AHU Systems

User Guide

DISTECH CONTROLS™

Innovative Solutions for Greener Buildings™

Legal

While all efforts have been made to verify the accuracy of information in this manual, Distech Controls is not responsible for damages or claims arising from the use of this manual. Persons using this manual are assumed to be trained HVAC professionals and are responsible for using the correct wiring procedures, correct override methods for equipment control and maintaining safe working conditions in fail-safe environments. Distech Controls reserves the right to change, delete or add to the information in this manual at any time without notice.

Distech Controls, the Distech Controls logo, Innovative Solutions for Greener Buildings, ECO-Vue,

and Allure are trademarks of Distech Controls Inc.; BACnet is a registered trademark of ASHRAE; Niagara Framework and Niagara^{AX} Framework is a registered trademark of Tridium, Inc.

Table of Contents

AHU General Information

Air Handling Units (AHUs) are the most important elements in most mechanical ventilation systems. They may be very broadly defined as mechanical systems designed to process and supply external air into the building while at the same time extracting used air from the building. However, their function and features, which are closely followed by their construction, may vary substantially.

The following is a list of the most important issues connected to the AHU system's design and construction.

Main functions of the AHUs

- ventilation – provision of a hygienic (or technologically required) volume of fresh air.
- ventilation and heating – combination of supplying the hygienic air volume and transfer of heat required to compensate for heat loss.
- air conditioning - provision of the hygienic volume of fresh air, heating up/cooling down of air to compensate for heat loss/gains in the building; with optional humidification and dehumidification of the supplied air.
- special use – to name a few - clean rooms, hospital operating theatres, pharmaceutical laboratories, data centres, etc.

Type of building in which installation is used

- is recirculation allowed and justified.
- is heat recovery possible – if yes then what type may be used – glycol, heat wheel, plate exchanger, heat pipe or other. Does it have to be gastight. Is heat exchanger anti-freeze protection required.
- is a heat exchange to be temperature or enthalpy based.
- is variable air flow/precise air pressure control required, therefore forcing use of variable frequency driven fans.

External conditions

- a wintertime minimum temperature – it defines a need for a heating system, anti-freeze protection and additional construction/installation requirements, namely an anti-freeze thermostat or/and sensor (water and/or air side), correct hydraulic circuit and equipment (pump, valve and piping)
- a wintertime minimum absolute humidity – it defines need for humidification system. It needs to be designed in regard to an absolute humidity demand (usually expressed in grams of water per kilogram of air).
- a summer maximum temperature and sun operation – they define a need for a cooling system
- a summer maximum absolute humidity – it defines a need for a dehumidification system. Therefore, an oversized cooling installation able to compensate for additional latent heat recovered during condensation process or an additional air-drying subsystem is required.

The AHUs require a high quality DDC (direct digital controller) to fully exploit their functionality. The DDC controller uses a set of logical and hardware signals to operate an AHU.

Signal Type	Description
AI/AO	Analogue input/output
MI/MO	Multistate input/output
BI/BO	Binary input/output
Sch	Scheduler signal
Cal	Calendar signal

The controller gathers information from logical and hardware input point passes it through an internal programme to determine control commands for execution elements and status indicators.

This documentation provides a detailed description of all the control blocks used in AHU control programmes from a gfxApplications library.

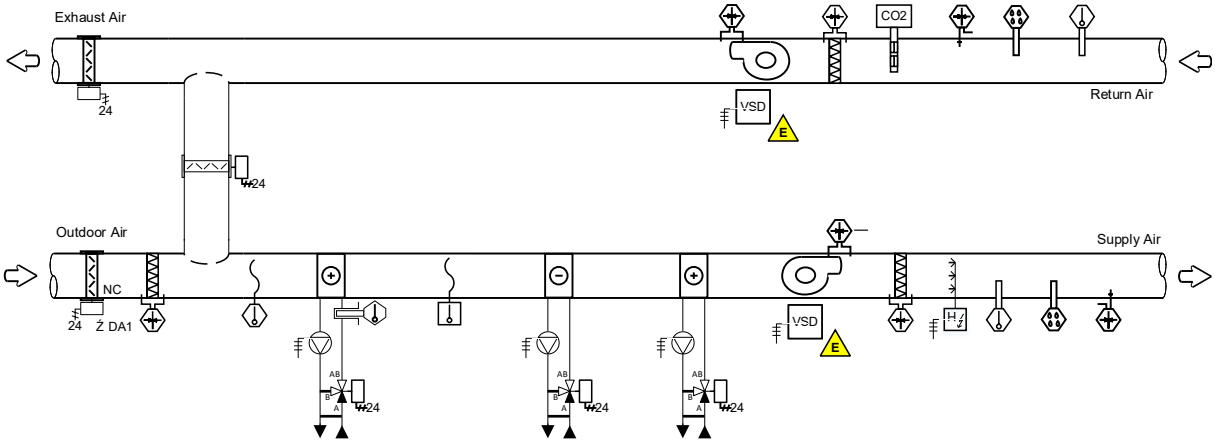


Figure 1: AHU scheme corresponding to the EC-gfxApplications AHU programme type A1

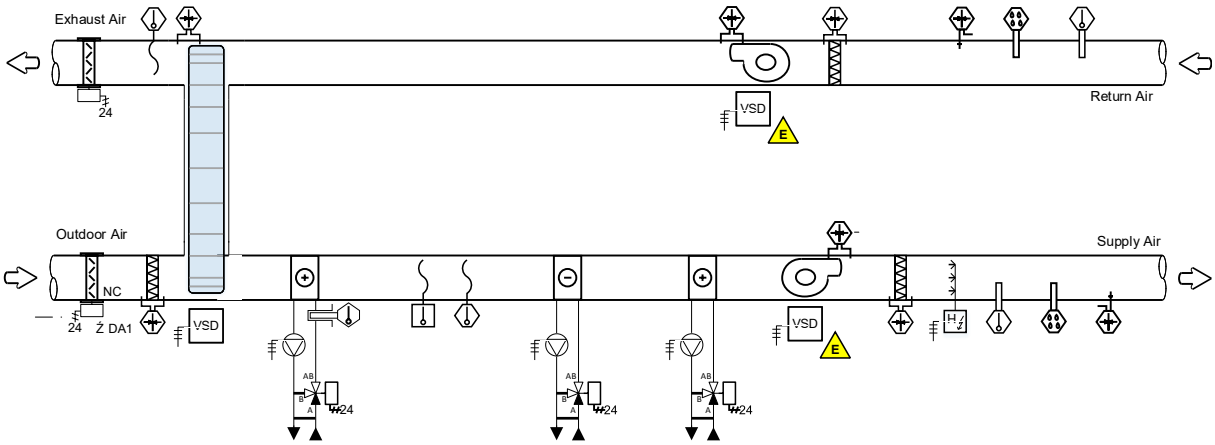


Figure 2: AHU scheme corresponding to the EC-gfxApplications AHU programme type B1

The AHU unit as it is used for EC-*gfx*Applications AHU program type A1/A2 and B1/B2 development is set up from following components.

Supply Air Side

Supply Side Component	Description
Dampers	- In the type A1/A2 – recirculation dampers used to for heat recovery - In the type B1/B2 – isolation dampers
Air filters	Up to three filters – two in the supply section and one in the return section.
Heat exchanger	Only in type B1/B2 – used for energy saving by recovering heat or cold from an exhaust air stream.
Preheater coil	Used as a first stage of heating (needs to be protected by an anti-freeze sensor in a cold climate areas).
Cooling coil	Used in the air conditioning and dehumidification processes.
Reheater coil	Used as a second stage of heating (particularly important in the dehumidification process).
Supply fan	Usually operated by a variable frequency drive.
Humidifier	Water or steam based.

Return Air Side

Return Side Component	Description
Air filter	Used to filter return air.
Return fan	Usually operated by a variable frequency drive.
Heat exchanger	Only in type B1/B2 – used for energy saving by recovering heat or cold from an exhaust air stream.
Dampers	- In the type A1/A2 – recirculation dampers used to for heat recovery - In the type B1/B2 – isolation dampers

AHU Operation – OperatingMode Page

An *OperatingMode* programming page provides general operating functions and reasoning to establish effective operation command and unit status. Additionally, it provides functions consisting of the average *OutsideAirTemp* calculation, filter alarm generation, alarm reset, and output active timer reset.

The AHU Unit Operation Command Block (AHU_UnitCommand)

An *AHU_UnitCommand* block gathers and prioritizes operation commands designed to start normal operation of the AHU.

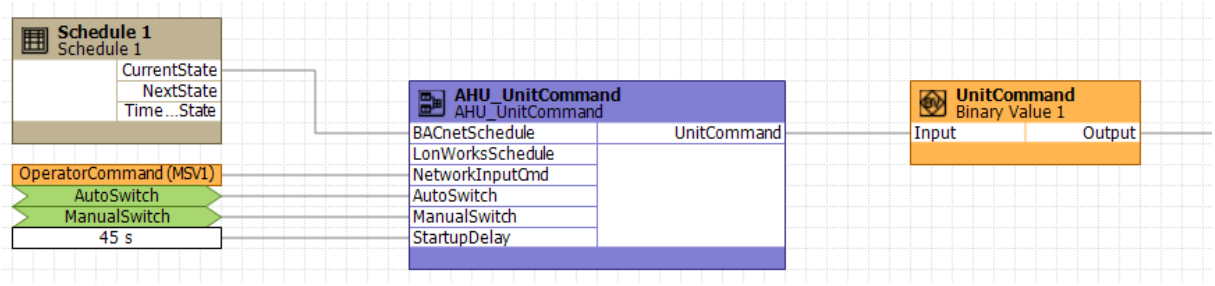


Figure 3: The AHU_UnitCommand programming block

Inputs

The *AHU_UnitCommand* block requires connection of input signals (hardware and logic).

Required Inputs

The block does not use mandatory inputs.

Optional Inputs

Input Parameter	Description
BACnetSchedule	A multistate value containing BACnet style occupancy command for the AHU. Default *null* (MI)
LonWorkstSchedule	A multistate value containing Lon Works style occupancy command for the AHU. Default *Unoccupied* (MI)
NetworkInputCmd	A multistate value used for a network originated operation command (designed to be used as BMS system switch). Default *Auto* (MI)
ManualSwitch	A binary input from a hardware switch used to enable the AHU's operation regardless of the schedule. Default *false* (BI)
AutoSwitch	A binary input from a hardware switch used to enable the schedule based operation the AHU. Default *true* (BI)
StartupDelay	A delay time that must elapse after controller reboot, before a command to start the AHU can be issued. Default *45s* (AI)

Outputs

The *AHU_UnitCommand* block output signals (hardware and logic).

Output Parameter	Description
UnitCommand	An effective AHU operating command. (BO)

Block Functions

UnitCommand Decision Algorithm

The lowest priority is assigned to the *BACnetSchedule* and *LonWorksSchedule.* The difference between the two is in the way the occupancy modes are coded into a multistate enumeration. If both the BACnet and LonWorks schedules are connected, it is the BACnet one which will operate the unit.

BACnet Schedule

State	Command Sent
0 – not a valid value	N/A
1 – Occupied	ON
2 – Unoccupied	OFF
3 – Bypass	ON
4 – Standby	ON

LonWorks Schedule

State	Command Sent
0 – Occupied	ON
1 – Unoccupied	OFF
2 – Bypass	ON
3 – Standby	ON

The signal form the schedules is then passed through the *NetworkInputCmd* section, which can adopt 3 valid states.

NetworkInputCmd

State	Command Sent
0 – not a valid value	N/A
1 – OFF	OFF
2 – ON	ON
3 – AUTO	The command from the schedule is sent.

The program then checks the state of the *AutoSwitch* input connected to a hardware switch. If it is active, then effective command resolved by the preceding stages is sent. If not, then an OFF command is sent.

The last in the queue is the *ManualSwitch* hardware input. If it is active, an ON command is sent, otherwise the command from the *AutoSwitch* is used.

The effective command is used to set a *UnitCommand* output.

StartupDelay

The block will delay any unit activation command after the controller is rebooted. The delay time is defined by the *StartupDelay* parameter.

The AHU Freeze Alarm Block (AHU_FreezeAlarm)

An *AHU_FreezeAlarm* block gathers information on freezing risk for the preheater (the main heating coil). Once hazardous conditions are detected, the block would activate *FreezeAlarm* output, which is used further in the program to transfer information to the *AHU_OperatingMode*, which in turn invokes operating mode *6.FreezeAlarm*.

Please note that it is up to all the equipment control blocks to undertake correct action in response to the *FreezeAlarm* status. Standard reaction is to stop the fans from operating, close dampers, start the preheater pump and open its valve to 100%.

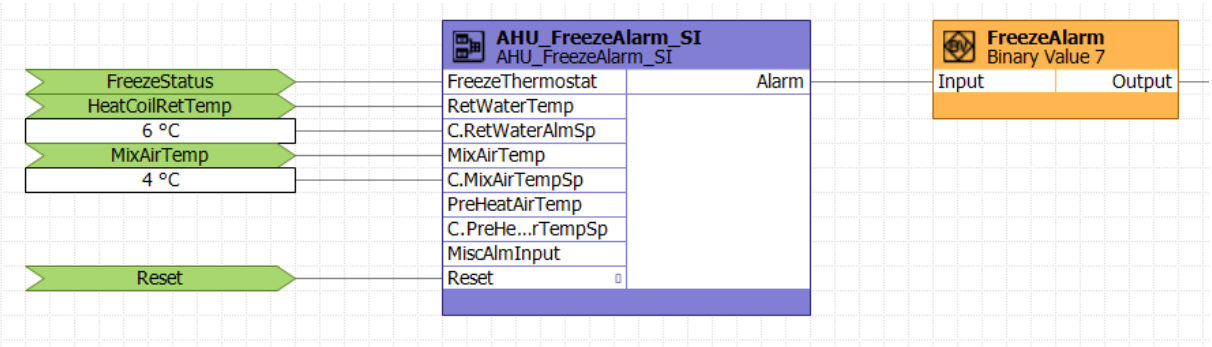


Figure 4: The AHU_FreezeAlarm programming block

Inputs

The *AHU_FreezeAlarm* block requires connection of input signals (hardware and logic).

Required Inputs

Input Parameter	Description
Reset	Alarm reset command (BI).

Optional Inputs

Input Parameter	Description
FreezeThermostat	A binary input signal from a freeze protection thermostat. Default false (BI)
RetWaterTemp	An input from a water temperature sensor installed in a return pipe, directly after the preheater. Default null (AI)
MixAirTemp	An input from an air temperature sensor located in a mixing compartment before the preheater. Default null (AI)
PreHeatAirTemp	An input from an air temperature sensor installed in an air stream directly after the preheater. Default null (AI)
MiscAlmInput	A binary input from other freeze alarm sensors. Default false (BI)

Outputs

Listed below are the *AHU_FreezeAlarm* block output signals (hardware and logic).

Output Parameter	Description
Alarm	An alarm output. (BO)

Block Functions

FreezeAlarm Decision Algorithm

The block checks all the connected inputs to verify if the heating coil is at risk of freezing. The inputs have an identical priority. Activation of just one alarm condition is sufficient to stop the AHU.

Parameter	Input Type	Description
FreezeThermostat	Binary	If the capillary thermostat installed on the heater detects a local temperature drop, the input is activated and results in a direct alarm activation. By default, a normally closed signal is used, so wire damage can also result in the alarm activation.
RetWaterTemp	Analog	If a measured value on the water temperature sensor in the return pipe of the heating coil drops below the *C.RetWaterAlmSp* parameter, the alarm is activated. The water temperature must rise by 2ºC above the *C.RetWaterAlmSp* before the alarm reset is possible.
MixAirTemp	Analog	If a measured value on the air temperature sensor that is installed in a mixing compartment before the heater drops below the *C.MixAirTempSp* parameter, the alarm is activated. The air temperature must rise by 2ºC above the *C.MixAirTempSp* before an alarm reset is possible.
PreHeatAirTemp	Analog	If a measured value on the air temperature sensor that is installed directly after the heater drops below the *C.PreHeatAirTempSp* parameter, the alarm is activated. The air temperature must rise by 2ºC above the *C.PreHeatAirTempSp* before an alarm reset is possible.
MiscAlmInput	Binary	A high state at the input results in a direct alarm activation. This is used as a miscellaneous alarm indication.

Activation of any alarm condition results in an immediate alarm activation. The block's *Alarm* output is connected to a *FreezeAlarm* binary value defined as a BACnet alarm point. Alarm reset is not possible if any of the conditions are active.

