Use of function blocks for DM-FCx

Abstract

The DetStudio offers a set of function blocks to facilitate programming of temperature control in rooms using input output modules DM-FCA and DM-FCT.

Author: Libor Urbačka
File: ap0036_en_01.pdf

Attachments

–	None

Contents

Revision history

Version	Date	Changes
001	08. 09. 2009	New document.

Related documentation

1. Help tab in the DetStudio development environment
2. Operation manual for module DM-FCA
   file: dm-fca_g_en_xxx.pdf
3. Operation manual for module DM-FCT
   file: dm-fct_g_en_xxx.pdf
4. Data sheet for controller NOA70
   file: noa70_d_en_xxx.pdf
5. Application note AP0016 – Principles of RS485 interface usage
   file: ap0016_en_xx.pdf
6. Application note AP0025 – Communication in ARION network – table definition
   file: ap0025_en_xx.pdf
7. Application note AP0031 – Using day plans in control systems
   file: ap0031_en_xx.pdf

1 Range of use of modules DM-FCA and DM-FCT

DM-FCA and DM-FCT are I/O modules designed for fancoil units control. The module inputs allow connection of an on-wall controller to measure the room temperature, set a setpoint temperature correction, switch between control room modes and the fan. The modules' outputs serve to control the fancoil unit fans and control the supply of heating and cooling medium.

Therefore, the field of application is the same as in DM-REFACO modules. However, unlike them, the DM-FCx modules are not capable of autonomous operation since they do not have their own control intelligence. A superior control system has to supply this intelligence; the modules are connected to it by means of the ARION communication network. The advantage is that the functions can be modified easily by means of a standard development environment DetStudio.

2 Programme operation of DM-FCx modules

In order to work with the control system successfully, all modules DM-FCx used must be integrated properly into the network of I/O ARION modules. That includes correct connection of the communication bus RS485 (see AP0016), setting addresses and communication speeds properly (manuals on DM-FCA and DM-FCT) as well as integration and parametrization in the control programme in DetStudio (AP0025).

2.1 Function blocks for individual steps of the process

The entire process of temperature control in a room using a fancoil unit with application of a DM-FCx module can be described in several steps. It entails:

1. Reading the inputs (values measured and statuses)
2. Determining the setpoint temperature of the room
3. Deciding whether the room needs to be cooled or heated
4. Determining the fan power
5. Setting the action for heating and for cooling
6. Recording the outputs

The system function blocks (roughly) correspond to these process steps (the blocks are installed along with DetStudio). They can facilitate the programming of the whole temperature control process significantly.

We can use the function blocks in the programme in the way they have been supplied, or modify them according to specific requirements, since their code is written in DetStudio language.

2.2 Limitations of function blocks

Function blocks facilitate programme writing, especially if the same function is to be applied repeatedly, but they do not save memory. During the generation stage, the function block code is inserted into the source code as many times as the function block is used in the programme. Similarly, internal variables and parameters are generated into the database. The memory load for the programme code is therefore (approximately) the same as if the same code has been written repeatedly and no function blocks were used. When using a large amount of function block calls, it may easily happen that the generation ends up in a failure – the programme cannot be constructed due to insufficient memory in the control system.

When using function blocks to operate DM-FCx modules, we need to take the following limitations into account:

- A single process accommodates operation of maximum six DM-FCx modules using a complete set of function blocks.
- The total amount of DM-FCx modules operable by means of system function blocks depends on the control system type:
  - For 40MHz systems (AMiNi-ES, AMiNi2S, ADOREG, APT3100S), it is possible to operate up to 30 DM-FCx modules by means of system function blocks, or 23 DM-FCx + 23 NOA70.
  - For older systems (AMiRiS99, AMAP99, ADIS, ADOS, ADiR), it is possible to operate up to 26 DM-FCx modules, or 21 DM-FCx + 21 NOA70.

  The numbers stated are maximum; in this case, the control system only controls the specified peripheries.

If we need to operate more DM-FCx modules, we need to:

- Create our own function blocks, e.g. by removing unnecessary functions from the system blocks (e.g. leave only a single mode of heating and cooling valves control in FCxHCReg).
- Not use individual switch between cooling and heating for each room but use a common switch for the whole object.

- Use multiple control systems.
- Use modules from the Room library with the same function.

2.3 Setting function blocks' parameters

Typically, multiple DM-FCx modules are connected to the control system and multiple rooms are controlled. In order to facilitate writing of function block calls and creation of variables, we opted the following way of parameter transfer.

Parameters that are set for each room and are/can be individual of each room, are transferred in matrix variables with m lines, where m is the number of rooms controlled. Such parameters include for example setpoint and measured temperatures, the room mode, the fan mode. Furthermore, the line index common for all variables is also transferred into these matrices into the function block. Calling function blocks for selection of the required value for six room subsequently looks like this:

```
fb_Tiset 0, RoomMode, @r01_stop, r01_TS_t, r01_TS_T, DaysOff, Tscor, Tsdcool, Tsconst, Ts
fb_Tiset 1, RoomMode, @r02_stop, r02_TS_t, r02_TS_T, DaysOff, Tscor, Tsdcool, Tsconst, Ts
..
fb_Tiset 5, RoomMode, @r06_stop, r06_TS_t, r06_TS_T, DaysOff, Tscor, Tsdcool, Tsconst, Ts
```

Where variables **RoomMode**, **Tscor**, **Tsdcool**, **Tsconst** and **Ts** are matrices of at least six lines. The values for room 1 are saved in matrix lines with index 0, values for room 2 in lines with index 1, to values for room 6 in lines with index 5.

Variables for time schedules must be autonomous for each room. They are matrices themselves.

2.4 Function blocks – description and use

Functionality of individual function blocks corresponds to the aforementioned steps of the control process.

