual mode" and `AutModLi` for "automatic mode" are used in pushbutton operation. In switch mode (requirement: `Feature Bit 4 = 1`, see Setting switch or button mode (Page 195)) connection `AutModLi` is used exclusively.

Note

You can access the variable parameters `AutModOp` and `ManModOp` from a normal SFC (in contrast to the instance of an SFC type). The SFC can thus change the operating mode without revoking the access rights of the operator (i.e. without setting `ModLiOp = 1`).

Switchover from automatic mode to manual mode

When changing over from "automatic mode" to "manual mode", the last valid control settings for the block set in "automatic mode" remain valid until you change the control settings manually.

Switchover from manual to automatic mode

You can set the following options for changing over from "manual mode" to "automatic mode" using the `Feature` bit Bumpless switchover to automatic mode (Page 197). Refer to the I/O descriptions for the relevant block.

- A switchover from manual to automatic mode is possible at any time (standard setting, `Feature` bit = 0). The control settings for the automatic mode become effective immediately.
- Switchover from manual to automatic mode is only possible if the control settings for the manual and automatic modes match (bumpless switchover), (`Feature` bit = 1). An error message is output if they do not match. In this case, you will need to adapt the control settings in "manual mode" to the control settings in "automatic mode".

Reaction of signals when operating mode is changed

Using the `Feature` bit Resetting the commands for changing the mode (Page 193), you can choose whether the block automatically resets the signal for changing the operating mode.

Resetting the commands for the control settings

With the `Feature` bit Enabling resetting of commands for the control settings (Page 193), you select how the block handles commands for the control settings (for example motor on) via the interconnected input parameters.

1.1.6 Program mode for closed-loop controllers

Program mode for closed-loop controllers - interface for higher-level control functions

The interface for primary controller functions (external Advanced Control software package) provides primary controller functions, which run on an external PC as an OPC client, the option of using the control from the controller function block and specifying the setpoint or manipulated variable from a remote location. This procedure is called program mode.

You can use the feature bit Enabling program mode (Page 192) to specify whether or not the controller block is intended for program mode.

Program mode requires an enable signal (input parameter `AdvCoEn = 1`) from a central control block. If this enable signal goes from 1 to 0, for example, due to errors in the OPC communication, the controller block returns to the operating mode it had before program mode.

You activate program mode in the standard view of the controller faceplate. In addition to switching from manual to automatic mode, you are also given the option of using program mode as the operating mode. You exit program mode by operator input or by switching back into manual or automatic mode.

A 0-1 edge transition of the interconnectable input parameter `AdvCoMstrOn` activates program mode depending on the conditions described below. You can use this to put an entire group of downstream controller blocks into program mode at the same time from a central control block. Both the input parameter `AdvCoOn` and the interconnectable input parameter `AdvCoMstrOn` can be used at the same time, since the parameter `AdvCoMstrOn` only reacts to edges of the binary signal.

Program mode is deactivated with a 1 - 0 edge transition.

The output parameter `AdvCoRdy = 1` indicates if the PID controller is ready to switch to program mode. At a central control block, you can use an AND operation for all `AdvCoRdy` signals of the downstream controllers to enable central switchover.

The output parameter `AdvCoAct = 1` indicates of the block is in program mode.

Selecting the type of program mode

There are two types of program mode:

- Program mode with setpoint (in automatic mode only)
- Program mode with setpoint (in manual mode only, not for step controllers without position feedback)

Program mode with setpoint: If you set the input parameter `AdvCoModSP = 1`, the analog value provided by the OPC client (`AdvCoMV`) is used as an external setpoint for the controller. The controller and faceplate otherwise react as they do with automatic mode and an external setpoint. Refer to section Setpoint input - internal and external (Page 96) for more about this.

Requirements for program mode with setpoint:

- `AdvCoModSP = 1`,
- `AdvCoEn = 1`,
- The controller is in automatic mode.

Program mode with manipulated value: If you set the input parameter `AdvCoModSP = 0`, the analog value provided by the OPC client (`AdvCoMV`) is used as an external manipulated value for the controller. The algorithm of the PID controller is bypassed. The controller and faceplate otherwise react as they do with tracking (`MV_TrkOn = 1`). Refer to section Tracking and limiting a manipulated value (Page 111) for more about this.

Requirements for program mode with manipulated value:

- `AdvCoModSP = 0`,
- `AdvCoEn = 1`,
- The controller is in manual mode.

Note

Program mode with setpoint is not available for step controllers without position feedback (available in PIDStepL, FmCont and FmTemp). `ErrorNum = 50` is output on the control block and the controller cannot switch into program mode (`AdvCoAct=0`).

1.1.7 Local mode

Areas of application for local mode

This operating mode is used for motors, valves and dosing units. The control settings are made directly or via a control station that is located "locally". In addition, you can set different control strategies with the parameter `LocalSetting`.

With `LocalSetting = 0`, you prevent a change to "local mode".

Changing to local mode

Changing to local mode is only possible from the manual and automatic operating modes. The change to this mode is initiated by:

- An operation on the faceplate (input parameter `LocalOp = 1`, valid if `LocalSetting = 3` or `LocalSetting = 4` and `ModLiOp = 0`) or
- The interconnected input parameter (`LocalLi = 1`, valid if `LocalSetting = 1` or `LocalSetting = 2`).

Exiting local mode

You leave local mode using:

- An operation on the faceplate (`LocalSetting = 3` or `LocalSetting = 4` and `ModLiOp = 0`) or
- The interconnected input parameter (`LocalSetting = 1` or `LocalSetting = 2`).

  In order to exit local mode via the interconnected input parameter, you can configure various reactions using a `Feature` bit Exiting local mode (Page 199).

Operator input in "local mode" using a faceplate

You are not permitted to functionally operate the block in local mode. You can only use the faceplate to exit local mode if you have also activated "local mode" using the faceplate. The rules you specified for exiting "local mode" apply here.

Input in "local mode" via interconnected inputs

In "local mode", the way the block functions is influenced via interconnected input parameters according to the settings of the `LocalSetting` parameter. You have the following options:

- `LocalSetting = 1` and `LocalSetting = 3`
  - The control settings for the block are adjusted (tracking) via an interconnected input parameter. The interconnected input parameter includes the control signal for the local operating station on the system.
  - The runtime monitoring of the block is effective in accordance with your configuration.
  - The configured interlocking functions of the block are activated in accordance with input parameter `BypProt = 0` or deactivated (`BypProt = 1`).
- `LocalSetting = 2` and `LocalSetting = 4`
  - The control settings for the block are made based on internal adjustment of the feedback value.
  - The configured runtime monitoring of the block is deactivated.

    Note: The configured runtime monitoring is deactivated for motor feedback of the `VlvMotL` motor valve. Opening and closing of the motor value continues to be monitored.
  - The configured interlock functions of the block are disabled.

Overview of behavior in local mode

LocalSetting =	0	1	2	3	4
Switch on operating mode	Cannot be set	CFC/SFC	CFC/SFC	Faceplate	Faceplate
Changing the operating mode: Local mode/to manual mode only (`Feature = 0`)	-	CFC/SFC	CFC/SFC	-	-
Changing the operating mode: Local mode/previous mode (`Feature = 1`)	-	CFC/SFC	CFC/SFC	-	-
Operating in the faceplate	-	Not supported	Not supported	Change the operating mode only	Change the operating mode only
Tracking via an external input	-	Yes	No	Yes	No
Reaction of the block	-	Monitoring the feedback value	Adjustment of the feedback value	Monitoring the feedback value	Adjustment of the feedback value
Interlock activated	-	Yes: (`BypProt = 0`) No: (`BypProt = 1`)	No	Yes: (`BypProt = 0`) No: (`BypProt = 1`)	No

1.1.8 Status diagram for the operating modes

Status diagram for the operating modes

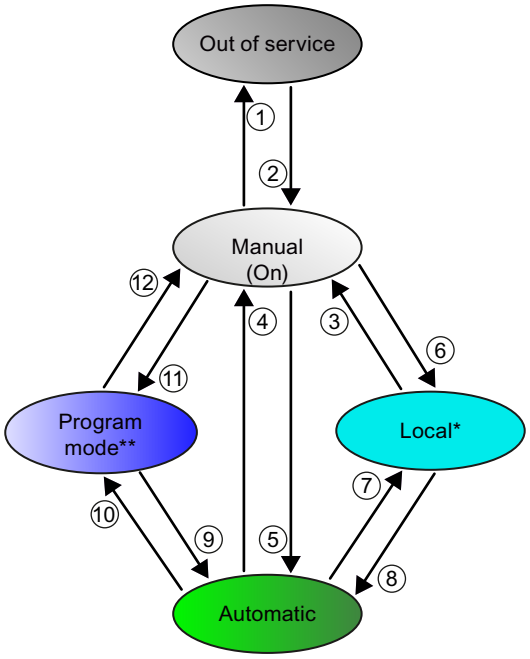


Figure 1-1 Status diagram for the operating modes

* This operating mode is used for motors, valves, and dosing units.

** This operating mode is used for controllers only.

Number in graphic (top)	Condition for status change
(1)	**Manual (on) → Out of service** • Via faceplate (`OosOp = 1`) if `ModLiOp = 0` or • Via edge transition 0 → 1 of `OosLi` if Feature bit Reaction to the out of service mode (Page 199) = 1
(2)	**Out of service → Manual (on)** • Via faceplate (`ManModOp = 1`)
(3)	**Local mode → Manual** • Via faceplate (`ManModOp = 1`) if `ModLiOp = 0` and `LocalSetting = 3` or `LocalSetting = 4` or • Via `LocalLi = 0` if `LocalSetting = 1` or `LocalSetting = 2`. See section Exiting local mode (Page 199) for more conditions.
(4)	**Automatic → Manual** • Via faceplate (`ManModOp = 1`) if `ModLiOp = 0` or • Via `ManModLi = 1` if `ModLiOp = 1` and Feature bit Setting switch or button mode (Page 195) = `0` or • Via `AutModLi = 0` if `ModLiOp = 1` and Feature bit Setting switch or button mode (Page 195) = `1`
(5)	**Manual → Automatic** • Via faceplate (`AutModOp = 1`) if `ModLiOp = 0` or • Via `AutModLi = 1` if `ModLiOp = 1`
(6)	**Manual → Local mode** • Via faceplate (`LocalOp = 1`) if `ModLiOp = 0` and `LocalSetting = 3` or `LocalSetting = 4` or • Via `LocalLi = 1` if `LocalSetting = 1` or `LocalSetting = 2`
(7)	**Automatic → Local mode** • Via faceplate (`LocalOp = 1`) if `ModLiOp = 0` and `LocalSetting = 3` or `LocalSetting = 4` or • Via `LocalLi = 1` if `LocalSetting = 1` or `LocalSetting = 2`
(8)	**Local mode → Automatic** • Via faceplate (`AutModOp = 1`) if `ModLiOp = 0` and `LocalSetting = 3` or `LocalSetting = 4` or • Via `LocalLi = 0` if `LocalSetting = 1` or `LocalSetting = 2`. See section Exiting local mode (Page 199) for more conditions.
(9)	**Program mode → Automatic** • Via faceplate (`AutModOp = 1`) if `ModLiOp = 0` or • Via `AutModLi = 1` if `ModLiOp = 1` or • Via edge transition 1 → 0 of `AdvCoMstrOn` if automatic is set before program mode.
(10)	**Automatic → Program mode** Requirement for changeover in program mode: `AdvCoEn = 1` • Via faceplate (`AdvCoOn = 1`) if `ModLiOp = 0` or • Via `AdvCoMstrOn = 1`

Number in graphic (top)	Condition for status change
(11)	**Manual → Program mode** Requirement for changeover from manual to program mode: `AdvCoEn = 1` and `AdvCoModSP = 0` • Via faceplate (`AdvCoOn = 1`) if `ModLiOp = 0` or • Via `AdvCoMstrOn = 1`
(12)	**Program mode → Manual** • Via faceplate (`ManModOp = 1`) if `ModLiOp = 0` or • Edge transition 1 → 0 of `AdvCoMstrOn` if manual is set before program mode.

1.2 General functions of the blocks

1.2.1 Operating, monitoring and reporting

1.2.1.1 Display and operator input area for process values and setpoints

Display and control field for process values and setpoints

You specify the upper and lower area limits in the faceplate using interconnectable input parameters for the following:

- Display areas (bar display)
- Operator input area (for example setpoint and manipulated value)
- Input range for limits (maximum 7 numbers possible in the faceplates)

The interconnectable input parameter is a structured variable that contains two analog values. Please refer to the descriptions of the individual blocks for the relevant input parameters.

Using a structured variable for scaling

There are three possibilities with which you can influence the contents of the structured variables, namely with:

- The corresponding channel function block, for example, the FbAnIn block
- A conversion block, for example, the StruScIn block
- Direct parameter assignment at the block input

Additional information on data types is available in the documentation of CFC and STEP 7.

1.2.1.2 Calling further faceplates

Opening additional faceplates

You can open standard views of other faceplates from various faceplate views. Here, you have the following options:

- Two buttons that you can assign freely and that are used to call faceplates of other blocks.
- Two predefined buttons for calling faceplates with a fixed assignment to the controller blocks.
- Buttons predefined for interlock functions

Freely assignable buttons

From the standard view and from the preview, you can use a button to open the standard view of a block that can be selected freely. In order to use this function, in the CFC you need to interconnect the `SelFp1` input parameter for the button in the standard view or `SelFp2` for the button in the preview to any given output parameter of the block whose faceplate is to be opened. This makes the buttons in the faceplates visible.

Button label

You can change button labels as follows:

- Open the process picture in WinCC GraphicsDesigner.
- Open the object properties of the block icon.
- Under Configurations, assign the desired text to the attribute UserButtonText1 or UserButtonText2.

Predefined buttons for controller blocks

You can open the standard view for the following blocks from a controller standard view or parameter view (for example PIDConL):

- ConPerMon (can be called from the standard view)
  - To do this, you need to interconnect the output parameter `CPI` of the ConPerMon block to the input parameter `CPI_In` of the controller block.
- GainSched (can be called from the parameter view)
  - To do this, you need to interconnect the output parameter `Link2Gain` of the GainSched block to the input parameter `Gain` of the controller block.

The labels of the buttons cannot be changed here.

Predefined buttons for interlocks

You can open the following interlock blocks from the standard view of the technological blocks:

- Activation enable
- Interlock without reset (interlock)
- Interlock with reset (protection)

The buttons intended for this are visible when the relevant input parameter (`Permit`, `Intlock` or `Protect`) is interconnected to an interlock block.

You can open the standard view for the following faceplates from the standard view of the interlock blocks:

- The blocks interconnected to the input values.
- The block interconnected to the output value.

The buttons intended for this are visible when the input parameters (for example `In01`) or the `Out` output parameter of the interlock block is interconnected to a block that has a faceplate.

Note

Interconnection of the `Out` output parameter to multiple blocks is not permitted. The reason for this is that a direct relationship must be established between the button in the faceplate and the faceplate to be opened by it.

1.2.1.3 Operator permissions

Operator control permissions for blocks

Four conditions apply to operator control permissions:

1. User administration in the PCS 7 OS:
   The following operator control permissions are used in APL:
   - Process controlling (for example manual /automatic mode changeover, changing setpoints and manipulated variables).

     With this operator control permission, operations can be performed in the standard view of all blocks and input can be made in the memo view.
   - Higher process controlling (for example, changing limits, controller parameters, and monitoring times).

     With this operator control permission, all operations in all views of all blocks are possible, with the exception of those listed under "Highest process controlling".
   - Highest process controlling (simulate process values and release of process tag for maintenance).

     With this operator control permission, simulation can be switched on and off in the parameter view and the process tag for maintenance work can be released.

   Exceptions to the operator control permissions described above are listed in the descriptions of the individual views.
2. You can execute various functions depending on the block mode. These permissions are stored in the block algorithm and are determined dynamically in online mode.
3. The block is either controlled by the operator or by the controller, depending on parameter settings or on the interconnection. Manual mode can be changed to automatic mode by the operator or by a master control system, for example. These permissions are stored in the block algorithm and are determined dynamically in online mode.
4. Controllable blocks have the `OS_Perm` input parameter which allows you to implement individual operator control strategies by setting the operator control permissions. A pressure relief valve, for example, can only be opened by the master control system. The operator may only close the valve. These authorizations are defined during configuration. These operator control permissions are displayed in the preview view of the faceplate. For information about setting the individual operator control permissions (`OS_Perm`) please refer to the description of the functions of the individual blocks.

   The relevant operator controls are enabled if the operator has suitable permissions. The block algorithm processes the input data.

1.2.1.4 Generating instance-specific messages

Generating instance-specific messages

- You can generate instance-specific messages for a binary signal of every block.

The number of interconnectable input parameters that can be used freely varies with relation to the blocks. The X in the parameter name designates the position.

You can specify the following messages for these instance-specific messages:

- Message class
- Priority of the message
- Message text
- Message auxiliary value
- Acknowledge behavior

Additional information is available in the descriptions of the message functionality of the individual blocks and in the PCS 7 Configuration Manual Operator Station under "How to configure the user-specific messages".

See also

Time stamp (Page 51)

1.2.1.5 Maintenance release

Issuing a maintenance release

The maintenance release serves as information about a process tag at which maintenance, service or calibration should be carried out. You can use the signal for maintenance release to transfer the information about the enabling of a process tag from the OS to a Maintenance Station.

Note

The block must be in either "Manual," "On" or "Out of service" mode to set the maintenance release.

You set the maintenance release (operator control permission "System control" required) in the parameter view using the input parameter `MS_RelOp = 1`. A maintenance release is then made available via the interconnectable output parameter `MS_Release = 1` for further processing. In order to make this information of the Maintenance Station available, you have to interconnect the output parameter `MS_Release` of the technological block with the input parameter `MS_Releaseof` the corresponding channel block.

The issuing of a maintenance release does not have any influence on the function of the block. An operation message is generated.

Use of the state "In progress" on the Maintenance Station

The status "In progress" is implemented on the Maintenance Station for a process tag or a field device using the channel blocks and the interconnectable output parameter `OosAct = 1`. You can interconnect the output parameter `OosAct` of the channel block with the input parameter `OosLi` of a technological block.

Use the `Feature` bit Reaction to the out of service mode (Page 199) to specify, in case the input parameter is `OosLi = 1`, if:

- there is a changeover to the "out of service" mode and the symbol for the "In progress" (see table) status is displayed. You can change to manual mode at any time.
- only the display "In progress" (see table) in the block icon and in the faceplate of the assigned technological block is to be carried out.

Display	Meaning
	In progress

Function sequence in the APL

- The OS operator issues the maintenance release (`MS_RelOp = 1`) in the technological block's parameter view.
- The technological block then sets the `MS_Release = 1` output parameter.
- The channel driver's `MS_Rel` input is now also 1.
- The channel driver signals the maintenance release to the diagnostic driver via the `DXCHG` parameter.
- The maintenance release is only signaled to the Maintenance Station once all 0 bits of the parameter `DXCHG_XX` are set on the diagnostic driver.
- The channel driver determines the "in progress" state of the Maintenance Station using the `MS` input parameter and makes this information available at the `OosAct` output parameter.

- On the technological block, the "working" state is displayed at input parameter `OosLi` and forwarded for display to the faceplate.

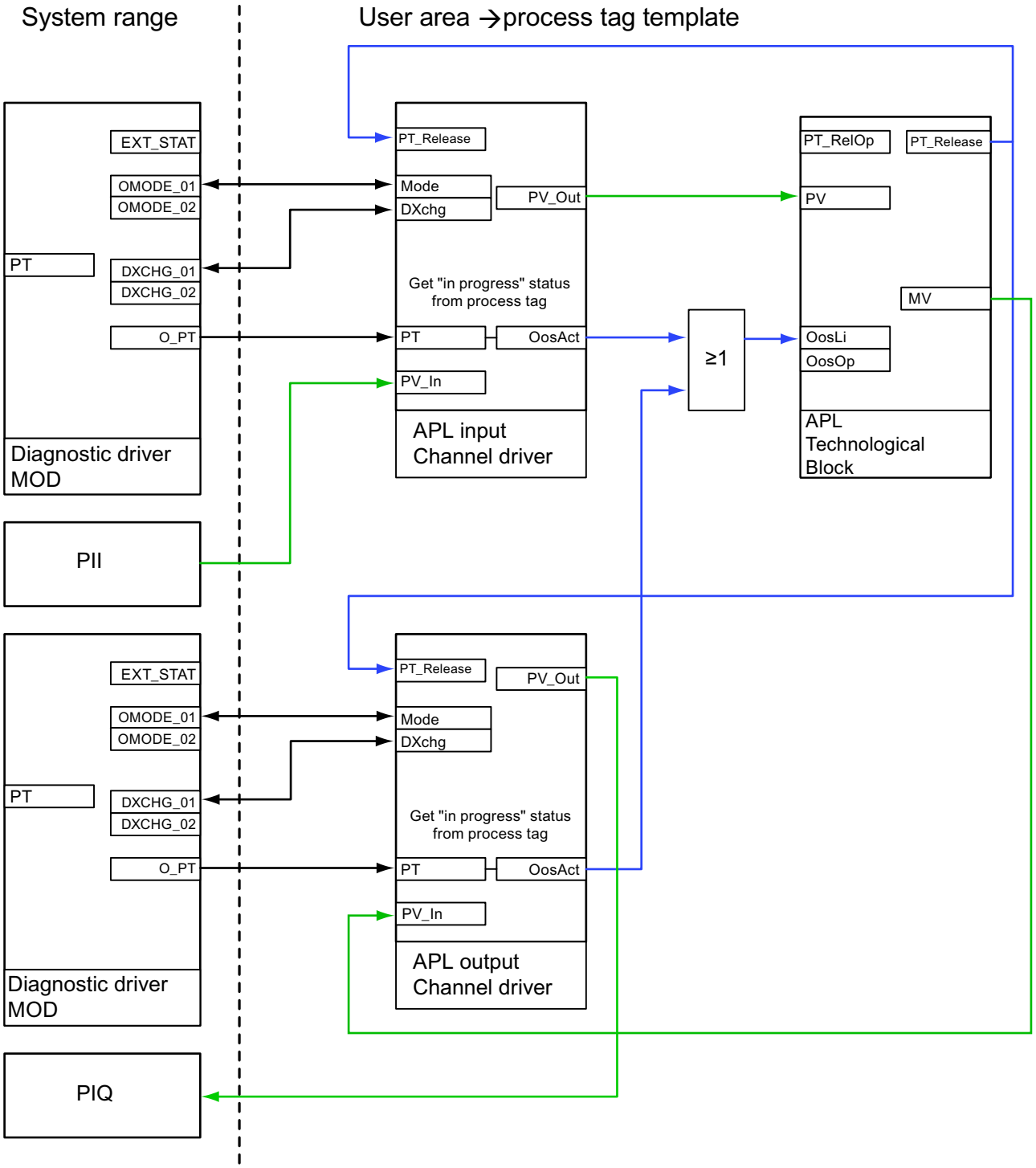


Figure 1-2 Maintenance release

Key to diagram:

PII	Process image of inputs
PIQ	Process image of outputs
Black lines	Automatic system connections
Green lines	Process value connections made by the planner
Blue lines	Connections for maintenance release made by the planner

Note

For additional information on the topic of maintenance please refer to PCS 7 OS process management.

1.2.1.6 Time stamp

Time stamp

The time stamp is the assignment of time information to the status change of a binary process signal. The status change of the signal is signaled together with the time information.

Use the EventTS block to report time stamped signals.

For more information on time stamping and how to configure it, please refer to the "PCS 7 - High-Precision Time Stamping Function Manual".

Areas of application

Areas of application of the time stamp are for example:

- Accurately-timed detection of problems in process-related equipment. The time stamp enables you to explicitly identify signals that indicate the cause of the failure of a process unit.
- Analysis of system-wide interrelationships
- Detection and reporting of the sequence of time-critical signal changes

Forming the time information

The time information is generated by one of the following methods and is specified at the block by means of the input parameter `TimeStampOn`:

- `TimeStampOn = 0`: Time stamping is carried out in the EventTS block (default when installed in the chart)
- `TimeStampOn = 1`: The high-precision time stamp is carried out in the process I/O suitable to this purpose.

Time stamping in the EventTS block

Connect the binary output parameter of another block (e.g. Pcs7DiIn) with a message input Inx (x = 1 ... 8) of the EventTS block.

When the EventTS block recognizes a change in the signal state at this message input, it uses the current time of the CPU as the time stamp. Only the signal changes that are slower than the cycle time of the block can be detected.

High-precision time stamp in the process I/O

You have configured the hardware of your system for high-precision time stamping as explained in the "PCS 7 - High-Precision Time Stamping Function Manual". The signal changes are recognized in the I/O devices and the time stamp assigned to them. This data is available at the output parameter `TS_Out` of the Pcs7DiIT block.

The high-precision time stamp is independent of the cycle time of the blocks. The actual time resolution for two different status changes depends on your plant configuration and the hardware you are using.

Interconnect the output parameter `TS_Out` of Pcs7DiIT with a message input `InTSx(x = 1 ... 8)` of the EventTS block.

Error handling

The system block ImDrvTs recognizes when the time stamp function in the I/O devices is defective and forwards this information to the Pcs7DiIT block. This then forms the time stamp using the current CPU time and sets the signal status of `TS_Out` output parameter to "Bad, due to device". The EventTS block then uses the current time of the CPU as the time stamp. You can find additional information in the "PCS 7 - High-Precision Time Stamping Function Manual".

1.2.1.7 Displaying auxiliary values

Displaying auxiliary values

Up to two auxiliary values can be displayed in the standard view of some faceplates. This feature can be used, for example, with motors to indicate the motor current and winding temperature.

To do so interconnect the value that you want to have displayed with the input parameters `UserAna1` or `UserAna2`.

In the object properties (I/Os > Identifier) of the block in CFC, you can specify the text to be displayed for these parameters in the standard view of the faceplate.

1.2.1.8 Selecting a unit of measure

Coded unit of measure

The parameter XXX_Unit is used to specify the unit of measure for the corresponding input parameter (XXX stands for a specific parameter, for example, PV_Unit). Entry is carried out in the form of a code. Exactly one unit of measure is assigned to each code and is displayed on the faceplate.
You can interconnect the XXX_Unit input parameter of a technological block with the XXXUnit output parameter of an analog input driver. At the analog input driver, enter the unit of measure at the XXXUnit input parameter (XXX stands for a specific parameter, for example PV_InUnit, PVOutUnit).

Using the unit of measure with controllers for the ConPerMon block

For control blocks the current unit of measure is output via output parameter XX_UnitOut . If you use the ConPerMon block, you must switch this output parameter with the corresponding input parameter XXX_Unit on the ConPerMon block.

Using the S7_unit attribute

If you set the xxx_Unit parameter to 0, the entry is displayed by the S7_Unit attribute in the faceplate and in the block icon.

Overview of the units of measure

The units of measure are listed in the following tables:

Short list of the units of measure most commonly used in accordance with the specifications "PA-Profile for Process Device" of "Profibus and Profinet International"

Value	Display	Description
1000	K	Kelvin
1001	°C	Degrees Celsius
1002	°F	Degrees Fahrenheit
1005	°	Degree
1006	'	Minute
1007	"	Second
1010	m	Meter
1013	mm	Millimeter
1018	ft	Foot
1023	m^2	Square meter
1038	L	Liter
1041	hl	Hectoliter
1054	s	Second
1058	min	Minute
1059	h	Hour
1060	d	Day

Value	Display	Description
1061	m/s	Meters per second
1077	Hz	Hertz
1081	kHz	Kilohertz
1082	1/s	Per second
1083	1/min	Per minute
1088	kg	Kilogram
1092	t	Metric ton
1100	g/cm^3	Grams per cubic centimeter
1105	g/L	Grams per liter
1120	N	Newton
1123	mN	Millinewton
1130	Pa	Pascal
1133	kPa	Kilopascal
1137	bar	Bar
1138	mbar	Millibar
1149	mmH2O	Millimeters of water
1175	W·h	Watt hour
1179	kW·h	Kilowatt hour
1181	$kcal_{th}$	Kilocalorie
1190	kW	Kilowatt
1209	A	Ampere
1211	mA	Milliampere
1221	A·h	Ampere hour
1240	V	Volt
1349	m^3/h	Cubic meters per hour
1353	L/h	Liters per hour
1384	mol	Mole
1422	pH	pH value

Listing of all the units of measure in accordance with the specifications "PA-Profile for Process Device" of "Profibus and Profinet International"

Value	Display	Description
1000	K	Kelvin
1001	°C	Degrees Celsius
1002	°F	Degrees Fahrenheit
1003	°R	Degree Rankine
1004	rad	Radian
1005	°	Degree
1006	'	Minute
1007	"	Second
1008	gon	Gon
1009	r	Revolution
1010	m	Meter
1011	km	Kilometer
1012	cm	Centimeter
1013	mm	Millimeter
1014	µm	Micrometer
1015	nm	Nanometer
1016	pm	Picometer
1017	Å	angstrom
1018	ft	Foot
1019	in	Inch
1020	yd	Yard
1021	mile	Mile
1022	nautical mile	Nautical mile
1023	m^2	Square meter
1024	km^2	Square kilometer
1025	cm^2	Square centimeter
1026	dm^2	Square decimeter
1027	mm^2	Square millimeter
1028	a	Are
1029	ha	Hectare
1030	in^2	Square inch
1031	ft^2	Square foot
1032	yd^2	Square yard
1033	$mile^2$	Square mile
1034	m^3	Cubic meter
1035	dm^3	Cubic decimeter
1036	cm^3	Cubic centimeter
1037	mm^3	Cubic millimeter
1038	L	Liter

Value	Display	Description
1039	cl	Centiliter
1040	ml	Milliliter
1041	hl	Hectoliter
1042	in^3	Cubic inch
1043	ft^3	Cubic foot
1044	yd^3	Cubic yard
1045	$mile^3$	Cubic mile
1046	pint	Pint
1047	quart	Quart
1048	gal	US gallon
1049	ImpGal	Imperial gallon
1050	bushel	Bushel
1051	bbl	Barrel = 42 gallons
1052	bbl(liq)	Liquid barrel = 31.5 gallons
1053	ft^3	Standard cubic foot
1054	s	Second
1055	ks	Kilosecond
1056	ms	Millisecond
1057	µs	Microsecond
1058	min	Minute
1059	h	Hour
1060	d	Day
1061	m/s	Meters per second
1062	mm/s	Millimeters per second
1063	m/h	Meters per hour
1064	km/h	Kilometers per hour
1065	knot	Knot
1066	in/s	Inches per second
1067	ft/s	Feet per second
1068	yd/s	Yards per second
1069	in/min	Inches per minute
1070	ft/min	Feet per minute
1071	yd/min	Yards per minute
1072	in/h	Inches per hour
1073	ft/h	Feet per hour
1074	yd/h	Yards per hour
1075	mi/h	Miles per hour
1076	m/s^2	Meter/second squared
1077	Hz	Hertz
1078	THz	Terahertz
1079	GHz	Gigahertz
1080	MHz	Megahertz

Value	Display	Description
1081	kHz	Kilohertz
1082	1/s	Per second
1083	1/min	Per minute
1084	rad/s	Revolutions per second
1085	r/min	Revolutions per minute
1086	rad/s	Radians per second
1087	$1/s^2$	Per second squared
1088	kg	Kilogram
1089	g	Gram
1090	mg	Milligram
1091	Mg	Megagram
1092	t	Metric ton
1093	oz	Ounce
1094	lb	Pound
1095	STon	US ton (short ton)
1096	LTon	British ton (long ton)
1097	kg/m^3	Kilograms per cubic meter
1098	Mg/dm^3	Megagrams per cubic meter
1099	kg/dm^3	Kilograms per cubic decimeter
1100	g/cm^3	Grams per cubic centimeter
1101	g/m^3	Grams per cubic meter
1102	t/m^3	Metric tons per cubic meter
1103	kg/L	Kilogram per liter
1104	g/ml	Grams per milliliter
1105	g/L	Grams per liter
1106	lb/in^3	Pounds per cubic inch
1107	lb/ft^3	Pounds per cubic foot
1108	lb/gal	Pounds per US gallon
1109	$STon/yd^3$	US tons per cubic yard
1110	°Twd	Degree Twaddell
1111	°Baum (hv)	Degree Baumé (heavy)
1112	°Baum (lt)	Degree Baumé (light)
1113	°API	Degrees API
1114	SGU	Specific gravity units
1115	kg/m	Kilograms per meter
1116	mg/m	Milligrams per meter
1117	tex	Tex
1118	$kg \cdot m^2$	Kilograms per square meter
1119	kg·m/s	Kilograms per meter per second
1120	N	Newton
1121	MN	Meganewton
1122	kN	Kilonewton

Value	Display	Description
1123	mN	Millinewton
1124	µN	Micronewton
1125	kg·m²/s	Kilograms per square meter per second
1126	N·m	Newton meter
1127	MN·m	Meganewton meter
1128	kN·m	Kilonewton meter
1129	mN·m	Millinewton meter
1130	Pa	Pascal
1131	GPa	Gigapascal
1132	MPa	Megapascal
1133	kPa	Kilopascal
1134	mPa	Millipascal
1135	µPa	Micropascal
1136	hPa	Hectopascal
1137	bar	Bar
1138	mbar	Millibar
1139	torr	Torr
1140	atm	Atmosphere
1141	psi	Pounds per square inch
1142	psia	Pounds per square inch (absolute)
1143	psig	Pounds per square inch (gauge)
1144	g/cm²	Grams per square centimeter
1145	kg/cm²	Kilograms per square centimeter
1146	inH2O	Inches of water
1147	inH2O (4°C)	Inches of water at 4 degrees Celsius
1148	inH2O (68°F)	Inches of water at 68 degrees Fahrenheit
1149	mmH2O	Millimeters of water
1150	mmH2O (4°C)	Millimeters of water at 4 degrees Celsius
1151	mmH2O (68°F)	Millimeters of water at 68 degrees Fahrenheit
1152	ftH2O	Feet of water
1153	ftH2O (4°C)	Feet of water at 4 degrees Celsius
1154	ftH2O (68°F)	Feet of water at 68 degrees Fahrenheit
1155	inHg	Inches of mercury
1156	inHg (0°C)	Inches of mercury at 0 degrees Celsius
1157	mmHg	Millimeters of mercury
1158	mmHg (0°C)	Millimeters of mercury at 0 degrees Celsius
1159	Pa·s	Pascal second
1160	m²/s	Square meters per second
1161	P	Poise
1162	cP	Centipoise
1163	St	Stokes

Value	Display	Description
1164	cSt	Centistokes
1165	N/m	Newtons per meter
1166	mN/m	Millinewtons per meter
1167	J	Joule
1168	EJ	Exajoule
1169	PJ	Petajoule
1170	TJ	Terajoule
1171	GJ	Gigajoule
1172	MJ	Megajoule
1173	kJ	Kilojoule
1174	mJ	Millijoule
1175	W·h	Watt hour
1176	TW·h	Terawatt hour
1177	GW·h	Gigawatt hour
1178	MW·h	Megawatt hour
1179	kW·h	Kilowatt hour
1180	cal_{th}	Calorie (thermochemical)
1181	$kcal_{th}$	Kilocalorie (thermochemical)
1182	$Mcal_{th}$	Megacalorie (thermochemical)
1183	Btu_{th}	British thermal unit
1184	datherm	Decatherm
1185	ft·lb	Foot pound
1186	W	Watt
1187	TW	Terawatt
1188	GW	Gigawatt
1189	MW	Megawatt
1190	kW	Kilowatt
1191	mW	Milliwatt
1192	µW	Microwatt
1193	nW	Nanowatt
1194	pW	Picowatt
1195	$Mcal_{th}/h$	Megacalorie per hour
1196	MJ/h	Megajoule per hour
1197	Btu_{th}/h	British thermal units per hour
1198	hp	Horsepower
1199	W/(m·K)	Watts per meter kelvin
1200	$W/(m^2·K)$	Watts per (square meter kelvin)
1201	$m^2·K/W$	Square meters kelvin per Watt
1202	J/K	Joules per kelvin
1203	kJ/K	Kilojoules per kelvin
1204	J/(kg·K)	Joules per (kilogram kelvin)
1205	kJ/(kg·K)	Kilojoules per (kilogram kelvin)

Value	Display	Description
1206	J/kg	Joules per kilogram
1207	MJ/kg	Megajoules per kilogram
1208	kJ/kg	Kilojoules per kilogramm
1209	A	Ampere
1210	kA	Kiloampere
1211	mA	Milliampere
1212	µA	Microampere
1213	nA	Nanoampere
1214	pA	Picoampere
1215	C	Coulomb
1216	MC	Megacoulomb
1217	kC	Kilocoulomb
1218	µC	Microcoulomb
1219	nC	Nanocoulomb
1220	pC	Picocoulomb
1221	A·h	Ampere hour
1222	C/m^3	Coulombs per cubic meter
1223	C/mm^3	Coulombs per cubic millimeter
1224	C/cm^3	Coulombs per cubic centimeter
1225	kC/m^3	Kilocoulombs per cubic meter
1226	mC/m^3	Millicoulombs per cubic meter
1227	$µC/m^3$	Microcoulombs per cubic meter
1228	C/m^2	Coulombs per square meter
1229	C/mm^2	Coulombs per square millimeter
1230	C/cm^2	Coulombs per square centimeter
1231	kC/m^2	Kilocoulombs per square meter
1232	mC/m^2	Millicoulombs per square meter
1233	$µC/m^2$	Microcoulombs per square meter
1234	V/m	Volts per meter
1235	MV/m	Megavolts per meter
1236	kV/m	Kilovolts per meter
1237	V/cm	Volts per centimeter
1238	mV/m	Millivolts per meter
1239	µV/m	Microvolts per meter
1240	V	Volt
1241	MV	Megavolt
1242	KV	Kilovolt
1243	mV	Millivolt
1244	µV	Microvolt
1245	F	Farad
1246	mF	Millifarad
1247	µF	Microfarad

Value	Display	Description
1248	nF	Nanofarad
1249	pF	Picofarad
1250	F/m	Farad per meter
1251	µF/m	Microfarad per meter
1252	nF/m	Nanofarad per meter
1253	pF/m	Picofarad per meter
1254	C·m	Coulomb meter
1255	A/m^2	Amperes per square meter
1256	MA/m^2	Megaamperes per square meter
1257	A/cm^2	Amperes per square centimeter
1258	kA/m^2	Kiloamperes per square meter
1259	A/m	Amperes per meter
1260	kA/m	Kiloamperes per meter
1261	A/cm	Amperes per centimeter
1262	T	Tesla
1263	mT	Millitesla
1264	µT	Microtesla
1265	nT	Nanotesla
1266	Wb	Weber
1267	mWb	Milliweber
1268	Wb/m	Webers per meter
1269	kWb/m	Kilowebers per meter
1270	H	Henry
1271	mH	Millihenry
1272	µH	Microhenry
1273	nH	Nanohenry
1274	pH	Picohenry
1275	H/m	Henries per meter
1276	µH/m	Microhenries per meter
1277	nH/m	Nanohenries per meter
1278	$A·m^2$	Ampere square meters
1279	$N·m^2/A$	Newton meter squared per ampere
1280	Wb·m	Weber meter
1281	Ω	Ohm
1282	GΩ	Gigaohm
1283	MΩ	Megaohm
1284	kΩ	Kiloohm
1285	mΩ	Milliohm
1286	µΩ	Microohm
1287	S	Siemens
1288	kS	Kilosiemens
1289	mS	Millisiemens

Value	Display	Description
1290	µS	Microsiemens
1291	Ω·m	Ohm meter
1292	GΩ·m	Gigaohm meter
1293	MΩ·m	Megaohm meter
1294	kΩ·m	Kiloohm meter
1295	Ω·cm	Ohm centimeter
1296	mΩ·m	Milliohm meter
1297	µΩ·m	Microohm meter
1298	nΩ·m	Nanoohm meter
1299	S/m	Siemens per meter
1300	MS/m	Megasiemens per meter
1301	kS/m	Kilosiemens per meter
1302	mS/cm	Millisiemens per centimeter
1303	µS/mm	Microsiemens per millimeter
1304	1/H	Per henry
1305	sr	Steradian
1306	W/sr	Watts per steradian
1307	W/(sr·m^2)	Watts per (steradian square meter)
1308	W/(m^2)	Watts per square meter
1309	lm	Lumen
1310	lm·s	Lumen second
1311	lm·h	Lumen hour
1312	lm/m^2	Lumens per square meter
1313	lm/W	Lumens per watt
1314	lx	Lux
1315	lx·s	Lux second
1316	cd	Candela
1317	cd/m^2	Candela per square meter
1318	g/s	Grams per second
1319	g/min	Grams per minute
1320	g/h	Grams per hour
1321	g/d	Grams per day
1322	kg/s	Kilograms per second
1323	kg/min	Kilograms per minute
1324	kg/h	Kilograms per hour
1325	kg/d	Kilograms per day
1326	t/s	Metric tons per second
1327	t/min	Metric tons per minute
1328	t/h	Metric tons per hour
1329	t/d	Metric tons per day
1330	lb/s	Pounds per second
1331	lb/min	Pounds per minute

Value	Display	Description
1332	lb/h	Pounds per hour
1333	lb/d	Pounds per day
1334	STon/s	US tons per second
1335	STon/min	US tons per minute
1336	STon/h	US tons per hour
1337	STon/d	US tons per day
1338	LTon/s	British tons per second
1339	LTon/min	British tons per minute
1340	LTon/h	British tons per hour
1341	LTon/d	British tons per day
1342	%	Percent
1343	% sol/wt	Percentage solids per weight unit
1344	% sol/vol	Percentage solids per volume unit
1345	% stm qual	Percentage steam quality
1346	°Plato	Degree plato
1347	m^3/s	Cubic meters per second
1348	m^3/min	Cubic meters per minute
1349	m^3/h	Cubic meters per hour
1350	m^3/d	Cubic meters per day
1351	L/s	Liters per second
1352	L/min	Liters per minute
1353	L/h	Liters per hour
1354	L/d	Liters per day
1355	ML/d	Megaliters per day
1356	ft^3/s	Cubic feet per second
1357	ft^3/m	Cubic feet per minute
1358	ft^3/h	Cubic feet per hour
1359	ft^3/d	Cubic feet per day
1360	ft^3/min std	Standard cubic feet per minute
1361	ft^3/h std	Standard cubic feet per hour
1362	gal/s	US gallons per second
1363	gal/min	US gallons per minute
1364	gal/h	US gallons per hour
1365	gal/d	US gallons per day
1366	Mgal/d	Mega US gallons per day
1367	ImpGal/s	Imperial gallons per second
1368	ImpGal/min	Imperial gallons per minute
1369	ImpGal/h	Imperial gallons per hour
1370	ImpGal/d	Imperial gallons per day
1371	bbl/s	Barrels per second
1372	bbl/min	Barrels per minute
1373	bbl/h	Barrels per hour

Value	Display	Description
1374	bbl/d	Barrels per day
1375	W/m^2	Watts per square meter
1376	mW/m^2	Milliwatts per square meter
1377	$\mu W/m^2$	Microwatts per square meter
1378	pW/m^2	Picowatts per square meter
1379	$Pa \cdot s/m^3$	Pascal seconds per cubic meter
1380	N·s/m	Newton seconds per meter
1381	Pa·s/m	Pascal seconds per meter
1382	B	Bel
1383	dB	Decibel
1384	mol	Mole
1385	kmol	Kilomole
1386	mmol	Millimole
1387	µmol	Micromole
1388	kg/mol	Kilograms per mole
1389	g/mol	Grams per mole
1390	m^3/mol	Cubic meters per mole
1391	dm^3/mol	Cubic decimeters per mole
1392	cm^3/mol	Cubic centimeters per mole
1393	L/mol	Liters per mole
1394	J/mol	Joules per mole
1395	kJ/mol	Kilojoules per mole
1396	J/(mol·K)	Joules per mole kelvin
1397	mol/m^3	Moles per cubic meter
1398	mol/dm^3	Moles per cubic decimeter
1399	mol/L	Moles per liter
1400	mol/kg	Moles per kilogram
1401	mmol/kg	Millimoles per kilogram
1402	Bq	Becquerel
1403	MBq	Megabecquerel
1404	kBq	Kilobecquerel
1405	Bq/kg	Becquerels per kilogram
1406	kBq/kg	Kilobecquerels per kilogram
1407	MBq/kg	Megabecquerels per kilogram
1408	Gy	Gray
1409	mGy	Milligray
1410	rd	Rad
1411	Sv	Sievert
1412	mSv	Millisievert
1413	rem	Rem
1414	C/kg	Coulombs per kilogram
1415	mC/kg	Millicoulombs per kilogram

Value	Display	Description
1416	R	Röntgen
1417	1/Jm^3	Density of magnetic energy
1418	e/Vm^3	
1419	m^3/C	Cubic meters per coulomb
1420	V/K	Volts per kelvin
1421	mV/K	Millivolts per kelvin
1422	pH	pH value
1423	ppm	Parts per million
1424	ppb	Parts per billion
1425	ppth	Parts per trillion
1426	°Brix	Degrees Brix
1427	°Ball	Degrees Balling
1428	proof/vol	Proof per volume
1429	proof/mass	Proof per mass
1430	lb/ImpGal	Pounds per Imperial gallon
1431	kcal$_{th}$/s	Kilocalories per second
1432	kcal$_{th}$/min	Kilocalories per minute
1433	kcalt$_h$/h	Kilocalories per hour
1434	kcal$_{th}$/d	Kilocalories per day
1435	Mcal$_{th}$/s	Megacalorie per second
1436	Mcal$_{th}$/min	Megacalories per minute
1437	Mcal$_{th}$/d	Megacalories per day
1438	kJ/s	Kilojoules per second
1439	kJ/min	Kilojoules per minute
1440	kJ/h	Kilojoule per hour
1441	kJ/d	Kilojoules per day
1442	MJ/s	Megajoules per second
1443	MJ/min	Megajoules per minute
1444	MJ/d	Megajoules per day
1445	Btu$_{th}$/s	British thermal units per second
1446	Btu$_{th}$/min	British thermal units per minute
1447	Btu$_{th}$/d	British thermal units per day
1448	µgal/s	Micro US gallons per second
1449	mgal/s	Milli US gallons per second
1450	kgal/s	Kilo US gallons per second
1451	Mgal/s	Mega US gallons per second
1452	µgal/min	Micro US gallons per minute
1453	mgal/min	Milli US gallons per minute
1454	kgal/min	Kilo US gallons per minute
1455	Mgal/min	Mega US gallons per minute
1456	µgal/h	Micro US gallons per hour
1457	mgal/h	Milli US gallons per hour

Value	Display	Description
1458	kgal/h	Kilo US gallons per hour
1459	Mgal/h	Mega US gallons per hour
1460	µgal/d	Micro US gallons per day
1461	mgal/d	Milli US gallons per day
1462	kgal/d	Kilo US gallons per day
1463	µImpGal/s	Micro Imperial gallons per second
1464	mImpGal/s	Milli Imperial gallons per second
1465	kImpGal/s	Kilo Imperial gallons per second
1466	MImpGal/s	Mega Imperial gallons per second
1467	µImpGal/min	Micro Imperial gallons per minute
1468	mImpGal/min	Milli Imperial gallons per minute
1469	kImpGal/min	Kilo Imperial gallons per minute
1470	MImpGal/min	Mega Imperial gallons per minute
1471	µImpGal/h	Micro Imperial gallons per hour
1472	mImpGal/h	Milli Imperial gallons per hour
1473	kImpGal/h	Kilo Imperial gallons per hour
1474	MImpGal/h	Mega Imperial gallons per hour
1475	µImpgal/d	Micro Imperial gallons per day
1476	mImpgal/d	Milli Imperial gallons per day
1477	kImpgal/d	Kilo Imperial gallons per day
1478	MImpgal/d	Mega Imperial gallons per day
1479	µbbl/s	Microbarrels per second
1480	mbbl/s	Millibarrels per second
1481	kbbl/s	Kilobarrels per second
1482	Mbbl/s	Megabarrels per second
1483	µbbl/min	Microbarrels per minute
1484	mbbl/min	Millibarrels per minute
1485	kbbl/min	Kilobarrels per minute
1486	Mbbl/min	Megabarrels per minute
1487	µbbl/h	Microbarrels per hour
1488	mbbl/h	Millibarrels per hour
1489	kbbl/h	Kilobarrels per hour
1490	Mbbl/h	Megabarrels per hour
1491	µbbl/d	Microbarrels per day
1492	mbbl/d	Millibarrels per day
1493	kbbl/d	Kilobarrels per day
1494	Mbbl/d	Megabarrels per day
1495	$\mu m^3/s$	Cubic micrometers per second
1496	mm^3/s	Cubic millimeters per second
1497	km^3/s	Cubic kilometers per second
1498	Mm^3/s	Cubic megameters per second
1499	$\mu m^3/min$	Cubic micrometers per minute

Value	Display	Description
1500	mm^3/min	Cubic millimeters per minute
1501	km^3/min	Cubic kilometers per minute
1502	mm^3/min	Cubic megameters per minute
1503	$\mu m^3/h$	Cubic micrometers per minute
1504	mm^3/h	Cubic millimeters per minute
1505	km^3/h	Cubic kilometers per minute
1506	Mm^3/h	Cubic megameters per minute
1507	$\mu m^3/d$	Cubic micrometers per day
1508	mm^3/d	Cubic millimeters per day
1509	km^3/d	Cubic kilometers per day
1510	Mm^3/d	Cubic megameters per day
1511	cm^3/s	Cubic centimeters per second
1512	cm^3/min	Cubic centimeters per minute
1513	cm^3/h	Cubic centimeters per hour
1514	cm^3/d	Cubic centimeters per day
1515	$kcal_{th}/kg$	Kilocalories per kilogram
1516	Btu_{th}/lb	British thermal units per pound
1517	kl	Kiloliter
1518	kl/min	Kiloliters per minute
1519	kl/h	Kiloliters per hour
1520	kl/d	Kiloliters per day
1551	S/cm	Siemens per centimeter
1552	µS/cm	Microsiemens per centimeter
1553	mS/m	Millisiemens per meter
1554	µS/m	Microsiemens per meter
1555	MΩ · cm	Megaohm centimeter
1556	kΩ · cm	Kiloohm centimeter
1557	Weight%	Weight percent
1558	mg/L	Milligram per liter
1559	µg/L	Microgram per liter
1560	%Sat	-
1561	vpm	-
1562	%vol	Volume percent
1563	ml/min	Milliliters per minute
1564	mg/dm^3	Milligrams per cubic centimeter
1565	mg/L	Milligram per liter
1566	mg/m^3	Milligrams per cubic meter
1567	ct	Carat (jewels) = $200.0 \cdot 10^{-6}$ kg
1568	lb (tr)	Pound (troy or apothecary) = 0.3732417216 kg
1569	oz (tr)	Ounce (troy or apothecary) = 1/12 lb (tr)
1570	fl oz (U.S.)	Ounce (U.S. fluid) = (1/128) gal

Value	Display	Description
1571	cm^3	Cubic centimeter = 10^{-6} m^3
1572	af	acre foot = 43560 ft^3
1573	m^3 normal	Cubic meter
1574	L normal	Liter
1575	m^3 std.	Standard cubic meter
1576	L std.	Standard liter
1577	ml/s	Milliliters per second
1578	ml/h	Milliliters per hour
1579	ml/d	Milliliters per day
1580	af/s	Acre foot per second
1581	af/min	Acre foot per minute
1582	af/h	Acre foot per hour
1583	af/d	Acre foot per day
1584	fl oz (U.S.)/s	Ounces per second
1585	fl oz (U.S.) /min	Ounces per minute
1586	fl oz (U.S.)/h	Ounces per hour
1587	fl oz (U.S.)/d	Ounces per day
1588	m^3/s normal	Standard cubic meters per second
1589	m^3/min normal	Standard cubic meters per minute
1590	m^3/h normal	Standard cubic meters per hour
1591	m^3/d normal	Standard cubic meters per day
1592	L/s normal	Standard liters per second
1593	L/min normal	Standard liters per minute
1594	L/h normal	Standard liters per hour
1595	L/d normal	Standard liters per second
1596	m^3/s std.	Standard cubic meters per second
1597	m^3/min std.	Standard cubic meters per minute
1598	m^3/h std.	Standard cubic meters per hour
1599	m^3/d std.	Standard cubic meters per day
1600	L/s std.	Standard liters per second
1601	L/min std.	Standard liters per minute
1602	L/h std.	Standard liters per hour
1603	L/d std.	Standard liters per day
1604	ft^3/s std.	Standard cubic feet per second
1605	ft^3/d std.	Standard cubic feet per day
1606	oz/s	Ounces per second
1607	oz/min	Ounces per minute
1608	oz/h	Ounces per hour
1609	oz/d	Ounces per day
1610	Paa	Pascal (absolute)
1611	Pag	Pascal (gauge)
1612	GPaa	Gigapasacal (absolute)

Value	Display	Description
1613	GPag	Gigapascal (gauge)
1614	MPaa	Megapascal (absolute)
1615	MPag	Megapascal (gauge)
1616	kPaa	Kilopascal (absolute)
1617	kPag	Kilopascal (gauge)
1618	mPaa	Millipascal (absolute)
1619	mPag	Millipascal (gauge)
1620	µPaa	Micropascal (absolute)
1621	µPag	Micropascal (gauge)
1622	hPaa	Hectopascal (absolute)
1623	hPag	Hectopascal (gauge)
1624	gf/cm^{2}a	
1625	gf/cm^{2}g	
1626	kgf/cm^{2}a	
1627	kgf/cm^{2}g	
1628	SD4°C	Standard density at 4°C
1629	SD15°C	Standard density at 15°C
1630	SD20°C	Standard density at 20°C
1631	PS	Metric horsepower
1632	ppt	Parts per trillion = 10^{12}
1633	hl/s	Hectoliters per second
1634	hl/min	Hectoliters per minute
1635	hl/h	Hectoliters per hour
1636	hl/d	Hectoliters per day
1637	bbl (liq)/s	Barrels (US liquid) per second
1638	bbl (liq)/min x	Barrels (US liquid) per minute
1639	bbl (liq)/h	Barrels (US liquid) per hour
1640	bbl (liq)/d	Barrels (US liquid) per day
1641	bbl (fed)	Barrel (U.S. federal) = 31 gallons
1642	bbl (fed)/s	Barrels (US federal) per second
1643	bbl (fed)/min	Barrels (US federal) per minute
1644	bbl (fed)/h	Barrels (US federal) per hour
1645	bbl (fed)/d	Barrels (US federal) per day
1998	Unknown unit	Use in configuration when the unit of measure is not known
1999	Special	Special units

1.2.2 Monitoring functions

1.2.2.1 Monitoring functions at the Advanced Process Library

Monitoring functions in the Advanced Process Library

This and the following section encompass the standard monitoring functions in the Advanced Process Library. The monitoring functions include:

- Limit value monitoring
- Alarms at limit violations
- Time delays for alarms, warnings and tolerances
- Feedback monitoring
- Suppression of messages for alarms, messages and tolerances

For further and detailed information please refer to the following section. For the block-specific monitoring functions, also refer to the description for the respective block.

1.2.2.2 Limit monitoring

Limit monitoring of the process value

You can monitor the process value to the following high and low alarm, warning and tolerance limits:

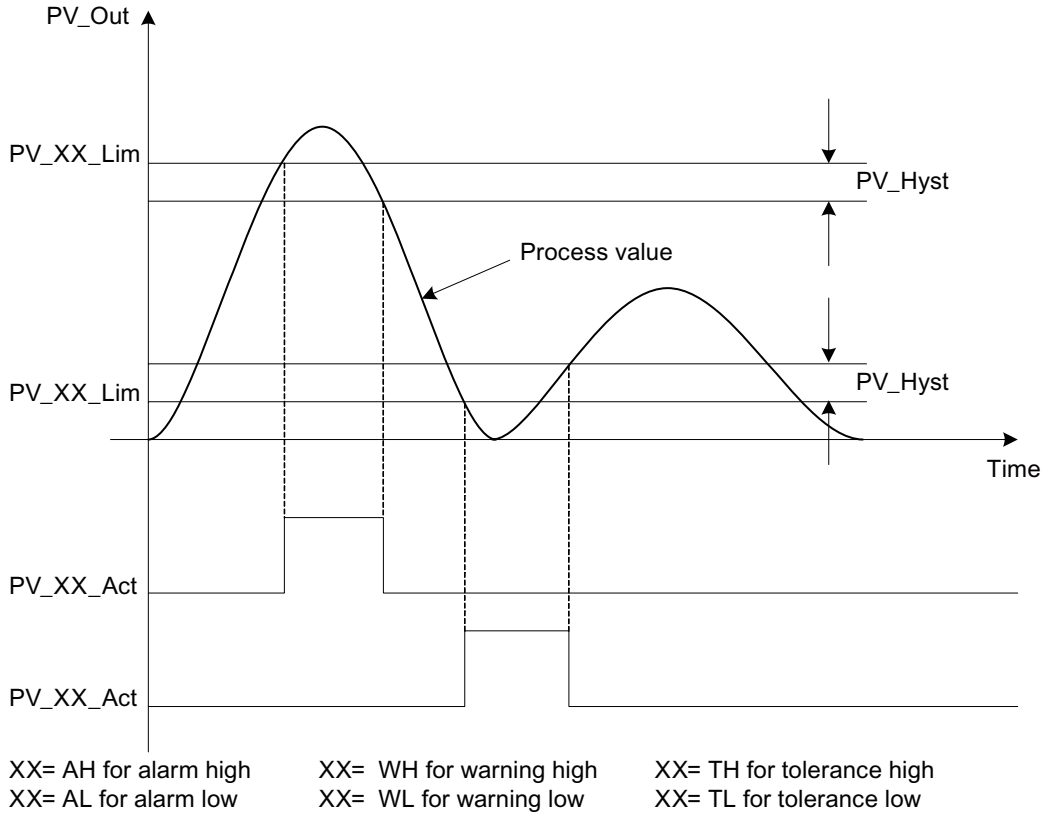
- `PV_AH_Lim`: Limit for high alarm
- `PV_AL_Lim`: Limit for low alarm
- `PV_WH_Lim`: Limit for high warning
- `PV_WL_Lim`: Limit for low warning
- `PV_TH_Lim`: Limit for the high tolerance
- `PV_TL_Lim`: Limit for the low tolerance

Result of the limit monitoring

The result of limit monitoring is made available at the interconnectable output parameters:

- `PV_AH_Act = 1`: Limit for high alarm reached or exceeded
- `PV_AL_Act = 1`: Limit for low alarm reached or undershot
- `PV_WH_Act = 1`: Limit for high warning reached or exceeded
- `PV_WL_Act = 1`: Limit for low warning reached or undershot
- `PV_TH_Act = 1`: Limit for high tolerance reached or exceeded
- `PV_TL_Act = 1`: Limit for low tolerance reached or undershot

made available (see figure). The `SumMsgAct = 1` output parameter is also set when at least one limit is reached or violated. This parameter can also be 1 if the limit for the position feedback value or the limit for the control error was reached or exceeded.



Switching on the limit monitoring

Monitoring is always enabled using the input parameters:

- PV_AH_En = 1: Monitoring of the high alarm limits
- PV_AL_En = 1: Monitoring of the low alarm limits
- PV_WH_En = 1: Monitoring of the high warning limits
- PV_WL_En = 1: Monitoring of the low warning limits
- PV_TH_En = 1: Monitoring of the high tolerance limits
- PV_TL_En = 1: Monitoring of the low tolerance limits

Predefinition: When the block is installed, monitoring of the tolerance limits is disabled, meaning that the parameters are configured with 0. To activate monitoring, assign 1 to these parameters.

All other monitoring functions are enabled.

Message suppression

The corresponding message is suppressed using the parameters:

- PV_AH_MsgEn = 0: Alarm (high) messages are suppressed
- PV_AL_MsgEn = 0: Alarm (low) messages are suppressed
- PV_WH_MsgEn = 0: Warning (high) messages are suppressed
- PV_WL_MsgEn = 0: Warning (low) messages are suppressed
- PV_TH_MsgEn = 0: Tolerance (high) messages are suppressed
- PV_TL_MsgEn = 0: Tolerance (low) messages are suppressed

The output of messages is not suppressed when the block is installed (all xx_MsgEn parameters are preset to 1). Messages can only be output if limit monitoring of the additional analog value has been enabled.

Hysteresis

You can specify a hysteresis (PV_Hyst) for the limits, for example, to suppress signal flutter. Refer to the Limit monitoring with hysteresis (Page 78) section for more on this.

Alarm delays

You can set alarm delays for coming and going alarms, warnings and tolerances. Refer to the Area of application of the alarm delays (Page 79) section for more on this.

Operating in the faceplate

You can also change the limits and the hysteresis using the faceplate. Refer to the Limit operation and display in the faceplate (Page 78) section for more on this.

See also

Two time values per limit pair (Page 80)

Two time values for each individual limit (Page 81)

Limit monitoring of an additional analog value

Limit monitoring of an additional analog value

Limit monitoring is performed for an additional analog value on the basis of the AV block, see Description of AV (Page 1207) section.

You can monitor an additional analog value to the following high and low limits for alarms warnings and tolerances at the technological block:

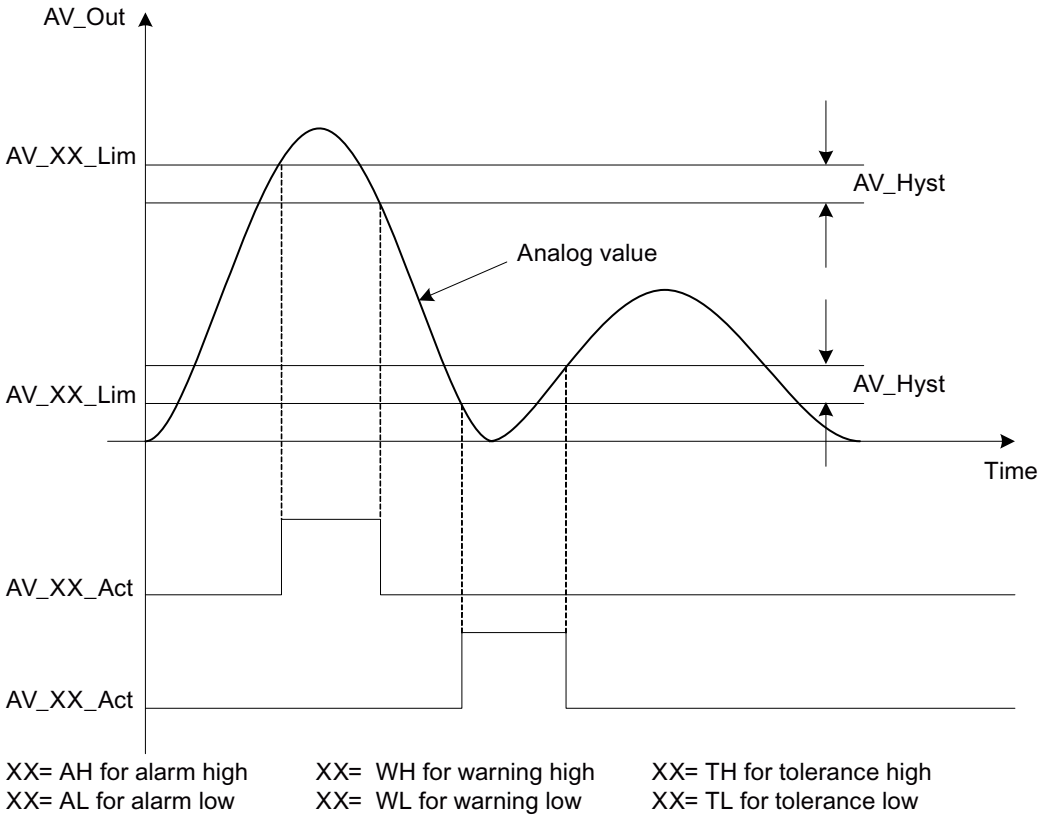
- AV_AH_Lim: Limit for high alarm
- AV_AL_Lim: Limit for low alarm
- AV_WH_Lim: Limit for high warning
- AV_WL_Lim: Limit for low warning
- AV_TH_Lim: Limit for the high tolerance
- AV_TL_Lim: Limit for the low tolerance

Result of the limit monitoring

The result of limit monitoring is made available at the interconnectable output parameters of the AV block:

- `AV_AH_Act = 1`: Limit for high alarm reached or exceeded
- `AV_AL_Act = 1`: Limit for low alarm reached or undershot
- `AV_WH_Act = 1`: Limit for high warning reached or exceeded
- `AV_WL_Act = 1`: Limit for low warning reached or undershot
- `AV_TH_Act = 1`: Limit for high tolerance reached or exceeded
- `AV_TL_Act = 1`: Limit for low tolerance reached or undershot

made available (see figure).



Switching on the limit monitoring

Monitoring is always enabled using the input parameters of the AV block:

- AV_AH_En = 1: Monitoring of the high alarm limits
- AV_AL_En = 1: Monitoring of the low alarm limits
- AV_WH_En = 1: Monitoring of the high warning limits
- AV_WL_En = 1: Monitoring of the low warning limits
- AV_TH_En = 1: Monitoring of the high tolerance limits
- AV_TL_En = 1: Monitoring of the low tolerance limits

Predefinition: When the block is installed, monitoring of the tolerance limits is disabled, meaning that the parameters are configured with 0. To activate monitoring, assign 1 to these parameters.

All other monitoring functions are enabled.

Message suppression

The corresponding message is suppressed at the block AV using the parameters:

- AV_AH_MsgEn = 0: Alarm (high) messages are suppressed
- AV_AL_MsgEn = 0: Alarm (low) messages are suppressed
- AV_WH_MsgEn = 0: Warning (high) messages are suppressed
- AV_WL_MsgEn = 0: Warning (low) messages are suppressed
- AV_TH_MsgEn = 0: Tolerance (high) messages are suppressed
- AV_TL_MsgEn = 0: Tolerance (low) messages are suppressed

The output of messages is not suppressed when the block is installed (all xx_MsgEn parameters are preset to 1). Messages can only be output if limit monitoring of the additional analog value has been enabled.

Hysteresis

You can specify a hysteresis (AV_Hyst) at the technological block for the limits, for example, to suppress signal flutter. Refer to the Limit monitoring with hysteresis (Page 78) section for more on this.

Alarm delays

You can set alarm delays for coming and going alarms, warnings and tolerances. Refer to the Area of application of the alarm delays (Page 79) section for more on this.

Operating in the faceplate

You can also change the limits and the hysteresis using the faceplate. Refer to the Limit operation and display in the faceplate (Page 78) section for more on this.

Limit monitoring of the feedback

Limit monitoring of position feedback

The position feedback of the manipulated variable can be monitored for the following high and low warning limits:

- `RbkWH_Lim`: Limit for high warning
- `RbkWL_Lim`: Limit for low warning

Result of limit monitoring of the position feedback

The result of limit monitoring of the position feedback is made available at the interconnectable output parameters:

- `RbkWH_Act = 1`: High limit reached or exceeded
- `RbkWL_Act = 1`: Low limit reached or undershot

. The `SumMsgAct = 1` output parameter is also set when at least one limit value is reached or violated. This parameter can also be 1 if the limit for the process value or the limit value for the control error was reached or exceeded.

When the limits are reached or exceeded, messages that can be suppressed are output.

Switching on the limit monitoring

Monitoring is always enabled using the input parameters:

- `RbkWH_En = 0`: Monitoring of the high warning limit is disabled
- `RbkWL_En = 0`: Monitoring of the low warning limit is disabled

Predefinition: When the block is installed, monitoring is disabled (default is 1).

Message suppression

The corresponding message is suppressed using the parameters:

- `RbkWH_MsgEn = 0`: Messages from the high limit monitoring are suppressed
- `RbkWL_MsgEn = 0`: Messages from the low limit monitoring are suppressed

The output of messages is not suppressed when the block is installed (for example, `RbkWH_MsgEn = 1`). Messages can only be output if limit monitoring of the position feedback has been enabled.

Hysteresis

You can specify a hysteresis (`RbkHyst`) for the limits, for example, to suppress signal flutter. Please also refer to the section Limit monitoring with hysteresis (Page 78).

Alarm delays (only for the PIDConR and MotSpdCL blocks)

You can set alarm delays for coming and going warnings. Please refer to the section Area of application of the alarm delays (Page 79).

Operating in the faceplate

You can also change the limits and the hysteresis using the faceplate. Please refer to the section Limit operation and display in the faceplate (Page 78).

Monitoring the error signal limits

Limit monitoring of control error

The control error of the manipulated variable can be monitored for the following high and low alarm limits:

- ER_AH_Lim: Limit for high alarm
- ER_AL_Lim: Limit for low alarm

Result of limit monitoring of the control error

The result of limit monitoring of the control error is made available at the interconnectable output parameters:

- ER_AH_Act =1: High limit violated (reached or exceeded)
- ER_AL_Act =1: Low limit (reached or undershot)

. The SumMsgAct = 1 output parameter is also set when at least one limit value is reached or violated. This parameter can also be 1 if the limit for the position feedback or the limit for the process value was reached or exceeded.

When the limits are reached or exceeded, messages that can be suppressed are output.

Enabling limit monitoring of the control error

Alarm monitoring is enabled using the input parameters:

- ER_AH_En = 1: Monitoring of the high alarm limit
- ER_AL_En = 1: Monitoring of the low alarm limit

Predefinition: When the block is installed, monitoring is disabled.

Message suppression

The corresponding message is suppressed using the parameters:

- ER_AH_MsgEn = 0: Messages from the high limit monitoring are suppressed
- ER_AL_MsgEn = 0: Messages from the low limit monitoring are suppressed

The output of messages is not suppressed when the block is installed (for example, ER_AH_MsgEn = 1). Messages can only be output if limit monitoring of the control error has been enabled.

Hysteresis

You can specify a hysteresis (ER_Hyst) for these limits, for example, in order to suppress signal flutter. Please also refer to the section Limit monitoring with hysteresis (Page 78).

Alarm delays

You can set alarm delays for coming and going alarms. Please refer to the section Area of application of the alarm delays (Page 79).

Operating in the faceplate

You can also influence the limits and the hysteresis by means of the faceplate. Please refer to the section Limit operation and display in the faceplate (Page 78).

Limit monitoring with hysteresis

Limit monitoring with hysteresis

You can additionally define a hysteresis for all limit monitoring functions (parameter xxx_Hyst, xxx can, for example, be PV for the process value). You use the hysteresis, for example, to suppress signal flutter.

Enter the hysteresis as a physical variable at the block and faceplate (if you have the appropriate operator control permissions (WinCC)).

For the WinCC operator control permissions, please refer to the help on WinCC.

Limit operation and display in the faceplate

Limit operation and display in the faceplate

The limit value view of the faceplate can be used to modify limits and the hysteresis if the corresponding operator control permission (higher process controlling) is available. The limits are displayed graphically in the standard view of the faceplate.

If the limits are reached or exceeded,, a message of the alarm class is, for example, triggered. This is indicated graphically as follows:

1.2.2.3 Alarm delays

Area of application of the alarm delays

Area of application

A sensible area of application for setting alarm delays can, for example, be a motor. When it is started, an elevated starting current can occur and this could be reported depending on the configured limit. Since this usually settles down to a value below the set limit, the alarm would not make sense. In this case, the alarm delay that is intended to bridge the duration of the active alarm is used.

Note

Alarms that are really wanted are, naturally, also delayed when an alarm delay is used. Therefore select the delay period prudently!

Alarm delays in the Advanced Process Library

There are three block types with a different application of the alarm delay for:

- One time value for all limits (Page 79)
- Two time values per limit pair (Page 80)
- Two time values for each individual limit (Page 81)

One time value for all limits

Blocks with one time value for the alarm delay

This form of alarm delay is used for blocks without a unit designation, for example, ConPerMon.

The alarm delay is used when brief violations of the set alarm thresholds are to be suppressed. The duration of the time delay is parameterized at the input `AlmDelay`. Parameter assignment is always carried out in seconds.

Activating the alarm delay

The alarm delay is disabled by default (`AlmDelay = 0`). To use the function, set a delay time [s] with the `AlmDelay` parameter.

Two time values per limit pair

Blocks with two time values for the alarm delay per limit pair

This form of alarm delay is used for blocks with an L in the unit designation. You recognize these blocks by the character L at the end of the block name, for example, Mot**L**.

The alarm delay is used when brief violations of the set alarm thresholds are to be suppressed. The alarm delay is parameterized at the following inputs:

Parameter for the delay time	Explanation
XXX_A_DC	Delay time for coming events of the class • Alarm **XXX_A**larm_**D**elay**C**oming XXX: Value to be monitored
XXX_A_DG	Delay time for going events of the class • Alarm **XXX_A**larm_**D**elay**G**oing XXX: Value to be monitored
XXX_W_DC	Delay time for coming events of the class • Warning **XXX_W**arning_**D**elay**C**oming XXX: Value to be monitored
XXX_W_DG	Delay time for going events of the class • Warning **XXX_W**arning_**D**elay**G**oing XXX: Value to be monitored
XXX_T_DC	Delay time for coming events of the class • Tolerance **XXX_T**olerance_**D**elay**C**oming XXX: Value to be monitored
XXX_T_DG	Delay time for going events of the class • Tolerance **XXX_T**olerance_**D**elay**G**oing XXX: Value to be monitored

Activating the alarm delay

By default, the alarm delay is disabled for each individual pair, meaning that each individual parameter has 0 [s] pre-assigned.

To use the function, set a delay time [s] for each parameter.

Alarms due to be dealt with

Alarms, warnings, or tolerances due to be dealt with are output at the corresponding output parameters:

- `XXX_A_Act = 1`: Alarm limit reached or violated
- `XXX_W_Act = 1`: Warning limit reached or violated
- `XXX_T_Act = 1`: Tolerance limit reached or violated

If a message is active at one of these outputs, this is indicated by a 1.

Two time values for each individual limit

Alarm delay for blocks with two time values for each individual limit

This form of alarm delay is used for blocks with an **R** in the unit designation. These blocks are identified by the character **R** at the end of the block name, e.g. PIDConR.

The alarm delay is used when brief violations of the set alarm thresholds are to be suppressed. The alarm delay is parameterized at the following inputs:

Parameter for the delay time	Explanation
`XX_AH_DC`	Delay time for coming events of the class • Alarm (for high limit) **XX_A**larm**H**igh_**D**elay**C**oming XX: Value to be monitored
`XX_AH_DG`	Delay time for going events of the class • Alarm (for high limit) **XX_A**larm**H**igh_**D**elay**G**oing XX: Value to be monitored
`XX_AL_DC`	Delay time for coming events of the class • Alarm (for low limit) **XX_A**larm**L**ow_**D**elay**C**oming XX: Value to be monitored
`XX_AL_DG`	Delay time for going events of the class • Alarm (for low limit) **XX_A**larm**L**ow_**D**elay**G**oing XX: Value to be monitored
`XX_WH_DC`	Delay time for coming events of the class • Warning (for high limit) **XX_W**arning**H**igh_**D**elay**C**oming XX: Value to be monitored
`XX_WH_DG`	Delay time for going events of the class • Warning (for high limit) **XX_W**arning**H**igh_**D**elay**G**oing XX: Value to be monitored

Parameter for the delay time	Explanation
XX_WL_DC	Delay time for coming events of the class • Warning (for low limit) **XX_W**arning**L**ow_**D**elay**C**oming XX: Value to be monitored
XX_WL_DG	Delay time for going events of the class • Warning (for low limit) **XX_W**arning**L**ow_**D**elay**G**oing XX: Value to be monitored
XX_TH_DC	Delay time for coming events of the class • Tolerance (for high limit) **XX_T**olerance**H**igh_**D**elay**C**oming XX: Value to be monitored
XX_TH_DG	Delay time for going events of the class • Tolerance (for high limit) **XX_T**olerance**H**igh_**D**elay**G**oing XX: Value to be monitored
XX_TL_DC	Delay time for coming events of the class • Tolerance (for low limit) **XX_T**olerance**L**ow_**D**elay**C**oming XX: Value to be monitored
XX_TL_DG	Delay time for going events of the class • Tolerance (for low limit) **XX_T**olerance**L**ow_**D**elay**G**oing XX: Value to be monitored

Activating the alarm delay

By default, the alarm delay is disabled for each individual limit, meaning that each individual parameter has 0 [s] pre-assigned.

To use the function, set a delay time [s] for each parameter.

Alarms due to be dealt with

Alarms, warnings, or tolerances due to be dealt with are output at the corresponding output parameters:

- XX_AL_Act = 1: Alarm limit (low) reached or violated
- XX_AH_Act = 1: Alarm limit (high) reached or violated
- XX_WL_Act = 1: Warning limit (low) reached or violated
- XX_WH_Act = 1: Warning limit (high) reached or violated
- XX_TL_Act = 1: Tolerance limit (low) reached or violated
- XX_TH_Act = 1: Tolerance limit (high) reached or violated

If a message is active at one of these outputs, this is indicated by a 1.

1.2.2.4 Feedbacks

Monitoring the feedbacks

Feedback monitoring

You can use the following monitoring functions:

- Monitoring the start-up and stop characteristics for motors or the run time of valves
- Monitoring the operation of motors or the maintenance of the position of valves
- Disabling feedbacks

This monitoring function is enabled via the `Monitor = 1` input.

Static and dynamic errors are reset by disabling the monitoring (`Monitor = 0`). If you reactivate monitoring during the plant runtime, only dynamic monitoring (`MonTiDynamic`) will be performed.

Monitoring the start-up and stop characteristics for motors or the run time of valves

Monitoring of the startup characteristics is implemented using the parameter `MonTiDynamic`. The monitoring time specifies the period within which the feedback value, for example, `FbkStart` with motors, must be available in response to a control signal. If this is not the case, the text "Control error" is displayed in the standard view of the faceplate. At the same time, an error message is generated. The block then goes to its safe position. In the case of motors, this is always the stop state. With other blocks, this is a safe position you have specified (parameter `SafePos`). The block signals this at the corresponding output parameter of the error message with 1, for example, with `MonDynErr = 1` for motors. In automatic mode, you will have to reset the monitoring error using the Reset button (`RstOp = 1` or via the interconnectable input parameter `RstLi = 1`). Use the `Feature` bit Resetting via input signals in the event of interlocks or errors (Page 194) to enable an automatic reset (bit = 1) at an edge transition. This is not necessary in manual mode.

Parameters are set in seconds.

Monitoring the operation of motors or the maintenance of the position of valves

Monitoring of the operation or the maintenance of the position of valves is implemented using the parameter `MonTiStatic`. The monitoring time specifies the period in which the feedback value can change its value briefly without an error message being output. An example would be a running motor with the feedback via the input parameter `FbkStart`. This parameter should be static in accordance with the control function. However, its value can change within the monitoring time. If the change in the `FbkStart` parameter takes longer than the monitoring time, the text "Runtime error" for motors or "Control error" for valves is displayed in the standard view of the faceplate. At the same time, an error message is generated. The block then goes to its safe position. In the case of motors, this is always the stop state. With other blocks, this is a safe position you have specified (parameter `SafePos`). The block signals this at the corresponding output parameter of the error message with 1, for example, with `MonStaErr = 1` for motors. In automatic mode, you will have to reset the monitoring error using the Reset button (`RstOp = 1` or via the interconnectable input parameter `RstLi = 1`) . Use the `Feature` bit Resetting via input signals in the event of interlocks or errors (Page 194) to enable an automatic reset (bit = 1) at an edge transition. This is not necessary in manual mode.

Parameters are set in seconds.

Note

Please note that `MonTiDynamic ≥ MonTiStatic` has to be configured.

Restarting after monitoring errors

There are different procedures depending on the operating mode for restarting after monitoring errors:

- Manual mode
- Automatic mode
- Local mode

Manual mode

The monitoring error cannot be reset when the control and feedback signals do not match.

When the control and feedback signals match, the monitoring error is automatically reset.

Automatic mode

The monitoring error cannot be reset when the control and feedback signals do not match. The Reset button in the faceplate is disabled.

When the control and feedback signals match, the reset is made depending on the setting made at the `Feature` bit Resetting via input signals in the event of interlocks or errors (Page 194).

Local mode

The monitoring error can only occur in local mode if you have set 1 or 3 for the input parameter `LocalSetting` (see Local mode (Page 36)). No monitoring error is output when `LocalSetting` is configured with 2 or 4.

The monitoring error cannot be reset when the control and feedback signals do not match.

When the control and feedback signals match, the monitoring error is reset by stopping (`StopLocal = 1`) the drive.

Disabling feedbacks

You can also disable feedback completely. Please refer to section Disabling feedback for valves (Page 85) for further information.

Disabling feedback for valves

Disabling monitoring of feedback for valves

You can operate a block without feedback. To do this, set the `NoFbkxxx = 1` parameter, whereby xxx stands for the respective function, for example, `NoFbkOpen` for the valve. This means, for example, that you do not have any feedbacks for the opened state of the valve. Monitoring is thus disabled for this feedback. The feedback at the block is adjusted according to the control signal.

1.2.3 Signals

1.2.3.1 Output signal as a static signal or pulse signal

Output signal as a static signal or pulse signal

When using motors, valves, dosers and digital operator control blocks (for example, OpDi01), you can output the control signals as:

- Static signal or as a
- Pulse signal with configurable pulse length.

You can find the signals in the I/O table of the individual blocks.

Note

This function is also available for the following blocks:

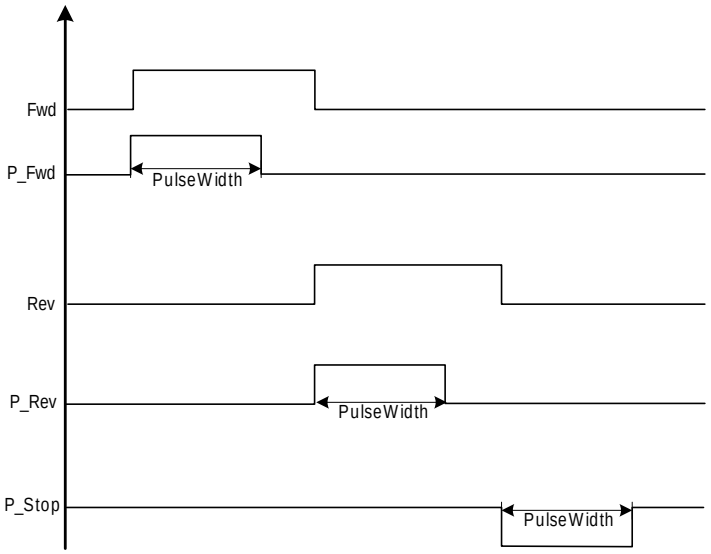
- Dose (`Out` and `Out2` parameters)
- OpDi01
- OpDi03

Output signal as a static signal

The control settings are made available as a static signal in the blocks in the form of interconnectable output parameters. The MotRevL block, for example, provides these as static signals via the alternative output parameters `Fwd`, `Rev` and `Run`.

Output signal as a pulse signal

The control is made available as pulse signals at the blocks using interconnectable output parameters. You specify the pulse length of the output signals in seconds using the input parameter `PulseWidth`. The MotRevL block, for example, provides these as pulse signals via the output parameters `P_Fwd`, `P_Rev` and `P_Stop`.



1.2.3.2 Recording the first signal for interlock blocks

Recording the first signal

You activate the function described below using the `Feature` bit " Activating recording of the first signal (Page 191)".

The number of the input that caused the last output signal change from 1 to 0 (good state to locked) is displayed for you in bit coding at the `FirstIn` output. The cause may be:

- A signal change at the input or a change in inversion

  Example: With an OR logic operation, the single 1 changes to 0. The output then changes from 1 to 0.

- A change to the I/O

  Example: Excluding the single 1 then results in an output of 0 with an OR logic operation.

- `FirstIn` is not changed if the following events occur, despite a change to the output:
  - Change in output value from or to `DefaultOut`

If several signals are at the same time responsible for the change, all responsible inputs are indicated in the faceplate and output in bit coding in the `FirstIn` output. If the input signals change without this causing the signal at the output to change, `FirstIn` does not change.

Inputs which are not interconnected or which are excluded are not taken into account.

You can reset `FirstIn` to 0 if you set the `RstLi` input from 0 to 1 (positive edge) or you can operate the `RstOp` input using the faceplate ("Reset" button).

If at least one bit in `FirstIn` is set, other signal changes are not taken into account.

1.2.3.3 Forming and outputting signal status for blocks

Outputting the signal status of the block

The process values of the function blocks are generated and transferred along with a signal status as a structured variable. This contains a statement about the signal quality. The function blocks determine the appropriate signal status for their process outputs depending on the signal status of the process inputs. The signal status is transferred via interconnections of the process signals. The signal status can be formed in the following ways within:

- Technological blocks, for example, controller and motor blocks
- Logical blocks
- Interlock blocks
- Channel driver blocks
- Driver blocks for field devices

Forming the signal status in technological blocks

In technological blocks, a group status is formed from the input parameters (see description of the relevant blocks) according to the priority table below (highest priority is 0). This group status is displayed in the status bar of the faceplate and of the block icon.

Signal status icon	Priority	Value	Meaning
	0	16#60	Local functional check / simulation
	1	16#00	Bad, device related
	2	16#28	Bad, process related
	3	16#68	Unknown, device related
	4	16#78	Unknown, process related
	5	16#A4	Maintenance request
	6	16#80	Good

Interconnectable output parameters (for example, PV_AH_Act) that can be influenced directly by an interconnectable input parameter (for example, PV) inherit the following status from this input parameter:

- Good
- Bad, process related
- Bad, device related

If an output parameter is influenced directly by several interconnectable input parameters (limit monitoring), it receives the status of the input parameter having the highest priority (see the overview above). Thus, for example, the system deviation is formed from the setpoint (SP) and the process value (PV). The output parameter ER_AH_Act, for example, that signals an active violation of the high limit, has a signal status based on the group status formed from the process value and setpoint.

Feedback signals (Fbk) and position feedbacks (Rbk) that have a direct influence on the block function and are monitored (for example, runtime monitoring), do not have any influence on the status of output signals.

Evaluation of the signal status in case of interlocks in technological blocks

The signal states of the interconnectable input parameters of the interlocking and protective signals are treated exactly like process values with the following exceptions:

- The signal status is displayed in the faceplate at the buttons for calling the series-connected interlocking blocks.
- If the input signal has the signal status for a simulation and if the interlocking condition is fulfilled (for example, `Protect = 1`), the input signal is interpreted in this case as a bypassed signal and is displayed using the icon for bypassing.
- If the input signal has the signal status for a simulation and if the interlocking condition is not fulfilled (for example, `Protect = 0`), the input signal is interpreted in this case as a simulated signal and is displayed using the icon for simulating:

- If the input signal has the signal status for the state "Bad, device related" state or "Bad, device related", this is evaluated as an unfulfilled interlocking condition (for example `Protect = 0`), regardless of its value (for example, `Protect = 1`).

Display of the signal status in the faceplate and block icon for technological blocks

The signal status is displayed for each individual parameter (except for the additional values) in the faceplate next to the process values. The group status is displayed in the block icon and in the group display of the faceplate.

Generating the signal status within logical blocks

Within logical blocks, a group status is generated from all input parameters, according to a priority table (highest priority is 0). The parameter `SelPrio` is used to define the priority specification used to carry out interlinking of the individual states (default setting is usually 0). You have the option between the following specifications:

Priority	SelPrio = 0	SelPrio = 1	SelPrio = 2	SelPrio = 3	SelPrio = 4	SelPrio = 5	SelPrio = 6	SelPrio = 7
0	 16#60	 16#80	 16#A4	 16#80	 16#A4	 16#60	 16#80	 16#60
1	 16#00	 16#00	 16#00	 16#A4	 16#80	 16#80	 16#A4	 16#80
2	 16#28	 16#28	 16#28	 16#60	 16#60	 16#A4	 16#78	 16#A4
3	 16#68	 16#68	 16#68	 16#00	 16#00	 16#00	 16#68	 16#78
4	 16#78	 16#78	 16#78	 16#28	 16#28	 16#28	 16#28	 16#68
5	16#A4	16#60	16#60	16#68	16#68	16#68	16#00	16#28
6	16#80	16#A4	16#80	16#78	16#78	16#78	16#60	16#00

Note

The `SelPrio` parameter can take a value from 0 to 7. If you enter a value greater than 7, the setting for 7 is used. If you select a value less than 0, the setting for 0 is used.

Evaluation of the signal status in the case of interlocking blocks (such as Intlk02)

The block determines the signal status of the output signal, based on the signal status of the input values, the logic operation, and the output value in accordance with the following rules:

General rules

- The best/worst signal status is generated according to the priority table for technological blocks.
- The signal status of the output is generated from the best (lowest priority) or worst (highest priority) signal status of those input signals (highest priority is 0), which are interconnected and not excluded (relevant input signals) in accordance with the logic operation and its result.
- If all inputs are excluded or not interconnected, the signal status of the output is set to simulation (16#60).

AND operation

- If the output value is 1, it has the worst signal status of all relevant input signals.
- If the output value is 0, it has the best signal status of the relevant input signals with a value of 0.
- If the output value is 1 and at least one input is excluded, the signal status of the output is set to simulation (16#60).

OR operation

- If the output value is 1, it has the best signal status of the relevant input signals with a value of 1.
- If the output value is 0, it has the worst signal status of all relevant input signals.
- If the output value is 0 and at least one input is excluded, the signal status of the output is set to simulation (16#60).

Signal status icon	Priority	Value	Meaning
	0	16#00	Bad, device related
	1	16#28	Bad, process related
	2	16#60	Local functional check / simulation
	3	16#68	Unknown, device related
	4	16#78	Unknown, process related
	5	16#A4	Maintenance request
No icon	6	16#80	Good

Display of the signal status in the faceplate and block icon for interlocking blocks

The signal status is displayed for each individual parameter (except for the analog values) in the faceplate next to the process values.

If you exclude a signal, it is displayed in the faceplate of the interlocking block next to the button for excluding, as well as in the block icon as follows:

The currently valid status for the output signal is also displayed in the faceplate.

Forming the signal status for PCS7 channel blocks

The signal status for PCS7 channel blocks can assume the following values:

Value	Meaning
16#80	Good
16#78	Unknown, process related: Limitation of input parameter PV_In is active (analog output drivers only)
16#60	Simulation, substitute value or last valid value
16#00	Bad, device related (value not valid)

Forming the signal status for Fb channel blocks (field devices)

The signal status of Fb channel blocks for field devices can assume the following values:

Value	Meaning
16#80	Good
16#78	Unknown, process related
16#68	Unknown, device related
16#60	Simulation, substitute value or last valid value
16#28	Bad, process related
16#00	Bad, device related (value not valid)
16#A4	Maintenance demand, maintenance request

1.2.3.4 Selecting signals for processing

Selecting signals for processing

Using the `SelInput` input parameter, you select the number of interconnectable input parameters to be used for processing in the block. The selected number denotes the input parameters used In_1 ... In_n.

If, for example, you set this input parameter to `SelInput = 3`, the input parameters In_1, In_2 and In_3 will be used for processing.

1.2.3.5 Simulating signals

Simulating signals

Simulation means the manipulation of a signal regardless of the actual source of the signal or logic that generates this signal.

Simulation is carried out either at the field device (externally from the control system) itself or at a block (internally in the control system).

In either case. the status of the signal is set to the simulation value (see also Forming and outputting signal status for blocks (Page 88)).

During the simulation, every block is considered in isolation. There are two different forms of simulation here, namely:

- Block-external simulation and
- Block-internal simulation.

Block-external simulation

Block-external simulation is characterized by the fact that:

- The simulation function is not executed in the block itself and
- A signal whose status has the simulation state, for example, a simulation of the signal at another block or directly in the I/O device, is applied at an input parameter.

The block-external simulation has the following effects on the functionality of the block:

- The technological functions are not influenced
- All the process-relevant output signals receive the simulation status, for example, signals for further processing in other blocks.
- In the case of blocks with operator control or monitoring functions (for example faceplates), these signals are identified in the faceplate with the status for the simulation as follows:

- Blocks with one or more input parameters for signals with status generate a group status from the individual states in accordance with the priority table. This group status is displayed in the status bar of the block icon and of the operator block with the simulation status as follows:

- The interlocking functions of the block are not influenced.

Block-internal simulation

Block-internal simulation is characterized by the simulation function being run in the block itself.

With operator control and monitoring blocks, all process values that cannot be controlled (e.g. PV, AV, `In`) can be simulated. This is used primarily as an aid for commissioning and servicing of the system. For example, the control settings of a motor can be simulated and the feedback values corrected without the monitoring functions being active.

Simulation can also be carried out for blocks (such as channel blocks) that cannot be controlled and monitored by the operator.

The control is simulated in the CFC by setting parameters directly in the block with the input parameters `SimOn = 1` and `Simxxxx =` for the desired simulation value (e.g. `SimPV`, `SimAV` or `SimIn`).

Note

With channel blocks, ensure that the `Mode` parameter is set correctly during simulation. Otherwise this is displayed on the `Bad = 1` output parameter with a higher-level error.

Simulation takes place in runtime using operator input in the faceplate's parameter view by clicking on the "Simulation" button.

This simulation is characterized by the fact that:

- The simulation can only be enabled / disabled with the operator authorization level for system authorization.
- The technological functions are not influenced.
- All the process-relevant output signals receive the simulation status, for example, signals for further processing in other blocks.
- In the case of blocks with operator control or monitoring functions (for example faceplates), these signals are identified in the faceplate with the status for the simulation as follows:

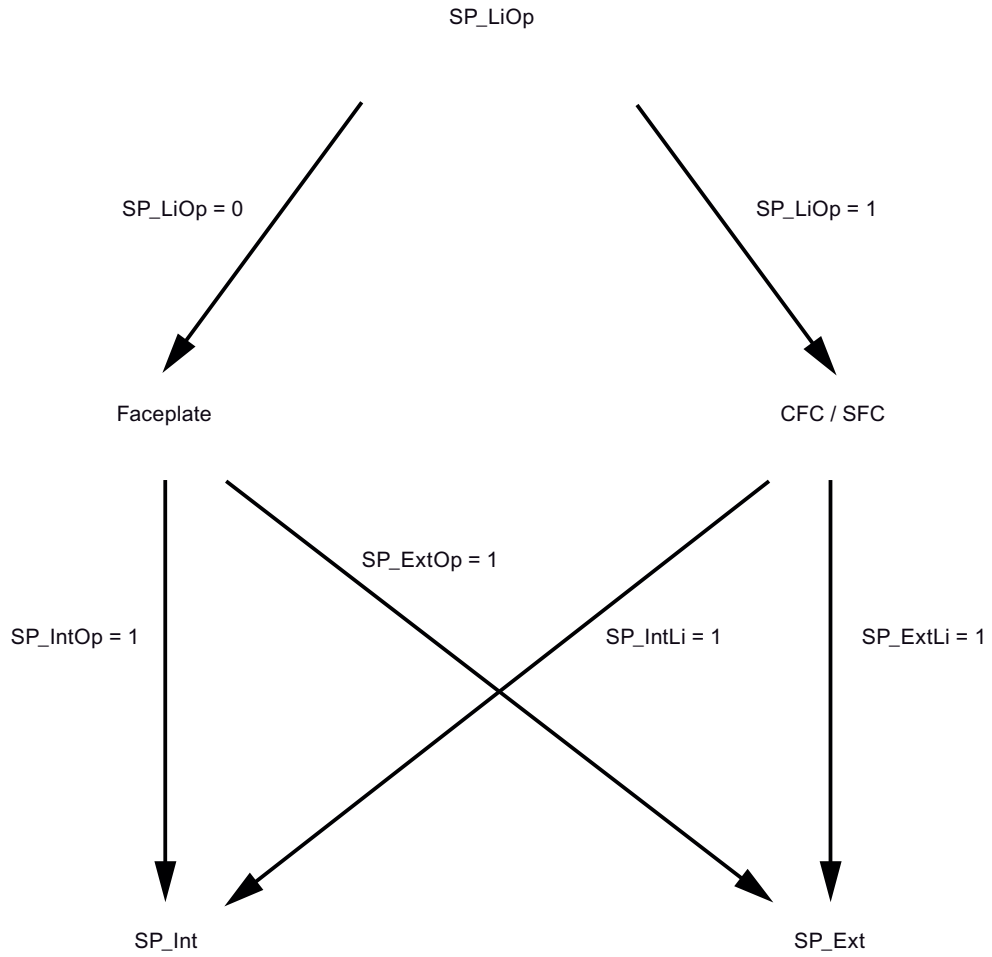
- The group status of the block is displayed in the status bar of the block icon and of the operator block with the simulation status as follows:

- All the process values displayed in the faceplate that cannot be operated-controlled in normal operation (e.g. PV).
- When the control function of the block can be manipulated, the read-back and feedback values (for example `Rbk`, `FbkSpd1`) are adjusted according to the manipulation of the control settings
- Associated values (for example `UserAna1`) cannot be simulated.
- The interlocking functions of the block are disabled and are displayed using the corresponding icon in the faceplate and in the block icon as follows:

1.2.3.6 Setpoint input - internal and external

Setpoint specification internal & external

Some blocks have a function that allows setpoints to be specified. This specification is carried out either by means of a CFC/SFC program or by means of the faceplate (operator). With doser blocks and frequency converters, the operator can specify the internal setpoint value (SP_Int) or a higher-level open-loop control will specify an external setpoint value (SP_Ext). In principle, the blocks operate according to the same scheme:



First you define whether the setpoint specification is to be carried out by means of a CFC/SFC program or by means of the faceplate. In the next step you specify whether the internal or the external setpoint is to be used.

Setpoint specification by means of faceplate or interconnection

With the SP_LiOp parameter, you define whether the setpoint will be set by a CFC/SFC program or using the faceplate.

- Parameterize Sp_LiOp with 0 so that the setpoint specification is carried out by means of the faceplate.
- Parameterize SP_LiOp with 1 so that the setpoint specification is carried out by means of a CFC / SFC program.

Setpoint specification internal & external

You have to set the corresponding parameters depending on how the setpoint specification is to be carried out.

If the setpoint is set in the faceplate (`SP_LiOp = 0`), you have to set the parameter:

- `SP_IntOp = 1` in order to achieve an internal setpoint specification by means of the faceplate.
- `SP_ExtOp = 1` to have an external setpoint set in the faceplate.

If both signals are set, `SP_IntOp = 1` has priority. If the setpoint is set by a CFC / SFC program (`SP_LiOp =1`), you have to set the parameter:

1. `SP_IntLi = 1` to have an internal setpoint set by a CFC / SFC program.
2. `SP_ExtLi = 1` in order to achieve an external setpoint specification by means of a CFC / SFC program.

If both signals are set, `SP_IntLi = 1` has priority.

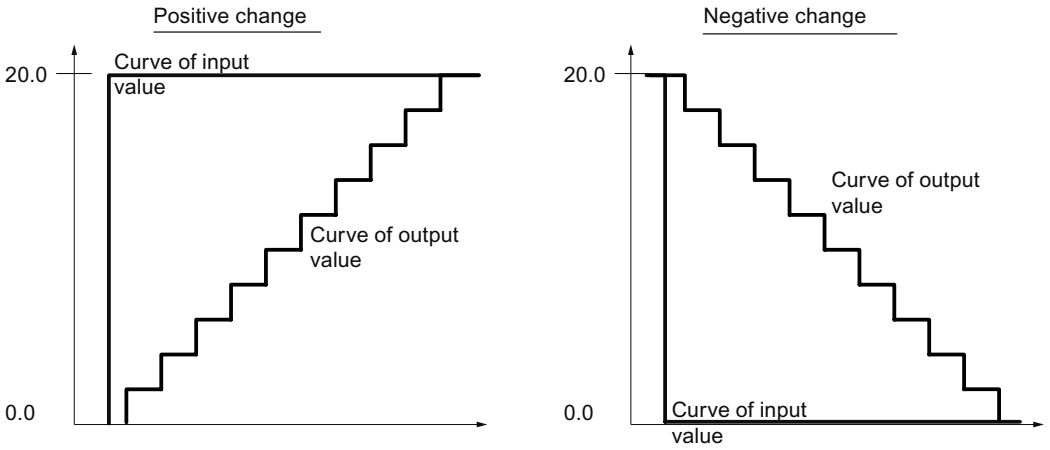
Bumpless switchover from external to internal setpoint

The parameter `SP_TrkExt = 1` is used so that the internal setpoint tracks the external setpoint to achieve a bumpless switchover from the external to the internal setpoint. This allows unwanted jumps at the output parameter to be avoided.

1.2.3.7 Using a setpoint ramp

Using setpoint ramp

Starting at the current internal setpoint, the setpoint can be set to a target setpoint value over a ramp-shaped function. In the faceplate, you can start the function in ramp view (`SP_RmpOn = 1`)).



Use the `SP_RmpModTime` input parameter or the ramp view of the faceplate to specify whether the setpoint ramp is defined over time or over the ramp:

- If you select time (`SP_RmpModTime = 1`): The ramp of the setpoint is calculated automatically by the block so that after the ramp has started (`SP_RmpOn = 1`), the setpoint will reach the target setpoint (`SP_Target`) after the selected time (`SP_RmpTime`).
- If you select ramp (`SP_RmpModTime = 0`): The inclination of the ramp matches the selected rates of change `SP_UpRaLim` (positive) or `SP_DnRaLim` (negative).

Once the setpoint has reached the target setpoint, the function is terminated automatically (`SP_RmpOn = 0`). The ramp trip can be prematurely aborted in the faceplate by setting `SP_RmpOn = 0`. The ramp trip is aborted automatically if automatic mode with an internal setpoint is ended.

Note

The setpoint ramp can only be used if the setpoint was selected internally.

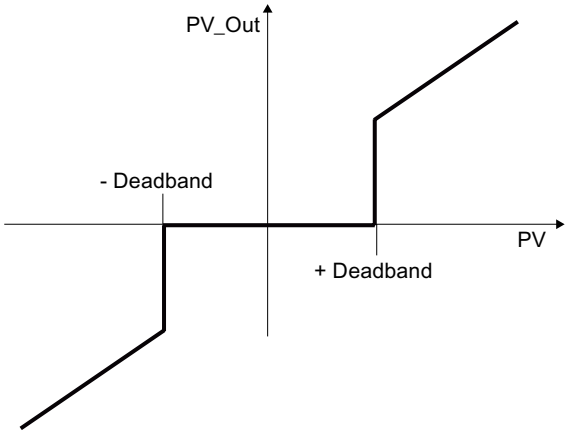
1.2.3.8 Deadband

Dead band

To suppress values fluctuating around zero, you can set a dead band (`Deadband`):

`Deadband = 0`: Dead band is disabled

`Deadband ≠ 0`: Dead band is enabled



1.2.4 Interlocking functions

1.2.4.1 Interlocks

Interlocks at blocks

A maximum of three types of interlock can be used depending on the block. Three separate inputs named `Intlock`, `Protect` and `Permit` are available for these functions. The following interlock types exist:

- **Activation enable (Permission):** The activation enable (input `Permit = 1`) makes it possible to leave the safe position of the block in response to operator input or a command from the program (CFC/SFC). The activation enable signal has no effect if the block is not in the safe position.
- **Interlock without reset (Interlock):** An active interlock condition brings the block to the safe position (input `Intlock = 0`). After the interlock condition has gone, the currently active control function becomes active again in automatic or local mode. In manual mode the faceplate can be operated again after the interlock condition has gone.
- **Interlock with reset (Protection):** An active interlock condition brings the block to the safe position (input `Protect = 0`). After the interlock conditions are cleared, the operator or an activation sequence must reset the control functions again to match the input parameters (only in automatic mode). The parameter `Features` can be used to specify the reset reaction of the block:
  - Feature Bit = 0: Reset is carried out either by operation in the faceplate, or using the interconnectable input parameter (`RstLi = 1`) at the block.
  - Feature Bit = 1: It is also possible to reset with a 0-1 edge change in the control signal in automatic mode.

  For additional information, refer to the description of the `Feature` bit Resetting via input signals in the event of interlocks or errors (Page 194).

Display of the interlock in the faceplate and in the block icon

The interlock state is visualized in the faceplate and in the block icon by a status display (padlock) as follows:

- Open padlock: No active interlock
- Closed padlock: One or more interlocks are activate
- No padlock: Individual interlocks are not active
  - `Perm_En = 0` or `Permit.ST = 16#FF`: of the input parameter `Permit` has no effect, the button in the faceplate is invisible
  - `Prot_En = 0` or `Protect.ST = 16#FF`: of the input parameter `Protect` has no effect, the button in the faceplate is invisible
  - `Intl_En = 0` or `Intlock.ST = 16#FF`: of the input parameter `Intlock` has no effect, the button in the faceplate is invisible

The block icon indicates the prioritized group status according to the active operating state. See also the Forming the group status for interlock information (Page 105) section for more on this.

The faceplate visualizes the state of each interlock type separately.

The padlock is not shown in the block icon if all parameters for enabling the button are set to 0 (`Perm_En = 0, Prot_En = 0, Intl_En = 0`) or all parameters have the signal status `16#FF` (`Permit.ST = 16#FF, Protect.ST = 16#FF, Intlock.ST = 16#FF`).

Note

Motors and values are not put into the safe position if one of the interlock inputs is active (e.g. `Intlock = 0`) and the corresponding signal status is `16#FF` (`Intlock.ST = 16#FF`).

Influence of the signal status on the interlock

Refer to the Influence of the signal status on the interlock (Page 103) section for more on this.

Messaging

No messages are assigned to the interlock types. However, if you want to have a message when an interlock condition is violated, you can use the freely interconnectable input parameters to generate the messages. See also the Generating instance-specific messages (Page 46) section for more on this.

1.2.4.2 Trip function

Trip function

The trip function is used to turn off the motor if there is heat overload (`Trip = 0`, interconnectable input parameter). The function can be deactivated with the `BypProt = 1` parameter in local mode and in the simulation status.

If the motor is turned off by the trip function, a message (process control message) is generated. This is indicated in the faceplate with the "Trip" text.

In automatic mode, you will have to reset the trip using the Reset button (`RstOp = 1` or via the interconnectable input parameter `RstLi = 1`) . This is not necessary in manual mode.

1.2.4.3 Rapid stop for motors

Rapid stop for motors

Rapid stop has the highest priority in all operating modes (manual and automatic mode as well as local mode) and operating states (such as the forcing of states). It is activated via the faceplate. This depends on the setting at the `Feature` bit Enabling rapid stop via faceplate (Page 192).

You issue the command for rapid stop state using the `RapidStp = 1` input parameter.

When you click on the "Rapid Stop" button in the faceplate, the drive stops immediately, shown as follows in the faceplate:

The `R_StpAct = 1` output parameter is set to implement the rapid stop function for local mode. You need to interconnect this parameter to the corresponding driver block and in the I/O to realize the rapid stop function in the hardware.

Rapid stop is unlocked for all operating modes with the "Reset" button in the faceplate (`ResetOp = 1`); in CFC it is unlocked with the `ResetLi = 1` input parameter.

1.2.4.4 Influence of the signal status on the interlock

Influence of the signal status on the interlock

There are three ways in which the signal status affects the interlocks:

- Simulation signal status (value 16#60)
- Signal status "Bad, device related" (value 16#00) or "Bad, process related" (value 16#28)
- Signal status ≠ "Simulation" and "Bad, device related"

"Simulation" signal status (value 16#60)

If the interlock signal with the status 16#60 brings about a cancellation of the interlock, this is processed as a bypass of the interlock in the block and displayed with the following icons in the faceplate:

If the interlock signal with the status 16#60 does not cancel the interlock, this is executed as a simulation in the block and displayed with the following icons in the faceplate:

Signal status "Bad, device related" (value 16#00) or "Bad, process related" (value 16#28)

An interlock signal with this status is always processed as an active interlock signal in the block and displayed with the following icons in the faceplate:

for 16#00 or

for 16#28 and

A motor protection signal (`Trip` parameter) with signal status 16#00 or 16#28 is used to activate motor protection. This is indicated by means of "Trip" in the standard view of the faceplates.

Signal status ≠ "Simulation" and "Bad, device related"

Only signal states "Simulation" and "Bad, device related" affect processing in the block; all others are displayed simply with their relevant icon in the faceplate.

1.2.4.5 Forming the group status for interlock information

Forming the group status for interlock information

A group status for interlock information is required for:

- Interlock state for:
  - Interlocked
  - Not interlocked
  - Deactivated

Group status for the interlock state

All the effective interlock states are combined and displayed in the block icon. The interlock states are displayed with the following prioritization:

1. Function interlocked, shown in the block icon with a closed padlock

2. Function not interlocked, shown in the block icon as an open padlock

3. Function disabled, shown in the block icon as a crossed-out, closed padlock

Overview: Display for the interlock status in faceplate

Interlocks are shown in the faceplate as follows:

Parameter: BypProt	Parameter: Perm_En or Prot_En or Intl_En	Value: Permission or Protection or Intlock	Status: Permission or Protection or Intlock	Button	Icon	Icon
x	x	X	16#FF	-	-	-
x	0	x	x	-	-	-
0	1	1	16#00	Visible		
0	1	1	16#28	Visible		
0	1	0	16#60	Visible		
0	1	1	16#60	Visible		
0	1	0	≠ 16#FF	Visible		Status based on priority
0	1	1	≠ (16#FF, 16#60, 16#00, 16#28)	Visible		Status based on priority
1 (for local mode and simulation only)	1	x	≠ 16#FF	Visible		Status based on priority

Comments on table:

- X: The value is irrelevant for the display of the icon.

Note

If values are forced in the block, this is indicated in the block icon by a crossed-out, closed padlock.

1.2.4.6 Disabling interlocks

Disabling individual interlocks

You can disable the block interlocks that are implemented using the input parameters `Intlock`, `Protect` and `Permit`.

If you want to disable the block interlock, you have to set the following parameters:

- `Perm_En = 0` or `Permit.ST = 16#FF`: The `Permit` input parameter has no effect
- `Prot_En = 0` or `Protect.ST = 16#FF`: The `Protect` input parameter has no effect
- `Intl_En = 0` or `Intlock.ST = 16#FF`: The `Intlock` input parameter has no effect

`LockAct = 0` is included in the conditions mentioned above.

Disabling of all the interlocks (only for local operation and for simulation)

You can use the input parameter `BypProt = 1` to disable all the interlocks, irrespective of the parameter assignment of the individual interlock, in local mode as well as for the simulation function.

1.2.5 General functions for controllers

1.2.5.1 Ramp limiting of the setpoint

Gradient limit of the setpoint

The gradient limit is activated via the `SP_RateOn = 1` input parameter.

The values are set at the `SP_UpRaLim` and `SP_DnRaLim` parameters.

- `SP_UpRaLim` sets the upper gradient limit
- `SP_DnRaLim` sets the lower gradient limit

Note

ParametersSP_UpRaLim and SP_DnRaLim are always evaluated according to their magnitude.

Displaying active limitation

A gradient limit is indicated at the following output parameters:

- `SP_UpRaAct = 1`: Gradient has an upper limit
- `SP_DnRaAct = 1`: Gradient has a lower limit

1.2.5.2 Inverting the control sense

Inverting the control action

For some processes (for example cooling processes), negative control gain is necessary. This is achieved by the inverting the control action by means of the input parameter `NegGain = 1`. The gain is always entered positively at the input parameter `Gain` . If there is an inversion, this is indicated at the output parameter `GainEff` by a negative number.

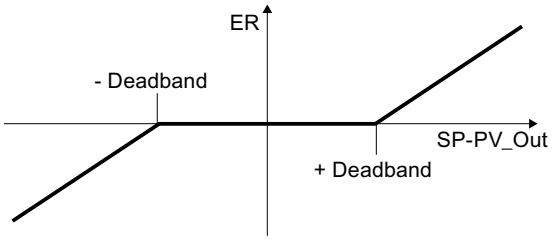
1.2.5.3 Control error generation and deadband

Control error generation and dead band

The control error is formed from the effective setpoint SP and the process valuePV (ER = SP - PV_Out) and is available at the ER output.

To suppress disturbances in the steady state, you can assign a dead band (Deadband):

- Deadband = 0: Dead band is disabled
- Deadband ≠ 0: Dead band is enabled



1.2.5.4 Using control zones

Using control zones

The control zone function is mainly used for temperature processes.

If ConZone ≠ 0, the controller works with a control zone; if ConZone = 0, the "control zone" function is deactivated. This means that the controller operates based on the following algorithm:

- If the process value PV exceeds the setpoint SP by more than the value defined for ConZone, the value MV_LoLim is output as the manipulated value (controlled closed-loop mode).
- If the process value PV falls below the setpoint SP by more than ConZone, MV_HiLim is output (controlled closed-loop mode).
- If the manipulated value PV stays within the control zone (ConZone), the manipulated value assumes the value of the PID algorithm (automatic closed-loop mode).

Note

The change from controlled closed-loop mode to automatic closed-loop mode is based on a hysteresis of 20% of the control zone. Make sure that the control zone has an adequate width before you manually activate the control zone. An insufficient width of the control zone leads to oscillation of the manipulated variable and of the process value.

Advantages of the control zone:

Once the control zone is entered, the manipulated variable is quickly reduced by the applied D action. The control zone is therefore only useful if the D action is activated. Without the control zone, only the reducing P action would reduce the manipulated variable to any degree. The control zone speeds up settling without overshoot or undershoot if the value of the output minimum or maximum manipulated variable is a long way from the stationary manipulated variable required for the new operating point.

1.2.5.5 Setpoint limiting for external setpoints

Setpoint limiting for external setpoints

With this function you can limit the external setpoint to a range by means of the parameters SP_ExHiLim (high limit) and SP_ExLoLim (low limit). If the setpoint lies outside the range defined by you, it is limited to the valid range.

If the external setpoint lies on or above the limit SP_ExHiLim, this is displayed at the output SP_ExHiAct = 1.

If the external setpoint lies on or below the limit of SP_ExLoLim, this is displayed at the output SP_ExLoAct = 1.

The external setpoint limits can be tracked internally via the interconnection of the output parameters SP_InHiOut or SP_InLoOut following SP_ExHiLim or SP_ExLoLim. You can control the two limit pairs from the faceplate.

1.2.5.6 Setpoint tracking the process variable in manual mode

Tracking setpoint in manual mode

To allow a bumpless transfer to automatic mode, the setpoint tracks the process value. When tracking,(SP_TrkPV = 1) the internal setpoint SP_Int is tracked to the process value PV.

Additional information on setpoint tracking is available in Manual and Automatic mode for control blocks (Page 29).

1.2.5.7 Tracking and limiting a manipulated value

Manipulated variable tracking

You adjust the manipulated value (tracking) in order to implement a bumpless switchover of controllers. A typical use case is cascade control: If the assigned secondary controller is no longer in automatic mode with external setpoint, the primary controller must track it.

To adjust the manipulated variable (tracking), you have to set the parameter `MV_TrkOn = 1`. Now the manipulated variable is taken from the interconnected tracking value `MV_Trk` and passed to the output `MV`.

Manual mode takes priority over tracking to allow a plant operator to set the controller to manual mode using the faceplate even when tracking the manipulated variable and therefore continue to normal operation.

The text "Tracking" is displayed additionally in the standard view of the faceplate.

Activating forced tracking mode for the manipulated variable

Forced tracking is used to set the controller output of a higher-level controller to a value that can be specified.

You can use forced tracking, for example, to implement a centralized emergency stop in the plant. This can be put into effect regardless of the operating mode the controller is currently in.

To force adjustment of the manipulated variable (tracking), you have to set the parameter `MV_ForOn = 1`. Now the manipulated variable is taken from the interconnected tracking value `MV_Forced` and passed to the output `MV`.

With forced tracking, it is not possible to limit the manipulated variable, nor can the plant operator change to manual mode in the faceplate. The text "Forced tracking" is displayed additionally in the standard view of the faceplate.

Note

This function is not available for the FmCont, FmTemp, and ModPreCon blocks.

Limiting the value of a manipulated variable in automatic mode

In automatic mode, the manipulated variable is set to its automatic manipulated variable limits as specified by the input parameters `MV_HiLim` and `MV_LoLim` and output at the output parameter `MV`. Reaching the limit is then displayed at the output parameter `MV_HiAct = 1` for the high limit and `MV_LoAct = 1` for the low limit.

Interconnecting the output parameter `ManHiOut` or `ManLoOut` to `MV_HiLim` or `MV_LoLim` allows the automatic manipulated variable limits to tracked to manual manipulated variable limits. You can keep both limit pairs in synch and control the manual manipulated variable limits in the faceplate.

Limiting the value of a manipulated variable in manual mode

In manual mode, the manipulated variable is set to its manual manipulated variable limits as specified by the input parameters `ManHiLim` and `ManLoLim` and output at the output parameter `MV`.

See also

Forcing operating states (Page 115)

1.2.5.8 Activating and limiting disturbance variable

Feedforward control and limitation

The feedforward control is used in order to compensate measurable disturbance variables, such as temperature or pressure, that can have an effect on the process. In automatic mode, the disturbance variable is added to the result of the PID algorithm.

The disturbance variable is connected to the `FFwd` Parameter. It is limited to the `FFwdHiLim` and `FFwdLoLim` limits. If the disturbance variable is outside or at the limits, this is indicated by the `FFwdHiAct = 1` or `FFwdLoAct = 1` output parameters.

1.2.5.9 Structure segmentation at controllers

Structure segmentation at controllers

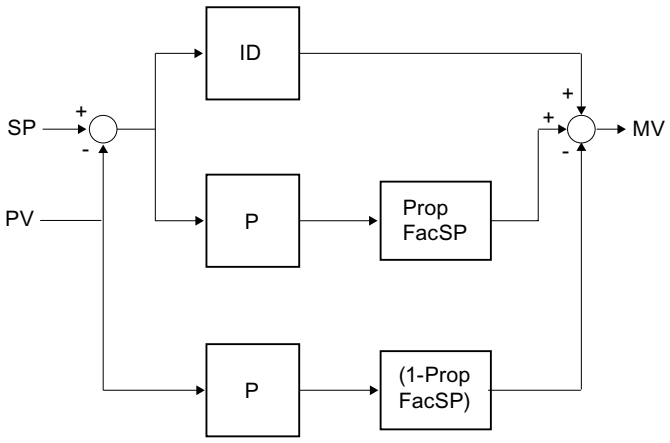
In order to avoid jumps at the manipulated value (controller output) during setpoint changes, proportional and derivative actions can be switched into the feedback path. I.e.: The proportional action (proportionally) and the derivative action are then only influenced by the process value.

Switching proportional action into the feedback path

A proportion of the P action can be placed in the feedback using the `PropFacSP` parameter. Setpoint jumps then only affect the derivative action proportionally.

`PropFacSP = 0`: Proportional action is completely in the feedback path

`PropFacSP = 1`: Proportional action is not in the feedback path (default setting)

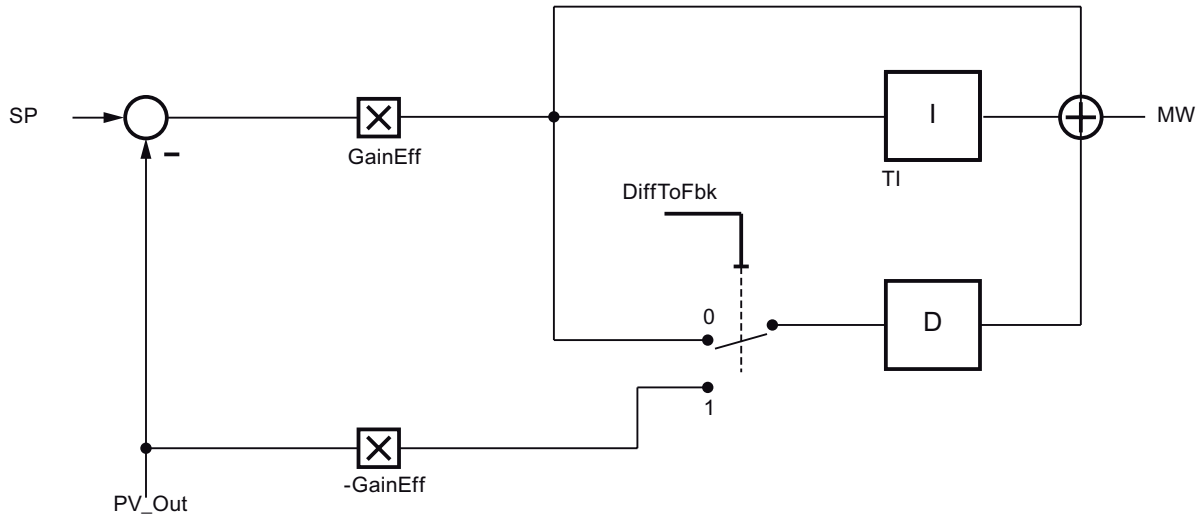


Switching derivative action into the feedback path

The parameter `DiffToFbk` can be used to switch the D action into the feedback path. Setpoint jumps then no longer affect the derivative action directly.

`DiffToFbk = 0`: Derivative action is not in the feedback path (default setting).

`DiffToFbk = 1`: Derivative action is in the feedback path.



1.2.6 Forcing operating states

Forcing operating states

The forcing of operating states function lets you set the function block into a different operating state using interconnectable input parameters, regardless of the currently active control. This can, for example, be:

- Forced tracking in closed-loop controllers
- Enabling and disabling at motors
- Opening and closing of valves

Forcing of operating states is only possible in the manual and automatic modes. Forcing operating states has the highest priority over both operating modes.

Note

It is not possible to force operating states in local mode.

Forcing operating states at closed-loop controllers

In control engineering, this procedure is also known as forced tracking of values. Please also refer to the section Tracking and limiting a manipulated value (Page 111).

Forcing operating states at motors and valves

The input parameter `xxxxForce = 1` (for example `OpenForce` and `CloseForce` at a valve) is used for forced controlling of the function block and thus an intervention in the function of the block, irrespective of currently active controls and interlock conditions. If the input parameters are inconsistent (for example `OpenForce = 1` and `CloseForce = 1` at valves), an error number (Page 117) is output at the parameter `ErrorNum` and the control remains unchanged.

Display in the faceplate and in the block icon

If an operating state is forced, this is displayed in the block icon and in the standard view of the faceplate:

Block icon: In the block icon, the display for motors, valves and dosers involves the use of a red F and a crossed-out padlock.

There is no display for closed-loop controllers.

Faceplate: An information text on the forced operating state is displayed in the standard view of the faceplate, for example, "Forced stop" at motors. This is also indicated by a crossed-out padlock:

Messaging

No messages are assigned to the forcing of operating states. However, if you want to have corresponding messages, you can use the freely interconnectable input parameters to generate the messages. See also the Generating instance-specific messages (Page 46) section for more on this.

1.2.7 Error handling

Error handling

The driver and technological blocks feature error handling routines. A distinction must be made between the following areas:

- Error numbers
- External control system fault (CSF)
- Process-specific errors
- Invalid signal states
- Mode changeover error
- Error in channel driver blocks (e.g. Pcs7AnIn)
- Error in channel driver blocks (e.g. FbAnIn)

Error numbers

Most blocks have an output parameter `ErrorNum` that can be used to output internal error states of the block as error numbers.

With some blocks, input parameters are checked for permissible values.
They are therefore only used to prevent the output value from remaining invalid when the input value is once again in the valid range. If an invalid value is detected,
and the corresponding output value is held at the last displayed value instead of an invalid value being displayed. If blocks do not have this check, an invalid value can appear at the output. However, a valid value is displayed again at the output as soon as the input values of the block have changed correspondingly.

Any value set over an interconnection or as a result of a parameter assignment that is outside the range of values (e.g. "Not a Number") is not processed by the block algorithm. The last valid value is processed instead.

In addition to the errors stated above, a limit violation is also signaled for example. Each error number is assigned to a specific error.

If there is more than one error, all error numbers have the same priority. The routine always displays the error number of the error most recently detected in a block cycle.

External control system fault (CSF)

An external control system fault always lies outside the process - it exists in the form of device or other hardware faults. If, for example, a run-time error occurs at a valve, there is an error or fault in the pneumatic system.

A control system fault is output if an external fault is set at the input `CSF`. You can enable this output function, for example, by interconnecting output `Bad` of the driver block with input `CSF` of the technological block.

The error message "$$BlockComment$$ External error occurred" is output at `CSF = 1`.

This state is visualized in the group display by an "S" character in the faceplate overview and in the block icon.

Process-specific errors

Process-specific errors can have the following causes:

- Runtime monitoring: If the feedback signals do not match the control settings after a selected time has expired, a process-related error is output.
- Feedback monitoring: Please refer to the section Monitoring the feedbacks (Page 83).

If the block algorithm detects a monitoring error while monitoring is enabled, the corresponding output parameter is set to 1 in the block. The "$$BlockComment$$ Feedback error xxx" error message is also output, where xxx, for example, stands for the valve.

This state is visualized in the group display by an "S" character in the faceplate overview and in the block icon.

The block must be reset after the monitoring error was cleared and if automatic mode is set.

Invalid input signals

This error is output if inconsistencies are detected between associated I/Os. The close and open commands cannot be output simultaneously to the valve, for example.

If the block algorithm detects an invalid combination of input signals, an error number (`ErrorNum`) is output that depends on the block type.

The text "Invalid signal" is output in the standard view of the faceplate.

Mode changeover error

This error is reported if you change the mode of the block from:

- Manual to automatic mode or
- Local mode to automatic mode

and the previous and target state are **inconsistent** (bumpless changeover). You can only change the block mode if the subsequent state corresponds with the previous state.

Bumpless changeover can be activated / deactivated using the `Feature` connection on the Enabling bumpless changeover to automatic mode for valves, motors, and dosers (Page 197) or Disabling bumpless changeover to automatic mode for controllers (Page 197) bit.

Bumpless changeover from local to automatic mode is undertaken using the `LocalSetting` parameter, as described in section Local mode (Page 36).

In the standard view of the faceplate, the text "Mode switch fail" is displayed in the event of an unwanted changeover with bumps.

The block retains local mode if the operator changes the mode from local to automatic and the error mentioned above occurs. The block changes to manual mode if the mode is changed from local to automatic over interconnected inputs and the error mentioned above occurs.

Error in channel driver blocks (e.g. Pcs7AnIn)

The following errors can be displayed by the channel driver blocks:

- Channel error
- Higher-level error
- Invalid measuring range

Error in channel driver blocks (e.g. FbAnIn)

The following errors can be displayed by the driver blocks:

- Device or module fault
- Higher-level error
- Invalid measuring range

1.2.8 Resetting the block in case of interlocks or errors

Resetting the block

A block reset is only relevant in automatic mode. It has no influence in manual mode or in local mode and is disabled in these modes.

The block must be reset when an interlock has been set via the `Protect` input or an error has occurred.

There are different ways to reset the block:

- Reset by interconnection (input `RstLi`).
- Reset by the operator using a button in the faceplate (input `RstOp`).
- Reset with a 0-1 edge change in the relevant automatic signal.

The operator must have the appropriate authorization to use the reset function in the faceplate (`OS_Perm`). To enable a block reset via the 0-1 edge change in the automatic signal, the response must be set in the parameters for the `Feature` bit Resetting via input signals in the event of interlocks or errors (Page 194). Refer to the relevant block description.

1.2.9 Safe position for motors, valves and controllers

Safe position for motors, valves and controllers

The safe position always represents the deenergized state.

Safe position for motors

Safe position for motors is always the stopped motor.

Safe position for valves

There are different forms of the deenergized state for valves:

- Valve is closed at a deenergized state
- Valve is open at a deenergized state
- Valve is stopped at a deenergized state (motor valve)

The input parameter `SafePos` is used to set these properties of the valve:

- `SafePos = 0`: Valve is closed at a deenergized state
- `SafePos = 1`: Valve is open at a deenergized state
- `SafePos = 2`: Valve is stopped in a deenergized state (motor valve)

The safe position is adopted when:

- The runtime monitoring function was addressed (see Setting the startup response (Page 187))
- One of the interlock conditions is active (see Interlocks (Page 100))

The safe position is adopted if at least one of these interlock conditions is present ("Protection" `[Protect]` or "Interlock" `[Intlock]`, see Interlocks (Page 100)).

Note

The motor valve will not be put into the safe position for monitoring errors (Runtime error and Control error).

Safe position for continuous controller (does not apply to controller modules)

Only the limits for the manual value are taken into consideration for the safe position with continuous controllers. The input parameter `SafePos` is used to specify the safe position:

- `SafePos = 0` corresponds to the low limit (`ManLoLim`)
- `SafePos = 1` corresponds to the high limit (`ManHiLim`)

The safe position is adopted:

- during start-up if the `Feature` bit Setting the startup response (Page 187) and the `Feature` bit Safety value of the manipulated variable effective at startup (Page 196) are set.
- in the "Out of service" mode if the `Feature` bit Safety manipulated variable with "out of service" operating mode in effect (Page 196) is set.

Safe position for step controller (does not apply to controller modules)

You can use the input parameter `SafePos` to determine if the step controller should close, open or stop the valve when it travels to the safe position:

`SafePos = 0`: close valve

`SafePos = 1`: open valve

`SafePos = 2`: stop valve

When the safe position (fully opened or fully closed) is reached and a limit signal (`FbkOpened` or `FbkClosed`) is set, the valve is stopped (Stop = true).

The safe position is adopted:

- during start-up if the `Feature` bit Setting the startup response (Page 187) and the `Feature` bit Safety value of the manipulated variable effective at startup (Page 196) are set.
- in the "Out of service" mode if the `Feature` bit Safety manipulated variable with "out of service" operating mode in effect (Page 196) is set.

Safety control for controllers of the FM 355 or FM 355-2 modules

The controller modules have their own mechanism for feedforwarding a safety value (see Temperature Controller FM 355-2 manual or Controller Module FM 355 manual)

1.2.10 Specifying warning times for control functions at motors and valves

Specifying warning times for control functions at motors and valves

You can generate warning signals when, for example, motors are started or valves are opened. Warning signals can be generated in the following modes:

- Manual mode (input parameter `WarnTiMan`)
- Automatic mode (input parameter `WarnTiAut`)

You specify the warning times in seconds using the input parameters `WarnTiMan` and `WarnTiAut`. If, for example, a motor is started then, this is displayed at the output parameter with `WarnAct = 1`. The motor then starts after the set warning time has expired.

A corresponding warning is not output if the warning times (`WarnTiMan` or `WarnTiAut`) are specified with a smaller value than the `SampleTime` parameter.

Disabling warnings

Configure each parameter with 0 seconds to generate no warnings.

1.2.11 SIMATIC BATCH functionality

SIMATIC BATCH functionality

Some blocks have an interface to SIMATIC BATCH. You use them when you connect `BatchEn`, `BatchID`, `BatchName`, `StepNo` and `Occupied` I/Os to the corresponding SIMATIC BATCH blocks. Refer to the SIMATIC BATCH documentation.

Please refer to the descriptions of the individual blocks for information about whether a block supports the SIMATIC BATCH functionality.

1.3 General functions of the faceplates

1.3.1 Structure of the faceplate

General functions of the faceplates

This section provides general information that applies to all faceplates.

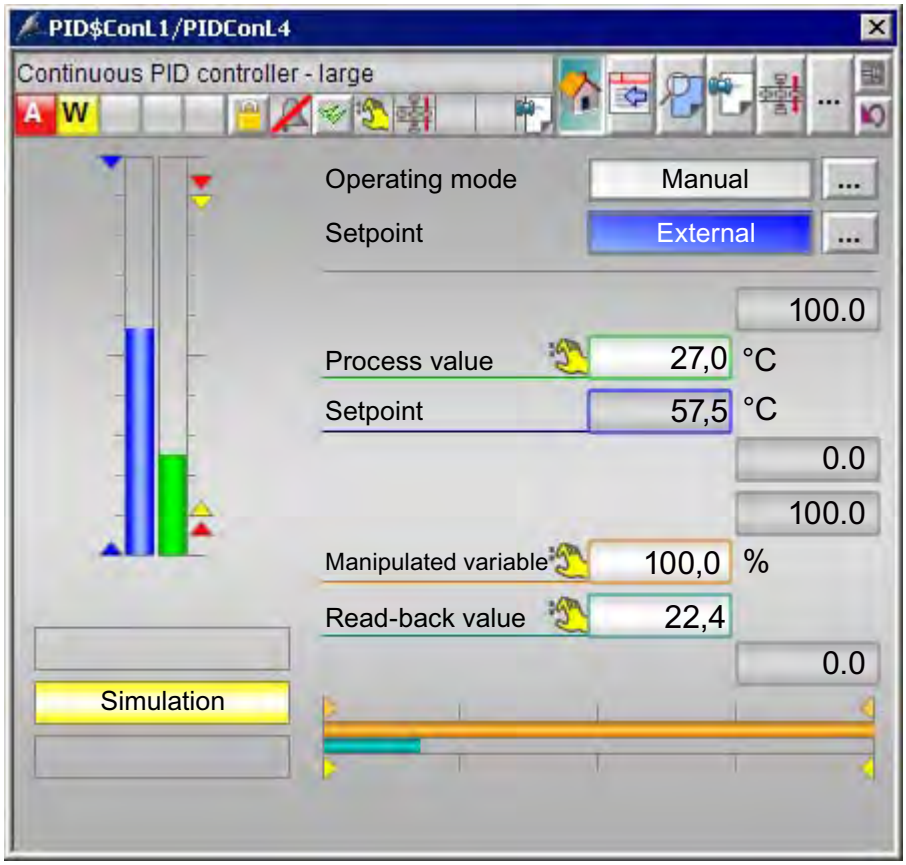
Recommended screen resolution

The faceplates are shown in full on the screen with a resolution of 1280 x 1024.

Use the F11 key to switch to full screen mode if the screen resolution is set lower.

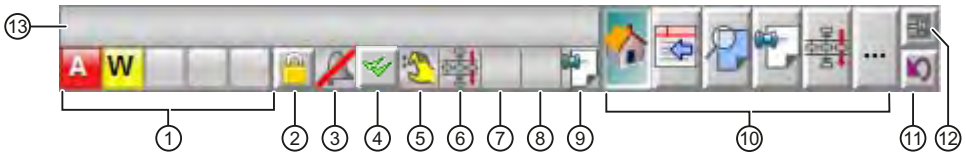
Opening the faceplate

Click on the block icon in WinCC to open the faceplate with the standard view; in this example this is the standard view of PIDConL:



The views differ depending on the block functions. All blocks that have faceplates provide a status bar where you can see the most important information relating to the block status. There are additional functions available that are described in the next sections.

Displays and operator controls



The faceplate provides the following display and operator controls:

(1) Group display

(2) Lock alarms

(3) Suppress alarms

(4) Acknowledge alarms

(5) Worst signal status

(6) Batch display

(7) Not used

(8) Maintenance request and release

(9) Memo display

(10) Open views of the block

(11) Return to block icon

(12) Pin block

(13) Instance name of the block

(1) Group display

The group display shows the information that is transferred from ALARM_8P of the block instance to WinCC.

- Alarms
- Warnings
- Tolerances
- Faults
- Operator prompts

(2) Lock/unlock alarms

You can use this button to lock or unlock block alarms.

If you lock the alarms of the block, it is shown in the group display of alarms as a white x. When alarms are locked, the block instance does not send any new alarms. Previous alarms are also no longer displayed in the alarm view of the faceplate.

The alarms are displayed again in the group view when you unlock the block's alarms. The block instance then resumes sending new alarms. Alarms generated in the locked phase are displayed when you enable the alarm function.

You can hide this button from specific users / user groups using the permissions in PCS 7-OS. Refer to the PCS 7-OS help system.

(3) Suppress alarms

Alarm suppression indicates whether or not the "Suppress process alarms" function in the AS block is activated with the `MsgLock` parameter. If alarm suppression is activated, all alarms in this block instance – except for process control alarms – are suppressed.

(4) Acknowledge alarms

You can acknowledge all alarms from the block instance using this button.

You can hide this button from specific users / user groups using the permissions in PCS 7-OS. Refer to the *Process Control System PCS 7; OS Process Control* documentation for more on this.

(5) Display for worst signal status

This display shows the worst signal status currently present. Refer to the Forming and outputting signal status for blocks (Page 88) section for more on this.

(6) Batch display

The batch display shows whether or not the block instance is in use by SIMATIC BATCH.

Refer to the SIMATIC BATCH functionality (Page 122) section for more on this.

(7) Not used

This field is currently not used.

(8) Maintenance request and release

This display shows you if a maintenance request or release has been made for this block.

Refer to the Maintenance release (Page 47) section for more on this.

(9) Memo display

This display shows you if a message has been left in the memo view for you. Refer to the Memo view (Page 171) section for more on this.

(10) Open views of a block

You can use this field to open the various views of a block. Refer to the block description to learn of the available views.

You can select from the following typical views here:

Icon	Term
	Standard view
	Alarm view
	Limits view (several limit value views within a block are possible)
	Trend view
	Ramp view
	Parameter view (several parameter views within a block are possible)
	Preview
	Memo view
	Batch view

Note

Some blocks feature block-specific views. They are not listed here.

(11) Return to block icon

Use this button to return to the block icon in the process image of the corresponding faceplate. You can use this function, for example, when you have pinned a block **(12)** and the process picture has changed in the meantime.

(12) Pin block

You can pin the faceplate on top of the user interface using this button. This allows you to switch to another picture or area without closing the faceplate.

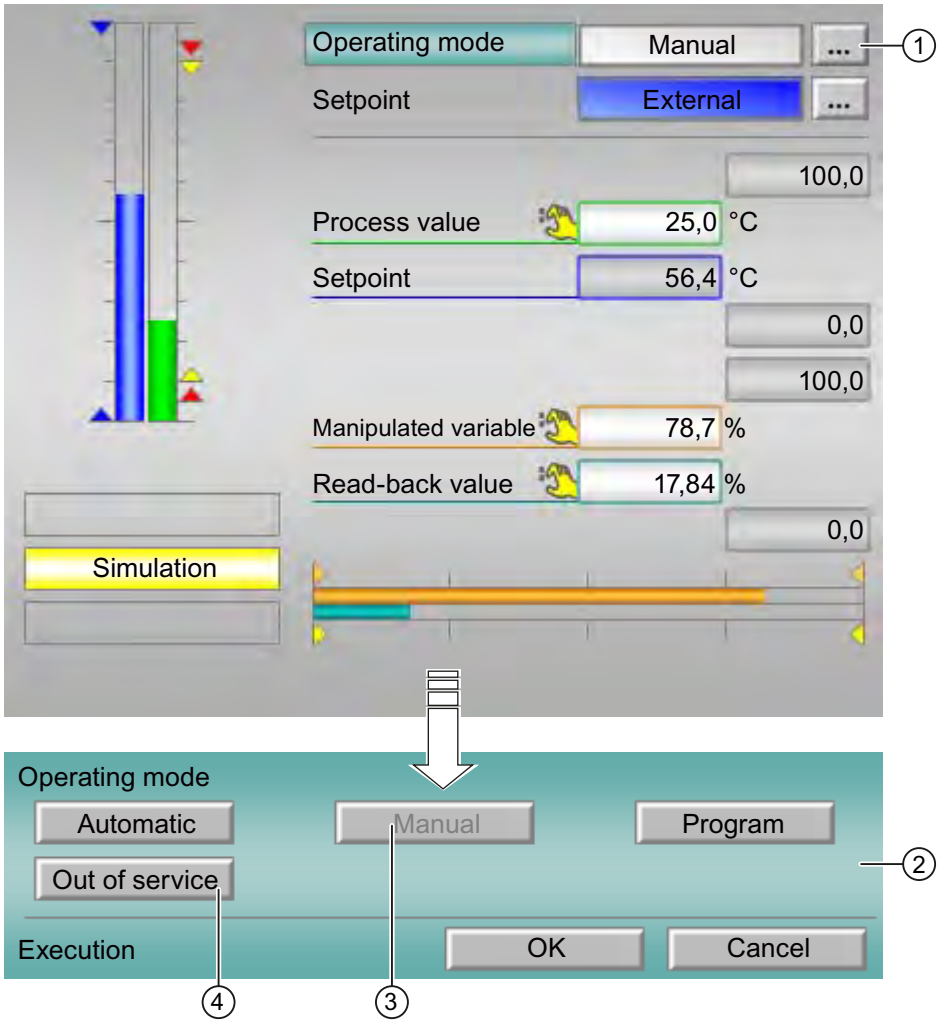
Note

You can learn about additional operator controls in the descriptions of the individual blocks.

1.3.2 Switching operating states and operating modes

Requirements

You can change the operating state, operating mode and other parameters if needed in faceplates if you have the corresponding operator control permission (`OS_Perm`). This operator control permission can be configured in the engineering system (ES).



(1) The mouse pointer changes when you place it over the following button:

The mouse pointer now looks as follows:

When you click on the button with the mouse pointer, the bottom of the faceplate expands. You now see the field for changing the operating mode, for example.

(2) Field for changing the operating mode, operating state etc. This example describes changing the operating mode.

(3) The text on this button is gray. You cannot select this operating mode due to the following reasons:

- This operator control permission for this operating mode cannot be configured in the engineering system (ES).

or

- The operating mode is already selected at this time.

or

- Due to the technology, you cannot switch from the operating mode currently set and the desired operating mode.

(4) The text on this button is black. You can switch to this operating mode.

How to change the operating mode (using the PIDConL block in standard view as an example)

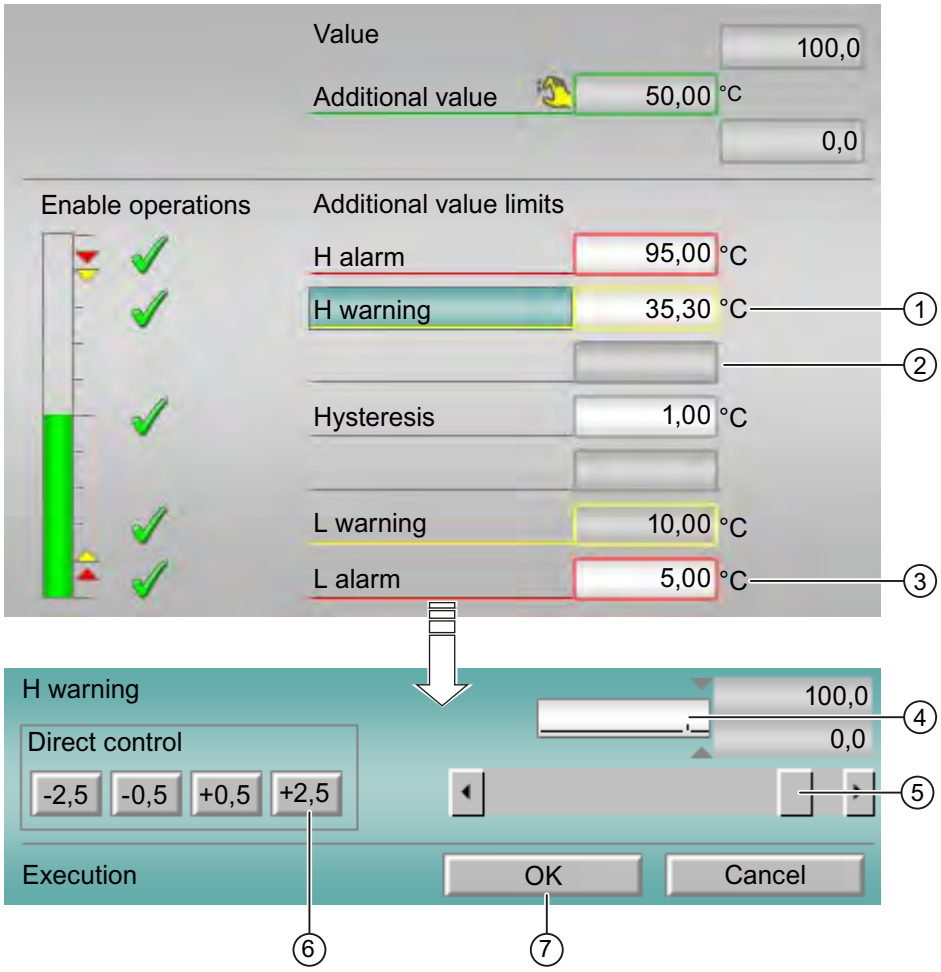
1. Click one of the selectable buttons in the operating mode field.
2. Confirm your selection by clicking "OK".
3. If you do not want to apply your selection, click "Cancel".

After clicking the "OK" or "Cancel" button, the faceplate is reduced again to its original form.

1.3.3 Changing values

Requirements

You can change the values in faceplates and in the block icons if you have the corresponding operator control permission (`OS_Perm`). This operator control permission can be configured in the engineering system (ES). The following example shows how values are changed via the faceplate.



(1) The background color of the input box is white. You can change the value. The mouse pointer changes when you place it over the input box: .

(2) The background color of the input box is gray. You cannot change the value.

(3) If you click on the input box, the bottom of the faceplate expands. You now see the field for changing values.

How to change values in faceplates

You have three options for changing values:

- Direct control: Click on a button such as "-2.5" in the box **(6)**. The value is immediately changed and applied. It is no longer necessary to confirm this value.
- Changing values using the slide control **(5)**: move the slide control until the desired value is shown in the box. Then confirm the value using the enter key or by clicking "OK". Read the section "Setting the multiple step operation".
- Change the values in the input box **(4)**: Click the input box and enter the new value. Then confirm the value using the enter key or by clicking "OK". Read the section "Setting the multiple step operation".

Setting the multiple step operation

You can use the internal @APLCommandExecutionSteps tag in the Tag Management of the WinCC Explorer to specify if values are to be changed in two or three steps.

Follow the steps outlined below:

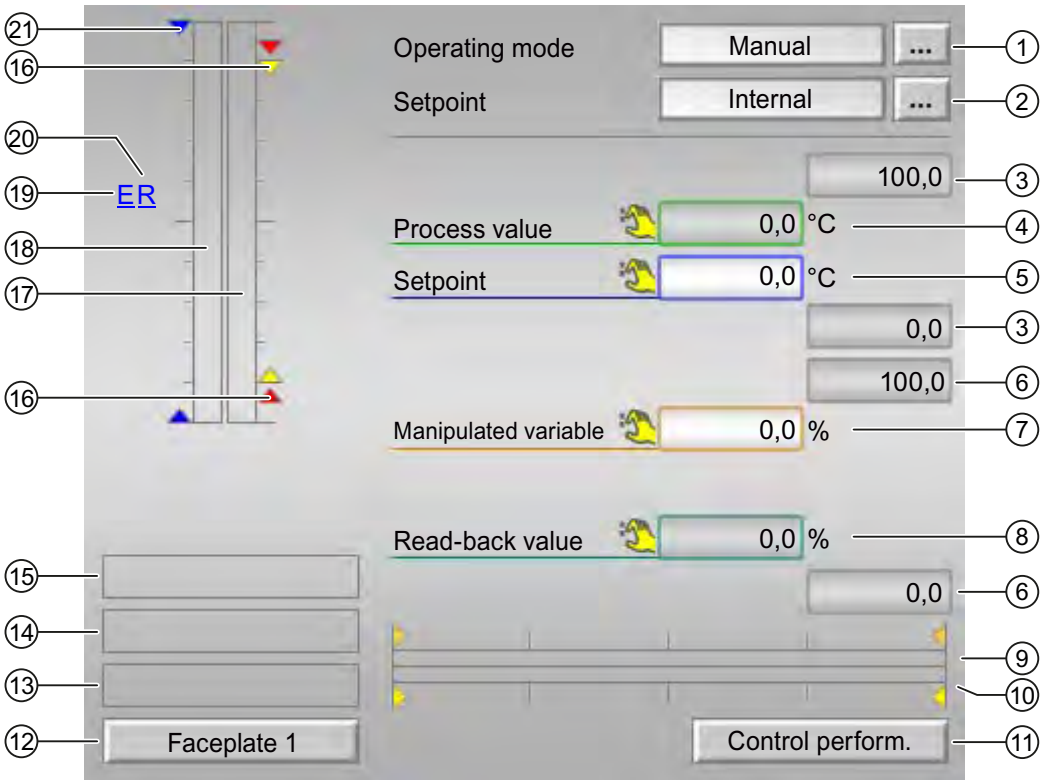
1. Double-click on the internal @APLCommandExecutionSteps tag
2. Change the start value to 2 or 3 in the Limits/Reporting tab.

   Start value = 2: It is no longer necessary to confirm the value in the faceplate by clicking "OK", values are applied immediately.

   Start value = 3: Each value change in the faceplate, except for direct operation, needs to be confirmed with "OK".

1.3.4 Standard view of FM controllers (analog)

Standard view (analog) of FM controllers



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 29)
- Automatic mode (Page 29)
- Program mode for closed-loop controllers (Page 34)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Display and switch the setpoint specification

This area shows how to specify the setpoint. The setpoint can specified as follows:

- By the application (External, CFC/SFC)
- By the user direct in the faceplate (Internal).

Refer to the Switching operating states and operating modes (Page 127) section for information on changing the setpoint specification.

For more information on the setpoint specification, refer to the Setpoint input - internal and external (Page 96) section.

(3) High and low scale range for the process value

These values provide information on the display range (`PV_OpScale`) for the bar graph of the process value. The scale range is defined in the engineering system.

(4) Display of the process value including signal status

This area provides information on the current process value (`PV`) with the corresponding signal status.

(5) Display and change the setpoint including signal status

This area provides information on the current setpoint (`SP`) with the corresponding signal status.

Refer to the Changing values (Page 129) section for information on changing the setpoint. The setpoint specification **(2)** also needs to be set to "Internal" for this block.

(6) High and low scale range for the setpoint

This area is already set and cannot be changed.

(7) Display and change the manipulated variable including signal status

This area provides information on the current manipulated variable (`MV`) with the corresponding signal status.

Refer to the Changing values (Page 129) section for information on changing the manipulated variable. You can only make a change in manual mode.

(8) Display of the position feedback including signal status

This area provides information on the current readback value of the manipulated variable with the corresponding signal status. This display is only available when the readback value in the box is interconnected to the `Rbk` input parameter.

(9) Bar graph for the manipulated variable

This area provides information on the current manipulated variable in the form of a bar graph (`MV_OpScale`). The visible area in the bar graph depends on the configuration in the engineering system (ES).

(10) Bar graph for the position feedback

This area shows the current position feedback in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(11) Navigation button for switching to the standard view of the ConPerMon block

Use this navigation button to reach the standard view of the ConPerMon block. The visibility of this navigation button depends on the configuration in the engineering system (ES).

See also the Calling further faceplates (Page 43) section for more on this.

(12) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

See also the Calling further faceplates (Page 43) section for more on this.

(13) Display area for states of the block

This area provides additional information on the operating state of the block:

- Maintenance

Additional information on the display area for states of the block is available in section Maintenance release (Page 47).

(14) Display area for states of the block

This area provides additional information on the operating state of the block:

- Simulation

Additional information is available in the section Simulating signals (Page 93).

(15) Display area for states of the block

This area provides additional information on the operating state of the block (from high to low according to priority):

- Fuzzy Optim. (FmCont only)
- Tracking FB
- Tracking FM
- Safe position FM
- Fuzzy control (FmCont only)

(16) Limit display

These colored triangles indicate the configured limits in the respective bar graph:

- Red: Alarm
- Yellow: Warning
- Blue: Tolerance

(17) Bar graph for the process value

This area shows the current process value in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(18) Bar graph for the setpoint

This area shows the current setpoint in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(19) Display for external setpoint

This display [E] is only visible when you have selected "internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

(20) Display for the target setpoint of the setpoint ramp

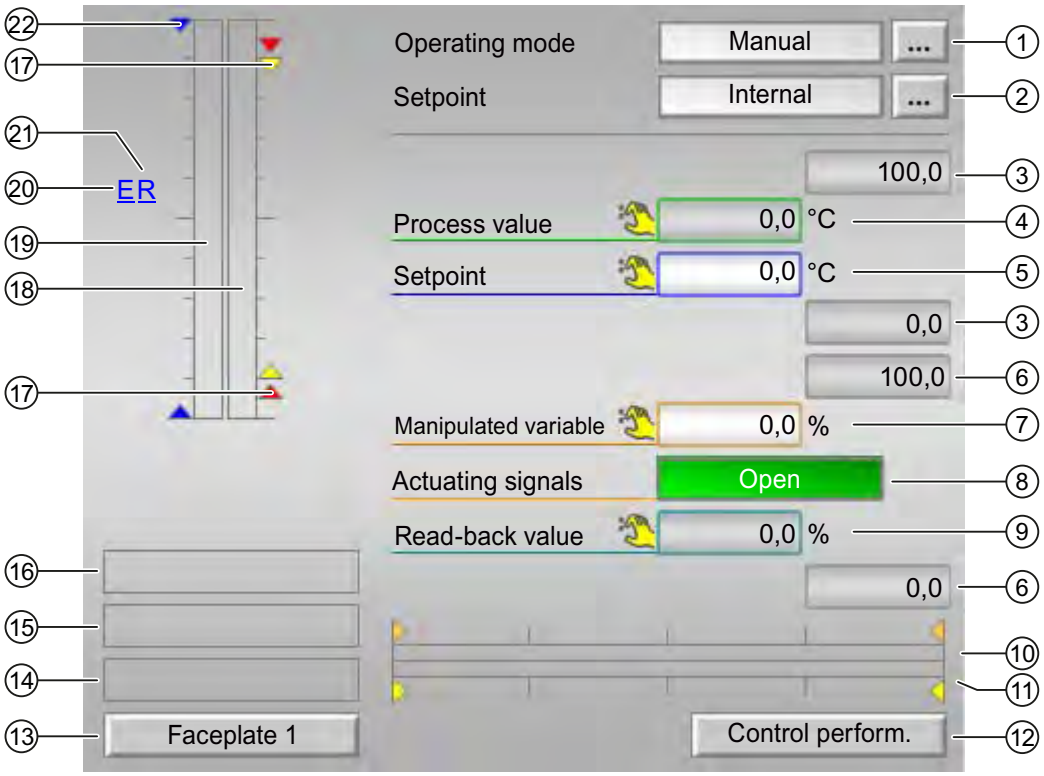
This display [R] shows you the target setpoint and is only visible if you have enabled ramp generation in the Ramp view (Page 167).

(21) Limit display for the setpoint

These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the ES.

1.3.5 Standard view of FM controllers (pulse controller)

Standard view (pulse) of FM controllers



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 29)
- Automatic mode (Page 29)
- Program mode for closed-loop controllers (Page 34)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Display and switch the setpoint specification

This area shows how to specify the setpoint. The setpoint can specified as follows:

- By the application (External, CFC/SFC)
- By the user direct in the faceplate (Internal).

Refer to the Switching operating states and operating modes (Page 127) section for information on changing the setpoint specification.

For more information on the setpoint specification, refer to the Setpoint input - internal and external (Page 96) section.

(3) High and low scale range for the process value

These values provide information on the display range (`PV_OpScale`) for the bar graph of the process value. The scale range is defined in the engineering system.

(4) Display of the process value including signal status

This area provides information on the current process value (`PV`) with the corresponding signal status.

(5) Display and change the setpoint including signal status

This area provides information on the current setpoint (`SP`) with the corresponding signal status.

Refer to the Changing values (Page 129) section for information on changing the setpoint. The setpoint specification **(2)** also needs to be set to "Internal" for this block.

(6) High and low scale range for the setpoint

This area is already set and cannot be changed.

(7) Display and change the manipulated variable including signal status

This area provides information on the current manipulated variable (`MV`) with the corresponding signal status.

Refer to the Changing values (Page 129) section for information on changing the manipulated variable. You can only make a change in manual mode.

(8) Display of the actuating signal

This area shows the current feedback of the actuating signal.

- Open
- Close

(9) Display of the position feedback including signal status

This area shows the current feedback of the manipulated variable with the corresponding signal status. This display is only available when the readback value in the box is interconnected to the Rbk input parameter.

(10) Bar graph for the manipulated variable

This area provides information on the current manipulated variable in the form of a bar graph (MV_OpScale). The visible area in the bar graph depends on the configuration in the engineering system (ES).

(11) Bar graph for the position feedback

This area shows the current position feedback in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(12) Navigation button for switching to the standard view of the ConPerMon block

Use this navigation button to reach the standard view of the ConPerMon block. The visibility of this navigation button depends on the configuration in the engineering system (ES).

See the section Calling further faceplates (Page 43) for more information.

(13) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

See the section Calling further faceplates (Page 43) for more information.

(14) Display area for states of the block

This area provides additional information on the operating state of the block:

- Maintenance

Additional information on the display area for states of the block is available in section Maintenance release (Page 47).

(15) Display area for states of the block

This area provides additional information on the operating state of the block:

- Simulation

Additional information is available in the section Simulating signals (Page 93).

(16) Display area for states of the block

This area provides additional information on the operating state of the block (from high to low according to priority):

- Fuzzy Optim. (FmCont only)
- Tracking FB
- Tracking FM
- Safe position FM
- Fuzzy control (FmCont only)

(17) Limit display

These colored triangles indicate the configured limits in the respective bar graph:

- Red: Alarm
- Yellow: Warning
- Blue: Tolerance

(18) Bar graph for the process value

This area shows the current process value in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(19) Bar graph for the setpoint

This area shows the current setpoint in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(20) Display for external setpoint

This display [E] is only visible when you have selected "internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

(21) Display for the target setpoint of the setpoint ramp

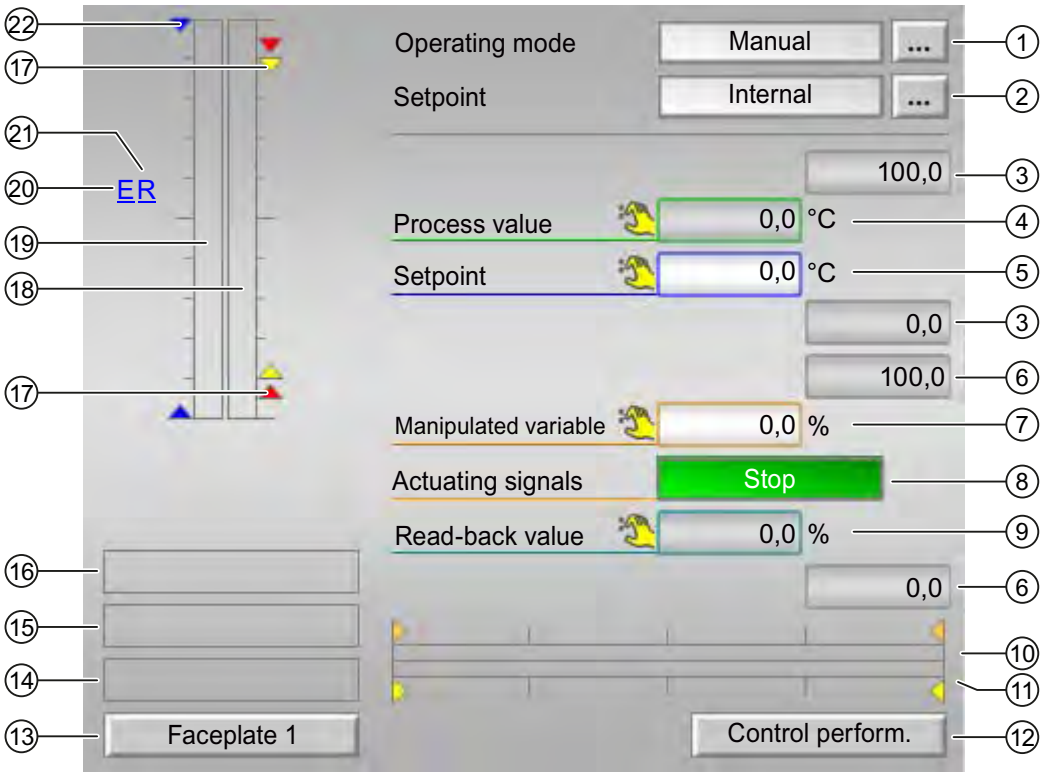
This display [R] shows the target setpoint and is only visible if you have enabled ramp generation in the Ramp view (Page 167).

(22) Limit display for the setpoint

These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the ES.

1.3.6 Standard view of FM controllers (step controller with position feedback)

Standard view with position feedback of FM controllers



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 29)
- Automatic mode (Page 29)
- Program mode for closed-loop controllers (Page 34)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Display and switch the setpoint specification

This area shows how to specify the setpoint. The setpoint can specified as follows:

- By the application (External, CFC/SFC)
- By the user direct in the faceplate (Internal).

Refer to the Switching operating states and operating modes (Page 127) section for information on changing the setpoint specification.

For more information on the setpoint specification, refer to the Setpoint input - internal and external (Page 96) section.

(3) High and low scale range for the process value

These values provide information on the display range (`PV_OpScale`) for the bar graph of the process value. The scale range is defined in the engineering system.

(4) Display of the process value including signal status

This area provides information on the current process value (`PV`) with the corresponding signal status.

(5) Display and change the setpoint including signal status

This area provides information on the current setpoint (`SP`) with the corresponding signal status.

Refer to the Changing values (Page 129) section for information on changing the setpoint. The setpoint specification **(2)** also needs to be set to "Internal" for this block.

(6) High and low scale range for the setpoint

This area is already set and cannot be changed.

(7) Display and change the manipulated variable including signal status

This area provides information on the current manipulated variable (`MV`)with the corresponding signal status.

Refer to the Changing values (Page 129) section for information on changing the manipulated variable. You can only make a change in manual mode.

(8) Display of the actuating signal

This area shows the current feedback of the actuating signal.

- Open
- Stop
- Close

(9) Display of the position feedback including signal status

This area shows the current feedback of the manipulated variable with the corresponding signal status. This display is only available when the readback value in the box is interconnected to the Rbk input parameter.

(10) Bar graph for the manipulated variable

This area provides information on the current manipulated variable in the form of a bar graph (MV_OpScale). The visible area in the bar graph depends on the configuration in the engineering system (ES).

(11) Bar graph for the position feedback

This area shows the current position feedback in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(12) Navigation button for switching to the standard view of the ConPerMon block

Use this navigation button to reach the standard view of the ConPerMon block. The visibility of this navigation button depends on the configuration in the engineering system (ES).

See the section Calling further faceplates (Page 43) for more information.

(13) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

See the section Calling further faceplates (Page 43) for more information.

(14) Display area for states of the block

This area provides additional information on the operating state of the block:

- Maintenance

Additional information on the display area for states of the block is available in section Maintenance release (Page 47).

(15) Display area for states of the block

This area provides additional information on the operating state of the block:

- Simulation

Additional information is available in the section Simulating signals (Page 93).

(16) Display area for states of the block

This area provides additional information on the operating state of the block (from high to low according to priority):

- Fuzzy Optim. (FmCont only)
- Tracking FB
- Tracking FM
- Safe position FM
- Fuzzy control (FmCont only)

(17) Limit display

These colored triangles indicate the configured limits in the respective bar graph:

- Red: Alarm
- Yellow: Warning
- Blue: Tolerance

(18) Bar graph for the process value

This area shows the current process value in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(19) Bar graph for the setpoint

This area shows the current setpoint in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(20) Display for external setpoint

This display [E] is only visible when you have selected "internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

(21) Display for the target setpoint of the setpoint ramp

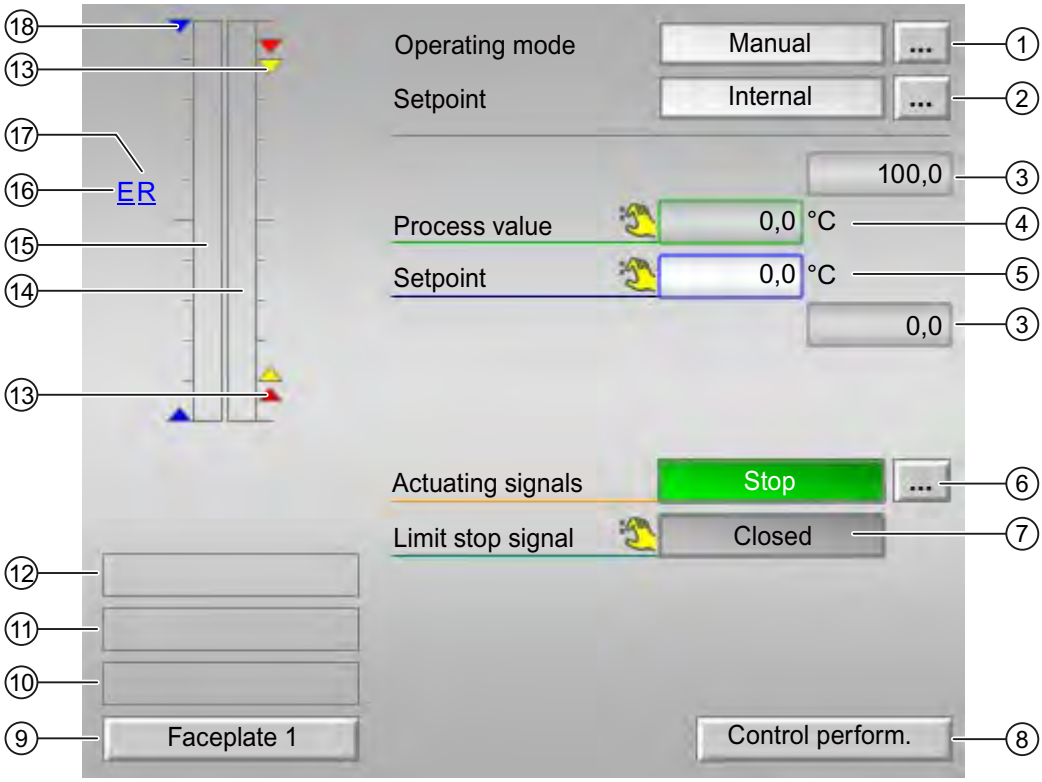
This display [R] shows the target setpoint and is only visible if you have enabled ramp generation in the Ramp view (Page 167).

(22) Limit display for the setpoint

These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the ES.

1.3.7 Standard view of FM controllers (step controller without position feedback)

Standard view without position feedback of FM controllers



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 29)
- Automatic mode (Page 29)
- Program mode for closed-loop controllers (Page 34)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Display and switch the setpoint specification

This area shows how to specify the setpoint. The setpoint can specified as follows:

- By the application (External, CFC/SFC)
- By the user direct in the faceplate (Internal).

Refer to the Switching operating states and operating modes (Page 127) section for information on changing the setpoint specification.

For more information on the setpoint specification, refer to the Setpoint input - internal and external (Page 96) section.

(3) High and low scale range for the process value

These values provide information on the display range (`PV_OpScale`) for the bar graph of the process value. The scale range is defined in the engineering system.

(4) Display of the process value including signal status

This area provides information on the current process value (PV) with the corresponding signal status.

(5) Display and change the setpoint including signal status

This area provides information on the current setpoint (SP) with the corresponding signal status.

Refer to the Changing values (Page 129) section for information on changing the setpoint. The setpoint specification **(2)** also needs to be set to "Internal" for this block.

(6) Operating and displaying the actuating signal

This area shows the current feedback of the actuating signal.

- Open
- Stop
- Close

The button is shown next to the display in manual mode. You can influence the actuating signal here. Refer to the Switching operating states and operating modes (Page 127) section for more on this.

(7) Display of the limit stop value including signal status

This area shows the limit stop signal with the corresponding signal status.

- Open
- Closed

(8) Navigation button for switching to the standard view of the ConPerMon block

Use this navigation button to reach the standard view of the ConPerMon block. The visibility of this navigation button depends on the configuration in the engineering system (ES).

See the section Calling further faceplates (Page 43) for more information.

(9) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

See the section Calling further faceplates (Page 43) for more information.

(10) Display area for states of the block

This area provides additional information on the operating state of the block:

- Maintenance

Additional information on the display area for states of the block is available in section Maintenance release (Page 47).

(11) Display area for states of the block

This area provides additional information on the operating state of the block:

- Simulation

Additional information is available in the section Simulating signals (Page 93).

(12) Display area for states of the block

This area provides additional information on the operating state of the block (from high to low according to priority):

- Fuzzy Optim. (FmCont only)
- Tracking FB
- Tracking FM
- Safe position FM
- Fuzzy control (FmCont only)

(13) Limit display

These colored triangles indicate the configured limits in the respective bar graph:

- Red: Alarm
- Yellow: Warning
- Blue: Tolerance

(14) Bar graph for the process value

This area shows the current process value in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(15) Bar graph for the setpoint

This area shows the current setpoint in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(16) Display for external setpoint

This display [E] is only visible when you have selected "internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

(17) Display for the target setpoint of the setpoint ramp

This display [R] shows the target setpoint and is only visible if you have enabled ramp generation in the Ramp view (Page 167).

(18) Limit display for the setpoint

These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the ES.

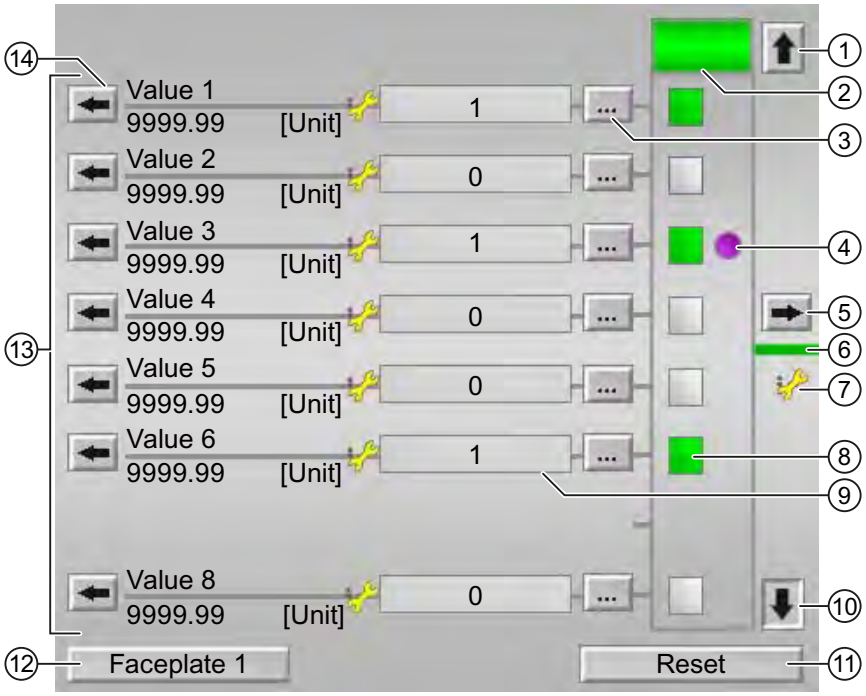
1.3.8 Standard view of interlock blocks

Standard view of interlock blocks Intlck02, Intlck04, Intlck08, Intlck16

The number of the displayed input values depends on the interlock block you have selected.

The operation and functions are identical for all interlock blocks and do not depend on the number of input values.

The **Intlck16** interlock block has two additional buttons for switching between the input values 1 to 8 and 9 to 16.



(1), (10) Switching among the input values 1 to 8 and 9 to 16 (for Intlck16 only)

The buttons (1) or (9) are displayed depending on the view you are in. These buttons are only available for the **Intlck16** block.

The **Intlck16** block provides two views:

- When you are in the first view, the input values 1 to 8 are available in the area (12). The button (9) is displayed. You switch to the second view by clicking on the (9) button.
- When you are in the second view, the input values 9 to 16 are available in the area (12). The button (1) is displayed. You switch back to the first view by clicking on the button (1).

(2) Status of the output signal of the interlock block

This area (2) shows the status of the output signal of the interlock block (priority from high to low). You can configure the logic in the engineering system (ES).

	Logic	
Color of the field	**AND**	**OR**
Gray	Interlock signal not used by the technological block.	
Blue	Excluded (bypass)	
Yellow	Simulated	
Red	Interlocked	
Green	Not interlocked	

(3) Exclude input values

You can use the button (3) to exclude input values from processing. Depending on the previous settings, you can Set or Reset this property.

If the input value has been excluded, the following icon appears in the field (8):

For more information on the operation, refer to the section Switching operating states and operating modes (Page 127).

(4) Status display First in

The following icon is displayed next to an input value, if this input value has caused the last output signal change from 1 to 0 (good state to locked):

You can reset the first-in (initial) signal with the button (10).

You can find additional information on this in the Recording the first signal for interlock blocks (Page 87) section.

For more information on the operation, refer to the section Switching operating states and operating modes (Page 127).

(5) Open faceplate of the output value

When you press the button (5), you can open the faceplate associated with the output value. The function of this button depends on the configuration in the engineering system (ES). See also the Calling further faceplates (Page 43) section for more on this.

(6) Status of the block output

The line color indicates the status of the block output:

Color of the line	Output status
Green	Output is enabled
Gray	Output is disabled

(7) Output of the signal status

This field shows you the signal status provided with the output signal of the interlock block.

(8) Display the status for further processing

The icon shows the status for further processing of the input values:

Icon	Further processing
	The input value will be further processed (value=1).
	The input value is excluded from further processing.
	The input value is not connected (value=0).

(9) Display of input values (BOOL) with signal status (in front of the field)

These fields show the interlock information associated with the analog value (13) with a signal status:

- 1 = "Good" state
- 0 = "Locked"

Changing the display

You can change the displays for 0 and 1 in the CFC in the object properties of the Interlock block:

- Navigate to the I/Os (object properties).
- Change the default for the input parameters (`Inxx`) in the Text 0 and Text 1 columns to what you later want to see in runtime.

(10) Switching input values

Read point (1) for this.

(11) Reset the settings for further processing

When you press the button (10), you can Reset all input values:

- Reset exclusions: the exclusions of the input values are reset.
- Reset first in: First-in detection / status display (4) is reset.

You can find additional information on this in the Recording the first signal for interlock blocks (Page 87) section.

(12) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

(13) Displaying analog input values

The interconnected analog input values (AVxx) are displayed in this area. Set a unit of measure (AVxx_Unit as shown in the picture with [Unit]) for each input in the engineering system (ES).

Changing the display

You can change the displays for 0 and 1 in the CFC in the object properties of the Interlock block:

- Navigate to the I/Os (object properties).
- In the Identifier column, change the default for the input parameter (AVxx) to what you later want to see in runtime.

The number of input values may vary depending on the selected interlock block:

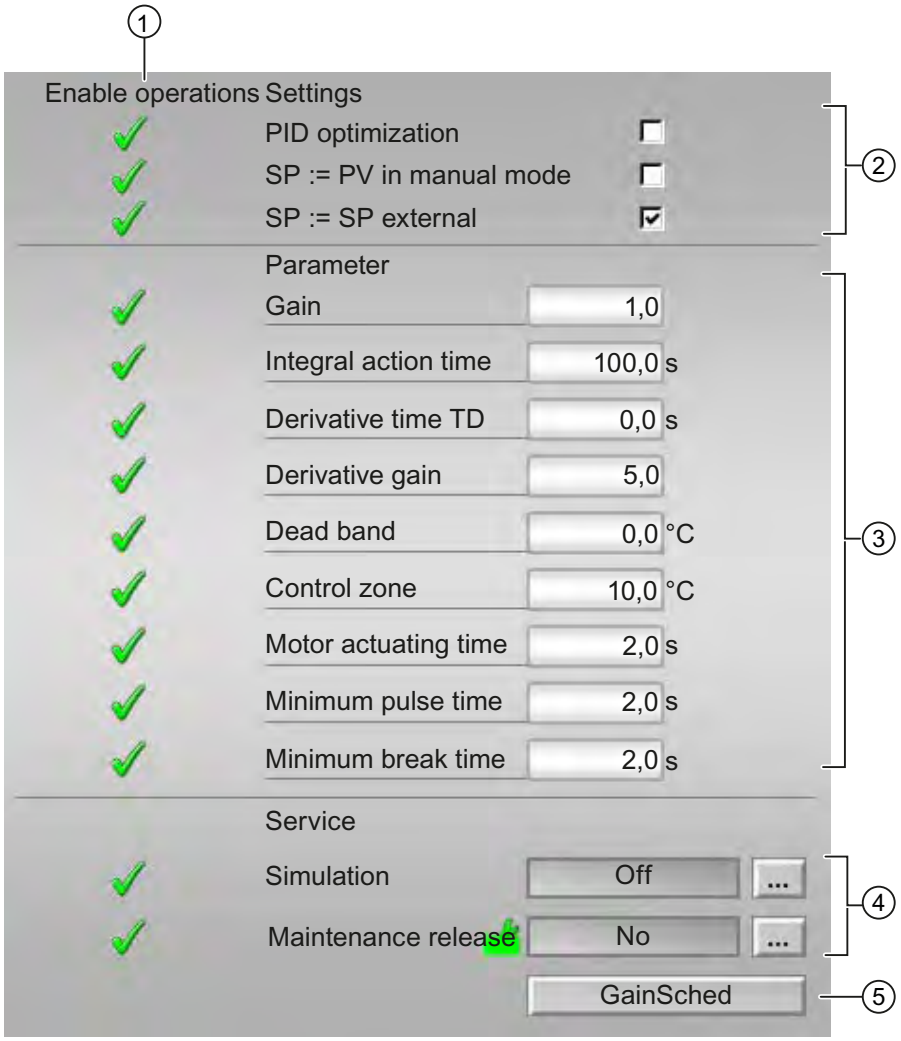
- **Intlck02:** input values 1 and 2 are available.
- **Intlck04:** input values 1 to 4 are available.
- **Intlck08:** input values 1 to 8 are available.
- **Intlck16:** input values 1 to 8 are available. The input values 9 to 16 become available by pressing the button (9). You can find additional information on this topic in the description for (1) and (9).

(14) Open faceplate of the input value

When you press the button (13), you can open the faceplate associated with each input value. The function of this button depends on the configuration in the engineering system (ES). See also the Calling further faceplates (Page 43) section for more on this.

1.3.9 Parameter view of PID controllers

Parameter view of PID controllers



(1) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions.

(2) Settings

You can activate the following functions for the controller in this area:

- PID optimization: ☑ activate controller optimization
- SP := PV in manual mode: ☑ Bumpless switchover from manual mode to automatic mode
- SP := SP external: ☑ Bumpless switchover of the setpoint for setpoint switchover from external to internal. The internal setpoint is tracked to the external one.
  - With the PIDConR block, this area is only visible if you have set the `Feature` bit Switching operator controls for external setpoint to visible (Page 189) to 1.

(3) Parameter

In this area, you change parameters and therefore influence the controller. Refer to the Changing values (Page 129) section for more on this.

You can influence the following parameters:

- Gain: Proportional gain
- Integral action time: Integral action time in [s]
- Derivative action time TD: Derivative action time in [s]
- Derivative gain: Gain of the derivative action
- Dead band: Width of dead band
- Control zone: Width of the control zone (only with PIDConL block)
- Motor actuating time: Motor actuating time [s] (for PIDStepL block only)
- Minimum pulse time: Minimum pulse time [s] (for PIDStepL block only)
- Minimum break time: Minimum break time [s] (for PIDStepL block only)

(4) Service

You can select the following functions in this area:

- Simulation
- Maintenance release (with display for a maintenance request)

Refer to the Switching operating states and operating modes (Page 127) section for more on this.

You can find information on this area in the section:

- Simulating signals (Page 93)
- Maintenance release (Page 47)

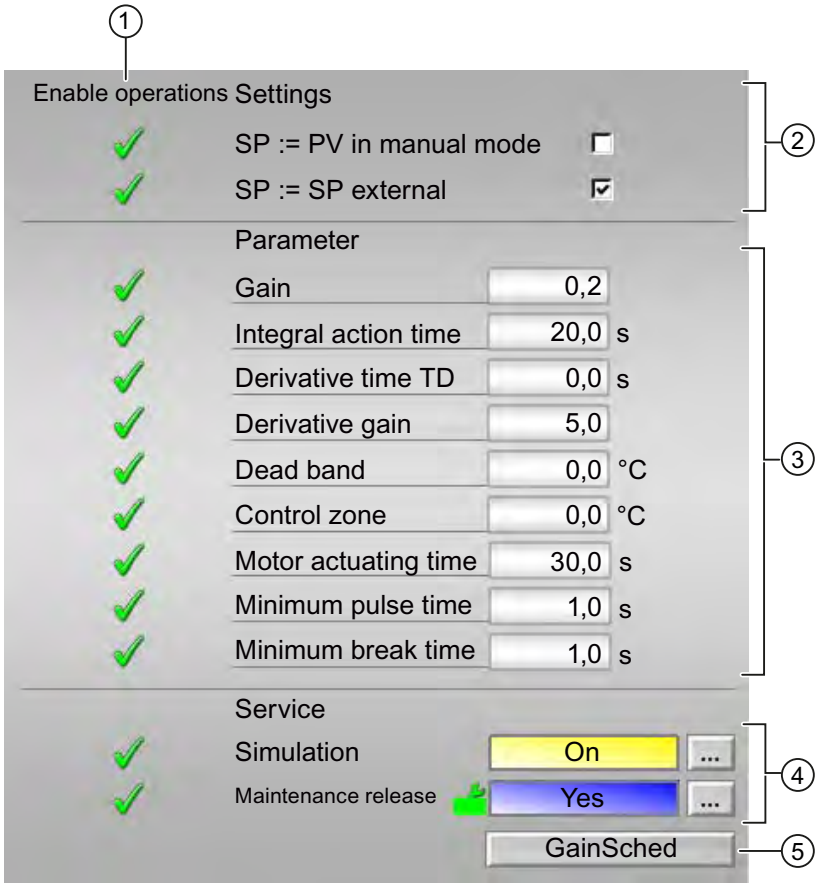
(5) Navigation button for the GainSched block

You can use this navigation button to reach the GainSched block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

1.3.10 Parameter view of FM controllers

Parameter view of FM controllers



(1) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions.

(2) Settings

You can activate the following functions for the controller in this area:

- SP := PV in manual mode: ☑ Bumpless switchover from manual mode to automatic mode
- SP := SP external: ☑ Bumpless switchover of the setpoint for setpoint switchover from external to internal. The internal setpoint is tracked to the external one.

(3) Parameter

In this area, you change parameters and therefore influence the controller. Refer to the Changing values (Page 129) section for more on this.

You can influence the following parameters:

- Gain: Proportional gain
- Integral action time: Integral action time in [s]
- Derivative action time TD: Derivative action time in [s]
- Derivative gain: Gain of the derivative action
- Dead band: Width of dead band
- Control zone: Width of the control zone (for FmTemp block only)
- Motor actuating time: Motor manipulating time [s]
- Minimum pulse time: Minimum pulse duration [s]
- Minimum break time: Minimum break time [s]

(4) Service

You can select the following functions in this area:

- Simulation
- Maintenance release (with display for a maintenance request)

Refer to the Switching operating states and operating modes (Page 127) section for more on this.

You can find information on this area in the section:

- Simulating signals (Page 93)
- Maintenance release (Page 47)

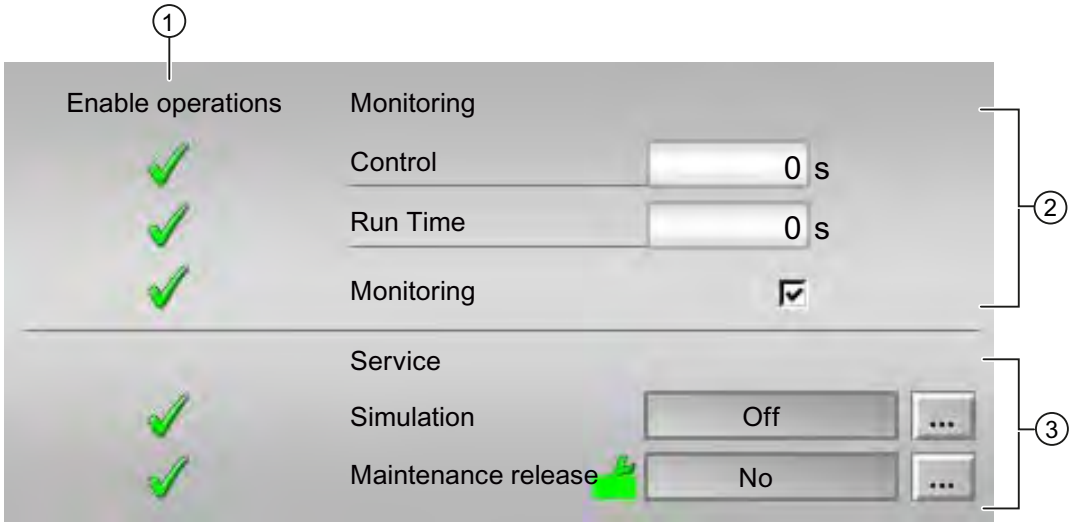
(5) Navigation button for the GainSched block

You can use this navigation button to reach the GainSched block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

1.3.11 Parameter view for motors and valves

Parameter view of motors



(1) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions.

(2) Monitoring

In this area, you change parameters and therefore influence the motor. Refer to the Changing values (Page 129) section for more on this.

You can influence the following parameters:

- Control: Monitoring time during startup and shutdown of the motor (dynamic)
- Runtime: Monitoring time during permanent operation of the motor (static)

Enable monitoring

You can enable monitoring by clicking the check box (☑)

You can find additional information on this in the Monitoring the feedbacks (Page 83) section.

(3) Service

You can select the following functions in this area:

- Simulation
- Maintenance release (with display for a maintenance request)

Refer to the Switching operating states and operating modes (Page 127) section for more on this.

You can find information on this area in the section:

- Simulating signals (Page 93)
- Maintenance release (Page 47)

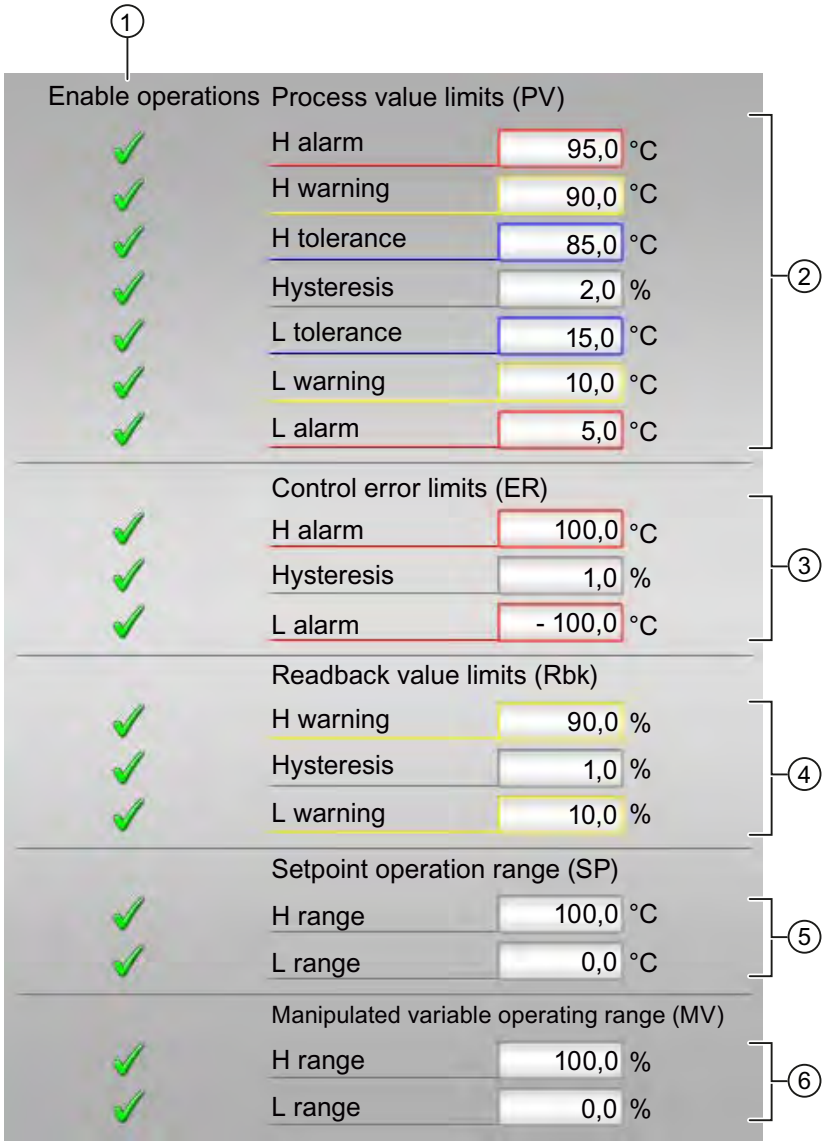
1.3.12 Limit value view of FM controllers

Limit value view of FM controllers

Several values are set in this view by default:

- Process value limits
- Control error limits
- Readback value limits
- Setpoint operation range

The toolbars of the faceplate and the block icon indicate when the limit(s) is reached or exceeded.



(1) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions

(2) Process value limits

In this area, you can enter the limits for the process value. Refer to the Changing values (Page 129) section for more on this.

You can change the following limits:

- H alarm: Alarm high
- H warning: Warning high
- H tolerance: Tolerance high
- Hysteresis
- L tolerance: Tolerance low
- L warning: Warning low
- L alarm: Alarm low

(3) Control error limits

In this area, you can enter the limits for the control error. Refer to the Changing values (Page 129) section for more on this.

You can change the following limits:

- H alarm: Alarm high
- Hysteresis
- L alarm: Alarm low

(4) Readback limits (Rbk)

In this area, you can enter the limits for the readback value. Refer to the Changing values (Page 129) section for more on this.

You can change the following limits:

- H warning: Warning high
- Hysteresis
- L warning: Warning low

(5) Setpoint operation range (SP)

In this area, you can enter the limits for the setpoint operation range. Refer to the Changing values (Page 129) section for more on this.

You can change the following limits:

- H range: Range limit high
- L range: Range limit low

(6) Manipulated variable operating range (MV)

In this area, you can enter the limits for the manipulated variable operation range. Refer to the Changing values (Page 129) section for more on this.

You can change the following limits:

- H range: Range limit high
- L range: Range limit low

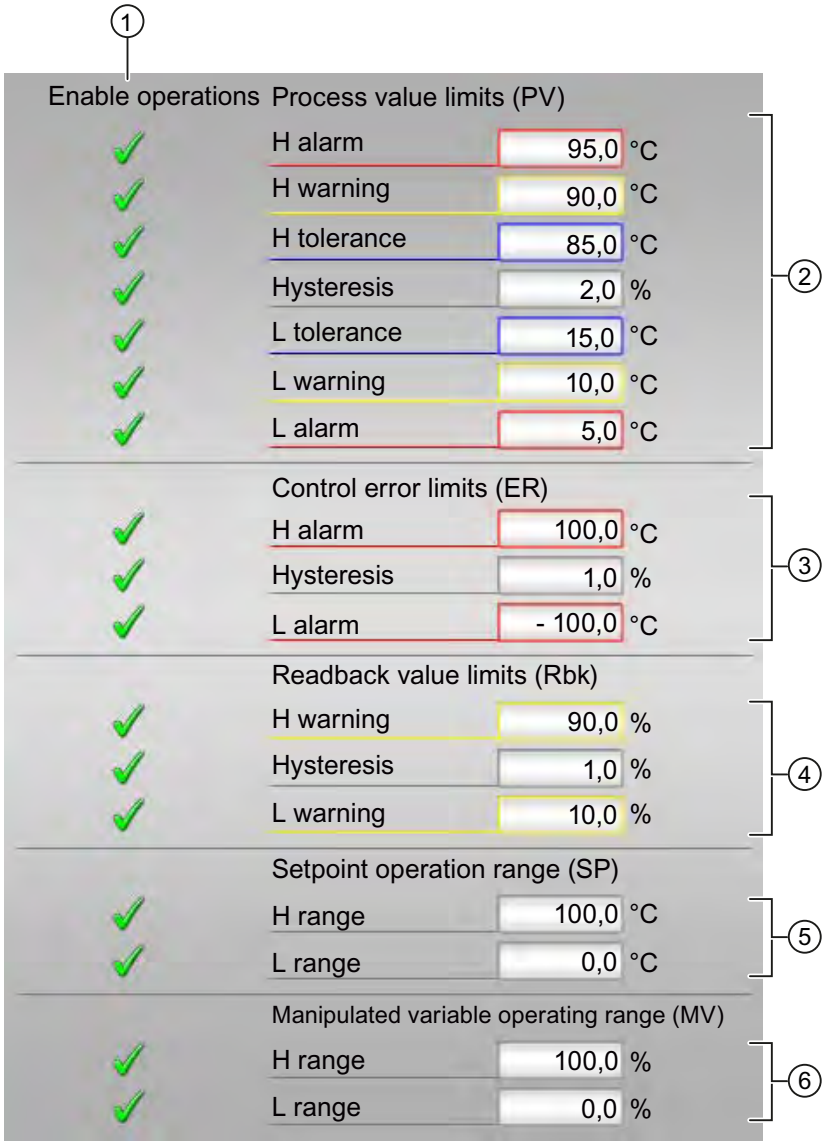
1.3.13 Limit value view of PID controllers

Limit value view of PID controllers

Several values are set in this view by default:

- Process value limits
- Control error limits
- Readback value limits
- Setpoint operation range

The toolbars of the faceplate and the block icon indicate when the limit(s) is reached or exceeded.



(1) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions

(2) Process value limits

In this area, you can enter the limits for the process value. Refer to the Changing values (Page 129) section for more on this.

You can change the following limits:

- H alarm: Alarm high
- H warning: Warning high
- H tolerance: Tolerance high
- Hysteresis
- L tolerance: Tolerance low
- L warning: Warning low
- L alarm: Alarm low

(3) Control error limits

In this area, you can enter the limits for the control error. Refer to the Changing values (Page 129) section for more on this.

You can change the following limits:

- H alarm: Alarm high
- Hysteresis
- L alarm: Alarm low

(4) Readback limits (MV)

In this area, you can enter the limits for the readback value. Refer to the Changing values (Page 129) section for more on this.

You can change the following limits:

- H warning: Warning high
- Hysteresis
- L warning: Warning low

(5) Setpoint operation range (SP)

In this area, you can enter the limits for the setpoint operation range. Refer to the Changing values (Page 129) section for more on this.

You can change the following limits:

- H range: Range limit high
- L range: Range limit low

(6) Manipulated variable operating range (MV)

In this area, you can enter the limits for the manipulated variable operation range. Refer to the Changing values (Page 129) section for more on this.