FreezeAlarm AHU Response

Once the *FreezeAlarm* is detected, the signal is passed to the *AHU_OperatingMode* block and the *FreezeAlarm* state is set as the active operating mode which is then propagated to other blocks.

It is then up to all the equipment control blocks to undertake the correct action in response to the *FreezeAlarm* status. The standard reaction is to stop the fans from operating, close the air dampers, start the preheater pump and open its valve to 100%.

The AHU Operating Mode Block (AHU_OperatingMode)

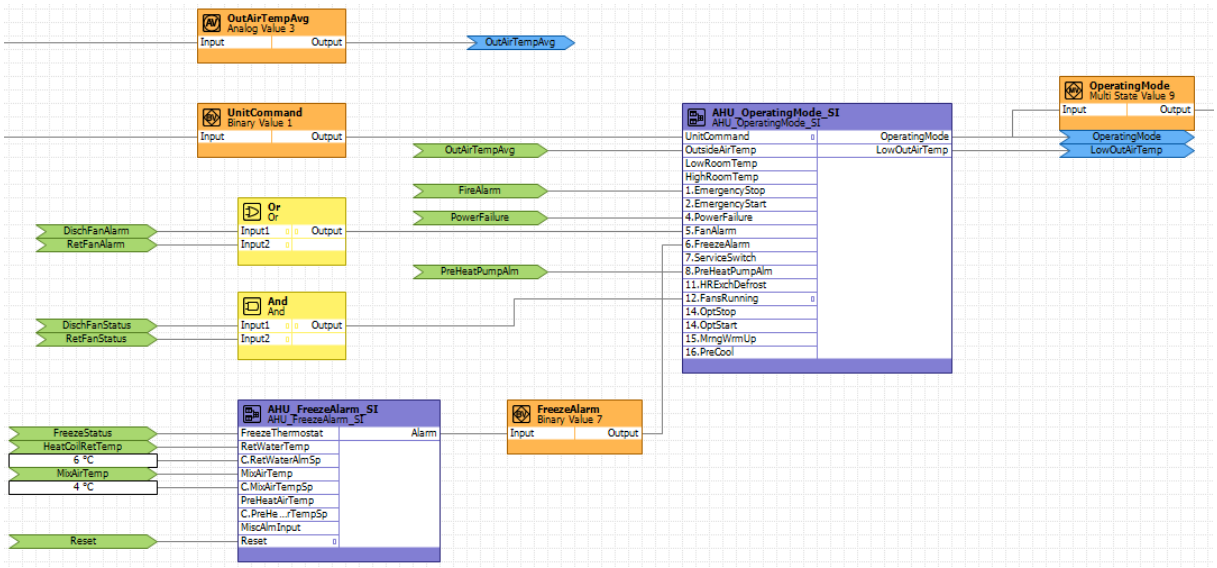


Figure 5: The AHU_OperatingMode programming block

Inputs

The *AHU_OperatingMode* block requires connection of input signals (hardware and logic).

Required Inputs

Input Parameter	Description
UnitCommand	A binary signal holding information on scheduled or manually forced activation of the AHU (BI).
FansRunnning	An operation confirmation signal from the AHU's fans indicating their normal operation (BI).

Optional Inputs

Input Parameter	Description
OutsideAirtTemp	A filtered, average value of the outside air temperature. Filtration time should be between 1-3h. (AI)
LowRoomTemp	The lowest temperature measured in the rooms served by the AHU. (AI)
HighRoomTemp	The highest temperature measured in the rooms served by the AHU. (AI)
1.EmergencyStop	An input activating emergency stop of the AHU. By default, a *FireAlarm*. (BI)
2.EmergencyStart	An input activating emergency start of the AHU. (BI)
4.PowerFailure	An input activating *PowerFailure* mode of the AHU. By default, an input from a phase presence and rotation sensor. (BI)
5.FanAlarm	A common alarm of supply or return fan. An external OR logic is used to connect the alarms. Their selection depends on the fans configuration, (BI)
6.FreezeAlarm	A common alarm indicating freezing danger of the preheating coil. (BI)
7.ServiceSwitch	An input indicating activation of a service switch installed on the AHU, or opening of the AHU service doors. (BI).
8.PreHeatPumpAlm	A preheater pump alarm. (BI).
11.HRExchDefrost	An input indicating need for a heat exchanger to be de-iced. (BI)
12.FansRunning	A common information that all the fans operate normally. (BI)
14.OptStart	An *OptimumStart* active indicator. (BI)
14.OptStop	An *OptimumStop* active indicator. (BI)
15.MrngWrmUp	A *MorningWarmUp* mode is active. (BI)
16.PreCool	A *PreCooling* mode is active. (BI)

Outputs

The *AHU_OperatingMode* block output signals (hardware and logic).

Output Parameter	Description
OperatingMode	An effective operating state of the AHU. (MO)
LowOutAirTemp	An indicator, that the Outside Air Temperature is blow a value at which freezing prevention operation sequences need to be activated. (BO)

Block Functions

OperatingMode Priority Mechanism

The *AHU_OperatingMode* program block checks the status of the unit, verifies the alarm inputs, verifies the operation commands, and then sets an operating state of the unit. This value is used by all the other control blocks to determine their operation not only in relation to their own IO slots, but also to the state of the unit as a whole.

The *OperatingMode* states are structured as follows (lower number means higher priority).

Operating Mode	Description
Emergency overrides section – non-alarm high priority conditions	
1.Emergency Stop	The unit unconditionally stopped (the fans switched off). This mode is activated directly by activation of the *1.EmergencyStop* input. This mode is usually used for a Fire Alarm driven AHU switch off. It is a life safety mode.
2.Emergency Start	The unit unconditionally started (the fans switched on). All the alarms which normally force the unit to be switched off are disregarded. This mode is activated directly by activation of the *2.EmergencyStart* input. This mode is usually used for a Fire Alarm driven AHU emergency operation. In this mode AHU will operate regardless of any alarm even at the expense of physical destruction of the AHU. It is a life safety mode.
3.Reserved	Reserved for future use
Alarms section	
4.Power Failure	The electrical power supply to the unit is cut off. The unit is stopped. This mode is directly activated by a binary signal connected to the *4.PowerFailure* input. The signal usually comes from a phase presence and rotation sensor which shows problems with the power supply of the high current section of the installation (fan motors, pumps, electrical heaters etc.).
5.Fan Alarm	The supply or return fan is malfunctioning. The unit is stopped. This mode is directly activated by a binary signal connected to the *5.FanAlarm* input, to which all the alarms concerning fans are linked via a logical OR block. Typical examples of alarms are: - fan overcurrent - fan belt failure (or fail to start) - variable frequency drive malfunction
6.Freeze Protection	The preheater coil protection mechanism detected a danger of the preheater coil freezing. This mode is directly activated by a binary signal connected to the *6.FreezeProtection* input. When activated, the unit is stopped, a heater pump is started, and a heater valve is opened.
7.Service Switch	A service switch on the unit was activated. This mode is directly activated by a binary signal connected to the *7.ServiceSwitch* input. The fans are immediately stopped. The signal usually comes from: - an axillary contact on a fan service switch - an emergency stop push button - activating switches installed on the AHU's service doors
8.PreHeat Pump Alarm	This mode is activated if a binary signal connected to the *8.PreHeatPumpAlarm* input is active and at the same time the Outside Air Temperature is low enough to create a hazard of the hot water coil freezing (by default below 6ºC). The unit is switched off.
Scheduled operation	
9.Delayed stop	This mode is active during a time defined by a *C.DelayedStopTime* parameter after the *UnitCommand* input changes state from ON to OFF and no alarms are present. When the unit is in this mode, the fans are running to cool down electrical heaters, dissipate cold from direct expansion coolers, dissipate moisture from the humidifier or enable a smooth shut down of any AHU components that do not react well to an instantaneous switch off.
10.Startup	This mode is activated when the *UnitCommand* input changes state from OFF to ON and no alarms are present. The unit enters a start-up sequence. All the AHU components follow a sequence defined by relations and connections between the preheater, damper, and fan programming blocks. When the unit is operational and fans are running (usually a combined signal form supply and return fan pressure switches is used to confirm the fan operation) a state *12.FansRunning* is set.
11.Heat Recovery Defrosting	This mode is directly activated by a binary signal connected to the *11.HeatRecoveryDefrosting* input. This mode is used if the heat recovery systems require defrosting due to a gradual hoarfrost build-up. This mode causes the supply fan to decrease speed while the return fan operates normally, thus increasing the heat exchanger's temperature and causing the heat recovery system to defrost.
12.Running	A normal unit operation. Activated when the *UnitCommand* input changes state from OFF to ON, no alarms are present, and the start-up procedure is completed.
13.Reserved	Reserved for future use
14.Optimized Start/Stop	This mode is activated when a binary signal connected to the *14.OptStart* input is active, the *UnitCommand* input is not active, and no alarms are present or when the *14.OptStop* and the *UnitCommand* inputs are active, and no alarms are present. This mode is used when an optimisation algorithm invokes an optimised start or stop of the AHU.

Operating Mode	Description
15.MrngWarmUp	This mode is activated when a binary signal connected to the *15.MrngWarmUp* input is active, the *UnitCommand* input is not active, and no alarms are present. The unit is started to provide fast warm up of the space after a long unoccupied period.
16.PreCool	This mode is activated when a binary signal connected to the *16.PreCool* input is active, the *UnitCommand* input is not active, and no alarms are present. The unit is started to prepare space to the occupation period by cooling it to a required temperature after a long unoccupied period.
Non-scheduled operation	
17.NightPurge	This mode is activated when the *HighRoomTemp* input is above a *C.NightPurgeRoomTemp* parameter, the OutsideAirTemp is below a *C.NightPurgeOATTemp* parameter, *UnitCommand* input is not active, and no alarms are present. The unit is switched on even though its schedule is in an unoccupied mode, and the outside air temperature is in a range which enables free cooling, and the space temperature is above its setpoint by 2ºC. The mechanical heating and cooling is disabled.
18.NightCycle	This mode is activated if the *LowRoomTemp* is below a *C.NightCycleHeatTemp* or the *HighRoomTemp* input is above a *C.NightCycleCoolTemp* parameter, the *UnitCommand* input is not active, and no alarms are present. The unit is switched on even though its schedule is in an unoccupied mode, because the space temperature drops out of max/min boundaries. The mechanical heating and cooling is enabled.
19.Reserved	Reserved for future use
Unit stopped	
20.Unit Off	The unit is stopped due to lack of an operating command. No alarms are active.

The active *OperatingMode* is decided by the central *AHU_OperatingMode* block. This information is then transferred to all programming blocks and it is up to them to apply the *OperatingMode* to their own logic and subordinate hardware. It means that actual actions corresponding to each of the states may vary depending on the programming sequence and configuration of the system.

The *OperatingMode* with the highest priority always wins, but it does not have to mean a lower priority alarm active at the same time will be ignored. If security action does not interfere with the highest active operating mode procedures, the affected block which is a source of the lower priority alarm may still decide to activate procedures required by it.

LowOutAirTemp Status Signal

The block checks if the *OutsideAirTemp* is below the *C.LowOATempStart* parameter. If so, it activates a *LowOutAirTemp* mode which invokes winter start-up and operation procedures in other sections of the program.

Average OutsideAirTemp Calculation (RollingAverage)

A *RollingAverage* block provides an averaging algorithm based on a first order dumping filter.

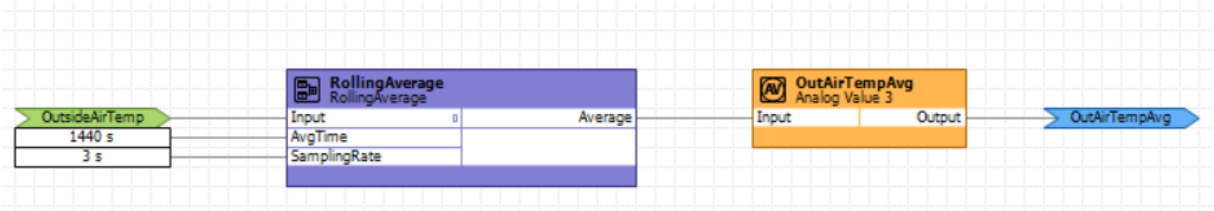


Figure 6: An average OutsideAirTemp calculation

Inputs

The *RollingAverage* block requires connection of input signals (hardware and logic).

Required Inputs

Input Parameter	Description
Input	An input value to be filtered (AO).

Optional Inputs

Input Parameter	Description
AvgTime	A time constant of the filter. Default 300s. (AI)
SamplingRate	A filter execution period. Default 15s (AI)

Outputs

The *RollingAverage* block output signals (hardware and logic).

Output Parameter	Description
Average	A current value of the filter output. (AO)

Block Functions

RollingAverage Operation

The *RollingAverage* block is executed once for each *SamplingRate* number of seconds. It uses the first order dumping filter algorithm, with a time constant equal to the *AvgTime* input. If the block's *Input* changes in a step manner, this type of filter requires five time constants to achieve 99.3% precision at the output. Thus, to achieve a 2h OutsideAirTemp average an *AvgTime* =24m=1440s is used.

In order for the block to operate correctly, the value of *AvgTime* must be significantly higher the value of *SamplingRate*.

The calculated average value is transmitted to the block's *Average* output.

Air Filter Alarms Supervision (DelayedAlarm)

DelayedAlarm blocks are used to generate alarm conditions based on the state of the air filters.

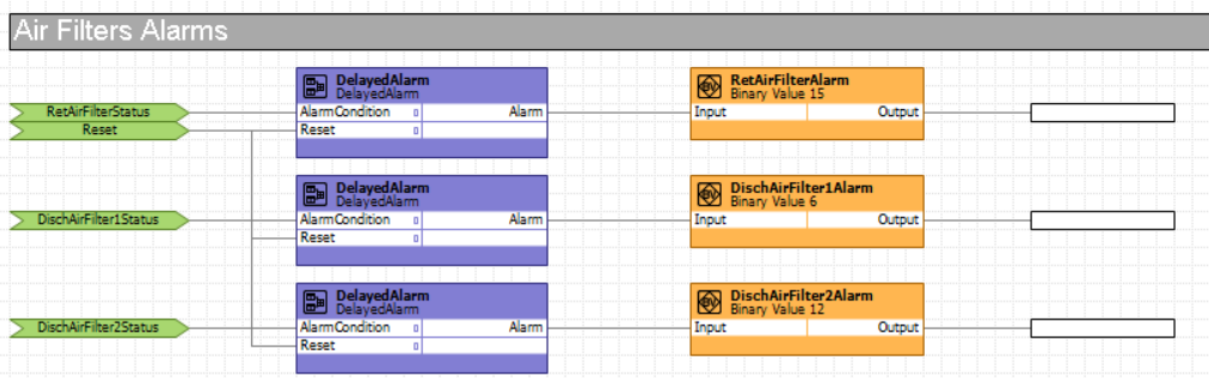


Figure 7: An air filter alarm supervision

Inputs

The *DelayedAlarm* block requires connection of input signals (hardware and logic).

Required Inputs

Input Parameter	Description
AlarmCondition	An alarm activate alarm command (BI).
Reset	An alarm reset command (BI).

Optional Inputs

The *DelayedAlarm* block does not have any optional inputs.

Outputs

The *DelayedAlarm* block output signals (hardware and logic).

Output Parameter	Description
Alarm	An alarm output. (BO)

Block Functions

DelayedAlarm Operation

The *DelayedAlarm* block checks status of the *AlarmCondition* input. If it is active for a time longer than the *C.AlarmDelay* parameter, an alarm status is activated. It is immediately propagated to the *Alarm* output. The *Alarm* state is latched and sustained even when the original *AlarmCondition* drops back to normal. The *Reset* input is used to deactivate the latched alarm, which will only be successful when the *AlarmCondition* is no longer active.

Alarms Reset and Output Active Timer Reset (AutoReset)

AutoReset blocks are used to revert BACnet reset points to an inactive value after a reset action has been executed.

