SoMachine

Booster Pumping Application Functions
Pumping Library Guide

06/2017

EIO0000001340.02

www.schneider-electric.com

The information provided in this documentation contains general descriptions and/or technical characteristics of the performance of the products contained herein. This documentation is not intended as a substitute for and is not to be used for determining suitability or reliability of these products for specific user applications. It is the duty of any such user or integrator to perform the appropriate and complete risk analysis, evaluation and testing of the products with respect to the relevant specific application or use thereof. Neither Schneider Electric nor any of its affiliates or subsidiaries shall be responsible or liable for misuse of the information contained herein. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us.

No part of this document may be reproduced in any form or by any means, electronic or mechanical, including photocopying, without express written permission of Schneider Electric.

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

© 2017 Schneider Electric. All Rights Reserved.

Table of Contents

Safety Information 7
About the Book 11
Chapter 1 Pumping Functions 15
1.1 PumpPidStag: Pump Stage / Destage 17
Description - PumpPidStag 18
Function Block Description - PumpPidStag 26
Structure Data Type Definitions - PumpPidStag 31
1.2 DevSwcPumpCtrl: Device Switching Pump Control 39
Description - DevSwcPumpCtrl 40
Function Block Description - DevSwcPumpCtrl 47
Structure Data Type Definitions - DevSwcPumpCtrl 51
Configuration Examples - DevSwcPumpCtrl 53
1.3 VsdCtrl: VSD Control 62
Description - VsdCtrl 63
Function Block Description - VsdCtrl 64
1.4 VsPumpCtrl: Variable Speed Pump Control 66
Description - VsPumpCtrl 67
Function Block Description - VsPumpCtrl 68
1.5 FsPumpCtrl: Fixed Speed Pump Control 70
Description - FsPumpCtrl 71
Function Block Description - FsPumpCtrl 72
1.6 CompSp: Setpoint Adaptation (Friction Loss Compensation) 74
Description - CompSp 75
Function Block Description - CompSp 78
Structure Data Type Definitions - CompSp 80
1.7 CompSpFlow: Setpoint Adaptation by Using Flow Value (Friction Loss Compensation) 81
Description - CompSpFlow 82
Function Block Description - CompSpFlow 84
Structure Data Type Definitions - CompSpFlow 86
1.8 CavtProt: Cavitation Protection 89
Description - CavtProt 90
Function Block Description - CavtProt 95
Structure Data Type Definitions - CavtProt 97

1.9 PumpAvai: Pump Availability Status 99
Description - PumpAvai 100
Function Block Description - PumpAvai 101
1.10 VsdAvai: VSD Available Status 103
Description - VsdAvai 104
Function Block Description - VsdAvai 105
1.11 OpPrty: Operation Priority Device 107
Description - OpPrty 108
Function Block Description - OpPrty 110
Structure Data Type Definitions - OpPrty 112
1.12 AuxPumpCtrl: Auxiliary Pump Control 113
Description - AuxPumpCtrl 114
Function Block Description - AuxPumpCtrl 118
Structure Data Type Definitions - AuxPumpCtrl 121
1.13 PipeFill: Pipe Fill Control 123
Description - PipeFill 124
Function Block Description - PipeFill 128
Structure Data Type Definitions - PipeFill 130
1.14 OpIntPump: Operator Interface for Pump Handling 133
Description - OpIntPump 134
Function Block Description - OpIntPump 137
Structure Data Type Definitions - OpIntPump 141
1.15 OpIntAuxPump: Auxiliary Pump Operator Interface Function 146
Description - OpIntAuxPump 147
Function Block Description - OpIntAuxPump 150
Structure Data Type Definitions - OpIntAuxPump 152
1.16 AnaIput: Analog Input Value Control 155
Description - AnaIput 156
Function Block Description - AnaIput 158
Structure Data Type Definitions - AnaIput 160
1.17 Common Structure Data Type Definitions 163
Common Structure Data Type Definitions 163
Appendices 175

Appendix A Function and Function Block Representation 177
Differences Between a Function and a Function Block 178
How to Use a Function or a Function Block in IL Language 179
How to Use a Function or a Function Block in ST Language 182
Glossary 185
Index 189

Safety Information

Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, service, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of this symbol to a “Danger” or “Warning” safety label indicates that an electrical hazard exists which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

DANGER

DANGER indicates a hazardous situation which, if not avoided, **will result in** death or serious injury.

WARNING

WARNING indicates a hazardous situation which, if not avoided, **could result in** death or serious injury.

CAUTION

CAUTION indicates a hazardous situation which, if not avoided, **could result** in minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

BEFORE YOU BEGIN

Do not use this product on machinery lacking effective point-of-operation guarding. Lack of effective point-of-operation guarding on a machine can result in serious injury to the operator of that machine.

⚠ WARNING

UNGUARDED EQUIPMENT

- Do not use this software and related automation equipment on equipment which does not have point-of-operation protection.
- Do not reach into machinery during operation.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

This automation equipment and related software is used to control a variety of industrial processes. The type or model of automation equipment suitable for each application will vary depending on factors such as the control function required, degree of protection required, production methods, unusual conditions, government regulations, etc. In some applications, more than one processor may be required, as when backup redundancy is needed.

Only you, the user, machine builder or system integrator can be aware of all the conditions and factors present during setup, operation, and maintenance of the machine and, therefore, can determine the automation equipment and the related safeties and interlocks which can be properly used. When selecting automation and control equipment and related software for a particular application, you should refer to the applicable local and national standards and regulations. The National Safety Council's Accident Prevention Manual (nationally recognized in the United States of America) also provides much useful information.

In some applications, such as packaging machinery, additional operator protection such as point-of-operation guarding must be provided. This is necessary if the operator's hands and other parts of the body are free to enter the pinch points or other hazardous areas and serious injury can occur. Software products alone cannot protect an operator from injury. For this reason the software cannot be substituted for or take the place of point-of-operation protection.

Ensure that appropriate safeties and mechanical/electrical interlocks related to point-of-operation protection have been installed and are operational before placing the equipment into service. All interlocks and safeties related to point-of-operation protection must be coordinated with the related automation equipment and software programming.

NOTE: Coordination of safeties and mechanical/electrical interlocks for point-of-operation protection is outside the scope of the Function Block Library, System User Guide, or other implementation referenced in this documentation.

START-UP AND TEST

Before using electrical control and automation equipment for regular operation after installation, the system should be given a start-up test by qualified personnel to verify correct operation of the equipment. It is important that arrangements for such a check be made and that enough time is allowed to perform complete and satisfactory testing.

⚠ WARNING

EQUIPMENT OPERATION HAZARD

- Verify that all installation and set up procedures have been completed.
- Before operational tests are performed, remove all blocks or other temporary holding means used for shipment from all component devices.
- Remove tools, meters, and debris from equipment.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Follow all start-up tests recommended in the equipment documentation. Store all equipment documentation for future references.

Software testing must be done in both simulated and real environments.

Verify that the completed system is free from all short circuits and temporary grounds that are not installed according to local regulations (according to the National Electrical Code in the U.S.A, for instance). If high-potential voltage testing is necessary, follow recommendations in equipment documentation to prevent accidental equipment damage.

Before energizing equipment:

- Remove tools, meters, and debris from equipment.
- Close the equipment enclosure door.
- Remove all temporary grounds from incoming power lines.
- Perform all start-up tests recommended by the manufacturer.

OPERATION AND ADJUSTMENTS

The following precautions are from the NEMA Standards Publication ICS 7.1-1995 (English version prevails):

- Regardless of the care exercised in the design and manufacture of equipment or in the selection and ratings of components, there are hazards that can be encountered if such equipment is improperly operated.
- It is sometimes possible to misadjust the equipment and thus produce unsatisfactory or unsafe operation. Always use the manufacturer's instructions as a guide for functional adjustments. Personnel who have access to these adjustments should be familiar with the equipment manufacturer's instructions and the machinery used with the electrical equipment.
- Only those operational adjustments actually required by the operator should be accessible to the operator. Access to other controls should be restricted to prevent unauthorized changes in operating characteristics.

About the Book

At a Glance

Document Scope

This document describes the implementation of the Pumping functionality in SoMachine.

This document is adapted to those individuals knowledgeable and experienced in pumping technologies.

The following basic knowledge is required:

- basic information on functionality, structure and configuration of the controllers, drives and HMI displays
- programming in the FBD, LD, ST, IL or CFC language

Validity Note

This document has been updated for the release of SoMachine V4.3.

Related Documents

Document title	Reference
SoMachine Programming Guide	*EIO0000000067 (ENG)*; *EIO0000000069 (FRE)*; *EIO0000000068 (GER)*; *EIO0000000071 (SPA)*; *EIO0000000070 (ITA)*; *EIO0000000072 (CHS)*

You can download these technical publications and other technical information from our website at *http://www.schneider-electric.com/en/download*.

Product Related Information

⚠ WARNING

LOSS OF CONTROL

- The designer of any control scheme must consider the potential failure modes of control paths and, for certain critical control functions, provide a means to achieve a safe state during and after a path failure. Examples of critical control functions are emergency stop and overtravel stop, power outage and restart.
- Separate or redundant control paths must be provided for critical control functions.
- System control paths may include communication links. Consideration must be given to the implications of unanticipated transmission delays or failures of the link.
- Observe all accident prevention regulations and local safety guidelines.[1]
- Each implementation of this equipment must be individually and thoroughly tested for proper operation before being placed into service.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

[1] For additional information, refer to NEMA ICS 1.1 (latest edition), "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control" and to NEMA ICS 7.1 (latest edition), "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" or their equivalent governing your particular location.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Only use software approved by Schneider Electric for use with this equipment.
- Update your application program every time you change the physical hardware configuration.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Terminology Derived from Standards

The technical terms, terminology, symbols and the corresponding descriptions in this manual, or that appear in or on the products themselves, are generally derived from the terms or definitions of international standards.

In the area of functional safety systems, drives and general automation, this may include, but is not limited to, terms such as *safety*, *safety function*, *safe state*, *fault*, *fault reset*, *malfunction*, *failure*, *error*, *error message*, *dangerous*, etc.

Among others, these standards include:

Standard	Description
EN 61131-2:2007	Programmable controllers, part 2: Equipment requirements and tests.
ISO 13849-1:2008	Safety of machinery: Safety related parts of control systems. General principles for design.
EN 61496-1:2013	Safety of machinery: Electro-sensitive protective equipment. Part 1: General requirements and tests.
ISO 12100:2010	Safety of machinery - General principles for design - Risk assessment and risk reduction
EN 60204-1:2006	Safety of machinery - Electrical equipment of machines - Part 1: General requirements
EN 1088:2008 ISO 14119:2013	Safety of machinery - Interlocking devices associated with guards - Principles for design and selection
ISO 13850:2006	Safety of machinery - Emergency stop - Principles for design
EN/IEC 62061:2005	Safety of machinery - Functional safety of safety-related electrical, electronic, and electronic programmable control systems
IEC 61508-1:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: General requirements.
IEC 61508-2:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Requirements for electrical/electronic/programmable electronic safety-related systems.
IEC 61508-3:2010	Functional safety of electrical/electronic/programmable electronic safety-related systems: Software requirements.
IEC 61784-3:2008	Digital data communication for measurement and control: Functional safety field buses.
2006/42/EC	Machinery Directive
2014/30/EU	Electromagnetic Compatibility Directive
2014/35/EU	Low Voltage Directive

In addition, terms used in the present document may tangentially be used as they are derived from other standards such as:

Standard	Description
IEC 60034 series	Rotating electrical machines
IEC 61800 series	Adjustable speed electrical power drive systems
IEC 61158 series	Digital data communications for measurement and control – Fieldbus for use in industrial control systems

Finally, the term *zone of operation* may be used in conjunction with the description of specific hazards, and is defined as it is for a *hazard zone* or *danger zone* in the *Machinery Directive* (*2006/42/EC*) and *ISO 12100:2010*.

NOTE: The aforementioned standards may or may not apply to the specific products cited in the present documentation. For more information concerning the individual standards applicable to the products described herein, see the characteristics tables for those product references.

Chapter 1
Pumping Functions

Overview

This chapter describes the functions included in the Pumping Library.

This library contains function blocks that allow you to execute the following tasks:

- pump PID stage control *(see page 17)*
- device switching control *(see page 39)*
- FS pump control *(see page 74)*, VS pump control *(see page 70)*, VSD control *(see page 66)*
- friction loss compensation basic *(see page 76)* and friction loss compensation by using the flow value to calculate an adjusted setpoint *(see page 77)*
- cavitation protection *(see page 91)*
- pump availability status verification *(see page 101)* and VSD availability status verification *(see page 105)*
- detection of the device priority *(see page 109)*
- auxiliary pump control *(see page 115)*
- pipe fill control *(see page 125)*
- control by operator interface *(see page 135)*
- auxiliary pump control by operator interface *(see page 148)*

What Is in This Chapter?

This chapter contains the following sections:

Section	Topic	Page
1.1	`PumpPidStag`: Pump Stage / Destage	17
1.2	`DevSwcPumpCtrl`: Device Switching Pump Control	39
1.3	`VsdCtrl`: VSD Control	64
1.4	`VsPumpCtrl`: Variable Speed Pump Control	68
1.5	`FsPumpCtrl`: Fixed Speed Pump Control	72
1.6	`CompSp`: Setpoint Adaptation (Friction Loss Compensation)	76
1.7	`CompSpFlow`: Setpoint Adaptation by Using Flow Value (Friction Loss Compensation)	83
1.8	`CavtProt`: Cavitation Protection	91
1.9	`PumpAvai`: Pump Availability Status	101
1.10	`VsdAvai`: VSD Available Status	105
1.11	`OpPrty`: Operation Priority Device	109

Section	Topic	Page
1.12	`AuxPumpCtrl`: Auxiliary Pump Control	115
1.13	`PipeFill`: Pipe Fill Control	125
1.14	`OpIntPump`: Operator Interface for Pump Handling	135
1.15	`OpIntAuxPump`: Auxiliary Pump Operator Interface Function	148
1.16	`AnaIput`: Analog Input Value Control	157
1.17	Common Structure Data Type Definitions	166

Section 1.1
PumpPidStag: Pump Stage / Destage

Overview

This section describes the PumpPidStag function block that consists of a PID function and a stage / destage pumping management.

It executes two main tasks to help to maintain a constant pressure in a water distribution network:

- The PID function is used to generate a PID control value (called the control value throughout this document). This value is used for the speed reference for the VSDs.
- The stage function is used to generate a stage value for the pumps. This stage value indicates whether an additional pump has to be switched on or a running pump has to be switched off.

NOTE: This function block is designed to be used in conjunction with other function blocks of the Pumping Library.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description - PumpPidStag	18
Function Block Description - PumpPidStag	25
Structure Data Type Definitions - PumpPidStag	30

Description - PumpPidStag

Overview

The PumpPidStag function block consists of a PID control function and a stage / destage function to execute the following main tasks:

- Generating a PID control value with the PID control function. This value is used for the speed reference for the VSDs.
- Generating a pump stage value with the stage function. The stage value indicates the demand of the number of pumps that is needed to maintain a constant pressure in a water distribution network. The stage value calculation is based either on pressure values (pressure, pressure setpoint influenced by other functions) or on the flow in the network. Both stage solutions use the control value as a speed reference for the VSDs.

PID Control Function

The PID control function of the PumpPidStag function block is used to generate a control value. This value is used for the speed reference for the VSDs and the stage function.

To achieve this, the PID control function of the PumpPidStag function block executes the following tasks:

- Generates a control PID control value in the range of 0...100% with the PID control function. This value is used for the speed reference for the VSDs.
- Limits the integral (I) part of the control as follows:
  Imax = 100% – P (proportional) part value if P = positive.
  This limitation of the PID control value leads to a saturation and helps to avoid long reaction times.
- Limits the I-part incremental to maximal increment = stPidInit.rMaxErrForInt. If the calculated increment is greater than stPidInit.rMaxErrForInt, then the value configured for this parameter is used as the incremental value for the I-part calculation. This function helps to limit the gradient of the I curve.
- Defines a dead band value for the error value *e* to help to reduce the frequency of switching the VSDs and thus to help limit wear. The dead band value reduces oscillation of the control value that is caused by disturbances of the discharge pressure. Oscillation of the control value by reaching a limit leads to superfluous switching actions.
- Accepts a setpoint that has been entered manually (with the input variable i_rManSp) if the PID control function is running in manual operating mode (stOpMode.xPidMan = TRUE (1)). This allows you to run the system with a high pressure in an exceptional situation, for example, to extinguish a fire.
- Verifies specific initialization data.
- Switches off the derivative part (D) if the PID control function is running in manual operating mode (stOpMode.xPidMan = TRUE (1)).
- Reinitializes the output and process variables to the default values if a reset is executed.
- Reinitializes the I part of the PID if a stop command is executed.
- Generates status bits (reporting the status) for the different modes and status situations.

Stage Function (Based on Pressure)

The stage function based on the control value is enabled by setting the input pin `i_esiStag` to the constant value `esiStag.Stage`.

This function calculates the number of pumps that is needed to maintain a constant pressure in the water distribution network based on the PID control value and a two-level control function (hysteresis-function). When the PID control value exceeds a limit, the stage value is increased or decreased.

The number of pumps is increased or decreased sequentially by using delay timers to avoid a high switching frequency and to allow the requested operation to be executed before the next operation is requested.

The delay principle works if one or more pumps are running and the PID parameters are adjusted correctly. However, up to the starting point of the first pump, the system has a delay, which can cause a fall below the pressure value under the tolerance range. To help avoid that, the function starts the first pump (stage value = 1) with the help of a deviation limit in percent (%). If the pressure value falls below this limit, the first pump is started (the stage value is set to the value 1).

Increase / decrease demand limits:

Stage limits for the control value of the PID controller

Limit	Description
`stStagInit.rLimStagPlus`	plus stage limit (0...100%)
`stStagInit.rLimStagMnus`	minus stage limit (0...100%)

The detection of the limit undershooting of the PID control value which is used to switch-off the last running pump is done by a dedicated PID control limit. This value should be lower than the stage minus limit value.

Limit	Description
`stStagInitC1.rLimPump1SwcOff`	control value limit pump 1 switch-off

As soon as the control value is beyond outside of these limits, the respective delay time is started. This delay time has to elapse before a switching process is executed. No switching processes are executed if the control value is between the 2 limit values (principle of a two-level PID controller).

Increase / decrease demand limits:

1 limit for decreasing stage value, if stage value = 1 (`stStagInit.rLimPump1SwcOff`)
2 limit for decreasing stage value, if stage value > 1 (`stStagInit.rLimStagMnus`)
3 limit for increasing stage value (`stStagInit.rLimStagPlus`)

This principle allows running the pumps in an energy efficient speed range (working range).

Exceeding the increase demand limit:

The switch-on delay timer (`stStagInit.tDlyStagPlus`) is started as soon as the control value exceeds the plus stage limit (`stStagInit.rLimStagPlus`).

After the switch-on delay time has elapsed, the stage value is increased by 1 until the `stStagInit.siStagPumpMax` value is reached. With this switching procedure, the switch-on delay timer (`stStagInit.tDlyStagPlus`) is re-started if the PID control value still exceeds the plus stage limit (`stStagInit.rLimStagPlus`).

As soon as the PID control value falls below the plus stage limit (`stStagInit.rLimStagPlus`), the switch-on delay timer (`stStagInit.tDlyStagPlus`) executes a reset.

Increase demand limit exceeded

1 start switch-on delay timer (`stStagInit.tDlyStagPlus`)
2 switch-on delay time elapsed and stage value increased by 1
3 reset switch-on delay timer (`stStagInit.tDlyStagPlus`)

Falling below the decrease demand limit:

The switch-off delay timer (`stStagInit.tDlyStagMnus`) is started as soon as the PID control value falls below the minus stage limit (`stStagInit.rLimStagMnus`).

After the switch-off delay time has elapsed, the stage value is decreased by 1 until the value reaches zero. With this switching procedure, the switch-off delay timer (`stStagInit.tDlyStagMnus`) is re-started, if the PID control value is still inferior to the following minus stage limits:

- `stStagInit.rLimStagMnus` and the stage value is > 1 or
- `stStagInit.rLimPump1SwcOff` and the stage value is 1

As soon as the PID control value exceeds the minus stage limit (`stStagInit.rLimStagMnus`, respectively `stStagInit.rLimPump1SwcOff`), the switch-off delay timer executes a reset.

Decrease demand limit undershot

1 start switch-off delay timer (`stStagInit.tDlyStagMnus`)
2 switch-off delay time elapsed and stage value decreased by 1
3 reset switch-off delay timer (`stStagInit.tDlyStagMnus`)

Extended Stage Function (Based on Flow and Pressure)

The extended stage function based on flow and pressure is enabled by setting the input pin `i_esiStag` to the constant `esiStag.Flow`.

NOTE: A flow meter is required in your system to use this function.

This function calculates the number of pumps that is needed to maintain a constant pressure in the water distribution network based on the PID control value and on the flow.

If the flow value becomes unavailable during operation, the application automatically activates the stage function using the calculated PID control value of the PID controller and the pressure value. In this case, the recording of the external value detects an error and the stage functions set a corresponding status flag (`stPidStagSta.xEmgyStagMode`), and the system continues operation.

Prerequisite

The flow value cannot be used to start the first pump. Due to this, the first pump is started if the PID control value exceeds the first pump switch-on limit (`stFlowStagInit.rCtrlVal-Pump1OnLim`). After a delay time has elapsed (`stFlowStagInit.tDelTimeSwcOn`), the first pump is started. As soon as the first pump is running, the extended stage function, based on flow, is available.

Starting the first pump

If the PID control value exceeds the limit defined with the parameter `stFlowStagInit.rCtrlValPump1OnLim`, then the delay time (`stFlowStagInit.tDelTimeSwcOn`) is started. After the delay time has elapsed, the first pump is started. With the start of the first pump, the flow value is rising.

The figure shows the switch-on condition for the first pump:

1 PID control limit (`stFlowStagInit.rCtrlValPump1OnLim`) to start the first pump
2 switch-on delay time (`stFlowStagInit.tDelTimeSwcOn`)
3 start of the first pump

Switching the pumps according to the flow

If the extended stage function is active, the pumps are switched according to the flow limits defined for a maximum of 8 pumps. Since there is no delay time used in this procedure, the reaction times are greater than for the stage function based on pressure. This kind of staging immediately stops the pumps in case of a pipe-burst, except the last pump.

Staging / destaging according to the flow value:

1 stage flow limit (stop) for a pump (here pump number 4)
2 stage flow limit (start) for a pump (here pump number 4)
3 maximum flow of one pump
4 maximum flow of two pumps
5 maximum flow of three pumps

Avoiding a deadlock by using the following principals

The speed of the running pump(s) must be reducible to a speed that is under the switch-off limit of the last pump to allow stopping the last fixed speed pump (FS pump), if FS pumps are used. Therefore, you must consider the minimum speed that is configured for the drive(s) to make sure that it is not greater than the switch-off limit.

Further, you need to have a minimum of 2 variable speed drives (VSDs) in the combination of variable speed pumps (VS pumps) and fixed speed pumps (FS pumps).

If the number of operational VSDs is reduced to under the 2 VSD minimum required by the stage flow method, the function `DevSwcPumpCtrl` sets the state of the output pin `q_xFlowStagEn` to FALSE. This pin is linked to the input pin `i_xFlowStageEn` of the `PumpPidStage` function. In this case, the stage function is switched to the method based on the pressure as opposed to the flow.

NOTE: Be sure that the stage function based upon pressure has been configured with the appropriate parameters to support this case.

Switching off the last pump:

Before switching off the last pump, there must be reserve pressure in the expansion tank to compensate for the delayed starting of the first pump when coming out of sleep mode. Therefore, it is not the simple matter of switching off the last pump when the flow value reaches a specific setpoint. When the flow reaches the lower limit, a new setpoint is calculated to create the required reserve pressure that is to be maintained by the auxiliary pump while in sleep mode. This new setpoint is the sum of the pressure reference value (`i_rSp`) plus a pressure increment that is configurable via `stFlowStagInit.rPresIncPump1SwcOff`. This new value is then used as the pressure lower limit for the purpose of switching off the last pump.

This incremental pressure value used to calculate the sleep mode pressure described above must be within the configured pressure limits of the auxiliary pump. A valid working value for the increment would correspond to the average of the limits of the auxiliary pump (`rPresAuxLimOn` + `rPresAuxLimOff`) / 2).

In addition, once the sleep mode pressure has been reached, a switch off delay timer is started before the last running pump is switched off. The delay timer is used to maintain a minimum runtime of the pump and minimum time between sleep mode intervals.

