NETCONTROL

Temperature regulator with permission

Application Note

rev. 1.2

10.11.2023

CONTENTS

Document versions

Version	Date	Brief description of the changes introduced
1.2	10.11.2023	Added new section 6
1.1	10.10.2023	Added note at the end of section 4. Added new section 5
1.0	09.06.2022	Initial version of the document

Legend:

The text contains additional and useful information that explains specific situations and features.

The text contains information of essential importance which you must get to know well!

1. Introduction

Automatic switching on/off of a load according to the value of a sensor is a classic task in automation systems.

On this principle, thermostats, automatic filling of tanks, ventilation of premises, control of pumps in solar systems and many others are realized.

NetControl series controllers have a built-in module "Automation", which is sufficient for solving such tasks. Combined with its network connectivity, this gives an even more complete level of remote control and monitoring.

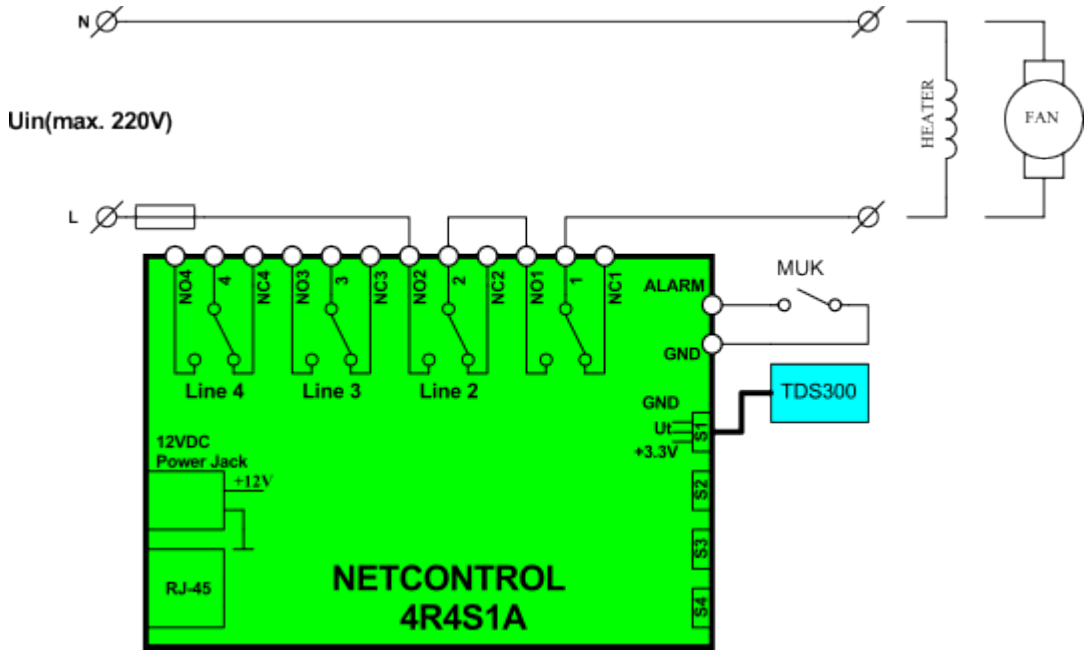
In this document we will illustrate the implementation of a classic heating thermostat with a function to activate/deactivate its operation.

With small changes in the macros or the connection of the load to the relay outputs, the action can easily be reversed and become a cooling thermostat.

2. Thermostat with resolution

2.1. Block circuit and initial conditions

The following image shows the schematic diagram of the thermostat. We use a ***NetControl 4R4S1A*** controller and ***TDS300*** temperature sensor. The load can be a heater, fan or other device, depending on the task to be solved.