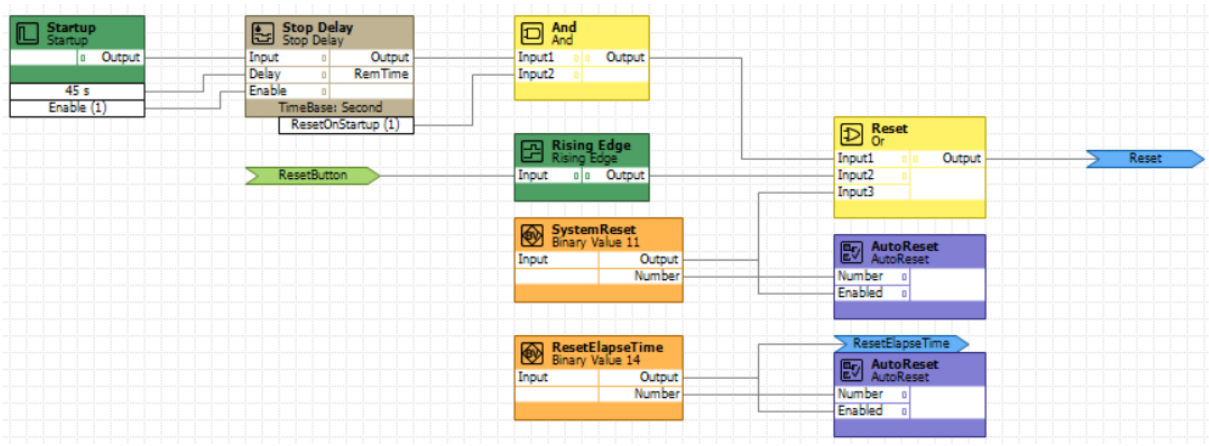


Figure 8: Auto reset blocks

Inputs

The *AutoReset* block requires connection of input signals (hardware and logic).

Required Inputs

Input Parameter	Description
Number	A binary variable reference number of the BACnet reset point. (BI)
Enable	A BACnet reset point deactivation needed trigger. (BI).

Optional Inputs

The *AutoReset* block does not have any optional inputs.

Outputs

The block does not provide any output signals (hardware nor logic).

Block Functions

Alarms Reset Operation

AHU Alarms can be reset by one of three events:

- During the first 45s after the controller start-up.
- By a rising edge of a Hardware Input *ResetButton*, which usually is connected to a pushbutton installed on the control switchboard.
- By activation of a *SystemReset* BACnet binary variable.

AutoReset Operation

When a BMS system operator activates the *SystemReset* BACnet binary variable (at any priority), the *AutoReset* block is activated. It takes the *Number* input which holds a reference number of the BACnet variable, and then using this reference, it invalidates all the priorities higher than 16 (by writing a *null* value) and assigns a *false* value to the priority 16, thus effectively providing an auto toggle function each time the *SystemReset* variable is activated.

ResetElapseTime Operation

When a BMS system operator activates the *ResetElapseTime* BACnet binary variable (at any priority), the *AutoReset* block is activated. It takes the *Number* input which holds a reference number of the BACnet variable, and then using this reference, it invalidates all the priorities higher than 16 (by writing a *null* value) and assigns a false value to the priority 16, thus effectively providing an auto toggle function each time the *ResetElapseTime* variable is activated.

The *ResetElapseTime* variable is used to reset internal activity timers of Hardware Outputs.

Temperature and Humidity Control – EquipmentControl Page

An *EquipemntControl* programming page provides algorithms for temperature and humidity control. It is divided into two sections *Temperature Regulation* and *Humidity regulation*.

Programming blocks can be divided into two major categories – general regulation and equipment related. The general regulation blocks take sensor input information along with setpoints, apply PID control, and set heating/cooling demand signals. These signals are then transferred to the equipment control blocks, which are adjusted to fit the mechanical configuration of AHU components. They set hardware outputs to provide optimal operation of AHU sections to satisfy the calculated demand.

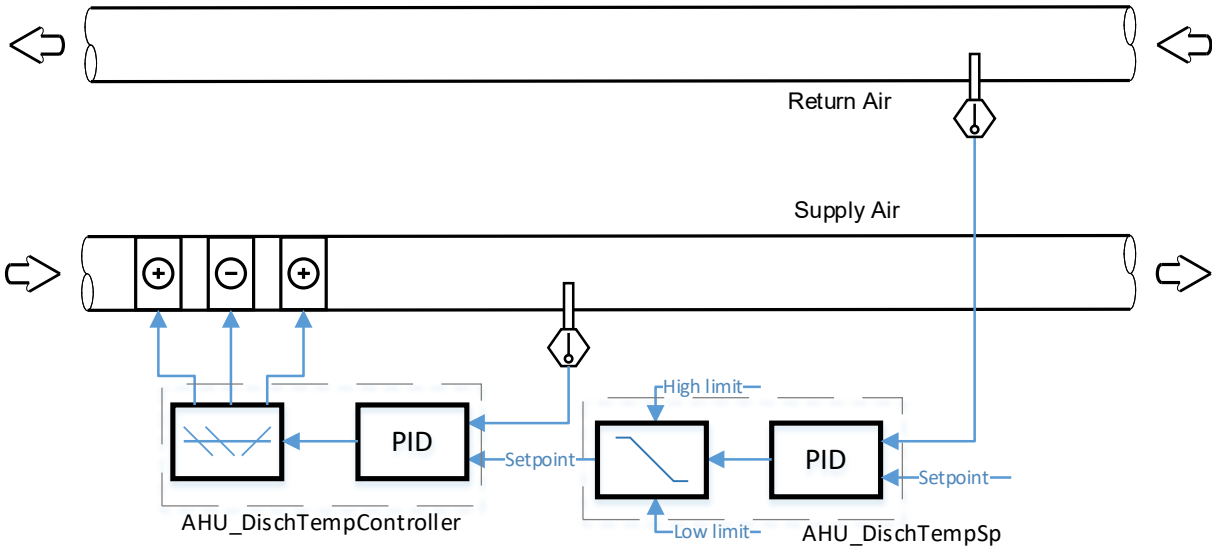


Figure 9: The AHU_DischTempSp and AHU_DischTempController cascade controller operation scheme

Discharge Air Temperature Setpoint Calculation (AHU_DischTempSp_SI)

An *AHU_DischTempSp_SI* block provides a discharge air temperature setpoint. It can be based on space temperature (room temperature or return temperature), outside air temperature compensation, or it can directly use a *SpaceSetpoint* input value as a *DischTempSp* output.

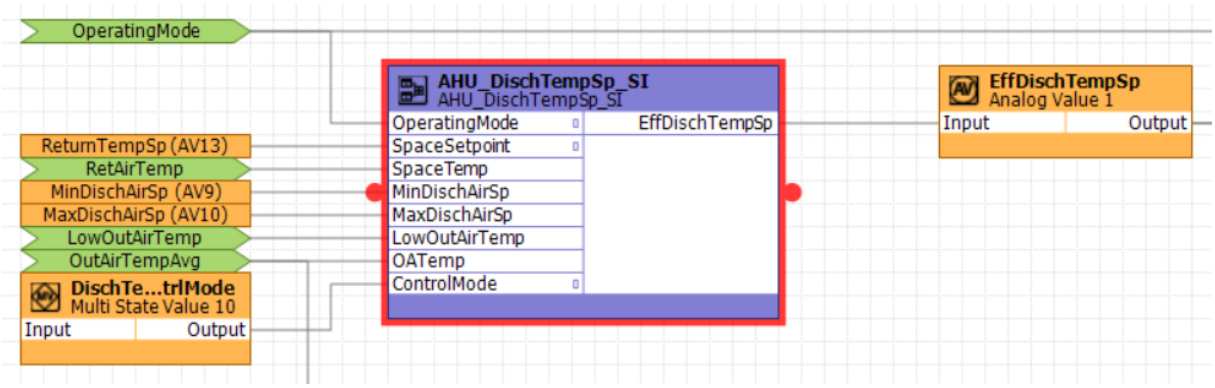


Figure 10: The AHU_DischTempSp_SI programming block

Inputs

The *AHU_DischTempSp_SI* block requires connection of input signals (hardware and logic).

Required Inputs

Input Parameter	Description
OperatingMode	A multistate value containing coded status of the AHU. (MI)
SpaceSetpoint	A space/return temperature setpoint value. (AI)
ControlMode	A multistate value containing coded operation mode of the block. (MI)

Optional Inputs

Input Parameter	Description
SpaceTemp	A space/return air temperature value. Default null (AI)
MinDischAirSp	A low limit for the DischTempSp value. Default 16ºC (AI)
MaxDischAirSp	A high limit for the DischTempSp value. Default 32ºC (AI)
LowOutAirTemp	A binary input signalling that AHU operates in winter conditions and freezing alarm prevention needs to be activated. Default false(BI)
OATemp	An outside air temperature value. Default 0ºC (AI)

Outputs

The *AHU_DischTempSp_SI* block output signals (hardware and logic).

Output Parameter	Description
EffDischTempSp	An effective value of the Discharge Air Temperature Setpoint. (AO)

Block Functions

ContolMode Selection

The *ControlMode* multistate input defines the discharge air temperature setpoint calculation method.

Control Mode	Action
1 – Cascade	The *SpaceTemp* to the *DischTempSp* cascade control.
2 – Direct	The *SpaceSetpoint* is directly used as the *DischTempSp*.
3 – OAT Compensated	The *DischTempSp* is based on an outside air temperature compensated characteristic.

Casdade DischTempSp Calculation Algorithm

When the *ControlMode* input is set to “1 – Cascade”, the block checks if the *SpaceTemp* input has a valid value. If the input is valid, then a PID regulator is used to calculate the discharge air temperature setpoint as a function of the *SpaceTemp* and the *SpaceSetpoint* values. The setpoint will be limited by *MinDischAirSp* and *MaxDischAirSp* parameters.

If the input is invalid, the setpoint will be set to an average of *MinDischAirSp* and *MaxDischAirSp*.

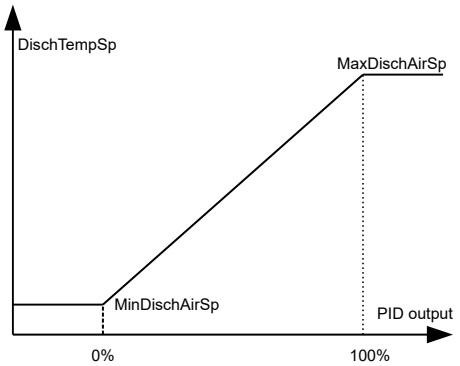


Figure 11: The cascade control of the DischTempSp

Direct DischTempSp Calculation Algorithm

If the *ControlMode* input is set to “2 – Direct”, the block transfers the *SpaceSetpoint* directly to the *DischTempSp* output value.

Outside Air Temperature Compensated DischTempSp Calculation Algorithm

When the *ControlMode* input is set to “3 – OAT Compensated”, the block starts calculation by checking if the current *OATemp* value is equal to the *SpaceSetpoint.* If it is, then *DischTempSp* is equal to the *SpaceSetpoint.*

If the *OATemp* is lower than the *SpaceSetpoint*, then the *DischTempSp* is increased by the *C.OATCompRatio* parameter (default value = 0.5ºC) for each 1ºC of the *OATemp* drop, until it reaches the *MaxDischAirSp*.

If the *OATemp* is higher than the *SpaceSetpoint*, then the *DischTempSp* is decreased by the *C.OATCompRatio* parameter for each 1ºC of the *OATemp* rise, until it drops to the *MinDischAirSp*.

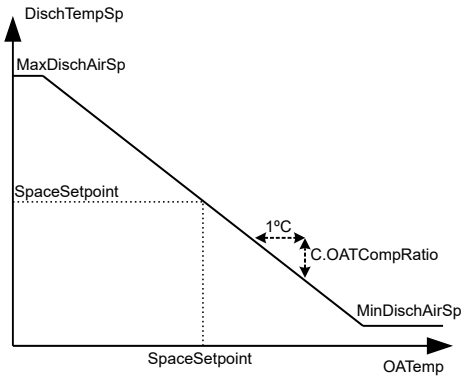


Figure 12: OATemp compensated control of the DischTempSp

LowOutAirTemp Operation

When the AHU is started in winter conditions, the program activates the freeze alarm prevention mechanism. The calculated *DistchTempSp* is initially increased by the *C.StartupSetpoint* parameter (default value = 15ºC), then it is gradually decreased to reach normal value provided by the block’s logic after a time defined by the *C.StartupTime* parameter (default value = 900s).

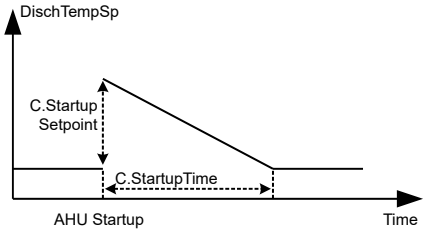


Figure 13: AHU Startup

The Discharge Air Temperature Controller (AHU_DischTempController_SI)

An *AHU_DischTempController_SI* provides load signals based on the *DischTemp* PID regulation algorithm for all the temperature control components of the AHU. These load signals are not used to directly operate hardware outputs, since they lack functions related to actual hardware configuration of the AHU (such as type of heater/cooler, type of valves, presence of pump etc.).

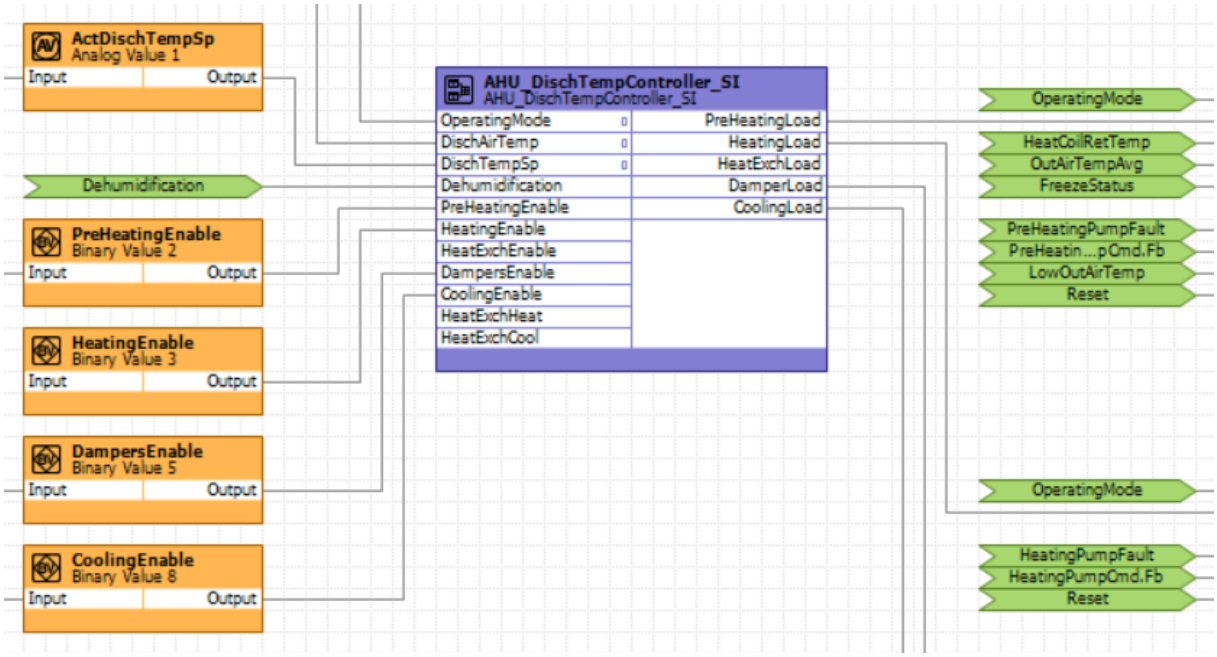


Figure 14: The AHU_DischTempController_SI programming block

Inputs

The *AHU_DischTempController_SI* block requires connection of input signals (hardware and logic).

Required Inputs

Input Parameter	Description
OperatingMode	A multistate value containing coded status of the AHU. (MI)
DischAirTemp	A discharge temperature value. (AI)
DischTempSp	A discharge temperature setpoint value. (AI)

Optional Inputs

Input Parameter	Description
Dehumidification	An interface signal from an *AHU_HumidityControl* block that is used to coordinate air dehumidification process. Default 0%. (AI)
HeatExchHeat	A heat exchanger winter mode activation. Default *true.* (BI)
HeatExchCool	A heat exchanger summer mode activation. Default *false.* (BI)
CoolingEnable	A cooling action activation signal. Default *true.*(BI)
HeatExchEnable	A heat exchanger activation signal. Default *true.*(BI)
DampersEnable	A recirculation dampers activation signal. Default *true.*(BI)
PreHeatingEnable	A preheater activation signal. Default *true.*(BI)
HeatingEnable	A reheater activation signal. Default *true.*(BI)

Outputs

The *AHU_DischTempController_SI* block output signals (hardware and logic).