You can change the following limits:

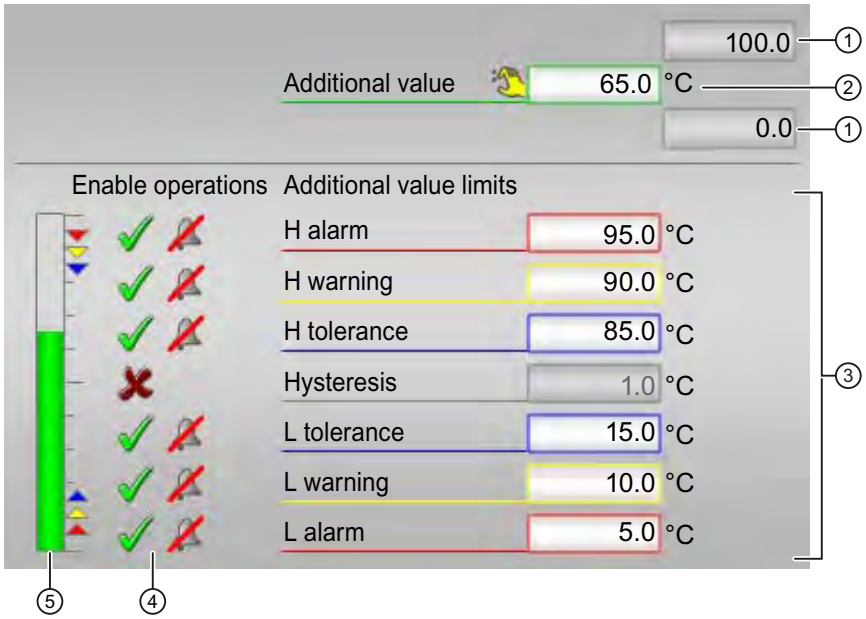
- H range: Range limit high
- L range: Range limit low

1.3.14 Limit value view of motors

Limit value view of motors

The limit value view for motors is only available when an AV block has been interconnected to the motor.

The toolbars of the faceplate and the block icon indicate when the limit(s) is reached or exceeded.



(1) High and low scale range for the additional value

These values provide information on the display range for the bar graph of the additional value. The scale range is defined in the engineering system.

(2) Display of the additional value including signal status

This area shows the current additional value with the corresponding signal status.

(3) Bar graph for the additional value

In this area, you can enter the limits for the additional value. Refer to the Changing values (Page 129) section for more on this.

You can change the following limits:

- H alarm: Alarm high
- H warning: Warning high
- H tolerance: Tolerance high
- Hysteresis
- L tolerance: Tolerance low
- L warning: Warning low
- L alarm: Alarm low

(4) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions

(5) Bar graph for the additional value

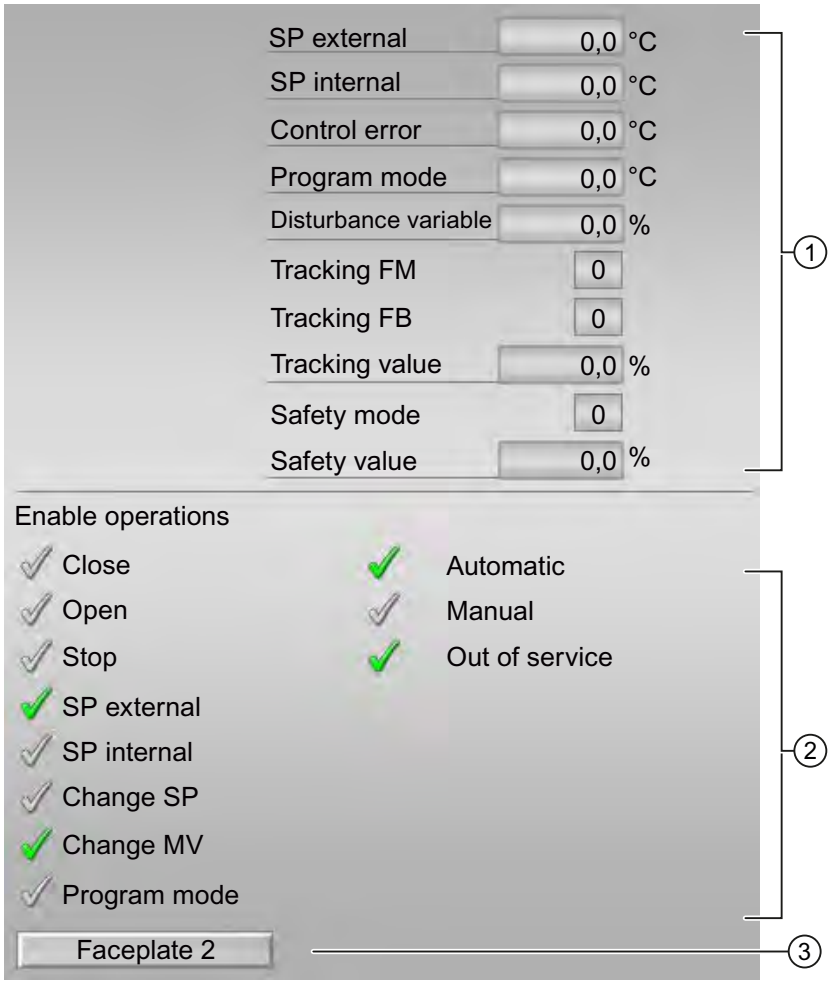
This area show you the current additional value in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

The colored triangles indicate the specified limits (3) for the additional value.

1.3.15 Preview of FM controllers

Preview of FM controllers

The preview shows you the parameters that you, as an OS operator, can control in the entire block. You cannot control anything in this view, however.



(1) Preview area

This area shows you a preview for the following values:

- SP external: currently applicable external setpoint
- SP internal: currently applicable internal setpoint
- Control error: Current control error
- Program mode: Specified value for program mode
- Disturbance: additive value for feedforward control
- Tracking FM: track a manipulated variable in the FM module (value is 1)
- Tracking FB: track manipulated variable at the driver block (value is 1)
- Tracking value: effective manipulated variable for "Track manipulated variable at driver block"
- Safety mode: safety mode in the FM module (value is 1)
- Safety value: effective manipulated variable for "Safety mode"

(2) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions

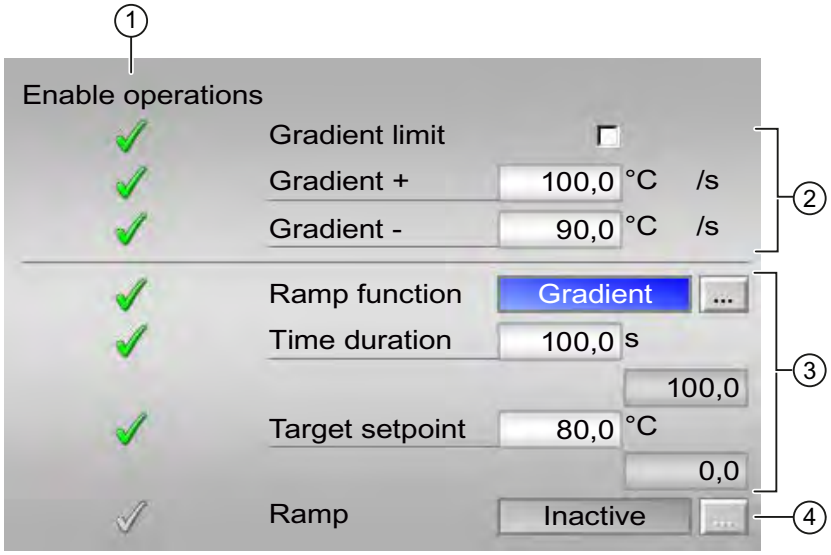
(3) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

See also the Calling further faceplates (Page 43) section for more on this.

1.3.16 Ramp view

Ramp view



(1) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions

(2) Gradient limit on

Use this check box to enable gradient limit of the setpoint. The gradient limit can be set separately for positive or negative setpoint changes (Gradient + or Gradient -). Refer to the Changing values (Page 129) section for more on this.

(3) Ramp function

In this area, you can set the type of ramp function for the setpoint.

You can set the following types of ramp functions:

- Time duration
- Target setpoint

You can set the time duration and the target setpoint. Refer to the Changing values (Page 129) section for more on this.

(4) Ramp on

You can use this control to enable or disable the configured function in the ramp function for the setpoint change.

You can only enable this when the setpoint specification is set to "Internal" in the standard view of the block. The enable is only valid for one setpoint change and is subsequently disabled again.

1.3.17 Message view

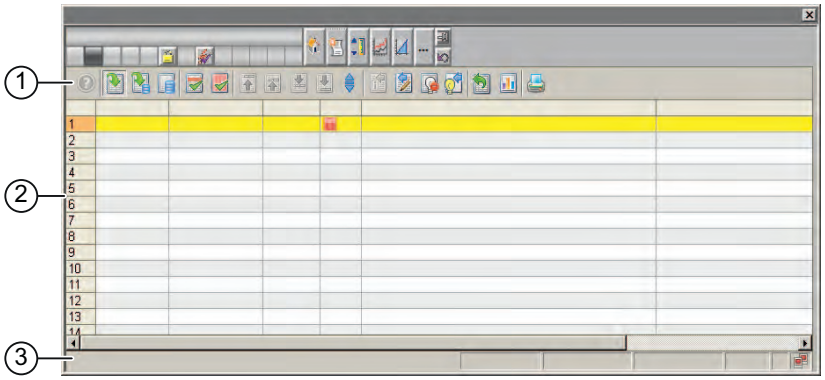
Alarm view

The alarm view is based on these areas:

(1) Toolbar

(2) Display area for alarms

(3) Status bar

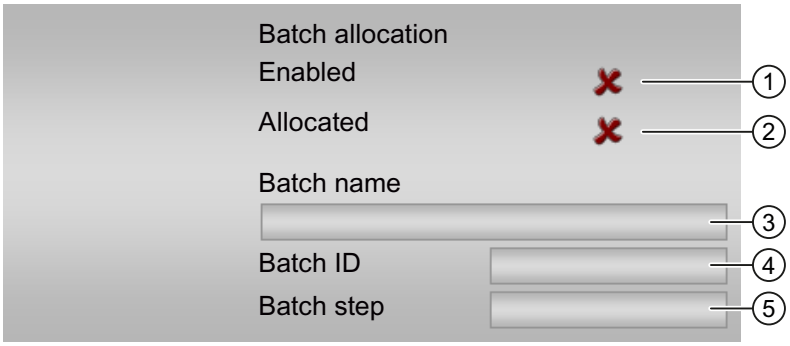


For additional information about the alarm view, refer to the *WinCC Information System* Online Help.

1.3.18 Batch view

Batch view

Batch view appears as follows:



(1) Enabled

This area show you if the block is enabled for operation via SIMATIC BATCH (`BatchEn = 1`).

(2) Allocated

This area shows if the block is currently in use by SIMATIC BATCH (Occupied = 1).

(3) Batch name

This area shows the name of the batch that is currently running (`Batchname`).

(4) Batch ID

This area shows the identification number of the batch that is currently running (`BatchID`).

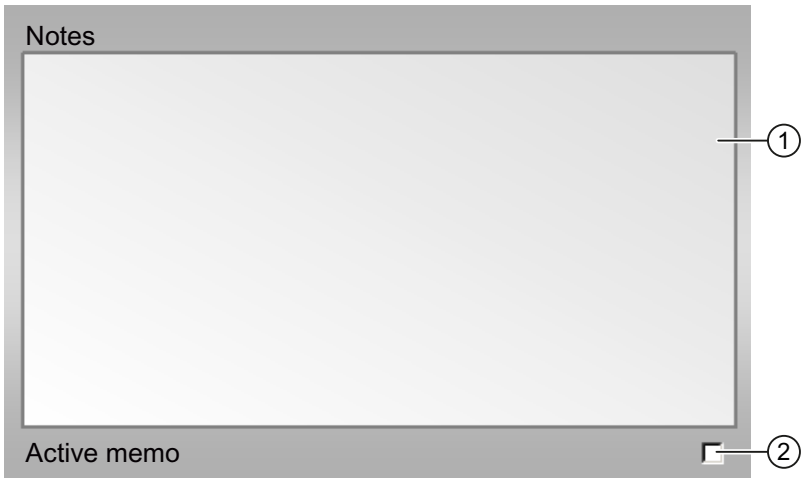
(5) Batch step

This area shows the step number of the batch that is currently running (`StepNo`).

1.3.19 Memo view

Memo view

You can leave temporary messages for other OS operators in this view. Messages are entered in the text box, and saved and activated by selecting the check box in the lower right corner of the faceplate.



(1) Text box for notes

(2) Check box for activating the note

The next time the faceplate is opened or there is a process picture change, you can see in the status bar of the block icon and the faceplate that there is a new message for you.

Clearing the check box deletes the indicators in the status bars.

The message is not deleted automatically.

Note

The content of the memo view is deleted when you perform a full compilation and download of the OS.

1.3.20 Trend view

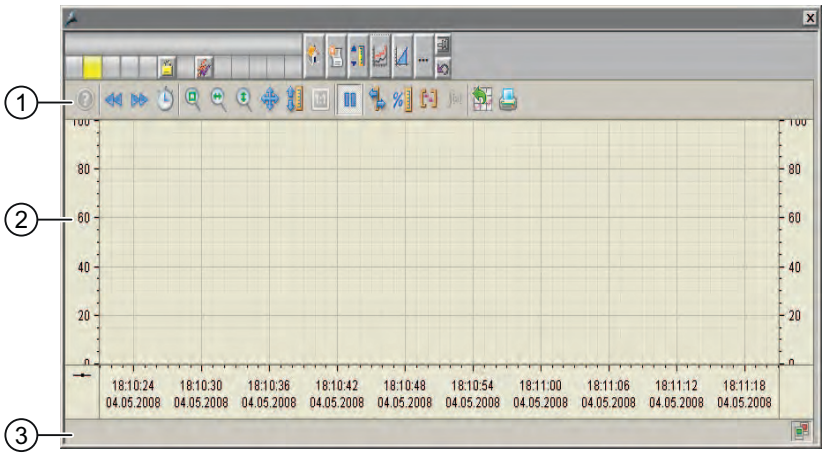
Trend view

The trend view is based on these areas:

(1) Toolbar

(2) Display area for trends

(3) Status bar



For additional information about the trend view, refer to the *WinCC Information System* Online Help.

You can switch between the online data and archive data using the first user button (number 1) in the toolbar. The status bar shows if the trend view is working with online data or archive data.

Special considerations for controllers

You can select two different representations for the display area:

1. Detailed display (default setting):

Display area consisting of three coordinate systems:

- Setpoint, actual value trend;
- Manipulated variable, control performance index trend;
- Binary trend via automatic/manual, manipulated variable at high or low limit

Open the scatterplot diagram with the second user button (number 2) in the toolbar. It shows a coordinate system with the process value on the value axis and the manipulated values or position feedback on the X axis. A new value pair is entered into the coordinate system with each cycle.

If you want to use the detail display, you need to enter the following in the block icon under Trends in the WinCC Graphics Designer:

TrendPictureName = @pg_apl_trendPID_Statistic.pdl

2. Simple display:

Display area consisting of two coordinate systems:

- Setpoint, actual value trend;
- Manipulated variable, control performance index trend;

If you want to use the detail display, you need to enter the following in the block icon under Trends in the WinCC Graphics Designer:

TrendPictureName =@pg_apl_trendPID.pdl

Notes on step controllers with position feedback:

If you use a step controller with position feedback as the controller type, you need to enter the following in the block icon under Trends in the WinCC Graphics Designer:

TrendConfiguration5 = *.MV#Value;...

TrendConfiguration6 = .RbkOut#Value;...

The following applies to all other controller types (default setting):

TrendConfiguration5 = .MV#Value;...

TrendConfiguration6 = *.RbkOut#Value;...

1.4 General functions of the block icons

1.4.1 Block icon structure

Block icon structure

There are two types of block icons, those with a display of the instance-specific name and those without:

	Block icon of MotRevL with instance-specific name
	Block icon of PIDConL with instance-specific name
	Block icon of MotRevL without instance-specific name The toolbar only shows the information that is actually available.
	Block icon of PIDConL without instance-specific name The toolbar only shows the information that is actually available.

You can select one of these block icons. See section Configuring the block icons (Page 180) for more information.

A block icon has several display areas:

- Instance-specific name
- Icon for the block
- Analog value display
- Status bar for the block status

Instance-specific name

The name of the associated block is shown in the instance-specific name, for example for the PIDConL block:

You can change this name in the object properties of the instance block.

There are block icons with or without display of the instance-specific name. Refer to the individual block descriptions to learn about them.

You can reach the visible display for blocks without display of the instance-specific name in two different ways:

- **Displaying individual instance-specific names**: Left-click on the block icon while pressing the Shift key: The name remains visible as long as the process picture is displayed.
- **Displaying all instance-specific names at once**: All instance-specific names can be made visible in a process picture at once by clicking a button. To do this, copy this button into the process picture of the chart from the @PCS7TypicalsAPL.PDL or insert this button in the "Key area" of WinCC. If you insert it into the key area, read the manual section "PCS 7 OS Process Control" > "Layout of the User Interface".

  The instance-specific names are hidden once more by clicking the button again.

Icon for the block

Block icons for technological blocks have their own operable icons (for example, for the MotRevL block) and represent a status display of the block.

This icon can be positioned at various locations, 0°, 90°, 180° and 270°. Refer to the Configuring the block icons (Page 180) section for more on this. You can change the operating mode of the block by right-clicking on the status display. Refer to the Operation via the block icon (Page 181) section for more on this.

Analog value display

For block icons with analog value displays (for example, for the PIDConL block), there are four basic settings that differ depending on the analog value display and the display of the associated unit of measure.

Refer to the Configuring the block icons (Page 180) section for more on this.

These analog values can be controlled according to the Operator permissions (Page 45). See also the Operation via the block icon (Page 181) section for more on this.

Status bar for the block status

The status bar of the block icon provides an overview of the overall status of the block (see figure below).

The arrangement of icons in the following tables is prioritized from high to low.

The following elements can be displayed:

Alarms, warnings, tolerances, and messages

Refer to the Monitoring functions at the Advanced Process Library (Page 70) section for more on this.

Icon	Meaning
	No messages are output.
	A fault has occurred.
	An alarm is triggered.
	An alarm for the high alarm limit is triggered.
	An alarm for the low alarm limit is triggered.
	An warning is triggered.
	An warning for the low alarm limit is triggered.
	An warning for the high alarm limit is triggered.
	A tolerance violation has occurred.
	An tolerance violation for the high tolerance limit has been triggered.
	An tolerance violation for the low tolerance limit has been triggered.
	The block has an operator prompt.
	There is a process status determination.

Operating modes

Refer to the Overview of the modes (Page 25) section for more on this.

Icon	Meaning
	The block is in automatic mode.
	The block is in the "On" operating mode.
	The block is in manual mode.
	A program is running.
	The block is in local mode.
	The block is in the "Out-of-service" operating mode. In this operating mode, no other icons are displayed and no values are shown in the analog value display except for: - Display for an active message in the memo view - Display for the maintenance enable

Internal or external setpoint

Refer to the section Setpoint input - internal and external (Page 96).

Icon	Meaning
	Internal setpoint specification
	External setpoint specification

Signal status

Refer to the Forming and outputting signal status for blocks (Page 88) section for more on this.

Icon	Meaning
	The block is in simulation.
	Signal status is "Bad, device related".
	Signal status is "Bad, process related".
	Signal status is "Uncertain, device related".
	Signal status is "Uncertain, process related".
	A maintenance request is pending.
	Block is released for maintenance

Tracking and forcing of values and bypasses

See sections Forcing operating states (Page 115) , Interlocks (Page 100) and Manual and Automatic mode for control blocks (Page 29).

Icon	Meaning
	At least one value has been forced
	Value is tracked
	There is a bypass of an interlock or a bypass condition of an upstream Intlock FB.

Interlocks

Refer to the Interlocks (Page 100) section for more on this.

Icon	Meaning
	Block is not interlocked.
	Block is interlocked.
	Bypass protection

Memo display

Icon	Meaning
	A message is available in the memo view.

See also

Trip function (Page 102)

Forming the group status for interlock information (Page 105)

1.4.2 Configuring the block icons

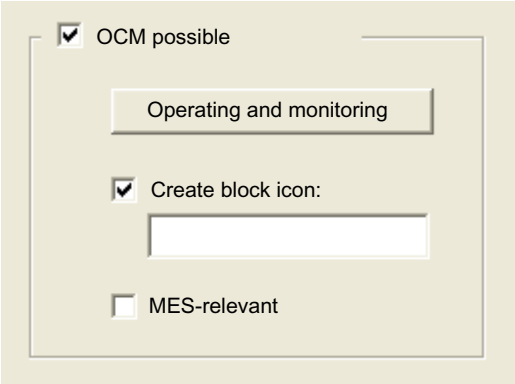
Configuring the block icons

There are two ways to configure your block icons:

- Automatically
- Manually

Automatic configuration of block icons

There is a variety of block icons, which you can select for a block. You select a block icon by entering the number of the block icon in the field OCM possible > Create block icon in the object properties of the block instance:



You can find the names (numerical entry) of the respective block icons in the description for the blocks (Operator Control and Monitoring section).

If you do not enter a number for a block icon, the number 1 is always used for the block icon. The template picture for automatic generation of block icons is @PCS7TypicalsAPL.PDL.

Additional information

- Manual *Process Control System PCS 7; Engineering System*

Manual configuration of block icons

You can configure block icons manually by copying the @TemplateAPL.PDL template and inserting it into plant pictures.

The connection to the process tag is established with the "Connect faceplate with process tag" Wizard, see WinCC Information System, section "Making Process Pictures Dynamic" and "Standard Dynamics".

You can find information on exporting/importing and updating these objects in the WinCC Information System, section "Graphic Object Update Wizard". Use the "TemplateControlAPL.cfg" configuration file for these wizards.

1.4.3 Operation via the block icon

Operation via the block icon

The block icon can be used to operate all elements displayed there if a relevant operator control permission (OS_Perm) is available. This operator control permission can be configured in the engineering system (ES).

The operation is performed by right-clicking on the element involved. The operable elements in the block icon include:

- Switching the operating mode
- Internal and external setpoint specification
- Changing the process value, setpoint and manipulated variable
- Changing the operating state

The operation is then performed in the same way as in the faceplate. Refer to the section Switching operating states and operating modes (Page 127) as well as Changing values (Page 129).

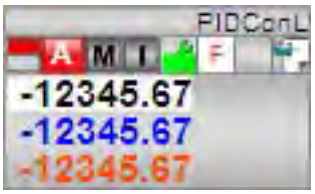
1.4.4 Block icons for PID and FM controller

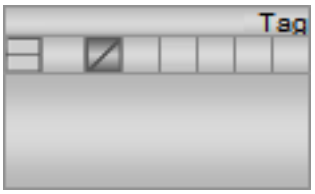
Block icons for PID and FM controller

A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as the control system fault CSF
- Operating modes
- Internal and external setpoint specification
- Signal status, maintenance release
- Memo display
- Process value (black, with and without decimal places)

- Setpoint (blue, with and without decimal places)
- Feedback value (red, with and without decimal places)

Icons	Selection of the block icon in CFC	Special features
	1	Block icon in the full display
	2	
	3	
	4	
	5	
	6	
	7	
	8	

Icons	Selection of the block icon in CFC	Special features
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181)

1.4.5 Block icon for interlock blocks

Properties of the block icon for interlock blocks Intlk02, Intlk04, Intlk08, Intlk16

A variety of block icons are available with the following functions:

- Signal status
- Memo display
- Output signal

Icons	Selection of the block icon in CFC	Block icon for
	1	Intlk02
	1	Intlk04
	1	Intlk08
	1	Intlk16
	2	Intlk02
	2	Intlk04
	2	Intlk08
	2	Intlk16
	3	Intlck02
	3	Intlck04
	3	Intlck08
	3	Intlck16

Block icons (mini) without display of instance-specific names

These icons only show the output signal. They take the form of a small rectangle.

Display of the output signal

The display can show the following states for the output signal (priority from high to low):

- Gray: No inputs interconnected at the interlock block, the block is not used
- Blue: The output signal is 1, at least one input signal is bypassed. There is no gray state.
- Yellow: The signal status is 16#60; the output signal is simulated. There are no gray and blue states.
- Red: The output signal is 0; there is an interlock. There are no gray, blue, and yellow states.
- Green: The output signal is 1; the block is in the good state. There are no gray, blue, yellow, and red states.

Addtional information on the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181)

1.5 Selectable block response

1.5.1 Functions that can be set via the Feature I/O

Configurable functions with the Feature I/O

Some blocks have an input called `Feature`. This input can be used to influence the way in which the block works.

The `Feature` bits are assigned in the following order:

Bit number	Meaning
0	Setting the startup response (Page 187)
1	Reaction to the out of service mode (Page 199)
2	Resetting the commands for changing the mode (Page 193)
3	Enabling resetting of commands for the control settings (Page 193)
4	Setting switch or button mode (Page 195)
5	Specifying the dosing type (Page 189)
6	Resetting the dosing quantity when dosing starts (Page 194)
7	Enabling direct changeover between forward and reverse (Page 189)
8	Unit for the rate of change (Page 190)
9	Resetting via input signals in the event of interlocks or errors (Page 194)
10	Exiting local mode (Page 199)
11	Stopping dosing at a flow alarm (Page 187)
12	Automatic dribble correction for underdosing in automatic mode (Page 188)
13	Creep rate is always detected in the dosing quantity (Page 195)
14	Enabling rapid stop via faceplate (Page 192)
15	Safety manipulated variable with "out of service" operating mode in effect (Page 196)
16	Safety value of the manipulated variable effective at startup (Page 196)
17	Enabling bumpless changeover to automatic mode for valves, motors, and dosers (Page 197)
18	Disabling bumpless changeover to automatic mode for controllers (Page 197)
19	Enabling program mode (Page 192)
20	Activating bumpless changes to proportional gain (Page 196)
21	Switching operator controls for external setpoint to visible (Page 189)
22	Update acknowledgment and error status of the message call (Page 193)
23 - 26	Not allocated
27	Selecting values associated with messages (Page 192)
28	Output invalid raw value (Page 198)
29	Issuing substitute value if raw value is invalid (Page 190)
30	Issuing last valid value if raw value is invalid (Page 191)
31	Activating recording of the first signal (Page 191)

1.5.2 Stopping dosing at a flow alarm

Stopping dosing at a flow alarm

You can use this feature bit to enable stopping dosing at a flow alarm.

The default setting is 0.

Bit = 0: Disabled, dosing is not stopped when a flow alarm occurs

Bit = 1: Enabled, dosing is stopped when a flow alarm occurs

1.5.3 Setting the startup response

Setting the startup response

With this `Feature` bit, you set the startup response of the function blocks, for example, for:

- Motors, valves and controllers
- Channel blocks
- Monitoring blocks, e.g. `MonAnL` and `MonDiL`.
- Mathematical, logical analog blocks and the OpAnL block
- The OpAnL block
- Counter blocks
- The Average block

The default setting is 0.

Defining startup characteristics for motors, valves and controllers

Bit = 0: Starting the block in manual mode and in safe position. With controllers, the setpoint is set to (`SP_Int`) internally. See also the Safe position for motors, valves and controllers (Page 120) section for more on this.

Bit = 1: Starting the block with the last stored values, in other words in the last operating mode set (manual, automatic or local mode) and at the last valid position.

Defining the startup characteristics for channel blocks

Bit = 0: The channel block uses either the process value `PV_In` or the value of `SimPV_In` as the startup value, depending on the setting of the input parameter `SimOn` (`PV_In = PV_Out` or `SimPV_In = PV_Out`).

Bit = 1: The channel block uses the value `StartVal` as the startup value (`StartVal = PV_Out`).

Defining the startup characteristics for monitoring blocks

Bit = 0: The most recently stored value is reset on startup.

Bit = 1: The most recently used value at the output parameter `Out` is output on startup.

Define startup characteristics for mathematical and logical analog blocks

Bit = 0: The `Out` output parameter is reset to 0 on startup.

Bit = 1: The most recently saved value is output at the `Out` output parameter on startup.

Defining startup characteristics for the OpAnL block

Bit = 0: The internal setpoint is used for startup.

Bit = 1: The most recently saved value is output at the `Out` output parameter on startup.

Defining the startup characteristics for counter blocks

Bit = 0: On startup, the counter is stopped and reset to the value specified in the `PresetVal` input parameter (for the CountScL block) or `PresetTime` (for the CountOh block).

Bit = 1: On startup, counting continues with the most recently stored values.

Defining startup characteristics for the Average block

Bit = 0: At startup, averaging begins with the value that is currently at the input parameter `(Out = In, NumCycles = 1)`.

Bit = 1: When starting-up, the last `Out` and `NumCycles` values saved are used as the last value for averaging `(Out ≠ In)`.

1.5.4 Automatic dribble correction for underdosing in automatic mode

Automatic dribble correction for underdosing in automatic mode

Use this `Feature` bit to enable automatic dribble correction for underdosing in automatic mode.

The default setting is 0.

Bit = 0: Disabled, no automatic dribble correction is started for underdosing in automatic mode.

Bit = 1: Enabled, automatic dribble correction is started for underdosing in automatic mode.

1.5.5 Switching operator controls for external setpoint to visible

Switching operator controls for external setpoint to visible

Use this `Feature` bit to switch all operator controls for the external setpoint to visible in the faceplates for the OS operator. This is always required when the external setpoint should actually be used, for example, for slave controllers in cascade and ration controlling.

The default setting is 0, which keeps the faceplate as clear as possible for simple applications.

Bit = 0: The operator controls for the external setpoint are **not visible** in the faceplate.

Bit = 1: The operator controls for the external setpoint are **visible** in the faceplate.

1.5.6 Enabling direct changeover between forward and reverse

Direct changeover between forward and reverse

With this `Feature` bit, you can enable direct reversal of the direction of motors.

The default setting is 0.

Bit = 0: Direct reversal of the direction is disabled.

You can only change the direction of the motor by first stopping and starting the motor again in the required direction. The motor can only be started again after the time set in the `IdleTime` parameter has elapsed.

Bit = 1: Direct changeover is enabled.

You can reverse the motor direction directly. The motor block reverses the direction automatically. The motor is stopped and is started in the other direction when the time set in the `IdleTime` parameter has elapsed.

1.5.7 Specifying the dosing type

Specifying the dosing type

You can specify the dosing type to be used for the block using this `Feature` bit.

The default setting is 0.

Bit = 0: Flow

Bit = 1: Scales

1.5.8 Unit for the rate of change

Specifying the unit for the rate of change

You can use this `Feature` bit to specify the unit for the rate of change:

The default setting is 0.

Bit = 0: unit for the rate of change in the unit of measurement to or from the field device

Bit = 1: unit for the rate of change as a percentage to or from the field device

1.5.9 Issuing substitute value if raw value is invalid

Output substitute value if raw value is invalid

Use this `Feature` bit to activate output of the substitute value for channel driver blocks (input parameter `SubsPV_In`) if there is no valid raw value.

The default setting is 0.

Bit = 0: The substitute value is not output.

Bit = 1: The substitute value is output. The signal status of the output value is set to "Local functional check / simulation".

If there is no valid raw value the output parameter `Bad = 1` is set automatically.

Prioritizing the `Feature` bits for driver blocks:

You need to configure three `Feature` bits for the response to an invalid raw value for the driver blocks.

If more than one of these `Feature` bits are set (=1), the following priority applies:

- Output invalid raw value (Page 198) (`Feature` bit 28 = highest priority)
- Output substitute value if raw value is invalid (`Feature` bit 29)
- Issuing last valid value if raw value is invalid (Page 191) (`Feature` bit 30 = lowest priority)

If none of the `Feature` bits 28, 29 and 30 are set (=0), the block sets `Feature` bit 28 (=1).

1.5.10 Activating recording of the first signal

Activating recording of the first signal

Use this `Feature` bit to activate recording of the first signal with interlock blocks. Please also refer to the section Recording the first signal for interlock blocks (Page 87).

Default setting is 0

Bit = 0: Recording of the first signal is deactivated.

Bit = 1: Recording of the first signal is activated.

1.5.11 Issuing last valid value if raw value is invalid

Issuing last valid value if raw value is invalid

Use this `Feature` bit to activate output of the last valid value for channel driver blocks if there is no valid raw value.

The default setting is 0.

Bit = 0: If there is no valid raw value the last valid value is not output.

Bit = 1: If there is no valid raw value the last valid value is output. The signal status of the output value is set to Local "functional check / simulation".

If there is no valid raw value the output parameter `Bad = 1` is set automatically.

Prioritizing the `Feature` bits for driver blocks:

You need to configure three `Feature` bits for the response to an invalid raw value for the driver blocks.

If more than one of these `Feature` bits are set (=1), the following priority applies:

- Output invalid raw value (Page 198) (`Feature` bit 28 = highest priority)
- Issuing substitute value if raw value is invalid (Page 190)(`Feature` bit 29)
- Output the last valid value if raw value is invalid (`Feature` bit 30, lowest priority)

If none of the `Feature` bits 28, 29 and 30 are set (=0), the block sets `Feature` bit 28 (=1).

1.5.12 Selecting values associated with messages

Selecting values associated with messages

Use this `Feature` bit to select which values associated with messages are to be output.

The default setting is 0.

Bit = 0: The signal status of the binary input is output as the value associated with messages.

Bit = 1: The associated analog value is output as the value associated with messages.

1.5.13 Enabling rapid stop via faceplate

Enabling rapid stop via faceplate

You can use the `Feature` bit "Enable rapid stop via faceplate" to specify if the OS operator can use rapid stop for the block via the standard view of the faceplate.

The default setting is 0.

Bit = 0: The "Rapid stop" button is not visible in the faceplate.

Bit = 1: The OS operator can use the button for rapid stop.

1.5.14 Enabling program mode

Enabling program mode

You can use this feature bit to specify whether or not the controller block should be used for program mode.

Default setting is 0.

Bit = 0: The block is not intended for program mode.

Bit = 1: The block can be used for program mode. The operator control elements required for this are then visible in the faceplate.

What is the program mode?

Program mode provides primary controller functions (external Advanced Control software package), which run on an external PC as an OPC client, the option of using the control from the controller function block and specifying the setpoint or manipulated variable from a remote location.

1.5.15 Update acknowledgment and error status of the message call

Update acknowledgment and error status of the message call

You can use the `Feature` bit to determine if the acknowledgment and error status of the message call at the block output should be updated.

The default setting is 0.

- Bit = 0
  The block outputs `MsgErr`, `MsgStat` and `MsgAckn` are set to the default setting and not updated. The block will run faster with this setting.
- Bit = 1
  The block outputs `MsgErr`, `MsgStat` and `MsgAckn` are updated based on the feedback of the lower level message blocks. The lower level message blocks are called every other cycle as long as an acknowledgment is expected or error information is pending.

1.5.16 Resetting the commands for changing the mode

Resetting the commands for changing the mode

Using this `Feature` bit, you define how the block handles the incoming control commands `SP_IntLi`, `SP_ExtLi` (for controllers) as well as `AutModLi` and `ManModLi` .

The default setting is 0.

Bit = 0: The control commands are not reset by the block. If there are two pending control commands for changing mode, the mode is not changed. In this case, the text "Invalid signal" is displayed in the faceplate.

Bit = 1: The control commands are reset by the block. This, for example, ensures that if a control command is sent from the SFC, the command is reset automatically after a step is exited.

1.5.17 Enabling resetting of commands for the control settings

Enabling resetting of commands for the control settings

With this `Feature` bit, you select how the block handles commands for the control settings (for example motor on) via the interconnected input parameters.

The default setting is 0.

Bit = 0: The control commands are not reset by the block. If there are two commands relating to the control settings at the same time, the status of the control settings is retained. In this case, the "Invalid signal" message is displayed in the standard view of the faceplate.

Bit = 1: The control commands are reset by the block. This, for example, ensures that if a control command is sent from the SFC, the command is reset automatically after a step is exited.

1.5.18 Resetting the dosing quantity when dosing starts

Resetting the dosing quantity when dosing starts

Use this feature bit to enable resetting the dosing quantity when dosing starts.

The default setting is 0.

Bit = 0: Disabled, the dosing quantity is not reset when dosing starts

Bit = 1: Enabled, the dosing quantity is reset when dosing starts

1.5.19 Resetting via input signals in the event of interlocks or errors

Enabling block reset via input signals in the event of interlocks or errors

With this `Feature` bit, you define how automatic control is to be re-enabled after an active interlock.

The default setting is 0.

Bit = 0: After an interlock or an error, the system can only be started again using a reset command. Reset is initiated either by operator input in the faceplate or via the interconnectable input parameter (`RstLi = 1`) in the block. Following this, the currently pending command is effective in automatic mode, whereas in manual mode, you must initiate the next control action yourself.

Bit = 1: It is also possible to reset with a 0-1 edge change in the control signal in automatic mode. In manual mode, you will need to initiate the next control action manually.

1.5.20 Setting switch or button mode

Setting switch or button mode (input signal as pulse signal or as static signal)

Use this `Feature` bit to define whether the command for controlling the module, for example:

- starting and stopping a motor
- Opening and closing a valve
- switching modes (parameters `AutModLi` and `ManModLi`)
- Setpoint input internal and external (parameters`SP_ExtLi` and `SP_IntLi`)

is given in the form of a pulse (pushbutton operation) or a static signal (switch operation).

You can find the commands for controlling the block in the relevant section on block operating modes. They are always the parameters that are used for the automatic operation of a block.

Bit = 0: Button mode: The signals from the logic applied to the input parameters take the form of a pulse signal. (evaluation only of 0-1 edge)

Example with a motorMotRevL: In this case, use the interconnectable input parameters `FwdAut`, `RevAut` and `StopAut` or `AutModLi` and `ManModLi`.

Bit = 1: Switch mode: The signal from the upstream logic that is applied to an input parameter takes the form of a static signal (evaluation of the signal state).

Example with a motorMotRevL: In this case, use the interconnectable input parameters `FwdAut` (`FwdAut = 1`: the motor runs forward, `FwdAut = 0`: the motor is off or `RevAut = 1`: the motor runs in reverse, `RevAut = 0`: the motor is off) and `AutModLi`.

1.5.21 Creep rate is always detected in the dosing quantity

Enable: Creep rate is always detected in the dosing quantity

Use the `Feature` bit to specify the response for detecting the creep rate in the dosing quantity.

The default setting is 0.

Bit = 0: Disabled, the creep rate is only detected in the dosing quantity over the limit `CR_AH_Lim` (high alarm for creep rate). With `CR_AH_En = 0`, the creep rate has no effect on the dosing quantity calculation.

Bit = 1: Enabled, the creep rate is always detected in the dosing quantity.

Note

Creep rate is the flow in the states "End", "Off", and "Pause".

1.5.22 Safety value of the manipulated variable effective at startup

Safety value of the manipulated variable effective at startup

With the safety position value bit Feature bit, you decide whether or not a controller changes to the safety position during startup.

Default setting is 0

Bit = 0: The controller does not change to the safety position during startup

Bit = 1: The controller changes to the safety position during startup

Please also refer to the section Safe position for motors, valves and controllers (Page 120).

1.5.23 Safety manipulated variable with "out of service" operating mode in effect

Safety manipulated variable with "out of service" operating mode in effect

With the feature bit "Safety manipulated variable with "out of service" operating mode in effect", you specify whether or not a controller travels to the safe position during startup.

Default position is "0".

Bit = 0: The controller does not travel to the safe position with the transition to the "out of service" operating mode.

Bit = 1: The controller travels to the safe position with the transition to the "out of service" operating mode.

Refer to the Safe position for motors, valves and controllers (Page 120) section for more information.

1.5.24 Activating bumpless changes to proportional gain

Activating bumpless changes to proportional gain

Use this `Feature` bit to activate bumpless changes to proportional gain `Gain` in automatic mode.

The default setting is 0.

Bit = 0: The bumpless changeover is deactivated.

Bit = 1: The bumpless changeover is activated.

1.5.25 Disabling bumpless changeover to automatic mode for controllers

Changeover with or without P step change when the internal setpoint is not in tracking mode

Use this `Feature` bit to specify if a changeover should occur with or without a P step change when the internal setpoint (`SP_TrkPv = 0`) does not track the process value.

The default setting is 0.

Bit = 0: Changeover without P step change (bumpless)

Bit = 1: Changeover with P step change (not bumpless)

For more detailed information, refer to the description of Manual and Automatic mode for control blocks (Page 29).

1.5.26 Enabling bumpless changeover to automatic mode for valves, motors, and dosers

Bumpless changeover

You can use this `Feature` bit to enable the bumpless changeover from local/manual mode to automatic mode.

Default setting is 0

Bit = 0: Bumpless changeover is disabled. You can switch from local/manual mode to automatic mode at any time.

Bit = 1: Bumpless changeover from local/manual mode to automatic mode is enabled. A changeover from local/manual mode to automatic mode is only possible if the control settings of the local/manual mode and automatic modes match.

Please also refer to the section Manual and automatic mode for motors, valves and dosers (Page 32).

1.5.27 Output invalid raw value

Output invalid raw value

Use this `Feature` bit to activate output of the invalid raw value for channel driver blocks.

The default setting is **1**(!).

Bit = 0: The invalid raw value is not output. Either the substitute value (`Feature` bit Issuing substitute value if raw value is invalid (Page 190)) or the last valid value (`Feature` bit Issuing last valid value if raw value is invalid (Page 191)) is output.

Bit = 1: The invalid raw value is output. The signal status of the output value is set to "Bad, device related" or "Bad, process related".

If there is no valid raw value the output parameter `Bad = 1` is set automatically.

Prioritizing the `Feature` bits for driver blocks:

You need to configure three `Feature` bits for the response to an invalid raw value for the driver blocks.

If more than one of these `Feature` bits are set (=1), the following priority applies:

- Output invalid raw value (`Feature` bit 28 = highest priority)
- Issuing substitute value if raw value is invalid (Page 190)(`Feature` bit 29)
- Issuing last valid value if raw value is invalid (Page 191) (`Feature` bit 30 = lowest priority)

If none of the `Feature` bits 28, 29 and 30 are set (=0), the block sets `Feature` bit 28 (=1).

1.5.28 Reaction to the out of service mode

Reaction to the out of service mode

With this Feature bit, you decide the reaction of the technological block depending on the interconnectable input parameter `OosLi = 1`.

The default setting is 0.

- **Bit = 0**: The symbol for the "In progress" status (see above) appears in the block icon and in the faceplate of the assigned technological block. A 0-1 edge transition at the input parameter OosLi has no further influence on the reaction of the technological block; the previous status is retained. No switch to the "Out of service" mode is performed.
- **Bit = 1**: The mode changes to "out of service" assuming that the block is on or manual mode. If this is not the case, the mode does not change. The symbol for the "In progress" (see below) status also appears in the block icon and in the faceplate of the assigned technological block regardless of the mode change. No message is output to indicate whether or not the mode change took place.

The status display for "In progress" appears as follows:

A 1-0 edge transition at the input parameter `OosLi` has no influence on the reaction of the technological block, the previous status is retained.

See also the Maintenance release (Page 47) section for more on this.

1.5.29 Exiting local mode

Reaction to exiting local mode

Use this `Feature` bit to define how the "Local mode" is to be exited with `LocalSetting = 1` or `LocalSetting = 2` and if the mode is not specified by `AutModLi` or `ManModLi`.

Default setting is 0

Bit = 0: Exiting local mode in manual mode (bumpless because the control signals are continuously adjusted).

Bit = 1: When local mode is exited, the mode changes back to the last mode that was active prior to local mode (not bumpless).

For more detailed information, refer to the description of Local mode (Page 36).

Operator control blocks

2

2.1 OpAnL- check and output analog signals

2.1.1 Description of OpAnL

Object name (type + number) and family

Type + number: FB 1865

Family: Operate

Area of application for OpAnL

The block is used for the following applications:

- Checking and transferring analog input values

How it works

The block checks incoming, internal (entered in the faceplate) or external (CFC/SFC) analog signals in terms of their limits at the input SP_Int or SP_Ext and forwards them to the output SP, depending on the setting of the input parameter SP_LiOp.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

Startup characteristics

Use the Feature bit Setting the startup response (Page 187) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at RunUpCyc.

Status word allocation for `Status1` parameter

The description for each parameter can be found in the OpAnL I/Os (Page 209) section.

Status bit	Parameter
0	`Occupied`
1	`BatchEn`
2	Not used
3	`OosAct.Value`
4	`OosLi.Value`
5	Not used
6	`OnAct.Value`
7	Not used
8	`SP_ExtAct.Value`
9	`SP_LoAct.Value`
10	`SP_HiAct.Value`
11 - 13	Not used
14	`SP_RmpOn`
15	`SP_RmpModTime`
16 - 31	Not used

See also

OpAnL functions (Page 204)

OpAnL messaging (Page 207)

OpAnL block diagram (Page 214)

OpAnL error handling (Page 206)

OpAnL modes (Page 203)

2.1.2 OpAnL modes

OpAnL modes

The block can be operated using the following modes:

- On (Page 27)
- Out of service (Page 27)

"On"

General information on the "On" mode is available in the section On (Page 27).

"Out of service"

You will find general information about the "Out of service" mode in the Out of service (Page 27) section.

See also

OpAnL I/Os (Page 209)

OpAnL messaging (Page 207)

OpAnL error handling (Page 206)

Description of OpAnL (Page 201)

OpAnL functions (Page 204)

OpAnL block diagram (Page 214)

Manual and Automatic mode for control blocks (Page 29)

Local mode (Page 36)

2.1.3 OpAnL functions

Functions of OpAnL

The functions for this block are listed below.

Internal or external setpoint selection

This block provides the standard function Setpoint input - internal and external (Page 96).

Setpoint limitation

Use the SP_HiLim and SP_LoLim input parameters to limit the setpoint to maximum and minimum limits. If a limit is violated, the setpoint is limited to the limits you have set. If the limits are infringed, the output parametersSP_HiAct and SP_LoAct display 1.

Using setpoint ramp

This block provides the standard function Using a setpoint ramp (Page 98).

Gradient limit of the setpoint

This block provides the standard function Ramp limiting of the setpoint (Page 108).

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status ST_Worst for the block is formed from the following parameters:

- SP_Out.ST
- PV_In.ST

The signal status of output parameter SP is always equivalent to the signal status of input parameter SP_Ext or SP_Int, depending on how the setpoint is specified. If the internal setpoint SP_Int is used, the signal status is always output as 16#80.

Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 53).

Configurable reactions using the Feature parameter

You can find an overview of all reactions provided by the Feature parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)
1	Reaction to the out of service mode (Page 199)
22	Update acknowledgment and error status of the message call (Page 193)

Operator control permissions via parameter OS_Perm

The block has the following Operator permissions (Page 45) for the OS_Perm parameter:

Bit	Function
0	Not allocated
1	1 = Operator can switch to On mode
2	Not allocated
3	1 = Operator can switch to out of service mode
4 - 9	Not allocated
10	1 = Operator can switch to internal
11	1 = Operator can switch to external
12	1 = Operator can perform bumpless switchover
13	1 = Operator can change SP_Int
14	1 = Operator can enable SP_RateOn
15	1 = Operator can change SP_UpRaLim
16	1 = Operator can change SP_DnRaLim
17	1 = Operator can activate the setpoint ramp (SP_RmpOn)
18	1 = Operator can switch between specification of the duration (SP_RmpTime) and gradient (SP_DnRaLim, SP_UpRaLim) for calculating the ramp slope
19	1 = Operator can change the time for the setpoint ramp (SP_RmpTime)
20	1 = Operator can change the target setpoint (SP_RmpTarget)
21-31	Not allocated

Specifying the display area for process and setpoint values as well as operations

The block provides the standard function Display and operator input area for process values and setpoints (Page 42).

Opening additional faceplates

The block provides the standard function Calling further faceplates (Page 43).

SIMATIC BATCH functionality

The block provides the standard function SIMATIC BATCH functionality (Page 122).

See also

Description of OpAnL (Page 201)
OpAnL messaging (Page 207)
OpAnL I/Os (Page 209)
OpAnL block diagram (Page 214)
OpAnL error handling (Page 206)
OpAnL modes (Page 203)

2.1.4 OpAnL error handling

OpAnL troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Predefined value when inserting the block, the block is not processed.
0	There is no error.
51	Invalid signal. `SP_LiOp = 1` and `SP_ExtLi = 1` and `SP_IntLi = 1`.

See also

OpAnL block diagram (Page 214)
OpAnL I/Os (Page 209)
OpAnL messaging (Page 207)
OpAnL functions (Page 204)
OpAnL modes (Page 203)
Description of OpAnL (Page 201)

2.1.5 OpAnL messaging

Messaging

The following messages can be generated for this block:

- Process messages

Process messages

Message instance	Message identifier	Message class	Event
MsgEvId	SIG 1	AS process control message - fault	$$BlockComment$$ External message 1
	SIG 2	AS process control message - fault	$$BlockComment$$ External message 2
	SIG 3	AS process control message - fault	$$BlockComment$$ External message 3
	SIG 4	AS process control message - fault	$$BlockComment$$ External message 4
	SIG 5	Reserved	$$BlockComment$$ Reserved
	SIG 6	Reserved	$$BlockComment$$ Reserved
	SIG 7	Reserved	$$BlockComment$$ Reserved
	SIG 8	Reserved	$$BlockComment$$ Reserved

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Associated values for message instance `MsgEvId`

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`
4	`ExtVa104`
5	`ExtVa105`
6	`ExtVa106`
7	`ExtVa107`

The associated values 4 ... 7 are allocated to the parameters `ExtVa104` ... `ExtVa107` and can be used by yourself. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

See also

OpAnL error handling (Page 206)

OpAnL modes (Page 203)

OpAnL block diagram (Page 214)

2.1.6 OpAnL I/Os

OpAnL I/Os

Input parameters

Parameter	Description	Type	Default
BatchEn	1 = Enable allocation	BOOL	0
BatchID	Batch ID	DWORD	16#00000000
BatchName	Batch name	STRING[32]	''
EN	1 = Called block will be processed	BOOL	1
ExtMsg1	Binary input for freely selectable message 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg2	Binary input for freely selectable message 2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg3	Binary input for freely selectable message 3	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg4	Binary input for freely selectable message 4	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtVa104	Associated value 4 for messages (MsgEvID1)	ANY	
ExtVa105	Associated value 5 for messages (MsgEvID1)	ANY	
ExtVa106	Associated value 6 for messages (MsgEvID1)	ANY	
ExtVa107	Associated value 7 for messages (MsgEvID1)	ANY	
Feature	I/O for additional functions (Page 204)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
MsgEvId	Message number (assigned automatically)	DWORD	16#FF
Occupied	1 = Occupied by batch control	BOOL	0
OnOp	1 = "On" mode via operator	BOOL	0

Parameter	Description	Type	Default
OosLi	1 = Out of service, via interconnection or SFC (0-1 edge transition)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OS_Perm	I/O for operator control permissions (Page 204)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
PV_In	Process value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_Unit	Unit of measure for process value	INT	0
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SP_DnRaLim	Limit (low) for ramp of setpoint [SP_Unit/s]	REAL	100.0
SP_Ext	external setpoint - (to interconnection)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_ExtLi	1 = Select internal setpoint (via interconnection)	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_ExtOp	1 = Select external setpoint (via operator)	BOOL	0
SP_HiLim	Limit (high) of setpoint	REAL	100.0
SP_LoLim	Limit (low) of setpoint	REAL	0.0
SP_Int	Internal setpoint for operation	REAL	0.0
SP_IntLi	1 = Select internal setpoint (via interconnection)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP_IntOp	1 = Select internal setpoint (via operator)	BOOL	1
SP_LiOp	Select setpoint source (internal/external): 1 = Via interconnection 0 = Via operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
SP_OpScale	OS display range for setpoint	STRUCT • High: REAL • Low: REAL	- 100.0 0.0
SP_RateOn	1 = Activate limitation of setpoint gradients	BOOL	0
SP_RmpModTime	1 = Use time (SP_RmpTime) for setpoint ramp, 0 = Use gradient	BOOL	0
SP_RmpOn	1 = Activate setpoint ramp to target setpoint SP_RmpTarget	BOOL	0
SP_RmpTarget	Target setpoint for setpoint ramp	REAL	0.0
SP_RmpTime	Time for setpoint ramp [s] from current SP up to SP_RmpTarget	REAL	0.0
SP_TrkExt	1 = Bumpless switchover from external to internal setpoint active	BOOL	0
SP_Unit	Unit of measure for setpoint	INT	0
SP_UpRaLim	Gradient limit (high) for setpoint [SP_Unit/s]	REAL	100.0
SelFp1	Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
StepNo	Batch step number	DWORD	16#00000000
UserStatus	Freely assignable user area for status word	BYTE	16#00

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see OpAnL error handling (Page 206)	INT	-1
MsgAckn	Message acknowledgement status (output STATUS of first ALARM_8P)	WORD	16#0000
MsgErr	1 = Alarm error (output ERROR of first ALARM_8P)	BOOL	0
MsgStat	Alarm status (output ERROR of first ALARM_8P)	WORD	16#0000
OnAct	1 = "On" mode enabled	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
OosAct	1 = Block is "Out of service"	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
SP_Out	Setpoint used by controller	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_DnRaAct	1 = Negative gradient limiting of setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP_ExtAct	1 = External setpoint is active 0 = Internal setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP_HiAct	High limit for SP_Ext or SP_Int reached or exceeded	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP_LoAct	Low limit for SP_Ext or SP_Int reached or exceeded	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP_UpRaAct	1 = Positive gradient limiting of setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word (Page 201)	DWORD	16#00000000

See also

OpAnL messaging (Page 207)
OpAnL block diagram (Page 214)
OpAnL modes (Page 203)

2.1.7 OpAnL block diagram

OpAnL block diagram

A block diagram is not provided for this block.

See also

OpAnL I/Os (Page 209)
OpAnL error handling (Page 206)
OpAnL functions (Page 204)
Description of OpAnL (Page 201)
OpAnL modes (Page 203)
OpAnL messaging (Page 207)

2.1.8 Operator control and monitoring

2.1.8.1 Views of OpAnL

Views of the OpAnL block

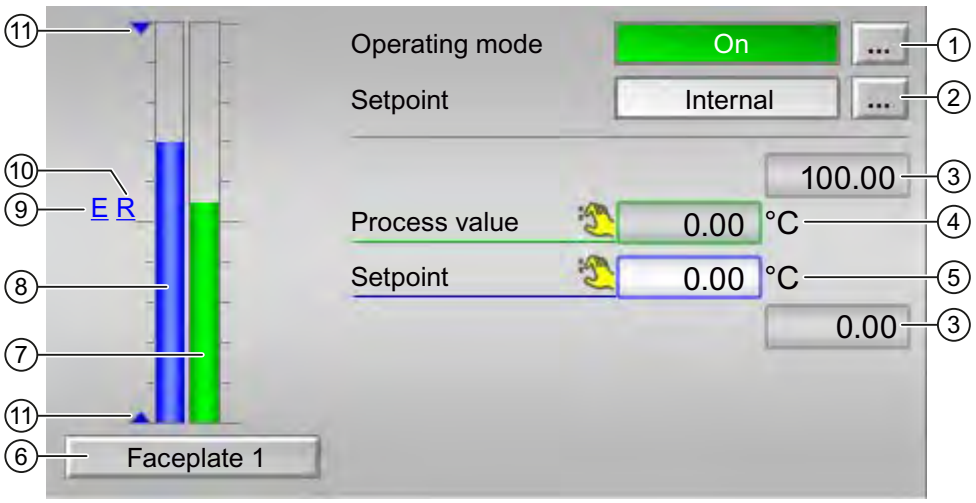
The block OpAnL provides the following views:

- OpAnL standard view (Page 215)
- Message view (Page 169)
- Trend view (Page 172)
- Ramp view (Page 167)
- OpAnL parameter view (Page 217)
- OpAnL preview (Page 218)
- Memo view (Page 171)
- Batch view (Page 170)
- Block icon for OpAnL (Page 219)

Refer to the sections Structure of the faceplate (Page 123) and Block icon structure (Page 174) for general information on the faceplate and block icon.

2.1.8.2 OpAnL standard view

OpAnL standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 27)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Display and switch the setpoint specification

This area shows how to specify the setpoint. The setpoint can specified as follows:

- By the application (External, CFC/SFC)
- By the user direct in the faceplate (Internal).

Refer to the Switching operating states and operating modes (Page 127) section for information on changing the setpoint specification.

You can find additional information on this in the Setpoint input - internal and external (Page 96) section.

(3) High and low scale range for the process value

These values provide information on the display range for the bar graph of the process value. The scale range is defined in the engineering system.

(4) Display of the process value including signal status

This area shows the current process value with the corresponding signal status.

(5) Display and change the setpoint including signal status

This area shows the current setpoint with the corresponding signal status.

Refer to the Changing values (Page 129) section for information on changing the setpoint. The setpoint specification also needs to be set to "Internal" for this block.

(6) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

See also the Calling further faceplates (Page 43) section for more on this.

(7) Bar graph for the process value

This area shows the current process value in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(8) Bar graph for the setpoint

This area shows the current setpoint in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(9) Display for external setpoint

This display [E] is only visible when you have selected "internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

(10) Display for the target setpoint of the setpoint ramp

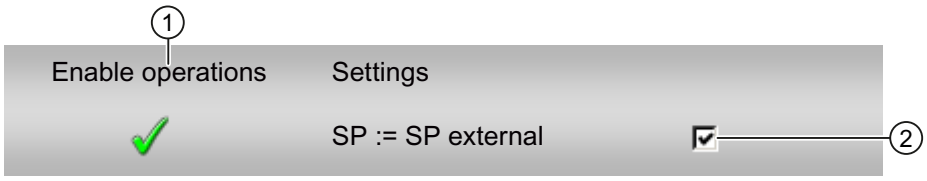
This display [R] shows the target setpoint and is only visible if you have enabled ramp generation in the Ramp view (Page 167).

(11) Display for limits

These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the ES.

2.1.8.3 OpAnL parameter view

OpAn parameter view



(1) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions.

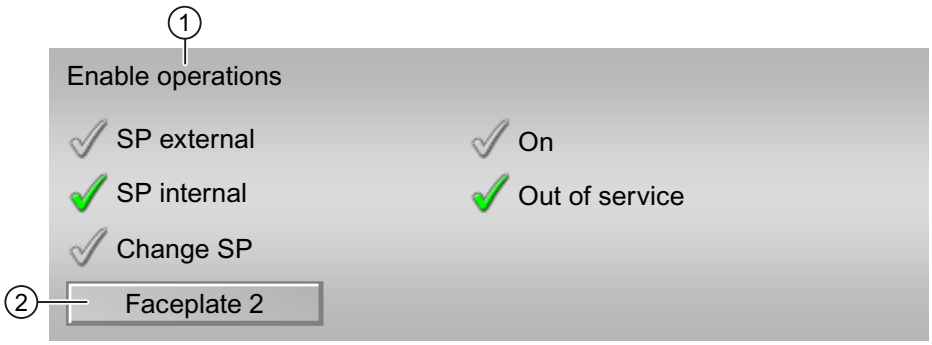
(2) Settings

You can select the following functions in this area:

- SP := SP external: ☑ Bumpless switchover of setpoint from external to internal. The internal setpoint is tracked to the external one.

2.1.8.4 OpAnL preview

OpAn preview



(1) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions

The following operator control enables are shown here:

- SP external: You can connect to the external setpoint.
- SP internal: You can connect to the internal setpoint.
- Change SP: You can change the setpoint.
- On: You can switch to On operating mode.
- Out of service: You can switch to the Out of service operating mode.

(2) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

Refer also to the Calling further faceplates (Page 43) section for more on this.

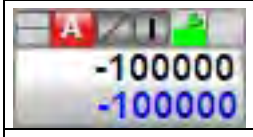
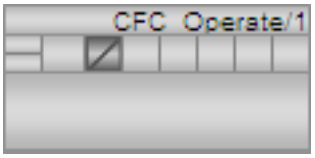
2.1.8.5 Block icon for OpAnL

Properties of the OpAnL block icon

A variety of block icons are available with the following functions:

- Process tag type
- Violation of alarm, warning, and tolerance limits
- Operating modes
- Internal and external setpoint specification
- Signal status, maintenance release
- Memo display

- Process value (black, with and without decimal places)
- Setpoint (blue, with and without decimal places)

Icons	Selection of the block icon in CFC	Special features
	1	Block icon in the full display
	2	
	3	
	4	
	5	
	6	
	7	
	8	
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181).

2.2 OpTrig - Manipulating a digital value (1 pushbutton)

2.2.1 Description of OpTrig

Object name (type + number) and family

Type + number: FB 1868

Family: Operate

Area of application for OpTrig

The block is used for the following applications:

- Generation of a pulse signal (trigger)

How it works

Operator control block is used to implement single pushbutton control (comparable with RESET pushbutton).

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for `Status1` parameter

For a description of the individual parameters, see the OpTrig I/Os (Page 228) section.

Status bit	Parameter
0 - 2	Not used
3	`OosAct.Value`
4	`OosLi.Value`
5	Not used
6	`OnAct.Value`
7	`Out.Value`
8	`LiOpAct.Value`
9	`FbkIn.Value`
10 - 31	Not used

See also

OpTrig functions (Page 225)
OpTrig messaging (Page 227)
OpTrig block diagram (Page 230)
OpTrig error handling (Page 227)
OpTrig modes (Page 224)

2.2.2 OpTrig modes

OpTrig operating modes

The block can be operated using the following modes

- On (Page 27)
- Out of service (Page 27)

"On"

General information on the "On" mode is available in the section On (Page 27).

"Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 27).

See also

OpTrig block diagram (Page 230)

OpTrig I/Os (Page 228)

OpTrig messaging (Page 227)

OpTrig functions (Page 225)

OpTrig error handling (Page 227)

Description of OpTrig (Page 222)

2.2.3 OpTrig functions

Functions of OpTrig

The functions for this block are listed below.

Issuing trigger signal internally or externally

Use the parameter `LiOp` to define whether the trigger signal is to be output by interconnection or by the OS operator:

`LiOp = 0`: Trigger signal by OS operator (input parameter `InOp`)

`LiOp = 1`: Trigger signal by interconnection (input parameter `InLi`)

Input parameter for feedback value

This block has a `FbkIn` input parameter for displaying a feedback value in the faceplate.

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `FbkIn.ST`
- `Out.ST`

Operator control permissions via parameter `OS_Perm`

The block has the following Operator permissions (Page 45) for the `OS_Perm` parameter:

Bit	Function
0	Not allocated
1	1 = Operator can switch to On mode
2	Not allocated
3	1 = Operator can switch to out of service mode
4	1 = Operator can set the input parameter `In`
5-31	Not allocated

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the section Functions that can be set via the Feature I/O (Page 186). The following functionality is available for this block at the relevant bits:

Bit	Function
1	Reaction to the out of service mode (Page 199)

See also

Description of OpTrig (Page 222)

OpTrig messaging (Page 227)

OpTrig I/Os (Page 228)

OpTrig block diagram (Page 230)

OpTrig error handling (Page 227)

OpTrig modes (Page 224)

2.2.4 OpTrig error handling

OpTrig error handling

The block does not report any errors.

See also

OpTrig block diagram (Page 230)
OpTrig I/Os (Page 228)
OpTrig messaging (Page 227)
OpTrig functions (Page 225)
OpTrig modes (Page 224)
Description of OpTrig (Page 222)

2.2.5 OpTrig messaging

Messaging

The block does not offer messaging.

See also

Description of OpTrig (Page 222)
OpTrig functions (Page 225)
OpTrig I/Os (Page 228)
OpTrig block diagram (Page 230)
OpTrig error handling (Page 227)
OpTrig modes (Page 224)

2.2.6 OpTrig I/Os

OpTrig I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
FbkIn	Input for feedback	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Features	I/O for additional functions (Page 225)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
InLi	Interconnectable binary input	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
InOp	Binary input for operator	BOOL	0
LiOp	Switchover of operating mode between: 1 = Interconnection 0 = Operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OnOp	1 = "On" mode via operator	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OS_Perm	I/O for operator control permissions (Page 225)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
SelFp1	Call a block saved in this parameter as additional faceplate (Page 43) in standard view	ANY	-
SelFp2	Call a block saved in this parameter as additional faceplate (Page 43) in the preview	ANY	-
UserStatus	Freely assignable user area for status word	BYTE	16#00

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Reserved		
LiOpAct	Operator/interconnection is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosAct	1 = Block is out of service	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OnAct	1 = "On" mode enabled	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
Out	Output	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Status	Status word (Page 222)	DWORD	16#00

See also

OpTrig messaging (Page 227)

OpTrig block diagram (Page 230)

OpTrig error handling (Page 227)

OpTrig modes (Page 224)

2.2.7 OpTrig block diagram

OpTrig block diagram

This block does not come with a block diagram.

See also

OpTrig I/Os (Page 228)

OpTrig messaging (Page 227)

OpTrig functions (Page 225)

OpTrig error handling (Page 227)

OpTrig modes (Page 224)

Description of OpTrig (Page 222)

2.2.8 Operator control and monitoring

2.2.8.1 Views of OpTrig

Views of the OpTrig block

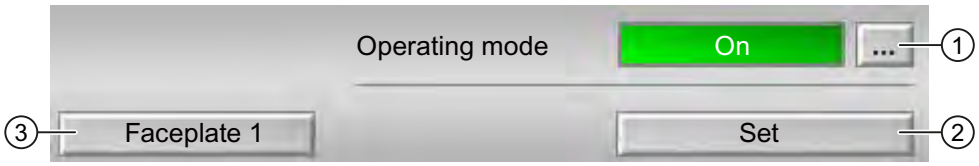
The block OpTrig provides the following views:

- OpTrig standard view (Page 231)
- OpTrig preview (Page 232)
- Memo view (Page 171)
- Block symbol for OpTrig (Page 233)

Refer to the sections Structure of the faceplate (Page 123) and Block icon structure (Page 174) for general information on the faceplate and block icon.

2.2.8.2 OpTrig standard view

OpTrig standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 27)
- Out of service (Page 27)

You can find additional information on this in the Switching operating states and operating modes (Page 127) section.

(2) Set

By clicking "Set", a pulse signal with the length of the cycle time is output at the `Out` output.

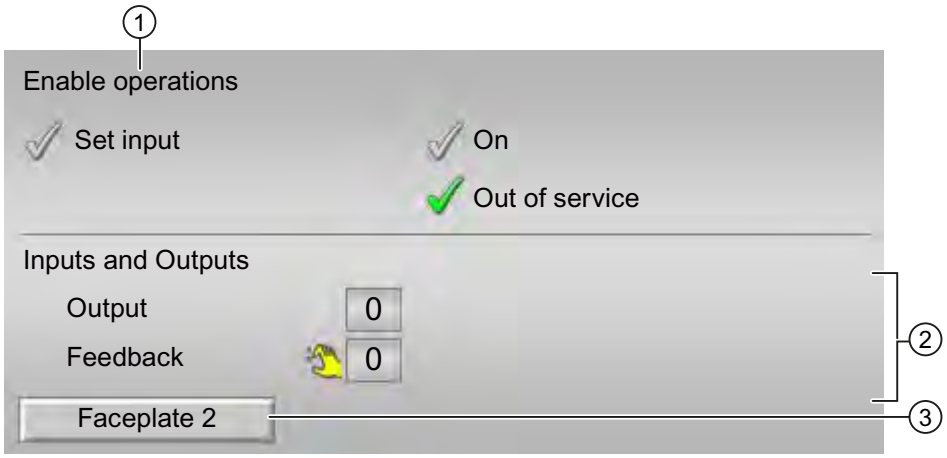
(3) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

2.2.8.3 OpTrig preview

OpTrig preview



(1) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions

The following operator control enables are shown here:

- Set input: You can set the input.
- On: You can switch to On operating mode.
- Out of service: You can switch to the "Out of service" mode.

(2) Display of current inputs and outputs

This area shows the most important parameters for this block with the current selection:

- Output: 1= Digital output value set
- Feedback: 1= Feedback set

(3) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

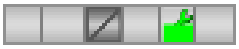
Refer also to the Calling further faceplates (Page 43) section for more on this.

2.2.8.4 Block symbol for OpTrig

Properties of the OpTrig block icon

A variety of block icons are available with the following functions:

- Process tag type
- Operating modes
- Signal status, maintenance release
- Memo display

Icons	Selection of the block icon in CFC	Special features
	1	Block icon in the full display
	2	
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181).

2.3 OpDi01 - Manipulating a digital value (2 pushbuttons)

2.3.1 Description of OpDi01

Object name (type + number) and family

Type + number: FB 1866

Family: Operate

Area of application for OpDi01

The block is used for the following applications:

- Manipulating a digital value

How it works

A digital value is manipulated by interconnection or via the faceplate.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for `Status1` parameter

For a description of the individual parameters, see the OpDi01 I/Os (Page 240) section.

Status bit	Parameter
0 - 2	Not used
3	`OosAct.Value`
4	`OosLi.Value`
5	Not used
6	`OnAct.Value`
7	`Out.Value`
8	`LiOp`
9	`FbkIn.Value`
10 - 15	Not used
16	`1 = Intlock` is active
17 - 22	Not used
23	"Interlock" button is enabled
24-25	Not used
26	Bypass information from previous function block
27 - 31	Not used

See also

OpDi01 functions (Page 237)
OpDi01 messaging (Page 239)
OpDi01 block diagram (Page 242)
OpDi01 error handling (Page 239)
OpDi01 modes (Page 236)

2.3.2 OpDi01 modes

OpDi01 operating modes

The block can be operated using the following modes:

- On (Page 27)
- Out of service (Page 27)

"On"

General information on the "On" mode is available in the section On (Page 27).

"Out of service"

You can find general information about the "Out of service" mode in the section Out of service (Page 27).

See also

OpDi01 block diagram (Page 242)

OpDi01 I/Os (Page 240)

OpDi01 messaging (Page 239)

OpDi01 error handling (Page 239)

OpDi01 functions (Page 237)

Description of OpDi01 (Page 234)

2.3.3 OpDi01 functions

Functions of OpDi01

The functions for this block are listed below.

Internal or external digital value

Use the `LiOp` input parameter to define whether the digital value is set (0-1, parameter `SetOp` or `SetLi`) or is reset (1-0, parameter `RstOp` or `RstLi`) via the faceplate or an interconnection.

`LiOp = 0`: Specification of digital value by faceplate (`SetOp` or `RstOp`)

`LiOp = 1`: Specification of digital value by interconnection (`SetLi` or `RstLi`)

Interlocks

Use the `Intl_En = 1` and `Intlock.ST ≠ 16#FF` input parameters to activate the interlock function on this block.

An active interlock condition brings the block to the safe position (`Intlock.Value = 0` or `Intlock.ST = 16#00` input). Output parameter `Out` is set to 0. When the interlocking condition no longer applies, the digital value currently valid is output again.

Input parameter for feedback value

This block has a `FbkIn` input parameter for displaying a feedback value in the faceplate.

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `Out.ST`
- `FbkIn.ST`
- `Intlock.ST`

Operator control permissions via parameter OS_Perm

The block has the following Operator permissions (Page 45) for the OS_Perm parameter:

Bit	Function
0	Not allocated
1	1 = Operator can switch to On mode
2	Not allocated
3	1 = Operator can switch to out of service mode
4	1 = Operator can set the digital value
5	1 = Operator can reset the digital value
6-31	Not allocated

Configurable reactions using the Feature parameter

You can find an overview of all reactions provided by the Feature parameter in the section Functions of blocks > Functions that can be set via the Feature I/O (Page 186).

The following functionality is available for this block at the relevant bits:

Bit	Function
0	Not used
1	Reaction to the out of service mode (Page 199)
2 - 31	Not used

See also

Description of OpDi01 (Page 234)

OpDi01 messaging (Page 239)

OpDi01 I/Os (Page 240)

OpDi01 block diagram (Page 242)

OpDi01 error handling (Page 239)

OpDi01 modes (Page 236)

2.3.4 OpDi01 error handling

OpDi01 trouble shooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Default value when implementing the block; block will not be processed.
0	There is no error.
51	`SetLi = 1` and `RstLi = 1` and `LiOp = 1`

See also

OpDi01 block diagram (Page 242)

OpDi01 I/Os (Page 240)

OpDi01 messaging (Page 239)

OpDi01 functions (Page 237)

OpDi01 modes (Page 236)

Description of OpDi01 (Page 234)

2.3.5 OpDi01 messaging

Messaging

The block does not have any message functionality.

See also

Description of OpDi01 (Page 234)

OpDi01 functions (Page 237)

OpDi01 I/Os (Page 240)

OpDi01 block diagram (Page 242)

OpDi01 error handling (Page 239)

OpDi01 modes (Page 236)

2.3.6 OpDi01 I/Os

OpDi01 I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
FbkIn	Input for feedback	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Features	I/O for additional functions (Page 237)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
Intlock	0 = Interlocking without reset is effective; you can operate the block without reset once the interlocking condition has disappeared 1 = Good state	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#FF
Intl_En	1 = Interlock without reset (interlock, parameter Intlock) can be used	BOOL	1
LiOp	Switchover of operating mode between: 1 = Interconnection 0 = Operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OnOp	1 = "On" mode via operator	BOOL	0
OosLi	Edge transition (0-1) = Out of service, via interconnection or SFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OS_Perm	I/O for operator control permissions (Page 237)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
RstLi	Reset via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RstOp	Reset by the operator	BOOL	0
SelFp1	Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-

Parameter	Description	Type	Default
SetLi	Connected digital input	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SetOp	Digital input for operator	BOOL	0
UserStatus	Freely assignable user area for status word	BYTE	16#00

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see OpDi01 error handling (Page 239)	INT	-1
OosAct	1 = Block is out of service	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OnAct	1 = "On" mode enabled	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
Out	Digital output value	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Status	Status word (Page 234)	DWORD	16#0000000

See also

OpDi01 messaging (Page 239)

OpDi01 block diagram (Page 242)

OpDi01 modes (Page 236)

2.3.7 OpDi01 block diagram

OpDi01 block diagram

This block does not come with a block diagram.

See also

OpDi01 I/Os (Page 240)
OpDi01 messaging (Page 239)
OpDi01 error handling (Page 239)
OpDi01 functions (Page 237)
OpDi01 modes (Page 236)
Description of OpDi01 (Page 234)

2.3.8 Operator control and monitoring

2.3.8.1 Views of OpDi01

Views of the OpDi01 block

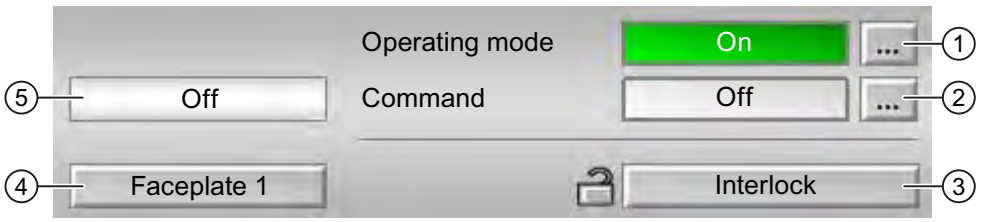
The block OpDi01 provides the following views:

- OpDi01 standard view (Page 243)
- Trend view (Page 172)
- OpDi01 preview (Page 245)
- Memo view (Page 171)
- Block symbol for OpDi01 (Page 247)

Refer to the Structure of the faceplate (Page 123) and Block icon structure (Page 174) sections for general information about the faceplate and block icon.

2.3.8.2 OpDi01 standard view

OpDi01 standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 27)
- Out of service (Page 27)

You can find additional information on this in the Switching operating states and operating modes (Page 127) section.

(2) Display and switch the command

This area shows you the current selection. You can output a continuous signal as follows:

- On: Continuous signal is output
- Off

You can find additional information on this in the Switching operating states and operating modes (Page 127) section.

(3) Display area for interlock functions of the block

You can use this button to control the interlock functions of the block. You can find additional information on this in the Interlocks (Page 100) section.

(4) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

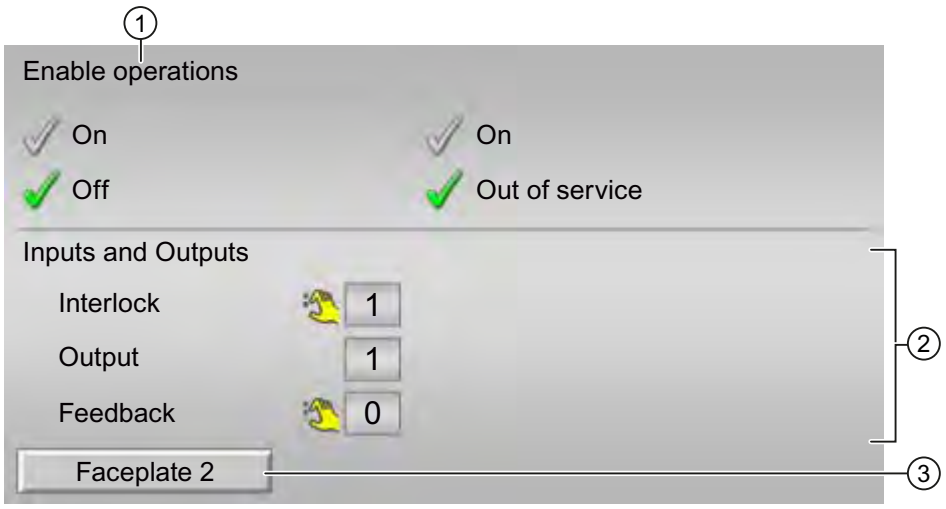
(5) Display the feedback of the command

This area shows you the currently valid command. The following commands can be shown here:

- On
- Off

2.3.8.3 OpDi01 preview

OpDi01 preview



(1) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions

The following operator control enables are shown here:

- On: You can set the digital value (0-1 edge).
- Off: You can set the digital value (1-0 edge).
- On: You can switch to On operating mode.
- Out of service: You can switch to the "Out of service" mode.

(2) Display of current inputs and outputs

This area shows the most important parameters for this block with the current selection:

- Interlock:
  - 0 = Interlocking without reset is effective; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = Good state
- Output: 1= Digital output value set
- Feedback: 1= Feedback set

(3) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

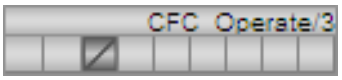
Refer also to the Calling further faceplates (Page 43) section for more on this.

2.3.8.4 Block symbol for OpDi01

Properties of the OpDi01 block icon

A variety of block icons are available with the following functions:

- Process tag type
- Operating modes
- Signal status, maintenance release
- Bypass
- Interlocks
- Output signal
- Memo display

Icons	Selection of the block icon in CFC	Special features
	1	Block icon in the full display
	2	
	3	
	4	
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181).

2.4 OpDi03 - Manipulating a digital value (3 pushbuttons)

2.4.1 Description of OpDi03

Object name (type + number) and family

Type + number: FB 1867

Family: Operate

Area of application for OpDi03

The block is used for the following applications:

- Manipulating a digital value (3 pushbuttons)

How it works

A digital value is manipulated at three possible outputs by interconnection or via the faceplate.

If two or three input parameters are set for an interconnection (`Parameter SetLix`), the input parameter with the highest index will be set to the corresponding output parameter. For example, if the `SetLi1` and `SetLi2` input parameters are set (=1), `Out2` will be set (=1).

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for `Status1` parameter

For a description of the individual parameters, see the OpDi03 I/Os (Page 256) section.

Status bit	Parameter
0 - 2	Not used
3	`OosAct.Value`
4	`OosLi.Value`
5	Not used
6	`OnAct.Value`
7	`Out1.Value`
8	`Out2.Value`
9	`Out3.Value`
10	`LiOp.Value`
11	`Fbk1In.Value`
12	`Fbk2In.Value`
13	`Fbk3In.Value`
14-15	Not used
16	`1 = Intlock` is active
17 - 22	Not used
23	"Interlock" button is enabled
24-25	Not used
26	Bypass information from previous function block
27 - 31	Not used

See also

OpDi03 functions (Page 251)
OpDi03 messaging (Page 255)
OpDi03 block diagram (Page 259)
OpDi03 error handling (Page 254)
OpDi03 modes (Page 250)

2.4.2 OpDi03 modes

OpDi03 operating modes

The block can be operated using the following modes

- On (Page 27)
- Out of service (Page 27)

"On"

You can find general information about the "On" mode in the On (Page 27) section.

"Out of service"

You can find general information about the "Out of service" mode in the Out of service (Page 27) section.

If you switch to out of service mode, the `Out3` output parameter is set (=1).

See also

OpDi03 block diagram (Page 259)

OpDi03 I/Os (Page 256)

OpDi03 messaging (Page 255)

OpDi03 error handling (Page 254)

OpDi03 functions (Page 251)

Description of OpDi03 (Page 248)

2.4.3 OpDi03 functions

Functions of OpDi03

The functions for this block are listed below.

Internal or external digital value

Use the `LiOp` input parameter to define whether the digital value is set (0-1) or reset (1-0, input parameter `RstOut`) using the faceplate or an interconnection by selecting 1 from 3.

`LiOp = 0`: Specification of digital value by faceplate. One of the input parameters `SetOp1`, `SetOp2` or `SetOp3` is now led to the relevant output `Out1`, `Out2` or `Out3`. For example if `SetOp2 = 1`, then `Out2 = 1`.

`LiOp = 1`: Specification of digital value by interconnection. One of the input parameters `SetLi1`, `SetLi2` or `SetLi3` is now led to the relevant output `Out1`, `Out2` or `Out3`. For example if `SetLi2 = 1`, then `Out2 = 1`.

The reset (1-0) is always undertaken using the `RstOut` input parameter.

Interlocks

Use the `Intl_En = 1` and `Intlock.ST ≠ 16#FF` input parameters to activate the interlock function on this block.

An active interlock condition brings the block to the safe position (`Intlock.Value = 0` or `Intlock.ST = 16#00` input). Output parameter `Out` is set to 0. When the interlocking condition no longer applies, the digital value currently valid is output again.

Input parameter for feedback value

This block has three input parameters `Fbk1In`, `Fbk2In` and `Fbk3In` for displaying three feedback values in the faceplate.

Resetting all output values

You can reset all output parameters (`Out1` to `Out3`) by setting all interconnected input parameters to be set (`SetLi1` to `SetLi3`) or operable input parameters to be set (`SetOp1` to `SetOp3`) to 0.

A 0-1 edge at the `RstOut` parameter then resets the three output parameters `Out1` to `Out3`.

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `Out1.ST`
- `Out2.ST`
- `Out3.ST`
- `Fbk1In.ST`
- `Fbk2In.ST`
- `Fbk3In.ST`
- `Intlock.ST`

Operator control permissions via parameter `OS_Perm`

The block has the following Operator permissions (Page 45) for the `OS_Perm` parameter:

Bit	Function
0	Not allocated
1	1 = Operator can switch to On mode
2	Not allocated
3	1 = Operator can switch to out of service mode
4	1 = Operator can set the digital value `SetOp1`
5	1 = Operator can set the digital value `SetOp2`
6	1 = Operator can set the digital value `SetOp3`
7-31	Not allocated

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the section Functions of blocks > Functions that can be set via the Feature I/O (Page 186).

The following functionality is available for this block at the relevant bits:

Bit	Function
0	Not used
1	Reaction to the out of service mode (Page 199)
2 - 31	Not used

See also

Description of OpDi03 (Page 248)

OpDi03 messaging (Page 255)

OpDi03 I/Os (Page 256)

OpDi03 block diagram (Page 259)

OpDi03 error handling (Page 254)

OpDi03 modes (Page 250)

2.4.4 OpDi03 error handling

OpDi03 troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Default value when implementing the block; block will not be processed.
0	There is no error.
51	The value 1 is set at two or more inputs. (`SetLi1 = 1` and `SetLi2 = 1`) or (`SetLi2 = 1` and `SetLi3 = 1`) or (`SetLi1 = 1` and `SetLi3 = 1`) and `LiOp = 1`

See also

OpDi03 block diagram (Page 259)

OpDi03 I/Os (Page 256)

OpDi03 messaging (Page 255)

OpDi03 functions (Page 251)

OpDi03 modes (Page 250)

Description of OpDi03 (Page 248)

2.4.5 OpDi03 messaging

Messaging

The block does not offer messaging.

See also

Description of OpDi03 (Page 248)
OpDi03 functions (Page 251)
OpDi03 I/Os (Page 256)
OpDi03 block diagram (Page 259)
OpDi03 error handling (Page 254)
OpDi03 modes (Page 250)

2.4.6 OpDi03 I/Os

OpDi03 I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
Fbk1In	Feedback for In1 input	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Fbk2In	Feedback for In2 input	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Fbk3In	Feedback for In3 input	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Feature	I/O for additional functions (Page 251)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
Intlock	0 = Interlocking without reset is effective; you can operate the block without reset once the interlocking condition has disappeared 1 = Good state	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
Intl_En	1 = Interlock without reset (interlock, parameter Intlock) can be used	BOOL	1
LiOp	Switchover of operating mode between: 1 = Interconnection 0 = Operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OnOp	1 = "On" mode via operator	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OS_Perm	I/O for operator control permissions (Page 251)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
RstOut	Reset by the operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
SelFp1	Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
SetLi1	Connected digital input 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SetLi2	Connected digital input 2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SetLi3	Connected digital input 3	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SetOp1	Digital input 1 for operator	BOOL	0
SetOp2	Digital input 2 for operator	BOOL	0
SetOp3	Digital input 3 for operator	BOOL	0
UserStatus	User status bits	BYTE	16#00

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see OpDi03 error handling (Page 254)	INT	-1
OosAct	1 = Block is out of service	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OnAct	1 = "On" mode enabled	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
OS_PermOut	Display of OS_Perm	DWORD	16#FF
Out1	Digital output value 1	BOOL	0
Out2	Digital output value 2	BOOL	0
Out3	Digital output value 3	BOOL	0
ST_Worst	Worst signal status	BYTE	16#80
Status	Status word (Page 248)	DWORD	16#00000000

See also

OpDi03 messaging (Page 255)

OpDi03 block diagram (Page 259)

OpDi03 modes (Page 250)

2.4.7 OpDi03 block diagram

OpDi03 block diagram

This block does not come with a block diagram.

See also

OpDi03 I/Os (Page 256)
OpDi03 messaging (Page 255)
OpDi03 error handling (Page 254)
OpDi03 functions (Page 251)
OpDi03 modes (Page 250)
Description of OpDi03 (Page 248)

2.4.8 Operator control and monitoring

2.4.8.1 Views of OpDi03

Views of the OpDi03 block

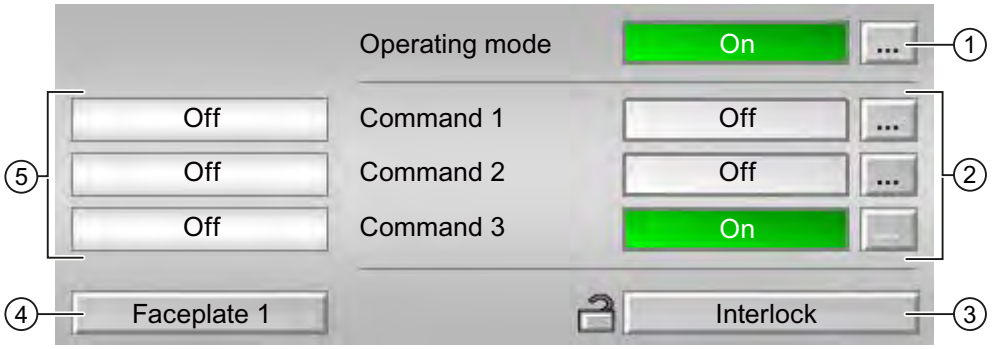
The block OpDi03 provides the following views:

- OpDi03 standard view (Page 260)
- Trend view (Page 172)
- OpDi03 preview (Page 262)
- Memo view (Page 171)
- Block symbol for OpDi03 (Page 264)

Refer to the Structure of the faceplate (Page 123) and Block icon structure (Page 174) sections for general information about the faceplate and block icon.

2.4.8.2 OpDi03 standard view

OpDi03 standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 27)
- Out of service (Page 27)

You can find additional information on this in the Switching operating states and operating modes (Page 127) section.

(2) Display and switch the commands 1 to 3

This area shows you the current selection. You can output a continuous signal at the outputs `Out1` to `Out3` as follows:

- On: Continuous signal is output
- Off

You can find additional information on this in the Switching operating states and operating modes (Page 127) section.

(3) Display area for interlock functions of the block

You can use this button to control the interlock functions of the block. You can find additional information on this in the Interlocks (Page 100) section.

(4) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

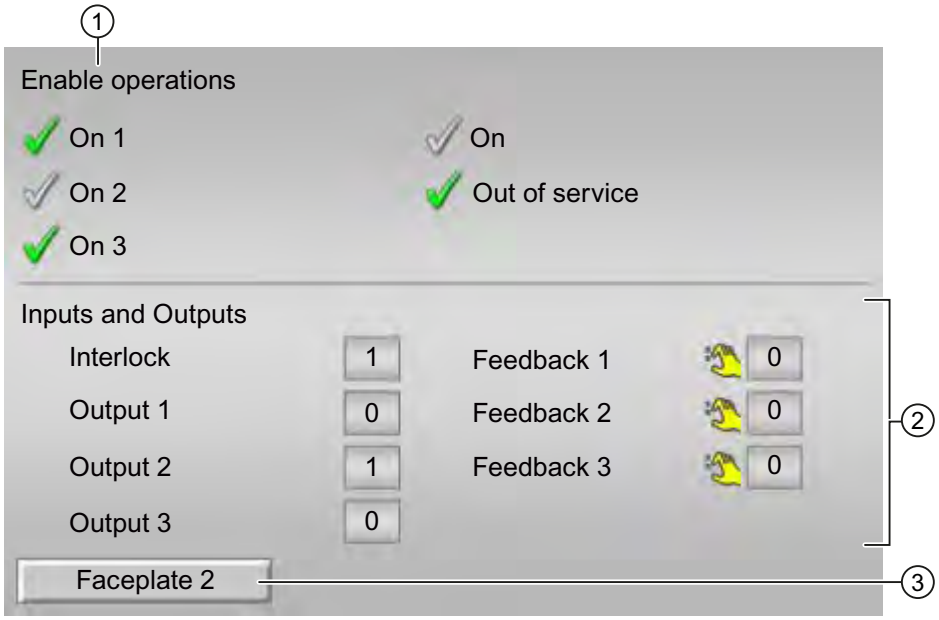
You can find additional information on this in the Calling further faceplates (Page 43) section.

(5) Display feedback from commands 1 to 3

This area shows you the current valid selection from `Out1` to `Out3`.

2.4.8.3 OpDi03 preview

OpDi03 preview



(1) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions

The following operator control enables are shown here:

- On 1 to 3: You can set the digital value (0-1 edge) for each.
- On (operating mode): You can switch to On operating mode.
- Out of service (operating mode): You can switch to the Out of service operating mode.

(2) Display of current inputs and outputs

This area shows the most important parameters for this block with the current selection:

- Interlock:
  - 0 = Interlocking without reset is effective; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = Good state
- Output 1 to 3: 1= Digital output value set
- Feedback 1 to 3: 1= Feedback set

(3) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

Refer also to the Calling further faceplates (Page 43) section for more on this.

2.4.8.4 Block symbol for OpDi03

Properties of the OpDi03 block icon

A variety of block icons are available with the following functions:

- Process tag type
- Operating modes
- Signal status, maintenance release
- Bypass
- Interlocks
- Output signal
- Memo display

Icons	Selection of the block icon in CFC	Special features
	1	Block icon in the full display
	2	
	3	
	4	
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181).

Generator blocks

3

3.1 NoiseGen - Rauschgenerator

3.1.1 Description of NoiseGen

Object name (type + number) and family

Object name: FB 1863

Family: Genrator

Area of application for NoiseGen

The block is used for the following applications:

- Noise generator

You do not need this block.

How it works

This block serves to generate signal noise. It is used for demonstration purposes in the example project so that the simulated signals react naturally.

You can employ it for simulation examples, demonstrations, trade fair models etc. It is never needed in real plants because real measuring devices always supply signals with noise.

There is an example project for the NoiseGen block (APL_Example_xx, xx refers to the language variant) with an application scenario for this block, which explains how the block works.

Application scenario in the example project:

- Process simulation including noise generator (Page 1456)

See also

NoiseGen I/Os (Page 266)

3.1.2 NoiseGen I/Os

NoiseGen I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
Enable	1 = generate noise signals	BOOL	1
Offset	Mean temporal value of the Noise output signal	REAL	20.0
Restart	1 = block restart	BOOL	1
StdDev	Standard deviation of the noise signal	REAL	1.0

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
Noise	Generated noise signal	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

See also

Description of NoiseGen (Page 265)

Control blocks

4

4.1 ConPerMon - monitoring of the control performance of control loops

4.1.1 Description of ConPerMon

Object name (type + number) and family

Type + number: FB 1805

Family: Control

Area of application for ConPerMon

The block is used for the following applications:

- Permanent monitoring of control performance of control loops for early detection of problems as they develop

The block calculates:

- Stochastic characteristics of the control performance with the process in a steady state
  - Mean value, variance and standard deviation of controlled variable
  - Mean value of the manipulated variable and control deviation
  - Control performance index
  - Estimated steady state process gain
- Deterministic characteristics of the control performance with step changes in the setpoint
  - Response time and settling time and the settling ratio
  - Overshoot absolute and relative to the step height

Other statistical and graphic evaluations of the signals in the control loop over longer, freely selectable periods are available in the faceplate of the ConPerMon block.

In an overview representation of a plant or unit, you can obtain a clear picture of the status of all control loops based on ConPerMon block icons (indicator light function).

The aim is to detect problems as they develop and to focus the attention of the user on the control loops in a plant that are no longer operating correctly.

How it works

The ConPerMon block evaluates the setpoint and process value signals and the manipulated variable of the PID controller in a sliding time window. The mode of the controller is also taken into account.

With the process in a steady state, the detected stochastic characteristics are compared with the reference values obtained during commissioning. If there is a step change in the setpoint, the stochastic characteristics are by definition irrelevant and are temporarily frozen. Instead, the monitoring of the deterministic characteristics is automatically activated.

If the control performance falls below a defined limit a message is generated. This is also the case when a defined limit for overshoot is exceeded when there is a step change in the setpoint.

Configuration

Each PID controller has a ConPerMon block assigned to it that is installed in the same CFC chart and interconnected with the controller. This already takes place with the corresponding process tag types.

You can open the standard view for the ConPerMon block from the standard view of a controller (for example PIDConL). Additional information on this topic is available in the section Calling further faceplates (Page 43).

After successful commissioning and optimization of the PID controller to be monitored, the ConPerMon block is initialized while the process is in a steady state and it stores the corresponding characteristic values as reference values.

Follow the steps outlined below:

- Change the PID controller you want to monitor to automatic mode and set the setpoint to the typical operating point. This operating status is intended to represent the normal operation of the process; in other words, the entire plant/unit should be running under production conditions. Monitor the process with a trend writer (CFC trend in the Engineering-System or WinCC Online-Trend-Control on the Operator Station) and wait until the process has settled
- To specify the length `TimeWindow` of the sliding time window, monitor the `PV_Variance` block output of the ConPerMonblock in a trend. The time window should be long enough to keep the variance fairly constant in the relevant decimal places. If the selected time window is too short in relation to the time constants in the control loop and the disturbance signal spectrum, the variance will have too much noise and no useful information.

  If the selected time window is too long, it takes longer before any deterioration of the control quality is detected by the ConPerMon block. It also takes longer following a step change in the setpoint before the monitoring of the stochastic characteristics can be resumed. A good starting value for the `TimeWindow` parameter is 10 times as long as the longest process time constant or 20 times as long as the reset time of the PID controller.

- If the controller
  - is set perfectly,
  - has achieved a steady state,
  - the time window has been defined and filled with values from the steady state,

  the ConPerMon block can be initialized. You do this by clicking the "Initialize" button in the parameter view of the ConPerMon faceplate or by setting the `InitRefVar = 1` parameter in the CFC block. This saves the `PV_Variance` parameter in the current time window as a reference value for calculating the control quality in the block along with reference values for manipulated variable and process variable.

The **C**ontrol **P**erformance **I**ndexCPI should now be approximately at 100% and therefore indicate that the control loop is operating correctly. Due to stochastic fluctuations, the CPI can also temporarily exceed the 100% mark. If, however, the CPI drops by a significant amount over a longer period, this indicates deterioration of the control performance.

For more detailed information on interpreting the calculation results of the block, refer to the section ConPerMon functions (Page 273).

Note

If the length of the time window is changed during runtime, the CPI will temporarily deviate considerably from its old value and then gradually settle to the new steady value. It is advisable to reinitialize the ConPerMon block after the CPI value has settled to a constant level.

The ConPerMon faceplate is opened from the faceplate of the assigned PID controller so that the ConPerMonicon pictures do not need to be installed separately in each OS picture. It is, in fact, advisable to group all ConPerMon block icons of a plant or unit in one overview picture at the appropriate hierarchy level.

You can expand this overview picture with the trend display of the control quality of all control loops over a longer time to allow you to recognize gradual deterioration (for example reflecting wear and tear). You can also display a further view of the message archive (WinCC AlarmLogging Control) as a hit list sorted according to the frequency in which they occur. In this list, the control loops that caused the most alarms will be shown at the top.

For the ConPerMon block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 1433)
- PID - control with operating-point-oriented parameter control (GainScheduling) (Page 1431)
- PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 1430)
- PIDConR with safety logic and control loop monitoring (PIDConR_ConPerMon) (Page 1431)
- Ratio control (Page 1440)
- Ratio control with PIDConR (RatioR) (Page 1441)

- PID controller with Smith predictor (SmithPredictorControl) (Page 1436)
- Step controller with direct access to the actuator and without position feedback (StepControlDirect) (Page 1437)

Startup characteristics

When the CPU starts up, the block is reinitialized but the stored reference values are retained. The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

Status word allocation for `Status1` parameter

Status bit	Parameter
0	`Occupied`
1	`BatchEn`
2	Not allocated
3	`OosAct.Value`
4	`OosLi.Value`
5	`PID_AutAct.Value`
6	`OnAct.Value`
7 - 14	Not allocated
15	`CPI_Suppress`
16 - 31	Not allocated

Status word allocation for `Status2` parameter

Status bit	Parameter
0	`MsgLock`
1	`OvsAH_Act.Value`
2	`OvsWH_Act.Value`
3 - 4	Not allocated
5	`CPI_WL_Act.Value`
6	`CPI_AL_Act.Value`
7	`OvsAH_En`
8	`OvsWH_En`
9 - 10	Not allocated
11	`CPI_WL_En`
12	`CPI_AL_En`
13	`OvsAH_MsgEn`
14	`OvsWH_MsgEn`
15 - 16	Not allocated
17	`CPI_WL_MsgEn`

Status bit	Parameter
18	CPI_AL_MsgEn
19 - 31	Not allocated

See also

ConPerMon messaging (Page 285)
ConPerMon I/Os (Page 287)
ConPerMon block diagram (Page 293)
ConPerMon error handling (Page 284)
ConPerMon modes (Page 272)

4.1.2 ConPerMon modes

ConPerMon operating modes

The block can be operated using the following modes

- On (Page 27)
- Out of service (Page 27)

"On"

You can find general information about the "On" mode in the On (Page 27) section.

"Out of service"

You can find general information about the "Out of service" mode in the Out of service (Page 27) section.

See also

ConPerMon block diagram (Page 293)

ConPerMon I/Os (Page 287)

ConPerMon messaging (Page 285)

ConPerMon error handling (Page 284)

ConPerMon functions (Page 273)

Description of ConPerMon (Page 267)

4.1.3 ConPerMon functions

Functions of ConPerMon

The functions for this block are listed below.

Monitoring of stochastic characteristics of the control performance

The mean value of a variable relating to an ergodic stochastic process (Page 1468) can be determined from a sliding time window with the length `n = TimeWindow / SampleTime`, for example for the `y = PV` controlled variable:

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y(i)$$

A recursive formulation of this calculation is included in the MeanTime block that is called by the ConPerMonblock. Most steady-state time series can be considered as being ergodic so that the expected value can be estimated by averaging over a window of finite length.

The mean control deviation is `ER_Mean = SP - PV_Mean`. A mean steady-state control deviation ≠ 0 at a constant setpoint is an indication for problems in the control loop if the controller has I action. You should then check the following potential causes:

- The actuator does not have sufficient capacity. As a result, the controller's manipulated signal constantly approaches its limit. This can be caused by unsuitably dimensioned actuators or may simply by wear and tear.
- The manipulated variable demanded by the controller does not take effect in the process, for example because the actuator is defective.

If a steady-state reference operating point (`MV_Ref` `PV_Ref`) is known, this can be used to estimate the current mean steady state gain of a linear process model if it is assumed that only disturbances with zero mean have an effect:

$$\mathrm{StatGain} = \frac{\mathrm{PV}_{\mathrm{Mean}} - \mathrm{PV}_{\mathrm{Ref}}}{\mathrm{MV}_{\mathrm{Mean}} - \mathrm{MV}_{\mathrm{Ref}}}$$

Normally the reference operating point is obtained during the initialization of the ConPerMon block. Estimation of the steady state gain is then, however, impossible precisely at this operating point. As an alternative, you can also enter the reference values `PV_Ref` and `MV_Ref` manually at the appropriate block inputs. Typical steady-state operating points are often known in advance, for example

- Flow control: `PV = 0` for `MV = 0`, in other words, valve closed,
- Temperature control: `PV = PV_Ambient` for `MV = 0`, in other words, ambient temperature

If the steady state gain changes gradually as time progresses, this is an indication of wear phenomena in the process, such as deposits on heat exchangers, valves or shutters, failing efficiency of process plant, etc.

If, for example, a temperature regulation circuit is closed by a heat exchanger and a deposit forms on the exchanger surfaces, the heat transfer coefficient, and consequently the process gain, is reduced. Within certain limits, this can be compensated by a closed control loop (so that the controller initially disregards the problem). Although the original control loop dynamics can be restored (to a certain extent) by suitable increase of the controller gain as the pollution increases, it is advisable to eliminate the cause of the problem; in other words, to clean the heat exchanger.

If the estimated steady state gain changes suddenly and temporarily, this tends to point to an external disturbance. This may be a normal occurrence in the operation of the process. If, however, these occurrences become more frequent, it is worth finding out the cause.

Due to the approach, the variance `PV_Variance` as second moment requires the calculation of differences of each current measured value from (constant !) mean value:

$$\sigma_y^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(y(i)-\bar{y}\right)^2 = \frac{1}{n}\left(\sum_{i=1}^{n} y^2(i)\right) - \bar{y}^2$$

Within the function block, however, a variant of the calculation is used that saves computing time. The standard deviation

$$PV\_StdDev = \sigma_y = \sqrt{\sigma_y^2}$$

as the square root of the variance is easier to interpret because it has the same physical unit as the measured value.

The control performance index CPI (**C**ontrol **P**erformance **I**ndex) in the unit [%] describes the current variance of the controlled variable relative to a benchmark (benchmark). It is defined as

$$\varsigma = \frac{\sigma_{ref}^2}{\sigma_y^2} 100\%$$

The CPI moves in the $0 < \varsigma \leq 100\%$ range. If the current variance corresponds to the benchmark, the index reaches the value 100. If, on the other hand, the current variance increases, the control performance index drops accordingly. Ideally, the benchmark is obtained in a defined good state of the control loop and stored when the ConPerMon block is initialized. It does not matter if the CPI temporarily reaches values higher than 100%. A CPI > 100% only means that the variance of the controlled variable is currently somewhat lower than in the reference state. Other alternatives for determining the benchmark will be explained in a separate section.

If you consider that the calculated CPI signal is too strongly affected by noise, you can smooth it using the integrated level pass filter (parameter `CPI_FiltFactor`) with the filter time constant `TimeWindow · CPI_FiltFactor`.

The disadvantage of these stochastic characteristics is that they assume an ergodic (Page 1468) or steady state in the process - at least in a statistical sense. Each step change in the standpoint in a controller is an elementary violation of this requirement and leads temporarily to incorrect statements of the stochastic characteristics, for example variances increasing too much. The basic principle of the combined approach implemented in the ConPerMonblock is to use both stochastic and deterministic characteristics for the control performance and to select the suitable characteristics automatically depending on the operating state.

If a step change in the setpoint is detected in a control loop, the ConPerMon block freezes the CPIvalue and automatically suppresses all messages relating to this. As the user, you can also force the suppression of the messages manually via the `ManSupprCPI = 1` binary input. This setting is useful to avoid false alarms when known disturbances occur, for example at a load change in a Conti process (Page 1468) or a dosing procedure in a Batch process (Page 1467). In such cases the variance of the controlled variable usually rises momentarily. This should not be interpreted as a worsening of the control performance.

Monitoring of deterministic characteristics of the control performance

Assessment of the control performance based on the response to a step change in the setpoint is relatively simple. In the sense of automatic monitoring, the ConPerMon block is capable of determining the essential characteristics of the control performance directly from the signal changes so that when necessary a message or an alarm can be generated automatically by the system.

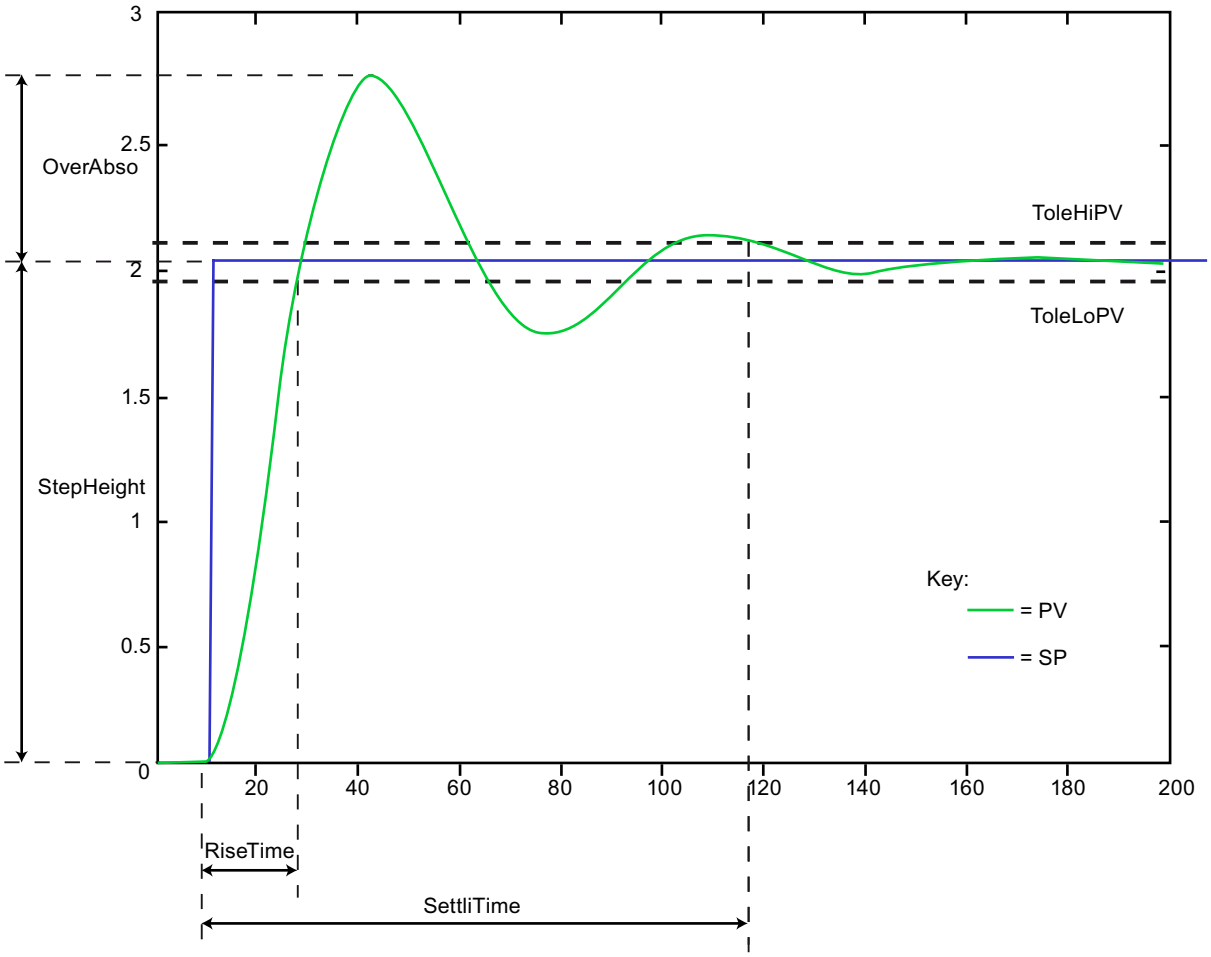
The first thing to look for is always the overshoot if it is present and clearly distinct from the noise level. For a positive step response,

`OverAbso = max(PV) - SP > 0`

is output where is for a negative step response (step response down), and negative values

`OverAbso = min(PV) - SP < 0`

are also output. For normalization, the absolute overshoot is related to the height of the step change in the setpoint and is therefore always positive. The relative overshoot (`Overshoot`) as a percentage is a measure of the damping of the control loop. If this is more than 20 or 30%, the loop gain (gain of the controller multiplied by the gain of the control system) is generally too high either because the controller was badly set from the beginning or because the properties of the control system have changed over the course of time. If overshoot is significantly too high, the control loop is generating weakly damped oscillations in the plant. The block sends a message to this effect if the relative overshoot is above a specified limit.



In every control loop, there is a general correlation between overshoot and phase reserve: The higher the overshoot, the lower the phase reserve. If the response of the closed control loop can be described approximately by a 2nd order transfer function

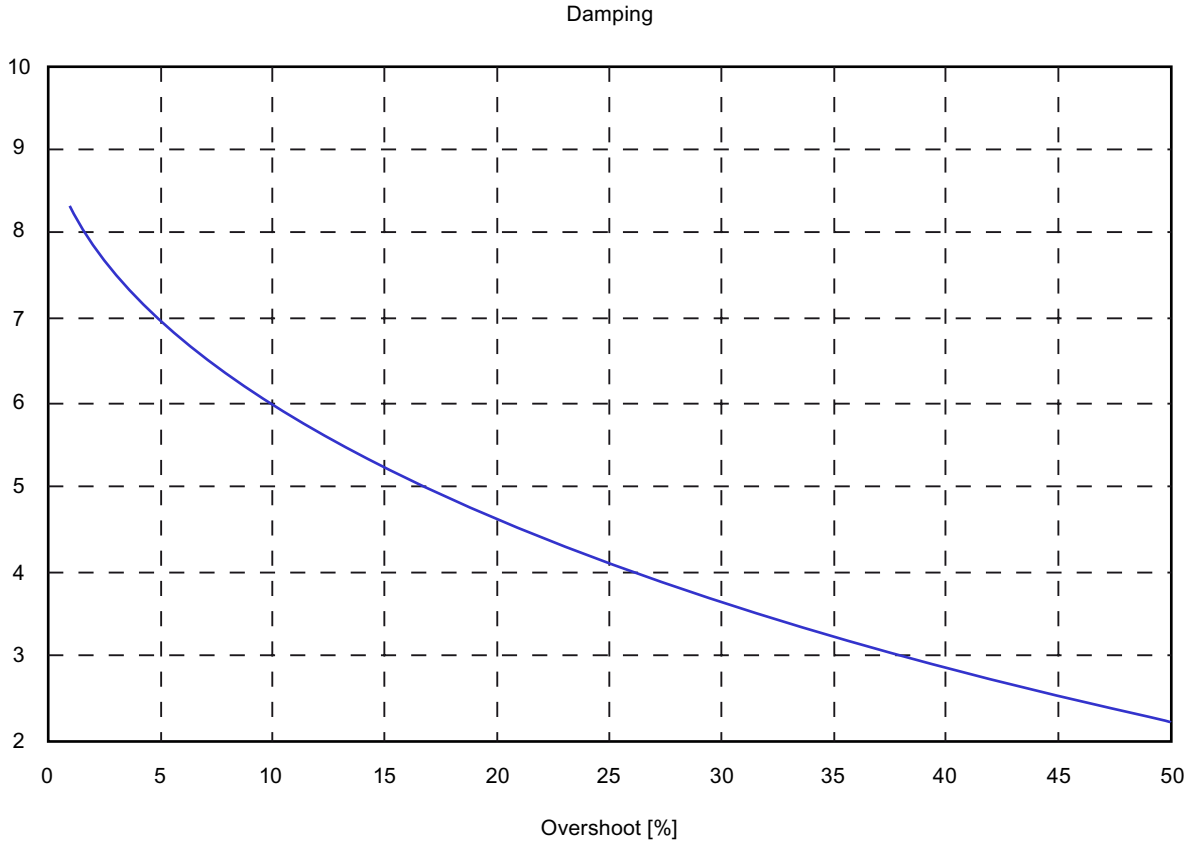
$$g_{cl}(s) = \frac{PV(s)}{SP(s)} = \frac{1}{\frac{1}{\omega_0^2}s^2 + 2\frac{\delta}{\omega_0}s + 1}$$

the following relationships are known:

- If δ ≥ 1, the overshoot is equal to zero and the settling response is asymptotic.
- if δ < 1, overshoot and oscillations occur.

The damping of the closed loop can be determined approximately from the overshoot:

$$\delta = \frac{-\ln\left(\frac{\text{Overshoot}}{100\%}\right)}{\sqrt{\ln^2\left(\frac{\text{Overshoot}}{100\%}\right) + \pi^2}}$$



An optimum controller setting typically aims for overshoot between 5 and 25%, which means damping between 0.7 and 0.4.

If overshoot is too high, it is often helpful to reduce the gain of the controller.

While overshoot primarily serves to check controller gain, there is a further characteristic that provides information on the setting of the I action: If the setting of the reset time is unsuitable, the process value will creep towards the new setpoint following a step change in the setpoint. To allow normalization, the settling time `SettliTime` is related to the rise time `RiseTime` of the step response of the control loop. If the settling ratio, in other words the quotient of the rise time and settling time, is less than approximately 25%, it can generally be assumed that the reset time of the controller is too slow. To determine the rise time and settling time, a 3σ tolerance band is placed around the setpoint and is also displayed in the faceplate of the ConPerMon block. The absolute values of the settling time and rise time can be assessed in terms of the concrete requirements of the process control for a specific application.

During a step change in the setpoint, larger mathematical variances of the controlled variable are bound to occur compared with the steady state so that generating alarms due to the variance limits being exceeded needs to be suppressed until the settling process has neared completion following the step change in the setpoint. The calculated deterministic characteristics are output and the stochastic evaluation is activated again.

Alternatives for determining the benchmark

During planned commissioning of a plant with integrated ConPerMon, following controller optimization, the ConPerMon block is initialized for every control loop and the calculated variance stored as the benchmark for calculating the CPI.

As an alternative, a benchmark can set via the `RefVarExt` input parameter by setting `RefVarExtOn = 1`. There are various ways of obtaining numeric values for the benchmark:

- Take the lowest variance that was ever measured in this control loop since the initialization of the ConPerMon block. This is displayed at the `PV_VarMin` output parameter. This value is only useful when the control loop has been in a stable and desirable operating state for a longer period of time at least once since the initialization of the ConPerMon block.
- Take the variance of the control loop with a theoretical minimum variance controller as can be obtained based on archived data using another supplier's CPM application. This depends only on the process dead time and the disturbance model. This form of CPI is known as the Harris index and represents a lower barrier that can generally not be reached by a PID controller which is why CPI seldom reaches the value 100% even by well tuned controllers. Low CPI values provide the first indication that the controller settings could be improved. You should, however, bear in mind that the minimum variance is only a theoretically achievable value and that the minimum variance controller has characteristics that are not desired in the real application, for example extremely high manipulated variable amplitudes. With minimum variance-based CPI, therefore, it is not worth making every effort to bring this as close as possible to 100%.

Cascade control

In a cascade control, you should only use the ConPerMon block for the primary controller and not for the secondary controller. The ConPerMon block cannot make any useful statements about the control performance of the secondary controller because

- the variance of the process value in the secondary control loop depends directly on the variance of the setpoint that is set as the manipulated variable by the primary controller,
- there are neither operating phases with a constant setpoint nor defined step changes in the setpoint.

Apart from this, from the perspective of process control, the primary control loop is, of course, the one whose control performance should be monitored while the control performance of the secondary loop is of secondary importance. It is nevertheless advisable to set the secondary controller carefully before optimization and monitoring of the primary controller is started because a poor response by the secondary controller cannot be compensated by the primary controller.

For additional information read about the process tag template Cascade control (Page 1442).

Split-range control

The split-range function block contains two separate (static) characteristics for both actuators. Any significant difference between the two actuators in terms of performance (in other words, different process gains for heating and cooling) can be compensated by setting different gradients for the characteristics, so that the controller is presented with a linear process response (regardless of the sign) as far as possible. It this does not work, the control performance will differ slightly in the two areas. The initialization of the ConPerMon block should then be performed in the worse area to avoid error alarms.

For additional information read about the process tag template Split-range control (Page 1438).

PID controller with gain scheduler

The aim of gain scheduling is to achieve consistent control performance over the entire operating range. If this does not work perfectly, the initialization of the ConPerMon block should be performed in an operating point with worse control performance to avoid error alarms. We recommend that you expand the alarm limits somewhat at ConPerMon-Baustein: permit lower `CPIs` and higher overshoot.

For additional information read about the process tag template PID - control with operating-point-oriented parameter control (GainScheduling) (Page 1431).

Override control

For change-over control, different controllers are active depending on the process state; their control performances differ, of course. We recommend using control loop monitoring only for the primary controller, and to suppress it using the `ManSuprCPI` input parameter if limit controlling is activated.

For additional information read about the process tag template Override control (Page 1445).

Feedforward control

The task of feedforward control is to avoid or at least to reduce degradation of the control performance caused by a measurable disturbance variable. Control loop monitoring therefore can basically be used as it is used for simple control loop. When the disturbance variable is quiet for a time and then acts up for a brief period, the resulting fluctuations of the control performance cannot be ruled out. The reason behind it is that feedforward control represents a model-based intervention, and a model is never a perfect reflection of reality.

For additional information read about the process tag template PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 1433).

Smith predictor

The Smith predictor enables higher control performance than a simple PI controller in control loops with dead time. Control loop monitoring therefore can basically be used as it is used for simple control loop. If the dead time changes during ongoing operation, control performance will most likely go down.

For additional information read about the process tag template PID controller with Smith predictor (SmithPredictorControl) (Page 1436).

Ratio control

With ratio controlling, the control loop monitoring should only be used in the primary control loop if the setpoints are to be determined for combined components from the actual value of the primary component. In this case, you can expect continuous setpoint changes in the control loops for the combined components - similar to sequential control loop of a cascade. If the setpoints for combined components are to be determined from the setpoint of the primary component, the lower-level control loops can be monitored as well.

For additional information read about the process tag template Ratio control (Page 1440).

Multivariable controller

The mathematical concept of the ConPerMon block is intended for monovariable control loops. If the variance in a control loop is found to be too high, the block cannot determine whether the actual cause is within the control loop or whether influences are being brought in due to interactions from the outside. If, therefore, you notice strong interactions between various control loops in your plant or even use multivariable controllers, the information provided by the ConPerMon block should be treated with caution.

It can nevertheless make sense to equip a multivariable controller such as the ModPreCon block with control loop monitoring to establish whether the control performance achieved during commissioning of the controller is also retained in runtime. In this case, each controller channel of the multivariable controller has a separate ConPerMon block. Several additional logic functions need to be configured upstream from the `ManSuprCPI` input parameter as shown in the corresponding specimen project Predictive control of a 2x2 multi-variable controlled system (Page 1464):

- If one or more other channels for the multivariable controller is in a non steady state (for example step change in the setpoint) indicated by the `CPI_SupRoot = 1` output parameter, the temporarily increased variance cannot be avoided in this controller channel and should not cause a CPI message.
- If one or more other channels of the multivariable controller have higher variances (poor control performance) indicated by the appropriate output `CPI_WrnAct = 1`, due to the interaction, these variances cause a higher variance in this controller channel as well that cannot be avoided and should not lead to a CPI warning. It is possible to find the actual cause of a disturbance in a multivariable system as follows: The channel that first detects higher variances, set the alarm while subsequent alarms in adjacent channels are suppressed.

Note

In the case of multivariables, the estimated process gains from the monovariable observation are irrelevant. By setting the input parameter `StGainValid = 0` , this status is also displayed in the faceplate as "Uncertain, process related".

If a PID controllers is remotely controlled in program mode (Page 34), it should be treated similar to a secondary controller for a cascade connection in regard to control performance monitoring, i.e. monitoring is usually impractical in this case.

If program mode is the typical operating mode of the controller involved, the corresponding ConPerMon block can be completely removed. If the controller involved is often used in automatic mode, however, monitoring can be temporarily disabled during program mode by connecting the output parameter `AdvCoAct` of the PIDConL block to the input parameter `ManSupprCPI` of the ConPerMon block.

Selecting a unit of measure

The block provides the standard function Selecting a unit of measure (Page 53).

Forming the signal status for blocks

The block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status for the block is formed by the following parameters:

- SP_Mon.ST
- PV_Mon.ST
- MV_Mon.ST
- ER_Mean.ST
- In addition, the ConPerMon block has the following special functions for determining status values:
- If you use a step controller without position feedback there is no manipulated variable that you could interconnect with the input parameterMV_Mon. Unlike most other input parameters, MV_Mon does not have the preset signal status "Uncertain, process related" (16#78). If there is no interconnected value, this status is transferred to the calculated output parameters MV_Mean and StatGain.
- StGainValid: Set this input to 0 if you use a multivariable controller or observe strong interactions between neighboring control loops. This gives the calculated output parameter StatGain the signal status "Uncertain, process related". If known disturbances affect your process, for example dosing procedures in a batch process, you can also set this input temporarily using the recipe control.
- Under normal circumstances the output parameter StatGain accepts the worse signal status of PV_Mon and MV_Mon. Other possible causes of uncertain status for StatGain are:
  - the process is currently very close to the reference operating point, or
  - the process is currently in transition, e.g. step change in the setpoint.
- The signal status of the CPI output parameter is dependent on output parameter CPI_Suppress: If CPI_Suppress=1, the control performance index CPI is uncertain. Apart from this, the CPI can also become uncertain in occasional situations when there are numeric problems in the calculation of the variance. Under normal circumstances the CPI signal status is the same as the PV_Mon signal status.
- The signal status of the OverAbso output parameter is set to invalid when step changes in the setpoint are evaluated whose step change height is too low in relation to the noise level.

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
1	Reaction to the out of service mode (Page 199)
22	Update acknowledgment and error status of the message call (Page 193)

Operator control permissions via parameters `OS_Perm` and `OS1Perm`

The block has the following operator control permissions (Page 45) for the parameter `OS_Perm`:

Bit	Function
0-1	Not allocated
2	1 = Operator can switch to out of service mode
3-27	Not allocated
28	1 = Operator can initialize the block
29	1 = Operator may input a value for the time window, the reference value for the controlled variable and the reference value of the manipulated variable
30-31	Not allocated

The block has the following operator control permissions for the `OS1Perm` parameter:

Bit	Function
0	1 = Operator can alter the limit (overshoot) for high alarm
1	1 = The operator can change the limit (process value) for the high warning
2	Not allocated
3	1 = Operator may change a value for the CPI hysteresis.
4	Not allocated
5	1 = The operator can change the limit (control performance index) `CPI` for the low warning
6	1 = Operator can alter the limit (control performance index `CPI`) for low alarm
7 - 31	Not allocated

Alarm delays with one time value

The block provides the standard function One time value for all limits (Page 79).

This function is used only for monitoring the control performance index `CPI`.

Limit operation and display in the faceplate

This block provides the standard function Limit operation and display in the faceplate (Page 78).

Generating instance-specific messages

The block provides the standard function Generating instance-specific messages (Page 46) without the time stamp function in the I/O devices.

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 122).

See also

Description of ConPerMon (Page 267)

ConPerMon messaging (Page 285)

ConPerMon I/Os (Page 287)

ConPerMon block diagram (Page 293)

ConPerMon error handling (Page 284)

ConPerMon modes (Page 272)

4.1.4 ConPerMon error handling

ConPerMon error handling

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The ErrorNum I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Predefined value when inserting the block, the block is not processed.
0	There is no error.
2	`SampleTime < 0.001`
10	`TimeWindow < 20 · SampleTime`

See also

ConPerMon block diagram (Page 293)

ConPerMon I/Os (Page 287)

ConPerMon messaging (Page 285)

ConPerMon functions (Page 273)

ConPerMon modes (Page 272)

Description of ConPerMon (Page 267)

4.1.5 ConPerMon messaging

Messaging

If the control performance falls below a defined limit a message is generated. This is also the case when a defined limit for overshoot is exceeded when there is a step change in the setpoint.

If the CPI temporarily exceeds the configured warning and alarm limits, it is not necessary to trigger an alarm immediately. The main aim of the control loop monitoring is to signal the need for maintenance or optimization measures in individual control loops. With the alarm delay, you can make sure that an alarm is triggered only after the cause exists for longer than a configured period `AlmDelay`.

The following messages can be generated for this block:

- Process messages
- Instance-specific messages

Process messages

Message instance	Message identifier	Message class	Event
MsgEvID	SIG 1	Alarm - high	$$BlockComment$$ Overshoot - high alarm limit violated
	SIG 2	Warning - high	$$BlockComment$$ Overshoot - high warning limit violated
	SIG 3	Warning - low	$$BlockComment$$ `CPI` - low warning limit violated
	SIG 4	Alarm - low	$$BlockComment$$ `CPI` - low alarm limit violated

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Instance-specific messages

You can use up to three instance-specific messages with this block.

Message instance	Message identifier	Message class	Event
MsgEvID	SIG 5	AS process control message - fault	$$BlockComment$$ External message 1 Status 16#@5%x@
	SIG 6	AS process control message - fault	$$BlockComment$$ External message 2 Status 16#@6%x@
	SIG 7	AS process control message - fault	$$BlockComment$$ External message 3 Status 16#@7%x@

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Associated values for message instance `MsgEvID`

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`
4	`CPI` (control performance index)
	`ExtMsg1.ST`
	`ExtMsg2.ST`
	`ExtMsg3.ST`
8	`ExtVa108`
9	`ExtVa109`
10	Reserved

The associated values 8 and 9 are assigned to the parameters `ExtVa108` ... `ExtVa109` and can be used by yourself. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

See also

Description of ConPerMon (Page 267)

ConPerMon functions (Page 273)

ConPerMon I/Os (Page 287)

ConPerMon block diagram (Page 293)

ConPerMon error handling (Page 284)

ConPerMon modes (Page 272)

4.1.6 ConPerMon I/Os

ConPerMon I/Os

Input parameters

Parameter	Description	Type	Default
AlmDelay	Alarm delay time [s] for monitoring the control performance index CPI 0 = Alarm delay deactivated	REAL	0.0
BatchEn	1 = Enable allocation for batch control	BOOL	0
BatchID	Current batch ID	DWORD	16#00000000
BatchName	Current batch designation	STRING[32]	''
BreakSuppress	1 = Cancel suppression of the control performance alarm at the end of the step response	BOOL	0
CPI_AlmHys	Alarm hysteresis of the control performance index [%]	REAL	5.0
CPI_AL_En	1 = Activate alarm (low) for monitoring of control performance	BOOL	1
CPI_AL_Lim	Low alarm limit for control performance [%]	REAL	30.0
CPI_FiltFactor	Low-pass filter for CPI, filter time constant = TimeWindow ·CPI_FiltFactor	REAL	10.0
CPI_WL_En	1 = Activate warning (low) for monitoring of control performance	BOOL	1
CPI_WL_Lim	Low warning limit for control performance [%]	REAL	50.0
CPI_AL_MsgEn	1 = Activate alarm message for: • Low limit for control performance	BOOL	0
CPI_WL_MsgEn	1 = Activate warning message for: • Low limit for control performance	BOOL	0
EN	1 = Called block will be processed	BOOL	1
ExtMsg1	Binary input for freely selectable message 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg2	Binary input for freely selectable message 2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg3	Binary input for freely selectable message 3	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
ExtVa108	Associated value 8 for messages (MsgEvID)	ANY	
ExtVa109	Associated value 9 for messages (MsgEvID)	ANY	
Feature	I/O for additional functions (Page 273)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
InitRefVar	1 = Initialization of the block. The benchmark of the controlled variable and the reference values of the controlled variable and manipulated variable are measured in the steady state.	BOOL	0
ManSupprCPI	1 = Manual suppression of the CPI calculation and message, e.g. during known disturbances	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MsgEvId	Message number (assigned automatically)	DWORD	16#FFFFFFFF
MsgLock	1 = Suppress process messages	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_Mon	Manipulated variable of the controller for monitoring	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#80
MV_Ref	Reference value of the manipulated variable	REAL	0.0
MV_Unit	Unit of measure for manipulated variable	INT	1342
Occupied	1 = Occupied by batch control	BOOL	0
OnOp	1 = "On" mode via operator	BOOL	1
OosLi	1 = Edge transition (0-1) = Out of service, via interconnection or SFC	STRUCT • Value: BOOL • ST: BYTE	- 0 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OS_Perm	I/O for operator control permissions (Page 273)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
OS1Perm	I/O for operator control permissions (Page 273)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
OvsAH_En	1 = Activate alarm (high) for overshoot	BOOL	1

Parameter	Description	Type	Default
OvsAH_Lim	Overshoot alarm limit [%]	REAL	25.0
OvsAH_MsgEn	1 = Enable alarm message for overshoot	BOOL	0
OvsWH_En	1 = Activate warning (high) for overshoot	BOOL	1
OvsWH_Lim	Overshoot warning limit [%]	REAL	20.0
OvsWH_MsgEn	1 = Enable warning message for overshoot	BOOL	0
PID_AutAct	1 = Controller is in auto mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_Mon	Controlled variable for monitoring	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_Ref	Reference value for controlled variable	REAL	0.0
PV_Unit	Unit of measure for process value	INT	1001
RefVarExt	Reference value for PV_Variance in control loop "good" status	REAL	0.0
ReVaExOn	1 = Use the external reference value of RefVarExt	BOOL	0
RefVariance	Variance of controlled variable in control loop "good" status	REAL	100.0
RunUpCyc	Number of start cycles in which messages are suppressed	INT	32000
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SelFp1	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
SP_Mon	Setpoint of the corresponding controller for monitoring	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
StepNo	Batch step number	DWORD	16#00000000
StGainValid	0 = Output parameter StatGain systematically invalid, e.g. for multi-variable processes	BOOL	1
TimeWindow	Width of the sliding time window [s] for statistical evaluations	REAL	120.0
UserStatus	Freely assignable bits for use in faceplate	BYTE	16#00

Output parameters

Parameter	Description	Type	Default
CPI	Control performanceindex	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
CPI_AL_Act	1 = Control performance alarm is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
CPI_Suppress	1 = Message for the control performance index is suppressed; retain last valid CPI value	BOOL	1
CPI_SuRoot	1 = CPI message suppression is locally active in this control loop	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
CPI_WL_Act	1 = Control performance warning is active for CPI	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ENO	1 = Block algorithm completed without errors	BOOL	0
ER_Mean	Mean value of the control deviation in the time window [PV_Unit]	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
ErrorNum	Output of current error number. For error numbers that can be output by this block, see ConPerMon error handling (Page 284)	INT	-1
MsgAckn	ALARM_8P: Output ACK_STATE; message acknowledgment state	WORD	0
MsgErr	1 = Message processing error occurred	BOOL	0
MsgStatus	ALARM_8P: STATUS output Error information of ALARM_8P	WORD	0
MV_Mean	Mean value of the manipulated variable in the time window [MV_Unit]	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
OnAct	1 = "On" mode enabled	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
OosAct	1 = Block is "Out of service"	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF

Parameter	Description	Type	Default
OS1PermOut	Display of OS1Perm	DWORD	16#FFFFFFFF
OS1PermLog	Display of OS1Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
OverAbso	Absolute overshoot of the step response [PV_Unit]	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Overshoot	Relative overshoot of the step response with respect to the step change height [%]	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
OvsAH_Act	1 = Alarm due to overshoot is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OvsWH_Act	1 = Warning due to overshoot is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_Mean	Mean value of the controlled variable in the time window [PV_Unit]	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_StdDev	Standard deviation of the controlled variable	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_ToleHi	Limit (high) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected	REAL	0.0
PV_ToleLo	Limit (low) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected	REAL	0.0
PV_Variance	Variance of the controlled variable	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_VarMin	Minimum observed value of the process variance (slave pointer)	REAL	1.000000e+004
RefStdDev	Standard deviation of controlled variable in control loop "good" status	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
RiseTime	Rise time of the step response [s]	REAL	0.0
SettlRatio	Ratio = Rise time / settling time *100%	REAL	0.0
SettliTime	Settling time of the step response [s]	REAL	0.0
StatGain	Steady-state process gain [PV_Unit / MV_Unit]	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word 1 (Page 267)	DWORD	16#00000000

Parameter	Description	Type	Default
Status2	Status word 2 (Page 267)	DWORD	16#00000000
StepPhase	Phase of the step response: 0 = Waiting 1 = Rising 2 = Overshoot 3 = Settled	INT	0

See also

ConPerMon messaging (Page 285)

ConPerMon block diagram (Page 293)

ConPerMon modes (Page 272)

4.1.7 ConPerMon block diagram

ConPerMon block diagram

A block diagram is not provided for this block.

See also

ConPerMon I/Os (Page 287)
ConPerMon messaging (Page 285)
ConPerMon error handling (Page 284)
ConPerMon functions (Page 273)
ConPerMon modes (Page 272)
Description of ConPerMon (Page 267)

4.1.8 Operator control and monitoring

4.1.8.1 ConPerMon views

Views of the ConPerMon block

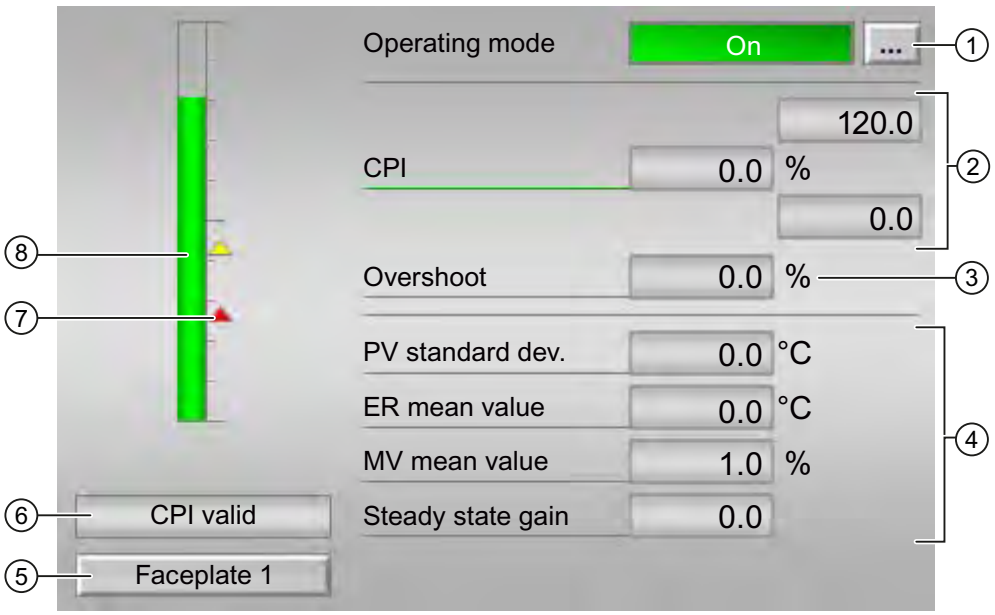
The block ConPerMon provides the following views:

- ConPerMon standard view (Page 294)
- Message view (Page 169)
- ConPerMon limit value view (Page 296)
- Trend view (Page 172)
- ConPerMon parameter view (Page 298)
- ConPerMon preview (Page 300)
- Memo view (Page 171)
- Batch view (Page 170)
- ConPerMon's setpoint view (Page 301)
- Block symbol for ConPerMon (Page 303)

Refer to the sections Structure of the faceplate (Page 123) and Block icon structure (Page 174) for general information on the faceplate and block icon.

4.1.8.2 ConPerMon standard view

ConPerMon standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 27)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Display area for control performance

This area shows the current control performance index.

(3) Display area for the overshoot

This area shows you the relative overshoot based on a step change [%].

(4) Display area for the static evaluation of the current time window (`TimeWindow`)

This area shows you the statistical evaluation of the current time window. The following values are evaluated:

- PV standard dev.: Standard deviation of the controlled variable
- ER mean value: Mean value of the control error
- MV mean value: Mean value of the manipulated variable
- Steady state gain: Steady-state process gain

(5) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

See also the Calling further faceplates (Page 43) section for more on this.

(6) Display for CPI valid / CPI invalid

This area shows you if the control performance index is valid or invalid:

- CPI valid: Control performance is valid
- CPI invalid: Control performance is invalid

You set the limits for the control performance index in the limits views, depending on the configuration in the engineering system (ES).

(7) Limit display

These colored triangles show you the configured limits in the respective bar graph.

(8) Bar graph for control performance index

This area shows you the current CPI control performance index in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

See also

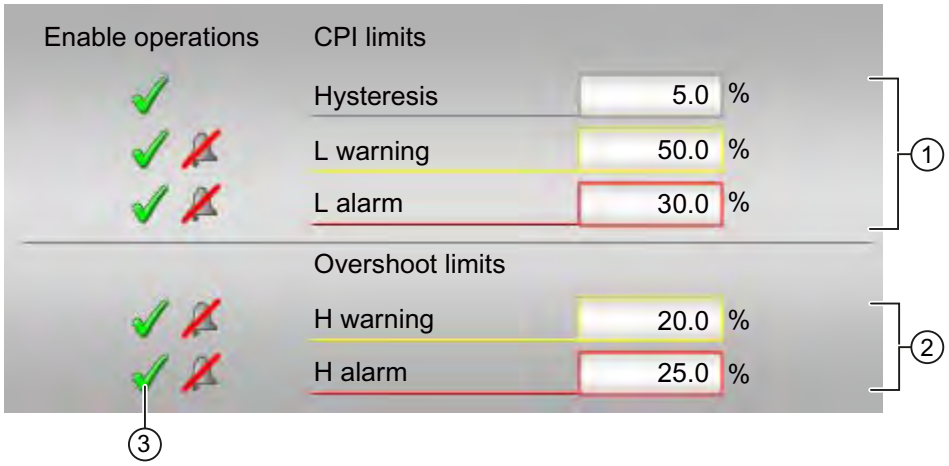
ConPerMon limit value view (Page 296)

ConPerMon parameter view (Page 298)

ConPerMon's setpoint view (Page 301)

4.1.8.3 ConPerMon limit value view

ConPerMon limit value view



(1) CPI limits

In this area, you can enter the limits for the CPI control performance index. Refer to the Changing values (Page 129) section for more on this.

You can change the following limits:

- Hysteresis
- L warning: Warning low
- L alarm: Alarm low

(2) Overshoot limits

In this area, you can enter the limits for the overshoot. Refer to the Changing values (Page 129) section for more on this.

You can change the following limits:

- H warning: Warning high
- H alarm: Alarm high

(3) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions

See also

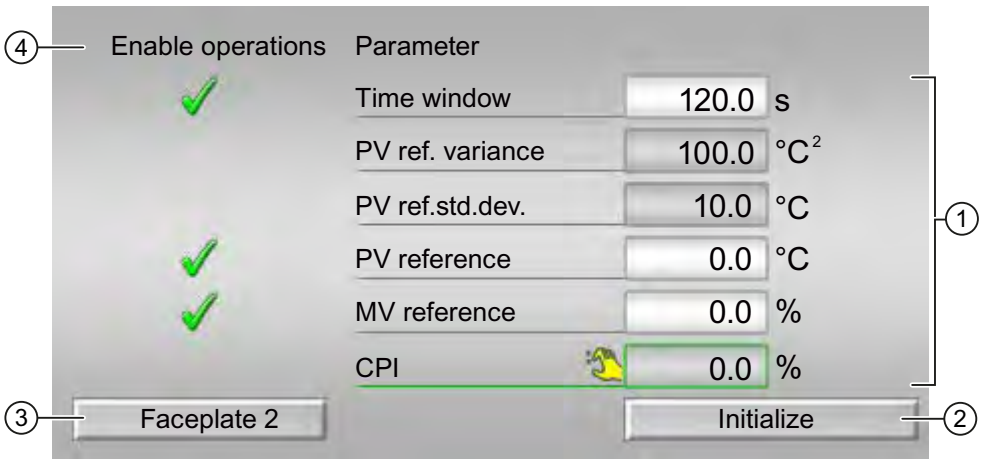
ConPerMon standard view (Page 294)

ConPerMon parameter view (Page 298)

ConPerMon's setpoint view (Page 301)

4.1.8.4 ConPerMon parameter view

ConPerMon parameter view



(1) Parameters

In this area, you change parameters and therefore influence the controller. Refer to the Changing values (Page 129) section for more on this.

You can influence the following parameters:

- Time window: Set the time window here, in which the statical evaluation for the following values is to be performed:
  - Standard deviation of the controlled variable
  - Mean value of the control error
  - Mean value of the manipulated variable
  - Steady-state process gain
- PV reference: Reference value for controlled variable
- MV reference: Reference value of the manipulated variable

(2) Initialize button

Clicking this button initializes the block. The benchmark of the controlled variable and the reference values of the controlled variable and manipulated variable are measured in the steady state.

(3) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

Refer also to the Calling further faceplates (Page 43) section for more on this.

(4) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions

See also

ConPerMon standard view (Page 294)

ConPerMon limit value view (Page 296)

ConPerMon's setpoint view (Page 301)

4.1.8.5 ConPerMon preview

ConPerMon preview

(1) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions

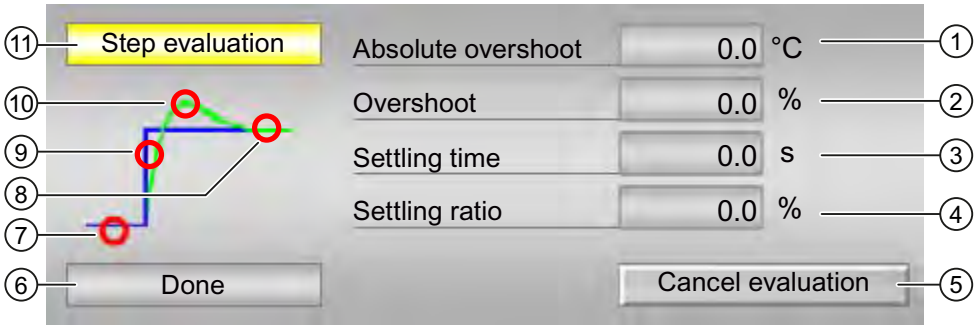
(2) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

Additional information is available in the section Calling further faceplates (Page 43).

4.1.8.6 ConPerMon's setpoint view

Setpoint view of ConPerMon



(1) Absolute overshoot

The absolute overshoot is given in the physical unit of the actual value

(2) Overshoot

Output of the relative overshoot base on a step change.

(3) Settling time

Settling time of the step response in seconds

(4) Settling time ratio

The settling time ratio is formed from the ramp time by the settling time.

(5) Cancel evaluation button

You can use the button to show the evaluation of the step response.

(6) and (7) - (10) and : Status of the step response

The following states are shown here:

- (6) Textual display of the states
- (7) Ready (stationary state)
- (8) Rising phase (from the initial state to the first time setpoint is reached)
- (9) Overshoot
- (10) Steady state, i.e. the actual value is within the tolerance range of the setpoint

(11) Display: Evaluation of the step response in progress:

- Step evaluation
- Constant PV

See also

ConPerMon standard view (Page 294)

ConPerMon limit value view (Page 296)

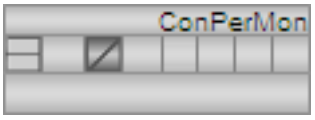
ConPerMon parameter view (Page 298)

4.1.8.7 Block symbol for ConPerMon

Properties of the ConPerMon block icon

A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits
- Operating modes
- Signal status, maintenance release
- Memo display
- Process value (black, with and without decimal places)

Icons	Selection of the block icon in CFC	Special features
	1	Block icon in the full display
	5	
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181)

4.2 FmCont - Interface to module FM 355

4.2.1 Description of FmCont

Object name (type + number) and family

Type + number: FB 1818

Family: Control

Area of application for FmCont

The block is used for the following applications:

- Fixed setpoint control
- Cascade control
- Ratio control
- Split-range control

How it works

Block FmCont is used to interface the FM 355 controller modules.

FmCont can be used for the C (continuous controllers) and S (step and pulse controllers) module types. It contains the algorithms of the setpoint ramp, the setpoint rise limitation, and the limit monitoring of the process value, the control error, and the position feedback. Limit monitoring is not used on the module. The control function itself (e.g. PID algorithm) is processed on the module.

You can use the FmCont block to monitor all relevant process values and to change all relevant controller parameters.

Application examples of the FM 355 and detailed descriptions of the associated input and output parameters can be found in the manual of the FM 355 controller module.

Process values such as temperatures, levels and flows can be controlled. However, pressure processes which are not excessively fast are also possible.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100). Set the input parameter `LogAddr` to the module address from HW Config and the input parameter `Channel` to the desired controller channel (1..4).

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The parameter `CoordNo` is set
- The parameter `FM355` is set in accordance with the module type C/S
- The in/out parameter `EnCoord` is interconnected with the output `EN_CO_x` of the FM_CO block of the basic library (x = number of the rack)
- The output parameter `EnCoNum` is interconnected with the input `ENCOx_yy` of the `FmCont` block (x = number of the rack, yy = coordination number).

For the control loop monitoring to work as planned in the trend view of the controller faceplates, the

`S7_xarchive:='Value,shortterm;'`

attributes in the process tag types for control loops at the controller function block must be set for the following tags:

- Input parameters:
  - `CPI_In`
- Output parameters
  - `MV`
  - `MV_HiAct`
  - `MV_LoAct`
  - `AutAct`
  - `SP`
  - `PV_Out`
  - `PV_ToleHi`
  - `PV_ToleLo`

Startup characteristics

Use the Setting the startup characteristics (Page 187) feature to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

Status word allocation for `Status1` parameter

For a description of the individual parameters see the section I/Os of FmCont (Page 324).

Status bit	Parameter
0	`Occupied`
1	`BatchEn`
2	`SimOn`
3	`OosAct.Value`
4	`OosLi.Value`
5	`AutAct.Value`
6	Not allocated
7	`ManAct.Value`
8	`SP_ExtAct.Value`
9	`MV_SafeOn.Value`
10	`MV_TrkOn.Value`
11	`MV.Value > ManLoLim` for continuous or pulse controller `NOT FbkClosed.Value` for step controller with/without position feedback
12	`Open.Value`
13	`Close.Value`
14	`Stop.Value`
15	`FbkOpened.Value`
16	`FbkClosed.Value`
17 - 18	Not allocated
19	`AdvCoAct`
20	1 = Input parameter `Rbk` is not interconnected (RbkOut.ST = 16#FF)
21	`NegGain`
22	`MV_TrkAct.Value`
23	`FuzCon`
24	`OptimOcc OR FuzOptAct`
25 - 27	Not allocated
28	1 = Analog controller (`FM355 = 1`)
29	1 = Pulse controller (`FM355 = 0 ANDStepCon = 0`)
30	1 = Step controller with position feedback (`FM355 = 0` AND `StepCon = 1` AND `WithRbk = 1`)
31	1= Step controller without position feedback (`FM355 = 0` AND `StepCon = 1` AND `WithRbk = 0`)

Status word allocation for `Status2` parameter

Status bit	Parameter
0	MsgLock
1	PV_AH_Act.Value
2	PV_WH_Act.Value
3	PV_TH_Act.Value
4	PV_TL_Act.Value
5	PV_WL_Act.Value
6	PV_AL_Act.Value
7	PV_AH_En
8	PV_WH_En
9	PV_TH_En
10	PV_TL_En
11	PV_WL_En
12	PV_AL_En
13	PV_AH_MsgEn
14	PV_WH_MsgEn
15	PV_TH_MsgEn
16	PV_TL_MsgEn
17	PV_WL_MsgEn
18	PV_AL_MsgEn
19	ER_AH_Act.Value
20	ER_AL_Act.Value
21	ER_AH_En
22	ER_AL_En
23	ER_AH_MsgEn
24	ER_AL_MsgEn
25	RbkWH_Act.Value
26	RbkWL_Act.Value
27	RbkWH_En
28	RbkWL_En
29	RbkWH_MsgEn
30	RbkWL_MsgEn
31	MS_RelOp

See also

FmCont messaging (Page 321)
FmCont block diagram (Page 338)
FmCont modes (Page 308)
FmCont error handling (Page 319)
FmCont functions (Page 309)

4.2.2 FmCont modes

FmCont operating modes

The block can be operated using the following modes:

- Automatic mode (Page 29)
- Manual mode (Page 29)
- Program mode for closed-loop controllers (Page 34)
- Out of service (Page 27)

The next section provides additional block-specific information relating to the general descriptions.

Automatic mode

You can find general information on "Automatic mode", switching modes and bumpless switchover in the Manual and Automatic mode for control blocks (Page 29) section.

Manual mode

You can find general information on "Manual mode", switching modes and bumpless switchover in the Manual and Automatic mode for control blocks (Page 29) section.

Program mode for controllers

General information on "Program mode for controllers" is available in the section Program mode for closed-loop controllers (Page 34).

Out of service

You will find general information about the "Out of service" mode in the Out of service (Page 27) section.

See also

FmCont block diagram (Page 338)

FmCont I/Os (Page 324)

Description of FmCont (Page 304)

FmCont functions (Page 309)

FmCont error handling (Page 319)

FmCont messaging (Page 321)

4.2.3 FmCont functions

Functions of FmCont

The functions for this block are listed below.

Module types

FmCont can be used for the C (continuous controllers) and S (step controllers with and without position feedback and pulse controllers) module types. You can use the following parameters to identify which module type and controller type has been set:

FM355	StepCon	WithRbk	Module type, controller type
1 or C	-	-	FM 355 C: Continuous controller
0 or S	1	1	FM 355 S: Step controller with position feedback
0 or S	1	0	FM 355 S: Step controller without position feedback
0 or S	0	-	FM 355 S: Pulse controller

You must set input parameter `StepCon` if you want to set the step controller with/without position feedback as the controller type.

Generating manipulated variables for continuous controllers, step controllers with position feedback, or pulse controllers

The manipulated variable MV and the actuating signals Open, Close and Stop are generated as follows:

MV_SafeOn	MV_FmTrkOn	ManAct	MV_TrkOn	AdvCoAct AND NOT AdvCoMod SP	MV =	Limitation of the manipulated variable	State	Open, Close, Stop
1	-	-	-	-	MV_Safe	MV_HiLim MV_LoLim	Tracking to safety value	**Cont. controller:** Open, Close, Stop = False **Step controller with position feedback:** Depending on Rbk and MV, the output signals Open, Close and Stop are generated using the algorithm of a positioner. **Pulse controller:** Depending on MV, the output signals Open and Close are generated using the algorithm of a pulse controller (Stop = 0).
0	1	-	-	-	Prepared FM analog input	MV_HiLim MV_LoLim	Tracking to an FM analog input	
0	0	1	-	-	Man	ManHiLim ManLoLim	Manual mode, set by the operator	
0	0	0	1	-	MV_Trk	MV_HiLim MV_LoLim	Tracking to block input MV_Trk	
0	0	0	0	1	AdvCoMV	MV_HiLim MV_LoLim	Higher-level program mode	
0	0	0	0	0	P_Part + I_Part + D_Part + FFwd	MV_HiLim MV_LoLim	Automatic mode (PID algorithm)	

Generating manipulated signals for step controllers without position feedback (WithRbk = 0)

The manipulated variable signals `Open`, `Close` and `Stop` can be generated as follows:

ManAct	Open, Close, Stop	State
1	The output signals are generated using input signals `OpenOp/Li`, `CloseOp/Li` or `StopOp/Li`.	Manual mode, set by the operator
0	The output signals are generated using PID output parameters `P_Part`, `I_Part`, `D_Part` and `FFwd`.	Automatic mode (PID algorithm)

Tracking and limiting a manipulated variable (cont. controller, step controller with position feedback and pulse controller)

The block provides the standard function Tracking and limiting a manipulated value (Page 111).

Safe position

The controller modules have their own mechanism for feedforwarding a safety value (see Temperature Controller FM 355-2 manual or Controller Module FM 355 manual)

"Actuator active" information

For continuous and pulse controllers: If the manipulated variable `MV` is greater than the minimum manual limit `ManLoLim`, this is recognized as actuator active.

For step controllers: If the parameter `FbkClosed` = 0, this is known as "Actuator active".

This status can be used to display a custom icon in the process image, for example, and is saved in the status word (see Status word section in Description of FmCont (Page 304)).

Limit monitoring of position feedback (cont. controller, step controller with position feedback and pulse controller)

The block provides the standard function Limit monitoring of the feedback (Page 76).

External/internal setpoint specification

The block provides the standard function Setpoint input - internal and external (Page 96).

Setpoint limiting for external setpoints

The block provides the standard function Setpoint limiting for external setpoints (Page 110).

Gradient limit of the setpoint

The block provides the standard function Ramp limiting of the setpoint (Page 108).

Using setpoint ramp

The block provides the standard function Using a setpoint ramp (Page 98).

Tracking setpoint in manual mode

The block provides the standard function Setpoint tracking the process variable in manual mode (Page 110).

Simulating signals

The block provides the standard function Simulating signals (Page 93).

You can simulate the following values:

- Process value (`SimPV`)
- Position feedback (`SimRbk`)

Note

The simulated process value `SimPV` only affects alarm processing and not the PID algorithm in the controller module.

Limit monitoring of the process value

The block provides the standard function Limit monitoring of the process value (Page 70).

Control error generation and dead band

The block provides the standard function Control error generation and deadband (Page 109).

Limit monitoring of control error

The block provides the standard function Monitoring the error signal limits (Page 77).

Inverting control direction

The block provides the standard function Inverting the control sense (Page 108).

Physical standardization of setpoint, manipulated variable and process value

Controller gain `Gain` is entered either using a physical variable or as standardized value.

- `Gain` as a physical variable:

  The standardized variables retain their default values:

  – `NormPV.High` = 100 and `NormPV.Low` = 0

  – `NormMV.High` = 100 and `NormMV.Low` = 0

  For step controllers with/without position feedback and pulse controllers, the values of `NormMV.High` and `NormMV.Low` are not taken into account. The algorithm uses default values 0 and 100 for internal calculations.

  The effective gain is: `GainEff` = `Gain`

- Entering a standardized gain (dimensionless):

  Change the standardized variables to the actual range of the process values and manipulated variables.

  – Internal and external setpoints; the process value and corresponding parameters are entered according to the physical measuring range of the process value.

  Continuous controller, pulse controller:

  – The manual value, the tracking value of the manipulated variable, feedforward control and the corresponding parameters are set according to the physical measuring range of the manipulated variable.

  Step controller with position feedback:

  – The manual value, the tracking value of the manipulated variable, feedforward control and the corresponding parameters are entered as a percentage between 0 ... 100.

  Step controller without position feedback:

  – No physical measuring range available.

  The effective gain is:

  – Step controller with/without position feedback:

    `GainEff` = 100,0 / (`NormPV.High` - `NormPV.Low`) · `Gain`

  – Continuous controller, pulse controller:

    `GainEff = (NormMV.High - NormMV.Low)/(NormPV.High - NormPV.Low) · Gain`

Selecting a unit of measure

The block provides the standard function Selecting a unit of measure (Page 53).

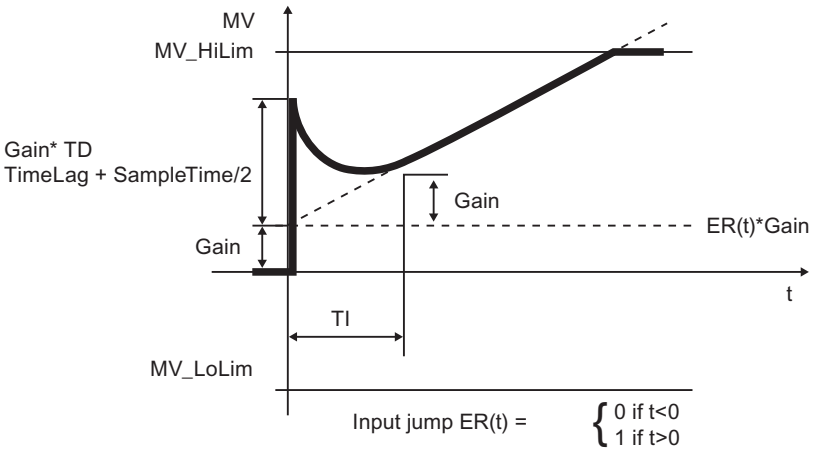
PID algorithm

The manipulated variable is generated in automatic mode according to the following algorithm:

```
MV = Gain · (1 + 1/TI · s) + TD/( 1 + TD/DiffGain · s) · ER
```

Where: s = Complex number

The following step response occurs:



Note

This formula describes a standard application where the P, I and D actions are activated and the P and D actions are not in the feedback circuit (`PropSel = 1, TI ≠ 0D_InSel = 0`and `P_FbkSel = 0`).

The D action delay is derived from `TD/DiffGain`.

- The P action can be shut down by `PropSel = 0`.
- The I action can be shut down by `TI = 0`.
- The D action can be shut down by `TD = 0`.

Structure segmentation at controllers

The PID controller algorithm of `FM355` features structure segmentation. It is activated via the `P_FbkSel` and `D_InSel` parameters. The precise functionality is described in the `FM355` manual.

Anti-windup

The PID control algorithm of `FM355` has an anti-windup function. The I action is frozen or tracked after the manipulated variable has reached its limits (`MV_HiLim` or `MV_LoLim`).

Activating and limiting disturbance variables

The block provides a function for activating the disturbance variable. The precise functionality is described in the FM355 manual.

Forming the signal status for blocks

The block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status ST_Worst for the block is formed from the following parameters:

- Rbk.ST
- FFwdOut.ST
- ER.ST
- MV.ST

Configurable reactions using the Feature parameter

You can find an overview of all reactions provided by the Feature parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)
1	Reaction to the out of service mode (Page 199)
2	Resetting the commands for changing the mode (Page 193)
4	Setting switch or button mode (Page 195)
22	Update acknowledgment and error status of the message call (Page 193)

Operator control permissions via parameters OS_Perm and OS1Perm

The block has the following operator control permissions (Page 45) for the parameter OS_Perm:

Bit	Function
0	1 = Operator can switch into automatic mode AutModOp
1	1 = Operator can switch into manual mode ManModOp
2	1 = Operator can switch into "out of service" mode OosOp
3	1 = Operator can switch into program mode AdvCoEn
4	1 = Operator can switch the setpoint to external SP_ExtOp
5	1 = Operator can switch the setpoint to internal SP_IntOp
6	1 = Operator can change the internal setpoint SP_Int
7	Continuous controllers, pulse controllers or step controllers with position feedback: 1 = Operator can change the manual parameter Man Step controller without position feedback: 1 = Operator can change the manual operation signals OpenOp, StopOp, CloseOp
8	1 = Operator can change maximum usage limit of the setpoint SP_InHiLim
9	1 = Operator can change minimum usage limit of the setpoint SP_InLoLim
10	1 = Operator can change maximum usage limit of the manipulated variable ManHiLim
11	1 = Operator can change minimum usage limit of the manipulated variable ManLiLim
12	1 = Operator can use the setpoint's gradient limitation function SP_RateOn
13	1 = Operator can change the setpoint's high limit for the ramp SP_UpRaLim
14	1 = Operator can change the setpoint's low limit for the ramp SP_DnRaLim
15	1 = Operator can change between the time value or the value for the ramp SP_RmpModTime
16	1 = Operator can change the ramp time SP_RmpTime
17	1 = Operator can change the target setpoint SP_RmpTarget for the setpoint ramp
18	1 = Operator can activate the setpoint ramp function SP_RmpOn
19	Not allocated
20	1 = Operator can activate the track setpoint in manual mode function SP_TrkPV
21	1 = Operator can activate the bumpless changeover from external to internal function SP_TrkExt
22	1 = Operator can change the gain parameter Gain
23	1 = Operator can change the integral action time parameter TI
24	1 = Operator can change the derivative action time parameter TD
25	1 = Operator can change the derivative gain parameter DiffGain
26	1 = Operator can change the dead band parameter DeadBand
27	Not allocated
28	1 = Operator can change the integral action time parameter MotorTime
29	1 = Operator can change the integral action time parameter PulseTime
30	1 = Operator can change the integral action time parameter BreakTime
31	Not allocated

The block has the following operator control permissions for the OS1Perm parameter:

Bit	Function
0	1 = Operator can change the limit (process value) for the high alarm PV_AH_Lim
1	1 = Operator can change the limit (process value) for the high warning PV_WH_Lim
2	1 = Operator can change the limit (process value) for the high tolerance PV_TH_Lim
3	1 = Operator can change the hysteresis (process value) PV_Hyst
4	1 = Operator can change the limit (process value) for the low tolerance PV_TL_Lim
5	1 = Operator can change the limit (process value) for the low warning PV_WL_Lim
6	1 = Operator can change the limit (process value) for the low alarm PV_AL_Lim
7	1 = Operator can change the limit (control error) for the high alarm ER_AH_Lim
8	1 = Operator can change the hysteresis (control error) ER_Hyst
9	1 = Operator can change the limit (control error) for the low alarm ER_AL_Lim
10	1 = Operator can change the limit (position feedback) for the high warning RbkWH_Lim
11	1 = Operator can change the hysteresis (position feedback) RbkHyst
12	1 = Operator can change the limit (position feedback) for the low warning RbkWL_Lim
13 - 15	Not allocated
16	1 = Operator can activate the simulation function SimOn
17	1 = Operator can activate the maintenance release function MS_RelOp
18 - 31	Not allocated

Maintenance release

The block provides the standard function Maintenance release (Page 47).

Generating instance-specific messages

The block provides the standard function Generating instance-specific messages (Page 46) without the time stamp function in the I/O devices.

Specifying the display area for process and setpoint values as well as operations

This block provides the standard function Display and operator input area for process values and setpoints (Page 42).

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 122).

See also

FmCont messaging (Page 321)
FmCont I/Os (Page 324)
FmCont block diagram (Page 338)
FmCont modes (Page 308)
FmCont error handling (Page 319)
Two time values per limit pair (Page 80)
Safety value of the manipulated variable effective at startup (Page 196)
Out of service (Page 27)
Safe position for motors, valves and controllers (Page 120)

4.2.4 FmCont error handling

FmCont error handling

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` output parameter can be used to output various error numbers. An overview of all error numbers is available in the section "Functions of blocks > Error handling (Page 117)".

Error numbers output by this block:

Error number	Meaning of the error number
-1	Default value when implementing the block; block will not be processed.
0	There is no error.
30	The value of `PV` can no longer be displayed in the REAL number field.
31	The value of `SP_Ext` can no longer be displayed in the REAL number field.
33	The value of `MV_Trk` can no longer be displayed in the REAL number field.
35	The value of `Rbk` can no longer be displayed in the REAL number field.
36	The value of `MV` can no longer be displayed in the REAL number field.
50	The controller cannot be switched in program mode, because program mode with setpoint specification (AdvCoModSP=false) is not possible with step controllers without position feedback (WithRbk=false).
60	`|Tl| < SampleTime/2`
61	`|TD| < SampleTime`
62	`DiffGain < 1 oder DiffGain > 10`
63	`TD/DiffGain < SampleTime/2`
64	`PropFacSP` < 0 or `PropFacSP` > 1
66	`NormPV_High` = `NormPV_Low`
67	`MotorTime` < `SampleTime`
68	`PulseTime` < `SampleTime`
69	`BreakTime` < `SampleTime`
70	`Channel < 1` or `Channel > 4`
71	(`D_InSel < 0` or `D_InSel > 4`) and `D_InSel` ≠ 17

See also

FmCont block diagram (Page 338)
FmCont I/Os (Page 324)
Description of FmCont (Page 304)
FmCont modes (Page 308)
FmCont functions (Page 309)
FmCont messaging (Page 321)
Setting switch or button mode (Page 195)

4.2.5 FmCont messaging

Messaging

The following messages can be generated for this block:

- Control system fault
- Process messages
- Instance-specific messages

Control system fault

The following control system error messages can be output:

Message instance	Message identifier	Message class	Event
MsgEvId2	SIG 6	AS process control message - fault	$$BlockComment$$ External error has occurred

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter CSF. If this signal changes to CSF = 1, a control system fault is triggered (MsgEvId2, SIG 6).

Process messages

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 1	Alarm - high	$$BlockComment$$ PV - high alarm limit violated
	SIG 2	Warning - high	$$BlockComment$$ PV - high warning limit violated
	SIG 3	Tolerance - high	$$BlockComment$$ PV - high tolerance limit violated
	SIG 4	Tolerance - low	$$BlockComment$$ PV - low tolerance limit violated
	SIG 5	Warning - low	$$BlockComment$$ PV - low warning limit violated
	SIG 6	Alarm - low	$$BlockComment$$ PV - low alarm limit violated
	SIG 7	Alarm - high	$$BlockComment$$ ER - high alarm limit violated
	SIG 8	Alarm - low	$$BlockComment$$ ER - low alarm limit violated

Message instance	Message identifier	Message class	Event
MsgEvId2	SIG 7	Warning - high	$$BlockComment$$ Rbk - high warning limit violated
	SIG 8	Warning - low	$$BlockComment$$ Rbk - low warning limit violated

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Instance-specific messages

You can use up to four instance-specific messages with this block.

Message instance	Message identifier	Message class	Event
MsgEvId2	SIG 1	AS process control message - fault	$$BlockComment$$ External message 1 Status 16#@5%x@
	SIG 2	AS process control message - fault	$$BlockComment$$ External message 2 Status 16#@6%x@
	SIG 3	AS process control message - fault	$$BlockComment$$ External message 3 Status 16#@7%x@
	SIG 4	AS process control message - fault	$$BlockComment$$ External message 4 Status 16#@8%x@

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Associated values for message instance MsgEvId1

Associated value	Block parameters
1	BatchName
2	StepNo
3	BatchID
4	Process value PV_Out
5	Control error ER
6	ExtVa106
7	ExtVa107
8	Not allocated
9	Not allocated
10	Not allocated

The associated values 6 and 7 are assigned to the parameters ExtVa106 ... ExtVa107 and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

Associated values for message instance MsgEvId2

Associated value	Block parameters
1	BatchName
2	StepNo
3	BatchID
4	Position feedback Rbk
5	Signal status ExtMsg1
6	Signal status ExtMsg2
7	Signal status ExtMsg3
8	Signal status ExtMsg4
9	ExtVa209
10	ExtVa210

The associated values 9 and 10 are assigned to the parameters ExtVa209 ... ExtVa210 and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

See also

FmCont block diagram (Page 338)

FmCont modes (Page 308)

FmCont error handling (Page 319)

4.2.6 FmCont I/Os

FmCont I/Os

Input parameters

Parameter	Description	Type	Default
AccMode	1 = Transfer of operating parameters SubN1_ID, SubN2_ID, RackNo, SlotNo and Channel to internal processing	BOOL	1
AdvCoEn	1 = Enable program mode via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AdvCoModSP	Type of program mode: 1 = Setpoint specification 0 = Manipulated variable specification	BOOL	1
AdvCoMstrOn	Activate (0-1) or deactivate (1-0) program mode via edge transition	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AdvCoMV	Specified value from the external program	REAL	0.0
AdvCoOn	1 = Enable program mode via faceplate	BOOL	0
AutModLi	1 = Automatic mode via interconnection or SFC (controlled via ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AutModOp	1 = Automatic mode by operator (controlled via ModLiOp = 0)	BOOL	0
BatchEn	1 = Enable allocation for batch control	BOOL	0
BatchID	Batch ID	DWORD	16#00000000
BatchName	Batch name	STRING[32]	''
BreakTime	Minimum break time [s]	REAL	1.0
Channel	Controller channel number (1..4)	INT	1
CloseLi	1 = Close via interconnection or CFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
CloseOp	1 = Close via operator	BOOL	0
CoordNo	Coordination number	INT	0
CPI_In	Input for control performance index, which is calculated by the assigned ConPerMon block	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#78
CSF	1 = External error (control system error)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
D_InSel	Input for differentiator: 0 = Control error 1..4 = Channel 1..4 17 = Actual value to feedback	INT	0
DeadBand	Width of dead band	REAL	0.0
DiffGain	Gain of differentiator [1...10] DiffGain = TD/(delay time of D action)	STRUCT • Value: REAL • ST: BYTE	- • 5.0 • 16#80
EN	1 = Called block will be processed	BOOL	1
ER_A_DC	Delay for incoming alarms during control error monitoring	REAL	0.0
ER_A_DG	Delay for outgoing alarms during control error monitoring	REAL	0.0
ER_AH_En	1 = Activate alarm (high) for control error monitoring	BOOL	1
ER_AH_Lim	Alarm limit (high) for control error monitoring	REAL	100.0
ER_AH_MsgEn	1 = Activate messages for alarm (high) for control error monitoring	BOOL	1
ER_AL_En	1 = Activate alarm (low) for control error monitoring	BOOL	1
ER_AL_Lim	Alarm limit (low) for control error monitoring	REAL	-100.0
ER_AL_MsgEn	1 = Activate messages for alarm (low) for control error monitoring	BOOL	1
ER_Hyst	Alarm hysteresis for control error	REAL	1.0
ExtMsg1	Binary input for freely selectable message 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg2	Binary input for freely selectable message 2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg3	Binary input for freely selectable message 3	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg4	Binary input for freely selectable message 4	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtVa106	Associated value 6 for messages (MsgEvID1)	ANY	
ExtVa107	Associated value 7 for messages (MsgEvID1)	ANY	
ExtVa209	Associated value 9 for messages (MsgEvID2)	ANY	
ExtVa210	Associated value 10 for messages (MsgEvID2)	ANY	

Parameter	Description	Type	Default
FbkClosed	Low limit stop signal of position feedback	STRUCT • Value: BOOL • ST: BYTE	- 0 16#80
FbkOpened	High limit stop signal of position feedback	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Feature	I/O for additional functions (Page 309)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
FM355	Module type: 0: FM 355 S; 1: FM 355 C	BOOL	0
FuzOptOn	Fuzzy optimization	BOOL	0
Gain	Proportional gain Gain.ST = 16#FF: Operable in faceplate	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#FF
LogAddr	Logical address FM 355	INT	0
Man	Manual specification for the manipulated variable	REAL	0.0
ManHiLim	Limit (high) for manual parameter Man	REAL	100.0
ManLoLim	Limit (low) for manual parameter Man	REAL	0.0
ManModLi	1 = Manual mode via interconnection or SFC (controlled via ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ManModOp	1 = Manual mode via OS operator (controlled via ModLiOp = 0)	BOOL	1
Mode	Operating mode	DWORD	16#0
ModLiOp	Operating mode switchover between: 0 = Operator 1 = Interconnection or SFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MotorTime	Motor manipulating time [s]	REAL	30.0
MS_RelOp	1 = Maintenance release by OS operator	BOOL	0
MsgEvId1	Message number (assigned automatically)	DWORD	16#FFFFFFFF
MsgEvId2	Message number (assigned automatically)	DWORD	16#FFFFFFFF
MsgLock	1 = Suppress process messages	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_FmTrkOn	1 = Manipulated variable tracking in the FM	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
MV_HiLim	Limit (high) for manipulated variable MV	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
MV_LoLim	Limit (low) for manipulated variable MV	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV_OpScale	OS display range for manipulated variable MV	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
MV_Safe	Safety manipulated variable	REAL	0.0
MV_SafeOn	1 = Output safety manipulated variable MV_Safe at output MV	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_Trk	Tracking value for the manipulated variable MV	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV_TrkOn	1 = Tracking of manipulated variable MV	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_Unit	Unit of measure for manipulated variable	INT	1342
NegGain	0 = Positive controller gain: ER = Gain · (SP - PV) 1 = Negative controller gain: ER = Gain · (PV - SP)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
NormMV	Manipulated variable range (MV) for standardizing the proportional gain (GAIN)	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
NormPV	Process value range (PV) for standardizing the proportional gain (GAIN)	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
Occupied	Occupied by batch control	BOOL	0
OosLi	Edge transition (0-1) = Out of service, via interconnection or SFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OP_Sel	Operation via OP 0 = Off (P bus) 1 = On (K bus)	BOOL	0
OpenLi	1 = Open via interconnection or CFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OpenOp	1 = Open via operator	BOOL	0

Parameter	Description	Type	Default
OptimEn	1 = Enable optimization of PID parameters by PID tuner	BOOL	0
OptimOcc	1 = Optimization running	BOOL	0
OS_Perm	I/O for operator control permissions (Page 309)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
OS1Perm	I/O for operator control permissions (Page 309)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
P_FbkSel	1 = P action in feedback	BOOL	0
PropSel	1 = Activate P action	BOOL	1
PulseTime	Minimum pulse duration [s]	REAL	1.0
PV_A_DC	Delay time for incoming PV alarms [s]	REAL	0.0
PV_A_DG	Delay time for outgoing PV alarms [s]	REAL	0.0
PV_AH_En	1 = Enable PV alarm limit (high)	BOOL	1
PV_AH_Lim	PV alarm limit (high)	REAL	95.0
PV_AH_MsgEn	1 = Enable message for PV alarm (high)	BOOL	1
PV_AL_En	1 = Enable PV alarm limit (low)	BOOL	1
PV_AL_Lim	PV alarm limit (low)	REAL	5.0
PV_AL_MsgEn	1 = Enable message for PV alarm (low)	BOOL	1
PV_Hyst	Hysteresis for PV alarm, warning and tolerance limits	REAL	1.0
PV_OpScale	Limit for scale in PV bar graph of faceplate	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
PV_T_DC	Delay time for incoming PV tolerance messages [s]	REAL	0.0
PV_T_DG	Delay time for outgoing PV tolerance messages [s]	REAL	0.0
PV_TH_En	1 = Enable PV tolerance limit (high)	BOOL	0
PV_TH_Lim	PV tolerance message limit (high)	REAL	85.0
PV_TH_MsgEn	1 = Enable message for PV tolerance message (high)	BOOL	1
PV_TL_En	1 = Enable PV tolerance limit (low)	BOOL	0
PV_TL_Lim	PV tolerance message limit (low)	REAL	15.0
PV_TL_MsgEn	1 = Enable message for PV tolerance message (low)	BOOL	1
PV_Unit	Unit of measure for process value	INT	1001
PV_W_DC	Delay time for incoming PV warnings [s]	REAL	0.0
PV_W_DG	Delay time for outgoing PV warnings [s]	REAL	0.0
PV_WH_En	1 = Enable PV warning limit (high)	BOOL	1

Parameter	Description	Type	Default
PV_WH_Lim	PV warning limit (high)	REAL	90.0
PV_WH_MsgEn	1 = Enable message for PV warning (high)	BOOL	1
PV_WL_En	1 = Enable PV warning limit (low)	BOOL	1
PV_WL_Lim	PV warning limit (low)	REAL	10.0
PV_WL_MsgEn	1 = Enable message for PV warning (low)	BOOL	1
RackNo	Rack number	BYTE	16#FF
Rbk	Position feedback for display on OS	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
RbkHyst	Alarm hysteresis for position feedback	REAL	1.0
RbkWH_En	1 = Enable warning (high) for position feedback	BOOL	1
RbkWH_Lim	Limit for position feedback of warning (high)	REAL	100.0
RbkWH_MsgEn	1 = Enable messages for warning (high) for position feedback	BOOL	1
RbkWL_En	1 = Enable warning (low) for position feedback	BOOL	1
RbkWL_Lim	Limit for position feedback of warning (low)	REAL	0.0
RbkWL_MsgEn	1 = Enable messages for warning (low) for position feedback	BOOL	1
RefStdDevIn	Reference value of PV standard deviation (sigma) in defined "good" state of control loop	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#78
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
S_RbkOnPIDTun	Simulation of position feedback on; For PCS 7 PID tuner only	BOOL	0
S_RbkPIDTun	Simulated position feedback	REAL	50.0
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SelFp1	Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
SimOn	1 = Simulation on	BOOL	0
SimPV	Process value used for SimOn = 1	REAL	0.0
SimRbk	Position feedback used for SimOn = 1	REAL	0.0
SlotNo	Slot number	BYTE	16#FF
SP_DnRaLim	Limit (low) for ramp of setpoint [SP_Unit/s]	REAL	100.0
SP_ExHiLim	Limit (high) for external setpoint	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80

Parameter	Description	Type	Default
SP_ExLoLim	Limit (low) for external setpoint	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_Ext	External setpoint of - (to interconnection)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_ExtLi	1 = Select external setpoint (via interconnection)	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_ExtOp	1 = Select external setpoint (via operator)	BOOL	0
SP_InHiLim	Limit (high) of internal setpoint	REAL	100.0
SP_InLoLim	Limit (low) of internal setpoint	REAL	0.0
SP_Int	Internal setpoint for operation	REAL	0.0
SP_IntLi	1 = Select internal setpoint (via interconnection)	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_IntOp	1 = Select internal setpoint (via operator)	BOOL	0
SP_LiOp	Select setpoint source (internal/external): 1 = Via interconnection 0 = Via operator	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_RateOn	1 = Activate limitation of setpoint gradients	BOOL	0
SP_RmpModTime	1 = Use time (SP_RmpTime) for setpoint ramp, 0 = Use gradient	BOOL	0
SP_RmpOn	1 = Activate setpoint ramp to target setpoint SP_RmpTarget	BOOL	0
SP_RmpTarget	Target setpoint for setpoint ramp	REAL	0.0
SP_RmpTime	Time for setpoint ramp [s] from current SP up to SP_RmpTarget	REAL	0.0
SP_TrkExt	1 = Bumpless switchover from external to internal setpoint active	BOOL	1
SP_TrkPV	1 = Setpoint follows PV in manual mode and with tracking	BOOL	0
SP_UpRaLim	Gradient limit (high) for setpoint [SP_Unit/s]	REAL	100.0
StepCon	Controller type in the FM 355 S: 0 = Pulse controller 1 = Step controller	BOOL	0
StepNo	Batch step number	DWORD	16#00000000
StopLi	1 = Stop via interconnection or CFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StopOp	1 = Stop via operator	BOOL	0
SubN1_ID	ID of the primary DP master system	BYTE	16#FF

Parameter	Description	Type	Default
SubN2_ID	ID of the redundant DP master system	BYTE	16#FF
TD	Derivative action time [s] TD.ST = 16#FF: Operable in faceplate	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
TI	Integral action time [s] TI.ST = 16#FF: Operable in faceplate	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#FF
UserStatus	Freely assignable bits for use in faceplate	BYTE	16#00

In/out parameters

Parameter	Description	Type	Default
EnCoord	Current coordination number	STRUCT • CO_ACT : INT	- • 0

Output parameters

Parameter	Description	Type	Default
AdvCoAct	1 = Program mode is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AdvCoRdy	1 = Program mode available	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AutAct	1 = Automatic mode is active 0 = Manual mode is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ChFM_Err	1 = Channel error on the module	BOOL	0
Close	Control output: 1 = Closed is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
EnCoNum	Coordination number	BYTE	16#0
ENO	1 = Block algorithm completed without errors	BOOL	0
ER	Control error	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
ER_AH_Act	1 = Alarm limit (high) for control error violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ER_AL_Act	1 = Alarm limit (low) for control error violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ErrorNum	Output of present error number, for error numbers that can be output by this block, see FmCont error handling (Page 319)	INT	-1
FbkClsOut	1 = Low limit stop of the position feedback reached	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FbkOpnOut	1 = High limit stop of the position feedback reached	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FFwdOut	Disturbance variable generated in the FM	STRUCT Value: REAL ST: BYTE	- 0.0 16#80

Parameter	Description	Type	Default
FuzCon	Controller type: 0 = PID controller 1 = Fuzzy controller	BOOL	0
FuzOptAct	1 = Optimization of fuzzy controller active	BOOL	0
FuzSP_PV_Act	Fuzzy controller display: Setpoint < actual value	BOOL	0
GainEff	Effective proportional gain, depends on Gain and NormPV	REAL	1.0
ManAct	1 = Manual mode active	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
ManARW_Act	1 = Tracking mode or anti-reset windup by secondary controller	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ManHiOut	Limit (high) for manual mode, corresponds to the input parameter ManHiLim	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
ManLoOut	Limit (low) for manual mode, corresponds to the input parameter ManLoLim	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
ModErr	1 = Module error	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_Release	Maintenance release: 1 = Release for OS operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MsgAckn1	Alarm acknowledgement status 1 (output STATUS of first ALARM_8P)	WORD	16#0000
MsgAckn2	Alarm acknowledgement status 2 (output STATUS of second ALARM_8P)	WORD	16#0000
MsgErr1	Alarm error 1 (output ERROR of first ALARM_8P)	BOOL	0
MsgErr2	Alarm error 2 (output ERROR of second ALARM_8P)	BOOL	0
MsgStat1	Alarm status 1 (output ERROR of first ALARM_8P)	WORD	16#0000
MsgStat2	Alarm status 2 (output ERROR of second ALARM_8P)	WORD	16#0000
MV	Manipulated variable	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

Parameter	Description	Type	Default
MV_FmTrkAct	1 = Tracking a manipulated variable in the FM is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_HiAct	1 = Limit (high) of manipulated variable violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_LoAct	1 = Limit (low) of manipulated variable violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_SafeAct	1 = Safety manipulated variable of the FM is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_SpliA	Manipulated variable A of split-range function	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV_SpliB	Manipulated variable B of split-range function	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV_UnitOut	Unit of measure for manipulated variable, for interconnecting to MV_Unit input parameter of the ConPerMon block	INT	0
OosAct	1 = Block is "out of service"	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Open	Control output 1 = Open is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
OS1PermOut	Display of OS_Perm1	DWORD	16#FFFFFFFF
OS1PermLog	Display of OS1Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
ParFM_Err	1 = Direct parameter-assignment error of the FM or input Channel configured incorrectly	BOOL	0
PerAccErr	1 = I/O access error	BOOL	0
PV_AH_Act	1 = PV alarm (high) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
PV_AL_Act	1 = PV alarm (low) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_Out	Output for process value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_TH_Act	1 = PV tolerance message (high) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_TL_Act	1 = PV tolerance message (low) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_ToleHi	Limit (high) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_ToleLo	Limit (low) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_UnitOut	Unit of measure for process value, for interconnecting with PV_Unit input parameter of the ConPerMon block	INT	0
PV_WH_Act	1 = PV warning (high) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_WL_Act	1 = PV warning (low) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RbkClsOut	Output for high limit stop signal of position feedback	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RbkOpnOut	Output for low limit stop signal of position feedback	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RbkOut	Output for position feedback	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
RbkWH_Act	1 = Warning (high) for position feedback active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RbkWL_Act	1 = Warning (low) for position feedback active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
RetValue	Return value of WRREC/RDREC	WORD	16#0
SP	Setpoint used by controller	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_DnRaAct	1 = Negative gradient limiting of setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_ExHiAct	1 = Limit (high) for external setpoint has been reached	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP_ExLoAct	1 = Limit (low) for external setpoint has been reached	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP_ExtAct	1 = External setpoint is active 0 = Internal setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP_ExtOut	External setpoint, corresponds to input parameter SP_Ext	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_InHiOut	Limit (high) for SP_Int corresponds to input parameter SP_InHiLim	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
SP_InLoOut	Limit (low) for SP_Int corresponds to input parameter SP_InLoLim	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_UpRaAct	Positive gradient limiting of setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
SplitRange	1 = Split-range function has been activated	BOOL	0
Status1	Status word 1 (Page 304)	DWORD	16#00000000
Status2	Status word 2 (Page 304)	DWORD	16#00000000
Stop	Control output 1 =Stopped is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
SumMsgAct	1 = Active process alarm	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
WithRbk	Controller type: 0 = Step controller without position feedback 1 = Step controller with position feedback	BOOL	0

See also

FmCont messaging (Page 321)

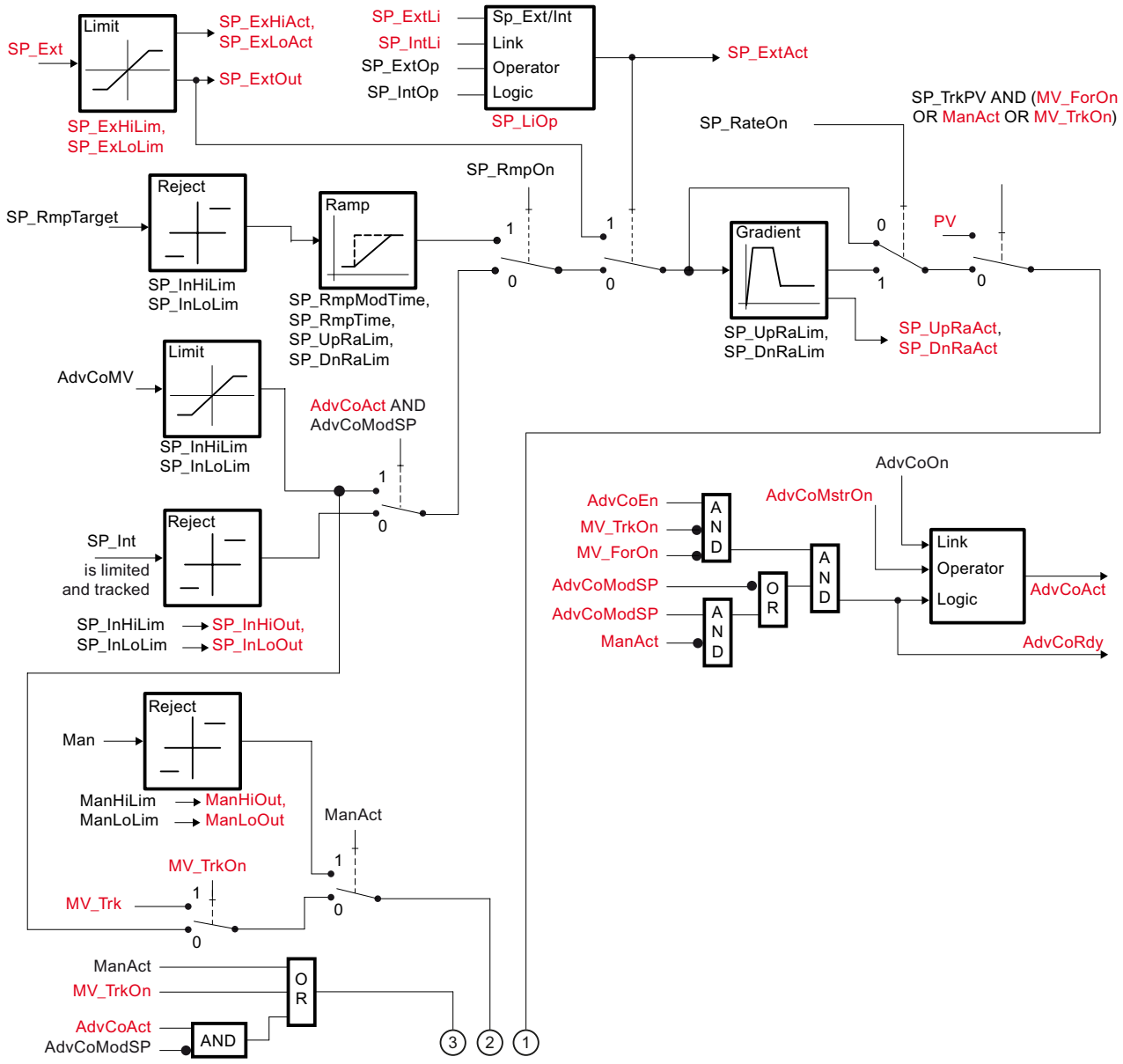
FmCont block diagram (Page 338)

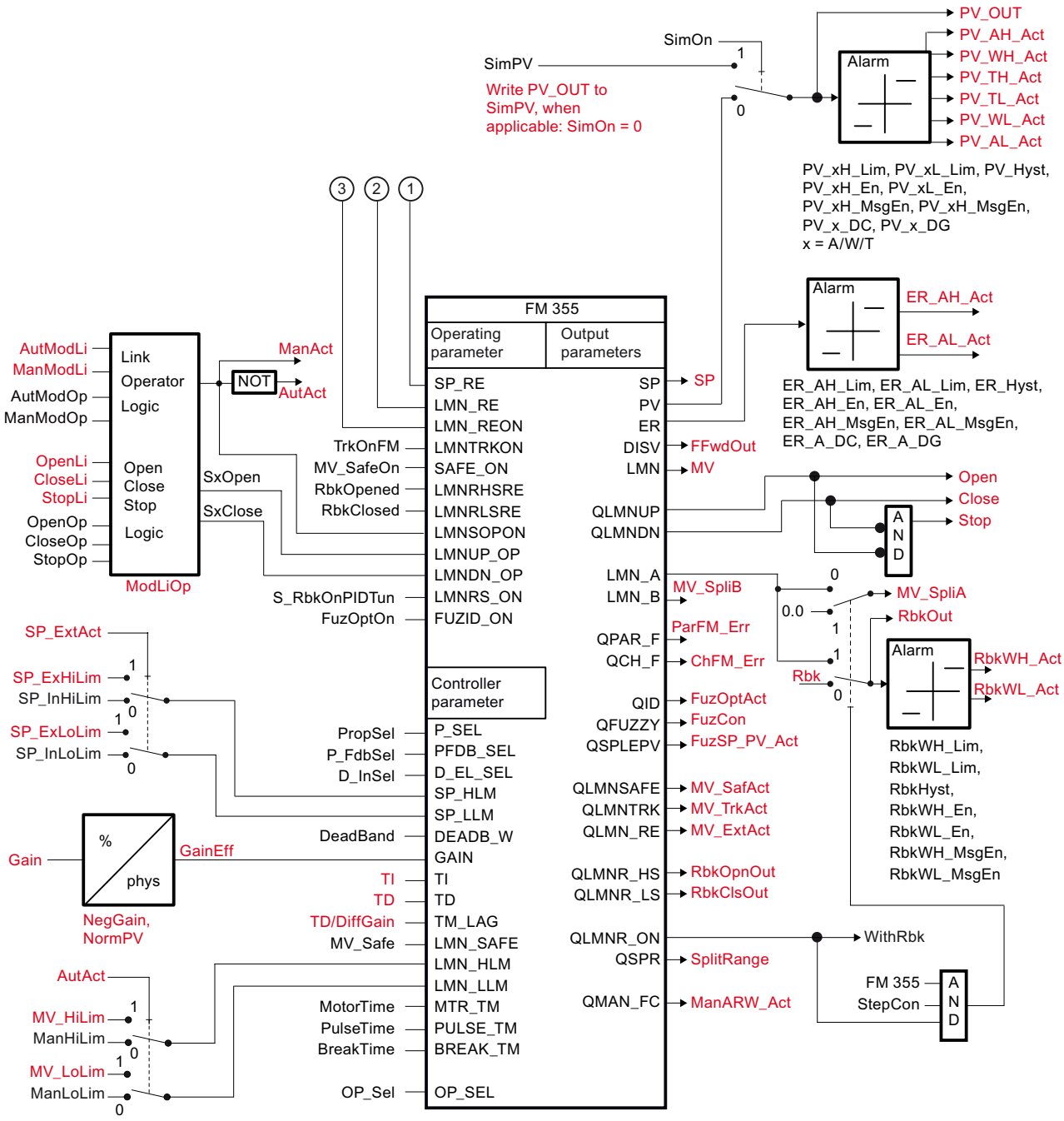
FmCont modes (Page 308)

Safe position for motors, valves and controllers (Page 120)

4.2.7 FmCont block diagram

FmCont block diagram





See also

FmCont I/Os (Page 324)

FmCont messaging (Page 321)

FmCont error handling (Page 319)

FmCont functions (Page 309)

FmCont modes (Page 308)

Description of FmCont (Page 304)

4.2.8 Operator control and monitoring

4.2.8.1 FmCont views

Views of the FmCont block

The block FmCont provides the following views:

- Standard view of FM controllers (analog) (Page 131)
- Standard view of FM controllers (pulse controller) (Page 135)
- Standard view of FM controllers (step controller with position feedback) (Page 139)
- Standard view of FM controllers (step controller without position feedback) (Page 143)
- Message view (Page 169)
- Limit value view of FM controllers (Page 157)
- Trend view (Page 172)
- Ramp view (Page 167)
- Parameter view of FM controllers (Page 153)
- Preview of FM controllers (Page 165)
- Memo view (Page 171)
- Batch view (Page 170)
- Block icons for PID and FM controller (Page 181)

Refer to the Structure of the faceplate (Page 123) and Block icon structure (Page 174) sections for general information about the faceplate and block icon.

4.3 FmTemp - Interface to temperature controller modules FM 355-2

4.3.1 Description of FmTemp

Object name (type + number) and family

Type + number: FB 1819

Family: Control

Area of application for FmTemp

The block is used for the following applications:

- Fixed setpoint control
- Cascade control
- Ratio control
- Split-range control

How it works

Block FmTemp is used to interface the FM 355-2 temperature controller modules.

FmTemp can be used for the C (continuous controllers) and S (step and pulse controllers) module types. It contains the algorithms of the setpoint ramp, the setpoint rise limitation, and the limit monitoring of the process value, the control error, and the position feedback. Limit monitoring is not used on the module.

The control function itself (e.g. PID algorithm) is processed on the module. You can use the FmTemp block to monitor all relevant process values and to change all relevant controller parameters.

Application examples of the FM 355-2 and detailed descriptions of the associated input and output parameters can be found in the manual of the FM 355-2 temperature controller.

