SoMachine HVAC

Application Function
HVAC Library Guide

02/2017

EIO0000002057.02

www.schneider-electric.com

Schneider Electric

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.

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.

Table of Contents

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 functions of the HVAC Library.

Validity Note

This document has been updated for the release of SoMachine HVAC 2.2.

The function blocks described in this document are compatible with the Modicon M171 / M172 logic controllers.

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.

Standard	Description
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*.

Part I
HVAC&R Function Blocks

What Is in This Part?

This part contains the following chapters:

Chapter 1
AHU Plant Mode Strategy: AHUPlantModeStrategy

What Is in This Chapter?

This chapter contains the following sections:

Section 1.1
Functional and Machine Overview

What Is in This Section?

This section contains the following topics:

Functional Overview

Functional Description

The `AHUPlantModeStrategy` function block is a control strategy function block and defines the operation mode of the air handling unit. This function block is used in conjunction with the other air handling unit function block `AHUTempCntrlStrategy`.

Why Use the `AHUPlantModeStrategy` Function Block?

The `AHUPlantModeStrategy` function block is used for the following purposes:

Purpose	Description
Improve comfort conditions	improve the comfort of occupants by means of scheduled operation.
Reduce energy consumption	reduce energy consumption by means of the operating modes: ● night cycle ● night purge
Handle alarm situations	● freeze alarm ● fan alarm ● fire alarm

Features of the `AHUPlantModeStrategy` Function Block

The `AHUPlantModeStrategy` function block provides the following features:

- cools down the building with cool outdoor air during summer periods (when building is unoccupied)
- the `AHUPlantModeStrategy` function block optimizes the AHU operation when the building is occupied, at night times and in alarm situations (freeze and fan alarms)
- helps reduce cooling energy
- helps maintain consistent machine operation
- supports the following operating modes:
  Night Cycle - The room temperature reduces (during heating season) or increases (during cooling season) to reduce energy consumption during unoccupied periods.
  Night Purge - The cool outside air is used to cool down the building during the night.
- Power up initialization delay: the Plant mode will be initialized with 0.
  It can change the value from 0 into the values 2, 3, 4 or 5 only once the `uiInitializationDelay` period has elapsed.
  This feature can be disabled by entering the value 0 for the parameter `uiInitializationDelay`.
- The AHU is switched off after a delay.
  This feature can be disabled by entering the value 0 for the parameter `uiAHUOffDelay`.

Error Avoidance Features

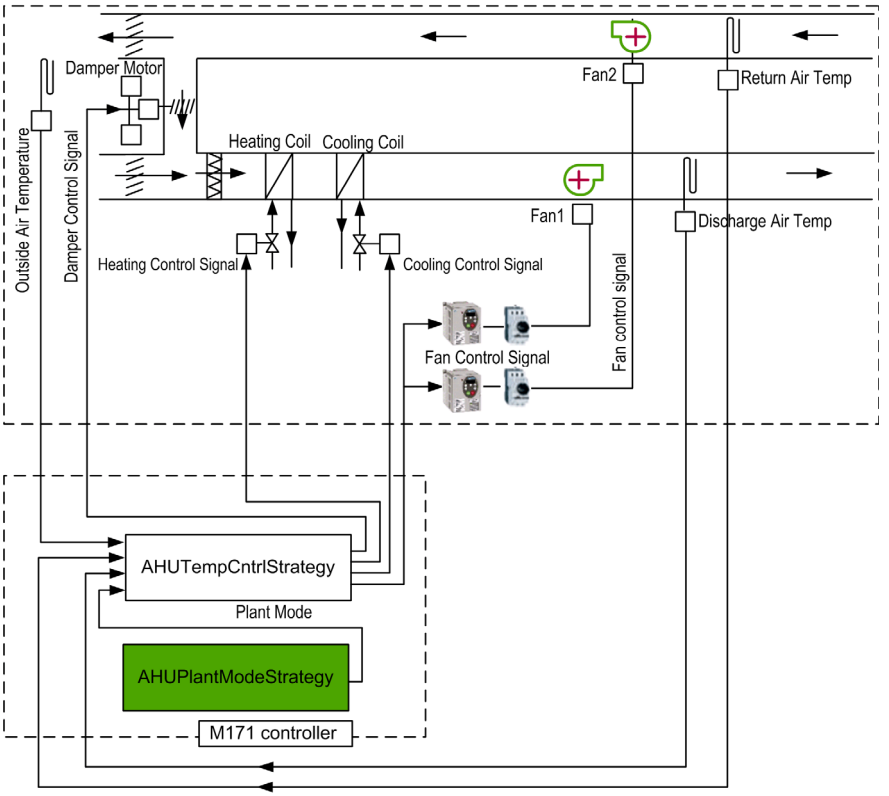
The `AHUPlantModeStrategy` function block provides the following error avoidance features to help you avoid the potentials of certain sources of machine malfunction:

Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.
Alarm/alert notification	If an invalid value is entered, an alarm or an alert is generated: ● Alarm: the compressors are switched off and the function block terminates in error. ● Alert: the function block keeps on operating, however with the possibility of reduced performance.
Controlled parameter	Parameters like `uiNightpurgeTimeStart`, `uiNightpurgeTimeStop` are controlled. The configuration of these parameters can be changed, however the changes are effective only after the restart of the function block.

Machine Overview

Machine View

The following picture presents the interaction between the function block and the machine:



AHUTempCntrlStrategy controls the discharge air temperature by sequencing fan speeds, heating coil, dampers and cooling coil.

AHUPlantModeStrategy is connected to the `AHUTempCntrlStrategy` and controls the operation modes of the Air Handling Unit.

Section 1.2
Architecture

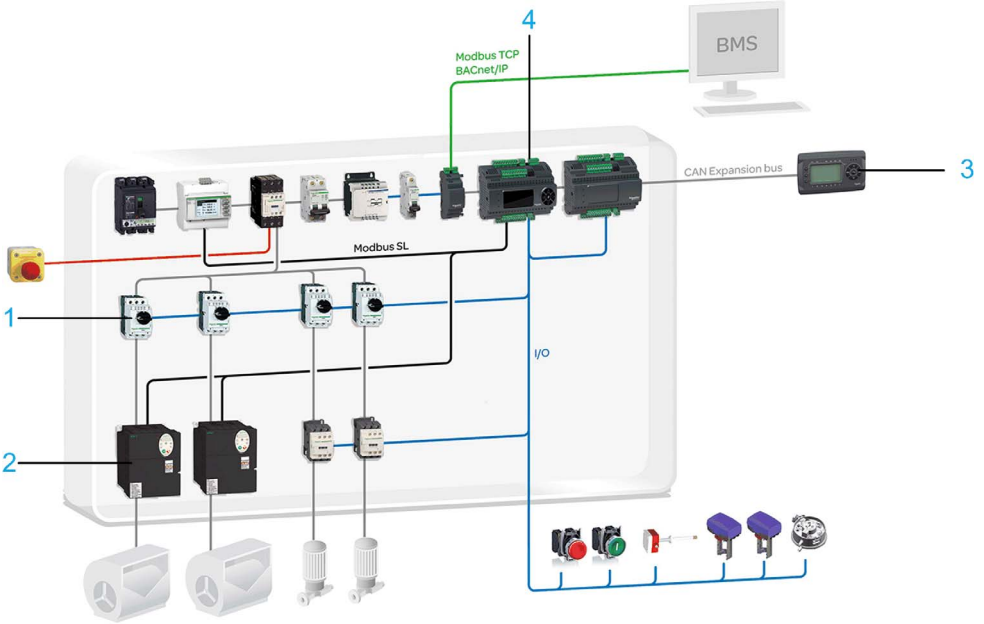
What Is in This Section?

This section contains the following topics:

Hardware Architecture

Hardware Architecture Overview

The following figure presents the hardware architecture of Air Handling Unit.

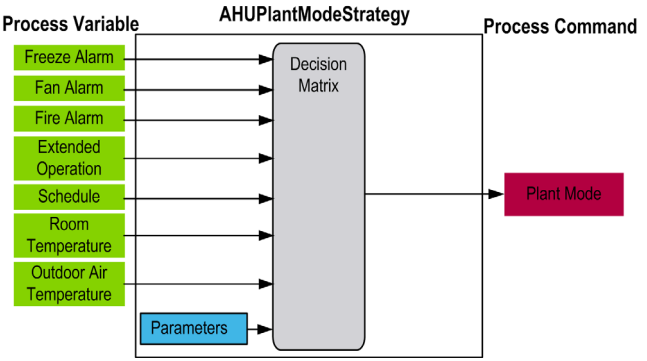


1 Motor control
2 Variable speed drive ATV••/••• Modbus
3 Graphic display
4 M171/M172 controller

Software Architecture

Function Block Diagram

The following function block diagram gives you an overview of the software architecture:



The block diagram presents on the left hand side the inputs, the **Process Variables**, on the right hand side the outputs, the **Process Commands**, as well as the function block `AHUPlantModeStrategy`:

The following table gives you an overview of the functions of `AHUPlantModeStrategy`

Function	Description
Decision Matrix	Sets plant modes depending on input and parameters.

Section 1.3
Function Block Description

AHUPlantModeStrategy Function Block

Function Block Description

The AHUPlantModeStrategy function block sets following operation modes of an air handling unit (AHU):

- AHU Off Mode
- Night Cycle Mode
- Night Purge Mode
- Occupied Operation Mode
- Extended Operation Mode
- AHU Off Delayed
- Freeze Alarm Mode
- Fan Alarm Mode
- Fire Alarm Mode

Plant Mode

The plant mode signal is a coded signal representing the operation mode for an air handling unit.

AHUPlantModeStrategy supports the following plant modes:

Value	Plant Mode
0	AHU Off
2	Night Cycle
3	Night Purge
4	Occupied operation
5	Extended operation
11	AHU Off Delayed
21	Freeze alarm
22	Fan alarm
23	Fire alarm

NOTE:

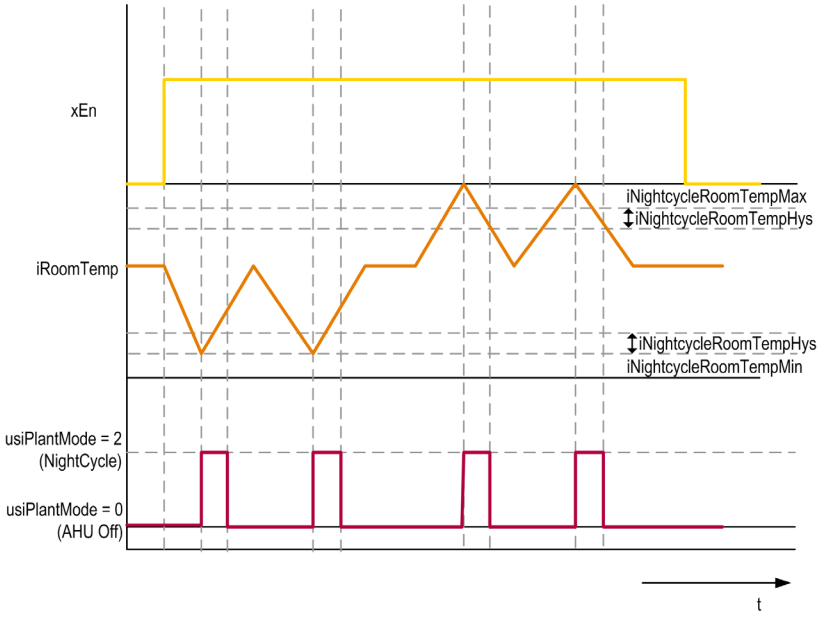
- If more than one plant mode condition is satisfied, the plant mode with the higher value takes priority and other plant modes are discarded.
- If the room temperature sensor (`iRoomTemp`) is not connected, the modes Night Cycle and Night Purge are disabled.
- If the outdoor air temperature sensor (`iOutdoorAirTemp`) is not connected, the Night Purge mode is disabled.

Night Cycle Mode

The Night Cycle mode maintains a lower room temperature during heating season and higher room temperature during cooling season.

This mode is selected when the room temperature input `iRoomTemp` exceeds the maximum limit `iRoomTempMaxLimit` and reduces below the minimum limit `iRoomTempMinLimit` when it is connected.

The functionality of Night Cycle mode with hysteresis is presented in the timing diagram below:



NOTE: The other plant mode conditions in the timing diagram are assumed as not active.

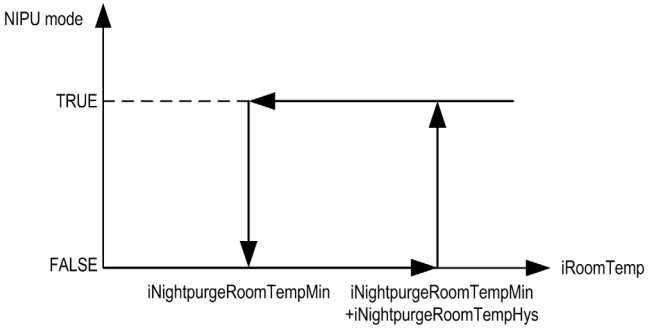
Night Purge Mode

The Night Purge Mode reduces morning cool down energy requirements by using cool outside air to purge and pre-cool the building.

Night Purge is active if the following conditions are satisfied:

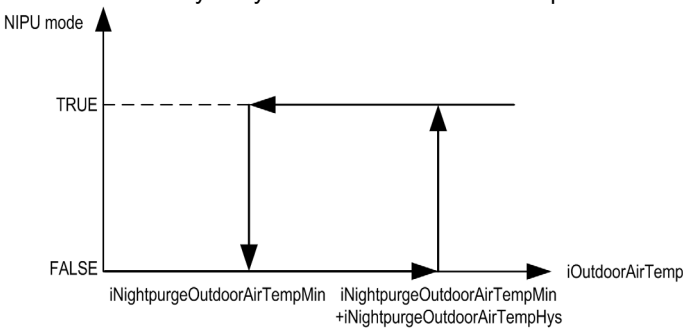
- If the current date and time is within the range of Start and Stop time of Night Purge mode.
- If `iRoomTemp` ≥ `iNightpurgeRoomTempMin` with hysteresis equal to `iNightpurgeRoomTempHys`.

  The functionality of hysteresis in this condition is presented below:



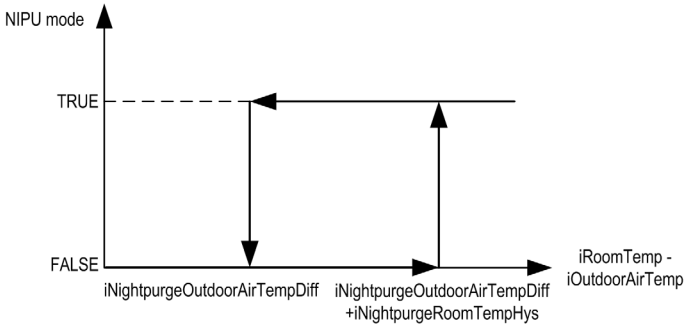
- If `iOutdoorAirTemp` ≥ `iNightpurgeOutdoorAirTempMin` with hysteresis equal to `iNightpurgeOutdoorAirTempHys`.

  The functionality of hysteresis in this condition is presented below:



- If actual difference between `iRoomTemp` and `iOutdoorAirTemp` ≥ `iNightpurgeOutdoorAirTempDiff` with hysteresis equal to `iNightpurgeRoomTempHys`.

The functionality of hysteresis in this condition is presented below:



The parameter `usiNightpurgeDays` is used to enable the Night Purge Mode based on the current day based on the real time clock of the controller. Each bit in the parameter represents one day as presented in the table below. If the real time clock has an error, Night Purge Mode is deactivated.

`usiNightpurgeDays` Bit	**Day**
0	Sunday
1	Monday
2	Tuesday
3	Wednesday
4	Thursday
5	Friday
6	Saturday
7	Not used

Occupied Operation Mode

The Occupied Operation mode is the standard mode during scheduled occupancy. The occupancy is signaled through the input `xSchedule`.

Extended Operation Mode

The Extended Operation mode is activated when the input `xExtendedOper` is TRUE. This mode is used during the extended occupancy due to overtime.

AHU Off Delayed Mode

AHU Off Delayed mode is required when the air handling unit is equipped with electrical heater or with humidifiers. In that case the temperature and humidification process is switched off, but the fans stays in operation to cool down electrical heating coils or remove excessive humidity from the ducts.

Switching off of the operation of the AHU is delayed by the period given by the parameter `uiAHUOffDelay`. This delay can be disabled by entering the value 0 for the parameter `uiAHUOffDelay`.

In case of an alarm, the AHU is switched off.

Freeze Alarm Mode

The Freeze Alarm Mode activates when the input `xFreezeStatAlarm` is TRUE. When this mode is TRUE, following operations are activated and executed by `AHUTempCntrlStrategy`:

- Dampers are closed
- Heating is set to 100%
- Cooling is disabled
- Fans are switched off

Fan Alarm Mode

The Fan Alarm Mode is activated when the input `xFanAlarm` is TRUE. When this mode is TRUE, following operations are activated and executed by other AHU function blocks:

- Temperature controller is switched off
- Dampers are closed
- Heating and cooling is disabled
- Fans are switched off

Fire Alarm Mode

The Fire Alarm Mode is activated when the input `xFireAlarm` is TRUE. When this mode is TRUE, following operations are activated and executed by other AHU function blocks:

- Temperature controller is switched off
- Dampers are closed
- Heating and cooling is disabled
- Fans are switched off

Section 1.4
Pin Description

What Is in This Section?

This section contains the following topics:

Input Pin Description

Pin Diagram

The following picture presents the pin diagram of `AHUPlantModeStrategy`:

Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
`xEn`	`BOOL`	TRUE or FALSE	N/A	TRUE: Enables the function block. FALSE: Disables the function block.
`xFreezeStatAlarm`	`BOOL`	TRUE or FALSE	N/A	Freeze Alarm
`xFanAlarm`	`BOOL`	TRUE or FALSE	N/A	Fan Alarm
`xFireAlarm`	`BOOL`	TRUE or FALSE	N/A	Fire Alarm

Input	Data Type	Range	Scaling / Unit	Description
xExtendedOper	BOOL	TRUE or FALSE	N/A	Extended Operation
xSchedule	BOOL	TRUE or FALSE	N/A	Schedule (Digital)
iRoomTemp	INT	-32768... +32767	0.1 °C/°F	Room Temperature If the iRoomTemp input is not connected, following services are disabled: ● Night Purge ● Night Cycle ● Room temperature sensor range verifications ● Room temperature short circuit alarm ● Room temperature alarm This is an optional input.
iOutdoorAirTemp	INT	-32768... +32767	0.1 °C/°F	Outdoor Air Temperature Night Purge Mode is deactivated if the input iOutdoorAirTemp is not connected. This is an optional input
iNightpurgeOutdoorAirTempMin	INT	-580...+3020	0.1 °C/°F	Night Purge minimum outdoor air temperature Default: 100
iNightpurgeRoomTempMin	INT	-580...+3020	0.1 °C/°F	Night Purge minimum room temperature Default: 180
iNightpurgeOutdoorAirTempDiff	INT	0...500	0.1 °C/°F	Night Purge difference between room temperature and outdoor air temperature Default: 40
iNightpurgeOutdoorAirTempHys	INT	0...500	0.1 °C/°F	Night Purge hysteresis outdoor air temperature Default: 20
iNightpurgeRoomTempHys	INT	0...500	0.1 °C/°F	Night Purge hysteresis room temperature Default: 20

Input	Data Type	Range	Scaling / Unit	Description
uiNightpurgeTimeStart	UINT	0...2359	N/A	Night Purge start time Default: 200 **NOTE:** For example, 200 means 2 am, and 2000 means 8 pm.
uiNightpurgeTimeStop	UINT	0...2359	N/A	Night Purge stop time Default: 600 **NOTE:** For example, 200 means 2 am, and 2000 means 8 pm.
usiNightpurgeDays	USINT	0...255	N/A	Night Purge days. Refer Night Purge-Plant mode *(see page 30)*. Default: 127
iNightcycleRoomTempMin	INT	-580...+3020	0.1 °C/°F	Night Cycle minimum room temperature Default: 100
iNightcycleRoomTempMax	INT	-580...+3020	0.1 °C/°F	Night Cycle maximum room temperature Default: 300
iNightcycleRoomTempHys	INT	0...500	0.1 °C/°F	Night Cycle hysteresis room temperature Default: 20
iRoomTempMaxLimit	INT	-580...+3020	0.1 °C/°F	Maximum room temperature Default: 400
iRoomTempMinLimit	INT	-580...+3020	0.1 °C/°F	Minimum room temperature Default: 120
uiInitializationDelay	UINT	0...600	s	After a power outage or a download, the Plant Mode is initialized with 0. Once the uiInitializationDelay has elapsed, the value can change from 0 to 2, 3, 4 or 5. The feature can be disabled by entering 0 for this parameter.
uiAHUOFFDelay	UINT	0...600	s	AHU is switched off after a delay. The feature can be disabled by entering 0 for this parameter.

NOTE: The parameters `uiNightpurgeTimeStart` and `uiNightpurgeTimeStop` are controlled parameters. Controlled parameters can be modified, however the modifications only become effective after the function block is disabled (`xEn` is set to FALSE) and subsequently re-enabled (`xEn` is set to TRUE).

Output Pin Description

Output Pin Description

Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
usiPlantMode	USINT	0...23	N/A	Plant Mode *(see page 29)*
uiAlertID	UINT	0...65536	N/A	Alert ID
uiAlarmID	UINT	0...65536	N/A	Alarm ID

Alarm ID Description

Each bit of the uiAlarmID output represents an alarm. The bits and their description are described in the following table:

Alarm Bit	Cause	Effect
0	(iNightCycleRoomTempMin > iNightCycleRoomTempMax)	The function block is disabled.

Alert ID Description

The uiAlertID output represents a value between 0 and 1023, whereby each bit represents an alert. The bits and their description are described in the following table:

Alert Bit	Cause	Effect
0	Invalid parameter range. An alert is generated if any of the following parameters is out of range. `uiAHUOffDelay` > 600 `uiInitializationDelay` > 600 `iNightpurgeOutdoorAirTempMin` < –580 or > +3020 `iNightpurgeRoomTempMin` < –580 or > +3020 `iNightpurgeOutdoorAirTempDiff` < 0 or > 500 `iNightpurgeOutdoorAirTempHys` < 0 or > 500 `uiNightpurgeTimeStart` < 0 or > 2359 `uiNightpurgeTimeStop` < 0 or > 2359 `uiNightpurgeTimeStart` = `uiNightpurgeTimeStop` `iNightpurgeRoomTempHys` < 0 or > 500 `iNightcycleRoomTempMin` < –580 or > +3020 `iNightcycleRoomTempMax` < –580 or > +3020 `iNightcycleRoomTempHys` < 0 or > 500 `iRoomTempMaxLimit` < –580 or > +3020 `iRoomTempMinLimit` < –580 or > +3020 If `uiNightpurgeTimeStart` or `uiNightpurgeTimeStop` values are not suitable to time format 00:00 (hours 00...23 and minutes 00...59).	Parameter values out of range that result in the equipment operating with less efficiency.
1	Alert due to analog input channel error detected for Room Temperature `iRoomTemp`	Night Cycle and Night Purge Mode are deactivated.
2	Alert due to analog input channel error detected for Outdoor Air Temperature `iOutdoorAirTemp`	Night Purge Mode is deactivated.
3	Room temperature ≥ Maximum room temperature	Alert to adjust the room temperature.
4	Room temperature ≤ Minimum room temperature	Alert to adjust the room temperature.
5	Freeze alarm	• Dampers are closed. • Heaters are set to 100%. • Cooling is disabled. • Fans are switched off.
6	Fan alarm	• Dampers are closed. • Temperature control is switched off. • Heating and cooling is disabled. • Fans are switched off.
7	Fire alarm	• Dampers are closed • Temperature control is switched off. • Heating and cooling is disabled • Fans are switched off.
8	Real time clock is not configured	Night Purge Mode is deactivated.

Alert Bit	Cause	Effect
9	A controlled parameter has been changed, which requires a machine restart. The new configuration parameter is effective after restart of the function block.	Alert due to a parameter change that has not been made effective yet.

Section 1.5
Troubleshooting

Troubleshooting

Alarm / Alert	Problem	Solution
`uiAlarmID.0 TRUE`	`iNightCycleRoomTempMin` > `iNightCycleRoomTempMax`	Verify that the parameter values are within their respective ranges.
`uiAlertID.0 TRUE`	Incorrect parameter value or configuration	Verify that the parameter values are within their respective ranges.
`uiAlertID.1 TRUE`	Room temperature input sensor is disconnected or short-circuited.	Verify whether the room temperature sensor is connected to the controller and working properly.
`uiAlertID.2 TRUE`	Outdoor air temperature input sensor is disconnected or short-circuited.	Verify whether the outdoor air temperature sensor is connected to the controller.
`uiAlertID.3 TRUE`	Room Temperature (RMT) input sensor value exceeds maximum RMT.	Verify that the temperature of the room corresponds to that of the input sensor value.
`uiAlertID.4 TRUE`	Room temperature value exceeds the maximum set value or is below the minimum value due to: • inoperable RMT sensor • error detected in configuration parameter • AHU calculated error value of DAT to high • inoperable cooling function	• Verify that the temperature of the room corresponds to that of the input sensor value. • Verify that the cooling coil is operational. • Verify that the fans are operational.
`uiAlertID.5 TRUE`	AHU or its components are inoperable.	• Verify and open the dampers. • Reset the heater temperature. • Set and enable the cooler. • Switch on fans.
`uiAlertID.6 TRUE`	Fan is not operating.	• Verify motor protection of fans. • Verify that the dampers are operational. • Reset the fans.
`uiAlertID.7 TRUE`	Fire alarm input activated	Validate that the input is in the correct state and if so, intervention required.
`uiAlertID.8 TRUE`	Real time clock is not operating.	Verify the real time clock. Set the time of the controller.
`uiAlertID.9 TRUE`	Controlled parameters are changed.	Restart the function block.

Chapter 2
AHU Temperature Control Strategy: AHUTempCntrlStrategy

What Is in This Chapter?

This chapter contains the following sections:

Section 2.1
Functional and Machine Overview

What Is in This Section?

This section contains the following topics:

Functional Overview

Functional Description

The `AHUTempCntrlStrategy` function block (Air Handling Unit Temperature Control Strategy) is a control strategy function block.

This function block controls the discharge air temperature of an air handling unit by modulating:

- heating coils
- cooling coils
- dampers
- fans

The function block `AHUTempCntrlStrategy` is used together with the function block `AHUPlantModeStrategy`.

Why Use the `AHUTempCntrlStrategy` Function Block?

The `AHUTempCntrlStrategy` function block is used for the following purposes:

Purpose	Description
Reduce energy consumption	Reduce energy consumption by means of: ● variable air flow ● economizer control ● summer compensation ● winter compensation
Optimize operation	● increase control accuracy ● increase comfort levels

Features of the `AHUTempCntrlStrategy` Function Block

The `AHUTempCntrlStrategy` function block provides the following features:

- Constant DAT (Discharge Air Temperature) control
- Return Air Compensated DAT control
- Fan Mode: variable and constant volume
- Cooling Mode: Modulating cooling coil/DX-cooling control
- Damper Mode: damper position controls air flow
- Sequential control of output signals: more than one actuator in series is controlled (damper, heating coil, cooling coil, discharge air fan, return air fan)
- Winter compensation: increases discharge air temperature at low temperature
- Summer compensation: increases RAT (Return Air Temperature) setpoint at high outside temperature
- Economizer function
- Freeze avoidance and freeze recovery

Erro Avoidance Features

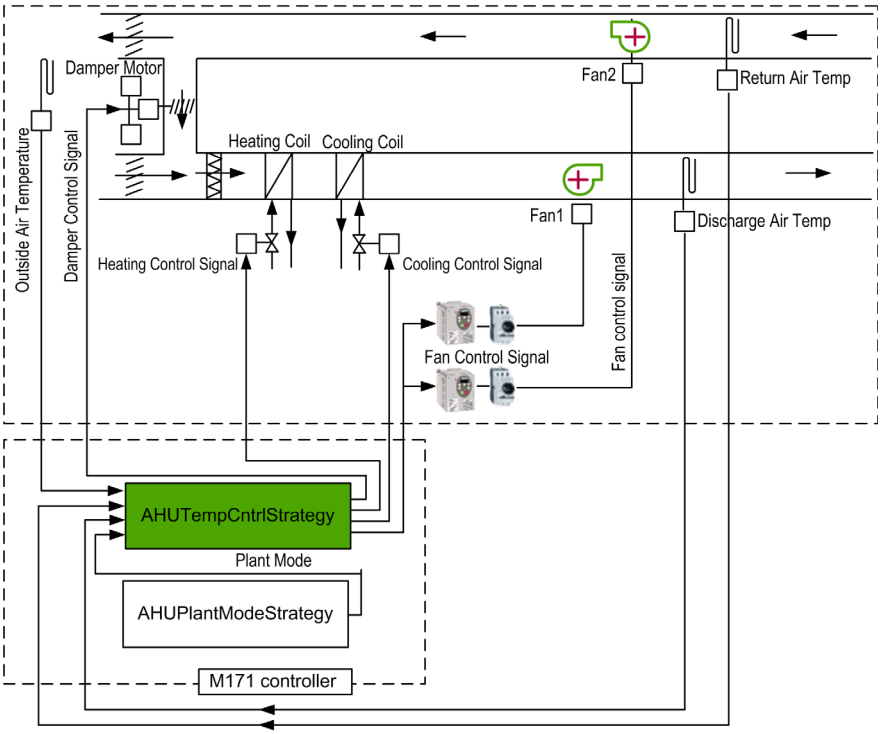
The `AHUTempCntrlStrategy` function block provides the following error avoidance features to help you avoid the potentials of certain sources of machine malfunction:

Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.
Alarm/alert notification	If an invalid value is entered, an alarm or an alert is generated: ● Alarm: the compressors are switched off and the function block terminates in error. ● Alert: the function block keeps on operating, however with the possibility of reduced performance.
Controlled parameter	The configuration of these parameters can be changed, however the changes are effective only after the restart of the function block. Several parameters are controlled. Please refer to the section Parameter Description *(see SoHVAC, HVAC&R Function Library, User Guide)*.

Machine Overview

Machine View

The following picture presents the interaction between the function block and the machine:



AHUTempCntrlStrategy controls the discharge air temperature by sequencing fan speeds, heating coil, dampers and cooling coil.

AHUPlantModeStrategy is connected to the `AHUTempCntrlStrategy` and controls the operation modes of the Air Handling Unit.

Section 2.2
Architecture

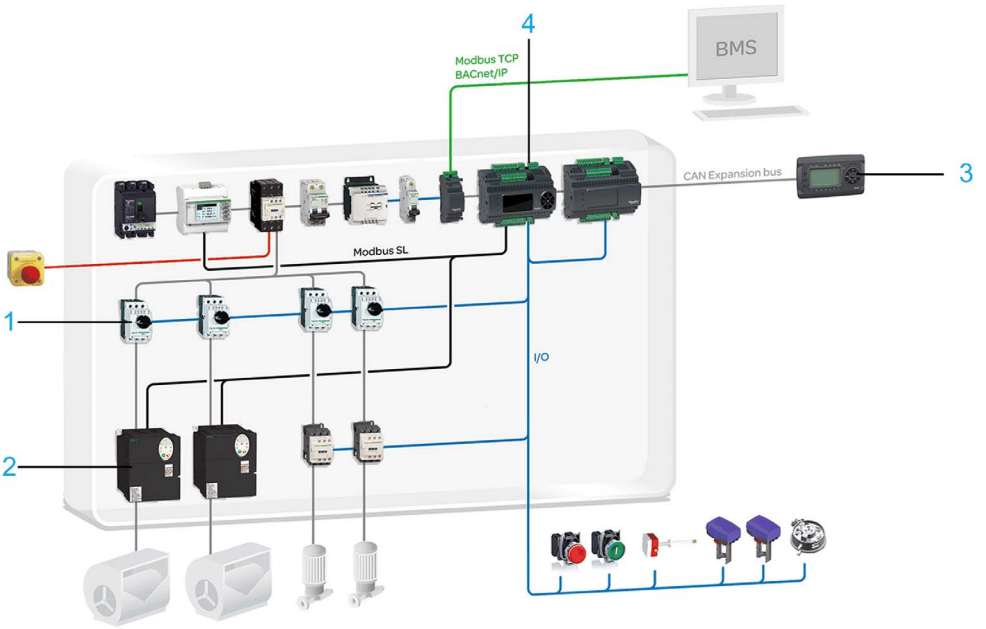
What Is in This Section?

This section contains the following topics:

Hardware Architecture

Hardware Architecture Overview

The following figure presents the hardware architecture of Air Handling Unit.

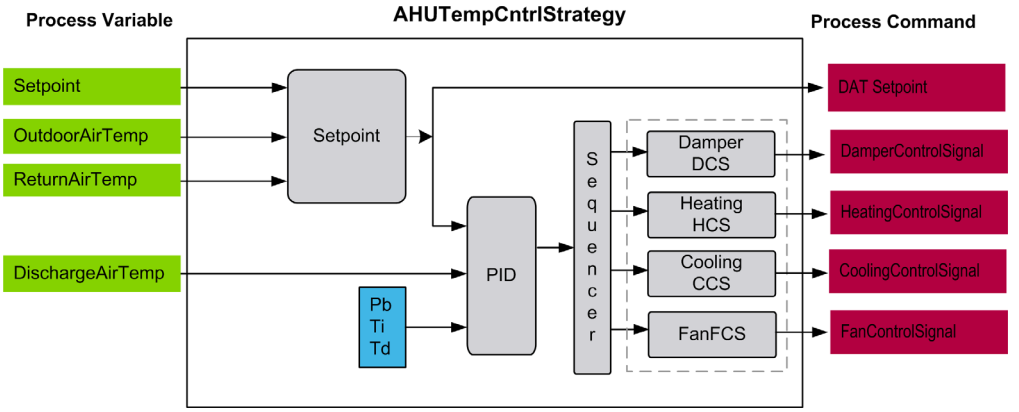


1 Motor control
2 Variable speed drive ATV••/••• Modbus
3 Graphic display
4 M171/M172 controller

Software Architecture

Function Block Diagram

The following function block diagram provides an overview of the software architecture:



The block diagram shows on the left hand side the inputs, the **Process Variables**, on the right hand side the outputs, the **Process Commands**, as well as the function block `AHUTempCntrlStrategy`.

The following table provides an overview of the functions of `AHUTempCntrlStrategy`:

Function	Description
Setpoint	calculates the DAT setpoint based on OAT, RAT, winter compensation and freeze avoidance function.
PID	• provides a control output by calculating the deviation between the actual DAT and the DAT setpoint. • is used to sequence ❍ fan speed ❍ heating coil ❍ dampers ❍ cooling coils
Sequencer	modulates control signals for fan speed, heating coil, cooling coils and dampers depending on the PID output.

Section 2.3
Function Block Description

AHUTempCntrlStrategy Function Block

Function Block Description

The AHUTempCntrlStrategy function block is a control strategy function block that regulates the temperature by controlling cooling coils, heating coils, dampers and fans.

AHUTempCntrlStrategy provides the following methods for temperature control:

- Constant DAT control
- RAT Compensated DAT control
- Sequential control of output signals
- Winter compensation of Discharge Air Temperature
- Summer compensation on Return Air Temperature
- Damper control and Fan Control
- Cooling mode: supports both chilled water cooling coils and direct expansion cooling (DX cooling)
- Economizer function
- Freeze avoidance and freeze avoidance recovery control

NOTE: This function block is to be used together with the AHUTempCntrlStrategy function block.

Constant DAT Control and Compensated DAT Control

The following table provides an overview of Constant and Compensated DAT control:

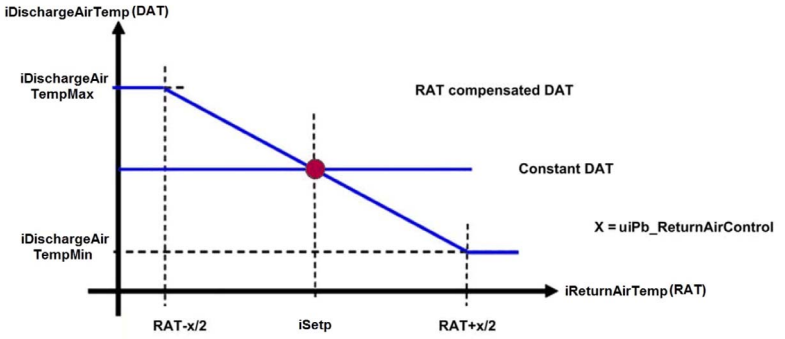
Mode	Description
Constant DAT Control	Discharge Air Temperature is kept constant as set by input setpoint. ● xControlMode=FALSE ● uses the DAT sensor to provide PID-control on the DAT. ● uses the following parameters: xControlMode uiKp uiTi uiTd

Mode	Description

Compensated DAT Control	• room / exhaust air compensated Discharge Air Temperature control (master-slave control)
	• `xControlMode`=TRUE
	• requires the RAT sensor; in case the RAT sensor is not available or when the sensor is in alarm, control will change to Constant DAT Control
	• provides the possibility of P-control on the compensated DAT setpoint and PID control on the DAT
	• for this function, it is necessary to connect a RAT sensor to the function block
	• uses the following parameters:
	`xControlMode`
	`uiPb_ReturnAirControl`
	`iDischargeAirTempMin`
	`iDischargeAirTempMax`
	`uiKp`
	`uiTi`
	`uiTd`

Both control modes sequence output signals for heating, for mixed air dampers, fans and for cooling.

The following graphic presents the Compensated DAT control. The DAT setpoint varies between the minimum and maximum DAT setpoint, depending on the deviation on the RAT.



Sequential Control of Output Signals

The sequencer controls more than one actuator in series. One actuator is controlled until the end of range before the next actuator is used.

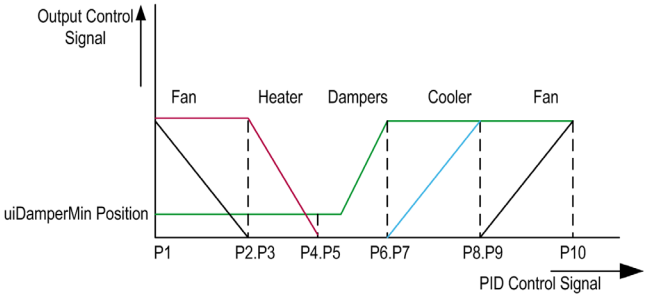
The PID control is divided over multiple actuators. The user can define the order and the percentage of the PID control signal for each actuator.

The sequencing of the output control signals is defined in the `AHUTempCntrlStrategy` function block for the actuators

- heater
- damper
- fan
- cooler

The start and end points of each output signal can be configured for each air handling unit using the parameters from `uiFanSeqStart1`...`uiFanSeqStop2`.

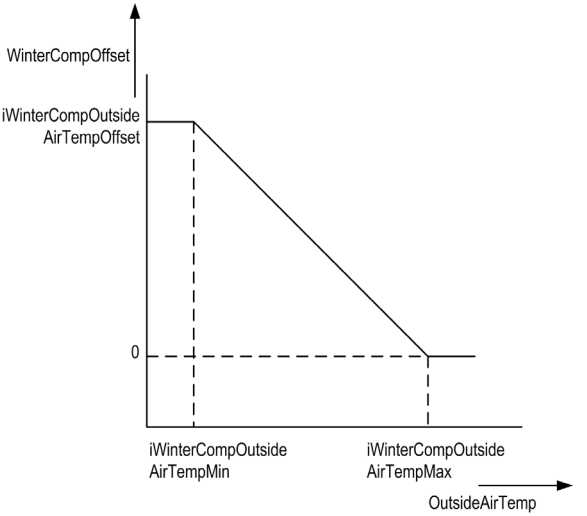
The following figure presents an example of sequencing Fans, Heating coil, Dampers, Cooling coil and Fans:

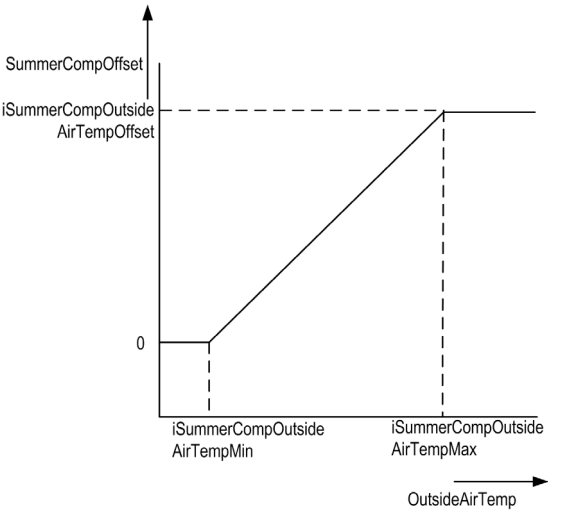


P1 `uiFanSeqStart1`
P2 `uiFanSeqStop1`
P3 `uiHeatingSeqStart`
P4 `uiHeatingSeqStop`
P5 `uiDamperSeqStart`
P6 `uiDamperSeqStop`
P7 `uiCoolingSeqStart`
P8 `uiCoolingSeqStop`
P9 `uiFanSeqStart2`
P10 `uiFanSeqStop2`

Winter and Summer Compensation

The following table provides an overview of winter and summer compensation:

Mode	Description
Winter Compensation	Winter compensation of Discharge Air Temperature: ● DAT setpoint increases at low outside temperature and hence increases the comfort level ● winter compensation is disabled if the OAT sensor is inoperable 

Mode	Description
Summer Compensation	Summer compensation on Return Air Temperature ● RAT setpoint increases at high Outside Air Temperature ● results in higher room temperature and thus helps avoid sudden temperature changes and reduces energy consumption ● is disabled if the OAT sensor is inoperable 

Damper Control and Fan Control

The damper position and the fan speed control the air flow.

NOTICE
INOPERABLE AIR HANDLING UNIT ● Make sure that the open/close dampers are opened before starting fans. ● Make sure that the parameter damper actuator run time is set to the actuator run time of the device. ● Make sure that the fans are started before heating. ● Make sure that the heating is stopped before stopping the fans. **Failure to follow these instructions can result in equipment damage.**

NOTE: A damper end switch indicates whether the dampers are fully opened.

The `AHUTempCntrlStrategy` function block allows configuring the following types:

Type	Description
Modulating dampers	• output varies between 0 - 100% • `xDamperMode`= FALSE • Economizer mode possible
Open / Close dampers	• ON / OFF (output is 0 or 100%) • `xDamperMode`= TRUE • Fans start operating, when the dampers are fully open. • `uiDamperActuatorRuntime` defines the time for the dampers to change from a close position to the open position. • Damper status is checked before starting fans. An alarm is raised, when the dampers are not open within the damper actuator run time.

Cooling Control

The function block supports cooling using chilled water or a refrigerant (DX direct expansion valve control).

Economizer Function

The built-in economizer function reduces the energy consumption

- by closing the dampers in case of a high OAT
- by fully opening the dampers in case of heating mode

To save energy, the modulating damper control can be reversed or set to a minimum position.

If...	Then...
the Outside Air Temperature is higher than the Return Air Temperature	modulating damper control is reversed: In cooling mode, the dampers are fully closed. In heating mode the dampers are fully open.
the Outside Air Temperature exceeds the value of `iEconomizerOutdoorAirTempMax`,	the modulating damper control is set to a minimum.

The economizer function of the dampers can be enabled or disabled by means of `xEconomizerMode`. This function is disabled if the OAT or RAT sensors are inoperable.

Freeze Avoidance Recovery Control

Freeze Recovery is activated on 2 conditions:

1. if the Plant mode `usiPlantMode` is changed from Freeze alarm mode (21) to any other mode 1 to 5, or
2. if the Plant mode `usiPlantMode` is changed from AHU OFF to any other mode 1 to 5 and the `iOutdoorAirTemp` is less than Freeze Limit.

The following actions are executed during the Freeze recovery mode:

- Dampers are closed and forced recirculation for a duration specified by `uiFreezeRecoveryDur`.
- The parameter `iFreezeRecoveryIncr` is used as offset value for the setpoint for a duration specified by `uiFreezeRecoveryDur`.

Once the system comes back to normal state, dampers position is based on the Plant mode and temperature control.

The following parameters are used in the freeze avoidance recovery feature:

- `iFreezeLimit`
- `iFreezeRecoveryIncr`
- `uiFreezeRecoveryDur`

NOTE: Freeze Recovery is applicable only for modulating dampers. This function is disabled for Open/Close damper.

Plant Mode

The `usiPlantMode` signal from the function block `AHUPlantModeStrategy` indicates the operation mode for the air handling unit.

Plant Mode supports following operation modes:

Plant Mode	Description
0	AHU Off
2	Night cycle
3	Night Purge
4	Occupied operation
5	Extended operation
11	AHU Off Delayed Mode
21	Freeze alarm
22	Fan alarm
23	Fire alarm

Section 2.4
Pin Description

What Is in This Section?

This section contains the following topics:

Input Pin Description

Pin Diagram

The following graphic presents the pin diagram of `AHUTempCntrlStrategy`.

Input Pin Description

Input	Data Type	Range	Scaling/ Unit	Description
xEn	BOOL	TRUE or FALSE	N/A	TRUE: Enables the function block. FALSE: Disables the function block. The outputs including the Alert and Alarm outputs are set to 0.
iDischargeAirTemp	INT	-32768... +32767	0.1 °C/°F	Discharge air temperature. The alarm due to this input sets the following outputs to 0: ● iHeatingControlSignal ● iDamperControlSignal ● iCoolingControlSignal ● iFanControlSignal
iReturnAirTemp	INT	-32768... +32767	0.1 °C/°F	Return air temperature (optional). Refer also to the description for iReturnAirTemp below this table.
iOutdoorAirTemp	INT	-32768... +32767	0.1 °C/°F	Outdoor air temperature. When this input indicates an unavailable or a short-circuited sensor alarm, the function block disables the following services: ● Summer compensation ● Winter compensation ● Economizer function
iSetp	INT	-32768... +32767	0.1 °C/°F	Setpoint
xDamperEndswitch	BOOL	TRUE or FALSE	N/A	Damper opened TRUE: Damper opened (default) FALSE: Damper closed
usiPlantMode	USINT	0...23	N/A	Refer Plant Mode *(see page 57)*.
xControlMode	BOOL	TRUE or FALSE	N/A	Control mode **TRUE:** Return air temperature compensated DAT **FALSE:** Constant DAT Default: FALSE
uiPb_ReturnAirControl	UINT	1...500	0.1 °C/°F	Pb RAT control Default: 40
iDischargeAirTempMin	INT	-580... +3020	0.1 °C/°F	Minimum DAT Setpoint Default:160
iDischargeAirTempMax	INT	-580... +3020	0.1 °C/°F	Maximum DAT Setpoint Default: 260

Input	Data Type	Range	Scaling/ Unit	Description
iHeatingLimit	INT	-580... +3020	0.1 °C/°F	OAT limit heating closed Default: 200
iCoolingLimit	INT	-580... +3020	0.1 °C/°F	OAT limit cooling closed Default: 140
iWinterCompOutsideAir TempMin	INT	-580... +3020	0.1 °C/°F	Winter compensation OAT low Default: -100
iWinterCompOutsideAir TempMax	INT	-580... +3020	0.1 °C/°F	Winter compensation OAT high Default: 100
iWinterCompOutsideAir TempOffset	INT	0... 200	0.1 °C/°F	Winter compensation OAT offset Default: 40
iSummerCompOutsideAir TempMin	INT	-580... +3020	0.1 °C/°F	Summer compensation OAT low Default: 200
iSummerCompOutsideAir TempMax	INT	-580... +3020	0.1 °C/°F	Summer compensation OAT high Default: 300
iSummerCompOutsideAir TempOffset	INT	0... 200	0.1 °C/°F	Summer compensation OAT offset Default: 40
uiKp	UINT	1... 3000	N/A	Value of proportional gain Default: 40
uiTi	UINT	0...3600	0.1 s	Ti DAT control Default: 0
uiTd	UINT	0...3600	0.1 s	Td DAT control Default: 0
uiFanSeqStart1	UINT	0...1000	0.1%	FCS startpoint 1 Default: 0 Controlled parameter. Changes only become effective after a reset of the function block.
uiFanSeqStop1	UINT	0...1000	0.1%	FCS endpoint 1 Default: 200 Controlled parameter. Changes only become effective after a reset of the function block.
uiHeatingSeqStart	UINT	0...1000	0.1%	HCS startpoint Default: 200 Controlled parameter. Changes only become effective after a reset of the function block.

Input	Data Type	Range	Scaling/ Unit	Description
uiHeatingSeqStop	UINT	0...1000	0.1%	HCS endpoint Default: 400 Controlled parameter. Changes only become effective after a reset of the function block.
uiDamperSeqStart	UINT	0...1000	0.1%	DCS startpoint Default: 400 Controlled parameter. Changes only become effective after a reset of the function block.
uiDamperSeqStop	UINT	0...1000	0.1%	DCS endpoint Default: 600 Controlled parameter. Changes only become effective after a reset of the function block.
uiCoolingSeqStart	UINT	0...1000	0.1%	CCS startpoint Default: 600 Controlled parameter. Changes only become effective after a reset of the function block.
uiCoolingSeqStop	UINT	0...1000	0.1%	CCS endpoint Default: 800 Controlled parameter. Changes only become effective after a reset of the function block.
uiFanSeqStart2	UINT	0...1000	0.1%	FCS startpoint 2 Default: 800 Controlled parameter. Changes only become effective after a reset of the function block.
uiFanSeqStop2	UINT	0...1000	0.1%	FCS endpoint 2 Default: 1000 Controlled parameter. Changes only become effective after a reset of the function block.
xDamperMode	BOOL	TRUE or FALSE	N/A	Mode dampers **TRUE:** Open-close **FALSE:** Modulating Default: FALSE Controlled parameter. Changes only become effective after a reset of the function block.

Input	Data Type	Range	Scaling/ Unit	Description
uiDamperActuatorRuntime	UINT	0... 300	s	Damper runtime Minimum delay to start Default: 150
uiDamperMinPosition	UINT	0...1000	0.1%	Minimum damper position Default: 200
uiMinFanFrequency	UINT	0...1000	0.1 Hz	Minimum frequency FCS Default: 200
uiMaxFanFrequency	UINT	0... 2000	0.1 Hz	Maximum frequency FCS Default: 500
xEconomizerMode	BOOL	TRUE or FALSE	N/A	Economizer mode **TRUE:** On **FALSE:** Off Default: TRUE Controlled parameter. Changes only become effective after a reset of the function block.
iEconomizerDiff	INT	1...100	0.1 °C/°F	Economizer difference Default: 20
iEconomizerOutdoorAir TempMax	INT	-580... +3020	0.1 °C/°F	Maximum OAT economizer mode Default: 300
xCoolingMode	BOOL	TRUE or FALSE	N/A	Cooling mode **TRUE:** DX **FALSE:** Modulating Default: FALSE Controlled parameter. Changes only become effective after a reset of the function block.
usiCoolingNbStages	USINT	0...8	N/A	Cooling DX number of stages Default: 0 Controlled parameter. Changes only become effective after a reset of the function block.
usiFanMode	USINT	0...2	N/A	Fan mode **0** variable speed based on DAT **1** variable speed based on RAT **2** 1 Stage Default: 0 Controlled parameter. Changes only become effective after a reset of the function block.
iFreezeLimit	INT	-580... +3020	0.1 °C/°F	OAT limit frost protection Default: 60

Input	Data Type	Range	Scaling/ Unit	Description
iFreezeRecoveryIncr	INT	0... 500	0.1 °C/°F	Freeze recovery DAT step increase Default: 100 Controlled parameter. Changes only become effective after a reset of the function block.
uiFreezeRecoveryDur	UINT	0...1200	s	Freeze recovery duration Default: 600 Controlled parameter. Changes only become effective after a reset of the function block.
iDischargeAirTempMaxLimit	INT	-580... +3020	0.1 °C/°F	DAT maximum temperature Default: 400
iDischargeAirTempMinLimit	INT	-580... +3020	0.1 °C/°F	DAT minimum temperature Default: 120
iReturnAirTempMaxLimit	INT	-580... +3020	0.1 °C/°F	RAT maximum temperature Default: 400
iReturnAirTempMinLimit	INT	-580... +3020	0.1 °C/°F	RAT minimum temperature Default: 120

NOTE: The parameters uiFanSeqStart1, uiFanSeqStop1, uiHeatingSeqStart, uiHeatingSeqStop, uiDamperSeqStart, uiDamperSeqStop, uiCoolingSeqStart, uiCoolingSeqStop, uiFanSeqStart2, uiFanSeqStop2, xDamperMode, xEconomizerMode, xCoolingMode, usiCoolingNbStages, xFanMode, iFreezeRecoveryIncr, uiFreezeRecoveryDur are controlled parameters. Controlled parameters can be modified, however the modifications only become effective after the function block is disabled (xEn is set to FALSE) and subsequently re-enabled (xEn is set to TRUE).

NOTE:

- A valid parameter input resets an alarm or alert automatically.
- A constant hysteresis of 10 is considered wherever the analog input channel is compared to a limit.

iReturnAirTemp

If this input is not connected, the function block does not generate an analog channel alarm and operates in the Constant DAT control regardless of the `xControlMode` input.

The following services are disabled if this input is not connected to the function block:

- Summer compensation
- Compensated DAT control
- Return air temperature sensor range verification
- Return air temperature short circuit alarm
- Return air temperature unavailable alarm
- Economizer mode

Output Pin Description

Output Pin Description

Output Pin Description

Output	Data Type	Range	Scaling/Unit	Description
iDischargeAirTempSetp	INT	-32768...+32767	0.1 °C/°F	DAT setpoint
iHeatingControlSignal	INT	0...1000	0.1%	Heating coil signal
iDamperControlSignal	INT	0...1000	0.1%	Damper control signal
iCoolingControlSignal	INT	0...1000	0.1%	Cooling coil signal
iFanControlSignal	INT	uiMinFanFrequency to uiMaxFanFrequency	0.1 Hz	Fan control signal
uiAlertID	UINT	0...65536	N/A	Alert identification
uiAlarmID	UINT	0...65536	N/A	Alarm identification
iPIDOutput	INT	0...1000	0.1%	PID output

Alert ID Description

The `uiAlertID` output represents a value between 0 and 255, whereby each bit represents an alert. The bits and their description are described in the following table:

Alert Bit	Alert Cause	Effect
0	Invalid parameter range. An alert is generated if any of the following parameters is out of range. `uiPb_ReturnAirControl` < 1 or > 500 `iDischargeAirTempMin` < –580 or > 3020 `iDischargeAirTempMax` < –580 or > 3020 `iHeatingLimit` < –580 or > +3020 `iCoolingLimit` < –580 or > +3020 `iWinterCompOutsideAirTempMin` < –580 or > +3020 `iWinterCompOutsideAirTempMax` < –580 or > +3020 `iWinterCompOutsideAirTempOffset` < 0 or > 200 `iSummerCompOutsideAirTempMin` < –580 or > +3020 `iSummerCompOutsideAirTempMax` < –580 or > +3020 `iSummerCompOutsideAirTempOffset` < 0 or > 200 `uiKp` < 1 or > 0300 `uiTi` < 0 or > 3600 `uiTd` < 0 or > 3600 `uiFanSeqStart1` < 0 or > 1000 `uiFanSeqStop1` < 0 or > 1000 `uiHeatingSeqStart` < 0 or > 1000 `uiHeatingSeqStop` < 0 or > 1000 `uiDamperSeqStart` < 0 or > 1000 `uiDamperSeqStop` < 0 or > 1000 `uiCoolingSeqStart` < 0 or > 1000 `uiCoolingSeqStop` < 0 or > 1000 `uiFanSeqStart2` < 0 or > 1000	Parameter values out of range that result in the equipment operating with less efficiency.