2.4.1 Reading the inputs (values measured and statuses)

Modules DM-FCA and DM-FCT are equipped with the following inputs and outputs:

Analogue inputs

TEMP	room temperature measured, Ni1000
POT0	fan control
POT1	setpoint temperature correction
Ni1000	room temperature measured (other), Ni1000

Digital inputs

BUTT	switch mode button
CONT	contact

Digital outputs

LED	control LED signalling
REL0	fan level 1 ON
REL1	fan level 2 ON
REL2	fan level 3 ON

Analogue outputs

AO0 / LT0	cooling and heating control
AO1 / LT1	cooling and heating control

Inputs TEMP, POT0, POT1 and BUTT and digital output LED are designed to connect on-wall controllers for temperature measurement, toggling room modes, fan control and setpoint temperature correction. We recommend using the controller SENSIT OF 2009.

The digital output CONT serves to connect the sensor of room occupation, open window detection, etc.

The analogue input Ni1000 may serve e.g. to measure outside temperature.

In terms of programmes, the four signalling LEDs (LED0 to LED3) on the front side of the module are controlled the same as digital outputs.

Modules DM-FCA are equipped with analogue outputs AO0 and AO1 (range from 0 to 10 V), modules DM-FCT with triac outputs LT0 and LT1. In terms of programmes, both are controlled as analogue outputs.

The function block **`FCxIO`** serves to read input values and write output values. Its outputs are measured room temperature ([°C]), setpoint temperature correction ([%], range -100 .. 100), values measured in electric or relative values, room mode, fan mode and CONT input status. Furthermore, values of digital and analogue outputs are transferred into function blocks which subsequently writes them into the module DM-FCx.

Function block FCxIO

Room temperature measured

We get a valid value when using the temperature sensor Ni1000/6180 ppm (part of the controller recommended). If a different-type sensor is connected to the TEMP input, it is necessary to use the parameter **`oAI`** to determine the temperature, either of the element **`oAI[Index, 3]`**, which is a value designed for conversion by the module **`Ni1000U2T`**, or **`oAI[Index, 4]`**, which is the voltage value measured, ranging from 0 to 2.5 V. Interpretation of this value then depends on the sensor applied and needs titten in a programme outside of the function block.

Setpoint temperature correction

Output ranging from -100 % to 100%. The conversion to °C also has to be performed outside the function blocks in the user programme.

Fan mode

When using the recommended controller (a resistor divider with four levels at 2.5 kΩ is used to switch the mode) the function block’s output is a fan mode in the following form:

Value	**Description**
0	OFF
1	Level 1
2	Level 2
3	Level 3
4	Automatic

When using another fan controller, it is necessary to use the output in parameter **`oAI[Index, 1]`** in the programme, in the range from 0 to 100 % and interpret it in individual fan levels.

Room mode

In function blocks, the room mode is used in the following sense

Value	**Description**
0	Time schedule
1	Energy-saving mode
2	Comfort

The input BUTT serves to set the room mode, and the LED output for signalling. By pushing the button on the controller (connected to the BUTT input) repeatedly, the mode Time Schedule and Comfort switch in cycles; the selected option is signalled by LED lights (Comfort lights, Time Schedule does not). We may also enter the energy-saving mode as well into the DM-FCx by means of the control system. The energy-saving mode is signalled in the same way as the Time Schedule mode, i.e. the LED is off.

The function block reads the mode from the DM-FCx in the form stated in the chart and transcribes it into the output variable.

We can also change the room mode by means of a programme in the control system (a programme from the terminal, by external entry from visualisation). In such case, the function block writes the mode value into DM-FCx, which also changes the signalling on the controller.

Therefore, when changes are made from the controller, the room mode value is then written from DM-FCx into the programme. When the mode is changed from the programme or we find that the mode value is invalid in DM-FCx (e.g. after resetting the DM-FCx), it is written from the programme into DM-FCx.

Temperature measured
The significance of temperature measurement on Ni1000 input is not fixed; temperature measurement is not necessary for temperature control in the room. That is why the function block does not have an autonomous output parameters for this temperature; the temperature measured is available in the parameter **`oAI[Index, 0]`** ranging from 0 to 2.5 V. When using the temperature sensor Ni1000, we can also convert the voltage measured into temperature by means of the module **`Ni1000U2T`** in the programme outside of the function block. Other application or omission of this value is up to the programmer.

CONT input
The output of a function block is the status of the digital input CONT. If the input CONT is connected (short-circuited), the output is **`@oCont == true`**. Interpretation of this value is up to the programmer.

Writing digital outputs
The function block also writes values of digital outputs into DM-FCx. They are transferred in the parameter **`iDO[Index, 0]`**, while the list of bits is as follows:

Bit	**Output**
0	REL0
1	REL1
2	REL2
3	LED0
4	LED1
5	LED2
6	LED3

Values for outputs REL0, REL1 and REL2 (fan control) may be created by means of the function block FCxFanReg; however, creating values for outputs LED0 to LED3 is up to the programmer.

Writing analogue outputs
The function block also writes analogue outputs values into DM-FCx. They are transferred in the parameter iAO[Index, *], while the list of bits is as follows:

Element	**Output**
iAO[Index, 0]	AO0 / LT0
iAO[Index, 1]	AO1 / LT1
iAO[Index, 2]	LED brightness

It applies for triac outputs that the output is turned off when the value is 0 and turned on when the output is higher than 0.

Devices NOA35 or NOA70 (in mode 2) may also be used as on-wall controllers; they do not connect to DM-FCx inputs, but to the ARION network. Special function blocks are designed for cooperation with NOA controllers. When we use the special blocks, parameters **`oTim`**, **`oTscor`**, **`ioRMode`** a **`oFanMode`** in the function block **`FCxIO`** may be filled with variables that will not be used any further (unfortunately, it is impossible to transfer value NONE into a parameter), or read AI and

DI and write DO by means of modules **ARI_AnIn**, **ARI_DigIn** a **ARI_DigOut** same as in ARION peripheries.