Output Parameter	Description
PreHeatingLoad	The preheater load signal used to drive a preheater control block. (AO)
HeatingLoad	The reheater load signal used to drive a reheater control block. (AO)
HeatExchLoad	The heat exchanger load signal used to drive a heat exchanger control block. (AO)
DamperLoad	The damper load signal used to drive a damper control block. (AO)
CoolingLoad	The cooling load signal used to drive a cooler control block. (AO)

Block Functions

Temperature Control

The *AHU_DischTempController_SI* block uses two independent PIDs, and a set of Enable inputs to define an effective temperature regulation sequence.

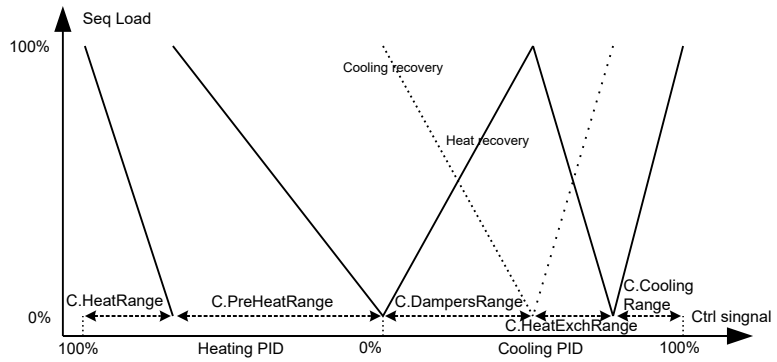


Figure 15: The heating and cooling PID outputs to load outputs sequence

When the AHU starts up, the *DischAirTemp* input is compared to the *DischTempSp*. If the measured temperature (*DischAirTemp*) is below the setpoint (*DischTempSp*), the heating PID will be activated. If the temperature (*DischAirTemp*) is above the setpoint (*DischTempSp*), the cooling PID will be activated. Activation of either the heating PID or cooling PID is interlocked with each other to prevent simultaneous operation of both regulators.

The block uses internal parameters to adjust configuration of the system to the hardware configuration of the AHU. *C.PreHeatConfig, C.HeatConfig, C.HeatExchConfig, C.DampersConfig,*and *C.CoolingConfig* parameters are used to enable/disable the preheater, reheater, heat exchanger, recirculation dampers and cooling coil operation. Disabled components are excluded from all the calculations and PID regulation.

The heating PID is responsible for operating the preheater and reheater sequences. *C.PreHeatRange* and *C.HeatRange* parameters define which portion of the heating PID output signal is used to drive the given sequence. Similarly, the cooling PID signal is divided based on *C.DampersRange*, *C.HeatExchRange,* and *C.CoolingRange* values. The algorithm checks which sequences are enabled, then recalculates all the effective ranges so that they can cover the entire PID output range. This way the PID signal is only transferred to the active sequences without any dead zones or gaps, providing a smooth regulation.

Preheater and Heater Control

The heating PID drives two sequences. The preheater which is a basic heating system of the AHU and the reheater which is a secondary heating system.

The preheater is usually much bigger than the reheater because its main function is to provide the heating during winter. The reheater is used mostly during summer when a main heat providing plant is disabled, or when need for dehumidification arises. The preheater is by default deactivated when the *OutsideAirTemp* rises above a defined value (by default 20ºC). Then entire heating PID signal is transferred to the reheater which is usually based on an electrical heater or technological hot water system, which is operational even in the summertime.

Dehumidification

The dehumidification process requires the air passing through the AHU to be cooled below a dew point temperature to force condensation. It can be achieved either by a specially dedicated air drier (usually based on a direct expansion heat pump which comes with its own dedicated controller), or by a combined operation of the preheater, cooling coil, and reheater. The latter process is supported by the *AHU_DischTempController* block.

Firstly, the preheater is disabled to avoid an unnecessary heating up of the air which needs to be cooled down below the dew point. The entire heat demand signal is transferred to the reheater, which is located after the cooling coil, so its operation would not jeopardize the dehumidification process. Because of this, the dehumidification must be blocked in the wintertime, when usage of the preheater is essential.

A cooling coil control block (*AHU_CoolingControl*), which receives a dehumidification demand signal from a humidity control block, combines this signal with the cooling load signal coming from the *AHU_DischTempController* block and uses the higher value to operate a cooling valve. This way it is certain that the dehumidification process has a higher priority than the cooling action.

The additional aperture of the cooling valve is likely to disturb the temperature control. The *AHU_DischTempController* block reacts to the discharge air temperature drop and increases the heat demand signal, which is entirely transferred to the reheater, thus enabling heating up of already dehumidified air without jeopardizing the dehumidification process.

Recirculation Dampers, Heat Exchanger, and Cooling Coil Control

The cooling PID operates three sequences: the recirculation dampers, the heat exchanger and the cooling coil. The control system is configured in this way to ensure that if the AHU is started in the winter conditions, heat recovery equipment will be activated at full capacity and its preheater is modulated to provide a heating energy necessary to achieve the discharge temperature setpoint. This approach limits a risk of a freeze alarm activation at the AHU start-up to a minimum.

When cooling load demand increases, the recirculation dampers react first by increasing a volume of the outside air which enters the AHU. When the unit operates at 100% of the outside air, the heat exchange ratio is decreased to limit heat recovery from the return air stream. When the heat recovery is set to 0% and the cooling demand is not satisfied, the system will activate the cooling coil.

The cooling coil will be activated when the *OutsideAirTemp* is above a defined value (by default 15ºC). Below this temperature the cold outside air should be sufficient to bring the supply temperature down.

Heat Versus Cold Recovery

The *AHU_DischTempController_SI* block uses two inputs, *HeatExchHeat* and *HeatExchCool*, to determine if heat recovery equipment should be used as a function of heat or cold recovery. The heat vs cold recovery decision making is external to the block itself, but in the standard application program, the heat recovery is activated when the *RetAirTemp* is higher than the *OutAirTemp* by at least 3ºC, then switched off when the *RetAirTemp* drops below the *OutAirTemp*. The cold recovery is activated when the *RetAirTemp* is lower than the *OutAirTemp* by at least 3ºC, then switched off when the *RetAirTemp* rises above the *OutAirTemp*.

The Main Heater Control Block (AHU_PreHeatingControl_SI)

An *AHU_PreHeatingControl_SI* block is designed to operate hot water heating coils in regions where winters with temperatures below freezing occur. In its maximum configuration, it can operate a heater equipped with a heating valve, a heating pump, a freeze thermostat, a return water temperature sensor, and a preheater air temperature sensor.

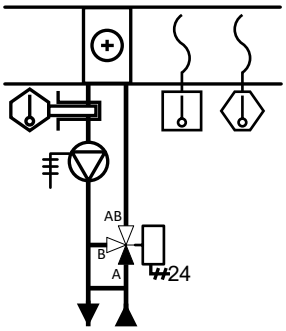


Figure 16: The preheater hardware configuration

The block transfers the *PreHeatingLoad* signal from the *AHU_DischTempController_SI* block to hardware outputs controlling operation of the hardware preheater components. The block provides a secure start-up in the wintertime and reliable operation throughout the whole year.

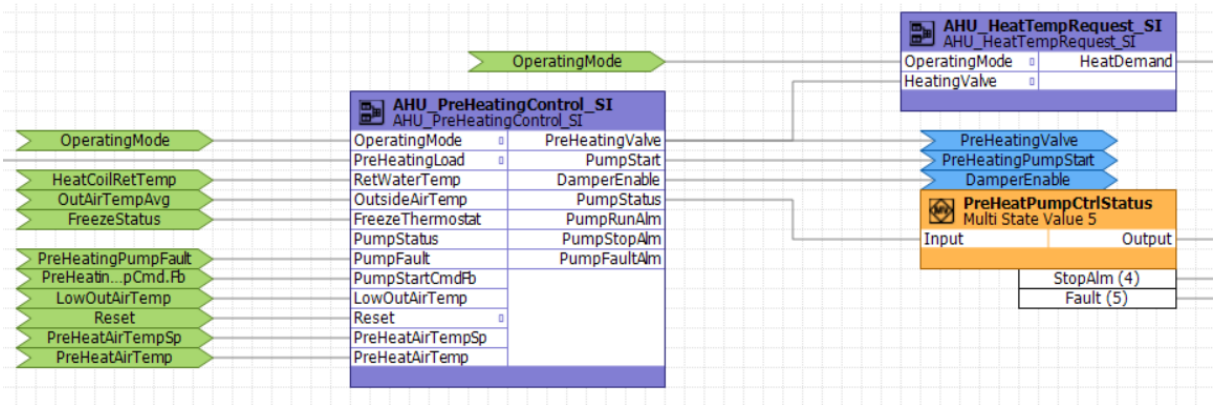


Figure 17: The AHU_PreHeatingControl_SI programming block

Inputs

The *AHU_PreHeatingControl_SI* block requires connection of input signals (hardware and logic).

Required Inputs

Input Parameter	Description
OperatingMode	A multistate value containing coded status of the AHU. (MI)
PreHeatingLoad	A control command from the *AHU_DischTempController_SI* block. (AI)
FreezeThermostat	A binary input from a freezing protection thermostat. (BI)
Reset	An alarm reset command (BI).

Optional Inputs

Input Parameter	Description
RetWaterTemp	A return water temperature value. Default *100°C.* (AI)
OutsideAirTemp	An outside air temperature value. Default *15°C.* (AI)
PumpStatus	Confirms operation of the pump (feedback from a contactor or a flow switch). Do not use simultaneously with the PumpStartCmdFb. Default *null.* (BI)
PumpStartCmdFb	A confirmation signal from the output of the controller that it is energized. Do not use simultaneously with the PumpStatus. Default *null.*(BI)
Pump Fault	A fault signal of the pump. Default *false.*(BI)
LowOutAirTemp	A binary input signalling that AHU operates in the winter conditions and freezing alarm prevention needs to be activated. Default *false.* (BI)
PreHeatAirTemp	A supply air after the preheater temperature value. Default *null.* (AI)
PreHeatAirTempSp	A supply air after the preheater temperature setpoint. Default *8°C.* (AI)

Outputs

The *AHU_PreHeatingControl_SI* block output signals (hardware and logic).

Output Parameter	Description
PreHeatingValve	A valve control signal. (AO)
PumpStart	A start signal for the heater pump. (BO)
PumpStatus	A multistate PumpStatus point. (MO)
PumpRunAlm	A fail to stop alarm active signal. (BO)
PumpStopAlm	A fail to start alarm active signal. (BO)
PumpFaultAlm	A direct fault alarm active signal. (BO)

Block Functions

Heating Valve Operation

When the AHU is operational, the *AHU_PreHeatingControl_SI* block takes the *PreHeatingLoad* input signal and passes it through a linear ratio block. By default, this linear conversion is scaled 0% -> 0%, 100% -> 100%, but if a need arises it can be used to drive multiple heaters with the same *PreHeatingLoad* signal. If multiple heaters need to be driven by the same *PreHeatingLoad* signal, an additional *AHU_PreHeatingControl_SI* block, with modifications of the ratio block settings in both blocks, are required.

The block then checks if the *PreHeatAirTemp* or the *RetWaterTemp* regulation algorithm increases the minimum opening of the heating valve. If the valve is opened for more than 5% or a *LowOutAirTemp* or *FreezeThermostat* are active, then the circulation pump is started.

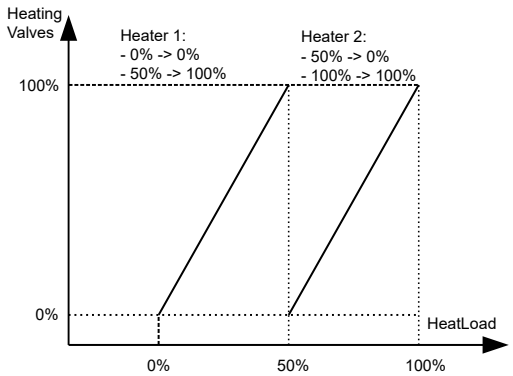


Figure 18: The ratio mechanism used to drive two heaters

Pump Operation and Alarm

The pump is started if the valve is opened more than 5% or if a *LowOutAirTemp* or *FreezeThermostat* are active. A start delay of 5s and stop delay of 300s is applied to pump operation command.

Depending on the connected status signals, *PumpStatus, PumpStartCmdFb* and *PumpFault*, alarms can be generated:

- *PumpFault* is treated as a direct fault signal and the pump alarm will be generated immediately when *PumpFaul*t input is true.
- *PumpStartCmdFb* and *PumpStatus* are treated as pump operation feedback signals. *PumpStatus* has priority over *PumpStartCmdFb* and if both are connected, the latter is ignored. If feedback fails to follow the *PumpCmd* signal, *RunAlarm* (fail to stop) and *StopAlarm* (fail to start) signals will be generated (with corresponding *C.RunAlmDelay* and *C.StopAlmDelay* times).

The algorithm outputs a *PumpStatus*. This multistate output signal provides a coded *PumpStatus*, with the following values:

Value	Description
1	Stop
2	Start
3	Run Alarm (fail to stop)
4	Stop Alarm (fail to start)
5	Fault Alarm

If the parameter *C.DisCmdInAlarm* is *true*, the pump command will be disabled when the pump trips into the Stop Alarm or Fault Alarm.

RetWaterTemp Control

When the optional *RetWaterTemp* sensor is present, the system will check the *C.RWTControlMode* parameter value to define the operation mode of the return water regulation. It can be configured as:

Parameter	Description
0 – Off	Regulation disabled.
1 – Only when AHU is Off	Regulation is enabled only when the AHU is stopped.
2 – Only when AHU is On	Regulation is enabled only when the AHU is started.
3 – All the time	Regulation is always enabled.

If regulation is activated, the block will calculate an *OutsideAirTemp* based setpoint for the *RetWaterTemp* (if the *OutsideAirTemp* is not available, the setpoint is fixed to 20ºC). Then a dedicated PID algorithm is used to calculate a *RetWaterTemp* based demand. This demand is compared against a demand based on the *PreHeatAirTemp* and the higher value of the two is used as a minimum valve opening.

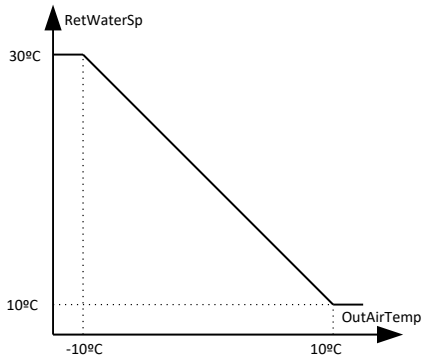
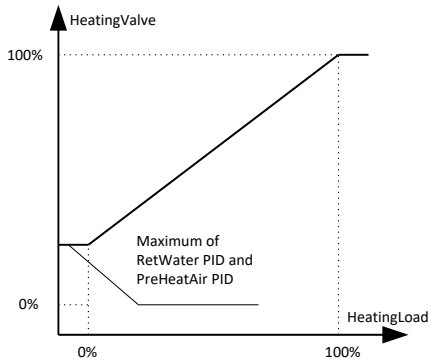


Figure 19: Left: RetWaterSp calculation.



Right: The HeatingLoad to HeatingValve recalculation

PreHeatAirTemp Control

If the optional *PreHeatAirTemp* sensor is present and the AHU is running, the block will use a dedicated PID algorithm to calculate a *PreHeatAirTemp* based demand. The setpoint is set by the *PreHeatAirTempSp.* This demand is compared against the demand based on the *RetWaterTemp,* and the higher value of the two is used as a minimum valve opening.

AHU Start-Up Procedure

When the AHU starts up, the *LowOutAirTemp* input indicates winter conditions, and if the *RetWaterTemp* sensor is present, a *PreHeat* mode will be activated. It will increase the *RetWaterSp* by 15ºC and activate the *RetWaterTemp* PID algorithm to force the opening of the *HeatingValve* and preheating of the heating coil. When the *RetWaterTemp* reaches the original *RetWaterSp* + 10ºC, the *PreHeat* mode is deactivated and the *DamperEnable* output signal is activated to enable opening of the dampers and starting of the AHU fans. This way when the fans begin to retrieve freezing cold outside air, it will come in contact with the heater coil already filled with hot water, thus limiting a risk of the *FreezeAlarm* activation.