Alert Bit	Alert Cause	Effect

0	`uiFanSeqStop2` < 0 or > 1000
	`uiFanSeqStart1` ≥ `uiFanSeqStop1`
	`uiHeatingSeqStart` ≥ `uiHeatingSeqStop`
	`uiDamperSeqStart` ≥ `uiDamperSeqStop`
	`uiCoolingSeqStart` ≥ `uiCoolingSeqStop`
	`uiFanSeqStart2` ≥ `uiFanSeqStop2`
	`uiDamperActuatorRuntime` < 0 or > 300
	`uiDamperMinPosition` < 0 or > 1000
	`uiMinFanFrequency` < 0 or > 1000
	`uiMaxFanFrequency` < 0 or > 1000
	`iEconomizerDiff` < 1 or > 100
	`iEconomizerOutdoorAirTempMax` < –580 or > +3020
	`usiCoolingNbStages` < 0 or > 8
	`iFreezeLimit` < –580 or > +3020
	`iFreezeRecoveryIncr` < 0 or > 500
	`uiFreezeRecoveryDur` < 0 or > 1200
	`iDischargeAirTempMaxLimit` < –580 or > +3020
	`iDischargeAirTempMinLimit` < –580 or > +3020
	`iReturnAirTempMaxLimit` < –580 or > +3020
	`iReturnAirTempMinLimit` < –580 or > +3020
	`iDischargeAirTempMin` > `iDischargeAirTempMax`
	`iWinterCompOutsideAirTempMin` > `iWinterCompOutsideAirTempMax`
	`iSummerCompOutsideAirTempMin` > `iSummerCompOutsideAirTempMax`
	`iDischargeAirTempMinLimit` > `iDischargeAirTempMaxLimit`
	`iReturnAirTempMinLimit` > `iReturnAirTempMaxLimit`
	`uiMinFanFrequency` > `uiMaxFanFrequency`	

1	Alert due to analog input channel for return air temperature `iReturnAirTemp`. This alert is generated if the return air temperature sensor is not connected or short-circuited.	• Function block operates. • Temperature control changes to constant DAT control • RAT compensated DAT control is disabled • Summer compensation is disabled • Default value is initialized to this input to differentiate between a disconnected sensor and a sensor with a detected error.

Alert Bit	Alert Cause	Effect
2	Alert due to analog input channel for outdoor air temperature `iOutdoorAirTemp`.	• Function block operates, however with limited performance. • Following functions are disabled: ❍ Summer compensation ❍ Winter compensation ❍ Economizer
3	Alert when discharge air temperature exceeds maximum limit: `iDischargeAirTemp` > `iDischargeAirTempMaxLimit`, Hysteresis = 10.	Alert for limited performance
4	Alert when discharge air temperature is below minimum limit: `iDischargeAirTemp` < `iDischargeAirTempMinLimit`, Hysteresis = 10.	Alert for limited performance
5	Alert when return air temperature exceeds maximum limit: `iReturnAirTemp` > `iReturnAirTempMaxLimit`, Hysteresis = 10.	Alert for limited performance
6	Alert when return air temperature is below minimum limit: `iReturnAirTemp` < `iReturnAirTempMinLimit`, Hysteresis = 10.	Alert for limited performance
7	Alert due to improper DX cooling configuration: `CoolingNbStages` is set as 0 and DX stage mode is selected.	Alert for limited performance
8	The change of a controlled parameter is not active. Changing a controlled parameter requires a machine restart. The controlled configuration parameter setting is effective only after restart of the function block.	Present changes are not active. Function block uses the previously set values.

Alarm ID Description

The `uiAlarmID` output represents a value between 0 and 7, whereby each bit represents a detected alarm. The bits and their description are described in the following table:

Alarm Bit	Alarm Cause	Effect
0	Invalid parameter value An alarm is generated when one of the following parameters is out of range: (`iWinterCompOutsideAirTempMin` >= `iWinterCompOutsideAirTempMax`) (`uiMinFanFrequency` > `uiMaxFanFrequency`) (`xControlMode` = TRUE) and (`iDischargeAirTempMin` > `iDischargeAirTempMax`) (`xControlMode` = TRUE) and (`SummerCompOutsideAirTempMin` > `SummerCompOutsideAirTempMax`) (`usiFanMode` < 0 or `usiFAnMode` > 2)	Function block is disabled and the outputs are set to 0.
1	Alarm due to discharge air temperature analog input `iDischargeAirTemp`. This alarm is generated when discharge air temperature sensor is short-circuited or disconnected.	Function block is disabled and the outputs are set to 0.
2	Dampers are not opened within damper actuator runtime	Function block is disabled and the outputs are set to 0.

Section 2.5
Troubleshooting

Troubleshooting

Troubleshooting

Alarm / Alert	Problem	Solution
`uiAlarmID.0 TRUE`	Invalid parameter value	Verify that the parameter values are within their respective ranges.
`uiAlarmID.1 TRUE`	Discharge air temperature sensor is short-circuited or disconnected	Verify whether the discharge air temperature sensor is connected to the controller.
`uiAlarmID.2 TRUE`	Dampers are not opened within the actuator runtime	Verify the damper end switch and the actuator run time.
`uiAlertID.0 TRUE`	Invalid parameter value	Verify that the parameter values are within their respective ranges.
`uiAlertID.1 TRUE`	Return air temperature sensor is short-circuited or disconnected	Verify whether the sensor is connected to the controller.
`uiAlertID.2 TRUE`	Outdoor air temperature sensor is short-circuited or disconnected	
`uiAlertID.3 TRUE`	Discharge air temperature exceeds maximum limit.	• Verify whether the heater valve is closed and cooling value is open. • Verify the cold water production.
`uiAlertID.4 TRUE`	Discharge air temperature is below the minimum limit.	
`uiAlertID.5 TRUE`	Return air temperature exceeds maximum limit.	• Verify whether the cooling valve is open. • Verify the air flow and cold water production.
`uiAlertID.6 TRUE`	Return air temperature is below the minimum limit.	• Verify whether the heater valve is open. • Verify the air flow and hot water production.
`uiAlertID.7 TRUE`	Improper configuration of DX cooling	Verify whether the value of parameter `usiCoolingNbStages` is greater than 0
`uiAlertID.8 TRUE`	Configuration parameters are changed which requires the function block to restart	Disable the function block and enable the function block again
Heating valve is closed	`iOutdoorAirTemp` > `iHeatingLimit`	Increase parameter `iHeatingLimit`
Cooling valve is closed	`iOutdoorAirTemp` < `iCoolingLimit`	Decrease parameter `iCoolingLimit`

Alarm / Alert	Problem	Solution
Fans are not started	Dampers must open before the fan starts	• Open the dampers first in case of Open/Close Damper control • Start the fan after expiration of the damper actuator run time
The outputs remain at 0	Function block is disabled	• Enable the function block • Verify the input `usiPlantMode`
Dampers are at minimum position	Economizer function is active	Set `iEconomizerOutdoorAirTempMax` < `iOutdoorAirTemp`
Dampers are working in reverse direction.	Economizer function is active	Set `iOutdoorAirTemp` > `iReturnAirTemp`
Discharge air temperature setpoint is high	Winter compensation is active	Verify the outside air temperature and winter compensation parameters
Discharge air temperature setpoint is high	Freeze recovery/avoidance is active	Verify the parameters for freeze recovery/avoidance

Chapter 3
ATV Modbus Communication: ATV••ModbusCom / ATV••• ModbusCom

What Is in This Chapter?

This chapter contains the following topics:

Functional Overview

Functional Description

The ATV••/•••ModbusCom function block manages the communication between the controller and the ATV••/••• drive through Modbus serial line communication protocol.

The function block can be used only with the Modicon M171 or M172 Performance logic controllers (M171P/M172P) and supports the following drives:

- ATV12
- ATV21
- ATV212
- ATV31
- ATV32
- ATV312
- ATV61
- ATV71

Why Use the ATV••/•••ModbusCom Function Block?

The ATV••/•••ModbusCom function block is used to manage the following ATV••/••• drive functionality:

- Control of Start and Stop operation
- Control of frequency
- Monitoring the connection

Features of the ATV••/•••ModbusCom Function Block

The ATV••/•••ModbusCom function block facilitates the following features:

- reads and monitors the ATV drive status
- writes the ATV command
- generates an alarm in case of a detected communication error
- reports the detected ATV••/••• drive error and alarms

Error Avoidance Features

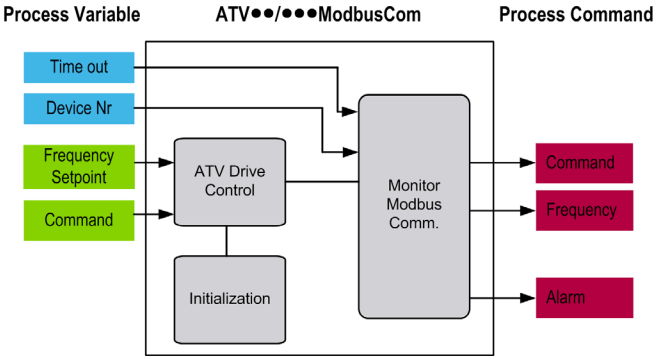
The `ATV••/•••ModbusCom` function block provides the following error avoidance features to help you avoid the potentials of certain sources of machine malfunction:

Error Avoidance Feature	Description
Input range validation	Input ranges are validated to help prevent out of range data.
Alarm/alert notification	If an invalid value is entered, an alarm or an alert is generated: ● Alarm: the compressors are switched off and the function block terminates in error. ● Alert: the function block keeps on operating, however with the possibility of reduced performance.
Controlled parameter	The input parameters `usiNodeNr` and `uiTimeOut` are controlled. The values of these parameters can be changed, however the changes are effective only after the restart of the function block.

Software Architecture

Function Block Diagram

The following function block diagram gives you an overview of the software architecture:



The block diagram presents on the left hand side the inputs, the **Process Variables**,on the right hand side the outputs, the **Process Commands**, as well as the function block `ATV••/•••ModbusCom`.

The following table gives you an overview of the functions of `ATV••/•••ModbusCom`:

Function	Description
ATV Drive Control	checks the ATV drive status
Initialization	• During initialization, the function block reads the minimum and maximum drive frequency. • If the requested frequency is not within the frequency limits, an alarm is generated.
Monitor Modbus Comm	helps to ensure a stable communication with the ATV drive

ATV••/•••ModbusCom Function Block

Function Block Description

The ATV••/•••ModbusCom function block is used for the communication between the controller and the ATV drive through the Modbus serial connection.

The ATV••/•••ModbusCom function block initializes the drive to Start or to Stop. It also checks the Modbus communication.

ATV Drive Operation

The ATV••/•••ModbusCom function block manages the ATV drive as follows:

- Initialize the drive
- Start the drive
- Stop the drive
- Check the Modbus communication

ATV Error Management

The ATV drive detected errors and alarms are indicated by the outputs uiAlarmID and uiAlertID.

The output uiAlarmID indicates which alarm occurs.

The output uiAlertID indicates which alert occurs.

⚠ 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.

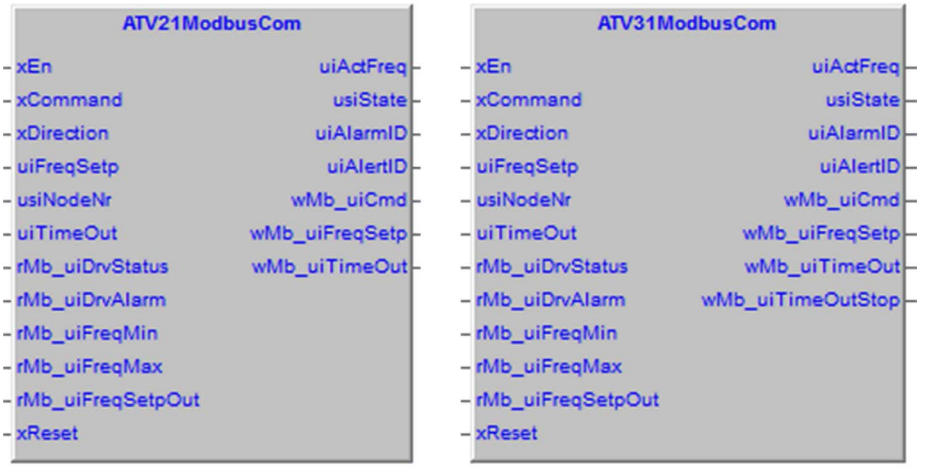
Failure to follow these instructions can result in death, serious injury, or equipment damage.

[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.

Input Pin Description

Pin Diagram

The following picture presents the pin diagram of `ATV••/•••ModbusCom`:



NOTE: There is one additional output (`wMb_uiTimeOutStop`) for the `ATV12/31/32/312/61/71ModbusCom` function blocks, see Output Pin Description *(see page 81)* for `ATV12/31/32/312/61/71ModbusCom`.

Input Pin Description

Input	Data Type	Range	Scaling/Unit	Description
`xEn`	`BOOL`	TRUE or FALSE	N/A	Enables the function block
`xCommand`	`BOOL`	TRUE or FALSE	N/A	TRUE: Start drive FALSE: Stop drive
`xDirection`	`BOOL`	TRUE or FALSE	N/A	TRUE: reverse direction FALSE: forward direction
`uiFreqSetp`	`UINT`	0...65535	0.1 Hz	Drive frequency
`usiNodeNr`	`USINT`	0…127	N/A	Modbus device address
`uiTimeOut`	`UINT`	0...100	1 s	Modbus device time out
`rMb_uiDrvStatus`	`UINT`	N/A	N/A	State of drive
`rMb_uiDrvAlarm`	`UINT`	N/A	N/A	Alarm of drive
`rMb_uiFreqMin`	`UINT`	N/A	N/A	Drive minimum frequency
`rMb_uiFreqMax`	`UINT`	N/A	N/A	Drive maximum frequency

Input	Data Type	Range	Scaling/Unit	Description
rMb_uiFreqSetpOut	UINT	N/A	N/A	Drive frequency
xReset	BOOL	TRUE or FALSE	N/A	Reset drive detected alarm

NOTE: Alarms are automatically reset when the input `xReset` is constant TRUE.

NOTE: The inputs `usiNodeNr` and `uiTimeOut` are controlled parameters. Controlled parameters can be modified, however the modifications only become effective after the function block is disabled (`xEn` is set to FALSE) and subsequently re-enabled (`xEn` is set to TRUE).

NOTE: A change of the `xDirection` input will not have an effect when the drive is in RUN mode. For example, if the drive is in RUN mode and the initial value of `xDirection` was FALSE (forward direction), changing the value of the input to TRUE (reverse direction) will not have an immediate effect; the drive will continue to command a forward direction. The drive must first be stopped and restarted before the input change is recognized and for the change of direction to take effect.

You must carefully manage the module Modbus network addresses because each device on the network requires a unique address. Having multiple devices with the same address can cause unpredictable operation of your network and associated equipment.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Verify that there is only one master controller configured on the network or remote link.
- Verify that all devices have unique addresses.
- Confirm that the address of the device is unique before placing the system into service.

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

Output Pin Description

Output Pin Description for ATV21/212ModbusCom

Output	Data Type	Range	Scaling/Unit	Description
uiActFreq	UINT	0...5000	0.1 Hz	Drive frequency
usiState	USINT	0...5	N/A	State of drive
uiAlertID	UINT	0...65535	N/A	Alert identifier
uiAlarmID	UINT	0...65535	N/A	Alarm identifier
wMb_uicmd	UINT	N/A	N/A	Command for drive
wMb_uiFreqSetp	UINT	N/A	N/A	Frequency setpoint for drive
wMb_uiTimeOut	UINT	N/A	N/A	Timeout for drive

NOTE: The ATV drive detected errors and alarms are indicated by the outputs uiAlertID and uiAlarmID.

Output Pin Description for ATV12/31/32/312/61/71ModbusCom

Output	Data Type	Range	Scaling/Unit	Description
uiActFreq	UINT	0...5000	0.1 Hz	Drive frequency
usiState	USINT	0...5	N/A	State of drive
uiAlertID	UINT	0...65535	N/A	Alert identifier
uiAlarmID	UINT	0...65535	N/A	Alarm identifier
wMb_uiCmd	UINT	N/A	N/A	Command for drive
wMb_uiFreqSetp	UINT	N/A	N/A	Frequency setpoint for drive
wMb_uiTimeOut	UINT	N/A	N/A	Timeout for drive
wMb_uiTimeOutStop	UINT	N/A	N/A	Enable the drive timeout

NOTE: The ATV drive detected errors and alarms are indicated by the outputs uiAlertID and uiAlarmID.

usiState

The output usiState represents a value between 0 and 5, whereby each number represents a state of the ATV drive. The following table describes the values and their description:

Value	Description
0	Drive initializing
1	Drive stops
2	Drive starting
3	Drive runs
4	Drive stopping
5	Drive in alarm

Alert ID Description

The uiAlertID output represents a value between 0 and 15, whereby each bit represents an alert. The bits and their description are described in the following table:

Alert Bit	Alert Cause	Effect
0	The controlled parameter usiNodeNr is changed. Changing a controlled parameter requires a machine restart. The controlled configuration parameter setting is effective only after a restart of the function block.	• Present changes are not active. • The function block uses the previously set values.
1	The controlled parameter uiTimeOut is changed. Changing a controlled parameter requires a machine restart. The controlled configuration parameter setting is effective only after a restart of the function block.	
2	Frequency is out of range. Frequency is above maximum frequency of the ATV drive.	The frequency input is ignored and not communicated to ATV drive.
3	Frequency is out of range. Frequency is below minimum frequency of the ATV drive.	
4-15	not used	N/A

Alarm ID Description

The `uiAlarmID` output represents a value between 0 and 255, whereby each bit represents a detected alarm. The bits and their description are described in the following table:

Alarm Bit	Alarm Cause	Effect
0	Invalid range input `usiNodeNr`	The function block is disabled.
1	Invalid range input `uiTimeOut`	
2	A communication error has been detected.	• Communication to ATV drive is not possible. • FB will try to automatically recover communication.
3	A motor input phase loss has been detected.	• The motor is stopped. • The function block is disabled.
4	An error caused by motor over current has been detected.	
5	An error caused by motor short circuit has been detected.	
6	An error caused by motor overload has been detected.	
7	An error caused by motor phase loss has been detected.	
8	An error caused by mains over voltage has been detected.	
9 to 15	Not used	N/A

Troubleshooting

Troubleshooting

Alarm / Alert	Problem	Solution
uiAlertID.0	A controlled parameter has been changed (usiNodeNr)	Enable the function block again.
uiAlertID.1	A controlled parameter has been changed (uiTimeOut)	Enable the function block again.
uiAlertID.2	The frequency is out of range. The frequency is above maximum frequency ATV drive	1. Verify the minimum and maximum frequency set in the ATV drive. 2. After changing the ATV drive parameters, disable and enable the function block. 3. Verify the application program parameters for maximum frequency settings.
uiAlertID.3	The frequency is out of range. The frequency is below minimum frequency ATV drive	1. Verify the minimum and maximum frequency set in the ATV drive. 2. After changing the ATV drive parameters, disable and enable the function block. 3. Verify the application program parameters for maximum frequency settings.
uiAlarmID.0	Invalid parameter value (usiNodeNr)	Verify that the parameter values are within their respective ranges.
uiAlarmID.1	Invalid parameter value (uiTimeOut)	Verify that the parameter values are within their respective ranges.
uiAlarmID.2	A communication time out error has been detected. No communication to the Modbus device	1. Check the wiring and addressing externally. 2. Disable and enable the function block.
uiAlarmID.3	A motor input phase loss error has been detected.	Verify the motor.
uiAlarmID.4	A motor over current error has been detected.	
uiAlarmID.5	A motor short circuit current error has been detected.	
uiAlarmID.6	A motor overload error has been detected.	
uiAlarmID.7	A motor phase loss error has been detected.	
uiAlarmID.8	A mains over voltage error has been detected.	

Implementation

Implementation for ATV21/212ModbusCom

You have to create variables in status variables in the application.

Read variables must be defined as read-only variables, Write variables must not be defined as read-only variables.

Address	Variable Name	Data Type	Read-Only	Description
N	rMBx_uiDrvStatus	UINT **Device Type:** Unsigned 16-Bit	TRUE	State of drive
N+1	rMBx_uiDrvAlarm	UINT **Device Type:** Unsigned 16-Bit	TRUE	Alarm of drive
N+2	rMBx_uiFreqMin	UINT **Device Type:** Unsigned 16-Bit	TRUE	Drive minimum frequency
N+3	rMBx_uiFreqMax	UINT **Device Type:** Unsigned 16-Bit	TRUE	Drive maximum frequency
N+4	rMBx_uiFreqSetpOut	UINT **Device Type:** Unsigned 16-Bit	TRUE	Drive frequency
N+5	rMBx_uiCmd	UINT **Device Type:** Unsigned 16-Bit	FALSE	Command for drive
N+6	rMBx_uiFreqSetp	UINT **Device Type:** Unsigned 16-Bit	FALSE	Frequency setpoint for drive
N+7	rMBx_uiTimeOut	UINT **Device Type:** Unsigned 16-Bit	FALSE	Timeout for drive

The status variables must be assigned to the Modbus profile ATV21 or ATV212.

Read-only variables must be assigned to the corresponding registers in the input registers:

Parameter	Address	Type	Variable	Type	DataBlock
rMb_uiDrvStatus	64770	UINT	rMbx_uiDrvStatus	UINT	MW110.1
rMb_uiDrvAlarm	64657	UINT	rMbx_uiDrvAlarm	UINT	MW110.0
rMb_uiFreqMin	20	UINT	rMbx_uiFreqMin	UINT	MW110.3
rMb_uiFreqMax	19	UINT	rMbx_uiFreqMax	UINT	MW110.2
rMb_uiFreqSetpOut	64769	UINT	rMbx_uiFreqSetpOut	UINT	MW110.4

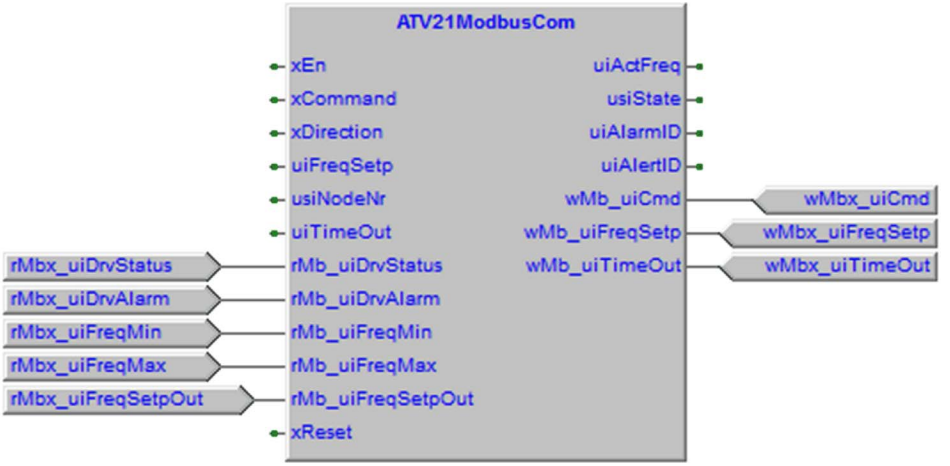
Write variables must be assigned to the corresponding registers in the output registers:

Parameter	Address	Type	Variable	Type	DataBlock
wMb_uiFreqSetp	64002	UINT	wMbx_uiFreqSetp	UINT	MW110.6
wMb_uiCmd	64001	UINT	wMbx_uiCmd	UINT	MW110.5
wMb_uiTimeOut	2052	UINT	wMbx_uiTimeOut	UINT	MW110.7

The status variables assigned to the Modbus registers must be connected to the function block:

- Read-only variables assigned to the input registers must be connected to the corresponding input pins of the function block.
- Write variables assigned to the output registers must be connected to the corresponding output pins of the function block.

NOTE: If several instances of the function block are created inside the project, you have to create 8 new variables in the status variable for each function block.



Implementation for ATV12/31/32/312/61/71ModbusCom

You have to create variables in status variables in the application.

Read variables must be defined as read-only variables, Write variables must not be defined as read-only variables.

Address	Variable Name	Data Type	Read-Only	Description
N	rMBx_uiDrvStatus	UINT **Device Type:** Unsigned 16-Bit	TRUE	State of drive
N+1	rMBx_uiDrvAlarm	UINT **Device Type:** Unsigned 16-Bit	TRUE	Alarm of drive
N+2	rMBx_uiFreqMin	UINT **Device Type:** Unsigned 16-Bit	TRUE	Drive minimum frequency
N+3	rMBx_uiFreqMax	UINT **Device Type:** Unsigned 16-Bit	TRUE	Drive maximum frequency
N+4	rMBx_uiFreqSetpOut	UINT **Device Type:** Unsigned 16-Bit	TRUE	Drive frequency
N+5	rMBx_uiCmd	UINT **Device Type:** Unsigned 16-Bit	FALSE	Command for drive
N+6	rMBx_uiFreqSetp	UINT **Device Type:** Unsigned 16-Bit	FALSE	Frequency setpoint for drive
N+7	rMBx_uiTimeOut	UINT **Device Type:** Unsigned 16-Bit	FALSE	Timeout for drive
N+8	rMBx_uiTimeOutStop	UINT **Device Type:** Unsigned 16-Bit	FALSE	Enable drive timeout

The status variables must be assigned to the Modbus profile ATV12, ATV31, ATV32, ATV312, ATV61 or ATV71.

Read-only variables must be assigned to the corresponding registers in the input registers:

Parameter	Address	Type	Variable	Type	DataBlock
rMb_uiDrvStatus	64770	UINT	rMbx_uiDrvStatus	UINT	MW110.1
rMb_uiDrvAlarm	64657	UINT	rMbx_uiDrvAlarm	UINT	MW110.0
rMb_uiFreqMin	20	UINT	rMbx_uiFreqMin	UINT	MW110.3
rMb_uiFreqMax	19	UINT	rMbx_uiFreqMax	UINT	MW110.2
rMb_uiFreqSetpOut	64769	UINT	rMbx_uiFreqSetpOut	UINT	MW110.4

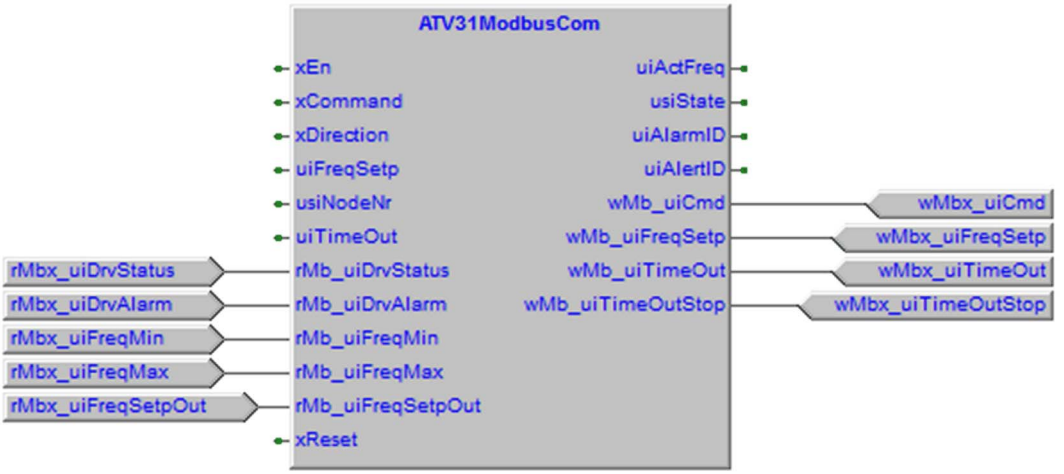
Write variables must be assigned to the corresponding registers in the output registers:

Parameter	Address	Type	Variable	Type	DataBlock
wMb_uiFreqSetp	8503	UINT	wMbx_uiFreqSetp	UINT	MW110.6
wMb_uiCmd	8502	UINT	wMbx_uiCmd	UINT	MW110.5
wMb_uiTimeOut	6006	UINT	wMbx_uiTimeOut	UINT	MW110.7
wMb_uiTimeOutStop	7011	UINT	wMbx_uiTimeOutStop	UINT	MW110.8

The status variables assigned to the Modbus registers must be connected to the function block:

- Read-only variables assigned to the input registers must be connected to the corresponding input pins of the function block.
- Write variables assigned to the output registers must be connected to the corresponding output pins of the function block.

NOTE: If several instances of the function block are created inside the project, you have to create 9 new variables in the status variable for each function block.



Chapter 4
Compressor Alarm Management: CompAlarmMgmt

What Is in This Chapter?

This chapter contains the following sections:

Section 4.1
Functional Overview

`CompAlarmMgmt` Function Block Description

Function Block Description

The `CompAlarmMgmt` function block is used to detect alarms of a compressor in a chiller unit or a compressor rack.

Why Use the `CompAlarmMgmt` Function Block?

The `CompAlarmMgmt` function block provides the following purposes:

- The `CompAlarmMgmt` function block manages up to 7 different alarms for a single compressor. Each alarm can be configured with an individual alarm delay time.
- The `CompAlarmMgmt` simplifies programming.

Features of the `CompAlarmMgmt` Function Block

The `CompAlarmMgmt` function block provides the following features:

- Emergency alarm
- Motor alarm
- Oil pressure switch alarm
- High / low pressure switch alarm
- High / low pressure sensor alarm
- Generic alarm
- Individual alarm delays
- Enable / disable alarm
- Manual reset *(see page 101)*

Error Avoidance Features

The function block provides the following error avoidance feature to help avoid certain sources of machine malfunction:

Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.

Section 4.2
Pin Description

Pin Description

Pin Diagram

The following picture presents the pin diagram of `CompAlarmMgmt`:

Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
xEn	BOOL	TRUE or FALSE	N/A	Enables the function block.
xEStop	BOOL	TRUE or FALSE	N/A	Emergency Stop **NOTE:** Reset when not active.
xMotorAlarm	BOOL	TRUE or FALSE	N/A	Motor alarm (for example circuit breaker, thermoswitch) **NOTE:** Reset when not active.
xOilPressSwitch	BOOL	TRUE or FALSE	N/A	Oil pressure switch alarm **NOTE:** Reset when not active.
xHPSwitch	BOOL	TRUE or FALSE	N/A	High pressure switch alarm **NOTE:** Reset when not active.
xLPSwitch	BOOL	TRUE or FALSE	N/A	Low pressure switch alarm **NOTE:** Can be reset when active.
xGenericAlarm	BOOL	TRUE or FALSE	N/A	Generic Alarm **NOTE:** Reset when not active.
xCompCommand	BOOL	TRUE or FALSE	N/A	Compressor command TRUE: compressor is running. FALSE: compressor is stopped.
iHPValue	INT	- 32768… +32767	0.01 bar	High pressure current value
iLPValue	INT	- 32768… +32767	0.01 bar	Low pressure current value
iMaxHPLimit	INT	- 32768… +32767	0.01 bar	Maximum value of the high pressure
iMinLPLimit	INT	- 32768… +32767	0.01 bar	Minimum value of the low pressure

Input	Data Type	Range	Scaling / Unit	Description
uiDelayEnAlarm	UINT	0…65535	s	Start delay for LP switch and LP alarm when the compressor is running. **NOTE:** The timer uiDelayEnAlarm is started when the input xCompCommand is set to TRUE.
uiDelayMotorAlarm	UINT	0…65535	s	Delay for the motor alarm **NOTE:** The timer uiDelayMotorAlarm is started when the input xMotorAlarm is set to TRUE.
uiDelayOilPressSwitch	UINT	0…65535	s	Delay for the oil pressure switch alarm **NOTE:** The timer uiDelayOilPressSwitch is started when the input xOilPressSwitch is set to TRUE.
uiDelayHPSwitch	UINT	0…65535	s	Delay for the HP switch alarm
uiDelayLPSwitch	UINT	0…65535	s	Delay for the LP switch alarm. **NOTE:** The timer uiDelayLPSwitch is started when the input xLPSwitch is set to TRUE.
uiDelayGenericAlarm	UINT	0…65535	s	Delay for the generic alarm
uiDelayHPAlarm	UINT	0…65535	s	Delay for the HP alarm **NOTE:** The timer uiDelayHPAlarm is started when the high pressure iLPValue is higher than the high pressure limit iHPMaxLimit.

Input	Data Type	Range	Scaling / Unit	Description
uiDelayLPAlarm	UINT	0…65535	s	Delay for the LP alarm **NOTE:** The timer uiDelayLPAlarm is started when the low pressure iLPValue is lower than the low pressure limit iLPMinLimit.
uiDisableAlarm	UINT	0…65535	N/A	Disable some alarms (optional)
xReset	BOOL	TRUE or FALSE	N/A	Resets the alarms which are not active or which can be reset when active.

xEn (Enable)

If the function block is disabled, the outputs uiAlarmID and xAlarm are set to 0 and the alarms are reset, even if all the alarms are active.

Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
xAlarm	BOOL	TRUE or FALSE	N/A	Alarm
uiAlarmID	UINT	0… 65535	N/A	Alarm identification

NOTE: If the function block is disabled, the outputs uiAlarmID and xAlarm are set to 0 and the alarms are reset even if all the alarms are active.

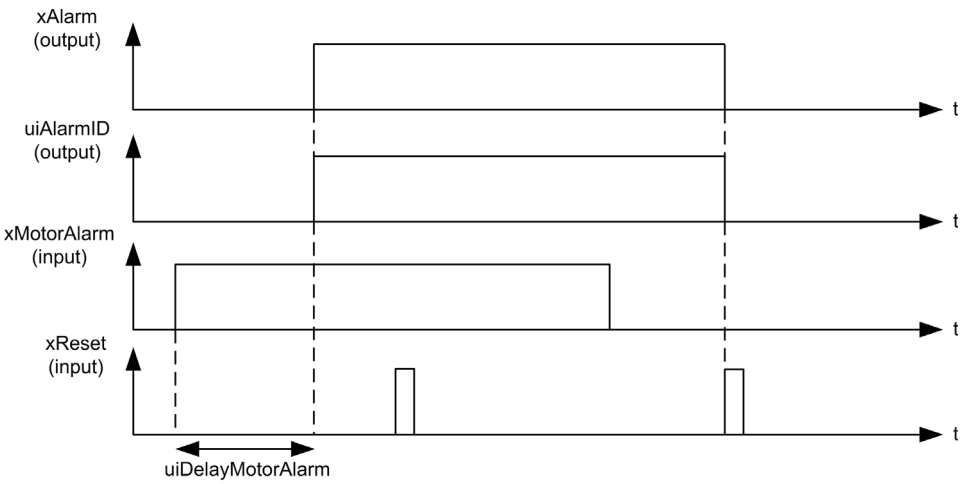
Motor Alarm

The timer uiDelayMotorAlarm is started when the input xMotorAlarm is set to TRUE.

A motor alarm can be detected when the motor alarm detection is enabled.

A motor alarm is detected when the input xMotorAlarm is set to TRUE and the timer uiDelayMotorAlarm has elapsed.

NOTE: The detected motor alarm can be reset when the input xMotorAlarm is FALSE and the reset input xReset is set to TRUE.



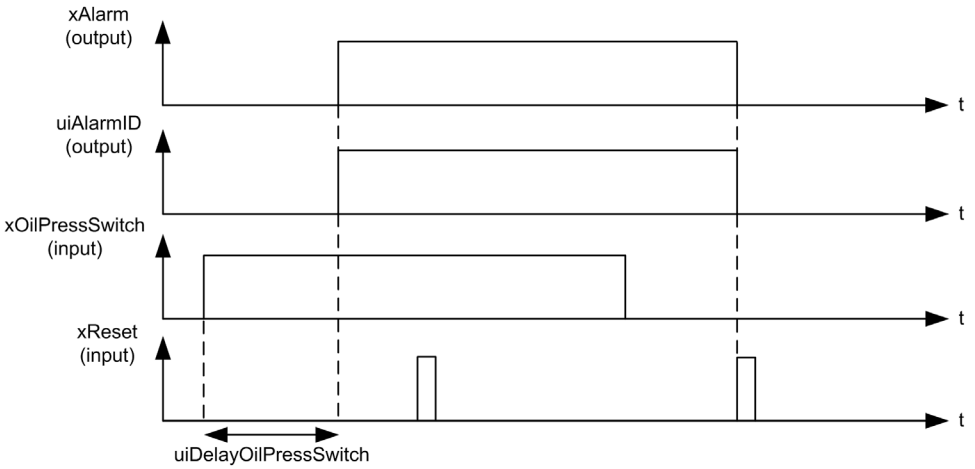
Oil Pressure Switch Alarm

The timer `uiDelayOilPressSwitch` is started when the input `xOilPressSwitch` is set to TRUE.

An oil pressure switch alarm can be detected when the oil pressure switch alarm detection is enabled.

An oil pressure switch alarm is detected when the input `xOilPressSwitchAlarm` is set to TRUE and the timer `uiDelayOilPressSwitchAlarm` has elapsed.

NOTE: The detected oil pressure switch alarm can be reset when the input `xOilPressSwitchAlarm` is FALSE and the reset input `xReset` is set to TRUE.



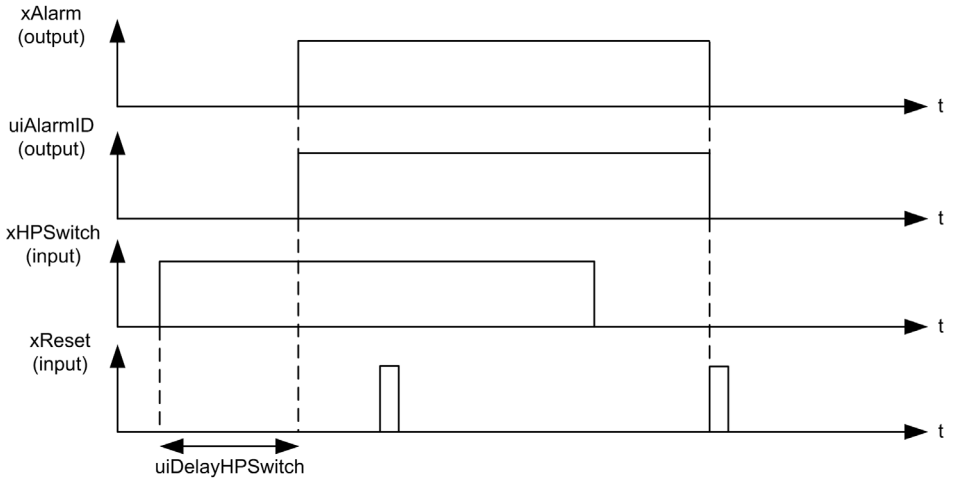
High Pressure Switch Alarm

The timer `uiHPSwitch` is started when the input `xHPSwitch` is set to TRUE.

A high-pressure switch alarm can be detected when the high-pressure switch alarm detection is enabled.

A high-pressure switch alarm is detected when the input `xHPSwitch` is set to TRUE and the timer `uiDelayHPSwitchAlarm` has elapsed.

NOTE: The detected high-pressure switch alarm can be reset when the input `xHPSwitch` is FALSE and the reset input `xReset` is set to TRUE.



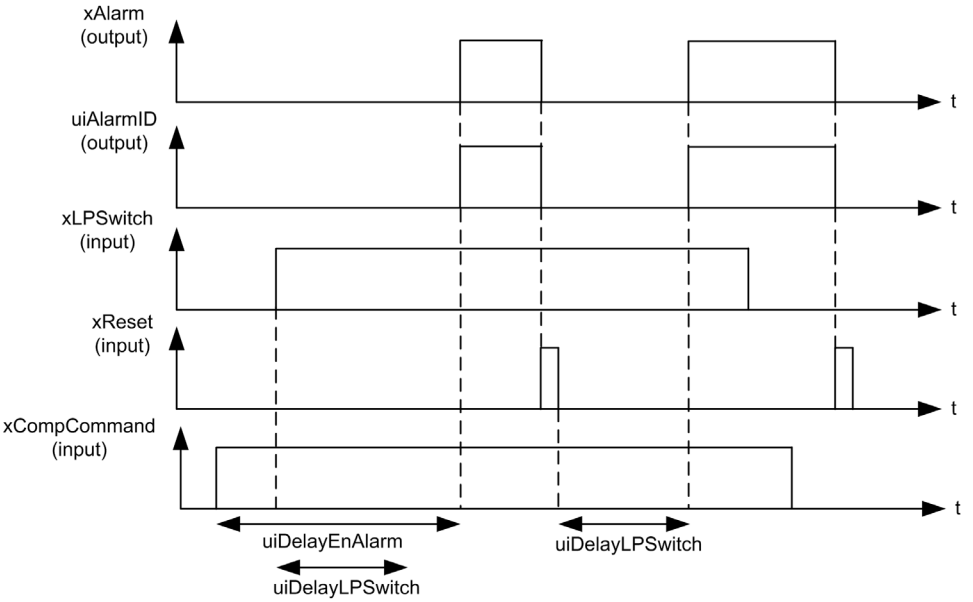
Low-Pressure Switch Alarm

The timer `uiDelayEnAlarm` is started when the input `xCompCommand` is set to TRUE. The timer `uiDelayLPSwitch` is started when the input `xLPSwitch` is set to TRUE.

A low-pressure switch alarm can be detected when the low-pressure switch alarm detection is enabled.

A low-pressure switch alarm is detected when the input `xLPSwitch` is set to TRUE and the timers `uiDelayEnAlarm` and `uiDelayLPSwitch` have elapsed.

NOTE: The detected low-pressure switch alarm is reset when the input `xReset` is set to TRUE.

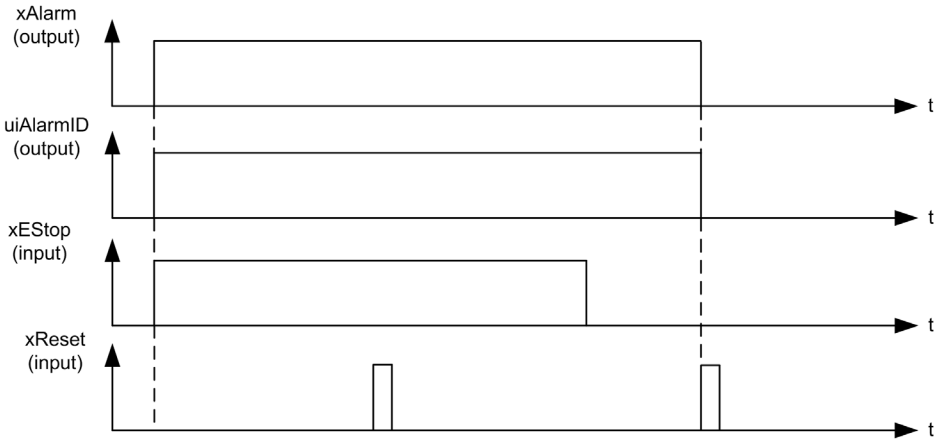


Emergency Stop Alarm

An emergency stop alarm can be detected when the emergency stop alarm detection is enabled.

An emergency stop alarm is detected when the input `xEStop` is set to TRUE.

NOTE: The detected emergency stop alarm can be reset when the input `xEStop` is FALSE and the reset input `xReset` is set to TRUE.



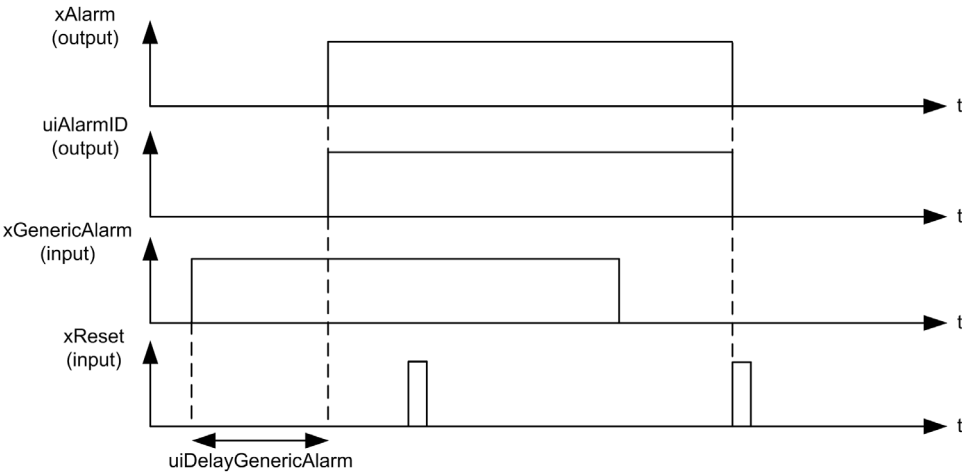
Generic Alarm

The timer `uiDelayGenericAlarm` is started when the input `xGenericAlarm` is set to TRUE.

A generic alarm can be detected when the generic alarm detection is enabled.

A generic alarm is detected when the input `xGenericAlarm` is set to TRUE and the timer `uiDelayGenericAlarm` has elapsed.

NOTE: The detected motor alarm can be reset when the input `xGenericAlarm` is FALSE and the reset input `xReset` is set to TRUE.



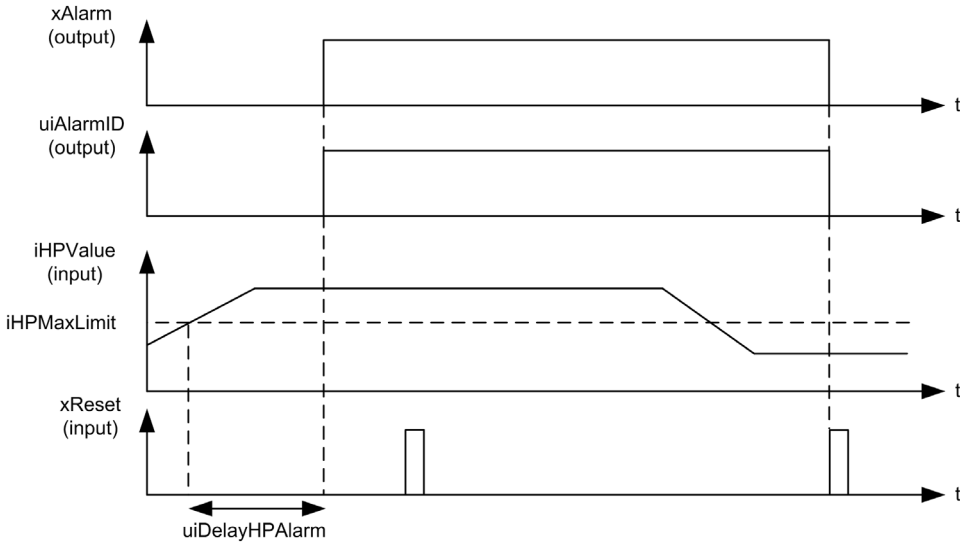
High-Pressure Alarm

The timer `uiDelayHPAlarm` is started when the high-pressure `iLPValue` is higher than the high-pressure limit `iHPMaxLimit`.

A high-pressure alarm can be detected when the high-pressure alarm detection is enabled.

A high-pressure alarm is detected when the input `iHPValue` is above the high-pressure limit (`iHPMaxLimit`) and the timer `uiDelayHPAlarm` has elapsed.

NOTE: The detected high-pressure alarm can be reset when the input `iHPValue` is below the high-pressure limit (`iHPMaxLimit`) and the reset input `xReset` is set to TRUE.



Low-Pressure Alarm

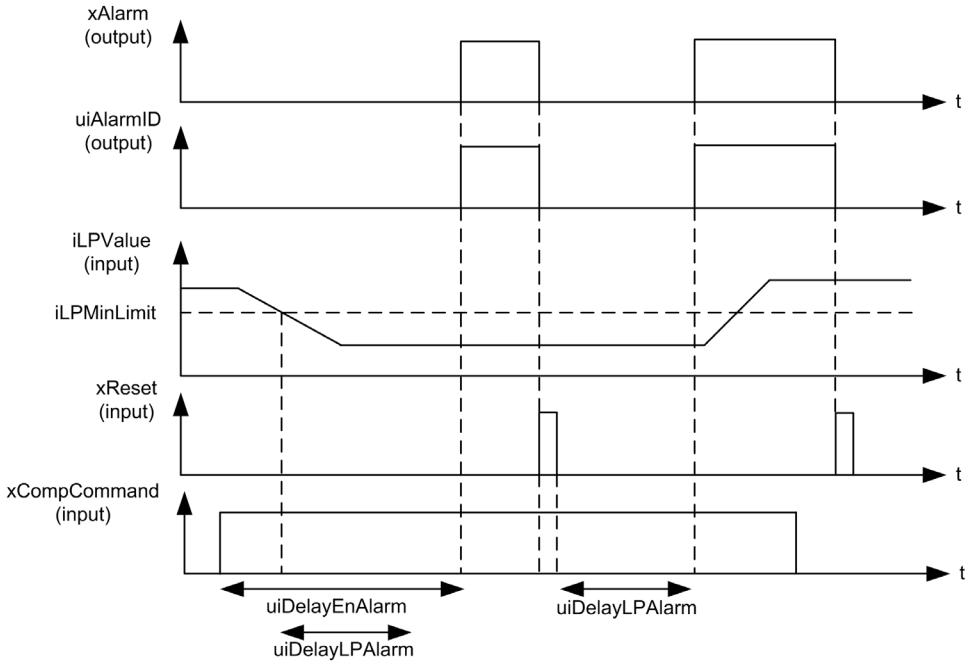
The timer `uiDelayEnAlarm` is started when the input `xCompCommand` is set to TRUE.

The timer `uiDelayLPAlarm` is started when the low-pressure `iLPValue` is lower than the low-pressure limit `iLPMinLimit`.

A low-pressure alarm can be detected when the low-pressure alarm detection is enabled.

A low-pressure alarm is detected when the input `iLPValue` is below the low- pressure limit (`iLPMinLimit`) and the timer `uiDelayLPAlarm` has elapsed.

NOTE: The detected low-pressure alarm can be reset when the input `iLPValue` is above the low-pressure limit (`iLPMinLimit`) and the reset input `xReset` is set to TRUE.



Alarm Reset

When the input `xReset` is set to TRUE, the outputs `uiAlarmID` and `xAlarm` are set to 0 if the following alarms are not active:

- motor alarm
- oil pressure switch alarm
- high-pressure switch alarm
- emergency stop alarm
- generic alarm
- high-pressure alarm

The outputs `uiAlarmID` and `xAlarm` can be reset even when the following alarms are active:

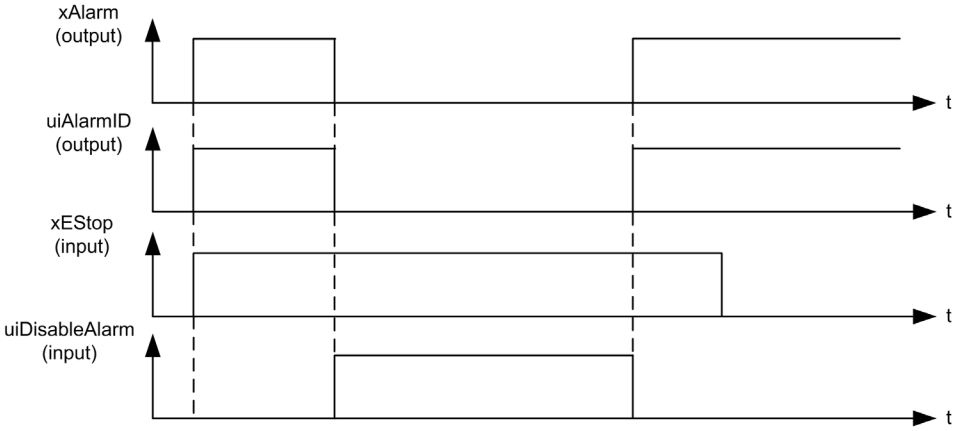
- low-pressure switch alarm
- low-pressure alarm

Alarm Disable `uiDisableAlarm`

An alarm can be disabled by setting the bit of the input `uiDisableAlarm` to TRUE. The bits and the alarms are described in the following table:

uiDisableAlarm	**Alarm**
`uiDisableAlarm.0` TRUE	Motor alarm is inactive.
`uiDisableAlarm.1` TRUE	Low pressure switch alarm is inactive.
`uiDisableAlarm.2` TRUE	High pressure switch alarm is inactive.
`uiDisableAlarm.3` TRUE	Low pressure switch alarm is inactive.
`uiDisableAlarm.4` TRUE	Emergency stop alarm is inactive.
`uiDisableAlarm.5` TRUE	Generic alarm is inactive.
`uiDisableAlarm.6` TRUE	High pressure alarm is inactive.
`uiDisableAlarm.7` TRUE	Low pressure alarm is inactive.

The following graphic presents an example for alarm disable of an emergency stop alarm:



NOTE: If an alarm is active when being disabled, the outputs `uiAlarmID` and `xAlarm` and the timers are reset.

Alarm ID Description

The `uiAlarmID` output represents a value between 0 and 255, whereby each bit represents a detected alarm. The table contains the bits and their description:

Alarm Bit	Alarm Cause	Effect
0	Motor alarm is active.	The compressor is stopped.
1	Oil pressure switch alarm is active.	
2	High pressure switch alarm is active.	
3	Low pressure switch alarm is active.	
4	Emergency stop is active.	
5	Generic alarm is active.	
6	High pressure alarm is active.	
7	Low pressure alarm is active.	
8-15	not used	N/A

Troubleshooting

In case of an alarm the compressor is stopped and the function block goes into alarm state. If the compressor goes into normal state, it is automatically restarted after the timers `uiMinCycleTime` and `uiMinOffTime` have elapsed:

Alarm	Problem	Solution
`uiAlarmID.0` TRUE	Motor alarm is active.	Verify the state of the circuit breaker or of the thermo switch of the compressor.
`uiAlarmID.1` TRUE	Oil pressure switch alarm is active.	Verify the state of the oil pressure switch.
`uiAlarmID.2` TRUE	High pressure switch alarm is active.	Verify the state of the high pressure switch.
`uiAlarmID.3` TRUE	Low pressure switch alarm is active.	Verify the state of the low pressure switch.
`uiAlarmID.4` TRUE	Emergency stop is active.	Verify the state of the emergency stop.
`uiAlarmID.5` TRUE	Generic alarm is active.	Verify the state of the generic alarm.
`uiAlarmID.6` TRUE	High pressure alarm is active.	Verify the value of the high pressure.
`uiAlarmID.7` TRUE	Low pressure alarm is active.	Verify the value of the low pressure.

Chapter 5
Compressor Application Limits: CompAppLimit

What Is in This Chapter?

This chapter contains the following sections:

Section 5.1
Functional Overview

`CompAppLimit` Function Block Description

Function Block Description

The function block monitors the application limits of a compressor. The evaporation temperature (To) and condensation temperature (Tc), which defines the operating point, must be inside defined limits. A limit alarm will be generate when the operating point is outside of the defined limits.

NOTE: The parameters must be set according to the specification of the compressor builder. Compressor operation is possible within the limits presented on the application diagrams. Compressor application limits should be chosen for design purposes or continuous operation. Restrictions to the operating limits may occur when using frequency converters.

Why Use the `CompAppLimit` Function Block?

For some processes, it is needed to control the operating point of compressors, for example, chiller.

Features of the `CompAppLimit` Function Block

The `CompAppLimit` function block provides the following features:

- Creating an alarm if the compressor is running beyond the boundary limits.
- Supports many application limits.

Error Avoidance Features

The function block provides the following error avoidance feature to help avoid certain sources of machine malfunction:

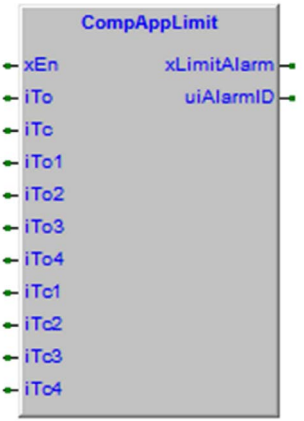
Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.