2.4.2 Determining the setpoint temperature of the room

Function block FCxTiset

The function block **FCxTiset** serves to determine the setpoint temperature in the room. Requests for heating, cooling, room mode, temperature time schedule, days off, setpoint temperature correction, increasing the setpoint temperature from the time schedule for cooling and constant setpoint temperatures for energy-saving and for Comfort are entered into the function block. The output is the setpoint temperature for heating (**oTiset[Index, 0]**) and the setpoint temperature for cooling (**oTiset[Index, 1]**).

If the parameter **@iOff == true**, the setpoint temperature for heating is 5 °C, the setpoint temperature for cooling is 30 °C. These values are entered in internal variables of the function block (**Tz_mmin**, **Tz_mmax**) and they are applied also for limiting both setpoint temperatures from both sides. If other temperatures are setpoint as limit temperatures, we need to change these values in the function block code.

In case of **@iOff == false** the setpoint temperature is determined according to the room mode.

In the Time Schedule mode (**iRMode[Index, 0] == 0**), the resulting setpoint temperatures are determined by means of parameters **iTPtime** (matrix of time shifts for the time schedule), **iTPtemp** (matrix of setpoint temperatures for time schedule), **iTPFeasts** (matrix of days off for time schedule), **iTscor** (setpoint temperature correction) and **iTscdif** (increasing the setpoint temperature for cooling).

The matrix of the time schedule, times and temperatures has to have 8 lines, and the individual lines stand for the following:

Line	Description
0	Monday
1	Tuesday
2	Wednesday
3	Thursday
4	Friday
5	Saturday
6	Sunday
7	Day Off

If another (lower) number of matrix rows is required, it is necessary to adjust the function block – parameters of the module **DayPlan2**.

More on the use of time plans in DetStudio documents (modules **DayPlan**, **DayPlan2**, **Holiday**) and in the application note AP0031.

The number of time shifts (matrix columns) is not fixed. Six time shifts in a day is sufficient for ordinary application, however, there can be more or less of them if needed. However, we also need to take into account that the more time breaks there are, the longer the period of execution of module **DayPlan2**. If we have a larger amount of rooms controlled and frequent assessments of time plants, the prolonging may have some negative effects.

The value of correction **iTscor[Index, 0]** is added to the setpoint temperature from the time plan. It is entered in °C. The result is temperature for heating. The setpoint temperature for cooling is gained by adding the cooling increase **iTscdif[Index, 0]** to the setpoint temperature for heating. The non-zero increase of setpoint temperature for cooling is important when toggling heating and cooling according to room temperatures (further in the description of the function block

FCxHCsw4). If heating and cooling are switched in another way (according to the outdoor temperature, manually), we may set zero increase.

In modes energy-saving (**iRMode[Index, 0] == 1**) and Comfort (**iRMode[Index, 0] == 2**), setpoint temperatures are taken from the parameter **iTiset**. In the energy-saving mode, the value in **iTiset[Index, 0]** is considered the setpoint temperature for heating and **iTiset[Index, 2]** for cooling. In the Comfort mode, the value in **iTiset[Index, 1]** is considered the setpoint temperature for heating and **iTiset[Index, 3]** for cooling. According to the mode names, it should apply that **iTiset[Index, 0] < iTiset[Index, 1]** and **iTiset[Index, 2] > iTiset[Index, 3]**. The comfort temperature for heating is higher than the energy-saving temperature, and on the contrary, the comfort temperature for cooling is lower than the energy-saving temperature.

The Comfort mode is temporary. If the module **DayPlan2** recognizes a time shift, the Comfort mode changes into Time Schedule. We can change the behaviour by adjusting the function block code (value of the variable **mskKomfR**) to cancel the Comfort mode at the moment of the change of the time plan value, at midnight; combine these two events or leave the Comfort mode as permanent.

2.4.3 Cooling and heating control

In general, fancoil units can be used for both heating and cooling of the room. This sub-chapter deals with systems that combine heating and cooling by means of fancoil units. In case of heating systems exclusively (or cooling systems exclusively), there is no need to deal with toggling between heating and cooling at all.

Combined heating and cooling systems can be classified to two-pipe and four-pipe systems, depending on the manner of supply of heating and cooling medium into fancoil units.

Four-pipe systems
These systems are equipped with autonomous supply systems of heating and cooling medium (a heating circuit – supply and gate, cooling circuit – supply and gate; four pipes total). Each fancoil unit has autonomous control valves for heating and for cooling.

Two-pipe systems
The medium supply is common for both heating and cooling. The entire heating circuit is connected centrally to the source of heat or cold.

The control programme has to provide switching between heating and cooling for both systems so that only heating or only cooling is applied at a given moment. On principle, this switching is common for the entire system in two-pipe systems; it is impossible that some units should heat and others cool at the same time in a two-pipe system. However, it is possible in a four-pipe system.

We may take the outdoor temperature as a criterion for central switching. Heating is necessary during low outdoor temperatures; cooling during high outdoor temperatures. Furthermore, we need to make sure that the switching does not occur too frequently – it is unsuitable and energetically demanding to heat the room up when temperature outside is low in the morning and to turn up cooling when the temperature outside rises in the afternoon, only to heat up again when the temperature drops in the evening.

We may use the room temperature to make individual switches. Same as in central switching, it is necessary that heating and cooling do not switch too frequently.

The function block **FCxHCswG** is designed for central switching between heating and cooling, and the function block **FCxHCsw4** serves for individual switching.

Function block FCxHCswG

The block is designed for central switching of heating and cooling systems; it is typically used once for the entire system. The switching is performed on the basis of temperature outside or direct enabling for heating or cooling. That may occur manually or generated by the programme – e.g. a signal of central heating shutdown disables heating and enables cooling. The function block provides that heating and cooling are not enabled at the same time; heating has the priority.