FreezeAlarm Operation

If the *FreezeThermostat* input is activated, the block will immediately open the *HeatingValve*, start the pump, and block the *DamperEnable* signal. The same signal should be connected to the *AHU_FreezeAlarm_SI* block which activates *FreezeAlarm* and connected to the *AHU_OperatingMode_SI* block to activate the *OperatingMode – 6 – FreezeAlarm*, thus stopping operation of the entire AHU.

Once the *FreezeThermostat* input goes back to its normal state, the valve is closed. This is done, because if the valves were left open for the duration of the *FreezeAlarm*, it would cause overheating of the AHU interior to temperatures almost as high as a hot water supply temperature which would lead to important complications and instabilities during a *FreezeAlarm* recovery restart, often resulting in a *FreezeAlarm* reactivation.

If the *RetWaterTemp* sensor is present, it is used to control water temperature and ensure required valve opening to prevent the *FreezeAlarm* activation. To achieve this, *RetWaterTemp* control needs be enabled when the unit is stopped or all the time (the AHU stopped and running).

The Reheater Control Block (AHU_HeatingControl_SI)

An *AHU_HeatingControl_SI* block transfers the *HeatingLoad* signal from the *AHU_DischTempController_SI* block to hardware outputs controlling the operation of the actual reheater components, providing all the necessary functions resulting from the hardware configuration of the AHU.

The block is designed to operate hot water heating coils or electric heaters. It can operate a heater equipped with a heating valve and a heating pump or an electric heater with a current valve and power supplied via contactor. Since this block provides control for a reheater, it is not designed to cope with a risk of coil freezing. If such a danger exists, the *AHU_PreHeatingControl_SI* block must be used.

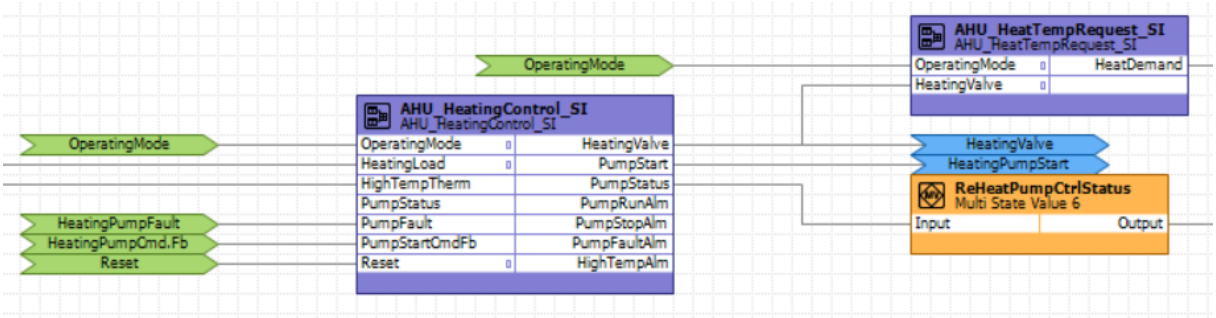


Figure 20: The AHU_HeatingControl_SI programming block

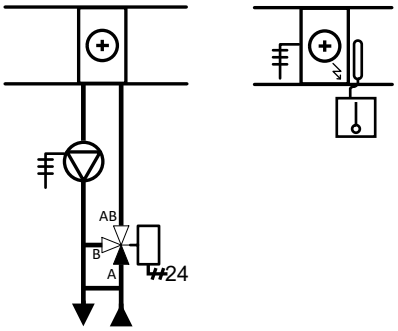


Figure 21: The reheater hardware configuration

Inputs

The *AHU_HeatingControl_SI* block requires connection of input signals (hardware and logic).

Required Inputs

Input Parameter	Description
OperatingMode	A multistate value containing coded status of the AHU. (MI)
HeatingLoad	A control command from the *AHU_DischTempController_SI* block. (AI)
Reset	An alarm reset command (BI).

Optional Inputs

Input Parameter	Description
HighTempTherm	An input from a high temperature thermostat protecting the electric heater. Default *null*. (BI)
PumpStatus	Confirms operation of the pump (feedback from a contactor or a flow switch). Do not use simultaneously with the PumpStartCmdFb. Default *null*. (BI)
PumpStartCmdFb	A confirmation signal from the output of the controller that it is energized. Do not use simultaneously with the *PumpStatus*. Default *null*.(BI)
Pump Fault	A fault signal of the pump. Default *false*.(BI)

Outputs

The *AHU_HeatingControl_SI* block output signals (hardware and logic).

Output Parameter	Description
HeatingValve	A valve control signal. If the block is used to control an electric heater, this output controls a current valve. (AO)
PumpStart	A start signal for a heater pump. If the block is used to control an electric heater, this output connects power supply. (BO)
PumpStatus	A multistate pump/electric power supply status point. (MO)
PumpRunAlm	A fail to stop alarm active signal. (BO)
PumpStopAlm	A fail to start alarm active signal. (BO)
PumpFaultAlm	A direct fault alarm active signal. (BO)
HighTempAlm	A high temperature alarm of the electric heater. (BO)

Block Functions

Description of the block's functions is based on the hot water heater operation. An adaptation to the electric heater operation is explained in the final paragraph of this block's description.

Heating Valve Operation

When the AHU is operational, the *AHU_HeatingControl_SI* block takes the *HeatingLoad* input signal and passes it through a linear ratio block. By default, this linear conversion is scaled 0% -> 0%, 100% -> 100%, but if a need arises it can be used to drive multiple heaters with the same *HeatingLoad* signal. If multiple heaters need to be driven by the same *HeatingLoad* signal, an additional *AHU_HeatingControl_SI* block, with modifications of the ratio block settings in both blocks, are required.

If the valve is opened more than 5%, then the circulation pump is started.

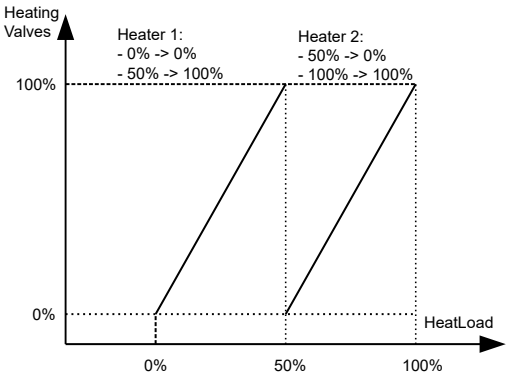


Figure 22: The ratio mechanism used to drive two heaters

Pump Operation and Alarm

The pump is started if the valve is opened more than 5%. A start delay of 5s and stop delay of 300s is applied to the pump operation command.

Depending on the connected status signals, *PumpStatus, PumpStartCmdFb* and *PumpFault*, alarms can be generated:

- *PumpFault* is treated as a direct fault signal and the pump alarm will be generated immediately when *PumpFault* input is true.
- *PumpStartCmdFb* and *PumpStatus* are treated as pump operation feedback signals. *PumpStatus* has priority over *PumpStartCmdFb* and if both are connected, the latter is ignored. If feedback fails to follow the *PumpCmd* signal, *RunAlarm* (fail to stop) and *StopAlarm* (fail to start) signals will be generated (with corresponding *C.RunAlmDelay* and *C.StopAlmDelay* times).

The algorithm outputs a *PumpStatus*. This multistate output signal provides a coded *PumpStatus*, with the following values:

Value	Description
1	Stop
2	Start
3	Run Alarm (fail to stop)
4	Stop Alarm (fail to start)
5	Fault Alarm

If the parameter *C.DisCmdInAlarm* is *true*, the pump command will be disabled when the pump trips into the Stop Alarm or Fault Alarm.

HighTempTherm and Electric Heater Operation

If the *HighTempTherm* input is connected to a valid value (not null), the block goes into electric heater mode.

In this mode, the pump output is used to switch electric power supply. And therefore switch-off delay time is changed from 300s to 0s.

If the *HighTempTherm* is active, the *HeatingValve* and *PumpStart* outputs are immediately deactivated to stop the heater operation. At the same time, the *HighTempAlm* is activated and the alarm is latched to prevent the restart of the heater. In the electric heater mode, deactivation of the *C.DisCmdInAlarm* is ignored – any alarm of the heater power supply results in the heater being switched off.

The Heat Exchanger Block (AHU_HeatExchControl_SI)

An *AHU_HeatExchControl_SI* block transfers the *HeatExchLoad* signal from the *AHU_DischTempController_SI* block to the hardware outputs controlling the operation of the actual heat exchanger components. This process provides all the necessary functions required by the hardware configuration of the AHU.

The block can be used to operate a thermal wheel, air-liquid-air, plate or heat-pipe heat exchanger (for the latter the block needs to be configured as the plate exchanger). It provides control of a rotor/pump motor operation along with a rotation speed/valve/damper opening and icing protection via a pressure status or an exhaust air temperature sensor.

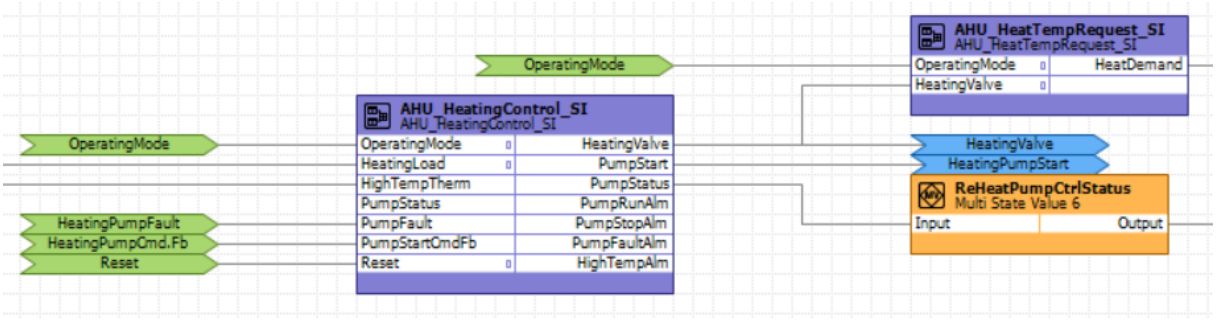


Figure 23: AHU_HeatingControl_SI block

Inputs

The *AHU_HeatExchControl_SI* block requires connection of input signals (hardware and logic).

Required Inputs

Input Parameter	Description
OperatingMode	A multistate value containing coded status of the AHU. (MI)
HeatExchLoad	A control command from the *AHU_DischTempController_SI* block. (AI)
Reset	An alarm reset command (BI).

Optional Inputs

Input Parameter	Description
ExhaustAirTemp	An input from an exhaust air temperature sensor. Default *null.* (BI)
PressSwitch	A pressure switch input, indicating too high pressure drop on the heat exchanger in the exhaust air stream. Default *false.*(BI)
HeatExchStatus	Confirms operation of the rotor motor/pump (feedback from a contactor or a flow switch). Do not use simultaneously with HeatExchEnableFb. Default *null.*(BI)
HeatExchEnableFb	A confirmation signal from the output of the controller that it is energized. Do not use simultaneously with HeatExchStatus. Default *null.*(BI)
HeatExchFault	A fault signal of the rotor motor/pump. Default *false.*(BI)

Outputs

The *AHU_HeatExchControl_SI* block output signals (hardware and logic).

Output Parameter	Description
HeatExchCtrl	A heat exchanger control signal (rotation speed / valve / damper opening). (AO)
HeatExchEnable	A start signal for the rotor motor / pump. (BO)
HeatExchStatus	A multistate rotor motor / pump status point. (MO)
HeatExchRunAlm	A fail to stop alarm active signal. (BO)
HeatExchStopAlm	A fail to start alarm active signal. (BO)
HeatExchFaultAlm	A direct fault alarm active signal. (BO)

Block Functions

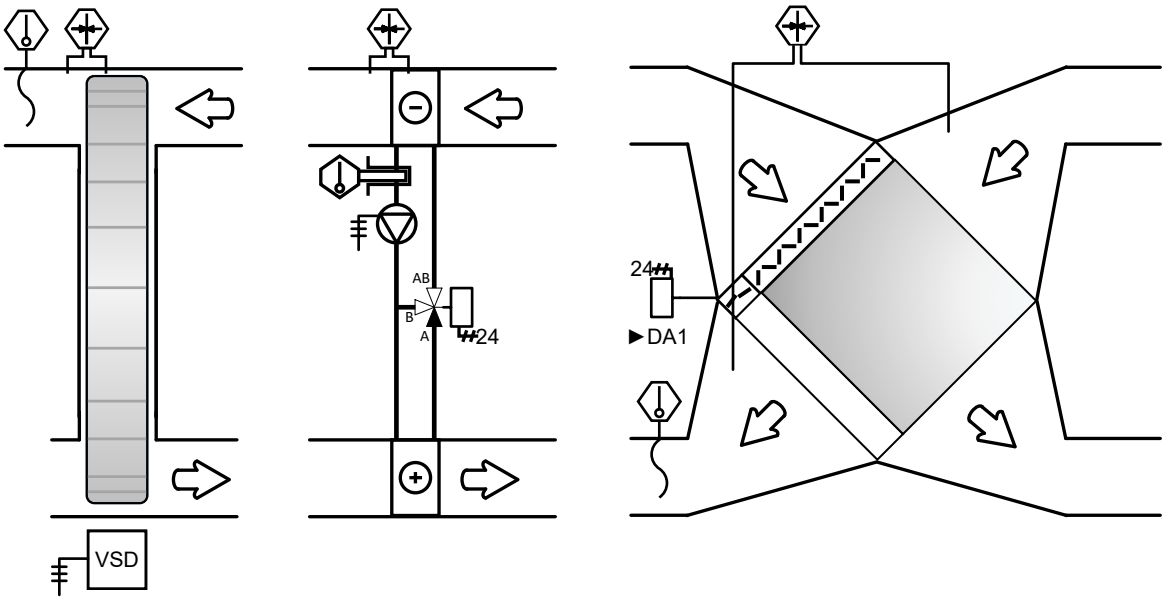


Figure 24: The heat exchanger hardware configuration (thermal wheel, air-liquid-air, plate exchanger)

Heat Exchanger Types

The *AHU_HeatExchControl_SI* block can be used to drive:

Thermal Wheel Heat Exchanger

Standard control equipment consists of a VSD driven electric motor used to rotate a thermal wheel rotor. The control is executed by a variation of the motor speed. Additional equipment is used to protect the rotor from icing which consist of the *ExhaustAirTemp* and the *PressSwitch*.

A thermal wheel rotation speed to heat output characteristic of the heat exchanger is strongly non-linear. This nonlinearity is wheel model specific and should be programmed by a thermal wheel manufacturer into the rotor's VSD. If a standard VSD is used, which lacks this manufacturer preprogramed feature, an additional linearization is required.

Air-Liquid-Air Heat Exchanger

Glycol or water can be used as the liquid depending on the winter temperatures. Standard control equipment consists of a pump and a three-way valve (variations with a VSD driven pump are also possible). Control is executed by the valve opening. Additional equipment is used to protect the heat exchanger from icing. It consists of the water temperature sensor (connected in place of the *ExhaustAirTemp*) and the *PressSwitch*.

Plate Heat Exchanger

Standard control equipment consists of a bypass damper actuator. It allows regulation of the portion of supply air that passes through the exchanger as compared to the portion that goes through the bypass. Additional equipment is used to protect the exchanger from icing. It consists of the *ExhaustAirTemp* and the *PressSwitch*.

The plate heat exchanger mode can be also used to operate a heat pipe exchanger. This type of heat exchanger is not very common. It is mostly used in health care and pharmaceutical applications.

Heat Exchanger Control

When the AHU is operational, the *AHU_HeatExchControl_SI* block takes the *HeatExchLoad* input signal and passes it through a linear ratio block. By default, this linear conversion is scaled 0% -> 0%, 100% -> 100%, but if need arises, it can be used to drive multiple heat exchangers with the same *HeatExchLoad* signal. Driving multiple heat exchangers with the same *HeatExchLoad* signal requires an additional *AHU_HeatExchControl_SI* block and modification of the ratio block settings in both blocks.

If the heat exchanger control signal is higher than 5% then the heat exchanger enable signal is activated.

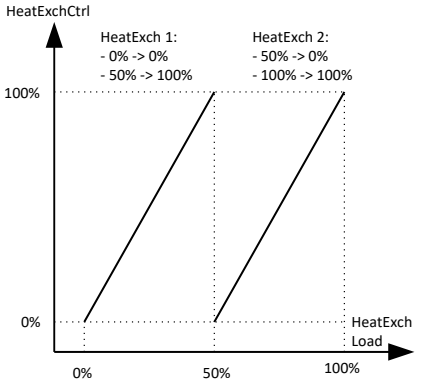


Figure 25: The ratio mechanism used to drive two heat exchangers

Rotor/Pump Operation and Alarm

The rotor/pump is started if the valve is opened more than 5%. A start delay of 5s and stop delay of 300s is applied to the pump operation command.