Switching off the last pump:

1 pressure for switching of the last pump (`stFlowStagInit.rPresIncPump1SwcOff` (2) + `i_rSp` (3))
2 configured pressure increment (`stFlowStagInit.rPresIncPump1SwcOff` (2))
3 pressure reference value (`i_rSp`)
4 switch-off demand based on flow for the last pump (`stFlowStagInit.rMinFlowLimPump1Off`)
5 sleep mode is executed
6 last pump is switched-off

Function Block Description - `PumpPidStag`

Pin Diagram

Brief Description

The `PumpPidStag` function block consists of a PID function and a stage / destage pumping management function to execute the following tasks:

- generating a PID control value. This value is used for the speed reference for the VSDs.
- generating a pump demand stage value

I/O Variables Description

The table describes the input/output variables of the function block:

Input/Output	Data Type	Description
`iq_stAlrmCtrl`	`stAlrmCtrl`	Structure used for alarm handling. Refer to the structure data type description *(see page 166)*.
`iq_stOpMode`	`stOpMode`	Determines the operating modes. If the element `xAuto` of the `stOpMode` structure data type is set to TRUE, the function block executes its tasks automatically. With another status of this element, the outputs of this function block are set to their defined fallback (default) states. If the element `xPidMan` of this structure data type is set to TRUE, it is allowed to enter a user-defined setpoint manually. If the element `xFill` of this structure data type is set to TRUE, the PID function is disabled if the pipe fill function is running in sequence 1. In this case the `iq_stFillDevInit` parameters are used to set the output pins `q_stRef_CtrlVal.stRef` and `q_stStagVal.siStag`. Refer to the structure data type description *(see page 166)*.
`iq_stPidInit`	`stPidInit`	Structure used for initialization data of the PID control function. Refer to the structure data type description *(see page 30)*.
`iq_stStagInit`	`stStagInit`	Structure used for initialization data of the stage function based on pressure. Refer to the structure data type description *(see page 31)*.
`iq_stFillDevInit`	`stFillDevInit`	Structure used for initialization data if sequence 1 of the pipe fill mode is active. Refer to the structure data type description *(see page 32)*.
`iq_stFlowStagInit`	`stFlowStagInit`	Structure used for initialization data of the stage function based on flow. Refer to the structure data type description *(see page 32)*.

The table describes the input variables to the function block:

Input	Data Type	Description
`i_xRele`	BOOL	Activate (TRUE) or stops (FALSE) the execution of the function block.
`i_esiStag`	`esiStag`	Activates the stage function and which method is selected: ● `i_esiStag = esiStag.Off`: stage function deactivated (default value) ● `i_esiStag = esiStag.Stage`: stage method by using only the pressure to generate the stage value is activated ● `i_esiStag = esiStag.Flow`: stage method by using additionally the flow value is activated
`i_rSp`	REAL	Contains the user-defined pressure setpoint that can be adapted by other functions (compensation functions *(see page 77)* or the cavitation protection function *(see page 92)*).
`i_stDiscPres`	`stDiscPres`	Contains the analog pressure input value provided by the `AnaIput` function block *(see page 161)*, and contains an optional pressure value from a second pressure sensor, if available in your system. Refer to the `stDiscPres` structure data type description *(see page 176)*.
`i_rManSp`	REAL	Contains the pressure setpoint that has been entered manually by the operator. This parameter is only taken into account if the pumping application is operated in PID manual mode (`stOpMode.xPidMan` = TRUE).
`i_stFillSta`	`stFillStat`	Structure that indicates the states of the pipe fill function. The PID control and stage functions of the `PumpPidStag` function block are disabled if the pipe fill function is running in sequence 1 (`xSequ1` = TRUE) because, in this case, a defined number of pumps is used at a fixed speed *(see page 127)*. If the pipe fill function has set the flag in the `stFillSta` structure to `xFillOK`, or if it is running in sequence 2 (`xSequ2` = TRUE), then the PID control, if released, enables the stage functions of the `PumpPidStag` function block. Otherwise the PID function including the stage function is disabled.

Input	Data Type	Description
i_stAna_Flow	stAna	Contains the flow value and its status. This input is only considered if the stage function based on flow is activated (input i_esiStag = Flow or i_esiStag = 2). Refer to the structure data type description *(see page 168)*.
i_xParaChg	BOOL	When TRUE, the new parameter values are used (mapping into internal variables after a valid verification).
i_xResSwcPrcs	BOOL	When TRUE, the counter of switching cycles (output q_auiTbl24hSwcPrcs) is reset to 0.
i_xFlowStageEn	BOOL	When TRUE, at least 2 VSDs and corresponding VS pumps are available. This means that the stage function based on flow is available. When FALSE, an insufficient number of VSDs / VS pumps is available. The stage function based on pressure is applied. The stage function based on flow is not allowed.

The table describes the output variables from the function block:

Output	Data Type	Description
q_xActv	BOOL	When TRUE, indicates that the PID function is active.
q_stRef_CtrlVal	stRef	Structure that indicates the PID control value and the state of this value. Refer to the structure data type description *(see page 177)*.
q_stPidFbck_Hmi	stPidFbck	Structure that is used to display the values of the different PID components on the HMI. Refer to the structure data type description *(see page 35)*.
q_stPidSta_Hmi	stPidSta	Structure that is used to display the status of the PID function on the HMI. Refer to the structure data type description *(see page 35)*.
q_stPidStagSta_Hmi	stPidStagSta	Structure that is used to display the status of the stage function on the HMI. Refer to the structure data type description *(see page 37)*.
q_stPidErr	stPidErr	Structure that indicates the alarm and alert states detected in the PID function of the PumpPidStag function block. Refer to the structure data type description of alarms and alerts *(see page 38)*.

Output	Data Type	Description
`q_stPidStagErr`	`stPidStagErr`	Structure that indicates the alarm and alert states detected in the stage function of the `PumpPidStag` function block. Refer to the structure data type description of alarms and alerts *(see page 38)*.
`q_auiTbl24hSwcPrcs`	ARRAY [0..23] of UINT	Contains counters indicating the number of switching cycles (pump on/off) per hour. The hour value (24-hour value) of the controller is used as a pointer to this 24-hour array. The 24-hour array is used as a circular buffer. **NOTE:** For example, after 48 hours, one hour value contains the switching value of 2 days. You can use this value to optimize the configuration of your system to reduce the number of switching cycles while maintaining the user-defined pressure. **NOTE:** This function is reserved to controllers that maintain a system clock.
`q_rSpAct`	REAL	Contains the present setpoint to display on the HMI.
`q_stStagVal`	`stStagVal`	Structure that indicates the number of pumps needed (demand) to maintain the user-defined pressure level. Refer to the structure data type description *(see page 169)*.

Structure Data Type Definitions - PumpPidStag

Overview

Structure data types serve as the major input and output data carriers between different function blocks, configuration, and parameterization data.

The following section lists the names of structures (data types) that are used in this function block, followed by a table that describes the elements of structure.

NOTE: For the structure data types that are used from this function block and others, refer to the Common Structure Data Type Definitions chapter *(see page 166)*.

Structure: stPidInit

The stPidInit structure data type contains the initialization data for the PID function of the PumpPidStag function block.

Element of stPidInit	Data Type	Description
rKp	REAL	Defines the proportional gain constant (Kp) for the proportional part (P) of the PID function. Default value: 0.001
rTn	REAL	Defines the integral time (Tn) for the integral part (I) of the PID function. Default value: 1000 ms
rTv	REAL	Defines the derivative time (Tv) for the derivative part (D) of the PID function. Default value: 0.0 ms
rTd	REAL	Defines the derivative damping time (Td) for the derivative part (D) of the PID function. Default value: 0.0 ms
rDbnd	REAL	Defines the value of the dead band for the error value (*e*) to help to reduce the frequency of switching the VSDs, and thus to help limit wear. A switching process is only initiated if the deviation from the error value (*e*) exceeds the value defined with this parameter. Default value: 0.0001
rLowLim	REAL	Defines the minimum output limit for the PID control value (in percent). Default value: 0.0
rHighLim	REAL	Defines the maximum output limit for the PID control value (in percent). Default value: 100.0

Element of stPidInit	Data Type	Description
rMaxErrForInt	REAL	Defines the maximum incremental value for the calculation of the I part. If the incremental value is greater than the value defined with this parameter, then the value of this parameter is used as incremental value. This function helps to limit the gradient of the I curve. If the value for this parameter is ≤ 0, a detected alarm is indicated with the parameter stPidAlrm.xMaxErrForIntPara. Default value: 50.0
tSwc	TIME	Defines the delay time in milliseconds for suppression of the D term in case it is switched from automatic setpoint to manual setpoint and vice versa. Default value: 100000.0

NOTE: The elements of the instance of the structure data type are set to the default values listed above if no other value is assigned.

Structure: stStagInit

The stStagInit structure data type contains the initialization data for the stage function of the PumpPidStag function block. The structure data type is embedded in the structure data type stInitApp.

Element of stStagInit	Data Type	Description
siStagPumpMax	SINT	Defines the total number of VS and FS pumps for the stage function.
rPresIncPump1SwcOff	REAL	Defines the increment that is added to the pressure setpoint value to calculate a pressure limit to switch off the last pump that is running *(see page 21)*. The calculated value consists of the pressure reference value (i_rSp) plus this incremental value to reach a pressure level that corresponds to the average value of the working range of the auxiliary pump. The increase of the pressure increases the quantity of water in the expansion tank. This procedure helps prevent an early restart of the pumps.
rLimStagPlus	REAL	Defines the limit to increment the stage value (0...100%) that is the demand to switch a pump to the ON state.
rLimStagMnus	REAL	Defines the limit to decrease the stage value (0...100%) that is the demand to switch a pump to the OFF state.

Element of `stStagInit`	Data Type	Description
`tDlyStagPlus`	TIME	Defines the delay time in milliseconds that has to elapse after the `rLimStagPlus` has been reached before a pump is switched on.
`tDlyStagMnus`	TIME	Defines the delay time in milliseconds that has to elapse after the `rLimStagMnus` has been reached before a pump is switched off.
`rPercDeriv`	REAL	Percent deviation value of the discharge pressure from the setpoint to start the first pump.

NOTE: The elements of the instance of the structure data type are set to the default values (0 or FALSE) assigned by the controller on system startup.

Structure: `stFillDevInit`

The `stFillDevInit` structure data type contains the initialization data for the `PumpPidStag` function block when sequence 1 of the pipe fill mode is active (`stOpMode.xFill` = TRUE and `stFillStat.xSequ1` = TRUE).

Element of `stFillDevInit`	Data Type	Description
`rCtrlVal`	REAL	Defines the output limit for the PID control value (in percent). Default value: 50.0%
`siNoPump`	SINT	Defines the number of pumps that have to run if sequence 1 of the pipe fill mode is active. Default value: 2

NOTE: The elements of the structure are set to the default values listed above.

Structure: `stFlowStagInit`

The `stFlowStagInit` structure data type contains the initialization data for the stage function of the `PumpPidStag` function block considering the flow value *(see page 21)*.

Element of `stFlowStagInit`	Data Type	Description
`siStagPumpMax`	SINT	Defines the total number of VS and FS pumps for the stage function based on flow.
`rMinFlowLimPump1Off`	REAL	Defines the minimum flow that has to be available to switch off the last pump (pump 1).

Element of `stFlowStagInit`	Data Type	Description
`rPresIncPump1SwcOff`	REAL	Defines the pressure that has to be available to switch off pump 1. This value consists of the pressure reference value (`i_rSp`) plus an additional incremental value to reach a pressure level that corresponds to the average value of the working range of the auxiliary pump.
`rCtrlValPump1OnLim`	REAL	Defines the minimum limit of the PID control value that has to be available to switch on pump 1. At least 1 pump has to run to generate a flow value. Only then you can use the stage function based on flow.
`tDelTimeSwcOn`	TIME	Defines the delay time in milliseconds that has to expire before the first pump is switched on.
`tDelTimeSwcOFF`	TIME	Defines a minimum time in milliseconds the first pump has to be running before it can be switched off again. It is the minimum runtime between 2 sleep mode intervals.
`rFlowValPump2OnLim`	REAL	Defines the flow value to increase the demand of number of pumps to 2.
`rFlowValPump2OffLim`	REAL	Defines the flow value to decrease the demand of number of pumps from 2 to 1.
`rFlowValPump3OnLim`	REAL	Defines the flow value to increase the demand of number of pumps to 3.
`rFlowValPump3OffLim`	REAL	Defines the flow value to decrease the demand of number of pumps from 3 to 2.
`rFlowValPump4OnLim`	REAL	Defines the flow value to increase the demand of number of pumps to 4.
`rFlowValPump4OffLim`	REAL	Defines the flow value to decrease the demand of number of pumps from 4 to 3.
`rFlowValPump5OnLim`	REAL	Defines the flow value to increase the demand of number of pumps to 5.
`rFlowValPump5OffLim`	REAL	Defines the flow value to decrease the demand of number of pumps from 5 to 4.
`rFlowValPump6OnLim`	REAL	Defines the flow value to increase the demand of number of pumps to 6.
`rFlowValPump6OffLim`	REAL	Defines the flow value to decrease the demand of number of pumps from 6 to 5.
`rFlowValPump7OnLim`	REAL	Defines the flow value to increase the demand of number of pumps to 7.
`rFlowValPump7OffLim`	REAL	Defines the flow value to decrease the demand of number of pumps from 7 to 6.

Element of `stFlowStagInit`	Data Type	Description
`rPresIncPump1SwcOff`	REAL	Defines the pressure that has to be available to switch off pump 1. This value consists of the pressure reference value (`i_rSp`) plus an additional incremental value to reach a pressure level that corresponds to the average value of the working range of the auxiliary pump.
`rCtrlValPump1OnLim`	REAL	Defines the minimum limit of the PID control value that has to be available to switch on pump 1. At least 1 pump has to run to generate a flow value. Only then you can use the stage function based on flow.
`tDelTimeSwcOn`	TIME	Defines the delay time in milliseconds that has to expire before the first pump is switched on.
`tDelTimeSwcOFF`	TIME	Defines a minimum time in milliseconds the first pump has to be running before it can be switched off again. It is the minimum runtime between 2 sleep mode intervals.
`rFlowValPump2OnLim`	REAL	Defines the flow value to increase the demand of number of pumps to 2.
`rFlowValPump2OffLim`	REAL	Defines the flow value to decrease the demand of number of pumps from 2 to 1.
`rFlowValPump3OnLim`	REAL	Defines the flow value to increase the demand of number of pumps to 3.
`rFlowValPump3OffLim`	REAL	Defines the flow value to decrease the demand of number of pumps from 3 to 2.
`rFlowValPump4OnLim`	REAL	Defines the flow value to increase the demand of number of pumps to 4.
`rFlowValPump4OffLim`	REAL	Defines the flow value to decrease the demand of number of pumps from 4 to 3.
`rFlowValPump5OnLim`	REAL	Defines the flow value to increase the demand of number of pumps to 5.
`rFlowValPump5OffLim`	REAL	Defines the flow value to decrease the demand of number of pumps from 5 to 4.
`rFlowValPump6OnLim`	REAL	Defines the flow value to increase the demand of number of pumps to 6.
`rFlowValPump6OffLim`	REAL	Defines the flow value to decrease the demand of number of pumps from 6 to 5.
`rFlowValPump7OnLim`	REAL	Defines the flow value to increase the demand of number of pumps to 7.
`rFlowValPump7OffLim`	REAL	Defines the flow value to decrease the demand of number of pumps from 7 to 6.

Element of stFlowStagInit	Data Type	Description
rFlowValPump8OnLim	REAL	Defines the flow value to increase the demand of number of pumps to 8.
rFlowValPump8OffLim	REAL	Defines the flow value to decrease the demand of number of pumps from 8 to 7.

NOTE: The elements of the instance of the structure data type are set to the default values (0 or FALSE) assigned by the controller on system startup.

Structure: stPidFbck

The stPidFbck structure data type is used to indicate the values of the different components of the PID function on the HMI:

Element of stPidFbck	Data Type	Description
rP_Part	REAL	Indicates the value of the proportional part (P).
rI_Part	REAL	Indicates the value of the integral part (I).
rD_Part	REAL	Indicates the value of the differential part (D).
xI_Part_Actv	BOOL	When TRUE, indicates that the integral part (I) is active.
xD_Part_Actv	BOOL	When TRUE, indicates that the differential part (D) is active.

Structure: stPidSta

The stPidSta structure data type is used to indicate the status of the PID function on the HMI:

Element of stPidSta	Data Type	Description
xErrLimActv	BOOL	When TRUE, indicates that the maximum increment value of the I part of the PID control value calculation is exceeded (defined with the parameter stPidInit.rMaxErrForInt) and that the limitation of the increment value from the I part of the PID control value is active.
xAntiWndoActv	BOOL	When TRUE, indicates that the anti-windup process of the I part is active. TRUE = I part – P part > 100%
xLowLimActv	BOOL	When TRUE, indicates that the calculation of the I part of the calculation of the PID control value is less than the minimum output limit for the PID control value (defined with the parameter stPidInit.RLowLim). Then the limitation of the calculation of the I part of the calculation of the PID control value is active.

Element of `stPidSta`	Data Type	Description
`xCtrlValHighLimActv`	BOOL	When TRUE, indicates that the high limit for the PID control value (defined with the parameter `rHighLim`) is exceeded and the limitation of the PID control value is active.
`xCtrlValLowLimActv`	BOOL	When TRUE, indicates that the low limit for the PID control value (defined with the parameter `rLowLim`) is undershot and the limitation of the PID control value is active.
`xResActv`	BOOL	Indicates that the PID function has executed a reset.
`xHoldActv`	BOOL	Indicates that the PID function is in stopped mode.
`xStopActv`	BOOL	Indicates that the PID function has been stopped.
`xManActv`	BOOL	Indicates that the PID function is operated in manual mode, allowing to enter a setpoint via the HMI.
`xAutoActv`	BOOL	Indicates that the PID function is operated in automatic mode.
`xDBndActv`	BOOL	Indicates that a deviation has been suppressed. The PID control value is unchanged.
`xErrActv`	BOOL	Indicates that an error has been detected in the PID function.
`xParaChg`	BOOL	Indicates that the parameter values will be mapped in an internal memory, if the verification of the parameter was successful (reaction of the input `i_xParaChg`).

Structure: stPidStagSta

The stPidStagSta structure data type is used to indicate the status of the stage function on the HMI:

Element of stPidStagSta	Data Type	Description
esiStag_Mode	enumeration esiStag	Indicates whether a stage function is active and which method is selected: ● esiStag_Mode = esiStag.Off or 0: stage function is deactivated ● esiStag_Mode = esiStag.Stage or 1: stage method by using only the pressure to generate the stage value is active ● esiStag_Mode = esiStag.Flow or 2: stage method by using additionally the flow value is active
xEmgyStagMode	BOOL	Indicates that the stage method based on flow is not executable and the stage method based on pressure is used instead.
xSlpMode	BOOL	Indicates that the booster pumping application is in sleep mode.

Structure: stPidErr

The stPidErr structure data type is used to indicate alarms or alerts.

Element of stPidErr	Data Type	Description
stPidAlrm	stPidAlrm	Use of the structure data type for detected alarms.
stPidAlrt	stPidAlrt	Use of the structure data type for detected alerts.

Structure: stPidAlrm

The stPidAlrm structure data type is used to indicate an alarm detected in the PID function:

Element of stPidAlrm	Data Type	Description
xInvdLimPara	BOOL	When TRUE, an invalid limit parameter has been detected.
xInvdDeadBandPara	BOOL	When TRUE, an invalid dead band limit parameter has been detected.
xKpPara	BOOL	When TRUE, an invalid Kp parameter has been detected.
xTnPara	BOOL	When TRUE, an invalid Tn parameter has been detected.
xTvPara	BOOL	When TRUE, an invalid Tv parameter has been detected.

Element of stPidAlrm	Data Type	Description
xMaxErrForIntPara	BOOL	When TRUE, an invalid maximum limit for the calculation of the I part has been detected.

Structure: stPidAlrt

The stPidAlrt structure data type is used to indicate a detected alert in the PID function:

Element of stPidAlrt	Data Type	Description
xNoRel	BOOL	When TRUE, the PID function is disabled.
xTdPara	BOOL	When TRUE, an invalid Td parameter has been detected.

Structure: stPidStagErr

The stPidStagErr structure data type is used to indicate alarms or alerts detected in the stage function.

Element of stPidStagErr	Data Type	Description
stPidStagAlrm	stPidStagAlrm	Use of the structure data type for alarms detected in the stage function.
stPidStagAlrt	stPidStagAlrt	Use of the structure data type for alerts detected in the stage function.

Structure: stPidStagAlrm

The stPidStagAlrm structure data type is used to indicate a detected alarm in the stage function:

Element of stPidStagAlrm	Data Type	Description
xFlowDataCheckAlrm	BOOL	When TRUE, the flow value is no longer valid.
xReserve	BOOL	This parameter is not used.

Structure: stPidStagAlrt

The stPidStagAlrt structure data type is used to indicate a detected alert in the stage function:

Element of stPidStagAlrt	Data Type	Description
xAltValActv	BOOL	When TRUE, the alternative pressure value provided by a second pressure value - if available in your system - is used.
xReserveII	BOOL	This parameter is not used.

Section 1.2

DevSwcPumpCtrl: Device Switching Pump Control

Overview

This section describes the DevSwcPumpCtrl function block for switching management of variable speed drives (VSD), variable speed pumps (VS pumps), and fixed speed pumps (FS pumps). The function supports up to 8 pumps. These 8 pumps can be any mix of VS pumps or FS pumps assuming at least one pump is a VS pump.

NOTE: This function block is designed to be used in conjunction with other function blocks of this library. The function block architecture helps simplify programming the booster pumping functionality.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description - DevSwcPumpCtrl	40
Function Block Description - DevSwcPumpCtrl	48
Structure Data Type Definitions - DevSwcPumpCtrl	52
Configuration Examples - DevSwcPumpCtrl	54

Description - DevSwcPumpCtrl

Overview

The DevSwcPumpCtrl function block switches the variable speed drives (VSD), variable speed (VS) pumps and fixed speed (FS) pumps according to different criteria in order to maintain a constant pressure in a water distribution network.

It executes the following main tasks:

- Switching a maximum of 4 VSDs into run or stop state in flexible operational mode.
- Switching a maximum of 8 VSDs into run or stop state in fixed link operational mode (with or without bypass).
- Establishing a permanent or flexible link between VSDs and VS pumps.
- Starting or stopping a maximum of 8 VS pumps and having them bypass the VSD, in case the VSD is unavailable and no FS pump is available as an alternative.
- Starting or stopping a maximum of 7 FS pumps.
- Verifying the availability of VSDs and VS pumps. In case there are not enough VSD/VS pumps for the flow staging method; it initiates the PumpPidStag function block to stop using the flow value as a basis for stage value calculation *(see page 21)* and to use only the pressure value *(see page 19)*.

The DevSwcPumpCtrl function block considers the following criteria for switching the VSDs, VS pumps, and FS pumps:

- availability of the devices (VSD *(see page 106)*, VS pumps, FS pumps *(see page 102)*)
- demand of the stage value *(see page 18)*
- priority of the devices (VSD, VS pumps, FS pumps *(see page 110)*)

Prerequisite

The DevSwcPumpCtrl function block executes its tasks only if the following conditions apply:

- The pumping application is running in automatic mode.
  The automatic mode is established by the structure data type stOpMode being set to xAuto (for more information refer to the chapter Common Structure Data Type Definitions *(see page 166)*).
  With another setting of this variable, the DevSwcPumpCtrl function block is disabled and the function block outputs are set to the default value (0 and FALSE).
- The alarm release bit xAlrmRele of the structure data type stAlrmCtrl *(see page 166)* is set to TRUE, indicating that no alarm has been detected in the pump group.
  If the alarm release bit xAlrmRele is set to FALSE, the DevSwcPumpCtrl function block is disabled and the function block outputs are set to the default value (0 and FALSE).

Error Detection

According to the type of error the `DevSwcPumpCtrl` function block detects, it either issues an alarm or an alert. The actions that are executed differ for detected alarms and alerts.

Actions when an alarm *(see page 53)* has been detected in the pump group:

If	Then
an alarm has been detected in the pump group	the `DevSwcPumpCtrl` function block sets the corresponding alarm bit (`q_stSwcErr`) to TRUE indicating which type of alarm has been detected.
	the variable `stOpMode` is set to manual mode (`xMan`).
	the `DevSwcPumpCtrl` function block stops processing variables.
	the outputs of the `DevSwcPumpCtrl` function block are set to the default value (0 and FALSE).
	the alarm release bit `iq_stAlrmCtrl.xAlrmRele` is set to FALSE to indicate the detected alarm state to the other function blocks.

Actions when an alert has been detected in the pump group:

If	Then
an alert has been detected in the pump group	the `DevSwcPumpCtrl` function block sets the corresponding alert bit (`q_stSwcErr.stSwcAlrt.xNoVsPlusFs`) to TRUE indicating that the stage value is greater than the number of usable pumps.
	the function block continues operation.

Switching VSDs or Pumps According to the Availability of the Devices

If the pumping application is running in automatic mode, the `DevSwcPumpCtrl` function block verifies the state of the running VSDs, VS pumps, and FS pumps.