The immediate value of the outdoor temperature is set into the parameter **`iTe`** in the function block. The function block uses the temperature to calculate the dimmed outdoor temperature (using a first order filter with a time constant transferred in the parameter **`iTeFc`**). With sufficient dimming (filter time constant of several hours), the dimmed outdoor temperature represents a long-term trend of outdoor temperature development, short-term deviations are suppressed.

In order to switch according to the outdoor temperature, the decisive limits are set in the parameter **`iTeLvls`**. The immediate and dimmed values of the outdoor temperature are compared to these limits. If both values exceed **`iTeLvls[0, 1]`**, it switches to cooling; if both values are lower than **`iTeLvls[0, 0]`**, it switches to heating. This also makes provisions for long-term trends, expressed by dimmed outdoor temperature, as well as for the immediate condition represented by the immediate value of the outdoor temperature.

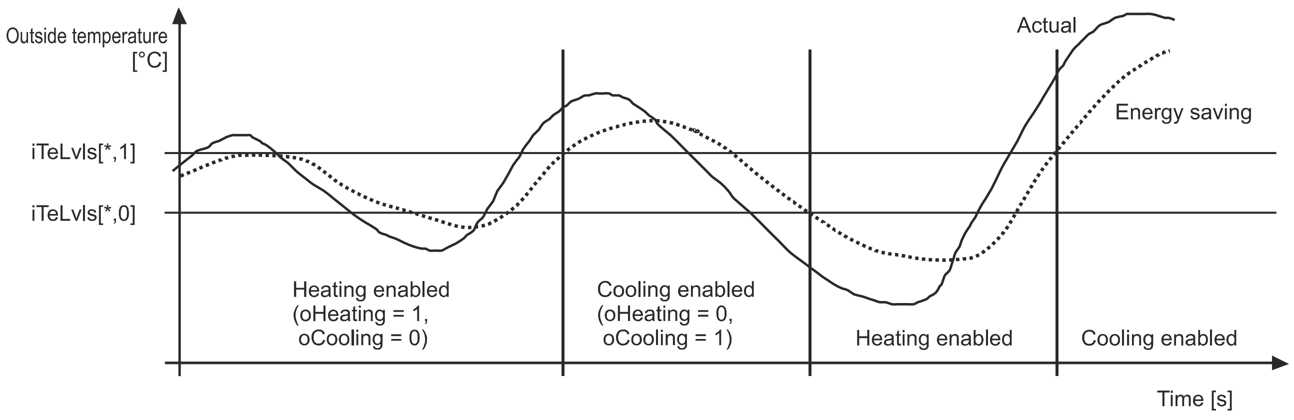

Fig. 1 – Automatic switching between heating and cooling according to the outdoor temperature

In order for it to function properly, the limit value for cooling must not be lower than the limit for heating (**`iTeLvls[0, 0] < iTeLvls[0, 1]`**).

Function block FCxHCsw4

It is designed for four-pipe systems (autonomous supply of heating and cooling medium) and individual switching between heating and cooling for individual rooms. Therefore, the function block is used as many times as there are rooms controlled.

The task of function blocks is to decide whether heating or cooling is needed, according to the setpoint room temperature (the parameter **`iTiset`**, setpoint for heating **`iTiset[Index, 0]`**, setpoint for cooling **`iTiset[Index, 1]`**; corresponds to the output of the function block **`FCxTiset`**) and room temperature measured (**`iTim`**). At the same time, the function blocks must provide that the heating and cooling are not switched too frequently. The function block uses two methods to eliminate frequent switching.

Various setpoint temperatures for heating and for cooling

The setpoint temperature for cooling must be higher than the setpoint temperature for heating. This is a logical request; it makes no sense to heat up a room to a higher temperature than the temperature it is cooled down to. In order for the function block **`FCxHCsw4`** to work properly, we must maintain the minimum difference of 1 °C. If the setpoint temperature for cooling is lower than the one for heating, the function block leaves the previous condition of switching (output bits

`@oHeating` and **`@oCooling`**) regardless of setpoint temperature values and of the temperature measured.
*If the function block **`FCxTiset`** is used to determine setpoint temperatures, the increase of setpoint temperature for cooling above the temperature for heating is set in the parameter **`iTscdif`**.*

Use of integer criterion
If the actual room temperature gets into the opposite zone (heating is enabled and the actual room temperature exceeds the setpoint temperature for cooling or vice versa if cooling is enabled and the actual temperature drops below the setpoint temperature for heating), the cooling and heating does not switch until the time integral of the deviation exceeds the setpoint limit (**`iHeatIntLevel[Index, 0]`** for swithing to heating and **`iCoolIntLevel[Index, 0]`** for switching to cooling). The deviation integration causes the switch not to be immediate but delayed, which is indirectly proportional to the amount of deviation – when the deviation is twice as high, the time from excess to switch is shortened by half. This prevents frequent switching in short-term temperature deviations.