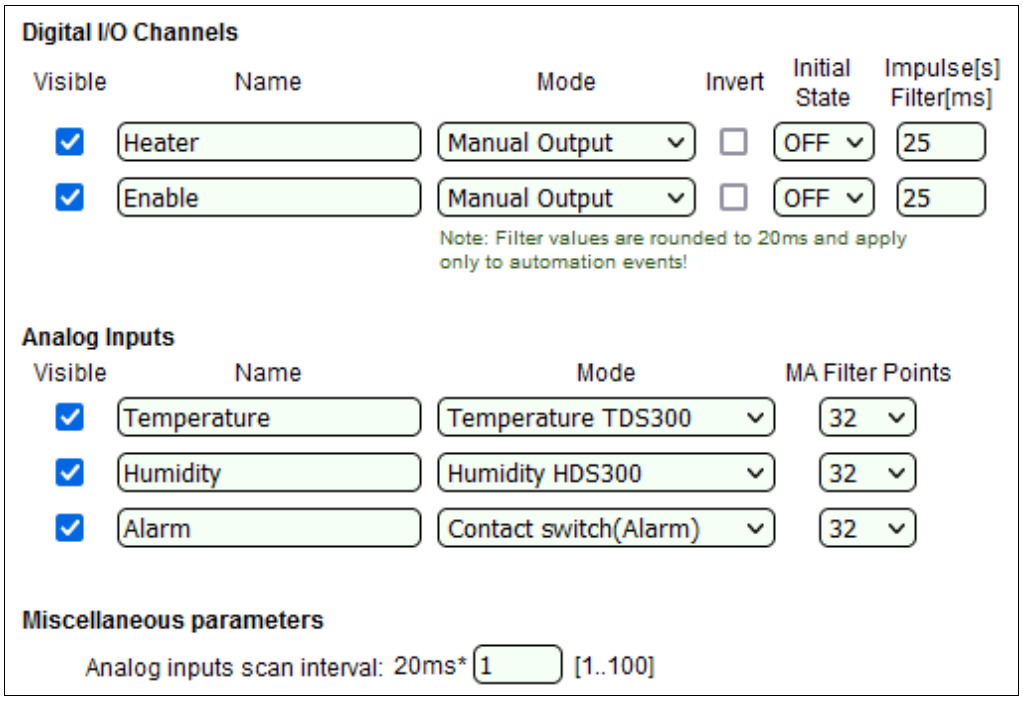


In our case we set ourselves the exemplary task of maintaining room temperature: we turn on the heater when the temperature drops below 21°C and turn it off when the temperature is above 23°C.

We use Line 1 relay output to control the load, and Line 2 output for resolution: if the Line 2 relay is on, only then the load is powered by the thermostat algorithm (since the thermostat algorithm cannot be turned off with one easy action, so the Line 2 button acts as a enable/disable switch).

2.2. Inputs and outputs setup

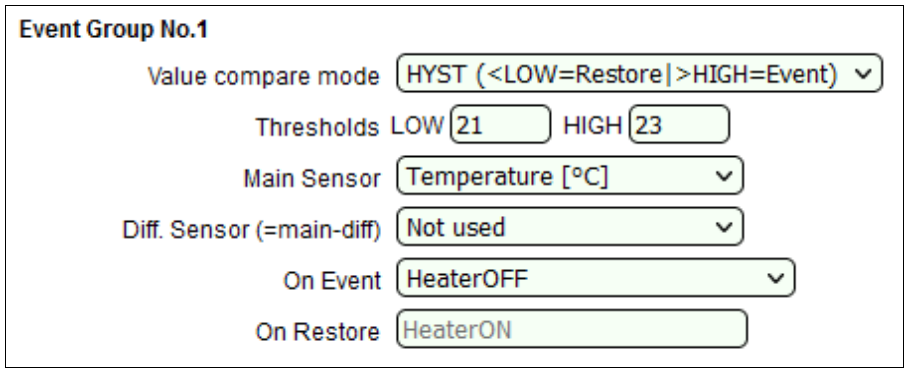
In the menu "IO Settings" we must have the following settings (these are the factory settings, only change names of the outputs, for greater convenience): Line 1 = Heater, Line 2 = Enable



2.3. Setup of "Automation"

Automation" menu and select the " HYST" operating mode (hysteresis). We set the lower and upper temperature threshold according to our assignment.

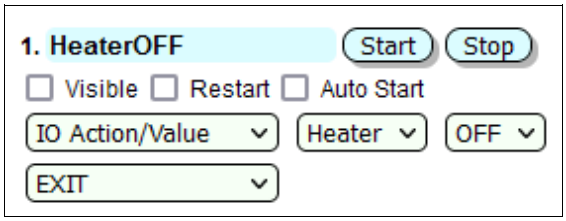
We must also select macros for " On Event" (when the temperature is >23°C) and for " On Restore" (when the temperature drops below 21°C). Since we have not set up the macros at this stage, you will not see the HeaterOFF/HeaterON names, but select Macros 1 (9) (after defining them in the next step, their names will appear here).