Heat exchanger status signals can be connected, and depending on the *HeatExchStatus*, *HeatExchCmdFb* and *HeatExchFault* signals, alarms can be generated:

- *HeatExchFault* is treated as a direct fault signal and the rotor/pump alarm will be generated immediately when *HeatExchFault* input is true.
- *HeatExchCmdFb* and *HeatExchStatus* are treated as pump operation feedback signals. *HeatExchStatus* has priority over *HeatExchCmdFb* therefore if both are connected, the latter is ignored. If a feedback fails to follow the *HeatExchEnable* signal, *RunAlarm* (fail to stop) and *StopAlarm* (fail to start) signals will be generated (with corresponding *C.RunAlmDelay* and *C.StopAlmDelay* times).

The algorithm outputs a *HeatExchStatus*. This multistate output signal provides a coded *HeatExchStatus*, with the following values:

Value	Description
1	Stop
2	Start
3	Run Alarm (fail to stop)
4	Stop Alarm (fail to start)
5	Fault Alarm

If the parameter *C.DisCmdInAlarm* is *true*, the pump command will be disabled when the pump trips into the Stop Alarm or Fault Alarm.

Icing Protection

If the heat exchanger equipment is used in a cold climate, hoarfrost build-up is a major issue. It occurs if the outside air temperature is low enough to cause exhaust air temperature to drop below 0ºC which results in condensation freezing on the heat exchanger and gradually blocking the return air flow. This situation is worsened by the fact that a drop in the return air flow causes further temperature decline resulting in faster heat exchanger icing. If it is not prevented, it may lead to an air flow being completely blocked or even equipment getting damaged.

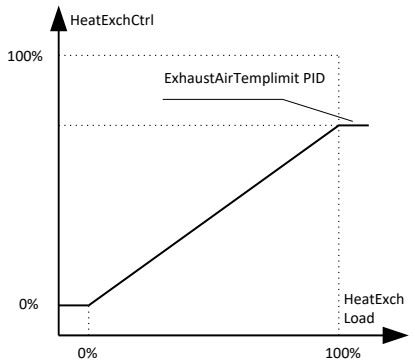


Figure 26: The HeatExchLoad to HeatExchCtrl recalculation

To prevent icing, the *AHU_HeatExchControl_SI* block provides two different remedies using the *ExhaustAirTemp* Input and the *PressSwitch.* An alternative deicing method at the level of the *AHU_OperatingMode* block is also possible. These methods are outlined below.

Defrost with ExhaustAirTemp Input

The *ExhaustAirTemp* input may be used to limit the *HeatExchCtrl.*This would limit heat recovery while at the same time increase exhaust air temperature. If the temperature drops below a setpoint defined by the *C.ExhTempSetPt* parameter (by default 2ºC), a PID regulator will recalculate a maximum limit for the heat exchanger signal. This prevents icing, but on the other hand causes energy loss because heat recovery is limited even if no icing occurs.

A correct localisation of the *ExhaustAirTemp* sensor is critical for this action to be efficiently executed.

Defrost with PressSwitch

The *PressSwitch* may be used to detect actual icing conditions indicated by the increase of the pressure drop on the exhaust side of the heat exchanger. When this happens, the *HeatExchCtrl* signal is set to a value of the *C.DeIcingProtCtrl* parameter (by default 20%). This limits heat recovery, increases exhaust air temperature, and melts the ice. Defrosting is sustained for the *C.DeIcingProtTime* period after the *PressSwitch* goes back to normal. This is a much more energy efficient method than the previous one, but it leads to occasional rapid drop in heat recovery, which might cause the preheater *FreezeAlarm* to activate.

Moreover, a pressure switch setpoint might be difficult to determine if the system is susceptible to large air flow variations (like in VAV based systems). In such a case, a pressure sensor with a flow dependent setpoint might be used rather than the mechanical pressure switch.

Alternate Defrost Method

Another defrost scheme that might be implemented is at the level of the *AHU_OperatingMode* block. When the *HRExchDefrost* input is activated, the block activates the *OperatingMode 11 – HExchDefros*, in which supply fan speed is decreased while the return fan speed rests unchanged. The advantage of this approach is that it would never trigger the activation of the preheater *FreezeAlarm* and heat recovery runs at maximum efficiency all the time. The drawback is that the pressure balance of the AHU is hindered. Fortunately, this method of defrosting is fast and usually lasts only a few minutes, while hoarfrost build-up may take up to a few hours.

The Damper Control Block (AHU_DamperControl_SI)

An *AHU_DamperControl_SI* block transfers the *DamperLoad* signal from the *AHU_DischTempController_SI* block to the hardware outputs controlling the operation of the recirculation dampers. The block also provides regulation of a return air CO_2 level and a mixed air temperature. The block is designed, to operate a set of recirculation dampers using one control signal. Exhaust air and outside air dampers operate on a direct 0% - 100% *DamperCtrl* signal, while recirculation damper action is opposite. This is realized by electromechanical configuration of the actuator rather than by generation of an additional control signal.

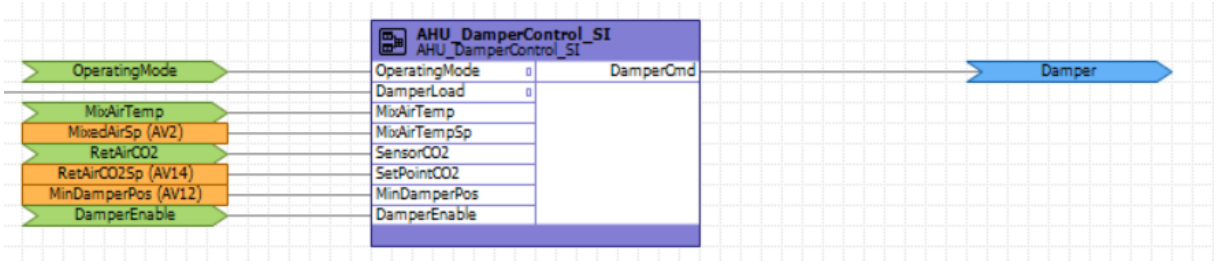


Figure 27: The AHU_DamperControl_SI programming block

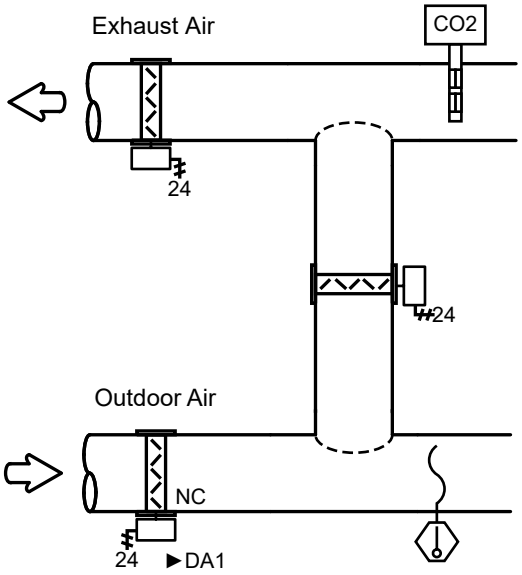


Figure 28: The recirculation dampers hardware configuration

Inputs

The *AHU_DamperControl_SI* block requires connection of input signals (hardware and logic).

Required Inputs

Input Parameter	Description
OperatingMode	A multistate value containing coded status of the AHU. (MI)
DamperLoad	A control command from the *AHU_DischTempController_SI* block. (AI)

Optional Inputs

Input Parameter	Description
MixAirTemp	An input from a mixed air temperature sensor. Default *null*.(AI)
MixAirTempSp	A setpoint for the mixed air temperature control. Default *7°C*.(AI)
RetAirCO2	An input from a return air CO_2 sensor. Default *null*.(AI)
RetAirCO2Sp	A setpoint for the return air CO_2 control. Default *900ppm*.(AI)
MinDamperPos	A minimum outside air damper opening requirement. Default *20%*.(AI)
DamperEnable	An enable signal from the preheater block. Default *true*.(BI)

Outputs

The *AHU_DamperControl_SI* block output signals (hardware and logic).

Output Parameter	Description
DamperCtrl	A damper control signal. (AO)

Block Functions

Dampers Operation

At the AHU start up, the *AHU_DamperControl_SI* block waits for the enable signal passed from the *AHU_PreHeatingControl_SI* via *DamperEnable* input. This ensures that in the wintertime, dampers would not open before the hot water is available in the preheater. This approach greatly limits the danger of the FreezeAlarm activation during the AHU start up.

Then the *AHU_DamperControl_SI* block takes the *DamperLoad* input signal and passes it through a linear ratio block. By default, this linear conversion is scaled 0% -> 0%, 100% -> 100%, but if need arises, it can be used to drive multiple sets of dampers with the same *DamperLoad* signal. Driving multiple sets of dampers with the same *DamperLoad* signal requires an additional *AHU_DamperControl_SI* block and modification of the ratio block settings in both blocks.

Next the *AHU_DamperControl_SI* block calculates a minimum and a maximum allowed damper opening based on the *MixAirTemp,* the *RerAirCO2,* the *MinDamperPos* inputs, and the *C.MaxDamperPos* parameter. These limits are applied to the calculated *DamperLoad* and exposed to the *DamperCtrl* output.

The *DamperCtrl* is used for the outside air and the exhaust air dampers as a direct control signal. 0% means that both dampers are fully closed, and 100% means that they are both fully open. However, the recirculation damper's operation is reversed. The recirculation damper is open when the outside and exhaust air dampers are closed and the recirculation damper is closed when the outside and exhaust air dampers are open. This action is achieved by the correct setup of the recirculation damper actuator rather than use of an additional output signal.

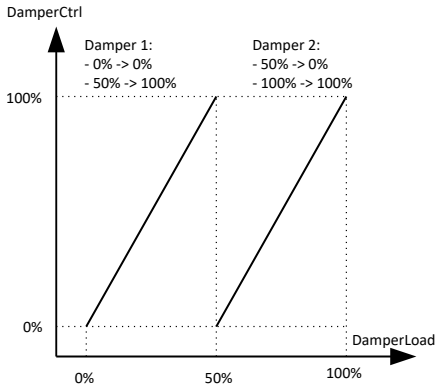


Figure 29: The ratio mechanism used to drive two sets of dampers.

The MixAirTemp Regulation

If the *MixAirTemp* input is connected, the *AHU_DamperControl_SI* block compares it to the *MixAirTempSp.* If the measured temperature (*MixAirTemp)* is below the setpoint (*MixAirTempSp),* an internal PID regulator will start to descend with the maximum allowed damper opening. The PID signal can force a maximum opening to go from the *C.MaxDamperPos* to the *MinDamperPos*.

The RetAirCO2 Regulation

If the *RetAirCO2* input is connected, the *AHU_DamperControl_SI* block compares it to the *RetAirCO2Sp.* If the CO_2 saturation in the return air is above the setpoint, an internal PID regulator will start to increase the minimum required damper opening. The PID signal can force a maximum opening to go from the *C.MaxDamperPos* to the *MinDamperPos*. If at the same time regulation of the *MixAirTemp* and *RetAirCO2* occurs, it is the damper opening limitation by the *MixAirTemp* control that has a higher priority.

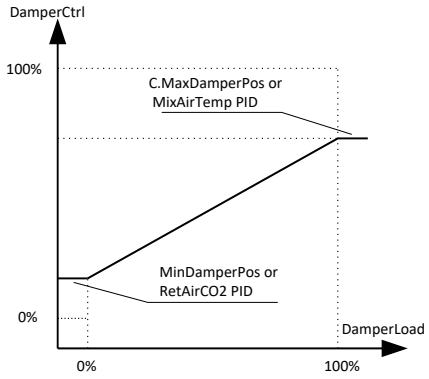


Figure 30: The MixAirTemp and the RetAirCO2 control

The Cooler Control Block (AHU_CoolingControl_SI)

An *AHU_CoolingControl_SI* block transfers the *CoolingLoad* signal from the *AHU_DischTempController_SI* block together with a *DehumidifyLoad* signal from an *AHU_HumidityControl* block to hardware outputs controlling operation of the actual cooling coil components, providing in the process all the necessary functions resulting from the hardware configuration of the AHU.

The block is designed, to operate chilled water cooling coils. It can operate the cooling coil equipped with a cooling valve and a pump.

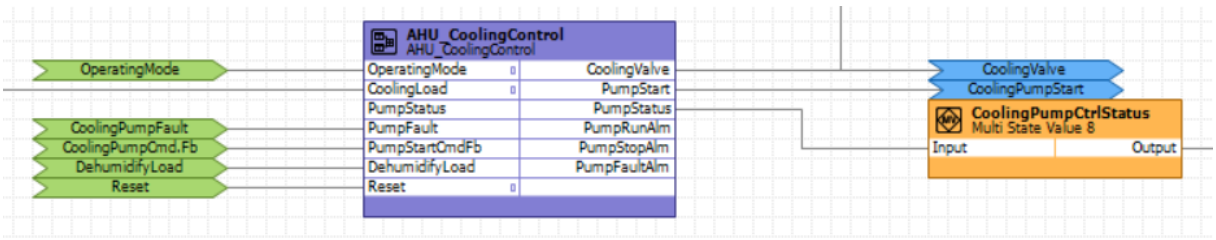


Figure 31: The AHU_CoolingControl_SI programming block

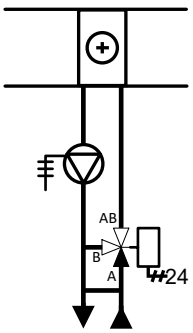


Figure 32: The cooling coil hardware configuration

Inputs

The *AHU_CoolingControl_SI* block requires connection of input signals (hardware and logic).

Required Inputs

Input Parameter	Description
OperatingMode	A multistate value containing coded status of the AHU. (MI)
CoolingLoad	A control command from the *AHU_DischTempController_SI* block. (AI)
Reset	An alarm reset command (AI).

Optional Inputs

Input Parameter	Description
DehumidifyLoad	An input from the *AHU_HumidityControl* block indicating dehumidification demand. Default *0.* (BI)
PumpStatus	Confirms operation of the pump (feedback from a contactor or a flow switch). Do not use simultaneously with the PumpStartCmdFb. Default *null.* (BI)
PumpStartCmdFb	A confirmation signal from the output of the controller that it is energized. Do not use simultaneously with the PumpStatus. Default *null.*(BI)
Pump Fault	A fault signal of the pump. Default *false.*(BI)

Outputs

The *AHU_CoolingControl_SI* block output signals (hardware and logic).

Output Parameter	Description
CoolingValve	A valve control signal. If the block is used to control an electric Cooler, this output controls a current valve. (AO)
PumpStart	A start signal for a Cooler pump. (BO)
PumpStatus	A multistate pump status point. (MO)
PumpRunAlm	A fail to stop alarm active signal. (BO)
PumpStopAlm	A fail to start alarm active signal. (BO)
PumpFaultAlm	A direct fault alarm active signal. (BO)

Block Functions

Cooling Valve Operation

When the AHU is operational, the *AHU_CoolingControl_SI* block takes the *CoolingLoad* input signal and passes it through a linear ratio block. By default, this linear conversion is scaled 0% -> 0%, 100% -> 100%, but if need arises it can be used to drive multiple coolers with the same *CoolingLoad* signal. Driving multiple coolers with the same *CoolingLoad* signal requires an additional *AHU_CoolingControl_SI* block and modification of the ratio block settings in both blocks.

The block then checks if the *DehumidificationLoad* algorithm increased minimum opening of the cooling valve. If the valve is opened more than 5% then the circulation pump is started.

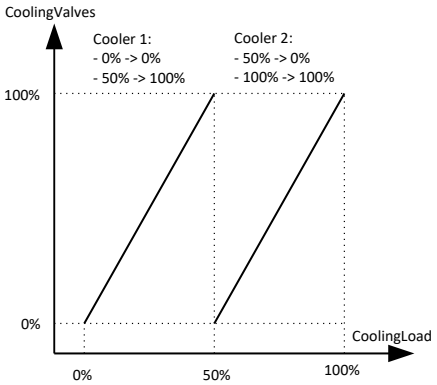


Figure 33: The ratio mechanism used to drive two coolers

Pump Operation and Alarm

The pump is started if the valve is opened more than 5%. A start delay of 5s and stop delay of 300s is applied to the pump operation command.