If one of these devices becomes unavailable, the `DevSwcPumpCtrl` function block starts another device of the same type, if available, or of a different type according to the following general rules:

If	Then
a VSD becomes unavailable and there is no other VSD available for replacement	an FS pump is used alternatively. If no FS pump is available, the VS pump that had been linked to the VSD can bypass the VSD (if the bypass option is enabled). If more than 1 VS pump is available, the function runs the VS pump contingent on priority conditions *(see page 110)*. The link between the VSD and the VS pump is maintained as long as the VSD and the VS pump are operational.
a VS pump becomes unavailable and there is no other VS pump for replacement	an FS pump is used alternatively and the VSD that had been linked to the VS pump is switched into stop state.

Switching VSDs or Pumps According to the Stage Value

If the pumping application is running in automatic mode, the `DevSwcPumpCtrl` function block continuously receives input from the `PumpPidStag` function block *(see page 25)* via the `i_stStagVal` input value *(see page 18)*. This input can initiate a switching procedure. Depending on the stage value from the `PumpPidStag` function block, it may be required to switch VSDs, VS pumps or FS pumps on or off considering the general rule (switching on VSDs and VS pumps first, then FS pumps) as shown in the following flowcharts.

Actions when the stage value increases and the `xFlexStart` flag is set to FALSE:

NOTE: The processes to start a VSD, a VS pump, or an FS pump, consider also the priority defined with the `OpPrty` function block *(see page 112)*.

Actions when the stage value increases and the `xFlexStart` flag is set to TRUE:

NOTE: The processes to start a VSD or a VS pump, consider also the priority defined with the `OpPrty` function block *(see page 112)*.

Actions when the stage value decreases:

NOTE: The processes to stop a VSD, a VS pump, or an FS pump, consider also the priority defined with the `OpPrty` function block *(see page 112)*.

NOTE: The stage value can only change sequentially (maximal one step up or down). However, if a VSD, a VS pump or an FS pump simultaneously is no longer available, more than one device can be switched at a time.

Switching VSDs or Pumps According to the Priority

In general, FS pumps are the last devices to be switched on and the first devices to be switched off. This means VSDs and the corresponding VS pumps are the last devices to be switched-off and the first devices to be switched-on. The priority is given to VS pumps over FS pumps. The priority in a device group (VSD, VS pump, FS pump) is flexible and generated by the `OpPrty` function block. This function block generates, with the help of the operating hours of the devices, a priority table for the `DevSwcPumpCtrl` function block. The priority table helps to keep the number of operating hours at an identical level for the different pumps.

The priority function of this library allows you to use another procedure of selecting the priority of the devices by using a user-specific defined priority table *(see page 110)*. If the `xFlexStart` (element of initialization structure `stSwcPumpInit`) is set to TRUE, the VS pumps are started with a VSD. If the VSDs are in use, an active VS pump linked to a VSD bypasses the VSD (direct online) and the now available VSD is used to start the needed VS pump.

The `DevSwcPumpCtrl` function block considers this dynamically changing priority via the outputs from the `OpPrty` function blocks *(see page 112)* for each device (VSD, VS pumps and FS pumps). This means that the devices are switched according to their priority defined in the priority table. If a device of the same type is not available, an alternative device is switched on, if possible.

Function Block Description - `DevSwcPumpCtrl`

Pin Diagram

Brief Description

The `DevSwcPumpCtrl` function block has the following functions:

1.	Verifying the operating mode. This function block only executes its tasks if the pumping application is running in automatic mode. In any other mode, the outputs are constantly set to their default values (0 / FALSE).
2.	Constantly verifying the availability of the used VSDs, VS pumps, and FS pumps. If one of these devices becomes unavailable, the function block switches on another device.
3.	Constantly verifying the stage value and switching VSDs, VS pumps, FS pumps if the stage value changes *(see page 43)*. ● If the stage value increases, switching a VSD, VS pump, or FS pump to run state considering the general rules and the device priority defined and generated by the `OpPrty` function block *(see page 112)*. If no alternative device is available, no device is switched. ● If the stage value decreases, switching a pump to stop state considering the general rules and the device priority defined and generated by the `OpPrty` function block *(see page 112)*.
4.	In automatic mode, stopping if an error is detected and switching to manual mode.

5.	Uses the available devices with the help of the general rules and the priority defined and generated by the `OpPrty` function block.
6.	Verifying whether it is possible to switch to flexible operational mode *(see page 57)* by evaluating the following questions: ● Is no FS pump installed? ● Is the number of VS pumps greater than the number of VSDs? ● Is the bypass option enabled (`xByPass` = TRUE)? ● Is the fixed link operational mode *(see page 59)* disabled (`xFixLink` = FALSE)? The flexible operational mode is the mode with the greatest availability of pumps because the VS pumps can be used independently of VSDs.

I/O Variables Description

The table describes the input/output variables of the function block:

Input/Output	Data Type	Description
`iq_stAlrmCtrl`	`stAlrmCtrl`	Structure used for alarm handling. Refer to the structure data type description *(see page 166)*.
`iq_stOpMode`	`stOpMode`	Determines the operating modes. If the element `xAuto` of the structure data type `stOpMode` is set to TRUE, the function block executes its tasks automatically. With another status of this element, the outputs of this function block are set to their defined fallback (default) states. Refer to the structure data type description *(see page 166)*.
`iq_stSwcInit`	`stSwcInit`	Structure used for initialization data. Refer to the structure data type description *(see page 52)*.

The table describes the input variables to the function block:

Input	Data Type	Description
`i_xRele`	BOOL	Activates (TRUE) or stops (FALSE) the execution of the function block.
`i_astVsdSta_Ex`	`astVsdSta`	The array of `stVsdSta` indicates external VSDs status. Refer to the structure data type description *(see page 171)*.
`i_stAvai_Vsd`	`stAvai`	Structure indicating 8 available states provided by the `VsdAvai` function block *(see page 107)* indicating whether the VSDs are available (maximum of 4). Refer to the structure data type description *(see page 168)*.

Input	Data Type	Description
i_stAvai_VsPump	stAvai	Structure indicating 8 available states provided by the PumpAvai function block *(see page 103)* indicating whether the VS pumps are available (maximum of 8). Refer to the structure data type description *(see page 168)*.
i_stAvai_FsPump	stAvai	Structure indicating 8 available states provided by the PumpAvai function block *(see page 103)* indicating whether the FS pumps are available (maximum of 8). Refer to the structure data type description *(see page 168)*.
i_asiPrty_Vsd	ARRAY [0..7] of SINT	Priority value table for the VSDs provided by the OpPrty function block *(see page 112)*.
i_asiPrty_VsPump	ARRAY [0..7] of SINT)	Priority value table for the VS pumps provided by the OpPrty function block *(see page 112)*.
i_asiPrty_FsPump	ARRAY [0..7] of SINT	Priority value table for the FS pumps provided by the OpPrty function block *(see page 112)*.
i_stStagVal	stStagVal	Structure of the stage value provided by the PumpPidStag function block *(see page 25)*. It indicates the number of pumps that is needed to generate sufficient power to maintain the user-defined pressure level. Refer to the structure data type description *(see page 169)*.

The table describes the output variables from the function block:

Output	Data Type	Description
q_stCmd_Vsd	stCmd	Structure used for switching the VSDs (maximum of 8) into run state. Refer to the structure data type description *(see page 169)*.
q_astLink_VsPump	ARRAY [0..7] of stLink	Output array of stLink to link the VS pumps (maximum of 8) dynamically or constantly to one of the VSDs or directly to the power source (direct online) by using contactors. Refer to the structure data type description *(see page 172)*.
q_stCmd_FsPump	stCmd	Structure used for switching the FS pumps (maximum of 8) into run state. Refer to the structure data type description *(see page 169)*.
q_siNoRunPump_Hmi	SINT	Contains the number of pumps that are in run state.

Output	Data Type	Description
`q_stSwcErr`	`stSwcErr`	Structure used for states of detected alarms and alerts. Refer to the structure data type description for alarms and alerts *(see page 166)*.
`q_stSwcOffInProg`	`stSwcOffInProg`	Structure to contain the switch-off-process (DCC) in progress message of up to 8 VSDs.
`q_xFlowStagEn`	BOOL	When TRUE, at least 2 VSD with a corresponding VS pump is available. If no VSD / VS pump is available, this output is set to FALSE. As a result, the `PumpPidStag` function block stops using the flow value as a basis for stage value calculation *(see page 21)*. It uses the pressure value only *(see page 19)*.

Structure Data Type Definitions - DevSwcPumpCtrl

Overview

Structure data types serve as the major input and output data carriers between different function blocks, configuration, and parameterization data.

The following section lists the names of structures (data types) that are used in this function block, followed by a table that describes the elements of structure.

NOTE: For the structure data types that are used from this function block and others, refer to the Common Structure Data Type Definitions chapter *(see page 166)*.

Structure: stSwcInit

The stSwcInit structure data type contains the initialization data for the DevSwcPumpCtrl function block (refer to the examples *(see page 54)*).

Element of stSwcInit	Data Type	Description
siNuVsdMax	SINT	Number of VSDs installed. Range: 0...8
siNuVsPumpMax	SINT	Number of VS pumps installed. Range: 0...8
siNuFsPumpMax	SINT	Number of FS pumps installed. Range: 0...8
siNuVsd	SINT	Maximum number of VSDs that can be used simultaneously. Range: 0...8
siNuVsPump	SINT	Maximum number of VS pumps that can be used simultaneously. Range: 0...8
siNuFsPump	SINT	Maximum number of FS pumps that can be used simultaneously. Range: 0...8
xFlexStart	BOOL	When TRUE, flexible start is used.
xFixLink	BOOL	When TRUE, the links between the VSDs and the VS pumps are fixed.
xByPass	BOOL	When TRUE, the VS pumps can bypass the VSDs.

NOTE: The elements of the structure data type are set to the default values (0 or FALSE) assigned by the controller on system startup.

Structure: stSwcErr

The stSwcErr structure data type is used to indicate the detected alarms or alerts.

Element of stSwcErr	Data Type	Description
stSwcAlrm	stSwcAlrm	The structure data type for detected alarms.
stSwcAlrt	stSwcAlrt	The structure data type for detected alerts.

Structure: stSwcAlrm

The stSwcAlrm structure data type is used to indicate the detected alarm states.

Element of stSwcAlrm	Data Type	Description
xLinkErr	BOOL	When TRUE, it has been detected that the VSD and the VS pump are not in the same state of operation (run or stop).
xStagErr	BOOL	When TRUE, it has been detected that the stage value is not valid because it does not comply with the number of pumps.
xParaErr	BOOL	When TRUE, an error in the configuration has been detected.

Structure: stSwcAlrt

The stSwcAlrt structure data type is substructure of stSwcErr. It is used to indicate the detected alert states.

Element of stSwcAlrt	Data Type	Description
xNoVsPlusFs	BOOL	When TRUE, a configuration error has been detected: The stage value demands a higher number of pumps than the maximum number of pumps (FS pumps and VS pumps) configured for simultaneous use.

Configuration Examples - DevSwcPumpCtrl

Overview

You can configure the DevSwcPumpCtrl function block to run in different operational modes.

The following modes are available:

Operational mode	Description
Normal mode	The DevSwcPumpCtrl function block can switch variable speed drives (VSDs), variable speed pumps (VS pumps), and fixed speed pumps (FS pumps) by establishing flexible links between VSDs and VS pumps. During normal operation, each type of pump runs in the defined operational mode. This means that VS pumps are operated in conjunction with a VSD. This operational mode is available if the following conditions apply: ● the flexible start FlexStart *(see page 43)* is disabled ● the fixed link operational mode is disabled ● the bypass option is enabled.
flexible mode	The VS pumps can be used independently of VSDs. This improves the availability of the machine. This operational mode is available if the following conditions apply: ● no FS pumps are configured ● the number of available VS pumps is greater than the number of VSDs ● the bypass option is enabled ● the fixed link operational mode is disabled If the flexible operational mode is available, the flexible start FlexStart *(see page 43)* can be used.
fixed link mode	Each VSD is physically connected to 1 VS pump. In this operational mode, you can enable or disable the bypass option with the xByPass parameter *(see page 52)*. This operational mode is available if the following conditions apply: ● the flexible start FlexStart *(see page 43)* is disabled ● the fixed link operational mode is enabled.

Bypass Option

It is possible to disconnect the VS pumps from the VSD and operate the pump directly with an independent contactor. With this bypass option, the VS pumps are connected to the power source (direct online) for the case a VSD becomes unavailable.

NOTE: In flexible operational mode, the bypass option is enabled. In fixed link operational mode, you can enable or disable the bypass option with the xByPass parameter *(see page 52)*.

Configuration Examples

This section provides configuration examples for each mode. These examples and their values only apply to one pump group for one pressure level.

- normal mode with redundant VS pump *(see page 56)*
- flexible operational mode *(see page 57)*
- fixed link operational mode
  - with bypass option enabled *(see page 59)*
  - with bypass option disabled *(see page 61)*

To configure the `DevSwcPumpCtrl` function block for one of these modes, set the initialization data with the structure data type `stSwcInit` as listed in the following examples.

NOTE: The following examples are for information only and do not reflect a specific installation.

Example 1: Normal Mode with Redundant VS Pump

This example demonstrates a pumping application in normal mode with 1 redundant VS pump. There are 3 VS pumps installed, however only 2 VS pumps can run simultaneously.

In this example:

- 3 VS pumps (1 of which is redundant)
- 3 FS pumps
- 2 VSDs

NOTE: Three-phase power lines are drawn as a single line for reasons of simplicity.

1 power supply
2 power switch
3 VSD (variable speed drives)
4 motor switches
5 contactors
6 VS (variable speed) pumps
7 FS (fixed speed) pumps

To configure the `DevSwcPumpCtrl` function block for normal mode, set the parameters of the instance of the `stSwcInit` structure data type as follows:

Parameter	Value	Description
`siNuVsdMax`	2	Number of VSDs installed.
`siNuVsPumpMax`	3	Number of VS pumps installed.
`siNuFsPumpMax`	3	Number of FS pumps installed.
`siNuVsd`	2	Maximum number of VSDs that can run simultaneously.
`siNuVsPump`	2	Maximum number of VS pumps that can run simultaneously.
`siNuFsPump`	3	Maximum number of FS pumps that can run simultaneously.
`xFlexStart`	FALSE (0)	Flexible start is disabled.
`xFixLink`	FALSE (0)	There are no fixed links between the VSDs and the VS pumps.
`xByPass`	TRUE (1)	It is allowed to operate the VS pumps bypassing the VSD.

Example 2: Flexible Operational Mode

This example shows a pumping application in flexible operational mode.

This mode is available if the following conditions apply:
- no FS pumps configured
- the number of available VS pumps is greater than the number of VSDs
- the bypass option is enabled
- the fixed link operational mode is disabled

The flexible operational mode is the mode with the greatest availability of pumps because the VS pumps can be used independently of VSDs.

This architecture is possible to run in flexible start. If the `xFlexStart` (element of initialization structure `stSwcPumpInit`) is set to TRUE the VS pumps are started with a VSD. If the VSDs are in use, an active VS pump linked to a VSD bypasses the VSD (direct online) and the now available VSD is used to start the needed VS pump.

In this example:
- 8 VS pumps
- 2 VSDs

NOTE: Three-phase power lines are drawn as a single line for reasons of simplicity.

1 power supply
2 power switch
3 VSD (variable speed drives)
4 motor switches
5 contactors
6 VS (variable speed) pumps

To configure the `DevSwcPumpCtrl` function block for flexible operational mode, set the parameters of the instance of the `stSwcInit` structure data type as follows:

Parameter	Value	Description
`siNuVsdMax`	2	Number of VSDs installed.
`siNuVsPumpMax`	8	Number of VS pumps installed.
`siNuFsPumpMax`	0	Number of FS pumps installed.
`siNuVsd`	2	Maximum number of VSDs that can run simultaneously.
`siNuVsPump`	8	Maximum number of VS pumps that can run simultaneously.
`siNuFsPump`	0	Maximum number of FS pumps that can run simultaneously.

Parameter	Value	Description
`xFlexStart`	TRUE (1)	Flexible start is used.
`xFixLink`	FALSE (0)	There are no fixed links between the VSDs and the VS pumps.
`xByPass`	TRUE (1)	It is allowed to operate the VS pumps bypassing the VSD.

Example 3: Fixed Link Operational Mode With Bypass Option Enabled

This example shows a pumping application in fixed link operational mode with the bypass option enabled. This means that each VSD is constantly linked to 1 VS pump but if a VSD becomes unavailable, the VS pump can be directly connected to the power source (direct online).

In this example:

- 2 VS pumps
- 2 FS pumps
- 2 VSDs

NOTE: Three-phase power lines are drawn as a single line for reasons of simplicity.

1 power supply
2 power switch
3 VSD (variable speed drives)
4 motor switches
5 contactors
6 VS (variable speed) pumps
7 FS (fixed speed) pumps

To configure the `DevSwcPumpCtrl` function block for fixed link operational mode with the bypass option enabled, set the parameters of the instance of the `stSwcInit` structure data type as follows:

Parameter	Value	Description
`siNuVsdMax`	2	Number of VSDs installed.
`siNuVsPumpMax`	2	Number of VS pumps installed.
`siNuFsPumpMax`	2	Number of FS pumps installed.

Parameter	Value	Description
siNuVsd	2	Maximum number of VSDs that can run simultaneously.
siNuVsPump	2	Maximum number of VS pumps that can run simultaneously.
siNuFsPump	2	Maximum number of FS pumps that can run simultaneously.
xFlexStart	FALSE (0)	Flexible start is disabled.
xFixLink	TRUE (1)	The links between the VSDs and the VS pumps are fixed.
xByPass	TRUE (1)	It is allowed to operate the VS pumps bypassing the VSD.

Example 4: Fixed Link Operational Mode With Bypass Option Disabled

This example shows a pumping application in fixed link operational mode with the bypass option disabled. This means that each VSD is constantly linked to 1 VS pump. Since the bypass option is disabled, a VS pump becomes unavailable if the corresponding VSD becomes unavailable.

In this example:

- 2 VS pumps
- 2 FS pumps
- 2 VSD

NOTE: Three-phase power lines are drawn as a single line for reasons of simplicity.

1 power supply
2 power switch
3 VSD (variable speed drives)
4 motor switches
5 contactors
6 VS (variable speed) pumps
7 FS (fixed speed) pumps

To configure the `DevSwcPumpCtrl` function block for fixed link operational mode with the bypass option disabled, set the parameters of the instance of the `stSwcInit` structure data type as follows:

Parameter	Value	Description
`siNuVsdMax`	2	Number of VSDs installed.
`siNuVsPumpMax`	2	Number of VS pumps installed.
`siNuFsPumpMax`	2	Number of FS pumps installed.
`siNuVsd`	2	Maximum number of VSDs that can run simultaneously.
`siNuVsPump`	2	Maximum number of VS pumps that can run simultaneously.
`siNuFsPump`	2	Maximum number of FS pumps that can run simultaneously.
`xFlexStart`	FALSE (0)	Flexible start is disabled.
`xFixLink`	TRUE (1)	The links between the VSDs and the VS pumps are fixed.
`xByPass`	FALSE (0)	It is not allowed to operate the VS pumps bypassing the VSD.

Section 1.3
VsdCtrl: VSD Control

Overview

This section describes the VsdCtrl function block for controlling and switching a maximum of 8 variable speed drives (VSDs).

NOTE: This function block is designed to be used in conjunction with other function blocks of the Pumping Library.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description - VsdCtrl	65
Function Block Description - VsdCtrl	66

Description - `VsdCtrl`

Overview

The `VsdCtrl` function block executes the following tasks:

- Providing an operating hour counter for the VSDs. This counter is realized by an input/output variable. This allows you to reset or adjust the variable even from HMI or SCADA (supervisory control and data acquisition) systems.
- Detecting alarm states of the VSDs by processing the inputs of external key switches, sensors, and contactors connected to the VSDs.
- Indicating a detected alarm for a VSD if a command has been sent to the VSD and the configured response time has expired without receiving a response.
- Before the switching commands for the VSDs are executed, this function block verifies whether the respective VSD is not in a detected alarm state and may be switched. If an error is detected, the VSD command is set to FALSE (0).

This function is independent of the type of connection to the VSD (fieldbus or hardwired).

The function does not consider inputs that are not used (unresolved inputs are declared inactive).

The inputs to this function block concerning the status of the VSD are only active if a variable is linked to them.

Function Block Description - `VsdCtrl`

Pin Diagram

Brief Description

The `VsdCtrl` function block executes the following tasks:

- Providing operating hour counters for the VSDs.
- Indicating a detected alarm for a VSD if the configured response time has expired.
- Verifying whether the respective VSDs are available before the switch command is mapped from the input to the output. Otherwise, the output (command bit) of the unavailable VSD is set to FALSE (stop).
- Generating of an alarm flag for each VSD.

I/O Variables Description

The table describes the input/output variables of the function block:

Input/Output	Data Type	Description
`iq_stAlrmCtrl`	`stAlrmCtrl`	Structure used for alarm handling. Refer to the structure data type description *(see page 166)*.
`iq_astMode_Vsd`	ARRAY [0..7] of `stMode`	Array of `stMode` to control and switch the operating mode of the VSDs (maximum of 8). Refer to the structure data type description *(see page 169)*.
`iq_audiOpHr`	ARRAY [0..7] of UDINT	This array is a table listing the operating hours of the VSDs (maximum of 8). This input/output variable can be reset or adjusted, including from HMI or SCADA systems.

The table describes the input variables to the function block:

Input	Data Type	Description
i_xRes	BOOL	Resets alarms detected in the VSDs if the cause has been corrected.
i_astVsdSta_Ex	ARRAY [0..7] of stVsdSta	Array of stVsdSta. Contains the external status information from up to 8 VSDs. Refer to the structure data type description *(see page 171)*.
i_stCmd_Vsd	stCmd	Structure for switching the VSDs into run state (maximum of 8). Refer to the structure data type description *(see page 169)*.
i_siVsdMax	SINT	Contains the number of installed VSDs.
i_tDelFbck	TIME	Defines the maximum time (in milliseconds) allowed for the VSD to send a response. Default: i_tDelFbck = T#1 s As soon as the command is not equal to the response message received from the VSD, a timer is started. If the time defined with this parameter expires without receiving a response equal to the command, then the VsdCtrl function block sets the output q_stRespErr.xDev(n) for the respective VSD to TRUE to indicate that an error has been detected *(see page 175)*.

The table describes the output variables from the function block:

Output	Data Type	Description
q_stRespErr	stRespErr	Structure that indicates detected errors in the pumps (maximum of 8). Refer to the structure data type description *(see page 169)*.
q_stCmd_Vsd_Ex	ARRAY [0..7] of stCmd	Structure used for switching the VSDs into run state (maximum of 8). Refer to the structure data type description *(see page 169)*.

Section 1.4
VsPumpCtrl: Variable Speed Pump Control

Overview

This section describes the VsPumpCtrl function block for controlling and switching a maximum of 8 VS pumps.

NOTE: This function block is designed to be used in conjunction with other function blocks of the Pumping Library.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description - VsPumpCtrl	69
Function Block Description - VsPumpCtrl	70

Description - `VsPumpCtrl`

Overview

The `VsPumpCtrl` function block executes the following tasks:

- Providing an operating hour counter for the VS pumps. This counter is realized by an input/output variable. This allows you to reset or adjust the variable even from HMI or SCADA systems.
- Detecting alarm states of the VS pumps by processing the inputs of external key switches, sensors, and contactors connected to the VS pumps.
- Indicating a detected alarm for a VS pump if a command has been sent to the VS pump and the configured response time has expired without receiving a response.
- Before the switching commands for the VS pumps are executed, this function block verifies whether the respective VS pump is not in a detected alarm state and may be switched. If an error is detected, the pump command is set to FALSE (0).

This function is independent of the type of connection to the VS pump (fieldbus or hardwired).

The function does not consider inputs that are not used (unresolved inputs are declared inactive).

The inputs to this function block concerning the status of the VS pump are only active if a variable is linked to them.

Function Block Description - VsPumpCtrl

Pin Diagram

Brief Description

The `VsPumpCtrl` function block executes the following tasks:

- Providing an operating hour counter for the VS pumps.
- Indicating a detected alarm for a VS pump if the configured response time has expired.
- Verifying whether the respective VS pumps are available before the switch command is mapped from the input to the output. Otherwise the output (command bit) of the not available VS pump is set to FALSE (stop).
- Generating of an alarm flag for each VSD.

I/O Variables Description

The table describes the input/output variables of the function block:

Input/Output	Data Type	Description
`iq_stAlrmCtrl`	`stAlrmCtrl`	Structure used for alarm handling. Refer to the structure data type description *(see page 166)*.
`iq_astMode_VsPump`	ARRAY [0..7] of `stMode`	Array of `stMode` to control and switch the operating mode of the VS pumps (maximum of 8). Refer to the structure data type description *(see page 169)*.
`iq_audiOpHr`	ARRAY [0..7] of UDINT	This array is a table listing the operating hours of the VS pumps (maximum of 8). This input/output variable can be reset or adjusted, including from HMI or SCADA systems.

