Concept

Functionblocks for Heating, Ventilation & Air Condition

HVAC

Version 2.5

Block Library

840 USE 478 00

12/03

Data, Illustrations, Alterations
Data and illustrations are not binding. We reserve the right to alter products in line with our policy of continuous product development. If you have any suggestions for improvements or amendments or have found errors in this publication, please notify us using the form on one of the last pages of this publication.

Training
Schneider Automation offers suitable further training on the system.

Hotline
See addresses for the Technical Support Centers at the end of this publication.

Trademarks
All terms used in this publication to denote Schneider Automation products are trademarks of Schneider Automation.

All other terms used in this publication to denote products may be registered trademarks and/or trademarks of the corresponding Corporations.
Microsoft and MS-DOS are registered trademarks of Microsoft Corporation, Windows is a brandname of Microsoft Corporation in the USA and other countries.
IBM is a registered trademark of International Business Machines Corporation.
Intel is a registered trademark of the Intel Corporation.

Copyright

Contents

About

This documentation helps you to configure EFBs for ventilation and air conditioning (VAC) which can be loaded into Concept at a later stage.

Layout of the documentation:

Chap. 1 Contains general information on assigning parameters to EFBs

Chap. 2 Contains general information on the use of EFBs for air conditioning

EFB descriptions Includes a description of EFBs in alphabetical order according to their respective abbreviations.

Glossary Includes a glossary in alphabetical order.

☞ **Note**
For the work with this software package, the user must have acquired knowledge of closed loop control.

Symbols used

Note
This symbol is used to draw your attention to important circumstances.

Caution
This symbol indicates error sources which occur frequently.

Warning
This symbol points out potential dangers to you which could result in financial damage, personal injury or other serious consequences.

Expert
This symbol is used if more detailed information is given which is intended exclusively for experts (with special training). Skipping this information does not affect comprehensibility of the document and does not restrict standard application of the product.

Tip
This symbol draws your attention to explanations given in special tips which help you in your dealings with the product.

Example: This symbol indicates an application example.

▶ **Proceed as follows ...**
The start of a sequence of applications, the execution of which is necessary to obtain a specific product function, is marked with this.

This symbol indicates manuals/other sources dealing with the topic in greater detail.

Terms and abbreviations used

The notation used for figures is in line with international practice, as well as a type of representation allowed by SI (Système International d' Unités): Thousands are separated by a space and the decimal point is used, e.g. 12 345.67.

Additional documentation

Description	Type
Concept Installation instructions	840 USE 502 00
Concept User Manual (Vol.1 + Vol. 2)	840 USE 503 00
IEC block library Concept (Vol. 1 + Vol. 2 + Vol. 3)	840 USE 504 00
Modicon TSX Quantum PLC Series, Hardware User Manual	840 USE 100 00
Modbus Plus Network User Manual	890 USE 100 00
Modlink User's Guide Modicon	GM–MLNK–001
User's Guide Modicon IBM Host Based Devices	GM–HBDS–001
User's Guide BM85 Modbus Plus Bridge / Multiplexer	GM–BM85–001
Planning and Installation Guide Modicon Quantum Hot Standby System	840 USE 106 00
Quantum Ethernet TCP/IP Module User Guide	890 USE 107 00

Note on validity

This documentation applies to Concept Version 2.5 / 2.6 under Microsoft Windows 95, Windows 98, Windows NT or Windows 2000.

Note
You will find additional up–to–date instructions in the Concept file called `README.PDF`.

Parameter assignment

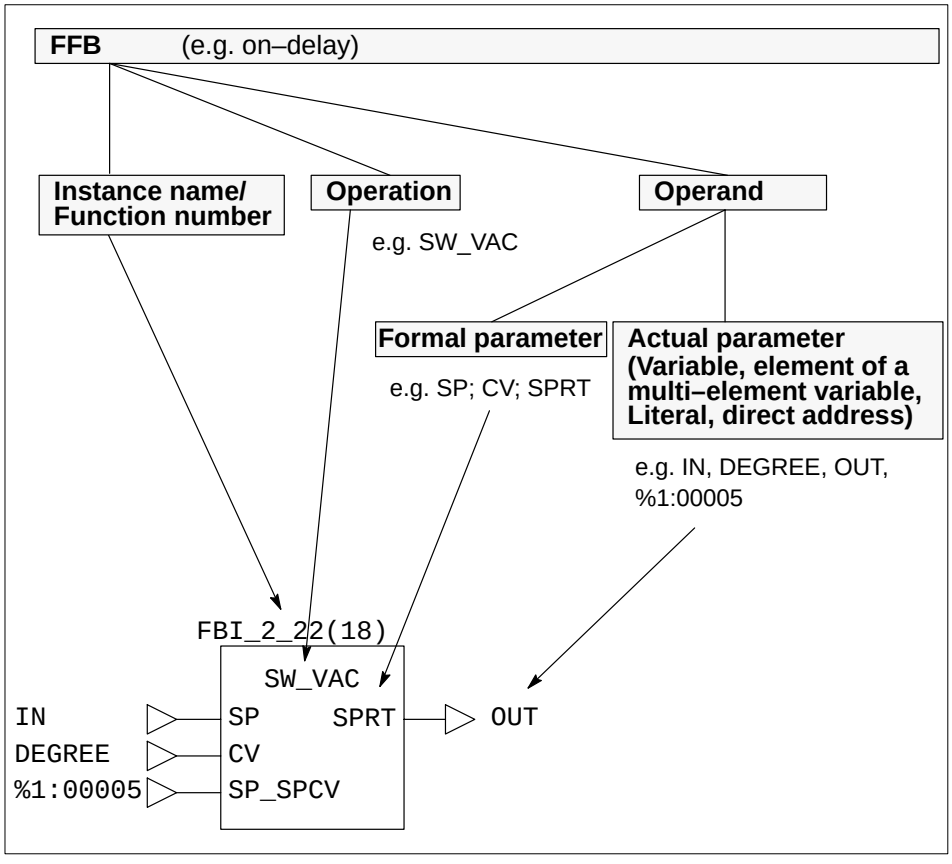
1

This chapter contains general notes for parameter assignment of the function and function blocks (FFBs).

1.1 General Information

Each FFB is composed of an operation, operands required for the operation, and an instance name/function number.

Figure 1 Parameter assignment with the function block SW_VAC as an example



1.1.1 Operation

The operation determines which functionality is to be executed by the FFB, e.g. shift register, conversion operations.

1.1.2 Operand

The operand determines what the operation will be executed with. In FFBs it consists of formal parameter and actual parameteralparameter und Aktualparameter.

1.1.3 Formal parameter/Actual parameter

The formal parameter is a placeholder for an operand. During parameter assignment, the formal parameter is allocated an actual parameter (actual parameter).

The actual parameter can be a variable, a multi–element variable, an element of a multi–element variable, a literal or a direct address.

1.1.4 Conditional/Unconditional Call

Each FFB has the capability of a ”conditional” or an ”unconditional” call. The condition will be evaluated via a prelink of the EN input.

- EN = ENABLE
  conditional call (the FFB will only be executed when ENABLE is set)
- EN hidden
  unconditional call (FFB is always executed)

☞ **Note**
If the EN–input is not parameterized, it must be hidden, or the FFB will never be executed.

General information

2

In this chapter, you will find general information about using EFB library air conditioning (HVAC) modules.

☞ **Note**
For the work with this software package, the user must have acquired knowledge of closed loop control.

2.1 General instructions

The HVAC EFB library puts at your disposal an extensive range of function blocks for the implementation of air conditioning systems using the Concept programming language.

2.2 Function blocks

11 function blocks are available, split up as follows:

- 6 basic function blocks (Basic Group , see Table 1)
- 5 complex function blocks (Complex Group, see Table 2)

Basic Function Blocks
The basic function blocks implement low level functions that are required for solving basic airconditioning problems.

Table 1 Basic Functions Blocks

Function Block	Description
PI_VAC	PI Controller for Air Conditioning
SEQ_VAC	Scaling / Sequence Block for Air Conditioning
SW_VAC	Summer / Winter Setpoint Compensation for Air Conditioning
THRS_VAC	Threshold Switch with Hysteresis for Air Conditioning
VQ_VAC	Measured Value Deadband Block
WASH_VAC	Basic Washer Block for Air Conitioning

Complex Function Blocks
The complex function blocks implement in the form of EFB's the more complex functions found in controllers which are frequently used in air conditioning systems. They are built using the basic function blocks.

Table 2 ComplexFunctions Blocks

Function Block	Description
CC2_VAC	Cascade Controller for Air Conditioning with 2 Outputs
CC3_VAC	Cascade Controller for Air Conditioning with 3 Outputs
MC_VAC	Air Mix Controller for Air Conditioning with 1 Output
UC2_VAC	PI Controller for Air Conditioning with 2 Outputs
UC3_VAC	PI Controller for Air Conditioning with 3 Outputs

☞ **Note**
It is strongly recommended that the user familiarizes himself with the operation of the basic function blocks before reading the complex function block sections.

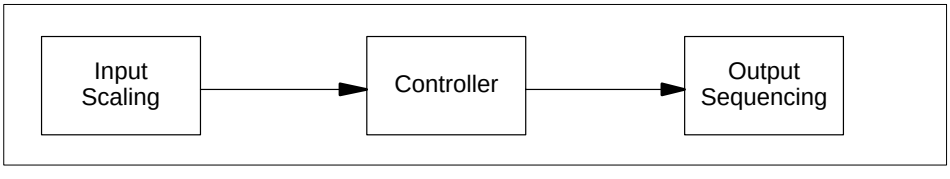
2.3 Scaling and descaling within a control program

2.3.1 General instructions

Basically, a control set up consists of 3 parts (see Figure 2):

- The scaling of input variables
- The controller
- The sequencing of output variables

Figure 2 Control Set up



"Output Sequencing" is used to take the controller single output, split it into several parts, and to scale the resultant outputs. It can be used for example, to split the controller output amongst multiple actuators. For the purposes of the Concept HVAC library, the controller and output sequencing are combined into one EFB.

The scaling of inputs can be handled using standard Concept libraries or the HVAC library SEQ_VAC EFB can be used.

The complex EFB's include the controller plus one or more freely parameterizable output sequence blocks. The behavior of a complex EFB can be determined by examining the basic EFB's from which it is built. The complex EFB's are described in this document by referring to their constituent basic EFB's .

The combination of the controller with the output sequencing has been done to simplify the parameterization of the EFB as all the output sequence parameters can be assigned at the same time as the controller parameters. Furthermore, the links from the controller output to the sequenced outputs are automatically created with a fixed structure.

2.3.2 Single loop controls

In the case of simple, single loop controls, the purpose of the output sequencing is simply to match the controller output to the process actuators.

For controllers operating in simple P–mode, the controller output will equal zero when the setpoint is equal to the process variable. In this case, the output sequencing must be able to handle negative controller outputs and convert them into signals for the actuators. By assigning the appropriate values to the output sequences, it is possible to design a P–controller with a 50% output when the process variable is equal to the setpoint. The advantage of such an approach is that on startup, the controller output will drive a heating action which will reduce the chances of icing up in winter conditions. However, where the risk of icing up is great, this mechanism should not be depended upon for freeze protection as unfavorable dynamics could result in the heating control valve closing. In this case one should control the temperature manually by opening the heating control valve 100% on start up for a specific period of time before putting the controller into automatic and starting the fan.

Where a project uses multiple controllers that have the same behavior, the user should define a consistent naming convention for the Concept project, e.g.,

Y1 / Y2 / Y3 = Heat / Air Mix / Cool.

Where a controller output is divided into multiple process outputs using the sequence functions, one may have to take into account the fact that different process actuators have different gains. As a result, the controller gain may vary depending on the position of its output, i.e., whether the output is driving, a hot water control valve, an air mixing damper or humidity control device. The different actuator gains can be compensated by using the appropriate sequence layout.

2.3.3 Multi–loop controls

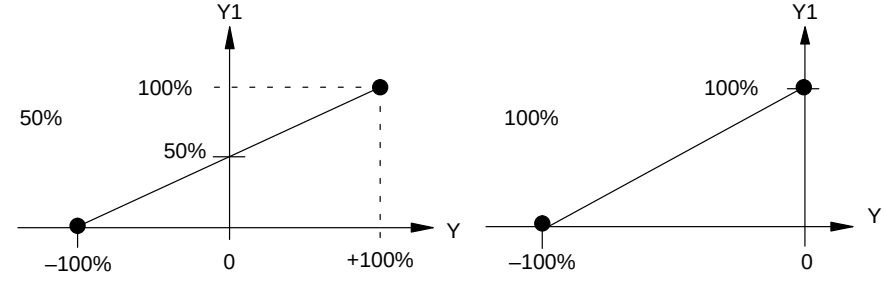
As well as single loops directly connected to process outputs, cascaded loops where the output of one controller is connected as the setpoint of another controller are often required for HVAC systems. The use of cascaded controllers not only gives improved dynamics, it also results in a clearer and easier to understand layout of the HVAC controls.

The output sequence functions allow the user to design a wide variety of control options for his HVAC strategy.

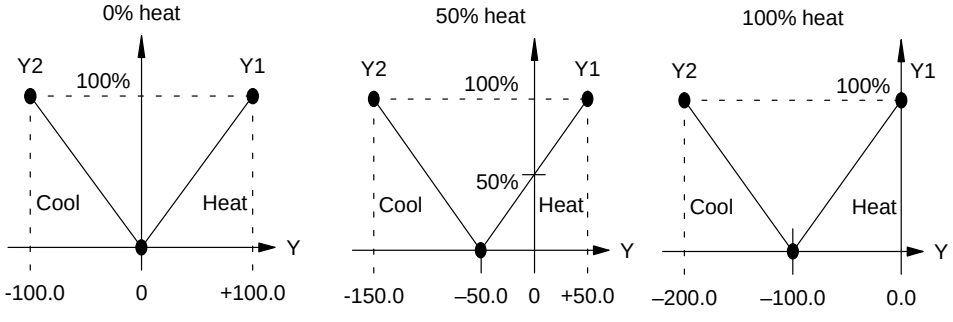
Examples of output sequencing are given in Figure 1. Y refers to the PI controller output, while Y1, Y2 and Y3 refer to the final outputs from the output sequencing which are either sent directly to the process actuators or to other programming controllers/functions.

Figure 1 Examples of sequences with direct output of output values as manipulated variables

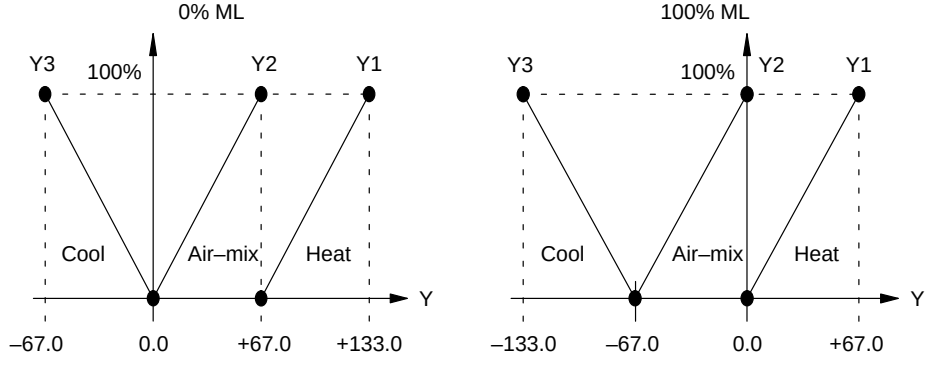
a) Single manipulated variable



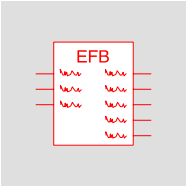
b) Double manipulated variable



:) Triple manipulated variable



EFB Descriptions



The EFB descriptions are arranged alphabetically according to their abbreviations.

CC2_VAC
Cascade controller for air conditioning with 2 outputs

1 Brief description

The CC2_VAC module is a cascade controller used to provide temperature or humidity control of the inlet air to a room. It consists of a P–only outer loop which uses the room temperature/humidity as process variable and an inner PI loop that controls the temperature/humidity of the inlet air supplying the room.

The EFB has a fixed structure where the setpoint of the P–controller is fed forward and added to the P–controller output to form the setpoint of the PI–controller. The output of the PI–controller has 2 output sequences, Y1 and Y2.

The EFB provides the following :

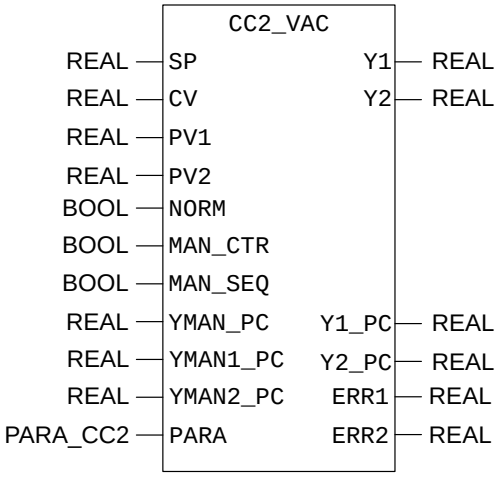
- Winter/summer setpoint compensation as per DIN 1946 pert 2.
- Full four quadrant operation of the output sequence scaling
- The display of output variables as percentages
- Upper and lower limits on outputs
- Presetting the controllers' gains in the form of GAIN or PROP with the possibility of using negative values for switching the control direction.
- Operation with Anti–Windup–Reset (AWR)
- Manual adjustment of either the PI–controller output or the individual sequence outputs (Y1 and Y2) using percentages. When the controller output is manually adjusted, the EFB tracks the I contribution in order to provide bump–less switching back to automatic mode.
- The display of the P and PI controller errors (SP–PV).

Note
Additional parameters EN and ENO should **not** be configured.

You will find this EFB in the HVAC library.

