



Level



Pressure



Flow



Temperature

Liquid
Analysis

Registration

Systems
Components

Services



Solutions

Operating Instructions

ControlCare Application Designer

Function Blocks

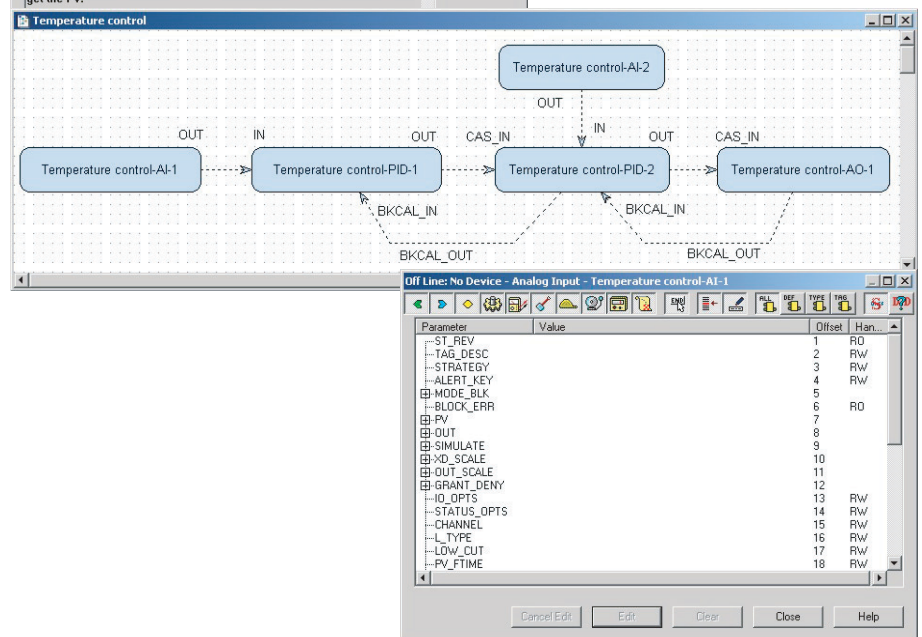
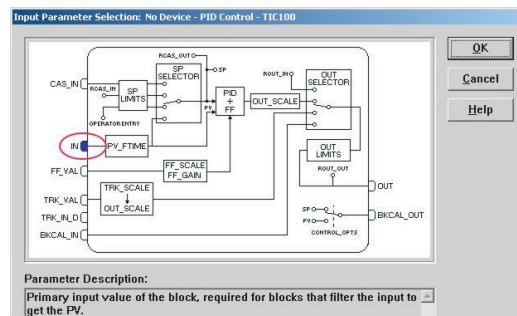


Table of Contents

Revision History	6	2.11 Changing function block parameters	46
Product Version	6	2.11.1 Off-line characterization	46
Registered Trademarks	6	2.11.2 On-line characterization	47
1 Safety	7	3 Resource Block	48
1.1 Designated use	7	3.1 Informative parameters	48
1.2 Installation, commissioning and operation	7	3.1.1 Device software	48
1.3 Operational safety	7	3.1.2 Device hardware	49
1.4 Conventions and icons	8	3.1.3 Hardware write lock	49
1.5 ControlCare documents	9	3.1.4 Simulation	49
1.6 About this manual	10	3.2 Operational parameters	50
2 Function Block Application Process .	11	3.2.1 Block operation	50
2.1 Introduction	11	3.2.2 Resource state	50
2.2 Block objects	12	3.2.3 Resource restart	51
2.3 Block parameters	13	3.2.4 Resource options	51
2.3.1 Parameter types	13	3.2.5 Alert reporting	52
2.3.2 Parameter identifiers	14	3.2.6 Software write lock	53
2.3.3 Universal parameters	15	3.2.7 Block errors	53
2.3.4 Parameters in Application Designer	16	3.3 Block parameters	54
2.4 Block operation	18	4 Transducer Blocks	56
2.4.1 Block mode parameter MODE_BLK	18	4.1 Hardware Configuration block	57
2.4.2 Block modes	18	4.1.1 Block configuration	57
2.4.3 Mode priority	20	4.1.2 Block errors	58
2.4.4 Mode shedding	22	4.1.3 Block parameters	58
2.5 Parameter calculation	23	4.2 Temperature Transducer block	59
2.5.1 Process variable PV	23	4.2.1 Block configuration	59
2.5.2 Scaling parameters XD_SCALE, PV_SCALE, OUT_SCALE	24	4.2.2 Block errors	59
2.5.3 Setpoint SP	26	4.2.3 Block parameters	60
2.5.4 Output variable, OUT	27	4.3 Device transducer blocks (Endress+Hauser)	61
2.5.5 Back calculation	27	4.3.1 Block configuration	62
2.6 Status handling	28	4.3.2 Block parameters	62
2.6.1 Status propagation	28	4.4 Display block (DISP)	63
2.6.2 Status message	28	4.4.1 Block configuration	63
2.6.3 Substatus and limit propagation	30	4.4.2 Block parameters	63
2.6.4 Control loop integrity	31	4.5 Diagnosis block (DIAG)	64
2.7 Fault state handling	34	4.5.1 Block configuration	64
2.7.1 Control blocks	34	4.5.2 Block parameters	64
2.7.2 Output blocks	34	5 Input Blocks	65
2.7.3 Forcing fault state	35	5.1 CHANNEL parameter	65
2.8 Function block options	36	5.2 Analog Input (AI)	66
2.8.1 GRANT_DENY	36	5.2.1 Functional description	66
2.8.2 IO_OPTS	37	5.2.2 Block Configuration	68
2.8.3 CONTROL_OPTS	38	5.2.3 Block operation	69
2.9 Alert handling	39	5.2.4 SFC445 temperature ranges (XD_SCALE)	70
2.9.1 Events and alarms	39	5.2.5 Block parameters	71
2.9.2 Reporting, priority and acknowledgement	40	5.3 Discrete Input (DI)	72
2.9.3 Limit alarms	41	5.3.1 Functional description	72
2.9.4 Event update and alarm summaries	42	5.3.2 Block configuration	73
2.9.5 Alarm display parameters	43	5.3.3 Block operation	73
2.9.6 Example	44	5.3.4 Block parameters	74
2.10 Simulation	45		

5.4	Pulse Input (PULS)	75	7.3	Signal Characterizer	130
5.4.1	Functional description	75	7.3.1	Functional Description	130
5.4.2	Block configuration	76	7.3.2	Block configuration	131
5.4.3	Block operation	76	7.3.3	Block operation	132
5.4.4	Block parameters	78	7.3.4	Block parameters	133
5.5	Multiple Analog Input (MAI)	79	7.4	Cascade Signal Characterizer	134
5.5.1	Functional description	79	7.4.1	Functional Description	135
5.5.2	Block configuration	79	7.4.2	Block configuration	136
5.5.3	Block operation	80	7.4.3	Block operation	138
5.5.4	Block parameters	80	7.4.4	Block Errors	138
5.6	Multiple Discrete Input (MDI)	81	7.4.5	Block parameters	139
5.6.1	Functional description	81	7.5	Integrator	140
5.6.2	Block configuration	81	7.5.1	Functional description	140
5.6.3	Block operation	82	7.5.2	Block configuration	144
5.6.4	Block parameters	82	7.5.3	Block operation	145
6	Control Blocks	83	7.5.4	Block parameters	146
6.1	PID Control	83	7.6	Analog Alarm	148
6.1.1	Functional description	84	7.6.1	Functional description	148
6.1.2	Closed-loop control	87	7.6.2	Block configuration	151
6.1.3	Feedforward control	88	7.6.3	Block operation	153
6.1.4	Cascade control	90	7.6.4	Block parameters	154
6.1.5	Output tracking	92	7.7	Enhanced Analog Alarm	155
6.1.6	Remote cascade	93	7.8	Input Selector	156
6.1.7	Remote output	95	7.8.1	Functional description	156
6.1.8	Block operation	97	7.8.2	Block configuration	158
6.1.9	Block parameters	98	7.8.3	Block operation	159
6.2	Enhanced PID Control	100	7.8.4	Block parameters	160
6.2.1	Bumpless transfer	100	7.9	Setpoint Ramp Generator (SPG)	161
6.2.2	Enhanced output tracking	101	7.9.1	Functional description	161
6.2.3	Additional parameters	102	7.9.2	Block configuration	164
6.3	Advanced PID Control	103	7.9.3	Block operation	166
6.3.1	PID algorithm	104	7.9.4	Block parameters	166
6.3.2	PI sampling	104	7.10	Enhanced Setpoint Ramp Generator	168
6.3.3	Adaptive gain	105	7.11	Timer and Logic	169
6.3.4	Anti-reset wind-up	105	7.11.1	Functional description	169
6.3.5	Special treatment for error	106	7.11.2	Block configuration	176
6.3.6	Mode indication	107	7.11.3	Block operation	177
6.3.7	Additional parameters	107	7.11.4	Block parameters	177
6.4	Step Output PID	108	7.12	Lead-Lag	179
6.4.1	Functional description	108	7.12.1	Functional description	179
6.4.2	Block configuration	111	7.12.2	Block configuration	182
6.4.3	Block operation	113	7.12.3	Block operation	183
6.4.4	Block parameters	114	7.12.4	Block parameters	183
7	Calculation and Logic Blocks	116	7.13	Output Signal Selector and Dynamic Limiter	184
7.1	Arithmetic	116	7.13.1	Functional description	184
7.1.1	Range extension	117	7.13.2	Block configuration	186
7.1.2	Calculated variables	119	7.13.3	Block operation	190
7.1.3	Block operation	122	7.13.4	Block parameters	191
7.1.4	Block parameters	123	7.14	Constant	192
7.2	Output Splitter	124	7.14.1	Functional description	192
7.2.1	Functional description	125	7.14.2	Block configuration	192
7.2.2	Block configuration	127	7.14.3	Block operation	193
7.2.3	Block operation	128	7.14.4	Block parameters	193
7.2.4	Block parameters	129	7.15	Constant and Contained RW	194
			7.15.1	Functional description	194
			7.15.2	Block configuration	196
			7.15.3	Block operation	198
			7.15.4	Block parameters	199

7.16	Flip-flop and Edge Trigger	201	9	PROFIBUS Blocks	244
7.16.1	Functional description	201	9.1	General description	245
7.16.2	Block configuration	205	9.1.1	Accessing cyclic PROFIBUS I/O data ..	245
7.16.3	Block operation	207	9.1.2	Failsafe/Fault state mechanism	246
7.16.4	Block parameters	208	9.1.3	Configuring PROFIBUS input and output values	247
7.17	Advanced Equations	209	9.2	PROFIBUS Transducer	248
7.17.1	Functional description	209	9.3	PROFIBUS Analog Input	252
7.17.2	Block configuration	209	9.3.1	Functional description	252
7.17.3	Block operation	210	9.3.2	Block operation	252
7.17.4	Block parameters	210	9.3.3	Block parameters	252
7.18	Density	211	9.4	PROFIBUS Multiple Analog Input	253
7.18.1	Functional description	211	9.4.1	Functional description	253
7.18.2	Block configuration	213	9.4.2	Block Configuration	254
7.18.3	Block operation	214	9.4.3	Block operation	254
7.18.4	Block parameters	215	9.4.4	Block parameters	255
7.19	Hybrid with Analog I/O	216	9.5	PROFIBUS Digital Input	257
7.19.1	Functional description	217	9.5.1	Functional description	257
7.19.2	Block operation	218	9.5.2	Block operation	257
7.19.3	Block parameters	218	9.5.3	Block parameters	257
7.20	Hybrid with Discrete I/O	219	9.6	PROFIBUS Multiple Discrete Input	258
7.20.1	Functional description	220	9.6.1	Functional description	258
7.20.2	Block operation	221	9.6.2	Block operation	258
7.20.3	Block parameters	221	9.6.3	Block parameters	259
7.21	Hybrid with Embedded I/O	222	9.7	PROFIBUS Totalizer	260
7.21.1	Functional description	223	9.7.1	Functional description	260
7.21.2	Block operation	224	9.7.2	Block operation	261
7.21.3	Block parameters	225	9.7.3	Block parameters	261
8	Output Blocks	226	9.8	PROFIBUS Analog Output	262
8.1	Analog Output	226	9.8.1	Functional Description	262
8.1.1	Functional description	226	9.8.2	Block configuration	263
8.1.2	Output configuration for SFC446	228	9.8.3	Block operation	267
8.1.3	Block configuration	229	9.8.4	Block parameters	267
8.1.4	Block operation	230	9.9	PROFIBUS Multiple Analog Output	269
8.1.5	Block parameters	231	9.9.1	Functional description	269
8.2	Discrete Output	232	9.9.2	Block operation	270
8.2.1	Functional description	232	9.9.3	Block Configuration	270
8.2.2	Output configuration for SFC428, SFC432, SFC435, SFC438	233	9.9.4	Block parameters	271
8.2.3	Block configuration	234	9.10	PROFIBUS Discrete Output	273
8.2.4	Block operation	235	9.10.1	Functional description	273
8.2.5	Block parameters	236	9.10.2	Block configuration	274
8.3	Multiple Analog Output	237	9.10.3	Block operation	275
8.3.1	Functional description	237	9.10.4	Block parameters	275
8.3.2	Block configuration	238	9.11	PROFIBUS Multiple Discrete Output	276
8.3.3	Block operation	239	9.11.1	Functional description	276
8.3.4	Block parameters	240	9.11.2	Block configuration	276
8.4	Multiple Discrete Output	241	9.11.3	Block operation	277
8.4.1	Functional description	241	9.11.4	Block parameters	278
8.4.2	Block configuration	242			
8.4.3	Block operation	242			
8.4.4	Block parameters	243			

10	Modbus Blocks	279	Appendix A:
10.1	ControlCare Implementation	279	Interpreting parameter status
10.1.1	Register access	280	Appendix B:
10.1.2	Data types	281	Standard data types and structures
10.2	MBCF Modbus Configuration Block	282	Appendix C:
10.2.1	Configuration as TCP/IP master	283	Manufacturer-specific data structures
10.2.2	Configuration as TCP/IP slave	283	Index
10.2.3	Configuration as serial master	284	
10.2.4	Configuration as serial slave	284	
10.2.5	Configuration as master and slave	284	
10.3	Modbus Control Master	286	
10.3.1	Block description	286	
10.3.2	Block configuration	288	
10.3.3	Block operation	291	
10.3.4	Block parameters	292	
10.4	Modbus Control Slave	293	
10.4.1	Block description	293	
10.4.2	Block configuration	296	
10.4.3	Block operation	298	
10.4.4	Block parameters	299	

Revision History

Product version	Manual	Changes	Remarks
1.00.xx	BA022S/04/en/07.02	Original manual	
2.01.xx	BA022S/04/en/01.06	Editorial	<ul style="list-style-type: none"> ■ In new Corporate design ■ SFC151 updated to SFC162 and SFC173. ■ Configuration Tool now named Application Designer ■ The descriptions of the function blocks have not changed and remain valid over the Product Versions. ■ The use of special blocks for PROFIBUS in control strategy creation are described in BA036S/04/en: Application Designer: PROFIBUS Tutorial
2.02.xx	BA022S/04/en/07.06	Editorial	<ul style="list-style-type: none"> ■ Note on Field Controller DIP-switch added to Simulation (Chap. 1.2) ■ Pulse Input description improved (Chap 5.5) ■ Enhanced Signal Characterizer (50 points, cascade) described (Chap 7.4)
2.03.xx	BA022S/04/en/06.07 (Not released)	Editorial	<ul style="list-style-type: none"> ■ Restructuring of manual ■ Parameter, block descriptions updated
2.04.xx	BA022S/04/en/12.08	Function Blocks	<ul style="list-style-type: none"> ■ Additional Enhanced Blocks ■ Hybrid Function blocks offer more parameters ■ Modbus blocks reduced and restructured
2.05.xx	BA022S/04/en/06.10	Modbus MBCF Block	■ Addition of master+slave option(Chapter 10.2.5)
		Input Selector ISEL Block	■ Description OP_SELECT corrected, p157

Product Version

Details of product version and the individual components of Application Designer Suite can be seen in the About ControlCare dialog:

Start=>Programs=>Endress+Hauser=>ControlCare=>Tools=>About ControlCare

Registered Trademarks

PROFIBUS®

Registered trademark of the PROFIBUS User Organisation, Karlsruhe Germany.

FOUNDATION™ Fieldbus

Trademark of the Fieldbus Foundation, Austin, TX 78759, USA

HART®

Registered trademark of the HART Communication Foundation, Houston, USA

Microsoft®, Windows®, Windows 2000®, Windows XP®, Windows 2003 Server®, Windows 2008 Server®, Windows 7®, Windows Vista® and the Microsoft logo are registered trademarks of the Microsoft Corporation.

Acrobat Reader® is a registered trade mark of the Adobe Systems Incorporated.

All other brand and product names are trademarks or registered trademarks of the companies and organisations in question

1 Safety

1.1 Designated use

ControlCare is a field-based control system comprising hardware and software components. It can be used to visualize, monitor and control production processes. The approved usage of the individual units used in the system can be taken from the corresponding parts of the operating instructions.

ControlCare Application Designer allows the engineering, configuring and commissioning of a ControlCare SFC162 FOUNDATION Fieldbus or SFC173 PROFIBUS Field Controller as well as the programming of the hybrid function block in IEC 61131-3 language. This manual describes the Function Block Application Process, the Function Blocks that can be created with Application Designer and which are supplied with Endress+Hauser instruments as well as examples for their use. Workflows and practical examples for the engineering of FOUNDATION Fieldbus, PROFIBUS, Modbus projects as well as for the programming of hybrid function blocks are also to be found in the appropriate tutorial, see Chapter 1.5.

1.2 Installation, commissioning and operation

ControlCare Field Controllers have been designed to operate safely in accordance with current technical safety and EU directives. Essential to their use is the ControlCare Application Designer software suite, which allows control strategies to be created for FOUNDATION Fieldbus and PROFIBUS I/O applications. Field devices, links, junction boxes, cables and other hardware comprising the Fieldbus system must also be designed to operate safely in accordance with current technical safety and EU directives.

If devices are installed incorrectly or used for applications for which they are not intended, or if the controller is not configured correctly, it is possible that dangers may arise. For this reason, the system must be installed, connected, configured, operated and maintained according to the instructions in this and the associated manuals: personnel must be authorised and suitably qualified.

1.3 Operational safety

Location

Field Controllers must be mounted in a permanent and weather-protected location in a safe area. The environment shall be a metal cabinet or an installation frame with a well grounded mounting plane. The environment shall be protected.

Hazardous areas

The controller must be connected to networks operating in explosion hazardous areas via barriers or other safety components. When installing components in explosion hazardous areas:

- Ensure that all installation and maintenance personnel are suitably qualified
- Check that all equipment has the appropriate safety certificates
- Observe the specifications in the device certificates as well as national and local regulations.

This topic is discussed in BA013S (FF Guidelines) and BA034S (PROFIBUS Guidelines).

EMC

All modules are suitable for industrial use and conform with the following standard, see Appendix:

- EN 61326: 1997/A1: 1998
Interference emission: Class A apparatus
Interference immunity: as per Annex A, industrial environment

Depending upon the environment in which the bus is operating, particular attention should be paid to the grounding of the bus cables. This topic is discussed in BA013S (FF Guidelines) and BA034S (PROFIBUS Guidelines).

IP Address

A ControlCare Field Controller is normally configured from a workstation connected into the control system backbone. You will require a unique IP address to set it up.

It is recommended that ControlCare Field Controllers and OPC servers are not installed in an office network, as the large data packets exchanged between office equipment may lead to timeouts and intermittent communication errors. Ideally, the ControlCare system network should operate within its own IP domain; if this is not possible it should be separated from other parts of the network by a managed switch.



Warning

- The use of IP addresses is strictly controlled. Usually your system administrator will be authorised to allocate unique addresses. Assigning an unauthorised address to a Field Controller may result in conflicts within your system and the failure of the associated devices!

Since the system can be accessed and manipulated through the various ControlCare tools, it is advisable to control access both to the workstation and the folders in which the configuration is stored. Always make a back-up of the project.

Technical improvement

Endress+Hauser reserves the right to make technical improvements to its software and equipment at any time and without prior notification. Where such improvements have no effect on the operation of the equipment, they are not documented. If the improvements effect operation, a new version of the operating instructions is normally issued.

1.4 Conventions and icons

In order to highlight safety relevant or alternative operating procedures in the manual, the following conventions have been used, each indicated by a corresponding icon in the margin.

Safety conventions

Icon	Meaning
	A note highlights actions or procedures which, if not performed correctly, may indirectly affect operation or may lead to an instrument response which is not planned
	Caution! Caution highlights actions or procedures which, if not performed correctly, may lead to personal injury or incorrect functioning of the instrument
	Warning! A warning highlights actions or procedures which, if not performed correctly, will lead to personal injury, a safety hazard or destruction of the instrument

1.5 ControlCare documents

Table 1.1 indicates the documents, planned and realized, containing safety relevant information, installation, commissioning and operating instructions for the equipment and software associated with ControlCare.

All documentation available at the time of release is included on the ControlCare CD-ROM and is installed in **Start=>Programs=>Endress+Hauser=ControlCare=Manuals** during set-up.

Component	Description	Document type	Designation	Order No.
System	ControlCare System Overview	Operating manual	BA016S/04/en	56004883
	ControlCare System Design	Operating manual	BA039S/04/en	Planned
	ControlCare System Specifications	Operating manual	BA040S/04/en	56004888
Software	Application Designer Overview	Operating manual	BA017S/04/en	70104301
	Application Designer: Local I/O Tutorial	Operating manual	BA032S/04/en	71095009
	Application Designer: FF Tutorial	Operating manual	BA019S/04/en	70101151
	Application Designer: PROFIBUS Tutorial	Operating manual	BA036S/04/en	70101152
	Application Designer: MODBUS Tutorial	Operating manual	BA037S/04/en	70101153
	Application Designer: IEC 61131-3 Ladder Logic Tutorial	Operating manual	BA038S/04/en	70101386
	Application Designer: IEC 61131-3 Structured Text Tutorial	Operating manual	BA056S/04/en	71060063
	Field Control (OPC) Servers	Operating manual	BA018S/04/en	71031428
	SFC162 Visitor	Operation manual	BA069S/04/en	71113457
Field Controller	Hardware Installation Guide	Operating manual	BA021S/04/en	56004885
	Commissioning and Configuration	Operating manual	BA035S/04/en	56004887
Function Blocks	Function Block Manual	Operating manual	BA022S/04/en	56004886
Set-Up	Getting Started	Operating manual	BA020S/04/en	56004884
General	FOUNDATION Fieldbus Guidelines	Operating manual	BA013S/04/en	70100707
	PROFIBUS Guidelines	Operating manual	BA034S/04/en	56004242

Tab. 1-1: ControlCare documentation

1.6 About this manual

This manual is concerned with the function block model used in Foundation Fieldbus devices. It discusses the function block application process and then proceeds to describe the various blocks that are to be found in Endress+Hauser devices. It is structured as follows:

- Chapter 1: Safety
gives general information on the ControlCare platform
- Chapter 2: Introduction
briefly describes the FOUNDATION Fieldbus function block application process together with the associated objects, parameters and options
- Chapter 3: Resource block
describes the resource block and its parameters. Every device has a resource block that is structured in the same way
- Chapter 4: Transducer block
describes the temperature transducer and other transducer blocks associated with the SFC162 and SFC173 Field Controllers. Descriptions of the transducer blocks of devices offered by Endress+Hauser and other vendors are to be found in the appropriate device manuals.
- Chapter 5: Input blocks
describes the Analog Input block (AI), Multiple Analog Input block (MAI), Discrete Input block (DI) and Multiple Discrete Input block (MDI), with particular reference to their use in the ControlCare platform.
- Chapter 6: Control blocks
describes the PID, PI, I controller block and other control blocks
- Chapter 7: Arithmetic and logic blocks
describes the various arithmetic and logic blocks, e.g. Signal Characterizer (SC), Input Selector block (IS), Integrator or Totalizer (IT), Arithmetic (AR) etc.
- Chapter 8: Output blocks
describes the Analog Output block (AO), Multiple Analog Output block (MAI), Discrete Output block (DO) and Multiple Discrete Output block (MDO), with particular reference to their use in the ControlCare platform.
- Chapter 9: PROFIBUS blocks
describes the blocks used to integrate PROFIBUS devices into the ControlCare platform
- Chapter 10: Modbus blocks
describes the blocks used to integrate Modbus devices into the ControlCare platform
- Appendices
contain procedures and information on advance applications, e.g. interpretation of parameter status in SCADA systems

The manual aims to provide a balance between background information and the practical use of the blocks within ControlCare Application designer. Further information on their practical use can be found in the Control Application Designer Tutorials, see Chapter 1.5. A full description of the function block application process is to be found in the Fieldbus Foundation Specifications FF 891S and FF 892S. These form the basis of many of the parameter descriptions found here.

2 Function Block Application Process

2.1 Introduction

FOUNDATION Fieldbus function blocks are "black boxes" which transform input parameters into output parameters via parametric algorithms. They are executed repeatedly, either to a schedule or on the occurrence of an event. Once invoked, they run to completion.

The output of one function block can be linked to the input of another, see ControlCare Application Designer Tutorials, Chapter 1.5. Such linkages exist both within a device or between devices. Interfaces between function blocks located in the same device are locally defined. External links use the services of the function block shell, which provides access to FMS application layer services.

Function blocks are supported by resource and transducer blocks as well as by view, trend, alert and link objects as illustrated in Fig. 2.1.

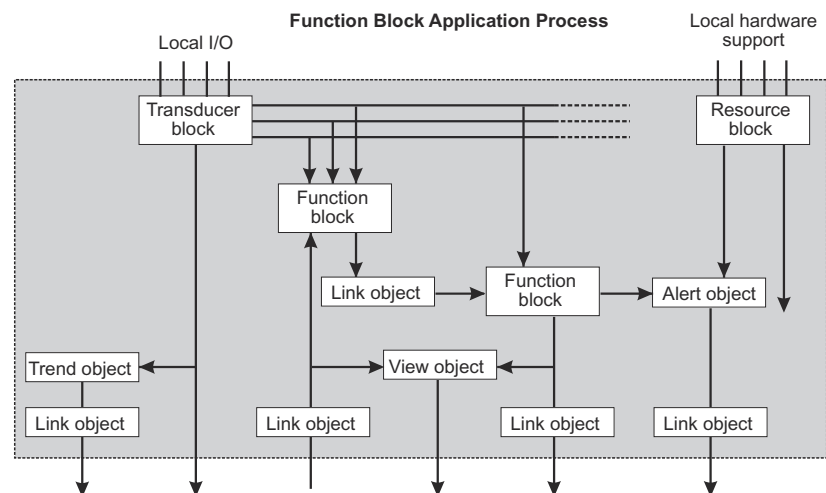


Fig. 2-1: Function block linkages (Fig 11, Foundation Fieldbus FF-800-1.3)

Figure 2-1 shows a general function block application process. The input parameter values to the transducer block and particular function blocks are collected from other blocks or, in the case of a transmitter, the sensor electronics. When the process is invoked, a snapshot is made of the momentary values, which are then used to execute the block. Any changes in input values during the execution are ignored.

Once the process has started, the algorithm takes the input values and generates outputs as it progresses. The execution is controlled by so-called contained parameters, which are not visible as input and output parameters, but which may be accessed and modified remotely as specified by the function block.

The operation of the algorithm may be affected by input events. The receipt of events during execution is regulated by a control function, which also generates corresponding output events.

On completion of the algorithm, the block data is saved for use on the next invocation and a snapshot is made of the output data. This is then released simultaneously for use in other function blocks.

2.2 Block objects

The various objects within the function block application process have the function described in Table 2-1 below, see also Fig. 2.1:

Object	Description
Resource blocks	<p>Only one resource block is defined for each fieldbus device. It contains the associated hardware-specific characteristics. These comprise a set of contained parameters but no input or output parameters.</p> <p>A resource block contains an algorithm that is used to control and monitor the operation of the device hardware. Its execution is dependent upon the characteristics of the physical device, as defined by the manufacturer. The algorithm might cause events to be generated during its execution.</p>
Transducer blocks	<p>Transducer blocks decouple local input/output functions from the function blocks. They read from sensor hardware and write to actuator hardware. This allows the transducer block to be executed as frequently as is necessary to obtain good data from the sensor or to ensure valid writes to the actuators without burdening the function block(s) to which they are linked. By having standardised signal as output, they also isolate the function block from vendor-specific characteristics of the sensor/actuator.</p>
Function blocks	<p>Function blocks comprise a complete set of parameters, an algorithm and events. Unlike resource and transducer blocks, function blocks are executed to a schedule or on the occurrence of an event. The algorithms perform input, output, calculation, logic and control functions for the application system. In the case of input and output functions, they are linked to the transducer blocks of actuators and sensors respectively, which are in turn connected to the actuator or sensor hardware.</p>
View and trend objects	<p>View objects aid the management and control of function blocks by making the configuration and operational parameters visible to the user. They allow operator data to be grouped together for efficient transfer and processing.</p> <p>Trend objects provide history information which allows the behaviour of a function block to be examined. They allow short-term history data can be collected and saved within the device.</p>
Alert objects	<p>Alert objects generate messages when alarms and events are detected. They report these by sending an event notification and waiting a specified period of time for an acknowledgement. This process is completed, even when the condition that triggered the alert no longer exists. If no acknowledgement is received within a predefined time-out, the event notification will be retransmitted. This ensures that the alert message is not lost.</p> <p>The function block model recognises two types of alert: events and alarms. Events are used to report a change of status of a function block, for example when a parameter crosses a particular threshold. Alarms are used to report not only the status change, but also the return to original status.</p>
Link objects	<p>Link objects contain information regarding the connecting together of the inputs and outputs of function blocks. They are also used to define external access to trend, view and alert objects. They identify the parameter or object to be linked, the field message service (FMS) used to transfer the data and the virtual communications relationship (VCR) used for transfer. In the case of input/output parameter links, the remote parameter is also identified.</p>

Tab. 2-1: Function block application process objects

2.3 Block parameters

2.3.1 Parameter types

Resource blocks, transducer blocks and function blocks all contain parameters. They can be classified according to two criteria:

- How they are generated, accessed and stored
- How they are used by the function block application process

Every parameter is characterized by one type attribute from each of these classes.

Generation, access and storage

The type attributes associated with generation, access and storage are listed in Table 2-2 below:

Parameter type	Description
Dynamic	Dynamic parameters are normally read only. Their value is calculated by the block algorithm, e.g. the output value of a function block. If there is a power failure, they are recalculated.
Static	Static parameters are normally read-write and have a specific configured value that seldom changes. If there is a power failure, they are restored by the device in which they reside. An interface or temporary device may write to them on an infrequent basis. All parameter changes are tracked by incrementing the static revision parameter ST_REV of the associated block, see Chapter 2.3.4, and generate an update event.
Non-volatile	Non-volatile parameters are normally read-write and are frequently overwritten. If there is a power failure, the last saved value must be restored by the device. Normally changes in the value are not tracked.

Tab. 2-2: Parameter type attributes associated with generation and storage

Usage

The type attributes associated with usage listed in Table 2-3 below:

Parameter type	Description
Input	Input parameters obtain their value from outside the block, normally through a link to an output parameter of another function block. The block algorithm will normally transform or operate on the input parameter to produce the block's output value. Most blocks use a single input value in their algorithm, the so-called primary input value. The execution may be supported by other so-called secondary input values, see e.g. PID Control Block, Chapter 6.1.
Contained	Contained parameters may be set by an operator, a higher level device, or calculated. They are parameters that reside within the block and cannot be linked to another function block input or output. They may be viewed, however, by an external application. An example of a contained parameter common to all blocks is the block mode parameter MODE_BLK.
Output	Output parameters are normally generated by the block algorithm. Their value may be dependent on the value of the mode parameter of the block. The primary output parameter of a block will normally be linked to an input parameter of another function block. Some blocks also contain secondary output parameters such as alarm and event parameters that play a supporting role to the primary output parameter.

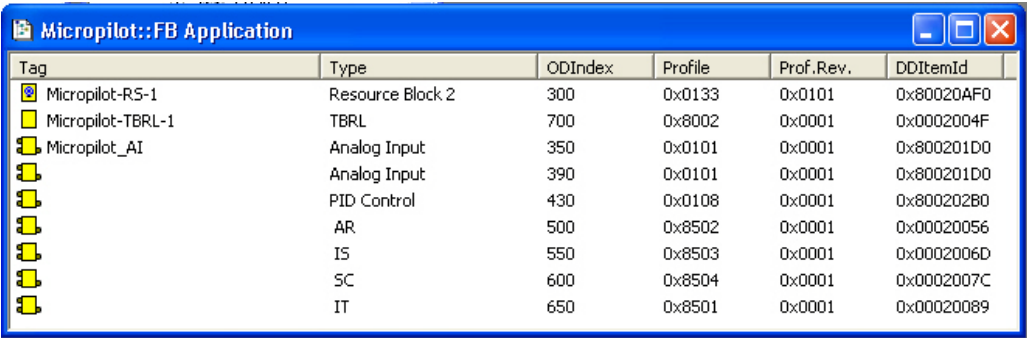
Tab. 2-3: Parameter type attributes associated with usage

2.3.2 Parameter identifiers

Details of the formal model for parameter identification can be taken from Foundation Fieldbus Specification FF-890, Function Block Application Process, Part 1. Expressed briefly, the host application must be able to find any parameter in the system by means of a unique identifier. This is composed of two elements, which are put together to form the construct Tag.Parameter

- Tag
is the block tag assigned to the function block by the host application
- Parameter
is the DD member ID of the parameter and is unique to the block.

Using the Tag.Parameter construct, the host application is able to access the parameter value or its description, its data type index and the parameter semantics. In the case of Application Designer, a view of this information can be obtained by clicking on the **FB VFD** node of the **Control Module** tree, when the associated device is on line, see Application Designer Tutorials. In this manual, the data type etc. is also to be found in the parameter tables that accompany every function block description.



Tag	Type	ODIndex	Profile	Prof.Rev.	DDItemId
Micropilot-RS-1	Resource Block 2	300	0x0133	0x0101	0x80020AF0
Micropilot-TBRL-1	TBRL	700	0x8002	0x0001	0x0002004F
Micropilot_AI	Analog Input	350	0x0101	0x0001	0x800201D0
	Analog Input	390	0x0101	0x0001	0x800201D0
	PID Control	430	0x0108	0x0001	0x800202B0
	AR	500	0x8502	0x0001	0x00020056
	IS	550	0x8503	0x0001	0x0002006D
	SC	600	0x8504	0x0001	0x0002007C
	IT	650	0x8501	0x0001	0x00020089

Fig. 2-2: DD member ID information for Micropilot viewed in the Block List dialog

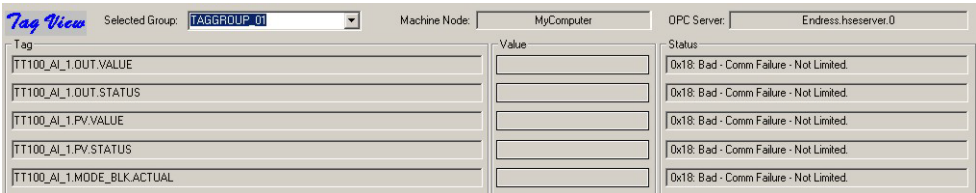
Export to OPC Server

Application Designer presents parameter information in easily readable form by using the parameter names as can be viewed in the Off Line and On Line Characterization dialogs, see Chapter 2.3.4. These are then identified using the same construct. Thus, for instance, the process value PV value of an Analog Input block with the tag "TT100_AI_1.PV" can be expanded to show its two elements Value and Status. The two parameters can thus be identified from the Tag.Parameter construct as:

TT100_AI_1.PV.Value and
TT100_AI_1.PV.Status

whereby the block tag is generated by default and can be influenced by the Preference settings in Application Designer, see Application Designer Tutorials.

Fig. 2-4 shows an example of export to an OPC Server with tags taken from the control strategy shown in Fig. 2-6. The project was off-line, hence the bad quality status.



Tag	Value	Status
TT100_AI_1.OUT.VALUE		0x18: Bad - Comm Failure - Not Limited.
TT100_AI_1.OUT.STATUS		0x18: Bad - Comm Failure - Not Limited.
TT100_AI_1.PV.VALUE		0x18: Bad - Comm Failure - Not Limited.
TT100_AI_1.PV.STATUS		0x18: Bad - Comm Failure - Not Limited.
TT100_AI_1.MODE_BLK.ACTUAL		0x18: Bad - Comm Failure - Not Limited.

Fig. 2-3: Example of the Tag.Parameter construct in the Tag View program supplied with ControlCare

Aliases

From ControlCare Version 2.04.xx onwards, it is possible to assign alias names to the parameter identifiers generated by the system. These are also exported to the OPC Server. Should a project made with Version 2.03.xx or less be updated by adding alias names, it is not necessary to update any SCADA application, as the parameter identifiers will still be found.

2.3.3 Universal parameters

Resource blocks, transducer blocks and function blocks all operate in the same manner. To this end, there are a number of parameters that are common to all. These appear in the order below and are used for the following purposes:

Parameter	Description
ST_REV	Indicates the revision level of the block's static parameters and may be used in configuration management
TAG_DESC	Allows the entry of a block description (up to 32 characters) which may be used in a human interface or in block documentation to clarify the block application
STRATEGY	Allows the entry of a user assigned value that may be used in configuration or diagnostics as a key in sorting block information
ALERT_KEY	Allows the entry of a user assigned value that may be used in sorting alarms or events generated by a block
MODE_BLK	Determines the block operating mode and available modes for a block instance
BLOCK_ERR	Indicates the error status of hardware and software components associated with the block

Tab. 2-4: Universal parameters found in all function blocks

The parameters **MODE_BLK** and **BLOCK_ERR** are described in Chapter 2.4 and Chapter 2.8 respectively.

ST_REV

As described in Section 2.3.1, each block contains dynamic data, which change during the execution of the block and static data which normally do not. Since static data are seldom changed, a means is provided to avoid reading them more than once, unless they change. This is the static revision parameter **ST_REV**, which is incremented every time a static parameter attribute value is changed. It may also be incremented if a static parameter attribute is overwritten but the value is not changed.

Devices using the static data of another device receive a copy of **ST_REV** when either the static or dynamic data are read. As long as two successive copies match, the static data has not changed. Should the **ST_REV** of a device change, the **UPDATE_EVT** parameter is output to alert other devices that one or more changes have occurred. A time stamp, relative parameter index and associated block index of the last parameter changed are included in the alert, along with the new value of **ST_REV**. If **ST_REV** has incremented by more than 1, the interface device will update all static data.

In order that downloads do not generate a burst of update alerts, these are not generated when a block is out of service. If a parameter is changed when the block is out of service, **ST_REV** is incremented as normal. This triggers an **UPDATE_EVT** alert when the block returns from OOS to an operational mode.

2.3.4 Parameters in Application Designer

Control strategy and navigation tree

Fig 2-4 shows a screenshot of a Process Cell navigation tree and the associated control strategy.

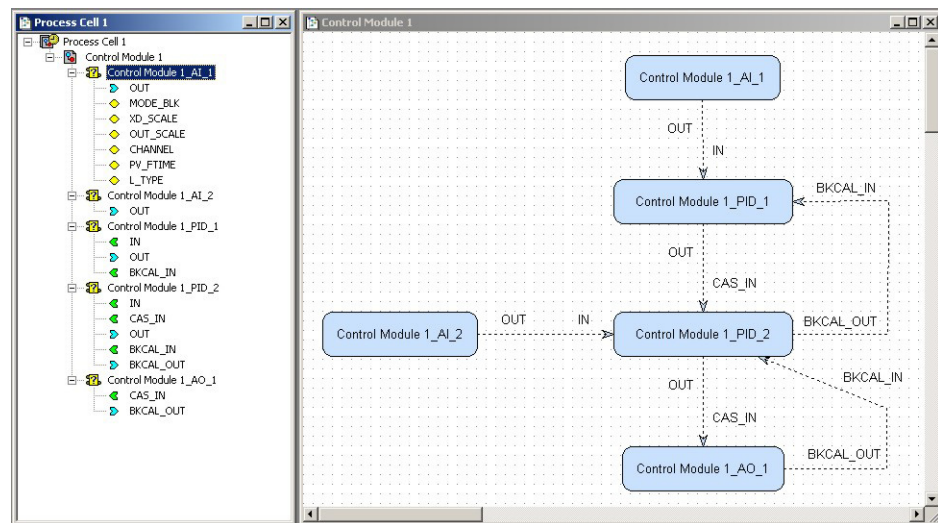




Fig. 2-4: Process Cell and cascade control strategy created off-line in ControlCare Application Designer

The control strategy is as created from the strategy window, i.e. the function blocks have yet to be assigned to the fieldbus devices (also indicated in the navigation tree by the "?" in the function block icon). When used in this way, Application Designer temporarily assigns a block tag that is named after the Control Module and the block tag, e.g. Control Module 1_PID_1. As the links are made between the function blocks, the input and output parameters, indicated by the icons  and , are added to the navigation tree. The arrows in the control strategy are in fact visualizations of the link objects, and the navigation tree a visualization of the tag.parameter construct.

Depending on the preferences set for tag naming, the tag name will change to that of the device when the function block is assigned to a device. Thus if the function block Control Module 1_PID_1 is assigned to a device with tag FC102, the tag name is automatically changed to FC102_PID_1, see Fig. 2-5.

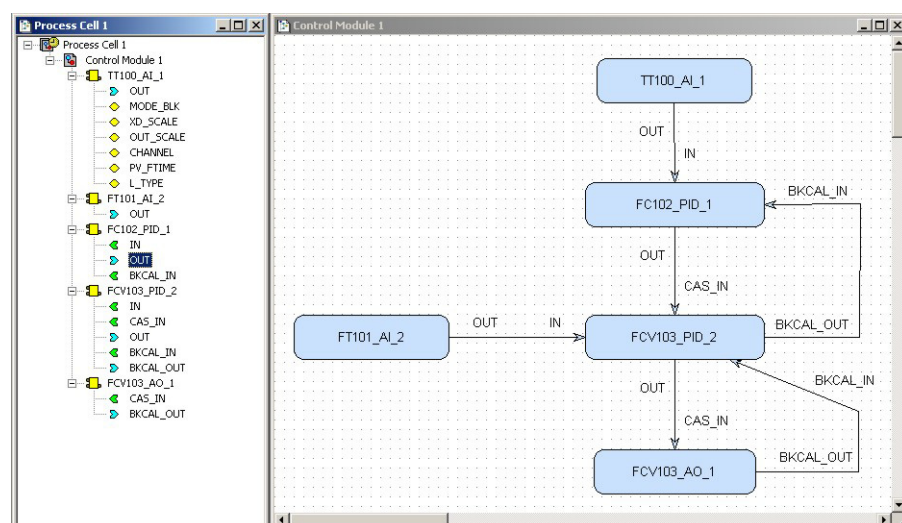



Fig. 2-5: Process Cell and cascade control strategy after assignment of function blocks to the devices

The parameters with the icon  are contained parameters that were automatically set when the device was created, or have been configured in the Off Line or On Line Characterization dialog.

Off Line Characterization dialog

When Application Designer is offline, a double-click on the Function Block leaf of the Control Module or Fieldbus navigation tree opens the corresponding Off Line Characterization dialog, see also Application Designer Tutorials.

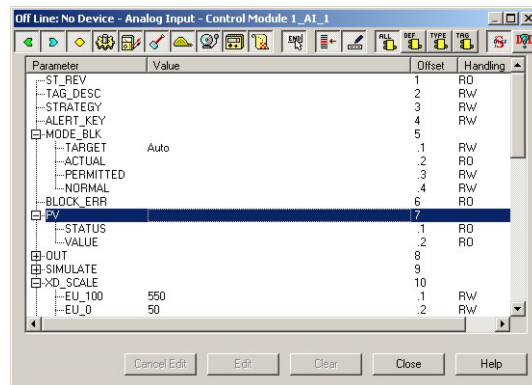


Fig. 2-6: Example of OffLine Characterization dialog

The dialog shows the parameters associated with the block in the dialog header. These are arranged in a tree, and depending upon data type, they can be expanded to show their elements. For example, the PV parameter expands to show the two elements PV.VALUE and PV.STATUS. The dialog also shows the value, the offset within the block and whether the parameter is read-write (RW) or Read Only (RO).

The row of buttons at the top of the dialog calls up several different views of the parameters (view objects). In addition to the input, output and contained parameters, views can be had of the operational, diagnostic, alarms, tuning parameters, etc., see Application Designer Overview, BA017S/04/en for more details. It is also possible to build customized views of parameters for configuration or online monitoring.

On Line Characterization dialog

When Application Designer is online, a double-click on the Function Block leaf of the Control Module or Fieldbus navigation tree opens the corresponding On Line Characterization dialog. Now the Read Only parameters will contain parameters read from the associated field devices.

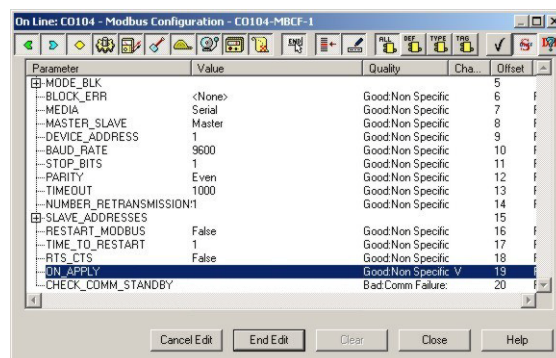


Fig. 2-7: Example of On Line Characterization dialog with parameter quality

Expand the PV parameter and observe the status. Now expand the MODE_BLK parameter and set the Target to OOS (out of service). To do this Press **Edit**, click in the **Value** space, select OOS from the drop-down menu, then click **End Edit**. Note that the PV status changes from Good to Bad. Toggle between status codes and texts with the button Put the block target back into Auto, and note that the quality changes from bad to good again.

The execution of a block involves the inputs, outputs, contained parameters, and the algorithm of the block. The execution time for a block algorithm is defined as a parameter of the block. Its value is dependent on how the block was implemented.

2.4 Block operation

2.4.1 Block mode parameter `MODE_BLK`

The **MODE_BLK** parameter controls the operational mode of the block. It has four elements with the functions summarized in Table 2-5.

Element	Description
Target mode	Sets the mode in which the block should normally operate. Most blocks have the target mode Auto, exceptions being output and cascaded PID blocks which normally operate in Cas mode. In addition, when parameters are changed on-line, the Target mode must normally be changed to OOS (out of service) before the change can be made.
Actual mode	Indicates the current mode of operation of the block. This may differ from the target mode, because under certain operating conditions, a block may not be able to reach the target mode, e.g. because it is not configured correctly or not receiving the correct information from an upstream or downstream block linked to it.
Permitted mode	Lists the modes in which a block is able to operate. The modes supported vary according to the type and function of a block.
Normal mode	The normal mode indicates the mode in which the block normally operates. This parameter is not supported by all devices.

Tab. 2-5: *MODE_BLK* parameter elements

The user sets the target mode when the block is configured. When it goes on-line, the algorithm checks if the block can be executed in the requested mode. If this is not possible, the mode with the priority nearest to the target mode is selected, see Chapter 2.4.3. The actual mode displays the current mode of block operation.

Retained target mode

When the target mode is OOS, Man, RCas or ROut, see Chapter 2.4.2, the target mode attribute may retain information about the previous target mode. This information may be used by the block in mode shedding and setpoint tracking, see Chapter 2.4.4.

2.4.2 Block modes

Resource blocks, transducer blocks or function blocks may support one or more of the modes listed in Table 2-6 below. They may be operated only in the modes they support.

Block Mode	Description
Out of service (OOS)	In this mode, the block is not executed. The output is maintained at last value or, in the case of output blocks, the output value may be maintained at an assigned fault state value. For control blocks, the setpoint is maintained at last value, see Chapter 2.4.4.
Initialization manual (Iman)	This mode cannot be requested through the target mode. Blocks connected to a downstream block by a back calculation link (BKCAL_OUT/BKCAL_IN) are forced into it when the status of the parameter indicates that the downstream block is not in cascade (Cas mode). The normal algorithm is not executed, and the output tracks the (BKCAL_IN) parameter. The setpoint may be maintained or initialized to the value of the process variable parameter.
Local override (LO)	This mode is to be found in control and output blocks that support a track input parameter (TRK_VAL). Tracking is enabled in the Control Options parameter group, see Chapter 2.8.2, and is initiated by the TRK_IN_D discrete parameter. For some devices a local lock-out switch must be used to enable Local Override. In the local override mode, the block output tracks the value of the track input parameter (TRK_VAL). The algorithm initializes so that no bump is experienced when the mode switches from LO back to the target mode. The setpoint may be maintained or initialized to the process variable parameter value.
Manual (Man)	In this mode, the block output is not calculated, but set by the operator. It may be limited. The algorithm initializes so that no bump is experienced when the mode switches. Any setpoint may be maintained or initialized to the value of the process variable parameter or to the setpoint value associated with the previous (retained) target mode.

Block Mode	Description
Automatic (Auto)	This is the normal operating mode of the majority of blocks. For control blocks operating in this mode, a local setpoint value SP is used by the normal block algorithm to determine the primary output value. The local setpoint value may be overwritten on-line – without changing the target mode to OOS – by a user using Application Designer or a SCADA program such as ControlCare P View.
Cascade (Cas)	This is the normal operating mode of a cascaded control block or output block. In this mode, the setpoint value used in the block algorithm is the cascade input value (CAS_IN) provided by an upstream function block. This and the BKCAL_OUT parameter are used to establish a bumpless cascade with the upstream block. Note: the target mode of PROFIBUS output blocks integrated via the SPC173 Field Controller is Auto.
Remote cascade (RCas)	In this mode, the setpoint value used in the block algorithm is the remote cascade input parameter (RCAS_IN) provided by an external control application. A remote cascade output parameter (RCAS_OUT) is maintained by the block to support initialization of the control application when the block mode is not remote cascade. The control application acts as a normal upstream block. Its algorithm is not synchronized to the function block schedule, however, and it does not use a link to transfer the setpoint to the block.
Remote output (ROut)	In this mode, the block output is not calculated but set to the value of the remote-output input parameter (ROUT_IN) provided by an external control application. A remote-output output parameter (ROUT_OUT) is maintained by the block to support initialization of the control application when the block output is not provided remotely. The control application acts as a normal upstream block. Its algorithm is not synchronized to the function block schedule, however, and it does not use a link to transfer the setpoint to the block.

Tab. 2-6: Possible operation modes for function blocks

Table 2-7 summarizes the source of the setpoint value SP and the output value OUT for the various modes.

Block mode	Source of SP	Source of OUT
OOS	User	User
Iman	User	Other function block – following BKCAL_IN parameter
LO	PID/EPID/APID: User AO/DO: Fault state (last value or FSTATE_VAL)	PID/EPID/APID: Other function block following TRK_VAL parameter AO/DO: Fault state (last value or FSTATE_VAL)
Man	User	User
Auto	User (for PROFIBUS AO, PID)	Block algorithm
Cas	Other function block – following CAS_IN parameter	Block algorithm
Rcas	Control application running on an interface device	Block algorithm
Rout	Block keeps last value	Control Application running on an interface device

Tab. 2-7: Source of SP and OUT as a function of block mode

2.4.3 Mode priority

Each mode has an associated priority, see Table 2-8. Priority is used when the block calculates the actual mode or checks whether write access is allowed for a particular mode or another of higher priority. To calculate the actual mode, the block checks whether certain conditions are present which will prevent a change to the mode with the next lower priority.

Mode	Description	Priority
OOS	Out of Service	7 - highest
IMan	Initialization Manual	6
LO	Local Override	5
Man	Manual	4
Auto	Automatic	3
Cas	Cascade	2
Rcas	Remote Cascade	1
Rout	Remote Output	0 - lowest

Tab. 2-8: Block modes with corresponding priorities

When a block goes online, the actual mode is set to OOS. The block now checks whether it may change to the IMan mode. This is the case when all block parameters are valid and the target mode is not set to OOS, see Table 2-10. This procedure is repeated through all the modes supported by the block until the target mode is achieved. For instance, if the target mode is Cas, the conditions for OOS, IMan, LO, Man and Auto must be checked in this order. If all conditions are false, the actual mode will be the same as target mode.

Table 2-9 summarizes the conditions that force a block to operate in a particular mode when they are true.

Block Mode	Conditions
OOS	<ul style="list-style-type: none"> Resource block is in OOS (resource state is Standby) Enumerated parameter has an invalid value
IMan	<ul style="list-style-type: none"> BKCAL_IN.status is Bad BKCAL_IN.status is Good - Fault State Active, Not Invited or Initialization Request.
LO	<ul style="list-style-type: none"> Fault state is active (in an output function block) CONTROL_OPTS.Track Enable active and TRK_IN_D is active. If target is Man then the CONTROL_OPTS.Track in Manual must be active
Man	<ul style="list-style-type: none"> Target mode has just been changed manually from OOS Status attribute of primary input parameter (IN parameter) is Bad or Uncertain with option to treat Uncertain as Bad and Bypass not set. Target mode is RCas or ROut and SHED_OPT=shed to Manual or shed to next
Auto	<ul style="list-style-type: none"> Target mode is Cas and CAS_IN.status=Bad or cascade initialization not completed Target mode is RCas and RCAS_IN.status=Bad and SHED_OPT=shed to Auto or shed to next Target mode is ROut and ROUT_IN.status=Bad and SHED_OPT = shed to Auto or shed to next.
Cas	<ul style="list-style-type: none"> Actual mode last execution was Cas. Target mode is Cas and cascade initialization has just completed Target mode is RCas and RCAS_IN.status=Bad and SHED_OPT=shed to next and cascade initialization has just completed Target mode is ROut and ROUT_IN.status=Bad and SHED_OPT=shed to next and cascade initialization has just completed
RCas	<ul style="list-style-type: none"> RCas cascade initialization has just completed or actual mode last execution was RCas.
ROut	<ul style="list-style-type: none"> ROut cascade initialization has just completed or actual mode last execution was ROut.

Tab. 2-9: Conditions governing Actual Mode calculation

Trouble-shooting

If a function block does not run in the desired target mode, Table 2-10 can be used to analyze the possible reason.

Target Mode	Symptom, cause and remedy
OOS	Block remains in OOS <ul style="list-style-type: none"> ■ Resource block target mode is OOS <ul style="list-style-type: none"> – Set the Resource block target mode to Auto ■ Bad status for enumerated parameters <ul style="list-style-type: none"> – Check that block has been configured correctly
Iman	Block remains in Iman <ul style="list-style-type: none"> ■ BKCAL_IN.status is Bad <ul style="list-style-type: none"> – Link failure in backward path BKCAL_OUT/ BKCAL_IN ■ BKCAL_IN.status is NotInitiated <ul style="list-style-type: none"> – Downstream block can not execute in Cas <ul style="list-style-type: none"> Check the target mode of downstream block as well fault state condition – Link failure in forward path OUT/CAS_IN
LO	Target mode LO cannot be attained <ul style="list-style-type: none"> ■ CONTROL_OPTS setting wrong <ul style="list-style-type: none"> – Check if setting is Track Enable or Track in Manual ■ Value and status of TRK_IN_D wrong <ul style="list-style-type: none"> – Check link, condition for activation, upstream block configuration ■ Value and status of TRK_VAL bad <ul style="list-style-type: none"> – Check parameter settings ■ CAS_IN status of downstream block bad <ul style="list-style-type: none"> – Check if CAS_IN and the delay time for fault state are being forced by FSTATE_TIME
Man	Block does not reach target mode Cas <ul style="list-style-type: none"> ■ IN status bad <ul style="list-style-type: none"> – Link failure in backward path BKCAL_OUT/ BKCAL_IN – Link failure in forward path OUT/CAS_IN Block does not reach target mode Rcas or Rout <ul style="list-style-type: none"> ■ Block not synchronized <ul style="list-style-type: none"> – Check that the update rate of RCAS_IN and ROUT_IN of the control application corresponds to the values in SHED_RCAS and SHED_ROUT respectively ■ SHED_OPT parameter set Shed to Manual, no return; remote communication has failed <ul style="list-style-type: none"> – When communication reestablished, reset target mode to RCas or ROut
Auto	Block does not reach Auto <ul style="list-style-type: none"> ■ Bad status for enumerated parameters <ul style="list-style-type: none"> – Check that block has been configured correctly – Check all parameters required for the block algorithm have been set Block does not reach target mode Cas <ul style="list-style-type: none"> ■ IN status bad <ul style="list-style-type: none"> – Link failure in backward path BKCAL_OUT/ BKCAL_IN – Link failure in forward path OUT/CAS_IN Block does not reach target mode Rcas or Rout <ul style="list-style-type: none"> ■ Block not synchronized <ul style="list-style-type: none"> – Check that the update rate of RCAS_IN and ROUT_IN of the control application corresponds to the values in SHED_RCAS and SHED_ROUT respectively ■ SHED_OPT parameter set Shed to Auto, no return; remote communication has failed <ul style="list-style-type: none"> – When communication reestablished, reset target mode to RCas or ROut
Cas	Block does not reach target mode Rcas or Rout <ul style="list-style-type: none"> ■ Block not synchronized <ul style="list-style-type: none"> – Check that the update rate of RCAS_IN and ROUT_IN of the control application corresponds to the values in SHED_RCAS and SHED_ROUT respectively

Tab. 2-10: Trouble-shooting table for block target mode

2.4.4 Mode shedding

Mode shedding is a mechanism that allows the function block application process to interface with external, proprietary control applications running e.g. on a host computer, distributed control system controller (DCS) or programmable logic controller (PLC). Such applications are allowed to adjust the setpoint value SP of a function block in RCas mode and/or its primary output value **OUT** in ROut mode. When they do, they provide the application with the value of each parameter together with its status.

In order to ensure bumpless transfer in the event of an interface failure or a breakdown in communication, the function block involved monitors the traffic between the external application and itself. If a new value is not received within a specified "update time", determined by the parameters **SHED_RCAS** and **SHED_ROUT** in the device resource block, or a bad status is received, then the function block mode is changed to a non-remote mode of higher priority.

SHED_OPT

The **SHED_OPT** parameter is to be found in control and output blocks and configures the desired behavior when a remote mode, i.e. Rcas and Rout is shed. It also determines whether the shed mode is to be maintained once the **RCAS_IN** or **ROUT_IN** parameter updating is recovered. Table 2-11 summarizes the options

Index	Meaning
0	Uninitialized - Invalid
1	Normal shed, Normal return – Actual mode changes to the next lowest priority non-remote mode permitted but returns to the target remote mode when the remote computer completes the initialization handshake.
2	Normal shed, No return – Target mode changes to the next lowest priority non-remote mode permitted. The target remote mode is lost so there is no automatic return to it.
3	Shed to Auto, Normal return – Target mode changes to Auto but returns to the target remote mode when the remote computer completes the initialization handshake.
4	Shed to Auto, No return – Target mode changes to Auto. The target remote mode is lost so there is no automatic return to it.
5	Shed to Manual, Normal return – Target mode changes Man but returns to the target remote mode when the remote computer completes the initialization handshake.
6	Shed to Manual, No return – Target mode changes to Man. The target remote mode is lost so there is no automatic return to it.
7	Shed to Retained target, Normal return – Target mode changes to retained target but returns to the target remote mode when the remote computer completes the initialization handshake.
8	Shed to Retained target, No return – Target mode changes to retained target. The target remote mode is lost so there is no automatic return to it.

Tab. 2-11: Mode shedding options offered by Application Designer in RCas and Rout modes

2.5 Parameter calculation

2.5.1 Process variable PV

The process variable (PV) parameter of a block reflects the value and status of the primary input value or calculated value based on a multiple input. Depending on the function block it is characterized as follows:

- In the PID control and analog alarm (AALM) blocks, PV is the IN parameter after filtering
- In the analog input (AI) and output (AO) blocks, PV reflects the value from the transducer or transducer readback respectively, after filtering
- In the Arithmetic block (ARTH), PV is the combination of two input parameters for range extension.

Although it is a contained parameter, PV has a status. This is a copy of the primary input status or, when PV is based on multiple inputs, the worst encountered status. The PV value reflects the value of the calculated input regardless the mode of the block, unless its input is not usable. In this case, PV holds the last usable value.

Linearization

The parameter **L_TYPE** allows one of the following linearisation types to be selected:

Direct	<p>With this setting the measured value from the transducer block (input value) avoids the linearisation function and is looped unchanged with the same unit through the block.</p> $PV = \text{Input value}$ <p>The scaling and unit of XD_SCALE and OUT_SCALE must be the identical, otherwise the block will switch to OOS.</p>
Indirect	<p>With this setting the measured value from the transducer block (input value) is re-scaled linearly via the input scaling XD_SCALE to the desired output range OUT_SCALE</p> $PV = \frac{X}{100} \cdot (Y - Z) - Z$ <p>where X = FIELD_VAL, Y = OUTSCALE_100%, Z = OUTSCALE_0%</p>
Indirect square root	<p>With this setting the measured value from the transducer block (input value) is re-scaled via the parameter group XD_SCALE and recalculated using an evolution function. It is then rescaled again to the desired output range via the parameter group OUT_SCALE..</p> $PV = \sqrt{\frac{X}{100}} \cdot (Y - Z) - Z$ <p>where X = FIELD_VAL, Y = OUTSCALE_100%, Z = OUTSCALE_0%</p>

Time constant

A damping filter may be applied in the PV signal through the time constant parameter PV_FTIME. This represents the time in seconds PV requires to reach 63.2% of the final value after a step change to the input. If the PV_FTIME value is zero, the filter is disabled.

2.5.2 Scaling parameters XD_SCALE, PV_SCALE, OUT_SCALE

A control system must deal with a number of different signals, e.g. 4 mA to 20 mA, relays, pulse, fieldbus etc. as well as a variety of engineering units. The scaling parameters are used to convert from one set of units to another. There are two main applications:

- Display devices require the range and engineering units for bar graphs and trending
- Control blocks require the range as percent of span, so that the tuning constants can remain dimensionless.

Scaling algorithm

There are three main scaling parameters which operate back-to-back across a function block, **XD_SCALE**, **PV_SCALE** and **OUT_SCALE**. Normalized parameters are used between the two scalings, which is reflected in a number of contained parameters that can be viewed only in % of scale, see Fig. 2-8. Each scaling parameter has four elements:

- EU_100: Upper range-end value in engineering units
- EU_0: Lower range-end value in engineering units
- UNITS_INDEX: Engineering units of range-end values
- DECIMAL: Number of figures after the decimal point (default = 2)

The scaled value is obtained from the formulae:

- Conversion to % of scale: $VALUE\% = (VALUE - EU_0) * 100 / (EU_100 - EU_0)$
- Conversion to engineering units: $VALUE = [VALUE\% * (EU_100 - EU_0) / 100] + EU_0$

By default, the EU_0 and EU_100 parameters are set to 0 and 100 respectively.

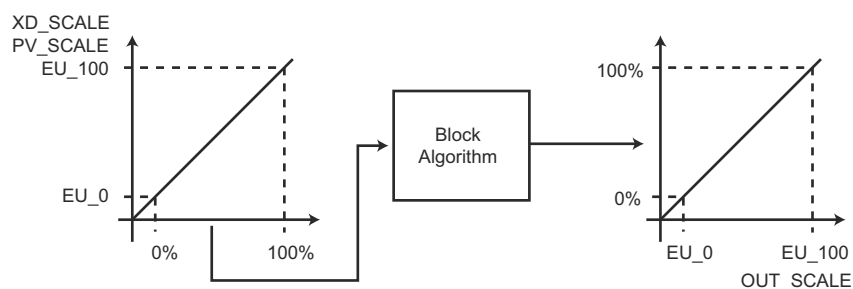


Fig. 2-8: Scaling of block parameters

Table 2-12 summarizes the typical use of the scaling parameters in the blocks in which they appear.

Block	XD_SCALE	PV_SCALE	OUT_SCALE
Analog Input (AI)	Scales transducer output range	n/a	Scales to engineering range
PIDcontrol	n/a	Engineering range to % of scale	% of scale to engineering range
Analog Output (AO)	Scales to transducer input range	Scales to engineering range	n/a

Tab. 2-12: Typical scaling parameter usage in different function blocks

OUT_RANGE parameters

A number of blocks offer an **OUT_RANGE** or similar range parameter which scales the associated value for output to a HMI. The ranging proceeds in exactly the same way as for the SCALE parameters.

Example using scaling parameters

The temperature of the liquid in a small feed tank is to be kept constant at around 60°C. The tank is heated by a heating element with a 4 mA – 20 mA control input and which is capable of heating the tank to 120°C. The temperature is measured by a 4 mA – 20 mA transmitter with a measuring range of -50°C to +150°C. A simple AI – PID – AO loop is used to control the input signal to the heater element controller.

Analog Input Block

The 4 mA to 20 mA input signal is converted to a % scale via the XD_SCALE parameter:

EU_100 = 20; EU_0 = 4; UNITS_INDEX = mA and DECIMAL = 2 (default)

The temperature signal in °C is obtained from the OUT_SCALE parameter:

EU_100 = 150; EU_0 = -50; UNITS_INDEX = °C and DECIMAL = 2 (default)

PID Control Block

The setpoint SP and the IN value are in °C, but PID algorithm works internally with values in percent of span. The PID block therefore converts the error to percentage (PV_SCALE), it calculates the output in percentage, and then it converts to engineering unit of output (OUT_SCALE).

The -50°C to 150°C input signal is converted to % of scale with the PV_SCALE parameter:

EU_100 = 150; EU_0 = -50; UNITS_INDEX = °C and DECIMAL = 2 (default)

If setpoint SP = 60°C and IN = 50°C, the scaled values are, see also previous page,

SP% = $(60+50) \cdot 100 / (150+50) = 55.00\%$

PV% = $(50+50) \cdot 100 / (150+50) = 50.00\%$

The PID algorithm calculates the error in percent. If reverse action is configured, the error is the difference between SP% and PV%.

Error% = SP% - PV % = 5.00%

The PID algorithm applies the Error% to the calculation of the P, I and D terms. If, for the sake of simplicity, only the proportional term is enabled, the value of the output is:

GAIN = 2.0; RESET = +INF; RATE = 0.0; OUT% = 10.00%

The output value could be converted from percentage to engineering units with the OUT_SCALE parameter, however, in order to illustrate scaling in the Analog Output block, in the example the output is set to %:

EU_100 = 100; EU_0 = 0; UNITS_INDEX = % and DECIMAL = 2 (default)

OUT = 10%

Analog Output Block

The heating coil controller requires a 4 – 20 mA signal at its input, which is done though the two scaling parameters PV_SCALE which converts the CAS_IN value to 0% – 100% and the XD_SCALE parameter which converts from 0% – 100% to the signal range required by the transducer, in our case, 4 mA – 20 mA.

For PV_SCALE: EU_100 = 100; EU_0 = 0; UNITS_INDEX = % and DECIMAL = 2 (default)

For XD_SCALE: EU_100 = 20; EU_0 = 4; UNITS_INDEX = mA and DECIMAL = 2 (default)

The CAS_IN signal of 10% is unchanged after PV scaling. The OUT value and the value at the transducer channel is:

$10 \cdot (16 \text{ mA} - 4 \text{ mA}) / 100 + 4 \text{ mA} = 5.6 \text{ mA}$

It should be noted that the readback signal from the transmitter is subject to a scaling in the other direction, i.e. XD_SCALE => (0% – 100%) => PV_SCALE, so that the BKCAL_OUT parameter is always in the same units as are output by the PID block.

2.5.3 Setpoint SP

A setpoint is required by all control and output blocks. In non-cascaded control blocks it is set by the user in Application Designer. For output blocks and cascaded control blocks, it is provided by the CAS_IN parameter, see also Table 2-7. In addition, it is also possible to write a setpoint from a remote application when the block is operating in RCas mode.

Setpoint limits

Application Designer allows the user to apply limits to the setpoint value **SP** by entering appropriate values in the **SP_HI_LIM** and **SP_LO_LIM** parameters. These apply only if the block is operating in Auto mode. If the limits are to apply when the block is operating in Cas or RCas mode, then the option "Obey SP limits if Cas or RCas" must be selected in the **CONTROL_OPTS** parameter.

Setpoint rate limits

To avoid bump when the setpoint **SP** is changed, a ramp can be applied by entering appropriate values in **SP_RATE_UP** and **SP_RATE_DN**. Whether this function is available depends upon the block type and the block mode.

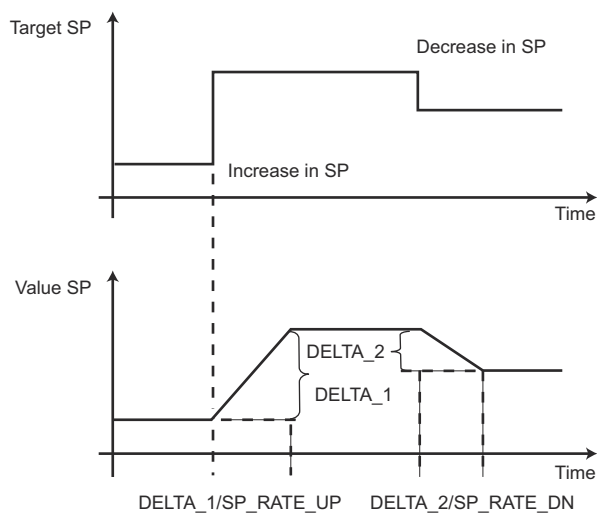


Fig. 2-9: Effect of *SP_RATE_UP* and *SP_RATE_DN* on a change in setpoint

The setpoint rate limits can be applied to the:

- PID block in Auto mode
- AO block in Auto, Cas or RCas modes.

When the block is in Auto mode and the user changes the setpoint **SP** to a value greater than the current value, then the **SP** value ramps upward at a rate based on the parameter **SP_RATE_UP**. If the new setpoint is less than the current value, the **SP** value ramps downward at a rate based on the **SP_RATE_DN**. If the parameter **SP_RATE_DN** and/or **SP_RATE_UP** is zero the rate limiting is disabled.

The Table 2-13 summarizes the conditions for setpoint limits and setpoint rate limits.

Block type	Mode	Required configuration for SP limits SP_HI_LIM/SP_LO_LIM	Required configuration for SP rate limits SP_RATE_UP/SP_RATE_DN
PID	Auto	None	SP_RATE_UP/SP_RATE_DN not zero
	Cas/RCas	CONTROL_OPTS "Obey SP limits if Cas or RCas" is true	Not applicable
AO	Auto	None	SP_RATE_UP/SP_RATE_DN not zero
	Cas/RCas	Not applicable	SP_RATE_UP/SP_RATE_DN not zero.

Tab. 2-13: Configuration of setpoint limit and rate limits

Setpoint tracking PV

Some control strategies require that the transition from a "manual" mode (Rout, Man, LO and Iman) to an "auto" mode (Auto, Cas, Rcas) must be done with the control error equal to zero. In this case, the setpoint SP must be set equal to process value PV.

The **CONTROL_OPTS** of the PID block and the **IO_OPTS** of the AO block allow setpoint tracking of the process value to be activated when the block is in a "manual" mode.

This option is summarized in the following Table 2-14:

Option	CONTROL_OPTS (PID)	IO_OPTS (AO)	Meaning
SP-PV Track in Man	X	X	The SP tracks the PV when the target mode is Man.
SP-PV Track in Rout	X		The SP tracks the PV when the actual mode is Rout.
SP-PV Track in LO or Iman	X	X	The SP tracks the PV when the actual mode is LO or Iman.

Tab. 2-14: Setpoint tracking in *CONTROL_OPTS* and *IO_OPTS*

2.5.4 Output variable, OUT

When the actual block mode is Auto, Cas or RCas, the normal block algorithm is executed and the result output as the **OUT** parameter. This calculation is specific for each function block type. If the mode is a "manual" mode, the output will follow a value provided by another block (LO, Iman), the user (Man) or a control application running on an interface device (Rout).

Output limits

In the PID and ARTH function blocks the parameters **OUT_HI_LIM** and **OUT_LO_LIM** can be used to limit the range of the **OUT** value. The limitation applies to all modes. For the PID block, output limits can be disabled in manual mode by selecting the option "No OUT limits in Manual" in the **CONTROL_OPTS** parameter.

2.5.5 Back calculation

The purpose of back calculation is to provide a value at the **BKCAL_OUT** (or **BKCAL_SEL_N**) parameter that will cause the output of the upstream block to do the right thing to provide a bumpless mode transfer should there be no path from the downstream block to the final control element. **BKCAL_OUT** is linked to **BKCAL_IN** in all cases where it is used, and **BKCAL_IN** always sets the output value when the status requires that action.

The **BKCAL_IN** parameter carries the feedback, i.e. value and status, from the **BKCAL_OUT** parameter of a downstream control or analog output block. When **BKCAL_IN** is linked, the PID Control block always acts on the status it carries. Should the status of **BKCAL_IN** indicate that there is no path to the final element, the block assumes IMan mode and initializes.

If the actual mode of the block is Auto, Cas, or Rcas and **BKCAL_IN** sees a limit status of high or low, the block will treat the value as the high or low **OUT** where the **OUT** limit may be the value offset by the block's **BKCAL_HYS** parameter. The actual mode will not change, and the output will be free to move away from the limit.

If the actual mode of the block is Auto, Cas, or Rcas and **BKCAL_IN** sees a limit status of constant, the block will set its **OUT** to the **BKCAL_IN** value. It will not set the actual mode to IMan, unless the sub-status requires it.

If the status of **BKCAL_IN** is Bad, the actual block mode will go to IMan. The source of **CAS_IN** or **BKCAL_IN** is normally a block that follows the initialization procedure. If the source is a calculation block, the status may be Good (Non-cascade). If the source is a constant, the status may be Uncertain with an Initial or Substitute Value sub-status. In these cases, the initialization procedure will not be followed. The block will go immediately to its target mode when all other conditions are met.

2.6 Status handling

Fieldbus devices have the ability to detect faults that make a measurement bad or prevent an actuator from responding. This information is transmitted together with the value generated by the device in the form of a status attribute. One of the most important features of the function block application process is that the status is passed with the value from one block to another block and can be accessed by an interface device for use in a human interface or a historian.

2.6.1 Status propagation

Fig. 2-9 shows the information flow in a simple control loop. In the so-called forward path, the primary output value of the Analog Input (AI) block provides the input to the PID control block and the primary output of the PID block provides the input to the Analog Output (AO) block. This acts on the information, e.g. by opening or closing the connected valve, and reports the position change to the PID block via the so-called backward path by means of the back calculation parameter.

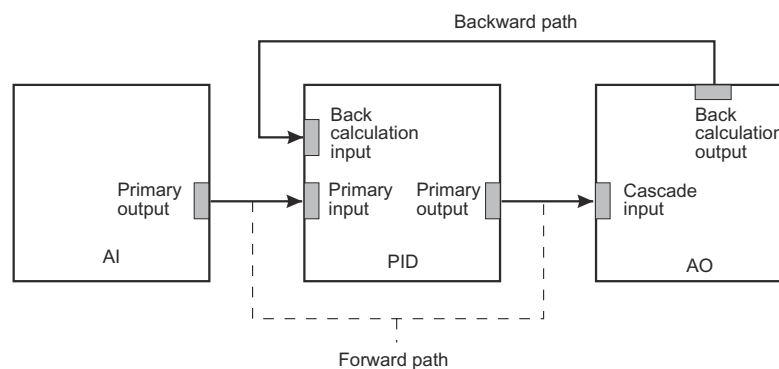



Fig. 2-10: Information flow in a simple control loop

Each value exchanged along the loop has an associated status. In our example, the status of the primary output of the AI block will be used as the status of the primary input of the PID block. Similarly, the status of the back calculation output of the AO block is passed on to the back calculation input of the PID block. There is no specific propagation rule for statuses within blocks, however, so it is perfectly feasible that the status of the primary input of the PID Control block is "Bad", but the primary output has the status "Good".

In normal operation all statuses within the loop will be "Good". A "Bad" status can be generated by a number of situations, e.g. by putting a block into manual mode when changing parameters on-line, incorrect configuration of the block, communication failure, device failure etc. Once the cause of a "Bad" status has been remedied, the follow-up statuses will normally turn to "Good" again. It is possible, however, to configure the application such that certain status changes require an acknowledgement before they return to "Good" again.

2.6.2 Status message

All input and output parameters and some contained parameter values, e.g. setpoint SP and process value PV, are accompanied by a status. This gives the receiver information not only about the quality of the value but also information that it can use in exceptional circumstances. The status itself comprises three components: quality, substatus and limits, as described in Table 2-15.

Table 2-16 lists the possible status messages that can occur during operation. The hexadecimal value indicates that the "Not Limited" limit status is active. Each message may be output with the alternative limit status "High Limited", "Low Limited" or "Constant". The associated hexadecimal values are 1, 2 or 3 units higher than that in the table. In ControlCare Application Designer, the numerical value of the status parameter can be displayed by pressing the **Symbol** button in the toolbar, , see also Appendix A.

Status attributes

Status attribute	Description
Quality	Provides an indication of whether the value can be used in subsequent block processing. It may assume one of four values: <ul style="list-style-type: none"> ■ Good Cascade: the quality of the value is good; it may be part of a cascade path ■ Good Non-Cascade: the quality of the value is good; the block has a cascade path ■ Uncertain: the quality of the value cannot be determined, but the value may still be useful ■ Bad: the value is not useful.
Substatus	Complements the quality and carries information to initialize or break cascade control, alarms and others. There are different sets of substatus for each quality, which are listed in Table 2.14.
Limits	Provides information on whether the associated value is limited or not, as well the direction of the limits. The limits are classified as: <ul style="list-style-type: none"> ■ Not Limited: the associated value is within limits or no limits have been set ■ High Limited: the associated value is violating a high limit ■ Low Limited: the associated value is violating a high limit ■ Constant: the block sending the status is operating in IMAN, MAN or LO mode

Tab. 2-15: Status attributes

Substatus values

Quality	Substatus	Hex value	Cascade path		
			Not in	Forward	Backward
Bad	0 = Non-specific (lowest priority)	0x00	X	X	X
Bad	1 = Configuration Error	0x04	X	X	X
Bad	2 = Not Connected	0x08			
Bad	3 = Device Failure	0x0c	X	X	X
Bad	4 = Sensor Failure	0x10	X	X	X
Bad	5 = No Communication, Last Usable Value	0x14	X		
Bad	6 = No Communication, No Usable Value	0x18	X		
Bad	7 = Out of Service (highest priority)	0x1c	X		

Quality	Substatus	Hex value	Cascade path		
			Not in	Forward	Backward
Uncertain	0 = Non-specific (lowest priority)	0x40	X		
Uncertain	1 = Last Usable Value	0x44	X		
Uncertain	2 = Substitute	0x48	X		
Uncertain	3 = Initial Value	0x4c	X		
Uncertain	4 = Sensor Conversion not Accurate	0x50	X		
Uncertain	5 = Engineering Unit Range Violation	0x54	X		
Uncertain	6 = Sub-normal	0x58	X		

Quality	Substatus	Hex value	Cascade path		
			Not in	Forward	Backward
GoodNC	0 = Non-specific (lowest priority)	0x80	X		
GoodNC	1 = Active Block Alarm	0x84	X		
GoodNC	2 = Active Advisory Alarm	0x88	X		
GoodNC	3 = Active Critical Alarm	0x8c	X		
GoodNC	4 = Unacknowledged Block Alarm	0x90	X		
GoodNC	5 = Unacknowledged Advisory Alarm	0x94	X		
GoodNC	6 = Unacknowledged Critical Alarm	0x98	X		

Quality	Substatus	Hex value	Cascade path		
			Not in	Forward	Backward
GoodC	0 = Non-specific (lowest priority)	0xc0		X	X
GoodC	1 = Initialization Acknowledge(IA)	0xc4		X	
GoodC	2 = Initialization Request(IR)	0xc8			X
GoodC	3 = Not Invited (NI)	0xcc			X
GoodC	4 = Not Selected(NS)	0xd0			X
GoodC	6 = Local Override(LO)	0xd8			X
GoodC	7 = Fault State Active(FSA)	0xdc			X
GoodC	8 = Initiate Fault State (IFS)	0xe0		X	

Tab. 2-16: Substatus values as a function of status quality

2.6.3 Substatus and limit propagation

The manner in which the status of block input parameters is propagated and used in the calculation of the block output status is determined for each function block type. The specific rules are described for each block, however, there are some general rules:

Rule	Upstream status	Received status
Bad "Out of Service", "No Communication" are not propagated, as they are local to the block where they occur	Bad. Out of Service, Not Limited Bad. No Communication, No Usable Value, Not Limited	Bad. Non-specific, Not Limited
Bad "Sensor failure", "Device failure" are propagated intact through non-control functions. Control blocks generate an alert based on the input receiving the bad status.	Bad. Sensor Failure. Not Limited Bad. Device Failure. Not Limited Bad. Configuration Error. Not Limited	Bad. Sensor Failure. Not Limited Bad. Device Failure. Not Limited Bad. Non-Specific. Not Limited
Control blocks do not normally propagate input status to their output parameters. Control Selector is an exception.	Input: Bad. Non-Specific. High Limited	Output: GoodC. Non-Specific. Not Limited
Uncertain substatusess are not propagated	Uncertain. Subnormal. Not limited	Uncertain. Non-specific. Not limited
GoodNC is propagated as GoodNC - Non-specific, since all of substatuses are local	GoodNC. ***. Not Limited	GoodNC. Non-Specific. Not Limited
GoodNC alarm substatuses are not propagated	GoodNC. Active Critical Alarm. Not Limited	GoodNC. Non-Specific. Not Limited
Substatus is not propagated when two or more measurements are combined	GoodNC. Active Advisory Alarm. Not Limited GoodNC. Non-Specific. Not Limited	GoodNC. Non-Specific. Not Limited
A manual mode output is constant, but never uncertain	MAN mode	GoodC. Non-Specific. Constant
GoodC substatuses are propagated according to the function block type	GoodC. ***. Not Limited	GoodC. ***. Not Limited

Tab. 2-17: General rules for status propagation

Status options

In addition to the general rules, specific options may be selected in some function blocks using the **STATUS_OPTS** parameter. :

Bit	Option	Description	Block
0	IFS if BAD IN	Sets Initiate Fault State status in the OUT parameter if the status of the IN parameter is Bad.	PID
1	IFS if BAD CAS_IN	Sets Initiate Fault State status in the OUT parameter if the status of the CAS_IN parameter is Bad	PID
2	Uncertain as Good	If the status of the IN parameter is Uncertain, treats it as Good.	PID, IS
3	Propagate Fail Forward	Propagates a transducer block status Bad.DeviceFailure or Bad.SensorFailure to OUT without generating an alarm	AI, DI
4	Propagate Fail Backward	If the status from the actuator is Bad "Device failure" or "Fault State Active", or Local Override is active, propagates this as Bad "Device Failure" or Good Cascade "Fault State Active", or Local Override to BKCAL_OUT respectively without generating an alarm.	AO, DO
5	Target to Manual if BAD IN	Sets the target mode to Man if the status of the IN parameter is BAD. This latches a PID block into the Man state if the input ever goes bad.	PID
6	Uncertain if Limited	Sets the output status of the block to Uncertain when limit alarms HI_HI_, HI_, LO_ or LO_LO_ are violated	AI
7	BAD if Limited	Sets the output status of the block to Bad when limit alarms HI_HI_, HI_, LO_ or LO_LO_ are violated	AI
8	Uncertain if Man	Sets the output status of the block to Uncertain if the actual mode of the block is Man	AI, DI, IS; IT
9	Target to next permitted mode if BAD CAS_IN	Sets the target mode to next permitted mode if the target mode is CAS and the status of CAS_IN is BAD.	PID
10 - 15	Reserved		

Tab. 2-18: Status options (STATUS_OPTS)

2.6.4 Control loop integrity

The status of control and output block parameters is used for a number of purposes, e.g. to ensure bumpless transfer from one operating mode to another, to prevent reset windup, etc.

Closed-loop control normally includes an integrator, which drives the valve to the position required to bring the measurement to the setpoint. As long as an error exists, the integrator will continue to move its output in a direction that corrects the error. For predictable control, the output must NOT move if the error cannot be corrected.

If the control loop is broken and integration is allowed to continue, it will proceed until it reaches a limit. If the loop is now closed, the process will be disturbed, perhaps dangerously, as the integrator forces the output to a value beyond the valve limits and must first decrease output before the valve begins to close again. This behaviour is known as reset or integral wind-up. The error between measurement and setpoint, therefore, cannot be brought to zero by integration. The integration must be brought back to a reasonable value before control can resume. This period of lost control is avoided by utilizing status information passed between function blocks.

There are four cases to be considered for status generation:

- the forward path
- the backward path
- cascade control
- cascade initialization

Each case requires its own part of the status information.

Forward path

In addition to normal operation with "Good" status, the following conditions might occur:

- Measurement is lost
- Connection to the process is lost
- Measurement is calculated using bad data, but the calculation has enough process data in it to allow control to continue

The following conditions can be passed forward from the source of the problem to the first integrating controller on the path:

Status*	Condition
GoodNC. Non-Specific. High Limited	Value is from a block that cannot generate a higher value because it is limited in that direction, either internally or by the transducer
GoodNC. Non-Specific. Low Limited	Value is from a block that cannot generate a lower value because it is limited in that direction, either internally or by the transducer
GoodNC. Non-Specific. Constant	Value cannot move; it is constant
GoodC. Initiate Fault State (IFS). Not Limited	Value is from a block that has failed. The failure may be propagated to the downstream block for alarm and display purposes
Bad. No Communication, Last Usable Value. Not Limited	Communication has failed along the path, the last usable value is offered
Bad. No Communication, No Usable Value. Not Limited	Communication has failed along the path, there is no usable value
Uncertain. Non-Specific. Not Limited	Quality of the value is uncertain i.e. not good, but not bad either. The loop is degraded, but not broken. The block has an option to use the data or treat it as bad
*Examples of status messages: if a higher priority substatus or limit status is present, it will be passed on	

Tab. 2-19: Forward path status and conditions

In the case of multiple sensors measuring the same value, "Good" status will indicate agreement of a specified number of sensors. "Uncertain" is used for anything less than the required number of sensors, provided at least one is working.

Backward path

In addition to normal operation with "Good" status, it is possible that the loop has been broken between control function block to the actuator.

In order that the upstream block is informed of the condition of the downstream block, a back-calculation value and status is sent along the backward path, i.e. from the downstream block to the upstream block. In the case of an actuator or control function block in cascade mode, this includes information about constrained movement and the working setpoint (or process variable). The information can be used for limiting or locking the output of an integrating controller or for initializing the upstream block.

The following conditions can be passed backward from the source of the problem to the first integrating controller on the path:

Status*	Condition
GoodC. Non-Specific. High Limited	Value is from a block that cannot generate a higher value because it is limited in that direction, either internally or externally
GoodC. Non-Specific. Low Limited	Value is from a block that cannot generate a lower value because it is limited in that direction, either internally or externally
GoodC. Non-Specific. Constant	Value cannot move; it is constant
GoodC. Initiate Fault State (IFS). Not Limited	Value is from a block that has failed. The failure may be propagated to the upstream block for alarm and display purposes
GoodC. Local Override (LO). Not Limited	Value is from an output block that has been locally overridden by a local key switch or has interlock logic active
GoodC. Fault State Active (FSA). Not Limited	Value is from an output block that has Fault State logic active
Bad. No Communication, Last Usable Value. Not Limited	Communication has failed along the path, the last usable value is offered
Bad. No Communication, No Usable Value. Not Limited	Communication has failed along the path, there is no usable value
Bad.Configuration Error.Not Limited	Value is from a Control Selector which has not selected the corresponding input
*Examples of status messages: if a higher priority substatus or limit status is present, it will be passed on	

Tab. 2-20: Backward path status and conditions

Cascade control

In cascade control, the output of control function blocks does not go to an actuator, but becomes the setpoint of a downstream output function block or another control function block.

As far as the upstream control block is concerned, the loop can be broken by anything that breaks the loop of the downstream block or by the downstream control or output block operating in the wrong mode. When the cascade loop is broken, the output of the upstream control block must be made to match exactly the setpoint (or process variable) of the downstream block, in order to ensure bumpless transfer into a cascade mode. The value passed from the downstream to the upstream block to do this is called the initialization value. While the upstream block is setting its output to the initialization value, it will adopt the Iman mode.

In some cases, the source of the setpoint of a block is an interface device which does not contain the function block application. The application running on the interface device still has the same initialization requirements.

In order to implement cascade control, the following conditions are passed between the blocks:

Status	Condition
Conditions passed from the downstream block to the upstream block or interface device	
GoodC. Not Invited. Not Limited	Value is an initialization value for a source (CAS_IN, RCAS_IN or ROUT_IN), because the downstream loop is broken or the mode of the downstream block is wrong. Cascade control is not invited.
GoodC. Initialization Request (IR). Not Limited	Downstream block invites an initialized response on CAS_IN, RCAS_IN or ROUT_IN because the downstream loop is no longer broken and the target mode is now correct for the input.
Condition passed from the upstream to the downstream block	
GoodC. Initialization Acknowledge (IA). Not Limited	Downstream block invites an initialized response on CAS_IN, RCAS_IN or ROUT_IN because the downstream loop is no longer broken and the target mode is now correct for the input.
*Examples of status messages: if a higher priority substatus or limit status is present, it will be passed on	

Tab. 2-21: Cascade status and conditions

The cascade path is always handled the same way for any block that supports a cascade structure (i.e. has at least one backward path output). It is handled in the same way for each of Cas, RCas, and ROut modes, but with different parameters in the forward and backward paths. Table 2-22 specifies the parameters.

Mode	Forward	Backward
Cas	OUT	BKCAL_OUT
RCas	RCAS_IN	RCAS_OUT
ROut	ROUT_IN	ROUT_OUT

Tab. 2-22: Back calculation parameters as a function of block mode

In the case of RCas and ROut modes, both forward and backward parameters are part of the same block and are contained parameters read or written by an interface device. For Cas mode, the forward path is the **OUT** parameter of the upstream block linked to the **CAS_IN** parameter of the downstream block. The backward path comprises the **BKCAL_OUT** parameter of the downstream block linked to **BKCAL_IN** parameter of the upstream block.

Cascade initialization

When two blocks are operating in cascade and the loop is broken, the substatuses exchanged between them control the recovery to normal operation. Provided both blocks have the target modes required for cascade control, the procedure is as follows:

Step	Mode*	Action
1	Not cascade mode	<ul style="list-style-type: none"> The loop is broken because the operating mode or conditions do not permit the cascade The downstream block sends "Not Invited" as its output status The operating mode of the upstream block moves to lman and a reply is sent
2	Initialize	<ul style="list-style-type: none"> Provided the upstream block has not replied with "Initialization Acknowledge", the downstream block moves to the next permissible operating mode. The downstream block now sends "Initialization Request" as its output status and waits for the reply "Initialization Acknowledge".
3	Initialization complete	<ul style="list-style-type: none"> The upstream block initializes, moves to the next permissible mode and replies with "Initialization Acknowledge" The downstream block moves to its normal operating mode, begins normal operation using the input value provided by the upstream control block, and sends a normal status, e.g. "Non-Specific" as its output
4	Cascade complete	<ul style="list-style-type: none"> The upstream block now begins normal operation and sends a normal status as its output, e.g. Non-Specific.

Tab. 2-23: Status generation on block initialization

2.7 Fault state handling

Fault state is a special condition for an Analog Output or Discrete Output block that allows the connected device to fail-to-safe when the block detects an abnormal situation or the user sets the condition "Fault State" in the resource block.

An abnormal situation occurs when:

- there is bad input to the block (bad sensor, for example)
- there is no communication between blocks for a period longer than a specified time

Blocks that support cascade control such as PID, OSDL and SPLT, propagate the fault state status forward to the output block. When the condition that activated the fault state is rectified, the fault state is cleared and the block returns to the normal operation.

2.7.1 Control blocks

Blocks such as PID, OSDL and SPLT that support cascade control must be configured to send an Initiate Fault State (IFS) status when they detect a bad input. The following options must be selected:

- Option "IFS if bad IN" in the **STATUS_OPTS** parameter and/or
- Option "IFS if bad CAS_IN" in the **OSDL_OPTS** parameter must be selected

The block now generates an IFS status in its **OUT** value when it detects a bad input.

Should a block operating in cascade mode (Cas, RCas) receive an IFS status, it is reported to forward path. For example, a PID block might receive a "Good Cascade IFS" status in the **CAS_IN** input from an upstream PID block. If the target mode of the PID block is Cas, then the IFS status will become the status of **OUT**, replacing the normal status. If the block is operating in Auto, however, the IFS status is not propagated forward.

2.7.2 Output blocks

Output blocks must be configured to respond to the IFS status. This is done by setting up a timeout parameter **FSTATE_TIME** and a fail-safe parameter **FSTATE_VAL** (AO block) or **FSTATE_VAL_D** (DO block) as follows:

- Set **FSTATE_TIME**, in units of 1/32 ms, to the time the output block can normally wait before responding to a fault state in or loss of communication with the upstream block
- Set **FSTATE_VAL/FSTATE_VAL_D** to the set point value (SP/SP_D) that causes the output device to assume fail-safe position.

The **IO_OPTS** parameter must now be configured to determine the block response to a fault state, whereby multiple choices are allowed. The option is set by ticking the selection box:

Option	Set	Clear)
Fault State to value	On fault state, FSTATE_VAL/ FSTATE_VAL_D substitutes SP/SP_D	On fault state, the last valid SP/SP_D value is used
Target to Man if IFS	On fault state, the block mode is forced to Man. In this mode, parameters can be changed	On fault state, the block mode is forced to Local Override, parameters cannot be changed until the fault state is cleared

Tab. 2-24: Configuration of IO_OPTS for fault state handling

A third option, "Use Fault State value on restart" causes the value of FSTATE_VAL/FSTATE_VAL_D to be used on device restart. In this case, the value is used, although no fault state exists.

The output block will now respond to the following fault conditions by entering the configured fault state:

- Loss of communication to **CAS_IN** for a time that exceeds **FSTATE_TIME**
- IFS status in the **CAS_IN** input when the target mode is Cas for a time that exceeds **FSTATE_TIME**
- IFS status in the **RCAS_IN** when the target mode is Rcas for a time that exceeds **FSTATE_TIME**

In the backward path, the block sends the Not Invited (NI) status to indicate that the block is in "Fault State".

After re-establishment of communication or closing of the circuit which initiated the action:

- blocks in Local Override recover automatically to the target mode
- blocks in Man must be reconfigured to adopt the target mode.

2.7.3 Forcing fault state

The resource block can be used to force all output function blocks associated with the device to change immediately to their fault state condition. The setup is as follows:

- Check that "Fault State supported" is available in the **FEATURES** parameter
- Select the option "Fault State supported" in **FEATURES_SEL** to enable the Fault State function in the resource block.
- Set **SET_FSTATE** to On to cause all output blocks to adopt their fault state behaviour (**FAULT_STATE** becomes active in the **STATUS_OPTS**)
- Set **CLEAR_FSTATE** to On to cause all output blocks to resume normal operation (**FAULT_STATE** is cleared in the **STATUS_OPTS**).
- **FAULT_STATE** indicates the current Fault State in the resource block.

2.8 Function block options

In addition to the common parameters, many blocks share common option parameters with which control the responses to input/output, control or status conditions. These are available in four bit strings that contain static configuration binary choices, see also previous chapters.

2.8.1 GRANT_DENY

GRANT_DENY controls access permission to various types of block parameter for higher level applications, e.g. a host computer or local operating panel. The parameter has two attributes named Grant and Deny. The operation of these parameters is as described below, but the actual usage (if any) depends on the philosophy of the plant.

Grant

Depending on the philosophy of the plant, the operator, a higher level device or when local is selected, a local operator's panel may turn on an item of the Grant attribute. By doing or allowing this action, the operator gives up control of the selected parameters to the intervening device.

The function block does not check writes to any of the selected parameters for grant-deny permission. It is up to other devices to obey and enforce the rules, because the function block has no way to know what is writing to it. When the operator wants to regain control of the parameters, he clears the Grant item. The function block will then automatically set the corresponding Denied item. This indicates to the intervening device that control has been taken away.

Deny

The Deny attribute is provided for use by a monitoring application in an interface device and may not be changed by an operator. It allows the monitoring application to determine if control has been temporarily taken away during the execution of a batch program. This is done by first clearing one or all of the Denied items before execution of a batch program, then checking the Denied item after execution. The Grant item itself should not be checked for this condition, because the operator may have cleared and subsequently set the Grant item during batch program execution, a sequence that might be missed by a slowly scanning monitor program. The Denied item may not be cleared by the operator, thus latching the fact that control was taken away.

Options

Table 2-25 indicates the various options of the GRANT_DENY parameter

Bit	Option	Description
GRANT		
0	Program	A higher level device may change the: <ul style="list-style-type: none"> ■ target mode, ■ setpoint (if the block mode is Man or Auto) or ■ output (if the block mode is Man) of the block
1	Tune	A higher level device may change the tuning parameters of the block
2	Alarm	A higher level device may change the alarm parameters of the block
3	Local	A local operator's panel or hand-held device may change the: <ul style="list-style-type: none"> ■ target mode, ■ setpoint (if the block mode is Man or Auto) or ■ output (if the block mode is Man) of the block
DENY		
0	Program Denied	The Program permission item has been turned off
1	Tune Denied	The Tune permission item has been turned off
2	Alarm Denied	The Alarm permission item has been turned off
3	Local Denied	The Local permission item has been turned off

Tab. 2-25: GRANT_DENY options

2.8.2 IO_OPTS

IO_OPTS activates the options for processing the input and output values of the function block (I/O options). The following options apply when they have been selected and the appropriate condition is true:

Bit	Option	Description	Block
0	Invert	Indicates whether the discrete input or output value should be logically inverted before it is stored in the process variable. Normally a discrete value of zero(0) will be considered to be a logical zero(0) and a non-zero discrete value will be considered to be a logical (1). If invert is selected, the logical NOT of a non-zero field value would result in a zero(0) discrete output, the logical NOT of a zero field value would result in a discrete output value of one(1).	DI, DO
1	SP-PV Track in MAN	Permits the setpoint to track the process variable when the target mode of the block is MAN.	AO, DO
2	Reserved		
3	SP-PV Track in LO	Permits the setpoint to track the process variable when the actual mode of the block is LO.	AO, DO
4	SP Track retained target	Permits the setpoint to track the RCAS or CAS parameter based on the retained target mode when the actual mode of the block is LO or MAN. When SP-PV track options are enabled, then SP Track retained target will have precedence in the selection of the value to track if the actual mode is MAN and LO	AO, DO
5	Increase to close	Indicates whether the output value should be inverted before it is communicated to the I/O channel.	AO
6	Fault State to value	Selects the output action to take when fault occurs. (0: freeze, 1: go to preset value)	AO, DO
7	Use Fault State value on restart	Uses the value of FSTATE_VAL(_D) if the device is restarted, otherwise use the non-volatile value. This does not act like Fault State, just uses the value	AO, DO
8	Target to Man if Fault State activated	Sets the target mode to MAN, thus losing the original target, if Fault State is activated. This latches an output block into the manual mode	AO, DO
9	Use PV for BKCAL_OUT	The BKCAL_OUT value is normally the working SP. This option changes it to the PV	AO, DO
10	Low cutoff	Enables the low cutoff algorithm for flow measurements	AI
11	Reserved		
12	Reserved		
13	Reserved		
14	Reserved		
15	Reserved		

Tab. 2-26: Input/Output block options

2.8.3 CONTROL_OPTS

CONTROL_OPTS contains the available controller options for specifying the automation strategy. The options apply when they have been selected and the appropriate condition is true:

Bit	Option	Description	Block
0	Bypass Enable	Allows BYPASS to be set. Some control algorithm applications cannot provide closed loop control if bypassed.	PID, EPID, APID, CHAR
1	SP-PV Track in MAN	Permits the setpoint to track the process variable when the target mode of the block is MAN..	PID, EPID, APID, STEP
2	SP-PV Track in ROUT	Permits the setpoint to track the process variable when the actual mode of the block is ROUT	PID, EPID, APID
3	SP-PV Track in LO or IMan	Permits the setpoint to track the process variable when the actual mode of the block is LO or IMAN.	PID, EPID, APID, STEP
4	SP Track retained target	Permits the setpoint to track the RCAS or CAS parameter based on the retained target mode when the actual mode of the block is IMAN, LO, MAN, or ROUT. When SP-PV track options are enabled, then SP Track retained target will have precedence in the selection of the value to track when the actual mode is MAN, IMAN, ROUT, and LO	PID, EPID, APID, STEP
5	Direct Acting	Defines the relationship between a change in PV and corresponding change in output. When Direct is selected, an increase in PV results in an increase in the output.	PID, EPID, APID; STEP
6	Balance Ramp	Not used	
7	Track Enable	Enables the external tracking function. If true, the value in TRK_VAL will replace the value of OUT if TRK_IN_D becomes true and the target mode is not MAN.	PID, E; PID, APID
8	Track in Manual	Enables TRK_VAL to replace the value of OUT when the target mode is MAN and TRK_IN_D is true. The actual mode will then be LO	PID, EPID, APIDP
9	Use PV for BKCAL_OUT	BKCAL_OUT and RCAS_OUT values are normally the working SP. If this option is enabled, then the PV value will be used after the cascade is closed.	PID, EPID, APID, STEP
10	BIAS may be adjusted	Not used	
11	Convert IN_1 to OUT_SCALE	Not used	
10	Act on IR	If this option is true, then when IR (initiation request) is received as the status in BKCAL_IN, the SP will be adjusted within setpoint limits to provide bumpless transfer when the cascade is closed. If the setpoint required to provide bumpless transfer is outside the setpoint limits, then any difference added to provide bumpless transfer will be removed in the BAL_TIME	PID, EPID, APID
11	Use BKCAL_OUT with IN_1	Normally, BKCAL_OUT is associated with initialization of an upstream block which is providing CAS_IN. If this option is set, then BKCAL_OUT is to be associated with the upstream block providing IN_1. This option may be used with the Ratio and Bias/Gain block to determine the value and status which must be provided in BKCAL_OUT for proper initialization and handshaking.	PID, EPID, APID
12	Restrict SP to limits in CAS or RCAS	Normally the setpoint will not be restricted to the setpoint limits except when entered by a human interface device. However, if this option is selected, the setpoint will be restricted to the setpoint absolute limits in the Cas and RCas modes	PID, EPID, APID, STEP
13	No OUT limits in MAN	Does not apply OUT_HI_LIM or OUT_LO_LIM when target and actual modes are MAN. Trust the operator to do the right thing.	PID, EPID, APID
14	Reserved		
15	Reserved		

Tab. 2-27: Control options

Tab. 2-28:

2.9 Alert handling

FOUNDATION Fieldbus Specification 890 describes an alert handling concept for FOUNDATION Fieldbus applications. Alerts are alarms and events, which represent state changes within function block applications. Within the model, devices supporting alert handling report to an interface device via an alert object. The interface device is responsible for alert processing. If e.g. a SCADA system is used as operational interface, this has the consequence that the alarming can be configured offline in the individual function block during the engineering phase. The operating system then accesses the alert parameters through the HSE OPC server.

Handling mechanism

Alerts are handled as follows. The alert notifier examines the results of resource, transducer and function block executions to determine if any of a defined set of alert states has been entered. When this is the case, the alert object builds a report message, a so-called event notification, and publishes it to the network. The time at which the alert state was detected is included as a time stamp in the alert message.

If several alerts are present at one time, they are queued and published according to their age and priority. The associated alert is removed from the queue as soon as a reply is received confirming the receipt of the notification. The time-out period for receiving replies and number of alerts awaiting reply can be configured in the Resource Block of each device. If the time-out period is exceeded, the alert is added to a retransmission list.

Alert handling also includes acknowledgement of alerts. This indicates that the alert has been processed by an interface device to satisfy operational interface requirements. The acknowledgement can be configured to be disabled, manual or automatic.

2.9.1 Events and alarms

Events are usually associated with a change in configuration caused by, e.g. a user intervention. As mentioned previously, specific changes in dynamic parameters are not tracked, as these are part and parcel of normal operation. The following events are signalled and can be tracked when the appropriate parameters are set:

- A change of status of the **WRITE_LOCK** parameter in the Resource Block, see Chapter 3.2.6
- Any change in a static parameter in any block

The parameter **UPDATE_EVT** provides an alert for any change in a static parameter and includes the relative parameter index and its associated block index together with the new value of **ST_REV**. The alert is auto-acknowledged. Alerts are not generated while a block is Out of Service, but the **ST_REV** parameter continues to increment. When the block returns to normal operation, an alert is generated if the **ST_REV** parameter has increased.

There are two basic types of alarm, whereby both entering and exiting a limiting alarm condition creates a change in alert status:

- Block alarms reflect errors in the hardware and software associated with the block
The alarms generated are displayed in the parameter **BLOCK_ERR** and depend upon the block. Block alarms always have Priority 2. The parameter **BLOCK_ALM** displays details, communicates the alarm to the interface device and allows the alarm to be acknowledged.
- Limit alarms are generated due to the limiting of process variables
The parameter **ALARM_SUM** summarizes the current status of all the limit alarms. Limits are set in the **xxx_LIM** parameter of the block and the **ALARM_HYS** parameter imposes a universal hysteresis. Alarm priority is set in the appropriate **xxx_PRI** parameter. The **xxx_ALM** parameter displays the current status of the limit and allows alarms to be acknowledged.

2.9.2 Reporting, priority and acknowledgement

Reporting

Alert reporting is set up for the entire resource through the device Resource Block. The message "Reports supported" in the **FEATURES** parameter of the resource block indicates that the device has an alert notifier. The reporting of alerts must be activated in the Resource Block by selecting the option "Reports supported" in the **FEATURE_SEL** parameter, see Chapter 3.2.4.

Three parameters control the queuing of alerts:

- **MAX_NOTIFY** can be read online and indicates the maximum number of alert reports that the device can send without receiving a confirmation
- **LIM_NOTIFY** allows the user to select the number of alert reports the device can send without receiving a confirmation. If **LIM_NOTIFY** is set to zero, then no alerts are reported.
- **CONFIRM_TIME** determines the time the resource will wait for confirmation of receipt of a report before trying again. If the **CONFIRM_TIME** = 0 the device will not retry.

Priority and acknowledgement

The alert priority is set in the corresponding alarm priority parameter of the function block concerned. By default, an alert must be manually acknowledged in the corresponding alarm display parameter, see Chapters 2.9.4 and 2.9.5. manual acknowledgement can be disabled for one or more alarm classes by ticking the option "xxx Alarm Disable" or "xxx Alarm Auto Acknowledge" (depends on the device DD) in the parameter **ACK_OPTION**. When disabled, the alert is automatically acknowledged.

Table 2-29 summarizes the parameters to be configured to activate alert reporting and set up alarm priorities and acknowledgement.