It is primarily used for controlling temperature processes, but can also control level and flow processes which are not excessively fast, for example.

Module FM 355-2 features online optimization of the PID parameters. You can set the corresponding parameters for performing online optimization in the CFC chart at block FmTemp.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100). Set the input `LogAddr` to the module address from HW Config and the input `Channel` to the desired controller channel (0..3).

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The parameter `CoordNo` is set
- The parameter `FM355_2` is set in accordance with the module type C/S
- The in/out parameter `EnCoord` is interconnected with the output `EN_CO_x` of the FM_CO block of the basic library (x = number of the rack)
- The output `EnCoNum` is interconnected with the input `ENCOx_yy` of the FM_CO block (x = number of the rack, yy = coordination number).

For the control loop monitoring to work as planned in the trend view of the controller faceplates, the

`S7_xarchive:='Value, shortterm;'`

attributes in the process tag types for control loops at the controller function block must be set for the following tags:

- Input parameters:
  - `CPI_In`
- Output parameters
  - `MV`
  - `MV_HiAct`
  - `MV_LoAct`
  - `AutAct`
  - `SP`
  - `PV_Out`
  - `PV_ToleHi`
  - `PV_ToleLo`

Startup characteristics

Use the Setting the startup response (Page 187) feature to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

Status word allocation for `Status1` parameter

For a description of the individual parameters see the section FmTemp I/Os (Page 361).

Status bit	Parameter
0	`Occupied`
1	`BatchEn`
2	`SimOn`
3	`OosAct.Value`
4	`OosLi.Value`
5	`AutAct.Value`
6	Not allocated
7	`ManAct.Value`
8	`SP_ExtAct.Value`
9	`MV_SafeOn.Value`
10	`MV_TrkOn.Value`
11	`MV.Value > ManLoLim` for continuous or pulse controller `NOT FbkClosed.Value` for step controller with/without position feedback
12	`Open.Value`
13	`Close.Value`
14	`Stop.Value`
15	`FbkOpened.Value`
16	`FbkClosed.Value`
17 - 18	Not allocated
19	AdvCoAct
20	1 = Input parameter `Rbk` is not interconnected (RbkOut.ST = 16#FF)
21	`NegGain`
22 - 27	Not allocated
28	1 = Analog controller (`FM355_2 = 1`)
29	1 = Pulse controller (`FM355_2 = 0 ANDStepCon = 0`)
30	1 = Step controller with position feedback (`FM355_2 = 0` AND `StepCon = 1` AND `WithRbk = 1`)
31	1= Step controller without position feedback (`FM355_2 = 0` AND `StepCon = 1` AND `WithRbk = 0`)

Status word allocation for `Status2` parameter

Status bit	Parameter
0	MsgLock
1	PV_AH_Act.Value
2	PV_WH_Act.Value
3	PV_TH_Act.Value
4	PV_TL_Act.Value
5	PV_WL_Act.Value
6	PV_AL_Act.Value
7	PV_AH_En
8	PV_WH_En
9	PV_TH_En
10	PV_TL_En
11	PV_WL_En
12	PV_AL_En
13	PV_AH_MsgEn
14	PV_WH_MsgEn
15	PV_TH_MsgEn
16	PV_TL_MsgEn
17	PV_WL_MsgEn
18	PV_AL_MsgEn
19	ER_AH_Act.Value
20	ER_AL_Act.Value
21	ER_AH_En
22	ER_AL_En
23	ER_AH_MsgEn
24	ER_AL_MsgEn
25	RbkWH_Act.Value
26	RbkWL_Act.Value
27	RbkWH_En
28	RbkWL_En
29	RbkWH_MsgEn
30	RbkWL_MsgEn
31	MS_RelOp

See also

FmTemp functions (Page 346)
FmTemp messaging (Page 358)
FmTemp modes (Page 345)
FmTemp error handling (Page 356)
FmTemp block diagram (Page 376)

4.3.2 FmTemp modes

FmTemp operating modes

The block can be operated using the following modes:

- Automatic mode (Page 29)
- Manual mode (Page 29)
- Program mode for closed-loop controllers (Page 34)
- Out of service (Page 27)

The next section provides additional block-specific information relating to the general descriptions.

Automatic mode

You can find general information on "Automatic mode", switching modes and bumpless switchover in the Manual and Automatic mode for control blocks (Page 29) section.

Manual mode

You can find general information on "Manual mode", switching modes and bumpless switchover in the Manual and Automatic mode for control blocks (Page 29) section.

Program mode for controllers

General information on "Program mode for controllers" is available in the section Program mode for closed-loop controllers (Page 34).

Out of service

You will find general information about the "Out of service" mode in the Out of service (Page 27) section.

See also

Description of FmTemp (Page 341)

FmTemp functions (Page 346)

FmTemp error handling (Page 356)

FmTemp messaging (Page 358)

FmTemp I/Os (Page 361)

FmTemp block diagram (Page 376)

4.3.3 FmTemp functions

Functions of FmTemp

The functions for this block are listed below.

Module types

FmTemp can be used for the C (continuous controllers) and S (step controllers with and without position feedback and pulse controllers) module types. You can use the following parameters to identify which module type and controller type has been set:

FM355	StepCon	WithRbk	Module type, controller type
1 or C	-	-	FM 355-2 C: Continuous controller
0 or S	1	1	FM 355-2 S: Step controller with position feedback
0 or S	1	0	FM 355-2 S: Step controller without position feedback
0 or S	0	-	FM 355-2 S: Pulse controller

Generating manipulated variables for continuous controllers, step controllers with position feedback, or pulse controllers

The manipulated variable MV and the actuating signals Open, Close, and Stop are generated as follows:

MV_SafeOn	MV_FMTrkOn	ManAct	MV_TrkOn	AdvCoAct AND NOT AdvCoMod SP	MV =	Limitation of the manipulated variable	State	Open, Close, Stop
1	-	-	-	-	MV_Safe	MV_HiLim MV_LoLim	Tracking to safety value	**Cont. controller:** Open, Close, Stop = False **Step controller with position feedback:** Depending on Rbk and MV, the output signals Open, Close and Stop are generated using the algorithm of a positioner. **Pulse controller:** Depending on MV, the output signals Open and Close are generated using the algorithm of a pulse controller (Stop = 0).
0	1	-	-	-	Prepared FM analog input	MV_HiLim MV_LoLim	Tracking to an FM analog input	
0	0	1	-	-	Man	ManHiLim ManLoLim	Manual mode, set by the operator	
0	0	0	1	-	MV_Trk	MV_HiLim MV_LoLim	Tracking to block input MV_Trk	
0	0	0	0	1	AdvCoMV	MV_HiLim MV_LoLim	Higher-level program mode	
0	0	0	0	0	P_Part + I_Part + D_Part + FFwd	MV_HiLim MV_LoLim	Automatic mode (PID algorithm)	

Generating manipulated signals for step controllers without position feedback (WithRbk = 0)

The manipulated variable signals `Open`, `Close` and `Stop` can be generated as follows:

ManAct	Open, Close, Stop	State	ManAct
1	The output signals are generated using input signals `OpenOp/Li`, `CloseOp/Li` or `StopOp/Li`.	Manual mode, set by the operator	1
0	The output signals are generated using PID output parameters `P_Part`, `I_Part`, `D_Part` and `FFwd`	Automatic mode (PID algorithm)	0

Tracking and limiting a manipulated variable (cont. controller, step controller with position feedback and pulse controller)

The block provides the standard function Tracking and limiting a manipulated value (Page 111).

Safe position

The controller groups have their own mechanism for connecting a safety value (see manual for Temperature Controller FM 355-2).

"Actuator active" information

For continuous and pulse controllers: If the manipulated variable `MV` is greater than the minimum manual limit `ManLoLim`, this is recognized as actuator active.

For step controllers: If the parameter `FbkClosed` = 0, this is known as "Actuator active".

This status can be used to display a custom icon in the process image, for example, and is saved in the status word (see Status word section in Description of FmTemp (Page 341)).

Limit monitoring of position feedback (cont. controller, step controller with position feedback and pulse controller)

The block provides the standard function Limit monitoring of the feedback (Page 76).

External/internal setpoint specification

The block provides the standard function Setpoint input - internal and external (Page 96).

Setpoint limiting for external setpoints

The block provides the standard function Setpoint limiting for external setpoints (Page 110).

Limitation of rate of change of setpoint

The block provides the standard function Ramp limiting of the setpoint (Page 108).

Using setpoint ramp

The block provides the standard function Using a setpoint ramp (Page 98).

Tracking setpoint in manual mode

The block provides the standard function Setpoint tracking the process variable in manual mode (Page 110).

Simulating signals

The block provides the standard function Simulating signals (Page 93).

You can simulate the following values:

- Process value (`SimPV`)
- Position feedback (`SimRbk`)

Note

The simulated process value `SimPV` only affects alarm processing and not the PID algorithm in the controller module.

Limit monitoring of the process value

The block provides the standard function Limit monitoring of the process value (Page 70).

Control error generation and dead band

The block provides the standard function Control error generation and deadband (Page 109).

Limit monitoring of control error

The block provides the standard function Monitoring the error signal limits (Page 77).

Inverting control direction

The block provides the standard function Inverting the control sense (Page 108).

Physical standardization of setpoint, manipulated variable and process value

Controller gain `Gain` is entered either using a physical variable or as standardized value.

- `Gain` as a physical variable:

  The standardized variables retain their default values:

  – `NormPV.High` = 100 and `NormPV.Low` = 0

  – `NormMV.High` = 100 and `NormMV.Low` = 0

  For step controllers with/without position feedback and pulse controllers, the values of `NormMV.High` and `NormMV.Low` are not taken into account. The algorithm uses default values 0 and 100 for internal calculations.

  The effective gain is: `GainEff` = `Gain`

- Entering a standardized `Gain` (dimensionless):

  Change the standardized variables to the actual range of the process values and manipulated variables.

  – Internal and external setpoints; the process value and corresponding parameters are entered according to the physical measuring range of the process value.

  Continuous controller, pulse controller:

  – The manual value, the tracking value of the manipulated variable, feedforward control and the corresponding parameters are set according to the physical measuring range of the manipulated variable.

  Step controller with position feedback:

  – The manual value, the tracking value of the manipulated variable, feedforward control and the corresponding parameters are entered as a percentage between 0 ... 100.

  Step controller without position feedback:

  – No physical measuring range available.

  The effective gain is:

  – Step controller with/without position feedback:

    `GainEff` = 100,0 / (`NormPV.High` - `NormPV.Low`) · `Gain`

  – Continuous controller, pulse controller:

    `GainEff = (NormMV.High - NormMV.Low)/(NormPV.High - NormPV.Low) · Gain`

Selecting a unit of measure

The block provides the standard function Selecting a unit of measure (Page 53).

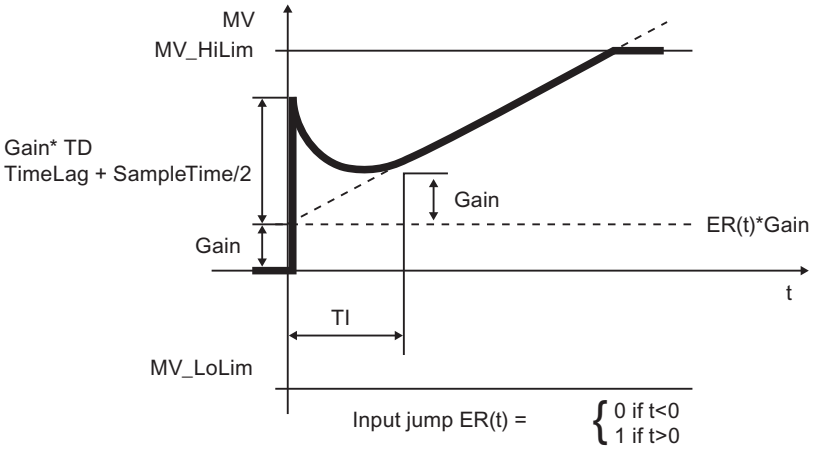
PID algorithm

The manipulated variable is generated in automatic mode according to the following algorithm:

`MV = Gain · (1 + 1/(TI · s) + TD/( 1 + TD/DiffGain · s) · ER`

Where: s = Complex number

The following step response occurs:



Note

This formula describes a standard application where P, I and D action is activated and the P and D actions are not in the feedback circuit (`PropSel` = 1, `TI ≠ 0 D_InSel = 0` and `PropFacSP` = 1).

The D action delay is derived from `TD/DiffGain`.

- The P action can be shut down by `PropSel = 0`.
- The I action can be shut down by `TI = 0`.
- The D action can be shut down by `TD = 0`.

Structure segmentation at controllers

The PID controller algorithm of `FM355` features structure segmentation. It is activated via the `PropFacSel` and `D_InSel` parameters. The precise functionality is described in the `FM355` manual.

Online optimization of the PID controller parameters

- Optimization sequence

  The optimization sequence is as follows:

  – Create a stationary state

  – Set `PID_On = 1` (if PID parameters are required)

  – Set `TunD_MV / TunC_MVLMN`

  – Set `TunOn = 1` (phase 1, ready for optimization)

  – Start the optimization using a step change in the setpoint or by setting `TunStart`

  If you have not made any configuration errors, the controller optimization is now in phase 2 and `StatusH` is 0.

  – When the point of inflection has been reached (PHASE ≥ 3), evaluate the diagnostics display at the `StatusH` parameter. Phase 0 is reached in a few cycles for process type I and the optimization is completed in full. For process types II and III, the optimization goes to phase 7 (checking the process type). If `StatusH` > 20000, a valuation error has occurred or the point of inflection has not been reached. In this case, repeat the procedure.

- Result

  – Once optimization is completed, the parameters `PropFacSP`, `GAIN`, `TI`, `TD`, `DiffGain` and `ConZone` are updated (for both the module and at FmTemp). Furthermore, the PI or PID parameter sets are saved on the `FM 355-2`.

  – The precise procedure is described in the `FM 355-2` manual of the temperature controller module.

- Permanent backup of optimized controller parameters

  – Save, compile and download the hardware configuration; the optimized controller parameters are now in the system data block (SDB).

  – Transfer the modified parameters to the offline data management of the CFC via Chart > Readback....

Anti-windup

The PID control algorithm of `FM355` has an anti-windup function. The I action is frozen or tracked after the manipulated variable has reached its limits (`MV_HiLim` or `MV_LoLim`).

Activating and limiting disturbance variables

The block provides a function for activating the disturbance variable. The precise functionality is described in the `FM355-2` manual.

Control zone

The block provides the standard function Using control zones (Page 109).

Forming the signal status for blocks

The block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `ER.ST`
- `FFwdOut.ST`
- `Rbk.ST`
- `MV.ST`

Configurable reactions using the Feature parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)
1	Reaction to the out of service mode (Page 199)
2	Resetting the commands for changing the mode (Page 193)
4	Setting switch or button mode (Page 195)
22	Update acknowledgment and error status of the message call (Page 193)

Operator control permissions via parameters OS_Perm and OS1Perm

The block has the following operator control permissions (Page 358) for the parameter `OS_Perm`:

Bit	Function
0	1 = Operator can switch into automatic mode `AutModOp`
1	1 = Operator can switch into manual mode `ManModOp`
2	1 = Operator can switch into "out of service" mode `OosOp`
3	1 = Operator can switch into program mode `AdvCoEn`
4	1 = Operator can switch the setpoint to external `SP_ExtOp`
5	1 = Operator can switch the setpoint to internal `SP_IntOp`
6	1 = Operator can change the internal setpoint `SP_Int`
7	Continuous controllers, pulse controllers or step controllers with position feedback: 1 = Operator can change the manual parameter `Man` Step controller without position feedback: 1 = Operator can change the manual operation signals `OpenOp, StopOp, CloseOp`
8	1 = Operator can change maximum usage limit of the setpoint `SP_InHiLim`
9	1 = Operator can change minimum usage limit of the setpoint `SP_InLoLim`
10	1 = Operator can change maximum usage limit of the manipulated variable `ManHiLim`
11	1 = Operator can change minimum usage limit of the manipulated variable `ManLiLim`

Bit	Function
12	1 = Operator can use the setpoint's gradient limitation function SP_RateOn
13	1 = Operator can change the setpoint's high limit for the ramp SP_UpRaLim
14	1 = Operator can change the setpoint's low limit for the ramp SP_DnRaLim
15	1 = Operator can change between the time value or the value for the ramp SP_RmpModTime
16	1 = Operator can change the ramp time SP_RmpTime
17	1 = Operator can change the target setpoint SP_RmpTarget for the setpoint ramp
18	1 = Operator can activate the setpoint ramp function SP_RmpOn
19	Not allocated
20	1 = Operator can activate the track setpoint in manual mode function SP_TrkPV
21	1 = Operator can activate the bumpless changeover from external to internal function SP_TrkExt
22	1 = Operator can change the gain parameter Gain
23	1 = Operator can change the integral action time parameter TI
24	1 = Operator can change the derivative action time parameter TD
25	1 = Operator can change the derivative gain parameter DiffGain
26	1 = Operator can change the dead band parameter DeadBand
27	1 = Operator can change the control zone parameter ConZone
28	1 = Operator can change the integral action time parameter MotorTime
29	1 = Operator can change the integral action time parameter PulseTime
30	1 = Operator can change the integral action time parameter BreakTime
31	Not allocated

The block has the following operator control permissions for the OS1Perm parameter:

Bit	Function
0	1 = Operator can change the limit (process value) for the high alarm PV_AH_Lim
1	1 = Operator can change the limit (process value) for the high warning PV_WH_Lim
2	1 = Operator can change the limit (process value) for the high tolerance PV_TH_Lim
3	1 = Operator can change the hysteresis (process value) PV_Hyst
4	1 = Operator can change the limit (process value) for the low tolerance PV_TL_Lim
5	1 = Operator can change the limit (process value) for the low warning PV_WL_Lim
6	1 = Operator can change the limit (process value) for the low alarm PV_AL_Lim
7	1 = Operator can change the limit (control error) for the high alarm ER_AH_Lim
8	1 = Operator can change the hysteresis (control error) ER_Hyst
9	1 = Operator can change the limit (control error) for the low alarm ER_AL_Lim
10	1 = Operator can change the limit (position feedback) for the high warning RbkWH_Lim
11	1 = Operator can change the hysteresis (position feedback) RbkHyst
12	1 = Operator can change the limit (position feedback) for the low warning RbkWL_Lim
13 - 15	Not allocated
16	1 = Operator can activate the simulation function SimOn
17	1 = Operator can activate the maintenance release function MS_RelOp
13 - 31	Not allocated

Maintenance release

The block provides the standard function Maintenance release (Page 47).

Generating instance-specific messages

The block provides the standard function Generating instance-specific messages (Page 46) without the time stamp function in the I/O devices.

Specifying the display area for process and setpoint values as well as operations

This block provides the standard function Display and operator input area for process values and setpoints (Page 42).

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 122).

See also

FmTemp I/Os (Page 361)

FmTemp modes (Page 345)

FmTemp block diagram (Page 376)

FmTemp error handling (Page 356)

Safety value of the manipulated variable effective at startup (Page 196)

Out of service (Page 27)

Safe position for motors, valves and controllers (Page 120)

Operator permissions (Page 45)

4.3.4 FmTemp error handling

FmTemp troubleshooting

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` output parameter can be used to output various error numbers. An overview of all error numbers is available in the section "Functions of blocks > Error handling (Page 117)".

Error numbers output by this block:

Error number	Meaning of the error number
-1	Default value when implementing the block; block will not be processed.
0	There is no error.
30	The value of `PV` can no longer be displayed in the REAL number field.
31	The value of `SP_Ext` can no longer be displayed in the REAL number field.
33	The value of `MV_Trk` can no longer be displayed in the REAL number field.
35	The value of `Rbk` can no longer be displayed in the REAL number field.
36	The value of `MV` can no longer be displayed in the REAL number field.
50	The controller cannot be switched in program mode, because program mode with setpoint specification (AdvCoModSP=false) is not possible with step controllers without position feedback (WithRbk=false).
60	`|Tl| < SampleTime/2`
61	`|TD| < SampleTime`
62	`DiffGain < 1` or `DiffGain > 10`
63	`TD/DiffGain < SampleTime/2`
64	`PropFacSP < 0` or `PropFacSP > 1`
66	`NormPV_High = NormPV_Low`
67	`MotorTime < SampleTime`
68	`PulseTime < SampleTime`
69	`BreakTime < SampleTime`
70	`Channel < 0` or `Channel > 3`
71	(`D_InSel < 0` or `D_InSel > 4`) and `D_InSel ≠ 17`

See also

Description of FmTemp (Page 341)
FmTemp modes (Page 345)
FmTemp functions (Page 346)
FmTemp messaging (Page 358)
FmTemp I/Os (Page 361)
FmTemp block diagram (Page 376)
Setting switch or button mode (Page 195)

4.3.5 FmTemp messaging

Messaging

The following messages can be generated for this block:

- Control system fault
- Process messages
- Instance-specific messages

Control system fault

The following control system error messages can be output:

Message instance	Message identifier	Message class	Event
MsgEvId2	SIG 6	AS process control message - fault	$$BlockComment$$ External error has occurred

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter CSF. If this signal changes to `CSF = 1`, a control system fault is triggered (`MsgEvId2`, SIG 6).

Process messages

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 1	Alarm - high	$$BlockComment$$ PV - high alarm limit violated
	SIG 2	Warning - high	$$BlockComment$$ PV - high warning limit violated
	SIG 3	Tolerance - high	$$BlockComment$$ PV - high tolerance limit violated
	SIG 4	Tolerance - low	$$BlockComment$$ PV - low tolerance limit violated
	SIG 5	Warning - low	$$BlockComment$$ PV - low warning limit violated
	SIG 6	Alarm - low	$$BlockComment$$ PV - low alarm limit violated
	SIG 7	Alarm - high	$$BlockComment$$ ER - high alarm limit violated
	SIG 8	Alarm - low	$$BlockComment$$ ER - low alarm limit violated

Message instance	Message identifier	Message class	Event
MsgEvId2	SIG 7	Warning - high	$$BlockComment$$ Rbk - high warning limit violated
	SIG 8	Warning - low	$$BlockComment$$ Rbk - low warning limit violated

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Instance-specific messages

You can use up to four instance-specific messages with this block.

Message instance	Message identifier	Message class	Event
MsgEvId2	SIG 1	AS process control message - fault	$$BlockComment$$ External message 1 Status 16#@5%x@
	SIG 2	AS process control message - fault	$$BlockComment$$ External message 2 Status 16#@6%x@
	SIG 3	AS process control message - fault	$$BlockComment$$ External message 3 Status 16#@7%x@
	SIG 4	AS process control message - fault	$$BlockComment$$ External message 4 Status 16#@8%x@

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Associated values for message instance `MsgEvId1`

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`
4	Process value `PV_Out`
5	Control error `ER`
6	`ExtVa106`
7	`ExtVa107`
8	Not allocated
9	Not allocated
10	Not allocated

The associated values 6 and 7 are assigned to the parameters `ExtVa106` ... `ExtVa107` and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

Associated values for message instance `MsgEvId2`

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`
4	Position feedback `Rbk`
5	Signal status `ExtMsg1`
6	Signal status `ExtMsg2`
7	Signal status `ExtMsg3`
8	Signal status `ExtMsg4`
9	`ExtVa209`
10	`ExtVa210`

The associated values 9 and 10 are assigned to the parameters `ExtVa209` ... `ExtVa210` and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

See also

Description of FmTemp (Page 341)

FmTemp functions (Page 346)

FmTemp I/Os (Page 361)

FmTemp modes (Page 345)

FmTemp error handling (Page 356)

FmTemp block diagram (Page 376)

4.3.6 FmTemp I/Os

FmTemp I/Os

Input parameters

Parameter	Description	Type	Default
AccMode	1 = Transfer of operating parameters SubN1_ID, SubN2_ID, RackNo, SlotNo and Channel to internal processing	BOOL	1
AdvCoEn	1 = Enable program mode via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AdvCoModSP	Type of program mode: 1 = Setpoint specification 0 = Manipulated variable specification	BOOL	1
AdvCoMstrOn	Activate (0-1) or deactivate (1-0) program mode via edge transition	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AdvCoMV	Specified value from the external program	REAL	0.0
AdvCoOn	1 = Enable program mode via faceplate	BOOL	0
AutModLi	1 = Automatic mode via interconnection or SFC (controlled via ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AutModOp	1 = Automatic mode by operator (controlled via ModLiOp = 0)	BOOL	0
BatchEn	1 = Enable allocation for batch control	BOOL	0
BatchID	Batch ID	DWORD	16#00000000
BatchName	Batch name	STRING[32]	''
BreakTime	Minimum break time [s]	REAL	1.0
Channel	Controller channel number (0..3)	INT	0
CloseLi	1 = Close via interconnection or CFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
CloseOp	1 = Close via operator	BOOL	0
ConZone	Control zone	REAL	0.0

Parameter	Description	Type	Default
CoordNo	Coordination number	INT	0
CPI_In	Input for control performance index, which is calculated by the assigned ConPerMon block	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#78
CSF	1 = External error (control system error)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
D_InSel	Input for differentiator: 0 = Control error 1..4 = Channel 0..3 17 = Actual value to feedback	INT	0
DeadBand	Width of dead band	REAL	0.0
DiffGain	Gain of differentiator [1...10] DiffGain = TD/(delay time of D action)	STRUCT • Value: REAL • ST: BYTE	- • 5.0 • 16#80
EN	1 = Called block will be processed	BOOL	1
ER_A_DC	Delay for incoming alarms during control error monitoring	REAL	0.0
ER_A_DG	Delay for outgoing alarms during control error monitoring	REAL	0.0
ER_AH_En	1 = Activate alarm (high) for control error monitoring	BOOL	1
ER_AH_Lim	Alarm limit (high) for control error monitoring	REAL	100.0
ER_AH_MsgEn	1 = Activate messages for alarm (high) for control error monitoring	BOOL	1
ER_AL_En	1 = Activate alarm (low) for control error monitoring	BOOL	1
ER_AL_Lim	Alarm limit (low) for control error monitoring	REAL	-100.0
ER_AL_MsgEn	1 = Activate messages for alarm (low) for control error monitoring	BOOL	1
ER_Hyst	Alarm hysteresis for control error	REAL	1.0
ExtMsg1	Binary input for freely selectable message 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
ExtMsg2	Binary input for freely selectable message 2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg3	Binary input for freely selectable message 3	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg4	Binary input for freely selectable message 4	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtVa106	Associated value 6 for messages (MsgEvID1)	ANY	
ExtVa107	Associated value 7 for messages (MsgEvID1)	ANY	
ExtVa209	Associated value 9 for messages (MsgEvID2)	ANY	
ExtVa210	Associated value 10 for messages (MsgEvID2)	ANY	
FbkClosed	Low limit stop signal of position feedback	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FbkOpened	High limit stop signal of position feedback	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Feature	I/O for additional functions (Page 346)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
FM355_2	Module type: 0: FM 355-2 S; 1: FM 355-2 C	BOOL	0
Gain	Proportional gain Gain.ST = 16#FF: Operable in faceplate	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#FF
LoadPID	Load optimized PI/PID parameters	BOOL	0
LogAddr	Logical address FM 355	INT	0
Man	Manual specification for the manipulated variable	REAL	0.0
ManHiLim	Limit (high) for manual parameter Man	REAL	100.0
ManLoLim	Limit (low) for manual parameter Man	REAL	0.0

Parameter	Description	Type	Default
ManModLi	1 = Manual mode via interconnection or SFC (controlled via ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ManModOp	1 = Manual mode via OS operator (controlled via ModLiOp = 0)	BOOL	1
Mode	Operating mode	DWORD	16#0
ModLiOp	Operating mode switchover between: 0 = Operator 1 = Interconnection or SFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MotorTime	Motor manipulating time [s]	REAL	30.0
MS_RelOp	1 = Maintenance release by OS operator	BOOL	0
MsgEvId1	Message number (assigned automatically)	DWORD	16#FFFFFFFF
MsgEvId2	Message number (assigned automatically)	DWORD	16#FFFFFFFF
MsgLock	1 = Suppress process messages	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_FmTrkOn	1 = Manipulated variable tracking in the FM	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_HiLim	Limit (high) for manipulated variable MV	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
MV_LoLim	Limit (low) for manipulated variable MV	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV_OpScale	OS display range for manipulated variable MV	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
MV_Safe	Safety manipulated variable	REAL	0.0
MV_SafeOn	1 = Output safety manipulated variable MV_Safe at output MV	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_Trk	Tracking value for the manipulated variable MV	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV_TrkOn	1 = Tracking of manipulated variable MV	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
MV_Unit	Unit of measure for manipulated variable	INT	1342; %
NegGain	0 = Positive controller gain: ER = Gain (SP - PV) 1 = Negative controller gain: ER = Gain (PV - SP)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
NormMV	Manipulated variable range (MV) for standardizing the proportional gain (GAIN)	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
NormPV	Process value range (PV) for standardizing the proportional gain (GAIN)	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
Occupied	Occupied by batch control	BOOL	0
OosLi	Edge transition (0-1) = Out of service, via interconnection or SFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OpenLi	1 = Open via interconnection or CFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OpenOp	1 = Open via operator	BOOL	0
OS_Perm	I/O for operator control permissions (Page 346)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
OS1Perm	I/O for operator control permissions (Page 346)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
PID_On	1 = PID mode on	BOOL	0
PropFacSP	Applying the P action to the feedback [0..1]. 0 = P action fully in feedback	REAL	1.0
PropSel	1 = Activate P action	BOOL	1
PulseTime	Minimum pulse duration [s]	REAL	1.0
PV_A_DC	Delay time for incoming PV alarms [s]	REAL	0.0
PV_A_DG	Delay time for outgoing PV alarms [s]	REAL	0.0
PV_AH_En	1 = Enable PV alarm limit (high)	BOOL	1
PV_AH_Lim	PV alarm limit (high)	REAL	95.0

Parameter	Description	Type	Default
PV_AH_MsgEn	1 = Enable message for PV alarm (high)	BOOL	1
PV_AL_En	1 = Enable PV alarm limit (low)	BOOL	1
PV_AL_Lim	PV alarm limit (low)	REAL	5.0
PV_AL_MsgEn	1 = Enable message for PV alarm (low)	BOOL	1
PV_Hyst	Hysteresis for PV alarm, warning and tolerance limits	REAL	1.0
PV_OpScale	Limit for scale in PV bar graph of faceplate	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
PV_T_DC	Delay time for incoming PV tolerance messages [s]	REAL	0.0
PV_T_DG	Delay time for outgoing PV tolerance messages [s]	REAL	0.0
PV_TH_En	1 = Enable PV tolerance limit (high)	BOOL	0
PV_TH_Lim	PV tolerance message limit (high)	REAL	85.0
PV_TH_MsgEn	1 = Enable message for PV tolerance message (high)	BOOL	1
PV_TL_En	1 = Enable PV tolerance limit (low)	BOOL	0
PV_TL_Lim	PV tolerance message limit (low)	REAL	15.0
PV_TL_MsgEn	1 = Enable message for PV tolerance message (low)	BOOL	1
PV_Unit	Unit of measure for process value	INT	1001; ^{O}C
PV_W_DC	Delay time for incoming PV warnings [s]	REAL	0.0
PV_W_DG	Delay time for outgoing PV warnings [s]	REAL	0.0
PV_WH_En	1 = Enable PV warning limit (high)	BOOL	1
PV_WH_Lim	PV warning limit (high)	REAL	90.0
PV_WH_MsgEn	1 = Enable message for PV warning (high)	BOOL	1
PV_WL_En	1 = Enable PV warning limit (low)	BOOL	1
PV_WL_Lim	PV warning limit (low)	REAL	10.0
PV_WL_MsgEn	1 = Enable message for PV warning (low)	BOOL	1
RackNo	Rack number	BYTE	16#FF
RatioFac	Ratio factor	REAL	0.0

Parameter	Description	Type	Default
Rbk	Position feedback for display on OS	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
RbkHyst	Alarm hysteresis for position feedback	REAL	1.0
RbkWH_En	1 = Enable warning (high) for position feedback	BOOL	1
RbkWH_Lim	Limit for position feedback of warning (high)	REAL	100.0
RbkWH_MsgEn	1 = Enable messages for warning (high) for position feedback	BOOL	1
RbkWL_En	1 = Enable warning (low) for position feedback	BOOL	1
RbkWL_Lim	Limit for position feedback of warning (low)	REAL	0.0
RbkWL_MsgEn	1 = Enable messages for warning (low) for position feedback	BOOL	1
RefStdDevIn	Reference value of PV standard deviation (sigma) in defined "good" state of control loop	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#78
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SavePar	1 = Save PID controller parameters	BOOL	0
SelFp1	Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
SimOn	1 = Simulation on	BOOL	0
SimPV	Process value used for SimOn = 1	REAL	0.0
SimRbk	Position feedback used for SimOn = 1	REAL	0.0
SlotNo	Slot number	BYTE	16#FF
SP_DnRaLim	Limit (low) for ramp of setpoint [SP_Unit/s]	REAL	100.0

Parameter	Description	Type	Default
SP_ExHiLim	Limit (high) for external setpoint	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
SP_ExLoLim	Limit (low) for external setpoint	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_Ext	External setpoint of - (to interconnection)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_ExtLi	1 = Select external setpoint (via interconnection)	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_ExtOp	1 = Select external setpoint (via operator)	BOOL	0
SP_InHiLim	Limit (high) of internal setpoint	REAL	100.0
SP_InLoLim	Limit (low) of internal setpoint	REAL	0.0
SP_Int	Internal setpoint for operation	REAL	0.0
SP_IntLi	1 = Select internal setpoint (via interconnection)	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_IntOp	1 = Select internal setpoint (via operator)	BOOL	0
SP_LiOp	Select setpoint source (internal/external): 1 = Via interconnection 0 = Via operator	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_RateOn	1 = Activate limitation of setpoint gradients	BOOL	0
SP_RmpModTime	1 = Use time (SP_RmpTime) for setpoint ramp, 0 = Use gradient	BOOL	0
SP_RmpOn	1 = Activate setpoint ramp to target setpoint SP_RmpTarget	BOOL	0
SP_RmpTarget	Target setpoint for setpoint ramp	REAL	0.0
SP_RmpTime	Time for setpoint ramp [s] from current SP up to SP_RmpTarget	REAL	0.0
SP_TrkExt	1 = Bumpless switchover from external to internal setpoint active	BOOL	1

Parameter	Description	Type	Default
SP_TrkPV	1 = Setpoint follows PV in manual mode and with tracking	BOOL	0
SP_UpRaLim	Gradient limit (high) for setpoint [SP_Unit/s]	REAL	100.0
StepNo	Batch step number	DWORD	16#00000000
StopLi	1 = Stop via interconnection or CFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StopOp	1 = Stop via operator	BOOL	0
SubN1_ID	ID of the primary DP master system	BYTE	16#FF
SubN2_ID	ID of the redundant DP master system	BYTE	16#FF
TD	Derivative action time [s] TD.ST = 16#FF: Operable in faceplate	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
TI	Integral action time [s] TI.ST = 16#FF: Operable in faceplate	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#FF
TunC_MV	Delta manipulated variable for cooling optimization	REAL	-20.0
TunC_Start	Start cooling optimization	BOOL	0
TunD_MV	Delta manipulated variable for process excitation	REAL	20.0
TunOn	Enable controller optimization	BOOL	0
TunStart	Start controller optimization	BOOL	0
UndoPar	Undo controller parameter changes	BOOL	0
UserStatus	Freely assignable bits for use in faceplate	BYTE	16#00

In/out parameters

Parameter	Description	Type	Default
EnCoord	Current coordination number	STRUCT • CO_ACT : INT	- • 0

Output parameters

Parameter	Description	Type	Default
AdvCoAct	1 = Program mode is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AdvCoRdy	1 = Program mode available	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AutAct	1 = Automatic mode is active 0 = Manual mode is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ChFM_Err	1 = Channel error on the module	BOOL	0
Close	Control output: 1 = Closed is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
EnCoNum	Coordination number	BYTE	16#0
ENO	1 = Block algorithm completed without errors	BOOL	0
ER	Control error	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
ER_AH_Act	1 = Alarm limit (high) for control error violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ER_AL_Act	1 = Alarm limit (low) for control error violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ErrorNum	Output of present error number, for error numbers that can be output by this block, see FmTemp error handling (Page 356)	INT	-1
FFwdOut	Disturbance variable generated in the FM	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

Parameter	Description	Type	Default
FbkClsOut	1 = Low limit stop of the position feedback reached	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FbkOpnOut	1 = High limit stop of the position feedback reached	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
GainEff	Effective proportional gain, depends on Gain and NormPV	REAL	1.0
ManAct	1 = Manual mode active	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
ManARW_Act	1 = Tracking mode or anti-reset windup by secondary controller	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ManHiOut	Limit (high) for manual mode, corresponds to the input parameter ManHiLim	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
ManLoOut	Limit (low) for manual mode, corresponds to the input parameter ManLoLim	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
ModErr	1 = Module error	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_Release	Maintenance release: 1 = Release for OS operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MsgAckn1	Alarm acknowledgement status 1 (output STATUS of first ALARM_8P)	WORD	16#0000
MsgAckn2	Alarm acknowledgement status 2 (output STATUS of second ALARM_8P)	WORD	16#0000
MsgErr1	Alarm error 1 (output ERROR of first ALARM_8P)	BOOL	0
MsgErr2	Alarm error 2 (output ERROR of second ALARM_8P)	BOOL	0
MsgStat1	Alarm status 1 (output ERROR of first ALARM_8P)	WORD	16#0000
MsgStat2	Alarm status 2 (output ERROR of second ALARM_8P)	WORD	16#0000
MV	Manipulated variable	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

Parameter	Description	Type	Default
MV_FmTrkAct	1 = Tracking a manipulated variable in the FM is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_HiAct	1 = Limit (high) of manipulated variable violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_LoAct	1 = Limit (low) of manipulated variable violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_SafeAct	1 = Safety manipulated variable of the FM is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_SpliA	Manipulated variable A of split-range function	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV_SpliB	Manipulated variable B of split-range function	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV_UnitOut	Unit of measure for manipulated variable, for interconnecting to MV_Unit input parameter of the ConPerMon block	INT	0
OosAct	1 = Block is "out of service"	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Open	Control output 1 = Open is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
OS1PermLog	Display of OS1Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS1PermOut	Display of OS_Perm1	DWORD	16#FFFFFFFF
ParFM_Err	1 = Direct parameter-assignment error of the FM or input Channel configured incorrectly	BOOL	0
PerAccErr	1 = I/O access error	BOOL	0
Phase	Phase of auto-tuning [0..7]	INT	0
PV_AH_Act	1 = PV alarm (high) active	STRUCT • Value: BOOL • ST: BYTE	- 0 16#80

Parameter	Description	Type	Default
PV_AL_Act	1 = PV alarm (low) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_Out	Output for process value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_TH_Act	1 = PV tolerance message (high) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_TL_Act	1 = PV tolerance message (low) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_ToleHi	Limit (high) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_ToleLo	Limit (low) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_UnitOut	Unit of measure for process value, for interconnecting with PV_Unit input parameter of the ConPerMon block	INT	0
PV_WH_Act	1 = PV warning (high) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_WL_Act	1 = PV warning (low) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RbkOut	Output for position feedback	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
RbkWH_Act	1 = Warning (high) for position feedback active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RbkWL_Act	1 = Warning (low) for position feedback active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RetValue	Return value of WRREC/RDREC	WORD	16#0
SP	Setpoint used by controller	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

Parameter	Description	Type	Default
SP_DnRaAct	1 = Negative gradient limiting of setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_ExHiAct	1 = Limit (high) for external setpoint has been reached	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_ExLoAct	1 = Limit (low) for external setpoint has been reached	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_ExtAct	1 = External setpoint is active 0 = Internal setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_ExtOut	External setpoint, corresponds to input parameter SP_Ext	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_InHiOut	Limit (high) for SP_Int corresponds to input parameter SP_InHiLim	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
SP_InLoOut	Limit (low) for SP_Int corresponds to input parameter SP_InLoLim	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_UpRaAct	Positive gradient limiting of setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SplitRange	1 = Split-range function has been activated	BOOL	0
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word 1 (Page 341)	DWORD	16#00000000
Status2	Status word 2 (Page 341)	DWORD	16#00000000
StatusC	Status of cooling optimization	INT	16#00
StatusD	Status of controller design	INT	16#00
StatusH	Status of heating optimization	INT	16#00
StepCon	1 = Step controller	BOOL	0
Stop	Control output 1 =Stopped is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SumMsgAct	1 = Active process alarm	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
TunAct	1 = Optimization running	BOOL	0

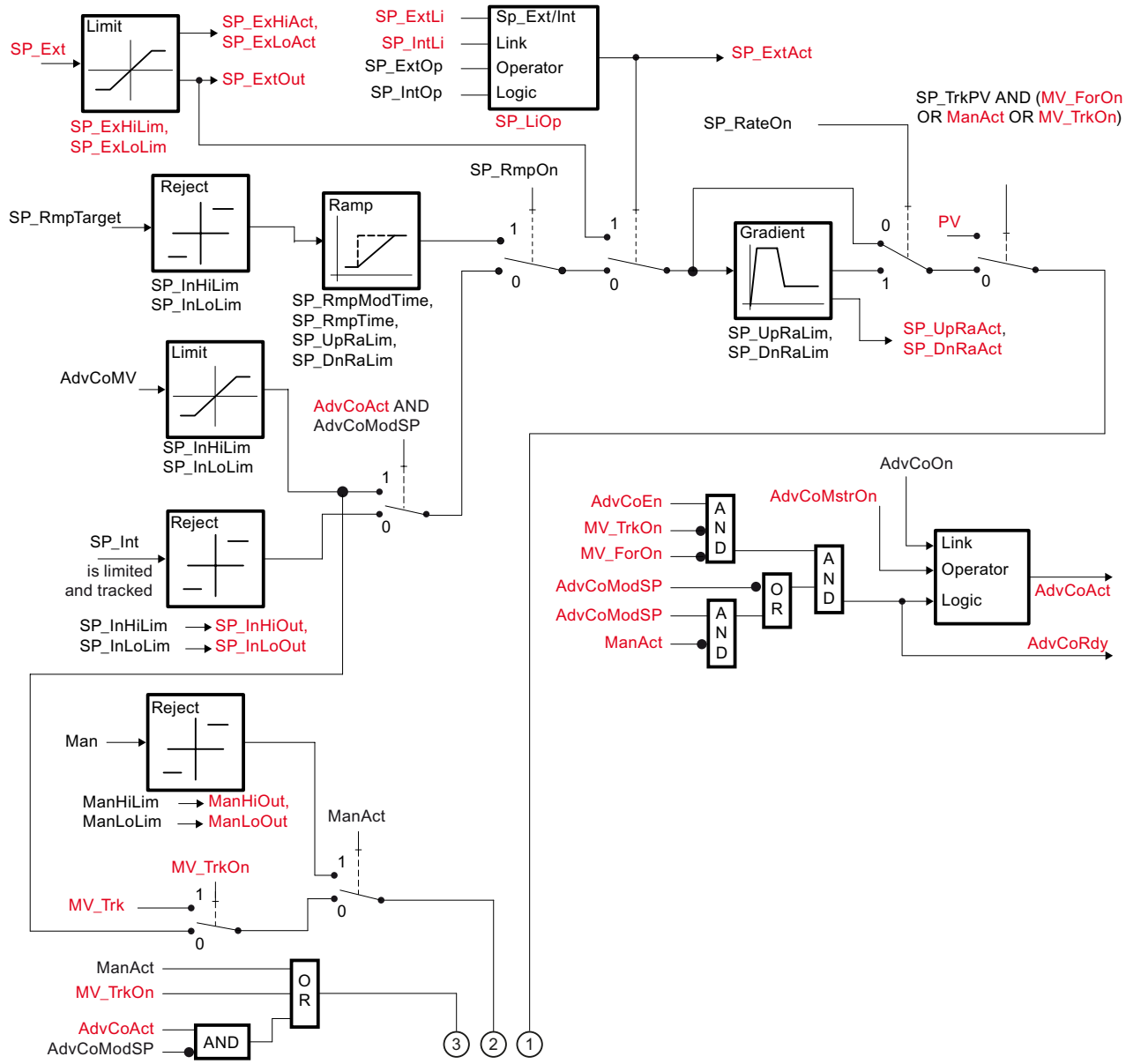
Parameter	Description	Type	Default
WithRbk	Controller type: 0 = Step controller without position feedback 1 = Step controller with position feedback	BOOL	0
ZoneTun	Controller channels grouped in one zone for parallel optimization	WORD	16#0

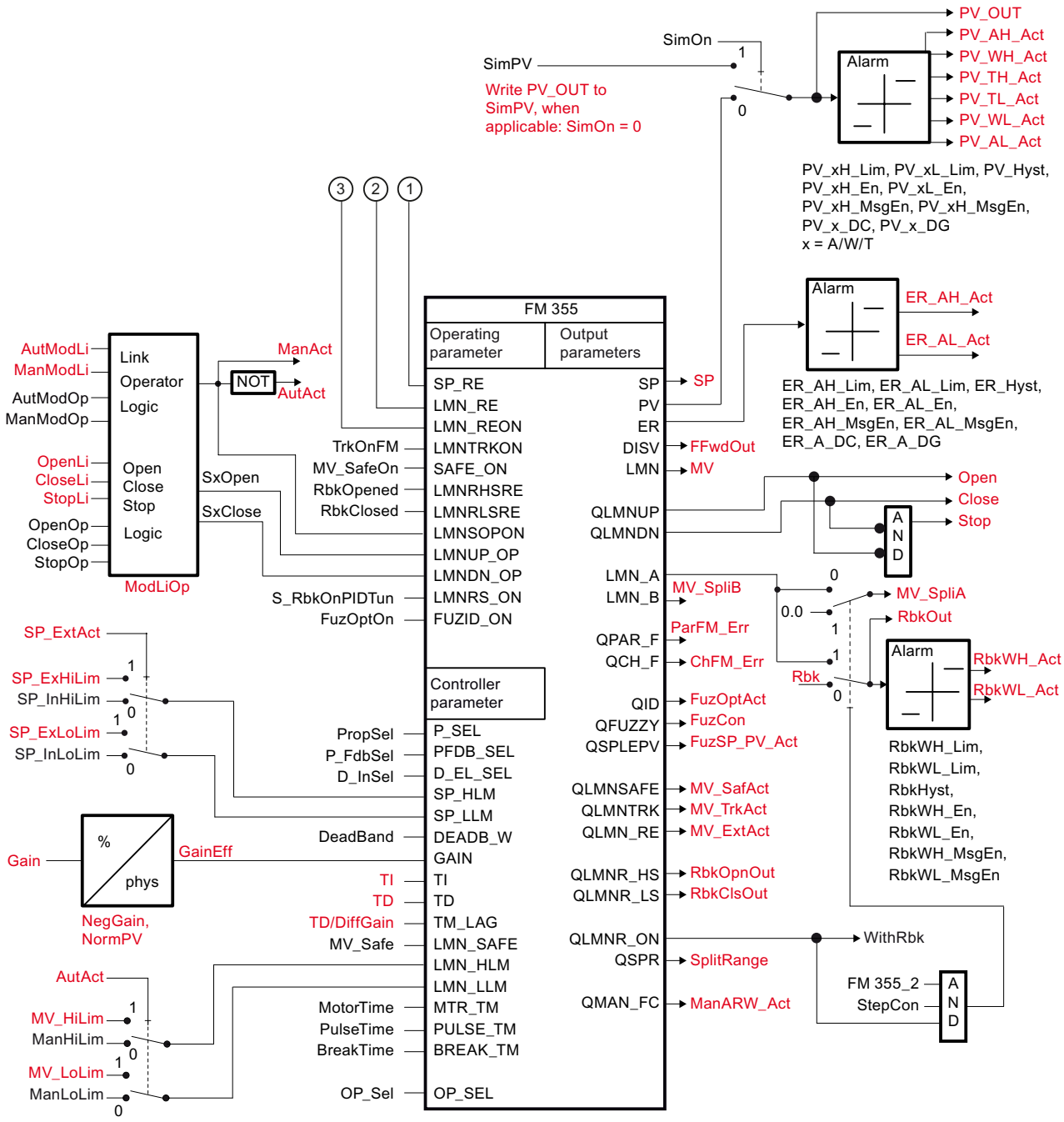
See also

FmTemp messaging (Page 358)
FmTemp modes (Page 345)
FmTemp block diagram (Page 376)

4.3.7 FmTemp block diagram

FmTemp block diagram





See also

Description of FmTemp (Page 341)

FmTemp modes (Page 345)

FmTemp functions (Page 346)

FmTemp error handling (Page 356)

FmTemp messaging (Page 358)

FmTemp I/Os (Page 361)

4.3.8 Operator control and monitoring

4.3.8.1 FmTemp views

Views of the FmTemp block

The block FmTemp provides the following views:

- Standard view of FM controllers (analog) (Page 131)
- Standard view of FM controllers (pulse controller) (Page 135)
- Standard view of FM controllers (step controller with position feedback) (Page 139)
- Standard view of FM controllers (step controller without position feedback) (Page 143)
- Message view (Page 169)
- Limit value view of FM controllers (Page 157)
- Trend view (Page 172)
- Parameter view of FM controllers (Page 153)
- Preview of FM controllers (Page 165)
- Memo view (Page 171)
- Batch view (Page 170)
- Block icons for PID and FM controller (Page 181)

Refer to the Structure of the faceplate (Page 123) and Block icon structure (Page 174) sections for general information about the faceplate and block icon.

4.4 GainSched - Adapting parameter values for a PID controller

4.4.1 Description of GainSched

Object name (type + number) and family

Type + number: FB 1820

Family: Control

Area of application for GainSched

The block is used for the following applications:

- Continuous adaptation of the parameter values of a PID controller to the current operating point of a non-linear process
- Controller gain
- Integral action time
- Derivative action time

How it works

If your process requires different PID controller parameters due to its non-linear response at different operating points, you can store optimum parameter sets for up to three different operating points in theGainSched block in the form of a table ("timetable"). The current operating point is represented by a continuously measurable variable X, typically by the actual value of the controller itself. The block ensures that the suitable optimum parameters `Gain`(j), `TI`(j) and `TD`(j) are made available to the controller for each operating point X(j).

If the process is between two operating points, the parameters are calculated by linear interpolation between the optimum values of the two nearest operating points. This allows a bumpless, continuous adaptation of the controller parameters while the process moves from one operating point to another.

The block should be considered a supplementary function for a PID controller to improve the control performance of the PID controller in non-linear processes. The GainSched faceplate is called from the parameter view of the corresponding PID controller using the "Gain scheduler" button.

In contrast to all other function blocks, the GainSched block is implemented as a CFC chart and is generated with the "Compile chart as block type" function. The source chart "FbGainSchedLim" is supplied with the library so that you have more options open to you:

- You can use the precompiled function block GainSched from the library if the standard functionality is adequate for your needs.
- If you require special additional functions for gain scheduling in your application (for example more than three operating points, additional logic functions for selecting the parameters), you will need to modify the CFC source chart and compile it as a block type with a different FB number.

If the current value of input parameter X is below the lowest value X1 in the table or above the highest value X3, precisely the controller parameters which are specified at the relevant boundary point X1 or X3 in the table are output.

Configuration

The GainSched block is placed in the same CFC chart as the assigned controller and interconnected with it as shown in the corresponding template: The output parameters Link2Gain, Link2TI and Link2TD are connected to the inputs Gain, TI and TD of the PID controller. The X input of GainSched is supplied with the measured value for the operating point, typically with the same value as PV of the controller.

You can open the standard view for the GainSched block from the parameter view of a controller (for example PIDConL). Additional information on this topic is available in the section Calling further faceplates (Page 43).

To specify the parameters for gain scheduling, run separate controller optimizations at each of the intended operating points, for example with a tool such as the PID tuner. Use amplitudes as small as possible to excite the process to capture the approximately linear response in the area of the operating point under investigation. The optimum parameter values calculated by the PID tuner are entered in the relevant row belonging to the operating point in the table of the GainSched block. The table is clearly displayed in the standard view of the faceplate. Make sure that the numeric values are also permanently stored in the data management of the engineering system by reading back the numeric values of the parameters from AS to the ES or enter them manually at the inputs of the CFC block.

For the GainSched block, the Advanced Process Library contains a template for process tag types as an example with an application scenario for this block.

Example of process tag types:

- PID - control with operating-point-oriented parameter control (GainScheduling) (Page 1431)

Gain scheduling for batch processes

A typical area of application for gain scheduling is in batch processes that, in contrast to continuous processes, cannot be linearized around a fixed operating point because they need to be moved backwards and forwards between different operating points during the course of the batch. Here, there are three application scenarios:

- The controller parameters depend on a single continuously measurable variable that is representative of the operating point, for example, the reactor temperature. This is the normal use case for the GainSched block: The management of the controller parameters is handled in the block and is independent of batch recipes.
- The controller parameters depend on a continuously measurable variable that is representative of the operating point, but there is also a dependency on the materials used in the reaction. Suitable parameter sets for gain scheduling can then be anchored in the recipe and transferred by SIMATIC Batch to the GainSched block.
- The controller parameters depend only the current phase of the batch. They can then be written directly from the Batch package to the PID controller and no gain scheduling block is necessary. The disadvantage of this is that there is bump in the controller parameters at the transfer from one phase to the next. The controller should be put into manual mode temporarily at the time of the transfer to avoid a bump in the manipulated variable.
- the recipe only specifies which of the controller parameter sets 1 to 3 is currently required from the GainSched block. The numerical values of the parameter, however, are not anchored in the recipe. In this case input variable `X` of the GainSched block can then be used as the number of the required data record and assigned by the recipe instead of being linked with a measurable process variable. In this case, there are only three discrete values for `X` and the precautions against a change of controller parameters with bump outlined above must be taken because the interpolation abilities of the GainSched block are not used.

In general, it is not necessary to manage the batch parameters (batch ID, batch name etc.) in theGainSched block because no separate messages are generated and there is always a 1:1 relationship with a controller block that recognizes the batch parameters.

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for `Status` parameter

This block does not provide the `Status` parameter.

See also

GainSched functions (Page 383)
GainSched messaging (Page 384)
GainSched I/Os (Page 385)
GainSched block diagram (Page 388)
GainSched error handling (Page 383)
GainSched modes (Page 382)

4.4.2 GainSched modes

GainSched modes

The block can be operated using the following modes:

- Automatic mode
- Manual mode

Automatic mode

In automatic mode (`ManParOn = 0`), the controller parameters correspond to the settings in the automatic area of the parameter view determined through a polygon.

Manual mode

In manual mode (`ManParOn = 1`), the controller parameters correspond to the settings in the manual area.

See also

GainSched block diagram (Page 388)
GainSched I/Os (Page 385)
GainSched messaging (Page 384)
GainSched error handling (Page 383)
GainSched functions (Page 383)
Description of GainSched (Page 379)

4.4.3 GainSched functions

Functions of GainSched

The functions for this block are listed below.

Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 53).

See also

Description of GainSched (Page 379)
GainSched messaging (Page 384)
GainSched I/Os (Page 385)
GainSched block diagram (Page 388)
GainSched error handling (Page 383)
GainSched modes (Page 382)
GainSched standard view (Page 389)

4.4.4 GainSched error handling

GainSched troubleshooting

The block does not report any errors.

See also

GainSched block diagram (Page 388)
GainSched I/Os (Page 385)
GainSched messaging (Page 384)
GainSched functions (Page 383)
GainSched modes (Page 382)
Description of GainSched (Page 379)

4.4.5 GainSched messaging

Messaging

This block does not have any message functionality.

See also

Description of GainSched (Page 379)
GainSched functions (Page 383)
GainSched I/Os (Page 385)
GainSched block diagram (Page 388)
GainSched error handling (Page 383)
GainSched modes (Page 382)

4.4.6 GainSched I/Os

GainSched I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
Gain1	PID gain for operating point 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Gain2	PID gain for operating point 2	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Gain3	PID gain for operating point 3	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
GainOp	PID gain: Input for manual mode	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
ManParOn	1 = input of PID parameters in manual mode 0 = use the planned controller parameters from the table	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
TD_Op	PID derivative action time [s]: Manual input for the operator	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
TD1	PID derivative action time [s] for operating point 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
TD2	PID derivative action time [s] for operating point 2	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
TD3	PID derivative action time [s] for operating point 3	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
TI_Op	PID integral action time [s]: Manual input for the operator	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
TI1	PID integral action time [s] for operating point 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

Parameter	Description	Type	Default
TI2	PID integral action time [s] for operating point 2	STRUCT • Value: REAL • ST: BYTE	1- • 0.0 • 16#80
TI3	PID integral action time [s] for operating point 3	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
X	Process value that defines the operating point	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
X_1	Operating point 1 (support point) for X	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
X_2	Operating point 2 (support point) for X	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
X_3	Operating point 3 (support point) for X	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
X_Unit	Unit of measure for the operating point	INT	1001

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
Link2Gain	Calculated controller gain	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Link2TD	Calculated integral action time	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Link2TI	Calculated derivative action time	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
OS_PermLog	Parameter used to hide operator controls in the faceplate	DWORD	16#FFFFFFFF

See also

Description of GainSched (Page 379)
GainSched functions (Page 383)
GainSched messaging (Page 384)
GainSched block diagram (Page 388)
GainSched error handling (Page 383)
GainSched modes (Page 382)

4.4.7 GainSched block diagram

GainSched block diagram

A block diagram is not provided for this block.

See also

GainSched I/Os (Page 385)
GainSched messaging (Page 384)
GainSched error handling (Page 383)
GainSched functions (Page 383)
GainSched modes (Page 382)
Description of GainSched (Page 379)

4.4.8 Operator control and monitoring

4.4.8.1 GainSched views

Views of the GainSched block

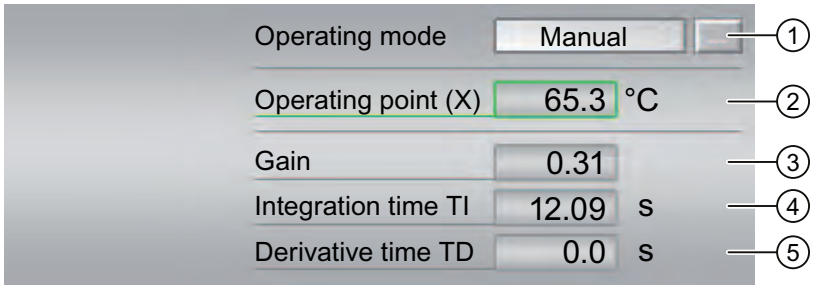
The block GainSched provides the following views:

- GainSched standard view (Page 389)
- GainSched parameter view (Page 390)
- Memo view (Page 171)

Refer to the Structure of the faceplate (Page 123) and Block icon structure (Page 174) sections for general information about the faceplate and block icon.

4.4.8.2 GainSched standard view

GainSched standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual
- Automatic

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

In manual mode, you can specify the values on the OS in the parameter view of this block; they are then output directly via the corresponding output parameters.

In automatic mode, an interpolation is performed through the interpolation points, which can also be specified in the parameter view.

(2) Displaying the operating point (X)

Currently used operating point.

(3) Displaying the gain

The controller gain currently output at the `Link2Gain` output parameter.

(4) Displaying the integration time TI

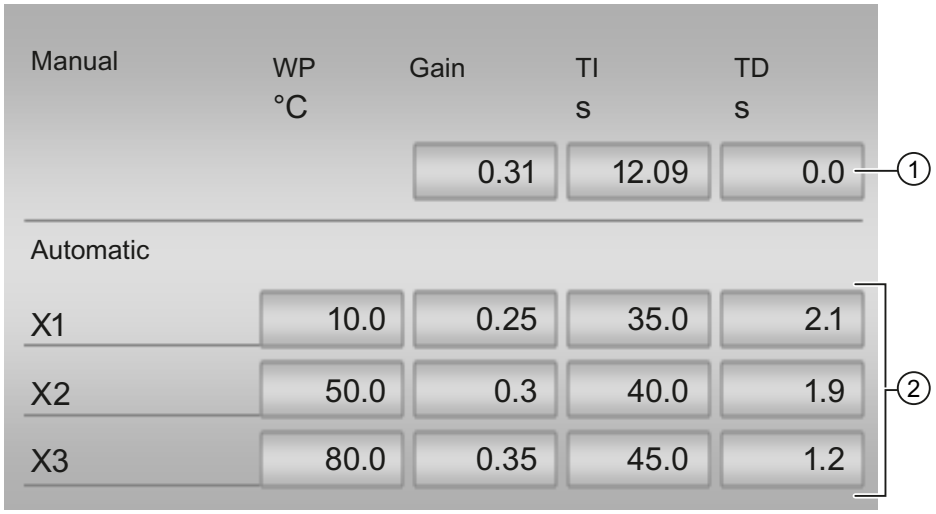
Integration time currently output at the `Link2TI` output parameter.

(5) Display and change the derivative time TD

Derivative time currently output at the `Link2TD` output parameter.

4.4.8.3 GainSched parameter view

GainSched parameter view



(1) Displaying and changing the values for the controller parameters in manual mode

This is where you enter the values for the parameters to be used in manual mode at the corresponding output parameters of the block:

- Gain (input parameter `GainOp`)
- TI (integration time, input parameter `TI_Op`)
- TD (derivative time, input parameter `TD_Op`)

Refer to the section Changing values (Page 129) for information on changing the values.

(2) Displaying and changing the values for the controller parameters in automatic mode

This is where you enter the values for the parameters to be used in automatic mode for the interpolation (max. 3 values):

- X1 (operating point 1, input parameter `X_1`)
- X2 (operating point 2, input parameter `X_2`)
- X3 (operating point 3, input parameter `X_3`)
- Gain (input parameters `Gain1` to `Gain3`)
- TI (integration time, input parameters `TI1` to `TI3`)
- TD (derivative time, input parameters `TD1` to `TD3`)

Refer to the section Changing values (Page 129) for information on changing the values.

4.5 ModPreCon - Model predictive controller

4.5.1 Description of ModPreCon

Object name (type + number) and family

Type + number: FB 1843

Family: Control

Area of application for ModPreCon

The block is used in much the same way as a ModPreCon block for the following applications:

- Fixed setpoint control
- Cascade control
- Ratio control
- Split-range control

In contrast to the PID controllers, this is a multivariable controller.

Method of operation and area of application

The block is used for multivariable control (Page 1469) of dynamic processes. It can handle up to four dependent manipulated and controlled variables as well as a measurable disturbance variable.

In special situations, the ModPreCon block can also be used for particularly difficult dynamic, single-variable controls. It is better than a PID controller, for example, in systems with non-minimum phase (Page 1469) or a strongly oscillating response.

The ModPreCon algorithm only works for stable processes with a step response that settles to a fixed value in a finite time. If the process is unstable or includes an integrator (tank level control for example), the corresponding partial transfer function must be stabilized with a secondary controller.

For integrating plugging, a simple P controller (proportional component only) is adequate as a secondary controller.

Note on the area of application of the controller: Longer run times

The run time of the multivariable controller is basically longer that that for PID controllers, because a great many matrices have to be multiplied in the algorithm and optimization of the control algorithm is performed in each sample step. The run time is also determined by the process and manipulated variables in the control algorithm. This is why the multivariable controller is unsuited for rapid control and is usually used for slow, complex control tasks.

The computation time required on the CPU is compensated for by the fact that very slow sample times of > 20s are used for the typical ModPreCon applications (see Advanced control templates). The ModPreCon is then typically in OB30 and can be interrupted by faster OBs.

Operating principle

The ModPreCon block is a model-based predictive multivariable controller. It uses a mathematical model of the process dynamics including all interactions as part of the controller. This model allows the process response to be predicted over a defined period in the future, also known as the prediction horizon.

Based on this prediction, a criterion for a fit (quality)

$$J = (\vec{w} - \vec{y})^T \cdot R \cdot (\vec{w}-\vec{y}) + \Delta\vec{u}^T \cdot Q \, \Delta\vec{u}$$

is optimized (minimized) where the following applies:

- w contains the time series of the future setpoints,
- y contains the vector of the controlled variables in the future,
- Δu contains the future changes to the manipulated variable.

If you increase the weighting in the Q diagonal matrix, the controller moves its manipulated variables more cautiously resulting in a slower but more robust control action. Using the weighting factors in the R diagonal matrix, you specify the relative significance of the individual controlled variables. A higher weighting (priority) for a controlled variable means that this moves more quickly towards the setpoint and remains more accurately at the setpoint in steady state if it is not possible to achieve all setpoints precisely.

The algorithm is a variant of the DMC procedure (**D**ynamic **M**atrix **C**ontrol) in which the optimization problem is solved in the design phase ignoring the constraints. The function block itself contains the analytical solution of the optimization problem. Manipulated variable limitations, both absolute and relating to the gradients, are treated in the algorithm of the function block as hard limits that must not be violated. This means that precise setpoints or target zones for the controlled variables are taken into account as well as possible in the optimization. The target zones for the controller variables are therefore soft limits, which are maintained as well as possible although they cannot be guaranteed. Using a reference variable filter for future setpoint settings, the control action of the controller can be finely adjusted during operation.

You can achieve significant improvements in control quality when individual disturbance variables can be measured, for example variations in throughput. In this case, it is a good idea to take into account a model of the influence of this disturbance variable on the controlled variables when predicting the controlled variables so that the controller can react preemptively to such disturbances.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

After installation in CFC, follow the steps outlined below:

1. Excite the process with the controller in manual mode by applying a series of manipulated variable step changes.
2. Record the measured data with the CFC trend display and export it to an archive file.
3. Enter the number of the data block at the DB_No input parameter of the ModPreCon block. The values are adopted in the controller by restarting the block using the Restart input parameter.
4. Select the ModPreCon instance in the CFC. Start the MPC Configurator with the command Edit > MPC Configurator.
5. Using the Configurator, create an SCL source code for the user data block (DB). It contains the models and matrices required for an ModPreCon instance.
6. Compile the SCL source code in the engineering system and download it to the AS.

Note

You can find a detailed description of this procedure in the MPC Configurator help.

During controller design in the MPC Configurator, a controller cycle time and an OB sampling time are calculated and displayed. You yourself are responsible for the ModPreCon block being called in the cyclic interrupt level suitable for the OB sampling time. When necessary, slower controller cycle times will be implemented automatically in the block by an internal pulse reduction ratio function.

For the ModPreCon block, the Advanced Process Library contains templates for a process tag type as examples and there is a example project (APL_Example_xx, where xx designates the language variant) containing different application cases for this block. Several application cases are simulated in the example project and serve to explain how the block works.

Example of a process tag type:

- Model-based predictive control (ModPreCon) (Page 1447)

Application cases in example project:

- Predictive control of a 2x2 multi-variable controlled system (Page 1464)
- Predictive control of a non-linear process (Page 1465)

Startup characteristics

When the CPU starts up, the block always starts in manual mode. It is only possible to change to automatic mode when a user data block is loaded and the internal measured value memory in ModPreCon is filled with data.

Use the Feature bit Setting the startup response (Page 187) to define the startup characteristics of this block.

Status word allocation for `Status1` parameter

Status bit	Parameter
0	`Occupied`
1	`BatchEn`
2	`SimOn`
3	`OosAct.Value`
4	`OosLi.Value`
5	`AutAct.Value`
6	Not allocated
7	`ManAct.Value`
8 - 9	Not allocated
10	`MV1TrkOn.Value AND NOT (ManAct.Value OR OosAct.Value)`
11	`MV2TrkOn.Value AND NOT (ManAct.Value OR OosAct.Value)`
12	`MV3TrkOn.Value AND NOT (ManAct.Value OR OosAct.Value)`
13	`MV4TrkOn.Value AND NOT (ManAct.Value OR OosAct.Value)`
14	Not allocated
15	`DB_Loaded`
16	`DV_ModelAvailable`
17-31	Not allocated

Status word allocation for `Status2` parameter

Status bit	Parameter
0 - 30	Not allocated
31	`MS_RelOp`

See also

ModPreCon functions (Page 396)
ModPreCon messaging (Page 407)
ModPreCon I/Os (Page 408)
ModPreCon block diagram (Page 416)
ModPreCon error handling (Page 406)
ModPreCon modes (Page 395)

4.5.2 ModPreCon modes

ModPreCon operating modes

The block can be operated using the following modes:

- Automatic mode (Page 29)
- Manual mode (Page 29)
- Out of service (Page 27)

The next section provides additional block-specific information relating to the general descriptions.

The operating modes mentioned above apply to the block with all (MV1 ... MV4) control channels. It is also possible to track a single control channel, refer to the section ModPreCon functions (Page 396) for information on this.

Automatic mode

You can find general information on "Automatic mode", switching modes and bumpless switchover in the Manual and Automatic mode for control blocks (Page 29) section.

Note

In contrast to PID controllers, it is permitted to run the ModPreCon block in automatic mode without its actuating signals affecting the process because there is no risk of integrator windup.

Manual mode

You can find general information on "Manual mode", switching modes and bumpless switchover in the Manual and Automatic mode for control blocks (Page 29) section.

Out of service

You can find general information about the "Out of service" mode in the Out of service (Page 27) section.

See also

ModPreCon block diagram (Page 416)

ModPreCon I/Os (Page 408)

ModPreCon messaging (Page 407)

ModPreCon error handling (Page 406)

Description of ModPreCon (Page 391)

4.5.3 ModPreCon functions

Functions of ModPreCon

The functions for this block are listed below.

Generating and limiting the manipulated variable

The manipulated variable MV1 ... MV4 (also referred to as MVx in the following, x = 1 ... 4) can be generated as follows:

ManAct	MVxTrkOn	MVx	Limit monitoring	State
1	-	Manx	ManxHiLim ManxLoLim	Manual mode, set by the operator
0	1	MVxTrk	MVxHiLim MVxLoLim	Tracking with limitation
0	0	Automatic manipulated variable	MVxHiLim MVxLoLim	Automatic mode: Predictive controller algorithm

Remark

If the controller is in "out of service" mode, the output parameters MV1 ... MV4 are set to the last valid value in manual mode or the safety manipulated variable (SafePos1 ... SafePos4) depending on the Feature bit (Safety value of the manipulated variable effective at startup (Page 196)). Refer to the Out of service (Page 27) section for more on this.

The limited operating range (between MVxHiLim and MVxLoLim) is typically smaller in automatic mode than in manual mode. With regard to the limited range of validity of a linear process model for approximating a non-linear process response, this allows the stability of the closed control loop to be guaranteed within the setting range in automatic mode.

The gradients of the manipulated variable (changes per second) are limited to MV1RaLim to MV4RaLim in automatic mode. Gradient limitation applies both to the positive and negative directions.

Tracking and limiting a manipulated variable

The block provides the standard function Tracking and limiting a manipulated value (Page 111).

In contrast to PID controllers, tracking the manipulated variables (MV1 ... MV4) channel-by-channel is activated via input parameters MV1TrkOn to MV4TrkOn. The corresponding manipulated variable is then tracked by the interconnectable input parameters MV1Trk to MV4Trk.

Setting the setpoint internally

With this block, the setpoint must always be set internally at the SP1 to SP4 I/Os. These are normally set in the faceplate. In special situations, you can interconnect the setpoints but they can then no longer be changed using the faceplate.

Setpoint tracking in manual mode

In this situation (`SP_TrackCV = 1`), the internal setpoints `SP1` to `SP4` are tracked to the assigned controlled variables `CV1` to `CV4`. This function allows a bumpless transfer to automatic mode. After the transfer, the setpoints can be changed by the operator again.

Setpoint filters

The setpoint filter is the only way of changing the action of the predictive controller without having to create a new user data block with the MPC Configurator and reinitialize the controller. The specified time constant `PreFilt1` to `PreFilt4` of the setpoint filter can be interpreted as the required settling time of the this CV channel following a setpoint step change. As the time constant setting increases, the controller works more slowly and less aggressively. In particular, this reduces the influence of a setpoint step change in one control channel on neighboring control channels.

Internally, the ModPreCon block works with sets of future setpoints that are compared with the predicted movements of the controlled variables. Without the setpoint filter, it is assumed that the current setpoint will continue to remain valid in the future within the prediction horizon. If there is a setpoint step change, this means that the full value of the new setpoint will be required in the near future although the process cannot achieve this (according to the prediction). With the setpoint filter, an asymptotic setpoint trajectory (first order) is calculated from the current process value to the required setpoint so that the required setpoint is reached in the specified time.

Note

The setpoint filter also comes into effect without a setpoint step change if the process value deviates significantly from the setpoint due to disturbances. This means that the filter not only slows down the control action but indirectly also the response to disturbances.

The control action can only be slowed down by the setpoint filter and not accelerated; when the value is 0, the prefilter is deactivated. It is therefore advisable to set the basic controller action in the MPC Configurator with the "Manipulated variable change penalty" parameter and then to optimize this in the software using the function for simulation of the closed control loop. The software filter should then only be used for fine modification of the action in the operational system.

Simulating signals

The block provides the standard function Simulating signals (Page 93).

Selecting a unit of measure

The block provides the standard function Selecting a unit of measure (Page 53).

Control error generation and dead band

The block provides the standard function Control error generation and deadband (Page 109).

In the predictive controller, the control error as the deviation between the predicted movement of the process (starting at the current value of the controlled variable CV1 to CV4) and future setpoint settings (ending at SP1 to SP4) is generated over the entire prediction horizon for each control channel and used to calculate the manipulated variable.

In principle, the effect of the dead bands SP1DeadBand to SP4Deadband is the same as in a PID controller but extends over the entire future prediction horizon. In other words, if, for example, the predicted controlled variable CV1 in the entire prediction horizon is within the band SP1 ± SP1DeadBand, the controller sees no reason whatsoever to change any manipulated variable. These are therefore also known as CV bands. In contrast to the manipulated variable limits, these are not hard constraints that need to be adhered to at all costs.

In multivariable controllers, it is advisable to make use of the fact that from the perspective of the application only some of the controlled variables need to move to a specified setpoint exactly while others only need to remain within a defined range.

A typical example would be quality characteristics for which a tolerance band is specified. While a dead band in a PID controller tends to put stability at risk, CV bands in individual controller channels generally relieve the multivariable controller overall.

Using CV bands, the action of a soft override control can be achieved.

Use case for control error generation with dead band

As long as the pressure in a reactor remains within the set safety limits, the controller is interested only in product quality. However, as soon as the pressure threatens to leave the permitted range (in other words, in the prediction it moves towards an illegal value in the future), the pressure control cuts in. By weighting the controlled variables in the fit criterion (see MPC Configurator), the user can specify that threatened violations of the pressure limits are given a particularly high weighting.

Predictive controller algorithm

The ModPreCon block is derived from the familiar DMC algorithm (**D**ynamic **M**atrix **C**ontrol). Future changes to the manipulated variable within the control horizon are calculated according to the formula:

$$\Delta\vec{u} = \underline{C} \cdot (\vec{w} - \vec{f})$$

Where:

- w contains the time series of the future setpoints
- f contains the predicted free movement of the controlled variables (with constant manipulated variables) in the future
- C is the constant controller matrix calculated by the MPC Configurator. C includes both the process model and the weighting of the manipulated variable changes and the controlled variables from the fit criterion of the optimization.

Based on the principle of the receding horizon, only the first value is taken from the vector of the optimum manipulated variable changes over the entire control horizon and applied to the process. In the next step, the newly arrived process values are taken into account and the calculation repeated over the entire prediction horizon.

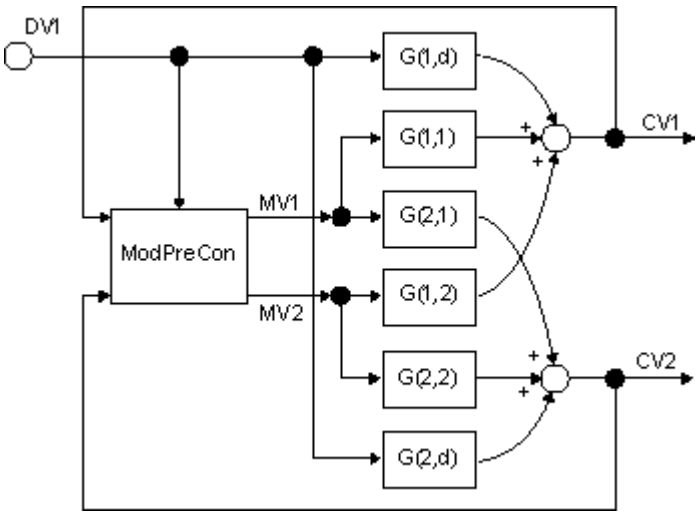
With predictive controllers, the manipulated variable changes are based on the control deviations predicted in the future, while with a PID controller, they are based on control errors of the past (possibly also integrated). This can be interpreted as a "looking ahead" strategy.

Anti-windup

When manipulated variable limits are active, anti-windup measures are taken automatically within the controller. The prediction equations use the real limited values of the manipulated variable instead of theoretically calculated values.

Model-based disturbance variable compensation

Model-based disturbance variable compensation can and should be used when a known disturbance has a strong influence on the process and its cause can be measured.



The effects of a measurable disturbance (DV1 I/O) on all controlled variables CV1 to CV4 can be estimated when the controller is put into manual mode. This means that no movements of the controlled variables whatsoever result from changes to the manipulated variable and all movements result from the disturbance variable. If the disturbance variable can be measured but cannot be actively adjusted, it may be necessary to search through a data archive to find the time segments in which the disturbance variable changed.

The identification of the transfer functions from the disturbance variable DV1 to all controlled variables CV1 to CV4 (disturbance model, in the graphic above G(1,d) and G(2,d)) is performed with the MPC Configurator and is analogous to the identification of the main transfer functions (G(1,1) to G(2,2)). The measured disturbance variable is then switched to the DV1 input of the ModPreCon block and disturbance variable compensation is activated with DV_On = 1 . As a result, the effect of the measurable disturbance is taken into account in the prediction and the controller can start countermeasures in advance before the disturbance can have a massive influence on the controlled variables. If there is no disturbance model in the user data block, the DV1 input is ignored.

Typical examples of measurable disturbance variables are inlet volumes in distillation columns or throughput of continuous reactors.

Control of square and non-square systems

In multivariable controllers, the number of manipulated variables should ideally be the same as the number of controlled variables. This is known as a "square system". As long as constraints to not influence operation, the controller can, in principle, control to the selected setpoints.

If there are less manipulated variables than controlled variables, or individual manipulated variables have reached their limits, there is no freedom in the control problem. This means that it is not possible for all setpoints to be reached exactly.

The ModPreCon algorithm then finds a compromise that can be influenced by the selection of controlled value weights (priorities) in the MPC Configurator: Controlled variables with higher priority will have lower control deviations.

Note

Since the ModPreCon block is a lean predictive controller algorithm without online optimization, there can be no general guarantee that the compromise found is optimum in a mathematical sense; in other words, it is the minimum of the fit function taking into account the manipulated variable limits. In most practical situations, however, the controller finds sensible compromises.

If there are more manipulated variables than controlled variables or if some of the controlled variables are already within their setpoint bands, there is surplus freedom in the control problem. A lean predictive controller algorithm, however, cannot recognize this situation explicitly and use the free manipulated variables for optimization. The ModPreCon block therefore moves all manipulated variables to values that meet the aims in terms of controlled variables and then leaves them there. In some situations, however, it can be useful to provide the controller with more manipulated variables than controlled variables, for example when the effect of individual manipulated variables is too restricted.

Another approach is to define the excess manipulated variables as pseudo controlled variables at the same time. You do this by assigning a setpoint with low priority to the pseudo controlled variables. The controller then attempts to achieve the important control aims as first priority and, at the same time, attempts to reach certain ideal values for the individual manipulated variables.

Control of linear and non-linear systems

The ModPreCon algorithm is based on a linear, time invariant process model. As a result, in much the same way as a PID controller, it is suitable above all for controlling non-linear systems around a fixed operating point.

Again analogous to the PID controller, there are, however, several possibilities with which the area of application can be extended with non-linear systems:

Compensation functions between controller and controlled system:

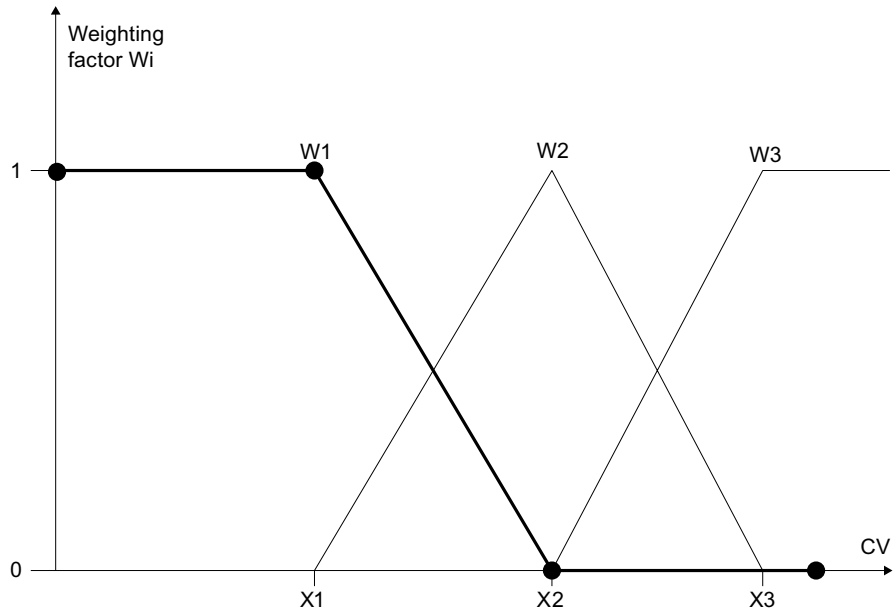
It is, for example, possible to compensate the effect of a non-linear valve characteristic curve using a polygon block between the MV output and the control input of the valve block. Care must be taken when implementing the manipulated variable limits. In the same way, the effect of a non-linearity at the output of the controlled system (for example a sensor characteristic curve) can be compensated by a polygon block before the CV input of the controller. Remember that the corresponding SP must also be transformed accordingly. In both cases, the compensation functions become part of the controlled system from the perspective of the controller. The aim is always to keep the overall response of the controlled system consisting of process and compensation elements as linear as possible.

Multimodel control:

This approach is related to the basic idea of operating-point-based parameter control with PID controllers. Since the model parameter of the ModPreCon block cannot be modified in runtime, however, the control strategy for selecting the suitable parameter set becomes a control strategy for selecting the suitable model.

Several ModPreCon instances with different models for different operating points run at the same time. The local optimal models are determined by starting the process at the various operating points with small amplitudes, so that only the reaction of the non-linear process in the ambience of this operating point is registered.

The final manipulated variable for each manipulated variable is formed as a weighted mean value of the manipulated variables proposed by the controller instances. (It is recommended that experiments for starting the process for the MPC Configurator are only performed after implementing the functions for adding the manipulated variables, in order to ensure that the same conditions applicable when the model is in operation actually take effect.)



The weighting factors 0 ... 1 are formed in the same way as the membership functions known from fuzzy logic so that the sum of all weights is always one and each controller has the highest weighting at its own operating point. A polygon with 4 or 5 interpolation points is used to calculating each individual weighting factor. The weighting factors are calculated based on a specific measurable PV variable of the process, which is representative for the operating point of the process. This can be one of the CVx controlled variables, although it does not have to be. The abscissa of the interpolation points of all polygons is selected in such a way that they cover the entire value range of PV in order to avoid extrapolation errors.

One should note in this regard that the only non-linear effects in the full multi-variable control loop that can be modeled are those that correlate exactly to a representative PV variable. This approach is therefore not suitable for cases in which individual partial transfer functions demonstrate non-linear effects that depend on various, totally independent variables.

To ensure the stability of the overall control loop, all subcontrollers must be at least stable at all operating points. In contrast to PID controllers however, an MPC is not affected by windup problems if it temporarily runs in automatic mode but cannot intervene in the real process (weighting factor zero).

One of the controller instances is defined as the main controller and shown in the operator faceplate on the OS. All others are connected in such a way that they adopt the operating mode (manual or automatic) and the setpoints from the main controller. The manual manipulated variables are passed to the secondary controller via the tracking inputs. This means that no operator intervention is required on secondary controllers. (However during experiments with the process start for the MPC Configurator, only the manipulated variable of the controller currently under consideration should be connected to the process, while the others should keep their manipulated variables constant. Interconnect the tracking inputs only after performing the experiments.)

You can find an example for multi-model controlling in the example project of Advanced Process Library under Predictive control of a non-linear process (Page 1465).

Trajectory control:

This approach neatly combines the advantages of an open loop controller (Feedforward Control) with those of a closed loop controller with process value feedback (Closed Loop Control). The controller follows a previously optimized trajectory of setpoints and manipulated variables; in other words, it only needs to compensate small deviations between the stored trajectory and the current plant state. A trajectory is an optimum series of manipulated variables over time and the process values that match them. The required manipulated variables are read into the ModPreCon block via the inputs MV1Traj to MV4Traj and added to the values of the manipulated variable calculated by the algorithm (in automatic mode only). Among other things, the advantage of this is that the effective manipulated variable acting on the process can be configured and is limited to the sum of the trajectory and controller action. The process values from the trajectory are switched to the corresponding setpoint inputs SP1 to SP4 of the controller. As long as the process reacts exactly as planned in the trajectory, it will respond to the series of manipulated variables from the trajectory with the corresponding series of process values and the control deviation is zero. It is generally known that a non-linear dynamic process can linearized around a fixed operating point or a steady state of the system. It is also possible to linearize it around a trajectory.

Forming the signal status for blocks

The block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `DV1.ST`
- `CV1.ST`
- `CV2.ST`
- `CV3.ST`
- `CV4.ST`

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions which are provided by the `Feature` parameter in chapter Functions that can be set via the Feature I/O (Page 186). The following functionality is available for this block at the relevant bits:

Bit	Function
1	Reaction to the out of service mode (Page 199)
2	Resetting the commands for changing the mode (Page 193)
4	Setting switch or button mode (Page 195)
15	Safety manipulated variable with "out of service" operating mode in effect (Page 196)
16	Safety value of the manipulated variable effective at startup (Page 196)

Operator control permissions via parameter `OS_Perm`

The block has the following operator control permissions (Page 45) for the parameter `OS_Perm`:

Bit	Function
0	1 = Operator can switch to automatic mode
1	Not allocated
2	1 = Operator can switch to out of service mode
3	Not allocated
4	Not allocated
5	1 = Operator can change setpoint 1
6	1 = Operator can change the manipulated variables of all channels
7	1 = Operator can change operating high limits of the setpoints for all channels
8	1 = Operator can change operating low limits of the setpoints for all channels
9	1 = Operator can change setpoint 2
10	1 = Operator can change setpoint 3
11	1 = Operator can change setpoint 4
12	1 = Operator can change the setpoint filter of all channels
13-16	Not allocated

Bit	Function
17	1 = Operator can activate the track setpoint in manual mode function
18	1 = Operator can activate the model-based disturbance compensation function
19 - 22	Not allocated
23	1 = Operator can change the dead band parameter of all channels
24-25	Not allocated
26	1 = Operator can activate the simulation function
27	1 = Operator can activate the release for maint. function
28	1 = Operator can change the manipulated variable limits of all channels
29	1 = Operator can change the gradient limits of manipulated variables of all channels
30 - 31	Not allocated

Maintenance release

The block provides the standard function Maintenance release (Page 47).

Specifying the display area for process and setpoint values as well as operations

This block provides the standard function Display and operator input area for process values and setpoints (Page 42).

In contrast to PID controllers, there are no separate parameters for bar limits. The setpoints limits are used for all setpoint and actual value bars; manual limits are used as bar limits for all manipulated variable bars.

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 122).

See also

Description of ModPreCon (Page 391)

ModPreCon messaging (Page 407)

ModPreCon I/Os (Page 408)

ModPreCon block diagram (Page 416)

ModPreCon error handling (Page 406)

ModPreCon modes (Page 395)

4.5.4 ModPreCon error handling

ModPreCon troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` output parameter can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Default value when the block is installed; this message is irrelevant.
0	There is no error.
2	SampleTime < 0,001 [s]
32	The value of CV1 can no longer be displayed in the real number field or is not a number.
33	The value of CV2 can no longer be displayed in the real number field or is not a number.
34	The value of CV3 can no longer be displayed in the real number field or is not a number.
35	The value of CV4 can no longer be displayed in the real number field or is not a number.
36	The MV_Trk1 value can no longer be displayed in the real number field or is not a number.
37	The MV_Trk2 value can no longer be displayed in the real number field or is not a number.
38	The MV_Trk3 value can no longer be displayed in the real number field or is not a number.
39	The MV_Trk4 value can no longer be displayed in the real number field or is not a number.
90	The controller matrix could not be loaded from the user data block.

See also

ModPreCon block diagram (Page 416)
ModPreCon I/Os (Page 408)
ModPreCon messaging (Page 407)
ModPreCon functions (Page 396)
ModPreCon modes (Page 395)
Description of ModPreCon (Page 391)

4.5.5 ModPreCon messaging

Messaging

The block does not offer messaging.

See also

Description of ModPreCon (Page 391)
ModPreCon functions (Page 396)
ModPreCon I/Os (Page 408)
ModPreCon block diagram (Page 416)
ModPreCon error handling (Page 406)
ModPreCon modes (Page 395)
Predictive control of a 2x2 multi-variable controlled system (Page 1464)

4.5.6 ModPreCon I/Os

ModPreCon I/Os

Input parameters

Parameter	Description	Type	Default
AutModLi	1 = Automatic mode via interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AutModOp	1 = Automatic mode via operator (controlled by ModLiOp = 0)	BOOL	0
BatchEn	1 = Enabled for allocation by batch control	BOOL	0
BatchID	Batch ID	DWORD	16#00000000
BatchName	Batch name	STRING[32]	''
CV1	Control variable 1 (process value)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
CV1_Unit	Unit of measure for control variable 1 (process value)	INT	1001
CV2	Control variable 2 (process value)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
CV2_Unit	Unit of measure for control variable 2 (process value)	INT	1001
CV3	Control variable 3 (process value)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
CV3_Unit	Unit of measure for control variable 3 (process value)	INT	1001
CV4	Control variable 4 (process value)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
CV4_Unit	Unit of measure for control variable 4 (process value)	INT	1001
DB_No	Number of the data block in which the controller variable is saved.	INT	0
DV_On	1 = Activate the feedforward control from DV1	BOOL	1
DV1	Disturbance variable	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
EN	1 = Called block will be processed	BOOL	1

Parameter	Description	Type	Default
Feature	I/O for additional functions (Page 396)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
ManModLi	1 = Manual mode via interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ManModOp	1 = Manual mode by OS operator (controlled by ModLiOp = 0)	BOOL	0
ModLiOp	Operating mode switchover by: • 0 = Operator • 1 = Interconnection or SFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_RelOp	1 = Maintenance release by OS operator	BOOL	0
MV1_Unit	Unit of measure for manipulated variable 1	INT	1342
MV1HiLim	High limit of manipulated variable MV1	REAL	100.0
MV1LoLim	Low limit of manipulated variable MV1	REAL	• 0.0
MV1Man	Manual value: Operator input for setting the manipulated variable MV1 in manual mode	REAL	0.0
MV1ManHiLim	High limit of manipulated variable MV1 in manual mode	REAL	100.0
MV1ManLoLim	Low limit of manipulated variable MV1 in manual mode	REAL	0.0
MV1RaLim	Gradient limit of the manipulated variable MV1 per sampling step	REAL	100.0
MV1Traj	Trajectory value that is added to the manipulated variable MV1	REAL	0.0
MV1Trk	Tracking value for the manipulated variable MV1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV1TrkOn	1 = Tracking of manipulated variable MV1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV2_Unit	Unit of measure for manipulated variable 2	INT	1342
MV2HiLim	High limit of manipulated variable MV2	REAL	100.0
MV2LoLim	Low limit of manipulated variable MV2	REAL	0.0
MV2Man	Manual value: Operator input for setting the manipulated variable MV2 in manual mode	REAL	0.0
MV2ManHiLim	High limit of manipulated variable MV2 in manual mode	REAL	100.0
MV2ManLoLim	Low limit of manipulated variable MV2 in manual mode	REAL	0.0

Parameter	Description	Type	Default
MV2RaLim	Gradient limit of the manipulated variable MV2 per sampling step	REAL	100.0
MV2Traj	Trajectory value that is added to the manipulated variable MV2	REAL	0.0
MV2Trk	Tracking value for the manipulated variable MV2	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV2TrkOn	1 = Tracking of manipulated variable MV2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV3_Unit	Unit of measure for manipulated variable 3	INT	1342
MV3HiLim	High limit of manipulated variable MV3	REAL	100.0
MV3LoLim	Low limit of manipulated variable MV3	REAL	0.0
MV3Man	Manual value: Operator input for setting the manipulated variable MV3 in manual mode	REAL	0.0
MV3ManHiLim	High limit of manipulated variable MV3 in manual mode	REAL	100.0
MV3ManLoLim	Low limit of manipulated variable MV3 in manual mode	REAL	0.0
MV3RaLim	Gradient limit of the manipulated variable MV3 per sampling step	REAL	100.0
MV3Traj	Trajectory value that is added to the manipulated variable MV3	REAL	0.0
MV3Trk	Tracking value for the manipulated variable MV3	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV3TrkOn	1 = Tracking of manipulated variable MV3	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV4_Unit	Unit of measure for manipulated variable 4	INT	1342
MV4HiLim	High limit of manipulated variable MV4	REAL	100.0
MV4LoLim	Low limit of manipulated variable MV4	REAL	0.0
MV4Man	Manual value: Operator input for setting the manipulated variable MV4 in manual mode	REAL	0.0
MV4ManHiLim	High limit of manipulated variable MV4 in manual mode	REAL	100.0
MV4ManLoLim	Low limit of manipulated variable MV4 in manual mode	REAL	0.0
MV4RaLim	Gradient limit of the manipulated variable MV4 per sampling step	REAL	100.0
MV4Traj	Trajectory value that is added to the manipulated variable MV4	REAL	0.0

Parameter	Description	Type	Default
MV4Trk	Tracking value for the manipulated variable MV4	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV4TrkOn	1 = Tracking of manipulated variable MV4	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Occupied	1 = Allocated by SIMATIC BATCH	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC (0-1 edge transition)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OS_Perm	I/O for operator control permissions (Page 396)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
PreFilt1	Time constant [s] of the setpoint filter for setpoint SP1	REAL	0.0
PreFilt2	Time constant [s] of the setpoint filter for setpoint SP2	REAL	0.0
PreFilt3	Time constant [s] of the setpoint filter for setpoint SP3	REAL	0.0
PreFilt4	Time constant [s] of the setpoint filter for setpoint SP4	REAL	0.0
Restart	1 = Restart of the block and adoption of the data from the user block that is entered at the input parameter DB_No	BOOL	1
SafePos1	Safe position for MV1 (Page 120)	BOOL	0
SafePos2	Safe position for MV2 (Page 120)	BOOL	0
SafePos3	Safe position for MV3 (Page 120)	BOOL	0
SafePos4	Safe position for MV4 (Page 120)	BOOL	0
SampleTime	Sampling time [s] (assigned automatically)	REAL	1.0
SelFp1	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
SimCV1	Controlled variable CV1 (process value) which is used with SimOn = 1	REAL	0.0
SimCV2	Controlled variable CV2 (process value) which is used with SimOn = 1	REAL	0.0
SimCV3	Controlled variable CV3 (process value) which is used with SimOn = 1	REAL	0.0
SimCV4	Controlled variable CV4 (process value) which is used with SimOn = 1	REAL	0.0

Parameter	Description	Type	Default
SimOn	1 = Simulation on	BOOL	0
SP_TrkCV	1 = Setpoints follow the CVs in manual mode and in tracking	BOOL	0
SP1	Setpoint 1 SP1.ST=FF: Operable in faceplate	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
SP1DeadBand	Width of the dead band control of CV1	REAL	0.0
SP1HiLim	High limit for setpoint 1	REAL	100.0
SP1LoLim	Low limit for setpoint 1	REAL	0.0
SP2	Setpoint 2 SP2.ST=FF: Operable in faceplate	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
SP2DeadBand	Width of the dead band control of CV2	REAL	0.0
SP2HiLim	High limit for setpoint 2	REAL	100.0
SP2LoLim	Low limit for setpoint 2	REAL	0.0
SP3	Setpoint 3 SP3.ST=FF: Operable in faceplate	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
SP3DeadBand	Width of the dead band control of CV3	REAL	0.0
SP3HiLim	High limit for setpoint 3	REAL	100.0
SP3LoLim	Low limit for setpoint 3	REAL	0.0
SP4	Setpoint 4 SP4.ST=FF: Operable in faceplate	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
SP4DeadBand	Width of the dead band control of CV4	REAL	0.0
SP4HiLim	High limit for setpoint 4	REAL	100.0
SP4LoLim	Low limit for setpoint 4	REAL	0.0
StepNo	Batch step number	DWORD	16#00000000
UserStatus	Freely assignable bits for use in faceplate	BYTE	16#00

Output parameters

Parameter	Description	Type	Default
ActInOuts	Status word displays active inputs and outputs in the faceplate	WORD	16#C0C0
AutAct	1 = Automatic mode is active 0 = Manual mode is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
CV1Out	Output of control variable 1 (process value)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
CV2Out	Output of control variable 2 (process value)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
CV3Out	Output of control variable 3 (process value)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
CV4Out	Output of control variable 4 (process value)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see ModPreCon error handling (Page 406)	INT	-1
ManAct	1 = Manual mode active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_Release	Maintenance release: 1 = Release by OS operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV1	Manipulated variable 1 (control signal)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV1HiAct	1 = High limit of manipulated variable 1 reached or exceeded	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV1LoAct	1 = Low limit of manipulated variable 1 reached or exceeded	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV1AutAct	1 = MV1 is set automatically by the algorithm, i.e. AutAct = 1 and MV1TrkOn = 0	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
MV2	Manipulated variable 2 (control signal)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV2HiAct	1 = High limit of manipulated variable 2 reached or exceeded	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV2LoAct	1 = Low limit of manipulated variable 2 reached or exceeded	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV2AutAct	1 = MV2 is set automatically by the algorithm, i.e. AutAct = 1 and MV2TrkOn = 0	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV3	Manipulated variable 3 (control signal)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV3HiAct	1 = High limit of manipulated variable 3 reached or exceeded	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV3LoAct	1 = Low limit of manipulated variable 3 reached or exceeded	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV3AutAct	1 = MV3 is set automatically by the algorithm, i.e. AutAct = 1 and MV3TrkOn = 0	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV4	Manipulated variable 4 (control signal)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV4HiAct	1 = High limit of manipulated variable 4 reached or exceeded	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV4LoAct	1 = Low limit of manipulated variable 4 reached or exceeded	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV4AutAct	1 = MV4 is set automatically by the algorithm, i.e. AutAct = 1 and MV4TrkOn = 0	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
NumberCVs	Number of controlled variables (process values) used	INT	0
NumberDVs	Number of disturbance variables used	INT	0
NumberMVs	Number of manipulated variables used	INT	0

Parameter	Description	Type	Default
OosAct	1 = Block is out of service	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
SP1Out	Setpoint 1 used by controller	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP2Out	Setpoint 2 used by controller	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP3Out	Setpoint 3 used by controller	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP4Out	Setpoint 4 used by controller	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word 1 (Page 391)	DWORD	16#00
Status2	Status word 2 (Page 391)	DWORD	16#00

See also

ModPreCon messaging (Page 407)
ModPreCon block diagram (Page 416)
ModPreCon modes (Page 395)

4.5.7 ModPreCon block diagram

ModPreCon block diagram

This block does not come with a block diagram.

See also

ModPreCon I/Os (Page 408)
ModPreCon messaging (Page 407)
ModPreCon error handling (Page 406)
ModPreCon functions (Page 396)
ModPreCon modes (Page 395)
Description of ModPreCon (Page 391)

4.5.8 Operator control and monitoring

4.5.8.1 ModPreCon views

Views of the ModPreCon block

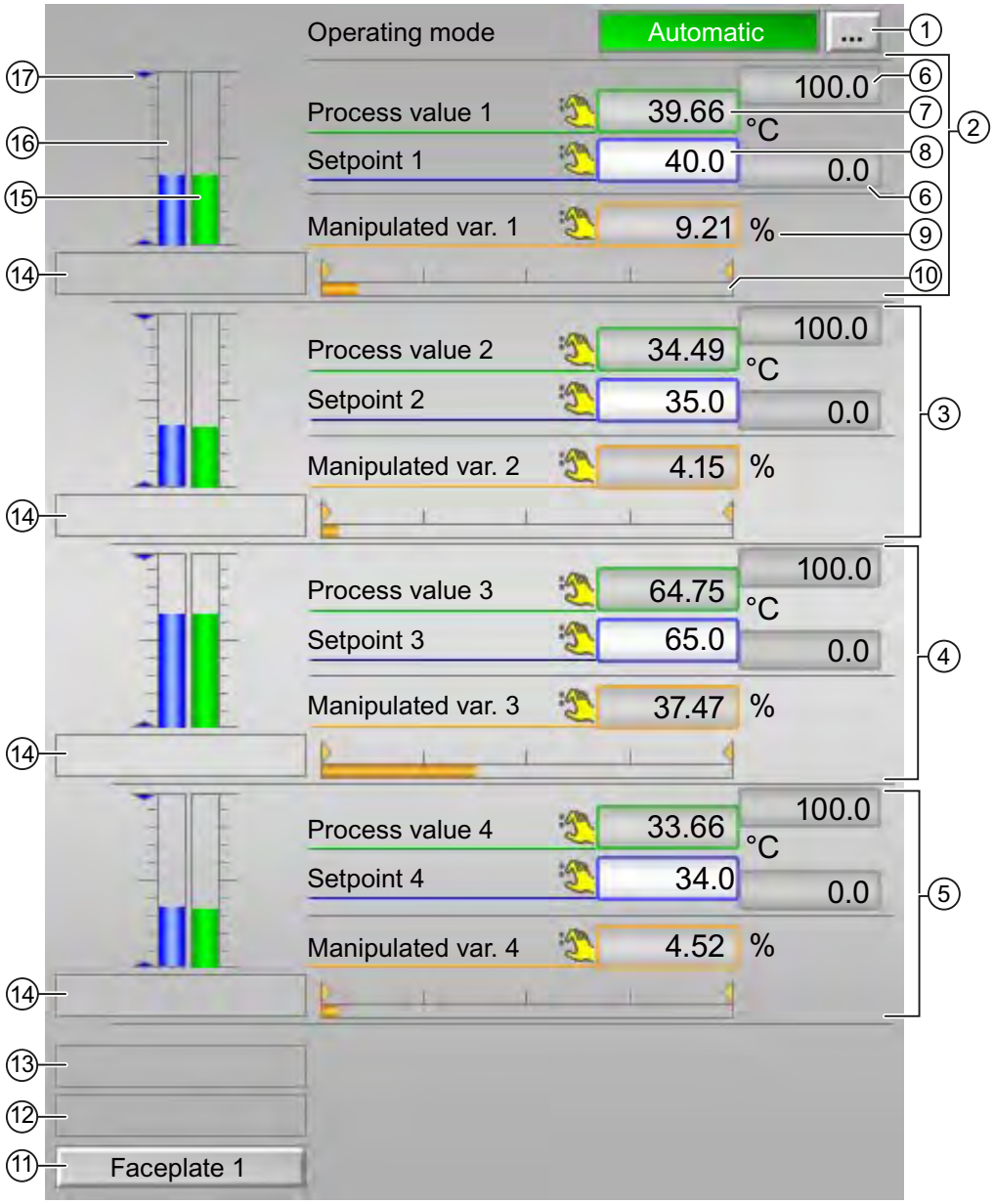
The block ModPreCon provides the following views:

- ModPreCon standard view (Page 417)
- Trend view (Page 172)
- ModPreCon parameter view (Page 420)
- Parameter view channel 1 to 4 of ModPreCon (Page 422)
- ModPreCon preview (Page 424)
- Memo view (Page 171)
- Batch view (Page 170)
- Block icon for ModPreCon (Page 426)

Refer to the sections Structure of the faceplate (Page 123) and Block icon structure (Page 174) for general information on the faceplate and block icon.

4.5.8.2 ModPreCon standard view

ModPreCon standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 29)
- Automatic mode (Page 29)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2), (3), (4), (5) Display and switchover for values for channels 1 to 4

This area always has the same layout for channels 1 to 4:

(6) High and low scale range for the process value

These values provide information on the display range for the bar graph of the process value. The scale range is defined in the engineering system.

(7) Displaying and changing the process value including the signal status

This area shows the current process value with the corresponding signal status.

(8) Displaying and changing the setpoint including the signal status

This area shows the current setpoint with the corresponding signal status. Refer to the Changing values (Page 129) section for information on changing the setpoint.

(9) Displaying and changing the manipulated variable including the signal status

This area shows you the current manipulated variable with the corresponding signal status. Refer to the Changing values (Page 129) section for information on changing the manipulated variable. You can only make a change in manual mode.

(10) Bar graph for the manipulated variable with limit display

This area shows the currently manipulated variable in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES):

- Limits: `MVxHiLim` and `MVxLoLim`
- Display area: `MVxManHiLim` and `MVxManLoLim`

(11) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

(12) Display area for states of the block

This area provides additional information on the operating state of the block:

- Maintenance

Additional information on the display area for states of the block is available in section Maintenance release (Page 47).

(13) Display area for states of the block

This area provides additional information on the operating state of the block:

- Simulation

You can find additional information on this in the Simulating signals (Page 93) section.

(14) Displaying states of the block

There is a display for the states of of the block for channels 1 to 4.

- Tracking

(15) Bar graph for the process value 1

There is a bar graph for the process value for each channel, 1 to 4.

This area shows the current process value in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(16) Bar graph for the setpoint 1

There is a bar graph for the setpoint for each channel, 1 to 4.

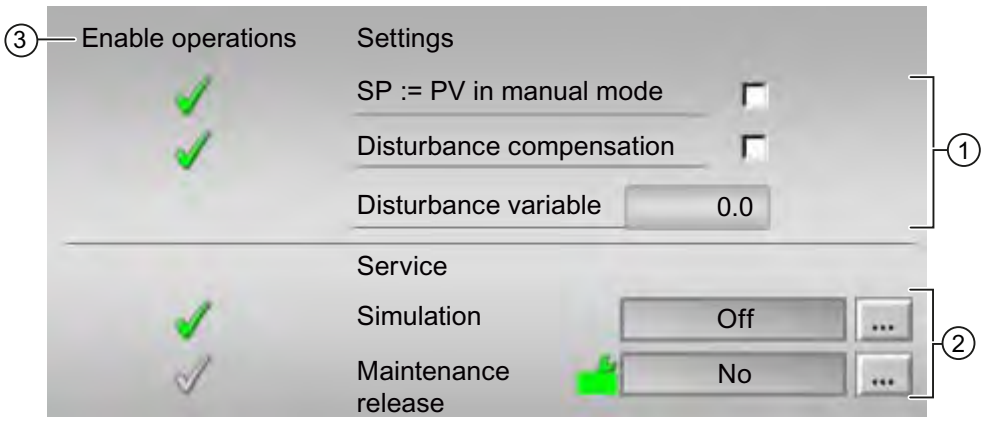
This area shows the current setpoint in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(17) Display for limits

These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the ES.

4.5.8.3 ModPreCon parameter view

ModPreCon parameter view



(1) Settings

You can activate the following functions for the controller in this area:

- SP := PV in manual mode: ☑ Bumpless switchover from manual mode to automatic mode
- Disturbance compensation: ☑ Select disturbance feedforward
- Disturbance variable

You can change the disturbance. Refer to the Changing values (Page 129) section for more on this.

(2) Service

You can select the following functions in this area:

- Simulation
- Maintenance release

Refer to the Switching operating states and operating modes (Page 127) section for more on this.

You can find information on this area in the section:

- Simulating signals (Page 93)
- Maintenance release (Page 47)

(3) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

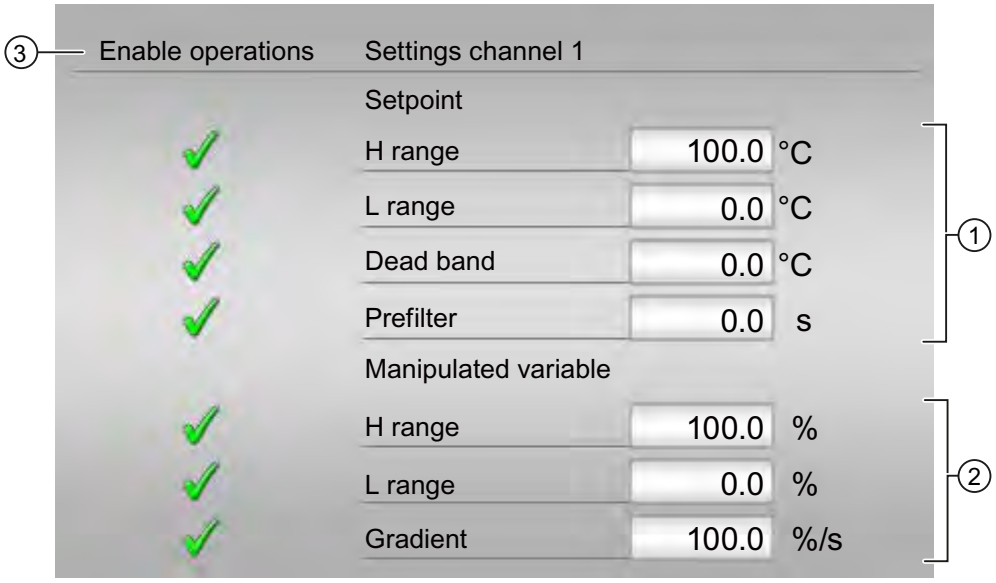
Symbols for the operator control enables:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions.

4.5.8.4 Parameter view channel 1 to 4 of ModPreCon

Parameter view channel 1 to 4 for ModPreCon

The layout of the parameter view for channels 1 to 4 is always identical:



(1) Display and change the parameters for the setpoint

You can change the following parameters for the setpoint in this area:

- H range
- L range
- Dead band
- Prefilter

You can find additional information on this in the Changing values (Page 129) section.

(2) Display and change the parameters for the manipulated variable

You can change the following parameters for the manipulated variable in this area:

- H range
- L range
- Gradient

You can find additional information on this in the Changing values (Page 129) section.

(3) Enable operations

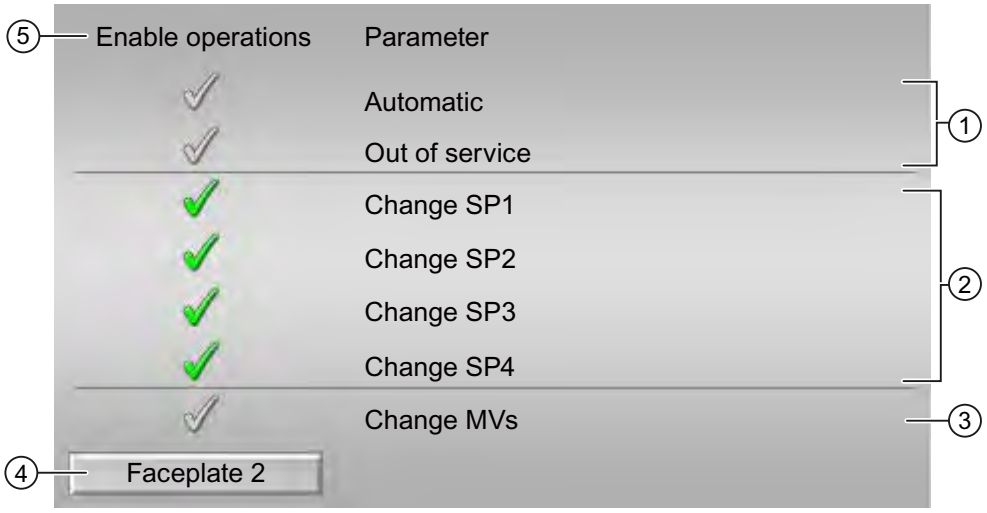
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions.

4.5.8.5 ModPreCon preview

ModPreCon preview



(1), (2), (3) The following operation control enables for parameters are shown here:

- Automatic: You can switch to automatic mode
- Out of service: You can switch to the Out of service operating mode.
- Change SP1: You can change the setpoint 1
- Change SP2: You can change the setpoint 2
- Change SP3: You can change the setpoint 3
- Change SP4: You can change the setpoint 4
- Change MVs: You can change the manipulated variables

Note

The OS operator must always be able to switch to manual mode. That is why the switch to manual mode is not shown here in the faceplate.

(4) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

(5) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

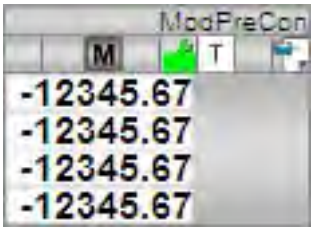
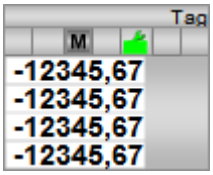
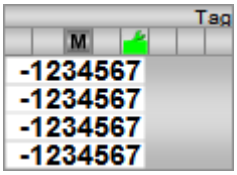
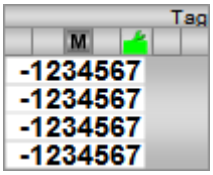
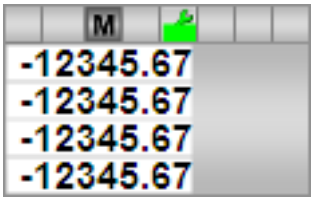
- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions.

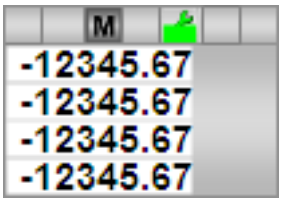
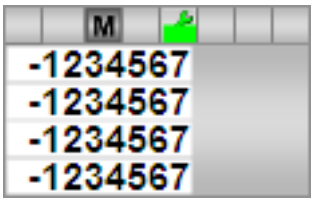
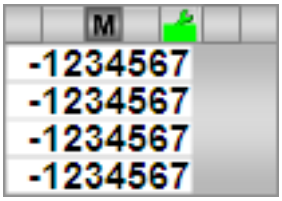
4.5.8.6 Block icon for ModPreCon

Properties of the ModPreCon block icon

A variety of block icons are available with the following functions:

- Process tag type
- Operating modes
- Signal status, maintenance release
- Tracking
- Memo display
- Process value (black, with and without decimal places)

Icons	Selection of the block icon in CFC	Special features
	1	Block icon in the full display
	2	
	3	
	4	
	5	

Icons	Selection of the block icon in CFC	Special features
	6	
	7	
	8	
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181).

4.6 PIDConL - Continuous PID controller

4.6.1 Description of PIDConL

Object name (type + number) and family

Type and number: FB 1874

Family: Control

Area of application for PIDConL

The block is used for the following applications:

- Fixed setpoint control
- Cascade control
- Ratio control
- Split-range control
- Smith predictor closed-loop control
- Override control (override)

How it works

The block is a PID controller with continuous output signal (manipulated variable). It is used to activate a final controlling element with continuous action input.

The block functions following the PID algorithm with a delayed D action and an integrator with double precision.

The block is suitable for controlling sluggish control loops, for example, for temperatures and filling levels, and high-speed control loops, for example, for flow rates and speed. For a given CPU, a compromise has to be made between the number of controllers and the frequency with which the individual controllers have to be processed. The faster the modulated control loops are, i.e. the more frequently the manipulated variables have to be calculated per time unit, the lower the number of controllers that can be installed.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

For the control loop monitoring to work as planned in the trend view of the controller faceplates, the

```
S7_xarchive:='Value, shortterm;'
```

attributes in the process tag types for control loops at the controller function block must be set for the following tags:

- Input parameters:
  - `CPI_In`
- Output parameters
  - `MV`
  - `MV_HiAct`
  - `MV_LoAct`
  - `AutAct`
  - `SP`
  - `PV_Out`
  - `PV_ToleHi`
  - `PV_ToleLo`

For the PIDConL block, the Advanced Process Library contains templates for process tag types as examples and there is a example project (APL_Example_xx, xx designates the language variant) containing different application cases for this block. Several application cases are simulated in the example project and serve to explain how the block works.

Examples of process tag types:

- PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 1430)
- Split-range control (Page 1438)
- Ratio control (Page 1440)
- Cascade control (Page 1442)
- PID - control with operating-point-oriented parameter control (GainScheduling) (Page 1431)
- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 1433)
- Override control (Page 1445)
- PID controller with Smith predictor (SmithPredictorControl) (Page 1436)

Application cases in example project:

- Process simulation including noise generator (Page 1456)
- Cascade control of a temperature by using the heat flow (Page 1458)
- Control loop monitoring with simulation of colored noise (Page 1460)
- Feedforward control to compensate a measurable disturbance variable (Page 1461)
- Operating point-oriented adaptation of parameters (gain scheduling) for non-linear processes (Page 1462)
- Override control on a pipeline (Page 1462)
- Smith predicator for a dead time controlled system (Page 1463)
- Filtering of noisy measured values in a control loop (Page 1463)

Startup characteristics

Use the Feature bit Setting the startup response (Page 187) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

Status word allocation for `Status1` parameter

For a description of the individual parameters, see the section PIDConL I/Os (Page 445).

Status bit	Parameter
0	`Occupied`
1	`BatchEn`
2	`SimOn`
3	`OosAct.Value`
4	`OosLi.Value`
5	`NOT ManAct.Value`
6	Not allocated
7	`ManAct.Value`
8	`SP_ExtAct.Value`
9	`MV_ForOn.Value`
10	`MV_TrkOn.Value AND NOT (ManAct.Value OR MV_ForOn.Value)`
11	`MV.Value > ManLoLim`
12 ... 18	Not allocated
19	`AdvCoAct`
20	1 = Input parameter `Rbk` is not interconnected (RbkOut.ST = 16#FF)
21	`NegGain`
22 - 23	Not allocated
24	`OptimOcc`
25 - 31	Not allocated

Status word allocation for `Status2` parameter

Status bit	Parameter
0	MsgLock
1	PV_AH_Act.Value
2	PV_WH_Act.Value
3	PV_TH_Act.Value
4	PV_TL_Act.Value
5	PV_WL_Act.Value
6	PV_AL_Act.Value
7	PV_AH_En
8	PV_WH_En
9	PV_TH_En
10	PV_TL_En
11	PV_WL_En
12	PV_AL_En
13	PV_AH_MsgEn
14	PV_WH_MsgEn
15	PV_TH_MsgEn
16	PV_TL_MsgEn
17	PV_WL_MsgEn
18	PV_AL_MsgEn
19	ER_AH_Act.Value
20	ER_AL_Act.Value
21	ER_AH_En
22	ER_AL_En
23	ER_AH_MsgEn
24	ER_AL_MsgEn
25	RbkWH_Act.Value
26	RbkWL_Act.Value
27	RbkWH_En
28	RbkWL_En
29	RbkWH_MsgEn
30	RbkWL_MsgEn
31	MS_RelOp

See also

PIDConL functions (Page 433)
PIDConL messaging (Page 442)
PIDConL block diagram (Page 456)
PIDConL error handling (Page 441)
PIDConL modes (Page 432)

4.6.2 PIDConL modes

PIDConL operating modes

The block can be operated using the following modes:

- Automatic mode (Page 29)
- Manual mode (Page 29)
- Program mode for closed-loop controllers (Page 34)
- Out of service (Page 27)

The next section provides additional block-specific information relating to the general descriptions.

Automatic mode

You can find general information on "Automatic mode", switching modes and bumpless switchover in the Manual and Automatic mode for control blocks (Page 29) section.

Manual mode

You can find general information on "Manual mode", switching modes and bumpless switchover in the Manual and Automatic mode for control blocks (Page 29) section.

Program mode for controllers

General information on "Program mode for controllers" is available in the section Program mode for closed-loop controllers (Page 34).

Out of service

You can find general information about the "Out of service" mode in the Out of service (Page 27) section.

See also

PIDConL block diagram (Page 456)

PIDConL I/Os (Page 445)

PIDConL messaging (Page 442)

PIDConL error handling (Page 441)

PIDConL functions (Page 433)

Description of PIDConL (Page 428)

4.6.3 PIDConL functions

Functions of PIDConL

The functions for this block are listed below.

Generation of manipulated variables

The manipulated variable MV can be generated as follows:

MV_ForOn	ManAct	MV_TrkOn	AdvCoAct AND NOT AdvCoModSP	MV	Limit monitoring	State
1	-	-	-	MV_Forced	none	Forced tracking through constraint without limitation
0	1	-	-	Man	ManHiLim ManLoLim	Manual mode, set by the operator
0	0	1	-	MV_Trk	MV_HiLim MV_LoLim	Tracking with limitation
0	0	0	1	AdvCoMV	MV_HiLim MV_LoLim	Higher-level program mode
0	0	0	0	P_Part + I_Part + D_Part+ FFwd	MV_HiLim MV_LoLim	Automatic mode (PID algorithm)

If the controller is in "out of service" mode, the output parameter MV is set to the last valid value in manual mode or the safety manipulated variable depending on the Feature bit (Safety value of the manipulated variable effective at startup (Page 196)). Refer to the Out of service (Page 27) section for more on this.

Tracking and limiting a manipulated variable

The block provides the standard function Tracking and limiting a manipulated value (Page 111).

Safe position

The block provides the standard function Safe position for motors, valves and controllers (Page 120)

"Actuator active" information

If the manipulated variable MV is greater than the minimum manual limit ManLoLim, this is recognized as actuator active. This status can be used to indicate, for example, a customized icon in the process image and is saved in the status word (see Status word section in Description of PIDConL (Page 428)).

Limit monitoring of position feedback

The block provides the standard function Limit monitoring of the feedback (Page 76).

External/internal setpoint specification

The block provides the standard function Setpoint input - internal and external (Page 96).

Setpoint limiting for external setpoints

The block provides the standard function Setpoint limiting for external setpoints (Page 110).

Gradient limit of the setpoint

The block provides the standard function Ramp limiting of the setpoint (Page 108).

Using setpoint ramp

The block provides the standard function Using a setpoint ramp (Page 98).

Tracking setpoint in manual mode

The block provides the standard function Setpoint tracking the process variable in manual mode (Page 110).

Simulating signals

The block provides the standard function Simulating signals (Page 93).

You can simulate the following values:

- Process value (`SimPV`)
- Position feedback (`SimRbk`)

Limit monitoring of the process value

The block provides the standard function Limit monitoring of the process value (Page 70).

Control error generation and dead band

The block provides the standard function Control error generation and deadband (Page 109).

Limit monitoring of control error

The block provides the standard function Monitoring the error signal limits (Page 77).

Inverting control direction

The block provides the standard function Inverting the control sense (Page 108).

Physical standardization of setpoint, manipulated variable and process value

Controller gain `Gain` is entered either using a physical variable or as standardized value.

Gain as physical variable:

The standardized variables retain their default values:

- `NormPV.High = 100` and `NormPV.Low = 0`
- `NormMV.High = 100` and `NormMV.Low = 0`

The effective gain is:

```
GainEff = Gain
```

Entering a standardized gain (dimensionless):

Change the standardized variables to the actual range of the process values and manipulated variables.

- Internal and external setpoints; the process value and corresponding parameters are entered according to the physical measuring range of the process value.
- The manual value, the tracking value of the manipulated variable, feedforward control and the corresponding parameters are set according to the physical measuring range of the manipulated variable.

The effective gain is:

```
GainEff = (NormMV.High - NormMV.Low) / (NormPV.High - NormPV.Low) · Gain
```

Selecting a unit of measure

The block provides the standard function Selecting a unit of measure (Page 53).

PID algorithm

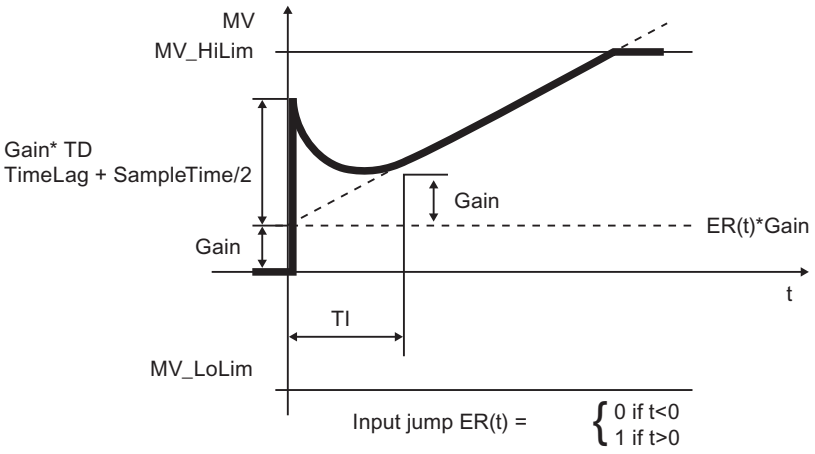
The manipulated variable is generated in automatic mode according to the following algorithm:

```
MV = Gain · (1 + 1/(TI · s) + TD/( 1 + TD/DiffGain · s) · ER
```

Where:

s = Complex number

The following step response occurs:



Note

This formula describes a standard application where P, I and D action is activated and the P and D actions are not in the feedback circuit (`PropSel = true, TI <> 0 DiffToFbk = false` and `PropFacSP = 1`).

The D action delay is derived from `TD/DiffGain`.

- The P action is displayed after `P_Part` and can be deactivated using `PropSel = 0`.
- The I action is displayed after `I_Part` and can be deactivated using `TI = 0`.
- The D action is displayed after `D_Part` and can be deactivated using `TD = 0`.

Structure segmentation at controllers

The block provides the standard function Structure segmentation at controllers (Page 113).

Anti-windup

The controller has an anti-windup function. The I action is frozen after the manipulated variable has reached limits (`MV_HiLim` or `MV_LoLim`).

Activating and limiting disturbance variables

The block provides the standard function Activating and limiting disturbance variable (Page 113).

Control zone

The block provides the standard function Using control zones (Page 109).

Forming the signal status for blocks

The block provides the standard function Forming and outputting signal status for blocks (Page 88).

- Signal status for the process value PV_Out:

  The signal status of output parameter PV_Out is always equivalent to the signal status of input parameter PV or, if the block is in simulation mode, 16#60.

- Signal status for the setpoint value SP:

  The signal status of output parameter SP is always equivalent to the signal status of input parameter SP_Ext or SP_Int, depending on how the setpoint is specified. If the internal setpoint SP_Int is used, the signal status is always output as 16#80.

- Signal status of the system deviation ER:

  The signal status of output parameterER is obtained from the worse signal status of the two output parameters PV_Out and SP and is output.

- Worst signal status:

  The worst signal status ST_Worst for the block is formed from the following parameters:

  – ER.ST
  – FFwd.ST
  – Rbk.ST

Configurable reactions using the Feature parameter

You can find an overview of all reactions provided by the Feature parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)
1	Reaction to the out of service mode (Page 199)
2	Resetting the commands for changing the mode (Page 193)
4	Setting switch or button mode (Page 195)
15	Safety manipulated variable with "out of service" operating mode in effect (Page 196)
16	Safety value of the manipulated variable effective at startup (Page 196)
18	Disabling bumpless changeover to automatic mode for controllers (Page 197)
22	Update acknowledgment and error status of the message call (Page 193)

Operator control permissions via parameters OS_Perm and OS1Perm

The block has the following operator control permissions (Page 45) for the parameter `OS_Perm`:

Bit	Function
0	1 = Operator can switch into automatic mode `AutModOp`
1	1 = Operator can switch into manual mode `ManModOp`
2	1 = Operator can switch into "out of service" mode `OosOp`
3	1 = Operator can switch into program mode `AdvCoEn`
4	1 = Operator can switch the setpoint to external `SP_ExtOp`
5	1 = Operator can switch the setpoint to internal `SP_IntOp`
6	1 = Operator can change the internal setpoint `SP_Int`
7	1 = Operator can change the manual parameter `Man`
8	1 = Operator can change maximum usage limit of the setpoint `SP_InHiLim`
9	1 = Operator can change minimum usage limit of the setpoint `SP_InLoLim`
10	1 = Operator can change maximum usage limit of the manipulated variable `ManHiLim`
11	1 = Operator can change minimum usage limit of the manipulated variable `ManLiLim`
12	1 = Operator can use the setpoint's gradient limitation function `SP_RateOn`
13	1 = Operator can change the setpoint's high limit for the ramp `SP_UpRaLim`
14	1 = Operator can change the setpoint's low limit for the ramp `SP_DnRaLim`
15	1 = Operator can change between the time value or the value for the ramp `SP_RmpModTime`
16	1 = Operator can change the ramp time `SP_RmpTime`
17	1 = Operator can change the target setpoint `SP_RmpTarget` for the setpoint ramp
18	1 = Operator can activate the setpoint ramp function `SP_RmpOn`
19	1 = Operator can permit the PID optimization function `OptimEn`
20	1 = Operator can activate the track setpoint in manual mode function `SP_TrkPV`
21	1 = Operator can activate the bumpless changeover from external to internal function `SP_TrkExt`
22	1 = Operator can change the gain parameter `Gain`
23	1 = Operator can change the integral action time parameter `TI`
24	1 = Operator can change the derivative action time parameter `TD`
25	1 = Operator can change the derivative gain parameter `DiffGain`
26	1 = Operator can change the dead band parameter `DeadBand`
27	1 = Operator can change the control zone parameter `ConZone`
28-31	Not allocated

The block has the following operator control permissions for the OS1Perm parameter:

Bit	Function
0	1 = Operator can change the limit (process value) for the high alarm PV_AH_Lim
1	1 = Operator can change the limit (process value) for the high warning PV_WH_Lim
2	1 = Operator can change the limit (process value) for the high tolerance PV_TH_Lim
3	1 = Operator can change the hysteresis (process value) PV_Hyst
4	1 = Operator can change the limit (process value) for the low tolerance PV_TL_Lim
5	1 = Operator can change the limit (process value) for the low warning PV_WL_Lim
6	1 = Operator can change the limit (process value) for the low alarm PV_AL_Lim
7	1 = Operator can change the limit (control error) for the high alarm ER_AH_Lim
8	1 = Operator can change the hysteresis (control error) ER_Hyst
9	1 = Operator can change the limit (control error) for the low alarm ER_AL_Lim
10	1 = Operator can change the limit (position feedback) for the high warning RbkWH_Lim
11	1 = Operator can change the hysteresis (position feedback) RbkHyst
12	1 = Operator can change the limit (position feedback) for the low warning RbkWL_Lim
13 - 15	Not allocated
16	1 = Operator can activate the simulation function SimOn
17	1 = Operator can activate the maintenance release function MS_RelOp
18 - 31	Not allocated

Maintenance release

The block provides the standard function Maintenance release (Page 47)

Generating instance-specific messages

The block provides the standard function Generating instance-specific messages (Page 46) without the time stamp function in the I/O devices.

Specifying the display area for process and setpoint values as well as operations

This block provides the standard function Display and operator input area for process values and setpoints (Page 42).

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 122).

See also

PIDConL I/Os (Page 445)
PIDConL block diagram (Page 456)
PIDConL error handling (Page 441)
PIDConL modes (Page 432)
Program mode for closed-loop controllers (Page 34)

4.6.4 PIDConL error handling

PIDConL troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` output parameter can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Default value when implementing the block; block will not be processed.
0	There is no error.
30	The value of `PV` can no longer be displayed in the REAL number field.
31	The value of `SP_Ext` can no longer be displayed in the REAL number field.
32	The value of `FFwd` can no longer be displayed in the REAL number field.
33	The value of `MV_Trk` can no longer be displayed in the REAL number field.
34	The value of `MV_Forced` can no longer be displayed in the REAL number field.
35	The value of `Rbk` can no longer be displayed in the REAL number field.
36	The value of `MV` can no longer be displayed in the REAL number field.
60	`|Tl| < SampleTime/2`
61	`|TD| < SampleTime`
62	`DiffGain < 1` or `DiffGain > 10`
63	`TD/DiffGain < SampleTime/2`
64	`PropFacSP < 0` or `PropFacSP > 1`
66	`NormPV_High = NormPV_Low`

See also

PIDConL block diagram (Page 456)

PIDConL I/Os (Page 445)

PIDConL messaging (Page 442)

PIDConL functions (Page 433)

PIDConL modes (Page 432)

Description of PIDConL (Page 428)

Setting switch or button mode (Page 195)

4.6.5 PIDConL messaging

Messaging

The following messages can be generated for this block:

- Control system fault
- Process messages
- Instance-specific messages

Control system fault

The following control system error messages can be output:

Message instance	Message identifier	Message class	Event
MsgEvId2	SIG 6	AS process control message - fault	$$BlockComment$$ External error has occurred

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter CSF. If this signal changes to CSF = 1, a control system fault is triggered (MsgEvId2, SIG 6).

Process messages

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 1	Alarm - high	$$BlockComment$$ PV - high alarm limit violated
	SIG 2	Warning - high	$$BlockComment$$ PV - high warning limit violated
	SIG 3	Tolerance - high	$$BlockComment$$ PV - high tolerance limit violated
	SIG 4	Tolerance - low	$$BlockComment$$ PV - low tolerance limit violated
	SIG 5	Warning - low	$$BlockComment$$ PV - low warning limit violated
	SIG 6	Alarm - low	$$BlockComment$$ PV - low alarm limit violated
	SIG 7	Alarm - high	$$BlockComment$$ ER - high alarm limit violated
	SIG 8	Alarm - low	$$BlockComment$$ ER - low alarm limit violated

Message instance	Message identifier	Message class	Event
MsgEvId2	SIG 7	Warning - high	$$BlockComment$$ Rbk - high warning limit violated
	SIG 8	Warning - low	$$BlockComment$$ Rbk - low warning limit violated

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Instance-specific messages

You can use up to four instance-specific messages with this block.

Message instance	Message identifier	Message class	Event
MsgEvId2	SIG 1	AS process control message - fault	$$BlockComment$$ External message 1 Status 16#@5%x@
	SIG 2	AS process control message - fault	$$BlockComment$$ External message 2 Status 16#@6%x@
	SIG 3	AS process control message - fault	$$BlockComment$$ External message 3 Status 16#@7%x@
	SIG 4	AS process control message - fault	$$BlockComment$$ External message 4 Status 16#@8%x@

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Associated values for message instance `MsgEvId1`

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`
4	Process value `PV_Out`
5	Control error `ER`
6	`ExtVa106`
7	`ExtVa107`
8	Not allocated
9	Not allocated
10	Not allocated

The associated values 6 and 7 are assigned to the parameters `ExtVa106` ... `ExtVa107` and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

Associated values for message instance `MsgEvId2`

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`
4	Position feedback `Rbk`
5	Signal status `ExtMsg1`
6	Signal status `ExtMsg2`
7	Signal status `ExtMsg3`
8	Signal status `ExtMsg4`
9	`ExtVa209`
10	`ExtVa210`

The associated values 9 and 10 are assigned to the parameters `ExtVa209` ... `ExtVa210` and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

See also

Description of PIDConL (Page 428)
PIDConL functions (Page 433)
PIDConL I/Os (Page 445)
PIDConL block diagram (Page 456)
PIDConL error handling (Page 441)
PIDConL modes (Page 432)

4.6.6 PIDConL I/Os

PIDConL I/Os

Input parameters

Parameter	Description	Type	Default
AdvCoEn	1 = Enable program mode via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AdvCoOn	1 = Enable program mode via faceplate	BOOL	0
AdvCoModSP	Type of program mode: 1 = Setpoint specification 0 = Manipulated variable specification	BOOL	1
AdvCoMstrOn	Activate (0-1) or deactivate (1-0) program mode via edge transition	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AdvCoMV	Specified value from the external program	REAL	0.0
AutModLi	1 = Automatic mode via interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AutModOp	1 = Automatic mode via operator (controlled by ModLiOp = 0)	BOOL	0
BatchEn	1 = Enable allocation for batch control	BOOL	0
BatchID	Batch ID	DWORD	16#00000000
BatchName	Batch name	STRING[32]	''
ConZone	Width of control zone	REAL	0.0
CPI_In	Input for control performance index, which is calculated by the assigned ConPerMon block	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#78
CSF	1 = External error (control system error)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
DeadBand	Width of dead band	REAL	0.0
DiffGain	Gain of differentiator [1...10] DiffGain = TD / (differentiation of D action)	STRUCT • Value: REAL • ST: BYTE	- • 5.0 • 16#80
DiffToFbk	1 = D action is placed in the feedback	BOOL	0
EN	1 = Called block will be processed	BOOL	1
ER_A_DC	Delay for incoming alarms during control error monitoring	REAL	0.0

Parameter	Description	Type	Default
ER_A_DG	Delay for outgoing alarms during control error monitoring	REAL	0.0
ER_AH_En	1 = Activate alarm (high) for control error monitoring	BOOL	1
ER_AH_Lim	Alarm limit (high) for control error monitoring	REAL	100.0
ER_AH_MsgEn	1 = Activate messages for alarm (high) for control error monitoring	BOOL	1
ER_AL_En	1 = Activate alarm (low) for control error monitoring	BOOL	1
ER_AL_Lim	Alarm limit (low) for control error monitoring	REAL	-100.0
ER_AL_MsgEn	1 = Activate messages for alarm (low) for control error monitoring	BOOL	1
ER_Hyst	Alarm hysteresis for control error	REAL	1.0
ExtMsg1	1 = Binary input for freely selectable message 1 is used	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg2	1 = Binary input for freely selectable message 2 is used	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg3	1 = Binary input for freely selectable message 3 is used	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg4	1 = Binary input for freely selectable message 4 is used	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtVa106	Associated value 6 for messages (MsgEvID1)	ANY	
ExtVa107	Associated value 7 for messages (MsgEvID1)	ANY	
ExtVa209	Associated value 9 for messages (MsgEvID2)	ANY	
ExtVa210	Associated value 10 for messages (MsgEvID2)	ANY	
Feature	I/O for additional functions (Page 433)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • ... • 0
FFwd	Input for additive disturbance variable activation	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
FFwdHiLim	Limit (high) for additive disturbance variable activation	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80

Parameter	Description	Type	Default
FFwdLoLim	Limit (low) for additive disturbance variable activation	STRUCT • Value: REAL • ST: BYTE	- • -100.0 • 16#80
Gain	Proportional gain Gain.ST = 16#FF: Operable in faceplate	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#FF
IntHoldNeg	1 = Integrator cannot run in negative direction	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
IntHoldPos	1 = Integrator cannot run in positive direction	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Man	Manual specification for the manipulated variable	REAL	0.0
ManHiLim	Limit (high) for manual parameter Man	REAL	100.0
ManLoLim	Limit (low) for manual parameter Man	REAL	0.0
ManModLi	1 = Manual mode via interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ManModOp	1 = Manual mode by OS operator (controlled by ModLiOp = 0)	BOOL	1
ModLiOp	Operating mode switchover between: • 0 = Operator • 1 = Interconnection or SFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_RelOp	1 = Maintenance release by OS operator	BOOL	0
MsgEvId1	Message number (assigned automatically)	DWORD	16#FFFFFFFF
MsgEvId2	Message number (assigned automatically)	DWORD	16#FFFFFFFF
MsgLock	1 = Suppress process messages	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_Forced	Forced manipulated variable that is not limited and assumes top priority	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV_ForOn	1 = Forced manipulated variable MV_Forced output unlimited at output MV	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_HiLim	Limit (high) for manipulated variable MV	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
MV_LoLim	Limit (low) for manipulated variable MV	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

Parameter	Description	Type	Default
MV_Offset	Manipulated variable for ER = 0, operating point for controller with deactivated I action	REAL	0.0
MV_OpScale	OS display range for manipulated variable MV	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
MV_Trk	Tracking value for the manipulated variable MV	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV_TrkOn	1 = Tracking of manipulated variable MV	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_Unit	Unit of measure for manipulated variable	INT	1342
NegGain	0 = Positive controller gain: ER = Gain · (SP - PV) 1 = Negative controller gain: ER = Gain · (PV - SP)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
NormMV	Manipulated variable range (MV) for standardizing the proportional gain (GAIN)	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
NormPV	Process value range (PV) for standardizing the proportional gain (GAIN)	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
Occupied	1 = Occupied by batch control	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC (0-1 edge transition)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OptimEn	1 = Enable optimization of PID parameters by PID tuner	BOOL	0
OptimOcc	1 = Optimization running	BOOL	0
OS_Perm	I/O for operator control permissions (Page 433)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
OS1Perm	I/O for operator control permissions (Page 433)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
PropFacSP	Applying the P action to the feedback [0..1]. 0 = P action fully in feedback	REAL	1.0
PropSel	1 = Activate P action	BOOL	1

Parameter	Description	Type	Default
PV	Process value (controlled variable)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_A_DC	Delay time for incoming PV alarms [s]	REAL	0.0
PV_A_DG	Delay time for outgoing PV alarms [s]	REAL	0.0
PV_AH_En	1 = Enable PV alarm limit (high)	BOOL	1
PV_AH_Lim	PV alarm limit (high)	REAL	95.0
PV_AH_MsgEn	1 = Enable message for PV alarm (high)	BOOL	1
PV_AL_En	1 = Enable PV alarm limit (low)	BOOL	1
PV_AL_Lim	PV alarm limit (low)	REAL	5.0
PV_AL_MsgEn	1 = Enable message for PV alarm (low)	BOOL	1
PV_Hyst	Hysteresis for PV alarm, warning and tolerance limits	REAL	1.0
PV_OpScale	Limit for scale in PV bar graph of faceplate	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
PV_T_DC	Delay time for incoming PV tolerance messages [s]	REAL	0.0
PV_T_DG	Delay time for outgoing PV tolerance messages [s]	REAL	0.0
PV_TH_En	1 = Enable PV tolerance limit (high)	BOOL	0
PV_TH_Lim	PV tolerance message limit (high)	REAL	85.0
PV_TH_MsgEn	1 = Enable message for PV tolerance message (high)	BOOL	1
PV_TL_En	1 = Enable PV tolerance limit (low)	BOOL	0
PV_TL_Lim	PV tolerance message limit (low)	REAL	15.0
PV_TL_MsgEn	1 = Enable message for tolerance message (low)	BOOL	1
PV_Unit	Unit of measure for process value	INT	1001
PV_W_DC	Delay time for incoming PV warnings [s]	REAL	0.0
PV_W_DG	Delay time for outgoing PV warnings [s]	REAL	0.0
PV_WH_En	1 = Enable PV warning limit (high)	BOOL	1
PV_WH_Lim	PV warning limit (high)	REAL	90.0
PV_WH_MsgEn	1 = Enable message for PV warning (high)	BOOL	1
PV_WL_En	1 = Enable PV warning limit (low)	BOOL	1
PV_WL_Lim	PV warning limit (low)	REAL	10.0
PV_WL_MsgEn	1 = Enable message for PV warning (low)	BOOL	1
Rbk	Position feedback for display on OS	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
RbkHyst	Alarm hysteresis for position feedback	REAL	1.0

Parameter	Description	Type	Default
RbkWH_En	1 = Enable warning (high) for position feedback	BOOL	1
RbkWH_Lim	Limit for position feedback of warning (high)	REAL	100.0
RbkWH_MsgEn	1 = Enable messages for warning (high) for position feedback	BOOL	1
RbkWL_En	1 = Enable warning (low) for position feedback	BOOL	1
RbkWL_Lim	Limit for position feedback of warning (low)	REAL	0.0
RbkWL_MsgEn	1 = Enable messages for warning (low) for position feedback	BOOL	1
RefStdDevIn	Reference value of PV standard deviation (sigma) in defined "good" state of control loop	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#78
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
SafePos	1 = Safe position (Page 120) for controller's manipulated variable is ManHiLim, 0 = Safe position for controller's manipulated variable is ManLoLim	BOOL	0
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SelFp1	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
SimOn	1 = Simulation on	BOOL	0
SimPV	Process value used for SimOn = 1	REAL	0.0
SimRbk	Position feedback used for SimOn = 1	REAL	0.0
SP_DnRaLim	Limit (low) for ramp of setpoint [SP_Unit/s]	REAL	100.0
SP_ExHiLim	Limit (high) for external setpoint	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
SP_ExLoLim	Limit (low) for external setpoint	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_Ext	external setpoint - (to interconnection)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_ExtLi	1 = Select external setpoint (via interconnection)	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_ExtOp	1 = Select external setpoint (via operator)	BOOL	0

Parameter	Description	Type	Default
SP_InHiLim	Limit (high) of internal setpoint	REAL	100.0
SP_InLoLim	Limit (low) of internal setpoint	REAL	0.0
SP_Int	Internal setpoint for operation	REAL	0.0
SP_IntLi	1 = Select internal setpoint (via interconnection)	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_IntOp	1 = Select internal setpoint (via operator)	BOOL	0
SP_LiOp	Select setpoint source (internal/external): 1 = Via interconnection 0 = Via operator	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_RateOn	1 = Activate limitation of setpoint gradients	BOOL	0
SP_RmpModTime	1 = Use time (SP_RmpTime) for setpoint ramp, 0 = Use gradient	BOOL	0
SP_RmpOn	1 = Activate setpoint ramp to target setpoint SP_RmpTarget	BOOL	0
SP_RmpTarget	Target setpoint for setpoint ramp	REAL	0.0
SP_RmpTime	Time for setpoint ramp [s] from current SP up to SP_RmpTarget	REAL	0.0
SP_TrkExt	1 = Bumpless switchover from external to internal setpoint active	BOOL	1
SP_TrkPV	1 = Setpoint follows PV in manual mode and with tracking	BOOL	0
SP_UpRaLim	Gradient limit (high) for setpoint [SP_Unit/s]	REAL	100.0
StepNo	Batch step number	DWORD	16#00000000
TD	Derivative action time [s] TD.ST = 16#FF: Operable in faceplate	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
TI	Integral action time [s] TI.ST = 16#FF: Operable in faceplate	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#FF
UserStatus	Freely assignable bits for use in faceplate	BYTE	16#00

Output parameters

Parameter	Description	Type	Default
AdvCoAct	1 = Program mode is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AdvCoRdy	1 = Program mode available	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AutAct	1 = Automatic mode is active 0 = Manual mode is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
CascaCut	Cascade connection: 1 = Control chain from primary to secondary controller is interrupted	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
D_Part	D action of PID algorithm	REAL	0.0
ENO	1 = Block algorithm completed without errors	BOOL	0
ER	Control error	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
ER_AH_Act	1 = Alarm limit (high) for control error violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ER_AL_Act	1 = Alarm limit (low) for control error violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ErrorNum	Output of current error number. For error numbers that can be output by this block, see PIDConL error handling (Page 441)	INT	-1
FFwdHiAct	1 = Limit (high) for additive disturbance variable activation violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FFwdLoAct	1 = Limit (low) for additive disturbance variable activation violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
GainEff	Effective proportional gain, depends on Gain, NormPV and NormMV	REAL	1.0
I_Part	I action of PID algorithm	REAL	0.0
ManAct	1 = Manual mode active	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80

Parameter	Description	Type	Default
ManHiOut	Limit (high) for manual mode, corresponds to the input parameter ManHiLim	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
ManLoOut	Limit (low) for manual mode, corresponds to the input parameter ManLoLim	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MS_Release	Maintenance release: 1 = Release for OS operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MsgAckn1	Alarm acknowledgement status 1 (output STATUS of first ALARM_8P)	WORD	16#0000
MsgAckn2	Alarm acknowledgement status 2 (output STATUS of second ALARM_8P)	WORD	16#0000
MsgErr1	1 = Alarm error 1 (output ERROR of first ALARM_8P)	BOOL	0
MsgErr2	1 = Alarm error 2 (output ERROR of second ALARM_8P)	BOOL	0
MsgStat1	Alarm status 1 (output ERROR of first ALARM_8P)	WORD	16#0000
MsgStat2	Alarm status 2 (output ERROR of second ALARM_8P)	WORD	16#0000
MV	Manipulated variable	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV_HiAct	1 = Limit (high) of manipulated variable violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_LoAct	1 = Limit (low) of manipulated variable violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_UnitOut	Unit of measure for manipulated variable, for interconnecting to the MV_Unit input parameter of the ConPerMon block	INT	0
OosAct	1 = Block is "Out of service"	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
OS1PermOut	Display of OS1Perm	DWORD	16#FFFFFFFF
OS1PermLog	Display of OS1Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
P_Part	P action of PID algorithm	REAL	0.0

Parameter	Description	Type	Default
PV_AH_Act	1 = PV alarm (high) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_AL_Act	1 = PV alarm (low) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_Out	Output for process value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_ToleHi	Limit (high) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_ToleLo	Limit (low) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_TH_Act	1 = PV tolerance message (high) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_TL_Act	1 = PV tolerance message (low) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_WH_Act	1 = PV warning (high) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_WL_Act	1 = PV warning (low) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_UnitOut	Unit of measure for process value, for interconnecting to the PV_Unit input parameter of the ConPerMon block	INT	0
RbkOut	Output for position feedback	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
RbkWH_Act	1 = Warning (high) for position feedback active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RbkWL_Act	1 = Warning (low) for position feedback active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP	Setpoint used by controller	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

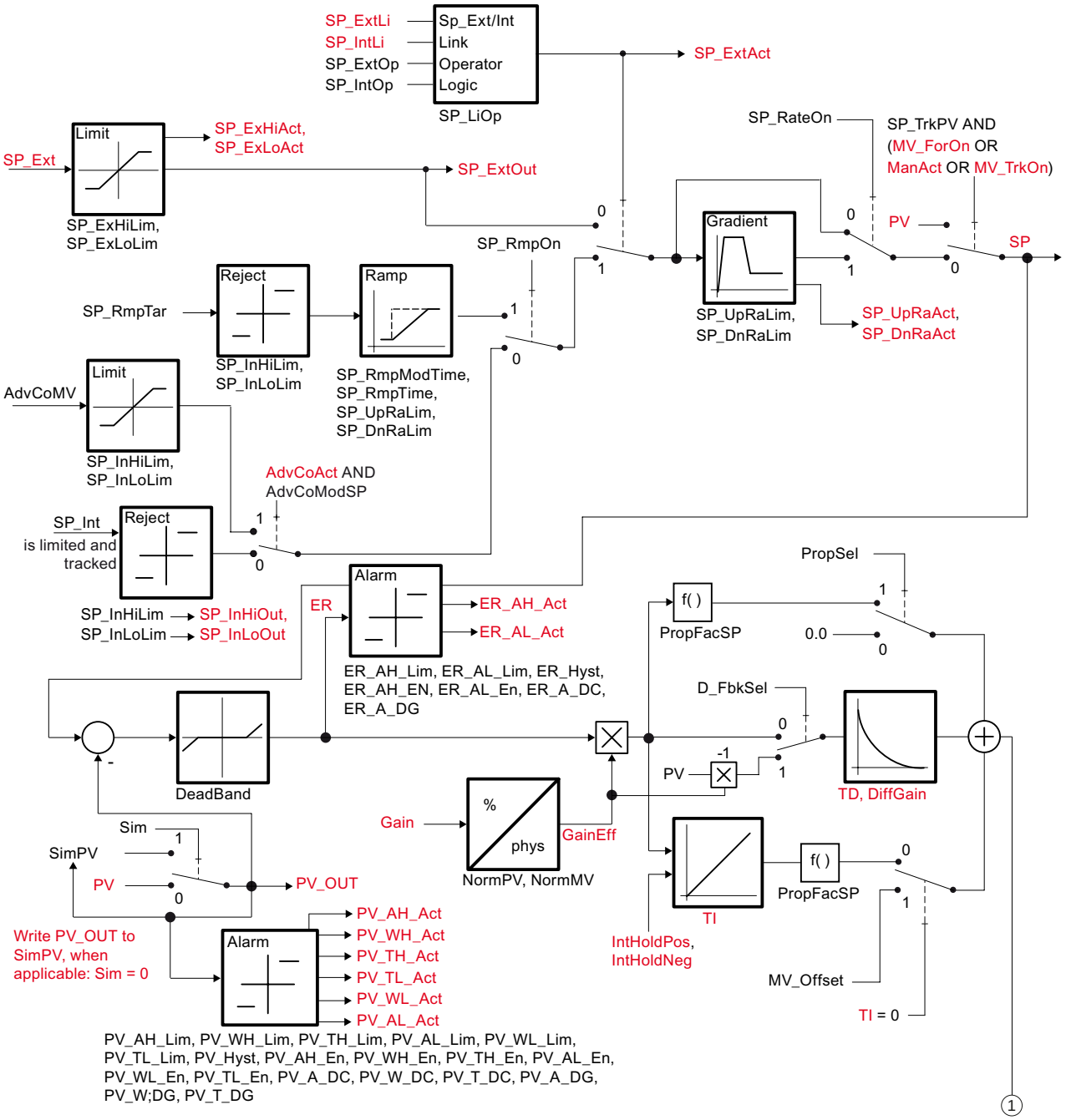
Parameter	Description	Type	Default
SP_DnRaAct	1 = Negative gradient limiting of setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_ExHiAct	1 = Limit (high) for external setpoint has been reached	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_ExLoAct	1 = Limit (low) for external setpoint has been reached	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_ExtAct	1 = External setpoint is active 0 = Internal setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_ExtOut	External setpoint, corresponds to input parameter SP_Ext	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_InHiOut	Limit (high) for SP_Int corresponds to input parameter SP_InHiLim	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
SP_InLoOut	Limit (low) for SP_Int corresponds to input parameter SP_InLoLim	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_UpRaAct	1 = Positive gradient limiting of setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word 1 (Page 428)	DWORD	16#00000000
Status2	Status word 2 (Page 428)	DWORD	16#00000000
SumMsgAct	1 = Active process alarm	STRUCT • Value: BOOL • ST: BYTE	00#80

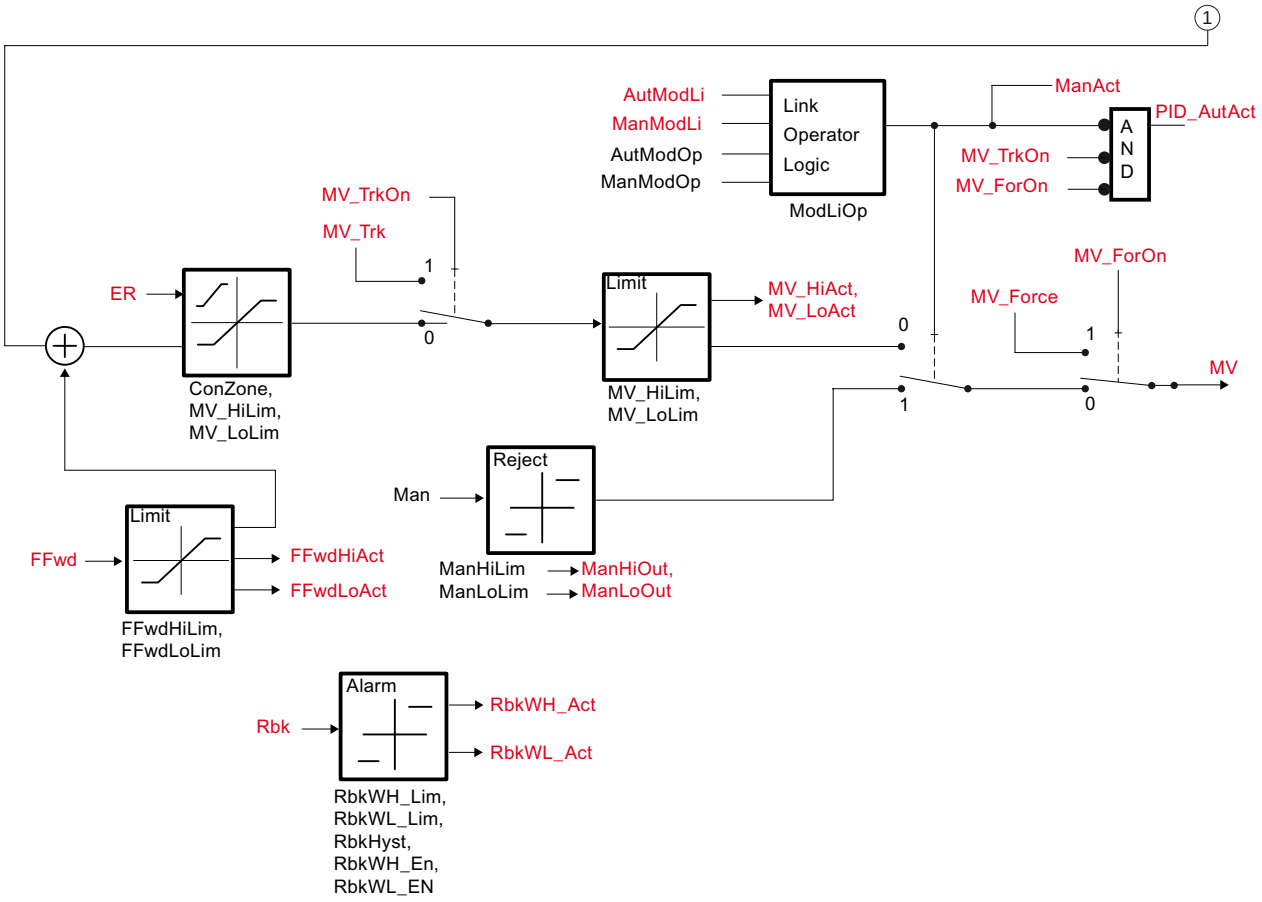
See also

PIDConL messaging (Page 442)
PIDConL block diagram (Page 456)
PIDConL modes (Page 432)

4.6.7 PIDConL block diagram

PIDConLblock diagram





See also

PIDConL I/Os (Page 445)

PIDConL messaging (Page 442)

PIDConL error handling (Page 441)

PIDConL functions (Page 433)

PIDConL modes (Page 432)

Description of PIDConL (Page 428)

4.6.8 Operator control and monitoring

4.6.8.1 PIDConL views

Views of the PIDConL block

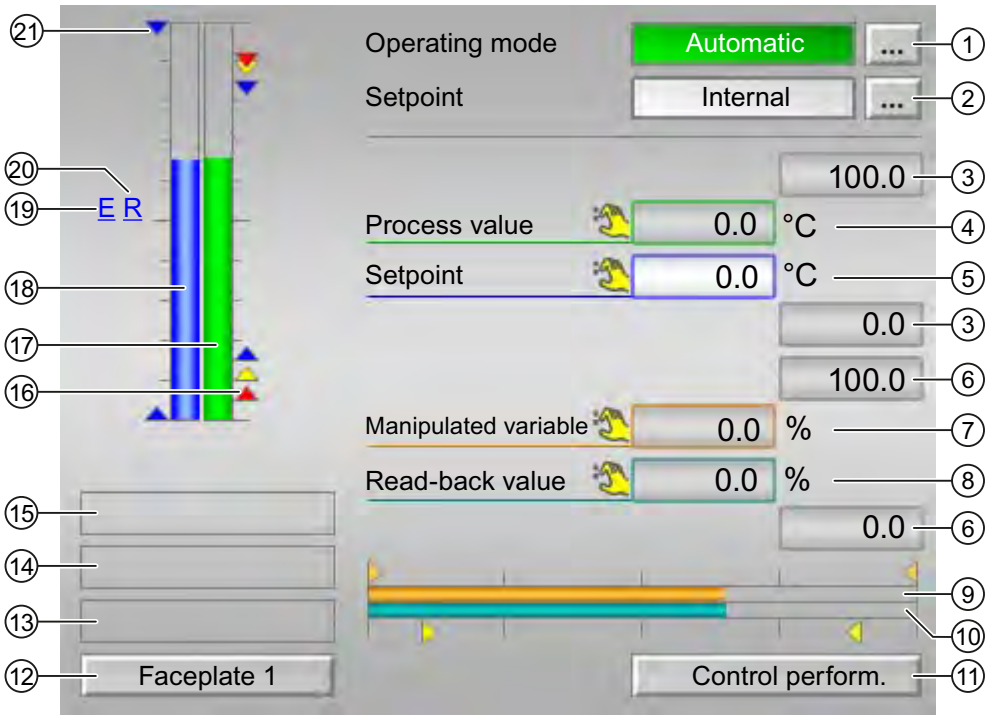
The block PIDConL provides the following views:

- Standard view of PIDConL and PIDConR (Page 459)
- Message view (Page 169)
- Limit value view of PID controllers (Page 160)
- Trend view (Page 172)
- Ramp view (Page 167)
- Parameter view of PID controllers (Page 151)
- Preview of PIDConL and PIDConR (Page 463)
- Memo view (Page 171)
- Batch view (Page 170)
- Block icons for PID and FM controller (Page 181)

Refer to the Structure of the faceplate (Page 123) and Block icon structure (Page 174) sections for general information about the faceplate and block icon.

4.6.8.2 Standard view of PIDConL and PIDConR

Standard view of PIDConL



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 29)
- Automatic mode (Page 29)
- Program mode for closed-loop controllers (Page 34)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Display and switch the setpoint specification

This area shows how to specify the setpoint. The setpoint can specified as follows:

- By the application (External, CFC/SFC)
- By the user direct in the faceplate (Internal).

Refer to the Switching operating states and operating modes (Page 127) section for information on changing the setpoint specification.

You can find additional information on this in the Setpoint input - internal and external (Page 96) section.

Note

With the PIDConR block, this area is only visible if you have set the `Feature` bit Switching operator controls for external setpoint to visible (Page 189) to 1.

(3) High and low scale range for the process value

These values provide information on the display range for the bar graph of the process value. The scale range is defined in the engineering system.

(4) Display of the process value including signal status

This area shows the current process value with the corresponding signal status.

(5) Display and change the setpoint including signal status

This area shows the current setpoint with the corresponding signal status.

Refer to the Changing values (Page 129) section for information on changing the setpoint. The setpoint specification also needs to be set to "Internal" for this block.

(6) High and low scale range for the setpoint

This area is already set and cannot be changed.

(7) Display and change the manipulated variable including signal status

This area shows the current manipulated variable with the corresponding signal status.

Refer to the Changing values (Page 129) section for information on changing the manipulated variable. You can only make a change in manual mode.

(8) Display of the position feedback including signal status

This area shows the current feedback of the manipulated variable with the corresponding signal status.

(9) Bar graph for the manipulated variable

This area shows the current manipulated variable in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(10) Bar graph for the position feedback

This area shows the current position feedback in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(11) Navigation button for switching to the standard view of the ConPerMon block

Use this navigation button to reach the standard view of the ConPerMon block. The visibility of this navigation button depends on the configuration in the engineering system (ES).

See also the Calling further faceplates (Page 43) section for more on this.

(12) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

See also the Calling further faceplates (Page 43) section for more on this.

(13) Display area for states of the block

This area provides additional information on the operating state of the block:

- Maintenance

Additional information on the display area for states of the block is available in section Maintenance release (Page 47).

(14) Display area for states of the block

This area provides additional information on the operating state of the block:

- Simulation

You can find additional information on this in the Simulating signals (Page 93) section.

(15) Display area for states of the block

This area provides additional information on the operating state of the block (from high to low according to priority):

- Optimizing
- Tracking
- Forced tracking

(16) Limit display

These colored triangles indicate the configured limits in the respective bar graph:

- Red: Alarm
- Yellow: Warning
- Blue: Tolerance

(17) Bar graph for the process value

This area shows the current process value in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(18) Bar graph for the setpoint

This area shows the current setpoint in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(19) Display for external setpoint

This display [E] is only visible when you have selected "internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

(20) Display for the target setpoint of the setpoint ramp

This display [R] shows you the target setpoint and is only visible if you have enabled ramp generation in the Ramp view (Page 167).

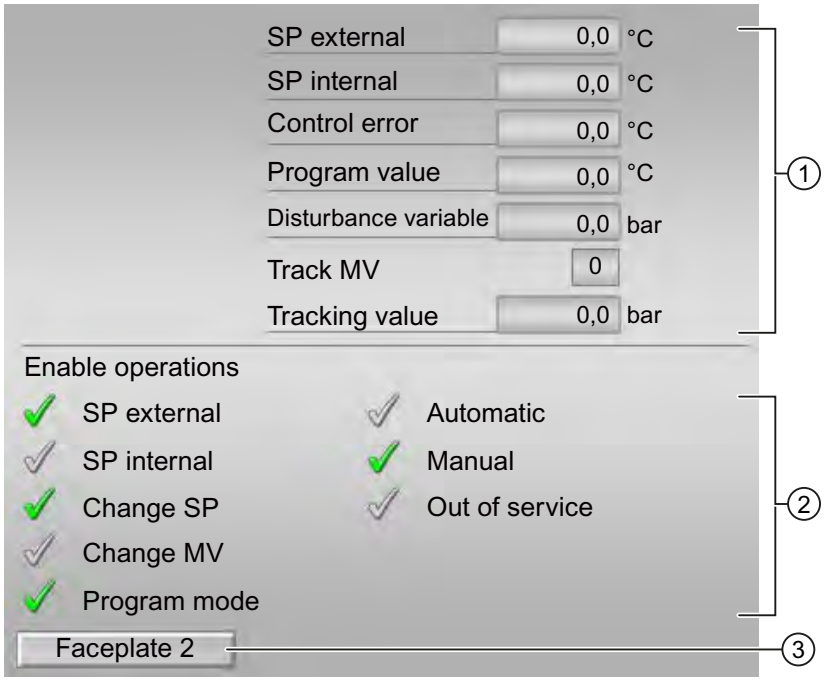
(21) Limit display for the setpoint

These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the ES.

4.6.8.3 Preview of PIDConL and PIDConR

Preview for PID controllers

The preview shows you the parameters that you, as an OS operator, can control. You cannot control anything in this view, however.



(1) Preview area

This area shows you a preview for the following values:

- SP external: currently applicable external setpoint
  - With the PIDConR block, this area is only visible if you have set the `Feature` bit Switching operator controls for external setpoint to visible (Page 189) to 1.
- SP internal: currently applicable internal setpoint
- Control error: Current control error
- Program value: specified value for program mode
- Disturbance: additive value for feedforward control
- Track MV: track manipulated variable (value is 1)
- Tracking value: effective manipulated variable for "Track manipulated variable"

(2) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions.

(3) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

See also the Calling further faceplates (Page 43) section for more on this.

4.7 PIDConR - Continuous PID controller with external reset

4.7.1 Description of PIDConR

Object name (type + number) and family

Type + number: FB 1875

Family: Control

Area of application for PIDConR

The block is used for the following applications:

- Fixed setpoint control
- Cascade control
- Ratio control
- Split-range control
- Smith predictor closed-loop control
- Override control (override)

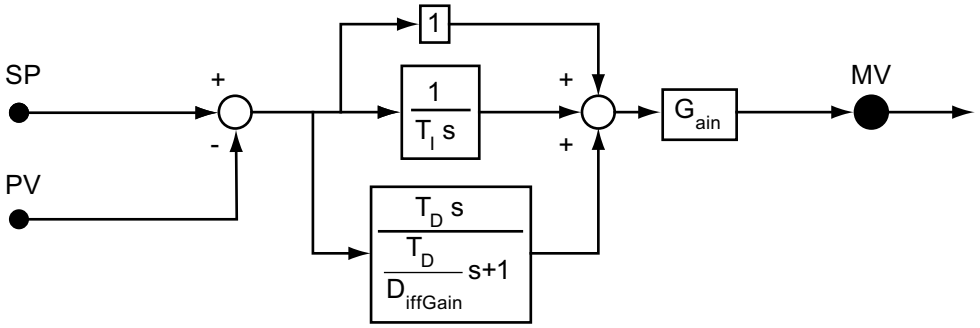
Unlike PIDConL, the PIDConR permits an external reset and satisfies the special requirements of the US market.

How it works

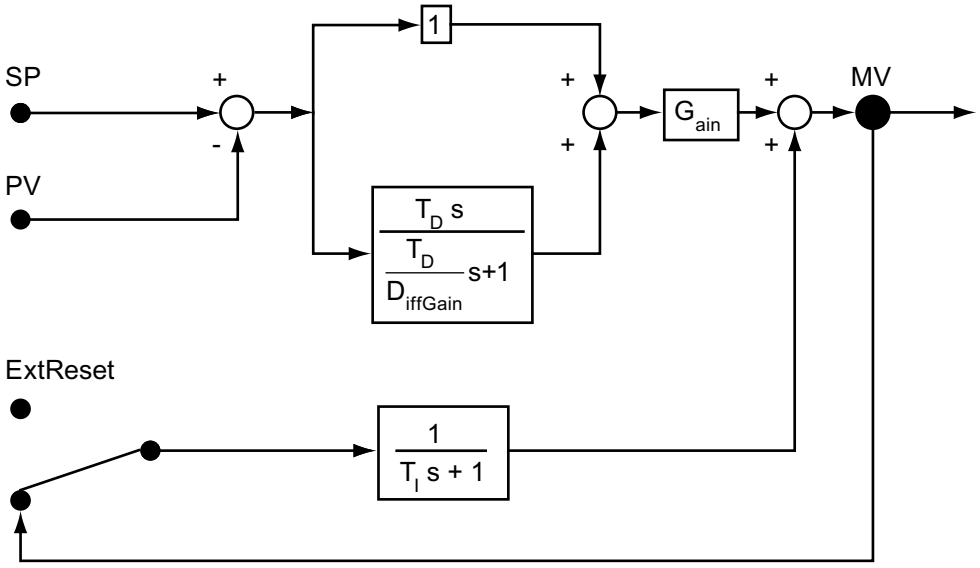
The block is a PID controller with continuous output signal (manipulated variable). It is used to activate a final controlling element with continuous action input.

The block functions following the PID algorithm with a delayed D action and an integrator with double precision. Its particular feature is that it is an incremental control algorithm with a serial-interactive structure. Incremental means that the current manipulated variable is calculated from the old manipulated variable of the last sampling step. In place of the old manipulated variable, an output point for the manipulated variable calculation (external reset) can be specified externally by interconnection. The difference between the parallel controller structure of the PIDConL and the serial-interactive structure of the PIDConR is shown in the next two diagrams.

PIDConL



PIDConR



The block is suitable for controlling sluggish control loops, for example, for temperatures and filling levels, and high-speed control loops, for example, for flow rates and speed. For a given CPU, a compromise has to be made between the number of controllers and the frequency with which the individual controllers have to be processed. The faster the modulated control loops are, i.e. the more frequently the manipulated variables have to be calculated per time unit, the lower the number of controllers that can be installed.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

For the control loop monitoring to work as planned in the trend view of the controller faceplates, the

`S7_xarchive:='Value, shortterm;'`

attributes in the process tag types for control loops at the controller function block must be set for the following tags:

- Input parameters:
  - `CPI_In`
- Output parameters
  - `MV`
  - `MV_HiAct`
  - `MV_LoAct`
  - `AutAct`
  - `SP`
  - `PV_Out`
  - `PV_ToleHi`
  - `PV_ToleLo`

For the PIDConR block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Cascade control with PIDConR (CascadeR) (Page 1444)
- Override control with PIDConR (OverrideR) (Page 1447)
- PIDConR with safety logic and control loop monitoring (PIDConR_ConPerMon) (Page 1431)
- Ratio control with PIDConR (RatioR) (Page 1441)

Note

The meaning of the controller parameters of both structures is different for all controller parameter settings with a D action. If you want to transfer parameter values from one structure to another, they have to be converted according to the formula in the PIDConR function section.

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

Status word allocation for `Status1` parameter

For a description of the individual parameters see

Status bit	Parameter
0	`Occupied`
1	`BatchEn`
2	`SimOn`
3	`OosAct.Value`
4	`OosLi.Value`
5	`AutAct.Value`
6	Not allocated
7	`ManAct.Value`
8-	`SP_ExtAct.Value`
9	`MV_ForOn.Value`
10	`MV_TrkOn.Value AND NOT (ManAct.Value OR OosAct.Value OR MV_ForOn.Value)`
11	`MV.Value > ManLoLim`
12-14	Not allocated
15	SP_LoadOn.Value
16	`SP_LoadOn.Value AND NOT (MV_TrkOn.Value OR OosAct.Value OR MV_ForOn.Value)`
17	`Feature` Bit 19
18	`Feature` Bit 21
19	`AdvCoAct`
20	1 = Input parameter `Rbk` is not interconnected (RbkOut.ST = 16#FF)
21	`NegGain`
22 - 23	Not allocated
24	`OptimOcc`
25 - 31	Not allocated

Status word allocation for `Status2` parameter

Status bit	Parameter
0	MsgLock
1	PV_AH_Act.Value
2	PV_WH_Act.Value
3	PV_TH_Act.Value
4	PV_TL_Act.Value
5	PV_WL_Act.Value
6	PV_AL_Act.Value
7	PV_AH_En
8	PV_WH_En
9	PV_TH_En
10	PV_TL_En
11	PV_WL_En
12	PV_AL_En
13	PV_AH_MsgEn
14	PV_WH_MsgEn
15	PV_TH_MsgEn
16	PV_TL_MsgEn
17	PV_WL_MsgEn
18	PV_AL_MsgEn
19	ER_AH_Act.Value
20	ER_AL_Act.Value
21	ER_AH_En
22	ER_AL_En
23	ER_AH_MsgEn
24	ER_AL_MsgEn
25	RbkWH_Act.Value
26	RbkWL_Act.Value
27	RbkWH_En
28	RbkWL_En
29	RbkWH_MsgEn
30	RbkWL_MsgEn
31	MS_RelOp

See also

PIDConR block diagram (Page 500)
PIDConR I/Os (Page 489)
PIDConR messaging (Page 486)
PIDConR error handling (Page 485)
PIDConR functions (Page 474)
PIDConR modes (Page 471)

4.7.2 PIDConR modes

PIDConR modes

The block can be operated using the following modes:

- Automatic mode (Page 29)
- Manual mode (Page 29)
- Program mode for closed-loop controllers (Page 34)
- Out of service (Page 27)

In "manual mode", the control settings for the device are made manually by the operator. The operator decides how to change the block's manipulated variable (output signal).

In "automatic mode", the controller's manipulated variable is calculated automatically by the block algorithm.

Changing between operating modes

The switchover between manual and automatic modes takes place as shown in the following schematic:

Changeover by using faceplates: The changeover between operating modes is carried out in the standard view of the faceplate. In the function block, the parameters `ManModOp` for "manual mode" and `AutModOp` for "automatic mode" are used.

Switchover per interconnection (CFC or SFC instance): The changeover between the operating modes is carried out by means of interconnection on the function block.

Note

You can access the variable parameters `AutModOp` and `ManModOp` from a normal SFC (in contrast to the instance of an SFC type). The SFC can thus change the operating mode without revoking the access rights of the operator.

Points to note for this block

With PIDConR the changeover between operating modes via interconnectable input parameters follows a different logic than that used for other controller blocks. This logic is oriented towards the special requirements of the US market. The basic approach is to issue the controller with certain commands by interconnection using an individual input parameter. The commands issued by interconnection take priority over the entries in the faceplate, i.e. if such a command input is made, the associated operator controls are locked in the faceplate:

- If the interconnectable `AutExtSet = 1` input parameter is set, the controller goes into automatic mode with an external setpoint. This is also known as the "cascade" operating mode.
- If the interconnectable `AutIntSet = 1` input parameter is set, the controller goes into automatic mode with an internal setpoint. `AutIntSet` has higher priority than `AutExtSet`. For details of the internal and external setpoint, see section (link!) PIDConR functions.
- If the interconnectable `ManSet = 1` input parameter is set, the controller goes into manual mode. This command has higher priority than `AutIntSet` and `AutExtSet`.
- If one of the interconnectable input parameters `MV_Close` or `MV_Open` is set, the controller also goes into manual operating mode and performs the corresponding command.

The `ModLiOp`, `ManModLi` and `AutModLi` input parameters are not therefore present in PIDConR.

Switchover from automatic mode to manual mode

When changing over from "automatic mode" to "manual mode", the last valid control settings (**M**anipiulated **V**alue `MV`)for the controller set in "automatic mode" remain valid until you change the control settings manually.

Switchover from automatic mode to manual mode

The switchover from manual to automatic mode can take place with or without the internal setpoint tracking the process value. You specify this behavior on the `SP_TrkPV` I/O, which can also be operated from the faceplate in the parameter view (Option "SP := PV in Manual").

- **Switchover with internal setpoint tracking process variable** (`SP_TrkPV = 1`) means that in "Manual" mode the setpoint (SP) tracks the process variable (PV) (bumpless switchover). After switching back to "Automatic" mode, the manipulated variable remains constant until the setpoint value (SP) is changed or the process value (PV) changes.
- **Switchover without internal setpoint tracking process variable** (`SP_TrkPV = 0`) means that the block immediately recalculates the value of the manipulated variable based on the setpoint and process value (PV) when the mode is changed. PIDConR only offers **switchover without P step**: During switchover, the I action of the controller is set in such a way that the switchover is carried out without a P step (virtually bumpless referring to the manipulated variable). A control deviation is only regulated via the I action.

Reaction of signals when operating mode is changed

Using the `Feature` bit Resetting the commands for changing the mode (Page 193), you can choose whether the block automatically resets the signal for changing the operating mode.

Program mode for controllers

You can find general information about the "Program mode for controller" in the section Program mode for closed-loop controllers (Page 34).

Out of service

You can find general information about the "Out of service" mode in the section Out of service (Page 27).

See also

PIDConR block diagram (Page 500)

PIDConR I/Os (Page 489)

PIDConR messaging (Page 486)

PIDConR error handling (Page 485)

PIDConR functions (Page 474)

Description of PIDConR (Page 465)

Disabling bumpless changeover to automatic mode for controllers (Page 197)

4.7.3 PIDConR functions

Functions of PIDConR

The functions for this block are listed below.

Generation of manipulated variables

The manipulated variable MV can be generated as follows:

MV_ ForOn	MV_ Close	MV_ Open	ManAct	MV_ TrkOn	AdvCoAct AND NOT AdvCoModSP	MV	Limit monitoring	State
1	-	-	-	-	-	MV_Forced	none	Forced tracking through constraint without limitation
0	1	-	-	-	-	ManLoLim	ManHiLim ManLoLim	Close by interconnection
0	0	1	-	-	-	ManHiLim	ManHiLim ManLoLim	Open by interconnection
0	0	0	1	-	-	Man	ManHiLim ManLoLim	Manual mode, set by the OS operator or via the ManSet = 1 command
0	0	0	0	1	-	MV_Trk	MV_HiLim MV_LoLim	Tracking with limitation
0	0	0	0	0	1	AdvCoMV	MV_HiLim MV_LoLim	Higher-level program mode
0	0	0	0	0	0	PID.OUT + FFwd	MV_HiLim MV_LoLim	Automatic mode (PID algorithm)

If the controller is in "out of service" mode, the output parameter MV is set to the last valid value in manual mode or the safety manipulated variable depending on the Feature bit Safety value of the manipulated variable effective at startup (Page 196). Refer to the Out of service (Page 27) section for more on this.

The PIDConR offers the following special ways of influencing the generation of manipulated variables via interconnectable input parameters. The commands issued by interconnection take priority over the entries in the faceplate, i.e. if such a command input is made, the associated operator controls are locked in the faceplate.

MV_BumpOn is used for sudden manipulation of the manipulated variable and has a similar effect to MV_ForOn, the difference being that the appropriate limits are applied. If MV_BumpOn = 1 is set in automatic mode, the MV_Bump manipulated variable is written in a limited manner between MV_HiLim and MV_LoLim to output MV. If MV_BumpOn = 1 is set in manual mode, the MV_Bump manipulated variable is written in a limited manner between ManHiLim and ManLoLim to output MV. If MV_BumpOn is reset to 0, the controller returns to its previous mode.

MV_Close is used to close the adjustment valve. MV_Close = 1 switches the controller into manual mode with manipulated variable MV = ManLoLim. If MV_Close is reset to 0, the controller remains in manual mode.

MV_Open is used to open the adjustment valve. `MV_Open = 1` switches the controller into manual mode with manipulated variable `MV = ManHiLim`. If `MV_Open` is reset to 0, the controller remains in manual mode.

These commands have higher priority than automatic mode, but lower priority than forced tracking by `MV_ForOn`.

For meshed controller structures, such as a cascade control and closed-loop control with the PIDConR block, the `ExtReset` input parameter is used with `ExtRstOn = 1` rather than `MV_Trk` with `MV_TrkOn = 1` (also refer to the process tag types Cascade control with PIDConR (CascadeR) (Page 1444) and Override control with PIDConR (OverrideR) (Page 1447)).

Displaying additional information relating to the manipulated variable on the output

The manual manipulated variable limitations `ManLoLim` and `ManHiLim` are copied to the `ManLoOut` and `ManHiOut` output parameters so that they can be further interconnected to the secondary controller as setpoint limits `SP_ExtLoLim` and `SP_ExtHiLim`. If you want to use the same limit pairs for manual and automatic mode, you can interconnect output parameters `ManLoOut` and `ManHiOut` to input parameters `MV_LoLim` and `MV_HiLim` of the same block and thereby control the limits for manipulated variable limitation in the faceplate. In most cases, a valve's complete adjustment range can be passed through in manual mode. Then you can further interconnect the `ManLoOut` and `ManHiOut` output parameters to the `LoScale` and `HiScale` input parameters of the assigned analog output driver. Caution: If the valves have an open safe position (`SafePos = 1`), this interconnection must be crossed over: `LoScale` = `ManHiOut` and `HiScale` = `ManLoOut`. On the controller, 0% is always interpreted as valve closed and 100% as valve open but the driver then issues a control signal with a 0% for a 100% controller manipulated variable.

Tracking and limiting a manipulated variable

The block provides the standard function Tracking and limiting a manipulated value (Page 111).

Safe position

The block provides the standard function Safe position for motors, valves and controllers (Page 120).

"Actuator active" information

If the manipulated variable `MV` is greater than the minimum manual limit `ManLoLim`, this is recognized as actuator active. This status can be used to display a custom icon in the process image, for example, and is stored in the status word (you can find additional information under "Status word" in the Description of PIDConR (Page 465) section).

Limit monitoring of position feedback

The block provides the standard function Limit monitoring of the feedback (Page 76).

External/internal setpoint specification

This input of setpoints is carried out either using a CFC/SFC program or using the faceplate (operator). The operator can specify an internal setpoint value (SP_Int) or a higher-level open-loop control will specify an external setpoint value (SP_Ext).

Setpoint input internally and externally using faceplate

With PIDConR, the setpoint signal source is selected via the block just as it is with other controllers. The SP_IntOp = 1 parameter is used on the function block for the internal setpoint input and SP_ExtOp = 1 for the external setpoint input.

Setpoint input internally and externally using interconnection

With PIDConR the changeover between internal and external setpoint via interconnectable input parameters follows a different logic than that used for other controller blocks. This logic is oriented towards the special requirements of the US market.

The setpoint signal source (internal/external) can be selected by interconnection along with selection of the operating mode using the AutIntSet input parameter for automatic with an internal setpoint and AutExtSet for automatic with an external setpoint (additional information is available in the PIDConR modes (Page 471) section). The commands issued by interconnection take priority over the entries in the faceplate, i.e. if such a command input is made, the associated operator controls are locked in the faceplate. Input parameters SP_LiOp, SP_ExtLi and SP_IntLi on PIDConR are not therefore needed.

The PIDConR also offers a special way of influencing the setpoint input via interconnectable input parameters. Loading setpoints. If the SP_LoadOn = 1 input parameter is set, the controller goes into automatic mode. The value of the SP_Load input parameter is limited according to an internal setpoint and used for control purposes. When the SP_LoadOn parameter changes back from 1 to 0, the controller remains in automatic mode and changes the setpoint specification back to the internal setpoint.
Loading a setpoint via SP_LoadOn takes priority over all other forms of setpoint specification.

Bumpless switchover from external to internal setpoint

The parameter SP_TrkExt = 1 is used so that the internal setpoint tracks the external setpoint to achieve a bumpless switchover from the external to the internal setpoint. This allows unwanted jumps at the output parameter to be avoided.

Setpoint limiting for external setpoints

The block provides the standard function Setpoint limiting for external setpoints (Page 110).

Gradient limit of the setpoint

The block provides the standard function Ramp limiting of the setpoint (Page 108).

Using setpoint ramp

The block provides the standard function Using a setpoint ramp (Page 98).

Tracking setpoint in manual mode

The block provides the standard function Setpoint tracking the process variable in manual mode (Page 110).

Simulating signals

The block provides the standard function Simulating signals (Page 93).

You can simulate the following values:

- Process value (`SimPV`)
- Position feedback (`SimRbk`)

Limit monitoring of the process value

The block provides the standard function Limit monitoring of the process value (Page 70).

The PIDConR is the only block to have separate input parameters for the alarm delay at the high and low limits.

Control error generation and dead band

The block provides the standard function Control error generation and deadband (Page 109).

Limit monitoring of control error

The block provides the standard function Monitoring the error signal limits (Page 77).

Inverting control direction

The block provides the standard function Inverting the control sense (Page 108).

Physical standardization of setpoint, manipulated variable and process value

Controller gain `Gain` is entered either using a physical variable or as standardized value.

`Gain` as physical variable [`MV_Unit/PV_Unit`]:

The standardized variables retain their default values:

- `NormPV.High = 100` and `NormPV.Low = 0`
- `NormMV.High = 100` and `NormMV.Low = 0`

The effective gain is:

```
GainEff = Gain
```

Entering a standardized `Gain` (dimensionless):

Change the standardized variables to the actual range of the process values and manipulated variables.

- Internal and external setpoints; the process value and corresponding parameters are entered according to the physical measuring range of the process value.
- The manual value, the tracking value of the manipulated variable, feedforward control and the corresponding parameters are set according to the physical measuring range of the manipulated variable.

The effective gain is:

```
GainEff = (NormMV.High - NormMV.Low) / (NormPV.High - NormPV.Low) · Gain
```

Selecting a unit of measure

The block provides the standard function Selecting a unit of measure (Page 53).

PID algorithm

The manipulated variable is generated in automatic mode according to the following algorithm:

$$MV = \quad = G_{ain}\left[1+\frac{T_D s}{\frac{T_D}{D_{iffGain}}s+1}\right](PV-SP)+R_{eset}\left(\frac{1}{T_I s+1}\right)$$

Where:

s = complex number of the Laplace transformation

This formula describes a standard application where P, I and D action is activated

Unlike with PIDConL, the `Gain` gain factor is not applied to the I action.

The D action delay is derived from `TD/DiffGain`.

- The P action can be deactivated by `PropSel = 0`.
- The I action can be deactivated by `IntSel = 0`. The `MV_Offset` input parameter can then be used to add a constant value to the manipulated variable. Select this value such that the remaining control error equals zero at least at the control loop's typical operating point.
- The D action can be deactivated by `TD = 0`.

If you set the `Feature` bit Activating bumpless changes to proportional gain (Page 196) parameters to 1, changes in the proportional gain `Gain` are carried out in a bumpless manner in automatic mode by converting the internal reset.

If a GainSched block is linked to the controller, you have to make the parameter settings for the `Feature` bit Activating bumpless changes to proportional gain (Page 196) 0.

The PID core algorithm is implemented in the PIDKernR function block which in turn calls a series of auxiliary functions for the 64-bit arithmetic. Every edge transition on the `InitPid` input parameter forces initialization equations to be carried out by PIDKernR. In so doing, the internal reset is calculated such that the `MV` output does not suffer a P step

Note

The meaning of the controller parameters with PIDConR is different to that with the other PID controllers for all controller parameter settings with a D action. If you want to transfer parameter values from one kind of controller to another, they have to be converted using the following formulas. The calculation is based on a simplified transfer function without a delay in the D action, and all three control channels are applied to the `ER= SP-PV` control error.

Parallel controller structure (e.g. PIDConL):

$$MV = G_{ain}\left(1+\frac{1}{T_I s}+T_D s\right)ER$$

The parameters of the serial-interactive controller structure (PIDConR)

$$MV = G'_{ain}\left(1+\frac{1}{T'_I s}\right)\left(T'_D s+1\right)ER$$

are marked using a speech mark. Both controllers calculate the same manipulated variable if the parameter values of the serial-interactive structure are determined by the following substitution:

$$G'_{ain} = \alpha G_{ain}\ ,\quad T'_I = \alpha T_I\ ,\quad T'_D = \frac{1}{\alpha}T_D$$

using the conversion factor

$$\alpha = \frac{1}{2}+\sqrt{\frac{1}{4}-\frac{T_D}{T_I}}$$

You can immediately see that the conversion factor only equals one if `TD = 0`. The conversion is also possible in the other direction with

$$\alpha = \frac{T'_I}{T'_I + T'_D}$$

This conversion is performed automatically in the PID tuner.

Use output point for the manipulated variable calculation (external reset)

For the `ExtResOn = 0` input parameter (default setting), the starting point for the manipulated variable calculation within the block is taken from manipulated variable `MV`. In other words, this is the manipulated variable of the last sampling step. If `ExtResOn = 1`, the `ExtReset` start parameter is used. This is used in particular for networked control structures, such as cascade or transfer control.

Anti-windup

A controller with incremental algorithm (external reset) inherently has an anti-windup reaction because the starting point for the manipulated variable calculation (external reset value) is limited provided it is taken internally from manipulated variable `MV` or is taken from another signal source with limitation. If the starting point for the manipulated variable calculation (external reset value) is at the limit (`MV_HiLim` or `MV_LoLim`), the I action is automatically frozen.

Activating and limiting disturbance variables

The block provides the standard function Activating and limiting disturbance variable (Page 113).

Forming the signal status for blocks

The block provides the standard function Forming and outputting signal status for blocks (Page 88).

- Signal status for the process value PV_Out:

  The signal status of output parameter PV_Out is always equivalent to the signal status of input parameter PV or, if the block is in simulation mode, 16#60.

- Signal status for the setpoint value SP:

  The signal status of SP output parameter is always equivalent to the signal status of input parameter SP_Ext or SP_Int , depending on how the setpoint is specified. If the internal setpoint SP_Int is used, the signal status is always output as 16#80.

- Signal status of the system deviation ER:

  The signal status of output parameterER is obtained from the worse signal status of the two output parameters PV_Out and SP and is output.

- Worst signal status:

  The worst signal status ST_Worst for the block is formed by the following parameters:

  – ER.ST
  – FFwd.ST
  – Rbk.ST

Configurable reactions using the Feature parameter

You can find an overview of all reactions provided by the Feature parameter in section Functions that can be set via the Feature I/O (Page 186) The following reactions are available to you for this block at the corresponding bits:

Bit	Function
0	Setting the startup response (Page 187)
1	Reaction to the out of service mode (Page 199)
2	Resetting the commands for changing the mode (Page 193)
15	Safety manipulated variable with "out of service" operating mode in effect (Page 196)
16	Safety value of the manipulated variable effective at startup (Page 196)
17	Safety manipulated variable with "out of service" operating mode in effect (Page 196)
19	Enabling program mode (Page 192)
20	Activating bumpless changes to proportional gain (Page 196)
21	Switching operator controls for external setpoint to visible (Page 189)
22	Update acknowledgment and error status of the message call (Page 193)

Operator control permissions via parameters OS_Perm and OS1Perm

The block has the following operator control permissions (Page 45) for the parameter `OS_Perm`:

Bit	Function
0	1 = Operator can switch into automatic mode `AutModOp`
1	1 = Operator can switch into manual mode `ManModOp`
2	1 = Operator can switch into "out of service" mode `OosOp`
3	1 = Operator can switch into program mode `AdvCoOn`
4	1 = Operator can switch the setpoint to external `SP_ExtOp`
5	1 = Operator can switch the setpoint to internal `SP_IntOp`
6	1 = Operator can change the internal setpoint `SP_Int`
7	1 = Operator can change the manual parameter `Man`
8	1 = Operator can change maximum usage limit of the setpoint `SP_InHiLim`
9	1 = Operator can change minimum usage limit of the setpoint `SP_InLoLim`
10	1 = Operator can change maximum usage limit of the manipulated variable `ManHiLim`
11	1 = Operator can change minimum usage limit of the manipulated variable `ManLiLim`
12	1 = Operator can use the setpoint's gradient limitation function `SP_RateOn`
13	1 = Operator can raise the gradient limit `SP_UpRaLim` of the setpoint
14	1 = Operator can lower the gradient limit `SP_DnRaLim` of the setpoint
15	1 = Operator can switch between the time value or the gradient value for specifying the ramp `SP_RmpModTime`
16	1 = Operator can change the ramp time `SP_RmpTime`
17	1 = Operator can change the target setpoint `SP_RmpTarget` for the setpoint ramp
18	1 = Operator can activate the setpoint ramp function `SP_RmpOn`
19	1 = Operator can permit the PID optimization function `OptimEn`
20	1 = Operator can activate the track setpoint in manual mode function `SP_TrkPV`
21	1 = Operator can activate the bumpless changeover from external to internal function `SP_TrkExt`
22	1 = Operator can change the gain parameter `Gain`
23	1 = Operator can change the integral action time parameter `TI`
24	1 = Operator can change the derivative action time parameter `TD`
25	1 = Operator can change the derivative gain parameter `DiffGain`
26	1 = Operator can change the dead band parameter `DeadBand`
27 - 31	Not allocated

The block has the following operator control permissions for the OS1Perm parameter:

Bit	Function
0	1 = Operator can change the limit (process value) for the high alarm PV_AH_Lim
1	1 = Operator can change the limit (process value) for the high warning PV_WH_Lim
2	1 = Operator can change the limit (process value) for the high tolerance PV_TH_Lim
3	1 = Operator can change the hysteresis (process value) PV_Hyst
4	1 = Operator can change the limit (process value) for the low tolerance PV_TL_Lim
5	1 = Operator can change the limit (process value) for the low warning PV_WL_Lim
6	1 = Operator can change the limit (process value) for the low alarm PV_AL_Lim
7	1 = Operator can change the limit (control error) for the high alarm ER_AH_Lim
8	1 = Operator can change the hysteresis (control error) ER_Hyst
9	1 = Operator can change the limit (control error) for the low alarm ER_AL_Lim
10	1 = Operator can change the limit (position feedback) for the high warning RbkWH_Lim
11	1 = Operator can change the hysteresis (position feedback) RbkHyst
12	1 = Operator can change the limit (position feedback) for the low warning RbkWL_Lim
13 - 15	Not allocated
16	1 = Operator can activate the simulation function SimOn
17	1 = Operator can activate the maintenance release function MS_RelOp
18 - 31	Not allocated

Maintenance release

The block provides the standard function Maintenance release (Page 47).