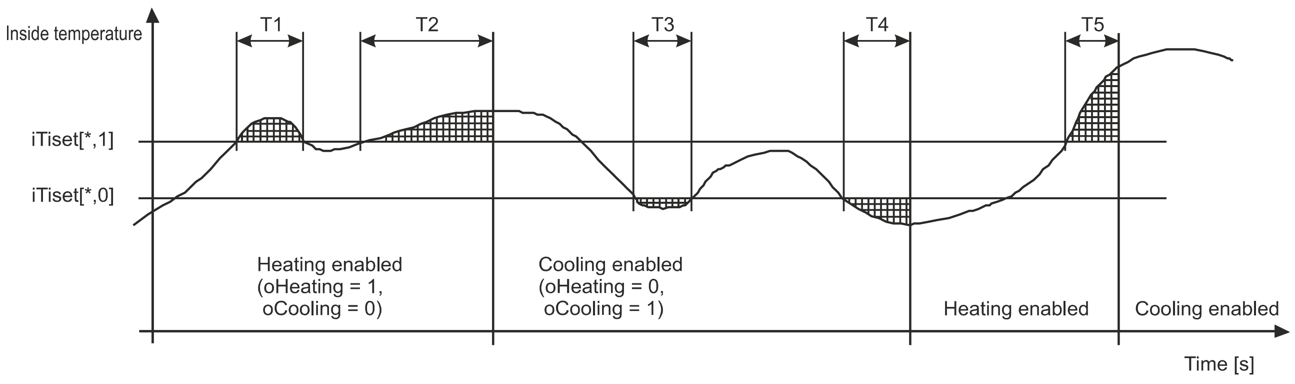

Fig. 2 – Automatic heating switch – cooling according to the room temperature

The figure illustrates automatic switch. The highlighted areas are integrals counted. Obviously, the overshoot in cases T1 and T3 was low and finished quickly, so the integral has not exceeded the set limit. Therefore, there was no switch. In cases T2, T4 and T5, the limit has been exceeded and the switch has occurred. The comparison of lengths T2 and T5 shows the feature of the integer criterion, when higher overshoot (T5) causes the mode to switch more quickly. We must note that in cases T1, T2 and T5, the integral was calculated for switch to cooling **`CoolInt`** and compared to the value **`iCoolIntLevel[Index, 0]`**. Similarly, in case T3 and T4, the integral for switch to heating was calculated **`HeatInt`** and compared to the value **`iHeatIntLevel[Index, 0]`**.

In some situations, the switching must be prevented. One of them occurs when the setpoint temperature for heating is higher than the setpoint temperature for cooling. This may happen e.g. by incorrect parameter setting, but also by switching into the Comfort mode which has a typically high setpoint temperature for heating and low setpoint temperature for cooling. In such a case, the function block keeps the previous values of output bits **`@oHeating`** and **`@oCooling`**. The same behaviour can be induced in the function block from the outside as well, by setting the parameter **`@iRetain`** to the value **`true`**.

After a cold start of the application, the function block's outputs **`@oHeating`** and **`@oCooling`** are **`false`**. The programme may impose the required values on them from the outside.

2.4.4 Determining the fan power

Modules DM-FCA and DM-FCT allow control of up to three-level fans using digital outputs REL0 to REL2. The function block **`FCxFanReg`** is designed for software operation.

Function block FCxFanReg

The function block determines the fan power in the range from 0 to 100 % on the basis of fan mode and setpoint room temperature and temperature measured and converts this value to the switch of max. one of these digital outputs.

The fan mode is identical to the mode that is the output of the function block **`FCxIO`**, parameter **`oFanMode`**. In other than automatic mode, the setpoint fan switching on (level 0 to 3) is only converted to digital outputs.

In automatic mode, the action using a PI controller is calculated on the basis of the setpoint room temperature and the room temperature measured. When **`@iHeating == true`**, the setpoint temperature for heating is used, for **`@iCooling == true`** (and **`@iHeating == false`**) the setpoint temperature for cooling is used. If neither heating nor cooling are enabled, the fan remains off in the automatic mode. Both gain and the integral time constant are added into the function block in parameters **`iPIK`** and **`iPITi`**.

The resulting value (0 to 100 %) is converted to switch on one of three digital outputs by comparison of action limit values transferred in **`iFanLvls`** (compared to hysteresis in order to prevent undesirable output overshoot). Values of digital outputs are available in three lowest bits of parameter **`oDO`** (**`oDO[Index, 0].0`** for level 1, **`oDO[Index, 0].1`** for level 2 and **`oDO[Index, 0].2`** for level 3).

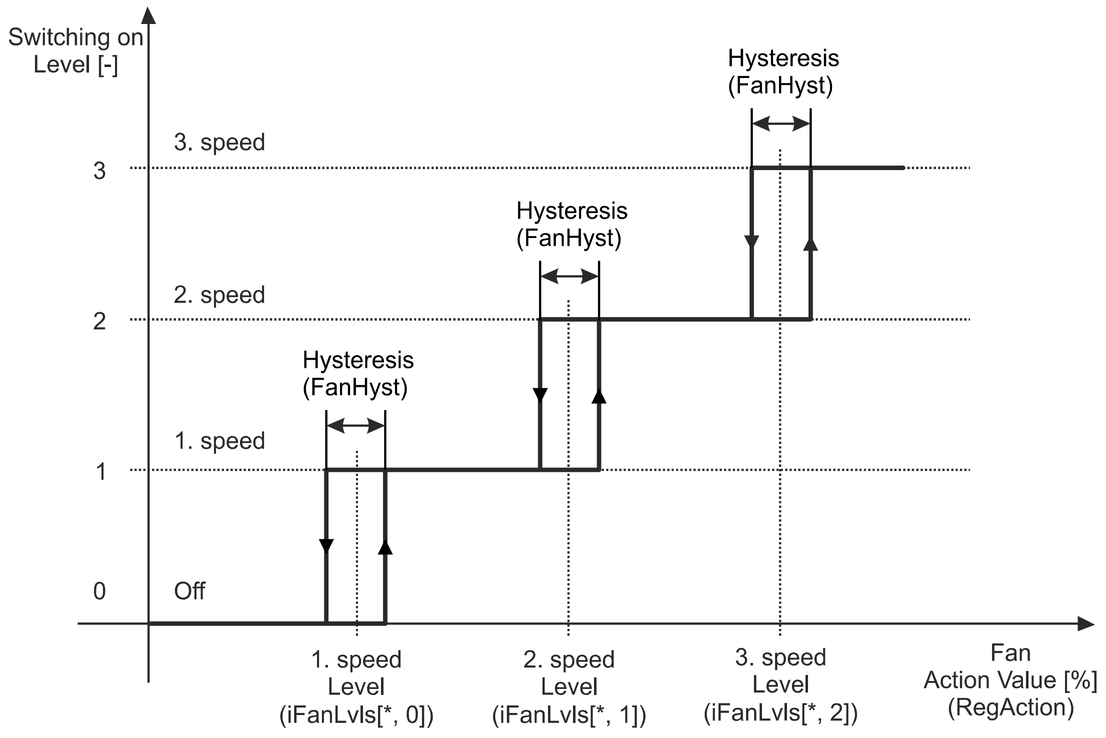
Switching on Level [-]
3
2
1
0
3. speed
2. speed
1. speed
Off
Hysteresis (FanHyst)
Hysteresis (FanHyst)
Hysteresis (FanHyst)
1. speed Level (iFanLvls[*, 0])
2. speed Level (iFanLvls[*, 1])
3. speed Level (iFanLvls[*, 2])
Fan Action Value [%] (RegAction)

Fig. 3 – Automatic control of a three-speed fan

If the counted action is **`RegAkce < iFanLvls[Index, 0]`**, all three digital outputs are **`false`**.