Parameter	Description
ALERT_KEY	Optional index that identifies the plant unit, see Chapter 2.3.4
FEATURE_SEL	Activates the salient feature provided it is supported by the device <ul style="list-style-type: none"> ■ Reports: When ticked, activates reporting for all blocks ■ Soft Write Lock: When ticked, activates reporting for change in WRITE_LOCK parameter, provided there is no hardware lock
MAX_NOTIFY	Indicates maximum number of unconfirmed reports that can be stored by the device
LIM_NOTIFY	Sets number of unconfirmed reports that to be stored by the device <ul style="list-style-type: none"> – LIM_NOTIFY may not exceed MAX_NOTIFY – If LIM_NOTIFY = 0, then no alerts are reported
CONFIRM_TIME	Sets time limit for receipt of alert notification confirmation from interface device. <ul style="list-style-type: none"> – Setting is multiplied by 1/32 milliseconds to obtain time period – If no confirmation is received within this time, the report is sent again
ACK_OPTION	Enables or disables auto-acknowledgment in block for each class of alarm. When disabled (no tick in box), the alarms must be acknowledged in the salient xxx_ALM parameter. For resource block: <ul style="list-style-type: none"> ■ 1: Write Lock Alarm Disabled /Write Lock Alarm Auto Acknowledge ■ 8: Block Alarm Disabled (Resource block)/Block Alarm Auto Acknowledge For other blocks, see also Table 2-31: <ul style="list-style-type: none"> ■ 0: Discrete Alarm Disabled ■ 1: HiHi Alarm Disabled ■ 2: Hi Alarm Disabled ■ 3: LoLo Alarm Disabled ■ 4: Lo Alarm Disabled ■ 5: Deviation Hi Alarm Disabled/Deviation Hi Alarm Auto Acknowledge ■ 6: Deviation Lo Alarm Disabled/Deviation Lo Alarm Auto Acknowledge ■ 7: Block Alarm Disabled/Block Alarm Auto Acknowledge
WRITE_PRI DISC_PRI LO_LO_PRI LO_PRI HI_PRI HI_HI_PRI DV_HI_PRI DV_LO_PRI	Priority of an alarm or event. If no setting is made the acknowledgement is "undefined". <ul style="list-style-type: none"> ■ 0-1: Associated alert is not sent as a notification; normally auto-acknowledged. ■ 2: Reserved for alerts that do not require the attention of a plant operator. Block alarm and update event have this priority; normally auto-acknowledged. ■ 3-7: Advisory alarms. Alarms of this priority will normally be acknowledged. ■ 8-15: Critical alarms. Alarms of this priority will normally be acknowledged.

Tab. 2-29: Parameters for alert reporting

2.9.3 Limit alarms

The user can set limits on a number of process values that are used by various function blocks. Table 2-30 summarizes the possibilities

Block	Parameter	Disc	Hi_Hi	Hi	Lo_Lo	Lo	Dev_Hi	Dev_Lo
AI	OUT		x	x	x	x		
DI	OUT_D	x						
PID, AALM	PV		x	x	x	x		
PID	PV-SP						x	x
SPG	BKCAL_IN/OUT						x	x

Tab. 2-30: Limit values as a function of parameter and block

An alarm occurs when a value meets, exceeds or drops below a limit.

- For a high alarm, an alarm is true when the analog value is greater than the limit. The status of the alarm remains true until the value drops below the limit minus any alarm hysteresis.
- For a low alarm, an alarm is true when the analog value is lower than the limit. The status of the alarm remains true until the value rises above the limit plus any alarm hysteresis.

An alarm type can be disabled by setting its respective alarm limit parameter to \pm infinity. This is the default value of all alarm limits. The value range is given by the scaling of the output parameter. If the entered value exceeds this range (excluding \pm infinity), a block configuration error is flagged in **BLOCK_ERR**.

The parameters required to set up process value limiting are summarized in Table 2-31.

Parameter	Block	Description
DISC_LIM	DI	Value of discrete output OUT_D (0 or 1) for which an alert is to be sent
HI_HI_LIM	AI, AALM, PID	Limit value in units of PV_SCALE or OUT_SCALE for upper critical limit
HI_LIM		Limit value in units of PV_SCALE or OUT_SCALE for upper advisory limit
LO_LIM		Limit value in units of PV_SCALE or OUT_SCALE for lower advisory limit
LO_LO_LIM		Limit value in units of PV_SCALE or OUT_SCALE for lower critical limit
DV_HI_LIM	PID, SPG	Limit value in units of PV_SCALE for PV or PV-SP upper limit
DV_LO_LIM		Limit value in units of PV_SCALE for PV or PV-SP lower limit
ALARM_HYS	AI, AALM, PID	Amount the PV or OUT value must return within the alarm limits, expressed as a percent of the PV/OUT span, before the alarm condition clears. The span used depends on the block type: <ul style="list-style-type: none"> ■ PID: PV_SCALE ■ AI, Setpoint generator, Analog alarm: OUT_SCALE
DISC_PRI LO_LO_PRI LO_PRI HI_PRI HI_HI_PRI DV_HI_PRI DV_LO_PRI	DI, AI, AALM, PID, SPG	Priority of an alarm or event. If no setting is made the acknowledgement is "undefined". <ul style="list-style-type: none"> ■ 0-1: Associated alert is not sent as a notification; normally auto-acknowledged. ■ 2: Reserved for alerts that do not require the attention of a plant operator. Block alarm and update event have this priority; normally auto-acknowledged. ■ 3-7: Advisory alarms. Alarms of this priority will normally be acknowledged. ■ 8-15: Critical alarms. Alarms of this priority will normally be acknowledged.
ACK_OPTION		Enables or disables auto-acknowledgment in block for each class of alarm. When disabled (no tick in box), the alarms must be acknowledged in the salient xxx_ALM parameter. <ul style="list-style-type: none"> ■ 0: Discrete Alarm Disabled ■ 1: HiHi Alarm Disabled ■ 2: Hi Alarm Disabled ■ 3: LoLo Alarm Disabled ■ 4: Lo Alarm Disabled ■ 5: Deviation Hi Alarm Disabled/Deviation Hi Alarm Auto Acknowledge ■ 6: Deviation Lo Alarm Disabled/Deviation Lo Alarm Auto Acknowledge ■ 7: Block Alarm Disabled/Block Alarm Auto Acknowledge

Tab. 2-31: Parameters for process value limiting

2.9.4 Event update and alarm summaries

BLOCK_ERR

The **BLOCK_ERR** parameter reflects the error status of hardware or software components associated with the block and that have a direct impact on its correct operation. It comprises a bit string in which block error conditions (0= inactive, 1 = active) are reflected as follows:

Bit	Condition	Bit	Condition
0	Other (LSB)	8	Output Failure
1	Block Configuration Error	9	Memory Failure
2	Link Configuration Error	10	Lost Static Data
3	Simulate Active	11	Lost NV Data
4	Local Override	12	Readback Check Failed
5	Device Fault State Set	13	Device Needs Maintenance Now
6	Device Needs Maintenance Soon	14	Power-up
7	Input Failure/ process variable has BAD status	15	Out-of-Service (MSB)

Tab. 2-32: Block error parameter conditions

The bits maintained depend upon the type and function of the block. The block alarms are displayed and communicated to the host system via the parameter **BLOCK_ALM**. The alarms must be acknowledged in the **BLOCK_ALM** parameter, see also Chapter 2.9, unless the option "Block Alarm Auto Acknowledge"/"Block Alarm Disabled" (Depends on Device DD) has been ticked in the parameter **ACK_OPTION**.

Note!



- For the resource block, simulate active is used to indicate whether the simulate hardware jumper is present on the associated device. An active state (1) of this attribute indicates that the jumper is present and that it is possible for the user to enable simulation in an input or output class function block.

XD_ERROR

The **XD_ERROR** parameter reflects the error status of hardware or software components associated with a transducer block and that have a direct impact on its correct operation. It comprises a bit string in which block error conditions (0= inactive, 1 = active) are reflected as follows:

Bit	Condition	Bit	Condition
16	Unspecified Error	21	Mechanical Failure
17	General Error	22	I/O Failure
18	Calibration Error	23	Data Integrity Error
19	Configuration Error	24	Software Error
20	Electronics Failure	25	Algorithm Error

Tab. 2-33: Transducer block parameter conditions

UPDATE_EVT

The **UPDATE_EVT** parameter captures the dynamic information associated with a write to a static parameter within the block. The parameter has the following attributes:

- **UNACKNOWLEDGED**: shows the status of the alert acknowledgement
- **UPDATE_STATE**: shows reported state of the update
- **TIME_STAMP**: shows the time at which the static parameter was changed
- **STATIC_REVISION**: shows the new static revision level (ST_REV)
- **RELATIVE_INDEX**: shows the index of the changed parameter

ALARM_SUM

The **ALARM_SUM** parameter summarizes the status of up to 16 process alarms belonging to the same block. The parameter has four attributes, see also Chapter 2.9.5:

- **CURRENT**: shows for each alarm whether it is active
- **UNACKNOWLEDGED**: shows for each alarm whether it has been acknowledged
- **UNREPORTED**: shows for each alarm whether it has been reported
- **DISABLED**: shows for each alarm whether it has been disabled in **ACK_OPTION**

2.9.5 Alarm display parameters

Each block has one or more parameters to capture the dynamic information associated with an alarm.

- Hardware and software errors for the block are captured in the parameter **BLOCK_ALM**
- For the resource block, **WRITE_LOCK** changes are captured in the parameter **WRITE_ALM**
- Limit value alarms are captured in the corresponding parameter **xxx_ALM**

All types have the structure shown in Table 2-34. The information contained is transferred to an alert object when the alarm is reported. If the alert has a priority greater than 2 and the alarm class has not been disabled in the **ACK_OPTIONS** parameter, it can be acknowledged in the parameter **UNACKNOWLEDGE**.

Parameter	Attribute	Value	Description
BLOCK_ALM WRITE_ALM DISC_ALM	UNACKNOWLEDGED	As read or entered	Indicates acknowledgement of current alert state: i.e. Undefined, Acknowledged, Unacknowledged ■ User can acknowledge via this parameter
HI_HI_ALM HI_ALM LO_ALM LO_LO_ALM DV_HI_ALM DV_LO_ALM	ALARM_STATE	As read	Indicates of whether the alert is active and reported. The alarm state is cleared when the block goes to Out of Service mode. Possible values are: ■ Clear-Reported, Clear-Not Reported, Active-Reported, Active-Not Reported
	TIME_STAMP	As read	Indicates time when the alarm state was detected. The value is maintained until alert confirmation has been received.
	SUB_CODE	As read	Displays number specifying the cause of the alert
	VALUE	As read	The value of the associated parameter at the time the alert was detected

Tab. 2-34: Alarm display parameters

2.9.6 Example

Fig. 2-11 shows a typical response to a HI limit violation. A HI_LIM of 190 is set for the OUT variable of a function block. The alarm hysteresis parameter ALARM_HYS is set to 10. When the OUT value exceeds 190, the HI_ALM.Unacknowledged parameter changes from Acknowledged to Unacknowledged and the HI_ALM.ALARM_STATE parameter changes to Active_Not_Reported. An alarm notification message is now output to the bus. The HI_ALM.ALARM_STATE parameter changes to Active_Reported. The Alarm Notification is answered within the set CONFIRM_TIME.

When the operator acknowledges the alarm in the HI_ALM.Unacknowledged parameter, this changes to Acknowledged.

The process value now returns to limits, the HI_ALM.ALARM_STATE parameter changes to Clear_Not_Reported only after it passes the HI limit minus the hysteresis (for LO limits, the hysteresis acts in the other direction). At this point an alarm notification message is output to the bus and the HI_ALM.ALARM_STATE parameter changes to Clear_Reported. In the example, no reply is received to the message within the set CONFIRM_TIME, and the message is output again.

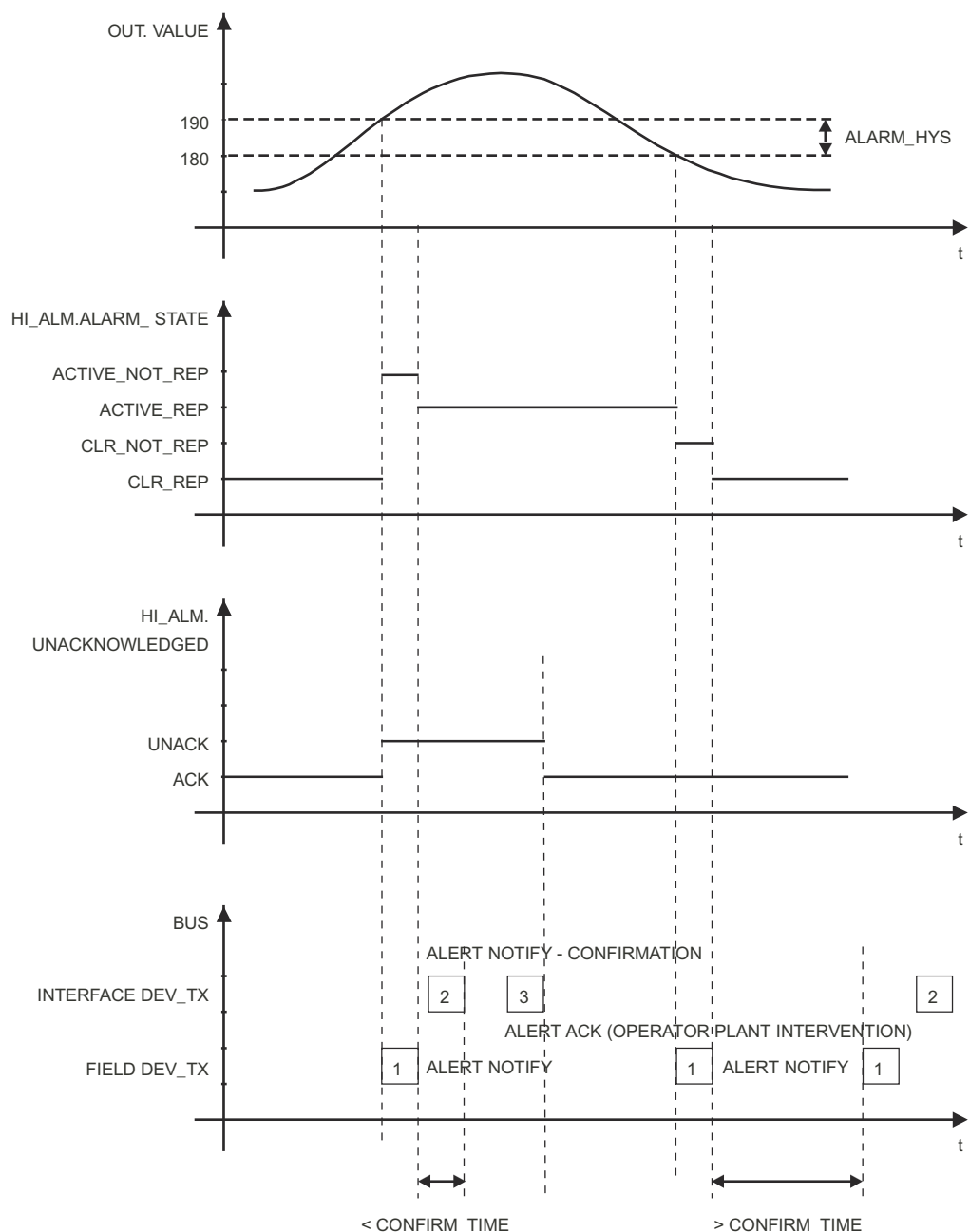


Fig. 2-11: Alert processing

2.10 Simulation

Simulation allows the checking and diagnosis of function blocks and control loops during installation and commissioning of the plant. To this end, all input and output function blocks have a **SIMULATE**, **SIMULATE_D** or **SIMULATE_P** parameter, which overrides the transducer block input or hardware channel output when the function is enabled.

Enable simulation

In order to prevent accidental switching to simulation mode during operation, every device has a hardware lock, i.e. a jumper, switch or relay that controls the simulation function within the device as a whole. The state of this lock is indicated in the resource block **BLOCK_ERR** parameter, which flags "Simulation Active" when it is enabled. If the hardware lock is in the disable position, it is not possible to write to the **SIMULATE** (**SIMULATE_D** or **SIMULATE_P**) parameters.

Once activated within the device as a whole, simulation can be enabled within an individual I/O block by expanding the **SIMULATE** (**SIMULATE_D** or **SIMULATE_P**) parameter.

- Set the status of the simulated variable in **SIMULATE_STATUS** etc.
- Set the value of the simulated variable in **SIMULATE_VALUE** etc.
- Set **ENABLE_DISABLE** to "Active" to start simulation in the block

The parameters **TRANSDUCER_STATUS** and **TRANSDUCER_VALUE** indicate the status and value of the raw process value or readback value. The block output/readback reflects the simulated parameters after processing in the block. When simulation is active, the **BLOCK_ERR** parameter of the I/O block flags "Simulation Active".

Disable simulation

Simulation in an I/O block is disabled by setting **ENABLE_DISABLE** to "Disable". In this state **SIMULATE_STATUS** and **SIMULATE_VALUE** etc. track **TRANSDUCER_STATUS** and **TRANSDUCER_VALUE** in order to provide a bumpless transfer from disabled to enabled.

Simulation for the whole device is disabled by resetting the hardware lock.

Note



- Some devices must be restarted in order to detect a change in the hardware lock, while others do this during normal device operation.

Effect

Table 2-35 indicates the effect of simulation on the output of the block. When simulation is enabled in an input block, the simulated parameter becomes the **OUT** parameter (Process Value) after scaling, linearization and filtering. The status value can be used to simulate transducer faults.

For output blocks, the simulated parameter becomes the **READBACK** parameter. The status can be used to simulate transducer faults. The transducer value and status reflect the transducer readback value and status. The transducer maintains last the output and ignores the **OUT** of the block.

Block	Simulation condition	Action
AI, DI, PUL	Enable	SIMULATE.Simulate Value and Status -> PV (after scaling, linearization and filtering)
	Disable	SIMULATE.Transducer Value and Status -> PV (after scaling, linearization and filtering) and SIMULATE.Simulate Value and Status
AO, DO	Enable	SIMULATE.Simulate Value and Status -> READBACK
	Disable	SIMULATE.Transducer Value and Status -> READBACK and SIMULATE.Simulate Value and Status

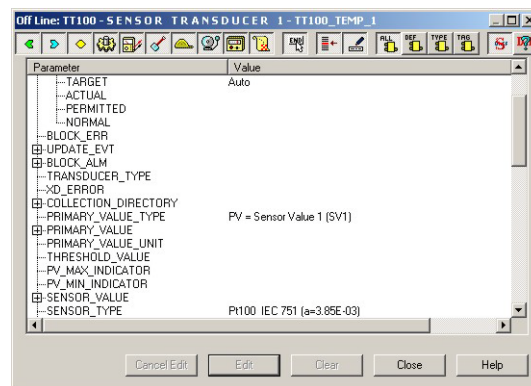
Tab. 2-35: Effect of simulation

2.11 Changing function block parameters

In Application Designer, function block parameters may be set either off-line, e.g. during the planning stage, or on-line when the project is running. When on-line, Read Only parameters show the current values generated by the devices in the project. With a few exceptions, e.g. SP of a PID block, Read/Write parameters are not directly editable on-line and the block must be put out of service as described in Chapter 2.11.2 below.

2.11.1 Off-line characterization

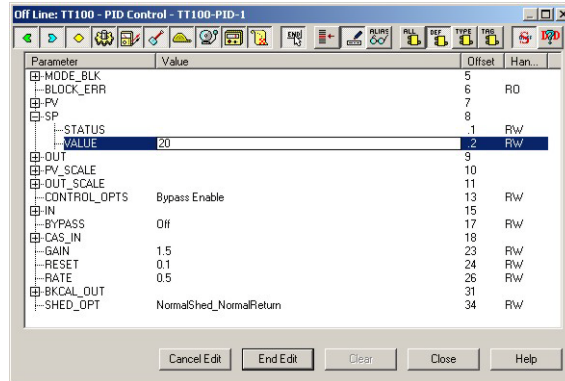
- 1 In the Fieldbus/PROFIBUS network or Control module workspace, right-click on the function block and select **Off-line Characterization**
 - The function block **Off-line Characterization** dialog appears



- 2 Change the parameters you wish to modify
 - If appropriate, open the parameter leaf and double-click in the space next to the parameter you require
 - Enter the new parameter or select it from the drop-down menu
 - Click **End Edit** to set the parameter
 - Repeat the procedure for all the parameters you wish to modify
 - Press **Close** to store the values
- 3 Click on the **Project View** workspace and **Export Tags...**,
 - Open **Project File**, then press **Save Entire Project** to save the project

2.11.2 On-line characterization

- 1 In the Control strategy workspace double-click on the function block you want to modify, or in the Fieldbus/ PROFIBUS network or Control module workspace, right-click on the function block and select **On-line Characterization**



- 2 The function block **On-line Characterization** dialog appears:
 - Open the **Mode** leaf and double-click in the space next to **Target**
 - Set the Target to **OOS** (Out of Service)
 - Click **End Edit** to set the parameter
- 3 Change the parameters you wish to modify
 - If appropriate, open the parameter leaf and double-click in the space next to the parameter you require
 - Enter the new parameter or select it from the drop-down menu
 - Click **End Edit** to set the parameter
 - Repeat the procedure for all the parameters you wish to modify
- 4 Put the function block back into standard operating mode
 - Open the **Mode** leaf and double-click in the space next to **Target**
 - Set the Target back to the original value (**Auto** (Automatic) or **Cas** (Cascade))
 - Click **End Edit** to set the parameter
 - Check that the **Mode** really changes to the Target Mode (failure to do so indicates a configuration error)
 - Press **Close** to store the values (if you are prompted – answer with **Yes**)
- 5 Click on the **Project View** workspace and **Export Tags...**,
 - Open **Project File**, then press **Save Entire Project** to save the project
- 6 If your changes have any effect on the function block schedule, e.g. modifying execution times, download the modified project
 - In the Project workspace right-click on **Fieldbus Network** and select **Download**

3 Resource Block

Every FOUNDATION Fieldbus device has a resource block comprising all data that uniquely identify and characterize its hardware and software. Since the resource block has contained parameters only, i.e. no block input/output data, it cannot be linked to other blocks.

Resource block parameters can be classified into two groups, informative and operational. Each group is further divided into static and dynamic data.

- Informative parameters provide an overview of the hardware and software implemented in the device. Parameters of this nature include device type, device name, manufacturer ID, serial number. In addition, when on-line, the block also indicates the settings of any hardware switches controlling the write lock and simulation functions.
- Operational parameters influence the execution of the other blocks in the field device. In this case, the resource block acts as a central unit that checks the device status, flags alarms when necessary and controls the operability of the other blocks and hence the device itself.

In the following, only the major parameters are described in detail. Short descriptions of parameters with specialised use and described elsewhere in this manual are to be found in Table 3.8. of Chapter 3.3.

3.1 Informative parameters

3.1.1 Device software

The resource block contains a number of parameters which act as an "electronic" nameplate for the associated device. In addition, there are a number of parameters that indicate or influence the performance of the device. They can be viewed in the **On Line Characterization** dialog.

Parameter	Description
DD_RESOURCE	Tag of the resource containing the device description
MANUFAC_ID	Manufacturer's identification number
DEV_TYPE	Manufacturer's model number associated with the resource
DEV_REV	Manufacturer's revision number associated with the resource
DD_REV	Revision of the DD associated with the resource
MIN_CYCLE_T	Minimum time, in units of $1/32$ ms, that the device requires for function block execution <ul style="list-style-type: none"> ■ Any function block schedule time downloaded to the device must be greater than this value
MEMORY_SIZE	Indicates the total memory in kbytes available in the device for function block configuration
FREE_SPACE	Indicates the percentage of configuration memory that is still available
FREE_TIME	Indicates the approximate percentage of time that the device has left for processing new function blocks, should they be configured
ITK_VER	Version number of the ITK test used as basis for certifying the device
ENP_VERSION	Version number of the Electronic Nameplate
DEVICE_TAG	Device tag downloaded from the project to the device
SERIAL_NUMBER	Manufacturer's serial number for the device
ORDER_CODE	Manufacturer's order code for the device
FIRMWARE_REVISION	Revision number of the firmware of the device

3.1.2 Device hardware

In addition to the "nameplate", the resource block has a number of parameters that indicate the hardware configuration of the device. They can be viewed in the **On Line Characterization** dialog. In some cases it is possible to influence the configuration of the device through an associated parameter.

Parameter	Description
HARD_TYPES	Read only bitstring that indicates the types of hardware that are available to the device <ul style="list-style-type: none"> ■ If an I/O block is configured that requires a type of hardware that is not available to the device, the BLOCK_ERR parameter indicates a configuration error
FEATURES	Read only parameter indicating the hardware and software features available in the device <ul style="list-style-type: none"> ■ See Chapter 3.2.4 for additional information
FEATURE_SEL	Read/Write parameter allowing the selected hardware and software features to be enabled <ul style="list-style-type: none"> ■ Click in the box to activate the feature; multiple selections allowed
WRITE_LOCK	Indicates the position of the hardware jumper controlling the write protection of the device <ul style="list-style-type: none"> ■ LOCKED: device data cannot be modified ■ NOT LOCKED: device data can be modified

3.1.3 Hardware write lock

If the device is provided with a hardware lock, e.g. jumper, switch or magnetic relay, for configuring write protection, the option "Hard Write lock supported" will appear in the **FEATURES** parameter.

After selection of the option "Hard Write lock" in the **FEATURE_SEL** parameter, the parameter **WRITE_LOCK** shows the status of the jumper. The following statuses are possible:

- LOCKED: static and non-volatile parameters cannot be changed. Block connections and calculation results will proceed normally, but the configuration will be locked
- NOT LOCKED: static and non-volatile parameters can be changed

When the jumper is configured for "LOCKED" and "Hard Write lock" is activated in **FEATURE_SEL**, there can be no write actions to parameters in the device. The hard lock stays active until the jumper position is changed.

If the position of the jumper is changed from "LOCKED" to "NOT LOCKED", the discrete alert **WRITE_ALM** is generated at the priority **WRITE_PRI**. Resetting the jumper will clear the alert, if it exists.

Note



- Some devices must be restarted in order to detect a change in the hardware lock, while others do this during normal device operation.

3.1.4 Simulation

Simulation is obligatory in all FOUNDATION Fieldbus devices and is controlled by a hardware lock, e.g. a jumper, switch or magnetic relay. When the hardware lock is in the position "Enable Simulation", the alert "Simulation active" appears in the **BLOCK_ERR** parameter.

Chapter 2.10 contains a full description of simulation.

Note



- Some devices must be restarted in order to detect a change in the simulation lock, while others do this during normal device operation.

3.2 Operational parameters

3.2.1 Block operation

Chapter 2.4 provides a general description of block operation and block parameters.

In the case of the resource block, the **MODE_BLK** parameter **MODE_BLK.Target** controls operation of the entire device. If the parameter is set to "Out of Service" (OOS), all function block execution within the complete device is disabled. The actual mode of all the function blocks within the device are changed to OOS, but their target mode is not changed.

The resource block operates normally when it is in AUTO mode. In some cases, the IMAN mode is also supported: the block assumes this status when the resource is initializing or receiving a software download.

When the block is out of service, the status is flagged with the message "Block status OOS" in the parameter **BLOCK_ERR**.

3.2.2 Resource state

The current operating status of the resource block can be viewed in the **On Line Characterization** dialog through the parameter **RS_STATE**. Depending upon the device, the following states are possible:

State	Description
UNDEFINED	The block is in an invalid state.
START/RESTART	This state is entered on power-up or on restoration of power to a device. In it, the memory and other hardware necessary for reliable operation are tested. If the hardware tests are successful, the initialization state will be entered. Otherwise, the failure state is assumed.
INITIALIZATION	The initialization state is entered from the start/restart or failure states. In it, all unreported function block alarms will be automatically confirmed and acknowledged. Once the system management SM_STATE object is detected to be operational, block execution may be scheduled and the resource state will move to on-line linking.
ON-LINE LINKING	This state is entered from the on-line and initialization states. In it, the status of defined links is evaluated. If all defined links are established, then the resource state move to on-line.
ON-LINE	This is the normal operating status; the resource block is in operating mode AUTO. it is entered from the on-line linking status. The connections configured between the function blocks have been linked. If any connections are missing, the state reverts to on-line linking.
STANDBY	This state is entered if the mode of the resource block is changed to out-of-service (OOS). In it, the actual mode of all function blocks in the resource will be forced to OOS mode. The mode of transducer blocks may not be affected. On a change in the resource block mode to AUTO, the state will revert to start/restart.
FAILURE	This state may be entered from any state except standby. Transition to this state is caused by the detection of a memory or other hardware failure which would prevent reliable operation. The failure may pertain either to the whole device or only to the resource. Based on this state being active, a function block of the output class may change its output to a fault state position. In this state, hardware status will be tested. If the hardware failure clears, then the state will move to initialization.

3.2.3 Resource restart

The **RESTART** parameter offers several options for restarting the device, see the table below. The device is restarted by selecting the restart mode from the drop-down menu in the **On Line Characterization** dialog, when the resource block is out of service. After restart, the **RESTART** parameter returns to the value "Run"

Index	Option	Description
1	Run	Passive state of the parameter
2	Restart resource	Restarts the resource as configured (clears up garbage problems)
3	Restart with defaults	Restarts the resource with default configuration
4	Restart processor	Restarts the processor only
5 – 10	Restart with factory defaults	Restarts with manufacturer specific default values <ul style="list-style-type: none"> ■ If these restarts are not implemented, the parameter defaults to Run

3.2.4 Resource options

The **FEATURES** parameter indicates the resource block options that are supported by the device (see FF Specification 890). The features themselves are enabled by selecting the appropriate option from the drop-down menu of **FEATURE_SEL** parameter in the **Off Line Characterization** dialog. Multiple selections are allowed.

If an option is selected in **FEATURE_SEL** that is not available in **FEATURES**, a block configuration error is flagged in the parameter **BLOCK_ERR**.

Index	Option	Description
0	Unicode strings	Device supports output as unicode strings
1	Reports supported	See Chapter 2.9, Alert handling Device supports alert reports
2	Fault State supported	See Chapter 2.7, Fault State handling Device causes all output blocks to assume the predefined fault state defined in Fault State type I/O action when the physical input or the SET_FSTATE parameter flags a fault.
3	Soft Write lock supported	Device has a software write lock
4	Hard Write lock supported	Device has a hardware write lock
5	Output readback supported	Device reads back the actual valve position to the associated output function block (actuators only)
6	Direct write to output hardware	Device writes one or more hardware values when the resource is dead (manufacturer-specific option)
7	Change of BYPASS in an automatic mode	Device allows the change of bypass when the associated function block is in an automatic mode <ul style="list-style-type: none"> ■ Normally this is only possible when the block is out of service or in manual
8	MVC Report Distribution supported	Device supports multivariable container functionality <ul style="list-style-type: none"> ■ The resource receiving the report must also support MVC
9	MVC Publishing/Subscribing supported	Device supports multivariable container publishing/subscribing <ul style="list-style-type: none"> ■ All resources in the system must also support MVC publishing/subscribing
109	Multi-bit Alarm (Bit-Alarm) Support	Device supports multibit alarm parameters <ul style="list-style-type: none"> ■ When not selected, the device treats multibit alarms as simple alarms ■ For more information, see Chapter 4.4.3.12 of FF Specification 890
11	Restart/Relink required after using FB_Action	Device requires restart/relink after instantiation or deletion of function blocks with the FB_Action service <ul style="list-style-type: none"> ■ For more information see Chapter 4.14.3.6 of FF Specification 890
12	Deferral of Inter-Parameter Write Checks	Disables inter-parameter write checks when the resource and function blocks are out of service or the transducer block is out of serve or in manual mode

3.2.5 Alert reporting

Alert reporting, see Chapter 2.9, is enabled for the entire resource by selecting the option "Reports supported" in the **FEATURE_SEL** parameter of the device resource block. The associated alert priority is set in the corresponding priority parameter **xxx_PRI** of the function block where the alert is generated.

The priority and time stamp of the alert determines the order of processing when more than one alarm is active at a particular time, whereby the oldest alert with highest priority is processed first. Three parameters control the queuing of alerts:

- **MAX_NOTIFY** can be read online and indicates the maximum number of alert reports that the device can send without receiving a confirmation
- **LIM_NOTIFY** allows the user to select the number of alert reports the device can send without receiving a confirmation. If LIM_NOTIFY is set to zero, then no alerts are reported.
- **CONFIRM_TIME** determines the time the resource will wait for confirmation of receipt of a report before trying again. If the CONFIRM_TIME = 0 the device will not retry.

Resource block alerts

For the resource block, two types of alert are of interest:

- Block alarms, which are automatically assigned priority 2
- Write (protection) alarms, which are normally configured as advisory alarms

The parameter **ACK_OPTION** configures the automatic acknowledgement of alerts. Depending on the device, it may be implemented as a tickbox to enable automatic acknowledgement or disable manual acknowledgement in the **xxx_ALM** parameter of the corresponding alert.

- Disabling automatic acknowledgement of the write protection alert, means that the alert must be manually acknowledged in the **WRITE_ALM** parameter
- Disabling automatic acknowledgement of block alarms, means that each alert must be manually acknowledged in the **BLOCK_ALM** parameter

The table below summarizes the parameters to be set to activate alert reporting and setting up of alarm priorities and acknowledgement.

Parameter	Description
ALERT_KEY	Optional index that identifies the plant unit. May be used in the host for sorting alarms, etc.
CONFIRM_TIME	Sets time limit for receipt of alert notification confirmation from interface device. – Setting is multiplied by 1/32 milliseconds to obtain time period – If no confirmation is received within this time, the report is sent again
MAX_NOTIFY	Indicates maximum number of unconfirmed reports that can be stored by the device
LIM_NOTIFY	Sets number of unconfirmed reports that to be stored by the device – LIM_NOTIFY may not exceed MAX_NOTIFY – If LIM_NOTIFY = 0, then no alerts are reported
ACK_OPTION	When unticked, disables auto-acknowledgment in block for each class of alarm. The alarms must then be acknowledged in the salient xxx_ALM parameter. ■ 1: Write Lock Alarm Disabled/Write Lock Alarm Auto Acknowledgement ■ 8: Block Alarm Disabled/Block Alarm Auto Acknowledgement (Resource block only)
WRITE_PRI	Priority of an alarm or event. If no setting is made the acknowledgement is "undefined". ■ 0-1: Associated alert is not sent as a notification; normally auto-acknowledged. ■ 2: Reserved for alerts that do not require the attention of a plant operator. Block alarm and update event have this priority; normally auto-acknowledged. ■ 3-7: Advisory alarms. Alarms of this priority will normally be acknowledged. ■ 8-15: Critical alarms. Alarms of this priority will normally be acknowledged.

3.2.6 Software write lock

If the device supports software write protection, the option "Soft Write lock supported" will appear in the **FEATURES** parameter. If the device also has a hardware write lock and this is active, see Chapter 3.1.2, the corresponding hardware jumper must be configured for "UNLOCKED" before the soft write lock can be activated. In some cases, the device must be restarted before the change in jumper position is registered.

To activate the soft write lock, first select the option "Soft Write lock" in the **FEATURE_SEL** parameter. The write protection can now be activated through the **WRITE_LOCK** parameter. The following statuses can be selected from the drop-down menu:

- LOCKED: static and non-volatile parameters cannot be changed. Block connections and calculation results will proceed normally, but the configuration will be locked
- NOT LOCKED: static and non-volatile parameters can be changed

If **WRITE_LOCK** is changed from LOCKED to NOT LOCKED, the discrete alert **WRITE_ALM** is generated at the priority **WRITE_PRI**. The priority of the alarm determines if it is to be acknowledged or not, see Chapter 2.9.2. Resetting **WRITE_LOCK** will clear the alert, if it exists.

3.2.7 Block errors

The **ALARM_SUM** parameter provides a summary of all alarms active for the resource block. The parameter **BLOCK_ERR** indicates the type of error that is present. If automatic acknowledgement of block alarms is disabled in the **ACK_OPTION** parameter, they must be acknowledged in the **BLOCK_ALM** parameter.

The table below indicates the possible alarms for the Resource Block..

Condition	Reason
Block Configuration Error	<ul style="list-style-type: none"> ■ I/O block has been configured that requires a type of hardware that is not available to the device ■ Feature selected in FEATURE_SEL that is not supported by the device ■ More than one CYCLE_TYPE selected in CYCLE_SEL ■ CYCLE_TYPE selected in CYCLE_SEL not supported by device
Simulate Active	<ul style="list-style-type: none"> ■ The hardware jumper for simulation is set to allow simulation <p>Note: this is not an alarm, but an indication the simulation is possible by selecting "Simulation Enable" in the I/O function block</p>
Device Fault State Set	<ul style="list-style-type: none"> ■ FAULT_STATE is active
Out-of-Service	<ul style="list-style-type: none"> ■ MODE_BLK.Target currently set to OOS

3.3 Block parameters

Resource block

Parameter	Valid range	Default value	Description/Action
ST_REV	–	0	See Chapter 2.3.4
TAG_DESC	–	Blank	
STRATEGY	–	0	
ALERT_KEY	1 – 255	0	
MODE_BLK	–	O/S	See Chapter 2.4
BLOCK_ERR	–		
RS_STATE	–		See Chapter 3.1.2.
TEST_RW	–		Read/write test parameter - used only for conformance testing.
DD_RESOURCE	–	Blank	Tag of the resource containing the device description
MANUFAC_ID	–	–	Manufacturer's identification number
DEV_TYPE	–	–	Manufacturer's model number associated with the resource
DEV_REV	–	–	Manufacturer's revision number associated with the resource
DD_REV	–	–	Revision of the DD associated with the resource
GRANT_DENY		0	See Chapter 2.8.1
HARD_TYPES		–	Types of hardware available as channel numbers.
RESTART	1 – 4		See Chapter 3.2.3
FEATURES	–		See Chapter 3.2.4.
FEATURE_SEL	–	0	
CYCLE_TYPE	–		Identifies the event input types supported by the device, which invoke the execution of the device function blocks <ul style="list-style-type: none"> ■ Scheduled: the blocks execute when a schedule event input is received from the LAS ■ Block execution: the blocks execute when a block execution event input is received from another block that has just completed its execution ■ Manufacturer specific: the blocks execute according to a manufacturer-specific schedule event input
CYCLE_SEL	–	0	Selects the block execution method for the device. <ul style="list-style-type: none"> ■ Endress+Hauser devices require no configuration
MIN_CYCLE_T	–		Indicates the interval, in units of 1/32 ms, at which modified non-volatile parameters are stored to a non-volatile memory <ul style="list-style-type: none"> ■ 0 indicates that parameters are not stored in normal operation ■ For Endress+Hauser devices, non-volatile parameters are stored on power down only
MEMORY_SIZE	–		Available configuration memory (kbytes) in empty resource. <ul style="list-style-type: none"> ■ Check before attempting a download.
NV_CYCLE_T	–		Interval between writing copies of NV parameters to non-volatile memory (1/32 ms). Zero means never.
FREE_SPACE	0 to 100		Percent of memory available for further configuration. Zero in a preconfigured resource.
FREE_TIME	0 to 100		Percent of the block processing time that is free to process additional blocks.
SHED_RCAS	–	640000	See Chapter 2.4.4 Timeout, in units of 1/32 ms, for communication with an external application writing to function block RCas locations <ul style="list-style-type: none"> ■ If the timeout is exceed, communication is assumed to be lost and the mode is shed, see Chapter 2.4.4 ■ If the timeout is 0, mode shedding is disabled
SHED_ROUT	–	640000	See Chapter 2.4.4 Timeout, in units of 1/32 ms, for communication with an external application writing to function block ROut locations <ul style="list-style-type: none"> ■ If the timeout is exceed, communication is assumed to be lost and the mode is shed, see Chapter 2.4.4 ■ If the timeout is 0, mode shedding is disabled
FAULT_STATE	–		Indicates whether the output function blocks of the device are being forced into failsafe mode, see Chapter 2.6 <ul style="list-style-type: none"> ■ 1: Blocks in normal operation ■ 2: Blocks being forced into failsafe mode (see SET_FSTATE)

Parameter	Valid range	Default value	Description/Action
SET_FSTATE	1, 2	1	Forces all output blocks of the device into their failsafe mode. ■ 1: Off, ■ 2: Enable force
CLR_FSTATE	1, 2	1	Disables the force of failsafe mode caused by SET_FSTATE ■ 1: Off, ■ 2: Disable force
MAX_NOTIFY	RO		See Chapter 3.2.5
LIM_NOTIFY	0 to MAX_NOTIFY	MAX_NOTIFY	
CONFIRM_TIME		640000	
WRITE_LOCK	1,2	1	See Chapters 3.1.2, 3.2.4
UPDATE_EVT			Alert generated by any change in static data, see Chapter 2.9.4.
BLOCK_ALM			See Chapter 3.2.7
ALARM_SUM			The current alert status, unacknowledged states, unreported states, and disabled states of the alarms associated with the function block.
ACK_OPTION	0, 1	0	See Chapter 3.2.5
WRITE_PRI	0 to 15	0	Priority of the alarm generated by clearing the write lock.
WRITE_ALM			Alert is generated if the write lock parameter is cleared.
ITK_VER			Version number of the ITK test used as basis for certifying the device
ENP_VERSION			Version number of the Electronic Nameplate
DEVICE_TAG			Device tag downloaded from the project to the device
SERIAL_NUMBER			Manufacturer's serial number for the device
ORDER_CODE			Manufacturer's order code for the device
FIRMWARE_REVISION			Revision number of the firmware of the device
MS_RESOURCE_DIRECTORY			Array describing the grouping of the enhanced parameters (not relevant to operation)
Legend: RO - Read only			

4 Transducer Blocks

A transducer block acts as a "buffer" between a measuring or actuating device and the function block application process. Its purpose is to:

- encapsulate all parameters influencing the physical set up of the device
- for measuring devices, provide a standardized signal to an analog input block
- for actuating devices, process the standardized signal provided by an analog output block

In addition to measuring and actuating devices, transducer blocks are also required for local I/O modules of the ControlCare Field Controller.

Measuring devices

Fig. 4-1 shows the how the transducer block functions in simple measuring device. The measuring signal of the device is converted to a process value by the transducer block and transmitted to the analog (or discrete) input block. Here the process value can be scaled or limits can be set before it is made available as the **IN** value to a control loop.

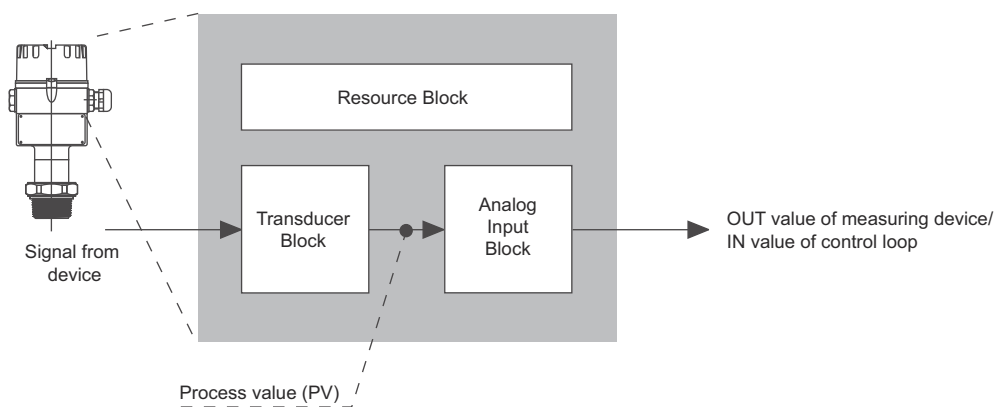


Fig. 4-1: Function of transducer block in an analog measuring device

The **CHANNEL** parameter links the transducer and analog input blocks. In the case of multivariable devices, each process variable is assigned a different channel number and the device provides an equal number of input blocks for use in the control strategy.

Actuating devices

For an actuator, see Fig. 4-2, the analog (or discrete) output block receives a setpoint value from the control loop, which may run centrally in the controller or distributed in the field devices. After scaling, e.g. to a current signal range, the setpoint value is transmitted to the transducer block as the output value of the function block. The transducer block now generates the physical signal that drives the valve to the desired position.

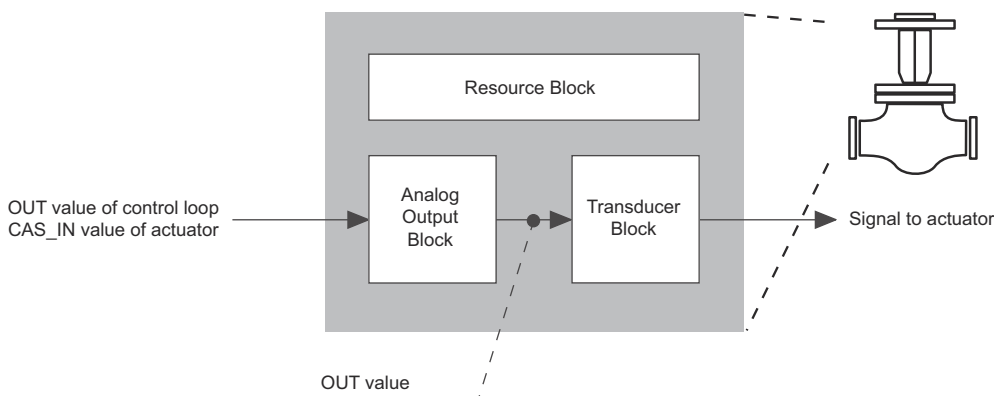


Fig. 4-2: Function of transducer block in an analog actuating device

4.1 Hardware Configuration block

The Hardware Configuration block configures the local I/O module type for each slot in a ControlCare Field Controller rack. In ControlCare Application Designer it is assigned the generic name **Device Tag-HC-n** and can be found by expanding the controller leaf in the fieldbus or process cell window.

In the following, only the major parameters are described in detail. Short descriptions of parameters with specialised use are to be found in Table 4.3. of Chapter 4.1.3.

4.1.1 Block configuration

MODE_BLK.Target

For normal operation, **MODE_BLK.Target** is set to Auto. This is set during offline configuration and comes into effect when the control strategy is downloaded to the controller.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

IO_TYPE_Rxx SLOT_x

The physical arrangement of the modules on the rack assemblies with the addresses 0 to 14 is declared in the parameters **IO_TYPE_R0** to **IO_TYPE_R14** respectively. Each **IO_TYPE_Rx** parameter has four slot parameters, **SLOT_0** to **SLOT_3**, where the module type is entered. The I/O types together with the permissible function blocks for each type are listed below. **IO_TYPE_R0** (Field Controller) is preset to "Not I/O" and should not be changed.

Designation	Description	I/O type	Function Block
	Available slot	No I/O	None
SFC050	Power Supply 90-264 VAC	No I/O	None
SFC056	Power Supply for Backplane 20-30 VDC	No I/O	None
SFC162	FF Field Controller, 1x100 Mbps, 4xH1	No I/O	RS/HC/DIAG
SFC173	PROFIBUS Field Controller, 1x100 Mbps, 4xH1	No I/O	RS/HC/DIAG
SFC252	AC Power Supply for Fieldbus	No I/O	None
SFC260	DC Power Supply for Fieldbus	No I/O	None
SFC353	4-channel Power Supply Impedance	No I/O	None
SFC411	2 Groups of 8 24 VDC Inputs (Isolated)	16-discrete input	MDI/DI
SFC415	2 Groups of 8 24 VDC Inputs (Sink, Isolated)	16- discrete input	MDI/DI
SFC420	1 Group of 8 On/Off Switches	8- discrete input	MDI/DI
SFC428	2 Groups of 8 NO Relays Outputs	16- discrete output	MDO/DO
SFC432	1 Group of 8 24 VDC Inputs and 1 Group of 4 NO Relays	8- discrete input/ 4- discrete output	MDI/DI MDO/DO
SFC435	1 Group of 8 24 VDC Inputs and 1 Group of 4 NC Relays	8- discrete input/ 4- discrete output	MDI/DI MDO/DO
SFC438	1 Group of 8 24 VDC Inputs and 1 Group of 2 NO and 2 NC Relays	8- discrete input/ 4- discrete output	MDI/DI MDO/DO
SFC441	2 Groups of 8 pulse inputs - low frequency	16-pulse input	PUL
SFC442	2 Groups of 8 pulse inputs - high frequency	16-pulse input	PUL
SFC444	1 Group of 8 analog inputs with shunt resistors	8-analog input	MAI/AI
SFC445	1 Group of 8 temperature inputs	8-temperature	MAI/AI
SFC446	1 Group of 4 analog output	4-analog output	MAO/AO
SFC457	1 Group of 8 differential analog inputs with shunt resistors	8-analog input	MAI/AI
SFC467	2 Groups of 8 pulse inputs - AC	16-pulse input	PUL

Tab. 4-1: I/O types with associated function blocks

Addressing and slot numbers are explained in Chapters 3.5.3 and 3.5.4 of the Field Controller Hardware manual, BA021S/04/en; the procedure for configuring the block in Chapter 5.2.1 of Field Controller Commissioning manual BA35S/04/en. For direct mapping of I/Os to the embedded hybrid function block see the I/O Mapping Tool manual, BA032S/04/en.

4.1.2 Block errors

On execution of the HC configuration block, the Field Controller writes to all declared output modules and reads all declared input modules. If any I/O module has failed in this scan, it is indicated in **BLOCK_ERR** and the **MODULE_STATUS_x** parameters. Other possible block errors are listed in the table below:

Condition	Reason
Block Configuration Error	<ul style="list-style-type: none"> Configured I/O type not present in controller I/O type present in controller that has not been configured in HC block Configured I/O type does not correspond to that present in controller
Out-of-Service	<ul style="list-style-type: none"> MODE_BLK.Target currently set to OOS

4.1.3 Block parameters

The table below gives an overview of the HC block parameters

Hardware Configuration block

Parameter	Valid range/ Options	Default value	Description/Action
ST_VER		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY	0 to 255	0	
ALERT_KEY	1 to 255	1	
MODE_BLK	TARGET	O/S	Set to AUTO
BLOCK_ERR	0 to 15		See Chapter 4.1.2
REMOTE_IO	RO		When on-line, indicates Remote I/O master (MODBUS applications only)
IO_TYPE_R0		0	Module type for rack 0, slots 0 to 3 Preset to "Not I/O" - do not change
IO_TYPE_R1		0	Select module type for rack 1, slots 0 to 3
IO_TYPE_R2		0	Select module type for rack 2, slots 0 to 3
IO_TYPE_R3		0	Select module type for rack 3, slots 0 to 3
IO_TYPE_R4		0	Select module type for rack 4, slots 0 to 3
IO_TYPE_R5		0	Select module type for rack 5, slots 0 to 3
IO_TYPE_R6		0	Select module type for rack 6, slots 0 to 3
IO_TYPE_R7		0	Select module type for rack 7, slots 0 to 3
IO_TYPE_R8		0	Select module type for rack 8, slots 0 to 3
IO_TYPE_R9		0	Select module type for rack 9, slots 0 to 3
IO_TYPE_R10		0	Select module type for rack 10, slots 0 to 3
IO_TYPE_R11		0	Select module type for rack 11, slots 0 to 3
IO_TYPE_R12		0	Select module type for rack 12, slots 0 to 3
IO_TYPE_R13		0	Select module type for rack 13, slots 0 to 3
IO_TYPE_R14		0	Select module type for rack 14, slots 0 to 3
MODULE_STATUS_R0_3	RO		Status of modules in rack 0-3
MODULE_STATUS_R4_7	RO		Status of modules in rack 4-7
MODULE_STATUS_R8_11	RO		Status of modules in rack 8-11
MODULE_STATUS_R12_14	RO		Status of modules in rack 12-14
UPDATE_EVT			Alert generated by any change in static data, see Chapter 2.9.4.
BLOCK_ALM			See Chapter 2.9
HFB_STATUS_R0_3	RO		Status of modules in rack 0-3 assigned to Hybrid FB
HFB_STATUS_R4_7	RO		Status of modules in rack 4-7 assigned to Hybrid FB
HFB_STATUS_R8_11	RO		Status of modules in rack 8-11 assigned to Hybrid FB
HFB_STATUS_R12_14	RO		Status of modules in rack 12-14 assigned to Hybrid FB
Legend: RO - Read only			

4.2 Temperature Transducer block

A temperature transducer block is required for the every SFC445 temperature module in a Field Controller rack. The module has eight input channels for RTD, TC, mV or Ohm. In ControlCare Application Designer the temperature block is assigned the generic name **Device Tag-TEMP-n** and can be found by expanding the controller leaf in the fieldbus or process cell window.

In the following, only the major parameters are described in detail. Short descriptions of parameters with specialised use are to be found in Table 4.5. of Chapter 4.2.3.

4.2.1 Block configuration

The block mode, channel, sensor connection and sensor type at each input channel must be configured for each module.

MODE_BLK.Target

For normal operation, **MODE_BLK.Target** is set to AUTO. This is set during offline configuration and comes into effect when the control strategy is downloaded to the controller.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

CHANNEL

The **CHANNEL** parameter is set to the rack number and slot, where the SFC445 is installed, e.g. for Rack 1, Slot 2, all inputs = Rack+Slot+09 = 1209 in accordance with the instructions in Chapter 5.1.1.

SENSOR_TYPE_x SENSOR_ CONNECTION_x

The type of temperature sensing element connected to the each input channel of the SFC445 and the method of connection are declared in the parameters:

- **SENSOR_TYPE_x**: select from the drop-down menu
- **SENSOR_CONNECTION_x**: select from the drop-down menu

If the **ALL** button in the **Off Line Characterization** dialog is active, the parameters are to be found under **VALUE_RANGE_x** in the reverse order.

When on-line the following parameters provide information on the status and configuration of each input:

- **TEMP_x**: value and status of the temperature measured at the input point
- **VALUE_RANGE_x**: copy of the XD_SCALE parameter of the AI Block connected to the input point

The **VALUE_RANGE_x** parameter can be viewed by pressing the **ALL** button in the **On Line Characterization** dialog.

4.2.2 Block errors

On execution of the Temperature Transducer block, the Field Controller checks the **CHANNEL** and **SENSOR_TYPE/SENSOR_CONNECTION** pairs for correct configuration. It is not able, however, to check whether the physical sensor type or connection corresponds to that configured. Block errors are indicated in the **BLOCK_ERR** parameter.

Condition	Reason
Block Configuration Error	<ul style="list-style-type: none"> ■ No channel no set or channel number does not end in 09 ■ One of the two parameters SENSOR_TYPE_x or SENSOR_CONNECTION_x is missing for a particular output
Out-of-Service	<ul style="list-style-type: none"> ■ MODE_BLK.Target currently set to OOS

4.2.3 Block parameters

The table below gives an overview of the Temperature Transducer block parameters

Temperature Transducer block

Parameter	Valid range/ Options	Default value	Description/Action
ST_VER		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY	0 to 255	0	
ALERT_KEY	1 to 255	1	
MODE_BLK	TARGET	O/S	Set to AUTO
BLOCK_ERR	0 to 15		See Chapter 4.2.2
CHANNEL.		0	Enter rack and slot number of the associated SFC445 module coded as per Chapter 5.3.1, whereby G = 0 and P = 9 (whole)
TEMP_0...TEMP_7	RO		When on-line, indicates temperature value and status at input point 0...7
VALUE_RANGE_0.... VALUE_RANGE_7	RO	0-100%	When on-line and connected to an Analog Input block, shows a copy of AI block's XD_SCALE. parameter
SENSOR_CONNECTION_0. .. SENSOR_CONNECTION_7	1, 2, 3	1	Select type of sensor connection for points 0...7 1 = differential, 2 = 2-wire, 3 = 3-wire
SENSOR_TYPE_0 ... SENSOR_TYPE_7	menu	Pt100	Select type of sensor connected to points 0...7 (Select from pull-down menu)
UPDATE_EVT			Alert generated by any change in static data, see Chapter 2.9.4.
BLOCK_ALM			See Chapter 2.9
Legend: RO - Read only			

4.3 Device transducer blocks (Endress+Hauser)

Device transducer blocks contain all parameters required to physically set up the measurement and present the result to the "outside" world. Thus they usually contain parameters associated with calibration, signal processing (square-root, linearization, inversion, cut-offs), output settings, safety settings etc. Depending on type, the measuring device may have a single or several transducer blocks.

Currently, there are no profiles for the various types of device, although work is going on to produce the first specifications in the near future. As a consequence, the user is faced with a variety of implementations.

- For older devices, the device is configured using manufacturer-specific parameters that are to be found at the end of the parameter list. In some cases, these parameters can be changed only when on-line and only after the entry of an enabling code.
- More recent devices, e.g. the TMT162 temperature measuring device, already follow Fieldbus Foundation guidelines and use a quasi-standard parameter set. In this case, no manufacturer-specific parameters are required and the device can usually be configured offline.

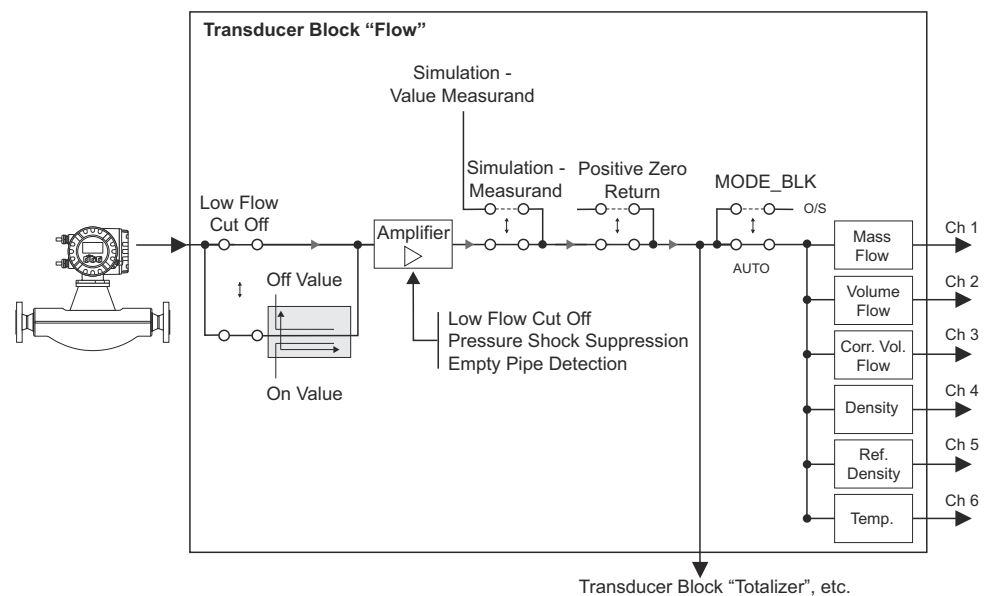


Fig. 4-3: Schematic diagram of Flow Transducer block of Promass 83 Coriolis flowmeter

Example of a device transducer block

Fig. 4-3 shows the example of the Endress+Hauser Promass 83 which has two transducer blocks as standard (flow and totalizer) and a number of others which are available for specific variants of the device. It can be seen that the flow transducer block allows the switching on and off of the functions "Low Flow Cut Off", "Pressure Shock Suppression" and "Empty Pipe Detection". "Low Flow Cut Off" requires the entry of an "Off" and an "On" value to complete configuration. The block also supports simulation of the measurand and positive zero return.

The **MODE_BLK** parameter is to be found, along with other standard parameters, in every device transducer block. The process values measured by the device, in this case six, are presented at an equal number of channels. Thus volume flow is made available to a connected Analog Input block when the block's **CHANNEL** parameter is set to "2". More information on the Promass 83 can be found in Device Function manual BA066D/06/en.

4.3.1 Block configuration

For more information on configuration of the transducer blocks of the individual devices, please consult the associated device manuals

MODE_BLK.Target

For normal operation, **MODE_BLK.Target** is set to AUTO. This is set during offline configuration and comes into effect when the control strategy is downloaded to the controller.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

When **MODE_BLK.Target** is set to MAN and block simulation is enabled, the process value output by the block can be entered manually.

Methods

The Foundation Fieldbus specification provides for the use of so-called methods to simplify the configuration of the device. A method is an interactive sequence of steps that must be followed in order to obtain a particular function from the device. Thus, for example, the steps for the basic calibration of a level device: reset, empty calibration, full calibration might comprise a method named "Basic calibration". The user can call up this method to calibrate the device. He need do nothing else but supply the information which the method asks for as it progresses through each step. The setting of the block mode, reading, writing and plausibility checking of the parameters etc. is automatically done by the program.

A method is part of the device description supplied with the device. Some Endress+Hauser devices support methods.

Block errors

On execution of the block, the device checks for correct configuration. Block errors are indicated in the **BLOCK_ERR** parameter.

Condition	Reason
Block Configuration Error	■ See associated device operating manual
Simulate Active	■ The current transducer block value has been entered manually
Input Failure/ process variable has BAD status	■ Device failure or device initiating etc.
Out-of-Service	■ MODE_BLK.Target currently set to OOS

4.3.2 Block parameters

The table below gives an overview of the common Device Transducer block parameters

Device transducer block

Parameter	Valid range/ Options	Default value	Description/Action
ST_VER		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY	0 to 255	0	
ALERT_KEY	1 to 255	1	
MODE_BLK	TARGET	O/S	Set to AUTO; Set to MAN to enter parameter by hand
BLOCK_ERR	0 to 15		See Chapter 4.3.2
Device specific parameters.		0	See device operating manual
UPDATE_EVT			Alert generated by any change in static data, see Chapter 2.9.4.
BLOCK_ALM			See Chapter 2.9
Legend: RO – Read only			

4.4 Display block (DISP)

The Display transducer block is provided by all instruments that support the use of an integral or pluggable display module. In addition to the common block parameters, it contains parameters which allow the on and off-line configuration of the display. When on-line, it is possible to read the selected variables.

Depending on device, the block is either automatically generated on creation of the device or must be created separately via the **New Block...** context menu. In this case, it is also possible that the generic name is **DEVICE_TAG_BLK_x**.

4.4.1 Block configuration

MODE_BLK.Target

For normal operation, **MODE_BLK.Target** is set to AUTO. This is set during offline configuration and comes into effect when the control strategy is downloaded to the controller.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Configuration parameters

The configuration parameters provided depend upon the device providing the block. Normally the user can configure:

- The source of the displayed value
- Accompanying text
- Number of decimal points
- Max and minimum value of bargraph, if provided
- Alternation time for multiple value displays

In addition, advanced functions such as display lighting and brightness might be influenced. The appropriate device manual should be consulted.

4.4.2 Block parameters

The table below gives an overview of the common Display transducer block parameters

Display transducer block

Parameter	Valid range/ Options	Default value	Description/Action
ST_VER		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY	0 to 255	0	
ALERT_KEY	1 to 255	1	
MODE_BLK	TARGET	O/S	Set to AUTO
BLOCK_ERR	0 to 15		See Chapter 4.3.2
UPDATE_EVT			Alert generated by any change in static data, see Chapter 2.9.4.
BLOCK_ALM			See Chapter 2.9
TRANSDUCER_TYPE			Transducer type to which the display block is connected
XD_ERROR			Transducer Block Errors, see Chapter 2.9.4
Device specific parameters.		0	See device operating manual
Legend: RO – Read only			

4.5 Diagnosis block (DIAG)

The Diagnosis transducer block provides information on the hardware and software features provided by the device. Some devices offer an additional Advanced Diagnosis transducer block which can be used if this option has been implemented in the device.

4.5.1 Block configuration

MODE_BLK.Target

For normal operation, **MODE_BLK.Target** is set to AUTO. This is set during offline configuration and comes into effect when the control strategy is downloaded to the controller.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Configuration parameters

The configuration parameters provided depend upon the device providing the block. For the SFC162 Field Controller, for example, it contains:

- Online measurement of block execution time
- Hardware revision
- Firmware revision
- Serial number of device
- Serial number of main board
- A parameter BEHAVIOR which defines which initial values for parameters will be used after a block instantiation.

For other devices it might contain locking statuses and access codes. The appropriate device manual should be consulted.

4.5.2 Block parameters

The table below gives an overview of the common Diagnosis transducer block parameters

Diagnosis transducer block

Parameter	Valid range/ Options	Default value	Description/Action
ST_VER		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY	0 to 255	0	
ALERT_KEY	1 to 255	1	
MODE_BLK	TARGET	O/S	Set to AUTO
BLOCK_ERR	0 to 15		See Chapter 4.3.2
UPDATE_EVT			Alert generated by any change in static data, see Chapter 2.9.4.
BLOCK_ALM			See Chapter 2.9
TRANSDUCER_TYPE			Transducer type to which the display block is connected (optional)
XD_ERROR			Transducer Block Errors, see Chapter 2.9.4 (Optional)
Device specific parameters.		0	See device operating manual
Legend: RO – Read only			

5 Input Blocks

5.1 CHANNEL parameter

Field Controller

The **CHANNEL** parameter connects the transducer block to the input function block of the device. For the Field Controller I/O modules, this is compiled according to the following hierarchy:

Level	Range
Rack (R)	0 – 14
Slot (S)	0 – 3
Group (G)	0 – 1
I/O Point (P)	0 – 7 9 = whole group

The value in the **CHANNEL** parameter is given by the value RSGP:

- Rack assembly address (R):
Each rack assembly has a unique address between 0 and 14.
The rack assembly with the address "0" cannot be used by the I/O modules.
- Slot (S): Each rack assembly has four slots. One slot supports one I/O module.
Slots are numbered from 0 (first slot) to 3 (last slot).
- Group (G): Ordinal number of group in the specified I/O module
Groups are numbered from 0 (first group) to number of groups minus 1.
- Point (P): Ordinal number of I/O point in a group
I/O points are numbered from 0 (first I/O point) to 7 (last I/O point in the group)
9 means the whole group of points and is used when a module is connected to a MAI, MDI, MAO or MDO function block.

Thus, a single I/O point in the Field Controller may be identified by specifying the rack (R), slot (S), group (G) and point (P). As the **CHANNEL** parameter in the multiple I/O blocks (MIO) must specify the whole group (8 points), the point must be 9, meaning the whole group. For example:

- The **CHANNEL** parameter 1203 means rack 1, slot 2, group 0 and I/O point 3.
- The **CHANNEL** parameter for a MAI block 10119 means rack 10, slot 1, group 1 and I/O point 9, i.e. the whole group.

MODE_BLK	5	RD
BLOCK_ERR	6	RD
CHANNEL	10119	7 PW
OUT_1	8	
OUT_2	9	

Provided that the HC transducer block is configured before the **CHANNEL** parameter, the system verifies that the I/O type configured in the HC block is suitable for block type. If the **CHANNEL** parameter of the AI block accesses an I/O of a different type, the setting will be rejected during download. In this case the channel is set to "0" and a corresponding error is set in the **BLOCK_ERR** parameter, which can be viewed online.

FF devices

The **CHANNEL** parameter of a FF device Input block determines which process value provided by the device is to be connected to the block. More information is to be found in the device manuals.

PROFIBUS devices

In the case of PROFIBUS devices, there is no **CHANNEL** parameter in I/O function blocks. Instead a function block is created for each of the parameters selected during device configuration in PROFIBUS Configurator. If the GSD uses generic block names, e.g. "AI", and the device has several process values, an AI block must be created for each value required. The "FREE SPACE" parameter can be used to ensure that the correct value is assigned to the block, i.e. if values 1 and 5 are required, the configuration AI, FREE SPACE, FREE SPACE, FREE SPACE, AI is created. Application designer will then create the blocks **Device Tag PB-AI-1** and **Device Tag PB-AI-2**.

5.2 Analog Input (AI)

The Analog Input block characterizes the incoming analog signal from a device transducer, temperature or hardware configuration block. The result of execution is the **OUT** parameter. In ControlCare Application Designer it is assigned the generic name **Device Tag-AI-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 5-1.

Note



- Information on the PROFIBUS Analog Input block can be found in Chapter 9.2

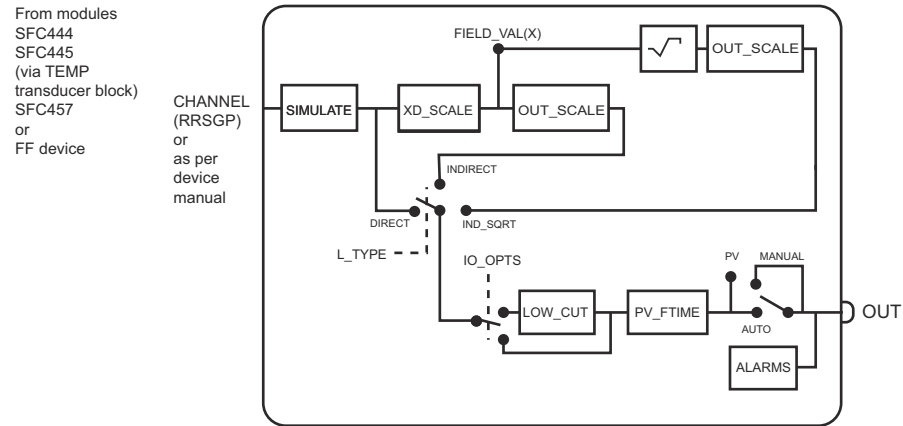


Fig. 5-1: Schematic diagram of the analog input block

5.2.1 Functional description

CHANNEL

The **CHANNEL** parameter determines the transducer block logical channel that is to be connected to the Analog Input block, see Chapter 5.1, and thus the process value to be used by the block. If more than one process value from a device is required for the control strategy, an Analog Input block must be created for each one.

L_TYPE

The linearization parameter **L_TYPE** determines how the process value from the transducer block will be scaled in the input block. One of the following options must be set:

- Direct**
no scaling, i.e. the process value is the **OUT** value.
- Indirect**
linear scaling, i.e. the process value is scaled by **XD_SCALE** and **OUT_SCALE** before it becomes the **OUT** value.
- Indirect Sq Root**
square root scaling, i.e. the **OUT** value is the square root of the process value after scaling in **XD_SCALE** and **OUT_SCALE**. This option is used for Δp flow measurement.

Note



- If a device transducer block already provides or has been configured to provide a square root output, then only the direct or indirect options should be used.

XD_SCALE

XD_SCALE is used prior to output scaling to convert the raw process value from the transducer block to a process value expressed as percentage of range. The result can be read from the **FIELD_VAL** parameter. The **XD_SCALE** parameter expands to reveal the following elements (the example illustrates the scaling of a current input with range limits of 4 mA and 20 mA):

- EU_100 is the upper range limit of the input signal, e.g. 20
- EU_0 is the lower range limit of the input signal, e.g. 4
- UNIT_INDEX is the unit of the input signal, e.g. mA
- DECIMAL is the number of figures behind the decimal point

Examples of input ranges for temperature sensing elements associated with the Temperature Transducer block are to be found in Chapter 5.2.3.

OUT_SCALE

OUT_SCALE is used to scale the process value to the range and units required by any follow-up block. The result is the **OUT** value. The **OUT_SCALE** parameter expands to reveal the following elements (the example illustrates the scaling to an output value range limits of 0% and 100%):

- EU_100 is the upper range limit of the output signal, e.g. 100
- EU_0 is the lower range limit of the output signal, e.g. 0
- UNIT_INDEX is the unit of the output signal, e.g. %
- DECIMAL is the number of figures behind the decimal point

IO_OPTS and LOW_CUT

For flow measurement, a low-flow cut-off limit can be set which causes the **OUT** value to be set to zero when it falls below the value entered in **LOW_CUT**.

- Activate the function by selecting the **Low Cut-Off** option in the **IO_OPTS** parameter.
- Enter the cut-off value in percent of **OUT_SCALE** in the **LOW_CUT** parameter.

PV_FTIME

The **PV_FTIME** parameter allows the output to be damped by entering a suitable positive value in seconds. The default value is zero.

**ALARM_HYS,
XXX_PRI
XXX_LIM**

The Analog Input block supports alarming of the **OUT** value using the parameters **ALARM_HYS**, **XXX_PRI** and **XXX_LIM**, in which global hysteresis, alarm priority and alarm values can be set individually for HI_HI_, HI_, LO_ and LO_LO_ limits. A full description of the function is to be found in Chapter 2.9.

**FSAFE_TYPE
FSAFE_VAL**

The latest Endress+Hauser devices have an enhanced AI function block with two extra parameters **FSAFE_TYPE** and **FSAFE_VAL**. When configured, these dictate how the block reacts on failure.

FSAFE_TYPE determines the value to be output on block failure:

- Fail Safe Value: the block outputs the value **FSAFE_VAL** with the last valid status
- Last Good Value: the block outputs last good value with the last valid status
- Wrong Value: the block outputs last good value with Bad status

If the first option is used, a value with the same engineering units as **OUT_SCALE** must be entered in **FSAFE_VAL**.

5.2.2 Block Configuration

The basic configuration of an Analog Input block is shown in the table below for a TMT162 temperature transmitter and a Promass 83F mass flowmeter. A full description of the configuration is to be found in the Operating Instructions

- BA224REN for TMT162
- BA065DEN for Promass 83F

The range limits for the TMT162 temperature transmitter are determined by the transducer block parameters SENSOR_TYPE and PRIMARY_OUTPUT_TYPE. For SENSOR_TYPE = Pt100 and PRIMARY_OUTPUT_TYPE = SV_1 the transducer block outputs a temperature signal in the range -200°C to +850°C, see Chapter 5.2.4. The XD_SCALE and OUT_SCALE parameters generate the OUT value of the Analog Input block from any part of this range, in our case -50°C to 150°C.

Parameter	Function	Temperature TMT162	Flow Promass 83F
MODE BLOCK/TARGET	Normal operating mode of block	Auto	Auto
XD_SCALE/EU_100* XD_SCALE/EU_0 XD_SCALE/UNITS_INDEX	Upper range value for process variable Lower range value for process variable Unit of process variable	150 (max.850) -50 (min. -200) °C	18 0 t/h (unit)
OUT_SCALE/EU_100 OUT_SCALE/EU_0 OUT_SCALE/UNITS_INDEX	Upper range limit for output variable Lower range limit for output variable Unit of output variable	100 0 %	100 0 %
CHANNEL	Output channels of Transducer Block assigned to Analog Input Block. <ul style="list-style-type: none"> ■ TMT = Primary, RJ or Sensor value 1/2 depending on whether one or two sensors are connected ■ Promass = 1 to 7, depending on which type of process value is required 	Sensor Value 1	1 = mass flow
L_TYPE	Selects the type of linearisation for the input value. <ul style="list-style-type: none"> ■ Direct: PV value = OUT value, Identical XD_SCALE and OUT_SCALE ■ Indirect: PV value scaled to OUT value ■ Indirect Square Root: as Indirect but scaling with root function 	Indirect	Indirect
PV_FTIME	Output damping constant (in seconds).	1	1
FSAFE_TYPE	Selects fail safe type to be output on block failure	Fail Safe Value	n/a
FSAFE_VAL	Sets value for FSAFE_TYPE option Fail Safe Value	0	n/a

Order of parameters

Some block parameters have a write check based on the value of others parameters. It is therefore important to set the parameters in the order shown in Table 4-1 (the same order in which they are displayed in the **Off Line Characterization** dialog.

After characterization of the block, the parameters will appear in the FOUNDATION Fieldbus tree. If you find a parameter in the wrong position, it can be moved by dragging and dropping to the correct one.

5.2.3 Block operation

Mode

The block normally operates in AUTO mode. When online, the status and value of block output is displayed in the **OUT** parameter. The value and status of the signal from the transducer block can be viewed in the **FIELD_VAL** parameter. Any limit alarms are displayed in the **ALARM_SUM** and, if necessary, can be acknowledged in the **HI_HI_ALM** etc. parameters.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

When **MODE_BLK.Target** is set to MAN, the block output can be manipulated by entering the desired status and value in the **OUT** parameter. The **PV** parameter indicates the correct state.

SIMULATE

The process value in the Analog Input block can be simulated in order to check the function of it and subsequent blocks during commissioning. A prerequisite for this is that the hardware lock of the device is set to enable. When simulation is enabled, "Simulation Active" appears in the resource block parameter **BLOCK_ERROR** by , see Chapter 2.9 and Chapter 3.2.7.

Simulation parameters can be changed when the block is in AUTO mode. Expand the **SIMULATE** parameter to enter a simulated value:

- Set the status of the simulated variable in **SIMULATE_STATUS**
- Set the value of the simulated variable in **SIMULATE_VALUE**
- Set **ENABLE_DISABLE** to "Active" to start simulation

The parameters **TRANSDUCER_STATUS** and **TRANSDUCER_VALUE** indicate the status and value of the raw process value. The block output **OUT** reflects the simulated parameters after processing in the block.

Simulation in an I/O block is disabled by setting **ENABLE_DISABLE** to "Disable". In this state **SIMULATE_STATUS** and **SIMULATE_VALUE** etc. track **TRANSDUCER_STATUS** and **TRANSDUCER_VALUE** in order to provide a bumpless transfer from disabled to enabled.

Status propagation

The Analog Input block supports the qualities Bad, Uncertain and GoodNC together with the associated substatures, see Chapter 2.3.2. For example, if the OUT value exceeds the OUT_SCALE range and no worse condition exists in the block then the OUT status will be "Uncertain, EU Range Violation".

Status propagation can be influenced by selecting the following options in the STATUS_OPTS parameter, see Chapter 2.8.3.

Option	Description
Propagate Fail Forward	Propagates a transducer block status Bad.DeviceFailure or Bad.SensorFailure to OUT without generating an alarm. – Allows the user to determine whether alarming (sending of an alert) will be done by the block or propagated downstream
Uncertain if Limited	Sets the output status of the block to Uncertain when limit alarms HI_HI_, HI_, LO_ or LO_LO_ are violated – Option when alarms have been set on the output value
BAD if Limited	Sets the output status of the block to Bad when limit alarms HI_HI_, HI_, LO_ or LO_LO_ are violated – Option when alarms have been set on the output value
Uncertain if Man	Sets the output status of the block to Uncertain if the actual mode of the block is Man

Block errors

On execution of the block, the device checks for correct configuration. Block errors are indicated in the **BLOCK_ERR** parameter.

Condition	Reason
Block Configuration Error	<ul style="list-style-type: none"> CHANNEL or L_TYPE parameter has an invalid value CHANNEL and HC transducer block configuration not compatible XD_SCALE does not have a suitable engineering unit or range for the sensor type of the transducer block XXX_LIM limit value is out of valid range
Simulate Active	<ul style="list-style-type: none"> The current OUT value and status is simulated
Input Failure/ process variable has BAD status	<ul style="list-style-type: none"> I/O module or device failure, device initiating etc.
Out-of-Service	<ul style="list-style-type: none"> MODE_BLK.Target currently set to OOS L_TYPE is "Uninitialised": set to one of other options

5.2.4 SFC445 temperature ranges (XD_SCALE)

The tables below indicate the valid temperature ranges for XD_SCALE as a function of sensing element type.

RTDs

RTD	2-wire or 3-wire		Differential	
	Range °C	Range °F	Range °C	Range °F
Cu10 GE	–20 to 250	–4 to 482	–270 to 270	–486 to 486
Ni 120 DIN	–50 to 270	–58 to 518	–320 to 320	–576 to 576
Pt50 IEC	–200 to 850	–328 to 1562	–1050 to 1050	–1890 to 1890
Pt100 IEC	–200 to 850	–328 to 1562	–1050 to 1050	–1890 to 1890
Pt500 IEC	–200 to 450	–328 to 842	–650 to 650	–1170 to 1170
Pt50 JIS	–200 to 600	–328 to 1112	–800 to 800	–1440 to 1440
Pt100 JIS	–200 to 600	–328 to 1112	–800 to 800	–1440 to 1440

Thermocouples

Thermocouples	2-wire or 3-wire		Differential	
	Range °C	Range °F	Range °C	Range °F
B NBS	+100 to 1800	+212 to 3272	–1700 to 1700	–3060 to 3060
E NBS	–100 to 1000	–238 to 1832	–1100 to 1100	–1980 to 1980
J NBS	–150 to 750	–238 to 1382	–900 to 900	–1620 to 1620
K NBS	–200 to 1350	–328 to 2462	–1550 to 1550	–2790 to 2790
N NBS	–100 to 1300	–148 to 2372	–1400 to 1400	–2520 to 2520
R NBS	0 to 1750	32 to 3182	–1750 to 1750	–3150 to 3150
S NBS	0 to 1750	32 to 3182	–1750 to 1750	–3150 to 3150
T NBS	–200 to 400	–328 to 752	–600 to 600	–1080 to 1080
L DIN	–200 to 900	–328 to 1652	–1100 to 1100	–1980 to 1980
U DIN	–200 to 600	–328 to 1112	–800 to 800	–1440 to 1440

Voltage and resistance

Electrical	Single	Differential
Unit	Range	Range*
mV	–6 to +22	–29 to + 28
	–10 to 100	–110 to 110
	–50 to 500	–550 to 550
Ohms	0 to 100	–100 to 100
	0 to 400	–400 to 400
	0 to 2000	–2000 to 2000
*Each sensor must be designed for the single range		

5.2.5 Block parameters

Analog Input block

Parameter	Valid range/ Options	Default value	Description/Action
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set to MODE_BLK.Target to AUTO
BLOCK_ERR	RO		Block errors, see Chapter 5.2.3
PV	RO		Process value used in executing the block
OUT			Output value calculated as a result of executing the block
SIMULATE		Disable	See Chapter 5.2.2, Enable/Disable options: ■ Disable ■ Active
XD_SCALE		0-100%	Transduced value scaling to % of range
OUT_SCALE		0-100%	OUT value scaling to engineering units
GRANT_DENY		0	Access options, see Chapter 2.8.1
IO_OPTS		0	I/O options, see Chapter 2.8.2 and Chapter 5.2.1
STATUS_OPTS		0	Status options, see Chapter 2.8.4 and Chapter 5.2.2
CHANNEL		0	The number of the logical hardware channel to the transducer that is connected to this I/O block, see Chapter 5.1.
L_TYPE	0 - 3	0	Linearization mode, see Chapter 5.2.1 ■ 1: Direct ■ 2: Indirect ■ 3: Indirect Square Root.
LOW_CUT		0 %	Low flow cut-off value in % of scale, see Chapter 5.2.1
PV_FTIME		0 s	Time constant (s) of a single exponential filter for process value
FIELD_VAL			Raw value of the field device in percent of PV range
UPDATE_EVT			Alert generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
ALARM_SUM			Alarm summary, see Chapter 2.9
ACK_OPTION	0, 1	0	See Chapter 2.9 ■ 0: Auto ACK Disable ■ 1: Auto ACK Enable
ALARM_HYS	0 to 50 %	0.5%	Alarm hysteresis in % of OUT_SCALE, see Chapter 2.9
HI_HI_PRI	0 to 15		Priority of the HI_HI alarm, see Chapter 2.9
HI_HI_LIM		+INF	Setting of HI_HI alarm in engineering units, see Chapter 2.9.
HI_PRI	0 to 15		Priority of the HI alarm, see Chapter 2.9
HI_LIM		+INF	Setting of HI alarm in engineering units, see Chapter 2.9.
LO_PRI	0 to 15		Priority of the LO alarm, see Chapter 2.9.
LO_LIM		-INF	Setting of LO alarm in engineering units, see Chapter 2.9.
LO_LO_PRI	0 to 15		Priority of the LO_LO alarm, see Chapter 2.9
LO_LO_LIM		-INF	Setting of LO_LO alarm in engineering units, see Chapter 2.9
HI_HI_ALM			Status of HI_HI alarm and its associated time stamp.
HI_ALM			Status of HI alarm and its associated time stamp.
LO_ALM			Status of LO alarm and its associated time stamp.
LO_LO_ALM			Status of LO_LO alarm and its associated time stamp.
FSAFE_TYPE			Customized AI block parameter Selects fail safe type to be output on block failure ■ Fail Safe Value: block outputs FSAFE_VAL ■ Last Good Value: block outputs last good value with status ■ Wrong Value: block outputs last good value with Bad status
FSAFE_VAL			Customized AI block parameter Sets value for FSAFE_TYPE option Fail Safe Value

Legend: RO = Read Only

5.3 Discrete Input (DI)

The Discrete Input block characterizes the incoming discrete signal from a limit switch transducer or hardware configuration block. The result of execution is the **OUT_D** parameter. In ControlCare Application Designer it is assigned the generic name **Device Tag-DI-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 5-2.

Note



- Information on the PROFIBUS Discrete Input block can be found in Chapter 9.3

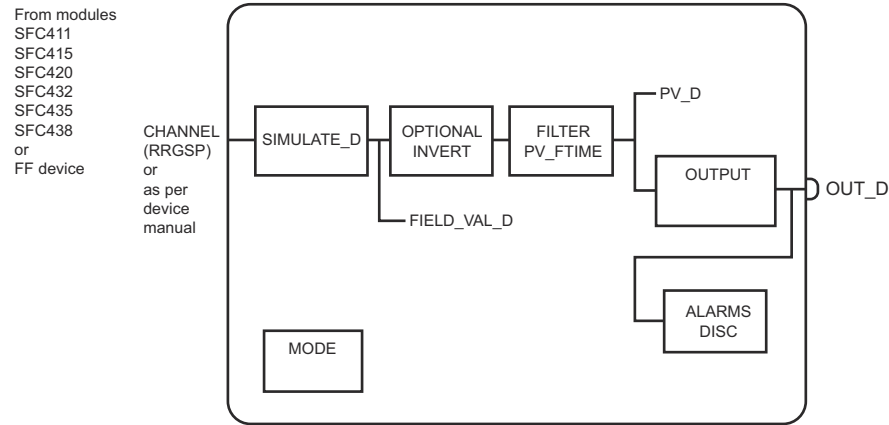


Fig. 5-2: Schematic diagram of Discrete Input block

5.3.1 Functional description

CHANNEL

The **CHANNEL** parameter determines the transducer block logical channel that is to be connected to the Discrete Input block, see Chapter 5.1.

IO_OPS.Invert

If required, the **INVERT** option in the parameter group **IO_OPTS** inverts the discrete signal by placing a boolean NOT between the field value **FIELD_VAL_D** and the output **OUT_D** when true.

False (default)	A discrete value of zero (0) is considered to be a logical zero (0), and a non-zero discrete value a logical one (1)
True (box ticked)	A discrete value of zero (0) is considered to be a logical one (1), and a non-zero discrete value a logical zero (0)

FIELD_VAL_D uses the **XD_STATE** value to indicate the true state on/off state of the connected hardware.

STATUS_OPTS

STATUS_OPTS controls the status propagation of the signal to downstream blocks, see Chapter 5.3.3.

PV_FTIME

PV_FTIME can be used to specify the time the hardware must be in a particular state before the logical state is passed on to **PV_D**. The default value is zero.

DISC_PRI
DISC_LIM

The Discrete Input block supports alarming of the **OUT_D** value using the parameters **DISC_PRI** and **DISC_LIM**, in which alarm priority and alarm value ("0" or "1") can be set. A full description of the function is to be found in Chapter 2.9.

5.3.2 Block configuration

The basic configuration of a Discrete Input block is shown in the table below. It is assumed that the discrete signal originates from rack 1, slot 2, group 0 and I/O point 3 of the local I/O, i.e. channel number 1203, and is to be inverted. In our case the default status options will be taken, so that the **STATUS_OPTS** parameter does not have to be configured (see Chapter 5.3.3).

Parameter	Function	Example value
MODE BLOCK/TARGET	Normal operating mode of block	Auto
IO_OPS	Causes signal to be inverted when option ticked	Invert
CHANNEL	Output channel of the local I/O connected to the block ■ Rack 1, slot 2, group 0 and I/O point 3 = 1203	1203
PV_FTIME	Output damping constant (in seconds).	1

Order of parameters

Some block parameters have a write check based on the value of others parameters. It is therefore important to set the parameters in the order shown (the same order in which they are displayed in the **Off Line Characterization** dialog.

After characterization of the block, the parameters will appear in the FOUNDATION Fieldbus tree. If you find a parameter in the wrong position, it can be move by dragging and dropping to the correct one.

5.3.3 Block operation

The block normally operates in AUTO mode. When online, the status and value of block output is displayed in the **OUT_D** parameter. The **XD_STATE** parameter indicates the true state on/off state of the connected hardware. Any discrete alarm is displayed in the **ALARM_SUM** and, if necessary, can be acknowledged in the **DISC_ALM** parameter.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

When **MODE_BLK.Target** is set to MAN, the block output can be manipulated by entering the desired status and value in the **OUT_D** parameter. The **PV_D** value indicates the correct state.

SIMULATE_D

The process value in the Discrete Input block can be simulated in order to check the function of it and subsequent blocks during commissioning. A prerequisite for this is that the hardware lock of the device is set to enable. When simulation is enabled, "Simulation Active" appears in the resource block parameter **BLOCK_ERROR** by , see Chapter 2.9 and Chapter 3.2.7.

Simulation parameters can be changed when the block is in AUTO mode. Expand the **SIMULATE_D** parameter to enter a simulated value:

- Set the status of the simulated variable in **SIMULATE_STATUS**
- Set the value of the simulated variable in **SIMULATE_VALUE**
- Set **ENABLE_DISABLE** to "Active" to start simulation

The parameters **TRANSDUCER_STATUS** and **TRANSDUCER_VALUE** indicate the status and value of the raw process value. The block output **OUT_D** reflects the simulated parameters after processing in the block.

Simulation in an I/O block is disabled by setting **ENABLE_DISABLE** to "Disable". In this state **SIMULATE_STATUS** and **SIMULATE_VALUE** etc. track **TRANSDUCER_STATUS** and **TRANSDUCER_VALUE** in order to provide a bumpless transfer from disabled to enabled.

Status propagation

The Discrete Input block supports the qualities Bad, Uncertain and GoodNC together with the associated substatures, see Chapter 2.3.2. Status propagation can be influenced by selecting the following options in the **STATUS_OPTS** parameter, see Chapter 2.8.3.

Option	Description
Propagate Fail Forward	Propagates a transducer block status Bad.DeviceFailure or Bad.SensorFailure to OUT_D without generating an alarm. – Allows the user to determine whether alarming (sending of an alert) will be done by the block or propagated downstream
Uncertain if Man	Sets the output status of the block to Uncertain if the actual mode of the block is Man

Block errors

On execution of the block, the device checks for correct configuration. Block errors are indicated in the **BLOCK_ERR** parameter.

Condition	Reason
Block Configuration Error	<ul style="list-style-type: none"> CHANNEL parameter has an invalid value CHANNEL and HC transducer block configuration not compatible
Simulate Active	<ul style="list-style-type: none"> The current OUT_D value and status is simulated
Input Failure/ process variable has BAD status	<ul style="list-style-type: none"> I/O module or device failure, device initiating etc.
Out-of-Service	<ul style="list-style-type: none"> MODE_BLK.Target currently set to OOS

5.3.4 Block parameters

Parameter	Valid range/ Options	Default value	Description/Action
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set to MODE_BLK.Target to AUTO
BLOCK_ERR	RO		Block errors, see Chapter 5.2.3
PV_D	RO		Process value used in executing the block
OUT_D			Output value calculated as a result of executing the block
SIMULATE_D		Disable	See Chapter 5.2.2, Enable/Disable options: <ul style="list-style-type: none"> Disable Active
XD_STATE	0, 1	0	True logic state of connected hardware
OUT_STATE	0, 1	0	Logic state used by PV_D and OUT_D
GRANT_DENY		0	Access options, see Chapter 2.8.1
IO_OPTS		0	"Invert" inverts signal, see Chapter 2.8.2 and Chapter 5.3.1
STATUS_OPTS		0	Status options, see Chapter 2.8.4 and Chapter 5.2.3
CHANNEL		0	The number of the logical hardware channel to the transducer that is connected to this I/O block, see Chapter 5.1.
PV_FTIME		0 s	Time constant (s) of a single exponential filter for process value
FIELD_VAL_D			Raw value of the field device
UPDATE_EVT			Alert generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
ALARM_SUM			Alarm summary, see Chapter 2.9
ACK_OPTION	0, 1	0	See Chapter 2.9 <ul style="list-style-type: none"> 0: Auto ACK Disable 1: Auto ACK Enable
DISC_PRI	0 to 15		Priority of the discrete alarm, see Chapter 2.9
DISC_LIM	0, 1	+INF	State for which alarm must be output, see Chapter 2.9.
DISC_ALM			Status of discrete alarm and its associated time stamp.
Legend: RO = Read Only			

5.4 Pulse Input (PULS)

The Pulse Input Block provides analog output values based on a pulse (counter) transducer input. Two primary outputs are available:

- **OUT** provides an analog rate based on the number of pulses counted per block execution. It is characterized by **PULSE_VAL** and **TIME_UNITS** and the engineering units are selected by scaling the output: the resulting range can be alarmed.
- **OUT_ACCUM** is the accumulated number of pulses and is intended to be connected to an integrator block for differencing, conversion, and integration. It is used when the count rate is low relative to the block execution rate.

In ControlCare Application Designer it is assigned the generic name **Device Tag-PULS-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 5-3

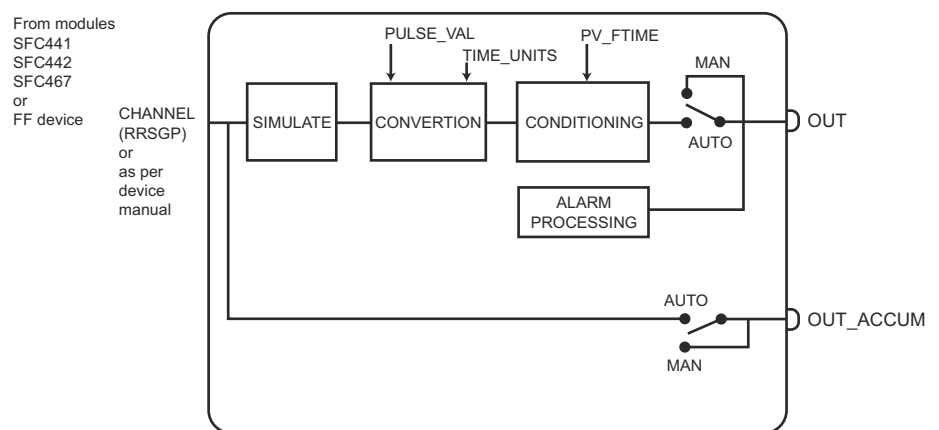


Fig. 5-3: Schematic diagram of Pulse Input block

5.4.1 Functional description

CHANNEL

The **CHANNEL** parameter determines the transducer block logical channel carrying the pulse value (device or local I/O), see Chapter 5.1. The number of pulse counted over the block execution cycle is available at **OUT_ACCUM**, an analog rate at **OUT**.

PULSE_VAL TIME_UNITS

If the **OUT** parameter is being used, the rate is set with the following parameters:

- The engineering quantity associated with one pulse in **PULSE_VAL**
- The time units of the measurement in **TIME_UNITS** (for HMI system)

OUT_SCALE

If the **OUT** parameter is being used, the process value is scaled to the range and units required by any follow-up block with **OUT_SCALE**. Expand it to reveal the following elements

- EU_100 is the upper range limit of the output signal
- EU_0 is the lower range limit of the output signal
- UNIT_INDEX is the unit of the output signal
- DECIMAL is the number of figures behind the decimal point

PV_FTIME

The **PV_FTIME** parameter allows the output to be damped by entering a suitable positive value in seconds. The default value is zero.

ALARM_HYS, XXX_PRI XXX_LIM

The Analog Input block supports alarming of the **OUT** value using the parameters **ALARM_HYS**, **XXX_PRI** and **XXX_LIM**, in which global hysteresis, alarm priority and alarm values can be set individually for HI_HI_, HI_, LO_ and LO_LO_ limits. A full description of the function is to be found in Chapter 2.9.

5.4.2 Block configuration

The basic configuration of a Pulse Input block is shown in the table below. It is assumed that the pulse signal is obtained from a SFC442 pulse input module located in rack 1, slot 3, group 0 and I/O point 0 of the local I/O, i.e. channel number 1300. Each pulse corresponds to 0.01 kg. The possible flow rate lies between 0 and 80 kg/min. In our case the default status options will be taken, so that the **STATUS_OPTS** parameter does not have to be configured (see Chapter 5.4.3).

Parameter	Function	Example value
MODE_BLOCK/TARGET	Normal operating mode of block	Auto
PULSE_VAL	Engineering quantity associated with one pulse	0.01
TIME_UNITS	Time units of the measurement	minutes
OUT_SCALE/EU_100	Upper range limit for output variable	80
OUT_SCALE/EU_0	Lower range limit for output variable	0
OUT_SCALE/UNITS_INDEX	Unit of output variable	kg
CHANNEL	Output channel of the local I/O connected to the block ■ Rack 1, slot 3, group 0 and I/O point 0 = 1300	1300
PV_FTIME	Output damping constant (in seconds).	1

5.4.3 Block operation

The block normally operates in AUTO mode. When online, the status and value of block output is displayed in the **OUT** parameter. The value and status of the signal from the transducer block can be viewed in the **FIELD_VAL** parameter. Any limit alarms are displayed in the **ALARM_SUM** and, if necessary, can be acknowledged in the **HI_HI_ALM** etc. parameters.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

When **MODE_BLK.Target** is set to MAN,

- the block output can be manipulated by entering the desired status and value in the **OUT** parameter. The **PV** parameter indicates the correct state.
- the **OUT_ACCUM** counter can reset by entering a new value (0) – it starts counting again as soon as the block is set to AUTO.

SIMULATE_P

The process value in the Pulse Input block can be simulated in order to check the function of it and subsequent blocks during commissioning. A prerequisite for this is that the hardware lock of the device is set to enable. When simulation is enabled, "Simulation Active" appears in the resource block parameter **BLOCK_ERROR** by , see Chapter 2.9 and Chapter 3.2.7.

Simulation parameters can be changed when the block is in AUTO mode. Expand the **SIMULATE_P** parameter to enter a simulated value:

- Set the status of the simulated variable in **SIMULATE_STATUS**
- Set the value of the simulated variable in **SIMULATE_VALUE**
- Set **ENABLE_DISABLE** to "Active" to start simulation

The parameters **TRANSDUCER_STATUS** and **TRANSDUCER_VALUE** indicate the status and value of the raw process value. The block output **OUT** reflects the simulated parameters after processing in the block.

Simulation in an I/O block is disabled by setting **ENABLE_DISABLE** to "Disable". In this state **SIMULATE_STATUS** and **SIMULATE_VALUE** etc. track **TRANSDUCER_STATUS** and **TRANSDUCER_VALUE** in order to provide a bumpless transfer from disabled to enabled.

Status propagation

The Pulse Input block supports the qualities Bad, Uncertain and GoodNC together with the associated substatures, see Chapter 2.3.2. Status propagation can be influenced by selecting the following options in the STATUS_OPTS parameter, see Chapter 2.8.3.

Option	Description
Propagate Fail Forward	Propagates a transducer block status Bad.DeviceFailure or Bad.SensorFailure to OUT without generating an alarm. – Allows the user to determine whether alarming (sending of an alert) will be done by the block or propagated downstream
Uncertain if Limited	Sets the output status of the block to Uncertain when limit alarms HI_HI_, HI_, LO_ or LO_LO_ are violated – Option when alarms have been set on the output value
BAD if Limited	Sets the output status of the block to Bad when limit alarms HI_HI_, HI_, LO_ or LO_LO_ are violated – Option when alarms have been set on the output value
Uncertain if Man	Sets the output status of the block to Uncertain if the actual mode of the block is Man

Block errors

On execution of the block, the device checks for correct configuration. Block errors are indicated in the **BLOCK_ERR** parameter.

Condition	Reason
Block Configuration Error	<ul style="list-style-type: none"> ■ CHANNEL parameter has an invalid value ■ CHANNEL and HC transducer block configuration not compatible ■ XXX_LIM limit value is out of valid range
Simulate Active	<ul style="list-style-type: none"> ■ The current OUT value and status is simulated
Input Failure/ process variable has BAD status	<ul style="list-style-type: none"> ■ I/O module or device failure, device initiating etc.
Out-of-Service	<ul style="list-style-type: none"> ■ MODE_BLK.Target currently set to OOS

5.4.4 Block parameters

Pulse Input block

Parameter	Valid range/ Options	Default value	Description/Action
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set to MODE_BLK.Target to AUTO
BLOCK_ERR	RO		Block errors, see Chapter 5.2.3
PV	RO		Process value used in executing the block
OUT			Output value calculated as a result of executing the block
OUT_ACCUM			Accumulated number of pulses since last reset
SIMULATE_P		Disable	See Chapter 5.2.2, Enable/Disable options: Disable; Active
PULSE_VAL			Engineering quantity associated with one pulse
TIME_UNITS			Time units of the measurement
OUT_SCALE		0-100%	OUT value scaling to require engineering units
GRANT_DENY		0	Access options, see Chapter 2.8.1
IO_OPTS		0	Not used
STATUS_OPTS		0	Status options, see Chapter 2.8.4 and Chapter 5.4.2
CHANNEL		0	The number of the logical hardware channel to the transducer that is connected to this I/O block, see Chapter 5.1.
PV_FTIME		0 s	Time constant (s) of a single exponential filter for process value
FIELD_VAL			Raw value of the field device in percent of PV range
UPDATE_EVT			Alert generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
ALARM_SUM	RO		Alarm summary, see Chapter 2.9
ACK_OPTION	0, 1	0	See Chapter 2.9 <ul style="list-style-type: none"> ■ 0: Auto ACK Disable; ■ 1: Auto ACK Enable
ALARM_HYS	0 to 50 %	0.5%	Alarm hysteresis in % of OUT_SCALE, see Chapter 2.9
HI_HI_PRI	0 to 15		Priority of the HI_HI alarm, see Chapter 2.9
HI_HI_LIM		+INF	Setting of HI_HI alarm in engineering units, see Chapter 2.9.
HI_PRI	0 to 15		Priority of the HI alarm, see Chapter 2.9
HI_LIM		+INF	Setting of HI alarm in engineering units, see Chapter 2.9.
LO_PRI	0 to 15		Priority of the LO alarm, see Chapter 2.9.
LO_LIM		-INF	Setting of LO alarm in engineering units, see Chapter 2.9.
LO_LO_PRI	0 to 15		Priority of the LO_LO alarm, see Chapter 2.9
LO_LO_LIM		-INF	Setting of LO_LO alarm in engineering units, see Chapter 2.9
HI_HI_ALM			Status of HI_HI alarm and its associated time stamp.
HI_ALM			Status of HI alarm and its associated time stamp.
LO_ALM			Status of LO alarm and its associated time stamp.
LO_LO_ALM			Status of LO_LO alarm and its associated time stamp.

5.5 Multiple Analog Input (MAI)

The Multiple Analog Input block scales up to eight electrical input signals of 4 mA to 20 mA or 1 V to 5 V to **OUT_x** values of 0% to 100%. The block can be used together with the ControlCare SFC444 and SFC457 analog input modules. If other input signals are used with these modules, Analog Input blocks must be used.

The block does not support linearization, scaling, low-flow cut off or simulation. If these properties are required, an Analog Input block must be created for the output channel in question.

In ControlCare Application Designer the Multiple Analog Input block is assigned the generic name **Device Tag-MAI-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 5-4.

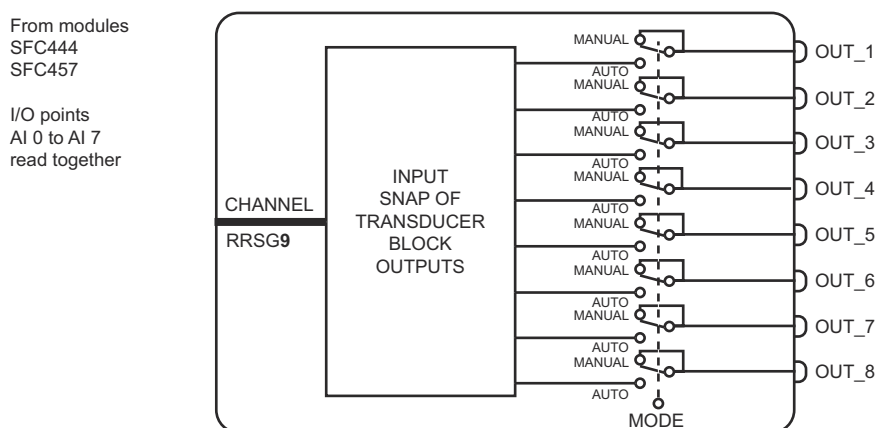


Fig. 5-4: Schematic diagram of Multiple Analog Input block

5.5.1 Functional description

CHANNEL

The **CHANNEL** parameter determines the transducer block logical channel carrying the analog inputs according to the scheme Rack+Slot+Group+9, see Chapter 5.1.

Channel configuration

The input channels of the SFC444 and SFC457 can be configured individually by jumpers on the printed circuit board to accept current or voltage signals, default setting current, see Chapter 3.5.1 of operating instructions BA021S/04/en, Field Controller: Hardware Installation.

5.5.2 Block configuration

The basic configuration of a Multiple Analog Input block is shown in the table below. It is assumed that the analog signals are obtained from a SFC444 analog input module located in rack 1, slot 4.

Parameter	Function	Example value
MODE BLOCK/TARGET	Normal operating mode of block	Auto
CHANNEL	Output channel of the local I/O connected to the block ■ Rack 1, slot 4, group 0 and I/O point 9 = 1409	1409

5.5.3 Block operation

The block normally operates in AUTO mode. When online, the status and value of block output is displayed in the **OUT_X** parameter.

The **OUT_X** value is proportional to the electrical signal over the range 0 mA to 40 mA and 0 V to 10 V. The MAI block converts the input as shown below

Value	Current			Voltage		
	Input value	OUT value	OUT status	Input value	OUT value	OUT status
Underrange limit	0 mA	-25%	GOOD	0 V	-25%	GOOD
Lower range limit	4 mA	0 %	GOOD	1 V	0 %	GOOD
Upper range limit	20 mA	100%	GOOD	5 V	100%	GOOD
Overrange limit	40 mA	225%	GOOD	10 V	225%	GOOD
Out of range	>40 mA	BAD	BAD	>10 V	BAD	BAD

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

When **MODE_BLK.Target** is set to MAN, the block output can be manipulated by entering the desired status and value in the **OUT_x** parameter.

Status propagation

The Multiple Analog Input block supports the qualities Bad, Uncertain and GoodNC together with the associated substatuses, see Table 5-11 above and Chapter 2.3.2.

Block errors

On execution of the block, the device checks for correct configuration. Block errors are indicated in the **BLOCK_ERR** parameter.

Condition	Reason
Block Configuration Error	<ul style="list-style-type: none"> CHANNEL has an invalid value CHANNEL and HC transducer block configuration not compatible
Input Failure/ process variable has BAD status	<ul style="list-style-type: none"> I/O module or device failure, device initiating etc.
Out-of-Service	<ul style="list-style-type: none"> MODE_BLK.Target currently set to OOS

5.5.4 Block parameters

Multiple Analog Input block

Parameter	Valid range/ Options	Default value	Description/Action
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set to MODE_BLK.Target to AUTO
BLOCK_ERR	RO		Block errors, see Chapter 5.2.3
CHANNEL		0	The number of the logical hardware channel to the transducer that is connected to this I/O block, see Chapter 5.1.
OUT_1...OUT_8			Output value and status for channel x calculated as a result of executing the block
UPDATE_EVT			Alert generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
Legend: RO = Read Only			

5.6 Multiple Discrete Input (MDI)

The Multiple Discrete Input block converts up to eight electrical input signals of low or high value into **OUT_Dx** values of 0 or 1. It can be used together with the SFC411, SFC415, SFC420, SFC432, SFC435 and SFC438 discrete input and discrete input/output modules. If signal characterization or simulation is required, Discrete Input blocks must be used.