Generating instance-specific messages

The block provides the standard function Generating instance-specific messages (Page 46) without the time stamp function in the I/O devices.

Specifying the display area for process and setpoint values as well as operations

This block provides the standard function Display and operator input area for process values and setpoints (Page 42).

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 122).

See also

PIDConR I/Os (Page 489)
PIDConR messaging (Page 486)
PIDConR error handling (Page 485)
Setpoint input - internal and external (Page 96)
Program mode for closed-loop controllers (Page 34)
Disabling bumpless changeover to automatic mode for controllers (Page 197)

4.7.4 PIDConR error handling

PIDConR troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` output parameter can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Default value when implementing the block; block will not be processed.
0	There is no error.
30	The value of `PV` can no longer be displayed in the REAL number field.
31	The value of `SP_Ext` can no longer be displayed in the REAL number field.
32	The value of `FFwd` can no longer be displayed in the REAL number field.
33	The value of `MV_Trk` can no longer be displayed in the REAL number field.
34	The value of `MV_Forced` can no longer be displayed in the REAL number field.
35	The value of `Rbk` can no longer be displayed in the REAL number field.
36	The value of `MV` can no longer be displayed in the REAL number field.
60	`|Tl| < SampleTime/2`
61	`|TD| < SampleTime`
62	`DiffGain < 1` or `DiffGain > 10`
63	`TD/DiffGain < SampleTime/2`
66	`NormPV_High = NormPV_Low`

See also

PIDConR block diagram (Page 500)

PIDConR I/Os (Page 489)

PIDConR messaging (Page 486)

PIDConR functions (Page 474)

PIDConR modes (Page 471)

Description of PIDConR (Page 465)

4.7.5 PIDConR messaging

Messaging

The following messages can be generated for this block:

- Control system fault
- Process messages
- Instance-specific messages

Control system fault

The following control system error messages can be output:

Message instance	Message identifier	Message class	Event
MsgEvId2	SIG 6	AS process control message - fault	$$BlockComment$$ External error has occurred

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter CSF. If this signal changes to CSF = 1, a control system fault is triggered (MsgEvId2, SIG 6).

Process messages

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 1	Alarm - high	$$BlockComment$$ PV - high alarm limit violated
	SIG 2	Warning - high	$$BlockComment$$ PV - high warning limit violated
	SIG 3	Tolerance - high	$$BlockComment$$ PV - high tolerance limit violated
	SIG 4	Tolerance - low	$$BlockComment$$ PV - low tolerance limit violated
	SIG 5	Warning - low	$$BlockComment$$ PV - low warning limit violated
	SIG 6	Alarm - low	$$BlockComment$$ PV - low alarm limit violated
	SIG 7	Alarm - high	$$BlockComment$$ ER - high alarm limit violated
	SIG 8	Alarm - low	$$BlockComment$$ ER - low alarm limit violated

Message instance	Message identifier	Message class	Event
MsgEvId2	SIG 7	Warning - high	$$BlockComment$$ Rbk - high warning limit violated
	SIG 8	Warning - low	$$BlockComment$$ Rbk - low warning limit violated

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Instance-specific messages

You can use up to four instance-specific messages with this block.

Message instance	Message identifier	Message class	Event
MsgEvId2	SIG 1	AS process control message - fault	$$BlockComment$$ External message 1 status 16#@5%x@
	SIG 2	AS process control message - fault	$$BlockComment$$ External message 2 status 16#@6%x@
	SIG 3	AS process control message - fault	$$BlockComment$$ External message 3 status 16#@7%x@
	SIG 4	AS process control message - fault	$$BlockComment$$ External message 4 status 16#@8%x@

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Associated values for message instance `MsgEvId1`

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`
4	Process value `PV_Out`
5	Control error `ER`
6	`ExtVa106`
7	`ExtVa107`
8	Not allocated
9	Not allocated
10	Not allocated

The associated values 6 and 7 are assigned to the parameters `ExtVa106` ... `ExtVa107` and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

Associated values for message instance `MsgEvId2`

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`
4	Readback value Rbk
5	Signal status `ExtMsg1`
6	Signal status `ExtMsg2`
7	Signal status `ExtMsg3`
8	Signal status `ExtMsg4`
9	`ExtVa209`
10	`ExtVa210`

The associated values 9 and 10 are assigned to the parameters `ExtVa209` ... `ExtVa210` and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

See also

PIDConR block diagram (Page 500)
PIDConR I/Os (Page 489)
PIDConR error handling (Page 485)
PIDConR functions (Page 474)
PIDConR modes (Page 471)
Description of PIDConR (Page 465)

4.7.6 PIDConR I/Os

PIDConR I/Os

Input parameters

Parameter	Description	Type	Default
AdvCoEn	1 = Enable program mode via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AdvCoOn	1 = Enable program mode via faceplate	BOOL	0
AdvCoModSP	Type of program mode: 1 = Setpoint specification 0 = Manipulated variable specification	BOOL	1
AdvCoMstrOn	1 = Activate (0-1) or deactivate (1-0) program via edge transition	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AdvCoMV	Specified value from the external program	REAL	0.0
AutExtSet	1 = Activate automatic mode with external setpoint by interconnection (SP = SP_Ext)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AutIntSet	1 = Activate automatic mode with internal setpoint by interconnection (SP = SP_Op). AutIntSet has higher priority thanAutExtSet.	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AutModOp	1 = Automatic mode by operator	BOOL	0
BatchEn	1 = Enable allocation for batch control	BOOL	0
BatchID	Batch ID	DWORD	16#00000000
BatchName	Batch name	STRING[32]	''
CPI_In	Input for control performance index, which is calculated by the assigned ConPerMon block	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#78
CSF	1 = External error (control system error)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
DeadBand	Width of dead band	REAL	0.0
DiffGain	Gain of differentiator [1...10] DiffGain = TD / (differentiation of D action)	STRUCT • Value: REAL • ST: BYTE	- • 5.0 • 16#80
EN	1 = Called block will be processed	BOOL	1
ER_A_DC	Delay for incoming alarms during control error monitoring	REAL	0.0
ER_A_DG	Delay for outgoing alarms during control error monitoring	REAL	0.0

Parameter	Description	Type	Default
ER_AH_En	1 = Activate alarm (high) for control error monitoring	BOOL	1
ER_AH_Lim	Alarm limit (high) for control error monitoring	REAL	100.0
ER_AH_MsgEn	1 = Activate messages for alarm (high) for control error monitoring	BOOL	1
ER_AL_En	1 = Activate alarm (low) for control error monitoring	BOOL	1
ER_AL_Lim	Alarm limit (low) for control error monitoring	REAL	-100.0
ER_AL_MsgEn	1 = Activate messages for alarm (low) for control error monitoring	BOOL	1
ER_Hyst	Alarm hysteresis for control error	REAL	1.0
ExtMsg1	1 = Binary input for freely selectable message 1 is used	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg2	1 = Binary input for freely selectable message 2 is used	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg3	1 = Binary input for freely selectable message 3 is used	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg4	1 = Binary input for freely selectable message 4 is used	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtReset	Value to which a reset is made if ExtRstOn = 1.	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
ExtRstOn	1 = Reset externally	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtVa106	Associated value 6 for messages (MsgEvID1)	ANY	
ExtVa107	Associated value 7 for messages (MsgEvID1)	ANY	
ExtVa209	Associated value 9 for messages (MsgEvID2)	ANY	
ExtVa210	Associated value 10 for messages (MsgEvID2)	ANY	
Feature	I/O for additional functions (Page 474)	STRUCT • Bit 0: BOOL • ... • Bit 20: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 1 • 0 • 0
FFwd	Input for additive disturbance variable activation	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

Parameter	Description	Type	Default
FFwdHiLim	Limit (high) for additive disturbance variable activation	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
FFwdLoLim	Limit (low) for additive disturbance variable activation	STRUCT • Value: REAL • ST: BYTE	- • -100.0 • 16#80
Gain	Proportional gain Gain.ST=16#FF: Operable in faceplate	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#FF
InitPid	InitPid edge transitions result in the PID algorithm's initialization equations being carried out. Is used for SP changes without MV jumps for example	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
IntSel	1 = I action activated	BOOL	1
Man	Manual specification for the manipulated variable	REAL	0.0
ManHiLim	Limit (high) for manual parameter Man	REAL	100.0
ManLoLim	Limit (low) for manual parameter Man	REAL	0.0
ManModOp	1 = Manual mode by OS operator	BOOL	1
ManSet	1 = Activate manual mode via interconnection. ManSet has higher priority thanAutIntSet.	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_RelOp	1 = Maintenance release by OS operator	BOOL	0
MsgEvId1	Message number (assigned automatically)	DWORD	16#FFFFFFFF
MsgEvId2	Message number (assigned automatically)	DWORD	16#FFFFFFFF
MsgLock	1 = Suppress process messages	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_Bump	Default value for controller output MV if MV_BumpOn = 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV_BumpOn	1 = Set controller output MV:= MV_Bump without (!) taking the controller into manual mode.	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_Close	1 = Close adjustment valve by interconnection, i.e. MV:= MV_LoLim MV_Close has higher priority than MV_Open	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_Forced	Forced manipulated variable that is not limited and assumes top priority	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV_ForOn	1 = Forced manipulated variable MV_Forced output unlimited at output MV	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
MV_HiLim	Limit (high) for manipulated variable MV	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
MV_LoLim	Limit (low) for manipulated variable MV	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV_Offset	Manipulated variable for ER = 0, operating point for controller with deactivated I action	REAL	0.0
MV_Open	1 = Open adjustment valve by interconnection, i.e. MV:= MV_HiLim MV_Open has higher priority than MV_TrkOn	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_OpScale	OS display range for manipulated variable MV	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
MV_Trk	Tracking value for the manipulated variable MV	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV_TrkOn	1 = Tracking of manipulated variable MV	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_Unit	Unit of measure for manipulated variable	INT	1342
NegGain	0 = Positive controller gain: ER = Gain · (SP - PV) 1 = Negative controller gain: ER = Gain · (PV - SP)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
NormMV	Manipulated variable range (MV) for standardizing the proportional gain (GAIN)	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
NormPV	Process value range (PV) for standardizing the proportional gain (GAIN)	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
Occupied	Occupied by batch control	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC (0-1 edge transition)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OptimEn	1 = Enable optimization of PID parameters by PID tuner	BOOL	0
OptimOcc	1 = Optimization running	BOOL	0
OS_Perm	I/O for operator control permissions (Page 474)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1

Parameter	Description	Type	Default
OS1Perm	I/O for operator control permissions (Page 474)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
PropSel	1 = Activate P action	BOOL	1
PV	Process value (controlled variable)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_Unit	Unit of measure for process value	INT	1001
PV_AH_DC	Delay time for incoming high PV alarms [s]	REAL	0.0
PV_AH_DG	Delay time for outgoing high PV alarms [s]	REAL	0.0
PV_AH_En	1 = Enable PV alarm limit (high)	BOOL	1
PV_AH_Lim	PV alarm limit (high)	REAL	95.0
PV_AH_MsgEn	1 = Enable message for PV alarm (high)	BOOL	1
PV_AL_En	1 = Enable PV alarm limit (low)	BOOL	1
PV_AL_Lim	PV alarm limit (low)	REAL	5.0
PV_AL_MsgEn	1 = Enable message for PV alarm (low)	BOOL	1
PV_AL_DC	Delay time for low incoming PV alarms [s]	REAL	0.0
PV_AL_DG	Delay time for low outgoing PV alarms [s]	REAL	0.0
PV_Hyst	Hysteresis for PV alarm, warning and tolerance limits	REAL	1.0
PV_OpScale	Limit for scale in PV bar graph of faceplate	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
PV_TH_DC	Delay time for incoming high PV tolerance messages [s]	REAL	0.0
PV_TH_DG	Delay time for outgoing high PV tolerance messages [s]	REAL	0.0
PV_TH_En	1 = Enable PV tolerance limit (high)	BOOL	0
PV_TH_Lim	PV tolerance message limit (high)	REAL	85.0
PV_TH_MsgEn	1 = Enable message for PV tolerance message (high)	BOOL	1
PV_TL_DC	Delay time for low incoming PV tolerance messages [s]	REAL	0.0
PV_TL_DG	Delay time for low outgoing PV tolerance messages [s]	REAL	0.0
PV_TL_En	1 = Enable PV tolerance limit (low)	BOOL	0
PV_TL_Lim	PV tolerance message limit (low)	REAL	15.0
PV_TL_MsgEn	1 = Enable message for PV tolerance message (low)	BOOL	1
PV_WH_DC	Delay time for incoming high PV warnings [s]	REAL	0.0
PV_WH_DG	Delay time for outgoing high PV warnings [s]	REAL	0.0
PV_WH_En	1 = Enable PV warning limit (high)	BOOL	1
PV_WH_Lim	PV warning limit (high)	REAL	90.0
PV_WH_MsgEn	1 = Enable message for PV warning (high)	BOOL	1
PV_WL_DC	Delay time for incoming low PV warnings [s]	REAL	0.0
PV_WL_DC	Delay time for outgoing low PV warnings [s]	REAL	0.0
PV_WL_En	1 = Enable PV warning limit (low)	BOOL	1

Parameter	Description	Type	Default
PV_WL_Lim	PV warning limit (low)	REAL	10.0
PV_WL_MsgEn	1 = Enable message for PV warning (low)	BOOL	1
Rbk	Position feedback for display on OS	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
RbkHyst	Alarm hysteresis for position feedback	REAL	1.0
RbkW_DC	Delay time for incoming Rbk warnings [s]	REAL	0.0
RbkW_DG	Delay time for outgoing Rbk warnings [s]	REAL	0.0
RbkWH_En	1 = Enable warning (high) for position feedback	BOOL	1
RbkWH_Lim	Limit for position feedback of warning (high)	REAL	100.0
RbkWH_MsgEn	1 = Enable messages for warning (high) for position feedback	BOOL	1
RbkWL_En	1 = Enable warning (low) for position feedback	BOOL	1
RbkWL_Lim	Limit for position feedback of warning (low)	REAL	0.0
RbkWL_MsgEn	1 = Enable messages for warning (low) for position feedback	BOOL	1
RefStdDevIn	Reference value of PV standard deviation (sigma) in defined "good" state of control loop	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#78
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
SafePos	1 = Safe position (Page 120) for controller's manipulated variable is ManHiLim, 0 = Safe position for controller's manipulated variable is ManLoLim	BOOL	0
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SelFp1	Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
SimOn	1 = Simulation on	BOOL	0
SimPV	Process value used for SimOn = 1	REAL	0.0
SimRbk	Position feedback used for SimOn = 1	REAL	0.0
SP_DnRaLim	Limit (low) for ramp of setpoint [SP_Unit/s]	REAL	100.0
SP_ExHiLim	Limit (high) for external setpoint	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
SP_ExLoLim	Limit (low) for external setpoint	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_Ext	external setpoint - (to interconnection)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_ExtOp	1 = Select external setpoint (via operator)	BOOL	0

Parameter	Description	Type	Default
SP_InHiLim	Limit (high) of internal setpoint	REAL	100.0
SP_InLoLim	Limit (low) of internal setpoint	REAL	0.0
SP_Int	Internal setpoint for operation	REAL	0.0
SP_IntOp	1 = Select internal setpoint (via operator)	BOOL	0
SP_Load	Defined setpoint if SP_LoadOn = 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_LoadOn	1 = Take controller into automatic mode with internal setpoint and set SP:= SP_Load setpoint. This kind of setpoint input has maximum priority.	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP_RateOn	1 = Activate limitation of setpoint gradients	BOOL	0
SP_RmpModTime	1 = Use time (SP_RmpTime) for setpoint ramp 0 = Use gradient	BOOL	0
SP_RmpOn	1 = Activate setpoint ramp to target setpoint SP_RmpTarget	BOOL	0
SP_RmpTarget	Target setpoint for setpoint ramp	REAL	0.0
SP_RmpTime	Time for setpoint ramp [s] from current SP up to SP_RmpTarget	REAL	0.0
SP_TrkExt	1 = Bumpless switchover from external to internal setpoint active	BOOL	1
SP_TrkPV	1 = Setpoint follows PV in manual mode and with tracking	BOOL	0
SP_UpRaLim	Gradient limit (high) for setpoint [SP_Unit/s]	REAL	100.0
StepNo	Batch step number	DWORD	16#00000000
TD	Derivative action time [s] TD.ST=16#FF: Operable in faceplate	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
TI	Integral action time [s] TI.ST=16#FF: Operable in faceplate	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#FF
UserStatus	Freely assignable bits for use in faceplate	BYTE	16#00

Output parameters

Parameter	Description	Type	Default
AdvCoAct	1 = Program mode is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AdvCoRdy	1 = Program mode available	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AutAct	1 = Automatic mode is active 0 = Manual mode is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
CascaCut	Cascade connection: 1 = Control chain from primary to secondary controller is interrupted	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ENO	1 = Block algorithm completed without errors	BOOL	0
ER	Control error	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
ER_AH_Act	1 = Alarm limit (high) for control error violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ER_AL_Act	1 = Alarm limit (low) for control error violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ErrorNum	Output of current error number. For error numbers that can be output by this block, see PIDConR error handling (Page 485)	INT	-1
FFwdHiAct	1 = Limit (high) for additive disturbance variable activation violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FFwdLoAct	1 = Limit (low) for additive disturbance variable activation violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
GainEff	Effective proportional gain, depends on Gain, NormPV, and NormMV	REAL	1.0
ManAct	1 = Manual mode active	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
ManHiOut	Limit (high) for manual mode, corresponds to the input parameter ManHiLim	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80

Parameter	Description	Type	Default
ManLoOut	Limit (low) for manual mode, corresponds to the input parameter ManLoLim	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MS_Release	Maintenance release: 1 = Release for OS operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MsgAckn1	Alarm acknowledgement status 1 (output STATUS of first ALARM_8P)	WORD	16#0000
MsgAckn2	Alarm acknowledgement status 2 (output STATUS of second ALARM_8P)	WORD	16#0000
MsgErr1	Alarm error 1 (output ERROR of first ALARM_8P)	BOOL	0
MsgErr2	Alarm error 2 (output ERROR of second ALARM_8P)	BOOL	0
MsgStat1	Alarm status 1 (output ERROR of first ALARM_8P)	WORD	16#0000
MsgStat2	Alarm status 2 (output ERROR of second ALARM_8P)	WORD	16#0000
MV	Manipulated variable	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV_HiAct	1 = Limit (high) of manipulated variable violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_LoAct	1 = Limit (low) of manipulated variable violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_UnitOut	Unit of measure for manipulated variable, for interconnecting to the MV_Unit input parameter of the ConPerMon block	INT	0
OosAct	1 = Block is "out of service"	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
OS1PermOut	Display of OS1Perm	DWORD	16#FFFFFFFF
OS1PermLog	Display of OS1Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
PV_AH_Act	1 = PV alarm (high) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_AL_Act	1 = PV alarm (low) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
PV_Out	Output for process value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_TH_Act	1 = PV tolerance message (high) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_TL_Act	1 = PV tolerance message (low) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_ToleHi	Limit (high) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_ToleLo	Limit (low) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_UnitOut	Unit of measure for process value, for interconnecting to the PV_Unit input parameter of theConPerMon block	INT	0
PV_WH_Act	1 = PV warning (high) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_WL_Act	1 = PV warning (low) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RbkOut	Output for position feedback	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
RbkWH_Act	1 = Warning (high) for position feedback active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RbkWL_Act	1 = Warning (low) for position feedback active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP	Setpoint used by controller	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_DnRaAct	1 = Negative gradient limiting of setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_ExHiAct	1 = Limit (high) for external setpoint has been reached	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80

Parameter	Description	Type	Default
SP_ExLoAct	1 = Limit (low) for external setpoint has been reached	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_ExtAct	1 = External setpoint is active 0 = Internal setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_ExtOut	External setpoint, corresponds to input parameter SP_Ext	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_InHiOut	Limit (high) for SP_Int corresponds to input parameter SP_InHiLim	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
SP_InLoOut	Limit (low) for SP_Int corresponds to input parameter SP_InLoLim	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_UpRaAct	Positive gradient limiting of setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word 1 (Page 465)	DWORD	16#00000000
Status2	Status word 2 (Page 465)	DWORD	16#00000000
SumMsgAct	1 = Active process alarm	STRUCT • Value: BOOL • ST: BYTE	00#80

See also

PIDConR block diagram (Page 500)

PIDConR messaging (Page 486)

PIDConR modes (Page 471)

4.7.7 PIDConR block diagram

PIDConRblock diagram

A block diagram is not provided for this block.

See also

PIDConR I/Os (Page 489)
PIDConR messaging (Page 486)
PIDConR error handling (Page 485)
PIDConR functions (Page 474)
PIDConR modes (Page 471)
Description of PIDConR (Page 465)

4.7.8 Operator control and monitoring

4.7.8.1 PIDConR views

Views of the PIDConR block

This block has the same views as the PIDConL block. Please refer to the section Operator control and monitoring (Page 458) of the PIDConL block.

4.8 PIDStepL - step controller

4.8.1 Description of PIDStepL

Object name (type + number) and family

Type + number: FB 1878

Family: Control

Area of application for PIDStepL

The block is used for the following applications:

- Fixed setpoint control
- Cascade control
- Ratio control
- Split-range control
- Smith predictor closed-loop control
- Override control (override)

How it works

The block is a PID step controller with binary output signals (manipulated variable signals). It is used to control integral action actuators (e.g. valves driven by motors).

The block functions following the PID algorithm with a delayed D action and an integrator with double precision.

The step controller can function both with and without a position feedback for the valve position.

The block is suitable for controlling sluggish control loops, for example, for temperatures and filling levels, and high-speed control loops, for example, for flow rates and speed. For a given CPU, a compromise has to be made between the number of controllers and the frequency with which the individual controllers have to be processed. The faster the modulated control loops are, i.e. the more frequently the manipulated variables have to be calculated per time unit, the lower the number of controllers that can be installed.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

For the control loop monitoring to work as planned in the trend view of the controller faceplates, the

```
S7_xarchive:='Value, shortterm;'
```

attributes in the process tag types for control loops at the controller function block must be set for the following tags:

- Input parameters:
  - `CPI_In`
- Output parameters
  - `MV`
  - `MV_HiAct`
  - `MV_LoAct`
  - `AutAct`
  - `SP`
  - `PV_Out`
  - `PV_ToleHi`
  - `PV_ToleLo`

For the PIDStepL block, the Advanced Process Library contains templates for process tag types as examples and there is a example project (APL_Example_xx, xx designates the language variant) containing different application cases for this block.

Several process tag types are simulated in the example project and serve to explain the block function.

Examples of process tag types:

- Step controller with direct access to the actuator and without position feedback (StepControlDirect) (Page 1437)
- Step controller with assigned actuator block and position feedback (StepControlActor) (Page 1437)

Startup characteristics

Use the Setting the startup response (Page 187) feature to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

Status word allocation for `Status1` parameter

For a description of the individual parameters, see the section PIDStepL I/Os (Page 519).

Status bit	Parameter
0	`Occupied`
1	`BatchEn`
2	`SimOn`
3	`OosAct.Value`
4	`OosLi.Value`
5	`AutAct.Value`
6	Not allocated
7	`ManAct.Value`
8	`SP_ExtAct.Value`
9	`MV_ForOn.Value`
10	`MV_TrkOn.Value`
11	`NOT RbkClosed.Value`
12	`Open.Value`
13	`Close.Value`
14	`Stop.Value`
15	`FbkOpened.Value`
16	`FbkClosed.Value`
17 - 18	Not allocated
19	`AdvCoAct`
20	1 = Input parameter `Rbk` is not interconnected (RbkOut.ST = 16#FF)
21	`NegGain`
22 - 23	Not allocated
24	`OptimOcc`
25- 30	Not allocated
31	`WithRbk = 1` (Step controller with position feedback)

Status word allocation for `Status2` parameter

Status bit	Parameter
0	MsgLock
1	PV_AH_Act.Value
2	PV_WH_Act.Value
3	PV_TH_Act.Value
4	PV_TL_Act.Value
5	PV_WL_Act.Value
6	PV_AL_Act.Value
7	PV_AH_En
8	PV_WH_En
9	PV_TH_En
10	PV_TL_En
11	PV_WL_En
12	PV_AL_En
13	PV_AH_MsgEn
14	PV_WH_MsgEn
15	PV_TH_MsgEn
16	PV_TL_MsgEn
17	PV_WL_MsgEn
18	PV_AL_MsgEn
19	ER_AH_Act.Value
20	ER_AL_Act.Value
21	ER_AH_En
22	ER_AL_En
23	ER_AH_MsgEn
24	ER_AL_MsgEn
25	RbkWH_Act.Value
26	RbkWL_Act.Value
27	RbkWH_En
28	RbkWL_En
29	RbkWH_MsgEn
30	RbkWL_MsgEn
31	MS_RelOp

See also

PIDStepL functions (Page 506)
PIDStepL messaging (Page 516)
PIDStepL modes (Page 505)
PIDStepL error handling (Page 514)
PIDStepL block diagram (Page 531)

4.8.2 PIDStepL modes

PIDStepL modes

The block can be operated using the following modes:

- Automatic mode (Page 29)
- Manual mode (Page 29)
- Program mode for closed-loop controllers (Page 34)
- Out of service (Page 27)

The next section provides additional block-specific information relating to the general descriptions.

Automatic mode

You can find general information on "Automatic mode", switching modes and bumpless switchover in the section Manual and Automatic mode for control blocks (Page 29).

Manual mode

You can find general information on "Manual mode", switching modes and bumpless switchover in the section Manual and Automatic mode for control blocks (Page 29).

Program mode for controllers

You can find general information about the "Program mode for controller" in the section Program mode for closed-loop controllers (Page 34).

Out of service

You can find general information about the "Out of service" mode in the section Out of service (Page 27).

See also

Description of PIDStepL (Page 501)

PIDStepL functions (Page 506)

PIDStepL error handling (Page 514)

PIDStepL messaging (Page 516)

PIDStepL I/Os (Page 519)

PIDStepL block diagram (Page 531)

4.8.3 PIDStepL functions

Functions of PIDStepL

The functions for this block are listed below.

Generation of manipulated variables with position feedback (`WithRbk = 1`)

The manipulated variable `MV` and therefore the manipulated signals Open, Close and Stop can be generated as follows:

MV_ ForOn	Man Act	MV_ TrkOn	AdvCoAct AND NOT AdvCoModSP	MV	Limit monitoring	State	Open, Close, Stop
1	-	-	-	`MV_Forced`	none	Forced tracking through constraint without limitation	Depending on `Rbk` and `MV`, the output signals `Open`, `Close` and `Stop` are generated using the algorithm of a positioner.
0	1	-	-	`Man`	`ManHiLim` `ManLoLim`	Manual mode, set by the operator	
0	0	1	-	`MV_Trk`	`MV_HiLim` `MV_LoLim`	Tracking with limitation	
0	0	0	1	`AdvCoMV`	`MV_HiLim` `MV_LoLim`	Higher-level program mode	
0	0	0	0	`P_Part + I_Part + D_Part+ FFwd`	`MV_HiLim` `MV_LoLim`	Automatic mode (PID algorithm)	

If the controller is in "out of service" mode, the output parameter `MV` is set to the last valid value in manual mode or the safety manipulated variable depending on the `Feature` bit (Safety value of the manipulated variable effective at startup (Page 196)). Refer to the Out of service (Page 27) section for more on this.

Generation of manipulated signal without position feedback (`WithRbk = 0`)

The manipulated variable MV can be generated as follows:

`ManAct`	`Open, Close, Stop`	State
1	The output signals are generated using input signals `OpenOp/Li`, `CloseOp/Li` or `StopOp/Li`	Manual mode, set by the operator
0	The output signals are generated using PID output parameters `P_Part`, `I_Part`, `D_Part` and `FFwd`	Automatic mode (PID algorithm)

If the controller is in "out of service" mode, the output parameter MV is set to the last valid value in manual mode or the safety manipulated variable depending on the `Feature` bit (Safety value of the manipulated variable effective at startup (Page 196)). Refer to the Out of service (Page 27) section for more on this.

Tracking and limiting a manipulated variable

The block provides the standard function Tracking and limiting a manipulated value (Page 111).

Note

This function is only available to you if position feedback is activated for the controller (`WithRbk = 1`).

Safe position

The block provides the standard function Safe position for motors, valves and controllers (Page 120).

"Actuator active" information

If the parameter is `FbkClosed = 0`, this is known as "Actuator active". This status can be used to display a custom icon in the process image, for example, and is saved in the status word (see Status word section in Description of PIDStepL (Page 501)).

Limit monitoring of position feedback

The block provides the standard function Limit monitoring of the feedback (Page 76).

Note

This function is only available to you if position feedback is activated for the controller (`WithRbk = 1`).

External/internal setpoint specification

The block provides the standard function Setpoint input - internal and external (Page 96).

Setpoint limiting for external setpoints

The block provides the standard function Setpoint limiting for external setpoints (Page 110).

Gradient limit of the setpoint

The block provides the standard function Ramp limiting of the setpoint (Page 108).

Using setpoint ramp

The block provides the standard function Using a setpoint ramp (Page 98).

Tracking setpoint in manual mode

The block provides the standard function Setpoint tracking the process variable in manual mode (Page 110).

Simulating signals

The block provides the standard function Simulating signals (Page 93).

You can simulate the following values:

- Process value (`SimPV`)
- Position feedback (`SimRbk`)

Limit monitoring of the process value

The block provides the standard function Limit monitoring of the process value (Page 70).

Control error generation and dead band

The block provides the standard function Control error generation and deadband (Page 109).

Limit monitoring of control error

The block provides the standard function Monitoring the error signal limits (Page 77).

Inverting control direction

The block provides the standard function Inverting the control sense (Page 108).

Physical standardization of setpoint, manipulated variable and process value

Controller gain `Gain` is entered either using a physical variable or as standardized value.

Gain as physical variable:

The standardized variables retain their default values:

- `NormPV.High = 100` and `NormPV.Low = 0`

The effective gain is:

```
GainEff = Gain
```

Entering a standardized gain (dimensionless):

Change the standardized variables to the actual range of the process values and manipulated variables.

- Internal and external setpoints; the process value and corresponding parameters are entered according to the physical measuring range of the process value.
- The manual value, the tracking value of the manipulated variable, feedforward control and the corresponding parameters are entered as a percentage between 0 ... 100.

The effective gain is:

```
GainEff = 100,0 / (NormPV.High - NormPV.Low) · Gain
```

Selecting a unit of measure

The block provides the standard function Selecting a unit of measure (Page 53).

PID algorithm

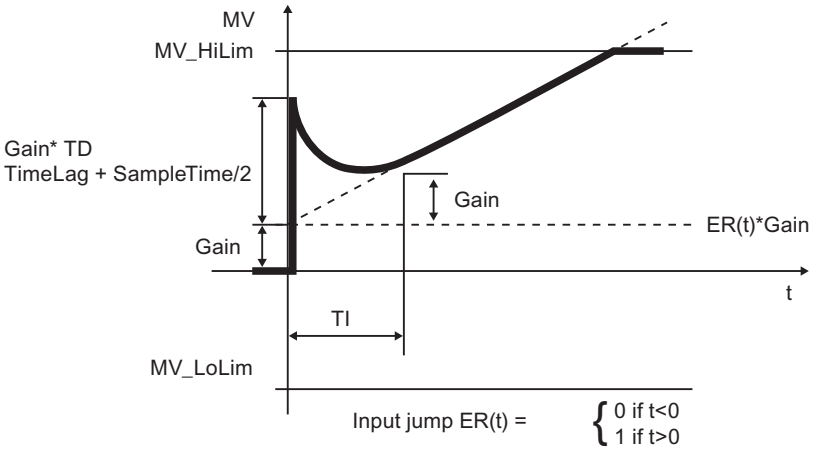
The manipulated variable is generated in automatic mode according to the following algorithm:

```
MV = Gain · (1 + 1/(TI · s) + TD/( 1 + TD/DiffGain · s) · ER
```

Where:

s = Complex number

The following step response occurs:



Note

This formula describes a standard application where P, I and D action is activated and the P and D actions are not in the feedback circuit (`PropSel = true, TI <> 0 DiffToFbk = false` and `PropFacSP = 1`).

The D action delay is derived from `TD/DiffGain`.

- The P action is displayed after `P_Part` and can be deactivated using `PropSel = 0`.
- The I action is displayed after `I_Part` and can be deactivated using `TI = 0`.
- The D action is displayed after `D_Part` and can be deactivated using `TD = 0`.

Structure segmentation at controllers

The block provides the standard function Structure segmentation at controllers (Page 113).

Anti-windup

The controller has an anti-windup function. The I action is frozen after the manipulated variable has reached limits (`MV_HiLim` or `MV_LoLim`).

Activating and limiting disturbance variables

The block provides the standard function Activating and limiting disturbance variable (Page 113).

Forming the signal status for blocks

The block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `ER.ST`
- `FFwd.ST`
- `Rbk.ST`

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)
1	Reaction to the out of service mode (Page 199)
2	Resetting the commands for changing the mode (Page 193)
4	Setting switch or button mode (Page 195)
15	Safety manipulated variable with "out of service" operating mode in effect (Page 196)
16	Safety value of the manipulated variable effective at startup (Page 196)
18	Disabling bumpless changeover to automatic mode for controllers (Page 197)
22	Update acknowledgment and error status of the message call (Page 193)

Operator control permissions via parameters OS_Perm and OS1Perm

The block has the following Operator permissions (Page 45) for the OS_Perm parameter:

Bit	Function
0	1 = Operator can switch into automatic mode AutModOp
1	1 = Operator can switch into manual mode ManModOp
2	1 = Operator can switch into "out of service" mode OosOp
3	1 = Operator can switch into program mode AdvCoEn
4	1 = Operator can switch the setpoint to external SP_ExtOp
5	1 = Operator can switch the setpoint to internal SP_IntOp
6	1 = Operator can change the internal setpoint SP_Int
7	Step controller with position feedback WithRbk =1: 1 = Operator can change the manual parameter Man Step controller without position feedback WithRbk =0: 1 = Operator can change the manual operation signals OpenOp, StopOp, CloseOp
8	1 = Operator can change maximum usage limit of the setpoint SP_InHiLim
9	1 = Operator can change minimum usage limit of the setpoint SP_InLoLim
10	1 = Operator can change maximum usage limit of the manipulated variable ManHiLim
11	1 = Operator can change minimum usage limit of the manipulated variable ManLiLim
12	1 = Operator can use the setpoint's gradient limitation function SP_RateOn
13	1 = Operator can change the setpoint's high limit for the ramp SP_UpRaLim
14	1 = Operator can change the setpoint's low limit for the ramp SP_DnRaLim
15	1 = Operator can change between the time value or the value for the ramp SP_RmpModTime
16	1 = Operator can change the ramp time SP_RmpTime
17	1 = Operator can change the target setpoint SP_RmpTarget for the setpoint ramp
18	1 = Operator can activate the setpoint ramp function SP_RmpOn
19	1 = Operator can permit the PID optimization function OptimEn
20	1 = Operator can activate the track setpoint in manual mode function SP_TrkPV
21	1 = Operator can activate the bumpless changeover from external to internal function SP_TrkExt
22	1 = Operator can change the gain parameter Gain
23	1 = Operator can change the integral action time parameter TI
24	1 = Operator can change the derivative action time parameter TD
25	1 = Operator can change the derivative gain parameter DiffGain
26	1 = Operator can change the dead band parameter DeadBand
27	Not allocated
28	1 = Operator can change the integral action time parameter MotorTime
29	1 = Operator can change the integral action time parameter PulseTime
30	1 = Operator can change the integral action time parameter BreakTime
31	Not allocated

The block has the following operator control permissions for the OS1Perm parameter:

Bit	Function
0	1 = Operator can change the limit (process value) for the high alarm PV_AH_Lim
1	1 = Operator can change the limit (process value) for the high warning PV_WH_Lim
2	1 = Operator can change the limit (process value) for the high tolerance PV_TH_Lim
3	1 = Operator can change the hysteresis (process value) PV_Hyst
4	1 = Operator can change the limit (process value) for the low tolerance PV_TL_Lim
5	1 = Operator can change the limit (process value) for the low warning PV_WL_Lim
6	1 = Operator can change the limit (process value) for the low alarm PV_AL_Lim
7	1 = Operator can change the limit (control error) for the high alarm ER_AH_Lim
8	1 = Operator can change the hysteresis (control error) ER_Hyst
9	1 = Operator can change the limit (control error) for the low alarm ER_AL_Lim
10	1 = Operator can change the limit (position feedback) for the high warning RbkWH_Lim
11	1 = Operator can change the hysteresis (position feedback) RbkHyst
12	1 = Operator can change the limit (position feedback) for the low warning RbkWL_Lim
13 - 15	Not allocated
16	1 = Operator can activate the simulation function SimOn
17	1 = Operator can activate the maintenance release function MS_RelOp
18 - 31	Not allocated

Maintenance release

The block provides the standard function Maintenance release (Page 47).

Generating instance-specific messages

The block provides the standard function Generating instance-specific messages (Page 46) without the time stamp function in the I/O devices.

Specifying the display area for process and setpoint values as well as operations

This block provides the standard function Display and operator input area for process values and setpoints (Page 42).

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 122).

See also

PIDStepL messaging (Page 516)
PIDStepL I/Os (Page 519)
PIDStepL error handling (Page 514)
PIDStepL modes (Page 505)
PIDStepL block diagram (Page 531)

4.8.4 PIDStepL error handling

PIDStepL troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` output parameter can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Default value when implementing the block; block will not be processed.
0	There is no error.
30	The value of `PV` can no longer be displayed in the REAL number field.
31	The value of `SP_Ext` can no longer be displayed in the REAL number field.
32	The value of `FFwd` can no longer be displayed in the REAL number field.
33	The value of `MV_Trk` can no longer be displayed in the REAL number field.
34	The value of `MV_Forced` can no longer be displayed in the REAL number field.
35	The value of `Rbk` can no longer be displayed in the REAL number field.
36	The value of `MV` can no longer be displayed in the REAL number field.
50	The controller cannot be switched in program mode, because program mode with setpoint specification (AdvCoModSP=false) is not possible with step controllers without position feedback (WithRbk=false).
60	`|Tl| < SampleTime/2`
61	`|TD| < SampleTime`
62	`DiffGain < 1` or `DiffGain > 10`
63	`TD/DiffGain < SampleTime/2`
64	`PropFacSP < 0` or `PropFacSP > 1`
66	`NormPV_High = NormPV_Low`
67	`MotorTime < SampleTime`
68	`PulseTime < SampleTime`
69	`BreakTime < SampleTime`

See also

PIDStepL functions (Page 506)
Description of PIDStepL (Page 501)
PIDStepL modes (Page 505)
PIDStepL messaging (Page 516)
PIDStepL I/Os (Page 519)
PIDStepL block diagram (Page 531)
Setting switch or button mode (Page 195)

4.8.5 PIDStepL messaging

Messaging

The following messages can be generated for this block:

- Control system fault
- Process messages
- Instance-specific messages

Control system fault

The following control system error messages can be output:

Message instance	Message identifier	Message class	Event
MsgEvId2	SIG 6	AS process control message - fault	$$BlockComment$$ External error has occurred

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter CSF. If this signal changes to CSF = 1, a control system fault is triggered (MsgEvId2, SIG 6).

Process messages

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 1	Alarm - high	$$BlockComment$$ PV - high alarm limit violated
	SIG 2	Warning - high	$$BlockComment$$ PV - high warning limit violated
	SIG 3	Tolerance - high	$$BlockComment$$ PV - high tolerance limit violated
	SIG 4	Tolerance - low	$$BlockComment$$ PV - low tolerance limit violated
	SIG 5	Warning - low	$$BlockComment$$ PV - low warning limit violated
	SIG 6	Alarm - low	$$BlockComment$$ PV - low alarm limit violated
	SIG 7	Alarm - high	$$BlockComment$$ ER - high alarm limit violated
	SIG 8	Alarm - low	$$BlockComment$$ ER - low alarm limit violated

Message instance	Message identifier	Message class	Event
MsgEvId2	SIG 7	Warning - high	$$BlockComment$$ Rbk - high warning limit violated
	SIG 8	Warning - low	$$BlockComment$$ Rbk - low warning limit violated

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Instance-specific messages

You can use up to four instance-specific messages with this block.

Message instance	Message identifier	Message class	Event
MsgEvId2	SIG 1	AS process control message - fault	$$BlockComment$$ External message 1 Status 16#@5%x@
	SIG 2	AS process control message - fault	$$BlockComment$$ External message 2 Status 16#@6%x@
	SIG 3	AS process control message - fault	$$BlockComment$$ External message 3 Status 16#@7%x@
	SIG 4	AS process control message - fault	$$BlockComment$$ External message 4 Status 16#@8%x@

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Associated values for message instance `MsgEvId1`

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`
4	Process value `PV_Out`
5	Control error `ER`
6	`ExtVa106`
7	`ExtVa107`
8	Not allocated
9	Not allocated
10	Not allocated

The associated values 6 and 7 are assigned to the parameters `ExtVa106` ... `ExtVa107` and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

Associated values for message instance `MsgEvId2`

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`
4	Readback value Rbk
5	Signal status `ExtMsg1`
6	Signal status `ExtMsg2`
7	Signal status `ExtMsg3`
8	Signal status `ExtMsg4`
9	`ExtVa209`
10	`ExtVa210`

The associated values 9 and 10 are assigned to the parameters `ExtVa209` ... `ExtVa210` and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

See also

Description of PIDStepL (Page 501)

PIDStepL functions (Page 506)

PIDStepL I/Os (Page 519)

PIDStepL modes (Page 505)

PIDStepL error handling (Page 514)

PIDStepL block diagram (Page 531)

4.8.6 PIDStepL I/Os

PIDStepL I/Os

Input parameters

Parameter	Description	Type	Default
AdvCoEn	1 = Enable program mode via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AdvCoOn	1 = Enable program mode via faceplate	BOOL	0
AdvCoModSP	Type of program mode: 1 = Setpoint specification 0 = Manipulated variable specification	BOOL	1
AdvCoMstrOn	Activate (0-1) or deactivate (1-0) program mode via edge transition	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AdvCoMV	Specified value from the external program	REAL	0.0
AutModLi	1 = Automatic mode via interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AutModOp	1 = Automatic mode via operator (controlled by ModLiOp = 0)	BOOL	0
BatchEn	1 = Enable allocation for batch control	BOOL	0
BatchID	Batch ID	DWORD	16#00000000
BatchName	Batch name	STRING[32]	''
BreakTime	Minimum break time [s]	REAL	1.0
CloseLi	1 = Close via interconnection or CFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
CloseOp	1 = Close via operator	BOOL	0
CPI_In	Input for control performance index, which is calculated by the assigned ConPerMon block	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#78
CSF	1 = External error (control system error)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
DeadBand	Width of dead band	REAL	0.0
DiffGain	Gain of differentiator [1...10] DiffGain = TD / (differentiation of D action)	STRUCT • Value: REAL • ST: BYTE	- • 5.0 • 16#80

Parameter	Description	Type	Default
DiffToFbk	1 = D action is placed in the feedback	BOOL	0
EN	1 = Called block will be processed	BOOL	1
ER_A_DC	Delay for incoming alarms during control error monitoring	REAL	0.0
ER_A_DG	Delay for outgoing alarms during control error monitoring	REAL	0.0
ER_AH_En	1 = Activate alarm (high) for control error monitoring	BOOL	1
ER_AH_Lim	Alarm limit (high) for control error monitoring	REAL	100.0
ER_AH_MsgEn	1 = Activate messages for alarm (high) for control error monitoring	BOOL	1
ER_AL_En	1 = Activate alarm (low) for control error monitoring	BOOL	1
ER_AL_Lim	Alarm limit (low) for control error monitoring	REAL	-100.0
ER_AL_MsgEn	1 = Activate messages for alarm (low) for control error monitoring	BOOL	1
ER_Hyst	Alarm hysteresis for control error	REAL	1.0
ExtMsg1	1 = Binary input for freely selectable message 1 is used	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg2	1 = Binary input for freely selectable message 2 is used	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg3	1 = Binary input for freely selectable message 3 is used	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg4	1 = Binary input for freely selectable message 4 is used	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtVa106	Associated value 6 for messages (MsgEvID1)	ANY	
ExtVa107	Associated value 7 for messages (MsgEvID1)	ANY	
ExtVa209	Associated value 9 for messages (MsgEvID2)	ANY	
ExtVa210	Associated value 10 for messages (MsgEvID2)	ANY	
FbkClosed	Low limit stop signal of position feedback	STRUCT • Value: BOOL • ST: BYTE	- 0 16#80
FbkOpened	High limit stop signal of position feedback	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
Feature	I/O for additional functions (Page 506)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
FFwd	Input for additive disturbance variable activation	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
FFwdHiLim	Limit (high) for additive disturbance variable activation	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
FFwdLoLim	Limit (low) for additive disturbance variable activation	STRUCT • Value: REAL • ST: BYTE	- • -100.0 • 16#80
Gain	Proportional gain Gain.ST=16#FF: Operable in faceplate	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#FF
IntHoldNeg	1 = Integrator cannot run in negative direction	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
IntHoldPos	1 = Integrator cannot run in positive direction	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Man	Manual specification for the manipulated variable	REAL	0.0
ManHiLim	Limit (high) for manual parameter Man	REAL	100.0
ManLoLim	Limit (low) for manual parameter Man	REAL	0.0
ManModLi	1 = Manual mode via interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ManModOp	1 = Manual mode by OS operator (controlled by ModLiOp = 0)	BOOL	1
ModLiOp	Operating mode switchover between: • 0 = Operator • 1 = Interconnection or SFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MotorTime	Motor manipulating time [s]	REAL	30.0
MS_RelOp	1 = Maintenance release by OS operator	BOOL	0
MsgEvId1	Message number (assigned automatically)	DWORD	16#FFFFFFFF
MsgEvId2	Message number (assigned automatically)	DWORD	16#FFFFFFFF
MsgLock	1 = Suppress process messages	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
MV_Forced	Forced manipulated variable that is not limited and assumes top priority	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV_ForOn	1 = Forced manipulated variable MV_Forced output unlimited at output MV	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_HiLim	Limit (high) for manipulated variable MV	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
MV_LoLim	Limit (low) for manipulated variable MV	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV_Offset	Manipulated variable for ER = 0, operating point for controller with deactivated I action	REAL	0.0
MV_Trk	Tracking value for the manipulated variable MV	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV_TrkOn	1 = Tracking of manipulated variable MV	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
NegGain	0 = Positive controller gain: ER = Gain · (SP - PV) 1 = Negative controller gain: ER = Gain · (PV - SP)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
NormPV	Process value range (PV) for standardizing the proportional gain (GAIN)	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
Occupied	1 = Occupied by batch control	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC (0-1 edge transition)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OpenLi	1 = Open via interconnection or CFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OpenOp	1 = Open via operator	BOOL	0
OptimEn	1 = Enable optimization of PID parameters by PID tuner	BOOL	0
OptimOcc	1 = Optimization running	BOOL	0
OS_Perm	I/O for operator control permissions (Page 506)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1

Parameter	Description	Type	Default
OS1Perm	I/O for operator control permissions (Page 506)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
PropFacSP	Applying the P action to the feedback [0..1]. 0 = P action fully in feedback	REAL	1.0
PropSel	1 = Activate P action	BOOL	1
PulseTime	Minimum pulse duration [s]	REAL	1.0
PV	Process value (controlled variable)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_Unit	Unit of measure for process value	INT	1001
PV_A_DC	Delay time for incoming PV alarms [s]	REAL	0.0
PV_A_DG	Delay time for outgoing PV alarms [s]	REAL	0.0
PV_AH_En	1 = Enable PV alarm limit (high)	BOOL	1
PV_AH_Lim	PV alarm limit (high)	REAL	95.0
PV_AH_MsgEn	1 = Enable message for PV alarm (high)	BOOL	1
PV_AL_En	1 = Enable PV alarm limit (low)	BOOL	1
PV_AL_Lim	PV alarm limit (low)	REAL	5.0
PV_AL_MsgEn	1 = Enable message for PV alarm (low)	BOOL	1
PV_Hyst	Hysteresis for PV alarm, warning and tolerance limits	REAL	1.0
PV_OpScale	Limit for scale in PV bar graph of faceplate	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
PV_T_DC	Delay time for incoming PV tolerance messages [s]	REAL	0.0
PV_T_DG	Delay time for outgoing PV tolerance messages [s]	REAL	0.0
PV_TH_En	1 = Enable PV tolerance limit (high)	BOOL	0
PV_TH_Lim	PV tolerance message limit (high)	REAL	85.0
PV_TH_MsgEn	1 = Enable message for PV tolerance message (high)	BOOL	1
PV_TL_En	1 = Enable PV tolerance limit (low)	BOOL	0
PV_TL_Lim	PV tolerance message limit (low)	REAL	15.0
PV_TL_MsgEn	1 = Enable message for PV tolerance message (low)	BOOL	1
PV_Unit	Unit of measure for process value	INT	1001; °C
PV_W_DC	Delay time for incoming PV warnings [s]	REAL	0.0
PV_W_DG	Delay time for outgoing PV warnings [s]	REAL	0.0
PV_WH_En	1 = Enable PV warning limit (high)	BOOL	1
PV_WH_Lim	PV warning limit (high)	REAL	90.0
PV_WH_MsgEn	1 = Enable message for PV warning (high)	BOOL	1

Parameter	Description	Type	Default
PV_WL_En	1 = Enable PV warning limit (low)	BOOL	1
PV_WL_Lim	PV warning limit (low)	REAL	10.0
PV_WL_MsgEn	1 = Enable message for PV warning (low)	BOOL	1
Rbk	Position feedback for display on OS	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
RbkHyst	Alarm hysteresis for position feedback	REAL	1.0
RbkWH_En	1 = Enable warning (high) for position feedback	BOOL	1
RbkWH_Lim	Limit for position feedback of warning (high)	REAL	100.0
RbkWH_MsgEn	1 = Enable messages for warning (high) for position feedback	BOOL	1
RbkWL_En	1 = Enable warning (low) for position feedback	BOOL	1
RbkWL_Lim	Limit for position feedback of warning (low)	REAL	0.0
RbkWL_MsgEn	1 = Enable messages for warning (low) for position feedback	BOOL	1
RefStdDevIn	Reference value of PV standard deviation (sigma) in defined "good" state of control loop	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#78
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
S_RbkOnPIDTun	Simulation of position feedback on; For PCS 7 PID tuner only	BOOL	0
S_RbkPIDTun	Simulated position feedback	REAL	50.0
SafePos	Safe position (Page 120) for step controller actuating signals: 0 = Close, 1 = Open, 2 = Stop	INT	0
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SelFp1	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
SimOn	1 = Simulation on	BOOL	0
SimPV	Process value used for SimOn = 1	REAL	0.0
SimRbk	Position feedback used for SimOn = 1	REAL	0.0
SP_DnRaLim	Limit (low) for ramp of setpoint [SP_Unit/s]	REAL	100.0
SP_ExHiLim	Limit (high) for external setpoint	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80

Parameter	Description	Type	Default
SP_ExLoLim	Limit (low) for external setpoint	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_Ext	External setpoint of - (to interconnection)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_ExtLi	1 = Select external setpoint (via interconnection)	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_ExtOp	1 = Select external setpoint (via operator)	BOOL	0
SP_InHiLim	Limit (high) of internal setpoint	REAL	100.0
SP_InLoLim	Limit (low) of internal setpoint	REAL	0.0
SP_Int	Internal setpoint for operation	REAL	0.0
SP_IntLi	1 = Select internal setpoint (via interconnection)	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_IntOp	1 = Select internal setpoint (via operator)	BOOL	0
SP_LiOp	Select setpoint source (internal/external): 1 = Via interconnection 0 = Via operator	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_RateOn	1 = Activate limitation of setpoint gradients	BOOL	0
SP_RmpModTime	1 = Use time (SP_RmpTime) for setpoint ramp, 0 = Use gradient	BOOL	0
SP_RmpOn	1 = Activate setpoint ramp to target setpoint SP_RmpTarget	BOOL	0
SP_RmpTarget	Target setpoint for setpoint ramp	REAL	0.0
SP_RmpTime	Time for setpoint ramp [s] from current SP up to SP_RmpTarget	REAL	0.0
SP_TrkExt	1 = Bumpless switchover from external to internal setpoint active	BOOL	1
SP_TrkPV	1 = Setpoint follows PV in manual mode and with tracking	BOOL	0
SP_UpRaLim	Gradient limit (high) for setpoint [SP_Unit/s]	REAL	100.0
StepNo	Batch step number	DWORD	16#00000000
StopLi	1 = Stop via interconnection or CFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StopOp	1 = Stop via operator	BOOL	0
TD	Derivative action time in [s] TD.ST=16#FF: Operable in faceplate	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF

Parameter	Description	Type	Default
ThrAdaOn	Adaptation of threshold 0 = Hold constant	BOOL	1
TI	Integral action time [s] TI.ST=16#FF: Operable in faceplate	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#FF
UserStatus	Freely assignable bits for use in faceplate	BYTE	16#00
WithRbk	1 = Feedback value available for the manipulated variable	BOOL	0

Output parameters

Parameter	Description	Type	Default
AdvCoAct	1 = Program mode is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AdvCoRdy	1 = Program mode available	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AutAct	1 = Automatic mode is active 0 = Manual mode is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
CascaCut	Cascade connection: 1 = Control chain from primary to secondary controller is interrupted	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Close	Control output: 1 = Closed is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
D_Part	D action of PID algorithm	REAL	0.0
ENO	1 = Block algorithm completed without errors	BOOL	0
ER	Control error	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
ER_AH_Act	1 = Alarm limit (high) for control error violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ER_AL_Act	1 = Alarm limit (low) for control error violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ErrorNum	Output of current error number. For error numbers that can be output by this block, see PIDStepL error handling (Page 514)	INT	-1

Parameter	Description	Type	Default
FFwdHiAct	1 = Limit (high) for additive disturbance variable activation violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FFwdLoAct	1 = Limit (low) for additive disturbance variable activation violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
GainEff	Effective proportional gain, depends on Gain and NormPV	REAL	1.0
I_Part	I action of PID algorithm	REAL	0.0
ManAct	1 = Manual mode active	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
ManHiOut	Limit (high) for manual mode, corresponds to the input parameter ManHiLim	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
ManLoOut	Limit (low) for manual mode, corresponds to the input parameter ManLoLim	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MS_Release	Maintenance release: 1 = Release for OS operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MsgAckn1	Alarm acknowledgement status 1 (output STATUS of first ALARM_8P)	WORD	16#0000
MsgAckn2	Alarm acknowledgement status 2 (output STATUS of second ALARM_8P)	WORD	16#0000
MsgErr1	1 = Alarm error 1 (output ERROR of first ALARM_8P)	BOOL	0
MsgErr2	1 = Alarm error 2 (output ERROR of second ALARM_8P)	BOOL	0
MsgStat1	Alarm status 1 (output ERROR of first ALARM_8P)	WORD	16#0000
MsgStat2	Alarm status 2 (output ERROR of second ALARM_8P)	WORD	16#0000
MV	Manipulated variable	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
MV_HiAct	1 = Limit (high) of manipulated variable violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MV_LoAct	1 = Limit (low) of manipulated variable violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
OosAct	1 = Block is "out of service"	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Open	Control output: 1 = Open is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
OS1PermOut	Display of OS1Perm	DWORD	16#FFFFFFFF
OS1PermLog	Display of OS1Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
P_Part	P action of PID algorithm	REAL	0.0
PV_AH_Act	1 = PV alarm (high) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_AL_Act	1 = PV alarm (low) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_Out	Output for process value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_TH_Act	1 = PV tolerance message (high) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_TL_Act	1 = PV tolerance message (low) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_ToleHi	Limit (high) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_ToleLo	Limit (low) of 3-sigma band around the setpoint is calculated when a ConPerMon block is connected	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_UnitOut	Unit of measure for process value, for interconnecting to the PV_Unit input parameter of the ConPerMon block	INT	0
PV_WH_Act	1 = PV warning (high) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
PV_WL_Act	1 = PV warning (low) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RbkOut	Output for position feedback	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
RbkWH_Act	1 = Warning (high) for position feedback active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RbkWL_Act	1 = Warning (low) for position feedback active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP	Setpoint used by controller	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_DnRaAct	1 = Negative gradient limiting of setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP_ExHiAct	1 = Limit (high) for external setpoint has been reached	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP_ExLoAct	1 = Limit (low) for external setpoint has been reached	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP_ExtAct	1 = External setpoint is active 0 = Internal setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP_ExtOut	External setpoint, corresponds to input parameter SP_Ext	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_InHiOut	Limit (high) for SP_Int corresponds to input parameter SP_InHiLim	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
SP_InLoOut	Limit (low) for SP_Int corresponds to input parameter SP_InLoLim	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_UpRaAct	Positive gradient limiting of setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word 1 (Page 501)	DWORD	16#00000000

Parameter	Description	Type	Default
Status2	Status word 2 (Page 501)	DWORD	16#00000000
Stop	Control output: 1 =Stopped is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SumMsgAct	1 = Active process alarm	STRUCT • Value: BOOL • ST: BYTE	00#80
Threson	Adaptive threshold [%]	REAL	0.0

See also

PIDStepL messaging (Page 516)

PIDStepL modes (Page 505)

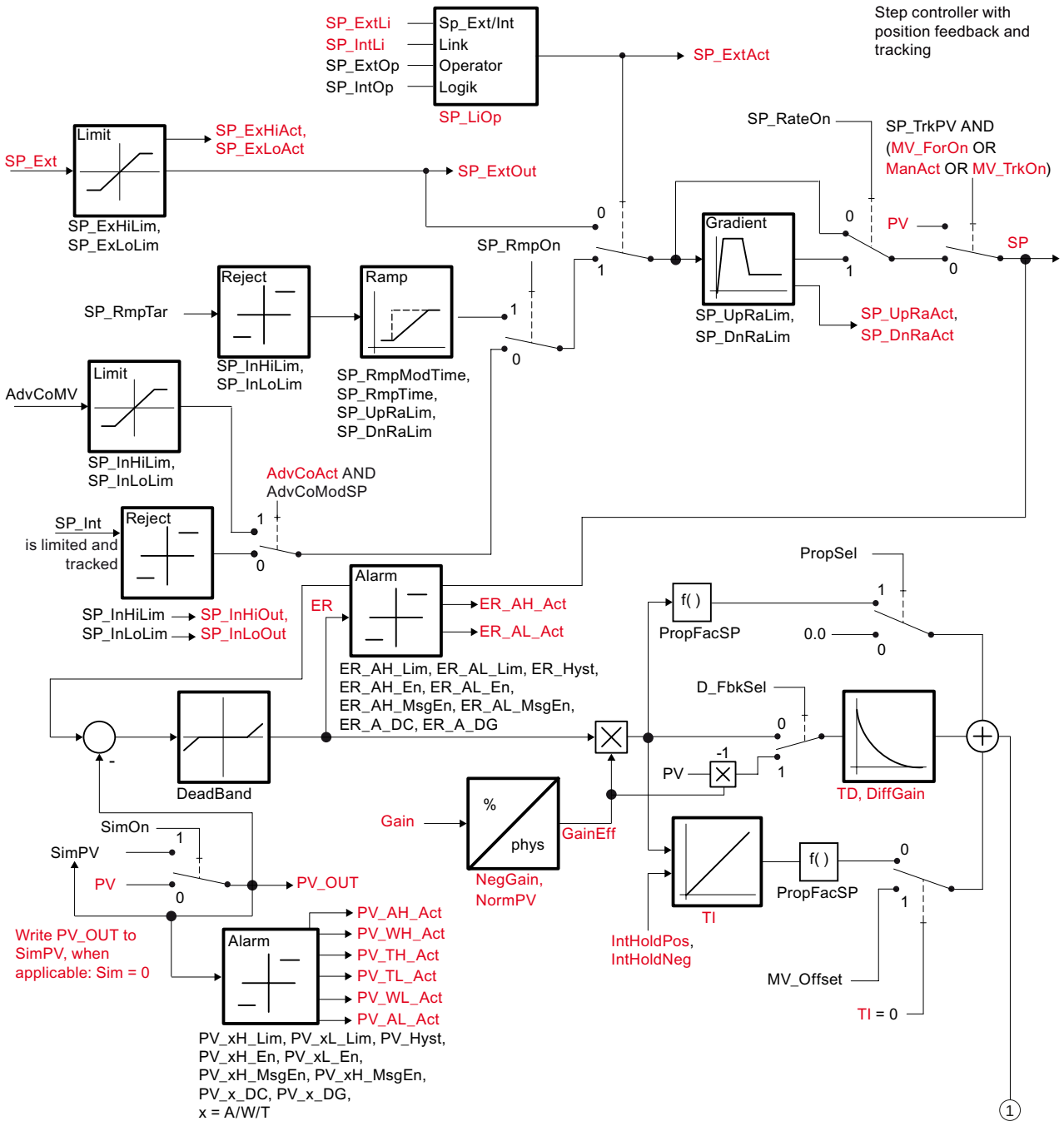
PIDStepL block diagram (Page 531)

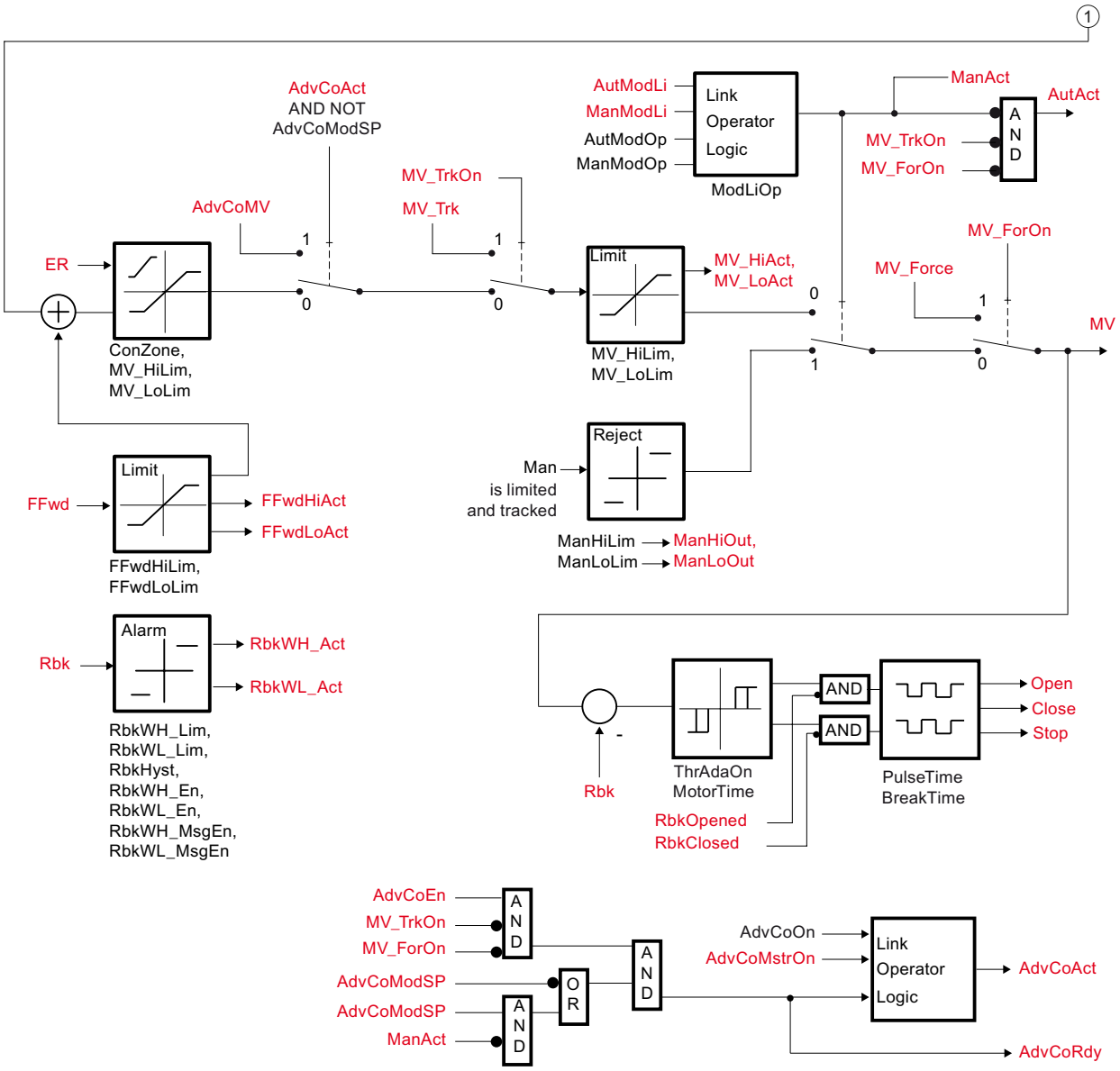
4.8.7 PIDStepL block diagram

PIDStepL block diagram

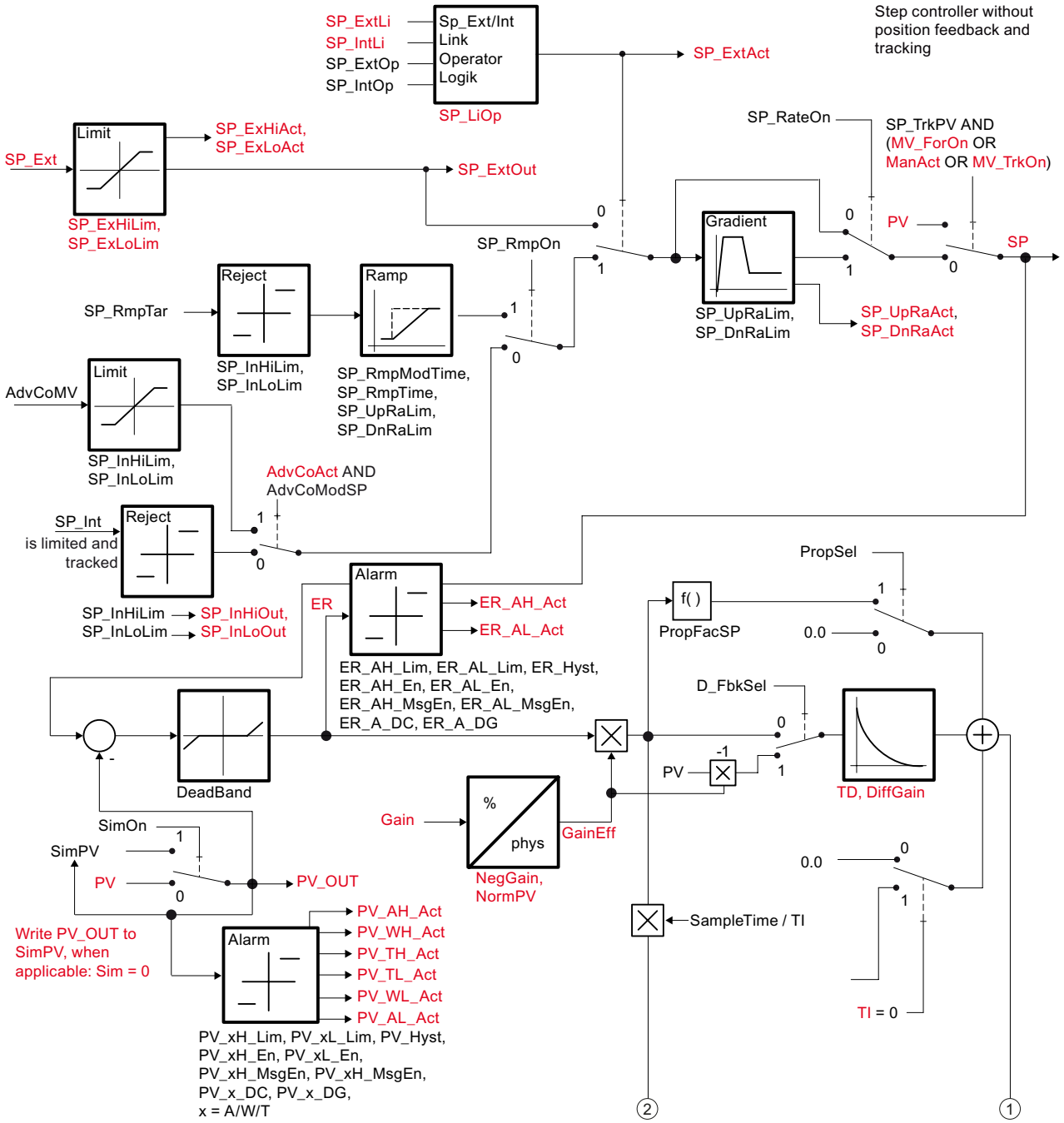
There are two block diagrams for this block: the step controller with and without position feedback.

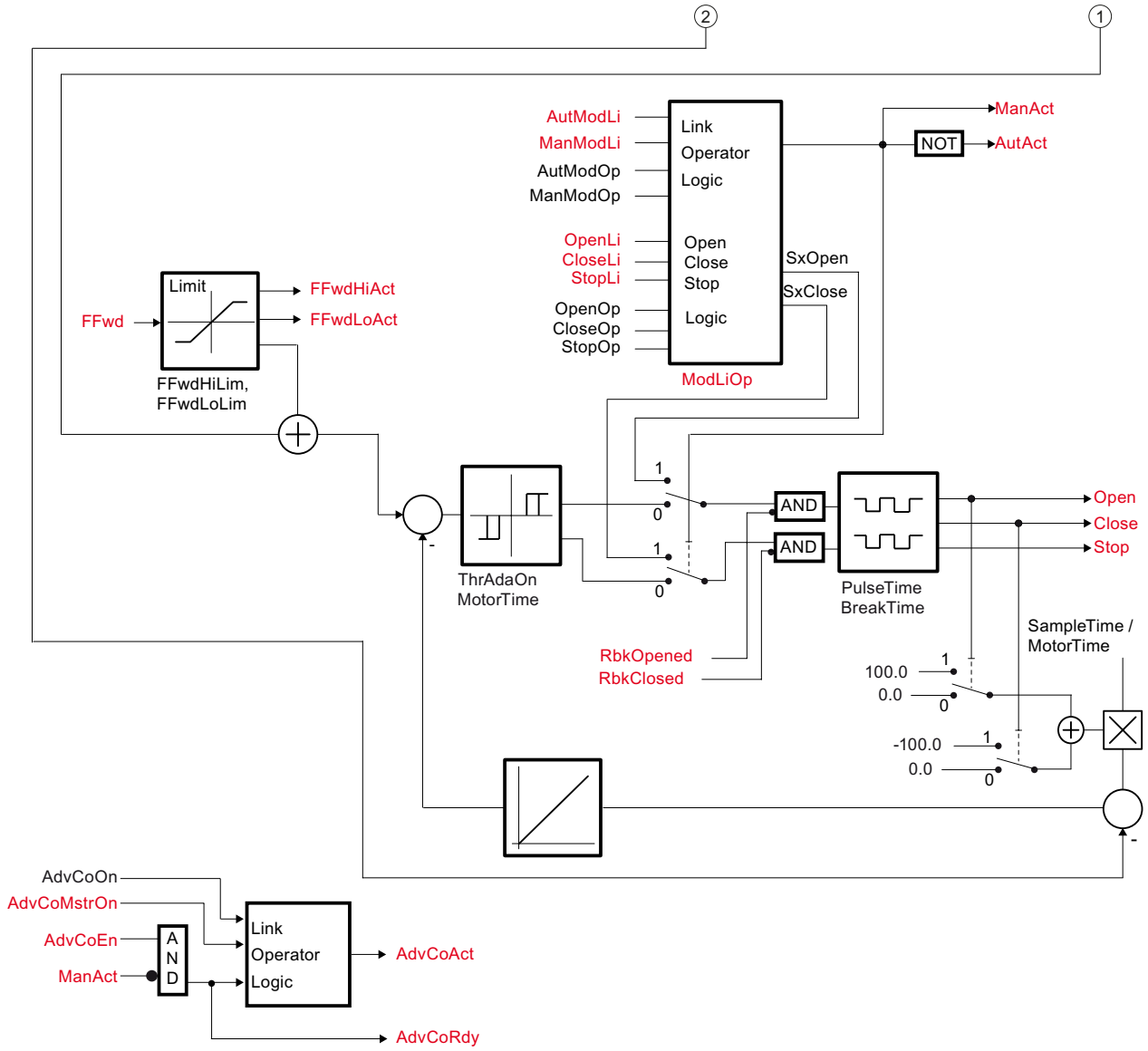
Step controller with position feedback


Step controller with position feedback and tracking
SP_ExtLi
SP_IntLi
SP_ExtOp
SP_IntOp
Sp_Ext/Int
Link
Operator
Logik
SP_LiOp
SP_ExtAct
SP_Ext
Limit
SP_ExHiAct,
SP_ExLoAct
SP_ExtOut
SP_ExHiLim,
SP_ExLoLim
SP_RateOn
SP_TrkPV AND
(MV_ForOn OR
ManAct OR MV_TrkOn)
Gradient
PV
SP
SP_RmpOn
SP_RmpTar
Reject
Ramp
SP_InHiLim,
SP_InLoLim
SP_RmpModTime,
SP_RmpTime,
SP_UpRaLim,
SP_DnRaLim
SP_UpRaLim,
SP_DnRaLim
SP_UpRaAct,
SP_DnRaAct
AdvCoMV
Limit
SP_InHiLim,
SP_InLoLim
AdvCoAct AND
AdvCoModSP
SP_Int
is limited and
tracked
Reject
SP_InHiLim → SP_InHiOut,
SP_InLoLim → SP_InLoOut
ER
Alarm
ER_AH_Act
ER_AL_Act
ER_AH_Lim, ER_AL_Lim, ER_Hyst,
ER_AH_En, ER_AL_En,
ER_AH_MsgEn, ER_AL_MsgEn,
ER_A_DC, ER_A_DG
PropSel
f( )
PropFacSP
0.0
D_FbkSel
DeadBand
PV
TD, DiffGain
SimOn
SimPV
PV
PV_OUT
Gain
%
phys
GainEff
NegGain,
NormPV
TI
f( )
PropFacSP
IntHoldPos,
IntHoldNeg
MV_Offset
TI = 0
Write PV_OUT to
SimPV, when
applicable: Sim = 0
Alarm
PV_AH_Act
PV_WH_Act
PV_TH_Act
PV_TL_Act
PV_WL_Act
PV_AL_Act
PV_xH_Lim, PV_xL_Lim, PV_Hyst,
PV_xH_En, PV_xL_En,
PV_xH_MsgEn, PV_xH_MsgEn,
PV_x_DC, PV_x_DG,
x = A/W/T
1


1
AdvCoAct
AND NOT
AdvCoModSP
AdvCoMV
MV_TrkOn
MV_Trk
AutModLi
ManModLi
AutModOp
ManModOp
Link
Operator
Logic
ModLiOp
ManAct
AutAct
MV_TrkOn
MV_ForOn
AND
ER
ConZone,
MV_HiLim,
MV_LoLim
Limit
MV_HiAct,
MV_LoAct
MV_HiLim,
MV_LoLim
MV_ForOn
MV_Force
MV
Reject
Man
is limited
and tracked
ManHiLim
ManLoLim
ManHiOut,
ManLoOut
Limit
FFwd
FFwdHiAct
FFwdLoAct
FFwdHiLim,
FFwdLoLim
Alarm
Rbk
RbkWH_Act
RbkWL_Act
RbkWH_Lim,
RbkWL_Lim,
RbkHyst,
RbkWH_En,
RbkWL_En,
RbkWH_MsgEn,
RbkWL_MsgEn
AND
Open
Close
Stop
ThrAdaOn
MotorTime
PulseTime
BreakTime
RbkOpened
RbkClosed
AdvCoEn
MV_TrkOn
MV_ForOn
AdvCoModSP
AdvCoModSP
ManAct
OR
AdvCoOn
AdvCoMstrOn
AdvCoAct
AdvCoRdy

Step controller without position feedback





See also

Description of PIDStepL (Page 501)

PIDStepL modes (Page 505)

PIDStepL functions (Page 506)

PIDStepL error handling (Page 514)

PIDStepL messaging (Page 516)

PIDStepL I/Os (Page 519)

4.8.8 Operator control and monitoring

4.8.8.1 PIDStepL views

Views of the PIDStepL block

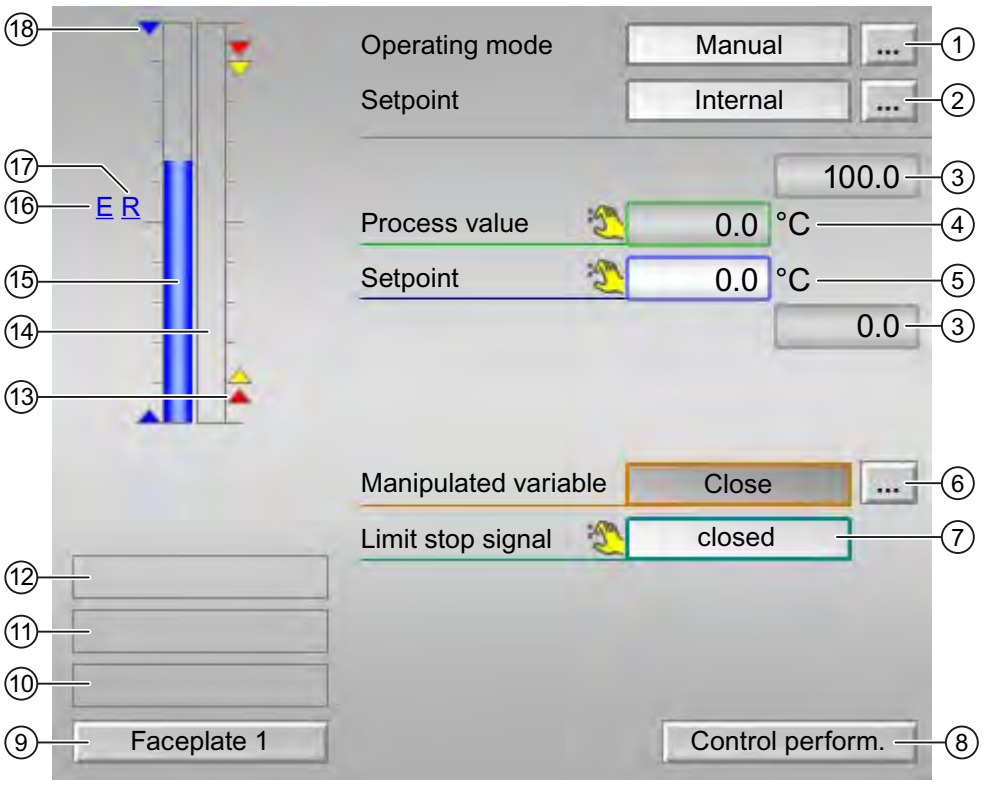
The block PIDStepL provides the following views:

- Standard view of PIDStepL without position feedback (Page 537)
- Standard view of PIDStepL with position feedback (Page 541)
- Message view (Page 169)
- Limit value view of PID controllers (Page 160)
- Trend view (Page 172)
- Ramp view (Page 167)
- Parameter view of PID controllers (Page 151)
- PIDStepL preview (Page 545)
- Memo view (Page 171)
- Batch view (Page 170)
- Block icons for PID and FM controller (Page 181)

Refer to the Structure of the faceplate (Page 123) and Block icon structure (Page 174) sections for general information about the faceplate and block icon.

4.8.8.2 Standard view of PIDStepL without position feedback

Standard view of PIDStepL without position feedback



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 29)
- Automatic mode (Page 29)
- Program mode for closed-loop controllers (Page 34)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Display and switch the setpoint specification

This area shows how to specify the setpoint. The setpoint can specified as follows:

- By the application (External, CFC/SFC)
- By the user direct in the faceplate (Internal).

Refer to the Switching operating states and operating modes (Page 127) section for information on changing the setpoint specification.

You can find additional information on this in the Setpoint input - internal and external (Page 96) section.

(3) High and low scale range for the process value

These values provide information on the display range for the bar graph of the process value. The scale range is defined in the engineering system.

(4) Display of the process value including signal status

This area shows the current process value with the corresponding signal status.

(5) Display and change the setpoint including signal status

This area shows the current setpoint with the corresponding signal status.

Refer to the Changing values (Page 129) section for information on changing the setpoint. The setpoint specification also needs to be set to "Internal" for this block.

(6) Display and change manipulated variables

This area shows you the currently valid manipulated variable. Refer to the Switching operating states and operating modes (Page 127) section for information on changing the manipulated variable.

The following manipulated variables can be selected:

- Open
- Stop
- Close

(7) Display feedback

The following feedback can be displayed:

- Opened
- Closed

(8) Navigation button for switching to the standard view of the ConPerMon block

Use this navigation button to reach the standard view of the ConPerMon block. The visibility of this navigation button depends on the configuration in the engineering system (ES).

See also the Calling further faceplates (Page 43) section for more on this.

(9) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

(10) Display area for states of the block

This area provides additional information on the operating state of the block:

- Maintenance

Additional information on the display area for states of the block is available in section Maintenance release (Page 47).

(11) Display area for states of the block

This area provides additional information on the operating state of the block:

- Simulation

You can find additional information on this in the Simulating signals (Page 93) section.

(12) Display area for states of the block

This area provides additional information on the operating state of the block (from high to low according to priority):

- Optimization
- Tracking
- Forced tracking

(13) Limit display

These colored triangles show you the configured limits in the respective bar graph.

(14) Bar graph for the process value

This area shows the current process value in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(15) Bar graph for the setpoint

This area shows the current setpoint in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(16) Display for external setpoint

This display [E] is only visible when you have selected "internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

(17) Display for the target setpoint of the setpoint ramp

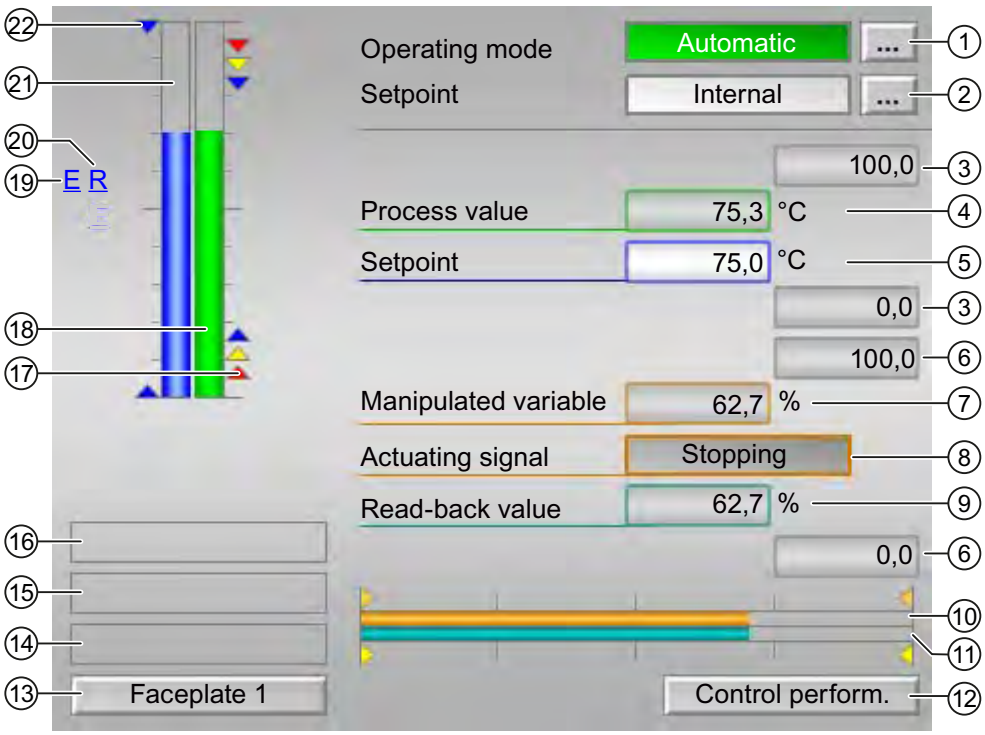
This display [R] shows you the target setpoint and is only visible if you have enabled ramp generation in the Ramp view (Page 167).

(18) Display for limits

These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the ES.

4.8.8.3 Standard view of PIDStepL with position feedback

Standard view of PIDStepL with position feedback



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 29)
- Automatic mode (Page 29)
- Program mode for closed-loop controllers (Page 34)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Display and switch the setpoint specification

This area shows how to specify the setpoint. The setpoint can specified as follows:

- By the application (External, CFC/SFC)
- By the user direct in the faceplate (Internal).

Refer to the Switching operating states and operating modes (Page 127) section for information on changing the setpoint specification.

You can find additional information on this in the Setpoint input - internal and external (Page 96) section.

(3) High and low scale range for the process value

These values provide information on the display range for the bar graph of the process value. The scale range is defined in the engineering system.

(4) Display of the process value including signal status

This area shows the current process value with the corresponding signal status.

(5) Display and change the setpoint including signal status

This area shows the current setpoint with the corresponding signal status.

Refer to the Changing values (Page 129) section for information on changing the setpoint. The setpoint specification also needs to be set to "Internal" for this block.

(6) High and low limits of the manipulated variable

You can only display and change a manipulated variable in manual mode.

(7) Display and change the manipulated variable

You can only display and change a manipulated variable in manual mode.

(8) Display and change manipulated variables

This area shows you the currently valid manipulated variable. Refer to the Switching operating states and operating modes (Page 127) section for information on changing the manipulated variable.

The following manipulated variables can be selected:

- Open
- Stop
- Close

(9) Display feedback

The following feedback can be displayed:

- Opened
- Closed

(10) Bar graph for the manipulated variable

The bar graph for the manipulated variable is only available in manual mode.

(11) Bar graph for the readback value

The bar graph for the readback value is only available in manual mode.

(12) Navigation button for switching to the standard view of the ConPerMon block

Use this navigation button to reach the standard view of the ConPerMon block. The visibility of this navigation button depends on the configuration in the engineering system (ES).

Refer also to the Calling further faceplates (Page 43) section for more on this.

(13) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

(14) Display area for states of the block

This area provides additional information on the operating state of the block:

- Maintenance

Additional information on the display area for states of the block is available in section Maintenance release (Page 47).

(15) Display area for states of the block

This area provides additional information on the operating state of the block:

- Simulation

You can find additional information on this in the Simulating signals (Page 93) section.

(16) Display area for states of the block

This area provides additional information on the operating state of the block (from high to low according to priority):

- Optimization
- Tracking
- Forced tracking

(17) Limit display

These colored triangles show you the configured limits in the respective bar graph.

(18) Bar graph for the process value

This area shows the current process value in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(19) Display for external setpoint

This display [E] is only visible when you have selected "internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

(20) Display for the target setpoint of the setpoint ramp

This display [R] shows you the target setpoint and is only visible if you have enabled ramp generation in the Ramp view (Page 167).

(21) Bar graph for the setpoint

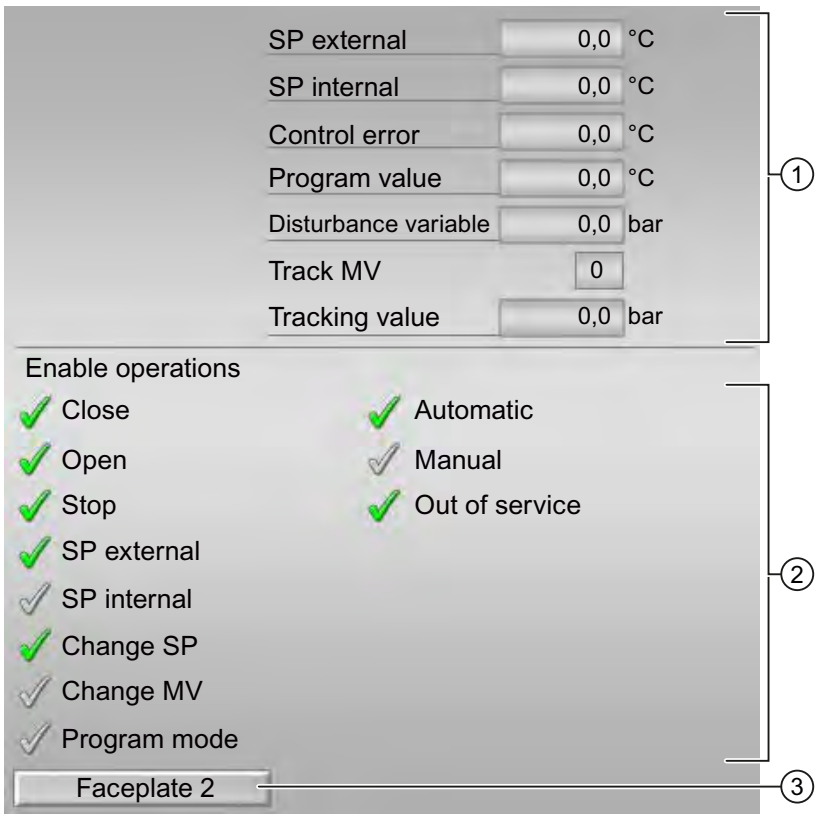
This area shows the current setpoint in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(22) Display for limits

These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the ES.

4.8.8.4 PIDStepL preview

PIDStepL preview



(1) Preview area

This area shows you a preview for the following values:

- SP external: currently applicable external setpoint
- SP internal: currently applicable internal setpoint
- Control error: Current control error
- Program value: specified value for program mode
- Disturbance: additive value for feedforward control
- Track MV: track manipulated variable (value is 1)
- Tracking value: effective manipulated variable for "Track manipulated variable"

(2) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions

(3) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

4.9 Ratio - ratio controlling

4.9.1 Description of Ratio

Object name (type + number) and family

Type + number: FB 1883

Family: Control

Area of application for Ratio

The block is used for the following applications:

- Forming a ratio

How it works

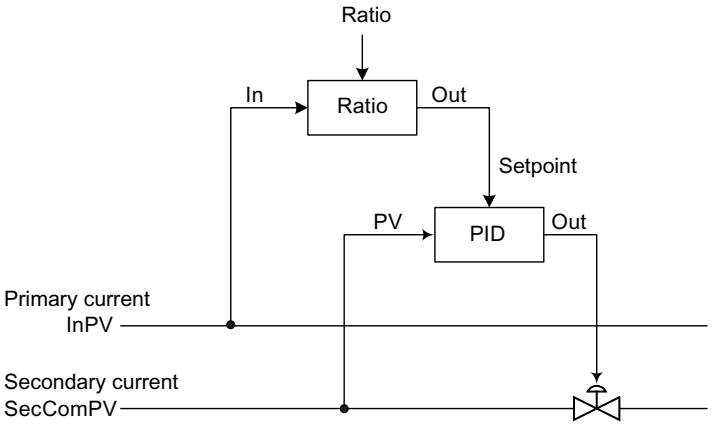
The block is used to create a ratio, for example in a ratio control. It is also used to set elements (for example, synchronization control loop), or to influence the reference variable of a cascade.

The block operates according to the equation: `Out = In · RatioX + Offset`

Where:

- `RatioX` is either the `RatioInt` or `RatioExt` I/O
- `Offset` is the offset value to be added to the output value

`In` is based on the interconnection whereas `RatioX` is selected based on the internal/external selection.



Please also refer to the Ratio control (Page 1440) section for information on calculating the current ratio.

Configuring OBs

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

Status word allocation for `Status` parameter

The description for each parameter can be found in the Ratio I/Os (Page 555) section.

Status bit	Parameter
0 – 1	Not used
2	Not used
3	`OosAct.Value`
4	`OosLi.Value`
5	Not used
6	`OnAct.Value`
7	Not used
8	`RatExtAct.Value`
9	`RatLoAct.Value`
10	`RatHiAct.Value`
11	`OutLoAct.Value`
12	`OutHiAct.Value`
13	`RatTrkExt`
14 - 31	Not used

See also

Ratio functions (Page 550)

Ratio messaging (Page 554)

Ratio block diagram (Page 559)

Ratio error handling (Page 553)

Ratio modes (Page 549)

4.9.2 Ratio modes

Ratio operating modes

The block can be operated using the following modes:

- On (Page 27)
- Out of service (Page 27)

"On"

General information on the "On" mode is available in the section On (Page 27).

"Out of service"

You can find general information about the "Out of service" mode in the Out of service (Page 27) section.

The last valid value is provided at the `Out` output in this operating mode. The `OutHiAct`, `OutLoAct`, `RatHiAct` and `RatLoAct` output parameters are reset.

See also

Ratio block diagram (Page 559)

Ratio I/Os (Page 555)

Ratio messaging (Page 554)

Ratio error handling (Page 553)

Ratio functions (Page 550)

Description of Ratio (Page 547)

4.9.3 Ratio functions

Functions of Ratio

The functions for this block are listed below.

Internal or external ratio

You can use the `RatLiOp` parameter to specify if the ratio should be specified internally or externally:

- `RatLiOp = 0`: the ratio (`RatioInt`) is specified by the operator. The operator can decide if the specification should be made internally (`RatIntOp = 1`) or externally (`RatExtOp = 1`).
- `RatLiOp = 1`: the ratio (`RatioExt`) is specified by an interconnection. The interconnection is used to determine if the specification should be made internally (`RatIntLi = 1`) or externally (`RatExtLi = 1`).

Bumpless switchover from external to internal ratio

The parameter `RatTrkExt = 1` is used so that the internal ratio tracks the external setpoint to achieve a bumpless switchover from the external to the internal ratio. This allows unwanted jumps at the output parameter to be avoided.

Limiting the ratio

You can limit the ratio using the `RatHiLim` parameter (high) or `RatLoLim` parameter (low).

The external ratio `RatioExt` is limited to the value you have specified if a limit is violated. This is then also used to form the output value `Out`. This is also displayed at the `RatHiAct` or `RatLoAct` output parameters with a 1.

The internal ratio `RatioInt` is checked against the value you have specified. If a value is outside the specified limit, it is reset to the most recent valid value.

Limiting the output value

You can limit the output value using the `OutHiLim` parameter (high) or `OutLoLim` parameter (low).

The limit violation is displayed at the `OutHiAct` or `OutLoAct` output parameters with a 1.

Simulating signals

This block provides the standard function Simulating signals (Page 93).

You can simulate the analog input value `SimIn`.

Display and control field for process values and setpoints

This block provides the standard function Display and operator input area for process values and setpoints (Page 42).

Selecting a unit of measure

The block provides the standard function Selecting a unit of measure (Page 53).

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

Operator control permissions

The block has the following operator control permissions (Page 45) for the parameter `OS_Perm`:

Bit	Function
0	Not allocated
1	1 = Operator can switch to On mode
2	Not allocated
3	1 = Operator can switch to out of service mode
4	1 = Operator can switch to external
5	1 = Operator can switch to internal
6	1 = Operator can change the internal ratio
7 - 10	Not allocated
11	1 = Operator can switch to simulation
12	1 = Operator can perform bumpless switchover
13	Not allocated
14	1 = Operator can change `RatHiLim`
15	1 = Operator can change `RatLoLim`
16	1 = Operator can change `OutHiLim`
17	1 = Operator can change `OutLoLim`
18 - 31	Not allocated

Forming and outputting signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status `ST_Worst` is formed from the following parameters:

- `Out.ST`
- `SecComPV.ST`
- `InPV.ST`
- `RatioExt.ST`
- `Offset.ST`

The signal status of the `Out` output parameter corresponds to the input parameter `In`.

The signal status of the current ratio calculation corresponds to the status of the input parameter `SecComPV`.

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the Feature parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)
1	Reaction to the out of service mode (Page 199)

See also

Description of Ratio (Page 547)

Ratio messaging (Page 554)

Ratio I/Os (Page 555)

Ratio block diagram (Page 559)

Ratio error handling (Page 553)

Ratio modes (Page 549)

4.9.4 Ratio error handling

Ratio troubleshooting

Refer to section Error handling (Page 117) for basic instructions on how to troubleshoot all blocks.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Default value when implementing the block; block will not be processed.
0	No active fault
30	The input value of `In` can no longer be displayed in the REAL number field.
51	`RatLiOp = 1` and `RatExtLi = 1` and `RatInLi = 1`

See also

Ratio block diagram (Page 559)

Ratio I/Os (Page 555)

Ratio messaging (Page 554)

Ratio functions (Page 550)

Ratio modes (Page 549)

Description of Ratio (Page 547)

4.9.5 Ratio messaging

Messaging

The block does not offer messaging.

See also

Description of Ratio (Page 547)
Ratio functions (Page 550)
Ratio I/Os (Page 555)
Ratio block diagram (Page 559)
Ratio error handling (Page 553)
Ratio modes (Page 549)

4.9.6 Ratio I/Os

Ratio I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
Feature	I/O for additional functions (Page 550)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
In	Analog input	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
InPV	Input for process variable	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
InUnit	Unit of measure for input parameter In	INT	1001
Offset	Offset	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
OnOp	1 = "On" mode via operator	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC (0-1 edge transition)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OS_Perm	I/O for operator control permissions (Page 550)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
OutHiLm	High limit for output value	REAL	100.0
OutLoLm	Low limit for output value	REAL	0.0
RatExtLi	1 = Select external ratio (via interconnection)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RatExtOp	1 = Select external ratio (via operator)	BOOL	0
RatHiLim	High limit	REAL	100.0
RatIntOp	1 = Select internal ratio (via operator)	BOOL	1

Parameter	Description	Type	Default
RatioExt	External ratio	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#80
RatioInt	Internal ratio	REAL	1.0
RatioUnit	Unit of measure for the RationInt, RatioExt input parameter or RatioPV (output parameter)	INT	0
RatLiOp	Select ratio source (internal/external): 1 = Via interconnection 0 = Via operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RatLoLim	Low limit	REAL	0.0
RatOpScale	Limit for scale in bar graph of faceplate	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
RatTrkExt	1 = Bumpless switchover from external to internal ratio active	BOOL	0
SecComPV	Process value of secondary component	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SecComUnit	Unit of measure for input parameter SecComPV	INT	1001
SelFp1	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
SimIn	Analog input value used for SimOn = 1	REAL	0.0
SimOn	1 = Simulation on	BOOL	0
UserStatus	Freely assignable user area for status word (bit 24 to bit 31)	BYTE	16#00

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see Ratio error handling (Page 553)	INT	-1
OnAct	1 = "On" mode enabled	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
OosAct	1 = Block is "Out of service"	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
Out	Output	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
OutHiAct	1 = High limit overshoot	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OutHiLmOut	Output of high limit	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
OutLoAct	1 = Low limit overshoot	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OutLoLmOut	Output of low limit	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
RatExtAct	1 = External ratio used 0 = Internal ratio used	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RatHiAct	1 = Limit (high) for ratio active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RatioPV	Current ratio	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
RatLoAct	1 = Limit (low) for ratio active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
ST_Worst	Worst signal status	BYTE	16#80
Status	Status word (Page 547)	DWORD	16#00

See also

Ratio messaging (Page 554)

Ratio block diagram (Page 559)

Ratio modes (Page 549)

4.9.7 Ratio block diagram

Ratio block diagram

A block diagram is not provided for this block.

See also

Ratio I/Os (Page 555)

Ratio messaging (Page 554)

Ratio error handling (Page 553)

Ratio functions (Page 550)

Ratio modes (Page 549)

Description of Ratio (Page 547)

4.9.8 Operator control and monitoring

4.9.8.1 Ratio views

Views of the Ratio block

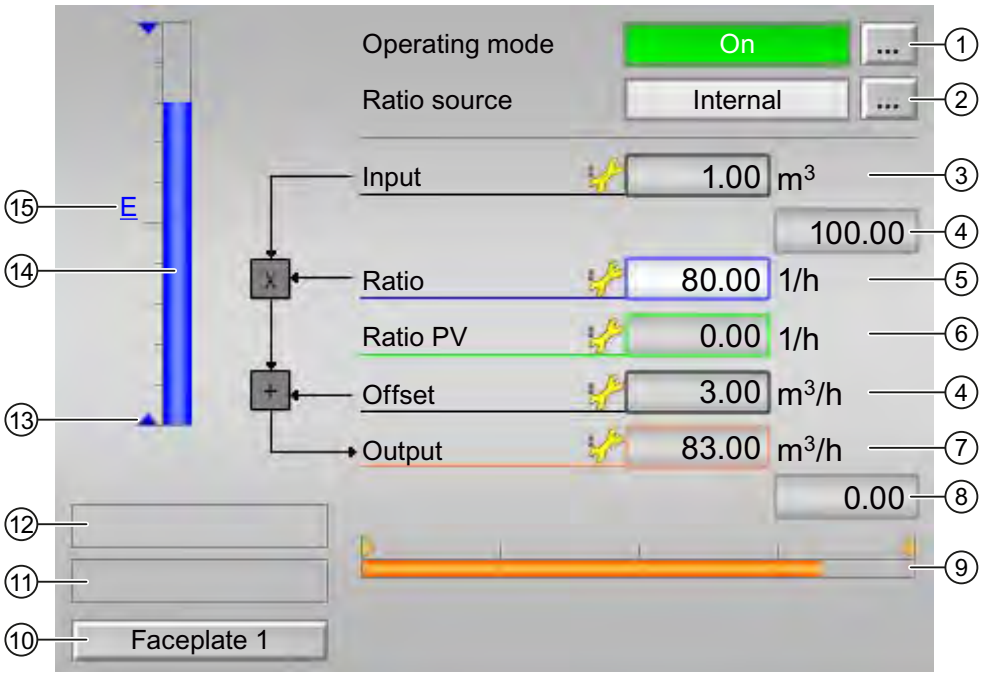
The block Ratio provides the following views:

- Ratio standard view (Page 560)
- Trend view (Page 172)
- Ratio parameter view (Page 563)
- Ratio preview (Page 565)
- Memo view (Page 171)
- Block symbol for ratio (Page 566)

Refer to the sections Structure of the faceplate (Page 123) and Block icon structure (Page 174) for general information on the faceplate and block icon.

4.9.8.2 Ratio standard view

Ratio standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 27)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Display and switch the ratio source

This area shows you the currently valid signal source for the ratio setpoint. The following signal sources can be shown here:

- External
- Internal

Refer to the Switching operating states and operating modes (Page 127) section for information on changing the signal setpoint.

(3) Display the Input value

The faceplate is a schematic representation of the function of the ratio block as a signal flow chart:

```
Output = Input * Ratio + Offset
```

The input is typically the flow setpoint or actual value of the primary component of a ratio controller.

(4) High and low scale range for the ratio

The scale range is based on the bar graph for the ratio specification.

(5) Display and switch the ratio specification

This area shows you the currently valid specification for the ratio.

The ratio source ② needs to be set to "Internal" in order to change this value.

Refer to the Changing values (Page 129) section for information on changing this value.

(6) Display and switch the ratio PV

This area shows you the current ratio actual value with the corresponding signal status, i.w. the ratio of the actually measured PV from the active controller. The task of the ratio controller is to set the flow of all components so that the actual ratio approximates the specified ratio as closely as possible.

(7) Display of the offset

This area shows the currently valid offset.

(8) Display the output value

This area shows the current output value Out, which typically serves as the setpoint for the flow of the secondary component.

(9) Bar graph for the output

This display is a graphic representation of the output value with the limits set in the ES (orange triangles, input parameters `OutHiLmOut` and `OutLoLmOut`).

(10) Display of limit

This status display is based on the limit of the output value Out.

(11) Display area for states of the block

This area shows if the output value has violated the range limits:

- "Output >= HL"
- "Output <= LL"

You can set the range limits in the parameter view (Page 563) of the block.

(12) Display area for states of the block

This area provides additional information on the operating state of the block:

- Simulation

You can find additional information on this in the Simulating signals (Page 93) section.

(13) Display of the limits for the ratio

This blue triangle shows the configured range limits for the ratio.

(14) Bar graph for the ratio specification

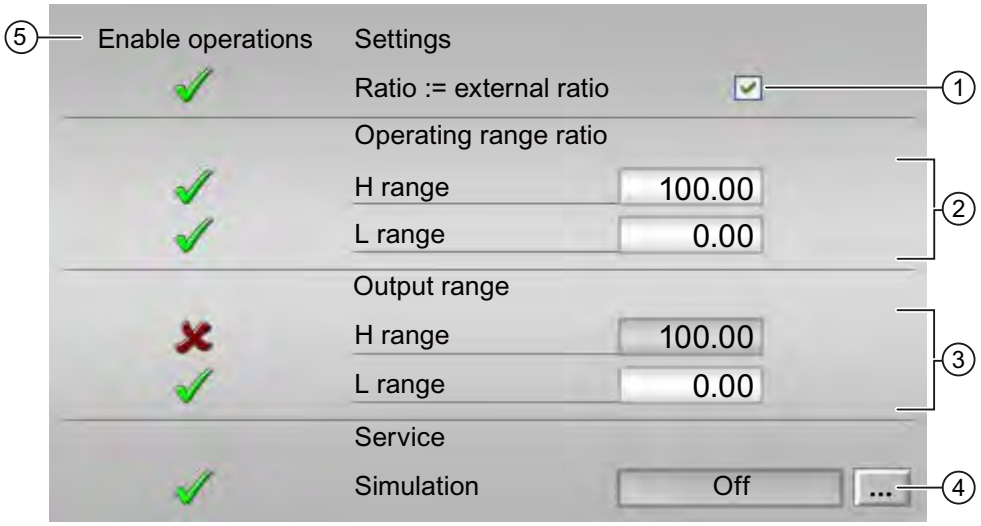
This area shows you the currently valid specification for the ratio in the form of a bar graph. The visible area in the bar graph depends on the configuration of the ratio in the engineering system (ES).

(15) Display for external setpoint

This display [E] is only visible when you have selected "internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

4.9.8.3 Ratio parameter view

Ratio parameter view



(1) Settings Ratio := external ratio

When the check box is selected ☑, the ratio is bumplessly switched from external to internal.

(2) Operating range ratio

This is where you specify the operating range for the ratio (input parameters `RatHiLim` or `RatLoLim`). The range is indicated by a blue triangle in the standard view of the bar graph.

(3) Range for output value

This is where you specify the operating range for the output value (input parameters `OutHiLmOut` or `OutLoLmOut`).

(4) Service

You can select the following functions in this area:

- Simulation

You can find information about this in the following sections:

- Switching operating states and operating modes (Page 127)
- Simulating signals (Page 93)
- Maintenance release (Page 47).

(5) Enable operations

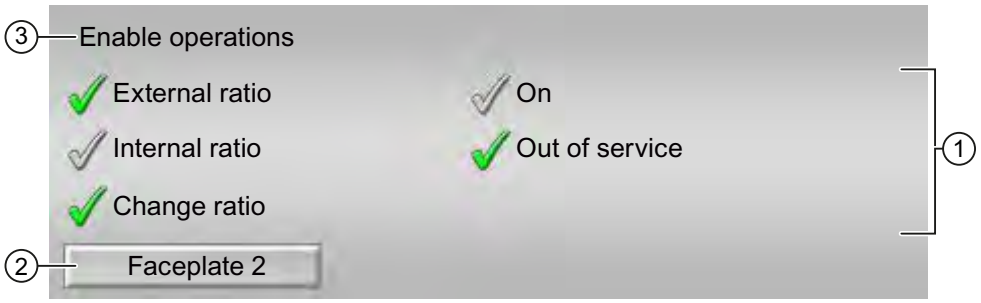
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions.

4.9.8.4 Ratio preview

Ratio preview



(1) The following operation enables for parameters are shown here:

- External ratio: You can change the external ratio
- Internal ratio: You can change the internal ratio
- Change ratio: You can change the ratio
- On: You can switch to On operating mode
- Out of service: You can switch to the out of service operating mode.

(2) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

(3) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

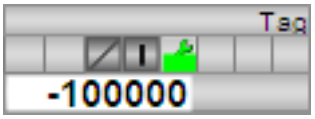
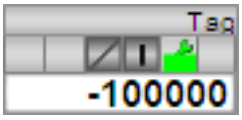
- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions.

4.9.8.5 Block symbol for ratio

Properties of the ratio block icon

A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits
- Operating modes
- Internal and external setpoint specification
- Signal status, maintenance release
- Displays for bridging interlocks
- Interlocks
- Memo display
- Motor state display

Icons	Selection of the block icon in CFC	Special features
	1	Block icon in the full display
	2	
	3	
	4	
	5	
	6	
	7	

Icons	Selection of the block icon in CFC	Special features
	8	
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181).

4.10 SplRange - signal splitter

4.10.1 Description of SplRange

Object name (type + number) and family

Type + number: FC 321

Family: Control

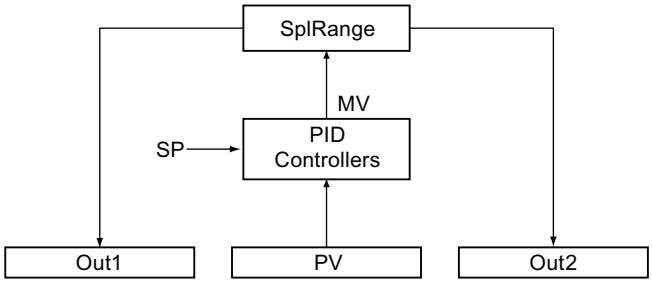
Area of application for SplRange

The block is used for the following applications:

- Splitting the output signal of a PID controller

How it works

The block is used to split signals output coming from a PID controller. You can use this controller to control two valves, for example.



The output parameters are calculated as follows:

```
Out1 = Out1Scale.Low + ((NeutPos - DeadBand - In) / (NeutPos -
DeadBand - InScale.Low)) · (Out1Scale.High - Out1Scale.Low)
```

```
Out2 = Out2Scale.Low + ((NeutPos + DeadBand - In) / (NeutPos +
DeadBand - InScale.High)) · (Out2Scale.High - Out2Scale.Low)
```

The neutral position (`NeutPos`) forms the reference point for selecting the individual splitter profiles. For details, refer to the SplRange functions (Page 571) section.

Refer to SplRange I/Os (Page 575) to learn the meaning of the parameters.

The block is installed in the run sequence downstream of the controller block. The (`MV`) manipulated variable output of the controller block is interconnected to the input `In` of the SplRange block.

The neutral position (`NeutPos`) and the dead band zone (`DeadBand`) are set by means of the corresponding parameters. `DeadBand` must be configured to be less than `NeutPos`.

`Out1` and `Out2` are adapted to the physical variable by configuring the high/low limits of `Out1` and `Out2`.

The Out1Act or Out2Act output parameter indicates (=1) that the corresponding Out1 or Out2 output parameter is active if the In input value is less than (for Out1) the neutral position (NeutPos) or the In input value is greater than the neutral position (NeutPos), depending on the dead band (Deadband).

Configuration

Use CFC editor to install the block in the OB in which the controller block runs whose manipulated variable is being processed.

A (Templates) template is provided for the SplRange block, an example (APL_Example_xx, xx refers to the language variant) for the process tag type in the Advanced Process Library with an application case for this block:

- Split-range control (Page 1438)

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status parameter

This block does not provide the Status parameter.

See also

SplRange messaging (Page 574)

SplRange block diagram (Page 577)

SplRange error handling (Page 574)

SplRange modes (Page 570)

4.10.2 SplRange modes

SplRange operating modes

This block does not provide any modes.

See also

SplRange I/Os (Page 575)
SplRange messaging (Page 574)
SplRange error handling (Page 574)
SplRange functions (Page 571)
Description of SplRange (Page 568)
SplRange block diagram (Page 577)

4.10.3 SplRange functions

Functions of SplRange

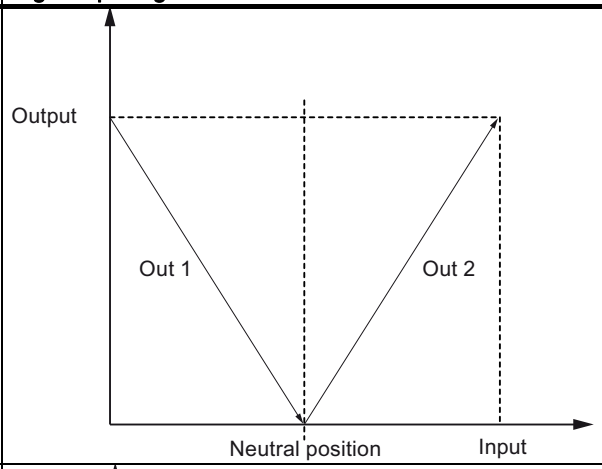
The functions for this block are listed below.

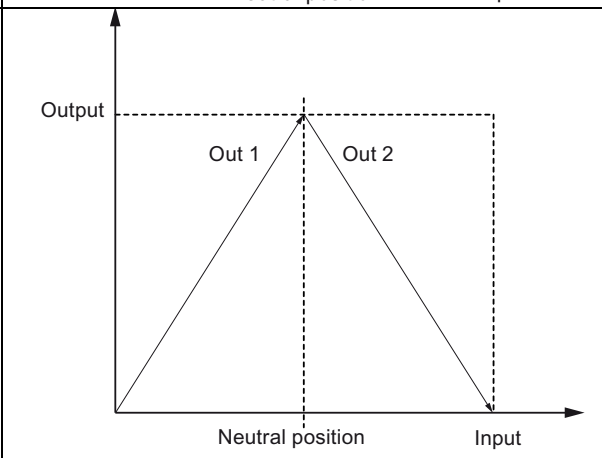
Splitting of the output signal of a controller

The `In` block input can be used to split the output signal of a controller.

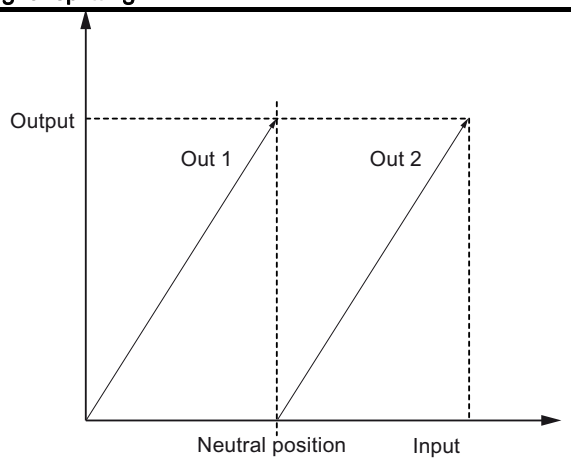
The table below shows the six types of signal splitting that are available:

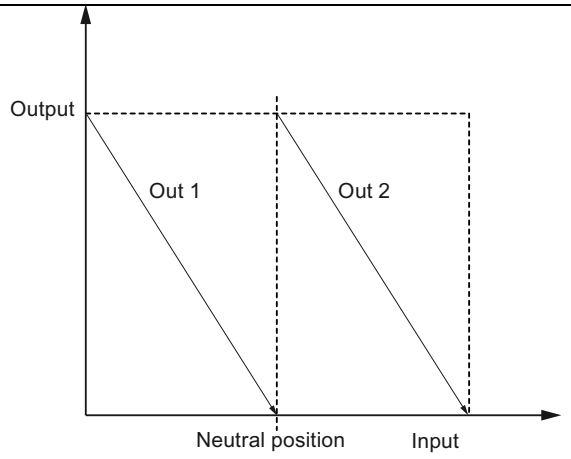
Case	Signal splitting	Setting the I/Os

1		`Out1Scale.Low = 0.0` and `Out2Scale.High = 100.0`
		`Out2Scale.Low = 0.0` and `Out1Scale.High = 100.0`
		`DeadBand = 0.0` and `NeutPos = 50.0`

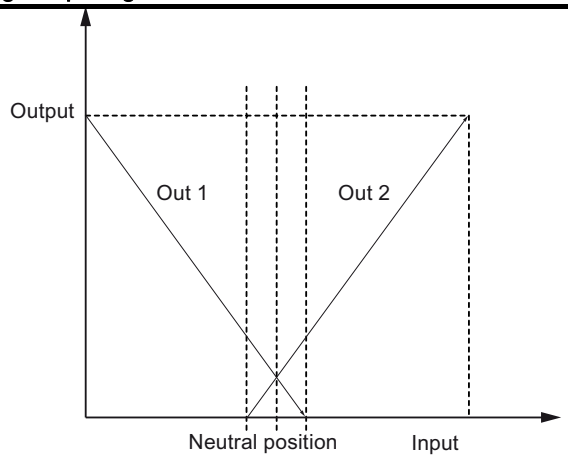
2		`Out1Scale.Low = 100.0` and `Out2Scale.High = 0.0`
		`Out2Scale.Low = 100.0` and `Out1Scale.High = 0.0`
		`DeadBand = 0.0` and `NeutPos = 50.0`

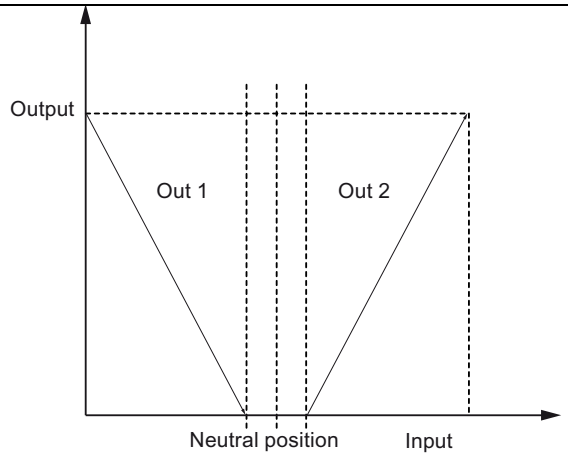
Case	Signal splitting	Setting the I/Os

3		`Out1Scale.Low = 100.0` and `Out2Scale.High = 0.0`
		`Out2Scale.Low = 0.0` and `Out1Scale.High = 100.0`
		`DeadBand = 0.0` and `NeutPos = 50.0`

4		`Out1Scale.Low = 0.0` and `Out2Scale.High = 100.0`
		`Out2Scale.Low = 100.0` and `Out1Scale.High = 0.0`
		`DeadBand = 0.0` and `NeutPos = 50.0`

Case	Signal splitting	Setting the I/Os

5		`Out1Scale.Low = 0.0` and `Out2Scale.High = 100.0`
		`Out2Scale.Low = 0.0` and `Out1Scale.High = 100.0`
		`DeadBand = -10.0` and `NeutPos = 50.0`

6		`Out1Scale.Low = 0.0` and `Out2Scale.High = 100.0`
		`Out2Scale.Low = 0.0` and `Out1Scale.High = 100.0`
		`DeadBand = 10.0` and `NeutPos = 50.0`

See also

Description of SplRange (Page 568)
SplRange messaging (Page 574)
SplRange I/Os (Page 575)
SplRange block diagram (Page 577)
SplRange error handling (Page 574)
SplRange modes (Page 570)

4.10.4 SplRange error handling

SplRange troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Predefined value when inserting the block, the block is not processed.
0	There is no error.
30	The value of In can no longer be displayed in the REAL number field.

See also

SplRange block diagram (Page 577)
SplRange I/Os (Page 575)
SplRange messaging (Page 574)
SplRange functions (Page 571)
SplRange modes (Page 570)
Description of SplRange (Page 568)

4.10.5 SplRange messaging

Messaging

The block does not have any message functionality.

See also

Description of SplRange (Page 568)
SplRange functions (Page 571)
SplRange I/Os (Page 575)
SplRange block diagram (Page 577)
SplRange error handling (Page 574)
SplRange modes (Page 570)

4.10.6 SplRange I/Os

SplRange I/Os

Input parameters

Parameter	Description	Type	Default
DeadBand	Width of dead band	REAL	0.0
EN	1 = Called block will be processed	BOOL	1
In	Input value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
InScale	Limit range for the input signal	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
NeutPos	Neutral position	REAL	50.0
Out1Scale	Limit range of output 1	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
Out2Scale	Limit range of output 2	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see SplRange error handling (Page 574).	INT	-1
Out1	Output value 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Out1Act	1 = Output 1 is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Out2	Output value 2	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Out2Act	1 = Output 2 is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

Description of SplRange (Page 568)

SplRange functions (Page 571)

SplRange messaging (Page 574)

SplRange block diagram (Page 577)

SplRange modes (Page 570)

4.10.7 SplRange block diagram

SplRange block diagram

A block diagram is not provided for this block.

See also

SplRange I/Os (Page 575)
SplRange messaging (Page 574)
SplRange error handling (Page 574)
SplRange functions (Page 571)
Description of SplRange (Page 568)
SplRange modes (Page 570)

Motor and valve blocks

5

5.1 MotL - Motor

5.1.1 Description of MotL

Object name (type + number) and family

Type + number: FB 1850

Family: Drives

Area of application for MotL

The block is used for the following applications:

- Control of motors with one control signal

How it works

The block is used to control motors. Various inputs are available for controlling the motor.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

Status word allocation for `Status1` parameter

Status bit	Parameter
0	`Occupied`
1	`BatchEn`
2	`SimOn`
3	`OosAct.Value`
4	`OosLi.Value`
5	`0 = ManAct.Value` `1 = AutoAct.Value`
6	`LocalAct.Value`
7	`LockAct.Value`
8	`Start.Value`
9	Motor is stopped
10	Not used
11	`MonStaErr.Value`
12	`MonDynErr.Value`
13	`BypProt`
14	Invalid signal status
15	Mode changeover error
16	`1 = Intlock` is active
17	`1 = Permit` is active
18	`1 = Protect` is active
19	`Trip`
20	`StartForce`
21	`StopForce`
22	Not used
23	"Interlock" button is enabled
24	Motor error
25	Not used
26	Bypass information from previous function block
27	Automatic preview for starting
28	Automatic preview for stopping
31	Not used
30	"Permission" button is enabled
31	"Protection" button is enabled

Status word allocation for `Status2` parameter

Status bit	Parameter
0	`MsgLock`
1	`AV_AH_Act.Value`
2	`AV_WH_Act.Value`
3	`AV_TH_Act.Value`
4	`AV_TL_Act.Value`
5	`AV_WL_Act.Value`
6	`AV_AL_Act.Value`
7	`AV_AH_En`
8	`AV_WH_En`
9	`AV_TH_En`
10	`AV_TL_En`
11	`AV_WL_En`
12	`AV_AL_En`
13	`AV_AH_MsgEn`
14	`AV_WH_MsgEn`
15	`AV_TH_MsgEn`
16	`AV_TL_MsgEn`
17	`AV_WL_MsgEn`
18	`AV_AL_MsgEn`
19	1 = No impact of input signals on local mode with `LocalSetting = 2` and `LocalSetting = 4`
20	Motor is stopped
21	The motor is stopped.
22	The motor is started.
23	the motor is running
24	Error in motor
25 - 29	Not used
30	Display for interlocks in block icon
31	`MS_RelOp`

Status word allocation for `Status3` parameter

Status bit	Parameter
0 - 18	Not used
19	1 = Enabled for emergency stop push button (`Feature` bit Enabling rapid stop via faceplate (Page 192))
20 - 22	Not used
23	Command for rapid stop
24	Output command for starting the motor
25	Output command for stopping the motor
26	Show automatic preview in the standard view
27 - 29	Not used
30	Auxiliary value 1 visible
31	Auxiliary value 2 visible

Status word allocation for `Status4` parameter

Status bit	Parameter
0	Effective `Signal1` of the message block connected via `EventTsIn`
1	Effective `Signal2` of the message block connected via `EventTsIn`
2	Effective `Signal3` of the message block connected via `EventTsIn`
3	Effective `Signal4` of the message block connected via `EventTsIn`
4	Effective `Signal5` of the message block connected via `EventTsIn`
5	Effective `Signal6` of the message block connected via `EventTsIn`
6	Effective `Signal7` of the message block connected via `EventTsIn`
7	Effective `Signal8` of the message block connected via `EventTsIn`
8	AV not interconnected
9 - 31	Not used

See also

MotL functions (Page 585)

MotL messaging (Page 591)

MotL I/Os (Page 593)

MotL block diagram (Page 600)

MotL error handling (Page 590)

MotL modes (Page 583)

5.1.2 MotL modes

MotL operating modes

The block supports all standard modes:

- Local mode (Page 36)
- "Automatic mode" (Page 32)
- "Manual mode" (Page 32)
- Out of service (Page 27)

The next section provides additional block-specific information relating to the general descriptions.

"Local mode"

You can find general information on "Local mode", switching modes and bumpless switchover in the Local mode (Page 36) section.

Motor actions you can control in local mode:

- Start (`StartLocal = 1`)
- Stop (`StopLocal = 1`)

A motor operated in local mode is controlled either by local signals or by the feedback signal (e.g the`FbkStart = 1` input parameter). Configuration takes place via the input parameter `LocalSetting`.

"Automatic mode"

You will find general information on "Automatic mode", switching modes and bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 32) section.

Motor actions you can control in auto mode:

- Start (`StartAut = 1`)
- Stop (`StopAut = 1`)

"Manual mode"

You will find general information on "Manual mode", switching modes and bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 32) section.

Motor actions you can control in manual mode:

- Start (`StartMan = 1`)
- Stop (`StopMan = 1`)

"Out of service"

You can find general information about the "Out of service" mode in the Out of service (Page 27) section.

See also

MotL block diagram (Page 600)

MotL I/Os (Page 593)

MotL messaging (Page 591)

MotL error handling (Page 590)

MotL functions (Page 585)

Description of MotL (Page 579)

5.1.3 MotL functions

Functions of MotL

The functions for this block are listed below.

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

Operator control permissions via parameter OS_Perm

The block has the following Operator permissions (Page 45) for the OS_Perm parameter:

Bit	Function
0	1 = Operator can switch to automatic mode
1	1 = Operator can switch to manual mode
2	1 = Operator can switch to local mode
3	1 = Operator can switch to out of service mode
4	1 = Operator can stop the motor
5	1 = Operator can start the motor
6	Not allocated
7	1 = Operator can reset the motor
8	1 = Operator can define the monitoring time for startup
9	1 = Operator can define the monitoring time for runtime
10	1 = Operator can activate the function monitoring time (bits 8 & 9)
11	1 = Operator can enable function simulation
12	1 = Operator can enable the maintenance release function
13	1 = Operator can change the limit (AV) for the high alarm
14	1 = Operator can change the limit (AV) for the high warning
15	1 = Operator can change the limit (AV) for the high tolerance
16	1 = Operator can change the limit (AV) for hysteresis
17	1 = Operator can change the limit (AV) for the low tolerance
18	1 = Operator can change the limit (AV) for the low warning
19	1 = Operator can change the limit (AV) for the low alarm
20 - 31	Not allocated

Limit monitoring of an additional analog value

This block provides the standard function Limit monitoring of an additional analog value (Page 73).

Limit monitoring with hysteresis

This block provides the standard function Limit monitoring with hysteresis (Page 78).

Interlocks

This block provides the following interlocks:

- Activation enable
- Interlock without reset (interlock)
- Interlock with reset (protection)

Refer to the section Interlocks (Page 100) as well as Influence of the signal status on the interlock (Page 103).

Trip function

This block provides the standard function Trip function (Page 102).

Disabling interlocks

This block provides the standard function Disabling interlocks (Page 107).

Rapid stop for motors

This block provides the standard function Rapid stop for motors (Page 102).

Resetting the block in case of interlocks or errors

This block provides the standard function Resetting the block in case of interlocks or errors (Page 119).

Time delay for changes to control

If the state of the motor is changed, this is only undertaken after the time you define in the `IdleTime` input parameter. For example, if the motor is being moved into the stop state, the command for starting can only be issued after the time saved in `IdleTime` has lapsed.

Forming the group status for interlock information

This block provides the standard function Forming the group status for interlock information (Page 105).

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `LocalLi.ST`
- `StartLocal.ST`
- `StopLocal.ST`
- `Protect.ST`
- `Inlock.ST`
- `Permit.ST`
- `Trip.ST`
- `FbkRunOut.ST`

Forcing operating states

This block provides the standard function Forcing operating states (Page 115).

The following states can be enforced:

- Start (`StartForce`)
- Stop (`StopForce`)

Feedback monitoring

This block provides the standard function Monitoring the feedbacks (Page 83).

Maintenance release

This block provides the standard function Maintenance release (Page 47).

Specify warning times for control functions

This block provides the standard function Specifying warning times for control functions at motors and valves (Page 122).

Simulating signals

This block provides the standard function Simulating signals (Page 93).

Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 53).

Safe position

This block provides the standard function Safe position for motors, valves and controllers (Page 120).

Output signal as a pulse signal or static signal

This block provides the standard function Output signal as a static signal or pulse signal (Page 86).

Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 46).

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the section Functions of blocks > Functions that can be set via the Feature I/O (Page 186). The following reactions are available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)
1	Reaction to the out of service mode (Page 199)
2	Resetting the commands for changing the mode (Page 193)
3	Enabling resetting of commands for the control settings (Page 193)
4	Setting switch or button mode (Page 195)
9	Resetting via input signals in the event of interlocks or errors (Page 194)
10	Exiting local mode (Page 199)
14	Enabling rapid stop via faceplate (Page 192)
17	Enabling bumpless changeover to automatic mode for valves, motors, and dosers (Page 197)
22	Update acknowledgment and error status of the message call (Page 193)

Displaying auxiliary values

This block provides the standard function Displaying auxiliary values (Page 52).

Time stamp

This block receives a time stamp value via the `EventTSIn` input parameter. Refer to EventTs functions (Page 997) for more information on this.

SIMATIC BATCH functionality

This block has a Batch interface. You can use SIMATIC BATCH by interconnecting the `BatchEn`, `BatchID`, `BatchName`, `StepNo` and `Occupied` I/Os with the corresponding BATCH blocks. Refer to the SIMATIC BATCH functionality (Page 122) documentation for more information.

See also

MotL messaging (Page 591)
MotL I/Os (Page 593)
MotL block diagram (Page 600)
MotL error handling (Page 590)
MotL modes (Page 583)
Time stamp (Page 51)
Error handling (Page 117)
Description of MotL (Page 579)

5.1.4 MotL error handling

MotL troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers
- Mode changeover error

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Predefined value when inserting the block, the block is not processed.
0	There is no error.
41	The value for the connection `LocalSetting` is not within the approved limit from 0 to 4.
42	`LocalSetting = 0` or `LocalSetting = 3` or `LocalSetting = 4` and `LocalLi = 1`

51	`StartLocal = 1` and `StopLocal = 1`
	`StartAut = 1` and `StopAut = 1`
	`AutModLi = 1` and `ManModLi = 1`
	`StartForce = 1` and `StopForce = 1`

52	`LocalAct = 1` and `LocalSetting = 2` or 4 and `SimOn = 1`

Mode changeover error

This error can be output by the block, see chapter Error handling (Page 117).

Invalid input signals

This error can be output by the block, see the section Error handling (Page 117).

See also

MotL block diagram (Page 600)

MotL I/Os (Page 593)

MotL messaging (Page 591)

MotL functions (Page 585)

MotL modes (Page 583)

Description of MotL (Page 579)

5.1.5 MotL messaging

Messaging

The following messages can be generated for this block:

- Control system fault
- Instance-specific messages

Control system fault

The following control system error messages can be output:

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 1	AS process control message - fault	$$BlockComment$$ Motor feedback error
	SIG 2	AS process control message - fault	$$BlockComment$$ Tripping triggered
	SIG 3	AS process control message - fault	$$BlockComment$$ External error has occurred

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter CSF. If this signal changes to CSF = 1, a control system error is triggered (MsgEvId1, SIG 3).

Instance-specific messages

You can use up to three instance-specific messages with this block.

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 4	AS process control message - fault	$$BlockComment$$ External message 1
	SIG 5	AS process control message - fault	$$BlockComment$$ External message 2
	SIG 6	AS process control message - fault	$$BlockComment$$ External message 3

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Associated values for message instance `MsgEvId1`

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`
4	`ExtVa104`
5	`ExtVa105`
6	`ExtVa106`
7	`ExtVa107`
8	`ExtVa108`
9	Reserved
10	Reserved

The associated values 4 ... 8 are allocated to the parameters `ExtVa104` ... `ExtVa108` and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

See also

Description of MotL (Page 579)

MotL functions (Page 585)

MotL I/Os (Page 593)

MotL block diagram (Page 600)

MotL error handling (Page 590)

MotL modes (Page 583)

Time stamp (Page 51)

5.1.6 MotL I/Os

MotL I/Os

Input parameters

Parameter	Description	Type	Default
AutModLi	1 = Automatic mode via interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AutModOp	1 = Automatic mode via operator (controlled by ModLiOp = 0)	BOOL	0
AV	Input additional analog value, to be connected to AV_Tech of the AV block	ANY	
AV_AH_Lim	Limit high alarm	REAL	95.0
AV_AL_Lim	Limit low alarm	REAL	5.0
AV_Hyst	Hysteresis for alarm, warning and tolerance limits	REAL	1.0
AV_TH_Lim	Limit high tolerance	REAL	85.0
AV_TL_Lim	Limit low tolerance	REAL	15.0
AV_WH_Lim	Limit high warning	REAL	90.0
AV_WL_Lim	Limit low warning	REAL	10.0
BatchEn	1 = Enable allocation	BOOL	0
BatchID	Batch ID	DWORD	16#00000000
BatchName	Batch name	STRING[32]	''
BypProt	1 = Bypassing interlock in local mode and simulation	BOOL	0
CSF	1 = External error (control system error)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
EN	1 = Called block will be processed	BOOL	1
EventTSIn	To wire the signal status of the EventTs message block; The EventTsIn input parameter is used to interconnect to the EventTsOut output parameter of the EventTs block. When this interconnection is configured, the messages of the EventTs block are displayed on the OS in the message view of the technological block and can also be acknowledged there.	STRUCT • Value: BYTE • ST: BYTE	- • 16#00 • 16#FF
ExtMsg1	Binary input for freely selectable message 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
ExtMsg2	Binary input for freely selectable message 2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg3	Binary input for freely selectable message 3	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtVa104	Associated value 4 for messages (MsgEvID1)	ANY	
ExtVa105	Associated value 5 for messages (MsgEvID1)	ANY	
ExtVa106	Associated value 6 for messages (MsgEvID1)	ANY	
ExtVa107	Associated value 7 for messages (MsgEvID1)	ANY	
ExtVa108	Associated value 8 for messages (MsgEvID1)	ANY	
FbkRun	Feedback for starting is present: 1 = Start 0 = Stop	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Feature	I/O for additional functions (Page 585)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
IdleTime	Wait time for restart in [s]	REAL	5.0
Intlock	0 = Interlocking without reset is effective; you can operate the block without reset once the interlocking condition has disappeared 1 = Good state	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#FF
Intl_En	1 = Interlock without reset (interlock, Intlock parameter) is active	BOOL	1
LocalLi	1 = Activate local mode via plant signal	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
LocalOp	1 = Local mode by operator	BOOL	0
LocalSetting	Properties for the Local mode (Page 36)	INT	0
ManModLi	1 = Manual mode via interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ManModOp	1 = Manual mode by OS operator (controlled by ModLiOp = 0)	BOOL	1

Parameter	Description	Type	Default
ModLiOp	Switchover of operating mode between: • 0 = Operator • 1 = Interconnection or SFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Monitor	1 = Feedback monitoring	BOOL	1
MonTiDynamic	Monitoring time for feedback errors after operation in [s]	REAL	3.0
MonTiStatic	Monitoring time for feedback errors without operation in [s]	REAL	3.0
MS_RelOp	1= Maintenance release via OS operator	BOOL	0
MsgEvId1	Message number (assigned automatically)	DWORD	16#FF
MsgLock	1 = Suppress process messages	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Occupied	1 = In use by a batch	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC (0-1 edge transition)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OS_Perm	I/O for operator control permissions (Page 585)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
Permit	1 = Enable for opening / closing from safe position 0 = No OS release for energizing motor	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#FF
Perm_En	1 = Activation enable (enable, Permit parameter) is active	BOOL	1
Protect	0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block 1 = Good state	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#FF
Prot_En	1 = Protective interlock (protection, Protect parameter) is active	BOOL	1
PulseWidth	Pulse width of control signal [s]	REAL	3.0
RapidStp	Rapid stop for the motor: • 0 = Motor On • 1 = Motor Off	BOOL	0
RstLi	1 = Reset via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RstOp	1 = Reset via operator	BOOL	0

Parameter	Description	Type	Default
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SimAV	Additional value used for SimOn = 1	REAL	0.0
SimOn	1 = Simulation on	BOOL	0
SelFp1	Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
StartAut	1 = Starting the motor in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StartForce	1 = Force motor start	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StartLocal	1 = Start the motor in local mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StartMan	1 = Starting the motor in manual mode	BOOL	0
StepNo	Batch step number	DWORD	16#00000000
StopAut	1 = Stop the motor in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StopForce	1 = Force motor stop	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StopLocal	1 = Stop the motor in local mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StopMan	1 = Stop the motor in manual mode	BOOL	0
Trip	1 = Motor is in "good" state	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
UserAna1	Analog auxiliary value 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
UA1unit	Unit of measure for analog auxiliary value 1	INT	0

Parameter	Description	Type	Default
UserAna2	Analog auxiliary value 2	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
UA2unit	Unit of measure for analog auxiliary value 2	INT	0
UserStatus	Freely assignable user area for status word (bit 24 to bit 31)	BYTE	16#00
WarnTiAut	Prewarning of motor start in automatic mode in [s]	REAL	3.0
WarnTiMan	Prewarning of motor start in manual mode in [s]	REAL	3.0

Output parameters

Parameter	Description	Type	Default
AutAct	1 = Automatic mode is active 0 = Manual mode is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AV_OpScale	Limit for scale in AV bar graph of faceplate	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
AV_Out	Output additional analog value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
AV_Unit	Unit of measure for additional analog value	INT	0
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see MotL error handling (Page 590)	INT	-1
FbkRunOut	Feedback for starting is present: 1 = Start 0 = Stop	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
LocalAct	1 = Local mode active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
LockAct	1 = Interlock (Inlock, Permit or Protect) is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ManAct	1 = Manual mode active	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
MonDynErr	1 = Feedback error due to control change	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MonStaErr	1 = Feedback error due to unexpected feedback change	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_Release	Maintenance release: 1 = Release for OS operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MsgAckn1	Alarm acknowledgement status 1 (output STATUS of first ALARM_8P)	WORD	16#0000

Parameter	Description	Type	Default
MsgErr1	Alarm error 1 (output ERROR of first ALARM_8P)	BOOL	0
MsgStat1	Alarm status 1 (output ERROR of first ALARM_8P)	WORD	16#0000
OosAct	1 = Block is out of service	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
P_Stop	Pulse control of motor 0 = Stop	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
P_Start	Pulse control of motor 1 = Start	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
R_StpAct	1 = Rapid stop of the motor is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Start	1 = Control of motor: started	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Status1	Status word 1 (Page 579)	DWORD	16#00000000
Status2	Status word 2 (Page 579)	DWORD	16#00000000
Status3	Status word 3	DWORD	16#00000000
Status4	Status word 4	DWORD	16#00000000
WarnAct	1 = Prewarning for motor start active (parameters WarnTiAut and WarnTiMan)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

MotL modes (Page 583)

MotL block diagram (Page 600)

MotL messaging (Page 591)

5.1.7 MotL block diagram

MotL block diagram

A block diagram is not provided for this block.

See also

MotL I/Os (Page 593)
MotL messaging (Page 591)
MotL error handling (Page 590)
MotL functions (Page 585)
MotL modes (Page 583)
Description of MotL (Page 579)

5.1.8 Operator control and monitoring

5.1.8.1 MotL views

Views of the MotL block

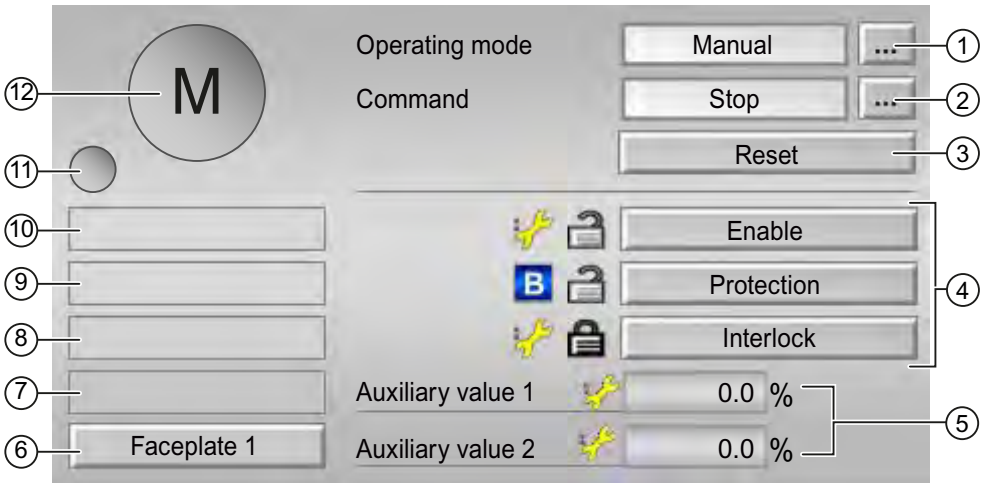
The block MotL provides the following views:

- MotL standard view (Page 601)
- Message view (Page 169)
- Limit value view of motors (Page 163)
- Trend view (Page 172)
- Parameter view for motors and valves (Page 155)
- MotL preview (Page 604)
- Memo view (Page 171)
- Batch view (Page 170)
- Block symbol for MotL (Page 607)

Refer to the sections Structure of the faceplate (Page 123) and Block icon structure (Page 174) for general information on the faceplate and block icon.

5.1.8.2 MotL standard view

MotL standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 32)
- Automatic mode (Page 32)
- Local mode (Page 36)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Starting and stopping a motor

This area shows you the default operating state for the motor. The following states can be shown and executed here:

- Start
- Stop
- Rapid stop

Refer to the Switching operating states and operating modes (Page 127) section for information on changing the state.

(3) Resetting the block

Click "Reset" for interlocks or errors. You can find additional information on this in the Resetting the block in case of interlocks or errors (Page 119) section.

(4) Operator control and display area for interlock functions of the block

You can use this button to control the interlock functions of the block. You can find additional information on this in the Interlocking functions (Page 100) section.

The following is displayed in addition to the buttons:

- Interlock status (see Forming the group status for interlock information (Page 105)), e.g.:

- Signal status (see Forming and outputting signal status for blocks (Page 88)), e.g.:

  If there is a bypass of one of the interlock signals, the symbol for the bypass is shown instead of the signal status.

- Bypass information:

  If there is a bypass, it is displayed instead of the signal status.

(5) Display of auxiliary values

You can use this area to display two auxiliary values that have been configured in the engineering system. You can find additional information on this in the Displaying auxiliary values (Page 52) section.

(6) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

(7) Display area for states of the block

This area provides additional information on the operating state of the block:

- Maintenance

Additional information on the display area for states of the block is available in section Maintenance release (Page 47).

(8) Display area for states of the block

This area provides additional information on the operating state of the block:

- Simulation

You can find additional information on this in the Simulating signals (Page 93) section.

(9) Display area for states of the block

This area provides additional information on the operating state of the block (from high to low according to priority):

- Trip
- Runtime error
- Control error
- Invalid signal
- Mode switch fail

Additional information on these errors is available in the sections Monitoring the feedbacks (Page 83) , Error handling (Page 117) (section "Invalid input signals" and "Mode changeover error") and Trip function (Page 102).

(10) Display area for states of the block

This area provides additional information on the operating state of the block:

- Force start
- Force stop

You can find additional information on this in the Forcing operating states (Page 115) section.

(11) Automatic preview

This display is only visible in manual mode or local mode when the current output signals are not identical to the control in automatic mode.

The display shows what state the motor would assume if you switched to automatic mode.

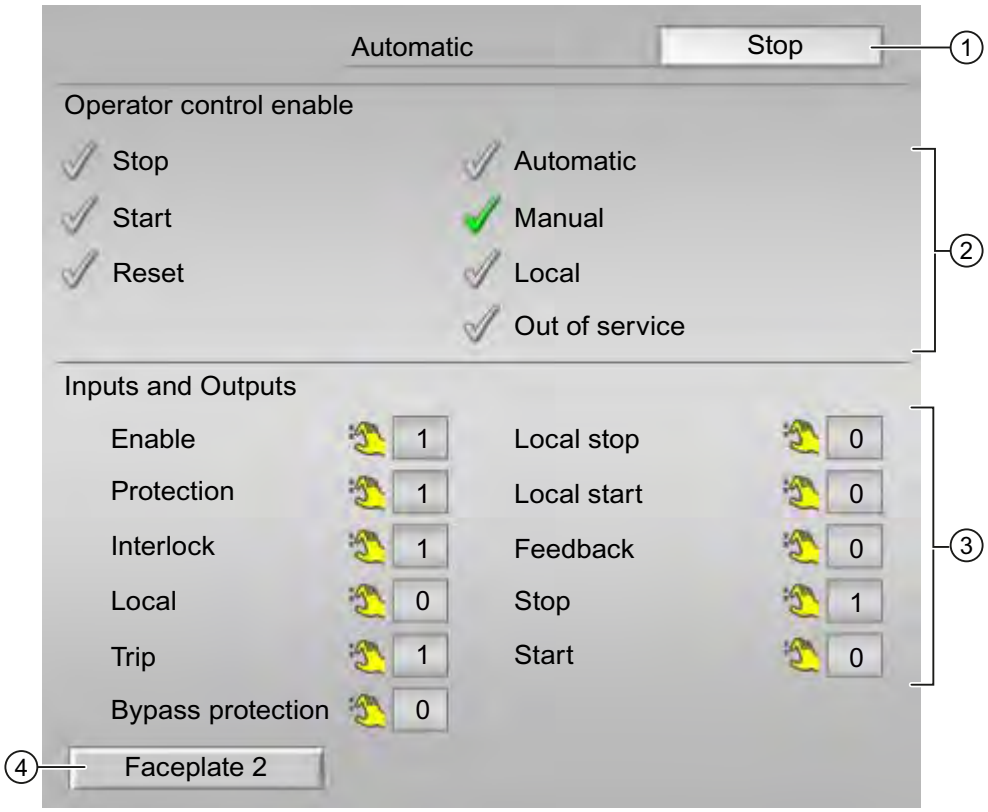
(12) Status display of motor

This area shows the operating state of the motor:

- Green: the motor is running
- Gray: Motor idle
- Red: A fault has occurred

5.1.8.3 MotL preview

MotL preview



(1) Automatic preview

This area shows you the block status after its has switched from manual to automatic mode.

If the block is in automatic mode, the current block state is displayed.

(2) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions

The following operator control enables are shown here:

- Stop: You can stop the motor.
- Start: You can start the motor.
- Reset: You can reset the motor after interlocks or errors.
- Automatic: You can switch to automatic mode.
- Manual: You can switch to manual mode
- Local: You can switch to local mode.
- Out of service: You can switch to out of service mode.

(3) Display current control signal

This area shows the most important parameters for this block with the current selection:

- Enable:
  - 0 = No OS release for energizing motor
  - 1 = Enable for starting / stopping from safe position
- Protection:
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = Good state
- Interlock:
  - 0 = Interlocking without reset is effective; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = Good state

- Local: 1= Block is controlled in local mode
- Trip: 1 = Motor is in "good" state
- Bypass protection:
  - 0 = Bridging deactivated
  - 1 = Bypassing interlock in local mode and simulation
- Local stop: 1 = Stop the motor in local mode
- Local start: 1 = Start the motor in local mode
- Feedback →: 1 = Motor has started and is running
- Stop: 1 = Motor stopped
- Start: 1 = Motor started

(4) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

5.1.8.4 Block symbol for MotL

Block icon for MotL

A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as the control system fault CSF
- Operating modes
- Signal status, maintenance release
- Displays for bridging interlocks
- Interlocks
- Memo display
- Motor state display

Icons	Selection of the block icon in CFC	Special features
	1	Block icon in the full display
	2	
	3	
	4	
	5	
	6	

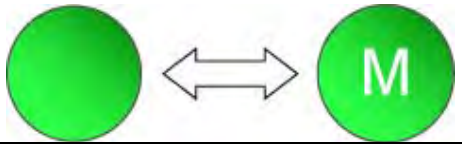
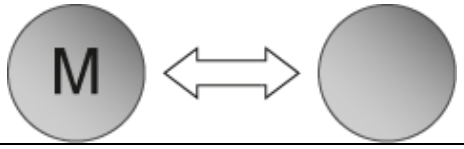
Icons	Selection of the block icon in CFC	Special features
	7	
	8	
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181).

Motor state display

The following motor states are shown here:

Icon	Meaning
	Motor started (motor icon changes)
	the motor is running
	Motor stopped (motor icon changes)
	Motor idle
	Error at motor

5.2 MotRevL - Reversible motor

5.2.1 Description of MotRevL

Object name (type + number) and family

Type + number: FB 1851

Family: Drives

Area of application for MotRevL

The block is used for the following applications:

- Control of reversible motors

How it works

The block is used to control reversible motors. Various inputs are available for controlling the motor.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

For the MotRevL block, the Advanced Process Library contains a template for process tag types as an example with an application scenario for this block.

Example of process tag types:

- Reversing motor (MotorReversible) (Page 1450)

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

Status word allocation for `Status1` parameter

For a description of the individual parameters, see the MotRevL I/Os (Page 624) section.

Status bit	Parameter
0	`Occupied`
1	`BatchEn`
2	`SimOn`
3	`OosAct.Value`
4	`OosLi.Value`
5	`0 = ManAct.Value` `1 = AutoAct.Value`
6	`LocalAct.Value`
7	`LockAct.Value`
8	`Fwd.Value`
9	Motor is stopped
10	`Rev.Value`
11	`MonStaErr.Value`
12	`MonDynErr.Value`
13	`BypProt`
14	Invalid signal status
15	Mode changeover error
16	`1 = Intlock` is active
17	`1 = Permit` is active
18	`1 = Protect` is active
19	`Trip`
20	`FwdForce`
21	`RevForce`
22	`StopForce`
23	"Interlock" button is enabled
24 - 25	Not used
26	Bypass information from previous function block
27	Automatic preview for forward mode
28	Automatic preview for stopping
29	Automatic preview for reverse mode
30	"Permission" button is enabled
31	"Protection" button is enabled

Status word allocation for `Status2` parameter

Status bit	Parameter
0	`MsgLock`
1	`AV_AH_Act.Value`
2	`AV_WH_Act.Value`
3	`AV_TH_Act.Value`
4	`AV_TL_Act.Value`
5	`AV_WL_Act.Value`
6	`AV_AL_Act.Value`
7	`AV_AH_En`
8	`AV_WH_En`
9	`AV_TH_En`
10	`AV_TL_En`
11	`AV_WL_En`
12	`AV_AL_En`
13	`AV_AH_MsgEn`
14	`AV_WH_MsgEn`
15	`AV_TH_MsgEn`
16	`AV_TL_MsgEn`
17	`AV_WL_MsgEn`
18	`AV_AL_MsgEn`
19	1 = No impact of input signals on local mode with `LocalSetting = 2` and `LocalSetting = 4`
20	Motor is stopped
21	Motor in forward mode is stopped
22	Motor in reverse mode is stopped
23	Motor in forward mode is started
24	Motor runs in forward mode
25	Motor in reverse mode is started
26	Motor runs in reverse mode
27	Error when stopping motor
28	Error in forward mode of motor
29	Error in reverse mode of motor
30	Display for interlocks in block icon
31	`MS_RelOp`

Status word allocation for `Status3` parameter

Status bit	Parameter
0 - 18	Not used
19	1 = Enabled for emergency stop push button (`Feature` bit Enabling rapid stop via faceplate (Page 192))
20 - 22	Not used
23	Command for rapid stop
24	Command for starting → the motor
25	Command for starting ← the motor
26	Show automatic preview in the standard view
27 - 29	Not used
30	Auxiliary value 1 visible
31	Auxiliary value 2 visible

Status word allocation for `Status4` parameter

Status bit	Parameter
0	Effective `Signal1` of the message block connected via `EventTsIn`
1	Effective `Signal2` of the message block connected via `EventTsIn`
2	Effective `Signal3` of the message block connected via `EventTsIn`
3	Effective `Signal4` of the message block connected via `EventTsIn`
4	Effective `Signal5` of the message block connected via `EventTsIn`
5	Effective `Signal6` of the message block connected via `EventTsIn`
6	Effective `Signal7` of the message block connected via `EventTsIn`
7	Effective `Signal8` of the message block connected via `EventTsIn`
8	AV not interconnected
9 - 31	Not used

See also

MotRevL functions (Page 615)
MotRevL messaging (Page 622)
MotRevL block diagram (Page 632)
MotRevL error handling (Page 620)
MotRevL modes (Page 613)

5.2.2 MotRevL modes

MotRevL operating modes

The block can be operated using the following modes:

- Local mode (Page 36)
- Automatic mode (Page 32)
- Manual mode (Page 32)
- Out of service (Page 27)

The next section provides additional block-specific information relating to the general descriptions.

"Local mode"

You can find general information on "Local mode", switching modes and bumpless switchover in the Local mode (Page 36) section.

Motor actions you can control in local mode:

- Start the motor in forward (`FwdLocal = 1`)
- Start the motor in reverse (`RevLocal = 1`)
- Stop (`StopLocal = 1`)

A motor operated in local mode is controlled either by local signals or by the feedback signal (input parameters `FbkFwd = 1` and `FbkRev = 1`). Configuration takes place via the input parameter `LocalSetting`.

"Automatic mode"

You can find general information on "Automatic mode", switching modes and bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 32) section.

Motor actions you can control in auto mode:

- Start the motor in forward (`FwdAut = 1`)
- Start the motor in reverse (`RevAut = 1`)
- Stop (`StopAut = 1`)

"Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 32) section.

Motor actions you can control in manual mode:

- Start the motor in forward (`FwdMan = 1`)
- Start the motor in reverse (`RevMan = 1`)
- Stop (`StopMan = 1`)

"Out of service"

You can find general information about the "Out of service" mode in the Out of service (Page 27) section.

See also

MotRevL block diagram (Page 632)

MotRevL I/Os (Page 624)

MotRevL messaging (Page 622)

MotRevL error handling (Page 620)

MotRevL functions (Page 615)

Description of MotRevL (Page 609)

5.2.3 MotRevL functions

Functions of MotRevL

The functions for this block are listed below.

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

Operator control permissions via parameter OS_Perm

The block has the following Operator permissions (Page 45) for the OS_Perm parameter:

Bit	Function
0	1 = Operator can switch to automatic mode
1	1 = Operator can switch to manual mode
2	1 = Operator can switch to local mode
3	1 = Operator can switch to out of service mode
4	1 = Operator can stop the motor
5	1 = Operator can start the motor in forward
6	1 = Operator can start the motor in reverse
7	1 = Operator can reset the motor
8	1 = Operator can define the monitoring time for startup
9	1 = Operator can define the monitoring time for runtime
10	1 = Operator can activate the function monitoring time (bits 8 & 9)
11	1 = Operator can enable function simulation
12	1 = Operator can enable the maintenance release function
13	1 = Operator can change the limit (AV) for the high alarm
14	1 = Operator can change the limit (AV) for the high warning
15	1 = Operator can change the limit (AV) for the high tolerance
16	1 = Operator can change the limit (AV) for hysteresis
17	1 = Operator can change the limit (AV) for the low tolerance
18	1 = Operator can change the limit (AV) for the low warning
19	1 = Operator can change the limit (AV) for the low alarm
20 - 31	Not allocated

Time delay after changing direction of rotation or restart of motor

Use the input parameter IdleTime to enter a time delay for changing the direction of rotation or restarting the motor. Use the Feature bit Enabling direct changeover between forward and reverse (Page 189) to define how the change is to take place.

Please note that IdleTime ≥ MonTiDynmic has to be configured.

Limit monitoring of an additional analog value

This block provides the standard function Limit monitoring of an additional analog value (Page 73).

Limit monitoring with hysteresis

This block provides the standard function Limit monitoring with hysteresis (Page 78). It is performed via the input parameter `AV_Hyst`.

Interlocks

This block provides the following interlocks:

- Activation enable
- Interlock without reset (interlock)
- Interlock with reset (protection)

Refer to the section Interlocks (Page 100) as well as Influence of the signal status on the interlock (Page 103).

Trip function

This block provides the standard function Trip function (Page 102).

Rapid stop for motors

This block provides the standard function Rapid stop for motors (Page 102).

Disabling interlocks

This block provides the standard function Disabling interlocks (Page 107).

Resetting the block in case of interlocks or errors

This block provides the standard function Resetting the block in case of interlocks or errors (Page 119).

Forming the group status for interlocks

This block provides the standard function Forming the group status for interlock information (Page 105).

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `FbkFwdOut.ST`
- `FbkRevOut.ST`
- `LocalLi.ST`
- `FwdLocal.ST`
- `StopLocal.ST`
- `RevLocal.ST`
- `Protect.ST`
- `Intlock.ST`
- `Permit.ST`
- `Trip.ST`

Forcing operating states

This block provides the standard function Forcing operating states (Page 115).

The following states can be enforced:

- Start forwards (`FwdForce`)
- Start in reverse (`RevForce`)
- Stop (`StopForce`)

Feedback monitoring

This block provides the standard function Monitoring the feedbacks (Page 83).

Maintenance release

This block provides the standard function Maintenance release (Page 47).

Specify warning times for control functions

This block provides the standard function Specifying warning times for control functions at motors and valves (Page 122).

Simulating signals

This block provides the standard function Simulating signals (Page 93).

Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 53).

Safe position

This block provides the standard function Safe position for motors, valves and controllers (Page 120).

Output signal as a pulse signal or static signal

This block provides the standard function Output signal as a static signal or pulse signal (Page 86).

Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 46).

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)
1	Reaction to the out of service mode (Page 199)
2	Resetting the commands for changing the mode (Page 193)
3	Enabling resetting of commands for the control settings (Page 193)
4	Setting switch or button mode (Page 195)
7	Enabling direct changeover between forward and reverse (Page 189)
9	Resetting via input signals in the event of interlocks or errors (Page 194)
10	Exiting local mode (Page 199)
14	Enabling rapid stop via faceplate (Page 192)
17	Enabling bumpless changeover to automatic mode for valves, motors, and dosers (Page 197)
22	Update acknowledgment and error status of the message call (Page 193)

Displaying auxiliary values

This block provides the standard function Displaying auxiliary values (Page 52).

Time stamp

This block receives a time stamp value via the `EventTSIn` input parameter. Refer to EventTs functions (Page 997) for more information on this.

SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 122).

See also

Description of MotRevL (Page 609)
MotRevL messaging (Page 622)
MotRevL I/Os (Page 624)
MotRevL block diagram (Page 632)
MotRevL error handling (Page 620)
MotRevL modes (Page 613)
Time stamp (Page 51)

5.2.4 MotRevL error handling

MotRevL troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers
- Mode changeover error
- Invalid input signals

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Default value when implementing the block; block will not be processed.
0	There is no error.
41	The value for the connection `LocalSetting` is not within the approved limit from 0 to 4.
42	`LocalSetting = 0` or `LocalSetting = 3` or `LocalSetting = 4` and `LocalLi = 1`

51	`FwdLocal = 1` and `StopLocal = 1`
	`RevLocal = 1` and `StopLocal = 1`
	`FwdLocal = 1` and `RevLocal = 1`
	`FwdAut = 1` and `StopAut = 1`
	`RevAut = 1` and `StopAut = 1`
	`FwdAut = 1` and `RevAut = 1`
	`AutModLi = 1` and `ManModLi = 1`
	`FwdForce = 1` and `StopForce = 1`
	`RevForce = 1` and `StopForce = 1`
	`FwdForce = 1` and `RevForce = 1`

52	`LocalAct = 1` and `LocalSetting = 2` or 4 and `SimOn = 1`

Mode changeover error

This error can be output by the block, see chapter Error handling (Page 117).

Invalid input signals

This error can be output by the block, see the section Error handling (Page 117).

See also

MotRevL block diagram (Page 632)
MotRevL I/Os (Page 624)
MotRevL messaging (Page 622)
Description of MotRevL (Page 609)
MotRevL modes (Page 613)
MotRevL functions (Page 615)

5.2.5 MotRevL messaging

Messaging

The following messages can be generated for this block:

- Control system fault
- Instance-specific messages

Control system fault

The following control system error messages can be output:

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 1	AS process control message - fault	$$BlockComment$$ Motor feedback error
	SIG 2	AS process control message - fault	$$BlockComment$$ Tripping triggered
	SIG 3	AS process control message - fault	$$BlockComment$$ External error has occurred

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter CSF. If this signal changes to CSF = 1, a control system error is triggered (MsgEvId1, SIG 3).

Instance-specific messages

You can use up to three instance-specific messages with this block.

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 4	AS process control message - fault	$$BlockComment$$ External message 1
	SIG 5	AS process control message - fault	$$BlockComment$$ External message 2
	SIG 6	AS process control message - fault	$$BlockComment$$ External message 3

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Associated values for message instance `MsgEvId1`

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`
4	`ExtVa104`
5	`ExtVa105`
6	`ExtVa106`
7	`ExtVa107`
8	`ExtVa108`
9	Reserved
10	Reserved

The associated values 4 to 8 are allocated to the parameters `ExtVa104` ... `ExtVa108` and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

See also

Description of MotRevL (Page 609)

MotRevL functions (Page 615)

MotRevL I/Os (Page 624)

MotRevL block diagram (Page 632)

MotRevL error handling (Page 620)

MotRevL modes (Page 613)

Time stamp (Page 51)

5.2.6 MotRevL I/Os

MotRevL I/Os

Input parameters

Parameter	Description	Type	Default
AutModLi	1 = Automatic mode via interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AutModOp	1 = Automatic mode via operator (controlled by ModLiOp = 0)	BOOL	0
AV	Input additional analog value, to be connected to AV_Out of the AV block	ANY	
AV_AH_Lim	Limit high alarm	REAL	95.0
AV_AL_Lim	Limit low alarm	REAL	5.0
AV_Hyst	Hysteresis for alarm, warning and tolerance limits	REAL	1.0
AV_TH_Lim	Limit high tolerance	REAL	85.0
AV_TL_Lim	Limit low tolerance	REAL	15.0
AV_WH_Lim	Limit high warning	REAL	90.0
AV_WL_Lim	Limit low warning	REAL	10.0
BatchEn	1 = Enable allocation	BOOL	0
BatchID	Batch ID	DWORD	16#00000000
BatchName	Batch name	STRING[32]	''
BypProt	1 = Bypassing interlock in local mode and simulation	BOOL	0
CSF	1 = External error (control system error)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
EN	1 = Called block will be processed	BOOL	1
EventTSIn	To wire the signal status of the EventTs message block; The EventTsIn input parameter is used to interconnect to the EventTsOut output parameter of the EventTs block. When this interconnection is configured, the messages of the EventTs block are displayed on the OS in the message view of the technological block and can also be acknowledged there.	STRUCT • Value: BYTE • ST: BYTE	- • 16#00 • 16#FF
ExtMsg1	Binary input for freely selectable message 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
ExtMsg2	Binary input for freely selectable message 2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg3	Binary input for freely selectable message 3	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtVa104	Associated value 4 for messages (MsgEvID1)	ANY	
ExtVa105	Associated value 5 for messages (MsgEvID1)	ANY	
ExtVa106	Associated value 6 for messages (MsgEvID1)	ANY	
ExtVa107	Associated value 7 for messages (MsgEvID1)	ANY	
ExtVa108	Associated value 8 for messages (MsgEvID1)	ANY	
FbkFwd	1 = Feedback for forward mode is present	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FbkRev	1 = Feedback for reverse mode is present	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Feature	I/O for additional functions (Page 615)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
FwdAut	1 = Activating forward mode of motor in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FwdForce	1 = Forcing activation of motor forward mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FwdLocal	1 = Activating forward mode of motor in local mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FwdMan	1 = Activating forward mode of motor in manual mode	BOOL	0
IdleTime	Wait time for change of direction or restart in [s]	REAL	5.0
Intlock	1 = Interlocking without reset is effective; you can operate the block without reset once the interlocking condition has disappeared 0 = Good state	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#FF

Parameter	Description	Type	Default
Intl_En	1 = Interlock without reset (interlock, Intlock parameter) is active	BOOL	1
LocalLi	1 = Activate local mode via plant signal	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
LocalOp	1 = Local mode by operator	BOOL	0
LocalSetting	Characteristics of Local mode (Page 36)	INT	0
ManModLi	1 = Manual mode via interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ManModOp	1 = Manual mode by OS operator (controlled by ModLiOp = 0)	BOOL	1
ModLiOp	Switchover of operating mode between: • 0 = Operator • 1 = Interconnection or SFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Monitor	1 = Feedback monitoring	BOOL	1
MonTiDynamic	Monitoring time for feedback errors after operation in [s]	REAL	3.0
MonTiStatic	Monitoring time for feedback errors without operation in [s]	REAL	3.0
MS_RelOp	1= Maintenance release via OS operator	BOOL	0
MsgEvId1	Message number (assigned automatically)	DWORD	16#FF
MsgLock	1 = Suppress process messages	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_RelOp	1 = Maintenance release via OS operator	BOOL	0
Occupied	1 = In use by a batch	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC (0-1 edge transition)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OS_Perm	I/O for operator control permissions (Page 615)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
Permit	1 = Enable for opening / closing from safe position 0 = No OS release for energizing motor	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#FF

Parameter	Description	Type	Default
Perm_En	1 = Activation enable (enable, Permit parameter) is active	BOOL	1
Protect	0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block 1 = Good state	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#FF
Prot_En	1 = Protective interlock (protection, Protect parameter) is active	BOOL	1
PulseWidth	Pulse width of control signal [s]	REAL	3.0
RapidStp	Rapid stop for the motor: • 0 = Motor On • 1 = Motor Off	BOOL	0
RevAut	1 = Activation of reverse motor operation in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RevForce	1 = Force activation of reverse motor operation	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RevLocal	1 = Activation of reverse motor operation in local mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RevMan	1 = Activation of reverse motor operation in manual mode	BOOL	0
RstLi	1 = Reset via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RstOp	1 = Reset via operator	BOOL	0
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SimAV	Additional value used for SimOn = 1	REAL	0.0
SimOn	1 = Simulation on	BOOL	0
SelFp1	Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
StepNo	Batch step number	DWORD	16#00000000
StopAut	1 = Stop the motor in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
StopForce	1 = Force motor stop	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StopLocal	1 = Stop the motor in local mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StopMan	1 = Stop the motor in manual mode	BOOL	0
Trip	1 = Motor is in "good" state	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
UserAna1	Analog auxiliary value 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
UA1unit	Unit of measure for analog auxiliary value 1	INT	0
UserAna2	Analog auxiliary value 2	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
UA2unit	Unit of measure for analog auxiliary value 2	INT	0
UserStatus	Freely assignable user area for status word (bit 24 to bit 31)	BYTE	16#00
WarnTiAut	Prewarning of motor start in automatic mode in [s]	REAL	3.0
WarnTiMan	Prewarning of motor start in manual mode in [s]	REAL	3.0

Output parameters

Parameter	Description	Type	Default
AutAct	1 = Automatic mode is active 0 = Manual mode is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AV_OpScale	Limit for scale in AV bar graph of faceplate	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
AV_Out	Output additional analog value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
AV_Unit	Unit of measure for additional analog value	INT	0
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see MotRevL error handling (Page 620)	INT	-1
F_Stop	1 = Rapid stop of the motor is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FbkFwdOut	Feedback: 1 = Forward operation active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FbkRevOut	Feedback: 1 = Reverse operation active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Fwd	1 = Control of motor forward	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
LocalAct	1 = Local mode active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
LockAct	1 = Interlock (Intlock, Permit or Protect) is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ManAct	1 = Manual mode active	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
MonDynErr	1 = Feedback error due to control change	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
MonStaErr	1 = Feedback error due to unexpected feedback change	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_Release	Maintenance release: 1 = Release for OS operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MsgAckn1	Alarm acknowledgement status 1 (output STATUS of first ALARM_8P)	WORD	16#0000
MsgErr1	Alarm error 1 (output ERROR of first ALARM_8P)	BOOL	0
MsgStat1	Alarm status 1 (output ERROR of first ALARM_8P)	WORD	16#0000
OosAct	1 = Block is out of service	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
P_Fwd	Pulse control of motor: 1 = Forward	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
P_Rev	Pulse control of motor: 1 = Reverse	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
P_Stop	Pulse control of motor : Stopping	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
R_StpAct	1 = Rapid stop of the motor is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Rev	1 = Control of motor: Reverse	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Status1	Status word 1 (Page 609)	DWORD	16#00000000
Status2	Status word 2 (Page 609)	DWORD	16#00000000
Status3	Status word 3 (Page 609)	DWORD	16#00000000
Status4	Status word 4 (Page 609)	DWORD	16#00000000
WarnAct	1 = Prewarning for motor start active (parameters WarnTiAut and WarnTiMan)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

MotRevL messaging (Page 622)
MotRevL block diagram (Page 632)
MotRevL modes (Page 613)

5.2.7 MotRevL block diagram

MotRevL block diagram

This block does not come with a block diagram.

See also

MotRevL I/Os (Page 624)
MotRevL messaging (Page 622)
MotRevL error handling (Page 620)
MotRevL functions (Page 615)
MotRevL modes (Page 613)
Description of MotRevL (Page 609)

5.2.8 Operator control and monitoring

5.2.8.1 MotRevL views

Views of the MotRevL block

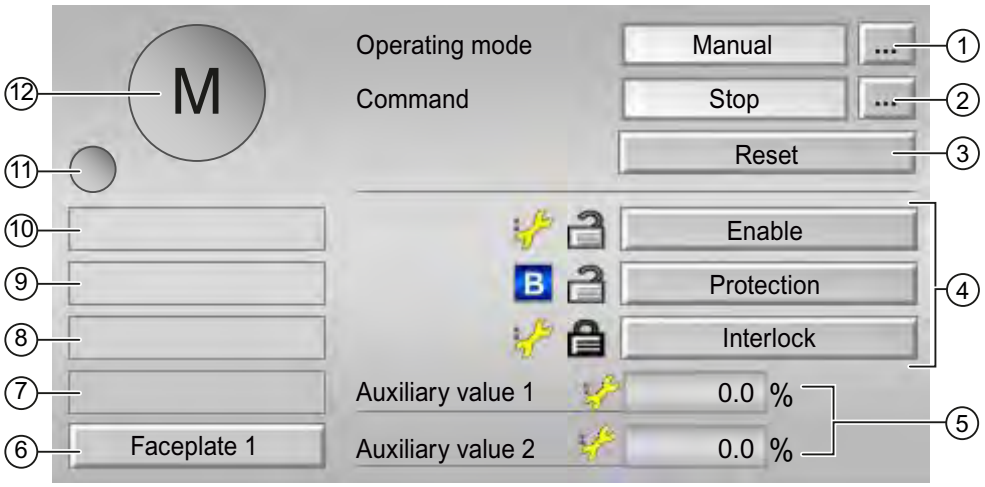
The block MotRevL provides the following views:

- MotRevL standard view (Page 633)
- Message view (Page 169)
- Limit value view of motors (Page 163)
- Trend view (Page 172)
- Parameter view for motors and valves (Page 155)
- MotRevL preview (Page 636)
- Memo view (Page 171)
- Batch view (Page 170)
- Block icon for MotRevL (Page 639)

Refer to the sections Structure of the faceplate (Page 123) and Block icon structure (Page 174) for general information on the faceplate and block icon.

5.2.8.2 MotRevL standard view

MotRevL standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 32)
- Automatic mode (Page 32)
- Local mode (Page 36)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Starting and stopping a motor

This area shows you the default operating state for the motor. The following states can be shown and executed here:

- Start →
- Start ←
- Stop
- Rapid stop

Refer to the Switching operating states and operating modes (Page 127) section for information on changing the state.

(3) Resetting the block

Click "Reset" for interlocks or errors. You can find additional information on this in the Resetting the block in case of interlocks or errors (Page 119) section.

(4) Display area for interlock functions of the block

You can use this button to control the interlock functions of the block. You can find additional information on this in the Interlocking functions (Page 100) section.

The following is displayed in addition to the buttons:

- Interlock status (see Forming the group status for interlock information (Page 105)), e.g.:

- Signal status (see Forming and outputting signal status for blocks (Page 88)), e.g.:

  If there is a bypass of one of the interlock signals, the symbol for the bypass is shown instead of the signal status.

- Bypass information:

  If there is a bypass, it is displayed instead of the signal status.

(5) Display of auxiliary values

You can use this area to display two auxiliary values that have been configured in the engineering system. You can find additional information on this in the Displaying auxiliary values (Page 52) section.

(6) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

(7) Display area for states of the block

This area provides additional information on the operating state of the block:

- Maintenance

Additional information on the display area for states of the block is available in section Maintenance release (Page 47).

(8) Display area for states of the block

This area provides additional information on the operating state of the block:

- Simulation

You can find additional information on this in the Simulating signals (Page 93) section.

(9) Display area for states of the block

This area provides additional information on the operating state of the block:

- Trip
- Runtime error
- Control error
- Invalid signal
- Mode switch fail

Additional information on these errors is available in the sections Monitoring the feedbacks (Page 83) , Error handling (Page 117) (section "Invalid input signals" and "Mode changeover error") and Trip function (Page 102).

(10) Display area for states of the block

This area provides additional information on the operating state of the block:

- Force stop
- Force start →
- Force start ←

You can find additional information on this in the Forcing operating states (Page 115) section.

(11) Automatic preview

This display is only visible in manual mode or local mode when the current output signals are not identical to the control in automatic mode.

The display shows what state the motor would assume if you switched to automatic mode.

The following icons can be displayed:

(12) Status display of motor

This area shows if the motor is running or idle:

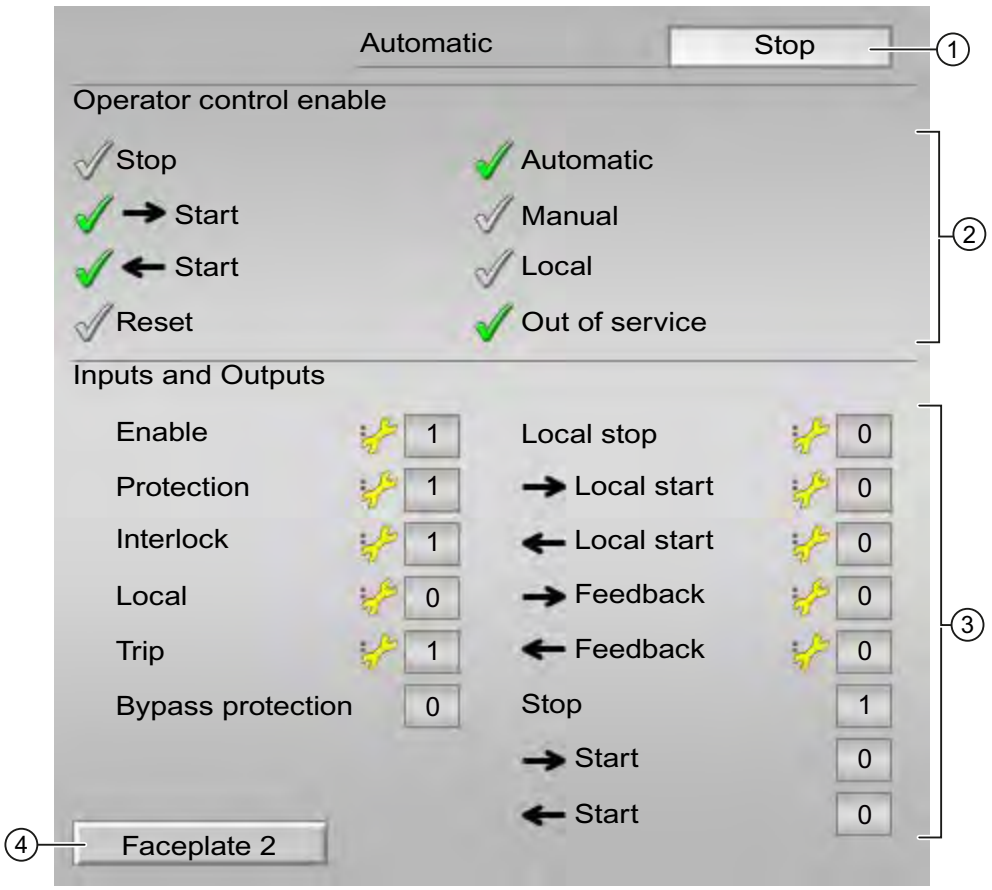
- Green: the motor is running
- Gray: Motor is idle (as shown in the figure above)
- Red: A fault has occurred

See also

Manual and Automatic mode for control blocks (Page 29)

5.2.8.3 MotRevL preview

MotRevL preview



(1) Automatic preview

This area shows you the block status after its has switched from manual to automatic mode.

If the block is in automatic mode, the current block state is displayed.

(2) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions

The following operator control enables are shown here:

- Stop: You can stop the motor.
- Start → : You can start the motor.
- Start ← : You can start the motor.
- Reset: You can reset the motor after interlocks or errors.
- Automatic: You can switch to automatic mode.
- Manual: You can switch to manual mode
- Local: You can switch to local mode.
- Out of service: You can switch to out of service mode.

(3) Display current control signal

This area shows the most important parameters for this block with the current selection:

- Enable:
  - 0 = No OS release for energizing motor
  - 1 = Enable for starting / stopping from safe position
- Protection:
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = Good state
- Interlock:
  - 0 = Interlocking without reset is effective; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = Good state
- Local: 1= Block is controlled in local mode
- Trip: 1 = Motor is in "good" state
- Bypass protection:
  - 0 = Bridging deactivated
  - 1 = Bypassing interlock in local mode and simulation

- Local stop: 1 = Stop the motor in local mode
- Local start →: 1 = Start the motor in local mode
- Local start ←: 1 = Start the motor in local mode
- Feedback →: 1 = Motor has started and is running
- Feedback ←: 1 = Motor has started and is running
- Stop: 1 = Motor stopped
- Start → : 1 = Motor started
- Start ← : 1 = Motor started

(4) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

5.2.8.4 Block icon for MotRevL

Block icons for MotRevL

A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as the control system fault CSF
- Operating modes
- Signal status, maintenance release
- Displays for bridging interlocks
- Interlocks
- Memo display
- Motor state display

Icons	Selection of the block icon in CFC	Special features
	1	Block icon in the full display
	2	
	3	
	4	
	5	
	6	
	7	

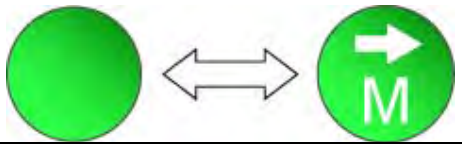
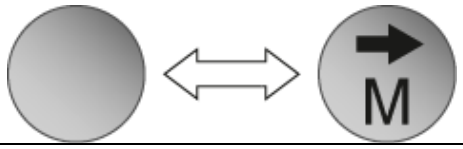
Icons	Selection of the block icon in CFC	Special features
	8	
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181).

Motor state display

The following motor states are shown here:

Icon	Meaning
	Motor started (motor icon changes)
	the motor is running
	Motor stopped (motor icon changes)
	Motor idle
	Error at motor

5.3 MotSpdCL - Controllable motor with two directions of rotation

5.3.1 MotSpdCL description

Object name (type + number) and family

Type + number: FB 1854

Family: Drives

Area of application for MotSpdCL

The block is used for the following applications:

- Control of motors with two directions of rotation and different speeds

How it works

The block is used to control motors with two directions of rotation for different speeds. Various inputs are available for controlling the motor.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

For the MotSpdCL block, the Advanced Process Library contains a template for process tag types as an example with an application scenario for this block.

Example of process tag types:

- Reversing motor with controllable speed (MotorSpeedControlled) (Page 1451)

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

Note

At a warm restart with the `Feature` bit set to 0, the block is switched to manual mode and the setpoint is set internal and 0.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

Status word allocation for `Status1` parameter

For a description of the individual parameters, see the MotSpdCL I/Os (Page 658) section.

Status bit	Parameter
0	`Occupied`
1	`BatchEn`
2	`SimOn`
3	`OosAct.Value`
4	`OosLi.Value`
5	`0 = ManAct.Value` `1 = AutoAct.Value`
6	`LocalAct.Value`
7	`LockAct.Value`
8	`Fwd.Value`
9	Emergency stop push button activated
10	`Rev.Value`
11	`MonStaErr.Value`
12	`MonDynErr.Value`
13	`BypProt`
14	Invalid signal status
15	Mode changeover error
16	`1 = Intlock` is active
17	`1 = Permit` is active
18	`1 = Protect` is active
19	`Trip`
20	`FwdForce`
21	`RevForce`
22	`StopForce`
23	"Interlock" button is enabled
24 - 25	Not used
26	Bypass information from previous function block
27	Automatic preview for forward mode
28	Automatic preview for stopping
29	Automatic preview for reverse mode
30	"Permission" button is enabled
31	"Protection" button is enabled

Status word allocation for `Status2` parameter

Status bit	Parameter
0	`MsgLock`
1	`AV_AH_Act.Value`
2	`AV_WH_Act.Value`
3	`AV_TH_Act.Value`
4	`AV_TL_Act.Value`
5	`AV_WL_Act.Value`
6	`AV_AL_Act.Value`
7	`AV_AH_En`
8	`AV_WH_En`
9	`AV_TH_En`
10	`AV_TL_En`
11	`AV_WL_En`
12	`AV_AL_En`
13	`AV_AH_MsgEn`
14	`AV_WH_MsgEn`
15	`AV_TH_MsgEn`
16	`AV_TL_MsgEn`
17	`AV_WL_MsgEn`
18	`AV_AL_MsgEn`
19	1 = No impact of input signals on local mode with `LocalSetting = 2` and `LocalSetting = 4`
20	Motor is stopped
21	Motor in forward mode is stopped
22	Motor in reverse mode is stopped
23	Motor in forward mode is started
24	Motor runs in forward mode
25	Motor in reverse mode is started
26	Motor runs in reverse mode
27	Error when stopping motor
28	Error in forward mode of motor
29	Error in reverse mode of motor
30	`SP_ExtAct.Value`
31	Display for interlocks in block icon

Status word allocation for `Status3` parameter

Status bit	Parameter
0	Not used
1	`RbkWH_Act.Value`
2 - 3	Not used
4	`RbkWL_Act.Value`
5 - 6	Not used
7	`RbkWH_En`
8 - 9	Not used
10	`RbkWL_En`
11 - 13	Not used
14	`RbkWH_MsgEn`
15 - 16	Not used
17	`RbkWL_MsgEn`
18	Motor is stopped
19	1 = Enabled for emergency stop push button (`Feature` bit Enabling rapid stop via faceplate (Page 192))
20	`SP_RmpModTime`
21	`SP_RmpOn`
22	Not used
23	Command for rapid stop of the motor
24	Command for starting → the motor
25	Command for starting ← the motor
26	Show automatic preview in the standard view
27 - 28	Not used
29	`MS_RelOp`
30	Auxiliary value 1 visible
31	Auxiliary value 2 visible

Status word allocation for `Status4` parameter

Status bit	Parameter
0	Effective `Signal1` of the message block connected via `EventTsIn`
1	Effective `Signal2` of the message block connected via `EventTsIn`
2	Effective `Signal3` of the message block connected via `EventTsIn`
3	Effective `Signal4` of the message block connected via `EventTsIn`
4	Effective `Signal5` of the message block connected via `EventTsIn`
5	Effective `Signal6` of the message block connected via `EventTsIn`
6	Effective `Signal7` of the message block connected via `EventTsIn`
7	Effective `Signal8` of the message block connected via `EventTsIn`
8	AV not interconnected
9 - 31	Not used

See also

MotSpdCL block diagram (Page 667)
MotSpdCL messaging (Page 656)
MotSpdCL error handling (Page 654)
MotSpdCL functions (Page 648)
MotSpdCL modes (Page 646)

5.3.2 MotSpdCL modes

MotSpdCL operating modes

The block can be operated using the following modes:

- Local mode (Page 36)
- Automatic mode (Page 32)
- Manual mode (Page 32)
- Out of service (Page 27)

The next section provides additional block-specific information relating to the general descriptions.

"Local mode"

You can find general information on "Local mode", switching modes and bumpless switchover in the Local mode (Page 36) section.

Motor actions you can control in local mode:

- Start the motor in forward (`FwdLocal = 1`)
- Start the motor in reverse (`RevLocal = 1`)
- Stop (`StopLocal = 1`)

A motor operated in local mode is controlled either by local signals or by the feedback signal (input parameters `FbkFwd = 1` and `FbkRev = 1`). Configuration takes place via the input parameter `LocalSetting`.

"Automatic mode"

You can find general information on "Automatic mode", switching modes and bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 32) section.

Motor actions you can control in auto mode:

- Start the motor in forward (`FwdAut = 1`)
- Start the motor in reverse (`RevAut = 1`)
- Stop (`StopAut = 1`)

"Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 32) section.

Motor actions you can control in manual mode:

- Start the motor in forward (`FwdMan = 1`)
- Start the motor in reverse (`RevMan = 1`)
- Stop (`StopMan = 1`)

"Out of service"

You can find general information about the "Out of service" mode in theOut of service (Page 27) section.

See also

MotSpdCL block diagram (Page 667)

MotSpdCL I/Os (Page 658)

MotSpdCL messaging (Page 656)

MotSpdCL error handling (Page 654)

MotSpdCL functions (Page 648)

MotSpdCL description (Page 641)

5.3.3 MotSpdCL functions

Functions of MotSpdCL

The functions for this block are listed below.

Alarm delays with two time values per limit pair

This block has the standard function alarm delay for Two time values per limit pair (Page 80) limit monitoring of the feedback.

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

Control permissions via parameters `OS_Perm` and `OS1Perm`

The block has the following Operator permissions (Page 45) for the `OS_Perm` parameter:

Bit	Function
0	1 = Operator can switch to automatic mode
1	1 = Operator can switch to manual mode
2	1 = Operator can switch to local mode
3	1 = Operator can switch to out of service mode
4	1 = Operator can stop the motor
5	1 = Operator can start the motor in forward
6	1 = Operator can start the motor in reverse
7	1 = Operator can reset the motor
8	1 = Operator can define the monitoring time for startup
9	1 = Operator can define the monitoring time for runtime
10	1 = Operator can activate the function monitoring time (bits 8 & 9)
11	1 = Operator can enable function simulation
12	1 = Operator can enable the maintenance release function
13	1 = Operator can change the limit (AV) for the high alarm
14	1 = Operator can change the limit (AV) for the high warning
15	1 = Operator can change the limit (AV) for the high tolerance
16	1 = Operator can change the limit (AV) for hysteresis
17	1 = Operator can change the limit (AV) for the low alarm
18	1 = Operator can change the limit (AV) for the low warning
19	1 = Operator can change the limit (AV) for the low tolerance
20	1 = Operator can activate the bumpless changeover from external to internal function `SP_TrkExt`
21	1 = Operator can change the internal setpoint `SP_Int`
22	1 = Operator can switch the setpoint to external `SP_ExtOp`
23	1 = Operator can switch the setpoint to internal `SP_IntOp`

Bit	Function
24	1 = Operator can use the setpoint's gradient limitation function SP_RateOn
25	1 = Operator can change the setpoint's high limit for the ramp SP_UpRaLim
26	1 = Operator can change the setpoint's low limit for the ramp SP_DnRaLim
27	1 = Operator can activate the setpoint ramp function SP_RmpOn
28	1 = Operator can change between the time value or the value for the ramp SP_RmpModTime
29	1 = Operator can change the ramp time SP_RmpTime
30	1 = Operator can change the target setpoint SP_RmpTarget for the setpoint ramp
31	Not allocated

The block has the following operator control permissions for the OS1Perm parameter:

Bit	Function
0	Not allocated
1	1 = Operator can change the limit (Rbk) for high warning
2	Not allocated
3	1 = The operator can change the limit (Rbk) for the hysteresis
4	Not allocated
5	1 = Operator can change the limit (Rbk) for low warning
6 - 31	Not allocated

Time delay after changing direction of rotation or restart of motor

Use the input parameter IdleTime to enter a time delay for changing the direction of rotation or restarting the motor. Use the Feature bit Enabling direct changeover between forward and reverse (Page 189) to define how the change is to take place.

Please note that IdleTime ≥ MonTiDynmic has to be configured.

Limit monitoring of an additional analog value

This block provides the standard function Limit monitoring of an additional analog value (Page 73).

Limit monitoring of the feedback

The block provides the standard function Limit monitoring of the feedback (Page 76). This limit monitoring is only active when the motor has started.

Limit monitoring with hysteresis

This block provides the standard function Limit monitoring with hysteresis (Page 78). It is performed via the input parameter AV_Hyst.

External/internal setpoint specification

The block provides the standard function Setpoint input - internal and external (Page 96).

Setpoint limitation

Limit the setpoint using the parameters:

- `SP_HiLim` (top)
- `SP_LoLim` (bottom)

Limit violations are displayed at the `SP_HiAct` and `SP_LoAct` output parameters with a 1.

Gradient limit of the setpoint

The block provides the standard function Ramp limiting of the setpoint (Page 108).

Using setpoint ramp

The block provides the standard function Using a setpoint ramp (Page 98).

Interlocks

This block provides the following interlocks:

- Activation enable
- Interlock without reset (interlock)
- Interlock with reset (protection)

Refer to the section Interlocks (Page 100) as well as Influence of the signal status on the interlock (Page 103).

Trip function

This block provides the standard function Trip function (Page 102).

Rapid stop for motors

This block provides the standard function Rapid stop for motors (Page 102).

Disabling interlocks

This block provides the standard function Disabling interlocks (Page 107).

Resetting the block in case of interlocks or errors

This block provides the standard function Resetting the block in case of interlocks or errors (Page 119).

Forming the group status for interlocks

This block provides the standard function Forming the group status for interlock information (Page 105).

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `FbkFwdOut.ST`
- `FbkRevOut.ST`
- `LocalLi.ST`
- `FwdLocal.ST`
- `StopLocal.ST`
- `RevLocal.ST`
- `Protect.ST`
- `Intlock.ST`
- `Permit.ST`
- `Trip.ST`
- `AV_Out.ST`
- `RbkOut.ST`
- `SP_Out.ST`

Forcing operating states

This block provides the standard function Forcing operating states (Page 115).

The following states can be enforced:

- Start forwards (`FwdForce`)
- Start in reverse (`RevForce`)
- Stop (`StopForce`)

Feedback monitoring

This block provides the standard function Monitoring the feedbacks (Page 83).

Maintenance release

This block provides the standard function Maintenance release (Page 47).

Specify warning times for control functions

This block provides the standard function Specifying warning times for control functions at motors and valves (Page 122).

Simulating signals

This block provides the standard function Simulating signals (Page 93).

Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 53).

Safe position

This block provides the standard function Safe position for motors, valves and controllers (Page 120).

Output signal as a pulse signal or static signal

This block provides the standard function Output signal as a static signal or pulse signal (Page 86).

Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 46).

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)
1	Reaction to the out of service mode (Page 199)
2	Resetting the commands for changing the mode (Page 193)
3	Enabling resetting of commands for the control settings (Page 193)
4	Setting switch or button mode (Page 195)
7	Enabling direct changeover between forward and reverse (Page 189)
9	Resetting via input signals in the event of interlocks or errors (Page 194)
10	Exiting local mode (Page 199)
14	Enabling rapid stop via faceplate (Page 192)
17	Enabling bumpless changeover to automatic mode for valves, motors, and dosers (Page 197)
22	Update acknowledgment and error status of the message call (Page 193)

Displaying auxiliary values

This block provides the standard function Displaying auxiliary values (Page 52).

Time stamp

This block receives a time stamp value via the `EventTSIn` input parameter. Refer to EventTs functions (Page 997) for more information on this.

SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 122).

See also

MotSpdCL block diagram (Page 667)

MotSpdCL I/Os (Page 658)

MotSpdCL messaging (Page 656)

MotSpdCL error handling (Page 654)

MotSpdCL modes (Page 646)

MotSpdCL description (Page 641)

5.3.4 MotSpdCL error handling

MotSpdCL troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers
- Mode changeover error

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Default value when implementing the block; block will not be processed.
0	There is no error.
41	The value for the connection `LocalSetting` is not within the approved limit from 0 to 4.
42	`LocalSetting = 0` or `LocalSetting = 3` or `LocalSetting = 4` and `LocalLi = 1`

51	`FwdLocal = 1` and `StopLocal = 1`
	`RevLocal = 1` and `StopLocal = 1`
	`FwdLocal = 1` and `RevLocal = 1`
	`FwdAut = 1` and `StopAut = 1`
	`RevAut = 1` and `StopAut = 1`
	`FwdAut = 1` and `RevAut = 1`
	`AutModLi = 1` and `ManModLi = 1`
	`FwdForce = 1` and `StopForce = 1`
	`RevForce = 1` and `StopForce = 1`
	`FwdForce = 1` and `RevForce = 1`
	`SP_LiOp = 1` and `SP_IntLi = 1` and `SP_ExtLi = 1`

52	`LocalAct = 1` and `LocalSetting = 2` or 4 and `SimOn = 1`

Mode changeover error

This error can be output by the block, see chapter Error handling (Page 117).

See also

MotSpdCL block diagram (Page 667)
MotSpdCL I/Os (Page 658)
MotSpdCL messaging (Page 656)
MotSpdCL functions (Page 648)
MotSpdCL modes (Page 646)
MotSpdCL description (Page 641)

5.3.5 MotSpdCL messaging

Messaging

The following messages can be generated for this block:

- Control system fault
- Instance-specific messages

Control system fault

The following control system error messages can be output:

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 1	AS process control message - fault	$$BlockComment$$ Motor feedback error
	SIG 2	AS process control message - fault	$$BlockComment$$ Tripping triggered
	SIG 3	AS process control message - fault	$$BlockComment$$ External error has occurred
	SIG 4	Warning - high	$$BlockComment$$ Rbk - high warning limit violated
	SIG 5	Warning - low	$$BlockComment$$ Rbk - low warning limit violated

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter CSF. If this signal changes to CSF = 1, a control system error is triggered (MsgEvId1, SIG 3).