2 Representation

2.1 Symbol



2.2 Parameter Specifications

Table 3 CC2_VAC

Parameter	Data Type	Meaning
SP	REAL	Setpoint
CV	REAL	Command variable
PV1	REAL	Actual value P controller
PV2	REAL	Actual value PI controller
NORM	BOOL	Basic setting
MAN_CTR	BOOL	MANUAL for controller
MAN_SEQ	BOOL	MANUAL for sequences
YMAN_PC	REAL	Total manual manipulated variable as a % for controller
YMAN1_PC	REAL	Individual manual manipulated variable as a % for 1 Sequence
YMAN2_PC	REAL	Individual manual manipulated variable as a % for 2 Sequence
PARA	PARA_CC2	Parameter structure
Y1	REAL	Manipulated variable / Output variable 1
Y2	REAL	Manipulated variable / Output variable 2
Y1_PC	REAL	Manipulated variable / Output variable 1 as a %
Y2_PC	REAL	Manipulated variable / Output variable 2 as a %
ERR1	REAL	Control difference P controller
ERR2	REAL	Control difference PI controller

Table 4 PARA_CC2

Element	Data Type	Meaning
SP_SPCV	BOOL	Setpoint / setpoint + command variable
P controller		
YMAX1	REAL	Upper limit manipulated variable
YMIN1	REAL	Lower limit manipulated variable
GAIN1	REAL	Controller gain
PROP1	REAL	Proportional value
PREF1	REAL	Proportional value reference
PI Controller		
YMAX2	REAL	Upper limit manipulated variable
YMIN2	REAL	Lower limit manipulated variable
GAIN2	REAL	Controller gain
PROP2	REAL	Proportional value
PREF2	REAL	Proportional value reference
TI2	TIME	Reset time
1. Sequence		
X1_1	REAL	1. Abcissa value
Y1_1	REAL	1. Ordinate value
X2_1	REAL	2. Abcissa value
Y2_1	REAL	2. Ordinate value
2. Sequence		
X1_2	REAL	1. Abcissa value
Y1_2	REAL	1. Ordinate value
X2_2	REAL	2. Abcissa value
Y2_2	REAL	2. Ordinate value

3 Detailed description

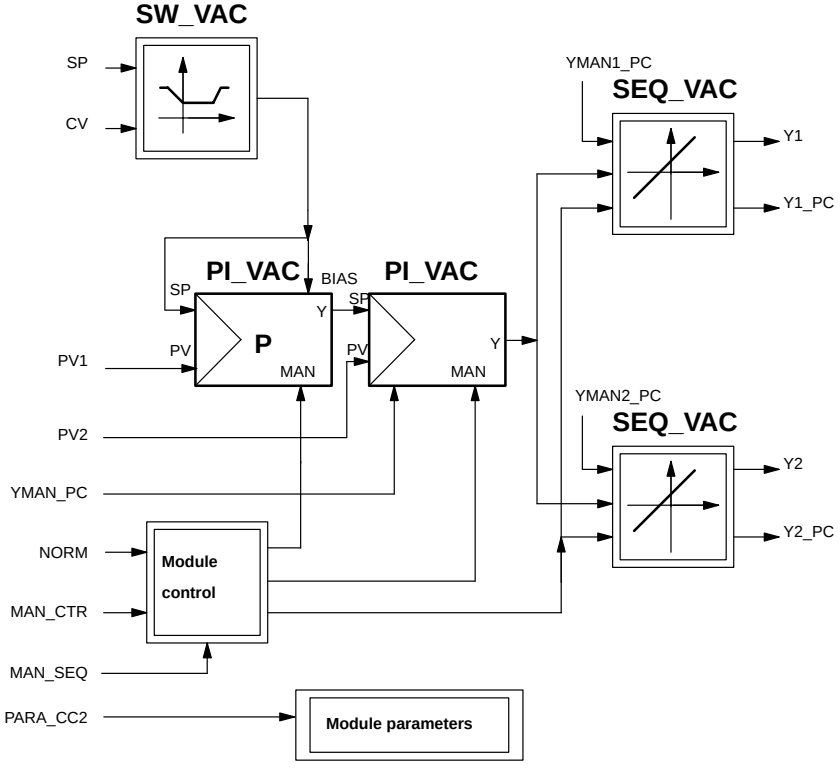
3.1 EFB structure

A block diagram representation of the CC2_VAC complex function block is shown in Figure 1. It is made up of the following basic function blocks:

- SW_VAC Summer/winter compensation (see page 54)
- PI_VAC Basic PI controller for HVAC applications (see page 45)
- SEQ_VAC Output Sequence/scaling module (see page 49)

Please refer to the respective individual basic function block description for detailed information.

Figure 1 Controller structure



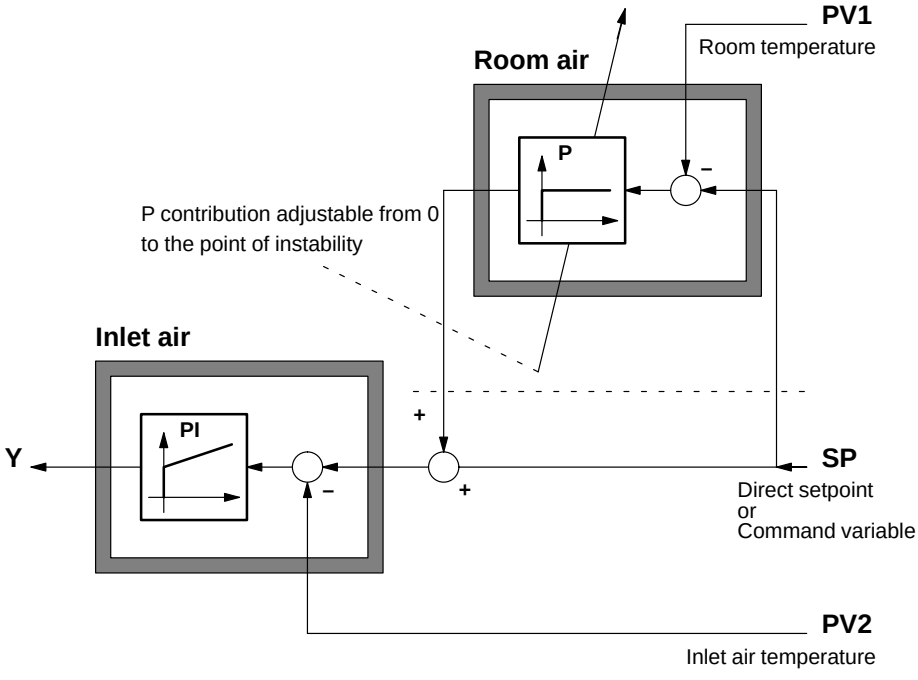
While the CC2_VAC function block can be used for both temperature or humidity control, the remainder of this description refers to temperature control only.

3.2 Basic Operation

The setpoint for the CC2_VAC function block is fed to the P–controller via the summer/winter compensation block SW_VAC (description see page 54).

The output of the SW_VAC block is fed not only to the setpoint of the room P–controller, but also to its BIAS input. In this way, the P–controller setpoint is added to the P–controller output (which is equal to the P–controller error amplified by the controller gain), and the result is fed as setpoint to the inlet air PI–controller. This is shown in Figure 2.

Figure 2 Diagram of setpoint formation



In order to understand the operation of this PPI controller, consider the case of a room with no temperature gains or losses. In this case, the room temperature PV1 would be equal to the PPI controller setpoint, and the inlet air temperature PV2 would be equal to the room temperature PV1. If we now consider the case of a heat source in the room, the room air temperature PV1 will rise to a value greater than the setpoint. As a result, the output of the P–controller, equal to controller error (SP–PV1) multiplied by the GAIN, would trim the setpoint to the PI–controller resulting in a lowering of the inlet air temperature to offset the heat source in the room. Because the inlet air dynamics are much faster than the room air dynamics, a stable steady state condition will be reached under fluctuating room conditions using this cascaded approach. A simple PI–controller using the room temperature as process variable would not achieve satisfactory control

because of the dead–time between the adjustment of the inlet air temperature and a reaction in the room air temperature.

Please note however that while this PPI control algorithm will result in a stable room temperature, there will be a small offset from the PPI controller setpoint, i.e., PV1 will never equal SP. The size of the offset will depend on the disturbance conditions in the room. If an application requires tight control of temperature to an absolute value, the CC2_VAC function block should not be used. However, for applications that do not require tight absolute control, the PPI controller provides a simple, stable and easy–to–use control algorithm.

The PI–controller setpoint can be maintained within minimum and maximum limits by using the P–controller output limits YMAX1 and YMIN1. This facility can be used to prevent large swings in the inlet air temperature.

3.3 Manual Operation

There are 3 possible modes of manual operation which are specified by setting the mutually exclusive NORM, MAN_CTR and MAN_SEQ parameters.

- **NORM mode (NORM = 1)**
  The output of the PI–controller output is set to zero. The zero value is fed to the output sequences that are active in NORM mode. As a result the actual values of Y1 and Y2 that are output to the control actuators will depend on the output sequence parameters. In the case where the controller output limits YMIN2 and YMAX2 have been set to values that do not enclose zero, the output will be set to the value of YMIN2 and YMAX2 that is closest to zero. In other words:

  Y = YMAX2 if YMAX2 < 0 AND YMIN2 < 0,
  Y = YMIN2 if YMAX2 > 0 AND YMIN2 > 0.

  Again, the actual values of Y1 and Y2 that are output to the control actuators will depend on the output sequence parameters.
- **MAN_CTR mode (MAN_CTR=1)**
  The output of the PI controller is set equal to the user–specified parameter YMAN_PC. Again, the output sequences are active in this mode and therefore the actual values of Y1 and Y2 that are output to the control actuators will depend on the output sequence parameters.
- **MAN_SEQ mode (MAN_SEQ=1)**
  The outputs Y1 and Y2 are set to the values specified by the parameters YMAN1_PC and YMAN2_PC, i.e., Y1_PC = YMAN1_PC and Y2_PC = YMAN2_PC.

In MAN_CTR mode, the I contribution of the PI–controller is tracked so that a bumpless transfer back to automatic mode may be carried out. In NORM and MAN–SEQ modes, the I contribution is set to zero.

3.4 Output Parameters

The CC2_VAC outputs Y1 and Y2 are available in real or percentage form (Y1, Y2, Y1_PC and Y2_P2).

The percentage values specified for the manual control of outputs (YMAN_PC, YMAN1_PC and YMAN2_PC) refer to the specified range of the appropriate output. For example, the variable YMAN_PC sets the total output of the PI_VAC basic function block. The range of this output is specified by the parameters YMIN (0%) and YMAX (100%).

The variables YMAN1_PC and YMAN2_PC set the outputs of the two SEQ_VAC blocks, the ranges of which are specified by their corresponding ordinate vales Y1..Y2 (where Y1 and Y2 refer to the SEQ_VAC ordinates, not the actual outputs Y1 and Y2 of the two SEQ_VAC blocks). It is important to note that the smaller value of Y1 and Y2 is always equated to 0% and the greater value to 100%. In other words, the percentage value is related to the size of the output variable Y irrespective of its direction:

Y1 < Y2 –> 0%..100% = Y1..Y2

Y1 > Y2 –> 0%..100% = Y2..Y1

For more information on the operation of the SEQ_VAC block, please refer to the respective description (see page 49).

CC3_VAC
Cascade controller for air conditioning with 3 outputs

1 Brief description

The CC3_VAC module is a cascade controller used to provide temperature or humidity control of the inlet air to a room. It consists of a P–only outer loop which uses the room temperature/humidity as process variable and an inner PI loop that controls the temperature/humidity of the inlet air supplying the room.

The EFB has a fixed structure where the setpoint of the P–controller is fed forward and added to the P–controller output to form the setpoint of the PI–controller. The output of the PI–controller has 3 output sequences, Y1, Y2 and Y3. The operation of CC3_VAC is identical to CC2_VAC, the only difference being the additional output.

The EFB provides the following :

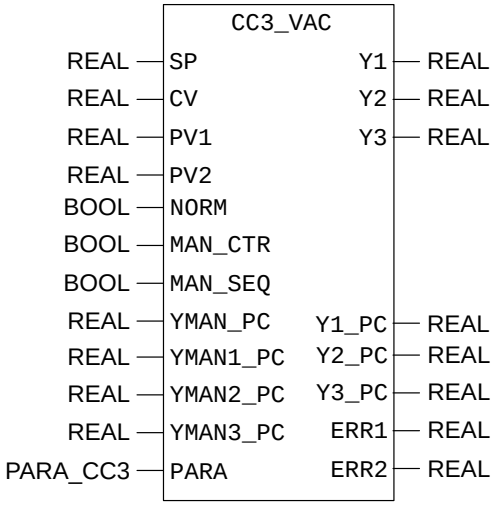
- Winter/summer setpoint compensation as per DIN 1946 part 2.
- Full four quadrant operation of the output sequence scaling
- The display of output variables as percentages
- Upper and lower limits on outputs
- Presetting the controllers gains in the form of GAIN or PROP with the possibility of using negative values for switching the control direction.
- Operation with Anti–Windup–Reset (AWR)
- Manual adjustment of either the PI–controller output or the individual sequence outputs (Y1 and Y2) using percentages. When the controller output is manually adjusted, the EFB tracks the I contribution in order to provide bump–less switching back to automatic mode.
- The display of the P and PI controller errors (SP–PV).

Note
Additional parameters EN and ENO should **not** be configured.

You will find this EFB in the HVAC library.

2 Representation

2.1 Symbol



2.2 Parameter Specifications

Table 1 CC3_VAC

Parameter	Data Type	Meaning
SP	REAL	Setpoint
CV	REAL	Command variable
PV1	REAL	Actual value P controller
PV2	REAL	Actual value PI controller
NORM	BOOL	Basic setting
MAN_CTR	BOOL	MANUAL for controller
MAN_SEQ	BOOL	MANUAL for sequences
YMAN_PC	REAL	Total manual manipulated variable as a % for controller
YMAN1_PC	REAL	Individual manual manipulated variable as a %, 1st sequence
YMAN2_PC	REAL	Individual manual manipulated variable as a %, 2nd sequence
YMAN3_PC	REAL	Individual manual manipulated variable as a %, 3rd sequence
PARA	PARA_CC3	Parameter structure
Y1	REAL	Manipulated variable / Output variable 1
Y2	REAL	Manipulated variable / Output variable 2
Y3	REAL	Manipulated variable / Output variable 3
Y1_PC	REAL	Manipulated variable / Output variable 1 as a %
Y2_PC	REAL	Manipulated variable / Output variable 2 as a %
Y3_PC	REAL	Manipulated variable / Output variable 3 as a %
ERR1	REAL	Control difference P controller
ERR2	REAL	Control difference PI controller

Table 2 PARA_CC3

Element	Data Type	Meaning
SP_SPCV	BOOL	Setpoint / setpoint + command variable
P controller		
YMAX1	REAL	Upper limit manipulated variable
YMIN1	REAL	Lower limit manipulated variable
GAIN1	REAL	Controller gain
PROP1	REAL	Proportional value
PREF1	REAL	Proportional value reference
PI Controller		
YMAX2	REAL	Upper limit manipulated variable
YMIN2	REAL	Lower limit manipulated variable
GAIN2	REAL	Controller gain
PROP2	REAL	Proportional value
PREF2	REAL	Proportional value reference
TI2	TIME	Reset time
1. Sequence		
X1_1	REAL	1. Abcissa value
Y1_1	REAL	1. Ordinate value
X2_1	REAL	2. Abcissa value
Y2_1	REAL	2. Ordinate value
2. Sequence		
X1_2	REAL	1. Abcissa value
Y1_2	REAL	1. Ordinate value
X2_2	REAL	2. Abcissa value
Y2_2	REAL	2. Ordinate value
3. Sequence		
X1_3	REAL	1. Abcissa value
Y1_3	REAL	1. Ordinate value
X2_3	REAL	2. Abcissa value
Y2_3	REAL	2. Ordinate value

3 Detailed description

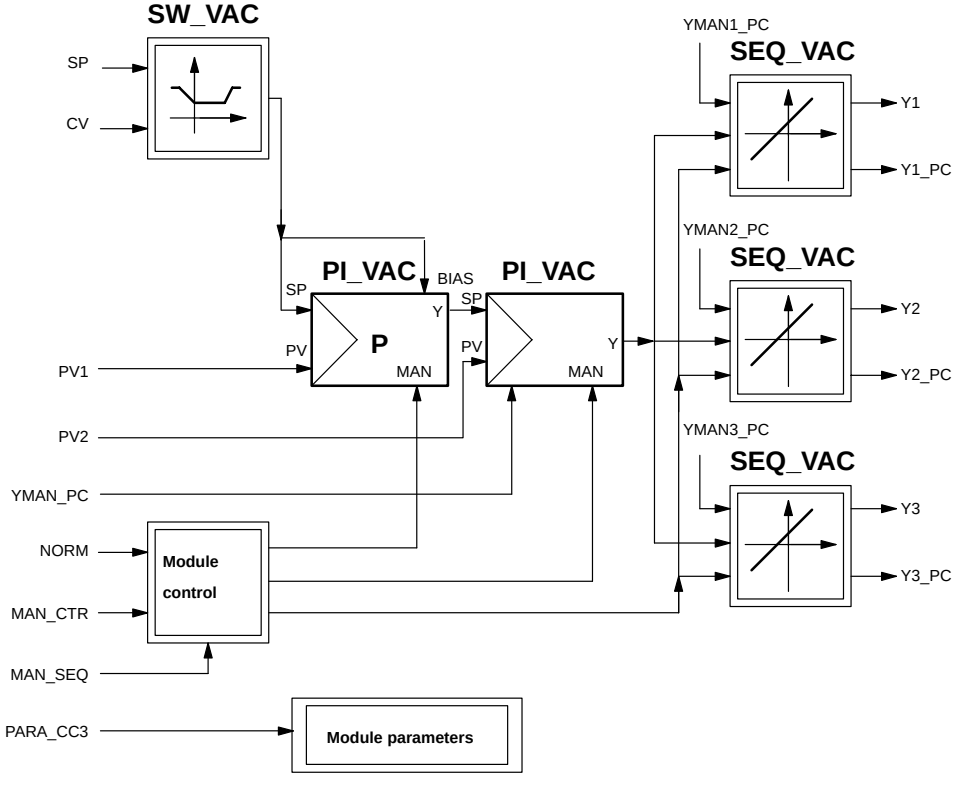
3.1 EFB structure

A block diagram representation of the CC3_VAC complex function block is shown in Figure 1. It is made up of the following basic function blocks:

- SW_VAC Summer/winter compensation (see page 54)
- PI_VAC Basic PI controller for HVAC applications (see page 45)
- SEQ_VAC Output Sequence/scaling module (see page 49)

Please refer to the respective individual basic function block description for detailed information.

Figure 1 Controller structure



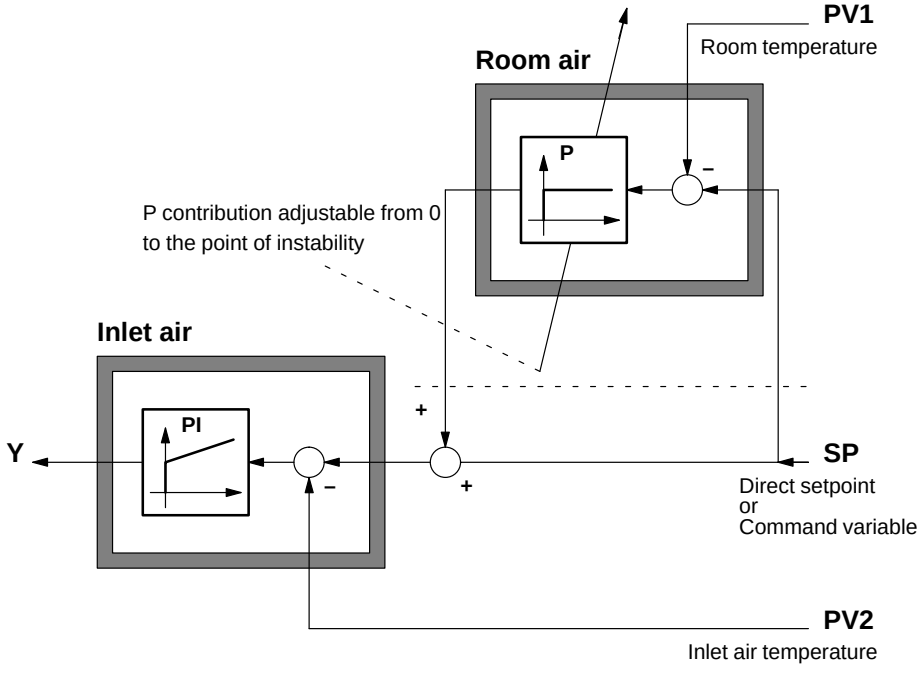
While the CC3_VAC function block can be used for both temperature or humidity control, the remainder of this description refers to temperature control only.

3.2 Basic Operations

The setpoint for the CC3_VAC function block is fed to the P–controller via the summer/winter compensation block SW_VAC (description see page 54).

The output of the SW_VAC block is fed not only to the setpoint of the room P–controller, but also to its BIAS input. In this way, the P–controller setpoint is added to the P–controller output (which is equal to the P–controller error amplified by the controller gain), and the result is fed as setpoint to the inlet air PI–controller. This is shown in Figure 2.

Figure 2 Diagram of setpoint formation



In order to understand the operation of this PPI controller, consider the case of a room with no temperature gains or losses. In this case, the room temperature PV1 would be equal to the PPI controller setpoint, and the inlet air temperature PV2 would be equal to the room temperature PV1. If we now consider the case of a heat source in the room, the room air temperature PV1 will rise to a value greater than the setpoint. As a result, the output of the P–controller, equal to controller error (SP–PV1) multiplied by the GAIN, would trim the setpoint to the PI–controller resulting in a lowering of the inlet air temperature to offset the heat source in the room. Because the inlet air dynamics are much faster than the room air dynamics, a stable steady state condition will be reached under fluctuating room conditions using this cascaded approach. A simple PI–controller using the room temperature as process variable would not achieve satisfactory control

because of the dead–time between the adjustment of the inlet air temperature and a reaction in the room air temperature.