In ControlCare Application Designer the Multiple Analog Input block is assigned the generic name **Device Tag-MDI-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 5-5.

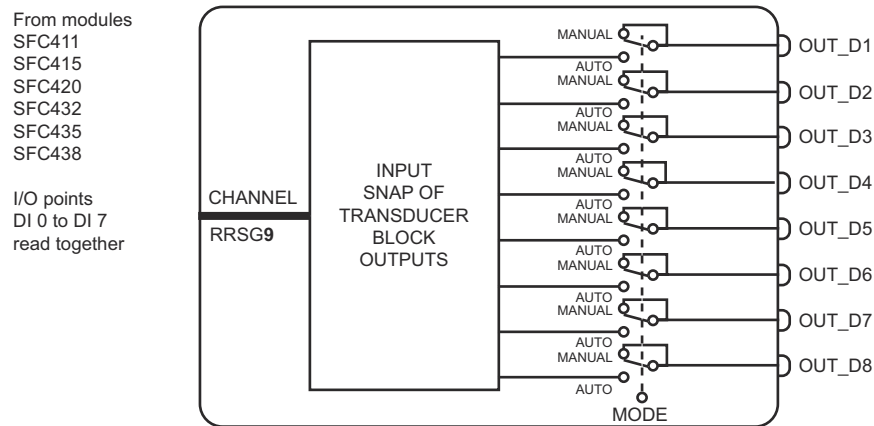


Fig. 5-5: Schematic diagram of Multiple Discrete Input block

5.6.1 Functional description

CHANNEL

The **CHANNEL** parameter determines the transducer block logical channel carrying the discrete inputs according to the scheme Rack+Slot+Group+9, see Chapter 5.1..

5.6.2 Block configuration

The basic configuration of a Multiple Analog Input block is shown in the table below. It is assumed that the analog signals are obtained from a SFC411 discrete input module located in rack 2, slot 2. The Group 1 connections are used:

Parameter	Function	Example value
MODE BLOCK/TARGET	Normal operating mode of block	Auto
CHANNEL	Output channel of the local I/O connected to the block ■ Rack 2, slot 2, group 1 and I/O point 9 = 2219	2219

5.6.3 Block operation

The block normally operates in AUTO mode. When online, the status and value of block output is displayed in the **OUT_Dx** parameter.

The MDI block converts the input signals as into **OUT_Dx** = "0" or "1" as shown below.

OUT_Dx Value	SFC411	SFC415	SFC420	SFC432, SFC435 SFC438
ON state level (True) OUT_Dx = 1	18 – 30 VDC	0 – 5 VDC, < 200 Ω	Switch latched	15 – 30 VDC
OFF state level (False) OUT_Dx = 0	0 – 5 VDC	20 – 30 VDC, > 10 kΩ	Switches unlatched	0 – 5 VDC

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

When **MODE_BLK.Target** is set to MAN, the block output can be manipulated by entering the desired status and value in the **OUT_Dx** parameter. Status propagation

The Multiple Discrete Input block supports the qualities Bad, Uncertain and GoodNC together with the associated substatures, see Chapter 2.3.2.

Block errors

On execution of the block, the device checks for correct configuration. Block errors are indicated in the **BLOCK_ERR** parameter.

Condition	Reason
Block Configuration Error	<ul style="list-style-type: none"> CHANNEL has an invalid value CHANNEL and HC transducer block configuration not compatible
Input Failure/ process variable has BAD status	<ul style="list-style-type: none"> I/O module or device failure, device initiating etc.
Out-of-Service	<ul style="list-style-type: none"> MODE_BLK.Target currently set to OOS

5.6.4 Block parameters

Multiple Discrete Input block

Parameter	Valid range/ Options	Default value	Description/Action
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set to MODE_BLK.Target to AUTO
BLOCK_ERR	RO		Block errors, see Chapter 5.2.3
CHANNEL		0	The number of the logical hardware channel to the transducer that is connected to this I/O block, see Chapter 5.1.
OUT_D1...OUT_D8		0 s	Output value and status after execution of the block
UPDATE_EVT			Alert generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
Legend: RO = Read Only			

6 Control Blocks

6.1 PID Control

The PID Control block provides proportional, integral and derivative control of the process. The block attempts to correct the error or control deviation between a measured process variable, **PV** and a desired setpoint, **SP**, by calculating an actuating or manipulated variable and then outputting a corrective action as an **OUT** value that adjusts the process accordingly.

The block can be configured to perform a number of control tasks including closed-loop control, feedforward control, cascade control and remote interfacing. A number of options, e.g. bypass or tracking, ensure that the connected actuator continues to operate or fails to safe on bad input under certain process conditions.

In ControlCare Application Designer the PID Control block is assigned the generic name **Device Tag-PID-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 6-1.

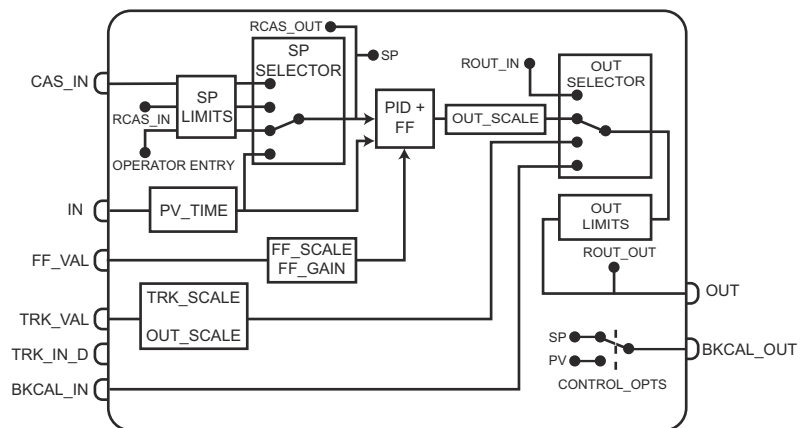


Fig. 6-1: Schematic diagram of PID Control block

Chapter 6.1.1 contains a general description of the block. Subsequent parts of Chapter 6.1 concentrate on its configuration for specific tasks and its behaviour during operation. Chapter 6.1.8 contains an overview of all parameters of the block.

6.1.1 Functional description

Fig. 6.2 shows the internal structure of the PID function block.

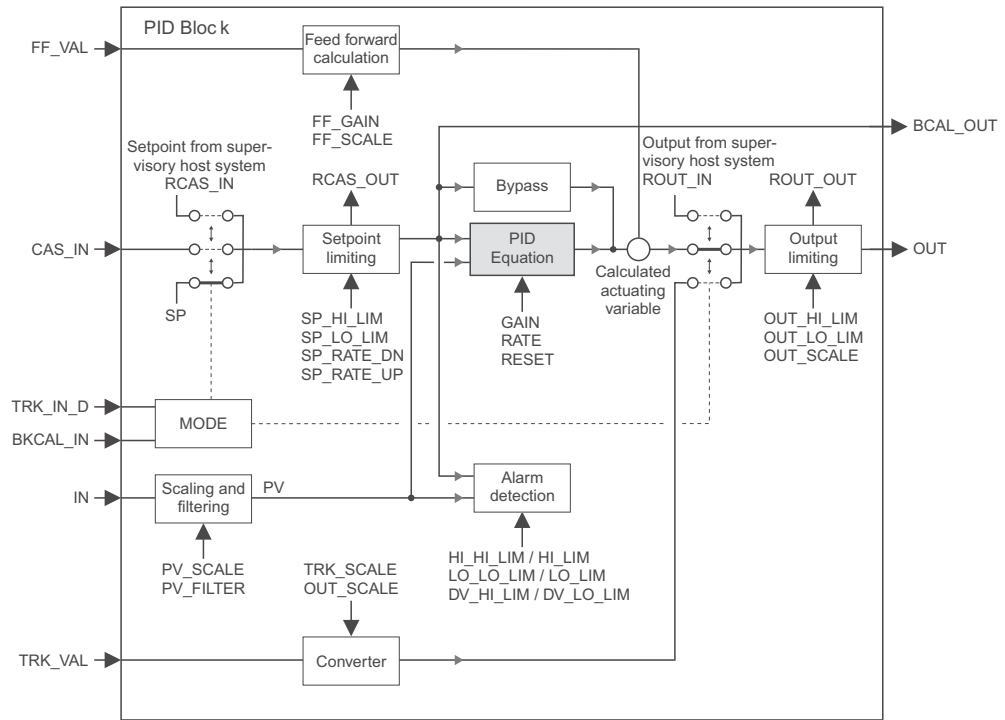


Fig. 6-2: Functional diagram of PID Control block

Process value

The control variable is connected to **IN**, scaled in **PV_SCALE** and passed through the filter **PV_FILTER**. If the value of **PV_FILTER** is zero (default), the filter is disabled. The resulting process value **PV** is used in conjunction with the setpoint **SP** in the PID algorithm. **PV** will normally adopt the status of **IN**, unless there is a block or process alarm, in which case its quality will become "Uncertain". If the block operates in Local Override (LO), the limit status of **PV** is "Constant" and the PID algorithm is not executed.

PV can be limited using the parameters **HI_HI_LIM**, **HI_LIM**, **LO_LO_LIM**, and **LO_LIM**, whereby the default value is no limit. A universal alarm hysteresis in percent of PV span can be set for all limits in the **ALARM_HYS** parameter. Alarm priorities are set in the associated parameter, i.e. **HI_HI_PRI** etc.

If **PV** is out of limits, an alarm is sent to the fieldbus host system. Alarms are displayed in the **ALARM_SUM** parameter and must be acknowledged in the associated alarm parameter, i.e. **HI_HI_ARM** etc. Automatic alarm acknowledgement can be enabled by selecting the appropriate option in **ACK_OPTS**.

Feedforward

The PID Control block supports the feedforward algorithm. An external value that is proportional to some disturbance in the control loop is connected to **FF_VAL**. The value is converted to percent of output span by **FF_SCALE**. This is multiplied by **FF_GAIN** and added to the target output of the PID algorithm ("F" in the PID equation).

To prevent bumping the output, the last good value of "F" is used if the status of **FF_VAL** is "Bad". When the status returns to good, the block adjusts its integral term to maintain the previous output.

Setpoint

Depending upon the operating mode of the PID Control block, the target setpoint and status is provided by **SP**, **CAS_IN** or **RCAS_IN** as shown below:

MODE_BLK	Setpoint value	Origin
AUTO	SP	Value is constant and entered manually by the operator.
CAS	CAS_IN	Originates from an upstream function or control block
RCAS	RCAS_IN	Originates from a supervisory host

The setpoint **SP** can be adjusted in AUTO mode. To ensure bumpless transfer when the setpoint **SP** changes, a downward or upward ramp in PV units per second can be set in the parameters **SP_RATE_DN** and **SP_RATE_UP** respectively. The setpoint then moves from the old to the new value at the set rate. If the ramp rate is set to zero, the new setpoint will be used immediately.

A valid range for setpoint operator entries can be defined by entering appropriate limits in **SP_HI_LIM** and **SP_LO_LIM** in units of PV_SCALE. If an invalid **SP** is entered, the parameter reverts to the last valid entry. Normally, the limits apply to **SP** only but they can be enabled for **CAS_IN** and **RCAS_IN** by selecting the option "Obey SP limits if CAS or RCAS" in **CONTROL_OPTS**.

PID control algorithm

The PID Control block algorithm involves three separate parameters:

- the proportional value or P-term determines the reaction to the current error
- the integral value of I-term determines the reaction based on the sum of recent errors and
- the derivative value or D-term, determines the reaction to the rate at which the error has been changing.

The weighted sum of these three actions is output to a control element such as the position of a control valve or power into a heating element according to the following equation:

$$y = \text{GAIN} \cdot \left[e + \frac{1}{\text{RESET}} \cdot \int e \cdot \Delta t + \frac{\text{RATE}}{(1 + 0.13 \times \text{RATE})} + \frac{\Delta \text{PV}}{\Delta t} \right] + F$$

whereby:

- y = Actuating (or manipulated) variable calculated from the PID algorithm (OUT).
- GAIN = Proportional gain factor (100/proportional band)
- RESET = Time constant for the integral action in seconds/repeat
- RATE = Time constant for the derivative action in seconds
- e = Control deviation or error ($e = \text{SP} - \text{PV}$)
- PV = Process value
- F = Disturbance variable ($F = \text{FF_VAL} \times \text{FF_GAIN}$)

By default, the PID algorithm is "Indirect Acting", i.e. the output increases when **PV** is less than **SP** and decreases when the reverse is true. This behaviour can be reversed by selecting the "Direct Acting" option in **CONTROL_OPTS**. The output will now decrease when **PV** is less than **SP** and vice versa. The option also controls whether positive or negative feedback is applied and affects the calculation of the limit state for **BKCAL_OUT**.

The control deviation "e" can be limited using the parameters **DV_HI_LIM** and **DV_LO_LIM**, whereby the default value is not limited-. A universal alarm hysteresis in percent of PV span can be set for both limits in the **ALARM_HYS** parameter (this also applies to **HI_HI_LIM** etc.). Alarm priorities are set in **DV_HI_PRI** and **DV_LO_PRI**.

If **DV** violates the defined limit values, an alarm is sent to the fieldbus host system. Alarms are displayed in the **ALARM_SUM** parameter and must be acknowledged in **DV_HI_ARM** and **DV_LO_ALM**. Automatic alarm acknowledgement can be enabled by selecting the appropriate option in **ACK_OPTS**.

Bypass

The bypass function allows an operator to bypass the PID algorithm if the **PV** of a secondary cascade control-loop is "Bad". The function is enabled by selecting the option "Bypass Enable" in **CONTROL_OPTS**. The bypass itself can be switched on and off by changing to MAN or OOS mode and selecting the appropriate option in **BYPASS**.

When **BYPASS** is "On", the value of **SP**, in percent of range, is passed directly to the target output, and the value of **OUT** is used for **BKCAL_OUT**. When the operating mode returns to CAS, the upstream block is requested to initialize to the value of **OUT**.

When **BYPASS** is reset to "Off" and the block returns to CAS, the upstream block is requested to initialize to the **PV** value, regardless of the "Use PV for BKCAL_OUT" option, see back calculation.

Back calculation

The **BKCAL_IN** parameter carries the feedback, i.e. value and status, from the **BKCAL_OUT** parameter of a downstream block and is used to prevent reset windup and to initialize the control loop. The PID Control block always acts on the status it carries, independent of any control options and independent of whether it is operating in AUTO, CAS or RCAS mode. Possible statuses and actions are listed in Chapter 2.3.

Normally, the **BKCAL_OUT** parameter of the PID Control block itself carries the value and status of **SP**. Depending on the **CONTROL_OPTS** parameter, it may be forced to a different condition.

When the block operates in RCAS, **RCAS_OUT** (= SP) provides the back calculation for the host, allowing action to be taken under limiting conditions or mode change.

CONTROL_OPTS option	BKCAL_OUT value and status
None	SP value and status
None, but RCAS mode	PV value and status after limiting
Bypass Enable, BYPASS "On"	OUT value and substatus "Initialization Request" on return to CAS OUT value and status after receipt of substatus "Initialization Acknowledge"
Bypass Enable, BYPASS "Off"	PV value and substatus "Initialization Request" on return to CAS, SP value and status after receipt of substatus "Initialization Acknowledge"
Use PV for BKCAL_OUT	PV value and status

Output

When the block is operating in AUTO, CAS or RCAS mode, the actuating variable becomes the output **OUT** after being being scaled in **OUT_SCALE**. It can also be limited in **OUT_HI_LIM** and **OUT_LOW_LIM**. If the calculated value of the output exceeds or drops below these limits, the appropriate limit value is used as output. The **BKCAL_HYS** parameter places a hysteresis, in percent of output scale, on the limit values.

If the block is in MAN mode, the **OUT** value and status may be set by hand. In order to prevent bumps on change from one mode to another, the **OUT** value is balanced. The time taken for the transition from the old to the new value can be entered in **BAL_TIME**.

If the block is in ROUT mode, the PID algorithm is not used and **OUT** is fed directly from a supervisory host system via the parameter **ROUT_IN**. It may also be replaced by an external tracking value as in output tracking. **ROUT_OUT** provides the feedback to the host

Output tracking

The PID Control block supports the track algorithm, allowing the output to be forced to a value on command by a discrete value. The actual mode then goes to local override, LO. Tracking is enabled for AUTO and RCAS modes by selecting the option "Track Enable" in **CONTROL_OPTS**. If tracking is to be enabled in MAN mode, the option "Track in Manual" must be selected as well. The **TRK_VAL** input brings in an external track value or may be set manually. The value is converted to percent of output span using the parameter values of **TRK_SCALE**. When **TRK_IN_D** is true, the converted **TRK_VAL** replaces **OUT** and the operating mode is forced to LO. The track request is ignored if the current mode is OOS.

If the status of **TRK_IN_D** is "Bad", its last usable value will be maintained and acted upon. If the device restarts, losing the last usable value, the status will be set to false. If there is no last usable value, the present value of the output will be used.

6.1.2 Closed-loop control

Basic closed-loop control requires process input, e.g. a measurement provided by an Analog Input block, that has an effect on the process output, e.g. the Analog Output block, which is calculated by the control block. The result (the control signal) is used as input to the process, closing the loop.

Example

A measurement of flow upstream of a valve ensures a constant flow rate downstream of a valve, by causing the valve to open and close, see Fig. 6-2.

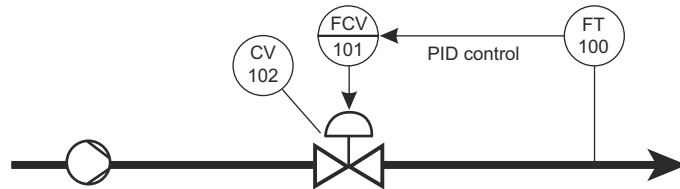


Fig. 6-3: Example of closed-loop control

Fig. 6-3 shows the control strategy together with links that must be made between the blocks.

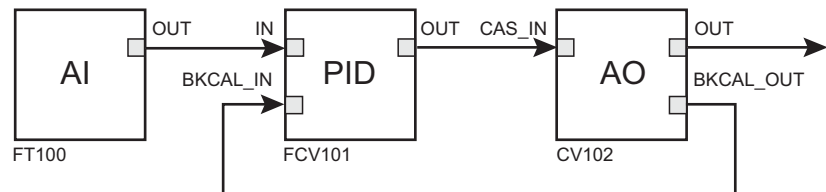


Fig. 6-4: Links required for basic closed-loop control

Block configuration

The block is configured as follows, see also the table below:

- Set **MODE_BLK.Target** to AUTO
- Enter the setpoint value in the units required by the PID algorithm in **SP**
- Scale the input from the AI block to the units required by the PID algorithm in **PV_SCALE**
- Scale the output of the block in the units required by the AO block in **OUT_SCALE**
- To bypass the PID algorithm on bad input status, select Bypass Enable in **CONTROL_OPTS**
- To ensure bumpless transfer should **SP** change, e.g. through a manual intervention, set a rate of change in the **SP_RATE_DN** and **SP_RATE_UP** parameters with which the old SP value adapts to the new SP value
- Set the parameters **GAIN**, **RESET** and **RATE** to values appropriate to your application.

Parameter	Function	Example value
MODE_BLK.Target	Normal operating mode of block	Auto
SP	Setpoint for ideal process control	60%
PV_SCALE.EU_100	Upper range limit for process variable	100
PV_SCALE.EU_0	Lower range limit for process variable	0
PV_SCALE.UNITS_INDEX	Unit of process variable	%
OUT_SCALE/EU_100	Upper range limit for output variable	100
OUT_SCALE/EU_0	Lower range limit for output variable	0
OUT_SCALE/UNITS_INDEX	Unit of output variable	%
CONTROL_OPTS	Sets control options for bad input	Bypass Enable
SP_RATE_DN	Rate of change from old to new, higher SP	0
SP_RATE_UP	Rate of change from old to new, lower SP	0
GAIN	Tuning constants for the P, I and D terms of the PID block respectively.	1.5
RESET		0.1 s
RATE		0.5 s

6.1.3 Feedforward control

Feedforward control may be used where a system responds to a measured disturbance in a predictable way. Provided that the disturbance is measurable, the effect of the disturbance on the output of the system is known and the controller affects the output before the disturbance does, feedforward control can be implemented. Feedforward control can also be used with closed-loop control to improve reference tracking performance.

Example

The temperature of water measured downstream of a heat exchanger controls the supply of steam by opening and closing a control valve, see Fig. 6-4. A flowmeter upstream of the heat exchanger gives information on the the amount of water to flowing through the exchanger. The flowmeter Analog Output block supplies the feedforward information, the temperature device Analog Input block the process value. The Analog Output block of the control valve controls the control valve actuator.

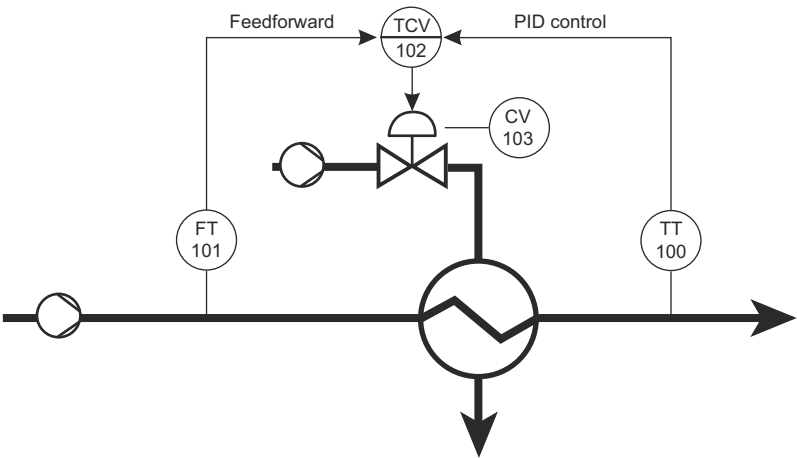


Fig. 6-5: Example of feedforward control

Fig. 6-5 shows the control strategy together with links that must be made between the blocks.

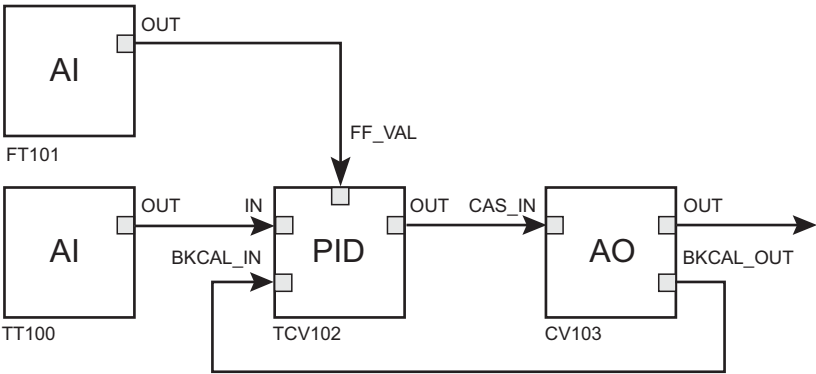


Fig. 6-6: Links required for feedforward and closed loop control

Block configuration

The table below lists the basic parameters for feedforward control with some examples of values. The block is configured as follows:

- Set **MODE_BLK.Target** to AUTO
- Enter the setpoint value in the units required by the PID algorithm in **SP**
- Scale the input from the AI block providing the control measurement to the units required by the PID algorithm in **PV_SCALE**
- Scale the output of the block in the units required by the AO block in **OUT_SCALE**
- To bypass the PID algorithm on bad input status, select Bypass Enable in **CONTROL_OPTS**
- To ensure bumpless transfer should **SP** change, e.g. through a manual intervention, set a rate of change in the **SP_RATE_DN** and **SP_RATE_UP** parameters with which the old SP value adapts to the new SP value
- Set the parameters **GAIN**, **RESET** and **RATE** to values appropriate to your application
- Scale the input from the AI block providing the feedforward measurement to the units required by the PID algorithm in **FF_SCALE**
- Enter the gain associated with the feedforward variable in **FF_GAIN**.

Parameter	Function	Example value
MODE_BLK.Target	Normal operating mode of block	Auto
SP	Setpoint for ideal process control	60%
PV_SCALE.EU_100	Upper range limit for process variable	100
PV_SCALE.EU_0	Lower range limit for process variable	0
PV_SCALE.UNITS_INDEX	Unit of process variable	%
OUT_SCALE/EU_100	Upper range limit for output variable	100
OUT_SCALE/EU_0	Lower range limit for output variable	0
OUT_SCALE.UNITS_INDEX	Unit of output variable	%
CONTROL_OPTS	Sets control options for bad input	Bypass Enable
SP_RATE_DN	Rate of change from old to new, higher SP	0
SP_RATE_UP	Rate of change from old to new, lower SP	0
GAIN	Tuning constants for the P, I and D terms of the PID block respectively.	1.5
RESET		0.1 s
RATE		0.5 s
FF_SCALE.EU_100	Upper range limit for feedforward variable	100
FF_SCALE.EU_0	Lower range limit for feedforward variable	0
FF_SCALE.UNITS_INDEX	Unit of feedforward variable	%
FF_GAIN	The gain that the feedforward input is multiplied by before it is added to the calculated control output.	1

6.1.4 Cascade control

Cascade control uses the output of the primary PID Control block to manipulate the setpoint of the secondary PID Control block as if it were the final control element. This allows disturbances in the secondary loop as well as non-linear valve and other final control element problems to be handled by the secondary PID Control block. In addition, the operator can directly control the secondary loop during certain modes of operation, such as startup

The requirements for cascade control are:

- the process dynamics of the secondary loop must be at least four times as fast as those of the primary loop
- the secondary loop must have influence over the primary loop.
- the secondary loop must be measurable and controllable.

Example

The temperature of water measured upstream of a heat exchanger is used as input to a primary PID Control block which manipulates a secondary block controlling the supply of steam by opening and closing a control valve, see Fig. 6-7. The input to the secondary block is the flowrate of the steam. The flowmeter Analog Output block supplies the input value to the secondary loop, the temperature device Analog Input block the process value. The Analog Output block of the control valve controls the control valve actuator.

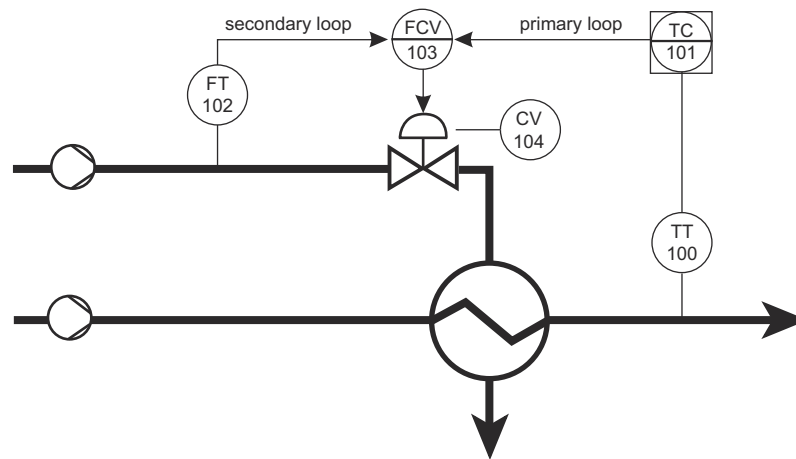


Fig. 6-7: Example of cascade control

Fig. 6-7 shows the control strategy together with links that must be made between the blocks.

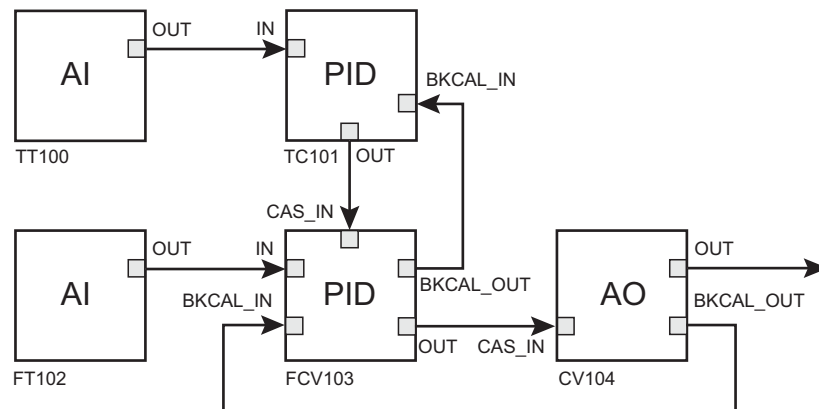


Fig. 6-8: Links required for cascade control

Block configuration

The table below lists the basic parameters for cascade control with some examples of values. The block is configured as follows:

Primary loop

- Set **MODE_BLK.Target** to AUTO
- Enter the setpoint value in the units required by the PID algorithm in **SP**
- Scale the input from the AI block to the units required by the PID algorithm in **PV_SCALE**
- Scale the output of the block in the units required by the AO block in **OUT_SCALE**
- To bypass the PID algorithm on bad input status, select Bypass Enable in **CONTROL_OPTS**
- To ensure bumpless transfer should **SP** change, e.g. through a manual intervention, set a rate of change in the **SP_RATE_DN** and **SP_RATE_UP** parameters with which the old SP value adapts to the new SP value
- Set the parameters **GAIN**, **RESET** and **RATE** to values appropriate to your application.

Secondary loop

- Set **MODE_BLK.Target** to CAS
- Enter the setpoint value in the units required by the PID algorithm in **SP**
- Scale the input from the AI block to the units required by the PID algorithm in **PV_SCALE**
- Scale the output of the block in the units required by the AO block in **OUT_SCALE**
- To bypass the PID algorithm on bad input status, select Bypass Enable in **CONTROL_OPTS**
- To ensure bumpless transfer should **SP** change, e.g. through a manual intervention, set a rate of change in the **SP_RATE_DN** and **SP_RATE_UP** parameters with which the old SP value adapts to the new SP value
- Set the parameters **GAIN**, **RESET** and **RATE** to values appropriate to your application
- To ensure automatic recovery to CAS if the mode changes, e.g. due to bad input, select the option "Normal shed, Normal return" in the **SHED_OPT** parameter
(For other options see Chapter 2.4.4)

Parameter	Function	Example value
MODE_BLK.Target	Normal operating mode of block	Primary loop: AUTO Secondary loop: CAS
SP	Setpoint for ideal process control	60%
PV_SCALE.EU_100	Upper range limit for process variable	100
PV_SCALE.EU_0	Lower range limit for process variable	0
PV_SCALE.UNITS_INDEX	Unit of process variable	%
OUT_SCALE/EU_100	Upper range limit for output variable	100
OUT_SCALE/EU_0	Lower range limit for output variable	0
OUT_SCALE/UNITS_INDEX	Unit of output variable	%
CONTROL_OPTS	Sets control options for bad input	Bypass Enable
SP_RATE_DN	Rate of change from old to new, higher SP	0
SP_RATE_UP	Rate of change from old to new, lower SP	0
GAIN	Tuning constants for the P, I and D terms of the PID block respectively.	1.5
RESET		0.1 s
RATE		0.5 s
SHED_OPT	Behaviour when shedding from remote mode	Normal shed, normal return

6.1.5 Output tracking

The output tracking function allows the PID Control block output to be overridden and replaced by a predefined value set in the **TRK_VAL** parameter. The override is only effective when tracking is enabled and the **TRK_IN_D** value is true. During this time, the block mode changes to LO (local override). The function is normally used to force the block to output a fail-safe value.

Example

A measurement of flow upstream of a valve is used to ensure a constant flow rate into a vessel, see Fig. 6-9. To prevent overspill, the vessel is equipped with a level limit switch. If the liquid rises to the switch the latter switches, causing the valve to close (and the pump to stop). The flowrate is controlled as in Chapter 6.1.1, the tracking signal is delivered through a Discrete Input block.

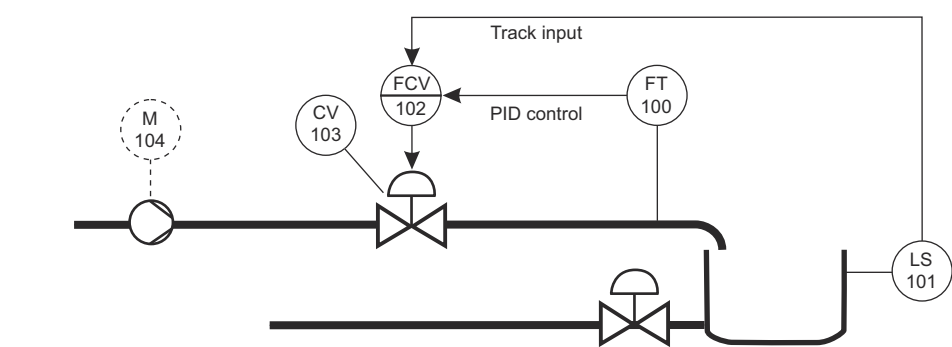


Fig. 6-9: Example of output tracking

Fig. 6-10 shows the control strategy together with links that must be made between the blocks.

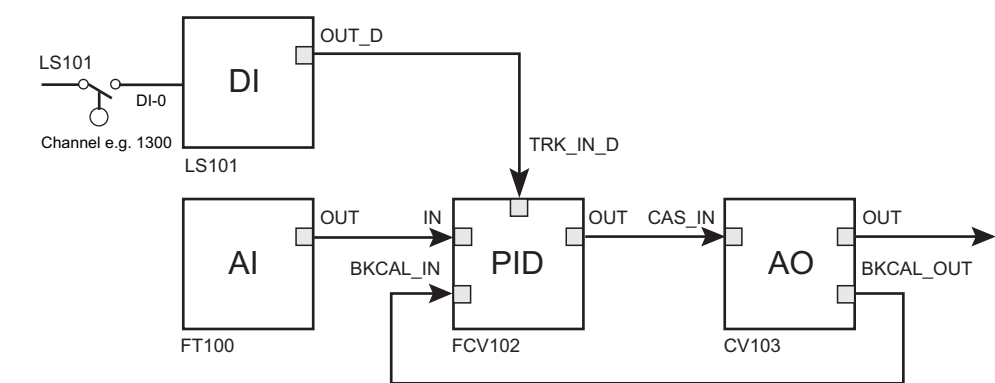


Fig. 6-10: Links required for tracking (with basic loop-control)

Block configuration

Output tracking is configured as follows:

- Depending upon type, configure the control loop is as described in Chapters 6.1.1 to 6.1.3.
- Enable tracking by selecting "Track Enable" in the **CONTROL_OPS** parameter
- Select the option "Use uncertain as good" in the **STATUS_OPS** parameter
- Scale the track value to the units of the **OUT** value in **TRK_SCALE**
- Enter the track value in **TRK_VAL**.

Parameter	Function	Example value
CONTROL_OPS	Enables output tracking	Track Enable
STATUS_OPS	Sets the way the status of the tracking value is used	Use uncertain as good
TRK_SCALE.EU_100	Upper range limit for track value	100
TRK_SCALE.EU_0	Lower range limit for track value	0
TRK_SCALE.UNITS_INDEX	Unit of process track value	%
TRK_VAL	Value to be output by the block	0%

6.1.6 Remote cascade

The remote cascade function allows a setpoint value calculated by an external application, e.g. a Modbus controller, to be used as the primary loop input of a cascade control loop. Since the external application is not contained in the function block schedule, a timeout must be set in the resource block as well as the response of the PID Control block when the timeout is exceeded.

Example

The temperature of water measured upstream of a heat exchanger is acquired by an external Modbus controller and used as input to a primary PID Control block that runs in it. The resulting setpoint manipulates a secondary block controlling the supply of steam by opening and closing a control valve, see Fig. 6-11. The input to the secondary block is the flowrate of the steam. The flowmeter Analog Output block supplies the input value to the secondary loop, the Modbus Control Slave block provides the setpoint to the secondary block, see Chapter 10.3. The Analog Output block of the control valve controls the control valve actuator.

Note



- The connection to the external application may also be made via OPC server, in which case the parameter **Device Tag-PID-n.RCAS_IN** carries the setpoint and the parameter **Device Tag-PID-n.RCAS_OUT** the feedback.

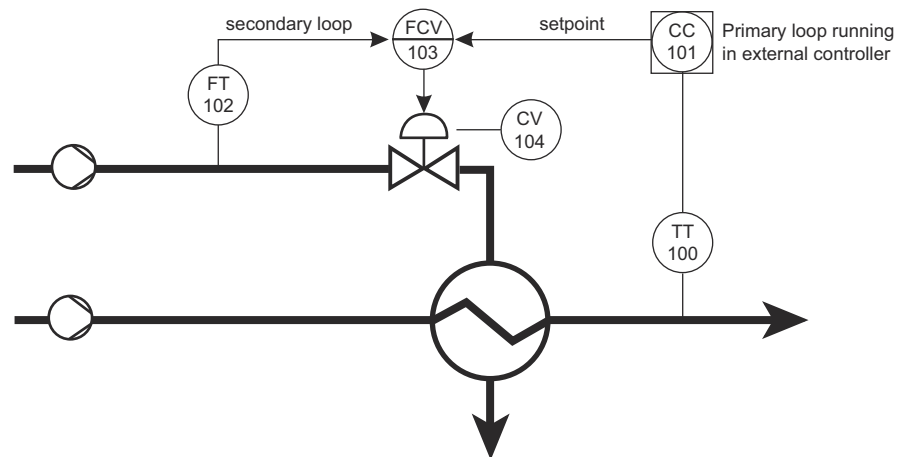


Fig. 6-11: Example of remote cascade

Fig. 6-12 shows the control strategy together with links that must be made between the blocks. For more details on creating control strategies, see the appropriate ControlCare tutorial.

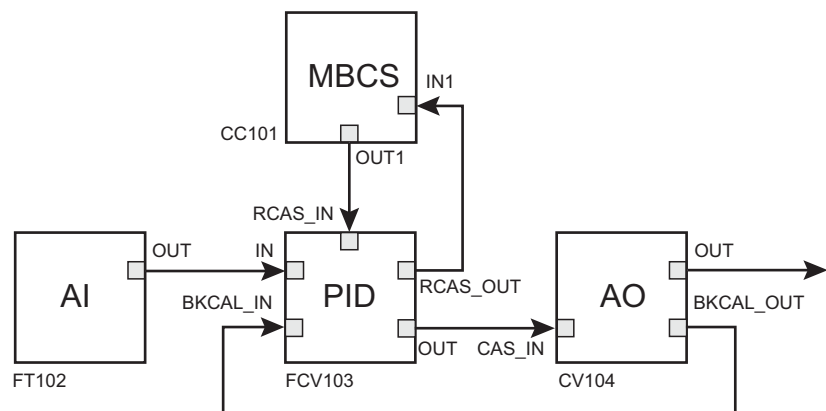


Fig. 6-12: Links required for remote cascade (through MBCS block)

Block configuration

The remote cascade is configured as follows:

- In the Resource Block of the device set **SHED_RCAS** to the maximum time allowable for the execution of the Control PID block
- In the PID Control block set **MODE_BLK.Target** to RCAS
- Scale the input from the AI block to the units required by the PID algorithm in **PV_SCALE**
- Scale the output of the block in the units required by the AO block in **OUT_SCALE**
- To bypass the PID algorithm on bad input status, select Bypass Enable in **CONTROL_OPTS**
- To ensure bumpless transfer should **SP** change, e.g. through a manual intervention, set a rate of change in the **SP_RATE_DN** and **SP_RATE_UP** parameters with which the old SP value adapts to the new SP value
- Set the parameters **GAIN**, **RESET** and **RATE** to values appropriate to your application
- To ensure automatic recovery to RCAS if the mode changes, e.g. due to a timeout, select the option "Shed to Auto, Normal return" in the **SHED_OPT** parameter

Parameter	Function	Value
Resource Block		
SHED_RCAS	Timeout, in units of 1/32 ms <ul style="list-style-type: none"> ■ If the timeout is exceed, communication is assumed to be lost and the mode is shed, SHED_OPTS ■ If the timeout is 0, mode shedding is disabled 	e.g. 6400 (= 200 ms)
PID Control Block		
MODE_BLK.Target	Normal operating mode of block	RCAS
SP	Setpoint for safe operation (used if RCAS_IN fails due to timeout)	60%
PV_SCALE.EU_100 PV_SCALE.EU_0 PV_SCALE.UNITS_INDEX	Upper range limit for process variable Lower range limit for process variable Unit of process variable	100 0 %
OUT_SCALE.EU_100 OUT_SCALE.EU_0 OUT_SCALE.UNITS_INDEX	Upper range limit for output variable Lower range limit for output variable Unit of output variable	100 0 %
CONTROL_OPTS	Sets control options for bad input Enables output tracking	Bypass Enable Track Enable
STATUS_OPTS	Sets the way the status of the tracking value is used	Use uncertain as good
BYPASS	When ON, SP value is transferred to the OUT without the calculation of PID terms.	OFF
SP_RATE_DN SP_RATE_UP	Rate of change from old to new, higher SP Rate of change from old to new, lower SP	0 0
GAIN RESET RATE	Tuning constants for the P, I and D terms, of the PID block respectively.	1.5 0.1 s 0.5 s
SHED_OPT	Behaviour when shedding from remote mode <ul style="list-style-type: none"> ■ 1: Normal shed, Normal return (sheds to CAS) ■ 2: Normal shed, No return (sheds to CAS) ■ 3: Shed to Auto, Normal return ■ 4: Shed to Auto, No return ■ 5: Shed to Manual, Normal return ■ 6: Shed to Manual, No return ■ 7: Shed to Retained target, Normal return ■ 8: Shed to Retained target, No return 	Shed to Auto, Normal return

SHED_OPT

The option "Shed to Auto, Normal return" causes the PID Control block to run in AUTO mode should the communication between the block and the external controller be interrupted. When communication is re-establishment the block automatically recovers to RCAS.

The option "Shed to Auto, No return" causes the PID Control block to run in AUTO mode should the communication between the block and the external controller be interrupted. When communication is re-establishment the block stays in AUTO mode. It must be manually reset to RCAS.

The other modes set the the block mode specified and behave analogously.

6.1.7 Remote output

The remote output function allows a the output of a PID Control block to be overridden by an external application. To do this the block target must be set to ROUT mode and a value must be written to **ROUT_IN**. Since the external application is not contained in the function block schedule, a timeout set in the resource block as well as the response of the PID Control block when the timeout is exceeded must also be set.

The connection to the external application may be made via OPC server, in which case:

- the parameter **Device Tag-PID-n.ROUT_IN** carries the output value,
- the parameter **Device Tag-PID-n.ROUT_OUT** the feedback
- the parameter **Device Tag-PID-n.MODE_BLK.Target** sets the mode block target.

For example, a measurement of flow upstream of a valve is used to ensure a constant flow rate into a vessel, see Fig. 6-13. To prevent overspill, the vessel is equipped with a level limit switch. If the liquid rises to the switch the latter switches, sending an alarm to the operating system. The operator now responds by setting the block mode to ROUT and writing a value to **ROUT_IN**.

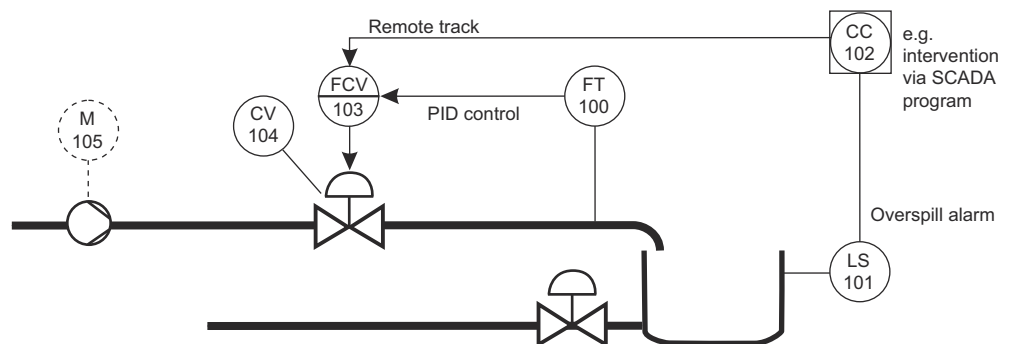


Fig. 6-13: Example of remote output

Fig. 6-11 shows the control strategy together with links that must be made between the blocks. For more details on creating control strategies, see the appropriate ControlCare tutorial.

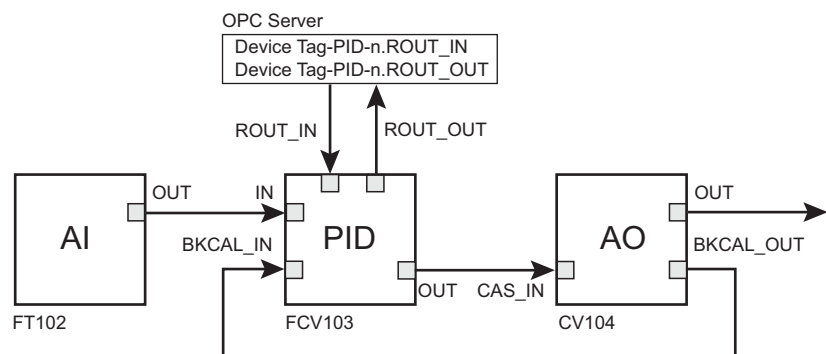


Fig. 6-14: Links required for remote output

Block configuration

The remote cascade is configured as follows:

- In the Resource Block of the device set **SHED_RCAS** to the maximum time allowable for the execution of the Control PID block
- To ensure automatic recovery to ROUT if the mode changes, e.g. due to a timeout, select the option "Normal shed, Normal return" in the **SHED_OPT** parameter

The remote output track is made operational on-line by writing ROUT to the **MODE_BLK.Target** parameter. When the override is to be lifted, the block target is reset to the normal operating mode.

Parameter	Function	Value
Resource Block		
SHED_RCAS	Timeout, in units of 1/32 ms <ul style="list-style-type: none"> ■ If the timeout is exceed, communication is assumed to be lost and the mode is shed, SHED_OPTS ■ If the timeout is 0, mode shedding is disabled 	e.g. 6400 (= 200 ms)
PID Control Block		
SHED_OPT	Behaviour when shedding from remote mode <ul style="list-style-type: none"> ■ 1: Normal shed, Normal return ■ 2: Normal shed, No return ■ 3: Shed to Auto, Normal return ■ 4: Shed to Auto, No return ■ 5: Shed to Manual, Normal return ■ 6: Shed to Manual, No return ■ 7: Shed to Retained target, Normal return ■ 8: Shed to Retained target, No return 	Normal shed, normal return

SHED_OPT

The option "Normal shed, Normal return" causes the PID Control block to run in CAS mode should the communication between the block and the external controller be interrupted. When communication is re-establishment the block automatically to recovers to RCAS.

The option "Normal shed, No return" causes the PID Control block to run in CAS mode should the communication between the block and the external controller be interrupted. When communication is re-establishment the block stays in CAS mode. it must be manually reset to RCAS.

The other modes set the the block mode specified and behave analogously.

6.1.8 Block operation

Operating mode

When configured as a primary control-loop, the block normally operates in AUTO mode. For secondary control-loops the operating mode may be CAS or RCAS. The value and status of the setpoint **SP** can be changed on-line in AUTO and CAS modes.

When **MODE_BLK.Target** is set to MAN, the block output can be manipulated by entering the desired status and value in the **OUT** parameter. Similarly, when the block is set to ROUT, the **OUT** parameter is overwritten by the value and status in **ROUT_IN**. In both cases, the **PV** parameter indicates the correct value and state.

The block is forced to LO, when output track is configured and **TRK_IN_D** is true. It recovers to the set target mode, when **TRK_IN_D** returns to false. The block will be forced to IMAN from CAS or RCAS if the **BKCAL_IN** parameter has the substatus "Initialisation Request". After initialisation and clearing of the substatus, the block will recover to target mode.

If other parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Operation can be checked on-line by viewing a number of parameters:

Parameter	Meaning
IN	Value and status of input from upstream block
PV	Value and status of process value after scaling
SP, CAS_IN, RCAS_IN	Depending upon operating mode, current value and status of set point
OUT	Value and status of block output
FF_VAL	Value and status of any variable connected to the feedforward input
TRK_VAL	Value and status of any variable connected to the output track input
TRK_IN_D	Value and status of any discrete variable connected to the output track trigger
BKCAL_IN	For a secondary control-loop, value and status of feedback from downstream block
BKCAL_OUT	Value and status of feedback for an upstream block

Status propagation

The PID Control block supports the qualities Bad, Uncertain and GoodC together with the associated sub- and limit statuses, see Chapter 2.3.2.

PV normally adopts the status quality of **IN**. If the status of **IN** is "Good", but there is a block or process alarm, **PV** will have the status "Uncertain". **OUT** follows **PV**, **BKCAL_IN** and the setpoint, carrying the quality, substatus and limit status with the highest priority.

Status propagation can be influenced by selecting any of the following options in the **STATUS_OPTS** parameter, see Chapter 2.8.3.

STATUS_OPTS option	Description
IFS if BAD IN	Sets Initiate Fault State status in the OUT parameter when IN is "Bad"
IFS if BAD CAS_IN	Sets Initiate Fault State status in the OUT parameter when CAS_IN is "Bad"
Uncertain as Good	If the status of the IN parameter is "Uncertain", treats it as "Good"
Target to Manual if BAD IN	Sets the target mode to "Man" if the status of IN is "Bad"
Target to next permitted mode if BAD CAS_IN	Sets the target mode to next permitted mode if the target mode is "Cas" and CAS_IN is "Bad"

Block errors

On execution of the block, the device checks for correct configuration. Block errors are indicated in the **BLOCK_ERR** parameter.

Condition	Reason
Block Configuration Error	<ul style="list-style-type: none"> Invalid parameter setting, e.g. Bypass or SHED_OPT Not all required parameters have a value
Link Configuration Error	<ul style="list-style-type: none"> Missing link, e.g. for secondary loop, BKCAL_IN not connected Parameter link mismatch
Local Override	<ul style="list-style-type: none"> The current OUT value is forced (TRK_VAL or ROUT_IN operative)
Device Fault State Set	<ul style="list-style-type: none"> The IFS option is active in STATUS_OPTS and the input is "Bad"
Input Failure/ process variable has BAD status	<ul style="list-style-type: none"> Input module or device failure, device initiating etc.
Output Failure	<ul style="list-style-type: none"> Output module or device failure, device initiating etc.
Out-of-Service	<ul style="list-style-type: none"> MODE_BLK.Target currently set to OOS

Tab. 6-1: Possible PID Control block errors

6.1.9 Block parameters**PID block**

Parameter	Valid range/ Options	Default value	Description
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set MODE_BLK.Target to AUTO, CAS or RCAS
BLOCK_ERR	RO		Block errors, see Chapter 6.1.9
PV	RO		Process value used in executing the block
SP			Setpoint used by the block in AUTO mode (PV_SCALE \pm 10%)
OUT			Output value calculated as a result of executing the block (OUT_SCALE \pm 10%).
PV_SCALE		0-100%	PV and SP value scaling to required engineering units
OUT_SCALE		0-100%	OUT value scaling to required engineering units
GRANT_DENY		0	Access options, see Chapter 2.8.1
CONTROL_OPTS		0	Control options, see Chapter 2.8.4 and Chapter 6.1.1
STATUS_OPTS		0	Status options, see Chapter 2.8.4 and Chapter 6.1.9
IN			Process value
PV_FTIME		0	Time constant of a single exponential filter for process value
BYPASS		0	Off/On toggle for CONTROL_OPTS option "Bypass"
CAS_IN			Setpoint used by the block in CAS mode
SP_RATE_DN	Positive	+INF	Ramp rate for positive setpoint change in PV units/s
SP_RATE_UP	Positive	+INF	Ramp rate for negative setpoint change in PV units/s
SP_HI_LIM		100	High limit for setpoint limiting (PV_SCALE \pm 10%)
SP_LO_LIM		0	Low limit for setpoint limiting (PV_SCALE \pm 10%)
GAIN		0	Proportional term of the PID (Kp value)
RESET	Positive	+INF	Integral term of the PID (Tr value)
BAL_TIME	Positive	0	Time allowed to balance OUT changes as a result of a change from Auto, Cas or Rcas to Man
RATE	Positive	0	Derivative term of the PID (Td value)
BKCAL_IN			Feedback from BKCAL_OUT of a downstream block
OUT_HI_LIM		100	Maximum allowable output value (OUT_SCALE \pm 10%)
OUT_LO_LIM		0	Minimum allowable output value (OUT_SCALE \pm 10%)
BKCAL_HYS	0 to 50%	0.5%	Output limit hysteresis in % of OUT_SCALE
BKCAL_OUT			Feedback for BKCAL_IN of an upstream block.
RCAS_IN			Setpoint used by the block in RCAS mode (from host)
ROUT_IN			Output of the block in ROUT mode (from host)
SHED_OPT	1 - 8	0	Mode shedding options, see Chapter 6.1.6
RCAS_OUT			Setpoint and status for host back calculation in RCAS mode
ROUT_OUT			Setpoint and status for host back calculation in ROUT mode
TRK_SCALE			TRK_VAL value scaling to required engineering units

Parameter	Valid range/ Options	Default value	Description
TRK_IN_D			Discrete input for output track, see Chapter 6.1.5
TRK_VAL			Analog input for output track, see Chapter 6.1.5
FF_VAL			Feedforward value and status, see Chapter 6.1.3.
FF_SCALE			FF_VAL scaling to required engineering units
FF_GAIN			Gain for feedforward input, see Chapter 6.1.3
UPDATE_EVT			This alert is generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
ALARM_SUM			Alarm summary, see Chapter 2.9
ACK_OPTION	0, 1	0	See Chapter 2.9 <ul style="list-style-type: none"> ■ 0: Auto ACK Disable ■ 1: Auto ACK Enable
ALARM_HYS	0 to 50 %	0.5%	Alarm hysteresis in % of OUT_SCALE, see Chapter 2.9
HI_HI_PRI	0 to 15		Priority of the HI_HI alarm, see Chapter 2.9
HI_HI_LIM		+INF	Setting of HI_HI alarm in engineering units, see Chapter 2.9.
HI_PRI	0 to 15		Priority of the HI alarm, see Chapter 2.9
HI_LIM		+INF	Setting of HI alarm in engineering units, see Chapter 2.9.
LO_PRI	0 to 15		Priority of the LO alarm, see Chapter 2.9.
LO_LIM		-INF	Setting of LO alarm in engineering units, see Chapter 2.9.
LO_LO_PRI	0 to 15		Priority of the LO_LO alarm, see Chapter 2.9
LO_LO_LIM		-INF	Setting of LO_LO alarm in engineering units, see Chapter 2.9
DV_HI_PRI	0 to 15	0	Priority of the control deviation HI alarm, see Chapter 2.9
DV_HI_LIM		+INF	Setting of control deviation HI in engineering units
DV_LO_PRI	0 to 15	0	Priority of the control deviation LO alarm, see Chapter 2.9
DV_LO_LIM		-INF	Setting of control deviation LO in engineering units
HI_HI_ALM			Status of HI_HI alarm and its associated time stamp.
HI_ALM			Status of HI alarm and its associated time stamp.
LO_ALM			Status of LO alarm and its associated time stamp.
LO_LO_ALM			Status of LO_LO alarm and its associated time stamp.
DV_HI_ALM			Status of deviation HI alarm and its associated time stamp.
DV_LO_ALM			Status for deviation LO alarm and its associated time stamp.
Legend: RO = Read Only			

6.2 Enhanced PID Control

The Enhanced PID Control block provides following enhancements with respect to the standard PID Control block:

- Additional mechanisms to ensure bumpless transfer from manual to automatic mode
- Additional options for output tracking

When the default values of these enhanced parameters are used, the block exhibits identical behaviour to the PID Control block.

In ControlCare Application Designer the Enhanced PID Control block is assigned the generic name **Device Tag-EPID-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 6-15.

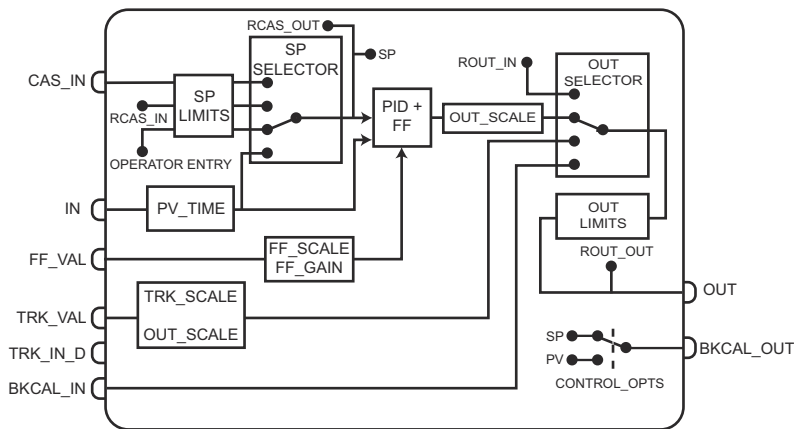


Fig. 6-15: Schematic diagram of EPID Control block

6.2.1 Bumpless transfer

One of four options for transfer from a "manual" mode, e.g. MAN to an "automatic" mode, e.g. AUTO, CAS or RCAS must be selected from the parameter **BUMPLESS_TYPE**:

BUMPLESS_TYPE option	Meaning
Bumpless (default)	block behaves as standard PID Control block and using the last value for "manual" mode to start the PID calculation
Last + proportional	block uses the last value for "manual" mode plus the proportional term to start the PID calculation
Bias	block uses the BIAS parameter to start the PID calculation
Bias + proportional	block uses the BIAS parameter plus the proportional term to start the PID calculation

When options 3 and 4 are used, a bias must be entered as % of PV_SCALE in the **BIAS** parameter during off-line configuration. The BIAS parameter is an additional parameter used in the PID algorithm:

$$y = GAIN \cdot \left[e + \frac{1}{RESET} \cdot \int e \cdot \Delta t + \frac{RATE}{(1 + 0.13 \times RATE)} \cdot \frac{\Delta PV}{\Delta t} \right] + BIAS + F$$

Fig. 6-16 shows the effect of the various options on the transfer from "manual" to "automatic" mode. Unless **OUT** has been set by hand, it takes the value of **SP** in manual mode and the PID algorithm is bypassed. Assuming the the setpoint value **SP** is 50% of **OUT_SCALE** and the process value **PV** is 40% of **OUT_SCALE**, then when the block mode changes from manual to automatic, the PID algorithm will try to reduce the control deviation between **SP** and **PV** by increasing the **OUT** value.

Default behaviour (**BUMPLESS_TYPE** = "Bumpless") may produce too steep a rise in **OUT**, causing problems with overshoot. For some control-loops, overshoot must be avoided at all costs, since it may lead to dangerous situations. The other **BUMPLESS_TYPE** options cause a more gradual rise in **OUT** with the starting points for calculation offset to a greater or lesser degree.

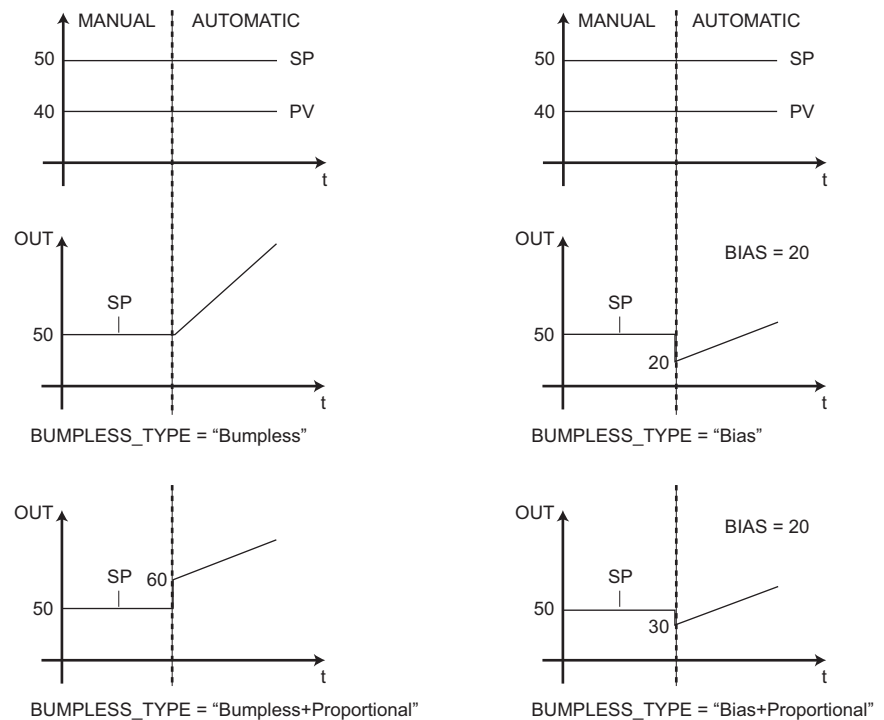


Fig. 6-16: Effect of **BUMPLESS_TYPE** option on **OUT** value

6.2.2 Enhanced output tracking

The parameter **PID_OPTS** provides additional options to deal with "Bad" input for **TRK_IN_D** or **TRK_VAL**. These cause:

- an IFS status to be generated, see Chapter 2.7
- the actual mode to be set to MAN
- the target mode to be changed to MAN

In addition the target mode can be set to MAN on power up or when tracking is active. The effect of the various options is summarized in the following tables, whereby "Auto" signifies AUTO, CAS or RCAS modes. If no selection is made in **PID_OPTS**, the block behaves as per the PID Control block.

Bad TRK_IN_D

PID_OPTS	Mode		Algorithm Action
	Target	Actual	
IFS if Bad TRK_IN_D		"auto" -> Iman	<ul style="list-style-type: none"> Output tracking is disabled The PID algorithm continues the normal calculation. OUT.Status is GoodC-IFS. When the output block goes to fault state, the upstream blocks are forced to Iman.
Man if Bad TRK_IN_D		Man	<ul style="list-style-type: none"> Output tracking is disabled; the mode is forced to Man The PID algorithm stops the calculation SP is passed to OUT unless the operator enters an OUT value
Target to Man if Bad TRK_IN_D	Man	Man	<ul style="list-style-type: none"> Output tracking is disabled; the target mode changes to Man The PID algorithm stops the calculation SP is passed to OUT unless the operator enters an OUT value

Bad TRK_VAL

PID_OPTS	Mode		Algorithm Action
	Target	Actual	
IFS if Bad TRK_VAL		"auto" -> Iman	<ul style="list-style-type: none"> Output tracking is disabled The PID algorithm continues the normal calculation. OUT.Status is GoodC-IFS. When the output block goes to fault state, the upstream blocks are forced to Iman.
Man if Bad TRK_VAL		Man	<ul style="list-style-type: none"> Output tracking is disabled; the mode is forced to Man The PID algorithm stops the calculation SP is passed to OUT unless the operator enters an OUT value
Target to Man if Bad TRK_VAL	Man	Man	<ul style="list-style-type: none"> Output tracking is disabled; the target mode changes to Man The PID algorithm stops the calculation SP is passed to OUT unless the operator enters an OUT value

Good TRK_IN_D
Good TRK_VAL

PID_OPTS	Mode		Algorithm Action
	Target	Actual	
All options		LO	Output Tracking is active, mode is LO (local override)
Target to Man if Tracking Active	Man	Man	Output Tracking is active, mode is Man
Target to Man on Power Up	Man	Man	Target mode set to Man on power up

6.2.3 Additional parameters

In addition to those listed in Chapter 6.1.10, the Enhanced PID Control block has the following parameters:

Parameter	Valid range/ Options	Default value	Description
BUMPLESS_TYPE		0	Algorithm action to start the output when the block transfers from a "manual" to an "automatic" mode <ul style="list-style-type: none"> 0: Bumpless 1: Last+Proportional 2: Bias 3: Bias+Proportional
BIAS		0	Bias value used in BUMPLESS_TYPE options Bias and Bias+Proprtional
PID_OPTS		0	Output tracking options
Legend: RO = Read Only			

6.3 Advanced PID Control

The Advanced PID Control block provides following enhancements with respect to the standard PID Control and Enhanced PID Control blocks:

- PID algorithm selection
- PI Sampling algorithm
- Adaptive gain
- Configurable limits for anti reset wind-up
- Special treatment of the error (control deviation)
- Discrete output to indicate the actual mode

When the default values of these enhanced parameters are used, the block exhibits identical behaviour to the PID Control block.

In ControlCare Application Designer the Enhanced PID Control block is assigned the generic name **Device Tag-APID-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 6-17.

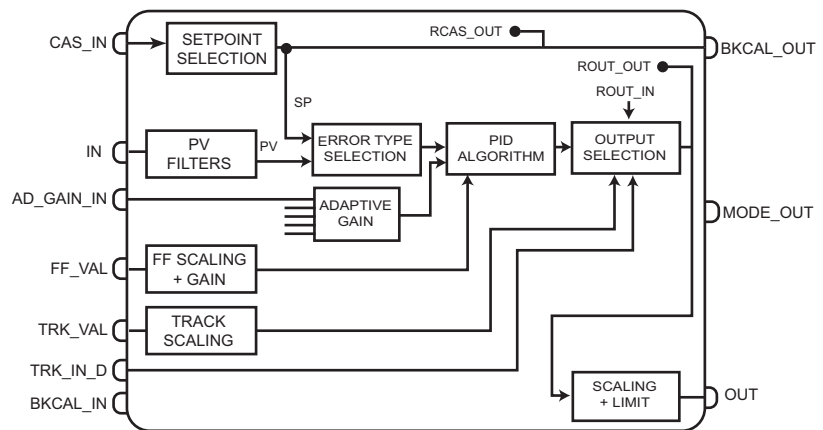


Fig. 6-17: Schematic diagram of APID Control block

6.3.1 PID algorithm

The algorithm to be used by the block when it calculates the **OUT** parameter is selected with the **PID_TYPE** parameter. The options are summarized in the table below:

PID_TYPE options	Description
PI.D + ISA	P and I terms are calculated based on the error and the D term on the PV (as PID Control block) $y = \text{GAIN} \cdot \left[e + \frac{1}{\text{RESET}} \cdot \int e \cdot \Delta t + \frac{\text{RATE}}{(1 + 0.13 \times \text{RATE})} \cdot \frac{\Delta \text{PV}}{\Delta t} \right] + F$
PI.D + Parallel	P and I terms are calculated based on the error and the D term on the PV $y = \text{GAIN}(e) + \frac{1}{\text{RESET}} \cdot \int e \cdot \Delta t + \frac{\text{RATE}}{(1 + 0.13 \times \text{RATE})} \cdot \frac{\Delta \text{PV}}{\Delta t} + F$
PI Sampling + ISA	P and I terms are calculated based on the error and the D term on the PV; the output is ramped according to the settings in PI Sampling) $y = \text{GAIN} \cdot \left[e + \frac{1}{\text{RESET}} \cdot \int e \cdot \Delta t + \frac{\text{RATE}}{(1 + 0.13 \times \text{RATE})} \cdot \frac{\Delta \text{PV}}{\Delta t} \right] + F$
PI Sampling + Parallel	P and I terms are calculated based on the error and the D term on the PV the output is ramped according to the settings in PI Sampling $y = \text{GAIN}(e) + \frac{1}{\text{RESET}} \cdot \int e \cdot \Delta t + \frac{\text{RATE}}{(1 + 0.13 \times \text{RATE})} \cdot \frac{\Delta \text{PV}}{\Delta t} + F$
PID + ISA	P, I and D terms are calculated based on the error $y = \text{GAIN} \cdot \left[e + \frac{1}{\text{RESET}} \cdot \int e \cdot \Delta t + \frac{\text{RATE}}{(1 + 0.13 \times \text{RATE})} \cdot \frac{\Delta e}{\Delta t} \right] + F$
PID + Parallel	P, I and D terms are calculated based on the error $y = \text{GAIN}(e) + \frac{1}{\text{RESET}} \cdot \int e \cdot \Delta t + \frac{\text{RATE}}{(1 + 0.13 \times \text{RATE})} \cdot \frac{\Delta e}{\Delta t} + F$
I.PD + ISA	I is calculated based on the error and the P and D terms on the PV $y = \text{GAIN} \cdot \left[\text{PV} + \frac{1}{\text{RESET}} \cdot \int e \cdot \Delta t + \frac{\text{RATE}}{(1 + 0.13 \times \text{RATE})} \cdot \frac{\Delta \text{PV}}{\Delta t} \right] + F$
I.PD + Parallel	I is calculated based on the error and the P and D terms on the PV $y = \text{GAIN}(e) + \frac{1}{\text{RESET}} \cdot \int e \cdot \Delta t + \frac{\text{RATE}}{(1 + 0.13 \times \text{RATE})} \cdot \frac{\Delta \text{PV}}{\Delta t} + F$

6.3.2 PI sampling

If the PI Sampling options are chosen, the output is ramped as illustrated in Fig. 6-18. The output is calculated based on the PI algorithm during time t_0 . After that, the algorithm stops calculating and it holds the last value during time t_1 .

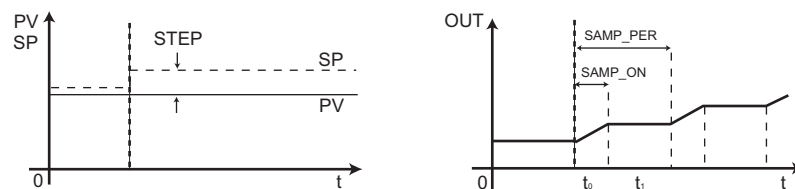


Fig. 6-18: Effect of step rise in SP on output when PI Sampling is active

Set the following parameters:

- Enter the calculation time t_0 in **SAMP_ON**
- Set sampling period for the ramp in **SAMP_PER**; $t_1 = \text{SAMP_PER} - \text{SAMP_ON}$.

If the sampling period **SAMP_PER** is less than the calculation time **SAMP_ON** or the latter is zero, then the algorithm works as an ordinary PI controller.

6.3.3 Adaptive gain

The adaptive gain option allows the PID terms in the algorithm to be modified according to function determined by the **CURVE_X** and **CURVE_Y** parameters. Depending upon the configuration, one or more of the PID constants GAIN, RESET and RATE are acted upon as follows:

- $GAIN' = G * GAIN$
- $RESET' = RESET / G$
- $RATE' = G * RATE$

Configuration

The function is configured with the following parameters:

Parameter	Action	Option/Value
AD_GAIN_ACTION	Define the PID parameter(s) on which the adaptive gain acts	Disable D I P PI PID
AD_GAIN_IN_SEL	Select the input value to be used for the adaptive gain calculation <ul style="list-style-type: none"> ■ If the AD_GAIN_IN option is selected the input value is supplied an upstream block ■ If the error option is selected, the ERROR_TYPE parameter must be set to "Normal" (default setting) 	SP PV Error OUT AD_GAIN_IN
CURVE_X	Enter up to twenty input points in the same engineering units as the selected variable	–
CURVE_Y	Enter the adaptive gain corresponding to each input point <ul style="list-style-type: none"> ■ If the curve has less than 20 configured points, the non-configured points are automatically set to +INF. 	–

Error handling

If the input/adaptive gain curve has a configuration error, it is indicated in **BLOCK_ALM** and the adaptive gain value assumes the CURVE_Y value corresponding to the highest CURVE_X point.

If the **AD_GAIN_IN** option is selected as the input value and this has bad status, the algorithm uses the last usable value to provide bumpless transfer.

6.3.4 Anti-reset wind-up

In order to prevent reset wind-up, LO and HI saturation limits can be imposed on the integral term by using the **ARW_LOW** and **ARW_UP** parameters respectively. When the output signal reaches the limits, the control algorithm stops the integral calculation. The proportional and derivative calculations are not affected.

The integral calculation is not stopped when **OUT** reaches the **OUT_LO_LIM** and **OUT_HI_LIM** values. For this reason, set **ARW_LOW** equal to or more than **OUT_LO_LIM** and **ARW_UP** equal to or less than **OUT_HI_LIM**.

6.3.5 Special treatment for error

Provided the option "error" has not been selected in **AD_GAIN_IN_SEL**, the **ERROR_TYPE** parameter provides various options for treatment of the error (or control deviation):

ERROR_TYPE option	Meaning
Normal (default)	Error type function is disabled
Quadratic (all terms)	Error term is applied to all PID calculations is a quadratic function of the input value
Quadratic (Integral)	Error term is applied to the integral calculation is a quadratic function of the input value
Special gain	Gain entered in the GAIN_BAND term is applied to the error when it lies within the range entered in ERROR_BAND, see Special gain

When a quadratic error option is chosen, the error is calculated as follows:

$$\hat{e} = e \cdot |e| / 100 \%$$

Fig 6-16 shows the effect of using this option.

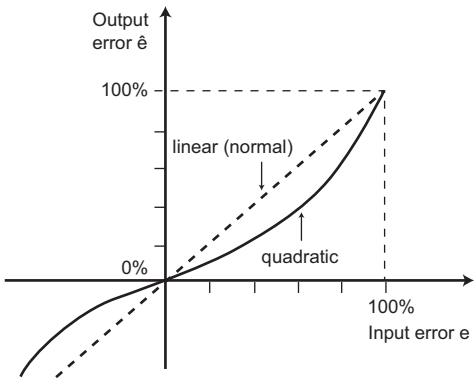


Fig. 6-19: Effect of quadratic error option in ERROR_TYPE parameter

Special gain

The special gain option allows adaptive control to be used in cases it would be normally unstable around the setpoint SP due the dead band of the actuator, noise or other factors. The option ramps the error over the band defined in **ERROR_BAND** according to the function:

$$\hat{e} = G \cdot e$$

Where G is the gain value entered in **GAIN_BAND**. Fig 6-20 shows the effects of this function.

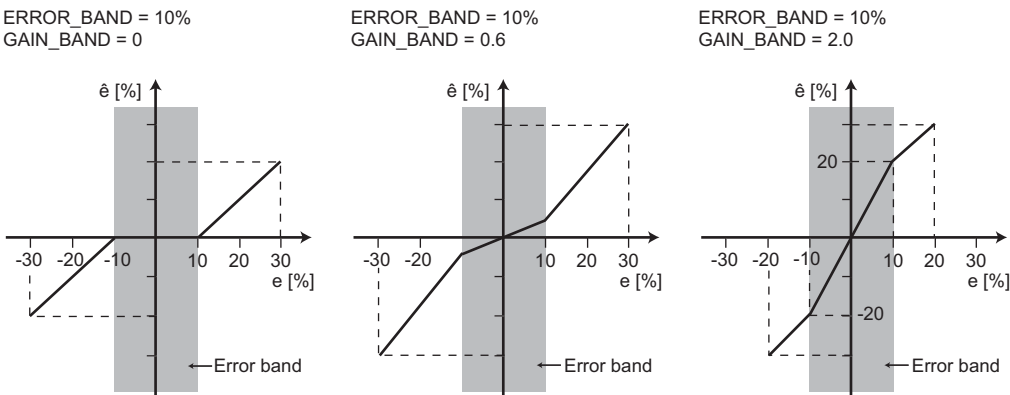


Fig. 6-20: Effect of ERROR_BAND and GAIN_BAND on error

6.3.6 Mode indication

The MODE_OUT parameter provides a discrete output that can be used to indicate the actual mode of the block. It outputs the value TRUE (=1) when block is operating in the the mode selected in the MODE_IND parameter. If more than one mode type is chosen in MODE_IND, OR logic is applied.

Parameter	Action	Option/Value
MODE_IND	Determines the actual block modes that will force the MODE_OUT parameter into TRUE status	OOS, IMAN LO MAN AUTO CAS RCAS ROUT

6.3.7 Additional parameters

In addition to those listed in Chapters 6.1.10 and 6.2.3, the Advanced PID Control block has the following parameters:

Parameter	Valid range/ Options	Default value	Description
MODE_OUT		RO	Discrete output indicating whether one of the modes selected in MODE_IND is the actual mode (TRUE/"1")
MODE_IND	All modes	OOS	Mode to be used by MODE_OUT, multiple selection supported
AD_GAIN_ACTION	–	Disable	PID parameter(s) on which the adaptive gain acts ■ Options: Disable, D, I, P, PI, PID
AD_GAIN_IN_SEL	–	SP	Input value to be used for the adaptive gain calculation ■ Options: SP, PV, Error, OUT, AD_GAIN_IN
AD_GAIN_IN		–	Input value provided by another block for adaptive gain when option "AD_GAIN_IN" is chosen in AD_GAIN_IN_SEL
CURVE_X	Max. 20	0	Input values of the adaptive gain characteristic ■ Engineering units as variable selected in AD_GAIN_IN_SEL
CURVE_Y	Max. 20	0	Output values of the adaptive gain characteristic ■ No. of output points must correspond to no. of input points
ERROR_TYPE		Normal	Type of error used by PID algorithm ■ Options: Normal, quadratic (all terms), quadratic (Integral), special gain
ERROR_BAND	0–300	0	Error band applicable to "special gain" option in ERROR_TYPE
GAIN_BAND	0–10	0	Gain band applicable to "special gain" option in ERROR_TYPE
PID_TYPE		PI.D+ISA	PID algorithm used by the block ■ Options: PI.D+ISA, PID+ISA, I.PD+ISA, PI Sampling+ISA, PI.D+Parallel, PID+Parallel, I.PD+Parallel, PI Sampling+Parallel
SAMP_ON		0 s	Active sampling time for "PI Sampling" options in PID_TYPE
SAMP_PER		0 s	Sampling period of "PI Sampling" options in PID_TYPE
BUMPLESS_TYPE		0	Type of transfer from "manual" mode to "automatic" mode ■ Options: Bumpless, Last+Proportional, Bias, Bias+Proportional
BIAS		0	Bias value for "Bias" options in BUMPLESS_TYPE
ARW_UP		+INF	High limit for anti reset windup
ARW_LOW		–INF	Low limit for anti reset windup
Legend: RO = Read Only			

6.4 Step Output PID

The Step Output PID block is used to manipulate final control elements that have an actuator driven by an electric motor. In addition to providing PID control, the block acts as transducer output for a ControlCare discrete output module connected to the fieldbus actuator or switch gear. Depending upon the value of the PID output, signals are provided at the output that drive the motor clockwise or anticlockwise to position the final control element. When the required position is reached, the motor is stopped.

In ControlCare Application Designer the Step Output PID block is assigned the generic name **Device Tag-STEP-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 6-21.

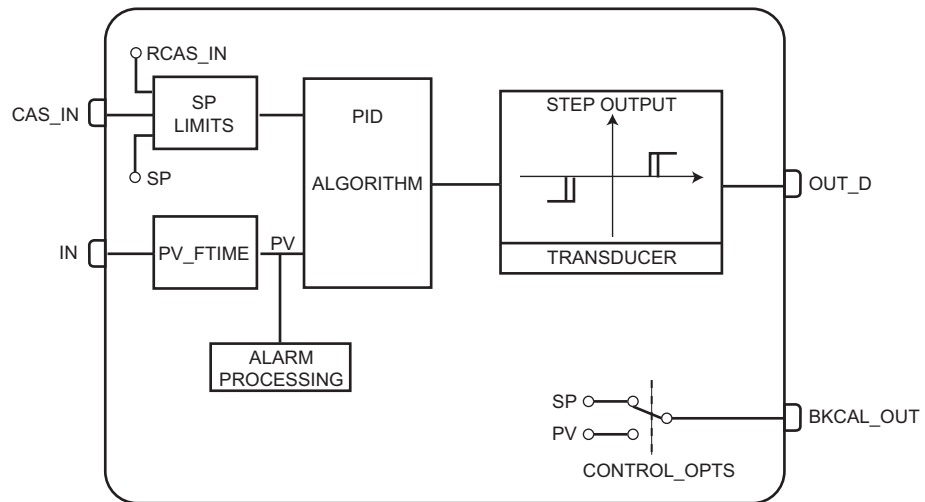


Fig. 6-21: Schematic diagram of Step Output PID block

6.4.1 Functional description

The Step Output PID block combines two basic functions:

- Basic PID control as described in Section 6.1, without feedforward, tracking and bypass functions, providing a manipulating signal for an electric motor
- Discrete output with valve actuator transducer block functions

It can be used as a positioner within a cascade loop or as a step controller.

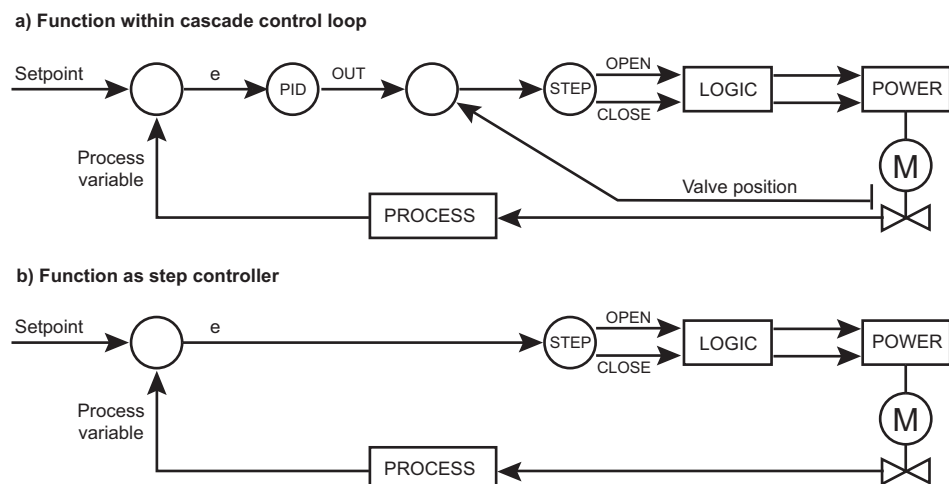


Fig. 6-22: Functional diagram of step control

PID control

The PID control uses the travel time of the actuator, **TRAVEL_TIME**, i.e. the time it takes to drive the final control element from one end limit to another, as the basis for signal output. The proportional action will actuate the final control element in the required direction for a time t_p proportional to:

$$t_p = \text{GAIN} * (\hat{e} / 100) * \text{TRAVEL_TIME} \dots (\text{s})$$

where \hat{e} is the difference between the setpoint SP and the process value PV. If the proportional action is not enough force \hat{e} to zero, the integral action moves the final control element at a speed of:

$$V = \text{GAIN} * \hat{e} / \text{RESET} \dots (\% / \text{s})$$

where the **RESET** is the integral time constant in seconds.

As most actuators work at a constant and fixed speed, they cannot move faster than:

$$\text{Maximum speed} = 100\% / \text{TRAVEL_TIME} \dots (\% / \text{s})$$

Slower responses required by the integral action are therefore obtained by generating pulses of a specified duration **PULSE_DUR**. Each pulse will move the final control element a distance $\Delta x \%$ in the required direction.

$$\Delta x \% = \text{PULSE_DUR} * 100\% / \text{TRAVEL_TIME} \dots (\%)$$

The pulse frequency f is given by:

$$f = V / \Delta x \% \dots (\text{pulses} / \text{s})$$

The derivative or rate action is given by:

$$t_D = \text{GAIN} * (d\hat{e}/dt) * \text{RATE}$$

Where **RATE** is the derivative time constant in seconds.

The Step Output PID activates the OPEN or CLOSE signals according to the modified deviation, \hat{e} , the PID parameters and the other parameters as follows:

- The signal is activated during a time equivalent to $t_p + t_D$
- If the modified deviation is not equal to zero, the integral or reset action will give pulses with a duration defined by **PULSE_DUR** and a frequency calculated by "f."
- t and f are dynamically modified by \hat{e} .

In order to avoid reset wind-up, the actuation time in one direction is integrated and limited. If the actuation time in one direction is larger than the **TRAVEL_TIME**, the output signal is not pulsed, but is continuously activated.

The block provides a full PV and deviation alarm support as described in Chapter 6.1.

Gap deviation

In order to prevent jitter around the desired valve position, the Step Output PID block also offers the possibility of gap deviation action, see Fig. 6-23. When **DEAD_BAND** is greater than zero, a zone ± 0.5 of its value is defined around the point of zero deviation, in which a change in the actual deviation e produces no effect on the modified deviation \hat{e} and thus no corrective action by the motor. The transition from either end of the dead band is via a **HYSTERESIS**.

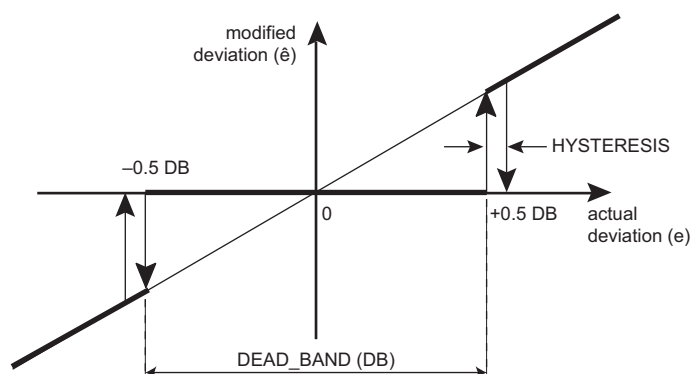


Fig. 6-23: Gap deviation using the **DEAD_BAND** and **HYSTERESIS** parameters

Discrete output

The **CHANNEL** parameter selects the output channel of the connected ControlCare module to be used by the open signal. The adjacent output, i.e. "**CHANNEL** + 1" is automatically reserved for the close signal. **OUT_D** provides an indication of the current signal being provided to the actuator:

OUT_D.value	Action	Signal at CHANNEL	Signal at CHANNEL + 1
0	Stop	0	0
1	Close	0	1
2	Open	1	0

The initial value of the output signal is zero, i.e. there is no action in either direction and the integral action value is also zero.

Normally, the **BKCAL_OUT** parameter of the Step Output PID block carries the value and status of **SP**. Depending on the **CONTROL_OPTS** parameter, it may be forced to a different condition.

When the block operates in RCAS, **RCAS_OUT** (= SP) provides the back calculation for the host, allowing action to be taken under limiting conditions or mode change.

CONTROL_OPTS option	BKCAL_OUT value and status
None	SP value and status
None, but RCAS mode	PV value and status after limiting
Use PV for BKCAL_OUT	PV value and status

Status propagation

The status of the output reflects the worst quality of the status of any connected input. The behaviour can be changed by selecting an appropriate option in the **STATUS_OPTS** parameter:

Option	Description
Uncertain as Good	If the status of the IN parameter is Uncertain, treat it as Good.
Propagate Fail Backward	If the status from the actuator is Bad "Device failure" or "Fault State Active", or Local Override is active, propagates this as Bad "Device Failure" or Good Cascade "Fault State Active", or Local Override to BKCAL_OUT respectively without generating an alarm.
Target to Manual if BAD IN	Sets the target mode to Man if the status of the IN parameter is BAD.
Target to next permitted mode if BAD CAS_IN	Sets the target mode to next permitted mode if the target mode is CAS and the status of CAS_IN is BAD.

The **SHED_OPT** parameter provides options which govern the recovery of the block after a change in mode e.g. due to a timeout.

The block also supports fault state handling as described in Chapter 6.5. When used in a cascade loop, it can be configured to respond to the IFS status by setting **FSTATE_TIME**, in units of 1/32 ms, to the time the output block can normally wait before responding to a fault state in or loss of communication with the upstream block. **FSTATE_VAL_D** is set to the set point value (SP_D = OUT_D) that causes the output device to assume fail-safe position.

The **IO_OPTS** parameter determines the block response to a fault state, whereby multiple choices are allowed. The option is set by ticking the selection box:

Option	Set	Clear)
Fault State to value	On fault state, FSTATE_VAL_D substitutes SP_D	On fault state, the last valid SP_D value is used
Target to Man if IFS	On fault state, the block mode is forced to Man. In this mode, parameters can be changed	On fault state, the block mode is forced to Local Override, parameters cannot be changed until the fault state is cleared

A third option, "Use Fault State value on restart" causes the value of **FSTATE_VAL_D** to be used on device restart. In this case, the value is used, although no fault state exists.

6.4.2 Block configuration

Fig. 6-24 shows a typical application for the Step Output PID block. The block is connected to the actuator switchgear via a ControlCare I/O module, in this example the SFC428 high density NO relay module, which supplies "Open" and "Close" signals according to the demand of the control loop. The switchgear powers and reverses the electric motor accordingly.

In addition, to the "Open" and "Close" signals, most of the electric actuators require an interlock circuit to prevent the motor from overheating when the actuator reaches one of its travel limits or when something blocks its movement, increasing the torque beyond an established limit. Such protection is normally provided by torque switches and limit switches. Other inputs might be a local override control and an enabling signal. These signals are not considered in the applications described here, but could be realized by the appropriate ControlCare module and strategy.

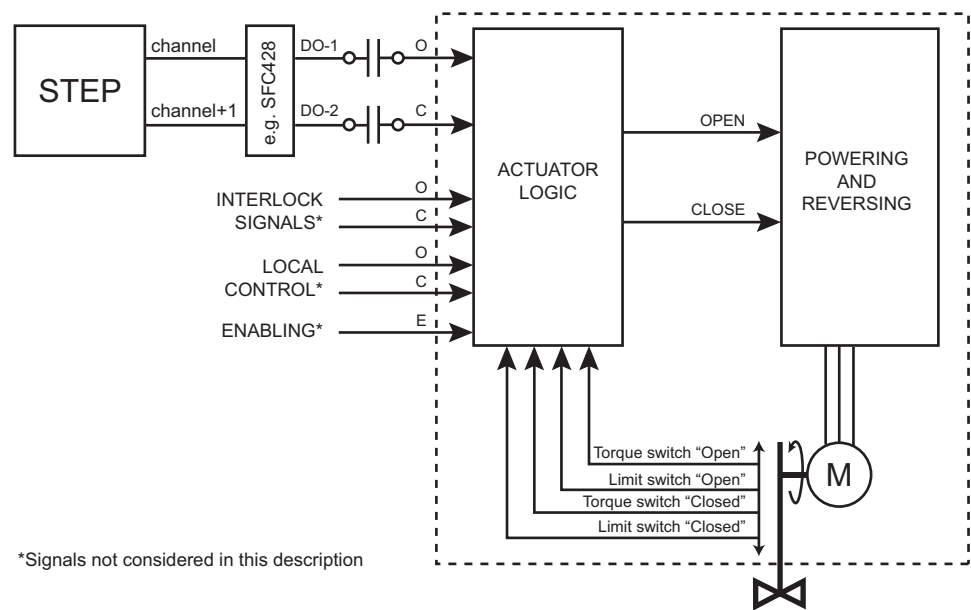


Fig. 6-24: Use of STEP PID Output block with motor switchgear

Closed-loop control

Fig. 6-25 shows a control strategy that uses the Step Output PID block for the simple closed-loop control example described in Fig. 6-2 of Chapter 6.1.2.

*

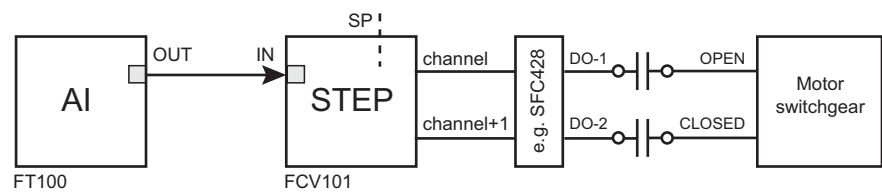


Fig. 6-25: Links required for basic closed-loop control with Step Output PID

Block configuration

The block is configured for operation without gap deviation as follows:

Parameter	Function	Example value
MODE_BLK.Target	Normal operating mode of block	Auto
SP	Setpoint for ideal process control	60%
STATUS_OPTS	Options for handling IN status	Uncertain as Good
PV_SCALE.EU_100	Upper range limit for process variable	100
PV_SCALE.EU_0	Lower range limit for process variable	0
PV_SCALE.UNITS_INDEX	Unit of process variable	%
SP_RATE_DN	Rate of change from old to new, higher SP	5
SP_RATE_UP	Rate of change from old to new, lower SP	5

Parameter	Function	Example value
GAIN RESET BAL_TIME RATE	Tuning constants for the P, I and D terms of the PID block respectively.	1.5 0.1 s 5 s 0.5 s
CHANNEL	Rack+Slot+Group+I/O point for OPEN signal ■ CLOSE signal at CHANNEL+1, e.g. 2101	e.g. 2100
SHED_OPTS	Mode shedding recovery options	Normal Shed, Normal Return
TRAVEL_TIME	Time (s) to drive from one end position to the other	3 s

Tab. 6-2: Basic parameters for closed-loop control

Cascade control

Fig. 6-26 shows a control strategy that uses the Step Output PID block for the cascade control example described in Fig. 6-4 of Chapter 6.1.4.

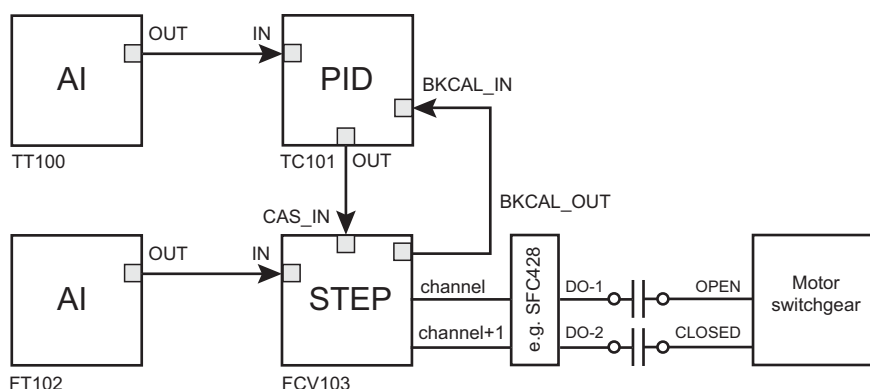


Fig. 6-26: Links required for cascade control with Step Output PID

Block configuration

The blocks are configured with gap deviation as follows:)

Parameter	Function	Example value
PID Block		
MODE_BLK.Target	Normal operating mode of block	AUTO
SP	Setpoint for ideal process control	60%
PV_SCALE.EU_100 PV_SCALE.EU_0 PV_SCALE.UNITS_INDEX	Upper range limit for process variable Lower range limit for process variable Unit of process variable	100 0 %
OUT_SCALE/EU_100 OUT_SCALE/EU_0 OUT_SCALE/UNITS_INDEX	Upper range limit for output variable Lower range limit for output variable Unit of output variable	100 0 %
CONTROL_OPTS	Sets control options for bad input	Bypass Enable
SP_RATE_DN SP_RATE_UP	Rate of change from old to new, higher SP Rate of change from old to new, lower SP	3 3
GAIN RESET RATE	Tuning constants for the P, I and D terms of the PID block respectively.	1.5 0.1 s 0.5 s
SHED_OPT	Behaviour when shedding from remote mode	Normal shed, normal return
STEP block		
MODE_BLK.Target	Normal operating mode of block	Cas
STATUS_OPTS	Options for handling IN status	Target to Manual if BAD IN
PV_SCALE.EU_100 PV_SCALE.EU_0 PV_SCALE.UNITS_INDEX	Upper range limit for process variable Lower range limit for process variable Unit of process variable	100 0 %
SP_RATE_DN SP_RATE_UP	Rate of change from old to new, higher SP Rate of change from old to new, lower SP	0 0

Parameter	Function	Example value
GAIN RESET BAL_TIME RATE	Tuning constants for the P, I and D terms of the PID block respectively.	1.5 0.1 s 5 s 0.5 s
CHANNEL	Rack+Slot+Group+I/O point for OPEN signal ■ CLOSE signal at CHANNEL+1, e.g. 2101	e.g. 2100
SHED_OPTS	Mode shedding recovery options	Normal Shed, Normal Return
TRAVEL_TIME	Time (s) to drive from one end position to the outer	3 s
DEAD_BAND	Interval, within which a change in input will not change the output	2%
HYSTERESIS	Difference between the switching points dependent on direction of action	1%

6.4.3 Block operation

Mode

The block supports the modes Out-of-service, Manual, Auto, Cas and Rcas.

In Rcas mode the block setpoint is set by a control application running on a computer, DCS or PLC, In Cas (cascade) mode, the by another function block through the **CAS_IN** parameter.

In Auto mode the setpoint value **SP** is provided by the operator and can be changed and downloaded while the block is operating.

When **MODE_BLK.Target** is set to "Man", the block stops output calculation. **OUT_D** can be set by the operator. In addition, the parameter **JOG_TIME** allows the operator to determine how long the selected **OUT_D** action is to be active.

If parameters other than **SP** are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Operation can be checked on-line by viewing a number of parameters:

IN	Value and status of the process value
SP, CAS_IN, RCAS_IN	Value and status of setpoint used to execute the block in Auto, Cas and RCas modes
PV	Value and status of the process value used to execute the block
OUT_D	Value and status of the current output action (0 = stop, 1 = open, 2 = close)

Status

The significance of the output status is indicated below.

OUT_D status	Meaning
Good	<ul style="list-style-type: none"> ■ Normal operation as configured ■ Input uncertain but option "Uncertain as Good" selected in STATUS_OPTS
Bad	<ul style="list-style-type: none"> ■ Block out of service ■ Input bad

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Out-of-Service	<ul style="list-style-type: none"> ■ MODE_BLK.Target currently set to OOS

6.4.4 Block parameters

Parameter	Valid range/ Options	Default value	Description
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set MODE_BLK.Target to AUTO, CAS or RCAS
BLOCK_ERR	RO		Block errors, see Chapter 6.1.9
PV	RO		Process value used in executing the block
SP			Setpoint used by the block in AUTO mode (PV_SCALE \pm 10%)
OUT_D			Discrete output value calculated as a result of executing the block
PV_SCALE		0-100%	PV and SP value scaling to required engineering units
XD_STATE		0-100%	Discrete state of the value obtained from the transducer.
GRANT_DENY		0	Access options, see Chapter 2.8.1
CONTROL_OPTS		0	Control options, see Chapter 2.8.4 and Chapter 6.1.1
STATUS_OPTS		0	Status options, see Chapter 2.8.4 and Chapter 6.1.9
IN			Process value
PV_FTIME		0	Time constant of a single exponential filter for process value
JOG_TIME		0	Duration of OUT_D in the active state when commanded by the operator to jog open or jog closed
CAS_IN			Setpoint used by the block in CAS mode
SP_RATE_DN	Positive	+INF	Ramp rate for positive setpoint change in PV units/s
SP_RATE_UP	Positive	+INF	Ramp rate for negative setpoint change in PV units/s
SP_HI_LIM		100	High limit for setpoint limiting (PV_SCALE \pm 10%)
SP_LO_LIM		0	Low limit for setpoint limiting (PV_SCALE \pm 10%)
GAIN		0	Proportional term of the PID (Kp value)
RESET	Positive	+INF	Integral term of the PID (Tr value)
BAL_TIME	Positive	0	Time allowed to balance OUT changes as a result of a change from Auto, Cas or Rcas to Man
RATE	Positive	0	Derivative term of the PID (Td value)
IO_OPTS			See Chapter 2.8.2
CHANNEL			Index of transducer block logical channel used by the block
FSTATE_TIME		0	Time in seconds from detection of a fault in the output block remote setpoint to the output action of the block output, if the condition still exists
FSTATE_VAL_D		0	Preset discrete SP_D value to be used when fault occurs, provided I/O option "Fault State to value" is selected.
BKCAL_OUT			Feedback for BKCAL_IN of an upstream block.
RCAS_IN			Setpoint used by the block in RCAS mode (from host)
SHED_OPT	1 - 8	0	Mode shedding options, see Chapter 6.1.6
RCAS_OUT			Setpoint and status for host back calculation in RCAS mode
TRAVEL_TIME		60	Time in seconds required by the actuator to drive the final control element from one end position to the other
PULSE_DUR		1	Width, in seconds, of the pulses given due to the integral action
DEAD_BAND		0	Interval, within which a change in input will not change the output
HYSTERESIS		0	Difference between the switching points dependent on direction of action
UPDATE_EVT			This alert is generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
ALARM_SUM			Alarm summary, see Chapter 2.9
ACK_OPTION	0, 1	0	See Chapter 2.9 <ul style="list-style-type: none"> ■ 0: Auto ACK Disable ■ 1: Auto ACK Enable
ALARM_HYS	0 to 50 %	0.5%	Alarm hysteresis in % of OUT_SCALE, see Chapter 2.9
HI_HI_PRI	0 to 15		Priority of the HI_HI alarm, see Chapter 2.9
HI_HI_LIM		+INF	Setting of HI_HI alarm in engineering units, see Chapter 2.9.

Parameter	Valid range/ Options	Default value	Description
HI_PRI	0 to 15		Priority of the HI alarm, see Chapter 2.9
HI_LIM		+INF	Setting of HI alarm in engineering units, see Chapter 2.9.
LO_PRI	0 to 15		Priority of the LO alarm, see Chapter 2.9.
LO_LIM		-INF	Setting of LO alarm in engineering units, see Chapter 2.9.
LO_LO_PRI	0 to 15		Priority of the LO_LO alarm, see Chapter 2.9
LO_LO_LIM		-INF	Setting of LO_LO alarm in engineering units, see Chapter 2.9
DV_HI_PRI	0 to 15	0	Priority of the control deviation HI alarm, see Chapter 2.9
DV_HI_LIM		+INF	Setting of control deviation HI in engineering units
DV_LO_PRI	0 to 15	0	Priority of the control deviation LO alarm, see Chapter 2.9
DV_LO_LIM		-INF	Setting of control deviation LO in engineering units
HI_HI_ALM			Status of HI_HI alarm and its associated time stamp.
HI_ALM			Status of HI alarm and its associated time stamp.
LO_ALM			Status of LO alarm and its associated time stamp.
LO_LO_ALM			Status of LO_LO alarm and its associated time stamp.
DV_HI_ALM			Status of deviation HI alarm and its associated time stamp.
DV_LO_ALM			Status for deviation LO alarm and its associated time stamp.
Legend: RO = Read Only			

7 Calculation and Logic Blocks

7.1 Arithmetic

The arithmetic block allows process values from two or more Analog Input blocks to be combined mathematically to produce a calculated output. It supports two basic functions:

- Range extension, by combining the outputs of a high-range and low-range measuring device
- Calculated variable, by combining the process variable with the outputs of up to three auxiliary measuring devices. Nine different calculated variables can be selected.

The block also allows the calculation of a variable from an input with an extended range.

The arithmetic block is not intended to be used in a control path, so that it does not support control status propagation or back calculation. It has no process alarms.

In ControlCare Application Designer the Arithmetic block (AR block) is assigned the generic name **Device Tag-ARTH-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 7-1.

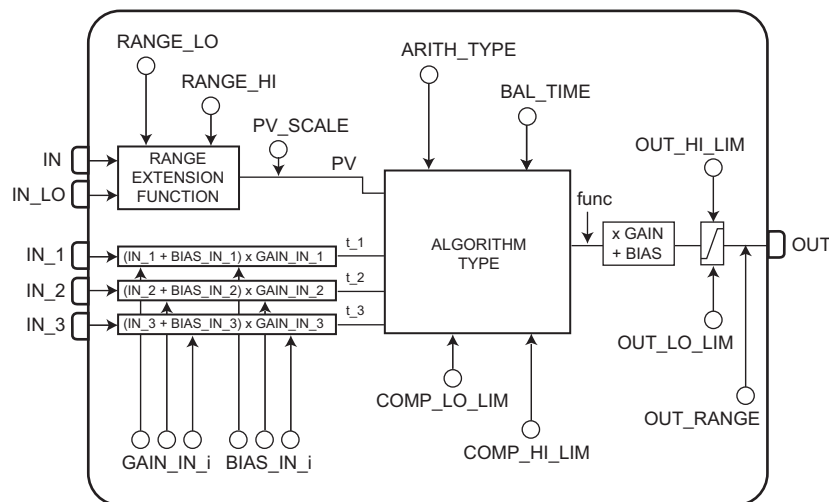


Fig. 7-1: Schematic diagram of ARTH block (AR Block).

7.1.1 Range extension

The range extension function ensures a gradual transfer between e.g. a standard measuring device connected to **IN** and low range measuring device connected to **IN_LO**. The function is controlled by two constants referenced to **IN** according to the following equation, see also Fig. 7-2:

$$PV = g \cdot IN + (1-g) \cdot IN_LO$$

where g is:

- zero when IN is less or equal to $RANGE_LO$
- one when IN is greater or equal to $RANGE_HI$
- $IN - RANGE_LO / (RANGE_HI - RANGE_LO)$ in the transition zone

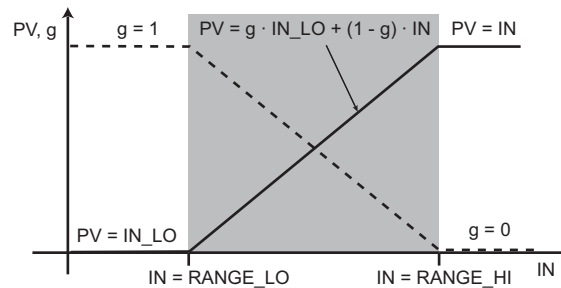


Fig. 7-2: Transition function for range extension

After combination, the extended range is scaled in **PV_SCALE** before being passed on to the arithmetic algorithm. If no option has been selected in **ARITH_TYPE** (=default), **PV** is passed on unchanged to the output calculation:

$$\blacksquare OUT = GAIN \cdot PV + BIAS$$

If **GAIN** is set to one and the **BIAS** to zero (default), **PV** is again passed unchanged to **OUT**. **OUT_RANGE** determines the range and units of **OUT**. The output may be limited by using the **OUT_HI_LIM** and **OUT_LO_LIM** parameters.

Status propagation

The status of **PV** is handled according to the following table. Provided one input value is good in the transition zone, **PV** will always have the status good.

IN_LO status	IN status	IN value	PV value	PV status
Bad, Uncertain	Good	$IN < RANGE_LO$	$PV = IN_LO$	Bad, Uncertain
Bad, Uncertain	Good	$IN > RANGE_LO$	$PV = IN$	Good
Good	Good	$RANGE_LO < IN < RANGE_HI$	$PV = g \cdot IN + (1-g) \cdot IN_LO$	Good
Good	Bad, Uncertain	$IN_LO > RANGE_HI$	$PV = IN$	Bad, Uncertain
Good	Bad, Uncertain	$IN_LO < RANGE_HI$	$PV = IN_LO$	Good

If the options "IN_LO Use uncertain" and "IN Use uncertain" are selected in the **INPUT_OPTS** parameter, then the status "Uncertain" will not force **PV** to **IN** or **IN_LO** within the transition zone, but the transition equation will be used. The status of **PV** will be "Good".

Control strategy

Fig. 7-3 shows the control strategy together with the links that must be made between the blocks:

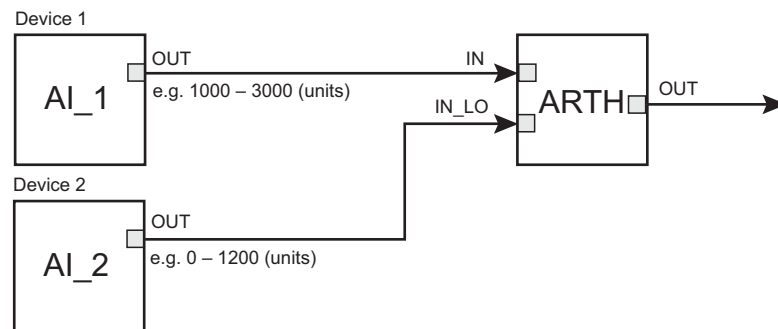


Fig. 7-3: Links required for range extension

The inputs **IN** and **IN_LO** must be linked to **OUT** signals that have the same engineering units.