Instance-specific messages

You can use up to three instance-specific messages with this block.

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 6	AS process control message - fault	$$BlockComment$$ External message 1
	SIG 7	AS process control message - fault	$$BlockComment$$ External message 2
	SIG 8	AS process control message - fault	$$BlockComment$$ External message 3

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Associated values for message instance `MsgEvId1`

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`
4	`ExtVa104`
5	`ExtVa105`
6	`ExtVa106`
7	`ExtVa107`
8	`ExtVa108`
9	Reserved
10	Reserved

The associated values 4 to 8 are allocated to the parameters `ExtVa104` ... `ExtVa108` and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

See also

MotSpdCL block diagram (Page 667)

MotSpdCL I/Os (Page 658)

MotSpdCL error handling (Page 654)

MotSpdCL functions (Page 648)

MotSpdCL modes (Page 646)

MotSpdCL description (Page 641)

Time stamp (Page 51)

5.3.6 MotSpdCL I/Os

MotSpdCL I/Os

Input parameters

Parameter	Description	Type	Default
AutModLi	1 = Automatic mode via interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AutModOp	1 = Automatic mode via operator (controlled by ModLiOp = 0)	BOOL	0
AV	Input additional analog value, to be connected to AV_Tech of the AV block	ANY	
AV_AH_Lim	Limit high alarm	REAL	95.0
AV_AL_Lim	Limit low alarm	REAL	5.0
AV_Hyst	Hysteresis for alarm, warning and tolerance limits	REAL	1.0
AV_TH_Lim	Limit high tolerance	REAL	85.0
AV_TL_Lim	Limit low tolerance	REAL	15.0
AV_WH_Lim	Limit high warning	REAL	90.0
AV_WL_Lim	Limit low warning	REAL	10.0
BatchEn	1 = Enable allocation	BOOL	0
BatchID	Batch ID	DWORD	16#00000000
BatchName	Batch name	STRING[32]	''
BypProt	1 = Bypassing interlock in local mode and simulation	BOOL	0
CSF	1 = External error (control system error)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
EN	1 = Called block will be processed	BOOL	1
EventTSIn	To wire the signal status of the EventTs message block; The EventTsIn input parameter is used to interconnect to the EventTsOut output parameter of the EventTs block. When this interconnection is configured, the messages of the EventTs block are displayed on the OS in the message view of the technological block and can also be acknowledged there.	STRUCT • Value: BYTE • ST: BYTE	- • 16#00 • 16#FF
ExtMsg1	Binary input for freely selectable message 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
ExtMsg2	Binary input for freely selectable message 2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg3	Binary input for freely selectable message 3	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtVa104	Associated value 4 for messages (MsgEvID1)	ANY	
ExtVa105	Associated value 5 for messages (MsgEvID1)	ANY	
ExtVa106	Associated value 6 for messages (MsgEvID1)	ANY	
ExtVa107	Associated value 7 for messages (MsgEvID1)	ANY	
ExtVa108	Associated value 8 for messages (MsgEvID1)	ANY	
FbkFwd	1 = Feedback for forward mode is present	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FbkRev	1 = Feedback for reverse mode is present	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Feature	I/O for additional functions (Page 648)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
FwdAut	1 = Activating forward mode of motor in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FwdForce	1 = Forcing activation of motor forward mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FwdLocal	1 = Activating forward mode of motor in local mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FwdMan	1 = Activating forward mode of motor in manual mode	BOOL	0
IdleTime	Wait time for change of direction or restart in [s]	REAL	5.0
Intlock	1 = Interlocking without reset is effective; you can operate the block without reset once the interlocking condition has disappeared 0 = Good state	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#FF

Parameter	Description	Type	Default
Intl_En	1 = Interlock without reset (interlock, Intlock parameter) is active	BOOL	1
LocalLi	1 = Activate local mode via plant signal	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
LocalOp	1 = Local mode by operator	BOOL	0
LocalSetting	Characteristics of Local mode	INT	0
ManModLi	1 = Manual mode via interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ManModOp	1 = Manual mode by OS operator (controlled by ModLiOp = 0)	BOOL	1
ModLiOp	Switchover of operating mode between: • 0 = Operator • 1 = Interconnection or SFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Monitor	1 = Feedback monitoring	BOOL	1
MonTiDynamic	Monitoring time for feedback errors after operation in [s]	REAL	3.0
MonTiStatic	Monitoring time for feedback errors without operation in [s]	REAL	3.0
MS_RelOp	1= Maintenance release via OS operator	BOOL	0
MsgEvId1	Message number (assigned automatically)	DWORD	16#FFFFFFFF
MsgLock	1 = Suppress process messages	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Occupied	1 = In use by a batch	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC (0-1 edge transition)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OS_Perm	I/O for operator control permissions (Page 648)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
OS1Perm	I/O for operator control permissions (Page 648)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
Permit	1 = Enable for opening / closing from safe position 0 = No OS release for energizing motor	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#FF

Parameter	Description	Type	Default
Perm_En	1 = Activation enable (enable, Permit parameter) is active	BOOL	1
Protect	0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block 1 = Good state	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#FF
Prot_En	1 = Protective interlock (protection, Protect parameter) is active	BOOL	1
PulseWidth	Pulse width of control signal [s]	REAL	3.0
RapidStp	Rapid stop for the motor: 0 = Motor On 1 = Motor Off	BOOL	0
Rbk	Additional analog value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
RbkW_DC	Delay time for incoming warnings [s	REAL	0.0
RbkW_DG	] Delay time for outgoing warnings [s]	REAL	0.0
RbkHyst	Hysteresis for warning limits	REAL	1.0
RbkOpScale	Limit for scale in bar graph of faceplate	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
RbkUnit	Unit of measure for additional analog value	INT	0
RbkWH_En	1 = Enable high warning	BOOL	1
RbkWH_Lim	Limit high warning	REAL	90.0
RbkWH_MsgEn	1 = Enable high warning message	BOOL	1
RbkWL_En	1 = Enable low warning	BOOL	0
RbkWL_Lim	Limit low warning	REAL	10.0
RbkWL_MsgEn	1 = Enable low warning message	BOOL	1
RevAut	1 = Activation of reverse motor operation in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RevForce	1 = Force activation of reverse motor operation	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RevLocal	1 = Activation of reverse motor operation in local mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RevMan	1 = Activation of reverse motor operation in manual mode	BOOL	0

Parameter	Description	Type	Default
RstLi	1 = Reset via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RstOp	1 = Reset via operator	BOOL	0
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SelFp1	Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
SimAV	Additional value used for SimOn = 1	REAL	0.0
SimOn	1 = Simulation on	BOOL	0
SimRbk	Position feedback used for SimOn = 1	REAL	0.0
SP_DnRaLim	Limit (low) for ramp of setpoint [SP_Unit/s]	REAL	100.0
SP_Ext	external setpoint - (to interconnection)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_ExtLi	1 = Select external setpoint (via interconnection)	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_ExtOp	1 = Select external setpoint (via operator)	BOOL	0
SP_HiLim	Limit (high) of setpoint	REAL	100.0
SP_LoLim	Limit (low) of setpoint	REAL	0.0
SP_Int	Internal setpoint for operation	REAL	1.0
SP_IntLi	1 = Select internal setpoint (via interconnection)	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_IntOp	1 = Select internal setpoint (via operator)	BOOL	1
SP_LiOp	Select setpoint source (internal/external): 1 = Via interconnection 0 = Via operator	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SP_OpScale	OS display range for setpoint SP	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
SP_RateOn	1 = Activate limitation of setpoint gradients	BOOL	0

Parameter	Description	Type	Default
SP_RmpModTime	1 = Use time (SP_RmpTime) for setpoint ramp 0 = Use gradient	BOOL	0
SP_RmpOn	1 = Activate setpoint ramp to target setpoint SP_RmpTarget	BOOL	0
SP_RmpTarget	Target setpoint for setpoint ramp	REAL	0.0
SP_RmpTime	Time for setpoint ramp [s] from current SP up to SP_RmpTarget	REAL	0.0
SP_TrkExt	1 = Bumpless switchover from external to internal setpoint active	BOOL	0
SP_TrkPV	1 = Setpoint follows PV in manual mode and with tracking	BOOL	0
SP_Unit	Unit of measure for setpoint	INT	1342
SP_UpRaLim	Gradient limit (high) for setpoint [SP_Unit/s]	REAL	100.0
StepNo	Batch step number	DWORD	16#00000000
StopAut	1 = Stop the motor in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StopForce	1 = Force motor stop	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StopLocal	1 = Stop the motor in local mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StopMan	1 = Stop the motor in manual mode	BOOL	0
Trip	1 = Motor is in "good" state	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
UserAna1	Analog auxiliary value 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
UA1unit	Unit of measure for analog auxiliary value 1	INT	0
UserAna2	Analog auxiliary value 2	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
UA2unit	Unit of measure for analog auxiliary value 2	INT	0
UserStatus	Freely assignable user area for status word (bit 24 to bit 31)	BYTE	16#00

Parameter	Description	Type	Default
WarnTiAut	Prewarning of motor start in automatic mode in [s]	REAL	3.0
WarnTiMan	Prewarning of motor start in manual mode in [s]	REAL	3.0

Output parameters

Parameter	Description	Type	Default
AutAct	1 = Automatic mode is active 0 = Manual mode is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AV_OpScale	Limit for scale in AV bar graph of faceplate	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
AV_Out	Output additional analog value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
AV_Unit	Unit of measure for additional analog value	INT	0
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see MotSpdCL error handling (Page 654)	INT	-1
FbkFwdOut	Feedback: 1 = Forward operation active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FbkRevOut	Feedback: 1 = Reverse operation active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Fwd	1 = Control of motor forward	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
LocalAct	1 = Local mode active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
LockAct	1 = Interlock (Inlock, Permit or Protect) is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ManAct	1 = Manual mode active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
MonDynErr	1 = Feedback error due to control change	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MonStaErr	1 = Feedback error due to unexpected feedback change	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_Release	Maintenance release: 1 = Release for OS operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MsgAckn1	Alarm acknowledgement status 1 (output STATUS of first ALARM_8P)	WORD	16#0000
MsgErr1	Alarm error 1 (output ERROR of first ALARM_8P)	BOOL	0
MsgStat1	Alarm status 1 (output ERROR of first ALARM_8P)	WORD	16#0000
OosAct	1 = Block is out of service	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
OS1PermOut	Display of OS1Perm	DWORD	16#FFFFFFFF
OS1PermLog	Display of OS1Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
P_Fwd	Pulse control of motor: Forward	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
P_Rev	Pulse control of motor: Reverse	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
P_Stop	Pulse control of motor : Stopping	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
R_StpAct	1 = Rapid stop of the motor is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RbkOut	Output of readback value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
RbkWH_Act	1 = High warning active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
RbkWL_Act	1 = Low warning active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Rev	1 = Control of motor: Reverse	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP_DnRaAct	1 = Negative gradient limiting of setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
SP_ExtAct	1 = External setpoint is active 0 = Internal setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
SP_HiAct	1 = Limit (high) for setpoint has been reached	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
SP_LoAct	1 = Limit (low) for setpoint has been reached	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
SP_Out	Setpoint used by controller	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_UpRaAct	1 = Positive gradient limiting of setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
Status1	Status word 1 (Page 641)	DWORD	16#00000000
Status2	Status word 2 (Page 641)	DWORD	16#00000000
Status3	Status word 3 (Page 641)	DWORD	16#00000000
Status4	Status word 4	DWORD	16#00000000
ST_Worst	Worst signal status	BYTE	16#80
WarnAct	1 = Prewarning for motor start active (parameters WarnTiAut and WarnTiMan	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

MotSpdCL block diagram (Page 667)

MotSpdCL messaging (Page 656)

MotSpdCL modes (Page 646)

Local mode (Page 36)

5.3.7 MotSpdCL block diagram

MotSpdCL block diagram

A block diagram is not provided for this block.

See also

MotSpdCL I/Os (Page 658)
MotSpdCL messaging (Page 656)
MotSpdCL error handling (Page 654)
MotSpdCL functions (Page 648)
MotSpdCL modes (Page 646)
MotSpdCL description (Page 641)

5.3.8 Operator control and monitoring

5.3.8.1 MotSpdCL views

Views of the MotSpdCL block

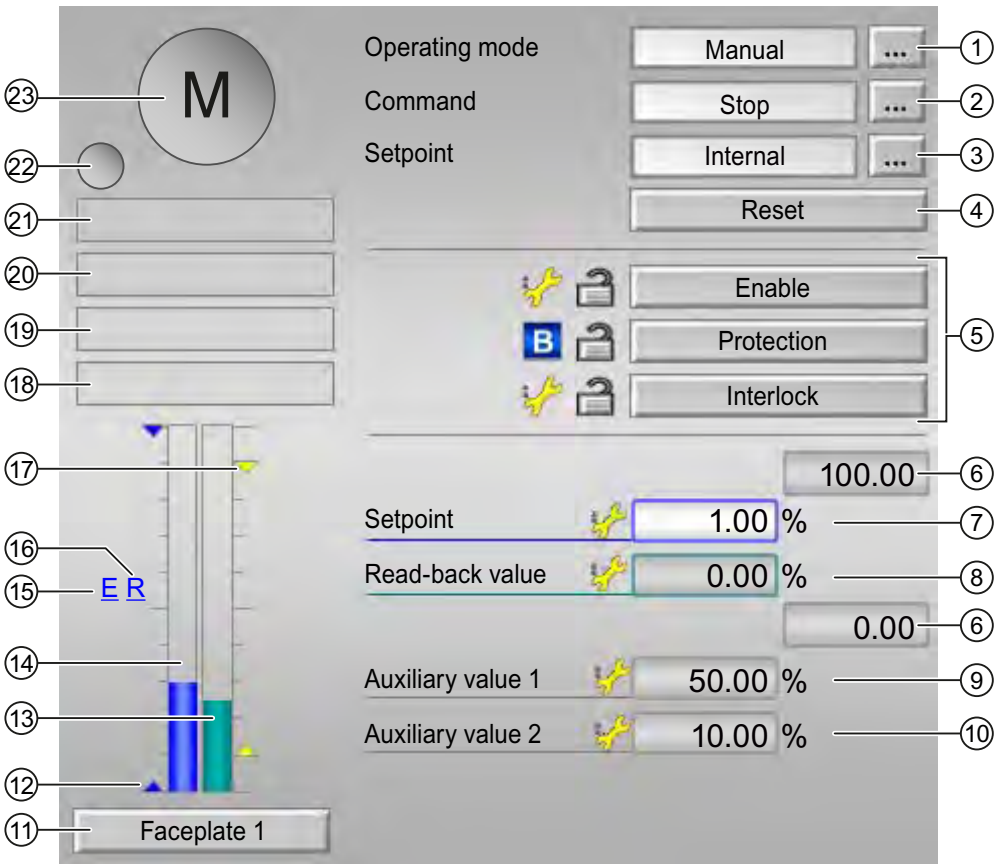
The block MotSpdCL provides the following views:

- MotSpdCL standard view (Page 668)
- Message view (Page 169)
- Limit value view of motors (Page 163)
- Limit view for readback values of MotSpdCL (Page 673)
- Trend view (Page 172)
- Ramp view (Page 167)
- MotSpdCL parameter view (Page 674)
- MotSpdCL preview (Page 676)
- Memo view (Page 171)
- Batch view (Page 170)
- Block symbol for MotSpdCL (Page 679)

Refer to the sections Structure of the faceplate (Page 123) and Block icon structure (Page 174) for general information on the faceplate and block icon.

5.3.8.2 MotSpdCL standard view

MotSpdCL standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 32)
- Automatic mode (Page 32)
- Local mode (Page 36)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Starting and stopping a motor

This area shows you the default operating state for the variable motor. The following states can be shown and executed here:

- Start |→
- Start ←|
- Stop
- Rapid stop

Refer to the Switching operating states and operating modes (Page 127) section for information on changing the state.

(3) Switch setpoint internal / external

This area shows how to specify the setpoint. The setpoint can specified as follows:

- By the application (External, CFC/SFC)
- By the user direct in the faceplate (Internal).

Refer to the Switching operating states and operating modes (Page 127) section for information on changing the setpoint specification.

You can find additional information on this in the Setpoint input - internal and external (Page 96) section.

(4) Resetting the block

Click "Reset" for interlocks or errors. You can find additional information on this in the Resetting the block in case of interlocks or errors (Page 119) section.

(5) Display area for interlock functions of the block

You can use this button to control the interlock functions of the block. You can find additional information on this in the Interlocking functions (Page 100) section.

The following is displayed in addition to the buttons:

- Interlock status (see Forming the group status for interlock information (Page 105)), e.g.:

- Signal status (see Forming and outputting signal status for blocks (Page 88)), e.g.:

  If there is a bypass of one of the interlock signals, the symbol for the bypass is shown instead of the signal status.

- Bypass information:

  If there is a bypass, it is displayed instead of the signal status.

(6) High and low scale range for the setpoint

This area is already set and cannot be changed.

(7) Display and change the setpoint including signal status

This area shows you the current setpoint with the corresponding signal status.

Refer to the Changing values (Page 129) section for information on changing the setpoint. The setpoint specification also needs to be set to "Internal" for this block.

(8) Display for readback value

This area shows you the current readback value with the corresponding signal status.

(9) and (10) Display of auxiliary values

You can use this area to display two auxiliary values that have been configured in the engineering system. You can find additional information on this in the Displaying auxiliary values (Page 52) section.

(11) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

(12) Display for limits

These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the ES.

(13) Bar graph for the readback value

This area shows you the current readback value in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(14) Bar graph for the setpoint

This area shows you the current setpoint in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(15) Display for external setpoint

This display [E] is only visible when you have selected "internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

(16) Display for the target setpoint of the setpoint ramp

This display [R] shows you the target setpoint and is only visible if you have enabled ramp generation in the Ramp view (Page 167).

(17) Limit display

These colored triangles show you the configured limits in the respective bar graph.

(18) Display area for states of the block

This area provides additional information on the operating state of the block:

- Maintenance

Additional information on the display area for states of the block is available in section Maintenance release (Page 47).

(19) Display area for states of the block

This area provides additional information on the operating state of the block:

- Simulation

You can find additional information on this in the Simulating signals (Page 93) section.

(20) Display area for states of the block

This area provides additional information on the operating state of the block:

- Trip
- Runtime error
- Control error
- Invalid signal
- Mode switch fail

Additional information on these errors is available in the sections Monitoring the feedbacks (Page 83) , Error handling (Page 117) (section "Invalid input signals" and "Mode changeover error") and Trip function (Page 102).

(21) Display area for states of the block

This area provides additional information on the operating state of the block:

- Force stop
- Force start |→
- Force start ←|

You can find additional information on this in the Forcing operating states (Page 115) section.

(22) Automatic preview

This display is only visible in manual mode or local mode when the current output signals are not identical to the control in automatic mode.

The display shows what state the motor would assume if you switched to automatic mode.

(23) Status display of motor

This area shows if the motor is running or idle:

- Green: the motor is running
- Gray: Motor is idle (as shown in the figure above)
- Red: Error in motor

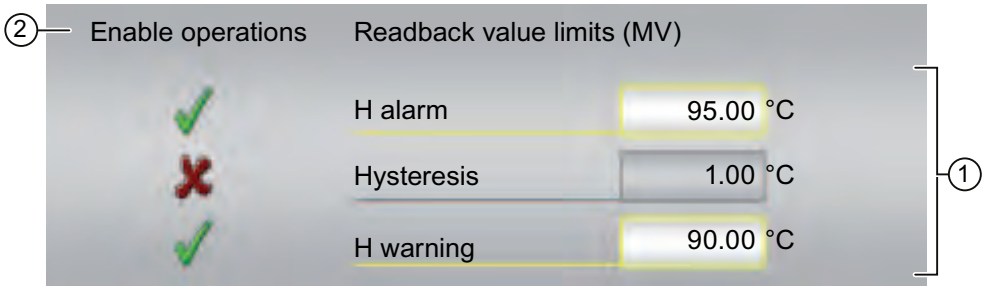
5.3.8.3 Limit view for readback values of MotSpdCL

Limit value view for readback values of MotSpdCL

Several values are set in this view by default:

- Readback value limits

The toolbars of the faceplate and the block icon indicate when the limit(s) is reached or exceeded.



(1) Display and change the limits for the readback value

In this area, you can enter the limits for the readback value. Refer to the Changing values (Page 129) section for more on this.

You can change the following limits:

- H alarm: Alarm high
- Hysteresis
- L alarm: Alarm low

(2) Enable operations

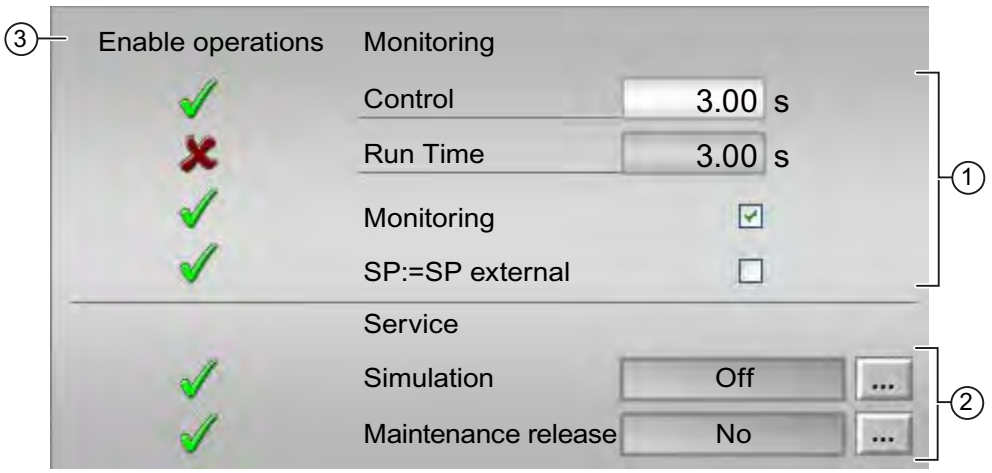
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions

5.3.8.4 MotSpdCL parameter view

MotSpdCL parameter view



(1) Monitoring

In this area, you change parameters and therefore influence the motor. Refer to the Changing values (Page 129) section for more on this.

You can influence the following parameters:

- Control: Monitoring time during startup and shutdown of the motor (dynamic)
- Runtime: Monitoring time during permanent operation of the motor (static)

Enable monitoring

You can enable monitoring by clicking the check box (☑)

You can find additional information on this in the Monitoring the feedbacks (Page 83) section.

SP := activate SP external

SP := SP external: ☑ Bumpless switchover of setpoint from external to internal. The internal setpoint is tracked to the external one.

(2) Service

You can select the following functions in this area:

- Simulation
- Maintenance release

Refer to the Switching operating states and operating modes (Page 127) section for more on this.

You can find information on this area in the section:

- Simulating signals (Page 93)
- Maintenance release (Page 47)

(3) Enable operations

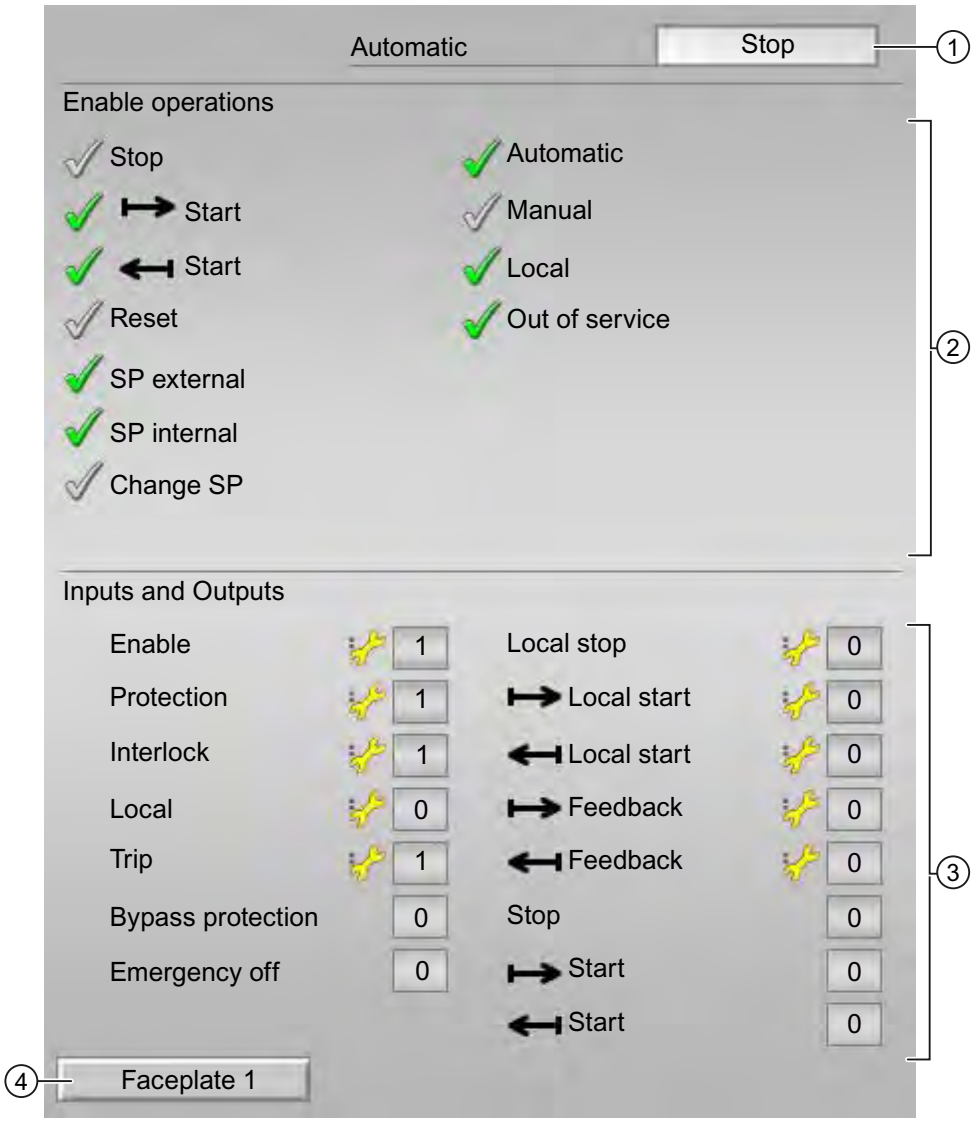
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions.

5.3.8.5 MotSpdCL preview

MotSpdCL preview



(1) Automatic preview

This area shows you the block status after its has switched from manual to automatic mode.

If the block is in automatic mode, the current block state is displayed.

(2) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions

The following operator control enables are shown here:

- Stop: You can stop the motor.
- Start |→ : You can start the motor.
- Start ←|: You can start the motor.
- Reset: You can reset the motor after interlocks or errors.
- SP external: You can connect to the external setpoint
- SP internal: You can connect to the internal setpoint
- Change SP: You can change the setpoint
- Automatic: You can switch to automatic mode.
- Manual: You can switch to manual mode
- Local: You can switch to local mode.
- Out of service: You can switch to out of service mode.

(3) Display current control signal

This area shows the most important parameters for this block with the current selection:

- Enable:
  - 0 = No OS release for energizing motor
  - 1 = Enable for opening / closing from safe position
- Protection:
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = Good state
- Interlock:
  - 0 = Interlocking without reset is effective; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = Good state
- Local: 1= Block is controlled in local mode
- Trip: 1 = Motor is in "good" state
- Bypass protection:
  - 0 = Bridging deactivated
  - 1 = Bypassing interlock in local mode and simulation
- Emergency off: 1=Activate emergency off (motor off)
- Local stop: 1= Block is controlled in local mode
- Local start |→ 1= Block is controlled in local mode
- Local start ←| 1= Block is controlled in local mode
- Feedback |→: 1 = Motor has started and is running
- Feedback←|: 1 = Motor has started and is running
- Stop: 1 = Motor stopped
- Start | → 1 = Start motor
- Start ←|: 1 = Motor started

(4) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

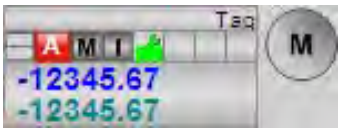
You can find additional information on this in the Calling further faceplates (Page 43) section.

5.3.8.6 Block symbol for MotSpdCL

Block icon for MotSpdCL

A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as the control system fault CSF
- Operating modes
- Internal and external setpoint specification
- Signal status, maintenance release
- Displays for bridging interlocks
- Interlocks
- Memo display
- Motor state display

Icons	Selection of the block icon in CFC	Special features
	1	Block icon in the full display
	2	
	3	
	4	

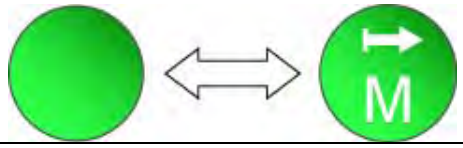
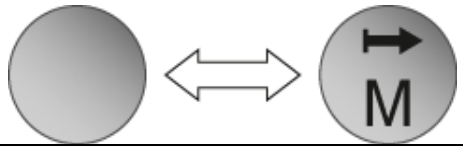
Icons	Selection of the block icon in CFC	Special features
	5	
	6	
	7	
	8	
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181).

Motor state display

The following motor states are shown here:

Icon	Meaning
	Motor started (motor icon changes)
	the motor is running
	Motor stopped (motor icon changes)
	Motor idle
	Error at motor

5.4 MotSpdL - Two-speed motor

5.4.1 Description of MotSpdL

Object name (type + number) and family

Type + number: FB 1856

Family: Drives

Area of application for MotSpdL

The block is used for the following applications:

- Control of reversible motors

How it works

The block is used to control motors with two speeds. Various inputs are available for controlling the motor. You can add monitoring of a maximum of two feedbacks produced by the contactor relay.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB3x). The block is also installed automatically in the startup OB (OB 100).

Further addressing is not required.

For the MotSpdL block, the Advanced Process Library contains a template for process tag types as an example with an application scenario for this block.

Example of process tag types:

- Two-speed motor (Motor2Speed) (Page 1450)

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

Status word allocation for `Status1` parameter

For a description of the individual parameters, see the MotSpdL I/Os (Page 697) section.

Status bit	Parameter
0	`Occupied`
1	`BatchEn`
2	`SimOn`
3	`OosAct.Value`
4	`OosLi.Value`
5	`0 = ManAct.Value` `1 = AutoAct.Value`
6	`LocalAct.Value`
7	`LockAct.Value`
8	`Spd1.Value`
9	Motor is stopped
10	`Spd2.Value`
11	`MonStaErr.Value`
12	`MonDynErr.Value`
13	`BypProt`
14	Invalid signal status
15	Mode changeover error
16	`1 = Intlock` is active
17	`1 = Permit` is active
18	`1 = Protect` is active
19	`Trip.Value`
20	`Spd1Force.Value`
21	`StopForce.Value`
22	`Spd2Force.Value`
23	"Interlock" button is enabled
24 - 25	Not used
26	Bypass information
27	Automatic preview for `(Speed1)`
28	Automatic preview for `(Stop)`
29	Automatic preview for `(Speed2)`
30	"Permission" button is enabled
31	"Protection" button is enabled

Status word allocation for `Status2` parameter

Status bit	Parameter
0	`MsgLock`
1	`AV_AH_Act.Value`
2	`AV_WH_Act.Value`
3	`AV_TH_Act.Value`
4	`AV_TL_Act.Value`
5	`AV_WL_Act.Value`
6	`AV_AL_Act.Value`
7	`AV_AH_En`
8	`AV_WH_En`
9	`AV_TH_En`
10	`AV_TL_En`
11	`AV_WL_En`
12	`AV_AL_En`
13	`AV_AH_MsgEn`
14	`AV_WH_MsgEn`
15	`AV_TH_MsgEn`
16	`AV_TL_MsgEn`
17	`AV_WL_MsgEn`
18	`AV_AL_MsgEn`
19	1 = No impact of input signals on local mode with `LocalSetting = 2` and `LocalSetting = 4`
20	Motor is stopped
21	Motor at speed 1 is stopped
22	Motor at speed 2 is stopped
23	Motor at speed 1 is started
24	Motor running at speed 1
25	Motor at speed 2 is started
26	Motor running at speed 2
27	Error when stopping motor
28	Error with motor speed 1
29	Error with motor speed 2
30	Display for interlocks in block icon
31	`MS_RelOp`

Status word allocation for `Status3` parameter

Status bit	Parameter
0 - 18	Not used
19	1 = Enabled for emergency stop push button (`Feature` bit Enabling rapid stop via faceplate (Page 192))
20 - 22	Not used
23	Command for rapid stop
24	Command for starting > the motor
25	Command for starting >> the motor
26	Show automatic preview in the standard view
27 - 29	Not used
30	Auxiliary value 1 visible
31	Auxiliary value 2 visible

Status word allocation for `Status4` parameter

Status bit	Parameter
0	Effective `Signal1` of the message block connected via `EventTsIn`
1	Effective `Signal2` of the message block connected via `EventTsIn`
2	Effective `Signal3` of the message block connected via `EventTsIn`
3	Effective `Signal4` of the message block connected via `EventTsIn`
4	Effective `Signal5` of the message block connected via `EventTsIn`
5	Effective `Signal6` of the message block connected via `EventTsIn`
6	Effective `Signal7` of the message block connected via `EventTsIn`
7	Effective `Signal8` of the message block connected via `EventTsIn`
8	AV not interconnected
9 - 31	Not used

See also

Functions of MotSpdL (Page 688)
MotSpdL messaging (Page 695)
MotSpdL block diagram (Page 704)
MotSpdL error handling (Page 693)
MotSpdL modes (Page 686)

5.4.2 MotSpdL modes

MotSpdL modes

The block can be operated using the following modes:

- Local mode (Page 36)
- Automatic mode (Page 32)
- Manual mode (Page 32)
- Out of service (Page 27)

The next section provides additional block-specific information relating to the general descriptions. This includes, for example, the parameters for mode changes.

"Local mode"

You will find general information on "Local mode", switching modes and bumpless switchover in the section Local mode (Page 36).

Motor actions you can control in local mode:

- starting with speed 1 (`Spd1Local = 1`)
- starting with speed 2 (`Spd2Local = 1`)
- Stop (`StopLocal = 1`).

A motor operated in local mode is controlled either by local signals (input parameters `Spd1Local = 1`, `Spd2Local = 1` and `StopLocal = 1`) or feedback signals (input parameters `FbkSpd1 = 1` and `FbkSpd2 = 1`). Configuration takes place via the input parameter `LocalSetting`.

"Automatic mode"

You will find general information on "Automatic mode", switching modes and bumpless switchover in the section Manual and automatic mode for motors, valves and dosers (Page 32).

Motor actions you can control in auto mode:

- starting with speed 1 (`Spd1Aut = 1`)
- starting with speed 2 (`Spd2Aut = 1`)
- Stop (`StopAut = 1`).

"Manual mode"

You will find general information on "Manual mode", switching modes and bumpless switchover in the section Manual and automatic mode for motors, valves and dosers (Page 32).

- starting with speed 1 (`Spd1Man = 1`)
- starting with speed 2 (`Spd2Man = 1`)
- Stop (`StopMan = 1`).

"Out of service"

You will find general information about the "Out of service" mode in the section Out of service (Page 27).

See also

MotSpdL block diagram (Page 704)

MotSpdL I/Os (Page 697)

MotSpdL messaging (Page 695)

MotSpdL error handling (Page 693)

Functions of MotSpdL (Page 688)

Description of MotSpdL (Page 682)

5.4.3 Functions of MotSpdL

Functions of MotSpdL

The functions for this block are listed below.

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

Operator control permissions via parameter OS_Perm

The block has the following Operator permissions (Page 45) for the OS_Perm parameter:

Bit	Function
0	1 = Operator can switch to automatic mode
1	1 = Operator can switch to manual mode
2	1 = Operator can switch to local mode
3	1 = Operator can switch to out of service mode
4	1 = Operator can stop the motor
5	1 = Operator can start the motor at speed 1
6	1 = Operator can start the motor at speed 2
7	1 = Operator can reset the motor
8	1 = Operator can define the monitoring time for startup
9	1 = Operator can define the monitoring time for runtime
10	1 = Operator can activate the function monitoring time (bits 8 & 9)
11	1 = Operator can enable function simulation
12	1 = Operator can enable the maintenance release function
13	1 = Operator can change the limit (AV) for the high alarm
14	1 = Operator can change the limit (AV) for the high warning
15	1 = Operator can change the limit (AV) for the high tolerance
16	1 = Operator can change the limit (AV) for hysteresis
17	1 = Operator can change the limit (AV) for the low tolerance
18	1 = Operator can change the limit (AV) for the low warning
19	1 = Operator can change the limit (AV) for the low alarm
20 - 31	Not allocated

Time delay for changes to control

If the state of the motor is changed, this is only undertaken after the time you define in the `IdleTime` input parameter. For example, if the motor is being moved into the stop state, the command for starting can only be issued after the time saved in `IdleTime` has lapsed.

Limit monitoring of an additional analog value

This block provides the standard function Limit monitoring of an additional analog value (Page 73).

Limit monitoring with hysteresis

This block provides the standard function Limit monitoring with hysteresis (Page 78).

Interlocks

This block provides the following interlocks:

- Activation enable
- Interlock without reset (interlock)
- Interlock with reset (protection)

Refer to the section Interlocks (Page 100) as well as Influence of the signal status on the interlock (Page 103).

Trip function

This block provides the standard function Trip function (Page 102).

Rapid stop for motors

This block provides the standard function Rapid stop for motors (Page 102).

Disabling interlocks

This block provides the standard function Disabling interlocks (Page 107).

Resetting the block in case of interlocks or errors

This block provides the standard function Resetting the block in case of interlocks or errors (Page 119).

Forming the group status for interlocks

This block provides the standard function Forming the group status for interlock information (Page 105).

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `FbkSpd1Out.ST`
- `FbkSpd2Out.ST`
- `LocalLi.ST`
- `Spd1Local.ST`
- `StopLocal.ST`
- `Spd2Local.ST`
- `Protect.ST`
- `Intlock.ST`
- `Permit.ST`
- `Trip.ST`

Forcing operating states

This block provides the standard function Forcing operating states (Page 115).

The following states can be enforced:

- Speed 1 (`Spd1Force`)
- Speed 2 (`Spd2Force`)
- Stop (`StopForce`)

Feedback monitoring

This block provides the standard function Monitoring the feedbacks (Page 83).

Maintenance release

This block provides the standard function Maintenance release (Page 47).

Specify warning times for control functions

This block provides the standard function Specifying warning times for control functions at motors and valves (Page 122).

Simulating signals

This block provides the standard function Simulating signals (Page 93).

Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 53).

Safe position

This block provides the standard function Safe position for motors, valves and controllers (Page 120).

Output signal as a pulse signal or static signal

This block provides the standard function Output signal as a static signal or pulse signal (Page 86).

Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 46).

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)
1	Reaction to the out of service mode (Page 199)
2	Resetting the commands for changing the mode (Page 193)
3	Enabling resetting of commands for the control settings (Page 193)
4	Setting switch or button mode (Page 195)
9	Resetting via input signals in the event of interlocks or errors (Page 194)
10	Exiting local mode (Page 199)
14	Enabling rapid stop via faceplate (Page 192)
17	Enabling bumpless changeover to automatic mode for valves, motors, and dosers (Page 197)
22	Update acknowledgment and error status of the message call (Page 193)

Displaying auxiliary values

This block provides the standard function Displaying auxiliary values (Page 52).

Time stamp

This block receives a time stamp value via the `EventTSIn` input parameter. Refer to EventTs functions (Page 997) for more information on this.

SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 122).

See also

MotSpdL block diagram (Page 704)

MotSpdL modes (Page 686)

Time stamp (Page 51)

MotSpdL I/Os (Page 697)

5.4.4 MotSpdL error handling

MotSpdL troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers
- Mode changeover error
- Invalid input signals

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Default value when implementing the block; block will not be processed.
0	There is no error.
41	The value for the connection `LocalSetting` is not within the approved limit from 0 to 4.
42	`LocalSetting = 0` or `LocalSetting = 3` or `LocalSetting = 4` and `LocalLi = 1`

51	`Spd1Local = 1` and `StopLocal = 1`
	`Spd2Local = 1` and `StopLocal = 1`
	`Spd1Local = 1` and `Spd2Local = 1`
	`Spd1Aut = 1` and `StopAut = 1`
	`Spd2Aut = 1` and `StopAut = 1`
	`Spd1Aut = 1` and `Spd2Aut = 1`
	`AutModLi = 1` and`ManModLi = 1`
	`Spd1Force = 1` and `StopForce = 1`
	`Spd2Force = 1` and `StopForce = 1`
	`Spd1Force = 1` and `Spd2Force = 1`

52	`LocalAct = 1` and `LocalSetting = 2` or `4` and `SimOn = 1`

Mode changeover error

This error can be output by the block, see chapter Error handling (Page 117).

Invalid input signals

This error can be output by the block, see the section Error handling (Page 117).

See also

MotSpdL block diagram (Page 704)
MotSpdL I/Os (Page 697)
Functions of MotSpdL (Page 688)
MotSpdL modes (Page 686)
Description of MotSpdL (Page 682)
MotSpdL messaging (Page 695)

5.4.5 MotSpdL messaging

Messaging

The following messages can be generated for this block:

- Control system fault
- Instance-specific messages

Control system fault

The following control system error messages can be output:

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 1	AS process control message - fault	$$BlockComment$$ Motor feedback error
	SIG 2	AS process control message - fault	$$BlockComment$$ Tripping triggered
	SIG 3	AS process control message - fault	$$BlockComment$$ External error has occurred

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter `CSF`. If this signal changes to `CSF = 1`, a control system error is triggered (`MsgEvId1, SIG 3`).

Instance-specific messages

You can use up to three instance-specific messages with this block.

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 4	AS process control message - fault	$$BlockComment$$ External message 1
	SIG 5	AS process control message - fault	$$BlockComment$$ External message 2
	SIG 6	AS process control message - fault	$$BlockComment$$ External message 3

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Associated values for message instance `MsgEvId1`

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`
4	`ExtVa104`
5	`ExtVa105`
6	`ExtVa106`
7	`ExtVa107`
8	`ExtVa108`
9	Reserved
10	Reserved

The associated values 4 to 8 are allocated to the parameters `ExtVa104` ... `ExtVa108` and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

See also

MotSpdL block diagram (Page 704)

MotSpdL modes (Page 686)

MotSpdL error handling (Page 693)

Time stamp (Page 51)

5.4.6 MotSpdL I/Os

MotSpdL I/Os

Input parameters

Parameter	Description	Type	Default
AutModLi	1 = Automatic mode via interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AutModOp	1 = Automatic mode via operator (controlled by ModLiOp = 0)	BOOL	0
AV	Input additional analog value, to be connected to AV_Tech of the AV block	ANY	
AV_AH_Lim	Limit high alarm	REAL	95.0
AV_AL_Lim	Limit low alarm	REAL	5.0
AV_Hyst	Hysteresis for alarm, warning and tolerance limits	REAL	1.0
AV_TH_Lim	Limit high tolerance	REAL	85.0
AV_TL_Lim	Limit low tolerance	REAL	15.0
AV_WH_Lim	Limit high warning	REAL	90.0
AV_WL_Lim	Limit low warning	REAL	10.0
BatchEn	1 = Enable allocation	BOOL	0
BatchID	Batch ID	DWORD	16#00000000
BatchName	Batch name	STRING[32]	''
BypProt	1 = Bypassing interlock in local mode and simulation	BOOL	0
CSF	1 = External error (control system error)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
EN	1 = Called block will be processed	BOOL	1
EventTSIn	To wire the signal status of the EventTs message block; The EventTsIn input parameter is used to interconnect to the EventTsOut output parameter of the EventTs block. When this interconnection is configured, the messages of the EventTs block are displayed on the OS in the message view of the technological block and can also be acknowledged there.	STRUCT • Value: BYTE • ST: BYTE	- • 16#00 • 16#FF
ExtMsg1	Binary input for freely selectable message 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
ExtMsg2	Binary input for freely selectable message 2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg3	Binary input for freely selectable message 3	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtVa104	Associated value 4 for messages (MsgEvID1)	ANY	
ExtVa105	Associated value 5 for messages (MsgEvID1)	ANY	
ExtVa106	Associated value 6 for messages (MsgEvID1)	ANY	
ExtVa107	Associated value 7 for messages (MsgEvID1)	ANY	
ExtVa108	Associated value 8 for messages (MsgEvID1)	ANY	
FbkSpd1	1 = Feedback for speed 1 is present	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FbkSpd2	1 = Feedback for speed 2 is present	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Feature	I/O for additional functions (Page 688)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
IdleTime	Wait time for restart of motor in [s]	REAL	5.0
Intlock	0 = Interlocking without reset is effective; you can operate the block without reset once the interlocking condition has disappeared 1 = Good state	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#FF
Intl_En	1 = Interlock without reset (interlock, Intlock parameter) is active	BOOL	1
LocalLi	1 = Activate local mode via plant signal	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
LocalOp	1 = Local mode by operator	BOOL	0
LocalSetting	Properties for the Local mode (Page 36)	INT	0
ManModLi	1 = Manual mode via interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
ManModOp	1 = Manual mode by OS operator (controlled by ModLiOp = 0)	BOOL	1
ModLiOp	Switchover of operating mode between: • 0 = Operator • 1 = Interconnection or SFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Monitor	1 = Feedback monitoring	BOOL	1
MonTiDynamic	Monitoring time for feedback errors after operation in [s]	REAL	3.0
MonTiStatic	Monitoring time for feedback errors without operation in [s]	REAL	3.0
MS_RelOp	1= Maintenance release via OS operator	BOOL	1
MsgEvId1	Message number (assigned automatically)	DWORD	16#FF
MsgLock	1 = Suppress process messages	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_RelOp	1 = Maintenance release via OS operator	BOOL	0
Occupied	1 = In use by a batch	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC (0-1 edge transition)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OS_Perm	I/O for operator control permissions (Page 688)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
Permit	1 = Enable for opening / closing from safe position 0 = No OS release for energizing motor	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#FF
Perm_En	1 = Activation enable (enable, Permit parameter) is active	BOOL	1
Protect	0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block 1 = Good state	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#FF
Prot_En	1 = Protective interlock (protection, Protect parameter) is active	BOOL	1
PulseWidth	Pulse width of control signal [s]	REAL	3.0
RapidStp	Rapid stop for the motor: • 0 = Motor On • 1 = Motor Off	BOOL	0

Parameter	Description	Type	Default
Spd1Aut	1 = Activation of motor speed 1 in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Spd2Aut	1 = Activation of motor speed 2 in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Spd1Force	1 = Force activation of motor speed 1 in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Spd2Force	1 = Force activation of motor speed 2 in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Spd1Local	1 = Activate motor speed 1 in local mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Spd2Local	1 = Activate motor speed 2 in local mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Spd1Man	1 = Activate motor speed 1 in manual mode	BOOL	0
Spd2Man	1 = Activate motor speed 2 in manual mode	BOOL	0
RstLi	1 = Reset via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RstOp	1 = Reset via operator	BOOL	0
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SimAV	Additional value used for SimOn = 1	REAL	0.0
SimOn	1 = Simulation on	BOOL	0
SelFp1	Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
StepNo	Batch step number	DWORD	16#00000000
StopAut	1 = Stop the motor in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
StopForce	1 = Force motor stop	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StopLocal	1 = Stop the motor in local mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StopMan	1 = Stop the motor in manual mode	BOOL	0
Trip	1 = Motor is in "good" state	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
UserAna1	Analog auxiliary value 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
UA1unit	Unit of measure for analog auxiliary value 1	INT	0
UserAna2	Analog auxiliary value 2	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
UA1unit	Unit of measure for analog auxiliary value 1	INT	0
UserStatus	Freely assignable user area for status word (bit 24 to bit 31)	BYTE	16#00
WarnTiAut	Prewarning of motor start in automatic mode in [s]	REAL	3.0
WarnTiMan	Prewarning of motor start in manual mode in [s]	REAL	3.0

Output parameters

Parameter	Description	Type	Default
AutAct	1 = Automatic mode is active 0 = Manual mode is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AV_OpScale	Limit for scale in AV bar graph of faceplate	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
AV_Out	Output additional analog value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
AV_Unit	Unit of measure for additional analog value	INT	0
		•	•
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see MotSpdL error handling (Page 693)	INT	-1
F_Stop	1 = Rapid stop of the motor is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FbkSpd1Out	Feedback: 1 = Speed 1 active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FbkSpd2Out	Feedback: 1 = Speed 2 active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
LocalAct	1 = Local mode active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
LockAct	1 = Interlock (Inlock, Permit or Protect) is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ManAct	1 = Manual mode active	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
MonDynErr	1 = Feedback error due to control change	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
MonStaErr	1 = Feedback error due to unexpected feedback change	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_Release	Maintenance release: 1 = Release for OS operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MsgAckn1	Alarm acknowledgement status 1 (output STATUS of first ALARM_8P)	WORD	16#0000
MsgAckn2	Alarm acknowledgement status 2 (output STATUS of second ALARM_8P)	WORD	16#0000
MsgErr1	Alarm error 1 (output ERROR of first ALARM_8P)	BOOL	0
MsgErr2	Alarm error 2 (output ERROR of second ALARM_8P)	BOOL	0
MsgStat1	Alarm status 1 (output ERROR of first ALARM_8P)	WORD	16#0000
MsgStat2	Alarm status 2 (output ERROR of second ALARM_8P)	WORD	16#0000
OosAct	1 = Block is out of service	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
P_Spd1	Pulse control of motor speed 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
P_Spd2	Pulse control of motor: Speed 2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
P_Stop	Pulse control of motor : Stopping	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Spd1	1 = Control of motor: Speed 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Spd2	1 = Control of motor: Speed 2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Status1	Status word 1 (Page 682)	DWORD	16#00000000
Status2	Status word 2 (Page 682)	DWORD	16#00000000
Status3	Status word 3 (Page 682)	DWORD	16#00000000

Parameter	Description	Type	Default
Status4	Status word 4 (Page 682)	DWORD	16#00000000
WarnAct	1 = Prewarning for motor start active (parameters WarnTiAut and WarnTiMan)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

MotSpdL messaging (Page 695)
MotSpdL block diagram (Page 704)
MotSpdL modes (Page 686)

5.4.7 MotSpdL block diagram

MotSpdL block diagram

A block diagram is not provided for this block.

See also

MotSpdL I/Os (Page 697)
MotSpdL messaging (Page 695)
MotSpdL error handling (Page 693)
Functions of MotSpdL (Page 688)
MotSpdL modes (Page 686)
Description of MotSpdL (Page 682)

5.4.8 Operator control and monitoring

5.4.8.1 MotSpdL views

Views of the MotSpdL block

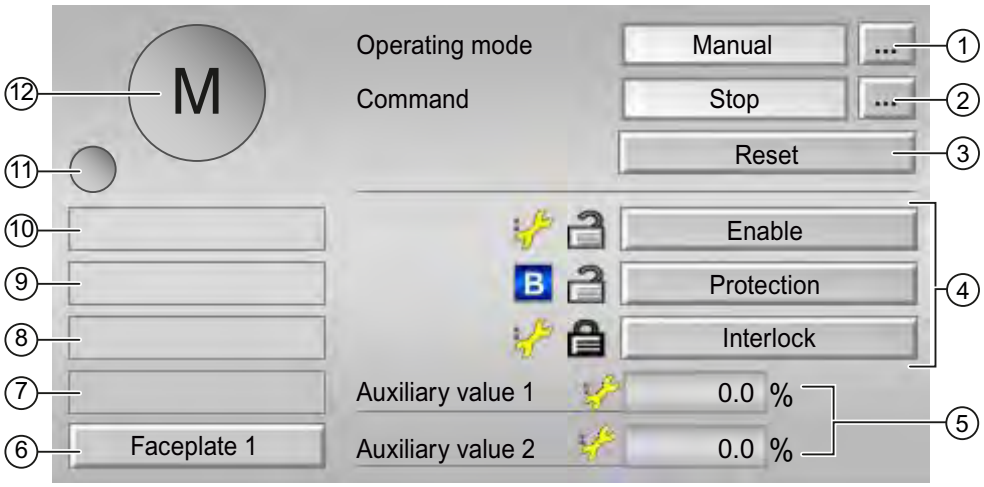
The block MotSpdL provides the following views:

- MotSpdL standard view (Page 706)
- Message view (Page 169)
- Limit value view of motors (Page 163)
- Trend view (Page 172)
- Parameter view for motors and valves (Page 155)
- MotSpdL preview (Page 709)
- Memo view (Page 171)
- Batch view (Page 170)
- Block symbol for MotSpdL (Page 712)

Refer to the sections Structure of the faceplate (Page 123) and Block icon structure (Page 174) for general information on the faceplate and block icon.

5.4.8.2 MotSpdL standard view

MotSpdL standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 32)
- Automatic mode (Page 32)
- Local mode (Page 36)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Starting and stopping a motor

This area shows you the default operating state for the motor. The following states can be shown and executed here:

- Start >
- Start >>
- Stop
- Rapid stop

Refer to the Switching operating states and operating modes (Page 127) section for information on changing the state.

(3) Resetting the block

Click "Reset" for interlocks or errors. You can find additional information on this in the Resetting the block in case of interlocks or errors (Page 119) section.

(4) Display area for interlock functions of the block

You can use this button to control the interlock functions of the block. You can find additional information on this in the Interlocking functions (Page 100) section.

The following is displayed in addition to the buttons:

- Interlock status (see Forming the group status for interlock information (Page 105)), e.g.:

- Signal status (see Forming and outputting signal status for blocks (Page 88)), e.g.:

  If there is a bypass of one of the interlock signals, the symbol for the bypass is shown instead of the signal status.

- Bypass information:

  If there is a bypass, it is displayed instead of the signal status.

(5) Display of auxiliary values

You can use this area to display two auxiliary values that have been configured in the engineering system. You can find additional information on this in the Displaying auxiliary values (Page 52) section.

(6) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

(7) Display area for states of the block

This area provides additional information on the operating state of the block:

- Maintenance

Additional information on the display area for states of the block is available in section Maintenance release (Page 47).

(8) Display area for states of the block

This area provides additional information on the operating state of the block:

- Simulation

You can find additional information on this in the Simulating signals (Page 93) section.

(9) Display area for states of the block

This area provides additional information on the operating state of the block:

- Trip
- Runtime error
- Control error
- Invalid signal
- Mode switch fail

Additional information on these errors is available in the sections Monitoring the feedbacks (Page 83) , Error handling (Page 117) (section "Invalid input signals" and "Mode changeover error") and Trip function (Page 102).

(10) Display area for states of the block

This area provides additional information on the operating state of the block:

- Force stop
- Force start >
- Force start >>

You can find additional information on this in the Forcing operating states (Page 115) section.

(11) Automatic preview

This display is only visible in manual mode or local mode when the current output signals are not identical to the control in automatic mode.

The display shows what state the motor would assume if you switched to automatic mode.

The following icons can be displayed:

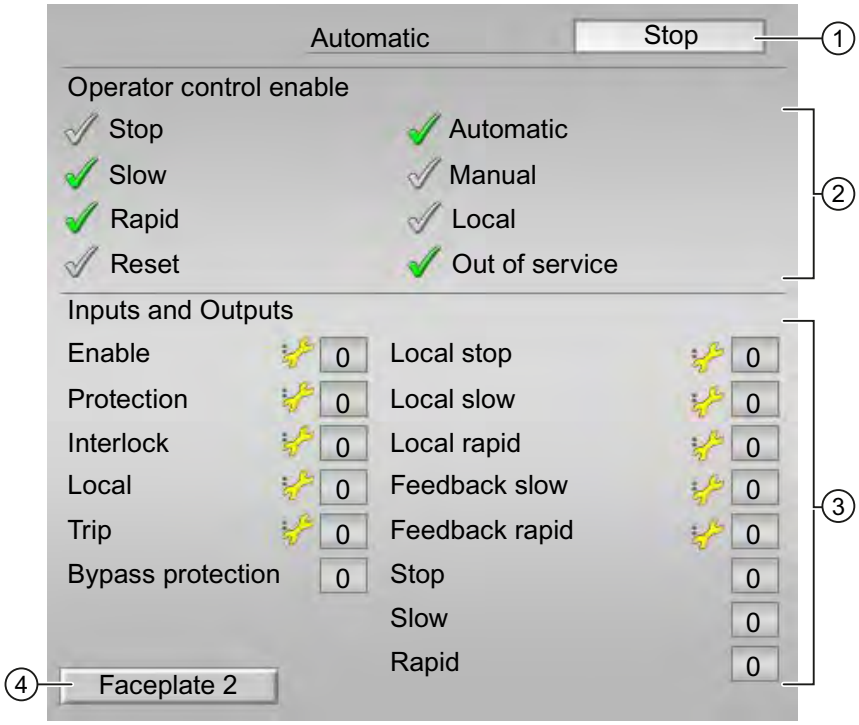
(12) Status display of motor

This area shows if the motor is running or idle:

- Green: the motor is running
- Gray: Motor is idle (as shown in the figure above)
- Red: A fault has occurred

5.4.8.3 MotSpdL preview

MotSpdL preview



(1) Automatic preview

This area shows you the block status after its has switched from manual to automatic mode.

If the block is in automatic mode, the current block state is displayed.

(2) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions

The following operator control enables are shown here:

- Stop: You can stop the motor.
- Slow: You can start the motor in the "slow" state
- Fast: You can start the motor in the "fast" state
- Reset: You can reset the motor after interlocks or errors.
- Automatic: You can switch to automatic mode.
- Manual: You can switch to manual mode
- Local: You can switch to local mode.
- Out of service: You can switch to out of service mode.

(3) Display current control signal

This area shows the most important parameters for this block with the current selection:

- Enable:
  - 0 = No OS release for energizing motor
  - 1 = Enable for starting / stopping from safe position
- Protection:
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = Good state
- Interlock:
  - 0 = Interlocking without reset is effective; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = Good state
- Local: 1= Block is controlled in local mode
- Trip: 1 = Motor is in "good" state
- Bypass protection:
  - 0 = Bridging deactivated
  - 1 = Bypassing interlock in local mode and simulation

- Local stop: 1 = Stop the motor in local mode
- Local slow: 1 = Start the motor in local mode, slow
- local fast ←: 1 = Start the motor in local mode, fast
- Feedback slow: 1 = Motor has started and is running slow
- Feedback fast: 1 = Motor has started and is running fast
- Stop: 1 = Motor stopped
- Slow: 1 = Motor is running slow
- Fast: 1 = Motor is running fast

(4) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

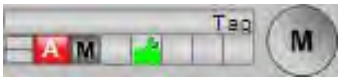
You can find additional information on this in the Calling further faceplates (Page 43) section.

5.4.8.4 Block symbol for MotSpdL

Properties of the MotSpdL block symbol

A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as the control system fault CSF
- Operating modes
- Signal status, maintenance release
- Displays for bridging interlocks
- Interlocks
- Memo display
- Motor state display

Icons	Selection of the block icon in CFC	Special features
	1	Block icon in the full display
	2	
	3	
	4	
	5	
	6	

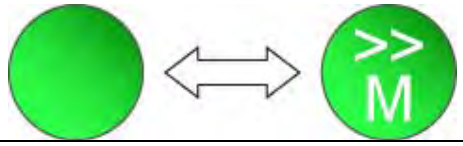
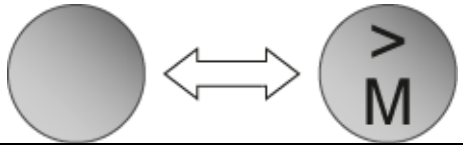
Icons	Selection of the block icon in CFC	Special features
	7	
	8	
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181).

Motor state display

The following motor states are shown here:

Icon	Meaning
	Motor started (motor icon changes)
	the motor is running
	Motor stopped (motor icon changes)
	Motor idle
	Error at motor

5.5 Vlv2WayL - Two-way valve

5.5.1 Description of Vlv2WayL

Object name (type + number) and family

Type + number: FB 1897

Family: Drives

Area of application for Vlv2WayL

The block is used for the following applications:

- Control of multi-way valves with up to three switching positions. One of these positions is the safe position (de-energized position)
- Control of three individual valves (valve network) to implement a 2-way valve circuit with safe position (de-energized position)

How it works

The multi-way valve (or valve network) is controlled via position 0 (safe position), position 1 (way 1), or position 2 (way 2). Various inputs are available for controlling the positions.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

For the Vlv2WayL block, the Advanced Process Library contains a template for process tag types as an example with an application scenario for this block.

Example of process tag types:

- Two-way valve (Valve2Way) (Page 1453)

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

Status word allocation for `Status1` parameter

For a description of the individual parameters, see the Vlv2WayL I/Os (Page 729) section.

Status bit	Parameter
0	`Occupied`
1	`BatchEn`
2	`SimOn`
3	`OosAct.Value`
4	`OosLi.Value`
5	`AutoAct.Value`
6	`LocalAct.Value`
7	`LockAct.Value`
8	Open/closed command for `V0`
9	Open/closed command for `V1`
10	Open/closed command for `V2`
11	Feedback error without control change
12	Feedback error due to control change
13	`BypProt` active
14	Invalid signal status
15	Mode changeover error
16	`1 = Intlock` is active
17	`1 = Permit` is active
18	`1 = Protect` is active
19	`Pos0Force.Value`
20	`Pos1Force.Value`
21	`Pos2Force.Value`
22	Feedback for Pos0 OK
23	Feedback for Pos1 OK
24	Feedback for Pos2 OK
25	Feedback for current position OK
26	Automatic preview Pos0
27	Automatic preview Pos1
28	Automatic preview Pos2
29	`SafeV0`
30	`SafeV1`
31	`SafeV2`

Status word allocation for `Status2` parameter

Status bit	Parameter
0	`MsgLock`
1	Forcing active
2	Display for interlocks in block icon
3-18	Not used
19	1 = No impact of input signals on local mode with `LocalSetting = 2` and `LocalSetting = 4`
20	`CtrlV0.Value`
21	`CtrlV1.Value`
22	`CtrlV2.Value`
23	`FbkV0Out.Value`
24	`FbkV1Out.Value`
25	`FbkV2Out.Value`
26	`FbkP0Out.Value`
27	Feedback V0 (`FbkV0`), for OS display only
28	Feedback V1 (FbkV1), for OS display only
29	Feedback V2 (`FbkV2`), for OS display only
30	Bypass information from previous function block
31	`MS_RelOp`

Status word allocation for `Status3` parameter

Status bit	Parameter
0	"Interlock" button is enabled
1	"Permission" button is enabled
2	"Protection" button is enabled
3	`Pos0Out`
4	Travel, position 0
5	Monitoring error, position 0
6	`Pos1Out`
7	Travel, position 1
8	Monitoring error, position 1
9	`Pos2Out`
10	Travel, position 2
11	Monitoring error, position 2
12	Preview for position 0, control `CtrlV0`
13	Preview for position 0, control `CtrlV1`
14	Preview for position 0, control `CtrlV2`
15	Preview for position 1, control `CtrlV0`
16	Preview for position 1, control `CtrlV1`
17	Preview for position 1, control `CtrlV2`

Status bit	Parameter
18	Preview for position 2, control `CtrlV0`
19	Preview for position 2, control `CtrlV1`
20	Preview for position 2, control `CtrlV2`
21	Preview for automatic control `CtrlV0`
22	Preview for automatic control `CtrlV1`
23	Preview for automatic control `CtrlV2`
24	`UserAna1` interconnected
25	`UserAna2` interconnected
26	Show automatic preview in the standard view
27 - 31	Not used

Status word allocation for `Status4` parameter

Status bit	Parameter
0	Effective `Signal1` of the message block connected via `EventTsIn`
1	Effective `Signal2` of the message block connected via `EventTsIn`
2	Effective `Signal3` of the message block connected via `EventTsIn`
3	Effective `Signal4` of the message block connected via `EventTsIn`
4	Effective `Signal5` of the message block connected via `EventTsIn`
5	Effective `Signal6` of the message block connected via `EventTsIn`
6	Effective `Signal7` of the message block connected via `EventTsIn`
7	Effective `Signal8` of the message block connected via `EventTsIn`
8 - 31	Not used

See also

Vlv2WayL functions (Page 720)
Vlv2WayL messaging (Page 727)
Vlv2WayL block diagram (Page 737)
Vlv2WayL error handling (Page 725)
Vlv2WayL modes (Page 718)
Resetting the block in case of interlocks or errors (Page 119)

5.5.2 Vlv2WayL modes

Vlv2WayL operating modes

The block can be operated using the following modes:

- Local mode (Page 36)
- Automatic mode (Page 32)
- Manual mode (Page 32)
- Out of service (Page 27)

The next section provides additional block-specific information relating to the general descriptions.

"Local mode"

You can find general information on "Local mode", switching modes and bumpless switchover in the Local mode (Page 36) section.

Valve actions you can control in local mode:

- Moving to safe position (`Pos0Local = 1`)
- Moving to position 1 (`Pos1Local = 1`)
- Moving to position 2 (`Pos2Local =1`.

A block operated in local mode is controlled either by local signals (input parameters `Pos0Local = 1, Pos1Local = 1` and `Pos2Local = 1`) or by feedback signals (input parameters `FdbV0, FdbV1`, `FdbV2` and `FdbP0;` if no position can be assigned, the last valid position is accepted). Configuration takes place via the input parameter `LocalSetting`.

"Automatic mode"

You can find general information on "Automatic mode", switching modes and bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 32) section.

Valve actions you can control in auto mode:

- Moving to safe position (`Pos0Aut = 1`)
- Moving to position 1 (`Pos1Aut = 1`)
- Moving to position 2 (`Pos2Aut =1`.

"Manual mode"

You will find general information on "Manual mode", switching modes and bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 32) section.

Valve actions you can control in manual mode:

- Moving to safe position (`Pos0Man = 1`)
- Moving to position 1 (`Pos1Man = 1`)
- Moving to position 2 (`Pos2Man = 1`.

"Out of service"

You will find general information about the "Out of service" mode in theOut of service (Page 27) section.

See also

Vlv2WayL block diagram (Page 737)

Vlv2WayL I/Os (Page 729)

Vlv2WayL messaging (Page 727)

Vlv2WayL error handling (Page 725)

Vlv2WayL functions (Page 720)

Description of Vlv2WayL (Page 714)

5.5.3 Vlv2WayL functions

Functions of Vlv2WayL

The functions for this block are listed below.

Defining valve positions for individual valves

The control outputs for position 1 and position 2 can be selected individually with `DefPos1` and `DefPos2`:

Way 1or Travel 2	Control outputs		
DefPos1 or DefPos2	Valve V0 (CtrlV0)	Valve V1 (CtrlV1)	Valve V2 (CtrlV2)
0	closed	closed	closed
1	closed	closed	open
2	closed	open	closed
3	closed	open	open
4	open	closed	closed
5	open	closed	open
6	open	open	closed
7	open	open	open

Position 0 is the safe position (de-energized state) and cannot be configured. In position 0 all control outputs are de-energized (`CtrlVx = 0`).

Output signal as a pulse signal or static signal

This block provides the standard function Output signal as a static signal or pulse signal (Page 86). In addition to the static control outputs `CtrlV0`, `CtrlV1` and `CtrlV2`, the block also has pulse outputs `P_CtrlV0`, `P_CtrlV1` and `P_CtrlV2`, which are dependent on the static control outputs. In addition the pulse signal `P_CtrlP0` is output for position 0.

Safe position

This block provides the standard function Safe position for motors, valves and controllers (Page 120). The safe position (de-energized state) is set individually using parameters `SafeV0`, `SafeV1` and `SafeV2` for each valve (`CtrlV0`, `CtrlV1` and `CtrlV2`).

- `SafeVx = 0` means that at `CtrlVx = 0` the valve drive closes and at`CtrlVx = 1` it opens (de-energized state is "closed")
- `SafeVx = 1` means that at `CtrlVx = 0` the valve drive opens and at`CtrlVx = 1` it closes(de-energized state is "open")

Specify warning times for control functions

This block provides the standard function Specifying warning times for control functions at motors and valves (Page 122). The warning signal is output before the valve moves to position 1 or position 2. No signal is output for position 0 (safe position).

Feedback monitoring

This block provides the standard function Monitoring the feedbacks (Page 83)

Startup characteristic monitoring is individually set up for every output signal `CtrlV0`, `CtrlV1` and `CtrlV2` via parameters `MonTiV0Dynamic`, `MonTiV1Dynamic` and `MonTiV2Dynamic` and is set for position 0 via `MonTiP0Dynamic`. The parameter `MonTiStatic` monitors compliance with the position.

Note

The monitoring function does not take account of the safe positions, i.e. feedback messages `FbkV0`, `FbkV1`, `FbkV2` and `FbkP0` are open or closed signals (e.g. `SafeV0 = 1` means that the `FbkV0 = 1` feedback is a closed signal).

`FbkP0` must not occur for positions 1 or 2; `FbkV0`, `FbkV1` and `FbkV2` must not occur for position 0.

If there are several feedback messages for position 0 (e.g. with a valve network), these must be combined at `FbkP0` using an upstream AND block.

Disabling feedbacks

This block provides the standard function Disabling feedback for valves (Page 85). Feedback monitoring can be deactivated separately for each feedback with `NoFdbV0`, `NoFdbV1`, `NoFdbV2` or `NoFdbP0` as required.

Forcing operating states

This block provides the standard function Forcing operating states (Page 115). The inputs `Pos0Force`, `Pos1Force`, and `Pos2Force` force the blocks into position 0, position 1 or position 2 respectively.

Simulating signals

This block provides the standard function Simulating signals (Page 93)

Interlocks

This block provides the following interlocks:

- Activation enable
- Interlock without reset (interlock)
- Interlock with reset (protection)

Refer to the Interlocks (Page 100) section for more on this.

Disabling interlocks

This block provides the standard function Disabling interlocks (Page 107)

Resetting the block in case of interlocks or errors

This block provides the standard function Resetting the block in case of interlocks or errors (Page 119)

Forming the group status for interlocks

This block provides the standard function Forming the group status for interlock information (Page 105).

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `FbkV0Out.ST`
- `FbkV1Out.ST`
- `FbkV2Out.ST`
- `FbkP0Out.ST`
- `Permit.ST`
- `Inlock.ST`
- `Protect.ST`

Maintenance release

This block provides the standard function Maintenance release (Page 47).

Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 53).

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 46).

Operator control permissions via parameter OS_Perm

The block has the following Operator permissions (Page 45) for the OS_Perm parameter:

Bit	Function
0	1 = Operator can switch to automatic mode
1	1 = Operator can switch to manual mode
2	1 = Operator can switch to local mode
3	1 = Operator can switch to out of service mode
4	1 = Operator can switch to position 0
5	1 = Operator can switch to position 1
6	1 = Operator can switch to position 2
7	1 = Operator can reset the valve
8	1 = Operator can define the monitoring time for startup
9	1 = Operator can define the monitoring time for runtime
10	1 = Operator can activate the function monitoring time (bits 8 & 9)
11	1 = Operator can enable function simulation
12	1 = Operator can enable the maintenance release function
13 - 31	Not allocated

Configurable reactions with the Feature I/O

You can find an overview of all reactions provided by the Feature parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)
1	Reaction to the out of service mode (Page 199)
2	Resetting the commands for changing the mode (Page 193)
3	Enabling resetting of commands for the control settings (Page 193)
4	Setting switch or button mode (Page 195)
9	Resetting via input signals in the event of interlocks or errors (Page 194)
10	Exiting local mode (Page 199)
17	Enabling bumpless changeover to automatic mode for valves, motors, and dosers (Page 197)
22	Update acknowledgment and error status of the message call (Page 193)

In pushbutton operation (Bit 4 = 0) the automatic commands in automatic mode are saved, in other words Pos0Aut, Pos1Aut and Pos2Aut can be reset to zero after switching to the selected position. In manual and local modes, however, the automatic commands are not saved and in the absence of automatic commands the position is tracked.

In switch operation (Bit 4 = 1), positions 1 and 2 are selected by static signals via inputs Pos1Aut and Pos2Aut. If inputs Pos1Aut and Pos2Aut are not set, the block changes to position 0. There is no need to control it with Pos0Aut.

Displaying auxiliary values

This block provides the standard function Displaying auxiliary values (Page 52).

Time stamp

This block receives a time stamp value via the `EventTSIn` input parameter. Refer to EventTs functions (Page 997) for more information on this.

SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 122).

See also

Description of Vlv2WayL (Page 714)

Vlv2WayL messaging (Page 727)

Vlv2WayL I/Os (Page 729)

Vlv2WayL block diagram (Page 737)

Vlv2WayL error handling (Page 725)

Vlv2WayL modes (Page 718)

Feedbacks (Page 83)

Time stamp (Page 51)

5.5.4 Vlv2WayL error handling

Vlv2WayL troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers
- Mode changeover error

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Predefined value when inserting the block, the block is not processed.
0	There is no error.
41	The value for the connection `LocalSetting` is not within the approved limit from 0 to 4.
42	`LocalSetting = 0` and `LocalLi = 1`

51	For `ModLiOp = 1`:
	`- AutModLi = 1` and `ManModLi = 1`
	If local mode is active:
	`- Pos0Local = 1` and `Pos1Local = 1`
	`- Pos0Local = 1` and `Pos2Local = 1`
	`- Pos1Local = 1` and `Pos2Local = 1`
	If automatic mode is active:
	`- Pos0Aut = 1` and `Pos1Aut = 1`
	`- Pos0Aut = 1` and `Pos2Aut = 1`
	`- Pos1Aut = 1` and `Pos2Aut = 1`
	Generally:
	`Pos0Force = 1` and `Pos1Force = 1`
	`Pos0Force = 1` and `Pos2Force = 1`
	`Pos1Force = 1` and `Pos2Force = 1`

52	`LocalAct = 1` and `LocalSetting = 2` or 4 and `SimOn = 1`

Mode changeover error

This error can be output by the block, see the section Error handling (Page 117).

Invalid input signals

This error can be output by the block, see the section Error handling (Page 117).

See also

Vlv2WayL block diagram (Page 737)
Vlv2WayL I/Os (Page 729)
Vlv2WayL messaging (Page 727)
Vlv2WayL functions (Page 720)
Vlv2WayL modes (Page 718)
Description of Vlv2WayL (Page 714)

5.5.5 Vlv2WayL messaging

Messaging

The following messages can be generated for this block:

- Control system fault
- Instance-specific messages

Control system fault

The following control system error messages can be output:

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 1	AS process control message - fault	$$BlockComment$$ Position 0 feedback error (fail-safe position)
	SIG 2	AS process control message - fault	$$BlockComment$$ Positions 1 or 2 feedback error
	SIG 3	AS process control message - fault	$$BlockComment$$ External error has occurred

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter CSF. If this signal changes to `CSF = 1`, a control system error is triggered (`MsgEvId1`, SIG 3).

Instance-specific messages

You can use up to three instance-specific messages with this block.

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 4	AS process control message - fault	$$BlockComment$$ External message 1
	SIG 5	AS process control message - fault	$$BlockComment$$ External message 2
	SIG 6	AS process control message - fault	$$BlockComment$$ External message 3

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Associated values for message instance `MsgEvId1`

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`
4	`ExtVa104`
5	`ExtVa105`
6	`ExtVa106`
7	`ExtVa107`
8	`ExtVa108`
9	Reserved
10	Reserved

The associated values 4 to 8 are allocated to the parameters `ExtVa104` ... `ExtVa108` and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

See also

Description of Vlv2WayL (Page 714)

Vlv2WayL functions (Page 720)

Vlv2WayL I/Os (Page 729)

Vlv2WayL block diagram (Page 737)

Vlv2WayL error handling (Page 725)

Vlv2WayL modes (Page 718)

5.5.6 Vlv2WayL I/Os

Vlv2WayL I/Os

Input parameters

Parameter	Description	Type	Default
AutModLi	1 = Auto mode via: Interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AutModOp	1 = Automatic mode via operator (controlled by ModLiOp = 0)	BOOL	0
BatchEn	1 = Enable allocation	BOOL	0
BatchID	Batch ID	DWORD	16#00000000
BatchName	Batch name	STRING[32]	''
BypProt	1 = Bypassing interlock in local mode and simulation	BOOL	0
CSF	1 = External error (control system error)	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
DefPos1	Output signal parameter setting for position 1	INT	3
DefPos2	Output signal parameter setting for position 2	INT	6
EN	1 = Called block will be processed	BOOL	1
EventTSIn	To wire the signal status of the EventTs message block; The EventTsIn input parameter is used to interconnect to the EventTsOut output parameter of the EventTs block. When this interconnection is configured, the messages of the EventTs block are displayed on the OS in the message view of the technological block and can also be acknowledged there.	STRUCT • Value: BYTE • ST: BYTE	- • 16#00 • 16#80
ExtMsg1	Binary input for freely selectable message 1	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
ExtMsg2	Binary input for freely selectable message 2	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
ExtMsg3	Binary input for freely selectable message 3	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
ExtVa104	Associated value 4 for messages (MsgEvID1)	ANY	
ExtVa105	Associated value 5 for messages (MsgEvID1)	ANY	
ExtVa106	Associated value 6 for messages (MsgEvID1)	ANY	
ExtVa107	Associated value 7 for messages (MsgEvID1)	ANY	
ExtVa108	Associated value 8 for messages (MsgEvID1)	ANY	
FbkP0	1 = Feedback for position 0	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
FbkV0	1 = Feedback for control output CtrlV0	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
FbkV1	1 = Feedback for control output CtrlV1	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
FbkV2	1 = Feedback for control output CtrlV2	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
Features	I/O for additional functions (Page 720)	STRUCT • Bit:0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
Intlock	0 = Interlocking without reset is effective; you can operate the block without reset once the interlocking condition has disappeared 1 = Good state	STRUCT • Value:BOOL • ST:BYTE	- • 1 • 16#FF
Intl_En	1 = Interlock without reset (interlock, Intlock parameter) is active	BOOL	1
LocalLi	1 = Activate local mode via plant signals	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
LocalOp	1 = Local mode by operator	BOOL	0
LocalSetting	Properties for the Local mode (Page 36)	INT	0
ManModLi	1 = Manual mode via interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
ManModOp	1 = Manual mode by OS operator (controlled by ModLiOp = 0)	BOOL	1

Parameter	Description	Type	Default
ModLiOp	Switchover of operating mode between: • 0 = Operator • 1 = Interconnection or SFC	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
Monitor	1 = Feedback monitoring	BOOL	1
MonSafePos	1 = Position 0 (safe position) for monitoring errors	BOOL	1
MonTiP0Dynamic	Monitoring time for position 0 after operation in [s]	REAL	3.0
MonTiV0Dynamic	Monitoring time for feedback errors FdbV0 after operation in [s]	REAL	3.0
MonTiV1Dynamic	Monitoring time for feedback errors FdbV1 after operation in [s]	REAL	3.0
MonTiV2Dynamic	Monitoring time for feedback errors FdbV2 after operation in [s]	REAL	3.0
MonTiStatic	Monitoring time for feedback errors without operation in [s]	REAL	3.0
MS_RelOp	1= Maintenance release via OS operator	BOOL	0
MsgEvId1	Message number (assigned automatically)	DWORD	16#FFFFFFFF
MsgLock	1 = Suppress process messages	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
NoFbkP0	1 = Feedback for position 0 not present	BOOL	0
NoFbkV0	1 = Feedback for control output CtrlV0 not present	BOOL	0
NoFbkV1	1 = Feedback for control output CtrlV1 not present	BOOL	0
NoFbkV2	1 = Feedback for control output CtrlV2 not present	BOOL	0
Occupied	1 = In use by a batch	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC (0-1 edge transition)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OS_Perm	I/O for operator control permissions (Page 720)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
Permit	1 = Enable for opening / closing from safe position 0 = Valve activation not enabled on OS	STRUCT • Value:BOOL • ST:BYTE	- • 1 • 16#FF
Perm_En	1 = Activation enable (enable, Permit parameter) is active	BOOL	1

Parameter	Description	Type	Default
Pos0Aut	1 = Select position 0 in automatic mode	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
Pos0Force	1 = Force position 0	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
Pos0Local	1 = Select position 0 in local mode	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
Pos0Man	1 = Select position 0 in manual mode	BOOL	0
Pos1Aut	1 = Select position 1 in automatic mode	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
Pos1Force	1 = Force position 1	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
Pos1Local	1 = Select position 1 in local mode	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
Pos1Man	1 =Select position 1 in manual mode	BOOL	0
Pos2Aut	1 = Select Position 2 in automatic mode	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
Pos2Force	1 = Force position 2	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
Pos2Local	1 = Select Position 2 in local mode	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
Pos2Man	1 = Select Position 2 in manual mode	BOOL	0
Protect	0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block 1 = Good state	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#FF
Prot_En	1 = Protective interlock (protection, Protect parameter) is active	BOOL	1
PulseWidth	Pulse width of control signal [s]	REAL	3.0
RstLi	1 = Reset via interconnection	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
RstOp	1 = Reset via operator	BOOL	0

Parameter	Description	Type	Default
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
SafeV0	1= Open, 0 = Closed: Safe position for valve V0 (CtrlV0)	BOOL	0
SafeV1	1= Open, 0 = Closed: Safe position for valve V1 (CtrlV1)	BOOL	0
SafeV2	1= Open, 0 = Closed: Safe position for valve V2 (CtrlV2)	BOOL	0
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SimOn	1 = Simulation on	BOOL	0
SelFp1	Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
StepNo	Batch step number	DWORD	16#00000000
UserAna1	Analog auxiliary value 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
UA1unit	Unit of measure for analog auxiliary value 1	INT	0
UserAna2	Analog auxiliary value 2	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
UA2unit	Unit of measure for analog auxiliary value 2	INT	0
UserStatus	Freely assignable user area for status word (bit 24 to bit 31)	BYTE	16#00
WarnTiAut	Prewarning for valve movement into position 1 or position 2 in automatic mode in [s]	REAL	0.0
WarnTiMan	Prewarning for valve movement into position 1 or position 2 in manual mode in [s]	REAL	0.0

Output parameters

Parameter	Description	Type	Default
AutAct	1 = Auto mode active 0 = Manual mode is active	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
CtrlV0	Control output V0	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
CtrlV1	Control output V1	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
CtrlV2	Control output V2	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see Vlv2WayL error handling (Page 725)	INT	-1
FbkP0Out	Feedback from position 0	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
FbkV0Out	Feedback from control output CtrlV0	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
FbkV1Out	Feedback from control output CtrlV1	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
FbkV2Out	Feedback from control output CtrlV2	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
LocalAct	1 = Local mode active	STRUCT • Value:BOOL • ST:BYTE	0- • 0 • 16#80
LockAct	1 = Interlock (Inlock, Permit or Protect) is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ManAct	1 = Manual mode active	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
MonDynP0	1 = Feedback error for position 0 due to a control change	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
MonDynV0	1 = Feedback error for FdbV0 due to control change	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
MonDynV1	1 = Feedback error for FdbV1 due to control change	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
MonDynV2	1 = Feedback error for FdbV2 due to control change	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
MonStaP0	1 = Feedback error for position 0 due to an unexpected feedback change	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
MonStaV0	1 = Feedback error for FdbV0 due to unexpected feedback change	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
MonStaV1	1 = Feedback error for FdbV1 due to unexpected feedback change	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
MonStaV2	1 = Feedback error for FdbV2 due to unexpected feedback change	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
MS_Release	Maintenance release: 1 = Release for OS operator	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
MsgAckn1	Alarm acknowledgement status 1 (output STATUS of first ALARM_8P)	WORD	16#0000
MsgErr1	Alarm error 1 (output ERROR of first ALARM_8P)	BOOL	0
MsgStat1	Alarm status 1 (output ERROR of first ALARM_8P)	WORD	16#0000
OosAct	1 = Block is out of service	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
P_CtrlP0	Pulse output from position 0	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
P_CtrlV0	Pulse output V0	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
P_CtrlV1	Pulse output V1	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
P_CtrlV2	Pulse output V2	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80
Pos0Out	1 = Position 0 is active	BOOL	0
Pos1Out	1 = Position 1 is active	BOOL	0
Pos2Out	1 = Position 2 is active	BOOL	0
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word 1 (Page 714)	DWORD	16#00000000
Status2	Status word 2 (Page 714)	DWORD	16#00000000
Status3	Status word 3 (Page 714)	DWORD	16#00000000
Status4	Status word 4 (Page 714)	DWORD	16#00000000
WarnAct	1 = Prewarning for valve movement to position 1 or 2 active (parameters WarnTiAut and WarnTiMan)	STRUCT • Value:BOOL • ST:BYTE	- • 0 • 16#80

See also

Vlv2WayL messaging (Page 727)

Vlv2WayL block diagram (Page 737)

Vlv2WayL modes (Page 718)

5.5.7 Vlv2WayL block diagram

Vlv2WayL block diagram

A block diagram is not provided for this block.

See also

Vlv2WayL I/Os (Page 729)
Vlv2WayL messaging (Page 727)
Vlv2WayL error handling (Page 725)
Vlv2WayL functions (Page 720)
Vlv2WayL modes (Page 718)
Description of Vlv2WayL (Page 714)

5.5.8 Operator control and monitoring

5.5.8.1 Vlv2WayL views

Views of the Vlv2WayL block

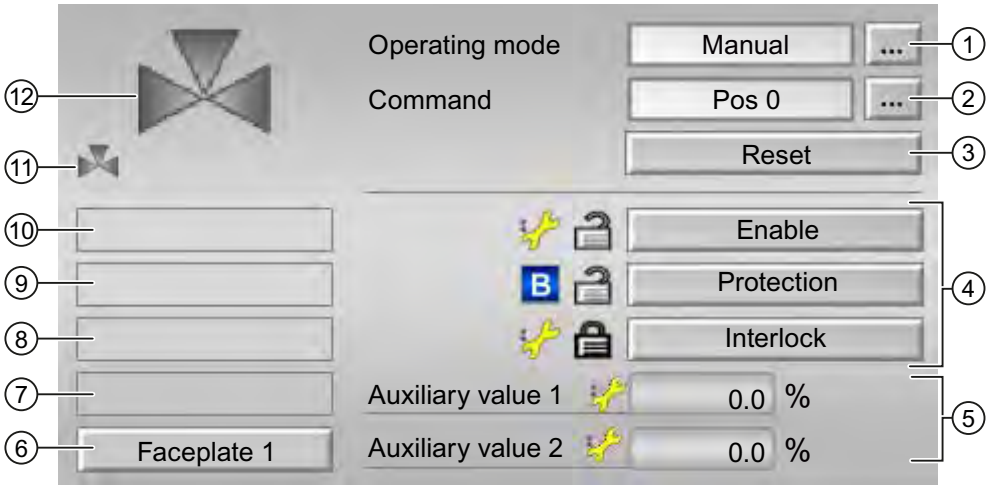
The block Vlv2WayL provides the following views:

- Vlv2WayL standard view (Page 738)
- Message view (Page 169)
- Trend view (Page 172)
- Vlv2WayL parameter view (Page 741)
- Vlv2WayL preview (Page 743)
- Memo view (Page 171)
- Batch view (Page 170)
- Block symbol for Vlv2WayL (Page 746)

Refer to the sections Structure of the faceplate (Page 123) and Block icon structure (Page 174) for general information on the faceplate and block icon.

5.5.8.2 Vlv2WayL standard view

Vlv2WayL standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 32)
- Automatic mode (Page 32)
- Local mode (Page 36)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Select position for 2-way valve

This area shows you the default operating state for the valve. The following states can be shown and executed here:

- Pos0
- Pos1
- Pos2

Refer to the Switching operating states and operating modes (Page 127) section for information on changing the state.

(3) Resetting the block

Click "Reset" for interlocks or errors. You can find additional information on this in the Resetting the block in case of interlocks or errors (Page 119) section.

(4) Display area for interlock functions of the block

You can use this button to control the interlock functions of the block. You can find additional information on this in the Interlocking functions (Page 100) section.

The following is displayed in addition to the buttons:

- Interlock status (see Forming the group status for interlock information (Page 105)), e.g.:

- Signal status (see Forming and outputting signal status for blocks (Page 88)), e.g.:

  If there is a bypass of one of the interlock signals, the symbol for the bypass is shown instead of the signal status.

- Bypass information:

  If there is a bypass, it is displayed instead of the signal status.

(5) Display of auxiliary values

You can use this area to display two auxiliary values that have been configured in the engineering system. You can find additional information on this in the Displaying auxiliary values (Page 52) section.

(6) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

(7) Display area for states of the block

This area provides additional information on the operating state of the block:

- Maintenance

Additional information on the display area for states of the block is available in section Maintenance release (Page 47).

(8) Display area for states of the block

This area provides additional information on the operating state of the block:

- Simulation

You can find additional information on this in the Simulating signals (Page 93) section.

(9) Display area for states of the block

This area provides additional information on the operating state of the block:

- Runtime error
- Control error
- Invalid signal
- Mode switch fail

Additional information on these errors is available in the sections Monitoring the feedbacks (Page 83) , Error handling (Page 117) (section "Invalid input signals" and "Mode changeover error") and Trip function (Page 102).

(10) Display area for states of the block

This area provides additional information on the operating state of the block:

- Force Pos0
- Force Pos1
- Force Pos2

You can find additional information on this in the Forcing operating states (Page 115) section.

(11) Automatic preview

This display is only visible in manual mode or local mode when the current output signals are not identical to the control in automatic mode.

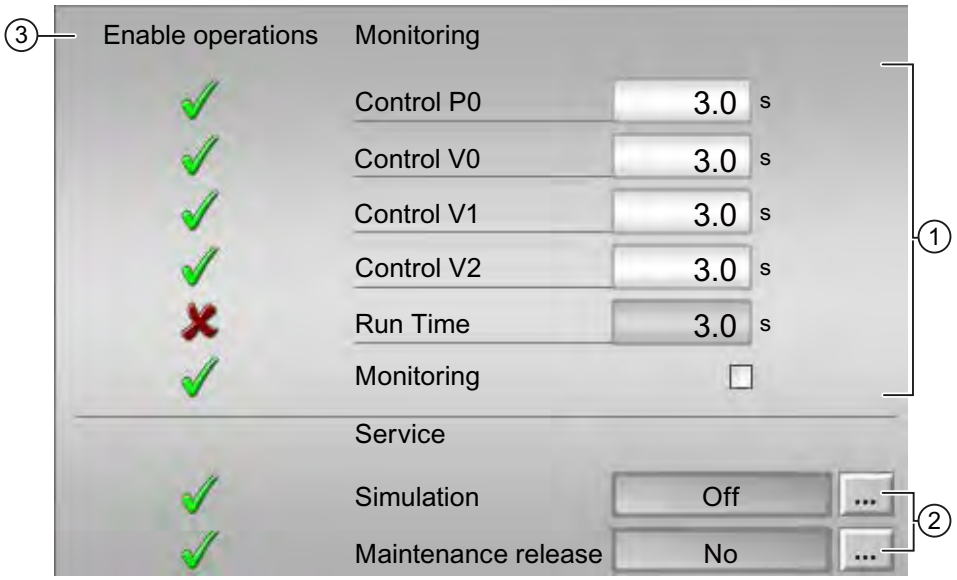
The display shows what state the valve would assume if you switched to automatic mode.

(12) Status display of valve

You can find additional information on this in the Block symbol for Vlv2WayL (Page 746) section.

5.5.8.3 Vlv2WayL parameter view

Vlv2WayL parameter view



(1) Monitoring

In this area, you change parameters and therefore influence the valve. Refer to the Changing values (Page 129) section for more on this.

You can influence the following parameters:

- Control V0: Monitoring time while opening/closing the valve
- Control V1: Monitoring time while opening/closing the valve
- Control V2: Monitoring time while opening/closing the valve
- Runtime: Monitoring time for maintaining the valve position

Enable monitoring

You can enable monitoring by clicking the check box (☑)

You can find additional information on this in the Monitoring the feedbacks (Page 83) section.

(2) Service

You can select the following functions in this area:

- Simulation
- Maintenance release

Refer to the Switching operating states and operating modes (Page 127) section for more on this.

(3) Enable operations

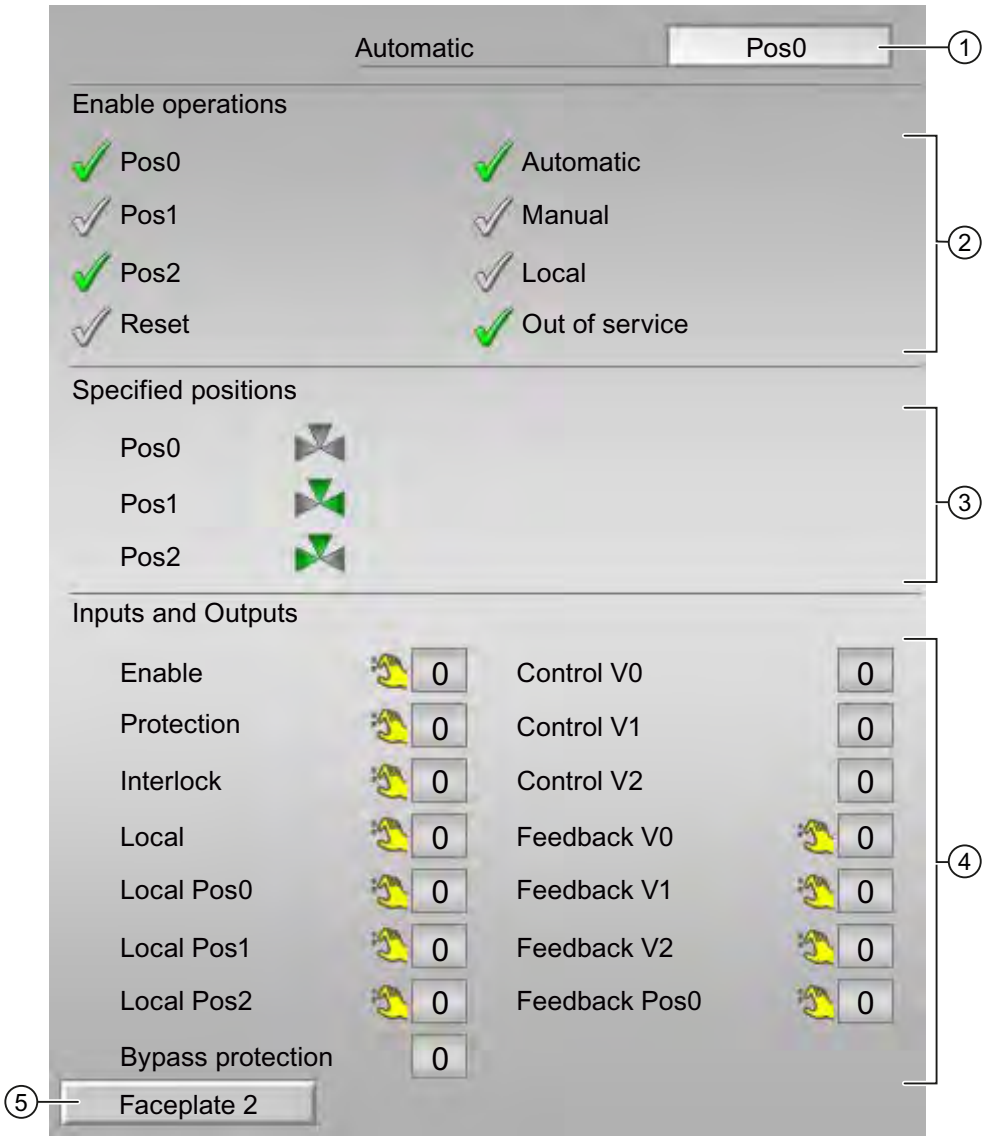
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions.

5.5.8.4 Vlv2WayL preview

Vlv2WayL preview



(1) Automatic preview

This area shows you the status of a block after its has switched to automatic mode.

If the block is in automatic mode, the current block state is displayed.

(2) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions

The following operator control enables are shown here:

- Pos0: You can set the valve to position 0
- Pos1: You can set the valve to position 1
- Pos2: You can set the valve to position 2
- Reset: You can reset the valve if errors occur.
- Automatic: You can switch to automatic mode.
- Manual: You can switch to manual mode
- Local: You can switch to local mode.
- Out of service: You can switch to out of service mode.

(3) Specified position

Preview of the valve positions, as configured in the engineering system (ES).

(4) Display current control signal

This area shows the most important parameters for this block with the current selection:

- Enable:
  - 0 = Valve activation not enabled on OS
  - 1 = Enable for starting / stopping from safe position
- Protection:
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = Good state
- Interlock:
  - 0 = Interlocking without reset is effective; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = Good state
- Local: 1= Block is controlled in local mode
- Local Pos0: 1 = Block has been set to position 0 in local mode
- Local Pos1: 1 = Block has been set to position 1 in local mode
- Local Pos2: 1 = Block has been set to position 2 in local mode
- Bypass protection:
  - 0 = Bridging deactivated
  - 1 = Bypassing interlock in local mode and simulation
- Control V0: 1 = Control signal for valve 0
- Control V1: 1 = Control signal for valve 1
- Control V2: 1 = Control signal for valve 2
- Feedback V0: 1 = Feedback when valve 0 is opened
- Feedback V1: 1 = Feedback when valve 1 is opened
- Feedback V2: 1 = Feedback when valve 2 is opened
- Feedback Pos0: 1 = Valve is in Pos0

(5) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

5.5.8.5 Block symbol for Vlv2WayL

Properties of the Vlv2WayL block symbol

A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as the control system fault CSF
- Operating modes
- Signal status, maintenance release
- Forcing states
- Displays for bridging interlocks
- Interlocks
- Memo display
- Valve status display

Icons	Selection of the block icon in CFC	Special features
	1	Block icon in the full display
	2	
	3	
	4	
	5	
	6	
	7	

Icons	Selection of the block icon in CFC	Special features
	8	
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181).

Valve status display

The following valve states are shown here:

Icon	Meaning
	Valve open
	Error at valve
	Valve is opening
	Valve is closing

5.6 VlvL - Valve

5.6.1 Description of VlvL

Object name (type + number) and family

Type + number: FB 1899

Family: Drives

Area of application for VlvL

The block is used for the following applications:

- Controlling a valve in two positions (open/closed) with adjustable safe position

How it works

The valve is opened or closed by a control signal. The signal 0 corresponds to the de-energized state (safe position) of the valve.

The control is monitored by the open/closed position (feedback) signals. The block can derive missing feedback from the control.

Various inputs are available for control purposes. The next sections provide additional detailed information on configuration, operating principles, visualization and operation.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

For the VlvL block, the Advanced Process Library contains a template for process tag types as an example with an application scenario for this block.

Example of process tag types:

- Valve (ValveLean) (Page 1453)

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

Status word allocation for `Status1` parameter

For a description of the individual parameters, see the VlvL I/Os (Page 761) section.

Status bit	Parameter
0	`Occupied`
1	`BatchEn`
2	`SimOn`
3	`OosAct.Value`
4	`OosLi.Value`
5	`AutoAct.Value`
6	`LocalAct.Value`
7	`LockAct.Value`
8	Open/closed command (1 = Open)
9	`FbkOpenOut.Value`
10	`FbkCloseOut.Value`
11	Feedback error without control change
12	Feedback error due to control change
13	`BypProt`
14	Invalid signal status
15	Mode changeover error
16	`1 = Intlock` is active
17	`1 = Permit` is active
18	`1 = Protect` is active
19	`OpenForce.Value`
20	`CloseForce.Value`
21	`Force`
22	Automatic preview (1=open)
23	Bumpless changeover to automatic mode is active
24	`SafePos`
25	`UserAna1` interconnected
26	`UserAna2` interconnected
27-31	Not used

Status word allocation for `Status2` parameter

Status bit	Parameter
0	`MsgLock`
1	Not used
2	Display for interlocks in block icon
3-18	Not used
19	1 = No impact of input signals on local mode with `LocalSetting = 2` and `LocalSetting = 4`
20	1 = Valve open
21	1 = Valve closed
22	1 = Valve opening
23	1 = Valve closing
24-29	Not used
30	Bypass information from previous function block
31	`MS_RelOp`

Status word allocation for `Status3` parameter

Status bit	Parameter
0	Effective `Signal1` of the message block connected via `EventTsIn`
1	Effective `Signal2` of the message block connected via `EventTsIn`
2	Effective `Signal3` of the message block connected via `EventTsIn`
3	Effective `Signal4` of the message block connected via `EventTsIn`
4	Effective `Signal5` of the message block connected via `EventTsIn`
5	Effective `Signal6` of the message block connected via `EventTsIn`
6	Effective `Signal7` of the message block connected via `EventTsIn`
7	Effective `Signal8` of the message block connected via `EventTsIn`
8	"Interlock" button is enabled
9	"Permission" button is enabled
10	"Protection" button is enabled
11 - 25	Not used
26	Show automatic preview in the standard view
27 - 31	Not used

See also

VlvL functions (Page 754)
VlvL messaging (Page 759)
Overview of the modes (Page 25)
VlvL block diagram (Page 768)
VlvL error handling (Page 758)
VlvL modes (Page 752)
Resetting the block in case of interlocks or errors (Page 119)

5.6.2 VlvL modes

VlvL operating modes

The block supports all standard modes:

- Local mode (Page 36)
- Automatic mode (Page 32)
- Manual mode (Page 32)
- Out of service (Page 27)

The next section provides additional block-specific information relating to the general descriptions.

"Local mode"

You can find general information on "Local mode", switching modes and bumpless switchover in the Local mode (Page 36) section.

Valve actions you can control in local mode:

- Open (`OpenLocal = 1`)
- Close (`CloseLocal = 1`)

A block operated in local mode is controlled either by local signals or by feedback signals (input parameters `FbkOpen` and `FbkClose`; if no position can be assigned, the last valid position is accepted). Configuration takes place via the input parameter `LocalSetting`.

"Automatic mode"

You can find general information on "Automatic mode", switching modes and bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 32) section.

Valve actions you can control in auto mode:

- Open (`OpenAut = 1`)
- Close (`CloseAut = 1`)

"Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 32) section.

Valve actions you can control in manual mode:

- Open (`OpenMan = 1`)
- Close (`CloseMan = 1`)

"Out of service"

You can find general information about the "Out of service" mode in the Out of service (Page 27) section.

See also

Description of VlvL (Page 748)

VlvL block diagram (Page 768)

VlvL I/Os (Page 761)

VlvL messaging (Page 759)

VlvL error handling (Page 758)

VlvL functions (Page 754)

5.6.3 VlvL functions

Functions of VlvL

The functions for this block are listed below.

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

Operator control permissions via parameter OS_Perm

The block has the following Operator permissions (Page 45) for the OS_Perm parameter:

Bit	Function
0	1 = Operator can switch to automatic mode
1	1 = Operator can switch to manual mode
2	1 = Operator can switch to local mode
3	1 = Operator can switch to out of service mode
4	1 = Operator can open the valve
5	1 = Operator can close the valve
6	1 = Operator can reset the valve
7	1 = Operator can define the monitoring time for startup
8	1 = Operator can define the monitoring time for runtime
9	1 = Operator can activate the function monitoring time (bits 7 & 8)
10	1 = Operator can enable function simulation
11	1 = Operator can enable the maintenance release function
12 - 31	Not allocated

Interlocks

This block provides the following interlocks:

- Activation enable
- Interlock without reset (interlock)
- Interlock with reset (protection)

Refer to the Interlocks (Page 100) section for more on this.

Disabling interlocks

This block provides the standard function Disabling interlocks (Page 107).

Resetting the block in case of interlocks

This block provides the standard function Resetting the block in case of interlocks or errors (Page 119).

Forming the group status for interlock information

This block provides the standard function Forming the group status for interlock information (Page 105).

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status for the block is formed by the following parameters:

- Feedback `FbkOpenOut.ST`
- Feedback `FbkCloseOut.ST`
- Enable `Permit.ST`
- Interlock without reset `Inlock.ST`
- Interlock with reset `Protect.ST`

Forcing operating states

This block provides the standard function Forcing operating states (Page 115). Inputs `OpenForce` and `CloseForce` force the block to open or close.

Feedback monitoring

This block provides the standard function Monitoring the feedbacks (Page 83). Startup behavior is monitored by setting parameter`MonTiDynamic`. The parameter `MonTiStatic` monitors compliance with the position.

Maintenance release

This block provides the standard function Maintenance release (Page 47).

Specify warning times for control functions

This block provides the standard function Specifying warning times for control functions at motors and valves (Page 122). The warning signal is output before the valve moves away from the safe position. No signal is output for movement to the safe position.

Simulating signals

This block provides the standard function Simulating signals (Page 93).

Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 53).

Safe position

This block provides the standard function Safe position for motors, valves and controllers (Page 120). The safe position (de-energized state) is set using the `SafePos` parameter.

- `SafePos = 0` means that at `Ctrl = 0` the valve drive closes and at`Ctrl = 1` it opens (de-energized state is "closed")
- `SafePos = 1` means that at `Ctrl = 0` the valve drive opens and at`Ctrl = 1` it closes(de-energized state is "open")

Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 46).

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)
1	Reaction to the out of service mode (Page 199)
2	Resetting the commands for changing the mode (Page 193)
3	Enabling resetting of commands for the control settings (Page 193)
4	Setting switch or button mode (Page 195)
9	Resetting via input signals in the event of interlocks or errors (Page 194)
10	Exiting local mode (Page 199)
17	Enabling bumpless changeover to automatic mode for valves, motors, and dosers (Page 197)
22	Update acknowledgment and error status of the message call (Page 193)

In pushbutton operation (Bit 4 = 0) the automatic commands in automatic mode are saved, in other words `OpenAut` and `CloseAut` can be reset to zero after changing the control. In manual and local modes, however, the automatic commands are not saved and in the absence of automatic commands the automatic control is tracked.

In switch operation (Bit 4 = 1), control is selected with the static signal `OpenAut`. If input `OpenAut` is not set the valve is closed. Control via `CloseAut` is not needed. If, in addition, the function "Activate command reset for control" (Bit 3 = 1) is switched on, than after evaluation in the block the input `OpenAut` is reset to the safe position.

Displaying auxiliary values

This block provides the standard function Displaying auxiliary values (Page 52).

SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 122).

Output signal as a pulse signal or static signal

This block provides the standard function Output signal as a static signal or pulse signal (Page 86). In addition to the static control output `Out`, the block also has pulse outputs `P_Open`, `P_Close`, which are dependent on the static control output.

Time stamp

This block receives a time stamp value via the `EventTSIn` input parameter. Refer to EventTs functions (Page 997) for more information on this.

Disabling feedbacks

This block provides the standard function Disabling feedback for valves (Page 85). Feedback monitoring can be deactivated separately for each feedback with `NoFbkOpen` or `NoFbkClose` as required.

See also

Description of VlvL (Page 748)

VlvL messaging (Page 759)

VlvL I/Os (Page 761)

Local mode (Page 36)

Manual and automatic mode for motors, valves and dosers (Page 32)

Error handling (Page 117)

VlvL modes (Page 752)

VlvL error handling (Page 758)

VlvL block diagram (Page 768)

Time stamp (Page 51)

5.6.4 VlvL error handling

VlvL troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers
- Mode changeover error
- Invalid input signals

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Default value when implementing the block; block will not be processed.
0	There is no error.
41	The value for the connection `LocalSetting` is not within the approved limit from 0 to 4.
42	`LocalSetting = 0` or `LocalSetting = 3` or `LocalSetting = 4` and `LocalLi = 1`

51	`AutModLi = 1` and `ManModLi = 1`
	`OpenLocal = 1` and `CloseLocal = 1`
	`OpenAut = 1` and `CloseAut = 1`
	`OpenForce = 1` and `CloseForce = 1`

52	`LocalAct = 1` and `LocalSetting = 2` or 4 and `SimOn = 1`

Mode changeover error

This error can be output by the block, see chapter Error handling (Page 117).

Invalid input signals

This error can be output by the block, see the section Error handling (Page 117).

See also

Description of VlvL (Page 748)

VlvL modes (Page 752)

VlvL block diagram (Page 768)

VlvL I/Os (Page 761)

VlvL messaging (Page 759)

VlvL functions (Page 754)

5.6.5 VlvL messaging

Messaging

The following messages can be generated for this block:

- Control system fault
- Instance-specific messages

Control system fault

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 1	AS process control message - fault	$$BlockComment$$ Feedback error
	SIG 2	AS process control message - fault	$$BlockComment$$ External error has occurred

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter CSF. If this signal changes to CSF = 1, a control system error is triggered (MsgEvId1, SIG 2).

Instance-specific messages

You can use up to three instance-specific messages with this block.

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 3	AS process control message - fault	$$BlockComment$$ External message 1
	SIG 4	AS process control message - fault	$$BlockComment$$ External message 2
	SIG 5	AS process control message - fault	$$BlockComment$$ External message 3

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Associated values for message instance `MsgEvId1`

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`
4	`ExtVa104`
5	`ExtVa105`
6	`ExtVa106`
7	`ExtVa107`
8	`ExtVa108`
9	Reserved
10	Reserved

The associated values 4 to 8 are allocated to the parameters `ExtVa104` ... `ExtVa108` and you can use them. See the "Process Control System PCS 7 - Engineering System" manual.

See also

VlvL modes (Page 752)

VlvL block diagram (Page 768)

VlvL error handling (Page 758)

Time stamp (Page 51)

5.6.6 VlvL I/Os

VlvL I/Os

Input parameters

Parameter	Description	Type	Default
AutModLi	1 = Automatic mode via interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AutModOp	1 = Automatic mode via operator (controlled by ModLiOp = 1)	BOOL	0
BatchEn	1 = Enable allocation	BOOL	0
BatchID	Batch ID	DWORD	16#00000000
BatchName	Batch name	STRING[32]	''
BypProt	1 = Bypassing interlock is active in local mode and simulation	BOOL	0
CloseAut	1 = Select "Close valve" in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
CloseForce	1 = Force valve closure	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
CloseLocal	1 = Select "Close valve" in local mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
CloseMan	1 = Select "Close valve" in manual mode	BOOL	0
CSF	1 = External error (control system error)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
EN	1 = Called block will be processed	BOOL	1
EventTSIn	To wire the signal status of the EventTs message block; The EventTsIn input parameter is used to interconnect to the EventTsOut output parameter of the EventTs block. When this interconnection is configured, the messages of the EventTs block are displayed on the OS in the message view of the technological block and can also be acknowledged there.	STRUCT • Value: BYTE • ST: BYTE	- • 16#00 • 16#80

Parameter	Description	Type	Default
ExtMsg1	Binary input for freely selectable message 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg2	Binary input for freely selectable message 2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg3	Binary input for freely selectable message 3	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtVa104	Associated value 4 for messages (MsgEvID1)	ANY	
ExtVa105	Associated value 5 for messages (MsgEvID1)	ANY	
ExtVa106	Associated value 6 for messages (MsgEvID1)	ANY	
ExtVa107	Associated value 7 for messages (MsgEvID1)	ANY	
ExtVa108	Associated value 8 for messages (MsgEvID1)	ANY	
FbkOpen	1 = Valve open feedback signal	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FbkClose	1 = Valve closed feedback signal	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Feature	I/O for additional functions (Page 754)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
Intlock	0 = Interlocking without reset is effective; you can operate the block without reset once the interlocking condition has disappeared 1 = Good state	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#FF
Intl_En	1 = Interlock without reset (interlock, Intlock parameter) is active	BOOL	1
LocalLi	1 = Activate local mode via plant signal	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
LocalOp	1 = Local mode by operator	BOOL	0
LocalSetting	Properties for the Local mode (Page 36)	INT	0

Parameter	Description	Type	Default
ManModLi	1 = Manual mode via interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ManModOp	1 = Manual mode via: OS operator (controlled by ModLiOp = 0)	BOOL	1
ModLiOp	Switchover of operating mode between: • 0 = Operator • 1 = Interconnection or SFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Monitor	1 = Feedback monitoring	BOOL	0
MonSafePos	1 = Safe position in the event of monitoring errors	BOOL	0
MonTiDynamic	Monitoring time after operation in [s]	REAL	3.0
MonTiStatic	Monitoring time for feedback errors without operation in [s]	REAL	3.0
MS_RelOp	1= Maintenance release via OS operator	BOOL	0
MsgEvId1	Message number (assigned automatically)	DWORD	16#FFFFFF
MsgLock	1 = Suppress process messages	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
NoFbkClose	1 = No feedback present for "valve closed"	BOOL	0
NoFbkOpen	1 = No feedback present for "valve open"	BOOL	0
Occupied	1 = In use by a batch	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OpenAut	1 = Select "Open valve" in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OpenForce	1 = Force valve opening	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OpenLocal	1 = Select "Open valve" in local mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OpenMan	1 = Select "Open valve" in manual mode	BOOL	0

Parameter	Description	Type	Default
OS_Perm	I/O for operator control permissions (Page 754)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
Permit	1 = Enable for opening / closing from safe position 0 = Valve activation not enabled on OS	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#FF
Perm_En	1 = Activation enable (enable, Permit parameter) is active	BOOL	1
Protect	0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block 1 = Good state	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#FF
Prot_En	1 = Protective interlock (protection, Protect parameter) is active	BOOL	1
PulseWidth	Pulse width of control signal [s]	REAL	3.0
RstLi	1 = Reset via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RstOp	1 = Reset via operator	BOOL	0
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
SafePos	1= Open, 0 = Closed: Safe position for valve	BOOL	0
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SimOn	1 = Simulation on	BOOL	0
SelFp1	Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
StepNo	Batch step number	DWORD	16#00000000
UserAna1	Analog auxiliary value 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
UA1unit	Unit of measure for analog auxiliary value 1	INT	0
UserAna2	Analog auxiliary value 2	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
UA2unit	Unit of measure for analog auxiliary value 2	INT	0

Parameter	Description	Type	Default
UserStatus	Freely assignable status word	BYTE	16#00
WarnTiAut	Prewarning of valve movement from safe position in automatic mode in [s]	REAL	0.0
WarnTiMan	Prewarning of valve movement from safe position in manual mode in [s]	REAL	0.0

Output parameters

Parameter	Description	Type	Default
AutAct	1 = Automatic mode is active 0 = Manual mode is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Ctrl	Control output	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see VlvL error handling (Page 758)	INT	-1
FbkCloseOut	Valve closed feedback	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FbkOpenOut	Valve open feedback	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
LocalAct	1 = Local mode active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
LockAct	1 = Interlock (Intlock, Permit or Protect) is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ManAct	1 = Manual mode active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MonDynErr	1 = Feedback error due to control change	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MonStaErr	1 = Feedback error due to unexpected feedback change	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_Release	Maintenance release: 1 = Release for OS operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MsgAckn1	Alarm acknowledgement status 1 (output STATUS of first ALARM_8P)	WORD	16#0000
MsgErr1	Alarm error 1 (output ERROR of first ALARM_8P)	BOOL	0

Parameter	Description	Type	Default
MsgStat1	Alarm status 1 (output ERROR of first ALARM_8P)	WORD	16#0000
OosAct	1 = Block is out of service	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
P_Close	Pulse signal to close valve	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
P_Open	1 = Pulse signal to open valve	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word 1 (Page 748)	DWORD	16#00000000
Status2	Status word 2 (Page 748)	DWORD	16#00000000
Status3	Status word 3 (Page 748)	DWORD	16#00000000
WarnAct	1 = Prewarning for valve movement away from safe position active (parameters WarnTiAut and WarnTiMan)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

VlvL messaging (Page 759)

VlvL modes (Page 752)

VlvL block diagram (Page 768)

5.6.7 VlvL block diagram

VlvL block diagram

A block diagram is not provided for this block.

See also

Description of VlvL (Page 748)
VlvL modes (Page 752)
VlvL error handling (Page 758)
VlvL messaging (Page 759)
VlvL I/Os (Page 761)
VlvL functions (Page 754)

5.6.8 Control and monitoring

5.6.8.1 VlvL views

Views of the VlvL block

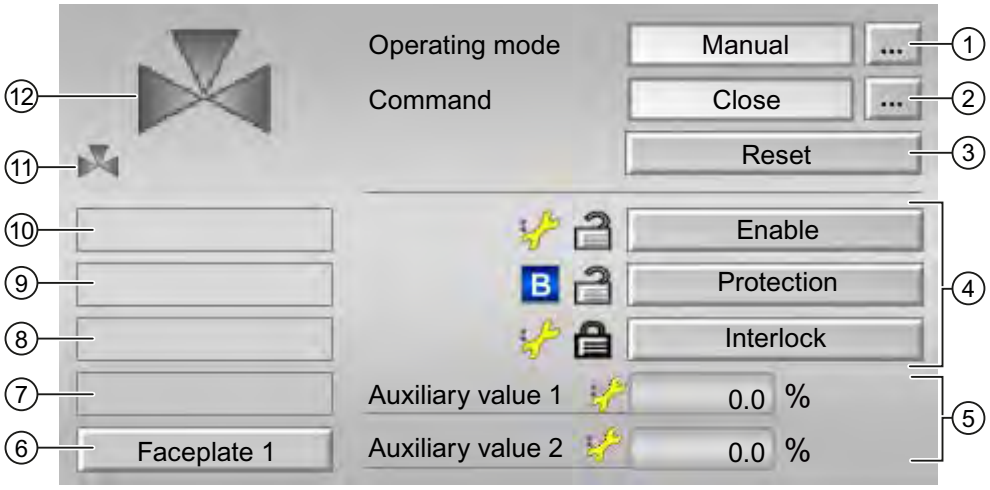
The block VlvL provides the following views:

- VlvL standard view (Page 769)
- Message view (Page 169)
- Trend view (Page 172)
- Parameter view for motors and valves (Page 155)
- VlvL preview (Page 772)
- Memo view (Page 171)
- Batch view (Page 170)
- VlvL block icon (Page 775)

Refer to the sections Structure of the faceplate (Page 123) and Block icon structure (Page 174) for general information on the faceplate and block icon.

5.6.8.2 VlvL standard view

VlvL standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 32)
- Automatic mode (Page 32)
- Local mode (Page 36)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Open and close valve

This area shows you the default operating state for the valve. The following states can be shown and executed here:

- Open
- Close

Refer to the Switching operating states and operating modes (Page 127) section for information on changing the state.

(3) Resetting the block

Click "Reset" for interlocks or errors. You can find additional information on this in the Resetting the block in case of interlocks or errors (Page 119) section.

(4) Display area for interlock functions of the block

You can use this button to control the interlock functions of the block. You can find additional information on this in the Interlocking functions (Page 100) section.

The following is displayed in addition to the buttons:

- Interlock status (see Forming the group status for interlock information (Page 105)), e.g.:

- Signal status (see Forming and outputting signal status for blocks (Page 88)), e.g.:

  If there is a bypass of one of the interlock signals, the symbol for the bypass is shown instead of the signal status.

- Bypass information:

  If there is a bypass, it is displayed instead of the signal status.

(5) Display of auxiliary values

You can use this area to display two auxiliary values that have been configured in the engineering system. You can find additional information on this in the Displaying auxiliary values (Page 52) section.

(6) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

(7) Display area for states of the block

This area provides additional information on the operating state of the block:

- Maintenance

Additional information on the display area for states of the block is available in section Maintenance release (Page 47).

(8) Display area for states of the block

This area provides additional information on the operating state of the block:

- Simulation

You can find additional information on this in the Simulating signals (Page 93) section.

(9) Display area for states of the block

This area provides additional information on the operating state of the block:

- Runtime error
- Control error
- Invalid signal
- Mode switch fail

Additional information on these errors is available in the sections Monitoring the feedbacks (Page 83) , Error handling (Page 117) (section "Invalid input signals" and "Mode changeover error") and Trip function (Page 102).

(10) Display area for states of the block

This area provides additional information on the operating state of the block:

- Force open
- Force close

You can find additional information on this in the Forcing operating states (Page 115) section.

(11) Automatic preview

This display is only visible in manual mode or local mode when the current output signals are not identical to the control in automatic mode.

The display shows what state the valve would assume if you switched to automatic mode.

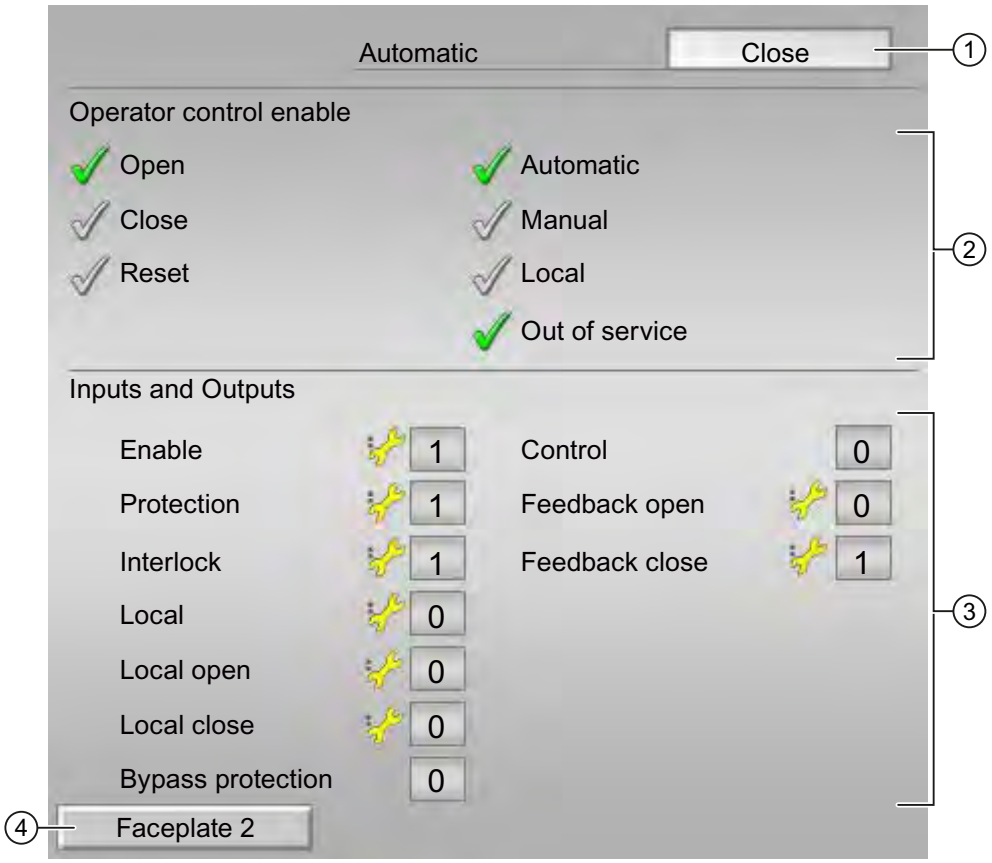
(12) Status display of valve

This area shows if the valve is open or closed:

- Green: Valve is open
- Gray: Valve is closed
- Red: Fault at valve

5.6.8.3 VlvL preview

VlvL preview



(1) Automatic preview

This area shows you the block status after its has switched from manual to automatic mode.

If the block is in automatic mode, the current block state is displayed.

(2) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions

The following operator control enables are shown here:

- Open: You can open the valve.
- Close: You can close the valve.
- Reset: You can reset the valve if interlocks or errors occur.
- Automatic: You can switch to automatic mode.
- Manual: You can switch to manual mode
- Local: You can switch to local mode.
- Out of service: You can switch to out of service mode.

(3) Display current control signal

This area shows the most important parameters for this block with the current selection:

- Enable:
  - 0 = Valve activation not enabled on OS
  - 1 = Enable for opening / closing from safe position
- Protection:
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = Good state
- Interlock:
  - 0 = Interlocking without reset is effective; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = Good state
- Local: 1= Block is controlled in local mode
- Local open: 1 = Open valve in local mode
- Local close: 1 = Close valve in local mode

- Bypass protection:
  - 0 = Bridging deactivated
  - 1 = Bypassing interlock in local mode and simulation
- Control: Display for valve control:
  - 0 = valve is closing
  - 1 = valve is opening
- Feedback for open: 1 = Valve is opened
- Feedback for closed: 1 = Valve is closed

(4) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

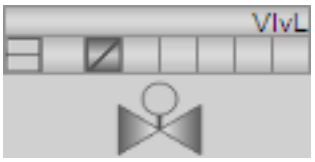
5.6.8.4 VlvL block icon

Block icons for Vlvl

A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as the control system fault CSF
- Operating modes
- Signal status, maintenance release
- Forcing states
- Displays for bridging interlocks
- Interlocks
- Memo display
- Valve status display

Icons	Selection of the block icon in CFC	Special features
	1	Block icon in the full display
	2	
	3	
	4	
	5	
	6	
	7	

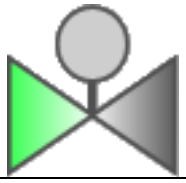
Icons	Selection of the block icon in CFC	Special features
	8	
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181).

Valve status display

The following valve states are shown here:

Icon	Meaning
	Valve open
	Valve closed
	Error at valve
	Valve is opening
	Valve is closing

5.7 VlvMotL - Motor valve

5.7.1 Description of VlvMotL

Object name (type + number) and family

Type + number: FB 1900

Family: Drives

Area of application for VlvMotL

The block is used for the following applications:

- Motor valve control

How it works

Various operating modes are available for controlling the motor-driven valve. This functionality allows you to set specific valve states. All changes of modes or states and faults occurring in this context are monitored, visualized in the faceplate and reported to the operator. Operators with suitable permissions can use the block icon and the faceplate to view the current states of the motor-driven valve and to operate it.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

For the VlvMotL block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Step controller with assigned actuator block and position feedback (StepControlActor) (Page 1437)
- Motor valve (ValveMotor) (Page 1454)

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

Status word allocation for `Status1` parameter

For a description of the individual parameters, see the VlvMotL I/Os (Page 794) section.

Status bit	Parameter
0	`Occupied`
1	`BatchEn`
2	`SimOn`
3	`OosAct.Value`
4	`OosLi.Value`
5	`AutoAct.Value`
6	`LocalAct.Value`
7	`LockAct.Value`
8	`Open.Value`
9	Motor is stopped
10	`Close.Value`
11	Torque shutdown active (`TorqOpen` or `TorqClose = 1`)
12	Not used
13	`V_MonStaErr.Value`
14	`V_MonStaErr.Value`
15	`Mode Switch Fail`
16	`1 = Intlock` is active
17	`1 = Permit` is active
18	`1 = Protect` is active
19	`Trip.Value`
20	`OpenForce.Value`
21	`StopForce.Value`
22	`CloseForce.Value`
23	"Interlock" button is enabled
24 - 25	Not used
26	Bypass information from previous function block
27	Bypass active (`BypProt = 1`) and `Local.Act = 1` or `SimOn = 1`
28	Invalid signal status
29	0 = Closed 1= Open
30	"Permission" button is enabled
31	"Protection" button is enabled

Status word allocation for `Status2` parameter

Status bit	Parameter
0	`MsgLock`
1	`AV_AH_Act.Value`
2	`AV_WH_Act.Value`
3	`AV_TH_Act.Value`
4	`AV_TL_Act.Value`
5	`AV_WL_Act.Value`
6	`AV_AL_Act.Value`
7	`AV_AH_En`
8	`AV_WH_En`
9	`AV_TH_En`
10	`AV_TL_En`
11	`AV_WL_En`
12	`AV_AL_En`
13	`AV_AH_MsgEn`
14	`AV_WH_MsgEn`
15	`AV_TH_MsgEn`
16	`AV_TL_MsgEn`
17	`AV_WL_MsgEn`
18	`AV_AL_MsgEn`
19	1 = No impact of input signals on local mode with `LocalSetting = 2` and `LocalSetting = 4`
20	1 = Valve closing
21	1 = Valve closed
22	1 = Valve stopped
23	1 = Valve opening
24	1 = Valve open
25	Feedback error for valve closed
26	Feedback error for valve open
27	Automatic preview for opening
28	Automatic preview for closing
29	Automatic preview for stopping
30	Display for interlocks in block icon
31	`MS_RelOp`

Status word allocation for `Status3` parameter

Status bit	Parameter
0	`M_MonStaErr.Value`
1	`M_MonDynErr.Value`
2 – 18	Not used
19	1 = Enabled for emergency stop push button (`Feature` bit Enabling rapid stop via faceplate (Page 192))
20 - 22	Not used
23	Command for rapid stop
24	Open command output
25	Close command output
26	Show automatic preview in the standard view
7 - 29	Not used
30	Auxiliary value 1 visible
31	Auxiliary value 2 visible

Status word allocation for `Status4` parameter

Status bit	Parameter
0	Effective `Signal 1` of the message block connected via `EventTsIn`
1	Effective `Signal 2` of the message block connected via `EventTsIn`
2	Effective `Signal 3` of the message block connected via `EventTsIn`
3	Effective `Signal 4` of the message block connected via `EventTsIn`
4	Effective `Signal 5` of the message block connected via `EventTsIn`
5	Effective `Signal 6` of the message block connected via `EventTsIn`
6	Effective `Signal 7` of the message block connected via `EventTsIn`
7	Effective `Signal 8` of the message block connected via `EventTsIn`
8	AV not interconnected
9 - 31	Not used

See also

VlvMotL functions (Page 784)
VlvMotL messaging (Page 792)
VlvMotL block diagram (Page 802)
VlvMotL error handling (Page 790)
VlvMotL modes (Page 782)

5.7.2 VlvMotL modes

VlvMotL operating modes

The block supports all standard modes:

- Local mode (Page 36)
- Automatic mode (Page 32)
- Manual mode (Page 32)
- Out of service (Page 27)

The next section provides additional block-specific information relating to the general descriptions.

"Local mode"

You can find general information on "Local mode", switching modes and bumpless switchover in the Local mode (Page 36) section.

Motor valve actions you can control in local mode

- Open (`OpenLocal = 1`)
- Close (`CloseLocal = 1`)
- Stop (`StopLocal = 1`).

A block operated in local mode is controlled either by local signals or by feedback signals (input parameters `FbkOpen` and `FbkClose`; if no position can be assigned, the last valid position is accepted). Configuration takes place via the input parameter `LocalSetting`.

Automatic mode

You can find general information on "Automatic mode", switching modes and bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 32) section.

Motor valve actions you can control in auto mode:

- Open (`OpenAut = 1`)
- Close (`CloseAut = 1`)
- Stop (`StopAut = 1`)

Manual mode

You will find general information on "Manual mode", switching modes and bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 32) section.

Motor valve actions you can control in manual mode:

- Open (`OpenMan = 1`)
- Close (`CloseMan = 1`)
- Stop (`StopMan = 1`)

Out of service

You can find general information about the "Out of service" mode in the Out of service (Page 27) section.

See also

VlvMotL block diagram (Page 802)

VlvMotL I/Os (Page 794)

VlvMotL error handling (Page 790)

VlvMotL functions (Page 784)

VlvMotL messaging (Page 792)

Description of VlvMotL (Page 778)

5.7.3 VlvMotL functions

Functions of VlvMotL

The functions for this block are listed below.

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

Operator control permissions via parameter OS_Perm

The block has the following Operator permissions (Page 45) for the OS_Perm parameter:

Bit	Function
0	1 = Operator can switch to automatic mode
1	1 = Operator can switch to manual mode
2	1 = Operator can switch to local mode
3	1 = Operator can switch to out of service mode
4	1 = Operator can stop the motor
5	1 = Operator can open the valve
6	1 = Operator can close the valve
7	1 = Operator can reset the valve
8	1 = Operator can define the monitoring time for the valve startup
9	1 = Operator can define the monitoring time for the valve runtime
10	1 = Operator can activate the function monitoring time for the valve (bits 8 & 9)
11 - 13	Not allocated
14	1 = Operator can enable function simulation
15	1 = Operator can enable the maintenance release function
16	1 = Operator can change the limit (AV) for the high alarm
17	1 = Operator can change the limit (AV) for the high warning
18	1 = Operator can change the limit (AV) for the high tolerance
19	1 = Operator can change the limit (AV) for hysteresis
20	1 = Operator can change the limit (AV) for the low alarm
21	1 = Operator can change the limit (AV) for the low warning
22	1 = Operator can change the limit (AV) for the low tolerance
23 - 31	Not allocated

Time delay for changes to control

If the state of the motor is changed, this is only undertaken after the time you define in the `IdleTime` input parameter. For example, if the motor is being moved into the stop state, the command for starting (open or close) can only be issued after the time saved in `IdleTime` has lapsed.

Please note that `IdleTime ≥ M_MonTiDynmic` has to be configured.

Limit monitoring of an additional analog value

This block provides the standard function Limit monitoring of an additional analog value (Page 73).

Limit monitoring with hysteresis

This block provides the standard function Limit monitoring with hysteresis (Page 78). It is performed via the input parameter `AV_Hyst`.

Interlocks

This block provides the following interlocks:

- Activation enable
- Interlock without reset (interlock)
- Interlock with reset (protection)

Refer to the section Interlocks (Page 100) as well as Influence of the signal status on the interlock (Page 103).

Trip function

This block provides the standard function Trip function (Page 102).

Rapid stop for motors

This block provides the standard function Rapid stop for motors (Page 102).

Torque monitoring

The block provides torque monitoring.

The signals of the torque monitoring switches are interconnected to input parameters `TorqOpen` and `TorqClose` for opening and closing the motor valve.

The Good state is indicated via this parameter by means of the value 1.

If the torque shutdown is active, the motor is stopped. You have the option of moving the valve in the opposite direction.

If, for example, the torque shutdown is active when the valve opens, you can still close the valve.

Disabling interlocks

This block provides the standard function Disabling interlocks (Page 107).

Resetting the block in case of interlocks

This block provides the standard function Resetting the block in case of interlocks or errors (Page 119).

Forming the group status for interlock information

This block provides the standard function Forming the group status for interlock information (Page 105).

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status for the block is formed by the following parameters:

- `FbkClsgOut.ST`
- `FbkOpngOut.ST`
- `FbkOpenOut.ST`
- `FbkCloseOut.ST`
- `LocalLi.ST`
- `OpenLocal.ST`
- `StopLocal.ST`
- `TorqClose.ST`
- `CloseLocal.ST`
- `Protect.ST`
- `Intlock.ST`
- `Permit.ST`
- `Trip.ST`
- `TorqOpen.ST`

Forcing operating states

This block provides the standard function Forcing operating states (Page 115). Inputs `OpenForce` and `CloseForce` and `StopForce`force the block to open, close or stop.

Feedback monitoring

This block provides the standard function Monitoring the feedbacks (Page 83).

The monitoring of the feedback for the valve will not be active if it was stopped during opening or closing.

Maintenance release

This block provides the standard function Maintenance release (Page 47).

Specify warning times for control functions

This block provides the standard function Specifying warning times for control functions at motors and valves (Page 122).

Simulating signals

This block provides the standard function Simulating signals (Page 93).

Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 53).

Safe position

This block provides the standard function Safe position for motors, valves and controllers (Page 120).

Output signal as a pulse signal or static signal

This block provides the standard function Output signal as a static signal or pulse signal (Page 86). In addition to the static control outputs `Open` and `Close`, the block also has pulse outputs `P_Open`, `P_Close`, and `P_Stop`, which are dependent on the static control output.

Monitoring valve feedback: The monitoring of valve feedback is set at the `V_Monitor` parameter.

Startup behavior is monitored by setting parameter`V_MonTiDynamic`. The parameter `V_MonTiStatic` monitors compliance with the position.

Feedback errors are displayed at the corresponding parameters `V_MonDynErr` and/or `V_MonStaErr`.

Monitoring the motor feedbacks: The monitoring of motor feedback is set at the `M_Monitor` parameter.

Startup behavior is monitored by setting parameter`M_MonTiDynamic`. The parameter `M_MonTiStatic` monitors compliance with the position.

Feedback errors are displayed at the corresponding parameters `M_MonDynErr` and/or `M_MonStaErr`.

Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 46).

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)
1	Reaction to the out of service mode (Page 199)
2	Resetting the commands for changing the mode (Page 193)
3	Enabling resetting of commands for the control settings (Page 193)
4	Setting switch or button mode (Page 195)
9	Resetting via input signals in the event of interlocks or errors (Page 194)
10	Exiting local mode (Page 199)
14	Enabling rapid stop via faceplate (Page 192)
17	Enabling bumpless changeover to automatic mode for valves, motors, and dosers (Page 197)
22	Update acknowledgment and error status of the message call (Page 193)

Displaying auxiliary values

This block provides the standard function Displaying auxiliary values (Page 52).

Time stamp

This block receives a time stamp value via the `EventTSIn` input parameter. Refer to EventTs functions (Page 997) for more information on this.

SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 122).

Disabling feedbacks

This block provides the standard function Disabling feedback for valves (Page 85). Feedback monitoring can be deactivated separately for each feedback with `NoFbkOpen` or `NoFbkClose` as required.

See also

Description of VlvMotL (Page 778)

VlvMotL messaging (Page 792)

VlvMotL I/Os (Page 794)

VlvMotL block diagram (Page 802)

VlvMotL error handling (Page 790)

VlvMotL modes (Page 782)

Time stamp (Page 51)

5.7.4 VlvMotL error handling

VlvMotL troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers
- Mode changeover error
- Invalid input signals

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Default value when implementing the block; block will not be processed.
0	There is no error.
41	The value for the connection `LocalSetting` is not within the approved limit from 0 to 4.
42	`LocalSetting = 0` or `LocalSetting = 3` or `LocalSetting = 4` and `LocalLi = 1`

51	`OpenLocal = 1` and `StopLocal = 1`
	`CloseLocal = 1` and `StopLocal = 1`
	`OpenLocal = 1` and `CloseLocal = 1`
	`OpenAut = 1` and `StopAut = 1`
	`CloseAut = 1` and `StopAut = 1`
	`OpenAut = 1` and `CloseAut = 1`
	`AutModLi = 1` and `ManModLi = 1`
	`OpenForce = 1` and `StopForce = 1`
	`CloseForce = 1` and `StopForce = 1`
	`OpenForce = 1` and `CloseForce = 1`

52	`LocalAct = 1` and `LocalSetting = 2` or 4 and `SimOn = 1`

Mode changeover error

This error can be output by the block, see the section Error handling (Page 117).

Invalid input signals

This error can be output by the block, see the section Error handling (Page 117).

See also

VlvMotL block diagram (Page 802)
VlvMotL I/Os (Page 794)
VlvMotL functions (Page 784)
VlvMotL modes (Page 782)
Description of VlvMotL (Page 778)

5.7.5 VlvMotL messaging

Messaging

The following messages can be generated for this block:

- Control system fault
- Instance-specific messages

Control system fault

The following control system error messages can be output:

Message instance	Message identifier	Message class	Event
`MsgEvId1`	SIG 1	AS process control message - fault	$$BlockComment$$ Motor feedback error
	SIG 2	AS process control message - fault	$$BlockComment$$ Tripping triggered
	SIG 3	AS process control message - fault	$$BlockComment$$ Valve feedback error
	SIG 4	AS process control message - fault	$$BlockComment$$ External error has occurred

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter `CSF`. If this signal changes to `CSF = 1`, a control system error is triggered (`MsgEvId1, SIG 4`).

Instance-specific messages

You can use up to three instance-specific messages with this block.

Message instance	Message identifier	Message class	Event
`MsgEvId1`	SIG 5	AS process control message - fault	$$BlockComment$$ External message 1
	SIG 6	AS process control message - fault	$$BlockComment$$ External message 2
	SIG 7	AS process control message - fault	$$BlockComment$$ External message 3

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Associated values for message instance `MsgEvId1`

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`
4	`ExtVa104`
5	`ExtVa105`
6	`ExtVa106`
7	`ExtVa107`
8	`ExtVa108`
9	Reserved
10	Reserved

The associated values 4 to 8 are allocated to the parameters `ExtVa104` ... `ExtVa108` and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

See also

VlvMotL block diagram (Page 802)

VlvMotL modes (Page 782)

Time stamp (Page 51)

5.7.6 VlvMotL I/Os

VlvMotL I/Os

Input parameters

Parameter	Description	Type	Default
AutModLi	1 = Automatic mode via interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AutModOp	1 = Automatic mode via operator (controlled by ModLiOp = 1)	BOOL	0
AV	Input additional analog value, to be connected to AV_Tech of the AV block	ANY	
AV_AH_Lim	Limit high alarm	REAL	95.0
AV_AL_Lim	Limit low alarm	REAL	5.0
AV_Hyst	Hysteresis for alarm, warning and tolerance limits	REAL	1.0
AV_TH_Lim	Limit high tolerance	REAL	85.0
AV_TL_Lim	Limit low tolerance	REAL	15.0
AV_WH_Lim	Limit high warning	REAL	90.0
AV_WL_Lim	Limit low warning	REAL	10.0
BatchEn	1 = Enable allocation	BOOL	0
BatchID	Batch ID	DWORD	16#00000000
BatchName	Batch name	STRING[32]	''
BypProt	1 = Bypassing interlock is active in local mode and simulation	BOOL	0
CloseAut	1 = Select "Close valve" in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
CloseForce	1 = Force valve closure	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
CloseLocal	1 = Select "Close valve" in local mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
CloseMan	1 = Select "Close valve" in manual mode	BOOL	0
CSF	1 = External error (control system error)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
EN	1 = Called block will be processed	BOOL	1

Parameter	Description	Type	Default
EventTSIn	To wire the signal status of the EventTs message block; The EventTsIn input parameter is used to interconnect to the EventTsOut output parameter of the EventTs block. When this interconnection is configured, the messages of the EventTs block are displayed on the OS in the message view of the technological block and can also be acknowledged there.	STRUCT • Value: BYTE • ST: BYTE	- • 16#00 • 16#80
ExtMsg2	Binary input for freely selectable message 2	STRUCT 1. Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg3	Binary input for freely selectable message 3	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtVa104	Associated value 4 for messages (MsgEvID1)	ANY	
ExtVa105	Associated value 5 for messages (MsgEvID1)	ANY	
ExtVa106	Associated value 6 for messages (MsgEvID1)	ANY	
ExtVa107	Associated value 7 for messages (MsgEvID1)	ANY	
ExtVa108	Associated value 8 for messages (MsgEvID1)	ANY	
FbkClose	1 = Valve closed feedback signal	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FbkClosing	1 = Valve closing feedback signal	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FbkOpen	1 = Valve open feedback signal	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FbkOpening	1 = Valve opening feedback signal	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Features	I/O for additional functions (Page 784)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
IdleTime	Wait time for change of direction or restart in [s]	REAL	5.0

Parameter	Description	Type	Default
Intlock	Interlocking without reset is effective; you can operate the block without reset once the interlocking condition has disappeared	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#FF
Intl_En	1 = Interlock without reset (interlock, Intlock parameter) is active	BOOL	1
LocalLi	1 = Activate local mode via plant signal	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
LocalOp	1 = Local mode by operator	BOOL	0
LocalSetting	Properties for the Local mode (Page 36)	INT	0
ManModLi	1 = Manual mode via interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ManModOp	1 = Manual mode via: OS operator (controlled by ModLiOp = 0)	BOOL	0
ModLiOp	Switchover of operating mode between: • 0 = Operator • 1 = Interconnection or SFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
M_Monitor	1 = Motor feedback monitoring	BOOL	1
M_MonTiDynamic	Motor monitoring time after operation in [s]	REAL	3.0
M_MonTiStatic	Monitoring time for feedback errors without operation in [s]	REAL	3.0
MS_RelOp	1= Maintenance release via OS operator	BOOL	0
MsgEvId1	Message number (assigned automatically)	DWORD	16#FFFFFF
MsgLock	1 = Suppress process messages	BOOL	0
MS_RelOp	1 = Maintenance release via OS operator	BOOL	0
NoFbkClose	1 = No feedback present for "valve closed"	BOOL	0
NoFbkOpen	1 = No feedback present for "valve open"	BOOL	0
Occupied	1 = In use by a batch	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OpenAut	1 = Select "Open valve" in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
OpenForce	1 = Force valve opening	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OpenLocal	1 = Select "Open valve" in local mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OpenMan	1 = Select "Open valve" in manual mode	BOOL	0
OS_Perm	I/O for operator control permissions (Page 784)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
Permit	1 = Enable for opening / closing from safe position 0 = Valve activation not enabled on OS	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#FF
Perm_En	1 = Activation enable (enable, Permit parameter) is active	BOOL	1
Protect	1 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#FF
Prot_En	1 = Protective interlock (protection, Protect parameter) is active	BOOL	1
PulseWidth	Pulse width of control signal [s]	REAL	3.0
RapidStp	Rapid stop for the motor: • 0 = Motor On • 1 = Motor Off	BOOL	0
RstLi	1 = Reset via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RstOp	1 = Reset via operator	BOOL	0
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
SafePos	Safe position for valve: 0 = Closed 1 = Open 2 = stop	INT	2
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SimAV	Additional value used for SimOn = 1	REAL	0.0
SimOn	1 = Simulation on	BOOL	0
SelFp1	Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-

Parameter	Description	Type	Default
SelFp2	Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
StepNo	Batch step number	DWORD	16#00000000
StopAut	1 = Stop the motor in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StopForce	1 = Force motor stop	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StopLocal	1 = Stop the motor in local mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StopMan	1 = Stop the motor in manual mode	BOOL	0
TorqOpen	0 = Torque shutdown active when opening 1 = Good state	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
TorqClose	0 = Torque shutdown active when closing 1 = Good state	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
Trip	1 = Motor is in "good" state	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
UserAna1	Analog auxiliary value 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
UA1unit	Unit of measure for analog auxiliary value 1	INT	0
UserAna2	Analog auxiliary value 2	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
UA2unit	Unit of measure for analog auxiliary value 2	INT	0
UserStatus	Freely assignable status word	BYTE	16#00
V_Monitor	1 = Valve feedback monitoring	BOOL	1
V_MonTiDynamic	Valve monitoring time after operation in [s]	REAL	5.0
V_MonTiStatic	Monitoring time for valve feedback errors without operation in [s]	REAL	5.0
WarnTiAut	Prewarning of valve movement from safe position in automatic mode in [s]	REAL	0.0
WarnTiMan	Prewarning of valve movement from safe position in manual mode in [s]	REAL	0.0

Output parameters

Parameter	Description	Type	Default
AutAct	1 = Automatic mode is active 0 = Manual mode is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AV_OpScale	Limit for scale in AV bar graph of faceplate	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
AV_Out	Output additional analog value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
AV_Unit	Unit of measure for additional analog value	INT	0
Close	Control output 1= Close valve	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see VlvMotL error handling (Page 790)	INT	-1
FbkCloseOut	Valve closed feedback	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FbkClsgOut	Valve closing feedback	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FbkOpenOut	Valve open feedback	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
FbkOpngOut	Valve opening feedback	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
LocalAct	1 = Local mode active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
LockAct	1 = Interlock (Inlock, Permit or Protect) is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
ManAct	1 = Manual mode active	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80

Parameter	Description	Type	Default
M_MonDynErr	1 = Motor feedback error due to control change	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
M_MonStaErr	1 = Motor feedback error due to unexpected feedback change	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_Release	Maintenance release: 1 = Release for OS operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MsgAckn1	Alarm acknowledgement status 1 (output STATUS of first ALARM_8P)	WORD	16#0000
MsgErr1	Alarm error 1 (output ERROR of first ALARM_8P)	BOOL	0
MsgStat1	Alarm status 1 (output ERROR of first ALARM_8P)	WORD	16#0000
OosAct	1 = Block is out of service	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Open	Control output 1 = Open the valve	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
P_Close	1 = Pulse signal to close valve	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
P_Open	1 = Pulse signal to open valve	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
P_Stop	1 = Pulse signal to stop valve	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
R_StpAct	1 = Rapid stop of the motor is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word 1 (Page 778)	DWORD	16#00000000
Status2	Status word 2 (Page 778)	DWORD	16#00000000
Status3	Status word 3	DWORD	16#00000000
Status4	Status word 4	DWORD	16#00000000

Parameter	Description	Type	Default
V_MonDynErr	1 = Valve feedback error due to control change	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
V_MonStaErr	1 = Valve feedback error due to unexpected feedback change	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
WarnAct	1 = Prewarning for valve movement away from safe position active (parameters WarnTiAut and WarnTiMan)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

VlvMotL messaging (Page 792)

VlvMotL block diagram (Page 802)

VlvMotL modes (Page 782)

5.7.7 VlvMotL block diagram

VlvMotL block diagram

A block diagram is not provided for this block.

See also

VlvMotL I/Os (Page 794)
VlvMotL messaging (Page 792)
VlvMotL error handling (Page 790)
VlvMotL functions (Page 784)
VlvMotL modes (Page 782)
Description of VlvMotL (Page 778)

5.7.8 Operator control and monitoring

5.7.8.1 VlvMotL views

Views of the VlvMotL block

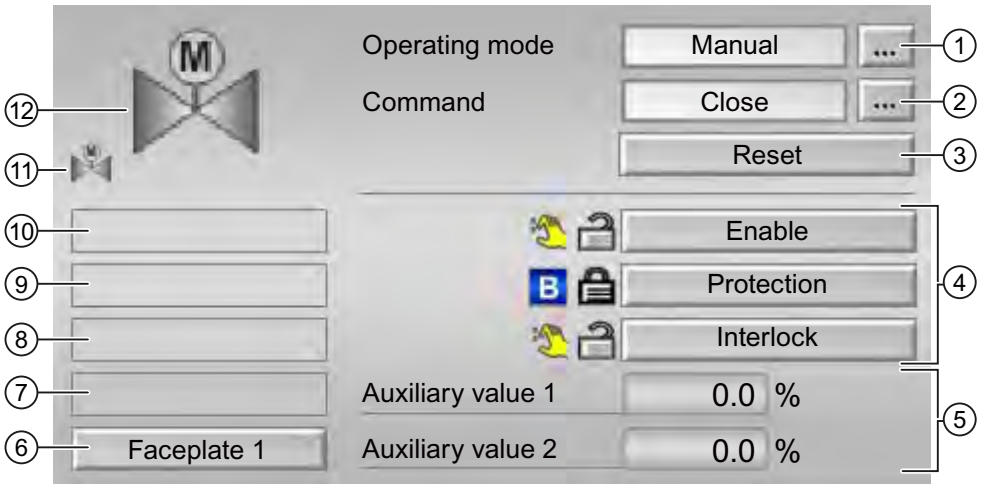
The block VlvMotL provides the following views:

- VlvMotL standard view (Page 803)
- Limit value view of motors (Page 163)
- Message view (Page 169)
- Trend view (Page 172)
- Parameter view for motors and valves (Page 155)
- VlvMotL preview (Page 806)
- Memo view (Page 171)
- Batch view (Page 170)
- Block symbol for VlvMotL (Page 809)

Refer to the sections Structure of the faceplate (Page 123) and Block icon structure (Page 174) for general information on the faceplate and block icon.

5.7.8.2 VlvMotL standard view

VlvMotL standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 32)
- Automatic mode (Page 32)
- Local mode (Page 36)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Open, close and stop the motor valve

This area shows you the default operating state for the motor valve. The following states can be shown and executed here:

- Open
- Close
- Stop
- Rapid stop

Refer to the Switching operating states and operating modes (Page 127) section for information on changing the state.

(3) Resetting the block

Click "Reset" for interlocks or errors. You can find additional information on this in the Resetting the block in case of interlocks or errors (Page 119) section.

(4) Display area for interlock functions of the block

You can use this button to control the interlock functions of the block. You can find additional information on this in the Interlocking functions (Page 100) section.

The following is displayed in addition to the buttons:

- Interlock status (see Forming the group status for interlock information (Page 105)), e.g.:

- Signal status (see Forming and outputting signal status for blocks (Page 88)), e.g.:

  If there is a bypass of one of the interlock signals, the symbol for the bypass is shown instead of the signal status.

- Bypass information:

  If there is a bypass, it is displayed instead of the signal status.

(5) Display of auxiliary values

You can use this area to display two auxiliary values that have been configured in the engineering system. You can find additional information on this in the Displaying auxiliary values (Page 52) section.

(6) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

(7) Display area for states of the block

This area provides additional information on the operating state of the block:

- Maintenance

Additional information on the display area for states of the block is available in section Maintenance release (Page 47).

(8) Display area for states of the block

This area provides additional information on the operating state of the block:

- Simulation

You can find additional information on this in the Simulating signals (Page 93) section.

(9) Display area for states of the block

This area provides additional information on the operating state of the block:

- Trip
- Runtime error
- Control error
- Invalid signal
- Mode switch fail

Additional information on these errors is available in the sections Monitoring the feedbacks (Page 83) , Error handling (Page 117) (section "Invalid input signals" and "Mode changeover error") and Trip function (Page 102).

(10) Display area for states of the block

This area provides additional information on the operating state of the block:

- Force open
- Force close
- Force stop

You can find additional information on this in the Forcing operating states (Page 115) section.

(11) Automatic preview

This display is only visible in manual mode or local mode when the current output signals are not identical to the control in automatic mode.

The display shows what state the valve would assume if you switched to automatic mode.

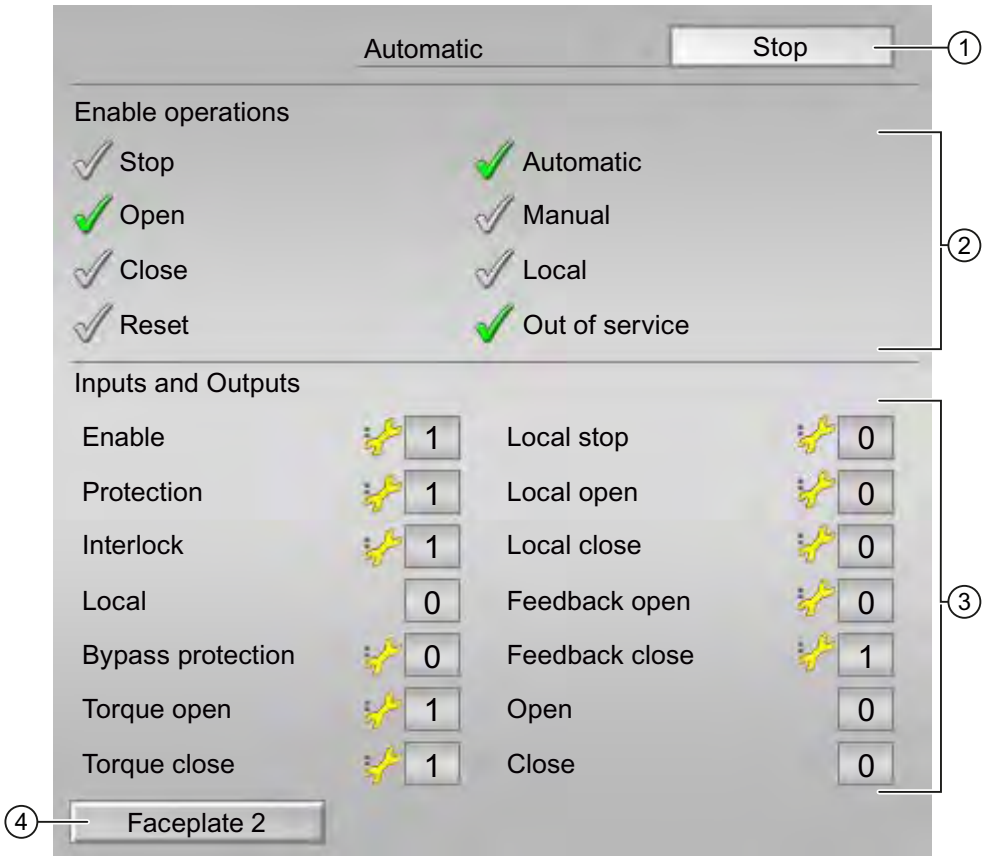
(12) Status display of motor valve

This area shows if the motor valve is open, closed or stopped:

- Green: Motor valve is open
- Gray: Motor valve is closed
- White motor, green-white valve: Motor is stopped
- Red: Fault at motor valve

5.7.8.3 VlvMotL preview

VlvMotL preview



(1) Automatic preview

This area shows you the block status after its has switched from manual to automatic mode.

If the block is in automatic mode, the current block state is displayed.

(2) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions

The following operator control enables are shown here:

- Stop: You can stop the motor of the valve.
- Open: You can open the motor valve.
- Close: You can close the motor valve.
- Reset: You can reset the motor valve if interlocks or errors occur.
- Automatic: You can switch to automatic mode.
- Manual: You can switch to manual mode
- Local: You can switch to local mode.
- Out of service: You can switch to out of service mode.

(3) Display current control signal

This area shows the most important parameters for this block with the current selection:

- Enable:
  - 0 = Motor valve activation not enabled on OS
  - 1 = Enable for opening / closing from safe position
- Protection:
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = Good state
- Interlock:
  - 0 = Interlocking without reset is effective; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = Good state
- Local: 1= Block is controlled in local mode
- Bypass protection:
  - 0 = Bridging deactivated
  - 1 = Bypassing interlock in local mode and simulation
- Torque open: 0 = Torque shutdown when opening
- Torque close: 0 = Torque shutdown when closing
- Local stop: 1 = Stop the motor valve in local mode
- Local open: 1 = Open motor valve in local mode
- Local close: 1 = Close motor valve in local mode
- Feedback for open: 1 = Motor valve is opened
- Feedback for closed: 1 = Motor valve is closed
- Open: 1 = Motor valve is opening
- Close: 1 = Motor valve is closing

(4) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

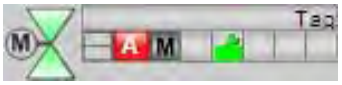
You can find additional information on this in the Calling further faceplates (Page 43) section.

5.7.8.4 Block symbol for VlvMotL

Block icons for VlvMotL

A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as the control system fault CSF
- Operating modes
- Signal status, maintenance release
- Displays for bridging interlocks
- Interlocks
- Memo display
- Valve status display

Icons	Selection of the block icon in CFC	Special features
	1	Block icon in the full display
	2	
	3	
	4	
	5	
	6	

Icons	Selection of the block icon in CFC	Special features
	7	
	8	
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181).

Valve status display

The following valve states are shown here:

Icon	Meaning
	Valve open
	Valve closed
	Error at valve
	Valve is opening
	Valve is closing

Channel blocks

6

6.1 Notes on using driver blocks

Notes on using driver blocks

- The descriptions of the driver blocks specify the OBs in which the blocks are installed. Please note that not all OBs listed will be generated for all CPUs. You can find additional information in the online help of the particular OB.
- If the driver generator uses the driver blocks of the PCS 7 libraries, you require a firmware version V3.1 or higher on the CPU.
- The CFC function "Generate module drivers" interconnects and configures the required I/Os automatically. The function is called and executed if hardware modifications are detected when compiling the program, for example.

Signal-processing blocks

The driver blocks of the available PCS 7 library offer a variety of channel blocks types for signal processing:

1. **Standard channel blocks**

   This applies to the following blocks:

   – Pcs7AnIn

   – Pcs7AnOu

   – Pcs7DiIn

   – Pcs7DiOu

   – Pcs7DiIT

   These blocks are used only for processing the signals of S7-300/400 SM modules. Use these standard blocks if you want to optimize memory and runtime utilization and do not need to process any PA devices.

2. **FF/PA channel blocks**

   This applies to the following blocks:

   – FbAnIn

   – FbAnOu

   – FbDiIn

   – FbDiOu

   These blocks are designed especially for use with PA field devices and the PROFIBUS 3.0 Class A and B or with FF field devices. In particular, you should use these blocks if you want to make use of the special features of these devices. In contrast to standard channel blocks, PA channel blocks not only process the signal itself but also all variables, according to the desired device configuration selected in the hardware configuration.

6.2 FbAnIn - Analog input driver for field devices

6.2.1 Description of FbAnIn

Object name (type + number) and family

Type + number: FB 1813

Family: Channel

Area of application for FbAnIn

The block is used for the following applications:

Signal processing (cyclic service) in accordance with "Transmitter" PROFIBUS PA profile of an analog input value:

- Of a PA field device in accordance with PROFIBUS 3.0 class A and B
- Of a main or auxiliary variable of a HART field device
- Of an FF field device

How it works

Block FbAnIn cyclically reads the process value and the signal status of the field device from the process image (partition). The process value is available as a physical variable. The signal status contains information about the status of the field device.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

Connect the symbol generated in HW Config (symbol table) for the input channel with the PV input parameter.

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The in/out parameter Mode is interconnected with the corresponding output parameter OMODE_xx of the MOD_PAL0 or MOD_PAX0 block.
- The in/out parameter DataXchg is interconnected with the corresponding output parameter DXCHG_xx of the MOD_PAL0 or MOD_PAX0 block.
- The icon for the signal status of the analog input channel is interconnected with input PV_ST.
- The MS parameter is interconnected with the O_MS output parameter of the diagnostics driver block.

You can use the FF_On = 1 input parameter to change the block to FF functionality. With FF_On = 0, the block works in PA mode.

If you are not using the CFC function "Generate module drivers" you must set the in/out parameter `Mode` manually. Refer to the section Mode settings for PA field devices (Page 912) or Mode settings for FF field devices (Page 913).

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for `Status` parameter

This block does not provide the `Status` parameter.

See also

FbAnIn block diagram (Page 823)
FbAnIn I/Os (Page 821)
FbAnIn messaging (Page 820)
FbAnIn error handling (Page 819)
FbAnIn functions (Page 816)
FbAnIn modes (Page 815)

6.2.2 FbAnIn modes

FbAnIn modes

This block does not provide any operating modes.

See also

FbAnIn block diagram (Page 823)
FbAnIn I/Os (Page 821)
FbAnIn messaging (Page 820)
FbAnIn error handling (Page 819)
FbAnIn functions (Page 816)
Description of FbAnIn (Page 814)
Out of service (Page 27)

6.2.3 FbAnIn functions

Functions of FbAnIn

The functions for this block are listed below.

Obtaining the standard value

The analog value of the process image (partition) is output as a standard value at the PV_Li output parameter.

Holding the last value if raw value is invalid

If the block is to hold the most recent valid value when the analog value is invalid, you must activate this function at the Feature bit Issuing last valid value if raw value is invalid (Page 191).

Note

FF field devices do not provide this function. If the raw value is invalid, only a substitute value or the invalid value can be output, as described below.

Output substitute value if raw value is invalid

If the block is to output a substitute value (SubsPV) when the analog value is invalid, you must activate this function at the Feature bit Issuing substitute value if raw value is invalid (Page 190).

Issuing an invalid value if analog value is invalid

If the module is to output an invalid value (PV_Li = PV), you must activate this function on Feature bit Output invalid raw value (Page 198).

This function is pre-selected.

Signal status for Fb channel blocks

The block provides the standard function Forming and outputting signal status for blocks (Page 88).

The signal status of process value PV_Li is generated from internal events such as channel errors, higher-level errors, or simulations, as well as from the signal status PV_ST, which comes directly from the device.

The signal status PV_ST can accept values of 16#00–16#FF.

The block recognizes a higher-level error, for example, failure of a DP/PA link, via the Mode input parameter.

- If the high byte is Mode = 16#80, then the values in the process image (partition) are valid.
- If the high byte is Mode = 16#40 (value status = higher-level error, ModErr = 1), then the analog value is treated as invalid.

The measuring type set in the low word of the Mode input parameter will be ignored.

The bit combinations of signal status PV_ST are output as output parameters (BOOL values). These conform to the bit combinations specified in PROFIBUS 3.0 "General Requirements".

For FF field devices only: The values of the signal status PV_ST_W = 16#84-16#87 and 16#90 – 16#93 are evaluated in the same way as signal status 16#80 – 16#83.

If the signal status PV_ST_W = 16#80 and a process value PV with the value 16#7FFFFFFF (invalid) are transferred from the FF field device, the block will deal with signal status 16#00 (invalid) from the FF field device.

Simulating signals

The block provides the standard function Simulating signals (Page 93).

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the section Functions that can be set via the Feature I/O (Page 186). The following functionality is available for this block at the relevant bits:

Bit	Function
28	Output invalid raw value (Page 198)
29	Issuing substitute value if raw value is invalid (Page 190)
30	Issuing last valid value if raw value is invalid (Page 191)

See also

FbAnIn block diagram (Page 823)

FbAnIn I/Os (Page 821)

FbAnIn messaging (Page 820)

FbAnIn error handling (Page 819)

FbAnIn modes (Page 815)

Description of FbAnIn (Page 814)

6.2.4 FbAnIn error handling

FbAnIn troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Channel error
- Higher-level error
- Invalid measuring range

Channel error

Channel errors are displayed as 1 at output parameter `Bad`. The channel error is generated from the signal status `PV_ST` or `PV_ST_W`.

For FF field devices only: If the signal status `PV_ST_W = 16#80` and a process value `PV` with the value `16#7FFFFFFF` (invalid) are transferred from the FF field device, the block will deal with signal status `16#00` (invalid) from the FF field device

Higher-level error

A higher-level error is displayed at the `ModErr` and `Bad` output parameters with a 1 if the signal status at High Word of input parameter `Mode` accepts the value `16#40`.

The signal status of the `PV_Li` output parameter is output depending on the settings at the `Feature` parameter.

Invalid measuring range

An invalid measuring range is indicated at the `ModErr` and `Bad` output parameters using a 1 if an invalid type of measurement is configured in the Low Word of the `Mode` input parameter.

The signal status of the `PV_Li` output parameter is output depending on the settings at the `Feature` parameter.

See also

FbAnIn block diagram (Page 823)

FbAnIn I/Os (Page 821)

FbAnIn messaging (Page 820)

FbAnIn functions (Page 816)

FbAnIn modes (Page 815)

Description of FbAnIn (Page 814)

6.2.5 FbAnIn messaging

Messaging

The block does not have any message functionality.

See also

FbAnIn block diagram (Page 823)
FbAnIn I/Os (Page 821)
FbAnIn error handling (Page 819)
FbAnIn functions (Page 816)
FbAnIn modes (Page 815)
Description of FbAnIn (Page 814)

6.2.6 FbAnIn I/Os

FbAnIn I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
Feature	I/O for additional functions (Page 816)	STRUCT • Bit 0: BOOL • ... • Bit 28: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 1 • 0 • 0
FF_On	0 = Block operating in PA mode 1 = Block operating in FF mode	BOOL	0
FlutEn	1 = Flutter suppression activated	BOOL	0
FlutTmIn	Flutter time: If a certain number of status changes take place within this time, fluttering is suppressed.	INT	0
ModeLw	Measuring type for FF devices	WORD	16#0001
MS	Maintenance status	DWORD	0
MS_Release	Maintenance release (interconnected with MS_Release of the technological block)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV	Process value (analog value)	REAL	0.0
PV_ST	Signal status for process value for PA devices	BYTE	16#80
PV_ST_W	Signal status for process value for FF devices	WORD	16#0080
PV_Unit	Unit of measure for process value	INT	1001
Scale	Scaling of the process value as a structure	STRUCT • High: REAL • Low:REAL	- • 100.0 • 0.0
SimOn	1 = Simulation on	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SimPV	Process value used for SimOn = 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SubsPV	Substitute value	REAL	0.0

In/out parameters

Parameter	Description	Type	Default
DataXchg	Data communication	DWORD	0
Mode	Value status and measuring type	DWORD	0

Output parameters

Parameter	Description	Type	Default
Bad	1 = Process value is not valid	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see FbAnIn error handling (Page 819)	INT	-1
ModErr	1 = Device/module is faulty	STRUCT • Value: BOOL • ST: BYTE	- • 0 16#80
OosAct	1 = Field device is undergoing maintenance	STRUCT • Value: BOOL • ST: BYTE	- • 0 16#80
PV_Li	Standard value (physical variable)	STRUCT • Value: BOOL • ST: BYTE	- • 0.000000e + 000 • 16#80
PV_LiUnit	Unit of the process value	INT	0
ScaleOut	Scaling of the process value as a structure	STRUCT • High: REAL • Low:REAL	- • 100.0 • 0.0
SimAct	1 = Simulation active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

FbAnIn block diagram (Page 823)
FbAnIn messaging (Page 820)
FbAnIn modes (Page 815)
Description of FbAnIn (Page 814)

6.2.7 FbAnIn block diagram

FbAnIn block diagram

This block does not come with a block diagram.

See also

FbAnIn I/Os (Page 821)
FbAnIn messaging (Page 820)
FbAnIn error handling (Page 819)
FbAnIn functions (Page 816)
FbAnIn modes (Page 815)
Description of FbAnIn (Page 814)

6.3 FbAnOu - Analog output driver for field devices

6.3.1 Description of FbAnOu

Object name (type + number) and family

Type + number: FB 1814

Family: Channel

Area of application for FbAnOu

The block is used for the following applications:

Signal processing (cyclic service) in accordance with "Actuator" PROFIBUS PA profile of an analog input value:

- Of a PA field device in accordance with PROFIBUS 3.0 class A and B
- Of a main or auxiliary variable of a HART field device
- Of an FF field device

How it works

Block FbAn0u cyclically reads the process value and the signal status of the field device from the process image (partition). The process values are available as physical variables. The signal status contains information about the status of the field device.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

You must interconnect each used signal of the block with the icons configured in HW Config or in the icon table according to your user data configuration:

PA device connection	Data type	I/O
Rbk	REAL	Input
RCasOut	REAL	Input
PosD	BYTE	Input
SP	REAL	Output
RCasIn	REAL	Output

FF device connection	Data type	I/O
Rbk	REAL	Input
RCasOutW	REAL	Input
PosD	WORD	Input
SP	REAL	Output
RCasIn	REAL	Output

You can use the FF_On = 1 input parameter to change the block to FF functionality. With FF_On = 0, the block works in PA mode.

When using FF devices, the ModeLw input also has to be configured in accordance with the reference data used. Refer to the section Mode settings for FF field devices (Page 913) for more information.

For FF devices, you need to interconnect each signal status manually:

FF device connection	Data type	I/O
RbkST_W	WORD	Input
RCasOutST_W	WORD	Input
PosD_ST_W	WORD	Input
SP_ST_W	WORD	Output
RCasIn_ST_W	WORD	Output
CbkBy0_W	WORD	Input
CbkBy1_W	WORD	Input
CbkBy2_W	WORD	Input

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The Mode in/out parameter is interconnected with the corresponding output parameter OMode_xx of the MOD_PAL0- or MOD_PAX0 block.
- The in/out parameter DataXchg is interconnected with the corresponding output parameter DataXchg_xx of the MOD_PAL0- or MOD_PAX0 block.

- The MS parameter is interconnected with the O_MS output parameter of the diagnostics driver block.
- The corresponding signal status is symbolically interconnected depending on the reference data configuration:

PA device connection	Data type	I/O
RbkST	BYTE	Input
RCasOutST	BYTE	Input
PosD_ST	BYTE	Input
SP_ST	BYTE	Output
RCasIn_ST	BYTE	Output
CbkBy0	BYTE	Input
CbkBy1	BYTE	Input
CbkBy2	BYTE	Input

Startup characteristics

The block will run through one time in OB100 at system start. The output and in/out parameters are calculated.

Status word allocation for Status parameter

This block does not provide the Status parameter.

See also

FbAnOu block diagram (Page 836)
FbAnOu I/Os (Page 831)
FbAnOu error handling (Page 830)
FbAnOu functions (Page 828)
FbAnOu modes (Page 827)
FbAnOu messaging (Page 831)

6.3.2 FbAnOu modes

FbAnOu modes

This block does not provide any modes.

See also

FbAnOu block diagram (Page 836)
FbAnOu I/Os (Page 831)
FbAnOu error handling (Page 830)
FbAnOu functions (Page 828)
Description of FbAnOu (Page 824)
Out of service (Page 27)
FbAnOu messaging (Page 831)

6.3.3 FbAnOu functions

Functions of FbAnOu

The functions for this block are listed below.

Obtaining the standard value

The signals of the FF field device are read from the process image (partition) of the inputs and written to the process image (partition) of the outputs. The controlled variable Rbk and discrete position feedbackPosD, are read, as well as the active reference variable RCasOut, together with its associated signal status RbkST, PosD_ST and RCasOutST are read and written to the output parameters SP and RCasIn, with the associated signal status RbkST and RCasInST.

Additional detailed device information (Cbk0-Cbk2) can be read as an option. The device information is available per bit at the block output.

The signal status RbkST or RCasOutST that comes directly from the device can have values from 16#00 – 16#FF.

The values of the signal status PosD_ST and RbkST of 16#84-16#87 and 16#90 – 16#93 of the FF field device are evaluated like the signal status 16#80 – 16#83 .

Signal status for Fb channel blocks

The block provides the standard function Forming and outputting signal status for blocks (Page 88).

The signal status of the process values (RCasInLi or RbkLi) is generated from internal events such as channel errors, higher-level errors, or simulations, as well as from the signal status RbkST or RCasInST, which comes directly from the device.

Value	Meaning
16#80	Valid value
16#60	Simulation
16#28	Bad, process related
16#68	Unknown, device related
16#78	Unknown, process related
16#A4	Maintenance request present
16#00	Invalid value

Simulating signals

The block provides the standard function Simulating signals (Page 93).

See also

FbAnOu block diagram (Page 836)
FbAnOu I/Os (Page 831)
FbAnOu error handling (Page 830)
FbAnOu modes (Page 827)
Description of FbAnOu (Page 824)
FbAnOu messaging (Page 831)

6.3.4 FbAnOu error handling

FbAnOu troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Channel error
- Higher-level error

Channel error

Channel errors are displayed as 1 at output parameter `Bad`. The channel error is generated from the signal states `RbkST`, `RCasOutST`, and `PosD_ST`. In addition to this, a channel error (`Bad = 1`) also occurs if the signal status has a valid value (`RbkST` or `RCasOutST` is `16#80`) and the present position of actuator `Rbk` or `RCasOut` has the value `16#7FFFFFFF` (invalid).

Higher-level error / invalid measuring range

A higher-level error or an invalid measuring range is output (output parameter `ModErr = 1, Bad = 1`) if the signal status at High Word of input parameter `Mode` accepts the value `16#40`.

Either `16#00` (in the event of an error) or `16#60` (in the event of simulation) is output at the `PV_Li` output parameter of the signal status.

See also

FbAnOu block diagram (Page 836)

FbAnOu I/Os (Page 831)

FbAnOu functions (Page 828)

FbAnOu modes (Page 827)

Description of FbAnOu (Page 824)

FbAnOu messaging (Page 831)

6.3.5 FbAnOu messaging

Messaging

The block does not have any message functionality.

See also

Description of FbAnOu (Page 824)
FbAnOu modes (Page 827)
FbAnOu functions (Page 828)
FbAnOu error handling (Page 830)
FbAnOu I/Os (Page 831)
FbAnOu block diagram (Page 836)

6.3.6 FbAnOu I/Os

FbAnOu I/Os

Input parameters

Parameter	Description	Type	Default
CbkBy0	Additional information about checkback for PA field devices (bit-coded)	BYTE	16#00
CbkBy0_W	Additional information about checkback for FF field devices (bit-coded)	WORD	16#0000
CbkBy1	Additional information about checkback for PA field devices (bit-coded)	BYTE	16#00
CbkBy1_W	Additional information about checkback for FF field devices (bit-coded)	WORD	16#0000
CbkBy2	Additional information about checkback for PA field devices (bit-coded)	BYTE	16#00
CbkBy2_W	Additional information about checkback for FF field devices (bit-coded)	WORD	16#0000
EN	1 = Called block will be processed	BOOL	1
FF_On	0 = Block operating in PA mode 1 = Block operating in FF mode	BOOL	0
FlutEn	1=Flutter suppression activated	BOOL	0
FlutTmIn	Flutter time: If a certain number of status changes take place within this time, fluttering is suppressed.	INT	0
ModeLw	Measuring type for FF devices	WORD	16#0000

Parameter	Description	Type	Default
MS	Maintenance status	DWORD	0
MS_Release	Maintenance release (interconnected with MS_Release of the technological block)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PosD	Discrete position feedback (current position) of valve for PA devices: • 0 = Not initialized • 1 = Closed • 2 = Opened • 3 = Intermediate	BYTE	16#00
PosD_ST	Signal status of PosD for PA devices	BYTE	16#80
PosD_ST_W	Signal status of PosD_W for FF devices	WORD	16#0080
PosD_W	Discrete position feedback (current position) of valve for FF devices: • 0 = Not initialized • 1 = Closed • 2 = Opened • 3 = Intermediate	WORD	16#0000
RCasInLi	Setpoint for remote cascade operating mode of PA or FF device	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
RCasOut	Setpoint from function block in device	REAL	0.0
RCasOutST	Signal status of RCasOut for PA devices	BYTE	16#80
RCasOutST_W	Signal status of RCasOut for FF devices	WORD	16#80
Rbk	Current position of actuator (actual value)	REAL	0.0
RbkST	Signal status of Rbk for PA devices	BYTE	16#80
RbkST_W	Signal status of Rbk for FF devices	WORD	16#0080
Scale	Scaling of the process value as structure for display	STRUCT • High: REAL • Low:REAL	- • 100.0 • 0.0
SP_Li	Setpoint	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_LiUnit	Unit of measure for setpoint	INT	1342
SimOn	1 = Simulation on	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SimPosD	Discrete position feedback for (output parameter PosD_Li) that is used for SimOn = 1.	BYTE	16#00
SimRCasInLi	Setpoint for the remote cascade operating mode RCasInLi used for SimOn = 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

Parameter	Description	Type	Default
SimRbk	Current position of valve (actual value) Rbk used for SimOn = 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SimSP_Li	Setpoint SP_Li used for SimOn = 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

In/out parameters

Parameter	Description	Type	Default
DataXchg	Data communication	DWORD	0
Mode	Value status and measuring type	DWORD	0

Output parameters

Parameter	Description	Type	Default
Bad	1 = Process value is not valid	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk0	1 = Field device in safe position	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk1	1 = Request for local operation	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk2	1 = Device is operated locally	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk3	1 = Emergency operation is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk4	1 = Deviation in direction of movement	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk5	1 = Stop reached (actuator fully open)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk6	1 = Stop reached (actuator fully closed)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
Cbk7	1 = Runtime overshoot	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk8	1 = Actuator is opening	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk9	1 = Actuator is closing	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk10	1 = The alarm generated by any change to the static data (function and transducer block)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk11	1 = Simulation mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk12	Not used	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk13	1 = Internal control loop interrupted	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk14	1 = Closed-loop control inactive	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk15	1 = Self-test is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk16	1 = Stroke integral exceeded	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk17	1 = Additional input is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see FbAnOu error handling (Page 830)	INT	-1
ModErr	1 = Device/module is faulty	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
OosAct	1 = Field device is undergoing maintenance	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PosD_Li	Discrete position feedback (current position) of valve for PA and FF devices: • 0 = Not initialized • 1 = Closed • 2 = Opened • 3 = Intermediate	BYTE	16#00
PosD_LiST	Signal status PosD_Li	BYTE	16#00
RCasIn	Setpoint for remote cascade operating mode of PA or FF device	REAL	0.0
RCasInST	Signal status of RCasIn for PA devices	BYTE	16#00
RCasInST_W	Signal status of RCasIn for FF devices	WORD	16#0000
RCasOutLi	Setpoint from function block in device	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
RbkLi	Current position of actuator (actual value)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
ScaleOut	Scaling of the process value as structure for display	STRUCT • High: REAL • Low:REAL	- • 100.0 • 0.0
SimAct	1 = Simulation active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP	Setpoint	REAL	0.0
SP_ST	Signal status of setpoint for PA devices	BYTE	16#00
SP_ST_W	Signal status of setpoint for FF devices	WORD	16#0000
SP_Unit	Unit of the process value (SP_LiUnit)	INT	0

See also

FbAnOu block diagram (Page 836)
FbAnOu functions (Page 828)
FbAnOu modes (Page 827)
Description of FbAnOu (Page 824)
FbAnOu messaging (Page 831)

6.3.7 FbAnOu block diagram

FbAnOu block diagram

A block diagram is not provided for this block.

See also

FbAnOu I/Os (Page 831)
FbAnOu error handling (Page 830)
FbAnOu functions (Page 828)
FbAnOu modes (Page 827)
Description of FbAnOu (Page 824)
FbAnOu messaging (Page 831)

6.4 FbDiIn - Digital input driver for field devices

6.4.1 Description of FbDiIn

Object name (type + number) and family

Type + number: FB 1815

Family: Channel

Area of application for FbDiIn

The block is used for the following applications:

Signal processing of digital input values (discrete input) of a field device (cyclic service in accordance with PROFIBUS PA):

- Of a PA field device in accordance with PROFIBUS 3.0 class A and B
- Of an FF field device

How it works

Block FbDiIn cyclically reads the process values and the signal status of the field device from the process image (partition). The process values are grouped in one byte. The signal status contains information about the status of the field device.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The Mode in/out parameter is interconnected with the corresponding output parameter OMode_xxof the MOD_PAL0- or MOD_PAX0 block.
- The in/out parameter DataXchg is interconnected with the corresponding output parameter DataXchg_xx of the MOD_PAL0- or MOD_PAX0 block.
- The icon for the signal status of the analog input channel is interconnected with input PV_ST.
- The MS parameter is interconnected with the O_MS output parameter of the diagnostics driver block.

You can use the FF_On = 1 input parameter to change the block to FF functionality. With FF_On = 0, the block works in PA mode.

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for `Status` parameter

This block does not provide the `Status` parameter.

See also

FbDiIn block diagram (Page 846)
FbDiIn I/Os (Page 843)
FbDiIn messaging (Page 843)
FbDiIn error handling (Page 842)
FbDiIn functions (Page 839)
FbDiIn modes (Page 839)

6.4.2 FbDiIn modes

FbDiIn modes

This block does not provide any operating modes.

See also

FbDiIn block diagram (Page 846)

FbDiIn I/Os (Page 843)

FbDiIn messaging (Page 843)

FbDiIn error handling (Page 842)

FbDiIn functions (Page 839)

Description of FbDiIn (Page 837)

Out of service (Page 27)

6.4.3 FbDiIn functions

Functions of FbDiIn

The functions for this block are listed below.

Obtaining the standard value

The digital values (WORD format) of the process image (partition) are output at the PV_Li0 - PV_Li7 output parameters.

The signal status PV_ST that comes directly from the device can have values from 16#00 – 16#FF.

The values of the signal status PV_ST of 16#84-16#87 and 16#90 – 16#93 of the FF field device are evaluated in the same way as 16#80 – 16#83.

Holding the last value if raw value is invalid

If the block is to hold the most recent valid value when the digital value is invalid, you must activate this function at the Feature bit Issuing last valid value if raw value is invalid (Page 191).

Output substitute value if raw value is invalid

If the module is to output a substitute value (SubsPV) when the digital value is invalid, you must activate this function on Feature bit Issuing substitute value if raw value is invalid (Page 190).

Output of invalid value if raw value is invalid

If the block is to output an invalid value (PV_Li = PV), you must activate this function at the Feature bit Output invalid raw value (Page 198).

This function is pre-selected.

Signal status for Fb channel blocks

The block provides the standard function Forming and outputting signal status for blocks (Page 88).

The signal status of process value PV_Li0 - PV_Li7 is generated from internal events such as channel errors, higher-level errors, or simulations, as well as from the signal status PV_ST, which comes directly from the device.

Value	Meaning
16#80	Valid value
16#60	Simulation
16#28	Bad, process related
16#68	Unknown, device related
16#78	Unknown, process related
16#A4	Maintenance request present
16#00	Invalid value

Simulating signals

The block provides the standard function Simulating signals (Page 93).

Configurable reactions using the Feature parameter

You can find an overview of all reactions provided by the Feature parameter in the section Functions that can be set via the Feature I/O (Page 186). The following functionality is available for this block at the relevant bits:

Bit	Function
28	Output invalid raw value (Page 198)
29	Issuing substitute value if raw value is invalid (Page 190)
30	Issuing last valid value if raw value is invalid (Page 191)

See also

FbDiIn block diagram (Page 846)
FbDiIn I/Os (Page 843)
FbDiIn messaging (Page 843)
FbDiIn error handling (Page 842)
FbDiIn modes (Page 839)
Description of FbDiIn (Page 837)

6.4.4 FbDiIn error handling

FbDiIn troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Channel error
- Higher-level error

Channel error

Channel errors are displayed as 1 at output parameter `Bad`. The channel error is generated from the signal status `PV_ST`.

Higher-level error / invalid measuring range

A higher-level error or an invalid measuring range is output (output parameter `ModErr = 1, Bad = 1`) if the signal status at High Word of input parameter `Mode` accepts the value `16#40`.

Either `16#00` (in the event of an error) or `16#60` (in the event of simulation) is output at the `PV_Li` output parameter of the signal status.

See also

FbDiIn block diagram (Page 846)

FbDiIn I/Os (Page 843)

FbDiIn messaging (Page 843)

FbDiIn functions (Page 839)

FbDiIn modes (Page 839)

Description of FbDiIn (Page 837)

6.4.5 FbDiIn messaging

Messaging

The block does not have any message functionality.

See also

FbDiIn block diagram (Page 846)
FbDiIn I/Os (Page 843)
FbDiIn error handling (Page 842)
FbDiIn functions (Page 839)
FbDiIn modes (Page 839)
Description of FbDiIn (Page 837)

6.4.6 FbDiIn I/Os

FbDiIn I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
Feature	I/O for additional FbDiIn functions (Page 839)	STRUCT • Bit 0: BOOL • ... • Bit 28: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 1 • 0 • 0
FF_On	0 = Block operating in PA mode 1 = Block operating in FF mode	BOOL	0
FlutEn	1=Flutter suppression activated	BOOL	0
FlutTmIn	Flutter time: If a certain number of status changes take place within this time, fluttering is suppressed.	INT	0
ModeLw	Measuring type for FF devices	WORD	16#0002
MS	Maintenance status	DWORD	0
MS_Release	Maintenance release (interconnected with MS_Release of the technological block)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV	Process value (digital value) for PA devices	BYTE	16#00

Parameter	Description	Type	Default
PV_ST	Signal status for process value for PA devices	BYTE	16#80
PV_ST_W	Signal status for process value for FF devices	WORD	16#0080
PV_W	Process value (digital value) for FF devices	WORD	16#0000
SimOn	1 = Simulation on	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SimPV0	Process value used for SimOn = 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SimPV1	Process value used for SimOn = 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SimPV2	Process value used for SimOn = 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SimPV3	Process value used for SimOn = 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SimPV4	Process value used for SimOn = 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SimPV5	Process value used for SimOn = 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SimPV6	Process value used for SimOn = 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SimPV7	Process value used for SimOn = 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SubsPV	Substitute value	BYTE	0

In/out parameters

Parameter	Description	Type	Default
DataXchg	Data communication	DWORD	0
Mode	Value status and measuring type	DWORD	0

Output parameters

Parameter	Description	Type	Default
Bad	1 = Process value is not valid	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see FbDiIn error handling (Page 842)	INT	-1
ModErr	1 = Device/module is faulty	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosAct	1 = Field device is undergoing maintenance	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_Li0	Process value 0	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_Li1	Process value 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_Li2	Process value 2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_Li3	Process value 3	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_Li4	Process value 4	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_Li5	Process value 5	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_Li6	Process value 6	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_Li7	Process value 7	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SimAct	1 = Simulation active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

FbDiIn block diagram (Page 846)
FbDiIn messaging (Page 843)
FbDiIn modes (Page 839)
Description of FbDiIn (Page 837)

6.4.7 FbDiIn block diagram

FbDiIn block diagram

A block diagram is not provided for this block.

See also

FbDiIn I/Os (Page 843)
FbDiIn messaging (Page 843)
FbDiIn error handling (Page 842)
FbDiIn modes (Page 839)
FbDiIn functions (Page 839)
Description of FbDiIn (Page 837)

6.5 FbDiOu - Digital output driver for field devices

6.5.1 Description of FbDiOu

Object name (type + number) and family

Type + number: FB 1816

Family: Channel

Area of application for FbDiOu

The block is used for the following applications:

Signal processing of max. 8 digital input/output values of a field device (cyclic service in accordance with PROFIBUS PA):

- Of a PA field device in accordance with PROFIBUS 3.0 class A and B
- Of an FF field device

How it works

Block FbDiOu cyclically reads the process value and the signal states of the field device from the process image (partition). The eight process values are grouped in one byte for each of the input parameters SP_Li and RCasInLi. The signal status contains information about the status of the field device.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

You must interconnect each used signal of the block with the icons configured in HW Config or in the icon table according to your user data configuration:

PA device connection	Data type	I/O
Rbk	BYTE	Input
RCasOut	BYTE	Input
SP	BYTE	Output
RCasIn	BYTE	Output

FF device connection	Data type	I/O
Rbk_W	WORD	Input
RCasOutW	WORD	Input
SP_W	WORD	Output
RCasIn_W	WORD	Output

You can use the FF_On = 1input parameter to change the block to FF functionality. With FF_On = 0, the block works in PA mode.

When using FF devices, the ModeLw input also has to be configured in accordance with the reference data used. Refer to the section Mode settings for FF field devices (Page 913) for more information.

For FF devices, you need to interconnect each signal status manually:

FF device connection	Data type	I/O
RbkST_W	WORD	Input
RCasOutW	WORD	Input
SP_ST_W	WORD	Output
RCasInST_W	WORD	Output
CbkBy0_W	WORD	Input
CbkBy1_W	WORD	Input
CbkBy2_W	WORD	Input

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The Mode in/out parameter is interconnected with the corresponding output parameter OMode_xxof the MOD_PAL0- or MOD_PAX0 block.
- The in/out parameter DataXchg is interconnected with the corresponding output parameter DataXchg_xx of the MOD_PAL0- or MOD_PAX0 block.
- The MS parameter is interconnected with the O_MS output parameter of the diagnostics driver block.
- The corresponding signal status is symbolically interconnected depending on the reference data configuration:

PA device connection	Data type	I/O
RbkST	BYTE	Input
RCasOutST	BYTE	Input
SP_ST	BYTE	Output
RCasInST	BYTE	Output
CbkBy0	BYTE	Input
CbkBy1	BYTE	Input
CbkBy2	BYTE	Input

Startup characteristics

The block will run through one time in OB100 at system start. The output and in/out parameters are calculated.

Status word allocation for `Status` parameter

This block does not provide the `Status` parameter.

See also

FbDiOu block diagram (Page 858)

FbDiOu I/Os (Page 854)

FbDiOu messaging (Page 853)

FbDiOu error handling (Page 852)

FbDiOu functions (Page 850)

FbDiOu modes (Page 850)

6.5.2 FbDiOu modes

FbDiOu modes

This block does not provide any modes.

See also

FbDiOu block diagram (Page 858)
FbDiOu I/Os (Page 854)
FbDiOu messaging (Page 853)
FbDiOu error handling (Page 852)
FbDiOu functions (Page 850)
Description of FbDiOu (Page 847)
Out of service (Page 27)

6.5.3 FbDiOu functions

Functions of FbDiOu

The functions for this block are listed below.

Obtaining the standard value

The signals of the FF field device are read from the process image (partition) of the inputs and written to the process image (partition) of the outputs. The input parameter `Rbk` and the active reference variable `RCasOut`, together with its associated signal status `RbkST` and `RCasOutST` are read and written to the output parameters `SP` and `RCasIn` with the associated signal status `RbkST` and `RCasOutSt`. Additional detailed device information (`CbkBy0-CbkBy2`) can be read as an option. The device information is available per bit at the block output.

The signal status `RbkST` or `RCasOutST` that comes directly from the device can have values from `16#00 – 16#FF`.

The values of the signal status `RbkST` and `RCasOutST` of `16#84-16#87`and `16#90 – 16#93`of the FF field device are evaluated as `16#80 – 16#83`.

Signal status for Fb channel blocks

The block provides the standard function Forming and outputting signal status for blocks (Page 88).

The signal status of the process values (`RCasIn` or `RbkLi`) is generated from internal events such as channel errors, higher-level errors, or simulations, as well as from the signal status `RbkLiST` or `RCasInST`, which comes directly from the device.

Value	Meaning
16#80	Valid value
16#60	Simulation
16#28	Bad, process related
16#68	Unknown, device related
16#78	Unknown, process related
16#A4	Maintenance request present
16#00	Invalid value

Simulating signals

The block provides the standard function Simulating signals (Page 93).

See also

FbDiOu block diagram (Page 858)

FbDiOu I/Os (Page 854)

FbDiOu messaging (Page 853)

FbDiOu error handling (Page 852)

FbDiOu modes (Page 850)

Description of FbDiOu (Page 847)

6.5.4 FbDiOu error handling

FbDiOu troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Channel error
- Higher-level error

Channel error

Channel errors are displayed as 1 at output parameter `Bad`. The channel error is generated from the signal states `ReadBackSt` and `InCasOutSt`.

Higher-level error / invalid measuring range

A higher-level error or an invalid measuring range is output (output parameter `ModErr = 1, Bad = 1`) if the signal status at High Word of input parameter `Mode` accepts the value `16#40`.

Either `16#00` (in the event of an error) or `16#60` (in the event of simulation) is output at the `PV_Li` output parameter of the signal status.

See also

FbDiOu block diagram (Page 858)

FbDiOu I/Os (Page 854)

FbDiOu messaging (Page 853)

FbDiOu functions (Page 850)

FbDiOu modes (Page 850)

Description of FbDiOu (Page 847)

6.5.5 FbDiOu messaging

Messaging

The block does not have any message functionality.