Output **`iDO[Index, 0].0`** has value **`true`** for **`RegAkce > iFanLvls[Index, 0]`** and at the same time **`RegAkce < iFanLvls[Index, 1]`**.

Output **`iDO[Index, 0].1`** has value **`true`** for **`RegAkce > iFanLvls[Index, 1]`** and at the same time **`RegAkce < iFanLvls[Index, 2]`**.

Output **`iDO[Index, 0].2`** has value **`true`** for **`RegAkce > iFanLvls[Index, 2]`**.

Writing values of digital outputs into module DM-FCx can be performed by means of the function block **`FCxIO`**, or by means of the module **`ARI_DigOut`** if the function block **`FCxIO`** is not used in the application.

2.4.5 Setting the action for heating and for cooling

Function block FCxHCReg

The action for heating and cooling is calculated on the basis of the setpoint temperature and the temperature measured and enabled heating and cooling. Those actions are subsequently converted into output values and written into the DM-FCx module. The function block provides that the there is maximum one non-zero action for heating and cooling at a given moment.

The heating (**`@iHeating`**) or cooling (**`@iCooling`**) enabled is transferred into the function block, the setpoint temperature for heating (**`iTiset[Index, 0]`**) and for cooling (**`iTiset[Index, 1]`**) and the room temperature measured (**`iTim[Index, 0]`**). The action is calculated either by means of a PI controller or a hysteresis controller. It depends on the manner of heating and cooling fans control which is transferred in the parameter **`iHCMode[Index, 0]`**.

Value	Control	For hw module
0	Proportional	DM-FCA
1	DAT (slow PWM)	DM-FCT
2	PAT (positional)	DM-FCT
3	ON/OFF	DM-FCT
4	Two-level	DM-FCT
5	Sequential	DM-FCA

Description of individual control modes

Proportional
Actions for heating and cooling are calculated by means of PI controllers in the range from 0 to 100 %. These values are written into analogue outputs AO0 (action for heating) and AO1 (action for cooling), while 0 % corresponds to 0 V, 100 % corresponds to 10 V.

PI controller parameters

FB Parameter	Description
iHPIK	Heating controller gain [%.°C^{-1}]
iHPITi	Integral time constant of the heating controller [s]
iCPIK	Cooling controller increase [%.°C^{-1}]
iCPIK	Integral time constant of the cooling controller [s]

DAT (slow PWM)
Actions for heating and cooling are calculated by means of PI controllers in the range from 0 to 100 %. These values are converted into a width modulated pulse and written onto LT0 outputs (action for heating) and LT1 (action for cooling).

The pulse width modulation means that the percentage size of the action is converted into a proportional size of the pulseagainst the modulation period. The Pulse ratio is the length of the section within a single period in which the output signal has value **`true`**.

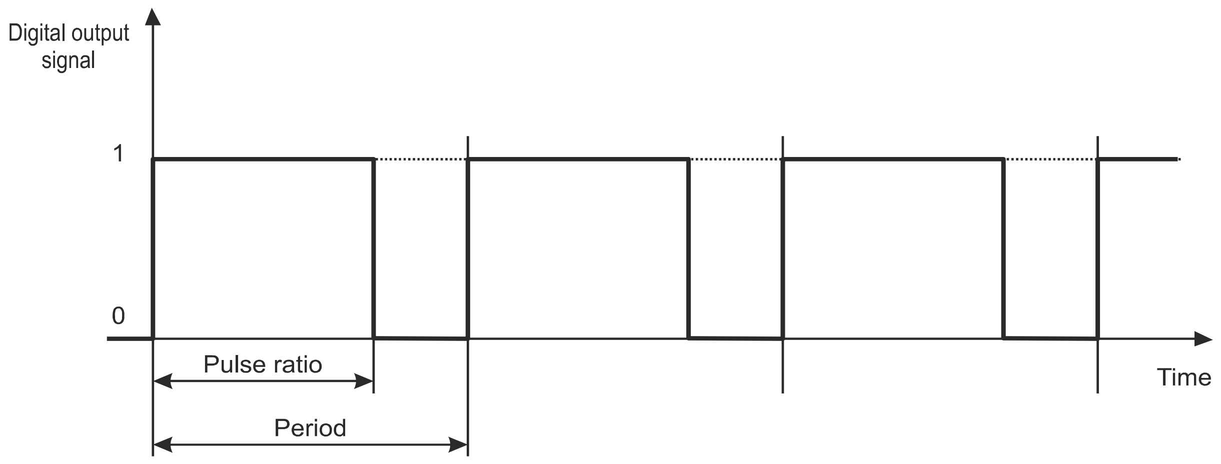

Fig. 4 – Pulse width modulation

The period for heating control is set by the parameter **`iOver1[Index, 0]`**, and for cooling by parameter **`iOver2[Index, 0]`**. The Pulse length is set according to the percentage value of the action variable (**`HeatAct`** for heating, **`CoolAct`** for cooling).

The Pulse length is limited in order to limit unnecessarily “short” output on or off switches that could overload action element. “Short” on or off switches represent pulses shorter than 5 % of the period. The Pulse is therefore limited so that either it is zero (the output permanently in “0”) or it ranges from 5 to 95 percent, or it is maximum (the output permanently in “1”).