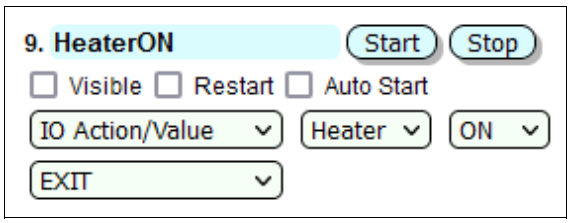


Have in mind that entered values for the temperature will be automatically rounded to the closest step of ADC.

2.4. Creation of macros HeaterOFF and HeaterON

In the previous step we chosed macros 1 and 9 to be responsible for load management and now we set them in the " Macros" menu:





With this the setting of the thermostat is ready and when the temperature changes you will see the changing state of Line 1/Heater. If the Line 2/Enable output is switched on - power will also be supplied to the load!

2.5. Change on speed on the thermostat

The presence of hysteresis in the on and off temperature setting ensures stable operation of the algorithm. But sometimes it is necessary to further limit the speed of an automatic algorithm.

In ***NetControl*** you can influence the speed by using the setting for “MA filter points” and “Analog inputs scan interval” in the “ IO Settings” menu (see picture in 2.2).

"MovingAverage” filter at each input averages the set number of recent measurements and the result is used to check for inequalities in Automation. If the temperature value in our example changes by one bit of the ADC, it will be necessary to enter at least 50% of the new value before the arithmetic mean is rounded to the new one.

Factory “MA Filter Points” = 32, and input scanning: every 20 ms. So we have 32 * 20 = 640 ms time to average the data in the filter. Practically (with some inaccuracy and conditionality) the reaction delay time with a stable change in the input value is 50% or 320 ms. With fluctuating values of the input value, the time will be longer.

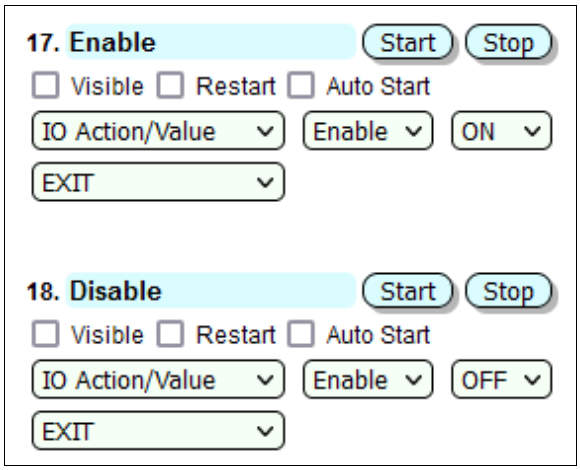
At maximum values of the parameters (256 * 2000 ms) we have averaging from 512s back.

3. Additional functions: enable by time of day

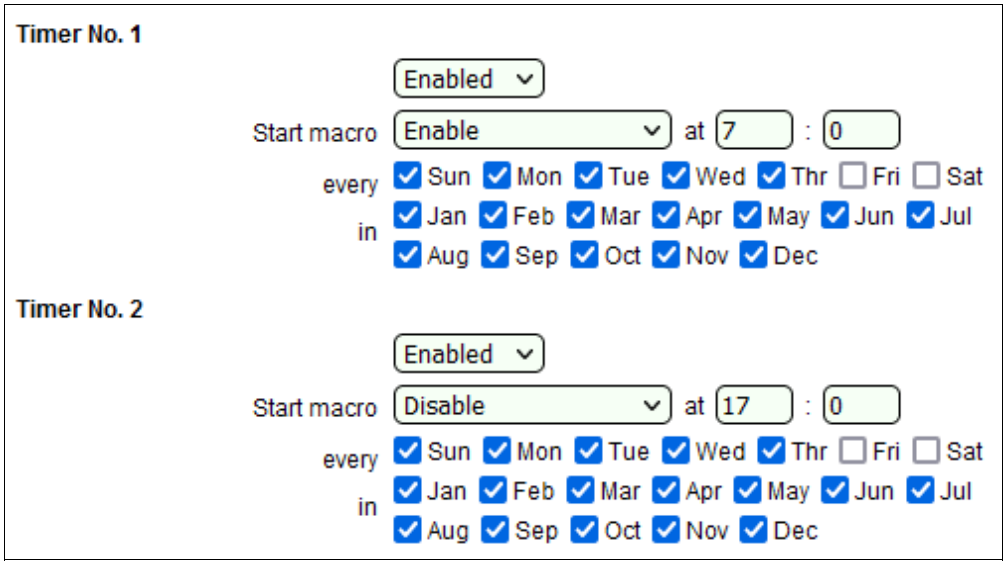
Let's expand the functionality of our thermostat by adding a function to enable it only at certain times of the day, for example: the thermostat only works in the interval from 07:00 to 17:00 on weekdays (space heating)