Please note however that while this PPI control algorithm will result in a stable room temperature, there will be a small offset from the PPI controller setpoint, i.e., PV1 will never equal SP. The size of the offset will depend on the disturbance conditions in the room. If an application requires tight control of temperature to an absolute value, the CC3_VAC function block should not be used. However, for applications that do not require tight absolute control, the PPI controller provides a simple, stable and easy–to–use control algorithm.

The PI–controller setpoint can be maintained within minimum and maximum limits by using the P–controller output limits YMAX1 and YMIN1. This facility can be used to prevent large swings in the inlet air temperature.

3.3 Manual Operation

There are 3 possible modes of manual operation which are specified by setting the mutually exclusive NORM, MAN_CTR and MAN_SEQ parameters.

- **NORM mode (NORM = 1)**
  The output of the PI–controller output is set to zero. The zero value is fed to the output sequences that are active in NORM mode. As a result the actual values of Y1 and Y2 that are output to the control actuators will depend on the output sequence parameters. In the case where the controller output limits YMIN2 and YMAX2 have been set to values that do not enclose zero, the output will be set to the value of YMIN2 and YMAX2 that is closest to zero. In other words:

  Y = YMAX2 if YMAX2 < 0 AND YMIN2 < 0,
  Y = YMIN2 if YMAX2 > 0 AND YMIN2 > 0.

  Again, the actual values of Y1 and Y2 that are output to the control actuators will depend on the output sequence parameters.
- **MAN_CTR mode (MAN_CTR=1)**
  The output of the PI controller is set equal to the user–specified parameter YMAN_PC. Again, the output sequences are active in this mode and therefore the actual values of Y1, Y2 and Y3 that are output to the control actuators will depend on the output sequence parameters.
- **MAN_SEQ mode (MAN_SEQ=1)**
  The outputs Y1, Y2 and Y3 are set to the values specified by the parameters YMAN1_PC, YMAN2_PC and YMAN3_PC, i.e., Y1_PC = YMAN1_PC, Y2_PC = YMAN2_PC and Y3_PC = YMAN3_PC.

In MAN_CTR mode, the I contribution of the PI–controller is tracked so that a bumpless transfer back to automatic mode may be carried out. In NORM and MAN–SEQ modes, the I contribution is set to zero.

3.4 Output Parameters

The CC3_VAC outputs Y1, Y2 and Y3 are available in real or percentage form (Y1, Y2, Y3, Y1_PC, Y2_PC and Y3_PC).

The percentage values specified for the manual control of outputs (YMAN_PC, YMAN1_PC, YMAN2_PC and YMAN3_PC) refer to the specified range of the appropriate output. For example, the variable YMAN_PC sets the total output of the PI_VAC basic function block. The range of this output is specified by the parameters YMIN (0%) and YMAX (100%).

The variables YMAN1_PC, YMAN2_PC and YMAN3_PC set the outputs of the two SEQ_VAC blocks, the ranges of which are specified by their corresponding ordinate vales Y1..Y2 (where Y1 and Y2 refer to the SEQ_VAC ordinates, not the actual outputs Y1, Y2 and Y3 of the two SEQ_VAC blocks). It is important to note that the smaller value of Y1 and Y2 is always equated to 0% and the greater value to 100%. In other words, the percentage value is related to the size of the output variable Y irrespective of its direction:

Y1 < Y2 –> 0%..100% = Y1..Y2

Y1 > Y2 –> 0%..100% = Y2..Y1

For more information on the operation of the SEQ_VAC block, please refer to the respective description (see page 49).

MC_VAC
Air mix controller for air conditioning with one output

1 Brief description

The MC_VAC module is a controller designed for applications that require switching of the controller direction, and maximum/minimum selection of outputs. Examples of such applications would be air mixing controls where the control direction will depend on the relative values of the outside air temperature and return air temperature, or heat exchanger controls used in energy conservation schemes.

The EFB provides the following :

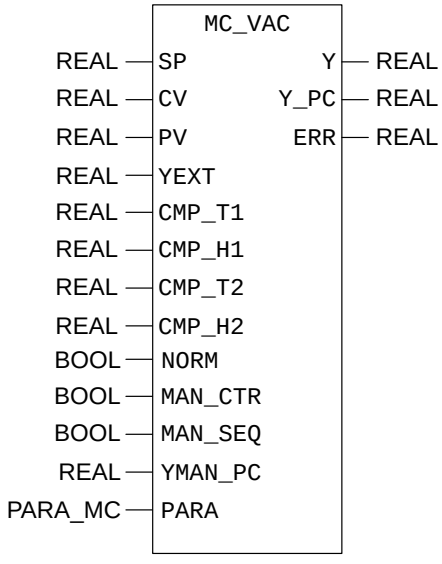
- Winter/summer setpoint compensation as per DIN 1946 part 2.
- Full four quadrant operation of the output sequence scaling
- The display of output variables as percentages
- Upper and lower limits on outputs
- Presetting the controllers' gains in the form of GAIN or PROP with the possibility of using negative values for switching the control direction.
- Control reversal using comparison inputs with built–in hysterisis
- Selection of operating mode as a "Direct Controller" or "Auxiliary Controller" with Max/Min selection.
- Operation with Anti–Windup–Reset (AWR)
- Manual adjustment of either the PI–controller output or the individual sequence outputs (Y1 and Y2) using percentages. When the controller output is manually adjusted, the EFB tracks the I contribution in order to provide bumpless switching back to automatic mode.
- The display of the P and PI controller errors (SP–PV).

☞ **Note**
Additional parameters EN and ENO should **not** be configured.

You will find this EFB in the HVAC library.

2 Representation

2.1 Symbol



2.2 Parameter Specifications

Table 1 MC_VAC

Parameter	Data Type	Meaning
SP	REAL	Setpoint
CV	REAL	Command variable
PC	REAL	Actual value
YEXT	REAL	Manipulated variable of higher–level controller in the case of cascade mode
CMP_T1	REAL	Reference value T1
CMP_H1	REAL	Reference value H1
CMP_T2	REAL	Reference value T2
CMP_H2	REAL	Reference value H2
NORM	BOOL	Basic setting
MAN_CTR	BOOL	MANUAL for controller
MAN_SEQ	BOOL	MANUAL for sequence
YMAN_PC	REAL	Total manual manipulated variable as a % or sequence manual manipulated variable as a %
PARA	PARA_MC	Parameter structure
Y	REAL	Manipulated variable / Output variable
Y_PC	REAL	Manipulated variable / Output variable as a %
ERR	REAL	System Deviation

Table 2 PARA_MC

Element	Data Type	Meaning
CASC	BOOL	Direct / cascade mode
SP_SPCV	BOOL	Setpoint / setpoint + command variable
MIN	BOOL	Max/min switch (max=0)
PI Controller		
YMAX	REAL	Upper limit of manipulated variable (YAU always 0)
GAIN	REAL	Controller gain
PROP	REAL	Proportional value
PREF	REAL	Proportional value reference
TI	TIME	Reset time
Output sequence		
X1	REAL	1. Abcissa value
Y1	REAL	1. Ordinate value
X2	REAL	2. Abcissa value
Y2	REAL	2. Ordinate value

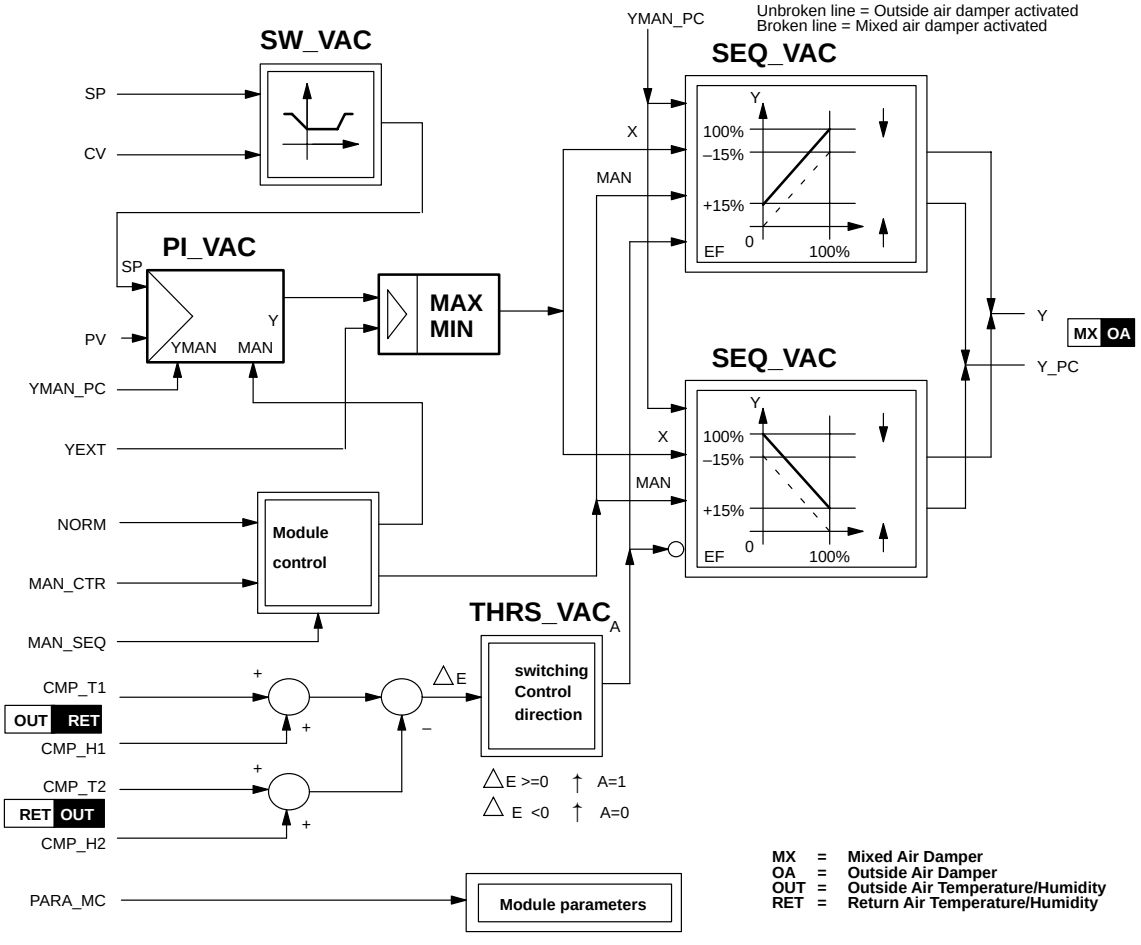
3 Detailed description

3.1 EFB structure

A block diagram representation of the CC2_VAC complex function block is shown in Figure 1. It is made up of the following basic function blocks:

- SW_VAC Summer/winter compensation (see page 54)
- PI_VAC Basic PI controller for HVAC applications (see page 45)
- THRS_VAC Limit Selector with hysteresis (see page 57)
- SEQ_VAC Output Sequence/scaling module (see page 49)

Figure 1 Controller structur



Please refer to the respective individual basic function block description for detailed information.

3.2 Setpoint

The setpoint for the MC_VAC function block is fed to the PI–controller via the summer/winter compensation block SW_VAC (description see page 54).

The PI–controller setpoint can be maintained within minimum and maximum limits by using the P–controller output limits YMAX1 and YMIN1. This facility can be used to prevent large swings in the inlet air temperature as a result of unusual room temperature fluctuations or a badly adjusted P–controller.

3.3 Control Direction

The function block has 4 reference variables CMP–T1, CMP_H1, CMP_T2 and CMP_H2. CMP_T1 is added to CMP_H1 and CMP_T2 is added to CMP_H2. The resulting additions are then compared with one another, and the result of the comparison is fed to the THRS_VAC block to provide switching with a built in hysteresis of 1.0. The output of the THRS_VAC block is used to ”reverse” the action of the output sequences.

The user is free to choose what inputs are connected to the reference variables. In general though, temperature comparisons should use CMP_T1 and CMP_T2 while humidity comparisons should use CMP_H1 and CMP_H2. If temperature comparison only is required, the variables CMP_H1 and CMP_H2 can be set to zero.

The comparison inputs can be used to compare the heat content of 2 air streams. The heat content of air can be calculated as follows:

$Q = Q_a + Q_{sw} + Q_{lw}$

Where
Q_a = sensible heat content of air (kJ)
Q_{sw} = sensible heat content of water vapor (kJ)
Q_{lw} = latent heat content of water vapor (kJ)

Therefore

$Q = m * C_a * T_a + m * x * C_w * T_a + m * x * h_w$

Where
m = mass of air (kg)
C_a = specific heat capacity of air = 1.006 kJ/(kg*K)
T_a = temperature of air (deg C)
x = absolute humidity (g/kg)
C_w = specific heat capacity of water vapour = 1.92 kJ/(kg*K)
h_w = specific enthalpy of water vapour = 2500 kJ/kg = 2.5 kJ/g

Bearing in mind that C_a is approximately equal to one and that h_w is much greater than C_w*T_a, the equation can be simplified to :

$Q = m * T_a + m * h_w * x$

Therefore, the enthalpy H of the air can be calculated as :

$H = Q / m = T_a + h_w * x = T_a + 2.5x$

Therefore by scaling the air temperature in degrees Celsius and by multiplying the absolute humidity (scaled in g/kg) by 2.5 the resultant addition gives a good indication of the enthalpy content of the air in kJ/kg. In this case, the built in hysteresis corresponds to 1.0 kJ/kg.

In cases where an accurate value for hysteresis is not necessary, the absolute humidity can be substituted by its corresponding dew point temperature. This can be obtained by the use of WASH_VAC, feeding the **measured** relative humidity to SP_RH and the measured temperature T_a to SP_RT. The resulting SP_TW replaces in this case the term 2.5x.

$H = T_a + SP\_TW$

SP_TW represents a nonlinear but fixed function or the unknown absolute humidity. Because both values of H, from RETURN AIR and OUTSIDE AIR, are calculated in the same way, the comparison of both values does not need to take care that these calculated values themselves are nonlinear to the true H values.

During the first execution cycle of the function block, the hysteresis is initialized on the assumption that CMP_T1 < CMP_T2. If after the first calculation cycle the comparison result is within the hysteresis band, the initial switching status is retained.

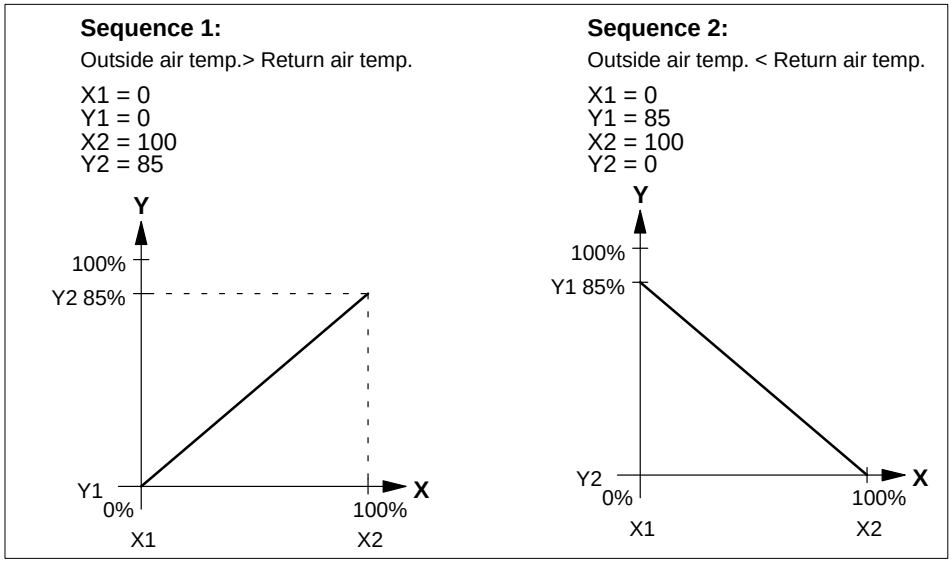
3.4 Output Sequences

☞ **Note**
Only one output sequence is specified by the user. The function block will then calculate the reversed sequence when the control direction is switched. The sequence specified must have an increasing slope, i.e., Y2>Y1.

In order to understand how the output sequencing works, the example of temperature control using a mixing air damper can be used. For a mixing air controller, there are normally 2 dampers – a control damper for mixing the return air with the outside air and a control damper on the outside air inlet (see examples on page 40). The control action on one damper is the reverse of the other. In practical terms, the controller output is sent to 1 damper only, the other damper being controlled mechanically or electrically. It therefore follows that the output sequences will depend on which damper is being controlled directly. Also, a minimum outside air flow is always guaranteed to provide some fresh air into the room. Assuming an outside air minimum of 15%, the output sequences would look like the following :

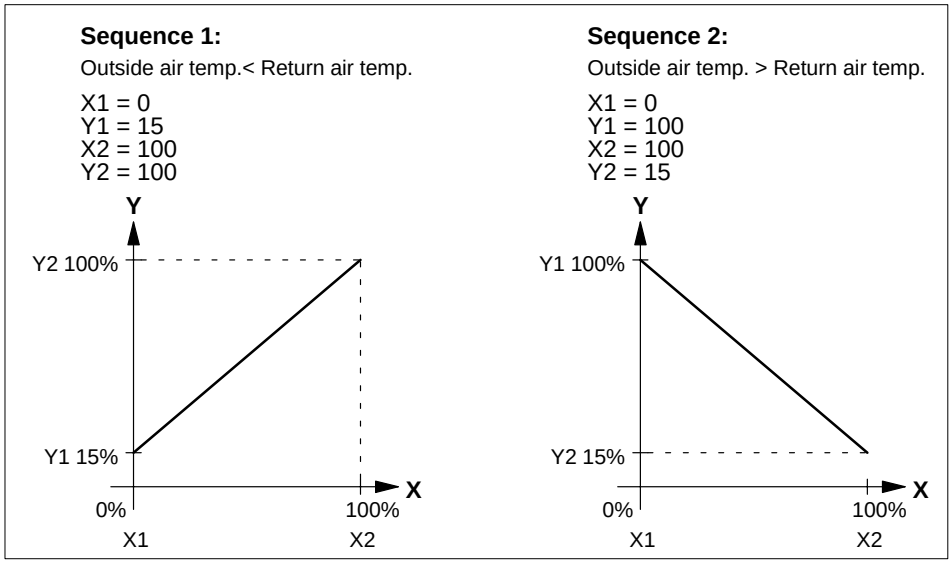
- If the air mixer damper is controlled, the sequences would be :

Figure 2 Air Mixing Damper



- If the outside air damper is controlled, the sequences would be :

Figure 3 Outside Air Damper



The reference values CMP_T1 and CMP_T2 will determine which of the sequences are activated as follows:

CMP_T1 > CMP_T2 => Sequence 1 is active

CMP_T1 < CMP_T2 => Sequence 2 is active

In order to get the desired control, the comparison inputs should be connected as follows:

- If the **air mixer damper** is controlled:
  CMP_T1 = Outside air temperature
  CMP_T2 = Return air temperature
- If the **outside air damper** is controlled:
  CMP_T1 = Return air temperature
  CMP_T2 = Outside air temperature

3.5 Direct and Auxiliary control

Direct or Auxiliary control is selected using the CASC parameter.

Direct control is specified by setting CASC = 0. In this mode, the Max/Min selection is disable and the external input YEXT is ignored.

Auxiliary control is specified by setting CASC = 1. In this mode, the Max/Min selection is enabled and the external input YEXT is compared to the PI controller output.

3.6 Manual operation

There are 3 possible modes of manual operation which are specified by setting the mutually exclusive NORM, MAN_CTR and MAN_SEQ parameters.