Block configuration

The arithmetic block is configured for range extension as follows:

Parameter	Action	Option/Value
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
PV_SCALE.EU_100 PV_SCALE.EU_0 PV_SCALE.UNITS_INDEX	Enter upper range limit of extended range Enter lower range limit for extended range Enter unit of process variable	e.g. 3000 units e.g. 0 units units
OUT_RANGE.EU_100 OUT_RANGE.EU_0 OUT_RANGE.UNITS_INDEX	Enter upper range limit for output variable Enter lower range limit for output variable Enter unit of output variable	e.g. 3000 units* e.g. 0 units* units*
INPUT_OPTS	If required, select the input options by ticking the box <ul style="list-style-type: none"> Multiple selections are allowed See status handling for more information 	IN Use uncertain IN_LO Use uncertain
RANGE_LO	Enter the PV value at which transition from IN_LO to IN begins	e.g. 1050 units
RANGE_HI	Enter the PV value at which transition from IN_LO to IN ends	e.g. 1150 units
BIAS	If required, enter a constant to be used in the calculation of OUT <ul style="list-style-type: none"> e.g. 273 for conversion °C to K 	0 units (default)*
GAIN	Enter the scaling value to be used in the calculation of OUT	1*
ARITH_TYPE	Do not make an entry here (= default) unless PV is to be used to produce a calculated value, see Chapter 7.1.2	0 (default)
OUT_HI_LIM	If required, enter upper limit to be placed on OUT value	–
OUT_LO_LIM	If required, enter lower limit to be placed on OUT value	–
*If BIAS is other than 0 or the GAIN is other than 1, the OUT_RANGE parameters must be adjusted accordingly		

7.1.2 Calculated variables

The Arithmetic Block contains nine algorithms that find frequent use in process engineering. The functions together with examples of use are to be found in Table 7-1 below. All algorithms use the process value **PV** supplied by **IN** as well as up to three auxiliary inputs, **IN_1**, **IN_2** and **IN_3**.

Function	Function/Example	IN (PV)	IN_1 (t_1)	IN_2 (t_2)	IN_3 (t_3)
Flow compensation, linear.	func = (t_1 / t_2), limited*				
	e.g. $Q_b = Q_f \cdot K \cdot P/T$	Q_f in m ³ /h	P in MPa	T in K	–
Flow compensation, square root.	func = $\sqrt{t_1 / (t_2 \cdot t_3)}$, limited*				
	e.g. $Q_b = Q_f \cdot K \cdot \sqrt{(P/T \cdot Z)}$	Q_f in m ³ /h	P in MPa	T in K	Compress. Z
Flow compensation, approximate	func = $\sqrt{(t_1 + t_2 + t_3^2)}$, limited*				
	The square root of the third power is available by connecting input to IN and IN_1. The square root of the fifth power is available by connecting the input to IN, IN_1 and IN_3.				
	e.g. $Q_b = Q_f \cdot K_1 \cdot \sqrt{(K_2 + K_3T + K_4T^2)}$	Q_f in m ³ /h	K ₂	T in K	T in K
	e.g. $Q_b = Q_f \cdot K_1 \cdot \sqrt{(K_2 + K_3P)}$	Q_f in m ³ /h	K ₂	P in MPa	–
BTU flow	func = t_1 - t_2				
	e.g. $Q_{heat} = K \cdot Q_{vol} \cdot (T_2 - T_1)$	Q_v in m ³ /h	T ₂	T ₁	–
Traditional Multiply Divide	func = (t_1 / t_2) + t_3, limited*				
	$Q_{sp} = Q_f \cdot \text{Ratio}$	Q_f	Quotient	Devisor	–
Average	func = (PV + t_1 + t_2 + t_3)/n where n = number of inputs used				
	e.g. $T_a = (T_1 + T_2 + T_3 + T_4)/4$	T ₁	T ₂	T ₃	T ₄
Traditional Summer.	func = PV + t_1 + t_2 + t_3				
	e.g. $P_{diff} = P_1 - P_2$	P ₁	P ₁	–	–
Fourth order polynomial	func = $PV + t_1^2 + t_2^3 + t_3^4$				
	e.g. $\rho_T = A + K_1T + K_2T^2 + K_3T^3$	T in K	T in K	T in K	–
Simple HTG compensated level,	func = (PV - t_1) / (PV - t_2)				
	$l_{bt} = l_{bm} \cdot (P_b - P_t) / (P_b - P_m)$ where P _b is the tank base pressure, P _t is the top pressure, P _m is the density correction pressure, l _{bm} is the GAIN (height of density tap)	P _b	P _t	P _m	–

*limited: the compensation function must be limited by the COMP_LO_LIM and COMP_HI_LIM parameters

Tab. 7-1: Algorithms and examples of use (units may differ from those shown here)

Process value

The process value **PV** supplied by **IN** is scaled in **PV_SCALE** before being passed on to the arithmetic algorithm. **PV_SCALE** will normally reflect the **OUT_SCALE** of the upstream block, and must be in the units required for calculation. If a range extension has been enforced, the extended range limits must be entered.

Depending upon the algorithm chosen in **ARITH_TYPE**, the output is generated by multiplying **PV** by the result of the function calculation, f, or **PV** is itself part of the function. The **OUT** value is therefore calculated either as:

$$\text{OUT} = \text{PV} \cdot f \cdot \text{GAIN} + \text{BIAS} \quad (1)$$

or

$$\text{OUT} = f \cdot \text{GAIN} + \text{BIAS} \quad (2)$$

The **GAIN** parameter can be used to simulate factors, e.g. the first order coefficient in the polynomial or height of the density tap, see Table 7-1. The **BIAS** parameter can be used to add or subtract constants from the final result. The result of the calculation function is always displayed in **PRE_OUT**.

For compensated outputs of type (1) the algorithm itself must be limited by entering appropriate values in **COMP_LO_LIM** and **COMP_HI_LIM**. The parameter **BAL_TIME** is used to provide bumpless transfer from Auto to Man mode, the output being ramped from the last **OUT** value to the new value in the time entered. **OUT** itself may also be limited by the use of **OUT_LO_LIM** and **OUT_HI_LIM**.

Auxiliary inputs

Three inputs **IN_1**, **IN_2** and **IN_3**, connected to the **OUT** parameter of an upstream block, are available for the calculation. Ideally, **OUT** has the engineering units required for the calculation. Each input has a **BIAS_IN_i** and a **GAIN_IN_i** parameter. **BIAS** is added to the input and **GAIN** is applied to the sum. The result is an internal value called **t_i** in the function equations:

$$t_i = (IN_i + BIAS_IN_i) \cdot GAIN_IN_i$$

BIAS is used to correct for absolute temperature, pressure etc.. **GAIN** can be used to represent a coefficient. The internal values **t_i** are used in the calculations as shown in Table 7-1.

Status propagation

The table below summarizes the status, which will normally follow that of PV, unless one or more of the auxiliary inputs is "Bad" or "Uncertain" . The status of unused inputs ignored.

IN_i status	IN status	OUT status
Good	Good	Good
Good	Bad, Uncertain	Bad, Uncertain
Bad, Uncertain	Good	Uncertain

If the options "IN_i Use uncertain" and/or "IN_i Use bad" are selected in the **INPUT_OPTS** parameter, then the corresponding input status will not force OUT to "Uncertain". Instead the status of OUT will be "Good". Similarly, The option "IN Use uncertain" will force OUT to "Good" if the status of PV is "Uncertain".

Control strategy

The links to the block inputs depend on the selected function. Fig. 7-4 shows those for a third order polynomial, for which all inputs come from the same source:

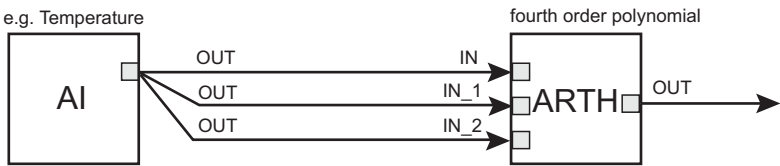


Fig. 7-4: Links for third order polynomial (an additional link to IN_3 is required for fourth order polynomial)

Fig. 7-5 shows the links for linear flow compensation. In this case each of the inputs comes from a separate analog input block that has been scaled to provide the correct engineering units for the selected algorithm.

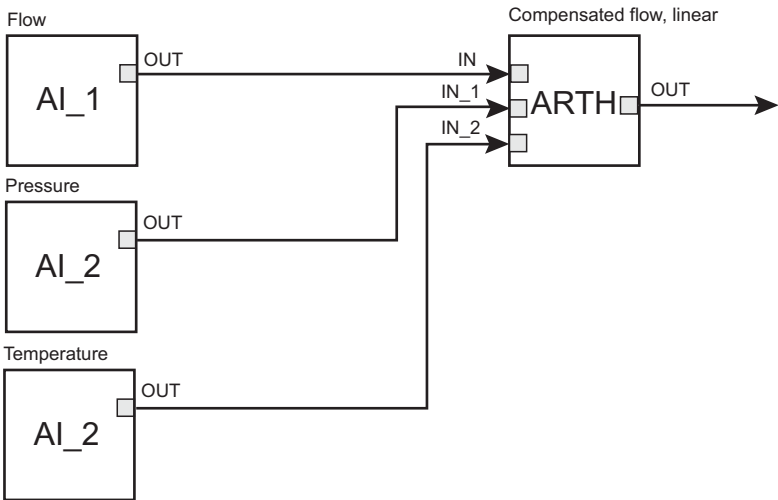


Fig. 7-5: Links required for linear flow compensation

Block configuration

The block is configured as follows:

Parameter	Action	Option/Value
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
PV_SCALE.EU_100	Enter upper range limit of PV	e.g. 2000 units
PV_SCALE.EU_0	Enter lower range limit of PV	e.g. 0 units
PV_SCALE.UNITS_INDEX	Enter unit of process variable	units
OUT_RANGE.EU_100	Enter upper range limit for output variable	e.g. 3000 units
OUT_RANGE.EU_0	Enter lower range limit for output variable	e.g. 0 units
OUT_RANGE.UNITS_INDEX	Enter unit of output variable	units
INPUT_OPTS	If required, select the input options by ticking the box <ul style="list-style-type: none"> Multiple selections are allowed See status handling for more information 	IN Use uncertain IN_i Use uncertain IN_i Use bad
BIAS_IN_1	Enter bias to be applied to first auxiliary input	0, see below
GAIN_IN_1	Enter gain to be applied to first auxiliary input	>0, see below
BIAS_IN_2	Enter bias to be applied to second auxiliary input, if used	0, see below
GAIN_IN_2	Enter gain to be applied to second auxiliary input, if used	>0, see below
BIAS_IN_3	Enter bias to be applied to third auxiliary input, if used	0, see below
GAIN_IN_3	Enter gain to be applied to third auxiliary input, if used	>0, see below
COMP_HI_LIM	Enter upper limit to which compensation applies <ul style="list-style-type: none"> Applies to the first five ARITH_TYPE functions 	e.g. 2000 units
COMP_LO_LIM	Enter upper limit to which compensation applies <ul style="list-style-type: none"> Applies to the first five functions in Table 7-1 only 	e.g. 0 units
ARITH_TYPE	Tick the function to be executed by the arithmetic block	Flow, linear Flow, sq root Flow, approx BTU flow Multiply/Divide Average Summer 4th order polynomial Compensated level
BAL_TIME	Enter the time in seconds for OUT to adjust to a new value if the block mode changes from Man to Auto	10 s
BIAS	If required, enter a constant to be used in the calculation of OUT	0 units (default), see below
GAIN	Enter the scaling value to be used in the calculation of OUT	>0, see below
OUT_HI_LIM	If required, enter upper limit to be placed on OUT value	–
OUT_LO_LIM	If required, enter lower limit to be placed on OUT value	–

The GAIN and BIAS values to be entered depend upon the application selected. The table below offers suggestions for each of the functions shown in the table above

Function/Example	OUT		IN_1		IN_2		IN_3	
	GAIN	BIAS	GAIN	BIAS	GAIN	BIAS	GAIN	BIAS
$Q_b = Q_f \cdot K \cdot P/T$	K	0	1*	0	1*	(273)	N/A	N/A
$Q_b = Q_f \cdot K \cdot \sqrt{(P/T \cdot Z)}$	K	0	1*	0	1*	(273)	1	0
$Q_b = Q_f \cdot K_1 \cdot \sqrt{(K_2 + K_3T + K_4T^2)}$	K ₁	0	1*	0	1*	(273)	1	273
$Q_b = Q_f \cdot K_1 \cdot \sqrt{(K_2 + K_3P)}$	K ₁	0	1*	0	1*	0	N/A	N/A
$Q_{\text{heat}} = K \cdot Q_{\text{vol}} \cdot (T_2 - T_1)$	K	0	1	0	1	0	N/A	N/A
$Q_{\text{sp}} = Q_f \cdot \text{Ratio}$	1	0	1*	0	1*	0	N/A	N/A
$T_a = (T_1 + T_2 + T_3 + T_4)/4$	1	0	1	0	1	0	1	0
$P_{\text{diff}} = P_1 - P_2$	1	0	1	0	N/A	N/A	N/A	N/A
$\rho_T = A + K_1T + K_2T^2 + K_3T^3$	K ₁	A	K ₂	(273)	K ₃	(273)	–	–
$l_{bt} = l_{bm} \cdot (P_b - P_i)/(P_b - P_m)$	1	0	1	0	1	0	1	0

*Assuming that input is already in the required units, otherwise GAIN must be used as conversion factor

7.1.3 Block operation

Operating mode

The block normally operates in AUTO mode. The algorithm never changes the mode, even when inputs go bad. Difficulties with a function, such as division by zero and roots of negative numbers, are handled such that they do not disturb the status of **OUT** or the mode. Division by zero produces a large number of the proper sign. Infinity is not used, as it has a special meaning for unused limits. Roots of negative numbers produce the root of the absolute value, with a negative sign.

When **MODE_BLK.Target** is set to MAN, the block output can be manipulated by entering the desired status and value in the **OUT** parameter. An internal value is set to the difference between **OUT** and the output of the selected function, so that bumpless transfer is possible when the block returns to AUTO. The difference decays to zero over the time period entered in **BAL_TIME**.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Operation can be checked on-line by viewing a number of parameters:

Parameter	Meaning
IN	Value and status of primary input from upstream block
IN_LO	Value and status of low range input from upstream block
IN_1	Value and status of first auxiliary input from upstream block
IN_2	Value and status of second auxiliary input from upstream block
IN_3	Value and status of third auxiliary input from upstream block
PV	Value and status of process value after scaling
PRE_OUT	Value and status of selected algorithm
OUT	Value and status of block output

Status

The Arithmetic block supports standard status handling as described in Chapter 2.6 with the additions detailed in Chapter 7.1.1.

Output status	Meaning
Good NonCascade	Normal operation as configured
Uncertain	<ul style="list-style-type: none"> IN input status "Uncertain", IN_x status "Good" IN_x input status "Bad" or Uncertain, IN input status good
Bad NonCascade	<ul style="list-style-type: none"> IN input status "Bad", IN_x input status "Good", "Uncertain" or "Bad" Block configuration error

Block Errors

On execution of the block, the device checks for correct configuration. Block errors are indicated in the **BLOCK_ERR** parameter.

Condition	Reason
Block Configuration Error	<ul style="list-style-type: none"> Invalid parameter setting in ARITH_TYPE
Out-of-Service	<ul style="list-style-type: none"> MODE_BLK.Target currently set to OOS

7.1.4 Block parameters

Arithmetic block

Parameter	Valid range/ Options	Default value	Description
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set MODE_BLK.Target to AUTO, CAS or RCAS
BLOCK_ERR	RO		Block errors, see Chapter 6.1.9
PV	RO		Process value used in executing the block
OUT			Output value calculated as a result of executing the block (OUT_RANGE \pm 10%)
PRE_OUT	RO		Output value of calculation.
PV_SCALE		0-100%	PV scaling to required engineering units
OUT_RANGE		0-100%	OUT value scaling to required engineering units
GRANT_DENY		0	Access options, see Chapter 2.8.1
INPUT_OPTS		0	Input options for handling block input status.
IN			Primary input value of the block
IN_LO			Input for low range device in a range extension application.
IN_1			Auxiliary input 1
IN_2			Auxiliary input 2
IN_3			Auxiliary input 3
RANGE_HI		0	PV value at which transition from IN_LO to IN begins
RANGE_LO		0	PV value at which transition from IN_LO to IN ends
BIAS_IN_1		0	Bias to be applied to first auxiliary input
GAIN_IN_1		0	Gain to be applied to first auxiliary input
BIAS_IN_2		0	Bias to be applied to second auxiliary input, if used
GAIN_IN_2		0	Gain to be applied to second auxiliary input, if used
BIAS_IN_3		0	Bias to be applied to third auxiliary input, if used
GAIN_IN_3		0	Gain to be applied to third auxiliary input, if used
COMP_HI_LIM		0	High limit imposed on the PV compensation term.
COMP_LO_LIM		0	Low limit imposed on the PV compensation term.
ARITH_TYPE		0	Function to be executed by the block <ul style="list-style-type: none"> ■ 1= Flow comp. linear ■ 2= Flow comp. square root ■ 3= Flow comp. approx. ■ 4= BTU flow ■ 5= Traditional mult. div. ■ 6= Average ■ 7= Traditional summer ■ 8= Fourth order polynomial ■ 9= HTG comp. level
BAL_TIME	Positive	0	Decay time for OUT value for change from man to Auto mode
BIAS		0	Constant to be used in the calculation of OUT
GAIN		0	Scaling value to be used in the calculation of OUT
OUT_HI_LIM		100	Upper limit to be placed on OUT value
OUT_LO_LIM		0	Lower limit to be placed on OUT value
UPDATE_EVT			This alert is generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
Legend: RO = Read Only			

7.2 Output Splitter

The Output Splitter block allows two control outputs to be driven from a single input, whereby each output is a linear function of some portion of the input. The block is intended for use in split ranging or sequencing of multiple valve applications.

- Split range application:
 - Normally both valves are closed when the splitter input is 50%.
 - One valve opens fully as the input drops to 0%.
 - The other valve opens as the input rises above 50%.
- Sequencing application
 - Normally both valves are closed at 0% input.
 - One valve opens fully as the input rises to 50%, and the other stays shut.
 - The second valve opens as the input rises above 50%; the first valve may remain open or shut off quickly.

The block provides back calculation support by using the same linear function in reverse. Cascade initialization is supported by a decision table for combinations of input and output conditions. As this block is in the control path, it is able to pass limit and cascade initialization information back to the upstream block.

In ControlCare Application Designer the Output Splitter block (OS block) is assigned the generic name **Device Tag-OS-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 7-6.

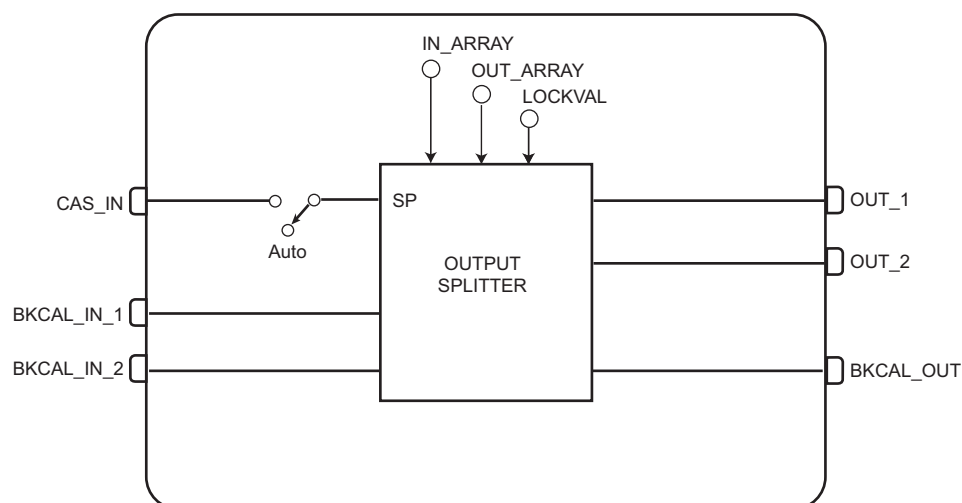


Fig. 7-6: Schematic diagram of Output Splitter block (OS Block)

7.2.1 Functional description

The **CAS_IN** input is connected to an upstream PID block which supplies the setpoint value **SP** in percentage of span to the block. The splitter outputs **OUT_1** and **OUT_2** are generated from **SP** by means of two linear functions.

Each function is defined by two end points, (X11, Y11) and (X12, Y12) for **OUT_1** and (X21, Y21) and (X22, Y22) for **OUT_2**. These parameters are configured by means of two arrays **IN_ARRAY** for the setpoint values (X) and **OUT_ARRAY** for the corresponding OUT values (Y).

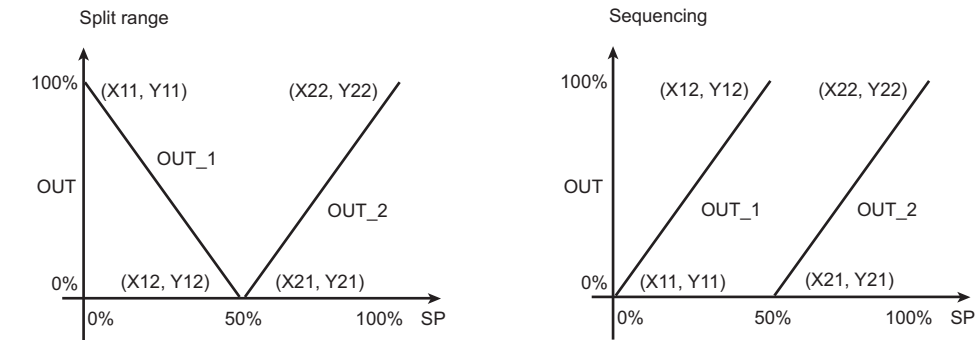


Fig. 7-7: Split range and sequencing operations

Fig. 7-7 shows the applications split range and sequencing. The corresponding array for these applications might be as follows:

	Split range			Sequencing	
	IN_ARRAY	OUT_ARRAY		IN_ARRAY	OUT_ARRAY
[1]	0	100	[1]	0	0
[2]	50	0	[2]	50	100
[3]	50	0	[3]	50	0
[4]	100	100	[4]	100	100

Other configurations are possible. If the array is configured such that a region of the input range is not specified, then the block will interpolate to the endpoint of the input value (0% or 100%).

In sequencing, the decision whether valve 1 remains open when the endpoint of **OUT_1** is reached is made in the parameter **LOCKVAL**.

- If **LOCKVAL** is true, **OUT_1** remains at its current value when **OUT_2** is non-zero
- If **LOCKVAL** is false, then as soon as the **OUT_2** becomes non-zero then **OUT_1** goes to zero

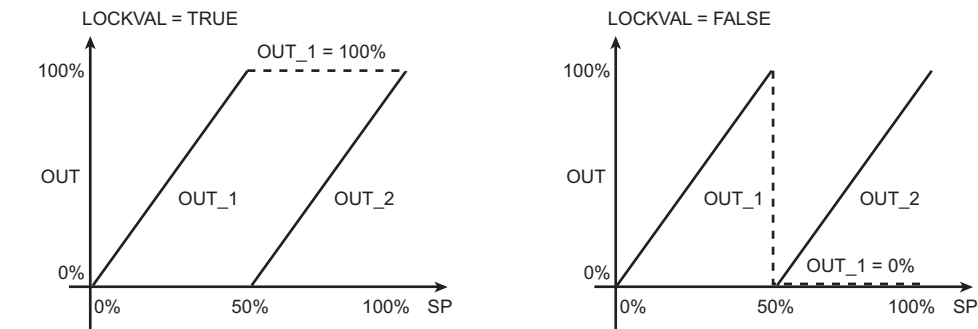


Fig. 7-8: Effect of **LOCKVAL** on **OUT_1**

The parameter **HYSTVAL** provides hysteresis in % of span for the switching point as follows:

Condition	Action
$X < X12 - \text{HYSTVAL}$	OUT_1 = calculated Y value
$X12 - \text{HYSTVAL} < X < X12$ and $X \neq X12$ since it was less than $X12 - \text{HYSTVAL}$	OUT_1 = calculated Y value
$X12 - \text{HYSTVAL} < X < X12$ and $X = X12$ since it was less than $X12 - \text{HYSTVAL}$	OUT_1 = as determined by LOCKVAL
$X12 < X$	OUT_1 = as determined by LOCKVAL

After splitting of the output, the values can be scaled in the **OUT_1_RANGE** and **OUT_2_RANGE** respectively before they are passed to the downstream application as **OUT_1** and **OUT_2**.

BKCAL_OUT normally provides the current SP to the upstream block. **BKCAL_IN_1** and **BACK_IN_2** provide the back calculation from the downstream blocks.

Status propagation

Substatus values received at **CAS_IN** are passed to both outputs when the cascade is closed. The back calculation status originates from an active output only. An output held by **LOCKVAL** is not considered to be active. If neither output is active, no limits are sent back on **BKCAL_OUT**. An IFS is passed on to both the active and the inactive outputs.

Since the splitter has two independent outputs, cascade initialization is handled slightly differently than is described in Chapter 2.6.4. When a downstream block sends an initialization request to the splitter, one of two things can happen:

- The request can be passed on by the splitter to its upstream block, so that all three blocks balance for bumpless transfer to cascade mode.
- The request cannot be passed on and splitter output moves to the requested value by calculating the offset between the requested and actual value. The offset is ramped to zero in **BAL_TIME** seconds immediately after the cascade closed.

Table 7-2 indicates the possible back calculation statuses together with the actions invoked by the Output Splitter block..

BKCAL_IN_1	BKCAL_IN_2	BKCAL_OUT	Action
NotInvited	NotInvited	NotInvited	Not specified
NotInvited	OK	OK	BKCAL_OUT limited to X21 low and X22 high
OK	NotInvited	OK	BKCAL_OUT limited to X11 low and X12 high
Initial.Request	NotInvited	Initial.Request	Initialize cascade to value given by curve X1 vs Y1
Initial.Request	OK	OK	Initialize OUT_1 using internal offset from Y1
NotInvited	Initial.Request	Initial.Request	Initialize cascade to value given by curve X2 vs Y2
OK	Initial.Request	OK	Initialize OUT_2 using internal offset from Y2
OK = cascade is closed with status other than NI or IR			

Tab. 7-2: Splitter block actions as a function of back calculation status of the connected blocks

Cascade initialization is also required when the block moves from Auto to Cas mode. This action is identical to that described in Chapter 2.6.4.

The option "IFS if Bad CAS_IN" in **STATUS_OPS** causes the fault status to be propagated to the downstream blocks without causing the block mode to change.

7.2.2 Block configuration

Control strategy

Fig. 7-9 shows the control strategy together with the links that must be made between the blocks. In this example the splitter receives its input from the **OUT** parameter of an upstream PID Control block. The split output is then passed on to a downstream Analog Output block and a second Control PID block, however, the two outputs can be connected to any combination of blocks that operate in cascade mode.

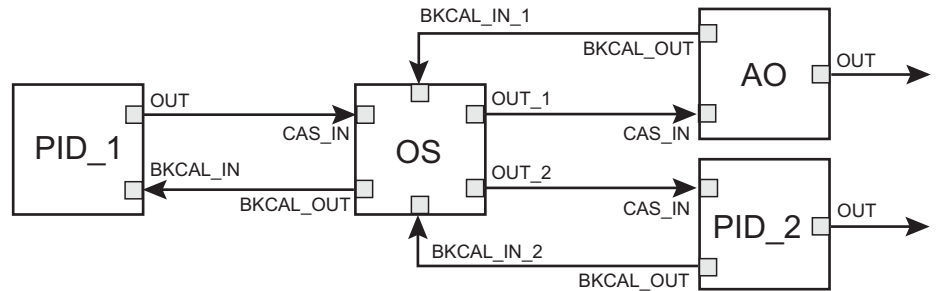


Fig. 7-9: Links to be made for Output Splitter block

Block configuration

The block is configured as follows:

Parameter	Action	Option/Value
MODE_BLK.Target	Set the target mode of the block to Cas	Cas
OUT_1_RANGE.EU_100	Enter upper range limit for OUT_1 output variable	default 100
OUT_1_RANGE.EU_0	Enter lower range limit for OUT_1 output variable	default 0
OUT_1_RANGE.UNITS_INDEX	Enter unit of OUT_1 output variable	default %
OUT_2_RANGE.EU_100	Enter upper range limit for OUT_2 output variable	default 100
OUT_2_RANGE.EU_0	Enter lower range limit for OUT_2 output variable	default 0
OUT_2_RANGE.UNITS_INDEX	Enter unit of OUT_2 output variable	default %
STATUS_OPTS	If required, select the status option by ticking the box ■ If selected, the block continues to operate with bad input and the downstream blocks handle the fault	IFS if Bad CAS_IN
IN_ARRAY.[1] IN_ARRAY.[2] IN_ARRAY.[3] IN_ARRAY.[4]	Enter setpoint values in upstream block units: ■ Corresponding to lower range value of OUT_1 ■ Corresponding to upper range value of OUT_1 ■ Corresponding to lower range value of OUT_2 ■ Corresponding to upper range value of OUT_2	e.g. 0 e.g. 50 e.g. 50 e.g.100
OUT_ARRAY.[1] OUT_ARRAY.[2] OUT_ARRAY.[3] OUT_ARRAY.[4]	Enter setpoint values in OUT_RANGE units: ■ Lower range value of OUT_1 ■ Upper range value of OUT_1 ■ Lower range value of OUT_2 ■ Upper range value of OUT_2	e.g. 0 e.g. 100 e.g. 0 e.g.100
LOCKVAL	For sequencing operation, select OUT_1 position	e.g. Lock (= 100%)
BAL_TIME	Enter the time in seconds for OUT to adjust to a new value if the block mode changes from Cas	e.g. 10 s
HYSTVAL	If required, enter a switching point hysteresis for OUT_1 in percentage of output span	e.g. 1

7.2.3 Block operation

Operating mode

The block normally operates in CAS mode and takes its setpoint SP from the **CAS_IN** parameter. If **BKCAL_IN_1** or **BKCAL_IN_2** indicate that the cascade is broken, see Chapter 7.1.2, Status propagation, the block will be forced to IMan.

When **MODE_BLK.Target** is set to AUTO, the block output can be manipulated on-line by entering the desired status and value in the **SP** parameter. An internal value is set to the difference between **OUT** and the output of the selected function, so that bumpless transfer is possible when the block is returns to CAS. The difference decays to zero over the time period entered in **BAL_TIME**.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Operation can be checked on-line by viewing a number of parameters:

Parameter	Meaning
CAS_IN	Value and status of setpoint value from upstream block
OUT_1	Value and status of output 1
OUT_2	Value and status of output 2

Status

The Output Splitter block supports standard status handling as described in Chapter 2.6 with the additions detailed in Chapter 7.2.1.

Output status	Meaning
Good NonCascade	Normal operation as configured - not operating in control loop
Good Cascade	Normal operation as configured - operating in control loop"
Uncertain	Input "Uncertain" - not operating in control loop
Bad NonCascade	<ul style="list-style-type: none"> ■ Input "Bad" - not operating in control loop ■ Block configuration error
Bad Cascade	<ul style="list-style-type: none"> ■ Input "Bad" - operating in control loop ■ Block configuration error

Block Errors

On execution of the block, the device checks for correct configuration. Block errors are indicated in the **BLOCK_ERR** parameter.

Condition	Reason
Block Configuration Error	<ul style="list-style-type: none"> ■ IN_ARRAY configuration error <ul style="list-style-type: none"> - Value [1] ≤ Value [2], - Value [3] ≤ Value [4], - Value [1] > Value [4]
Out-of-Service	<ul style="list-style-type: none"> ■ MODE_BLK.Target currently set to OOS

7.2.4 Block parameters

Output Splitter block

Parameter	Valid range/ Options	Default value	Description
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set MODE_BLK.Target to AUTO, CAS or RCAS
BLOCK_ERR	RO		Block errors, see Chapter 6.1.9
SP			Setpoint used by the block in AUTO mode (PV_SCALE \pm 10%)
OUT_1			1 st output value calculated as a result of executing the block
OUT_2			2 nd output value calculated as a result of executing the block
OUT_1_RANGE		0-100%	OUT_1 value scaling to required engineering units
OUT_2_RANGE		0-100%	OUT_2 value scaling to required engineering units
GRANT_DENY		0	Access options, see Chapter 2.8.1
STATUS_OPTS		0	Status options, see Chapter 2.8.4 and Chapter 6.1.9
CAS_IN			Setpoint used by the block in CAS mode
BKCAL_OUT			Feedback for BKCAL_IN of an upstream block.
IN_ARRAY		0, 0, 0, 0	2x 2 SP range-end values for OUT_1 and OUT_2
OUT_ARRAY		0, 0, 0, 0	2x 2 output range-end values for OUT_1 and OUT_2
LOCKVAL		No Lock	Boolean value for controlling OUT_1 value when OUT_2 >0 ■ Lock = OUT_1 = 100% ■ No Lock = OUT_1 = 0%
BKCAL_IN_1			Feedback from BKCAL_OUT of downstream block connected to OUT_1
BKCAL_IN_2			Feedback from BKCAL_OUT of downstream block connected to OUT_2
BAL_TIME	Positive	0	Decay time for OUT value for change from Auto to Cas mode
HYSTVAL		0	Switch point hysteresis for OUT_1
UPDATE_EVT			This alert is generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
Legend: RO = Read Only			

7.3 Signal Characterizer

The Signal Characterizer block is used to enter of a strapping table, comprising up to 21 pivot points, that allows the mapping of a non-linear relationship between input and output values. The block has two paths, which use a single look-up table. The second path allows an optional swapping of the x and y axes, allowing its use in a backward control path.

The status of an input is copied to the corresponding output, so that the block can be used in the control or process signal path.

In ControlCare Application Designer the Signal Characterizer block (SC block) is assigned the generic name **Device Tag-CHAR-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 7-10.

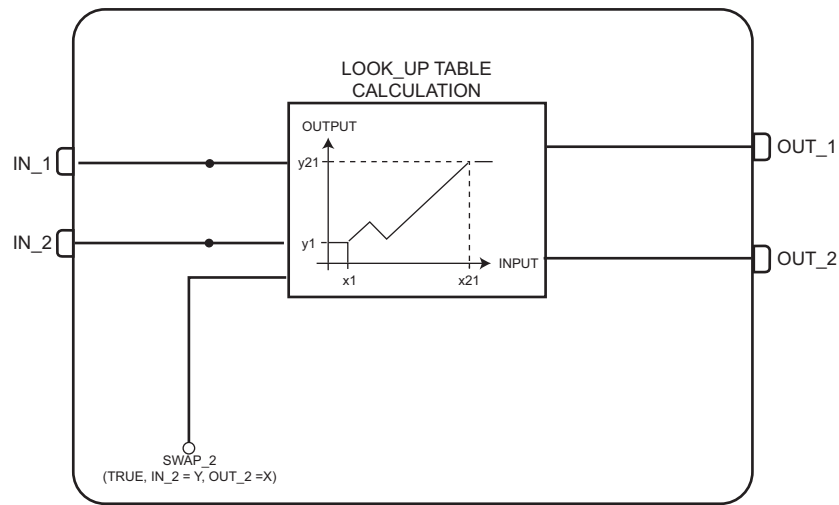


Fig. 7-10: Schematic diagram of Signal Characterizer block (SC Block)

7.3.1 Functional Description

Standard operation

In standard operating mode, **IN_1** and **IN_2** comprise a value and status from an upstream function block. The block calculates **OUT_1** from **IN_1** and **OUT_2** from **IN_2** using the same look-up table. The curve used is defined by up to 21 points:

$[x_1; y_1], [x_2; y_2] \dots [x_{21}; y_{21}]$ in the parameters **CURVE_X; CURVE_Y**

where x corresponds to the input and y to the output. The x-coordinates are entered in the units of IN and the y-coordinates in the units of OUT. The units and range limits can be made visible to an external application by using the parameters **X_RANGE** (= IN) and **Y_RANGE** (= OUT).

The output value is calculated by linear interpolation between the two table values bracketing the input value. For this reason, the values of x must rise monotonically. If this is not the case, a configuration error is set in **BLOCK_ERR** and the block switches to OOS mode.

If the look-up table has m points, where m is less than 21, all non-configured points $[x_{m+1}; y_{m+1}]$, $[x_{m+2}; y_{m+2}]$ etc. will be set to +INFINITY to mark them as unused. Since x_1 is the smallest value for the input and x_m the largest value, the output is set to y_1 when the input value is smaller than x_1 , and to y_m when the input value is larger than x_m . When the input value is out of the limits, set by x_1 and x_m , the OUT status indicates the condition.

Use in backward control path

If the parameter **SWAP_2** is set to "Swap", then the x and y axes of the look-up table are swapped on the **IN_2** to **OUT_2** path. This means that **IN_2** must be connected to a downstream block and **OUT_2** to an upstream block, building a backward control path. In this case **IN_2** must be in the units of **OUT_1 (Y_RANGE)** and **OUT_2** in the units of **IN_1 (X_RANGE)**.

In this case, the y values must also be monotonic, i.e. the y values must always increase or always decrease when the x values increase. If this is not the case, a configuration error is set in **BLOCK_ERR** and the block switches to OOS mode.

Status propagation

The output values **OUT_1** and **OUT_2** replicate the status of the corresponding **IN** values or are generated by the block. If the **BLOCK_ERR** parameter indicates a block error, the status of the output is set to "Bad"

The sub-status is also passed to the outputs. If one of the **CURVE_X/CURVE_Y** limits is reached or the input is limited, "High Limited" or "Low Limited" will be indicated in the output limit status. The limits are reversed if the curve slope is negative.

When the block is operating in a control loop, i.e. **SWAP_2** = "Swap", cascade initialization is controlled by the lower block, see Chapter 2.6. When the block is out of service, the cascade to both the lower and upper blocks is broken by setting "Bad" at the outputs. When the block moves back to Auto mode, the lower block begins cascade initialization with status values that pass along the backward control path (**IN_2** => **OUT_2**) to the upper block. The answering status signals from the upper block pass through this block to the lower block via **IN_1** => **OUT_1**.

7.3.2 Block configuration**Control strategy**

Fig. 7-11 shows the control strategy together with the links that must be made between the blocks for the standard mode of operation. In this example, the characterizer receives its inputs from the **OUT** parameters of two upstream Analog Input blocks. The application requires that both signals are characterized in exactly the same way. The output of the characterizer is then passed on to two independent Control PID blocks.

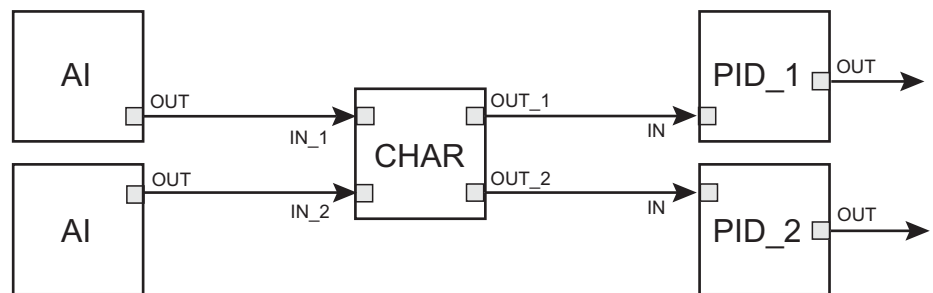


Fig. 7-11: Links to be made for Signal Characterizer block in standard operation

Fig. 7-12 shows the characterizer operating in a backward control path, i.e. **SWAP_2** = Swap. In this case it is used to characterize the OUT value of a Control PID block before this is used by the downstream analog output block. In order that the back calculation can be used by the upstream PID block it must be reconverted by the characterizer, by connecting **BKCAL_OUT** of the output block to **IN_2** and **OUT_2** to **BKCAL_IN** of the Control PID block.

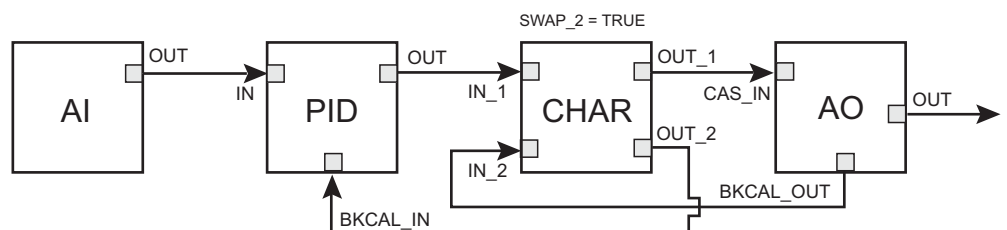


Fig. 7-12: Links to be made for Signal Characterizer block in a backward control path

Block configuration

The block is configured as follows:

Parameter	Action	Option/Value
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
X_RANGE.EU_100 X_RANGE.EU_0 X_RANGE.UNITS_INDEX	Enter upper range limit for the input variables Enter lower range limit for input variables Enter unit of the input variables	e.g. 1000 e.g. 0 e.g. kg/hr
Y_RANGE.EU_100 Y_RANGE.EU_0 Y_RANGE.UNITS_INDEX	Enter upper range limit for the output variables Enter lower range limit for output variables Enter unit of the output variables	e.g. 100 e.g. 0 e.g. %
SWAP_2	Set to "Swap" if characterizer is to be used in a backward control path.	e.g. No Swap (default)
CURVE_X.[1] ... CURVE_X.[21]	Enter X values (input variable) of strapping table <ul style="list-style-type: none"> Values must rise or fall monotonically Values must be within the range defined by X_RANGE Unedited values are automatically set to infinity, which means they are not used 	0 (kg/hr) ... 1000 (kg/hr)
CURVE_Y.[1] ... CURVE_Y.[21]	Enter Y values (output variable) of strapping table^ <ul style="list-style-type: none"> Exactly the same number of values must be entered as for the X_CURVE. Values must be within the range defined by Y_RANGE If SWAP_2 = TRUE, values must rise or fall monotonically Unedited values are automatically set to infinity, which means they are not used 	0 (%) ... 100 (%)

7.3.3 Block operation**Operating mode**

The block normally operates in Auto mode and takes its input from the **IN_1** and **IN_2**. If there is a block configuration error, the block switches to OOS (Out of Service).

When **MODE_BLK.Target** is set to Man, the block output can be manipulated on-line by entering the desired status and value in the **OUT_1** and/or **OUT_2** parameter.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Operation can be checked on-line by viewing a number of parameters:

IN_1	Input value and status used by the characterizer for path 1
IN_2	Input value and status used by the characterizer for path 2.
OUT_1	Block output value and status for path 1
OUT_2	Block output value and status for path 2

Status

The Signal Characterizer block supports the qualities Bad, Uncertain and Good NonCascade together with the associated sub- and limit statuses, see Chapter 2.3.2. When SWAP_2 = "Swap", the qualities Good Cascade and Bad Cascade are also supported. "

Status	Meaning
Good NonCascade	Block operating in standard mode; input and block execution "Good"
Good Cascade	Block operating in control loop (SWAP_2 = "Swap"); input and block execution "Good"
Uncertain	Block operating in standard mode; input "Uncertain"
Bad NonCascade	Block operating in standard mode; input "Bad" or block configuration error
Bad Cascade	Block operating in control loop (SWAP_2 = "Swap"); input "Bad" or block configuration error

Block Errors

On execution of the block, the device checks for correct configuration. Block errors are indicated in the **BLOCK_ERR** parameter.

Condition	Reason
Block Configuration Error	<ul style="list-style-type: none"> Values in CURVE_X are not monotonic SWAP_2 is true and values in CURVE_Y are not monotonic
Input Failure/ process variable has BAD status	<ul style="list-style-type: none"> Upstream transducer block has status "Uncertain" or "Bad" Upstream Analog Output block is out of service
Out-of-Service	<ul style="list-style-type: none"> MODE_BLK.Target currently set to OOS

7.3.4 Block parameters**Signal Characterizer block**

Parameter	Valid range/ Options	Default value	Description
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set MODE_BLK.Target to AUTO, CAS or RCAS
BLOCK_ERR	RO		Block errors, see Chapter 6.1.9
OUT_1			Block output value and status for path 1
OUT_2			Block output value and status for path 2
X_RANGE		0-100%	IN value scaling to required engineering units, for HMI
Y_RANGE		0-100%	OUT value scaling to required engineering units, for HMI
GRANT_DENY		0	Access options, see Chapter 2.8.1
IN_1		0	Input value and status used by the characterizer for path 1
IN_2			Input value and status used by the characterizer for path 2.
SWAP_2		No Swap	Toggle to invert characteristic curve in path 2 when the block is linked to the back control path
CURVE_X		INF	X values (input variable) of characteristic in units of X_RANGE
CURVE_Y		INF	Y values (output variable) of characteristic in units of Y_RANGE
UPDATE_EVT			This alert is generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
Legend: RO = Read Only			

7.4 Cascade Signal Characterizer

The Cascade Signal Characterizer block differs from the signal characterizer in that:

- up to 51 pivot points can be entered for the strapping table
- several blocks can be used in cascade to map strapping tables with more than 51 values

Like the Signal Characterizer, the block has two paths, which use a single look-up table. The second path allows an optional swapping of the x and y axes, allowing its use in a backward control path.

The status of an input is copied to the corresponding output, so that the block can be used in the control or process signal path.

In ControlCare Application Designer the Signal Characterizer block (SC block) is assigned the generic name **Device Tag-CCHAR-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 7-13.

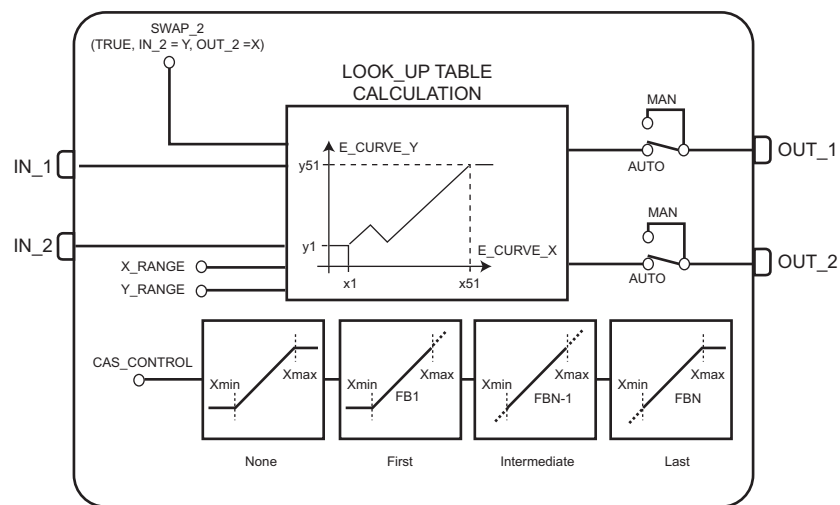


Fig. 7-13: Schematic diagram of Cascade Signal Characterizer block

7.4.1 Functional Description

Standard operation

The Cascade Signal Characterizer block is identical in parameters and function to the Signal Characterizer, see Chapter 7.3.1, with two exceptions:

- The strapping table of up to 51 pivot points is entered in the parameters **E_CURVE_X** and **E_CURVE_Y**
- The cascading of the blocks is controlled by the parameter **CAS_CONTROL**

CAS_CONTROL determines whether the block is cascaded and what position it has in the chain. It can be set to one of four values

- None: the block is used on its own
- First: the block is the first block in the cascade
- Intermediate: the block is at a position between the ends of the cascade
- Last: the block is the last block in the cascade

When cascaded, the units of **E_CURVE_X** and **E_CURVE_Y** must be consistent throughout the cascade. The controller sees the complete characteristic curve, and generates an **OUT_1** and/or **OUT_2** value at the final block according to the corresponding value of **IN_1** and/or **IN_2** at the first block. The x-coordinates are entered in the units of IN and the y-coordinates in the units of OUT. The units and range limits can be made visible to an external application by using the parameters **X_RANGE** (= IN) and **Y_RANGE** (= OUT), whereby these must be set individually for each block.

Use in backward control path

If the parameter **SWAP_2** is set to "Swap", then the x and y axes of the look-up table are swapped on the **IN_2** to **OUT_2** path. This means that **IN_2** must be connected to a downstream block and **OUT_2** to an upstream block, building a backward control path. In this case **IN_2** must be in the units of **OUT_1** (**Y_RANGE**) and **OUT_2** in the units of **IN_1** (**X_RANGE**).

In this case, the y values must also be monotonic, i.e. the y values must always increase or always decrease when the x values increase. When blocks are cascaded, this applies to all values in the cascade. If this is not the case, a configuration error is set in **BLOCK_ERR** and the block switches to OOS mode.

Status propagation

The output values **OUT_1** and **OUT_2** replicate the status of the corresponding **IN** values or are generated by the block. If the **BLOCK_ERR** indicates a block error, the status of the output is set to "Bad"

The sub-status is also passed to the outputs. If one of the **E_CURVE_X/E_CURVE_Y** limits is reached or the input is limited, "High Limited" or "Low Limited" will be indicated in the output limit status. The limits are reversed if the curve slope is negative.

When the block is operating in a control loop, i.e. **SWAP_2** = "Swap", cascade initialization is controlled by the lower block, see Chapter 2.6. When the block is out of service, the cascade to both the lower and upper blocks is broken by setting "Bad" at the outputs. When the block moves back to Auto mode, the lower block begins cascade initialization with status values that pass along the backward control path (**IN_2** => **OUT_2**) to the upper block. The answering status signals from the upper block pass through this block to the lower block via **IN_1** => **OUT_1**.

7.4.2 Block configuration

Control strategy

Fig. 7-14 shows the control strategy together with the links that must be made between the blocks for the standard mode of operation. In this example the cascade signal characterizer receives its inputs from the **OUT** parameters of upstream Analog Input blocks. The application requires that both signals are characterized in exactly the same way. The output of the characterizer is then passed on to two independent Control PID blocks.

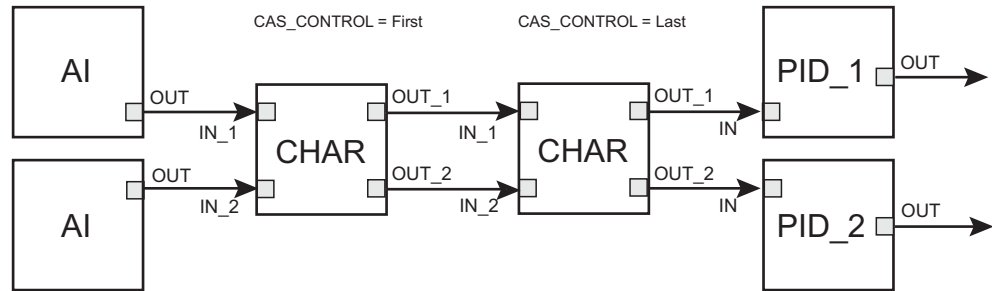


Fig. 7-14: Links to be made for Cascade Signal Characterizer block in standard operation (2 blocks cascaded)

Fig. 7-15 shows the cascade characterizer operating in a backward control path, i.e. **SWAP_2 = Swap**. In this case it is used to characterize the **OUT** value of a Control PID block before this is used by the downstream analog output block. In order that the back calculation can be used by the upstream PID block it must be reconverted by the characterizer, by connecting **BKCAL_OUT** of the output block to **IN_2** of the last characterizer block, **OUT_2** of the last characterizer block to **IN_2** of the first characterizer block and **OUT_2** of the first characterizer block to **BKCAL_IN** of the Control PID block.

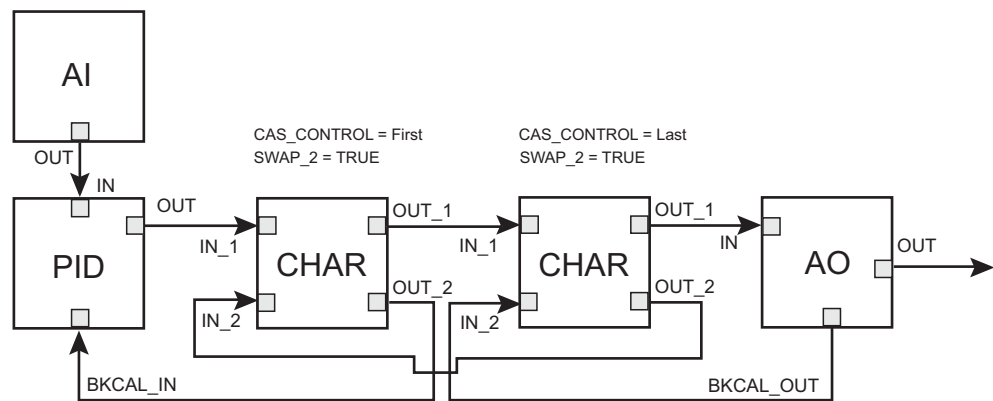


Fig. 7-15: Links to be made for Cascade Signal Characterizer block in a backward control path (2 blocks cascaded)

Block configuration

The blocks are configured as follows:

Parameter	Action	Option/Value
CCHAR Block 1		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
X_RANGE.EU_100 X_RANGE.EU_0 X_RANGE.UNITS_INDEX	Enter upper range limit for the input variables Enter lower range limit for input variables Enter unit of the input variables	e.g. 500 e.g. 0 e.g. kg/hr
Y_RANGE.EU_100 Y_RANGE.EU_0 Y_RANGE.UNITS_INDEX	Enter upper range limit for the output variables Enter lower range limit for output variables Enter unit of the output variables	e.g. 50 e.g. 0 e.g. %
SWAP_2	Set to "Swap" if characterizer is to be used in a backward control path.	e.g. No Swap (default)
E_CURVE_X.[1] ... E_CURVE_X.[51]	Enter X values (input variable) of strapping table <ul style="list-style-type: none"> Values must rise or fall monotonically Values must be within the range defined by X_RANGE For use in cascade, all 51 values must be entered If not used in cascade, unedited values are automatically set to infinity, which means they are not used 	0 (kg/hr) ... 500 (kg/hr)
E_CURVE_Y.[1] ... E_CURVE_Y.[51]	Enter Y values (output variable) of strapping table^ <ul style="list-style-type: none"> Exactly the same number of values must be entered as for the X_CURVE. Values must be within the range defined by Y_RANGE If SWAP_2 = TRUE, values must rise or fall monotonically 	0 (%) ... 50 (%)
CAS_CONTROL	Set to "First" if to be cascaded, otherwise "None"	First
CCHAR Block 2... (when used in cascade)		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
X_RANGE.EU_100 X_RANGE.EU_0 X_RANGE.UNITS_INDEX	Enter upper range limit for the input variables Enter lower range limit for input variables Enter unit of the input variables	e.g. 1000 e.g. 510 e.g. kg/hr
Y_RANGE.EU_100 Y_RANGE.EU_0 Y_RANGE.UNITS_INDEX	Enter upper range limit for the output variables Enter lower range limit for output variables Enter unit of the output variables	e.g. 100 e.g. 51 e.g. %
SWAP_2	Set to "Swap" if characterizer is to be used in a backward control path.	e.g. No Swap (default)
E_CURVE_X.[1] ... E_CURVE_X.[51]	Enter X values (input variable) of strapping table <ul style="list-style-type: none"> Values must rise or fall monotonically Values must be within the range defined by X_RANGE For use in intermediate cascade, all 51 values must be entered If last in cascade, unedited values are automatically set to infinity, which means they are not used 	510 (kg/hr) ... 1000 (kg/hr)
E_CURVE_Y.[1] ... E_CURVE_Y.[51]	Enter Y values (output variable) of strapping table^ <ul style="list-style-type: none"> Exactly the same number of values must be entered as for the X_CURVE. Values must be within the range defined by Y_RANGE If SWAP_2 = TRUE, values must rise or fall monotonically Unedited values are automatically set to infinity, which means they are not used 	51 (%) ... 100 (%)
CAS_CONTROL	Set to "Last" if last in cascade, otherwise "Intermediate"	Last

7.4.3 Block operation

Operating mode

The block normally operates in Auto mode and takes its input from the **IN_1** and **IN_2**. If there is a block configuration error, the block switches to OOS (Out of Service).

When **MODE_BLK.Target** is set to Man, the block output can be manipulated on-line by entering the desired status and value in the **OUT_1** and/or **OUT_2** parameter.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Operation can be checked on-line by viewing a number of parameters:

IN_1	Input value and status used by the characterizer for path 1
IN_2	Input value and status used by the characterizer for path 2.
OUT_1	Block output value and status for path 1
OUT_2	Block output value and status for path 2

Status

The Signal Characterizer block supports the qualities Bad, Uncertain and Good NonCascade together with the associated sub- and limit statuses, see Chapter 2.3.2. When **SWAP_2** = "Swap", the qualities Good Cascade and Bad Cascade are also supported.

Status	Meaning
Good NonCascade	Block operating in standard mode; input and block execution "Good"
Good Cascade	Block operating in control loop (SWAP_2 = "Swap"); input and block execution "Good"
Uncertain	Block operating in standard mode; input "Uncertain"
Bad NonCascade	Block operating in standard mode; input "Bad" or block configuration error
Bad Cascade	Block operating in control loop (SWAP_2 = "Swap"); input "Bad" or block configuration error

7.4.4 Block Errors

On execution of the block, the device checks for correct configuration. Block errors are indicated in the **BLOCK_ERR** parameter.

Condition	Reason
Block Configuration Error	<ul style="list-style-type: none"> ■ Values in CURVE_X are not monotonic (throughout cascade) ■ SWAP_2 is true and values in CURVE_Y are not monotonic (throughout cascade) ■ Less than 51 values in E_CURVE_X/E_CURVE_Y when CAS_CONTROL options "First" or "Intermediate" selected
Input Failure/ process variable has BAD status	<ul style="list-style-type: none"> ■ Upstream transducer block has status "Uncertain" or "Bad" ■ Upstream Analog Output block is out of service
Out-of-Service	<ul style="list-style-type: none"> ■ MODE_BLK.Target currently set to OOS

7.4.5 Block parameters

Cascade Signal Characterizer block

Parameter	Valid range/ Options	Default value	Description
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set MODE_BLK.Target to AUTO, CAS or RCAS
BLOCK_ERR	RO		Block errors, see Chapter 6.1.9
OUT_1			Block output value and status for path 1
OUT_2			Block output value and status for path 2
X_RANGE		0-100%	IN value scaling to required engineering units, for HMI
Y_RANGE		0-100%	OUT value scaling to required engineering units, for HMI
GRANT_DENY		0	Access options, see Chapter 2.8.1
IN_1		0	Input value and status used by the characterizer for path 1
IN_2			Input value and status used by the characterizer for path 2.
SWAP_2		No Swap	Toggle to invert characteristic curve in path 2 when the block is linked to the back control path
E_CURVE_X		INF	X values (input variable) of characteristic in units of X_RANGE
E_CURVE_Y		INF	Y values (output variable) of characteristic in units of Y_RANGE
UPDATE_EVT			This alert is generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
CAS_CONTROL		None	Determines whether the block is cascaded with others and what position it has in the chain <ul style="list-style-type: none"> ■ None: the block is used on its own ■ First: the block is the first block in the cascade ■ Intermediate: the block is at a position between the ends of the cascade ■ Last: the block is the last block in the cascade
Legend: RO = Read Only			

7.5 Integrator

The Integrator function block is normally used to totalise flow, giving total mass or volume over a given time, or to totalise power, providing the total energy. It:

- integrates an analog rate value over time or
- accumulates pulses originating from pulse input blocks or other integrator blocks

The block may be used as a totalizer that counts up until reset or as a batch totalizer that has a setpoint, where the integrated or accumulated value is compared to pre-trip and trip settings, generating discrete signals when these settings are reached.

The integrated value may counted up, starting from zero, or down, starting from the trip value. The block has two flow inputs so that it can calculate and integrate net flow. This can be used to calculate volume or mass variation in vessels or as an optimizing tool for flow ratio control.

In order to determine the amount of uncertain or bad readings, the block integrates the variables with bad or bad and uncertain status separately. The values used in this second integration are the values with good status just before they went from good to bad or uncertain. The ratio of good to total counts determines the output status. Absolute values are used to avoid problems with changing signs.

In ControlCare Application Designer the Integrator block (IT block) is assigned the generic name **Device Tag-INTG-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 7-16.

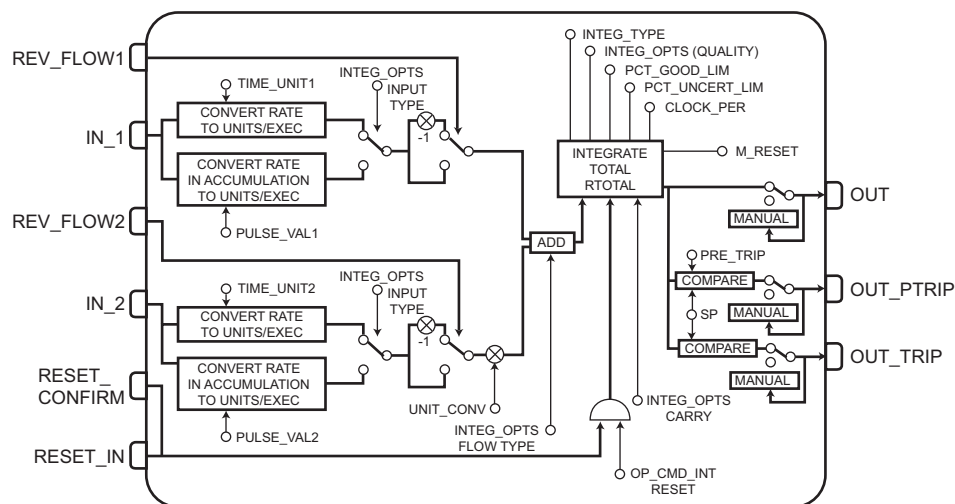


Fig. 7-16: Schematic diagram of Integrator block

7.5.1 Functional description

The integrator block has two dual purpose inputs **IN_1** and **IN_2** that may receive a measurement per unit time (rate) or an accumulated number of pulses (accumulation). If **IN_2** is not connected, no calculations are made along this path.

By default, the input type is rate. It can be changed to accumulation by selecting the integration options "Input 1 accumulate" and if used "Input 2 accumulate" in **INTEG_OPTS**.

Rate	Used when the variable connected to the input is e.g. kg/s, gal/h etc. The input may originate from the OUT value of a pulse input block or from the OUT value of an analog input block.
Accum	Used when the input originates from the OUT_ACCUM output of a pulse input block, which represents the accumulated number of pulse counts from a transducer, or from the output of another integrator block.

Input type: Rate (default)

When the inputs are connected to rate outputs, the parameters **TIME_UNIT1** and if required **TIME_UNIT2** are used to define the rate time unit. This allows the rate to be converted to units of mass, volume or energy per second. When the second input is in use, it may be necessary to convert it into the same units as the first. This is done by entering the conversion factor in **UNIT_CONV**.

When multiplied by the block execution time each rate gives the mass, volume or energy increment per block execution, see Fig. 7-17.

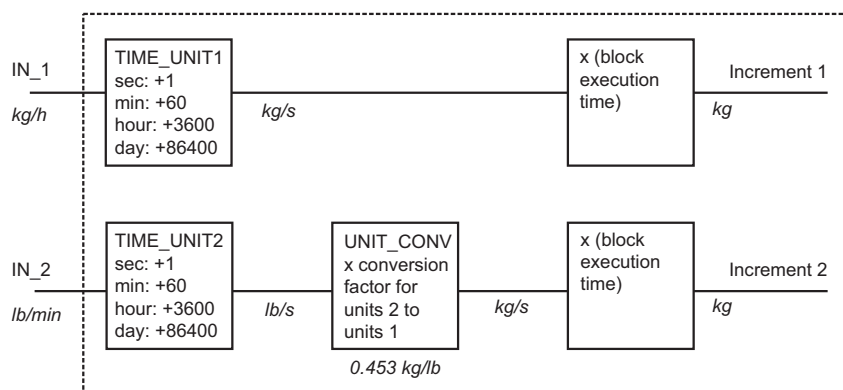


Fig. 7-17: Increment calculation with rate input

Input type: Accum

When operating in "Accum" mode the input may be connected to the **OUT_ACCUM** parameter of an pulse input block or a the **OUT** parameter of another Integrator block. The connected block must count to 999,999 and have a maximum increment/decrement of 499,999 per cycle. The integrator block then determines the number of additional counts since the last execution of the block.

The variation of each input is then multiplied by the value, in engineering units, of each pulse as given by the parameters **PULS_VAL1** and **PULSE_VAL2** respectively. If necessary, a unit conversion factor can be entered in **UNIT_CONV**. The result is the increment in engineering units of e.g. mass, volume or energy per block execution, see Fig. 7-18.

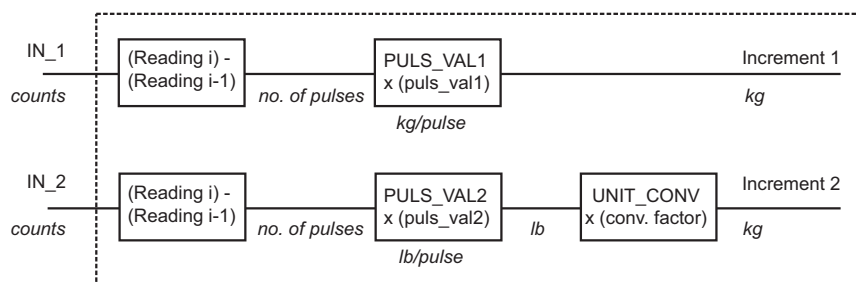


Fig. 7-18: Increment calculation with accumulated input

Net flow

The integrator considers a positive sign to be an indication of forward flow and negative sign to be an indication of reverse flow. Some flowmeters already indicate the forward and reverse flows by adding a sign to their output, others use a separate binary output. A binary signal can be connected to the inputs **REV_FLOW1** and **REV_FLOW2**, whereby a true status indicates a reverse flow condition.

The net flow over two inputs is obtained by adding the two increments. The net increment has a positive or negative sign to indicate the net flow direction. If the flow into and out of a tank is being monitored, for example, the inflow should be assigned a positive sign and the outflow a negative sign. This can be done, e.g. by connecting the outflow to **IN_2** and assigning a negative value to the unit conversion parameter.

The net flow direction to be considered in the totalization is set with the the **INTEG_OPTS** parameters "Flow forward" and "Flow reverse". The following totalizing options are available and apply after application of **REV_FLOW1** or **REV_FLOW2**.

Flow forward	Positive flows are totalized. The negative values are treated as zero. ■ Select the option "Flow forward" in INTEG_OPTS only
Flow reverse	Only negative flows are totalized. The positive values are treated as zero. ■ Select the option "Flow reverse" in INTEG_OPTS only
Any	Both positive and negative values are totalized. ■ Select the both options "Flow forward" and "Flow reverse" in INTEG_OPTS

Integration of inputs

Three internal registers are used for the totalization.

Total	The net increment is added every cycle, dependent on status. ■ Total provides the value that can be read in the output OUT
Atotal	The absolute value of the net increment is added every cycle, regardless of status. ■ This value is used in the calculation of the percentage of good counts PCT_INCL.
Rtotal	The absolute value of the net increments with bad status (rejects) are added to this register. ■ RTotal provides the value that can be read in the output RTOTAL and is used in the calculation of the percentage of good counts PCT_INCL

A further value, **OUT_RANGE** is used only for display of the totals by a host. The high and low range values of OUT_RANGE have no effect on the block.

Types of integration

The integration can start from zero and increase or it can start from a setpoint value (TOTAL_SP) and decrease. Reset may be automatic, periodic, or on demand. This is defined by the enumerated parameter INTEG_TYPE:

UP_AUTO	Counts up with automatic reset when TOTAL_SP is reached. – Uses the TOTAL_SP, OUT_TRIP and OUT_PTRIP parameters
UP_DEM	Counts up with demand reset.
DN_AUTO	Counts down with automatic reset when zero is reached – Uses the TOTAL_SP, OUT_TRIP and OUT_PTRIP parameters
DN_DEM	Counts down with demand reset
PERIODIC	Counts up and is reset periodically according to CLOCK_PER. – Disables the operator reset OP_CMD_INT
DEMAND	Counts up and is reset on demand
PER&DEM	Counts up and is reset periodically or on demand

Resetting the totals

The block uses a discrete input **RESET_IN** to reset the internal integration registers. The operator can also send an operator command to reset the same registers by making **OP_CMD_INT** = RESET. This is a momentary switch, which is turned off when the block is evaluated. Either method causes a reset of the registers.

A reset occurs at the end of the block execution after the totals have been adjusted. The block takes a snapshot of Total, Rtotal and TOTAL_SP prior to the reset and moves the values to the registers **STOTAL**, **SRTOTAL** and **SSP**, respectively. This information is kept until the next reset.

The integrator rejects renewed reset requests for at least 5 seconds after a reset. This allows time for other devices to read the snapshot values before they are overwritten. If the option "Confirm Reset" in **INTEG_OPTS** is set, another reset is prevented from occurring until the value 1 has been written to **RESET_CONFIRM**. This procedure provides a guarantee that a host has recorded the snapshot values before the next reset occurs.

The number of resets is counted in the register **N_RESET**. This counter can not be written or reset. It provides verification that the total has not been reset since N_RESET was last checked. The counter rolls over from 999999 to 0.

Reset always clears the internal registers Total, Atotal and Rtotal. If the option "UP_AUTO" or "DN_AUTO" is selected in **INTEG_TYPE** and the option "Carry" is selected in **INTEG_OPTS**, however, a residual value beyond the trip value may be carried to the next integration. In this case, TOTAL_SP is subtracted from Total, leaving the residual value.

The option "Generate reset event" in **INTEG_OPTS** causes an analog event to be generated at each reset. This provides a timestamp, the value of Total prior to the reset in units of OUT_RANGE and the substatus of OUT.

Batch totalizer outputs

When the integration is counting up, it starts from zero and counts to **TOTAL_SP**; When counting down, it starts from **TOTAL_SP** and counts to zero. The discrete output **OUT_TRIP** indicates when the target count is reached.

The parameter **PRE_TRIP** allows a warning to be output in the form of the discrete signal **OUT_PTRIP**. The conditions are summarized in the table below.

INTEG_TYPE	OUT	OUT_PTRIP	OUT_TRIP
"UP_AUTO" or "UP_DEM"	< TOTAL_SP - PRE_TRIP	"0" (False)	"0" (False)
	≥ TOTAL_SP - PRE_TRIP and < TOTAL_SP	"1" (True)	"0" (False)
	≥ TOTAL_SP	"1" (True)	"1" (True)
"DOWN_AUTO" or "DOWN_DEM"	> PRE_TRIP	"0" (False)	"0" (False)
	≤ PRE_TRIP and >0	"1" (True)	"0" (False)
	= 0	"0" (False)	"0" (False)

OUT_TRIP remains set for five seconds after an automatic reset (**INTEG_TYPE** options "UP_AUTO" or "DOWN_AUTO"). If "Confirm Reset" has been selected in **INTEG_OPTS** and or **RESET_CONFIRM** is connected, it remains set until the reset has been confirmed.

Status propagation

The input status is checked and handled as described below, whereby when two inputs are in use, the resulting status is the worst of the two. The limit status of the inputs is ignored..

INTEG_OPTS	Input status	Output status	Remarks
–	Good NonCascade	Good NonCascade	Normal operation
–	Good Cascade	Good Cascade	Normal operation
–	Uncertain	Bad	Last good value is used as increment
Use Uncertain	Uncertain	Good	New value is used as increment
–	Bad	Bad	Last good value is used as increment
Use Bad	Bad	Good	New value is used as increment
Add zero if Bad	Bad	Good	Zero is used as increment

If the block mode is Man, then the status of **OUT**, **OUT_PTRIP**, and **OUT_TRIP** will be Good (NC) constant. If the option "Uncertain if Man" is selected in the **STATUS_OPTS** parameter, then the status of these three outputs will be Uncertain constant. No limits are applied to the output.

Percentage bad counts

The parameter **PCT_INCL** indicates the percentage of good counts in the current total. This parameter can be limited by entering an allowable level of rejects in percent in the parameters **GOOD_LIM** and **UNCERT_LIM**. These parameters influence the **OUT** status when the block is in Auto according to the table below:

Condition	Output status
PCT_INCL ≥ GOOD_LIM	Good NonCascade
GOOD_LIM > PCT_INCL ≥ UNCERT_LIM	Uncertain
PCT_INCL < UNCERT_LIM	Bad

7.5.2 Block configuration

Control strategy: Rate

Fig. 7-19 shows a control strategy example together with the links that must be made between the blocks. Here, a vessel is filled with a quantity of Liquid 1 determined by **PRE_TRIP**, with the flowrate being measured by Flowmeter 1. Afterwards, Liquid 2 is added until **TOTAL_SP** is reached, the flowrate being measured by Flowmeter 2. The flowmeters indicate reverse flow by a discrete signal (DI links are omitted if flowmeter sends positive and negative flow values). The **OUT** value is used to record the filling of the vessel. The discrete outputs **OUT_PTRIP** and **OUT_TRIP** are used e.g. to switch off pumps and close valves (can be connected to more than one DO block).

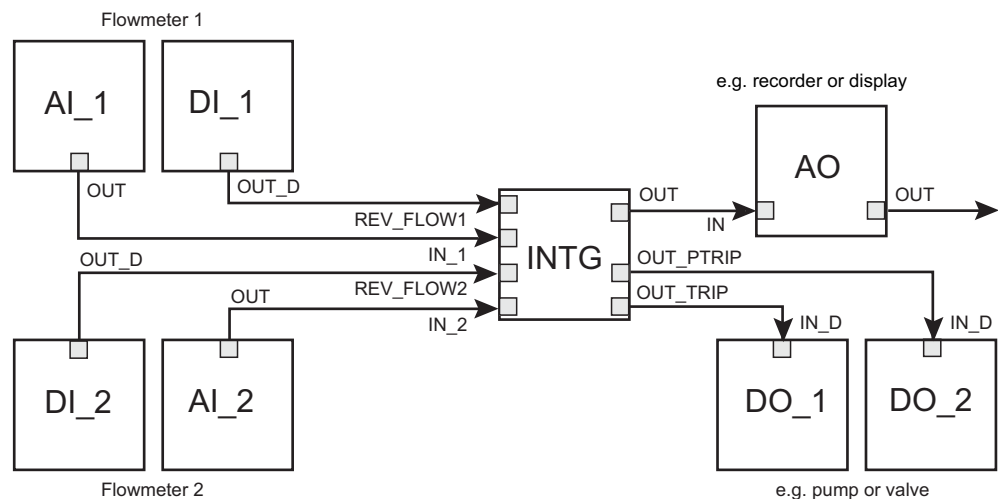


Fig. 7-19: Links to be made for Integrator block for control strategy described above

Block configuration: Rate

The integration block is configured for the strategy in Fig 7-19 as follows:

Parameter	Action	Option/Value
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
TOTAL_SP	For batching, enter target quantity	e.g. 10 000
OUT_RANGE.EU_100	Enter upper range limit for the input variables (HMI)	e.g. 10 000
OUT_RANGE.EU_0	Enter lower range limit for input variables (HMI)	e.g. 0
OUT_RANGE.UNITS_INDEX	Enter unit of the input variables (HMI)	e.g. 1
TIME_UNIT1	Select time units in which the rate 1 is measured: ■ Select "days", "hours", "minutes" or "seconds"	e.g. min (for l/min)
TIME_UNIT2	Select time units in which the rate 2 is measured: ■ Select "days", "hours", "minutes" or "seconds"	e.g. min (for l/min)
UNIT_CONV	Enter conversion top match input 2 to input 1 ■ Default 1, in our case no entry required	1
INTEG_TYPE	Select type of integration and reset ■ UP_AUTO: counts up, automatic reset	UP_AUTO
INTEG_OPTS	Select flow totalization type ■ FORWARD: forward flow is counted	FORWARD
PRE_TRIP	For batching, enter target quantity at which discrete output OUT_PTRIP is to switch	e.g. 6 000

The block will totalize all sampled rates with a good status. Rates with bad and uncertain status can also be totalized if the appropriate option is set in **INTEG_OPTS**, see Chapter 7.5.3, Block operation.

Control strategy: Accum

Fig. 7-20 shows a control strategy in which the Integrator block operates with the **OUT_ACCUM** parameter of a pulse input block. In this case, the Integrator is to count for a period determined by the clock or until stopped on demand. The demand reset signal is provided by a separate Discrete Input block. The **OUT** value is e.g. used by an Analog Output block connected to a recorder or display. It is desired that both negative and positive flows (=TOTAL) as well as flows with uncertain and bad status (= ANY) should be counted.

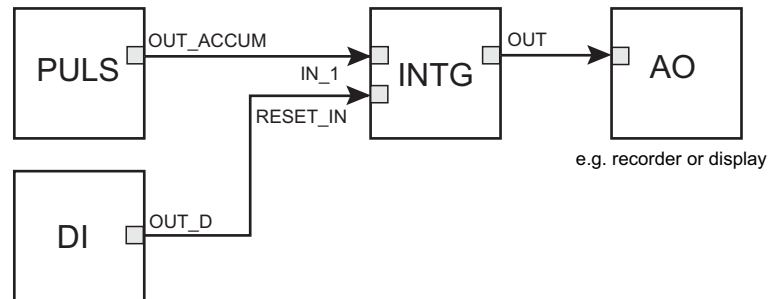


Fig. 7-20: Links to be made for the Integrator block for the control strategy described above

Block configuration: Accum

The integration block is configured for the strategy in Fig 7-19 as follows:

Parameter	Action	Option/Value
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
OUT_RANGE.EU_100	Enter upper range limit for the input variables (HMI)	e.g. 10 000
OUT_RANGE.EU_0	Enter lower range limit for input variables (HMI)	e.g. 0
OUT_RANGE.UNITS_INDEX	Enter unit of the input variables (HMI)	e.g. 1
PULS_VAL1	Enter value associated with each pulse	e.g. 100 (l)
INTEG_TYPE	Select type of integration and reset <ul style="list-style-type: none"> ■ PER&DEM: counts up until reset at CLOCK per or by externally, whichever comes first 	PER&DEM
INTEG_OPTS	Select flow totalization type <ul style="list-style-type: none"> ■ FORWARD: forward flow is counted ■ REVERSE: reverse flow is counted ■ Use Uncertain: counts when input status "Uncertain" ■ Use Bad: uses last "Good" or "Uncertain" value when status is "Bad" 	FORWARD REVERSE Use Uncertain Use Bad
CLOCK_PER	Enter period of count until reset in seconds	e.g. 3 600

7.5.3 Block operation

Mode

The block normally operates in Auto mode and takes its input from **IN_1** and **IN_2**. If there is a block configuration error, the block switches to OOS (Out of Service).

When **MODE_BLK.Target** is set to Man, the block output can be manipulated on-line by entering the desired status and value in **OUT**, **RTOTAL**, **OUT_TRIP** and **OUT_PTRIP**. No integration takes place. Each write to **OUT** or **RTOTAL** increments the **N_RESET** counter. When the block is switched to Auto again, the integration starts from the value set manually.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller. An exception to this rule is the parameter **OP_CMD_INT** which can be used to reset the totalizer when the block is in Auto mode.

Operation can be checked on-line by viewing a number of parameters:

REV_FLOW1/ REV_FLOW2	Value and status of reverse flow input: false = forward flow, true = reverse flow
IN_1/IN_2	Value and status of inputs 1 and 2
OUT	Value and status of totalizer
OUT_PTRIP	Value and status of pre-trip indicator: see batch totalizer outputs
OUT_TRIP	Value and status of trip indicator: see batch totalizer outputs

Status

When two inputs are in use, the worst status of the two is propagated to the output. The significance of the output status is indicated below.

Output status	Meaning
Good NonCascade	Normal operation as configured
Good Cascade	Normal operation as configured
Uncertain	<ul style="list-style-type: none"> Block is in Man mode and "Uncertain if Man" selected in STATUS_OPTS PCT_INCL is limited and $GOOD_LIM > PCT_INCL \geq UNCERT_LIM$
Bad	<ul style="list-style-type: none"> Block out of service One or both inputs have "Bad" or "Uncertain" status PCT_INCL is limited and $PCT_INCL < UNCERT_LIM$

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Block configuration error	<ul style="list-style-type: none"> No option selected in INTEG_TYPE For rate input, no time unit entered in TIME_UNITx For batching, $PRE_TRIP > TOTAL_SP$
Out-of-Service	<ul style="list-style-type: none"> MODE_BLK.Target currently set to OOS

7.5.4 Block parameters

Integrator block

Parameter	Valid range/ Options	Default value	Description
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set MODE_BLK.Target to AUTO, CAS or RCAS
BLOCK_ERR	RO		Block errors, see Chapter 6.1.9
TOTAL_SP	0 - 999999	0	Target count for batching – For INTEG_TYPE options "UP_AUTO" or "DN_AUTO"
OUT		0	Block output value and status
OUT_RANGE		0	Block output range and units for HMI
GRANT_DENY		0	Access options, see Chapter 2.8.1
STATUS_OPTS		0	Status options, see Chapter 2.8.4 and Chapter 6.1.9
IN_1		RO	Value and status of input 1
IN_2		RO	Value and status of input 2
OUT_TRIP	0/1	0	Value and status of trip (TOTAL_SP) indicator – For INTEG_TYPE options "UP_AUTO" or "DN_AUTO"
OUT_PTRIP	0/1	0	Value and status of PRE_TRIP indicator – For INTEG_TYPE options "UP_AUTO" or "DN_AUTO"
TIME_UNIT1		–	Rate units of input 1 (Days, hours, minutes, seconds) – For INTEG_OPTS "Flow forward" and "Flow reverse"
TIME_UNIT2		–	Rate units of input 2 (Days, hours, minutes, seconds) – For INTEG_OPTS "Flow forward" and "Flow reverse"
UNIT_CONV		1	Conversion factor for input 2 to input 1 units
PULS_VAL1		0	Pulse value for input 1 – For INTEG_OPTS "Input 1 accumulate"
PULS_VAL2		0	Pulse value for input 2 – For INTEG_OPTS "Input 2 accumulate"

Parameter	Valid range/ Options	Default value	Description
REV_FLOW1	0/1		Value and status of reverse flow indicator input 1 – 0 = forward, 1 = reverse
REV_FLOW2	0/1		Value and status of reverse flow indicator input 2 – 0 = forward, 1 = reverse
RESET_IN	0/1		Value and status of external reset input 1
STOTAL		RO	Snapshot of OUT value immediately prior to reset
RTOTAL			Current total including bad and uncertain counts – See 7.5.1, Status handling
SRTOTAL		RO	Snapshot of RTOTAL immediately prior to reset
SSP		RO	Snapshot of TOTAL_SP immediately prior to reset – For INTEG_TYPE options "UP_AUTO" or "DN_AUTO"
INTEG_TYPE		–	Type of integration and reset <ul style="list-style-type: none"> ■ UP_AUTO: counts up, automatic reset ■ UP_DEM: counts up, external reset ■ DN_AUTO: counts down, automatic reset ■ DN_DEM: counts down, external reset ■ PERIODIC: counts up until reset at CLOCK_PER ■ DEMAND: counts up until externally reset ■ PER&DEM: counts up until reset at CLOCK per or by externally, whichever comes first
INTEG_OPTS			Integration options, multiple selections possible <ul style="list-style-type: none"> ■ Input 1 accumulate: accumulated value at input 1 ■ Input 2 accumulate: accumulated value at input 2 ■ Flow forward: forward flow values counted ■ Flow reverse: reverse flow values counted ■ Use Uncertain: values counted when status Uncertain ■ Use Bad: last good value counted when status Bad ■ Carry: residual value carried over to next count cycle (For INTEG_TYPE options "UP_AUTO" or "DN_AUTO") ■ Add zero if bad: zero counted when status Bad ■ Confirm reset: next count cycle begins only after reset has been confirmed ■ Generate reset event: event generated on reset
CLOCK_PER		0 s	Period of count in seconds – For INTEG_TYPE options "PERIODIC" and "PER&DEM"
PRE_TRIP	0 – 999999	0	Intermediate target count for batching – For INTEG_TYPE options "UP_AUTO" or "DN_AUTO"
N_RESET		RO	Number of resets made since commissioning of project
PCT_INCL		RO	Percentage of good counts in total
GOOD_LIM	0 – 100		Limit value for PLC_INCL, see 7.5.1, Percentage bad counts
UNCERT_LIM	0 – 100		Limit value for PLC_INCL, see 7.5.1, Percentage bad counts
OP_CMD_INT			Internal command reset command <ul style="list-style-type: none"> ■ Select "Reset" in Auto mode
OUTAGE_LIM			Maximum tolerated duration of power failure <ul style="list-style-type: none"> ■ Not supported
RESET_CONFIRM			Value and status of external reset confirmation input
UPDATE_EVT			This alert is generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
Legend: RO = Read Only			

7.6 Analog Alarm

The Analog Alarm block provides alarm condition reporting on an analog output of any block. Alarm conditions include high-high, high, low, and low-low alarms. The associated limits may be user-entered or computed based on gains and biases from a process setpoint input, thus providing dynamic deviation alarming. A discrete output is provided which indicates the existence of one or more (configurable) alarm conditions.

An option to temporarily expand alarm limits after a setpoint change is provided. In addition, a new alarm condition may be ignored for a specified period of time after detection to avoid nuisance alarm reporting. The block can also be used as a comparator by comparing the input value to the process setpoint.

In ControlCare Application Designer the Analog Alarm block (AAL block) is assigned the generic name **Device Tag-AALM-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 7-21.

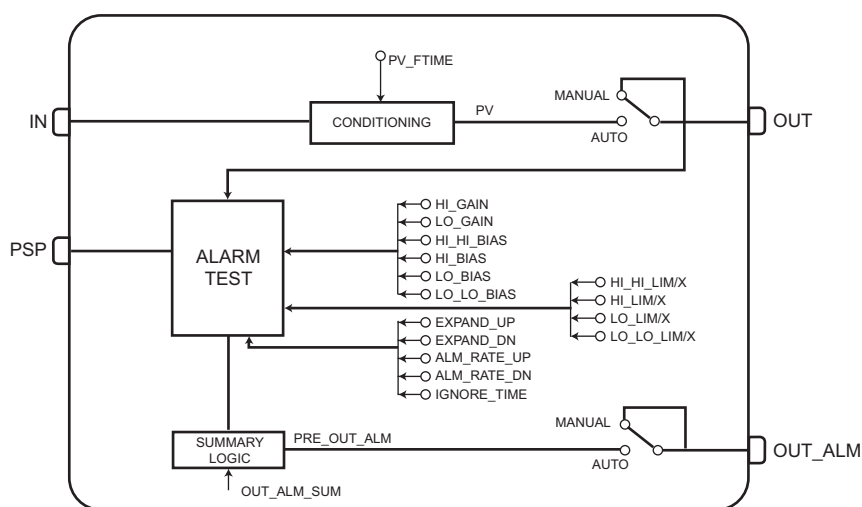


Fig. 7-21: Schematic diagram of Analog Alarm block

7.6.1 Functional description

The Analog Alarm block has two inputs, **IN**, which is connected to an analog output of an upstream block and **PSP**, which may be connected to a process setpoint input. The latter is used to compute alarm limits which may also be dynamically adjusted on a change in process setpoint. After conditioning with a time filter **PV_FTIME**, the analog input value becomes the process value **PV**, which is made available to downstream blocks at the analog output **OUT**. The parameter **OUT_RANGE** is used make the range and engineering units of **OUT** to a HMI.

In Auto mode, the process value **PV** is also alarmed, whereby the current value is compared to user-entered or computed limits. **OUT_ALM_SUM** determines which alarm conditions are to be signalled and a discrete signal **PRE_OUT_ALM** is generated for output as **OUT_ALM**. This has the value "0" (false) when no alarms are present and "1" (true) when one or more limits have been violated.

User-entered limits

When used as a standard alarm block, the user may enter up to four limits in the engineering units of the incoming **IN** signal, **LO_LO_LIM**, **LO_LIM**, **HI_LIM** and **HI_HI_LIM**, together with an alarm hysteresis **ALARM_HYS** as a percentage of the input span. Each alarm can be assigned a priority in the parameters **LO_LO_PRI**, **LO_PRI**, **HI_PRI** and **HI_HI_PRI**. The parameter **ACK_OPTION** enables automatic acknowledgement of the alarms, which must otherwise be individually acknowledged in the associated **LO_LO_ALM**, **LO_ALM**, **HI_ALM** and **HI_HI_ALM** parameters. A full description of alert handling is to be found in Chapter 2.9.

Computed limits

When the process setpoint **PSP** is connected to the block, alarm limits may be dynamically calculated. The operating limits (with suffix "X") are calculated from specified gains and biases as follows:

- $HI_HI_LIMX = PSP * HI_GAIN + HI_HI_BIAS + EXPAND_UP$
- $HI_LIMX = PSP * HI_GAIN + HI_BIAS + EXPAND_UP$
- $LO_LIMX = PSP * LO_GAIN - LO_BIAS - EXPAND_DN$
- $LO_LO_LIMX = PSP * LO_GAIN - LO_LO_BIAS - EXPAND_DN$

The operating limits default to **HI_HI_LIM**, **HI_LIM**, **LO_LIM** and **LO_LO_LIM** respectively if any parameter is undefined.

Alarm limit expansion

In order to avoid nuisance alarms, the effective alarm limits are temporarily expanded on step setpoint change. High alarm limits are increased by a calculated term, **EXPAND_UP** and low alarm limits are decreased by a calculated term, **EXPAND_DN**. The expansion terms correspond to the absolute value of the step change.

In order to avoid alarms on the initial change in normal and over-damped process responses and on overshooting or ringing in under-damped process, the expansions decay toward the base limits at a rate determined by **ALM_RATE_UP** and **ALM_RATE_DN**. Fig. 7-22 illustrates the function.

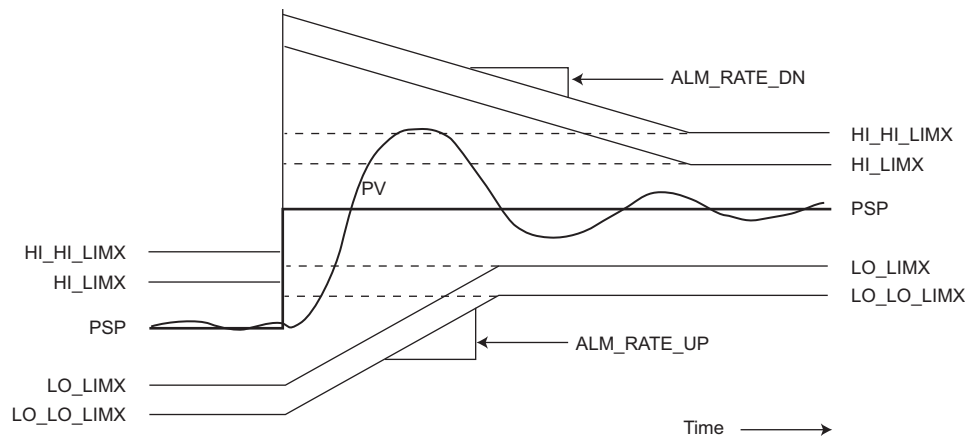


Fig. 7-22: Alarm limit expansion on step change in process setpoint

The expansion feature may be suppressed in the upwards or downwards direction by either not entering a value in the parameters **ALM_RATE_UP** and **ALM_RATE_DN** or setting them to zero.

Ignoring alarm conditions

The existence of a new alarm condition may be temporarily ignored by setting the **IGNORE_TIME** parameter to the number of seconds to disregard the alarm. Both the reporting of the alarm and the possible change to **PRE_OUT_ALM** will be ignored during this time. This parameter does not delay the clearing of the existence of the alarm on return-to-normal. If the alarm condition does not persist for **IGNORE_TIME** seconds, it will not be reported.

Alarm summary

The **OUT_ALM_SUM** parameter specifies the alarm conditions to be signalled at the **OUT_ALM** output. The following options can be selected:

Option OUT_ALM_SUM	Alarm conditions included			
	HI_HI_ALM	HI_ALM	LO_ALM	LO_LO_ALM
ANY	X	X	X	X
LOWs			X	X
HIGHs	X	X		
LEVEL1		X	X	
LEVEL2	X			X
LO_LO				X
LO			X	
HI		X		
HI_HI	X			
NONE				

Alarm acknowledgement

Current alarms are summarized in the **ALARM_SUM** parameter. If the user has opted to acknowledge individual alarms, this must be done in the corresponding parameter xxx_ALM. The structure of the alarm parameter is as shown below:

Parameter	Attribute	Value	Description
ALARM_SUM HI_HI_ALM HI_ALM LO_ALM LO_LO_ALM	UNACKNOWLEDGED	As read or entered	Indicates acknowledgement of current alert state: i.e. Undefined, Acknowledged, Unacknowledged ■ User can acknowledge via this parameter
	ALARM_STATE	As read	Indicates of whether the alert is active and reported. The alarm state is cleared when the block goes to Out of Service mode. Possible values are: ■ Clear-Reported, Clear-Not Reported, Active-Reported, Active-Not Reported
	TIME_STAMP	As read	Indicates time when the alarm state was detected. The value is maintained until alert confirmation has been received.
	SUB_CODE	As read	Displays number specifying the cause of the alert
	VALUE	As read	The value of the associated parameter at the time the alert was detected

Status propagation

The status of the **OUT** value is checked and handled as described below..

STATUS_OPTS	IN status	OUT status	Remarks
–	Good	Good	IN filtered and used as PV => OUT
Use Uncertain as Good	Uncertain	Uncertain	IN filtered and used as PV => OUT
–	Bad, Uncertain	Bad, Uncertain	IN not filtered, last usable PV => OUT

The status of the **OUT_ALM** depends upon the status of both **IN** and **PSP**, see below:

STATUS_OPTS	IN status	PSP status	OUT_ALM status	Remarks
–	Good	Good	Good	Alarm test performed
Use Uncertain as Good	Good	Uncertain	Uncertain	Alarm test performed
Use Uncertain as Good	Uncertain	Good	Uncertain	Alarm test performed
–	Good	Bad, Uncertain	Bad	Alarm test not performed
–	Bad, Uncertain	Good	Bad	Alarm test not performed
–	Bad, Uncertain	Bad, Uncertain	Bad	Alarm test not performed

While the alarm test is not being performed, existing alarms will not be cleared and new alarms will not be generated. Alarm conditions existing prior to the unusable status may still be acknowledged.

When operating in manual mode, the limit status of **OUT_ALM** is set to double-limited.

7.6.2 Block configuration

**Control strategy:
user limits**

Fig. 7-23 shows a control strategy example together with the links that must be made between the blocks. Here, an analog value, e.g. the level of liquid in a vessel, is monitored for two limits. When the level reaches the high limit, the inlet valve is to be closed and the outlet valve opened. The outlet valve is to be closed and the inlet valve opened when the level reaches the low limit. Alarms are to be automatically acknowledged. The control logic is done in a discrete hybrid block, which outputs to a Multiple Discrete Output block connected to the valves via e.g. a relay module. An Analog Output block connects the analog value to a display or recorder which displays the current level in the tank.

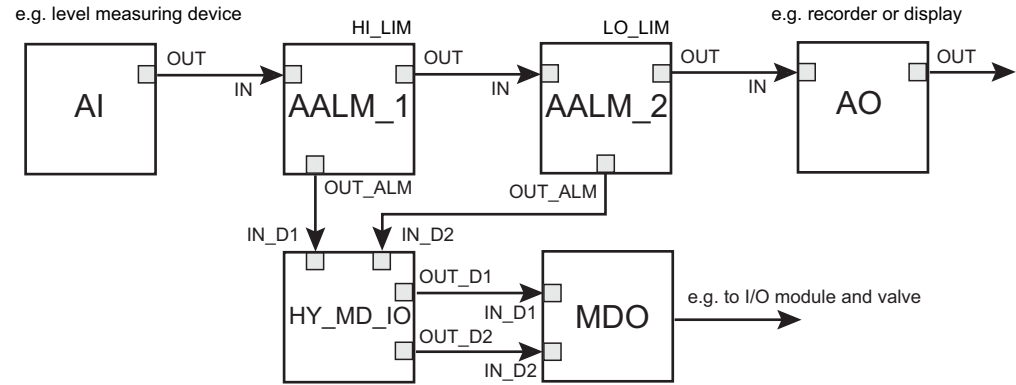


Fig. 7-23: Links to be made for Analog Alarm block for control strategy described above

**Block configuration:
user limits**

The two Analog Alarm blocks are configured for the strategy in Fig 7-23 as follows:

Parameter	Action	Option/Value
Block AALM_1		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
OUT_RANGE.EU_100	Enter upper range limit for the input variables (HMI)	e.g. 100
OUT_RANGE.EU_0	Enter lower range limit for input variables (HMI)	e.g. 0
OUT_RANGE.UNITS_INDEX	Enter unit of the input variables (HMI)	e.g. %
PV_FTIME	Enter time constant in seconds for 1st. order damping	e.g. 3
ALARM_HYS	Enter alarm hysteresis in % of OUT_RANGE	e.g. 1
HI_LIM	Enter limit of HI alarm in engineering units	e.g. 90
Block AALM_2		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
OUT_RANGE.EU_100	Enter upper range limit for the input variables (HMI)	e.g. 100
OUT_RANGE.EU_0	Enter lower range limit for input variables (HMI)	e.g. 0
OUT_RANGE.UNITS_INDEX	Enter unit of the input variables (HMI)	e.g. %
PV_FTIME	Enter time constant in seconds for 1st. order damping	e.g. 0
ALARM_HYS	Enter alarm hysteresis in % of OUT_RANGE	e.g. 1
LO_LIM	Enter limit of LO alarm in engineering unit	e.g. 10

Control strategy:
computed limits

Fig. 7-24 shows a control strategy example together with the links that must be made between the blocks. Here, a liquid is to be heated and maintained with a set temperature band (Setpoint $\pm 5^{\circ}\text{C}$) by an immersion heater. Control is by simply switching on and off the heater when the temperature leaves the set band, the logic being done in a discrete hybrid block. The setpoint is set remotely, in this case by a constant block and can change, hence the limits must be dynamic. The temperature is measured by a RTD thermometer and monitored by a recorder or display.

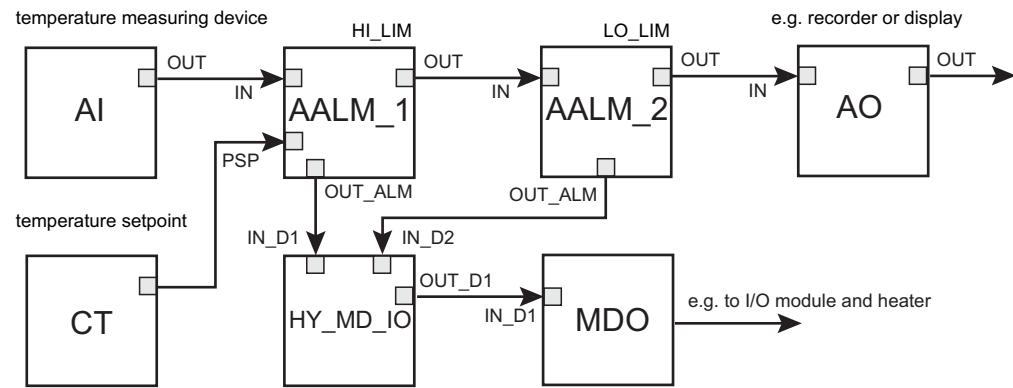


Fig. 7-24: Links to be made for Analog Alarm block for control strategy described above

Block configuration:
computed limits

The two Analog Alarm blocks are configured for the strategy in Fig 7-24 as follows:

Parameter	Action	Option/Value
Block AALM_1		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
OUT_RANGE.EU_100	Enter upper range limit for the input variables (HMI)	e.g. 200
OUT_RANGE.EU_0	Enter lower range limit for input variables (HMI)	e.g. 0
OUT_RANGE.UNITS_INDEX	Enter unit of the input variables (HMI)	e.g. $^{\circ}\text{C}$
PV_FTIME	Enter time constant in seconds for 1st. order damping	e.g. 3
HI_GAIN	Enter gain value for HI limit	1
HI_BIAS	Enter bias value for HI limit	5
ALM_RATE_UP	Enter ramp rate for HI limits in engineering units/s	e.g. 2
Block AALM_2		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
OUT_RANGE.EU_100	Enter upper range limit for the input variables (HMI)	e.g. 200
OUT_RANGE.EU_0	Enter lower range limit for input variables (HMI)	e.g. 0
OUT_RANGE.UNITS_INDEX	Enter unit of the input variables (HMI)	e.g. $^{\circ}\text{C}$
PV_FTIME	Enter time constant in seconds for 1st. order damping	e.g. 0
LO_GAIN	Enter gain value for HI limit	1
LO_BIAS	Enter bias value for HI limit	5
ALM_RATE_DN	Enter ramp rate for HI limits in engineering units/s	e.g. 2

The configured blocks maintain the temperature within band of $\pm 5^{\circ}\text{C}$ on a change of setpoint.

7.6.3 Block operation

Mode

The block normally operates in Auto mode and takes its input from **IN** and, if connected, **PSP**.

When **MODE_BLK.Target** is set to Man, the block output can be manipulated on-line by entering the desired status and value in **OUT** and **OUT_ALM**.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Operation can be checked on-line by viewing a number of parameters:

IN	Value and status of input
PSP	Value and status of process setpoint, if connected
HI_HI_LIMX	Value and status of dynamic HI_HI limit, if used
HI_LIMX	Value and status of dynamic HI limit, if used
LO_LIMX	Value and status of dynamic LO limit, if used
LO_LO_LIMX	Value and status of dynamic LO_LO limit, if used
OUT	Value and status of analog output
OUT_ALM	Value and status of discrete alarm output
OUT_TRIP	Value and status of trip indicator: see batch totalizer outputs

Status

When two inputs are in use, the worst status of the two is propagated to the output. The significance of the output status is indicated below.

OUT/OUT_ALM status	Meaning
Good	Normal operation as configured
Uncertain	Block has an "Uncertain" input and "Use Uncertain as Good " selected in STATUS_OPTS
Bad	<ul style="list-style-type: none"> ■ Block out of service ■ One or both inputs have "Bad" or "Uncertain" status

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Out-of-Service	<ul style="list-style-type: none"> ■ MODE_BLK.Target currently set to OOS

7.6.4 Block parameters

Analog Alarm block

Parameter	Valid range/ Options	Default value	Description
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set MODE_BLK.Target to AUTO, CAS or RCAS
BLOCK_ERR	RO		Block errors, see Chapter 6.1.9
OUT		0	Block output value and status
OUT_RANGE		0	Block output range and units for HMI
GRANT_DENY		0	Access options, see Chapter 2.8.1
STATUS_OPTS		0	Status options, see Chapter 2.8.4 and Chapter 7.6.1
PV_FTIME		0 s	Time constant (s) of a single exponential filter for process value
IN	RO		Value and status of input
PSP	RO		Value and status of process set point
HI_GAIN		+INF	Gain value for calculation of dynamic HI and HI_HI limits
LO_GAIN		+INF	Gain value for calculation of dynamic LO and LO_LO limits
HI_HI_BIAS	Positive	+INF	Bias value for calculation of dynamic HI_HI limit
HI_BIAS	Positive	+INF	Bias value for calculation of dynamic HI limit
LO_BIAS	Positive	+INF	Bias value for calculation of dynamic LO limit"
LO_LO_BIAS	Positive	+INF	Bias value for calculation of dynamic LO_LO limit"
PRE_OUT_ALM	RO		Display value for OUT_ALM value and status in Auto mode
OUT_ALM			Display value for actual OUT_ALM value and status
OUT_ALM_SUM			Selection parameter for OUT_ALM signal conditions
ALM_RATE_UP		0	Ramp rate for HI limits in engineering units/s
ALM_RATE_DN		0	Ramp rate for HI limits in engineering units/s
EXPAND_UP	RO		Calculated shift of HI and HI_HI limits on PSP step
EXPAND_DN	RO		Calculated shift of LO and LO_LO limits on PSP step
IGNORE_TIME	Positive	0	Time delay between detection and reporting of limit violation
UPDATE_EVT			Alert generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
ALARM_SUM			Alarm summary, see Chapter 2.9
ACK_OPTION	0, 1	0	See Chapter 2.9 <ul style="list-style-type: none"> ■ 0: Auto ACK Disable ■ 1: Auto ACK Enable
ALARM_HYS	0 to 50 %	0.5%	Alarm hysteresis in % of OUT_SCALE, see Chapter 2.9
HI_HI_PRI	0 to 15		Priority of the HI_HI alarm, see Chapter 2.9
HI_HI_LIM		+INF	Setting of HI_HI alarm in engineering units, see Chapter 2.9.
HI_HI_LIMX			Computed HI_HI operating limit
HI_PRI	0 to 15		Priority of the HI alarm, see Chapter 2.9
HI_LIM		+INF	Setting of HI alarm in engineering units, see Chapter 2.9.
HI_LIMX			Computed HI operating limit
LO_PRI	0 to 15		Priority of the LO alarm, see Chapter 2.9.
LO_LIM		-INF	Setting of LO alarm in engineering units, see Chapter 2.9.
LO_LIMX			Computed LO operating limit
LO_LO_PRI	0 to 15		Priority of the LO_LO alarm, see Chapter 2.9
LO_LO_LIM		-INF	Setting of LO_LO alarm in engineering units, see Chapter 2.9
LO_LO_LIMX			Computed LO_LO operating limit
HI_HI_ALM			Status of HI_HI alarm and its associated time stamp.
HI_ALM			Status of HI alarm and its associated time stamp.
LO_ALM			Status of LO alarm and its associated time stamp.
LO_LO_ALM			Status of LO_LO alarm and its associated time stamp.
Legend: RO = Read Only			

7.7 Enhanced Analog Alarm

The Enhanced Analog Alarm block differs from the Analog Alarm block in one aspect only. It has an additional discrete output **OUT_D** which maps the **OUT_ALM** output. The **OUT_D** signals can be inverted, however, when the **INVERT_OPTS** parameter is set to "4".

In ControlCare Application Designer the Enhanced Analog Alarm block is assigned the generic name **Device Tag-EAAL-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 7-25.

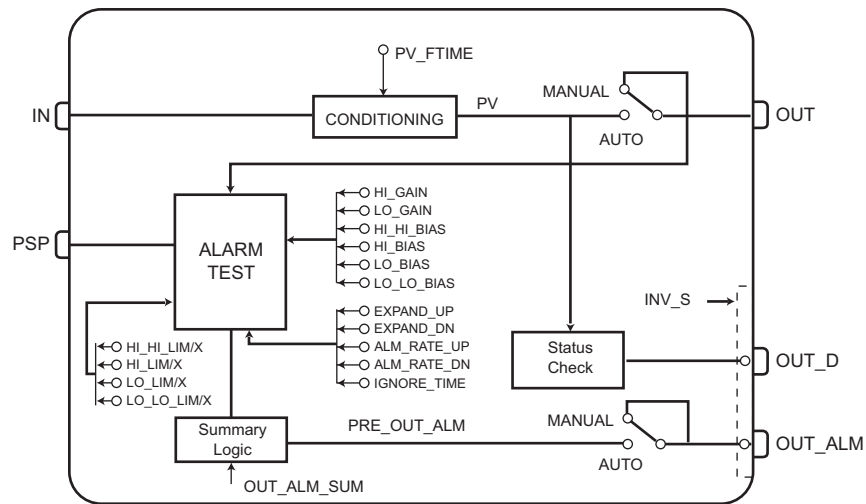


Fig. 7-25: Schematic diagram of Enhanced Analog Alarm block

Additional parameters

Parameter	Valid range/Options	Default value	Description
OUT_D			Display value for actual OUT_D value and status
INVERT_OPTS			Inverts OUT_D signals when set to "4" (=Invert)
Legend: RO = Read Only			

7.8 Input Selector

The Input Selector block provides selection of four inputs, normally from analog input blocks, and generates an output based on the configured action. It supports two actions:

- Internal signal selection, based on maximum, minimum, middle, average and 'first good' algorithms
- External signal selection using the the **DISABLE_n** parameter.