The table describes the input variables to the function block:

Input	Data Type	Description
i_xRes	BOOL	Resets detected VS pump alarm states, if the cause has been corrected.
i_astSta_VsPump_Ex	ARRAY [0..7] of stSta	Array of stSta. Contains the external status information from up to 8 VS pumps. Refer to the structure data type description *(see page 170)*.
i_astLink_VsPump	ARRAY [0..7] of stLink	Array of stLink to link the VS pumps (maximum of 8) dynamically or constantly to one of the VSDs or directly to the power source (direct online) by using contactors. Refer to the structure data type description *(see page 172)*.
i_siVsPumpMax	SINT	Contains the number of installed VS pumps.
i_tDelFbck	TIME	Defines the maximum time (in milliseconds) allowed for the VS pump to send a response. Default: i_tDelFbck = T#1 s As soon as the command is not equal to the response message received from the main contactor of the VS pump, a timer is started. If the time defined with this parameter expires without receiving a response equal to the command, then the VsPumpCtrl function block sets the output q_stRespErr.xDev(n) for the respective VS pump to TRUE to indicate that an error has been detected *(see page 175)*.

The table describes the output variables from the function block:

Output	Data Type	Description
q_stRespErr	stRespErr	Structure that indicates detected response errors in the pumps (maximum of 8). Refer to the structure data type description *(see page 175)*.
q_astLink_VsPump_Ex	ARRAY [0..7] of stLink	Array of stLink to link the VS pumps (maximum of 8) dynamically or constantly to one of the VSDs or directly to the power source (direct online) by using contactors. Refer to the structure data type description *(see page 172)*.
q_abyVsdNu_Hmi	ARRAY [0..7] of Byte	Output array of the used VSD numbers. Refer to the structure data type description *(see page 176)*.

Section 1.5
FsPumpCtrl: Fixed Speed Pump Control

Overview

This section describes the FsPumpCtrl function block for controlling and switching a maximum of 8 FS pumps.

NOTE: This function block is designed to be used in conjunction with other function blocks of the Pumping Library.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description - FsPumpCtrl	73
Function Block Description - FsPumpCtrl	74

Description - `FsPumpCtrl`

Overview

The `FsPumpCtrl` function block executes the following tasks:

- Providing an operating hour counter for the FS pumps. This counter is realized by an input/output variable. This allows you to reset or adjust the variable even from HMI or SCADA systems.
- Detecting alarm states of the FS pumps by processing the inputs of external key switches, sensors, and contactors connected to the FS pumps.
- Indicating a detected alarm for an FS pump if a command has been sent to the FS pump and the configured response time has expired without receiving a response.
- Before the switching commands for the FS pumps are executed, this function block verifies whether the respective FS pump is not in a detected alarm state and may be switched. If an error is detected, the pump command is set to FALSE (0).

This function is independent of the type of connection to the FS pump (fieldbus or hardwired).

The function does not consider inputs that are not used (unresolved inputs are declared inactive).

The inputs to this function block concerning the status of the FS pump are only active if a variable is linked to them.

Function Block Description - FsPumpCtrl

Pin Diagram

Brief Description

The `FsPumpCtrl` function block executes the following tasks:

- Providing an operating hour counter for the FS pumps.
- Indicating a detected alarm for an FS pump if the configured response time has expired.
- Verifying whether the respective FS pumps are available before the switch command is mapped from the input to the output. Otherwise, the output (command bit) of the not available FS pump is set to FALSE (stop).
- Generating of an alarm flag for each FS pump.

I/O Variables Description

The table describes the input/output variables of the function block:

Input/Output	Data Type	Description
`iq_stAlrmCtrl`	`stAlrmCtrl`	Structure used for alarm handling. Refer to the structure data type description *(see page 166)*.
`iq_astMode_FsPump`	ARRAY [0..7] of `stMode`	Array of `stMode` to control and switch the operating mode of the FS pumps (maximum of 8). Refer to the structure data type description *(see page 169)*.
`iq_audiOpHr`	ARRAY [0..7] of UDINT	This array is a table listing the operating hours of the FS pumps (maximum of 8). This input/output variable can be reset or adjusted, including from HMI or SCADA systems.

The table describes the input variables to the function block:

Input	Data Type	Description
i_xRes	BOOL	Resets detected FS pump alarm states, if the cause has been corrected.
i_astSta_FsPump_Ex	ARRAY [0..7] of stSta	Structure contains the external status information from 8 FS pumps. Refer to the structure data type description *(see page 170)*.
i_stCmd_FixPump	stCmd	Structure for switching the FS pumps into run state (maximum of 8). Refer to the structure data type description *(see page 169)*.
i_siFsPumpMax	SINT	Contains the number of installed FS pumps.
i_tDelFbck	TIME	Defines the maximum time (in milliseconds) allowed for the FS pump to send a response. Default: i_tDelFbck = T#1 s As soon as the command is not equal to the response message received from the main contactor of the FS pump, a timer is started. If the time defined with this parameter expires without receiving a response equal to the command, then the FsPumpCtrl function block sets the output q_stRespErr.xDev(n) for the respective FS pump to TRUE to indicate that an error has been detected *(see page 175)*.

The table describes the output variables from the function block:

Output	Data Type	Description
q_stRespErr	stRespErr	Structure that indicates detected response errors from the pumps (maximum of 8). Refer to the structure data type description *(see page 175)*.
q_stCmd_FixPump_Ex	stCmd	Structure used for switching the pumps into run state (maximum of 8). Refer to the structure data type description *(see page 169)*.

Section 1.6

CompSp: Setpoint Adaptation (Friction Loss Compensation)

Overview

This section describes the CompSp function block for friction loss compensation by adapting the pressure setpoint according to the stage value.

The following criteria are considered for recalculation of the setpoint:
- the number of pumps that is requested by the stage value
- a ramp function for setpoint integration

NOTE: This function block is designed to be used in conjunction with other function blocks of the Pumping Library.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description - CompSp	77
Function Block Description - CompSp	80
Structure Data Type Definitions - CompSp	82

Description - CompSp

Overview

In order to help to compensate for friction loss, the `CompSp` function block adapts the pressure setpoint (`i_rSp`) after a change of the stage value (`i_stStagVal)` provided by the `PumpPidStag` function block *(see page 25)* has been detected.

The following criteria are considered for recalculation of the pressure setpoint:

- the demand of the number of pumps (as indicated by the stage value) used as a pointer on a correction value (parameter) for the pressure setpoint
- a ramp function for the adaptation of the last used setpoint value to the newly adapted setpoint.

Increasing the Pressure Setpoint According to the Stage Value

The diagrams illustrate the relationship between the rising flow level (due to a rising stage value) and the increase of the pressure setpoint (`i_rSp`) to reduce friction loss:

Diagram 1

Diagram 2

Diagram 3

Diagram 4

Diagram 1: The diagram shows the dependence of the total number of active pumps to the flow value and in dependence on the time.

Diagram 2: The diagram shows the active pumps in dependence on the time.

Diagram 3: The diagram shows the increase of the setpoint value by a fixed correction value depending on the number of pumps used in dependence of the time.

Diagram 4: This diagram shows the variation of the setpoint adjustment as shown in diagram 3, but with a change in the setpoint corrections with the help of a ramp function.

The flow is increased with every pump that is switched to the run state. To increase the pressure accordingly, a user-defined adjustment value is added to the pressure setpoint (`i_rSp`). For the stage value of each pump, a specific adjustment value can be configured with the parameters `iq_stCompInit.rCorrPump1` to `iq_stCompInit.rCorrPump8`.

To help avoid an abrupt rise of pressure by adding the adjustment value to the setpoint pressure, a ramp time is defined with the parameter `iq_stCompInit.rTn`.

Function Block Description - CompSp

Pin Diagram

Brief Description

When the stage value changes, the `CompSp` function block helps to reduce friction loss by adapting the pressure setpoint considering the number of the pump demand. The steepness of the adaptation is limited by using a ramp function.

I/O Variables Description

The table describes the input/output variables of the function block:

Input/Output	Data Type	Description
`iq_stAlrmCtrl`	`stAlrmCtrl`	Structure used for alarm handling. Refer to the structure data type description *(see page 166)*.
`iq_stOpMode`	`stOpMode`	Determines the operating modes. If the element `xAuto` of the structure data type `stOpMode` is set to TRUE, the function block executes its tasks automatically. With another status of this element, the outputs of this function block are set to their defined fallback (default) states. Refer to the structure data type description *(see page 166)*.
`iq_stCompInit`	`stCompInit`	Structure used for initialization data for the compensation function. Refer to the structure data type description *(see page 82)*.

The table describes the input variables to the function block:

Input	Data Type	Description
`i_xRele`	BOOL	Activates (TRUE) or stops (FALSE) the execution of the function block.
`i_rSp`	REAL	Contains the user-defined pressure setpoint and can be adapted by this function, or the pipe fill, or the cavitation protection function.
`i_stStagVal`	`stStagVal`	Structure of the stage value provided by the `PumpPidStag` function block *(see page 25)*. It indicates the number of pumps that is needed to generate sufficient power to maintain the pressure level. The `CompSp` function block will increase the pressure setpoint (`i_rSp`) by the respective correction values (`iq_stCompInit.rCorrPump(n)`) for the number of pumps requested by the stage value. Refer to the structure data type description *(see page 169)*.

The table describes the output variable from the function block:

Output	Data Type	Description
`q_stRef_Sp`	`stRef`	Contains the adapted pressure setpoint and the state of this value. Refer to the structure data type description *(see page 175)*.

Structure Data Type Definitions - CompSp

Overview

Structure data types serve as the major input and output data carriers between different function blocks, configuration, and parameterization data.

The following section lists the names of structures (data types) that are used in this function block, followed by a table that describes the elements of structure.

NOTE: For the structure data types that are used from this function block and others, refer to the Common Structure Data Type Definitions chapter *(see page 166)*.

Structure: stCompInit

The stCompInit structure data type contains the initialization data for the CompSp function block.

Element of stCompInit	Data Type	Description
rTn	REAL	Defines the stage time (Tn, in milliseconds) for the reference ramp to reach the increased pressure setpoint (i_rSp + iq_stCompInit.rCorrPump(n)). Range: rTn > 0.0 ms Time required (Tn, in milliseconds) to increase the correction value by 1 unit.
rCorrPump1	REAL	Correction value that is added to the pressure setpoint (i_rSp) if the stage value requests 1 pump.
rCorrPump2	REAL	Correction value that is added to the pressure setpoint (i_rSp) if the stage value requests 2 pumps.
rCorrPump3	REAL	Correction value that is added to the pressure setpoint (i_rSp) if the stage value requests 3 pumps.
rCorrPump4	REAL	Correction value that is added to the pressure setpoint (i_rSp) if the stage value requests 4 pumps.
rCorrPump5	REAL	Correction value that is added to the pressure setpoint (i_rSp) if the stage value requests 5 pumps.
rCorrPump6	REAL	Correction value that is added to the pressure setpoint (i_rSp) if the stage value requests 6 pumps.
rCorrPump7	REAL	Correction value that is added to the pressure setpoint (i_rSp) if the stage value requests 7 pumps.
rCorrPump8	REAL	Correction value that is added to the pressure setpoint (i_rSp) if the stage value requests 8 pumps.

Section 1.7

CompSpFlow: Setpoint Adaptation by Using Flow Value (Friction Loss Compensation)

Overview

This section describes the CompSpFlow function block for friction loss compensation by adapting the pressure setpoint according to the flow value provided by the flow meter.

A flow meter is required in your system to use this function.

NOTE: This function block is designed to be used in conjunction with other function blocks of the Pumping Library.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description - CompSpFlow	84
Function Block Description - CompSpFlow	86
Structure Data Type Definitions - CompSpFlow	88

Description - CompSpFlow

Overview

In order to help to compensate for friction loss, the CompSpFlow function block adapts the pressure setpoint (i_rSp) according to the flow value provided by the flow meter. This helps to maintain a constant pressure even at the highest and/or most distant points of the system.

At least 2 pressure setpoint values together with the respective flow values (the pressure setpoint at minimum flow and the pressure setpoint at maximum flow) have to be configured as measuring points. The pressure values can be defined as a relative increment value (in percent) or an absolute value. The CompSpFlow function block allows you to configure a maximum of 6 pressure setpoints with the respective flow values. With the Newton interpolation formula, the function block generates, with the help of the defined measured points, an adapted setpoint.

Increasing the Pressure Setpoint According to the Flow

The diagram illustrates the relationship between the flow and the pressure. 2 measuring points are used to increment the pressure setpoint (i_rSp) in order to maintain a constant pressure even at the highest and/or most distant points of the system:

Last = Value from 2...6, here the value is 2

The 2 pressure setpoint values iq_stCompFlowInit.stPnt1 and iq_stCompFlowInit.stPnt(Last) (the pressure setpoint at minimum flow and the pressure setpoint at maximum flow) are defined.

If relative values for the pressure values are used, the pressure value of the measuring point 1 (`iq_stCompFlowInit.stPnt1`) is typically zero. The value of succeeding measuring points (`iq_stCompFlowInit.stPnt(Last)` in this example) must be greater than the preceding measuring point.

If absolute values are used, the pressure value of the measuring point 1 (`iq_stCompFlowInit.stPnt1`) is equal to the user-defined pressure setpoint (`i_rSp`). The value of measuring point `iq_stCompFlowInit.stPnt(Last)` must be greater than the user-defined pressure setpoint (`i_rSp`) and must be less than the user-defined setpoint maximum limit (`iq_stCompFlowInit.rSpMax`). The minimum calculated setpoint is limited by the `i_rSp` value. The maximum calculated setpoint is limited by the `iq_stCompFlowInit.rSpMax` value.

NOTE: If the highest measuring point is not close to the maximum flow, it is possible (due to interpolation) that the setpoint is increased until the value of the parameter `iq_stCompFlowInit.rSpMax` is reached.

Function Block Description - CompSpFlow

Pin Diagram

Brief Description

The CompSpFlow function block helps to maintain a constant pressure even at the highest and/or most distant points of the system. To achieve this, the pressure setpoint (i_rSP) is adapted according to the flow value provided by the flow meter to reduce friction loss.

I/O Variables Description

The table describes the input/output variables of the function block:

Input/Output	Data Type	Description
iq_stAlrmCtrl	stAlrmCtrl	Structure used for alarm handling. Refer to the structure data type description *(see page 166)*.
iq_stOpMode	stOpMode	Determines the operating modes. If the element xAuto of the structure data type stOpMode is set to TRUE, the function block executes its tasks automatically. With another status of this element, the outputs of this function block are set to their defined fallback (default) states. Refer to the structure data type description *(see page 166)*.
iq_stCompFlowInit	stCompFlowInit	Structure used for initialization data for the CompSpFlow function block. Refer to the structure data type description *(see page 82)*.

The table describes the input variables to the function block:

Input	Data Type	Description
`i_xRele`	BOOL	Activates (TRUE) or stops (FALSE) the execution of the function block.
`i_rSp`	REAL	Contains the user-defined pressure setpoint. This constant setpoint is considered during startup (pipe fill function) of the pumping process. It is adjusted by this compensation function and by other functions of this library. Range: 0.0...n
`i_stAna_Flow`	`stAna`	Structure for analog values, contains the flow input value and the state of this value. Refer to the structure data type description *(see page 168)*.

The table describes the output variables from the function block:

Output	Data Type	Description
`q_stRef_Sp`	`stRef`	Contains the adapted pressure setpoint and the state of this value. Refer to the structure data type description *(see page 175)*.
`q_stCompFlowErr`	`stCompFlowErr`	Structure that indicates the alarm and alert states detected by the `CompSpFlow` function block. Refer to the structure data type description of alarms *(see page 89)*.

Structure Data Type Definitions - CompSpFlow

Overview

Structure data types serve as the major input and output data carriers between different function blocks, configuration, and parameterization data.

The following section lists the names of structures (data types) that are used in this function block, followed by a table that describes the elements of structure.

NOTE: For the structure data types that are used from this function block and others, refer to the Common Structure Data Type Definitions chapter *(see page 166)*.

Structure: stCompFlowInit

The stCompFlowInit structure data type contains the initialization data for the CompSpFlow function block.

Structure: stSwcAlrm

The stSwcAlrm structure is used to indicate the detected alarm states.

Element of stSwcAlrm	Data Type	Description
rSpMax	REAL	Defines the maximum value allowed for the calculated pressure setpoint.
siNbPnt	SINT	Defines the number of measuring points (stPntn) used (pressure setpoint values together with the respective flow values). At least 2 measuring points are required to use this function. Range: 2...6
esiCompTyp	enumeration SINT	Defines whether the setpoint values are relative or absolute values. Range: ● relative (= 1) ● absolute (= 2)
stPnt1	stPnt	Structure defining the flow and pressure value for the measuring points. Refer to the structure data type stPnt *(see page 89)*.
stPnt2	stPnt	
stPnt3	stPnt	
stPnt4	stPnt	
stPnt5	stPnt	
stPnt6	stPnt	

Structure: stPnt

The stPnt structure data type contains a flow and a pressure value. These values define one measuring point.

Element of stPnt	Data Type	Description
rX_Flow	REAL	Defines the flow value for the measuring point (iq_stCompFlowInit.stPnt(n)).
rY_Pres	REAL	Defines the pressure value for the measuring point. Range: ● If relative: 0...100% ● If absolute: i_rSp <=rY_Pres <=rSpMax

Structure: stCompFlowErr

The stCompFlowErr structure data type is used to indicate alarms or alerts.

Element of stCompFlowErr	Data Type	Description
stCompFlowAlrm	stCompFlowAlrm	Structure data type for detected alarms.
stCompFlowAlrt	stCompFlowAlrt	Structure data type for detected alerts.

Structure: stCompFlowAlrm

The stCompFlowAlrm structure data type is used to indicate a detected alarm:

Element of stCompFlowAlrm	Data Type	Description
xNuAlrm	BOOL	When TRUE, the value of the parameter defining the number of measuring points (stCompFlowInit.siNbPnt) is not within the allowed range (2...6).
xPntXValAlrm	BOOL	When TRUE, the same flow value has been detected more than once.
xPntXSeqAlrm	BOOL	When TRUE, it has been detected that the flow values in stPnt structure are not monotonically increasing.
xPntYSeqAlrm	BOOL	When TRUE, it has been detected that the pressure values in stPnt structure are not monotonically increasing.
xPntYHighLimAlrm	BOOL	When TRUE, it has been detected that the adjusted pressure setpoint has exceeded the maximum value. This parameter is available if relative values are used and if the Y value of the measuring point is greater than 100.0%. rY_Pres > 100.0%

Element of `stCompFlowAlrm`	Data Type	Description
`xPntYLowLimAlrm`	BOOL	When TRUE, it has been detected that the adjusted pressure setpoint is below the minimum value. This parameter is available if absolute values are used and if the Y value of the measuring point is less than the pressure setpoint. `rY_Pres` > pressure setpoint (`i_rSp`)

Structure: `stCompFlowAlrt`

The `stCompFlowAlrt` structure data type is used to indicate a detected alert.

Element of `stCompFlowAlrt`	Data Type	Description
`xCalSpLimMaxAlrt`	BOOL	When TRUE, it has been detected that the adjusted setpoint has exceeded the limit. Adjusted setpoint > value defined for `stCompFlowInit.rSpMax`.
`xCalSpLimMinAlrt`	BOOL	When TRUE, it has been detected that the adjusted setpoint is below the setpoint value. Adjusted setpoint < pressure setpoint (`i_rSP`).

Section 1.8
CavtProt: Cavitation Protection

Overview

This section describes the CavtProt function block that helps to avoid operating the VS and FS pumps in a cavitation situation. To achieve this, it monitors the suction pressure and adapts the pressure setpoint, if necessary.

NOTE: A pressure sensor providing absolute pressure values is required to implement this function.

NOTICE
CAVITATION CAUSING INOPERABLE EQUIPMENT Select a pressure sensor providing absolute values and make sure to configure the correct NPSH+ value defined by the manufacturer of the pump you are using. **Failure to follow these instructions can result in equipment damage.**

NOTE: This function block is designed to be used in conjunction with other function blocks of the Pumping Library.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description - CavtProt	92
Function Block Description - CavtProt	97
Structure Data Type Definitions - CavtProt	99

Description - CavtProt

Overview

The CavtProt function block helps to avoid operating the VS and FS pumps in a cavitation situation caused by a low-pressure situation on the suction side.

To achieve this, it constantly verifies the suction pressure in the pumping application. The CavtProt function block starts to adapt the pressure setpoint if the suction pressure value drops below a certain cavitation protection start limit. If this measure does not stop the decrease of the suction pressure, a second limit value, referred to as NPSH+, is used. This value is a value provided by the manufacturer of the pumps. It indicates the minimum pressure that has to be available at the entry point of the pump to prevent the liquid from boiling and thus to avoid a cavitation situation. If the suction pressure drops below the NPSH+ value, the CavtProt function block indicates an alarm. As a result, the pumps are stopped.

Cavitation Protection Procedure

The diagram illustrates the influence of the suction pressure on the pressure setpoint during cavitation protection:

1 Suction pressure curve
2 The suction pressure becomes inferior to the suction pressure alert on-limit (`iq_stCavtInit.rNpshAlrtOn`), triggering the active cavitation protection.
3 The suction pressure reaches the NPSH+ threshold; alarm state = TRUE (1); pumps are stopped.
4 The suction pressure leaves the NPSH+ alarm area.
5 A user reset command and a user automatic command starts the machine.
6 The suction pressure exceeds the suction pressure alert off-limit (`iq_stCavtInit.rNpshAlrtOff`), terminating the active cavitation protection.
7 During the active phase of the cavitation protection, the setpoint is adjusted.
8 During an alarm state, the value of the setpoint is set to zero.
9 After a user reset command, the active cavitation protection starts to adjust the setpoint with the `iq_stCavtInit.rSpMin` as start value.
10 If the active cavitation protection ends, the setpoint is set to 100% (`q_stRef_Sp.rVal:=i_rSp`).

The cavitation protection process consists of the following stages:

Stage	Description
1	The suction pressure in the pumping application is constantly verified during normal operation.
2	If the suction pressure drops under the lower suction pressure limit value (position 1: `q_stCavtSta_Hmi.rNpshAlrtOn`), the cavitation protection function starts and the `CavtProt` function block executes the following actions: ● The operating mode is set to cavitation margin mode (`iq_stOpMode.xCavtMarg`). ● The output value `q_stCavtSta_Hmi.xProtActv` is set to TRUE (1) to indicate that the cavitation protection function is in progress. ● The output value `q_stCavtErr.stCavtAlrt.xSuctPresLim` is set to TRUE (1) to indicate that a cavitation alert has been detected. ● The pressure setpoint is adjusted according to the suction pressure value in order to reduce the flow and to increase the suction pressure.
3	If the suction pressure drops to the NPSH+ limit (`iq_stCavtInit.rNpshPlus`) (position 2), the `CavtProt` function block stops the pumps by executing the following actions: ● It indicates a detected cavitation alarm (by setting output `q_stCavtErr.stCavtAlrm.xSuctPresNpsh` to TRUE (1)). ● It sets the outputs to the default values (0 and FALSE (0)). ● It sets the alarm release bit (`iq_stAlrmCtrl.xAlrmRele`) to FALSE (0). ● It sets the operating mode to manual mode.
4	Once the suction pressure is greater than the NPSH+ limit (`iq_stCavtInit.rNpshPlus`), a reset must be executed to restart the pumps.
5	After the reset has been carried out, the cavitation logic resumes while the suction pressure is below the upper suction pressure alert on-limit (`iq_stCavtInit.rNpshAlrtOn`). The `CavtProt` function block executes the following actions: ● The cavitation margin mode flag (`iq_stOpMode.xCavtMarg`) into the operating mode structure is set to TRUE. ● The output value `q_stCavtSta_Hmi.xProtActv` is set to TRUE (1) to indicate that the cavitation protection function is in progress. ● The output value `q_stCavtErr.stCavtAlrt.xSuctPresLim` is set to TRUE (1) to indicate that a cavitation alert has been detected. ● The value configured for the parameter `iq_stCavtInit.rSpMin` (position 9) is used as the first pressure setpoint during restart after a reset. In the course of time the pressure setpoint is adapted according to the sign of the measured suction pressure gradient. If the suction pressure is rising, the pressure setpoint is incremented, and vice versa. The adjustment is realized with the value of the time constants TN (`iq_stCavtInit.rTn`).

Stage	Description
6	As soon as the suction pressure alert off-limit (`iq_stCavtInit.rNpshAlrtOff`) is reached, the cavitation protection action is stopped, and the `CavtProt` function block executes the following actions: ● The cavitation margin mode flag (`iq_stOpMode.xCavtMarg`) of the operating mode is set to FALSE (0). ● The output value `q_stCavtSta_Hmi.xProtActv` is set to FALSE (0). ● The setpoint is set to 100% (`q_stRef_Sp.rVal:=i_rSp`). ● The suction pressure in the pumping application is constantly verified.

Pressure Setpoint Adaptation

The diagram shows the pressure setpoint being adapted according to the suction pressure.