Section 5.2
Pin Description

Pin Description

Pin Diagram

The following picture presents the pin diagram of the `CompAppLimit` function block:



Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
xEn	BOOL	TRUE or FALSE	N/A	Enable/disable the function block.
iTo	INT	-32768...+32767	0.1 °C	Evaporation temperature
iTc	INT	-32768...+32767	0.1 °C	Condensation temperature
iTo1	INT	-1000...+1000	0.1 °C	Evaporation temperature operation limit 1
iTo2	INT	-1000...+1000	0.1 °C	Evaporation temperature operation limit 2
iTo3	INT	-1000...+1000	0.1 °C	Evaporation temperature operation limit 3
iTo4	INT	-1000...+1000	0.1 °C	Evaporation temperature operation limit 4

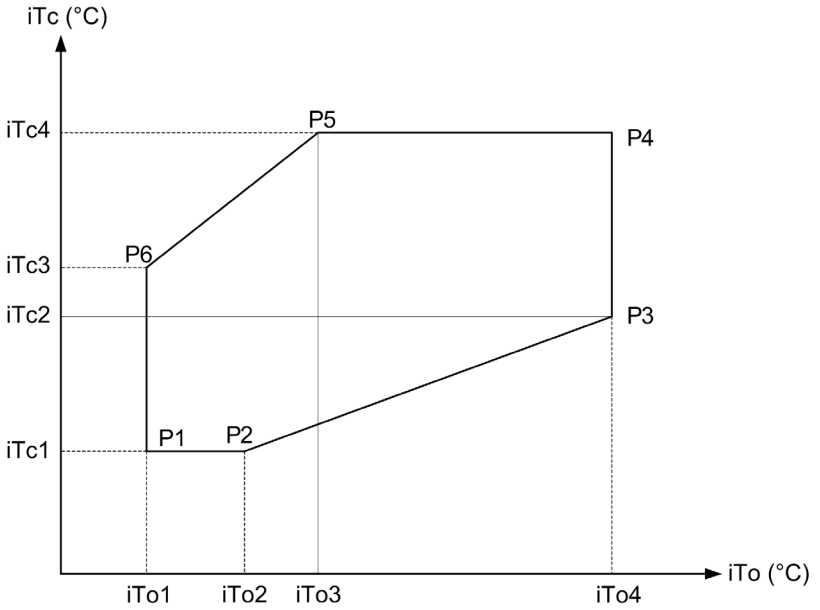
Input	Data Type	Range	Scaling / Unit	Description
iTc1	INT	-1000...+1000	0.1 °C	Condensation temperature operation limit 1
iTc2	INT	-1000...+1000	0.1 °C	Condensation temperature operation limit 2
iTc3	INT	-1000...+1000	0.1 °C	Condensation temperature operation limit 3
iTc4	INT	-1000...+1000	0.1 °C	Condensation temperature operation limit 4

Temperature Limits

The polygon has to be defined by

- 6 points: P1...P6
- 8 parameter inputs:
  - iTo1...iTo4
  - iTc1...iTc4

The following standard application limit will be provided:



The 6 points are defined as described below:

P1: iTo1, iTc1

P2: iTo2, iTc1

P3: iTo4, iTc2
P4: iTo4, iTc4
P5: iTo3, iTc4
P6: iTo1, iTc3

NOTE:

- It is possible to set identical coordinates (e. g. P1=P2) to set other supported application limits.
- The parameters must be set as described below:
  - iTo1<=iTo2
  - iTo2<=iTo4
  - iTo3<=iTo4
  - iTo1<=iTo4
  - iTo1<=iTo3
  - iTc1<=iTc2
  - iTc2<=iTc4
  - iTc3<=iTc4

Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
xLimitAlarm	BOOL	TRUE or FALSE	N/A	FALSE = iTo and iTc are inside the operation limits. TRUE = iTo and iTc are outside the operation limits or uiAlarmID is greater than 0.
uiAlarmID	UINT	N/A	N/A	Alarm ID

Alarm ID Description

The `uiAlarmID` output represents a value from 0 to 3, whereby each bit represents a detected alarm. The table contains the bits and their description:

Alarm Bit	Alarm Cause	Effect
0	Invalid temperature operation limit, for example, `iTo1>iTo2`.	`xLimitAlarm` is set to TRUE.
1	Invalid input values are set, for example, `iTo1`>+100.0.	`xLimitAlarm` is set to TRUE.
2...15	Not used	N/A

NOTE: The alarms are automatically reset when the parameters are set within their ranges.

Troubleshooting

Alarm	Problem	Solution
`uiAlarmID.0`	Invalid temperature operation limit, for example, `iTo1>iTo2`.	Set the parameters within the valid range.
`uiAlarmID.1`	Invalid input values are set, for example, `iTo1`>+100.0.	Set the parameters within the valid range.

Chapter 6
Compressor Control for Generic On/Off Compressors: CompCntrl_OnOff

What Is in This Chapter?

This chapter contains the following sections:

Section 6.1
Functional Overview

`CompCntrl_OnOff` Function Block Description

Function Block Description

The `CompCntrl_OnOff` function block controls 1 on-off compressor.

This function block can be used with fix speed scroll, piston (1 stage reciprocal compressor) and screw compressors.

NOTE: This function block must be used together with the function block `CompAlarmMgmt` to help prevent damages of the compressor.

Why Use the `CompCntrl_OnOff` Function Block?

The `CompCntrl_OnOff` provides the following purposes:

- The compressor can be operated in automatic, manual or in maintenance mode.
- The integrated timers help to prevent the compressor from frequent switching.

Features of the `CompCntrl_OnOff` Function Block

- 3 different operating modes *(see page 124)*: automatic, manual, maintenance
- Quick stop of the compressor
- Timers *(see page 124)*: `uiMinOnTime`, `uiMinOffTime`, `uiMinCycleTime`
- Display compressor operating hours
- Display compressor number of starts
- Display remaining time:
  - minimum on timer
  - minimum off timer
  - cycle timer

Error Avoidance Features

The function block provides the following error avoidance feature to help avoid certain sources of machine malfunction:

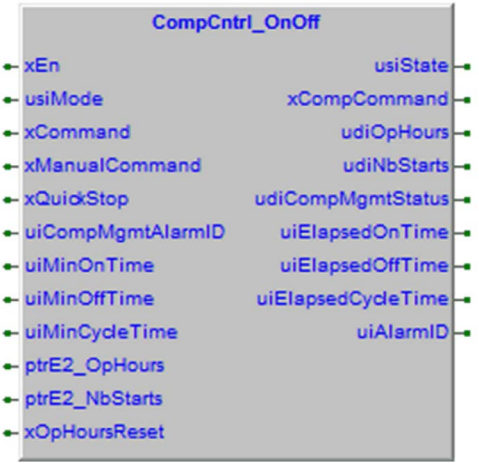
Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.

Section 6.2
Pin Description

Pin Description

Pin Diagram

The following picture presents the pin diagram of `CompCntrl_OnOff`:



Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
`xEn`	`BOOL`	TRUE or FALSE	N/A	Enables the function block. For more information refer to Enable *(see page 126)*.
`usiMode`	`USINT`	1…3	N/A	Mode control of the compressor: ● 1: automatic ● 2: manual ● 3: maintenance For more information refer to Operating Mode *(see page 124)*.
`xCommand`	`BOOL`	TRUE or FALSE	N/A	Command of the compressor in automatic mode

Input	Data Type	Range	Scaling / Unit	Description
xManualCommand	BOOL	TRUE or FALSE	N/A	Command of the compressor in manual or in maintenance mode (optional).
xQuickStop	BOOL	TRUE or FALSE	N/A	Quick stop of the compressor **NOTE:** If the input xQuickStop is set to TRUE, the compressor is stopped even if the timer uiMinOnTime has not elapsed.
uiCompMgmtAlarmID	UINT	0…65535	N/A	AlarmID input from the function block CompAlarmMgmt.
uiMinOnTime	UINT	0…65535	s	Minimum time the compressor is running.
uiMinOffTime	UINT	0…65535	s	Minimum time the compressor is stopped.
uiMinCycleTime	UINT	0…65535	s	Minimum time between 2 consecutive starts of the compressor.
ptrE2_OpHours	@UDINT	N/A	N/A	Pointer to variables defined in the non-volatile memory for operating hours.
ptrE2_NbStarts	@UDINT	N/A	N/A	Pointer to variables defined in the non-volatile memory for number of starts.
xOpHoursReset	BOOL	TRUE or FALSE	N/A	Reset the operating hours udiOpHours and the number of starts udiNbStarts.

Output Pin Description

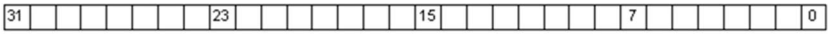
Output	Data Type	Range	Scaling / Unit	Description
usiState	USINT	0… 255	N/A	Current state: 1: idle 20: Run 99: Alarm
xCompCommand	BOOL	TRUE or FALSE	N/A	Compressor command
udiOpHours	UDINT	0… 4294967295	hours	Total number of operating hours
udiNbStarts	UDINT	0… 4294967295	N/A	Total number of starts
udiCompMgmtStatus	UDINT	0… 4294967295	N/A	Status of the compressor management

Output	Data Type	Range	Scaling / Unit	Description
uiElapsedOnTime	UINT	0… 65535	s	Remaining time before the minimum On time elapsed.
uiElapsedOffTime	UINT	0… 65535	s	Remaining time before the minimum Off time elapsed.
uiElapsedCycleTime	UINT	0… 65535	s	Remaining time before the minimum cycle time elapsed.
uiAlarmID	UINT	0… 65535	N/A	Alarm ID

Non-Volatile Memory Variables

Address	Variable	Data Type	Range	Scaling / Unit	Description
N	E2_OpHours	UDINT	0...4294967295	hours	Total number of operating hours
N+1	E2_NbStarts	UDINT	0...4294967295	N/A	Total numbers of starts

udiCompMgmtStatus



Bit 0-23 operating hours (maximum 16777215 hours)
Bit 24: reserved
Bit 25: reserved
Bit 26-27: compressor mode 1 = auto, 2 = manual, 3 = maintenance
Bit 28: On timer active
Bit 29: Off timer active
Bit 30: Cycle timer active
Bit 31: Alarm

The following table details the compressor mode desciption for bit 26 and bit 27:

Bit 26	Bit 27	Compressor mode description
0	0	Compressor FB disabled
0	1	Auto
1	0	Manual
1	1	Maintenance

Alarm ID Description

The `uiAlarmID` output represents a value between 0 and 3, whereby each bit represents a detected alarm. The table contains the bits and their description:

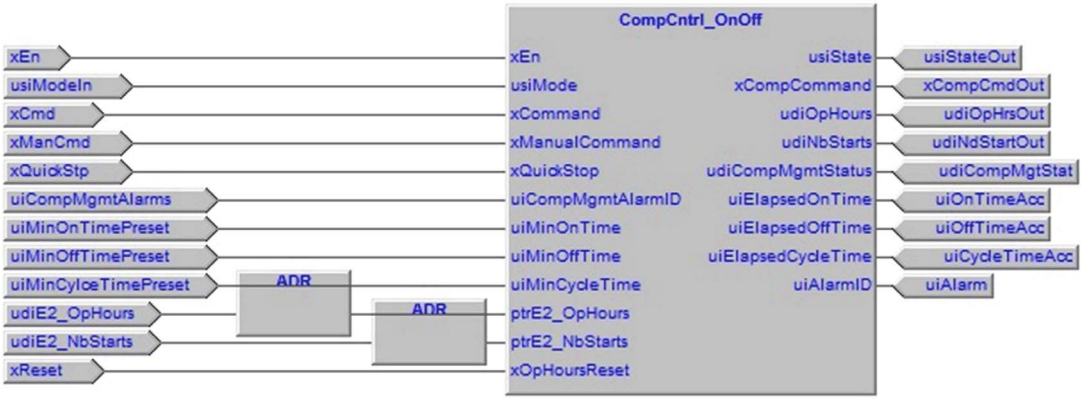
Alarm Bit	Alarm Cause	Effect
0	The value of the input `usiMode` is invalid.	The compressor is stopped.
1	`uiCompMgmtAlarmID` input is not equal to 0.	The compressor is stopped.
2-15	not used	N/A

Troubleshooting

In case of an alarm the compressor is stopped and the function block goes into alarm state. If the compressor goes into normal state, it is automatically restarted after the timers `uiMinCycleTime` and `uiMinOffTime` have elapsed:

Alarm	Problem	Solution
`uiAlarmID.0` TRUE	The value of the input `usiMode` is invalid.	Check the parameter ranges. Set the value within the defined range.
`uiAlarmID.1` TRUE	`uiCompMgmtAlarmID` input is not equal to 0.	Verify the alarms of the function block `CompAlarmMgmt`.

Connectivity Diagram



NOTE:

- You have to define the parameters of the data type specified in the table *Non-Volatile Memory Variables* in the non-volatile memory parameters in the application.
- You have to use an `ADR` block to connect the following input pins to the variable defined in the non-volatile memory:
  - `ptrE2_OpHours`
  - `ptrE2_NbStarts`
- If several instances of the function block are created inside the project, you have to create 8 new parameters in the non-volatile memory for each function block:
  - `E2_OpHours`
  - `E2_NbStarts`

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Be sure that all variables are initialized to an appropriate value before their first use as array indices or pointers.
- Write programming instructions to test the validity of operands intended to be used as array indices and memory pointers.
- Do not attempt to access an array element outside the defined bounds of the array.

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

Chapter 7
Compressor Control for Screw Compressor with Slider Capacity: CompCntrl_Slider

What Is in This Chapter?

This chapter contains the following sections:

Section 7.1
Functional Overview

`CompCntrl_Slider` Function Block Description

Function Block Description

The `CompCntrl_Slider` function block controls the operation and the cooling power of one screw compressor with modulating capacity control.

NOTE: This function block must be used together with the function block `CompAlarmMgmt` *(see page 90)* to help prevent potential damages of the compressor.

Why Use the `CompCntrl_Slider` Function Block?

The `CompCntrl_Slider` function block provides the following purposes:

- The compressor can be operated in automatic, manual or in maintenance mode.
- The integrated timers help to prevent the compressor from frequent switching.

Features of the `CompCntrl_Slider` Function Block

- 3 different operating modes *(see page 124)*: automatic, manual, maintenance
- Quick stop of the compressor
- Support part winding
- Internal temperature controller
- Timers *(see page 124)*: `uiMinOnTime`, `uiMinOffTime`, `uiMinCycleTime`
- Display compressor operating hours
- Display compressor number of starts
- Display remaining time;
  - minimum On timer
  - minimum Off timer
  - cycle timer

Error Avoidance Features

The function block provides the following error avoidance feature to help avoid certain sources of machine malfunction:

Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.

Section 7.2
Pin Description

Pin Description

Pin Diagram

The following picture presents the pin diagram of `CompCntrl_Slider`:

Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
xEn	BOOL	TRUE or FALSE	N/A	Enables the function block.
usiMode	USINT	1…3	N/A	Mode control of the compressor: • 1: automatic • 2: manual • 3: maintenance
iProcessValue	INT	-32768…+32767	0.1 User Units	Process value iProcessValue must have the same unit as iSetp and iManualSetp.
xCommand	BOOL	TRUE or FALSE	N/A	Starts/stops the compressor in automatic mode.
iSetp	INT	-32768…+32767	0.1 User Units	Set-point in automatic mode
xManualCommand	BOOL	TRUE or FALSE	N/A	Starts/stops the compressor in manual or in maintenance mode (optional).
iManualSetp	INT	-32768…+32767	0.1 User Unit	Set-point in manual mode or in maintenance mode (optional)
xQuickStop	BOOL	TRUE or FALSE	N/A	Quick stop of the compressor **NOTE:** If the input xQuickStop is set to TRUE, the compressor is stopped even if the timer uiMinOnTime has not elapsed and the stopping mode (to move the piston to the initial position) is not active.
uiCompMgmtAlarmID	UINT	0…65535	N/A	AlarmID input from the function block CompAlarmMgmt.

Input	Data Type	Range	Scaling / Unit	Description
iDeadband	INT	-32768...+32767	0.1 User Units	Dead band around the set-point, where both CR3 and CR4 valves are closed. Default value: 2.0 °C (35.6 °F)
uiMinOnTime	UINT	0…65535	s	Minimum time the compressor is running.
uiMinOffTime	UINT	0…65535	s	Minimum time the compressor is stopped.
uiMinCycleTime	UINT	0…65535	s	Minimum time between 2 consecutive starts of the compressor.
uiDelayPartWinding	UINT	100…65535	ms	Delay for part winding for the start of the compressor. Default value: 500 ms
uiPulseTimeCR3	UINT	100…65535	ms	Pulse time of the valve CR3. Default value: 500 ms
uiPauseTimeCR3	UINT	100…65535	ms	Pause time of the valve CR3 (between 2 pulses). Default value: 500 ms
uiPulseTimeCR4	UINT	100…65535	ms	Pulse time of the valve CR4. Default value: 500 ms
uiPauseTimeCR4	UINT	100…65535	ms	Pause time of the valve CR4 (between 2 pulses). Default value: 500 ms
uiValve100Time	UINT	100…65535	ms	Time to move the piston from initial position to maximum position. Default value: 500 ms
ptrE2_OpHours	@UDINT	N/A	N/A	Pointer to variables defined in the non-volatile memory for operating hours.
ptrE2_NbStarts	@UDINT	N/A	N/A	Pointer to variables defined in the non-volatile memory for number of starts.

Input	Data Type	Range	Scaling / Unit	Description
xOpHoursReset	BOOL	TRUE or FALSE	N/A	Reset the operating hours udiOpHours and the number of starts udiNbStarts.

Operating Mode usiMode

The CompCntrl_Slider function block provides 3 operating modes:

Automatic mode, usiMode = 1	• The compressor is controlled with the inputs iSetp and xCommand. • The timers and the alarms are enabled.
Manual mode, usiMode = 2	• The compressor is controlled with the inputs iManualSetp and xManualCommand. • The timers and the alarms are enabled.
Maintenance mode, usiMode = 3	• The compressor is controlled with the inputs iManualSetp and xManualCommand. • The timers are disabled. • The alarms are enabled.

Timers uiMinOnTime, uiMinOffTime, uiMinCycleTime

The timers help prevent that the compressor is stopped and started too often. You can set these durations:

- uiMinOnTime
- uiMinOffTime
- uiMinCycleTime

NOTE: Refer to the compressor documentation for minimum recommended cycle times.

NOTE: When the compressor is started (xCommand or xManualCommand is set to TRUE) or stopped (xCommand or xManualCommand is set to FALSE), the compressor cannot be stopped or started until the timer uiMinOnTime or uiMinOffTime has elapsed or if the time duration during 2 starts of the compressor is lower than uiMinCycleTime.

Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
usiState	USINT	0…99	N/A	Current state: 1: idle 10: part winding 1 11: part winding 2 20: Run 30: Stop 99: Alarm

Output	Data Type	Range	Scaling / Unit	Description
xPartWinding1	BOOL	TRUE or FALSE	N/A	Command of part winding 1
xPartWinding2	BOOL	TRUE or FALSE	N/A	Command of part winding 2
xCommandCR3	BOOL	TRUE or FALSE	N/A	Command of valve CR3 **NOTE:** Use a solid-state relay to control the valve CR3.
xCommandCR4	BOOL	TRUE or FALSE	N/A	Command of valve CR4 **NOTE:** Use a solid-state relay to control the valve CR4.
udiOpHours	UDINT	0…4294967295	hours	Total number of operating hours
udiNbStarts	UDINT	0…4294967295	N/A	Total number of starts
udiCompMgmtStatus	UDINT	0…4294967295	N/A	Status of the FB
uiElapsedOnTime	UINT	0…65535	s	Remaining time before the minimum On time elapsed.
uiElapsedOffTime	UINT	0…65535	s	Remaining time before the minimum Off time elapsed.
uiElapsedCycleTime	UINT	0…65535	s	Remaining time before the minimum cycle time elapsed.
uiAlarmID	UINT	0…65535	N/A	Alarm identification

NOTE: Every time the compressor is started, the value `udiNbStarts` is incremented and the timer `udiOpHours` is started without reset.
When the input `xOpHoursReset` is set to TRUE, the number of starts and the operating hours are reset: the values `udiNbStarts` and `udiOpHours` are set to 0 and remain at 0, as long as the input `xOpHours` is set to TRUE.

Non-Volatile Memory Variables

Address	Variable	Data Type	Range	Scaling / Unit	Description
N	E2_OpHours	UDINT	0...4294967295	hours	Total number of operating hours
N+1	E2_NbStarts	UDINT	0...4294967295	N/A	Total numbers of starts

Enable

If the function block is disabled, the compressor is switched off and the function block is not executed.

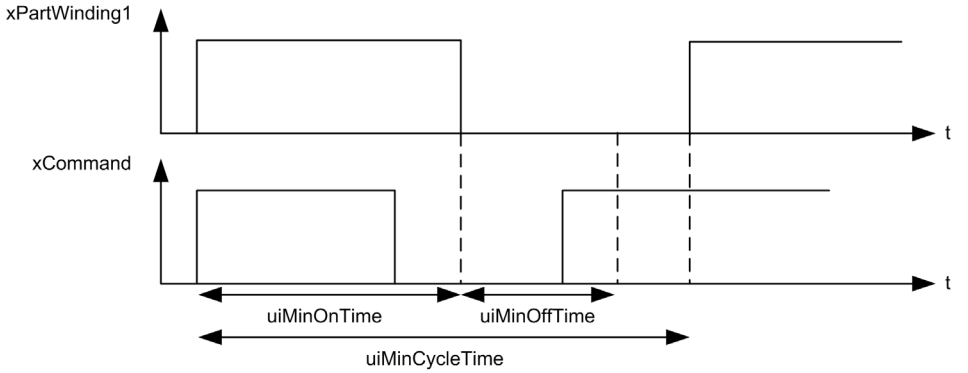
NOTE: After the function block is enabled, the compressor can be started even if the duration of the stop is lower than `uiMinOffTime` or if the duration between 2 consecutive starts is lower than `uiMinCycleTime`.

Normal Operation

To help avoid that the compressor is stopped and started too often, you can set the duration `uiMinOnTime`, `uiMinOffTime` and `uiMinCycleTime` with the appropriate values.

If...	Then...
the compressor is started	`xCommand` or `xManualCommand` is set to TRUE
the compressor is stopped	`xCommand` or `xManualCommand` is set to FALSE

NOTE: The compressor cannot be stopped or started until the timer `uiMinOnTime` or `uiMinOffTime` has elapsed or when the time duration during 2 starts of the compressor is lower than `uiMinCycleTime`.



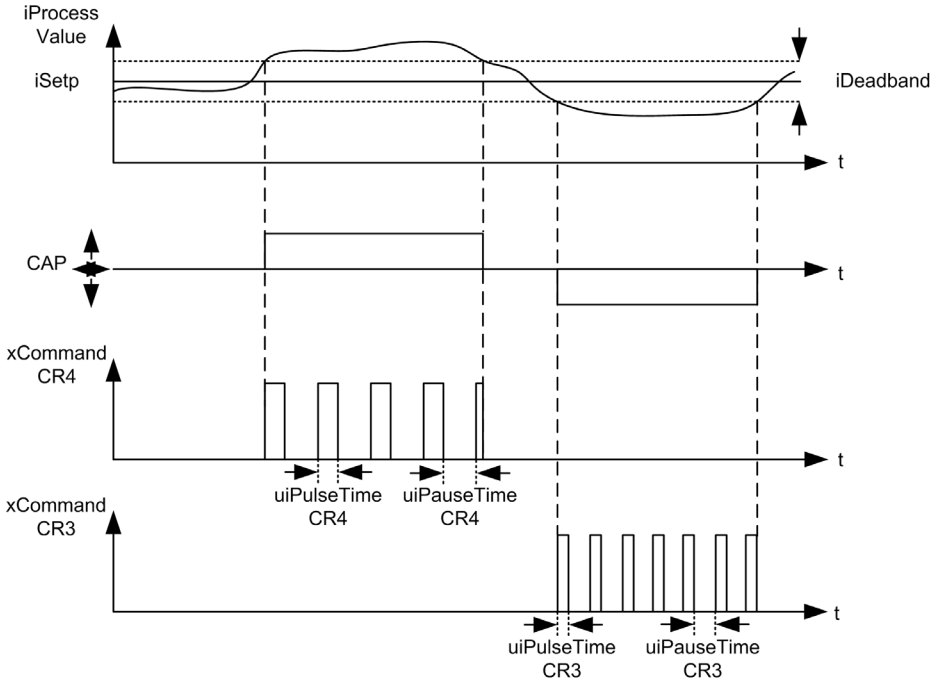
Operating Hours and Number of Starts

`udiNbStarts` accumulates the number of starts and the timer `udiOpHours` accumulates the operating hours when the compressor is started. These values should be retained even when the controller is power cycled.

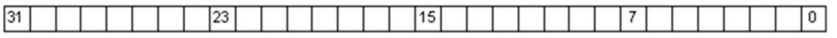
If the input `xOpHoursReset` is set to TRUE, the number of starts and the operating hours are reset: the values `udiNbStarts` and `udiOpHours` are set to 0 and remain set to 0 as long as the input `xOpHours` is set to TRUE.

Control of the Cooling Capacity

The following figure presents how the regulation of the cooling power is done internally by adjusting the cooling capacity of the compressor. `CAP` indicates if the cooling capacity must be increased or reduced.



udiCompMgmtStatus



Bit 0-23 operating hours (maximum 16.777.216 hours)
Bit 24: reserved
Bit 25: reserved
Bit 26-27: compressor mode 1 = auto, 2 = manual, 3 = maintenance
Bit 28: On timer active
Bit 29: Off timer active
Bit 30: Cycle timer active
Bit 31: Alarm

The following table details the compressor mode desciption for bit 26 and bit 27:

Bit 26	Bit 27	Compressor mode description
0	0	Compressor FB disabled
0	1	Auto
1	0	Manual
1	1	Maintenance

Alarm ID Description

The `uiAlarmID` output represents a value from 0 to 3, whereby each bit represents a detected alarm. The table contains the bits and their description:

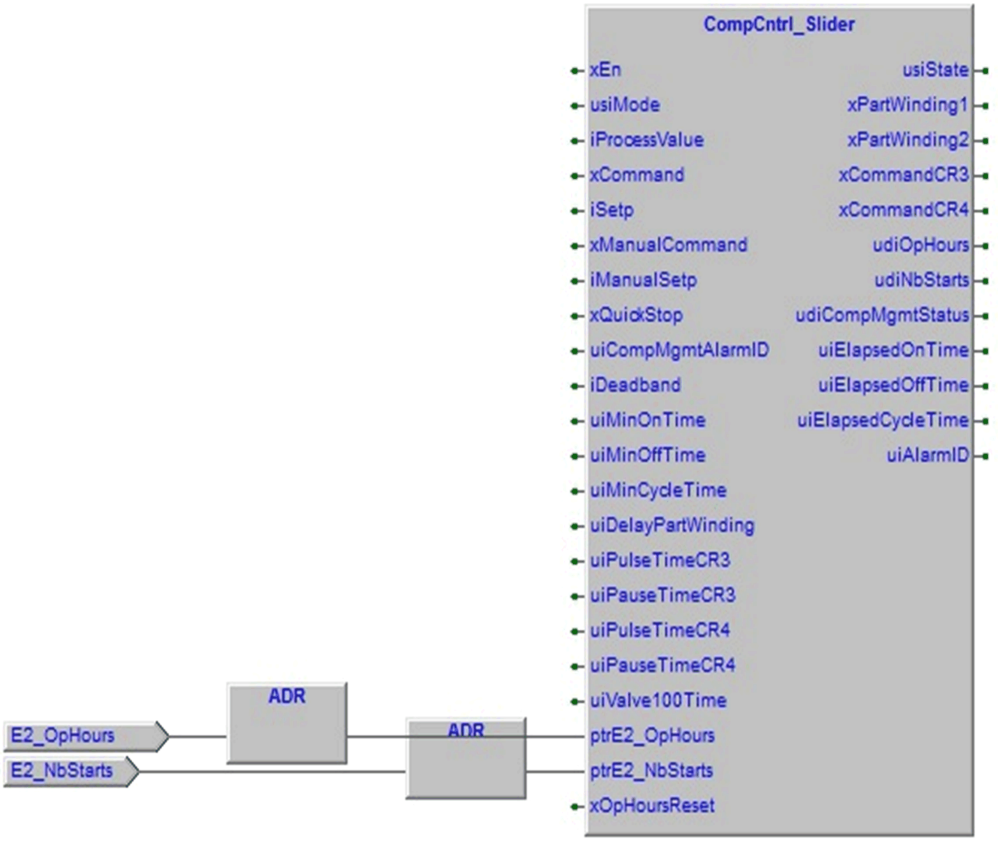
Alarm Bit	Alarm Cause	Effect
0	The value of the input `usiMode` is invalid.	The compressor is stopped.
1	`uiCompMgmtAlarmID` input is not equal to 0.	The compressor is stopped.
2-15	not used	N/A

Troubleshooting

In case of an alarm the compressor is stopped and the function block goes into alarm state. If compressor goes into normal state, it is automatically restarted after the timers `uiMinCycleTime` and `uiMinOffTime` have elapsed:

Alarm	Problem	Solution
`uiAlarmID.0` TRUE	The value of the input `usiMode` is invalid.	Check the parameter ranges. Set the value within the defined range.
`uiAlarmID.1` TRUE	`uiCompMgmtAlarmID` input is not equal to 0.	Verify the alarms of the function block `CompAlarmMgmt`.

Connectivity Diagram



NOTE:

- You have to define the parameters of the data type specified in the table *Non-Volatile Memory Variables* in the non-volatile memory parameters in the application.
- You have to use an `ADR` block to connect the following input pins to the variable defined in the non-volatile memory:
  - `ptrE2_OpHours`
  - `ptrE2_NbStarts`
- If several instances of the function block are created inside the project, you have to create 8 new parameters in the non-volatile memory for each function block:
  - `E2_OpHours`
  - `E2_NbStarts`

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Be sure that all variables are initialized to an appropriate value before their first use as array indices or pointers.
- Write programming instructions to test the validity of operands intended to be used as array indices and memory pointers.
- Do not attempt to access an array element outside the defined bounds of the array.

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

Chapter 8
Compressor Control for Variable Speed Compressors: CompCntrl_VS

What Is in This Chapter?

This chapter contains the following sections:

Section 8.1
Functional Overview

CompCntrl_VS Function Block Description

Function Block Description

The CompCntrl_VS function block controls 1 scroll, screw or reciprocating compressor through variable speed drive in order to control compressor capacity.

NOTE: This function block must be used together with the function block CompAlarmMgmt to help prevent damages of the compressor.

Why Use the CompCntrl_VS Function Block?

The CompCntrl_VS provides the following purposes:

- The compressor can be operated in automatic, manual or in maintenance mode.
- The integrated timers help to prevent the compressor from frequent switching.
- The CompCntrl_VS function block provides an oil recovery function which increases temporarily the variable speed drive frequency to circulate oil through the compressor.
- The CompCntrl_VS function block suppresses resonance frequencies to reduce noise and increase compressor life time.

Features of the CompCntrl_VS Function Block

- 3 different operating modes *(see page 124)*: automatic, manual, maintenance
- Quick stop of the compressor
- Start/stop procedure
- Oil recovery *(see page 137)*
- Timers *(see page 124)*: uiMinOnTime, uiMinOffTime, uiMinCycleTime
- Eliminate resonance frequency range
- Display compressor operating hours
- Display compressor number of starts
- Display remaining time:
  - minimum On timer
  - minimum Off timer
  - cycle timer

Error Avoidance Features

The function block provides the following error avoidance feature to help avoid certain sources of machine malfunction:

Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.

Section 8.2
Pin Description

Pin Description

Pin Diagram

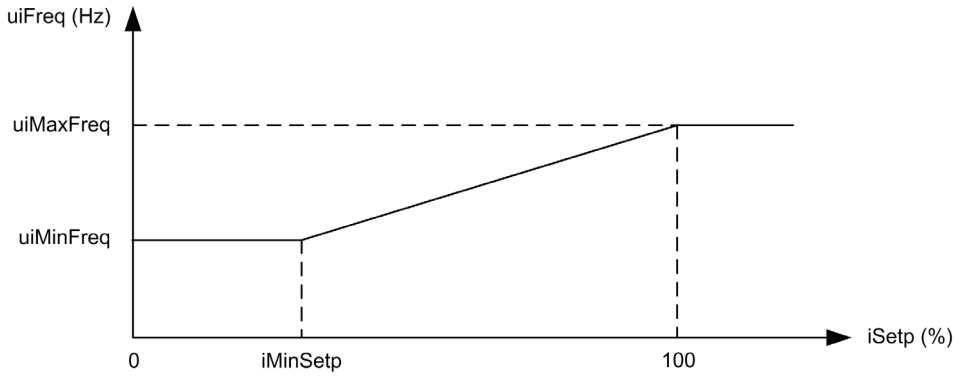
The following picture presents the pin diagram of `CompCntrl_VS`:

Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
xEn	BOOL	TRUE or FALSE	N/A	Enables the function block.
usiMode	USINT	1…3	N/A	Mode control of the compressor: ● 1: automatic ● 2: manual ● 3: maintenance For more information, refer to Operating Mode *(see page 124)*.
xCommand	BOOL	TRUE or FALSE	N/A	Command to start or stop the compressor regulation.
xManualCommand	BOOL	TRUE or FALSE	N/A	Command to start the compressor in manual mode.
xQuickStop	BOOL	TRUE or FALSE	N/A	Quick stop of the compressor **NOTE:** If the input xQuickStop is set to TRUE, the compressor is stopped even if the timer uiMinOnTime has not elapsed.
iSetp	INT	0… 1000	0.1 %	Set-point in automatic mode
iManualSetp	INT	0… 1000	0.1 %	Set-point in manual or maintenance mode (optional)
iMinSetp	INT	0…1000	0.1 %	Minimum set-point that corresponds to the uiMinFequency. **NOTE:** iMinSetp must be lower than 100.0%. For more information, refer to the graphic after this table.
uiCompMgmtAlarmID	UINT	0…65535	N/A	AlarmID input from the function block CompAlarmMgmt.
uiStartingTime	UINT	0…65535	s	Duration of the starting mode
uiStoppingTime	UINT	0…65535	s	Duration of the stopping mode
uiOilRecoveryTime	UINT	0…65535	min	Maximum time oil recovery mode will be started.
uiOilRecoveryPeriod	UINT	0…65535	s	Duration of the oil recovery mode
uiMinOnTime	UINT	0…65535	s	Minimum time the compressor is running.
uiMinOffTime	UINT	0…65535	s	Minimum time the compressor is stopped.

Input	Data Type	Range	Scaling / Unit	Description
uiMinCycleTime	UINT	0…65535	s	Minimum time between 2 consecutive starts of the compressor.
uiMinFreq	UINT	0…5000	0.1 Hz	Minimum frequency of the compressor that corresponds to the minimum set-point. **NOTE:** uiMinFreq must be lower than uiMaxFreq.
uiMaxFreq	UINT	0…5000	0.1 Hz	Maximum frequency of the compressor
ptrArr_FreqLimits	@UINT	N/A	N/A	Pointer to array variable for frequency bands disabled to avoid resonance.
ptrE2_OpHours	@UDINT	N/A	N/A	Pointer to variables defined in the non-volatile memory for operating hours
ptrE2_NbStarts	@UDINT	N/A	N/A	Pointer to variables defined in the non-volatile memory for number of starts
xOpHoursReset	BOOL	TRUE or FALSE	N/A	Reset the operating hours udiOpHours and the number of starts udiNbStarts.

The following graphic presents the iMinSetp:

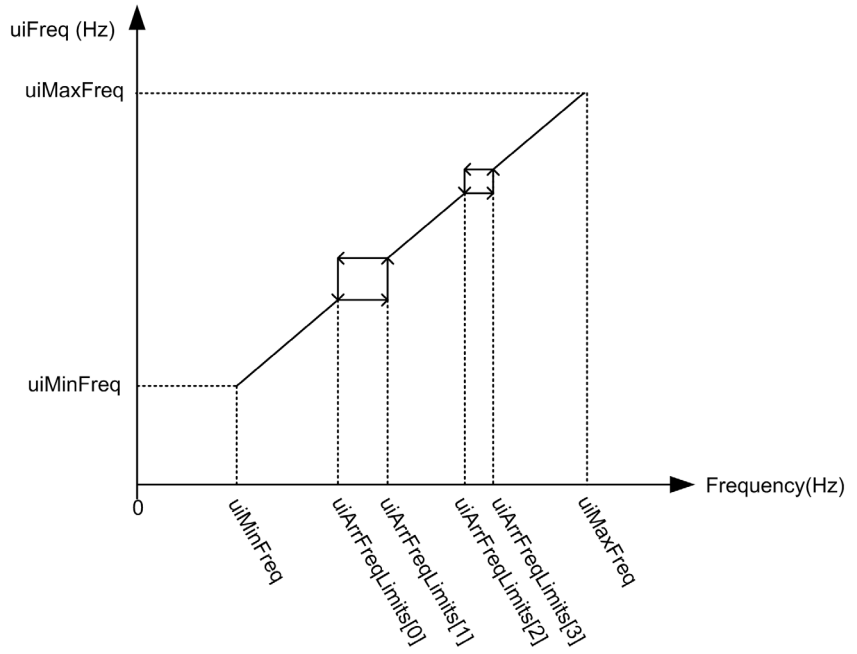


Frequency Limitations

The frequency of the variable speed drive is limited by the minimum frequency uiMinFreq and by the maximum frequency uiMaxFreq.

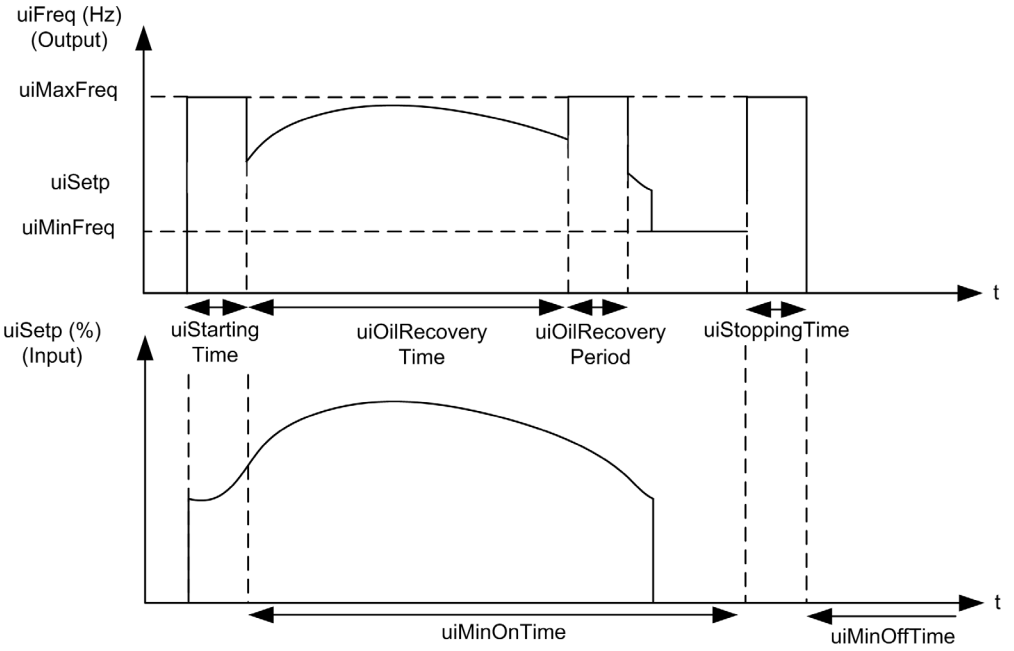
NOTE: `uiMinFreq` must be lower than `uiMaxFreq`.

The following graphic presents the resonance frequencies exclusion:



Oil Recovery

If...	Then...
the compressor is started,	`xCompCommand` is set to TRUE and `uiFreq` is set to `uiMaxFreq` during the duration of `uiStartingTime`.
the compressor has run for the set time `uiOilRecoveryTime`,	`uiFreq` is set to `uiMaxFreq` during the duration `uiOilRecoveryPeriod`.
the compressor has the order to stop,	`uiFreq` is set to `uiMaxFreq` during the duration `uiStoppingTime`.
`uiStoppingTime` has elapsed,	`xCompCommand` and `uiFreq` are set to 0.



Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
usiState	USINT	0… 255	N/A	Present state: 1: Idle 10: Starting 20: Run 30: Oil recovery 40: Stopping 99: Alarm
xCompCommand	BOOL	TRUE or FALSE	N/A	Compressor command
uiFreq	UINT	0… 5000	0.1 Hz	Frequency set-point for the variable speed drive.
udiOpHours	UDINT	0… 4294967295	hours	Total number of operating hours
udiNbStarts	UDINT	0… 4294967295	N/A	Total number of starts

Output	Data Type	Range	Scaling / Unit	Description
udiCompMgmtStatus	UDINT	0… 4294967295	N/A	Status of the function block. **NOTE:** For more information on the compressor mode refer to udiCompMgmtStatus *(see page 139)*.
uiElapsedOnTime	UINT	0… 65535	s	Remaining time before the minimum On time elapsed.
uiElapsedOffTime	UINT	0… 65535	s	Remaining time before the minimum Off time elapsed.
uiElapsedCycleTime	UINT	0… 65535	s	Remaining time before the minimum cycle time elapsed.
uiAlarmID	UINT	0… 65535	N/A	Alarm ID

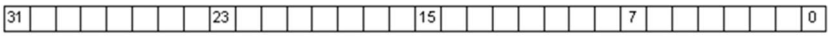
Non-Volatile Memory Variables

Address	Variable	Data Type	Range	Scaling / Unit	Description
N	E2_OpHours	UDINT	0...4294967295	hours	Total number of operating hours
N+1	E2_NbStarts	UDINT	0...4294967295	N/A	Total numbers of starts

Array Variables

Variable	Data Type	Array	Range	Scaling / Unit	Description
uiArrFreqLimits	UINT	0...9	0...5000	0.1 Hz	Frequency bands disabled to avoid resonance.

udiCompMgmtStatus



Bit 0-23 operating hours (maximum 16.777.216 hours)
Bit 24: reserved
Bit 25: reserved
Bit 26-27: compressor mode 1 = auto, 2 = manual, 3 = maintenance
Bit 28: On timer active
Bit 29: Off timer active
Bit 30: Cycle timer active
Bit 31: Alarm

The following table details the compressor mode desciption for bit 26 and bit 27:

Bit 26	Bit 27	Compressor mode description
0	0	Compressor FB disabled
0	1	Auto
1	0	Manual
1	1	Maintenance

Normal Operation with `xCompCommand` and `uiFreq`

The output `xCompCommand` enables the variable speed drive and `uiFreq` is the frequency set-point for the variable speed drive.

The output `xCompCommand` is controlled by the input `xCommand`.

If...	Then...
`xCommand` is set to 1	the output `xCompCommand` is set to 1 and the variable speed drive frequency `uiFreq` is controlled.
`iSetp` is lower than `iMinSetp`	the output `uiFreq` is set to the minimum frequency `uiMinFreq`.
`xCommand` is set to 0	the output `xCompCommand` is set to 0 and the variable speed drive frequency `uiFreq` is set to 0.

NOTE: `iMinSetp` must be lower than 100.0%.

To help avoid that the compressor is stopped and started too often, you can set the duration: `uiMinOnTime`, `uiMinOffTime` and `uiMinCycleTime`.

If...	Then...
the timer `uiMinOffTime` has not elapsed	the compressor is not started.
the time duration during 2 starts of the compressor is lower than `uiMinCycleTime`	the compressor is not started.
the compressor has the order to stop and the `uiMinOnTime` has not elapsed	`uiFreq` is set to `uiMinFreq`.

Alarm ID Description

The `uiAlarmID` output represents a value from 0 to 15, whereby each bit represents a detected alarm. The table contains the bits and their description:

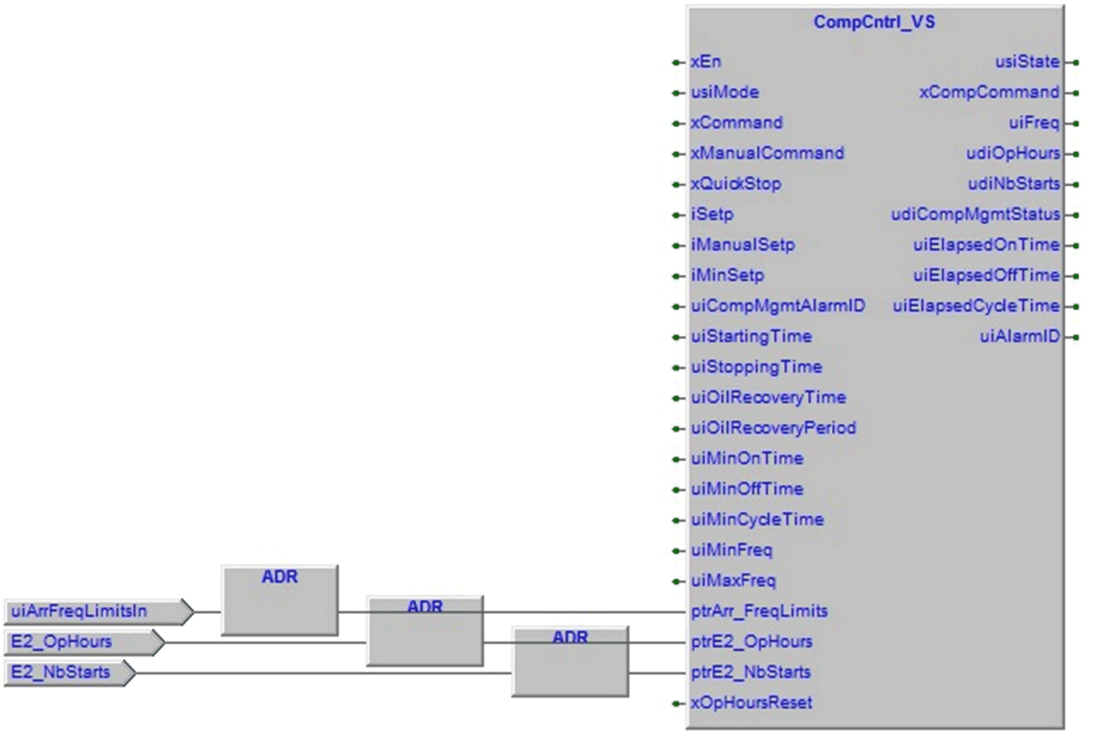
Alarm Bit	Alarm Cause	Effect
0	The parameter `uiMinFreq` is higher than the parameter `uiMaxFreq`.	The compressor is stopped.
1	The parameter `iMinSetp` is higher than 100.0 %.	
2	The value of the input `usiMode` is invalid.	
3	`uiCompMgmtAlarmID` input is not equal to 0.	
4-15	not used	N/A

Troubleshooting

In case of an alarm the compressor is stopped and the function block goes into alarm state. If the compressor goes into normal state, it is automatically restarted after the timers `uiMinCycleTime` and `uiMinOffTime` have elapsed:

Alarm	Problem	Effect
`uiAlarmID.0` TRUE	Parameter `uiMinFreq` is higher than the parameter `uiMaxFreq`.	Check the parameter ranges. Set the value within the defined range.
`uiAlarmID.1` TRUE	Parameter `iMinSetp` is higher than 100.0 %.	
`uiAlarmID.2` TRUE	The value of the input `usiMode` is invalid.	
`uiAlarmID.3` TRUE	`uiCompMgmtAlarmID` input is not equal to 0.	Verify the alarms of the function block `CompAlarmMgmt`.

Connectivity Diagram



NOTE:

- You have to define the parameters of the data type specified in the table *Non-Volatile Memory Variables* in the non-volatile memory parameters in the application.
- You have to use an `ADR` block to connect the following input pins to the variable defined in the non-volatile memory:
  - `ptrE2_OpHours`
  - `ptrE2_NbStarts`
- If several instances of the function block are created inside the project, you have to create 8 new parameters in the non-volatile memory for each function block:
  - `E2_OpHours`
  - `E2_NbStarts`

NOTE:

- You have to define the local or global variable of the data type specified in the table *Array Variables* with an array of 10 elements.
- You have to use an ADR block to connect the input pin `ptrArr_FreqLimits` to the array variable.

NOTE: During power cycle the array variable will be reset with the default value.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Be sure that all variables are initialized to an appropriate value before their first use as array indices or pointers.
- Write programming instructions to test the validity of operands intended to be used as array indices and memory pointers.
- Do not attempt to access an array element outside the defined bounds of the array.

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

Chapter 9
Compressor Management: CompMgmt

What Is in This Chapter?

This chapter contains the following sections:

Section 9.1
Functional and Machine Overview

What Is in This Section?

This section contains the following topics:

Functional Overview

Functional Description

The CompMgmt (Compressor Management) function block calculates the number of compressors required to control the water temperature or the refrigerant pressure.

In a refrigeration machine, compressors need to be managed in a way to help prolong their proper operation and balance machine lifetime.

The CompMgmt function block controls up to eight compressors and aims to manage the optimum functionality of compressors. For this purpose, CompMgmt provides features for compressor status management and to optimize operation.

The CompMgmt must be used together with the function block CompCntrl_OnOff which controls the operation of a single compressor.

Why Use the CompMgmt Function Block?

The CompMgmt function block is used for the following purposes:

Purpose	Description
Water temperature control or refrigerant pressure control	• maintain a constant water temperature or a constant refrigerant pressure • calculates the required number of compressors • can be used in cooling mode or heating mode
Runtime optimization	• increase the control accuracy • balance operating hours • avoid frequent On/Off switching of compressors
Status management	• switch off a compressor in case of a detected error and switch on another

Features of the CompMgmt Function Block

The CompMgmt function block provides the following features:

- supports 1 to 8 compressors
- switches on and off the number of compressors required to control the water temperature, or the refrigerant low pressure.
- balances compressor operating hours by using one of the 3 methods:
  - FIFO
  - Runtime
  - LIFO + Runtime
- prevents frequent switching of a compressor by On/Off time management
- compressor changeover management in case of a detected compressor error
- compressor maintenance notification: CompMgmt calculates the compressor operating hours
- can be used in cooling mode or heating mode.

Error Avoidance Features

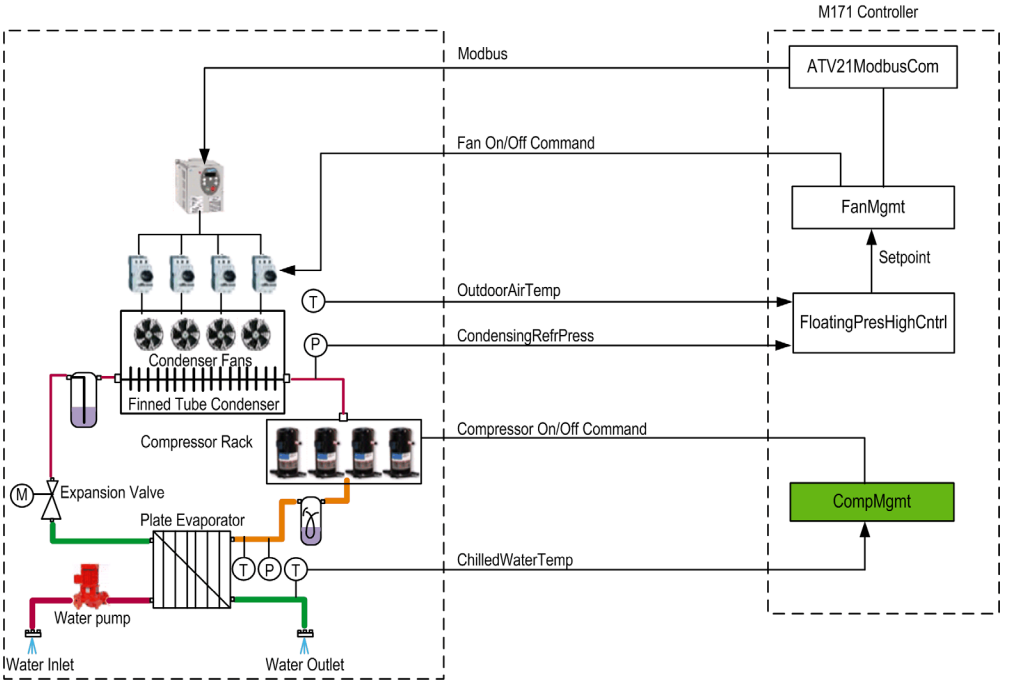
The `CompMgmt` function block provides the following error avoidance features to help you avoid the potentials of certain sources of machine malfunction:

Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.
Alarm/alert notification	If an invalid value is entered, an alarm or an alert is generated: ● Alarm: the compressors are switched off and the function block terminates in error. ● Alert: the function block keeps on operating, however with the possibility of reduced performance.
Controlled parameter	Parameters like `usiCompMode`, `usiNbComp`, `xHeatCool` and `xCntrlMode` are controlled. The configuration of these parameters can be changed, however the changes are effective only after the restart of the function block.

Machine Overview

Machine View

The following picture presents the interaction between the function block and the machine:



The `CompMgmt` function block controls the water temperature or the refrigerant pressure and calculates the required number of compresssors.

In this example, the chilled water temperature or the low pressure are controlled (cooling mode).

This function block controls up to 8 compressors and manages the switch on/off of compressors.

Section 9.2
Architecture

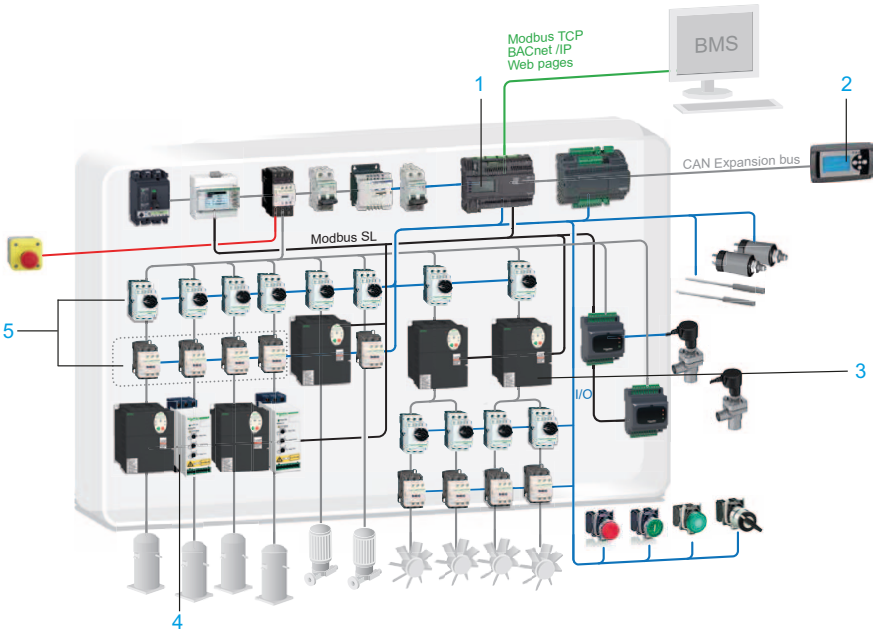
What Is in This Section?

This section contains the following topics:

Hardware Architecture

Hardware Architecture Overview

The following figure presents the hardware architecture of the Air Cooled Chiller:

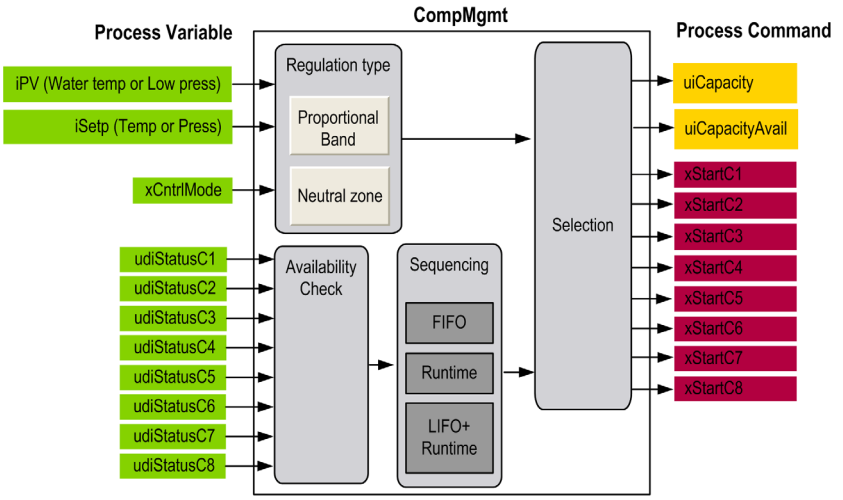


1 M171/M172 Controller
2 Graphic display
3 Variable speed drive ATV••/••• Modbus
4 Soft starters
5 Motor Control

Software Architecture

Function Block Diagram

The following function block diagram provides an overview of the software architecture:



The block diagram presents on the left hand side the inputs, the **Process Variables**, on the right hand side the outputs, the **Process Commands**, as well as the function block `CompMgmt`.

The following table gives you an overview of the functions of `CompMgmt`:

Function	Description
Regulation Type	defines the regulation type and the number of requested compressors based on the setpoint and the process value (water temperature or pressure)
Availability Check	• checks the availability for the configured number of compressors • evaluates compressor alarms • evaluates compressor minimum ON/Off time • evaluates compressor operation hours for switching on
Sequencing	determines, which compressor is switched on/off next based on three methods: • FIFO • Runtime • LIFO + Runtime
Selection	selects the compressor to be switched on next, based on Compressor Sequencing and Availability.

Section 9.3
Function Block Description

CompMgmt Function Block

Function Block Description

The CompMgmt function block controls up to 8 compressors. Operating hours are balanced using various methods. The CompMgmt status management function automatically commands a new compressor when an operating compressor is no longer available.

The following methods for compressor management are provided:

- Compressor Sequence Control
- Compressor Status Management
- Compressor On/Off Timing Control
- Compressor Selection Timing Control
- Compressor Operation Hours

Compressor Sequence Control

The purpose of the Compressor Sequence Control is to balance the number of operating hours and starts/stops between the compressors.

Compressor Sequence Control helps to ensure an even usage of the compressors and hence helps to protect the compressors and optimizes power consumption.

Compressors are controlled based on the following sequences:

Sequence as per CompMode	Description
FIFO = First In First Out	• The compressor with the least operating hours is switched on first. • The first compressor which is switched on is also the first to be switched off. • Advantage: operation time is limited.
Runtime	• The compressor with the least operating hours is the first compressor to be switched on. • The compressor with the greatest operating hours is the first compressor to be switched off. • Advantage: balanced operation hours

Sequence as per `CompMode`	Description
LIFO = Last In First Out + Runtime	• The parameter pointer `ptrArr_Priority` determines this sequence. • The first compressor to be switched on is the first one in the sequence. • The first compressor to be switched off is the last one that has been switched on. • Advantage: priority of compressor usage can be set. **NOTE:** If several compressors have the same priority, the compressor with the least operating hours is the first compressor to be switched on. The compressor with the greatest operating is the first compressor to be switched off.