The **DISABLE_N** parameter can also be used in Case 1 to generate a validated priority selection.

The block supports the concept of a middle selection. Although the normal configuration for this feature would be with three signals, the block will generate an average of the middle two if four signals are configured or the average of two if three are configured and a bad status is passed to one of the inputs. Logic is provided for handling uncertain and bad signals in conjunction with configured actions.

The intended application of this block is to provide control signal selection in the forward path only, therefore, no back calculation support is provided.

In ControlCare Application Designer the Input Selector block (IS Block) is assigned the generic name **Device Tag-ISEL-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 7-26.

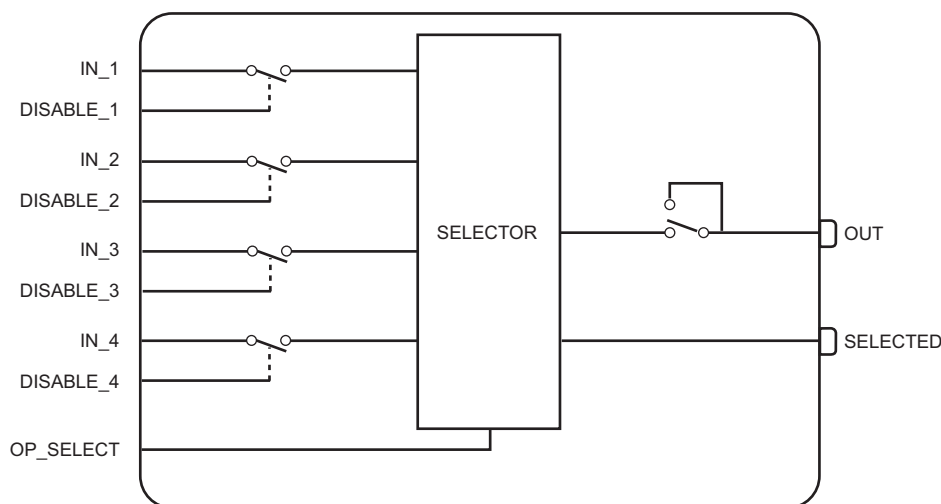


Fig. 7-26: Schematic diagram of Input Selector block

7.8.1 Functional description

The Analog Alarm block has four inputs, **IN_1** to **IN_4**, which may be connected to the **OUT** outputs of upstream Analog Input blocks. Four discrete inputs **DISABLE_1** to **DISABLE_4**, which may be connected to discrete signals, control the signal processing of the associated analog input. If **DISABLE_n** is true, the corresponding signal is not used in the execution of the block.

After execution of the chosen selection algorithm, the result is made available at the **OUT** output. The parameter **OUT_RANGE** is used make the range and engineering units of **OUT** known to a HMI.

The analog input **OP_SELECT** allows the selection algorithm or a single input to be selected externally. **SELECTED** outputs an integer that normally corresponds to the number of the input used.

The block is not intended for use with controller outputs and has no back calculation support or propagation of control status values.

Selection algorithm

The selection process is controlled by the external parameter **OP_SELECT** or the internal parameter **SELECT_TYPE**.

- If **OP_SELECT** is zero or no signal is connected, the selection is made in **SELECT_TYPE**
- If **OP_SELECT** is greater than zero, then this value is used to select the Input (1 to 4) to be used as output by the block. The selection made by the **SELECT_TYPE** algorithm is then ignored.

The options available in **SELECT_TYPE/OP_SELECT** are shown below. **MIN_GOOD** allows the operator to set a limit on the number of bad signals allowed for a particular algorithm

SELECT_TYPE when OP_SELECT = 0	Meaning
No Selection	All inputs bad, or number of Good inputs < MIN_GOOD – If OP_SELECT = 0, the selection is always made with SELECT_TYPE
First Good	The value of the first usable input is transferred to the output of the block. – SELECTED is set to the number of the input used
Minimum	The usable inputs are sorted by value and the lowest value is transferred to the output of the block. – SELECTED is set to the number of the input used
Maximum	The usable inputs are sorted by value and the highest value is transferred to the output of the block. – SELECTED is set to the number of the input used
Middle	The usable inputs are sorted by value and the highest and lowest values discarded. If two values are remaining or there were only two values originally, their average is computed. The value is transferred to the output of the block. – SELECTED is set to zero if an average was used, otherwise it contains the number of the input with the middle value.
Average	The average of the usable inputs is calculated and the value is transferred to the output of the block. – SELECTED is set to the number of inputs used in the average.

Status propagation

The block checks the status of the inputs before processing them. Inputs with status Good or Uncertain are processed, those with status Bad are ignored. The table below summarizes the status of **OUT** under various conditions. The **SELECTED** output always has Good(NC) status, unless the block is out of service..

STATUS_OPTS	IN status	Usable inputs	OUT status	Remarks
–	Good	>MINGOOD	Good	Block executed
–	Good	<MINGOOD	Bad	Block not executed
–	Uncertain	>MINGOOD	Uncertain	Block executed
Use Uncertain as Good	Uncertain	>MINGOOD	Good	Block executed
–	Uncertain	<MINGOOD	Bad	Block not executed
–	Bad	<MINGOOD	Bad	Block not executed
–	All Bad	–	Bad	Block not executed, SELECTED = 0
–	Non connected	–	Bad	Block not executed, SELECTED = 0

When the block is operating in Man mode, the block is not executed and the following applies:

STATUS_OPTS	OUT status	Remarks
–	As entered	Block not executed
Use Uncertain if Man	Uncertain	Block not executed

7.8.2 Block configuration

Control strategy:
internal priority selection

Fig. 7-27 shows a control strategy example for internal priority selection together with the links that must be made between the blocks. Here, a critical temperature measuring point has two redundant temperature transmitters. The first valid reading is taken for the subsequent control task.

The application can be upgraded to a validated priority selection, in that the disable signals **DISABLE_1** and **DISABLE_2** are used to control the status of the associated inputs. When **DISABLE_n** is true, the **IN_n** is disabled and will not be considered by the "First Good" algorithm.

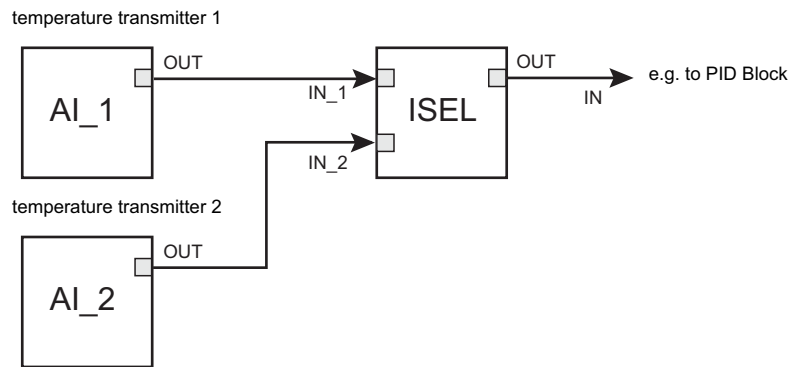


Fig. 7-27: Links to be made for Input Selector block for control strategy described above

Block configuration:
internal priority selection

The Input Selector block is configured for the strategy in Fig 7-27 as follows:

Parameter	Action	Option/Value
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
OUT_RANGE.EU_100	Enter upper range limit for the input variables (HMI)	e.g. 200
OUT_RANGE.EU_0	Enter lower range limit for input variables (HMI)	e.g. 0
OUT_RANGE.UNITS_INDEX	Enter unit of the input variables (HMI)	e.g. °C
SELECT_TYPE	Enter block algorithm, here "First Good"	First Good

Control strategy:
externally switched input

Fig. 7-28 shows a control strategy example for switched input together with the links that must be made between the blocks. Here, a process is fed by two tanks which are used alternatively, i.e. first one is emptied then the other. The temperature of the fluid in each tank is to be used in a feedforward control loop. The stream selection is made in the HMI, which uses a hybrid discrete I/O block to generate the switching signals. The outputs OUT_D1 and OUT_D2 indicate which tank is being used (= 0 when in use).

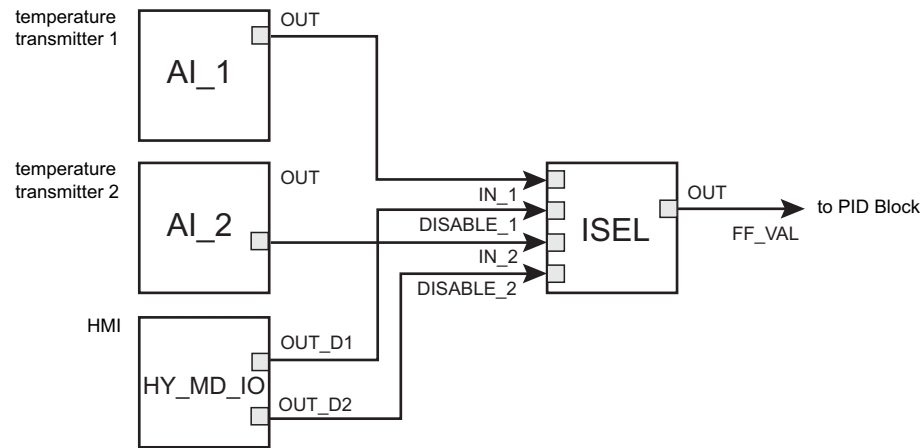


Fig. 7-28: Links to be made for Input Selector block for control strategy described above

**Block configuration:
externally switched input**

The Input Selector block is configured for the strategy in Fig 7-20 as follows:

Parameter	Action	Option/Value
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
OUT_RANGE.EU_100	Enter upper range limit for the input variables (HMI)	e.g. 200
OUT_RANGE.EU_0	Enter lower range limit for input variables (HMI)	e.g. 0
OUT_RANGE.UNITS_INDEX	Enter unit of the input variables (HMI)	e.g. °C
SELECT_TYPE	Make no selection in this parameter	Not Selected

7.8.3 Block operation**Mode**

The block normally operates in Auto mode and, depending on configuration, takes its input from one or more inputs **IN_n**.

When **MODE_BLK.Target** is set to Man, the block output can be manipulated on-line by entering the desired status and value in **OUT**. The status will be automatically set to Man if "Use Uncertain if Man" has been selected in STATUS_OPTS

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Operation can be checked on-line by viewing a number of parameters:

IN_n	Value and status of input value 1 to 4
DISABLE_n	Value and status of input disabling signal 1 to 4.
OP_SELECT	Value and status of external selection parament
OUT	Value and status of block output
SELECTED	Integer indicating which input has been selected

Status propagation

When two inputs are in use, the worst status of the two is propagated to the output. The significance of the output status is indicated below.

OUT status	Meaning
Good	Normal operation as configured
Uncertain	<ul style="list-style-type: none"> Block has an "Uncertain" input and "Use Uncertain as Good " selected in STATUS_OPTS Block in Man mode and "Use Uncertain if Man" selected in STATUS_OPTS
Bad	<ul style="list-style-type: none"> Block out of service One or both inputs have "Bad" or "Uncertain" status Number of usable inputs < MINGOOD

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Out-of-Service	<ul style="list-style-type: none"> MODE_BLK.Target currently set to OOS
Block Configuration Error	<ul style="list-style-type: none"> SELECT_TYPE parameter has an invalid value

7.8.4 Block parameters

Input Selector block

Parameter	Valid range/ Options	Default value	Description
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set MODE_BLK.Target to AUTO, CAS or RCAS
BLOCK_ERR	RO		Block errors, see Chapter 6.1.9
OUT		0	Block output value and status
OUT_RANGE		0	Block output range and units for HMI
GRANT_DENY		0	Access options, see Chapter 2.8.1
STATUS_OPTS		0	Status options, see Chapter 2.8.4 and Chapter 7.6.1
PV_FTIME		0 s	Time constant (s) of a single exponential filter for process value
IN_1	RO		Value and status of input 1
IN_2	RO		Value and status of input 2
IN_3	RO		Value and status of input 3
IN_4	RO		Value and status of input 4
DISABLE_1	RO		Value and status of discrete disable signal for input 1
DISABLE_2	RO		Value and status of discrete disable signal for input 2
DISABLE_3	RO		Value and status of discrete disable signal for input 3
DISABLE_4	RO		Value and status of discrete disable signal for input 4
SELECT_TYPE	0 – 5		Selection parameter for block algorithm <ul style="list-style-type: none"> ■ 0: No Selection ■ 1: First Good ■ 2: Minimum ■ 3: Maximum ■ 4: Middle ■ 5: Average
MIN_GOOD			Minimum number of good outputs allowed by user for selected algorithm
OP_SELECT			Value and status of external selection parameter for block algorithm: – When >0, this parameter overrides SELECT_TYPE
UPDATE_EVT			Alert generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
Legend: RO = Read Only			

7.9 Setpoint Ramp Generator (SPG)

The Setpoint Ramp Generator block generates an output value that changes with time. It is normally used to generate a setpoint for a PID block in applications such as temperature control, batch reactors, etc., where controlled temperature ramps are required during the process.

The block provides a “guaranteed soak” function that stops the timer when the deviation between the generated setpoint and an external variable is larger than a configured value. When operating in manual mode, the operator can pause the timer, select one point on the profile, advance or return the position in time. The first point of the profile can be balanced to the BKCAL_OUT value of a control block.

In ControlCare Application Designer the Setpoint Ramp Generator block (SPG Block) is assigned the generic name **Device Tag-SPG-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 7-29.

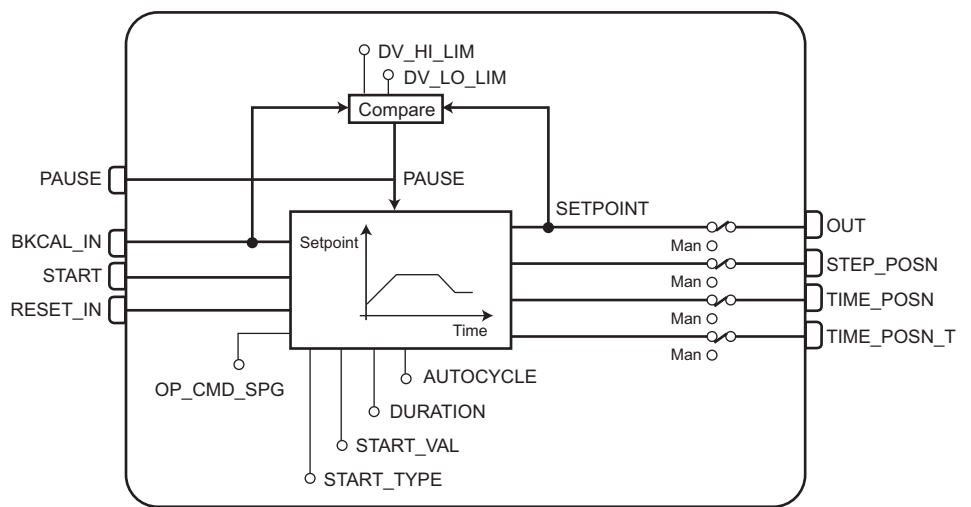


Fig. 7-29: Schematic diagram of Setpoint Ramp Generator block

7.9.1 Functional description

Ramp generator

The core element of the Setpoint Ramp Generator block is the ramp generator, which maps the desired the setpoint movement. Up to ten periods can be defined, each characterized by:

- a start value, **START_VAL [x]**,
- an end value **START_VAL [x+1]** that is also the starting value of the next time period
- and a time duration, **TIME_DURATION [x]** – set to zero for the true "end value"

The parameter **TIME_UNITS** specifies the time units for display. Fig. 7-30 illustrates the concept:

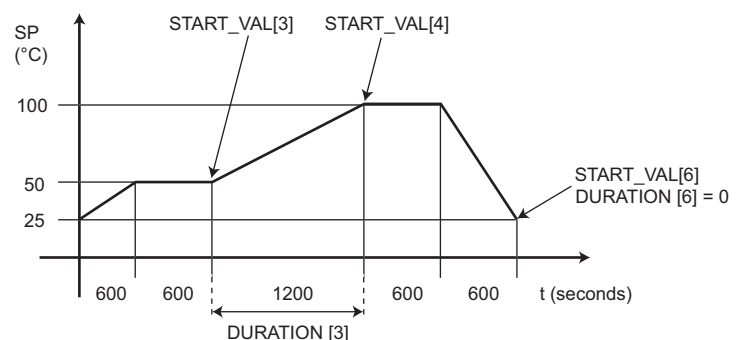


Fig. 7-30: Example of setpoint ramp generation based on five time periods

Timer

The position in the time axis of the curve defining the setpoint ramp is determined by an internal timer. The timer is started automatically by a transition from false to true at the **START** discrete input or manually by the operator by setting Status true in the **START** parameter. The timer normally runs for the sum of the **TIME_DURATION** values and then stops. It must then be reset before it can be restarted.

The operator can reset the timer independent of block mode by selecting the RESET option in the **OP_CMD_SPG** parameter. The timer is set to zero, i.e. to the beginning of the ramp. The timer is restarted manually by selecting the START option in the **OP_CMD_SPG** parameter.

The timer may also be reset automatically by a discrete signal at the **RESET_IN** input. The block will remain in reset while this input parameter has value TRUE. It can be restarted only after **RESET_IN** changes from TRUE to FALSE.

When the parameter **AUTO_CYCLE** is set to "Autocycle", the timer will return to zero (RESET) and restart (START) automatically when the time reaches the last point of the profile.

Timer interruption

The timer runs only when the block mode is Auto: a change to Man stops it. When the mode returns to Auto mode, the timer will reinitialize from the point that it stopped. If **BK_CAL_IN** is connected to **BK_CAL_OUT** of a PID block and **START_TYPE** option "Cascade" is selected, the time will stop and the block be forced to IMan should cascade initialization be in progress.

The timer may be interrupted at any time by changing the discrete signal **PAUSE** from false to true. It resumes running when **PAUSE** is set to false. The block remains in Auto mode throughout the pause.

The timer will also be paused automatically when the deviation between **BKCAL_IN** and the generated setpoint exceeds the limit **DV_HI_LIM** or **DV_LO_LIM**. This allows the user to implement a temperature driven ramping, i.e. the time will pause until the temperature is within the set tolerance, see Fig- 7-31. If the deviation is out of limits, an alarm is indicated in **DV_HI_ALM** or **DV_LO_ALM** respectively and the timer is stopped. It resumes normal operation when the deviation returns to a value within the prescribed limits.

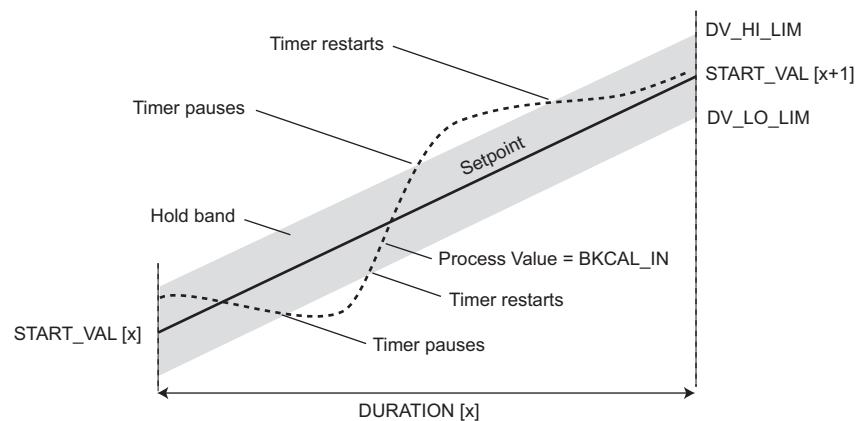


Fig. 7-31: Automatic extension of the duration when the setpoint is out of limits

Operational status

The parameter **SPG_STATE** indicates the status of setpoint ramp generator:

- **READY**: the ramp generator is initialized and waiting for the starting signal
- **ACTIVE**: the timer is running
- **PAUSE**: the PAUSE signal stopped the timer
- **DONE**: the setpoint generator has reached the last point of the ramp

The parameter **PAUSE_CAUSE** displays the cause of the PAUSE state:

- 1 = Operator Pause
- 2 = Logic Pause
- 3 = Operator & Logic
- 4 = Deviation pause
- 5 = Operator & Deviation
- 6 = Logic & Deviation
- 7 = Operator & Logic & Deviation

A logic pause may be due to the violation of deviation limits or a downstream PID block not operating in Cascade mode.

OUT value

After execution of the block, the calculated setpoint value is made available at the **OUT** output. The calculated setpoint value is also to be found in **PRE_OUT**, even when the block is in Man mode. The parameter **OUT_RANGE** is used make the range and engineering units of **OUT** known to a HMI.

Three parameters provide information on the point reached within the ramp:

- **STEP_POSN**: indicates the current segment or step
- **TIME_POSN**: indicates the time elapsed since the beginning of the current step
- **TIME_POSN_T**: indicates the time elapsed since the beginning of the profile

BKCAL_IN

Depending upon application, the input **BKCAL_IN** may be unconnected, connected to the **BKCAL_OUT** output of a downstream PID control block or connected to the **OUT** output of an upstream Analog Input block.

If there is no **BKCAL_IN** input, the timer will start at zero and the ramp will be generated as programmed. When there is an input, it is possible to start the curve from the **BKCAL_IN** value by selecting the appropriate option in the **START_TYPE** parameter. This ensures that the timer can always be started and/or the control starts without a large upset: :

START_TYPE option	BKCAL_IN connection	Action
Cascade	PID block	The time value is calculated from BKCAL_IN (= last valid OUT) using the curve. If no value can be calculated (BKCAL_IN not in curve range) the time starts at zero and BAL_TIME is used to bring the value of OUT to match PRE_OUT.
Use Duration	AI block	Time starts at zero. The START_VAL for the first segment is temporarily made equal to BKCAL_IN (= OUT from e.g. temperature transmitter)
Use Rate	AI block	The time starts at zero. The curve starts from the BKCAL_IN value (= OUT from e.g. temperature transmitter) and uses the rate specified by the first values of START_VAL and the first value of TIME_DURATION.

If a PID is connected, the **CONTROL_OPTS** parameter of the PID block should be configured to use PV for BKCAL_OUT and be set to operate in Cas mode.

Status propagation

The status of the **OUT** value is checked and handled as described below..

STATUS_OPTS	START_TYPE	BKCAL_IN status	OUT status	Remarks
—	Cascade	Good	GoodCascade	Block in Auto
—	Not "Cascade"	Good	GoodNC	Block in Auto
—	Not "Cascade"	Uncertain	Bad	Block forced to Man
Use Uncertain as Good	Not "Cascade"	Uncertain	GoodNC	Block in Auto
—	Cascade	Bad	Bad	Block forced to Man
—	Not "Cascade"	Bad	Bad	Block forced to Man
Use Uncertain as Good	Not "Cascade"	Bad	GoodNC	Block in Auto, deviation limits ignored
Target to Manual if BAD IN	Any	Bad	Bad	Block forced to Man

7.9.2 Block configuration

Control strategy:
time driven ramp

Fig. 7-32 shows a control strategy example for the time driven generation of a temperature ramp. This might be required to quickly bring an annealing furnace to its operating temperature, the temperature of the workpieces being secondary at this point. The SPG block is connected a PID control block and provides its setpoint. The temperature measuring device monitors the generated temperature. The PID block generates the control output for e.g. a heater and sends the current SP as feedback to the SPG block. In this configuration, the ramp will be executed according to the time durations set. If the control loop is broken, the SPG block will reset to the last valid SP.

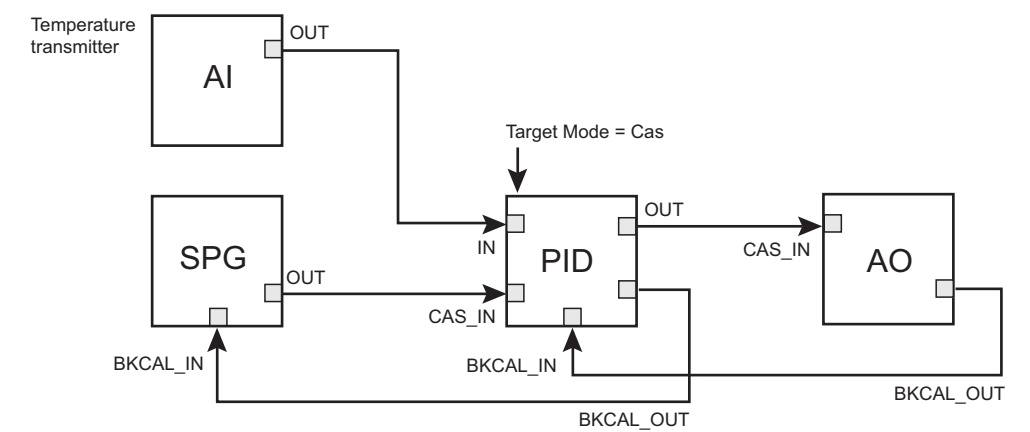


Fig. 7-32: Links to be made for Setpoint Ramp Generator block for control strategy described above

Block configuration:
time driven ramp

The Setpoint ramp generator is configured for the strategy in Fig 7-32 as follows, whereby it is assumed that the ramp has the profile in Fig. 7-30:

Index/Parameter	[1]	[2]	[3]	[4]	[5]	[6]
START_VAL	25	50	50	100	100	25
DURATION	600	600	1200	600	600	–

Parameter	Action	Option/Value
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
OUT_RANGE.EU_100	Enter upper range limit for the input variables (HMI)	e.g. 200
OUT_RANGE.EU_0	Enter lower range limit for input variables (HMI)	e.g. 0
OUT_RANGE.UNITS_INDEX	Enter unit of the input variables (HMI)	e.g. °C
START_VAL.[1] START_VAL.[2] START_VAL.[3] START_VAL.[4] START_VAL.[5] START_VAL.[6]	Enter the start values for each segment and the "end marker"	25 50 50 100 100 25 (end marker)
TIME_DURATION.[1] TIME_DURATION.[2] TIME_DURATION.[3] TIME_DURATION.[4] TIME_DURATION.[5]	Enter the duration for each segment (empty parameters are = 0 by default)	600 600 1200 600 600
TIME_UNITS	Display units for TIME_POSN and TIME_POSN_T	seconds

**Control strategy:
temperature driven ramp**

Fig. 7-33 shows a control strategy example for the temperature driven generation of a temperature ramp. This might be required, e.g. to ensure that a workpiece is held at a specific temperature for a minimum period of time, e.g. during annealing. The SPG block is connected a PID control block and provides its setpoint. The temperature measuring device monitors the generated temperature. By selecting the **CONTROL_OPTS** option "Use PV as BKCAL_OUT", the SPG receives feedback on the current temperature. This is used to set tolerances on the setpoint, and to pause the timer if the temperature leaves the hold band, see Fig. 7-31. The current setpoint will be held until the temperature returns to the hold band. In order to be able to start the counter again when the ramp cycle is completed, the **START_TYPE** option "Cascade" is selected.

Note

- Temperature feedback may also be provided by a direct connection from an AI block to **BKCAL_IN**, in which case, the **START_TYPE** option "Use Duration" or "Use Rate" must be selected. In this case, no **CONTROL_OPTS** options are required in the PID block

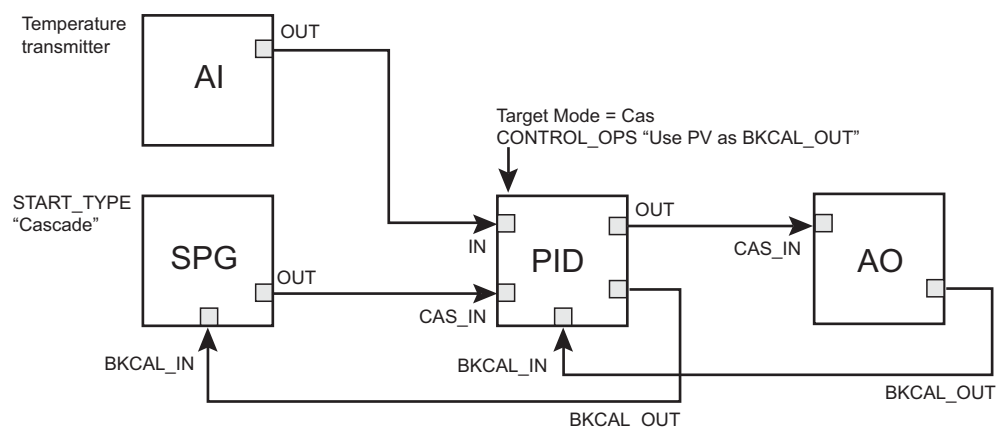


Fig. 7-33: Links to be made for Setpoint Ramp Generator block for control strategy described above

**Block configuration:
temperature driven ramp**

The Setpoint Ramp Generator block is configured for the strategy in Fig 7-33 as follows:

Parameter	Action	Option/Value
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
OUT_RANGE.EU_100	Enter upper range limit for the input variables (HMI)	e.g. 200
OUT_RANGE.EU_0	Enter lower range limit for input variables (HMI)	e.g. 0
OUT_RANGE.UNITS_INDEX	Enter unit of the input variables (HMI)	e.g. °C
START_VAL.[1]	Enter the start values for each segment and the "end marker"	25
START_VAL.[2]		50
START_VAL.[3]		50
START_VAL.[4]		100
START_VAL.[5]		100
START_VAL.[6]		25 (end marker)
TIME_DURATION.[1]	Enter the duration for each segment (empty parameters are = 0 by default)	600
TIME_DURATION.[2]		600
TIME_DURATION.[3]		1200
TIME_DURATION.[4]		600
TIME_DURATION.[5]		600
TIME_UNITS	Display units for TIME_POSN and TIME_POSN_T	seconds
START_TYPE	Determines the start value for a new rampo cycle	Cascade
BAL_TIME	Sets time to balance the OUT value to the PRE_OUT value on cascade initialization	60
ALARM_HYS	Alarm hysteresis in % of OUT_RANGE	1
DV_HI_LIM	Max. deviation between setpoint and measured value	2 (°C)
DV_LO_LIM	Max. deviation between setpoint and measured value	2 (°C)
PID block		
MODE_BLK.Target	Set the target mode of the block to Cas	Cas
CONTROL_OPTS	Select option	Use PV as BKCAL_OUT

7.9.3 Block operation

Mode

The block normally operates in Auto mode.

When **MODE_BLK.Target** is set to Man, the block output can be manipulated on-line. The operator can write on the outputs **STEP_POSN**, **TIME_POSN** and **TIME_POSN_T** in order to select a particular point of the profile. When the block is switched back to Auto, the ramp will start from that point. The timer is restarted by activating the input **START**.

The operator can also write to **OUT**. As the adjusted value may correspond to more than one point on the profile or to none, if the operator adjusts a value beyond the profile limits, the **OUT** value goes from the last adjusted value to the point before mode switching following a ramp defined by **BAL_TIME**.

it is also possible to advance or return the time through the following operator commands (OP_CMD_SPG):

- **ADVANCE**: sets the time to the beginning of the next step
- **REPEAT**: sets the time to the beginning of the current step

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Operation can be checked on-line by viewing a number of parameters:

START	Value and status of start input signal
RESET_IN	Value and status of reset input signal.
PAUSE_CAUSE	Reason for timer pause
SPG_STATE	Current state of the Sepoint Ramp Generator
STEP_POSN	Current position in ramp generator profile
TIME_POSN	Time elapsed since beginning of ramp step
TIME_POS_T	Time elapsed since beginning of ramp cycle
OUT	Value and status of block output

Status

When two inputs are in use, the worst status of the two is propagated to the output. The significance of the output status is indicated below.

OUT status	Meaning
Good	Normal operation as configured
Bad	<ul style="list-style-type: none"> ■ Block out of service ■ One or both inputs have "Bad" or "Uncertain" status

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Out-of-Service	<ul style="list-style-type: none"> ■ MODE_BLK.Target currently set to OOS
Block Configuration Error	<ul style="list-style-type: none"> ■ START_TYPE parameter has an invalid value

7.9.4 Block parameters

Setpoint Ramp Generator block

Parameter	Valid range/ Options	Default value	Description
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set MODE_BLK.Target to AUTO, CAS or RCAS
BLOCK_ERR	RO		Block errors, see Chapter 6.1.9
OUT			Output value calculated as a result of executing the block.
OUT_RANGE		0	Block output range and units for HMI
GRANT_DENY		0	Access options, see Chapter 2.8.1

Parameter	Valid range/ Options	Default value	Description
STATUS_OPTS		0	Status options, see Chapter 2.8.4 and Chapter 6.1.9
START_VAL			Up to 11 values defining the start value of each time period ■ If ten time periods are in use, an eleventh value must be entered as end marker
TIME_DURATION		0	Up to 10 values defining the duration in seconds of each time period ■ If less than ten time periods are in use, the "end value" must be paired with "0"
TIME_UNITS		0	Display units for TIME_POSN and TIME_POSN_T
BKCAL_IN			Feedback from BKCAL_OUT of a downstream block or OUT value of a Analog Input block
START			Value and status of start input, if connected ■ In Man, a change from false to true starts the timer, provided it is READY
START_TYPE			Initialization type for ramp start when BKCAL_IN connected
PAUSE			Value and status of pause input, if connected
PAUSE_CAUSE			Reason for current pause, see Chapter 7.9.1
AUTO_CYCLE			When set, causes ramp to automatically reset and start
STEP_POSN			Displays current position in ramp generator profile ■ In Man, adjusts ramp generator to entered position
TIME_POSN			Displays time elapsed since beginning of step ■ In Man, adjusts time to entered value
TIME_POS_T			Displays time elapsed since beginning of ramp cycle ■ In Man, adjusts time to entered value
OP_CMD_SPG			Operator commands for setpoint ramp generator ■ ADVANCE: advances to next step ■ No action ■ REPEAT: repeats current step ■ RESET: resets setpoint ramp generator
SPG_STATE			Displays the current state of the Sepoint Ramp Generator ■ READY: the ramp generator is initialized and waiting for the starting signal ■ ACTIVE: the timer is running ■ PAUSE: the PAUSE signal stopped the timer ■ DONE: the setpoint generator has reached the last point of the ramp
PRE_OUT			Displays the OUT value resulting from block execution
RESET_IN			Value and status of reset input, if connected
BAL_TIME	Positive	0	Time allowed to balance OUT changes as a result of a change from Auto, Cas or Rcas to Man
OUTAGE_LIM			Not used
UPDATE_EVT			This alert is generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
ALARM_SUM			Alarm summary, see Chapter 2.9
ACK_OPTION	0, 1	0	See Chapter 2.9 ■ 0: Auto ACK Disable ■ 1: Auto ACK Enable
ALARM_HYS	0 to 50 %	0.5%	Alarm hysteresis in % of OUT_SCALE, see Chapter 2.9
DV_HI_PRI	0 to 15	0	Priority of the control deviation HI alarm, see Chapter 2.9
DV_HI_LIM		+INF	Setting of control deviation HI in engineering units
DV_LO_PRI	0 to 15	0	Priority of the control deviation LO alarm, see Chapter 2.9
DV_LO_LIM		-INF	Setting of control deviation LO in engineering units
DV_HI_ALM			Status of deviation HI alarm and its associated time stamp.
DV_LO_ALM			Status for deviation LO alarm and its associated time stamp.
Legend: RO = Read Only			

7.10 Enhanced Setpoint Ramp Generator

The Enhanced Setpoint Ramp Generator has all parameters of the SPG block, and an additional output parameter, OUT_1, that indicates the current step or segment of profile. This is the same information as is carried by the parameter **STEP_POSN**, but value is transmitted in floating point format.

In ControlCare Application Designer the Setpoint Ramp Generator block is assigned the generic name **Device Tag-ESPG-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 7-34.

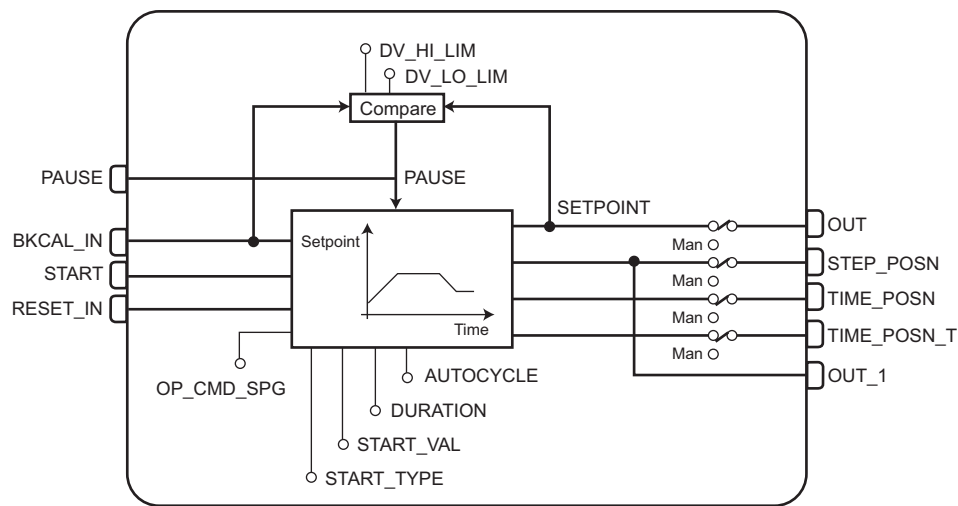


Fig. 7-34: Schematic diagram of Enhanced Setpoint Ramp Generator block

Block parameters

Parameter	Valid range/ Options	Default value	Description
OUT_1	0–4	RO	Displays current position in ramp generator profile in floating point format <ul style="list-style-type: none"> ■ 0=None ■ 1= Step1 ■ 2= Step2 ■ 3= Step 3 ■ 4= Step 4
Legend: RO = Read Only			

7.11 Timer and Logic

The Timer and Logic function block provides logic combination and timing functions, e.g.:

- Combine multiple inputs as OR, AND, vote, or EXACTLY count.
- Measure the duration of a combined discrete input signal
- Accumulate, until reset, the duration of a combined input signal
- Count changes of a combined discrete input signal
- Set a discrete output if the duration of a combined input signal exceeds a limit
- Extend, Delay, Pulse, or Debounce a combined input as an output
- Provide outputs indicating amount of time expired and amount of time remaining
- Selectively invert any connected discrete input or output
- Reset timer

In ControlCare Application Designer the Timer and Logic block (TMR Block) is assigned the generic name **Device Tag-TMR-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 7-35.

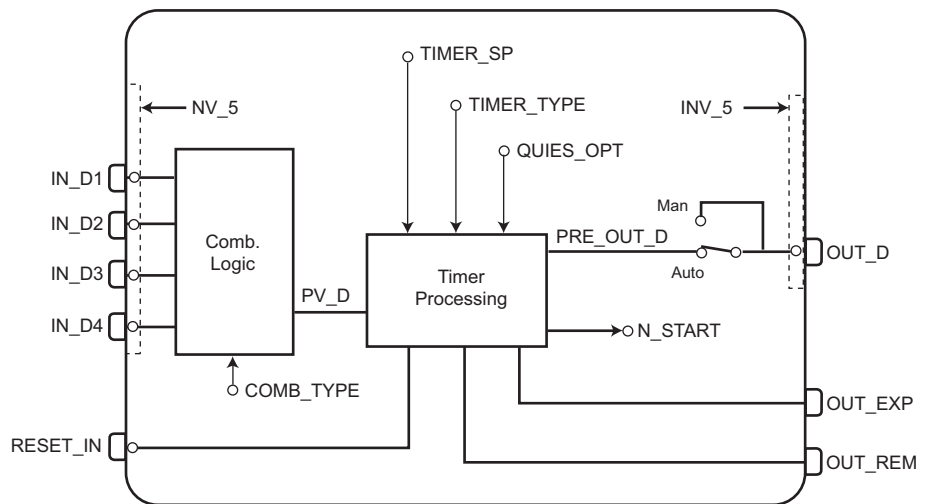


Fig. 7-35: Schematic diagram of Timer and Logic block

7.11.1 Functional description

Input combination

Depending upon the option set in **COMB_TYPE**, see table below, up to four inputs may be

- logically combined (AND, OR),
- voted (any 2 or more true, any 3 or more true),
- or counted (exactly 1 true, exactly 2 true, exactly 3 true, odd count or even count).

Connected inputs may have the values of true (1) or false (0). If a connected input is undefined, it is treated as "Bad" (Out-of service). Non-connected inputs (operator entries) may have the values of true (1) or false (0). If a non-connected input is undefined, it is ignored.

COMB_TYPE	PV_D value
OR	true if one or more used inputs are true
ANY2	true if two or more used inputs are true
ANY3	true if three or more used inputs are true
AND	true if all used inputs are true
EXACTLY1	true if exactly 1 used input is true
EXACTLY2	true if exactly 2 used inputs are true
EXACTLY3	true if exactly 3 used inputs are true
EVEN	true if exactly 0, 2 or 4 used inputs are true (OR non exclusive)
ODD	true if exactly 1 or 3 used inputs are true (OR exclusive)

Timer Processing

The timer processing type is specified by **TIMER_TYPE**, see below. The timer may operate to produce a measurement, delay, extension, pulse (non-re-triggerable or re-triggerable) or debounce of the combined input signal.

TIMER_SP specifies the time duration of the delay, extension, pulse, debounce filter or comparison limit. The result of block execution is **PRE_OUT_D** which is made available at **OUT_D** when the block is in Auto mode. **OUT_EXP** indicates the time elapsed and **OUT_REM** indicates the remaining time when **TIME_TYPE** option is measurement, comparison, delay, extension, debounce, or pulse.

QUIES_OPT allows the user to select the behavior for **OUT_EXP** and **OUT_REM** when the timer is quiescent— that is, the timer is stopped, waiting to start. The following table indicates the definition of quiescent state for each option of **TIMER_TYPE**:

Definition of quiescent state start and end as a function of TIMER_TYPE		
TIMER_TYPE	Quiescence state starts when combined input (PV_D):	Quiescence state ends when combined input (PV_D):
MEASURE	returns to false	changes from false-to-true
ACCUM	[QUIES_OPT does not apply]	[QUIES_OPT does not apply]
COMPARE	returns to false	changes from false-to-true
DELAY	returns to false	changes from false-to-true
EXTEND	returns to true	changes from true-to-false
DEBOUNCE	has changed and timer has expired	changes
PULSE	as returned to false and timer has expired	changes from false-to-true
RT_PULSE	has returned to false and timer has expired	changes from false-to-true

QUIES_OPT has two options, CLEAR and LAST:

- CLEAR causes both **OUT_EXP** and **OUT_REM** to be set to zero during quiescence.
- LAST causes both **OUT_EXP** and **OUT_REM** to hold their last values when the block becomes quiescent. That is, the time expired and time remaining will remain available until the quiescence ends with the start of the next activation.

Note that a false-to-true transition on **RESET_IN** will also reset **OUT_EXP** and **OUT_REM**.

N_START is a count of the number of starts (false-to-true) transitions of the combined input, **PV_D**, since the last false-to-true change seen on **RESET_IN**.

Timer type

TIMER_TYPE may be one of the following, operating on the combined input signal:

TIMER_TYPE	Quiescence state starts when combined input (PV_D):
MEASURE	Indicates the duration of the most recent true signal
ACCUM	Accumulates the durations of a true signal
COMPARE	Compares a true signal duration to specified duration
DELAY	Delays a false-to-true transition, eliminating it if short
EXTEND	Extends a true-to-false transition, eliminating it if short
DEBOUNCE	Delays any transition, eliminating it if short
PULSE	Generates a true pulse on a false-to-true transition, non-retriggerable
RT_PULSE	Generates a true pulse on a false-to-true transition, retriggerable

MEASURE

If **TIMER_TYPE** is MEASURE, **PRE_OUT_D** will be the same as the combined input, **PV_D**. **OUT_EXP** indicates the length of time, in seconds, that the combined signal is true. **OUT_REM** is set to 0. Fig. 7-36 shows an example, whereby **QUIES_OPT** has been set to CLEAR. If **QUIES_OPT** were set to LAST, **OUT_EXT** would accumulate and not return to zero, see "ACCUM".

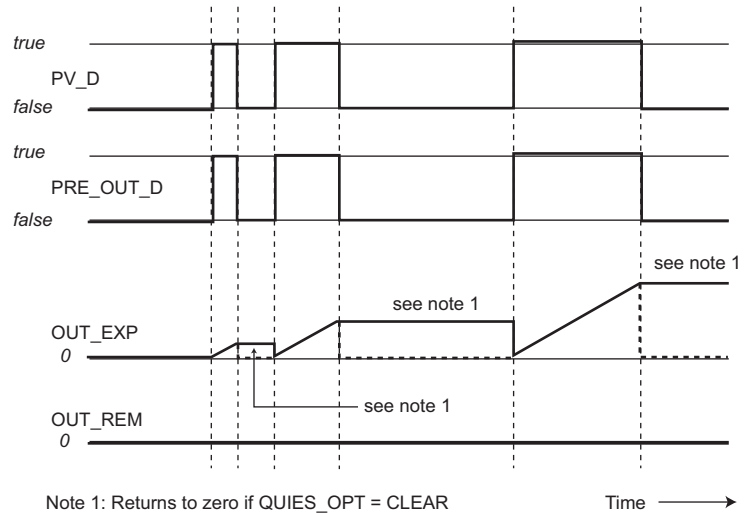


Fig. 7-36: Timer example for **TIMER_TYPE** = MEASURE

ACCUM

If **TIMER_TYPE** is ACCUM, **PRE_OUT_D** will be the same as the combined input, **PV_D**. **OUT_EXP** indicates the accumulated length of time, in seconds, that the combined signal has been true. Unlike **TIMER_TYPE** = MEASURE, it will not be automatically reset by the time of the next occurrence of a false-to-true change of **PV_D**. Instead, it will continue to accumulate "on" time or "run" time until reset to 0 by a false-to-true change on **RESET_IN**. **OUT_REM** is unused (set to 0.0) for this timer type. **QUIES_OPT** is disabled for this option.

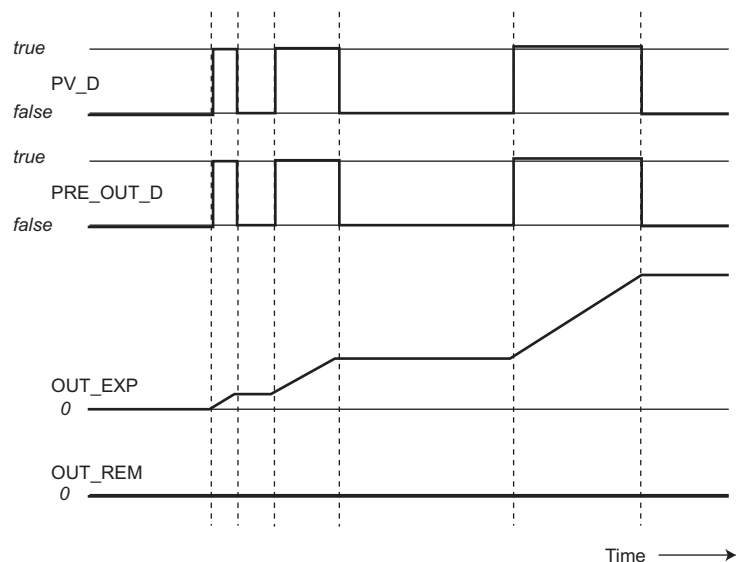


Fig. 7-37: Timer example for **TIMER_TYPE** = ACCUM

COMPARE

If **TIMER_TYPE** is COMPARE, the block will measure the time since a false-to-true change on the combined input, **PV_D**. The current duration will be indicated by **OUT_EXP**. **OUT_REM** will indicate the time remaining between the current expired duration, **OUT_EXP**, and current limit, **TIMER_SP**. If **OUT_EXP** does not exceed **TIMER_SP**, **PRE_OUT_D** will be set to false. If **OUT_EXP** equals or exceeds **TIMER_SP**, **PRE_OUT_D** will be set to true and **OUT_REM** will be set to zero. When the combined input returns to false, either with or without exceeding the limits specified by **TIMER_SP**, **OUT_D** will be set to false. (Note that this type of behavior is the same as **TIMER_TYPE** = DELAY. The difference is merely in the application perspective).

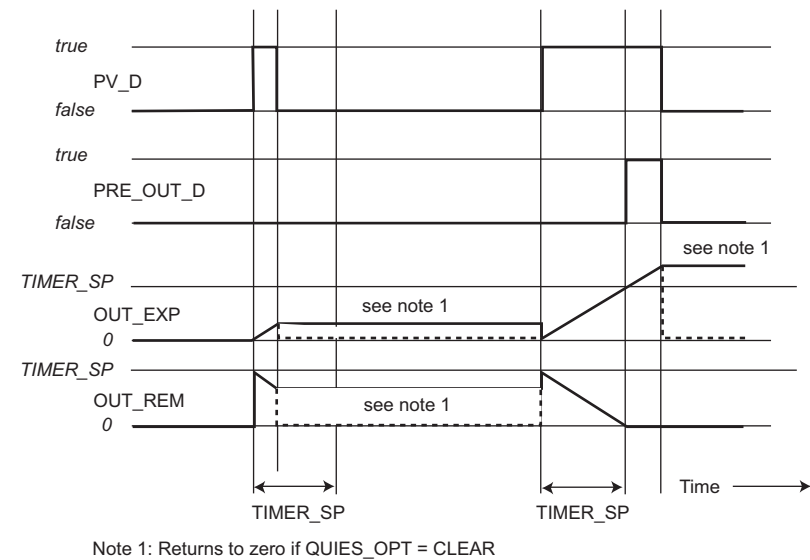


Fig. 7-38: Timer example for *TIMER_TYPE* = COMPARE

DELAY

If **TIMER_TYPE** is DELAY, a false-to-true change on the combined input, **PV_D**, will be delayed at the output, **PRE_OUT_D**, until the amount of time specified by **TIMER_SP** has been expired. If the combined input returns to false before the time expires, the output will remain as false, concealing the input transitions. If the **PRE_OUT_D** output has been set to true because the time has expired, a true-to-false transition in the combined input will be presented to **PRE_OUT_D** immediately. [Note that this type of behavior is the same as **TIMER_TYPE** = COMPARE. The difference is merely in the application perspective.]

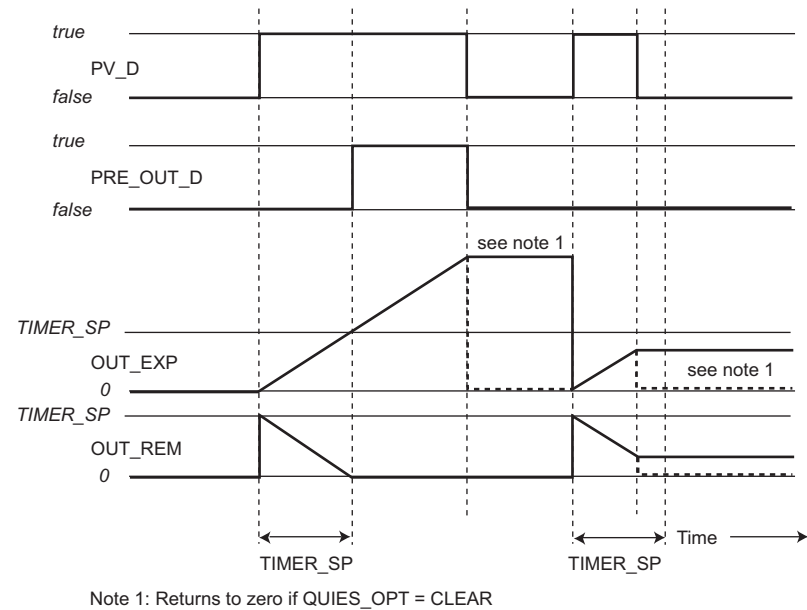
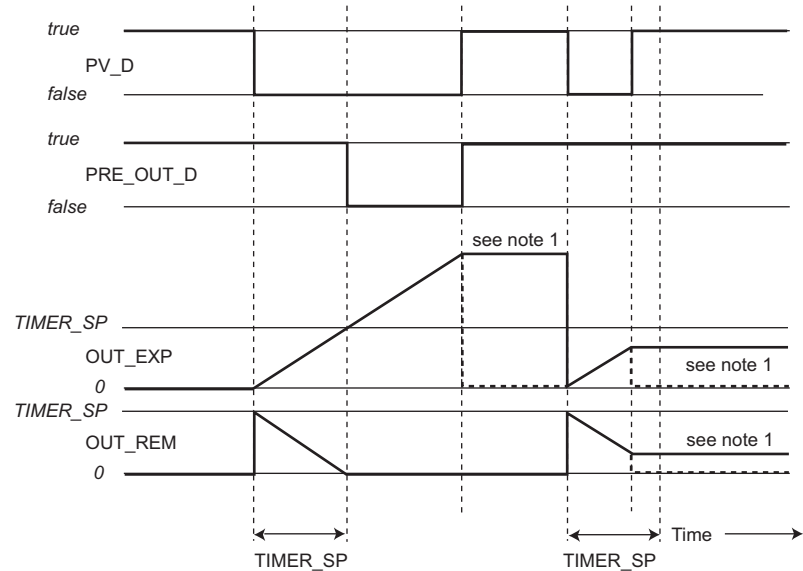


Fig. 7-39: Timer example for *TIMER_TYPE* = DELAY

EXTEND

If **TIMER_TYPE** is **EXTEND**, a true-to-false change on the combined input, **PV_D**, will be delayed at the output, **PRE_OUT_D**, until the amount of time specified by **TIMER_SP** has expired. If the combined input returns to true before the time expires, the output will remain as true, concealing the input transitions. If the **PRE_OUT_D** output has been set to false because the time has expired, a false-to-true transition in the combined input will be presented to **PRE_OUT_D** immediately.

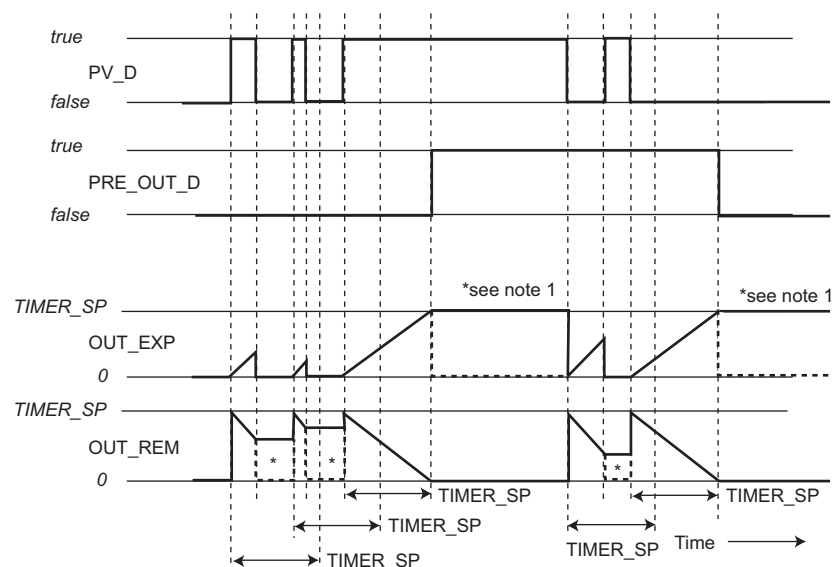


Note 1: Returns to zero if QUIES_OPT = CLEAR

Fig. 7-40: Timer example for **TIMER_TYPE** = **EXTEND**

DEBOUNCE

If **TIMER_TYPE** is **DEBOUNCE**, and if **PRE_OUT_D** is false, a false-to-true change on the combined input, **PV_D**, will be delayed at the output, **PRE_OUT_D**, until the amount of time specified by **TIMER_SP** has expired. If the combined input returns to false before the time expires, the output will remain as false, concealing the input transitions. If **PRE_OUT_D** is true, a true-to-false change on the combined input, **PV_D**, will be delayed at the output, **PRE_OUT_D**, until the amount of time specified by **TIMER_SP** has expired. If the combined input returns to true before the time expires, the output will remain as true, concealing the input transitions. This both delays true initiations and extends true terminations, acting as a filter for intermittent state changes.



*Note 1: Returns to zero if QUIES_OPT = CLEAR

Fig. 7-41: Timer example for **TIMER_TYPE** = **DEBOUNCE**

PULSE

If **TIMER_TYPE** is PULSE, a false-to-true change on the combined input, **PV_D**, will initiate a true pulse at **PRE_OUT_D** whose duration is determined by the **TIMER_SP** value. At the end of the time duration, the output, will return to false. Further false-to-true transitions of the combined input while **PRE_OUT_D** is true will be ignored.

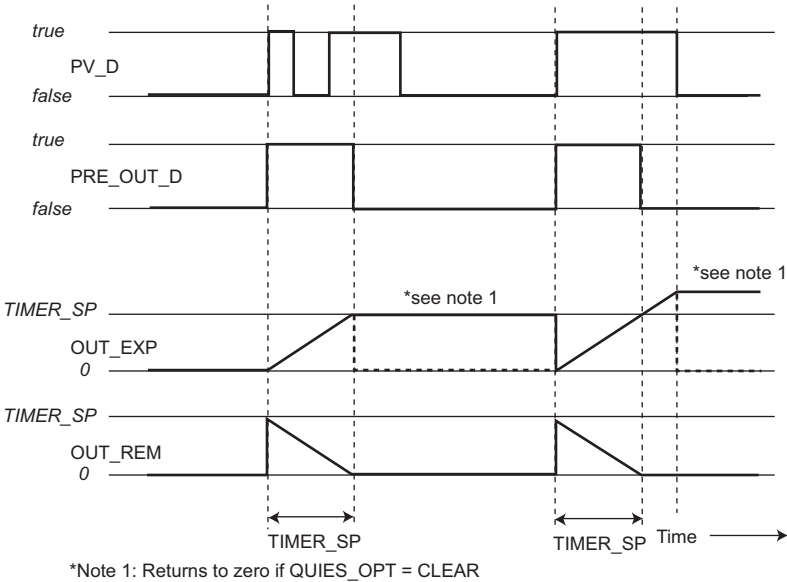


Fig. 7-42: Timer example for *TIMER_TYPE* = PULSE

RT_PULSE

If **TIMER_TYPE** is RT_PULSE, (Re-Triggerable pulse type) a false-to-true change on the combined input, **PV_D**, will initiate a true pulse at **PRE_OUT_D** whose duration is determined by the **TIMER_SP** value. At the end of that time duration **PRE_OUT_D** will return to false. If the combined input returns to false and presents a subsequent false-to-true transition while the timer is timing, the timer will be reinitialized and **PRE_OUT_D** continues to be true.

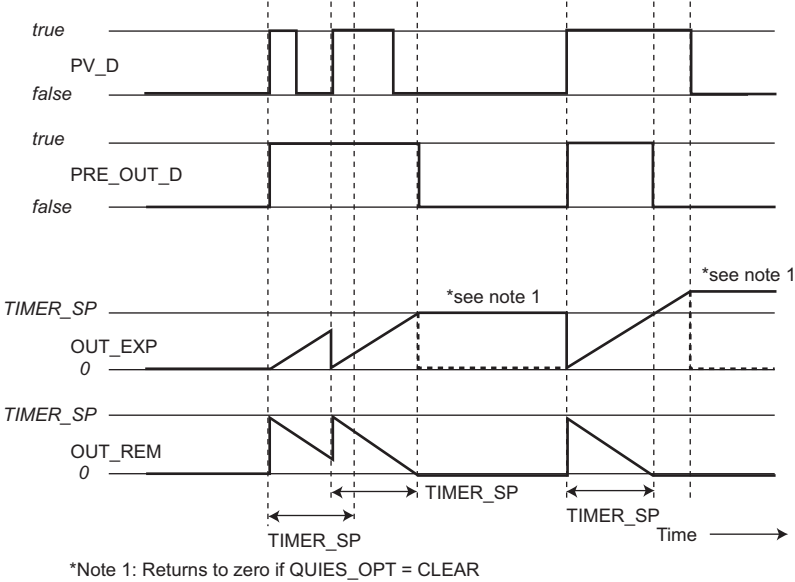


Fig. 7-43: Timer example for *TIMER_TYPE* = RT_PULSE

Timer reset

RESET_IN is a discrete input which, on a false-to-true transition, resets the timer. **OUT_EXP** output is set to 0, and then the timer follows processing described in "Initialization Treatment" for the values **PRE_OUT_D** and **OUT_REM**.

INVERT_OPTS allows the discrete input or output values to be inverted. Each input or output can be set individually.

Status propagation

The block checks the status of the inputs before processing them. The table below summarizes the status of **OUT** under various conditions. ..

STATUS_OPTS	IN_Dx status	OUT status	Remarks
–	Good	Good	Block executed
–	Uncertain	Bad	Block not executed, timer stops
Use Uncertain as Good	Uncertain	Good	Block executed
–	Bad	Bad	Block not executed, timer stops

When the outputs return to usable status, the timer returns to the measurement and the **OUT_EXP** and **OUT_REM** status is set to Uncertain while they are in quiescent state or a Reset occurs.

Initialization Treatment

The following table summarizes the values of **PRE_OUT_D**, **OUT_EXP** and **OUT_REM** after the initial execution, as a function of **TIMER_TYPE** and the initial value of the combined input, **PV_D**:

TIMER_TYPE	PV_D	PRE_OUT_D	OUT_EXP	OUT_REM	Timer Status
MEASURE	False	False	0.0	0.0	Inactive
MEASURE	True	True	0.0	0.0	Inactive
ACCUM	False	False	0.0	0.0	Inactive
ACCUM	True	True	0.0	0.0	Inactive
COMPARE	False	False	TIMER_SP*	0.0	Inactive
COMPARE	True	False	0.0	TIMER_SP*	Active
DELAY	False	False	TIMER_SP*	0.0	Inactive
DELAY	True	False	0.0	TIMER_SP*	Active
EXTEND	False	True	0.0	TIMER_SP*	Active
EXTEND	True	True	TIMER_SP*	0.0	Inactive
DEBOUNCE	False	False	TIMER_SP*	0.0	Inactive
DEBOUNCE	True	True	TIMER_SP*	0.0	Inactive
PULSE	False	False	0.0	0.0	Inactive
PULSE	True	False	TIMER_SP*	0.0	Inactive
RT_PULSE	False	False	0.0	0.0	Inactive
RT_PULSE	True	False	TIMER_SP*	0.0	Inactive

* Initialize to TIMER_SP value if QUIES_OPT = LAST, initialize to 0.0 if QUIES_OPT = CLEAR.

7.11.2 Block configuration

Control strategy:
safety interlock

Fig. 7-44 shows a control strategy example for the use of the TMR block as a safety interlock. A motor can be started manually only when a key switch in the controlroom is turned on and a local pushbutton is latched. The motor is to run as long as the button is latched. The inputs are connected to a relay input module and made available to the control strategy via a multiple digital input block. The AND function of the timer block is used to check the true condition of both inputs and the MEASURE function is used to give a defined signal to a multiple discrete output block, to which the motor is connected via a output relay module.

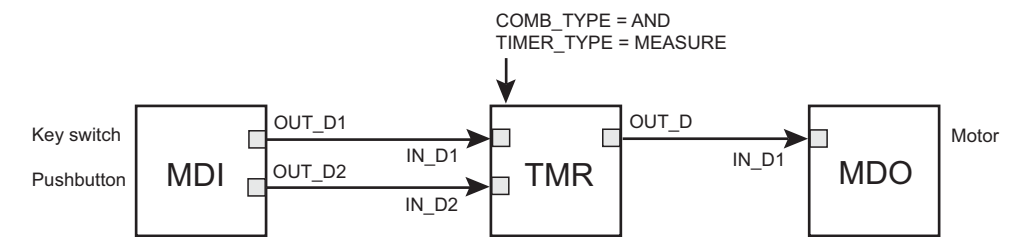


Fig. 7-44: Links to be made for Timer and Logic block for control strategy described above

Block configuration:
safety interlock

The Timer and Logic block is configured for the strategy in Fig 7-44 as follows:

Parameter	Action	Option/Value
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
COMB_TYPE	Logic to be used by block	AND
TIMER_TYPE	Timer to be used by block	MEASURE

Control strategy:
timed sequence

Fig. 7-45 shows a control strategy example for the use of the TMR block for a timed output. One of two silos is used to feed solid material to a reactor filled with water. After feeding is completed, the mixture must be stirred for a period of fifteen minutes. The signals used to close the silo valves, fed in via a MDI block, are used to start the timer, the OR function to detect the closing of either valve. The PULSE function ensures a motor signal of duration 15 minutes is generated.

Note



- In this application, the outputs OUT_EXP and OUT_REM provide an analog timer output that may be used, e.g. in a hybrid function block, to time logic sequences.

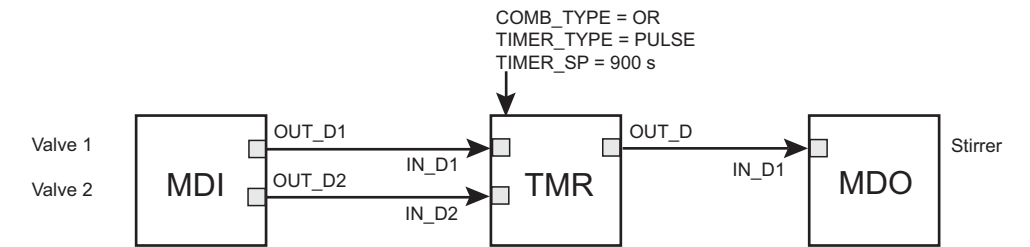


Fig. 7-45: Links to be made for Timer and Logic block for control strategy described above

Block configuration:
timed sequence

The Timer and Logic block is configured for the strategy in Fig 7-44 as follows:

Parameter	Action	Option/Value
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
TIMER_SP	Length of pulse, in seconds, generated after detection of closed valve	900
COMB_TYPE	Logic to be used by block	OR
TIMER_TYPE	Timer to be used by block	PULSE

7.11.3 Block operation

Mode

The block normally operates in Auto mode, whereby **PRE_OUT_D** is passed to the **OUT_D** output.

When **MODE_BLK.Target** is set to Man, the block output can be manipulated on-line. The operator can write on the output **OUT_D** outputs. The calculations continue and can be read in **PRE_OUT_D**.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Operation can be checked on-line by viewing a number of parameters:

IN_Dx	Value and status of discrete input signal 1 to 4
RESET_IN	Value and status of reset input signal.
PRE_OUT_D	Result of block execution, irrespective of operating mode
OUT_D	Value and status of discrete output signal
OUT_EXP	Time expired sing timer started
OUT_REM	Time remaining until timer stops

Status

When two inputs are in use, the worst status of the two is propagated to the output. The significance of the output status is indicated below.

OUT status	Meaning
Good	<ul style="list-style-type: none"> Normal operation as configured
Bad	<ul style="list-style-type: none"> Block out of service One or more inputs have "Uncertain" status and STATUS_OPTS "Use Uncertain as Good" is not set One or more input have "Bad" status

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Out-of-Service	<ul style="list-style-type: none"> MODE_BLK.Target currently set to OOS
Block Configuration Error	<ul style="list-style-type: none"> TIME_UNITS or QUIES_OPT parameter has an invalid value

7.11.4 Block parameters

Timer block

Parameter	Valid range/ Options	Default value	Description
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set MODE_BLK.Target to AUTO, CAS or RCAS
BLOCK_ERR	RO		Block errors, see Chapter 6.1.9
PV_D			Either the primary discrete value for use in executing the function, or a process value associated with it
OUT_D			Primary discrete value calculated as a result of executing the function
TIMER_SP		0	Time used by the TMR block to set the delay, extend, debouncing and processing the pulse time.
PV_STATE		0	Index to the text describing the states of a discrete PV.
OUT_STATE		0	Index to the text describing the states of a discrete output.
GRANT_DENY		0	Access options, see Chapter 2.8.1
INVERT_OPTS		0	Signal inversion options
STATUS_OPTS		0	Status options, see Chapter 2.8.4 and Chapter 6.1.9
IN_D1			Value and status of discrete input 1
IN_D2			Value and status of discrete input 2

Parameter	Valid range/ Options	Default value	Description
IN_D3			Value and status of discrete input 3
IN_D4			Value and status of discrete input 4
COMB_TYPE		Or	Determines how the inputs IN_D[i] are combined <ul style="list-style-type: none"> ■ AND ■ OR ■ ANY2 ■ ANY3 ■ EXACTLY1 ■ EXACTLY2 ■ EXACTLY3 ■ EVEN ■ ODD
TIMER_TYPE		Measure	Type of processing applied to PV_D to determine PRE_OUT_D. <ul style="list-style-type: none"> ■ MEASURE ■ ACCUM ■ COMPARE ■ DELAY ■ EXTEND ■ DEBOUNCE ■ PULSE ■ RT_PULSE
PRE_OUT_D		RO	Combined and time-processed output of the timer block
N_START		RO	Count of false-to-true transitions of the combined input, PV_D. – Reset by false-to-true transition of RESET_IN.
OUT_EXP		RO	Time expired. – Stops when TIMER_SP is reached. – Reset to zero (1) by RESET_IN, (2) at start of next timer event if QUIES_OPT = LAST, or (3) when block becomes quiescent if QUIES_OPT = CLEAR.
OUT_REM		RO	Time remaining if the timer is active. – Stops when event ceases (block becomes quiescent). – Reset to 0.0 if QUIES_OPT = CLEAR, and the timer is inactive.
RESET_IN			Resets the timer
QUIES_OPTS			Behavior option for OUT_EXP and OUT_REM during quiescence. <ul style="list-style-type: none"> ■ CLEAR resets them to zero. ■ LAST causes last values to be held
TIME_UNITS			Not used: This parameter has fixed unit: seconds
UPDATE_EVT			This alert is generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
ALARM_SUM			Alarm summary, see Chapter 2.9
Legend: RO = Read Only			

7.12 Lead-Lag

The Lead-Lag block provides dynamic compensation of an input signal. Its primary use is in feedforward control when the dynamic response of the manipulated variable differs significantly from that of a disturbance variable, e.g. as would be the case for flow and temperature measurement. Depending upon configuration, the block can function as a lead, lag, or both.

The block may also be used to implement some special initialization functions required by a control scheme.

In ControlCare Application Designer the Timer and Logic block (LL Block) is assigned the generic name **Device Tag-LL-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 7-46.

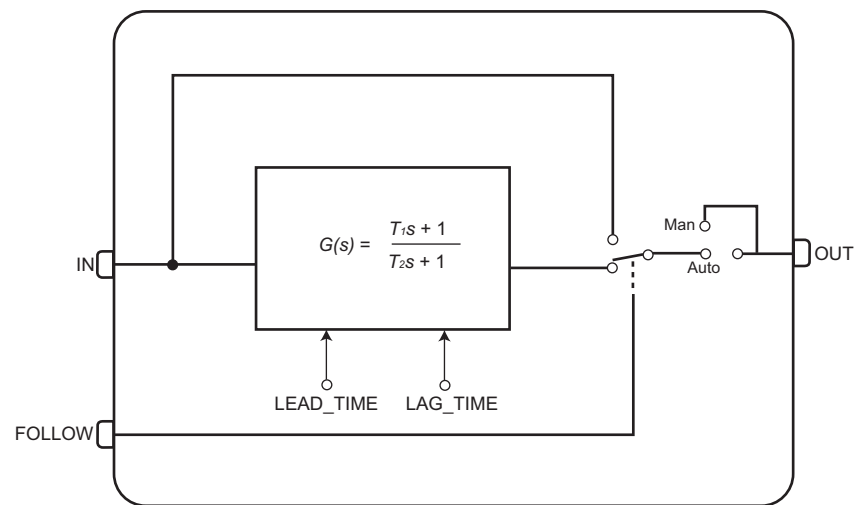


Fig. 7-46: Schematic diagram of Lead-Lag block

7.12.1 Functional description

The LEAD_TIME and LAG_TIME parameters are configured to obtain the desired input/output relationship.

- **LAG_TIME** specifies the time constant for the block. Based on a step change to the input this is the time to reach 63.2% of the final value. As a rule of thumb, it requires five time constants to reach the final value based on a first order function applied to the input.
- **LEAD_TIME** specifies the gain or impulse applied to the input parameter.
- **FOLLOW** causes the block to perform tracking functions.
When **FOLLOW** is true, it forces **OUT** to track **IN**.

The generalized form of the equation describing the action is as follows:

$$G(s) = \frac{T_1s + 1}{T_2s + 1}$$

Where: $G(s)$ = Gain
 T_1 = Lead time constant
 T_2 = Lag time constant

Application examples

In the following applications, the initial input signal is $IN = 10$. This input receives a positive step change equal to 10% at a time of $t = 5s$. At $t = 20s$, the input receives a negative step change equal to 10%. Depending upon the lead lag settings, the action is as follows :

- 1) $LEAD_TIME = 0$ and $LAG_TIME = 5$

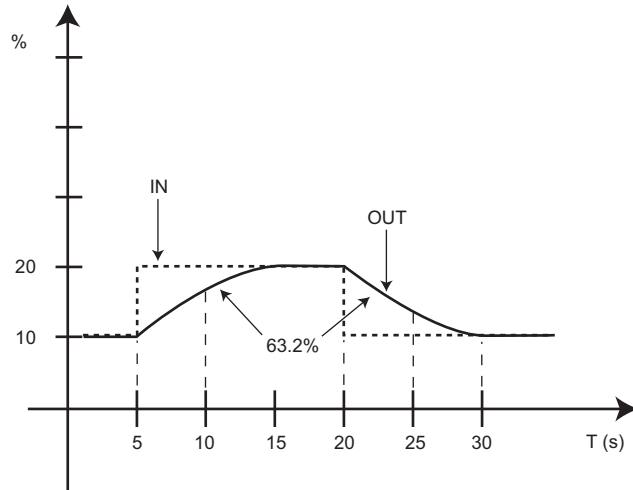


Fig. 7-47: Example with $Lead_TIME = 0$ and $LAG_TIME = 5$

- 2) $LEAD_TIME = 5$ and $LAG_TIME = 0$

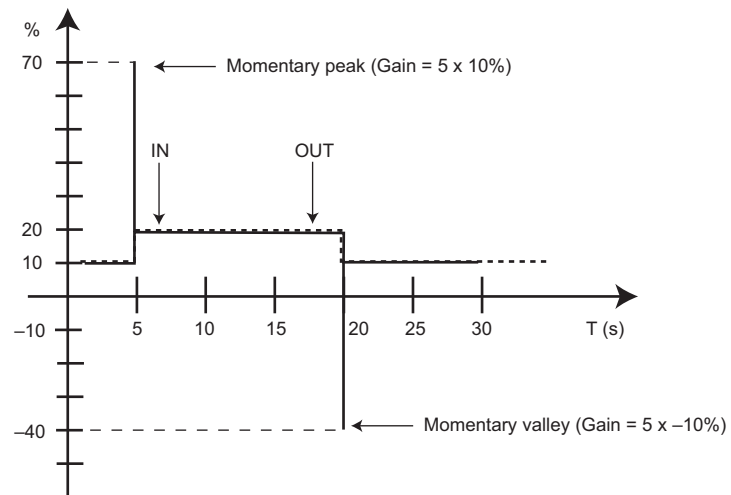


Fig. 7-48: Example with $Lead_TIME = 5$ and $LAG_TIME = 0$

3) LEAD_TIME = 5 and LAG_TIME = 10

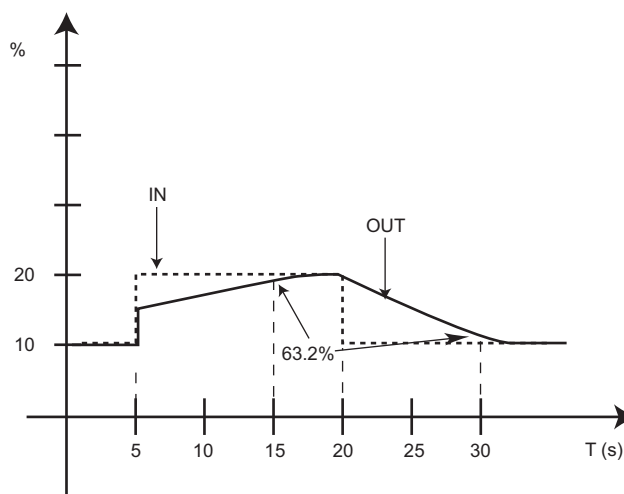


Fig. 7-49: Example with Lead_TIME = 5 and LAG_TIME = 10

4) LEAD_TIME = 10, LAG_TIME = 5

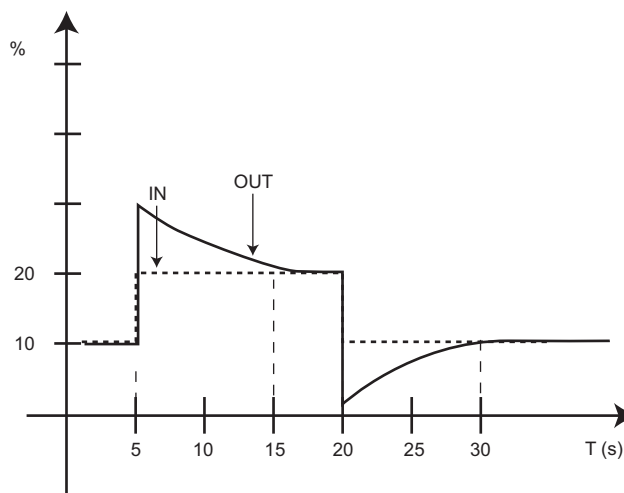


Fig. 7-50: Example with Lead_TIME = 10 and LAG_TIME = 5

5) LEAD_TIME = 10, LAG_TIME = 5, FOLLOW = TRUE

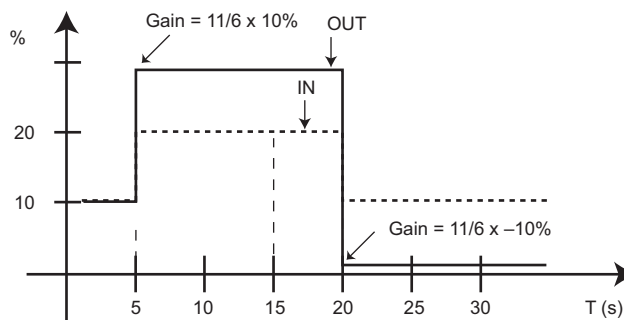


Fig. 7-51: Example with Lead_TIME = 10 and LAG_TIME = 5, FOLLOW = TRUE

7.12.2 Block configuration

Control strategy:
feedforward with lead/lag

Fig. 7-52 shows an example for feedforward control. The temperature of product leaving a reactor vessel is controlled using a temperature measurement to manipulate the control valve in the heating stream. The temperature of the feed is a disturbance variable, which is fed forward to the control loop. Lead/lag is used to compensate for a change in feed temperature.

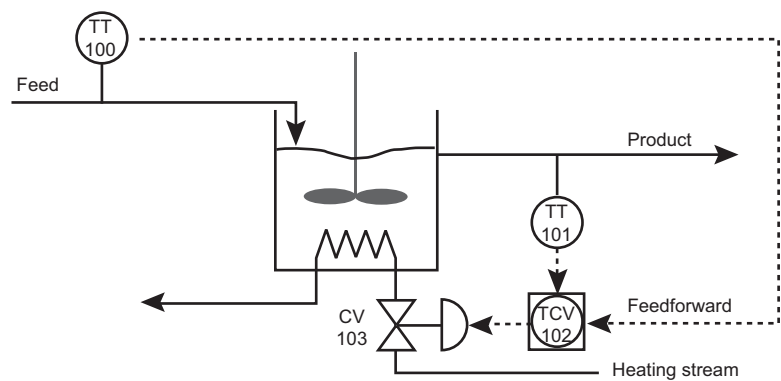


Fig. 7-52: Feedforward control application for product temperature

Fig. 7-53 shows the control strategy together with the links that must be made for the application shown in Fig. 7-52.

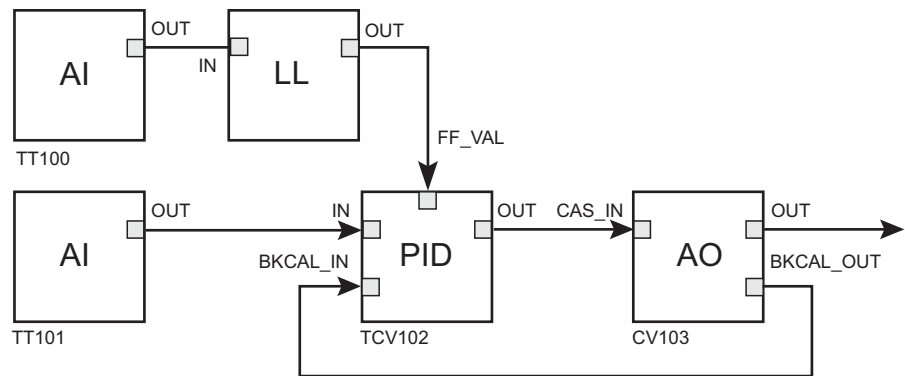


Fig. 7-53: Links to be made for Lead/Lag block for control strategy described above

Block configuration:
safety interlock

The Lead/Lag block is configured for the strategy in Fig 7-53 as follows:

Parameter	Action	Option/Value
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
LAG_TIME	Lag time constant for the block	e.g. 5
LEAD_TIME	Lead time constant for the block	e.g.10
BAL_TIME	Time in seconds allowed to balance OUT changes as a result of a change from Auto to Man	10 (s)

Note



- The values for LAG_TIME and LEAD_TIME in the table should not be taken as being realistic for the application above. The user must determine these by calculating the dynamic compensation and tune them by observing the effects on the loop.

7.12.3 Block operation

Mode

The block normally operates in Auto mode.

When **MODE_BLK.Target** is set to Man, the block output **OUT** can be manipulated on-line. If the **BAL_TIME** parameter is not configured (=0), the transition from the written value (Man) to the calculated value (Auto) will be done with bump. A bumpless transfer can be made by entering an appropriate value, in seconds, in **BAL_TIME**.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Operation can be checked on-line by viewing a number of parameters:

IN	Value and status of analog input signal
OUT	Value and status of output signal

Status

The significance of the output status is indicated below.

OUT status	Meaning
Good	<ul style="list-style-type: none"> Normal operation as configured
Uncertain	<ul style="list-style-type: none"> Input has "Uncertain" status and STATUS_OPTS "Use Uncertain as Good" is not set Block in Man and STATUS_OPTS "Uncertain if Man" set
Bad	<ul style="list-style-type: none"> Block out of service Inputs has "Bad" status

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Out-of-Service	<ul style="list-style-type: none"> MODE_BLK.Target currently set to OOS

7.12.4 Block parameters

Lead-Lag block

Parameter	Valid range/ Options	Default value	Description
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set MODE_BLK.Target to AUTO, CAS or RCAS
BLOCK_ERR	RO		Block errors, see Chapter 6.1.9
OUT			Primary analog value calculated as a result of executing the function
OUT_RANGE			Output range and units for HMI application
GRANT_DENY		0	Access options, see Chapter 2.8.1
STATUS_OPTS		0	Status options, see Chapter 2.8.4 and Chapter 6.1.9
IN			Value and status of analog input
FOLLOW		Or	Tracking input <ul style="list-style-type: none"> TRUE causes OUT to follow IN
LAG_TIME		Measure	Specifies the lag time constant for the block. <ul style="list-style-type: none"> Based on a step change to the input this is the time to reach 63.2% of the final value.
LEAD_TIME		RO	Specifies the lead time constant for the block.
BAL_TIME	Positive	0	Time in seconds allowed to balance OUT changes as a result of a change from Auto to Man
OUTAGE_LIM			Not used
UPDATE_EVT			This alert is generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
Legend: RO = Read Only			

7.13 Output Signal Selector and Dynamic Limiter

The OSDL block can be used as either an output signal selector or dynamic limiter.

- As output selector, the cascade input may be routed to one of two outputs
- As dynamic limiter, the cascade input is transferred to both outputs, but is limited by the secondary input multiplied by a gain and a bias.

It is a customized block that is extremely useful for combustion control applications with double cross limits.

In ControlCare Application Designer the Output Selector and Dynamic Limiter block is assigned the generic name **Device Tag-OSDL-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 7-54.

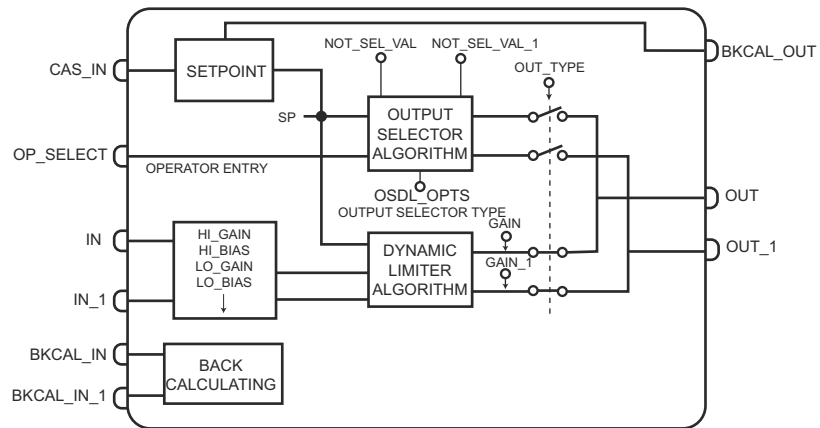


Fig. 7-54: Schematic diagram of Output Signal Selector and Dynamic Limiter block

7.13.1 Functional description

The output signal selector and dynamic limiter block provides two different algorithms which are selected through the **OSDL_TYPE** parameter.

- As Output Selector the cascade input may be routed for one of two outputs based on the value of the **OP_SELECT** input parameter. The output not selected may be treated in two ways: it retains the last value when not selected or receives a internal value.
- As Dynamic Limiter the cascade input is transferred to both outputs, but it is limited by the secondary inputs multiplied by a gain and offset by a bias.

Output Signal Selector

When the "Output Selector" option is selected in **OSDL_TYPE**, the setpoint value **SP** may be entered by an operator (Auto mode) or provided by a another function block through the **CAS_IN** parameter (Cas mode). The inputs **IN** and **IN_1** are not used in this algorithm and their value and status is ignored.

OP_SELECT is a discrete input parameter that selects one of two outputs to receive the **SP** value:

- If **OP_SELECT** = "0", **SP** is passed on to the output **OUT**.
- If **OP_SELECT** = "1", **SP** is passed on to the output **OUT_1**.

If the **OP_SELECT** status is not usable, the block changes to Auto, but the algorithm goes on working with the unusable value. Normally, **SP** is transferred to the selected output no matter what status it has. Therefore, an unusable value and status in the **CAS_IN** will be reflected to the selected output. The status of upper cascade initialization, however, is not copied to the selected output.

The non-selected output is handled in one of two ways. By default, the values contained in **NOT_SEL_VAL** or **NOT_SEL_VAL_1** are transferred to the outputs **OUT** and **OUT_1** respectively. If the option "Keep last value if not selected" is selected in **OSDL_OPTS**, however, the non-selected output will retain the last value.

OS status propagation

Normally, the status of **CAS_IN** is propagated to the selected output. The status of the non-selected output is always "Uncertain" indicating to the downstream block that it is currently not selected. The **STATUS_OPTS** parameter in the downstream block must be configured to deal with it.

If the block is in "Cas" mode and the input is "Bad" the block will be forced to "Auto" mode unless one of the IFS options in **OSDL_OPTS** has been set:

OSDL_OPTS	Input status	Output status	Remarks
Uncertain as Good	Uncertain	Good	Uncertain input status is treated as "Good"
IFS if Bad CAS_IN	Bad	Good IFS	IFS propagated to both outputs
IFS only for selected value	Bad	Good IFS	IFS propagated to selected output only

By default, the status "GoodCascade, IFS" is propagated to both outputs when "IFS if Bad CAS_IN" is selected. If the "IFS only for selected output" option is selected in **OSDL_OPTS**, however, the fault state status will be propagated to selected output only.

If the downstream block of the selected output is not operating in cascade mode, the OSDL block will go to "Iman" mode and the status of **BKCAL_OUT** will change to "GoodCascade, Not Invited". This forces the upstream block into "Iman" mode as well. The OSDL ignores the status of the downstream block connected to the non-selected output.

Dynamic Limiter

When the "Dynamic Limiter" option is selected in **OSDL_TYPE**, the outputs are the value of the **CAS_IN** parameter limited by the following values:

- **OUT:**
 High limit = **HI_GAIN_1** * **IN_1** + **HI_BIAS_1**
 Low limit = **LO_GAIN_1** * **IN_1** - **LO_BIAS_1**
- **OUT_1:**
 High limit = **HI_GAIN** * **IN** + **HI_BIAS**
 Low limit = **LO_GAIN** * **IN** - **LO_BIAS**

After limitation, the parameters **GAIN** and **GAIN_1** are applied as gain to the outputs **OUT** and **OUT_1** respectively.

DL status propagation

The normal mode of operation for the OSDL block as well the two downstream blocks is "Cas". If an input is "Bad" the block will be forced to "Auto" mode unless the IFS options in **OSDL_OPTS** have been set (multiple choices allowed).

OSDL_OPTS	Input status	Output status	Remarks
Uncertain as Good	Uncertain	Good	Uncertain input status is treated as "Good"
IFS if Bad CAS_IN	Bad	Good IFS	IFS propagated to both outputs
IFS if Bad IN	Bad	Good IFS	IFS propagated to both outputs
IFS if Bad IN_1	Bad	Good IFS	IFS propagated to both outputs

The setpoint value **SP** is returned to the upper block through **BKCAL_OUT** by default. One of two alternatives can also be set in **OSDL_OPTS**:

OSDL_OPTS	Remarks
Use OUT for BKCAL_OUT	The OUT value is returned to the upper block
Use OUT_1 for BKCAL_OUT	The OUT_1 value is returned to the upper block

If one downstream block is not in cascade mode, indicated by not invited status (NI) on its **BKCAL_OUT**, the OSDL block still continues in cascade mode. Only if both downstream blocks are not in cascade, does the block change to "Iman" mode. Its **BKCAL_OUT** output then changes to Not Invited (NI).

If the OSDL block is in "Iman" mode and the cascade is initialized with a Initialization Request (IR) by a downstream block, Initialization Acknowledge (IA) is transmitted to the downstream block via the appropriate output (**OUT** or **OUT_1**). depending on the path initializing, the value **BKCAL_IN** or **BKCAL_IN_1** is then passed on to **BKCAL_OUT**. The OSDL block remains in "Iman" mode until the downstream cascade is initialized. At this point OSDL block goes to "Auto" mode and sends an Initialization Request to the upstream block to initialize the cascade.

After a downstream cascade initialization, the corresponding output must ramp from the last **BKCAL_IN** value to the calculated value in **BAL_TIME** seconds. The required actions as a dynamic limiter algorithm are summarized in the following table:

Mode Target/Actual	BKCAL_IN	BKCAL_IN1	BKCAL_OUT	Action
Cas/Iman	NI or IR	NI or IR	NI	
Cas/Cas	NI or IR	OK	OK	BKCAL_OUT receives the CAS_IN value
Cas/Cas	OK	NI or IR	OK	BKCAL_OUT receives the CAS_IN value
Cas/Cas	OK	OK	OK	BKCAL_OUT receives the CAS_IN value
Legend: NI = .Not Invited, IR = Initialization Request, OK = Working in Cascade				

7.13.2 Block configuration

Output Selector

Fig. 7-55 shows an application for the use of the OSDL block as an output selector. The application requires an equal percentage valve with a rangeability considerably in excess of that available with a standard control valve. To achieve this, two valves are chosen, a smaller one which has a Cv range of 2 to 60 and a large one with a Cv range of 20 to 500. Calculations show that the small valve should operate in the range 0% to 60% and the large valve in the range 40% to 100%.

At any value of the controller output only one valve should be open. As the controller output rises from 0%, the smaller valve should open until just below 60%. At this point the large valve should open to approximately 33% and the smaller one close. As the controller output falls from 100%, the small valve should be closed and large valve should gradually close until the controller output is just above 40%. At this point the small valve opens to approx. 66% and the large valve closes.

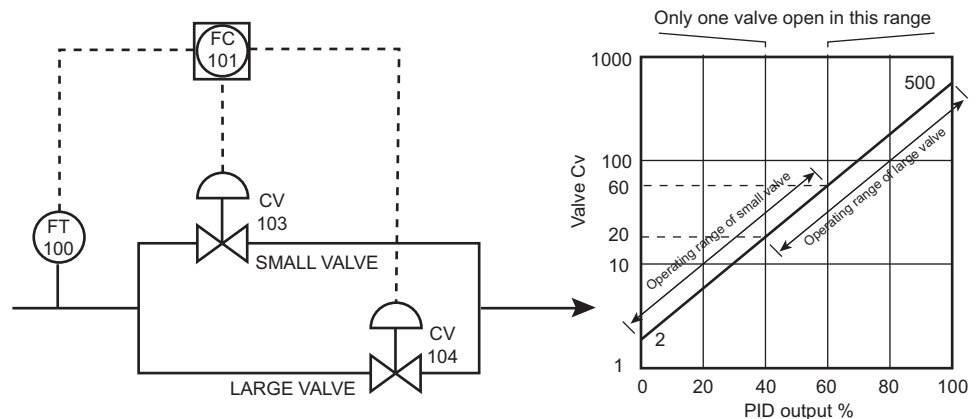


Fig. 7-55: Increasing valve rangeability

This strategy is implemented using the OSDL Block as shown in Fig. 7-56. The valves CV103 and CV104 are controlled by the flow rate measured at FT100 and the PID block. The AALM block monitors the output of the PID block against a HI limit of 60%. If the output value exceeds 60%, the **OUT_D** digital signal changes from false = "0" to true = "1". **OP_SELECT** follows, switching the control from AO_1 to AO_2, causing the small valve to close and the large valve to open.

If the **ALARM_HYS** parameter in the AALM block is set to 20%, **OUT_D** will remain true until the output falls below $60\% - 20\% = 40\%$. At this point the large valve closes and the small valve opens.

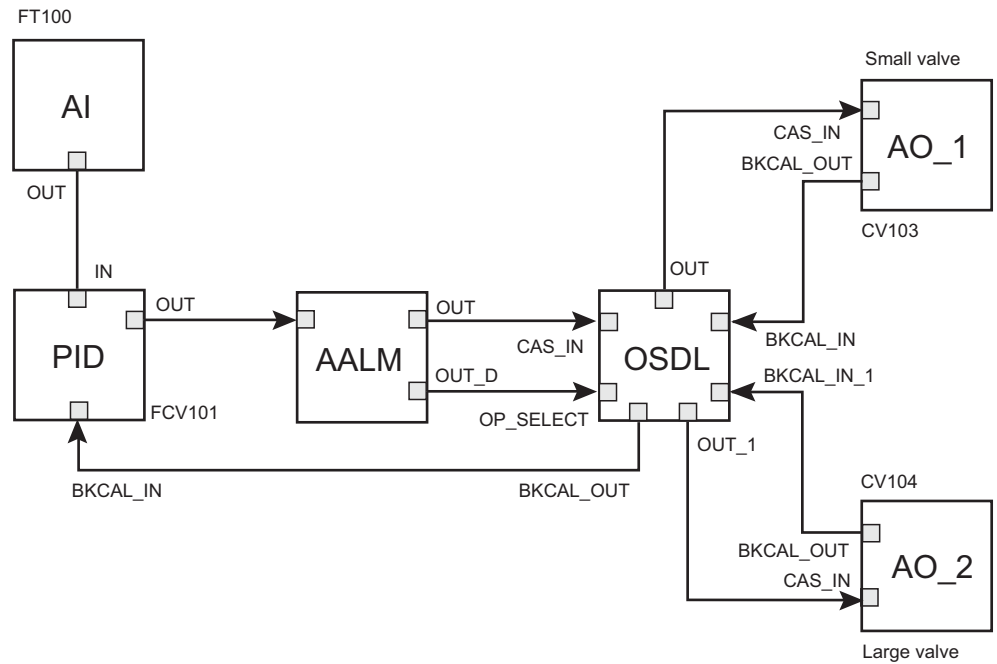


Fig. 7-56: Links to be made for Output Selector and Dynamic Limiter block for control strategy described above

Block configuration: output selector

The AALM, OSDL and AO blocks are configured for the strategy in Fig 7-56 as follows:

Parameter	Action		Option/ Value
AALM block			
MODE_BLK.Target	Set the target mode of the block to Auto	Auto	
OUT_RANGE.EU_100	Enter upper range limit for the input variables (HMI)	e.g. 100	
OUT_RANGE.EU_0	Enter lower range limit for input variables (HMI)	e.g. 0	
OUT_RANGE.UNITS_INDEX	Enter unit of the input variables (HMI)	e.g. %	
PV_FTIME	Enter time constant in seconds for 1st. order damping	e.g. 3	
ALARM_HYS	Enter alarm hysteresis in % of OUT_RANGE	e.g. 20	
HI_LIM	Enter limit of HI alarm in engineering units	e.g. 60	
OSDL block			
MODE_BLK.Target	Set the target mode of the block to Auto	Cas	
OUT_TYPE	Mode in which the block operates	Output Selector	
OSDL_OPTS	If required, block options for IFS,	IFS if Bad CAS_IN	
BAL_TIME	Time in seconds allowed to balance OUT changes as a result of a change from Cas or Auto to Man	10 (s)	
NOT_VAL_SEL	Value of OUT when not selected (= valve closed)	0	
NOT_VAL_SEL_1	Value of OUT_1 when not selected (= valve closed)	0	
AO blocks		AO_1	AO_2
MODE_BLK.Target	Set the target mode of the block to Cas	Cas	Cas
PV_SCALE.EU_100	Set PID output value equivalent to 100% valve travel	60	100
PV_SCALE.EU_0	Set PID output value equivalent to 0% valve travel	0	40
PV_SCALE.UNITS_INDEX	Set units to %	%	%
XD_SCALE.EU_100	Set transducer value equivalent to 100% valve travel	100	100
XD_SCALE.EU_0	Set transducer value equivalent to 0% valve travel	0	0
XD_SCALE.UNITS_INDEX	Set units to %	%	%
CHANNEL	Enter transducer channel number connected to block	1	1

Dynamic Limiter

Fig. 7-57 shows an application for the use of the OSDL block as a dynamic limiter. The control strategy is designed to keep the air-fuel ratio to a furnace burner strictly within predetermined limits. A sudden change in load would require a corresponding change in air and fuel.

The master controller supplies setpoint values to the air and fuel flow controllers while it is stabilized. During a transition, the air flow determines the maximum upper and lower limits that the fuel flow cannot exceed. The same occurs for the air flow, whose limits are fixed by those of the fuel flow. In this way, even when there is a large shift in the master signal, the air/fuel ratio is maintained very close to the desired value. The “double cross limits” prevent that the fastest variable unbalances the desired ratio.

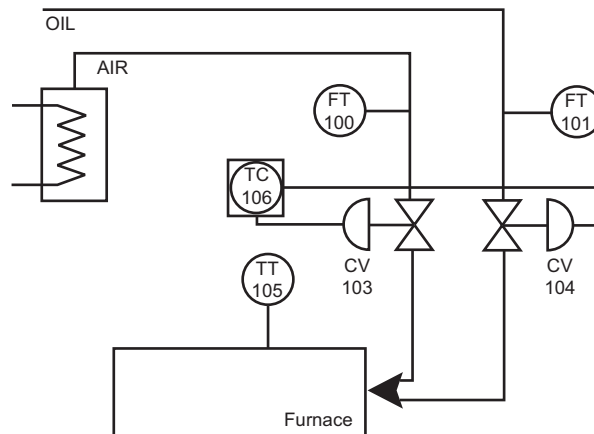


Fig. 7-57: Double cross limits control of a furnace burner

This strategy is implemented using the OSDL Block as shown in Fig. 7-58. This generates the setpoint for the air and fuel controllers based on:

- the output of the master controller,
- air flow ($Q_a \rightarrow IN$ parameter)
- and fuel flow ($Q_c \rightarrow IN_1$ parameter).

This configuration allows the:

- air flow setpoint to vary between $(Q_c - LO_BIAS)$ and $(Q_c + HI_BIAS)$
- fuel flow setpoint to vary just between $(Q_a - LO_BIAS_1)$ and $(Q_a + HI_BIAS_1)$.

When the double crossed limit is interfered, an unexpected change in the consumption upsets the desired ratio. Similarly when there is a transient in the master signal of the air/fuel flow, the block it is able to maintain a value very close to the desired ratio.

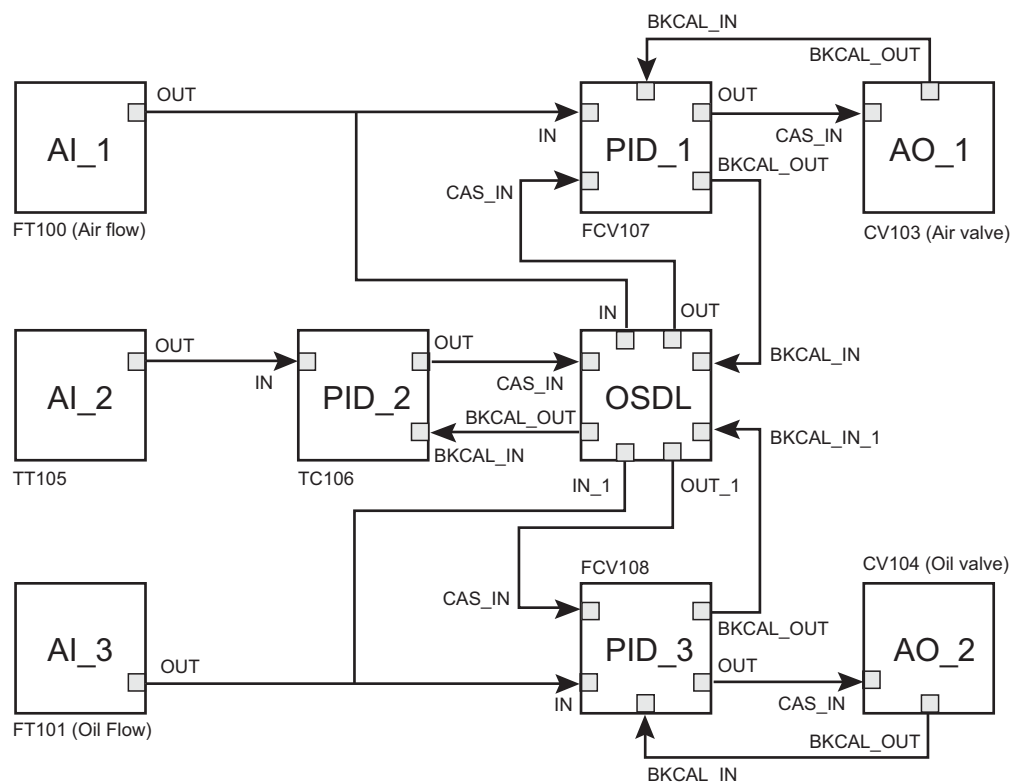


Fig. 7-58: Links to be made for Output Selector and Dynamic Limiter block for control strategy described above

Block configuration: dynamic limiter

The OSDL block is configured for the strategy in Fig 7-58 as follows:

Parameter	Action	Option/Value
MODE_BLK.Target	Set the target mode of the block to Cas	Cas
OUT_TYPE	Mode in which the block operates	Dynamic Limiter
OSDL_OPTS	If required, block options for IFS,	IFS if Bad CAS_IN IFS if Bad IN IFS if Bad IN_1
HI_GAIN	Gain on OUT_1 high limit	1
HI_BIAS	Bias on OUT_1 high limit	5%
LO_GAIN	Gain on OUT_1 low limit	1
LO_BIAS	Bias on OUT_1 low limit	2%
HI_GAIN_1	Gain on OUT high limit	1
HI_BIAS_1	Bias on OUT high limit	2%
LO_GAIN_1	Gain on OUT low limit	1
LO_BIAS_1	Bias on OUT low limit	5%
GAIN	Gain on OUT value	1
GAIN_1	Gain on OUT_1 value	1
BAL_TIME	Time in seconds allowed to balance OUT changes as a result of a change from Cas or Auto to Man	10 (s)

7.13.3 Block operation

Mode

The block normally operates in "Cas" mode.

When the "Output Selector" option is selected in **OSDL_TYPE** the block may also operate in "Auto" mode. In this case the operator must set **MODE_BLK.Target** to "Auto" and enter a setpoint **SP**. The setpoint can be changed during operation without changing the block mode to OOS.

When **MODE_BLK.Target** is set to "Man", the block outputs **OUT** and **OUT_1** can be manipulated. If the **BAL_TIME** parameter is not configured (=0), the transition from the written value (Man) to **SP** (Auto) or **CAS_IN** value (Cas) will be done with bump. A bumpless transfer can be made by entering an appropriate value, in seconds, in **BAL_TIME**.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Operation can be checked on-line by viewing a number of parameters:

CAS_IN	Value and status of setpoint supplied by upstream block
SP	Value and status of operator entered setpoint
IN	Value and status of analog input signal
IN_1	Value and status of analog input signal 1
OUT	Value and status of output signal
OUT_1	Value and status of output signal 1

Status

The significance of the output status is indicated below.

OUT/OUT_1 status	Meaning
Good	<ul style="list-style-type: none"> Normal operation as configured Uncertain input but OSDL_OPTS "Uncertain as Good" set Bad input but IFS options set
Uncertain	<ul style="list-style-type: none"> Block operating normally as Output Selector, but output currently not selected
Bad	<ul style="list-style-type: none"> Block out of service One or more inputs has "Bad" status and IFS not set (block goes to Auto)

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Out-of-Service	<ul style="list-style-type: none"> MODE_BLK.Target currently set to OOS
Block configuration error	<ul style="list-style-type: none"> OUT_TYPE has an invalid value

7.13.4 Block parameters

Output Selector and Dynamic Limiter block

Parameter	Valid range/ Options	Default value	Description
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set MODE_BLK.Target to AUTO, CAS or RCAS
BLOCK_ERR	RO		Block errors, see Chapter 6.1.9
CAS_IN	RO		Setpoint used by block on CAS mode
SP			Setpoint used by the block in AUTO mode
IN			Primary input value of the block
IN_1			Secondary input value of block
OP_SELECT	0 - 4		Operator adjustable parameter to force a given input to be used
OUT			Primary output value as a result of executing the block
OUT_1			Secondary output value as a result of executing the block
GRANT_DENY		0	Access options, see Chapter 2.8.1
OSDL_TYPE		0	Algorithm to be used by the block <ul style="list-style-type: none"> Invalid value (default) Output Selector Dynamic Limiter
OSDL_OPTS		0	Status options for block, see also Chapter 2.8.4
HI_GAIN	Positive	0	Gain used for high limit of OUT_1 (Dynamic Limiter)
HI_BIAS		1.1	Bias used for high limit of OUT_1 (Dynamic Limiter)
LO_GAIN	Positive	0	Gain used for low limit of OUT_1 (Dynamic Limiter)
LO_BIAS		0.9	Bias used for low limit of OUT_1 (Dynamic Limiter)
HI_GAIN_1	Positive	0	Gain used for high limit of OUT (Dynamic Limiter)
HI_BIAS_1		1.1	Bias used for high limit of OUT (Dynamic Limiter)
LO_GAIN_1	Positive	0	Gain used for low limit of OUT (Dynamic Limiter)
LO_BIAS_1		0.9	Bias used for low limit of OUT (Dynamic Limiter)
GAIN		1	Gain applied to OUT after limiting (Dynamic Limiter)
GAIN_1		1	Gain applied to OUT after limiting (Dynamic Limiter)
BKCAL_IN			Feedback from BKCAL_OUT of a downstream block
BKCAL_IN_1			Feedback from BKCAL_OUT of a downstream block
BKCAL_OUT			Feedback for BKCAL_IN of an upstream block.
BAL_TIME	Positive	0	Time allowed to balance OUT changes as a result of a change from Auto, Cas or Rcas to Man
NOT_SEL_VAL			Value of OUT when not selected (Output Selector)
NOT_SEL_VAL_1			Value of OUT_1 when not selected (Output Selector)
UPDATE_EVT			This alert is generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
Legend: RO = Read Only			

7.14 Constant

The Constant block generates constant values for use as inputs parameters to other blocks. Each Constant block allows up to six analog values (floating point) and two discrete values (0, 1) to be simulated.