In this example, the reduction of the pressure setpoint and the flow is sufficient to prevent the suction pressure from dropping below the NPSH+ limit:

1 The suction pressure falls under the suction pressure alert on-limit (`iq_stCavtInit.rNpshAlrtOn`), triggering the starting of the active cavitation protection phase.

2 The suction pressure exceeds the suction pressure alert off-limit (`iq_stCavtInit.rNpshAlrtOff`), terminating the active cavitation protection phase.

The cavitation protection function adapts the pressure setpoint according to the suction pressure as follows:

Stage	Description
1	The suction pressure in the pumping application is constantly verified during normal operation.
2	If the suction pressure drops below the suction pressure alert on-limit value (position 1: `iq_stCavtInit.rNpshAlrtOn`), the cavitation protection function starts and the `CavtProt` function block executes the following actions: ● The cavitation margin mode flag (`iq_stOpMode.xCavtMarg)` into the operating mode structure is set to TRUE. ● The output value `q_stCavtSta_Hmi.xProtActv` is set to TRUE (1) to indicate that the cavitation protection function is in progress. ● The output value `q_stCavtErr.stCavtAlrt.xSuctPresLim` is set to TRUE (1) to indicate that a cavitation alert has been detected. ● The pressure setpoint is adjusted according to the sign of the measured suction pressure gradient. If the suction pressure is rising, the pressure setpoint is incremented, and vice versa. The adjustment is realized with the value of the time constants TN (`rTn`).
3	As soon as the suction pressure value reaches the suction pressure alert off-limit position 2: `iq_stCavtInit.rNpshAlrtOff`, the cavitation protection adjustment is stopped and the `CavtProt` function block executes the following actions: ● The cavitation margin mode flag (`iq_stOpMode.xCavtMarg`) of the operating mode is set to FALSE (0). ● The output value `q_stCavtSta_Hmi.xProtActv` is set to FALSE (0). ● The setpoint is set to 100% (`q_stRef.rVal:=i_rSp`). ● The suction pressure in the pumping application is constantly verified.

Function Block Description - CavtProt

Pin Diagram

Brief Description

The `CavtProt` function block helps to avoid operating the VS and FS pumps in a cavitation situation. To achieve this, it constantly verifies the suction pressure in the pumping application and adapts the setpoint of the PID control function, if necessary. If in spite of this measure the suction pressure drops below the NPSH+ value, the `CavtProt` function block indicates a detected alarm, induces the stop of the pumping application, by setting the `iq_stAlrmCtrl.xAlrmRele` bit to FALSE (0) to help avoid pump damage.

I/O Variables Description

The table describes the input/output variables of the function block:

Input/Output	Data Type	Description
`iq_stAlrmCtrl`	`stAlrmCtrl`	Structure used for alarm handling. Refer to the structure data type description *(see page 166)*.
`iq_stOpMode`	`stOpMode`	Determines the operating modes. If the element `xAuto` of the structure data type `stOpMode` is set to TRUE, the function block executes its tasks automatically. With another status of this element, the outputs of this function block are set to their defined fallback (default) states. Refer to the structure data type description *(see page 166)*.
`iq_stCavtInit`	`stCavtInit`	Structure used for initialization data for the cavitation protection function. Refer to the structure data type description *(see page 166)*.

The table describes the input variables to the function block:

Input	Data Type	Description
`i_xRele`	BOOL	Activates (TRUE) or stops (FALSE) the execution of the function block.
`i_stAna_SuctPres`	`stAna`	Structure used for the analog suction pressure input value and the state of this value. Refer to the structure data type description *(see page 168)*.
`i_rSp`	REAL	Contains the user-defined pressure setpoint. This constant setpoint is considered during startup (pipe fill function *(see page 126)*) of the pumping process. It is adapted by this cavitation function and can be adapted by other functions of this library, if used (ComSp *(see page 80)* and CompSpFLow *(see page 86)*). To help to prevent a cavitation situation, this value is adjusted by the `CavtProt` function block. The adjusted setpoint is provided by the output parameter `q_stRef_Sp`.

The table describes the output variables from the function block:

Output	Data Type	Description
`q_stRef`	`stRef`	Structure for setpoints contains the adapted pressure setpoint and the state of this value. Refer to the structure data type description *(see page 175)*.
`q_stCavtSta_Hmi`	`stCavtSta`	Structure that is used to display the status of the cavitation protection function on the HMI. Refer to the structure data type description *(see page 174)*.
`q_stCavtErr`	`stCavtErr`	Structure used for states of alarms and alerts detected by the cavitation protection function. Refer to the structure data type description for detected cavitation protection alarms *(see page 97)*.

Structure Data Type Definitions - CavtProt

Overview

Structure data types serve as the major input and output data carriers between different function blocks, configuration, and parameterization data.

The following section lists the names of structures (data types) that are used in this function block, followed by a table that describes the elements of structure.

NOTE: For the structure data types that are used from this function block and others, refer to the Common Structure Data Type Definitions chapter *(see page 166)*.

Structure: stCavtInit

The stCavtInit structure data type contains the initialization data for the CavtProt function block.

Element of stCavtInit	Data Type	Description
rNpshPlus	REAL	Defines the NPSH+ value as defined by the manufacturer of the pumps. If the suction pressure drops below this value, the pumping application is stopped and an alarm is detected.
rNpshAlrtOn	REAL	Defines the lower suction pressure limit value. If the suction pressure drops below this value, the cavitation protection procedure starts and an alert is detected. Range: rNpshAlrtOn > rNpshPlus
rNpshAlrtOff	REAL	Defines the upper suction pressure limit value. If the suction pressure exceeds this value, the cavitation protection procedure stops and an alert is cleared. Range: rNpshAlrtOff > rNpshAlrtOn
rSpMin	REAL	Defines the minimal setpoint value for the pressure. The setpoint adaptation of the function uses this limit value for the minimum of the adapted pressure setpoint value. After a reset of a detected NPSH+ alarm, this is the first (minimum) pressure value that is used as a setpoint. In the course of time, the pressure setpoint is adjusted according to the measured suction pressure value. Range: rSpMin > 0

Element of `stCavtInit`	Data Type	Description
`rTn`	REAL	Defines the stage time (*Tn*, in milliseconds) for the reference ramp. Range: `rTn` > 0.0 ms The time required to increase the correction value by 1 unit.

Structure: `stCavtErr`

The `stCavtErr` structure data type is used to indicate alarms or alert states detected during cavitation protection.

Element of `stCavtErr`	Data Type	Description
`stCavtAlrm`	`stCavtAlrm`	Structure data type for detected alarms.
`stCavtAlrt`	`stCavtAlrt`	Structure data type for detected alerts.

Structure: `stCavtAlrm`

The `stCavtAlrm` structure data type is a substructure of `stCavtErr`. The `stCavtAlrm` structure data type is used to indicate the alarm states detected during cavitation protection.

Element of `stCavtAlrm`	Data Type	Description
`xSuctPresNpsh`	BOOL	When TRUE, the suction pressure has dropped below the NPSH+ value.
`xSuctPresSta`	BOOL	When TRUE, it has been detected that the suction pressure value is not valid.

Structure: `stCavtAlrt`

The `stCavtAlrt` structure data type is a substructure of `stCavtErr` . The `stCavtAlrt` structure data type is used to indicate the alert states detected during cavitation protection.

Element of `stCavtAlrt`	Data Type	Description
`xSuctPresLim`	BOOL	When TRUE, the actual suction pressure has dropped below the lower suction pressure limit value. As a result of this detected alert state, the cavitation protection function is started.
`xReserve`	BOOL	This parameter is currently not used.

Section 1.9

PumpAvai: Pump Availability Status

Overview

This section describes the PumpAvai function block for indicating the availability of a maximum of 8 pumps.

NOTE: This function block is designed to be used in conjunction with other function blocks of the Pumping Library.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description - PumpAvai	102
Function Block Description - PumpAvai	103

Description - PumpAvai

Overview

The PumpAvai function block indicates the availability of a maximum of 8 pumps by processing the inputs of external key switches, sensors, and contactors.

NOTE: It indicates the availability of a functional homogenous set of pumps. You would need one instance of this function block for VS pumps and one instance of this function block for FS pumps, if any.

The inputs processed by the function block may include:

- the mode of the pump (hand mode or maintenance mode)
- a detected alarm (for example, dry run or temperature)
- a detected hardware error (for example, motor circuit breaker tripped)
- a detected connection loss between the controller and distributed I/O modules that may be involved in detecting the status of the pump

Function Block Description - PumpAvai

Pin Diagram

Brief Description

The PumpAvai function block verifies the status and the operating modes of a maximum of 8 pumps to indicate their availability to the other function blocks.

I/O Variables Description

The table describes the input/output variables of the function block:

Input/Output	Data Type	Description
iq_stAlrmCtrl	stAlrmCtrl	Structure used for alarm handling. Refer to the structure data type description *(see page 166)*.
iq_stOpMode	stOpMode	Determines the operating modes. If the element xAuto of the structure data type stOpMode is set to TRUE, the function block executes its tasks automatically. With another status of this element, the outputs of this function block are set to their defined fallback (default) states. Refer to the structure data type description *(see page 166)*.
iq_astMode	ARRAY [0..7] of stMode	Array of stMode to control and switch the operating mode of the 8 pumps. If the element xAuto of the structure data type stMode is set to TRUE, the pump is available for automatic mode. With another setting of this element, the respective pump is not available. Refer to the structure data type description *(see page 169)*.

The table describes the input variables to the function block:

Input	Data Type	Description
`i_xRele`	BOOL	Activates (TRUE) or stops (FALSE) the execution of the function block.
`i_siNuPump`	SINT	Defines the number of installed pumps. Range: 1...8
`i_astSta_Ex`	ARRAY [0..7] of `stSta`	Array of `stSta`. Contains the external status information from 8 pumps. Refer to the structure data type description *(see page 170)*.

The table describes the output variable from the function block:

Output	Data Type	Description
`q_stAvai`	`stAvai`	Structure that indicates the availability for a maximum of 8 pumps. Refer to the structure data type description *(see page 168)*.

Section 1.10
VsdAvai: VSD Available Status

Overview

This section describes the VsdAvai function block for indicating the availability of a maximum of 8 VSDs.

NOTE: This function block is designed to be used in conjunction with other function blocks of the Pumping Library.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description - VsdAvai	106
Function Block Description - VsdAvai	107

Description - VsdAvai

Overview

The VsdAvai function block indicates the availability of a maximum of 8 VSDs by processing the inputs of the individual VSDs.

The inputs processed by the function block may include:

- the physical state (power OK)
- the mode of the VSD (automatic or manual mode)
- a detected alarm state (for example, no response)
- a detected hardware error (for example, motor circuit breaker tripped)
- a detected connection loss between the controller and distributed I/O modules that may be involved in detecting the status of the VSD

Function Block Description - VsdAvai

Pin Diagram

Brief Description

The `VsdAvai` function block verifies the status and the operating modes of a maximum of 8 VSDs to indicate their availability to the other function blocks.

I/O Variables Description

The table describes the input/output variables of the function block:

Input/Output	Data Type	Description
`iq_stAlrmCtrl`	`stAlrmCtrl`	Structure used for alarm handling. Refer to the structure data type description *(see page 166)*.
`iq_stOpMode`	`stOpMode`	Determines the operating modes. If the element `xAuto` of the structure data type `stOpMode` is set to TRUE, the function block executes its tasks automatically. With another status of this element, the outputs of this function block are set to their defined fallback (default) states. Refer to the structure data type description *(see page 166)*.
`iq_astMode`	ARRAY [0..7] of `stMode`	Array of `stMode` to control and switch the operating mode of the 8 VSDs. If the element `xAuto` of the structure data type `stMode` is set to TRUE, the VSD is available for automatic mode. With another setting of this element, the respective VSD is not available. Refer to the structure data type description *(see page 169)*.

The table describes the input variables to the function block:

Input	Data Type	Description
`i_xRele`	BOOL	Activates (TRUE) or stops (FALSE) the execution of the function block.
`i_siNuVsd`	SINT	Defines the number of installed VSDs. Range: 1...8
`i_astVsdSta_Ex`	ARRAY [0..7] of `stVsdSta`	Array of `stVsdSta`. Contains the extern status information from 8 VSDs. Refer to the structure data type description *(see page 171)*.

The table describes the output variable from the function block:

Output	Data Type	Description
`q_stAvai`	`stAvai`	Structure that indicates the availability for a maximum of 8 VSDs. Refer to the structure data type description *(see page 168)*.

Section 1.11
OpPrty: Operation Priority Device

Overview

This section describes the OpPrty function block for assigning a priority to the VSDs or pumps.

The function block can process 2 different types of priority:
- the operating hours of each VSD or pump in order to help to keep the number of operating hours at relatively the same level for the different VSDs / pumps,
- a user-defined priority table that allows you to define preferences for the different VSDs / pumps.

NOTE: This function block is designed to be used in conjunction with other function blocks of the Pumping Library.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description - OpPrty	110
Function Block Description - OpPrty	112
Structure Data Type Definitions - OpPrty	114

Description - OpPrty

Overview

The OpPrty function block is used to assign a priority to the different VSDs or pumps available in your system.

NOTE: It sets the priority of a functional homogenous set of pumps and VSDs. You would need one instance of this function block for VS pumps / VSDs and one instance of this function block for FS pumps, if any.

The function block can process 2 different types of priority:

- The operating hours of each pump to help balance the operating hours between pumps / VSDs.
- A user-defined priority table that allows you to define your own preferences for the different pumps / VSDs.

Assigning a Priority According to the Operating Hours

The graphic shows an example of the priority assignment to devices (pumps or VSDs) according to their operating hours:

To activate this type of priority assignment, set the input parameter i_esiPrtyMode to the value esiPrtyMode.Op.

The OpPrty function block evaluates the operating hour values provided as input i_audiOpHr by the function blocks FsPumpCtrl, VsPumpCtrl, and VsdCtrl to assign a priority to the pumps or VSDs.

The devices (pumps or VSDs) are arranged in their order of priority, with the highest priority being at the top of the output array table. The numbers of the output array q_asiPrty directly point to the respective pump or VSD (used as an index into an array of structures). This list starts with 0, which results in device 1 being assigned array index 0, and so on.

In case of identical operating hour values for 2 devices, the device that is on a higher place in the input table is also placed higher in the output table. This is illustrated in the graphic. Devices 1 and 3 both have a value of 100 operating hours. In the output table, they are on place 1 (device 1 --> array index 0) and place 2 (device 3 --> array index 1).

Assigning a Priority According to User-Defined Inputs

The graphic shows an example of the priority assignment to pumps according to a priority table that is configured by the operator:

To activate this type of priority assignment, set the input parameter `i_esiPrtyMode` to the value `esiPrtyMode.User`.

The `OpPrty` function block assigns the priority to the VSDs and pumps according to the values provided as input `i_stUsrTbl` by the operator.

The devices (pumps or VSDs) are arranged in their order of priority, with the highest priority being at the top of the output array table. The numbers of the output array `q_asiPrty` directly point to the respective pump or VSD (used as an index into an array of structures). This list starts with 0, which results in device 1 being assigned array index 0, and so on.

In case of identical values for 2 devices, the device that is on a higher place in the input table is also placed higher in the output table. This is illustrated in the graphic. Devices 1 and 4 are both assigned priority 1 by the operator. In the output table, they are on places 1 (device 1 --> array index 0) and 2 (device 4 --> array index 1).

Function Block Description - OpPrty

Pin Diagram

Brief Description

The OpPrty function block is used to assign a priority to the different pumps or VSDs available in a system.

The function block can process 2 different types of priority:

- the operating hours of each VSD or pump in order to help to keep the number of operating hours at relatively the same level for the different VSDs / pumps,
- a user-defined priority table that allows you to define preferences for the different VSDs / pumps.

I/O Variables Description

The table describes the input/output variable of the function block:

Input/Output	Data Type	Description
`iq_stAlrmCtrl`	`stAlrmCtrl`	Structure used for alarm handling. Refer to the structure data type description *(see page 166)*.

The table describes the input variables to the function block:

Input	Data Type	Description
`i_xRele`	BOOL	Activates (TRUE) or stops (FALSE) the execution of the function block.
`i_siNb`	SINT	Defines the number of pumps or VSDs that are installed in the system. Range: 1...8
`i_audiOpHr`	ARRAY [0..3] of UDINT	This array is a table listing the operating hours of the pumps or VSDs (maximum of 8). The operating hour values are provided by the following function blocks: ● `FsPumpCtrl` for FS pumps *(see page 74)* ● `VsPumpCtrl` for VS pumps *(see page 70)* ● `VsdCtrl` for VSDs *(see page 66)*
`i_stUsrTbl`	`stUsrTbl`	The user-defined priority table is defined as input. Refer to the structure data type description *(see page 114)*.
`i_esiPrtyMode`	`esiPrtyMode`	Defines the type of priority that is used. Range: ● `esiPrtyMode.Op` (=0) ● `esiPrtyMode.User` (=1) Refer to the structure data type description *(see page 114)*.

The table describes the output variable from the function block:

Output	Data Type	Description
`q_asiPrty`	ARRAY [0..8] of SINT	This array is a table listing the pumps or VSDs in the order according to the calculated priority. Refer to the structure data type description *(see page 176)*.

Structure Data Type Definitions - OpPrty

Overview

Structure data types serve as the major input and output data carriers between different function blocks, configuration, and parameterization data.

The following section lists the names of structures (data types) that are used in this function block, followed by a table that describes the elements of structure.

NOTE: For the structure data types that are used from this function block and others, refer to the Common Structure Data Type Definitions chapter *(see page 166)*.

Structure: stUsrTbl

The stUsrTbl structure data type contains the user-defined priority table. It allows you to assign a priority value to each of the 8 pumps or VSDs:

Element of stUsrTbl	Data Type	Description
siDev1Prty	SINT	Defines the priority value for VSD 1 or pump 1.
...	...	...
siDev8Prty	SINT	Defines the priority value for VSD 8 or pump 8.

Enumeration: esiPrtyMode

The enumeration esiPrtyMode data type is used to define the type of priority that is used.

Enumeration of esiPrtyMode	Data Type	Description
Op	UDINT	The priority is assigned to the pumps or VSDs according to the operating hours. The operating hour values are provided via the parameter audiOpHr by the following function blocks: ● FsPumpCtrl for FS pumps *(see page 74)* ● VsPumpCtrl for VS pumps *(see page 70)* ● VsdCtrl for VSDs *(see page 66)* Value: 0
User	UDINT	The priority is assigned to the pumps or VSDs according to the user table. The priority values for the different pumps or VSDs are defined by the operator via the parameter i_stUsrTbl. Value: 1

Section 1.12
AuxPumpCtrl: Auxiliary Pump Control

Overview

This section describes the AuxPumpCtrl function block for controlling and switching an auxiliary pump. The auxiliary pump is used in times of low demand (for example at night time) to maintain the pressure within a certain pressure range without using the main pumps.

NOTE: This function block is designed to be used in conjunction with other function blocks of the Pumping Library.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description - AuxPumpCtrl	116
Function Block Description - AuxPumpCtrl	120
Structure Data Type Definitions - AuxPumpCtrl	123

Description - AuxPumpCtrl

Overview

The AuxPumpCtrl function block controls and switches the auxiliary pump in times of low demand (for example at night time).

If the demand of water is low and the last main pump is running, the PumpPidStag function block *(see page 25)* increases the pressure setpoint to a value that corresponds to the average of the working range of the auxiliary pump. When this pressure is reached, the PumpPidStag function changes the mode to sleep mode by setting the stage value to zero and the iq_stOpMode.xSle flag bit to TRUE (1). The result is that the main pumps are stopped. The auxiliary pump is working only in sleep mode to maintain the pressure within a certain pressure range avoiding to use the main pumps.

To maintain the pressure within the limits defined for sleep mode, the AuxPumpCtrl function block switches on or off the auxiliary pump. Additionally, the AuxPumpCtrl function block contains a flow limit and a pressure limit that allow to switch off the sleep mode and to return to operation of the main pumps if the demand rises.

Prerequisite

The AuxPumpCtrl function block executes its tasks only if the following conditions apply:

- The auxiliary pumping application is running on demand during the sleep mode.
  The sleep mode is established by the PumpPidStag function block *(see page 25)* setting the element of the structure data type iq_stOpMode.xSle to TRUE (for more information refer to the chapter Common Structure Data Type Definitions *(see page 166)*).
  With another setting of the stOpMode element xSle of the structure data type iq_stOpMode, the AuxPumpCtrl function block is disabled and the function block outputs are set to the default value (0 and FALSE).
- The alarm release bit xAlrmRele of the structure data type stAlrmCtrl *(see page 166)* is set to TRUE, indicating that no alarm has been detected in the pump group.
  If the alarm release bit xAlrmRele is set to FALSE (0), the AuxPumpCtrl function block is disabled and the function block outputs are set to the default value (0 and FALSE).

Auxiliary Pump Control with Reset of Sleep Mode Using the Flow Value

The graphic shows the switch procedures of the `AuxPumpCtrl` function block during sleep mode and the reset of the sleep mode with the help of the flow value:

1 OFF limit for auxiliary pump (`rPresAuxLimOff`)
2 ON limit for sleep mode (defined in `PumpPidStag` function block (`irSP` + `stStagInit.rPresIncPump1SwcOff`))
3 ON limit for auxiliary pump (`rPresAuxLimOn`)
4 OFF limit by pressure for sleep mode (`rPresSleOff`) (not used in this graphic)
5 OFF limit by flow for sleep mode (`rFlowSleOff`))

The auxiliary pump control process consists of the following stages when the sleep mode is reset with the help of the flow value:

Stage (Time)	Description
1 (T1)	The `PumpPidStag` function block *(see page 25)* detects a low flow situation and sets the operating mode to sleep mode. **Result**: The `AuxPumpCtrl` function block is activated and monitors the pressure and the flow values.
2 (T2)	As soon as the pressure value drops below the user-defined ON limit for the auxiliary pump (`rPresAuxLimOn`), the auxiliary pump is set to the ON state.
3 (T3)	As soon as the pressure value rises above the OFF limit for the auxiliary pump (`rPresAuxLimOff`), the auxiliary pump is set to the OFF state.
4 (T4)	If the flow value rises above the user-defined OFF limit by flow for sleep mode (`rFlowSleOff`), the `AuxPumpCtrl` function block executes the following actions: ● setting the sleep mode `iq_stOpMode.xSle` to FALSE (0) ● setting the outputs to the default values (0 and FALSE)

Auxiliary Pump Control with Reset of Sleep Mode Using the Pressure Value

The graphic shows the switch procedures of the `AuxPumpCtrl` function block during sleep mode and the reset of the sleep mode with the help of the pressure value:

1 OFF limit for auxiliary pump (`iq_stAuxPumpCtrlInit.rPresAuxLimOff`)
2 ON limit for sleep mode (defined in `PumpPidStag` function block (`i_rSp + stStagInit.rPresIncPump1SwcOff`)
3 ON limit for auxiliary pump (`iq_stAuxPumpCtrlInit.rPresAuxLimOn`)
4 OFF limit by pressure for sleep mode (`iq_stAuxPumpCtrlInit.rPresSleOff`)

The auxiliary pump control process consists of the following stages when the sleep mode is reset with the help of the pressure value:

Stage (Time)	Description
1 (T1)	The `PumpPidStag` function block *(see page 25)* sets the operating mode to sleep mode. **Result**: The `AuxPumpCtrl` function block is activated and monitors the pressure value.
2 (T2)	As soon as the pressure value drops below the user-defined ON limit for the auxiliary pump (`iq_stAuxPumpCtrlInit.rPresAuxLimOn`), the auxiliary pump is set to the ON state.
3 (T3)	As soon as the pressure value rises above the OFF limit for the auxiliary pump (`iq_stAuxPumpCtrlInit.rPresAuxLimOff`, the auxiliary pump is set to the OFF state.
4 (T4)	If the pressure value falls below the user-defined OFF limit by pressure for sleep mode (`rPresSleOff`), the `AuxPumpCtrl` function block executes the following actions: ● setting the sleep mode to FALSE (`iq_stOpMode.xSle` = FALSE (0)) ● setting the outputs to the default values (0 and FALSE)

Reset of Sleep Mode by the `PumpPidStag` function block

The `PumpPidStag` function block *(see page 25)* can also stop the sleep mode.

If the `PumpPidStag` function block detects that the stage function switches a main pump into run state, the function block sets the element `xSle` of `iq_stOpMode` to FALSE (0). This deactivation of the sleep mode is detected by the `AuxPumpCtrl` function block. It stops processing variables and sets the outputs to the default values (0 and FALSE).

Function Block Description - AuxPumpCtrl

Pin Diagram

Brief Description

The `AuxPumpCtrl` function block is used for controlling and switching the auxiliary pump in times of low demand (during sleep mode) to avoid using the main pumps.

I/O Variables Description

Structure used for alarm handling. Refer to the structure data type description *(see page 166)*.