NOTE:

- If a compressor is not available (Off timer or Cycle timer are active, or the compressor is in alarm state), another compressor is started based on the sequence defined by `usiCompMode`.
- If a compressor cannot be stopped (On timer is active), another compressor is stopped based on the sequence defined by `usiCompMode`.

Compressor Status Management

The compressor that is no longer available (non-operating) is switched off. The next available compressor in the start sequence will be switched on.

The non-operating compressor cannot be started until the compressor is returned to an operational state.

Cooling and Heating Mode

The heating-cooling mode is changed by the input `xHeatCool`.

The changes are applied, when the function block is restarted, which means a rising edge on `xEn`.

In cooling mode, the chilled water temperature or the refrigerant low pressure is controlled.

In heating mode, the warmed water temperature or the refrigerant high pressure is controlled.

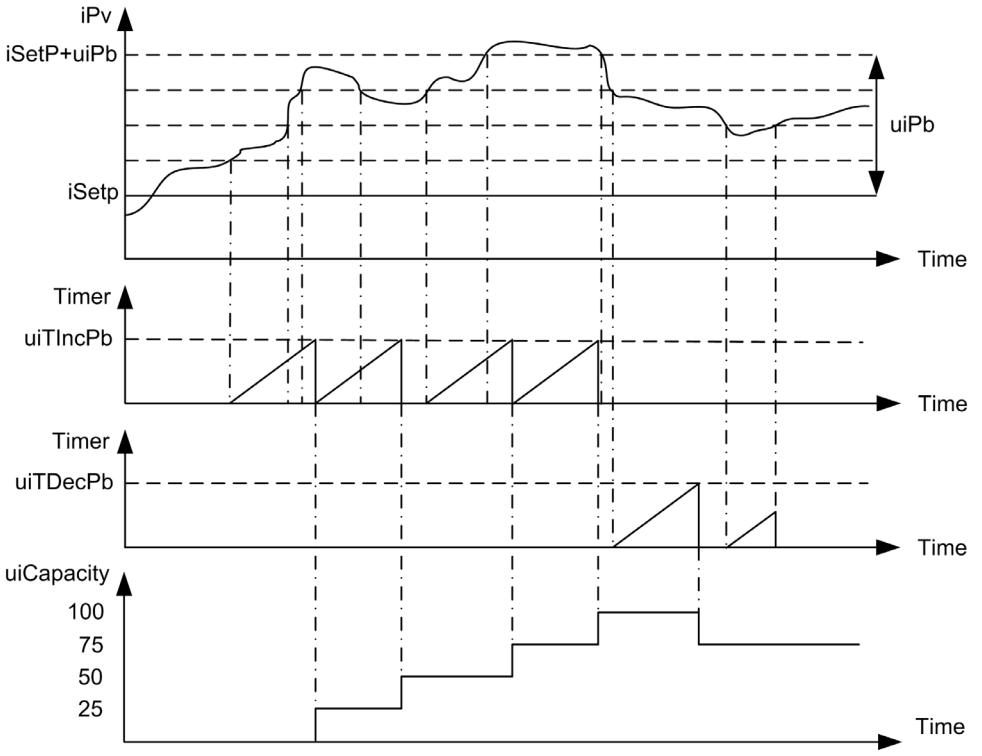
Regulation With Proportional Band

If the input `xCntrlMode` is set to TRUE, the regulation with proportional band is active.

The proportional band regulation is based on the proportional band `uiPb` and the timers can be set to define the incrementing time `uiTIncPb` and decrementing time `uiTDecPb` of the compressor.

For example in cooling mode:

If the temperature or the pressure `iPv` goes...	Then ...
above the `iSetp` + `uiPb`/`usiNbComp`,	the 1st compressor is started after the timer `uiTincPb` has elapsed.
above the `iSetp` + 2x(`uiPb`/`usiNbComp`),	the 2nd compressor is started after the timer `uiTincPb` has elapsed.
below the `iSetp` + `uiPb`/`usiNbComp`,	the 2nd running compressor is stopped after the timer `uiTDecPb` has elapsed.
below the `iSetp`,	the last running compressor is stopped after the timer `uiTDecPb` has elapsed.



NOTE: Only 1 compressor is started or stopped in the same time after the timer `uiTIncPb` or `uiTDecPb`.

NOTE: This regulation is enabled also for heating mode.

Regulation With Neutral Zone

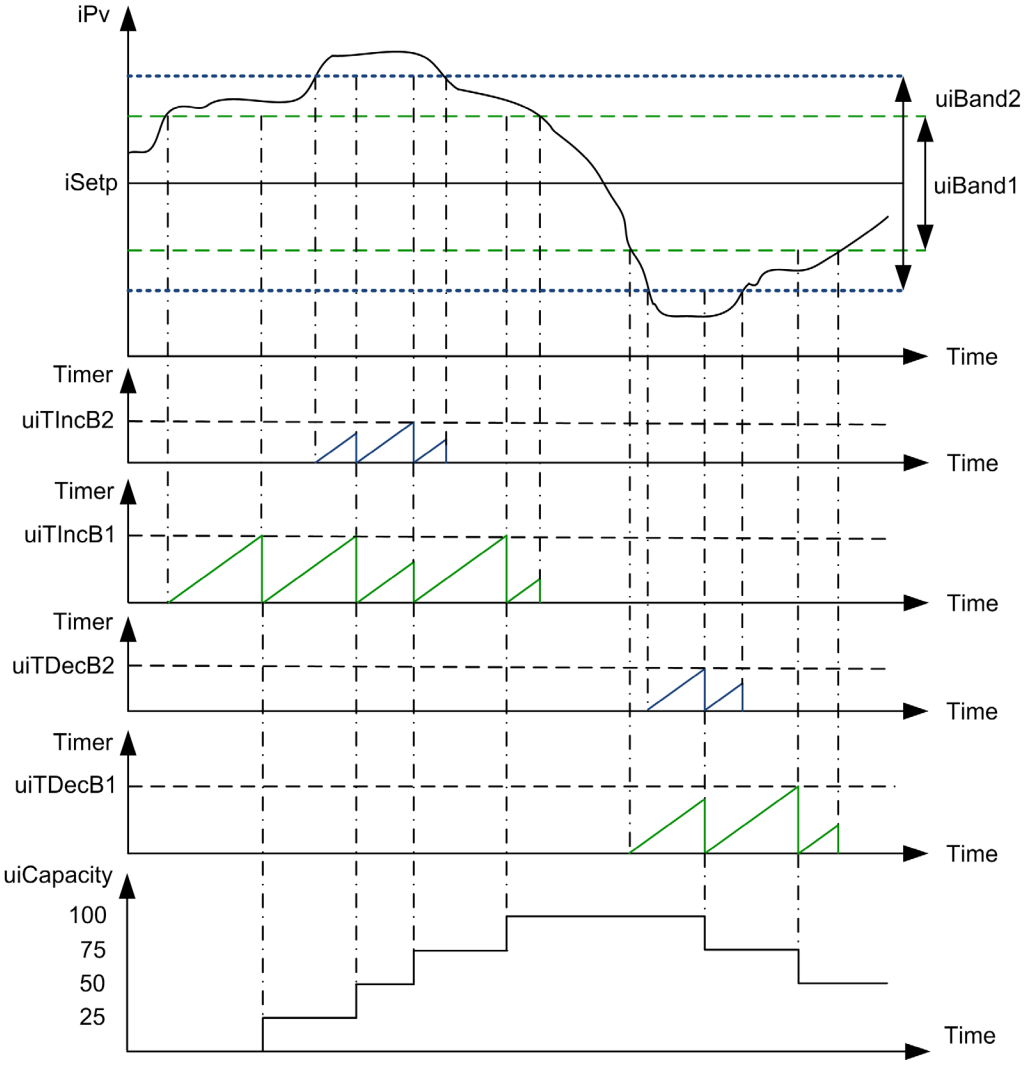
If the input `xCntrlMode` is set to FALSE, the regulation with neutral zone is active.

With the neutral zone regulation the number of running compressors remains constant when the process value `iPv` (temperature or pressure) remain inside the neutral zone.

The regulation with neutral zone is based on 2 neutral zones: `uiBand1` and `uiBand2` and timers can be set to define the incrementing time and decrementing time of the compressor for each band.

For example in cooling mode:

If the temperature or the pressure `iPv` goes...	Then ...
above the band `uiBand1`,	an extra compressor is started after the timer `uiTIncB1` has elapsed.
above the band `uiBand2`,	an extra compressor is started after the timer uiTIncB2 has elapsed.
below the band `uiBand1`,	a running compressor is stopped after the timer `uiTDecB1` has elapsed.
below the band `uiBand2`,	a running compressor is stopped after the timer `uiTDecB2` has elapsed.



NOTE: If `iPv` goes above the band `uiBand1` and before the timer `uiTIncB1` has elapsed, `iPv` goes above the band `uiBand2`, the timer `uiTIncB1` is not reset and the timer `uiTIncB2` is started.
After the 1st timer has elapsed (`uiTIncB1` or `uiTIncB2`), an extra compressor is started and the timers are reset and restart.

NOTE: This regulation is enabled also for heating mode.

Section 9.4
Pin Description

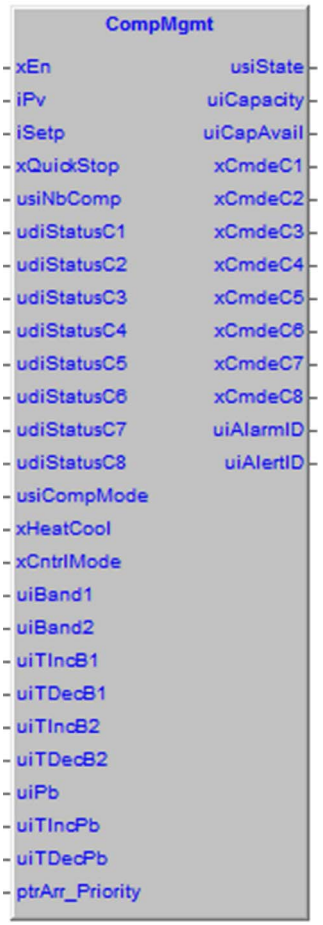
What Is in This Section?

This section contains the following topics:

Input Pin Description

Pin Diagram

The following picture presents the pin diagram of `CompMgmt`:



Input Pin Description

Input	Data Type	Range	Scaling/Unit	Description
xEn	BOOL	TRUE or FALSE	N/A	TRUE: Enables the function block. FALSE: Disables the function block.
iPv	INT	-32768... +32767	User unit, precision	Process value: water temperature or pressure **NOTE:** iPv and iSetp must have the same unit and same precision.
iSetp	INT	-32768... +32767	User unit, precision of iPv	Setpoint value for the process value **NOTE:** iPv and iSetp must have the same unit and same precision.
xQuickStop	BOOL	TRUE or FALSE	N/A	Quick stop of all the compressors
usiNbComp	USINT	0...8	N/A	Number of compressors
udiStatusC1	UDINT	0...4294967295	N/A	Status of compressor 1
udiStatusC2	UDINT	0...4294967295	N/A	Status of compressor 2
udiStatusC3	UDINT	0...4294967295	N/A	Status of compressor 3
udiStatusC4	UDINT	0...4294967295	N/A	Status of compressor 4
udiStatusC5	UDINT	0...4294967295	N/A	Status of compressor 5
udiStatusC6	UDINT	0...4294967295	N/A	Status of compressor 6
udiStatusC7	UDINT	0...4294967295	N/A	Status of compressor 7
udiStatusC8	UDINT	0...4294967295	N/A	Status of compressor 8
usiCompMode	USINT	0...2	N/A	Compressor control sequence mode: **0** FIFO **1** Runtime **2** LIFO + Runtime
xHeatCool	BOOL	TRUE or FALSE	N/A	Cooling and heating mode: TRUE: Heating mode FALSE: Cooling mode
xCntrlMode	BOOL	TRUE or FALSE	N/A	Regulation mode: TRUE: Proportional mode FALSE: Neutral mode
uiBand1	UINT	0...65535	Unit, precision of iPv	Band low for neutral zone Default: 20
uiBand2	UINT	0...65535	Unit, precision of iPv	Band high for neutral zone Default: 40
uiTIncB1	UINT	0...65535	1 s	Delay to increment the number of requested compressors for band 1 only in neutral zone regulation Default: 60

Input	Data Type	Range	Scaling/Unit	Description
uiTDecB1	UINT	0...65535	1 s	Delay to decrement the number of requested compressors for band 1 only in neutral zone regulation Default: 30
uiTIncB2	UINT	0...65535	1 s	Delay to increment the number of requested compressors for band 2 only in neutral zone regulation Default: 30
uiTDecB2	UINT	0...65535	1 s	Delay to decrement the number of requested compressors for band 2 only in neutral zone regulation Default: 15
uiPb	UINT	0...32767	Unit, precision of iPv	Proportional band for proportional band control Default: 4.0
uiTIncPb	UINT	0...65535	1 s	Delay to increment the number of requested compressors only in proportional band regulation Default: 30
uiTDecPb	UINT	0...65535	1 s	Delay to decrement the number of requested compressors only in proportional band regulation Default: 30
ptrArr_Priority	@USINT	N/A	N/A	Priority of the compressors in sequencing mode LIFO. This is an optional input. **NOTE:** If several compressors have the same priority, runtime mode is applied between these compressors.

⚠ DANGER

REFRIGERANT POISONING OR FREEZER BURNS

- Stop the compressor operation in case of a high pressure alarm.
- Interlock the high pressure alarm switch with the compressors using contactors in the electrical cabinet.

Failure to follow these instructions will result in death or serious injury.

xEn

If the input xEn is FALSE:

- The compressor commands xCmdeC1...xCmdeC8 are set to FALSE.
- uiAlarmID and uiAlertID are reset to 0.

Compressor Status

The inputs udiStatusC1...udiStatusC8 must be connected to the output udiCompMgmt-Status of the function block CompCntrl_OnOff.

Compressor status inputs are used to define the availability of the compressors and the actual number of running compressors.

The status of the compressor udiStatusC1...udiStatusC8 represents a value between 0 and 4294967295, whereby each bit provides some informations about the compressor:

Bit	Description
0...23	Operating hours (maximum 16.777.216 hours)
24	Reserved
25	Reserved
26...27	Compressor mode: **1** automatic **2** manual **3** maintenance
28	On timer is active
29	Off timer is active
30	Cycle timer is active
31	Compressor in alarm state

Off timer, cycle timer and alarm state are used to define the availability of the compressor.

On timer is used to define, if the compressor can be stopped.

Quick Stop

If the input xQuickStop is set to TRUE, all the compressors are stopped. The outputs xCmdeC1...xCmdeC8 are set to FALSE.

Output Pin Description

Output Pin Description

Output Pin Description

Output	Data Type	Range	Scaling/Unit	Description
usiState	USINT	0...99	N/A	State: **1** Idle **20** Run **90** Alert **99** Alarm
uiCapacity	UINT	0...1000	0.1 %	Actual running capacity
uiCapAvail	UINT	0...1000	0.1 %	Actual total available capacity
xCmdeC1	BOOL	TRUE or FALSE	N/A	Compressor 1 command
xCmdeC2	BOOL	TRUE or FALSE	N/A	Compressor 2 command
xCmdeC3	BOOL	TRUE or FALSE	N/A	Compressor 3 command
xCmdeC4	BOOL	TRUE or FALSE	N/A	Compressor 4 command
xCmdeC5	BOOL	TRUE or FALSE	N/A	Compressor 5 command
xCmdeC6	BOOL	TRUE or FALSE	N/A	Compressor 6 command
xCmdeC7	BOOL	TRUE or FALSE	N/A	Compressor 7 command
xCmdeC8	BOOL	TRUE or FALSE	N/A	Compressor 8 command
uiAlarmID	UINT	0...65535	N/A	Alarm identification
uiAlertID	UINT	0...65535	N/A	Alert identification

Array Variables

Variable	Data Type	Array	Range	Scaling / Unit	Description
Arr_usiPriority	USINT	0...7	0...255	N/A	Priority of the compressors in LIFO mode: Arr_usiPriority[0] = priority comp 1 Arr_usiPriority[1] = priority comp 2 ... Arr_usiPriority[7] = priority comp 8

Alert ID Description

The `uiAlertID` output represents a value between 0 and 2047, whereby each bit represents an alert. The bits and their description are described in the following table:

Alert Bit	Alert Cause	Effect
0	A controlled parameter has been changed, which requires a machine restart. The new configuration parameter is effective only after restart of the function block. List of the latched parameters: • `usiNbComp` • `usiCompMode` • `xHeatCool`	Present changes are not active. Function block uses the previously set values.
1	The band low `uiBand1` is greater than the high band `uiBand2`.	The low band `uiBand1` is disabled.
2	The parameter `uiPb` is not set within the specified range.	The value is limited
3	Compressor 1 is in alarm state	Compressor 1 is switched off and another compressor is started.
4	Compressor 2 is in alarm state	Compressor 2 is switched off and another compressor is started.
5	Compressor 3 is in alarm state	Compressor 3 is switched off and another compressor is started.
6	Compressor 4 is in alarm state	Compressor 4 is switched off and another compressor is started.
7	Compressor 5 is in alarm state	Compressor 5 is switched off and another compressor is started.
8	Compressor 6 is in alarm state	Compressor 6 is switched off and another compressor is started.
9	Compressor 7 is in alarm state	Compressor 7 is switched off and another compressor is started.
10	Compressor 8 is in alarm state	Compressor 8 is switched off and another compressor is started.
11...15	Not used	N/A

Alarm ID Description

The `uiAlarmID` output represents a value between 0 and 3, whereby each bit represents a detected alarm. The bits and their description are described in the following table:

Alarm Bit	Alarm Cause	Effect
0	The parameter `usiNbComp` is not set within the specified range.	Function block is disabled.
1	All the compressors are in alarm state.	The compressors are not operating.
2...15	Not used	N/A

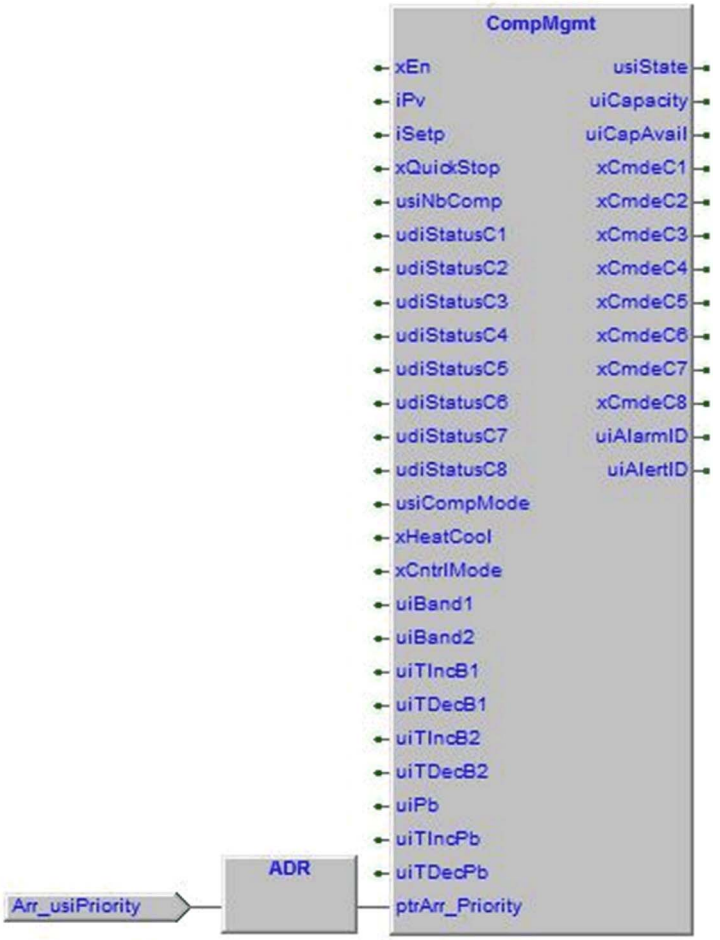
Section 9.5
Troubleshooting

Troubleshooting

Troubleshooting

Alarm / Alert	Problem	Solution
`uiAlarmID.0 TRUE`	Invalid parameter value.	1. Check the parameter ranges. 2. Set the values within the defined ranges.
`uiAlarmID.1 TRUE`	The compressors are not operating.	Verify the compressor alarm inputs.
`uiAlertID.0 TRUE`	Controlled parameters are changed.	Restart the function block.
`uiAlertID.1 TRUE`	Invalid parameter value.	Check the parameter ranges.
`uiAlertID.2 TRUE`	Invalid parameter value.	Check the parameter ranges.
`uiAlertID.3... uiAlertID.10 TRUE`	A compressor is not operating.	Verify the compressor alarm inputs.

Connectivity Diagram



NOTE:

- You have to define the local or global variable of the data type specified in the table *Array Variables* with an array of 8 elements.
- You have to use an `ADR` block to connect the input pin `ptrArr_Priority` to the array variable.

NOTE: During the power cycle the array variable would be reset with the default value.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Be sure that all variables are initialized to an appropriate value before their first use as array indices or pointers.
- Write programming instructions to test the validity of operands intended to be used as array indices and memory pointers.
- Do not attempt to access an array element outside the defined bounds of the array.

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

Chapter 10
Compressor Management Variable Speed: CompMgmtVS

What Is in This Chapter?

This chapter contains the following sections:

Section 10.1
Functional and Machine Overview

What Is in This Section?

This section contains the following topics:

Functional Overview

Functional Description

The function block CompMgmtVS (Compressor Management Variable Speed) is used to control several compressor types, and supports only one refrigerant circuit.

The CompMgmtVS function block calculates the number of compressors and the speed required to control the water temperature or the refrigerant pressure.

In a refrigeration machine, compressors need to be managed in a way to help prolong their proper operation and balance machine lifetime.

The CompMgmtVS function block controls up to four compressors, which can be a mix of fixed and variable speed compressors, and aims to manage compressors. For this purpose, CompMgmtVS provides features for compressor breakdown management and to optimize operation.

The CompMgmtVS must be used together with the function blocks CompCntrl_OnOff and CompCntrl_VS which controls the operation of a single compressor.

Why Use the CompMgmtVS Function Block?

The CompMgmtVS function block is used for the following purposes:

Purpose	Description
Water temperature control or refrigerant pressure control	● Maintain a constant water temperature or a constant refrigerant pressure ● Calculates the required number of compressors. ● Can be used in cooling mode or heating mode: ❍ Heating: if rKp of the Application Function Block PIDAdvanced is positive. ❍ Cooling: if rKp of the Application Function Block PIDAdvanced is negative.
Runtime optimization	● Increase the control accuracy. ● Balance operating hours. ● Avoid frequent On/Off switching of compressors.
Status management	● Switch off a compressor in case of a detected error and switch on another.

Features of the CompMgmtVS Function Block

The CompMgmtVS function block provides the following features:

- Supports 1 to 4 compressors, variable speed, and/or fixed speed.
- Switches on and off the number of required compressors and adjusts the speed to control the water temperature, or the refrigerant low pressure.
- Balances compressor operating hours by using one of the 3 methods:

  - FIFO
  - Runtime
  - LIFO + Runtime
- Helps to prevent frequent switching of a compressor by increasing/decreasing delay management or hysteresis.
- Compressor changeover management in case of a detected compressor error.
- Can be used in cooling mode or heating mode.

Error Avoidance Features

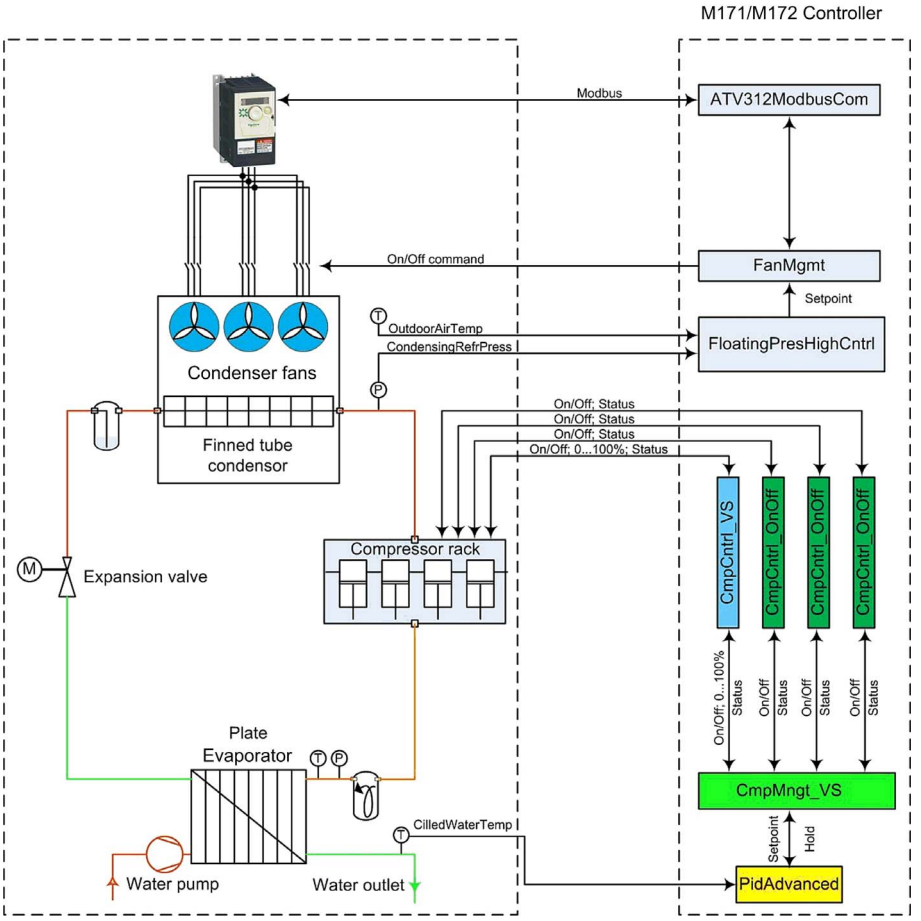
The `CompMgmtVS` function block provides the following error avoidance features to help you avoid the potentials of certain sources of machine malfunction:

Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.
Alarm/alert notification	If an invalid value is entered, an alarm or an alert is generated: ● Alarm: the compressors are switched off and the function block terminates in error. ● Alert: the function block keeps on operating, however with the possibility of reduced performance.
Controlled parameter	Parameters like `usiCompMode`, `usiNbOnOff`, `usiNbVs` and `xCntrlMode` are controlled. The configuration of these parameters can be changed, however the changes are effective only after the restart of the function block.

Machine Overview

Machine View

The following picture presents the interaction between the function block and the machine:



The `CompMgmtVS` function block controls the water temperature or the refrigerant pressure and calculates the required number of compresssors.

In this example, the chilled water temperature or the low pressure are controlled (cooling mode).

This function block controls up to 4 compressors and manages different compressor types.

Section 10.2
Architecture

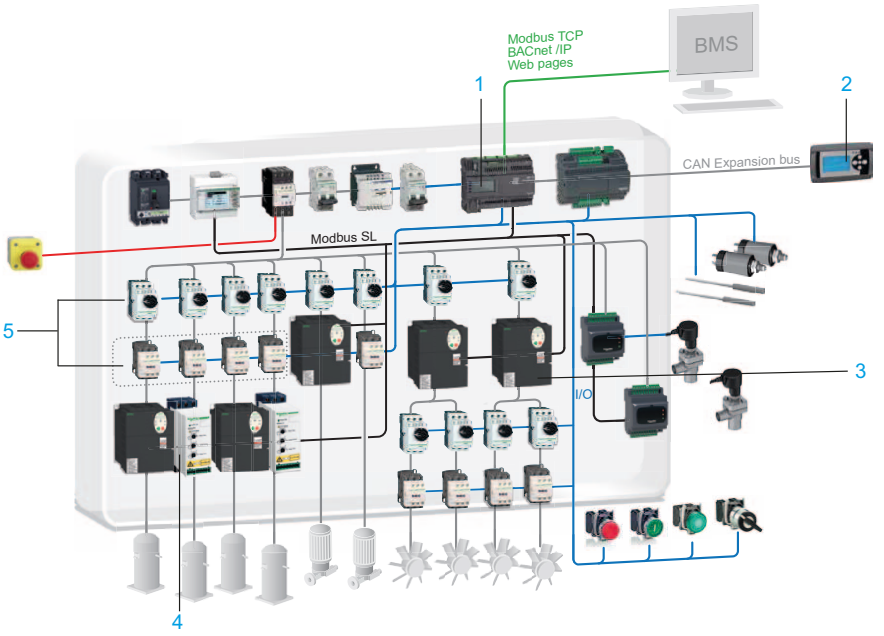
What Is in This Section?

This section contains the following topics:

Hardware Architecture

Hardware Architecture Overview

The following figure presents the hardware architecture of the Air Cooled Chiller:

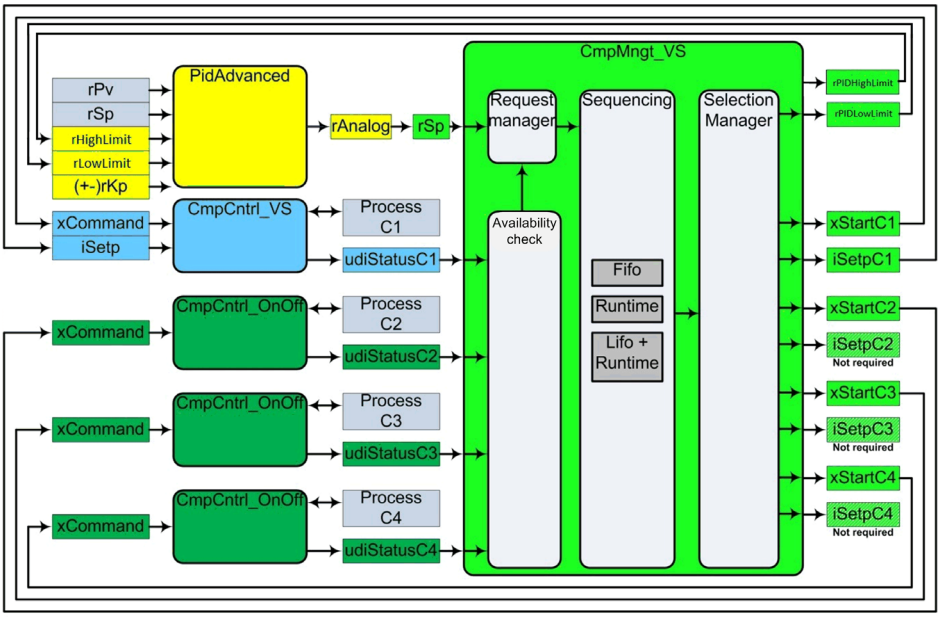


1 M171/M172 Controller
2 Graphic display
3 Variable speed drive ATV••/••• Modbus
4 Soft starters
5 Motor Control

Software Architecture

Function Block Diagram

The following function block diagram provides an overview of the software architecture:



rPv Process variable
rSp Setpoint
(+-) rKp +/- to change control direction

The function block diagram presents on the left-hand side the inputs, the process variables `rPv` and `rSp`. On the right-hand side the diagram presents the outputs, the process commands (for example `xStartC1`), as well as the function block `CompMgmtVS`.

The following table gives you an overview of the functions of `CompMgmtVS`:

Function	Description
Request manager	Calculates the number of requested variable speed compressors and fixed speed compressors, depending on their availability and the strategy used to control them.
Availability Check	• Checks the availability for the configured number of compressors. • Evaluates compressor alarms. • Evaluates compressor minimum On/Off time. • Evaluates compressor operation hours for switching on.

Function	Description
Sequencing	Determines, which compressor is switched on/off next, based on three methods: ● FIFO ● Runtime ● LIFO + Runtime
Selection manager	Selects the compressor to be switched on next, based on compressor sequencing and availability.

Section 10.3
Function Block Description

CompMgmtVS Function Block

Function Block Description

The CompMgmtVS function block controls up to four compressor types. Operating hours are balanced using various methods.

The CompMgmtVS breakdown management function automatically commands a new compressor when an operating compressor has a detected error.

The CompMgmtVS function block creates a linear law between the production with the needs, even with different sizes of fixed or variable speed compressors.

The following methods for compressor management are provided:
- Compressor sequence control
- Compressor breakdown management
- Compressor on/off timing control or hysteresis
- Compressor selection timing control
- Compressor operation hours

Compressor Sequence Control

The purpose of the compressor sequence control is to balance the number of operating hours and starts/stops between the compressors.

Compressor sequence control helps to ensure equal usage of the compressors and optimize power consumption.

Compressors are controlled based on the following sequences:

Sequence as per **usiCompMode**	Description
FIFO = First In First Out	• The compressor with the least operating hours is switched on first. • The first compressor which is switched on is also the first to be switched off. • Advantage: operation time is limited.
Runtime	• The compressor with the least operating hours is the first compressor to be switched on. • The compressor with the most operating hours is the first compressor to be switched off. • Advantage: balanced operation hours.

Sequence as per `usiCompMode`	Description
LIFO = Last In First Out + Runtime	• The parameter pointer `usiLifoSeq` determines this sequence. • The first compressor to be switched on is the first one in the sequence. • The first compressor to be switched off is the last one that has been switched on. • Advantage: priority of compressor usage can be set. **NOTE:** If several compressors have the same priority, the compressor with the least operating hours is the first compressor to be switched on.

NOTE:

- If a variable speed drive is available, the variable speed drive is switched on before the OnOff-compressors switch on, and the variable speed drive is stopped last.
- If a compressor is not available (Off timer or cycle timer are active, or the compressor is in alarm state), another compressor is started based on the sequence defined by `usiCompMode`.
- If a compressor cannot be stopped (On-timer is active), another compressor is stopped based on the sequence defined by `usiCompMode`.

Compressor Breakdown Management

The compressor which is detected as non-operating, is switched off.

The next available compressor in the start sequence is switched on.

The non-operating compressor cannot be started until the compressor is returned to an operational state.

Cooling and Heating Mode

The heating-cooling mode is changed by the input `rKp` of the `PIDAdvanced` Application Function Block.

- If `rKp` is set with a positive value (for example +2.0), heating mode is set.
- If `rKp` is set with a negative value (for example -2.0), cooling mode is set.

In cooling mode, the chilled water temperature or the refrigerant low pressure is controlled.

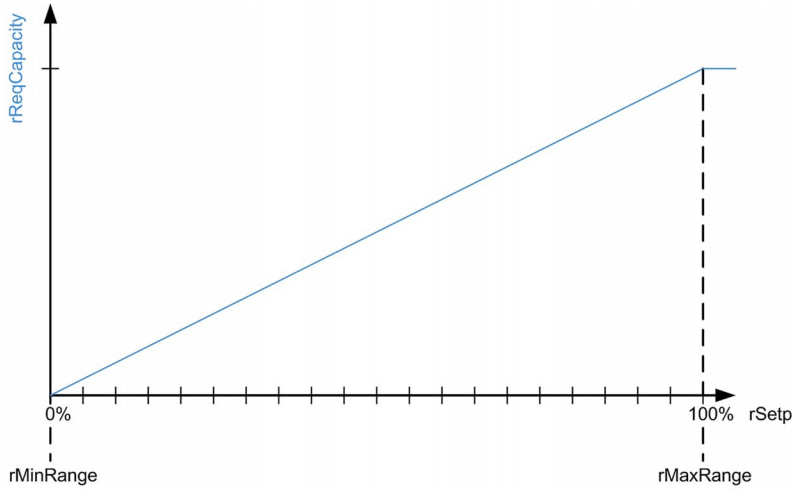
In heating mode, the warmed water temperature or the refrigerant high pressure is controlled.

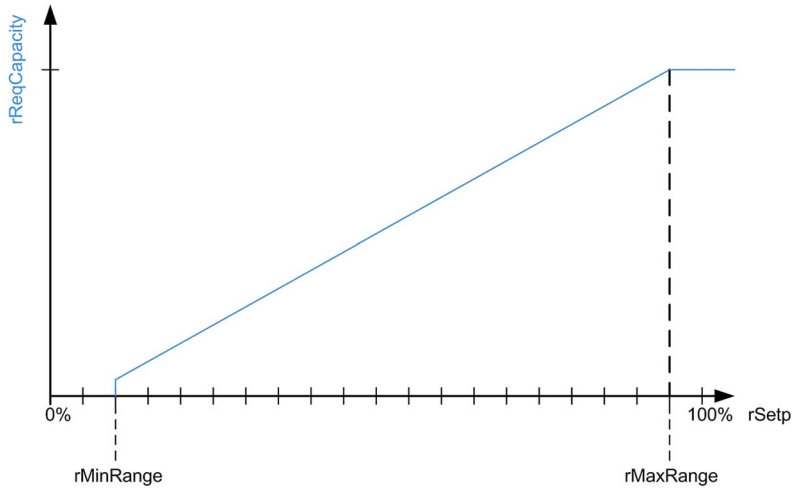
Regulation with `rMinRange` and `rMaxRange`

To increase the stability of the system when the input `rSp` is close to 0% or 100%, the regulation range for the compressor is adapted to the range `rMinRange` to `rMaxRange`.

`xCtrlMode` is set to FALSE (`uiDelayOn` and `uiDelayOff` is active).

The graphics show an example for `rMinRange`, `rMaxRange` (1VS):





NOTE: The parameters `rMinRange` and `rMaxRange` must be set according to the thermal capacity of the machine. The default value for `rMinRange` is 5.0% and the default value for `rMaxRange` is 98.0%.

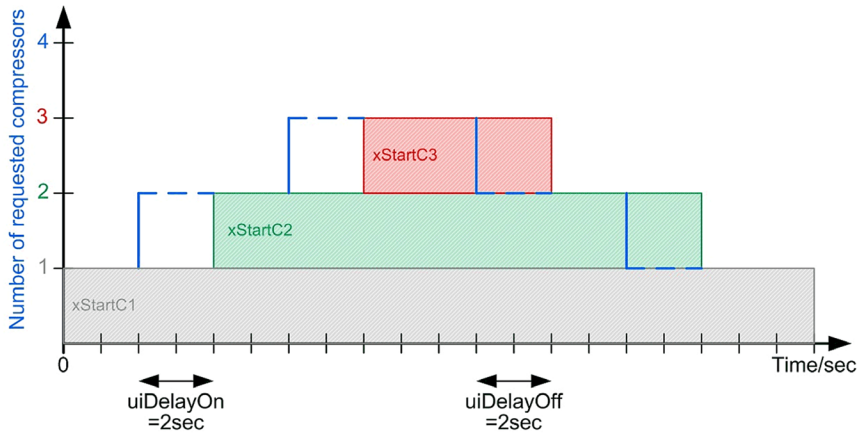
Regulation With Delay

The input `xCntrlMode` is set to FALSE.

To increase the stability of the system, the input `uiDelayOn` can be set to delay the increment, or `uiDelayOff` to decrement the number of requested compressors.

NOTE: When the first variable speed compressor has to be switched on, the delay is not active. When the first On/Off compressor has to be switched on, the delay is active.

Example `uiDelay` overview:



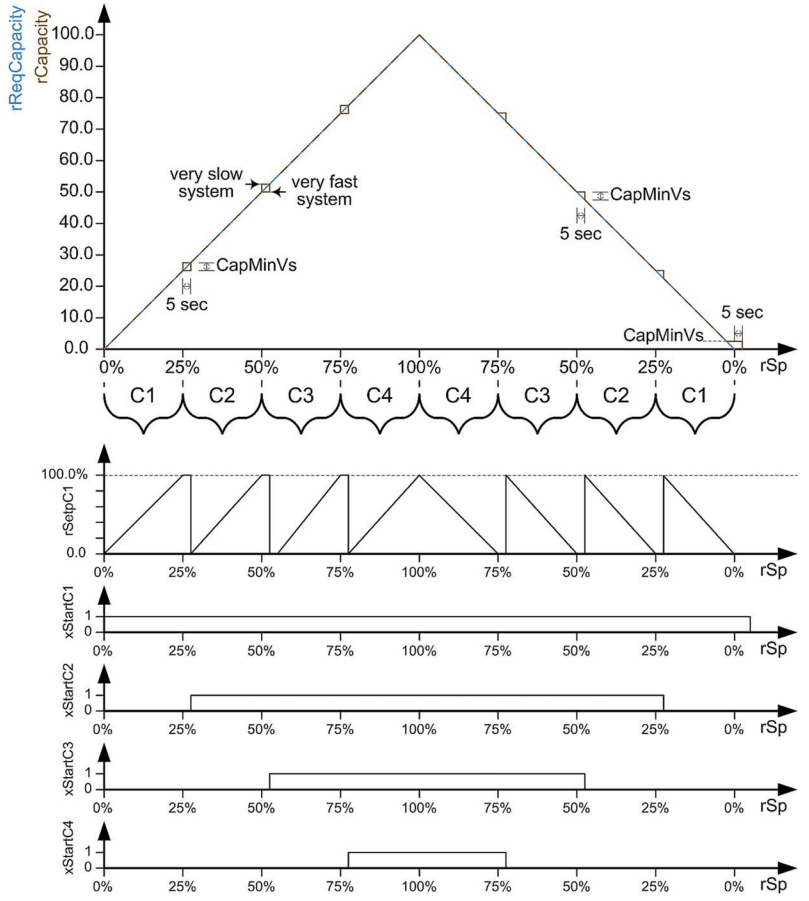
The following example applies for one variable speed compressor and three On/Off compressors with control mode (FIFO, Runtime, LIFO+ Runtime).

The following values were set to show the diagram:

Parameter	Value/Unit
`rMaxRange`	100.0%
`rMinRange`	0.01%(∿ 0.0%)
`usiNbVs`	1
`usiNbOnOff`	3
`rMaxFreq`	50.0 Hz
`rMinFreq`	5.0 Hz
`CapMaxVs`	25.0 (for 1 variable speed compressor, internally calculated)
`CapMinVs`	2.5 (for 1 variable speed compressor, internally calculated)
`rReqCapacity`	0.0...100.0
`rCapacity`	2.5...100.0

Parameter	Value/Unit
uiDelayOn	5 s
uiDelayOff	5 s
rNomFreq	50 Hz
xCtrlMode	FALSE
uiCapVs	25
uiCapOnOff	25

The graphic shows an example for rMinRange, rMaxRange (1Vs+ 3 OnOff)



Regulation with Minimum Frequency, Maximum Frequency, and Nominal Frequency

If...	Then...
If a variable speed compressor is available and ready,	A variable speed compressor is started first and a variable speed compressor is stopped last.
If more than one variable speed compressor is set by usiNbVs,	The variable speed compressors run until the maximum frequency (`rMaxFreq`) is reached before a new compressor is started. This is also valid if several variable speed drives are used.
If `xStartC1` is set to TRUE and `rSetpC1` is equal to 100.0%,	The variable speed compressor C1 is running at the maximum frequency `rMaxFreq`.
If more than one variable speed compressor is required,	The analog outputs `rSetpC*` has the same calculated values (analog for the value `iSetpC*`).

The following examples A and B are with a control mode (FIFO, LIFO, Runtime). In both examples, `uiDelayOn` and `uiDelayOff` are set to 0.

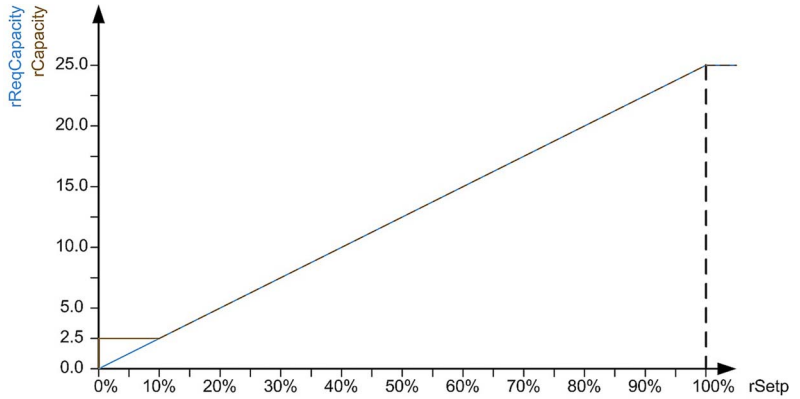
Example A

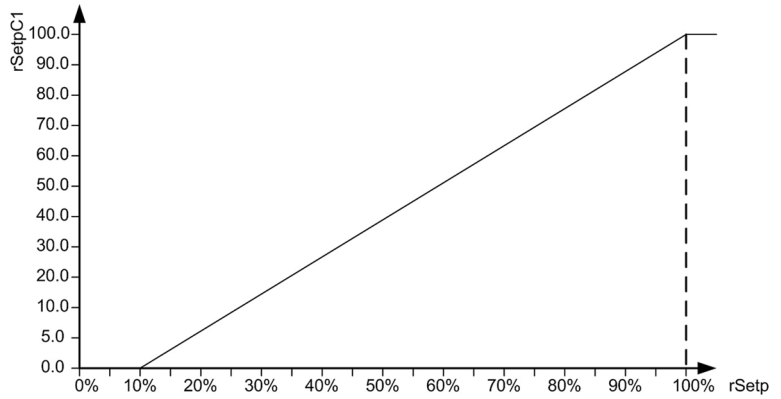
Parameter	Value / Unit
`usiNbVs`	1
`usiNbOnOff`	0
`rMaxFreq`	50 Hz
`rMinFreq`	5 Hz
`rMinRange`	0.01% (∿ 0.0%)
`rMaxRange`	100%
`rNomFreq`	50 Hz
`xCtrlMode`	FALSE
`uiCapVs`	25
`uiCapOnOff`	25

Result: Maximum and minimum capacity (`CapMaxVs`, `CapMinVs`) of a variable speed drive, required capacity `rReqCapacity` (output).

Parameter	Value / Unit
`CapMaxVs`	25.0 (for 1 variable speed compressor, internally calculated)
`CapMinVs`	2.5 (for 1 variable speed compressor, internally calculated)
`rReqCapacity`	0.0…25.0
`rCapacity`	2.5…25.0

Example `rMinFreq`, `rMaxFreq`, `rNomFreq` (1 Vs):





Example B

Parameter	Scale / Unit
usiNbVs	4
usiNbOnOff	0
rMaxFreq	50 Hz
rMinFreq	30 Hz
rMinRange	0.01% (∿ 0.0%)
rMaxRange	100%
rNomFreq	50 Hz
xCtrlMode	FALSE
uiCapVs	25
uiCapOnOff	25

Result: Maximum and minimum capacity (CapMaxVs, CapMinVs) of a variable speed drive, required capacity rReqCapacity (output).

Parameter	Scale / Unit
CapMaxVs	25.0 (for 1 variable speed compressor, internally calculated)
CapMinVs	15 (for 1 variable speed compressor, internally calculated)
rReqCapacity	0.0…100.0
rCapacity	15…100.0

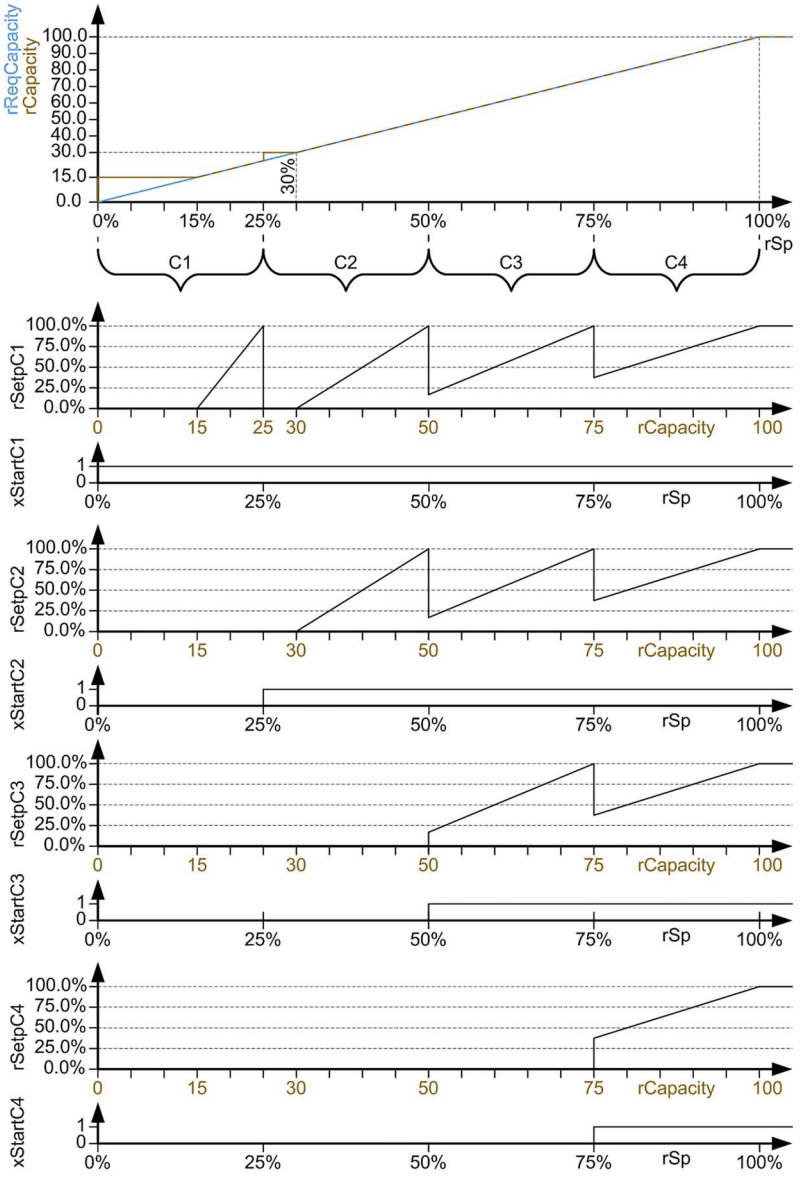
The outputs rSetpC1 and rSetpC2 are not started at rCapacity = 0 and rCapacity = 25.

rMinFreq is set to 30.0 Hz and CapMinVs = 15.

If...	Then...
If rSp is greater than rMinRange,	The output xStartC1 is set to TRUE. **Result**: A value of 15.0 is set at the output rCapacity, but the calculated required capacity rReqCapacity is less than the set rCapacity. In this case, the calculation of the output rSetpC1 starts by rReqCapacity > rCapacity.

If...	Then...
If `rReqCapacity` > `CapMaxVs`,	The system requires two variable speed drives. **Result:** ● `rReqCapacity` is for example 25.1 (> `CapMaxVs`) and the outputs `xStartC1` and `xStartC2` are set to TRUE. ● `rCapacity` is 30; also `rCapacity` > `rReqCapacity` and no further capacity is needed. ● Both outputs `rSetpC1` and `rSetpC2` are 0. In this case, the calculation of the outputs `rSetpC1` and `rSetpC2` start by `rReqCapacity` > `rCapacity` (here 30).
If `rReqCapacity` > 2 x `CapMaxVs`,	The system requires three variable speed drives. **Result:** ● `rReqCapacity` is for example 50.1 (> 2 x `CapMaxVs`) and the outputs `xStartC1` and `xStartC2` are set to TRUE. ● `rCapacity` is 45, also `rCapacity` < `rReqCapacity`. ● The required outputs `rSetpC1` and `rSetpC2` and `rSetpC3` are set to 17.0%.

Example `rMinFreq`, `rMaxFreq`, `rNomFreq` (4 Vs):



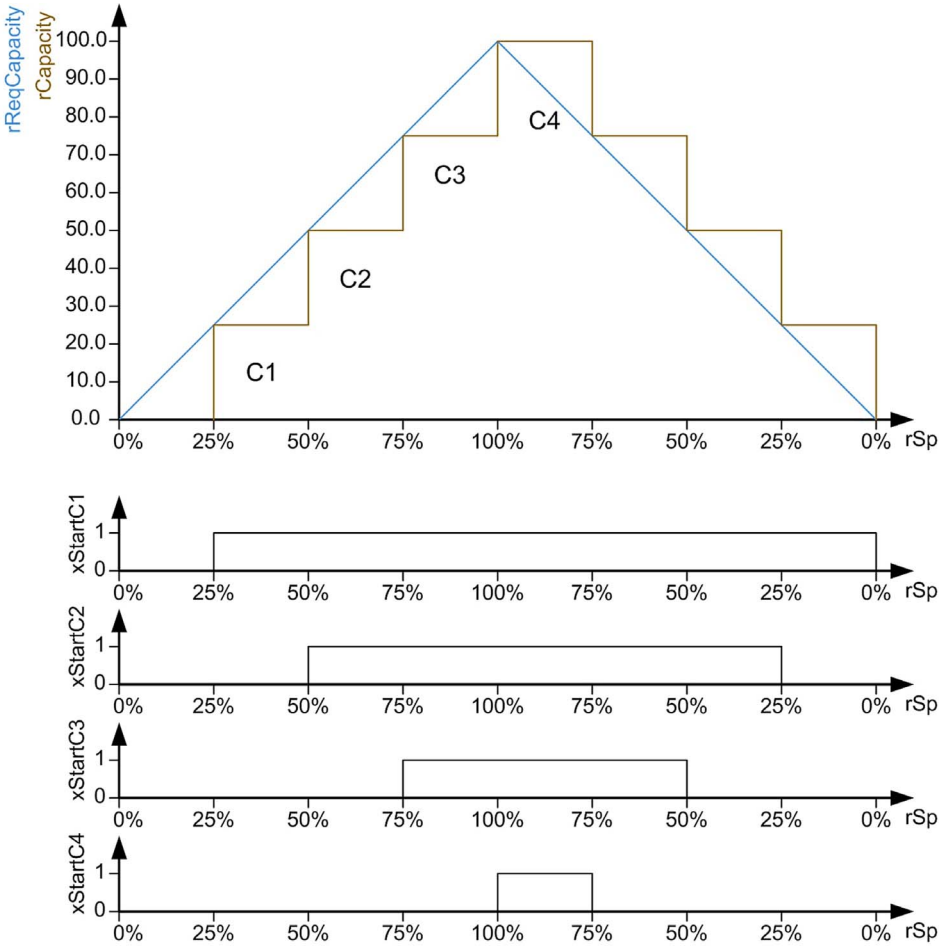
Regulation with 4 On/Off Compressors

This regulation is possible but not as precise as variable speed compressors.

The following example is with control mode (FIFO, LIFO, Runtime).

The following values were set to show an exemplary diagram:

Parameter	Scale / Unit
rMaxRange	100
rMinRange	0.01% (∿ 0.0%)
uiDelayOn	0
uiDelayOff	0
xCtrlMode	FALSE



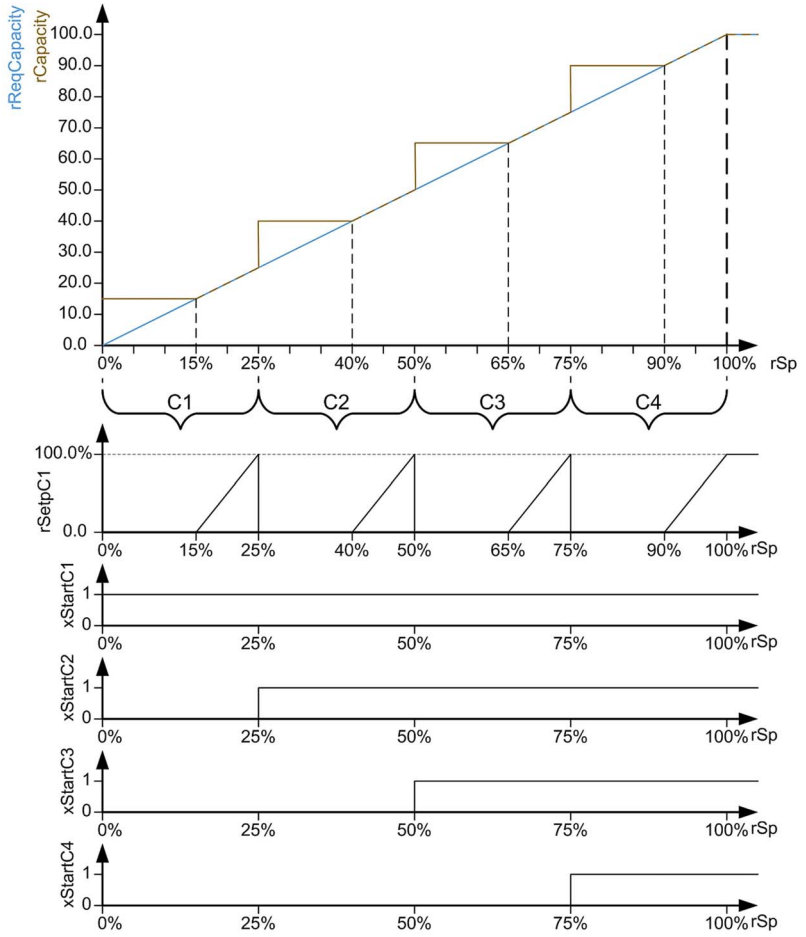
Regulation with One Variable Speed Compressor and 3 On/Off Compressors

The following example is with a control mode (FIFO, LIFO, Runtime).