PI controller parameters are the same as in proportional control.

Valve control parameters

FB Parameter	**Description**
iOver1	Period of the width modulated pulse for heating [s]
iOver2	Period of the width modulated pulse for cooling [s]

PAT (positional control)
This mode of control can only be used for two-pipe systems where both heating and cooling use a single common control valve controlled by outputs LT0 (valve more) and LT1 (valve less).

Actions for heating and cooling are calculated by means of PI controllers in the range from 0 to 100 %. The selected action (depending on enabled heating or cooling; parameters **`@iHeating`** and **`@iCooling`**) is converted to valve position control by means of two digital outputs according to the valve run time from one extreme position to the other.

PI controller parameters are the same as in proportional control.

Valve control parameters

FB Parameter	**Description**
iOver1	Valve run time from one extreme position to the other [s]

ON/OFF
Actions for heating and cooling are calculated by means of hysteresis controllers (on/off).

These values are written into outputs LT0 (heating on) and LT1 (cooling on).

The figure illustrates heating control.

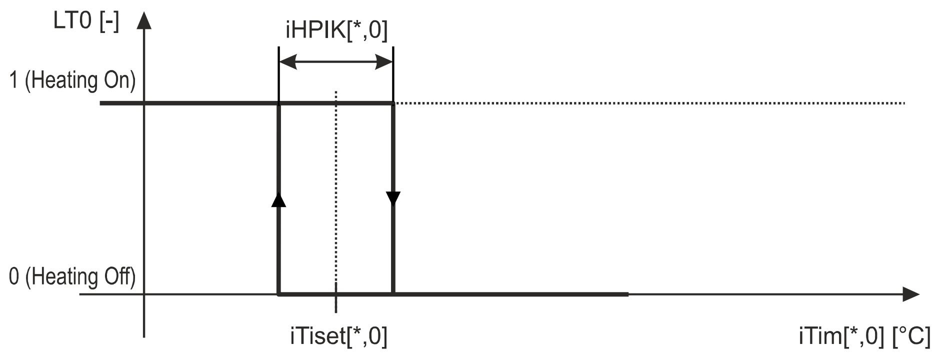

Fig. 5 – ON-OFF heating control

Another figure illustrates cooling control.

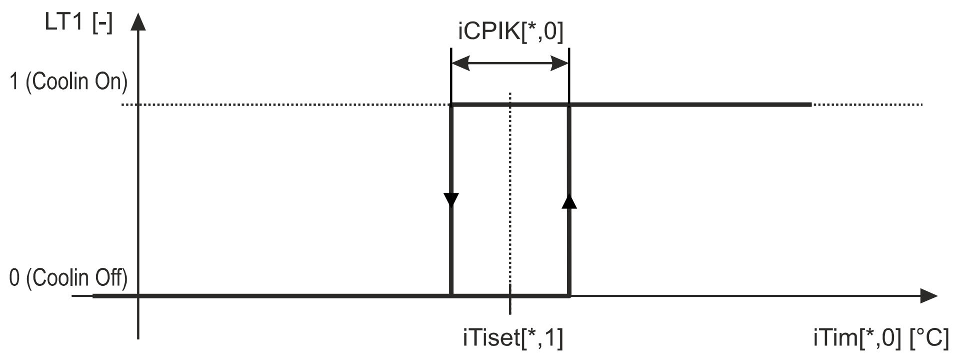

Fig. 6 – ON-OFF cooling control

Hysteresis controller parameters

FB Parameter	Description
iHPIK	Hysteresis for heating [°C]
iCPIK	Hysteresis for cooling [°C]

Two-level
This mode of control can only be used for two-pipe systems where both heating and cooling use a single common control element controlled by outputs LT0 (level 1 on) and LT1 (level 2 on).

Actions for heating and cooling are calculated by means of hysteresis controllers. For level 2, the setpoint temperature value moves down by value **`iOver1[Index, 0]`** (for heating) up by value **`iOver2[Index, 0]`** (for cooling).

The figure illustrates heating control.

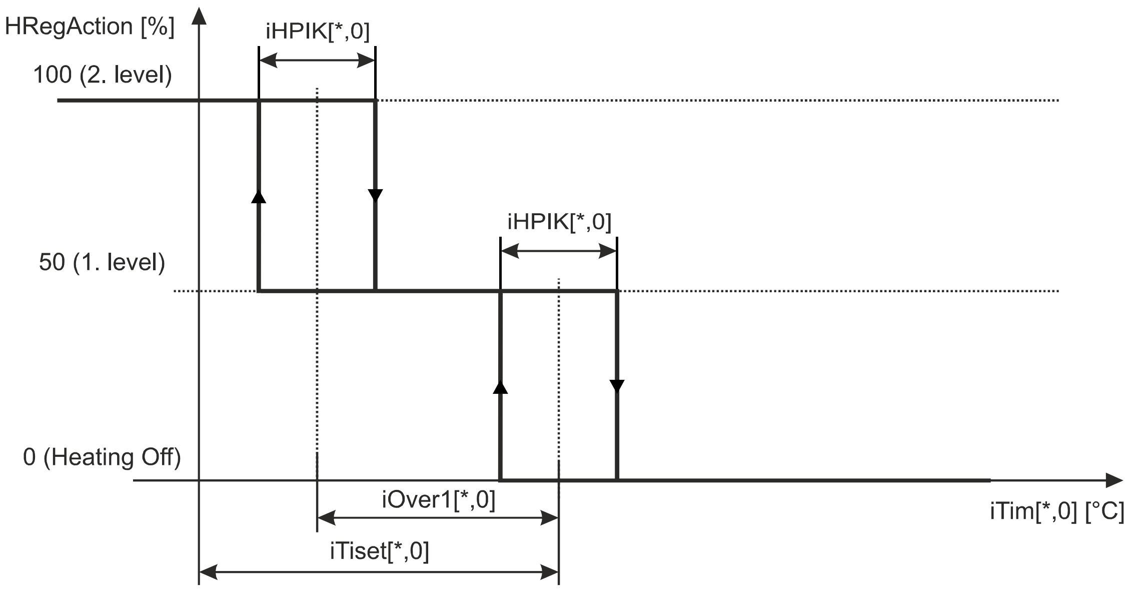

Fig. 7 – Two-level heating control

Another figure illustrates cooling control.