- **NORM mode (NORM = 1):**
  The output of the PI–controller output is set to zero. However, the actual value forwarded to the output sequences depends on whether the controller is set up in ”Direct” or ”Auxiliary mode. In ”Direct” mode the PI–controller output is fed through the Max/Min block to the output sequence. In ”Auxiliary” mode, the value passed through to the output sequence depends on the result of the comparison between the controller output and the value of YEXT which is set by the higher level controller. Therefore, in order to operate in true NORM mode, both the Auxiliary controller and the higher level controller must be set to NORM. The actual value that is output to the control actuator depends on the output sequence parameters.
- **MAN_CTR mode (MAN_CTR=1):**
  The output of the PI controller is set equal to the user–specified parameter YMAN_PC. However, the YEXT value coming from the higher level controller may override the YMAN_PC value. Again, the output sequence is active in this mode and therefore the actual value of Y that is output to the control actuator will depend on the output sequence parameters and the control direction.

- **MAN_SEQ mode (MAN_SEQ=1):**
  The sequence output Y is set to the value specified by the parameter YMAN_PC, i.e., Y_PC = YMAN_PC. When in MAN_SEQ mode, switching the control direction has no effect on the output.

In MAN_CTR mode, the I contribution of the PI–controller is tracked so that a bump–less transfer back to automatic mode may be carried out. In NORM and MAN–SEQ modes, the I contribution is set to zero.

3.7 Output Parameters

The MC_VAC output Y is available in real or percentage form (Y, and Y_PC).

The percentage values specified for the manual control of outputs (YH_PC, and YMAN_PC) refer to the specified range of the appropriate output. For example, the variable YH_PC sets the total output of the PI_VAC basic function block. The range of this output is specified by the parameters YMIN (0%) and YMAX (100%).

The variable YMAN_PC sets the output of the SEQ_VAC block, the range of which is specified by the corresponding ordinate vales Y1..Y2 (where Y1 and Y2 refer to the SEQ_VAC ordinates). It is important to note that the smaller value of Y1 and Y2 is always equated to 0% and the greater value to 100%. In other words, the percentage value is related to the size of the output variable Y irrespective of its direction:

Y1 < Y2 –> 0%..100% = Y1..Y2

Y1 > Y2 –> 0%..100% = Y2..Y1

For more information on the operation of the SEQ_VAC block, please refer to the respective description (see page 49).

3.8 Example Applications

You will find two examples for using the MC_VAC block:

- Air Mixing
- Heat Recovery Exchangers

☞ **Note**

In both of the examples, the user must carefully select all control actuators in order to provide failsafe operation and prevent freezing of water coils. Hardwired antifreeze controls should also be used to override the PLC controls and if necessary shutdown the make up air handling unit to protect the coils.

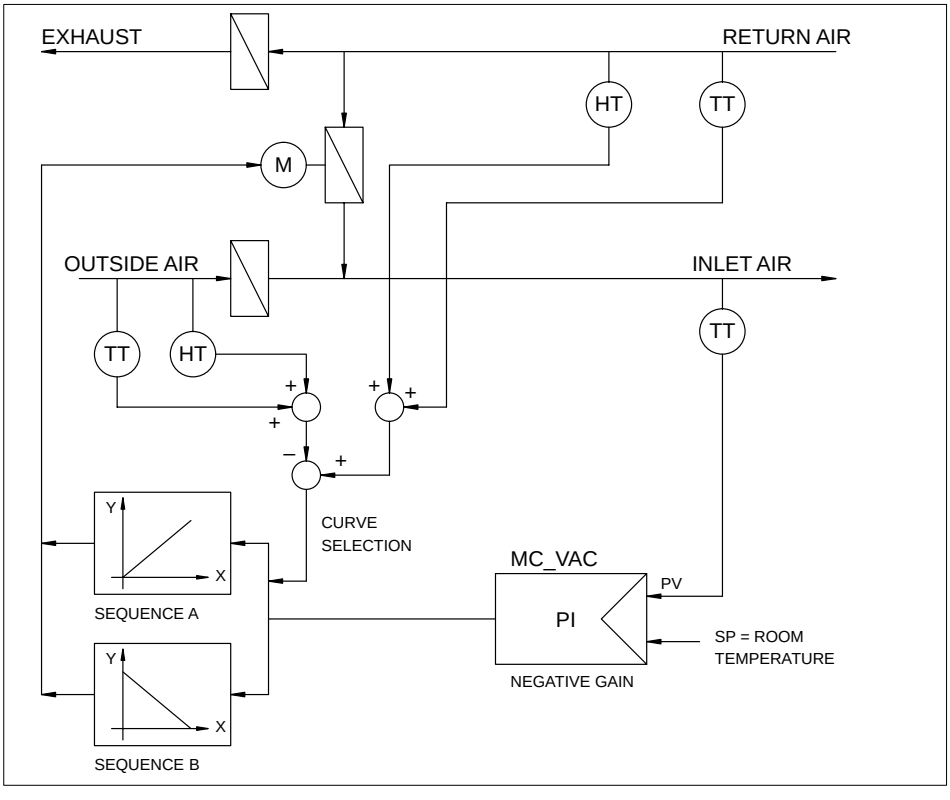
Example:

Air Mixing

An example where switching of control direction is commonly used is air mixing and is shown in Figure 4. In this application, return air from a room is mixed with the outside air to supply the inlet air to the room. In the winter, the outside air is at a lower temperature and humidity than the return air. The return air is therefore mixed with the outside air to provide free heating of the outside air. In the summer, the return air may have a lower temperature and humidity than the outside air. In this case, the return air is mixed with the outside air to provide free cooling. The example shows the use of temperature and humidity sensors on the return air and outside air being used to determine the control direction of the controller.

The example also shows control of the mixing air damper. It is then assumed that the outside air damper is connected mechanically or electro–mechanically to the mixing air damper.

Figure 4 Example for Air Mixing



In the winter, if H_OUTSIDE < H_RETURN, sequence B is used. In this case it can be seen that if the inlet air temperature is too low, the PI controller output (whose gain is

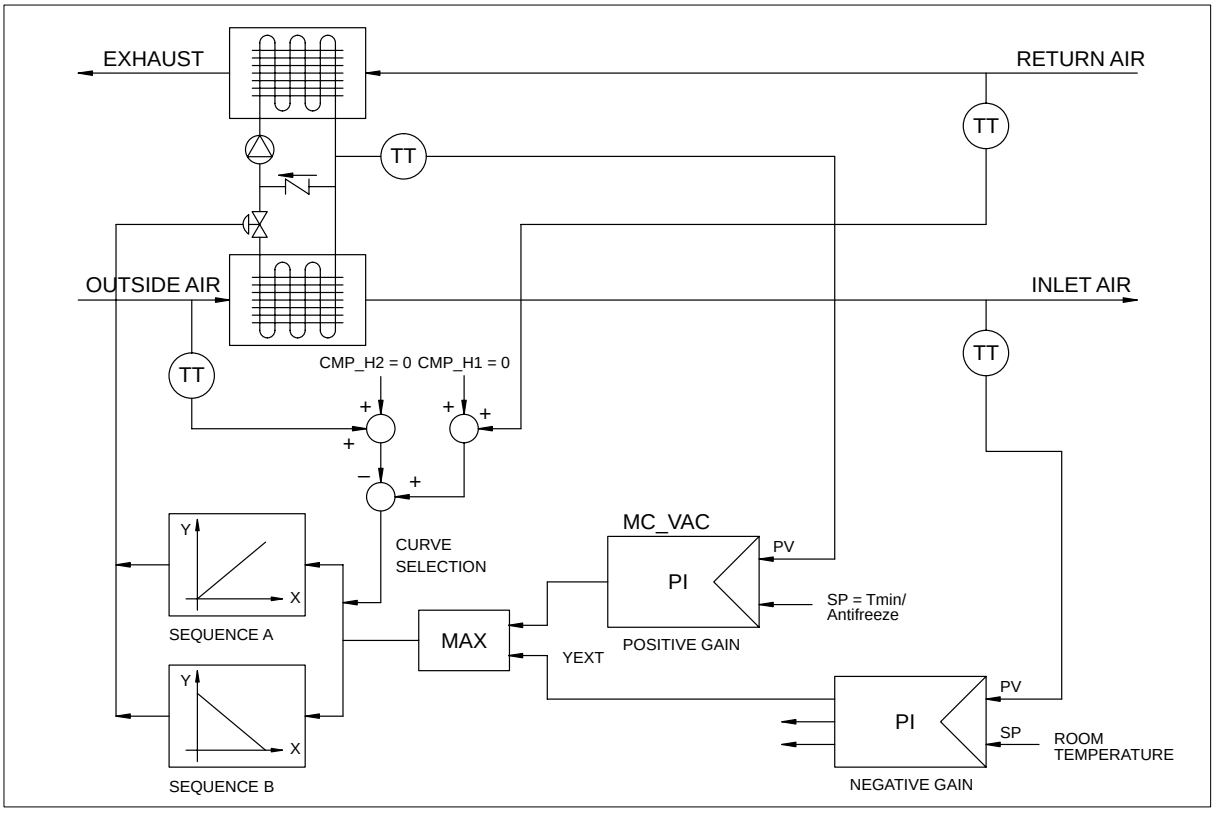
negative) will be increasingly negative, opening the mixing air damper in order to mix a greater quantity of warmer air with the colder outside air, thereby increasing the inlet air temperature. Conversely, if the inlet air temperature is too warm, the PI controller output will be increasingly more positive, resulting in a closing of the mixing air damper in order to mix a smaller quantity of the warmer return air with the cooler outside air, thereby decreasing the inlet air temperature.

In the summer, if H_OUTSIDE > H_RETURN, sequence A is used. In this case it can be seen that if the inlet air temperature is too low, the PI controller output (whose gain is negative) will be increasingly negative, closing the mixing air damper in order to mix a smaller quantity of colder return air with the warmer outside air, thereby increasing the inlet air temperature. Conversely, if the inlet air temperature is too warm, the PI controller output will be increasingly more positive, resulting in the opening of the mixing air damper in order to mix a greater quantity of cooler return air with the warmer outside air, thereby decreasing the inlet air temperature.

Example: **Heat Recovery Exchangers**

Another example of heat recovery exchangers is shown in Figure 5. In this example, a closed water loop between two heat exchangers is used to transfer heat between the return air and incoming outside air. A re–circulation pump is used in combination with a check valve and control valve. Closing the control valve will result in more of the water circulating through the check valve and less through the outside air heat exchanger. Conversely, opening the control valve, will result in more water circulating through the outside air heat exchanger.

Figure 5 Example Heat Recovery Exchangers



The example shows two control loops – an inlet air controller and an anti–freeze controller. The anti–freeze controller measures the water loop temperature and is used to guarantee a minimum temperature so as to prevent freezing of return air condensed water on the return air exchanger surfaces. The setpoint is set to a suitable value above 8the freezing point of water. Under normal conditions, the PV will be greater than the SP, the controller output will be zero, and the MAX selection block will ensure that the inlet air controller controls the heat exchangers. If however, the water temperature decreases

below the minimum setpoint, the anti–freeze controller output will increase and take over from the inlet air controller to maintain the minimum water temperature through the return air heat exchanger, thereby preventing freezing of any condensation that may have occurred.

The inlet air controller controls the inlet air temperature. An MC_VAC block is used for the anti–freeze controller and is run in CASCADE mode. The output of the inlet air controller is fed as the input Yext to the anti–freeze controller, thereby controlling the heat exchanger control valve.

If T_OUTSIDE < T_RETURN, sequence B is used. In this case it can be seen that if the inlet air temperature is too low, the PI controller output (whose gain is negative) will be increasingly negative, opening the control valve in order to divert a greater quantity of warmer water to the outside air heat exchanger, thereby increasing the inlet air temperature. Conversely, if the inlet air temperature is too warm, the PI controller output will be increasingly more positive, resulting in a closing of the control valve in order to divert a smaller quantity of the warmer water to the outside air heat exchanger, thereby decreasing the inlet air temperature.

If T_OUTSIDE > T_RETURN, sequence A is used. In this case it can be seen that if the inlet air temperature is too low, the PI controller output (whose gain is negative) will be increasingly negative, closing the control valve in order to divert a smaller quantity of cooler water to the outside air heat exchanger, thereby increasing the inlet air temperature. Conversely, if the inlet air temperature is too warm, the PI controller output will be increasingly more positive, resulting in the opening of the control valve in order to divert a greater quantity of the cooler water to the outside air heat exchanger, thereby decreasing the inlet air temperature.

PI_VAC
PI controller for air conditioning

1 Brief description

The PI_VAC basic function block is a general PI–controller designed for air conditioning applications. It can be used for temperature control, humidity control, air mixing control or other general functions. As it is a basic function block it consists of a PI controller only with no setpoint compensation, output sequencing, or other functions. It is used extensively by the complex function blocks.

Where the standard complex function block library does not meet the requirements of a particular application, the PI_VAC basic function block can be combined with other basic function blocks to create the user's own complex function block using the Concept Derived Function Block facilities.

The EFB provides the following :

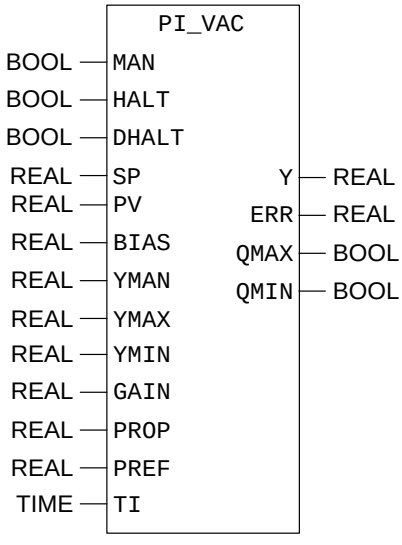
- SP (setpoint), PV (process variable) and BIAS inputs.
- Ability to operate in P, I, or PI modes
- Bump–less initialization of the I contribution as well as bump–less switching between I and PI operation
- Presetting the controller's gain in the form of GAIN or PROP with the possibility of using negative values for switching the control direction.
- Bump–less switching between GAIN and PROP operation in the case of PI control.
- Operation with Anti–Windup–Reset (AWR)
- Manual operation mode with tracking of the I contribution in order to provide bump–less switching back to automatic mode.
- HALT operating mode for freezing the controller output at its current value with tracking of the I contribution
- DYNAMIC HALT operating mode that prevents step–changes in the controller output when the setpoint is changed.
- Upper and lower limits on controller output with indication when limits are reached using the QMAX and QMIN outputs.
- The display of the PI controller error (SP–PV).

☞ **Note**
Additional parameters EN and ENO should **not** be configured.

You will find this EFB in the HVAC library.

2 Representation

2.1 Symbol



2.2 Parameter Specifications

Parameter	Data Type	Meaning
MAN	BOOL	MANUAL mode
HALT	BOOL	HALT mode
DHALT	BOOL	Dyn. HALT for next scanning step
SP	REAL	Setpoint
PV	REAL	Actual value
BIAS	REAL	DEVIATION (deviation compensation)
YMAN	REAL	Manual manipulated variable
YMAX	REAL	Upper limit manipulated variable
YMIN	REAL	Lower limit manipulated variable
GAIN	REAL	Controller gain
PROP	REAL	Proportional value
PREF	REAL	Proportional value reference
TI	TIME	Integral Time
Y	REAL	Manipulated variable
ERR	REAL	Control deviation
QMAX	BOOL	Upper limit of signalling device reached
QMIN	BOOL	Lower limit of signalling device reached

3 Detailed description

3.1 Parameter Specifications

SP, PV, BIAS

The setpoint SP and process variable PV are connected directly to the function block as real values. The controller error is calculated as:

ERR = SP – PV.

The error value is available for display.

A BIAS value may also be specified as a real value. This BIAS value is added to the result of the PI calculation. In this way, the user can ensure that the controller output is ”biased” with a specific value during the first cycle. This is shown in Figure 6.

GAIN, PROP, PREF, TI

The controller gain can be specified by either setting a value for the GAIN, or alternatively, by specifying the proportional rate using the PROP and PREF parameters as follows:

- **If GAIN is not equal to zero**, then the GAIN mode is activated. In this case, the controller output for P–only control is then calculated as :
  Y = GAIN * (SP – PV)
  A negative value of GAIN may be specified to reverse the action of the controller.
- **If GAIN = 0 and PROP is not equal to zero**, then the PROP mode is activated. In this mode, a change in the PV by an amount PROP will result in a change of the controller output by an amount PREF. In other words, for a P–only controller , the output is calculated as :
  Y = PREF * (SP – PV) / PROP
  In the case where an actuator is connected directly to the output of the controller, a PREF of 100 can be set to indicate 100% output.
  A negative value of PROP may be specified to reverse the action of the controller. However, PREF must always be specified as positive.
- **If GAIN = 0 and PROP = 0**, the proportional value of the controller is effectively switched off. By specifying an integral time using the TI input, the controller will act as an I–only controller. In the event that the integral time TI is set to zero, there will be no PI action at all, and only the BIAS value will be forwarded to the controller output. In this case, the operating modes MAN, HALT and DHALT will still be active.

3.2 Manual Operation

There are 3 possible modes of manual operation which are specified by setting the mutually exclusive MAN, HALT and DHALT parameters.

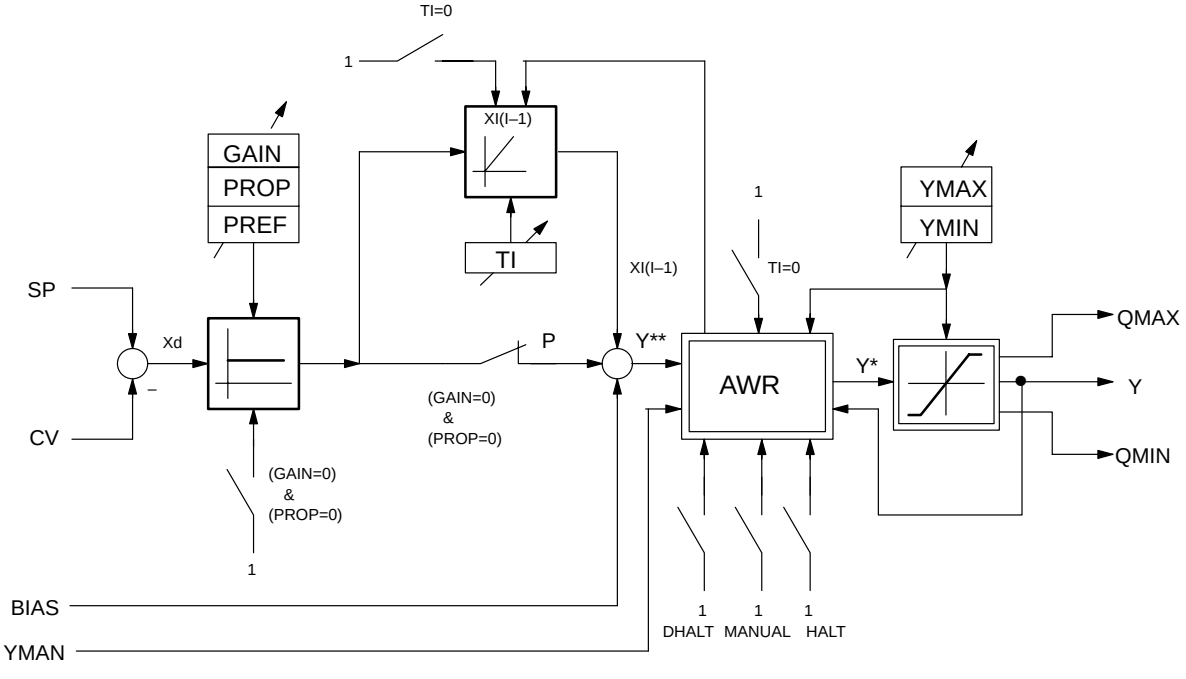
- **MAN mode (MAN = 1):**
  The value specified at YMAN is written to the controller loop output Y. This is effective in both P and PI modes. The I contribution is continually tracked to allow bump–less switching of the controller back to automatic mode. The output limits and anti–reset–windup are therefore still active in MAN mode.