The table describes the input/output variables of the function block:

Input/Output	Data Type	Description
`iq_stAlrmCtrl`	`stAlrmCtrl`	Structure used for alarm handling. Refer to the structure data type description *(see page 166)*.
`iq_stOpMode`	`stOpMode`	Determines the operating modes. If the element `xSle` of the structure data type `stOpMode` is set to TRUE, the function block executes its tasks. With another setting of this element, the outputs of this function block are set to their defined fallback (default) states. Refer to the structure data type description *(see page 166)*.

Input/Output	Data Type	Description
iq_stMode_Aux	stMode	Structure for the mode states to control and switch the operating mode of the auxiliary pump. Refer to the structure data type description *(see page 169)*.
iq_stAux_PumpCtrlInit	stAuxPumpCtrlInit	Structure used for initialization data for the auxiliary pump. Refer to the structure data type description *(see page 123)*.
iq_udiAuxPumpOpHr	UDINT	Lists the operating hours of the auxiliary pump. This input/output variable can be reset or adjusted even from HMI or SCADA systems.

The table describes the input variables to the function block:

Input	Data Type	Description
i_xRele	i.param	Activates (TRUE) or stops (FALSE) the execution of the function block.
i_stSta_AuxPump_Ex	stSta	Structure that contains the status of the auxiliary pump. Refer to the structure data type description *(see page 170)*.
i_xManCmd	BOOL	When TRUE, commands the start of the auxiliary pump.
i_stAna_DiscPres	stAna	Structure that contains the discharge pressure and its status. Refer to the structure data type description *(see page 168)*.
i_stAna_Flow	stAna	Structure that contains the flow input value and its status. Refer to the structure data type description *(see page 168)*.
i_xRes	BOOL	Resets alarms detected in the auxiliary pump when the cause has been corrected.

The table describes the output variables from the function block:

Output	Data Type	Description
`q_stAuxPumpCtrlSta_Hmi`	`stAuxPumpCtrlSta`	Structure is used to display the status of the auxiliary pump on the HMI. Refer to the structure data type description *(see page 124)*.
`q_stAuxPumpCom_Ex`	`stAuxPumpCom`	Structure that is used to activate the auxiliary pump. Refer to the structure data type description *(see page 173)*.
`q_stAuxPumpAlrm`	`stAuxPumpAlrm`	Structure is used to indicate alarm states of the auxiliary pump. Refer to the structure data type description *(see page 124)*.

Structure Data Type Definitions - AuxPumpCtrl

Overview

Structure data types serve as the major input and output data carriers between different function blocks, configuration, and parameterization data.

The following section lists the names of structures (data types) that are used in this function block, followed by a table that describes the elements of structure.

NOTE: For the structure data types that are used from this function block and others, refer to the Common Structure Data Type Definitions chapter *(see page 166)*.

Structure: stAuxPumpCtrlInit

The stAuxPumpCtrlInit structure data type contains the initialization data for the AuxPumpCtrl function block.

Element of stAuxPumpCtrlInit	Data Type	Description
rFlowSleOff	REAL	Contains the OFF limit by flow for the sleep mode. If this flow value is reached, the sleep mode variable iq_stOpMode.xSle is set to FALSE by the AuxPumpCtrl function block.
rPresSleOff	REAL	Contains the OFF limit by pressure for the sleep mode. If this pressure value is reached, the sleep mode variable iq_stOpMode.xSle is set to FALSE by the AuxPumpCtrl function block.
rPresAuxLimOn	REAL	Contains the ON limit by pressure for the auxiliary pump. If this pressure value is reached, the AuxPumpCtrl function block starts the auxiliary pump to keep the pressure value within the pressure range allowed during sleep mode.
rPresAuxLimOff	REAL	Contains the OFF limit by pressure for the auxiliary pump. If this pressure value is reached, the AuxPumpCtrl function block stops the auxiliary pump.
tFbckDel	TIME	Defines the time allowed for the auxiliary pump to send a response. Range: tFbckDel > T > 0 s As soon as the command is not equal to the response message received from the auxiliary pump, the delay timer of the feedback alarm is started. If the time elapses without receiving a response message equal to the command, an alarm is detected.

Element of **stAuxPumpCtrlInit**	Data Type	Description
tMinTime	TIME	Defines the minimum time the auxiliary pump has to be running before it can be switched off again. This time span is also used as minimum stop time (time the auxiliary pump has to be in stop state before it can be switched on again).

NOTE: Thoroughly test all functions during verification and commissioning.

Structure: stAuxPumpCtrlSta

The stAuxPumpCtrlSta structure data type displays the status of the auxiliary pump on the HMI.

Element of **stAuxPumpCtrlSta**	Data Type	Description
xAuxPumpAct	BOOL	When TRUE, the auxiliary pump is active.
xSleAct	BOOL	When TRUE, the sleep mode is active.
xSleInt	BOOL	When TRUE, the AuxPumpCtrl function block has set the sleep mode to OFF because it has detected that the flow is above the allowed limit or the pressure is below the allowed limit.

Structure: stAuxPumpAlrm

The stAuxPumpAlrm output structure data type is used to indicate the detected alarm states.

Element of **stAuxPumpAlrm**	Data Type	Description
xParaVal	BOOL	When TRUE, a configuration error has been detected.
xStaErr	BOOL	When TRUE, a hardware error of the auxiliary pump has been detected.
xFbck	BOOL	When TRUE, a feedback error has been detected because no response has been received from the auxiliary pump within the defined feedback time.

Section 1.13
PipeFill: Pipe Fill Control

Overview

This section describes the PipeFill function block for verifying the pressure in the pipes.

After a start or a restart of the pumping application, it starts specific processes to reach the pressure working range as quickly as possible.

During operation, it constantly monitors whether the pressure level is within the working range. If the pressure level is higher or lower than the working range, the function block generates an alarm after a certain delay time.

NOTE: This function block is designed to be used in conjunction with other function blocks of the Pumping Library.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description - PipeFill	126
Function Block Description - PipeFill	130
Structure Data Type Definitions - PipeFill	132

Description - PipeFill

Overview

The PipeFill function block verifies the pressure in the pipe of the discharge side.

After start or restart of the pumping application, it verifies the discharge pressure value. If the pressure is not within the working range (user-defined pressure setpoint i_rSp ± the setpoint tolerance range iq_stFillInit.rTnce), it starts the adequate procedure to reach the working range of the pressure by using a systematic process.

During normal operation, the PipeFill function block constantly monitors whether the pressure level is within the working range. If the pressure level is higher or lower than the working range for a certain delay time, the function block indicates the detected alarm.

Actions After Start / Restart of the Pumping Application

After a start / restart of the pumping application, the PipeFill function block verifies whether the pressure in the discharge pipe is within the working range.

Depending on the discharge pressure value, the function block reacts differently:

If...	Then the PipeFill function block
the discharge pressure is **within** the working range	• indicates that the pressure is OK by setting the input/output value iq_stOpMode.xFillOk to TRUE. • continues to verify the pressure value as described in the paragraph *Actions during Normal Operation* *(see page 129)*.
the discharge pressure is **below** the working range	• sets the input/output value iq_stOpMode.xFill to TRUE to indicate that the pipe fill function is in progress. • starts the adequate sequence (sequence 1 *(see page 127)* or sequence 2 *(see page 128)*) of the pipe fill procedure to reach the working range of the pressure as quickly as possible.

The procedure to reach the working range of the pressure is divided in 2 different sequences. It depends on the discharge pressure value which sequence is started.

The diagram shows the pressure setpoint values for the 2 sequences and their influences on the pressure in the pumping application:

1 user-defined pressure setpoint value (`i_rSp`)
2 pressure working range (that is the allowed deviation (`iq_stFillInit.rTnce`) from the user-defined pressure setpoint value (`i_rSp`)
3 minimal startup discharge pressure limit (`stFillInit.rFillPres`)
4 present discharge pressure of this pressure zone
5 minimal startup discharge pressure limit (`iq_stFillInit.rFillPres`). If the pressure is below this limit, sequence 1 is started. If the pressure is above this limit and not in the working range (2), then sequence 2 of the pipe fill procedure is started.
6 The setpoint is constantly incremented and provided to the PID control unit. After the working range has been reached, the user-defined pressure setpoint value (1) is used by the PID control unit.

Sequence 1 of the Start / Restart Procedure

Sequence 1 of the pipe fill procedure is started, if the pressure value is between 0 and the pipe fill pressure (`iq_stFillInit.rFillPres`, item 5 in the diagram).

Actions executed during sequence 1

Step	Description
1	The operating mode is set to pipe fill mode (`iq_stOpMode.xFill`) to TRUE to indicate the mode to the other function blocks of the pumping application.
2	A timer (defined with the parameter `iq_stFillInit.tSequ1`) is started with the start of sequence 1.
3	The output value `iq_stOpMode.xFill` is set to TRUE to indicate that the pipe fill function is in progress.

Step	Description
4	The output value `q_stFillStat.xSequ1` is set to TRUE to indicate that the pipe fill function is running in sequence 1. **Result**: The 2 outputs `iq_stOpMode.xFill` and `iq_stFillInit.xSequ1` are read by the `PumpPidStag` function block. If they are both set to TRUE, then the `PumpPidStag` function block detects that sequence 1 of the pipe fill function is running.
5	The `PumpPidStag` function block *(see page 25)* initiates the start of a fixed number of pumps with a fixed speed (configured for this sequence 1 of the pipe fill case with the parameter `iq_stFillDevInit.siNoPump`).
6	The `PumpPidStag` function block stops PID control by setting `q_stPidSta.xStopActv` to TRUE because it is not used in sequence 1.
7	If the discharge pressure reaches the pipe fill pressure limit (item 5 in the previous diagram) within the time allowed for sequence 1 (`iq_stFillInit.tSequ1`), then sequence 2 starts. If the timer expires and the pipe fill pressure is not reached, the `PipeFill` function block executes the following actions: ● It indicates a detected alarm (by setting the output `q_stFillAlrm.xSequ1TimeOut` to TRUE). ● It sets the outputs to the default values (0 and FALSE). ● It sets the operating mode to manual mode.

Sequence 2 of the Start / Restart Procedure

Sequence 2 of the pipe fill procedure is started if the discharge pressure value is higher than the pipe fill pressure limit (item 5 in the previous diagram) but lower than the working pressure range (item 2 in the diagram).

Actions executed during sequence 2

Step	Description
1	The operating mode is set to pipe fill mode (`iq_stOpMode.xFill`) to indicate the mode to the other function blocks of the pumping application.
2	A timer (defined with the parameter `iq_stFillInit.tSequ2`) is started with the start of sequence 2.
3	The `PipeFill` function block uses the pressure as the start up value for the setpoint value at the output `q_stRef_FillRef.rVal`.
4	The output value `q_stFillStat.xSequ2` is set to TRUE to indicate that the pipe fill function is running in sequence 2. **Result**: `iq_stOpMode.xFill` and `iq_stFillInit.xSequ2` are read by the `PumpPidStag` function block. If they are both set to TRUE, then the `PumpPidStag` function block detects that sequence 2 of the pipe fill function is running.

Step	Description
5	The `PumpPidStag` function block starts PID control by setting `q_stPidSta.xStopActv` to FALSE.
6	The `PipeFill` function block automatically increases the pressure setpoint in a ramp until the pressure reaches the working pressure range.
7	If the working pressure range (item 2 in the previous diagram) is reached within the time allowed for sequence 2 (`iq_stFillInit.tSequ2`), the `PipeFill` function block indicates that the pressure is OK by setting the input/ output value `iq_stOpMode.xFillOk` to TRUE. It continues to verify the pressure value as described in the paragraph *Actions during Normal Operation*. If the pressure working range is not reached within the defined time, the `PipeFill` function block executes the following actions: ● It indicates a detected alarm (by setting the output `q_stFillAlrm.xSequ2TimeOut` to TRUE). ● It sets the outputs to the default values (0 and FALSE). ● It sets the operating mode to manual mode (`iq_stOpMode.xMan`).

Actions during Normal Operation

During normal operation, the `PipeFill` function block constantly monitors whether the discharge pressure value is within the working pressure range (setpoint influenced by other functions during operation `i_rSpAct` ± the setpoint tolerance range `iq_stFillInit.rTnce`).

If...	Then the `PipeFill` function block
the pressure level is **higher** than the working range for a certain delay time (`iq_stFillInit.tPresAlrmDel`)	indicates a detected alarm by setting the output `q_stFillAlrm.xHighPres` to TRUE
the pressure level is **lower** than the working range for a certain delay time (`iq_stFillInit.tPresAlrmDel`)	indicates a detected alarm by setting the output `q_stFillAlrm.xLowPres` to TRUE

If an alarm is detected, the `PipeFill` function block executes the following actions:

- It sets the outputs to the default values (0 and FALSE).
- It sets the operating mode to manual mode.

Function Block Description - `PipeFill`

Pin Diagram

Brief Description

The `PipeFill` function block verifies the discharge pressure in the discharge pipe to help to reach the working pressure range of the pressure quickly after start or restart of the pumping application. During normal operation, it generates an alarm if it detects that the pressure level is higher or lower than the working range for a certain time.

I/O Variables Description

The table describes the input/output variables of the function block:

Input/Output	Data Type	Description
`iq_stAlrmCtrl`	`stAlrmCtrl`	Structure used for alarm handling. Refer to the structure data type description *(see page 166)*.
`iq_stOpMode`	`stOpMode`	Determines the operating modes. If the element `xAuto` of the structure data type `stOpMode` is set to TRUE, the function block executes its tasks automatically. With another status of this element, the outputs of this function block are set to their defined fallback (default) states. Refer to the structure data type description *(see page 166)*.
`iq_stFillInit`	`stFillInit`	Structure used for initialization data. Refer to the structure data type description *(see page 132)*.

The table describes the input variables to the function block:

Input	Data Type	Description
`i_xExe`	BOOL	Input from the operator to start the pipe fill mode manually. When TRUE, the pipe fill mode is started.
`i_stAna_DiscPres`	`stAna`	Structure used for the analog input value containing the discharge pressure and the state of this value. Refer to the structure data type description *(see page 168)*.
`i_rSp`	REAL	Contains the user-defined pressure setpoint which can be adapted by other functions, like compensation functions *(see page 77)* using the flow value from flow meter, if available *(see page 84)* or the cavitation protection function *(see page 92)*.
`i_rSpAct`	REAL	Contains the adapted pressure setpoint that is influenced by other functions, like compensation functions *(see page 77)* (using the flow value from the flow meter, if available *(see page 84)*) or the cavitation protection function *(see page 92)*. The value is constantly adjusted to the situation of the pumping application.

The table describes the output variables from the function block:

Output	Data Type	Description
`q_stFillSta_Hmi`	`stFillStat`	Structure that is used to display the status of the pipe fill function on the HMI. Refer to the structure data type description *(see page 174)*.
`q_stRef_FillRef`	`stRef`	Structure that contains the adjusted pressure value. This is the user-defined pressure setpoint that has been corrected by the pipe fill function. Refer to the structure data type description *(see page 175)*.
`q_stFillAlrm`	`stFillAlrm`	Structure that indicates the alarm states detected in pipe fill mode. Refer to the structure data type description *(see page 134)*.

Structure Data Type Definitions - PipeFill

Overview

Structure data types serve as the major input and output data carriers between different function blocks, configuration, and parameterization data.

The following section lists the names of structures (data types) that are used in this function block, followed by a table that describes the elements of structure.

NOTE: For the structure data types that are used from this function block and others, refer to the Common Structure Data Type Definitions chapter *(see page 166)*.

Structure: stFillInit

The stFillInit structure data type contains the initialization data for the PipeFill function block.

Element of stFillInit	Data Type	Description
rFillPres	REAL	This pipe fill pressure value (legend item 5 in the overview diagram of the start / restart procedure) *(see page 126)* is dedicated to the startup or restart procedure. It defines the pressure level that has to be reached to change from the first startup sequence to the second startup sequence. The second startup sequence uses the PumpPidStag function block *(see page 25)*.
tSequ1	TIME	Defines the time allowed for the first startup sequence. Range: tSequ1 > T#0s A timer is started when the PipeFill function block starts sequence 1. If the time defined with this parameter expires without reaching the pressure level defined with the parameter rFillPres, then the PipeFill function block indicates that an alarm has been detected. This is achieved by setting the output q_stFillAlrm.xSequ1Timeout to TRUE *(see page 134)*.

Element of `stFillInit`	Data Type	Description
`tSequ2`	TIME	Defines the time allowed for the second startup sequence. Range: `tSequ2` > `T#0s` A timer is started when the `PipeFill` function block starts sequence 2. If the time defined with this parameter expires without reaching the pressure working range (defined with the parameter `irSP` and the tolerance `rTnce`), then the `PipeFill` function block indicates that an alarm has been detected. This is achieved by setting the output `q_stFillAlrm.xSequ2Timeout` to TRUE *(see page 134)*.
`rTn`	REAL	Defines the stage time (Tn, in milliseconds) for the reference ramp during PID controlling in the second startup sequence. Range: `rTn` > 0.0 ms The time required to increase the pressure value by 1 unit.
`rTnce`	REAL	Defines the tolerance range. The tolerance is a ± value in %. It defines the range that the pressure may deviate from the user-defined pressure setpoint value (`i_rSp`). No detected alarm is indicated as long as the pressure is within this range. Range: 10 > `rTnce` ≥ 0 Default value: 5 (allowing a deviation of 5% from the pressure setpoint value (`i_rSp`))
`tPresAlrmDel`	TIME	Defines the time the pressure value has to be continuously out of the tolerance range before a detected alarm is indicated on the alarm output. Range: `tPresAlrmDel` > `T#10s` Default value: `T#30s`

NOTE: The elements of the structure are set to the default values (0 or FALSE) assigned by the controller on system startup.

Structure: stFillAlrm

The stFillAlrm structure data type is used to indicate the alarm states detected in pipe fill mode.

Element of stFillAlrm	Data Type	Description
xSequ1TimeOut	BOOL	When TRUE, a timeout has been detected in sequence 1 of the pipe fill mode.
xSequ2TimeOut	BOOL	When TRUE, a timeout has been detected in sequence 2 of the pipe fill mode.
xDiscPresSta	BOOL	When TRUE, the discharge pressure value is not available.
xLowPres	BOOL	When TRUE, it has been detected that the pressure has been lower than the defined working range for the delay time defined with stFillInit.tPresAlrmDel.
xHighPres	BOOL	When TRUE, it has been detected that the pressure has been higher than the defined working range for the delay time defined with stFillInit.tPresAlrmDel.

Section 1.14
OpIntPump: Operator Interface for Pump Handling

Overview

This section describes the OpIntPump function block representing the operator interface to the functions of this library.

If the pumping application is running in manual mode, the operator can directly modify the configuration parameters.

NOTE: This function block is designed to be used in conjunction with other function blocks of the Pumping Library.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description - OpIntPump	136
Function Block Description - OpIntPump	139
Structure Data Type Definitions - OpIntPump	143

Description - OpIntPump

Overview

The OpIntPump function block allows the operator to influence the pumping application via inputs on an HMI terminal. This application function block represents the interface from the pump application to the potential HMI. The handling and the architecture of the HMI application is not part of this document.

The figure illustrates the different inputs and outputs of the OpIntPump function block:

The `OpIntPump` function block executes the following tasks:

- It detects states and modes of the individual pumps, VSDs, and of the pumping application.
- It transforms states and modes of the individual pumps, VSDs, of the pump group, and of the pumping system into short integer values. This helps to save configuration times when assigning HMI symbols. During operation, it helps to optimize communication exchanges.
- It receives automatic commands for the VSDs, pumps, and for the auxiliary pump. If the pumping application is running in automatic mode, it forwards these commands consecutively to the outputs.
- It receives operator commands from the HMI and forwards them if the pumping application is running in automatic mode.
- It verifies the operating mode of the pumping application:
  - If it is running in manual mode, then it accepts only the commands from the HMI.
  - If it is running in automatic mode, then it accepts only the internal commands.
- When a pump or VSD is in hand or maintenance mode and is subsequently released, the `OpIntPump` function block reassigns the previous mode that was valid in the pumping application for this device.
- It allows the operator to assign the control of the application to a local or a remote HMI.

The graphic illustrates the relationship between the input and output pins of the `OpIntPump` function block and the information provided on the HMI.

HMI

Function Block Description - `OpIntPump`

Pin Diagram

Brief Description

The `OpIntPump` function block allows the operator to influence the pumping application via inputs on an HMI terminal.

I/O Variables Description

The table describes the input/output variables of the function block:

Input/Output	Data Type	Description
iq_stAlrmCtrl	stAlrmCtrl	Structure used for alarm handling. Refer to the structure data type description *(see page 166)*.
iq_stOpMode	stOpMode	Determines the operating modes. If this structure data type is set to xMan, the function block accepts and executes commands entered by the operator. Automatic commands are rejected. If this structure data type is set to xAuto, the function block verifies whether an error has been detected. If not, it forwards the commands consecutively to the output pins. If an error has been detected, (iq_stOpMode.xAlrm) the outputs of this function block are set to their defined fallback (default) states. Refer to the structure data type description *(see page 166)*.
iq_astMode_Vsd	ARRAY [0..7] of stMode	Array of stMode to control and switch the operating mode of the VSDs (maximum of 8). Refer to the structure data type description *(see page 169)*.
iq_astMode_VsPump	ARRAY [0..7] of stMode	Array of stMode to control and switch the operating mode of the VS pumps (maximum of 8). Refer to the structure data type description *(see page 169)*.
iq_astMode_FsPump	ARRAY [0..7] of stMode	Array of stMode to control and switch the operating mode of the FS pumps (maximum of 8). Refer to the structure data type description *(see page 169)*.
iq_stVsdCmd_Hmi	stVsdCmd	Structure used to control and switch the states of the VSDs. Refer to the structure data type description *(see page 143)*.
iq_stPumpCmd_Hmi	stPumpCmd	Structure used to control and switch the states of the main pumps. Refer to the structure data type description *(see page 143)*.
iq_stPlantCmd_Loc_Hmi	stPlantCmd	Structure used to control and switch the states of the machine application. This input/output variable is useful to control and switch the state of several pump groups from one local HMI. Refer to the structure data type description *(see page 145)*.

Input/Output	Data Type	Description
iq_stPlantCmd_Rem_Hmi	stPlantCmd	Structure used to control and switch the states of the machine application. This input/output variable is useful to control and switch the state of several pump groups from one remote HMI. Refer to the structure data type description *(see page 145)*.

The table describes the input variables to the function block:

Input	Data Type	Description
i_xRele	BOOL	Activates (TRUE) or stops (FALSE) the execution of the function block.
i_astVsdSta_Ex	ARRAY [0..7] of stVsdSta	Array of stVsdSta. Contains the external status information from 8 VSDs. Refer to the structure data type description *(see page 171)*.
i_astSta_VsPump_Ex	ARRAY [0..7] of stSta	Array of stSta. Contains the external status information from 8 VS pumps. Refer to the structure data type description *(see page 170)*.
i_astSta_FsPump_Ex	ARRAY [0..7] of stSta	Array of stSta. Contains the external status information from 8 FS pumps. Refer to the structure data type description *(see page 170)*.
i_stSwcInit	stSwcInit	The structure contains the number of pumps installed. Refer to the structure data type description *(see page 52)*.
i_stCmd_Vsd	stCmd	Structure used for switching the FS pumps into run state (maximum of 8). Refer to the structure data type description *(see page 169)*.
i_astLink_VsPump	ARRAY [0..7] of stLink	Array of stLink to link the VS pumps (maximum of 8) dynamically or constantly to one of the VSDs, or directly to the power source (direct online) by using contactors. Refer to the structure data type description *(see page 169)*.
i_stCmd_FsPump	stCmd	Array of stCmd for switching the FS pumps (maximum of 8) into run state. Refer to the structure data type description *(see page 169)*.

Input	Data Type	Description
i_stAna_CtrlVal	stAna	Structure that indicates the PID control value and the state of this value. Refer to the structure data type description *(see page 173)*.
i_arVsdSpd_Man	ARRAY [0...7] of REAL	The array of REAL contains the manual speed values of the VSDs (0..7).

The table describes the output variables from the function block:

Output	Data Type	Description
q_astVsdFbck_Hmi	ARRAY [0..7] of stVsdFbck	Array that is used to display the status and the modes of the VSDs (maximum of 8) on the HMI. Refer to the structure data type description *(see page 146)*.
q_astPumpFbck_Vs_Hmi	ARRAY [0..7] of stPumpFbck	Array that is used to display the status and the modes of the VS pumps (maximum of 8) on the HMI. Refer to the structure data type description *(see page 146)*.
q_astPumpFbck_Fs_Hmi	ARRAY [0..7] of stPumpFbck	Array that is used to display the status and the modes of the FS pumps (maximum of 8) on the HMI. Refer to the structure data type description *(see page 146)*.
q_stPlant_Sta_Hmi	stPlant	Structure that is used to display the status and the mode of the machine application on the HMI. Refer to the structure data type description *(see page 147)*.
q_stCmd_Vsd	stCmd	Structure that is used for switching the VS pumps into run state (maximum of 8). Refer to the structure data type description *(see page 146)*.
q_astLink_VsPump	ARRAY [0..7] of stLink	Array of stLink to link the VS pumps (maximum of 8) dynamically or constantly to one of the VSDs, or directly to the power source (direct online) by using contactors. Refer to the structure data type description *(see page 172)*.
q_stCmd_FsPump_Ex	stCmd	Structure used for switching the FS pumps into run state (maximum of 8). Refer to the structure data type description *(see page 169)*.
q_arVsdSpd_Ex	ARRAY [0..7] of REAL	Contains the speed values for the VSDs (maximum of 8) in percent.