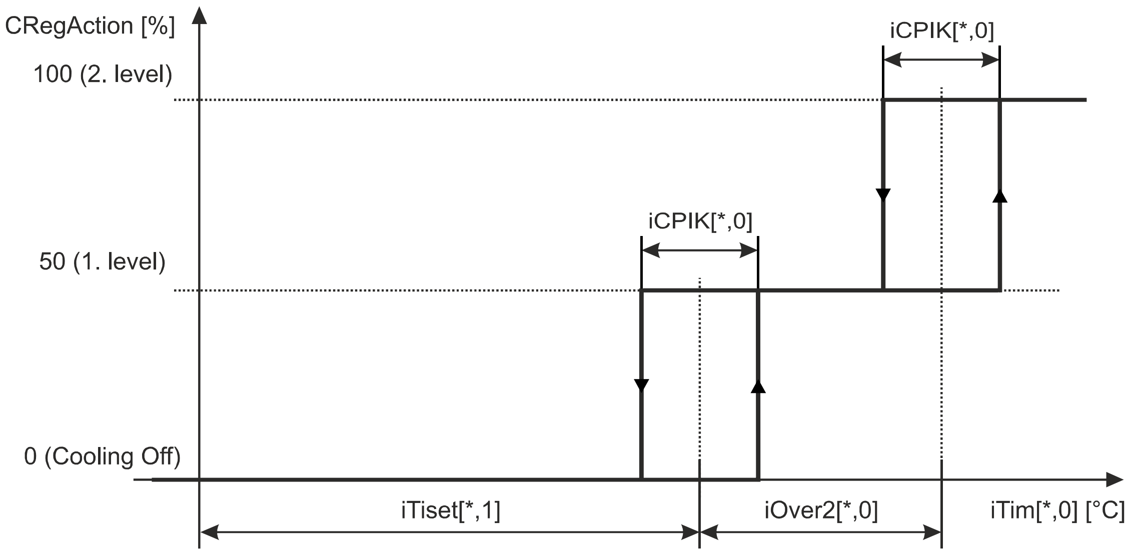

Fig. 8 – Two-level cooling control

The selected action (depending on heating or cooling enabled; parameters **`@iHeating`** a **`@iCooling`**) is written on outputs LT0 (level 1 on) and LT1 (level 2 on).

Hysteresis controller parameters

FB Parameter	Description
iHPIK	Hysteresis for heating [°C]
iCPIK	Hysteresis for cooling [°C]
iOver1	Shift in the setpoint heating temperature to turn on level 2 [°C]
iOver2	Shift in the setpoint cooling temperature to turn on level 2 [°C]

Sequential

This mode of control can only be applied in case the control elements for both heating and cooling are controlled by the common output AO0.

Actions for heating and cooling are calculated by means of PI controllers in the range from 0 to 100 %. These values are written on the analogue output AO0 in the following manner: A zero action limit is set. The range of the analogue output is therefore divided into two parts. The part of the range below the zero limit is designed for cooling; the part above the zero limit is designed for heating.

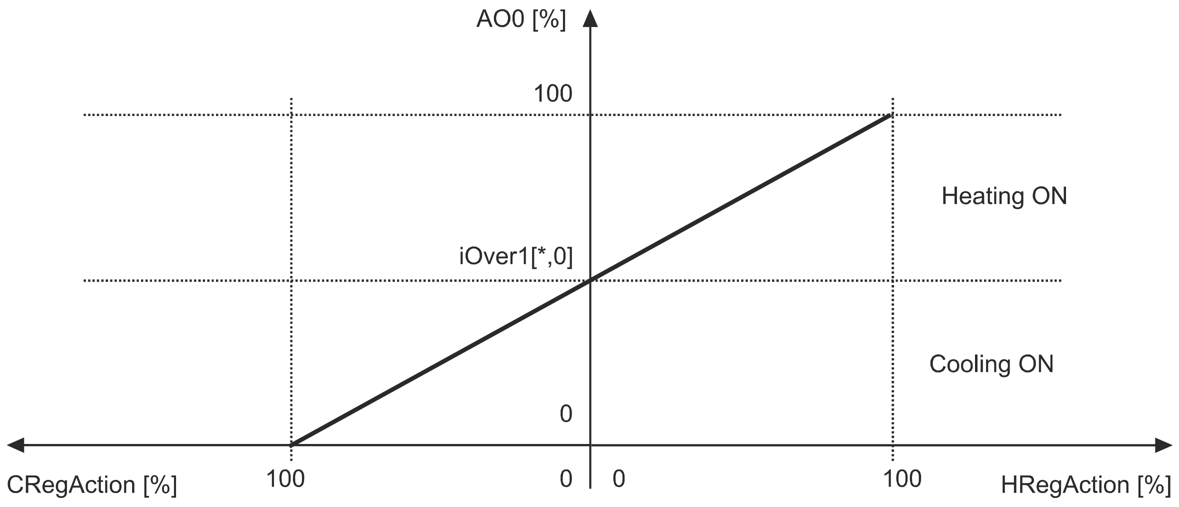

Fig. 9 – Conversion of action variables to the analogue output of the sequence control

PI controller parameters are the same as in proportional control.

Valve control parameters

FB Parameter	Description
iOver1	Zero action limit [%]

3 Technical support

All information on the application of function blocks for cooperation with DM-FCA and DM-FCT modules will be provided by the technical support department of the company AMiT. Do not hesitate to contact the technical support via e-mail using the following address: **support@amit.cz**.

4 Warning