In ControlCare Application Designer the Constant block is assigned the generic name **Device Tag-CT-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 7-59.

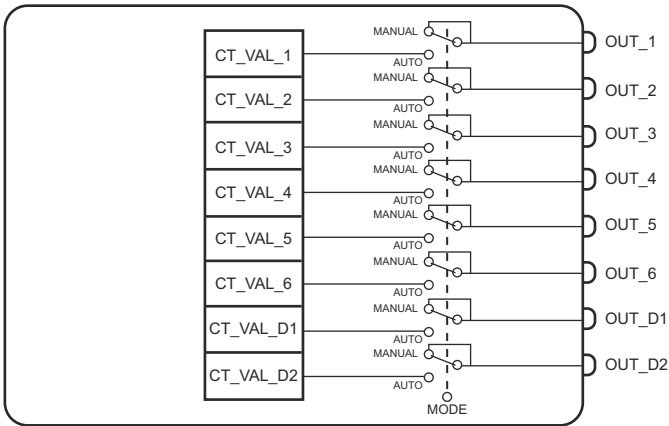


Fig. 7-59: Schematic diagram of Constant block

7.14.1 Functional description

The Constant block allows the entry of

- up to six analogue values in **CT_VAL_1** to **CT_VAL_6**
- as well as two discrete values in **CT_VAL_D1** and **CT_VAL_D2**.

The inputs are connected to the

- outputs **OUT_1** to **OUT_6**
- and **OUT_D1** and **OUT_D2** respectively.

The outputs are connected to other downstream blocks as required.

7.14.2 Block configuration

The constant block may be used for a number of purposes, e.g. for providing setpoints and tracking values to PID Control or Analog Alarm blocks. Fig. 7-23 in Chapter 7.6.2 shows an example of its use in a simple temperature control application. Similarly, it may be used to supply a digital switching signal, e.g. to stop or control the execution of a variety of function blocks.

The block is configured by simply entering the desired value into one of the constant parameters, e.g. for a setpoint temperature of 300°C at **OUT_1** and a discrete signal "1" at **OUT_D1** the following entries are made. Default value for unconfigured inputs is "0":

Parameter	Function	Value
MODE BLOCK/TARGET	Normal operating mode of block	Auto
CT_VAL_1	Value to be output as OUT_1 by the constant block	300
CT_VAL_D1	Value to be output as OUT_D1 by the constant block	1

7.14.3 Block operation

Mode

The block normally operates in "Auto" mode.

When **MODE_BLK.Target** is set to "Man", the block outputs **OUT_1** to **OUT_6** as well as **OUT_D1** and **OUT_D2** can be manipulated.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Operation can be checked on-line by viewing a number of parameters:

OUT_1 to OUT_6	Value and status of corresponding analog output signal
OUT_D1, OUT_D2	Value and status of corresponding discrete signal

Status

The significance of the output status is indicated below.

OUT/OUT_1 status	Meaning
Good	■ Normal operation as configured
Bad	■ Block out of service

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Out-of-Service	■ MODE_BLK.Target currently set to OOS

7.14.4 Block parameters

Constant block

Parameter	Valid range/ Options	Default value	Description
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set MODE_BLK.Target to AUTO, CAS or RCAS
BLOCK_ERR	RO		Block errors, see Chapter 6.1.9
OUT_1			Output value generated from CT_VAL_1
OUT_2			Output value generated from CT_VAL_2
OUT_3			Output value generated from CT_VAL_3
OUT_4			Output value generated from CT_VAL_4
OUT_5			Output value generated from CT_VAL_5
OUT_6			Output value generated from CT_VAL_6
OUT_D1			Output value generated from CT_VAL_D1
OUT_D2			Output value generated from CT_VAL_D2
CT_VAL_1		0	Constant value connected to OUT_1
CT_VAL_2		0	Constant value connected to OUT_2
CT_VAL_3		0	Constant value connected to OUT_3
CT_VAL_4		0	Constant value connected to OUT_4
CT_VAL_5		0	Constant value connected to OUT_5
CT_VAL_6		0	Constant value connected to OUT_6
CT_VAL_D1		0	Constant value connected to OUT_D1
CT_VAL_D2		0	Constant value connected to OUT_D2
UPDATE_EVT			This alert is generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
Legend: RO = Read Only			

7.15 Constant and Contained RW

The Constant and Contained Read/Write block has two functions:

- generating constant values to use as inputs parameters to other blocks. In this case, it has exactly the same function as the constant block, see Chapter 7.14.
- reading from or writing to contained parameters of blocks being executed within the device from which it originates.

In ControlCare Application Designer the Constant and Contained RW block is assigned the generic name **Device Tag-CTRW-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 7-59.

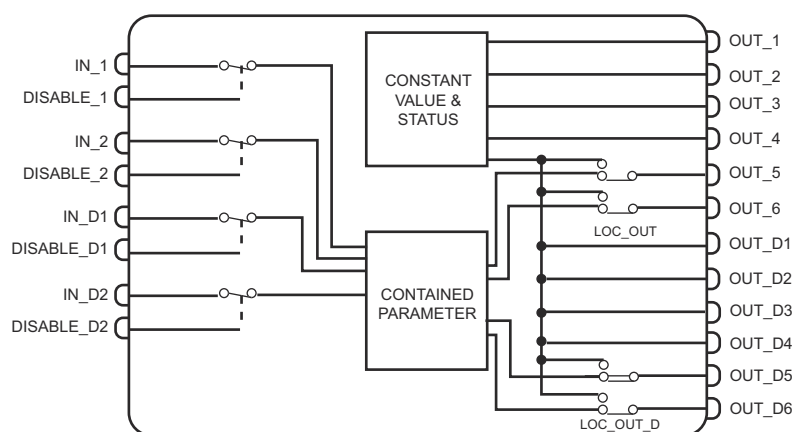


Fig. 7-60: Schematic diagram of Constant and Contained RW block

7.15.1 Functional description

Constant generator

The Constant block allows the entry of

- up to four analogue values in **CT_VAL_1** to **CT_VAL_4**
- as well as up to four discrete values in **CT_VAL_D1** to **CT_VAL_D4**.

The inputs are connected to the

- outputs **OUT_1** to **OUT_4**
- and **OUT_D1** to **OUT_D4** respectively.

The outputs are connected to other downstream blocks as required.

If the outputs **OUT_5**, **OUT_6**, **OUT_D5** and **OUT_D6** are not configured for read/write operation, they are also linked to the values in **CT_VAL_5**, **CT_VAL_6**, **CT_VAL_D5** and **CT_VAL_D6**.

Contained Read/Write

Contained parameters of other blocks can be overwritten or read when the block is in Auto mode. The contained parameter is addressed through the **LOC_xxx** parameter which comprises the elements below (index and subindex can be read from the Offline Characterization window):

Element name	Data type	Size	Description
BLOCK_TAG_xxx	VisibleString (32)	32	Tag of the block containing the contained parameter (case sensitive!)
INDEX_:RELATIVE_xxx	Unsigned16	2	Relative index of contained parameter (= offset)
SUB_INDEX_xxx	Unsigned8	1	Subindex of structured parameter, counted from 1. <ul style="list-style-type: none"> ■ Simple parameters: no entry is required ■ Structured parameters: number of structure element ■ Bitstrings: byte of the parameter to be considered.

For example, to address **IN_1**, the parameters **BLOCK_TAG_IN1**, **INDEX_RELATIVE_IN1** and for a structured parameter or bit string **SUB_INDEX_IN1** must be configured.

The input/output is considered to be not configured when:

- the parameter **BLOCK_TAG_XXX** is blank or
- **INDEX_RELATIVE_XXX** and **SUB_INDEX_XXX** are equal to zero.

In this case, the constant value **CT_VAL_XXX** is routed to the corresponding output.

If there is a configuration error in the contained parameter, the correspondent bit of the input/output in **CONFIG_STATUS** will be set and the **BLOCK_ERR** will indicate "Configuration Error".

Reading or writing is not supported in all parameter types. The table below summarizes the operations supported by the block:

Parameter Input/Output	Data type	Direction		Data type of contained parameter
		From	To	
IN_1, IN_2	Float	IN_1, IN_2	Any other block parameter	Boolean* Float Integer8* Integer16* Integer32 Unsigned8* Unsigned16* Unsigned32 Bitstring*
IN_D1, IN_D2	Unsigned8	IN_D1, IN_D2	Any other block parameter	Boolean Float Integer8 Integer16 Integer32 Unsigned8 Unsigned16 Unsigned32 Bitstring*
OUT_5, OUT_6	Float	Block parameter	OUT_5, OUT_6	Boolean Float Integer8 Integer16 Integer32 Unsigned8 Unsigned16 Unsigned32 Bitstring
OUT_D5, OUT_D6	Unsigned8*	Block parameter	OUT_D5, OUT_D6	Boolean Float Integer8 Integer16 Integer32 Unsigned8 Unsigned16 Unsigned32 Bitstring
Note 1: For data types with "*" the block cuts the values out of the range of the correspondent type: <ul style="list-style-type: none"> ■ Unsigned8/Bitstring: 0 to 255 ■ Integer8: -127 to +127 ■ Boolean: 0 or 1 ■ Unsigned16: 0 to 65535 ■ Integer16: -32767 to +32767 				
Note 2: For the BitString type, the Subindex identifies the correspondent Byte.				
Note 3: The reading or writing in contained parameters are not supported in the same block Constant.				

Writing to contained parameters

When the block is in Auto mode, the value of the **IN_xx** parameter is written into the contained parameter addressed by the **LOC_xxx** parameters according to the following table:

Parameter	Value	Status	DISABLE_xx	Action
IN_x	>DEAD_BAND_x or < DEAD_BAND_x	Good	FALSE (= "0")	Value written to contained parameter
IN_x,	>DEAD_BAND_x or < DEAD_BAND_x	Uncertain	FALSE (= "0")	If STATUS_OPTS "Use Uncertain as Good" is set, value written to contained parameter
IN_x	–	Good, Uncertain	TRUE (= "1")	No write
IN_x	–	Bad	–	No write
IN_Dx	Changes from 0 to 1 or changes from 1 to 0	Good	FALSE (= "0")	Value written to contained parameter
IN_Dx,	Changes from 0 to 1 or changes from 1 to 0	Uncertain	FALSE (= "0")	If STATUS_OPTS "Use Uncertain as Good" is set, value written to contained parameter
IN_Dx	Remains the same	Good, Uncertain	TRUE (= "1")	No write
IN_Dx	–	Bad	–	No write

To avoid cyclical writing in static parameters with the resulting cyclical increment of the **ST_REV** and generation of an event by **UPDATE_EVT** two mechanisms are provided:

- For the **IN_x** inputs, **DEAD_BAND_x** allows a band either side of the current value of the contained parameter to be defined for which there shall be no change of the parameter. If **DEAD_BAND_X** is set to zero, the contained value will be overwritten every time the block executes, provided the other conditions are met.
- For the **IN_Dx** inputs, a value is written to the contained parameter only when there is a change in condition from FALSE (= "0") to TRUE (= "1") or vice versa.

If the write to the desired contained parameter block fails, the input concerned appears in the **BAD_STATUS** parameter.

Reading from contained parameters

When the block is in Auto mode, the value of the contained parameter addressed by the **LOC_xxx** parameters is read to the **OUT_xx** parameter according to the following table.

Parameter	Value	Status	Action
OUT_x	Contained value	GoodNonCascade	Value read from contained parameter
OUT_x,	Contained value	BadNoComm	Value could not be read from contained parameter
OUT_x	CT_VAL_x	CT_STATUS_x	No read: BLOCK_TAG_OUT_x not configured or INDEX_RELATIVE_x and SUB_INDEX_x = zero
OUT_Dx	Contained value	GoodNonCascade	Value read from contained parameter
OUT_Dx,	Contained value	BadNoComm	Value could not be read from contained parameter
OUT_Dx	CT_VAL_x	CT_STATUS_x	No read: BLOCK_TAG_OUT_x not configured or INDEX_RELATIVE_x and SUB_INDEX_x = zero

Note: the outputs concerned are OUT_5, OUT_6, OUT_D5 and OUT_D6

If the read from the desired contained parameter block fails, the output concerned appears in the **BAD_STATUS** parameter.

7.15.2 Block configuration

Constant generator

The constant block may be used for a number of purposes, e.g. for providing setpoints and tracking values to PID Control or Analog Alarm blocks. Fig. 7-23 in Chapter 7.6.2 shows an example of its use in a simple temperature control application. Similarly, it may be used to supply a digital switching signal, e.g. to stop or control the execution of a variety of function blocks. Chapter 7.14.2 contains an example of the block configuration for this example.

Writing to contained parameters

A write to a contained parameter might be necessary to automatically override the output of a PID block during start-up. For example, it might be desirable to heat quickly until a certain temperature is reached and then control normally. The conditions for this might be:

- PID block: PV_SCALE.EU_100 = 100; PV_SCALE.EU_0 = 0; PV_SCALE.UNIT_INDEX = °C
- Setpoint temperature for start up 100°C
- Setpoint temperature for normal operation 70°C
- Start of normal operation $T > 60^{\circ}\text{C}$
- Dead band $\pm 15^{\circ}\text{C}$

The output from the temperature AI block is checked for HI_LIM = 60°C and LO_LIM = 60°C by two analog alarm blocks. When the heating is started up, the temperature is less than 60°C, so that AALM_1.OUT_ALM = FALSE and AALM_2.OUT_ALM = TRUE. Under these conditions, the setpoint provided by OUT_1 (= 100°C) of the CTRW block is used in the PID block. When the temperature reaches 60°C, AALM_1.OUT_ALM = TRUE and AALM_2.OUT_ALM = FALSE. Under these conditions, the setpoint provided by OUT_2 (= 70°C) of the CTRW block is used in the PID block. The dead band ensures that the setpoint is only changed once, and that there is no jitter around the switching point. Please note that this example does not consider what happens on shut down.

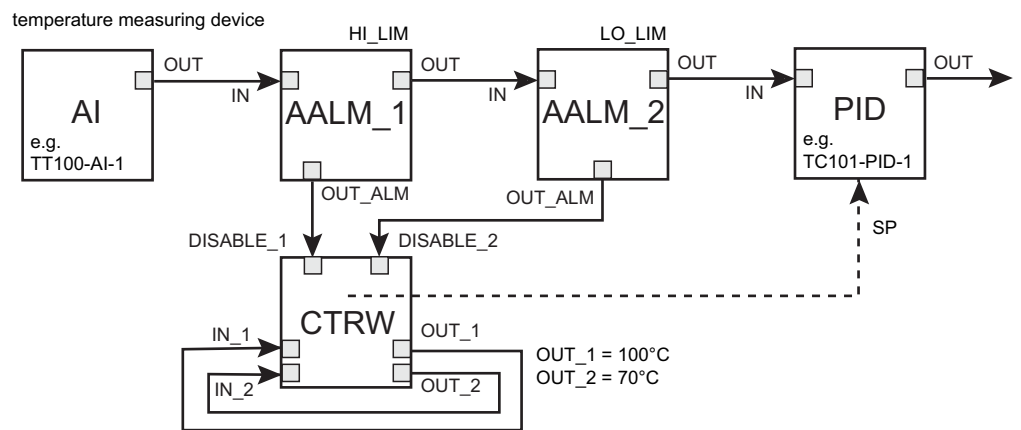


Fig. 7-61: Links to be made in the Constant and Contained RW block for the strategy above

Block configuration

The Analog Alarm and CTRW blocks are configured as follows for the above applications:

Parameter	Action	Option/Value
Block AALM_1		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
PV_FTIME	Enter time constant in seconds for 1st. order damping	e.g. 3
ALARM_HYS	Enter alarm hysteresis in % of OUT_RANGE	e.g. 0
HI_LIM	Enter limit of HI alarm in engineering units	e.g. 60
Block AALM_2		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
PV_FTIME	Enter time constant in seconds for 1st. order damping	e.g. 3
ALARM_HYS	Enter alarm hysteresis in % of OUT_RANGE	e.g. 0
LO_LIM	Enter limit of LO alarm in engineering unit	e.g. 60
Block CTRW		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
CT_VAL_1	Value output at OUT_1 in Auto mode	100
CT_VAL_2	Value output at OUT_2 in Auto mode	60
BLK_TAG_IN1	Block tag of PID block as found in control strategy	e.g. TC101-PID-1
INDEX_RELATIVE_IN1	The offset of the SP parameter in the PID block	8
DEAD_BAND_1	Dead band associated with IN_1 in same units	15
BLK_TAG_IN2	Block tag of PID block as found in control strategy	e.g. TC101-PID-1
INDEX_RELATIVE_IN2	The offset of the SP parameter in the PID block	8
DEAD_BAND_2	Dead band associated with IN_2 in same units	15

Reading from contained parameters

A possible application for reading a contained parameter would be to read the value of the TIME_POS_T (TIME_POS_T.VALUE) parameter in the Setpoint Ramp Generator block and use this to initiate another control action at a specific time in the ramp cycle. Fig 7-62 shows the links to be made for such an application.

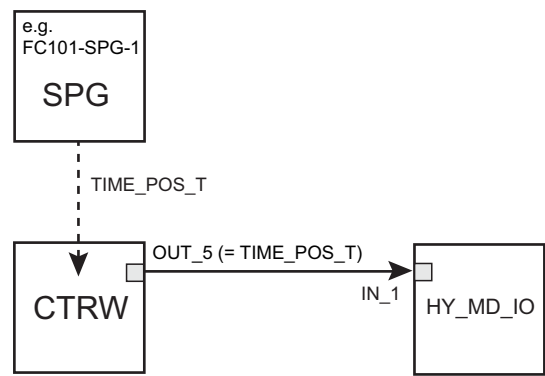


Fig. 7-62: Links to be made in the Constant and Contained RW block for the strategy above

Block configuration

The CTRW block is configured as follows for the above application:

Parameter	Action	Option/Value
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
BLK_TAG_OUT_5	Block tag of PID block as found in control strategy	e.g. FC101-SPG-1
INDEX_RELATIVE_OUT_5	Offset of the TIME_POS_T parameter in the SPG block	22
SUB_INDEX_OUT_5	Subindex of the VALUE part of TIME_POS_T	2

7.15.3 Block operation

Mode

The block normally operates in "Auto" mode.

When **MODE_BLK.Target** is set to "Man", the block outputs **OUT_1** to **OUT_6** as well as **OUT_D1** to **OUT_D6** can be manipulated.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Operation can be checked on-line by viewing a number of parameters:

IN_1 to IN_4	Value and status of corresponding contained parameter
OUT_1 to OUT_6	Value and status of corresponding analog output signal or contained parameter
OUT_D1 to OUT_D6	Value and status of corresponding discrete signal or contained parameter

Status

The significance of the output status is indicated below.

OUT/OUT_1 status	Meaning
Good	■ Normal operation as configured; where appropriate, contained parameter could be read
Bad	■ Block out of service or where appropriate, contained parameter could not be read

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Out-of-Service	■ MODE_BLK.Target currently set to OOS
Block configuration error	■ BLOCK_TAG_XXX has an invalid value ■ INDEX_RELATIVE_XXX has an invalid value ■ SUB_INDEX-XXX has an invalid value ■ DEAD_BAND_XXX has an invalid value

7.15.4 Block parameters

Constant and Contained RW block

Parameter	Valid range/ Options	Default value	Description
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set MODE_BLK.Target to AUTO, CAS or RCAS
BLOCK_ERR	RO		Block errors, see Chapter 6.1.9
OUT_1			Output value generated from CT_VAL_1
OUT_2			Output value generated from CT_VAL_2
OUT_3			Output value generated from CT_VAL_3
OUT_4			Output value generated from CT_VAL_4
OUT_5			Output value generated from CT_VAL_5/contained parameter
OUT_6			Output value generated from CT_VAL_6/contained parameter
OUT_D1			Output value generated from CT_VAL_D1
OUT_D2			Output value generated from CT_VAL_D2
CT_VAL_1		0	Constant value connected to OUT_1
CT_VAL_2		0	Constant value connected to OUT_2
CT_VAL_3		0	Constant value connected to OUT_3
CT_VAL_4		0	Constant value connected to OUT_4
CT_VAL_5		0	Constant value connected to OUT_5
CT_VAL_6		0	Constant value connected to OUT_6
CT_VAL_D1		0	Constant value connected to OUT_D1
CT_VAL_D2		0	Constant value connected to OUT_D2
UPDATE_EVT			This alert is generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
CT_VAL_D3		0	Constant value connected to OUT_D3
CT_VAL_D4		0	Constant value connected to OUT_D5
CT_VAL_D5		0	Constant value connected to OUT_D5
CT_VAL_D6		0	Constant value connected to OUT_D6
CT_STATUS_1		GoodNC	Status of constant value connected to OUT_1, default "Good"
CT_STATUS_2		GoodNC	Status of constant value connected to OUT_2, default "Good"
CT_STATUS_3		GoodNC	Status of constant value connected to OUT_3, default "Good"
CT_STATUS_4		GoodNC	Status of constant value connected to OUT_4, default "Good"
CT_STATUS_5		GoodNC	Status of constant value connected to OUT_5, default "Good"
CT_STATUS_6		GoodNC	Status of constant value connected to OUT_6, default "Good"
CT_STATUS_D1		GoodNC	Status of constant value connected to OUT_D1, default "Good"
CT_STATUS_D2		GoodNC	Status of constant value connected to OUT_D2, default "Good"
CT_STATUS_D3		GoodNC	Status of constant value connected to OUT_D3, default "Good"
CT_STATUS_D4		GoodNC	Status of constant value connected to OUT_D4, default "Good"
CT_STATUS_D5		GoodNC	Status of constant value connected to OUT_D5, default "Good"
CT_STATUS_D6		GoodNC	Status of constant value connected to OUT_D6, default "Good"
OUT_D3			Output value generated from CT_VAL_D3
OUT_D4			Output value generated from CT_VAL_D4
OUT_D5			Output value generated from CT_VAL_D5/contnd parameter
OUT_D6			Output value generated from CT_VAL_D6/contnd parameter
IN_1			Input value and status to be written to contained parameter
DISABLE_1			Discrete signal controlling IN_1 write to contained parameter
BLK_TAG_IN1		blank	Tag of block with contained parameter (case sensitive!)
INDEX_RELATIVE_IN1		0	Offset of contained parameter (see Offline Configuration)
SUB_INDEX_IN1		0	Subindex of contained parameter (see Offline Configuration)
DEAD_BAND_1		1.0	Dead band associated with IN_1 in same units
IN_1			Input value and status to be written to contained parameter
DISABLE_2			Discrete signal controlling IN_2 write to contained parameter
BLK_TAG_IN2		blank	Tag of block with contained parameter (case sensitive!)
INDEX_RELATIVE_IN2		0	Offset of contained parameter (see Offline Configuration)

Parameter	Valid range/ Options	Default value	Description
SUB_INDEX_IN2		0	Subindex of contained parameter (see Offline Configuration)
DEAD_BAND_2		1.0	Dead band associated with IN_2 in same units
IN_D1			Discrete value and status to be written to contained parameter
DISABLE_D1			Discrete signal controlling IN_D1 write to contained parameter
BLK_TAG_IND11		blank	Tag of block with contained parameter (case sensitive!)
INDEX_RELATIVE_IN_D1		0	Offset of contained parameter (see Offline Configuration)
SUB_INDEX_IN_D1		0	Subindex of contained parameter (see Offline Configuration)
IN_D2			Discrete value and status to be written to contained parameter
DISABLE_D2			Discrete signal controlling IN_D2 write to contained parameter
BLK_TAG_IN_D2		blank	Tag of block with contained parameter (case sensitive!)
INDEX_RELATIVE_IN_D2		0	Offset of contained parameter (see Offline Configuration)
SUB_INDEX_IN_D2		0	Subindex of contained parameter (see Offline Configuration)
OUT_5			Value and status read from contained parameter
BLK_TAG_OUT_5		blank	Tag of block with contained parameter (case sensitive!)
INDEX_RELATIVE_OUT_5		0	Offset of contained parameter (see Offline Configuration)
SUB_INDEX_OUT_5		0	Subindex of contained parameter (see Offline Configuration)
OUT_6			Value and status read from contained parameter
BLK_TAG_OUT_6		blank	Tag of block with contained parameter (case sensitive!)
INDEX_RELATIVE_OUT_6		0	Offset of contained parameter (see Offline Configuration)
SUB_INDEX_OUT_6		0	Subindex of contained parameter (see Offline Configuration)
OUT_D5			Discrete value and status read from contained parameter
BLK_TAG_OUT_D5		blank	Tag of block with contained parameter (case sensitive!)
INDEX_RELATIVE_OUT_D5		0	Offset of contained parameter (see Offline Configuration)
SUB_INDEX_OUT_D5		0	Subindex of contained parameter (see Offline Configuration)
OUT_D6			Discrete value and status read from contained parameter
BLK_TAG_OUT_D6		blank	Tag of block with contained parameter (case sensitive!)
INDEX_RELATIVE_OUT_D6		0	Offset of contained parameter (see Offline Configuration)
SUB_INDEX_OUT_D6		0	Subindex of contained parameter (see Offline Configuration)
BAD_OPTS			Indicates contained parameter that cannot be read <ul style="list-style-type: none"> ■ 0: IN_1 ■ 1: IN_2 ■ 2: IN_D1 ■ 3: IN_D2 ■ 4: OUT_5 ■ 5: OUT_6 ■ 6: OUT_D5 ■ 7: OUT_D6
CONFIG_STATUS			Indicates contained parameter input with configuration error <ul style="list-style-type: none"> ■ 0: IN_1 ■ 1: IN_2 ■ 2: IN_D1 ■ 3: IN_D2 ■ 4: OUT_5 ■ 5: OUT_6 ■ 6: OUT_D5 ■ 7: OUT_D6
STATUS_OPTS		0	Status options, see Chapter 2.8.4
Legend: RO = Read Only			

7.16 Flip-flop and Edge Trigger

The Flip-flop and Edge Trigger block acts as a logic element within a control strategy, passing on a discrete signal to downstream blocks, the value of which is dependent upon the discrete signals at two or more block inputs. It can be configured to work as a:

- SR flip-flop
- RS flip-flop
- D-latch
- Rising edge trigger
- Falling edge trigger
- Bi-directional edge trigger

In ControlCare Application Designer the Flip-flop and Edge Trigger block is assigned the generic name **Device Tag-FFET-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 7-60.

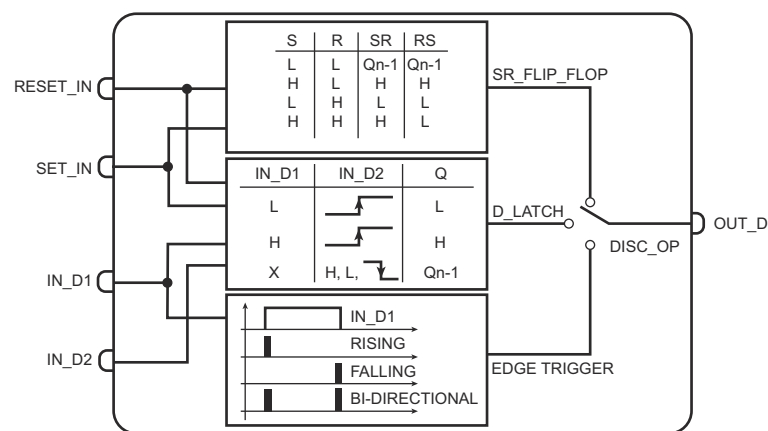


Fig. 7-63: Schematic diagram of Flip-flop and Edge Trigger block

7.16.1 Functional description

The operating mode of the Flip-flop and Edge Trigger block is determined by the option selected in the parameter **DISC_OP**. There are six possible modes of operation which are described below.

DISC_OP = SR flip-flop

The SR flip-flop (or latch) generates a discrete output value **OUT_D** based on the NAND logic of the **SET_IN** (S) and **RESET_IN** (R) discrete inputs.

- If the S and R inputs are both low, **OUT_D** is maintained at a constant state.
- If S is pulsed high while R is held low, then **OUT_D** is forced high, and stays high when S returns to low.
- Similarly, if R is pulsed high while S is held low, then **OUT_D** is forced low, and stays low when R returns to low.
- The combination R high and S high forces **OUT_D** to high.

SET_IN	RESET_IN	OUT_D
FALSE (= 0)	FALSE (= 0)	Keep last state
FALSE (= 0)	TRUE (=1)	FALSE (= 0)
TRUE (=1)	FALSE (= 0)	TRUE (=1)
TRUE (=1)	TRUE (=1)	TRUE (=1)

The SR flip-flop can be used in applications requiring a continuous TRUE condition that is to be activated by a momentary switch and where a set always produces a TRUE signal.

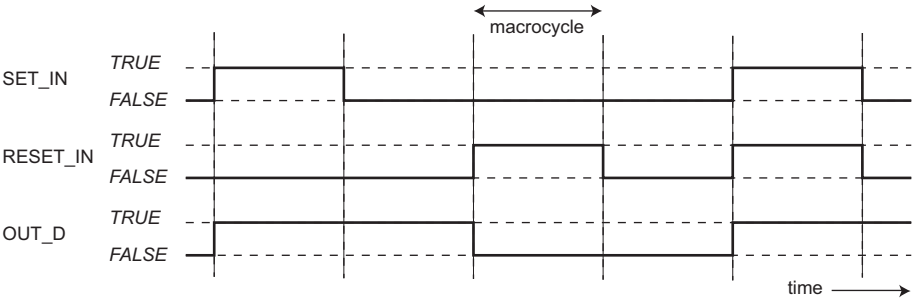


Fig. 7-64: Output of SR flip-flop as a function of SET_IN and RESET_IN inputs

DISC_OP = RS flip-flop

The RS flip-flop (or latch) generates a discrete output value **OUT_D** based on the NOR logic of the **SET_IN** (S) and **RESET_IN** (R) discrete inputs.

- If the S and R inputs are both low, **OUT_D** is maintained at the last state.
- If S is pulsed high while R is held low, then **OUT_D** is forced high, and stays high when S returns to low.
- Similarly, if R is pulsed high while S is held low, then **OUT_D** is forced low, and stays low when R returns to low.
- The combination R high and S high forces **OUT_D** to low.

SET_IN	RESET_IN	OUT_D (RS flip-flop)
FALSE (= 0)	FALSE (= 0)	Keep last state
FALSE (= 0)	TRUE (=1)	State = FALSE (= 0)
TRUE (=1)	FALSE (= 0)	State = TRUE (=1)
TRUE (=1)	TRUE (=1)	State = FALSE (= 0)

The RS flip-flop can be used in applications requiring a continuous TRUE condition that is to be activated by a momentary switch and where a reset always produces a FALSE signal.

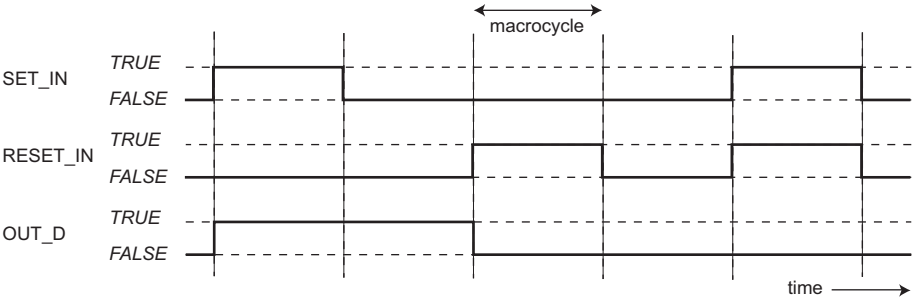


Fig. 7-65: Output of RS flip-flop as a function of SET_IN and RESET_IN inputs

DISC_OP = D-latch

The D-latch option uses the two discrete inputs **IN_D1** and **IN_D2**.

- When **IN_D1** rises, **IN_D2** inverts.
- When **IN_D1** is steady or falls, **IN_D2** keeps the last state (latches)

IN_D1	IN_D2	OUT_D (D-latch)
Rising (0 to 1)	1	0
Rising (0 to 1)	0	1
0, 1 or Falling (1 to 0)	0 or 1	Keep state

The D-latch can be used in applications where a momentary switch or timing pulse is to be used to start and stop the equipment:

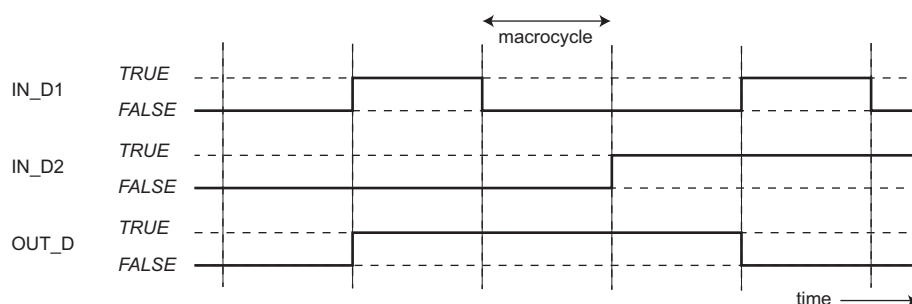


Fig. 7-66: Output of D-latch as a function of *IN_D1* and *IN_D2* inputs

DISC_OP = Rising edge

The rising or positive edge trigger uses the discrete input **IN_D1**. When there is a change of **IN_D1** from 0 to 1, the output at **OUT_D** is set high. **OUT_D** is set low on the next execution of the block.

IN_D1	OUT_D (rising edge)
Rising	1
Falling	0
No transition	0

The positive edge trigger can be used, e.g. to activate a machine starter when a valve opens.

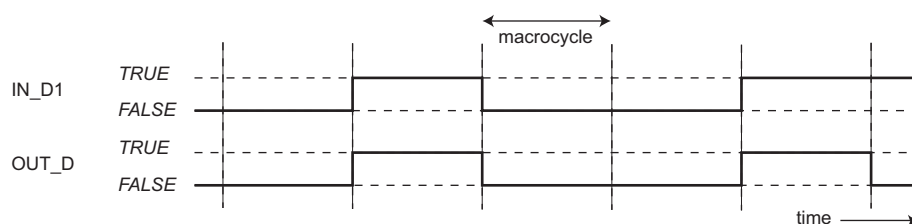


Fig. 7-67: Output of positive edge trigger as a function of *IN_D1*

DISC_OP = Falling edge

The falling of negative edge trigger uses the discrete input **IN_D1**. When there is a change of **IN_D1** from 1 to 0, the output at **OUT_D** is set high. **OUT_D** is set low on the next execution of the block.

IN_D1	OUT_D (falling edge)
Rising	0
Falling	1
No transition	0

The negative edge trigger can be used, e.g. to start actions associated with the shutdown of a machine.

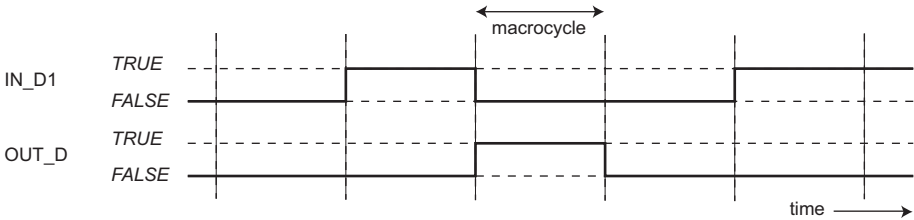


Fig. 7-68: Output of negative edge trigger as a function of IN_D1

DISC_OP = Bi-directional

The bidirectional edge trigger functions uses the discrete input **IN_D1**. When there is a change of **IN_D1** from 0 to 1 or from 1 to 0, **OUT_D** is set high. If there is no transition on the next execution of the block, **OUT_D** is set low..

IN_D1	OUT_D (bidirectional)
Rising	1
Falling	1
No transition	0

The bi-directional edge trigger can be used, e.g. to start actions associated with both the start-up and the shutdown of a machine.

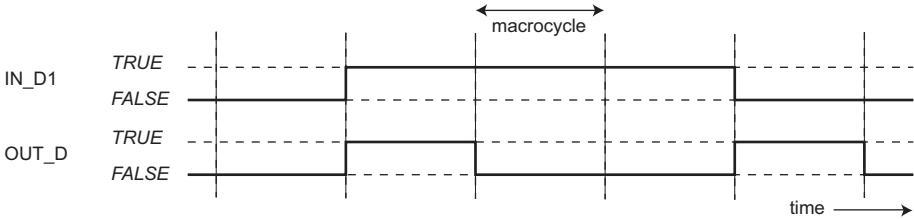


Fig. 7-69: Output of bi-directional edge trigger as a function of IN_D1

Status propagation

The block checks the status of the inputs before processing them. The table below summarizes the status of **OUT** under various conditions. ..

STATUS_OPTS	IN status	OUT status	Remarks
–	Good	Good	Block executed
–	Uncertain	Bad	Block not executed
Use Uncertain as Good	Uncertain	Good	Block executed
–	Bad	Bad	Block not executed

7.16.2 Block configuration

Note



- When using momentary switches and pushbuttons in connection with digital input modules, the period for which the change in state exists must exceed the macrocycle of the control strategy, otherwise it is possible that the change will not be sensed by the controller.
- The same applies to momentary clicks on HMI pushbuttons

RS Flip-flop

The SR flip-flop and RS flip-flop can be used in applications requiring a continuous TRUE condition that is to be activated by a momentary switch. They differ only in their behaviour when both **SET_IN** and **RESET_IN** inputs are true. For the SR flip-flop the output is forced to TRUE, for the RS flip-flop it is forced to FALSE.

Fig. 7-70 shows a control strategy for a simple pump control application. The **SET_IN** and **RESET_IN** signals are connected to pushbuttons at the DI-1 and DI-2 inputs of a SFC432 voltage input/relay output module via a Multiple Discrete Input block. **OUT_D** is connected to the pump motor the at the DO-1 output via a Multiple Discrete Output block. For this example it is assumed that the SFC432 module is located at Slot 0 of the rack with address 1.

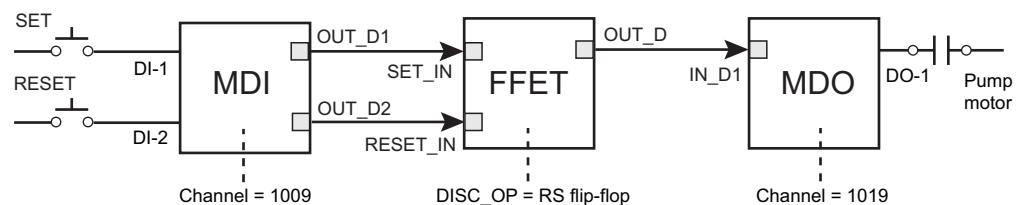


Fig. 7-70: Links to be made to the Flip-flop and Edge Trigger block for the above strategy.

Block configuration

The MDO, FFET and MDI blocks are configured as follows for the above applications:

Parameter	Action	Option/Value
Block MDI		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
CHANNEL	Rack + Slot + Group + 9	1009
Block FFET		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
DISC_OP	Block execution mode	RS flip-flop
Block CTRW		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
CHANNEL	Rack + Slot + Group + 9	1019

D-latch

The D-latch can be used in applications where a momentary switch or timing pulse is to be used to start and stop the equipment. Since the output is the inverse of the signal **IN_D2**, it also offers itself as an interlock solution.

Fig. 7-71 shows a control strategy for a pump start-up with interlock for dry running. The **IN_D1** signal is connected to pushbuttons at the DI-1 input of a SFC432 voltage input/relay output module via a Multiple Discrete Input block. **IN_D2** is connected to a level limit switch at DI-2 that is set to signal TRUE when it is uncovered and FALSE when it is covered. **OUT_D** is connected to the pump motor the at the DO-1 output via a Multiple Discrete Output block. For this example it is assumed that the SFC432 module is located at Slot 0 of the rack with address 1.

When the pushbutton is pressed and the level limit switch is covered, the FALSE state of the switch results in a TRUE state for the pump motor, which starts up. If the switch is not covered, i.e. there is a danger of dry running, the TRUE state of the switch results in a FALSE state for the motor, which cannot be started.

This interlock works on start up only. Should the liquid level fall below the limit switch during operation of the motor, this will not be signalled.

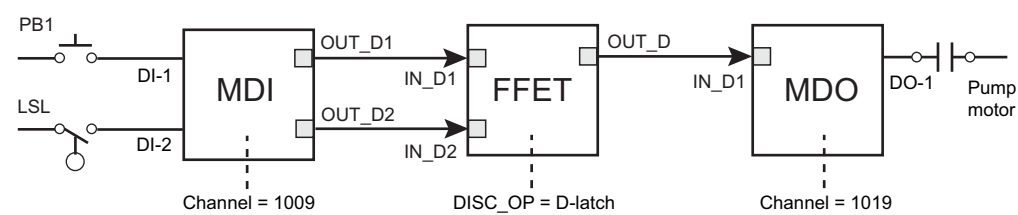


Fig. 7-71: Links to be made to the Flip-flop and Edge Trigger block for the above strategy

Block configuration

The MDO, FFET and MDI blocks are configured as follows for the above application:

Parameter	Action	Option/Value
Block MDI		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
CHANNEL	Rack + Slot + Group + 9	1009
Block FFET		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
DISC_OP	Block execution mode	D-latch
Block CTRW		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
CHANNEL	Rack + Slot + Group + 9	1019

Bi-directional edge trigger

In the case of the bi-directional edge trigger, a change from both FALSE to TRUE and TRUE to FALSE at **IN_D** triggers a change from FALSE to TRUE at **OUT_D**. This signal can be used to trigger a actions required at the start and the shutdown of plant equipment.

Fig. 7-72 shows a control strategy for the start-up and shutdown of a conveyor based on the starting and shutdown of an upstream conveyor. The checkback signal from the motor starter of the upstream conveyor is used start and stop the downstream conveyor, which requires a continuous TRUE signal to start and a FALSE signal to stop. This is produced by the output of the bi-directional trigger acting on the checkback signal from the second motor in a second FFET block operating as a D-latch. The checkback signals are wired to inputs DI-3 and DI-4 of the SFC432, the motor starter to output DO-4. The interfaces between the input signals and output signals are provided by the MDI and MDO blocks respectively.

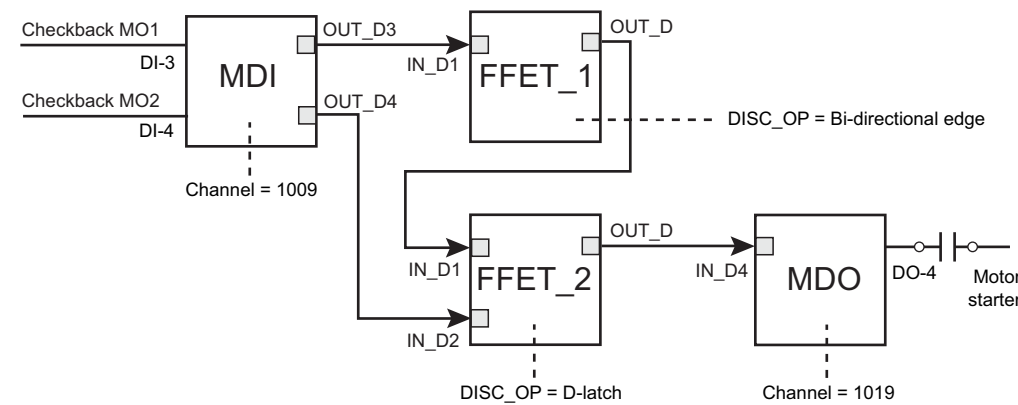


Fig. 7-72: Links to be made to the Flip-flop and Edge Trigger block for the above strategy

Block configuration

The MDO, FFET and MDI blocks are configured as follows for the above application:

Parameter	Action	Option/Value
Block MDI		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
CHANNEL	Rack + Slot + Group + 9	1009
Block FFET1		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
DISC_OP	Block execution mode	Bi-directional edge
Block FFET2		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
DISC_OP	Block execution mode	D-latch
Block MDO		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
CHANNEL	Rack + Slot + Group + 9	1019

7.16.3 Block operation**Mode**

The block normally operates in "Auto" mode.

When **MODE_BLK.Target** is set to "Man", the **OUT_D** can be manipulated.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Operation can be checked on-line by viewing a number of parameters:

IN_D1	Value and status of input 1 supplied by upstream block
IN_D2	Value and status of input 2 supplied by upstream block
SET_IN	Value and status of set input of RS flip-flop or SR flip-flop supplied by upstream block
RESET_IN	Value and status of reset input of RS flip-flop or SR flip-flop supplied by upstream block
OUT_D	Value and status of corresponding discrete output signal

Status

The significance of the output status is indicated below.

OUT/OUT_1 status	Meaning
Good	<ul style="list-style-type: none"> Normal operation as configured Uncertain input but STATUS_OPTS "Use Uncertain as Good" selected
Uncertain	<ul style="list-style-type: none"> Block operating in Man mode
Bad	<ul style="list-style-type: none"> Block out of service

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Out-of-Service	<ul style="list-style-type: none"> MODE_BLK.Target currently set to OOS

7.16.4 Block parameters

Flip-flop and Edge Trigger block

Parameter	Valid range/ Options	Default value	Description
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set MODE_BLK.Target to AUTO, CAS or RCAS
BLOCK_ERR	RO		Block errors, see Chapter 6.1.9
DISC_OP	0 – 5		Discrete operation selector <ul style="list-style-type: none"> ■ 0: SR flip-flop ■ 1: RS flip-flop ■ 2: D-.latch ■ 3: rising edge ■ 4: falling edge ■ 5: bi-directional edge
STATUS_OPTS			Status options <ul style="list-style-type: none"> ■ Use Uncertain as Good
IN_D1			Input value and status of discrete input 1
IN_D2			Input value and status of discrete input 2
SET_IN			Input value and status of set discrete input
RESET_IN			Input value and status of reset discrete input
OUT_D			Discrete value and status as a result of executing the block
UPDATE_EVT			This alert is generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
Legend: RO = Read Only			

7.17 Advanced Equations

The Advanced Equations block has been designed to support specific calculations relevant to process engineering.

In ControlCare Application Designer the Advanced Equations block is assigned the generic name **Device Tag-AEQU-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 7-73.

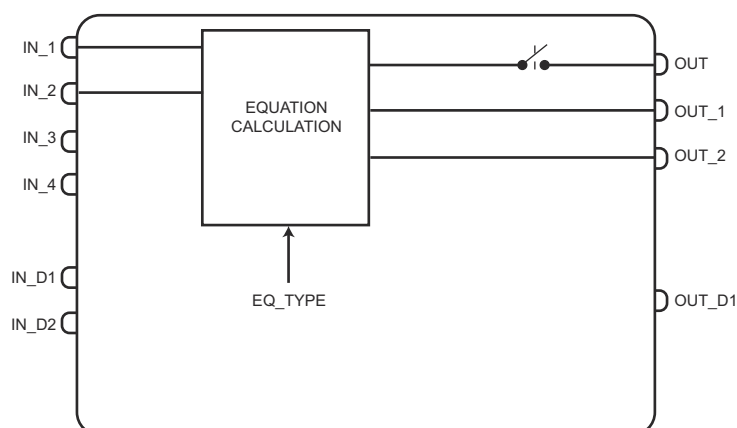


Fig. 7-73: Schematic diagram of Advanced Equations block

7.17.1 Functional description

The Advanced Equations block provides calculation of the following functions based on the value of the **IN_1** and in the case of dew point temperature **IN_2**:

- Natural logarithm, $\ln x$
- Logarithm to the base 10, $\log x$
- e to the power x, $\exp x$
- Dew point temperature

The type of equation used is selected by the parameter **EQ_TYPE**. The result of the calculation is published at **OUT** and in case of dew point temperature, **OUT_1** and **OUT_2**. All other input parameters in the block are not used, with the exception of **OUT_HI_LIM** and **OUT_LO_LIM**, which may be used to limit the output. The table below summarizes the configuration for the various options.

EQ_TYPE	Parameter	Description	Inputs	Outputs
0	$\ln x$	Natural logarithm	IN_1 = x	OUT = $\ln x$
1	$\log x$	Logarithm to the base 10	IN_1 = x	OUT = $\log x$
2	$\exp x$	e to the power x	IN_1 = x	OUT = $\exp x$
3	Dew point temperature	Dewpoint temperature °F	IN_1 = dry bulb temperature °F IN_2 = relative humidity %	OUT = dewpoint temperature OUT_1 = water vapour saturation pressure (psia) OUT_2 = water vapour pressure (pw)
255	Special			

7.17.2 Block configuration

The parameter **MODE_BLK.Target** must be set to Auto. If required, limits can be set on the OUT value with the parameters **OUT_HI_LIM** and **OUT_LO_LIM**.

7.17.3 Block operation

Mode

The block normally operates in "Auto" mode.

When **MODE_BLK.Target** is set to "Man", the block outputs **OUT**, **OUT_1** and **OUT_2** can be manipulated depending upon **EQ_TYPE**.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Operation can be checked on-line by viewing a number of parameters:

IN_1, IN_1	Value and status of corresponding input signal
OUT, OUT_1, OUT_2	Value and status of corresponding analog output signal

Status

The significance of the output status is indicated below.

Output status	Meaning
Good	■ Normal operation as configured
Bad	■ Block out of service

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Out-of-Service	■ MODE_BLK.Target currently set to OOS
Block Configuration Error	■ Abnormal result of calculation (+/- INF, NaN)

7.17.4 Block parameters

Advanced Equations block

Parameter	Valid range/ Options	Default value	Description
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set MODE_BLK.Target to AUTO, CAS or RCAS
BLOCK_ERR	RO		Block errors, see Chapter 6.1.9
EQ_TYPE	0-3, 255	0	Selection of equation to be used: ■ 0: ln x ■ 1: log x ■ 2: exp x ■ 3: dew point ■ 255: reserved (currently no function)
IN_1			■ EQ_TYPE = 0, 1, 2: input value for equation; ■ EQ_TYPE = 3: dry bulb temperature °F
IN_2			■ EQ_TYPE = 3: relative humidity %
IN_3, IN_4			Input values, not used
IN_D1, IN_D2			Discrete input values, not used
OUT_1			■ EQ_TYPE = 0, 1, 2: result of executing block; ■ EQ_TYPE = 3: dewpoint temperature °F
OUT_D1			Discrete output value, not used
OUT_2			■ EQ_TYPE = 3: saturation pressure water vapour (psia)
OUT_3			■ EQ_TYPE = 3: pressure water vapour (pw)
CT_VAL_1 - CT_VAL_6			Not used
CT_VAL_D1, CT_VAL_D2		0	Not used
OUT_HI_LIM			HI limit for OUT value
OUT_LO_LIM			LO limit for OUT value
BLOCK_ALM			Block alarms, see Chapter 2.9
Legend: RO = Read Only			

7.18 Density

The Density block has an algorithm to calculate density in various engineering units, e.g. °Plato, °Brix, TC and °INPM.

In ControlCare Application Designer the Density block is assigned the generic name **Device Tag-DENS-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 7-74.

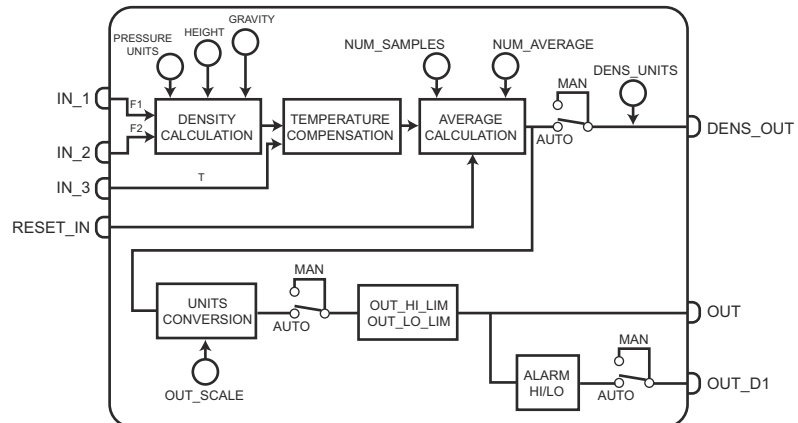


Fig. 7-74: Schematic diagram of Density block

7.18.1 Functional description

The Density block calculates the density of a liquid based on the pressure measured at two points within the tank with a known difference in height, **DEN_HEIGHT**.

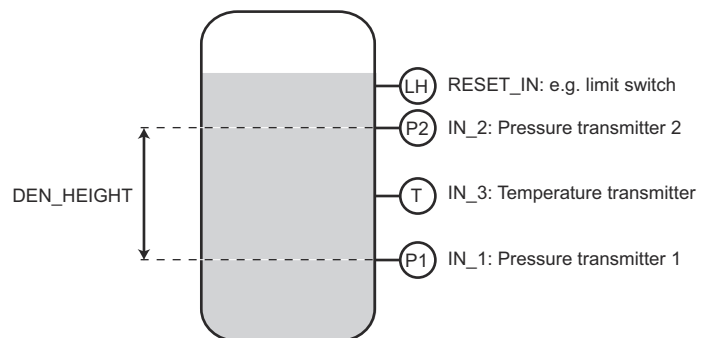


Fig. 7-75: Arrangement of pressure and temperature transmitters for density measurement

The input parameters are as follows:

- **IN_1** is the pressure measured by the lower transmitter
- **IN_2** is the pressure measured by the higher transmitter
- **IN_3** is the temperature in °C measured by the temperature transmitter

The calculation is made using the average from several pressure transmitter samples, whereby the number of the samples is determined by the **NUM_SAMPLES** parameter. When **RESET_IN** is true (= 1), the average calculation is reset. This allows the calculation to be automatically reset by a level limit switch when the vessel is filled and pressure transmitter is covered or by operator intervention from the HMI.

The density in g/cm^3 is calculated as follows:

$$D = \frac{f \times (P1 - P2)}{\text{DENHEIGHT} \times \text{GRAVITY}}$$

where: f is a conversion factor to transform the formula coefficients to the same units

$P1$ is the average pressure measured at **IN_1**

$P2$ is the average pressure measured at **IN_2**

DEN_HEIGHT is the distance between the two measuring points in the same units as the pressure measurement

GRAVITY is the acceleration due to gravity in m/s^2

The average pressure is calculated in a rolling buffer as follows:

$$P_i = \frac{\sum_{i=1}^n \text{IN}_i}{n}$$

where: P_i is the average pressure

IN_i is the pressure measured at **IN_i** at sample j

n is the number of samples **NUM_SAMPLES**

The temperature is compensated as part of the the density calculation. The output **DENS_OUT** is the compensated density in units of g/cm^3 . The **OUT** parameter is the compensated density in the engineering units selected by **EU_SEL** parameter.

The density block allows limits to be placed on the **OUT** output. If the density exceeds **HI_LIM** or drops below **LO_LIM** an alarm is indicated in **HI_ALM** or **LO_ALM** and the output **OUT_D** is set to true. The alarm is disabled when one or both of the limit parameters are set to $\pm \text{INF}$

When **RESET_IN** changes from false to true, the density block, including the previous density, the rolling buffer and all outputs of the block, is reset. The status is taken from the last cycle.

Valid pressure units

The following units can be selected for use by the block

Index	Pressure unit	HEIGHT unit
1130	Pa	m
1133	KPa	m
1132	MPa	m
1137	bar	cm
1138	millibar	cm
1139	torr	mm
1140	atm	cm
1141	psi	in
1144	g/cm^2	cm
1145	kg/cm^2	cm
1148	in H ₂ O	in
1147	in H ₂ O at 4 °C	in
1151	mm H ₂ O	mm
1150	mm H ₂ O at 4 °C	mm
1154	ft H ₂ O	ft
1156	in Hg	in
1158	mm Hg	mm

For an accurate density measurement, the distance between the two pressure transmitters must be measured with an accuracy of ± 1 mm. The corresponding number of decimal places must be used when entering the **HEIGHT** parameter in metres, centimetres, feet or inches.

Status propagation

The primary input status of **IN_1** and **IN_2** is propagated to the outputs. If the status becomes bad or it becomes uncertain and option “Use Uncertain as Good” in **STATUS_OPTS** is not set, the block mode will be forced to Man and the algorithm stops the calculation.

If the secondary input **IN_3** is unusable, the algorithm uses the last usable value and the output status will be Uncertain. A bad status in **RESET_IN** input does not stop the algorithm.

If target mode is Man then the output status is Good. Block configuration

The parameter **MODE_BLK.Target** must be set to Auto. If required, limits can be set on the OUT value with the parameters **OUT_HI_LIM** and **OUT_LO_LIM**.

7.18.2 Block configuration**Control strategy**

Fig. 7-76 shows a control strategy for an extract measurement in a beer tank. The input to the block is provided by three analog input blocks for the continuous temperature and pressure measurements and a multiple discrete input block connected to the level switch. A matched pair of pressure transmitters with a measuring range of 0 - 5000 mbar is to be used, the distance between the mounting points being 1750 mm.

The density is to be measured in °Plato, and a signal is to be given to the operator when the measured value drops below 3.00. To this end, the **OUT** value is limited, and the **OUT_D** signal is used to actuate a visual alert via a multiple discrete output block.

The connection to both the limit switch and the visual alarm is made with a ControlCare SFC432 I/O module located in Rack 1, Slot 0.

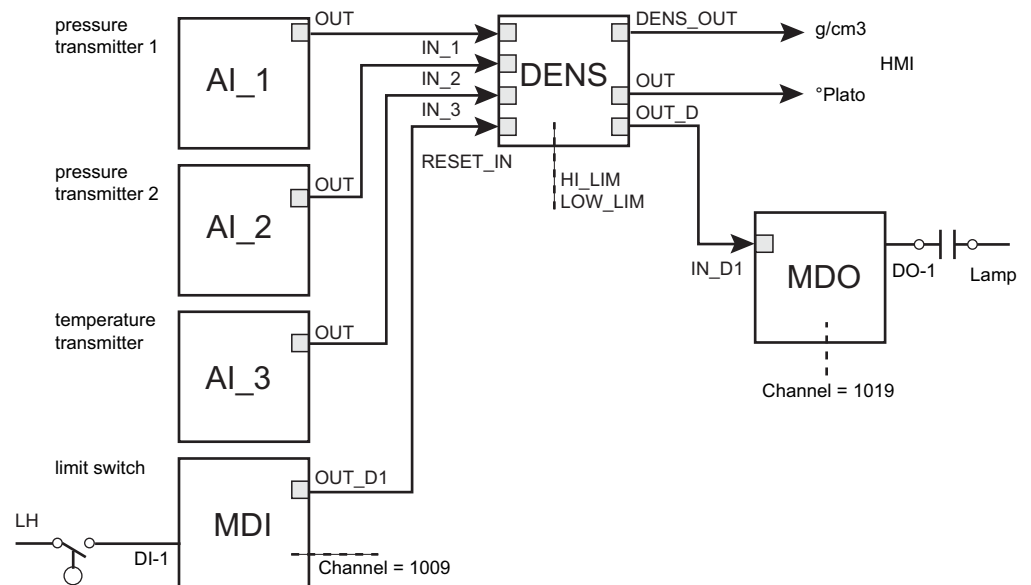


Fig. 7-76: Control strategy for extract measurement in beer

Block configuration

The MDO, AI, DENS and MDI blocks are configured as follows for the above application:

Parameter	Action	Option/Value
Block AI_1		
Block AI_2		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
XD_SCALE.EU_100	Input scaling, upper range value	0
XD_SCALE.EU_0	Input scaling, lower range value	5000
XD_SCALE.UNIT_INDEX	Input scaling, engineering units	mbar
OUT_SCALE.EU_100	Input scaling, upper range value	0
OUT_SCALE.EU_0	Input scaling, lower range value	5000
OUT_SCALE.UNIT_INDEX	Input scaling, engineering units	mbar

Parameter	Action	Option/Value
Block AI_3		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
XD_SCALE.EU_100	Input scaling, upper range value	-200
XD_SCALE.EU_0	Input scaling, lower range value	600
XD_SCALE.UNIT_INDEX	Input scaling, engineering units	°C
OUT_SCALE.EU_100	Input scaling, upper range value	-200
OUT_SCALE.EU_0	Input scaling, lower range value	600
OUT_SCALE.UNIT_INDEX	Input scaling, engineering units	°C
Block MDI		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
CHANNEL	Rack + Slot + Group + 9	1009
Block DENS		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
OUT_SCALE.EU_100	Input scaling, upper range value	20
OUT_SCALE.EU_0	Input scaling, lower range value	0
OUT_SCALE.UNIT_INDEX	Input scaling, engineering units (for HMI)	Plato Degree
EU_SEL	Select engineering units for OUT value	Plato Degree
PRESSURE_UNITS	Select pressure unit output by transmitters	mbar
DEN_HEIGHT	Distance between the pressure transmitters in the same units as the pressure measurement	175.0 (cm)
LO_PRI	Priority of LO_LIM alarm	1
LO_LIM	Value of lower limit in units of OUT	3
Block MDO		
MODE_BLK.Target	Set the target mode of the block to Auto	Auto
CHANNEL	Rack + Slot + Group + 9	1019

7.18.3 Block operation

Mode

The block normally operates in "Auto" mode.

When **MODE_BLK.Target** is set to "Man", the block outputs **OUT**, **OUT_1** and **OUT_2** can be manipulated depending upon **EQ_TYPE**.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Operation can be checked on-line by viewing a number of parameters:

IN_1, IN_1	Value and status of corresponding input signal
OUT, OUT_1, OUT_2	Value and status of corresponding analog output signal

Status

The significance of the output status is indicated below.

Output status	Meaning
Good	■ Normal operation as configured
Bad	■ Block out of service

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Out-of-Service	■ MODE_BLK.Target currently set to OOS

7.18.4 Block parameters

Density block

Parameter	Valid range/ Options	Default value	Description
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set MODE_BLK.Target to AUTO, CAS or RCAS
BLOCK_ERR	RO		Block errors, see Chapter 6.1.9
OUT		0	Value output after execution of the block
OUT_SCALE	0 - 3	0	Scaling of output value
EU_SEL			Select engineering units for OUT value <ul style="list-style-type: none"> ■ 0: Plato Degree ■ 1: Brix ■ 2: TC ■ 3: INPM
GRANT_DENY		0	Access options, see Chapter 2.8.1
STATUS_OPTS			Status options <ul style="list-style-type: none"> ■ Use Uncertain as Good
IN_1			Input value and status of pressure transmitter 1
IN_2			Input value and status of pressure transmitter 2
IN_3			Input value and status of temperature transmitter (°C)
PRESSURE_UNITS	see table	g/cm ²	Selection of pressure unit used by pressure transmitters 1 and 2
DEN_HEIGHT	positive	1000	Distance between the pressure transmitters in the same units as the pressure measurement, see table
GRAVITY	positive	9.80665	Acceleration due to gravity in m/s ²
NUM_SAMPLES	1 - 1000	10	Number of samples per average
NUM_AVERAGES	1 - 30	10	Number of averages in rolling buffer
DENS_OUT	RO		Density in g/cm ² compensated for temperature
DENS_UNITS		g/cm ²	Units of DENS_OUT for HMI (always g/cm ²)
OUT_D			Discrete output indicating limit alarm
RESET_IN	RO		Discrete input for resetting the density calculation and buffer
OUT_HI_LIM		100	Upper limit for OUT value, above which the status is "Limited"
OUT_LO_LIM		0	Lower limit for OUT value, below which the status is "Limited"
UPDATE_EVT			Alert generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
ALARM_SUM			Alarm summary, see Chapter 2.9
ACK_OPTION	0, 1	0	See Chapter 2.9 <ul style="list-style-type: none"> ■ 0: Auto ACK Disable ■ 1: Auto ACK Enable
ALARM_HYS	0 to 50 %	0.5%	Alarm hysteresis in % of OUT_SCALE, see Chapter 2.9
HI_PRI	0 to 15		Priority of the HI alarm, see Chapter 2.9
HI_LIM		+INF	Setting of HI alarm in engineering units, see Chapter 2.9.
LO_PRI	0 to 15		Priority of the LO alarm, see Chapter 2.9.
LO_LIM		-INF	Setting of LO alarm in engineering units, see Chapter 2.9.
HI_ALM			Status of HI alarm and its associated time stamp.
LO_ALM			Status of LO alarm and its associated time stamp.
Legend: RO = Read Only			

7.19 Hybrid with Analog I/O

The Hybrid Analog I/O block allows freely programmable logical sequences and calculations to be performed with in a strategy based on analog input values. After execution of the block, the results are presented as analog output values.

In ControlCare Application Designer the Hybrid Analog I/O block is assigned the generic name **Device Tag-HY-MA-IO-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 7-77.

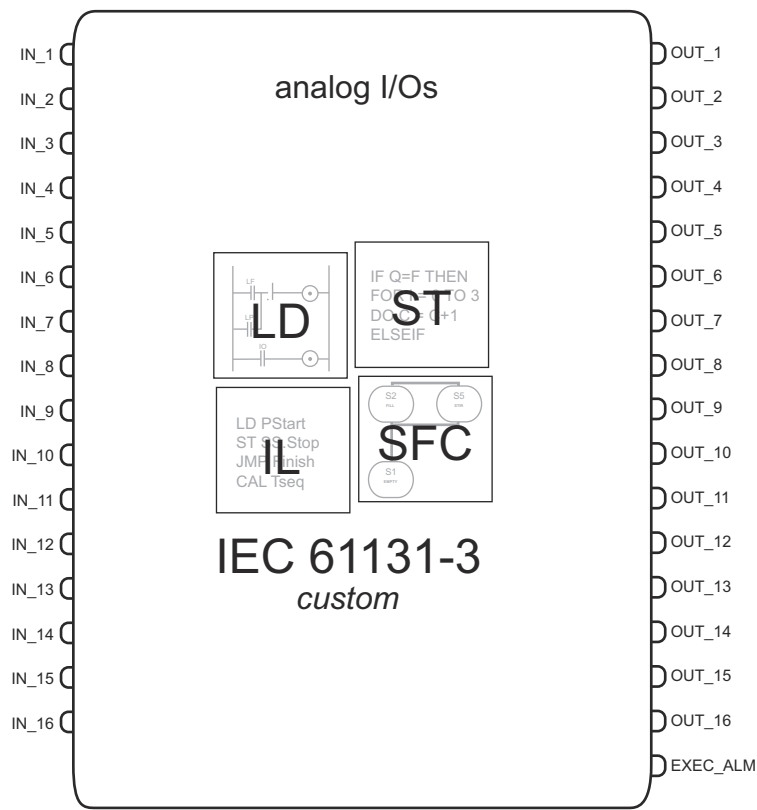


Fig. 7-77: Schematic diagram of Analog I/O hybrid block

7.19.1 Functional description

The Hybrid Analog I/O block is created in Application Designer and programmed in OpenPCS. The programming of the block is described in the IEC 61131-3 Tutorials BA038S/04/en (Ladder Logic) and BA056S/04/en (Structured Text).

The block allows up to sixteen analog signals **IN_1** to **IN_16** to be used as inputs to a control routine that has been programmed in a IEC 61131-3 function block language. The result of executing the routine is output at up to eight outputs **OUT_1** to **OUT_16**.

Note

- For ControlCare Versions <2.04.xx, the Analog Hybrid block has only eight inputs and outputs



Block execution

The parameter **EXEC_TIME_TARGET** allows a target execution time to be set for the block, default value is 10 ms. This is normally set after observation of the actual execution time in **EXEC_TIME_ACTUAL**. Details of the block execution can be found by expanding the parameter **EXEC_DETAILS**:

Parameter	Meaning
EXECUTIONS	No of executions of the block to since the block was put into the current mode
EXCEEDINGS	No of times the execution of the block took longer than target execution time
RATIO	Ratio of executions within permitted time to number exceeding permitted time
START_DATE	Date and time when block was last set to Auto (not used)
EXEC_ERROR	IEC 61311 execution error; "0" if no error, see description
ALM_SEL	Selects which IEC 61131 execution alarm causes EXEC_ERROR to be set <ul style="list-style-type: none"> ■ Select from the list of EXEC_ERROR options offered ■ Multiple choices are allowed

The **EXEC_ERROR** parameter indicates the current execution error as follows:

Error	Bit value	Description
None	0x0000	IEC 61131 program is running as expected.
OOS	0x0001	IEC 61131 program is not running because the target block mode is OOS
Runtime stopped	0x0002	IEC 61131 runtime state is not running <ul style="list-style-type: none"> – Set runtime state to running in OpenPCS.
Program error	0x0004	IEC 61131 program is not available. <ul style="list-style-type: none"> – Has to be downloaded in OpenPCS. – Interrupt name of the IEC1131 program does not match the hybrid block tag.
Time exceeded	0x0008	Execution of the IEC1131 program has taken longer than the permitted three times in a row.
Endless loop	0x0010	IEC 61131 program has been in an endless loop and was stopped
Trigger error	0x0020	The interrupt trigger is not valid
HYB table error	0x0040	The hybrid FB IEC program mapping table is invalid
Link pointer error	0x0080	No IO links configured or at least one of the pointers to the linkable IO's is not valid <ul style="list-style-type: none"> ■ Make/check the links in the I/O mapping tool
PI error	0x0100	Not used
PB error	0x0200	Not used
LOC error	0x0400	Not used
MB error	0x0800	Not used

EXEC_ALM is set if there is a program execution error, and clears automatically when the error no longer exists.

Status propagation

The parameter **HYB_STATUS_OPS** determines the status to be used with the output parameters. When the option "Set Outputs to Good Non Cascade" is selected, all outputs carry the status "GoodNonCascade:NonSpecific:NotLimited" when the block is in Auto, otherwise they are Bad. If no option is selected, the status is bad by default but can be handled within the IEC 61131 program.

7.19.2 Block operation

Mode

The block normally operates in "Auto" mode. If the IEC 61131 program fails to execute, the block mode is automatically set to OOS (Out of Service).

When **MODE_BLK.Target** is set to "Man", the IEC 61131 program is not executed and the last good value with status is retained. The block outputs can be manipulated in OpenPCS.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Operation can be checked on-line by viewing a number of parameters:

IN_1 to IN_8	Value and status of corresponding input signal
OUT_1 to OUT_8	Value and status of corresponding analog output signal

Status

The significance of the output status is indicated below.

Output status	Meaning
Good	<ul style="list-style-type: none"> Normal operation as configured
Bad	<ul style="list-style-type: none"> Block out of servic IEC 61131 program not running, see EXEC_ERROR in EXEC_DETAILS

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Out-of-Service	<ul style="list-style-type: none"> MODE_BLK.Target currently set to OOS IEC 61131 execution error, see EXEC_ERROR in EXEC_DETAILS

7.19.3 Block parameters

Analog Hybrid block

Parameter	Valid range/ Options	Default value	Description
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set MODE_BLK.Target to AUTO
BLOCK_ERR	RO		Block errors, see Chapter 6.1.9
PI_POINTER			Indicates IEC 61131 index, 0 when not running
CONTENTS_REV			Not used
EXEC_TIME_TARGET	0 - 300	10	Permitted execution time of block in milliseconds – Must be consistent with value used for FF schedule calculation.
EXEC_TIME_ACTUAL	RO		Current execution time of block in milliseconds
IO_RESPONSE_TIME	RO		Worst case IO response time which includes the interval of the input scan plus the measured time for reading the inputs, executing the IEC61131 program and writing the outputs.
HYB_STATUS_OPS		0	Hybrid block status options <ul style="list-style-type: none"> 0: Bad, status handled by IEC 61131 program 1: Set Outputs to Good Non Cascade
EXEC_DETAILS			Details of block execution, see Chapter 7.19.1
BLK_ALM			Block alarms, see Chapter 2.9
EXEC_ALM			Indicates presence of programm execution error
IN_1 ... IN_16			Value and status of IN_1 ... IN_16
OUT_1 ... OUT_16			Value and status of OUT_1 ... OUT_16 as result of executing block
Legend: RO = Read Only			

7.20 Hybrid with Discrete I/O

The Hybrid Discrete I/O block allows freely programmable logical sequences and calculations to be performed with in a strategy based on analog input values. After execution of the block, the results are presented as analog output values.

In ControlCare Application Designer the Hybrid Discrete I/O block is assigned the generic name **Device Tag-HY-MD-IO-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 7-78.

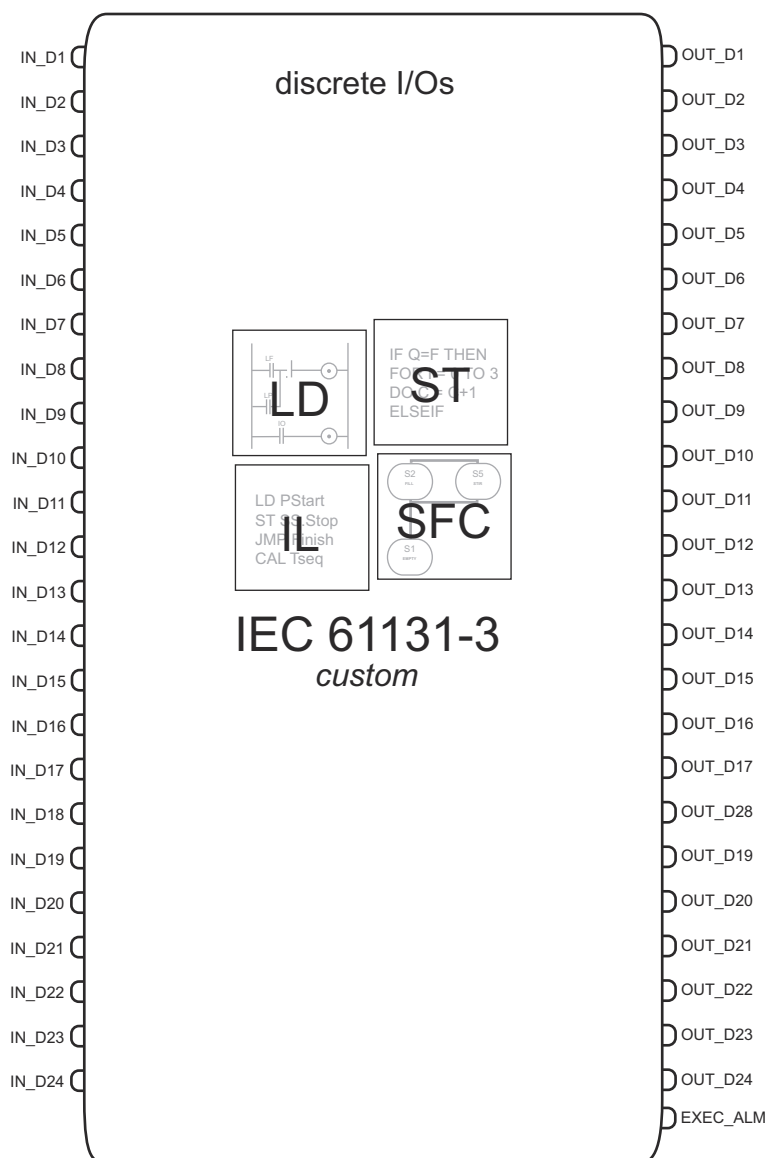


Fig. 7-78: Schematic diagram of Discrete I/O hybrid block

7.20.1 Functional description

The Hybrid Discrete I/O block is created in Application Designer and programmed in OpenPCS. The programming of the block is described in the IEC 61131-3 Tutorials BA038S/04/en (Ladder Logic) and BA056S/04/en (Structured Text).

The block allows up to 24 discrete signals **IN_D1** to **IN_D24** to be used as inputs to a control routine that has been programmed in a IEC 61131-3 function block language. The result of executing the routine is output at up to 24 discrete outputs **OUT_D1** to **OUT_D24**.

Note

- For ControlCare Versions <2.04.xx, the Discrete Hybrid block has only 16 inputs and outputs

Block execution



The parameter **EXEC_TIME_TARGET** allows a target execution time to be set for the block, default value is 10 ms. This is normally set after observation of the actual execution time in **EXEC_TIME_ACTUAL**. Details of the block execution can be found by expanding the parameter **EXEC_DETAILS**:

Parameter	Meaning
EXECUTIONS	No of executions of the block to since the block was put into the current mode
EXCEEDINGS	No of times the execution of the block took longer than target execution time
RATIO	Ratio of executions within permitted time to number exceeding permitted time
START_DATE	Date and time when block was last set to Auto (not used)
EXEC_ERROR	IEC 61311 execution error; "0" if no error, see description
ALM_SEL	Selects which IEC 61131 execution alarm causes EXEC_ERROR to be set <ul style="list-style-type: none"> ■ Select from the list of EXEC_ERROR options offered ■ Multiple choices are allowed

The **EXEC_ERROR** parameter indicates the current execution error as follows:

Error	Bit value	Description
None	0x0000	IEC 61131 program is running as expected.
OOS	0x0001	IEC 61131 program is not running because the target block mode is OOS
Runtime stopped	0x0002	IEC 61131 runtime state is not running <ul style="list-style-type: none"> – Set runtime state to running in OpenPCS.
Program error	0x0004	IEC 61131 program is not available. <ul style="list-style-type: none"> – Has to be downloaded in OpenPCS. – Interrupt name of the IEC1131 program does not match the hybrid block tag.
Time exceeded	0x0008	Execution of the IEC1131 program has taken longer than the permitted three times in a row.
Endless loop	0x0010	IEC 61131 program has been in an endless loop and was stopped
Trigger error	0x0020	The interrupt trigger is not valid
HYB table error	0x0040	The hybrid FB IEC program mapping table is invalid
Link pointer error	0x0080	No IO links configured or at least one of the pointers to the linkable IO's is not valid <ul style="list-style-type: none"> ■ Make/check the links in the I/O mapping tool
PI error	0x0100	Not used
PB error	0x0200	Not used
LOC error	0x0400	Not used
MB error	0x0800	Not used

EXEC_ALM is set if there is a program execution error, and clears automatically when the error no longer exists.

Status propagation

The parameter **HYB_STATUS_OPS** determines the status to be used with the output parameters. When the option "Set Outputs to Good Non Cascade" is selected, all outputs carry the status "GoodNonCascade:NonSpecific:NotLimited" when the block is in Auto, otherwise they are Bad. If no option is selected, the status is bad by default but can be handled within the IEC 61131 program.

7.20.2 Block operation

Mode

The block normally operates in "Auto" mode. If the IEC 61131 program fails to execute, the block mode is automatically set to OOS (Out of Service).

When **MODE_BLK.Target** is set to "Man", the IEC 61131 program is not executed and the last good value with status is retained. The block outputs can be manipulated in OpenPCS.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Operation can be checked on-line by viewing a number of parameters:

IN_D1 to IN_D24	Value and status of corresponding input signal
OUT_D1 to OUT_D24	Value and status of corresponding analog output signal

Status

The significance of the output status is indicated below.

Output status	Meaning
Good	<ul style="list-style-type: none"> Normal operation as configured
Bad	<ul style="list-style-type: none"> Block out of service IEC 61131 program not running, see EXEC_ERROR in EXEC_DETAILS

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Out-of-Service	<ul style="list-style-type: none"> MODE_BLK.Target currently set to OOS IEC 61131 execution error, see EXEC_ERROR in EXEC_DETAILS

7.20.3 Block parameters

Discrete Hybrid block

Parameter	Valid range/ Options	Default value	Description
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set MODE_BLK.Target to AUTO
BLOCK_ERR	RO		Block errors, see Chapter 6.1.9
PI_POINTER			Indicates IEC 61131 index, 0 when not running
CONTENTS_REV			Not used
EXEC_TIME_TARGET	0 - 300	10	Permitted execution time of block in milliseconds – Must be consistent with value used for FF schedule calculation.
EXEC_TIME_ACTUAL	RO		Current execution time of block in milliseconds
IO_RESPONSE_TIME	RO		Worst case IO response time which includes the interval of the input scan plus the measured time for reading the inputs, executing the IEC61131 program and writing the outputs.
HYB_STATUS_OPS		0	Hybrid block status options <ul style="list-style-type: none"> 0: Bad, status handled by IEC 61131 program 1: Set Outputs to Good Non Cascade
EXEC_DETAILS			Details of block execution, see Chapter 7.19.1
BLK_ALM			Block alarms, see Chapter 2.9
EXEC_ALM			Indicates presence of programm execution error
IN_D1 ... IN_D24			Value and status of IN_D1 ... IN_D24
OUT_D1 ... OUT_D24			Value and status of OUT_D1 ... OUT_D24 as result of executing block
Legend: RO = Read Only			

7.21 Hybrid with Embedded I/O

The Hybrid Embedded I/O block allows freely programmable logical sequences and calculations to be performed with in a strategy based on analog and discrete input values. After execution of the block, the results are presented as analog and discrete output values.

In ControlCare Application Designer the Hybrid Discrete I/O block is assigned the generic name **Device Tag-HY-EMB-IO-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 7-79.

Note

- The Hybrid Embedded I/O block replaces the Hybrid Mixed I/O block of previous ControlCare versions (< 2.04.xx, Device Tag-HY-MAD-IO-n). On updating a project, any mixed block is automatically updated to an embedded block.

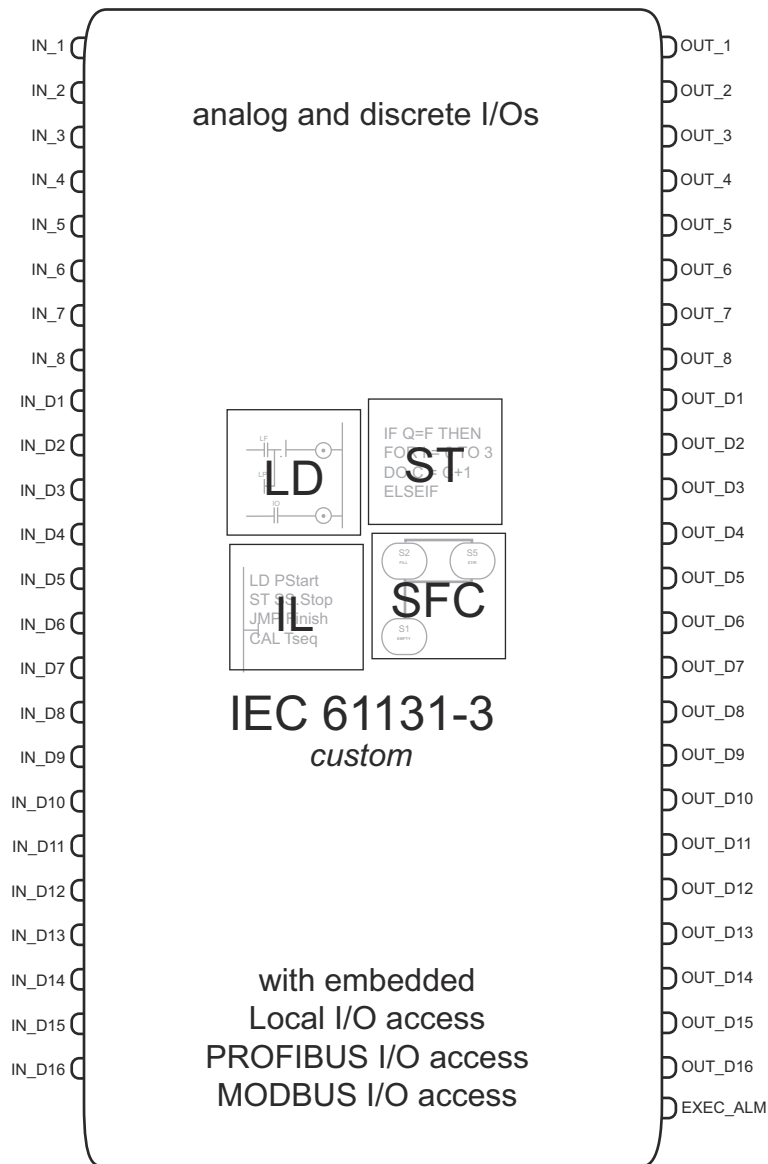


Fig. 7-79: Schematic diagram of Embedded I/O hybrid block

7.21.1 Functional description

The Hybrid Embedded I/O block is created in Application Designer and programmed in OpenPCS. The configuration and programming of the block is described in the I/O mapping tool Tutorial BA032S/04/en.

The block allows up to eight analog signals **IN_1** to **IN_8** and sixteen discrete signals **IN_D1** to **IN_D16** to be used as inputs to a control routine that has been programmed in a IEC 61131-3 function block language. The result of executing the routine is output at up to eight analog outputs **OUT_1** to **OUT_8** and sixteen discrete outputs **OUT_D1** to **OUT_D16**.

The parameters **FS_VALUE_DISCRETE** and **FS_VALUE_ANALOG** allow all local and linkable outputs to be forced to a universal failsafe discrete or analog value when the block fails.

Block execution

The parameter **EXEC_TIME_TARGET** allows a target execution time to be set for the block, default value is 10 ms. This is normally set after observation of the actual execution time in **EXEC_TIME_ACTUAL**. Details of the block execution can be found by expanding the parameter **EXEC_DETAILS**:

Parameter	Meaning
EXECUTIONS	No of executions of the block to since the block was put into the current mode
EXCEEDINGS	No of times the execution of the block took longer than target execution time
RATIO	Ratio of executions within permitted time to number exceeding permitted time
START_DATE	Date and time when block was last set to Auto (not used)
EXEC_ERROR	IEC 61311 execution error; "0" if no error, see description
ALM_SEL	Selects which IEC 61131 execution alarm causes EXEC_ERROR to be set <ul style="list-style-type: none"> ■ Select from the list of EXEC_ERROR options offered ■ Multiple choices are allowed

The **EXEC_ERROR** parameter indicates the current execution error as follows:

Error	Bit value	Description
None	0x0000	IEC 61131 program is running as expected.
OOS	0x0001	IEC 61131 program is not running because the target block mode is OOS
Runtime stopped	0x0002	IEC 61131 runtime state is not running <ul style="list-style-type: none"> – Set runtime state to running in OpenPCS.
Program error	0x0004	IEC 61131 program is not available. <ul style="list-style-type: none"> – Has to be downloaded in OpenPCS. – Interrupt name of the IEC1131 program does not match the hybrid block tag.
Time exceeded	0x0008	Execution of the IEC1131 program has taken longer than the permitted three times in a row.
Endless loop	0x0010	IEC 61131 program has been in an endless loop and was stopped
Trigger error	0x0020	The interrupt trigger is not valid
HYB table error	0x0040	The hybrid FB IEC program mapping table is invalid
Link pointer error	0x0080	No IO links configured or at least one of the pointers to the linkable IO's is not valid <ul style="list-style-type: none"> ■ Make/check the links in the I/O mapping tool
PI error	0x0100	Process image for local and PROFIBUS IO's is invalid
PB error	0x0200	At least one of the assigned PROFIBUS points has a problem <ul style="list-style-type: none"> ■ Check in the PROFIBUS Configurator and PROFIBUS Mapping Tool
LOC error	0x0400	At least one of the assigned Local I/O points has a problem <ul style="list-style-type: none"> ■ Check in I/O Mapping Tool
MB error	0x0800	At least one of the assigned Modbus points has a problem <ul style="list-style-type: none"> ■ Check in I/O Mapping Tool

EXEC_ALM is set if there is a program execution error, and clears automatically when the error no longer exists.

Status propagation

The parameter **HYB_STATUS_OPS** determines the status to be used with the output parameters. When the option "Set Outputs to Good Non Cascade" is selected, all outputs carry the status "GoodNonCascade:NonSpecific:NotLimited" when the block is in Auto, otherwise they are Bad. If no option is selected, the status is bad by default but can be handled within the IEC 61131 program.

7.21.2 Block operation

Mode

The block normally operates in "Auto" mode. If the IEC 61131 program fails to execute, the block mode is automatically set to OOS (Out of Service).

When **MODE_BLK.Target** is set to "Man", the IEC 61131 program is not executed and the last good value with status is retained. The block outputs can be manipulated in OpenPCS.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

Operation can be checked on-line by viewing a number of parameters:

IN_1 to IN_8	Value and status of corresponding analog input signal
IN_D1 to IN_D16	Value and status of corresponding discrete input signal
OUT_1 to OUT_8	Value and status of corresponding analog output signal
OUT_D1 to OUT_D16	Value and status of corresponding discrete output signal
LOCAL_IO_STATUS	Indicates read/write errors for local IOs assigned to the block
PROFIBUS_IO_STATUS	Indicates read/write errors for PROFIBUS IOs assigned to the block
LOCAL_IO_STAT_R0 ... LOCAL_IO_STAT_R14	Indicates the status of each local IO group

Possible read/write statuses in **LOCAL_IO_STATUS** and **PROFIBUS_IO_STATUS** are:

Error	Bit value	Description
Inputs not used	0x0001	No local inputs assigned to this block or IO configuration not yet downloaded to IEC 61131 runtime
Outputs not used	0x0002	No local outputs assigned to this block or IO configuration not yet downloaded to IEC 61131 runtime
Read error	0x0004	One of the assigned PROFIBUS inputs has a read error
Write error	0x0008	One of the assigned PROFIBUS outputs has a write error
Inputs OK	0x0010	All assigned inputs of the block are in order
Outputs OK	0x0020	All assigned outputs of the block are in order
OpenPCS Dwl needed	0x0040	An OpenPCS download is required to initialize the IO tables

Possible read/write statuses in **LOCAL_IO_STAT_R0 ... LOCAL_IO_STAT_R14** are:

Error	Bit value	Description
Not used	0x01	Group not assigned to this block
OK	0x02	Group assigned and working as expected
Wrong PI	0x04	Process image does not match the IO configuration
No config	0x08	Group not configured in OpenPCS
Invalid pData	0x10	No memory in the local IO PI in HeapX
Wrong module	0x20	Module present does not correspond to that configured
Multiple used	0x40	Output group used by more than one output block
Unplugged	0x80	IO module assigned to this group is no longer present

Status

The significance of the output status is indicated below.

Output status	Meaning
Good	<ul style="list-style-type: none"> Normal operation as configured
Bad	<ul style="list-style-type: none"> Block out of service IEC 61131 program not running, see EXEC_ERROR in EXEC_DETAILS

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Out-of-Service	<ul style="list-style-type: none"> MODE_BLK.Target currently set to OOS IEC 61131 execution error, see EXEC_ERROR in EXEC_DETAILS

7.21.3 Block parameters

Parameter	Valid range/ Options	Default value	Description
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set MODE_BLK.Target to AUTO
BLOCK_ERR	RO		Block errors, see Chapter 6.1.9
PI_POINTER			Indicates IEC 61131 index, 0 when not running
CONTENTS_REV			Not used
EXEC_TIME_TARGET	0 - 300	10	Permitted execution time of block in milliseconds – Must be consistent with value used for FF schedule calculation.
EXEC_TIME_ACTUAL	RO		Current execution time of block in milliseconds
IO_RESPONSE_TIME	RO		Worst case IO response time which includes the interval of the input scan plus the measured time for reading the inputs, executing the IEC61131 program and writing the outputs.
HYB_STATUS_OPS		0	Hybrid block status options <ul style="list-style-type: none"> ■ 0: Bad, status handled by IEC 61131 program ■ 1: Set Outputs to Good Non Cascade
EXEC_DETAILS			Details of block execution, see Chapter 7.21.1
BLK_ALM			Block alarms, see Chapter 2.9
EXEC_ALM			Indicates presence of programm execution error
LOCAL_IO_STATUS			Indicates read/write errors for local IOs assigned to the block. <ul style="list-style-type: none"> ■ Not used: no local IO modules assigned to block ■ OK: local IO assigned and working properly ■ Other: see Chapter 7.21.2
PROFIBUS_IO_STATUS			Indicates read/write errors for PROFIBUS IOs assigned to the block. <ul style="list-style-type: none"> ■ Not used: no PROFIBUS IO assigned to block ■ OK: PROFIBUS IO assigned and working properly ■ Other: see Chapter 7.21.2
LOC_IO			Indicates how the IO modules have been assigned to the block. <ul style="list-style-type: none"> ■ Rx: Local I/O rack number ■ SyGz: Slot(s) and group(s) in associated rack that have been assigned to the block ■ Not used: no points are used from associated rack
PB_IN			Indicates which sections of the Profibus input image have been assigned to this block. <ul style="list-style-type: none"> ■ BEGIN_x: start of the image section ■ END_x: End of the image section"
PB_OUT			Indicates which sections of the Profibus output image have been assigned to this block. <ul style="list-style-type: none"> ■ BEGIN_x: start of the image section ■ END_x: End of the image section"
FS_VALUE_DISCRETE			Failsafe value for all local and linkable discrete outputs
FS_VALUE_ANALOG			Failsafe value for all local and linkable analogue outputs
IN_1 ... IN_8			Value and status of IN_1 ... IN_8
OUT_1 ... OUT_8			Value and status of OUT_1 ... OUT_8 as result of executing block
IN_D1 ... IN_D16			Value and status of IN_D1 ... IN_D16
OUT_D1 ... OUT_D16			Value and status of OUT_D1 ... out_d!& as result of executing block
LOCAL_IO_STAT_R0 ... LOCAL_IO_STAT_R14			Indicates the status of each local IO group <ul style="list-style-type: none"> ■ See Chapter 2.21.2
Legend: RO = Read Only			

8 Output Blocks

8.1 Analog Output

The Analog Output block receives a setpoint value from an upstream function block or higher level process control system and, after scaling, outputs a signal to a downstream transducer block through an internal channel reference. It is used by instruments that act as output elements in a control loop such as valves, actuators, positioners and the SFC446 analog output module.

In ControlCare Application Designer the Analog Output block is assigned the generic name **Device Tag-AO-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 8-1.

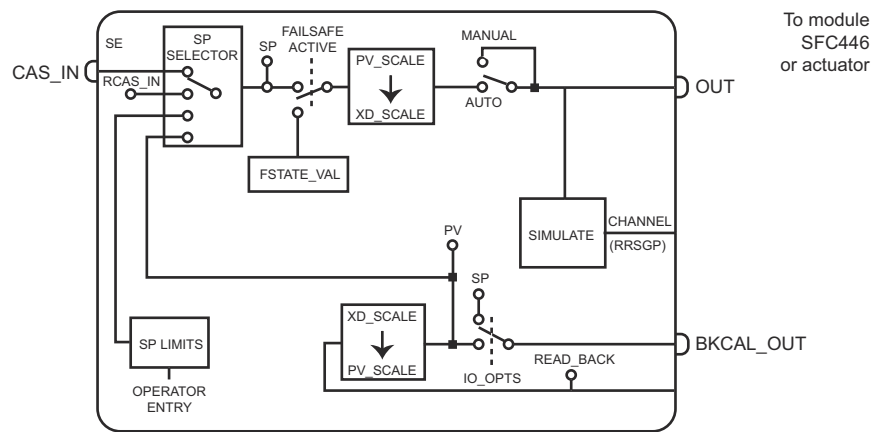


Fig. 8-1: Schematic diagram of Analog Output block

8.1.1 Functional description

Setpoint

Depending upon the setting of the **MODE_BLK.Target** parameter, the Analog Output block receives its setpoint value from the operator, an upstream block or a remote process control system.

MODE_BLK.Target	Setpoint	Remarks
Auto	SP	Setpoint value is constant and is entered manually by the operator.
Cas	CAS_IN	Setpoint value originates from an upstream block
Rcas	RCAS_IN	Setpoint value originates from a supervisory host

On a change in setpoint, the parameters **SP_RATE_DN** and **SP_RATE_UP** determine the downward and upward ramp rate at which a change is made from the old to the new value. If the ramp rate is set to zero, there is a step change in the setpoint.

SP_HI_LIM and **SP_LO_LIM** determine the valid range of setpoint operator entries that can be used with the block. They apply in Auto only, unless the option "Obey SP limits if CAS or RCAS" is selected in **CONTROL_OPTS**. If the setpoint exceeds the limits, the corresponding value **SP_HI_LIM** or **SP_LO_LIM** is used as setpoint and the output status quality becomes "Limited".

Scaling

PV_SCALE is used to convert the input signal (setpoint) to percent of span.

- EU_100 is the upper range limit of the input signal, e.g. 100
- EU_0 is the lower range limit of the input signal, e.g. 0
- UNIT_INDEX is the unit of the input signal, e.g. %
- DECIMAL is the number of figures behind the decimal point for displays

XD_SCALE is used to convert percent of span to the engineering units used by the hardware.

- EU_100 is the upper range limit of the output signal, e.g. 20
- EU_0 is the lower range limit of the output signal, e.g. 4
- UNIT_INDEX is the unit of the output signal, e.g. mA
- DECIMAL is the number of figures behind the decimal point for displays

This allows portions of the setpoint span to cause full span movement of the output. If the "Increase to close" option is selected in **IO_OPTS**, the output signal is inverted and a rise in setpoint causes a fall in output value.

Output

When the block is operating in Cas, Rcas or Auto mode, the value obtained after block execution is placed in **OUT**. The same value is transmitted to the transducer through an internal channel reference given by the **CHANNEL** parameter, see Chapter 5.1.

If the transducer hardware supports a readback value, such as valve position, that value is run backwards through the XD scaling to act as the PV for the block. If not supported, **READBACK** is generated from **OUT**.

The current setpoint value and status is communicated to an upstream block by the **BKCAL_OUT** parameter. If the option "PV for BKCAL_OUT" is selected in **IO_OPTS**, then the PV will be transmitted instead of the setpoint.

When the block is operating in Rcas mode, **RCAS_OUT** carries the block setpoint and status after ramping for output to a supervisory host as back calculation and to allow action to be taken under limiting conditions or mode change.

Fault state

By default, the last valid value is held if a fault state is detected.

If the "Fault state to value" option is selected in **IO_OPTS**, **OUT** will assume the value entered in **FSTATE_VAL** when a fault condition exists longer than defined in the parameter **FSTATE_TIME** or when the option "Fault State" is selected the the resource block parameter **SET_FSTATE**. This ensures that the hardware connected to the block fails to safe, see Chapter 2.7.

When the option "Fault state restart" in **IO_OPTS** is selected, **FSTATE_VAL** will also be used on restart.

Mode shedding

SHED_OPT allows the behaviour of the block to be determined after a change in mode, e.g. forced by the system or through operator intervention. "Normal return" options allow the block to recover automatically to target mode, for "No return" options, the operator must enter the target mode manually.

Setpoint tracking

IO_OPTS also allows the setpoint to track PV or an input value when the the target mode is Auto and this changes to Man or LO. Possible behaviour is "SP tracks PV when Man", "SP tracks PV when LO", "SP tracks RCAS or CAS when Man or LO"

Simulation

The block supports simulation, see Chapter 2.10. The parameter group **SIMULATE** is used to specify the output value (setpoint value and status), transducer value (PV value and status) and to enable/disable simulation, provided the associated jumper on the I/O board is set, see Chapter 3.2. When simulation is active in the Analog Output block, **BLOCK_ERROR** flags "Simulation Active".

Status propagation

The status of OUT can assume the following values:

GoodCascade	The output value OUT is valid and can be used for further processing
Bad	The output value OUT is invalid. <ul style="list-style-type: none"> ■ Operating mode OOS ■ Input status bad

If the "Propagate Fault Bkwd" option in **STATUS_OPTS** is selected, the **BKCAL_OUT** and **RCAS_OUT** status carries the current **OUT** status to an upstream block or supervisory system.

8.1.2 Output configuration for SFC446

The AO block uses the **OUT** value to force electrical signals as shown in Tables 8-1 to 8-3 below. The type of output is determined by the wiring (current or voltage) and DIP switch (Range end value 5 DC or 10 VDC), see BA021S/04/en Chapter 3.5.1

Current output

The parameters **XD_SCALE.EU_100** and **XD_SCALE.EU_0** set the range end values of the current signal output of the module. This is normally a standard 0 mA to 20 mA/4 mA to 20 mA signal, but other ranges within these limits are possible. The current output is independent of the setting of the Voltage Range DIP switch on the module and is directly proportional to the **OUT** value.

XD_SCALE	Range 0 mA to 20 mA		Range 4 mA to 20 mA	
	Value	Output	Value	Output
EU_100	20	20 mA	20	20 mA
EU_0	0	0 mA	4	4 mA
Unit	mA	–	mA	–

Tab. 8-1: Current output of SFC446 module as a function of the XD_SCALE settings

Voltage output DIP switch OFF = range end 5V

The parameters **XD_SCALE.EU_100** and **XD_SCALE.EU_0** set the range end values of the voltage signal output of the module. This is normally a standard 1V to 5 V/–5 V to +5 V signal, but other ranges within these limits are possible. The voltage output is dependent of the setting of the Voltage Range DIP switch on the module and is directly proportional to the **OUT** value.

XD_SCALE	Range 1 V to 5 V		Range –5 V to +5 V	
	Value	Output	Value	Output
EU_100	5	5 VDC	5	5 VDC
EU_0	1	1 VDC	–5	–5 VDC
Unit	V	–	V	–

Tab. 8-2: Voltage output of SFC446 module with DIP switch OFF as a function of the XD_SCALE settings

Voltage output DIP switch ON = range end 10V

The parameters **XD_SCALE.EU_100** and **XD_SCALE.EU_0** set the range end values of the voltage signal output of the module. This is normally a standard 2V to 10 V/–10 V to +10 V signal, but other ranges within these limits are possible. The voltage output is dependent of the setting of the Voltage Range DIP switch on the module and is directly proportional to the **OUT** value. The readback and **OUT** value reflect the **XD_SCALE** entries, not the voltage that can be measured at the output

XD_SCALE	Range 2 V to 10 V		Range –10V to +10V	
	Value	Output	Value	Output
EU_100	5	10 VDC	5	10 VDC
EU_0	1	2 VDC	–5	–10 VDC
Unit	V	–	V	–

Tab. 8-3: Voltage output of SFC446 module with DIP switch ON as a function of the XD_SCALE settings

8.1.3 Block configuration

Fig. 8-2 shows an example of basic closed-loop control which uses the Analog Output block. A measurement of flow upstream of a valve ensures a constant flow rate downstream of a valve, by causing the valve to open and close. The valve is operated by an analog electrical positioner which operates across a span of 4 mA to 20 mA, 4 mA being closed and 20 mA being fully open. The connection to the controller is made via output AO-0 (terminals 2A, 3A) of a SFC446 Analog Output module, which is to be found in Rack 2, Slot 2 of the controller backplane.

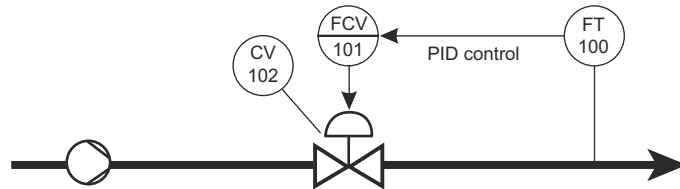


Fig. 8-2: Example of closed-loop control

Fig. 8-3 shows the control strategy together with links that must be made between the blocks.

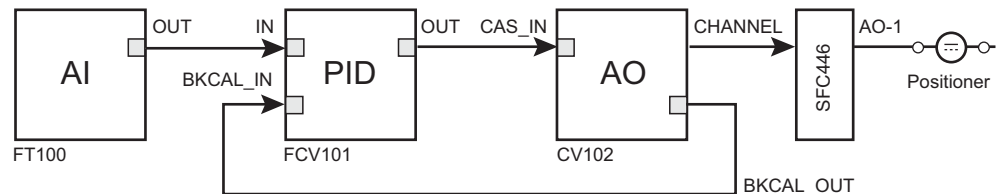


Fig. 8-3: Links required for basic closed-loop control

Block configuration

It is assumed that PID block has been configured as described in Chapter 6.1 to give an **OUT** value in the range 0% to 100%. This becomes the value of **CAS_IN** and must be scaled using **XD_SCALE** and **PV_SCALE** to provide a value between 4 mA and 20 mA to the transducer block.

Before the AO block is configured, however, the HC transducer block of the controller must be configured to recognise the SFC446 analog output module

Parameter	Function	Example value
Controller HC block		
MODE_BLK.Target	Normal operating mode of block	Auto
IO_TYPE_R2.SLOT_2	Declaration of SFC446 module in HC block	4-Analog Output
AO block		
MODE_BLK.Target	Normal operating mode of block	Cas
PV_SCALE.EU_100	Upper range limit for process variable	100
PV_SCALE.EU_0	Lower range limit for process variable	0
PV_SCALE.UNITS_INDEX	Unit of process variable	%
XD_SCALE/EU_100	Upper range limit for transducer variable	20
XD_SCALE/EU_0	Lower range limit for transducer variable	4
XD_SCALE/UNITS_INDEX	Unit of transducer variable	mA
SP_RATE_DN	Rate of change from old to new, higher SP	5
SP_RATE_UP	Rate of change from old to new, lower SP	5
CHANNEL	Rack + Slot + Group + Point	2200
FSTATE_VAL	Fail-to-safe value for OUT (= closed)	4
FSTATE_TIME	Time that elapses before fault state is activated	5 (s)
SHED_OPT	Target mode and recovery behaviour on change of mode	NormalShed_ NormalReturn

8.1.4 Block operation

Mode

The block normally operates in "Auto", "Cas" or "RCas" mode, depending on the origin of the setpoint, see Chapter 8.1.1.

When **MODE_BLK.Target** is set to "Man", the block output **OUT** can be manipulated.

The setpoint value **SP** can be changed when the block is in Auto. If other parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

The block assumes Local override (LO) when a fault state has been detected and the corresponding function has been configured. The **OUT** value then corresponds to **FSTATE_VAL**.

Iman is temporarily assumed during a cascade loop reset initiated by an upstream block.

Operation can be checked on-line by viewing a number of parameters:

SP, CAS_IN, RCAS_IN	Value and status of setpoint set by operator, upstream block or supervisor system respectively
OUT	The primary analog value calculated as a result of executing the function.
BKCAL_OUT RCAS_OUT	The block output and status provided to an upstream block or host for back calculation
READBACK	Readback of the actual analog valve or other actuator position, in the transducer state.

Status

The significance of the output status is indicated below.