See also

FbDiOu block diagram (Page 858)
FbDiOu I/Os (Page 854)
FbDiOu error handling (Page 852)
FbDiOu functions (Page 850)
FbDiOu modes (Page 850)
Description of FbDiOu (Page 847)

6.5.6 FbDiOu I/Os

FbDiOu I/Os

Input parameters

Parameter	Description	Type	Default
CbkBy0	Additional information about checkback for PA field devices (bit-coded)	BYTE	16#00
CbkBy0_W	Additional information about checkback for FF field devices (bit-coded)	WORD	16#0000
CbkBy1	Additional information about checkback for PA field devices (bit-coded)	BYTE	16#00
CbkBy1_W	Additional information about checkback for FF field devices (bit-coded)	WORD	16#0000
CbkBy2	Additional information about checkback for PA field devices (bit-coded)	BYTE	16#00
CbkBy2_W	Additional information about checkback for FF field devices (bit-coded)	WORD	16#0000
EN	1 = Called block will be processed	BOOL	1
FF_On	0 = Block operating in PA mode 1 = Block operating in FF mode	BOOL	0
FlutEn	1=Flutter suppression activated	BOOL	0
FlutTmIn	Flutter time: If a certain number of status changes take place within this time, fluttering is suppressed.	INT	0
ModeLw	Measuring type for FF devices	WORD	16#0000
MS_Release	Maintenance release (interconnected with MS_Release of the technological block)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RCasInLi	Setpoint for remote cascade operating mode of PA or FF device	BYTE	16#00
RCasInLiST	Signal status for the setpoint value (RCasInLi)	BYTE	16#80
Rbk	Current position of actuator (actual value) for PA devices	BYTE	16#00
RbkST	Signal status of Rbk for PA devices	BYTE	16#80
RbkW	Current position of actuator (actual value) for FF devices	WORD	16#0000
RbkST_W	Signal status of RbkW for FF devices	WORD	16#0080
RCasOut	Setpoint from function block in device	BYTE	16#00
RCasOutST	Signal status of RCasOut	BYTE	16#80
RCasOutW	Setpoint from function block in device	WORD	16#0000
RCasOutST_W	Signal status of RCasOut	WORD	16#0080
SP_Li	Setpoint	BYTE	16#00
SP_LiST	Signal status of setpoint (SP_Li)	BYTE	16#80

Parameter	Description	Type	Default
SimOn	1 = Simulation on	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SimRCasInLi	Setpoint for the remote cascade operating mode RCasInLi used for SimOn = 1	BYTE	16#00
SimRbk	Current position of valve (actual value) Rbk used for SimOn = 1	BYTE	16#00
SimSP_Li	Setpoint SP_Li used for SimOn = 1	BYTE	16#00

In/out parameters

Parameter	Description	Type	Default
DataXchg	Data communication	DWORD	16#0000
Mode	Value status and measuring type	DWORD	16#0000

Output parameters

Parameter	Description	Type	Default
Bad	1 = Process value is not valid	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk0	1 = Field device in safe position active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk1	1 = Request for manual mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk2	1 = Field device in manual mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk3	1 = Emergency override active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk4	1 = Current position differs from expected position	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk5	1 = Valve connection break	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
Cbk6	1 = Indicates a short-circuit at the valve connection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk7	1 = Not used	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk8	1 = Actuator is opening	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk9	1 = Actuator is closing	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk10	1 = The alarm generated by any change to the static data (function and transducer block)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk11	1 = Simulation of process values is enabled	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk12	1 = Not used	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk13	1 = Internal control loop interrupted	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk14	1 = Valve not active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk15	1 = Device under self test	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk16	1 = Valve travel limit has been exceeded	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk17	1 = Limit of break time exceeded when changing from OPEN to CLOSE	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk18	1 = Limit of break time exceeded when changing from CLOSE to OPEN	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
Cbk19	1 = Error occurred in the internal cycle test	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk20	1 = Timeout during the transition from OPEN to CLOSE	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk21	1 = Timeout during the transition from CLOSE to OPEN	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk22	1 = Valve blocked mechanically	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Cbk23	1 = Zero point not reached	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see FbDiOu error handling (Page 852)	INT	-1
ModErr	1 = Device/module is faulty	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosAct	1 = Field device is undergoing maintenance	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RCasOutLi	Setpoint from function block in device	BYTE	16#00
RCasOutLiST	Signal status for setpoint from function block in device (RCasOutLi)	BYTE	16#00
RCasIn	Setpoint for remote cascade operating mode of PA device	BYTE	16#00
RCasInST	Signal status of RCasIn for PA devices	BYTE	16#00
RCasInST_W	Signal status of RCasInW for FF devices	WORD	16#0000
RCasInW	Setpoint for remote cascade operating mode of FF device	WORD	16#0000
RbkLi	Current position of actuator (actual value)	BYTE	16#00
RbkLiST	Signal status of current position of actuator (RbkLi)	BYTE	16#00
SimAct	1 = The block is in simulation	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP	Setpoint for PA devices	BYTE	16#00
SP_ST	Signal status of setpoint for PA devices	BYTE	16#00

Parameter	Description	Type	Default
SP_ST_W	Signal status of setpoint for FF devices	WORD	16#0000
SP_W	Setpoint for FF devices	WORD	16#0000

See also

FbDiOu block diagram (Page 858)
FbDiOu messaging (Page 853)
FbDiOu functions (Page 850)
FbDiOu modes (Page 850)
Description of FbDiOu (Page 847)

6.5.7 FbDiOu block diagram

FbDiOu block diagram

A block diagram is not provided for this block.

See also

FbDiOu I/Os (Page 854)
FbDiOu messaging (Page 853)
FbDiOu error handling (Page 852)
FbDiOu functions (Page 850)
FbDiOu modes (Page 850)
Description of FbDiOu (Page 847)

6.6 Pcs7AnIn - Analog input driver

6.6.1 Description of Pcs7AnIn

Object name (type + number) and family

Type + number: FB 1869

Family: Channel

Area of application for Pcs7AnIn

The block is used for the following applications:

- Signal processing of an analog input value from S7-300/400 SM analog input groups

How it works

The cyclic block processes all channel-specific signal functions of an analog input module.

It reads a raw analog value from the process image (partition) and converts it to its physical value or calculates a percentage value based on this raw value. Use the status at input parameter `Mode` to define the format of the raw value and how it is processed.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The in/out parameter `Mode` is interconnected with the corresponding `OMode_xx` output parameter of the MOD block.
- The in/out parameter `DataXchg` is interconnected with the corresponding `DataXchg_xx` output parameter of the MOD block.
- The `MS` parameter is interconnected with the `O_MS` output parameter of the diagnostics driver block.

Connect the symbol generated in HW Config (symbol table) for the input channel with the `PV_In` input parameter.

Note

If you are not using the CFC function "Generate module drivers" you must set the in/out parameter `Mode` manually. Refer to the Mode Settings for SM Modules (Page 905) section for more on this.

For the Pcs7AnIn block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Monitoring an analog process tag (AnalogMonitoring) (Page 1449)
- Cascade control (Page 1442)
- Cascade control with PIDConR (CascadeR) (Page 1444)
- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 1433)
- PID - control with operating-point-oriented parameter control (GainScheduling) (Page 1431)
- Override control (Page 1445)
- Override control with PIDConR (OverrideR) (Page 1447)
- PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 1430)
- PIDConR with safety logic and control loop monitoring (PIDConR_ConPerMon) (Page 1431)
- Ratio control (Page 1440)
- Ratio control with PIDConR (RatioR) (Page 1441)
- PID controller with Smith predictor (SmithPredictorControl) (Page 1436)
- Step controller with assigned actuator block and position feedback (StepControlActor) (Page 1437)
- Step controller with direct access to the actuator and without position feedback (StepControlDirect) (Page 1437)
- Model-based predictive control (ModPreCon) (Page 1447)
- Reversing motor with controllable speed (MotorSpeedControlled) (Page 1451)
- Dosing (DoseLean) (Page 1449)

Startup characteristics

The accept value delay is started when `CountLim` ≠ `0`.

Status word allocation for `Status` parameter

This block does not provide the `Status` parameter.

See also

Pcs7AnIn block diagram (Page 870)

Pcs7AnIn I/Os (Page 867)

Pcs7AnIn messaging (Page 867)

Pcs7AnIn error handling (Page 866)

Pcs7AnIn functions (Page 861)

Pcs7AnIn modes (Page 861)

6.6.2 Pcs7AnIn modes

Pcs7AnIn modes

This block does not provide any modes.

See also

Pcs7AnIn block diagram (Page 870)
Pcs7AnIn I/Os (Page 867)
Pcs7AnIn messaging (Page 867)
Pcs7AnIn error handling (Page 866)
Pcs7AnIn functions (Page 861)
Description of Pcs7AnIn (Page 859)
Out of service (Page 27)

6.6.3 Pcs7AnIn functions

Functions of Pcs7AnIn

The functions for this block are listed below.

Checking the raw value

The nominal range sets the range for converting analog signals into digital values (raw values), depending on the measuring type and the range of the analog input module. The nominal range is defined in the hardware configuration and is automatically saved in the in/out parameter when the block driver `Mode` is created.

This includes an overshoot/undershoot range within which an analog signal can still be converted to a digital value. This range is defined in relation to the nominal range (roughly 18.5%). Outside this range an overflow or underflow occurs and output parameter `Bad = 1` is set.

- Output parameter `PV_LoAct = 1` is set if the value is outside the nominal low range,
- Output parameter `PV_HiAct = 1` is set if the value is outside the nominal high range,

NAMUR limit checking (4 to 20 mA modules only)

In "Life Zero" monitoring the process signal is invalid (`Bad = 1`) if the measured current is less than 3.6 mA or more than 21 mA (defined by NAMUR).

The NAMUR limits are set as fixed defaults for limit monitoring. You can define other limits by setting input parameter `NamurOff = 1`, and by setting corresponding new limits in [mA] at the `HighLimit` and `LowLimit` input parameters. `Bad = 1` if a `Life Zero` analog signal is outside the current limits (`PV_HiAct` and `PV_LoAct = 1`).

Note

The limits that can be selected must lie within the overshoot and undershoot range of the module. Values outside the NAMUR range are also possible, if the module does not automatically limit the measured values.

Obtaining the standard value

The standard value (a physical quantity) is obtained from the raw value using parameters `Scale` and `Mode`. Set two scale values on the structured parameter `Scale`.

- High scale value (`Scale.High`)
- Low scale value (`Scale.Low`)

If you are not using the CFC function "Generate module drivers" you must set the in/out parameter `Mode` manually. Refer to the Mode Settings for SM Modules (Page 905) section for more on this.

The settings of the parameter `Scale` are copied to the output parameter`ScaleOut`. The output parameter can be interconnected to a corresponding input parameter of a technological block (e.g. `PV_OpScale`).

The standard value is obtained using a linear characteristic. `Scale.Low` is the lowest physical value that the process variable can take and `Scale.High` is the highest.

If `Scale.Low = 0` and `Scale.High = 100` a percentage is obtained.

Special cases when obtaining the standard value using the `Scale` parameter:

- If you set `Scale.High = Scale.Low`, you obtain the analog input module's input signal (e.g., mA) according to the `Mode` parameter setting.
- If the raw value is already a physical quantity, set `Scale.Low = 0` and `Scale.High = 1`. This will cause the raw value to be output unchanged, as a physical quantity.

- When using measuring type PTC (**P**ositive **T**emperature **C**oefficient binary evaluation of resistance thermometers), the analog value contains an encoded binary signal. The PV_Out output provides the following information:
  - If the measured resistance is within the normal range, PV_Out = 0.0.
  - If the measured resistance is in the prewarning range, PV_Out = 4.0.
  - If the measured resistance is in the operating range, PV_Out = 1.0.

    This is only true if the input parameters are Scale.Low = 0 and Scale.High = 1. During simulation or if the substitute value is output, you must only set the input parameters SimPV_In and SubsPV_In to 0.0 or 1.0.
- With the measuring type "External or internal comparison of thermocouple values", the physical unit is adapted to the ± 80 mV range in S7 300 modules. You have to determine the temperature using the corresponding conversion tables in the module manual. The physical equivalent in [mV] is returned by the module as raw value. Set Scale to ± 80 mV.

Holding the last value if raw value is invalid

If the module is to hold the most recent valid value when the raw value is invalid, you must activate this function on Feature bit Issuing last valid value if raw value is invalid (Page 191).

You can also influence this function via the input parameter DeltaVal.

- DeltaVal ≤ 0: the last value is retained and is not influenced
- DeltaVal > 0: the last or the next to last value is output

If you set the parameter DeltaVal > 0, the last PV_Out(k-1) or next to last PV_Out(k-2) valid output value is output (PV_Out(k) is the current value, k is the current time).

At parameter DeltaVal you can preset a permitted process value change (PV_Out) between two calls.

You have the following options:

- For invalid raw values and DeltaVal > 0:
  - If |PV_Out(k-1) - PV_Out(k-2)| > DeltaVal, then PV_Out = PV_Out(k-2) (last but one valid output value is output)
  - If |PV_Out(k) - PV_Out(k-1)| ≤ DeltaVal, then PV_Out = PV_Out(k-1) (last valid output value is output)
- For valid raw values and DeltaVal > 0:
  - |PV_Out(k) - PV_Out(k-1)| > DeltaVal, so for one cycle PV_Out = PV_Out(k-1) is output, i.e. DeltaVal is used to limit the change made to the valid raw value. In addition, the signal status at the output parameter PV_Out is set to 16#60 and the output parameter is set to Bad = 0.

    The value of DeltaVal should be selected with due care. If the value is too low, the quality code may flutter between16#80 and 16#60, regardless whether or not the raw value is OK.

Output substitute value if raw value is invalid

If the module is to output a substitute value `SubsPV_In` when the raw value is invalid, you must activate this function on `Feature` bit Issuing substitute value if raw value is invalid (Page 190).

Output of invalid value if raw value is invalid

If the module is to output an invalid value (`PV_Out = PV_In`), you must activate this function on `Feature` bit Output invalid raw value (Page 198).

This function is pre-selected.

Value acceptance delay

After a restart, or if the output parameter `Bad` changes its value from 1 to 0, the signal status and the value of output parameter `PV_Out` are not updated until the number of cycles for delayed acceptance of the value (input parameter `CountLim`) have elapsed. During the value acceptance delay the signal status at the output parameters is `PV_Out = 16#00` and `Bad = 1`. The last value is retained during the value acceptance delay.

If `CountLim = 0`, the function is deactivated.

Signal status for PCS7 channel blocks

The block provides the standard function Forming and outputting signal status for blocks (Page 88).

Simulating signals

The block provides the standard function Simulating signals (Page 93).

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions which are provided by the `Feature` parameter in section Functions that can be set via the Feature I/O (Page 186). The following functionality is available for this block at the relevant bits:

Bit	Function
28	Output invalid raw value (Page 198)
29	Issuing substitute value if raw value is invalid (Page 190)
30	Issuing last valid value if raw value is invalid (Page 191)

See also

Pcs7AnIn block diagram (Page 870)
Pcs7AnIn I/Os (Page 867)
Pcs7AnIn messaging (Page 867)
Pcs7AnIn error handling (Page 866)
Pcs7AnIn modes (Page 861)
Description of Pcs7AnIn (Page 859)

6.6.4 Pcs7AnIn error handling

Pcs7AnIn troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Channel error
- Higher-level error

Channel error

A channel error indicates either a broken wire or a defective sensor. Channel errors are displayed as 1 at output parameter `Bad`. They can be detected using the raw value or the NAMUR check.

`PV_LoAct` or `PV_HiAct`remain set to 1 if a channel error occurs due to the module diagnoses "Undershoot or overshoot of the measurement range“.

Higher-level error

A higher-level error is output (output parameter `ModErr = 1`) if either:

- the signal status in the High Word of input parameter `Mode` takes the value 16~40, or
- there is an invalid measuring type in the Low Word of the input parameter `Mode`.

See also

Pcs7AnIn block diagram (Page 870)

Pcs7AnIn I/Os (Page 867)

Pcs7AnIn messaging (Page 867)

Pcs7AnIn functions (Page 861)

Pcs7AnIn modes (Page 861)

Description of Pcs7AnIn (Page 859)

6.6.5 Pcs7AnIn messaging

Messaging

The block does not have any message functionality.

See also

Pcs7AnIn block diagram (Page 870)
Pcs7AnIn I/Os (Page 867)
Pcs7AnIn error handling (Page 866)
Pcs7AnIn functions (Page 861)
Pcs7AnIn modes (Page 861)
Description of Pcs7AnIn (Page 859)

6.6.6 Pcs7AnIn I/Os

Pcs7AnIn I/Os

Input parameters

Parameter	Description	Type	Default
CountLim	Startup counter limit	INT	0
DeltaVal	Delta value (PV_In - Last valid value)	REAL	0.0
EN	1 = Called block will be processed	BOOL	1
Feature	I/O for additional functions (Page 861)	STRUCT • Bit 0: BOOL • ... • Bit 28: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 1 • 0 • 0
FlutEn	1=Flutter suppression activated	BOOL	0
FlutTmIn	Flutter time: If a certain number of status changes take place within this time, fluttering is suppressed.	INT	0
HighLimit	High limit used if NAMUR check is deactivated (NamurOff = 1)	REAL	21.5
LowLimit	Low limit used if NAMUR check is deactivated (NamurOff = 1)	REAL	3.3
MS	Maintenance status	DWORD	0

Parameter	Description	Type	Default
MS_Release	Maintenance release (interconnected with MS_Release of the technological block)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
NamurOff	1 = NAMUR check deactivated	BOOL	0
PV_In	Process value (raw value)	WORD	0
PV_InUnit	Unit of measure for process value	INT	1001
Scale	Scaling of the process value as a structure	STRUCT • High: REAL • Low:REAL	- • 100.0 • 0.0
SimOn	1 = Simulation on	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SimPV_In	Process value used for SimOn = 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SubsPV_In	Substitute value	REAL	0.0

In/out parameters

Parameter	Description	Type	Default
DataXchg	Data communication	DWORD	0
Mode	Value status and measuring type	DWORD	0

Output parameters

Parameter	Description	Type	Default
Bad	1 = Process value is not valid	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see Pcs7AnIn error handling (Page 866)	INT	-1
ModErr	1 = Device/module is faulty	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosAct	1 = Field device is undergoing maintenance	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_HiAct	1 = Overshoot of process value	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_LoAct	1 = Undershoot of process value	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_Out	Standard value (physical variable)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_OutUnit	Unit of the process value	INT	0
ScaleOut	Scaling of the process value for display	STRUCT • High: REAL • Low:REAL	- • 100.0 • 0.0
SimAct	1 = Simulation active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

Pcs7AnIn block diagram (Page 870)
Pcs7AnIn messaging (Page 867)
Pcs7AnIn modes (Page 861)
Description of Pcs7AnIn (Page 859)

6.6.7 Pcs7AnIn block diagram

Pcs7AnIn block diagram

A block diagram is not provided for this block.

See also

Pcs7AnIn I/Os (Page 867)
Pcs7AnIn messaging (Page 867)
Pcs7AnIn error handling (Page 866)
Pcs7AnIn functions (Page 861)
Pcs7AnIn modes (Page 861)
Description of Pcs7AnIn (Page 859)

6.7 Pcs7AnOu - Analog output driver

6.7.1 Description of Pcs7AnOu

Object name (type + number) and family

Type + number: FB 1870

Family: Channel

Area of application for Pcs7AnOu

The block is used for the following applications:

- Signal processing of an analog output value from S7-300/400 SM analog output groups

How it works

The block outputs the process value as analog raw value for a process image (partition). Use the Mode in/out parameter to define how the raw value is to be obtained.

The current raw value is always output to the process image (partition).

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The in/out parameter `Mode` is interconnected with the corresponding `OMode_xx` output parameter of the MOD block.
- The in/out parameter `DataXchg` is interconnected with the corresponding `DataXchg_xx` output parameter of the MOD block.
- The `MS` parameter is interconnected with the `O_MS` output parameter of the diagnostics driver block.
- The `Feature` bit 0 (Setting the startup response (Page 187)) is set with a default automatically when the module driver is generated.

Connect the icon generated in HW Config (icon table) for the input channel with the `PV_Out` output parameter.

The templates of the Advanced Process Library contain an example of an Pcs7AnOu application.

Note

If you are not using the CFC function "Generate module drivers" you must set the in/out parameter `Mode` manually. Refer to the Mode Settings for SM Modules (Page 905) section for more on this.

For the Pcs7AnOu block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Cascade control (Page 1442)
- Cascade control with PIDConR (CascadeR) (Page 1444)
- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 1433)
- PID - control with operating-point-oriented parameter control (GainScheduling) (Page 1431)
- Model-based predictive control (ModPreCon) (Page 1447)
- Reversing motor with controllable speed (MotorSpeedControlled) (Page 1451)
- Override control (Page 1445)
- Override control with PIDConR (OverrideR) (Page 1447)
- PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 1430)
- PIDConR with safety logic and control loop monitoring (PIDConR_ConPerMon) (Page 1431)
- Ratio control (Page 1440)
- Ratio control with PIDConR (RatioR) (Page 1441)
- PID controller with Smith predictor (SmithPredictorControl) (Page 1436)

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

Status word allocation for `Status` parameter

This block does not provide the `Status` parameter.

See also

Pcs7AnOu block diagram (Page 880)
Pcs7AnOu I/Os (Page 877)
Pcs7AnOu messaging (Page 877)
Pcs7AnOu error handling (Page 876)
Pcs7AnOu modes (Page 873)
Pcs7AnOu functions (Page 874)

6.7.2 Pcs7AnOu modes

Pcs7AnOu modes

This block does not provide any modes.

See also

Pcs7AnOu block diagram (Page 880)
Pcs7AnOu I/Os (Page 877)
Pcs7AnOu messaging (Page 877)
Pcs7AnOu error handling (Page 876)
Description of Pcs7AnOu (Page 871)
Out of service (Page 27)
Pcs7AnOu functions (Page 874)

6.7.3 Pcs7AnOu functions

Functions of Pcs7AnOu

The functions for this block are listed below.

Forming an I/O value

The peripheral value PV_Out is formed from:

- the scale value (input parameter Scale)
- the process value (input parameter PV_In)
- the measuring type (in/out parameter Mode)

Example of measuring type 4 - 20 mA

If this measuring type is to be used you must set the Mode parameter with 16#203 accordingly. In the measuring type, the peripheral value for 4 mA is output for PV_In = Scale.Low and the peripheral value for 20 mA is output for PV_In = Scale.High.

The block writes the input parameter Scale directly to the output parameter ScaleOut and interconnects it directly to a technological block. This can be, for example, the input parameter MV_Opscale of a control block.

Limiting the process or peripheral value

The peripheral value can be limited in two different ways:

- Limited to within range limits
- Limited to scale values

Limited to within range limits (physical limits of the module): If you want to restrict the peripheral value (PV_Out), you must activate this function via the ScaleOff = 1 parameter.

The peripheral value is now limited to the following range limits:

- Top: 16#7EFF (32511 dec.)
- Bottom (unipolar): 0 or
- Bottom (unipolar; 4 - 20 mA; 1 - 5 V): 16#E500(-6912 dec.)
- Bottom (bipolar): 16#8100 (-32512 dec.)

If the limits are undershot or overshot PV_HiAct = 1 (top) or PV_LowAct = 1 (bottom) is displayed at the output parameters. The signal status of the PV_ChnST output parameter is set to 16#78.

Limited to scale values: If you want to restrict the peripheral value (PV_Out) to the scale values, you must activate this function via the ScaleOff = 0 parameter. You define the high and low scale limits in the Scale parameter. If one of the limits is violated, the limit you have entered is output at the PV_Out output parameter. This is displayed at the PV_HiAct or PV_LoAct = 1 output parameter. The signal status of the PV_ChnSToutput parameter is set to 16#78.

Simulating signals

The block provides the standard function Simulating signals (Page 93).

Forming the signal status for PCS7 channel blocks

The block provides the standard function Forming and outputting signal status for blocks (Page 88).

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section.

The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)

See also

Pcs7AnOu block diagram (Page 880)

Pcs7AnOu I/Os (Page 877)

Pcs7AnOu messaging (Page 877)

Pcs7AnOu error handling (Page 876)

Pcs7AnOu modes (Page 873)

Description of Pcs7AnOu (Page 871)

6.7.4 Pcs7AnOu error handling

Pcs7AnOu troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Higher-level error
- Configuration error

Higher-level error

A higher-level error is output (output parameter `ModErr = 1`) if either:

- the signal status in the High Word of input parameter `Mode` takes the value 16~40, or
- there is an invalid measuring type in the Low Word of the input parameter `Mode`.

Configuration error

If parameter `Mode` is set incorrectly, this is indicated at output parameter `Bad` with the value 1.

See also

Pcs7AnOu block diagram (Page 880)

Pcs7AnOu I/Os (Page 877)

Pcs7AnOu messaging (Page 877)

Pcs7AnOu modes (Page 873)

Description of Pcs7AnOu (Page 871)

Pcs7AnOu functions (Page 874)

6.7.5 Pcs7AnOu messaging

Message functionality

The block does not have any message functionality.

See also

Pcs7AnOu block diagram (Page 880)
Pcs7AnOu I/Os (Page 877)
Pcs7AnOu error handling (Page 876)
Pcs7AnOu modes (Page 873)
Description of Pcs7AnOu (Page 871)
Pcs7AnOu functions (Page 874)

6.7.6 Pcs7AnOu I/Os

Pcs7AnOu I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
Feature	I/O for additional functions (Page 874)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
FlutEn	1=Flutter suppression activated	BOOL	0
FlutTmIn	Flutter time: If a certain number of status changes take place within this time, fluttering is suppressed.	INT	0
MS	Maintenance status	DWORD	0
MS_Release	Maintenance release (interconnected with MS_Release of the technological block)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_In	Process value (raw value)	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_InUnit	Unit of measure for process value	INT	1342

Parameter	Description	Type	Default
`Scale`	Scaling of the process value as a structure	STRUCT • High: REAL • Low:REAL	- • 100.0 • 0.0
`ScaleOff`	0 = Limitation to scale limits (`Scale`) active 1 = Limitation to within range limits (physical limits of the module) active	BOOL	0
`SimOn`	1 = Simulation on	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
`SimPV_In`	Process value used for `SimOn = 1`	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
`StartVal`	Starting value that is used on starting the block if `Feature` Bit0 = 1.	REAL	0.0

In/out parameters

Parameter	Description	Type	Default
`DataXchg`	Data communication	DWORD	0
`Mode`	Value status and measuring type	DWORD	0

Output parameters

Parameter	Description	Type	Default
Bad	1 = Process value is not valid	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Reserved for future error numbers	INT	-1
ModErr	1 = Device/module is faulty	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosAct	1 = Field device is undergoing maintenance	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_ChnST	Signal status of the output channel and value of PV_Out	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_HiAct	1 = Overshoot of process value	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_LoAct	1 = Undershoot of process value	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_Out	Process value	WORD	0
PV_OutUnit	Unit of the process value	INT	0
ScaleOut	Scaling of the process value as a structure	STRUCT • High: REAL • Low:REAL	- • 100.0 • 0.0
SimAct	1 = Simulation active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

Pcs7AnOu block diagram (Page 880)
Pcs7AnOu messaging (Page 877)
Pcs7AnOu error handling (Page 876)
Pcs7AnOu modes (Page 873)
Description of Pcs7AnOu (Page 871)

6.7.7 Pcs7AnOu block diagram

Pcs7AnOu block diagram

This block does not come with a block diagram.

See also

Pcs7AnOu I/Os (Page 877)
Pcs7AnOu messaging (Page 877)
Pcs7AnOu error handling (Page 876)
Pcs7AnOu modes (Page 873)
Description of Pcs7AnOu (Page 871)
Pcs7AnOu functions (Page 874)

6.8 Pcs7DiIn - Digital input driver

6.8.1 Description of Pcs7DiIn

Object name (type + number) and family

Type + number: FB 1871

Family: Channel

Area of application for Pcs7DiIn

The block is used for the following applications:

- Signal processing of an digital input value from S7-300/400 SM digital input groups

How it works

The cyclic block processes all channel-specific signal functions of a digital input module.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The in/out parameter `Mode` is interconnected with the corresponding `OMode_xx` output parameter of the MOD block.
- The in/out parameter `DataXchg` is interconnected with the corresponding `DataXchg_xx` output parameter of the MOD block.
- The `MS` parameter is interconnected with the `O_MS` output parameter of the diagnostics driver block.

Connect the symbol generated in HW Config (symbol table) for the input channel with the `PV_In` input parameter.

Note

If you are not using the CFC function "Generate module drivers" you must set the in/out parameter `Mode` manually. Refer to the Mode Settings for SM Modules (Page 905) section for more on this.

For the Pcs7DiIn block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Monitoring eight digital process tags (Digital8Monitoring) (Page 1448)
- Monitoring of a digital process tag (DigitalMonitoring) (Page 1448)
- Reversing motor with controllable speed (MotorSpeedControlled) (Page 1451)
- Step controller with assigned actuator block and position feedback (StepControlActor) (Page 1437)
- Step controller with direct access to the actuator and without position feedback (StepControlDirect) (Page 1437)
- Two-speed motor (Motor2Speed) (Page 1450)
- Reversing motor (MotorReversible) (Page 1450)
- Valve (ValveLean) (Page 1453)
- Two-way valve (Valve2Way) (Page 1453)
- Motor valve (ValveMotor) (Page 1454)

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for `Status` parameter

This block does not provide the `Status` parameter.

See also

Pcs7DiIn block diagram (Page 888)

Pcs7DiIn I/Os (Page 886)

Pcs7DiIn messaging (Page 885)

Pcs7DiIn error handling (Page 885)

Pcs7DiIn functions (Page 883)

Pcs7DiIn modes (Page 883)

6.8.2 Pcs7DiIn modes

Pcs7DiIn modes

This block does not provide any modes.

See also

Pcs7DiIn block diagram (Page 888)
Pcs7DiIn I/Os (Page 886)
Pcs7DiIn messaging (Page 885)
Pcs7DiIn error handling (Page 885)
Pcs7DiIn functions (Page 883)
Description of Pcs7DiIn (Page 881)

6.8.3 Pcs7DiIn functions

Functions of Pcs7DiIn

The functions for this block are listed below.

Obtaining the standard value

The digital value of the process image (partition) is sent to output parameter `PV_Out` with the signal status 16#80.

If you are not using the CFC function "Generate module drivers" you must set the in/out parameter `Mode` manually for further settings. Refer to the Mode Settings for SM Modules (Page 905) section for more on this.

Holding the last value if raw value is invalid

If the module is to hold the most recent valid value when the raw value is invalid, you must activate this function on `Feature` bit Issuing last valid value if raw value is invalid (Page 191).

Output substitute value if raw value is invalid

If the module is to output a substitute value `SubsPV_In` when the raw value is invalid, you must activate this function on `Feature` bit Issuing substitute value if raw value is invalid (Page 190).

Output of invalid value if raw value is invalid

If the module is to output an invalid value (`PV_Out = PV_In`), you must activate this function on `Feature` bit Output invalid raw value (Page 198).

This function is pre-selected.

Signal status for PCS7 channel blocks

The block provides the standard function Forming and outputting signal status for blocks (Page 88).

Simulating signals

The block provides the standard function Simulating signals (Page 93).

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
28	Output invalid raw value (Page 198)
29	Issuing substitute value if raw value is invalid (Page 190)
30	Issuing last valid value if raw value is invalid (Page 191)

See also

Pcs7DiIn block diagram (Page 888)

Pcs7DiIn I/Os (Page 886)

Pcs7DiIn messaging (Page 885)

Pcs7DiIn error handling (Page 885)

Pcs7DiIn modes (Page 883)

Description of Pcs7DiIn (Page 881)

6.8.4 Pcs7DiIn error handling

Pcs7DiIn troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Higher-level error

Higher-level error

A higher-level error is output (output parameter `ModErr = 1`) if either:

- the signal status in the High Word of input parameter `Mode` takes the value 16~40, or

an invalid measuring type in the Low Word

See also

Pcs7DiIn block diagram (Page 888)
Pcs7DiIn I/Os (Page 886)
Pcs7DiIn messaging (Page 885)
Pcs7DiIn functions (Page 883)
Pcs7DiIn modes (Page 883)
Description of Pcs7DiIn (Page 881)

6.8.5 Pcs7DiIn messaging

Messaging

The block does not have any message functionality.

See also

Pcs7DiIn block diagram (Page 888)
Pcs7DiIn I/Os (Page 886)
Pcs7DiIn error handling (Page 885)
Pcs7DiIn functions (Page 883)
Pcs7DiIn modes (Page 883)
Description of Pcs7DiIn (Page 881)

6.8.6 Pcs7DiIn I/Os

Pcs7DiIn I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
Feature	I/O for additional functions (Page 883)	STRUCT • Bit 0: BOOL • ... • Bit 28: BOOL • .... • Bit 31: BOOL	- • 0 • 0 • 1 • 0 • 0
FlutEn	1=Flutter suppression activated	BOOL	0
FlutTmIn	Flutter time: If a certain number of status changes take place within this time, fluttering is suppressed.	INT	0
MS	Maintenance status	DWORD	0
MS_Release	Maintenance release (interconnected with MS_Release of the technological block)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ProlmQB	Quality information from the process image	BOOL	0
PV_In	Process value (raw value)	BOOL	0
SelQB	1 = Use the quality bit from the process image	BOOL	0
SimOn	1 = Simulation on	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SimPV_In	Process value used for SimOn = 1	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
SubsPV_In	Substitute value	BOOL	0

In/out parameters

Parameter	Description	Type	Default
DataXchg	Data communication	DWORD	0
Mode	Value status and measuring type	DWORD	0

Output parameters

Parameter	Description	Type	Default
Bad	1 = Process value is not valid	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see Pcs7DiIn error handling (Page 885)	INT	-1
ModErr	1 = Device/module is faulty	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosAct	1 = Field device is undergoing maintenance	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_Out	Standard value (physical variable)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SimAct	1 = Simulation active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

Pcs7DiIn block diagram (Page 888)

Pcs7DiIn messaging (Page 885)

Pcs7DiIn functions (Page 883)

Description of Pcs7DiIn (Page 881)

6.8.7 Pcs7DiIn block diagram

Pcs7DiIn block diagram

A block diagram is not provided for this block.

See also

Pcs7DiIn I/Os (Page 886)
Pcs7DiIn messaging (Page 885)
Pcs7DiIn error handling (Page 885)
Pcs7DiIn functions (Page 883)
Pcs7DiIn modes (Page 883)
Description of Pcs7DiIn (Page 881)

6.9 Pcs7DiIT - Digital input driver with time stamp

6.9.1 Description of Pcs7DiIT

Object name (type + number) and family

Type + number: FB 1872

Family: Channel

Area of application for Pcs7DiIT

The block is used for the following applications:

- Signal processing of a digital input value from S7-300/400 SM digital input modules with time stamp

How it works

The cyclic block processes all channel-specific signal functions of a digital input module with configured time stamp.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

Connect the symbol generated in HW Config (symbol table) for the input channel with the `PV_In` input parameter.

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The in/out parameter `Mode` is interconnected with the corresponding `OMode_xx` output parameter of the MOD block.
- The in/out parameter `DataXchg` is interconnected with the corresponding `DataXchg_xx` output parameter of the MOD block.
- The `MS` parameter is interconnected with the `O_MS` output parameter of the diagnostics driver block.
- If the process image (partition) also contains the value status (status bit) of the digital input channel, the corresponding symbol is connected with input `ProImQB` and the input `SelQB = 1` set.

- The TS_In parameter is interconnected with the TS_XX output parameter of the IMDRV_TS block.
- The in/out parameter TS_C is interconnected with the corresponding TS_C_xx output parameter of the IMDRV_TS block.

Note

If you are not using the CFC function "Generate module drivers" you must set the in/out parameter Mode manually. Refer to the section Mode Settings for SM Modules (Page 905).

Consult the section on configuring the channel drivers for information on configuring.

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status parameter

This block does not provide the Status parameter.

See also

Pcs7DiIT block diagram (Page 896)

Pcs7DiIT I/Os (Page 894)

Pcs7DiIT messaging (Page 893)

Pcs7DiIT error handling (Page 893)

Pcs7DiIT functions (Page 891)

Pcs7DiIT modes (Page 891)

6.9.2 Pcs7DiIT modes

Pcs7DiIT modes

This block does not provide any modes.

See also

Pcs7DiIT block diagram (Page 896)

Pcs7DiIT I/Os (Page 894)

Pcs7DiIT messaging (Page 893)

Pcs7DiIT error handling (Page 893)

Pcs7DiIT functions (Page 891)

Description of Pcs7DiIT (Page 889)

Out of service (Page 27)

6.9.3 Pcs7DiIT functions

Functions of Pcs7DiIT

The functions for this block are listed below.

Obtaining the standard value

The digital value of the process image (partition) is sent to output parameter `PV_Out` with the signal status 16#80.

If you are not using the CFC function "Generate module drivers" you must set the in/out parameter `Mode` manually for further settings. Refer to the section Mode Settings for SM Modules (Page 905).

Holding the last value if raw value is invalid

If the block is to hold the most recent valid value when the raw value is invalid, you must activate this function at the `Feature` bit Issuing last valid value if raw value is invalid (Page 191).

Output substitute value if raw value is invalid

If the block is to output a substitute value (`SubsPV_In`) if the raw value is invalid, you must activate this function at the `Feature` bit Issuing substitute value if raw value is invalid (Page 190).

Output of invalid value if raw value is invalid

If the block is to output an invalid value (`PV_Out = PV_In`), you must activate this function at the `Feature` bit Output invalid raw value (Page 198).

This function is pre-selected.

Signal status for PCS7 channel blocks

The block provides the standard function Forming and outputting signal status for blocks (Page 88).

Simulating signals

The block provides the standard function Simulating signals (Page 93).

Time stamp

The block provides the standard function Time stamp (Page 51).

Interconnect the signal with the time stamp from the I/O devices with input parameter `TS_In`.

Interconnect input parameter `TS_In` with the channel-specific output parameter `TS_Oxx` of block IMDRV_TS.

Interconnect in/out parameter `TS_C` with the channel-specific output parameter `TS_Cxx` of block IMDRV_TS. This happens automatically when the CFC function "Generate module drivers" is used.

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the Feature parameter in the section Functions that can be set via the Feature I/O (Page 186). The following functionality is available for this block at the relevant bits:

Bit	Function
28	Output invalid raw value (Page 198)
29	Issuing substitute value if raw value is invalid (Page 190)
30	Issuing last valid value if raw value is invalid (Page 191)

See also

Pcs7DiIT block diagram (Page 896)

Pcs7DiIT I/Os (Page 894)

Pcs7DiIT messaging (Page 893)

Pcs7DiIT error handling (Page 893)

Pcs7DiIT modes (Page 891)

Description of Pcs7DiIT (Page 889)

6.9.4 Pcs7DiIT error handling

Pcs7DiIT error handling

The block does not report any errors.

See also

Pcs7DiIT block diagram (Page 896)
Pcs7DiIT I/Os (Page 894)
Pcs7DiIT messaging (Page 893)
Pcs7DiIT functions (Page 891)
Pcs7DiIT modes (Page 891)
Description of Pcs7DiIT (Page 889)

6.9.5 Pcs7DiIT messaging

Messaging

The block does not have any message functionality.

See also

Pcs7DiIT block diagram (Page 896)
Pcs7DiIT I/Os (Page 894)
Pcs7DiIT error handling (Page 893)
Pcs7DiIT functions (Page 891)
Pcs7DiIT modes (Page 891)
Description of Pcs7DiIT (Page 889)

6.9.6 Pcs7DiIT I/Os

Pcs7DiIT I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
Feature	I/O for additional functions (Page 891)	STRUCT • Bit 0: BOOL • ... • Bit 28: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 1 • 0 • 0
FlutEn	1=Flutter suppression activated	BOOL	0
FlutTmIn	Flutter time: If a certain number of status changes take place within this time, fluttering is suppressed.	INT	0
MS	Maintenance status	DWORD	0
MS_Release	Maintenance release (interconnected with MS_Release of the technological block)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_In	Process value (raw value)	BOOL	0
ProlmQB	Quality information from the process image	BOOL	0
SimOn	1 = Simulation on	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SimPV_In	Process value used for SimOn = 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SelQB	1 = Use the quality bit from the process image	BOOL	0
SubsPV_In	Substitute value	BOOL	0
TimeStampOn	1 = Time stamp on, time stamp received via input parameter TS_In	BOOL	1
TS_In	Time stamps of IMDRV_TS The time is in ISP format	STRUCT • MsgSig: BOOL • TriInf: BOOL • HdSh: BOOL • ST: BYTE • TS0: DWORD • TS1: DWORD	- • 0 • 0 • 0 • 16#80 • 16#0 • 16#0

In/out parameters

Parameter	Description	Type	Default
DataXchg	Data communication	DWORD	0
Mode	Value status and measuring type	DWORD	0
TS_C	Time-stamp communication	BYTE	16#00

Output parameters

Parameter	Description	Type	Default
Bad	1 = Process value is not valid	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Reserved for future error numbers	INT	-1
ModErr	1 = Device/module is faulty	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosAct	1 = Field device is undergoing maintenance	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_Out	1 = Standard value (physical variable)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SimAct	1 = Simulation active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
TS_C_Out	Time stamp communication	BYTE	16#00
TS_Out	Time stamp	STRUCT • MsgSig: BOOL • TriInf: BOOL • HdSh: BOOL • ST: BYTE • TS0: DWORD • TS1: DWORD	- • 0 • 0 • 0 • 16#80 • 16#0 • 16#0

See also

Pcs7DiIT block diagram (Page 896)
Pcs7DiIT messaging (Page 893)
Pcs7DiIT error handling (Page 893)
Pcs7DiIT modes (Page 891)
Description of Pcs7DiIT (Page 889)

6.9.7 Pcs7DiIT block diagram

Pcs7DiIT block diagram

This block does not come with a block diagram.

See also

Pcs7DiIT I/Os (Page 894)
Pcs7DiIT messaging (Page 893)
Pcs7DiIT error handling (Page 893)
Pcs7DiIT functions (Page 891)
Pcs7DiIT modes (Page 891)
Description of Pcs7DiIT (Page 889)

6.10 Pcs7DiOu - Digital output driver

6.10.1 Description of Pcs7DiOu

Object name (type + number) and family

Type + number: FB 1873

Family: Channel

Area of application for Pcs7DiOu

The block is used for the following applications:

- Signal processing of a digital output value from S7-300/400 SM digital output groups.

How it works

The cyclic block processes all channel-specific signal functions of a digital output module.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

When the CFC function "Generate module drivers" is used, the following occurs automatically:

- The in/out parameter `Mode` is interconnected with the corresponding `OMode_xx` output parameter of the MOD block.
- The in/out parameter `DataXchg` is interconnected with the corresponding `DataXchg_xx` output parameter of the MOD block.
- The `MS` parameter is interconnected with the `O_MS` output parameter of the diagnostics driver block.
- The `Feature` bit 0 (Setting the startup response (Page 187)) is set with a default automatically when the module driver is generated.

Connect the symbol generated in HW Config (symbol table) for the input channel with the `PV_In` input parameter.

The templates of the Advanced Process Library contain an example of an Pcs7DiOu application.

Note

If you are not using the CFC function "Generate module drivers" you must set the in/out parameter `Mode` manually. Refer to the Mode Settings for SM Modules (Page 905) section for more on this.

For the Pcs7DiOu block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Dosing (DoseLean) (Page 1449)
- Two-speed motor (Motor2Speed) (Page 1450)
- Reversing motor (MotorReversible) (Page 1450)
- Reversing motor with controllable speed (MotorSpeedControlled) (Page 1451)
- Step controller with assigned actuator block and position feedback (StepControlActor) (Page 1437)
- Step controller with direct access to the actuator and without position feedback (StepControlDirect) (Page 1437)
- Valve (ValveLean) (Page 1453)
- Two-way valve (Valve2Way) (Page 1453)
- Motor valve (ValveMotor) (Page 1454)

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

Status word allocation for `Status` parameter

This block does not provide the `Status` parameter.

See also

Pcs7DiOu block diagram (Page 904)

Pcs7DiOu I/Os (Page 902)

Pcs7DiOu messaging (Page 902)

Pcs7DiOu error handling (Page 901)

Pcs7DiOu functions (Page 899)

Pcs7DiOu modes (Page 899)

6.10.2 Pcs7DiOu modes

Pcs7DiOu modes

The block does not have any operating modes.

See also

Pcs7DiOu block diagram (Page 904)

Pcs7DiOu I/Os (Page 902)

Pcs7DiOu messaging (Page 902)

Pcs7DiOu error handling (Page 901)

Pcs7DiOu functions (Page 899)

Description of Pcs7DiOu (Page 897)

6.10.3 Pcs7DiOu functions

Functions of Pcs7DiOu

The functions for this block are listed below.

Forming an I/O value

The digital value is written to the process image (partition) and the signal status at output parameter `PV_Out` is set to "good" (16#80).

Simulating signals

The block provides the standard function Simulating signals (Page 93).

Configurable reactions using the `Feature` parameter

You can find an overview of all behavior patterns which are provided by the `Feature` parameter in section Functions that can be set via the Feature I/O (Page 186).

The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)

See also

Pcs7DiOu block diagram (Page 904)
Pcs7DiOu I/Os (Page 902)
Pcs7DiOu messaging (Page 902)
Pcs7DiOu error handling (Page 901)
Pcs7DiOu modes (Page 899)
Description of Pcs7DiOu (Page 897)

6.10.4 Pcs7DiOu error handling

Pcs7DiOu troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Higher-level error

Higher-level error

A higher-level error is output (output parameter `ModErr = 1`) if either:

- the signal status in the High Word of input parameter `Mode` takes the value 16~40, or

there is an invalid measuring type in the Low Word of the input parameter `Mode`.

See also

Pcs7DiOu block diagram (Page 904)
Pcs7DiOu I/Os (Page 902)
Pcs7DiOu messaging (Page 902)
Pcs7DiOu functions (Page 899)
Pcs7DiOu modes (Page 899)
Description of Pcs7DiOu (Page 897)

6.10.5 Pcs7DiOu messaging

Messaging

The block does not have any message functionality.

See also

Pcs7DiOu block diagram (Page 904)
Pcs7DiOu I/Os (Page 902)
Pcs7DiOu error handling (Page 901)
Pcs7DiOu functions (Page 899)
Pcs7DiOu modes (Page 899)
Description of Pcs7DiOu (Page 897)

6.10.6 Pcs7DiOu I/Os

Pcs7DiOu I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
Feature	I/O for additional functions (Page 899)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
FlutEn	1=Flutter suppression activated	BOOL	0
FlutTmIn	Flutter time: If a certain number of status changes take place within this time, fluttering is suppressed.	INT	0
MS	Maintenance status	DWORD	0
MS_Release	Maintenance release (interconnected with MS_Release of the technological block)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_In	Process value (raw value)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SimOn	1 = Simulation on	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
SimPV_In	Process value used for SimOn = 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StartVal	Starting value that is used on starting the block if Feature Bit0 = 1.	BOOL	0

In/out parameters

Parameter	Description	Type	Default
DataXchg	Data communication	DWORD	0
Mode	Value status and measuring type	DWORD	0

Output parameters

Parameter	Description	Type	Default
Bad	1 = Process value is not valid	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see Pcs7DiOu error handling (Page 901)	INT	-1
ModErr	1 = Device/module is faulty	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosAct	1 = Field device is undergoing maintenance	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_ChnST	Signal status of the output channel and value of PV_Out	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_Out	Process value	BOOL	0
SimAct	1 = Simulation active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

Pcs7DiOu block diagram (Page 904)
Pcs7DiOu messaging (Page 902)
Pcs7DiOu modes (Page 899)
Description of Pcs7DiOu (Page 897)

6.10.7 Pcs7DiOu block diagram

Pcs7DiOu block diagram

This block does not come with a block diagram.

See also

Pcs7DiOu I/Os (Page 902)
Pcs7DiOu messaging (Page 902)
Pcs7DiOu error handling (Page 901)
Pcs7DiOu functions (Page 899)
Pcs7DiOu modes (Page 899)
Description of Pcs7DiOu (Page 897)

6.11 Annex for channel blocks

6.11.1 Mode Settings for SM Modules

Measuring range coding of the analog input modules

Depending on the measuring range coding of the analog input modules, the parameter Mode (measuring-range coding) corresponding to the channel must be specified in accordance with the table. When thermocouples are used there are various options for combining the measurement type (coding A) with the measuring range (coding B). In this case, parameter Mode must be calculated according to the following formula and the result entered at the Mode input parameter as an INTEGER value:

Mode = 256 · code A + code B

Please note: The table displays codes **A** and **B** in binary format, and the result in hexadecimal format in the Mode parameter.

Measuring type	Code (A)	Measuring range	Code(B)	Mode (256*A+B)
Deactivated				16#0000
Voltage	2#0001	± 25 mV	2#1010	16#010A
		± 50 mV	2#1011	16#010B
		± 80 mV	2#0001	16#0101
		± 250 mV	2#0010	16#0102
		± 500 mV	2#0011	16#0103
		± 1 V	2#0100	16#0104
		± 2.5 V	2#0101	16#0105
		± 5 V	2#0110	16#0106
		1 to 5 V	2#0111	16#0107
		0 to 10 V	2#1000	16#0108
		± 10 V	2#1001	16#0109
		± 100 mV	2#1100	16#010C
4-wire measuring transducer	2#0010	± 3.2 mA	2#0000	16#0200
		± 5 mA	2#0101	16#0205
		± 10 mA	2#0001	16#0201
		0 to 20 mA	2#0010	16#0202
		4 to 20 mA	2#0011	16#0203
		± 20 mA	2#0100	16#0204
HART interface	2#0111	4 to 20 mA	2#1100	16#070C
2-wire measuring transducer	2#0011	4 to 20 mA	2#0011	16#0303
		± 20 mA	2#0100	16#0204

Measuring type	Code (A)	Measuring range	Code(B)	Mode (256*A+B)
Resistor 4-wire connection	2#0100	48 Ω	2#0000	16#0400
		150 Ω	2#0010	16#0402
		300 Ω	2#0100	16#0404
		600 Ω	2#0110	16#0406
		1000 Ω	2#0111	16#040E
		3000 Ω	2#0111	16#0407
		6000 Ω	2#1000	16#0408
		PTC	2#1111	16#040F
Resistor 3-wire connection	2#0101	48 Ω	2#0000	16#0500
		150 Ω	2#0010	16#0502
		300 Ω	2#0100	16#0504
		600 Ω	2#0110	16#0506
		1000 Ω	2#0111	16#050E
		3000 Ω	2#0111	16#0507
		6000 Ω	2#1000	16#0508
		PTC	2#1111	16#050F
Resistor 2-wire connection	2#0110	48 Ω	2#0000	16#0600
		150 Ω	2#0010	16#0602
		300 Ω	2#0100	16#0604
		600 Ω	2#0110	16#0606
		1000 Ω	2#0111	16#060E
		3000 Ω	2#0111	16#0607
		6000 Ω	2#1000	16#0608
		PTC	2#1111	16#060F
Thermocouple, linear, 4-wire connection	2#1000	Pt 100 climate range	2#0000	16#0800
		Pt 200 climate range	2#0111	16#0807
		Pt 500 climate range	2#1000	16#0808
		Pt 1000 climate range	2#1001	16#0809
		Ni 100 climate range	2#0001	16#0801
		Ni 1000 climate range	2#1010	16#080A
		Pt 100 standard range	2#0010	16#0802
		Pt 200 standard range	2#0011	16#0803
		Pt 500 standard range	2#0100	16#0804
		Pt 1000 standard range	2#0101	16#0805
		Ni 100 standard range	2#1011	16#080B
		Ni 1000 standard range	2#0110	16#0806
		Ni 120 standard range	2#1100	16#080C
		Ni 120 climate range	2#1101	16#080D
		Cu 10 climate range	2#1110	16#080E
		Cu 10 standard range	2#1111	16#080F
		Ni 200 standard range	2#10000	16#0810
		Ni 200 climate range	2#10001	16#0811

Measuring type	Code (A)	Measuring range	Code(B)	Mode (256*A+B)
		Ni 500 standard range	2#10010	16#0812
		Ni 500 climate range	2#10011	16#0813
		Pt 10 GOST climatic	2#10100	16#0814
		Pt 10 GOST standard (TC = 3910)	2#10101	16#0815
		Pt 50 GOST climatic	2#10110	16#0816
		Pt 50 GOST standard (TC = 3910)	2#10111	16#0817
		Pt 100 GOST climatic	2#11000	16#0818
		Pt 100 GOST standard (TC = 3910)	2#11001	16#0819
		Pt 500 GOST climatic	2#11010	16#081A
		Pt 500 GOST standard (TC = 3910)	2#11011	16#081B
		Cu 10 GOST climatic	2#11100	16#081C
		Cu 10 GOST standard (TC = 426)	2#11101	16#081D
		Cu 50 GOST climatic	2#11110	16#081E
		Cu 50 GOST standard (TC = 426)	2#11111	16#081F
		Cu 100 GOST climatic	2#100000	16#0820
		Cu 100 GOST standard (TC = 426)	2#100001	16#0821
		Ni 100 GOST climatic	2#100010	16#0822
		Ni 100 GOST standard	2#100011	16#0823
		Pt 10 GOST standard (TC = 3850)	2#1010101	16#0855
		Pt 50 GOST standard (TC = 3850)	2#1010111	16#0857
		Pt 100 GOST standard (TC = 3850)	2#1011001	16#0859
		Pt 500 GOST standard (TC = 3850)	2#1011011	16#085B
		Cu 10 GOST standard (TC = 428)	2#10011101	16#089D
		Cu 50 GOST standard (TC = 428)	2#10011111	16#089F
		Cu 100 GOST standard (TC = 428)	2#10100001	16#08A1

Measuring type	Code (A)	Measuring range	Code(B)	Mode (256*A+B)
Thermocouple, linear, 3-wire connection	2#1001	Pt 100 climate range	2#0000	16#0900
		Pt 200 climate range	2#0111	16#0907
		Pt 500 climate range	2#1000	16#0908
		Pt 1000 climate range	2#1001	16#0909
		Ni 100 climate range	2#0001	16#0901
		Ni 1000 climate range	2#1010	16#090A
		Pt 100 standard range	2#0010	16#0902
		Pt 200 standard range	2#0011	16#0903
		Pt 500 standard range	2#0100	16#0904
		Pt 1000 standard range	2#0101	16#0905
		Ni 100 standard range	2#1011	16#090B
		Ni 1000 standard range	2#0110	16#0906
		Ni 120 standard range	2#1100	16#090C
		Ni 120 climate range	2#1101	16#090D
		Cu10 climate range	2#1110	16#090E
		Cu10 standard range	2#1111	16#090F
		Ni 200 standard range	2#10000	16#0910
		Ni 200 climate range	2#10001	16#0911
		Ni 500 standard range	2#10010	16#0912
		Ni 500 climate range	2#10011	16#0913
		Pt 10 GOST climatic	2#10100	16#0914
		Pt 10 GOST standard (TC = 3910)	2#10101	16#0915
		Pt 50 GOST climatic	2#10110	16#0916
		Pt 50 GOST standard (TC = 3910)	2#10111	16#0917
		Pt 100 GOST climatic	2#11000	16#0918
		Pt 100 GOST standard (TC = 3910)	2#11001	16#0919
		Pt 500 GOST climatic	2#11010	16#091A
		Pt 500 GOST standard (TC = 3910)	2#11011	16#091B
		Cu 10 GOST climatic	2#11100	16#091C
		Cu 10 GOST standard (TC = 426)	2#11101	16#091D
		Cu 50 GOST climatic	2#11110	16#091E
		Cu 50 GOST standard (TC = 426)	2#11111	16#091F
		Cu 100 GOST climatic	2#100000	16#0920
		Cu 100 GOST standard (TC = 426)	2#100001	16#0921
		Ni 100 GOST climatic	2#100010	16#0922
		Ni 100 GOST standard	2#100011	16#0923

Measuring type	Code (A)	Measuring range	Code(B)	Mode (256*A+B)
		Pt 10 GOST standard (TC = 3850)	2#1010101	16#0955
		Pt 50 GOST standard (TC = 3850)	2#1010111	16#0957
		Pt 100 GOST standard (TC = 3850)	2#1011001	16#0959
		Pt 500 GOST standard (TC = 3850)	2#1011011	16#095B
		Cu 10 GOST standard (TC = 428)	2#10011101	16#099D
		Cu 50 GOST standard (TC = 428)	2#10011111	16#099F
		Cu 100 GOST standard (TC = 428)	2#10100001	16#09A1
Thermocouple, linear, 2-wire connection	2#1111	Pt 100 climate range	2#0000	16#0F00
		Pt 200 climate range	2#0111	16#0F07
		Pt 500 climate range	2#1000	16#0F08
		Pt 1000 climate range	2#1001	16#0F09
		Ni 100 climate range	2#0001	16#0F01
		Ni 1000 climate range	2#1010	16#0F0A
		Pt 100 standard range	2#0010	16#0F02
		Pt 200 standard range	2#0011	16#0F03
		Pt 500 standard range	2#0100	16#0F04
		Pt 1000 standard range	2#0101	16#0F05
		Ni 100 standard range	2#1011	16#0F0B
		Ni 1000 standard range	2#0110	16#0F06
		Ni 120 standard range	2#1100	16#0F0C
		Ni 120 climate range	2#1101	16#0F0D
		Cu10 climate range	2#1110	16#0F0E
		Cu10 standard range	2#1111	16#0F0F
		Ni 200 standard range	2#10000	16#0F10
		Ni 200 climate range	2#10001	16#0F11
		Ni 500 standard range	2#10010	16#0F12
		Ni 500 climate range	2#10011	16#0F13

Measuring type	Code (A)	Measuring range	Code(B)	Mode (256*A+B)
Thermocouple, linear, reference temperature 0 ºC	2#1010	Type B [PtRh-PtRh]	2#0000	16#0A00
		Type N [NiCrSi-NiSi]	2#0001	16#0A01
		Type E [NiCr-CuNi]	2#0010	16#0A02
		Type R [PtRh-Pt]	2#0011	16#0A03
		Type S [PtRh-Pt]	2#0100	16#0A04
		Type J [Fe-CuNi IEC]	2#0101	16#0A05
		Type L [Fe-CuNi DIN]	2#0110	16#0A06
		Type T [Cu-CuNi IEC]	2#0111	16#0A07
		Type K [NiCr-Ni]	2#1000	16#0A08
		Type U [Cu-CuNi DIN]	2#1001	16#0A09
		Type C	2#1010	16#0A0A
		Type TXK/XK(L)	2#1011	16#0A0B
Thermocouple, linear, reference temperature 50 ºC	2#1011	Type B [PtRh-PtRh]	2#0000	16#0B00
		Type N [NiCrSi-NiSi]	2#0001	16#0B01
		Type E [NiCr-CuNi]	2#0010	16#0B02
		Type R [PtRh-Pt]	2#0011	16#0B03
		Type S [PtRh-Pt]	2#0100	16#0B04
		Type J [Fe-CuNi IEC]	2#0101	16#0B05
		Type L [Fe-CuNi DIN]	2#0110	16#0B06
		Type T [Cu-CuNi IEC]	2#0111	16#0B07
		Type K [NiCr-Ni]	2#1000	16#0B08
		Type U [Cu-CuNi DIN]	2#1001	16#0B09
		Type C	2#1010	16#0B0A
		Type TXK/XK(L)	2#1011	16#0B0B
Thermocouple, linear, internal compensation	2#1101	Type B [PtRh-PtRh]	2#0000	16#0D00
		Type N [NiCrSi-NiSi]	2#0001	16#0D01
		Type E [NiCr-CuNi]	2#0010	16#0D02
		Type R [PtRh-Pt]	2#0011	16#0D03
		Type S [PtRh-Pt]	2#0100	16#0D04
		Type J [Fe-CuNi IEC]	2#0101	16#0D05
		Type L [Fe-CuNi DIN]	2#0110	16#0D06
		Type T [Cu-CuNi IEC]	2#0111	16#0D07
		Type K [NiCr-Ni]	2#1000	16#0D08
		Type U [Cu-CuNi DIN]	2#1001	16#0D09
		Type C	2#1010	16#0D0A
		Type TXK/XK(L)	2#1011	16#0D0B

Measuring type	Code (A)	Measuring range	Code(B)	Mode (256*A+B)
Thermocouple, linear, external compensation	2#1110	Type B [PtRh-PtRh]	2#0000	16#0E00
		Type N [NiCrSi-NiSi]	2#0001	16#0E01
		Type E [NiCr-CuNi]	2#0010	16#0E02
		Type R [PtRh-Pt]	2#0011	16#0E03
		Type S [PtRh-Pt]	2#0100	16#0E04
		Type J [Fe-CuNi IEC]	2#0101	16#0E05
		Type L [Fe-CuNi DIN]	2#0110	16#0E06
		Type T [Cu-CuNi IEC]	2#0111	16#0E07
		Type K [NiCr-Ni]	2#1000	16#0E08
		Type U [Cu-CuNi DIN]	2#1001	16#0E09
		Type C	2#1010	16#0E0A
		Type TXK/XK(L)	2#1011	16#0E0B

Effect of the temperature coefficients on the measuring range

- Setting TC = 3850 at GOST Standard Pt10, Pt50, Pt100, Pt500 sets Bit 7 in the measuring range byte (0x40)
- Setting TC = 428 at GOST Standard Cu10, Cu50, Cu100 sets Bit 8 in the measuring range byte (0x80)

Measuring range coding of the analog output modules

Depending on the measuring range coding of the analog output modules, the parameter Mode (measuring-range coding) corresponding to the channel must be specified in accordance with the table.

Measuring type	Measuring range	Mode
Voltage	± 5 V	16#0106
	1 to 5 V	16#0107
	0 to 10 V	16#0108
	± 10 V	16#0109
Current	0 to 20 mA	16#0202
	4 to 20 mA	16#0203
	± 20 mA	16#0204
HART interface	4 to 20 mA	16#070C

Measuring-Range Coding of the Digital Input and Output Modules

There is no measuring type and no measuring range for digital input modules and digital output modules:

- Mode **= 16#FFFF** (for DiIn)
- Mode **= 16#FFFE** (for DiOu)

6.11.2 Mode settings for PA field devices

Mode settings for PA field devices

Block	Parameter at block ...	...and at field device	Input/output from block view (PLS view)	Mode 16#xxyy, O=xx I=yy
Analog input (FbAnIn)	PV	OUT	I	16#0001
Analog output (FbAnOu)	SP	SP	O	16#0100
Analog output (FbAnOu)	SP Rbk PosD	SP READBACK POS_D	O I I	16#0103
Analog output (FbAnOu)	SP CbkBy0 - CbkBy2	SP CHECK_BACK	O I	16#0104
Analog output (FbAnOu)	SP Rbk PosD CbkBy0 - CbkBy2	SP READBACK POS_D CHECK_BACK	O I I I	16#0105
Analog output (FbAnOu)	RCasIn, RCasOut	RCAS_IN, RCAS_OUT	O I	16#0206
Analog output (FbAnOu)	RCasIn, RCasOut, CbkBy0 - CbkBy2	RCAS_IN, RCAS_OUT, CHECK_BACK	O I I	16#0207
Analog output (FbAnOu)	SP RCasIn Rbk RCasOut PosD CbkBy0 - CbkBy2	SP RCAS_IN READBACK RCAS_OUT POS_D CHECK_BACK	O O I I I I	16#0308
Binary input (FbDiIn)	PV	OUT_D	I	16#0002
Binary output (FbDiOu)	SP	SP_D	O	16#0400
Binary output (FbDiOu)	SP Rbk	SP_D READBACK_D	O I	16#0409
Binary output (FbDiOu)	SP CbkBy0 - CbkBy2	SP_D CHECKBACK_D	O I	16#040A
Binary output (FbDiOu)	SP Rbk CbkBy0 - CbkBy2	SP_D READBACK_D CHECK_BACK_D	O I I	16#040B
Binary output (FbDiOu)	RCasIn RCasOut	RCAS_IN_D RCAS_OUT_D	O I	16#050C
Binary output (FbDiOu)	RCasIn RCasOut CbkBy0 - CbkBy2	RCAS_IN_D RCAS_OUT_D CHECK_BACK_D	O I I	16#050D
Binary output (FbDiOu)	SP RCasIn Rbk RCasOut CbkBy0 - CbkBy2	SP_D RCAS_IN_D READBACK_D RCAS_OUT_D CHECK_BACK_D	O O I I I	16#060E

6.11.3 Mode settings for FF field devices

`Mode_LW` settings for FF field devices

Block	Parameter at block ...	...and at field device	Input/output from block view (PLS view)	`Mode_LW` 16#xxyy, O=xx I=yy
Analog input (FbAnIn)	PV	OUT	I	16#0001
Analog output (FbAnOu)	SP	SP	O	16#0100
Analog output (FbAnOu)	SP Rbk PosD_W	SP READBACK POS_D	O I I	16#0103
Analog output (FbAnOu)	SP_Li CbkBy0_W - CbkBy2_W	SP CHECK_BACK	O I	16#0104
Analog output (FbAnOu)	SP Rbk PosD_W CbkBy0_W - CbkBy2_W	SP READBACK POS_D CHECK_BACK	O I I I	16#0105
Analog output (FbAnOu)	RCasIn, RCasOut	RCAS_IN RCAS_OUT	O I	16#0206
Analog output (FbAnOu)	RCasIn, RCasOut, CbkBy0_W - CbkBy2_W	RCAS_IN RCAS_OUT CHECK_BACK	O I I	16#0207
Analog output (FbAnOu)	SP RCasIn Rbk RCasOut PosD_W CbkBy0_W - CbkBy2_W	SP RCAS_IN READBACK RCAS_OUT POS_D CHECK_BACK	O O I I I I	16#0308
Binary input (FbDiIn)	PV	OUT_D	I	16#0002
Binary output (FbDiOu)	SP_W	SP_D	I	16#0400
Binary output (FbDiOu)	SP_W Rbk	SP_D READBACK_D	O I	16#0409
Binary output (FbDiOu)	SP_W CbkBy0_W - CbkBy2_W	SP_D CHECK_BACK_D	O I	16#040A
Binary output (FbDiOu)	SP_W Rbk_W CbkBy0_W - CbkBy2_W	SP_D READBACK_D CHECK_BACK_D	O I I	16#040B
Binary output (FbDiOu)	RCasInW RCasOutW	RCAS_IN_D RCAS_OUT_D	O I	16#050C
Binary output (FbDiOu)	RCasInW RCasOutW CbkBy0_W - CbkBy2_W	RCAS_IN_D RCAS_OUT_D CHECK_BACK_D	O I I	16#050D
Binary output (FbDiOu)	SP_W RCasInW RbkW RCasOutW CbkBy0_W - CbkBy2_W	SP_D RCAS_IN_D READBACK_D RCAS_OUT_D CHECK_BACK_D	O O I I I	16#060E

Conversion blocks

7

7.1 StruAnIn - separating an analog structured variable

7.1.1 Description of StruAnIn

Object name (type + number) and family

Type + number: FC 324

Family: Convert

Area of application for StruAnIn

The block is used for the following applications:

- Separation of an analog value with a structure into a variable of the REAL data type and a signal status

How it works

The block separates an analog value with a structure interconnected to the `In` input parameter into a variable (`Out`) of the REAL data type and a (`ST`) signal status.

Configuration

Use the CFC editor to install the block in any OB.

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for parameter `Status`

This block does not provide the `Status` parameter.

See also

StruAnIn block diagram (Page 919)
StruAnIn I/Os (Page 918)
StruAnIn messaging (Page 917)
StruAnIn error handling (Page 917)
StruAnIn functions (Page 916)
StruAnIn modes (Page 916)

7.1.2 StruAnIn modes

StruAnIn modes

This block does not provide any modes.

See also

StruAnIn block diagram (Page 919)
StruAnIn I/Os (Page 918)
StruAnIn messaging (Page 917)
StruAnIn error handling (Page 917)
StruAnIn functions (Page 916)
Description of StruAnIn (Page 915)
Out of service (Page 27)

7.1.3 StruAnIn functions

Functions of StruAnIn

There are no other functions for this block.

See also

StruAnIn block diagram (Page 919)
StruAnIn I/Os (Page 918)
StruAnIn messaging (Page 917)
StruAnIn error handling (Page 917)
StruAnIn modes (Page 916)
Description of StruAnIn (Page 915)

7.1.4 StruAnIn error handling

StruAnIn error handling

The block does not report any errors.

See also

StruAnIn block diagram (Page 919)
StruAnIn I/Os (Page 918)
StruAnIn messaging (Page 917)
StruAnIn functions (Page 916)
StruAnIn modes (Page 916)
Description of StruAnIn (Page 915)

7.1.5 StruAnIn messaging

Messaging

The block does not have any message functionality.

See also

StruAnIn block diagram (Page 919)
StruAnIn I/Os (Page 918)
StruAnIn error handling (Page 917)
StruAnIn functions (Page 916)
StruAnIn modes (Page 916)
Description of StruAnIn (Page 915)

7.1.6 StruAnIn I/Os

StruAnIn I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
In	Analog value with structure	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ST	Signal status	BYTE	16#80
Value	Analog value	REAL	0.0

See also

StruAnIn block diagram (Page 919)
StruAnIn messaging (Page 917)
StruAnIn error handling (Page 917)
StruAnIn functions (Page 916)
StruAnIn modes (Page 916)
Description of StruAnIn (Page 915)

7.1.7 StruAnIn block diagram

StruAnIn block diagram

A block diagram is not provided for this block.

See also

StruAnIn I/Os (Page 918)
StruAnIn messaging (Page 917)
StruAnIn error handling (Page 917)
StruAnIn functions (Page 916)
StruAnIn modes (Page 916)
Description of StruAnIn (Page 915)

7.2 StruAnOu - creating an analog structured variable

7.2.1 Description of StruAnOu

Object name (type + number) and family

Type + number: FC 325

Family: Convert

Area of application for StruAnOu

The block is used for the following applications:

- Merging a variable of the REAL data type and a signal status into an analog process value.

How it works

The block merges an analog value (`Value`) of the REAL data type and a signal status (`ST`) to form an analog value (`Out`) with a structure.

Configuration

Use the CFC editor to install the block in any OB.

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status parameter

This block does not provide the `Status` parameter.

See also

StruAnOu modes (Page 921)

StruAnOu functions (Page 921)

StruAnOu error handling (Page 922)

StruAnOu messaging (Page 922)

StruAnOu I/Os (Page 923)

StruAnOu block diagram (Page 924)

7.2.2 StruAnOu modes

StruAnOu modes

This block does not provide any modes.

See also

Description of StruAnOu (Page 920)
StruAnOu functions (Page 921)
StruAnOu error handling (Page 922)
StruAnOu messaging (Page 922)
StruAnOu I/Os (Page 923)
StruAnOu block diagram (Page 924)
Out of service (Page 27)

7.2.3 StruAnOu functions

Functions of StruAnOu

There are no other functions for this block.

See also

Description of StruAnOu (Page 920)
StruAnOu modes (Page 921)
StruAnOu error handling (Page 922)
StruAnOu messaging (Page 922)
StruAnOu I/Os (Page 923)
StruAnOu block diagram (Page 924)

7.2.4 StruAnOu error handling

StruAnOu error handling

The block does not report any errors.

See also

Description of StruAnOu (Page 920)
StruAnOu modes (Page 921)
StruAnOu functions (Page 921)
StruAnOu messaging (Page 922)
StruAnOu I/Os (Page 923)
StruAnOu block diagram (Page 924)

7.2.5 StruAnOu messaging

Messaging

The block does not have any message functionality.

See also

Description of StruAnOu (Page 920)
StruAnOu modes (Page 921)
StruAnOu functions (Page 921)
StruAnOu error handling (Page 922)
StruAnOu I/Os (Page 923)
StruAnOu block diagram (Page 924)

7.2.6 StruAnOu I/Os

StruAnOu I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
ST	Signal status	BYTE	16#80
Value	Analog value	REAL	0.0

Output parameters

Parameter	Description	Type	Default
Bad	1 = (ST <= 16#3F)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ENO	1 = Block algorithm completed without errors	BOOL	0
Out	Analog value with structure	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

See also

Description of StruAnOu (Page 920)
StruAnOu modes (Page 921)
StruAnOu functions (Page 921)
StruAnOu error handling (Page 922)
StruAnOu messaging (Page 922)
StruAnOu block diagram (Page 924)

7.2.7 StruAnOu block diagram

StruAnOu block diagram

A block diagram is not provided for this block.

See also

Description of StruAnOu (Page 920)
StruAnOu modes (Page 921)
StruAnOu functions (Page 921)
StruAnOu error handling (Page 922)
StruAnOu messaging (Page 922)
StruAnOu I/Os (Page 923)

7.3 StruDiIn - separating a digital structured variable

7.3.1 Description of StruDiIn

Object name (type + number) and family

Type + number: FC 326

Family: Convert

Area of application for StruDiIn

The block is used for the following applications:

- Separating a binary process value into a variable of the BOOL data type, a process value and signal status.

How it works

The block separates a binary process value interconnected to the `In` input parameter into a variable of the BOOL data type and a signal status.

Configuration

Use the CFC editor to install the block in any OB

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status parameter

This block does not provide the `Status` parameter.

See also

StruDiIn block diagram (Page 929)

StruDiIn I/Os (Page 928)

StruDiIn messaging (Page 927)

StruDiIn error handling (Page 927)

StruDiIn functions (Page 926)

StruDiIn modes (Page 926)

7.3.2 StruDiIn modes

StruDiIn modes

This block does not provide any modes.

See also

StruDiIn block diagram (Page 929)
StruDiIn I/Os (Page 928)
StruDiIn messaging (Page 927)
StruDiIn error handling (Page 927)
StruDiIn functions (Page 926)
Description of StruDiIn (Page 925)
Out of service (Page 27)

7.3.3 StruDiIn functions

Functions of StruDiIn

There are no other functions for this block.

See also

StruDiIn block diagram (Page 929)
StruDiIn I/Os (Page 928)
StruDiIn messaging (Page 927)
StruDiIn error handling (Page 927)
StruDiIn modes (Page 926)
Description of StruDiIn (Page 925)

7.3.4 StruDiIn error handling

StruDiIn error handling

The block does not report any errors.

See also

StruDiIn block diagram (Page 929)
StruDiIn I/Os (Page 928)
StruDiIn messaging (Page 927)
StruDiIn functions (Page 926)
StruDiIn modes (Page 926)
Description of StruDiIn (Page 925)

7.3.5 StruDiIn messaging

Message behavior

The block does not have any message behavior.

See also

StruDiIn block diagram (Page 929)
StruDiIn I/Os (Page 928)
StruDiIn error handling (Page 927)
StruDiIn functions (Page 926)
StruDiIn modes (Page 926)
Description of StruDiIn (Page 925)

7.3.6 StruDiIn I/Os

StruDiIn I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
In	Binary process value	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ST	Signal status	BYTE	16#80
Value	Binary variable	BOOL	0

See also

StruDiIn block diagram (Page 929)
StruDiIn messaging (Page 927)
StruDiIn error handling (Page 927)
StruDiIn functions (Page 926)
StruDiIn modes (Page 926)
Description of StruDiIn (Page 925)

7.3.7 StruDiIn block diagram

StruDiIn block diagram

A block diagram is not provided for this block.

See also

StruDiIn I/Os (Page 928)
StruDiIn messaging (Page 927)
StruDiIn error handling (Page 927)
StruDiIn functions (Page 926)
StruDiIn modes (Page 926)
Description of StruDiIn (Page 925)

7.4 StruDiOu - creating a digital structured variable

7.4.1 Description of StruDiOu

Object name (type + number) and family

Type + number: FC 327

Family: Convert

Area of application for StruDiOu

The block is used for the following applications:

- Merging a variable of the BOOL data type and a signal status into a binary process value.

How it works

The block merges a variable of the BOOL data type and a signal status into a binary process value.

Configuration

Use the CFC editor to install the block in any OB.

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status parameter

This block does not provide the `Status` parameter.

See also

StruDiOu block diagram (Page 934)
StruDiOu I/Os (Page 933)
StruDiOu messaging (Page 932)
StruDiOu error handling (Page 932)
StruDiOu functions (Page 931)
StruDiOu modes (Page 931)

7.4.2 StruDiOu modes

StruDiOu modes

This block does not provide any modes.

See also

StruDiOu block diagram (Page 934)
StruDiOu I/Os (Page 933)
StruDiOu messaging (Page 932)
StruDiOu error handling (Page 932)
StruDiOu functions (Page 931)
Description of StruDiOu (Page 930)
Out of service (Page 27)

7.4.3 StruDiOu functions

Functions of StruDiOu

There are no other functions for this block.

See also

StruDiOu block diagram (Page 934)
StruDiOu I/Os (Page 933)
StruDiOu messaging (Page 932)
StruDiOu error handling (Page 932)
StruDiOu modes (Page 931)
Description of StruDiOu (Page 930)

7.4.4 StruDiOu error handling

StruDiOu error handling

The block does not report any errors.

See also

StruDiOu block diagram (Page 934)
StruDiOu I/Os (Page 933)
StruDiOu messaging (Page 932)
StruDiOu functions (Page 931)
StruDiOu modes (Page 931)
Description of StruDiOu (Page 930)

7.4.5 StruDiOu messaging

Message behavior

The block does not have any message behavior.

See also

StruDiOu block diagram (Page 934)
StruDiOu I/Os (Page 933)
StruDiOu error handling (Page 932)
StruDiOu functions (Page 931)
StruDiOu modes (Page 931)
Description of StruDiOu (Page 930)

7.4.6 StruDiOu I/Os

StruDiOu I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
ST	Signal status	BYTE	16#80
Value	Binary variable	BOOL	0

Output parameters

Parameter	Description	Type	Default
Bad	1 = (ST <= 16#3F)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ENO	1 = Block algorithm completed without errors	BOOL	0
Out	Binary process value	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

StruDiOu block diagram (Page 934)
StruDiOu messaging (Page 932)
StruDiOu error handling (Page 932)
StruDiOu functions (Page 931)
StruDiOu modes (Page 931)
Description of StruDiOu (Page 930)

7.4.7 StruDiOu block diagram

StruDiOu block diagram

A block diagram is not provided for this block.

See also

StruDiOu I/Os (Page 933)
StruDiOu messaging (Page 932)
StruDiOu error handling (Page 932)
StruDiOu functions (Page 931)
StruDiOu modes (Page 931)
Description of StruDiOu (Page 930)

7.5 StruScIn - separating a display area into two variables

7.5.1 Description of StruScIn

Object name (type + number) and family

Type + number: FC 332

Family: Convert

Area of application for StruScIn

The block is used for the following applications:

- Separation of a display area into two variables of data type REAL.

How it works

The block separates a display area interconnected to the `Scale` input parameter into two variables of the REAL. data type

Configuration

Use the CFC editor to install the block in any OB.

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status parameter

This block does not provide the `Status` parameter.

See also

StruScIn block diagram (Page 939)

StruScIn I/Os (Page 938)

StruScIn messaging (Page 937)

StruScIn error handling (Page 937)

StruScIn functions (Page 936)

StruScIn modes (Page 936)

7.5.2 StruScIn modes

StruScIn modes

This block does not provide any modes.

See also

StruScIn block diagram (Page 939)
StruScIn I/Os (Page 938)
StruScIn messaging (Page 937)
StruScIn error handling (Page 937)
StruScIn functions (Page 936)
Description of StruScIn (Page 935)
Out of service (Page 27)

7.5.3 StruScIn functions

Functions of StruScIn

There are no other functions for this block.

See also

StruScIn block diagram (Page 939)
StruScIn I/Os (Page 938)
StruScIn messaging (Page 937)
StruScIn error handling (Page 937)
StruScIn modes (Page 936)
Description of StruScIn (Page 935)

7.5.4 StruScIn error handling

StruScIn error handling

The block does not report any errors.

See also

StruScIn block diagram (Page 939)
StruScIn I/Os (Page 938)
StruScIn messaging (Page 937)
StruScIn functions (Page 936)
StruScIn modes (Page 936)
Description of StruScIn (Page 935)

7.5.5 StruScIn messaging

Messaging

The block does not have any message functionality.

See also

StruScIn block diagram (Page 939)
StruScIn I/Os (Page 938)
StruScIn error handling (Page 937)
StruScIn functions (Page 936)
StruScIn modes (Page 936)
Description of StruScIn (Page 935)

7.5.6 StruScIn I/Os

StruScIn I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
Scale	Display area	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
HiScale	High limit of display area	REAL	100.0
LoScale	Low limit of display area	REAL	0.0

See also

StruScIn block diagram (Page 939)
StruScIn messaging (Page 937)
StruScIn error handling (Page 937)
StruScIn functions (Page 936)
StruScIn modes (Page 936)
Description of StruScIn (Page 935)

7.5.7 StruScIn block diagram

StruScIn block diagram

A block diagram is not provided for this block.

See also

StruScIn I/Os (Page 938)
StruScIn messaging (Page 937)
StruScIn error handling (Page 937)
StruScIn functions (Page 936)
StruScIn modes (Page 936)
Description of StruScIn (Page 935)

7.6 StruScOu - merging two variables into a display area

7.6.1 Description of StruScOu

Object name (type + number) and family

Type + number: FC 333

Family: Convert

Area of application for StruScOu

The block is used for the following applications:

- Merging two variables of the REAL data type into a display area.

How it works

The block merges two variables of the REAL data type into a display area.

Configuration

Use the CFC editor to install the block in any OB.

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status parameter

This block does not provide the `Status` parameter.

See also

StruScOu block diagram (Page 944)
StruScOu I/Os (Page 943)
StruScOu messaging (Page 942)
StruScOu error handling (Page 942)
StruScOu functions (Page 941)
StruScOu modes (Page 941)

7.6.2 StruScOu modes

StruScOu modes

This block does not provide any modes.

See also

StruScOu block diagram (Page 944)
StruScOu I/Os (Page 943)
StruScOu messaging (Page 942)
StruScOu error handling (Page 942)
StruScOu functions (Page 941)
Description of StruScOu (Page 940)
Out of service (Page 27)

7.6.3 StruScOu functions

Functions of StruScOu

There are no other functions for this block.

See also

StruScOu block diagram (Page 944)
StruScOu I/Os (Page 943)
StruScOu messaging (Page 942)
StruScOu error handling (Page 942)
StruScOu modes (Page 941)
Description of StruScOu (Page 940)

7.6.4 StruScOu error handling

StruScOu error handling

The block does not report any errors.

See also

StruScOu block diagram (Page 944)
StruScOu I/Os (Page 943)
StruScOu messaging (Page 942)
StruScOu functions (Page 941)
StruScOu modes (Page 941)
Description of StruScOu (Page 940)

7.6.5 StruScOu messaging

Message behavior

The block does not have any message behavior.

See also

StruScOu block diagram (Page 944)
StruScOu I/Os (Page 943)
StruScOu error handling (Page 942)
StruScOu functions (Page 941)
StruScOu modes (Page 941)
Description of StruScOu (Page 940)

7.6.6 StruScOu I/Os

StruScOu I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
HiScale	High limit of display area	REAL	100.0
LoScale	Low limit of display area	REAL	0.0

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
Scale	Display area	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0

See also

StruScOu block diagram (Page 944)
StruScOu messaging (Page 942)
StruScOu error handling (Page 942)
StruScOu functions (Page 941)
StruScOu modes (Page 941)
Description of StruScOu (Page 940)

7.6.7 StruScOu block diagram

StruScOu block diagram

A block diagram is not provided for this block.

See also

StruScOu I/Os (Page 943)
StruScOu messaging (Page 942)
StruScOu error handling (Page 942)
StruScOu functions (Page 941)
StruScOu modes (Page 941)
Description of StruScOu (Page 940)

7.7 STIn - separating the signal status into individual binary displays

7.7.1 Description of STIn

Object name (type + number) and family

Type + number: FC 322

Family: Convert

Area of application for STIn

The block is used for the following applications:

- Separation of signal status into individual binary displays

How it works

The block separates a signal status interconnected to the input parameter into individual binary displays.

Configuration

Use the CFC editor to install the block in any OB.

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status parameter

This block does not provide the `Status` parameter.

See also

STIn block diagram (Page 949)

STIn I/Os (Page 948)

STIn messaging (Page 947)

STIn error handling (Page 947)

STIn functions (Page 946)

STIn modes (Page 946)

7.7.2 STIn modes

STIn modes

This block does not provide any modes.

See also

Out of service (Page 27)
STIn block diagram (Page 949)
STIn I/Os (Page 948)
STIn messaging (Page 947)
STIn error handling (Page 947)
STIn functions (Page 946)
Description of STIn (Page 945)

7.7.3 STIn functions

Functions of STIn

There are no other functions for this block.

See also

STIn block diagram (Page 949)
STIn I/Os (Page 948)
STIn messaging (Page 947)
STIn error handling (Page 947)
STIn modes (Page 946)
Description of STIn (Page 945)

7.7.4 STIn error handling

STIn error handling

The block does not report any errors.

See also

STIn block diagram (Page 949)
STIn I/Os (Page 948)
STIn messaging (Page 947)
STIn functions (Page 946)
STIn modes (Page 946)
Description of STIn (Page 945)

7.7.5 STIn messaging

Message behavior

The block does not have any message behavior.

See also

STIn block diagram (Page 949)
STIn I/Os (Page 948)
STIn error handling (Page 947)
STIn functions (Page 946)
STIn modes (Page 946)
Description of STIn (Page 945)

7.7.6 STIn I/Os

STIn I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
In	Signal status	BYTE	16#80

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ST_00	1 = Bad, device related	BOOL	0
ST_28	1 = Bad, process related	BOOL	0
ST_60	1 = Local functional check/simulation	BOOL	0
ST_68	1 = Unknown, device related	BOOL	0
ST_78	1 = Unknown, process related	BOOL	0
ST_80	1 = Good	BOOL	0
ST_A4	1 = Maintenance request	BOOL	0

See also

STIn error handling (Page 947)
STIn block diagram (Page 949)
Description of STIn (Page 945)
STIn modes (Page 946)
STIn functions (Page 946)
STIn messaging (Page 947)

7.7.7 STIn block diagram

STIn block diagram

A block diagram is not provided for this block.

See also

Description of STIn (Page 945)
STIn modes (Page 946)
STIn functions (Page 946)
STIn error handling (Page 947)
STIn messaging (Page 947)
STIn I/Os (Page 948)

7.8 STOu - merging individual binary signals into a signal status

7.8.1 Description of STOu

Object name (type + number) and family

Type + number: FC 323

Family: Convert

Area of application for STOu

The block is used for the following applications:

- Merging individual binary signals into a signal status

How it works

The block merges individual binary signals into a `Out` signal status.

If several binary signals are set, the one with the highest priority becomes effective, as described in the section Forming and outputting signal status for blocks (Page 88) for technological blocks.

If no binary signal is set, the "Bad, process related" signal status is set.

Configuration

Use the CFC editor to install the block in any OB.

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status parameter

This block does not provide the Status parameter.

See also

STOu block diagram (Page 954)

STOu I/Os (Page 953)

STOu messaging (Page 952)

STOu error handling (Page 952)

STOu functions (Page 951)

STOu modes (Page 951)

7.8.2 STOu modes

STOu modes

This block does not provide any modes.

See also

Out of service (Page 27)
STOu block diagram (Page 954)
STOu I/Os (Page 953)
STOu messaging (Page 952)
STOu error handling (Page 952)
STOu functions (Page 951)
Description of STOu (Page 950)

7.8.3 STOu functions

Functions of STOu

There are no other functions for this block.

See also

STOu block diagram (Page 954)
STOu I/Os (Page 953)
STOu messaging (Page 952)
STOu error handling (Page 952)
STOu modes (Page 951)
Description of STOu (Page 950)

7.8.4 STOu error handling

STOu error handling

The block does not report any errors.

See also

STOu block diagram (Page 954)
STOu I/Os (Page 953)
STOu messaging (Page 952)
STOu functions (Page 951)
STOu modes (Page 951)
Description of STOu (Page 950)

7.8.5 STOu messaging

Message behavior

The block does not have any message behavior.

See also

STOu block diagram (Page 954)
STOu I/Os (Page 953)
STOu error handling (Page 952)
STOu functions (Page 951)
STOu modes (Page 951)
Description of STOu (Page 950)

7.8.6 STOu I/Os

STOu I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
ST_00	1 = Bad, device related	BOOL	0
ST_28	1 = Bad, process related	BOOL	0
ST_60	1 = Local functional check/simulation	BOOL	0
ST_68	1 = Unknown, device related	BOOL	0
ST_78	1 = Unknown, process related	BOOL	0
ST_80	1 = Good	BOOL	0
ST_A4	1 = Maintenance request	BOOL	0

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
Out	Signal status	BYTE	16#80

See also

STOu error handling (Page 952)
STOu block diagram (Page 954)
STOu messaging (Page 952)
STOu functions (Page 951)
STOu modes (Page 951)
Description of STOu (Page 950)

7.8.7 STOu block diagram

STOu block diagram

A block diagram is not provided for this block.

See also

STOu I/Os (Page 953)
STOu messaging (Page 952)
STOu error handling (Page 952)
STOu functions (Page 951)
STOu modes (Page 951)
Description of STOu (Page 950)

7.9 MSTIn - separating the maintenance status into individual status displays

7.9.1 Description of MSTIn

Object name (type + number) and family

Type + number: FB 1858

Family: Convert

Area of application for MSTIn

The block is used for the following applications:

- Separation of the maintenance status into individual status displays

How it works

The block separates a maintenance status interconnected to the `In` input parameter into individual status displays.

If the input parameter receives information that at least one value is simulated, for example (`In = 16#00000003`), this is indicated at output parameter `MST_03` with the value 1.

Configuration

Use the CFC editor to install the block in any OB.

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status parameter

This block does not provide the `Status` parameter.

See also

MSTIn block diagram (Page 959)

MSTIn I/Os (Page 958)

MSTIn messaging (Page 957)

MSTIn error handling (Page 957)

MSTIn functions (Page 956)

MSTIn modes (Page 956)

7.9.2 MSTIn modes

MSTIn modes

This block does not provide any modes.

See also

MSTIn I/Os (Page 958)
MSTIn messaging (Page 957)
MSTIn error handling (Page 957)
MSTIn functions (Page 956)
MSTIn block diagram (Page 959)
Description of MSTIn (Page 955)
Out of service (Page 27)

7.9.3 MSTIn functions

Functions of MSTIn

There are no other functions for this block.

See also

MSTIn block diagram (Page 959)
MSTIn I/Os (Page 958)
MSTIn messaging (Page 957)
MSTIn error handling (Page 957)
MSTIn modes (Page 956)
Description of MSTIn (Page 955)

7.9.4 MSTIn error handling

MSTIn error handling

The block does not report any errors.

See also

MSTIn block diagram (Page 959)
MSTIn I/Os (Page 958)
MSTIn messaging (Page 957)
MSTIn functions (Page 956)
MSTIn modes (Page 956)
Description of MSTIn (Page 955)

7.9.5 MSTIn messaging

Message behavior

The block does not have any message behavior.

See also

MSTIn block diagram (Page 959)
MSTIn I/Os (Page 958)
MSTIn error handling (Page 957)
MSTIn functions (Page 956)
MSTIn modes (Page 956)
Description of MSTIn (Page 955)

7.9.6 MSTIn I/Os

MSTIn I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
In	Maintenance status	DWORD	16#00

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
MST_00	1 = Good	BOOL	0
MST_01	1 = Passivated	BOOL	0
MST_02	1 = out of service	BOOL	0
MST_03	1 = At least one process value is simulated	BOOL	0
MST_04	1 = Local mode	BOOL	0
MST_05	1 = Maintenance requirement	BOOL	0
MST_06	1 = Maintenance request	BOOL	0
MST_07	1 = Maintenance alarm	BOOL	0
MST_08	1 = Unknown	BOOL	0
MST_09	1 = Configuration changed	BOOL	0

See also

MSTIn block diagram (Page 959)
MSTIn messaging (Page 957)
MSTIn error handling (Page 957)
MSTIn functions (Page 956)
MSTIn modes (Page 956)
Description of MSTIn (Page 955)

7.9.7 MSTIn block diagram

MSTIn block diagram

A block diagram is not provided for this block.

See also

MSTIn I/Os (Page 958)
MSTIn messaging (Page 957)
MSTIn error handling (Page 957)
MSTIn functions (Page 956)
MSTIn modes (Page 956)
Description of MSTIn (Page 955)

7.10 MSTOu - merging individual status displays into a maintenance status

7.10.1 Description of MSTOu

Object name (type + number) and family

Type + number: FB 1859

Family: Convert

Area of application for MSTOu

The block is used for the following applications:

- Merging individual status displays into a maintenance status

How it works

The block merges individual status displays into a maintenance status. If several status displays are set, the status display with the highest number becomes effective.

If status display `MST_03 = 1` is set, for example, this is indicated at output parameter `Out` with the value `16#00000003`.

If no status display is set, the `Out = 16#00` maintenance status is set.

Configuration

Use the CFC editor to install the block in any OB.

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for parameter Status

This block does not provide the `Status` parameter.

See also

MSTOu block diagram (Page 964)

MSTOu I/Os (Page 963)

MSTOu messaging (Page 962)

MSTOu error handling (Page 962)

MSTOu functions (Page 961)

MSTOu modes (Page 961)

7.10.2 MSTOu modes

MSTOu modes

This block does not provide any modes.

See also

MSTOu block diagram (Page 964)
MSTOu I/Os (Page 963)
MSTOu messaging (Page 962)
MSTOu error handling (Page 962)
MSTOu functions (Page 961)
Description of MSTOu (Page 960)
Out of service (Page 27)

7.10.3 MSTOu functions

Functions of MSTOu

There are no other functions for this block.

See also

MSTOu block diagram (Page 964)
MSTOu I/Os (Page 963)
MSTOu messaging (Page 962)
MSTOu error handling (Page 962)
MSTOu modes (Page 961)
Description of MSTOu (Page 960)

7.10.4 MSTOu error handling

MSTOu error handling

The block does not report any errors.

See also

MSTOu block diagram (Page 964)
MSTOu I/Os (Page 963)
MSTOu messaging (Page 962)
MSTOu functions (Page 961)
MSTOu modes (Page 961)
Description of MSTOu (Page 960)

7.10.5 MSTOu messaging

Message behavior

The block does not have any message behavior.

See also

MSTOu block diagram (Page 964)
MSTOu I/Os (Page 963)
MSTOu error handling (Page 962)
MSTOu functions (Page 961)
MSTOu modes (Page 961)
Description of MSTOu (Page 960)

7.10.6 MSTOu I/Os

MSTOu I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
MST_00	1 = Good	BOOL	0
MST_01	1 = Passivated	BOOL	0
MST_02	1 = out of service	BOOL	0
MST_03	1 = At least one process value is simulated	BOOL	0
MST_04	1 = Local mode	BOOL	0
MST_05	1 = Maintenance requirement	BOOL	0
MST_06	1 = Maintenance request	BOOL	0
MST_07	1 = Maintenance alarm	BOOL	0
MST_08	1 = Unknown	BOOL	0
MST_09	1 = Configuration changed	BOOL	0

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
Out	Maintenance status	DWORD	16#00

See also

MSTOu block diagram (Page 964)
MSTOu messaging (Page 962)
MSTOu error handling (Page 962)
MSTOu functions (Page 961)
MSTOu modes (Page 961)
Description of MSTOu (Page 960)

7.10.7 MSTOu block diagram

MSTOu block diagram

A block diagram is not provided for this block.

See also

MSTOu I/Os (Page 963)
MSTOu messaging (Page 962)
MSTOu error handling (Page 962)
MSTOu functions (Page 961)
MSTOu modes (Page 961)
Description of MSTOu (Page 960)

Message blocks

8

8.1 Event - Creating messages

8.1.1 Description of Event

Object name (type + number) and family

Type + number: FB 1811

Family: Report

Area of application for Event

The block is used for the following applications:

- Generation of messages requiring acknowledgment

How it works

The block is used to simultaneously output up to eight different messages that require acknowledgment.

The individual messages are assigned to the monitored signals via the inputs and the messages released or blocked depending on the process status data. If a change is made to one or more of the monitored signals enabled for reporting, a message is output.

The signals to be monitored are interconnected with inputs `In1` to `In8`. Each In_x signal can also be inverted via input $InvIn_x$. A message is output if a signal value is changed (but taking inversion into account).

Each input is assigned a separate message text. A message received at input `In5`, for example, is output with the message text for the `SIG5` signal.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

Further addressing is not required.

Startup characteristics

During startup, the message block suppresses all messages. The duration (number of cycles) of message suppression is set at the RunUpCyc parameter. An internal counter that is initialized with this parameter value during restart (OB100) decrements each time the block is executed. Messages are not generated if the counter value does not equal zero.

Messages whose alarm delay has not elapsed during this period are then output.

Status word allocation for `Status1` parameter

Status bit	Parameter
0	In1.Value
1	In2.Value
2	In3.Value
3	In4.Value
4	In5.Value
5	In6.Value
6	In7.Value
7	In8.Value
8	InvIn1
9	InvIn2
10	InvIn3
11	InvIn4
12	InvIn5
13	InvIn6
14	InvIn7
15	InvIn8
16	In1 with inversion
17	In2 with inversion
18	In3 with inversion
19	In4 with inversion
20	In5 with inversion
21	In6 with inversion
22	In7 with inversion
23	In8 with inversion
24	In1 not interconnected
25	In2 not interconnected
26	In3 not interconnected
27	In4 not interconnected
28	In5 not interconnected
29	In6 not interconnected
30	In7 not interconnected
31	In8 not interconnected

Status word allocation for `Status2` parameter

Status bit	Parameter
0	`In1MsgEn`
1	`In2MsgEn`
2	`In3MsgEn`
3	`In4MsgEn`
4	`In5MsgEn`
5	`In6MsgEn`
6	`In7MsgEn`
7	`In8MsgEn`
8	`AV1` not interconnected
9	`AV2` not interconnected
10	`AV3` not interconnected
11	`AV4` not interconnected
12	`AV5` not interconnected
13	`AV6` not interconnected
14	`AV7` not interconnected
15	`AV8` not interconnected
16	Active signal 1 for messages
17	Active signal 2 for messages
18	Active signal 3 for messages
19	Active signal 4 for messages
20	Active signal 5 for messages
21	Active signal 6 for messages
22	Active signal 7 for messages
23	Active signal 8 for messages
24	`MsgLock`
25-31	Not used

See also

Event functions (Page 969)
Event messaging (Page 972)
Event I/Os (Page 974)
Event block diagram (Page 978)
Event error handling (Page 971)
Event modes (Page 968)

8.1.2 Event modes

Event operating modes

The block can be operated using the following modes:

- On (Page 27)
- Out of service (Page 27)

"On"

You can find general information about the "On" mode in the On (Page 27) section.

"Out of service"

You can find general information about the "Out of service" mode in the Out of service (Page 27) section.

See also

Event block diagram (Page 978)

Event I/Os (Page 974)

Event messaging (Page 972)

Event error handling (Page 971)

Event functions (Page 969)

Description of Event (Page 965)

8.1.3 Event functions

Functions of Event

The functions for this block are listed below.

Activation and deactivation of messages

Set the I/Os `In1MsgEn` to `In8MsgEn` accordingly to activate or deactivate the messages applied to inputs `In1` to `In8`. All messages are activated by default.

To deactivate messages received at I/O `In4`, for example, you set I/O `In4MsgEn = 0` accordingly.

You can deactivate all messages via I/O `MsgLock = 1`.

Delay of alarms

You can delay alarms for signal changes (please also refer to the section Alarm delays (Page 79)).

For incoming alarms (signal change 0 - 1), set the delay at parameter `AlmOnDly`; for outgoing alarms (signal change 1 - 0), set it at parameter `AlmOffDly`.

Enter 0 or a negative value to deactivate the delay.

Maintenance release

This block provides the standard function Maintenance release (Page 47).

Operator control permissions via parameter `OS_Perm`

The block has the following Operator permissions (Page 45) for the `OS_Perm` parameter:

Bit	Function
0	Not allocated
1	1 = Operator can switch to "On" mode
2	Not allocated
3	1 = Operator can switch to "Out of service" mode
4 - 31	Not allocated

This block does not have a faceplate yet; the operator control permissions have already been assigned in the planning phase for these faceplates.

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` I/O in the section Functions of blocks > Functions that can be set via the Feature I/O (Page 186).

The following functionality is available for this block at the relevant bits:

Bit	Function
1	Reaction to the out of service mode (Page 199)
22	Update acknowledgment and error status of the message call (Page 193)
27	Selecting values associated with messages (Page 192)

Forming the signal status for blocks

The block provides the standard function Forming and outputting signal status for blocks (Page 88).

The block determines the worst signal status via all interconnected binary and analog inputs, and outputs this value at `ST_Worst`.

- `In1`

etc. to

- `In8`

See also

Description of Event (Page 965)

Event messaging (Page 972)

Event I/Os (Page 974)

Event block diagram (Page 978)

Event error handling (Page 971)

Event modes (Page 968)

8.1.4 Event error handling

Event troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Predefined value when inserting the block, the block is not processed.
0	There is no error.

See also

Event block diagram (Page 978)

Event I/Os (Page 974)

Event messaging (Page 972)

Event functions (Page 969)

Event modes (Page 968)

Description of Event (Page 965)

8.1.5 Event messaging

Messaging

Messages which can be acknowledged are generated using ALARM_8P. The block uses the PMC communication channel and has 8 digital inputs and 8 associated values.

Every edge transition detected for one or more digital inputs results in a message. The associated values are assigned consistently to the message at the time of edge evaluation. All eight signals are assigned a common message number, which is split at the OS into eight messages. The ES assigns the message number automatically by calling the message server.

Process messages

Message instance	Message identifier	Message class	Event
`MsgEvId`	SIG 1	AS process control message - error	Text 1
	SIG 2	AS process control message - error	Text 2
	SIG 3	AS process control message - error	Text 3
	SIG 4	AS process control message - error	Text 4
	SIG 5	AS process control message - error	Text 5
	SIG 6	AS process control message - error	Text 6
	SIG 7	AS process control message - error	Text 7
	SIG 8	AS process control message - error	Text 8

Associated values for message instance `MsgEvId`

You can use the `Feature` bit Selecting values associated with messages (Page 192) to define whether the signal status of the signal or the associated analog value is to be used as the associated value for the message.

Associated value	Block parameters
1	`In1.ST`
2	`In2.ST`
3	`In3.ST`
4	`In4.ST`
5	`In5.ST`
6	`In6.ST`
7	`In7.ST`
8	`In8.ST`
9	Not allocated
10	Not allocated

Associated values for message instance `MsgEvId`

Associated value	Block parameters
1	`AV1.Value`
2	`AV2.Value`
3	`AV3.Value`
4	`AV4.Value`
5	`AV5.Value`
6	`AV6.Value`
7	`AV7.Value`
8	`AV8.Value`
9	Not allocated
10	Not allocated

See also

Description of Event (Page 965)

Event functions (Page 969)

Event I/Os (Page 974)

Event block diagram (Page 978)

Event error handling (Page 971)

Event modes (Page 968)

8.1.6 Event I/Os

Event I/Os

Input parameters

Parameter	Description	Type	Default
AlmOnDly	Alarm delay time [s] for signal transition 0 -> 1	REAL	0.0
AlmOffDly	Alarm delay time [s] for signal transition 1 -> 0	REAL	0.0
AV1	Message associated value for In1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV1Unit	Unit for AV1	INT	0
AV2	Message associated value for In2	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV2Unit	Unit for AV2	INT	0
AV3	Message associated value for In3	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV3Unit	Unit for AV3	INT	0
AV4	Message associated value for In4	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV4Unit	Unit for AV4	INT	0
AV5	Message associated value for In5	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV5Unit	Unit for AV5	INT	0
AV6	Message associated value for In6	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV6Unit	Unit for AV6	INT	0
AV7	Message associated value for In7	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV7Unit	Unit for AV7	INT	0
AV8	Message associated value for In8	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF

Parameter	Description	Type	Default
AV8Unit	Unit for AV8	INT	0
EN	1 = Called block will be processed	BOOL	1
Features	I/O for additional functions (Page 969)	DWORD	16#00000000
In1	Input In1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In1MsgEn	1 = Activate message for input In1	BOOL	1
In2	Input In2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In2MsgEn	1 = Activate message for input In2	BOOL	1
In3	Input In3	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In3MsgEn	1 = Activate message for input In3	BOOL	1
In4	Input In4	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In4MsgEn	1 = Activate message for input In4	BOOL	1
In5	Input In5	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In5MsgEn	1 = Activate message for input In5	BOOL	1
In6	Input In6	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In6MsgEn	1 = Activate message for input In6	BOOL	1
In7	Input In7	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In7MsgEn	1 = Activate message for input In7	BOOL	1
In8	Input In8	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In8MsgEn	1 = Activate message for input In8	BOOL	1
InvIn1	1 = Invert input In1	BOOL	0
InvIn2	1 = Invert input In2	BOOL	0
InvIn3	1 = Invert input In3	BOOL	0
InvIn4	1 = Invert input In4	BOOL	0
InvIn5	1 = Invert input In5	BOOL	0
InvIn6	1 = Invert input In6	BOOL	0

Parameter	Description	Type	Default
InvIn7	1 = Invert input In7	BOOL	0
InvIn8	1 = Invert input In8	BOOL	0
MS_RelOp	1 = Maintenance release by OS operator	BOOL	0
MsgEvId	Message number (assigned automatically)	DWORD	16#FFFFFFFF
MsgLock	1 = Suppress process messages	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OnOp	1 = "On" mode via operator	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC (0-1 edge transition)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OS_Perm	I/O for operator control permissions (Page 969)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SelFp1	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
UserStatus	Freely assignable user area for status word (bit 24 to bit 31)	BYTE	16#00

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see Event error handling (Page 971)	INT	-1
MsgAckn	Message acknowledgment status (output ACK_STATE of ALARM_8P)	WORD	16#0000
MsgErr	1 = Alarm error (output ERROR of ALARM_8)	BOOL	0
MsgStat	Alarm status (output STATUS of ALARM_8P)	WORD	16#0000
MS_Release	Maintenance release: 1 = Release for OS operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OnAct	1 = "On" mode enabled	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
OosAct	1 = Block is "Out of service"	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word 1 (Page 965)	DWORD	16#00000000
Status2	Status word 2 (Page 965)	DWORD	16#00000000

See also

Event messaging (Page 972)

Event block diagram (Page 978)

Event modes (Page 968)

8.1.7 Event block diagram

Event block diagram

A block diagram is not provided for this block.

See also

Event I/Os (Page 974)
Event messaging (Page 972)
Event error handling (Page 971)
Event functions (Page 969)
Event modes (Page 968)
Description of Event (Page 965)

8.2 EventNck - Generating messages without acknowledgment

8.2.1 Description of EventNck

Object name (type + number) and family

Type + number: FB 1904

Family: Report

Area of application for EventNck

The block is used for the following applications:

- Generation of messages not requiring acknowledgment

How it works

The block is used to simultaneously output up to eight different messages that do not require acknowledgment.

The individual messages are assigned to the monitored signals via the inputs and the messages released or blocked depending on the process status data. If a change is made to one or more of the monitored signals enabled for reporting, a message is output.

The signals to be monitored are interconnected with inputs `In1` to `In8`. Each $\texttt{In}_x$ signal can also be inverted via input $\texttt{InvIn}_x$. A message is output if a signal value is changed (but taking inversion into account).

Each input is assigned a separate message text. A message received at input `In5`, for example, is output with the message text for the `SIG5` signal.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

Further addressing is not required.

Startup characteristics

During startup, the message block suppresses all messages. The duration (number of cycles) of message suppression is set at the `RunUpCyc` parameter. An internal counter that is initialized with this parameter value during restart (OB100) decrements each time the block is executed. Messages are not generated if the counter value does not equal zero.

Messages whose alarm delay has not elapsed during this period are then output.

Status word allocation for `Status1` parameter

Status bit	Parameter
0	`In1.Value`
1	`In2.Value`
2	`In3.Value`
3	`In4.Value`
4	`In5.Value`
5	`In6.Value`
6	`In7.Value`
7	`In8.Value`
8	`InvIn1`
9	`InvIn2`
10	`InvIn3`
11	`InvIn4`
12	`InvIn5`
13	`InvIn6`
14	`InvIn7`
15	`InvIn8`
16	`In1` with inversion
17	`In2` with inversion
18	`In3` with inversion
19	`In4` with inversion
20	`In5` with inversion
21	`In6` with inversion
22	`In7` with inversion
23	`In8` with inversion
24	`In1` not interconnected
25	`In2` not interconnected
26	`In3` not interconnected
27	`In4` not interconnected
28	`In5` not interconnected
29	`In6` not interconnected
30	`In7` not interconnected
31	`In8` not interconnected

Status word allocation for `Status2` parameter

Status bit	Parameter
0	`In1MsgEn`
1	`In2MsgEn`
2	`In3MsgEn`
3	`In4MsgEn`
4	`In5MsgEn`
5	`In6MsgEn`
6	`In7MsgEn`
7	`In8MsgEn`
8	`AV1` not interconnected
9	`AV2` not interconnected
10	`AV3` not interconnected
11	`AV4` not interconnected
12	`AV5` not interconnected
13	`AV6` not interconnected
14	`AV7` not interconnected
15	`AV8` not interconnected
16	Active signal 1 for messages
17	Active signal 2 for messages
18	Active signal 3 for messages
19	Active signal 4 for messages
20	Active signal 5 for messages
21	Active signal 6 for messages
22	Active signal 7 for messages
23	Active signal 8 for messages
24	`MsgLock`
25-31	Not used

See also

EventNck modes (Page 982)
EventNck functions (Page 983)
EventNck error handling (Page 985)
EventNck messaging (Page 986)
EventNck I/Os (Page 988)
EventNck block diagram (Page 992)

8.2.2 EventNck modes

EventNck modes

The block provides the following modes:

- On (Page 27)
- Out of service (Page 27)

"On"

General information on the "On" mode is available in the section On (Page 27).

"Out of service"

You will find general information about the "Out of service" mode in the Out of service (Page 27) section.

See also

Description of EventNck (Page 979)

EventNck functions (Page 983)

EventNck error handling (Page 985)

EventNck messaging (Page 986)

EventNck I/Os (Page 988)

EventNck block diagram (Page 992)

8.2.3 EventNck functions

Functions of EventNck

The functions for this block are listed below.

Activation and deactivation of messages

Set the I/Os `In1MsgEn` to `In8MsgEn` accordingly to activate or deactivate the messages applied to inputs `In1` to `In8`. All messages are activated by default.

To deactivate messages received at I/O `In4`, for example, you set I/O `In4MsgEn = 0` accordingly.

You can deactivate all messages via I/O `MsgLock = 1`.

Delay of alarms

You can delay the alarms indicating signal changes. You can find additional information on this in the Alarm delays (Page 79) section.

For incoming alarms (signal change 0 - 1), set the delay at parameter `AlmOnDly`; for outgoing alarms (signal change 1 - 0), set it at parameter `AlmOffDly`.

Enter 0 or a negative value to deactivate the delay.

Maintenance release

This block provides the standard function Maintenance release (Page 47).

Operator control permissions via parameter `OS_Perm`

The block has the following Operator permissions (Page 45) for the `OS_Perm` parameter:

Bit	Function
0	Not allocated
1	1 = Operator can switch to "On" mode
2	Not allocated
3	1 = Operator can switch to out of service mode
4 - 31	Not allocated

This block does not have a faceplate yet; the operator control permissions have already been assigned in the planning phase for these faceplates.

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` I/O in the section Functions of blocks > Functions that can be set via the Feature I/O (Page 186).

The following functionality is available for this block at the relevant bits:

Bit	Function
1	Reaction to the out of service mode (Page 199)
22	Update acknowledgment and error status of the message call (Page 193)
27	Selecting values associated with messages (Page 192)

Forming the signal status for blocks

The block provides the standard function Forming and outputting signal status for blocks (Page 88).

The block determines the worst signal status via all interconnected binary and analog inputs, and outputs this value at `ST_Worst`.

- AV1

etc. to

- AV8

See also

Description of EventNck (Page 979)

EventNck modes (Page 982)

EventNck error handling (Page 985)

EventNck messaging (Page 986)

EventNck I/Os (Page 988)

EventNck block diagram (Page 992)

8.2.4 EventNck error handling

EventTs troubleshooting

Refer to the section Error handling (Page 117) in the basic instructions to learn how to troubleshoot all blocks.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Predefined value when inserting the block, the block is not processed.
0	There is no error.

See also

Description of EventNck (Page 979)

EventNck modes (Page 982)

EventNck functions (Page 983)

EventNck messaging (Page 986)

EventNck I/Os (Page 988)

EventNck block diagram (Page 992)

8.2.5 EventNck messaging

Messaging

Messages not requiring acknowledgment are generated with `NOTIFY_8P` . The block uses the PMC communication channel and has 8 digital inputs and 10 associated values.

Every edge transition detected for one or more digital inputs results in a message. The associated values are assigned consistently to the message at the time of edge evaluation. All eight signals are assigned a common message number, which is split at the OS into eight messages. The ES assigns the message number automatically by calling the message server.

Process messages

Message instance	Message identifier	Message class	Event
`MsgEvId`	SIG 1	Operating message – without acknowledgment	Text 1
	SIG 2	Operating message – without acknowledgment	Text 2
	SIG 3	Operating message – without acknowledgment	Text 3
	SIG 4	Operating message – without acknowledgment	Text 4
	SIG 5	Operating message – without acknowledgment	Text 5
	SIG 6	Operating message – without acknowledgment	Text 6
	SIG 7	Operating message – without acknowledgment	Text 7
	SIG 8	Operating message – without acknowledgment	Text 8

Associated values for message instance `MsgEvId`

You can use the `Feature` bit Selecting values associated with messages (Page 192) to define whether the signal status of the signal or the associated analog value is to be used as the associated value for the message.

Associated value	Block parameters
1	`In1.ST`
2	`In2.ST`
3	`In3.ST`
4	`In4.ST`
5	`In5.ST`
6	`In6.ST`
7	`In7.ST`
8	`In8.ST`
9	Not allocated
10	Not allocated

Associated values for message instance `MsgEvId`

Associated value	Block parameters
1	`AV1.Value`
2	`AV2.Value`
3	`AV3.Value`
4	`AV4.Value`
5	`AV5.Value`
6	`AV6.Value`
7	`AV7.Value`
8	`AV8.Value`
9	Not allocated
10	Not allocated

See also

Description of EventNck (Page 979)
EventNck modes (Page 982)
EventNck functions (Page 983)
EventNck error handling (Page 985)
EventNck I/Os (Page 988)
EventNck block diagram (Page 992)

8.2.6 EventNck I/Os

EventNck I/Os

Input parameters

Parameter	Description	Type	Default
AlmOnDly	Alarm delay time [s] for signal transition 0 -> 1	REAL	0.0
AlmOffDly	Alarm delay time [s] for signal transition 1 -> 0	REAL	0.0
AV1	Message associated value for In1	STRUCT	-
		Value: REAL	0.0
		ST: BYTE	16#FF
AV1Unit	Unit for AV1	INT	0
AV2	Message associated value for In2	STRUCT	-
		Value: REAL	0.0
		ST: BYTE	16#FF
AV2Unit	Unit for AV2	INT	0
AV3	Message associated value for In3	STRUCT	-
		Value: REAL	0.0
		ST: BYTE	16#FF
AV3Unit	Unit for AV3	INT	0
AV4	Message associated value for In4	STRUCT	-
		Value: REAL	0.0
		ST: BYTE	16#FF
AV4Unit	Unit for AV4	INT	0
AV5	Message associated value for In5	STRUCT	-
		Value: REAL	0.0
		ST: BYTE	16#FF
AV5Unit	Unit for AV5	INT	0
AV6	Message associated value for In6	STRUCT	-
		Value: REAL	0.0
		ST: BYTE	16#FF
AV6Unit	Unit for AV6	INT	0
AV7	Message associated value for In7	STRUCT	-
		Value: REAL	0.0
		ST: BYTE	16#FF
AV7Unit	Unit for AV7	INT	0
AV8	Message associated value for In8	STRUCT	-
		Value: REAL	0.0
		ST: BYTE	16#FF

Parameter	Description	Type	Default
AV8Unit	Unit for AV8	INT	0
EN	1 = Called block will be processed	BOOL	1
Features	I/O for additional functions (Page 983)	DWORD	16#00000000
In1	Input In1	STRUCT	-
		Value: BOOL	0
		ST: BYTE	16#FF
In1MsgEn	1 = Activate message for In1 input	BOOL	1
In2	Input In2	STRUCT	-
		Value: BOOL	0
		ST: BYTE	16#FF
In2MsgEn	1 = Activate message for In2 input	BOOL	1
In3	Input In3	STRUCT	-
		Value: BOOL	0
		ST: BYTE	16#FF
In3MsgEn	1 = Activate message for In3 input	BOOL	1
In4	Input In4	STRUCT	-
		Value: BOOL	0
		ST: BYTE	16#FF
In4MsgEn	1 = Activate message for In4 input	BOOL	1
In5	Input In5	STRUCT	-
		Value: BOOL	0
		ST: BYTE	16#FF
In5MsgEn	1 = Activate message for In5 input	BOOL	1
In6	Input In6	STRUCT	-
		Value: BOOL	0
		ST: BYTE	16#FF
In6MsgEn	1 = Activate message for In6 input	BOOL	1
In7	Input In7	STRUCT	-
		Value: BOOL	0
		ST: BYTE	16#FF
In7MsgEn	1 = Activate message for In7 input	BOOL	1
In8	Input In8	STRUCT	-
		Value: BOOL	0
		ST: BYTE	16#FF
In8MsgEn	1 = Activate message for In8 input	BOOL	1
InvIn1	1 = Invert input In1	BOOL	0
InvIn2	1 = Invert input In2	BOOL	0
InvIn3	1 = Invert input In3	BOOL	0
InvIn4	1 = Invert input In4	BOOL	0
InvIn5	1 = Invert input In5	BOOL	0
InvIn6	1 = Invert input In6	BOOL	0
InvIn7	1 = Invert input In7	BOOL	0

Parameter	Description	Type	Default
InvIn8	1 = Invert input In8	BOOL	0
MS_RelOp	1 = Maintenance release by OS operator	BOOL	0
MsgEvId	Message number (assigned automatically)	DWORD	16#FFFFFFFF
MsgLock	1 = Suppress process messages	STRUCT	-
		Value: BOOL	0
		ST: BYTE	16#80
OnOp	1 = "On" mode via operator	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC (0-1 edge transition)	STRUCT	-
		Value: BOOL	0
		ST: BYTE	16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OS_Perm	I/O for operator control permissions (Page 983)	STRUCT	-
		Bit 0: BOOL	1
		...	1
		Bit 31: BOOL	1
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SelFp1	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
SelFp2	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
UserStatus	Freely assignable user area for status word (bit 24 to bit 31)	BYTE	16#00

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see EventNck error handling (Page 985).	INT	-1
MsgErr	1 = Alarm error (output ERROR of NOTIFY_8P)	BOOL	0
MsgStat	Alarm status (output STATUS of NOTIFY_8P)	WORD	16#0000
MS_Release	Maintenance release: 1 = Release for OS operator	STRUCT Value: BOOL ST: BYTE	- 0 16#80
OnAct	1 = "On" mode enabled	STRUCT Value: BOOL ST: BYTE	- 1 16#80
OosAct	1 = Block is "Out of service"	STRUCT Value: BOOL ST: BYTE	- 0 16#80
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word 1 (Page 979)	DWORD	16#00000000
Status2	Status word 2 (Page 979)	DWORD	16#00000000

See also

EventNck modes (Page 982)

EventNck messaging (Page 986)

EventNck block diagram (Page 992)

Functions that can be set via the Feature I/O (Page 186)

8.2.7 EventNck block diagram

EventNck block diagram

A block diagram is not provided for this block.

See also

Description of EventNck (Page 979)
EventNck modes (Page 982)
EventNck functions (Page 983)
EventNck error handling (Page 985)
EventNck messaging (Page 986)
EventNck I/Os (Page 988)

8.3 EventTs - Creating messages with time stamp

8.3.1 Description of EventTs

Object name (type + number) and family

Type + number: FB 1812

Family: Report

Area of application for EventTs

The block is used for the following applications:

- Generating messages which have to be acknowledged for time-stamped signals

How it works

The block must be linked to a driver block and monitors up to eight different binary signals. From these, it generates time-stamped messages requiring acknowledgment that appear in the message view of the technological block to which it is connected.

The individual messages are assigned to the monitored signals via the inputs and the messages released or blocked depending on the process status data. If a change is made to one or more of the monitored signals enabled for reporting, a message is output containing a time stamp for the signal change.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB100) and in OB1 because of the time stamp.

In CFC interconnect the `EventTsOut` output parameter of the EventTs block to the `EventTsIn` input parameter of the technological block.

Depending on how the time stamp is generated, the signals to be monitored for this are interconnected at inputs `In1` to `In8` or `InTS1` to `InTS8`. Each In_x or $InTS_x$ signal can also be inverted via input Inv_x. A message is output if a signal value is changed (but taking inversion into account).

Each input is assigned a separate message text. A message received at input `InTS5`, for example, is output with the message text for the `SIG5` signal.

For the EventTs block, the Advanced Process Library contains process tag type templates; these serve as examples by providing various application scenarios for this block.

Examples of process tag types:

- Motor with an additional analog value and time-stamped signals (Motor_AV_EventTs) (Page 1451)

Startup characteristics

During startup, the message block suppresses all messages. The duration (number of cycles) of message suppression is set at the RunUpCyc parameter. An internal counter that is initialized with this parameter value during restart (OB100) decrements each time the block is executed. Messages are not generated if the counter value does not equal zero.

Status word allocation for `Status1` parameter

Status bit	Parameter
0	Active signal 1
1	Active signal 2
2	Active signal 3
3	Active signal 4
4	Active signal 5
5	Active signal 6
6	Active signal 7
7	Active signal 8
8	InvIn1
9	InvIn2
10	InvIn3
11	InvIn4
12	InvIn5
13	InvIn6
14	InvIn7
15	InvIn8
16	Active signal 1 with inversion
17	Active signal 2 with inversion
18	Active signal 3 with inversion
19	Active signal 4 with inversion
20	Active signal 5 with inversion
21	Active signal 6 with inversion
22	Active signal 7 with inversion
23	Active signal 8 with inversion
24	Active signal 1 not interconnected
25	Active signal 2 not interconnected
26	Active signal 3 not interconnected
27	Active signal 4 not interconnected
28	Active signal 5 not interconnected
29	Active signal 6 not interconnected
30	Active signal 7 not interconnected
31	Active signal 8 not interconnected

Status word allocation for Status2 parameter

Status bit	Parameter
0	In1MsgEn
1	In2MsgEn
2	In3MsgEn
3	In4MsgEn
4	In5MsgEn
5	In6MsgEn
6	In7MsgEn
7	In8MsgEn
8	Effective signal 1 for message
9	Effective signal 2 for message
10	Effective signal 3 for message
11	Effective signal 4 for message
12	Effective signal 5 for message
13	Effective signal 6 for message
14	Effective signal 7 for message
15	Effective signal 8 for message
16	MsgLock
17 - 31	Not used

See also

EventTs block diagram (Page 1007)
EventTs I/Os (Page 1002)
EventTs messaging (Page 1000)
EventTs error handling (Page 999)
EventTs functions (Page 997)
EventTs modes (Page 996)

8.3.2 EventTs modes

EventTs operating modes

The block provides the following modes:

- On (Page 27)
- Out of service (Page 27)

"On"

General information on the "On" mode is available in the section On (Page 27).

"Out of service"

You will find general information about the "Out of service" mode in the Out of service (Page 27) section.

See also

EventTs block diagram (Page 1007)

EventTs I/Os (Page 1002)

EventTs messaging (Page 1000)

EventTs error handling (Page 999)

EventTs functions (Page 997)

Description of EventTs (Page 993)

8.3.3 EventTs functions

Functions of EventTs

The functions for this block are listed below.

Activation and deactivation of messages

Set the I/Os `InMsgEn1` to `InMsgEn8` accordingly to activate or deactivate the messages applied to inputs `In1` to `In8` or `InTS1` to `InTS8`. All messages are activated by default.

To deactivate messages received at I/O `InTS4`, for example, you set I/O `InMsgEn4 = 0` accordingly.

You can deactivate all messages via I/O `MsgLock = 1`.

Time stamp as associated value of a message

You can use input `TimeStampOn` to select how the time stamp for the signals of EventTs will be generated:

- If you want to use the high-precision time stamp from the I/O devices, set `TimeStampOn = 0`. Interconnect one of the $\texttt{InTS}_x$ inputs with output `TS_Out` of block Pcs7DilT.
- If you want to use the time stamp from the CPU, set `TimeStampOn = 1`. Interconnect one of the $\texttt{In}_x$ inputs with output `PV_Out` of block Pcs7DilT or with an appropriate output of a different block.

For additional time stamp properties, refer to the description of the standard function Time stamp (Page 51).

Signal status as associated value of a message

The signal status as an associated value of the message is output for every signal, as is the time stamp.

Maintenance release

This block provides the standard function Maintenance release (Page 47).

Operator control permissions via parameter OS_Perm

The block has the following Operator permissions (Page 45) for the OS_Perm parameter:

Bit	Function
0	Not allocated
1	1 = Operator can switch to "On" mode
2	Not allocated
3	1 = Operator can switch to out of service mode
4 - 31	Not allocated

This block does not have a faceplate yet; the operator control permissions have already been assigned in the planning phase for these faceplates.

Configurable functions using the Feature parameter

You can find an overview of all reactions provided by the Feature I/O in the section Functions of blocks > Functions that can be set via the Feature I/O (Page 186). The following functionality is available for this block at the relevant bits:

Bit	Function
1	Reaction to the out of service mode (Page 199)
22	Update acknowledgment and error status of the message call (Page 193)

Displaying and outputting the signal status

The block determines the worst signal status via all interconnected binary inputs (according to TimeStampOn) and outputs this value at ST_Worst.

You can find general information about the signal status in the section Functions of blocks > Forming and outputting signal status for blocks (Page 88).

- TimeStampOn = 0
  - In1

  etc. to

  - In8
- TimeStampOn=1
  - InTS1

  etc. to

  - InTS8

See also

EventTs block diagram (Page 1007)

EventTs I/Os (Page 1002)

EventTs messaging (Page 1000)

EventTs error handling (Page 999)

EventTs modes (Page 996)

Description of EventTs (Page 993)

8.3.4 EventTs error handling

EventTs troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions for the troubleshooting of all blocks.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Predefined value when inserting the block, the block is not processed.
0	There is no error.

See also

EventTs block diagram (Page 1007)

EventTs I/Os (Page 1002)

EventTs messaging (Page 1000)

EventTs functions (Page 997)

EventTs modes (Page 996)

Description of EventTs (Page 993)

8.3.5 EventTs messaging

Messaging

Messages which can be acknowledged are generated using ALARM_8P. ALARM_8P is assigned eight digital inputs and 10 associated values. Every edge transition detected for one or more digital inputs results in a message. The associated values are assigned consistently to the message at the time of edge evaluation. All eight signals are assigned a common message number, which is split at the OS into eight messages. The ES assigns the message number automatically by calling the message server.

Process messages

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 1	AS process control message - fault	$$BlockComment$$ InTS1 Status 16#@1%x@
	SIG 2	AS process control message - fault	$$BlockComment$$ InTS2 Status 16#@2%x@
	SIG 3	AS process control message - fault	$$BlockComment$$ InTS3 Status 16#@3%x@
	SIG 4	AS process control message - fault	$$BlockComment$$ InTS1 Status 16#@4%x@
	SIG 5	AS process control message - fault	$$BlockComment$$ InTS5 Status 16#@5%x@
	SIG 6	AS process control message - fault	$$BlockComment$$ InTS6 Status 16#@6%x@
	SIG 7	AS process control message - fault	$$BlockComment$$ InTS7 Status 16#@7%x@
	SIG 8	AS process control message - fault	$$BlockComment$$ InTS8 Status 16#@8%x@

Explanation:

$$BlockComment$$: Content of the instance-specific comment

16#@n%x@ (n = 1 ... 8): Statement on the validity of the time stamp. If the value is 80, then the time stamp is valid. If the value is ≠ 80, then the time stamp is inaccurate. In other words, the time stamp does not come from the I/O devices but the CPU.

Associated values for message instance `MsgEvId`

You can use the `Feature` bit Selecting values associated with messages (Page 192) to define whether the signal status of the signal or the associated analog value is to be used as the associated value for the message.

Associated value	Block parameters
1	`TimeStampOn = 1: In1.ST`
2	`TimeStampOn = 1: In2.ST`
3	`TimeStampOn = 1: In3.ST`
4	`TimeStampOn = 1: In4.ST`
5	`TimeStampOn = 1: In5.ST`
6	`TimeStampOn = 1: In6.ST`
7	`TimeStampOn = 1: In7.ST`
8	`TimeStampOn = 1: In8.ST`
9	Not allocated
10	Not allocated

Associated values for message instance `MsgEvId`

Associated value	Block parameters
1	`TimeStampOn = 1: InTS1.ST`
2	`TimeStampOn = 1: InTS2.ST`
3	`TimeStampOn = 1: InTS3.ST`
4	`TimeStampOn = 1: InTS4.ST`
5	`TimeStampOn = 1: InTS5.ST`
6	`TimeStampOn = 1: InTS6.ST`
7	`TimeStampOn = 1: InTS7.ST`
8	`TimeStampOn = 1: InTS8.ST`
9	Not allocated
10	Not allocated

See also

EventTs block diagram (Page 1007)

EventTs I/Os (Page 1002)

EventTs error handling (Page 999)

EventTs functions (Page 997)

EventTs modes (Page 996)

Description of EventTs (Page 993)

8.3.6 EventTs I/Os

EventTs I/Os

Inputs

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
Feature	I/O for additional functions (Page 186)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
In1	Signal 1 for CPU time stamp	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In2	Signal 2 for CPU time stamp	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In3	Signal 3 for CPU time stamp	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In4	Signal 4 for CPU time stamp	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In5	Signal 5 for CPU time stamp	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In6	Signal 6 for CPU time stamp	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In7	Signal 7 for CPU time stamp	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In8	Signal 8 for CPU time stamp	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF

Parameter	Description	Type	Default
InTS1	Signal 1 with I/O-device time stamp	STRUCT • MsgSig: BOOL • TriInf: BOOL • HdSh: BOOL • ST: BYTE • TS0: DWORD • TS1: DWORD	- • 0 • 0 • 0 • 16#FF • 16#00000000 • 16#00000000
InTS2	Signal 2 with I/O-device time stamp	STRUCT • MsgSig: BOOL • TriInf: BOOL • HdSh: BOOL • ST: BYTE • TS0: DWORD • TS1: DWORD	- • 0 • 0 • 0 • 16#FF • 16#00000000 • 16#00000000
InTS3	Signal 3 with I/O-device time stamp	STRUCT • MsgSig: BOOL • TriInf: BOOL • HdSh: BOOL • ST: BYTE • TS0: DWORD • TS1: DWORD	- • 0 • 0 • 0 • 16#FF • 16#00000000 • 16#00000000
InTS4	Signal 4 with I/O-device time stamp	STRUCT • MsgSig: BOOL • TriInf: BOOL • HdSh: BOOL • ST: BYTE • TS0: DWORD • TS1: DWORD	- • 0 • 0 • 0 • 16#FF • 16#00000000 • 16#00000000
InTS5	Signal 5 with I/O-device time stamp	STRUCT • MsgSig: BOOL • TriInf: BOOL • HdSh: BOOL • ST: BYTE • TS0: DWORD • TS1: DWORD	- • 0 • 0 • 0 • 16#FF • 16#00000000 • 16#00000000
InTS6	Signal 6 with I/O-device time stamp	STRUCT • MsgSig: BOOL • TriInf: BOOL • HdSh: BOOL • ST: BYTE • TS0: DWORD • TS1: DWORD	- • 0 • 0 • 0 • 16#FF • 16#00000000 • 16#00000000

Parameter	Description	Type	Default
InTS7	Signal 7 with I/O-device time stamp	STRUCT • MsgSig: BOOL • TriInf: BOOL • HdSh: BOOL • ST: BYTE • TS0: DWORD • TS1: DWORD	- • 0 • 0 • 0 • 16#FF • 16#00000000 • 16#00000000
InTS8	Signal 8 with I/O-device time stamp	STRUCT • MsgSig: BOOL • TriInf: BOOL • HdSh: BOOL • ST: BYTE • TS0: DWORD • TS1: DWORD	- • 0 • 0 • 0 • 16#FF • 16#00000000 • 16#00000000
Inv1	1 = Invert input In1 or InTS1	BOOL	0
Inv2	1 = Invert input In2 or InTS2	BOOL	0
Inv3	1 = Invert input In3 or InTS3	BOOL	0
Inv4	1 = Invert input In4 or InTS4	BOOL	0
Inv5	1 = Invert input In5 or InTS5	BOOL	0
Inv6	1 = Invert input In6 or InTS6	BOOL	0
Inv7	1 = Invert input In7 or InTS7	BOOL	0
Inv8	1 = Invert input In8 or InTS8	BOOL	0
MsgEn1	1 = Activate message for input In1 or InTS1	BOOL	1
MsgEn2	1 = Activate message for input In2 or InTS2	BOOL	1
MsgEn3	1 = Activate message for input In3 or InTS3	BOOL	1
MsgEn4	1 = Activate message for input In4 or InTS4	BOOL	1
MsgEn5	1 = Activate message for input In5 or InTS5	BOOL	1
MsgEn6	1 = Activate message for input In6 or InTS6	BOOL	1
MsgEn7	1 = Activate message for input In7 or InTS7	BOOL	1
MsgEn8	1 = Activate message for input In8 or InTS8	BOOL	1
MsgEvId	Message number (assigned automatically)	DWORD	16#FFFFFFFF

Parameter	Description	Type	Default
MsgLock	1 = Suppress process messages	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_RelOp	1 = Maintenance release by OS operator	BOOL	0
OnOp	1 = "On" mode via operator	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC (0-1 edge transition)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OS_Perm	I/O for operator control permissions (Page 997)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SelFp1	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
TimeStampOn	0 = Use time stamp of the I/O devices 1 = Use time stamp of the CPU	BOOL	0
UserStatus	Freely assignable bits for use in faceplate	BYTE	16#00

Outputs

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see EventTs error handling (Page 999)	INT	-1
EventTsOut	For wiring the input signals of a technological block	STRUCT • Value: BYTE • ST: BYTE	- • 16#00 • 16#80
MS_Release	Maintenance release: 1 = Release for OS operator	BOOL	0
MsgAckn	Message acknowledgment status (output ACK_STATE of ALARM_8P)	WORD	16#0000
MsgErr	1 = Alarm error (output ERR0R of ALARM_8P)	BOOL	0
MsgStat	Alarm status (output STATUS of ALARM_8P)	WORD	16#0000
OnAct	1 = "On" mode enabled	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
OosAct	1 = Block is "Out of service"	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word 1 (Page 993)	DWORD	16#00000000
Status2	Status word 2 (Page 993)	DWORD	16#00000000

See also

EventTs block diagram (Page 1007)

EventTs messaging (Page 1000)

EventTs modes (Page 996)

8.3.7 EventTs block diagram

EventTs block diagram

A block diagram is not provided for this block.

See also

EventTs I/Os (Page 1002)
EventTs messaging (Page 1000)
EventTs error handling (Page 999)
EventTs functions (Page 997)
EventTs modes (Page 996)
Description of EventTs (Page 993)

Dosing blocks

9

9.1 DoseL - Dosing device

9.1.1 Description of DoseL

Object name (type + number) and family

Type + number: FB 1809

Family: Dosage

Area of application for DoseL

The block is used for the following applications:

- Single-component dosing using flow measurement
- Weighing of fill/removal volume using dosing scales

How it works

Dosing using flow measurement is performed discretely via a coarse/fine flow control with flow monitoring and setpoint input. The dosing flow can be determined via a maximum of 16 cycles.

Flow monitoring is not carried out for the weighing/dosing process. The dosing procedure is performed via a coarse/fine flow control.

A dribble phase and dribbling correction can be implemented for both processes. The input value can also be supplied by a pulse module.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

For the DoseL block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Dosing (DoseLean) (Page 1449)

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

Status word allocation for `Status1` parameter

For a description of the individual parameters, see the DoseL I/Os (Page 1031) section.

Status bit	Parameter
0	`Occupied`
1	`BatchEn`
2	`SimOn`
3	`OosAct.Value`
4	`OosLi.Value`
5	`AutoAct.Value`
6	`LocalAct.Value`
7	`LockAct.Value`
8	`SP_ExtAct`
9	`Ctrl`
10	`Ctrl2`
11	1 = Dosing using scales
12	0 = Dosing using scales, filling, 1 = Dosing using scales, removal
13	`BypProt` active
14	Invalid signal status
15	Mode changeover error
16	`1 = Intlock` is active
17	`1 = Permit` is active
18	`1 = Protect` is active
19	`DosOn`
20	`DosRelax`
21	`DosEnd`
22	`DosOff`
23	`DosPause`
24	`DosStart`
25	1 = Dribble correction
26	"Start" command
27	"Pause" command
28	"Continue" command
29	"Cancel" command
30	`UserAna1` interconnected
31	`UserAna2` interconnected

Status word allocation for `Status2` parameter

Status bit	Parameter
0	`MsgLock`
1	`DQ_AH_Act`
2	`DQ_AL_Act`
3	`DQ_AH_En`
4	`DQ_AL_En`
5	`DQ_AH_MsgEn`
6	`DQ_AL_MsgEn`
7	`PV_AH_Act`
8	`PV_AL_Act`
9	`PV_AH_En`
10	`PV_AL_En`
11	`PV_AH_MsgEn`
12	`PV_AL_MsgEn`
13	`PV_AH2_Act`
14	`PV_AL2_Act`
15	`PV_AH2_En`
16	`PV_AL2_En`
17	`PV_AH2_MsgEn`
18	`PV_AL2_MsgEn`
19	`CR_AH_Act`
20	`CR_AH_En`
21	`CR_AH_MsgEn`
22	Display for interlocks in block icon
23	1 = Scales tared
24	Automatic preview 1 = Dosing on
25	Automatic preview 1 = Dosing dribble
26	Automatic preview 1 = Dosing end
27	Automatic preview 1 = Dosing off
28	Automatic preview 1 = Dosing pause
29	Forcing active
30	Bypass information from previous function block
31	`MS_RelOp`

Status word allocation for `Status3` parameter

Status bit	Parameter
0	"Interlock" button is enabled
1	"Permission" button is enabled
2	"Protection" button is enabled
3	`DosCancelMsgEn`
4 - 18	Not used
19	`StartForce`
20	`CancelForce`
21	`PauseForce`
22	`ContForce`
23 - 31	Not used

Status word allocation for `Status4` parameter

Status bit	Parameter
0	Effective `Signal1` of the message block connected via `EventTsIn`
1	Effective `Signal2` of the message block connected via `EventTsIn`
2	Effective `Signal3` of the message block connected via `EventTsIn`
3	Effective `Signal4` of the message block connected via `EventTsIn`
4	Effective `Signal5` of the message block connected via `EventTsIn`
5	Effective `Signal6` of the message block connected via `EventTsIn`
6	Effective `Signal7` of the message block connected via `EventTsIn`
7	Effective `Signal8` of the message block connected via `EventTsIn`
8 - 31	Not used

See also

DoseL functions (Page 1015)
DoseL messaging (Page 1028)
DoseL modes (Page 1013)
DoseL error handling (Page 1026)
DoseL block diagram (Page 1045)

9.1.2 DoseL modes

DoseL operating modes

The block can be operated using the following modes:

- Local mode (Page 36)
- Automatic mode (Page 32)
- Manual mode (Page 32)
- Out of service (Page 27)

The next section provides additional block-specific information relating to the general descriptions.

"Local mode"

You can find general information on "Local mode", switching modes and bumpless switchover in the Local mode (Page 36) section.

Dosing actions you can control in local mode:

- Start (positive edge `StartLocal`)
- Cancel (`CancelLocal = 1`)
- Pause (`PauseLocal = 1`)
- Continue (`ContLocal = 1`)

If you operate the block in local mode, control will only take place via local signals (input parameters `StartLocal = 1`, `CancelLocal = 1`, `PauseLocal = 1`, and `ContLocal = 1`).

Note

Unlike the general description, only the 0, 1 and 3 values can be set at the `LocalSetting` parameter, i.e. tracking in local mode is not possible with DoseL.

"Automatic mode"

You can find general information on "Automatic mode", switching modes and bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 32) section.

Dosing actions you can control in automatic mode:

- Start (positive edge `StartAut`)
- Cancel (`CancelAut = 1`)
- Pause (`PauseAut = 1`)
- Continue (`ContAut = 1`)

"Manual mode"

You can find general information on "Manual mode", switching modes and bumpless switchover in the Manual and automatic mode for motors, valves and dosers (Page 32) section.

Dosing actions you can control in manual mode:

- Start (`StartMan = 1`)
- Cancel (`CancelMan = 1`)
- Pause (`PauseMan = 1`)
- Continue (`ContMan = 1`)

"Out of service"

You will find general information about the "Out of service" mode in the Out of service (Page 27) section.

See also

Description of DoseL (Page 1009)

DoseL functions (Page 1015)

DoseL error handling (Page 1026)

DoseL messaging (Page 1028)

DoseL I/Os (Page 1031)

DoseL block diagram (Page 1045)

9.1.3 DoseL functions

Functions of DoseL

The functions for this block are listed below.

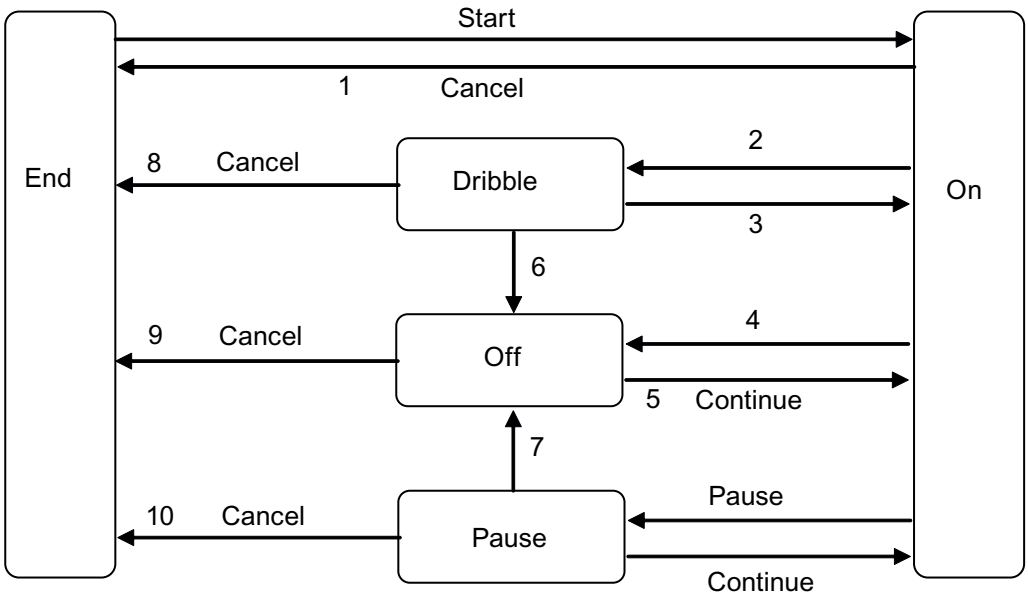
Status diagram

The block provides the following states:

- End
- On
- Dribble
- Off
- Pause

Statuses are changed using the following commands:

- Start
- Cancel
- Pause
- Continue



Dosing can only be started if the dosing setpoint is greater than the current dosing quantity (`DQ_Out`) or if `Feature` bit 8 is set and no interlock is pending.

However, the following status changes are also performed automatically:

Number in graphic (top)	Function
1	If the dosing quantity is reached, dosing finishes (DQ_Out ≥ DQ_SP).
2	If the dosing quantity reaches the dribble quantity (relative to dosing setpoint DQ_Out ≥ DQ_SP – DribbOut)), the Dribble status becomes active.
3	Automatic dribbling in automatic mode via the Feature bit 12, if an underdosage is identified after the dribble time (RelaxTime).
4	• Flow alarm (see input parameter Feature bit 11) or • interlock.
5	• The flow alarm is acknowledged (also provided via the "Continue" command if Feature bit 9 = 1) or • The interlock is acknowledged (also provided via the "Continue" command if Feature bit 9 = 1). • "Continue" command after underdosing.
6	Underdosage identified after the dribble time (RelaxTime).
7	Interlock
8	Dribble time (RelaxTime) has expired and dosing quantity is above the low tolerance limit (DQ_Out >= DQ_SP - DQ_AL_Tol).
9	The underdosage is acknowledged or the dosing quantity has been reached (DQ_Out ≥ DQ_SP) via creep flow, for example.
10	Dosing quantity has been reached (DQ_Out ≥ DQ_SP) via creep flow, for example.

Control outputs

The coarse flow (Ctrl) or fine flow (Ctrl2) control outputs are issued in the "On" status.

The coarse flow control output is active if:

- DQ_Out < DQ_SP – DQ2_SP

The fine flow control output is active if:

- DQ_Out >= DQ_SP – DQ2_SP
- Where DQ_Out: Dosing quantity actual value
- DQ_SP: Dosing quantity setpoint
- DQ2_SP: Fine dosing quantity setpoint (calculated)

Output signal as a pulse signal or static signal

This block provides the standard function Output signal as a static signal or pulse signal (Page 86).

In addition to the static control outputs Ctrl and Ctrl2, the block also has pulse outputs P_Ctrl and P_Ctrl2, which are dependent on the static control outputs.

Determining the dosing quantity when using flow dosing

When using flow dosing, the dosing quantity is determined using the following equation:

- `DQ_Out = DQ_Out + SampleTime/TI · (PV_OUTalt + PV_OUT)/2`.

Where:

- `DQ_Out`: Dosing quantity actual value
- `SampleTime`: Sampling time [s]
- `TI`: Integral time constant [s]
- `PV_OUTalt`: Last value `PV_OUT`

The output parameter `PV_OUT` is determined using an average calculation:

- `PV_OUT := (PV_OUT(t) + PV_OUT(t-1) + ... + PV_OUT(t- N+1))/ NumSample (N)`

Where:

- `PV_OUT(t) = PV *Gain`

The average calculation is only active if `NumSample > 1` and `NumSample ≥16` and serves to smooth systematically pulsating measuring signals.

Parameters `Gain` and `TI` can be used to adapt the calculation of the dosing quantity to a pulse module.

Example: If one pulse is 14 kg/h, `TI` is set to 3600 and `Gain` to 14.

The dosing quantity is determined in the "On" and "Tracking" state. In the "End", "Off" and "Pause" states, the flow (creep rate) is determined based on `Feature` bit 13 (Creep rate is always detected in the dosing quantity (Page 195)) and the `CR_AH_Lim` value. Creep rate monitoring is disabled with `CR_AH_En = 0`.

Determining the dosing quantity when dosing using scales

The dosing quantity is determined for a scale dosing in the "Start" state following a positive edge at the input parameter `StandStill` or in the "Tracking" state. The `StandStill` input parameter is a feedback signal of the scale.

If the signal is no longer available, you need to configure `StandStill` with 1 permanently; the dosing quantity will then be determined right at the start of dosing.

When weighing the fill volume (`MeterType = 0`), the dosing quantity is determined using the following equations:

- `DQ_Out = PV_OUT - DQ_Tare`

When weighing the removal volume (`MeterType = 1`):

- `DQ_Out = DQ_Tare - PV_Out`

with `PV_Out = PV · Gain`

In the "Start" state, the `DQ_Tare` tare memory is set with `PV_Out` with the first positive edge of `StandStill`.

If you have configured `StandStill` with 1 permanently, the tare memory is set right at the start of dosing.

Dribble

The "Dribble" status is entered automatically in accordance with

- `DQ_Out => DQ_SP – DribbOut`

Where:

- `DQ_Out`: Dosing quantity actual value
- `DQ_SP`: Dosing quantity setpoint
- `DribbOut`: Dribble quantity

The dribble quantity is specified with the input `DribbIn`.

`DribCor = 1` can be used to automatically determine the dribble quantity from previous dosing actions:

- `DribbOut = DribbOut - (DQ_SP - DQ_Out) * DCF/100`

`DCF` represents the weighting factor of the last dosing action as a percentage and cannot be set to below 0 or above 100. `DribbOut` is calculated at the end of dosing or the first time an underdosage occurs, and is limited to `DribbMax`. Dribble correction is not taken into account.

The "Dribble" status is active for the period `RelaxTime`. `DribbOut = 0` deactivates the "Dribble" status.

Overdosing/underdosing

Once the "Dribble" status has expired, the dosing quantity is checked for overdosages or underdosages. If the "Dribble" status is deactivated, the check is performed in the "End" status.

An overdosage has occurred if:

- `DQ_Out > DQ_SP + DQ_AH_Tol`

An underdosage has occurred if:

- `DQ_Out < DQ_SP - DQ_AL_Tol`
- Where `DQ_Out`: Dosing quantity actual value
- `Q_SP`: Dosing quantity setpoint
- `DQ_AH_Tol`: High tolerance limit
- `DQ_AL_Tol`: Low tolerance limit

If an underdosage is identified after the "Dribble" status, the "Off" status is entered. The underdosage can be acknowledged in this status (`U_AckOp = 1` or positive edge `U_AckLi`) and the dosing action completed, or dribble correction can be started with the "Continue" command.

Dribble correction

If an underdosage is detected after dribbling, dribble correction can be performed using the "Continue" command. In automatic mode, feature bit 12 can be used to start dribble correction automatically. Dribble correction is active for the period P_DoseTime. The block then enters the "Dribble" status or, if that status is deactivated, the "End" status. If the conditions for the "Dribble" or "End" states are met within the period P_DoseTime, the block switches to these states immediately.

Dribble correction is not possible if dosing is cancelled.

Resetting the dosing quantity

The dosing quantity can only be reset in the "End" status using RstDQ_Op = 1 or positive edge RstDQ_Li.

Feature bit 8 can be used to reset the dosing quantity automatically when dosing starts.

External/internal setpoint specification

This block provides the standard function Setpoint input - internal and external (Page 96).

The block always requires the setpoint for the dosing quantity (dosing setpoint). All I/Os for the dosing setpoint start with DQ_... . The dosing setpoint is comprised of the coarse and fine setpoints for the coarse/fine flow control. All I/Os for the fine setpoint start with DQ2_... . The coarse setpoint is generated internally from the dosing and fine setpoints and is displayed at the output with DQ1_SP. If no fine setpoint is available, the coarse setpoint is equal to the dosing setpoint.

Flow setpoints can be predefined for the coarse/fine flow setpoint input. All I/Os for the coarse flow start with SP_... , and all I/Os for the fine flow with SP2_... . The setpoint for coarse or fine flow is displayed at the output SP in the "On" status, depending on the coarse/fine flow control.

Setpoint limitation

The setpoints are limited in the block using shared parameters:

- Coarse metering volume: DQ_HiLim, DQ_LoLim
- Fine metering volume: DQ2_HiLim, DQ2_LoLim
- Coarse flow rate: SP_HiLim, SP_LoLim
- Fine flow rate: SP2_HiLim, SP2_LoLim

If the setpoint limit is violated, external setpoints are limited to the limit you specified. If there are internal setpoints, the last valid value is output.

Bumpless switchover from external to internal setpoint

The parameter SP_TrkExt = 1 is used so that the internal setpoint tracks the external setpoint to achieve a bumpless switchover from the external to the internal setpoint. This allows unwanted jumps at the output parameter to be avoided.

Limit monitoring of the process value

This block provides the standard function Limit monitoring of the process value (Page 70). The calculated flow PV_OUT is monitored in terms of the limit pairs PV_AH_Lim/PV_AL_Lim (coarse flow) or PV_AH2_Lim/PV_AL2_Lim (fine flow), depending on the coarse/fine flow control. Monitoring is only active in the "On" status.

Forcing operating states

This block provides the standard function Forcing operating states (Page 115). Inputs StartForce, CancelForce, PauseForce, and ContForce force the blocks into the states "On", "End", or "Pause". The "Pause" status can only be forced if the block is in the "On" status. The "On" status can only be forced from the "End", "Off", and "Pause" states, and only if the dosing setpoint has not yet been reached.

Simulating signals

This block provides the standard function Simulating signals (Page 93).

You can simulate the following values:

- Process value (SimPV)

Interlocks

This block provides the following interlocks:

- Activation enable
- Interlock without reset (interlock)
- Interlock with reset (protection)

Refer to the section Interlocks (Page 100).

Dosing can only be started if no interlock is present. The interlock function is not active in the "Dribble" status; nor is the activation enable active in the "On" status. An interlock in the "On" or "Pause" states will cause the block to enter the "Off" status.

Disabling interlocks

This block provides the standard function Disabling interlocks (Page 107).

Resetting the block in case of interlocks or errors

This block provides the standard function Resetting the block in case of interlocks or errors (Page 119).

Forming the group status for interlocks

This block provides the standard function Forming the group status for interlock information (Page 105).

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status ST_Worst for the block is formed by the following parameters:

- DQ_Out.ST
- DQ_SP.ST
- SP.ST
- Permit.ST
- Inlock.ST
- Protect.ST
- Standstill.ST

The following signal status is formed by:

Signal status	Used status
DQ_Out.ST	Formed by the PV_Out.ST signal status and the last DQ_Out.ST, the DQ_Tare.ST signal status is also included for scale dosing . The worst signal status is applied. The signal status is reset with Reset.
DQ_SP.ST	With an external setpoint DQ_Ext.ST (for DQ_HiLim.ST or DQ_LoLim.ST limit), otherwise good status
SP.ST	With an external setpoint SP_Ext.ST (coarse flow) / SP2_Ext.ST (fine flow) (with SP_HiLim.ST or SP_LoLim.ST or SP2_HiLim.ST or SP2_LoLim.ST limit), otherwise good status

Maintenance release

This block provides the standard function Maintenance release (Page 47).

Alarm delays with two time values per limit pair

This block includes the standard function alarm delay for Two time values per limit pair (Page 80) .

Display and control field for process values and setpoints

This block includes the standard function Display and operator input area for process values and setpoints (Page 42).

Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 53).

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 46).

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)
1	Reaction to the out of service mode (Page 199)
2	Resetting the commands for changing the mode (Page 193)
3	Enabling resetting of commands for the control settings (Page 193)
4	Setting switch or button mode (Page 195) *1
5	Specifying the dosing type (Page 189)
6	Resetting the dosing quantity when dosing starts (Page 194)
9	Resetting via input signals in the event of interlocks or errors (Page 194)
10	Exiting local mode (Page 199)
11	Stopping dosing at a flow alarm (Page 187)
12	Automatic dribble correction for underdosing in automatic mode (Page 188)
13	Creep rate is always detected in the dosing quantity (Page 195)
17	Enabling bumpless changeover to automatic mode for valves, motors, and dosers (Page 197)
22	Update acknowledgment and error status of the message call (Page 193)

Comments on table

*1Pushbutton operation (`Feature` bit 4 = 0): The automatic commands in automatic mode are pulse signals, in other words `StartAut`, `CancelAut`, `PauseAut`, and `ContAut` can be reset to zero after changeover to the selected status. In manual and local modes, the automatic commands are static signals and the block state is tracked in the absence of automatic commands.
Pushbutton operation (`Feature` bit 4 = 1): The states are selected with static signals via the inputs `StartAut` (0: Cancel, 1: Start) and `PauseAut` (0: Continue, 1: Pause).

Operator control permissions via parameters OS_Perm and OS1Perm

The block has the following operator control permissions for the OS_Perm parameter:

Bit	Function
0	1 = Operator can switch to automatic mode
1	1 = Operator can switch to manual mode
2	1 = Operator can switch to local mode
3	1 = Operator can switch to out of service mode
4	1 = Operator can start dosing
5	1 = Operator can stop dosing
6	1 = Operator can continue dosing
7	1 = Operator can cancel dosing
8	1 = Operator can reset interlock and flow alarms
9	1 = Operator can reset dosing quantity
10	1 = Operator can acknowledge underdosage
11	1 = Operator can select external setpoints
12	1 = Operator can select internal setpoints
13	1 = Operator can enter dosing setpoint
14	1 = Operator can enter dosing setpoint factor for fine dosing
15	1 = Operator can enter flow setpoint for coarse flow
16	1 = Operator can enter flow setpoint factor for fine flow
17	1 = Operator can activate external/internal bumpless changeover
18	1 = Operator can activate the simulation function
19	1 = Operator can activate the maintenance release function
20	Not allocated
21	1 = Operator can enter dribble quantity
22	1 = Operator can enter maximum dribble quantity
23	1 = Operator can activate/deactivate automatic generation of dribble quantity
24	1 = Operator can enter weighting factor for generation of dribble quantity
25 - 31	Not allocated

The block has the following operator control permissions for the OS1Perm parameter:

Bit	Function
0	1 = Operator can change the limit for the high alarm for PV_OUT (coarse flow)
1	1 = Operator can change the limit for the low alarm for PV_OUT (coarse flow)
2	1 = Operator can change the limit for the hysteresis PV_OUT (coarse flow)
3	1 = Operator can change the limit for the high alarm for PV_OUT (fine flow)
4	1 = Operator can change the limit for the low alarm for PV_OUT (fine flow)
5	1 = Operator can change the limit for the hysteresis PV_OUT (fine flow)
6	1 = Operator can change maximum usage limit of the dosing setpoint
7	1 = Operator can change minimum usage limit of the dosing setpoint
8	1 = Operator can change maximum usage limit of the dosing setpoint factor for fine dosing
9	1 = Operator can change minimum usage limit of the dosing setpoint factor for fine dosing
10	1 = Operator can change the tolerance value for overdosage
11	1 = Operator can change the tolerance value for overdosage
12	1 = Operator can change maximum usage limit of the flow setpoint for coarse flow
13	1 = Operator can change minimum usage limit of the flow setpoint for coarse flow
14	1 = Operator can change maximum usage limit of the flow setpoint for fine flow
15	1 = Operator can change minimum usage limit of the flow setpoint for fine flow
16	1 = Operator can change the overrun time
17	1 = Operator can change the follow-up dosing
18	1 = Operator can change the limit for the alarm above for PV_OUT (creep flow)
19	1 = Operator can change the limit for the hysteresis for PV_OUT (creep flow)
20 - 31	Not allocated

Displaying auxiliary values

This block provides the standard function Displaying auxiliary values (Page 52).

Time stamp

This block receives a time stamp value via the EventTSIn input parameter. Refer to EventTs functions (Page 997) for more information on this.

SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 122).

See also

Description of DoseL (Page 1009)
DoseL messaging (Page 1028)
DoseL I/Os (Page 1031)
DoseL modes (Page 1013)
DoseL error handling (Page 1026)
DoseL block diagram (Page 1045)
Ramp limiting of the setpoint (Page 108)
Limit monitoring with hysteresis (Page 78)

9.1.4 DoseL error handling

DoseL troubleshooting

The following errors can be displayed for this block:

- Error numbers
- Mode changeover error

Overview of error numbers

The `ErrorNum` output parameter can be used to output various error numbers. An overview of all error numbers is available in the section "Functions of blocks > Error handling (Page 117)".

Error numbers output by this block:

Error number	Meaning of the error number
-1	Default value when implementing the block; block will not be processed.
0	There is no error.
11	Configuration error `TI <= 0`

12	Configuration error:
	`PV_AH_Lim < PV_AL_Lim`
	`PV_AH2_Lim <= PV_AL2_Lim`
	`DQ_HiLim.Value <= DQ_LoLim.Value`
	`DQ2_HiLim.Value <= DQ2_LoLim.Value`
	`SP_HiLim.Value <= SP_LoLim.Value`
	`SP2_HiLim.Value <= SP2_LoLim.Value`

30	PV is not a valid number
31	`PV_Out` is not a valid number
41	The value for the I/O `LocalSetting` is not within the approved limit of 0, 1, or 3.
42	`LocalSetting = 0` and `LocalLi = 1`
48	`SP_LiOp = 1` and `SP_IntLi = 1` and `SP_ExtLi = 1`
51	`AutModLi = 1` and `ManModLi = 1`or `SP_LiOp = 1` and `SP_IntLi = 1` and `SP_ExtLi = 1` and `Feature` bit Setting switch or button mode (Page 195) = 0 or two I/Os are set at the same time: • in local mode `StartLocal` (pos. edge), `CancelLocal`, `PauseLocal`, `ContLocal` • When forcing states: `StartForce`, `CancelForce`, `PauseForce`, `ContForce` • In automatic mode with pushbutton operation: `StartAut` (pos.edge), `CancelAut`, `PauseAut`, `ContAut` • In manual mode: `StartMan`, `CancelMan`, `PauseMan`, `ContMan`

Mode changeover error

This error can be output by the block, see the section Error handling (Page 117).

See also

Description of DoseL (Page 1009)
DoseL modes (Page 1013)
DoseL functions (Page 1015)
DoseL messaging (Page 1028)
DoseL I/Os (Page 1031)
DoseL block diagram (Page 1045)

9.1.5 DoseL messaging

Messaging

The following messages can be generated for this block:

- Control system fault
- Process messages
- Instance-specific messages

Control system fault

The following control system error messages can be output:

Message instance	Message identifier	Message class	Event
MsgEvId2	SIG 1	AS process control message - fault	$$BlockComment$$ External error has occurred

Explanation:

$$BlockComment$$: Content of the instance-specific comment

You can interconnect an external fault (signal) to input parameter CSF. If this signal changes to CSF = 1, a control system error is triggered (MsgEvId02, SIG 1).

Process messages

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 1	Alarm - high	$$BlockComment$$ Overdosage
	SIG 2	Alarm - low	$$BlockComment$$ Underdosage
	SIG 3	Alarm - high	$$BlockComment$$ PV_OUT - High alarm limit for coarse flow exceeded
	SIG4	Alarm - low	$$BlockComment$$ PV_OUT - Low alarm limit for coarse flow violated
	SIG 5	Alarm - high	$$BlockComment$$ PV_OUT - High alarm limit for coarse flow exceeded
	SIG 6	Alarm - low	$$BlockComment$$ PV_OUT - Low alarm limit for fine flow violated
	SIG 7	Alarm - high	$$BlockComment$$ PV_OUT – Creep flow too great

Message instance	Message identifier	Message class	Event
	SIG 8	Process message - with acknowledgment	$$BlockComment$$ Dosing canceled

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Instance-specific messages

You can use up to three instance-specific messages with this block.

Message instance	Message identifier	Message class	Event
MsgEvId2	SIG 2	AS process control message - fault	$$BlockComment$$ External message 1
	SIG 3	AS process control message - fault	$$BlockComment$$ External message 2
	SIG 4	AS process control message - fault	$$BlockComment$$ External message 3

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Associated values for message instance `MsgEvId1`

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`
4	`DQ_OUT`
5	`PV_OUT`
6	`ExtVal06`
7	`ExtVa107`
8	`ExtVa108`
9	Reserved
10	Reserved

The associated values 6 and 8 are assigned to the parameters `ExtVa106` ... `ExtVa108` and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

Associated values for message instance MsgEvId2

Associated value	Block parameters
1	BatchName
2	StepNo
3	BatchID
4	ExtVa204
5	ExtVa205
6	ExtVa206
7	ExtVa207
8	ExtVa208
9	Reserved
10	Reserved

The associated values 4 ... 8 are allocated to the parameters ExtVa204 ... ExtVa208 and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

See also

Description of DoseL (Page 1009)

DoseL functions (Page 1015)

DoseL I/Os (Page 1031)

DoseL modes (Page 1013)

DoseL error handling (Page 1026)

DoseL block diagram (Page 1045)

Time stamp (Page 51)

9.1.6 DoseL I/Os

DoseL I/Os

Input parameters

Parameter	Description	Type	Default
AutModLi	1 = Automatic mode via interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AutModOp	1 = Automatic mode via operator (controlled by ModLiOp = 0)	BOOL	0
BatchEn	1 = Enable allocation	BOOL	0
BatchID	Batch ID	DWORD	16#00000000
BatchName	Batch name	STRING[32]	''
BypProt	1 = Bypassing interlock in local mode and simulation	BOOL	0
CancelAut	1 = Select "Cancel" in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
CancelForce	1 = Force cancel	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
CancelLocal	1 = Select "Cancel" in local mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
CancelMan	1 = Select "Cancel" in manual mode	BOOL	0
ContAut	1 = Select "Continue" in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ContForce	1 = Force continue	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ContLocal	1 = Select "Continue" in local mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ContMan	1 = Select "Continue" in manual mode	BOOL	0
CR_A_DC	Creep flow: Delay time for incoming alarms [s]	REAL	3.0
CR_A_DG	Creep flow: Delay time for outgoing alarms [s]	REAL	3.0
CR_AH_En	Creep flow: 1 = Enable high alarm	BOOL	1

Parameter	Description	Type	Default
CR_AH_Lim	Creep flow: Limit high alarm	REAL	0.0
CR_AH_MsgEn	Creep flow: 1 = Enable high alarm message	BOOL	1
CR_Hyst	Creep flow: Hysteresis for alarm limit	REAL	1.0
CSF	1 = External error (control system error)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
DCF	Dribbling correction [%]	REAL	25.0
DosCancelMsgEn	1 = Cancel dosing message	BOOL	0
DQ_A_DC	Overdosage/underdosage: Delay time for incoming alarms [s]	REAL	0.0
DQ_A_DG	Overdosage/underdosage: Delay time for outgoing alarms [s]	REAL	0.0
DQ_AH_En	Overdosage: 1 = Enable high alarm	BOOL	1
DQ_AH_MsgEn	Overdosage: 1 = Enable high alarm message	BOOL	1
DQ_AH_Tol	Overdosage: Limit high alarm (relative to dosing setpoint)	REAL	0.0
DQ_AL_En	Underdosage: 1 = Enable low alarm	BOOL	1
DQ_AL_MsgEn	Underdosage: 1 = Enable low alarm message	BOOL	1
DQ_AL_Tol	Underdosage: Limit low alarm (relative to dosing setpoint)	REAL	0.0
DQ_Ext	Dosing quantity: External setpoint	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
DQ_HiLim	Dosing quantity: High setpoint limitation	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
DQ_Int	Dosing quantity: Internal setpoint	REAL	0.0
DQ_LoLim	Dosing quantity: Low setpoint limitation	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
DQ_OpScale	Dosing quantity: Limit for scale in bar graph of faceplate	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
DQ_Unit	Unit of measure for dosing quantity	INT	1088
DQ2_Ext	Dosing quantity: External setpoint factor for fine dosing [%]	STRUCT • Value: REALL • ST: BYTE	- • 0.0 • 16#80
DQ2_HiLim	Dosing quantity: High limitation of setpoint factor for fine dosing [%]	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80

Parameter	Description	Type	Default
DQ2_Int	Dosing quantity: Internal setpoint factor for fine dosing [%]	REAL	0.0
DQ2_LoLim	Dosing quantity: Low limitation of setpoint factor for fine dosing [%]	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
DribbIn	Dribble quantity	REAL	0.0
DribbMax	Maximum value of automatic determination of dribble quantity	REAL	999.0
DribbCor	1 = Automatic determination of dribble quantity	BOOL	0
EN	1 = Called block will be processed	BOOL	1
EventTSIn	For wiring the signal status of an EventTs message block	STRUCT • Value: BYTE • ST: BYTE	- • 16#00 • 16#FF
ExtMsg1	Binary input for freely selectable message 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg2	Binary input for freely selectable message 2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg3	Binary input for freely selectable message 3	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtVa106	Associated value 6 for messages (MsgEvID1)	ANY	
ExtVa107	Associated value 7 for messages (MsgEvID1)	ANY	
ExtVa108	Associated value 8 for messages (MsgEvID1)	ANY	
ExtVa204	Associated value 4 for messages (MsgEvID2)	ANY	
ExtVa205	Associated value 5 for messages (MsgEvID2)	ANY	
ExtVa206	Associated value 6 for messages (MsgEvID2)	ANY	
ExtVa207	Associated value 7 for messages (MsgEvID2)	ANY	
ExtVa208	Associated value 8 for messages (MsgEvID2)	ANY	
Features	I/O for additional functions (Page 1015)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0

Parameter	Description	Type	Default
Gain	Gain factor	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#80
Intlock	0 = Interlocking without reset is effective; you can operate the block without reset once the interlocking condition has disappeared 1 = Good state	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
Intl_En	1 = Interlock without reset (interlock, Intlock parameter) is active	BOOL	1
LocalLi	1 = Activate local mode via plant signals	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
LocalOp	1 = Local mode by operator	BOOL	0
LocalSetting	Properties for the Local mode (Page 36)	INT	0
ManModLi	1 = Manual mode via interconnection or SFC (controlled by ModLiOp = 1)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ManModOp	1 = Manual mode by OS operator (controlled by ModLiOp = 0)	BOOL	1
MeterType	Dosing using scales: 0 = Up; 1 = Down	BOOL	0
ModLiOp	Switchover of operating mode between: 0 = Operator 1 = Interconnection or SFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_RelOp	1= Maintenance release via OS operator	BOOL	0
MsgEvId1	Message number (assigned automatically)	DWORD	16#FFFFFFFF
MsgEvId2	Message number (assigned automatically)	DWORD	16#FFFFFFFF
MsgLock	1 = Suppress process messages	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
NumSample	Number of values for average calculation	INT	0
Occupied	1 = In use by a batch	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0

Parameter	Description	Type	Default
OS_Perm	I/O for operator control permissions (Page 1015)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
OS1Perm	I/O for operator control permissions (Page 1015)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
P_DoseTime	Duration of dribble [s]	REAL	0.0
PauseAut	1 = Select "Pause" in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PauseForce	1 = Force pause	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PauseLocal	1 = Select "Pause" in local mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PauseMan	1 = Select "Pause" in manual mode	BOOL	0
Perm_En	1 = Activation enable (enable, Permit parameter) is active	BOOL	1
Permit	1 = Enable for opening / closing from safe position 0 = Valve activation not enabled on OS	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
Prot_En	1 = Protective interlock (protection, Protect parameter) is active	BOOL	1
Protect	0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block 1 = Good state	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#FF
PulseWidth	Pulse width of control signal [s]	REAL	3.0
PV	Process value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_A_DC	PV: Delay time for incoming alarms [s] when using flow dosing (coarse dosing phase)	REAL	0.0
PV_A_DG	PV: Delay time for outgoing alarms [s] when using flow dosing (coarse dosing phase)	REAL	0.0
PV_A2_DC	PV: Delay time for incoming alarms [s] when using flow dosing (fine dosing phase)	REAL	0.0

Parameter	Description	Type	Default
PV_A2_DG	PV: Delay time for outgoing alarms [s] when using flow dosing (fine dosing phase)	REAL	0.0
PV_AH_En	PV: 1 = Enable high alarm limit when using flow dosing (coarse dosing phase)	BOOL	1
PV_AH_Lim	PV: Limit high alarm when using flow dosing (coarse dosing phase)	REAL	100.0
PV_AH_MsgEn	PV: 1 = Enable high alarm message when using flow dosing (coarse dosing phase)	BOOL	1
PV_AH2_En	PV: 1 = Enable high alarm when using flow dosing (fine dosing phase)	BOOL	1
PV_AH2_Lim	PV: Limit high alarm when using flow dosing (fine dosing phase)	REAL	100.0
PV_AH2_MsgEn	PV: 1 = Enable high alarm message when using flow dosing (fine dosing phase)	BOOL	1
PV_AL_En	PV: 1 = Enable low alarm limit when using flow dosing (coarse dosing phase)	BOOL	1
PV_AL_Lim	PV: Limit low alarm when using flow dosing (coarse dosing phase)	REAL	0.0
PV_AL_MsgEn	PV: 1 = Enable low alarm message when using flow dosing (coarse dosing phase)	BOOL	1
PV_AL2_En	PV: 1 = Enable low alarm when using flow dosing (fine dosing phase)	BOOL	1
PV_AL2_Lim	PV: Limit low alarm when using flow dosing (fine dosing phase)	REAL	0.0
PV_AL2_MsgEn	PV: 1 = Enable low alarm message when using flow dosing (fine dosing phase)	BOOL	1
PV_Hyst	PV: Hysteresis for alarm limits when using flow dosing (coarse dosing phase)	REAL	1.0
PV_Hyst2	PV: Hysteresis for alarm limits when using flow dosing (fine dosing phase)	REAL	1.0
PV_OpScale	PV: Limit for scale in bar graph of faceplate	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
PV_Unit	Unit of measure for process value	INT	1349
RelaxTime	Duration of dribble phase [s]	REAL	3.0
RstDQ_Li	1 = Reset dosing quantity via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RstDQ_Op	1 = Reset dosing quantity via operator	BOOL	0

Parameter	Description	Type	Default
RstLi	1 = Reset interlock/flow monitoring via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RstOp	1 = Reset interlock/flow monitoring via operator	BOOL	0
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SelFp1	Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
SimOn	1 = Simulation on	BOOL	0
SimPV	PV: Value used for SimOn = 1	REAL	0.0
SP_Ext	External flow setpoint - coarse dosing	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_ExtLi	1 = Select external setpoints via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP_ExtOp	1 = Select external setpoints via operator	BOOL	0
SP_HiLim	High limit for flow setpoint - coarse dosing	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
SP_Int	Internal flow setpoint - coarse dosing	REAL	0.0
SP_IntLi	1 = Select internal setpoints via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP_IntOp	1 = Select internal setpoints via operator	BOOL	0
SP_LiOp	Select setpoint source (internal/external): 1 = Via interconnection 0 = Via operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP_LoLim	Low limit for flow setpoint - coarse dosing	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_TrkExt	1 = Bumpless switchover from external to internal setpoint active	BOOL	1

Parameter	Description	Type	Default
SP2_Ext	External flow setpoint - fine dosing	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP2_HiLim	High limit for flow setpoint - fine dosing	STRUCT • Value: REAL • ST: BYTE	- • 100.0 • 16#80
SP2_Int	Internal flow setpoint - fine dosing	REAL	0.0
SP2_LoLim	Low limit for flow setpoint - fine dosing	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
StandStill	1 = Scales at a standstill, dosing device feedback	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
StartAut	1 = Select "Start" in automatic mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StartForce	1 = Force start	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StartLocal	1 = Select "Start" in local mode	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StartMan	1 = Select "Start" in manual mode	BOOL	0
StepNo	Batch step number	DWORD	16#00000000
TI	Integral action time [s]	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#80
U_AckLi	1 = Acknowledgment of underdosage via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
U_AckOp	1 = Acknowledgment of underdosage via operator	BOOL	0
UA1unit	Unit of measure for analog auxiliary value 1	INT	0
UA2unit	Unit of measure for analog auxiliary value 1	INT	0
UserAna1	Analog auxiliary value 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF

Parameter	Description	Type	Default
UserAna2	Analog auxiliary value 2	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
UserStatus	Freely assignable user area for status word (bit 24 to bit 31)	BYTE	16#00

Output parameters

Parameter	Description	Type	Default
AutAct	1 = Automatic mode is active 0 = Manual mode is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
CR_AH_Act	Creep flow: 1 = High alarm active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Ctrl	Control output for coarse dosing	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Ctrl2	Control output for fine dosing	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
DosEnd	1 = End of dosing	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
DosOff	1 = Dosing off	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
DosOn	1 = Dosing on	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
DosPause	1 = Dosing pause	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
DosRelax	1 = Dosing dribble	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
DosStart	1 = Dosing started	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
DQ_AH_Act	AV: 1 = High alarm active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
DQ_AL_Act	AV: 1 = Low alarm active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
DQ_ExHiAct	Dosing quantity: 1 = High limit for external setpoint has been reached	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
DQ_ExLoAct	Dosing quantity: 1 = Low limit for external setpoint has been reached	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
DQ_ExtOut	Dosing quantity: External setpoint, corresponds to input parameter DQ_Ext	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
DQ_Out	Dosing quantity	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
DQ_SP	Dosing quantity setpoint used by block	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
DQ_Tare	Tare memory when dosing using scales	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
DQ1_SP	Coarse dosing quantity setpoint used by block	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
DQ2_ExHiAct	Dosing quantity: 1 = High limit for external setpoint factor for fine dosing has been reached	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
DQ2_ExLoAct	Dosing quantity: 1 = Low limit for external setpoint factor for fine dosing has been reached	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
DQ2_ExtOut	Dosing quantity: External setpoint factor for fine dosing [%], corresponds to input parameter DQ2_Ext	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
DQ2_SP	Fine dosing quantity setpoint used by block	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
DribbOut	Dribble quantity	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see DoseL error handling (Page 1026)	INT	-1
LocalAct	1 = Local mode active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
LockAct	1 = Interlock (Inlock, Permit or Protect) is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ManAct	1 = Manual mode active	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
MS_Release	Maintenance release: 1 = Release for OS operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MsgAckn1	Alarm acknowledgement status 1 (output STATUS of first ALARM_8P)	WORD	16#0000
MsgAckn2	Alarm acknowledgement status 2 (output STATUS of second ALARM_8P)	WORD	16#0000
MsgErr1	Alarm error 1 (output ERROR of first ALARM_8P)	BOOL	0
MsgErr2	Alarm error 2 (output ERROR of second ALARM_8P)	BOOL	0
MsgStat1	Alarm status 1 (output ERROR of first ALARM_8P)	WORD	16#0000
MsgStat2	Alarm status 2 (output ERROR of second ALARM_8P)	WORD	16#0000
OosAct	1 = Block is out of service	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
OS_PermOut	Displays of OS_Perm	DWORD	16#FFFFFFFF
OS1PermLog	Display of OS1Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
OS1PermOut	Display of OS1Perm	DWORD	16#FFFFFFFF
P_Ctrl	Pulse output for coarse dosing	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
P_Ctrl2	Pulse output for fine dosing	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_AH_Act	PV: 1 = High alarm active (coarse dosing phase)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_AH2_Act	PV: 1 = High alarm active (fine dosing phase)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
PV_AL_Act	PV: 1 = Low alarm active (coarse dosing phase)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_AL2_Act	PV: 1 = Low alarm active (fine dosing phase)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_Out	Output for process value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP	Active flow setpoint	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP_ExHiAct	Flow for coarse dosing: 1 = High limit for external setpoint has been reached	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP_ExLoAct	Flow for coarse dosing: 1 = Low limit for external setpoint has been reached	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP_ExtAct	1 = External setpoint is active 0 = Internal setpoint is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP_ExtOut	Flow for coarse dosing: External setpoint, corresponds to input parameter SP_Ext	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP1	Coarse dosing flow setpoint used by block	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP2	Fine dosing flow setpoint used by block	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SP2_ExHiAct	Flow for fine dosing: 1 = High limit for external setpoint has been reached	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP2_ExLoAct	Flow for fine dosing: 1 = Low limit for external setpoint has been reached	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SP2_ExtOut	Flow for fine dosing: External setpoint, corresponds to input parameter SP2_Ext	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word 1 (Page 1009)	DWORD	16#00000000

Parameter	Description	Type	Default
Status2	Status word 2 (Page 1009)	DWORD	16#00000000
Status3	Status word 3 (Page 1009)	DWORD	16#00000000
Status4	Status word 4 (Page 1009)	DWORD	16#00000000

See also

DoseL messaging (Page 1028)

DoseL modes (Page 1013)

DoseL block diagram (Page 1045)

9.1.7 DoseL block diagram

DoseL block diagram

A block diagram is not provided for this block.

See also

Description of DoseL (Page 1009)
DoseL modes (Page 1013)
DoseL functions (Page 1015)
DoseL error handling (Page 1026)
DoseL messaging (Page 1028)
DoseL I/Os (Page 1031)

9.1.8 Operator control and monitoring

9.1.8.1 Views of DoseL

Views of the DoseL block

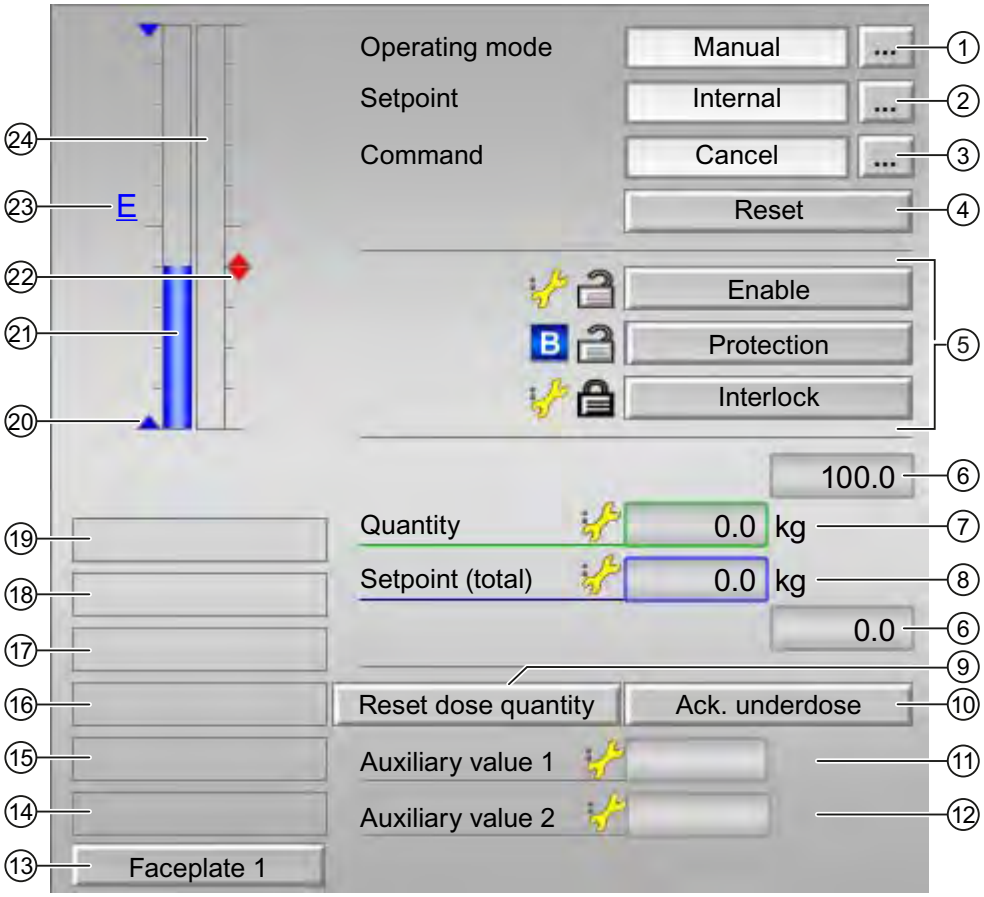
The block DoseL provides the following views:

- DoseL standard view (Page 1046)
- Message view (Page 169)
- DoseL limit value view (Page 1050)
- Trend view (Page 172)
- DoseL parameter view (Page 1052)
- Flow setpoint view of DoseL (Page 1054)
- Quantity setpoint view of DoseL (Page 1056)
- DoseL preview (Page 1058)
- Memo view (Page 171)
- Batch view (Page 170)
- Block symbol for DoseL (Page 1061)

Refer to the sections Structure of the faceplate (Page 123) and Block icon structure (Page 174) for general information on the faceplate and block icon.

9.1.8.2 DoseL standard view

DoseL standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- Manual mode (Page 32)
- Automatic mode (Page 32)
- Local mode (Page 36)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Display and change the setpoint

The following states can be shown and changed here:

- Internal
- External

You can find additional information on this in the Manual and automatic mode for motors, valves and dosers (Page 32) section.

(3) Display and change: Start, continue, pause and cancel

This area shows you the default operating state for the doser. The following states can be shown and changed here:

- Start
- Continue
- Pause
- Cancel

You can find additional information on this in the Switching operating states and operating modes (Page 127) section.

(4) Resetting the block

Click "Reset" for interlocks or errors. You can find additional information on this in the Resetting the block in case of interlocks or errors (Page 119) section.

(5) Display area for interlock functions of the block

You can use this button to control the interlock functions of the block. You can find additional information on this in the Interlocks (Page 100) section.

(6) High and low scale range for the setpoint

This area is already set and cannot be changed.

(7) Display and change the quantity

You can find additional information on this in the Changing values (Page 129) section.

(8) Display and change the setpoint

You can find additional information on this in the Changing values (Page 129) section.

(9) Button: Reset dose quantity

The dosing quantity can only be reset in the "End" status.

(10) Button: Acknowledge underdosing

Acknowledgment of underdosing can only be made in the "Off" status.

(11) and (12) Display of auxiliary values

You can use this area to display two auxiliary values that have been configured in the engineering system. You can find additional information on this in the Calling further faceplates (Page 43) section.

(13) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

(14) and (15) Display area for states of the block

This area provides additional information on the operating state of the block:

- Simulation
- Maintenance

You can find additional information on this in the Simulating signals (Page 93) section.

(16) Display area for states of the block

This area provides additional information on the operating state of the block:

- Invalid signal
- Flow

(17) Display area for states of the block

This area provides additional information on the operating state of the block:

- Force start
- Force resume
- Force pause
- Force stop

(18) Display area for states of the block

This area provides additional information on the operating state of the block:

- Underdosed
- Overdosed

(19) Display area for states of the block

This area provides additional information on the operating state of the block:

- Coarse dosing
- Fine dosing
- Relaxing
- Pause
- Off
- End
- Taring

(20) Limit display for the setpoint

These triangles show the SP_HiLim and SP_LoLim setpoint limits configured in the ES.

(21) Bar graph for the setpoint

This area shows you the current setpoint in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(22) Limit display

These colored triangles indicate the configured limits in the respective bar graph:

- Red: Alarm
- Yellow: Warning
- Blue: Tolerance

(23) Display for external setpoint

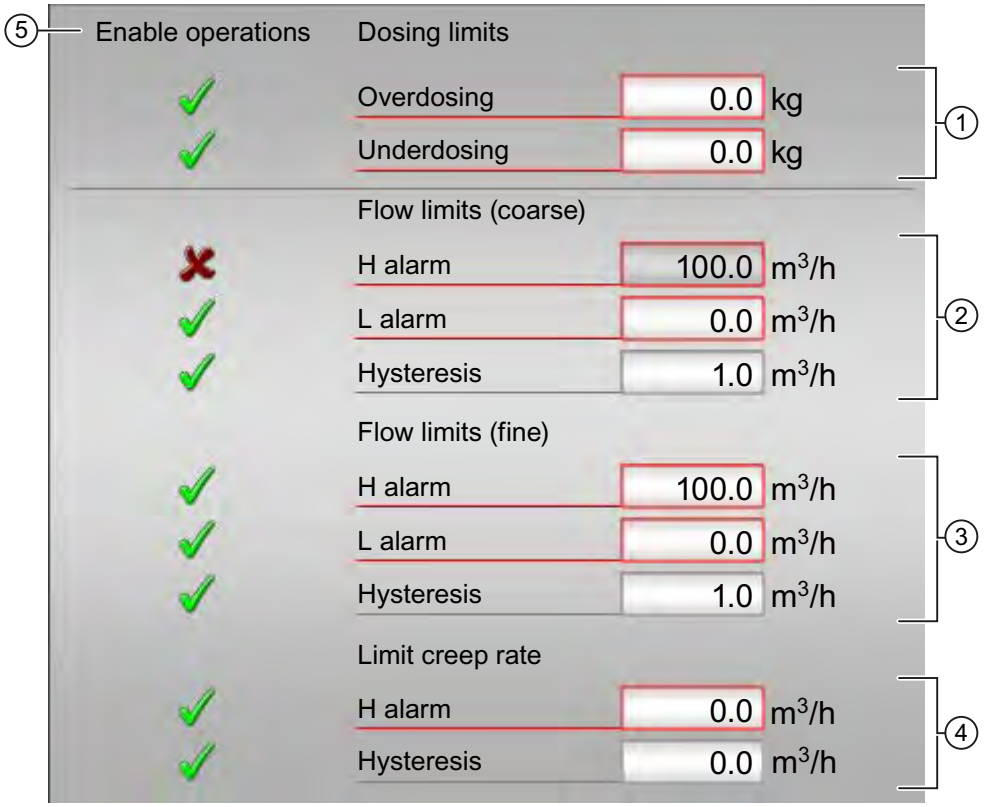
This display [E] is only visible when you have selected "internal" setpoint specification. It shows the external setpoint that would apply if you were to change the setpoint specification to "external".

(24) Bar graph for the quantity

This area shows you the current quantity in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

9.1.8.3 DoseL limit value view

DoseL limit value view



(1) Display and change the dosing gain

You can change the dosing limits in this area:

- Overdosing
- Underdosing

You can find additional information on this in the Changing values (Page 129) section.

(2) Display and change the dosing limit (coarse)

You can change the dosing limits (coarse) in this area:

- H alarm
- L alarm
- Hysteresis

You can find additional information on this in the Changing values (Page 129) section.

(3) Display and change the dosing limit (fine)

You can change the dosing limits (fine) in this area:

- H alarm
- L alarm
- Hysteresis

You can find additional information on this in the Changing values (Page 129) section.

(4) Display and change the limits for the creep rate

You can change the limit for the creep rate in this area:

- H alarm
- Hysteresis

You can find additional information on this in the Changing values (Page 129) section.

(5) Enable operations

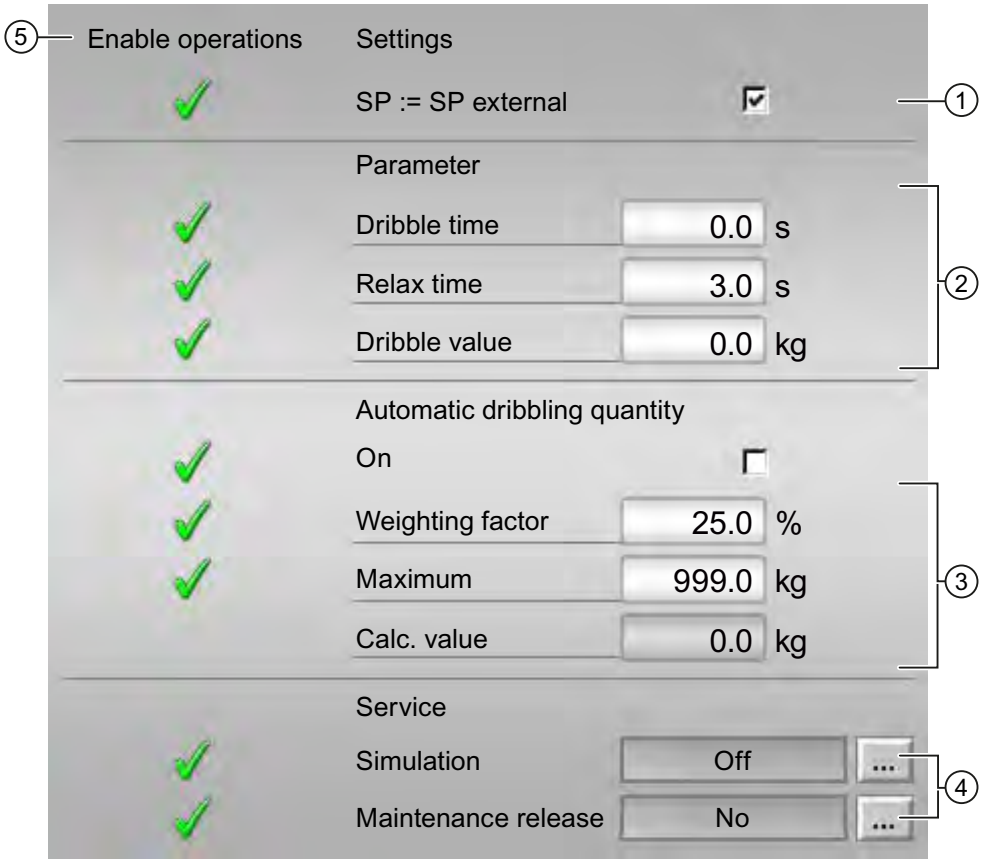
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions

9.1.8.4 DoseL parameter view

DoseL parameter view



(1) Settings

You can select the following functions in this area:

- SP:=SP external: ☑ Bumpless switchover of setpoint from external to internal. The internal setpoint is tracked to the external one.

(2) Parameters

In this area, you change parameters and therefore influence the doser. Refer also to the Changing values (Page 129) section for more on this.

You can influence the following parameters:

- Relax time
- Dribble time
- Dribble value

(3) Automatic dribbling quantity

In this area, you change the automatic dribbling quantity parameters and therefore influence the doser. Refer also to the Changing values (Page 129) section for more on this.

You can change the parameters when the On check box is selected ☑.

You can influence the following parameters:

- Weighting factor
- Maximum
- Calc. value

(4) Service

You can select the following functions in this area:

- Simulation
- Maintenance release

Refer to the Switching operating states and operating modes (Page 127) section for more on this.

You can find information on this area in the section:

- Simulating signals (Page 93)
- Maintenance release (Page 47)

(5) Enable operations

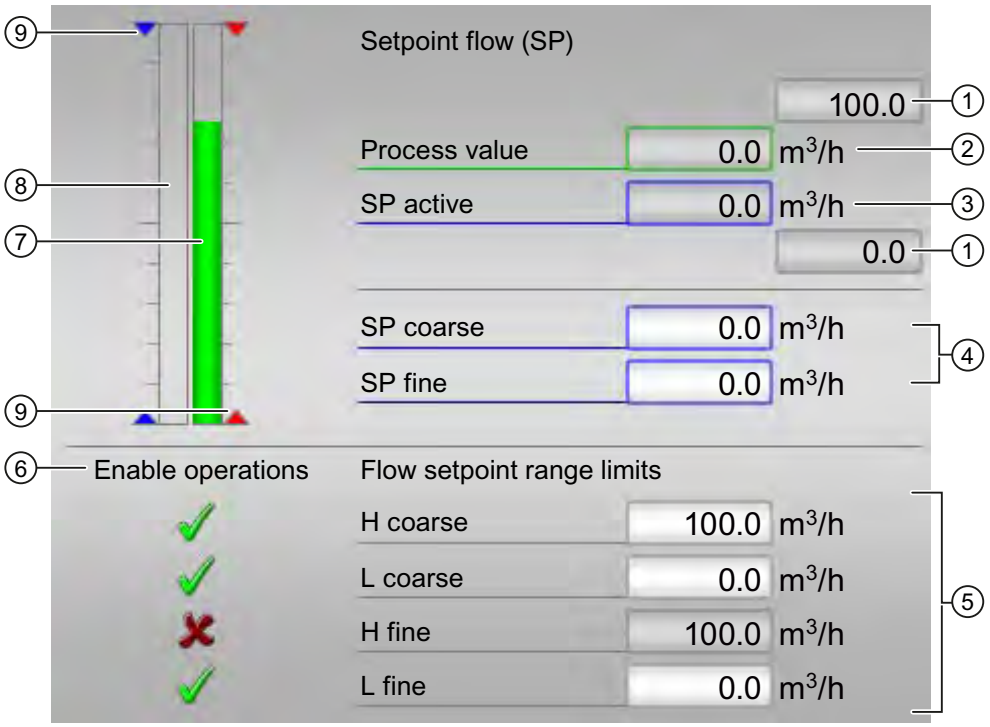
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions.