Structure Data Type Definitions - OpIntPump

Overview

Structure data types serve as the major input and output data carriers between different function blocks, configuration, and parameterization data.

The following section lists the names of structures (data types) that are used in this function block, followed by a table that describes the elements of structure.

NOTE: For the structure data types that are used from this function block and others, refer to the Common Structure Data Type Definitions chapter *(see page 166)*.

Structure: stVsdCmd

The stVsdCmd structure data type is used to control and switch the states of the VSDs.

Element of stVsdCmd	Data Type	Description
siNb	SINT	Defines the number of VSDs that are available in the system. Range: 1...8
xAutoMode	BOOL	automatic mode When TRUE, the VSD is running in automatic mode without user intervention.
xManMode	BOOL	manual mode When TRUE, the VSD is running in manual mode, that means it is controlled by operator inputs.
xMainMode	BOOL	maintenance mode When TRUE, the VSD is running in maintenance mode and is not available for the application.

Structure: stPumpCmd

The stPumpCmd structure data type is used to control and switch the states of the pumps. It is available if the pumping application is running in manual mode (stOpMode.xMan = TRUE).

Element of stPumpCmd	Data Type	Description
siNb	SINT	Indicates the number of the pump on which the commands will be executed. Range: 1...8
esiCmdType	SINT / Enumeration	1 = esiCmdType.FixSpeed (Fixed Speed) 2 = esiCmdType.VariSpeed (Variable Speed)
xAutoMode	BOOL	When TRUE, the pump is switched to automatic mode.
xManMode	BOOL	When TRUE, the pump is switched to manual mode.

Element of `stPumpCmd`	Data Type	Description
`xMainMode`	BOOL	When TRUE, the pump is switched to maintenance mode.
`xStrt`	BOOL	When TRUE, the pump is switched into run state.
`xStop`	BOOL	When TRUE, the pump is switched into stop state.
`xByps`	BOOL	Used if fixed link operational mode is selected. When TRUE, the bypass contactor of the corresponding pump is selected (direct online mode).
`xVsd`	BOOL	Used if fixed link operational mode is selected. When TRUE, the contactor of the corresponding pump to link the VSD is selected.
`rManSp`	–	Manual pressure setpoint

The following table provides an example of the settings that must be set at the HMI to switch the variable speed pump 1 to run state:

Element of `stPumpCmd`	Value	Description	Type of Action
`siNb`	1	number of the pump	selection
`esiCmdType`	2	2 = VariSpeed	selection
`xAutoMode`	–	automatic mode	command
`xManMode`	–	manual mode	command
`xMainMode`	–	maintenance mode	command
`xStrt`	TRUE	start	command
`xStop`	–	stop	command
`xByps`	–	bypass	command
`xVsd`	–	VSD mode	command
`rManSp`	–	manual pressure setpoint	command

Structure: stPlantCmd

The OpIntPump function block uses the stPlantCmd structure data type as an input/output variable. The structure is used to control and switch the states of the machine application. This variable is used if there is a local and a remote HMI configured for controlling the pumping application.

The remote HMI control allows you to remotely monitor the application, to perform various maintenance activities, including modifications to data and configuration parameters, and to change the state of the application and associated hardware. Strict care must be taken to be sure that the immediate physical environment of the machine is in a state that will not present safety risks to people or property before exercising control remotely.

NOTE: The documentation of the programming, in particular the parameterization of the HMI, is not part of this document.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Configure and install the mechanism that enables the remote HMI local to the machine, so that local control over the machine can be maintained regardless of the remote commands sent to the application.
- Have a complete understanding of the application and the machine before attempting to control the application remotely.
- Take the precautions necessary to assure that you are operating remotely on the intended machine by having clear, identifying documentation within the application and its remote connection.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Element of stPlantCmd	Data Type	Description
xRem	BOOL	Command: switch to remote mode When TRUE, the OpIntPump function block switches the operating mode stOpMode.xRem to TRUE.
xLoc	BOOL	Command: switch to local mode When TRUE, the OpIntPump function block switches the operating mode stOpMode.xAuto to TRUE.
xAuto	BOOL	Command: switch to automatic mode When TRUE, the OpIntPump function block switches the operating mode stOpMode.xLoc to TRUE.
xMan	BOOL	Command: switch to manual mode When TRUE, the OpIntPump function block switches the operating mode stOpMode.xMan to TRUE.

See the related output variable `stPlant`.

Structure: `stVsdFbck`

The `OpIntPump` function block uses the `stVsdFbck` structure data type as an output variable. The `stVsdFbck` structure data type contains status bits designed to represent the status and mode of related VSDs on the HMI (for example, by using different colors to represent different states or modes).

Element of `stVsdFbck`	Data Type	Description
`siSta`	SINT	Indicates the status of a specific VSD. Range: Bit 0 = stop Bit 1 = run Bit 2 = alert detected Bit 3 = alarm detected Bit 4 = maintenance
`siMode`	SINT	Indicates the mode of a specific VSD. Range: Bit 0 = manual mode Bit 1 = automatic mode Bit 2 = maintenance mode

Structure: `stPumpFbck`

The `stPumpFbck` structure data type contains status bits designed to represent the status and mode of related pumps on the HMI (for example, by using different colors to represent different states or modes).

Element of `stPumpFbck`	Data Type	Description
`siSta`	SINT	Indicates the status of a specific pump. Range: Bit 0 = stop Bit 1 = run Bit 2 = alert detected Bit 3 = alarm detected Bit 4 = maintenance
`siMode`	SINT	Indicates the mode of a specific pump. Range: Bit 0 = manual mode Bit 1 = automatic mode Bit 2 = maintenance mode

Element of `stPumpFbck`	Data Type	Description
`siVsdNu`	SINT	Indicates which VSD is serving the VS pump. Range: Bit 0 = not used Bit 1 = VSD1 (with flexible operational mode enabled) Bit 2 = VSD2 (with flexible operational mode enabled) Bit 3 = VSD3 (with flexible operational mode enabled) Bit 4 = VSD4 (with flexible operational mode enabled) Bit 5 = bypass option is enabled Bit 6 = VSD is set to run state (with fixed link operational mode enabled)
`siType`	SINT	Indicates the type of a specific pump. Range: Bit 0 = VS pump bypassing the VSD or FS pump Bit 1 = VS pump

Structure: `stPlant`

The `stPlant` structure data type is used to indicate the status and the mode of the machine application on the HMI.

Element of `stPlant`	Data Type	Description
`siMode`	SINT	Indicates the mode of the machine application. Range: Bit 0 = controlled by remote HMI Bit 1 = controlled by local HMI
`siGrpMode`	SINT	Indicates the operating mode of the pump group. Range: Bit 0 = manual mode Bit 1 = automatic mode

Section 1.15

OpIntAuxPump: Auxiliary Pump Operator Interface Function

Overview

This section describes the OpIntAuxPump function block representing the operator interface to the functions of this library.

If the pumping application is running in manual mode, the operator can directly modify the run state.

NOTE: This function block is designed to be used in conjunction with other function blocks of the Pumping Library.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description - OpIntAuxPump	149
Function Block Description - OpIntAuxPump	152
Structure Data Type Definitions - OpIntAuxPump	154

Description - OpIntAuxPump

Overview

The OpIntAuxPump function block allows the operator to influence the pumping application via inputs on an HMI terminal. This application function block represents the interface from the pump application to the potential HMI. The handling and the architecture of the HMI application is not part of this document.

The figure illustrates the different inputs and outputs of the OpIntAuxPump function block:

The OpIntAuxPump function block executes the following tasks:

- It detects states and modes of the auxiliary pump.
- It transforms the state and the mode of the auxiliary pump into short integer values. This helps to save configuration times when assigning HMI symbols. During operation, it helps to optimize communication exchanges.

- It receives automatic commands for the auxiliary pump. If the pumping application is running in automatic mode, it forwards these commands consecutively to the output.
- It receives operator commands from the HMI and forwards them if the pumping application is running in manual mode.
- It verifies the operating mode of the pumping application:
  - If it is running in manual mode, then it accepts only the commands from the HMI.
  - If it is running in automatic mode, then it accepts only the controller commands.
- When the auxiliary pump is in hand or maintenance mode and is subsequently released, the `OpIntAuxPump` function block reassigns the previous mode that was valid in the pumping application for this device.
- It allows the operator to assign the control of the application to a local or a remote HMI.

The graphic illustrates the relationship between the input and output pins of the `OpIntAuxPump` function block and the information provided on the HMI.

Function Block Description - `OpIntAuxPump`

Pin Diagram

Brief Description

The `OpIntAuxPump` function block allows the operator to influence the auxiliary pump via inputs on an HMI terminal.

I/O Variables Description

The table describes the input/output variables of the function block:

Input/Output	Data Type	Description
`iq_stAlrmCtrl`	`stAlrmCtrl`	Structure used for alarm handling. Refer to the structure data type description *(see page 166)*.
`iq_stOpMode`	`stOpMode`	Determines the operating modes. If this structure data type is set to `xMan`, the function block accepts and executes commands entered by the operator. Automatic commands are rejected. If this structure data type is set to `xAuto`, the function block verifies whether an error has been detected. If not, it forwards the commands consecutively to the output pins. If an error has been detected (`iq_stOpMode.xAlrm`), the outputs of this function block are set to their defined fallback (default) states. Refer to the structure data type description *(see page 166)*.

Input/Output	Data Type	Description
iq_stMode_Aux	stMode	Structure to control and switch the operating mode of the auxiliary pump. Refer to the structure data type description *(see page 169)*.
iq_stPumpCmd_Aux_Hmi	stPumpCmd	Structure used to control and switch the states of the auxiliary pump. Refer to the structure data type description *(see page 143)*.

The table describes the input variables to the function block:

Input	Data Type	Description
i_xRele	BOOL	Activates (TRUE) or stops (FALSE) the execution of the function block.
i_stSta_AuxPumpEx	stSta	Structure that contains the status of the auxiliary pump. Refer to the structure data type description *(see page 170)*.
i_xFbckErr	BOOL	This pin is linked to the feedback error, detected by the AuxPumpCtrl function block *(see page 120)*.
i_stAuxPumpCom	stAuxPumpCom	The structure contains the command bit for the auxiliary pump inclusive of the state. If the state bit is set to TRUE the command bit is valid. Refer to the structure data type description *(see page 156)*.

The table describes the output variables from the function block:

Output	Data Type	Description
q_stPumpFbck_Aux_Hmi	stPumpFbck	The structure stPumpFbck is used to display the status and the modes of the auxiliary pump. Refer to the structure data type description *(see page 155)*.
q_stAuxPumpCom_Ex	stAuxPumpCom	The structure contains the command bit for the auxiliary pump inclusive of the state. If the state bit is set to TRUE the command bit is valid. Refer to the structure data type description *(see page 156)*.

Structure Data Type Definitions - OpIntAuxPump

Overview

Structure data types serve as the major input and output data carriers between different function blocks, configuration, and parameterization data.

The following section lists the names of structures (data types) that are used in this function block, followed by a table that describes the elements of structure.

NOTE: For the structure data types that are used from this function block and others, refer to the Common Structure Data Type Definitions chapter *(see page 166)*.

Structure: stPumpCmd

The stPumpCmd structure data type is used to control and switch the states of the pumps. This structure is used as a command interface from the OpIntAuxPump function block..

Element of stPumpCmd	Data Type	Description
siNb	SINT	Indicates the number of the pump on which the commands will be executed. Range: 1...4
ID	SINT	Constant = 1
xAutoMode	BOOL	When TRUE, the pump is switched to automatic mode.
xManMode	BOOL	When TRUE, the pump is switched to manual mode.
xMainMode	BOOL	When TRUE, the pump is switched to maintenance mode.
xStrt	BOOL	When TRUE, the pump is switched into run state.
xStop	BOOL	When TRUE, the pump is switched into stop state.
xByps	BOOL	Reserved.
xVsd	BOOL	Reserved.
rManSp	REAL	Reserved.

The following table provides an example of the settings that must be set at the HMI to switch pump 1 to run state:

Element of stPumpCmd	Value	Description	Type of Action
siNb	1	number of the pump	selection
ID	1	constant	selection
xAutoMode	–	automatic mode	command
xManMode	–	manual mode	command
xMainMode	–	maintenance mode	command

Element of stPumpCmd	Value	Description	Type of Action
xStrt	TRUE	start	command
xStop	–	stop	command
xByps	–	bypass	command
xVsd	–	VSD mode	command
rManSp	REAL	–	Not used

Structure: stPumpFbck

The stPumpFbck structure data type contains status bits designed to represent the status and mode of related pumps on the HMI (for example, by using different colors to represent different states or modes).

Element of stPumpFbck	Data Type	Description
siSta	SINT	Indicates the status of a specific pump. Range: Bit 0 = stop Bit 1 = run Bit 2 = alert detected Bit 3 = alarm detected Bit 4 = maintenance
siMode	SINT	Indicates the mode of a specific pump. Range: Bit 0 = manual mode Bit 1 = automatic mode Bit 2 = maintenance mode
siVsdNu	SINT	Indicates which VSD is serving the VS pump. Value of siVsdNu: 0 = not used 1 = VSD1 (with flexible operational mode enabled) 2 = VSD2 (with flexible operational mode enabled) 3 = VSD3 (with flexible operational mode enabled) 4 = VSD4 (with flexible operational mode enabled) 5 = bypass option is enabled 6 = VSD is set to run state (with fixed link operational mode enabled)
siType	SINT	Indicates the operating mode of the pump group. Range: Bit 0 = manual mode Bit 1 = automatic mode

Structure: stAuxPumpCom

The stAuxPumpCom structure data type contains the command bit to run the auxiliary pump and its state:

Element of stAuxPumpCom	Data Type	Description
stAuxPumpCom	BOOL	This element contains the command bit to run the auxiliary pump.
xSta	BOOL	If the state bit xSta is set to TRUE the command bit is valid.

Section 1.16
AnaIput: Analog Input Value Control

Overview

This section describes the AnaIput function block for detecting and controlling analog input values.

NOTE: This function block is designed to be used in conjunction with other function blocks of the Pumping Library. Although it is possible to use this function block independently, the use of the other function blocks of the library helps simplify programming this function block.

What Is in This Section?

This section contains the following topics:

Topic	Page
Description - AnaIput	158
Function Block Description - AnaIput	161
Structure Data Type Definitions - AnaIput	163

Description - `AnaIput`

Overview

This section describes the `AnaIput` function block for detecting and controlling analog input values.

It performs the following main tasks:

- Obtains the raw analog value that is provided by the pressure sensor or flow meter.
- Scales the raw analog value to provide a floating point number in to physical units.
- Detects a connection loss between the controller and the distributed I/O hardware module, if configured, and provides the status of the sensor or meter if a connection loss has been detected. The `AnaIput` function block is disabled and the function block outputs are set to the default values.
- Detects errors in configuration parameters.
- Detects a sensor error, for example, a broken wire, if the raw value provided by the sensor or meter is below the minimum value.
- Validates whether the analog raw value is within defined limits and, if not, issues alarms and alerts after a configurable time delay has elapsed.
- Limits the output value to a configurable maximum value.

Scaling Example

The following graphic illustrates the relationship between the analog raw value and the scaled value (floating point number).

It shows an example of an analog value (4 mA = 6553):

`wRawDigHigh`	maximum raw value
`wRawValLow`	minimum raw value
`rPhyValHigh`	maximum physical value (floating point number)
`rPhyValLow`	minimum physical value (floating point number)

Scaling Formula

The following formula is used for scaling the analog raw value:

y = mx+b

Variable	Definition
m	The values defined for the structure data type `iq_stAnaInit` are used to determine this constant. It is only established at a cold start. (`wRawDigHigh` – `wRawValLow`) / (`rPhyValHigh` – `rPhyValLow`) **NOTE:** The check of the parameters is only executed before the m value is calculated during the cold start of a controller.
b	offset `rPhyValLow`
x	input value

Broken Wire Detection

In the scaling example, the minimum scaling value of the raw value is greater than zero (for example, 6553 for the 4 to 20 mA sensor). This allows for broken wire detection: An alarm indicating a detected wire break is issued when the actual raw value is less than the configured minimum raw value (`wRawValLow`) minus 10%.

Detecting Alarms and Alerts

The `AnaIput` function block can detect and indicate alarms and alerts. To achieve this, you can define different limits with the structure data type `stAnaCtrl`.

If you do not want to use the alert function, enter higher values for the alert parameters than for the alarm parameters.

Function Block Description - `AnaIput`

Pin Diagram

Brief Description

The `AnaIput` function block is used for detecting and controlling analog input values.

I/O Variables Description

The table describes the in/output variable of the function block:

In/Output	Data Type	Description
`iq_stAnaInit`	`stAnaInit`	Contains the structure for initialization data. Refer to the structure data type description *(see page 163)*.

The table describes the input variables to the function block:

Input	Data Type	Description
i_stAnaCtrl	stAnaCtrl	Contains the input structure defining the limit and control values used. Refer to the structure data type description *(see page 163)*.
i_xNoComErr	BOOL	When TRUE, the connection between the controller and the distributed I/O hardware module providing the status of the analog pressure sensor or flow meter is available, given that the distributed I/O is configured for your application. If a connection loss is detected, the AnaIput function block is disabled and the function block outputs are set to the default value. In contrast to the other parameters, this parameter is by default set to TRUE. No error is detected if no distributed I/O hardware module is used in the application.
i_iAnaVal	INT	Contains the analog raw input value provided by the analog pressure sensor or flow meter.

The table describes the output variables from the function block:

Output	Data Type	Description
q_stAna	stAna	Contains the output structure for the scaled analog value and the status of this value. Refer to the structure data type description *(see page 168)*.
q_stAnaLim	stAnaLim	Contains the output structure for indicating whether the actual physical value has exceeded a defined limit. Refer to the structure data type description *(see page 164)*.
q_stAnaAlrm	stAnaAlrm	Contains the output structure identifying the type of alarm that has been detected. Refer to the structure data type description *(see page 165)*.
q_xAlrm	BOOL	When TRUE, indicates that an alarm has been detected by the AnaIput function block.

Structure Data Type Definitions - AnaIput

Overview

Structure data types serve as the major input and output data carriers between different function blocks, configuration, and parameterization data.

The following section lists the names of structures (data types) that are used in this function block, followed by a table that describes the elements of structure.

NOTE: For the structure data types that are used from this function block and others, refer to the Common Structure Data Type Definitions chapter *(see page 166)*.

NOTE: Select the limits carefully. Any improper values configured here are not detected automatically. They may lead to an interruption of the water flow.

NOTE: Thoroughly test all functions during verification and commissioning.

Structure: stAnaInit

The stAnaInit structure data type contains the initialization data for the AnaIput function block.

Element of stAnaInit	Data Type	Description
wRawValLow	WORD	Contains the minimum raw value.
wRawDigHigh	WORD	Contains the maximum raw value.
rPhyValLow	REAL	Defines the minimum level that is allowed for the scaled value (floating point number). Range: rPhyValLow ≥ 0.0
rPhyValHigh	REAL	Defines the maximum level that is allowed for the scaled value (floating point number). Range: rPhyValHigh > rPhyValLow

Structure: stAnaCtrl

The stAnaCtrl structure data type is used to define the limit and control values.

Element of stAnaCtrl	Data Type	Description
rHiHi	REAL	Defines the highest limit for the scaled value. If this limit is reached, the function block indicates a detected alarm. The function block is disabled and the outputs are set to the default value (0 and FALSE).
rHi	REAL	Defines a high limit for the scaled value. If this limit is reached, the library indicates a detected alert. If you do not want to use this alert function, enter a value that is bigger than the value for rHiHi.

Element of `stAnaCtrl`	Data Type	Description
`rLo`	REAL	Defines a low limit for the scaled value. If this limit is reached, the library indicates a detected alert. If you do not want to use this alert function, enter a value that is bigger than the value for `rLoLo`.
`rLoLo`	REAL	Defines the lowest limit for the scaled value. If this limit is reached, the function block indicates a detected alarm. The function block is disabled and the outputs are set to the default value (0 and FALSE).
`xLimAct`	BOOL	When TRUE, the scaled value is limited to the maximum value defined for the parameter `rHiHi` and to the minimum value defined for the parameter `rLoLo`.
`tDelAlrm`	TIME	Defines the time to elapse (in ms) before an alarm is issued indicating that the scaled value is above or below the configured limits.

Structure: `stAnaLim`

The `stAnaLim` structure data type is used to indicate whether the actual scaled value has exceeded a defined limit.

Element of `stAnaLim`	Data Type	Description
`xLimAlrmHigh`	BOOL	When TRUE, indicates that a high alarm has been detected. This means that the scaled value is higher than the value defined for the parameter `rHiHi`.
`xLimAlrtHigh`	BOOL	When TRUE, indicates that a high alert has been detected. This means that the scaled value is higher than the value defined for the parameter `rHi`.
`xLimAlrmLow`	BOOL	When TRUE, indicates that a low alarm has been detected. This means that the scaled value is lower than the value defined for the parameter `rLoLo`.
`xLimAlrtLow`	BOOL	When TRUE, indicates that a low alert has been detected. This means that the scaled value is lower than the value defined for the parameter `rLo`.

Structure: stAnaAlrm

The stAnaAlrm structure data type is used to indicate which type of alarms has been detected.

Element of stAnaAlrm	Data Type	Description
xParaErr	BOOL	When TRUE, indicates that an error in the configuration has been detected. This means that a parameter for a high limit is assigned a lower value than a parameter for a low limit or vice versa. Or the maximum raw value (wRawDigHigh) is assigned a higher value than 32767. After the detected error has been corrected, a cold start is required.
xWireBrake	BOOL	When TRUE, indicates that a wire break has been detected. This means that the actual raw value is 10% below the value configured for the minimum raw value (wRawValLow).
xLimAlrm	BOOL	When TRUE, indicates that the scaled value is higher or lower than the values defined for stAnaCtrl.rHiHi and stAnaCtrl.rLoLo.

Section 1.17
Common Structure Data Type Definitions

Common Structure Data Type Definitions

Overview

Structure data types serve as the major input and output data carriers between different function blocks, conveying input, configuration, and parameterization data.

The following sections list the names of the structure data types that are used in several function blocks of this library, followed by a table that describes the internal parameters that populate the structure.

Structure: `stAlrmCtrl`

The alarm control structure data type is used to indicate a detected alarm within the pump group and to reset detected alarms.

Element of `stAlrmCtrl`	Data Type	Description
`xAlrmRele`	BOOL	When FALSE, an alarm has been detected. The functions set the outputs to the default state. If a function detects a system level alarm, then the function sets this bit to FALSE. The pump group is stopped.
`xRstAlrm`	BOOL	Resets the detected alarms. When set to TRUE, the alarm values are set to zero, when the cause has been corrected.

NOTE: This alarm control structure data type needs to be defined only once per pump in the system.

Structure: `stOpMode`

The `stOpMode` structure data type controls / switches the operating mode of the pumping application.

If a function block detects an alarm state in the pump group, this function block automatically sets the operating mode to manual mode. This is achieved by setting the structure data type `stOpMode` to `xAuto` = FALSE and `xMan` = TRUE. At the same time, the function block indicates that an alarm has been detected by setting `xAlrm` to TRUE. Operator intervention is required to change from manual mode to another operating mode and to acknowledge the detected alarm.

Element of `stOpMode`	Data Type	Description
`xAlrm`	BOOL	alarm mode When TRUE, indicates that an alarm has been detected in the pump group during data processing.
`xAuto`	BOOL	automatic mode When TRUE, the pumping application is working in automatic mode without user intervention.
`xMan`	BOOL	manual mode When TRUE, the pumping application is working in manual mode, which means it is controlled by user inputs.
`xFill`	BOOL	pipe fill mode When TRUE, the pumping application is working in pipe fill mode. After start or restart of the pumping application the pipe fill function starts specific processes (as defined with the `PipeFill` function block *(see page 130)*) to reach the pressure working range as quickly as possible.
`xCavtMarg`	BOOL	cavitation margin mode When TRUE, the suction pressure is below the cavitation margin and the pumping application is running in cavitation margin mode as defined with the `CavtProt` function block *(see page 97)*.
`xCaviFlow`	BOOL	Reserved.
`xPidMan`	BOOL	PID manual mode When TRUE, the pumping application is running with a PID (proportional integral derivative) manual setpoint. This mode is set by the operator using the operator interface. In this mode, the `PumpPidStag` function block *(see page 25)* uses an alternative pressure setpoint manually entered by the operator.
`xPIDStop`	BOOL	PID stopped mode When TRUE, the PID has been stopped.
`xPIDAuto`	BOOL	PID automatic mode When TRUE, the PID is running in automatic mode.
`xPrim`	BOOL	Reserved.