Output status	Meaning
GoodCascade	<ul style="list-style-type: none"> Normal operation as configured
Bad	<ul style="list-style-type: none"> Input to block Bad Block out of service

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Block Configuration Error	<ul style="list-style-type: none"> Invalid SHED_OPT value Invalid CHANNEL value <ul style="list-style-type: none"> value not supported by transducer block value not compatible with configuration in Controller HC block XD_SCALE range or value not supported by transducer block Transducer block out of service
Simulate Active	<ul style="list-style-type: none"> Simulation enabled in block
Device Fault State Set	<ul style="list-style-type: none"> FAULT_STATE is active (Local override)
Output Failure	<ul style="list-style-type: none"> Associated I/O module has failed
Out-of-Service	<ul style="list-style-type: none"> MODE_BLK.Target currently set to OOS

8.1.5 Block parameters

Parameter	Valid range/ Options	Default value	Description/Action
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set to MODE_BLK.Target to Cas
BLOCK_ERR	RO		Block errors, see Chapter 5.2.3
PV	RO		Process value used in executing the block
SP			Setpoint used by the block in AUTO mode (PV_SCALE \pm 10%)
OUT			Output value calculated as a result of executing the block
SIMULATE		Disable	See Chapter 5.2.2, Enable/Disable options: ■ Disable ■ Active
PV_SCALE		0-100%	PV value scaling to to % of range
XD_SCALE		0-100%	Transducer value scaling to required engineering units
GRANT_DENY		0	Access options, see Chapter 2.8.1
IO_OPTS		0	I/O options, see Chapter 2.8.2 and Chapter 5.2.1
STATUS_OPTS		0	Status options, see Chapter 2.8.4 and Chapter 5.2.2
READBACK		0	Value sent back by actuator indicating current position
CAS_IN			Setpoint used by the block in CAS mode
SP_RATE_DN	Positive	+INF	Ramp rate for positive setpoint change in PV units/s
SP_RATE_UP	Positive	+INF	Ramp rate for negative setpoint change in PV units/s
SP_HI_LIM		100	High limit for setpoint limiting (PV_SCALE \pm 10%)
SP_LO_LIM		0	Low limit for setpoint limiting (PV_SCALE \pm 10%)
CHANNEL		0	The number of the logical hardware channel to the transducer that is connected to this I/O block, see Chapter 5.1.
FSTATE_TIME		0	Time in seconds to ignore a new fault state condition
FSAFE_VAL			Sets value for IO_OPTS option Fault State to Value
BKCAL_OUT			Feedback for BKCAL_IN of an upstream block.
RCAS_IN			Setpoint used by the block in RCAS mode (from host)
SHED_OPT	1 - 8	0	Mode shedding options, see Chapter 6.1.6
RCAS_OUT			Setpoint and status for host back calculation in RCAS mode
UPDATE_EVT			This alert is generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
Legend: RO = Read Only			

8.2 Discrete Output

The Discrete Output block receives a discrete setpoint value from an upstream function block or higher level process control system and, after possible inversion, outputs a signal to a downstream transducer block through an internal channel reference. It is used by instruments that act as discrete output elements in a control loop such as on-off valves or in connection with the ControlCare Discrete Output modules SFC428, SFC432, SFC435 or SFC438.

In ControlCare Application Designer the Discrete Output block is assigned the generic name **Device Tag-DO-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 8-4.

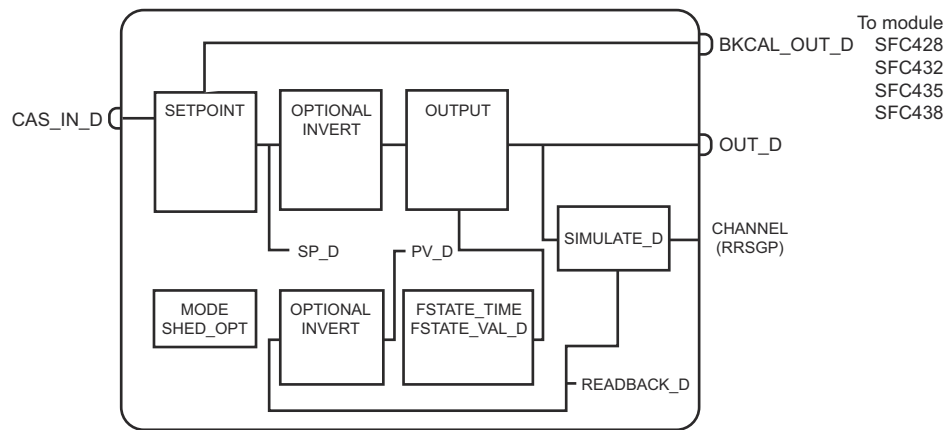


Fig. 8-4: Schematic diagram of the Discrete Output block

8.2.1 Functional description

Setpoint

Depending upon the setting of the **MODE_BLK.Target** parameter, the Discrete Output block receives its setpoint value from the operator, an upstream block or a remote process control system.

MODE_BLK.Target	Setpoint	Remarks
Auto	SP_D	Setpoint value is constant and is entered manually by the operator.
Cas	CAS_IN_D	Setpoint value originates from an upstream block
Rcas	RCAS_IN_D	Setpoint value originates from a supervisory host

Optional invert

The "Invert" option in the parameter group **IO_OPTS** inverts the input signal by placing a boolean NOT between the setpoint (**SP_D**, **CAS_IN_D**, **RCAS_IN_D**) and the output **OUT_D**.

False	A discrete value of zero (0) is considered to be a logical zero (0), and a non-zero discrete value a logical one (1)
True	A discrete value of zero (0) is considered to be a logical one (1), and a non-zero discrete value a logical zero (0)

The **XD_STATE** value indicates the true on/off state of the connected hardware

Output

When the block is operating in Cas, Rcas or Auto mode, the value obtained after block execution is placed in **OUT_D**. The same value is transmitted to the transducer through an internal channel reference given by the **CHANNEL** parameter, see Chapter 5.1.

If the transducer hardware supports a readback value, such as valve position, that value is run backwards through the inversion to act as the **PV_D** for the block. If not supported, **READBACK_D** is generated from **OUT_D**.

The current setpoint value and status is communicated to an upstream block by the **BKCAL_OUT_D** parameter. If the option "PV for BKCAL_OUT" is selected in **IO_OPTS**, then **PV_D** will be transmitted instead of the setpoint.

When the block is operating in Rcas mode, **RCAS_OUT_D** carries the block setpoint and status after ramping for output to a supervisory host as back calculation and to allow action to be taken under limiting conditions or mode change.

Fault state

By default, the last valid value is held if a fault state is detected.

If the "Fault state to value" option is selected in **IO_OPTS**, **OUT_D** will assume the value entered in **FSTATE_VAL_D** when a fault condition exists longer than defined in the parameter **FSTATE_TIME** or when the option "Fault State" is selected the the resource block parameter **SET_FSTATE**. This ensures that the hardware connected to the block fails to safe, see Chapter 2.7.

When the option "Fault state restart" in **IO_OPTS** is selected, **FSTATE_VAL_D** will also be used on restart.

Mode shedding

SHED_OPT allows the behaviour of the block to be determined after a change in mode, e.g. forced by the system or through operator intervention. "Normal return" options allow the block to recover automatically to target mode, for "No return" options, the operator must enter the target mode manually.

Setpoint tracking

IO_OPTS also allows the setpoint to track PV_D or an input value when the the target mode is Auto and this changes to Man or LO. Possible behaviour is "SP tracks PV when Man", "SP tracks PV when LO", "SP tracks RCAS or CAS when Man or LO"

Simulation

The block supports simulation, see Chapter 2.10. The parameter group **SIMULATE_D** is used to specify the output value (setpoint value and status), transducer value (PV value and status) and to enable/disable simulation, provided the associated jumper on the I/O board is set, see Chapter 3.2. When simulation is active in the Analog Output block, **BLOCK_ERROR** flags "Simulation Active".

Status propagation

The status of OUT can assume the following values:

GoodCascade	The output value OUT_D is valid and can be used for further processing
Bad	The output value OUT_D is invalid. <ul style="list-style-type: none"> ■ Operating mode OOS ■ Input status bad

If the "Propagate Fault Bkwd" option in **STATUS_OPTS** is selected, the **BKCAL_OUT_D** and **RCAS_OUT_D** status carries the current **OUT_D** status to an upstream block or supervisory system.

8.2.2 Output configuration for SFC428, SFC432, SFC435, SFC438

The DO block forces the output signals as shown in Table 8-4. NO (Normally open) and NC (normally closed) is the state of the relay when no power is applied to the module.

OUT_D	Signal	NO Relay SFC428, SFC432, SFC438	NC Relay SFC435, SFC438
0	FALSE	Open	Closed
1	TRUE	Closed	Open

Tab. 8-4: Relay status of the output module as a function of OUT_D

8.2.3
Block configuration

Fig. 8-5 shows an example of on-off control of process tank level that uses the Discrete Output block. The level in a process tank with random inflow is controlled by opening and closing the outlet valve (LV100) based on the level sensed in the tank by the level switch (LSH100). The level switch outputs a true signal when it is covered with liquid; the valve is operated by a solenoid and is open when this is activated. The connection to the controller is made via input DI-0 and output DO-1 of a SFC432 Voltage Input and NO Relay Output module, which is to be found in Rack 1, Slot 3 of the controller backplane.

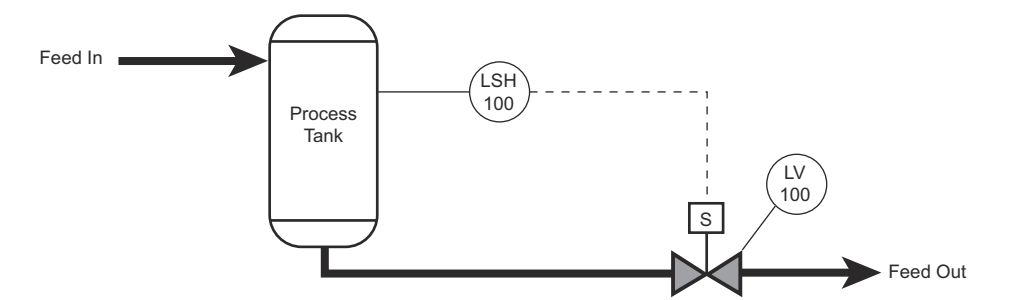


Fig. 8-5: Example of on-off control of process tank level

Fig. 8-6 shows the control strategy together with links that must be made between the blocks.

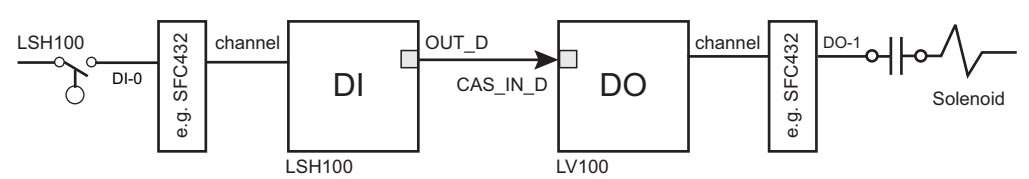


Fig. 8-6: Links required for basic closed-loop control

Block configuration

Before the DI and DO blocks are configured, the HC transducer block of the controller must be configured to recognise the SFC432 Voltage Input and NO Relay Output module

Parameter	Function	Example value
Controller HC block		
MODE_BLK.Target	Normal operating mode of block	Auto
IO_TYPE_R1.SLOT_3	Declaration of SFC432 module in HC block	8-DiscIn 4-DiscOut
DI block		
MODE_BLK.Target	Normal operating mode of block	Auto
CHANNEL	Rack + Slot + Group + Point	1300
DO block		
MODE_BLK.Target	Normal operating mode of block	Cas
CHANNEL	Rack + Slot + Group + Point	1310
FSTATE_VAL_D	Fail-to-safe value for OUT_D (= open, prevents overspill)	1
FSTATE_TIME	Time that elapses before fault state is activated	5 (s)
SHED_OPT	Target mode and recovery behaviour on change of mode	NormalShed_ NormalReturn

8.2.4 Block operation

Mode

The block normally operates in "Auto", "Cas" or "RCas" mode, depending on the origin of the setpoint, see Chapter 8.2.1.

When **MODE_BLK.Target** is set to "Man", the block output **OUT** can be manipulated.

The setpoint value **SP_D** can be changed when the block is in Auto. If other parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

The block assumes Local override (LO) when a fault state has been detected and the corresponding function has been configured. The **OUT_D** value then corresponds to **FSTATE_VAL_D**.

Iman is assumed when there is no path to the final control element.

Operation can be checked on-line by viewing a number of parameters:

SP_D, CAS_IN_D, RCAS_IN_D	Value and status of setpoint set by operator, upstream block or supervisor system respectively
OUT	The primary discrete value calculated as a result of executing the function.
BKCAL_OUT_D RCAS_OUT_D	The block output and status provided to an upstream block or host for back calculation
READBACK_D	Readback of the actual valve or actuator position, in the transducer state.

Status

The significance of the output status is indicated below.

Output status	Meaning
GoodCascade	<ul style="list-style-type: none"> Normal operation as configured
Bad	<ul style="list-style-type: none"> Input to block Bad Block out of service

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Block Configuration Error	<ul style="list-style-type: none"> Invalid SHED_OPT value Invalid CHANNEL value <ul style="list-style-type: none"> value not supported by transducer block value not compatible with configuration in Controller HC block
Simulate Active	<ul style="list-style-type: none"> Simulation enabled in block
Device Fault State Set	<ul style="list-style-type: none"> FAULT_STATE is active (Local override)
Output Failure	<ul style="list-style-type: none"> Associated I/O module has failed
Out-of-Service	<ul style="list-style-type: none"> MODE_BLK.Target currently set to OOS

8.2.5 Block parameters

Parameter	Valid range/ Options	Default value	Description/Action
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set to MODE_BLK.Target to Cas
BLOCK_ERR	RO		Block errors, see Chapter 5.2.3
PV_D	RO		Process value used in executing the block
SP_D			Setpoint used by the block in AUTO mode (PV_SCALE \pm 10%)
OUT			Output value calculated as a result of executing the block
SIMULATE_D		Disable	See Chapter 5.2.2, Enable/Disable options: ■ Disable ■ Active
PV_STATE		0-100%	Discrete PV value
XD_STATE		0-100%	Discrete transducer value
GRANT_DENY		0	Access options, see Chapter 2.8.1
IO_OPTS		0	I/O options, see Chapter 2.8.2 and Chapter 5.2.1
STATUS_OPTS		0	Status options, see Chapter 2.8.4 and Chapter 5.2.2
READBACK_D		0	Value sent back by actuator indicating current position
CAS_IN_D			Setpoint used by the block in CAS mode
CHANNEL		0	The number of the logical hardware channel to the transducer that is connected to this I/O block, see Chapter 5.1.
FSTATE_TIME		0	Time in seconds to ignore a new fault state condition
FSTATE_VAL_D			Sets value for IO_OPTS option Fault State to Value
BKCAL_OUT_D			Feedback for BKCAL_IN of an upstream block.
RCAS_IN_D			Setpoint used by the block in RCAS mode (from host)
SHED_OPT	1 - 8	0	Mode shedding options, see Chapter 6.1.6
RCAS_OUT_D			Setpoint and status for host back calculation in RCAS mode
UPDATE_EVT			This alert is generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
Legend: RO = Read Only			

8.3 Multiple Analog Output

The Multiple Analog Output block receives up to eight analog input values from upstream function blocks and outputs up to eight signals through an internal channel reference an analog output module. This forces a voltage or current output signal proportional to the input value. In the case of the SFC446 analog output module only the first four inputs are used.

In ControlCare Application Designer the Multiple Analog Output block is assigned the generic name **Device Tag-MAO-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 8-7.

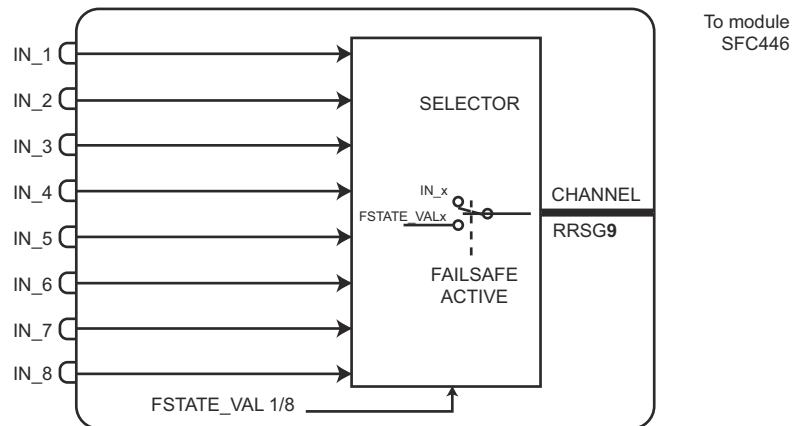


Fig. 8-7: Schematic diagram of the Multiple Analog Output block

8.3.1 Functional description

The Multiplier Analog Output block forces up to eight electrical output signals between 0 mA and 20 mA, -5 VDC and 5 VDC or -10 VDC to 10 VDC depending upon the wiring and the DIP switch setting of the connected SFC446 module as well as the value range of the input signal. There are no scaling, limiting or tracking functions. The output current or voltage of **Point x** is directly proportional to the **IN_x** value, see Table 8-5.)

IN_x	Current	Voltage (DIP switch OFF)	Voltage (DIP switch ON)
100 %	20 mA	5 VDC	10 VDC
0 %	4 mA	1 VDC	2 VDC
-25%	0 mA	0 VDC	0 VDC
-150%	N/A	-5 VDC	-10 VDC

Tab. 8-5: Current and voltage output of SFC446 module as a function of configuration and input signal

When the block is operating in Auto mode, the value obtained after block execution is transmitted to the transducer through an internal channel reference given by the **CHANNEL** parameter, see Chapter 5.1.

Fault state

By default, the last valid value is held if a fault state is detected.

If the "Fault state to value x" option is selected in **MO_OPTS**, the output will force a signal proportional to the value entered in **FSTATE_VALx** when a fault condition exists longer than defined in the parameter **FSTATE_TIME** or when the option "Fault State" is selected the the resource block parameter **SET_FSTATE**. This ensures that the hardware connected to the block fails to safe, see Chapter 2.7.

When the option "Use fault state value on restart x" in **MO_OPTS** is selected, **FSTATE_VALx** will also be used on restart.

8.3.2 Block configuration

Output module

Fig. 8-8 shows an example of a control strategy using the Multiple Analog Output block. Currently the only analog output module available, the SFC446, has four outputs which can be wired as current or voltage outputs depending on which terminals are used.

The four input values IN_1 to IN_4 are connected to the OUT outputs of upstream blocks. These are scaled to 0% to 100%, -25% to +100% or -150% to +100%, depending upon the output signal range required at the SFC446 module, see Table 8-5. The relationship between input signal to output in the SFC446 is as follows:

- IN_1 drives the AO-0 output
- IN_2 drives the AO-1 output
- IN_3 drives the AO-2 output
- IN_4 drives the AO-3 output

For the configuration example it is assumed that the SFC446 module is at Rack 2, Slot 0 of the controller backplane.

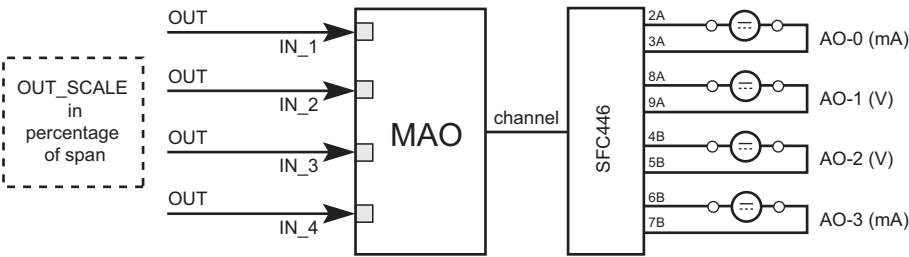


Fig. 8-8: Example of links required for Multiple Analog Output block used with the SFC446

Block configuration

Before the MDO block is configured, the HC transducer block of the controller must be configured to recognise the SFC446 Analog Output module

Parameter	Function	Example value
Controller HC block		
MODE_BLK.Target	Normal operating mode of block	Auto
IO_TYPE_R2.SLOT_0	Declaration of SFC446 module in HC block	4-Analog Output
MDO block		
MODE_BLK.Target	Normal operating mode of block	Auto
CHANNEL	Rack + Slot + Group + 9	2009
FSTATE_TIME	Time that elapses before fault state is activated	5 (s)
FSTATE_VAL1	Fail-to-safe value for AO-0 (= IN_1)	e.g.0
FSTATE_VAL2	Fail-to-safe value for AO-1 (= IN_2)	e.g.0
FSTATE_VAL3	Fail-to-safe value for AO-2 (= IN_3)	e.g.100
FSTATE_VAL4	Fail-to-safe value for AO-3 (= IN_4)	e.g.0

Use in control loops

Fig 8-9 shows the use of the Multiple Discrete Output block in a control loop. In order that the PID can work, it has to receive a **BKCAL_IN** signal, which is not provided by the MAO block. This is achieved by linking **OUT** signal of the PID block to the **BKCAL_IN** signal of the same block.

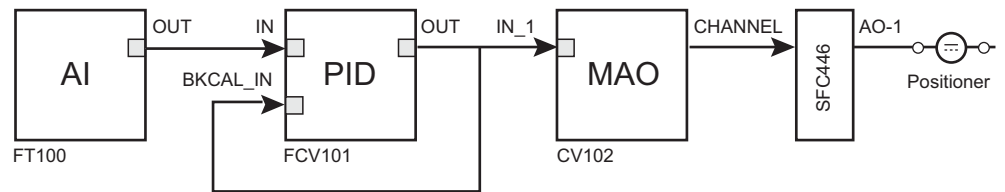


Fig. 8-9: Use of MAO block in control loops

Block configuration

Before the MDO block is configured, the HC transducer block of the controller must be configured to recognise the SFC446 Analog Output module

Parameter	Function	Example value
Controller HC block		
MODE_BLK.Target	Normal operating mode of block	Auto
IO_TYPE_R2.SLOT_0	Declaration of SFC446 module in HC block	4-Analog Output
MDO block		
MODE_BLK.Target	Normal operating mode of block	Auto
CHANNEL	Rack + Slot + Group + 9	2009
FSTATE_TIME	Time that elapses before fault state is activated	5 (s)
FSTATE_VAL1	Fail-to-safe value for AO-0 (= IN_1)	e.g.0

8.3.3 Block operation**Mode**

The block normally operates in "Auto" mode. Manual mode "Man" is implemented, but only for the recovery from OOS and LO to Auto.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

The block assumes Local override (LO) when a fault state has been detected and the corresponding function has been configured. The output value then corresponds to **FSTATE_VALx**. The **FSTATE_STATUS** parameter indicates which outputs are using fault state values.

Operation can be checked on-line by viewing the IN_x parameter

Status

As there is no linkable output, there is also no status handling.

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Block Configuration Error	<ul style="list-style-type: none"> Invalid CHANNEL value <ul style="list-style-type: none"> value not supported by transducer block value not compatible with configuration in Controller HC block
Device Fault State Set	<ul style="list-style-type: none"> FAULT_STATE is active (Local override)
Output Failure	<ul style="list-style-type: none"> Associated I/O module has failed
Out-of-Service	<ul style="list-style-type: none"> MODE_BLK.Target currently set to OOS

8.3.4 Block parameters

Parameter	Valid range/ Options	Default value	Description/Action
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set to MODE_BLK.Target to Cas
BLOCK_ERR	RO		Block errors, see Chapter 5.2.3
CHANNEL		0	The number of the logical hardware channel to the transducer that is connected to this I/O block, see Chapter 5.1.
IN_1	RO		Value and status of input signal 1
IN_2	RO		Value and status of input signal 2
IN_3	RO		Value and status of input signal 3
IN_4	RO		Value and status of input signal 4
IN_5	RO		Value and status of input signal 5
IN_6	RO		Value and status of input signal 6
IN_7	RO		Value and status of input signal 7
IN_8	RO		Value and status of input signal 8
MO_OPTS			Selects fail safe vlues and type to be output on block failure <ul style="list-style-type: none"> ■ Faultstate to value x: block outputs FSAFE_VALx on fault ■ Use fault state value on restart x: block outputs FSAFE_VALx on restart
FSTATE_TIME		0	Time iin seconds to ignore a new fault state conditioin
FSTATE_VAL1		0	Fail-to-safe value for IN_1
FSTATE_VAL2		0	Fail-to-safe value for IN_2
FSTATE_VAL3		0	Fail-to-safe value for IN_3
FSTATE_VAL4		0	Fail-to-safe value for IN_4
FSTATE_VAL5		0	Fail-to-safe value for IN_5
FSTATE_VAL6		0	Fail-to-safe value for IN_6
FSTATE_VAL7		0	Fail-to-safe value for IN_7
FSTATE_VAL8		0	Fail-to-safe value for IN_8
FSTATE_STATUS			Indicates which failsafe values are active
UPDATE_EVT			This alert is generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
Legend: RO = Read Only			

8.4 Multiple Discrete Output

The Multiple Discrete Output block receives up to eight discrete input values from upstream function blocks and outputs up to eight discrete signals through an internal channel reference to a SFC428, SFC432, SFC435 or SFC438 discrete output module. If signal inversion is required, the Discrete Output block must be used.

In ControlCare Application Designer the Multiple Discrete Output block is assigned the generic name **Device Tag-MDO-n**. It can be found by expanding the appropriate device or Field Controller leaf in the fieldbus or process cell window and appears as shown in Fig. 8-10.

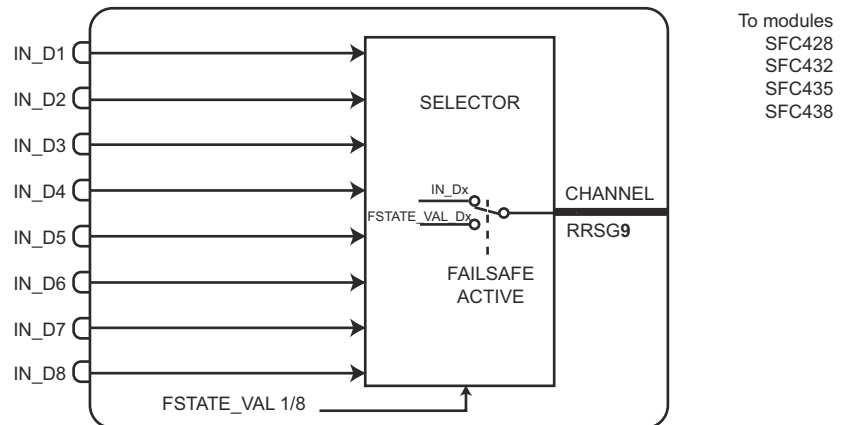


Fig. 8-10: Schematic diagram of Multiple Discrete Output block

8.4.1 Functional description

The Multiple Discrete Output block forces the electrical outputs of the connected relay output module SFC428, SFC432, SFC435 or SFC438 to "True/1" or "False/0" according to the value of **IN_Dx**, which must be "0" or "1". The relay status is as shown in Table 8-6. NO (Normally open) and NC (normally closed) is the state of the relay when no power is applied to the module.:

IN_Dx	Signal	NO Relay SFC428, SFC432, SFC438	NC Relay SFC435, SFC438
0	FALSE	Open	Closed
1	TRUE	Closed	Open

Tab. 8-6: Relay status of the output module as a function of IN_Dx

When the block is operating in Auto mode, the value obtained after block execution is transmitted to the transducer through an internal channel reference given by the **CHANNEL** parameter, see Chapter 5.1.

Fault state

By default, the last valid value is held if a fault state is detected.

If the "Fault state to value x" option is selected in **MO_OPTS**, the output will force a signal proportional to the value entered in **FSTATE_VALx** when a fault condition exists longer than defined in the parameter **FSTATE_TIME** or when the option "Fault State" is selected the the resource block parameter **SET_FSTATE**. This ensures that the hardware connected to the block fails to safe, see Chapter 2.7.

When the option "Use fault state value on restart x" in **MO_OPTS** is selected, **FSTATE_VALx** will also be used on restart.

8.4.2 Block configuration

Output module

Fig. 8-11 shows an example of a control strategy using the Multiple Discrete Output block and the SFC438 module. This has eight voltage inputs and four output relays, two NO and two NC. In this case, and in the cases of the SFC432 (four NO relays) and SFC435 (four NC relays) only the first four inputs are used:

- IN_D1 drives the DO-1 output
- IN_D2 drives the DO-2 output
- IN_D3 drives the DO-3 output
- IN_D4 drives the DO-4 output

The SFC428 high density relay module has 16 NO relays and requires two MDO blocks, one for each relay group. All eight inputs can be used:

- IN_D1 drives the DO-0 output
- IN_D2 drives the DO-1 output
- ...
- IN_D8 drives the DO-7 output

For the configuration example it is assumed that the SFC438 module is at Rack 3, Slot 3 of the controller backplane.

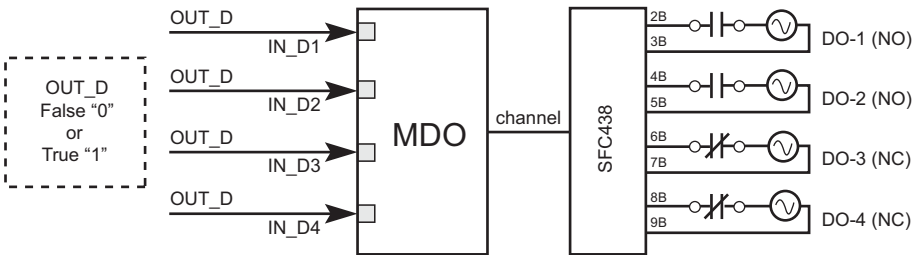


Fig. 8-11: Example of links required for Multiple Discrete Output block used with the SFC438

Block configuration

Before the MDO block is configured, the HC transducer block of the controller must be configured to recognise the SFC446 Analog Output module

Parameter	Function	Example value
Controller HC block		
MODE_BLK.Target	Normal operating mode of block	Auto
IO_TYPE_R3.SLOT_3	Declaration of SFC446 module in HC block	8-DiscIn 4-DiscOut
MDO block		
MODE_BLK.Target	Normal operating mode of block	Auto
CHANNEL	Rack + Slot + Group + 9	3319
FSTATE_TIME	Time that elapses before fault state is activated	5 (s)
FSTATE_VAL_D1	Fail-to-safe value for AO-0 (= IN_1)	e.g.0
FSTATE_VAL_D2	Fail-to-safe value for AO-1 (= IN_2)	e.g.0
FSTATE_VAL_D3	Fail-to-safe value for AO-2 (= IN_3)	e.g.1
FSTATE_VAL_D4	Fail-to-safe value for AO-3 (= IN_4)	e.g.1

8.4.3 Block operation

Mode

The block normally operates in "Auto" mode. Manual mode "Man" is implemented, but only for the recovery from OOS and LO to Auto.

If parameters are to be changed while the block is online, **MODE_BLK.Target** must be temporarily set to OOS (Out of Service). A parameter change is then downloaded directly to the controller.

The block assumes Local override (LO) when a fault state has been detected and the corresponding function has been configured. The output value then corresponds to **FSTATE_VAL_Dx**. The **FSTATE_STATUS** parameter indicates which outputs are using fault state values.

Operation can be checked on-line by viewing the **IN_Dx** parameter

Status

As there is no linkable output, there is also no status handling.

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Block Configuration Error	<ul style="list-style-type: none"> Invalid CHANNEL value <ul style="list-style-type: none"> value not supported by transducer block value not compatible with configuration in Controller HC block
Device Fault State Set	FAULT_STATE is active (Local override)
Output Failure	Associated I/O module has failed
Out-of-Service	MODE_BLK.Target currently set to OOS

8.4.4 Block parameters

Parameter	Valid range/ Options	Default value	Description/Action
ST_REV		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY		0	
ALERT_KEY	1 to 255	0	
MODE_BLK		O/S	Block mode, set to MODE_BLK.Target to Cas
BLOCK_ERR	RO		Block errors, see Chapter 5.2.3
CHANNEL		0	The number of the logical hardware channel to the transducer that is connected to this I/O block, see Chapter 5.1.
IN_D1	RO		Value and status of input signal 1
IN_D2	RO		Value and status of input signal 2
IN_D3	RO		Value and status of input signal 3
IN_D4	RO		Value and status of input signal 4
IN_D5	RO		Value and status of input signal 5
IN_D6	RO		Value and status of input signal 6
IN_D7	RO		Value and status of input signal 7
IN_D8	RO		Value and status of input signal 8
MO_OPTS			Selects fail safe vlues and type to be output on block failure <ul style="list-style-type: none"> Faultstate to value x: block outputs FSAFE_VALx on fault Use fault state value on restart x: block outputs FSAFE_VALx on restart
FSTATE_TIME		0	Time iin seconds to ignore a new fault state conditioin
FSTATE_VAL_D1		0	Fail-to-safe value for IN_1D
FSTATE_VAL_D2		0	Fail-to-safe value for IN_D2
FSTATE_VAL_D3		0	Fail-to-safe value for IN_D3
FSTATE_VAL_D4		0	Fail-to-safe value for IN_D4
FSTATE_VAL_D5		0	Fail-to-safe value for IN_D5
FSTATE_VAL_D6		0	Fail-to-safe value for IN_D6
FSTATE_VAL_D7		0	Fail-to-safe value for IN_D7
FSTATE_VAL_D8		0	Fail-to-safe value for IN_D8
FSTATE_STATUS			Indicates which failsafe values are active
UPDATE_EVT			This alert is generated by any change to the static data.
BLOCK_ALM			Block alarms, see Chapter 2.9
Legend: RO = Read Only			

9 PROFIBUS Blocks

The PROFIBUS Field Controller SFC173, PROFIBUS-DP devices and PROFIBUS-PA devices are modelled in ControlCare Application using a number of PROFIBUS function blocks. In contrast to Foundation Fieldbus devices, however, these blocks are normally not configured within Application Designer. Instead, PROFIBUS Configurator is used to set up the network and configure the Field Controller and slave devices. The results are then mapped to the PROFIBUS function blocks.

The block configuration of every PROFIBUS function block can be viewed in the Online and Offline Characterization dialogues. Changes to the PROFIBUS parameters, however, must be made in PROFIBUS Configurator and not in the parameter block.

The function blocks provided are as follows:

Special Function Block types:

- PROFIBUS Transducer block (PBTrd)

Function Block types for accessing cyclic PROFIBUS I/O data:

- PROFIBUS Analog Input for one analog process value
- PROFIBUS Multiple Analog Input for up to four analog process values
- PROFIBUS Digital Input for one discrete process value
- PROFIBUS Multiple Discrete Input for up to eight discrete process values
- PROFIBUS Totalizer for one totalizer value
- PROFIBUS Analog Output for one analogue process value
- PROFIBUS Multiple Analog Output for up to four analog process values
- PROFIBUS Discrete Output for one discrete process value
- PROFIBUS Multiple Discrete Output for up to 8 discrete process values

In addition to the above blocks, the PROFIBUS Field Controller contains a set of standard Foundation Fieldbus blocks which can be used and configured as normal.

In this chapter, only the blocks together with their schematics and parameters are described. The actual use of a block, including links to be made can be taken from the corresponding block description in Chapters 4 to 8.

9.1 General description

9.1.1 Accessing cyclic PROFIBUS I/O data

PROFIBUS input and output function blocks exchange their values with the PROFIBUS master via the Process Image (PI) which has two sections:

- Process Image Output Data and
- Process Image Input Data.

The PI is a byte array which size depends on the size of the dual port memory. A process value within the PI is selected by an offset. The size of the representing variable can be derived from its datatype. The datatype is given by the user manual for the device which provides/expects the process value.

The Process Image is set up automatically when a device is added to a project in PROFIBUS Configurator and has been mapped by the PROFIBUS Mapping tool. The device GSD file is used in this process.

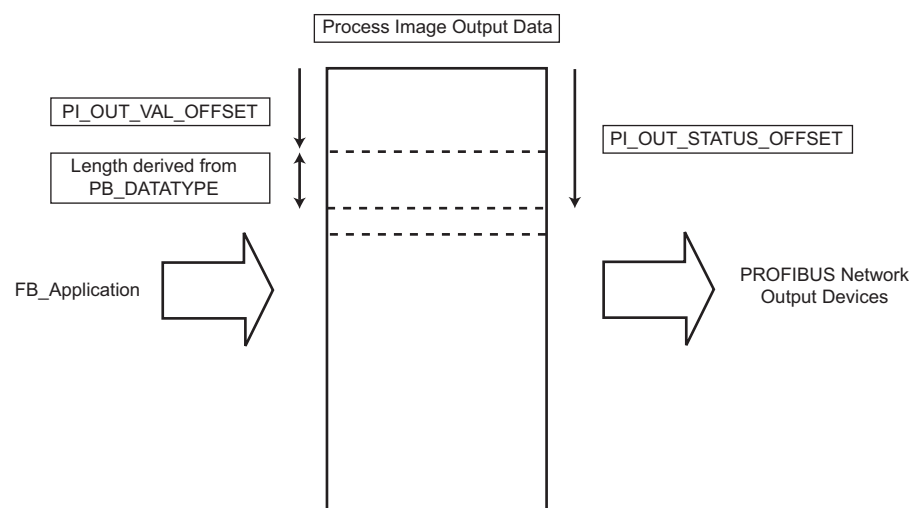


Fig. 9-1: Schematic diagram of the Process Image Output Data

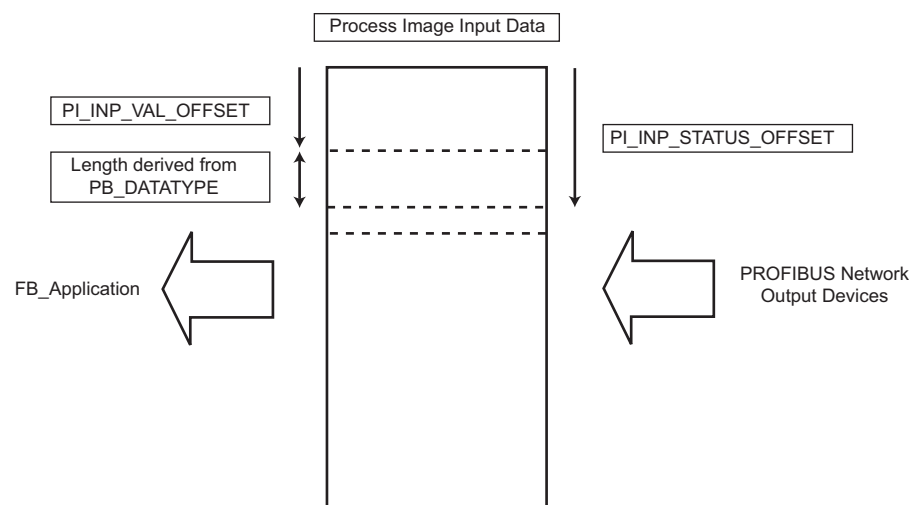


Fig. 9-2: Schematic diagram of the Process Image Input Data

9.1.2 Failsafe/Fault state mechanism

The FailSafe mechanism implemented in the PROFIBUS output function blocks has a similar behavior to the Fault State mechanism present in the standard FF output function blocks and ensures that a loop fails to safe should a parameter go bad or not be propagated. There are, however, some differences which have been realized in order to facilitate the user comprehension and configuration.

Configuration

It is possible to configure the parameters Mode, Status, Value and Time for each input independently in the Online or Offline Characterization dialogue. The parameters have been implemented in the various blocks as follows :

Function	Block	Parameter	Description/Action
Mode	PA_AO PA_DO	FAILSAFE_MODE	Sets the operating mode of the failsafe function: <ul style="list-style-type: none"> ■ Disable: the FailSafe mechanism will not be used, even in case of loop failure ■ Get FS Data: the values present in the failsafe status and value parameters will be used at the input of the function block in case of loop failure ■ Get Last Data: the last good status and value present in the input of the function block will be reused at the same input of the function block in case of loop failure
	PA_MAO PA_MDO	FS_MODE	
Status	PA_AO PA_DO	FAILSAFE_STATUS	Status copied to the status parameter of the input of the function block in case of loop failure, provided the option "Get FS Data" is set in the FailSafe parameter "Mode"
	PA_MAO PA_MDO	FS_STATUS	
Value	PA_AO	FAILSAFE_AVALUE	Value copied to the value parameter of the input of the function block in case of loop failure, provided the option "Get FS Data" is set in the FailSafe parameter "Mode"
	PA_DO	FAILSAFE_DVALUE	
	PA_MAO	FS_AVALUE	
	PA_MDO	FS_DVALUE	
Time	PA_AO PA_DO	FAILSAFE_TIME	Delay (in seconds) before the FailSafe mechanism becomes active. It is used in case of loop failure, provided the option "Get FS Data" or "Get Last Data" has been set in the FailSafe parameter "Mode"
	PA_MAO PA_MDO	FS_TIME	

Time

The failsafe time parameter functions as follows. If the parameter "Mode" is set to "Get FS Data", the FailSafe "Status" and "Value" will be used in case of loop failure. If the project designer wants the FailSafe mechanism hold the last good value and status in the input of the function block for 10 seconds before entering failsafe mode, the parameter "Time" must be set to 10. On loop failure, the system will now wait 10 seconds before entering failsafe mode. If the control loop recovers before the 10 seconds have elapsed, the FailSafe mechanism remains on hold and the FailSafe values are not applied.

Cascade loops

The failsafe mechanism behaves in a special manner for cascade loops. For example, assume that in a PA_AO function block the "Mode" parameter is set to "Get Last Data", the "Time" parameter is set to 10 and this function block is part of a PID loop. Before a failure condition, the input of this PA_AO block receives a status equal to "Good_Cascade" and a constant value equal to 20 from an external link.

Now assume that the external link breaks. The FailSafe mechanism is activated, holds the value 20 in the value parameter of the input of the PA_AO block, sets a specific bit of the last good status, changing it to "Good_Cascade:InitiateFaultState", and holds this status in the status parameter of the input of the PA_AO block for 10 seconds. If the system does not recover after 10 seconds, the status parameter of the input of the PA_AO block is changed to "Good_Cascade:FaultStateActive". The FailSafe status and value are now applied at the input of the PA_AO block until the external link is recovered. This special behavior is important because there are some PROFIBUS field devices can use the "Fault State" information present in the status to activate their own FailSafe rules.

SETPOINT and CAS_IN

The function blocks PA_AO and PA_DO also exhibit special behaviour regarding the inputs "SETPOINT" and "RCAS_IN". As these function blocks have 2 inputs but just one set of FailSafe parameters. For this reason when the "SETPOINT" input is configured, the FailSafe mechanism is applied to it. Otherwise, when only the RCAS_IN is configured as input, the FailSafe mechanism is applied only to it.

Block	Parameter	Failsafe parameters
PA_AO	SETPOINT	Apply when AO_MODE_SEL is set to <ul style="list-style-type: none"> ■ "Only SP", ■ "SP, CHECK_BACK", ■ "SP, READBACK, POS_D", ■ "SP, READBACK, POS_D, CHECK_BACK" or ■ "SP, READBACK, POS_D, RCAS_IN, RCAS_OUT, CHECK_BACK"
	RCAS_IN	Apply when AO_MODE_SEL is set to <ul style="list-style-type: none"> ■ "RCAS_IN, RCAS_OUT" or ■ "RCAS_IN, RCAS_OUT, CHECK_BACK".
PA_DO	SETPOINT	Apply when DO_MODE_SEL is set to <ul style="list-style-type: none"> ■ "Only SP_D", ■ "SP_D, CHECK_BACK", ■ "SP, READBACK_D, CHECK_BACK" or ■ "SP_D, READBACK_D, RCAS_IN_D, RCAS_OUT_D, CHECK_BACK".
	RCAS_IN	Apply when DO_MODE_SEL is set to <ul style="list-style-type: none"> ■ RCAS_IN_D, RCAS_OUT_D" or ■ "RCAS_IN_D, RCAS_OUT_D, CHECK_BACK".

Note!

- The options in AO_MODE_SEL and DO_MOD_SEL are automatically set during the configuration of the device in PROFIBUS Configurator and reflect the attributes of the selected GSD module.
- These parameters may not be changed in the Online or Offline Characterization dialogue

9.1.3 Configuring PROFIBUS input and output values

PROFIBUS function blocks do not have a **CHANNEL** parameter to select the process variable to be used. Instead blocks are created for each of the process variables selected during device configuration in PROFIBUS Configurator, see Chapters 3.8 and 3.9 of the Profibus Tutorial, BA036S/04/en.

In the **Slave Configuration** dialogue of PROFIBUS Configurator, the blocks offered by the device are to be seen in the middle of the workspace. The blocks are assigned to slots in by selecting the one required and pressing the **Append Module** button. The block then appears in the lower workspace.

The designation of the blocks depends on the device GSD. Some use the generic block names, e.g. "AI" others specific variables, e.g. "Temperature". Some devices require that the blocks are assigned to specific slots. The FREE SPACE/EMPTY_MODULE parameter is used to ensure that the block is assigned to the correct slot, i.e. if variables 1 and 5 are required, the configuration AI, FREE SPACE, FREE SPACE, FREE SPACE, AI must be created in PROFIBUS Configurator. The blocks offered by a device together with the associated slots can be taken from the device manual.

On completion, the configuration is by Application Designer. Normally, the mapping proceeds automatically and must only be confirmed by the user. In the case of Remote I/Os, however, the user must know the data format of the input or output value, in particular, how the status is transmitted.

After confirmation of the mapping Application Designer creates the required blocks and assigns the generic names **Device Tag PB-XX-1**, **Device Tag PB-XX-2**, etc.

9.2 PROFIBUS Transducer

When a "New Gateway" is selected in the HSE Network, a PROFIBUS transducer block is created for SFC173 PROFIBUS Field Controller. In ControlCare Application Designer the block is assigned the generic name **Device Tag-PBTRD-n** and can be found by expanding the controller leaf in the Profibus or process cell window.

The table below lists the parameters to be found in the PROFIBUS transducer block, whereby:

- The parameters **PB_NUM_SEL** to **PB_DEVICE_CLASS** mirror the information contained in the PROFIBUS live list.
- The parameters from **PB_DIAG_AVAILABLE** to **PB_DATA_CONTROL_TIME** mirror the PROFIBUS master configuration which is created in PROFIBUS Configurator.
- The parameters from **PB_ALARM_MAX** to **PB_TIMEOUT_COUNT** mirror the protocol diagnostic interface between the application and the PROFIBUS master. They show the global bus states and the states of managed slave stations.

Note!



- The live list should not be read with the aid of the **PBTRD** block but by clicking on the PROFIBUS node in Application Designer

Block parameters

Parameter	Valid range/ Options	Default value	Description/Action
ST_VER		0	See Chapter 2.3.4
TAG_DESC		blanks	
STRATEGY	0 to 255	0	
ALERT_KEY	1 to 255	1	
MODE_BLK	TARGET	O/S	Block mode, set to Auto
BLOCK_ERR	0 to 15		Block errors, see Chapter 5.2.3
PB_COMMAND_REQ	RO	blanks	Profibus Command Request: Host writes the ID of the command to be executed by the SFC173 to this parameter
PB_COMMAND_RSP	RO	blanks	Profibus Command Response: Host reads the response to the last command written to the Profibus Command Request parameter from this parameter
PB_NUM_LINKS	RO	0	Number of links supported by the SFC173.
PB_LINK_SEL	0 - 3	0	Selects the PROFIBUS link, the live list information of which is to be displayed in parameters PB_LINK_ID to PB_DEV_SEL <ul style="list-style-type: none"> ■ 0: First ■ 1: Next ■ 2: Previous ■ 3: Last
PB_LINK_ID	RO		ID of PROFIBUS selected PB_LINK_SEL
PB_LINK_ID_REV	RO	0	Live list revision of current PROFIBUS
PB_NUM_DEV	RO	0	Number of devices: Number of devices in the live list
PB_DEV_SEL	0 - 3	0	Selects the PROFIBUS device, the live list information of which is to be displayed in parameters PB_DEVICE_ID to PB_DEVICE_CLASS <ul style="list-style-type: none"> ■ 0: First ■ 1: Next ■ 2: Previous ■ 3: Last
PB_DEVICE_ID	RO	blanks	ID of device selected in PB_DEV_SEL
PB_DEVICE_TAG	RO	blanks	Tag of device selected in PB_DEV_SEL
PB_DEVICE_ADDR	RO	0	Address of device selected in PB_DEV_SEL.
PB_DEVICE_MANF	RO	blanks	Device Manufacturer: Manufacturer of device selected in PB_DEV_SEL
PB_DEVICE_MODEL_NAME	RO	blanks	Model name of device selected in PB_DEV_SEL.
PB_DEVICE_CLASS	0 - 3		Device class of device selected in PB_DEV_SEL <ul style="list-style-type: none"> ■ 0: Active PB Master ■ 1: Backup PB Master ■ 2: PA Slave ■ 3: DP Slave

Parameter	Valid range/ Options	Default value	Description/Action
PB_DIAG_AVAILABLE	0, 1	0	Indicates whether device displays diagnostic data ■ 0: 'No' ■ 1: 'Yes'
PB_DEVICE_REVISION		blanks	Indicates the software revision of a device.
PB_MASTER_ADDR	0...125		Indicates node address of the PROFIBUS Master
PB_BAUDRATE	0 - 9		Sets baudrate for PROFIBUS network: 0: 9600 Baud 1: 19.2 kBd 2: 93.75 kBd 3: 187.5 kBd 4: 500 kBd 6: 1.5 MBd 7: 3.0 MBd 8: 6.0 MBd 9: 12.0 MBd Note: 45,45 kBd (Siemens link) and 31,25 kBd (requires coupler) are not supported
PB_TSL			PROFIBUS Slot Time
PB_MIN_TSDR			PROFIBUS Min Station Delay
PB_MAX_TSDR			PROFIBUS Max Station Delay
PB_TQUI			PROFIBUS Quiet Time
PB_TSET			PROFIBUS Setup Time
PB_TTR			PROFIBUS Target Rotation Time
PB_GAP			PROFIBUS GAP Update Factor
PB_HSA			PROFIBUS Highest Station Address
PB_MAX_RETRY_LIM			PROFIBUS Max Retry Limit
PB_BP_FLAG			PROFIBUS Bp Flag: – SFC173 supports only Bit7: Error_Action_Flag
PB_MIN_SLAVE_INTV	1 - 216		PROFIBUS Min Slave Interval Specifies the smallest allowed period of time between two consecutive DPslave poll cycles. This ensures that the sequence of cyclic service requests from the DPmaster (Class 1) can be handled by the DP-slave. This period of time will be complied by the DP-master (Class 1) for every cyclic Master-slave service except for the Global Control service.
PB_POLL_TO	1 - 216		PROFIBUS Poll Timeout Specifies the maximum period of time needed to get the response
PB_DATA_CONTROL_TIME			PROFIBUS Data Control Time Specifies the maximum period of time for an Data Exchange with every activated DP-slave. This Time is used for sending out the Global Control cyclically
PB_ALARM_MAX	7 - 32	RO	Maximum number of alarms per DP-slave that can be handled by the DP-master.
PB_MAX_USER_GLOBAL_CONTROL			Maximum number of Global Control requests that may be started by the user at the same time
PB_MASTER_STATE			Current state of PROFIBUS master ■ 0x0: OFFLINE ■ 0x40: STOP ■ 0x80: CLEAR ■ 0xC0: OPERATE

Parameter	Valid range/ Options	Default value	Description/Action
PB_PROTOCOL_STATE_BITS			<p>Current state of PROFIBUS Protocol</p> <ul style="list-style-type: none"> ■ 0: CONTROL-ERROR Parameterization error ■ 1: AUTO-CLEAR-ERROR Master stopped the communication to all slaves and reached the auto-clear end state ■ 2: NON-EXCHANGE-ERROR At least one slave has not reached the data exchange state and no process data are exchanged with it. ■ 3: FATAL-ERROR Bus error, no further bus communication is possible. ■ 4: EVENT-ERROR Master has detected bus short circuits. The number of detected events is fixed in the parameter ■ 5: PROFIBUS Bus Error Count Indicates number of bus errors since application started ■ 6: HOST-NOT-READY_NOTIF. Indicates whether host application is operative (bit not set) ■ 7: TIMEOUT-ERROR Communication timeout detected; indicates bus malfunction
PB_ERROR_REMOTE_ADR			Station address of error in PB_PROTOCOL_STATE_BITS
PB_ERROR_EVENT			<p>Error number of error in PB_PROTOCOL_STATE_BITS</p> <p>See PROFIBUS Configurator online help</p>
PB_BUSERROR_COUNT			Counter indicating number of serious bus malfunctions
PB_TIMEOUT_COUNT			Counter indicating number of telegram rejections due to serious bus malfunctions
PB_SLAVE_CFG_STATE			<p>Current configuration state of slave in master</p> <ul style="list-style-type: none"> – '1', slave configured in the master and serviced in its states. – '0', slave not configured in the master
PB_SLAVE_STATE			<p>Current state of each slave station when master running in OPERATE state</p> <ul style="list-style-type: none"> – '1', slave and master are exchanging their I/O data. – '0', slave and master are not exchanging their I/O data.
Notes on table: RO – Read only			

9.3 PROFIBUS Analog Input

The PROFIBUS Analog Input block provides a single analog output and is created in PROFIBUS Configurator when a measuring device, e.g. Cerabar M, is selected that is compliant with the PROFIBUS PA profile and uses the appropriate GSD. Application Designer maps the slave configuration and creates the blocks **Device Tag PA-AI-1**, **Device Tag PA-AI-2**, etc, according to the number of input variables selected.

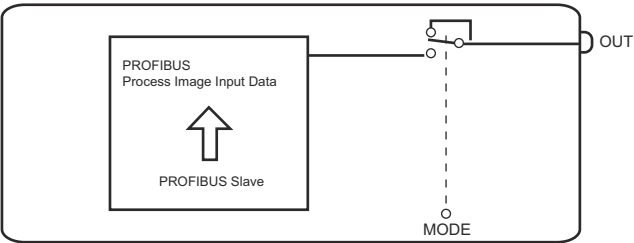


Fig. 9-3: Schematic diagram of the PROFIBUS Analog Input

9.3.1 Functional description

The PROFIBUS Analog Input block provides an output **OUT** of one process variable according to PROFIBUS PA Profile 3.0, represneted in the GSD by Identifier Byte 0x94 (5 Bytes: 4 bytes value in IEEE754 coding plus one byte status showing the PA-status) or the extended identifier format 0x42,0x84, 0x08, 0x05.

9.3.2 Block operation

The PROFIBUS Analog Input block normally operates in Auto mode.
If **MODE_BLK.Target** is set to Man, the **OUT** value and status can be overwritten.
The block can also be put out of service (OOS).

Status

Status	Meaning
GoodNC: NonSpecific	■ Block operating correctly, OUT value valid
Bad:NoCommunication, Last Usable Value	■ No PROFIBUS communication, last valid OUT value used
Bad:NoCommunication, No Usable Value	■ No PROFIBUS communication, no valid value (OUT = 0)
Bad:Out of Service	■ Block currently out of service

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Output Failure	■ No PROFIBUS communication or device failure
Out-of-Service	■ MODE_BLK.Target currently set to OOS

9.3.3 Block parameters

ParameterI	Valid Range	Default Value	Description
ST_REV	0 - 65535	0	See Chapter 2.3.4.
TAG_DESC		Spaces	
STRATEGY	0 - 65535	0	
ALERT_KEY	0 - 255	0	
MODE_BLK			Block mode, set to Auto
BLOCK_ERR			Block errors, see Chapter 5.2.3
OUT		0.0	Value and status of the block output
PI_INP_VAL_OFFSET	RO	–	Indicates the position in the process image of OUT value
PI_INP_STATUS_OFFSET	RO	–	Indicates the position in the process image of OUT status

9.4 PROFIBUS Multiple Analog Input

The PROFIBUS Multiple Analog Input block provides up to four analog outputs and is created in PROFIBUS Configurator when an analog or temperature input module of a PROFIBUS Remote I/O is selected. Application Designer maps the module and creates the block **Device Tag DP-MAI-1**. The number of blocks created depends on the hardware mapped in PROFIBUS Configurator.

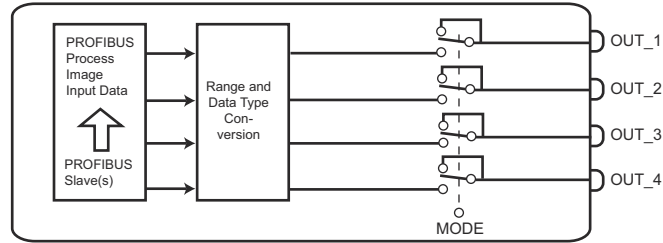


Fig. 9-4: Schematic diagram of the PROFIBUS Multiple Analog Input (DP-MAI) block

9.4.1 Functional description

The PROFIBUS Multiple Analog Input block provides four outputs **OUT_x** to downstream blocks.

The **SCALE_LOC_OUT_x** parameter expands to show information about the conversion and allows the individual scaling of each **OUT_x** value.

Scaling

The scaling parameters **FROM_EU_0%**, **FROM_EU_100%**, **TO_EU_0%**, **TO_EU_100%** must be set by the user to scale the input from the remote I/O to the desired **OUT_x** engineering units, see Fig. 9-5.

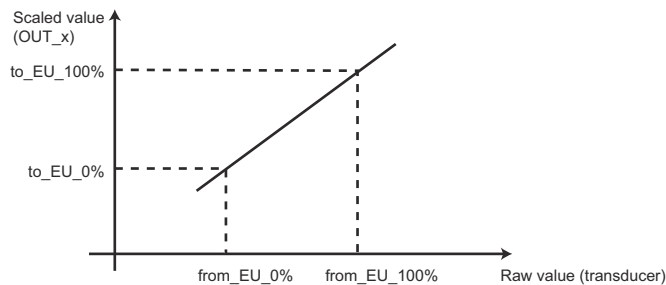


Fig. 9-5: Scaling of raw value to **OUT_x** value engineering units

OUT Status

The **OUT** signal must always have a valid status. In the case of remote I/Os that do not provide values with status, a status must be entered at **STATUS_SUBSTITUTION_VAL**. The default value 0x80 is "Good:NonCascade, NonLimited". Remote I/Os requiring this procedure are indicated in **STATUS_DETAILS** by the text "STATUS_NA_SUBST_VAL", see Chapter 9.4.4.

The parameter **STATUS_DETAILS** in **SCALE_LOC_OUT_x** specifies how the slave offers the status of the process value to the process image. This parameter must be set by the user, e.g. with the help of the mapping tool.

9.4.2 Block Configuration

It is assumed that a Pt100 thermocouple is connected to the first point of the Remote I/O. The temperature range -150°C to $+600^{\circ}\text{C}$ is represented by the Remote I/O as a range 0 to 8000.

The second third and fourth points of the analog inputs are connected to a pressure device with a range 0 mbar to 2000 mbar, a level device with a range 0% to 100% and a flow device with a range 0 l/min to 500 l/min. These outputs must also be scaled to the same Remote I/O range.

The DP_MAI block must be configured as follows: f

Parameter	Function	Example value
MODE_BLK.Target	Normal operating mode of block	Auto
SCALE_LOC_OUT_1 FROM_EU_0 FROM_EU_100 TO_EU_0 TO_EU_100	Lower range limit of input signal Upper range limit of input signal Lower range limit of OUT_1 Upper range limit of OUT_1	0 8000 -150 600
SCALE_LOC_OUT_2 FROM_EU_0 FROM_EU_100 TO_EU_0 TO_EU_100	Lower range limit of input signal Upper range limit of input signal Lower range limit of OUT_2 Upper range limit of OUT_2	0 8000 0 2000
SCALE_LOC_OUT_3 FROM_EU_0 FROM_EU_100 TO_EU_0 TO_EU_100	Lower range limit of input signal Upper range limit of input signal Lower range limit of OUT_3 Upper range limit of OUT_3	0 8000 0 100
SCALE_LOC_OUT_4 FROM_EU_0 FROM_EU_100 TO_EU_0 TO_EU_100	Lower range limit of input signal Upper range limit of input signal Lower range limit of OUT_4 Upper range limit of OUT_4	0 8000 0 500

9.4.3 Block operation

The PROFIBUS Analog Input block normally operates in Auto mode.

If **MODE_BLK.Target** is set to Man, the **OUT** value and status can be overwritten.

The block can also be put out of service (OOS).

Status

Status	Meaning
GoodNC:xxx	■ Block operating correctly, OUT value valid
Bad:NoCommunication, Last Usable Value	■ No PROFIBUS communication, last valid OUT value used
Bad:NoCommunication, No Usable Value	■ No PROFIBUS communication, no valid value (OUT = 0)
Bad:Out of Service	■ Block currently out of service

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Output Failure	■ No PROFIBUS communication or device failure
Out-of-Service	■ MODE_BLK.Target currently set to OOS

9.4.4 Block parameters

Parameter	Valid Range	Default Value	Description
ST_REV	0 - 65535	0	See Chapter 2.3.4.
TAG_DESC		Spaces	
STRATEGY	0 - 65535	0	
ALERT_KEY	0 - 255	0	
MODE_BLK			Block mode, set to Auto
BLOCK_ERR			Block errors, see Chapter 5.2.3
OUT_1			Value and status of the block output OUT_1
SCALE_LOC_OUT_1			Conversion and scaling of OUT_1
OUT_2			Value and status of the block output OUT_2
SCALE_LOC_OUT_2			Conversion and scaling of OUT_2
OUT_3			Value and status of the block output OUT_3
SCALE_LOC_OUT_3			Conversion and scaling of OUT_3
OUT_4			Value and status of the block output OUT_4
SCALE_LOC_OUT_4			Conversion and scaling of OUT_4

SCALE_LOC_OUT_x

Parameter	Valid Range	Default Value	Description
ACTIVE_FLAG		Disabled	Flags whether OUT_x value has been refreshed <ul style="list-style-type: none"> ■ Disable: OUT_x value not refreshed ■ Enabled: OUT_x value refreshed
PB_DATATYPE	2 ... 14	8	Profibus data type as communicated over PROFIBUS <ul style="list-style-type: none"> ■ 2: Integer8 ■ 3: Integer16 ■ 4: Integer32 ■ 5: Unsigned8 ■ 6: Unsigned16 ■ 7: Unsigned32 ■ 8: Floating Point ■ 9: Reserved: ■ 10: Integer16 in Little Endian format ■ 11: Integer32 in Little Endian format ■ 12: Unsigned16 in Little Endian format ■ 13:> Unsigned32 in Little Endian format ■ 14: Floating Point swapped, inverse byte order in comparison to that of PB spec.
PI_INP_VAL_OFFSET			Indicates the positions in the process image of OUT_x values
FROM_EU_0		0.0	Raw value from EU_0
FROM_EU_100		1.0	Raw value from EU_100
TO_EU_0		0.0	Scaled value to EU_0
TO_EU_100		1.0	Scaled value to EU_100
STATUS_DETAILS			Profibus Status Detail
STATUS_SUBSTITUTION_VAL		0x80	Status Substitution Value: status set by user for devices offering no status themselves, defaultvalue 0x80
PI_INP_STATUS_OFFSET	RO	–	Indicates the positions in the process image of OUT_x status; should these be provided by the Remote I/O
PV_TAG			Tag of process value mapped to OUT_x parameters

STATUS_DETAILS

No.	Type	Function
1	STANDARD_PA	Status is coded in one byte according to PA/FF <ul style="list-style-type: none"> Value for the status read from the process image input data at PI_INP_STATUS_OFFSET
2	MSB_BYTE_STATUS_0_GOOD	MSB of the byte with the value shows the status <ul style="list-style-type: none"> MSB = 0: OK => will result in 0x80 at status of OUTx MSB = 1: NOT OK => will result in 0x0 at status of OUTx
3	MSB_BYTE_STATUS_1_GOOD	MSB of the byte with the value shows the status <ul style="list-style-type: none"> MSB = 0: NOT OK => will result in 0x0 at status of OUTx MSB = 1: OK => will result in 0x80 at status of OUTx
4	MSB_WORD_STATUS_0_GOOD	MSB of the word with the value shows the status <ul style="list-style-type: none"> MSB = 0: OK => will result in 0x80 at status of OUTx MSB = 1: NOT OK => will result in 0x0 at status of OUTx
5	MSB_WORD_STATUS_1_GOOD	MSB of the word with the value is shows the status <ul style="list-style-type: none"> MSB = 0: NOT OK => will result in 0x0 at status of OUTx MSB = 1: OK => will result in 0x80 at status of OUTx
6	MSB_DOUBLEWORD_STATUS_0_GOOD	MSB of the double word with the value shows the status <ul style="list-style-type: none"> MSB = 0: OK => will result in 0x80 at status of OUTx MSB = 1: NOT OK => will result in 0x0 at status of OUTx
7	MSB_DOUBLEWORD_STATUS_1_GOOD	MSB of the word with the value shows the status <ul style="list-style-type: none"> MSB = 0: NOT OK => will result in 0x0 at status of OUTx MSB = 1: OK => will result in 0x80 at status of OUTx
8	STATUS_NA_SUBST_VAL	PB slave does not offer a status <ul style="list-style-type: none"> Value entered in STATUS_SUBSTITUTION_VAL will assigned to the status of OUTx
9	STATUS_NA_ALWAYS_OK	PB slave does not offer a status, <ul style="list-style-type: none"> 0x80 (GoodNonCascade, Notlimited, Nonspecific, will be assigned to the status of OUTx as long as no failure of the communication is signaled 0x18 (Bad, No Communication) will be assigned if there is a communication failure
80	BYTE_PACKED_STATI_0_GOOD_BIT=	Bit0 of value at PI_INP_STATUS_OFFSET shows the status
81 ... 87	BYTE_PACKED_STATI_0_GOOD_BITx	Bitx of value at PI_INP_STATUS_OFFSET shows the status
A0	BYTE_PACKED_STATI_1_GOOD_BIT0	Bit0 of value at PI_INP_STATUS_OFFSET shows the status <ul style="list-style-type: none"> BITx = 0: NOT OK => will result in 0x0 at status of OUTx BITx = 1: OK => will result in 0x80 at status of OUTx
A1 ... A7	BYTE_PACKED_STATI_1_GOOD_BITx	Bitx of value at PI_INP_STATUS_OFFSET shows the status <ul style="list-style-type: none"> BITx = 0: NOT OK => will result in 0x0 at status of OUTx BITx = 1: OK => will result in 0x80 at status of OUTx

9.5 PROFIBUS Digital Input

The PROFIBUS Discrete Input block provides a single discrete output and is created in PROFIBUS Configurator when a PROFIBUS switch, e.g. Liquiphant M, is selected. Normally a switch has only one process variable for which a single PROFIBUS Discrete Input block will be created. The block is assigned the generic name **Device Tag PA-DI-1**.

If the device offers more than one discrete output, then the configuration in PROFIBUS Configurator is analogous to that described in Chapter 9.2 and a block will be created for each process variable.

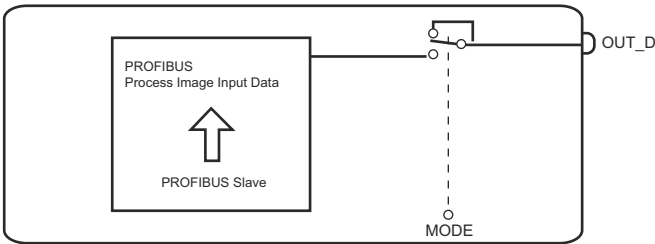


Fig. 9-6: Schematic diagram of the PROFIBUS Digital Input block

9.5.1 Functional description

The PROFIBUS Digital Input Block provides an output **OUT_D** to a downstream block.

9.5.2 Block operation

Mode

The PROFIBUS Analog Input block normally operates in Auto mode.

If **MODE_BLK.Target** is set to Man, the **OUT_D** value and status can be overwritten.

The block can also be put out of service (OOS).

Status	Meaning
GoodNC: NonSpecific	■ Block operating correctly, OUT value valid
Bad:NoCommunication, Last Usable Value	■ No PROFIBUS communication, last valid OUT_D value used
Bad:NoCommunication, No Usable Value	■ No PROFIBUS communication, no valid value (OUT = 0)
Bad:Out of Service	■ Block currently out of service

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Output Failure	■ No PROFIBUS communication or device failure
Out-of-Service	■ MODE_BLK.Target currently set to OOS

9.5.3 Block parameters

Parameter Label	Valid Range	Default Value	Description
ST_REV	0 ... 65535	0	See Chapter 2.3.4.
TAG_DESC		Spaces	
STRATEGY	0 ... 65535	0	
ALERT_KEY	0 to 255	0	
MODE_BLK			Block mode, set to Auto
BLOCK_ERR			Block errors, see Chapter 5.2.3
OUT_D			Value and status of the block output OUT_D
PI_INP_VAL_OFFSET	RO	–	Indicates the position in the process image of OUT_D value
PI_INP_STATUS_OFFSET	RO	–	Indicates the position in the process image of OUT_D status

9.6 PROFIBUS Multiple Discrete Input

The PROFIBUS Multiple Discrete Input block provides up to eight discrete outputs and is created in PROFIBUS Configurator when a discrete input module of a PROFIBUS Remote I/O is selected. Application Designer maps the device and creates the block **Device Tag DP-MDI-1**. The number of blocks created depends on the hardware mapped in PROFIBUS Configurator.

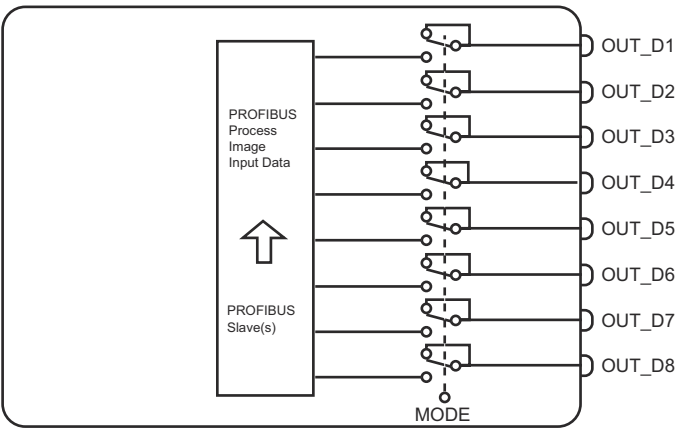


Fig. 9-7: Schematic diagram of the PROFIBUS Multiple Discrete Input

9.6.1 Functional description

The PROFIBUS Multiple Discrete Input block provides up to eight discrete outputs. Parameters are available at the **OUT_Dx** outputs as one byte value plus one byte status. In order to support the several datatypes provided by remote I/O systems, each process variable is converted before being placed in the output.

The **LOC_OUT_x** parameter expands to show information about the **OUT_Dx** value.

The **OUT_D** signal must always have a valid status. In the case of remote I/Os that do not provide values with status, a status must be entered at **STATUS_SUBSTITUTION_VAL**. The default value 0x80 is "Good:NonCascade, NonLimited". Remote I/Os requiring this procedure are indicated in **STATUS_DETAILS** by the text "STATUS_NA_SUBST_VAL", see Chapter 9.4.2.

9.6.2 Block operation

Mode

The PROFIBUS Analog Input block normally operates in Auto mode.
If **MODE_BLK.Target** is set to Man, the **OUT_Dx** value and status can be overwritten.
The block can also be put out of service (OOS).

Status

Status	Meaning
GoodNC:xxx	■ Block operating correctly, OUT value valid
Bad:NoCommunication, Last Usable Value	■ No PROFIBUS communication, last valid OUT value used
Bad:NoCommunication, No Usable Value	■ No PROFIBUS communication, no valid value (OUT = 0)
Bad:Out of Service	■ Block currently out of service

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Output Failure	■ No PROFIBUS communication or device failure
Out-of-Service	■ MODE_BLK.Target currently set to OOS

9.6.3 Block parameters

Block parameters

Parameter Label	Valid Range	Default Value	Description
ST_REV	0 ... 65535	0	See Chapter 2.3.4.
TAG_DESC		Spaces	
STRATEGY	0 ... 65535	0	
ALERT_KEY	0 to 255	0	
MODE_BLK			Block mode, set to Auto
BLOCK_ERR			Block errors, see Chapter 5.2.3
OUT_D1			Value and status of the block output OUT_D1
LOC_OUT_D1			Conversion of OUT_D1
OUT_D2			Value and status of the block output OUT_D2
LOC_OUT_D2			Conversion of OUT_D2
OUT_D3			Value and status of the block output OUT_D3
LOC_OUT_D3			Conversion of OUT_D3
OUT_D4			Value and status of the block output OUT_D4
LOC_OUT_D4			Conversion of OUT_D4
OUT_D5			Value and status of the block output OUT_D15
LOC_OUT_D5			Conversion of OUT_D5
OUT_D6			Value and status of the block output OUT_D6
LOC_OUT_D6			Conversion of OUT_D6
OUT_D7			Value and status of the block output OUT_D7
LOC_OUT_D7			Conversion of OUT_D7
OUT_D8			Value and status of the block output OUT_D8
LOC_OUT_D8			Conversion of OUT_D8

LOC_OUT_Dx

Component Label	Valid Range	Default Value	Description
ACTIVE_FLAG		Disabled	Flags whether OUT_x value has been refreshed <ul style="list-style-type: none"> ■ Disable: OUT_x value not refreshed ■ Enabled: OUT_x value refreshed
PI_INP_VAL_OFFSET			Indicates the positions in the process image of OUT_Dx values Default: 0, 2, 4, 6, 8, 10, 12, 14
VAL_BIT_OFFSET		0	Indicates the position of the bit in the byte referenced by PI_INP_VAL_OFFSET in which the binary value is to be found – A value other than zero will be indicates if the PROFIBUS device packs several binary values into one word/byte
STATUS_DETAILS			See Chapter 9.4.4
STATUS_SUBSTITUTION_VAL		0x80	Status Substitution Value: status set by user for devices offering no status themselves, defaultvalue 0x80
PI_INP_STATUS_OFFSET	RO	–	Indicates the positions in the process image of OUT_Dx status; should these be provided by the Remote I/O Default: 1, 3, 5, 7, 9, 11, 13, 15
PV_TAG			Tag of process value mapped to OUT_Dx parameters

9.7 PROFIBUS Totalizer

The PROFIBUS Totalizer block provides one analog totalizer output and is created in PROFIBUS Configurator when a device, e.g. a flowmeter, offers this process variable and it is selected. Application Designer maps the configuration and creates the blocks **Device Tag PA-TO-1**, **Device Tag PA-TO-2**, etc., depending upon the slave configuration.

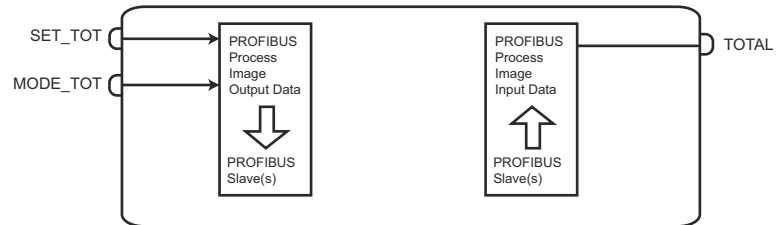


Fig. 9-8: Schematic diagram of the PROFIBUS Totalizer block

9.7.1 Functional description

The Profibus Totalizer block provides a totalized value at its output parameter **TOTAL** as four bytes value and one byte status.

The input **SET_TOT** allows the totalizer to be set by cyclically communicated data, i.e. by the control system. Three values are valid:

- 0: TOTALIZE – starts the totalizer
- 1: RESET – resets the totalizer to zero
- 2: PRESET – resets the totalizer to a preset value

If the option "PRESET" is selected in **SET_TOT**, a preset value must have been set in the flowmeter by either by using the display or a Class II master, e.g. FieldCare.

The input **MODE_TOT** allows the operating mode of the totalizer to be set by cyclically communicated data. Four values are valid:

- 0: BALANCED – the totalizer uses both negative and positive flows
- 1: POS_ONLY – the totalizer uses positive flows only
- 2: NEG_ONLY – the totalizer uses negative flows only
- 3: HOLD – the totalizer is stopped

The available totalizer options are dependent upon the device and are contained within its GSD. The inputs must be configured in PROFIBUS configurator by selecting an option and placing it in the appropriate slot.

Totalizer option	Action
TOTAL	The totalizer block supports totalizing by cyclic data exchange only – Reset are made by acyclic data exchange, e.g. by FieldCare
SETTOT_TOTAL (SET_TOTAL*)	The totalizer supports totalizing and setting by cyclic data exchange – Any PRESET value must be set by acyclic data exchange, e.g. by FieldCare
SETTOT_MODETOT_TOTAL (SET_MODE_TOTAL*)	The totalizer supports totalizing, setting and mode selection by cyclic data exchange
(SET_UNIT_PRESET_TOTAL*)	The totalizer supports totalizing, setting, specification of totalizer unit and specification of a preset value by cyclic data exchange
(SET_MODE_UNIT_PRESET_TOTAL*)	The totalizer supports totalizing, setting, mode selection, specification of totalizer unit and specification of a preset value by cyclic data exchange
* Prowirl 73 PA only	

The totalizer must be placed at the appropriate slot number. The table below summarizes the current possibilities for Endress+Hauser Profile 3.0 flowmeters:

Totalizer slots

Device	Slots	No.	Device	Slots	No.
Promass 83 PA	7, 8, 9	3	Prosonic DP*	3	1
Prosonic Flow 90 PA	4	1	Promag 53 DP/PA	2, 3, 4	3
Prowirl 72 PA*	3	1	Promass 80 PA	5	1
Prowirl 73 PA	5, 6	2	Promass 83 DP	7, 8, 9	3
Prosonic Flow 92 PA	5, 6	2	Prosonic Flow 93 DP/PA	9, 10, 11	3
Promag 50 PA/DP	2	1	T-mass 65	4, 5	2

9.7.2 Block operation

The PROFIBUS Totalizer block normally operates in Auto mode.

If **MODE_BLK.Target** is set to Man, the **TOTAL** value and status can be overwritten.

The block can also be put out of service (OOS).

Status

Status	Meaning
GoodNC:xxx	■ Block operating correctly, TOTAL value valid
Bad:NoCommunication, Last Usable Value	■ No PROFIBUS communication, last valid TOTAL value used
Bad:Out of Service	■ Block currently out of service

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Output Failure	■ No PROFIBUS communication or device failure
Out-of-Service	■ MODE_BLK.Target currently set to OOS

9.7.3 Block parameters

Parameter Label	Valid Range	Default Value	Description
ST_REV	0 ... 65535	0	See Chapter 2.3.4.
TAG_DESC		Spaces	
STRATEGY	0 ... 65535	0	
ALERT_KEY	0 to 255	0	
MODE_BLK		OOS	Block mode, set to Auto
BLOCK_ERR		0, 0	Block errors, see Chapter 5.2.3
TOTAL			Value and status of totalizer after execution of block
SET_TOT	0:	0	Totalizer setting 0: TOTALIZE 1: RESET 2: PRESET
MODE_TOT		0	Totalizer mode 0: BALANCED 1: POS_ONLY 2: NEG_ONLY 3: HOLD
PI_INP_TOTAL_OFFSET	RO	–	Position in the input process image of the TOTAL value
PI_INP_TOT_STAT_OFFSET	RO	–	Position in the input process image of the TOTAL status
PI_OUT_SET_TOT_OFFSET	RO	–	Position in the outout process image of the SET_TOT value
PI_OUT_MODE_TOT_OFFSET	RO	–	Position in the output process image of the MODE_TOT value
TOT_MODE_SELECT		0	Configuration of the totalizer block ■ 0: TOTAL only ■ 1: TOTAL plus SET_TOT ■ 2: TOTAL plus SET_TOT plus MODE_TOT

*A value 0xFFFF of an offset parameter means that there is no assignment of a PB input/output data to the corresponding TOTAL/SET_TOT/MODE_TOT parameter. In this case TOTAL/SET_TOT/MODE_TOT will not be cyclically refreshed

9.8 PROFIBUS Analog Output

The PROFIBUS Analog Output block provides one analog input to a transducer block and is created in PROFIBUS Configurator when an actuator is selected. Application Designer maps the slave configuration and creates the block **Device Tag PA-AO-1**.

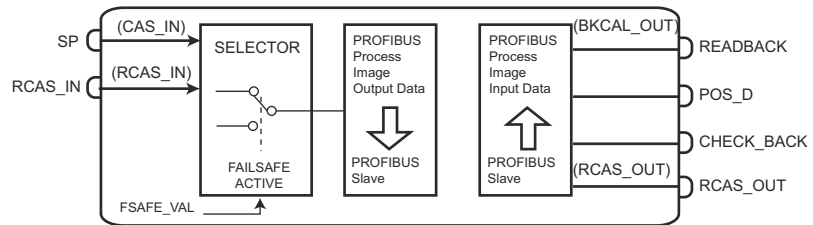


Fig. 9-9: Schematic diagram of the PROFIBUS Analog Output block

9.8.1 Functional Description

Depending on the configuration selected and control strategy, the Profibus Analog Output block receives its setpoint from:

- an upstream block, **SP** (=CAS_IN) or
- an external system, **RCAS_IN**

The setpoint is passed on to the device transducer block, which drives the actuator to the required position.

Depending on configuration, the block outputs the following transducer information:

- **READBACK:** current position and status of the actuator in units of PV_SCALE, whereby PV_SCALE is set in the transducer block and is usually 0% to 100% by default
- **POS_D:** Current position of actuator as a discrete value with status, indicating whether the valve is closed, open or in an intermediate position
- **CHECK_BACK:** Detailed information on the valve status in the form of a 3 byte bit map capable of carrying several messages at once
- **RCAS_OUT:** current position and status of the actuator in units of PV_SCALE for an external system, whereby PV_SCALE is set in the transducer block and is usually 0% to 100% by default

The available analog output options are dependent upon the device and are contained within its GSD. The inputs and outputs are activated by configuring the appropriate mode in the Slave Configuration dialogue of PROFIBUS Configurator:

Analog Output option	Action
SP	The output block uses the input SP from an upstream block only
SP/READBACK/POS_D	The output block uses the input SP from an upstream block and provides the outputs READBACK and POS_D
SP/CHECK_BACK	The output block uses the input SP from an upstream block and provides the output CHECK_BACK
SP/READBACK/POS_D/CHECK_BACK	The output block uses the input SP from an upstream block and provides the outputs READBACK, POS_D and CHECK_BACK
RCAS_IN/ RCAS_OUT	The output block the input RCAS_IN from an external application and provides the output RCAS_OUT
RCAS_IN/ RCAS_OUT/ CHECK_BACK	The output block uses the input RCAS_IN from an external application and provides the outputs RCAS_OUT and CHECK_BACK
SP/ READBACK/ RCAS_IN/ RCAS_OUT/ POS_D/ CHECK_BACK	The output block uses the input SP from an upstream block or the input RCAS_IN from an external application and provides the outputs READBACK, RCAS_OUT, POS_D and CHECK_BACK

The block supports the failsafe mechanism as described in Chapter 9.1.2.

9.8.2
Block configuration

Cascade control

Fig. 9-10 shows an example of cascade control using PROFIBUS devices. Chapter 6.1.4 has an identical application for FOUNDATION Fieldbus devices. The temperature of water measured upstream of a heat exchanger is used as input to a primary PID Control block which manipulates a secondary block controlling the supply of steam by opening and closing a control valve, see Fig. 9-11. The input to the secondary block is the flowrate of the steam. The flowmeter Analog Output block supplies the input value to the secondary loop, the temperature device Analog Input block the process value. The Analog Output block of the control valve controls the control valve actuator.

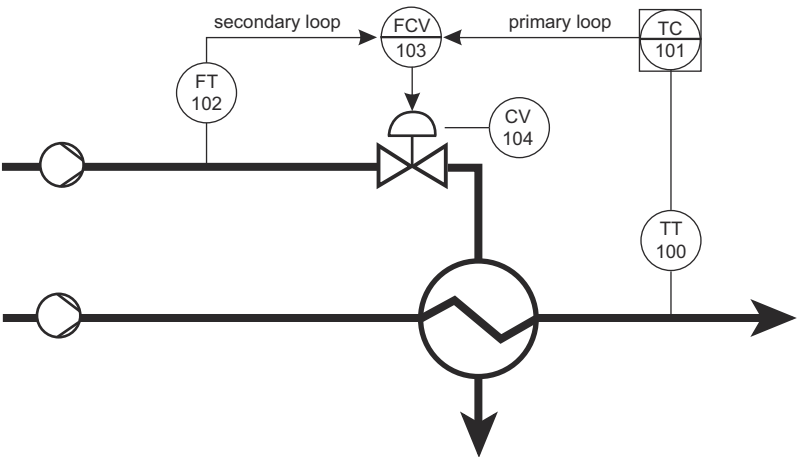


Fig. 9-10: Example of cascade control

Fig. 9-11 shows the control strategy together with links that must be made between the blocks. The PID blocks are standard function blocks which run in the SFC173 field controller. Note that the downstream PID links to the **SP** input and the back calculation is provided by the **READBACK** output. The **POS_D** and **CHECK_BACK** outputs are available to downstream blocks or a SCADA system.

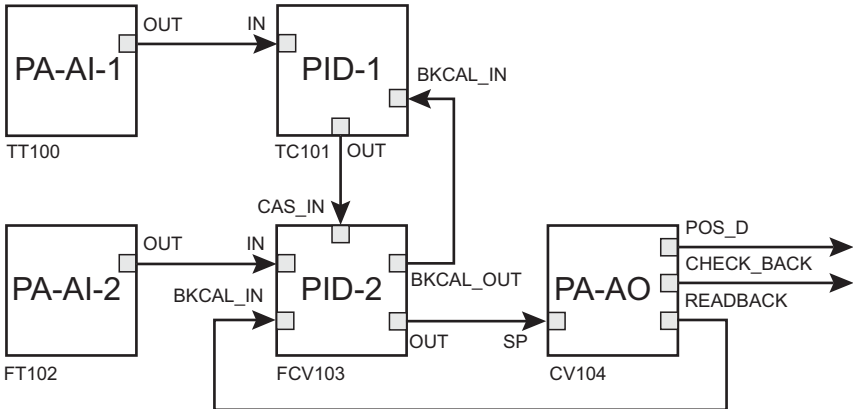


Fig. 9-11: Links required for cascade control

**Block configuration
cascade control**

PROFIBUS Configurator is used to create the Analog Input and Analog Output blocks. The measuring devices must be configured to supply an "AI" value. For the Analog Output the option "SP/READBACK/POS_D/CHECK_BACK" is selected.

The transducer blocks of the measuring devices are configured to provide an output scaled to 0% – 100% by means of acyclic communication, e.g. by using FieldCare.

The table below describes the configuration of the function blocks.

Parameter	Function	Example value
PB-AI-1		
MODE_BLK.Target	Normal operating mode of block	AUTO
PB-AI-2		
MODE_BLK.Target	Normal operating mode of block	AUTO
PID-1, primary loop		
MODE_BLK.Target	Normal operating mode of block	AUTO
SP	Setpoint for ideal process control	60%
PV_SCALE.EU_100	Upper range limit for process variable	100
PV_SCALE.EU_0	Lower range limit for process variable	0
PV_SCALE.UNITS_INDEX	Unit of process variable	%
OUT_SCALE/EU_100	Upper range limit for output variable	100
OUT_SCALE/EU_0	Lower range limit for output variable	0
OUT_SCALE/UNITS_INDEX	Unit of output variable	%
CONTROL_OPTS	Sets control options for bad input	Bypass Enable
SP_RATE_DN	Rate of change from old to new, higher SP	0
SP_RATE_UP	Rate of change from old to new, lower SP	0
GAIN	Tuning constants for the P, I and D terms of the PID block respectively.	1.5
RESET		0.1 s
RATE		0.5 s
SHED_OPT	Behaviour when shedding from remote mode	Normal shed, normal return
PID-2, secondary loop		
MODE_BLK.Target	Normal operating mode of block	CAS
PV_SCALE.EU_100	Upper range limit for process variable	100
PV_SCALE.EU_0	Lower range limit for process variable	0
PV_SCALE.UNITS_INDEX	Unit of process variable	%
OUT_SCALE/EU_100	Upper range limit for output variable	100
OUT_SCALE/EU_0	Lower range limit for output variable	0
OUT_SCALE/UNITS_INDEX	Unit of output variable	%
CONTROL_OPTS	Sets control options for bad input	Bypass Enable
SP_RATE_DN	Rate of change from old to new, higher SP	0
SP_RATE_UP	Rate of change from old to new, lower SP	0
GAIN	Tuning constants for the P, I and D terms of the PID block respectively.	1.5
RESET		0.1 s
RATE		0.5 s
SHED_OPT	Behaviour when shedding from remote mode	Normal shed, normal return
PB-AO		
MODE_BLK.Target	Normal operating mode of block	AUTO
FAILSAFE_MODE	Operating mode of failsafe function, see Chapter 9.1.2	Get FS Data
FAILSAFE_AVALUE	Value to be used on failsafe case, e.g. closed = 0%	0
FAILSAFE_STATUS	Status to be output on failsafe case ■ Bad NoUsableValue (00x14 = 20)	20
FAILSAFE_TIME	Time delay (s) between loss of input and failsafe case	3

Remote cascade

Fig. 9-12 shows an example of remote cascade control using a PROFIBUS actuator. Chapter 6.1.6 has an identical application for FOUNDATION Fieldbus devices. The temperature of water measured upstream of a heat exchanger and the flow rate measured downstream are acquired by an external Modbus controller and used as inputs to the cascade loop. The Modbus Control Slave block provides the setpoint to the Analog Output block, see Chapter 10.3, which drives the control valve actuator.

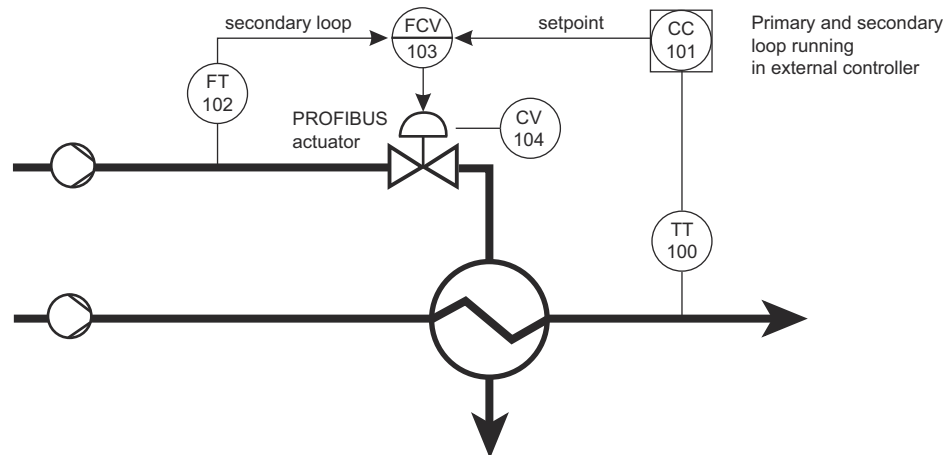


Fig. 9-12: Example of remote cascade

Fig. 9-13 shows the control strategy together with links that must be made between the blocks. Note that the Modbus Controller Slave block links to the **RCAS_IN** input and the back calculation is provided by the **RCAS_OUT** output. The **CHECK_BACK** output is available to a downstream block or a SCADA system.

Note



- The connection to the external application may also be made via OPC server, in which case the parameter **Device Tag-PB-AO-n.RCAS_IN** carries the setpoint, the parameter **Device Tag-PB-AO-n.RCAS_OUT** the feedback and the parameter **Device Tag-PB-AO-n.CHECK_BACK** the checkback data.

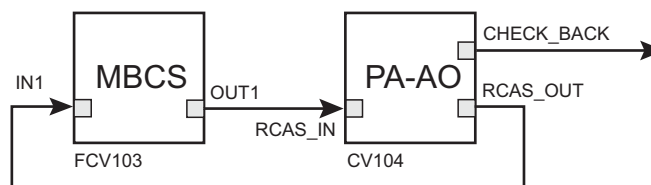


Fig. 9-13: Links required for remote cascade (through MBCS block)

**Block configuration
remote cascade**

PROFIBUS Configurator is used to create the Analog Output block. The option RCAS_IN/RCAS_OUT/ CHECK_BACK is selected in the Slave Configuration dialogue.

The configuration of the Modbus blocks is described in the Modbus Tutorial, BA037S/04/en. A timeout must be set in the MBCF block to ensure that the status of the OUT1 signal goes bad, should the controller fail. The MBCS block must be configured to provide an output scaled to 0% – 100%. For the strategy we require the IN_1, and OUT_1 channels. When LOCAL_MOD_MAP is set to zero the data will be read and written as follows:

,

Parameter	Channel	Register	Data type	Access
Positioner value	OUT1	40009	Float	Read
Status		40021		
Back calculation	IN1	40001	Float	Write
Status		40017		

The table below describes the configuration of the function blocks.

Parameter	Function	Example value
MBCF		
MODE_BLK.Target	Normal operating mode of block	AUTO
MEDIA	Media used to connect with controller	TCP/IP
MASTER_SLAVE	Operating mode of SFC173 controller	Slave
DEVICE_ADDRESS	Modbus device address	e.g. 1
TIMEOUT	Time (ms) before reaction to a loss in communication	1000
MBCS		
MODE_BLK.Target	Normal operating mode of block	AUTO
LOCAL_MOD_MAP	Register location for input and output data	0
SCALE_LOC_CONV_IN1	Scaling from RCAS_OUT parameter Scaling from RCAS_OUT parameter Scaling to controller Scaling to controller Data type	0
.FROM_EU_0		100
.FROM_EU_100		e.g. 0
.TO_EU_0		e.g. 100
.TO_EU_100		Float
.DATA_TYPE		
SCALE_LOC_CONV_OUT1	Scaling from controller Scaling from controller Scaling to RCAS_IN parameter Scaling to RCAS_IN parameter Data type Status to be transmitted to output when "Good"	e.g. 0
.FROM_EU_0		e.g. 1
.FROM_EU_100		0
.TO_EU_0		100
.TO_EU_100		Float
.DATA_TYPE		"Good_Cascade::Non Specific:NotLimited"
.STATUS_OUTPUT		
PB-AO		
MODE_BLK.Target	Normal operating mode of block	AUTO
FAILSAFE_MODE	Operating mode of failsafe function, see Chapter 9.1.2	Get FS Data
FAILSAFE_AVALUE	Value to be used on failsafe case, e.g. closed = 0%	0
FAILSAFE_STATUS	Status to be output on failsafe case ■ Bad NoUsableValue (00x14 = 20)	20
FAILSAFE_TIME	Time delay (s) between loss of input and failsafe case	3

9.8.3 Block operation

The PROFIBUS Analog Output block normally operates:

- in Auto mode when the setpoint is provided by the **SP** input
- in RCas mode when the setpoint is provided by the **RCAS_IN** input

The block supports the failsafe mechanism as described in Chapter 9.1.2.

If **MODE_BLK.Target** is set to Man, the **SP** or **RCAS_IN** value and status can be overwritten.

The block can also be put out of service (OOS).

Status

Status	Meaning
GoodNC:xxx	■ Block operating correctly, TOTAL value valid
Bad:NoCommunication, Last Usable Value	■ No PROFIBUS communication, last valid output value used
Bad:NoCommunication, No Usable Value	■ No PROFIBUS communication, output value = 0
Bad:Out of Service	■ Block currently out of service

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Output Failure	■ No PROFIBUS communication or device failure
Out-of-Service	■ MODE_BLK.Target currently set to OOS

9.8.4 Block parameters

Parameter/I	Valid Range	Default Value	Description
ST_REV	0 ... 65535	0	See Chapter 2.3.4.
TAG_DESC		Spaces	
STRATEGY	0 ... 65535	0	
ALERT_KEY	0 to 255	0	
MODE_BLK		OOS	Block mode, set to Auto for SP, RCas for RCAS_IN
BLOCK_ERR		0, 0	Block errors, see Chapter 5.2.3
SP		0, 0	Setpoint value and status in units of PV_SCALE
RCAS_IN			Remote setpoint value and status in units of PV scale provided by an external system
READBACK			Current position and status of the actuator in units of PV_SCALE
POS_D			Current position of actuator (discrete) with status <ul style="list-style-type: none"> ■ 0: Not initialized ■ 1: C+losed ■ 2: Open ■ 3: Intermediate position
CHECK_BACK			Detailed device information (3 bytes) as bit coded octet string
RCAS_OUT			Current position and status of the actuator in units of PV_SCALE for an external system
PI_OUT_SP_OFFSET	RO	–	Position in the input process image of the SP value
PI_OUT_SP_STAT_OFFSET	RO	–	Position in the input process image of the SP status
PI_OUT_RCAS_IN_OFFSET	RO	–	Position in the input process image of the RCAS_IN value
PI_OUT_RCAS_STAT_OFFSET	RO	–	Position in the input process image of the RCAS_IN status
PI_INP_RD_BACK_OFFSET	RO	–	Position in the input process image of the READBACK value
PI_INP_RD_BACK_STAT_OFFSET	RO	–	Position in the input process image of the READBACK status
PI_INP_POS_D_OFFSET	RO	–	Position in the input process image of the POS_D value
PI_INP_POS_D_STAT_OFFSET	RO	–	Position in the input process image of the POS_D status
PI_INP_CHK_BACK_OFFSET	RO	–	Position in the input process image of the CHECKBACK value
PI_INP_RCAS_OUT_OFFSET	RO	–	Position in the input process image of the RCAS_OUT value
PI_INP_RCAS_OUT_STAT_OFFSET	RO	–	Position in the input process image of the RCAS_OUT status

AO_MODE_SEL	0 - 6		Configuration of analog output block <ul style="list-style-type: none"> ■ 0: only SP ■ 1: SP, READBACK, POS_D; ■ 2: SP, CHECK_BACK; ■ 3: SP, READBACK, POS_D, CHECK_BACK; ■ 4: RCAS_IN, RCAS_OUT; ■ 5: RCAS_IN, RCAS_OUT, CHECK_BACK; ■ 6: SP, READBACK, RCAS_IN, RCAS_OUT, POS_D, CHECK_BACK
FAILSAFE_MODE	1 - 3		Selection of fault state mode <ul style="list-style-type: none"> ■ 1: Disable ■ 2: Get Last Data ■ 3: Get FS Data
FAILSAFE_STATUS	0 .. 0xFF		Status to be output when fault state active
FAILSAFE_AVALUE			FaultState Value
FAILSAFE_TIME)		FaultState Time

9.9 PROFIBUS Multiple Analog Output

The PROFIBUS Multiple Analog Output block provides four analog inputs to a transducer block and is created in PROFIBUS Configurator when an analog output module of a remote I/O is selected. Application Designer maps the slave configuration and creates the block **Device Tag DP-MAO-1**.

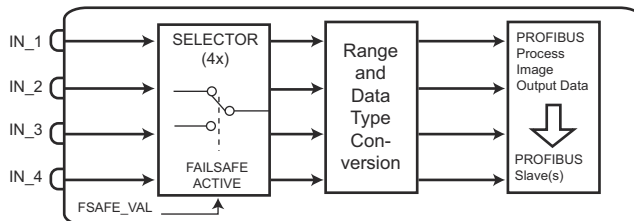


Fig. 9-14: Schematic diagram of the PROFIBUS Multiple Analog Output

9.9.1 Functional description

The PROFIBUS Multiple Analog Output block has four analog inputs **IN_1** to **IN_4** which drive the four outputs of the Remote I/O analog output module selected in PROFIBUS Configurator. In order to support the several datatypes provided by remote I/O systems, each process variable is converted before being placed in the output data image.

The **SCALE_LOC_IN_x** parameter expands to show information about the conversion and allow the scaling of the **IN_x** value.

PB_DATATYPE indicates the way in which the value and status are transmitted over PROFIBUS.

Scaling

The scaling parameters **FROM_EU_0%**, **FROM_EU_100%**, **TO_EU_0%**, **TO_EU_100%** must be set by the user to scale the **IN_x** value to the engineering units required by the remote I/O, see Fig. 9-15.

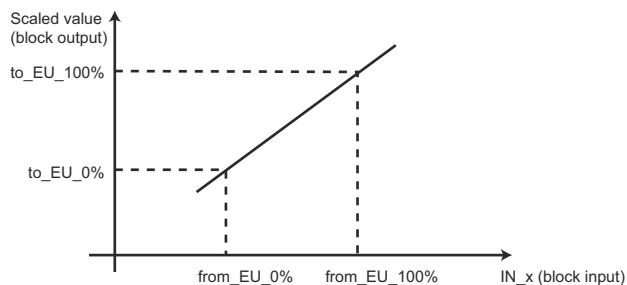


Fig. 9-15: Scaling of IN_x value to engineering units required by the remote I/O

SCALE_LOC_IN_x also allows a fault state behaviour can be set for each individual input by means of the parameters **FS_MODE**, **FS_STATUS**, **FS_AVALUE** and **FS_TIME**.

The parameter **STATUS_DETAILS** in **SCALE_LOC_IN_x** specifies whether and how the slave expects the status of **IN_x**. This parameter must be set by the user, e.g. with the help of the mapping tool.

9.9.2 Block operation

The PROFIBUS Multiple Analog Output block normally operates in Auto mode.

If **MODE_BLK.Target** is set to Man, the **IN_x** value and status can be overwritten.

The block can also be put out of service (OOS).

Status

Status	Meaning
GoodNC:xxx	■ Block operating correctly, OUT value valid
Bad:NoCommunication, Last Usable Value	■ No PROFIBUS communication, last valid OUT value used
Bad:NoCommunication, No Usable Value	■ No PROFIBUS communication, no valid value (OUT = 0)
Bad:Out of Service	■ Block currently out of service

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Output Failure	■ No PROFIBUS communication or device failure
Out-of-Service	■ MODE_BLK.Target currently set to OOS

9.9.3 Block Configuration

It is assumed that **IN_1** is scaled to 0% to 100% and the actuator connected to the first point of the Remote I/O expects a signal of 4 mA to 20 mA. **IN_2** is scaled to 0% to 100% and the actuator connected to the second point of the Remote I/O expects a signal of 4 mA to 20 mA. **IN_3** is scaled to 0% to 100% and the actuator connected to the third point of the Remote I/O expects a signal of 0 V to 10 V. **IN_4** is not used. The DP_MOI block must be configured as follows: f

Parameter	Function	Example value
MODE_BLK.Target	Normal operating mode of block	Auto
SCALE_LOC_IN_1		
FROM_EU_0	IN_1 lower range limit	0
FROM_EU_100	IN_1 upper range limit	100
TO_EU_0	Actuator output lower range limit	4
TO_EU_100	Actuator output upper range limit	20
FS_MODE	Operating mode of failsafe function, see Chapter 9.1.2	Get FS Data
FS_STATUS	Status to be output on failsafe case	20
	Bad NoUsableValue (00x14 = 20)	
FS_AVALUE	Value to be used on failsafe case, e.g. closed = 4 mA	4
FS_TIME	Time delay (s) between loss of input and failsafe case	3
SCALE_LOC_IN_2		
FROM_EU_0	IN_2 lower range limit	0
FROM_EU_100	IN_2 upper range limit	100
TO_EU_0	Actuator output lower range limit	4
TO_EU_100	Actuator output upper range limit	20
FS_MODE	Operating mode of failsafe function, see Chapter 9.1.2	Get FS Data
FS_STATUS	Status to be output on failsafe case	20
	Bad NoUsableValue (00x14 = 20)	
FS_AVALUE	Value to be used on failsafe case, e.g. closed = 4 mA	4
FS_TIME	Time delay (s) between loss of input and failsafe case	3
SCALE_LOC_IN_3		
FROM_EU_0	IN_3 lower range limit	0
FROM_EU_100	IN_3 upper range limit	100
TO_EU_0	Actuator output lower range limit	0
TO_EU_100	Actuator output upper range limit	10
FS_MODE	Operating mode of failsafe function, see Chapter 9.1.2	Get FS Data
FS_STATUS	Status to be output on failsafe case	20
	Bad NoUsableValue (00x14 = 20)	
FS_AVALUE	Value to be used on failsafe case, e.g. closed = 0 V	0
FS_TIME	Time delay (s) between loss of input and failsafe case	3

9.9.4 Block parameters

Parameterl	Valid Range	Default Value	Description
ST_REV	0 ... 65535	0	See Chapter 2.3.4.
TAG_DESC		Spaces	
STRATEGY	0 ... 65535	0	
ALERT_KEY	0 to 255	0	
MODE_BLK			Block mode, set to Auto
BLOCK_ERR			Block errors, see Chapter 5.2.3
IN_1			Value and status of the block input IN_1
SCALE_LOC_IN_1			Conversion of IN_1
IN_2			Value and status of the block input IN_2
SCALE_LOC_IN_2			Conversion of IN_2
IN_3			Value and status of the block input IN_3
SCALE_LOC_IN_3			Conversion of IN_3
IN_4			Value and status of the block input IN_4
SCALE_LOC_IN_4			Conversion of IN_4

SCALE_LOC_IN_x

Parameterl	Valid Range	Default Value	Description
ACTIVE_FLAG		Disabled	Flags whether IN_x value has been refreshed <ul style="list-style-type: none"> ■ Disable: IN_x value not refreshed ■ Enabled: IN_x value refreshed
PB_DATATYPE	2 ... 14	8	Profibus data type as communicated over PROFIBUS <ul style="list-style-type: none"> ■ 2: Integer8 ■ 3: Integer16 ■ 4: Integer32 ■ 5: Unsigned8 ■ 6: Unsigned16 ■ 7: Unsigned32 ■ 8: Floating Point ■ 9: Reserved: ■ 10: Integer16 in Little Endian format ■ 11: Integer32 in Little Endian format ■ 12: Unsigned16 in Little Endian format ■ 13:> Unsigned32 in Little Endian format ■ 14: Floating Point swapped, inverse byte order in comparison to that of PB spec.
PI_OUT_VAL_OFFSET		0, 5, 10, 15	Indicates the positions in the process image of IN_x values
FROM_EU_0		0.0	Raw value from EU_0
FROM_EU_100		1.0	Raw value from EU_100
TO_EU_0		0.0	Scaled value to EU_0
TO_EU_100		1.0	Scaled value to EU_100
STATUS_DETAILS			Profibus Status Detail, see next table
PI_OUT_STATUS_OFFSET		4, 8, 12, 16	Indicates the positions in the process image of IN_x status; should these be provided by the Remote I/O
PV_TAG			Tag of process value mapped to OUT_x parameters
FS_MODE			Selection of fault state mode <ul style="list-style-type: none"> ■ 1: Disable ■ 2: Get Last Data ■ 3: Get FS Data
FS_STATUS	0 .. 0xFF		Status to be output when fault state active
FS_AVALUE			FaultState Value
FS_TIME			FaultState Time

STATUS_DETAILS

No.	Type	Function
1	STANDARD_PA	Status is coded in one byte according to PA/FF ■ Value for the status written to the process image output data at PI_OUT_STATUS_OFFSET
2	MSB_BYTE_STATUS_0_GOOD	MSB of the byte with the value sends the status: ■ If bit 7 of status of IN_x is set to '1' then MSB = 0: OK, else MSB = 1: NOT OK
3	MSB_BYTE_STATUS_1_GOOD	MSB of the byte with the value sends the status: ■ If bit 7 of status of IN_x is set to '1' then MSB = 1: OK, else MSB = 0: NOT OK
4	MSB_WORD_STATUS_0_GOOD	MSB of the word with the value sends the status: ■ If bit 7 of status of IN_x is set to '1' then MSB = 0: OK, else MSB = 1: NOT OK
5	MSB_WORD_STATUS_1_GOOD	MSB of the word with the value sends the status: ■ If bit 7 of status of IN_x is set to '1' then MSB = 1: OK, else MSB = 0: NOT OK
6	MSB_DOUBLEWORD_STATUS_0_GOOD	MSB of the double word with the value sends the status: ■ If bit 7 of status of IN_x is set to '1' then MSB = 0: OK, else MSB = 1: NOT OK
7	MSB_DOUBLEWORD_STATUS_1_GOOD	MSB of the double word with the value sends the status: ■ If bit 7 of status of IN_x is set to '1' then MSB = 1: OK, else MSB = 0: NOT OK
8	STATUS_NA_SUBST_VAL	PB slave does not require a status; status of IN_x not used
9	Reserved	
80	BYTE_PACKED_STATI_0_GOOD_BIT=	Bit0 of value at PI_OUT_STATUS_OFFSET sends the status ■ If bit7 of status of IN_x is set to '1' then Bit0 = 0 (OK), else Bit0 = 1 (NOT OK)
81 ... 87	BYTE_PACKED_STATI_0_GOOD_BITx	BitX of value at PI_OUT_STATUS_OFFSET sends the status ■ If bit7 of status of IN_x is set to '1' then BitxX = 0 (OK), else BitxX = 1 (NOT OK)
A0	BYTE_PACKED_STATI_1_GOOD_BIT0	Bit0 of value at PI_OUT_STATUS_OFFSET sends the status ■ If bit7 of status of IN_x is set to '1' then Bit0 = 1 (OK), else Bit0 = 0 (NOT OK)
A1 ... A7	BYTE_PACKED_STATI_1_GOOD_BITx	BitX of value at PI_OUT_STATUS_OFFSET sends the status ■ If bit7 of status of IN_x is set to '1' then BitxX = 1 (OK), else BitxX = 0 (NOT OK)

9.10 PROFIBUS Discrete Output

The PROFIBUS Discrete Output block provides one discrete input to a transducer block and is created in PROFIBUS Configurator when e.g. an on/off actuator is selected. Application Designer maps the slave configuration and creates the block **Device Tag PA-DO-1**.