- **HALT mode (HALT = 1):**
  In HALT mode the controller output is frozen at its current value. The I contribution is tracked to allow bump–less switching back to automatic mode.
- **DHALT mode (DHALT=1):**
  This mode is used to prevent step changes to the controller output when the setpoint is changed. When the setpoint is changed, the controller I contribution is adjusted so that the controller output does not change on the next controller cycle. The controller will subsequently integrate the output in a ramp–like fashion until the PV reaches the new SP.

3.3 Output Parameters

The controller output is set at Y. The range of the output is defined by the parameters YMIN and YMAX. When these limits are reached, the parameters QMIN and QMAX are set. In the event that an output limit is reached, anti–reset–windup is activated. This prevents the I–contribution from continually integrating, and guarantees that when the controller inputs create a change of direction of the output, the output Y will be immediately released from the upper or lower limit.

Figure 6 Controller structure



Priority controller operating mode:
Operating mode switchover (P,I,PI or GAIN, PROP)...DHALT...MAN...Limitation

HIGH ——————————————————————— LOW

SEQ_VAC
Scaling/Sequence Block for Air Conditioning

1 Brief description

SEQ_VAC is a basic function block that can be used to scale real input variables to real output variables using a linear relationship. The block can be used to scale both process variable inputs as well as controller outputs.

The scaling is performed in the form of cartesian coordinates. The input variable range is specified as X1...X2. The output variable range is specified as Y1...Y2.

The EFB provides the following:

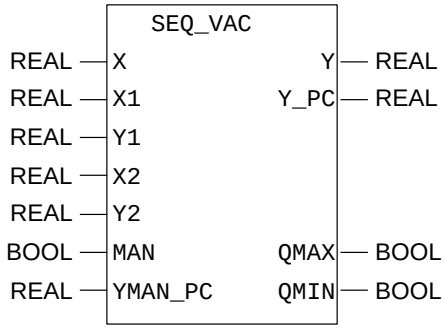
- Full four quadrant operation, allowing the creation of both positive, negative, forward and reverse acting sequences.
- Limitation of the outputs Y1...Y2 when the input range X1...X2 is exceeded. When this happens, the variables QMIN and QMAX are set.
- A manual operation mode to allow the presetting of the output in percentage form.
- Display of the output in both REAL and percentage form.

Additional parameters EN and ENO may be configured.

You will find this EFB in the HVAC library.

2 Representation

2.1 Symbol



2.2 Parameter Specifications

Parameter	Data Type	Meaning
X	REAL	Input variable
X1	REAL	1. Abcissa value }1. Value pair
Y1	REAL	1. Ordinate value }1. Value pair
X2	REAL	1. Abcissa value }2. Value pair
Y2	REAL	1. Ordinate value }2. Value pair
MAN	BOOL	MANUAL mode
YMAN_PC	REAL	Manual control value 0 – 100%
Y	REAL	Output variable (control value)
Y_PC	REAL	Output variable as a %
QMAX	BOOL	Upper limit of signalling device reached
AMIN	BOOL	Lower limit of signalling device reached

3 Detailed description

3.1 Basic Operation

The SEQ_VAC module is processed in each active cycle. Examples of the use of SEQ_VAC can be find in chapter 2 ”General Information”, Figure 2**.**

3.2 Parameter Specifications

The input variable to be scaled is denoted as X. The range over which it is to be scaled is specified by X1 and X2. X1 must be less than X2. The corresponding output range is specified by the parameters Y1 and Y2.

If the input range X1 <= X <= X2 is exceeded, the output variable is clamped to the limits represented by the values Y1 and Y2 such that:

$Y_{(X>X2)}$ = Y2 and
$Y_{(X<X1)}$ = Y1.

If the input range is exceeded and clamping is activated, this is indicated by the setting of the variables QMAX and QMIN.

An increasing sequence is specified by setting Y1 < Y2.

A decreasing (reverse acting) sequence is specified by specifying Y1>Y2.

The percentage value of the scaling result is output at Y_PC. It is important to note that the smaller value of Y1 and Y2 is always equated to 0% and the greater value to 100%. In other words, the percentage value is related to the size of the output variable Y independent of the direction (increasing/decreasing) of the sequence.

This is shown in Figure 1 and Figure 2.

Figure 1 Ascending sequence, 1. Quadrant

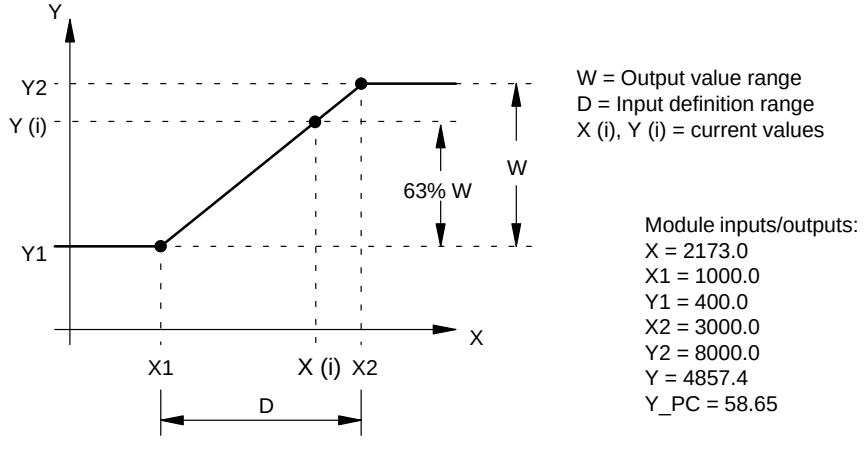
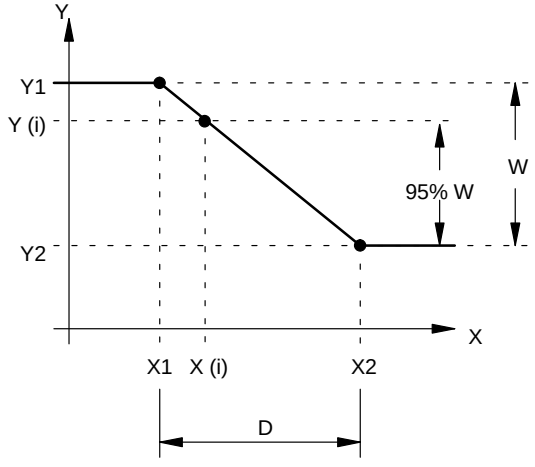


Figure 2 Descending sequence, 1. Quadrant



W = Output value range
D = Input definition range
X (i), Y (i) = current values

Module inputs/outputs:
X = 117.0
X1 = 100.0
Y1 = 7500.0
X2 = 450.0
Y2 = 1250.0
Y = 7196.428
Y_PC = 95.143

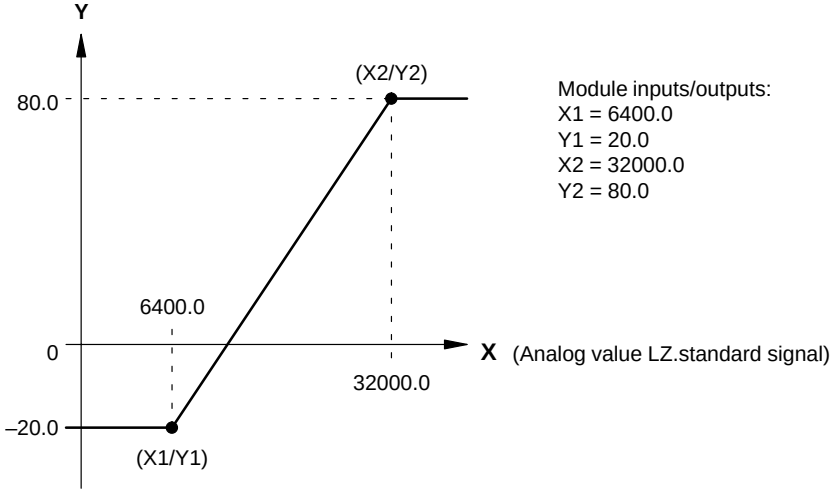
3.3 Manual Operation

The SEQ_VAC block may be put in manual by setting MAN = 1. In manual mode, the percentage value specified by YMAN_PC is written to the block's output, i.e., Y_PC = YMAN_PC. The real value Y is determined by the scaling of the sequence.

Example: An analogue input 4 – 20 ma must be converted to a REAL value corresponding to a temperature range of –20 to +80 degrees Celsius. The corresponding parameters are :

X1 = 6400.0 (accordingly –20 degrees C)
X2 = 32000.0 (accordingly +80 degrees C)
Y1 = –20.0 (accordingly –20 degrees C)
Y2 = 80.0 (accordingly +80 degrees C)

Figure 3 Sequence 4. –1. Quadrant



SW_VAC
Summer / Winter Setpoint Compensation for Air Conditioning

1 Brief description

SW_VAC is a basic function block that provides summer/winter compensation of a temperature setpoint based on the outside air temperature command variable CV.

The EFB provides the following :

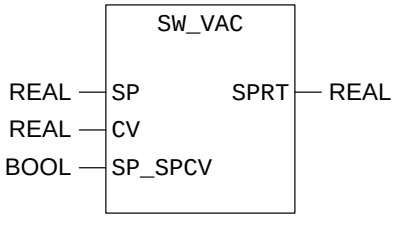
- Summer/winter setpoint compensation with summer compensation following the DIN 1946 Part 2 standard.
- Switchable operating modes offering the choice of compensation or no compensation.

Additional parameters EN and ENO may be configured.

You will find this EFB in the HVAC library.

2 Representation

2.1 Symbol



2.2 Parameter Specifications

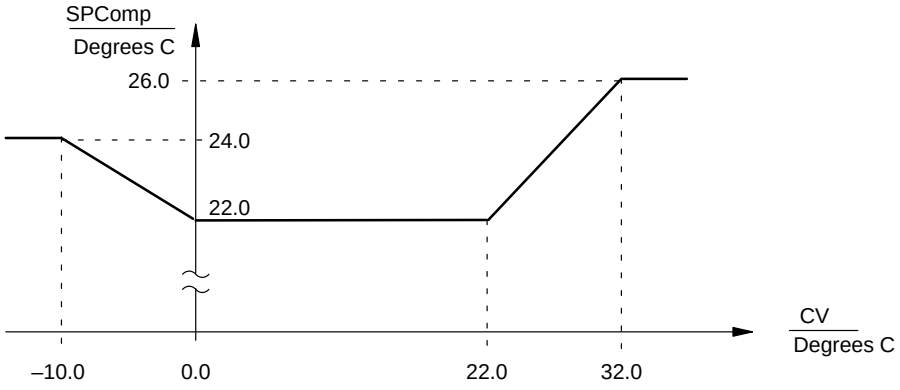
Parameter	Data Type	Meaning
SP	REAL	Setpoint
CV	REAL	Command variable, (as a rule the outside air temperature)
SP_SPCV	BOOL	Setpoint / setpoint + command variable
SPRT	REAL	Setpoint room temperature

3 Detailed description

3.1 Basic Operation

The SW_VAC function block is processed in each cycle. The function block is used to adjust the room or inlet air temperature setpoint SPRT based on the outside air temperature command variable CV. The compensation curves are shown in Figure 1.

Figure 1 Summer / Winter compensation curve

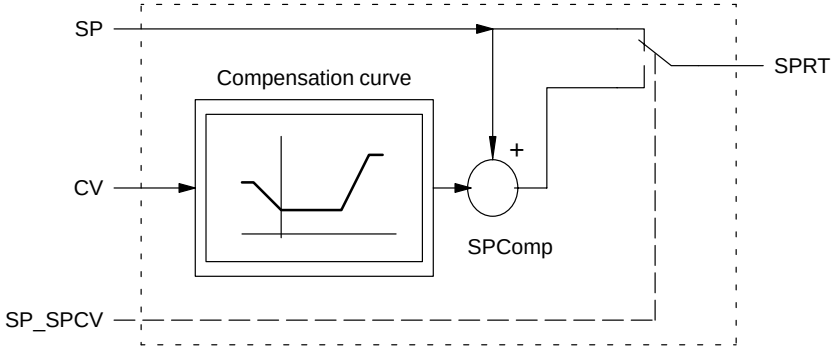


The SP_SPCV input can be used to switch the block's mode of operation.

In "Setpoint" mode (SP_SPCV = 0), no compensation is performed and SPRT = SP.

In "Setpoint with command variable" mode (SP_SPCV = 1), compensation is performed and SPRT = SP + SPComp as shown in Figure 2).

Figure 2 Module structure



Under normal circumstances, when operating in compensation mode, SP is set to zero and SPRT = SPComp. However, the user may choose to input a value for SP. This will

have the effect of moving the summer/winter compensation curve shown in Figure 1 vertically up or down.

3.2 Parameter Specifications

The command variable is specified in degrees Celsius. The setpoint SP is a real value and should be scaled as appropriate. The output of the block SPRT is scaled as a real value.

THRS_VAC
Threshold Switch with Hysteresis for Air Conditioning

1 Brief description

THRS_VAC is a basic function block that is used to detect a threshold on a REAL variable with a built in hysteresis.

The EFB provides the following :

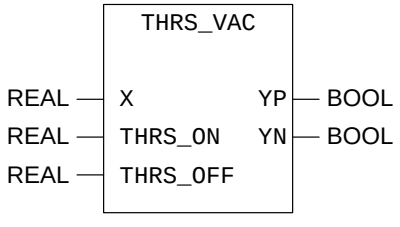
- The setting of an on and off threshold on a real input variable.
- The possibility to set the control direction by selecting any threshold values.
- Positive and negative output signals

Additional parameters EN and ENO may be configured.

You will find this EFB in the HVAC library.

2 Representation

2.1 Symbol



2.2 Parameter Specifications

Parameter	Data Type	Meaning
X	REAL	Input variable
THRS_ON	REAL	‘Energize’ limit value
THRS_OFF	REAL	‘Switch off’ limit value
YP	BOOL	Positive reply signal
YN	BOOL	Negative reply signal

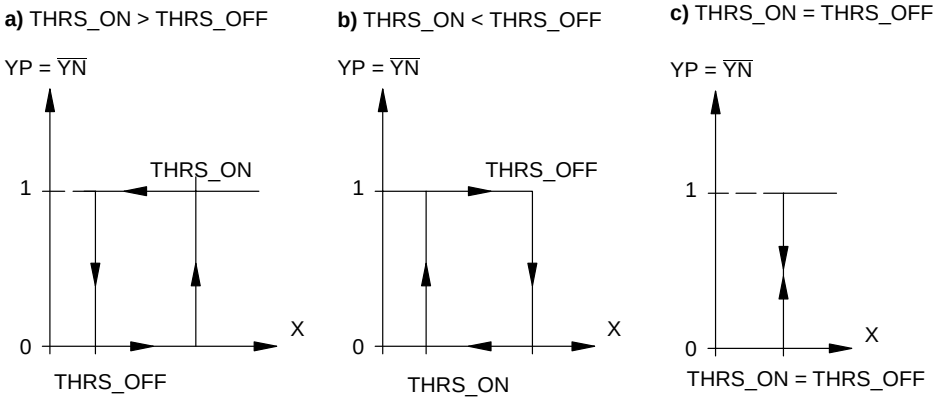
3 Detailed description

3.1 Basic Operation

The block monitors an input variable X for 2 limits/thresholds. When the on–threshold THRS_ON is reached, the output YP is set equal to one, and when the off–threshold THRS_OFF is reached, the output YP is set to zero. The output YN is set as the complement of YP.

The user is free to set any values for the thresholds. The behavior of the outputs based on the relative sizes of the 2 thresholds is shown in Figure 1.

Figure 1 Hysteresis and switching function of THRS_VAC with different configurations



In the case where THRS_ON is not equal to THRS_OFF, when the variable X lies between the 2 thresholds, the output YP remains in its current state until the opposite threshold is reached. In this way, a hysteresis function is provided (see Figure 1). When THRS_ON = THRS_OFF, the block acts as a comparator switch.

The THRS_VAC function block is processed in each cycle. During the first cycle, YP = 0 and YN = 1.

The THRS_VAC block may be used as a switch in many applications such as summer/winter compensation, day/night temperature drops, anti–freeze device, etc.

UC2_VAC
Universal PI controller for air conditioning with 2 outputs

1 Brief description

The UC2_VAC complex function block is a general PI–controller designed for air conditioning applications. It can be used for temperature control, humidity control, air mixing control or other general functions. It consists of a PI controller, summer/winter setpoint compensation and 2 output sequences.

The EFB provides the following :

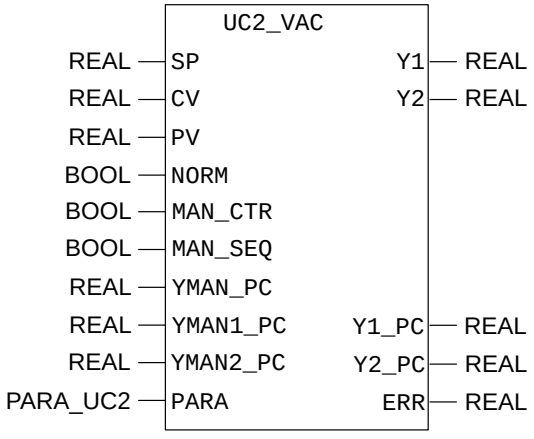
- Winter/summer setpoint compensation as per DIN 1946 part 2.
- Full four quadrant operation of the output sequence scaling
- The display of output variables as percentages
- Upper and lower limits on outputs
- Presetting the controllers' gains in the form of GAIN or PROP with the possibility of using negative values for switching the control direction.
- Operation with Anti–Windup–Reset (AWR)
- Manual adjustment of either the PI–controller total output or the individual sequence outputs (Y1 and Y2) using percentages. When the controller output is manually adjusted, the EFB tracks the I contribution in order to provide bump–less switching back to automatic mode.
- The display of the P and PI controller errors (SP–PV).

Note
Additional parameters EN and ENO should **not** be configured.

You will find this EFB in the HVAC library.