The following values were set to show a hypothetical diagram:

Parameter	Value / Unit
rMaxRange	100%
rMinRange	0.01% (∿ 0.0%)
usiNbVs	1
usiNbOnOff	3

Parameter	Value / Unit
rMaxFreq	50 Hz
rMinFreq	30 Hz
CapMaxVs	25.0 (for 1 variable speed compressor, internally calculated)
CapMinVs	15.0 (for 1 variable speed compressor, internally calculated)
rReqCapacity	0.0…100.0
rCapacity	15.0…100.0
uiDelayOn	0
uiDelayOff	0
xCtrlMode	FALSE
rNomFreq	50 Hz
uiCapVs	25
uiCapOnOff	25


rReqCapacity
rCapacity
100.0
90.0
80.0
70.0
60.0
50.0
40.0
30.0
20.0
10.0
0.0
0%
15%
25%
40%
50%
65%
75%
90%
100%
rSp
C1
C2
C3
C4
100.0%
rSetpC1
0.0
xStartC1
xStartC2
xStartC3
xStartC4
1
0
25%
50%
75%

Regulation With Hysteresis (rHys, rMinRange, rMaxRange)

The regulation with hysteresis is more adaptive than the regulation with delay because the time before starting or stopping a compressor depends on the variation speed of the value PID rSp.

If...	Then...
If the setpoint of the PIDAdvanced rAnalog varies fast, for example when the machine is started or when the load reduces significantly,	The time to start or stop a compressor is reduced which improves the response time of the system.
If the setpoint of the PIDAdvanced rAnalog varies slowly, for example when the load is constant,	The time to start or stop a compressor is longer which increases the stability of the system.

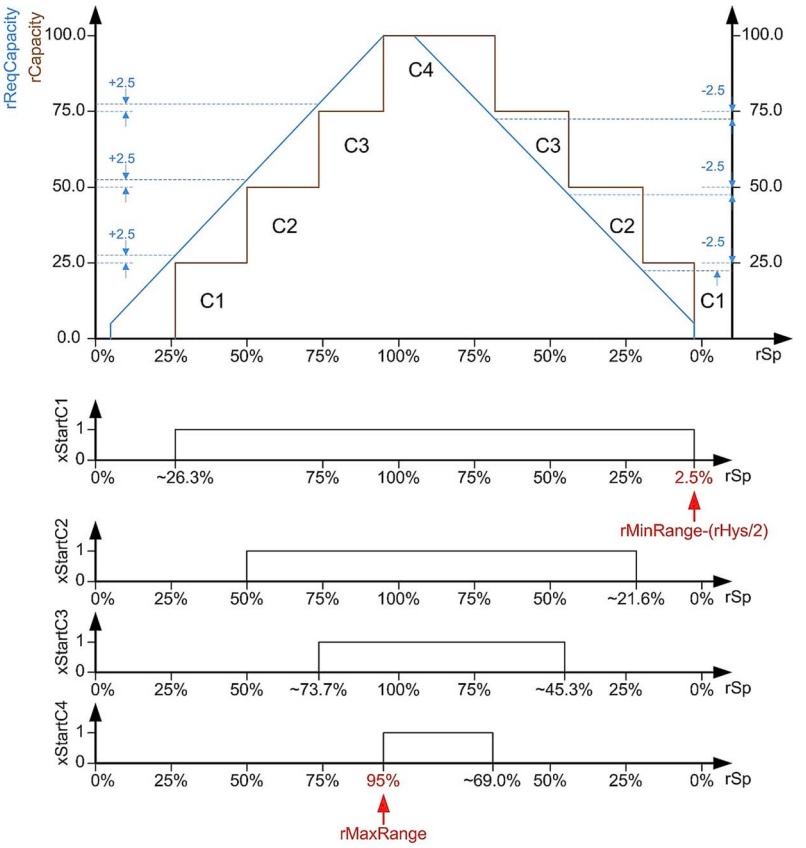
The value of rHys has more priority than rMinRange and rMaxRange.

If...	Then...
$\left(\frac{rHys}{2.0}\right) \leq rMinRange$	The last compressor stops by the value of $rMinRange - \left(\frac{rHys}{2.0}\right)$
$\left(\frac{rHys}{2.0}\right) > rMinRange$	The last compressor will stop by the setpoint=0.0 (rSp).

Example A:

Parameter	Value / Unit
rMaxRange	95%
rMinRange	5%
usiNbVs	0
usiNbOnOff	4
xCtrlMode	TRUE
rHys	5.0%

Example regulation with hysteresis (4 On/Off):



Example B:

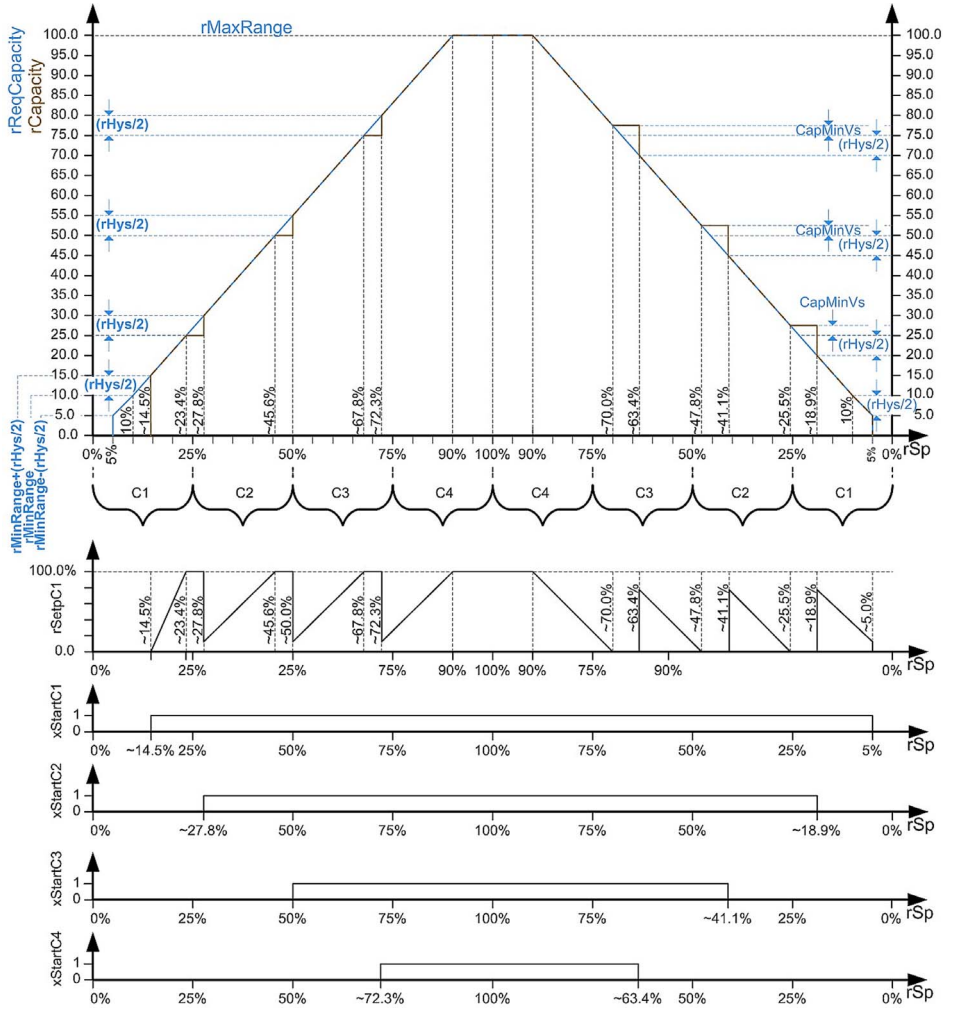
Parameter	Value / Unit
rMaxRange	90%
rMinRange	10%
usiNbVs	1
usiNbOnOff	3
xCtrlMode	TRUE
rHys	10.0%
rMaxFreq	50 Hz
rMinFreq	5 Hz

Parameter	Value / Unit
rNomFreq	50 Hz
uiCapVs	25 (user unit)
uiCapOnOff	25 (user unit)

Result: Maximum and minimum capacity (CapMaxVs, CapMinVs) of a variable speed drive, required capacity rReqCapacity (output).

Parameter	Scale / Unit
CapMaxVs	25.0 (for 1 variable speed compressor, internally calculated)
CapMinVs	2.5 (for 1 variable speed compressor, internally calculated)
rReqCapacity	5.0…100.0 $\left(5.0 = rMinRange - \left(\frac{rHys}{2}\right)\right)$
rCapacity	15…100.0 $\left(5.0 = rMinRange - \left(\frac{rHys}{2}\right)\right)$

Example regulation with hysteresis (1 VS, 3 On/Off):



Section 10.4
Pin Description

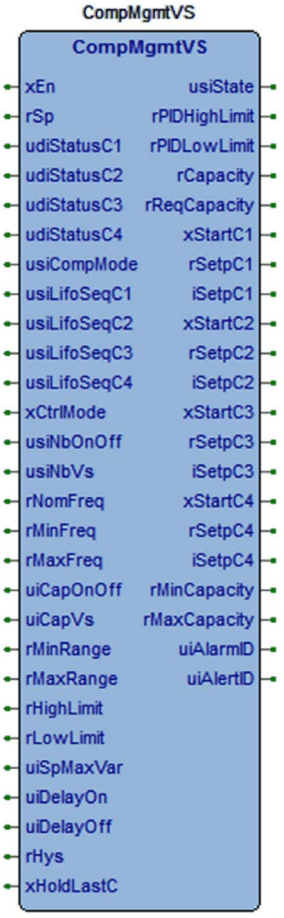
What Is in This Section?

This section contains the following topics:

Input Pin Description

Pin Diagram

The following picture presents the pin diagram of `CompMgmtVS`:



Input Pin Description

Input	Data Type	Range	Scaling/Unit	Description
xEn	BOOL	TRUE or FALSE	N/A	Enables the function block.
rSp	REAL	0.0...100.0	%	Setpoint
udiStatusC1	UDINT	0...4294967295	N/A	Status of compressor 1
udiStatusC2	UDINT	0...4294967295	N/A	Status of compressor 2
udiStatusC3	UDINT	0...4294967295	N/A	Status of compressor 3
udiStatusC4	UDINT	0...4294967295	N/A	Status of compressor 4
usiCompMode	USINT	0...2	N/A	Compressor control sequence mode **NOTE:** ● 0: FIFO ● 1: Runtime ● 2: LIFO
usiLifoSeqC1	USINT	1...4	N/A	Compressor 1 start sequence (priority in sequencing mode LIFO). **NOTE:** Variable speed drives have the highest priority.
usiLifoSeqC2	USINT	1...4	N/A	Compressor 2 start sequence (priority in sequencing mode LIFO). **NOTE:** Variable speed drives have the highest priority.
usiLifoSeqC3	USINT	1...4	N/A	Compressor 3 start sequence (priority in sequencing mode LIFO). **NOTE:** Variable speed drives have the highest priority.
usiLifoSeqC4	USINT	1...4	N/A	Compressor 4 start sequence (priority in sequencing mode LIFO). **NOTE:** Variable speed drives have the highest priority.
xCtrlMode	BOOL	TRUE or FALSE	N/A	Control mode (switch on/off of the compressors). FALSE: Delay TRUE: Hysteresis
usiNbOnOff	USINT	0...4	N/A	Number of On/Off speed compressors usiNbOnOff+ usiNbVS ≤4
usiNbVS	USINT	0...4	N/A	Number of variable speed compressors usiNbOnOff + usiNbVS ≤4
rNomFreq	REAL	0...500.0	Hz	Nominal speed of the compressors Default value: 50.0 Hz

Input	Data Type	Range	Scaling/Unit	Description
rMinFreq	REAL	0.0...500.0	Hz	Minimum frequency of the variable speed compressors rMinFreq < rMaxFreq
rMaxFreq	REAL	0.0...500.0	Hz	Maximum frequency of the variable speed compressors rMinFreq < rMaxFreq
uiCapOnOff	UINT	0...1000	N/A	Capacity of the On/Off compressors
uiCapVS	UINT	0...1000	N/A	Capacity of the variable speed compressors at 50.0 Hz
rMinRange	REAL	0...99.9	%	Low limit of the range Default value: 5% rRangeMin < rRangeMax
rMaxRange	REAL	0...100.0	0.1%	High limit of the range Default value: 98% rRangeMin < rRangeMax
rHighLimit	REAL	0...100.0	%	High limit of PID output
rLowLimit	REAL	0...100.0	%	Low limit of PID output
uiSpMaxVar	UINT	0...1000	s	Maximum setpoint variation Time to increase/decrease the setpoint of 10% 0 leaves rSp unchanged.
uiDelayOn	UINT	0...6000	1 s	Delay to increment the number of requested compressors.
uiDelayOff	UINT	0...6000	1 s	Delay to decrement the number of requested compressors.
rHys	REAL	0...99.9	%	Hysteresis to increment and decrement the number of requested compressors
xHoldLastC	BOOL	TRUE or FALSE	N/A	Hold last compressor On, if shut off is requested. Used to implement pump-down procedure.

xEn

If...	Then...
If the input xEn is FALSE,	The outputs are reset to 0.
If the input xEn gives a rising edge,	The valid inputs are triggered.

udiStatusC1...udiStatusC4

The inputs `udiStatusC1...udiStatusC4` must be connected to the output `udiCompMgmt` - status of the function blocks `CompCntrl_VS` and `CompCntrl_OnOff`.

Compressor status inputs are used to define the availability of the compressors and the actual number of running compressors.

The status of the compressor `udiStatusC1...udiStatusC4` represents a value from 0 to 4.294.967.295, whereby each bit provides some information on the compressor:

Bit	Description
0…23	Operating hours (limited to 16,700,100 hours)
24	Reserved
25	Reserved
26...27	1: automatic 2: manual 3: maintenance
28	TRUE: On timer is active.
29	TRUE: Off timer is active.
30	TRUE: Cycle timer is active.
31	TRUE: Compressor is in alarm state.

If...	Then...
If the operating hours are > 16,700,100 hours,	The AFB `CompMgmtVS` gets an alert and freezes the value of 16,700,100 operating hours internal.
If the compressor is not in automatic (bit 26…27),	The AFB `CompMgmtVS` regards this as an alarm.

The Off timer, cycle timer and alarm state are used to define the availability of the compressor.

The On timer is used to define if the compressor can be stopped.

A long On timer can create situations of not-controlled plant, for example, if the temperature or pressure is low and the compressors cannot be stopped.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Ensure that the low-pressure switch and the low-pressure probe are properly set in the AFB `CompAlarmMgmt`, if used.
- Implement other low pressure safety-related devices if the low pressure probe is not set in the AFB `CompAlarmMgmt`.
- Ensure that measures to avoid freezing and/or low water temperature have been established.

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

uiSpMaxVar

It is possible to limit the slope of `rSP` signal coming from `PidAdvanced` in order to prevent too quick variation and a consequently too fast switching on or off of the compressors (for example during first power on).

The input `uiSpMaxVar` indicates the time necessary to increase or decrease the `rSp` of 10% value.

xHoldLastC

If this input is TRUE, at least one compressor remains On. Therefore, if `rSp` requires the compressors to be switched off, the last compressor is stopped if `xHoldLastC` is FALSE. This input can be used to implement a pump-down procedure.

If the input `xHoldLastC` is On for a long time, the plant cannot be stopped and the temperature or pressure can become too low.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Ensure that the low-pressure switch and the low-pressure probe are properly set in the AFB `CompAlarmMgmt`, if used.
- Implement other low pressure safety-related devices if the low pressure probe is not set in the AFB `CompAlarmMgmt`.
- Ensure that measures to avoid freezing and/or low water temperature have been established.
- Implement a coherent pump down procedure setting the input `xHoldLastC` to FALSE, when the pressure is low.

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

Output Pin Description

Output Pin Description

Output	Data Type	Range	Scaling/Unit	Description
usiState	USINT	0...99	N/A	Current state: 1: Idle 20: Run 21: Holding last compressor on (pump down) 90: Alert 99: Alarm
rPIDHighLimit	REAL	0...100.0	%	Change the current value of the PID HighLimit.
rPIDLowLimit	REAL	0...100.0	%	Change the current value of the PID LowLimit.
uiCapacity	REAL	-3.40E+38... +3.40E+38	N/A	Total capacity of the running compressors.
uiReqCapacity	REAL	-3.40E+38... +3.40E+38	N/A	Required capacity of the system (calculated).
xStartC1	BOOL	TRUE or FALSE	N/A	Start command for compressor 1.
rSetpC1	REAL	0...100.0	%	Setpoint for compressor 1. Not used if connected to a fixed speed compressor.
iSetpC1	INT	0...1000	0.1%	Set point for compressor 1. Not used if connected to a fixed speed compressor.
xStartC2	BOOL	TRUE or FALSE	N/A	Start command for compressor 2.
rSetpC2	REAL	0...100.0	%	Setpoint for compressor 2. Not used, if connected to a fixed speed compressor.
iSetpC2	INT	0...1000	0.1%	Setpoint for compressor 2. Not used if connected to a fixed speed compressor.
xStartC3	BOOL	TRUE or FALSE	N/A	Start command for compressor 3.
rSetpC3	REAL	0...100.0	%	Setpoint for compressor 3. Not used if connected to a fixed speed compressor.
iSetpC3	INT	0...1000	0.1%	Setpoint for compressor 3. Not used if connected to a fixed speed compressor.
xStartC4	BOOL	TRUE or FALSE	N/A	Start command for compressor 4.

Output	Data Type	Range	Scaling/Unit	Description
rSetpC4	REAL	0...100.0	%	Setpoint for compressor 4. Not used if connected to a fixed speed compressor.
iSetpC4	INT	0...1000	0.1%	Setpoint for compressor 4. Not used if connected to a fixed speed compressor.
rMinCapacity	REAL	–	N/A	Minimum capacity of the system.
rMaxCapacity	REAL	–	N/A	Maximum capacity of the system.
uiAlarmID	UINT	0...65535	N/A	Alarm ID
uiAlertID	UINT	0...65535	N/A	Alert ID

rPIDHighLimit

It is possible to connect the output rPIDHighLimit directly with the input rHighLimit of the AFBPidAdvanced.

To help prevent windup of PID integral part and increase the stability of the system, the output rSp of the PidAdvanced should be limited with the maximum reachable value of the system.

The maximum ready capacity value is the sum of capacities of available compressors, where available compressors are compressors that are not in alarm, and with the minimum Off timers or mimimum cycle timers not active.

The total system capacity (rlCapMaxSysKw) is the sum of capacities of all compressors:

$$rPIDHighLimit = \frac{rlCapSysRdyKw}{rlCapMaxSysKw} \times 100$$

NOTE: If calculated rPIDHighLimit is higher than rHighLimit , rPIDHighLimit is set to rHighLimit input.

NOTE: If all the compressors are in alarm state, uiAlarmID is set to a value > 0, and all outputs are set to 0.

rPIDLowLimit

It is possible to connect the output `rPIDLowLimit` directly with the input `rLowLimit` of the `PidAdvanced`.

To help prevent windup of the PID integral part and increase the stability of the system, the output `rAnalog` of the `PidAdvanced` should be low-limited with the current minimum reachable value, considering that some compressors could not be turned off.

The system minimum capacity value (`rlCapSysOffKw`) is the sum of capacities of the compressors that have the minimum On timers active.

The total system capacity (`rlCapMaxSysKw`) is the sum of capacities of all compressors.

$$rPIDLowLimit = \frac{rlCapSysOffKw}{rlCapMaxSysKw} \times 100$$

NOTE: If the calculated `rPIDLowLimit` is smaller than `rLowLimit`, `rPIDLowLimit` is set to the value of the input `rLowLimit`.

Alarm ID Description

The output `uiAlarmID` represents a value from 0 to 15, whereby each bit represents an alarm. The bits and their description are described in the following table:

Alarm Bit	Alarm Cause	Effect
0	The parameter `rSp` is not set within the specified range.	The compressors are not operating.
1	The parameter `usiCompMode` is not set within the specified range.	
2	The parameters `usiLifoSeqC1`, `usiLifoSeqC2`, `usiLifoSeqC3`, `usiLifoSeqC4` are not set within the specified range.	

3	The parameters `usiNbOnOff` and `usiNbVs` are not set within the specified range or
	`(usiNbOnOff + usiNbVs) > 4 or`
	`usiNbOnOff + usiNbVs) = 0.`	

4	`usiNbOnOff > 0 or usiNbVs>0 and` `rNomFreq <= 0.0 or rNomFreq > 500.0.`	

5	`usiNbVs` > 0 and
	`rMinFreq <= 0.0 or rMinFreq > 500.0 or`
	`rMaxFreq <= 0.0 or rMaxFreq > 500.0 or`
	`rMaxFreq < rMinFreq.`	

6	`usiNbOnOff >0 and`
	`uiCapOnOff <= 0 or`
	`uiCapOnOff>1000`	

7	`usiNbVs > 0 and`
	`uiCapVs <= 0 or`
	`uiCapVs > 1000`	

8	`rMinRange > 99.90 or rMinRange <= 0.0 or`
	`rMaxRange > 100.0 or rMaxRange <= 0.0 or`
	`rMinRange > rMaxRange`	

9	The parameters `uiDelayOn` and `uiDelayOff` are not set within the specified range.	
10	The parameter `rHys` is not set within the specified range.	
11	All the compressors are in alarm state.	
12	System Clock Alarm, the value of the controller clock is not valid, for example 0:0:0:0:0:0. The internal calculation requires a valid value.	
13...15	Not used	N/A

Alert ID Description

In the event of an alert, the still available compressors are running and the output `uiAlertID` gives some indications about the alert.

The `uiAlertID` output represents a value from 0 to 15, whereby each bit represents an alert. The bits and their description are described in the following table:

Alert Bit	Alert Cause	Effect
0	Operating hours of compressor 1 > 16,700,100.	At 16,700,100 operating hours, the value is frozen and the AFB cannot calculate with the real operating hours of compressor 1.
1	Operating hours of compressor 2 > 16,700,100.	At 16,700,100 operating hours, the value is frozen and the AFB cannot calculate with the real operating hours of compressor 2.
2	Operating hours of compressor 3 > 16,700,100.	At 16,700,100 operating hours, the value is frozen and the AFB cannot calculate with the real operating hours of compressor 3.
3	Operating hours of compressor 4 > 16,700,100.	At 16,700,100 operating hours, the value is frozen and the AFB cannot calculate with the real operating hours of compressor 4.
4	Compressor 1 is in alarm state or not in auto mode	Compressor 1 is switched off and another available compressor is started.
5	Compressor 2 is in alarm state or not in auto mode	Compressor 2 is switched off and another available compressor is started.
6	Compressor 3 is in alarm state or not in auto mode	Compressor 3 is switched off and another available compressor is started.
7	Compressor 4 is in alarm state or not in auto mode	Compressor 4 is switched off and another available compressor is started.
8	`usiCompMode`: This controlled parameter has been changed, which requires a machine restart. The new configuration parameter is effective only after restart of the function block.	Present changes are not active. The function block uses the previously set values.
9	`xCtrlMode`: These controlled parameters have been changed, which requires a machine restart. The new configuration parameter is effective only after restart of the function block.	
10	`usiNbOnOff` or `usiNbVs`: These controlled parameters have been changed, which requires a machine restart. The new configuration parameter is effective only after restart of the function block.	
11...15	Not used	N/A

Section 10.5
Troubleshooting

Troubleshooting

Troubleshooting

Alarm / Alert	Problem	Solution
`uiAlertID` 0000 0000 0000 0001	The operating hours of compressor 1 > 16,700,100	Until 16,700,100 operating hours, it is possible to reset the operating hours of compressor 1 with the input `xOpHoursReset` of the AFB `CmpCtrl_VS` or `CmpCtrl_OnOff`.
`uiAlertID` 0000 0000 0000 0010	The operating hours of compressor 2 > 16,700,100	Until 16,700,100 operating hours, it is possible to reset the operating hours of compressor 2 with the input `xOpHoursReset` of the AFB `CmpCtrl_VS` or `CmpCtrl_OnOff`.
`uiAlertID` 0000 0000 0000 0100	The operating hours of compressor 3 > 16,700,100	Until 16,700,100 operating hours, it is possible to reset the operating hours of compressor 3 with the input `xOpHoursReset` of the AFB `CmpCtrl_VS` or `CmpCtrl_OnOff`.
`uiAlertID` 0000 0000 0000 1000	The operating hours of compressor 4 > 16,700,100	Until 16,700,100 operating hours, it is possible to reset the operating hours of compressor 4 with the input `xOpHoursReset` of the AFB `CmpCtrl_VS` or `CmpCtrl_OnOff`.
`uiAlarmID` 0001 0000 0000 0000	System clock alarm	Set a valid date and time value for using the AFB `sysClockWrite` in the target block library.

Chapter 11
Coefficient of Performance Calculation: COPCalculation

What Is in This Chapter?

This chapter contains the following sections:

Section 11.1
Functional Overview

`COPCalculation` Function Block Description

Function Block Description

The coefficient of performance (COP) indicates the efficiency of heating and cooling machines. The higher the COP, the more efficient the machine operates.

The COP is defined by the ratio heat dissipation and electrical power intake.

The `COPCalculation` function block calculates the COP based on the following formula:

COP = thermal power (kW) / electrical power (kW)

seasonal COP = thermal energy (kWh) / electrical energy (kWh)

Why Use the `COPCalculation` Function Block?

The `COPCalculation` function block:

- simplifies programming.
- reduces engineering and commissioning time.
- provides proper functionality verified on real installations.

Features of the `COPCalculation` Function Block

The `COPCalculation` function block provides the following features:

- implements methods to calculate the instant machine COP.
- provides a short-term COP indicating the average COP over a duration of 5 min.
- calculates the average COP for a user-defined duration (seasonal COP).
- provides an input to reset all accumulated COP values and indicates the date of the last reset.

Error Avoidance Features

The function block provides the following error avoidance feature to help avoid certain sources of machine malfunction:

Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.

Section 11.2
Pin Description

Pin Description

Pin Diagram

The following picture presents the pin diagram of the `COPCalculation` function block:

Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
`xEn`	`BOOL`	TRUE or FALSE	N/A	Enables the function block.
`rElecPower`	`REAL`	N/A	N/A / kW	Electrical power
`rThermalPower`	`REAL`	-3.40E+38... +3.40E+38	N/A / kW	Thermal power
`rElecEnergy`	`REAL`	-3.40E+38... +3.40E+38	N/A / kWh	Electrical energy
`rThermalEnergy`	`REAL`	-3.40E+38... +3.40E+38	N/A / kWh	Thermal energy
`ptrE2_EnergyRead`	`@BOOL`	N/A	N/A	EnergyRead pointer for non-volatile memory variable
`ptrE2_ElecEnergySaved`	`@REAL`	N/A	N/A	ElecEnergySaved pointer for non-volatile memory variable

Input	Data Type	Range	Scaling / Unit	Description
ptrE2_Thermal EnergySaved	@REAL	N/A	N/A	ThermalEnergySaved pointer for non-volatile memory variable
ptrE2_COPLastReset	@USINT	N/A	N/A	COP last reset
xReset	BOOL	TRUE or FALSE	N/A	Reset input to restart long-term COP calculation.

Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
uiLongTermCOP	USINT	0...65535	0.01	Long-term COP (user-defined)
uiShortTermCOP	USINT	0...65535	0.01	Short-term COP (sliding window of 5 min)
uiInstantCOP	USINT	0...65535	0.01	Instant COP
ptrSt_COPLastReset	@USINT	N/A	N/A	Pointer for date and time of last reset

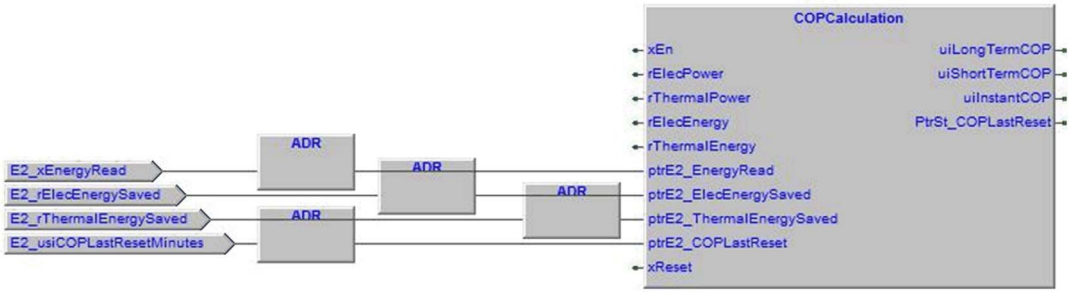
Non-Volatile Memory Variables

Address	Variable	Data Type	Range	Scaling / Unit	Description
N	E2_xEnergyRead	BOOL	0...1	N/A	First time energy read
N+1	E2_rElecEnergy Saved	REAL	-3.40E+38 ...+3.40E+38	kWh	Electric energy saved
N+3	E2_rThermalEnergy Saved	REAL	-3.40E+38 ...+3.40E+38	kWh	Thermal energy saved
N+5	E2_usiCOPLast ResetMinutes	USINT	0...59	Minutes	Minutes when reset is activated.
N+6	E2_usiCOPLast ResetHours	USINT	1...23	Hours	Hours when reset is activated.
N+7	E2_usiCOPLast ResetMonthDay	USINT	1...31	N/A	MonthDay when the Reset command is activated.
N+8	E2_usiCOPLast ResetMonth	USINT	1...12	Minutes	Month when the Reset command is activated
N+9	E2_usiCOPLast ResetYear	USINT	10...99	Years	Year when the Reset command is activated.

uiLongTermCOP

The output uiLongTermCOP provides the average COP for a user-defined duration. The long-term COP calculation is started with a rising edge on the xReset input. Before performing the reset to start a new measurement period, archive the uiLongTermCOP value by storing it to a separate variable.

Connectivity Diagram



NOTE:

- You have to define the parameters of the data type specified in the table Non-Volatile Memory Variables *(see page 212)* in the non-volatile memory parameters in the application.
- You have to use an ADR block to connect the following input pins to the variable defined in the non-volatile memory:
  - ptrE2_EnergyRead
  - ptrE2_ElecEnergySaved
  - ptrE2_ThermalEnergySaved
  - ptrE2_COPLastReset
- If several instances of the function block are created inside the project, you have to create 8 new parameters in the non-volatile memory for each function block:
  - E2_xEnergyRead
  - E2_rElecEnergySaved
  - E2_rThermalEnergySaved
  - E2_usiCOPLastResetMinutes
  - E2_usiCOPLastResetHours
  - E2_usiCOPLastResetMonthDay
  - E2_usiCOPLastResetMonth
  - E2_usiCOPLastResetYear

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Be sure that all variables are initialized to an appropriate value before their first use as array indices or pointers.
- Write programming instructions to test the validity of operands intended to be used as array indices and memory pointers.
- Do not attempt to access an array element outside the defined bounds of the array.

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

Chapter 12
Transform Counted Values to Energy: Counter2Energy

What Is in This Chapter?

This chapter contains the following sections:

Section 12.1
Functional Overview

`Counter2Energy` Function Block Description

Function Block Description

The function block `Counter2Energy` transforms a numeric value to energy information. The `Counter2Energy` function block may be used used with the `Pulse2Counter` function block *(see page 352)* to provide energy information based on pulses received by a meter, or may be used independently.

Why Use the `Counter2Energy` Function Block?

The function block `Counter2Energy`:

- simplifies programming.
- reduces engineering and commissioning time.

Features of the `Counter2Energy` Function Block

The function block `Counter2Energy` provides the following features:

- converts pulses to energy information according to user-defined scaling definitions.
- calculates the active power based on the frequency the input value increases.
- provides active energy information of the counter value from the function block `Pulse2Counter` *(see page 352)* or a fast counter input on the controller.

Error Avoidance Features

The function block provides the following error avoidance feature to help avoid certain sources of machine malfunction:

Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.

Section 12.2
Pin Description

Pin Description

Pin Diagram

The following picture presents the pin diagram of the `Counter2Energy` function block:

Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
`xEn`	`BOOL`	TRUE or FALSE	N/A	Enables the function block
`udiPulseCnt`	`UDINT`	0…4294967295	1 pulse	Pulse count value from `Pulse2Counter` or fast counter input.
`uiPulseConstant`	`UINT`	1...1000	1 pulse / kWh	Number of pulses per kWh provided by the meter.
`ptrE2_TotalActive Energy`	`@REAL`	N/A	N/A	TotalActiveEnergy Pointer for non-volatile memory variable.

uiPulseConstant

The pulses are transformed to energy information according to:
rTotalActiveEnergy = (udiPulseCnt / uiPulseConstant)

If...	Then...
uiPulseConstant is configured as 1 pulse and the number of pulses counted is 100,	the output rTotalActiveEnergy is 100 kWh.
uiPulseConstant is configured as 10 pulses and the number of pulses counted is 100,	the output rTotalActiveEnergy is10 kWh.

Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
rTotalActivePower	REAL	-3.40E+38 ...+3.40E+38	N/A / kW	Active power **NOTE:** The value is zero when udiPulseCnt is zero.
rTotalActiveEnergy	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Total active energy **NOTE:** If rTotalActiveEnergy value reaches the maximum limits, the value is reset and the function block continues to operate.
uiAlarmID	UINT	0...65535	N/A	Alarm identification

Non-Volatile Memory Variable

Address	Variable	Data Type	Range	Scaling / Unit	Description
N	E2_rTotalActive Energy	REAL	-3.40E+38 ...+3.40E+38	kWh	Real total active energy

Alarm ID Description

The uiAlarmID output represents a value from 0 to 3, whereby each bit represents a detected alarm. The table contains the bits and their description:

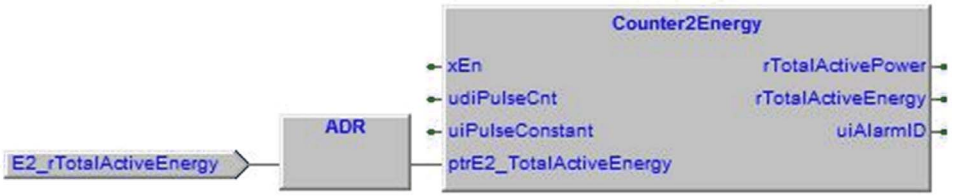
Alarm Bit	Alarm Cause	Effect
0	Invalid configuration of uiPulseConstant. The value is less than 1 or greater than 1000.	The function block stops calculating. The rTotalActiveEnergy value is retained.

Alarm Bit	Alarm Cause	Effect
1	Overflow alarm	The total active energy value has reached the maximum limit. The energy accumulation restarts from zero.
2-15	N/A	N/A

Troubleshooting

Alarm	Problem	Solution
`uiAlarmID.0`	The value is less than 1 or greater than 1000.	Verify that the `uiPulseConstant` value is configured between 1 to 1000.
`uiAlarmID.1`	Overflow alarm	The energy accumulation restarts from zero automatically.

Connectivity Diagram



NOTE:

- You have to define the parameter of the data type specified in the table *Non-Volatile Memory Variables* in the non-volatile memory parameters in the application.
- You have to use an ADR block to connect the input pin `ptrE2_TotalActiveEnergy` to the variable defined in the non-volatile memory.
- If several instances of the function block are created inside the project, you have to create one new parameter (`E2_rTotalActiveEnergy`) in the non-volatile memory for each function block.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Be sure that all variables are initialized to an appropriate value before their first use as array indices or pointers.
- Write programming instructions to test the validity of operands intended to be used as array indices and memory pointers.
- Do not attempt to access an array element outside the defined bounds of the array.

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

Chapter 13
Day Light Saving: DayLightSaving

What Is in This Chapter?

This chapter contains the following sections:

Section 13.1
Functional Overview

`DayLightSaving` Function Block Description

Function Block Description

The function block advances the clock of the controller forward by one hour near the start of spring and adjusted backward to original time in autumn.

Why Use the `DayLightSaving` Function Block?

The `DayLightSaving` function block:

- simplifies programming.
- reduces engineering and commissioning time.

Error Avoidance Features

The function block provides the following error avoidance feature to help avoid certain sources of machine malfunction:

Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.

Section 13.2
Pin Description

Pin Description

Pin Diagram

The following picture presents the pin diagram of the `DayLightSaving` function block:

Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
`usiZone`	`USINT`	0...2	N/A	Zone

`usiZone`

The `usiZone` input must be configured according to the location where the controller is used:

`usiZone`	Zone	Description
0	No Time Zone specified	Daylight saving functionality is disabled.
1	Europe	Daylight saving functionality will start on last Sunday of March at 1:00 a.m. DST and end on last Sunday of October at 2:00 a.m.
2	US/Canada	Daylight saving functionality will start on second Sunday of March at 2:00 a.m. local time and end on first Sunday of November at 3:00 a.m.

Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
`usiStatus`	`USINT`	0...255	N/A	DayLightSaving active status.

usiStatus

The output `usiStatus` represents a value from 0 to 255, whereby each bit represents a status. The table contains the bits and their description:

Bit	Description
0	`usiZone` = 0: function block disabled
1	Summer Time
2	Winter Time
3...5	Reserved
6	`usiZone` input is invalid.
7	Error in setting time.

Chapter 14
Yearly Event: YearlyEvent

What Is in This Chapter?

This chapter contains the following sections:

Section 14.1
Functional and Machine Overview

Functional Overview

Functional Description

The `YearlyEvent` function block allows to trigger one event depending on the current date and time over the year.

Why Use the `YearlyEvent` Function Block?

The `YearlyEvent` function block is used for the following purposes:

- Allows to define one generic On/Off event along the year.
- Generates an error message in case of inadequate input variable data or a detected system clock error.

Features of the `YearlyEvent` Function Block

The function block provides the following features:

- Every time the real time clock (by functions `SysClock` and `SysClockError`) has a Date & Hours between Start Date & Hours (`i_usiStartMonth`, `i_usiStartDaymonth` and `i_usiStartHours`) and Stop Date & Hours (`i_usiStopMonth`, `i_usiStopDaymonth` and `i_usiStopHours`), then the digital output `q_xEvent` is set to TRUE.
- Date & Hours means month (1...12), day of the month (1...28 or 29; 30 or 31 depending which month) and hours (0...23). Note, that the information on year and minutes are is not considered.
- The Stop Date & Hours can take place before or after the Start Date & Hours. This way it is possible to set an event over the end of the year, for example a Start Date & Hours on 24 Dec at 20:00, and a Stop Date & Hours on 7 Jan at 8:00.
- A digital error is generated (`q_xError`) in case of a detected clock error or in case of an incorrect configuration. If Start Date & Hours is out of range, a limited error control is possible for February, where 29 days are always accepted.

Error Avoidance Features

The function block provides the following error avoidance features to help prevent certain sources of machine malfunction:

Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.
System clock validation	System clock configuration or error.

Section 14.2
Pin Description

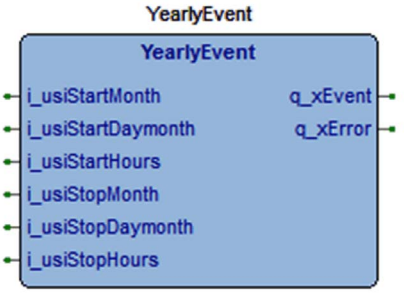
What Is in This Section?

This section contains the following topics:

Input Pin Description

Pin Diagram

The following graphic shows the pin diagram of `YearlyEvent`:



The following graphic shows an example of an instance:

Input Pin Description

Input	Data Type	Range	Scaling/Unit	Description
`i_usiStartMonth`	`USINT`	1...12	N/A	Indicates the month when the event starts.
`i_usiStartDaymonth`	`USINT`	1...31	N/A	Indicates the day of the month when the event starts. The maximum number depends on the number of days of the month.
`i_usiStartHours`	`USINT`	0...23	N/A	Indicates the hour when the event starts.
`i_usiStopMonth`	`USINT`	1...12	N/A	Indicates the month when the event stops.

Input	Data Type	Range	Scaling/Unit	Description
i_usiStopDaymonth	USINT	1...31	N/A	Indicates the day of the month when the event stops. The maximum depends on number of days of the month.
i_usiStopHours	USINT	1...23	N/A	Indicates the hour when the event stops.

Output Pin Description

Output Pin Description

Output Pin Description

Output	Data Type	Range	Scaling/Unit	Description
q_xEvent	BOOL	TRUE or FALSE	N/A	FALSE: Event inactive TRUE: Event active
q_xError	BOOL	TRUE or FALSE	N/A	FALSE: No error active TRUE: Error active

Error Description

The q_xError output represents a boolean value from FALSE to TRUE, if TRUE means the alarm condition active, in particular:

Error Cause	Result
Real time clock is not configured or in error. This alarm is set immediately to TRUE and it remains TRUE as long as the clock is not configured or is in error.	q_xError = TRUE
At least one input variable is out of range.This alarm is set immediately to TRUE and it remains TRUE until correction.	

Section 14.3
Troubleshooting

Troubleshooting

Troubleshooting

Problem	Solution
Real time clock is not configured or in error.	Verify the real time clock. Set the time of the controller.
At least one input variable is out of range.	Verify the input variables and set the value within the range.

Chapter 15
Week Schedule: WeekSchedule

What Is in This Chapter?

This chapter contains the following sections:

Section 15.1
Functional and Machine Overview

Functional Overview

Functional Description

The `WeekSchedule` function block allows to trigger one or more events depending on the time and day of week.

When the day and the time of the system clock match with the day and time of the event, the block activates a digital output and indicates the event ID.

Why Use the `WeekSchedule` Function Block?

The `WeekSchedule` function block is used for the following purposes:

- Allows to define many type events, apart from ON and OFF.
- Enables event triggering and indicates the overlap of events.
- Allows to define up to 255 different events or different timings for the same event. This number depends also on the non-volatile memory *(see page 237)* capabilities of the device.

Features of the `WeekSchedule` Function Block

The function block provides the following features:

- Every time the real time clock (by the functions `SysClock` and `SysClockError`) reaches one of the predefined time events of the week, the output of `xTriggerEvent` is set to TRUE, and `usiLastEvent` takes the ID value related to the event.
- If there is a transition from FALSE to TRUE in `xResetEvent`, `xTriggerEvent` is set to FALSE (note that `usiLastEvent` remains at the last value).

Error Avoidance Features

The function block provides the following error avoidance features to help prevent certain sources of machine malfunction:

Error Avoidance Features	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.
Verification of overlapping events	If some events are overlapping an alert regarding a possible loss of events is given.

Section 15.2
Pin Description

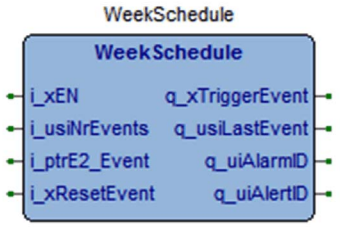
What Is in This Section?

This section contains the following topics:

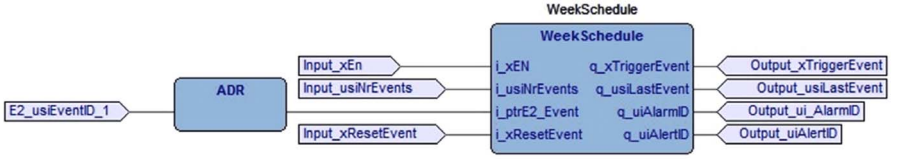
Input Pin Description

Pin Diagram

The following graphic shows the pin diagram of `WeekSchedule`:



The following graphic shows an example of an instance:



Input Pin Description

Input	Data Type	Range	Scaling/Unit	Description
`i_xEn`	`BOOL`	TRUE or FALSE	N/A	Enables the function block.
`i_usiNrEvents`	`USINT`	1...20	N/A	Number of events.
`i_ptrE2_Event`	`USINT`	N/A	N/A	Event pointer for non-volatile memory variable.
`i_xResetEvent`	`BOOL`	TRUE or FALSE	N/A	Resets the trigger output. TRUE = reset trigger output

Non-Volatile Memory Variables

Address	Variable	Data Type	Range	Scaling/Unit	Description
N	E2_usiEventID_1	USINT	0...255	N/A	ID of event 1.
N+1	E2_usiEventWeekDays_1	USINT	0...127	N/A	Days of week *(see page 237)* when event 1 is triggered.
N+2	E2_uiEventTime_1	UINT	0...1439 0 = 00:00 1439 = 23:59	min	Time when event 1 is triggered.
N+3	E2_usiEventID_2	USINT	0...255	N/A	ID of event 2.
N+4	E2_usiEventWeekDays_2	USINT	0...127	N/A	Days of week when event 2 is triggered.
N+5	E2_uiEventTime_2	UINT	0...1439 0 = 00:00 1439 = 23:59	h:min	Time when event 2 is triggered.
...	...	...	...	...	...
N + 3 x (byNrEvents -1)	E2_usiEventID_byNrEvents	USINT	0...255	N/A	ID of event byNrEvents.
N + 3 x (byNrEvents - 1) + 1	E2_usiEventWeekDays_byNrEvents	USINT	0...127	N/A	Days of week when event byNrEvents is triggered.
N + 3 x (byNrEvents - 1) + 2	E2_uiEventTime_byNrEvents	UINT	0...1439 0=00:00 1439 = 23:59	h:min	Time when event byNrEvents is triggered.

E2_usiEventWeekDays_x Bit

E2_usiEventWeekDays_x Bit	Day
0	Sunday
1	Monday
2	Tuesday
3	Wednesday
4	Thursday
5	Friday
6	Saturday
7	Not used

Output Pin Description

Output Pin Description

Output Pin Description

Output	Data Type	Range	Scaling/Unit	Description
qxTriggerEvent	BOOL	TRUE or FALSE	N/A	FALSE: Waiting for event. TRUE: Event occurred.
q_usiLastEvent	USINT	0...255	N/A	ID of last triggered event. Default: 0
q_uiAlarmID	UINT	0…65535	N/A	Alarm ID
q_uiAlertID	UINT	0…65535	N/A	Alert ID

Alarm ID Description

The q_uiAlarmID output represents a value from 0 to 65535, whereby each bit represents a detected alarm. The table contains the bit positions and their description:

Alarm Bit	Alarm Cause	Effect
0	Real time clock is not configured or is detected in error. This alarm is set to TRUE immediately and it remains TRUE, as long as the clock is not configured or is in error.	xTriggerEvent = FALSE usiLastEvent = 0
1...15	N/A	N/A

Alert ID Description

The q_uiAlertID output represents a value from 0 to 65535, whereby each bit represents a detected alert. The table contains the bit positions and their description:

Alarm Bit	Alarm Cause	Effect
0	Overlapping events. This alert is set to TRUE since the time/day coincidence and it remains TRUE until the next event without coincidence.	If 2 or more events are coincident, usiLastEvent = event with bigger ID.
1	Parameter E2_uiEventTime_x is out of range. This alert is set immediately to TRUE and it remains TRUE, until parameter correction.	The event x has no effects on xTriggerEvent and usiLastEvent.
2...15	N/A	N/A

Section 15.3
Troubleshooting

Troubleshooting

Troubleshooting

Problem	Solution
Real time clock is not configured or has a detected error.	Verify the real time clock. Set the time of the controller.
Overlapping events	Verify the parameters in `E2` area with the same day (`E2_usiEventWeekDays_x`) and the same time (`E2_uiEventTime_x`).
`Parameter E2_uiEventTime_x` is out of range.	Verify that `E2_uiEventTime_x` parameters are within their range (0...1439).

Chapter 16
Lagrange Interpolation on a 5x8 Table: DoubleInterpo_5x8

What Is in This Chapter?

This chapter contains the following sections:

Section 16.1
Functional Overview

`DoubleInterpo_5x8` Function Block Description

Function Block Description

The function block `DoubleInterpo_5x8` makes a lagrange interpolation on a 5x8 table.

Why Use the `DoubleInterpo_5x8` Function Block?

The function block `DoubleInterpo_5x8`

- is based on the conversion lagrange interpolation (polynomial interpolation).
- can be used with any 5x8 table.

Lagrange polynomials are used for polynomial interpolation. For a given set of distinct points x_j and numbers y_j, the lagrange polynomial is the polynomial of the least degree that at each point x_j assumes the corresponding value y_j (i.e. the functions coincide at each point)[1]. Here we will use second order polynomial interpolation (also called quadratic interpolation)[2].

1 http://en.wikipedia.org/wiki/Lagrange_polynomial

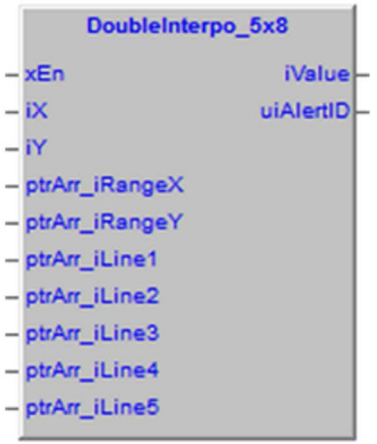
2 http://mathforcollege.com/nm/mws/gen/05inp/mws_gen_inp_txt_lagrange.pdf - page 6

Section 16.2
Pin Diagram

Pin Description

Pin Diagram

The following picture presents the pin diagram of the `DoubleInterpo_5x8` function block:



Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
`xEn`	`BOOL`	TRUE or FALSE	N/A	TRUE: Enables the function block. FALSE: Disables the function block.
`iX`	`INT`	-32768...+32767	User precision and Unit	Input value for interpolation
`iY`	`INT`	-32768...+32767	User precision and Unit	Input value for interpolation
`ptrArr_iRangeX`	`@INT`	N/A	N/A	Table header pointer (lines)
`ptrArr_iRangeY`	`@INT`	N/A	N/A	Table header pointer (columns)
`ptrArr_iLine1`	`@INT`	N/A	N/A	Table values pointer (line 1)

Input	Data Type	Range	Scaling / Unit	Description
ptrArr_iLine2	@INT	N/A	N/A	Table values pointer (line 2)
ptrArr_iLine3	@INT	N/A	N/A	Table values pointer (line 3)
ptrArr_iLine4	@INT	N/A	N/A	Table values pointer (line 4)
ptrArr_iLine5	@INT	N/A	N/A	Table values pointer (line 5)

Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
iValue	INT	-32768...+32767	User precision and Unit	Result of the interpolation
uiAlertID	UINT	0...6	N/A	Alert identification

iValue

The output `iValue` returns the result of the interpolation based on the inputs `iX` and `iY` and the parameters of the 5X8 table.

Double interpolation illustration:

	Arr_iRangeY[0]	...	**iY**	...	Arr_iRangeY[7]
Arr_iRangeX[0]	Arr_iLine1[0]	...		...	Arr_iLine1[7]
Arr_iRangeX[1]	Arr_iLine2[0]	...		...	Arr_Line2[7]
Arr_iRangeX[2]	Arr_iLine3[0]	...		...	Arr_Line3[7]
Arr_iRangeX[3]	Arr_iLine4[0]	...		...	Arr_Line4[7]
iX			**iValue**		
Arr_iRangeX[4]	Arr_iLine5[0]	...		...	Arr_Line5[7]

Array Variables

Name	Data Type	Range	Array	Scaling/Unit	Description
Arr_iRangeX	INT	-32768...+32767	[0...4]	User precision and Unit	Table header (lines)
Arr_iRangeY	INT	-32768...+32767	[0...7]	User precision and Unit	Table header (columns)
Arr_iLine1					Table values (line 1)
Arr_iLine2					Table values (line 2)
Arr_iLine3					Table values (line 3)
Arr_iLine4					Table values (line 4)
Arr_iLine5					Table values (line 5)

Alert ID Description

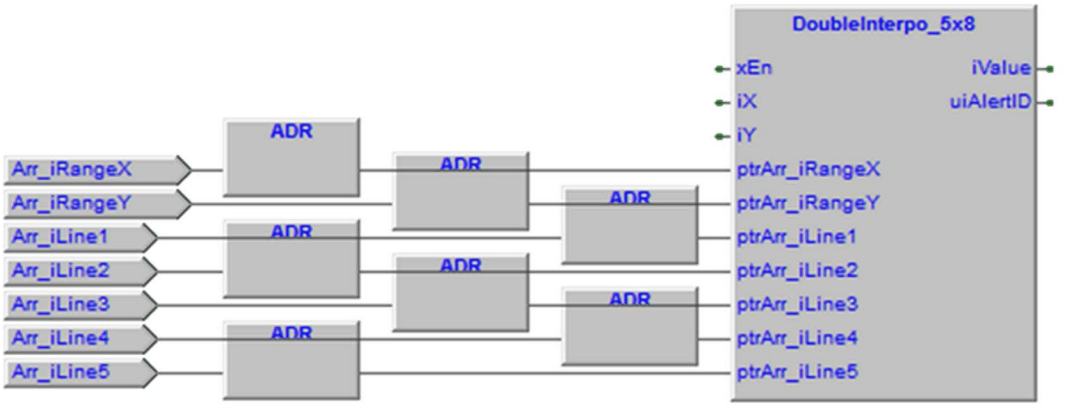
The `uiAlertID` output represents a value from 0 to 6, whereby each bit represents a detected alert. The table contains the bits and their description:

Alert Bit	Alarm Cause	Effect
0	`iX` is not within the specified range.	The value of `iX` is limited between `iRangeX[0]` and `iRangeX[4]`.
1	`iY` is not within the specified range.	The value of `iY` is limited between `iRangeY[0]` and `iRangeY[7]`.
2-15	N/A	N/A

Troubleshooting

Alarm	Problem	Solution
`uiAlertID.0`	`iX` is not within the specified range.	Check the value of `iX` and `iRangeX[0]` to `iRangeX[4]`.
`uiAlertID.1`	`iY` is not within the specified range.	Check the value of `iY` and `iRangeY[0]` to `iRangeY[7]`.

Connectivity Diagram



NOTE:

- You have to define local or global variables of INT type:
  - `Arr_iRangeX` with array of 5
  - `Arr_iRangeY` with array of 8
  - `Arr_iLine1...Arr_iLine5` with array of 8

- You have to use an ADR block to connect the following input pins to the array variable:
  - `ptrArr_iRangeX`
  - `ptrArr_iRangeY`
  - `ptrArr_iLine1...ptrArr_iLine5` with array of 8
- During the power cycle the array variable would be reset with the default value.
- The precision has no effect on the function block as long as it remains coherent inside lines, columns and ranges.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Be sure that all variables are initialized to an appropriate value before their first use as array indices or pointers.
- Write programming instructions to test the validity of operands intended to be used as array indices and memory pointers.
- Do not attempt to access an array element outside the defined bounds of the array.

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

Chapter 17
Energy Meter Data Trend: EnergyTrend

What Is in This Chapter?

This chapter contains the following sections:

Section 17.1
Functional Overview

`EnergyTrend` Function Block Description

Function Block Description

The function block `EnergyTrend` is provided for applications that require energy monitoring where the detailed energy consumption is needed for user-defined periods. It stores actual and historical energy information on a daily, weekly, monthly, yearly, user-defined base.

It may be used with the function block `Counter2Energy` *(see page 216)* it may be used independently.

Why Use the `EnergyTrend` Function Block?

The function block `EnergyTrend`:

- simplifies programming.
- reduces engineering and commissioning time.

Features of the `EnergyTrend` Function Block

The function block `EnergyTrend` provides the following features:

- stores the total energy consumption per day, week, month, and year.
- stores the total energy consumption for a user-defined period.
- stores all total energy consumption values of the previous measurement period.

Error Avoidance Features

The function block provides the following error avoidance feature to help avoid certain sources of machine malfunction:

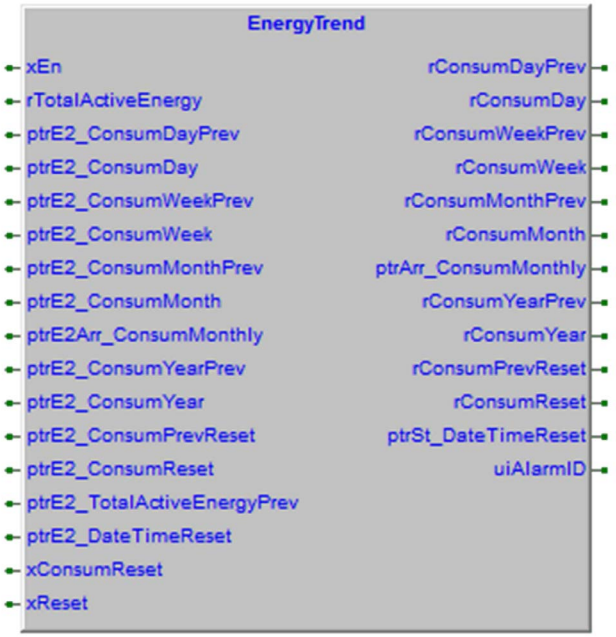
Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.