Element of stOpMode	Data Type	Description
xSle	BOOL	sleep mode When TRUE, the pumping application is working in sleep mode in times of low water demand in order to avoid using the main pumps (VS pumps or FS pumps). The sleep mode is started by the PumpPidStag function block *(see page 25)*. It is stopped either by the PumpPidStag function block or by the AuxPumpCtrl function block *(see page 120)*.
xRem	BOOL	remote mode When TRUE, disables the local HMI and enables the remote HMI to control the application.
xLoc	BOOL	local mode When TRUE, disables the remote HMI and enables the local HMI to control the application.

NOTE: This alarm control structure data type needs to be defined only once per pump in the system.

Structure: stAna

The stAna structure data type is used for the analog input value from the pressure sensor (measuring the pressure) or the flow meter (measuring the flow).

Element of stAna	Data Type	Description
rAnaVal	REAL	Scaled (physical units) analog value. The pressure / suction pressure value used as an input by the PipeFill function block *(see page 130)* and CavtProt) function block *(see page 97)*.
xStat	BOOL	When TRUE, indicates that the value of rAnaVal is valid.

Structure: stAvai

The stAvai structure data type is used to indicate the availability of 8 devices. This variable is used for VSDs, VS pumps, and FS pumps.

Element of stAvai	Data Type	Description
xAvai1	BOOL	When TRUE, indicates that device 1 is available.
...	...	...
xAvai8	BOOL	When TRUE, indicates that device 8 is available.

Structure: stStagVal

The stStagVal structure data type contains the stage value and its state. The stage value represents the demand for the number of pumps which are needed to maintain the required pressure level.

Element of stStagVal	Data Type	Description
siStag	SINT	Indicates the number of pumps which are needed to maintain the required pressure level.
xSta	BOOL	When TRUE, indicates that no error has been detected in the pump.

Structure: stCmd

The stCmd structure data type is used to switch the 8 devices (VSDs or FS pumps) into run or stop state.

Element of stCmd	Data Type	Description
xDev1	BOOL	When TRUE, switches device 1 into run state.
...	...	...
xDev8	BOOL	When TRUE, switches device 8 into run state.

Structure: stMode

The stMode structure data type controls / switches the modes of the individual pumps and VSDs.

Element of stMode	Data Type	Description
xRem	BOOL	remote mode When TRUE, the pump is locked for control by an HMI at a remote station.
xLoc	BOOL	local mode When TRUE, the pump is locked for control by the local HMI.
xHand	BOOL	hand mode When TRUE, the pump is locked for inputs that are made directly at the pump. Any commands from HMIs are blocked.
xMain	BOOL	maintenance mode When TRUE, the pump is in maintenance mode and is not available for the application.
xAuto	BOOL	automatic mode When TRUE, the pump is operated in automatic mode without user intervention.

Element of stMode	Data Type	Description
xManu	BOOL	manual mode When TRUE, the pump is controlled by user inputs.
xVsd	BOOL	VSD mode When TRUE, the VS pump is fixed linked to a VSD.
xByps	BOOL	bypass When TRUE, the VS pump bypasses the VSD because the VSD, to which the VS pump had been linked, is not available.
xAlrm	BOOL	alarm mode When TRUE, an alarm has been detected in the pump.
xAlrt	BOOL	Reserved.

Structure: stSta

The stSta structure data type indicates the status of the individual pumps as indicated by external devices (key switches, sensors, contactors).

Element of stSta	Data Type	Description
xHandMod	BOOL	hand mode When TRUE, the pump has been locked for operation in hand mode by a key switch. It is not available for the pumping application.
xMain	BOOL	maintenance mode When TRUE, the pump has been locked for operation in maintenance mode by a key switch. It is not available for the pumping application.
xDryRun	BOOL	dry run When TRUE, a sensor has indicated that the pump is running dry. It is not available for the pumping application.
xTempAlrm	BOOL	high temperature When TRUE, a sensor has indicated that the temperature in the pump is too high. It is not available for the pumping application.
xErr	BOOL	hardware error detected When TRUE, a hardware error has been detected in the pump. It is not available for the pumping application.

Element of `stSta`	Data Type	Description
`xLinkOk`	BOOL	bus link OK When TRUE, the data connection to the distributed I/O hardware module which provides status information on the pump is available. In contrast to the other parameters, this parameter is by default set to TRUE. No error is detected if no distributed I/O hardware module is used in the application.
`xRun`	BOOL	pump run When TRUE, a contactor has indicated that the pump has been switched in run status.
`xStop`	BOOL	pump stop When TRUE, a contactor has indicated that the pump has been switched in stop status.

Structure: `stVsdSta`

The `stVsdSta` structure data type indicates the status of the individual VSDs as indicated by the VSD and/or external devices (key switches, sensors, contactors).

Element of `stVsdSta`	Data Type	Description
`xPowerOk`	BOOL	VSD power OK When TRUE, the power supply for the VSD is available. In contrast to the other parameters, this parameter is by default set to TRUE. No error is detected if no power supply detection is used in the application.
`xLinkOk`	BOOL	bus link OK When TRUE, the data connection to the distributed I/O hardware module which provides status information on the VSD is available. In contrast to the other parameters, this parameter is by default set to TRUE. No error is detected if no distributed I/O hardware module is used in the application.
`xAlarm`	BOOL	detected alarm state When TRUE, an alarm has been detected for the VSD. It is not available for the pumping application.
`xRun`	BOOL	VSD run When TRUE, indicates that the VSD is in run state.

Structure: stLink

The stLink structure data type is used to link one VS pump dynamically to one of the up to 8 possible VSDs or constantly to one of the 4 VSDs or directly to the power source (direct online) by switching the respective contactors (see examples below).

Element of stLink	Data Type	Description
xVsd1	BOOL	When TRUE, the VS pump is linked to VSD 1.
xVsd2	BOOL	When TRUE, the VS pump is linked to VSD 2.
xVsd3	BOOL	When TRUE, the VS pump is linked to VSD 3.
xVsd4	BOOL	When TRUE, the VS pump is linked to VSD 4.
xByp5	BOOL	When TRUE, the VS pump bypasses the VSD. In this mode, the VS pump is directly linked to the power source (direct online).
xRun	BOOL	When TRUE, the VS pump is constantly linked to a VSD. This is only used in fixed link operational mode.

Example of using the stLink structure data type with variable links (fixed link operational mode disabled):

Example of using the `stLink` structure data type with fixed links (fixed link operational mode enabled):

Structure: `stAuxPumpCom`

The `stAuxPumpCom` structure data type is used to switch the auxiliary pump. The structure contains a binary element to switch the auxiliary pump into run state and a binary element to indicate the status of this procedure.

Element of `stAuxPumpCom`	Data Type	Description
`xAuxPumpRunCom`	BOOL	When TRUE, the auxiliary pump is switched into run state.
`xSta`	BOOL	When TRUE, the `xAuxPumpRunCom` command is valid.

Structure: stFillStat

The stFillStat structure data type is used to indicate the state of the pipe fill function.

Element of stFillStat	Data Type	Description
xFill	BOOL	When TRUE, indicates that the pipe fill function is in progress.
xFillOk	BOOL	When TRUE, indicates that the pressure value has reached the defined working range. **NOTE:** If the pipe fill function is not used, the parameter stFillStat.xFillOk has to be set to TRUE to enable the PID and stage functions of the PumpPidStag function block *(see page 25)*.
xSequ1	BOOL	When TRUE, the pipe fill function is running in sequence 1.
xSequ2	BOOL	When TRUE, the pipe fill function is running in sequence 2.
xAlrm	BOOL	alarm mode When TRUE, indicates that an alarm has been detected in the pipe fill function.

Structure: stCavtSta

The stCavtSta structure data type is used to indicate the state of the cavitation protection function.

Element of stCavtSta	Data Type	Description
xProtActv	BOOL	When TRUE, indicates that the cavitation protection function is in progress.
xSpRedu	BOOL	When TRUE, indicates that the setpoint of the control pressure has been reduced because the suction pressure is dropping.
xRstPos	BOOL	When TRUE, indicates that a reset of the cavitation alarm can be executed because the suction pressure is above the minimal setpoint value (legend item 5 in the diagram representing the working principle of the cavitation protection procedure *(see page 93)*).
xGdntPlus	BOOL	When TRUE, indicates that the suction pressure is rising (suction pressure gradient is positive).
xGdntMnus	BOOL	When TRUE, indicates that the suction pressure is dropping (suction pressure gradient is negative).

Structure: stSp

The stSp structure data type is used to indicate a setpoint value.

Element of stSp	Data Type	Description
rVal	REAL	Indicates a reference value.
xsta	BOOL	When TRUE, indicates that the value of rVal is valid.

Structure: stDeviceAlrm

The stDeviceAlrm structure data type is used to indicate detected errors of the (maximum 8) FS or VS pumps or VSDs.

Element of stDeviceAlrm	Data Type	Description
xDev1Alrm	BOOL	When TRUE, an error has been detected for pump 1 / VSD 1.
...	...	...
xDev8Alrm	BOOL	When TRUE, an error has been detected for pump 8 / VSD 8.

Structure: stRespErr

The stRespErr structure data type is used to indicate a detected error in the FS or VS pumps or VSDs (maximum of 8) because the time defined with the parameter i_tDelFbck has expired without receiving a response from the pump / VSD.

Element of stRespErr	Data Type	Description
xDev1	BOOL	When TRUE, an error has been detected for pump 1 / VSD 1 because no response has been received within the defined time.
...	...	...
xDev8	BOOL	When TRUE, an error has been detected for pump 8 / VSD 8 because no response has been received within the defined time.

Structure: abyVsdNu

The array of BYTE abyVsdNu is used to contain the number of the VSD associated with a given pump. The index to the array indicates the VS pump number.

Element of abyVsdNu	Data Type	Description
abyVsdNu	ARRAY [0..7] of BYTE	Defines the VSD numbers. Range: ● 0 = unused ● 1...4 = VSD numbers ● 5 = bypass ● 6 = run (fixed link operational mode)

Structure: asiPrty

The asiPrty is used to contain the priority assignment:

Element of asiPrty	Data Type	Description
asiPrty	ARRAY [0..7] of SINT	The array of short integer is used to list in sequence the priority of the VSDs or pumps as defined by the operating hours or by operator inputs: The VSD / pump that is the first in the list is assigned the highest priority.

Structure: stDiscPres

The stDiscPres contains 2 elements, one of which is a structure. stAna described below, contains the primary analog value. The second element is the value of the optional secondary pressure sensor. The secondary analog value is used, assuming it is available in your system, only in the case that the primary analog value is not valid.

Element of stDiscPres	Data Type	Description
stAna	stAna	Refer to the stAna structure *(see page 168)*.
rSurr	REAL	Analog input value provided by a second pressure sensor via the AnaIput function block *(see page 161)*.

Structure: `stRef`

The `stRef` structure data type is used to indicate a reference value. This user-defined data type has been used by compensation functions (*CompSp: Setpoint Adaptation (see page 76)*, and *CompSpFlow: Setpoint Adaptation by Using Flow Value (see page 83)*), by the pipe fill function *(see page 125)* and/or the cavitation protection function *(see page 91)*.

Element of `stRef`	Data Type	Description
`rVal`	REAL	Indicates a reference value.
`xsta`	BOOL	When TRUE, indicates that the value of `rVal` is valid.

Appendices

Appendix A
Function and Function Block Representation

Overview

Each function can be represented in the following languages:
- IL: Instruction List
- ST: Structured Text
- LD: Ladder Diagram
- FBD: Function Block Diagram
- CFC: Continuous Function Chart

This chapter provides functions and function blocks representation examples and explains how to use them for IL and ST languages.

What Is in This Chapter?

This chapter contains the following topics:

Topic	Page
Differences Between a Function and a Function Block	178
How to Use a Function or a Function Block in IL Language	179
How to Use a Function or a Function Block in ST Language	182

Differences Between a Function and a Function Block

Function

A function:

- is a POU (Program Organization Unit) that returns one immediate result.
- is directly called with its name (not through an instance).
- has no persistent state from one call to the other.
- can be used as an operand in other expressions.

Examples: boolean operators (`AND`), calculations, conversion (`BYTE_TO_INT`)

Function Block

A function block:

- is a POU (Program Organization Unit) that returns one or more outputs.
- needs to be called by an instance (function block copy with dedicated name and variables).
- each instance has a persistent state (outputs and internal variables) from one call to the other from a function block or a program.

Examples: timers, counters

In the example, `Timer_ON` is an instance of the function block `TON`:

```
1  PROGRAM MyProgram_ST
2  VAR
3      Timer_ON: TON;  // Function Block Instance
4      Timer_RunCd: BOOL;
5      Timer_PresetValue: TIME := T#5S;
6      Timer_Output: BOOL;
7      Timer_ElapsedTime: TIME;
8  END_VAR
```

```
1  Timer_ON(
2      IN:=Timer_RunCd,
3      PT:=Timer_PresetValue,
4      Q=>Timer_Output,
5      ET=>Timer_ElapsedTime);
```

How to Use a Function or a Function Block in IL Language

General Information

This part explains how to implement a function and a function block in IL language.

Functions `IsFirstMastCycle` and `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in IL Language

This procedure describes how to insert a function in IL language:

Step	Action
1	Open or create a new POU in Instruction List language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see SoMachine, Programming Guide)*.
2	Create the variables that the function requires.
3	If the function has 1 or more inputs, start loading the first input using LD instruction.
4	Insert a new line below and: ● type the name of the function in the operator column (left field), or ● use the **Input Assistant** to select the function (select **Insert Box** in the context menu).
5	If the function has more than 1 input and when Input Assistant is used, the necessary number of lines is automatically created with `???` in the fields on the right. Replace the `???` with the appropriate value or variable that corresponds to the order of inputs.
6	Insert a new line to store the result of the function into the appropriate variable: type ST instruction in the operator column (left field) and the variable name in the field on the right.

To illustrate the procedure, consider the Functions `IsFirstMastCycle` (without input parameter) and `SetRTCDrift` (with input parameters) graphically presented below:

Function	Graphical Representation
without input parameter: `IsFirstMastCycle`	
with input parameters: `SetRTCDrift`	

In IL language, the function name is used directly in the operator column:

Function	Representation in POU IL Editor
IL example of a function without input parameter: `IsFirstMastCycle`	1 `PROGRAM MyProgram_IL` 2 `VAR` 3 `FirstCycle: BOOL;` 4 `END_VAR` 5 1 `IsFirstMastCycle` `ST` `FirstCycle`

IL example of a function with input parameters: `SetRTCDrift`	1 `PROGRAM MyProgram_IL`
	2 `VAR`
	3 `myDrift: SINT (-29..29) := 5;`
	4 `myDay: DAY_OF_WEEK := SUNDAY;`
	5 `myHour: HOUR := 12;`
	6 `myMinute: MINUTE;`
	7 `myDiag: RTCSETDRIFT_ERROR;`
	8 `END_VAR`
	9
	
	1 `LD` `myDrift`
	`SetRTCDrift` `myDay`
	`myHour`
	`myMinute`
	`ST` `myDiag`

Using a Function Block in IL Language

This procedure describes how to insert a function block in IL language:

Step	Action
1	Open or create a new POU in Instruction List language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see SoMachine, Programming Guide)*.
2	Create the variables that the function block requires, including the instance name.
3	Function Blocks are called using a `CAL` instruction: ● Use the **Input Assistant** to select the FB (right-click and select **Insert Box** in the context menu). ● Automatically, the `CAL` instruction and the necessary I/O are created. Each parameter (I/O) is an instruction: ● Values to inputs are set by "`:=`". ● Values to outputs are set by "`=>`".
4	In the `CAL` right-side field, replace `???` with the instance name.
5	Replace other `???` with an appropriate variable or immediate value.

To illustrate the procedure, consider this example with the `TON` Function Block graphically presented below:

Function Block	Graphical Representation
`TON`	

In IL language, the function block name is used directly in the operator column:

Function Block	Representation in POU IL Editor

TON	1 PROGRAM MyProgram_IL
	2 VAR
	3 Timer_ON: TON; // Function Block instance declaration
	4 Timer_RunCd: BOOL;
	5 Timer_PresetValue: TIME := T#5S;
	6 Timer_Output: BOOL;
	7 Timer_ElapsedTime: TIME;
	8 END_VAR
	9
	
	1 CAL Timer_ON(
	IN:= Timer_RunCd,
	PT:= Timer_PresetValue,
	Q=> Timer_Output,
	ET=> Timer_ElapsedTime)

How to Use a Function or a Function Block in ST Language

General Information

This part explains how to implement a Function and a Function Block in ST language.

Function `SetRTCDrift` and Function Block `TON` are used as examples to show implementations.

Using a Function in ST Language

This procedure describes how to insert a function in ST language:

Step	Action
1	Open or create a new POU in Structured Text language. **NOTE:** The procedure to create a POU is not detailed here. For more information, refer to Adding and Calling POUs *(see SoMachine, Programming Guide)*.
2	Create the variables that the function requires.
3	Use the general syntax in the **POU ST Editor** for the ST language of a function. The general syntax is: `FunctionResult:= FunctionName(VarInput1, VarInput2,.. VarInputx);`

To illustrate the procedure, consider the function `SetRTCDrift` graphically presented below:

Function	Graphical Representation
`SetRTCDrift`	

The ST language of this function is the following:

Function	Representation in POU ST Editor

`SetRTCDrift`	`PROGRAM MyProgram_ST`
	`VAR myDrift: SINT(-29..29) := 5;`
	`myDay: DAY_OF_WEEK := SUNDAY;`
	`myHour: HOUR := 12;`
	`myMinute: MINUTE;`
	`myRTCAdjust: RTCDRIFT_ERROR;`
	`END_VAR`
	`myRTCAdjust:= SetRTCDrift(myDrift, myDay, myHour, myMinute);`

Using a Function Block in ST Language

This procedure describes how to insert a function block in ST language:

Step	Action
1	Open or create a new POU in Structured Text language. **NOTE:** The procedure to create a POU is not detailed here. For more information on adding, declaring and calling POUs, refer to the related documentation *(see SoMachine, Programming Guide)*.
2	Create the input and output variables and the instance required for the function block: ● Input variables are the input parameters required by the function block ● Output variables receive the value returned by the function block
3	Use the general syntax in the **POU ST Editor** for the ST language of a Function Block. The general syntax is: `FunctionBlock_InstanceName(Input1:=VarInput1, Input2:=VarInput2,... Ouput1=>VarOutput1, Ouput2=>VarOutput2,...);`

To illustrate the procedure, consider this example with the `TON` function block graphically presented below:

Function Block	Graphical Representation
`TON`	

This table shows examples of a function block call in ST language:

Function Block	Representation in POU ST Editor

TON	1 PROGRAM MyProgram_ST
	2 VAR
	3 Timer_ON: TON; // Function Block Instance
	4 Timer_RunCd: BOOL;
	5 Timer_PresetValue: TIME := T#5S;
	6 Timer_Output: BOOL;
	7 Timer_ElapsedTime: TIME;
	8 END_VAR
	
	1 Timer_ON(
	2 IN:=Timer_RunCd,
	3 PT:=Timer_PresetValue,
	4 Q=>Timer_Output,
	5 ET=>Timer_ElapsedTime);

Glossary

A

application

A program including configuration data, symbols, and documentation.

B

bypass

If a variable speed (VS) pump is operated without a variable speed drive (VSD), because there is no VSD available, it can bypass the VSD. For this, the pump is directly connected to the power source and its speed cannot be changed.

byte

A type that is encoded in an 8-bit format, ranging from 00 hex to FF hex.

C

cavitation

The process of cavities being produced in a liquid due to rapid changes of pressure. The implosion of these *bubbles* in pumps may cause significant wear or may even damage them. This is why cavitation situations should be avoided for pumps.

CFC

(*continuous function chart*) A graphical programming language (an extension of the IEC 61131-3 standard) based on the function block diagram language that works like a flowchart. However, no networks are used and free positioning of graphic elements is possible, which allows feedback loops. For each block, the inputs are on the left and the outputs on the right. You can link the block outputs to the inputs of other blocks to create complex expressions.

configuration

The arrangement and interconnection of hardware components within a system and the hardware and software parameters that determine the operating characteristics of the system.

controller

Automates industrial processes (also known as programmable logic controller or programmable controller).

E

expansion bus

An electronic communication bus between expansion I/O modules and a controller.

F

FB

(*function block*) A convenient programming mechanism that consolidates a group of programming instructions to perform a specific and normalized action, such as speed control, interval control, or counting. A function block may comprise configuration data, a set of internal or external operating parameters and usually 1 or more data inputs and outputs.

fixed link operational mode

In this mode, each variable speed drive is constantly linked to 1 variable speed pump.

flexible operational mode

This mode of the pumping application is only available if there are no FS pumps configured and if the number of VS pumps is greater than the number of VSDs. The bypass option is enabled and the fixed link operational mode is disabled by software parameters.The flexible operational mode is the mode with the greatest availability of pumps because the VS pumps can be used independently of VSDs.

FS pump

(*fixed speed pump*) A pump that is directly connected to the power source. The speed of this pump cannot be changed.

function block diagram

One of the 5 languages for logic or control supported by the standard IEC 61131-3 for control systems. Function block diagram is a graphically oriented programming language. It works with a list of networks where each network contains a graphical structure of boxes and connection lines representing either a logical or arithmetic expression, the call of a function block, a jump, or a return instruction.

H

HMI

(*human machine interface*) An operator interface (usually graphical) for human control over industrial equipment.

I

I/O

(*input/output*)

IL

(*instruction list*) A program written in the language that is composed of a series of text-based instructions executed sequentially by the controller. Each instruction includes a line number, an instruction code, and an operand (refer to IEC 61131-3).

INT

(*integer*) A whole number encoded in 16 bits.

L

LD

(*ladder diagram*) A graphical representation of the instructions of a controller program with symbols for contacts, coils, and blocks in a series of rungs executed sequentially by a controller (refer to IEC 61131-3).

N

NPSH

(*net positive suction head*) The minimum pressure energy that has to be available at the entry point of a pump to help prevent the liquid from boiling and thus to help prevent acavitation situation.]]

The NPSH value is provided by manufacturers of pumps throughout Europe.]

The NPSH+ value is provided by manufacturers of pumps throughout the USA.

P

PID

(*proportional, integral, derivative*) A generic control loop feedback mechanism (controller) widely used in industrial control systems.

POU

(*program organization unit*) A variable declaration in source code and a corresponding instruction set. POUs facilitate the modular re-use of software programs, functions, and function blocks. Once declared, POUs are available to one another.

program

The component of an application that consists of compiled source code capable of being installed in the memory of a logic controller.

S

ST

(*structured text*) A language that includes complex statements and nested instructions (such as iteration loops, conditional executions, or functions). ST is compliant with IEC 61131-3.

V

variable

A memory unit that is addressed and modified by a program.

VS pump

(*variable speed pump*) A pump that is linked to a variable speed drive. The variable speed drive controls the speed of the pump.

VSD

(*variable speed drive*) An equipment that makes a variable and regulates the speed and rotational force, or torque output, of an electric motor.

Index

A

AnaInput
    function block, *158*
AuxPumpCtrl
    function block, *116*

C

cavitation protection, *92*
CavtProt
    function block, *92*
CompSp
    function block, *77*
CompSpFlow
    function block, *84*

D

DevSwcPumpCtrl
    function block, *40*

F

FsPumpCtrl
    function block, *73*
function block
    AnaInput, *158*
    AuxPumpCtrl, *116*
    CavtProt, *92*
    CompSp, *77*
    CompSpFlow, *84*
    DevSwcPumpCtrl, *40*
    FsPumpCtrl, *73*
    OpIntAuxPump, *149*
    OpIntPump, *136*
    OpPrty, *110*
    PipeFill, *126*
    PumpAvai, *102*
    PumpPidStag, *18*
    VsdAvai, *106*
    VsdCtrl, *65*
    VsPumpCtrl, *69*
functions
    differences between a function and a function block, *178*
    how to use a function or a function block in IL language, *179*
    how to use a function or a function block in ST language, *183*

O

OpIntAuxPump
    function block, *149*
OpIntPump
    function block, *136*
OpPrty
    function block, *110*

P

PipeFill
    function block, *126*
PumpAvai
    function block, *102*
pumping
    AnaInput, *158*
    AuxPumpCtrl, *116*
    CavtProt, *92*
    CompSp, *77*
    CompSpFlow, *84*
    DevSwcPumpCtrl, *40*
    FsPumpCtrl, *73*
    OpIntAuxPump, *149*
    OpIntPump, *136*
    OpPrty, *110*
    PipeFill, *126*
    PumpAvai, *102*
    PumpPidStag, *18*
    VsdAvai, *106*
    VsdCtrl, *65*
    VsPumpCtrl, *69*
PumpPidStag
    function block, *18*

V

VsdAvai
    function block, *106*
VsdCtrl
    function block, *65*
VsPumpCtrl
    function block, *69*