2 Representation

2.1 Symbol



2.2 Parameter Specifications

Table 1 UC2_VAC

Parameter	Data Type	Meaning
SP	REAL	Setpoint
CV	REAL	Command variable
PV	REAL	Actual value
NORM	BOOL	Basic setting
MAN_CTR	BOOL	MANUAL for controller
MAN_SEQ	BOOL	MANUAL for sequences
YMAN_PC	REAL	Total manual manipulated variable as a % for controller
YMAN1_PC	REAL	Individual manual manipulated variable as a % for 1 Sequence
YMAN2_PC	REAL	Individual manual manipulated variable as a % for 2. Sequence
PARA	PARA_UC2	Parameter structure
Y1	REAL	Manipulated variable / Output variable 1
Y2	REAL	Manipulated variable / Output variable 2
Y1_PC	REAL	Manipulated variable / Output variable 1 as a %
Y2_PC	REAL	Manipulated variable / Output variable 2 as a %
ERR	REAL	System Deviation

Table 2 PARA_UC2

Element	Data Type	Meaning
SP_SPCV	BOOL	Setpoint / setpoint + command variable
PI Controller		
YMAX	REAL	Upper limit manipulated variable
YMIN	REAL	Lower limit manipulated variable
GAIN	REAL	Controller gain
PROP	REAL	Proportional value
PREF	REAL	Proportional value reference
TI	TIME	Reset time
1. Sequence		
X1_1	REAL	1. Abcissa value
Y1_1	REAL	1. Ordinate value
X2_1	REAL	2. Abcissa value
Y2_1	REAL	2. Ordinate value
2. Sequence		
X1_2	REAL	1. Abcissa value
Y1_2	REAL	1. Ordinate value
X2_2	REAL	2. Abcissa value
Y2_2	REAL	2. Ordinate value

3 Detailed description

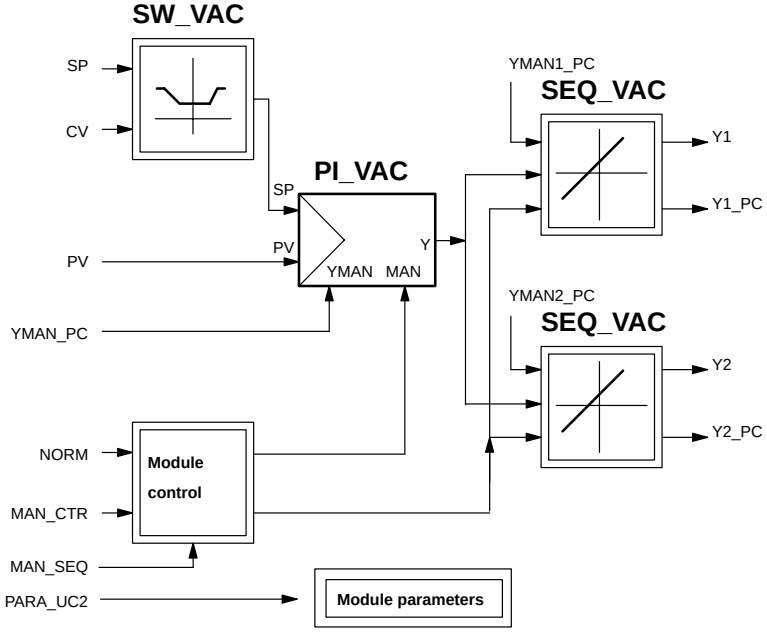
3.1 EFB Structure

A block diagram representation of the UC2_VAC complex function block is shown in Figure 1. It is made up of the following basic function blocks:

- SW_VAC Summer/winter compensation (see page 54)
- PI_VAC Basic PI controller for HVAC applications (see page 45)
- SEQ_VAC Output Sequence/scaling module (see page 49)

Please refer to the respective individual basic function block description for detailed information.

Figure 1 Controller structure



3.2 Basic Operation

The setpoint for the UC2_VAC function block is fed to the PI–controller via the summer/winter setpoint compensation block SW_VAC (description see page 54). The output of the SW–VAC block is fed to the setpoint input of the PI controller

The PI–controller output is fed to 2 sequence output blocks SEQ_VAC. The PI–controller output can be maintained within minimum and maximum limits by using the PI–controller output limits YMAX1 and YMIN1.

3.3 Manual Operation

There are 3 possible modes of manual operation which are specified by setting the mutually exclusive NORM, MAN_CTR and MAN_SEQ parameters.

- **NORM mode (NORM = 1):**
  The output of the PI–controller output is set to zero. The zero value is fed to the output sequences that are active in NORM mode. As a result the actual values of Y1 and Y2 that are output to the control actuators will depend on the output sequence parameters. In the case where the controller output limits YMIN2 and YMAX2 have been set to values that do not enclose zero, the output will be set to the value of YMIN2 and YMAX2 that is closest to zero. In other words:

  Y = YMAX2 if YMAX2 < 0 AND YMIN2 < 0,
  Y = YMIN2 if YMAX2 > 0 AND YMIN2 > 0.

  Again, the actual values of Y1 and Y2 that are output to the control actuators will depend on the output sequence parameters.
- **MAN_CTR mode (MAN_CTR=1):**
  The output of the PI controller is set equal to the user–specified parameter YMAN_PC. Again, the output sequences are active in this mode and therefore the actual values of Y1 and Y2 that are output to the control actuators will depend on the output sequence parameters.
- **MAN_SEQ mode (MAN_SEQ=1):**
  The outputs Y1 and Y2 are set to the values specified by the parameters YMAN1_PC and YMAN2_PC:
  i.e., Y1_PC = YMAN1_PC and Y2_PC = YMAN2_PC.

In MAN_CTR mode, the I contribution of the PI–controller is tracked so that a bumpless transfer back to automatic mode may be carried out. In NORM and MAN–SEQ modes, the I contribution is set to zero.

3.4 Output Parameters

The UC2_VAC outputs Y1 and Y2 are available in real or percentage form (Y1, Y2, Y1_PC and Y2_P2).

The percentage values specified for the manual control of outputs (YMAN_PC, YMAN1_PC and YMAN2_PC) refer to the specified range of the appropriate output. For example, the variable YMAN_PC sets the total output of the PI_VAC basic function block. The range of this output is specified by the parameters YMIN (0%) and YMAX (100%).

The variables YMAN1_PC and YMAN2_PC set the outputs of the two SEQ_VAC blocks, the ranges of which are specified by their corresponding ordinate vales Y1..Y2 (where Y1 and Y2 refer to the SEQ_VAC ordinates, not the actual outputs Y1 and Y2 of the two SEQ_VAC blocks). It is important to note that the smaller value of Y1 and Y2 is always equated to 0% and the greater value to 100%. In other words, the percentage value is related to the size of the output variable Y irrespective of its direction:

Y1 < Y2 –> 0%..100% = Y1..Y2

Y1 > Y2 –> 0%..100% = Y2..Y1

For more information on the operation of the SEQ_VAC block, please refer to the respective description (see page 49).

UC3_VAC
Universal PI controller for air conditioning with 3 outputs

1 Brief description

The UC3_VAC complex function block is a general PI–controller designed for air conditioning applications. It can be used for temperature control, humidity control, air mixing control or other general functions. It consists of a PI controller, summer/winter setpoint compensation and 3 output sequences.

The EFB provides the following :

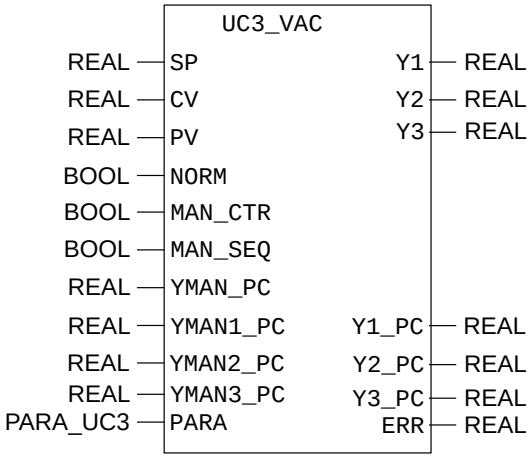
- Winter/summer setpoint compensation as per DIN 1946 part 2.
- Full four quadrant operation of the output sequence scaling
- The display of output variables as percentages
- Upper and lower limits on outputs
- Presetting the controllers' gains in the form of GAIN or PROP with the possibility of using negative values for switching the control direction.
- Operation with Anti–Windup–Reset (AWR)
- Manual adjustment of either the PI–controller total output or the individual sequence outputs (Y1 and Y2) using percentages. When the controller output is manually adjusted, the EFB tracks the I contribution in order to provide bump–less switching back to automatic mode.
- The display of the P and PI controller errors (SP–PV).

Note
Additional parameters EN and ENO should **not** be configured.

You will find this EFB in the HVAC library.

2 Representation

2.1 Symbol



2.2 Parameter Specifications

Table 1 UC3_VAC

Parameter	Data Type	Meaning
SP	REAL	Setpoint
CV	REAL	Command variable
PV	REAL	Actual value
NORM	BOOL	Basic setting
MAN_CTR	BOOL	MANUAL for controller
MAN_SEQ	BOOL	MANUAL for sequences
YMAN_PC	REAL	Total manual manipulated variable as a % for controller
YMAN1_PC	REAL	Individual manual manipulated variable as a % for 1 Sequence
YMAN2_PC	REAL	Individual manual manipulated variable as a % for 2 Sequence
YMAN3_PC	REAL	Individual manual manipulated variable as a % for 3 Sequence
PARA	PARA_UC3	Parameter structure
Y1	REAL	Manipulated variable / Output variable 1
Y2	REAL	Manipulated variable / Output variable 2
Y3	REAL	Manipulated variable / Output variable 3
Y1_PC	REAL	Manipulated variable / Output variable 1 as a %
Y2_PC	REAL	Manipulated variable / Output variable 2 as a %
Y3_PC	REAL	Manipulated variable / Output variable 3 as a %
ERR	REAL	System Deviation

Table 2 PARA_UC3

Element	Data Type	Meaning
SP_SPCV	BOOL	Setpoint / setpoint + command variable
PI Controller		
YMAX	REAL	Upper limit manipulated variable
YMIN	REAL	Lower limit manipulated variable
GAIN	REAL	Controller gain
PROP	REAL	Proportional value
PREF	REAL	Proportional value reference
TI	TIME	Reset time
1. Sequence		
X1_1	REAL	1. Abcissa value
Y1_1	REAL	1. Ordinate value
X2_1	REAL	2. Abcissa value
Y2_1	REAL	2. Ordinate value
2. Sequence		
X1_2	REAL	1. Abcissa value
Y1_2	REAL	1. Ordinate value
X2_2	REAL	2. Abcissa value
Y2_2	REAL	2. Ordinate value
3 Sequence		
X1_3	REAL	1. Abcissa value
Y1_3	REAL	1. Ordinate value
X2_3	REAL	2. Abcissa value
Y2_3	REAL	2. Ordinate value

3 Detailed description

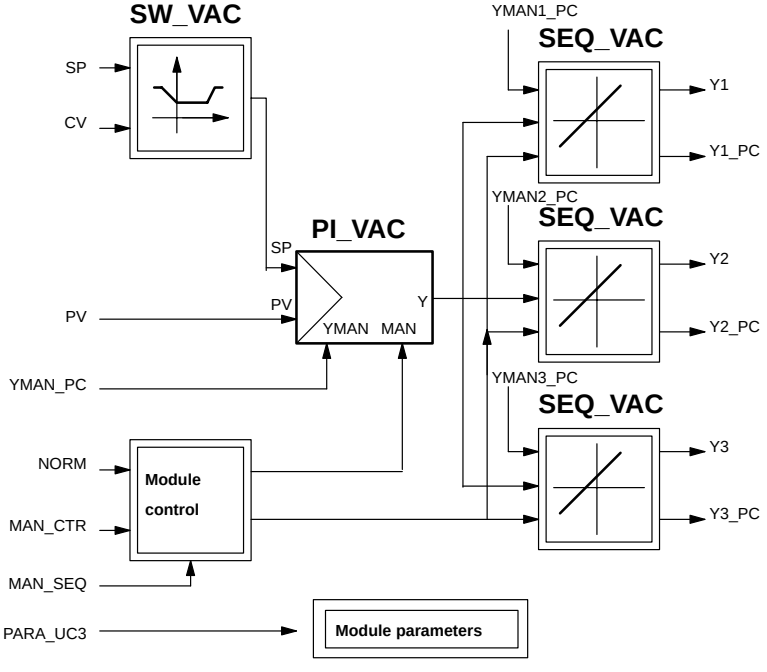
3.1 EFB Structure

A block diagram representation of the UC3_VAC complex function block is shown in Figure 1. It is made up of the following basic function blocks:

- SW_VAC Summer/winter compensation (see page 54)
- PI_VAC Basic PI controller for HVAC applications (see page 45)
- SEQ_VAC Output Sequence/scaling module (see page 49)

Please refer to the respective individual basic function block description for detailed information.

Figure 1 Controller structure



3.2 Basic Operation

The setpoint for the UC3_VAC function block is fed to the PI–controller via the summer/winter setpoint compensation block SW_VAC (description see page 54). The output of the SW–VAC block is fed to the setpoint input of the PI controller.

The PI–controller output is fed to 3 sequence output blocks SEQ_VAC. The PI–controller output can be maintained within minimum and maximum limits by using the PI–controller output limits YMAX1 and YMIN1.

3.3 Manual operation

There are 3 possible modes of manual operation which are specified by setting the mutually exclusive NORM, MAN_CTR and MAN_SEQ parameters.

- **NORM mode (NORM = 1):**
  The output of the PI–controller output is set to zero. The zero value is fed to the output sequences that are active in NORM mode. As a result the actual values of Y1, Y2 and Y3 that are output to the control actuators will depend on the output sequence parameters. In the case where the controller output limits YMIN2 and YMAX2 have been set to values that do not enclose zero, the output will be set to the value of YMIN2 and YMAX2 that is closest to zero. In other words:

  Y = YMAX2 if YMAX2 < 0 AND YMIN2 < 0,
  Y = YMIN2 if YMAX2 > 0 AND YMIN2 > 0.

  Again, the actual values of Y1, Y2 and Y3 that are output to the control actuators will depend on the output sequence parameters.
- **MAN_CTR mode (MAN_CTR=1):**
  The output of the PI controller is set equal to the user–specified parameter YMAN_PC. Again, the output sequences are active in this mode and therefore the actual values of Y1, Y2 and Y3 that are output to the control actuators will depend on the output sequence parameters.
- **MAN_SEQ mode (MAN_SEQ=1):**
  The outputs Y1, Y2 and Y3 are set to the values specified by the parameters YMAN1_PC, YMAN2_PC and YMAN3_PC,
  i.e., Y1_PC = YMAN1_PC, Y2_PC = YMAN2_PC and Y3_PC = YMAN3_PC.

In MAN_CTR mode, the I contribution of the PI–controller is tracked so that a bumpless transfer back to automatic mode may be carried out. In NORM and MAN–SEQ modes, the I contribution is set to zero.

3.4 Output Parameters

The UC3_VAC outputs Y1, Y2 and Y3 are available in real or percentage form (Y1, Y2, Y3, Y1_PC, Y2_PC and Y3_PC).

The percentage values specified for the manual control of outputs (YMAN_PC, YMAN1_PC, YMAN2_PC and YMAN3_PC) refer to the specified range of the appropriate output. For example, the variable YMAN_PC sets the total output of the PI_VAC basic function block. The range of this output is specified by the parameters YMIN (0%) and YMAX (100%).

The variables YMAN1_PC, YMAN2_PC and YMAN3_PC set the outputs of the two SEQ_VAC blocks, the ranges of which are specified by their corresponding ordinate vales Y1..Y2 (where Y1 and Y2 refer to the SEQ_VAC ordinates, not the actual outputs Y1, Y2 and Y3 of the two SEQ_VAC blocks). It is important to note that the smaller value of Y1 and Y2 is always equated to 0% and the greater value to 100%. In other words, the percentage value is related to the size of the output variable Y irrespective of its direction:

Y1 < Y2 –> 0%..100% = Y1..Y2

Y1 > Y2 –> 0%..100% = Y2..Y1

For more information on the operation of the SEQ_VAC block, please refer to the respective description (see page 49).

VQ_VAC
Measured Value Deadband Block for Air Conditioning

1 Brief description

The VQ_VAC basic function block performs the function of adding a deadband DX to an input variable X. If X changes by more than the amount DX, the output Y is updated with the value of X.

The function block may be used to:

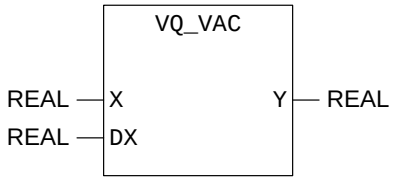
- Stabilize a process variable that has a certain amount of ”noise”, e.g., room pressure measurement.
- Set a preset minimum modification that must be made to a value before it will be changed.
- Provide a dynamic matching of the range of a minimum modification around a current input variable.

Additional parameters EN and ENO may be configured.

You will find this EFB in the HVAC library.

2 Representation

2.1 Symbol



2.2 Parameter Specifications

Parameter	Data Type	Meaning
X	REAL	Input variable
DX	REAL	Minimum modification input
Y	REAL	Output variable

3 Detailed description

3.1 Basic Operation

The VQ_VAC basic function block updates an output Y, with an input value X when X changes by an amount greater than the value set at the input DX.

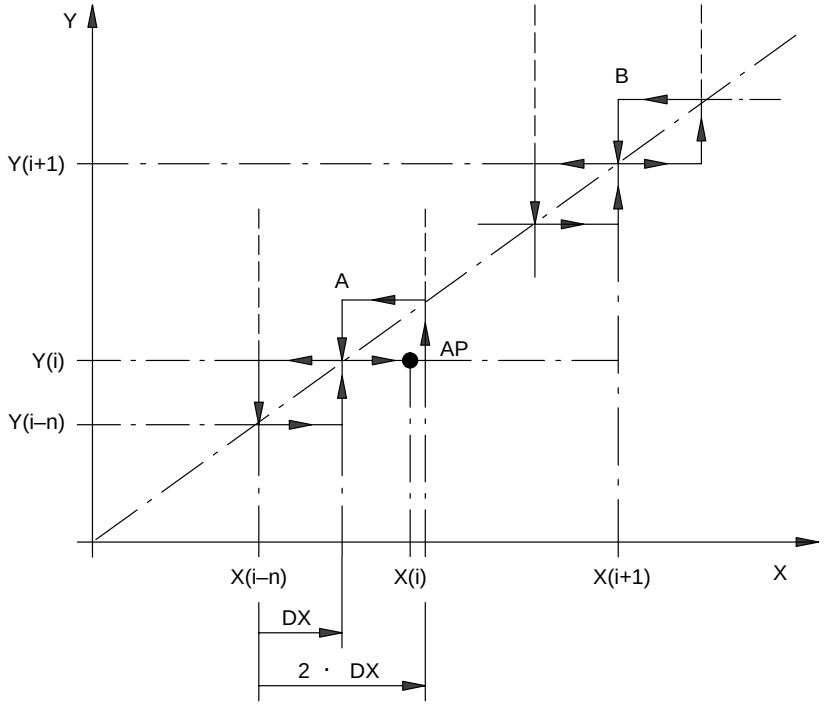
During the first cycle, the output Y is set to equal the value of input X. Y will then retain this value until :

Xnew <= X – DX or Xnew >= X + DX

At which point, Y = Xnew

This is shown in Figure 1.

Figure 1 Quantization diagram VQ_VAC



A: Quantization diagram in the case of ‘slow’ movement of the working point AP from an input value X(i–n) lying behind n steps of the current input value X(i)
B: New quantization diagram in the case of stepped modification from X(i) to X(i+1), if the jump is >= DX.

The function block can be used to either stabilize a varying input or to set a dynamic deadband range around a value. The range is dynamic in the sense that it is not

absolute, but is relative to the input value X. One example of where it could be used is to set a trigger point relative to a controller setpoint which may itself be changed by an operator.

Where the block is used to manipulate a process variable to a controller, care must taken by performing on site tests that the stability of the control loop is not adversely affected by the introduced deadband.

3.2 Parameters

The input X and output Y are real variables. The variable DX must be set as a positive real value. The scaling basic function block SEQ_VAC may be used before or after VQ_VAC in order to scale the measured value X/Y.

WASH_VAC
Basic Washer Block for Air Conditioning

1 Brief description

The WASH_VAC basic function block provides the Calculation of a dew point value SP_TW based upon a dry bulb temperature SP_T and relative humidity value SP_RH.

One possible application for the WASH_VAC block is the possibility to control a room's humidity by regulating the outlet temperature of a washer.

☞ **Note**
It can only be used with air washers that guarantee a 100% saturated air at the washer outlet.

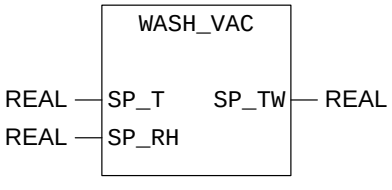
In other words, the block assumes that the outlet washer dry bulb temperature is equivalent to the dew point temperature. Such an approach has the advantage of controlling a humidity using a cost–effective temperature sensor rather than a more expensive humidity sensor.

Additional parameters EN and ENO may be configured.

You will find this EFB in the HVAC library.