Section 17.2
Pin Diagram

Pin Description

Pin Diagram

The following picture presents the pin diagram of the `EnergyTrend` function block:



Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
`xEn`	`BOOL`	TRUE or FALSE	N/A	Enables the function block.
`rTotalActiveEnergy`	`REAL`	-3.40E+38 ...+3.40E+38	N/A / kWh	Accumulated energy
`ptrE2_ConsumDayPrev`	`@REAL`	N/A	N/A	ConsumDayPrev pointer for non-volatile memory variable.

Input	Data Type	Range	Scaling / Unit	Description
ptrE2_ConsumDay	@REAL	N/A	N/A	ConsumDay pointer for non-volatile memory variable.
ptrE2_ConsumWeekPrev	@REAL	N/A	N/A	ConsumWeekPrev pointer for non-volatile memory variable.
ptrE2_ConsumWeek	@REAL	N/A	N/A	ConsumWeek pointer for non-volatile memory variable.
ptrE2_ConsumMonthPrev	@REAL	N/A	N/A	ConsumMonthPrev pointer for non-volatile memory variable.
ptrE2_ConsumMonth	@REAL	N/A	N/A	ConsumMonth pointer for non-volatile memory variable.
ptrE2Arr_ConsumMonthly	@REAL	N/A	N/A	ConsumMonthly pointer for non-volatile memory variable.
ptrE2_ConsumYearPrev	@REAL	N/A	N/A	ConsumYearPrev pointer for non-volatile memory variable.
ptrE2_ConsumYear	@REAL	N/A	N/A	ConsumYear pointer for non-volatile memory variable.
ptrE2_ConsumPrevReset	@REAL	N/A	N/A	ConsumPrevReset pointer for non-volatile memory variable.
ptrE2_ConsumReset	@REAL	N/A	N/A	ConsumReset pointer for non-volatile memory variable.
ptrE2_TotalActiveEnergyPrev	@REAL	N/A	N/A	TotalActiveEnergyPrev pointer for non-volatile memory variable.
ptrE2_DateTimeReset	@USINT	N/A	N/A	DateTimeReset pointer for non-volatile memory variable.
xConsumReset	BOOL	TRUE or FALSE	N/A	Trigger to store the current value of rConsumReset into rConsumPrevReset, followed by setting rConsumReset to 0. **NOTE:** The reset is performed on a rising edge.
xReset	BOOL	TRUE or FALSE	N/A	Used to reset the alarm. **NOTE:** The alarm is reset at the rising edge.

Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
rConsumDayPrev	REAL	-3.40E+38 ...+3.40E+38	kWh	Total active energy consumed the previous day.
rConsumDay	REAL	-3.40E+38 ...+3.40E+38	kWh	Total active energy consumed today.
rConsumWeekPrev	REAL	-3.40E+38 ...+3.40E+38	kWh	Total active energy consumed the previous week.
rConsumWeek	REAL	-3.40E+38 ...+3.40E+38	kWh	Total active energy consumed this week.
rConsumMonthPrev	REAL	-3.40E+38 ...+3.40E+38	kWh	Total active energy consumed the last month.
rConsumMonth	REAL	-3.40E+38 ...+3.40E+38	kWh	Total active energy consumed this month.
ptrArr_ConsumMonthly	@REAL	N/A	N/A	Total active energy consumed the last months (January to December)[1].
rConsumYearPrev	REAL	-3.40E+38 ...+3.40E+38	kWh	Total active energy consumed the last year.
rConsumYear	REAL	-3.40E+38 ...+3.40E+38	kWh	Total active energy consumed this year.
rConsumPrevReset	REAL	-3.40E+38 ...+3.40E+38	kWh	User-defined measurement period. Total active energy consumed between the last 2 xConsumReset.
rConsumReset	REAL	-3.40E+38 ...+3.40E+38	kWh	User-defined measurement period. Total active energy consumed since the last xConsumReset.
ptrSt_DateTimeReset	@USINT	N/A	N/A	Date and time of the last xConsumReset performed.
uiAlarmID	UINT	0...65535	N/A	Alarms

1 ptrArr_ConsumMonthly[0] – January consumption,....ptrArr_ConsumMonthly[11] – December consumption.

NOTE:

- Calculations are based on the controller internal clock. Make sure that the clock is configured properly.
- Week data reflect consumption from Sunday 00:00 AM to Saturday 11:59 PM.
- Month data reflect consumption from first day 00:00 AM to last day 11:59 PM.

Non-Volatile Memory Variables

Address	Variable	Data Type	Range	Scaling / Unit	Description
N	E2_rConsumDay Prev	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of the previous day
N+2	E2_rConsumDay	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of the day
N+4	E2_rConsumWeek Prev	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of the previous week
N+6	E2_rConsumWeek	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of the current week
N+8	E2_rConsumMonth Prev	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of the previous month
N+10	E2_rConsumMonth	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of the current month
N+12	E2_rConsumMonth _Jan	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of January
N+14	E2_rConsumMonth _Feb	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of February
N+16	E2_rConsumMonth _March	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of March
N+18	E2_rConsumMonth _April	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of April
N+20	E2_rConsumMonth _May	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of May
N+22	E2_rConsumMonth _June	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of June
N+24	E2_rConsumMonth _July	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of July
N+26	E2_rConsumMonth _Aug	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of August
N+28	E2_rConsumMonth _Sept	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of September
N+30	E2_rConsumMonth _Oct	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of October
N+32	E2_rConsumMonth _Nov	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of November
N+34	E2_rConsumMonth _Dec	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of December
N+36	E2_rConsumYear Prev	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of the previous year

Address	Variable	Data Type	Range	Scaling / Unit	Description
N+38	E2_rConsumYear	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of the current year
N+40	E2_rConsumPrv Reset	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of the previous reset
N+42	E2_rConsumReset	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Energy consumption of the reset
N+44	E2_rTotalActive EnergyPrev	REAL	-3.40E+38 ...+3.40E+38	N/A / kWh	Total active energy of the previous reset
N+46	E2_usiConsume ResetMinutes	USINT	0…59	Minutes	Minutes when Reset command is active.
N+47	E2_usiConsume ResetHours	USINT	0…23	Hours	Hours when Reset command is active.
N+48	E2_usiConsume ResetMonthDay	USINT	1…31	N/A	MonthDay when Reset command is active.
N+49	E2_usiConsume ResetMonth	USINT	1…12	N/A	Month when Reset command is active.
N+50	E2_usiConsume ResetYear	USINT	10…99	Year	Year when Reset command is active.

Alarm ID Description

The uiAlarmID output represents a value from 0 to 1, whereby each bit represents a detected alarm. The table contains the bits and their description:

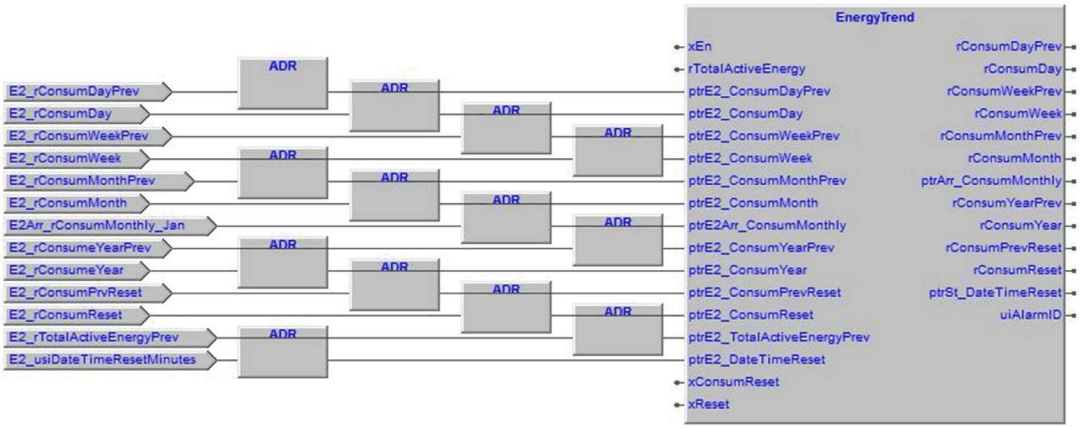
Alarm Bit	Alarm Cause	Effect
0	The value of rTotalActiveEnergy is decreasing.	The function block stops calculating. All total active energy values are retained.
1-15	Not used	N/A

NOTE: Reset the alarm bit 0 with the input pin xReset. After resetting the alarm the outputs are updated with new calculated values.

Troubleshooting

Alarm	Problem	Solution
uiAlarmID.0	The value of rTotalActiveEnergy is decreasing.	Check the source connected to the input rTotalActiveEnergy to provide decreasing energy values.

Connectivity Diagram



NOTE:

- You have to define a parameter in the non-volatile memory parameters in the application.
- You have to use an `ADR` block to connect the 13 input *(see page 249)* pins (`ptrE2_ConsumDayPrev...ptrE2_DateTimeReset`) to the variable defined in the non-volatile memory.
- If several instances of the function block are created inside the project, you have to create 28 new parameters (`E2_rConsumDayPrev...E2_usiDateTimeResetYear`) in the non-volatile memory for each function block. For more details see the table Non-Volatile Memory Variables *(see page 252)*.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Be sure that all variables are initialized to an appropriate value before their first use as array indices or pointers.
- Write programming instructions to test the validity of operands intended to be used as array indices and memory pointers.
- Do not attempt to access an array element outside the defined bounds of the array.

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

Chapter 18
Electronically Commutated Fan Management: EcFanMgmt

What Is in This Chapter?

This chapter contains the following sections:

Section 18.1
Functional and Machine Overview

Functional Overview

Functional Description

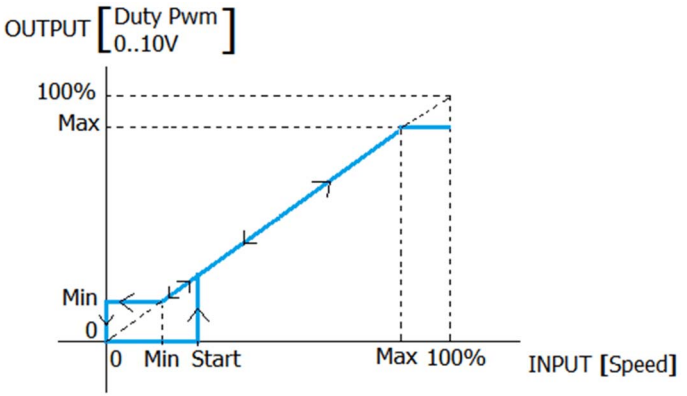
The `EcFanMgmt` (Electronically Commutated Fan Management) function block allows you to operate an electronic controlled fan device, and to evaluate the tacho generator signal coming from the fan itself.

Why Use the `EcFanMgmt` Function Block?

The `EcFanMgmt` function block is used for the following purposes:

- Interface between regulation control and electronic controlled fan.
- Sets PWM Duty or DC voltage values to drive an electronic controlled fan.
- Evaluates the tachometer signal coming from the fan and generates an error message in case of mismatch speeds.

Features of the `EcFanMgmt` Function Block



The `EcFanMgmt` function block manages the following functions of the fan:

- Start-up: The fan input speed must be higher than `i_uiStartSpeed` for a time longer than `i_usiStartTime1s`. The output speed is in any case higher than `i_uiMinSpeed`.
  Stop: When the input speed is equal to 0, the fan is stopped.

- You can set the maximum limit for output speed with `i_uiMaxSpeed`, if required, to a value under 100%.
- A ramp up / ramp down limit (`i_uiRampStep`) can optionally be provided to help prevent quick speed changes.
- The input variables `i_uiSpeed`, `i_uiMinSpeed`, `i_uiMaxSpeed`, `i_uiStartSpeed`, and the output variable `q_uiTacho`, can use 2 different units, based on the input `i_xUnitRpm`: with `i_xUnit` = FALSE, these variables are expressed in decimal percentage values; if `i_xUnitRpm` = TRUE, they are in Rpm.
- Often the electronically commutated fan sources a signal proportional to its real speed, referred to as tachometer. It can be read in decimal of Hertz, with `i_uiPulseRevol` (pulse per revolution). The tachometer measure can be converted to Rpm unit. Another conversion is required to obtain the tachometer measure in decimal of %. In this case, `i_uiRpmAt100` (Rpm at 100% speed) is required.
- The error output (`q_xError`) indicates a mismatching between the output signal (`q_uiDutyPwm_010V`) and the tachometer feedback. To avoid spikes in `q_xError`, a threshold (relative error `i_uiErrorRel`) is required to specify the amount of acceptable difference. The output error is generated only if the speed difference exceeds the specified `i_uiErrorRel` for a time longer than specified `i_usiErrorEvalTime1s` parameter.
  The relative error `i_uiErrorRel` is referred to `i_uiMaxSpeed`. For example, with `i_uiMaxSpeed` = 2700 rpm and `i_uiErrorRel` = 10, the `q_xError` is set to TRUE, if the difference exceeds 270 rpm. In the same way, if `i_uiMaxSpeed` is represented in decimal of %, for example with `i_uiMaxSpeed` = 900 decimal of % and `i_uiErrorRel` = 10, the `q_xError` is set to TRUE if the difference exceeds 90 decimal of %.

Error Avoidance Features

The `EcFanMgmt` function block provides the following error avoidance features to help you avoid the potentials of certain sources of machine malfunction:

Error Avoidance Feature	Description
Input range validation	Input ranges are validated to help prevent out of range data.
Incorrect input setting	Some input variables are validated in terms of their values to help prevent errors regarding functionality.

Section 18.2
Pin Description

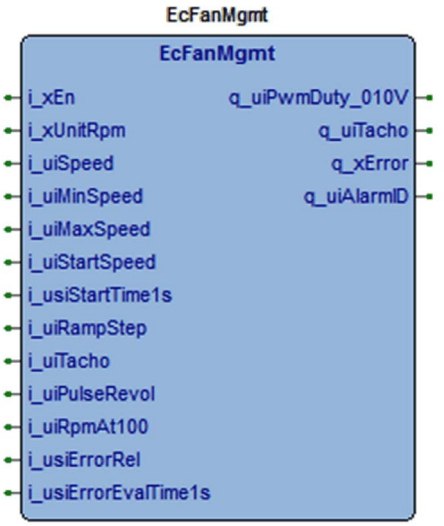
What Is in This Section?

This section contains the following topics:

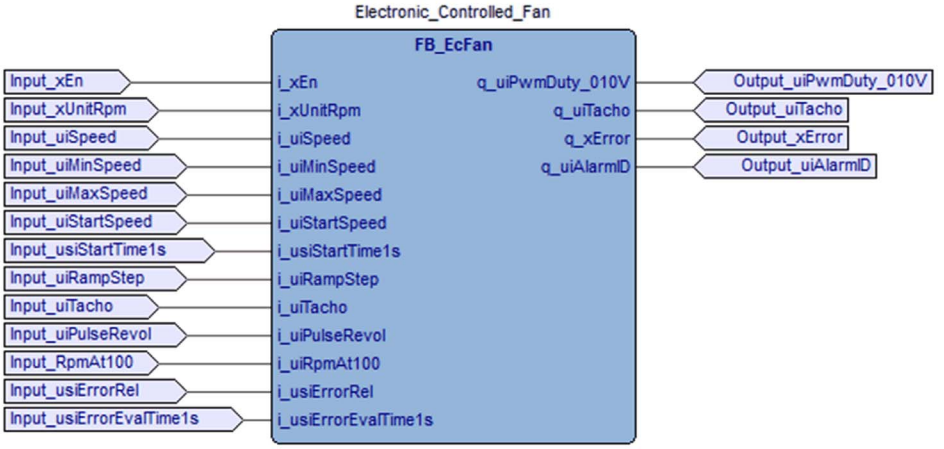
Input Pin Description

Pin Diagram

The following graphic shows the pin diagram of `EcFanMgmt`:



The following graphic shows an example of an instance:



Input Pin Description

Input	Data Type	Range	Scaling/Unit	Description
i_xEn	BOOL	TRUE or FALSE	N/A	Enables the function block.
i_xUnitRpm	BOOL	TRUE or FALSE	N/A	Unit of variables FALSE = Decimal of % TRUE = Rpm
i_uiSpeed	UINT	0...1000 or 0...65535	Decimal of % or Rpm	Speed required
i_uiMinSpeed	UINT	0...1000 or 0...65535	Decimal of % or Rpm	Minimum speed
i_uiMaxSpeed	UINT	0...1000 or 0...65535	Decimal of % or Rpm	Maximum speed i_uiMaxSet ≥ i_uiStartSet
i_uiStartSpeed	UINT	0...1000 or 0...65535	Decimal of % or Rpm	Start speed i_uiStartSet ≥ i_uiMinSet
i_usiStartTime1s	USINT	0...255	s	Minimum time for start speed.
i_uiRampStep	UINT	0...65535	Decimal of % or Rpm	Ramp up / down per unit of time [1 s].
i_uiTacho	UINT	0...65535	Decimal of Hz	Tacho frequency
i_uiPulseRevol	UINT	0...65535	N/A	Tacho pulse in 1 revolution.
i_uiRpmAt100	UINT	0...65535	Rpm	Rpm at 100% of speed
i_usiErrorRel	USINT	0...100	%	Acceptable relation to maximum error in % of speed. Minimum relative error between q_uiPwmDuty_010V and q_uiTacho to raise q_xError
i_usiErrorEvalTime1s	USINT	0...255	s	Waiting time from changed speed to verify the speed error detected.

Output Pin Description

Output Pin Description

Output Pin Description

Output	Data Type	Range	Scaling/Unit	Description
q_uiPwmDuty_010V	UINT	0..1000	Decimal of % or 10 mV	Pwm Duty output or 0...10 V output.
q_uiTacho	UINT	0..1000 or 0..65535	Decimal of % or Rpm	Tachometer output
q_xError	BOOL	FALSE or TRUE	N/A	Detected error over i_usiErrorRel in relation to maximum speed, between speed output and tachometer input.
q_uiAlarmID	UINT	0..65535	N/A	Alarm code

Alarm ID Description

The q_uiAlarmID output represents a value from 0 to 65535, whereby each bit represents a detected alarm. The bits and their description are described in the following table:

Alarm Bit	Alarm Description	Result
0	At least one input is out of range. This alarm is set immediately to TRUE and remains TRUE until correction with limits.	q_uiPwmDuty_010V = 0 q_uiTacho =0 q_xError = FALSE
1	Incorrect input setting, for example: i_uiMinSet > i_uiMaxSet or i_uiMinSet > i_uiStartSet or i_uiStartSet > i_uiMaxSet.	
2-15	N/A	N/A

Section 18.3
Troubleshooting

Troubleshooting

Troubleshooting

Alarm/Alert	Problem	Solution
`uiAlarmID.0`	At least one input is out of range.	Verify the range for each input variable considering the `i_xUnitRpm` selection.
`uiAlarmID.1`	Incorrect input setting.	Verify the values. Required value setting: `i_uiMinSet` ≤ `uiStartSet` ≤ `i_uiMaxSet`.

Chapter 19
Fan Management: FanMgmt

What Is in This Chapter?

This chapter contains the following sections:

Section 19.1
Functional and Machine Overview

What Is in This Section?

This section contains the following topics:

Functional Overview

Functional Description

The `FanMgmt` (Fan Management) function block manages together with the function block `FloatingHighPresCntrl` (Floating Pressure High Control) the condensing pressure in an HVAC&R system.

The `FanMgmt` function block controls up to 4 stages and up to 12 fans per stage and aims to manage the optimum number of fans and their required frequency depending on the required air flow in the machine. For this purpose, `FanMgmt` provides features for switching management and to optimize operation.

Why Use the `FanMgmt` Function Block?

The `FanMgmt` function block is used for the following purposes:

Purpose	Description
Runtime optimization	● increase the control accuracy ● balance operation hours ● avoid frequent ON/OFF switching of fans.
Status management	● switch off a fan in case of a detected error and switch on another.

Features of the `FanMgmt` Function Block

The `FanMgmt` function block manages the following functions of the fan:

- supports the management of 1 to 4 stages and 1 to 12 fans per stage
- balances fan operation hours by using one of the following two methods:
  - FIFO
  - Runtime
  - LIFO
- controls the On/Off sequence of condenser fan stages
- fan switching management in case of a detected fan error
- calculates the required fan speed

Error Avoidance Features

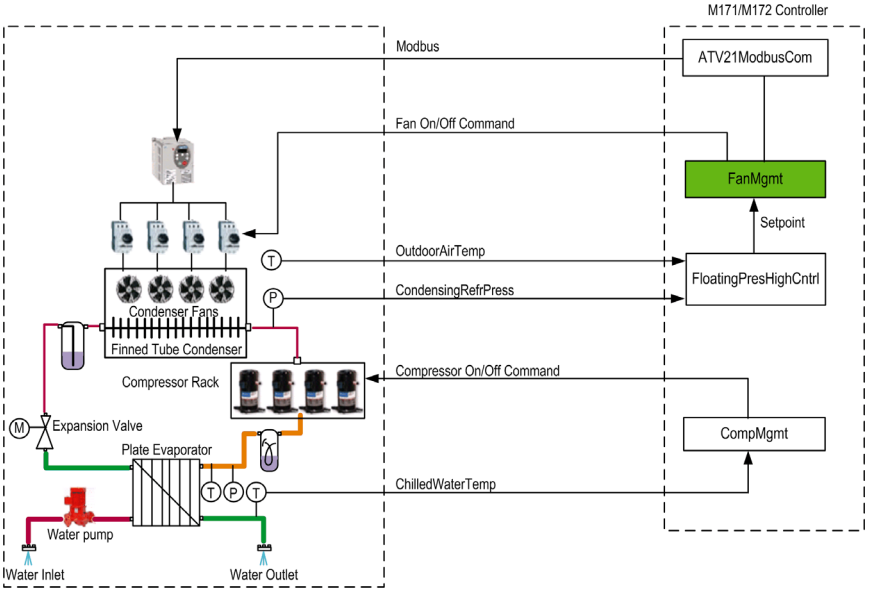
The `FanMgmt` function block provides the following error avoidance features to help you avoid the potentials of certain sources of machine malfunction:

Error Avoidance Features	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.
Alarm/alert notification	If an invalid value is entered, an alarm or an alert is generated: ● Alarm: the compressors are switched off and the function block terminates in error. ● Alert: the function block keeps on operating, however with the possibility of reduced performance.
Controlled parameter	Parameters like `usiFanMode`, `usiNbFanStage1`, `usiNbFanStage2`, `usiNbFanStage3` and `usiNbFanStage4` are controlled. The configuration of these parameters can be changed, however the changes are effective only after the restart of the function block.

Machine Overview

Machine View

The following picture presents the interaction between the function block and the machine:



`FloatingHighPresCntrl` This function block gives out the setpoint for `FanMgmt`.

`FanMgmt` This function block controls the optimum number of fans and the frequency depending on the required air flow in the machine. `FanMgmt` manages the switch-on/off of fan stages.

Section 19.2
Architecture

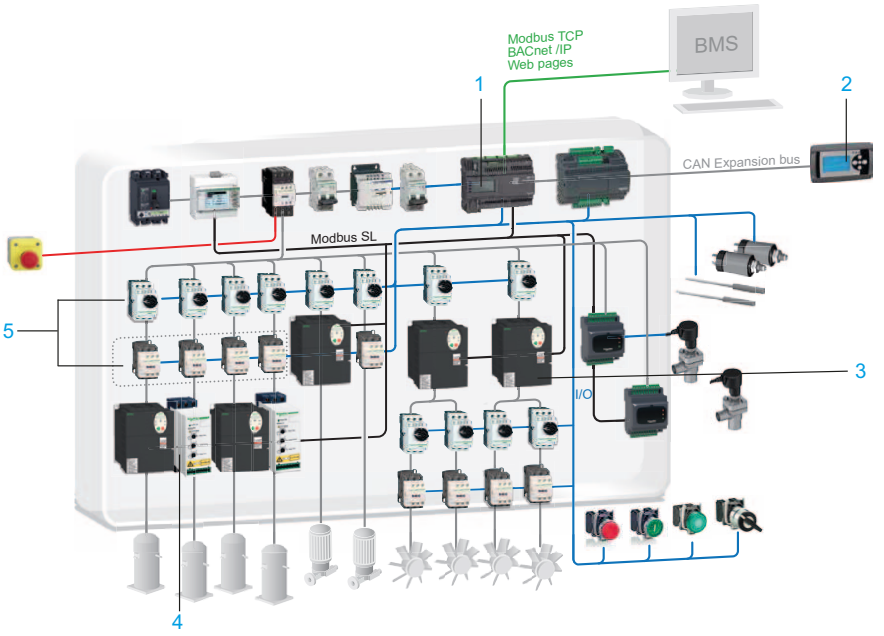
What Is in This Section?

This section contains the following topics:

Hardware Architecture

Hardware Architecture Overview

The following figure presents the hardware architecture of the Air Cooled Chiller.

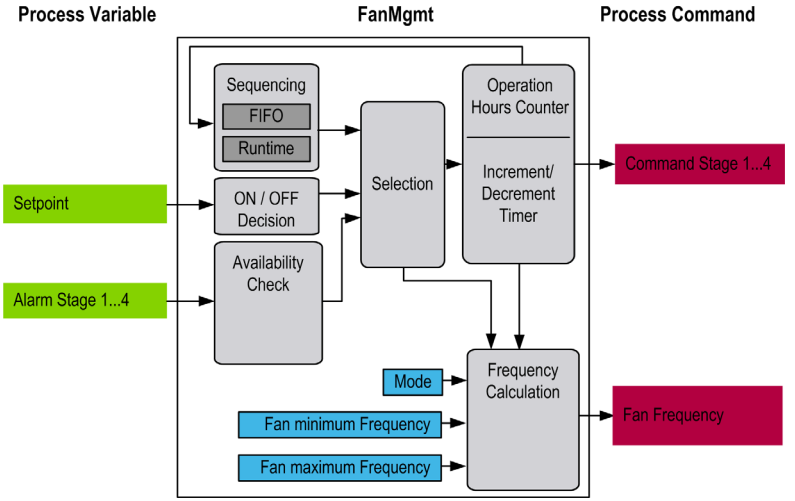


1 M171/M172 Controller
2 Graphic display
3 Variable speed drive ATV••/••• Modbus
4 Soft starters
5 Motor Control

Software Architecture

Function Block Diagram

The following function block diagram gives you an overview of the software architecture:



The block diagram presents on the left hand side the inputs, the **Process Variables**, on the right hand side the outputs, the **Process Commands**, as well as the function block `FanMgmt`

The following table gives you an overview of the functions of `FanMgmt`:

Function	Description
On/Off Decision	evaluates the number of requested fans in regard of the number of stages and fans actually running
Availability Check	• checks the availability for the configured number of fan stages • evaluates fan stage alarms • evaluates fan stage minimum Off time • evaluates fan stage operation hours for switching on
Sequencing	determines which fan is switched on/off next based on two methods: • FIFO • Runtime • LIFO
Operation Hours Counter	counts the accumulated runtime of the fans for: • switch On/Off decisions

Function	Description
Increment/Decrement Timer	controls the fan increment/decrement sequence by means of the following parameters: ● `uiFanDelayIncr` ● `uiFanDelayDecr`
Selection	determines the fan stages to be switched on next, based on the On/Off Timer and Operation Hours Counter
Frequency Calculation	calculates the frequency for currently operating fans based on: ● Mode: Automatic or Manual ● Fan minimum/maximum frequency

Section 19.3
Function Block Description

`FanMgmt` Function Block

Function Block Description

The `FanMgmt` function block controls up to 4 stages and up to 12 fans per stage.

The `FanMgmt` function block manages runtime balancing, detected error switch over and minimum On/Off cycle time of fan stages. This function block works in combination with the `Floating-HighPresCntrl` function block to control the condensation temperature.

The following methods for fan management are provided:
- Fan Stages Sequence Control
- Fan Stages Status Management
- Fan Stages Operation Hours Control
- Fan Frequency Calculation
- Fan Stages Increment / Decrement Timer

Fan Sequence Control

The purpose of the Fan Sequence Control is to balance the number of operation hours and starts/stops between the fan stages.

Fan Sequence Control helps to ensure an even usage of the fan stages and hence helps to protect the fans and optimizes power consumption.

Fan stages are controlled based on the following sequences:

Sequence	Description
FIFO = First In First Out	● The fan with the least operating hours is switched on first. ● The first fan which is switched on is also the first to be switched off. ● Advantage: operation time is limited.
Runtime	● The fan with the least operating hours is the first fan to be switched on. ● The fan with the greatest operating hours is the first fan to be switched off. ● Advantage: balanced operation hours.

Sequence	Description
LIFO = Last In First Out	• The parameter `usiPriorityStage1...usiPriorityStage4` determines this sequence. • The first fan stage to be switched on is the first one in the sequence. • The first fan stage to be switched off is the last one that has been switched on. • Advantage: priority of fan stage usage can be set. • If fan stages have the same priority, the starting sequence is based on the operating hours.

NOTE: If the number of fans per stage is not equal for each stage, only the LIFO mode is available.

Fan Status Management

If an input `xStage.Alarm` is set to TRUE, a fan stage has at least a non-operating fan, the function block assumes that only 1 fan per stage is not operating.

The function block re-calculates the capacity and adapts the frequency setpoint and the number of running fans to compensate the fan loss.

Fan Stages Operation Hours Control

The fan stage operation hours are calculated for each fan stage and influence the switch ON/OFF behavior of the fan stage.

The total accumulated operation hours for each fan are displayed for each fan by the outputs `udiStage1OperHours...udiStage4OperHours`.

The fan stages operationg hours are reset when the input `xStageOperHoursRst` is set to TRUE.

Fan Frequency Calculation

The fan frequency calculation is controlled by 2 modes specified in the `xMode` parameter.

- Manual mode: `xMode` = TRUE
- Automatic mode: `xMode` = FALSE

The following table provides an overview of the different modes:

Mode	Description
Manual	The fan speed signal is set to the frequency `uiManualFreq` and the fan stages are controlled with the inputs `xManualStage1`, `xManualStage2`, `xManualStage3` and `xManualStage4`. **NOTE:** Take care that the frequency `uiManualFreq` is set within the specified ranges of the drive and that the input signals `xManualStage1`, `xManualStage2`, `xManualStage3` and `xManualStage4` are set to TRUE only if the fan stages are present.

Mode	Description
Auto	• The fan frequency and the number of operating fan stages are automatically calculated. • When all fans are in operation, `FanControlSignal` is set as fan speed signal after limiting between `FanFreqMin` and `FanFreqMax` values. • If `xLowNoiseOper` = TRUE, the maximum frequency `uiFanFreqMax` is reduced by a value specified in the parameter `uiLowNoiseMaxFreq`.

Fan Increment / Decrement Timer

The `FanMgmt` function block controls the fan increment sequence by the delay time set in the input parameters:

- `uiFanDelayIncr`
- `uiFanDelayDecr`

The `FanMgmt` function block differs the increment and the decrement sequence:

Sequence	Description
Increment Sequence	When the number of fans is incremented, if the timer `uiDelayFanIncr` is active, the operating fan stages run at the frequency `uiFanFreqMax`.
Decrement Sequence	When the number of fans is decremented, if the timer `uiDelayFanDecr` is active, the operating fan stages run at the frequency `uiFanFreqMin`.

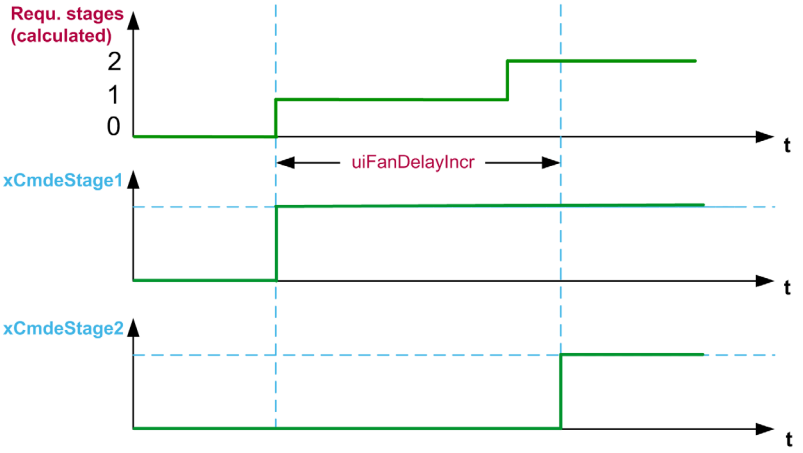
Fan Stage Minimum OFF Time

The function block controls the minimum Off time of the fan stages. When a fan stage is switched off, the timer `uiMinOffTime` is started. The fan stages is not available until the timer `uiMinOffTime` has elapsed.

If all the fan stages are not available, the function block starts anyway a fan stages according to the priority defined by the fan mode `usiFanMode`.

Fan Increment Sequence

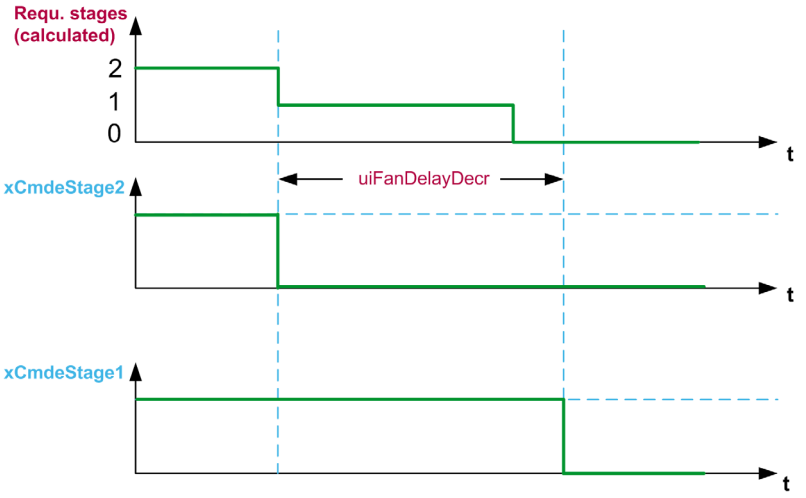
The timing diagram below describes the incrementing sequence of the `FanMgmt` function block:



If...	Then...
the number of required fans is set from 0 to 1	• `xCmdeStage1` is switched on `FanDelayCmd` • The next fan stage can be switched on only after the `uiFanDelayIncr` timer is complete
the number of required fans is set from 1 to 2	`xCmdeStage2` is switched on after the time delay `uiFanDelayIncr` is complete.

Fan Decrement Sequence

This timing diagram presents the decrement sequence of the `FanMgmt` function block:



If...	Then...
the number of required fans is set from 2 to 1	• `xCmdeStage2` is switched Off after the `uiFanDelayIncr` is complete. • The next fan can be switched Off only after the `uiFanDelayIncr` timer is complete
the number of required fans is set from 1 to 0	`xCmdeStage1` is switched Off after the `uiFanDelayIncr` is complete.

Section 19.4
Pin Description

What Is in This Section?

This section contains the following topics:

Input Pin Description

Pin Diagram

The following picture presents the pin diagram of `FanMgmt`:

Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
xEn	BOOL	TRUE or FALSE	N/A	TRUE: Enables the function block FALSE: Disables the function block
uiFanCntrlSignal	UINT	0...1000	0.1%	Fan control signal
xStage1Alarm	BOOL	TRUE or FALSE	N/A	TRUE: Stage1 alarm is active FALSE: Stage1 alarm is not active
xStage2Alarm	BOOL	TRUE or FALSE	N/A	TRUE: Stage2 alarm is active FALSE: Stage2 alarm is not active
xStage3Alarm	BOOL	TRUE or FALSE	N/A	TRUE: Stage3 alarm is active FALSE: Stage3 alarm is not active
xStage4Alarm	BOOL	TRUE or FALSE	N/A	TRUE: Stage4 alarm is active FALSE: Stage4 alarm is not active
xLowNoiseOper	BOOL	TRUE or FALSE	N/A	TRUE: Low noise operation activates FALSE: Low noise operation deactivates
usiNbStage	USINT	1...4	N/A	Number of stages
usiNbStage1	USINT	1...12	N/A	Number of fans for stage1
usiNbStage2	USINT	1...12	N/A	Number of fans for stage2
usiNbStage3	USINT	1...12	N/A	Number of fans for stage3
usiNbStage4	USINT	1...12	N/A	Number of fans for stage4
usiFanMode	USINT	0...2	N/A	Fan On/Off sequence mode **0** FIFO **1** Runtime **2** LIFO
usiPriorityStage1	USINT	0...255	N/A	Priority of stage1 in LIFO mode
usiPriorityStage2	USINT	0...255	N/A	Priority of stage2 in LIFO mode
usiPriorityStage3	USINT	0...255	N/A	Priority of stage3 in LIFO mode
usiPriorityStage4	USINT	0...255	N/A	Priority of stage4 in LIFO mode
uiFanFreqMin	UINT	50...5000	0.1 Hz	VSD minimum frequency
uiFanFreqMax	UINT	51...5000	0.1 Hz	VSD maximum frequency
xMode	BOOL	TRUE or FALSE	N/A	TRUE: Manual mode FALSE: Automatic mode
xManualStage1	BOOL	TRUE or FALSE	N/A	Manual mode of stage1
xManualStage2	BOOL	TRUE or FALSE	N/A	Manual mode of stage2
xManualStage3	BOOL	TRUE or FALSE	N/A	Manual mode of stage3
xManualStage4	BOOL	TRUE or FALSE	N/A	Manual mode of stage4

Input	Data Type	Range	Scaling / Unit	Description
uiManualFreq	UINT	0...65535	0.1 Hz	Frequency in manual mode
uiLowNoiseMaxFreq	UINT	uiFanFreqMin ... uiFanFreqMax	0.1 Hz	VSD low noise operation frequency reduction
uiFanDelayIncr	UINT	0...60	1 s	Delay to increment the running fan stages
uiFanDelayDecr	UINT	0...60	1 s	Delay to decrement the running fan stages
uiMinOffTime	UINT	0...65535	1 s	Minimum Off time of the stages
xOpHoursRst	BOOL	TRUE or FALSE	N/A	Reset the operating hours of all the stages
uiHysteresis	UINT	1...5000	0.1 Hz	Hysteresis for the frequency
ptrE2_OpHrsSt1	UDINT	N/A	N/A	Operating hours of stage1 pointer for non-volatile memory variable
ptrE2_OpHrsSt2	UDINT	N/A	N/A	Operating hours of stage2 pointer for non-volatile memory variable
ptrE2_OpHrsSt3	UDINT	N/A	N/A	Operating hours of stage3 pointer for non-volatile memory variable
ptrE2_OpHrsSt4	UDINT	N/A	N/A	Operating hours of stage4 pointer for non-volatile memory variable

Output Pin Description

Output Pin Description

Output Pin Description

Output	Data Type	Range	Scaling/Unit	Description
usiState	USINT	0...3	N/A	State: **0** Idle **1** Run **2** Alert **3** Alarm
xVSD	BOOL	TRUE or FALSE	N/A	Command for the variable speed drive
uiFanFreq	UINT	0...1000	0.1 / Hz	Fan speed signal to VSD
xCmdeStage1	BOOL	TRUE or FALSE	N/A	Stage1 command
xCmdeStage2	BOOL	TRUE or FALSE	N/A	Stage2 command
xCmdeStage3	BOOL	TRUE or FALSE	N/A	Stage3 command
xCmdeStage4	BOOL	TRUE or FALSE	N/A	Stage4 command
udiOpHoursSt1	UDINT	$0...2^{32}-1$	1 / hours	Stage1 total operating hours
udiOpHoursSt2	UDINT	$0...2^{32}-1$	1 / hours	Stage2 total operating hours
udiOpHoursSt3	UDINT	$0...2^{32}-1$	1 / hours	Stage3 total operating hours
udiOpHoursSt4	UDINT	$0...2^{32}-1$	1 / hours	Stage4 total operating hours
uiAlertID	UINT	0...65535	N/A	Alert identification
uiAlarmID	UINT	0...65535	N/A	Alarm identification

Non-Volatile Memory Variables

Address	Variable	Data Type	Range	Scaling / Unit	Description
N	E2_udiOpHrsSt1	UDINT	$0...2^{32}$	Hours	Operating hours of stage1
N+2	E2_udiOpHrsSt2	UDINT	$0...2^{32}$	Hours	Operating hours of stage2
N+4	E2_udiOpHrsSt3	UDINT	$0...2^{32}$	Hours	Operating hours of stage3
N+6	E2_udiOpHrsSt4	UDINT	$0...2^{32}$	Hours	Operating hours of stage4

Alert ID Description

The uiAlertID output represents a value between 0 and 65535, whereby each bit represents an alert. The bits and their description are described in the following table:

Alert Bit	Alert Description	Result
0	A controlled parameter has been changed. The new configuration parameter is effective only after a restart of the function block. List of the latched parameters: • usiFanMode • uiFanFreqMax • uiFanFreqMin • uiHysteresis • usiNbStage • usiNbFanStage1 • usiNbFanStage2 • usiNbFanStage3 • usiNbFanStage4 • usiPriorityStage1 • usiPriorityStage2 • usiPriorityStage3 • usiPriorityStage4	Present changes are not active. Function block uses the previously set values.
1	The value of the input uiFanCntrlSignal is not set within the specified range.	The value is limited.
2	The value of the parameter uiLowNoiseMaxFreq is not set within the specified range.	The value is limited.
3	The value of the parameter uiFanDelayIncr is not set within the specified range.	Function is in operation with limited performance.
4	The value of the parameter uiFanDelayDecr is not set within the specified range.	Function is in operation with limited performance.
5	xStage1Alarm input is active	An fan of the stage1 is in alarm state, the fan loss is compensated.
6	xStage2Alarm input is active	An fan of the stage2 is in alarm state, the fan loss is compensated.
7	xStage3Alarm input is active	An fan of the stage3 is in alarm state, the fan loss is compensated.
8	xStage4Alarm input is active	An fan of the stage4 is in alarm state, the fan loss is compensated.
9...15	Not used	N/A

Alarm ID Description

The `uiAlarmID` output represents a value between 0 and 3, whereby each bit represents a detected alarm. The bits and their description are described in the following table:

Alarm Bit	Alarm Description	Result
0	The value of the parameter `usiNbStage` is not set within the specified range.	Function block is disabled.
1	The minimum frequency is greater than the maximum frequency (`uiFanFreqMin` > `uiFanFreqMax`)	Function block is disabled.
2	The value of the parameters `usiNbFanStage1`...`usiNbFanStage4` are not set within the specified range.	Function block is disabled.
3	The value of the parameter `usiFanMode` is not set within the specified range.	Function block is disabled.
4	The value of the parameter `uiHysteresis` is not set within the specified range.	Function block is disabled.
5	The value of the parameter `uiFanFreqMin` is not set within the specified range.	Function block is disabled.
6	The value of the parameter `uiFanFreqMax` is not set within the specified range.	Function block is disabled.
7	The number of fans per stage `usiNbFanStage`, is different and the fan mode `usiFanMode` is not equal to 2 (LIFO mode).	Function block is disabled.

Section 19.5
Troubleshooting

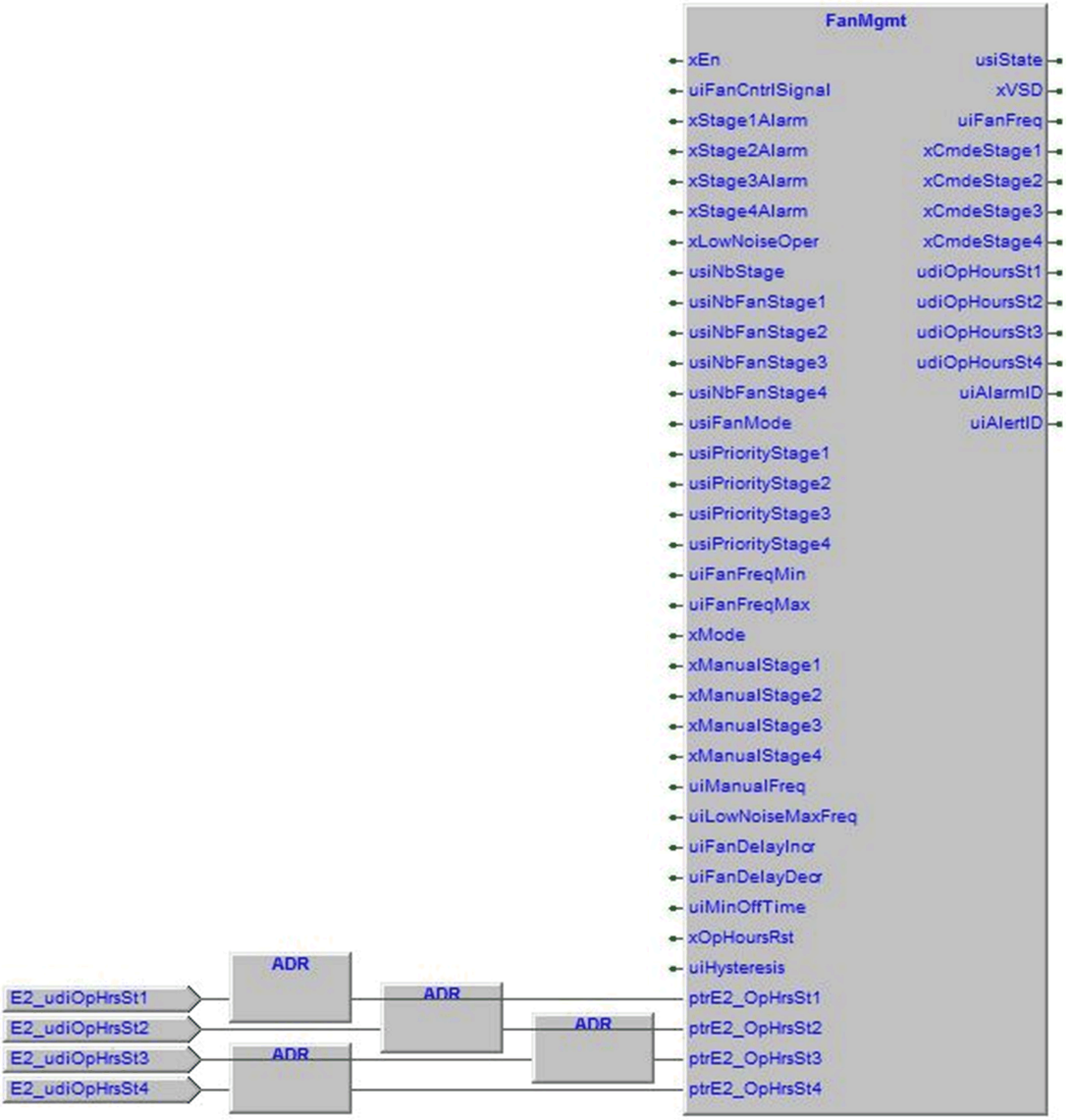
Troubleshooting

Troubleshooting

Alarm/Alert	Problem	Solution
`uiAlarmID.0 TRUE`	The value of the parameter `usiNbStage` is not set within the specified range.	Verify that the parameter values are within their respective ranges.
`uiAlarmID.1 TRUE`	The minimum frequency is greater than the maximum frequency `uiFanFreqMin` > `uiFanFreqMax`	Set the minimum frequency lower than the maximum frequency.
`uiAlarmID.2 TRUE`	The value of the parameter `usiNbStage1`...`usiNbStage4` is not set within the specified range.	Verify that the parameter values are within their respective ranges.
`uiAlarmID.3 TRUE`	The value of the parameter `usiFanMode` is not set within the specified range.	Verify that the parameter values are within their respective ranges.
`uiAlarmID.4 TRUE`	The value of the parameter `uiHysteresis` is not set within the specified range.	Verify that the parameter values are within their respective ranges.
`uiAlarmID.5 TRUE`	The value of the parameter `uiFanFreqMin` is not set within the specified range.	Verify that the parameter values are within their respective ranges.
`uiAlarmID.6 TRUE`	The value of the parameter `uiFanFreqMax` is not set within the specified range.	Verify that the parameter values are within their respective ranges.
`uiAlarmID.7 TRUE`	The number of fans per stage `usiNbFanStage` is different and the fan mode `usiFAnMode` is not equal to 2 (LIFO mode)	If the number of fans per stage is different, set the value 2 to the parameter `usiFanMode`.

Alarm/Alert	Problem	Solution
uiAlertID.0 TRUE	A controlled parameter has been changed, which requires a machine restart. The new configuration parameter is effective only after restart of the function block. List of the latched parameters: • usiFanMode • uiFanFreqMax • uiFanFreqMin • uiHysteresis • usiNbStage • usiNbFanStage1 • usiNbFanStage1 • usiNbFanStage3 • usiNbFanStage4 • usiPriorityStage1 • usiPriorityStage2 • usiPriorityStage3 • usiPriorityStage4	Restart the function block.
uiAlertID.1 TRUE	The value of the input uiFanCntrlSignal is not set within the specified range.	Verify that the parameter values are within their respective ranges.
uiAlertID.2 TRUE	The value of the parameter uiLowNoiseMaxFreq is not set within the specified range.	Verify that the parameter values are within their respective ranges.
uiAlertID.3 TRUE	The value of the parameter uiFanDelayIncr is not set within the specified range.	Verify that the parameter values are within their respective ranges.
uiAlertID.4 TRUE	The value of the parameter uiFanDelayDecr is not set within the specified range.	Verify that the parameter values are within their respective ranges.
uiAlertID.5 TRUE	xStage1Alarm input is active	Check the value of the input xStage1Alarm.
uiAlertID.6 TRUE	xStage2Alarm input is active	Check the value of the input xStage2Alarm.
uiAlertID.7 TRUE	xStage3Alarm input is active	Check the value of the input xStage3Alarm.
uiAlertID.8 TRUE	xStage4Alarm input is active	Check the value of the input xStage4Alarm.

Connectivity Diagram


FanMgmt
xEn
uiFanCntrlSignal
xStage1Alarm
xStage2Alarm
xStage3Alarm
xStage4Alarm
xLowNoiseOper
usiNbStage
usiNbFanStage1
usiNbFanStage2
usiNbFanStage3
usiNbFanStage4
usiFanMode
usiPriorityStage1
usiPriorityStage2
usiPriorityStage3
usiPriorityStage4
uiFanFreqMin
uiFanFreqMax
xMode
xManualStage1
xManualStage2
xManualStage3
xManualStage4
uiManualFreq
uiLowNoiseMaxFreq
uiFanDelayIncr
uiFanDelayDecr
uiMinOffTime
xOpHoursRst
uiHysteresis
usiState
xVSD
uiFanFreq
xCmdeStage1
xCmdeStage2
xCmdeStage3
xCmdeStage4
udiOpHoursSt1
udiOpHoursSt2
udiOpHoursSt3
udiOpHoursSt4
uiAlarmID
uiAlertID
E2_udiOpHrsSt1
E2_udiOpHrsSt2
E2_udiOpHrsSt3
E2_udiOpHrsSt4
ADR
ADR
ADR
ADR
ptrE2_OpHrsSt1
ptrE2_OpHrsSt2
ptrE2_OpHrsSt3
ptrE2_OpHrsSt4

NOTE:

- You have to define the parameters of the data type specified in the table Non-Volatile Memory Variables *(see page 281)* in the non-volatile memory parameters in the application.
- You have to use an `ADR` block to connect the input pin `ptrE2_OpHrsSt1...ptrE2_OpHrsSt4` to the variable defined in the non-volatile memory.
- If several instances of the function block are created inside the project, you have to create 4 new parameters (`E2_OpHrsSt1...E2_OpHrsSt4`) in the non-volatile memory for each function block.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Be sure that all variables are initialized to an appropriate value before their first use as array indices or pointers.
- Write programming instructions to test the validity of operands intended to be used as array indices and memory pointers.
- Do not attempt to access an array element outside the defined bounds of the array.

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

Chapter 20
Floating High Pressure Control: FloatingHighPresCntrl

What Is in This Chapter?

This chapter contains the following sections:

Section 20.1
Functional and Machine Overview

What Is in This Section?

This section contains the following topics:

Functional Overview

Functional Description

High variations of the refrigerant pressure can result in a high energy consumption of the condensing fans and compressors.

The `FloatingHighPresCntrl` (Floating High Pressure Control) function block controls the condensing pressure by modulating air volume through the condenser.

The air volume is controlled by the number and the speed of the condenser fans.

Why Use the `FloatingHighPresCntrl` Function Block?

The `FloatingHighPresCntrl` function block is used for the following purposes:

Purpose	Description
High pressure control	• maintain a constant high pressure • help avoid high pressure alarms • feature that helps to protect the compressor in case of a detected error

Features of the `FloatingHighPresCntrl` Function Block

The `FloatingHighPresCntrl` function block provides the following features:

- sets the floating condensing temperature based on outdoor air temperature
- supports standard refrigerant types R404A, R22, R410A, R407C, R134a, R407A, R290, R407F, R507A, R717, R723, R1234ze (*https://en.wikipedia.org/wiki/List_of_refrigerants*) and R744
- supports temperature units °C and °F
- supports pressure units Bar and PSI
- provides high pressure alarm monitoring
- handles sensor alarms
- supports PID control and deadband control
- supports initial VSD ramp

Error Avoidance Features

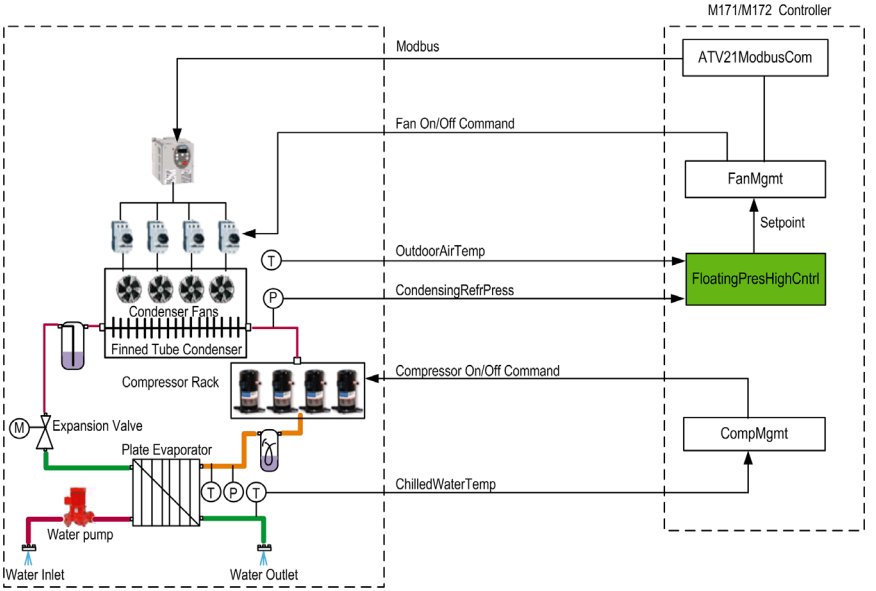
The `FloatingHighPresCntrl` function block provides the following error avoidance features to help you avoid the potentials of certain sources of machine malfunction:

Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.
Alarm/alert notification	If an invalid value is entered, an alarm or an alert is generated: ● Alarm: the compressors are switched off and the function block terminates in error. ● Alert: the function block keeps on operating, however with the possibility of reduced performance. In case of a high pressure alarm or when the high pressure sensor is disconnected or short-circuited, the fans will be started with maximum frequency.
Controlled parameter	Parameters like `uiUnitTypeTemp`, `xUnitTypePress` and `usiRefType` are controlled. The configuration of these parameters can be changed, however the changes are effective only after the restart of the function block.

Machine Overview

Machine View

The following picture presents the interaction between the function block and the machine:



`FloatingHighPresCntrl` This function block monitors the internal pressure of the refrigerant (using a high pressure sensor) and the external outdoor air temperature and operates the fan motors in association with the `FanMgmt` function block.

`FanMgmt` This function block controls the optimum number of fans and the frequency depending on the required air flow in the machine. `FanMgmt` manages the switch on/off of fans.