9.1.8.5 Flow setpoint view of DoseL

Flow setpoint view for DoseL



(1) High and low scale range for the process value

These values provide information on the display range for the bar graph **(7)** of the process value. The scale range is defined in the engineering system.

(2) Display of the process value including signal status

This area shows the current process value with the corresponding signal status.

(3) Display and change the SP active value

You can find additional information on this in the Changing values (Page 129) section.

(4) Additional setpoint parameters

You can change the following setpoint parameters in this area:

- SP coarse
- SP fine
- SP fine factor

You can find additional information on this in the Changing values (Page 129) section.

(5) Display and change the limits

You can change the limits in this area:

- H coarse
- L coarse
- H fine
- L fine

You can find additional information on this in the Changing values (Page 129) section.

(6) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions.

(7) Bar graph for the process value

This area shows the current process value in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(8) Bar graph for the setpoint SP active

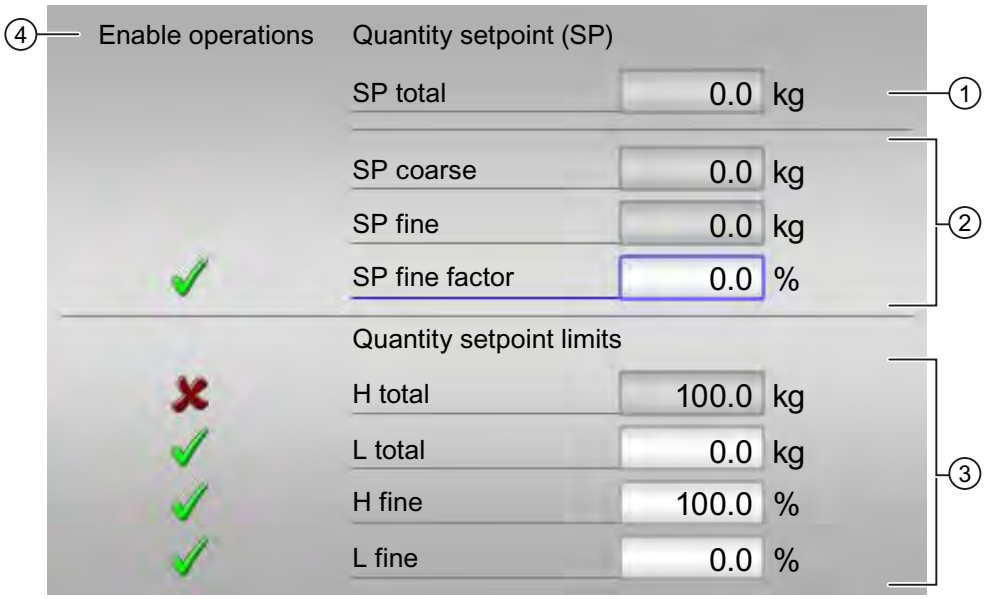
This area shows you the current setpoint SP active in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(9) Limit display for the setpoint

These triangles show the `SP_HiLim` and `SP_LoLim` setpoint limits configured in the ES.

9.1.8.6 Quantity setpoint view of DoseL

Quantity setpoint view for DoseL



(1) Display and change the setpoint SP total

You can change the setpoint SP total in this area:

You can find additional information on this in the Changing values (Page 129) section.

(2) Display and change additional setpoints

You can change the limits in this area:

- H coarse
- L coarse
- H fine
- L fine

You can find additional information on this in the Changing values (Page 129) section.

(3) Display and change the limits

You can change the limits in this area:

- SP coarse
- SP fine
- SP fine factor

You can find additional information on this in the Changing values (Page 129) section.

(4) Enable operations

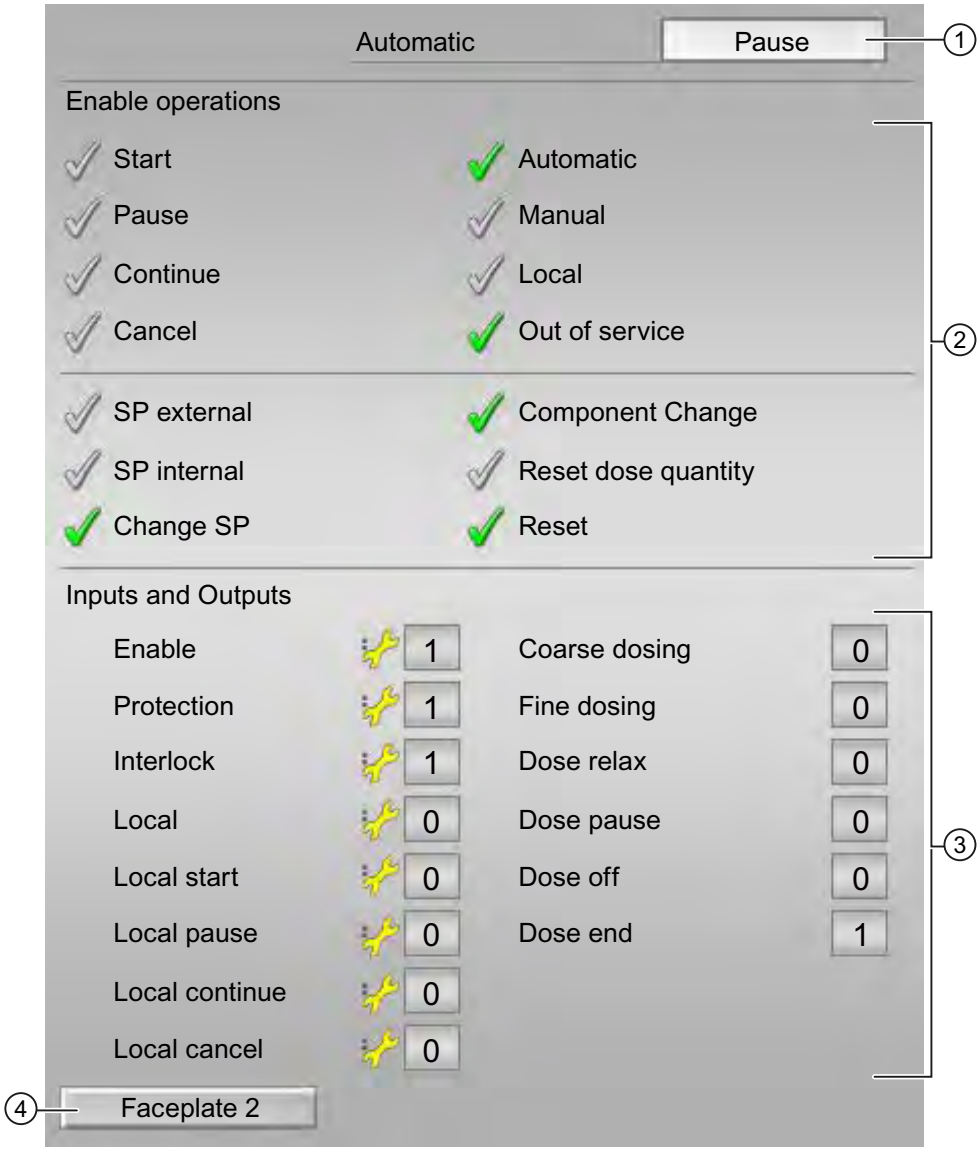
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions.

9.1.8.7 DoseL preview

DoseL preview



(1) Automatic preview

This area shows you the block status after its has switched from manual to automatic mode.

If the block is in automatic mode, the current block state is displayed.

(2) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions

(3) Display current control signal

This area shows the most important parameters for this block with the current selection:

- Enable:
  - 0 = No OS release for energizing motor
  - 1 = Enable for starting / stopping from safe position
- Protection:
  - 0 = Protective interlocking is effective; once the interlocking condition has disappeared, you will have to reset the block
  - 1 = Good state
- Interlock:
  - 0 = Interlocking without reset is effective; you can operate the block without reset once the interlocking condition has disappeared
  - 1 = Good state
- Local: 1= Block is controlled in local mode
- Local start: 1 = Block is started in local mode
- Local pause: 1= Dosing is paused in local mode
- Local continue: 1 = Dosing is continued in local mode
- Local cancel: 1= Dosing is cancelled in local mode
- Coarse dosing: 1 = Coarse dosing is performed
- Fine dosing: 1 = Fine dosing is performed
- Dose relax: 1 = Dosing is in the relax phase
- Dose pause: 1 = Dosing is paused
- Dose off: 1 = No dosing taking place
- Dose end: 1 = Dosing is stopped

(4) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

9.1.8.8 Block symbol for DoseL

Properties of the DoseL block symbol

A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as the control system fault CSF
- Operating modes
- Internal and external setpoint specification
- Signal status, maintenance release
- Track, force, and bypass
- Interlocks
- Memo display
- Process value (black, with and without decimal places)
- Setpoint (blue, with and without decimal places)
- Feedback value (red, with and without decimal places)

Icons	Selection of the block icon in CFC	Special features
	1	- Block icon in full display
	2	
	3	
	4	
	5	

Icons	Selection of the block icon in CFC	Special features
	6	
	7	
	8	
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181).

Mathematical blocks

10

10.1 Add04 - Adder with 4 values

10.1.1 Description of Add04

Object name (type + number) and family

Type + number: FC 300

Family: Math

Area of application for Add04

The block is used for the following applications:

- Adding values
- Output of the total value for further processing

How it works

The Add04 block calculates the sum of up to 4 values:

$V = U_1 + ... + U_n$ ($n \leq 4$),

Where:

V = total (`Out` output)

U_1 ... U_4 = values to add (interconnectable input parameters `In1` to `In4`)

The sum of all input parameters and the worst input parameter signal status is always output.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

For the Add04 block, the Advanced Process Library contains templates for process tag types as examples and there is a example project (APL_Example_xx, xx designates the language variant) containing different application cases for this block. Several application cases are simulated in the example project and serve to explain how the block works.

Examples of process tag types:

- Source chart for GainSched function block (gain scheduling) (Page 1444)
- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 1433)
- PID controller with Smith predictor (SmithPredictorControl) (Page 1436)

Application scenario in the example project:

- Process simulation including noise generator (Page 1456)

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status parameter

This block does not provide the `Status` parameter.

See also

Add04 functions (Page 1065)

Add04 messaging (Page 1066)

Add04 I/Os (Page 1067)

Add04 block diagram (Page 1068)

Add04 error handling (Page 1066)

Add04 modes (Page 1065)

10.1.2 Add04 modes

Add04 modes

This block does not provide any operating modes.

See also

Add04 block diagram (Page 1068)
Add04 I/Os (Page 1067)
Add04 messaging (Page 1066)
Add04 error handling (Page 1066)
Description of Add04 (Page 1063)
Add04 functions (Page 1065)
Out of service (Page 27)

10.1.3 Add04 functions

Functions of Add04

There are no other functions for this block.

See also

Description of Add04 (Page 1063)
Add04 messaging (Page 1066)
Add04 I/Os (Page 1067)
Add04 block diagram (Page 1068)
Add04 error handling (Page 1066)
Add04 modes (Page 1065)
Selecting signals for processing (Page 93)

10.1.4 Add04 error handling

Add04 troubleshooting

The block does not report any errors.

See also

Add04 block diagram (Page 1068)
Add04 I/Os (Page 1067)
Add04 messaging (Page 1066)
Description of Add04 (Page 1063)
Add04 modes (Page 1065)
Add04 functions (Page 1065)
Error handling (Page 117)

10.1.5 Add04 messaging

Message functionality

The block does not have any message functionality.

See also

Description of Add04 (Page 1063)
Add04 functions (Page 1065)
Add04 I/Os (Page 1067)
Add04 block diagram (Page 1068)
Add04 modes (Page 1065)
Add04 error handling (Page 1066)

10.1.6 Add04 I/Os

Add04 I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
In1	Value 1 to add	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In2	Value 2 to add	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In3	Value 3 to add	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In4	Value 4 to add	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
Out	Output of the sum	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

See also

Description of Add04 (Page 1063)
Add04 functions (Page 1065)
Add04 messaging (Page 1066)
Add04 block diagram (Page 1068)
Add04 modes (Page 1065)
Add04 error handling (Page 1066)
Selecting signals for processing (Page 93)

10.1.7 Add04 block diagram

Add04 block diagram

A block diagram is not provided for this block.

See also

Description of Add04 (Page 1063)
Add04 modes (Page 1065)
Add04 functions (Page 1065)
Add04 error handling (Page 1066)
Add04 messaging (Page 1066)
Add04 I/Os (Page 1067)

10.2 Add08 - Adder with 8 values

10.2.1 Description of Add08

Object name (type + number) and family

Type + number: FC 301

Family: Math

Area of application for Add08

The block is used for the following applications:

- Adding values
- Output of the total value for further processing

How it works

The Add08 block calculates the sum of up to 8 values:

$V = U_1 + ... + Un$ ($n \leq 8$),

Where:

V = total (`Out` I/O)

U_1 ... U8 = values to be added (interconnectable input parameters `In1` to `In8`)

The sum of all input parameters and the worst input parameter signal status is always output.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

For the Add04 (Add08) block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Refer to Description of Add04 (Page 1063) for more information.

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status parameter

This block does not provide the `Status` parameter.

See also

Add08 functions (Page 1071)
Add08 messaging (Page 1072)
Add08 I/Os (Page 1073)
Add08 block diagram (Page 1074)
Add08 error handling (Page 1072)
Add08 modes (Page 1071)

10.2.2 Add08 modes

Add08 modes

This block does not provide any modes.

See also

Description of Add08 (Page 1069)
Add08 functions (Page 1071)
Add08 error handling (Page 1072)
Add08 messaging (Page 1072)
Add08 I/Os (Page 1073)
Add08 block diagram (Page 1074)
Out of service (Page 27)

10.2.3 Add08 functions

Functions of Add08

There are no other functions for this block.

See also

Description of Add08 (Page 1069)
Add08 messaging (Page 1072)
Add08 I/Os (Page 1073)
Add08 block diagram (Page 1074)
Add08 error handling (Page 1072)
Add08 modes (Page 1071)
Selecting signals for processing (Page 93)

10.2.4 Add08 error handling

Add08 troubleshooting

The block does not report any errors.

See also

Add08 block diagram (Page 1074)
Add08 I/Os (Page 1073)
Add08 messaging (Page 1072)
Description of Add08 (Page 1069)
Add08 functions (Page 1071)
Add08 modes (Page 1071)
Error handling (Page 117)

10.2.5 Add08 messaging

Message functionality

The block does not have any message functionality.

See also

Description of Add08 (Page 1069)
Add08 functions (Page 1071)
Add08 I/Os (Page 1073)
Add08 error handling (Page 1072)
Add08 modes (Page 1071)
Add08 block diagram (Page 1074)

10.2.6 Add08 I/Os

Add08 I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
In1	Value 1 to add	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In2	Value 2 to add	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In3	Value 3 to add	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In4	Value 4 to add	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In5	Value 5 to add	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In6	Value 6 to add	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In7	Value 7 to add	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In8	Value 8 to add	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
Out	Output of the sum	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

See also

Description of Add08 (Page 1069)
Add08 functions (Page 1071)
Add08 messaging (Page 1072)
Add08 block diagram (Page 1074)
Add08 error handling (Page 1072)
Add08 modes (Page 1071)

10.2.7 Add08 block diagram

Add08 block diagram

A block diagram is not provided for this block.

See also

Description of Add08 (Page 1069)
Add08 functions (Page 1071)
Add08 error handling (Page 1072)
Add08 I/Os (Page 1073)
Add08 modes (Page 1071)
Add08 messaging (Page 1072)

10.3 Average - mean value calculation

10.3.1 Description of Average

Object name (type + number) and family

Type + number: FB1804

Family: Math

Area of application for Average

The block is used for the following applications:

- Calculation of a time-based mean value

How it works

The block calculates the time-based mean value of an analog value `In` based on the time which has expired since its start. The equation below is used:

$$Out = \frac{Out_{Old} \cdot NumCycles + In}{NumCycles + 1}$$

Where:

`In` = input variable

`Out` = present average value

Out_{Old} = average value calculated after start-up

`NumCycles` = Number of cycles for creating the average value from the 0-1 edge transition of the Run input parameter.

The calculation is started by a 0-1 edge at the `Run = 1` input parameter. The `Out` output parameter is overwritten by the `In` input parameter.

The result at output value `Out` is recalculated in the next cycles, and cycle counter `NumCycles` is incremented.

The `NumCycles` cycle counter has the DINT data type and can therefore assume the maximum value of 2147483647. When this value is reached, the `NumCycles` cycle counter is reset to 1. The formation of the mean is then started once again. With a cycle time of 100 ms, this case occurs every 6.8 years.

Calculation is terminated by resetting (1-0 edge) the `Run = 0` input parameter. The last results set at `Out` and `NumCycles` are saved.

The parameter `Feature` can be used to specify the startup characteristics of the block.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

Status word allocation for Status1 parameter

This block does not provide the `Status` parameter.

See also

Average functions (Page 1077)

Average messaging (Page 1079)

Average I/Os (Page 1080)

Average modes (Page 1076)

Average error handling (Page 1078)

Average block diagram (Page 1081)

10.3.2 Average modes

Average modes

This block does not provide any modes.

See also

Description of Average (Page 1075)

Average functions (Page 1077)

Average error handling (Page 1078)

Average messaging (Page 1079)

Average I/Os (Page 1080)

Average block diagram (Page 1081)

10.3.3 Average functions

Functions of Average

The functions for this block are listed below.

Configurable reactions using the Feature I/O

You can find an overview of all behavior patterns which are provided by the `Feature` parameter in section Functions that can be set via the Feature I/O (Page 186). The following functionality is available for this block at the relevant bits:

Bit	Functions
0	Setting the startup response (Page 187)

See also

Description of Average (Page 1075)

Average messaging (Page 1079)

Average I/Os (Page 1080)

Average modes (Page 1076)

Average error handling (Page 1078)

Average block diagram (Page 1081)

10.3.4 Average error handling

Average troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Default value displayed after installation in the block.
0	No active fault
30	The value of `In` can no longer be displayed in the REAL number field. The last valid value is provided at the `Out` output.
31	The value of `Out` can no longer be displayed in the REAL number field.

See also

Average modes (Page 1076)

Average functions (Page 1077)

Description of Average (Page 1075)

Average messaging (Page 1079)

Average I/Os (Page 1080)

Average block diagram (Page 1081)

10.3.5 Average messaging

Messaging

The block does not offer messaging.

See also

Description of Average (Page 1075)
Average functions (Page 1077)
Average I/Os (Page 1080)
Average modes (Page 1076)
Average error handling (Page 1078)
Average block diagram (Page 1081)

10.3.6 Average I/Os

Average I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
Feature	I/O for additional functions (Page 1077)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
In	Analog input value for mean value calculation	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Run	1 = Start time-based mean value calculation	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80

Output parameters

Parameter	Description	Type	Default
EndTime	1 = Time for ending time-based mean value calculation	DT	
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see Average error handling (Page 1078)	INT	-1
NumCycles	Cycle counter	DINT	1
Out	Output for average value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
StartTime	1 = Time for starting time-based mean value calculation	DT	

See also

Description of Average (Page 1075)
Average messaging (Page 1079)
Average modes (Page 1076)
Average block diagram (Page 1081)

10.3.7 Average block diagram

Average block diagram

A block diagram is not provided for this block.

See also

Average modes (Page 1076)
Average functions (Page 1077)
Average error handling (Page 1078)
Average messaging (Page 1079)
Description of Average (Page 1075)
Average I/Os (Page 1080)

10.4 DeadTime - delayed signal output

10.4.1 Description of DeadTime

Object name (type + number) and family

Type + number: FB 1807

Family: Math

Area of application for DeadTime

The block provides the following functions:

- Delay [s] of signal output

How it works

This block delays the output of an input value by a time `DeadTime` [s] selected by the user.

Dead times up to 100 seconds can be realized. If you want to set a longer dead time, you need to connect several dead time blocks in sequence. You can also insert the dead time block in a runtime group with a longer sampling time or scale down the runtime group, but this would lead to loss of input values.

The block operates as follows:

```
Out = In (t-DeadTime)
```

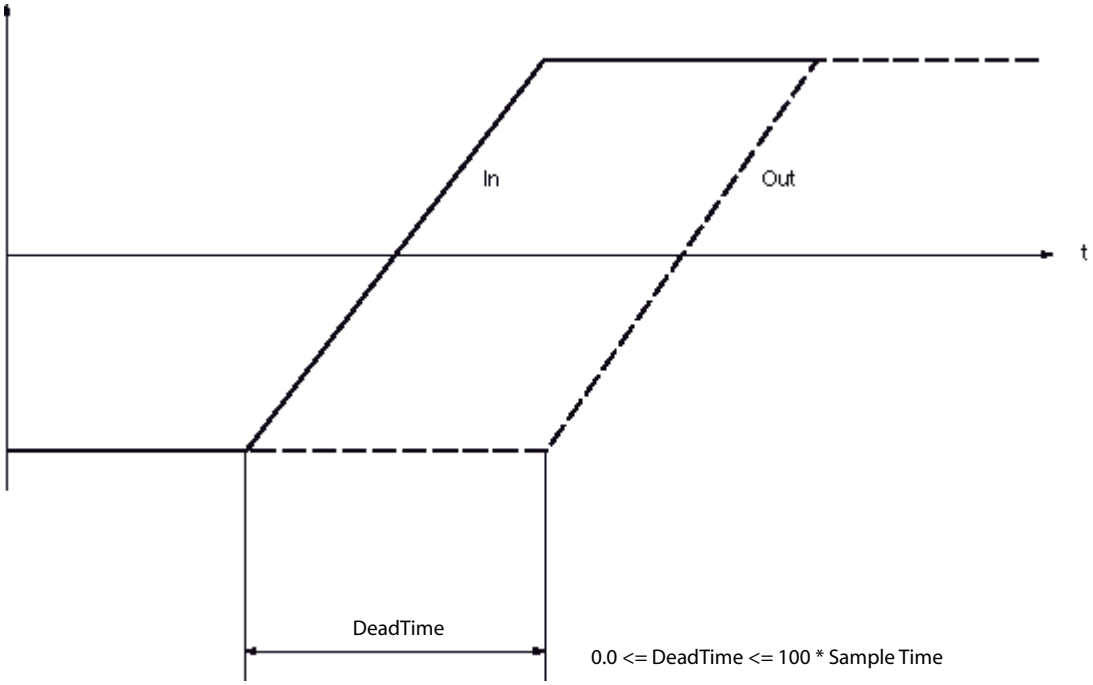
Where:

`Out` = output value

`t` = current time

`DeadT_Cyc` = `DeadTime` / `SampleTimeDeadT_Cyc` can only assume values 0 to 100.

An analog value of input `In` is only output after the variable dead time at `DeadTime` has expired. It is output at `Out`.



Note

If you change the value for the dead time during the runtime of the block, the input value is written to the output. The change to the dead time then takes effect.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The resulting sampling time of the block must be long enough for the desired dead time to be specified at the `DeadTime` parameter.

For the DeadTime block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 1433)
- PID controller with Smith predictor (SmithPredictorControl) (Page 1436)

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

Status word allocation for Status1 parameter

This block does not provide the `Status` parameter.

See also

DeadTime functions (Page 1085)

DeadTime messaging (Page 1087)

DeadTime I/Os (Page 1088)

DeadTime block diagram (Page 1089)

DeadTime modes (Page 1085)

DeadTime error handling (Page 1086)

10.4.2 DeadTime modes

DeadTime operating modes

This block does not provide any operating modes.

See also

DeadTime block diagram (Page 1089)

DeadTime I/Os (Page 1088)

DeadTime messaging (Page 1087)

DeadTime error handling (Page 1086)

DeadTime functions (Page 1085)

Description of DeadTime (Page 1082)

10.4.3 DeadTime functions

Functions of DeadTime

The functions for this block are listed below.

Configurable reactions using the Feature I/O

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Functions
0	Setting the startup response (Page 187)

See also

Description of DeadTime (Page 1082)

DeadTime messaging (Page 1087)

DeadTime I/Os (Page 1088)

DeadTime block diagram (Page 1089)

DeadTime modes (Page 1085)

DeadTime error handling (Page 1086)

10.4.4 DeadTime error handling

DeadTime troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Predefined value when inserting the block, the block is not processed.
0	There is no error.
30	The value of `In` can no longer be displayed in the REAL number field. The last valid value is provided at the `Out` output.
45	The value of `DeadTime` is within the permitted range `0 ≤ INT (DeadTime / SampleTime) ≤ 100` If the value of `DeadTime` is greater than 100 x `SampleTime`, the block works internally with the maximum dead time of 100 x `SampleTime`. If you want to set a longer dead time, you need to connect several dead time blocks in sequence. You can also insert the dead time block in a runtime group with a longer sampling time or scale down the runtime group, but this would lead to loss of input values.

See also

DeadTime block diagram (Page 1089)

DeadTime I/Os (Page 1088)

DeadTime messaging (Page 1087)

DeadTime modes (Page 1085)

Description of DeadTime (Page 1082)

DeadTime functions (Page 1085)

10.4.5 DeadTime messaging

Messaging

The block does not offer messaging.

See also

Description of DeadTime (Page 1082)
DeadTime functions (Page 1085)
DeadTime I/Os (Page 1088)
DeadTime block diagram (Page 1089)
DeadTime modes (Page 1085)
DeadTime error handling (Page 1086)

10.4.6 DeadTime I/Os

DeadTime I/Os

Input parameters

Parameter	Description	Type	Default
DeadTime	Dead time [s]: Time by which the analog input value is delayed	REAL	1.0
EN	1 = Called block will be processed	BOOL	1
Feature	I/O for additional functions (Page 1085)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
In	Analog input value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see DeadTime error handling (Page 1086).	INT	-1
DeadT_Cyc	Number of cycles [0 to 100] by which the analog input value is delayed	INT	0
Out	Analog output value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

See also

Description of DeadTime (Page 1082)

DeadTime messaging (Page 1087)

DeadTime block diagram (Page 1089)

DeadTime modes (Page 1085)

10.4.7 DeadTime block diagram

DeadTime block diagram

A block diagram is not provided for this block.

See also

DeadTime I/Os (Page 1088)
DeadTime messaging (Page 1087)
DeadTime error handling (Page 1086)
DeadTime functions (Page 1085)
DeadTime modes (Page 1085)
Description of DeadTime (Page 1082)

10.5 Derivative - Obtaining a derivative

10.5.1 Description of Derivative

Object name (type + number) and family

Type + number: FB 1808

Family: Math

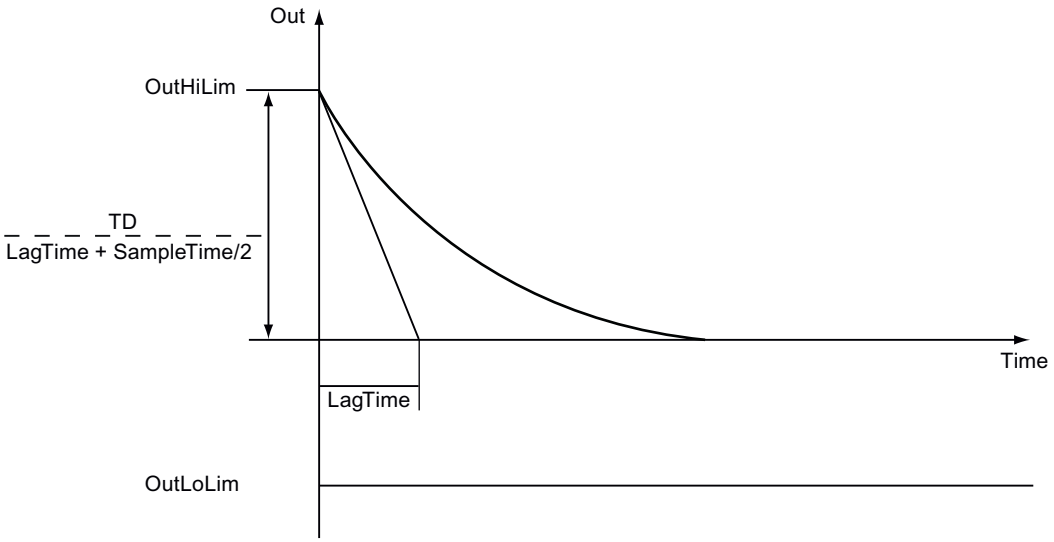
Area of application for Derivative

The block is used for the following applications:

- Obtaining a derivative of the input signal according to the trapezoidal rule

How it works

The block can be used as part of user-designed closed-loop controller. It obtains the derivative of the input signal according to the trapezoidal rule.



The block works according to the following formula:

$Out = TD \cdot (LagTime + SampleTime / 2) \cdot (In - In_{last})$

Where:

$In_{last} = In_{last} + (SampleTime \cdot Out) / TD$

In_{last} = last input value from the I/O In

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB3x). The block is also installed automatically in startup OB100.

Further addressing is not required.

For the Derivative block, the Advanced Process Library contains a template for process tag types as an example with an application scenario for this block.

Example of process tag types:

- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 1433)

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

Status word allocation for `Status` parameter

This block does not provide the `Status` parameter.

See also

Derivative messaging (Page 1094)

Derivative functions (Page 1092)

Derivative I/Os (Page 1095)

Derivative block diagram (Page 1096)

Derivative error handling (Page 1093)

Derivative modes (Page 1091)

10.5.2 Derivative modes

Derivative operating modes

This block does not provide any operating modes.

See also

Derivative block diagram (Page 1096)

Derivative I/Os (Page 1095)

Derivative messaging (Page 1094)

Derivative error handling (Page 1093)

Derivative functions (Page 1092)

Description of Derivative (Page 1090)

10.5.3 Derivative functions

Functions of Derivative

The functions for this block are listed below.

Monitoring the output parameter for limits

The `Out` output parameter can be checked for limit values:

- `OutHiLim`: High limit
- `OutLoLim`: Low limit

If the limits are infringed, this is indicated to you at the corresponding output parameters (output parameter `OutHiAct` or `OutLoAct = 1`).

Configurable reactions using the Feature I/O

You can find an overview of all behavior patterns which are provided by the `Feature` parameter in section Functions that can be set via the Feature I/O (Page 186). The following behavior patterns are available for this block at the relevant bits

Bit	Function
0	Setting the startup response (Page 187)

See also

Description of Derivative (Page 1090)

Derivative messaging (Page 1094)

Derivative I/Os (Page 1095)

Derivative block diagram (Page 1096)

Derivative error handling (Page 1093)

Derivative modes (Page 1091)

10.5.4 Derivative error handling

Derivative troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Default value when implementing the block; block will not be processed.
0	There is no error.
30	The value of `In` can no longer be displayed in the REAL number field. The last valid value is provided at the `Out` output.
31	The value of `Out` can no longer be displayed in the REAL number field.

See also

Derivative block diagram (Page 1096)

Derivative I/Os (Page 1095)

Derivative messaging (Page 1094)

Description of Derivative (Page 1090)

Derivative modes (Page 1091)

Derivative functions (Page 1092)

10.5.5 Derivative messaging

Messaging

This block does not have any message functionality.

See also

Description of Derivative (Page 1090)
Derivative functions (Page 1092)
Derivative I/Os (Page 1095)
Derivative block diagram (Page 1096)
Derivative modes (Page 1091)
Derivative error handling (Page 1093)

10.5.6 Derivative I/Os

Derivative I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
Feature	I/O for additional functions (Page 1092)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
In	Analog input value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
LagTime	Delay time [s]	REAL	10.0
OutHiLim	Limit (high) for output value	REAL	100.0
OutLoLim	Limit (low) for output value	REAL	0.0
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
TD	Derivative time [s]	REAL	1.0

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see Derivative error handling (Page 1093)	INT	-1
Out	Output value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
OutHiAct	1= Limit (high) violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OutLoAct	1= Limit (low) violated	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

Description of Derivative (Page 1090)
Derivative messaging (Page 1094)
Derivative block diagram (Page 1096)
Derivative modes (Page 1091)

10.5.7 Derivative block diagram

Derivative block diagram

A block diagram is not provided for this block.

See also

Description of Derivative (Page 1090)
Derivative modes (Page 1091)
Derivative functions (Page 1092)
Derivative error handling (Page 1093)
Derivative messaging (Page 1094)
Derivative I/Os (Page 1095)

10.6 Div02 - division of two values

10.6.1 Description of Div02

Object name (type + number) and family

Type + number: FC 307

Family: Math

Area of application for Div02

The block is used for the following applications:

- Dividing two values
- Output of the division for further processing

How it works

The block is used to divide two values as follows:

```
Out = In1 / In2
```

If the input parameter is `In2 = 0`, the last valid output value is retained until another division is permitted mathematically.

The worst signal status is also always output at the output parameter.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status1 parameter

This block does not provide the `Status parameter.`

See also

Div02 block diagram (Page 1102)
Div02 I/Os (Page 1101)
Div02 messaging (Page 1100)
Div02 error handling (Page 1100)
Div02 functions (Page 1099)
Div02 modes (Page 1099)

10.6.2 Div02 modes

Div02 operating modes

This block does not provide any operating modes.

See also

Div02 block diagram (Page 1102)
Div02 I/Os (Page 1101)
Div02 messaging (Page 1100)
Div02 error handling (Page 1100)
Div02 functions (Page 1099)
Description of Div02 (Page 1097)

10.6.3 Div02 functions

Functions of Div02

There are no other functions for this block.

See also

Div02 block diagram (Page 1102)
Div02 I/Os (Page 1101)
Div02 messaging (Page 1100)
Div02 error handling (Page 1100)
Div02 modes (Page 1099)
Description of Div02 (Page 1097)
Selecting signals for processing (Page 93)

10.6.4 Div02 error handling

Div02 troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Default value when implementing the block; block will not be processed.
0	There is no error.
30	It has been divided by 0.

See also

Div02 block diagram (Page 1102)

Div02 messaging (Page 1100)

Div02 functions (Page 1099)

Div02 modes (Page 1099)

Description of Div02 (Page 1097)

Div02 I/Os (Page 1101)

10.6.5 Div02 messaging

Message behavior

The block does not have any message behavior.

See also

Div02 block diagram (Page 1102)

Div02 I/Os (Page 1101)

Description of Div02 (Page 1097)

Div02 modes (Page 1099)

Div02 functions (Page 1099)

Div02 error handling (Page 1100)

10.6.6 Div02 I/Os

Div02 I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
In1	Dividend	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In2	Divisor	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#80

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see Div02 error handling (Page 1100).	INT	-1
Out	Output of division	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

See also

Div02 block diagram (Page 1102)
Description of Div02 (Page 1097)
Div02 modes (Page 1099)
Div02 functions (Page 1099)
Div02 messaging (Page 1100)

10.6.7 Div02 block diagram

Div02 block diagram

A block diagram is not provided for this block.

See also

Div02 I/Os (Page 1101)
Div02 messaging (Page 1100)
Div02 error handling (Page 1100)
Div02 functions (Page 1099)
Div02 modes (Page 1099)
Description of Div02 (Page 1097)

10.7 Integral - Generating a time integral

10.7.1 Description of Integral

Object name (type + number) and family

Type + number: FB 1823

Family: Math

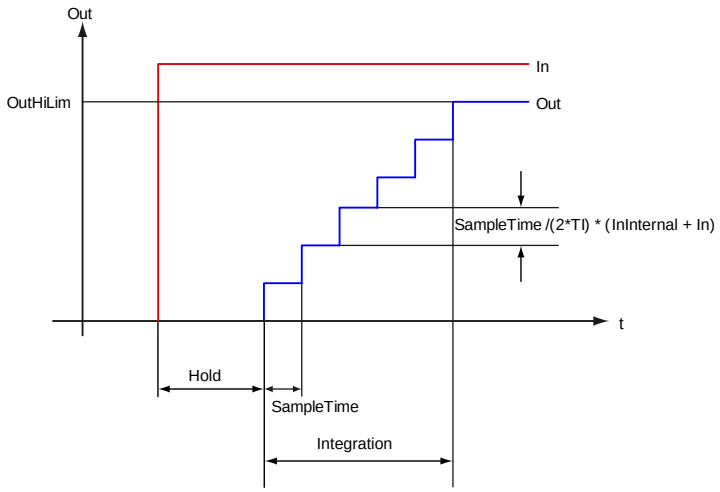
Area of application for Integral

The block is used for the following application:

- Obtaining the time integral of the connected input signal

How it works

The block can be used as part of user-designed closed-loop controller. It integrates the input signal In and outputs the result at Out.



The block works according to the following formula:

$Out = Out_{Internal} + [SampleTime/(2 \cdot TI) \cdot (In_{Internal} + In)]$

Where:

- `Out` = Integrated value between high and low limits
- `SampleTime` = Sampling time [s]
- `TI` = Integral action time [s]
- `In` = Input value
- $In_{Internal}$ = Last input value
- $Out_{Internal}$ = Last output value

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

There is an example project for the Integral block (APL_Example_xx, xx refers to the language variant) with an application scenario for this block, which explains how the block works.

Application scenario in the example project:

- Process simulation including noise generator (Page 1456)

Startup characteristics

Use the feature bit Setting the startup response (Page 187) to define the startup characteristics of this block.

Status word allocation for `Status` parameter

This block does not provide the `Status` parameter.

See also

Integral functions (Page 1106)

Integral messaging (Page 1109)

Integral I/Os (Page 1109)

Integral block diagram (Page 1111)

Integral error handling (Page 1108)

Integral modes (Page 1105)

10.7.2 Integral modes

Integral modes

This block does not provide any modes.

See also

Integral block diagram (Page 1111)
Integral I/Os (Page 1109)
Integral messaging (Page 1109)
Integral error handling (Page 1108)
Integral functions (Page 1106)
Description of Integral (Page 1103)
Out of service (Page 27)

10.7.3 Integral functions

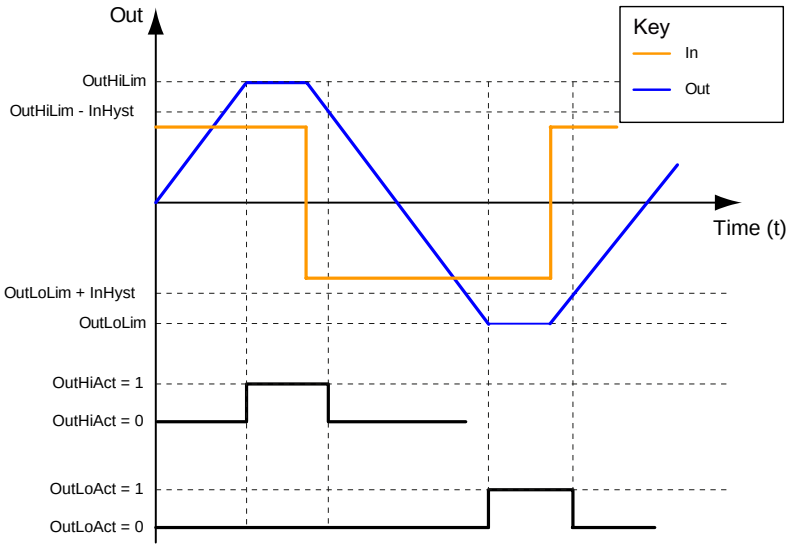
Functions of Integral

The functions for this block are listed below.

Monitoring limits

The `OutHiLim` and `OutLoLim` I/Os can be used to define the limits of the integrated value `Out`.

If limits are reached or exceeded (`OutHiAct` or `OutLoAct = 1` I/Os), the output value is set to user-defined limits and output at `Out`.



Tracking values

You use input parameter `OutTrkOn = 1` to activate tracking of a value, which is in turn defined at input parameter `OutTrk`.

After tracking mode has been terminated, the block uses the value currently set at `Out` as the value to be integrated.

Note

If the integration is stopped, the function (`Hold = 1`) has priority over the tracking.

Stopping integration

Set input parameter `Hold = 1` if you want to stop the integration. The integration is stopped and limits are no longer monitored. If you start the integration again, the value currently present at output parameter `Out` will be used for the integration process.

Configurable reactions using the Feature I/O

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)

See also

Description of Integral (Page 1103)

Integral messaging (Page 1109)

Integral I/Os (Page 1109)

Integral block diagram (Page 1111)

Integral error handling (Page 1108)

Integral modes (Page 1105)

Limit monitoring with hysteresis (Page 78)

10.7.4 Integral error handling

Integral troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Predefined value when inserting the block, the block is not processed.
0	There is no error.
11	An integral action time to be integrated (`TI` I/O) is specified as <= 0 seconds. The block works with a value of 1.0.
15	`Out_Trk` > `OutHiLim` `Out_Trk` < `OutLoLim`
30	The value of `In` can no longer be displayed in the REAL number field. The last valid value is provided at the `Out` output.
31	The value of `Out` can no longer be displayed in the REAL number field.

See also

Integral I/Os (Page 1109)

Integral messaging (Page 1109)

Integral functions (Page 1106)

Integral modes (Page 1105)

Description of Integral (Page 1103)

Integral block diagram (Page 1111)

10.7.5 Integral messaging

Messaging

The block does not offer messaging.

See also

Description of Integral (Page 1103)
Integral functions (Page 1106)
Integral I/Os (Page 1109)
Integral block diagram (Page 1111)
Integral error handling (Page 1108)
Integral modes (Page 1105)

10.7.6 Integral I/Os

Integral I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
Feature	I/O for additional functions (Page 1106)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
Hold	1 = Integration is held	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
In	Analog input value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
InHyst	Hysteresis	REAL	0.0
OutHiLim	High limit of output value	REAL	100.0
OutLoLim	Low limit of output value	REAL	0.0
OutTrkOn	1 = Output Out tracks the set value (OutTrk I/O)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
OutTrk	Predefined value used for OutTrkOn = 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
TI	Integral action time [s]	REAL	1.0

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see Integral error handling (Page 1108)	INT	-1
Out	Integral value output	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
OutHiAct	1 = High limit (OutHiLim) reached	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OutLoAct	1 = Low limit (OutLoLim) reached	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

Description of Integral (Page 1103)
Integral messaging (Page 1109)
Integral block diagram (Page 1111)
Integral modes (Page 1105)

10.7.7 Integral block diagram

Integral block diagram

A block diagram is not provided for this block.

See also

Integral I/Os (Page 1109)
Integral messaging (Page 1109)
Integral functions (Page 1106)
Integral modes (Page 1105)
Description of Integral (Page 1103)
Integral error handling (Page 1108)

10.8 Lag - Low-pass filter

10.8.1 Description of Lag

Object name (type + number) and family

Type + number: FB 1828

Family: Math

Area of application for Lag

The block is used for the following application:

- Smoothing the Input value (low-pass filter)

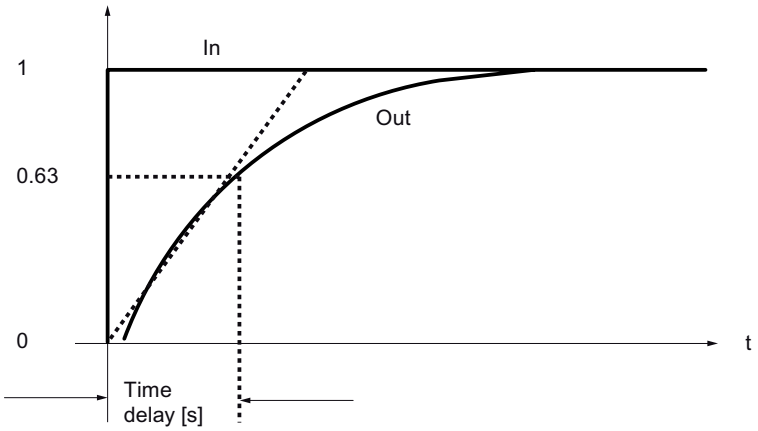
How it works

The block smoothes the input variable (`In` input) using a 1st order delay. This delay time can be selected (`LagTime` connection). The block works according to the following formula:

```
Out = 1 – Exp (- SampleTime/LagTime) · In
```

Where:

- `Out` = Output value
- `LagTime` = Delay time
- `SampleTime` = Sampling time
- `In` = Input value



Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB3x). Further addressing is not required.

For the Lag block, the Advanced Process Library contains templates for process tag types as examples and there is a example project (APL_Example_xx, xx designates the language variant) containing different application cases for this block. Several application cases are simulated in the example project and serve to explain how the block works.

Examples of process tag types:

- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 1433)
- PID - control with operating-point-oriented parameter control (GainScheduling) (Page 1431)
- Model-based predictive control (ModPreCon) (Page 1447)
- PIDConR with safety logic and control loop monitoring (PIDConR_ConPerMon) (Page 1431)
- PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 1430)
- Ratio control (Page 1440)
- Ratio control with PIDConR (RatioR) (Page 1441)
- PID controller with Smith predictor (SmithPredictorControl) (Page 1436)

Application scenario in the example project:

- Process simulation including noise generator (Page 1456)

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

Status word allocation for `Status` parameter

This block does not provide the `Status` parameter.

See also

Lag functions (Page 1115)

Lag messaging (Page 1117)

Lag I/Os (Page 1117)

Lag block diagram (Page 1118)

Lag error handling (Page 1116)

Lag modes (Page 1114)

10.8.2 Lag modes

Lag modes

This block does not provide any modes.

See also

Lag block diagram (Page 1118)
Lag I/Os (Page 1117)
Lag messaging (Page 1117)
Lag error handling (Page 1116)
Lag functions (Page 1115)
Description of Lag (Page 1112)
Out of service (Page 27)

10.8.3 Lag functions

Functions of Lag

The functions for this block are listed below.

Hold and restart calculation

You can interrupt the calculation by setting `Hold = 1`. The output value is retained for the duration of this period. Resume calculation by setting `Hold=0`. Calculation restarts with the last signal value output.

Reset values

You need to set the `Reset = 1` I/O if you want to reset the output value back to the input value. The output is reset by a positive 0 to 1 edge.

Configurable reactions using the Feature I/O

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)

See also

Description of Lag (Page 1112)

Lag messaging (Page 1117)

Lag I/Os (Page 1117)

Lag block diagram (Page 1118)

Lag error handling (Page 1116)

Lag modes (Page 1114)

10.8.4 Lag error handling

Lag troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Default value when implementing the block; block will not be processed.
0	There is no error.
11	`LagTime < 0`
30	The value of `In` can no longer be displayed in the REAL number field. The last valid value is provided at the `Out` output.

See also

Lag block diagram (Page 1118)

Lag I/Os (Page 1117)

Lag messaging (Page 1117)

Lag functions (Page 1115)

Lag modes (Page 1114)

Description of Lag (Page 1112)

10.8.5 Lag messaging

Messaging

The block does not offer messaging.

See also

Description of Lag (Page 1112)
Lag functions (Page 1115)
Lag I/Os (Page 1117)
Lag block diagram (Page 1118)
Lag modes (Page 1114)
Lag error handling (Page 1116)

10.8.6 Lag I/Os

Lag I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
Feature	I/O for additional functions (Page 1115)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
Hold	1 = Hold calculation	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
In	Analog input value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
LagTime	Delay time [s]	REAL	1.0
Reset	1 = Reset Out output to the value of the In input	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see Lag error handling (Page 1116)	INT	-1
Out	Delayed value output	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

See also

Description of Lag (Page 1112)

Lag messaging (Page 1117)

Lag block diagram (Page 1118)

Lag modes (Page 1114)

10.8.7 Lag block diagram

Lag block diagram

A block diagram is not provided for this block.

See also

Lag I/Os (Page 1117)

Lag messaging (Page 1117)

Lag error handling (Page 1116)

Lag functions (Page 1115)

Lag modes (Page 1114)

Description of Lag (Page 1112)

10.9 MeanTime - Averaging

10.9.1 Description of MeanTime

Object name (type + number) and family

Type + number: FB 1832

Family: Math

Area of application for MeanTime

The block is used for the following applications:

- Averaging an analog value over a previous, definable period

How it works

The MeanTime block is used to calculate the time-based mean value of an analog input signal `In` over a previous, definable period (`TimeWindow` input) according to the formula:

`Out` = (In_1 + In_2 +...+In_n) / (`TimeWindow` / `SampleTime`)

Where:

- In_1...In_n are the detected values to be averaged.
- The time window for the averaging is set at the `TimeWindow` parameter.
- The block determines the number of values to be saved based on the integer part of the `TimeWindow` / `SampleTime` quotient.
- The block can save up to 32 previous values in its internal memory. Data is reduced it the time window is longer.

If `SampleTime` or `TimeWindow` is changed, the mean time value is reset.

Configuration

Open the CFC editor to install the block in a cyclic interrupt OB (all OB3x blocks). The block is also installed automatically in the startup OB (OB 100).

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

Status word allocation for `Status` parameter

This block does not provide the `Status` parameter.

See also

MeanTime functions (Page 1121)

MeanTime messaging (Page 1123)

MeanTime I/Os (Page 1123)

MeanTime block diagram (Page 1124)

MeanTime error handling (Page 1122)

MeanTime modes (Page 1120)

10.9.2 MeanTime modes

MeanTime operating modes

This block does not provide any operating modes.

See also

MeanTime block diagram (Page 1124)

MeanTime I/Os (Page 1123)

MeanTime messaging (Page 1123)

MeanTime error handling (Page 1122)

MeanTime functions (Page 1121)

Description of MeanTime (Page 1119)

10.9.3 MeanTime functions

Functions of MeanTime

The functions for this block are listed below.

Stopping the caclulation of mean values

Mean value calculation can be stopped by setting the `Hold` input. To do this, make the following parameter settings:

- `Hold = 1` and calculation is stopped. The output value is retained for the duration of this period.
- `Hold = 0` and calculation is resumed.

Setting a mean value constant

You can set the mean value using the `Reset` input. To do this, make the following parameter settings:

`Reset = 1`. The value at the `In` input is now set directly at the `Out` output. All internal values of the block are aslo adapted to the input value.

Restart averaging by setting 0 at `Reset`.

Configurable reactions using the Features I/O

You can find an overview of all behavior patterns which are provided by the `Feature` parameter in section Functions that can be set via the Feature I/O (Page 186). The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)

See also

Description of MeanTime (Page 1119)

MeanTime messaging (Page 1123)

MeanTime I/Os (Page 1123)

MeanTime block diagram (Page 1124)

MeanTime error handling (Page 1122)

MeanTime modes (Page 1120)

10.9.4 MeanTime error handling

MeanTime troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning
-1	Predefined value when inserting the block, the block is not processed.
0	There is no error.
30	The value of `In` can no longer be displayed in the REAL number field. The last valid value is provided at the `Out` output.
31	The value of Out can no longer be displayed in the REAL number field. The `In` value is output unmodified at the `Out` output.

See also

MeanTime block diagram (Page 1124)

MeanTime I/Os (Page 1123)

MeanTime messaging (Page 1123)

Description of MeanTime (Page 1119)

MeanTime modes (Page 1120)

MeanTime functions (Page 1121)

10.9.5 MeanTime messaging

Messaging

The block does not have any message functionality.

See also

Description of MeanTime (Page 1119)
MeanTime functions (Page 1121)
MeanTime I/Os (Page 1123)
MeanTime block diagram (Page 1124)
MeanTime modes (Page 1120)
MeanTime error handling (Page 1122)

10.9.6 MeanTime I/Os

MeanTime I/Os

Input parameters

Parameter	Description	Type	Default
`EN`	1 = Called block will be processed	BOOL	1
`Feature`	I/O for additional functions (Page 1121)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
`Hold`	1 = Hold calculation of mean value	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
`In`	Analog input value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
`Reset`	1 = Reset output `Out`	STRUCT • Value: BOOL • ST: BYTE	- • 0.0 • 16#80
`SampleTime`	Sampling time [s] (assigned automatically)	REAL	0.1
`TimeWindow`	Size of the time window [s]	REAL	32.0

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see MeanTime error handling (Page 1122)	INT	-1
Out	Output for the average time	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

See also

Description of MeanTime (Page 1119)

MeanTime messaging (Page 1123)

MeanTime block diagram (Page 1124)

MeanTime modes (Page 1120)

10.9.7 MeanTime block diagram

MeanTime block diagram

This block is not provided a block diagram.

See also

Description of MeanTime (Page 1119)

MeanTime modes (Page 1120)

MeanTime functions (Page 1121)

MeanTime error handling (Page 1122)

MeanTime messaging (Page 1123)

MeanTime I/Os (Page 1123)

10.10 Mul04 - Multiplier with 4 values

10.10.1 Description of Mul04

Object name (type + number) and family

Type + number: FC 309

Family: Math

Area of application for Mul04

The block is used for the following applications:

- Multiplication of values
- Output of the product for further processing

How it works

The Mul04 block calculates the product of up to four values and returns the result at the Out output.

$V = U_1 \cdot ... \cdot U_n$ ($n \leq 4$),

Where:

V = product (`Out` output)

U_1 ... U_4 = values to multiply (interconnectable input parameters `In1` to `In4`)

The product of all input parameters and the worst input parameter signal status is always output.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

For the Mul04 block, the Advanced Process Library contains templates for process tag types as examples and there is a example project (APL_Example_xx, xx designates the language variant) containing an application case for this block. An application case is simulated in the example project and serves to explain how the block works.

Examples of process tag types:

- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 1433)
- PID - control with operating-point-oriented parameter control (GainScheduling) (Page 1431)
- Model-based predictive control (ModPreCon) (Page 1447)
- PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 1430)
- PIDConR with safety logic and control loop monitoring (PIDConR_ConPerMon) (Page 1431)

- Ratio control with PIDConR (RatioR) (Page 1441)
- Ratio control (Page 1440)
- PID controller with Smith predictor (SmithPredictorControl) (Page 1436)

Application scenario in the example project:

- Process simulation including noise generator (Page 1456)

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status1 parameter

This block does not provide the `Status` parameter.

See also

Mul04 functions (Page 1127)
Mul04 messaging (Page 1128)
Mul04 I/Os (Page 1128)
Mul04 block diagram (Page 1129)
Mul04 error handling (Page 1127)
Mul04 modes (Page 1126)

10.10.2 Mul04 modes

Mul04 modes

This block does not provide any operating modes.

See also

Mul04 block diagram (Page 1129)
Mul04 I/Os (Page 1128)
Mul04 messaging (Page 1128)
Mul04 error handling (Page 1127)
Mul04 functions (Page 1127)
Description of Mul04 (Page 1125)
Out of service (Page 27)

10.10.3 Mul04 functions

Functions of Mul04

There are no other functions for this block.

See also

Description of Mul04 (Page 1125)
Mul04 messaging (Page 1128)
Mul04 I/Os (Page 1128)
Mul04 error handling (Page 1127)
Mul04 block diagram (Page 1129)
Mul04 modes (Page 1126)
Selecting signals for processing (Page 93)

10.10.4 Mul04 error handling

Mul04 error handling

The block does not report any errors.

See also

Mul04 functions (Page 1127)
Mul04 block diagram (Page 1129)
Mul04 I/Os (Page 1128)
Mul04 messaging (Page 1128)
Description of Mul04 (Page 1125)
Mul04 modes (Page 1126)

10.10.5 Mul04 messaging

Messaging

The block does not have any message functionality.

See also

Description of Mul04 (Page 1125)
Mul04 functions (Page 1127)
Mul04 I/Os (Page 1128)
Mul04 block diagram (Page 1129)
Mul04 modes (Page 1126)
Mul04 error handling (Page 1127)

10.10.6 Mul04 I/Os

Mul04 I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
In1	Value 1 to multiply	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#80
In2	Value 2 to multiply	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#80
In3	Value 3 to multiply	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#80
In4	Value 4 to multiply	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#80

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
Out	Product version	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

See also

Description of Mul04 (Page 1125)
Mul04 functions (Page 1127)
Mul04 messaging (Page 1128)
Mul04 block diagram (Page 1129)
Mul04 modes (Page 1126)
Mul04 error handling (Page 1127)

10.10.7 Mul04 block diagram

Mul04 block diagram

A block diagram is not provided for this block.

See also

Description of Mul04 (Page 1125)
Mul04 modes (Page 1126)
Mul04 functions (Page 1127)
Mul04 error handling (Page 1127)
Mul04 messaging (Page 1128)
Mul04 I/Os (Page 1128)

10.11 Mul08 - Multiplier with 8 values

10.11.1 Description of Mul08

Object name (type + number) and family

Type + number: FC 310

Family: Math

Area of application for Mul08

The block is used for the following applications:

- Multiplication of values
- Output of the product for further processing

How it works

The Mul08 block calculates the product of up to 8 values and returns the result at the Out output.

$V = U_1 \cdot ... \cdot U_n$ ($n \leq 8$),

Where:

V = product (`Out` output)

U_1 ... U_8 = values to multiply (interconnectable input parameters `In1` to `In8`)

The product of all input parameters and the worst input parameter signal status is always output.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

For the Mul04 (Mul08) block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Refer to Description of Mul04 (Page 1125) for more information.

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status1 parameter

This block does not provide the `Status` parameter.

See also

Mul08 functions (Page 1132)

Mul08 messaging (Page 1133)

Mul08 I/Os (Page 1133)

Mul08 block diagram (Page 1135)

Mul08 error handling (Page 1132)

Mul08 modes (Page 1131)

10.11.2 Mul08 modes

Mul08 modes

This block does not provide any modes.

See also

Mul08 block diagram (Page 1135)

Mul08 I/Os (Page 1133)

Mul08 messaging (Page 1133)

Mul08 error handling (Page 1132)

Description of Mul08 (Page 1130)

Mul08 functions (Page 1132)

Out of service (Page 27)

10.11.3 Mul08 functions

Functions of Mul08

There are no other functions for this block.

See also

Description of Mul08 (Page 1130)
Mul08 messaging (Page 1133)
Mul08 I/Os (Page 1133)
Mul08 block diagram (Page 1135)
Mul08 error handling (Page 1132)
Mul08 modes (Page 1131)
Selecting signals for processing (Page 93)

10.11.4 Mul08 error handling

Mul08 error handling

The block does not report any errors.

See also

Mul08 block diagram (Page 1135)
Mul08 I/Os (Page 1133)
Mul08 messaging (Page 1133)
Mul08 modes (Page 1131)
Mul08 functions (Page 1132)
Description of Mul08 (Page 1130)

10.11.5 Mul08 messaging

Message functionality

The block does not have any message functionality.

See also

Description of Mul08 (Page 1130)
Mul08 functions (Page 1132)
Mul08 I/Os (Page 1133)
Mul08 block diagram (Page 1135)
Mul08 modes (Page 1131)
Mul08 error handling (Page 1132)

10.11.6 Mul08 I/Os

Mul08 I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
In1	Value 1 to multiply	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#80
In2	Value 2 to multiply	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#80
In3	Value 3 to multiply	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#80
In4	Value 4 to multiply	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#80
In5	Value 5 to multiply	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#80

Parameter	Description	Type	Default
In6	Value 6 to multiply	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#80
In7	Value 7 to multiply	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#80
In8	Value 8 to multiply	STRUCT • Value: REAL • ST: BYTE	- • 1.0 • 16#80

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
Out	Product output	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

See also

Description of Mul08 (Page 1130)
Mul08 functions (Page 1132)
Mul08 messaging (Page 1133)
Mul08 block diagram (Page 1135)
Mul08 modes (Page 1131)
Mul08 error handling (Page 1132)

10.11.7 Mul08 block diagram

Mul08 block diagram

A block diagram is not provided for this block.

See also

Mul08 I/Os (Page 1133)
Mul08 messaging (Page 1133)
Mul08 error handling (Page 1132)
Mul08 functions (Page 1132)
Mul08 modes (Page 1131)
Description of Mul08 (Page 1130)

10.12 Polygon - Converting the first signal (non-linear)

10.12.1 Description of Polygon

Object name (type + number) and family

Type + number: FB 1881

Family: Math

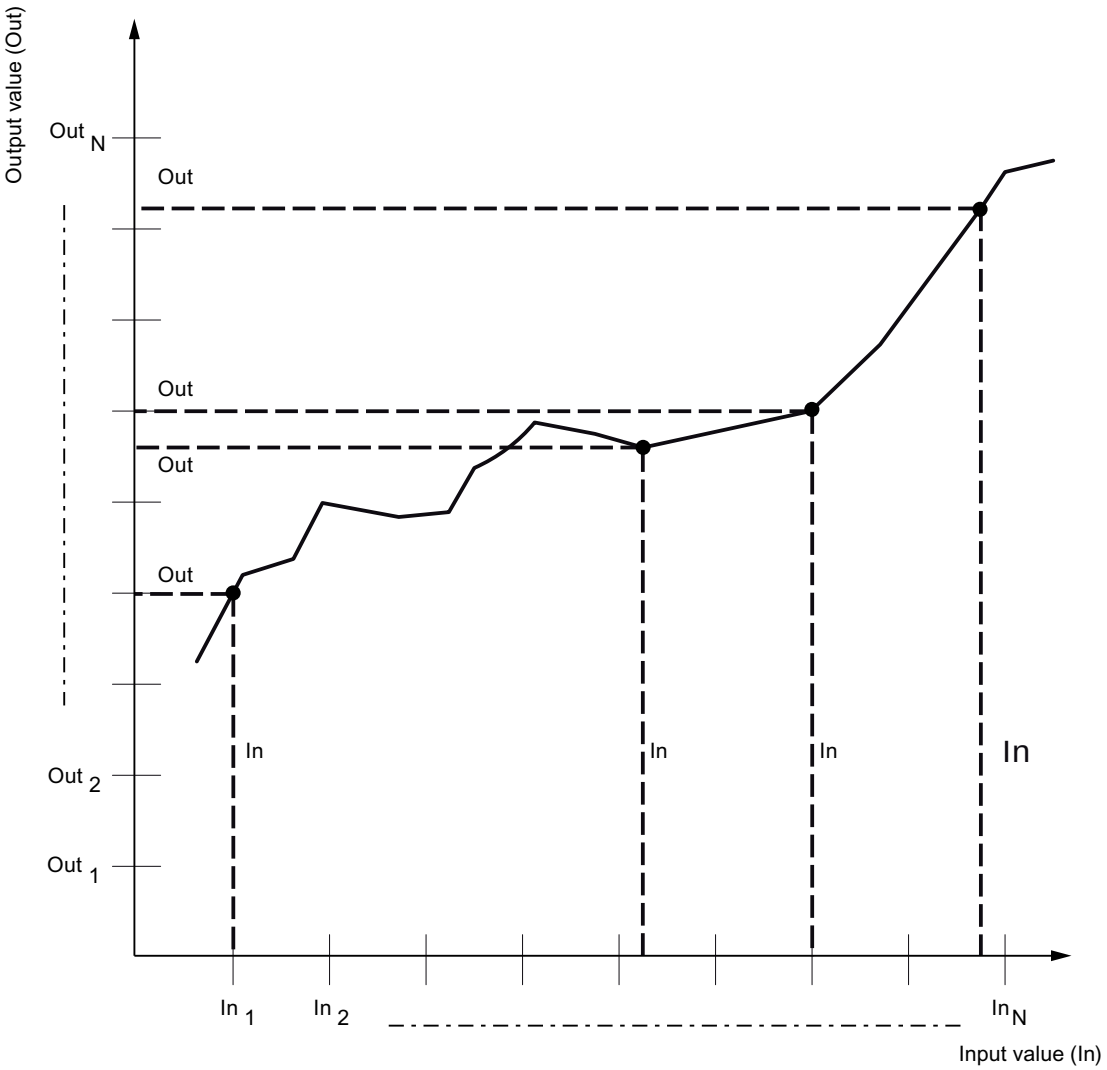
Area of application for Polygon

The block is used for the following applications:

- Conversion of the input signal according to a non-linear characteristic

How it works

An input `In` converted to output `Out` based on a non-linear characteristic with up to 16 interpolation points.



Num (N)	- Number of intermediate points on the curve Minimum number of points = 2 Maximum number of points = 16
In	- Analog input value
Out	- Corresponding output value

The block operates as described below after you have defined the N interpolation points (coordinate pairs In_i, Out_i with `i = 1...N` in a sequence with no gaps) and set the number N:

- Interpolation between the interpolation points is linear
- Outside the end interpolation points values are extrapolated based on the first two or last two interpolation points.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB3x).

For the Polygon block, the Advanced Process Library contains templates for process tag types as an example with various application scenarios for this block.

Example of process tag types:

- Source chart for GainSched function block (gain scheduling) (Page 1444)

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for `Status` parameter

This block does not provide the `Status` parameter.

See also

Polygon functions (Page 1139)
Polygon messaging (Page 1140)
Polygon I/Os (Page 1141)
Polygon block diagram (Page 1144)
Polygon error handling (Page 1140)
Polygon modes (Page 1138)

10.12.2 Polygon modes

Polygon modes

This block does not provide any modes.

See also

Polygon block diagram (Page 1144)
Polygon I/Os (Page 1141)
Polygon messaging (Page 1140)
Polygon error handling (Page 1140)
Polygon functions (Page 1139)
Description of Polygon (Page 1136)
Out of service (Page 27)

10.12.3 Polygon functions

Functions of Polygon

There are no other functions for this block.

See also

Description of Polygon (Page 1136)
Polygon messaging (Page 1140)
Polygon I/Os (Page 1141)
Polygon block diagram (Page 1144)
Polygon error handling (Page 1140)
Polygon modes (Page 1138)
Error handling (Page 117)

10.12.4 Polygon error handling

Polygon troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Default value when the block is installed; this message is irrelevant
0	There is no error.
49	`Num < 2` or `Num> 16`, in the event of an error, the last valid value is output $In_{(N)} \leq In_{(N-1)}$, N = 2 ... 16

See also

Polygon block diagram (Page 1144)

Polygon I/Os (Page 1141)

Polygon messaging (Page 1140)

Description of Polygon (Page 1136)

Polygon modes (Page 1138)

Polygon functions (Page 1139)

10.12.5 Polygon messaging

Messaging

The block does not offer messaging.

See also

Description of Polygon (Page 1136)

Polygon functions (Page 1139)

Polygon I/Os (Page 1141)

Polygon block diagram (Page 1144)

Polygon error handling (Page 1140)

Polygon modes (Page 1138)

10.12.6 Polygon I/Os

Polygon I/Os

I/Os of input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
Feature	I/O for additional functions (Page 1139)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
In	Analog input value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In1	Input In1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In2	Input In2	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In3	Input In3	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In4	Input In4	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In5	Input In5	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In6	Input In6	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In7	Input In7	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In8	Input In8	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

Parameter	Description	Type	Default
In9	Input In9	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In10	Input In10	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In11	Input In11	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In12	Input In12	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In13	Input In13	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In14	Input In14	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In15	Input In15	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In16	Input In16	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Num	Out value interpolation point	INT	16
Out1	Interpolation point 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Out2	Interpolation point 2	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Out3	Interpolation point 3	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Out4	Interpolation point 4	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Out5	Interpolation point 5	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

Parameter	Description	Type	Default
Out6	Interpolation point 6	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Out7	Interpolation point 7	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Out8	Interpolation point 8	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Out9	Interpolation point 9	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Out10	Interpolation point 10	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Out11	Interpolation point 11	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Out12	Interpolation point 12	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Out13	Interpolation point 13	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Out14	Interpolation point 14	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Out15	Interpolation point 15	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
Out16	Interpolation point 16	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see Polygon error handling (Page 1140)	INT	-1
Out	Interpolation value output	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

See also

Description of Polygon (Page 1136)

Polygon messaging (Page 1140)

Polygon block diagram (Page 1144)

Polygon modes (Page 1138)

10.12.7 Polygon block diagram

Polygon block diagram

A block diagram is not provided for this block.

See also

Polygon I/Os (Page 1141)

Polygon messaging (Page 1140)

Polygon error handling (Page 1140)

Polygon functions (Page 1139)

Polygon modes (Page 1138)

Description of Polygon (Page 1136)

10.13 Smooth - low pass filter

10.13.1 Description of Smooth

Object name (type + number) and family

Type + number: FB 1890

Family: Math

Area of application for Smooth

The block is used for the following applications:

- Butterworth low pass filter, 2nd order with maverick detection

How it works

The block is used as a low pass filter. This filter allows the signal portions with frequencies below the cutoff frequency to pass practically unattenuated, whereas portions with high frequencies are attenuated. The enables you filter out high frequency interference in the signal (for example, signal noise) and smooth the signal.

In comparison to a first order low pass filter, the Butterworth filter has the advantage that the transition from the passing section to the blocking section is sharper in the Bode diagram. If the frequency area of the interference is known, it can be filtered out with minimal influence on the wanted signal.

The maverick (Page 1468) detection monitors adjacent signals. If signal mavericks are detected, they are not processed any further. The block outputs the last valid signal.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

You then set the filter time constant for the low pass filter to achieve the desired effect.

To set the filter time constant, it is helpful to observe the original and filtered signals in the CFC trend plotter. The filtered signal should be smoothed as required but not too delayed. An increase in the filter time constant increases the smoothing effect but also increases the delay. Typical starting values for the time constant are around ten times the sample rate of the signal.

There is an example project for the `Smooth` block (APL_Example_xx, xx refers to the language variant) with an application case for this block:

- Filtering of noisy measured values in a control loop (Page 1463)

Startup characteristics

When OB100 is called, the state variables of the Butterworth filter are initialized based on the current process values.

Status word allocation for Status1 parameter

The block does not have a status word allocation.

See also

Smooth functions (Page 1147)
Smooth messaging (Page 1149)
Smooth I/Os (Page 1149)
Smooth block diagram (Page 1150)
Smooth error handling (Page 1148)
Smooth modes (Page 1146)

10.13.2 Smooth modes

Smooth modes

This block does not provide any modes.

See also

Smooth block diagram (Page 1150)
Smooth I/Os (Page 1149)
Smooth messaging (Page 1149)
Smooth error handling (Page 1148)
Smooth functions (Page 1147)
Description of Smooth (Page 1145)
Out of service (Page 27)
On (Page 27)

10.13.3 Smooth functions

Smooth functions

The block provides the following functions:

- Restart low pass filter
- Activate and deactivate maverick detection

Restart low pass filter

You can recalculated the coefficients of the Butterworth filter. To do this, you must restart filter (`Restart = 1`).

The filter algorithm is then reinitialized, exactly as it is when the CPU is restarted or a change is made to the count value at the input parameter `TimeConstant`. The coefficients of the Butterworth filter are recalculated and the internal state memory of the filter is initialized so that the `CleanPV` output parameter is equal to the `PV` input parameter.

Activate and deactivate maverick detection

The maverick detection (`OutlDetOn = 1`) monitors the process value `PV` for the difference between two sequential sample values. These have to be within a tolerance range you have specified (`OutlTreshold`).

If the tolerance is violated, the maverick is set to the most recent valid value. The most recent valid measured value is then held constant by the block for a maximum of `OutlCycles` sample steps. If the maverick continues longer than this, it is accepted as a valid measured value.

See also

Description of Smooth (Page 1145)

Smooth messaging (Page 1149)

Smooth I/Os (Page 1149)

Smooth block diagram (Page 1150)

Smooth error handling (Page 1148)

Smooth modes (Page 1146)

10.13.4 Smooth error handling

Smooth troubleshooting

Refer to the Error handling (Page 117) section in the basic instructions to learn how to troubleshoot all blocks.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Predefined value when inserting the block, the block is not processed.
0	There is no error.
2	SampleTime < 0,001 [s]
30	The value of PV can no longer be displayed in the REAL number field. The last valid value is provided at the `CleanPV` output.
61	TimeConstant < 2 · SampleTime

See also

Smooth block diagram (Page 1150)

Smooth I/Os (Page 1149)

Smooth messaging (Page 1149)

Smooth functions (Page 1147)

Smooth modes (Page 1146)

Description of Smooth (Page 1145)

10.13.5 Smooth messaging

Messaging

This block does not have any message functionality.

See also

Description of Smooth (Page 1145)
Smooth functions (Page 1147)
Smooth I/Os (Page 1149)
Smooth block diagram (Page 1150)
Smooth error handling (Page 1148)
Smooth modes (Page 1146)

10.13.6 Smooth I/Os

Smooth I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
Feature	This parameter currently has no use and is reserved for future functions.	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
FilterOn	1 = Filter active	BOOL	1
OutlCycles	Number of sampling steps to bridge mavericks	INT	3
OutlDetOn	1 = Maverick detection is activated	BOOL	0
OutlThreshold	Limit (trigger threshold) for maverick detection	REAL	10.0
PV	Analog input (process value)	STRUCT: Value: REAL ST: BYTE	- 0.0 16#80
Restart	Restart the filter algorithm	BOOL	1
SampleTime	Sampling time [s] (assigned automatically)	REAL	1.0
TimeConstant	Butterworth filter time constant [s]	REAL	10.0

Output parameters

Parameter	Description	Type	Default
CleanPV	Output of the corrected process value	STRUCT: Value: REAL ST: BYTE	- 0.0 16#80
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see Smooth error handling (Page 1148).	INT	0
OutlDetected	1 = Maverick detected	BOOL	0

See also

Description of Smooth (Page 1145)
Smooth functions (Page 1147)
Smooth messaging (Page 1149)
Smooth block diagram (Page 1150)
Smooth modes (Page 1146)

10.13.7 Smooth block diagram

Smooth block diagram

This block does not come with a block diagram.

See also

Smooth I/Os (Page 1149)
Smooth messaging (Page 1149)
Smooth error handling (Page 1148)
Smooth functions (Page 1147)
Smooth modes (Page 1146)
Description of Smooth (Page 1145)

10.14 Sub02 - subtracting two values

10.14.1 Description of Sub02

Object name (type + number) and family

Type + number: FC 334

Family: Math

Area of application for Sub02

The block is used for the following applications:

- Subtracting two values

How it works

The block subtracts one value from another and outputs the subtraction at the Out output parameter as follows:

```
Out = In1 - In2
```

The worst signal status is also always output at the output parameter.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

For the Sub02 block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Source chart for GainSched function block (gain scheduling) (Page 1444)
- PID controller with Smith predictor (SmithPredictorControl) (Page 1436)
- Override control (Page 1445)

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status1 parameter

This block does not provide the `Status` parameter.

See also

Sub02 block diagram (Page 1156)
Sub02 I/Os (Page 1155)
Sub02 messaging (Page 1154)
Sub02 error handling (Page 1153)
Sub02 functions (Page 1153)
Sub02 modes (Page 1152)

10.14.2 Sub02 modes

Sub02 operating modes

This block does not provide any operating modes.

See also

Sub02 block diagram (Page 1156)
Sub02 I/Os (Page 1155)
Sub02 messaging (Page 1154)
Sub02 error handling (Page 1153)
Sub02 functions (Page 1153)
Description of Sub02 (Page 1151)

10.14.3 Sub02 functions

Functions of Sub02

There are no other functions for this block.

See also

Sub02 block diagram (Page 1156)
Sub02 I/Os (Page 1155)
Sub02 messaging (Page 1154)
Sub02 error handling (Page 1153)
Sub02 modes (Page 1152)
Description of Sub02 (Page 1151)
Selecting signals for processing (Page 93)
Forming and outputting signal status for blocks (Page 88)

10.14.4 Sub02 error handling

Sub02 troubleshooting

The block does not report any errors.

See also

Sub02 block diagram (Page 1156)
Sub02 I/Os (Page 1155)
Sub02 messaging (Page 1154)
Sub02 functions (Page 1153)
Sub02 modes (Page 1152)
Description of Sub02 (Page 1151)

10.14.5 Sub02 messaging

Messaging

This block does not have any message functionality.

See also

Sub02 block diagram (Page 1156)
Sub02 I/Os (Page 1155)
Sub02 error handling (Page 1153)
Sub02 functions (Page 1153)
Sub02 modes (Page 1152)
Description of Sub02 (Page 1151)

10.14.6 Sub02 I/Os

Sub02 I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
In1	Value 1 to subtract	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In2	Value 2 to subtract	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
Out	Product version	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

See also

Sub02 block diagram (Page 1156)
Sub02 messaging (Page 1154)
Sub02 error handling (Page 1153)
Sub02 functions (Page 1153)
Sub02 modes (Page 1152)
Description of Sub02 (Page 1151)

10.14.7 Sub02 block diagram

Sub02 block diagram

A block diagram is not provided for this block.

See also

Sub02 I/Os (Page 1155)
Sub02 messaging (Page 1154)
Sub02 error handling (Page 1153)
Sub02 functions (Page 1153)
Sub02 modes (Page 1152)
Description of Sub02 (Page 1151)

11 Interlock blocks

11.1 Intlk02 - Interlock display with 2 input signals

11.1.1 Description of Intlk02

Object name (type + number) and family

Type + number: FB 1824

Family: Interlck

Area of application for Intlk02

The block is used for the following applications:

- Standardized interlock with display

How it works

The block is used to calculate a standardized interlock that can be displayed on the OS. A maximum of 2 input signals can be supplied to the block. They are linked using selectable binary logic. The signal status of the output signal is also determined. You can assign an analog value with the signal status and unit to each input value for display in the faceplate.

The current state is displayed at the `Out` output parameter:

- `Out = 0`: Interlock
- `Out = 1`: Good state

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

For the Intlk02 block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Dosing (DoseLean) (Page 1449)
- Two-speed motor (Motor2Speed) (Page 1450)
- Reversing motor (MotorReversible) (Page 1450)
- Reversing motor with controllable speed (MotorSpeedControlled) (Page 1451)
- Two-way valve (Valve2Way) (Page 1453)
- Motor valve (ValveMotor) (Page 1454)

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status1 parameter

Status bit	Parameter
0	1 = `Logic` = OR
1	1 = `Logic` = AND
2	Not allocated
3	Result of logic operation `Out.Value`
4	1 = At least one input value is excluded. See the section Excluding input values in Intlk02 functions (Page 1161).
5	1 = All input values are excluded
6	1= No input value is interconnected or `NotUsed.Value`
7 - 31	Not allocated

Status word allocation for Status2 parameter

Status bit	Parameter
0	`In01.Value`
1	`In02.Value`
2 - 31	Not allocated

Status word allocation for Status3 parameter

Status bit	Parameter
0	`InvIn01`
1	`InvIn02`
2 - 31	Not allocated

Status word allocation for Status4 parameter

Status bit	Parameter
0	`In01` with inversion
1	`In02` with inversion
2 - 31	Not allocated

Status word allocation for Status5 parameter

Status bit	Parameter
0	BypIn01
1	BypIn02
2 - 31	Not allocated

Status word allocation for Status6 parameter

Status bit	Parameter
0	1 = In01 not interconnected
1	1 = In02 not interconnected
2 - 31	Not allocated

Status word allocation for Status7 parameter

Status bit	Parameter
0	1 = AV01 not interconnected
1	1 = AV02 not interconnected
2 - 31	Not allocated

Status word allocation for Status8 parameter

Identical to FirstIn

See also

Intlk02 messaging (Page 1163)
Intlk02 I/Os (Page 1164)
Intlk02 block diagram (Page 1167)
Intlk02 error handling (Page 1163)
Intlk02 modes (Page 1160)

11.1.2 Intlk02 modes

Intlk02 modes

This block does not provide any modes.

See also

Intlk02 block diagram (Page 1167)
Intlk02 I/Os (Page 1164)
Intlk02 messaging (Page 1163)
Intlk02 error handling (Page 1163)
Intlk02 functions (Page 1161)
Description of Intlk02 (Page 1157)
Out of service (Page 27)

11.1.3 Intlk02 functions

Functions of Intlk02

The functions for this block are listed below.

Logic operators

Use the `Logic` input to specify the logic operator that the block should employ when determining the interlock state. Make the following settings:

- `Logic = 0: OR`
- `Logic = 1: AND`

Inversion of logic signals

You can invert the input signals by setting the input parameter `InvIn`$_x$ for the input concerned to `In`$_x$ `= 1`, e.g. for input `In01` set the I/O `InvIn01`.

The inversion is displayed in the faceplate. If you invert signals using any other method this is not shown in the faceplate.

Bypass

Note

Bypassing the interlock means that the interlock signal (input signal) is excluded from the logic of the interlock block, in other words, this signal is ignored in the logic operation.

You can exclude input signals that are temporarily not to be used for the calculation in the block by setting the corresponding I/O `BypIn`$_x$ `= 1`. The I/O is shown in the faceplate by the icon  .

Special situation: If all input parameters are excluded, the output value is defined using the `DefaultOut` parameter.

Handling non-connected inputs

Inputs which are not interconnected are not evaluated. They are not displayed in the faceplate either.

Special situation: If no inputs are connected, the output value is defined using the `DefaultOut` parameter. The signal status is set to "Simulation".

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43). However only one additional faceplate can be called up using `SelFp1 = 1`.

Recording the first signal for interlock blocks

This block provides the standard function Recording the first signal for interlock blocks (Page 87).

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `Out.ST`

Operator control permissions (OS Perm)

You can configure the following operator control permissions for this block:

Bit	Function
0	1 = Operator can exclude values
1	1 = Operator can reset the exclusion of input values
2	1 = Operator can reset the recording of the first signal
3 - 31	Not allocated.

Configurable reactions using the Feature I/O

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section.

The following functionality is available for this block at the relevant bits:

Bit	Function
0 - 30	Not allocated
31	Activating recording of the first signal (Page 191)

See also

Description of Intlk02 (Page 1157)

Intlk02 messaging (Page 1163)

Intlk02 I/Os (Page 1164)

Intlk02 block diagram (Page 1167)

Intlk02 error handling (Page 1163)

Intlk02 modes (Page 1160)

11.1.4 Intlk02 error handling

Intlk02 troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Predefined value when inserting the block, the block is not processed.
0	There is no error.
13	The `Logic` parameter has not been set to 0 or 1.

See also

Intlk02 block diagram (Page 1167)
Intlk02 I/Os (Page 1164)
Intlk02 messaging (Page 1163)
Intlk02 functions (Page 1161)
Intlk02 modes (Page 1160)
Description of Intlk02 (Page 1157)

11.1.5 Intlk02 messaging

Message functionality

The block does not have any message functionality.

See also

Description of Intlk02 (Page 1157)
Intlk02 functions (Page 1161)
Intlk02 I/Os (Page 1164)
Intlk02 block diagram (Page 1167)
Intlk02 error handling (Page 1163)
Intlk02 modes (Page 1160)

11.1.6 Intlk02 I/Os

Intlk02 I/Os

Input parameters

Parameter	Description	Type	Default
AV01	Analog value of In01	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV02	Analog value of In02	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV01_Unit	Unit of measure for AV01	INT	0
AV02_Unit	Unit of measure for AV02	INT	0
BypIn01	1 = Input In01 is not used	BOOL	0
BypIn02	1 = Input In02 is not used	BOOL	0
DefaultOut	Output value for cases where all inputs are excluded or not interconnected. Refer to the Intlk02 functions (Page 1161) section of the block.	BOOL	1
EN	1 = Called block will be processed	BOOL	1
Feature	I/O for additional functions (Page 1161)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
In01	Input In01	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In02	Input In02	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
InvIn01	Invert input In01	BOOL	0
InvIn02	Invert input In02	BOOL	0
Logic	Logical operation: 0 = Logical OR 1 = Logical AND	INT	0
NotUsed	1 = Block is not used (only for display in the faceplate)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
OS_Perm	I/O for operator control permissions (Page 1161)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
RstBypOp	1 = Reset via BypIn01 and BypIn02	BOOL	0
RstLi	1 = Reset via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RstOp	1 = Reset via operator	BOOL	0
SelFp1	Call a block saved in this parameter as additional faceplate (Page 43) in the standard view	ANY	-
UserStatus	Freely assignable bits for use in faceplate	BYTE	16#00

Output parameters

Parameter	Description	Data type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see Intlk02 error handling (Page 1163).	INT	-1
FirstIn	Bit-coded number of the first signal that has resulted in a change at the output	DWORD	16#0
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
Out	Output	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word 1 (Page 1157)	DWORD	16#00000000
Status2	Status word 2 (Page 1157)	DWORD	16#00000000
Status3	Status word 3 (Page 1157)	DWORD	16#00000000
Status4	Status word 4 (Page 1157)	DWORD	16#00000000
Status5	Status word 5 (Page 1157)	DWORD	16#00000000
Status6	Status word 6 (Page 1157)	DWORD	16#00000000
Status7	Status word 7 (Page 1157)	DWORD	16#00000000
Status8	Status word 8 (Page 1157)	DWORD	16#00000000

See also

Intlk02 messaging (Page 1163)

Intlk02 block diagram (Page 1167)

Intlk02 modes (Page 1160)

11.1.7 Intlk02 block diagram

Intlk02 block diagram

A block diagram is not provided for this block.

See also

Intlk02 I/Os (Page 1164)

Intlk02 messaging (Page 1163)

Intlk02 error handling (Page 1163)

Intlk02 functions (Page 1161)

Intlk02 modes (Page 1160)

Description of Intlk02 (Page 1157)

11.1.8 Operator control and monitoring

11.1.8.1 Views of interlock blocks

Views of the blocks Intlk02, Intlk04, Intlk08, Intlk16

The blocks provide the following views:

- Standard view of interlock blocks (Page 147)
- Block icon for interlock blocks (Page 184)

Refer to the Structure of the faceplate (Page 123) and Block icon structure (Page 174) sections for general information about the faceplate and block icon.

11.2 Intlk04 - Interlock display with 4 input signals

11.2.1 Description of Intlk04

Object name (type + number) and family

Type + number: FB 1825

Family: Interlck

Area of application for Intlk04

The block is used for the following applications:

- Standardized interlock with display

How it works

The block is used to calculate a standardized interlock that can be displayed on the OS. A maximum of 4 input signals can be supplied to the block. They are linked using selectable binary logic. The signal status of the output signal is also determined. You can assign an analog value with the signal status and unit to each input value for display in the faceplate.

The current state is displayed at the `Out` output parameter:

- `Out = 0`: Interlock
- `Out = 1`: Good state

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

For the Intlk02 (Intlck04) block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Refer to Description of Intlk02 (Page 1157) for more information on this.

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status1 parameter

Status bit	Parameter
0	1= `Logic` = OR
1	1 = `Logic` = AND
2	Not allocated
3	Result of logic operation `Out.Value`
4	1 = At least one input value is excluded. See the section Excluding input values in Intlk04 functions (Page 1171).
5	1 = All input values are connected
6	1= No input value is interconnected or `NotUsed.Value`
7 - 31	Not allocated

Status word allocation for Status2 parameter

Status bit	Parameter
0	`In01.Value`
1	`In02.Value`
2	`In03.Value`
3	`In04.Value`
4 - 31	Not allocated

Status word allocation for Status3 parameter

Status bit	Parameter
0	`InvIn01`
1	`InvIn02`
2	`InvIn03`
3	`InvIn04`
4 - 31	Not allocated

Status word allocation for Status4 parameter

Status bit	Parameter
0	`In01` with inversion
1	`In02` with inversion
2	`In03` with inversion
3	`In04` with inversion
4 - 31	Not allocated

Status word allocation for Status5 parameter

Status bit	Parameter
0	BypIn01
1	BypIn02
2	BypIn03
3	BypIn04
4 - 31	Not allocated

Status word allocation for Status6 parameter

Status bit	Parameter
0	1 = In01 not interconnected
1	1 = In02 not interconnected
2	1 = In03 not interconnected
3	1 = In04 not interconnected
4 - 31	Not allocated

Status word allocation for Status7 parameter

Status bit	Parameter
0	1 = AV01 not interconnected
1	1 = AV02 not interconnected
2	1 = AV03 not interconnected
3	1 = AV04 not interconnected
4 - 31	Not allocated

Status word allocation for Status8 parameter

Identical to FirstIn.

See also

Intlk04 messaging (Page 1175)
Intlk04 I/Os (Page 1175)
Intlk04 block diagram (Page 1178)
Intlk04 error handling (Page 1174)
Intlk04 modes (Page 1171)

11.2.2 Intlk04 modes

Intlk04 modes

This block does not provide any modes.

See also

Intlk04 block diagram (Page 1178)
Intlk04 I/Os (Page 1175)
Intlk04 messaging (Page 1175)
Intlk04 error handling (Page 1174)
Intlk04 functions (Page 1171)
Description of Intlk04 (Page 1168)
Out of service (Page 27)

11.2.3 Intlk04 functions

Functions of Intlk04

The functions for this block are listed below.

Logic operators

Use the `Logic` input to specify the logic operator that the block should employ when determining the interlock state. Make the following settings:

- `Logic = 0: OR`
- `Logic = 1: AND`

Inversion of logic signals

You can invert the input signals by setting the input parameter $\texttt{InvIn}_x$ for the input concerned to $\texttt{In}_x$ `= 1`, e.g. for input `In01` set the I/O `InvIn01`.

The inversion is displayed in the faceplate. If you invert signals using any other method this is not shown in the faceplate.

Bypass

Note

Bypassing the interlock means that the interlock signal (input signal) is excluded from the logic of the interlock block, in other words, this signal is ignored in the logic operation.

You can exclude input signals that are temporarily not to be used for the calculation in the block by setting the corresponding I/O $BypIn_x$ = 1. The I/O is shown in the faceplate by the icon  .

Special situation: If all input parameters are excluded, the output value is defined using the DefaultOut parameter.

Handling non-connected inputs

Inputs which are not interconnected are not evaluated. They are not displayed in the faceplate either.

Special situation: If no inputs are connected, the output value is defined using the DefaultOut parameter. The signal status is set to "Simulation".

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43). However only one additional faceplate can be called up using SelFp1 = 1.

Recording the first signal for interlock blocks

This block provides the standard function Recording the first signal for interlock blocks (Page 87).

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status ST_Worst for the block is formed from the following parameters:

- Out.ST

Operator control permissions (OS Perm)

You can configure the following operator control permissions for this block:

Bit	Function
0	1 = Operator can exclude values
1	1 = Operator can reset the exclusion of input values
2	1 = Operator can reset the recording of the first signal
3 - 31	Not allocated.

Configurable reactions using the Features I/O

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section.

The following functionality is available for this block at the relevant bits:

Bit	Function
0 - 30	Not allocated.
31	Activating recording of the first signal (Page 191)

See also

Description of Intlk04 (Page 1168)

Intlk04 messaging (Page 1175)

Intlk04 I/Os (Page 1175)

Intlk04 block diagram (Page 1178)

Intlk04 error handling (Page 1174)

Intlk04 modes (Page 1171)

11.2.4 Intlk04 error handling

Intlk04 troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Description
-1	Predefined value when inserting the block, the block is not processed.
0	There is no error.
13	The `Logic` parameter has not been set to 0 or 1.

See also

Intlk04 block diagram (Page 1178)

Intlk04 I/Os (Page 1175)

Intlk04 messaging (Page 1175)

Description of Intlk04 (Page 1168)

Intlk04 modes (Page 1171)

Intlk04 functions (Page 1171)

11.2.5 Intlk04 messaging

Message functionality

The block does not have any message functionality.

See also

Description of Intlk04 (Page 1168)
Intlk04 functions (Page 1171)
Intlk04 I/Os (Page 1175)
Intlk04 block diagram (Page 1178)
Intlk04 error handling (Page 1174)
Intlk04 modes (Page 1171)

11.2.6 Intlk04 I/Os

Intlk04 I/Os

Input parameters

Parameter	Description	Type	Default
Av01	Analog value of In01	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
Av02	Analog value of In02	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV03	Analog value of In03	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV04	Analog value of In04	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV01_Unit	Unit of measure for AV01	INT	0
AV02_Unit	Unit of measure for AV02	INT	0
AV03_Unit	Unit of measure for AV03	INT	0
AV04_Unit	Unit of measure for AV04	INT	0
BypIn01	1 = Input In01 is not used	BOOL	0
BypIn02	1 = Input In02 is not used	BOOL	0

Parameter	Description	Type	Default
BypIn03	1 = Input In03 is not used	BOOL	0
BypIn04	1 = Input In04 is not used	BOOL	0
DefaultOut	Output value for the case that all inputs are excluded or not interconnected. Refer to the section Intlk04 functions (Page 1171) of the block.	BOOL	1
EN	1 = Called block will be processed	BOOL	1
Features	I/O for additional functions (Page 1171)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
In01	Input In01	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In02	Input In02	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In03	Input In03	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In04	Input In04	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
InvIn01	Invert input In01	BOOL	0
InvIn02	Invert input In02	BOOL	0
InvIn03	Invert input In03	BOOL	0
InvIn04	Invert input In04	BOOL	0
Logic	Logical operation: 0 = Logical OR 1 = Logical AND	INT	0
NotUsed	1 = Block is not used (only for display in the faceplate)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_Perm	I/O for operator control permissions (Page 1171)	STRUCT • Bit 0: BOOL • .... • Bit 31: BOOL	- • 1 • 1 • 1
RstBypOp	1 = Reset inputs BypIn01 to BypIn04	BOOL	0
RstLi	1 = Reset via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RstOp	1 = Reset via operator	BOOL	0

Parameter	Description	Type	Default
SelFp1	Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
UserStatus	Freely assignable bits for use in faceplate	BYTE	16#00

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see Intlk04 error handling (Page 1174)	INT	-1
FirstIn	Bit-coded number of the first signal that has resulted in a change at the output	DWORD	16#0
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
Out	Output	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Staus1	Status word 1 (Page 1168)	DWORD	16#00000000
Status2	Status word 2 (Page 1168)	DWORD	16#00000000
Status3	Status word 3 (Page 1168)	DWORD	16#00000000
Status4	Status word 4 (Page 1168)	DWORD	16#00000000
Status5	Status word 5 (Page 1168)	DWORD	16#00000000
Status6	Status word 6 (Page 1168)	DWORD	16#00000000
Status7	Status word 7 (Page 1168)	DWORD	16#00000000
Status8	Status word 8 (Page 1168)	DWORD	16#00000000

See also

Intlk04 messaging (Page 1175)

Intlk04 block diagram (Page 1178)

Intlk04 modes (Page 1171)

11.2.7 Intlk04 block diagram

Intlk04 block diagram

A block diagram is not provided for this block.

See also

Intlk04 I/Os (Page 1175)

Intlk04 messaging (Page 1175)

Intlk04 error handling (Page 1174)

Intlk04 functions (Page 1171)

Intlk04 modes (Page 1171)

Description of Intlk04 (Page 1168)

11.2.8 Operator control and monitoring

11.2.8.1 Views of interlock blocks

Views of the blocks Intlk02, Intlk04, Intlk08, Intlk16

The blocks provide the following views:

- Standard view of interlock blocks (Page 147)
- Block icon for interlock blocks (Page 184)

Refer to the Structure of the faceplate (Page 123) and Block icon structure (Page 174) sections for general information about the faceplate and block icon.

11.3 Intlk08 - Interlock display with 8 input signals

11.3.1 Description of Intlk08

Object name (type + number) and family

Type + number: FB 1826

Family: Interlck

Area of application of Intlk08

The block is used for the following applications:

- Standardized interlock with display

How it works

The block is used to calculate a standardized interlock that can be displayed on the OS. A maximum of 2 input signals can be supplied to the block. They are linked using selectable binary logic. The signal status of the output signal is also determined. You can assign an analog value with the signal status and unit to each input value for display in the faceplate.

The current state is displayed at the `Out` output parameter:

- `Out = 0`: Interlock
- `Out = 1`: Good state

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

For the Intlk02 (Intlck08) block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Refer to Description of Intlk02 (Page 1157) for more information on this.

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status1 parameter

Status bit	Parameter
0	1 = `Logic` = OR
1	1 = `Logic` = AND
2	Not allocated
3	Result of logic operation `Out.Value`
4	1 = At least one input value is excluded. See the section Excluding input values in Intlk08 functions (Page 1183).
5	1 = All input values are excluded
6	1= No input value is interconnected or `NotUsed.Value`
7 - 31	Not allocated

Status word allocation for Status2 parameter

Status bit	Parameter
0	`In01.Value`
1	`In02.Value`
2	`In03.Value`
3	`In04.Value`
4	`In05.Value`
5	`In06.Value`
6	`In07.Value`
7	`In08.Value`
8 - 31	Not allocated

Status word allocation for Status3 parameter

Status bit	Parameter
0	`InvIn01`
1	`InvIn02`
2	`InvIn03`
3	`InvIn04`
4	`InvIn05`
5	`InvIn06`
6	`InvIn07`
7	`InvIn08`
8 - 31	Not allocated

Status word allocation for Status4 parameter

Status bit	Parameter
0	In01 with inversion
1	In02 with inversion
2	In03 with inversion
3	In04 with inversion
4	In05 with inversion
5	In06 with inversion
6	In07 with inversion
7	In08 with inversion
8 - 31	Not allocated

Status word allocation for Status5 parameter

Status bit	Parameter
0	BypIn01
1	BypIn02
2	BypIn03
3	BypIn04
4	BypIn05
5	BypIn06
6	BypIn07
7	BypIn08
8 - 31	Not allocated

Status word allocation for Status6 parameter

Status bit	Parameter
0	In01 not interconnected
1	In02 not interconnected
2	In03 not interconnected
3	In04 not interconnected
4	In05 not interconnected
5	In06 not interconnected
6	In07 not interconnected
7	In08 not interconnected
8 - 31	Not allocated

Status word allocation for Status7 parameter

Status bit	Parameter
0	AV01 not interconnected
1	AV02 not interconnected
2	AV03 not interconnected
3	AV04 not interconnected
4	AV05 not interconnected
5	AV06 not interconnected
6	AV07 not interconnected
7	AV08 not interconnected
8 - 31	Not allocated

Status word allocation for Status8 parameter

Identical to FirstIn.

See also

Intlk08 messaging (Page 1185)
Intlk08 I/Os (Page 1186)
Intlk08 block diagram (Page 1190)
Intlk08 error handling (Page 1185)
Intlk08 modes (Page 1182)

11.3.2 Intlk08 modes

Intlk08 modes

This block does not provide any modes.

See also

Intlk08 block diagram (Page 1190)
Intlk08 I/Os (Page 1186)
Intlk08 messaging (Page 1185)
Intlk08 error handling (Page 1185)
Intlk08 functions (Page 1183)
Description of Intlk08 (Page 1179)
Out of service (Page 27)

11.3.3 Intlk08 functions

Functions of Intlk08

The functions for this block are listed below.

Logic operators

Use the `Logic` input to specify the logic operator that the block should employ when determining the interlock state. Make the following settings:

- `Logic = 0: OR`
- `Logic = 1: AND`

Inversion of logic signals

You can invert the input signals by setting the input parameter `InvIn`$_x$ for the input concerned to `In`$_x$ `= 1`, e.g. for input `In01` set the I/O `InvIn01`.

The inversion is displayed in the faceplate. If you invert signals using any other method this is not shown in the faceplate.

Bypass

Note

Bypassing the interlock means that the interlock signal (input signal) is excluded from the logic of the interlock block, in other words, this signal is ignored in the logic operation.

You can exclude input signals that are temporarily not to be used for the calculation in the block by setting the corresponding I/O `BypIn`$_x$ `= 1`. The I/O is shown in the faceplate by the icon  .

Special situation: If all input parameters are excluded, the output value is defined using the `DefaultOut` parameter.

Handling non-connected inputs

Inputs which are not interconnected are not evaluated. They are not displayed in the faceplate either.

Special situation: If no inputs are connected, the output value is defined using the `DefaultOut` parameter. The signal status is set to "Simulation".

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43). However only one additional faceplate can be called up using `SelFp1 = 1`.

Recording the first signal for interlock blocks

This block provides the standard function Recording the first signal for interlock blocks (Page 87).

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `Out.ST`

Operator control permissions (OS Perm)

You can configure the following operator control permissions for this block:

Bit	Function
0	1 = Operator can exclude values
1	1 = Operator can reset the exclusion of input values
2	1 = Operator can reset the recording of the first signal
3 - 31	Not allocated.

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section.

The following functionality is available for this block at the relevant bits:

Bit	Function
0 - 30	Not allocated
31	Activating recording of the first signal (Page 191)

See also

Description of Intlk08 (Page 1179)

Intlk08 messaging (Page 1185)

Intlk08 I/Os (Page 1186)

Intlk08 block diagram (Page 1190)

Intlk08 error handling (Page 1185)

Intlk08 modes (Page 1182)

11.3.4 Intlk08 error handling

Intlk08 troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Description
-1	Predefined value when inserting the block, the block is not processed.
0	No active fault
13	The `Logic` parameter has not been set to 0 or 1.

See also

Intlk08 block diagram (Page 1190)
Intlk08 I/Os (Page 1186)
Intlk08 messaging (Page 1185)
Intlk08 functions (Page 1183)
Intlk08 modes (Page 1182)
Description of Intlk08 (Page 1179)

11.3.5 Intlk08 messaging

Message functionality

The block does not have any message functionality.

See also

Description of Intlk08 (Page 1179)
Intlk08 functions (Page 1183)
Intlk08 I/Os (Page 1186)
Intlk08 block diagram (Page 1190)
Intlk08 error handling (Page 1185)
Intlk08 modes (Page 1182)

11.3.6 Intlk08 I/Os

Intlk08 I/Os

Input parameters

Parameter	Description	Type	Default
AV01	Analog value of In01	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV02	Analog value of In02	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV03	Analog value of In03	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV04	Analog value of In04	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV05	Analog value of In05	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV06	Analog value of In06	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV07	Analog value of In07	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV08	Analog value of In08	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV01_Unit	Unit of measure for AV01	INT	0
AV02_Unit	Unit of measure for AV02	INT	0
AV03_Unit	Unit of measure for AV03	INT	0
AV04_Unit	Unit of measure for AV04	INT	0
AV05_Unit	Unit of measure for AV05	INT	0
AV06_Unit	Unit of measure for AV06	INT	0
AV07_Unit	Unit of measure for AV07	INT	0
AV08_Unit	Unit of measure for AV08	INT	0
BypIn01	1 = Input In01 is not used	BOOL	0
BypIn02	1 = Input In02 is not used	BOOL	0

Parameter	Description	Type	Default
BypIn03	1 = Input In03 is not used	BOOL	0
BypIn04	1 = Input In04 is not used	BOOL	0
BypIn05	1 = Input In05 is not used	BOOL	0
BypIn06	1 = Input In06 is not used	BOOL	0
BypIn07	1 = Input In07 is not used	BOOL	0
BypIn08	1 = Input In08 is not used	BOOL	0
DefaultOut	Output value for cases where all inputs are excluded or not interconnected. Refer to the block's Intlk08 functions (Page 1183) section.	BOOL	1
EN	1 = Called block will be processed	BOOL	1
Features	I/O for additional functions (Page 1183)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
In01	Input In01	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In02	Input In02	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In03	Input In03	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In04	Input In04	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In05	Input In05	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In06	Input In06	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In07	Input In07	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
In08	Input In08	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
InvIn01	Invert Input In01	BOOL	0
InvIn02	Invert Input In02	BOOL	0
InvIn03	Invert Input In03	BOOL	0

Parameter	Description	Type	Default
InvIn04	Invert Input In04	BOOL	0
InvIn05	Invert Input In05	BOOL	0
InvIn06	Invert Input In06	BOOL	0
InvIn07	Invert Input In07	BOOL	0
InvIn08	Invert Input In08	BOOL	0
Logic	Logical operation: 0 = Logical OR 1 = Logical AND	INT	0
NotUsed	1 = Block is not used (only for display in the faceplate)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_Perm	I/O for operator control permissions (Page 1183)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
RstBypOp	1 = Reset inputs BypIn01 to BypIn08	BOOL	0
RstLi	1 = Reset via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RstOp	1 = Reset via operator	BOOL	0
SelFp1	Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
UserStatus	Freely assignable bits for use in faceplate	BYTE	16#00

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see Intlk08 error handling (Page 1185)	INT	-1
FirstIn	Bit-coded number of the first signal that has resulted in a change at the output	DWORD	16#0
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
Out	Output	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word 1 (Page 1179)	DWORD	16#00000000
Status2	Status word 2 (Page 1179)	DWORD	16#00000000
Status3	Status word 3 (Page 1179)	DWORD	16#00000000
Status4	Status word 4 (Page 1185)	DWORD	16#00000000
Status5	Status word 5 (Page 1179)	DWORD	16#00000000
Status6	Status word 6 (Page 1179)	DWORD	16#00000000
Status7	Status word 7 (Page 1179)	DWORD	16#00000000
Status8	Status word 8 (Page 1179)	DWORD	16#00000000

See also

Intlk08 block diagram (Page 1190)

Intlk08 modes (Page 1182)

11.3.7 Intlk08 block diagram

Intlk08 block diagram

A block diagram is not provided for this block.

See also

Intlk08 I/Os (Page 1186)
Intlk08 messaging (Page 1185)
Intlk08 error handling (Page 1185)
Intlk08 functions (Page 1183)
Intlk08 modes (Page 1182)
Description of Intlk08 (Page 1179)

11.3.8 Operator control and monitoring

11.3.8.1 Views of interlock blocks

Views of the blocks Intlk02, Intlk04, Intlk08, Intlk16

The blocks provide the following views:

- Standard view of interlock blocks (Page 147)
- Block icon for interlock blocks (Page 184)

Refer to the Structure of the faceplate (Page 123) and Block icon structure (Page 174) sections for general information about the faceplate and block icon.

11.4 Intlk16 - Interlock display with 16 input signals

11.4.1 Description of Intlk16

Object name (type + number) and family

Type + number: FB 1827

Family: Interlck

Area of application for Intlk16

The block is used for the following applications:

- Standardized interlock with display

How it works

The block is used to calculate a standardized interlock that can be displayed on the OS. A maximum of 2 input signals can be supplied to the block. They are linked using selectable binary logic. The signal status of the output signal is also determined. You can assign an analog value with the signal status and unit to each input value for display in the faceplate.

The current state is displayed at the `Out` output parameter:

- `Out = 0`: Interlock
- `Out = 1`: Good state

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

For the Intlk02 (Intlck16) block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Refer to Description of Intlk02 (Page 1157) for more information on this.

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status1 parameter

Status bit	Parameter
0	1 = `Logic` = OR
1	1 = `Logic` = AND
2	Not allocated
3	Result of logic operation `Out.Value`
4	1 = At least one input value is excluded. See the section Excluding input values in Intlk16 functions (Page 1196).
5	1 = All input values are excluded
6	1= No input value is interconnected or `NotUsed.Value`
7 - 31	Not allocated

Status word allocation for Status2 parameter

Status bit	Parameter
0	`In01.Value`
1	`In02.Value`
2	`In03.Value`
3	`In04.Value`
4	`In05.Value`
5	`In06.Value`
6	`In07.Value`
7	`In08.Value`
8	`In09.Value`
9	`In10.Value`
10	`In11.Value`
11	`In12.Value`
12	`In13.Value`
13	`In14.Value`
14	`In15.Value`
15	`In16.Value`
16 - 31	Not allocated

Status word allocation for Status3 parameter

Status bit	Parameter
0	InvIn01
1	InvIn02
2	InvIn03
3	InvIn04
4	InvIn05
5	InvIn06
6	InvIn07
7	InvIn08
8	InvIn09
9	InvIn10
10	InvIn11
11	InvIn12
12	InvIn13
13	InvIn14
14	InvIn15
15	InvIn16
16 - 31	Not allocated

Status word allocation for Status4 parameter

Status bit	Parameter
0	In01 with inversion
1	In02 with inversion
2	In03 with inversion
3	In04 with inversion
4	In05 with inversion
5	In06 with inversion
6	In07 with inversion
7	In08 with inversion
8	In09 with inversion
9	In10 with inversion
10	In11 with inversion
11	In12 with inversion
12	In13 with inversion
13	In14 with inversion
14	In15 with inversion
15	In16 with inversion
16 - 31	Not allocated

Status word allocation for Status5 parameter

Status bit	Parameter
0	BypIn01
1	BypIn02
2	BypIn03
3	BypIn04
4	BypIn05
5	BypIn06
6	BypIn07
7	BypIn08
8	BypIn09
9	BypIn10
10	BypIn11
11	BypIn12
12	BypIn13
13	BypIn14
14	BypIn15
15	BypIn16
16 - 31	Not allocated

Status word allocation for Status6 parameter

Status bit	Parameter
0	In01 not interconnected
1	In02 not interconnected
2	In03 not interconnected
3	In04 not interconnected
4	In05 not interconnected
5	In06 not interconnected
6	In07 not interconnected
7	In08 not interconnected
8	In09 not interconnected
9	In10 not interconnected
10	In11 not interconnected
11	In12 not interconnected
12	In13 not interconnected
13	In14 not interconnected
14	In15 not interconnected
15	In16 not interconnected
16 - 31	Not allocated

Status word allocation for Status7 parameter

Status bit	Parameter
0	AV01 not interconnected
1	AV02 not interconnected
2	AV03 not interconnected
3	Av04 not interconnected
4	AV05 not interconnected
5	AV06 not interconnected
6	AV07 not interconnected
7	AV08 not interconnected
8	AV09 not interconnected
9	AV10 not interconnected
10	AV11 not interconnected
11	AV12 not interconnected
12	AV13 not interconnected
13	AV14 not interconnected
14	AV15 not interconnected
15	AV16 not interconnected
16 - 31	Not allocated

Status word allocation for Status8 parameter

Identical to FirstIn.

See also

Intlk16 block diagram (Page 1206)

Intlk16 error handling (Page 1199)

Intlk16 modes (Page 1196)

Intlk16 I/Os (Page 1200)

Intlk16 messaging (Page 1199)

11.4.2 Intlk16 modes

Intlk16 modes

This block does not provide any modes.

See also

Intlk16 block diagram (Page 1206)

Intlk16 I/Os (Page 1200)

Intlk16 messaging (Page 1199)

Intlk16 functions (Page 1196)

Intlk16 error handling (Page 1199)

Description of Intlk16 (Page 1191)

Out of service (Page 27)

11.4.3 Intlk16 functions

Functions of Intlk16

The functions for this block are listed below.

Logic operators

Use the `Logic` input to specify the logic operator that the block should employ when determining the interlock state. Make the following settings:

- `Logic = 0: OR`
- `Logic = 1: AND`

Inversion of logic signals

You can invert the input signals by setting the input parameter $\texttt{InvIn}_x$ for the input concerned to $\texttt{In}_x$ `= 1`, e.g. for input `In01` set the I/O `InvIn01`.

The inversion is displayed in the faceplate. If you invert signals using any other method this is not shown in the faceplate.

Bypass

Note

Bypassing the interlock means that the interlock signal (input signal) is excluded from the logic of the interlock block, in other words, this signal is ignored in the logic operation.

You can exclude input signals that are temporarily not to be used for the calculation in the block by setting the corresponding I/O $BypIn_x$ = 1. The I/O is shown in the faceplate by the icon  .

Special situation: If all input parameters are excluded, the output value is defined using the DefaultOut parameter.

Handling non-connected inputs

Inputs which are not interconnected are not evaluated. They are not displayed in the faceplate either.

Special situation: If no inputs are connected, the output value is defined using the DefaultOut parameter. The signal status is set to "Simulation".

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43). However only one additional faceplate can be called up using SelFp1 = 1.

Recording the first signal for interlock blocks

This block provides the standard function Recording the first signal for interlock blocks (Page 87).

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status ST_Worst for the block is formed from the following parameter:

- OUT.ST

Operator control permissions (OS Perm)

You can configure the following operator control permissions for this block:

Bit	Function
0	1 = Operator can exclude values
1	1 = Operator can reset the exclusion of input values
2	1 = Operator can reset the recording of the first signal
3 - 31	Not allocated.

Configurable reactions using the Features I/O

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section.

The following functionality is available for this block at the relevant bits:

Bit	Function
0 - 30	Not allocated
31	Activating recording of the first signal (Page 191)

See also

Intlk16 block diagram (Page 1206)

Intlk16 error handling (Page 1199)

Intlk16 modes (Page 1196)

Description of Intlk16 (Page 1191)

Intlk16 I/Os (Page 1200)

Intlk16 messaging (Page 1199)

11.4.4 Intlk16 error handling

Intlk16 troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Predefined value when inserting the block, the block is not processed.
0	There is no error.
13	The `Logic` parameter has not been set to 0 or 1.

See also

Intlk16 block diagram (Page 1206)

Intlk16 I/Os (Page 1200)

Intlk16 messaging (Page 1199)

Intlk16 functions (Page 1196)

Intlk16 modes (Page 1196)

Description of Intlk16 (Page 1191)

11.4.5 Intlk16 messaging

Message functionality

The block does not have any message functionality.

See also

Intlk16 block diagram (Page 1206)

Intlk16 error handling (Page 1199)

Intlk16 modes (Page 1196)

Description of Intlk16 (Page 1191)

Intlk16 I/Os (Page 1200)

Intlk16 functions (Page 1196)

11.4.6 Intlk16 I/Os

Intlk16 I/Os

Input parameters

Parameter	Description	Type	Default
AV01	Analog value of In01	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV02	Analog value of In02	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV03	Analog value of In03	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV04	Analog value of In04	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV05	Analog value of In05	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV06	Analog value of In06	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV07	Analog value of In07	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV08	Analog value of In08	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV09	Analog value of In09	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV10	Analog value of In10	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV11	Analog value of In11	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF

Parameter	Description	Type	Default
AV12	Analog value of In12	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV13	Analog value of In13	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV14	Analog value of In14	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV15	Analog value of In15	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV16	Analog value of In16	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
AV01_Unit	Unit of measure for AV01	INT	0
AV02_Unit	Unit of measure for AV02	INT	0
AV03_Unit	Unit of measure for AV03	INT	0
AV04_Unit	Unit of measure for AV04	INT	0
AV05_Unit	Unit of measure for AV05	INT	0
AV06_Unit	Unit of measure for AV06	INT	0
AV07_Unit	Unit of measure for AV07	INT	0
AV08_Unit	Unit of measure for AV08	INT	0
AV09_Unit	Unit of measure for AV09	INT	0
AV10_Unit	Unit of measure for AV10	INT	0
AV11_Unit	Unit of measure for AV11	INT	0
AV12_Unit	Unit of measure for AV12	INT	0
AV13_Unit	Unit of measure for AV13	INT	0
AV14_Unit	Unit of measure for AV14	INT	0
AV15_Unit	Unit of measure for AV15	INT	0
AV16_Unit	Unit of measure for AV16	INT	0
BypIn01	1 = Input In01 is not used	BOOL	0
BypIn02	1 = Input In02 is not used	BOOL	0
BypIn03	1 = Input In03 is not used	BOOL	0
BypIn04	1 = Input In04 is not used	BOOL	0
BypIn05	1 = Input In05 is not used	BOOL	0
BypIn06	1 = Input In06 is not used	BOOL	0
BypIn07	1 = Input In07 is not used	BOOL	0
BypIn08	1 = Input In08 is not used	BOOL	0
BypIn09	1 = Input In09 is not used	BOOL	0
BypIn10	1 = Input In10 is not used	BOOL	0

Parameter	Description	Type	Default
BypIn11	1 = Input In11 is not used	BOOL	0
BypIn12	1 = Input In12 is not used	BOOL	0
BypIn13	1 = Input In13 is not used	BOOL	0
BypIn14	1 = Input In14 is not used	BOOL	0
BypIn15	1 = Input In15 is not used	BOOL	0
BypIn16	1 = Input In16 is not used	BOOL	0
DefaultOut	Output value for cases where all inputs are excluded or not interconnected. Refer to the block's Intlk16 functions (Page 1196) section.	BOOL	1
Features	I/O for additional functions (Page 1196)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
In01	Input In01	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In02	Input In02	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In04	Input In04	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In05	Input In05	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In06	Input In06	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In07	Input In07	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In08	Input In08	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In09	Input In09	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In10	Input In10	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF

Parameter	Description	Type	Default
In11	Input In11	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In12	Input In12	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In13	Input In13	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In14	Input In14	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In15	Input In15	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
In16	Input In16	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#FF
InvIn01	Invert input In01	BOOL	0
InvIn02	Invert input In02	BOOL	0
InvIn03	Invert input In03	BOOL	0
InvIn04	Invert input In04	BOOL	0
InvIn05	Invert input In05	BOOL	0
InvIn06	Invert input In06	BOOL	0
InvIn07	Invert input In07	BOOL	0
InvIn08	Invert input In08	BOOL	0
InvIn09	Invert input In09	BOOL	0
InvIn10	Invert input In10	BOOL	0
InvIn11	Invert input In11	BOOL	0
InvIn12	Invert input In12	BOOL	0
InvIn13	Invert input In13	BOOL	0
InvIn14	Invert input In14	BOOL	0
InvIn15	Invert input In15	BOOL	0
InvIn16	Invert input In16	BOOL	0
Logic	Logical operation: 0 = Logical OR 1 = Logical AND	INT	0
NotUsed	1 = Block is not used (only for display in the faceplate)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
OS_Perm	I/O for operator control permissions (Page 1196)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
RstBypOp	1 = Reset inputs BypIn01 to BypIn16	BOOL	0
RstLi	1 = Reset via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RstOp	1 = Reset via operator	BOOL	0
SelFp1	Call a block saved in this parameter as Calling further faceplates (Page 43) in standard view	ANY	-
UserStatus	Freely assignable bits for use in faceplate	BYTE	16#00

Output parameters

Parameter	Description	Type	Default
ErrorNum	Output of current error number. For error numbers that can be output by this block, see Intlk16 error handling (Page 1199)	INT	-1
FirstIn	Bit-coded number of the first signal that has resulted in a change at the output	DWORD	16#0
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Prem with settings changed by the block algorithm	DWORD	16#FFFFFFFF
Out	Output	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word 1 (Page 1191)	DWORD	16#00000000
Status2	Status word 2 (Page 1191)	DWORD	16#00000000
Status3	Status word 3 (Page 1191)	DWORD	16#00000000
Status4	Status word 4 (Page 1191)	DWORD	16#00000000
Status5	Status word 5 (Page 1191)	DWORD	16#00000000
Status6	Status word 6 (Page 1191)	DWORD	16#00000000
Status7	Status word 7 (Page 1191)	DWORD	16#00000000
Status8	Status word 8 (Page 1191)	DWORD	16#00000000

See also

Intlk16 block diagram (Page 1206)

Intlk16 modes (Page 1196)

Intlk16 messaging (Page 1199)

11.4.7 Intlk16 block diagram

Intlk16 block diagram

A block diagram is not provided for this block.

See also

Intlk16 I/Os (Page 1200)

Intlk16 messaging (Page 1199)

Intlk16 functions (Page 1196)

Intlk16 error handling (Page 1199)

Intlk16 modes (Page 1196)

Description of Intlk16 (Page 1191)

11.4.8 Operator control and monitoring

11.4.8.1 Views of interlock blocks

Views of the blocks Intlk02, Intlk04, Intlk08, Intlk16

The blocks provide the following views:

- Standard view of interlock blocks (Page 147)
- Block icon for interlock blocks (Page 184)

Refer to the Structure of the faceplate (Page 123) and Block icon structure (Page 174) sections for general information about the faceplate and block icon.

Monitoring blocks

12

12.1 AV - displaying and monitoring additional value

12.1.1 Description of AV

Object name (type + number) and family

Type + number: FB 1903

Family: Monitor

Area of application for AV

The block is used for the following applications:

- Monitoring an additional analog value at a technological block (for example a motor, valve).

How it works

The block must be connected to a driver block and monitors an additional analog value. The messages of the AV block appear in the message view of the technological block connected to it. Monitoring limits are configured and controlled at the technological block.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

Interconnect the AV_Tech output parameter of the AV block to the AV input parameter of the technological block in the CFC.

Note

Interconnection of the block to multiple technological blocks is not permitted.

For the AV block, the Advanced Process Library contains process tag type templates; these serve as examples by providing various application scenarios for this block.

Examples of process tag types:

- Motor with an additional analog value and time-stamped signals (Motor_AV_EventTs) (Page 1451)

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

Status word allocation for `Status1` parameter

For a description of the individual parameters, see the AV I/Os (Page 1214) section.

Status bit	Parameter
0	`Occupied`
1	`BatchEn`
2	`SimOn`
3	`MsgLock`
4	`AV_AH_Act.Value`
5	`AV_WH_Act.Value`
6	`AV_TH_Act.Value`
7	`AV_TL_Act.Value`
8	`AV_WL_Act.Value`
9	`AV_AL_Act.Value`
10	`AV_AH_En`
11	`AV_WH_En`
12	`AV_TH_En`
13	`AV_TL_En`
14	`AV_WL_En`
15	`AV_AL_En`
16	`AV_AH_MsgEn`
17	`AV_WH_MsgEn`
18	`AV_TH_MsgEn`
19	`AV_TL_MsgEn`
20	`AV_WL_MsgEn`
21	`AV_AL_MsgEn`

See also

AV modes (Page 1209)

AV functions (Page 1209)

AV error handling (Page 1211)

AV messaging (Page 1212)

AV block diagram (Page 1216)

12.1.2 AV modes

AV operating modes

The block does not have any of its own operating modes.

If the interconnected technological block is set to the operating mode "Out of service" the block AV is also set to the operating mode "Out of service". In this case `AV_Out = AV`.

See also

Description of AV (Page 1207)

AV functions (Page 1209)

AV error handling (Page 1211)

AV messaging (Page 1212)

AV I/Os (Page 1214)

AV block diagram (Page 1216)

12.1.3 AV functions

Functions of AV

The functions for this block are listed below.

Alarm delays with two time values per limit pair

This block includes the standard function alarm delay for Two time values per limit pair (Page 80).

Limit monitoring of an additional analog value

This block provides the standard function Limit monitoring of an additional analog value (Page 73).

Limit monitoring with hysteresis

This block provides the standard function Limit monitoring with hysteresis (Page 78). It is performed via the input parameter `AV_Hyst`.

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status `ST_Worst` for the block is formed from the following parameter:

- `AV_Out.ST`

Simulating signals

The simulation in block AV is activated at the technological block (`SimOn` = 1). The simulation value `SimAV` for the block AV is also set there.

Maintenance release

The maintenance release in block AV is activated at the technological block (`MS_Release` = 1).

Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 53).

Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 46).

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)
22	Update acknowledgment and error status of the message call (Page 193)

See also

AV modes (Page 1209)

AV error handling (Page 1211)

AV messaging (Page 1212)

AV I/Os (Page 1214)

AV block diagram (Page 1216)

12.1.4 AV error handling

AV troubleshooting

The following errors can be displayed for this block:

- Error numbers
- Mode changeover error

Overview of error numbers

The `ErrorNum` output parameter can be used to output various error numbers. An overview of all error numbers is available in the section "Functions of blocks > Error handling (Page 117)".

Error numbers output by this block:

Error number	Meaning of the error number
-1	Default value when implementing the block; block will not be processed.
0	No active fault
32	The value of AV can no longer be displayed in the REAL number field.

Mode changeover error

This error can be output by the block, see the section Error handling (Page 117).

See also

Description of AV (Page 1207)

AV modes (Page 1209)

AV functions (Page 1209)

AV messaging (Page 1212)

AV I/Os (Page 1214)

AV block diagram (Page 1216)

12.1.5 AV messaging

Messaging

The following messages can be generated for this block:

- Process messages

Process messages

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 1	Alarm - high	$$BlockComment$$ AV - High alarm limit violated
	SIG 2	Warning - high	$$BlockComment$$ AV - High warning limit violated
	SIG 3	Tolerance - high	$$BlockComment$$ AV - High tolerance limit violated
	SIG 4	Tolerance - low	$$BlockComment$$ AV - Low tolerance limit violated
	SIG 5	Warning - low	$$BlockComment$$ AV - Low warning limit violated
	SIG 6	Alarm - low	$$BlockComment$$ AV - Low alarm limit violated

Explanation

$$BlockComment$$: Content of the instance-specific comment

Associated values for message instance `MsgEvId1`

Associated value	Block parameters
1	Reserved
2	Reserved
3	Reserved
4	AV
5	`ExtVa105`
6	`ExtVa106`
7	`ExtVa107`
8	Reserved
9	Reserved
10	Reserved

The associated values 5 to 7 are allocated to the parameters `ExtVa105` ... `ExtVa107` and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

See also

Description of AV (Page 1207)

AV modes (Page 1209)

AV functions (Page 1209)

AV error handling (Page 1211)

AV I/Os (Page 1214)

AV block diagram (Page 1216)

12.1.6 AV I/Os

AV I/Os

Input parameters

Parameter	Description	Type	Default
AV	Analog value	STRUCT • Value:REAL • ST:BYTE	- • 0.0 • 16#80
AV_Unit	Unit of measure for input parameter	INT	1001
AV_A_DC	Delay time for incoming alarms [s]	REAL	0.0
AV_A_DG	Delay time for outgoing alarms [s]	REAL	0.0
AV_AH_En	1 = Enable high alarm	BOOL	1
AV_AH_MsgEn	1 = Enable high alarm message	BOOL	1
AV_AL_En	1 = Enable low alarm	BOOL	1
AV_AL_MsgEn	1 = Enable low alarm message	BOOL	1
AV_OpScale	Limit for scale in bar graph of faceplate for the analog value	STRUCT • High:REAL • Low:REAL	- • 100.0 • 0.0
AV_T_DC	Delay time for incoming tolerances [s]	REAL	0.0
AV_T_DG	Delay time for outgoing tolerances [s]	REAL	0.0
AV_TH_En	1 = Enable high tolerance	BOOL	0
AV_TH_MsgEn	1 = Enable high tolerance message	BOOL	1
AV_TL_En	1 = Enable low tolerance	BOOL	0
AV_TL_MsgEn	1 = Enable low tolerance message	BOOL	1
AV_W_DC	Delay time for incoming warnings [s]	REAL	0.0
AV_W_DG	Delay time for outgoing warnings [s]	REAL	0.0
AV_WH_En	1 = Enable high warning	BOOL	1
AV_WH_MsgEn	1 = Enable high warning message	BOOL	1
AV_WL_En	1 = Enable low warning	BOOL	1
AV_WL_MsgEn	1 = Enable low warning message	BOOL	1
EN	1 = Called block will be processed	BOOL	1
ExtVa105	Associated value 5 for messages (MsgEvID1)	ANY	0.0
ExtVa106	Associated value 6 for messages (MsgEvID1)	ANY	0.0
ExtVa107	Associated value 7 for messages (MsgEvID1)	ANY	0.0

Parameter	Description	Type	Default
Feature	I/O for additional functions (Page 1209)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
MsgEvId1	Message number (assigned automatically)	DWORD	16#FF
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1

Output parameters

Parameter	Description	Type	Default
AV_AH_Act	1 = High alarm active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AV_AL_Act	1 = Low alarm active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AV_Out	Analog value output	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
AV_Tech	Output parameter with which the input parameter AV of the technological block has to be connected. This includes tags which are passed on to the technological block for additional processing.	STRUCT	-
AV_TH_Act	1 = High tolerance active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AV_TL_Act	1 = Low tolerance active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AV_WH_Act	1 = High warning active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AV_WL_Act	1 = Low warning active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ENO	1 = Block algorithm completed without errors	BOOL	0

Parameter	Description	Type	Default
ErrorNum	Output of current error number. For error numbers that can be output by this block, see AV error handling (Page 1211)	INT	-1
MS_Release	Maintenance release: 1 = Release for OS operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MsgAckn1	Alarm acknowledgement status 1 (output STATUS of first ALARM_8P)	WORD	16#0000
MsgErr1	Alarm error 1 (output ERROR of first ALARM_8P)	BOOL	0
MsgStat1	Alarm status 1 (output ERROR of first ALARM_8P)	WORD	16#0000
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word 1 (Page 1207)	DWORD	16#00000000

See also

AV modes (Page 1209)

AV messaging (Page 1212)

AV block diagram (Page 1216)

12.1.7 AV block diagram

AV block diagram

A block diagram is not provided for this block.

See also

Description of AV (Page 1207)

AV modes (Page 1209)

AV functions (Page 1209)

AV error handling (Page 1211)

AV messaging (Page 1212)

AV I/Os (Page 1214)

12.2 MonAnL - Monitoring of an analog process tag

12.2.1 Description of MonAnL

Object name (type+number) and family

Type + number: FB 1845

Family: Monitor

Area of application for MonAnL

The block is used for the following fields of applications:

- Monitoring an analog process process value, e.g. from a channel driver block
- Monitoring of the gradient of an analog process value

How it works

The MonAnL block is used to monitor an analog process tag and the corresponding limits. It also monitors the gradient of these signals. The block generates and outputs corresponding messages if limits are violated or if a signal gradient does not meet requirements.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

For the MonAnL block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Monitoring an analog process tag (AnalogMonitoring) (Page 1449)

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

Status word allocation for `Status1` parameter

For a description of the individual parameters, see the section MonAnL I/Os (Page 1230).

Status bit	Parameter
0	`Occupied`
1	`BatchEn`
2	`SimOn`
3	`OosAct.Value`
4	`OosLi.Value`
5	Not used
6	`OnAct.Value`
7 - 29	Not used
30	Auxiliary value 1 visible
31	Auxiliary value 2 visible

Status word allocation for `Status2` parameter

Status bit	Parameter
0	`MsgLock`
1	`PV_AH_Act.Value`
2	`PV_WH_Act.Value`
3	`PV_TH_Act.Value`
4	`PV_TL_Act.Value`
5	`PV_WL_Act.Value`
6	`PV_AL_Act.Value`
7	`PV_AH_En`
8	`PV_WH_En`
9	`PV_TH_En`
10	`PV_TL_En`
11	`PV_WL_En`
12	`PV_AL_En`
13	`PV_AH_MsgEn`
14	`PV_WH_MsgEn`
15	`PV_TH_MsgEn`
16	`PV_TL_MsgEn`
17	`PV_WL_MsgEn`
18	`PV_AL_MsgEn`
19 - 30	Not used
31	`MS_RelOp`

Status word allocation for `Status3` parameter

Status bit	Parameter
0	Effective `Signal1` of the message block connected via `EventTsIn`
1	Effective `Signal2` of the message block connected via `EventTsIn`
2	Effective `Signal3` of the message block connected via `EventTsIn`
3	Effective `Signal4` of the message block connected via `EventTsIn`
4	Effective `Signal5` of the message block connected via `EventTsIn`
5	Effective `Signal6` of the message block connected via `EventTsIn`
6	Effective `Signal7` of the message block connected via `EventTsIn`
7	Effective `Signal8` of the message block connected via `EventTsIn`
8 - 31	Not used

See also

MonAnL functions (Page 1221)
MonAnL messaging (Page 1227)
MonAnL block diagram (Page 1235)
MonAnL error handling (Page 1226)
MonAnL modes (Page 1220)

12.2.2 MonAnL modes

MonAnL operating modes

The block can be operated using the following modes:

- On (Page 27)
- Out of service (Page 27)

"On"

You can find general information about the "On" mode in the On (Page 27) section.

"Out of service"

You can find general information about the "Out of service" mode in the Out of service (Page 27) section.

See also

MonAnL block diagram (Page 1235)

MonAnL I/Os (Page 1230)

MonAnL messaging (Page 1227)

MonAnL error handling (Page 1226)

MonAnL functions (Page 1221)

Description of MonAnL (Page 1217)

12.2.3 MonAnL functions

Functions of MonAnL

The functions for this block are listed below.

Alarm delays with two time values per limit pair

This block includes the standard function alarm delay for Two time values per limit pair (Page 80).

Limit monitoring of the process value

This block provides the standard function Limit monitoring of the process value (Page 70).

Gradient monitoring

The gradient:

- Limit (high) for positive gradients `GradHUpLim`
- Limit (high) for negative gradients `GradHDnLim`
- Limit (low) for absolute gradients `GradLLim`

The individual monitoring functions are activated at the corresponding "Enable" parameters, e.g. `GradHUpEn` for activating the high gradient limit for positive gradients (`GradHUpLim`).

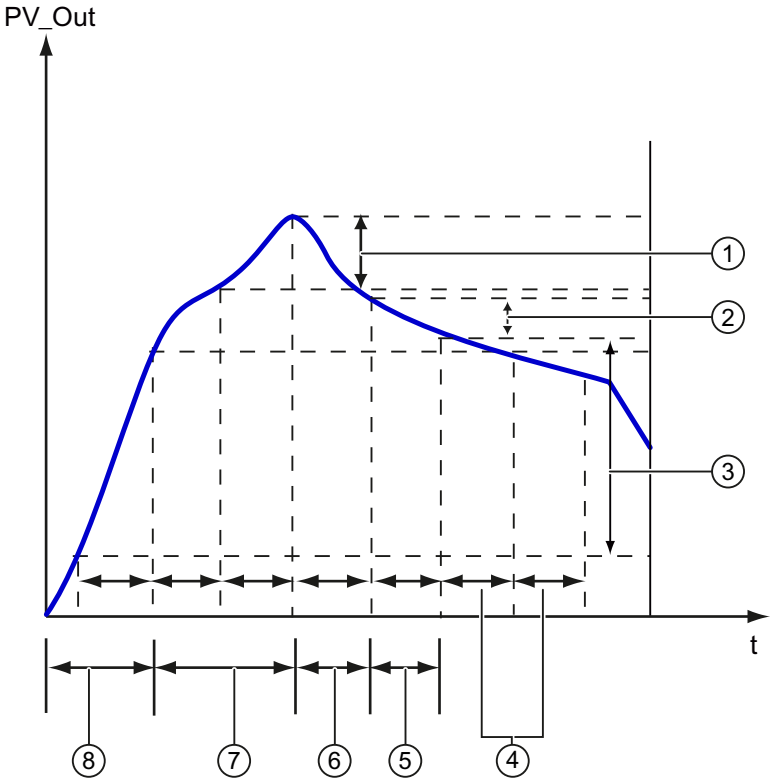
When the values you have defined are reached or exceeded, this is indicated at the corresponding "Active" output parameters, e.g. with `GradHUpAct = 1` for the limit (high) for positive gradients.

Messages can be output for these alarms. You activate these as follows:

- Message for alarms (high) for positive gradients: `GradHUpMsgEn = 1`
- Message for alarms (high) for negative gradients: `GradHDnMsgEn = 1`
- Message for alarms (low) for absolute gradients: `GradLMsgEn = 1`

Example of generating alarms for gradient monitoring

The following example shows how alarms for gradient monitoring are generated.



Number	Description
1	Absolute gradient difference ≥ `GradHDnLim`
2	Absolute gradient difference ≤ `GradLLim`
3	Gradient difference ≥ `GradHUpLim`
4	Sampling time (`SampleTime`)
5	Alarm (low) for absolute gradients
6	Alarm (high) for negative gradients
7	No alarm
8	Alarm (high) for positive gradients

Displaying auxiliary values

This block provides the standard function Displaying auxiliary values (Page 52).

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `PV_Out.ST`

Dead band

To suppress values that fluctuate about zero, this block has the standard function Deadband (Page 99).

Maintenance release

This block provides the standard function Maintenance release (Page 47).

Simulating signals

This block provides the standard function Simulating signals (Page 93).

Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 53).

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)
1	Reaction to the out of service mode (Page 199)
22	Update acknowledgment and error status of the message call (Page 193)

Operator control permissions

The block has the following Operator permissions (Page 45) for the `OS_Perm` parameter:

Bit	Function
0	Not allocated
1	1 = Operator can switch to "On" mode
2	Not allocated
3	1 = Operator can switch to "Out of service" mode
4 - 6	Not allocated
7	1 = The operator can reset the maximum values
8 - 10	Not allocated
11	1 = Operator can enable function simulation
12	1 = Operator can enable the maintenance release function
13	1 = Operator can change the limit (PV) for high alarm
14	1 = Operator can change the limit (PV) for high warning
15	1 = Operator can change the limit (PV) for high tolerance
16	1 = The operator can change the limit (PV) for hysteresis
17	1 = The operator can change the limit (PV) for the low tolerance
18	1 = Operator can change the limit (PV) for low warning
19	1 = Operator can change the limit (PV) for low alarm
20	1 = The operator can change the value for the high gradient limit for positive slopes (`GradHUpLim`)
21	1 = The operator can change the value for the high gradient limit for negative slopes (`GradHDnLim`)
22	1 = The operator can change the value for the low gradient limit for positive and negative slopes (`GradLLim`)
22 - 31	Not allocated

Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 46).

Specifying the display area for process and setpoint values as well as operations

This block provides the standard function Display and operator input area for process values and setpoints (Page 42).

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

Time stamp

This block receives a time stamp value via the EventTSIn input parameter. Refer to EventTs functions (Page 997) for more information on this.

SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 122).

See also

Description of MonAnL (Page 1217)
MonAnL messaging (Page 1227)
MonAnL I/Os (Page 1230)
MonAnL block diagram (Page 1235)
MonAnL error handling (Page 1226)
MonAnL modes (Page 1220)
Time stamp (Page 51)

12.2.4 MonAnL error handling

MonAnL troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following error message can be generated at this block:

- Error numbers
- Control System Fault (CSF)

Overview of error numbers

The `ErrorNum` I/O can be used to output various error numbers:

Error number	Meaning of the error number
-1	Predefined value when inserting the block, the block is not processed.
0	There is no error.
30	The value of PV can no longer be displayed in the REAL number field.
43	`TimeFactor < 0` or `TimeFactor > 2`

Control System Fault (CSF)

An external signal can be activated via the CSF input. A control system fault is triggered if this signal changes to 1. Refer to the Error handling (Page 117) section for more on this.

See also

MonAnL block diagram (Page 1235)

MonAnL I/Os (Page 1230)

MonAnL messaging (Page 1227)

MonAnL functions (Page 1221)

MonAnL modes (Page 1220)

Description of MonAnL (Page 1217)

12.2.5 MonAnL messaging

Messaging

The following messages can be generated for this block:

- Control system fault
- Process messages
- Instance-specific messages

Control system fault

You can interconnect an external fault (signal) to input parameter CSF. If this signal changes to CSF = 1, a control system fault is triggered (MsgEvId1, SIG 3).

Process messages

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 1	Alarm - high	$$BlockComment$$ PV - High alarm limit violated
	SIG 2	Warning - high	$$BlockComment$$ PV - High warning limit violated
	SIG 3	Tolerance - high	$$BlockComment$$ PV - High tolerance limit violated
	SIG 4	Tolerance - low	$$BlockComment$$ PV - Low tolerance limit violated
	SIG 5	Warning - low	$$BlockComment$$ PV - Low warning limit violated
	SIG 6	Alarm - low	$$BlockComment$$ PV - Low alarm limit violated
	SIG 7	Alarm - high	$$BlockComment$$ Limit (high) for positive gradients violated
	SIG 8	Alarm - high	$$BlockComment$$ Limit (high) for negative gradients violated

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Instance-specific messages

You can use up to three instance-specific messages with this block.

Message instance	Message identifier	Message class	Event
MsgEvId2	SIG 1	Alarm - low	$$BlockComment$$ Limit (low) for absolute gradients violated
	SIG 2	AS process control message - fault	$$BlockComment$$ External error has occurred
	SIG 3	AS process control message - fault	$$BlockComment$$ External message 1
	SIG 4	AS process control message - fault	$$BlockComment$$ External message 2
	SIG 5	AS process control message - fault	$$BlockComment$$ External message 3

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Associated values for message instance MsgEvId1

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`
4	`ExtVa104`
5	`ExtVa105`
6	`ExtVa106`
7	`ExtVa107`
8	`ExtVa108`
9	Reserved
10	Reserved

The associated values 4 to 8 are allocated to the parameters `ExtVa104` ... `ExtVa108` and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

Associated values for message instance MsgEvId2

Associated value	Block parameters
1	BatchName
2	StepNo
3	BatchID
4	ExtVa204
5	ExtVa205
6	ExtVa206
7	ExtVa207
8	ExtVa208
9	Reserved
10	Reserved

The associated values 4 to 8 are allocated to the parameters ExtVa204...ExtVa208 and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

See also

Description of MonAnL (Page 1217)

MonAnL functions (Page 1221)

MonAnL I/Os (Page 1230)

MonAnL block diagram (Page 1235)

MonAnL error handling (Page 1226)

MonAnL modes (Page 1220)

Time stamp (Page 51)

12.2.6 MonAnL I/Os

MonAnL I/Os

Input parameters

Parameter	Description	Type	Default
BatchEn	1 = Enable allocation	BOOL	0
BatchID	Batch ID	DWORD	16#00000000
BatchName	Batch name	STRING[32]	''
CSF	1 = External error (control system error)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
DeadBand	Width of dead band	REAL	0.0
EN	1 = Called block will be processed	BOOL	1
EventTSIn	For wiring the signal status of an EventTs message block: The EventTsIn input parameter serves to interconnect the EventTsOut output parameter of the EventTs block. When this interconnection is configured, the messages of the EventTs block are displayed on the OS in the message view of the technological block and can also be acknowledged there.	STRUCT • Value: BYTE • ST: BYTE	- • 16#00 • 16#FF
ExtMsg1	Binary input for freely selectable message 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg2	Binary input for freely selectable message 2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg3	Binary input for freely selectable message 3	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtVa104	Associated value 4 for messages (MsgEvID1)	ANY	
ExtVa105	Associated value 5 for messages (MsgEvID1)	ANY	
ExtVa106	Associated value 6 for messages (MsgEvID1)	ANY	
ExtVa107	Associated value 7 for messages (MsgEvID1)	ANY	
ExtVa108	Associated value 8 for messages (MsgEvID1)	ANY	
ExtVa204	Associated value 4 for messages (MsgEvID2)	ANY	
ExtVa205	Associated value 5 for messages (MsgEvID2)	ANY	
ExtVa206	Associated value 6 for messages (MsgEvID2)	• ANY	
ExtVa207	Associated value 7 for messages (MsgEvID2)	• ANY	
ExtVa208	Associated value 8 for messages (MsgEvID2)	ANY	

Parameter	Description	Type	Default
Feature	I/O for additional functions (Page 1221)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
GradHUpLim	Gradient monitoring: Limit (high) for positive gradients	REAL	10.0
GradHDnLim	Gradient monitoring: Limit (high) for negative gradients	REAL	10.0
GradLLim	Gradient monitoring: Limit (low) for absolute gradients	REAL	1.0
GradHUpEn	1 = Activate gradient monitoring (high) for positive changes	BOOL	1
GradHDnEn	1 = Activate gradient monitoring (high) for negative changes	BOOL	1
GradLEn	1 = Activate gradient monitoring (low)	BOOL	1
GradHUpMsgEn	1 = Activate gradient (high) message for positive changes	BOOL	1
GradHDnMsgEn	1 = Activate gradient (high) message for negative changes	BOOL	1
GradLMsgEn	1 = Activate gradient (low) message	BOOL	0
LagTime	Delay time [s]	REAL	1.0
MS_RelOp	1 = Maintenance release by OS operator	BOOL	0
MsgEvID1	Message number (assigned automatically)	DWORD	16#FF
MsgEvID2	Message number (assigned automatically)	DWORD	16#FF
MsgLock	1 = Suppress process messages	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Occupied	1 = Occupied by batch control	BOOL	0
OnOp	1 = "On" mode via operator	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC (0-1 edge transition)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OS_Perm	I/O for operator control permissions (Page 1221)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
PV	Process value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_A_DC	Delay time for incoming PV alarms [s]	REAL	0.0
PV_A_DG	Delay time for outgoing PV alarms [s]	REAL	0.0
PV_AH_En	1 = Enable PV alarm limit (high)	BOOL	1

Parameter	Description	Type	Default
PV_AH_Lim	PV alarm limit (high)	REAL	95.0
PV_AH_MsgEn	1 = Enable message for PV alarm (high)	BOOL	1
PV_AL_En	1 = Enable PV alarm limit (low)	BOOL	1
PV_AL_Lim	PV alarm limit (low)	REAL	5.0
PV_AL_MsgEn	1 = Enable message for PV alarm (low)	BOOL	1
PV_Hyst	Hysteresis for PV alarm, warning and tolerance limits	REAL	0.0
PV_OpScale	Limit for scale in PV bar graph of faceplate	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
PV_T_DC	Delay time for incoming PV tolerance messages [s]	REAL	0.0
PV_T_DG	Delay time for outgoing PV tolerance messages [s]	REAL	0.0
PV_TH_En	1 = Enable PV tolerance limit (high)	BOOL	0
PV_TH_Lim	PV tolerance message limit (high)	REAL	85.0
PV_TH_MsgEn	1 = Enable message for PV tolerance message (high)	BOOL	1
PV_TL_En	1 = Enable PV tolerance limit (low)	BOOL	0
PV_TL_Lim	PV tolerance message limit (low)	REAL	15.0
PV_TL_MsgEn	1 = Enable message for PV tolerance message (low)	BOOL	1
PV_Unit	Unit of measure for process value	INT	1001
PV_W_DC	Delay time for incoming PV warnings [s]	REAL	0.0
PV_W_DG	Delay time for outgoing PV warnings [s]	REAL	0.0
PV_WH_En	1 = Enable PV warning limit (high)	BOOL	1
PV_WH_Lim	PV warning limit (high)	REAL	90.0
PV_WH_MsgEn	1 = Enable message for PV warning (high)	BOOL	1
PV_WL_En	1 = Enable PV warning limit (low)	BOOL	1
PV_WL_Lim	PV warning limit (low)	REAL	10.0
PV_WL_MsgEn	1 = Enable message for PV warning (low)	BOOL	1
RstOp	1 = Operator reset of the gradient's peak values	BOOL	0
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SelFp1	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
SimOn	1 = Simulation on	BOOL	0
SimPV	Process value used for SimOn = 1	REAL	0.0

Parameter	Description	Type	Default
StepNo	Batch step number	DWORD	16#00000000
TimeFactor	Time unit: • 0 = Seconds • 1 = Minutes • 2 = Hours	INT	0
UA1unit	Unit of measure for analog auxiliary value 1	INT	0
UA2unit	Unit of measure for analog auxiliary value 2	INT	0
UserAna1	Analog auxiliary value 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
UserAna2	Analog auxiliary value 2	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
UserStatus	Freely assignable bits for use in faceplate	BYTE	16#00

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see MonAnL error handling (Page 1226)	INT	-1
GradHUpAct	1 = Activate gradient alarm (high) for positive changes	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
GradHDnAct	1 = Activate gradient alarm (high) for negative changes	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
GradLAct	1 = Low gradient alarm	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_Release	Maintenance release: 1 = Release for OS operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MsgAckn1	Alarm acknowledgement status 1 (output STATUS of first ALARM_8P)	WORD	16#0000
MsgAckn2	Alarm acknowledgement status 2 (output STATUS of second ALARM_8P)	WORD	16#0000
MsgErr1	1 = Alarm error 1 (output ERROR of first ALARM_8P)	BOOL	0
MsgErr2	1 = Alarm error 2 (output ERROR of second ALARM_8P)	BOOL	0

Parameter	Description	Type	Default
MsgStat1	Alarm status 1 (output ERROR of first ALARM_8P)	WORD	16#0000
MsgStat2	Alarm status 2 (output ERROR of second ALARM_8P)	WORD	16#0000
OnAct	1 = "On" mode enabled	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
OosAct	1 = Block is "Out of service"	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
PV_AH_Act	1 = PV alarm (high) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_AL_Act	1 = PV alarm (low) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_Grad	Gradient value.	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_GradPP	Gradient maximum peak value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_GradNP	Gradient minimum peak value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_Out	Output for process value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV_TH_Act	1 = PV tolerance message (high) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_TL_Act	1 = PV tolerance message (low) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
PV_WH_Act	1 = PV warning (high) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
PV_WL_Act	1 = PV warning (low) active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word 1 (Page 1217)	DWORD	16#00000000
Status2	Status word 2 (Page 1217)	DWORD	16#00000000
Status3	Status word 3 (Page 1217)	DWORD	16#00000000
SumMsgAct	1 = Group message is active	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

MonAnL messaging (Page 1227)
MonAnL block diagram (Page 1235)
MonAnL modes (Page 1220)

12.2.7 MonAnL block diagram

MonAnL block diagram

A block diagram is not provided for this block.

See also

MonAnL I/Os (Page 1230)
MonAnL messaging (Page 1227)
MonAnL error handling (Page 1226)
MonAnL functions (Page 1221)
MonAnL modes (Page 1220)
Description of MonAnL (Page 1217)

12.2.8 Operator control and monitoring

12.2.8.1 Views of MonAnL

Views of the MonAnL block

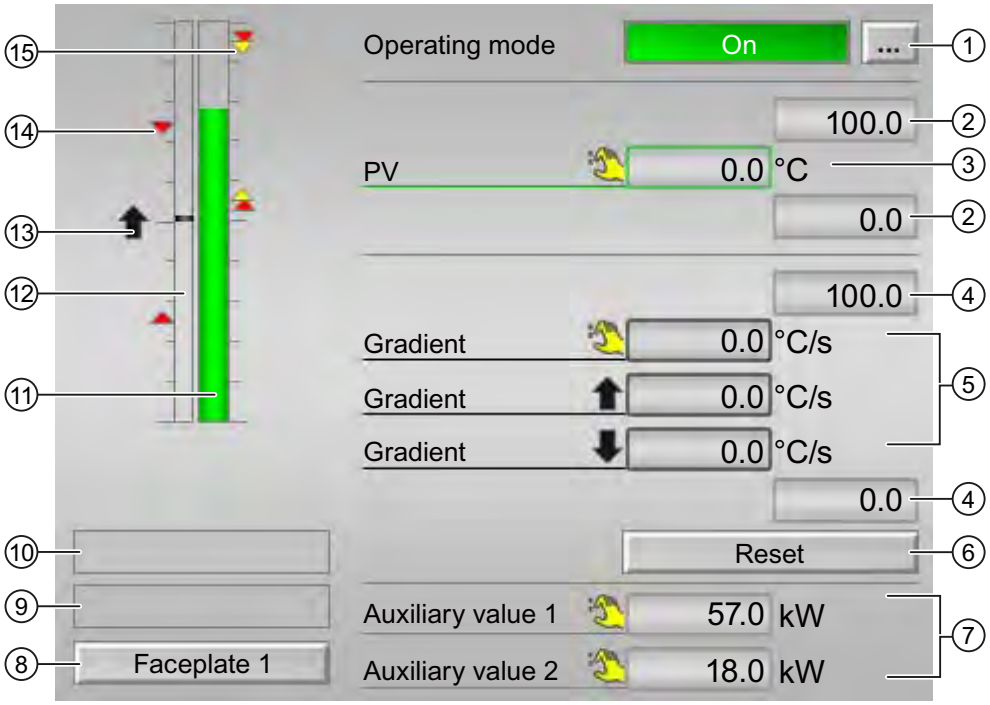
The block MonAnL provides the following views:

- MonAnL standard view (Page 1237)
- Message view (Page 169)
- MonAnL limit value view (Page 1240)
- Trend view (Page 172)
- MonAnL parameter view (Page 1242)
- MonAnL preview (Page 1243)
- Memo view (Page 171)
- Batch view (Page 170)
- Block symbol for MonAnL (Page 1244)

Refer to the Structure of the faceplate (Page 123) and Block icon structure (Page 174) sections for general information about the faceplate and block icon.

12.2.8.2 MonAnL standard view

MonAnL standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 27)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) High and low scale range for the process value

These values provide information on the display range for the bar graph of the process value. The scale range is defined in the engineering system.

(3) Display of the process value including signal status

This area shows the current process value with the corresponding signal status.

(4) High and low scale range for the gradient

These values provide information on the display range for the bar graph of the gradient. The scale range corresponds to 10% of the scale range of the process value: For example, once you have specified a process value scale range of 0 to 100, the scale range of the gradient will be automatically set to a value between 0 and 10.

(5) Gradient display

This area shows the current gradient value and the rise and fall of the value. This display functions like a slave pointer.

(6) Reset the peak values of the gradient

You can use this button to reset the maximum or minimum peak value of the gradient (`PV_GradPP` and `PV_GradNP` output parameters).

(7) Display of auxiliary values

You can use this area to display two auxiliary values that have been configured in the engineering system. You can find additional information on this in the Displaying auxiliary values (Page 52) section.

(8) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

(9) Display area for states of the block

This area provides additional information on the operating state of the block:

- Maintenance

(10) Display area for states of the block

This area provides additional information on the operating state of the block:

- Simulation

(11) Bar graph for the process value

This area shows the current process value in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(12) Bar graph for the gradient

This area shows the current gradient value in the form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

(13) Gradient display

This display indicates the movement of the gradient up (↑) or down (↓).

(14) Display for limits in the bar graph

This area shows you the specified limits. Refer to the MonAnL limit value view (Page 1240) section for more on this.

(15) Limit display

These colored triangles show you the configured limits in the respective bar graph.

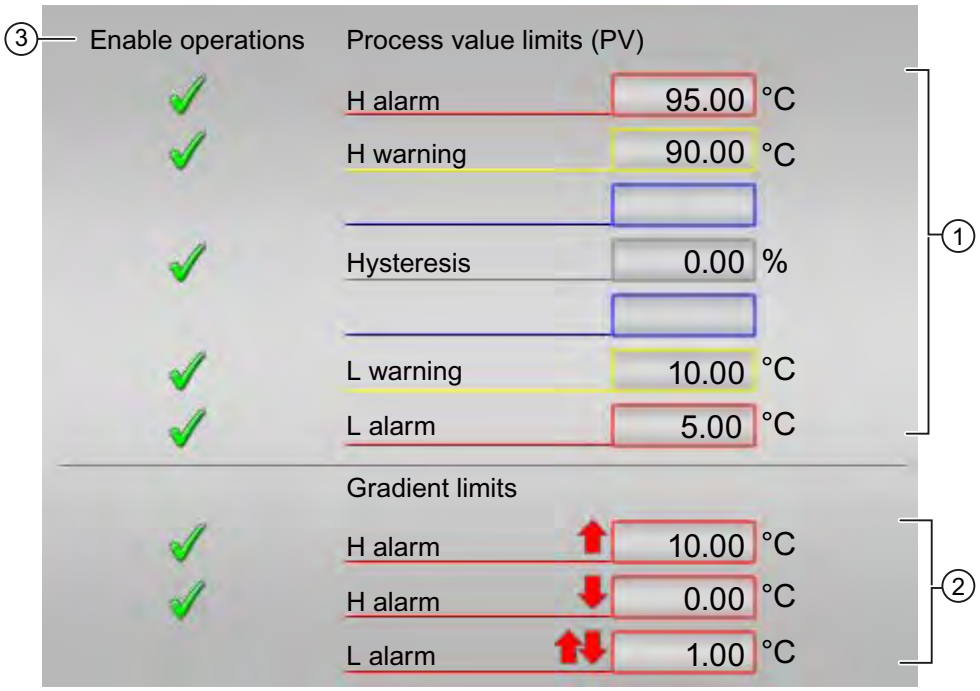
12.2.8.3 MonAnL limit value view

MonAnL limit value view

Several values are set in this view by default:

- Process value limits
- Gradient limits

The toolbars of the faceplate and the block icon indicate when the limit(s) is reached or exceeded.



(1) Process value limits

In this area, you can enter the limits for the process value. Refer to the Changing values (Page 129) section for more on this.

You can change the following limits:

- H alarm: Alarm high
- H warning: Warning high
- H tolerance: Tolerance high
- Hysteresis
- L tolerance: Tolerance low
- L warning: Warning low
- L alarm: Alarm low

(2) Gradient limits

You can enter the gradient limits in this area. Refer to the Changing values (Page 129) section for more on this.

You can change the following limits:

- H alarm ⇑: Gradient for the high slope for positive changes
- H alarm ⇓: Gradient for the high slope for negative changes
- L alarm ⇑⇓: Gradient for the low slope (absolute)

(3) Enable operations

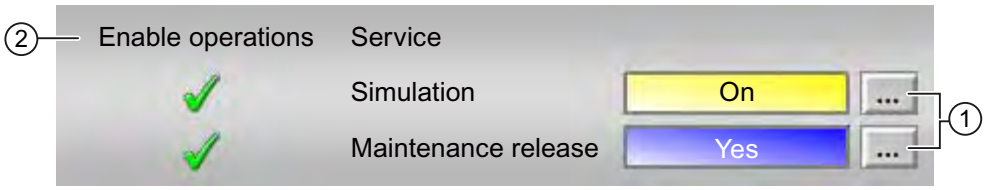
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions

12.2.8.4 MonAnL parameter view

MonAnL parameter view



(1) Service

You can select the following functions in this area:

- Simulation
- Maintenance release

Refer to the Switching operating states and operating modes (Page 127) section for more on this.

You can find information on this area in the section:

- Simulating signals (Page 93)
- Maintenance release (Page 47)

(2) Enable operations

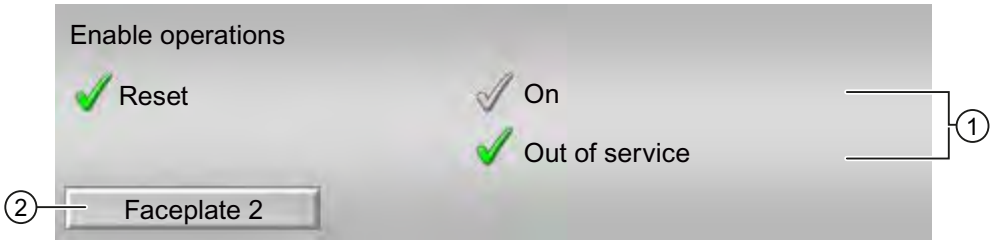
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions.

12.2.8.5 MonAnL preview

MonAnL preview



① Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions

The following operator control enables are shown here:

- Reset: You can reset block errors.
- On: You can switch to On operating mode.
- Out of service: You can switch to out of service mode.

⑪ Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

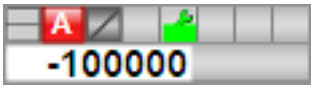
You can find additional information on this in the Calling further faceplates (Page 43) section.

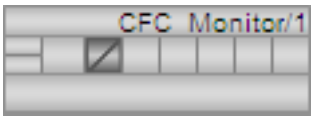
12.2.8.6 Block symbol for MonAnL

Properties of the MonAnL block icon

A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as the control system fault CSF
- Operating modes
- Signal status, maintenance release
- Memo display
- Process value

Icons	Selection of the block icon in CFC	Special features
	1	Block icon in the full display
	2	
	3	
	4	
	5	
	6	
	7	
	8	

Icons	Selection of the block icon in CFC	Special features
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181).

12.3 MonDiL - Monitoring of a digital process tag

12.3.1 Description of MonDiL

Object name (type + number) and family

Type + number: FB 1848

Family: Monitor

Area of application for MonDiL

The block is used for the following applications:

- Monitoring a digital process tag

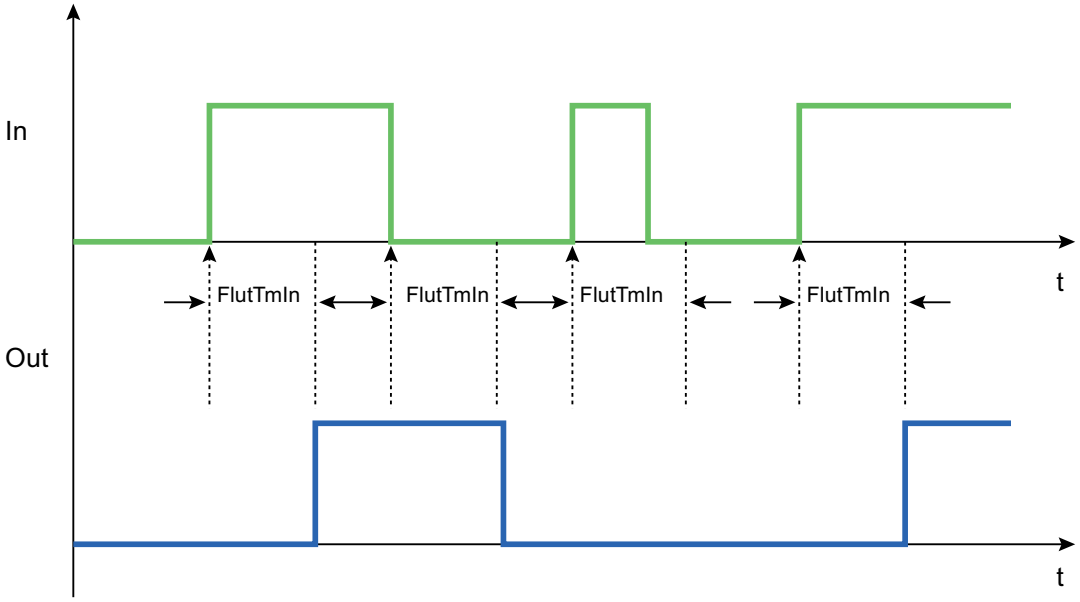
How it works

The MonDiL block is used to monitor a digital process tag with flutter suppression. The block reports excess flutter signals which are generated within a defined period.

The digital value to be monitored is interconnected to the `In` input parameter. Every time there is a signal change (1-0 or 0-1), the configurable timer (`FlutTmIn`) for flutter suppression is started, as shown in the figure below.

When the time you have specified expires and no single change occurs, the input signal is written to the `Out` output parameter.

Set the time in the timer (`FlutTmIn`) to 0 seconds; the input signal is written directly to the output.



Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

For the MonDiL block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Monitoring of a digital process tag (DigitalMonitoring) (Page 1448)

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block. The `Out` and `FlutAct` parameters are affected.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

Status word allocation for `Status1` parameter

For a description of the individual parameters, see the section MonDiL I/Os (Page 1257).

Status bit	Parameter
0	`Occupied`
1	`BatchEn`
2	`SimOn`
3	`OosAct.Value`
4	`OosLi.Value`
5	Not used
6	`OnAct.Value`
7	`Out.Value`
8	`AlmMsgEn`
9 - 29	Not used
30	Auxiliary value 1 is visible
31	Auxiliary value 2 is visible

Status word allocation for `Status2` parameter

Status bit	Parameter
0	`MsgLock.Value`
1	`FlutAct.Value`
2	`FlutEn`
3	`FlutMsgEn`
4 - 30	Not used
31	`MS_RelOp`

Status word allocation for `Status3` parameter

Status bit	Parameter
0	Effective `Signal1` of the message block connected via `EventTsIn`
1	Effective `Signal2` of the message block connected via `EventTsIn`
2	Effective `Signal3` of the message block connected via `EventTsIn`
3	Effective `Signal4` of the message block connected via `EventTsIn`
4	Effective `Signal5` of the message block connected via `EventTsIn`
5	Effective `Signal6` of the message block connected via `EventTsIn`
6	Effective `Signal7` of the message block connected via `EventTsIn`
7	Effective `Signal8` of the message block connected via `EventTsIn`
8 - 31	Not used

See also

MonDiL functions (Page 1251)
MonDiL messaging (Page 1255)
MonDiL block diagram (Page 1260)
MonDiL error handling (Page 1254)
MonDiL modes (Page 1250)

12.3.2 MonDiL modes

MonDiL operating modes

This block provides the following modes.

- On (Page 27)
- Out of service (Page 27)

"On"

You can find general information about the "On" mode in the On (Page 27) section.

Out of service

You can find general information about the "Out of service" mode in the Out of service (Page 27) section.

See also

MonDiL block diagram (Page 1260)

MonDiL I/Os (Page 1257)

MonDiL messaging (Page 1255)

MonDiL error handling (Page 1254)

MonDiL functions (Page 1251)

Description of MonDiL (Page 1246)

12.3.3 MonDiL functions

Functions of MonDiL

The functions for this block are listed below.

Suppression and reporting of signal flutter

The block is operated as "flutter filter". The block receives digital signals at the `In` input parameter and ideally these do not develop any flutter. The block monitors this if you activate the function via the `FlutEn = 1` input parameter.

Use the `FlutTmIn` input parameter to set how long a continuous signal should last in order to be transferred to the process without flutter.

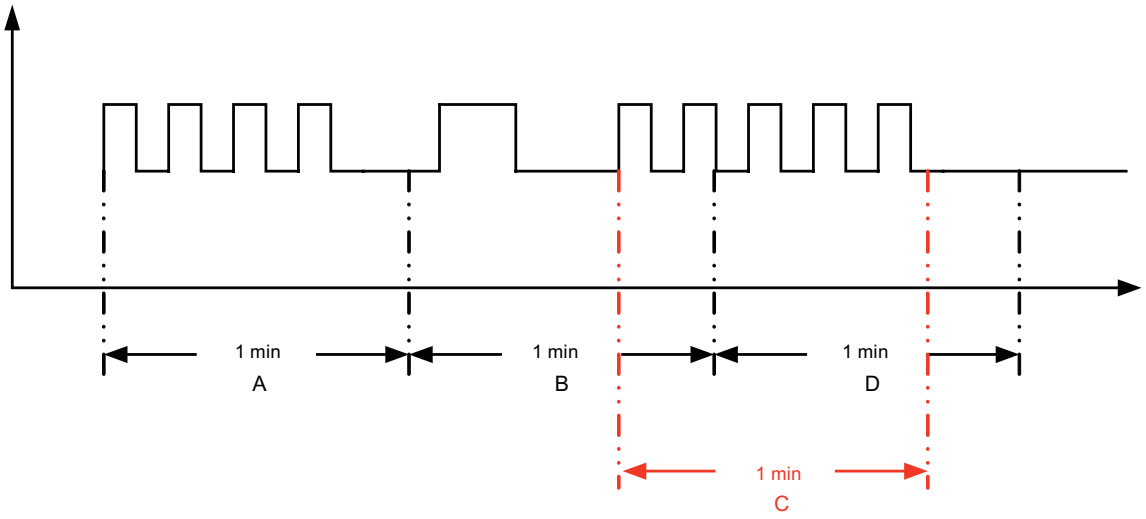
Output parameter `Out` transfers the "clean" signal to the process.

The block can be set up to report signal flutter. To do this, set parameter `FlutMsgEn = 1`. Monitoring starts at the next 0-1-0 edge transition at the input signal `In`.

Use the `FlutFactor` input parameter to specify the maximum number of signals to be filtered per minute by the block. If this maximum value is reached or exceeded, this is indicated by a 1 at output parameter `FlutAct`. The message is cleared (`FlutAct = 0`) after the maximum value has dropped again by more than half.

Example of the flutter suppression

You can use the `FlutFactor` input parameter to limit flutter signals to 4 per minute.



Case A:

The number of flutter signals is 4 per minute and the same as the specified number. Therefore, no alarm is generated.

Case B:

The number of flutter signals is 3 per minute and lower than the specified number. Therefore, no alarm is generated.

Case C:

The number of flutter signals is 5 per minute and higher than the specified number. Therefore, an alarm is generated.

Case D:

The number of flutter signals is 3 per minute and lower than the specified number. Therefore, no alarm is generated.

Displaying auxiliary values

This block provides the standard function Displaying auxiliary values (Page 52).

Generating instance-specific messages

This block provides the standard function Generating instance-specific messages (Page 46).

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `In.ST`

Maintenance release

This block provides the standard function Maintenance release (Page 47).

Simulating signals

This block provides the standard function Simulating signals (Page 93).

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)
1	Reaction to the out of service mode (Page 199)
22	Update acknowledgment and error status of the message call (Page 193)

Operator control permissions

The block has the following Operator permissions (Page 45) for the `OS_Perm` parameter:

Bit	Function
0	Not allocated
1	1 = Operator can switch to "On" mode
2	Not allocated
3	1 = Operator can switch to "Out of service" mode
4 - 10	Not allocated
11	1 = Operator can enable function simulation
12	1 = Operator can enable the maintenance release function
13	Not allocated
14	1 = The user must input the duration for flutter suppression.
15	1 = The user must input the number of flutter signals that have to be suppressed.
16 - 31	Not allocated

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

Time stamp

This block receives a time stamp value via the `EventTSIn` input parameter. Refer to EventTs functions (Page 997) for more information on this.

SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 122).

See also

Description of MonDiL (Page 1246)

MonDiL messaging (Page 1255)

MonDiL I/Os (Page 1257)

MonDiL block diagram (Page 1260)

MonDiL error handling (Page 1254)

MonDiL modes (Page 1250)

Time stamp (Page 51)

Selecting a unit of measure (Page 53)

12.3.4 MonDiL error handling

MonDiL troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers
- Control System Fault (CSF)
- Flutter alarm

Overview of error numbers

The `ErrorNum` I/O can be used to output various error numbers:

Error number	Meaning of the error number
-1	Predefined value when inserting the block, the block is not processed.
0	There is no error.
11	`FlutTmIn` < 0
16	`FlutFactor` is -ve or > 100

Control System Fault (CSF)

An external signal can be activated via the CSF input. A control system fault is triggered if this signal changes to 1. Refer to the Error handling (Page 117) section for more on this.

Flutter alarm

An alarm is output with 1-signal at output FlutAct if signal flutter is detected. Refer to the functions of the block > Suppression and reporting of signal flutter (Page 1251).

See also

MonDiL block diagram (Page 1260)

MonDiL I/Os (Page 1257)

MonDiL messaging (Page 1255)

MonDiL modes (Page 1250)

Description of MonDiL (Page 1246)

12.3.5 MonDiL messaging

Messaging

The following messages can be generated for this block:

- Control system fault
- Process messages
- Instance-specific messages

Control system fault

You can interconnect an external fault (signal) to input parameter CSF. If this signal changes to CSF = 1, a control system fault is triggered (MsgEvId1, SIG 3).

Process messages

You can use up to four instance-specific messages with this block.

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 1	Alarm - high	$$BlockComment$$ Out - Binary value set
	SIG 2	Warning - high	$$BlockComment$$ Flutter limits violated
	SIG 3	AS process control message - fault	$$BlockComment$$ External error has occurred
	SIG 4	AS process control message - fault	$$BlockComment$$ External message 1
	SIG 5	AS process control message - fault	$$BlockComment$$ External message 2
	SIG 6	AS process control message - fault	$$BlockComment$$ External message 3

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Associated values for reporting instance MsgEvId1

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`
4	`ExtVa104`
5	`ExtVa105`
6	`ExtVa106`
7	`ExtVa107`
8	`ExtVa108`
9	Reserved
10	Reserved

The associated values 4 to 8 are allocated to the parameters `ExtVa104` ... `ExtVa108` and you can use them. Additional information is available in the "Process Control System PCS 7 - Engineering System" manual.

See also

Description of MonDiL (Page 1246)

MonDiL functions (Page 1251)

MonDiL I/Os (Page 1257)

MonDiL block diagram (Page 1260)

MonDiL error handling (Page 1254)

MonDiL modes (Page 1250)

Time stamp (Page 51)

12.3.6 MonDiL I/Os

MonDiL I/Os

Input parameters

Parameter	Description	Type	Default
AlmMsgEn	1 = Alarms are output.	BOOL	1
BatchEn	1 = Enable allocation	BOOL	0
BatchID	Batch ID	DWORD	16#00000000
BatchName	Batch name	STRING[32]	''
CSF	1 = External error (control system error)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
EN	1 = Called block will be processed	BOOL	1
EventTSIn	For wiring the signal status of an EventTs message block: The EventTsIn input parameter serves to interconnect the EventTsOut output parameter of the EventTs block. When this interconnection is configured, the messages of the EventTs block are displayed on the OS in the message view of the technological block and can also be acknowledged there.	STRUCT • Value: BYTE • ST: BYTE	- • 16#00 • 16#FF
ExtMsg1	Binary input for freely selectable message 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg2	Binary input for freely selectable message 2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtMsg3	Binary input for freely selectable message 3	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ExtVa104	Associated value 4 for messages (MsgEvID1)	ANY	
ExtVa105	Associated value 5 for messages (MsgEvID1)	ANY	
ExtVa106	Associated value 6 for messages (MsgEvID1)	ANY	
ExtVa107	Associated value 7 for messages (MsgEvID1)	ANY	
ExtVa108	Associated value 8 for messages (MsgEvID1)	ANY	
Feature	I/O for additional functions (Page 1251)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
FlutEn	1 = Flutter suppression on	BOOL	1
FlutFactor	Number of flutter signals that can be suppressed	INT	2

Parameter	Description	Type	Default
FlutMsgEn	1 = Signal flutter is reported if the period of the signal flutter is < FlutTmIn .	BOOL	1
FlutTmIn	Period during which signal flutter is suppressed.	REAL	0.0
In	Digital input value	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_RelOp	1 = Maintenance release by OS operator	BOOL	0
MsgEvId1	Message number (assigned automatically)	DWORD	16#FFFFFFFF
MsgLock	1 = Suppress process messages	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Occupied	1 = Occupied by batch control	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC (0-1 edge transition)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OnOp	1 = "On" mode via operator	BOOL	0
OosOp	1 = Out of service, via OS operator	BOOL	0
OS_Perm	I/O for operator control permissions (Page 1251)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SelFp1	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
SimOn	1 = Simulation on	BOOL	0
SimIn	Value used for SimOn = 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
StepNo	Batch step number	DWORD	16#00000000
UserAna1	Analog auxiliary value 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
UA1unit	Unit of measure for analog auxiliary value 1	INT	0
UserAna2	Analog auxiliary value 2	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#FF
UA2unit	Unit of measure for analog auxiliary value 2	INT	0
UserStatus	Freely assignable bits for use in faceplate	BYTE	16#00

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see MonDiL error handling (Page 1254)	INT	-1
FlutAct	1 = Suppresses flutter signal	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_Release	Maintenance release: 1 = Release for OS operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MsgAckn1	Alarm acknowledgement status 1 (output STATUS of first ALARM_8P)	WORD	16#0000
MsgErr1	1 = Alarm error 1 (output ERROR of first ALARM_8P)	BOOL	0
MsgStat1	Alarm status 1 (output ERROR of first ALARM_8P)	WORD	16#0000
OnAct	1 = "On" mode enabled	BOOL	1
OosAct	1 = Block is "Out of service"	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
Out	Output	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word 1 (Page 1246)	DWORD	16#00000000
Status2	Status word 2 (Page 1246)	DWORD	16#00000000
Status3	Status word 3 (Page 1246)	DWORD	16#00000000

See also

MonDiL messaging (Page 1255)

MonDiL block diagram (Page 1260)

MonDiL modes (Page 1250)

12.3.7 MonDiL block diagram

MonDiL block diagram

A block diagram is not provided for this block.

See also

MonDiL I/Os (Page 1257)
MonDiL messaging (Page 1255)
MonDiL error handling (Page 1254)
MonDiL functions (Page 1251)
MonDiL modes (Page 1250)
Description of MonDiL (Page 1246)

12.3.8 Operator control and monitoring

12.3.8.1 Views of MonDiL

Views of the MonDiL block

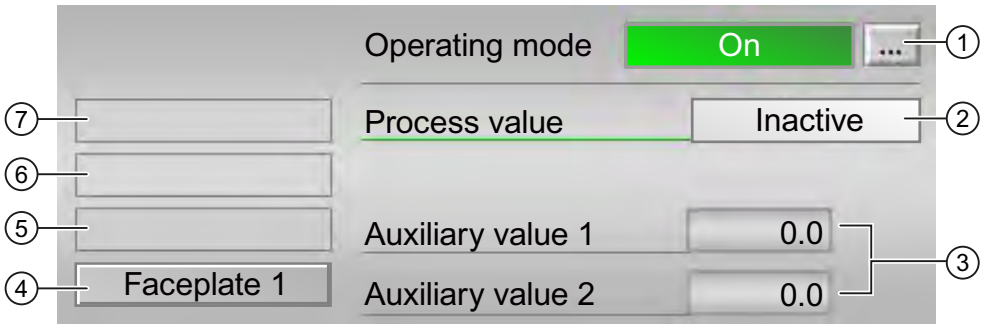
The block MonDiL provides the following views:

- MonDiL standard view (Page 1261)
- Message view (Page 169)
- Trend view (Page 172)
- MonDiL parameter view (Page 1263)
- MonDiL preview (Page 1265)
- Memo view (Page 171)
- Batch view (Page 170)
- Block symbol for MonDiL (Page 1266)

Refer to the sections Structure of the faceplate (Page 123) and Block icon structure (Page 174) for general information on the faceplate and block icon.

12.3.8.2 MonDiL standard view

MonDiL standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 27)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Display of process value enabled/disabled

If the block is in simulation, you can enable or disable the process value. To do this, click on the display to open the control field.

(3) Display of auxiliary values

You can use this area to display two auxiliary values that have been configured in the engineering system. You can find additional information on this in the Displaying auxiliary values (Page 52) section.

(4) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

(5) Display area for states of the block

This area provides additional information on the operating state of the block:

- Maintenance

(6) Display area for states of the block

This area provides additional information on the operating state of the block:

- Simulation

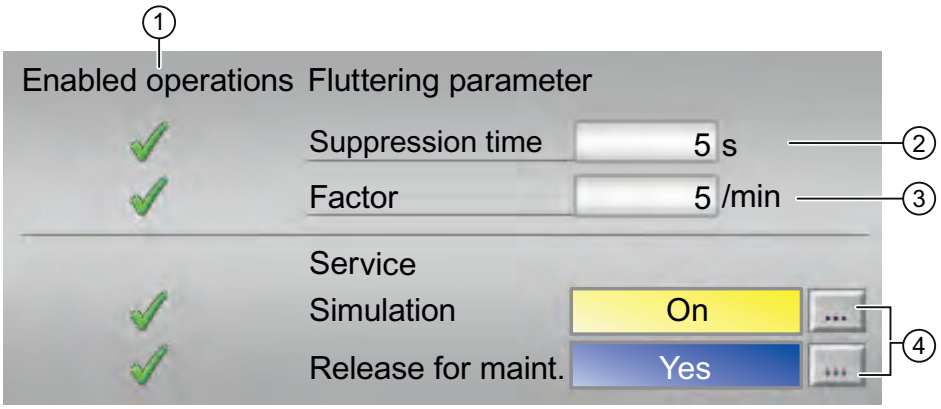
(7) Display area for states of the block

This area provides additional information on the operating state of the block:

- Flutter

12.3.8.3 MonDiL parameter view

MonDiL parameter view



(1) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions.

(2) Suppression time

Enter the time period during which signal flutter is suppressed. You can find additional information on this in the Changing values (Page 129) section.

(3) Factor

Enter the number of flutter signals that can be suppressed. You can find additional information on this in the Changing values (Page 129) section.

(4) Service

You can select the following functions in this area:

- Simulation
- Maintenance release

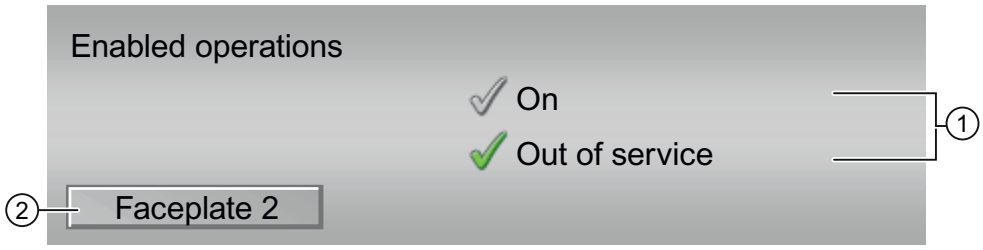
Refer to the Switching operating states and operating modes (Page 127) section for more on this.

You can find information on this area in the section:

- Simulating signals (Page 93)
- Maintenance release (Page 47)

12.3.8.4 MonDiL preview

MonDiL preview



(1) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions

The following operator control enables are shown here:

- On: You can switch to On operating mode.
- Out of service: You can switch to out of service mode.

(2) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

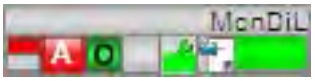
You can find additional information on this in the Calling further faceplates (Page 43) section.

12.3.8.5 Block symbol for MonDiL

Properties of the MonDiL block icon

A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as the control system fault CSF
- Operating modes
- Signal status, maintenance release
- Memo display
- Display of the output signal

Icons	Selection of the block icon in CFC	Special features
	1	Block icon in the full display
	2	
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181).

12.4 MonDi08 - Monitoring 8 digital process tags

12.4.1 Description of MonDi08

Object name (type + number) and family

Type + number: FB 1847

Family: Monitor

Area of application for MonDi08

The block is used for the following applications:

- Monitoring of up to eight digital process tags

How it works

The MonDi08 block is used to monitor up to eight digital process tags with flutter suppression.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

For the MonDi08 block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Monitoring eight digital process tags (Digital8Monitoring) (Page 1448)

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

The messages are suppressed after startup for the number of cycles set at `RunUpCyc`.

Status word allocation for `Status1` parameter

For a description of the individual parameters, see the section MonDi08 I/Os (Page 1276).

Status bit	Parameter
0 - 2	Not used
3	`OosAct.Value`
4	`OosLi.Value`
5	Not used
6	`OnAct.Value`
7	`Out1.Value`
8	`Out2.Value`
9	`Out3.Value`
10	`Out4.Value`
11	`Out5.Value`
12	`Out6.Value`
13	`Out7.Value`
14	`Out8.Value`
15 - 19	Not used
20	`In1` is used
21	`In2` is used
22	`In3` is used
23	`In4` is used
24	`In5` is used
25	`In6` is used
26	`In7` is used
27	`In8` is used
28 - 31	Not used

Status word allocation for `Status2` parameter

Status bit	Parameter
0	`MsgLock.Value`
1	`Alm1MsgEn`
2	`Alm2MsgEn`
3	`Alm3MsgEn`
4	`Alm4MsgEn`
5	`Alm5MsgEn`
6	`Alm6MsgEn`
7	`Alm7MsgEn`
8	`Alm8MsgEn`
9 - 30	Not used
31	`MS_RelOp`

Status word allocation for `Status3` parameter

Status bit	Parameter
0	Effective `Signal1` of the message block connected via `EventTsIn`
1	Effective `Signal2` of the message block connected via `EventTsIn`
2	Effective `Signal3` of the message block connected via `EventTsIn`
3	Effective `Signal4` of the message block connected via `EventTsIn`
4	Effective `Signal5` of the message block connected via `EventTsIn`
5	Effective `Signal6` of the message block connected via `EventTsIn`
6	Effective `Signal7` of the message block connected via `EventTsIn`
7	Effective `Signal8` of the message block connected via `EventTsIn`
8 - 31	Not used

See also

MonDi08 messaging (Page 1274)
MonDi08 functions (Page 1271)
MonDi08 block diagram (Page 1279)
MonDi08 error handling (Page 1273)
MonDi08 modes (Page 1270)

12.4.2 MonDi08 modes

MonDi08 operating modes

This block provides the following modes.

- On (Page 27)
- Out of service (Page 27)

"On"

You can find general information about the "On" mode in the On (Page 27) section.

Out of service

You can find general information about the "Out of service" mode in the Out of service (Page 27) section.

See also

MonDi08 block diagram (Page 1279)

MonDi08 I/Os (Page 1276)

MonDi08 messaging (Page 1274)

MonDi08 error handling (Page 1273)

MonDi08 functions (Page 1271)

Description of MonDi08 (Page 1267)

12.4.3 MonDi08 functions

Functions of MonDi08

The functions for this block are listed below.

Monitoring and output of digital signals

The block is operated as "flutter filter". The block receives digital signals at the input `Inx (x = 1 ... 8)` which ideally do not develop any flutter. The block monitors these signals. Use the `FlutTmInx (x = 1 ... 8)` input to determine the duration of a continuous signal in order to transfer it to the process without flutter.

The "clean" signal is transferred from output `Outx (x = 1 ... 8)` to the process.

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `In1.ST`
- `In2.ST`
- `In3.ST`
- `In4.ST`
- `In5.ST`
- `In6.ST`
- `In7.ST`
- `In8.ST`

Maintenance release

This block provides the standard function Maintenance release (Page 47).

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)
1	Reaction to the out of service mode (Page 199)
22	Update acknowledgment and error status of the message call (Page 193)

Operator control permissions

The block has the following Operator permissions (Page 45) for the `OS_Perm` parameter:

Bit	Function
0	Not allocated
1	1 = Operator can switch to "On" mode
2	Not allocated
3	1 = Operator can switch to "Out of service" mode
4 - 11	Not allocated
12	1 = Operator can enable the maintenance release function
13	1 = Operator can change the time for flutter suppression at the `In1` input
14	1 = Operator can change the time for flutter suppression at the `In2` input
15	1 = Operator can change the time for flutter suppression at the `In3` input
16	1 = Operator can change the time for flutter suppression at the `In4` input
17	1 = Operator can change the time for flutter suppression at the `In5` input
18	1 = Operator can change the time for flutter suppression at the `In6` input
19	1 = Operator can change the time for flutter suppression at the `In7` input
20	1 = Operator can change the time for flutter suppression at the `In8` input
21 - 31	Not allocated

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

Time stamp

This block receives a time stamp value via the `EventTSIn` input parameter. Refer to EventTs functions (Page 997) for more information on this.

See also

MonDi08 messaging (Page 1274)

MonDi08 I/Os (Page 1276)

Description of MonDi08 (Page 1267)

MonDi08 block diagram (Page 1279)

MonDi08 error handling (Page 1273)

MonDi08 modes (Page 1270)

12.4.4 MonDi08 error handling

MonDi08 troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers
- Flutter alarm

Overview of error numbers

The `ErrorNum` I/O can be used to output various error numbers:

Error number	Meaning of the error number
-1	Predefined value when inserting the block, the block is not processed.
0	There is no error.
11	Flutter time < 0

Flutter alarm

An alarm is output with 1-signal at output FlutAct if signal flutter is detected. Refer to the functions of the block > Monitoring and reporting flutter signals (Page 1271).

See also

MonDi08 block diagram (Page 1279)

MonDi08 I/Os (Page 1276)

MonDi08 messaging (Page 1274)

MonDi08 modes (Page 1270)

Description of MonDi08 (Page 1267)

12.4.5 MonDi08 messaging

Messaging

The following messages can be generated for this block:

- Process messages
- Instance-specific messages

Process messages

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 1	Alarm - high	$$BlockComment$$ Signal change Signal 1 has occurred
	SIG 2	Alarm - high	$$BlockComment$$ Signal change Signal 2 has occurred
	SIG 3	Alarm - high	$$BlockComment$$ Signal change Signal 3 has occurred
	SIG 4	Alarm - high	$$BlockComment$$ Signal change Signal 4 has occurred
	SIG 5	Alarm - high	$$BlockComment$$ Signal change Signal 5 has occurred
	SIG 6	Alarm - high	$$BlockComment$$ Signal change Signal 6 has occurred
	SIG 7	Alarm - high	$$BlockComment$$ Signal change Signal 7 has occurred
	SIG 8	Alarm - high	$$BlockComment$$ Signal change Signal 8 has occurred

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Associated values for message instance MsgEvId1

Associated value	Block parameters
1	`Out1.ST`
2	`Out2.ST`
3	`Out3.ST`
4	`Out4.ST`
5	`Out5.ST`
6	`Out6.ST`
7	`Out7.ST`
8	`Out8.ST`

See also

Description of MonDi08 (Page 1267)

MonDi08 functions (Page 1271)

MonDi08 I/Os (Page 1276)

MonDi08 block diagram (Page 1279)

MonDi08 error handling (Page 1273)

MonDi08 modes (Page 1270)

Time stamp (Page 51)

12.4.6 MonDi08 I/Os

MonDi08 I/Os

Input parameters

Parameter	Description	Type	Default
Alm1MsgEn	1 = Activate error message	BOOL	1
Alm2MsgEn	1 = Activate error message	BOOL	1
Alm3MsgEn	1 = Activate error message	BOOL	1
Alm4MsgEn	1 = Activate error message	BOOL	1
Alm5MsgEn	1 = Activate error message	BOOL	1
Alm6MsgEn	1 = Activate error message	BOOL	1
Alm7MsgEn	1 = Activate error message	BOOL	1
Alm8MsgEn	1 = Activate error message	BOOL	1
CSF	1 = External error (control system error)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
EN	1 = Called block will be processed	BOOL	1
EventTSIn	For wiring the signal status of an EventTs message block: The EventTsIn input parameter serves to interconnect the EventTsOut output parameter of the EventTs block. When this interconnection is configured, the messages of the EventTs block are displayed on the OS in the message view of the technological block and can also be acknowledged there.	STRUCT • Value: BYTE • ST: BYTE	- • 16#00 • 16#80
Feature	I/O for additional functions (Page 1271)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
FlutTmIn1	Flutter time [s]	REAL	0.0
FlutTmIn2	Flutter time [s]	REAL	0.0
FlutTmIn3	Flutter time [s]	REAL	0.0
FlutTmIn4	Flutter time [s]	REAL	0.0
FlutTmIn5	Flutter time [s]	REAL	0.0
FlutTmIn6	Flutter time [s]	REAL	0.0
FlutTmIn7	Flutter time [s]	REAL	0.0
FlutTmIn8	Flutter time [s]	REAL	0.0
In1	Binary input In1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
In2	Binary input In2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
In3	Binary input In3	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
In4	Binary input In4	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
In5	Binary input In5	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
In6	Binary input In6	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
In7	Binary input In7	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
In8	Binary input In8	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_RelOp	1 = Maintenance release by OS operator	BOOL	0
MsgEvId1	Message number (assigned automatically)	DWORD	16#FF
MsgLock	1 = Suppress process messages	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OnOp	1 = "On" mode via operator	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC (0-1 edge transition)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OS_Perm	I/O for operator control permissions (Page 1271)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SelFp1	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	BOOL	0

Parameter	Description	Type	Default
SelFp2	1 = Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	BOOL	0
UserStatus	User status bits	BYTE	16#00

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see MonDi08 error handling (Page 1273)	INT	-1
MS_Release	Maintenance release: 1 = Release for OS operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MsgAckn1	Alarm acknowledgement status 1 (output STATUS of first ALARM_8P)	WORD	16#0000
MsgErr1	1 = Alarm error 1 (output ERROR of first ALARM_8P)	BOOL	0
MsgStat1	Alarm status 1 (output ERROR of first ALARM_8P)	WORD	16#0000
OnAct	1 = "On" mode enabled	BOOL	0
OosAct	1 = Block is "Out of service"	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
Out1	Output	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Out2	Output	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Out3	Output	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Out4	Output	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Out5	Output	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
Out6	Output	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Out7	Output	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Out8	Output	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word 1 (Page 1267)	DWORD	16#00000000
Status2	Status word 2 (Page 1267)	DWORD	16#00000000
Status3	Status word 3 (Page 1267)	DWORD	16#00000000

See also

MonDi08 messaging (Page 1274)
MonDi08 block diagram (Page 1279)
MonDi08 modes (Page 1270)

12.4.7 MonDi08 block diagram

MonDi08 block diagram

A block diagram is not provided for this block.

See also

MonDi08 I/Os (Page 1276)
MonDi08 messaging (Page 1274)
MonDi08 error handling (Page 1273)
MonDi08 functions (Page 1271)
MonDi08 modes (Page 1270)
Description of MonDi08 (Page 1267)

12.4.8 Operator control and monitoring

12.4.8.1 Views of MonDi08

Views of the MonDi08 block

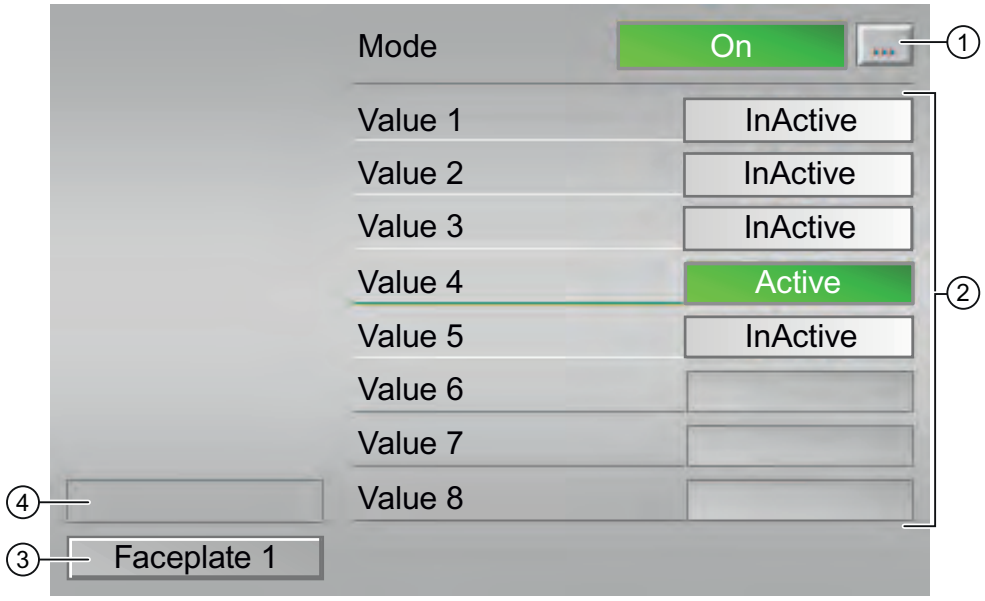
The block MonDi08 provides the following views:

- MonDi08 standard view (Page 1281)
- Message view (Page 169)
- Trend view (Page 172)
- MonDi08 parameter view (Page 1282)
- MonDi08 preview (Page 1284)
- Memo view (Page 171)
- Block symbol for MonDi08 (Page 1285)

Refer to the Structure of the faceplate (Page 123) and Block icon structure (Page 174) sections for general information about the faceplate and block icon.