2 Representation

2.1 Symbol



2.2 Parameter Specifications

Parameter	Data Type	Meaning
SP_T	REAL	Setpoint temperature at target location (room temperature)
SP_RH	REAL	nominal value rel. humidity at target location (room humidity)
SP_TW	REAL	Setpoint washer outlet temperature

3 Detailed description

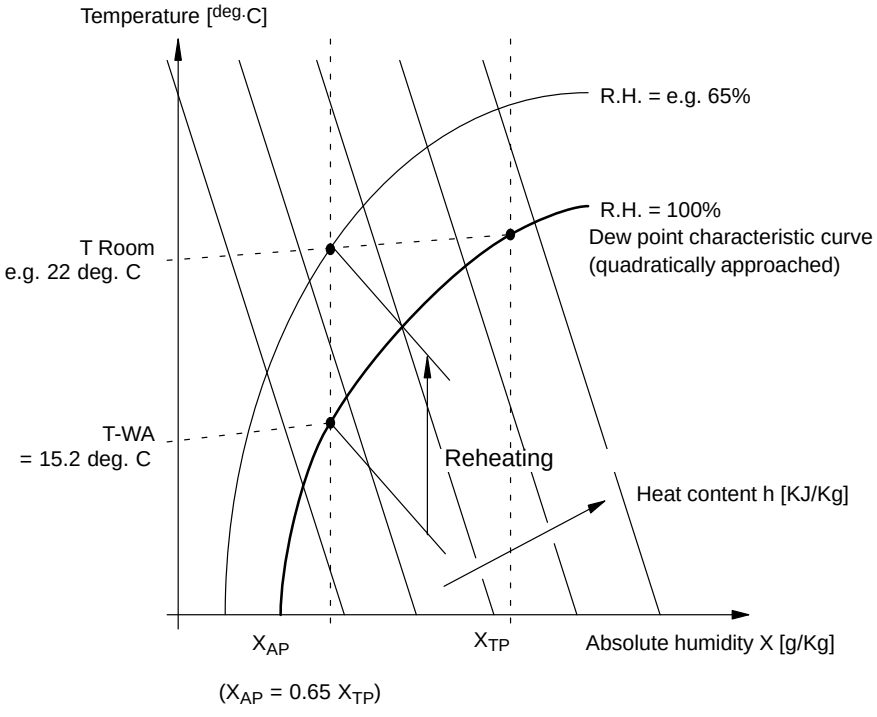
3.1 Basic Operation

The principle of operation of the WASH_VAC basic function block relies on the fact that the dewpoint temperature of air is equal to the dry bulb temperature when the air is fully saturated, i.e., its relative humidity is equal to 100%. The purpose of the WASH_VAC basic function block is to calculate the dewpoint temperature setpoint SP_TW, based on a specified room temperature setpoint SP_T and relative humidity setpoint SP_RH. For a washer application, by controlling the washer outlet temperature to SP_TW, one guarantees that when the air is reheated to the room temperature setpoint SP_T, the room relative humidity value will equal SP_RH. This can be done because reheating the air does not change the air's absolute humidity, but only its dry bulb temperature and therefore its relative humidity.

It therefore follows that if one can guarantee that the washer outlet air has a relative humidity of 100%, the dewpoint temperature can be measured and controlled using a simple dry bulb temperature sensor rather than a more expensive humidity sensor.

An example is shown on the figure "H/X Diagram"Figure 1. For a room temperature of 22 degrees Celsius and a relative humidity of 65%, the WASH_VAC function block will calculate the dewpoint setpoint as 15.2 degrees Celsius. By controlling the washer outlet temperature to 15.2 degrees Celsius and subsequently reheating the air to 22 degrees Celsius, the room relative humidity will equal 65%

Figure 1 h/x diagram with clarification of the method of operation of the WASH_VAC



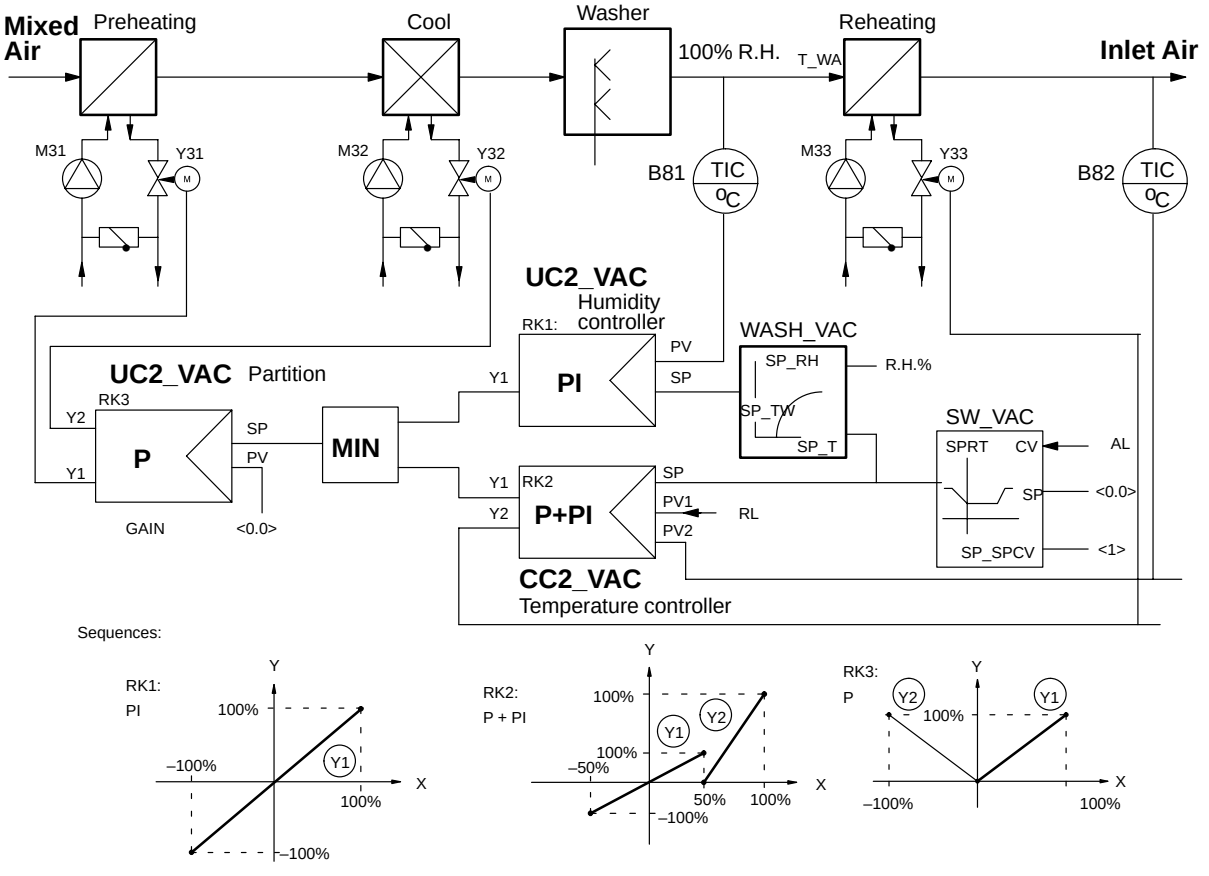
3.2 **Parameters**

The air temperature dry bulb temperature setpoint is specified by the variable SP_T in degrees Celsius. The relative humidity is specified by the variable SP_RH as a % value.

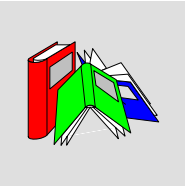
The washer outlet temperature setpoint is output in degrees Celsius at the location specified at SP_TW.

Example: An example application consisting of a preheat coil, cooling coil, washer and reheat coil is shown in the Figure 2. The setpoint calculated by WASH_VAC is fed to a lower level PI controller whose PV is connected to the washer temperature transmitter and whose output is used to control the preheat and cooling coils. In other words, this PI controller maintains the appropriate dewpoint temperature off the washer. In addition, a temperature controller CC2_VAC is used to control the room and inlet air temperature by driving the preheat/reheat coils and cooling coil. A minimum select is used to switch between the humidity and temperature controller. The controller RK3 is configured as a P controller only with a PV set to zero and the gain set to one. In this way, it simply acts as an output sequencer that divides the setpoint input between the preheat and cooling coils. The washer outlet temperature is set to a range of 5 to 30 degrees Celsius to prevent icing up or overheating of the washer.

Figure 2 Integration of WASH_VAC in a room / inlet air_cascade control system



Glossary



Here you will find a short description of the terms.

984LL
Refer to Ladder Logic 984

A

Active window
The window selected at present. Only one window can be active at any given time. When a window becomes active, its title bar changes color to differentiate it from the other windows. Non–selected windows are inactive.

Actual parameter
Currently connected input/output parameter.

Addresses
(Direct) addresses are memory areas in the PLC. They are located in State RAM and can be assigned to input/output modules.

ANL_IN
ANL_IN represents the data type ”analog input”. It is used for analog value processing. The data type is automatically assigned the 3x references specified in the I/O map of the configured analog input module. Therefore, only unlocated variables can be assigned.

ANL_OUT
ANL_OUT represents the data type ”analog output”. It is used for analog value processing. The data type is automatically assigned the 4x references specified in the I/O map of the configured analog output module. Therefore, only unlocated variables can be assigned.

ANY
In this version, ”ANY” includes the data types ANL_IN, ANL_OUT, BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME, and WORD, as well as data types derived from those.

ANY_BIT
In this version, ”ANY_BIT” includes the data types BOOL, BYTE, and WORD.

ANY_ELEM
In this version, ”ANY_ELEM” includes the data types BOOL, BYTE, DINT, INT, REAL, UDINT, UINT, TIME, and WORD.

ANY_INT
In this version, ”ANY_INT” includes the data types DINT, INT, UDINT, and UINT.

ANY_NUM
In this version, ”ANY_NUM” includes the data types DINT, INT, REAL, UDINT, and UINT.

ANY_REAL
In this version, ”ANY_REAL” includes the data type REAL.

Application Window
The window containing the workspace, the menu bar, and the tool bar for the application program. The name of the application program appears in the title bar. One application window may contain several document windows.

In Concept, the application window corresponds to a project.

Argument
Synonymous with actual parameter.

Array variables
Variables that are assigned a defined derived data type using the keyword ARRAY (field).

An field is a collection of data elements of the same data type.

ASCII mode
American Standard Code for Information Interchange.
The ASCII mode is used for communication with different host devices. ASCII works with 7 data bits.

Atrium
The PC based controller which is based on a 386 EX microprocessor, is mounted on a standard AT–Platine and can be used inside a Host–Computers on an ISA bus slot. The module has a motherboard (SA85 driver needed) with 2 sockets for PC104 daughter boards. One of this PC104–Daughter–Board is used as CPU and the other is used for Interbus S controlling.

B

Backup file (Concept EFB)
The backup file is a copy of the last source code file. The name of this backup file is ”backup??.c” (assuming that there are never more than 100 copies of your source code file). The name of the first backup file is ”backup00.c”.

If the definition file has been modified without causing any interface change in the EFB, it is not necessary to create a backup file by editing your source code file (`Objects` → `Source`).

If a backup file is generated, you can name it Source file.

Base 2 literals
Base 2 literals are used to specify integer values in the binary system. The base is identified by the prefix 2#. The values cannot have a sign (+/–). Individual underscore symbols (_) between the numbers have no significance.

Example
2#1111_1111 or 2#11111111 (255 decimal)
2#1110_0000 or 2#11100000 (224 decimal)

Base 8 literals
Base 8 literals are used to specify integer values in the octal system. The base is identified using the prefix 8#. The values cannot have a sign (+/–). Individual underscore symbols (_) between the numbers have no significance.

Example
8#3_77 or 8#377 (decimal 255)
8#34_0 or 8#340 (decimal 224)

Base 16 literals
Base 16 literals are used to specify integer values in the hexadecimal system. The base is identified using the prefix 16#. The values cannot have a sign (+/–). Individual underscore symbols (_) between the numbers have no significance.

Example
16#F_F or 16#FF (decimal 255)
16#E_0 or 16#E0 (decimal 224)

Binary links
Links between outputs and inputs of FFBs in data type BOOL.

Bit string
A data element consisting of one or more bits.

BOOL
BOOL represents the data type ”boolean”. The length of the data elements is 1 bit (stored in memory in 1 byte). The value range for variables of this data type is 0 (FALSE) and 1 (TRUE).

Bridge
A bridge is a device that connects networks. It enables communication between nodes on the two networks. Each network has its own token rotation sequence – the token is not passed along through bridges.

BYTE
BYTE represents the data type ”bit string 8”. It is entered as base 2 literal, base 8 literal or base 16 literal. The length of the data elements is 8 bits. This data type cannot be assigned a numeric value range.

C

Coil
A coil is an LDelement that transfers the state of the horizontal link at its left side without any change to the horizontal link at its right side. Through this, the state is stored in the attached variable/direct address.

Compact format (4:1)
The first digit of the reference is separated from the address that follows by a colon (:), and no leading zeros are entered in the address.

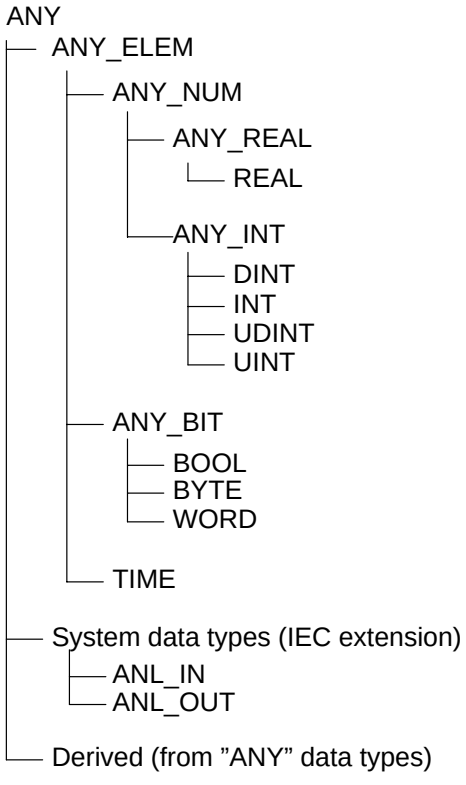
Constants
Constants are unlocated variables that are assigned a value that cannot be changed by the program logic (read only).

Contact
A contact is an LDelement that transfers a state to the horizontal link at its right side. This state is a result of the boolean AND–link of the state of the horizontal link at its left side with the state of the attached variable/direct address. A contact does not change the value of the attached variable/direct address.

D

Data transfer settings
Settings that determine how information is transferred from your programming unit to a PLC.

Data types



The overview shows the hierarchy of generic data types as used with inputs and outputs of functions and function blocks. Generic data types are identified by the prefix ”ANY”.

DCP drop
A Distributed Control Processor (D908) can be used to set a distributed network with a superior PLC. When using a D908 with distributed PLC, the primary PLC regards the distributed PLC as a head setup station. The D908 and the distributed PLC are communicating via the system bus which results in high performance with minimal effect on scan time. Data exchange between the D908 and the primary PLC is carried out via the distributed I/O bus at 1.5 mega bits per second. A primary PLC can support up to 32 D908 processors.

DDE (Dynamic Data Exchange)
The DDE interface is used for a dynamic data exchange between two programs that are using Windows.

With the DDE link between Concept and Concept Graphic Tool plant signals can be displayed as a Timing Diagram.

DDT
see Derived Data Type

Declaration
Mechanism for specifying the definition of a language element. A declaration normally involves attaching an identifier to a language element and allocating attributes such as data types and algorithms.

Definition file (Concept EFB)
The definition file contains general descriptive information concerning the EFB and its formal parameters.

Derived data type
Derived data types are data types that have been derived from Elementary data types and/or other Derived data types. Derived data types are defined in the Data Type editor of Concept.

There is a differentiation between Global data types and Local data types.

Derived Function Block (DFB)
A derived function block represents the invocation of a derived function block type. Details about the graphical form of the invocation can be found in the definition ”Function Block (instance)”. Contrary to invocations of EFB types, invocations of DFB types are identified by double vertical lines to the left and right side of the rectangular block symbol.

The body of a derived function block type is designed in FBD language; however, only in the current version of the programming system. At this time, other IEC languages cannot be utilized for the definition of DFB types, nor can derived functions be defined in the current version.

There is a distinction between Local and Global DFBs.

DFB
see Derived function block

DINT
DINT represents the data type ”double integer”. It is entered as an integer literal, base 2 literal, base 8 literal or base 16 literal. The length of data elements is 32 bits. The value range for variables of this data type is from –2 exp (31) to 2 exp (31) –1.

Direct representation
A form of representation for a variable in the PLC program from which the allocation to the logical memory location – and indirectly its physical memory location – can be directly determined.

Distributed network
Distributed programming in the Modbus Plus network facilitates maximum performance during data transfer and in special requirements to links. Programming of a distributed network is easy. Setting up the network does not require additional ladder diagram logic. By making the appropriate entries in the Peer Cop processor, all requirements for data transfer are taken care of.

Dummy
An emtpy file that contains a header with genenral file information, e.g. author, editing date, EFB name etc. The user has to complete this Dummy file by editing.

Duration literals
The units allowed for durations (TIME) are days (D), hours (H), minutes (M), seconds (S), and milliseconds (MS), or combinations thereof. The duration must be identified by the prefix t#, T#, time# or TIME#. The ”overflow” of the most significant unit is allowed; e.g. the entry T#25H15M is allowed.

Example
t#14MS, T#14.7S, time#18M, TIME#19.9H, t#20.4D, T#25H15M, time#5D14H12M18S3.5MS

Document window
A window within an application window. Several document windows can be opened simultaneously in one application window. But only one document window at a time can be active.

Examples of document windows in Concept are sections, the message window, the Reference Data editor, and the PLC configuration.

DX Zoom
This is a feature that allows you to interface with a programming object to observe it’s data values and alter them if necessary.

E

EFB
see Elementary functions/Function blocks

Elementary functions/Function blocks (EFB)
A term for functions or function blocks with type definitions that are not formulated in one of the IEC languages; for instance, the DFB editor (Concept DFB) cannot be used to modify their bodies. EFB types are programmed in ”C” and are made available in compiled form via libraries.

EN / ENO (enable / enable out)
If the value of EN is equal to ”0” when the FFB is invoked, the algorithms defined by the FFB will not be executed and all ouptupts remain with their old values. In this case, the value of ENO is automatically set to ”0”.

If the value of EN is equal to ”1” when the FFB is invoked, the algorithms defined by the FFB will be executed. After these algorithms have executed without an error, the value of ENO is automatically set to ”1”.

Should an error occur while these algorithms are executing, ENO is automatically set to ”0”.

The output behavior of the FFBs is independent of the FFBs being invoked without EN/ENO or with EN=1.

If display of EN/ENO is turned on, the EN–input must be definitely connected. Otherwise, the FFB will never be executed.

The configuration of EN and ENO is turned on or off in the dialog box of the block properties. The dialog box is invoked with the menu commands `Objects` → `Properties...` or by double–clicking at an FFB.

Errors
If an error is detected while an FFBs or a Step is processing (e.g. unauthorized input values or time errors), an error message will appear that can be viewed with the menu command `Online` → `Online events...`. The ENO output for FFBs is set to ”0”.

Evaluation
The process used to establish a value for a function or for the outputs of a function block during program execution.

Expression
Expressions consist of operators and operands.

F

FB
see Function Block (instance)

FBD
see Function Block Diagram

FFB (Functions/Function blocks)
Collective term for EFB (Elementary functions/Function blocks) and DFB (Derived function blocks).

FIR Filter
(**F**inite **I**mpulse **R**esponse Filter)
Finite Impluse Response Filter

Formal parameters
Input/output parameters that are used within the logic of an FFB and are brought out of the FFB as inputs/outputs.

Function (FUNC)
A program organization unit which, when executed, will yield exactly one data element. A function has no internal state information. Multiple calls to the same function with the same input parameter values will always yield the same output values.

Details about the graphical form of function calls can be found in the definition ”Function Block (instance)”. Contrary to function block calls, function calls have only one single unnamed output because its name is the name of the function itself. In FBD, each call is identified by a unique number above the graphical block; this number is automatically created and cannot be changed.