Section 20.2
Architecture

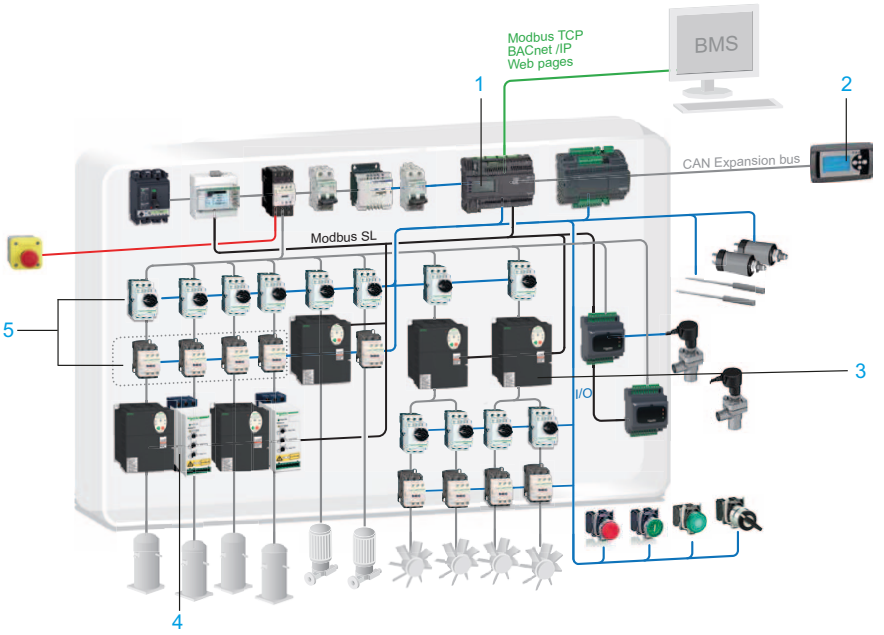
What Is in This Section?

This section contains the following topics:

Hardware Architecture

Hardware Architecture Overview

The following figure presents the hardware architecture for the `FloatingHighPresCntrl` function block associated with Air Cooled Chiller.

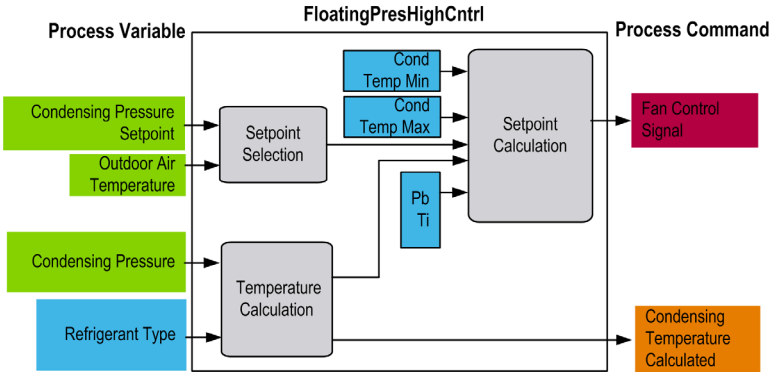


1 M171/M172 Controller
2 Graphic display
3 Variable speed drive ATV••/••• Modbus
4 Soft starters
5 Motor Control

Software Architecture

Function Block Diagram

The following function block diagram provides an overview of the software architecture:



The block diagram diagram presents on the left hand side the inputs, the **Process Variables**, on the right hand side the outputs, the **Process Commands**, as well as the function block `FloatingHighPresCntrl`.

The following table provides an overview of the functions of `FloatingHighPresCntrl`:

Function	Description
Setpoint Selection	Calculates and limits the setpoint between the input parameters `iCondTempMin` and `iCondTempMax` based on 3 modes: ● Fixed setpoint ● Floating setpoint with fixed offset ● Floating setpoint with variable offset
Temperature Calculation	Uses high refrigerant pressure measurement to calculate the condensing temperature `iCondTemp` on the basis of the refrigerant properties
Setpoint Calculation	Calculates control signal `uiFanCntrlSignal` for currently operating fan stages.

Section 20.3
Function Block Description

`FloatingHighPresCntrl` Function Block

Function Block Description

The `FloatingHighPresCntrl` function block calculates the control signal based on a PID used by the function block `FanMgmt` to control the high pressure refrigerant. The condensing refrigerant pressure setpoint can be calculated by means of 3 different algorithms. The condensing refrigerant temperature is calculated based on the refrigerant type and the measured refrigerant pressure.

The function block `FloatingHighPresCntrl` must be used with the following function blocks:

- `FanMgmt`
- `PIDAdvanced`

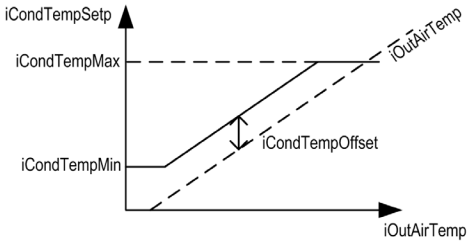
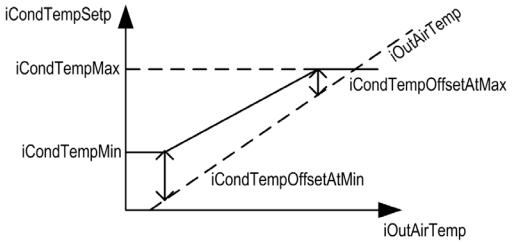
High Pressure Refrigerant Temperature Setpoint Calculation

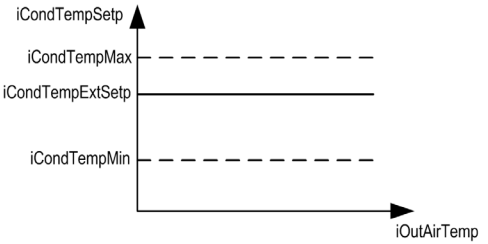
The `FloatingHighPresCntrl` function block calculates the high pressure refrigerant temperature setpoint by means of 3 algorithms:

- Floating
- Floating with variable offset
- Fixed

The algorithm type is selected by means of the parameter `usiControlMode`.

The following table provides an overview of the 3 setpoints:

Setpoint	Description
Floating	• `usiControlMode` = 0 • The condensing temperature setpoint `iCondTempSetp` is calculated on the basis of the actual outdoor air temperature `iOutAirTemp` and a constant offset `iCondTempOffset`. • The condensing temperature setpoint is limited between `iCondTempMin` and `iCondTempMax`. • `iCondTempSetp` = `iOutAirTemp` + `iCondTempOffset` 
Floating with variable offset	• `usiControlMode` = 1 • The condensing temperature setpoint `iCondTempSetp` is calculated on the basis of the outdoor air temperature `iOutAirTemp` and a variable offset. The variable offset is calculated with the parameters `iCondTempOffsetAtMin` which is related to `iCondTempMin` and `iCondTempOffsetAtMax` which is related to `iCondTempMax`. • The condensing temperature setpoint is limited between `iCondTempMin` and `iCondTempMax`. • `iCondTempSetp` = `iOutAirTemp` + variable offset 

Setpoint	Description
Fixed	• usiControlMode = 2 • The condensing temperature setpoint iCondTempSetp is constant and is provided by the parameter iCondTempExtSetp. • The condensing temperature setpoint is limited between iCondTempMin and iCondTempMax. • iCondTempSetp = iCondTempExtSetp 

Refrigerant List

The refrigerant is selected with the parameter usiRefType.

The table below gives the conversion precision from relative pressure to the temperature and the range for each refrigerant.

If the input for the pressure iCondPress is not set within the specified range, an alert is triggered and the conversion may not be accurate.

Refrigerant	usiRefType	Relative Pressure Range (Bar)	Maximum Calculation Error (°C)
R22	0	0.10...35.70	0.1
R134a	1	-0.50...+25.40	0.2
R404A	2	0.30...36.00	0.2
R407C	3	0.10...40.80	0.2
R410A	4	0.70...47.30	0.1
R407A	5	0.10...40.90	0.2
R407F	6	0.00...44.40	0.2
R290	7	0.00...31.00	0.1
R507A	8	-0.60...+35.90	0.2
R717	9	0.30...41.30	0.1
R723	10	-0.30...+42.50	0.1
R1234ze	11	-0.70...+19.50	0.1
R744	12	8.70...72.70	0.1

Unit Selection

You can select the unit type for the temperature and the pressure and the calculated values are returned in the selected unit type.

The unit type for the temperature is selected with the input `xUnitTypeTemp` and the unit type for the pressure is selected with the input `xUnitTypePress`.

If the input `xUnitTypeTemp` is set to...	Then ...
FALSE (°C),	the following inputs must be in °C: • `iCondTempExtSetp` • `iCondTempOffset` • `iCondTempOffsetAtMin` • `iCondTempOffsetAtMax` • `iHeatRecSetp`
	the following outputs are calculated in °C: • `iCondTemp` • `iCondTempSetp`
TRUE (°F),	the following inputs must be in °F: • `iCondTempExtSetp` • `iCondTempOffset` • `iCondTempOffsetAtMin` • `iCondTempOffsetAtMax` • `iHeatRecSetp`
	the following outputs are calculated in °F: • `iCondTemp` • `iCondTempSetp`

The unit type for the pressure is selected with the input `xUnitTypePress`.

If the input `xUnitTypePress` is set to...	Then ...
FALSE (Bar r),	the following inputs must be in Bar r: • `iCondPress` • `iCondPressMaxLimit`
TRUE (PSI)	the following inputs must be in PSI: • `iCondPress` • `iCondPressMaxLimit`

Heat Recovery Mode

If the input `xHeatRecovery` is set to TRUE, the Heat recovery mode is active. The condensing temperature setpoint `iCondTempSetp` is equal to the parameter `iHeatRecoverySetp`.

Status Management

An alarm informs you when condensing refrigerant pressure and outdoor air temperature sensor inputs are unavailable or short-circuited.

In case of an `xCondPressAlarm` or when the pressure `iCondPress` is greater than the pressure limit `iCondPressMaxLimit`, the output `uiFanCntrlSignal` is set to the maximum value 100.0%.

The output `uiAlarmID` indicates which alarm occurs.

The output `uiAlertID` indicates which alert occurs.

Section 20.4
Pin Description

What Is in This Section?

This section contains the following topics:

Input Pin Description

Pin Diagram

The following picture presents the pin diagram of `FloatingHighPresCntrl`:

Input Pin Description

Input	Data Type	Range	Scaling/Unit	Description
`xEn`	`BOOL`	TRUE or FALSE	N/A	TRUE: Enables the function block. FALSE: Disables the function block.
`iCondPress`	`INT`	-32768... +32767	0.01 Bar 0.1 PSI	Relative high pressure refrigerant

Input	Data Type	Range	Scaling/Unit	Description
iCondTempExtSetp	INT	iCondTempMin ...iCondTempMax	0.1 (°C/°F)	External setpoint for the condensing temperature in fixed mode Default: 400
iOutAirTemp	INT	-32768... +32767	0.1 (°C/°F)	Outdoor air temperature
xCondPressAlarm	BOOL	TRUE or FALSE	N/A	High pressure alarm
xHeatRecovery	BOOL	TRUE or FALSE	N/A	Heat recovery activation
iCondTempOffset	INT	0...200	0.1 (°C/°F)	Condensing temperature offset Default: 100
iCondTempOffsetAtMin	INT	0...200	0.1 (°C/°F)	Condensing temperature offset at minimum condensing temperature Default: 150
iCondTempOffsetAtMax	INT	0...200	0.1 (°C/°F)	Condensing temperature offset at maximum condensing temperature Default: 200
iCondTempMin	INT	-580...+3020	0.1 (°C/°F)	Minimum condensing temperature Default: 300
iCondTempMax	INT	-580...+3020	0.1 (°C/°F)	Maximum condensing temperature Default: 500
usiRefType	USINT	0...12	N/A	Refrigerant type: **0** = R22 **1** = R134A **2** = R404A **3** = R407C **4** = R410A **5** = R407A **6** = R407F **7** = R290 **8** = R507A **9** = R717 **10** = R723 **11** = R1234ze **12** = R744
uiKp	UINT	1...500	0.1 User unit	Gain Default: 60
uiTi	UINT	0...3600	0.1 s	Integration time
uiDeadband	UINT	0...500	0.1 User unit	Deadband for condensing temperature Default: 0

Input	Data Type	Range	Scaling/Unit	Description
usiControlMode	USINT	0...2	1	High pressure control type: **0** Floating **1** Floating with variable offset **2** Fixed
iCondPressMaxLimit	INT	0...14500	0.01 Bar 0.1 PSI	Condensing pressure maximum limit Default: 2000
xUnitTypeTemp	BOOL	TRUE or FALSE	N/A	Unit selection of the temperature TRUE = °F FALSE = °C
xUnitTypePress	BOOL	TRUE or FALSE	N/A	Unit selection of the relative pressure TRUE = PSI FALSE = Bar
iHeatRecoverySetp	INT	-32768... +32767	0.1 (°C/°F)	Heat recovery setpoint
uiRampSetpFilter	UINT	0...65535	(°C/F) / min	Ramp setpoint filter for switching from normal mode to heat recovery mode Default: 6

High Pressure Alarm indicates an alarm when the pressure exceeds a maximum limit.

⚠ DANGER

REFRIGERANT POISONING OR FREEZER BURNS

- Stop the compressor operation in case of a high pressure alarm.
- Interlock the high pressure alarm switch with the compressors using contactors in the electrical cabinet.

Failure to follow these instructions will result in death or serious injury.

Output Pin Description

Output Pin Description

Output Pin Description

Output	Data Type	Range	Scaling/Unit	Description
uiFanCntrlSignal	UINT	0...1000	0.1%	Control signal for the function block FanMgmt
iCondTemp	INT	-32768... +32767	0.1 (°C/°F)	Condensing temperature
iCondTempSetp	INT	iCondTempMin ... iCondTempMax	0.1 (°C/°F)	Condensing temperature setpoint
uiAlertID	UINT	0...65535	N/A	Alert identification
uiAlarmID	UINT	0...65535	N/A	Alarm identification

Alert ID Description

The uiAlertID output represents a value between 0 and 65535, whereby each bit represents an alert. The bits and their description are described in the following table:

Alert Bit	Alert Cause	Effect
0	Invalid parameter (iCondTempOffset < 0 or iCondTempOffset > 200) (iCondTempOffsetAtMin < 0 or iCondTempOffsetAtMin > 200) (iCondTempOffsetAtMax < 0 or iCondTempOffsetAtMax > 200) (iCondTempMin < -580 or iCondTempMin > 3020) (iCondTempMax < -580 or iCondTempMax > 3020) (iCondPressMaxLimit < 0 or iCondPressMaxLimit > 14500)	Function block operates at limited performance.
1	The input condensing pressure iCondPress is not set within the specified range of the refrigerant.	Function block operates at limited performance.
2	iCondTempExtSetp < iCondTempMin or iCondTempExtSetp > iCondTempMax	The value is limited.
3	Latched parameter • uiDeadband • uiRampSetpFilter • usiRefType • xUnitTypeTemp • xUnitTypePress	Change take effect after reset, the input xEn of the function must be set to FALSE.
4	uiTi is not set within the specified range.	The value is set to 0.
5	The condensing pressure iCondPress exceeds the maximum limit iCondPressMaxLimit.	The output signal uiFanCntrlSignal is set to 100.0%.

Alert Bit	Alert Cause	Effect
6	The input `xCondPressAlarm` is set to TRUE.	The output signal `uiFanCntrlSignal` is set to 100.0%.
7	Sensor alarm of the input `iCondPress`.	The output signal `uiFanCntrlSignal` is set to 100.0%.
8	Sensor alarm of the input `iOutAirTemp`.	The regulation switches to the mode with HP fixed (`usiControlMode` = 2)
9...15	Not used	N/A

Alarm ID Description

The `uiAlarmID` output represents a value between 0 and 65535, whereby each bit represents a detected alarm. The bits and their description are described in the following table:

Alarm Bit	Alarm Cause	Effect
0	Invalid parameter range: `iCondTempMin` > `iCondTempMax`	Function block is disabled.
1	The parameter `usiRefType` is not set within the specified range.	Function block is disabled.
2	The parameter `usiControlMode` is not set within the specified range.	Function block is disabled.
3	The parameter `uiKp` is not set within the specified range.	Function block is disabled.
4	The parameter `uiDeadband` is not set within the specified range.	Function block is disabled.
5...15	Not used	N/A

Section 20.5
Troubleshooting

Troubleshooting

Alarm/Alert	Problem	Solution
`uiAlertID.0 TRUE`	Invalid parameter values: `iCondTempOffset` < 0 or `iCondTempOffset` > 200 `iCondTempOffsetAtMin` < 0 or `iCondTempOffsetAtMin` > 200 `iCondTempOffsetAtMax` < 0 or `iCondTempOffsetAtMax` > 200 `iCondTempMin` < -580 or `iCondTempMin` > 3020 `iCondTempMax` < -580 or `iCondTempMax` > 3020 `iCondPressMaxLimit` < 0 or `iCondPressMaxLimit` > 14500	Verify that the parameter values are within their respective ranges.
`uiAlertID.1 TRUE`	Invalid parameter values of `iCondPress`.	Verify that the parameter values are within their respective ranges.
`uiAlertID.2 TRUE`	Invalid parameter values: `iCondTempExtSetp` < `iCondTempMin` or `iCondTempExtSetp` > `iCondTempMax`	Verify that the parameter values are within their respective ranges.
`uiAlertID.3 TRUE`	Latched parameter values are changed	Restart the function block, the input `xEn` must be set to FALSE.
`uiAlertID.4 TRUE`	Invalid parameter values of `uiTi`.	Verify that the parameter values are within their respective ranges.
`uiAlertID.5 TRUE`	Condensing pressure `iCondPress` exceeds the maximum limit `iCondPressMaxLimit`.	Check the value of `iCondPress` and `iCondPressMaxLimit`.
`uiAlertID.6 TRUE`	Condensing pressure alarm `xCondPressAlarm`.	Check the value of the input `xCondPressAlarm`.
`uiAlertID.7 TRUE`	Alarm due to the analog input channel used for condensing pressure refrigerant.	Verify whether the condensing pressure sensor is connected to the controller and working properly.

Alarm/Alert	Problem	Solution
`uiAlertID.8 TRUE`	Alarm due to the analog input channel used for outdoor air temperature.	Verify whether the outdoor temperature sensor is connected to the controller and working properly.
`uiAlarmID.0 TRUE`	Invalid parameter values: `iCondTempMin` > `iCondTempMax`	Verify that the parameter values are within their respective ranges.
`uiAlarmID.1 TRUE`	Invalid parameter values of `usiRefType`.	Verify that the parameter values are within their respective ranges.
`uiAlarmID.2 TRUE`	Invalid parameter values of `usiControlMode`.	Verify that the parameter values are within their respective ranges.
`uiAlarmID.3 TRUE`	Invalid parameter values of `uiKp`.	Verify that the parameter values are within their respective ranges.
`uiAlarmID.4 TRUE`	Invalid parameter values of `uiDeadband`.	Verify that the parameter values are within their respective ranges.

Chapter 21
Refrigerant Densitiy Calculation: Fluid_Density

What Is in This Chapter?

This chapter contains the following sections:

Section 21.1
Functional Overview

`Fluid_Density` Function Block Description

Function Block Description

The function block calculates the density of a refrigerant based on the

- refrigerant pressure,
- refrigerant temperature, and
- refrigerant type.

The refrigerant pressure assumes a relative pressure in Bar.

The refrigerant temperature assumes a Celsius degree.

The refrigerant density unit is in kg/m^3.

Why Use the `Fluid_Density` Function Block?

The `Fluid_Density` function block:

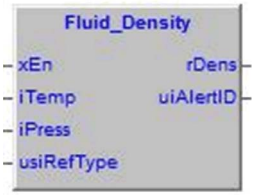
- is based on the conversion tables given by REFPROP 9.0 database.
- is used with refrigerant in vapor state.

Section 21.2
Pin Description

Pin Description

Pin Diagram

The following picture presents the pin diagram of the `Fluid_Density` function block:



Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
`xEn`	`BOOL`	TRUE or FALSE	N/A	Enable/disable the function block. TRUE = Enable
`iTemp`	`INT`	-32768...+32767	0.1 °C	Refrigerant temperature
`iPress`	`INT`	-32768...+32767	0.01 Bar	Refrigerant relative pressure
`usiRefType`	`USINT`	0...12	N/A	Refrigerant type: **0** = R22 **1** = R134a **2** = R404A **3** = R407C **4** = R410A **5** = R407A **6** = R407F **7** = R290 **8** = R507A **9** = R717 **10** = R723 **11** = R1234ze Default: 0

`iPress`

The refrigerant pressure assumes a relative pressure.

usiRefType, usiRefState

The table contains the relative pressure iPress range, the temperature iTemp range, and the maximum calculation error for each the type of refrigerant, in each state.

Refrigerant Type	Relative Pressure Range (Bar)	Temperature Range (°C)	Maximum Calculation Error (%)
R22	-0.90...+19.00	-40.0...+80.0	0.6
R134a	-0.50...+14.00	-40.0...+150.0	0.8
R404A	+0.00...+19.00	-40.0...+76.0	1.0
R407C	-50.0...+19.00	-40.0...+80.0	1.0
R410A	+0.00...+19.00	-40.0...+76.0	1.0
R407A	+0.00...+19.00	-40.0...+80.0	0.9
R407F	+0.00...+19.00	-39.0...+80.0	1.0
R290	+0.00...+14.00	-40.0...+80.0	0.5
R507A	+0.00...+19.00	-40.0...+80.0	1.0
R717	-0.50...+19.00	-40.0...+80.0	0.9
R723	-0.50...+19.00	-40.0...+80.0	0.8
R1234ze	-0.50...+9.00	-40.0...+80.0	0.3

Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
rDens	REAL	-3.40E+38 ... +3.40E+38	N/A / kg/m^3	Refrigerant density
uiAlertID	UINT	0...6	N/A	Alert ID

Alert ID Description

The output uiAlertID represents a value from 0 to 3, whereby each bit represents a detected alert. The table contains the bits and their description:

Alert Bit	Alert Cause	Effect
0	iTemp is not within the specified range.	No confidence on the maximum rDens error (%)
1	iPress is not within the specified range.	No confidence on the maximum rDens error (%)
2...15	Not used	N/A

Troubleshooting

Alert	Problem	Solution
`uiAlertID.0`	`iTemp` is not within the specified range.	Check the proper running of the sensor.
`uiAlertID.1`	`iPress` is not within the specified range.	Check the proper running of the sensor.

Chapter 22
Refrigerant Enthalpy Calculation: Fluid_Enthalpy

What Is in This Chapter?

This chapter contains the following sections:

Section 22.1
Functional Overview

`Fluid_Enthalpy` Function Block Description

Function Block Description

The function block calculates the enthalpy of a refrigerant based on the

- refrigerant pressure,
- refrigerant temperature,
- refrigerant type, and
- refrigerant state.

The refrigerant pressure assumes a relative pressure in Bar.

The refrigerant temperature assumes a Celsius degree.

The refrigerant enthalpy assumes a specific enthalpy in kJ/kg.

Why Use the `Fluid_Enthalpy` Function Block?

The `Fluid_Enthalpy` function block:

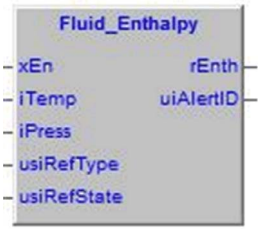
- is based on the conversion tables given by REFPROP 9.0 database.
- can be used with refrigerant in liquid state or vapor state.

Section 22.2
Pin Description

Pin Description

Pin Diagram

The following picture presents the pin diagram of the `Fluid_Enthalpy` function block:



Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
`xEn`	`BOOL`	TRUE or FALSE	N/A	Enable/disable the function block. TRUE = Enable
`iTemp`	`INT`	-32768...+32767	0.1 °C	Refrigerant temperature
`iPress`	`INT`	-32768...+32767	0.01 Bar	Refrigerant relative pressure
`usiRefType`	`USINT`	0...12	N/A	Refrigerant type: **0** = R22 **1** = R134a **2** = R404A **3** = R407C **4** = R410A **5** = R407A **6** = R407F **7** = R290 **8** = R507A **9** = R717 **10** = R723 **11** = R1234ze Default: 0

Input	Data Type	Range	Scaling / Unit	Description
usiRefState	USINT	1...2	N/A	Refrigerant state: **1** = Vapor **2** = Liquid Default: 1

iPress

The refrigerant pressure assumes a relative pressure.

usiRefType, usiRefState

The table contains the relative pressure iPress range,the temperature iTemp and the maximum calculation error for each the type of refrigerant, in each state.

Refrigerant Type	Refrigerant State	Refrigerant Pressure Range (Bar)	Temperature Range (°C)	Maximum Calculation Error (bar)
R22	Vapor	-0.90...+39.0	-40.0...+150.0	5.0
R22	Liquid	+4.00...+39.0	-40.0...+95.0	2.0
R134a	Vapor	-0.90...+29.0	-40.0...+150.0	4.7
R134a	Liquid	+0.00...+29.0	-40.0...+86.0	4.5
R404A	Vapor	-0.50...+34.0	-40.0...+150.0	3.8
R404A	Liquid	+4.00...+34.0	-40.0...+60.0	1.9
R407C	Vapor	-0.90...+34.0	-40.0...+150.0	5.0 (3 values out of range, under 6.9)
R407C	Liquid	+4.00...+34.0	-40.0...+77.0	3.9
R410A	Vapor	+0.00...+29.0	-40.0...+150.0	5.0 (2 values out of range, under 6.6)
R410A	Liquid	+4.00...+39.0	-40.0...+60.0	3.9
R407A	Vapor	-0.90...+34.0	-36.0...+130.0	5.0 (2 values out of range, under 6.7)
R407A	Liquid	+4.00...+39.0	-40.0...+75.0	4.9
R407F	Vapor	-0.50...+34.0	-40.0...+150.0	4.9
R407F	Liquid	+4.00...+39.0	-40.0...+73.0	5.0
R290	Vapor	-0.50...+24.0	-40.0...+120.0	4.9
R290	Liquid	+4.00...+34.0	-40.0...+78.0	4.7
R507A	Vapor	-0.50...+34.0	-40.0...+150.0	4.6
R507A	Liquid	+0.00...+29.0	-40.0...+62.0	4.2
R717	Vapor	-0.50...+39.0	-40.0...+115.0	5.0
R717	Liquid	+4.00...+39.0	-40.0...+78.0	4.7

Refrigerant Type	Refrigerant State	Refrigerant Pressure Range (Bar)	Temperature Range (°C)	Maximum Calculation Error (bar)
R723	Vapor	-0.50...+34.0	-40.0...+120.0	4.8
R723	Liquid	+0.00...+39.0	-40.0...+77.0	3.5
R1234ze	Vapor	-0.50...+19.0	-40.0...+150.0	3.0
R1234ze	Liquid	-0.50...+34.0	-40.0...+60.0	1.2

Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
`rEnth`	`REAL`	-3.40E+38 ... +3.40E+38	N/A / kJ/kg	Refrigerant enthalpy
`uiAlertID`	`UINT`	0...6	N/A	Alert ID

Alert ID Description

The output `uiAlertID` represents a value from 0 to 3, whereby each bit represents a detected alert. The table contains the bits and their description:

Alert Bit	Alert Cause	Effect
0	`iTemp` is not within the specified range.	No confidence on the maximum `rEnth` error (kJ/kg)
1	`iPress` is not within the specified range.	No confidence on the maximum `rEnth` error (kJ/kg)
2...15	Not used	N/A

Troubleshooting

Alert	Problem	Solution
`uiAlertID.0`	`iTemp` is not within the specified range.	Check the proper running of the sensor.
`uiAlertID.1`	`iPress` is not within the specified range.	Check the proper running of the sensor.

Chapter 23
Operating Hours: OperatingHours

What Is in This Chapter?

This chapter contains the following sections:

Section 23.1
Functional Overview

`OperatingHours` Function Block Description

Function Block Description

The function block calculates the operating hours of the device. It internally increment operating time every minute for which machine is running and display the time on hourly basis.

Why Use the `OperatingHours` Function Block?

The `OperatingHours` function block:

- simplifies programming.
- reduces engineering and commissioning time.

Features of the `OperatingHours` Function Block

The `OperatingHours` function block provides the following features:

- Calculating the operating hours of the device connected to its input.
- Retains the operating hour on power outage as well.

Section 23.2
Pin Description

Pin Description

Pin Diagram

The following picture presents the pin diagram of the `OperatingHours` function block:

Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
`xDeviceOn`	`BOOL`	TRUE or FALSE	N/A	Device on bit
`xReset`	`BOOL`	TRUE or FALSE	N/A	Operating hours are reset on rising edge of this input.
`ptrE2_OpHours`	`@UDINT`	N/A	N/A	Pointer pointing the variable defined in the non-volatile memory to retain the operating hours.

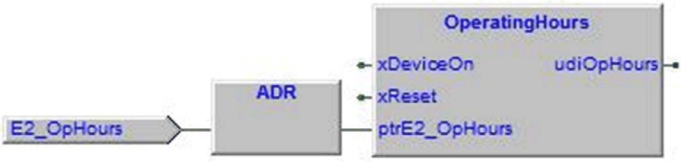
Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
`udiOpHours`	`UDINT`	0...4294967295	hours	Operating hours of the device

Non-Volatile Memory Variables

Address	Variable	Data Type	Range	Scaling / Unit	Description
N	`E2_OpHours`	`UDINT`	0...4294967295	hours	Total number of operating hours

Connectivity Diagram



NOTE:

- You have to define the parameter of the data type specified in the table *Non-Volatile Memory Variables* in the non-volatile memory parameters in the application.
- Use an `ADR` block to connect the input pin `ptrE2_OpHours` to the parameter defined in the non-volatile memory.
- If several instances of the function block are created inside the project, you have to create a new parameter in the non-volatile memory for each function block.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Be sure that all variables are initialized to an appropriate value before their first use as array indices or pointers.
- Write programming instructions to test the validity of operands intended to be used as array indices and memory pointers.
- Do not attempt to access an array element outside the defined bounds of the array.

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

Chapter 24
PID Control Function Block: PIDAdvanced

What Is in This Chapter?

This chapter contains the following sections:

Section 24.1
Functional Overview

PIDAdvanced Function Block Description

Function Block Description

The PIDAdvanced function block is designed for monitoring and controlling a wide variety of dependant processes. It includes the functions of dead band, manual mode, and hold.

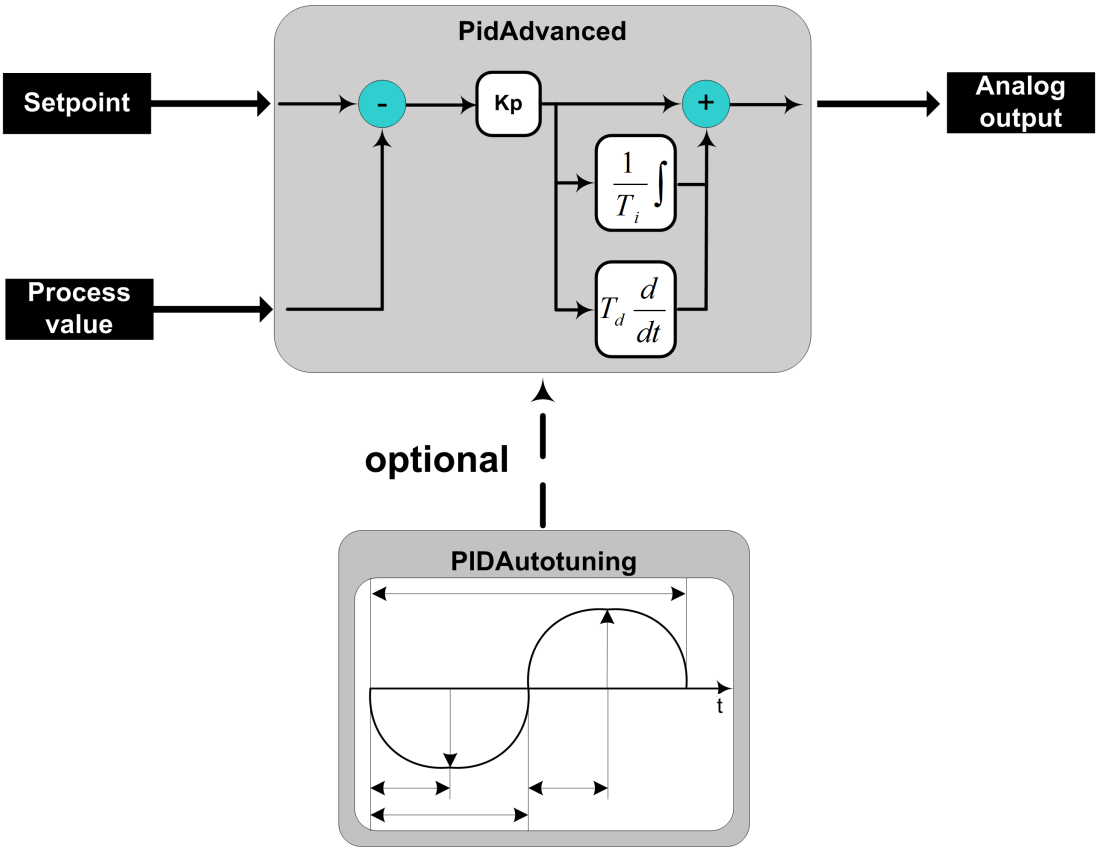
The PIDAdvanced function block can be used on various applications requiring regulation such as temperature control, pressure control, and flow control.

This function block incorporates a PID algorithm.

NOTE: The PIDAdvanced function block can be extended with the autotuning function *(see page 340)*.

Functional Overview of the PIDAdvanced

The following graphic presents a functional overview of the PIDAdvanced



Transfer Function

The PIDAdvanced function block works according to the following transfer function:

$$Y_k = Kp * \left\{ e_k + \frac{Ta}{2 * Ti} * \sum_{i=0}^{k} e_k + \frac{Tv}{Td} * \frac{1}{2} * Ta\left(e_k - e_{(k-2)}\right) \right\}$$

Why Use the PIDAdvanced Function Block?

The PIDAdvanced function block provides the following purposes:

- The PIDAdvanced function block can be used in heating or cooling mode (direct / reverse control).
- The embedded integral anti-windup function helps to prevent the PID controller from large integral corrective actions

Features of the PIDAdvanced Function Block

The PIDAdvanced function block provides the following features:

- Operating modes: automatic and manual
- Hold function: freezes the output value of the PID
- Integral anti-windup
- Dead band for more stable control
- Adjustable high or low limits
- Direct and reverse control

Operating Modes

The PIDAdvanced function block provides 2 operating modes:

Automatic mode	• Closed loop • In automatic mode, the control output is generated by PIDAdvanced
Manual mode	• Open loop • In manual mode, you enter the value of the control output.

Error Avoidance Features

The function block provides the following error avoidance feature to help avoid certain sources of machine malfunction:

Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.

Section 24.2
Pin Description

Pin Description

Pin Diagram

The following picture presents the pin diagram of `PIDAdvanced`:

Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
`xEn`	`BOOL`	TRUE or FALSE	N/A	Enable/Disable the function block. TRUE: Enable
`rPv`	`REAL`	-3.40E+38...+3.40E+38	N/A / User unit	Actual value from the process. **NOTE:** The process value `rPV` and the setpoint `rSetp` must have the same unit.

Input	Data Type	Range	Scaling / Unit	Description
rSetp	REAL	-3.40E+38...+3.40E+38	N/A / User unit	Value of setpoint
xManualMode	BOOL	TRUE or FALSE	N/A	TRUE: Enables manual mode FALSE: Disables manual mode Default value: FALSE
rManualValue	REAL	rLowLimit...rHighLimit	N/A / %	Manual value for the analog outputs iAnalog and rAnalog
xHold	BOOL	TRUE or FALSE	N/A	This pin is used to freeze the analog output (xHold=TRUE). The internal calculation of the integral term also freezes. ● TRUE: Holds the PID action ● FALSE: Resumes the PID action Default value: FALSE **NOTE:** If set to FALSE the calculation starts at the last frozen value off the analog outputs (rAnalog and iAnalog).
rHighLimit	REAL	rLowLimit to 100.0	N/A / %	High limit of PID output. Default value: 100.0
rLowLimit	REAL	0.0 to rHighLimit	N/A / %	Low limit of PID output. Default value: 0.0
rDeadband	REAL	0.0 to rHighLimit	N/A / User unit	Smoothes the control behavior. Default value: 0.0
xAutoTune	BOOL	TRUE or FALSE	N/A	FALSE: Not active TRUE: External autotuning function is active. Default value: FALSE
rAutoTuneValue	REAL	0.0...100.0	N/A / %	Value for rAnalogOut determined by the external autotuning function. Default value: 0.0
rKp	REAL	-300.0...+300.0	N/A	Value of proportional gain configured by you.

Input	Data Type	Range	Scaling / Unit	Description
uiTi	UINT	0..6000	0.1 / s	Value of the integral time configured by you or autotuning. Default value: 0
uiTd	UINT	0..6000	0.1 / s	Value of derivative damping time configured by you. Default value: 0
xReset	BOOL	TRUE or FALSE	N/A	FALSE: not active TRUE: Alarms are reset by a rising edge. **NOTE:** This pin is used to reset only the alarms. To reset the alarms, change the parameter which caused the alarm and give a rising edge on pin xReset. **NOTE:** Alerts are reset automatically.

xManualMode, rManualValue

If...	Then...
xManualMode is set to TRUE	the manual mode is active and the outputs rAnalog and iAnalog take the value of rManualValue.
xManualMode is set to FALSE	the calculation starts at the last entered manual value rManualValue.

rLowLimit, rHighLimit

The values at these pins define the range of PIDAdvanced outputs (rAnalog and iAnalog). These values are configured as per the analog output module specification.

Example:

rHighLimit = 100.0 (%) rLowLimit = 0.0 (%)	The PIDAdvanced controls the analog output between 0.0 and 100.0 %.
rHighLimit = 50.0 (%) rLowLimit = 10.0 (%)	The PIDAdvanced controls the analog output between 10.0 and 50.0 %.

rDeadband

The value at this pin defines the limit of the dead band for a detected error. This dead band function is used to make the `PIDAdvanced` output to settle down.

Example:

`rSetp` = 45.0 °C

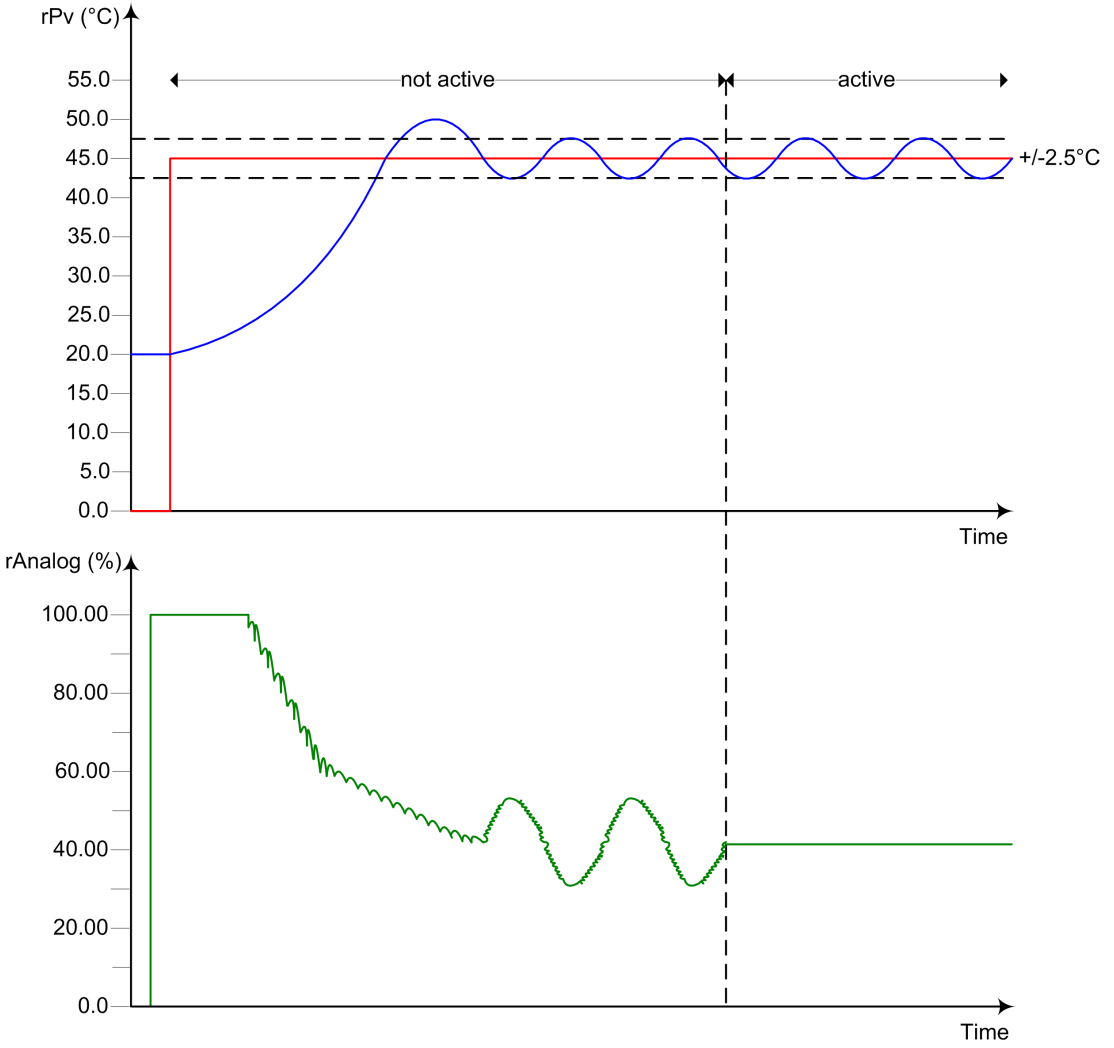
`rPv` = 0…60.0 °C

`rHighLimit` = 100.0 (%)

`rLowLimit` = 0.0 (%)

`rDeadband` = 2.5 °C (+/-)

The following graphic presents the dead band function:



Proportional Gain `rKp`

Example:

`rKp` > 0	Direct mode, for example, control for heating.
`rKp` = 0	The outputs `rAnalog` and `iAnalog` are set to 0.

rKp < 0	Reverse mode, for example, control for cooling (inverse control).

Integral Time uiTi

Example:

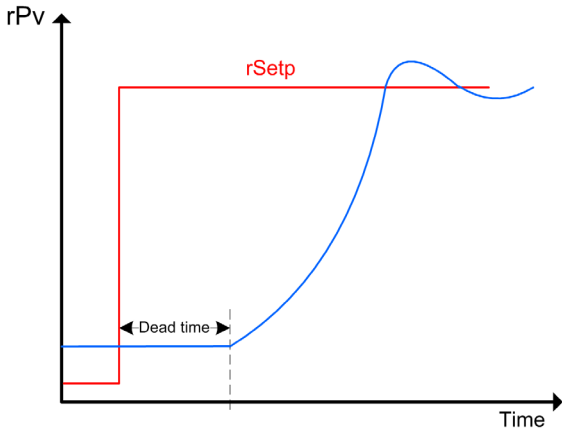
uiTi = 1	Fast integration time, causes a fast influence on the outputs rAnalog and iAnalog.
uiTi = 10	10 times slower than the fast integration time (a) and causes a slower influence on the outputs rAnalog and iAnalog.
uiTi = 0	uiTi is deactivated.

Derivate Time uiTd

Example:

uiTd = 1	The smallest damping, causes a high influence to the outputs rAnalog and iAnalog.
uiTd = 10	1/10 of the smallest damping, causes a lower influence on the outputs rAnalog and iAnalog.
uiTd = 0	uiTd is deactivated.

NOTE: In systems with dead time, uiTd should be set to 0. The value of uiTd must be greater than the cycle time. If it is less than the cycle time, then the uiTd value is overwritten with the value of the cycle time.



Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
rAnalog	REAL	rLowLimit to rHighLimit	N/A / %	Analog output
iAnalog	INT	0...1000	0.1 / %	Analog output
rError	REAL	-3.40E+38...+3.40E+38	N/A	Actual detected error between rSp and rPv.
uiAlarmID	UINT	0...65535	N/A	Alarm identification
uiAlertID	UINT	0...65535	N/A	Alert identification

Alarm ID Description

The output uiAlarmID represents a value from 0 to 15 whereby each bit represents a detected alarm. The table contains the bits and their description:

Alarm Bit	Alarm Cause	Effect
0	Application cycle time is greater than 2000 ms	The outputs rAnalog and iAnalog are set to 0.
1	Invalid value for the parameters (rHighLimit, rLowLimit)	
2	Invalid value of the parameter rDeadband	
3	Invalid values of the parameters rKp, uiTi or uiTd	
4-15	not used	N/A

Alert ID Description

The output uiAlertID represents a value from 0 to 15 whereby each bit represents a detected alert. The table contains the bits and their description:

Alert Bit	Alert Cause	Effect
0	Invalid value of the input rManualValue	rAnalog and iAnalog outputs are set to rHighLimit or rLowLimit
1	rLowlimit is equal to rHighLimit	rAnalog and iAnalog outputs are set to rLowLimit
2	Autotuning is active, the input xAutoTune is set to TRUE.	rAnalog and iAnalog outputs are set to rAutoTuneValue
3	rKp is set to 0.0	rAnalog and iAnalog outputs are set to 0.
4-15	not used	N/A

Troubleshooting

Alarm/Alert	Problem	Solution
uiAlarmID.0	Application cycle time is greater than 2000 ms.	Reduce the application and size.
uiAlarmID.1	rLowLimit > rHighLimit or rLowLimit > 100.0 or rLowLimit < 0.0 or rHighLimit > 100.0 or rHighLimit < 0.0	Verify that the parameter values are within their respective ranges.
uiAlarmID.2	rDeadBand < 0.0 rDeadBand > rHighLimit	Verify that the parameter values are within their respective ranges.
uiAlarmID.3	rKp < -300.0 or rKp > +300.0 or uiTi < 0 or uiTi > 6000 or uiTd < 0 or uiTd > 6000	Verify that the parameter values are within their respective ranges.
uiAlertID.0	rManualValue > rHighLimit rManualValue < rLowLimit	Verify that the parameter values are within their respective ranges.
uiAlertID.1	rLowLimit is equal to rHighLimit	Verify that the parameter values are within their respective ranges.
uiAlertID.2	Autotuning is active, the input xAutoTune is set to TRUE.	Wait until autotuning is completed.
uiAlertID.3	rKp is set to 0.0	Verify that the parameter values are within their respective ranges.

Chapter 25
PID Autotuning: PIDAutoTuning

What Is in This Chapter?

This chapter contains the following sections:

Section 25.1
Functional Overview

PIDAutoTuning Function Block Description

Function Block Description

The PIDAutoTuning function block measures the dynamic response of a control system and calculates automatically the parameters uiTi (integral time in seconds) and rKp (proportional gain) for that control system. These parameters can be connected to the parameter inputs of a PID AFB (for example PIDAdvanced).

NOTE: This function block must be used together with the PIDAdvanced.

When autotuning is enabled, it induces oscillations in the process around the set-point. After the completion of 3 oscillations, autotuning calculates a set of PI parameters.

NOTE: During the autotuning process the system is set to the minimum and maximum operating limits to measure the process response time. Ensure that the minimum and maximum operating limits are set correctly.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Only qualified persons with the skills and knowledge of electrical control systems, and having the related safety training, are allowed to program, install, alter, and otherwise apply this product.
- Understand and follow all local, regional, and national safety codes and standards.

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

Why Use the PIDAutoTuning Function Block?

The PIDAutoTuning function block is used to calculate the rKp and uiTi parameters for the control system.

Features of the PIDAutoTuning Function Block

The PIDAutoTuning function block provides the following features:

- Manual / automatic PI calculation
- Calculates 3 series of rKp and uiTi: fast, medium, and slow control
- Direct / reverse control

Required Actions Before Autotuning

Proceed as follows before starting autotuning:

1. Ensure that the process value and the setpoint have the same unit and precision.
2. Ensure that the process value `rPv` is not a fluctuating signal.
   If the process value `rPv` is fluctuating, the signal must be filtered.
3. Verify whether your system is direct (for example, heating) or inverse (for example, cooling).
   If your system is inverse, set `xInversCntrl` to TRUE.
4. Check the plausibility of the process value `rPv`.
   The measured process value must be plausible outside of the `PIDAutoTuning` function block.
5. Check the set-point `rSp` limits.
   Outside of the `PIDAutoTuning` function block, you have to ensure, that the set-point limit is checked by your program.

⚠ WARNING
UNINTENDED EQUIPMENT OPERATION
Ensure that your program respects the setpoint `rSp` limits
Failure to follow these instructions can result in death, serious injury, or equipment damage.

   If the limits are exceeded, you have to ensure that this is not a potential hazard to personal health or your system.
6. Enable the `PIDAutoTuning` function block.

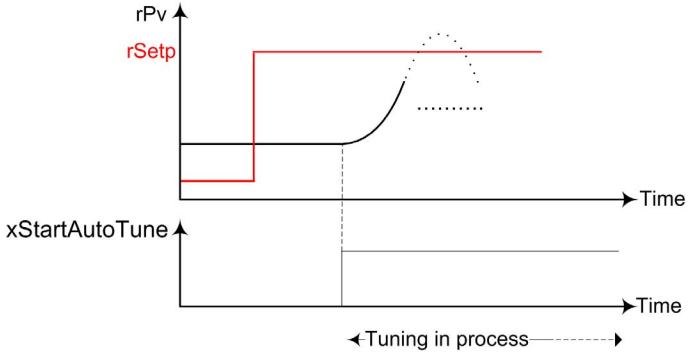
NOTE: Due to the alarms of the periphery (for example, emergency stop) you have to ensure that the `PIDAutoTuning` function block is disabled and `xStartAutotune` is set to FALSE.
A fluctuating process value gives an incorrect calculation.

Start Autotuning

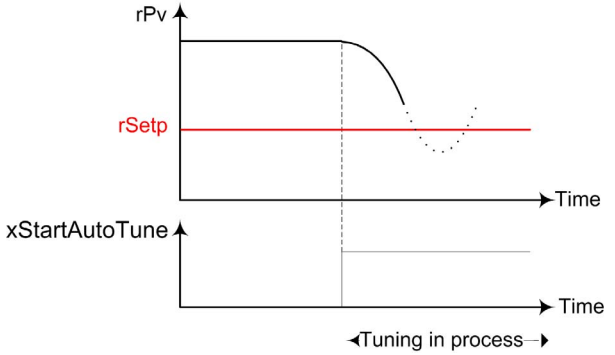
Proceed as follows to start autotuning:

1. Set the set-point `rSetP`.
2. Set `xStartAutoTune` to TRUE.
3. Wait until the autotuning is completed.

Autotuning procedure in direct mode (`xInversCntrl` = 0):



Autotuning procedure in reverse mode (`xInversCntrl` = 1):



NOTE: Autotuning is finished when `xStartAutoTune` is set from TRUE to FALSE.

Error Avoidance Features

The function block provides the following error avoidance feature to help avoid certain sources of machine malfunction:

Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.

Section 25.2
Pin Description

Pin Description

Pin Diagram

The following picture presents the pin diagram of `PIDAutoTuning`:

Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
`xEn`	`BOOL`	TRUE or FALSE	N/A	Enables the function block.
`xStartAutoTune`	`BOOL`	TRUE or FALSE	N/A	A rising edge activates auto- tuning TRUE: the autotuning process starts on a rising edge. FALSE: autotuning is disabled
`xInversCntrl`	`BOOL`	TRUE or FALSE	N/A	Inverses the control. TRUE: Active (for example, for cooling systems) FALSE: Not active (for example, for heating systems)

Input	Data Type	Range	Scaling / Unit	Description
usiMode	USINT	0...3	N/A	Changes the PI parameters of the outputs rKp and uiTi. 0= Manual parameter 1= ATR Para (slow) 2= ATR Para (medium) 3= ATR Para (fast) **NOTE:** If you have not performed autotuning before, set usiMode to 0 (manual parameters).
rPv	REAL	-3.40E+38 ...+3.40E+38	N/A / User unit	Process value
rSetp	REAL	-3.40E+38 ...+3.40E+38	N/A / User unit	Set-point **NOTE:** rSetp is the value of the set-point and is the intended working process value for the PIDAdvanced AFB. **NOTE:** Reset when not active.
rManualKp	REAL	-300.0...+300.0	N/A	Manual value of proportional gain configured by you. **NOTE:** The value is active at rKp if usiMode is set to 0.
uiManualTi	UINT	0...6000	0.1 / s	Manual value of integral time configured by you. **NOTE:** The value is active at uiTi if usiMode is set to 0.
rHighLimit	REAL	0.0...100.0	N/A / %	High limit of PID output. **NOTE:** rHighLimit of the PIDAdvanced AFB must have the same value.
rLowLimit	REAL	0.0...100.0	N/A / %	Low limit of PID output. **NOTE:** rLowLimit of the PIDAdvanced AFB must have the same value.

rSetp, rPv

NOTE: A fluctuating signal causes an incorrect calculation of the PI parameters.

NOTE: The set-point rSetp and the process value rPv must be of the same unit.

xInversCntrl

If xInversCntrl is set to TRUE, the input xInversCntrl inverts the calculation.

Example:

If...	Then...
the output rAutoTuneValue is 100.0% and rPv increases	set xInversCntrl to FALSE.
the output rAutoTuneValue is 100.0% and rPv decreases	set xInversCntrl to TRUE.

rLowLimit, rHighLimit

- These parameters define the range of your process (rAutoTuneValue).
- The PID controller (for example, PIDAdvanced AFB) must have the same values. The maximum range is 0.0 to 100.0%.

NOTE: In oversized or undersized systems it is necessary to limit the range of the output rAutoTuneValue (for example, 0.0 to 50.0% for an oversized system).

Results:
- The PIDAutoTuning AFB controls the output rAutoTuneValue between 0.00 and 50.00 %.
- The process value increases slower (for example temperature).
- The calculated PI parameters work better for an oversized system.

usiMode

0 = manual	Manual PI- parameters are set to the outputs rKp and uiTi
1 = slow	Calculated PI- parameters are set to the outputs rKp and uiTi.
2 = medium	Calculated PI- parameters are set to the outputs rKp and uiTi
3 = fast	Calculated PI- parameters are set to the outputs rKp and uiTi.

Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
xAutoTune	BOOL	TRUE or FALSE	N/A / %	TRUE: Autotuning process is active. FALSE: Autotuning process is not active. **NOTE:** If xAutoTune is set from TRUE to FALSE (falling edge), autotuning is finished. **NOTE:** It is possible to connect this output directly to the input xAutoTune of the PIDAdvanced AFB.
rAutoTuneValue	REAL	rLowLim to rHighLim	N/A / %	Analog output **NOTE:** It is possible to connect this output directly to the input rAutoTuneValue of the PIDAdvanced AFB.
rKp	REAL	-300.0... +300.0	N/A	Manual or calculated rKp **NOTE:** It is possible to connect this output directly to the input rKp of the PIDAdvanced AFB.
uiTi	UINT	0...6000	0.1/ s	Manual or calculated uiTi **NOTE:** It is possible to connect this output directly to the input uiTi of the PIDAdvanced AFB.
uiMessageID	UINT	0...65535	N/A	Message identification
uiAlarmID	UINT	0...65535	N/A	Alarm identification

uiMessageID

The output `uiMessageID` represents a value from 0 to 15, whereby each bit represents a message. The table contains the bits and their description:

Message Bit	Message Cause	Effect
0	Tuning is in progress.	Autotuning is active.
1	Tuning is completed. System OK	Autotuning is finished and the controlling was successful.
2	Tuning is completed. System is oversized.	Autotuning is finished and the calculated `PI` parameters indicates an oversized system. The calculated parameters or manual values can be used. **NOTE:** The range of the output `rAutoTuneValue` can be limited with `rHighLimit`.
3	Tuning is completed. System is undersized.	Autotuning is finished and the calculated `PI` parameters indicates an undersized system. The calculated parameters or manual values can be used.
4-15	not used	N/A

Alarm ID Description

The output `uiAlarmID` represents a value from 0 to 63, whereby each bit represents a detected alarm. The table contains the bits and their description:

Alarm Bit	Alarm Cause	Effect
0	Invalid cycle time.	`rAutoTuneValue` is set to 0 and autotuning will not start or will be canceled.
1	Invalid value of the parameters `rHighLimit` or `rLowLimit`	
2	Invalid value of the parameters `rManualKp` or `uiManualTi`	
3	Invalid value of the parameter `usiMode`)	
4	`xInversCntrl` is changed during autotuning is running.	
5	The `PI` parameters cannot be calculated for this system.	• Autotuning is finished and the controlling detected an incalculable system. • `rAutoTuneValue` is set to 0 and autotuning will not start or will be canceled. • The previously calculated `PI` parameters (`rKp`, `uiTi`) will be deleted.
6-15	Not used	N/A

Troubleshooting

Alarm	Problem	Solution
`uiAlarmID.0` TRUE	Cycle time > 2000 msec	The application is too large. Reduce the application and check the cycle time.

`uiAlarmID.1` TRUE	`rHighLimit` > `rLowLimit`
	`rHighLimit` > 100.0%
	`rLowLimit` = `rHighLimit`
		Verify that the parameter values are within their respective ranges.

`uiAlarmID.2` TRUE	`rManualKp` < -300.0 `rManualKp` > +300.0 `uiManualTi` < 0 `uiManualTi` > 6000	Verify that the parameter values are within their respective ranges.
`uiAlarmID.3` TRUE	Invalid value of the parameter `usiMode`	Verify that the parameter values are within their respective ranges.