Depending on the connected status signals, *PumpStatus, PumpStartCmdFb* and *PumpFault*, alarms can be generated:

- *PumpFault* is treated as a direct fault signal and the pump alarm will be generated immediately when *PumpFault* input is true.
- *PumpStartCmdFb* and *PumpStatus* are treated as pump operation feedback signals. *PumpStatus* has priority over *PumpStartCmdFb* and if both are connected, the latter is ignored. If feedback fails to follow the *PumpCmd* signal, *RunAlarm* (fail to stop) and *StopAlarm* (fail to start) signals will be generated (with corresponding *C.RunAlmDelay* and *C.StopAlmDelay* times).

The algorithm outputs a *PumpStatus*. This multistate output signal provides a coded *PumpStatus* with the following values:

Value	Description
1	Stop
2	Start
3	Run Alarm (fail to stop)
4	Stop Alarm (fail to start)
5	Fault Alarm

If the parameter *C.DisCmdInAlarm* is *true*, the pump command will be disabled when the pump trips into the Stop Alarm or Fault Alarm.

The Dehumidification Process

If the *DehumidifyLoad* signal from the *AHU_HumidityControl* block is connected, it will be used as the low limit for the *CollingLoad* to the *CollingvValve* transition. This way dehumidification forces a minimum opening of the valve, while the temperature driven cooling action operates between this limit and 100%.

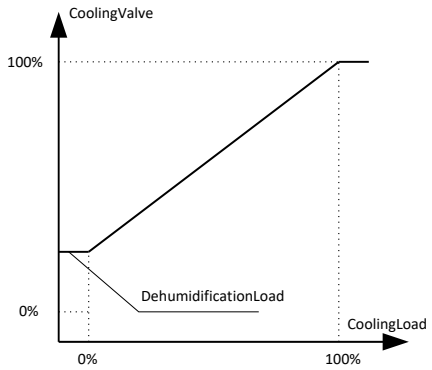


Figure 34: The CoolingLoad to the CoolingValve recalculation

The Heat Demand Calculation Block (AHU_HeatTempRequest_SI)

The *AHU_HeatTempRequest_SI* block checks the opening of the heater valve and calculates a heat demand signal that can be transferred to a heating system management controller to adjust a heat generation system to satisfy the AHU requirements.

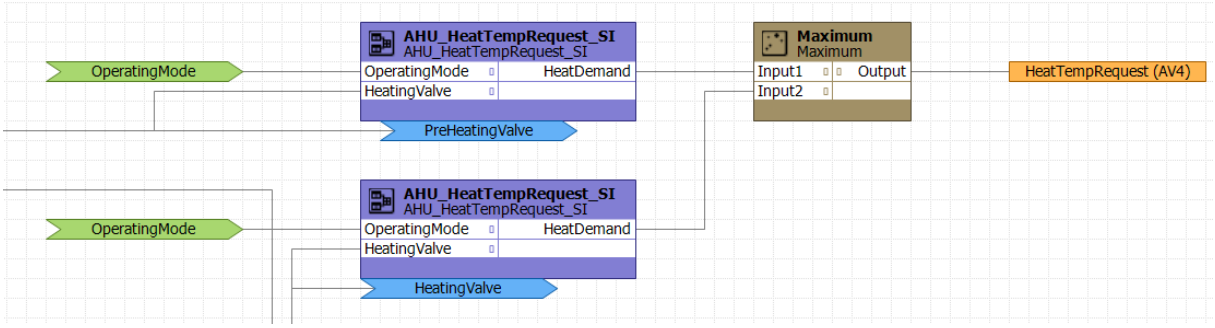


Figure 35: The AHU_HeatTempRequest_SI programming block

Inputs

The *AHU_HeatTempRequest_SI* block requires connection of input signals (hardware and logic).

Required Inputs

Input Parameter	Description
OperatingMode	A multistate value containing coded status of the AHU. (MI)
HeatingValve	A heating valve control signal. (AI)

Optional Inputs

The *AHU_HeatTempRequest_SI* block does not use any optional inputs.

Outputs

The *AHU_HeatTempRequest_SI* block output signals (hardware and logic).

Output Parameter	Description
HeatDemand	A heat demand to be sent to the heating system controller. (AO)

Block Functions

The HeatDemand Calculation

When the AHU is operational, the *AHU_HeatTempRequest_SI* block checks if value of the *HeatingValve* is above a 5% threshold (with 3% hysteresis) to determine if system is in the heating mode. If so, it then applies a time based *HeatDemand* calculation.

If the valve opening is above 90%, the block increases the *HeatDemand* output with the pace of the *C.SetPtChgRatio* degrees per hour (default 5ºC/h) until it reaches *C.MaxHeatingSetPt* (default 85ºC).

If the valve opening is below 30%, but above the 5% threshold (with 3% hysteresis), the block decreases the *HeatDemand* output with the pace of the *C.SetPtChgRatio* degrees per hour (default 5ºC/h) until it reaches *C.MinHeatingSetPt* (default 40ºC).

After a controller reboot, the *CoolingDemand* calculation restarts from 50ºC.

When the AHU is not operational nor in heating mode, the *AHU_HeatTempRequest_SI* stops the *HeatDemand* adjustment and outputs -100ºC to indicate that it does not require heat. However, the block retains the last active value and picks up from it when the *HeatDemand* adjustment is restarted.

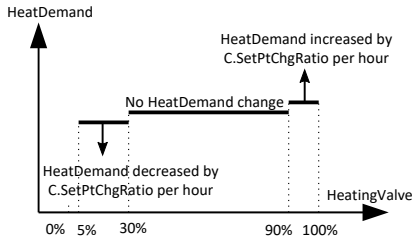


Figure 36: The HeatDemand calculation mechanism

The Chiller Demand Calculation Block (AHU_CoolTempRequest_SI)

The *AHU_CoolTempRequest_SI* block checks the opening of the cooling coil valve and calculates a cooling demand signal that can be transferred to a chiller system management controller to adjust a chilled water generation system to satisfy the AHU requirements.

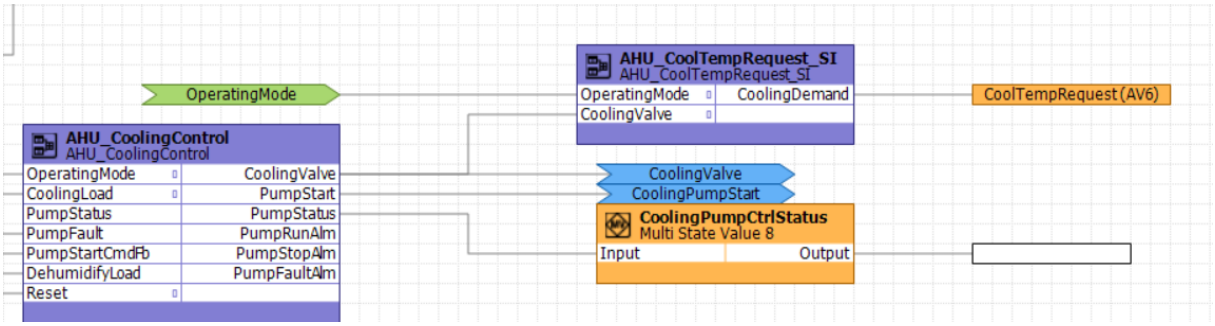


Figure 37: The AHU_CoolTempRequest_SI programming block

Inputs

The *AHU_CoolTempRequest_SI* block requires connection of input signals (hardware and logic).

Required Inputs

Input Parameter	Description
OperatingMode	A multistate value containing coded status of the AHU. (MI)
CoolingValve	A cooling valve control signal. (AI)

Optional Inputs

The *AHU_CoolTempRequest_SI* block does not use any optional inputs.

Outputs

The *AHU_CoolTempRequest_SI* block output signals (hardware and logic).

Output Parameter	Description
CoolingDemand	A cooling demand to be sent to the chiller system controller. (AO)

Block Functions

The CoolingDemand Calculation

When the AHU is operational, the *AHU_CoolTempRequest_SI* block checks the value of the *Cooling-Valve* if it is above a 5% threshold (with 3% hysteresis) to determine that system is in the cooling mode and then applies a time based *CoolingDemand* calculation.

If the valve opening is above 90%, the block decreases the *CoolingDemand* output with the pace of the *C.SetPtChgRatio* degrees per hour (default 1ºC/h) unit it reaches the *C.MinCoolingSetPt*(default 6ºC).

If the valve opening is below 30% but above the 5% threshold (with 3% hysteresis), the block increases the *CoolingDemand* output with the pace of the *C.SetPtChgRatio* degrees per hour (default 1ºC/h) unit it reaches the *C.MaxCoolingSetPt* (default 12ºC).

After a controller reboot, the *CoolingDemand* calculation restarts from 8ºC.

When the AHU is not operational or not in the cooling mode, the *AHU_CoolTempRequest_SI* stops the *CoolingDemand* adjustment and outputs 100ºC to indicate that it does not require chilled water. However, the block remembers the last active value and picks up from there when the *CoolingDemand* adjustment is restarted.

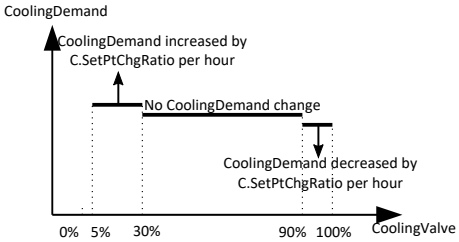


Figure 38: The CooloingDemand calculation mechanism

The Return / Room Air Humidity Control (AHU_HumidityControl)

The *AHU_HumidityControl* block calculates the *HumidifierCtrl* and the *DehumidifyLoad* signals providing precise control of the return air humidity using a PID algorithm. The block handles direct analogue control, enable signal, and alarm supervision of the humidifier. However, for the dehumidification process it only provides the *DehumidifyLoad* and the *Dehumidification* signals, which as transmitted to the *AHU_CoolingControl* and the *AHU_DischTempController_SI* to force them to engage in the air dehumidification process.

The humidification and dehumidification action have their own PID regulators, setpoints, and enable inputs. Both actions are interlocked to avoid simultaneous operation.

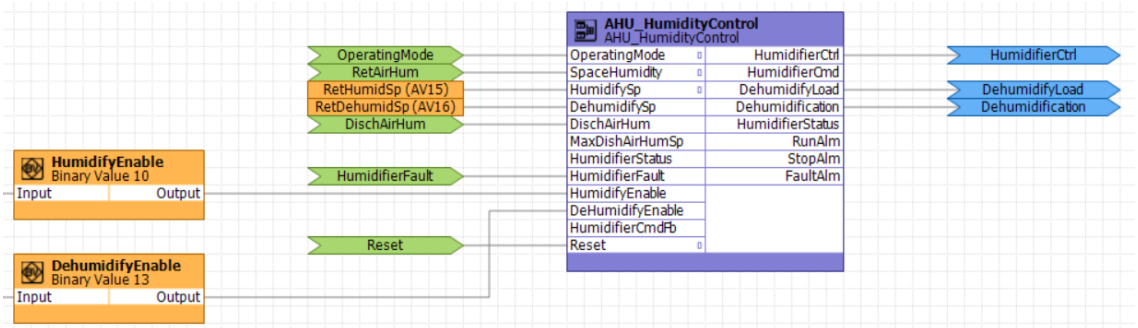


Figure 39: The AHU_HumidityControl programming block

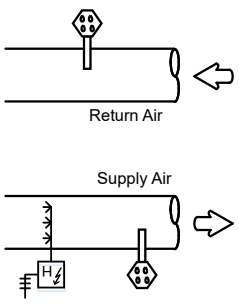


Figure 40: The humidifier hardware configuration

Inputs

The *AHU_HumidityControl* block requires connection of input signals (hardware and logic).

Required Inputs

Input Parameter	Description
OperatingMode	A multistate value containing coded status of the AHU. (MI)
RetAirHum	A space/return air relative humidity value. (AI)
RetAirHumSp	A space/return air humidification setpoint value. (AI)
Reset	An alarm reset command (BI).

Optional Inputs

Input Parameter	Description
RetAirDehumSp	A space/return air dehumidification setpoint value. Default *100%rh*. (AI)
DischAirHum	A space/return air relative humidity value. (AI)
MaxDishAirHumSp	A maximum discharge air humidity setpoint. Default *85%rh*. (AI)
HumidifyEnable	A humidification enable signal. Default *true*.(BI)
DeHumidifyEnable	A dehumidification enable signal. Default *false*.(BI)
HumidifierStatus	Confirms operation of the humidifier (feedback from a contactor or a flow switch). Do not use simultaneously with *HumidifierCmdFb*. Default *null*.(BI)
HumidifierCmdFb	A confirmation signal from the output of the controller that it is energized. Do not use simultaneously with *HumidifierStatus*. Default *null*.(BI)
HumidifierFault	A fault signal of the pump. Default *false*.(BI)

Outputs

The *AHU_HumidityControl* block output signals (hardware and logic).

Output Parameter	Description
HumidifierCtrl	The humidifier control signal directly used to drive humidifier valve or electric humidifier. (AO)
HumidifierCmd	A start signal for the humidifier. (BO)
DehumidifyLoad	A dehumidification control signal used to interface with the *AHU_CoolingControl* block. (AO)
Dehumidification	A dehumidification activation command signal used to interface with the *AHU_DischTempController_SI* block. (AO)
HumidifierStatus	A multistate HumidifierStatus point. (MO)
RunAlm	A humidifier's fail to stop alarm active signal. (BO)
StopAlm	A humidifier's fail to start alarm active signal. (BO)
FaultAlm	A humidifier's direct fault alarm active signal. (BO)

Block Functions

Humidification Control

The *AHU_HumidityControl* block checks if the *HumidifyEnable* input is active and the dehumidification PID is inactive. It then activates the humidification PID, which compares the *RetAirHum* with the *RetAirHumSp.* If the measured value is below the setpoint, the humidification control signal is increased. Next the block checks if the *DischAirHum* does not exceed the *MaxDishAirHumSp.* If it does, the humidification control signal is limited. Next the effective control signal is exposed via the *HumidifierCtrl* output. It is also drives activation of the *HumidifierCmd* binary output used to operate the humidifier enable signal or a humidifier pump.

The *HumidifyEnable* input is usually connected to the an *OutsideAirTemp*-based hysteresis block. It prevents humidification from being activated in the summertime when the absolute outside air humidity is very high and need for humidification does not occur.

Electric Humidifier / Humidifier Pump Operation and Alarm

The *HumidifierCmd* output is used to drive the electric humidifier/humidifier pump. It is activated if *HumidifierCtrl* exceeds 5%. A start delay of 5s and stop delay of 300s is applied to the operation command.

If the humidifier pump is used without the analogue control signal, the stop delay time needs to be modified to 0s.

Depending on the connected status signals *HumidifierStatus*, *HumidifierStartCmdFb*, and *HumidifierFault* alarms can be generated:

- *HumidifierFault* is treated as a direct fault signal and the pump alarm will be generated immediately when *HumidifierFault* input is true..
- *HumidifierStartCmdFb* and *HumidifierStatus* are treated as pump operation feedback signals. *HumidifierStatus* has priority over *HumidifierStartCmdFb* and if both are connected, the latter is ignored. If feedback fails to follow the *HumidifierCmd* signal, *RunAlarm* (fail to stop) and *StopAlarm* (fail to start) signals will be generated (with corresponding *C.RunAlmDelay* and *C.StopAlmDelay* times).

An algorithm outputs a *HumidifierStatus*. This multistate output signal provides a coded *HumidifierStatus*, with the following values:

Value	Description
1	Stop
2	Start
3	Run Alarm (fail to stop)
4	Stop Alarm (fail to start)
5	Fault Alarm

If the parameter *C.DisCmdInAlarm* is *true*, the pump command will be disabled when the pump trips into the Stop Alarm or Fault Alarm.

Maximum DischAirHum Limitation

If the relative humidity in the supply duct is too high, there is a risk of water condensation on the duct walls if they are colder than the supply air. To prevent this from happening, an additional PID controller compares the *DischAirHum* to the *MaxDischAirHumSp* (default 85%rh). If the discharge air humidity rises too high, the output of this PID limits the *HumidifierCtrl* output. With the PID's P_{band} = 8%rh, the humidifier control signal is sure to be 0% when the *DischAirHum* reaches 93%rh.