Function block diagram (FBD)
One or several sections containing graphically represented networks consisting of functions, function blocks, and links.

Function block (instance) (FB)
A function block is a program organization unit which provides values for its outputs and internal variable(s) according to the algorithms defined in its function block type description, when executed as a specific instance. All values of the outputs and internal variables of a specific function block instance are maintained from one invocation of the function block to the next. Therefore, multiple calls of the same function block instance with the same arguments (values of input parameters) do not necessarily yield the same output value(s).

Each function block instance is graphically displayed by a rectangular block symbol. The name of the function block type is centered on top, inside the rectangle. The name of the function block instance is also on top, but outside the rectangle. It is automatically

generated when building an instance, but it can be modified by the user, if necessary. Inputs are shown to the left, outputs to the right of the block. The names of the formal input/output parameters are shown inside the rectangle at the corresponding input/output points.

Above description of the graphical presentation applies generally to function calls and to DFB calls as well. Any differences are described in the respective definitions.

Function block type
A language element that consists of: 1. the definition of a data structure subdivided into input, output, and internal variables; 2. a set of operations processed with the elements of the data structure whenever an instance of the function block type is invoked. This set of operations can either be formulated in one of the IEC languages (DFB type) or in ”C” (EFB type).

A function block type can be instantiated multiple times.

Function number
The function number is used to uniquely identify a function in a program or DFB. The function number cannot be edited; it is automatically assigned. The function number always has the structure: .n.m

n = number of section (consecutive number)
m = number of the FFB object in the section (consecutive number)

G

Generic Data Type
A data type representing more than one type of data.

Global Derived data types
Global Derived data types are available in every Concept project; they are deposited in the DFB directory immediately below the Concept directory.

Global DFBs
Global DFBs are available in every Concept project; they are deposited in the DFB directory immediately below the Concept directory.

Global Macros
Global Macros are available in every Concept project; they are deposited in theDFB directory immediately below the Concept directory.

Groups (EFBs)
Some EFB libraries (e.g. the IEC library) are subdivided into groups. This makes it easier to find the EFBs, especially in very large libraries.

H

Hot Standby
A hot standby system consists of two identically configured PLC units that are communicating with each other via hot standby processors.. If there is a failure of the primary PLC, the secondary PLC will assume the control check. Under normal conditions, the secondary PLC assumes no control functions, it only checks status information to detect errors.

I

Icon
Graphical display of various objects in Windows, e.g. drives, application programs, and document windows.

Identifier
refer to IEC naming convention

IEC 1131–3
International Standard: Programmable Controllers – Part 3: Programming Languages, March 1993.

IEC naming conventions (Identifier)
An identifier is a string of letters, numbers, and underscore symbols that must begin with a letter or underscore symbol (for instance, the name of a function block type, an instance, a variable or a section). Letters from National character sets (e.g: ö, ü, é, õ) can be used, except in project and DFB names.

Underscore symbols in identifiers are significant; for example, ”A_BCD” and ”AB_CD” will be interpreted as different identifiers. Several leading and multiple underscore symbols sequentially are not allowed.

Spaces are not allowed in identifiers.

Upper or lower case is not significant; for example, ”ABCD” and ”abcd” will be interpreted as the same identifier.

Key words are not allowed as identifiers.

IIR Filter
(**I**nfinite **I**mpuls **R**esponse Filter)
Infinite Impulse Reponse Filter

IL
refer to Instruction List (IL)

Initial Step (Starting step)
The starting step of a sequential chain. One initial step must be defined in each sequence. This initial step starts the sequence when first invoked.

Initial Value
The value assigned to a variable at the start of a program.

Input bits (1x references)
The 1/0 state of input bits is controlled by process data that are received by the CPU from an input unit.

Note:
The x placed after the first digit of the reference type represents a five–digit memory location in user data memory, e.g. reference 100201 signifies an input bit at address 201 of State RAM.

Input parameter (input)
At the invocation of an FFB it transfers the corresponding argument.

Input registers (3x references)
An input register contains information from an external source which is representing a 16–bit number. A 3x register can also contain 16 sequential input bits that were read into the register in binary or BCD (binary coded decimal) format.

Note:
The x placed after the first digit of the reference type represents a five–digit memory location in user data memory, e.g. reference 300201 signifies a 16 bit input register at address 201 of state RAM.

Instance
see Function Block (instance)

Instance Name
An identifier associated with a specific function block instance.

The name of the instance is used to uniquely identify a function block in a program organization unit. This instance name is automatically created, but it can be edited. The instance name must be unique throughout the program organization unit; there is no distinction between upper and/or lower case. If the name entered already exists, you will be warned and another name must be chosen. The instance name must comply with the IEC naming conventions or an error message will appear. The automatically created instance name will always have the structure: FBI_n_m

FBI = function block instance
n = number of section (consecutive number)
m = number of the FFB object in the section (consecutive number)

Instantiation
The creation of an instance.

Instruction (IL)
Instructions are the ”operations” of the programming language IL. Each instruction begins in a new line and is followed by an operator, sometimes with modifier, and if required for a respective operation, by one or several operands. If several operands are used, they will be separated by commas.

Instruction (LL984)
Programming for electrical controls involves a user who implements Operational Coded instructions in the form of visual objects organized in a recognizable ladder form.. The program objects designed, at the user level, is converted to computer usable OP codes during the download process. The Op codes are decoded in the CPU and acted upon by the controllers firmware functions to implement the desired control.

Instruction list
IL is a text language as per IEC 1131 that displays operations, such as conditional or unconditional calls of function blocks and functions, conditional or unconditional jumps, etc. through instructions.

INT
INT represents the data type ”integer”. It is entered as an integer literal, base 2 literal, base 8 literal or base 16 literal. The length of the data elements is 16 bits. The value range for variables of this data type is from –2 exp (15) to 2 exp (15) –1.

Integer Literals
Integer literals are used to denote integer values in the decimal system. The values can have a preceding sign (+/–). Individual underscore symbols (_) between the numbers have no significance.

Example
–12, 0, 123_456, +986

Invocation
The process that initiates the execution of operations specified by an FFB type.

The instruction can be preceded by a label that is followed by a colon. If there is a comment, it must be the last element in the line.

I/O map
The I/O stations of the different central processing units are configured in the I/O map.

J

Jump
An element of the SFC language. Jumps are used to skip over areas of the sequence.

K

Keywords
Keywords are unique character combinations that are used as specific syntactic elements as defined in Appendix B of IEC 1131–3. All keywords used in the IEC 1131–3 and therefore in Concept are listed in Appendix C of the IEC 1131–3. These listed keywords may not be used for any other purpose, such as for names of variables, sections, instances, etc.

L

Ladder diagram
Refer to
Ladder Diagram (LD),
Ladder Logic 984 (LL)

Ladder Diagram (LD)
Ladder diagram is a graphical programming language as per IEC1131that orientates itself optically at the ”rungs” of a relay ladder diagram.

Ladder Logic 984 (LL)
In the terms Ladder Logic and Ladder diagram the word ladder is a reference to technique. In contrast to a schematic diagram, a ladder diagram is used by electricians to draw a circuit (using electrical symbols) and intended to show the sequence of events not the actual wires that connect the parts. A common user interface to direct the actions of programmable controllers, allows a ladder diagram interface so that electricians do not need to learn an unfamiliar programming language to implement a control program.

The construction of the actual ladder diagram allows the electrical elements to be connected in such a way as to create control output dependent on some logical power flow through the electrical objects used represent the previously required condition of a physical electrical device.

In a simple form the user interface is a video display produced by the PLC programming application that establishes a vertical and horizontal grid into which programming objects are arranged. The diagram has power available at the left side of the grid and when connected to objects that are activated the power will flow from left to right.

Landscape format
Landscape (horizontal) format means that the page width, when looking at the printed text, is larger than its height.

Language element
Every basic element in one of the IEC programming languages, e.g. a step in SFC, a function block instance in FBD or the initial value of a variable.

LD
Refer to
Ladder Diagram (LD),
Ladder Logic 984 (LL)

Library
A collection of software objects intended for reuse when programming new projects, or even to build new libraries. Examples are the library of Elementary function block types.

EFB libraries can be subdivided into groups.

Link
A control or data flow connection between graphical objects (e.g. steps in the SFC editor, function blocks in the FBD editor) within a section, graphically represented as a line.

Literals
Literals are used to provide inputs of FFBs, transition conditions, etc. directly with values. These values cannot be overwritten by the program logic (read only).

LL
Refer to Ladder Logic 984 (LL)

Local Derived data types
Local Derived data types are only available in one single Concept project and its local DFBs, and are deposited in the DFB directory below the project directory.

Local DFBs
Local DFBs are only available in one single Concept project and are deposited in the DFB directory below the project directory.

Local link
The local network link is the network that connects the local node with other nodes either directly or through a repeater.

Local Macros
Local Macros are only available in one single Concept project and are deposited in the DFB directory below the project directory.

Local network node
The local node is the one that is currently being configured.

Located Variable
The variable is assigned an address in the PLC. Located variables are used in the SFC and FBD editors in order to read signal states from the PLC and to turn them over to the PLC. Additionally located variables can be exported and displayed via a DDE interface.

M

Macro
Macros are created by using the software Concept DFB.

Macros are used to duplicate often used sections and networks (including their logic, variables and variable declaration).

Local and global macros are available.

Macros have the following characteristics:

- Macros can only be created in the FBD programming language
- Macros contain only one single section
- Macros may contain any complex section
- With regard to the program, an instantiated macro, i.e. a macro inserted in a section, does not differ from a conventionally created section.
- DFBs can be invoked in a macro.
- Macro–intrinsic variables can be declared for the macro.
- Macro–intrinsic data structures can be used.
- Automatic acceptance of variables declared in the macro.
- Initial values for the macro variables are possible.
- The multiple instantiation of a macro in the total program with different variables is possible
- The section name, the names of variables, and the data structure name can have the ~ character as a swap marking.

MMI
Man–Machine–Interface

Multi–element variables
Variables that are assigned a Derived data type that is defined with STRUCT or ARRAY.

There is a differentiation between array variables and structured variables.

N

Network
A network is the interconnection of units along a shared data path that are communicating with each other via a common protocol.

Network Node
A node is a unit with an address (1...64) on the Modbus Plus network.

Node address
The node address is used to uniquely identify a network node in the routing path. The address is set directly on the node; e.g. with a BCD switch at the rear of the module.

O

Operand
An operand is a literal, a variable, a function call or an expression.

Operator
An operator is a symbol for an arithmetic or boolean operation to be executed.

Output/holding bits (0x references)
An output/holding bit can be used to control real output data through an output unit of the control system, or to define one or more discrete outputs in State RAM.

Note:
The x placed after the first digit of the reference type, represents a five–digit memory location in user data memory, e.g. reference 000201 signifies an output or holding bit at address 201 of State RAM.

Output/holding register (4x references)
An output/holding register can be used to store numerical data (binary or decimal) in State RAM and/or, to send data from the CPU to an output unit in the control system.

Note:
The x placed after the first digit of the reference type represents a five–digit memory location in user data memory, e.g. reference 400201 signifies a 16 bit output/holding register at address 201 of State RAM.

Output parameter (output)
A parameter used to return the result(s) of the evaluation of an FFB.

P

Peer processor
The peer processor manipulates the token passes and the data flow between the Modbus Plus network and the PLC user logic.

PLC
Programmable Logic Controller

Program
The top level program organization unit. A program is downloaded into a single PLC as a whole. A program is further refined by IEC language elements.

Program cycle (scan)
A program cycle consists of reading the inputs, processing of program logic, and writing of the outputs.

Programming unit
Hardware and software that supports programming, configuring, testing, delivering, and troubleshooting in PLC applications as well as in distributed system application. in order provide for source documentation and archiving (backup).The programming unit can also be used for process visualization, if necessary.

Program organization unit
A function, a function block or a program. This term can refer either to a type or to an instance.

Project
General term for the highest level of a software tree structure which defines the overall project name of a PLC application. After the definition of the project name, the system configuration and the control programm can be saved with this name. All data which is created while creating the configuration and the program are belonging to project for this special automation task.

General term for the complete set of programming and configuration information in the project data base which represents the source code that describes the automation of a system.

Project data base
The data base in the programming unit that contains the configuration information for a project.

Portrait format
Portrait (vertical) format means that the height of the page, when looking at the printed text is larger than its width.

Prototype file (Concept EFB)
The prototype file contains all prototypes of the respective functions. A type definition of the internal state structure is also included, if available.

Q

R

REAL
REAL represents the data type ”floating point number”. It is entered as a real literal or as a real literal with exponent. The length of data elements is 32 bits. The value range for variables of this data type is from 8.43E–37 to 3.36E+38.

Real literals
Real literals are used to indicate floating point values in the decimal system. Real literals are identified by a decimal point. The values can have a preceding sign (+/–). Individual underscore symbols (_) between the numbers have no significance.

Example
–12.0, 0.0, +0.456, 3.14159_26

Real literals with exponent
Real literals with exponent are used to indicate floating point values in the decimal system. Real literals with exponent are identified by a decimal point. The exponent indicates the power of ten for the multiplication of the previous number in order to arrive at the value that will be displayed. The values can have a preceding sign (+/–). Individual underscore symbols (_) between the numbers have no significance.

Example
–1.34E–12 or –1.34e–12
1.0E+6 or 1.0e+6
1.234E6 or 1.234e6

Reference
Every direct address is a reference beginning with an identification character that denotes whether this is an input or an output, and whether it is a bit or a register. References that begin with the identification number 6, represent registers in the extended memory of State RAM.
0x range = Output/holding bits
1x range = Input bits
3x range = Input registers
4x range = Output/holding registers
6x range =register in extended memory

Note:
The x placed after the first digit of each reference type represents a five–digit memory location in user data memory, for example, reference 400201 means this is a 16 bit output or holding register at address 201 of State RAM.

Registers in the extended memory (6X–Reference)
6x–References are marker words in the extended memory of the PLC.They can only be used with LL984 user programs with a CPU 213 04 or CPU 424 02.

RIO (Remote I/O)
Remote I/O refers to physical location of I/O point control units with regard to the Processor controlling them. Remote I/O are connected to the control unit through a wired communications cable.

RTU mode
Remote Terminal Unit
The RTU mode is used for communication between the PLC and an IBM–compatible PC. RTU works with 8 data bits.

Runtime Error
Errors that occur while the program is processing on the PLC, in SFC objects (e.g. Steps) or FFBs. These are, for instance, value range overflows in counters or time errors in steps.

S

SA85 module
The SA module is a Modbus Plus adapter for IBM–AT or compatible computers.

Section
A section, for instance, can be used to describe the mode of operation of a technological unit such as a motor.

A program or DFB consists of one or several sections. Sections can be programmed with the IEC programming languages FBD and SFC. Within a section, only one of the listed programming languages can be used.

In Concept, each section has its own document window. However, to have a better overview, it is recommended to subdivide a large section into several smaller ones. The scroll bar is used to move around within a section.

Separator format (4:00001)
A colon (:) separates the first digit of the reference from the following five–digit address.

Sequential Function Chart (SFC)
The SFC language elements allow the subdividing of a PLC program organization unit into a set of steps and transitions which are interconnected through directional links. Each step is associated with a set of actions, and each transition is linked with a transition condition.

Serial ports
Serial ports (COM) transfer information bit by bit.

SFC
see Sequential Function Chart

ST
refer to Structured Text (ST)

Standard format (400001)
The five–digit address is placed immediately after the first digit of the reference.

Statement (ST)
Statements are the ”commands” of the programming language ST. Statements must end with semi–colons. Several statements (separated by semi–colons) may be placed into one line.

State RAM
State RAM (state memory) is the memory location for all variables addressed in the user program through references (direct representation). For example, input bits, output/holding bits, input registers, and output/holding registers are located in State RAM.

Status bits
There is one status bit for each node with global input, specific input or output of peer cop data. If a defined group of data was successfully transferred within the set timeouts, the corresponding status bit will be set to 1. If not, this bit is set to 0, and all data belonging to this group (to 0) will be deleted.

Step
SFC language element: A situation in which the behavior of a program with respect to its inputs and outputs follows a set of rules defined by the associated actions of the step.

Step Name
The step name is used to uniquely identify a step in a program organization unit. The step name is automatically created, but it can be edited. The step name must be unique throughout the program organization unit or an error message will occur. The automatically created step name always has the structure: S_n_m

S = step
n = number of section (consecutive number)
m = number of step in the section (consecutive number)

Structured Text (ST)
ST is a text language as per IEC 1131that displays operations such as invocations of function blocksfunctions, conditional execution of instructions, repeat of instructions, etc. through instructions.

Structured variables
Variables that are assigned a Derived data type that is defined with STRUCT (structure).

A structure is a collection of data elements, generally with different data types (Elementary data types and/or Derived data types).

Source code file (Concept EFB)
The source code file is an ordinary C++ source file. After executing the menu command `Library`→ `Generate files` , this file will contain an EFB code frame, where a specific code for the selected EFB must be entered by invoking the menu command `Objects` → `Source` auf.

SY/MAX
In Quantum controllers, Concept includes the provision to I/O Map SY/MAX I/O modules for RIO control by the Quantum PLC. The SY/MAX remote rack has a remote I/O adapter in slot 1 that communicates via a Modicon S908 R I/O system. The SY/MAX I/O modules are listed for your selection and inclusion in the I/O map of the Concept configuration.

System data types
In the current version, system data types include the data types ANL_IN and ANL_OUT.

T

Template file (Concept EFB)
The template file is an ASCII file containing layout information for the Concept FBD editor and parameters for generating code.

Temporary storage
Temporary storage is temporary memory for cut or copied objects. These objects can be inserted into sections. Each time something new is cut or copied, the old content in temporary storage is overwritten.

TIME
TIME represents the data type ”duration”. It is entered as a duration literal. The length of data elements is 32 bits. The value range for variables of this data type is from 0 to 2 exp (32) –1. The unit for the data type TIME is 1 ms.

Token
The network ”token” controls the temporary possession of transmittal rights by an individual node. The token passes the nodes in a rotating (ascending) address sequence. All nodes are tracking the token rotation and can receive any data that is sent along.

Traffic Cop
The Traffic Cop is an I/O map, which is generated from the user I/O map. The Traffic Cop is scheduled in the PLC and contains e.g. status information in addition to the user I/O map.

Transition
The condition whereby control passes from one or more predecessor steps to one or more successor steps along a directed link.

U

UDEFB (user–defined elementary functions/function blocks)
Functions or function blocks created in C which Concept makes available in libraries.

UDINT
UDINT represents the data type ”unsigned double integer”. It is entered as an integer literal, base 2 literal, base 8 literal or base 16 literal. The length of data elements is 32 bits. The value range for variables of this data type is from 0 to 2 exp (32) –1.

UINT
UINT represents the data type ”unsigned integer”. It is entered as an integer literal, base 2 literal, base 8 literal or base 16 literal. The length of the data elements is 16 bits. The value range for variables of this data type is from 0 to 2 exp (16) –1.

Unlocated Variable
The variable is maintained and stored by the system. The assigned address in the PLC is not published because the variable is addressed by its symbolic name.

V

Variables
Variables are used for data exchange within sections, between several sections, and between the program and the PLC.

If a variable is assigned a direct address (reference), it is called a located variable. If a variable is not assigned a direct address, it is called an unlocated variable. If the variable is assigned a Derived data type, it is called a multi–element variable.

In addition, there are constants and literals.

W

Warning
If a critical state is detected while an FFBs or step is processing (e.g. critical input values or time limit was exceeded), a warning occurs that can be viewed using the menu command `Online` → `Online events...`.In FFBs, the ENO output remains at ”1”.

WORD
WORD represents the data type ”bit string 16”. It is entered as base 2 literal, base 8 literal or base 16 literal. The length of the data elements is 16 bits. This data type cannot be assigned a numeric value range.

X

Y

Z