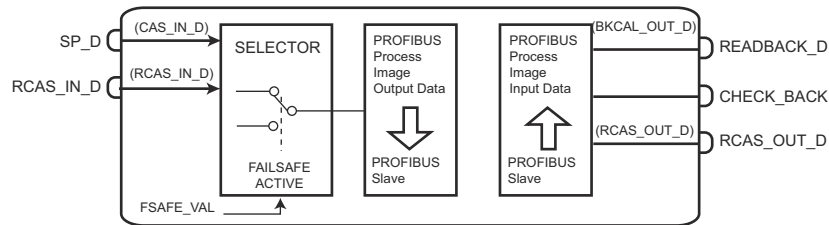


Fig. 9-16: Schematic diagram of the PROFIBUS Discrete Output

9.10.1 Functional description

Depending on the configuration selected and control strategy, the Profibus Discrete Output block receives its setpoint from:

- an upstream block, **SP_D** (=CAS_IN_D) or
- an external system, **RCAS_IN_D**

The setpoint is passed on to the device transducer block, which drives the actuator to the required position.

Depending on configuration, the block outputs the following transducer information:

- **READBACK_D**: current position and status of the actuator
- **CHECK_BACK**: Detailed information on the valve status in the form of a 3 byte bit map capable of carrying several messages at once
- **RCAS_OUT_D**: current position and status of the actuator for an external system

The available discrete output options are dependent upon the device and are contained within its GSD. The inputs and outputs are activated by configuring the appropriate mode in the Slave Configuration dialogue of PROFIBUS Configurator:

Discrete Output option	Action
SP_D	The output block uses the input SP_D from an upstream block only
SP_D/READBACK_D	The output block uses the input SP_D from an upstream block and provides the output READBACK_D
SP_D/CHECK_BACK	The output block uses the input SP_D from an upstream block and provides the output CHECK_BACK
SP_D/READBACK_D/CHECK_BACK	The output block uses the input SP_D from an upstream block and provides the outputs READBACK_D and CHECK_BACK
RCAS_IN_D/ RCAS_OUT_D	The output block the input RCAS_IN_D from an external application and provides the output RCAS_OUT_D
RCAS_IN_D/ RCAS_OUT_D/ CHECK_BACK	The output block uses the input RCAS_IN_D from an external application and provides the outputs RCAS_OUT_D and CHECK_BACK
SP_D/ READBACK_D/ RCAS_IN_D/ RCAS_OUT_D/ CHECK_BACK	The output block uses the input SP_D from an upstream block or the input RCAS_IN_D from an external application and provides the outputs READBACK_D, RCAS_OUT_D and CHECK_BACK

The block supports the failsafe mechanism as described in Chapter 9.1.2.

9.10.2 Block configuration

Fig. 9-17 shows an example of on-off control of process tank level that uses the PROFIBUS Discrete Output block. The level in a process tank with random inflow is controlled by opening and closing the outlet valve (LV100) based on the level sensed in the tank by the Liquiphant level switch (LSH100). The Liquiphant outputs a true signal when it is covered with liquid; the valve is operated by a solenoid and is open when the corresponding point of the Remote I/O this is activated. The checkback information is assessed by the SCADA system.

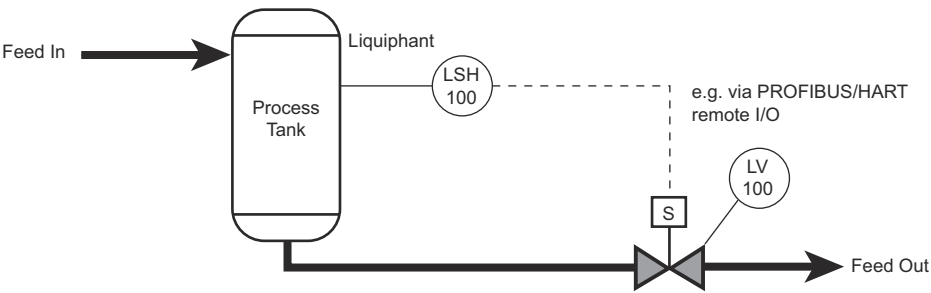


Fig. 9-17: Example of on-off control of process tank level

Fig. 8-3 shows the control strategy together with links that must be made between the blocks.

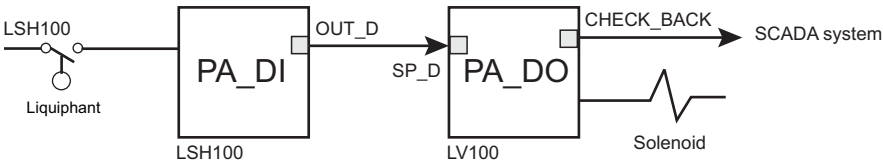


Fig. 9-18: Links required for basic closed-loop control

Block configuration

PROFIBUS Configurator is used to create the Digital Input Block of the Liquiphant and Digital Output block for the PROFIBUS on-off valve. For the Liquiphant the module "Discrete In" is selected in the Slave Configuration dialog, for the valve the module "SP_D/ CHECK_BACK"..

Parameter	Function	Example value
PA_DI block		
MODE_BLK.Target	Normal operating mode of block	Auto
PA_DO block		
MODE_BLK.Target	Normal operating mode of block	Cas
FS_MODE	Operating mode of failsafe function, see Chapter 9.1.2	Get FS value
FS_STATUS	Status to be output on failsafe case Bad NoUsableValue (00x14 = 20)	20
FS_DVALUE	Fail-to-safe value for valve (= open, prevents overspill)	1
FS_TIME	Time delay (s) between loss of input and failsafe case	3

9.10.3 Block operation

Mode

The PROFIBUS Analog Output block normally operates:

- in Auto mode when the setpoint is provided by the **SP_D** input
- in RCas mode when the setpoint is provided by the **RCAS_IN_D** input

The block supports the failsafe mechanism as described in Chapter 9.1.2.

If **MODE_BLK.Target** is set to Man, the **SP** or **RCAS_IN** value and status can be overwritten.

The block can also be put out of service (OOS).

Status

Status	Meaning
GoodNC:xxx	■ Block operating correctly, TOTAL value valid
Bad:NoCommunication, Last Usable Value	■ No PROFIBUS communication, last valid output value used
Bad:NoCommunication, No Usable Value	■ No PROFIBUS communication, output value = 0
Bad:Out of Service	■ Block currently out of service

9.10.4 Block parameters

Parameter/I	Valid Range	Default Value	Description
ST_REV	0 ... 65535	0	See Chapter 2.3.4.
TAG_DESC		Spaces	
STRATEGY	0 ... 65535	0	
ALERT_KEY	0 to 255	0	
MODE_BLK			Block mode, set to Auto for SP_D, RCas for RCAS_IN_D
BLOCK_ERR		0, 0	Block errors, see Chapter 5.2.3
SP_D		0, 0	Discrete setpoint value and status
RCAS_IN_D			Remote discrete setpoint value and status
READBACK_D			Current position and status of the actuator
CHECK_BACK			Detailed device information (3 bytes) as bit coded octet string
RCAS_OUT_D			Current position and status of the actuator for external system
PI_OUT_SP_D_OFFSET		0xFFFF	Position in the input process image of the SP_D value
PI_OUT_SP_STAT_D_OFFSET		0xFFFF	Position in the input process image of the SP_D status
PI_OUT_RCAS_IN_D_OFFSET		0xFFFF	Position in the input process image of the RCAS_IN_D value
PI_OUT_RCAS_STAT_OFFSET		0xFFFF	Position in the input process image of the RCAS_IN_D status
PI_INP_RD_BACK_D_OFFSET		0xFFFF	Position in the input process image of the READBACK_D value
PI_INP_RD_BACK_STAT_OFFSET		0xFFFF	Position in the input process image of the READBACK_D status
PI_INP_CHK_BACK_OFFSET		0xFFFF	Position in the input process image of the CHECKBACK value
PI_INP_RCAS_OUT_D_OFFSET		0xFFFF	Position in the input process image of the RCAS_OUT_D value
PI_INP_RCAS_OUT_STAT_OFFSET		0xFFFF	Position in the input process image of the RCAS_OUT_D status
DO_MODE_SEL			Configuration of discrete output block 0: only SP_D 1: SP_D, READBACK_D; 2: SP_D, CHECK_BACK; 3: SP_D, READBACK_D, CHECK_BACK; 4: RCAS_IN_D, RCAS_OUT_D; 5: RCAS_IN_D, RCAS_OUT_D, CHECK_BACK; 6: SP_D, READBACK_D, RCAS_IN_D, RCAS_OUT_D, CHECK_BACK
FS_MODE			Selection of fault state mode 1: Disable 2: Get Last Data 3: Get FS Data
FS_STATUS			Status to be output when fault state active
FS_DVALUE			FaultState Value
FS_TIME			FaultState Time

9.11 PROFIBUS Multiple Discrete Output

The PROFIBUS Multiple Discrete Output block provides eight discrete inputs to a transducer block and is created in PROFIBUS Configurator when a discrete output module of a PROFIBUS Remote I/O is selected. Application Designer maps the slave configuration and creates the block **Device Tag DP-MDO-1**.

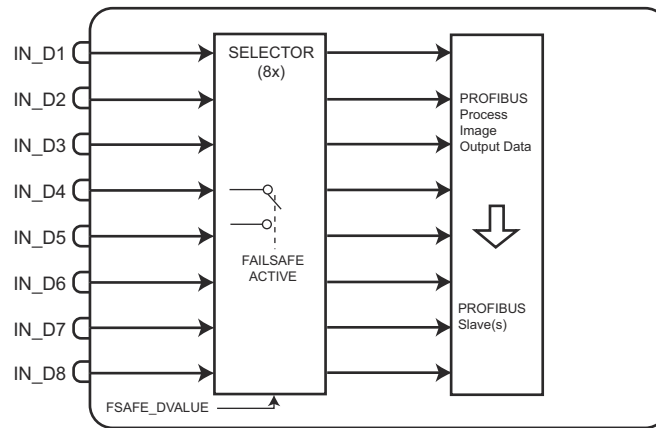


Fig. 9-19: Schematic diagram of the PROFIBUS Multiple Discrete Output block

9.11.1 Functional description

Description

The PROFIBUS Multiple Discrete Output block has eight discrete inputs **IN_D1** to **IN_D8** which drive the eight outputs of the Remote I/O discrete output module selected in PROFIBUS Configurator.

The **LOC_IN_x** parameter expands to show information about the **IN_Dx** value. A fault state behaviour can be set within this parameter by means of the parameters **FS_MODE**, **FS_STATUS**, **FS_DVALUE** and **FS_TIME**.

9.11.2 Block configuration

Fig. 9-20 shows an example of a control strategy using the PROFIBUS Multiple Discrete Output block and four channels of the 8-channel NO relay module of a PROFIBUS Remote I/O.

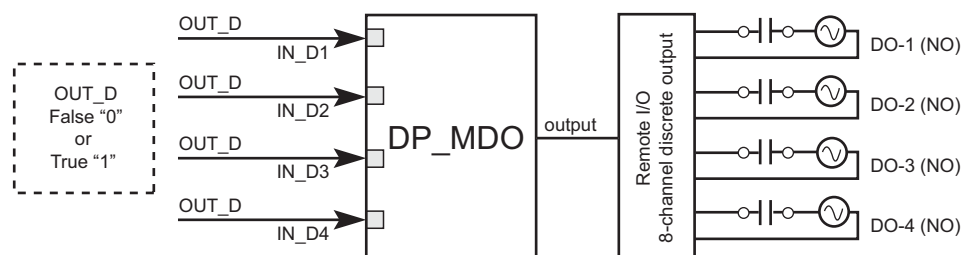


Fig. 9-20: Example of links required for Multiple Discrete Output block used with Remote I/O

In the example, only the first four inputs are used:

- IN_D1 drives the DO-1 output
- IN_D2 drives the DO-2 output
- IN_D3 drives the DO-3 output
- IN_D4 drives the DO-4 output

The block configuration comprises setting the block mode and fault state values.

Block configuration

The Multiple Discrete Output block is created according to the module selected for the Remote I/O in PROFIBUS Configurator. The way in which the value and status are transferred is configured in the PROFIBUS mapping tool.

Parameter	Function	Example value
DP_MDO block		
MODE_BLK.Target	Normal operating mode of block	Auto
LOC_IN_D1 FS_MODE FS_STATUS FS_AVALUE FS_TIME	Operating mode of failsafe function, see Chapter 9.1.2 IN_D1 status to be output on failsafe case Bad NoUsableValue (00x14 = 20) IN_D1 value to be used on failsafe case, e.g. open = 0 Time delay (s) between loss of input and failsafe case	Get FS Data 20 0 3
LOC_IN_D2 FS_MODE FS_STATUS FS_AVALUE FS_TIME	Operating mode of failsafe function, see Chapter 9.1.2 IN_D2 status to be output on failsafe case Bad NoUsableValue (00x14 = 20) IN_D2 value to be used on failsafe case, e.g. open = 0 Time delay (s) between loss of input and failsafe case	Get FS Data 20 0 3
LOC_IN_D3 FS_MODE FS_STATUS FS_AVALUE FS_TIME	Operating mode of failsafe function, see Chapter 9.1.2 IN_D3 status to be output on failsafe case Bad NoUsableValue (00x14 = 20) IN_D3 value to be used on failsafe case, e.g. open = 0 Time delay (s) between loss of input and failsafe case	Get FS Data 20 0 3
LOC_IN_D4 FS_MODE FS_STATUS FS_AVALUE FS_TIME	Operating mode of failsafe function, see Chapter 9.1.2 IN_D4 status to be output on failsafe case Bad NoUsableValue (00x14 = 20) IN_D4 value to be used on failsafe case, e.g. open = 0 Time delay (s) between loss of input and failsafe case	Get FS Data 20 0 3

9.11.3 Block operation

The PROFIBUS Multiple Discrete Output block normally operates in Auto mode.

If **MODE_BLK.Target** is set to Man, the **IN_Dx** value and status can be overwritten.

The block can also be put out of service (OOS).

Status

Status	Meaning
GoodNC:xxx	■ Block operating correctly, OUT value valid
Bad:NoCommunication, Last Usable Value	■ No PROFIBUS communication, last valid OUT value used
Bad:NoCommunication, No Usable Value	■ No PROFIBUS communication, no valid value (OUT = 0)
Bad:Out of Service	■ Block currently out of service

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Output Failure	■ No PROFIBUS communication or device failure
Out-of-Service	■ MODE_BLK.Target currently set to OOS

9.11.4 Block parameters

Parameter	Valid Range	Default Value	Description
ST_REV	0 ... 65535	0	See Chapter 2.3.4.
TAG_DESC		Spaces	
STRATEGY	0 ... 65535	0	
ALERT_KEY	0 to 255	0	
MODE_BLK			Block mode, set to Auto
BLOCK_ERR			Block errors, see Chapter 5.2.3
IN_D1			Value and status of the block input IN_D1
LOC_IN_D1			Conversion of IN_D1
IN_D2			Value and status of the block input IN_D2
LOC_IN_D2			Conversion of IN_D2
IN_D3			Value and status of the block input IN_D3
LOC_IN_D3			Conversion of IN_D3
IN_D4			Value and status of the block input IN_D4
LOC_IN_D4			Conversion of IN_D4
IN_D5			Value and status of the block input IN_D5
LOC_IN_D5			Conversion of IN_D5
IN_D6			Value and status of the block input IN_D6
LOC_IN_D6			Conversion of IN_D6
IN_D7			Value and status of the block input IN_D7
LOC_IN_D7			Conversion of IN_D7
IN_D8			Value and status of the block input IN_D8
LOC_IN_D8			Conversion of IN_D8

LOC_IN_Dx

Parameter	Valid Range	Default Value	Description
ACTIVE_FLAG		Disabled	Flags whether IN_Dx value has been refreshed <ul style="list-style-type: none"> ■ Disable: IN_Dx value not refreshed ■ Enabled: IN_Dx value refreshed
PI_OUT_VAL_OFFSET		0, 5, 10, 15	Indicates the positions in the process image of IN_Dx values
VAL_BIT_OFFSET		0.0	Indicates at which bit of the byte referenced by PI_OUT_VAL_OFFSET the binary value is to be written, <ul style="list-style-type: none"> ■ Default value bit0; ■ Another location is normal when several binary values are packed into one byte/word
STATUS_DETAILS			See Chapter 9.9.2
PI_OUT_STATUS_OFFSET		4, 8, 12, 16	Indicates the positions in the process image of IN_x status; should these be provided by the Remote I/O
PV_TAG			Tag of process value mapped to OUT_x parameters
FS_MODE			Selection of fault state mode <ul style="list-style-type: none"> ■ 1: Disable ■ 2: Get Last Data ■ 3: Get FS Data
FS_STATUS	0 .. 0xFF		Status to be output when fault state active
FS_DVALUE			FaultState Value
FS_TIME			FaultState Time

Status details can be taken from the table in Chapter 9.9.2

10 Modbus Blocks

10.1 ControlCare Implementation

ControlCare Field Controllers SFC162 and SFC173 are equipped with both a Modbus serial and Ethernet interface. As a result, they can be operated in one of the following roles:

- Modbus Serial or TCP Master
- Modbus Serial or TCP Slave
- Modbus TCP Master and Serial Slave, Modbus TCP Slave and Serial Master or Modbus TCP Master and TCP Slave

The Modbus role and corresponding parameters are configured in the Modbus Configuration Block, MBCF. As serial master, a Field Controller supports up to 128 Modbus slaves; as TCP master eight Modbus slaves only. In serial (RTU) operation baudrates from 9600 to 115200 are possible. If configured as a serial or TCP slave, a field controller can be accessed by one Modbus master only.

Establishing communication

Modbus communication is initially established by pressing the **ON_APPLY** button in the MBCF block. If changes are made to the Modbus configuration, the button must always be pressed to re-establish communication with the new parameters.

On recovery after a power failure, the Field Controller resumes communication automatically within 30 seconds. If a Modbus slave device loses power or is disconnected, the Field Controller will continue to poll it until it recovers, is re-connected or is removed from the strategy.

When the Field Controller is configured as a TCP slave, access to the Modbus registers is controlled by entering the IP address of the Modbus master. The Field Controller will normally communicate with the master via Port 502. If this port is unavailable, it is possible to specify a secondary TCP/IP port number.

Modbus commands

Field Controller supports the following Modbus functions

Function	Function Code	Hex	Applies To
Read discrete inputs	2	0x02	Master, Slave
Read coils	1	0x01	Master, Slave
Write single coil	5	0x05	Master, Slave
Write multiple coils	15	0x0F	Master, Slave
Read input register	4	0x04	Master, Slave
Read holding register	3	0x03	Master, Slave
Write single register	6	0x06	Master, Slave
Write multiple registers	16	0x10	Master, Slave
Read/Write multiple register	23	0x17	Slave only

Registers

Modbus specifies four different types of register:

- Discrete input registers contain the discrete input values and possibly status
- Input registers contain analog input values and status
- Coil registers contain discrete output values and possibly status
- Holding registers contain analog output values and status

In addition, it is possible to "pack" discrete inputs and outputs into words, which are then stored as appropriate in the input or holding registers. Fig. 2-1 gives an overview of the register and reference address ranges used for each register type in Field Controller. The table shows the relationship between the function block input and output parameters and the registers for master and slave roles.

FB Parameter	Field Controller as Modbus Master	Field Controller as Modbus Slave
IN_D1 to IN_D4	Value written to slave coil	Value read by master from discrete input
OUT_D1 to OUT_D4	Value read from slave discrete input/coil	Value written by master to coil
IN_1 to IN_4	Value written to slave holding register	Value read from master from input register
OUT_1 to OUT_4	Value read from slave input/holding register	Value written by master to holding register

10.1.1 Register access

Depending upon the Modbus implementation, the registers are accessed by specifying:

- the function code and the register address or
- the reference address

When Field Controller is used as a master, the addresses are entered in the **LOCATOR** and **SCALE_LOC** parameters. When it is acting as a slave, it offers the values by means of an address table which is dependent upon the unique MBCS block identifier **LOCAL_MOD_MAP**.

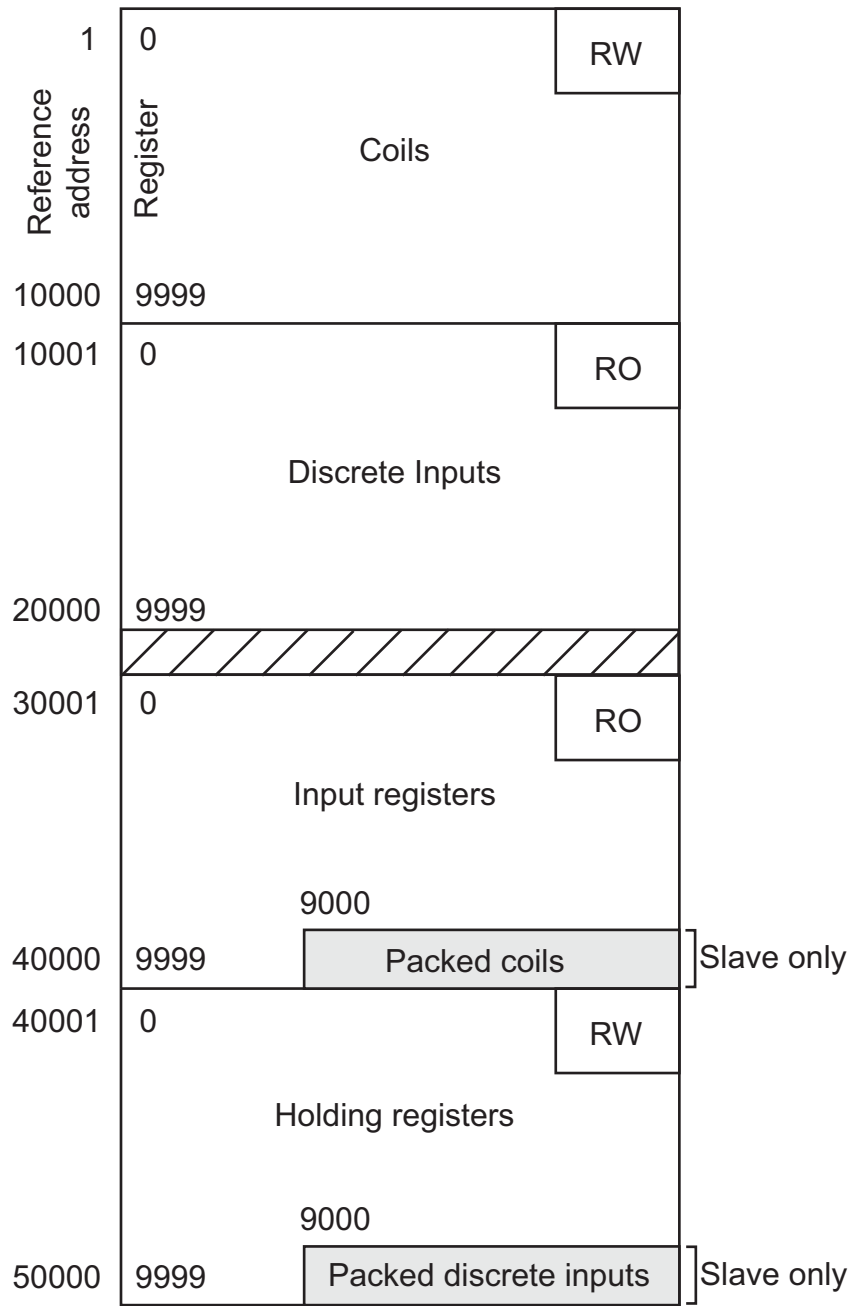


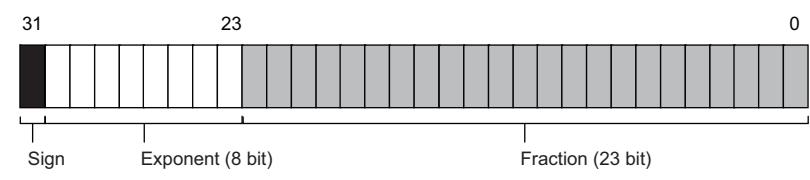
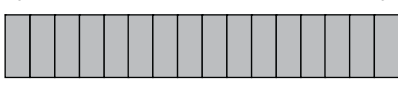
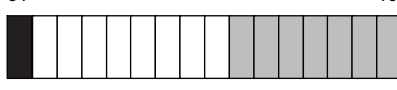
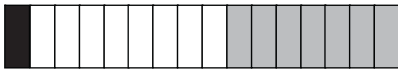

Fig. 10-1: Mapping of Modbus registers in ControlCare Field Controller

10.1.2 Data types

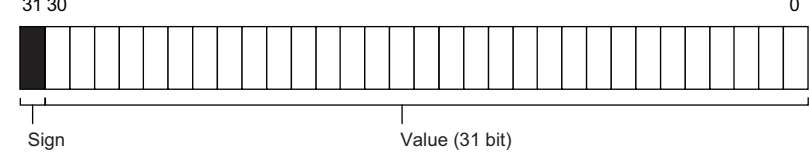
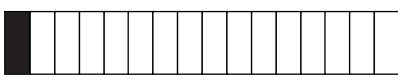
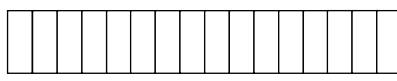
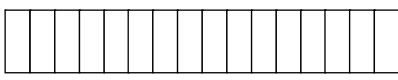
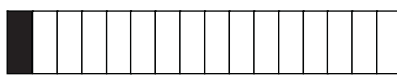
ControlCare Field Controllers support seven different data types, which are interpreted and stored as described below:

- Floating point
- Integer32, Unsigned Integer32, Integer 16, Unsigned Integer16, Integer8, Unsigned Integer8

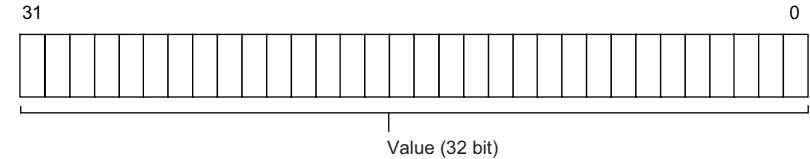
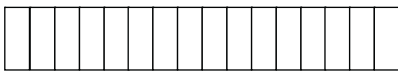
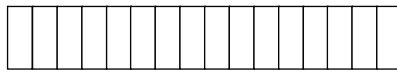
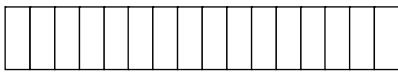
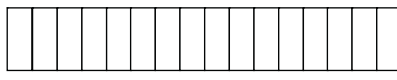
Floating point

Value range	$(\pm) 1.175 \cdot 10^{-38}$ to $3.403 \cdot 10^{38}$	
Byte Structure		
Storage	Float	Swapped Float
Modbus Register 1		
Modbus Register 2		

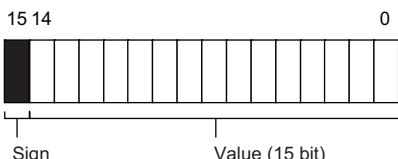


Integer32

Value range	-2147483648 to +2147483647	
Byte Structure		
Storage	Integer32	Swapped Integer32
Modbus Register 1		
Modbus Register 2		

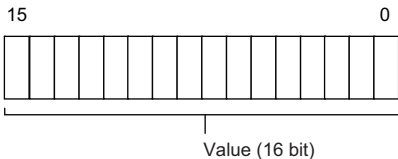


Unsigned32

Value range	0 to 4.294.967.295	
Byte Structure		
Storage	Unsigned Integer32	Swapped Unsigned Integer32
Modbus Register 1		
Modbus Register 2		

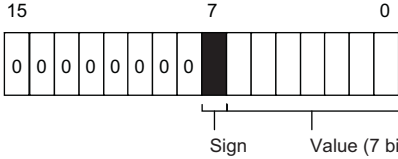


Integer16

Value range	-32.768 to 32.767	
Byte Structure		
Storage	Integer16	Swapped Integer16
Modbus Register 1		

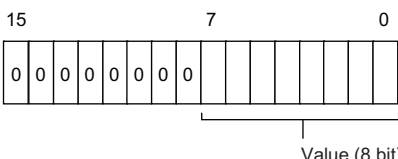
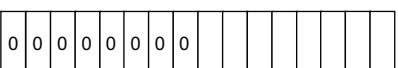
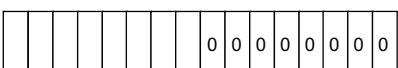
Unsigned16

Value range	0 to 65.535	
Byte Structure		
Storage	Unsigned Integer16	Swapped Unsigned Integer16
Modbus Register 1		

Integer8

Value range	-128 to +127	
Byte Structure		
Storage	Integer8	Swapped Integer8
Modbus Register 1		

Unsigned 8

Value range	0 to 255	
Byte Structure		
Storage	Unsigned Integer8	Swapped Unsigned Integer8
Modbus Register 1		

10.2 MBCF Modbus Configuration Block

The Modbus Configuration Block MBCF is used to configure the Field Controller for a role as master or slave. The table below lists the parameters and gives a short explanation of their function.

Block parameters

Parameter	Valid range	Default value	Description/Action
ST_VER		0	Indicates the revision level of the block's static parameters and may be used in configuration management
TAG_DESC		blanks	Allows the entry of a block description (up to 32 characters) which may be used in a human interface or in block documentation to clarify the block application
STRATEGY	0 to 255	0	Allows the entry of a user assigned value that may be used in configuration or diagnostics as a key in sorting block information
ALERT_KEY	1 to 255	1	Allows the entry of a user assigned value that may be used in sorting alarms or events generated by a block
MODE_BLK	TARGET	O/S	Block mode, set to Auto
BLOCK_ERR	0 to 15		Block errors, see Chapter 5.2.3
MEDIA	0 to 2	Serial	Defines Modbus medium, "()" slave in "master + slave" below 0: Serial (+TCP/IP), 1: TCP/IP (+Serial), 2: TCP/IP (+TCP/IP)
MASTER_SLAVE	0 to 2	Slave	Defines if Field Controller is master or slave ■ 0: Master, 1: Master + Slave, 2: Slave
TIMEOUT	0 - 65535	1000	Time allowed for a response from a slave (for Field Controller master) or for the OUTs be updated (for Field Controller slave). Value 0 is used to disable. – In the case of a slave, the TIMEOUT must be set to a value greater than the write cycle of the Modbus master, otherwise the status of the mapped values is BAD.
SERIAL_CONFIG	–	–	Configures serial interface
.BAUD_RATE	6 - 10	19200	Defines the baud rate ■ 6: 9600, 7: 19200, 8: 38400, 9: 57600, 10: 115200
.STOP_BITS	0, 1	1	Defines the number of stop bits (only for media serial). ■ 0: 1, 1: 2
.PARITY	1 - 3	Even	Defines the parity (only for media serial) ■ 0 :None, 1: Even, 2: Odd
TCP_IP_CONFIG	–	–	Configures TCP/IP interface
.SECOND_MOD_PORT			Second communication port (Port 502 is always open)
MASTER_CONFIG			To be configured when Field Controller is Modbus master
.NUMBER_RETRIES	0 - 255	1	Number of times Field Controller tries to retransmit the data if it does not receive a response from a slave
.MOD_SCAN_TIME			When online, displays Modbus scan time in ms when Field Controller is Modbus master (average of the last 10 cycles)
.MAX_DATA_LENGTH			Max. number of registers that can be read from a slave with a single telegram – Use only if slave does not support the standard length of 125 registers (250 byte)
SLAVE_CONFIG			To be configured when Field Controller is Modbus slave
.DEVICE_ADDRESS	1 - 247	1	Modbus slave address of Field Controller
.MAX_MOD_REACTION_TIME			When online, displays max. time to respond to a master
TCP_SLAVE_ADDRESSES			To be configured when Field Controller is Modbus TCP master
.IP_SLAVE_X			IP address of Modbus TCP slave
.DEVICE_ID_X			Unit address of Modbus TCP slave
.COMM_STATUS_X			When online, displays status of output data
TCP_ACCESS_LIST			To be configured when Field Controller is Modbus TCP slave
.IP_x			IP address of TCP master allowed to access to registers
ON_APPLY	0, 1	None	Applies the changes made in the Modbus blocks ■ 0: None, 1: Apply
UPDATE_EVT			This alert is generated by any change to the static data
MB_COMM_STATUS			Displays status of Modbus communication (OK when working)
USED_MOD_REGISTERS	RO		When online, displays number of Modbus points in use
FREE_MOD_REGISTERS	RO		When online, displays available Modbus points in percentage

10.2.1 Configuration as TCP/IP master

To configure the Field Controller as a Modbus TCP/IP master, set the following parameters in the **Off Line Characterization** dialog of the MBCF block:

Parameter	Function	MBCF
MODE_BLOCK.TARGET	Normal operating mode of block	Auto
MEDIA	Channel for Modbus communication	TCP/IP (+ TCP/IP)
MASTER_SLAVE	Role of Field Controller in Modbus network	Master
TIMEOUT	Time allowed for slave response If there is no response, the slave status is set to BAD	e.g. 1000 (ms)
TCP/IP_CONFIG SECOND_MOD_PORT	Configures TCP/IP interface Second communication port (Port 502 is always open)	e.g. 1024
MASTER_CONFIG NUMBER_OF_RETRIES MAX_DATA_LENGTH	Configures Controller when acting as master Number of retransmits if no response from a slave Max. number of registers that can be read from a slave with a single telegram (default = 250)	default 250
TCP_SLAVE_ADDRESSES IP_SLAVE_x DEVICE_ID_x	Configures up to eight Modbus TCP slaves IP address of Modbus TCP slave x Unit address of Modbus TCP slave x	e.g. 10.125.35.50 e.g. 12

When the block is on-line, selecting the "Apply" option in the **ON_APPLY** parameter starts Modbus communication. This is necessary after a download or a change in the Modbus configuration.

10.2.2 Configuration as TCP/IP slave

To configure the Field Controller as a Modbus TCP/IP slave, set the following parameters in the **Off Line Characterization** dialog of the MBCF block:

Parameter	Function	MBCF
MODE_BLOCK.TARGET	Normal operating mode of block	Auto
MEDIA	Channel for Modbus communication	TCP/IP (+ TCP/IP)
MASTER_SLAVE	Role of Field Controller in Modbus network	Slave
TIMEOUT	Time allowed for OUT value update If there is no update, the output status is set to BAD	e.g. 1000 (ms)
TCP/IP_CONFIG SECOND_MOD_PORT	Configures TCP/IP interface Second communication port (Port 502 is always open)	e.g. 1024
SLAVE_CONFIG DEVICE_ADDRESS	Configures Controller when acting as slave Modbus address of Field Controller	e.g. 1
TCP_ACCESS_LIST IP_x	List of up to eight masters that are allowed to access field controller registers IP address of TCP master x	e.g. 10.125.35.90

When the block is on-line, selecting the "Apply" option in the **ON_APPLY** parameter starts Modbus communication. This is necessary after a download or a change in the Modbus configuration.

10.2.3 Configuration as serial master

To configure the Field Controller as a Modbus serial master, set the following parameters in the **Off Line Characterization** dialog of the MBCF block:

Parameter	Function	MBCF
MODE_BLOCK.TARGET	Normal operating mode of block	Auto
MEDIA	Channel for Modbus communication	Serial (+TCP/IP)
MASTER_SLAVE	Role of Field Controller in Modbus network	Master
TIMEOUT	Time allowed for slave response If there is no response, the slave status is set to BAD	e.g. 1000 (ms)
SERIAL_CONFIG BAUDRATE STOP_BITS PARITY	Configures serial interface (default for Promass 83) Baudrate used for communication Number of stop bits used in telegram Parity used in telegram	e.g. 38400 e.g. 1 e.g. Even
MASTER_CONFIG NUMBER_OF_RETRIES MAX_DATA_LENGTH	Configures Controller when acting as master Number of retransmits if no response from a slave Max. number of registers that can be read from a slave with a single telegram (default = 250)	default 250

When the block is on-line, selecting the "Apply" option in the **ON_APPLY** parameter starts Modbus communication. This is necessary after a download or a change in the Modbus configuration.

10.2.4 Configuration as serial slave

To configure the Field Controller as a Modbus serial master, set the following parameters in the **Off Line Characterization** dialog of the MBCF block:

Parameter	Function	MBCF
MODE_BLOCK.TARGET	Normal operating mode of block	Auto
MEDIA	Channel for Modbus communication	TCP/IP (+Serial)
MASTER_SLAVE	Role of Field Controller in Modbus network	Slave
TIMEOUT	Time allowed for OUT value update If there is no update, the output status is set to BAD	e.g. 1000 (ms)
SERIAL_CONFIG BAUDRATE STOP_BITS PARITY	Configures serial interface (default for Promass 83) Baudrate used for communication Number of stop bits used in telegram Parity used in telegram	e.g. 38400 e.g. 1 e.g. Even
SLAVE_CONFIG DEVICE_ADDRESS	Configures Controller when acting as slave Modbus address of Field Controller	e.g. 1

When the block is on-line, selecting the "Apply" option in the **ON_APPLY** parameter starts Modbus communication. This is necessary after a download or a change in the Modbus configuration.

10.2.5 Configuration as master and slave

The Field Controller can also be configured to act simultaneously as both master and slave. The configuration of the media depends on the selection and is as described in the Chapter 10.2.1 to Chapter 10.2.4.

Parameter	Function	MBCF
MEDIA	Channel for Modbus communication <ul style="list-style-type: none"> ■ Serial (+TCP/IP): Serial master, TCP/IP slave ■ TCP/IP (+Serial): TCP/IP master, serial slave ■ TCP/IP (+ TCP/IP): TCP/IP master, TCP/IP slave 	e.g. TCP/IP (+Serial)
MASTER_SLAVE	Role of Field Controller in Modbus network	Master + slave

10.3 Modbus Control Master

The Modbus Control Master block is required when the Field Controller is to act as a Modbus master. In this role Field Controller reads from or writes to registers in a Modbus slave. The Modbus values are mapped through a FOUNDATION Fieldbus function block that has four sets of channels (AI, DI, AO, DO), allowing connection to other function blocks in a control strategy, see Fig. 10-2.

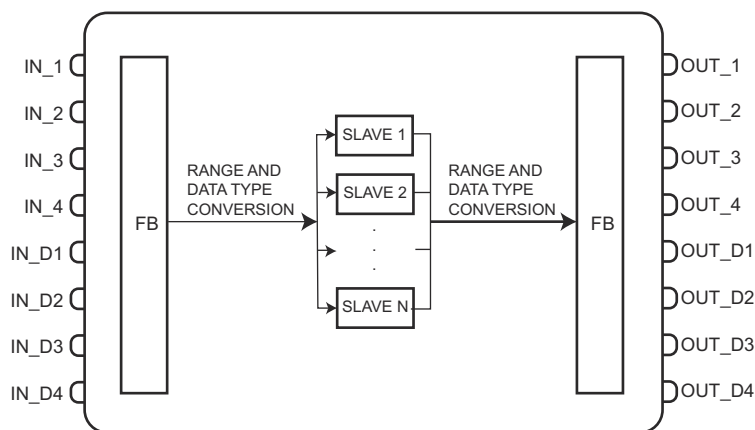


Fig. 10-2: Schematic diagram of the Modbus Control Master block

10.3.1 Block description

Up to 16 MBCM blocks can be created in a project. The blocks are managed via the parameter **LOCAL_MOD_MAP** which must have a unique value (0 to 15) for each block.

The mapping of the Modbus values is controlled by **SCALE_LOC_XXX** parameters for analog values and **LOCATOR_XXX** parameters for discrete values. These point the Field Controller to the register addresses to which data are to be written or from which data are to be read. The table gives an overview of the relationships of the input and output parameters to the mapping parameters:

Parameter	Type	Function in slave	Mapping parameter	Reference address
IN_1 to IN_4	Analog input	Write to holding register	SCALE_LOC_INx	40001 + register address
IN_D1 to IN_D4	Discrete input	Write to coil	LOCATOR_IN_Dx	1 + coil address
OUT_1 to OUT_4	Analog output	Read from input register	SCALE_LOC_OUTx	30001 + register address
		Read from holding register		40001 + register address
OUT_D1 to OUT_D4	Discrete output	Read from discrete input	LOCATOR_OUT_Dx	10001 + register address
		Read from coil		1 + coil address

Register addresses

Both **SCALE_LOC_XXX** and **LOCATOR_XXX** use three parameters to locate the register containing the Modbus value:

- **SLAVE_ADDRESS** points to the Modbus address of the connected slave
- **MODBUS_ADDRESS_OF_VALUE** points to the register address of the required value
- **MODBUS_ADDRESS_OF_STATUS** points to the register address of the associated status

In the case of analog data types that require two registers, only the first one is entered in **MODBUS_ADDRESS_OF_VALUE**. The second, adjacent address is automatically reserved according to the data type.

If the slave variable does not support a status or the status does not conform to the FF format, then zero must be entered in **MODBUS_ADDRESS_OF_STATUS**, see also Status handling. If required, an additional characterization can be made in the output status parameter **OUT_x.STATUS** or **OUT_Dx.STATUS**, e.g. "GoodCascade".

Scaling

SCALE_LOC_XXX contains additional parameters concerning data type and scaling of analog values:

- **DATA_TYPE** determines the data format of the value, see Chapter 10.1.2
- **FROM_EU_XX** determines the scaling of the "input" parameter
- **TO_EU_XX** determines the scaling of the "output" parameter

The scaling is performed as shown in Fig. 10-3, whereby the value to be scaled may lie outside the given limits.

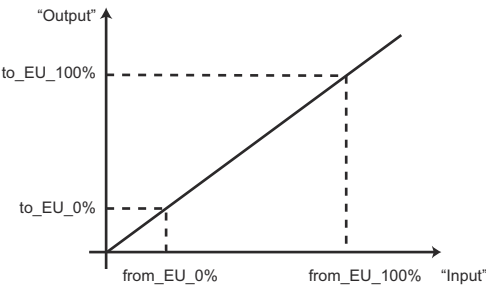


Fig. 10-3: Scaling of "input" to "output" units

The meaning of the scaling parameters in **SCALE_LOC_INx** and **SCALE_LOC_OUTx** is as follows:

Parameter	SCALE_LOC_INx	SCALE_LOC_OUTx
.FROM_EU_0	Lower range limit IN_x value	Lower range limit Modbus slave value
.FROM_EU_100	Upper range limit IN_x value	Upper range limit Modbus slave value
.TO_EU_0	Lower range limit Modbus slave value	Lower range limit OUT_x value
.TO_EU_100	Upper range limit Modbus slave value	Upper range limit OUT_x value

Status propagation

The status of a slave output value is mapped in the corresponding **OUT_XX.STATUS** parameter. The information it carries is dependent on the entry in **MODBUS_ADDRESS_OF_STATUS** and whether the period entered in the **TIMEOUT** parameter in the MBCF block has elapsed without a response being received from the slave.

MODBUS_ADDRESS_OF_STATUS	TIMEOUT	Status
Slave address register	Slave responding (communicated within timeout period)	As Slave
	Slave not responding (timeout elapsed)	Bad
0 (no status or status not FF conform)	Slave responding (communicated within timeout period)	Good
	Slave not responding (timeout elapsed)	Bad

The communication status of each Modbus variable can be read from the **COMM_STATUS** parameter. Each bit corresponds to one variable and corresponds to a logical OR between its status and value, whereby:

- If only the value is used, the status is considered zero
- If only the status is used, the value is considered zero.

If the bit is set (=1), there was an error during writing/reading of the respective parameter. The table shows the relationship between bit number and parameter.

BIT	PARAMETER	BIT	PARAMETER	BIT	PARAMETER	BIT	PARAMETER
0	IN_1	4	IN_D1	8	OUT_1	12	OUT_D1
1	IN_2	5	IN_D2	9	OUT_2	13	OUT_D2
2	IN_3	6	IN_D3	10	OUT_3	14	OUT_D3
3	IN_4	7	IN_D4	11	OUT_4	15	OUT_D4

10.3.2 Block configuration

The Modbus Control Master block is often used when measured values are to be acquired from and/or actuators are to be controlled in an existing Modbus network. In this example it is assumed that a serial Modbus Remote I/O is to be connected to the Field Controller:

- Slave address "5"
- Baud rate 19.2 kbit/s, Parity odd, Stopbits one
- Watchdog 2000 ms.

Fig. 10-4 shows the signals to be used by the application:

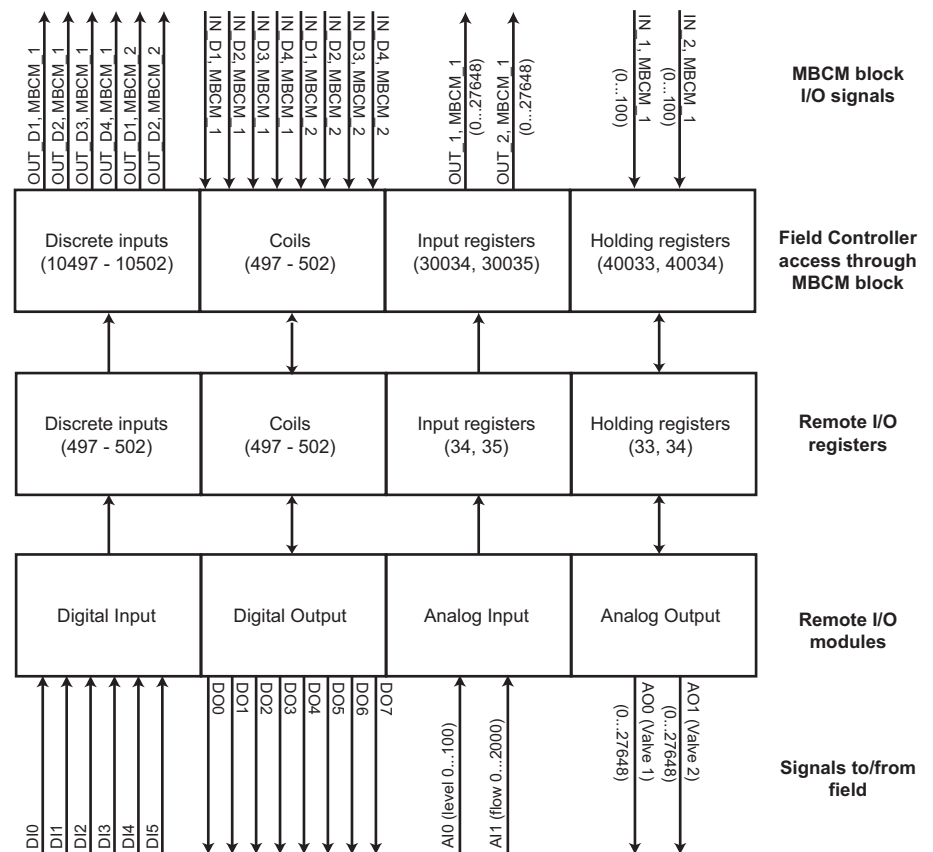


Fig. 10-4: Signal flow through MBCM block and Remote I/O

Six digital inputs and eight digital outputs are to be used together with two analog inputs (level and flow) and two analog outputs (valve positioners). The discrete inputs DI4 and DI5 are used directly to drive the outputs DO6 and DO7. The RIO registers start at one. The table shows the reference address range and location of the **MODBUS_ADDRESS_OF_VALUE** parameter. The status of the digital input will not be used by the Field Controller, so the **MODBUS_ADDRESS_OF_STATUS** must be set to "0" for all **OUT** parameters.

Input signals					
RIO Module	Signals used	Register	Coil	Action	LOCATOR_OUT_Dx/ SCALE_LOC_OUTx
Discrete Input	DI 0 ... DI 5	32	497 – 502	Read	10497 – 10502
	Status DI 0 ... DI5	33	513 – 518	–	–
Analog Input	AI0, AI1	34, 35	–	Read	30034, 30035
Output signals					
Module	Signals	Register	Coil	Action	LOCATOR_IN_Dx/ SCALE_LOC_INx
Discrete Output	DO0 ... DO7	32	497 – 504	Write	497 – 502
Analog Output	AO0, AO1	33, 34	–	Write	40033, 40034

The Analog Input has a level and flow meter connected to it, the Analog Output drives two valve actuators. The values offered by the RIO in the range 0...27648 must be scaled to or from the values used in the control strategy:

Input signal	ROI scale/FROM_EU	AI 0 scale/TO_EU	AI 1 scale/TO_EU
4 mA	0	0%	0
20 mA	27648	100%	2000 l/h
Output signal	ROI scale/TO_EU	AO 0 scale/FROM_EU	AO 1 scale/FROM_EU
4 mA	0	0%	0%
20 mA	27648	100%	100%

To accommodate the six discrete inputs and eight discrete outputs, two MBCM blocks are required. Fig. 10-5 shows a control strategy for the above application, whereby the upstream and downstream blocks, have been left out for clarity. The MBCM block can be connected to any function block that is available in the network, e.g. a PID block, and vice versa.

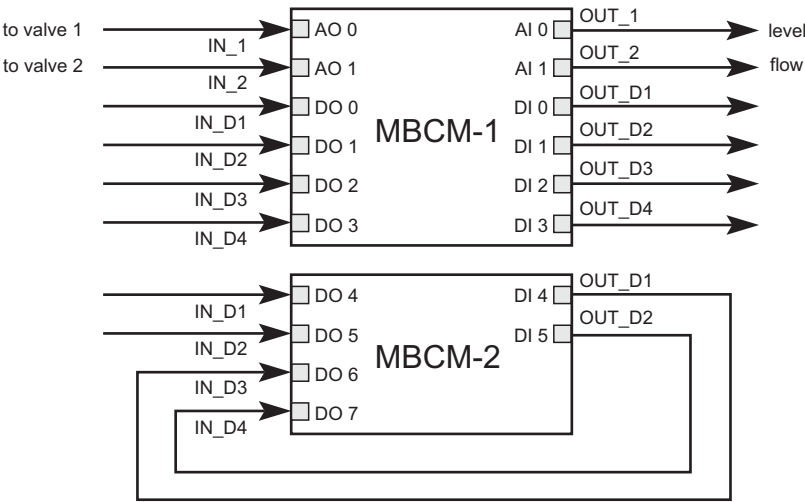


Fig. 10-5: Control Strategy for integration of Modbus variables

MBCF block configuration

The MBCF and MBCM blocks must be configured as follows:

Parameter	Function	Example value
MODE_BLOCK.TARGET	Normal operating mode of block	Auto
MEDIA	Channel for Modbus communication	Serial (+TCP/IP)
MASTER_SLAVE	Role of Field Controller in Modbus network	Master
TIMEOUT	Time allowed for slave response (= Watchdog) If there is no response, the slave status is set to BAD	2000 (ms)
SERIAL_CONFIG	Configures serial interface (default for Promass 83)	
BAUDRATE	Baudrate used for communication	19200
STOP_BITS	Number of stop bits used in telegram	1
PARITY	Parity used in telegram	Odd
MASTER_CONFIG	Configures Controller when acting as master	
NUMBER_OF_RETRIES	Number of retransmits if no response from a slave	3
MAX_DATA_LENGTH	Max. number of registers that can be read from a slave with a single telegram	250 (default)

MBCS block configuration The MBCM blocks must be configured as follows:

Parameter	Function	Example value
MBCM-1 block		
MODE_BLK.Target	Normal operating mode of block	Auto
LOCAL_MOD_MAP	Identifier of Modbus block (0 - 15)	0
SCALE_LOC_IN1 FROM_EU_0 FROM_EU_100 TO_EU_0 TO_EU_100 DATA_TYPE SLAVE_ADDRESS MODBUS_ADDRESS_OF_VALUE MODBUS_ADDRESS_OF_STATUS	Scaling and conversion of IN_1 (valve 1) Lower range limit of IN_1 Upper range limit of IN_1 Lower range limit of ROI signal Upper range limit of ROI signal Type of data transmitted Modbus address of variable source Holding register to which value is to be written Register to which status is to be written	0 100 0 27468 float 5 40034 0
SCALE_LOC_IN2 FROM_EU_0 FROM_EU_100 TO_EU_0 TO_EU_100 DATA_TYPE SLAVE_ADDRESS MODBUS_ADDRESS_OF_VALUE MODBUS_ADDRESS_OF_STATUS	Scaling and conversion of IN_2 (valve 2) Lower range limit of IN_2 Upper range limit of IN_2 Lower range limit of ROI signal Upper range limit of ROI signal Type of data transmitted Modbus address of variable source Holding register to which value is to be written Register to which status is to be written	0 100 0 27468 float 5 40035 0
LOCATOR_IN_D1 SLAVE_ADDRESS MODBUS_ADDRESS_OF_VALUE MODBUS_ADDRESS_OF_STATUS	Location of IN_D1 Modbus address of variable source Coil to which value is to be written Register/coil to which status is to be written	5 497 0
LOCATOR_IN_D2 SLAVE_ADDRESS MODBUS_ADDRESS_OF_VALUE MODBUS_ADDRESS_OF_STATUS	Location of IN_D2 Modbus address of variable source Coil to which value is to be written Register/coil to which status is to be written	5 498 0
LOCATOR_IN_D3 SLAVE_ADDRESS MODBUS_ADDRESS_OF_VALUE MODBUS_ADDRESS_OF_STATUS	Location of IN_D3 Modbus address of variable source Coil to which value is to be written Register/coil to which status is to be written	5 499 0
LOCATOR_IN_D4 SLAVE_ADDRESS MODBUS_ADDRESS_OF_VALUE MODBUS_ADDRESS_OF_STATUS	Location of IN_D4 Modbus address of variable source Coil to which value is to be written Register/coil to which status is to be written	5 500 0
SCALE_LOC_OUT1 FROM_EU_0 FROM_EU_100 TO_EU_0 TO_EU_100 DATA_TYPE SLAVE_ADDRESS MODBUS_ADDRESS_OF_VALUE MODBUS_ADDRESS_OF_STATUS	Scaling and conversion of OUT_1 (level) Lower range limit of ROI signal Upper range limit of ROI signal Lower range limit of OUT_1 Upper range limit of OUT_1 Type of data transmitted Modbus address of variable source Input register containing value Register containing status	0 27468 0 100 float 5 30034 0
SCALE_LOC_OUT2 FROM_EU_0 FROM_EU_100 TO_EU_0 TO_EU_100 DATA_TYPE SLAVE_ADDRESS MODBUS_ADDRESS_OF_VALUE MODBUS_ADDRESS_OF_STATUS	Scaling and conversion of OUT_2 (flow) Lower range limit of ROI signal Upper range limit of ROI signal Lower range limit of OUT_2 Upper range limit of OUT_2 Type of data transmitted Modbus address of variable source Input register containing value Register containing status	0 27468 0 2000 float 5 30035 0
LOCATOR_OUT_D1 SLAVE_ADDRESS MODBUS_ADDRESS_OF_VALUE MODBUS_ADDRESS_OF_STATUS	Location of OUT_D1 Modbus address of variable source Discrete input containing value Register containing status	5 10497 0
LOCATOR_OUT_D2 SLAVE_ADDRESS MODBUS_ADDRESS_OF_VALUE MODBUS_ADDRESS_OF_STATUS	Location of OUT_D2 Modbus address of variable source Discrete input containing value Register containing status	5 10498 0

Parameter	Function	Example value
LOCATOR_OUT_D3 SLAVE_ADDRESS MODBUS_ADDRESS_OF_VALUE MODBUS_ADDRESS_OF_STATUS	Location of OUT_D3 Modbus address of variable source Discrete input containing value Register containing status	5 10499 0
LOCATOR_OUT_D4 SLAVE_ADDRESS MODBUS_ADDRESS_OF_VALUE MODBUS_ADDRESS_OF_STATUS	Location of OUT_D4 Modbus address of variable source Discrete input containing value Register containing status	5 10500 0
MBCM-2 block		
MODE_BLK.Target	Normal operating mode of block	Auto
LOCAL_MOD_MAP	Identifier of modbus block (0 - 15)	1
LOCATOR_IN_D1 SLAVE_ADDRESS MODBUS_ADDRESS_OF_VALUE MODBUS_ADDRESS_OF_STATUS	Location of IN_D1 Modbus address of variable source Coil to which value is to be written Holding register/coil to which status is to be written	5 501 0
LOCATOR_IN_D2 SLAVE_ADDRESS MODBUS_ADDRESS_OF_VALUE MODBUS_ADDRESS_OF_STATUS	Location of IN_D2 Modbus address of variable source Coil to which value is to be written Holding register/coil to which status is to be written	5 502 0
LOCATOR_IN_D3 SLAVE_ADDRESS MODBUS_ADDRESS_OF_VALUE MODBUS_ADDRESS_OF_STATUS	Location of IN_D3 Modbus address of variable source Coil to which value is to be written Holding register/coil to which status is to be written	5 503 0
LOCATOR_IN_D4 SLAVE_ADDRESS MODBUS_ADDRESS_OF_VALUE MODBUS_ADDRESS_OF_STATUS	Location of IN_D4 Modbus address of variable source Coil to which value is to be written Holding register/coil to which status is to be written	5 504 0
LOCATOR_OUT_D1 SLAVE_ADDRESS MODBUS_ADDRESS_OF_VALUE MODBUS_ADDRESS_OF_STATUS	Location of OUT_D1 Modbus address of variable source Discrete input containing value Register containing status	5 10501 0
LOCATOR_OUT_D2 SLAVE_ADDRESS MODBUS_ADDRESS_OF_VALUE MODBUS_ADDRESS_OF_STATUS	Location of OUT_D2 Modbus address of variable source Discrete input containing value Register containing status	5 10502 0

10.3.3 Block operation

The Modbus Control Master block normally operates in Auto mode. It can also be put out of service (OOS) to change parameters.

After download of the strategy and after the change of any parameter, Modbus communication must be configured and initialized by toggling the **APPLY_ON** parameter in the MBCF block, i.e set to "Apply".

Status

Status	Meaning
GoodNC:xxx	■ Block operating correctly, OUT value valid
Bad:NoCommunication	■ No Modbus communication
Bad:Out of Service	■ Block currently out of service

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Other	■ Value out of range for data type selected – Check correct data type – Check correct engineering units scaling
Out-of-Service	■ MODE_BLK.Target currently set to OOS

10.3.4 Block parameters

Parameter	Valid range/ Options	Default value	Description/Action
ST_VER		0	See Chapter 2.3.3
TAG_DESC		blanks	
STRATEGY	0 to 255	0	
ALERT_KEY	1 to 255	1	
MODE_BLK	TARGET	O/S	Block mode, set to Auto
BLOCK_ERR	0 to 15		Block errors, see Chapter 2.9.4
LOCAL_MOD_MAP	0 to 15	0	Unique identifier for MBCM block
COMM_STATUS		0	Indicates if communication from slave is good or not (each bit corresponds to a Modbus variable)
IN_1			Value and status of analog input 1
SCALE_LOC_IN1			Scaling, data format and register addresses for input signal 1
IN_2			Value and status of analog input 2
SCALE_LOC_IN2			Scaling, data format and register addresses for input signal 2
IN_3			Value and status of analog input 3
SCALE_LOC_IN3			Scaling, data format and register addresses for input signal 3
IN_4			Value and status of analog input 4
SCALE_LOC_IN4			Scaling, data format and register addresses for input signal 4
IN_D1			Value and status of discrete input 1
LOCATOR_IN_D1			Register addresses for discrete input signal 1
IN_D2			Value and status of discrete input 2
LOCATOR_IN_D2			Register addresses for discrete input signal 2
IN_D3			Value and status of discrete input 3
LOCATOR_IN_D3			Register addresses for discrete input signal 3
IN_D4			Value and status of discrete input 4
LOCATOR_IN_D4			Register addresses for discrete input signal 4
OUT_1			Value and status of analog output 1
SCALE_LOC_OUT1			Scaling, data format and register addresses for output signal 1
OUT_2			Value and status of analog output 2
SCALE_LOC_OUT2			Scaling, data format and register addresses for output signal 2
OUT_3			Value and status of analog output 3
SCALE_LOC_OUT3			Scaling, data format and register addresses for output signal 3
OUT_4			Value and status of analog output 4
SCALE_LOC_OUT4			Scaling, data format and register addresses for output signal 4
OUT_D1			Value and status of discrete output 1
LOCATOR_OUT_D1			Register addresses for discrete output signal 1
OUT_D2			Value and status of discrete output 2
LOCATOR_OUT_D2			Register addresses for discrete output signal 2
OUT_D3			Value and status of discrete output 3
LOCATOR_OUT_D3			Register addresses for discrete output signal 3
OUT_D4			Value and status of discrete output 4
LOCATOR_OUT_D4			Register addresses for discrete output signal 4
UPDATE_EVT			This alert is generated by any change to the static data
BLOCK_ALM			Block alarms

10.4 Modbus Control Slave

The Modbus Control Slave block is required when the Field Controller is to act as a Modbus slave. In this role the Modbus Master reads from or writes to registers in the Field Controller. The Modbus values are mapped through a FOUNDATION Fieldbus function block that has four sets of channels (AI, DI, AO, DO), allowing connection to other function blocks in a control strategy, see Fig. 10-6.

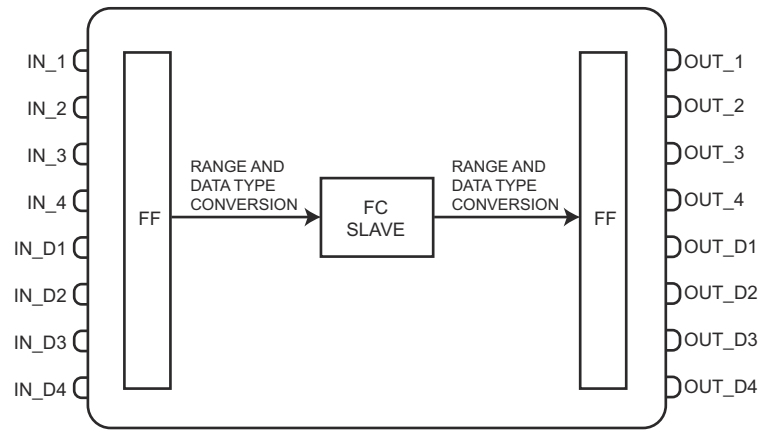


Fig. 10-6 Schematic diagram of the Modbus Control Slave block

10.4.1 Block description

Up to 16 MBCS blocks can be created in a project. The blocks are managed via the parameter **LOCAL_MOD_MAP** which must have a unique value (0 to 15) for each block.

Register addresses

The value of **LOCAL_MOD_MAP** is used to define a unique set of Modbus register address ranges for the particular MBCS block, whereby:

- Register address = Constant + 8 x Value of LOCAL_MOD_MAP for analog values
- Register address = Constant + 4 x Value of LOCAL_MOD_MAP for discrete values/status.

The Modbus master accesses the Field Controller slave registers by specifying the function code and the register address. The registers are assigned to the analog/discrete input/outputs as follows:

IN_x

Parameter	Type	Register Address	Access by Function Code
IN_1.Value	Input register	0 + (8 x LOCAL_MOD_MAP)	0x04 (Read input register)
IN_2.Value		2 + (8 x LOCAL_MOD_MAP)	
IN_3.Value		4 + (8 x LOCAL_MOD_MAP)	
IN_4.Value		6 + (8 x LOCAL_MOD_MAP)	
IN_1.Status		128 + (4 x LOCAL_MOD_MAP)	
IN_2.Status		129 + (4 x LOCAL_MOD_MAP)	
IN_3.Status		130 + (4 x LOCAL_MOD_MAP)	
IN_4.Status		131 + (4 x LOCAL_MOD_MAP)	

IN_Dx

Parameter	Type	Register Address	Access by Function Code
IN_D1.Value	Discrete Input	0 + (4 x LOCAL_MOD_MAP)	0x02 (Read discrete input)
IN_D2.Value		1 + (4 x LOCAL_MOD_MAP)	
IN_D3.Value		2 + (4 x LOCAL_MOD_MAP)	
IN_D4.Value		3 + (4 x LOCAL_MOD_MAP)	
IN_D1.Status	Input Register	192 + (4 x LOCAL_MOD_MAP)	0x04 (Read input register)
IN_D2.Status		193 + (4 x LOCAL_MOD_MAP)	
IN_D3.Status		194 + (4 x LOCAL_MOD_MAP)	
IN_D4.Status		195 + (4 x LOCAL_MOD_MAP)	

OUT_x

Parameter	Type	Register Address	Access by Function Code
OUT_1.Value	Holding register	0 + (8 x LOCAL_MOD_MAP)	0x03 (Read holding registers)
OUT_2.Value		2 + (8 x LOCAL_MOD_MAP)	0x06 (Write single register)
OUT_3.Value		4 + (8 x LOCAL_MOD_MAP)	0x10 (Write multiple registers)
OUT_4.Value		6 + (8 x LOCAL_MOD_MAP)	0x17 (Read/Write multiple registers)
OUT_1.Status		128 + (4 x LOCAL_MOD_MAP)	
OUT_2.Status		129 + (4 x LOCAL_MOD_MAP)	
OUT_3.Status		130 + (4 x LOCAL_MOD_MAP)	
OUT_4.Status		131 + (4 x LOCAL_MOD_MAP)	

OUT_Dx

Parameter	Type	Register Address	Access by Function Code
OUT_D1.Value	Coil	0 + (4 x LOCAL_MOD_MAP)	0x01 (Read coils)
OUT_D2.Value		1 + (4 x LOCAL_MOD_MAP)	0x05 (Write single coil)
OUT_D3.Value		2 + (4 x LOCAL_MOD_MAP)	0x0F (Write multiple coils)
OUT_D4.Value		3 + (4 x LOCAL_MOD_MAP)	
OUT_D1.Status	Holding register	192 + (4 x LOCAL_MOD_MAP)	0x03 (Read holding registers)
OUT_D2.Status		193 + (4 x LOCAL_MOD_MAP)	0x06 (Write single register)
OUT_D3.Status		194 + (4 x LOCAL_MOD_MAP)	0x10 (Write multiple registers)
OUT_D4.Status		195 + (4 x LOCAL_MOD_MAP)	0x17 (Read/Write multiple registers)

The Field Controller also offers Discrete Inputs and Coils as packed words. The associated status is not available as packed words and must be acquired individually from the input or holding register.

Packed words

Type	Parameters	Register Address	Access by Function Code
Input register (Discrete inputs)	DI0 (Bit0) - DI15 (Bit15)	9000	0x04 (Read input register)
	DI16 - DI31	9001	
	...	9002 - 9624	
	DI9984 - DI9999	9625	
Holding register (Coils)	DO0 (Bit0) - DO15 (Bit15)	9000	0x03 (Read holding registers) 0x06 (Write single register) 0x10 (Write multiple registers) 0x17 (Read/Write multiple registers)
	DO16 - DO31	9001	
	...	9002 - 9624	
	DO9984 - DO9999	9625	

Configuration

The **SCALE_CONV_XXX** and **STATUS_OUT_D** parameters allow the configuration of the analog I/O and discrete output channels respectively.

- The Modbus register is assigned automatically according to the tables above.
- **STATUS_OUT_Dx** and **STATUS_OUTPUT** determine how the status of the **OUT_Dx** and **OUT_x** parameters is generated

The status can be generated in one of two ways:

- If the option "Set by master" in the picklist is used, the output status written by the Modbus master in the associated input (IN_Dx, IN_x) or holding register (OUT_Dx, OUT_x) is used.
- If any other option is set, the output status will be set automatically to the status selected, e.g Good_NonCascade: NonSpecific: NotLimited.

In both cases, a communication failure will force the status to Bad: NoCommunication, see Status handling

Scaling

SCALE_CONV_XXX contains additional parameters concerning data type and scaling of analog values:

- **DATA_TYPE** determines the data format of the value, see Chapter 10.1.2
- **FROM_EU_XX** determines the scaling of the "input" parameter
- **TO_EU_XX** determines the scaling of the "output" parameter

The scaling is performed as shown in Fig. 10-7, whereby the value to be scaled may lie outside the given limits.

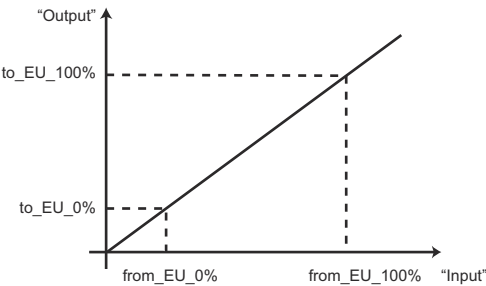


Fig. 10-7: Scaling of raw value to OUT_x value engineering units

The meaning of the scaling parameters in **SCALE_CONV_INx** and **SCALE_CONV_OUTx** is as follows:

Parameter	SCALE_CONV_INx	SCALE_CONV_OUTx
.FROM_EU_0	Lower range limit IN_x value	Lower range limit Modbus master value
.FROM_EU_100	Upper range limit IN_x value	Upper range limit Modbus master value
.TO_EU_0	Lower range limit Modbus master value	Lower range limit OUT_x value
.TO_EU_100	Upper range limit Modbus master value	Upper range limit OUT_x value

Status propagation

The status of a slave output value is mapped in the corresponding **OUT_XX.STATUS** parameter. The information it carries is dependent on the entry in **STATUS_OUT_Dx** or **STATUS_OUTPUT** and whether the period entered in the **TIMEOUT** parameter in the MBCF block has elapsed.

STATUS_OUT_Dx or STATUS_OUTPUT	TIMEOUT	Status
Set by master	Slave responding (communicated within timeout period)	As master
	Slave not responding (timeout elapsed)	Bad
Status from pick list	Slave responding (communicated within timeout period)	As set
	Slave not responding (timeout elapsed)	Bad

10.4.2 Block configuration

The Modbus Control Slave block is often used when measured values are to be acquired from and/or actuators are to be controlled in an FOUNDATION Fieldbus or PROFIBUS network by an existing Modbus controller. In this case, the FieldController acts as a Remote I/O, presenting and receiving the measured and manipulated variables.

As an example it is assumed a new section of plant has built comprising two tanks, each fitted with two level limit switches, a level meter, a flow meter, a temperature meter, an on/off valve and a control valve, see Fig. 10-8.

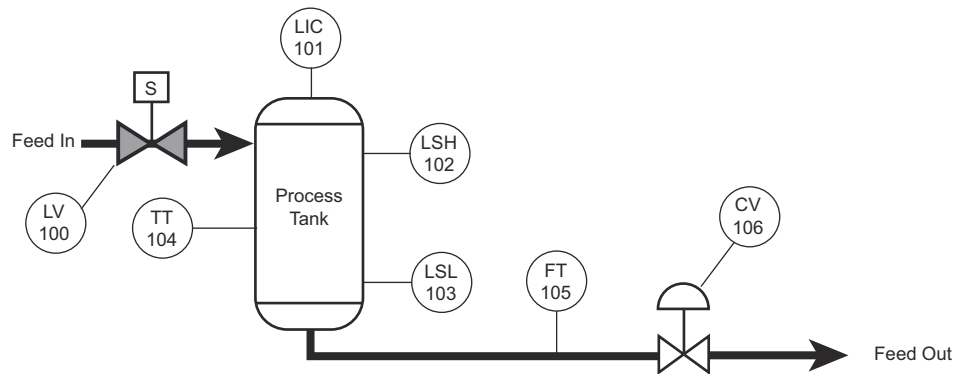


Fig. 10-8: Example for use of Modbus Control Slave (no control-loops shown)

Communication between the Field Controller and Modbus Controller is via Ethernet and the designated Modbus slave address 16. A timeout of 2000 ms is to be used.

An example for the strategy is shown in Fig. 10-9. As two blocks have to be created to accommodate the six analog inputs, all I/O relating to Tank 1 are in block MBCS-1 and all relating to Tank 2 in MBCS-2.

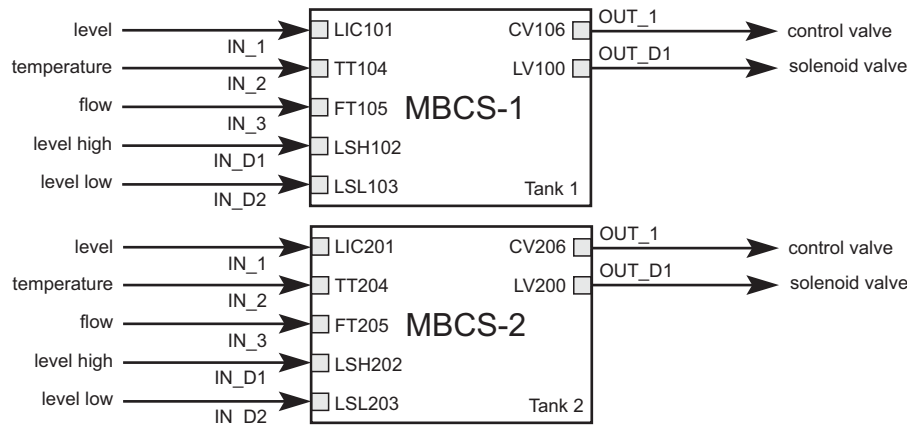


Fig. 10-9: Control strategy for the Modbus Control Slave (no control-loops shown)

The required scaling is as follows:

Input signal	FROM_EU_0 (IN)	FROM_EU_100 (IN)	TO_EU_0 (ROI)	TO_EU_100 (ROI)
Level	0%	100%	0	27648
Temperature	-150°C	+600°C	0	27648
Flow	0 l/h	2000 l/h	0	27648
Output signal	FROM_EU_0 (ROI)	FROM_EU_100 (ROI)	TO_EU_0 (OUT)	TO_EU_100 (OUT)
Actuator	0	27648	0%	100%

Fig. 10-10 shows the associated signal flow:

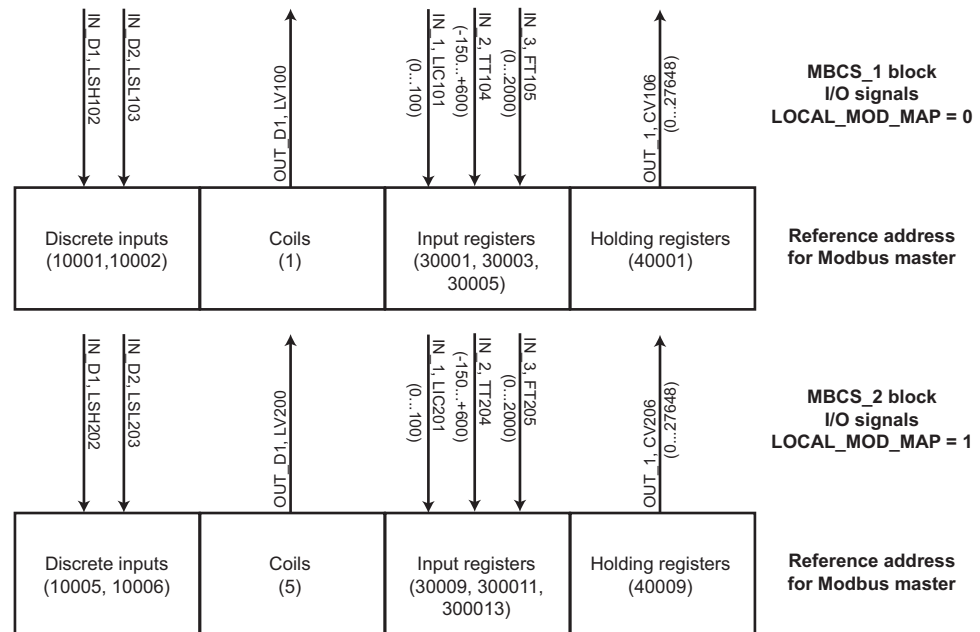


Fig. 10-10: Signal flow (values) through MBCS block to Modbus master

MBCF block configuration

Parameter	Function	MBCF
MODE_BLOCK.TARGET	Normal operating mode of block	Auto
MEDIA	Channel for Modbus communication	TCP/IP (+TCP/IP)
MASTER_SLAVE	Role of Field Controller in Modbus network	Slave
TIMEOUT	Time allowed for OUT value update If there is no update, the output status is set to BAD	e.g. 1000 (ms)
TCP/IP_CONFIG SECOND_MOD_PORT	Configures TCP/IP interface Second communication port (Port 502 is always open)	e.g. 1024
SLAVE_CONFIG DEVICE_ADDRESS	Configures Controller when acting as slave Modbus address of Field Controller	e.g. 1
TCP_ACCESS_LIST IP_x	List of up to eight masters that are allowed to access field controller registers IP address of TCP master x	e.g. 10.125.35.90

MBCS block configuration

Parameter	Function	Example value
MBCS-1 block		
MODE_BLK.Target	Normal operating mode of block	Auto
LOCAL_MOD_MAP	Identifier of modbus block (0 - 15)	0
SCALE_CONV_IN1 FROM_EU_0 FROM_EU_100 TO_EU_0 TO_EU_100 DATA_TYPE	Scaling and conversion of IN_1 (level) Lower range limit of IN_1 Upper range limit of IN_1 Lower range limit of ROI signal Upper range limit of ROI signal Type of data transmitted	0 100 0 27468 float
SCALE_CONV_IN2 FROM_EU_0 FROM_EU_100 TO_EU_0 TO_EU_100 DATA_TYPE	Scaling and conversion of IN_2 (temperature) Lower range limit of IN_2 Upper range limit of IN_2 Lower range limit of ROI signal Upper range limit of ROI signal Type of data transmitted	-150 600 0 27468 float
SCALE_CONV_IN3 FROM_EU_0 FROM_EU_100 TO_EU_0 TO_EU_100 DATA_TYPE	Scaling and conversion of IN_3 (flow) Lower range limit of IN_3 Upper range limit of IN_3 Lower range limit of ROI signal Upper range limit of ROI signal Type of data transmitted	0 2000 0 27468 float

Parameter	Function	Example value
SCALE_CONV_OUT1	Scaling and conversion of OUT_1 (valve)	
FROM_EU_0	Lower range limit of ROI signal	0
FROM_EU_100	Upper range limit of ROI signal	27468
TO_EU_0	Lower range limit of OUT_1	0
TO_EU_100	Upper range limit of OUT_1	100
DATA_TYPE	Type of data transmitted	float
STATUS_OUTPUT	Status to be used by good output	GoodNC:NSpecNLim
STATUS_OUT_D1	Status to be used by good output	GoodNC:NSpecNLim
MBCS-2 block		
MODE_BLK.Target	Normal operating mode of block	Auto
LOCAL_MOD_MAP	Identifier of modbus block (0 - 15)	1
SCALE_CONV_IN1	Scaling and conversion of IN_1 (level)	
FROM_EU_0	Lower range limit of IN_1	0
FROM_EU_100	Upper range limit of IN_1	100
TO_EU_0	Lower range limit of ROI signal	0
TO_EU_100	Upper range limit of ROI signal	27468
DATA_TYPE	Type of data transmitted	float
SCALE_CONV_IN2	Scaling and conversion of IN_2 (temperature)	
FROM_EU_0	Lower range limit of IN_2	-150
FROM_EU_100	Upper range limit of IN_2	600
TO_EU_0	Lower range limit of ROI signal	0
TO_EU_100	Upper range limit of ROI signal	27468
DATA_TYPE	Type of data transmitted	float
SCALE_CONV_IN3	Scaling and conversion of IN_3 (flow)	
FROM_EU_0	Lower range limit of IN_3	0
FROM_EU_100	Upper range limit of IN_3	2000
TO_EU_0	Lower range limit of ROI signal	0
TO_EU_100	Upper range limit of ROI signal	27468
DATA_TYPE	Type of data transmitted	float
SCALE_CONV_OUT1	Scaling and conversion of OUT_1 (valve)	
FROM_EU_0	Lower range limit of ROI signal	0
FROM_EU_100	Upper range limit of ROI signal	27468
TO_EU_0	Lower range limit of OUT_1	0
TO_EU_100	Upper range limit of OUT_1	100
DATA_TYPE	Type of data transmitted	float
STATUS_OUTPUT	Status to be used by good output	GoodNC:NSpecNLim
STATUS_OUT_D1	Status to be used by good output	GoodNC:NSpecNLim

After configuration the data are made available by or to the Modbus master at the following reference addresses:

Tag	Master Action	Value	Status	Tag	Master Action	Value	Status
LV100	Write	1	40192	LV200	Write	5	40196
LIC101	Read	30001, 30002	30128	LIC201	Read	30009, 30010	30132
LSH102	Read	10001	30192	LSH202	Read	10005	30196
LSL103	Read	10002	30193	LSL203	Read	10006	30197
TT104	Read	30003, 30004	30129	TT204	Read	30011, 30012	30133
FT105	Read	30005, 30006	30130	FT205	Read	30013, 30014	30134
CV106	Write	40001	40128	CV206	Write	40009	40132

10.4.3 Block operation

The Modbus Control Slave block normally operates in Auto mode. It can also be put out of service (OOS) to change parameters.

After download of the strategy and after the change of any parameter, Modbus communication must be initialized by toggling the **APPLY_ON** parameter in the MBCF block, i.e set to "1".

Status

Status	Meaning
GoodNC:xxx	■ Block operating correctly, OUT value valid
Bad:NoCommunication	■ No Modbus communication
Bad:Out of Service	■ Block currently out of service

Block errors

Block errors are indicated in the **BLOCK_ERR** parameter

Condition	Reason
Other	<ul style="list-style-type: none"> Value out of range for data type selected <ul style="list-style-type: none"> Check correct data type Check correct engineering units scaling
Out-of-Service	<ul style="list-style-type: none"> MODE_BLK.Target currently set to OOS

10.4.4 Block parameters

Parameter	Valid range/ Options	Default value	Description/Action
ST_VER		0	See Chapter 2.3.3.
TAG_DESC		blanks	
STRATEGY	0 to 255	0	
ALERT_KEY	1 to 255	1	
MODE_BLK	TARGET	O/S	Block mode, set to Auto
BLOCK_ERR	0 to 15		Block errors, see Chapter 2.9.4
LOCAL_MOD_MAP	0 to 15	0	Unique identifier for MBCS block
IN_1			Value and status of analog input 1
SCALE_CONV_IN1			Scaling and data format for analog input signal 1
IN_2			Value and status of analog input 2
SCALE_CONV_IN2			Scaling and data format for analog input signal 2
IN_3			Value and status of analog input 3
SCALE_CONV_IN3			Scaling and data format for analog input signal 3
IN_4			Value and status of analog input 4
SCALE_CONV_IN4			Scaling and data format for analog input signal 4
IN_D1			Value and status of discrete input 1
IN_D2			Value and status of discrete input 2
IN_D3			Value and status of discrete input 3
IN_D4			Value and status of discrete input 4
OUT_1			Value and status of analog output 1
SCALE_CONV_OUT1			Scaling, data format and status for analog output signal 1
OUT_2			Value and status of analog output 2
SCALE_CONV_OUT2			Scaling, data format and status for analog output signal 2
OUT_3			Value and status of analog output 3
SCALE_CONV_OUT3			Scaling, data format and status for analog output signal 3
OUT_4			Value and status of analog output 4
SCALE_CONV_OUT4			Scaling, data format and status for analog output signal 4
OUT_D1			Value and status of discrete output 1
STATUS_OUT_D1			Status for discrete output signal 1
OUT_D2			Value and status of discrete output 2
STATUS_OUT_D2			Status for discrete output signal 2
OUT_D3			Value and status of discrete output 3
STATUS_OUT_D3			Status for discrete output signal 3
OUT_D4			Value and status of discrete output 4
STATUS_OUT_D4			Status for discrete output signal 4
UPDATE_EVT			This alert is generated by any change to the static data
BLOCK_ALM			Block alarms

Appendix A: Interpreting parameter status

The Status has the following composition:

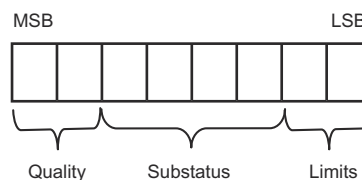


Fig. 10-11: Composition of status byte

The quality, sub-status, and limit components of status are defined as follows:

Quality - The quality used is determined by the highest priority condition:

- 0 = Bad
- 1 = Uncertain
- 2 = Good (Non-cascade)
- 3 = Good (Cascade)

Substatus - Sub-status values in the status attribute are defined as shown in the table below.

Limit - The following limit conditions will be always available in the status attribute.

- 0 = Not limited
- 1 = Low limited
- 2 = High limited
- 3 = Constant

Examples:

- 0xC1 (in hexadecimal) is "Good-Cascade Non Specific and Low Limited " status
- 0xCF(in hexadecimal) is "Good-Cascade Not invited and Constant " status
- 0x4E(in hexadecimal) is "Uncertain Initial Value and High Limited" status

Substatus values

Substatus values as a function of quality

Quality	Substatus	Hex value	Cascade path		
			Not in	Forward	Backward
Bad	0 = Non-specific	0x00	X	X	X
Bad	1 = Configuration Error	0x04	X	X	X
Bad	2 = Not Connected	0x08			
Bad	3 = Device Failure	0x0c	X	X	X
Bad	4 = Sensor Failure	0x10	X	X	X
Bad	5 = No Communication	last usable value	0x14		
Bad	6 = No Communication	no usable value	0x18		
Bad	7 = Out of Service (highest priority)	0x1c			

Quality	Substatus	Hex value	Cascade path		
			Not in	Forward	Backward
Uncertain	0 = Non-specific	0x40	X		
Uncertain	1 = Last Usable Value	0x44	X		
Uncertain	2 = Substitute	0x48	X		
Uncertain	3 = Initial Value	0x4c	X		
Uncertain	4 = Sensor Conversion not Accurate	0x50	X		
Uncertain	5 = Engineering Unit Range Violation	0x54	X		
Uncertain	6 = Sub-normal	0x58	X		

Quality	Substatus	Hex value	Cascade path		
			Not in	Forward	Backward
Uncertain	0 = Non-specific	0x40	X		
Uncertain	1 = Last Usable Value	0x44	X		
Uncertain	2 = Substitute	0x48	X		
Uncertain	3 = Initial Value	0x4c	X		
Uncertain	4 = Sensor Conversion not Accurate	0x50	X		
Uncertain	5 = Engineering Unit Range Violation	0x54	X		
Uncertain	6 = Sub-normal	0x58	X		

Quality	Sub-status	Hex value	Cascade path		
			Not in	Forward	Backward
GoodNC	0 = Non-specific (lowest priority)	0x80	X		
GoodNC	1 = Active Block Alarm	0x84	X		
GoodNC	2 = Active Advisory Alarm	0x88	X		
GoodNC	3 = Active Critical Alarm	0x8c	X		
GoodNC	4 = Unacknowledged Block Alarm	0x90	X		
GoodNC	5 = Unacknowledged Advisory Alarm	0x94	X		
GoodNC	6 = Unacknowledged Critical Alarm	0x98	X		

Quality	Sub-status	Hex value	Cascade path		
			Not in	Forward	Backward
GoodC	0 = Non-specific	0xc0		X	X
GoodC	1 = Initialization Acknowledge(IA)	0xc4		X	
GoodC	2 = Initialization Request(IR)	0xc8			X
GoodC	3 = Not Invited (NI)	0xcc			X
GoodC	4 = Not Selected(NS)	0xd0			X
GoodC	6 = Local Override(LO)	0xd8			X
GoodC	7 = Fault State Active(FSA)	0xdc			X
GoodC	8 = Initiate Fault State (IFS)	0xe0		X	

Conversion from index to number

The following formula is used to obtain the enumeration number of a determinate status attribute:

- $\text{Decimal Value Status} = 64 * \text{Quality} + 4 * \text{Sub-Status} + \text{Limit}$

For example, considering the following status:

- "Uncertain - Initial Value - High Limited"

Quality = "uncertain" = 1, Sub-Status = "Initial Value" = 3, Limit = "High Limited" = 2.

Applying the formula:

- $\text{Decimal Value Status} = 64 * 1 + 4 * 3 + 2 = 78$ (in decimal) or 0x4E (in Hexadecimal)

Conversion from number to index

a) Expressing the number in binary.

- $\text{Hex Value Status} = 78 = 0x4E = 01001110$ (in binary)

Dividing this binary number in quality, sub-status and limit fields:

- Quality = $01 = 1$ = "Uncertain", Sub-Status = $0011 = 3$ = "Initial Value",
Limit = $10 = 2$ = "High Limited"
The corresponding status is "Uncertain - Initial Value - High Limited".

b) Expressing the value of status in decimal format.

- $\text{Decimal Value Status} = 78$
Divide the number by 64. The quotient will be the quality and save the remainder:
 $\text{Quality} = 78 / 64 = 1$
- $\text{Remainder} = 14$. Divide the remainder by 4. The quotient will be the substatus and the remainder will be the limit:
 $\text{SubStatus} = 14 / 4 = 3$
Limit = 2

Appendix B: Standard data types and structures

This chapter contains definitions of every standard data type and structure used in the system.

Data Type	Description
Boolean	True or false
Integer8	
Integer16	
Integer32	
Unsigned8	
Unsigned16	
Unsigned32	
FloatingPoint	
VisibleString	they are one byte per character, and include the 7 bit ASCII character set.
OctetString	Octet strings are binary.
Date	
TimeofDay	
TimeDifference	
BitString	
DateTimeValue	

Block DS-64

This data structure comprises the attributes of a block.

E	Element Name	Data Type	Size
1	Block Tag	VisibleString	32
2	DD MemberId	Unsigned32	4
3	DD ItemId	Unsigned32	4
4	DD Revision	Unsigned16	2
5	Profile	Unsigned16	2
6	Profile Revision	Unsigned16	2
7	Execution Time	Unsigned32	4
8	Period of Execution	Unsigned32	4
9	Number of Parameters	Unsigned16	2
10	Next FB to Execute	Unsigned16	2
11	Starting Index of Views	Unsigned16	2
12	NumberOfVIEW_3	Unsigned8	1
13	NumberOfVIEW_4	Unsigned8	1

Analog Value & Status DS-65

This data structure comprises the value and status of inputs and outputs that are floating point parameters.

E	Element Name	Data Type	Size
1	Status	Unsigned8	1
2	Value	Float	4

Discrete Value & Status DS-66

This data structure consists of the value and status of discrete value parameters.

E	Element Name	Data Type	Size
1	Status	Unsigned8	1
2	Value	Unsigned8	1

**Scaling
DS-68**

This data structure consists of the static data used to scale floating point values for display purposes.

E	Element Name	Data Type	Size
1	EU at 100%	Float	4
2	EU at 100%	Float	4
3	Units Index	Unsigned16	2
4	Decimal Point	Integer8	1

**Mode
DS-69**

This data structure consists of bit strings for target, actual, permitted, and normal modes.

E	Element Name	Data Type	Size
1	Target	Bitstring	1
2	Actual	Bitstring	1
3	Permitted	Bitstring	1
4	Normal	Bitstring	1

**Access Permissions
DS-70**

This data structure consists of access control flags for access to block parameters.

E	Element Name	Data Type	Size
1	Grant	Bitstring	1
2	Deny	Bitstring	1

**Alarm Float
DS-71**

This data structure consists of data that describes floating point alarms.

E	Element Name	Data Type	Size
1	Unacknowledged	Unsigned8	1
2	Alarm State	Unsigned8	1
3	Time Stamp	Time Value	8
4	Subcode	Unsigned16	2
5	Value	Float	4

**Alarm Discrete
DS-72**

This data structure consists of data that describes discrete alarms.

E	Element Name	Data Type	Size
1	Unacknowledged	Unsigned8	1
2	Alarm State	Unsigned8	1
3	Time Stamp	Time Value	8
4	Subcode	Unsigned16	2
5	Value	Unsigned8	1

**Event Update
DS-73**

This data structure consists of data that describes a static revision alarm.

E	Element Name	Data Type	Size
1	Unacknowledged	Unsigned8	1
2	Update State	Unsigned8	1
3	Time Stamp	Time Value	8
4	Static Revision	Unsigned16	2
5	Relative Index	Unsigned16	2

**Alarm Summary
DS-74**

This data structure consists of data that summarize 16 alerts.

E	Element Name	Data Type	Size
1	Current	Bit String	2
2	Unacknowledged	Bit String	2
3	Unreported	Bit String	2
4	Disabled	Bit String	2

**Simulate Floating Point
DS-82**

This data structure consists of a simulate and transducer floating point value and status and a simulate enable/disable discrete.

E	Element Name	Data Type	Size
1	Simulate Status	Unsigned8	1
2	Simulate Value	Float	4
3	Transducer Status	Unsigned8	1
4	Transducer Value	Float	4
5	Simulate En/Disable	Unsigned8	1

**Simulate Discrete
DS-83**

This data structure consists of a simulate and transducer discrete value and status and a simulate enable/disable discrete.

E	Element Name	Data Type	Size
1	Simulate Status	Unsigned8	1
2	Simulate Value	Unsigned8	4
3	Transducer Status	Unsigned8	1
4	Transducer Value	Unsigned8	4
5	Simulate En/Disable	Unsigned8	1

**Test Read/Write
DS-85**

This data structure consists of function block test read/write data.

E	ElementName	Data Type	Size
1	Value1	Boolean	1
2	Value2	Integer8	1
3	Value3	Integer16	2
4	Value4	Integer32	4
5	Value5	Unsigned8	1
6	Value6	Unsigned16	2
7	Value7	Unsigned32	4
8	Value8	FloatingPoint	4
9	Value9	VisibleString	32
10	Value10	OctetString	32
11	Value11	Date	7
12	Value12	Time of Day	6
13	Value13	Time Difference	6
14	Value14	Bitstring	2
15	Value15	Time Value	8

**Value And Status
Unsigned16
DS-145**

This data structure is used to transmit value and status of PROFIBUS device values.

E	Element Name	Data Type	Size
1	Status	Unsigned16	1
2	Value	Unsigned16	2

**Discrete Structure
DS-159**

This data structure consists of one status and eight discrete value parameters.

E	Element Name	Data Type	Size
1	Status	Unsigned8	1
2	Value1	Unsigned8	1
3	Value2	Unsigned8	1
4	Value3	Unsigned8	1
5	Value4	Unsigned8	1
6	Value5	Unsigned8	1
7	Value6	Unsigned8	1
8	Value7	Unsigned8	1
9	Value8	Unsigned8	1

**Discrete Structure
DS-160**

This data structure consists of one status and sixteen discrete value parameters.

E	Element Name	Data Type	Size
1	Status	Unsigned8	1
2	Value1	Unsigned8	1
3	Value2	Unsigned8	1
4	Value3	Unsigned8	1
5	Value4	Unsigned8	1
6	Value5	Unsigned8	1
7	Value6	Unsigned8	1
8	Value7	Unsigned8	1
9	Value8	Unsigned8	1
10	Value9	Unsigned8	1
11	Value10	Unsigned8	1
12	Value11	Unsigned8	1
13	Value12	Unsigned8	1
14	Value13	Unsigned8	1
15	Value14	Unsigned8	1
16	Value15	Unsigned8	1
17	Value16	Unsigned8	1

Appendix C: Manufacturer-specific data structures

This appendix contains the manufacturer-specific data structures used in the system.

**Scaling Conversion
DS-256**

This data structure consists of data used to generate constants A and B in equation $Y = A \cdot X + B$.

E	Element Name	Data Type	Size
1	From EU 100%	Float	4
2	From EU 0%	Float	4
3	To EU 100%	Float	4
4	To EU 0%	Float	4
5	Data Type	Unsigned8	1

**Scaling Conversion with
Status
DS-257**

This data structure consists of data used to generate constants A and B in equation $Y = A \cdot X + B$ plus the output status.

E	Element Name	Data Type	Size
1	From EU 100%	Float	4
2	From EU 0%	Float	4
3	To EU 100%	Float	4
4	To EU 0%	Float	4
5	Data Type	Unsigned8	1
6	Output Status	Unsigned8	1

**Scaling Locator
DS-258**

This data structure consists of data used to generate constants A and B in equation $Y = A \cdot X + B$ plus the addresses in a slave device.

E	Element Name	Data Type	Size
1	From EU 100%	Float	4
2	From EU 0%	Float	4
3	To EU 100%	Float	4
4	To EU 0%	Float	4
5	Data Type	Unsigned8	1
6	Slave Address	Unsigned8	1
7	Modbus Address of Value	Unsigned16	2

**Scaling Locator with Status
DS-259**

This data structure consists of data used to generate constants A and B in equation $Y = A \cdot X + B$ plus the addresses in a slave device.

E	Element Name	Data Type	Size
1	From EU 100%	Float	4
2	From EU 0%	Float	4
3	To EU 100%	Float	4
4	To EU 0%	Float	4
5	Data Type	Unsigned8	1
6	Slave Address	Unsigned8	1
7	Modbus Address of Value	Unsigned16	2
8	Modbus Address of Status	Unsigned16	2

**Modbus Variable Locator
DS-260**

This data structure consists of data indicating the addresses in a slave device.

E	Element Name	Data Type	Size
1	Slave Address	Unsigned8	1
2	Modbus Address of Value	Unsigned16	2

**Modbus Variable Locator
with Status
DS-261**

This data structure consists of data indicating the addresses in a slave device.

E	Element Name	Data Type	Size
1	Slave Address	Unsigned8	1
2	Modbus Address of Value	Unsigned16	2
3	Modbus Address of Status	Unsigned16	2

**FF Parameter ID
DS-262**

This data structure consists of data informing the position of the FF parameter requested.

E	Element Name	Data Type	Size
1	Block Tag	VisibleString(32)	32
2	Relative Index	Unsigned16	2
3	Sub Index	Unsigned8	1

**Slave Address
DS-263**

This data structure consists of data informing the IP address and the Modbus address of the slaves.

E	Element Name	Data Type	Size
1	IP Slave1	VisibleString(16)	16
2	IP Slave2	VisibleString(16)	16
3	IP Slave3	VisibleString(16)	16
4	IP Slave4	VisibleString(16)	16
5	IP Slave5	VisibleString(16)	16
6	IP Slave6	VisibleString(16)	16
7	IP Slave7	VisibleString(16)	16
8	IP Slave8	VisibleString(16)	16
9	Slave Address1	Unsigned8	1
10	Slave Address2	Unsigned8	1
11	Slave Address3	Unsigned8	1
12	Slave Address4	Unsigned8	1
13	Slave Address5	Unsigned8	1
14	Slave Address6	Unsigned8	1
15	Slave Address7	Unsigned8	1
16	Slave Address8	Unsigned8	1

Index

A

ACK_OPTION	52
Adaptive gain	105
Advanced Equations block	209
Advanced PID Control	103
Alarm	12
Alarm display	43
Alert handling	39
Alert objects	12
Analog Alarm block	148
Analog Input block	66
Analog output block	226
Anti-reset wind-up	105
Application Designer	16
Arithmetic block	116
Automatic (Auto)	19

B

Back calculation	27, 86
Backward path	32
Block modes	18
Block objects	12
Block parameters	
AALM block	154
Advanced PID block	107
AEQU block	210
AI block	71
AO block	231
ARTH block	123
CCHAR block	139
CHAR block	133
CT block	193
CTRW block	199
DENS block	215
Device transducer block	62
DI block	74
DO block	236
DP_MAO	271
DP_MAO block	255
DP_MDI block	259
DP_MDO block	278
Enhanced PID control	102
ESPG block	168
FFET block	208
HC block	58
HY_EMB_IO block	225
HY_MA_IO block	218
HY_MD_IO block	221
INTG block	146
ISEL block	160
LL block	183
MAI block	80
MAO block	240
MBCM block	292
MBCS block	299
MDI block	82

MDO block	243
OS block	129
OSDL block	191
PA_AI block	252
PA_AO block	267
PA_DI block	257
PA_DO block	275
PA_TO block	261
PI block	78
PID block	98
PROFIBUS Transducer block	248
Resource block	54
SPG block	166
STEP block	114
TEMP block	60
TMR block	177
BLOCK_ERR	42
Bock parameters	
EAAL block	155

C

Calculated variables	119
CAS_IN	26
Cascade (Cas)	19
Cascade control	32, 90, 112, 264
Cascade initialization	33
Cascade Signal Characterizer block	134
Characterization dialog	17
Closed-loop control	87, 111
Commissioning	7
Constant and Contained RW block	194
Constant block	192
Contained parameters	13
Control Blocks	83
Control loop integrity	31
Control strategy	16
CONTROL_OPTS	26
ControlCare documents	9

D

Density block	211
Discrete Input block	72
Discrete output block	232
Dynamic parameters	13

E

EMV	7
Enhanced PID Control	100
Enhanced Setpoint Ramp Generator block	168
Error	106
Event	12
Events and alarms	39

F

Fault state handling	34
Fault state mechanism, PROFIBUS	246
Feedforward control	88

Field Controller	9	ALARM_HYS	67, 75
Flip-flop and Edge Trigger block	201	ALARM_SUM	42, 69, 73, 76
Forward path	31	ALERT_KEY	15
Function block	12	ARITH_TYPE	117, 119
Function block application process	11	BIAS	117
Function block options	36	BKCAL_IN, BKCAL_OUT	27, 86
G		BLOCK_ERR, BLOCK_ALM	42
Gain	85, 106	CHANNEL	59, 61, 65, 66, 72, 75, 79, 81
H		CONTROL_OPTS	27, 38
Hardware Configuration	57	DISC_ALM	72
Hybrid block with Analog I/O	216	DISC_PRI	72
Hybrid block with Discrete I/O	219	FEATURES	40
Hybrid block with Mixed I/O	222	FEATURES, FEATURE_SEL	51, 53
I		FF_VAL	84
Initialization manual (Iman)	18	FIELD_VAL	67, 69, 76
Input parameters	13	FSAFE_TYPE	67
Input Selector block	156	FSAFE_VAL	67
Installation	7	FSTATE_TIME, FSTATE_VAL, FSTATE_VAL_D	34
Integrator block	140	GAIN	117, 119
L		GRANT_DENY	36
Lead-Lag block	179	INPUT_OPTS	117
Limit alarms	41	IO_OPTS	27, 34, 37, 67
Linearization	23	IO_TYPE_Rxx	57
Link objects	12	L_TYPE	23, 66
Local override (LO)	18	MODE_BLK	18
M		OSDL_OPTS	34
Manual (Man)	18	OUT	27, 45, 69, 76, 80, 82
MBCF Modbus Configuration Block	283	OUT_D	73
MBCM Modbus Control Master Block	286	OUT_RANGE	24, 117
MBCS Modbus Control Slave block	293	OUT_SCALE	24, 67, 75
Modbus Blocks	279	PRE_OUT	119
Mode priority	20	PV	23, 69, 76, 84, 97
Mode shedding	22	PV_D	73
Multiple Analog Input block	79	PV_FTIME	23, 67, 72, 75
Multiple Analog Output block	237	RCAS_IN, RCAS_OUT	93
Multiple Discrete Input block	81	READBACK	45
Multiple Discrete Output block	241	RESTART	51
N		ROUT_IN, ROUT_OUT	95
Non-volatile parameters	13	RS_STATE	50
O		SHED_OPT	22
Off Line Characterization	17, 46	SIMULATE	45, 69, 76
On Line Characterization	17, 47	SIMULATE_D	73
Operation	7	SLOT_x	57
Output limits	27	SP	26, 85
Out of service (OOS)	18	ST_REV	15
Output parameters	13	STATUS_OPTS	34
Output Signal Selector and Dynamic Limiter block	184	STRATEGY	15
Output Splitter block	124	TAG_DESC	15
Output tracking	86, 92	TRK_VAL	92
P		UPDATE_EVT	15, 42
Parameter		WRITE_LOCK	53
ACK_OPTION	40	XD_ERROR	42
		XD_SCALE	24, 67
		XD_STATE	73
		XXX_ALM	67, 75
		XXX_PRI	67, 75
		xxx_PRI	52
		Parameter identifiers	14
		Parameter status	

Substatus values	300
Parameter types	13
PID algorithm	85, 104
PID Control	83
PID control algorithm	85
Process value	84
PROFIBUS	
Analog Input block	66, 72
PROFIBUS Analog Input block	252
PROFIBUS Analog Output block	262
PROFIBUS Blocks	244
PROFIBUS Digital Input block	257
PROFIBUS Discrete Output block	273
PROFIBUS Multiple Analog Input block	253
PROFIBUS Multiple Analog Output block	269
PROFIBUS Multiple Discrete Input block	258
PROFIBUS Multiple Discrete Output block	276
PROFIBUS output block	19
PROFIBUS process image	245
PROFIBUS slave configuration	247
PROFIBUS Totalizer block	260
PROFIBUS Transducer block	248
Project File	46, 47
Pulse Input block	75
PV_SCALE	24
R	
Ramp generator	161
Range extension	117
Rate	85
Remote cascade	93, 265
Remote cascade (RCas)	19
Remote output	95
Remote output (ROut)	19
Reporting, priority and acknowledgement	40
Reset	85
Resource block	12

S

Safety conventions	8
Scaling parameters	24
Setpoint limits	26
Setpoint Ramp Generator block	161
Setpoint rate limits	26
Setpoint tracking	27
Setpoint value	85
SFC445 temperature ranges	70
Signal Characterizer block	130
Simulation	45
ST_REV	39
Static parameters	13
Status propagation	28
Step Output PID	108
Substatus and limit propagation	30
Substatus values	29

T

Temperature block	59
Time constant	23
Timer	162, 170
Timer and Logic block	169
Transducer block	12, 56
Trouble-shooting	21

U

Universal parameters	15
UPDATE_EVT	39

V

View and trend objects	12
------------------------------	----

W

Write lock	53
WRITE_LOCK	39

www.endress.com/worldwide