12.4.8.2 MonDi08 standard view

MonDi08 standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 27)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Display of the status for each parameter

(3) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

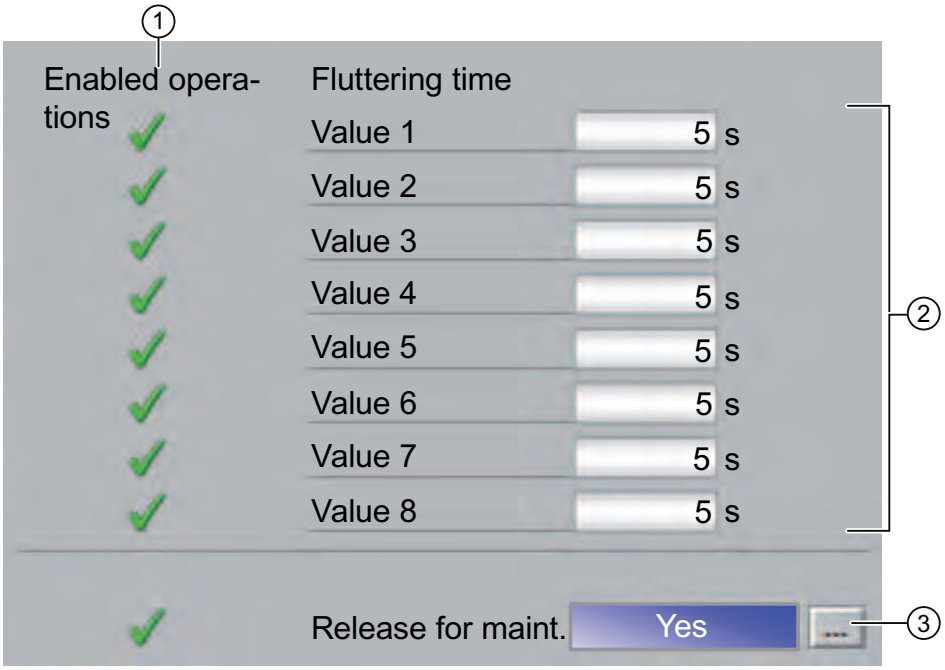
(4) Display area for states of the block

This area provides additional information on the operating state of the block:

- Maintenance

12.4.8.3 MonDi08 parameter view

MonDi08 parameter view



(1) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions.

(2) Area for entering flutter time

Use this area to set the time period to determine how long a continuous signal should last in order for it to be transferred to the process without flutter.

Refer to the Changing values (Page 129) section for more on this.

(3) Service

You can select the following function in this area:

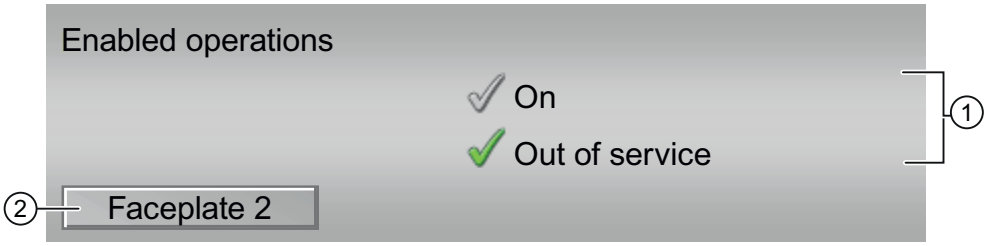
- Maintenance release

Refer to the Switching operating states and operating modes (Page 127) section for more on this.

You can find information on this area in the Maintenance release (Page 47) section.

12.4.8.4 MonDi08 preview

MonDi08 preview



(1) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions

The following operator control enables are shown here:

- On: You can switch to On operating mode.
- Out of service: You can switch to out of service mode.

(2) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

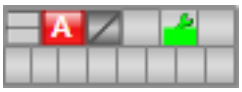
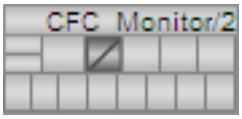
You can find additional information on this in the Calling further faceplates (Page 43) section.

12.4.8.5 Block symbol for MonDi08

Properties of MonDi08 block icon

A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits as well as the control system fault CSF
- Operating modes
- Signal status, maintenance release
- Memo display
- Display of the output signal

Icons	Selection of the block icon in CFC	Special features
	1	Block icon in the full display
	2	
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181).

Counter blocks

13

13.1 CountScL - Counter with up and down counting direction

13.1.1 Description of CountScL

Object name (type + number) and family

Type + number: FB 1806

Family: Count

Area of application for CountScL

The block is used for the following applications:

- Count up or count down with rising edge of the binary input signal.

How it works

Positive edges at the binary input signal In increment or decrement the count value at Out, depending on settings.

1. Count up (`UpOp` = 1 or `UpLi` =1)

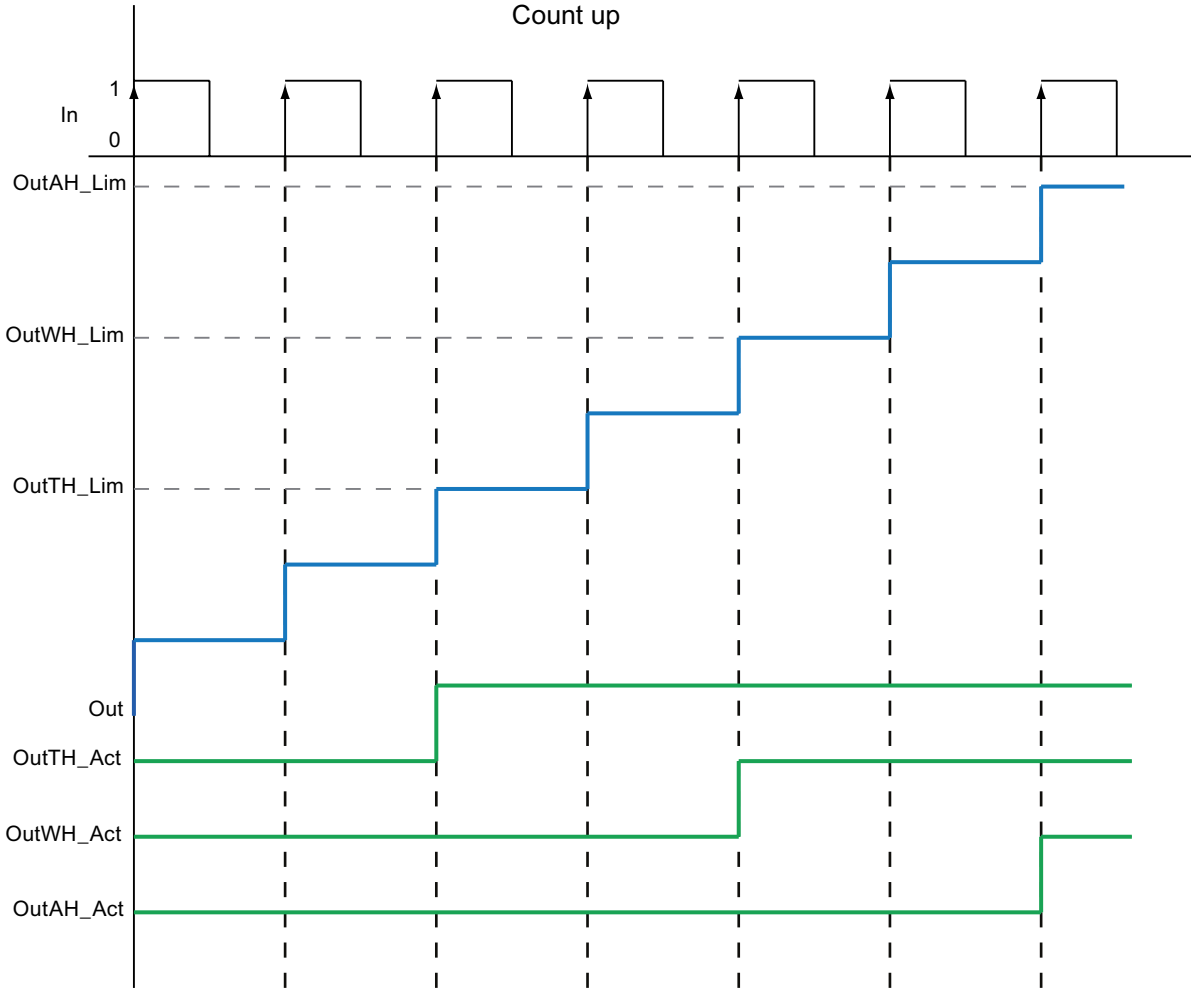
   The block counts up for each rising edge of `In`. (output parameter `CountMode` = 1)

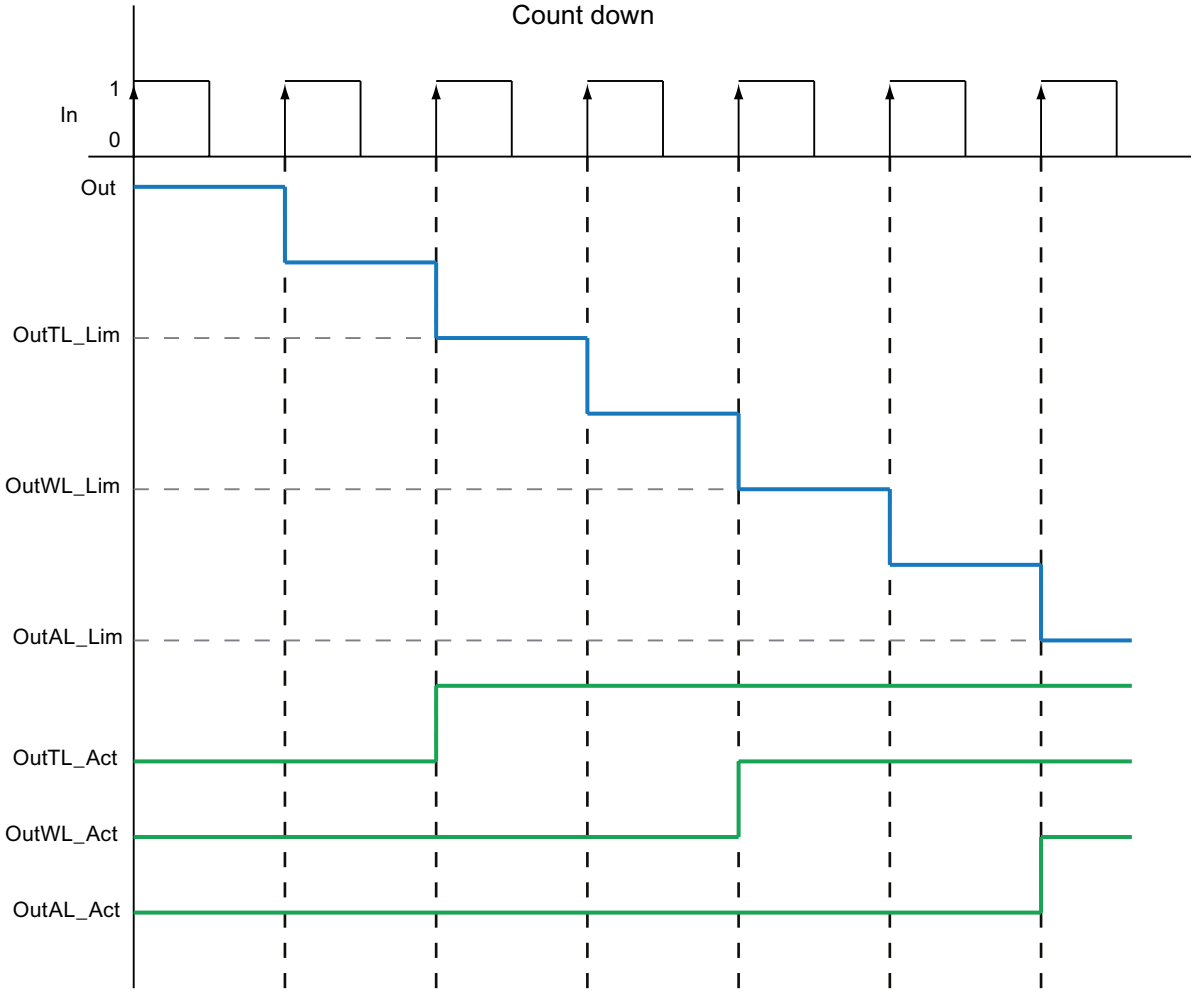
2. Count down (`DnOp` =1 or `DnLi` = 1)

   The block counts down for each rising edge of `In`. (output parameter `CountMode` = 2)

3. Off (`OffOp` = 1 or `OffLi` = 1)

   The block is disabled (output parameter `CountMode` = 0). No counting takes place.





Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

Time response

The block does not have any time response.

Status word allocation for Status1 parameter

Status bit	Parameter
0 - 2	Not used
3	OosAct.Value
4	OosLi.Value
5	Not used
6	OnAct.Value
7 - 10	Not used
11	LiOp
12 - 31	Not used

Status word allocation for Status2 parameter

Status bit	Parameter
0	MsgLock.Value
1	OutAH_Act.Value
2	OutWH_Act.Value
3	OutTH_Act.Value
4	OutTL_Act.Value
5	OutWL_Act.Value
6	OutAL_Act.Value
7	OutAH_En
8	OutWH_En
9	OutTH_En
10	OutTL_En
11	OutWL_En
12	OutAL_En
13	OutAH_MsgEn
14	OutWH_MsgEn
15	OutTH_MsgEn
16	OutTL_MsgEn
17	OutWL_MsgEn
18	OutAL_MsgEn
19	Not used
20	Count up
21	Counter off
22	Count down
23 - 30	Not used
31	MS_RelOp

See also

CountScL functions (Page 1293)
CountScL messaging (Page 1297)
CountScL I/Os (Page 1299)
CountScL block diagram (Page 1302)
CountScL error handling (Page 1296)
CountScL modes (Page 1292)

13.1.2 CountScL modes

CountScL operating modes

This block provides the following operating modes:

- On (Page 27)
- Out of service (Page 27)

"On"

General information on the "On" mode is available in the section On (Page 27).

Out of service

You can find general information about the "Out of service" mode in the section Out of service (Page 27).

See also

CountScL block diagram (Page 1302)

CountScL I/Os (Page 1299)

CountScL messaging (Page 1297)

CountScL error handling (Page 1296)

Description of CountScL (Page 1287)

CountScL functions (Page 1293)

13.1.3 CountScL functions

Functions of CountScL

The functions for this block are listed below.

Time measurement

The counts are displayed within the following limits:

1. Alarm (`OutAH_Lim` and `OutAL_Lim`)
2. Warning (`OutWH_Lim` and `OutWL_Lim`)
3. Tolerance (`OutTH_Lim` and `OutTL_Lim`)

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 53).

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status `ST_Worst` for the block is formed from the following parameter:

- `In.ST`

Configurable reactions using the Features I/O

You can find an overview of all reactions provided by the `Features` I/O in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)
1	Reaction to the out of service mode (Page 199)
22	Update acknowledgment and error status of the message call (Page 193)

Operator control permissions (OS Perm)

The block has the following Operator permissions (Page 45) for the `OS_Perm` parameter:

Bit	Function
0	Not used
1	1 = Operator can switch to "On" mode
2	Not used
3	1 = Operator can switch to "Out of service" mode
4	1 = Operator can stop counting
5	1 = Operator can switch to count up operating mode
6	1 = Operator can switch to count down operating mode
7 - 11	Not used
12	1 = Operator can enable the maintenance release function
13	1 = Operator can change the limit (`OutAH_Lim`) for high alarm
14	1 = The operator can change the limit (`OutWH_Lim`) for the high warning
15	1 = The operator can change the limit (`OutTH_Lim`) for the high tolerance
16	Not allocated
17	1 = Operator can change the limit (`OutTL_Lim`) for low alarm
18	1 = Operator can change the limit (`OutWL_Lim`) for low warning
19	1 = The operator can change the limit value (`OutAL_Lim`) for the low tolerance
20	Not allocated
21	1 = The operator can set the counter to the default value (`PresetEn`)
22	1 = Operator can change the default value (`PresetTime`)
23 - 31	Not used

Read back the last counted value

When counting (visible at the `Out` output parameter), this count value is passed directly to the `OldOut` input parameter:

```
OldOut = Out
```

If a warm restart is performed at this point, the (`Out`) count value is automatically reset to the predefined value, if the `Feature` bit Setting the startup response (Page 187) is set accordingly (=0) (`OldOut` ≠ `Out`). In this case, the value from `OldOut` is only updated when the (`Out`) count value is changed again by a pulse. Now `OldOut = Out` applies again.

Reset counter to zero

The counted value at the `Out` output parameter is reset with the interconnectable `ResetCount` parameter. The reset is made with a 0-1 edge.

The count cannot be reset to zero from the faceplate.

Setting the count to the default setting

You can use the `PresetVal` input parameter to enter a value at which the count should start, if the command has been given for setting the count to the default via the `PresetEn` input parameter with 1. This can also be done with the faceplate.

In this case, the `Out` output parameter is set to the `PresetVal` value.

Maintenance release

This block provides the standard function Maintenance release (Page 47).

SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 122).

See also

Description of CountScL (Page 1287)

CountScL messaging (Page 1297)

CountScL I/Os (Page 1299)

CountScL block diagram (Page 1302)

CountScL error handling (Page 1296)

CountScL modes (Page 1292)

Limit operation and display in the faceplate (Page 78)

13.1.4 CountScL error handling

CountScL troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of troubleshooting by CountScL

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Predefined value when inserting the block, the block is not processed.
0	There is no error.
51	Invalid signal at `LiOp = 1`: • `OffLi = 1` and `UpLi = 1` and/or `DnLi = 1` • `OffLi = 0` and `UpLi = 1` and `DnLi = 1`

See also

CountScL block diagram (Page 1302)

CountScL I/Os (Page 1299)

CountScL messaging (Page 1297)

Description of CountScL (Page 1287)

CountScL modes (Page 1292)

CountScL functions (Page 1293)

13.1.5 CountScL messaging

Messaging

The following messages can be generated for this block:

- Functions for displaying measured limits

Messages generated as a reaction to limit violations, can be suppressed by the settings `MsgEn` and `MsgLock`.

Associated values for message instance `MsgEvId`

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`

Process messages

Message instance	Message identifier	Message class	Event
MsgEvId	SIG 1	Alarm - high	$$BlockComment$$ High alarm limit violated
	SIG 2	Warning - high	$$BlockComment$$ High warning limit violated
	SIG 3	Tolerance - high	$$BlockComment$$ High tolerance limit violated
	SIG 4	Tolerance - low	$$BlockComment$$ Low tolerance limit violated
	SIG 5	Warning - low	$$BlockComment$$ Low warning limit violated
	SIG 6	Alarm - low	$$BlockComment$$ Low alarm limit violated
	SIG 7	Reserved	$$BlockComment$$
	SIG 8	Reserved	$$BlockComment$$

Explanation:

$$BlockComment$$: Content of the instance-specific comment

See also

Description of CountScL (Page 1287)
CountScL functions (Page 1293)
CountScL I/Os (Page 1299)
CountScL block diagram (Page 1302)
CountScL modes (Page 1292)
CountScL error handling (Page 1296)

13.1.6 CountScL I/Os

CountScL I/Os

Input parameters

Parameter	Description	Type	Default
BatchEn	1 = Enable allocation	BOOL	0
BatchID	Batch ID	DWORD	16#00000000
BatchName	Batch name	STRING[32]	''
DnLi	1 = Down counter, via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
DnOp	1 = Down counter, via operator	BOOL	0
Feature	I/O for additional functions (Page 1293)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
In	Binary input value	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
LiOp	1 = Interconnection, 0 = Operator	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MsgEvID	Message number (assigned automatically)	DWORD	16#FF
MsgLock	1 = Suppress process messages	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_RelOp	Operator can switch maintenance release	BOOL	0
Occupied	1 = In use by a batch	BOOL	0
OffLi	1 = Counter disabled via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OffOp	1 = Counter disabled via operator	BOOL	1
OldOut	Previous output value	DINT	0
OnOp	1 = "On" mode via operator	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC (0-1 edge transition)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0

Parameter	Description	Type	Default
OS_Perm	I/O for operator control permissions (Page 1293)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
OutAH_En	1 = Enable alarm (above) for count	BOOL	1
OutAH_Lim	Limit for count alarm (above)	DINT	95
OutAH_MsgEn	1 = Enable message for count alarm (above)	BOOL	1
OutAL_En	1 = Enable alarm (below) for count	BOOL	1
OutAL_Lim	Limit for count alarm (below)	DINT	0
OutAL_MsgEn	1 = Enable message for count alarm (below)	BOOL	1
OutOpHiScale	High limit for scale in count bar graph of faceplate	DINT	100
OutOpLoScale	Low limit for scale in count bar graph of faceplate	DINT	0
OutTH_En	1 = Enable tolerance message (high)	BOOL	0
OutTH_Lim	Count tolerance limit message (high)	DINT	85
OutTH_MsgEn	1 = Enable message for count tolerance message (high)	BOOL	1
OutTL_En	1 = Enable count tolerance message (low)	BOOL	0
OutTL_Lim	Count tolerance message limit (low)	DINT	0
OutTL_MsgEn	1 = Enable message for count tolerance message (low)	BOOL	1
OutUnit	Unit of measure for count Out	INT	0
OutWH_En	1 = Enable count warning (high)	BOOL	1
OutWH_Lim	Count warning limit (high)	DINT	90
OutWH_MsgEn	1 = Enable message for count warning (high)	BOOL	1
OutWL_En	1 = Enable count warning (low)	BOOL	1
OutWL_Lim	Count warning limit (low)	DINT	0
OutWL_MsgEn	1 = Enable message for count warning (low)	BOOL	1
PresetEn	Preset Out	• BOOL	0
PresetVal	Preset value	DINT	0
ResetCount	1 = Counter restart	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RunUpCyc	Number of operating cycles for which all messages are suppressed	INT	3
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1
SelFp1	Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
StepNo	Batch step number	DWORD	16#00000000
UpLi	1 = Forward counter, via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
UpOp	1 = Forward counter, via operator	BOOL	0

Output parameters

Parameter	Description	Type	Default
CountMode	0 = Counter off 1 = Count up 2 = Count down	INT	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see CountScL error handling (Page 1296)	INT	-1
MsgAckn	ALARM_8P: Output ACK_STATE; message acknowledgment state	WORD	16#0000
MsgErr	1 = Message processing error occurred	BOOL	0
MsgStat	ALARM_8P: STATUS output Error information of ALARM_8P	WORD	16#0000
MS_Release	Maintenance release	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OnAct	1 = "On" mode enabled	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
OosAct	1 = Block is "Out of service"	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
Out	Count	DINT	0
OutAH_Act	1 = Count alarm (above) enabled	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OutAL_Act	1 = Count alarm (below) enabled	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OutTH_Act	1 = Count tolerance message (high) enabled	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OutTL_Act	1 = Count tolerance message (low) enabled	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OutWH_Act	1 = Count warning (high) enabled	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Parameter	Description	Type	Default
OutWL_Act	1 = Count warning (low) enabled	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ST_Worst	Worst signal status	BYTE	16#80
Status1	Status word 1 (Page 1287)	DWORD	16#00000000
Status2	Status word 2 (Page 1287)	DWORD	16#00000000

See also

CountScL messaging (Page 1297)
CountScL block diagram (Page 1302)
CountScL modes (Page 1292)

13.1.7 CountScL block diagram

CountScL block diagram

A block diagram is not provided for this block.

See also

Description of CountScL (Page 1287)
CountScL modes (Page 1292)
CountScL functions (Page 1293)
CountScL error handling (Page 1296)
CountScL messaging (Page 1297)
CountScL I/Os (Page 1299)

13.1.8 Operator control and monitoring

13.1.8.1 Views of CountScL

Views of the CountScL block

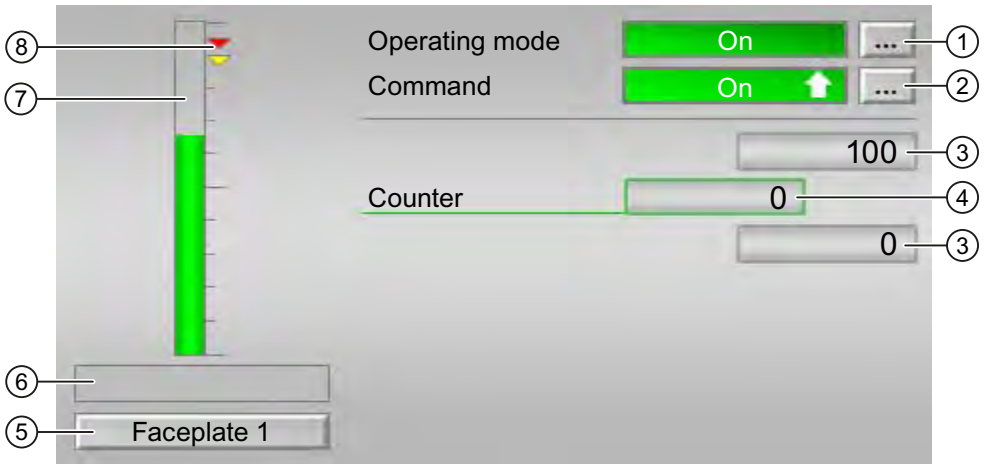
The block CountScL provides the following views:

- CountScL standard view (Page 1304)
- Message view (Page 169)
- CountScL limit value view (Page 1306)
- Trend view (Page 172)
- CountScL parameter view (Page 1307)
- CountScL preview (Page 1308)
- Memo view (Page 171)
- Batch view (Page 170)
- Block icon for CountScL (Page 1309)

Refer to the sections Structure of the faceplate (Page 123) and Block icon structure (Page 174) for general information on the faceplate and block icon.

13.1.8.2 CountScL standard view

CountScL standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 27)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Enable and disable the counter

This area shows you the default operating state for the counter. The following states can be shown and executed here:

- On ⇑
- On ⇓
- Off

Refer to the Switching operating states and operating modes (Page 127) section for information on changing the state.

(3) High and low scale range for the count value

These values provide information on the display range for the bar graph of the count. The scale range is defined in the engineering system.

(4) Display for count

The current count is displayed here.

(5) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

(6) Display area for states of the block

This area provides additional information on the operating state of the block:

- Maintenance

(7) Limit display

These colored triangles indicate the configured limits in the respective bar graph:

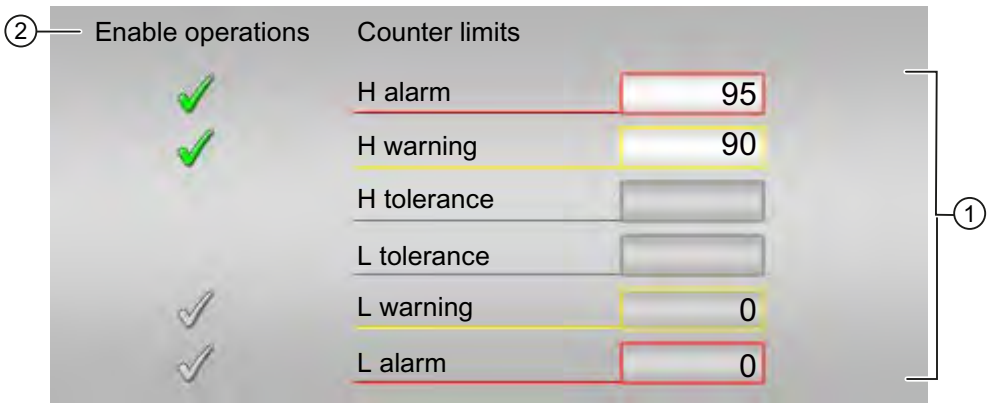
- Red: Alarm
- Yellow: Warning
- Blue: Tolerance

(8) Graphic display for the current count

This area shows you the current count in form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

13.1.8.3 CountScL limit value view

CountScL limit value view



(1) Limits for the counter

In this area, you can enter the limits for the counter. Refer to the Changing values (Page 129) section for more on this.

You can change the following limits:

- H alarm: Alarm high
- H warning: Warning high
- H tolerance: Tolerance high
- L tolerance: Tolerance low
- L warning: Warning low
- L alarm: Alarm low

(2) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions

13.1.8.4 CountScL parameter view

CountScL parameter view

(1) Preset counter

Enter the default setting here, where the counter should start. You can find additional information on this in the Changing values (Page 129) section.

(2) Set to default

Set the counter to the default value here. You can find additional information on this in the Switching operating states and operating modes (Page 127) section.

(3) Service

You can select the following function in this area:

- Maintenance release

Refer to the Switching operating states and operating modes (Page 127) section for more on this.

You can find information on this area in the Maintenance release (Page 47) section.

(4) Enable operations

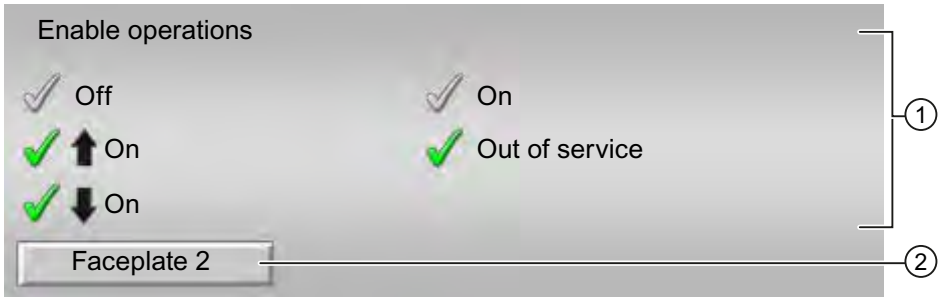
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions.

13.1.8.5 CountScL preview

CountScL preview



(3) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions

The following operator control enables are shown here **(1)**:

- Off: You can disable the counter
- On ⇑: You can control the incremental counter
- On ⇓: You can control the decremental counter
- On: You can switch to On operating mode.
- Out of service: You can switch to out of service mode.

(2) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

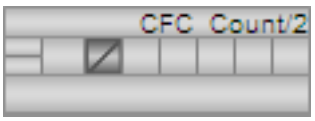
You can find additional information on this in the Calling further faceplates (Page 43) section.

13.1.8.6 Block icon for CountScL

Properties of the block icon CountScL

A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits
- Operating modes
- Signal status, maintenance release
- Memo display
- Display counter running

Icons	Selection of the block icon in CFC	Special features
	1	Block icon in the full display
	2	
	3	
	4	
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181).

13.2 CountOh - determining runtime

13.2.1 Description of CountOh

Object name (type + number) and family

Type + number: FB 1864

Family: Count

Area of application for CountOh

The block is used for the following applications:

- Calculating the runtime of a unit

How it works

The block calculates the operating time of a unit.

1. Off (`OffOp = 1` or `OffLi = 1`)

   The block is disabled (output parameter `CountMode` = 0). No counting takes place.

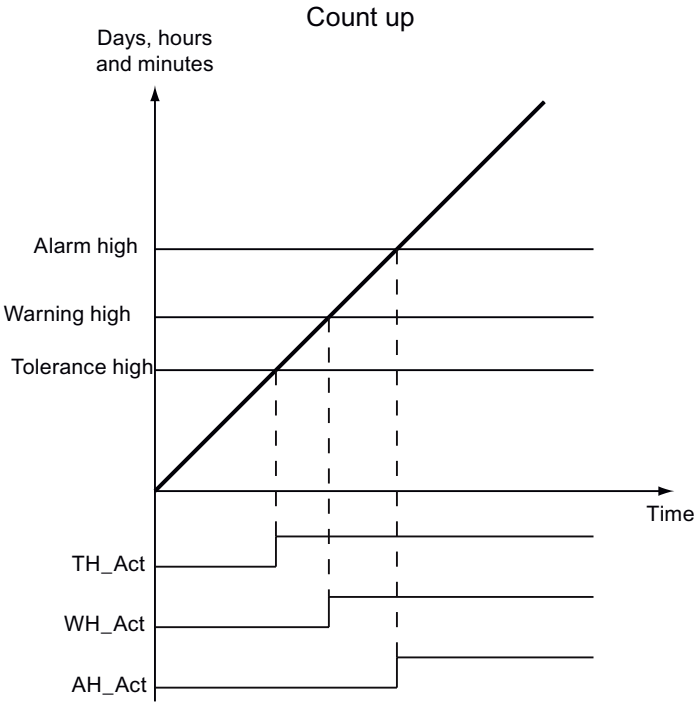
2. Count up (`UpOp = 1` or `UpLi = 1`)

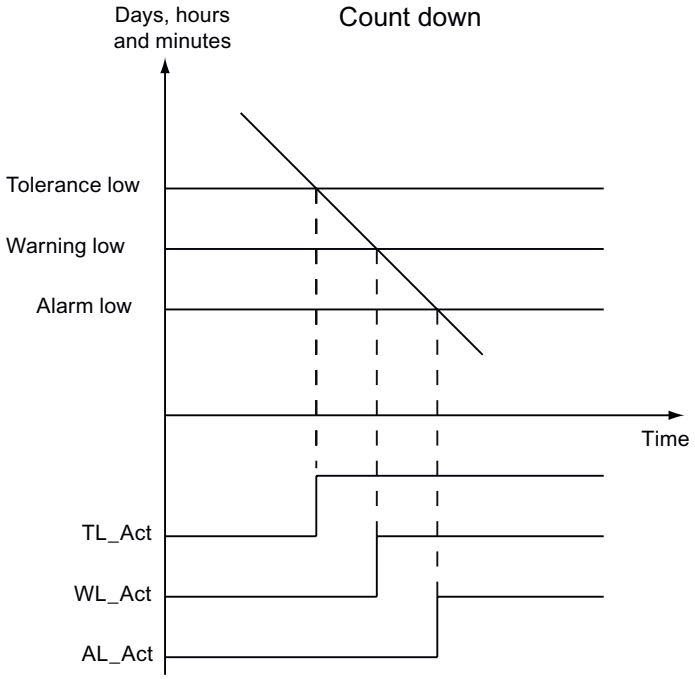
   The operating time of the connected unit is incremented (output parameter `CountMode` = 1).

3. Count down (`DnOp = 1` or `DnLi = 1`)

   The operating time of the connected unit is decremented (output parameter `CountMode` = 2).

The operating time is shown depending on the course of days, hours, minutes and seconds. If the operating time exceeds the configured limits, an alarm is triggered.





The operating time can be preset. The value can be specified by the operator. The operator can set values for days, hours and minutes to begin counting, if the required operator control permission has been issues for this.

The maximum configurable value for the operating time is 24855 days, 3 hours and 14 minutes.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38). The block is also installed automatically in the startup OB (OB 100).

Startup characteristics

Use the `Feature` bit Set startup characteristics (Page 187) to define the startup characteristics of this block.

Time response

The block only functions properly in a cyclic interrupt OB. To ensure correct time acquisition, it should be installed (if configured in CFC) in the same runtime group as the control block of the monitored unit.

Status word allocation for Status1 parameter

Status bit	Parameter
0	Occupied
1	BatchEn
2	Not used
3	OosAct.Value
4	OosLi.Value
5	Not used
6	OnAct.Value
7 - 10	Not used
11	LiOp
12 - 31	Not used

Status word allocation for Status2 parameter

Status bit	Parameter
0	MsgLock.Value
1	AH Act.Value
2	WH_Act.Value
3	TH_Act.Value
4	TL_Act.Value
5	WL_Act.Value
6	AL_Act.Value
7	AH_En
8	WH_En
9	TH_En
10	TL_En
11	WL_En
12	AL_En
13	AH_MsgEn
14	WH_MsgEn
15	TH_MsgEn
16	TL_MsgEn
17	WL_MsgEn
18	AL_MsgEn
19	Not used
20	Count up
21	Counter off
22	Count down
23 - 30	Not used
31	MS_RelOp

See also

CountOh functions (Page 1316)
CountOh messaging (Page 1320)
CountOh I/Os (Page 1322)
CountOh block diagram (Page 1327)
CountOh error handling (Page 1319)
CountOh modes (Page 1315)

13.2.2 CountOh modes

CountOh operating modes

The block can be operated using the following modes:

- On (Page 27)
- Out of service (Page 27)

On

You can find general information about the "On" mode in the On (Page 27) section.

Out of service

To put the motor in out of service mode, you must either:

1. set the `OosLi = 1` I/O or
2. activate the mode in the standard view of the faceplate (`OosOp = 1`) assuming you have the necessary permission.

You will find general information about the "Out of service" mode in the Out of service (Page 27) section.

See also

CountOh block diagram (Page 1327)

CountOh I/Os (Page 1322)

CountOh messaging (Page 1320)

CountOh error handling (Page 1319)

Description of CountOh (Page 1310)

CountOh functions (Page 1316)

13.2.3 CountOh functions

Functions of CountOh

The block provides the following functions:

Time measurement

The block is used to register the operating time of a block or of a functional unit.

The total time in minutes is displayed for the following limits:

1. Alarm (AH_Minutes and AL_Minutes)
2. Warning (WH_Minutes and WL_Minutes)
3. Tolerance (TH_Minutes and TL_Minutes)

When the block counts up and the total operating time (`TimeMin`) of the connected unit is greater than or equal to the limits (TH_Minutes, WH_Minutes and AH_Minutes), the alarms `TH_Act`, `WH_Act` and `AH_Act` are set.

When the block counts down and the total operating time (`TimeMin`) of the connected unit is less than the limits (TL_Minutes, WL_Minutes and AL_Minutes), the alarms `TL_Act`, `WL_Act` and `AL_Act` are set.

Configurable reactions using the Feature I/O

You will find an overview of all reactions provided by the `Feature` parameter in the section titled Functions of blocks >. Functions that can be set via the Feature I/O (Page 186)

The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)
1	Reaction to the out of service mode (Page 199)
22	Update acknowledgment and error status of the message call (Page 193)

Read back the last counted value

When counting (visible at the Days output parameter), this count value is passed directly to the OldDays input parameter:

```
OldDays = Days
```

If a warm restart is performed at this point, the (Days) count value is automatically reset to the predefined value, if the Feature bit Setting the startup response (Page 187) is set accordingly (=0) (OldDays ≠ Days). In this case, the value from OldDays is only updated when the (Days) count value is changed again. Now OldDays = Days applies again.

This readback function is only possible for counts for:

- Days (Days and OldDays)
- Hours (Hours and OldHours)
- Minutes (Minutes and OldMinutes)
- Seconds (Seconds and OldSeconds)

Display and control field for process values and setpoints

This block provides the standard function Display and operator input area for process values and setpoints (Page 42).

Reset counter to zero

The counted value at the Out output parameter is reset with the interconnectable Reset parameter. The reset is made with a 0-1 edge.

The count cannot be reset to zero from the faceplate.

Setting the count to the default setting

You can use the PresetTime input parameter to enter a value at which the count should start, if the command has been given for setting the count to the default via the PresetEn input parameter with 1. This can also be done with the faceplate. In this case, the Out output parameter is set to the PresetTime value.

The PresetTime time value is always indicated in seconds.

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status ST_Worst for the block is formed from the following parameter:

- In.ST

Operator control permissions via parameter OS_Perm

The block has the following Operator permissions (Page 45) for the OS_Perm parameter:

Bit	Function
0	Not used
1	1 = Operator can start the block
2	Not used
3	1 = Operator can switch to out of service mode
4	1 = Operator can stop the block
5	1 = Operator can switch to incrementing mode
6	1 = Operator can switch to decrementing mode
7 - 11	Not used
12	1 = Operator can enable maintenance release
13	1 = Operator can enter AH_Lim
14	1 = Operator can enter WH_Lim
15	1 = Operator can enter TH_Lim
16	Not used
17	1 = Operator can enter AL_Lim
18	1 = Operator can enter WL_Lim
19	1 = Operator can enter TL_Lim
20	Not used
21	1 = Operator can enable default
22	1 = Operator can apply configured time
23 - 31	Not used

Maintenance release

This block provides the standard function Maintenance release (Page 47).

SIMATIC BATCH functionality

This block provides the standard function SIMATIC BATCH functionality (Page 122).

See also

Description of CountOh (Page 1310)

CountOh messaging (Page 1320)

CountOh I/Os (Page 1322)

CountOh block diagram (Page 1327)

CountOh error handling (Page 1319)

CountOh modes (Page 1315)

13.2.4 CountOh error handling

CountOh troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Predefined value when inserting the block, the block is not processed.
0	There is no error.
14	Invalid signal at `LiOp = 1` If two of the three inputs are `OffLi`, `UpLi` and `DnLi = 1`

51	`LiOp = 1`
	`UpLi = 1`
	`DnLi = 1`

See also

CountOh block diagram (Page 1327)

CountOh I/Os (Page 1322)

CountOh messaging (Page 1320)

CountOh functions (Page 1316)

CountOh modes (Page 1315)

Description of CountOh (Page 1310)

13.2.5 CountOh messaging

Messaging

The following messages can be generated for this block:

- Functions for displaying measured limits

Messages generated as a reaction to limit violations, can be suppressed by the settings `xx_MsgEn` and `MsgLock`.

Process messages

Message instance	Message identifier	Message class	Event
MsgEvId1	SIG 1	Alarm - high	$$BlockComment$$ High alarm limit violated
	SIG 2	Warning - high	$$BlockComment$$ High warning limit violated
	SIG 3	Tolerance - high	$$BlockComment$$ High tolerance limit violated
	SIG 4	Tolerance - low	$$BlockComment$$ Low tolerance limit violated
	SIG 5	Warning - low	$$BlockComment$$ Low warning limit violated
	SIG 6	Alarm - low	$$BlockComment$$ Low alarm limit violated
	SIG 7	Reserved	$$BlockComment$$ For internal use
	SIG 8	Reserved	$$BlockComment$$ For internal use

Explanation:

$$BlockComment$$: Content of the instance-specific comment

Associated values for message instance `MsgEvId1`

Associated value	Block parameters
1	`BatchName`
2	`StepNo`
3	`BatchID`

See also

Description of CountOh (Page 1310)

CountOh functions (Page 1316)

CountOh I/Os (Page 1322)

CountOh block diagram (Page 1327)

CountOh error handling (Page 1319)

CountOh modes (Page 1315)

13.2.6 CountOh I/Os

CountOh I/Os

Input parameters

Parameter	Description	Type	Default
AH_En	High alarm enabled	BOOL	1
AH_MsgEn	Message for high alarm enabled	BOOL	1
AL_En	Low alarm enabled	BOOL	1
AL_MsgEn	Message for low alarm enabled	BOOL	1
BatchEn	1 = Enable allocation	BOOL	0
BatchID	Batch ID (batch ID)	DWORD	16#00000000
BatchName	Batch name	STRING[32]	"
DayOpLoScale	Operating stage - Low limit of bar display for OS	INT	0
DaysAHLim	Days - high limit alarm	INT	95
DaysALLim	Days - low limit alarm	INT	0
DaysTHLim	Days - high limit tolerance	INT	85
DaysTLLim	Days - low limit tolerance	INT	0
DaysWHLim	Days - limit for high warning	INT	90
DaysWLLim	Days - limit for low warning	INT	0
DnLi	1 = Count down	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
DnOp	1 = Count down	BOOL	0
Feature	I/O for additional functions (Page 1316)	STRUCT Bit 0 - Bit 31 BOOL	0
HrsAHLim	Hours - high limit alarm	INT	0
HrsALLim	Hours - low limit alarm	INT	0
HrsOpHiScale	Operating hours - high limit of bar display for OS	INT	23
HrsOpLoScale	Operating hours - low limit of bar display for OS	INT	0
HrsTHLim	Hours - high limit tolerance	INT	0
HrsTLLim	Hours - low limit tolerance	INT	0
HrsWHLim	Hours - high limit warning	INT	0
HrsWLLim	Hours - low limit warning	INT	0
In	Device status 1=ON, 0=OFF	STRUCT • Value: BOOL • ST: Byte	- • 0 • 16#80

Parameter	Description	Type	Default
LiOp	Switchover of operating mode between: 1 = Interconnection 0 = Operator	STRUCT • Value: BOOL • ST: Byte	- • 0 • 16#80
MinOpHiScale	Operating minutes - high limit of bar display for OS	INT	59
MinOpLoScale	Operating minutes - low limit of bar display for OS	INT	0
MinsAHLim	Minutes - high limit alarm	INT	0
MinsALLim	Minutes - low limit alarm	INT	0
MinsTHLim	Minutes - high limit tolerance	INT	0
MinsTLLim	Minutes - low limit tolerance	INT	0
MinsWHLim	Minutes - high limit warning	INT	0
MinsWLLim	Minutes - low limit warning	INT	0
MsgEvId	Message number (assigned automatically)	DWORD	16#FF
MsgLock	1 = Suppress process messages	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MS_RelOp	Operator input for maintenance release, 1: Maintenance release request	BOOL	0
Occupied	1 = In use by a batch	BOOL	0
OffLi	Counter disabled via interconnection, 1 = Off	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OffOp	Counter disabled via operator, 1 = Off	BOOL	1
OldDays	Previous day value	INT	0
OldHours	Previous hour value	INT	0
OldMinutes	Previous minute value	INT	0
OldSeconds	Previous second value	INT	0
OnOp	1 = "On" mode via operator	BOOL	0
OosLi	1 = Out of service, via interconnection	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via operator	BOOL	0
OS_Perm	I/O for CountOh functions (Page 1316)	STRUCT Bit 0 - Bit 31 BOOL	 1
PresetEn	1 = Set block to default time value (PresetTime)	BOOL	0
PresetTime	Default setting for the time value [s]	DWORD	16#00000000

Parameter	Description	Type	Default
Reset	1 = Reset counter	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
RunUpCyc	Number of cycles in startup; messages are suppressed during these cycles	INT	3
SampleTime	Sampling time [s]	REAL	0.1
SelFp1	Open faceplate 1	ANY	-
SelFp2	Open faceplate 2	ANY	-
StepNo	Batch step number	DWORD	16#00000000
TH_En	High tolerance enabled	BOOL	0
TH_MsgEn	Alarm for high tolerance enabled	BOOL	1
TL_En	Low tolerance enabled	BOOL	0
TL_MsgEn	Alarm for low tolerance enabled	BOOL	1
WH_En	High warning enabled	BOOL	1
WH_MsgEn	Alarm for high warning enabled	BOOL	1
WL_En	Low warning enabled	BOOL	1
WL_MsgEn	Alarm for low warning enabled	BOOL	1
UpLi	1 = Increment (via interconnection)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
UpOp	1 = Increment (via faceplate)	BOOL	0

Output parameters

Parameter	Description	Type	Default
AH_Act	High alarm enabled	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AH_Minutes	High alarm time [in min]	DINT	0
AL_Act	Low alarm enabled	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
AL_Minutes	Low alarm time [in min]	DINT	0
BarOpHiScale	High limit for bar display [min]	DINT	100
BarOpLoScale	Low limit for bar display [min]	DINT	0
CountMode	Count mode: 0 = Off 1 = Increment, 2 = Decrement	INT	0
Days	Operating stage	INT	0
DeviceOn	1 = Unit on	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
ErrorNum	Error number	INT	-1
Hours	Operating hours	INT	0
Minutes	Operating minutes	INT	0
MS_Release	Maintenance release	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
MsgAckn	ALARM_8P: Output ACK_STATE; message acknowledgment state	WORD	16#0000
MsgErr	1 = Message processing error occurred	BOOL	0
MsgStat	ALARM_8P: STATUS output Error information of ALARM_8P	WORD	16#0000
OnAct	Block running	STRUCT • Value: BOOL • ST: BYTE	1 16#80
OosAct	Block not running	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermLog	Operator control permission: Output for OS	DWORD	16#FFFFFFFF
OS_PermOut	Operator control permission: Output for OS	DWORD	16#FFFFFFFF
ST_Worst	Worst signal status	BYTE	16#80
Seconds	Operating time [in sec]	INT	0
Status1	Status word 1 (Page 1310)	DWORD	16#00000000
Status2	Status word 2 (Page 1310)	DWORD	16#00000000

Parameter	Description	Type	Default
TH_Act	High tolerance enabled	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
TH_Minutes	Tolerance high	DINT	0
TimeMin	Operating period [min]	DINT	0
TL_Act	Low tolerance enabled	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
TL_Minutes	Low tolerance time [in min]	DINT	0
TotalTime	Total operating time	DWORD	16#00000000
WH_Act	High warning enabled	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
WH_Minutes	High warning time [in min]	DINT	0
WL_Act	Low warning enabled	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
WL_Minutes	Low alarm time [in min]	DINT	0

See also

CountOh messaging (Page 1320)
CountOh block diagram (Page 1327)
CountOh error handling (Page 1319)
CountOh modes (Page 1315)

13.2.7 CountOh block diagram

CountOh block diagram

A block diagram is not provided for this block.

See also

CountOh I/Os (Page 1322)

CountOh messaging (Page 1320)

CountOh error handling (Page 1319)

CountOh functions (Page 1316)

CountOh modes (Page 1315)

Description of CountOh (Page 1310)

13.2.8 Operator control and monitoring

13.2.8.1 Views of CountOh

Views of the CountOh block

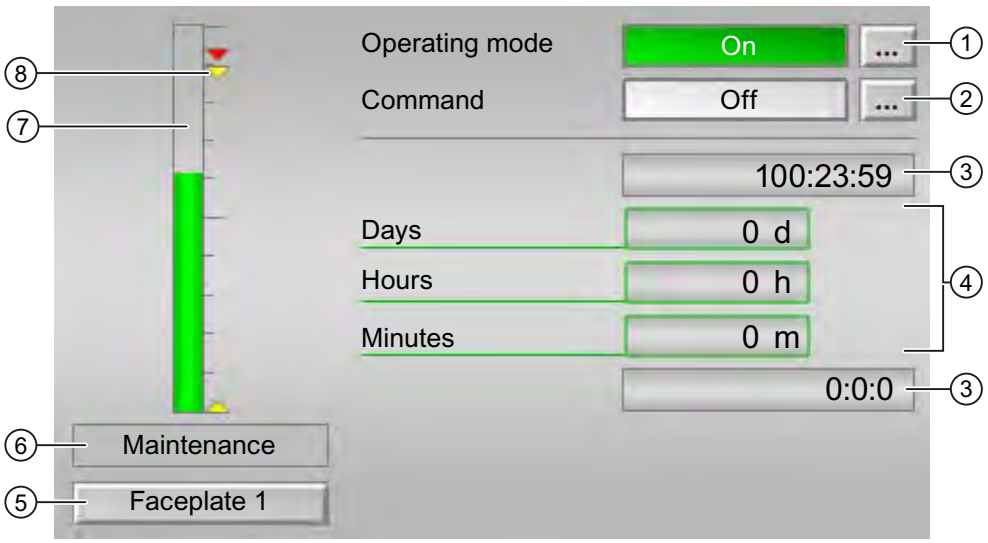
The block CountOh provides the following views:

- CountOh standard view (Page 1328)
- Message view (Page 169)
- CountOh limit value view (Page 1330)
- Trend view (Page 172)
- CountOh parameter view (Page 1331)
- CountOh preview (Page 1333)
- Memo view (Page 171)
- Batch view (Page 170)
- Block icon for CountOh (Page 1334)

Refer to the Structure of the faceplate (Page 123) and Block icon structure (Page 174) sections for general information about the faceplate and block icon.

13.2.8.2 CountOh standard view

CountOh standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 27)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Enable and disable the counter

This area shows you the default operating state for the counter. The following states can be shown and executed here:

- On ⇑
- On ⇓
- Off

Refer to the Switching operating states and operating modes (Page 127) section for information on changing the state.

(3) High and low scale range for the count value

These values provide information on the display range for the bar graph (5) of the count value. The scale range is defined in the engineering system.

(4) Display for counts

The following counts are shown here:

- Days
- Hours
- Minutes

(5) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

Additional information is available in the section Calling further faceplates (Page 43).

(6) Display area for states of the block

This area provides additional information on the operating state of the block:

- Maintenance

(7) Graphic display for the current count

This area shows you the current count in form of a bar graph. The visible area in the bar graph depends on the configuration in the engineering system (ES).

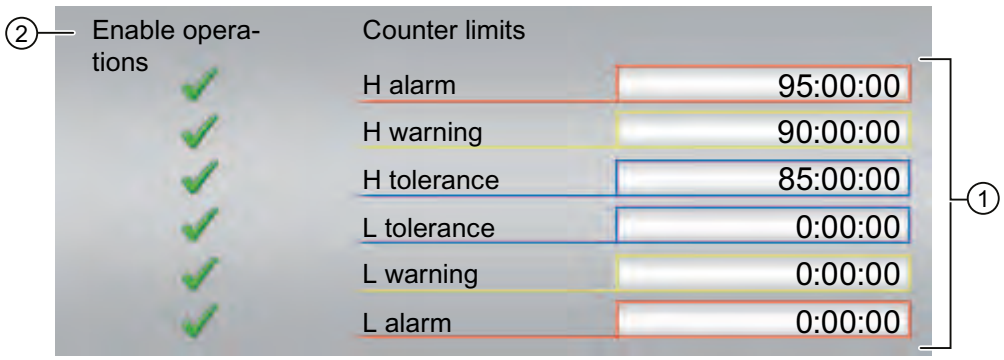
(8) Limits

These colored triangles indicate the configured limits in the respective bar graph:

- Red: Alarm
- Yellow: Warning
- Blue: Tolerance

13.2.8.3 CountOh limit value view

CountOh limit value view



(1) Limits for the counter

In this area, you can enter the limits for the counter. Refer to the Changing values (Page 129) section for more on this.

You can change the following limits:

- H alarm: Alarm high
- H warning: Warning high
- H tolerance: Tolerance high
- L tolerance: Tolerance low
- L warning: Warning low
- L alarm: Alarm low

(2) Enable operations

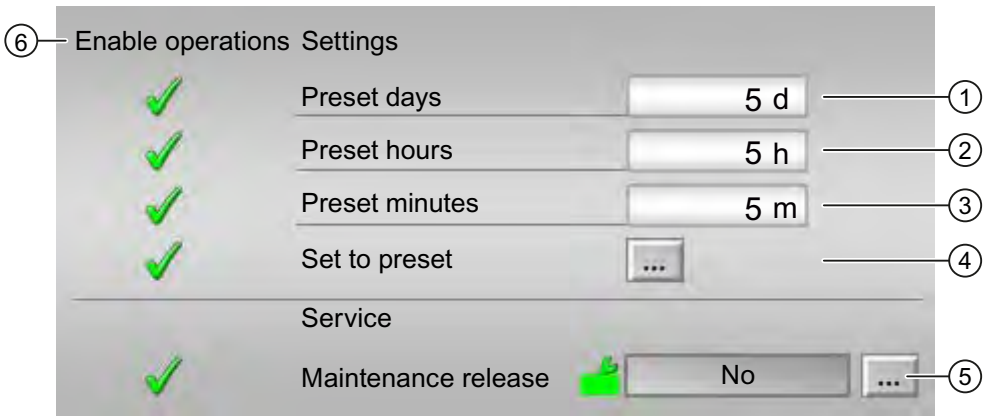
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions

13.2.8.4 CountOh parameter view

CountOh parameter view



(1) (2) and (3) preset counter

Enter the default setting here, where the counter should start.

You can change the following presets:

- Days
- Hours
- Minutes

You can find additional information on this in the Changing values (Page 129) section.

(4) Set to default

Set the counter to the default value here. You can find additional information on this in the Switching operating states and operating modes (Page 127) section.

(5) Service

You can select the following function in this area:

- Maintenance release

Refer to the Switching operating states and operating modes (Page 127) section for more on this.

You can find information on this area in the Maintenance release (Page 47) section.

(6) Enable operations

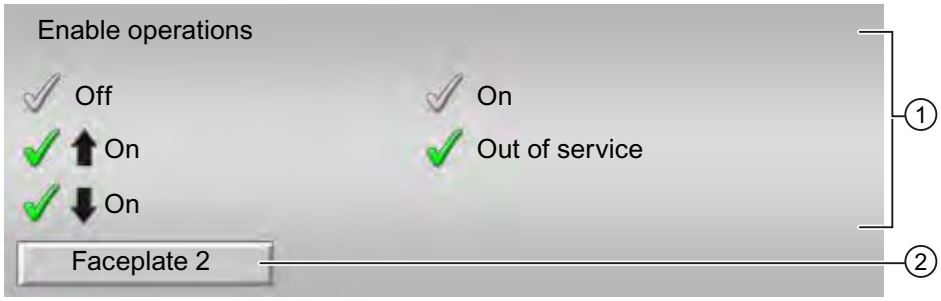
This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions.

13.2.8.5 CountOh preview

CountOh preview



(1) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- **Green check mark**: the OS operator can control this parameter
- **Gray check mark**: the OS operator cannot control this parameter at this time due to the process
- **Red cross**: the OS operator cannot control this parameter due to the configured operator control permissions

The following operator control enables are shown here:

- Off: You can disable the counter
- On ⇑: You can control the incremental counter
- On ⇓: You can control the decremental counter
- On: You can switch to On operating mode.
- Out of service: You can switch to out of service mode.

(2) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

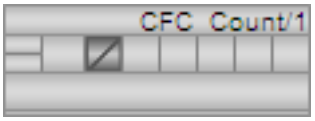
You can find additional information on this in the Calling further faceplates (Page 43) section.

13.2.8.6 Block icon for CountOh

Properties of the CountOh block icon

A variety of block icons are available with the following functions:

- Process tag type
- Limits (high/low)
- Violation of alarm, warning, and tolerance limits
- Operating modes
- Signal status, maintenance release
- Memo display
- Display counter running

Icons	Selection of the block icon in CFC	Special features
	1	Block icon in the full display
	2	
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181).

Logical analog blocks

14

14.1 Limit - Limiting an analog value

14.1.1 Description of Limit

Object name (type + number) and family

Type + number: FB 1829

Family: LogicAn

Area of application for Limit

The block is used for the following applications:

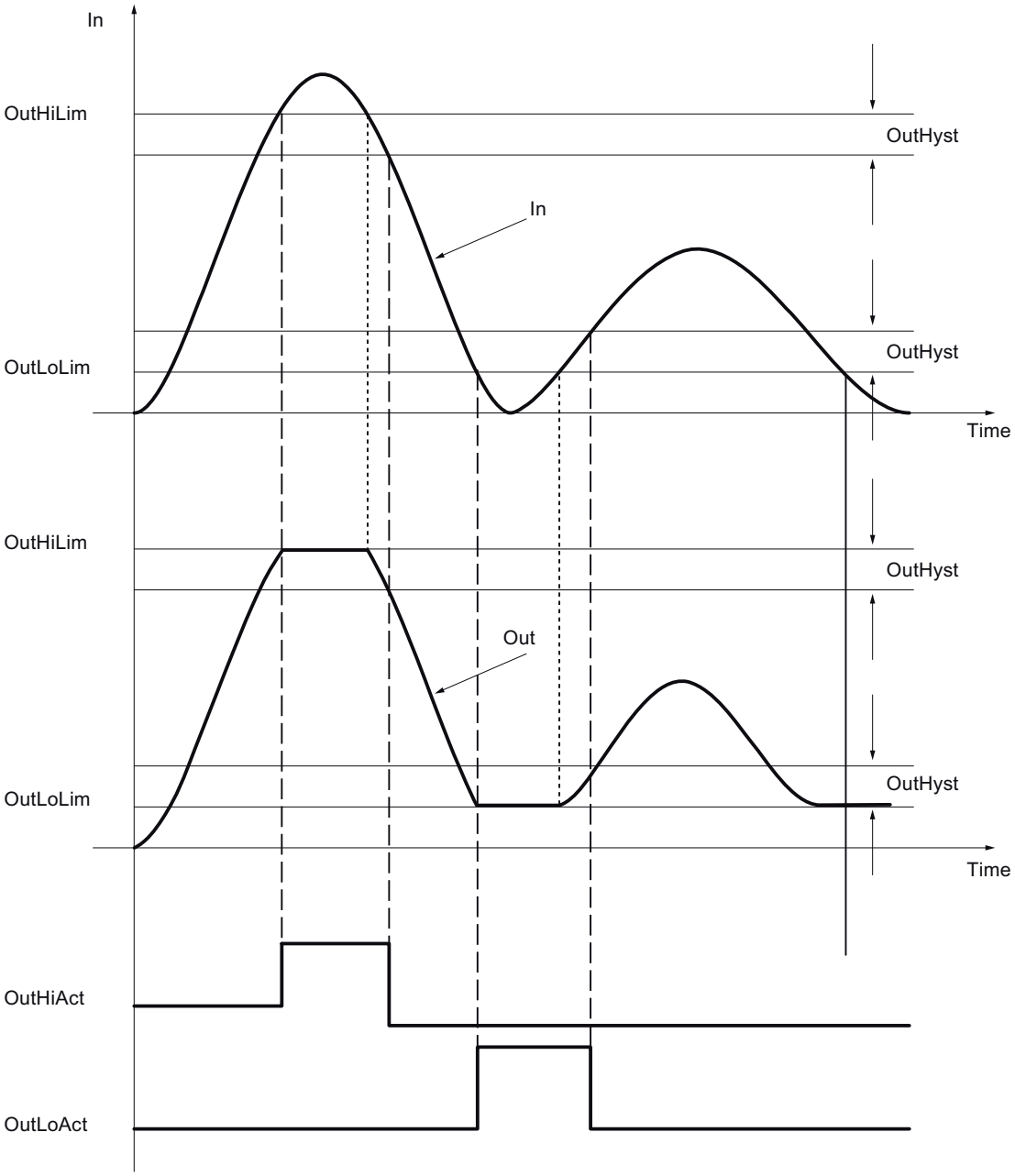
- Limiting of values

How it works

The Limit block is used to limit an analog value to an adjustable range.

The limits are set at the `OutHiLim` (high limit) and `OutLoLim` (low limit) input parameters. If a limit is violated, the limit you entered is output. The limit violation is also shown at the two output parameters `OutHiAct` (high) or `OutLoAct` (low).

You can configure hysteresis (`OutHyst` input parameter) to suppress signal flutters close to the limits.



Active conditions for setting limits	Active conditions for resetting limits
High limit active: In >= OutHiLim	High limit active: In < OutHiLim - OutHyst
Low limit active: In <= OutLoLim	Low limit active: In > OutLoLim + OutHyst

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

Further addressing is not required.

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for `Status1` parameter

This block does not provide the `Status` parameter.

See also

Limit block diagram (Page 1341)
Limit I/Os (Page 1340)
Limit messaging (Page 1339)
Limit error handling (Page 1339)
Limit functions (Page 1338)
Limit modes (Page 1338)

14.1.2 Limit modes

Limit modes

This block does not provide any modes.

See also

Limit block diagram (Page 1341)
Limit I/Os (Page 1340)
Limit messaging (Page 1339)
Limit error handling (Page 1339)
Limit functions (Page 1338)
Description of Limit (Page 1335)
Out of service (Page 27)

14.1.3 Limit functions

Functions of Limit

There are no other functions for this block.

See also

Limit block diagram (Page 1341)
Limit I/Os (Page 1340)
Limit messaging (Page 1339)
Limit error handling (Page 1339)
Limit modes (Page 1338)
Description of Limit (Page 1335)

14.1.4 Limit error handling

Limit error handling

The block does not report any errors.

See also

Limit block diagram (Page 1341)
Limit I/Os (Page 1340)
Limit messaging (Page 1339)
Limit functions (Page 1338)
Limit modes (Page 1338)
Description of Limit (Page 1335)

14.1.5 Limit messaging

Messaging

The block does not offer messaging.

See also

Limit block diagram (Page 1341)
Limit I/Os (Page 1340)
Limit error handling (Page 1339)
Limit functions (Page 1338)
Limit modes (Page 1338)
Description of Limit (Page 1335)

14.1.6 Limit I/Os

Limit I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
Feature	Reserved for future functions		
In	Analog input value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
OutHyst	Hysteresis in %	REAL	0.0
OutHiLim	High limit of output value	REAL	100.0
OutLoLim	Low limit of output value	REAL	0.0

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Reserved for future functions	Int	-1
Out	Analog output value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
OutHiAct	1 = High limit overshoot	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OutLoAct	1 = Low limit undershoot	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

Limit block diagram (Page 1341)
Limit messaging (Page 1339)
Limit error handling (Page 1339)
Limit functions (Page 1338)
Limit modes (Page 1338)
Description of Limit (Page 1335)

14.1.7 Limit block diagram

Limit block diagram

A block diagram is not provided for this block.

See also

Limit I/Os (Page 1340)
Limit messaging (Page 1339)
Limit error handling (Page 1339)
Limit functions (Page 1338)
Limit modes (Page 1338)
Description of Limit (Page 1335)

14.2 MuxAn03 - Selection of an analog value to increase availability / certainty

14.2.1 Description of MuxAn03

Object name (type + number) and family

Type + number: FB 1860

Family: LogicAn

Area of application for MuxAn03

The block is used for the following applications:

- Selection of an analog value to increase availability or certainty in the input of analog values

How it works

The block uses up to three PV1 ... PV3 process values to determine an output value and outputs this with the corresponding signal status at the PV output parameter.

Configuration

Use the CFC editor to install the block in any OB.

Startup characteristics

The block does not have any special startup characteristics.

Status word allocation for Status1 parameter

This block does not provide the Status parameter.

See also

MuxAn03 block diagram (Page 1347)

MuxAn03 I/Os (Page 1346)

MuxAn03 messaging (Page 1345)

MuxAn03 error handling (Page 1345)

MuxAn03 functions (Page 1343)

MuxAn03 modes (Page 1343)

14.2.2 MuxAn03 modes

MuxAn03 operating modes

This block does not provide any operating modes.

See also

MuxAn03 block diagram (Page 1347)

MuxAn03 I/Os (Page 1346)

MuxAn03 messaging (Page 1345)

MuxAn03 error handling (Page 1345)

MuxAn03 functions (Page 1343)

Description of MuxAn03 (Page 1342)

Out of service (Page 27)

14.2.3 MuxAn03 functions

Functions of MuxAn03

The functions for this block are listed below.

Selection of output signal

Use the `SelValue` input parameter to select whether the output value is to offer higher availability or higher certainty.

Increasing availability

- **Selection 1 of 2** (`SelValue = 0`): The block determines the input parameter with the higher priority from the two `PV1` and `PV2` input parameters using their signal states. The value determined is written to the `PV` output parameter.

  If both input parameters have the same signal status, the `PV1` input parameter is written to the `PV` output parameter.

- **Selection 1 of 3** (`SelValue = 1`): The block determines the input parameter with the highest priority from the three `PV1`, `PV2` , and `PV3` input parameters using their signal states. The value determined is written to the `PV` output parameter.

  If two or more of the input parameters have the same signal status, the one with the lowest index is written to the `PV` output parameter.

Increasing certainty

- **Selection 2 of 2** (`SelValue = 2`): The block determines whether the two `PV1` and `PV2` input parameters have the same signal status and whether they deviate from one another by no more than the setting at input parameter `PlDiff`. Only then does the `PV` output parameter also have this signal status. The analog value of `PV` is set to a value of `PV1`.

  If `PV1` and `PV2` deviate from one another by more than `PlDiff`, the PV signal status is set to "Bad, due to device". The analog value of `PV` is set to a value of `PV1`.

  If `PV1` and `PV2` do not deviate from one another by more than `PlDiff`, but have different signal states, the `PV` output parameter is generated from the lower priority signal status of the two input parameters and the associated analog value.

- **Selection 2 of 3** (`SelValue = 3`): The block determines whether two of the three input parameters `PV1`, `PV2` and `PV3` have the same high priority signal status and whether they deviate from one another by no more than the setting at input parameter `PlDiff`. Only if this is the case, does the `PV` output parameter also have this signal status. The analog value of `PV` is set to the value of the input parameter with the lowest index (`PV1` or `PV2`) of the values determined.

  If all `PV1`, `PV2` and `PV3` input parameters deviate from one another by more than `PlDiff`, the signal status of `PV`is set to "Bad, due to device". The analog value of `PV`is set to the value of the `PV1` input parameter.

  If at least two input parameters do not deviate from one another by more than `PlDiff`, but have different signal states, the value with the signal status of the second highest priority is written to the `PV` output parameter.

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

See also

MuxAn03 block diagram (Page 1347)

MuxAn03 I/Os (Page 1346)

MuxAn03 messaging (Page 1345)

MuxAn03 error handling (Page 1345)

MuxAn03 modes (Page 1343)

Description of MuxAn03 (Page 1342)

Error handling (Page 117)

Selecting signals for processing (Page 93)

14.2.4 MuxAn03 error handling

MuxAn03 troubleshooting

Parameter assignment errors are handled as follows:

- If the input parameter is `SelValue < 0`, the parameter is automatically set to `SelValue = 0`.
- If the input parameter is `SelValue > 3`, the parameter is automatically set to `SelValue = 3`.
- If the input parameter is `SelPrio < 0`, the parameter is automatically set to `SelPrio = 0`.
- If the input parameter is `SelPrio > 7`, the parameter is automatically set to `SelPrio = 7`.

An error number is not output in any of these cases.

See also

MuxAn03 block diagram (Page 1347)
MuxAn03 I/Os (Page 1346)
MuxAn03 messaging (Page 1345)
MuxAn03 functions (Page 1343)
MuxAn03 modes (Page 1343)
Description of MuxAn03 (Page 1342)

14.2.5 MuxAn03 messaging

Messaging

The block does not have any message functionality.

See also

MuxAn03 block diagram (Page 1347)
MuxAn03 I/Os (Page 1346)
MuxAn03 error handling (Page 1345)
MuxAn03 functions (Page 1343)
MuxAn03 modes (Page 1343)
Description of MuxAn03 (Page 1342)

14.2.6 MuxAn03 I/Os

MuxAn03 I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
PV1	Process value 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV2	Process value 2	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PV3	Process value 3	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
PlDiff	Plausibility check value	REAL	0.0
SelPrio	Setting of the prioritization for forming the best signal status	INT	6
SelValue	Selection criterion for the search	INT	0

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
PV	Process value determined	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

See also

MuxAn03 block diagram (Page 1347)
MuxAn03 messaging (Page 1345)
MuxAn03 error handling (Page 1345)
MuxAn03 functions (Page 1343)
MuxAn03 modes (Page 1343)
Description of MuxAn03 (Page 1342)

14.2.7 MuxAn03 block diagram

MuxAn03 block diagram

A block diagram is not provided for this block.

See also

MuxAn03 I/Os (Page 1346)
MuxAn03 messaging (Page 1345)
MuxAn03 error handling (Page 1345)
MuxAn03 functions (Page 1343)
MuxAn03 modes (Page 1343)
Description of MuxAn03 (Page 1342)

14.3 RateLim - Signal ramp

14.3.1 Description of RateLim

Object name (type + number) and family

Type + number: FB 1882

Family: LogicAn

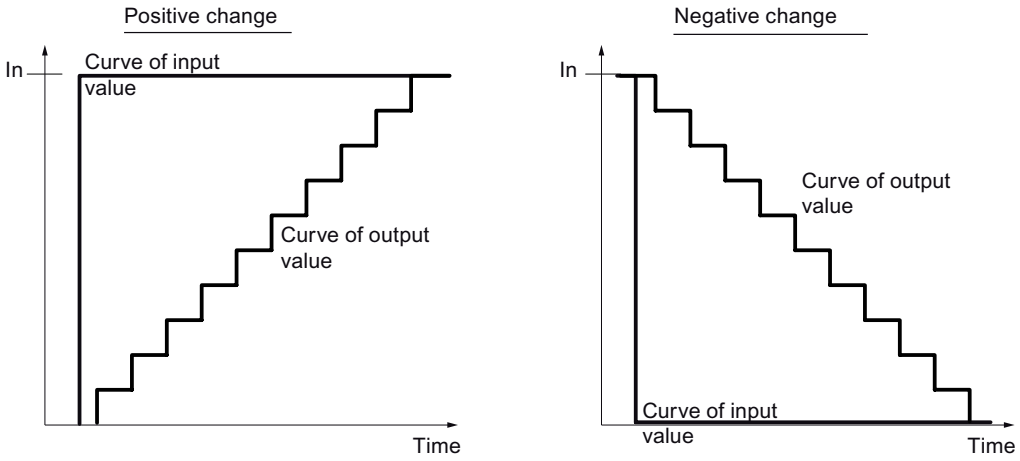
Area of application for RateLim

The block is used for the following applications:

- Limitation of the ramp of an analog signal

How it works

Based on the input value, the output value is output following a throughput and via a ramp, as is shown in the following graphic.



The input value and the ramp gradient can be limited.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB3x). The block is also installed automatically in startup OB100.

Further addressing is not required.

Startup characteristics

Use the `Feature` bit Setting the startup response (Page 187) to define the startup characteristics of this block.

Status word allocation for `Status` parameter

This block does not provide the `Status` parameter.

See also

RateLim functions (Page 1350)
RateLim messaging (Page 1353)
RateLim I/Os (Page 1353)
RateLim block diagram (Page 1355)
RateLim error handling (Page 1352)
RateLim modes (Page 1349)

14.3.2 RateLim modes

RateLim operating modes

This block does not provide any operating modes.

See also

RateLim block diagram (Page 1355)
RateLim I/Os (Page 1353)
RateLim messaging (Page 1353)
RateLim error handling (Page 1352)
RateLim functions (Page 1350)
Description of RateLim (Page 1348)

14.3.3 RateLim functions

Functions of RateLim

The functions for this block are listed below.

Limitation of the ramp of an analog signal

The block calculates the gradient of the input signal over time and compares it with the `UpRaLim` (positive change) `DnRaLim` (negative change) limits. See the table below.

- If the gradient value exceeds the corresponding limit (`UpRaLim` or `DnRaLim`), the output `Out` will be corrected by the valid limit and the corresponding limit display `UpRaAct = 1` or `DnRaAct = 1` will be set.
- If the gradient is within the valid range, the input value is transferred (`In = Out`) and `UpRaAct = 1` or `DnRaAct = 1` are reset.
- If the corresponding limit is 0 (`UpRaLim` with positive gradient or `DnRaLim` with negative gradient), the input value `In` is written directly to the output `Out`.

RateOn	ΔIn / Δt	Meaning	Output `Out`	UpRaAct	DnRaAct
1	`< | DnRaLim |`	Input value dropping too fast	`Out - ( | DnRaLim | ·SampleTime)`	0	1
1	`| DnRaLim |` to `UpRaLim`	Change speed `In` is permitted	`In`	0	0
1	`> UpRaLim`	Input value `In` rising too fast	`Out + ( | UpRaLim | ·SampleTime)`	1	0
0	-	`RateOn` is deactivated	`In`	0	0

Activation / deactivation of the ramp

The `RateOn = 0` input deactivates the generation of ramps, the `In` input value is written directly to the `Out` output. Limits are no longer monitored.

Configurable reactions using the Feature I/O

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section.

The following functionality is available for this block at the relevant bits:

Bit	Function
0	Setting the startup response (Page 187)
8	Unit for the rate of change (Page 190)

See also

Description of RateLim (Page 1348)
RateLim messaging (Page 1353)
RateLim I/Os (Page 1353)
RateLim block diagram (Page 1355)
RateLim modes (Page 1349)
RateLim error handling (Page 1352)

14.3.4 RateLim error handling

RateLim troubleshooting

Please refer to the section Error handling (Page 117) in the basic instructions.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Default value when implementing the block; block will not be processed.
0	There is no error.
30	The value of `In` can no longer be displayed in the REAL number field.
43	Incorrect time unit set at TimeFactor.

See also

RateLim block diagram (Page 1355)

RateLim I/Os (Page 1353)

RateLim messaging (Page 1353)

RateLim modes (Page 1349)

Description of RateLim (Page 1348)

RateLim functions (Page 1350)

14.3.5 RateLim messaging

Message functionality

The block does not have any message functionality.

See also

Description of RateLim (Page 1348)
RateLim functions (Page 1350)
RateLim I/Os (Page 1353)
RateLim block diagram (Page 1355)
RateLim error handling (Page 1352)
RateLim modes (Page 1349)

14.3.6 RateLim I/Os

RateLim I/Os

Input parameters

Parameter	Description	Type	Default
DnRaLim	Maximum negative change in the output value in units/s or %/s	REAL	3.0
EN	1 = Called block will be processed	BOOL	1
Feature	I/O for additional functions (Page 1350)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
In	Analog input value	STRUCT • Value: REAL • ST:BYTE	- • 0.0 • 16#80
InScale	Scaling of the measuring range as a structure	STRUCT • High: REAL • Low: REAL	- • 100.0 • 0.0
RateOn	1 = Ramp function is active	STRUCT • Value: BOOL • ST:BYTE	- • 0 • 16#80
SampleTime	Sampling time [s] (assigned automatically)	REAL	0.1

Parameter	Description	Type	Default
TimeFactor	Time unit: • 0 = Seconds • 1 = Minutes • 2 = Hours	INT	0
UpRaLim	Maximum positive change in the output value in units/s or %/s	REAL	3.0

Output parameters

Parameter	Description	Type	Default
DnRaAct	1 = Negative change in the output value	STRUCT • Value: BOOL • ST:BYTE	- • 0 • 16#80
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see RateLim error handling (Page 1352)	INT	-1
Out	Output value	STRUCT • Value: REAL • ST:BYTE	- • 0.0 • 16#80
UpRaAct	1 = Positive change in the output value	STRUCT • Value: BOOL • ST:BYTE	- • 0 • 16#80

See also

Description of RateLim (Page 1348)

RateLim messaging (Page 1353)

RateLim block diagram (Page 1355)

RateLim modes (Page 1349)

14.3.7 RateLim block diagram

Ramp block diagram

A block diagram is not provided for this block.

See also

RateLim I/Os (Page 1353)
RateLim messaging (Page 1353)
RateLim error handling (Page 1352)
RateLim functions (Page 1350)
RateLim modes (Page 1349)
Description of RateLim (Page 1348)

14.4 SelA02In - Output of two analog values

14.4.1 Description of SelA02In

Object name (type + number)

Type + number: FB 1886

Family: LogicAn

Area of application for SelA02In

The block is used for the following applications:

- Selecting one of two analog values and switching it through to the output.

How it works

Depending on the setting of parameter `SelMode`, the block selects one of the two input parameters In1 or In2 and writes its value to the output parameter Out.

This is displayed at the `In2Selected` output parameter.

Configuration

Use the CFC editor to install the block in any OB.

For the SelA02In block, the Advanced Process Library contains templates for process tag types as examples and there is a example project (APL_Example_xx, xx designates the language variant) with an application scenario for this block, which describes how the block works.

Examples of process tag types:

- Source chart for GainSched function block (gain scheduling) (Page 1444)
- Override control (Page 1445)
- Override control with PIDConR (OverrideR) (Page 1447)

Application scenario in the example project:

- Process simulation including noise generator (Page 1456)

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status parameter

This block does not provide the `Status` parameter.

See also

SelA02In block diagram (Page 1362)
SelA02In I/Os (Page 1361)
SelA02In messaging (Page 1360)
SelA02In error handling (Page 1360)
SelA02In functions (Page 1358)
SelA02In modes (Page 1358)

14.4.2 SelA02In modes

SelA02In modes

This block does not provide any modes.

See also

SelA02In block diagram (Page 1362)

SelA02In I/Os (Page 1361)

SelA02In messaging (Page 1360)

SelA02In error handling (Page 1360)

SelA02In functions (Page 1358)

Description of SelA02In (Page 1356)

14.4.3 SelA02In functions

Functions of SelA02In

The functions for this block are listed below.

Select input parameter

The `SelMode` parameter can affect the selection as follows:

- `SelMode` ≤ 0: Selection depends on the parameter `Sel_In2`
  - `Sel_In2` = FALSE: Input parameter `In1` is written with its signal status to output parameter `Out`.
  - `Sel_In2` = TRUE: Input parameter `In2` is written with its signal status to output parameter `Out`.
- `SelMode = 1`: The input parameter with the lower value (`In1` or `In2`) is written to the `Out` output parameter, together with its signal status.
- `SelMode ≥ 2`: The input parameter with the higher value (`In1` or `In2`) is written to the `Out` output parameter, together with its signal status.

The signal status of `Sel_In2` is output at the `In2Selected` output parameter.

See also

Forming and outputting signal status for blocks (Page 88)
SelA02In block diagram (Page 1362)
SelA02In I/Os (Page 1361)
SelA02In messaging (Page 1360)
SelA02In error handling (Page 1360)
SelA02In modes (Page 1358)
Description of SelA02In (Page 1356)

14.4.4 SelA02In error handling

SelA02In error handling

The block does not report any errors.

See also

SelA02In block diagram (Page 1362)
SelA02In I/Os (Page 1361)
SelA02In messaging (Page 1360)
SelA02In functions (Page 1358)
SelA02In modes (Page 1358)
Description of SelA02In (Page 1356)

14.4.5 SelA02In messaging

Message functionality

The block does not have any Message response.

See also

SelA02In block diagram (Page 1362)
SelA02In I/Os (Page 1361)
SelA02In error handling (Page 1360)
SelA02In functions (Page 1358)
SelA02In modes (Page 1358)
Description of SelA02In (Page 1356)

14.4.6 SelA02In I/Os

SelA02In I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
In1	Analog process value 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In2	Analog process value 2	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
SelMode	Selecting the function 0 = Select with Sel_In2 1 = Minimum 2 = Maximum	INT	0
Sel_In2	Selecting the input parameter 0 = In1 1 = In2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
In2Selected	Selected input parameter 0 = In1 1 = In2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
Out	Analog process value	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

See also

SelA02In block diagram (Page 1362)
SelA02In messaging (Page 1360)
SelA02In error handling (Page 1360)
SelA02In functions (Page 1358)
SelA02In modes (Page 1358)
Description of SelA02In (Page 1356)

14.4.7 SelA02In block diagram

SelA02In block diagram

A block diagram is not provided for this block.

See also

SelA02In I/Os (Page 1361)
SelA02In messaging (Page 1360)
SelA02In error handling (Page 1360)
SelA02In functions (Page 1358)
SelA02In modes (Page 1358)
Description of SelA02In (Page 1356)

14.5 SelA16In - Output of 16 analog values

14.5.1 Description of SelA16In

Object name (type + number) and family

Type + number: FB 1888

Family: LogicAn

Area of application for SelA16In

The block is used for the following applications:

- Selecting one of 16 analog values and switching it through to the output.

How it works

The block writes the value of one of the input parameters `In01` through `In16` to the output parameter `Out`. The selection is performed via the `SelInt` input parameter.

The signal status of the selected input parameter is written to the signal status of the output parameter `Out`. There is no additional signal status evaluation.

The unit `InxxUnit` of the selected input parameter `Inxx` (x = 1 ...16) is written to the output parameter `OutUnit`.

Configuration

Use the CFC editor to install the block in any OB.

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status1 parameter

For a description of the individual parameters, see the SelA16In I/Os (Page 1369) section.

Status bit	Parameter
0 - 2	Not used
3	`OosAct.Value`
4	`OosLi.Value`
5	Not used
6	`OnAct.Value`
7 - 11	Not used
12 – 27	Used to highlight the green line in the standard view
28 - 31	Not used

See also

SelA16In block diagram (Page 1373)

SelA16In messaging (Page 1369)

SelA16In error handling (Page 1368)

SelA16In functions (Page 1366)

SelA16In modes (Page 1365)

14.5.2 SelA16In modes

SelA16In modes

The block can be operated using the following modes:

- On (Page 27)
- Out of service (Page 27)

"On"

You will find general information about the "On" mode in the section On (Page 27).

"Out of service"

You will find general information about the "Out of service" mode in the section Out of service (Page 27).

See also

SelA16In block diagram (Page 1373)

SelA16In I/Os (Page 1369)

SelA16In messaging (Page 1369)

SelA16In error handling (Page 1368)

SelA16In functions (Page 1366)

Description of SelA16In (Page 1363)

14.5.3 SelA16In functions

Functions of SelA16In

The functions for this block are listed below.

Opening additional faceplates

This block provides the standard function Calling further faceplates (Page 43).

Operator control permissions via parameter OS_Perm

The block has the following Operator permissions (Page 45) for the OS_Perm parameter:

Bit	Function
0	Not allocated
1	1 = Operator can switch to "On" mode
2	Not allocated
3	1 = Operator can switch to out of service mode
4	1 = Operator can set input 1
5	1 = Operator can set input 2
6	1 = Operator can set input 3
7	1 = Operator can set input 4
8	1 = Operator can set input 5
9	1 = Operator can set input 6
10	1 = Operator can set input 7
11	1 = Operator can set input 8
12	1 = Operator can set input 9
13	1 = Operator can set input 10
14	1 = Operator can set input 11
15	1 = Operator can set input 12
16	1 = Operator can set input 13
17	1 = Operator can set input 14
18	1 = Operator can set input 15
19	1 = Operator can set input 16
20 - 31	Not allocated

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

The worst signal status `ST_Worst` for the block is formed from the following parameters:

- `In1.ST`

etc. to

- `In16.ST`

Selecting a unit of measure

This block provides the standard function Selecting a unit of measure (Page 53).

Configurable reactions using the `Feature` parameter

You can find an overview of all reactions provided by the `Feature` parameter in the Functions that can be set via the Feature I/O (Page 186) section. The following functionality is available for this block at the relevant bits:

Bit	Function
0	Not used
1	Reaction to the out of service mode (Page 199)
2 - 31	Not used

See also

SelA16In block diagram (Page 1373)

SelA16In I/Os (Page 1369)

SelA16In messaging (Page 1369)

SelA16In error handling (Page 1368)

SelA16In modes (Page 1365)

Description of SelA16In (Page 1363)

Setting the startup response (Page 187)

14.5.4 SelA16In error handling

SelA16In troubleshooting

Refer to Error handling (Page 117) section of the basic instructions to learn how to troubleshoot all the blocks.

The following errors can be displayed for this block:

- Error numbers

Overview of error numbers

The `ErrorNum` I/O can be used to output the following error numbers:

Error number	Meaning of the error number
-1	Predefined value when inserting the block, the block is not processed.
0	There is no error.

See also

SelA16In block diagram (Page 1373)

SelA16In I/Os (Page 1369)

SelA16In messaging (Page 1369)

SelA16In functions (Page 1366)

SelA16In modes (Page 1365)

Description of SelA16In (Page 1363)

14.5.5 SelA16In messaging

Message functionality

The block does not have any Message response.

See also

SelA16In block diagram (Page 1373)
SelA16In I/Os (Page 1369)
SelA16In error handling (Page 1368)
SelA16In functions (Page 1366)
SelA16In modes (Page 1365)
Description of SelA16In (Page 1363)

14.5.6 SelA16In I/Os

SelA16In I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	0
Feature	I/O for additional functions (Page 1366)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 0 • 0 • 0
In01	Analog input parameter 1	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In02	Analog input parameter 2	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In03	Analog input parameter 3	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In04	Analog input parameter 4	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80

Parameter	Description	Type	Default
In05	Analog input parameter 5	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In06	Analog input parameter 6	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In07	Analog input parameter 7	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In08	Analog input parameter 8	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In09	Analog input parameter 9	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In10	Analog input parameter 10	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In11	Analog input parameter 11	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In12	Analog input parameter 12	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In13	Analog input parameter 13	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In14	Analog input parameter 14	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In15	Analog input parameter 15	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In16	Analog input parameter 16	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
In01Unit	Unit for parameter In01	INT	1001
In02Unit	Unit for parameter In02	INT	1001
In03Unit	Unit for parameter In03	INT	1001
In04Unit	Unit for parameter In04	INT	1001
In05Unit	Unit for parameter In05	INT	1001

Parameter	Description	Type	Default
In06Unit	Unit for parameter In06	INT	1001
In07Unit	Unit for parameter In07	INT	1001
In08Unit	Unit for parameter In08	INT	1001
In09Unit	Unit for parameter In09	INT	1001
In10Unit	Unit for parameter In10	INT	1001
In11Unit	Unit for parameter In11	INT	1001
In12Unit	Unit for parameter In12	INT	1001
In13Unit	Unit for parameter In13	INT	1001
In14Unit	Unit for parameter In14	INT	1001
In15Unit	Unit for parameter In15	INT	1001
In16Unit	Unit for parameter In16	INT	1001
OnOp	1 = "On" mode via operator	BOOL	0
OosLi	1 = Out of service, via interconnection or SFC (0-1 edge transition)	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OosOp	1 = Out of service, via OS operator	BOOL	0
OS_Perm	I/O for operator control permissions (Page 1366)	STRUCT • Bit 0: BOOL • ... • Bit 31: BOOL	- • 1 • 1 • 1
SelFp1	Call a block saved in this parameter as an additional faceplate (Page 43) in the standard view	ANY	-
SelFp2	Call a block saved in this parameter as an additional faceplate (Page 43) in the preview	ANY	-
SelInt	Selecting the input parameter: 1 = In01 selected, 16 = In16 selected. You cannot select anything outside the range of 1 to 16.	BOOL	1
UserStatus	Freely assignable user area for status word (bit 24 to bit 31)	BYTE	16#0

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
ErrorNum	Output of current error number. For error numbers that can be output by this block, see SelA16In error handling (Page 1368).	INT	-1
InSelected	Selected input parameter: 1 = In01 selected, 16 = In16 selected.	STRUCT • Value: INT • ST: BYTE	- • 1 • 16#80
OnAct	1 = "On" mode enabled	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
OosAct	1 = Block is "Out of service"	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
OS_PermLog	Display of OS_Perm with settings changed by the block algorithm	DWORD	16#FFFFFFFF
OS_PermOut	Display of OS_Perm	DWORD	16#FFFFFFFF
Out	Output	STRUCT • Value: REAL • ST: BYTE	- • 0.0 • 16#80
OutUnit	Unit for parameter Out	INT	1001
ST_Worst	Worst signal status	BYTE	16#80
Status	Status word (Page 1363)	DWORD	16#00

See also

SelA16In block diagram (Page 1373)

SelA16In messaging (Page 1369)

SelA16In modes (Page 1365)

14.5.7 SelA16In block diagram

SelA16In block diagram

A block diagram is not provided for this block.

See also

SelA16In I/Os (Page 1369)
SelA16In messaging (Page 1369)
SelA16In error handling (Page 1368)
SelA16In functions (Page 1366)
SelA16In modes (Page 1365)
Description of SelA16In (Page 1363)

14.5.8 Operator control and monitoring

14.5.8.1 Views of SelA16In

Views of the SelA16In block

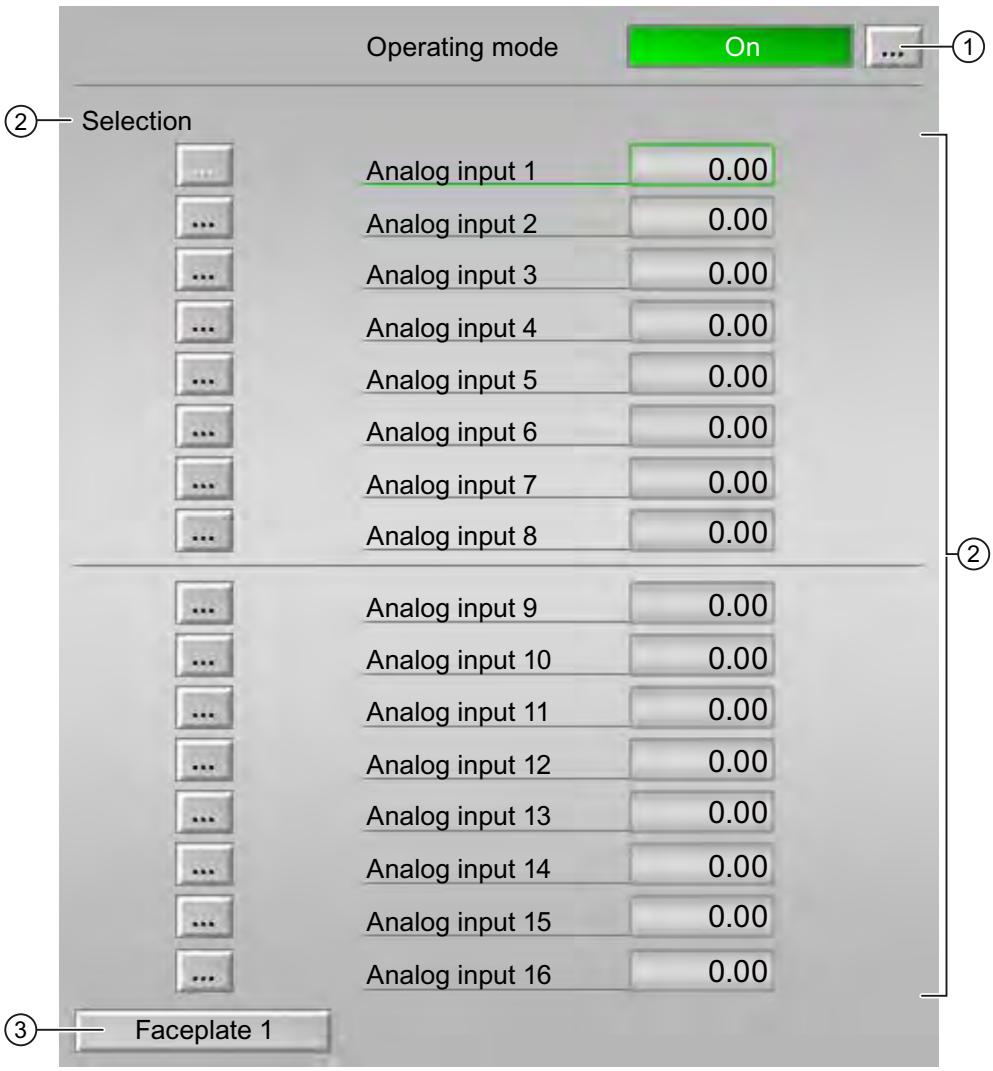
The block SelA16In provides the following views:

- SelA16In standard view (Page 1375)
- SelA16In preview (Page 1377)
- Memo view (Page 171)
- Block icon for SelA16In (Page 1379)

Refer to the Structure of the faceplate (Page 123) and Block icon structure (Page 174) sections for general information about the faceplate and block icon.

14.5.8.2 SelA16In standard view

SelA16In standard view



(1) Display and switch the operating mode

This area provides information on the currently valid operating mode. The following operating modes can be shown here:

- On (Page 27)
- Out of service (Page 27)

Refer to the Switching operating states and operating modes (Page 127) section for information on switching the operating mode.

(2) Switch analog values

This area shows you the analog values connected in the ES for this block.

You can find additional information on this in Switching operating states and operating modes (Page 127).

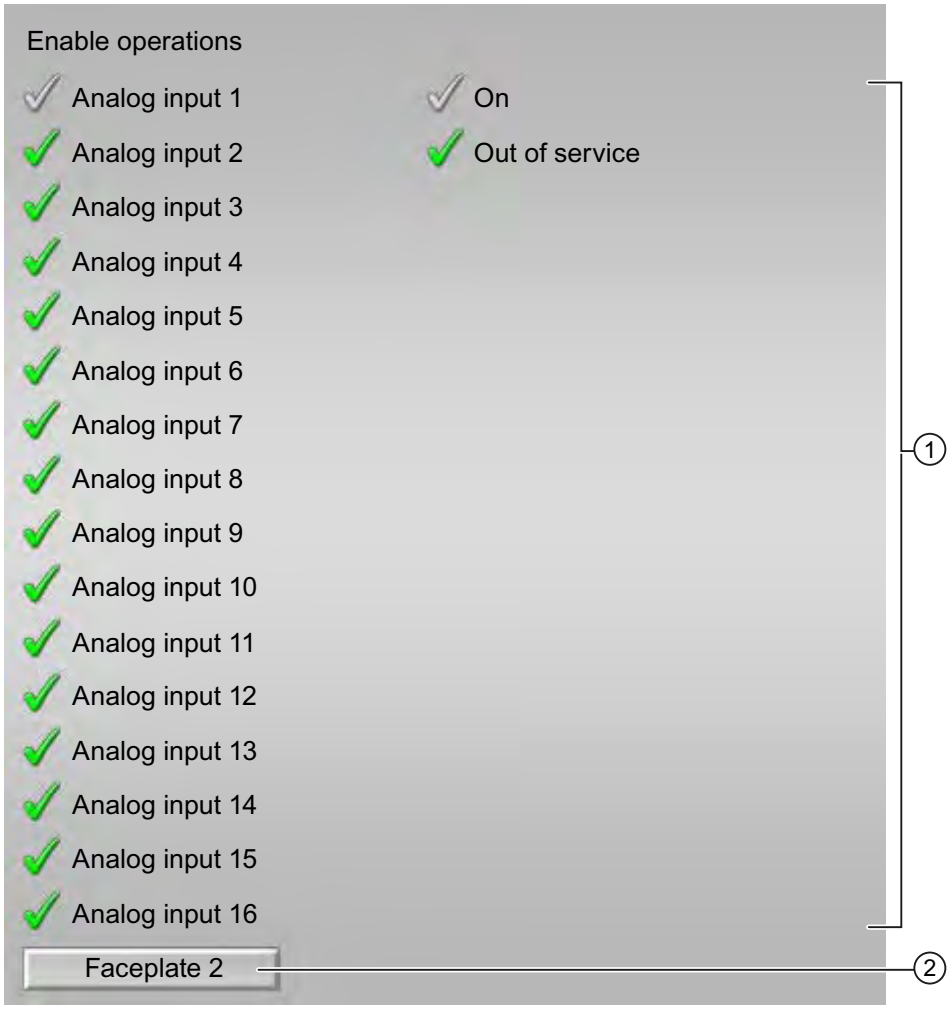
(3) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

You can find additional information on this in the Calling further faceplates (Page 43) section.

14.5.8.3 SelA16In preview

SelA16In preview



(1) Enable operations

This area shows all operations for which special operator control permissions are assigned. They depend on the configuration in the engineering system (ES) that applies to this block.

Symbols for the operator control enables:

- Green check mark: the OS operator can control this parameter
- Gray check mark: the OS operator cannot control this parameter at this time due to the process
- Red cross: the OS operator cannot control this parameter due to the configured operator control permissions

The following operator control enables are shown here:

- Analog input 1 to 16: You can switch to this analog input.
- On: You can switch to On operating mode.
- Out of service: You can switch to out of service mode.

(2) Navigation button for switching to the standard view of any faceplate

Use this navigation button to reach the standard view of a block configured in the engineering system. The visibility of this navigation button depends on the configuration in the engineering system (ES).

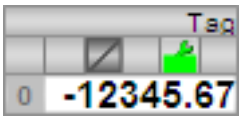
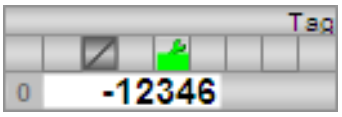
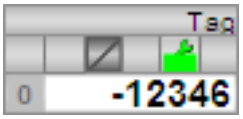
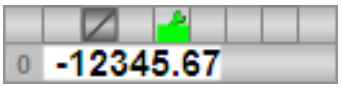
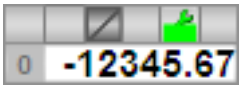
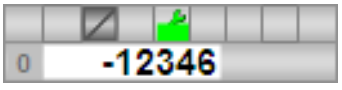
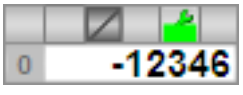
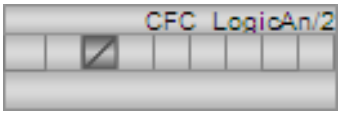
You can find additional information on this in the Calling further faceplates (Page 43) section.

14.5.8.4 Block icon for SelA16In

Properties of the SelA16In block icon

A variety of block icons are available with the following functions:

- Process tag type
- Operating modes
- Signal status, maintenance release
- Memo display
- Display of the selected analog value

Icons	Selection of the block icon in CFC	Special features
	1	Block icon in the full display
	2	
	3	
	4	
	5	
	6	
	7	
	8	
	-	Block icon in the "Out of service" mode (example with type 1 block icon)

Additional information on the block icon and the control options in the block icon is available in the following sections:

- Configuring the block icons (Page 180)
- Block icon structure (Page 174)
- Operation via the block icon (Page 181).

Logical digital blocks

15

15.1 And04 - Forming an AND signal from 4 binary input signals

15.1.1 Description of And04

Object name (type + number) and family

Type + number: FC 304

Family: LogicDi

Area of application for And04

The block is used for the following applications:

- Forming an AND output signal from four binary input values

How it works

Four input parameters are combined into one output value `Out` using the AND function ("and-ing").

You can use this block to e.g. start or stop a device if all incoming signals are the same.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status1 parameter

This block does not provide the `Status` parameter.

See also

And04 block diagram (Page 1385)
And04 I/Os (Page 1384)
And04 messaging (Page 1383)
And04 error handling (Page 1383)
And04 functions (Page 1382)
And04 modes (Page 1382)

15.1.2 And04 modes

And04 modes

This block does not provide any modes.

See also

And04 block diagram (Page 1385)
And04 I/Os (Page 1384)
And04 messaging (Page 1383)
And04 error handling (Page 1383)
And04 functions (Page 1382)
Description of And04 (Page 1381)
Out of service (Page 27)

15.1.3 And04 functions

Functions of And04

There are no other functions for this block.

See also

And04 block diagram (Page 1385)
And04 I/Os (Page 1384)
And04 messaging (Page 1383)
And04 error handling (Page 1383)
And04 modes (Page 1382)
Description of And04 (Page 1381)

15.1.4 And04 error handling

And04 troubleshooting

The block does not report any errors.

See also

And04 block diagram (Page 1385)
And04 I/Os (Page 1384)
And04 messaging (Page 1383)
And04 functions (Page 1382)
And04 modes (Page 1382)
Description of And04 (Page 1381)

15.1.5 And04 messaging

Message behavior

The block does not have any message functionality.

See also

And04 block diagram (Page 1385)
And04 I/Os (Page 1384)
And04 error handling (Page 1383)
And04 functions (Page 1382)
And04 modes (Page 1382)
Description of And04 (Page 1381)

15.1.6 And04 I/Os

And04 I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
In1	Input 1	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
In2	Input 2	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
In3	Input 3	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
In4	Input 4	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
Out	Output	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

And04 block diagram (Page 1385)
And04 messaging (Page 1383)
And04 error handling (Page 1383)
And04 functions (Page 1382)
And04 modes (Page 1382)
Description of And04 (Page 1381)

15.1.7 And04 block diagram

And04 block diagram

A block diagram is not provided for this block.

See also

And04 I/Os (Page 1384)
And04 messaging (Page 1383)
And04 error handling (Page 1383)
And04 functions (Page 1382)
And04 modes (Page 1382)
Description of And04 (Page 1381)

15.2 And08 - Forming an AND signal from 8 binary input signals

15.2.1 Description of And08

Object name (type + number) and family

Type + number: FC 305

Family: LogicDi

Area of application for And08

The block is used for the following applications:

- Forming an AND output signal from eight binary input values

How it works

Eight input parameters are combined into one output value `Out` using the AND function ("and-ing") .

You can use this block to e.g. start or stop a device if all incoming signals are the same.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status1 parameter

This block does not provide the `Status` parameter.

See also

And08 block diagram (Page 1390)

And08 I/Os (Page 1389)

And08 messaging (Page 1388)

And08 error handling (Page 1388)

And08 functions (Page 1387)

And08 modes (Page 1387)

15.2.2 And08 modes

And08 modes

This block does not provide any modes.

See also

And08 block diagram (Page 1390)
And08 I/Os (Page 1389)
And08 messaging (Page 1388)
And08 error handling (Page 1388)
And08 functions (Page 1387)
Description of And08 (Page 1386)
Out of service (Page 27)

15.2.3 And08 functions

Functions of And08

There are no other functions for this block.

See also

And08 block diagram (Page 1390)
And08 I/Os (Page 1389)
And08 messaging (Page 1388)
And08 error handling (Page 1388)
And08 modes (Page 1387)
Description of And08 (Page 1386)

15.2.4 And08 error handling

And08 troubleshooting

The block does not report any errors.

See also

And08 block diagram (Page 1390)
And08 I/Os (Page 1389)
And08 messaging (Page 1388)
And08 functions (Page 1387)
And08 modes (Page 1387)
Description of And08 (Page 1386)

15.2.5 And08 messaging

Messaging

This block does not have any message functionality.

See also

And08 block diagram (Page 1390)
And08 I/Os (Page 1389)
And08 error handling (Page 1388)
And08 functions (Page 1387)
And08 modes (Page 1387)
Description of And08 (Page 1386)

15.2.6 And08 I/Os

And08 I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
In1	Input 1	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
In2	Input 2	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
In3	Input 3	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
In4	Input 4	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
In5	Input 5	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
In6	Input 6	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
In7	Input 7	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80
In8	Input 8	STRUCT • Value: BOOL • ST: BYTE	- • 1 • 16#80

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
Out	Output	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

And08 block diagram (Page 1390)
And08 messaging (Page 1388)
And08 error handling (Page 1388)
And08 functions (Page 1387)
And08 modes (Page 1387)
Description of And08 (Page 1386)

15.2.7 And08 block diagram

And08 block diagram

A block diagram is not provided for this block.

See also

And08 I/Os (Page 1389)
And08 messaging (Page 1388)
And08 error handling (Page 1388)
And08 functions (Page 1387)
And08 modes (Page 1387)
Description of And08 (Page 1386)

15.3 Or04 - Forming an OR signal from 4 binary input signals

15.3.1 Description of Or04

Object name (type + number) and family

Type + number: FC 313

Family: LogicDi

Area of application for Or04

The block is used for the following applications:

- Forming an OR output signal from four binary input values

How it works

Four input parameters are combined into one output value `Out` using the OR function ("or-ing") .

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

For the Or04 block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Cascade control (Page 1442)
- Cascade control with PIDConR (CascadeR) (Page 1444)
- Dosing (DoseLean) (Page 1449)
- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 1433)
- PID - control with operating-point-oriented parameter control (GainScheduling) (Page 1431)
- Model-based predictive control (ModPreCon) (Page 1447)
- Override control (Page 1445)
- Override control with PIDConR (OverrideR) (Page 1447)
- PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 1430)
- PIDConR with safety logic and control loop monitoring (PIDConR_ConPerMon) (Page 1431)
- Ratio control (Page 1440)
- Ratio control with PIDConR (RatioR) (Page 1441)
- PID controller with Smith predictor (SmithPredictorControl) (Page 1436)

- Step controller with assigned actuator block and position feedback (StepControlActor) (Page 1437)
- Valve (ValveLean) (Page 1453)

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status1 parameter

This block does not provide the `Status` parameter.

See also

Or04 block diagram (Page 1396)

Or04 I/Os (Page 1395)

Or04 messaging (Page 1394)

Or04 error handling (Page 1394)

Or04 functions (Page 1393)

Or04 modes (Page 1393)

15.3.2 Or04 modes

Or04 modes

This block does not provide any modes.

See also

Or04 block diagram (Page 1396)
Or04 I/Os (Page 1395)
Or04 messaging (Page 1394)
Or04 error handling (Page 1394)
Or04 functions (Page 1393)
Description of Or04 (Page 1391)
Out of service (Page 27)

15.3.3 Or04 functions

Functions of Or04

There are no other functions for this block.

See also

Or04 block diagram (Page 1396)
Or04 I/Os (Page 1395)
Or04 messaging (Page 1394)
Or04 error handling (Page 1394)
Or04 modes (Page 1393)
Description of Or04 (Page 1391)

15.3.4 Or04 error handling

Or04 troubleshooting

The block does not report any errors.

See also

Or04 block diagram (Page 1396)
Or04 I/Os (Page 1395)
Or04 messaging (Page 1394)
Or04 functions (Page 1393)
Or04 modes (Page 1393)
Description of Or04 (Page 1391)

15.3.5 Or04 messaging

Message functionality

The block does not have any message functionality.

See also

Or04 block diagram (Page 1396)
Or04 I/Os (Page 1395)
Or04 error handling (Page 1394)
Or04 functions (Page 1393)
Or04 modes (Page 1393)
Description of Or04 (Page 1391)

15.3.6 Or04 I/Os

Or04 I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
In1	Value 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
In2	Value 2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
In3	Value 3	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
In4	Value 4	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
Out	Output	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

Or04 block diagram (Page 1396)
Or04 messaging (Page 1394)
Or04 error handling (Page 1394)
Or04 functions (Page 1393)
Or04 modes (Page 1393)
Description of Or04 (Page 1391)

15.3.7 Or04 block diagram

Or04 block diagram

A block diagram is not provided for this block.

See also

Or04 I/Os (Page 1395)
Or04 messaging (Page 1394)
Or04 error handling (Page 1394)
Or04 functions (Page 1393)
Or04 modes (Page 1393)
Description of Or04 (Page 1391)

15.4 Or08 - Forming an OR signal from 8 binary input signals

15.4.1 Description of Or08

Object name (type + number) and family

Type + number: FC 314

Family: LogicDi

Area of application for Or08

The block is used for the following applications:

- Forming an OR output signal from eight binary input values

How it works

Eight input parameters are combined into one output value `Out` using the OR function ("oring") .

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

For the Or08 block, the Advanced Process Library contains templates for process tag types as examples with various application scenarios for this block.

Examples of process tag types:

- Monitoring eight digital process tags (Digital8Monitoring) (Page 1448)
- Step controller with assigned actuator block and position feedback (StepControlActor) (Page 1437)
- Step controller with direct access to the actuator and without position feedback (StepControlDirect) (Page 1437)
- Two-speed motor (Motor2Speed) (Page 1450)
- Reversing motor (MotorReversible) (Page 1450)
- Reversing motor with controllable speed (MotorSpeedControlled) (Page 1451)
- Two-way valve (Valve2Way) (Page 1453)
- Motor valve (ValveMotor) (Page 1454)

Startup characteristics

The block does not have any startup characteristics.

Status word allocation for Status1 parameter

This block does not provide the `Status` parameter.

See also

Or08 block diagram (Page 1402)
Or08 I/Os (Page 1401)
Or08 messaging (Page 1400)
Or08 error handling (Page 1399)
Or08 functions (Page 1399)
Or08 modes (Page 1398)

15.4.2 Or08 modes

Or08 modes

This block does not provide any modes.

See also

Or08 block diagram (Page 1402)
Or08 I/Os (Page 1401)
Or08 messaging (Page 1400)
Or08 error handling (Page 1399)
Or08 functions (Page 1399)
Description of Or08 (Page 1397)
Out of service (Page 27)

15.4.3 Or08 functions

Functions of Or08

There are no other functions for this block.

See also

Or08 block diagram (Page 1402)
Or08 I/Os (Page 1401)
Or08 messaging (Page 1400)
Or08 error handling (Page 1399)
Or08 modes (Page 1398)
Description of Or08 (Page 1397)

15.4.4 Or08 error handling

Or08 troubleshooting

The block does not report any errors.

See also

Or08 block diagram (Page 1402)
Or08 I/Os (Page 1401)
Or08 messaging (Page 1400)
Or08 functions (Page 1399)
Or08 modes (Page 1398)
Description of Or08 (Page 1397)

15.4.5 Or08 messaging

Messaging

The block does not have any message functionality.

See also

Or08 block diagram (Page 1402)
Or08 I/Os (Page 1401)
Or08 error handling (Page 1399)
Or08 functions (Page 1399)
Or08 modes (Page 1398)
Description of Or08 (Page 1397)

15.4.6 Or08 I/Os

Or08 I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
In1	Value 1	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
In2	Value 2	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
In3	Value 3	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
In4	Value 4	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
In5	Value 5	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
In6	Value 6	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
In7	Value 7	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80
In8	Value 8	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
Out	Output	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

Or08 block diagram (Page 1402)
Or08 messaging (Page 1400)
Or08 error handling (Page 1399)
Or08 functions (Page 1399)
Or08 modes (Page 1398)
Description of Or08 (Page 1397)

15.4.7 Or08 block diagram

Or08 block diagram

A block diagram is not provided for this block.

See also

Or08 I/Os (Page 1401)
Or08 messaging (Page 1400)
Or08 error handling (Page 1399)
Or08 functions (Page 1399)
Or08 modes (Page 1398)
Description of Or08 (Page 1397)

15.5 Not01 - Inversion of an input signal

15.5.1 Description of Not01

Object name (type + number) and family

Type + number: FC 335

Family: LogicDi

Area of application for Not01

The block is used for the following applications:

- Inversion of an input signal

How it works

The block inverts the binary signal available at the `In` input parameter and writes the result to its output parameter `Out`.

For the Not01 block, the Advanced Process Library contains a template for process tag types as an example with an application scenario for this block.

Example of process tag types:

- Step controller with assigned actuator block and position feedback (StepControlActor) (Page 1437)

See also

Not01 block diagram (Page 1407)

Not01 I/Os (Page 1406)

Not01 messaging (Page 1405)

Not01 error handling (Page 1405)

Not01 functions (Page 1404)

Not01 modes (Page 1404)

15.5.2 Not01 modes

Not01 operating modes

This block does not provide any operating modes.

See also

Not01 block diagram (Page 1407)
Not01 I/Os (Page 1406)
Not01 messaging (Page 1405)
Not01 error handling (Page 1405)
Not01 functions (Page 1404)
Description of Not01 (Page 1403)

15.5.3 Not01 functions

Functions of Not01

There are no other functions for this block.

See also

Not01 block diagram (Page 1407)
Not01 I/Os (Page 1406)
Not01 messaging (Page 1405)
Not01 error handling (Page 1405)
Not01 modes (Page 1404)
Description of Not01 (Page 1403)

15.5.4 Not01 error handling

Not01 troubleshooting

The block does not report any errors.

See also

Not01 block diagram (Page 1407)
Not01 I/Os (Page 1406)
Not01 messaging (Page 1405)
Not01 functions (Page 1404)
Not01 modes (Page 1404)
Description of Not01 (Page 1403)

15.5.5 Not01 messaging

Messaging

The block does not offer messaging.

See also

Not01 block diagram (Page 1407)
Not01 I/Os (Page 1406)
Not01 error handling (Page 1405)
Not01 functions (Page 1404)
Not01 modes (Page 1404)
Description of Not01 (Page 1403)

15.5.6 Not01 I/Os

Not01 I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
In	Input value	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
Out	Output	STRUCT • Value: BOOL • ST: BYTE	- • 0 • 16#80

See also

Not01 block diagram (Page 1407)
Not01 messaging (Page 1405)
Not01 error handling (Page 1405)
Not01 functions (Page 1404)
Not01 modes (Page 1404)
Description of Not01 (Page 1403)

15.5.7 Not01 block diagram

Not01 block diagram

A block diagram is not provided for this block.

See also

Not01 I/Os (Page 1406)
Not01 messaging (Page 1405)
Not01 error handling (Page 1405)
Not01 functions (Page 1404)
Not01 modes (Page 1404)
Description of Not01 (Page 1403)

Maintenance blocks

16

16.1 MuxMST - Determination of the worst maintenance status

16.1.1 Description of MuxMST

Object name (type + number) and family

Type + number: FB 1861

Family: Maint

Area of application for MuxMST

The block is used for the following applications:

- Determination of the worst maintenance status (from a maximum of 10 states)

How it works

The block determines the worst of several maintenance states. Each status is made available to the block via interconnection to the In_x input parameter (x = 1 to 10). The states are compared and the highest value written to the `Out` output parameter.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

Startup characteristics

During startup, the In_x input parameters (x = 1 to 10) and output parameter are reset to their defaults.

Status word allocation for Status parameter

This block does not provide the `Status` parameter.

See also

MuxMST block diagram (Page 1413)
MuxMST I/Os (Page 1412)
MuxMST messaging (Page 1411)
MuxMST error handling (Page 1411)
MuxMST functions (Page 1410)
MuxMST modes (Page 1410)

16.1.2 MuxMST modes

MuxMST operating modes

This block does not provide any operating modes.

See also

MuxMST block diagram (Page 1413)
MuxMST I/Os (Page 1412)
MuxMST messaging (Page 1411)
MuxMST error handling (Page 1411)
MuxMST functions (Page 1410)
Description of MuxMST (Page 1409)
Out of service (Page 27)

16.1.3 MuxMST functions

Functions of MuxMST

This block provides no other functions.

See also

MuxMST block diagram (Page 1413)
MuxMST I/Os (Page 1412)
MuxMST messaging (Page 1411)
MuxMST error handling (Page 1411)
MuxMST modes (Page 1410)
Description of MuxMST (Page 1409)

16.1.4 MuxMST error handling

MuxMST troubleshooting

The block does not report any errors.

See also

MuxMST block diagram (Page 1413)
MuxMST I/Os (Page 1412)
MuxMST messaging (Page 1411)
MuxMST functions (Page 1410)
MuxMST modes (Page 1410)
Description of MuxMST (Page 1409)

16.1.5 MuxMST messaging

Messaging

This block does not have any message functionality.

See also

MuxMST block diagram (Page 1413)
MuxMST I/Os (Page 1412)
MuxMST error handling (Page 1411)
MuxMST functions (Page 1410)
MuxMST modes (Page 1410)
Description of MuxMST (Page 1409)

16.1.6 MuxMST I/Os

MuxMST I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
In1	Input for signal of maintenance status 1	DWORD	16#00
In2	Input for signal of maintenance status 2	DWORD	16#00
In3	Input for signal of maintenance status 3	DWORD	16#00
In4	Input for signal of maintenance status 4	DWORD	16#00
In5	Input for signal of maintenance status 5	DWORD	16#00
In6	Input for signal of maintenance status 6	DWORD	16#00
In7	Input for signal of maintenance status 7	DWORD	16#00
In8	Input for signal of maintenance status 8	DWORD	16#00
In9	Input for signal of maintenance status 9	DWORD	16#00
In10	Input for signal of maintenance status 10	DWORD	16#00

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
Out	Output signal with the worst signal status	DWORD	16#00

See also

MuxMST block diagram (Page 1413)
MuxMST messaging (Page 1411)
MuxMST error handling (Page 1411)
MuxMST functions (Page 1410)
MuxMST modes (Page 1410)
Description of MuxMST (Page 1409)

16.1.7 MuxMST block diagram

MuxMST block diagram

A block diagram is not provided for this block.

See also

MuxMST I/Os (Page 1412)
MuxMST messaging (Page 1411)
MuxMST error handling (Page 1411)
MuxMST functions (Page 1410)
MuxMST modes (Page 1410)
Description of MuxMST (Page 1409)

16.2 MuxST- Determination of the worst signal status

16.2.1 Description of MuxST

Object name (type + number) and family

Type + number: FB 1862

Family: Maint

Area of application for MuxST

The block is used for the following applications:

- Determination of the worst signal status (from a maximum of 10 states)

How it works

The block determines the worst of several signal states. Each status is made available to the block via interconnection to the `In`$_x$ input parameter (x = 1 to 10). The states are compared and the value with the highest priority is written to the `Out` output parameter.

Please also refer to the section for information on priorities.

Configuration

Use the CFC editor to install the block in a cyclic interrupt OB (OB30 to OB38).

Startup characteristics

During startup, the `In`$_x$ input parameters (x = 1 to 10) and output parameter are reset to their defaults.

Status word allocation for `Status1` parameter

This block does not provide the `Status` parameter.

See also

MuxST block diagram (Page 1418)

MuxST I/Os (Page 1417)

MuxST messaging (Page 1416)

MuxST error handling (Page 1416)

MuxST functions (Page 1415)

MuxST modes (Page 1415)

16.2.2 MuxST modes

MuxST operating modes

This block does not provide any operating modes.

See also

MuxST block diagram (Page 1418)

MuxST I/Os (Page 1417)

MuxST messaging (Page 1416)

MuxST error handling (Page 1416)

MuxST functions (Page 1415)

Description of MuxST (Page 1414)

Out of service (Page 27)

16.2.3 MuxST functions

Functions of MuxST

The functions for this block are listed below.

Selecting signals for processing

This block provides the standard function Selecting signals for processing (Page 93).

Forming the signal status for blocks

This block provides the standard function Forming and outputting signal status for blocks (Page 88).

See also

MuxST block diagram (Page 1418)

MuxST I/Os (Page 1417)

MuxST messaging (Page 1416)

MuxST error handling (Page 1416)

MuxST modes (Page 1415)

Description of MuxST (Page 1414)

16.2.4 MuxST error handling

MuxST troubleshooting

The block does not report any errors.

See also

MuxST block diagram (Page 1418)
MuxST I/Os (Page 1417)
MuxST messaging (Page 1416)
MuxST functions (Page 1415)
MuxST modes (Page 1415)
Description of MuxST (Page 1414)

16.2.5 MuxST messaging

Message behavior

The block does not have any message behavior.

See also

MuxST block diagram (Page 1418)
MuxST I/Os (Page 1417)
MuxST error handling (Page 1416)
MuxST functions (Page 1415)
MuxST modes (Page 1415)
Description of MuxST (Page 1414)

16.2.6 MuxST I/Os

MuxST I/Os

Input parameters

Parameter	Description	Type	Default
EN	1 = Called block will be processed	BOOL	1
In1	Input for signal of signal status 1	BYTE	16#80
In2	Input for signal of signal status 2	BYTE	16#80
In3	Input for signal of signal status 3	BYTE	16#80
In4	Input for signal of signal status 4	BYTE	16#80
In5	Input for signal of signal status 5	BYTE	16#80
In6	Input for signal of signal status 6	BYTE	16#80
In7	Input for signal of signal status 7	BYTE	16#80
In8	Input for signal of signal status 8	BYTE	16#80
In9	Input for signal of signal status 9	BYTE	16#80
In 10	Input for signal of signal status 10	BYTE	16#80
SelInput	Selection of the input parameter for forming the worst signal status	INT	2
SelPrio	Setting of the prioritization for forming the worst signal status	INT	0

Output parameters

Parameter	Description	Type	Default
ENO	1 = Block algorithm completed without errors	BOOL	0
Out	Output of the worst signal status	BYTE	16#80

See also

MuxST block diagram (Page 1418)
MuxST messaging (Page 1416)
MuxST error handling (Page 1416)
MuxST functions (Page 1415)
MuxST modes (Page 1415)
Description of MuxST (Page 1414)

16.2.7 MuxST block diagram

MuxST block diagram

A block diagram is not provided for this block.

See also

MuxST I/Os (Page 1417)
MuxST messaging (Page 1416)
MuxST error handling (Page 1416)
MuxST functions (Page 1415)
MuxST modes (Page 1415)
Description of MuxST (Page 1414)

System blocks

17

17.1 AddInt64 - Addition of two 64-bit integer variables

17.1.1 Description of AddInt64

Object name (type + number) and family

Type + number: FC 302

Family: System

Area of application for AddInt64

The block is used for the following applications:

- Addition of two 64-bit integer variables

This block is used by the block PIDConR for calculations that are twice as accurate (64 bit number format).

Therefore this block will not be described in detail.

17.2 AddR64 - Addition of two 64-bit REAL variables

17.2.1 Description of AddR64

Object name (type + number) and family

Type + number: FC 303

Family: System

Area of application for AddR64

The block is used for the following applications:

- Addition of two 64-bit REAL variables

This block is used by the block PIDConR for calculations that are twice as accurate (64 bit number format).

Therefore this block will not be described in detail.

17.3 DiToInt64 - Converting from DINT to Int64

17.3.1 Description of DiToInt64

Object name (type + number) and family

Type + number: FC 306

Family: System

Area of application for DiToInt64

The block is used for the following applications:

- Converting from DINT to Int64

This block is used by the block PIDConR for calculations that are twice as accurate (64 bit number format).

Therefore this block will not be described in detail.

17.4 Int64ToDi - Converting from Int64 to DINT

17.4.1 Description of Int64ToDi

Object name (type + number) and family

Type + number: FC 308

Family: System

Area of application for Int64ToDi

The block is used for the following applications:

- Converting from Int64 to DINT

This block is used by the block PIDConR for calculations that are twice as accurate (64 bit number format).

Therefore this block will not be described in detail.

17.5 NegInt64 - Negation of an Int64 variable

17.5.1 Description of NegInt64

Object name (type + number) and family

Type + number: FC 311

Family: System

Area of application for NegInt64

The block is used for the following applications:

- Negation of an Int64 variable

This block is used by the block PIDConR for calculations that are twice as accurate (64 bit number format).

Therefore this block will not be described in detail.

17.6 NegR64 - Negation of a Real64 variable

17.6.1 Description of NegR64

Object name (type + number) and family

Type + number: FC 312

Family: System

Area of application for NegR64

The block is used for the following applications:

- Negation of a Real64 variable

This block is used by the block PIDConR for calculations that are twice as accurate (64 bit number format).

Therefore this block will not be described in detail.

17.7 PIDCoefR - Calculation of coefficients

17.7.1 Description of PIDCoefR

Object name (type + number) and family

Type + number: FC 315

Family: System

Area of application for PIDCoefR

The block is used for the following applications:

- Calculation of coefficients of discrete-time difference equations for PIDConR from the continuous-time input parameters (e.g. TI, TD)

This block is used by the block PIDConR for calculations that are twice as accurate (64 bit number format).

Therefore this block will not be described in detail.

17.8 R64ToReal - Converting Real64 to REAL

17.8.1 Description of R64ToReal

Object name (type + number) and family

Type + number: FC 316

Family: System

Area of application for R64ToReal

The block is used for the following applications:

- Converting Real64 to REAL (32 bit)

This block is used by the block PIDConR for calculations that are twice as accurate (64 bit number format).

Therefore this block will not be described in detail.

17.9 RealToR64 - Converting REAL to Real64

17.9.1 Description of RealToR64

Object name (type + number) and family

Type + number: FC 317

Family: System

Area of application for RealToR64

The block is used for the following applications:

- Converting REAL (32 bit) to Real64

This block is used by the block PIDConR for calculations that are twice as accurate (64 bit number format).

Therefore this block will not be described in detail.

17.10 SelST16 - Output of the best or worst signal status

17.10.1 Description of SelST16

Object name (type + number) and family

Type and number: FC 318

Family: System

Area of application for SelST16

The block is used for the following applications:

- Output the best or worst signal status

This block is used by technological blocks which edit and output a signal status.

Therefore this block will not be described in detail.

17.11 ShLeInt64 - Left shift of an Int64 variable

17.11.1 Description of ShLeInt64

Object name (type + number) and family

Type + number: FC 319

Family: System

Area of application for ShLeInt64

The block is used for the following applications:

- Left shift of a (positive) Int64 variable

This block is used by the block PIDConR for calculations that are twice as accurate (64 bit number format).

Therefore this block will not be described in detail.

17.12 ShRiInt64 - Right shift of an Int64 variable

17.12.1 Description of ShRiInt64

Object name (type + number) and family

Type + number: FC 320

Family: System

Area of application for ShRiInt64

The block is used for the following applications:

- Right shift of a (positive) Int64 variable

This block is used by the block PIDConR for calculations that are twice as accurate (64 bit number format).

Therefore this block will not be described in detail.

17.13 PIDKernR - calculation of the manipulated variable

17.13.1 Description of PIDKernR

Object name (type + number) and family

Type and number: FB 1877

Family: System

Area of application for PIDKernR

The block is used for the following applications:

- Calculation of the manipulated variable in PIDConR

This block is used by the block PIDConR for calculations that are twice as accurate (64 bit number format).

Therefore this block will not be described in detail.

PCS 7 Advanced Process Control Templates 18

18.1 Process tag types (insertible templates)

18.1.1 Introduction to process tag types

Introduction

You can find control engineering information on the standard process tag types (insertable templates) of the Advanced Process Library in a separate section in this help:

- PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 1430)
- PIDConR with safety logic and control loop monitoring (PIDConR_ConPerMon) (Page 1431)
- Step controller with direct access to the actuator and without position feedback (StepControlDirect) (Page 1437)
- Step controller with assigned actuator block and position feedback (StepControlActor) (Page 1437)
- Split-range control (Page 1438)
- Ratio control (Page 1440)
- Ratio control with PIDConR (RatioR) (Page 1441)
- Cascade control (Page 1442)
- Cascade control with PIDConR (CascadeR) (Page 1444)

In addition to the process tag types mentioned above, you will also find the following in the Templates folder of the library:

- PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 1433)
- PID - control with operating-point-oriented parameter control (GainScheduling) (Page 1431)
- Override control (Page 1445)
- Override control with PIDConR (OverrideR) (Page 1447)
- PID controller with Smith predictor (SmithPredictorControl) (Page 1436)
- Model-based predictive control (ModPreCon) (Page 1447)

Descriptions for the following process tag types are also available:

- Monitoring eight digital process tags (Digital8Monitoring) (Page 1448)
- Monitoring of a digital process tag (DigitalMonitoring) (Page 1448)
- Monitoring an analog process tag (AnalogMonitoring) (Page 1449)
- Dosing (DoseLean) (Page 1449)
- Two-speed motor (Motor2Speed) (Page 1450)
- Reversing motor (MotorReversible) (Page 1450)
- Reversing motor with controllable speed (MotorSpeedControlled) (Page 1451)
- Valve (ValveLean) (Page 1453)
- Two-way valve (Valve2Way) (Page 1453)
- Motor valve (ValveMotor) (Page 1454)

This section contains prototype process tag types for several of the control loop structures shown in the example project (Page 1455). These prototypes are simply examples of how such process tag types might appear. In most process plants, it is assumed that high-level control loop structures must be configured individually some in combination with lower-level control loops. This means that mass production of high-level control loop structures as instances of process tag type represents the exception.

Using process tag types

When using process tag types in a PCS 7 multiproject, the following procedure is recommended:

- Copy all the required function blocks from the Advanced Process Library to the block folder of the master data library of the multiproject (<Projectname>_Lib).
- Then in the plant view, copy the required process tag types from the Advanced Process Library to the "Process_tag_types" folder of the master data library of the multiproject.
- Make any customizations of the process tag types.

Generating the process tags

Process tags are the instances of the process tag types.

There are two ways of generating process tags:

1. Copy the process tag type from the master data library and insert it in the target folder in the plant hierarchy. You can then set the parameters for and interconnect the CFC chart in the CFC Editor.
2. Using the Import-Export assistant, you can then generate the required process tags of the process tag type that you then assign parameters to and interconnect based on an import file.

Notes on the "Measured value failure" application scenario for controller block process tag types

If there is no valid measured value for PV, the controller must not remain in automatic mode because a closed control loop no longer exists. The controller is unable to calculate a useful manipulated variable if there is no feedback of the actual process state. The controller must then be changed to manual or tracking mode. OS operators must be made aware of this situation by a corresponding message. Various reactions to the loss of measured values are conceivable depending on the application background:

1. Tracking a fixed, configured safety value, for example Valve closed or Heater off.
2. Holding the last valid manipulated variable MV constant to retain a steady process state as far as possible (if the process was already in such a state).
3. Change to manual mode, so that the OS operator can take over responsibility for process control.

A combination of reactions 2 and 3 is implemented in the template. The controller changes to tracking with the last valid manipulated variable as start value. The switchover is made via the input MV_TrkOn of the PID block. Since manual mode already has priority over tracking, the OS operator can subsequently use the command.

Caution: It is not advisable to freeze the last valid manipulated value if the signals are subject to heavy noise, or in control loops with frequent setpoint changes. In such situations, it is advisable to change to variant 1.

If a controller intervenes locally in the process via an actuator (for example, a valve with electropneumatic positioner Sipart PS) specific measures similar to those for a cascade controller must be taken:

- If there is local intervention, the controller tracks the current actuator position. In this case, the feedback signal is applied to input MV_Trk and an OR element is set before input MV_TrkOn.

Notes on analog position feedback for controller block process tag types

If there is no position feedback, delete the associated analog input driver MV_Rbk in the CFC chart. This will also make the corresponding display elements in the standard view of the faceplate invisible.

18.1.2 PID controller (PIDConLean)

PID controller

This process tag type forms the basis for generating PID control instances for continuous processes. In addition to the actual PID block, it contains functionality that should be implemented consistently in each control loop. This functionality is, however, not implemented within the PID block itself so that users can adapt the functions to a specific project:

- An analog input driver for the process value PV, the position feedback Rbk (if available), and an analog output driver for the manipulated variable MV.
- Certain logic blocks that change the control loop to a safe mode if the measurement of the process value fails. This state is signaled by the signal status of PV.

Note

Read the information for process tag types with controller blocks in Introduction to process tag types (Page 1427).

18.1.3 PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon)

PID controller with safety logic and control loop monitoring

This process tag type forms the basis for generating PID control instances for continuous processes. In addition to the actual PID block, it contains functionality that should be implemented consistently in each control loop. This functionality is, however, not implemented within the PID block itself so that users can adapt the functions to a specific project:

- An analog input driver for the process value PV, the position feedback Rbk (if available), and an analog output driver for the manipulated variable MV.
- A simple process simulation of the first order, controlled by the manipulated variable MV which provides simulation values to analog input PV. This functionality allows at least elementary function tests control loop before a real process is connected.
- Certain logic blocks that change the control loop to a safe mode if the measurement of the process value fails. This state is signaled by the signal status of PV.
- An additional function block for control loop monitoring that should be installed in all PID control loop.

Note

Read also the information on controller block process tag types in Introduction to process tag types (Page 1427).
You can find information on the use of control loop monitoring in ConPerMon functions (Page 273).

18.1.4 PIDConR with safety logic and control loop monitoring (PIDConR_ConPerMon)

PIDConR with safety logic and control loop monitoring

This process tag type corresponds to the process tag type PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 1430) and control loop monitoring when PIDConR is used rather than PIDConL.

18.1.5 PID - control with operating-point-oriented parameter control (GainScheduling)

PID - control with operating-point-oriented parameter control

Many technical processes have a non-linear response due to non-linear physical, chemical or thermodynamic effects. When such a process needs to be kept in the close vicinity of a fixed operating point, the transfer response can be linearized around this operating point. A linear PID controller can be designed for this linearized transfer function. If, however, the process has a strongly non-linear response and/or operates at different operating points, no constantly good control response can be expected throughout the entire operating range. Due to the non-linearity, various gain factors or process time constants are in effect at different operating points. In keeping with this, different controller parameters will be considered to be optimum.

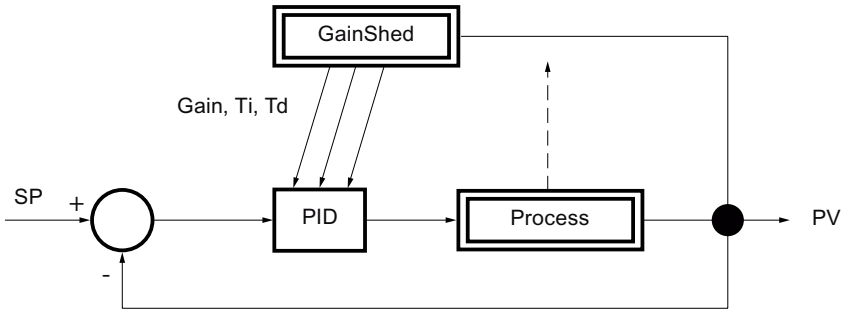


Figure 18-1 Operating-point-oriented parameter adaptation

One possible (the simplest) solution to this problem is known as gain scheduling or parameter scheduling. Using a tool such as the PCS 7 PID Tuner, various experiments are performed at different operating points, in each case with low signal amplitudes. This results in different PID parameter sets for each operating point. Up to three such parameter sets can be stored in the GainSched Function block. The suitable parameter set is selected depending on a continuously measurable variable that describes the state of the process, typically the control variable PV itself. Between the operating points for which there are exact parameter values, the values are calculated by linear interpolation of the neighboring interpolation points so that soft and bumpless transitions are possible between the operating points. The term "parameter scheduling" makes it clear that the "timetable" for adjusting the parameters is specified in advance. In contrast, an adaptive controller adapts itself automatically to the differing process response during operation.

The function block GainSched is produced from the CFC chart "fbGainSched" by compiling it as a block type. This CFC chart is supplied with the library so that the user has the option of expanding the existing basic functionality as necessary, for example to more than three operating points.

Note

The combination of several locally optimized controllers by gain scheduling to form a non-linear controller does not necessarily represent an optimum non-linear controller for the non-linear process when considered from a mathematical point of view. This becomes clear even with benign non-linearities (that are continuous and can be differentiated) when setpoint step changes are made between different operating points. With non-linearities that are discontinuous or cannot be differentiated or with non-monotonic non-linearities, great caution is needed.

Examples of applications

- Control (especially temperature control) of batch processes, for example, batch reactors and batch columns
- pH value control
- Temperature control with phase transitions (for example, fluid/vapor form)
- Control of semi-batch plants (continuous plants with operating point changes, for example, polymerization reactors)
- Control in power plants with load changes

18.1.6 PID controller with dynamic feedforward control (FfwdDisturbCompensat)

PID controller with dynamic feedforward control

Feedforward can be used when a known, strong disturbance affects the process and its cause can be measured. In these cases, the following general strategy applies: "Control as much as possible (if known in advance and described by a model), control as much as necessary (the rest including the model error and immeasurable disturbances)".

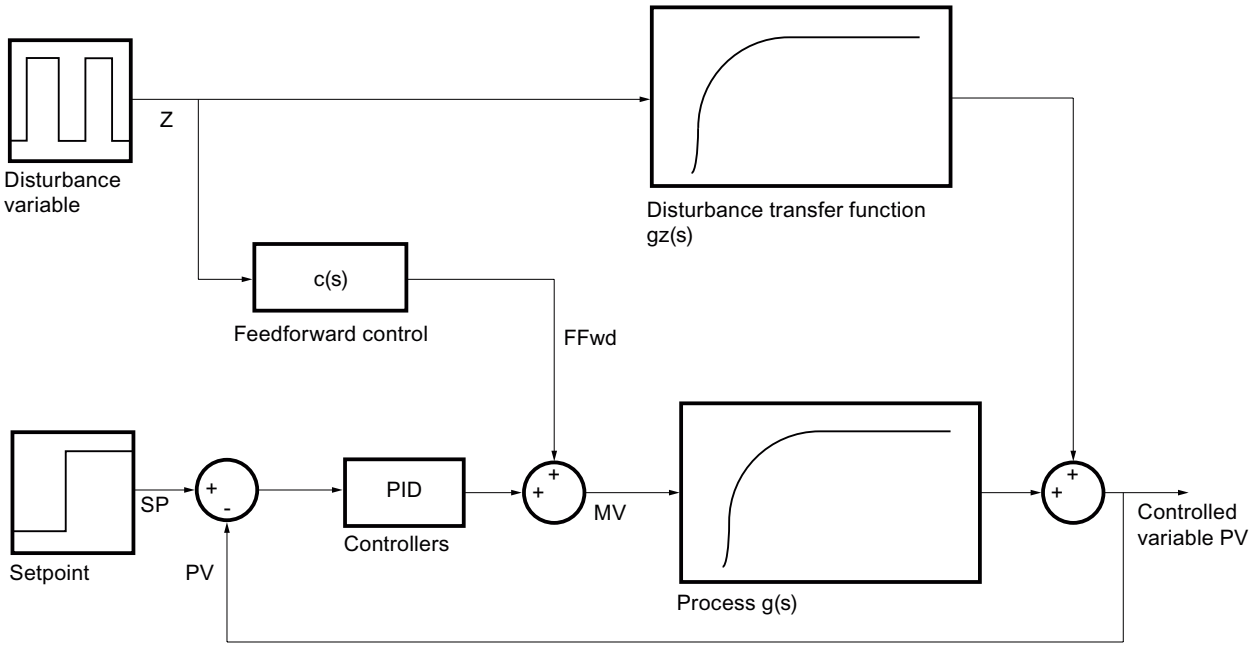


Figure 18-2 Dynamic feedforward control

The effect of a measurable disturbance can be estimated in the form of a transfer function $g_z(s) = y(s) / z(s)$ when the controller is running in manual mode so that no changes whatsoever to the controlled variable y = PV are caused by the manipulated variable and all changes can be attributed to the disturbance z(s).

The transfer function of an ideal feedforward control c(s) can be derived from the requirement that the effect of z on y should be zero for any disturbance signal z(s):

$g_z(s) \cdot z + c(s) \cdot g(s) \cdot z = (g_z(s) \cdot g(s)) \cdot z = 0$

To meet this equation, the compensation block must approximate the equation

$$c(s) = -\frac{g_z(s)}{g(s)}$$

as well as possible. For this to happen, the disturbance transfer function $g_z(s) = y(s) / z(s)$ must be known and the transfer function of the main controlled system $g(s) = y(s) / u(s)$, u = MV must be inverted. If both transfer functions can be modeled as first order with dead time

$$g(s) = \frac{k_s}{1 + t_1 s} \cdot e^{-s\theta}$$

and

$$g(s) = \frac{k_{sz}}{1 + t_{1z} s} \cdot e^{-s\theta_z}$$

and $\theta < \theta_z$ applies, the resulting compensation element must represent the transfer function

$$c(s) = -\frac{k_{sz}}{k_s} \cdot \frac{1 + t_1 s}{1 + t_{1z} s} \cdot e^{-s(\theta_z - \theta)}$$

.

In general, the following dynamic transfer function is required for additive feedforward control:

$$FFwd(s) = -k_c \cdot \frac{t_{cd} s + 1}{t_{cl} + 1} \cdot e^{-\theta_c s} \cdot z(s)$$

where:

$MV = MV_{PID} + FFwd$

In the example above, this function includes the following parameters:

$$k_c = \frac{k_{sz}}{k_s}, \quad t_{cd} = t_1, \quad t_{cl} = t_{1z}, \quad \theta_c = \theta_z - \theta'$$

This transfer function can be created outside the controller with a combination of elementary CFC blocks as the series connection of a DT1 (Diff), of a PT1 (TimeLag) and a DeadTime block.

The input parameters k_c, t_{cd}, t_{c1} need to be configured by the user. For a static feedforward control, both time constants are set to zero.

Examples of applications

- Controlling the outlet temperature of a heat exchanger via steam pressure or heating/cooling medium flow. Flow and inlet temperature of the medium are measurable disturbance variables.
- Fill level control in a drum steam generator using the inlet volume. The outflow is the measurable disturbance variable that is determined by the variable steam consumption in the plant.
- Temperature control in a distillation column using the reflux ration or or heating steam volume. The measurable disturbance variable is the mixture inlet.
- Temperature and concentration control in an agitated tank reactor using cooling medium flow and discharge volume. The temperature and possibly also the concentration of the inflow are measurable disturbance variables.

18.1.7 PID controller with Smith predictor (SmithPredictorControl)

PID control with Smith predictor

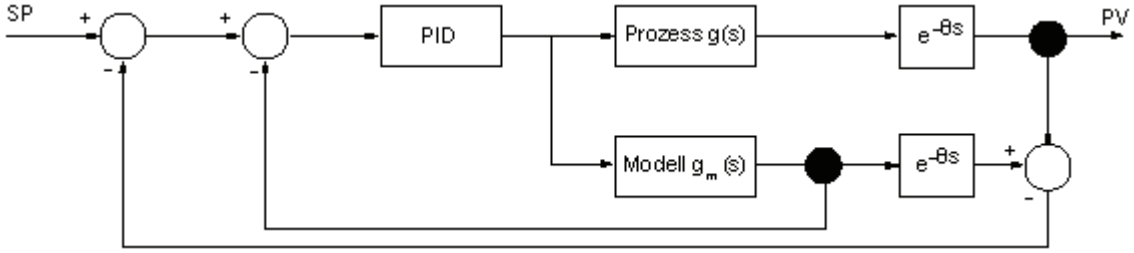


Figure 18-3 PID control with Smith predictor

In processes with large dead times (relative to the dominating time lag constant), a standard PI controller must be set very slowly and compromises must therefore be accepted in the control quality. The control quality can be significantly improved with a Smith predictor that can be derived from the IMC principle (**I**nternal **M**odel **C**ontrol) of model-based control. To achieve this, the transfer function $g_s(s) = g(s) \cdot e^{-s\theta}$ of the controlled system is split up into a part without dead time g(s) and a purely dead time slice $e^{-s\theta}$ with dead time θ. Only the controlled variable y affected by dead time can be measured in the real process. However, a virtual estimate of the controlled variable free of dead time can be taken from the process model (that will become part of the controller) and fed to the controller. This means that the controller itself can be designed for the process without a dead time slice and can therefore be set much more tightly. To compensate unknown disturbances, an estimate of the controlled variable affected by dead time is made in the model and compared with the genuine measured controlled variable. This difference is also fed back to the controller.

In terms of practical application, it must be pointed out that the performance of the Smith predictor depends largely on the model fit, in other words, the dead time must be known. The dead time must be constant or its value must be permanently adapted.

Note

To control processes with large dead times, a model predictive controller (see Description of ModPreCon (Page 391)) is also suitable in a single variable situation. It provides greater flexibility in the system modeling and is more convenient thanks to the integrated design procedures, it does however require more CPU resources.

18.1.8 Step controller with direct access to the actuator and without position feedback (StepControlDirect)

Step controller with direct access to the actuator and without position feedback

The output of the PIDStepL block is directly connected to the process via two digital output drivers. This is the simplest form of a step control. The operator can control the actuator (push-button open/close) in the faceplate of the controller. This push-button operation is preferable when operating simple actuators without position feedback since it allows bumpless changeover to auto mode. This changeover is not possible if actuators without position feedback are operated manually from any location outside the control block. For this reason, the template is designed with a PIDStepL block without position feedback.

18.1.9 Step controller with assigned actuator block and position feedback (StepControlActor)

Step controller with assigned actuator block and position feedback

The output of the PIDStepL block is directly interconnected with the process by way of an actuator block (for example, a motor or a valve). This additional effort is justified when special automation functions of the actuator block, for example motor current monitoring, tripping and operation in local mode are required. The option of such combinations allows users to do without special functions for operation in local mode or actuator monitoring functions within the controller block.

If the actuator block is in automatic mode, the actuator is capable of accepting and executing controller commands.

If the actuator block is in manual or local mode, tracking must also be activated for the controller for the actuator to be able to accept and execute controller commands.

In this context, the resulting structure must be interpreted as a "cascade circuit" consisting of the (primary) controller and actuator block (secondary controller) and measures similar to those for cascade control must be taken. This means that, the AutAct-Bit of the actuator block must be set to 'False' and it must be interconnected inverted with the `TrkOn` input of the primary controller in all operating modes in which the actuator block does not accept automatic commands.

To ensure the bumpless switchover from local or manual mode back to cascade mode, the current actuator position must be fed back to the tracking input of the primary controller. This is, however, only possible when using actuators with position feedback. For this reason, the template is designed for operation with a PIDStepL block with position feedback.

18.1.10 Split-range control

Split-range control

A PID closed-loop controller can use a Split-Range block downstream of the controller output to distribute its manipulated variable to several actuators that influence the same control variable based on different physical principles and in different directions. A typical example of such an application is a temperature control, with heating via a live steam valve and cooling via a cooling water valve. The controller can request heating or cooling energy, depending on the sign of the control error (or more precisely of the manipulated variable). In other words, it can work with a bidirectional MV output (for example: -100% < MV < 100%), although each actuator only supports unipolar operation (for example: 0% < valve position < 100%).

The Split-Range function block contains two separate (static) characteristics for both actuators. Any significant difference between the two actuators in terms of performance (can be interpreted as different process gains for heating and cooling), can be compensated by setting different gradients for the characteristics, so that as far as possible, the controller is presented with a linear process response (independent of the sign).

Example: The cooling effect is not as strong as the heating effect.

If cooling is half as effective as heating, the split-range characteristic for cooling should have twice the gradient.

Note

The effective maximum values of the manipulated variables are obtained from the manipulated variable limits of the controller multiplied by the gradients of the split-range characteristics.

The cooling valve cannot go beyond fully open, in other words, the effective limitation for opening the valve is 100%. If a split range characteristic with gradient 2 is used for cooling, the low limit for the manipulated variable must be set to MV_LoLim = -50% on the controller.

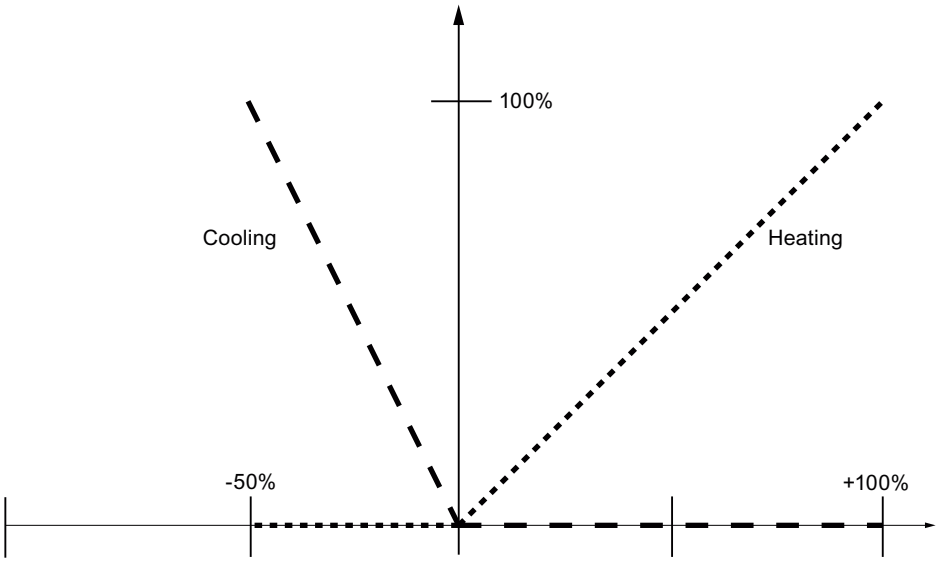


Figure 18-4 Split-range function of the example

The actual split-range function is supplied as separate SplitRange function block that is described in detail in the online help of the block.

Examples of applications:

- Control of the temperature of chemical reactors by means of steam heating valves and water cooling valves.
- Temperature control of glass furnaces or sprue channels by means of gas burners and cooling fans
- Temperature control of extruders by means of electrical heating and cooling fans
- Pressure control in a gas phase reactor by means of inlet and outlet valves
- Pressure control at a steam collecting track that supplies steam to several units by means of inlet valves from several steam generators, or the heating power of several steam generators.

18.1.11 Ratio control

Ratio control

Specific mixtures of several liquids or gases can be produced by means of a ratio control system consisting of several flow controllers and a Ratio block. The flow setpoint of an additive is obtained from one of these values:

1. From the current PV value of the main flow.
   This is the preferred alternative if the primary flow controller operates with steady-state deviation.
   or
2. From the setpoint SP of the main flow.
   This alternative provides a smooth, noise-free setpoint signal to the second controller, and allows for a more precise control of specified ratios in transition states where both flow control loops have approximately the same dynamic response.

It is generally advisable to use the second "setpoint-oriented" alternative for main flow controllers with I action.

Interconnect the selected reference value (actual value or setpoint of the main flow) to the In input parameter of the Ratio block.

The ratio control can be expanded by adding further components, in other words, setpoints 3 to n can be derived from SP1 (or PV1) with the help of additional Ratio blocks.

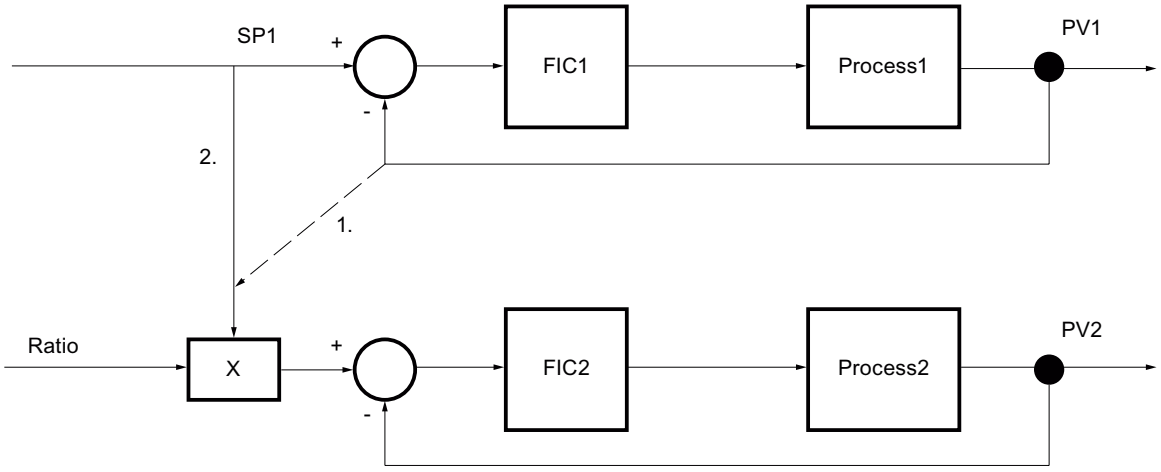


Figure 18-5 Ratio control, process value-oriented (1.) and setpoint-oriented (2.)

The main task of the Ratio block is to define the external setpoint of the secondary (added) component in accordance with

$$Out = In \cdot Ratio + Offset$$.

The actual ratio of the two flow actual values is also calculated in accordance with

$$RatioPV = \frac{SecComPV - Offset}{InPV}$$

and displayed on the control panel for checking purposes. In addition, interconnect the actual value of the main flow to the input parameter `InPV` of the Ratio block and the actual value of the flow of the secondary component to the input parameter `SecComPV`.

In both cases 1 and 2, the current value of the ratio is not controlled directly in the closed loop (no feedback-control) but rather uses feedforward-control.

18.1.12 Ratio control with PIDConR (RatioR)

Ratio control with PIDConR

This process tag type corresponds to the process tag type Ratio control (Page 1440) when PIDConR is used rather than PIDConL.

You need to set feature bit 21 to 1 with a flow controller for the secondary component.

18.1.13 Cascade control

General information on cascade control

A cascade control involves two or more PID controllers connected in series. The manipulated variable of the primary controller is interconnected to the external setpoint of the secondary controller so that both control loops are nested. The advantage of cascade control is that disturbances affecting the inner loop can be compensated much more quickly in the secondary loop than in the slower primary loop. In some situations, non-linear effects of the actuator can be compensated in the secondary loop so that a linear process response can be created for the primary loop. Cascade control is possible only when there are further measurable variables in the process in addition to the main control variable and when the internal control loop is significantly faster than the external loop.

Aspects to be clarified for the cascade control

With any cascade control, the following aspects must be carefully considered and clarified:

- The actuation range of the primary controller must match the setpoint range of the secondary controller to ensure proper operation of the anti-Windup functions of the primary controller.
- If the secondary controller is not operated in "cascade" mode (automatic mode with external setpoint) but in any other mode (for example manual or automatic mode with local setpoint) and therefore does not respond to commands of the primary controller, the primary controller must be put into "tracking" mode to prevent integration of the I action in the primary controller. The manipulated value of the primary controller tracks the process value of the secondary controller to allow a bumpless return to cascade mode. The difference between tracking the setpoint and tracking the process value becomes apparent when the secondary controller put into manual mode. If the process value is tracked, the response is similar to the "Track setpoint to process value in manual mode" of a simple controller.
- If the secondary controller reaches a (high/low) manipulated variable limit, the integrator of the primary controller should be blocked to prevent it going any further in this direction (up / down). The secondary controller cannot go any further in this direction anyway. This prevents any windup of the primary controller when the real actuator has already reached its physical limits while the primary controller has not yet reached its manipulated variable limits.

  Caution:

  – Swap the two bits of this interconnection if the secondary controller has a negative gain.
  – If the secondary controller then reaches a high or low limit, the integrator of the primary controller must be prevented from further integration in this direction.

Procedure

When you set up and commission the controller, you work from the "inside to the outside", in other words, you start by making the settings for the secondary controller and putting it into auto mode. You then set the primary controller parameters and change the secondary controller to cascade mode. When you set the parameters for the primary controller, remember that from its perspective the entire, inner closed control loop is the "controlled system". The parameters you set for the primary controller depend to a greater or lesser extent on the settings you made for the secondary controller. This becomes less important the greater the difference in dynamic response between the primary and secondary control loop.

Priorities

The tracking by the primary controller that is initiated by the secondary controller has lower priority than the manual mode on the primary controller. In turn, manual mode has lower priority than forced mode on the master controller as can, for example, be requested by external logic during an emergency shutdown of the plant (controller input `MV_Forced`, unlimited, activated by `MV_ForOn` with highest priority). For this reason, the PIDConL block for cascade control has an additional tracking input `MV_Trk` that is activated by `MV_TrkOn` and is subject to the normal manipulated variable limiting and has lower priority than manual mode.

Additional application examples

- Temperature control of a distillation column (primary controller) based on the reflux ratio (secondary controller at the top of the column), and heating steam flow rate (secondary controller at the column sump),
- Temperature control of a furnace using a secondary controller to control the fuel flow rate,
- Fill level control for a container using a secondary controller for the inlet and/or outlet flow.
- Position control (in drive engineering) with a secondary controller for the speed and torque.

Using secondary flow controllers

Secondary controllers are usually used for flow control to prevent detrimental effects on the primary controller resulting from changes in flow rate. The non-linearities frequently often found in a flow control element (for example a valve) are "hidden" in the secondary loop because the secondary closed loop has a linear response and non-linearities have no effect on the primary controller or its tuning.

18.1.14 Cascade control with PIDConR (CascadeR)

Cascade control with PIDConR

This process tag type corresponds to a large extent to the process tag type Cascade control (Page 1442) when PIDConR is used rather than PIDConL.

Special note: The current `PV_Out` output of the PID_Slave secondary controller is used as an external Reset `ExtReset` on the primary controller. Unlike cascade control, the PID_Master primary controller does not therefore have to be changed to tracking when the cascade is disconnected with all other PID controllers. There is also no need for the direction-dependent blocking of the integrator for the primary controller.

Since the primary controller is reliant on the external Reset, any control deviations in the secondary closed loop interfere with the primary controller; secondary controllers without I action are not therefore recommended for PIDConR.

You need to set the Feature bit Switching operator controls for external setpoint to visible (Page 189) to 1 with a secondary controller of the cascade.

18.1.15 Source chart for GainSched function block (gain scheduling)

Source chart for GainSched function block

In contrast to all other function blocks, the `GainSched` block is implemented as a CFC chart and is generated with the "Compile chart as block type" function. Several application options are available in this case:

- You can use the precompiled function block `GainSched` from the library if the standard functionality is adequate for your needs.
- If you require special additional functions for gain scheduling in your application (for example more than three operating points, additional logic functions for selecting the parameters), you will need to modify the CFC source chart and compile it as a block type with a different FB number.

Internally, the `GainSched` block consists essentially of three instances of the Polygon block, one for each of the three controller parameters, Gain, TI and TD.

The polygon block itself would extrapolate linearly outside of the boundary points and thereby exceed the value range of its interpolation points. However, as this presents too high a risk for controller parameters, the controller parameters which are output are effectively limited to the table values at the boundary operating points by the automatic introduction of additional interpolation points with a horizontal end tangent outside of the specified range.

18.1.16 Override control

Override control

In an override control, two or more controllers share a common actuator. Depending on the current process state, a decision is made as to which controller actually has access to the actuator, in other words, the various controllers can override each other.

A typical use case is a gas pipeline with pressure and flow control using a single valve. The main aim of the control is to achieve a certain flow rate, however due to safety considerations, the pressure must be kept within certain limits. The pressure controller is therefore known as the "limiting controller" or "secondary controller".

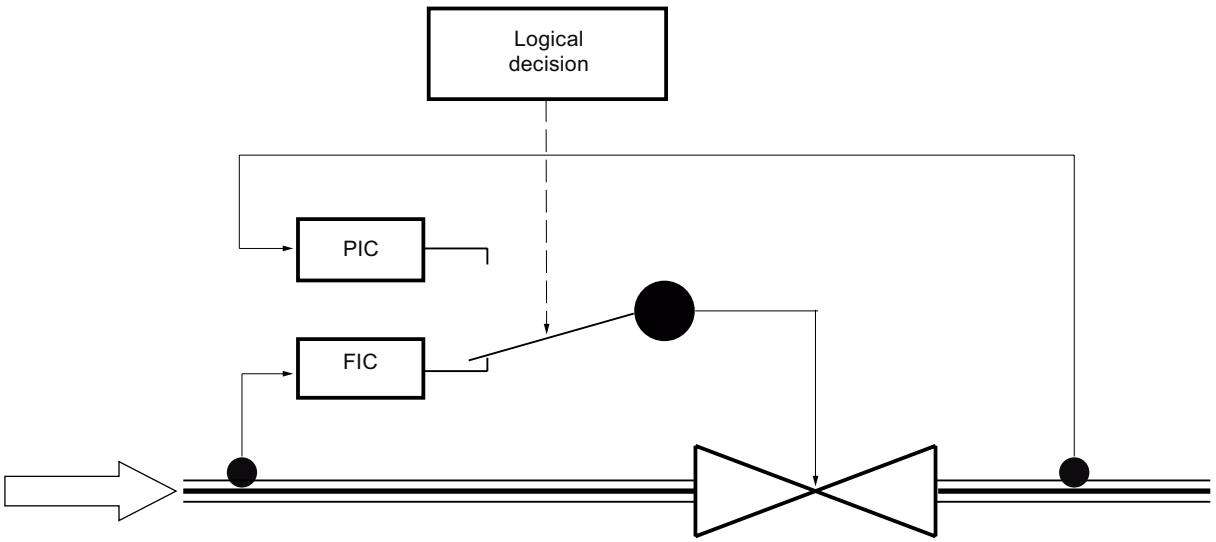


Figure 18-6 Override control with main controller FIC and limiting controller PIC

Criteria for the type of override control

The logical decision as to which controller should be active can be made based on two different criteria resulting in two different types of override controls:

1. The decision is based on a measurable process output variable, for example one of the two controlled variables. In the example above, the warning limits of the pressure controller can be used to decide whether the pressure controller should be active. The passive controller is in tracking mode to avoid Windup problems and to ensure bumpless transfer. The setpoint of the secondary controller must be somewhat lower than the switchover threshold so that the transfer can be reversed again. This type of override control is easy to understand and to implement. Its advantage is that the high and low limit of the secondary controlled variable (for example pressure) can be monitored; its disadvantage is that a limit cycle oscillation results as soon as the limiting controller needs to intervene. The secondary controller will always attempt to return its controlled variable to the safe range and to return command to the main controller (for example flow rate) so that the active and passive controllers swap over continuously. This variant is therefore only recommended when the secondary controller is seldom required and functions mainly as a safety or backup system.

2. The decision is based on a comparison of the manipulated variables of both controllers, for example the controller that demands the higher (or lower) controlled variable takes control of the actuator. In the example above, the controller that wants to open the valve further takes over control. The setpoint of the secondary controller defines the switching threshold. Both controllers run the entire time in automatic mode. To avoid Windup problems, the manipulated variable limits must be tracked in a crossover structure: When the higher (lower) manipulated variable wins, the low (high) limits of all controllers of the currently highest (lowest) manipulated variable must be corrected slightly up or down by, for example, 2% of the manipulated variable range. This means that this scheme can also be used in applications with more than two controlled variables. There is no Windup problem at the high limit because the highest manipulated variable takes over control anyway. This approach avoids the limit cycle oscillation of alternative 1 but is, in principle asymmetrical, in other words either a high or a low limit of the secondary controlled variable can be monitored but not both.
   This type of override control is described in most control textbooks, particularly in the USA. It can, however, only be used with PID algorithms that allow online manipulation of the manipulated variable limits (in PCS 7 as of V6.0).

Additional application examples

- Steam generator: The primary controlled variable is the steam pressure but the water level in the steam tank must be monitored so that the heating coils remain completely covered by water and the tank does not overflow. The only manipulated variable is the outlet valve.
- Compressor: The primary controlled variable is the throughput but the pressure must be monitored to make sure it does not exceed a safety limit. The only manipulated variable is the motor speed.
- Steam distribution system: Every plant involving industrial processes has a network of pipes to distribute steam at various pressures throughout the plant. The high pressure of the steam is reduced to lower levels via a valve. The primary controlled variable is the pressure at the lower-level stage, however the pressure in the high pressure piping must also be monitored to make sure that it does not exceed a safety limit.

18.1.17 Override control with PIDConR (OverrideR)

Override control with PIDConR

This process tag type corresponds to a large extent to the process tag type Override control (Page 1445) when PIDConR is used rather than PIDConL.

Special note: The current manipulated variable output at the final controlling element (e.g. the maximum `MaxMV.Out` manipulated variable proposed by the main controller and limiting controller) is used as the external Reset`ExtReset` for both controllers. Unlike override control, the manipulated variable limits (e.g. `MV_LoLim`) of both controllers do not have to be tracked within a defined space (e.g. `MaxMV_Minus2.Out`) by the manipulated value output at the final controlling element with all other PID controllers.

18.1.18 Model-based predictive control (ModPreCon)

Model-based predictive control

The process tag type shows how users can expand the ModPreCon block by adding extra functions:

- External alarming with the MonAnL block
- Control quality monitoring with the ConPerMon block
- Safety logic for measure value loss.

You can find details about the ModPreCon block in the Description of ModPreCon (Page 391).

Note on using the ConPerMon block in a system with multi-variable

The mathematical concept of the ConPerMon block is designed for single variable applications. If any increase of variance is detected in the channel of a multi-variable control, the ConPerMon algorithm is unable to determine whether this problem is caused by its own internal control channel or by interaction of neighboring channels. It may be helpful, however, to include a ConPerMon block for each control channel of a multi-variable system to monitor whether the control quality during operation remains within the range determined during commissioning. To achieve this, several logic operations must be performed before the `ManSuprCPI` input bit of each ConPerMon block:

- If one or several of the other channels of the multi-variable system are are not in a steady state ("root caused in this channel") due to local causes (for example a setpoint step change) that is indicated by output bit `CPI_SuRoot = 1`, the increase of variance in its own channel cannot be avoided and should not trigger a CPI warning in its own channel.
- If one or several other channels of the multi-variable system exhibit strong variations as indicated by output bit `CPI_WrnAct = 1`, the increase in variance in its own channel cannot be avoided and should not trigger a CPI warning in its own channel. This may be helpful to localize the actual cause of the problem. The channel in which excessive variations are first detected outputs the first alarm; the other channels which may only be affected by errors resulting from the first error do not generate their own alarms.

Examples of applications

- Quality control in distillation columns, for example control of the column top and sump temperature based on the reflux ratio and heating steam volume
- Temperature control of several adjacent zones of furnaces with several burners, for example, tunnel furnaces, glass smelting plants, glass sprue channels etc.
- Quality control in chemical reactors by adjusting of reaction conditions such as pressure, temperature, feed/drain etc.
- Vaporizers, for example, drum steam generators
- Mills, for example, cement mills, sieve mills: quality control (grain size) in combination with flow rate maximized, manipulated variables: sieve speed and mill feed

Note

The statistical evaluation of the service factor (time slice in automatic mode) and the time slices in the manipulated variable limits can be performed in a WinCC trend control, as in the customary case of a single variable. The binary variables from ModPreCon required for this are archived automatically. However, an appropriate trend control must be manually configured on the operator station, since the view archive in the ConPerMon block is not designed for such quantities.

18.1.19 Monitoring of a digital process tag (DigitalMonitoring)

Monitoring a digital process tag

This process tag type serves as a basis for monitoring a digital process tag using the MonDiL block.

The digital measurement signal is read from the I/O through the PCS7DiIn block.

The process tag type contains the required interconnections between Pcs7DiIn and MonDiL.

18.1.20 Monitoring eight digital process tags (Digital8Monitoring)

Monitoring eight digital process tags

The process tag type serves as a basis for monitoring up to eight digital process tags using the MonDi08 block.

The digital measurement signals are read from the I/O through the PCS7DiIn block.

The process tag type contains the required interconnections between the eight Pcs7DiIn and MonDi08 blocks.

18.1.21 Monitoring an analog process tag (AnalogMonitoring)

Monitoring an analog process tag

This process tag type serves as a basis for monitoring an analog process tag using the MonAnL block.

The analog value is read from the I/O through the PCS7AnIn block.

The process tag type contains the required interconnections between Pcs7AnIn and MonAnL.

If you wish to monitor and forward the slope of the process value, you usually need to smooth the process value. You can insert a filter block such as Smooth between Pcs7AnIn and MonAnL for this purpose. The gradient of the process value can also be smoothed using the TimeLag parameter at MonAnL.

Procedure for smoothing the process value and its gradient:

1. Check the process value PV in the trend recorder and smooth it using a filter block such as Smooth only as much as is needed.

2. Check the gradient PV_Grad of the process value in the trend recorder and smooth the trend at the TimeLag parameter of MonAnL.

The sum of the time constants of Timelag from MonAnL and TimeConstant from Smooth approximately result in the length of a varying time window in which the gradient is averaged.

18.1.22 Dosing (DoseLean)

Dosing

This process tag type serves as a basis for dosing either as "Single component dosing via flow measurement" or as "Fill/extraction weighing via dosing scale" using the DoseL block.

The analog measurement signal is read from the I/O through the PCS7AnIn block.

The interlock signals of DoseL are connected to the IntLk02 interlock blocks. These interlock blocks in turn are connected to other blocks, for example, to digital process tags via Pcs7DiIn.

The digital output signals are sent to the I/O through PCS7DiOu blocks.

The process tag type contains the required interconnections between the blocks mentioned above.

18.1.23 Motor (MotorLean)

Motor with current monitoring

This process tag type serves as a basis for controlling motors with one control signal using the MotL block.

The feedback signal of the motor is read from the I/O through the PCS7DiIn block.

The interlock signals of MotL are connected to the IntLk02 interlock blocks. These interlock blocks in turn are connected to other blocks, for example, to digital process tags via Pcs7DiIn.

The digital output signals are sent to the I/O through PCS7DiOu blocks.

The process tag type contains the required interconnections between the blocks mentioned above.

18.1.24 Two-speed motor (Motor2Speed)

Two-speed motor

This process tag type serves as a basis for controlling motors with two speeds using the MotSpdL block.

The speed feedback signals of the motor are read from the I/O through PCS7DiIn blocks.

The interlock signals of MotSpdL are connected to the IntLk02 interlock blocks. These interlock blocks in turn are connected to other blocks, for example, to digital process tags via Pcs7DiIn.

The digital output signals are sent to the I/O through PCS7DiOu blocks.

The process tag type contains the required interconnections between the blocks mentioned above.

18.1.25 Reversing motor (MotorReversible)

Reversing motor

This process tag type serves as a basis for controlling reversing motors using the MotRevL block.

The feedback signals of the motor are read from the I/O through PCS7DiIn blocks.

The interlock signals of MotRevL are connected to the IntLk02 interlock blocks. These interlock blocks in turn are connected to other blocks, for example, to digital process tags via Pcs7DiIn.

The digital output signals are sent to the I/O through PCS7DiOu blocks.

The process tag type contains the required interconnections between the blocks mentioned above.

18.1.26 Reversing motor with controllable speed (MotorSpeedControlled)

Reversing motor with controllable speed

This process tag type serves as a basis for controlling reversing motors with controllable speed using the MotSpdCL block.

The digital feedback signals of the motor are read from the I/O through PCS7DiIn blocks.

The speed feedback of the motor is read from the I/O through the PCS7AnIn block.

The interlock signals of MotSpdCL are connected to the IntLk02 interlock blocks. These interlock blocks in turn are connected to other blocks, for example, to digital process tags via Pcs7DiIn.

The digital output signals are sent to the I/O through PCS7DiOu blocks.

The speed control signal is output to the I/O through the PCS7AnOu block.

The process tag type contains the required interconnections between the blocks mentioned above.

18.1.27 Motor with an additional analog value and time-stamped signals (Motor_AV_EventTs)

Motor with an additional analog value and time-stamped signals

This process tag type shows the monitoring of an additional analog value, for example, for motor current monitoring (AV block) and additional binary signals (EventTs block) at a technological block with MotL as an example.

By interconnecting the AV or EventTs block, messages of this block are displayed in the message view of the technological block connected to it and can be acknowledged there.

Use of the AV block

The analog signal to be monitored can be read via the Pcs7AnIn block, for example. This has to be interconnected to the AV block. The AV_Tech output parameter of the AV block has to be interconnected to the AV input parameter of the technological block.

Use of the EventTs block

A non-time-stamped binary signal from the I/O can be read via the Pcs7DiIn block, for example. The PV_Out output parameter of Pcs7DiIn has to be interconnected to the Inx input parameter of the EventTs block. The time stamp of this signal is generated by the EventTs block at signal change.

A time-stamped binary signal from the I/O must be read via the Pcs7DiIT block. The TS_Out output parameter of Pcs7DiIT has to be interconnected to the InTSx input parameter of the EventTs block.

The EventTsOut output parameter of the EventTs block has to be interconnected to the EventTsIn input parameter of the technological block.

Additional interconnections

The interlock signals of MotL are interconnected to the IntLk02 interlock blocks. In turn, these interlock blocks are interconnected to other blocks, for example, to digital process tags via the Pcs7DiIn block.

The digital output signal is output to the I/O by means of the PCS7DiOu block.

The process tag type contains the required interconnections between the blocks mentioned above.

18.1.28 Valve (ValveLean)

Valve

This process tag type serves as a basis for controlling a valve with two positions (open/close) using the VlvL block.

The feedback signals of the valve are read from the I/O through PCS7DiIn blocks.

The interlock signals of VlvL are connected to the IntLk02 interlock blocks. These interlock blocks in turn are connected to other blocks, for example, to digital process tags via Pcs7DiIn.

The digital output signals are sent to the I/O through PCS7DiOu blocks.

The process tag type contains the required interconnections between the blocks mentioned above.

18.1.29 Two-way valve (Valve2Way)

Two-way valve

This serves as a basis for controlling a

- multi-way valves with up to three switching positions or
- three individual valves (valve network) to implement a 2-way valve circuit with safe position

using the Vlv2WayL block.

The feedback signals of the valve or valves are read from the I/O through PCS7DiIn blocks.

The interlock signals of Vlv2WayL are connected to the IntLk02 interlock blocks. These interlock blocks in turn are connected to other blocks, for example, to digital process tags via Pcs7DiIn.

The digital output signals are sent to the I/O through PCS7DiOu blocks.

The process tag type contains the required interconnections between the blocks mentioned above.

18.1.30 Motor valve (ValveMotor)

Motor valve

This process tag type serves as a basis for controlling a motor valve using the VlvMotL block.

The feedback signals of the valve are read from the I/O through PCS7DiIn blocks.

The interlock signals of VlvMotL are connected to the IntLk02 interlock blocks. These interlock blocks in turn are connected to other blocks, for example, to digital process tags via Pcs7DiIn.

The digital output signals are sent to the I/O through PCS7DiOu blocks.

The process tag type contains the required interconnections between the blocks mentioned above.

18.2 Example project APL_Example_xx

18.2.1 Introduction to the PCS 7 example project for Advanced Process Control

Introduction

This document relates to the PCS 7 example project, Process Control (APL_Example_xx, xx indicates the language variant) for the PCS7 Advanced Process Library.

In contrast to the process tag types (insertable templates), the example project is primarily intended for training purposes:

The main aim was to familiarize users with with the new advanced process control structures by allowing them to experiment without having to intervene in the real process. The examples provide realistic process simulation. Working with these examples helps you to understand the concept and specific structural requirements, and to assess the uses of these functions before you implement them in a real system. For this reason, the examples contain a third order simulation model with gain, equivalence value and measurement noise but no analog driver blocks. The process model is supplied as the "ProcSimC" CFC chart and incorporated in the examples using the chart-in-chart technique.

The following examples are provided:

- Cascade control of a temperature by using the heat flow (Page 1458)
- Control loop monitoring with simulation of colored noise (Page 1460)
- Feedforward control to compensate a measurable disturbance variable (Page 1461)
- Operating point-oriented adaptation of parameters (gain scheduling) for non-linear processes (Page 1462)
- Override control on a pipeline (Page 1462)
- Smith predicator for a dead time controlled system (Page 1463)
- Filtering of noisy measured values in a control loop (Page 1463)
- Predictive control of a 2x2 multi-variable controlled system (Page 1464)
- Predictive control of a non-linear process (Page 1465)

A template for Fuzzy Control using the *PCS 7 add-on product FuzzyControl++* can be downloaded from the Internet pages of *Siemens I&S* and is therefore not included in the demo project.

A separate tree folder ("Unit") is provided for each example in the PCS7 example project. Each folder contains a CFC chart with an interconnection example, a brief explanatory text, and an assigned OS picture with self-explanatory visualization of the process example based on a pre-configured trend recorder. A short text in the OS picture describes commissioning and presentation.

18.2.2 Process simulation including noise generator

Process simulation including noise generator

Users require only a few standard blocks to create a dynamic process model that reflects the response pattern of many technological processes with adequate precision. This model is used in all example projects (APL_Example_xx). However, you can also use this model for sales presentations or to test closed-loop control functions, in other words in a project phase in which the real plant is not yet available ("virtual process", "shadow plant"). Process simulation is supplied as an open source CFC chart "ProcSimC" that users can install in other CFC charts as a nested chart (chart-in-chart). It contains three first order delay elements, a gain factor, an equivalence value PV (for `MV = 0`), and a noise generator for white measurement noise. An additive input is provided for (artificial) interference of the input. The Laplace transformation function described below is implemented by means of this model.

$$PV(s) = \frac{Gain}{(TimeLag1 \cdot s + 1)(TimeLag2 \cdot s + 1)(TimeLag3 \cdot s + 1)}(MV(s) + DisV(s)) + PV_0 + Noise$$

Use cases

Users can adapt this flexible model to suit the requirements of various use cases, for example:

- **Simulation of temperature control systems:** `PV0` represents the temperature without heating, for example, the ambient temperature. The value of `TimeLag1` is typically significantly higher than `TimeLag2` and `TimeLag3`. The value of the latter can also be zero. The sensor develops a typical quantization noise of 0.1° C. The gain is positive can be interpreted as the theoretical maximum temperature that can be reached at full heating power. However, in most situations this cannot be measured experimentally because many actuators are dimensioned so that the only require approximately one third of the heating power for constant operation at the operating point. The power reserve is only intended to cover operating point changes and heating up phases.
- **Simulation of pressure control systems:** If you define the valve position so that it is closed at 0% and open at 100%, the process Gain of a container pressure control system is normally a negative value because the pressure reduces (>0) when the outlet valve of the container is opened. In contrast to this, the gain of an overpressure value is positive. `PV0 > 0` is the pressure when the value is completely closed. The situation is, of course, the opposite when pressures below ambient pressure are involved, for example, in vacuum systems. Note that most valves do not return a reproducible characteristic in the region of their closed position (actuation ratio 1:20 or 1:50). The time constants for pressure control of liquids are typically fast, whereas with pressure control in gas tanks, particularly in large tanks, they are slower. The magnitude of the process gain depends largely on the physical units of pressure, for example, Bar or Pa. Pressure sensors typically develop higher measurement noise than temperature sensors.

- **Simulation of flow control systems:** If you define the valve position so that it is closed at 0% and open at 100%, the process `Gain` is usually positive, since the flow rate increases when the valve opens. `PV0 = 0` if the flow stops completely when the valve is closed, in other words, the valve closes tight. The time constants are significantly faster than in temperature controls and are usually all of the same order. The magnitude of the process gain depends largely on the physical units of flow, for example, m3/s or l/min. The measurement noise affecting flow sensors is normally higher than with temperature sensors.

To implement a dead time for process simulation, you can insert a DeadTime block before the ProcSimC input and call it in a different cyclic interrupt OB (OB3x).

Model variants

Two different model variants are supplied:

1. Continuous process simulation ProcSimC in which the `MV` input is an analog value, for example, heating power or valve position.
2. Process simulation ProcSimS for step controllers, with control of the actuator by two binary inputs "up"/"down" or "open"/"close". Internally, the actuator is modeled as an integrator, whereby:

   - MotorHiLim = 100%,

   - MotorLoLim = 0%

   - TI = MotorTime.

   The integrator input is derived from the binary inputs according to the following formula:

$$\underset{Integ.Input}{\overset{I}{}} = \left\{ \begin{array}{rl} 100 & if\ Up = True \\ -100 & if\ Down = True \\ 0 & otherwise \end{array} \right\}$$

See also

NoiseGen I/Os (Page 266)

18.2.3 Cascade control of a temperature by using the heat flow

Cascade control of a temperature by using the heat flow

This template contains simulation models for a flow and temperature controlled system and the parameters of the noise generator block (see the I/O table). You can use this model to test the mode transitions described in the Cascade control (Page 1442) section. You can also try out the properties of different parameter sets for the primary and secondary controllers. The following features are typical for this type of application:

- The controlled temperature system is slower than the controlled flow system.
- There are two time constants that are far apart.
- There is an offset corresponding to ambient temperature.
- It has less noise than the controlled flow system.

Process parameters of the example project for cascade control

ProcSimC	Gain	TimeLag1	TimeLag2	PV0	NoiceVariance
Flow control loop	8	1	1	0	0.05
Temperature control loop	0.3	8	1	20	0.01

Parameters for PID controllers --> PI cascade with fast control response

The parameters listed in the table below apply to a fast control response with low control deviation but with strong actuator intervention.

PID	Gain	TI	TD
TIC-501	10	8.8	2.6
FIC501	0.1	1.8	0

Controller parameters for PI --> P cascade with soft controller intervention

The advantage of the parameters listed in the table below is that the controller "goes easy" on final control element (for example a valve).

PID	Gain	TI	TD
TIC-501	10.5	6.8	0
FIC501	0.1	0	0

It is generally advisable to make the secondary controller "simpler" than the primary controller, in other words to reduce the number of different dynamic channels so that it is genuinely secondary to the primary controller.

The steady state control deviation in the secondary loop is not normally relevant for the application. On the other hand, the reaction time of the secondary loop is important because the time constants of the secondary closed control loop are part of the controlled system for the primary controller. If you do without I action in the secondary controller for these reasons, it is not advisable limit the setpoint ranges of the secondary controller precisely to the achievable physical range of the process value in the secondary loop, as you would not be able to use the full actuating range of the secondary controller due to the steady state deviation. You should instead set more generous setpoint limits for the secondary controller and manipulated variable limits for the primary controller. The anti-Windup measures of the primary controller are oriented on the interconnection of `IntHoldNeg` and `IntHoldPos`. If the secondary controller does not have any I action, if will not be capable of a bumpless manual-automatic changeover. You should therefore set an `MV_Offset` that approximates the typical `MV` value for the operating point of the process.

A cascade temperature control system with a secondary controller for heating and/or cooling medium flow is commonly used for

- Heat exchangers
- Reactors without a cooling jacket

18.2.4 Control loop monitoring with simulation of colored noise

Control loop monitoring with simulation of colored noise

The interconnection of the ConPerMon block with a PID controller can be found in the process tag type PID_Control (see PID controller with safety logic and control loop monitoring (PIDConL_ConPerMon) (Page 1430)). The example project supports you and helps to familiarize you with the concept and the potential of closed control loop monitoring. To do this, the template includes a process simulation with disturbance model. The colored noise is generated with the aid of a shape filter from a white noise signal. This produces a spectrum of disturbance signals that also contains energy components in the lower frequency ranges of the bandwidth of the closed control loop. Part of the disturbances can therefore be compensated by the PID controller while the high-frequency measurement noise cannot be corrected by any controller.

Application

After commissioning the controller and ConPerMon block, you should be able to watch the effects of the following actions that demonstrate the potential of control loop monitoring:

- Switch the controller to manual mode:

  The variance of the controlled variable will rise but the CPI becomes invalid because no statements can be made about the control fit unless the control loop is closed.

- Change the parameters of the process simulation, for example, change TimeLag2 from 2s to 8s:

  This deterioration of the dynamic characteristics of the process (for example due to wear and tear) brings about a deterioration of the control quality that becomes visible in the CPI value long before it can be seen with the naked eye in the standard PV trends. If the control quality drops below a defined level, a CPI warning or even an alarm is generated.

- Request a setpoint step change from the controller:

  The CPI will become temporarily invalid because all stochastic characteristics of the control quality, such as the variance, are based on the assumption of a steady state with a constant mean value. Select the "Setpoint" view from the drop-down list box in the ConPerMon faceplate to be able to watch the deterministic characteristics such as overshoot and settling ratio. Once a steady state is achieved again at the new setpoint and the entire time window is filled with data from the steady state, the monitoring of the stochastic characteristics is reactivated automatically.

You can find detailed information on the ConPerMon block and notes on interpreting its displays in the online help on the block (Page 267).

18.2.5 Feedforward control to compensate a measurable disturbance variable

Feedforward control to compensate a measurable disturbance variable

The example is based on the process tag type PID controller with dynamic feedforward control (FfwdDisturbCompensat) (Page 1433) and uses the following parameter sets:

Main controlled system:

$$g(s) = \frac{2}{2s+1} e^{-1.2s}$$

Disturbance transfer function:

$$g_z(s) = \frac{1}{3s+1} e^{-1.6s}$$

- PID: Gain = 0.197
- TI= 1.9
- TD= 0

Feedforward control:

$$c(s) = -\frac{g_z(s)}{g(s)} = -\frac{1}{2} \cdot \frac{2s+1}{3s+1} e^{-0.4s}$$

The same process simulation is set up twice, one instance with disturbance feedforward and the other without (all other process and controller parameters identical). The advantages of the feedforward control can be tested in a direct comparison ("benchmark simulation", "parallel slalom ").

18.2.6 Operating point-oriented adaptation of parameters (gain scheduling) for non-linear processes

Operating point-oriented adaptation of parameters (gain scheduling) for non-linear processes

The example is based on the process tag type PID - control with operating-point-oriented parameter control (GainScheduling) (Page 1431).

In the simulation template, the settings of the two most important process parameters are changed based on polylines depending on the operating point. The process and controller parameters for the example are shown in the following table.

Operating point	X=PV	ProcSim.Gain	ProcSim.TmLag1	ProcSim.TmLag2	Gain	TI	TD
1	20	4	5	10	0.6	14.7	3.7
2	100	3	3	10	1	8.8	2.2
3	200	2	1	10	10	4.1	1.1

The same process simulation is set up twice, one instance with gain scheduling and the other without (all other process and controller parameters are identical). The advantages of the gain scheduling can be tested in a direct comparison ("benchmark simulation", "parallel slalom").

18.2.7 Override control on a pipeline

Override control on a pipeline

The example is based on the process tag type Override control (Page 1445) and uses the following parameter sets:

Primary process (flow control):

$$g(s) = \frac{3}{(2s+1)^2}$$

Flow increases when the valve is opened and it disappears when the valve is closed.

PI flow controller: Gain= 0.33 , TI= 2.7

Secondary process (pressure control):

$$g_p(s) = \frac{-0.8}{(7s+1)(1s+1)}$$,

The pressure rises when the valve is opened and is 80 bar when it is fully open.

PI pressure controller: Gain= -2.8 , TI= 4

Switching limits 15 bar < pressure < 70 bar.

18.2.8 Smith predicator for a dead time controlled system

Smith predicator for a dead time controlled system

The example is based on the process tag type Predictive control of a 2x2 multi-variable controlled system (Page 1464).

In the example, the same process simulation is set up twice, one instance with Smith predictor and the other without (all other process parameters are identical). The advantages of the Smith predictor can be tested in a direct comparison ("benchmark simulation", "parallel slalom").

18.2.9 Filtering of noisy measured values in a control loop

Filtering of noisy measured values in a control loop

The example illustrates the use of the Smooth block in a closed control loop. The block can be connected to any signal source without specialist knowledge so there is no need for a special process tag type. The simulation template is useful in testing the effects of a low-pass filter on a closed control loop by simulation. Increasing the filter time constant improves the smoothing effect but also causes a phase lag in the control loop that can have detrimental effects on the control quality and even the stability.

Parameters used

The following parameters are used in the simulation example:

Process transfer function:

$$g(s) = \frac{3}{(15s+1)(2s+1)}$$

with white noise on the output signal.

PI controller:

- Gain = 0,5
- TI = 7 s
- Sample time = 0.1s

Butterworth filter:

- TimeConstant = 3 s.

At 0.3 seconds, hardly any smoothing effect can be recognized, at 15 seconds significant deterioration of the control quality is already noticeable.

Processes with signals strongly affected by noise are a typical area of application (for example pressure sensors) and sensitive actuators (for example valves).

You can find detailed information on the Smooth block in the online help on the block (Page 1145).

18.2.10 Predictive control of a 2x2 multi-variable controlled system

Predictive control of a 2x2 multi-variable controlled system

The example is based on the process tag type Model-based predictive control (ModPreCon) (Page 1447).

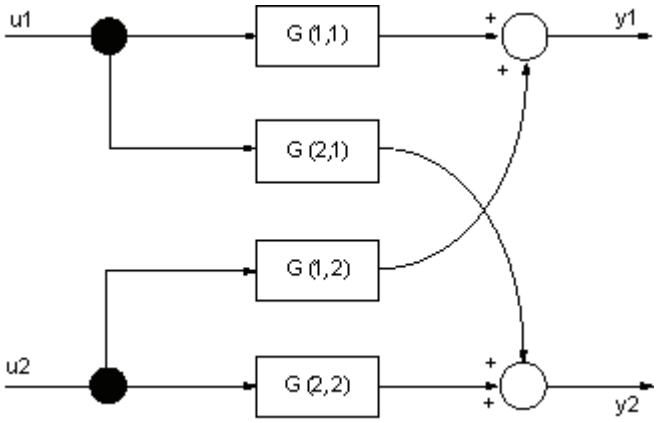


Figure 18-7 MIMO 2x2 process with a p-canonical structure

The example shows the application of the ModPreCon block for simulating a 2x2 multi-variable process consisting of the following four transfer functions:

$$\underline{G}(s) = \begin{bmatrix} G(1,1) & G(1,2) & \cdots & G(1,n_u) \\ G(2,1) & G(2,2) & \cdots & G(2,n_u) \\ \vdots & \vdots & \ddots & \vdots \\ G(n_y,1) & G(n_y,2) & \cdots & G(n_y,n_u) \end{bmatrix} = \begin{bmatrix} \dfrac{3}{(30s+1)(4s+1)} & \dfrac{1.2}{(34s+1)(14s+1)(6s+1)} \\ \dfrac{1.3}{(28s+1)(12s+1)(6s+1)} & \dfrac{4}{(26s+1)(6s+1)} \end{bmatrix}$$

where n_y = 2 = number of controlled variables, n_u = 2 = number of manipulated variables, and $G(i_y, i_u)$ the transfer function from input i_u to output i_y. This simplest of multi-variable control systems helps familiarize newcomers with the concept and application of model-based multi-variable controllers.

18.2.11 Predictive control of a non-linear process

Predictive control of a non-linear process

The example is based on the approach used by multi-model controlling as described in the section ModPreCon functions (Page 396), Controlling linear and non-linear processes.

A multi-variable process is observed with two input variables and two output variables. The non-linear reaction of four partial transfer functions Proc511, Proc512, Proc521 and Proc522 depends on a measurable process value, in this case the controlled variable PV511.

The assumption that all non-linearities of the multi-variable process depend on the current operating point, which is defined by a single measurable variable, restricts the range of application but is reasonable in many practical applications. The approach of the presented multi-model controlling only makes sense with this assumption.

In the example it is assumed that the operating point is defined by a temperature, and the process reaction is different at high temperatures (200° C) than it is at low temperatures (20° C).

Some parameters of the third order partial transfer function

$$Proc(i, j) = \frac{CV(i)}{MV(j)} = \frac{Gain}{(TmLag1 \cdot s + 1) \cdot (TmLag2 \cdot s + 1)(TmLag3 \cdot s + 1)}$$

are set using polylines continually depending on the operating point.

The external values of the operating-point-oriented parameters for the example are shown in the following table:

Operating point	PV511	Proc511.Gain	Proc511.TmLag2	Proc521.Gain	Proc512.TmLag2	Proc522.Gain
1	20	4	12	0,9	19	3
2	200	2	2	1,7	9	5

The changes of the process parameters are so substantial that a single linear controller cannot achieve enough control quality over the entire operating range. The changes, however, are continual and reproducible, which is an important requirement for the multi-model approach.

All other process parameters are constant:

	Gain	TmLag1	TmLag2	TmLag3
Proc511	variable	30	variable	0
Proc512	1,2	34	variable	6
Proc521	variable	28	12	6
Proc522	variable	26	6	0

The primary controller TIC511522Low was designed using the MPC Configurator for the low operating point at 20° C, the secondary controller TIC511522High is made for the high operating point at 200° C.

The matching functions for relevance of the two controllers are polylines with four interpolation points at 0, 30, 190 and 300° C. In the range between 0 and 30° C, only the controller designed for 20° C is active; only the controller designed for 200° C is active between 190 and 300° C. The manipulated variables of both controllers overlap between 30 and 190°.

19 Definitions

19.1 Batch process

Batch process

A batch process is a process control process, which is executed according to a recipe control batch by batch, i.e. intermittently, in a continually repeating sequence, for example, dosing raw material, tempering, performing chemical reactions, cooling, discharging reactors.

19.2 Approximation

Approximation

An approximation method in the mathematical sense.

19.3 Prediction horizon

Prediction horizon

For predictive controller: Time period running from the present to the future with a defined length. A process reaction is predicted within the prediction horizon.

19.4 Trajectory

Trajectory

In physics: refers to a flight path or track.

In control engineering: course of a variable over time, described by a sequence of values in a specified time scale.

19.5 Maverick

Maverick

A maverick in a continuous physical measurement is a numerical value that changes from one sampling point to another more than would be physically plausible. In other words, the difference between two neighboring values is greater than a specified tolerance range.

19.6 Ergodic process

Ergodic process

An ergodic process in mathematical statistics is a stationary process, in which the expected value can be estimated by generating the mean value over a time period of infinite length.

19.7 Conti process

Conti process

A Conti process is a process control process, whereby raw material is fed in a continual flow, and the products are continually output.

19.8 Multivariable controller

Multivariable controller

With a multivariable controller, one manipulated variable can influence several controlled variables and one controlled variable can be influenced by several manipulated variables, as is shown in the following diagram.

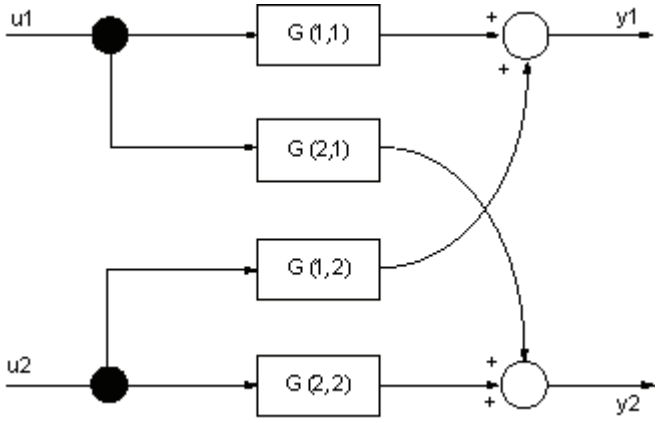


Figure 19-1 Example of a multivariable controller

When a multivariable section is automated using several individual PID controllers, the individual controllers do not take account of the interactions or connections in the process. The greater the connections between the sub-sections, the harder it is to set individual controllers and the worse the control performance.

In such cases a multivariable controller offers higher control performance and simpler controller settings.

19.9 non-phase minimum

Non-phase minimum behavior mean

A phase minimum system is described by a linear, time-invariant transfer function, the frequency of which displays the smallest possible clockwise phase rotation for the given number of poles and zeros if the frequency interval is run through from negative to positive in full an infinite number of times. This means that both the transfer function and its inverse are causal and stable.

Non-phase minimum behavior means for example that the process initially deflects downwards when the manipulated variable jumps positively before moving in a positive direction. Dead time systems are also non-phase minimal.

Index

B

C

D

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

X