Alarm	Problem	Solution
`uiAlarmID.4` TRUE	`xInversCntrl` changes during auto- tuning	Do not change the value as long as autotuning is in progress.
`uiAlarmID.5` TRUE	• Auto- tuning run for a long time (>45 minutes). • The system is slow.	• Stop autotuning. • Set the `PI` parameters manually.
`uiAlarmID.5` TRUE	• Internal variables detected values which are not calculable (for example `Time1`<=1 second or `Time4`>2700 seconds). • There is a fluctuating signal of the actual process value (but you cannot see it in the program). • The message **Tuning completed system is incalculable** will be presented.	If the parameters do not work, set the manual `PI` parameters (`usiMode` = 0).

Chapter 26
Totalizer for Digital Input Pulses: Pulse2Counter

What Is in This Chapter?

This chapter contains the following sections:

Section 26.1
Functional Overview

Pulse2Counter Function Block Description

Function Block Description

The function block Pulse2Counter counts the pulses provided by an energy meter. With the Counter2Energy *(see page 216)* the counted values are converted to active power and energy.

NOTE: The function block Pulse2Counter must be used on controllers that do not provide fast counter inputs.

Why Use the Pulse2Counter Function Block?

The Pulse2Counter function block:

- simplifies programming.
- reduces engineering and commissioning time.

Features of the Pulse2Counter Function Block

The Pulse2Counter provides the following features:

- accumulates the number of pulses of a digital input.
- is used to process meter information provided by pulses.
- provides an input to reset the counter value and indicates the date of the last reset.

Energy Meter

As part of the energy management, energy data from the energy meter can be read by pulse communication using the AFBs Pulse2Counter and Counter2Energy *(see page 216)*.

Error Avoidance Features

The function block provides the following error avoidance feature to help avoid certain sources of machine malfunction:

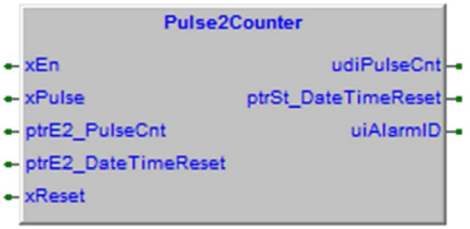
Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.

Section 26.2
Pin Description

Pin Description

Pin Diagram

The following picture presents the pin diagram of the `Pulse2Counter` function block:



Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
`xEn`	`BOOL`	TRUE or FALSE	N/A	Enables function block/ counting
`xPulse`	`BOOL`	TRUE or FALSE	N/A	Input pulses from energy meter
`ptrE2_PulseCnt`	`@UDINT`	N/A	N/A	Pulse count pointer for non-volatile memory variable.
`ptrE2_DateTimeReset`	`@USINT`	N/A	N/A	Date Time Reset pointer for non-volatile memory variable.
`xReset`	`BOOL`	TRUE or FALSE	N/A	Resets the accumulated pulse value. **NOTE:** The rising edge of the input `xReset` resets the internal values to zero.

Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
`udiPulseCnt`	`UDINT`	0…4294967295	N/A	Pulse count
`ptrSt_DateTimeReset`	`@USINT`	N/A	N/A	Pointer for Last Reset Date Time variable.

Output	Data Type	Range	Scaling / Unit	Description
uiAlarmID	UINT	0...65535	N/A	Alarm identification

Non-Volatile Memory Variables

Address	Variable	Data Type	Range	Scaling / Unit	Description
N	E2_udiPulseCnt	UDINT	0…4294967295	N/A	Pulse count
N+2	E2_usiDateTime ResetMinutes	USINT	0…59	Minutes	Minutes when Reset is activated.
N+3	E2_usiDateTime ResetHours	USINT	0…23	Hours	Hours when Reset is activated.
N+4	E2_usiDateTime ResetMonthDay	USINT	1…31	N/A	Day of the month when Reset is activated.
N+5	E2_usiDateTime ResetMonth	USINT	1…12	N/A	Month when Reset is activated.
N+6	E2_usiDateTime ResetYear	USINT	00…99	Year	Year when Reset is activated.

Alarm ID Description

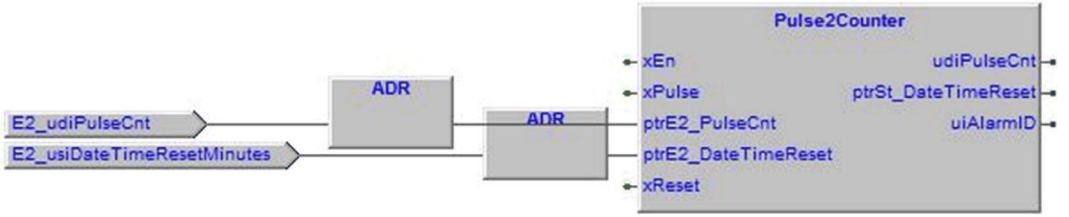
The uiAlarmID output represents a value from 0 to 3, whereby each bit represents a detected alarm. The table contains the bits and their description:

Alarm Bit	Alarm Cause	Effect
0	The pulse duration is too short.	A reliable pulse count is not possible. **NOTE:** This alarm is generated when the frequency of the input is higher than 10 Hz or the time between 2 ON states of the pulse input is less than 100 ms.
1	Overflow alarm	The pulse count register has reached the maximum limit value. The pulse count is not continued. udiPulseCnt value is retained.
2-15	Not used	N/A

Troubleshooting

Alarm	Problem	Solution
`uiAlarmID.0`	The pulse counting has discontinued, `udiPulseCnt` value is retained.	Verify and increase the pulse width provided by an energy meter.
`uiAlarmID.1`	Overflow alarm	Reset the function block by a rising edge on the `xReset` input.

Connectivity Diagram



NOTE:

- You have to define a parameter in the non-volatile memory parameters in the application.
- You have to use an `ADR` block to connect the input pin `ptrE2_PulseCnt` and `ptrE2_DateTimeReset` to the variable defined in the non-volatile memory.
- If several instances of the function block are created inside the project, you have to create 6 new parameters in the non-volatile memory for each function block:
  - `E2_udiPulseCnt`
  - `E2_usiDateTimeResetMinutes`
  - `E2_usiDateTimeResetHours`
  - `E2_usiDateTimeResetMonthDay`
  - `E2_usiDateTimeResetMonth`
  - `E2_usiDateTimeResetYear`

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Be sure that all variables are initialized to an appropriate value before their first use as array indices or pointers.
- Write programming instructions to test the validity of operands intended to be used as array indices and memory pointers.
- Do not attempt to access an array element outside the defined bounds of the array.

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

Chapter 27
Pressure to Temperature: Press2Temp

What Is in This Chapter?

This chapter contains the following sections:

Section 27.1
Functional Overview

`Press2Temp` Function Block Description

Function Block Description

The function block calculates the temperature of a refrigerant based on the

- refrigerant pressure,
- refrigerant type, and
- refrigerant state.

The refrigerant pressure assumes a relative pressure in Bar.

The refrigerant temperature assumes a Celsius degree.

For fluids with glide, the function block allows to calculate the liquid phase and the vapor phase and also the average value.

Why Use the `Press2Temp` Function Block?

The `Press2Temp` function block:

- is based on the conversion tables given by REFPROP 9.0 database.
- can be used with refrigerant in liquid state, vapor state or mixed state.

Section 27.2
Pin Description

Pin Description

Pin Diagram

The following picture presents the pin diagram of the `Press2Temp` function block:

Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
`xEn`	`BOOL`	TRUE or FALSE	N/A	Enable/disable the function block. TRUE = Enable
`iPress`	`INT`	-32768...+32767	0.01 / Bar	Refrigerant relative pressure
`usiRefType`	`USINT`	0...12	N/A	Refrigerant type: **0** = R22 **1** = R134a **2** = R404A **3** = R407C **4** = R410A **5** = R407A **6** = R407F **7** = R290 **8** = R507A **9** = R717 **10** = R723 **11** = R1234ze **12** = R744 Default: 0

Input	Data Type	Range	Scaling / Unit	Description
usiRefState	USINT	0...2	N/A	Refrigerant state: **0** = Vapor (50%) + Liquid (50%) **1** = Vapor **2** = Liquid Default: 0

iPress

The refrigerant pressure assumes a relative pressure.

usiRefType

The table contains the relative pressure iPress range and the maximum calculation error for each the type of refrigerant.

Refrigerant Type	Relative Pressure Range (Bar)	Maximum Calculation Error (°C)
R22	+0.10...+35.70	0.1
R134a	-50.0...+25.40	0.2
R404A	+0.30...+36.0	0.2
R407C	+0.10...+40.8	0.2
R410A	+0.70...+47.30	0.1
R407A	+0.10...+40.90	0.2
R407F	+0.00...+44.40	0.2
R290	+0.00...+31.0	0.1
R507A	-0.60...+35.90	0.2
R717	+0.30...+41.30	0.1
R723	-0.30...+ 42.50	0.1
R1234ze	-0.70...+19.50	0.1
R744	+8.70...+72.70	0.1

Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
iTemp	INT	-32768...+32767	0.1 / °C	Refrigerant temperature
uiAlertID	UINT	0...6	N/A	Alert identification

Alert ID Description

The output `uiAlertID` represents a value from 0 to 1, whereby each bit represents a detected alert. The table contains the bits and their description:

Alert Bit	Alert Cause	Effect
0	`iPress` is not within the specified range.	No confidence on the maximum `iTemp` error (°C)
1...15	Not used	N/A

Troubleshooting

Alert	Problem	Solution
`uiAlertID.0`	`iPress` is not within the specified range.	Check the proper running of the sensor.

Chapter 28
Temperature to Pressure: Temp2Press

What Is in This Chapter?

This chapter contains the following sections:

Section 28.1
Functional Overview

`Temp2Press` Function Block Description

Function Block Description

The function block calculates the pressure of a refrigerant based on the

- refrigerant temperature,
- refrigerant type, and
- refrigerant state.

The refrigerant pressure assumes a relative pressure in Bar.

The refrigerant temperature assumes a Celsius degree.

For fluids with glide, the function block allows to calculate the

- liquid phase,
- vapor phase, and
- average value.

Why Use the `Temp2Press` Function Block?

The `Temp2Press` function block:

- is based on the conversion tables given by REFPROP 9.0 database.
- can be used with refrigerant in liquid state, vapor state or mixed state.

Section 28.2
Pin Description

Pin Description

Pin Diagram

The following picture presents the pin diagram of the `Temp2Press` function block:

Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
`xEn`	`BOOL`	TRUE or FALSE	N/A	Enable/disable the function block. TRUE = Enable
`iTemp`	`INT`	-32768...+32767	0.1 / °C	Refrigerant temperature
`usiRefType`	`USINT`	0...12	N/A	Refrigerant type: **0** = R22 **1** = R134a **2** = R404A **3** = R407C **4** = R410A **5** = R407A **6** = R407F **7** = R290 **8** = R507A **9** = R717 **10** = R723 **11** = R1234ze **12** = R744 Default: 0

Input	Data Type	Range	Scaling / Unit	Description
usiRefState	USINT	0...2	N/A	Refrigerant state: **0** = Vapor (50%) + Liquid (50%) **1** = Vapor **2** = Liquid Default: 0

usiRefType

The table contains the relative pressure iTemp range and the maximum calculation error for each the type of refrigerant.

Refrigerant Type	Temperature Range (°C)	Maximum Calculation Error (bar)
R22	-40.0...+80.0	0.04
R134a	-40.0...+80.0	0.02
R404A	-40.0...+60.0	0.04
R407C	-40.0...+77.5	0.04
R410A	-40.0...+69.5	0.04
R407A	-40.0...+76.0	0.04
R407F	-40.0...+75.0	0.04
R290	-40.0...+80.0	0.01
R507A	-39.0...+68.5	0.03
R717	-40.0...+80.0	0.01
R723	-40.0...+80.0	0.01
R1234ze	-40.0...+80.0	0.01
R744	-40.0...+29.5	0.03

Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
iPress	INT	-32768...+32767	0.01 / bar	Refrigerant relative pressure
uiAlertID	UINT	0...6	N/A	Alert ID

iPress

The refrigerant pressure assumes a relative pressure.

Alert ID Description

The output `uiAlertID` represents a value from 0 to 65535, whereby each bit represents a detected alert. The table contains the bits and their description:

Alert Bit	Alert Cause	Effect
0	`iTemp` is not within the specified range.	No confidence on the maximum `iPress` error (°C)
1...15	N/A	N/A

Troubleshooting

Alert	Problem	Solution
`uiAlertID.0`	`iTemp` is not within the specified range.	Check the proper running of the sensor.

Chapter 29
Psychrometric Calculation: Psychrometric

What Is in This Chapter?

This chapter contains the following sections:

Section 29.1
Functional Overview

`Psychrometric` Function Block Description

Function Block Description

The function block calculates the
- absolute humidity,
- enthalpy,
- dew point temperature, and
- the specific volume.

The calculation is based on the
- dry bulb temperature,
- relative humidity, and
- as optional, the atmospheric pressure.

These are common used functions in air conditioning units for humidification control, economizer control.

Why Use the `Psychrometric` Function Block?

The `Psychrometric` function block allows calculating the enthalpy, and by difference of enthalpy to calculate each capacity exchanged, when the flow is known.

The specific volume allows converting the volumetric flow (from fan characteristics) in mass flow (used for capacity calculations).

The absolute humidity is a value useful when an Air Handling Unit (AHU) is used to control the humidity.

The dew point allows calculating a limit value where the cooling coils will not dry the air.

The `Psychrometric` function block
- simplifies the programming, and
- is based of the method specified by the formulas developed by Hyland and Wexler (1983a), and discussed by Olivieri (1996) used as reference by the *ASHRAE Handbook 2001, fundamental*.

Error Avoidance Features

The function block provides the following error avoidance feature to help avoid certain sources of machine malfunction:

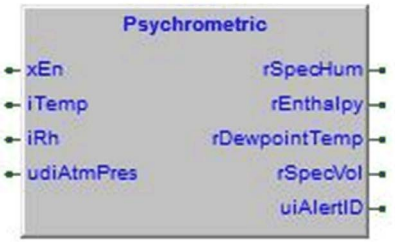
Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.

Section 29.2
Pin Description

Pin Description

Pin Diagram

The following picture presents the pin diagram of the `Psychrometric` function block:



Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
`xEn`	`BOOL`	TRUE or FALSE	N/A	Enable/disable the function block. TRUE = Enable
`iTemp`	`INT`	-500...+900	0.1 / °C	Dry bulb temperature
`iRh`	`INT`	0...1000	0.1 / %	Relative humidity
`udiAtmPres`	`UDINT`	0...618275	1 / Pa	Atmospheric pressure Default: 101325 Pa

Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
`rSpecHum`	`REAL`	-3.40E+38... +3.40E+38	N/A / g (water) / kg (air)	Spcific humidity (humidity ratio)
`rEnthalpy`	`REAL`	-3.40E+38... +3.40E+38	N/A / kJ/kg (dry air)	Moist air specific enthalpy
`rDewpointTemp`	`REAL`	-3.40E+38... +3.40E+38	N/A / °C	Dew-point temperature
`rSpecVol`	`REAL`	-3.40E+38... +3.40E+38	N/A / m^3/kg	Specific volume

Output	Data Type	Range	Scaling / Unit	Description
uiAlertID	UINT	0...65535	N/A	Alert ID

rSpecHum

The specific humidity rSpecHum (humidity ratio) is the mass of water vapour in a given volume air, measured in g (water) / kg (dry air).

rEnthalpy

The enthalpy rEnthalpy is a thermodynamic quantity equal to the sum of the internal energy of a system plus the product of the pressure volume work done on the system. Enthalpy refers to the total of sensible and latent heat or energy, measured in kJ/kg (dry air).

rDewpointTemp

Dew point temperature rDewpointTemp is the temperature of moist air saturated at the same pressure, with the same humidity ratio as that of the given sample of moist air.

Cooling of the temperature below the dewpoint temperature causes condensation of the water vapour (or frost) to occur.

rSpecVol

Specific volume rSpecVol is defined as the total volume of dry air and water vapour mixture per kg of dry air in m^3.

Alert ID Description

The output uiAlertID represents a value from 0 to 7, whereby each bit represents a detected alert. The table contains the bits and their description:

Alert Bit	Alert Cause	Effect
0	iTemp is not within the specified range.	The value of iTemp used for the calculations is limited according to the range.
1	iRh is not within the specified range.	The value of iRh used for the calculations is limited according to the range.
2	udiAtmPres is not within the specified range.	The value of udiAtmPres used for the calculations is limited according to the range.
3...15	N/A	N/A

Troubleshooting

Alert	Problem	Solution
`uiAlertID.0`	`iTemp` is not within the specified range.	Verify that the value of `iTemp` is within its specified range.
`uiAlertID.1`	`iRh` is not within the specified range.	Verify that the value of `iRh` is within its specified range.
`uiAlertID.2`	`udiAtmPres` is not within the specified range.	Verify that the value of `udiAtmPres` is within its specified range.

Chapter 30
Balance of two Pumps: RedundantPumpCntrl

What Is in This Chapter?

This chapter contains the following sections:

Section 30.1
Functional Overview

`RedundantPumpCntrl` Function Block Description

Function Block Description

The function block balances the operating hours of two pumps, and takes into account the status of each pump: operating time, detected error, maintenance mode.

If one of the pump is not available, the second pump will run continuously.

When the pumps are not used for a long time, to help avoid damage (corrosion, blocking, etc.) the pumps are switch on one after the other during a user-defined time.

All time settings can be configured.

Why Use the `RedundantPumpCntrl` Function Block?

The `RedundantPumpCntrl` function block:
- simplifies programming.
- reduces engineering and commissioning time.

Features of the `RedundantPumpCntrl` Function Block

The `RedundantPumpCntrl` function block provides the following features:
- balances of the running pumps
- loss compensation in case of error
- locked rotor protection

Error Avoidance Features

The function block provides the following error avoidance feature to help avoid certain sources of machine malfunction:

Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.

Section 30.2
Pin Description

Pin Description

Pin Diagram

The following picture presents the pin diagram of the `RedundantPumpCntrl` function block:

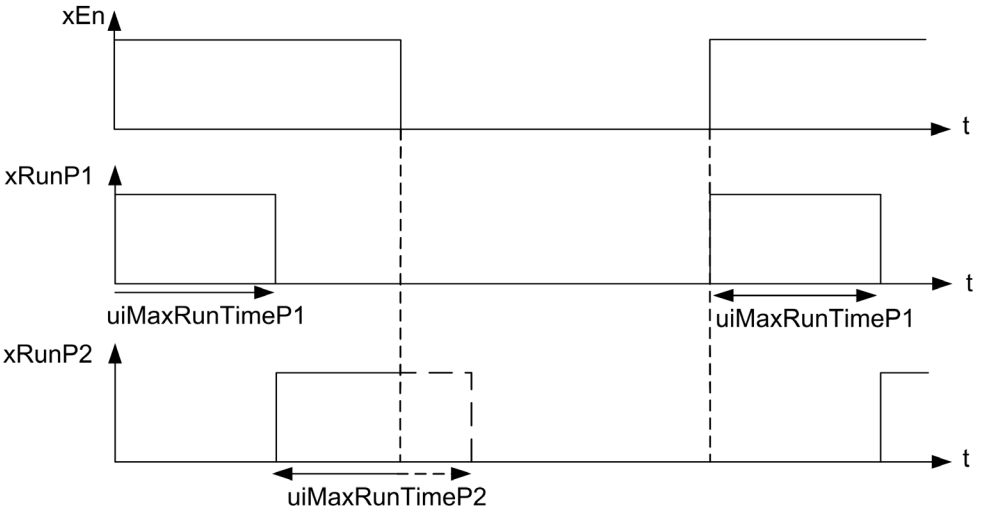
Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
`xEn`	`BOOL`	TRUE or FALSE	N/A	Enable/disable the function block.
`xStartStop`	`BOOL`	TRUE or FALSE	N/A	Start the pump TRUE = Start FALSE = Stop
`xAutoManual`	`BOOL`	TRUE or FALSE	N/A	Automatic or manual mode TRUE = Manual mode FALSE = Auto mode
`xManualRunP1`	`BOOL`	TRUE or FALSE	N/A	Start pump 1 in manual mode.
`xManualRunP2`	`BOOL`	TRUE or FALSE	N/A	Start pump 2 in manual mode.

Input	Data Type	Range	Scaling / Unit	Description
xMotorAlarmP1	BOOL	TRUE or FALSE	N/A	Signal from motor protection.
xMotorAlarmP2	BOOL	TRUE or FALSE	N/A	Signal from motor protection.
uiMaxRunTimeP1	UINT	1...1000	Hours	Maximum running time of pump 1.
uiMaxRunTimeP2	UINT	1...1000	Hours	Maximum running time of pump 2.
uiPModeRunTime	UINT	1...60000	Minutes	Duration of the locked rotor protection (PMode).
uiPModeIdleTime	UINT	0...1000	Hours	Max Idle time - Pump will be started for uiPModeRunTime.
xReset	BOOL	TRUE or FALSE	N/A	Alarm reset

Enable

If input xEn is FALSE, all pump outputs are switched off, all the timers and pump commands are reset.



Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
usiState	USINT	0...99	N/A	Current status
xRunP1	BOOL	TRUE or FALSE	N/A	Start/Stop command of pump 1.
xRunP2	BOOL	TRUE or FALSE	N/A	Start/Stop command of pump 2.

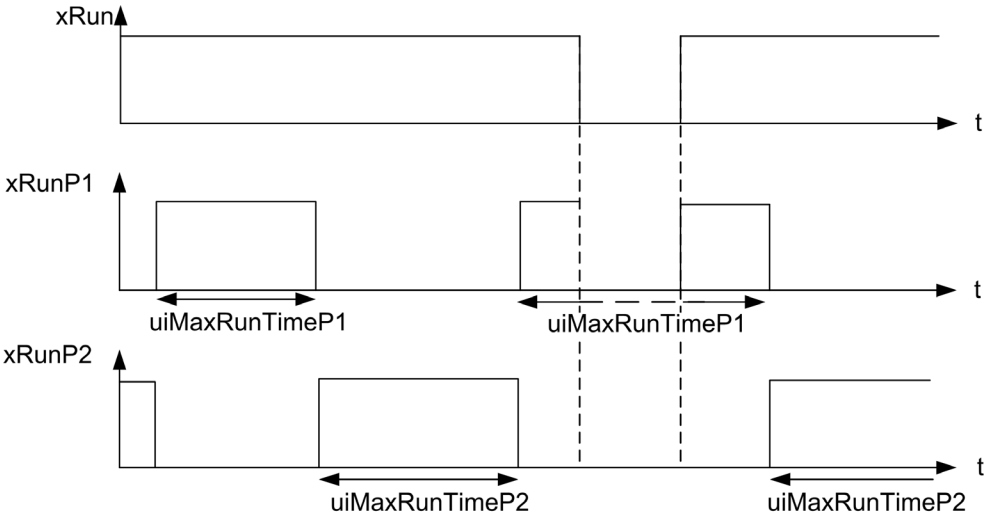
Output	Data Type	Range	Scaling / Unit	Description
uiRemainTimeP1	UINT	0...1000	Hours	Remaining time before pump 1 switches over to pump 2.
uiRemainTimeP2	UINT	0...1000	Hours	Remaining time before pump 2 switches over to pump 1.
uiRemainIdleTime	UINT	0...1000	Hours	Remaining time before pump starts for uiPModeRunTime.
uiAlarmID	UINT	0...6	N/A	Alarm ID

Normal Operation

- Pump 1 and pump 2 are available
- uiMaxRunTimeP1 or uiMaxRunTimeP2 > 0
- xEn = TRUE
- xRun = TRUE

When a pump did run for the set time, the AFB changes to the other pump.

If uiMaxRunTimeP1 or uiMaxRunTimeP2 is set to zero, the associated pump is unavailable.

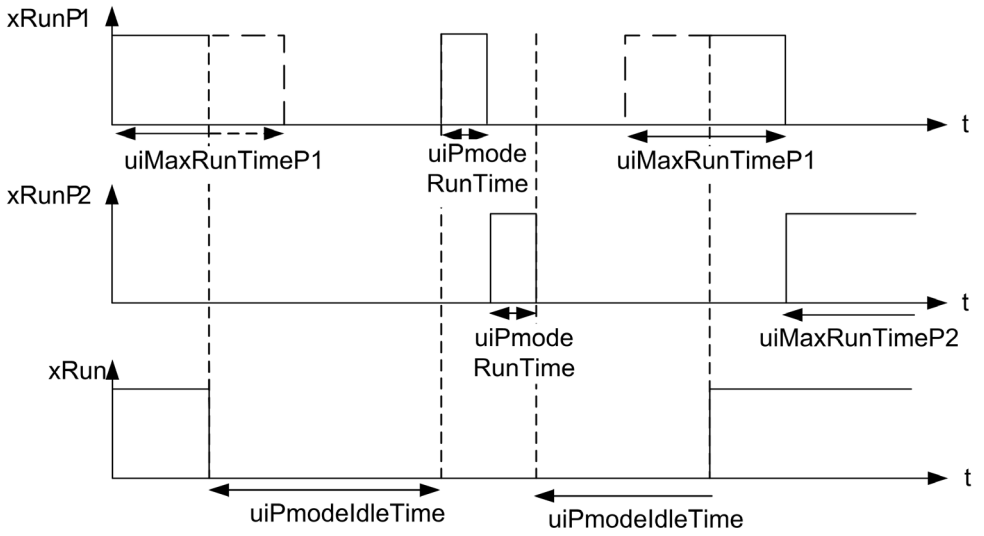


Locked Rotor Protection - Pmode

When the pumps (P1 and P2) are not used for a long time, the pumps are switch on to avoid damage (corrosion, blocking, etc.). The duration when the pumps are switch off and the duration of the Protection mode (Pmode) are set by the user.

If `uiPModeIdleTime` or `uiPModeRunTime` are set to zero, the locked rotor protection is not started and `uiRemainIdleTime` is set to zero.

If the locked rotor protection is running (P1 or P2 are switched on), `uiRemainIdleTime` is set to `uiPModeIdleTime`.



Alarm ID Description

If ...	Then ...
one of the inputs, `xMotorAlarmP1` or `xMotorAlarmP2`, is set to TRUE,	the other pump runs continously to avoid the shutdown of the machine.
the 2 inputs, `xMotorAlarmP1` and `xMotorAlarmP2`, are set to TRUE,	the pumps are switched off.

The `uiAlarmID` output represents a value from 0 to 65535, whereby each bit represents a detected alarm. The table contains the bits and their description:

Alarm Bit	Alarm Cause	Effect
0	Inputs `xMotorAlarmP1` and `xMotorAlarmP2` are set to TRUE.	Outputs `xRunP1` and `xRunP2` are set to FALSE.
1	Input `xMotorAlarmP1` is set to TRUE.	Outputs `xRunP1` is set to FALSE and `xRunP2` is set to TRUE.
2	Input `xMotorAlarmP2` is set to TRUE.	Outputs `xRunP2` is set to FALSE and `xRunP1` is set to TRUE.
3...15	N/A	N/A

Troubleshooting

Alarm	Problem	Solution
`uiAlarmID.0`	Input `xMotorAlarmP1` and `xMotorAlarmP2` are set to TRUE.	Check alarm of pump 1 and pump 2.
`uiAlarmID.1`	Input `xMotorAlarmP1` is set to TRUE.	Check alarm of pump 1.
`uiAlarmID.2`	Input `xMotorAlarmP2` is set to TRUE.	Check alarm of pump 2.

Chapter 31
Step Controller With Analog Output: StepControllerAnalog

What Is in This Chapter?

This chapter contains the following sections:

Section 31.1
Functional Overview

`StepControllerAnalog` Function Block Description

Function Block Description

The function block can command up to 6 digital outputs either as a linear step controller or as a binary step controller. Typically this function block is used in electrical heating applications. The linear / binary mode can be selected by using the `xMode` input.

The function block allows you to define 2 time values:
- Delay time On (`uiDelayTimeOn`)
- Delay time Off (`uiDelayTimeOff`)

These time delays prevent instantaneous on/off switching of the individual outputs, for example, heating.

Why Use the `StepControllerAnalog` Function Block?

The `StepControllerAnalog` function block:
- simplifies programming.
- reduces engineering and commissioning time.

Features of the `StepControllerAnalog` Function Block

The `StepControllerAnalog` function block provides the following features:
- balances of the running pumps
- loss compensation in case of error detected
- locked rotor protection
- analog control between two steps
- delay time between on/off switching

Error Avoidance Features

The function block provides the following error avoidance feature to help avoid certain sources of machine malfunction:

Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.

Section 31.2
Pin Description

Pin Description

Pin Diagram

The following picture presents the pin diagram of the `StepControllerAnalog` function block:

Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
`xEn`	`BOOL`	TRUE or FALSE	N/A	Enables the function block
`xAlarm`	`BOOL`	TRUE or FALSE	N/A	Alarm input
`uiXc`	`UINT`	0...10000	0.01 / %	Control signal
`usiSteps`	`USINT`	1...6	N/A	Number of steps
`xMode`	`BOOL`	TRUE or FALSE	N/A	Mode: FALSE = linear TRUE = binary
`uiDelayTimeOn`	`UINT`	0...600	1 / s	Minimum On Time
`uiDelayTimeOff`	`UINT`	0...600	1 / s	Minimum Off Time

xAlarm

If xAlarm is set to TRUE (e.g. maximum temperature thermostat) all outputs are set to 0.

The alarm management has to be programmed outside of the AFB.

usiSteps

The input usiSteps defines the number of steps. If the value is not set within the specified range, the outputs xStep are set to 0.

Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
uiDeltaAnalog	UINT	0...10000	0.01 / %	Output for fine control
xHold	BOOL	TRUE or FALSE	N/A	Hold signal for the regulator
xStep1	BOOL	TRUE or FALSE	N/A	Digital output step 1
xStep2	BOOL	TRUE or FALSE	N/A	Digital output step 2
xStep3	BOOL	TRUE or FALSE	N/A	Digital output step 3
xStep4	BOOL	TRUE or FALSE	N/A	Digital output step 4
xStep5	BOOL	TRUE or FALSE	N/A	Digital output step 5
xStep6	BOOL	TRUE or FALSE	N/A	Digital output step 6

xHold

The output xHold can be used to hold the signal of a PID for example during stage up time uiDelayOn or stage down time uiDelayOff runs.

Step Controller: Linear Mode

If the input xMode is set to FALSE the linear mode is active.

Regulation without delay:

The following figure presents a sample for a binary step controller with 4 digital outputs.
xMode = FALSE (the linear mode is active)
usiSteps = 4 (the total number of steps is 4)
uiDelaytimeOn = 0s
uiDelaytimeOff = 0s

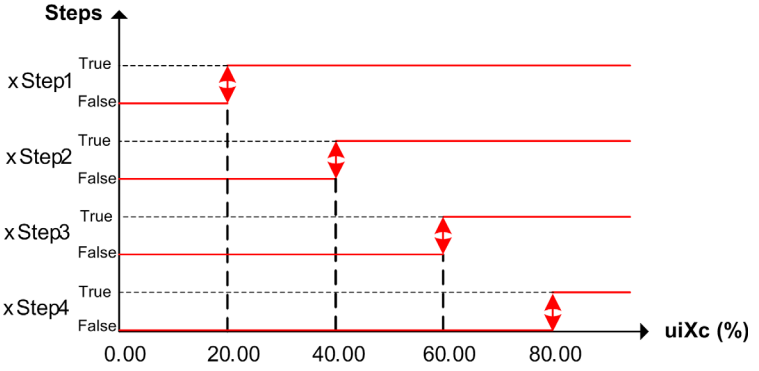
The input signal uiXc (0...100%) is divided over 4 steps:

xStep1 is set to TRUE, when uiXc >= 20%, xStep1 is set to FALSE, when uiXc <= 20%

xStep2 is set to TRUE, when uiXc >= 40%, xStep2 is set to FALSE, when uiXc <= 40%

xStep3 is set to TRUE, when uiXc >= 60%, xStep3 is set to FALSE, when uiXc <= 60%

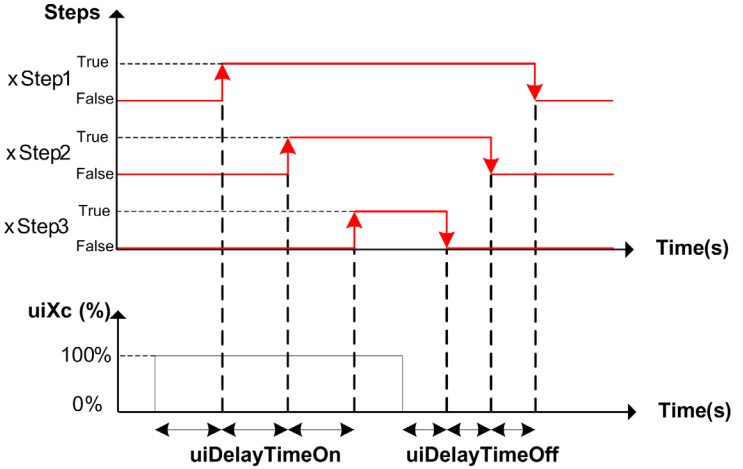
`xStep4` is set to TRUE, when `uiXc` >= 80%, `xStep4` is set to FALSE, when `uiXc` <= 80%



Regulation with delay:

The following figure presents a sample for a binary step controller with 3 digital outputs.
xMode = FALSE (the linear mode is active)
usiSteps = 3 (the total number of steps is 3)
uiDelaytimeOn > 0s
uiDelaytimeOff > 0s

The hysteresis is defined by the timers `uiDelaytimeOn` and `uiDelaytimeOff`. The delays `uiDelayTimeOn` and `uiDelayTimeOff` are identical for each step.



Step Controller: Binary Mode

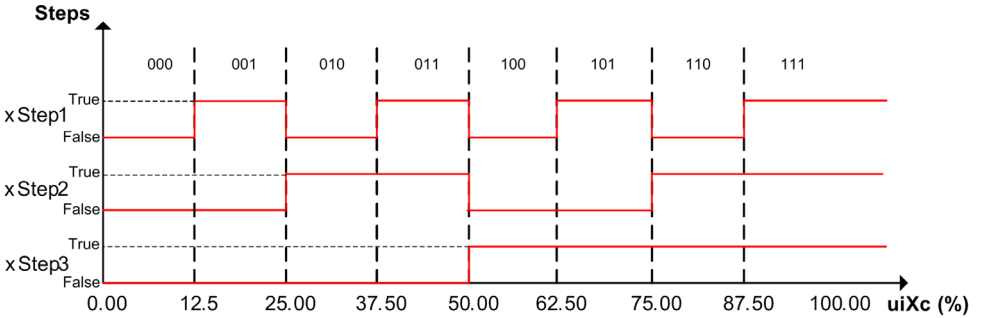
If `xMode` is set to TRUE, the binary mode is active and the steps are controlled based on the binary signal.

This mode is typically used when the loads connected to the steps do not have the same capacity.

The following figure presents a sample for a binary step controller with 3 digital outputs.
xMode = TRUE (the binary mode is active)
usiSteps = 3 (the total number of possible steps is 8)

The total number of possible intermediate steps is 8 (0...7).

Binary code:
0 = 000
1 = 001
2 = 010
3 = 011
4 = 100
5 = 101
6 = 110
7 = 111



Analog Controller

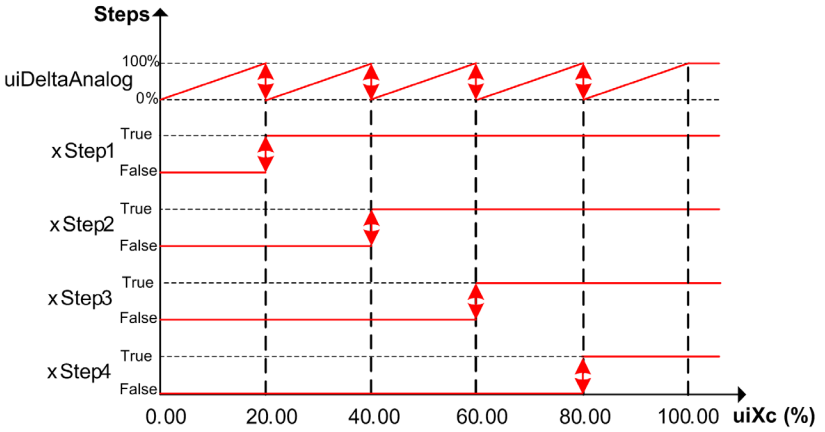
The analog output `uiDeltaAnalog` is used for analog control between 2 steps.

It can be used with both modes: linear and binary step mode.

The following figure presents a sample for the linear step mode. The analog control works between the steps.
xMode = FALSE (the linear mode is active)
usiSteps = 6 (the total number of possible steps is 6)

If the setpoint `uiXc` is increasing, the output `uiDeltaAnalog` increases from 0.00…100.00% between the steps.

If the setpoint `uiXc` is decreasing the output `uiDeltaAnalog` decreases from 100.00…0.00% between the steps.


Steps
100%
uiDeltaAnalog
0%
True
xStep1
False
True
xStep2
False
True
xStep3
False
True
xStep4
False
0.00
20.00
40.00
60.00
80.00
100.00
uiXc (%)

Chapter 32
Thermal Power and Energy Calculation: ThermalPower_Enthalpy

What Is in This Chapter?

This chapter contains the following sections:

Section 32.1
Functional Overview

`ThermalPower_Enthalpy` Function Block Description

Function Block Description

The function calculates thermal power and thermal energy based on mass flow and on enthalpies.

The enthalpy assumes a specific enthalpy in kJ/kg.

The mass flow assumes a unit in g/s.

Why Use the `ThermalPower_Enthalpy` Function Block?

The `ThermalPower_Enthalpy` function block:

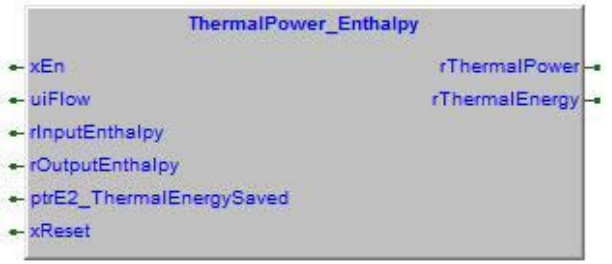
- Calculates thermal power and thermal energy.
- Simplify programming

Section 32.2
Pin Description

Pin Description

Pin Diagram

The following picture presents the pin diagram of the `ThermalPower_Enthalpy` function block:



Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
`xEn`	`BOOL`	TRUE or FALSE	N/A	Enable/disable the function block. TRUE = Enable
`uiFlow`	`UINT`	0...65535	0.1 / g/s	Mass flow
`rInputEnthalpy`	`REAL`	-3.40E+38 ... +3.40E+38	N/A / kJ/kg	Input enthalpy
`rOutputEnthalpy`	`REAL`	-3.40E+38 ... +3.40E+38	N/A / kJ/kg	Output enthalpy
`ptrE2_ThermalEnergySaved`	`REAL`	N/A	N/A	`rThermalEnergySaved` pointer for non-volatile memory variable.
`xReset`	`BOOL`	TRUE or FALSE	N/A	Reset the calculation of the thermal energy.

`xReset`

A rising edge on the `xReset` input resets the `rThermalEnergy` to zero.

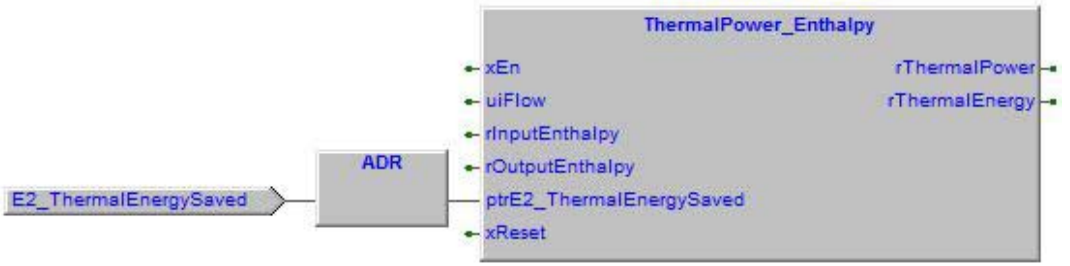
Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
rThermalPower	REAL	-3.40E+38 ... +3.40E+38	N/A / kW	Thermal power
rThermalEnergy	REAL	-3.40E+38 ... +3.40E+38	N/A / kWh	Thermal energy

Non-Volatile Memory Variables

Address	Variable	Data Type	Range	Scaling / Unit	Description
N	E2_rThermalEnergySaved	REAL	-3.40E+38 ... +3.40E+38	N/A / kWh	Thermal energy saved

Connectivity Diagram



NOTE:

- You have to define a parameter in the non-volatile memory parameters in the application.
- You have to use an ADR block to connect the input pin ptrE2_ThermalEnergySaved to the variable defined in the non-volatile memory.
- If several instances of the function block are created inside the project, you have to create 1 new parameter (E2_rThermalEnergySaved) in the non-volatile memory for each function block.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Be sure that all variables are initialized to an appropriate value before their first use as array indices or pointers.
- Write programming instructions to test the validity of operands intended to be used as array indices and memory pointers.
- Do not attempt to access an array element outside the defined bounds of the array.

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

Chapter 33
Thermal Power Calculation: ThermalPowerCalculation

What Is in This Chapter?

This chapter contains the following sections:

Section 33.1
Function Block Description

ThermalPowerCalculation Function Block Description

Function Block Description

The ThermalPowerCalculation function block calculates the thermal power and the thermal energy produced. The calculation of the thermal power of a hydraulic system is based on the mass flow rate, the heat capacity of the medium, and the temperature difference before and after heat dissipation.

The ThermalPowerCalculation function block may be used with the COPcalculation function block *(see page 209)* to determine the machine efficiency, or may be used independently.

Why Use the ThermalPowerCalculation Function Block?

The ThermalPowerCalculation function block:

- simplifies programming.
- reduces engineering and commissioning time.

Features of the ThermalPowerCalculation Function Block

The ThermalPowerCalculation function block provides the following features:

- provides the instant thermal power and the total thermal energy.
- allows you to measure total thermal energy over a used-defined period.
- can work with metering devices providing either mass flow or volume flow information.
- provides thermal energy information in the right format to be used on the COPcalculation function block *(see page 209)*.

The thermal power is calculated based on the following formula:

$P_{th} = m^* \cdot c \cdot \Delta T \cdot 1000$

[Pth] = kg/s * kW*s / (kg * K) * K = kW

with

m* = mass flow in kg/s

c = heat capacity in kWs / (kg*K) = kJ/(kg*K)

ΔT = temperature difference

For meters providing a volume flow, the mass flow is calculated according to:

$\dot{m} = \dot{V} * p$
$[\dot{m}] = l/s * kg/l$
with
$\dot{V}$ = volume flow in l/s
p = density in kg/l

Error Avoidance Features

The function block provides the following error avoidance feature to help avoid certain sources of machine malfunction:

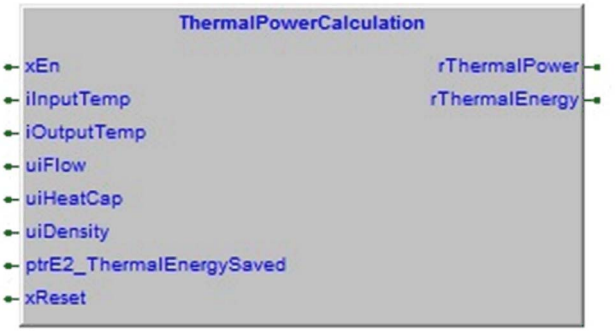
Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.

Section 33.2
Pin Description

Pin Description

Pin Diagram

The following picture presents the pin diagram of the `ThermalPowerCalculation` function block:



Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
`xEn`	`BOOL`	TRUE or FALSE	N/A	Enable the function block.
`iInputTemp`	`INT`	-32768...+32767	0.1 / °C (°K)	Input temperature
`iOutputTemp`	`INT`	-32768...+32767	0.1 / °C (°K)	Output temperature
`uiFlow`	`UINT`	0... 65535	0.1 / kg/s	Mass flow
`uiHeatCap`	`UINT`	0... 65535	0.1 / kJ / (kg * K)	Heat capacity of the medium
`uiDensity`	`UINT`	0... 65535	0.1	Density correction value (see the values in the table for `uiDensity` below). Default: 1.0
`ptrE2_Thermal EnergySaved`	`@REAL`	N/A	N/A	`rThermalEnergySaved` pointer for non-volatile memory variable
`xReset`	`BOOL`	TRUE or FALSE	N/A	Reset

NOTE: A rising edge on the `xReset` input resets the energy calculation to 0.

Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
`rThermalPower`	`REAL`	-3.40E+38... +3.40E+38	kW	Actual thermal power
`rThermalEnergy`	`REAL`	-3.40E+38... +3.40E+38	kWh	Accumulated thermal energy

Non-Volatile Memory Variable

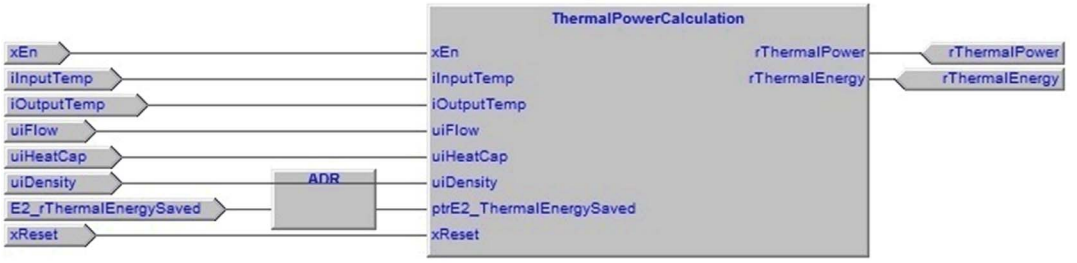
Address	Variable	Data Type	Range	Scaling / Unit	Description
N	`E2_rThermalEnergySaved`	`REAL`	-3.40E+38 ... +3.40E+38	N/A / kWh	Thermal energy saved

`uiDensity`

Depending on the flow type, the correction value is taken from the input `uiDensity`. The following table presents the correction on the example of water and brine:

Flow	Unit	`uiDensity` Water (4° C)	`uiDensity` Brine (0° C / 25%)
Mass	kg/s	1	1.05
Volume	l/s	1 kg/l	1.05 kg/l
Volume	m^3/s	1000 kg/m^3	1050 kg/m^3
Volume	m^3/h	1000/3600 kg/m^3	1050/3600 kg/m^3
Water (4° C): c = 4.205 kJ / (kg * K), p = 1 kg/l			
Brine (0° C / 25%): c = 3.700 kJ / (kg * K), p = 1.05 kg/l			

Connectivity Diagram



NOTE:

- You have to define a parameter in the non-volatile memory parameters in the application.
- You have to use an `ADR` block to connect the input pin `ptrE2_ThermalEnergySaved` to the variable defined in the non-volatile memory.
- If several instances of the function block are created inside the project, you have to create 1 new parameter (`E2_rThermalEnergySaved`) in the non-volatile memory for each function block.

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

- Be sure that all variables are initialized to an appropriate value before their first use as array indices or pointers.
- Write programming instructions to test the validity of operands intended to be used as array indices and memory pointers.
- Do not attempt to access an array element outside the defined bounds of the array.

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

Chapter 34
Three Point Actuator: ThreePointActuator

What Is in This Chapter?

This chapter contains the following sections:

Section 34.1
Functional Overview

ThreePointActuator Function Block Description

Function Block Description

This function block controls a 3-point actuator.

3-point actuator has two digital outputs, one for closing and one for opening action.

The function block calculates the required commands based on the control signal and the actual position. The actual position is calculated internally.

Calibration function allows the actuator to synchronize the actual position with the internal `uiPosition` value. After the calibration function is triggered, the function block will set the `xClose` output for the `uiMaxRunTime`.

The function block does not store the position in the persistent memory. Therefore, run the calibration after each power cycle of the controller.

Why Use the ThreePointActuator Function Block?

The `ThreePointActuator` function block:
- simplifies programming,
- reduces engineering and commissioning time,
- can be used with various type of 3-point actuator.

Features of the ThreePointActuator Function Block

The `ThreePointActuator` function block provides the following features:
- hold input to freeze the position of the actuator
- calibration input to control the calibration of the actuator
- overdriving function of 110% of `uiMaxRunTime` in case of calibration

Error Avoidance Features

The function block provides the following error avoidance feature to help avoid certain sources of machine malfunction:

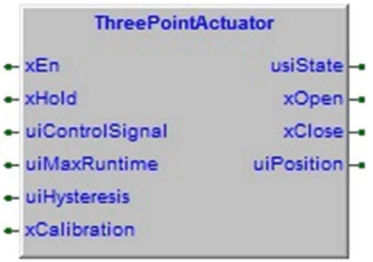
Error Avoidance Feature	Description
Input range validation	Input and parameter ranges are validated to help prevent out of range data.

Section 34.2
Pin Description

Pin Description

Pin Diagram

The following picture presents the pin diagram of the `ThreePointActuator` function block:



Input Pin Description

Input	Data Type	Range	Scaling / Unit	Description
`xEn`	`BOOL`	TRUE or FALSE	N/A	Enable/disable the function block.
`xHold`	`BOOL`	TRUE or FALSE	N/A	Hold input
`uiControlSignal`	`UINT`	0...10000	0.01%	Setpoint for the actuator
`uiMaxRuntime`	`UINT`	20...40000	0.1/s	Actuator run time. The time that is needed by the actuator to run from 0 to 100%
`uiHysteresis`	`UINT`	0...3000	0.01%	Hysteresis based on the actual position of the actuator. The actuator remains if `uiControlSignal` is within the band `uiPosition`+/- `uiHysteresis`. Setting a value greater than 0 for `uiHysteresis` increases the lifetime of the actuator.
`xCalibration`	`BOOL`	TRUE or FALSE	N/A	Calibration command

Enable

A rising edge enables the function block and starts the calibration. The control of the actuator starts once the calibration is completed.

Calibration

The input `xCalibration` allows the actuator to synchronize the actual position with the internal `uiPosition` value. After the calibration function is triggered the function block will set the `xClose` output for the `uiMaxRunTime`.

The input `xCalibration` must be maintained to TRUE during all the calibration, otherwise the calibration stops.

Output Pin Description

Output	Data Type	Range	Scaling / Unit	Description
`usiState`	`USINT`	0...99	N/A	Present state of the machine: **1** Idle **20** Target position reached **21** Opening **22** Closing **23** Calibrating **24** Hold **99** Alarm state
`xOpen`	`BOOL`	TRUE or FALSE	N/A	Actuator open signal
`xClose`	`BOOL`	TRUE or FALSE	N/A	Actuator close signal
`uiPosition`	`UINT`	0...10000	0.01%	Actuator position

Troubleshooting

If the function block goes in alarm state `usiState` = 99 the outputs `xOpen`, `xClose` and `uiPosition` are set to zero.

Alarm	Problem	Solution
`uiState` = 99	Invalid value of the input `uiControlSignal`.	**1.** Verify the input ranges. **2.** Set the values within the defined ranges.
`uiState` = 99	Invalid value of the parameter `uiMaxRuntime`.	**1.** Verify the parameter ranges. **2.** Set the values within the defined ranges.
`uiState` = 99	Invalid value of the parameter `uiHysteresis`.	**1.** Verify the parameter ranges. **2.** Set the values within the defined ranges.

Chapter 35
Quick Response Code Generator: QRcodeGenerator

What Is in This Chapter?

This chapter contains the following sections:

Section 35.1
Functional and Machine Overview

Functional Overview

Functional Description

The `QRCodeGenerator` (Quick Response Code Generator) function block converts a string of ASCII characters to the respective quick response code. The maximum allowable number of ASCII characters is 321, when the QR code maximum output is in version 11 format, as referenced in ISO/IEC 18004 international standard.

http://go2se.com/ref=TM172PDG42R

Why Use the `QRCodeGenerator` Function Block?

The `QRCodeGenerator` function block is used for the following purposes:

- Allows to compact much information in a small area of display.
- The code can be read by many commonly used mobile devices.

It allows to provide more customized services, for example direct contact to customer care, link to an URL or an App.

Features of the `QRCodeGenerator` Function Block

The function block provides the following features:

- Allows to convert the input string to output map from version 1 to version 11 of QR standard with 7% as error correction level recovery.
- Allows to convert:
  - maximum 321 characters with version 11 output;
  - maximum 271 characters with version 10 output;
  - maximum 230 characters with version 9 output;
  - maximum 192 characters with version 8 output;
  - maximum 154 characters with version 7 output;
  - maximum 134 characters with version 6 output;
  - maximum 106 characters with version 5 output;
  - maximum 78 characters with version 4 output;
  - maximum 53 characters with version 3 output;
  - maximum 32 characters with version 2 output;
  - maximum 17 characters with version 1 output;
- Selectable automatic version output based on the input string.

Input Range Validation

Input and parameter ranges are validated to help prevent out of range data.

Section 35.2
Pin Description

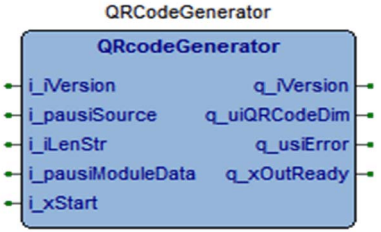
What Is in This Section?

This section contains the following topics:

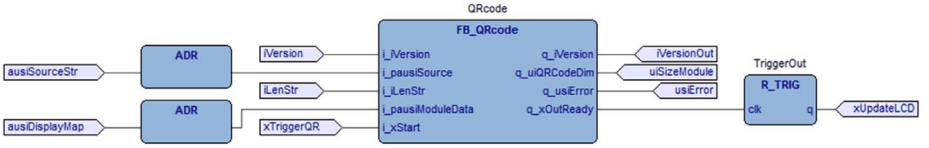
Input Pin Description

Pin Diagram

The following graphic shows the pin diagram of `QRCodeGenerator`:



The following graphic shows an example of an instance:



Input Pin Description

Input	Data Type	Range	Scaling/Unit	Description
`i_iVersion`	`Int`	0...11	N/A	If 0 is selected, then the lower version capable to contain the input string is automatically selected.
`i_pausiSource`	`Pointer`	N/A	N/A	Pointer to string. Pointer to input string to convert to QR code.
`i_iLenStr`	`Int`	0...321	N/A	String length
`i_pausiModuleData`	`Pointer`	N/A	N/A	Pointer to the output area where to locate the QR output. The size of the area pointed depends on the version required, as in the formula: (size (Module/Data) =(`i_iversion` x 4 +17)^2 bytes. ● Version 10 has 3,249 bytes ● Version 1 has 441 bytes.
`i_xStart`	`Bool`	N/A	N/A	Trigger FALSE to TRUE, to start conversion.

Output Pin Description

Output Pin Description

Output Pin Description

Output	Data Type	Range	Scaling/Unit	Description
q_iVersion	INT	0..11	N/A	Output version of QR code. Redundancy with q_i_QRCodeDim. q_i_QRCodeDim=q_i_Version x 4 + 17
q_iQRCodeDim	INT	0...61	N/A	Size of the QR-code. Redundancy with q_i_Version.
q_usiError	USINT	0, 1, 2, 4	N/A	Error code
q_xOutReady	BOOL	N/A	N/A	Data available End of QR-conversion.

Error Description

The error output represents a value from 0 to 255, whereby each bit represents a detected alarm. The bits and their description are described in the following table:

Error Bit	Alarm Description	Result
0	OUT_OF_RANGE At least one input is out of range.	No QR code generation.
1	VER_TOO_SMALL The version is too small in relation to the input string data.	
2-15	N/A	N/A

Section 35.3
Troubleshooting

Troubleshooting

Troubleshooting

Alarm / Alert	Problem	Solution
0	At least one input is out of range.	Verify the range for each input variable considering the `i_xUnitRpm` selection.
1	The required output `Version` is too small to contain the input string data.	• Increase the required version. • Set the automatic version selection. • Reduce the input string length.

Glossary

A

AFB

Application Function Block

AHU

Air Handling Unit

C

CCS

Cooling Coil Signal

COP

Coefficient of Performance

D

DAT

Discharge Air Temperature

DCS

Damper Control Signal

Deadband

A range of the controlled variable in which no corrective action is taken by the controlled system and no energy is used.

F

FCS

Fan Control Signal

H

HCS

Heating Coil Signal

HVAC&R

Heating, Ventilation, Air Conditioning and Refrigeration

M

Modbus

The protocol that allows communications between many devices connected to the same network.

O

OAT

Outdoor Air Temperature

P

PID

The Proportional, Integral and Derivative control is a generic control loop feedbackmechanism (controller) widely used in industrial control systems.

R

RAT

Return Air Temperature

RMT

Room temperature

V

VSD

Variable Speed Drive

Index

Y