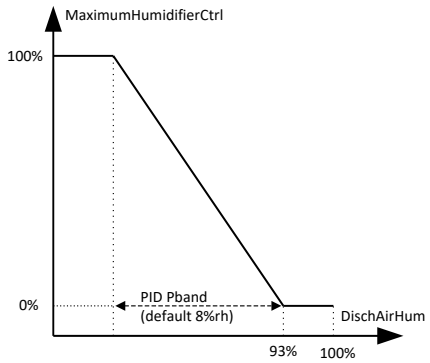


Figure 41: The maximum HumidifierCtrl limit

Dehumidification Control

The *AHU_HumidityControl* block checks if the *DehumidifyEnable* input is active and the humidification PID is inactive. It then activates the dehumidification PID, which comperes the *RetAirHum* with the *RetAirDehumSp.* If the measured value is above the setpoint, the *DehumidifyLoad* control signal will be increased. Next, the block checks if the load signal surpasses 10%, then the *Dehumidification* output is activated in order to force the *AHU_DischTempController_SI* to block operation of the preheater. An on/off delay is applied to the *Dehumidification* output with an on time of 5s and an off time of 60s.

The *DehumidifyEnable* should be connected to the *OutsideAirTemp* based hysteresis block to prevent dehumidification in the wintertime. Firstly, because the absolute outside air humidity is very low and the need for dehumidification is not necessary and secondly because blocking of the preheater operation that is inherently connected with the dehumidification process might lead to the entire AHU falling into the *FreezeAlarm*.

Pressure Regulation and Fan Operation – FanControl Page

A *FanControl* programming page provides algorithms for pressure control and correct operation of fans and isolation dampers.

Programming blocks can be divided into two major categories – general regulation and equipment related. The general regulation blocks take sensor input information (from pressure inputs) together with setpoints, apply PID control, and set fan speed demand signals. These signals are then transferred to the equipment control blocks which are adjusted to fit the mechanical configuration of the AHU components. They operate hardware outputs to provide optimal operation of the AHU sections to satisfy calculated demand.

Isolation Dampers Control Block (AHU_IsolationDamper)

An *AHU_IsolationDamper* block takes the *DamperEnable* signal from the *AHU_PreHeaterControl_SI* block, generates a *DamperCmd* signal to open the isolation dampers, and waits for opening confirmation before it sends a *FanEnable* command to start the fans. It also generates a fail to open alarm if the block does not receive a confirmation from an end switch in time.

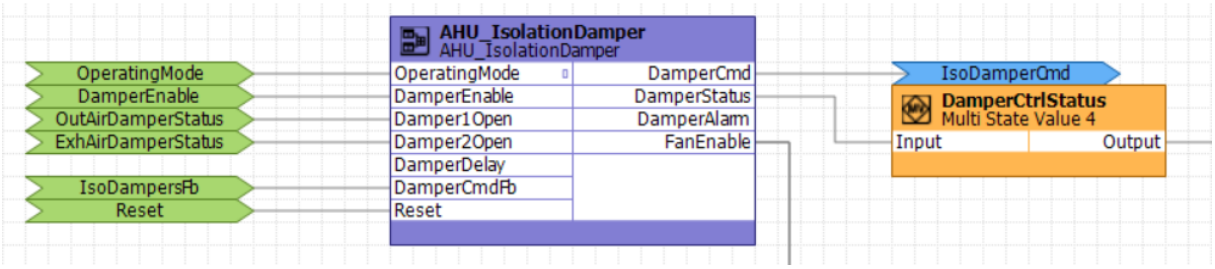


Figure 42: The AHU_IsolationDamper programming block

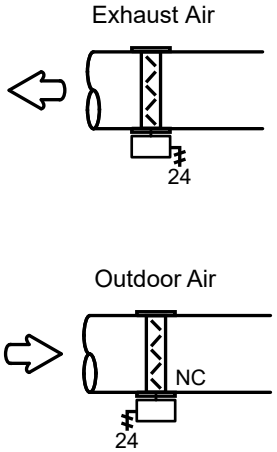


Figure 43: The isolation dampers hardware configuration

Inputs

The *AHU_IsolationDamper* block requires connection of input signals (hardware and logic).

Required Inputs

Input Parameter	Description
OperatingMode	A multistate value containing coded status of the AHU. (MI)

Optional Inputs

Input Parameter	Description
DamperEnable	An enable signal from the preheater block. Default *true*.(BI)
Damper1Open	An opening confirmation signal from the first damper end switch. Default *null*.(AI)
Damper2Open	An opening confirmation signal from the second damper end switch. Default *null*.(AI)
DamperDelay	A damper running time. Default *60s*.(AI)
DamperCmdFb	A confirmation signal from the output of the controller that it is energized. Default *null*.(BI)

Outputs

The *AHU_IsolationDamper* block output signals (hardware and logic).

Output Parameter	Description
DamperCmd	A damper command signal. (BO)
DamperStatus	A multistate damper status point. (MO)
DamperAlarm	A fail to open alarm active signal. (BO)
FanEnable	An enable signal to the fan control block. (BO)

Block Functions

Damper Operation

At the AHU start up, the *AHU_IsolationDamper* block waits for the enable signal to be passed from the *AHU_PreHeatingControl_SI* via the *DamperEnable* input. This ensures that in the wintertime the dampers would not open before hot water is available at the preheater. This approach greatly reduces the danger of the *FreezeAlarm* activating during the AHU start up.

The *AHU_IsolationDamper* block takes the *DamperEnable* input ,and if *DamperAlarm* is not active, it sets the *DamperCmd* output. The block checks if the *DamperCmdFb* input is connected, and if so, it waits to receive confirmation that a hardware output driving the dampers has been energized. Then the block checks if the *Damper1Opened* and *Damper2Opend* end switch inputs are connected to activate the *FanEnable* command when both dampers are opened. If however, any of the two are not connected, a delay timer will substitute for the missing input. Delay time is defined by the *DamperDelay* input (default value 60s).

The Dampers Fail to Open Alarm

The block checks if the *DamperCmdFb* input indicates that the damper command output is active (if this input is not available, the block will take the state of the *DamperCmd* output) and waits twice the *DamperDelay* time to receive the opening confirmation from both dampers. If any of the dampers fail to open in time, the *DamperAlarm* is generated. If any of the end switch inputs are not connected, then it is excluded from the alarm generation algorithm.

Damper Status Output

The *AHU_IsolationDamper* block provides the multistate *DamperStatus* output, which provides a coded value of the isolation dampers status:

Value	Description
1	Open
2	Closed
3	Fail to Open alarm

The Pressure Control Block (AHU_PressureControl_SI)

The *AHU_PressureControl_SI* block provides pressure regulation by means of fan speed variation.

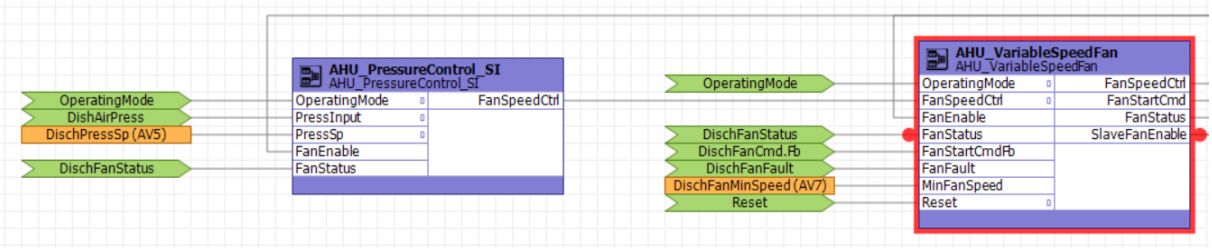


Figure 44: The AHU_PressureControl_SI programming block

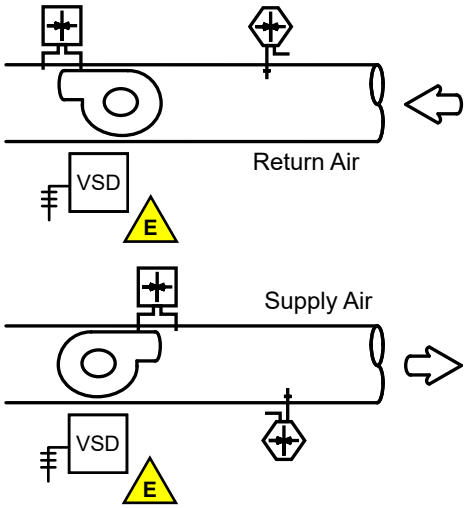


Figure 45: The supply/return fan hardware configuration

Inputs

The *AHU_PressureControl_SI* block requires connection of input signals (hardware and logic).

Required Inputs

Input Parameter	Description
OperatingMode	A multistate value containing coded status of the AHU. (MI)
PressInput	An input from a duct pressure sensor. (AI)
PressSp	A duct pressure setpoint. (AI)

Optional Inputs

Input Parameter	Description
FanEnable	An enable signal from the dampers control block. Default *true* (BI)
FanStatus	A fan operation confirmation signal (feedback from a pressure switch). Default *true* (BI)

Outputs

The *AHU_PressureControl_SI* block output signals (hardware and logic).

Output Parameter	Description
FanSpeedCtrl	A fan speed control signal. (AO)

Block Functions

The Pressure Controller Operation

When the AHU is operational and the *FanEnable* input is active, the *AHU_PressureControl_SI* block activates a PID regulator which takes the *PressInput* and compares it with the *PerssSp* to set the *FanSpeedCtrl* output.

If the pressure input is invalid, the block sets the *FanSpeedCtrl* output to a *C.FanSpeedFault* parameter defined value.

Fan Type Based Operation

The binary configuration parameter *C.SupplyFan* sets the fan type. If it is active, then the fan is considered as a supply fan and if the *OperatingMode* input is set to *11-HRExchDefros,* the block halves the pressure setpoint to enable heat exchanger defrosting by limiting the outside air volume intake.

Due to the nonlinear nature of a fan speed to pressure characteristic, the PID's *Proportional band* and *Integral Time* parameters should be adjusted with care during the commissioning process. Distech Controls also provides an advanced procedure for a fan to pressure characteristic linearisation, which greatly improves operation of AHU systems which are susceptible to big flow variations (for example VAV based systems). For further details check the "Fan Control Optimisation" white paper document on our website.

Damper Status Output

The *AHU_IsolationDamper* block provides the multistate *DamperStatus* output, which provides a coded value of the isolation dampers status:

Value	Description
1	Open
2	Closed
3	Fail to Open alarm

The Variable Speed Fan Control Block (AHU_VariableSpeedFan)

The *AHU_VariableSpeedFan* block provides control for a fan equipped with a Variable Speed Drive. It takes the *FanSpeedCtrl* signal from the *AHU_PressureControl_SI* block and hardware inputs indicating status information on the fan, in turn signalling the VSD to issue fan start and fan speed control commands with additional fan status and alarms points.

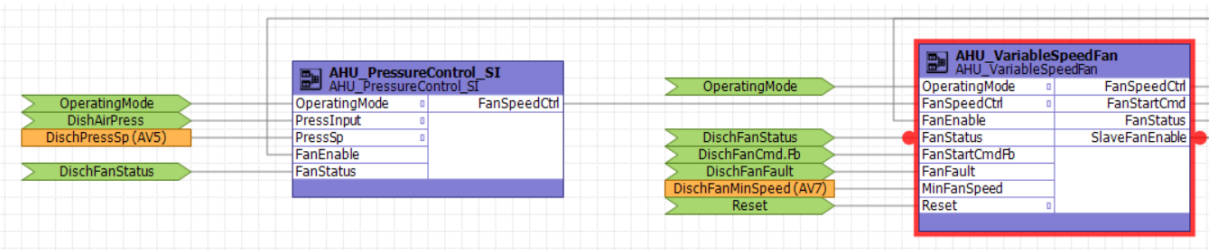


Figure 46: The AHU_VariableSpeedFan programming block

Inputs

The *AHU_VariableSpeedFan*block requires connection of input signals (hardware and logic).

Required Inputs

Input Parameter	Description
OperatingMode	A multistate value containing coded status of the AHU. (MI)
FanSpeedCtrl	A fan speed control signal from the pressure controller. (AI)
Reset	An alarm reset command (BI).

Optional Inputs

Input Parameter	Description
FanEnable	An enable signal from the dampers control block. Default *true*.(BI)
FanStatus	Confirms operation of the fan or VSD (feedback from a contactor or a pressure switch). Do not use simultaneously with the FanStartCmdFb. Default *null*.(BI)
FanStartCmdFb	A confirmation signal from the output of the controller that it is energized. Do not use simultaneously with the FanStatus. Default *null*.(BI)
FanFault	A fault signal of the fan or VSD. Default *false*.(BI)
MinFanSpeed	A minimum value of the FanSpeedCtrl output. Default *20%*.(AI)

Outputs

The *AHU_VariableSpeedFan* block output signals (hardware and logic).

Output Parameter	Description
FanSpeedCtrl	A fan speed control signal. (AO)
FanStartCmd	A fan start signal. (BO)
FanStatus	A multistate fan status output. (MO)
FanRunAlm	A fail to stop alarm active signal. (BO)
FanStopAlm	A fail to start alarm active signal. (BO)
FanFaultAlm	A direct fault alarm active signal. (BO)
SlaveFanEnable	An output signal to enable the next fan. (BO)

Block Functions

The Fan Speed Control

When the AHU is operational and the *FanEnable* input is active, the *AHU_VariableSpeedFan* block activates the *FanStartCmd* output and takes the *FanSpeedCtrl* input signal and passes it through a linear ratio block with the *MinFanSpeed* parameter as a minimum output reference. The minimum value should be set to force the fan to run fast enough for the pressure switch connected to the *FanStatus* input to toggle. By default, this linear conversion is scaled 0% -> *MinFanSpeed*, 100% -> 100%. If need arises, it can also be used to drive multiple fans with the same *FanSpeedCtrl* input signal which would require an additional *AHU_VariableSpeedFan* block and modification of the ratio block settings in both blocks.

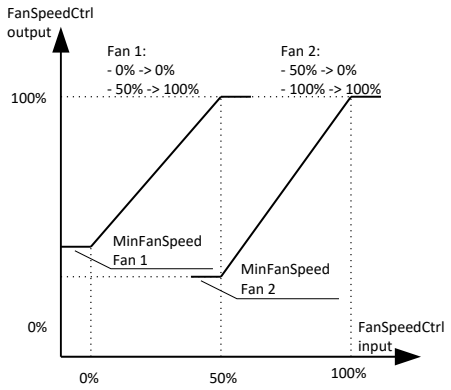


Figure 47: The ratio mechanism used to drive two fans

Fan Operation and Alarm

The fan is started when the AHU unit is operational and the *FanEnable* signal is active.

A start delay defined by the *C.FanStartDelay* parameter is applied. Variation of start delay times helps to prevent an excessive inrush current which may occur in big installations when multiple AHUs start simultaneously (for example after a power failure).

Depending on the connected status signals, *FanStatus*, *FanStartCmdFb,* and *FanFault* alarms can be generated:

- *FanFault* is treated as a direct fault signal and the fan alarm will be generated immediately when the *FanFault* input is true.
- *FanStartCmdFb* and *FanStatus* are treated as fan operation feedback signals. *FanStatus* has priority over *FanStartCmdFb*, and if both are connected, the latter is ignored. If feedback fails to follow *FanCmd* signal, the *RunAlarm* (fail to stop) and *StopAlarm* (fail to start) signals will be generated (with corresponding *C.RunAlmDelay* and *C.StopAlmDelay* times).

An algorithm outputs a *FanStatus*. This multistate output signal provides a coded *FanStatus*, with the following values:

Value	Description
1	Stop
2	Start
3	Run Alarm (fail to stop)
4	Stop Alarm (fail to start)
5	Fault Alarm

If the parameter *C.DisCmdInAlarm* is *true*, the fan command will be disabled when the fan trips into the Stop Alarm or Fault Alarm.

Emergency Operation Mode

If the *OperatingMode* input goes into *2-EmergencyStart,* the *AHU_VariableSpeedFan* block immediately activates the *FanStartCmd* ignoring all active alarms. At the same time, the *FanSpeedCtrl* output is controlled according to the *C.EmergStartConstSpeed* parameter value. If the parameter is active, the *FanSpeedCtrl* is set to the *C.EmergStartFanSpeed* parameter value, otherwise if it is inactive, it follows the *FanSpeedCtrl* input.

Please bear in mind that this operation mode should be used only when a real emergency occurs because it forces fan operation despite any alarm conditions and may cause damage to AHU components.