For this purpose we will use the block "Timers" (real-time clock events). We remind you that a properly configured SNTP server is required for this service to work.

Will use the already defined Line 2/Enable output, setting two new macros 17 and 18 to turn it on and off:



Add the two new timers in “Timers” menu:



If you still want to keep the manual function Enable/Disable, which can globally stop the thermostat (without making adjustments each time), you can connect the normally open contact of Line 3 in series with the circuit Line 2 → Line 1 Then, through the state of Line 3, we will be able to deactivate the thermostat completely.

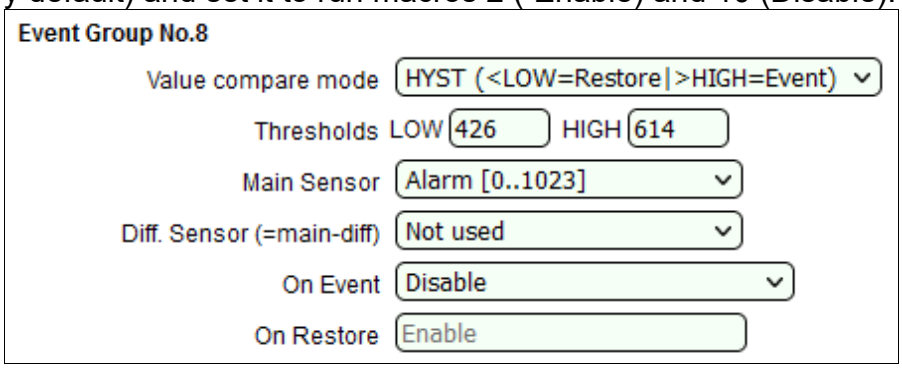
4. Additional functions: disable by Alarm input

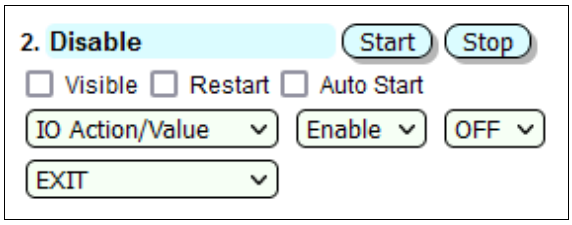
In hotels and other intelligent heating systems, the heat/cooling source is switched off when opening a door in a room.

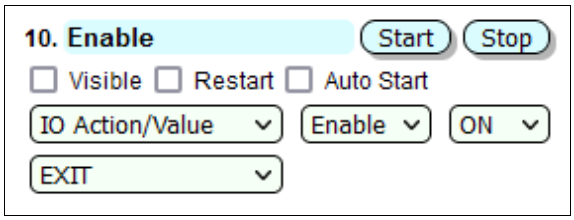
We can easily add such a feature to our ***NetControl*** thermostat using the Alarm input and connect a magnetic sensor (MUK) to the door / window.

Again, we will use the available Line 2/Enable as a signal to stop the power supply to the thermostat.

We add an Automation block to intercept the change in the state of Alarm (there is one defined by default) and set it to run macros 2 (Enable) and 10 (Disable):







When the door is opened, the sensor interrupts the circuit and the Alarm input becomes high (> 614, typically over 1020) - the macro " 2. Disable" is started, Line 2/Enable is switched off and the power supply to the heater is interrupted.

When sensor is closed, a low level occurs (ie <426, typically around 10) - the macro “10. Enable” is stared and the heating restored.

The combination of both functions - time enable and via alarm input is not possible with one additional relay. Since the time enable/disable is executed once at the set times, subsequent changes in the state of the alarm input will neutralize the time enable. In order to implement such a combination, it is necessary to include another relay output in series with the current ones. One enabler to be controlled by the alarm input and the other by the timers: so a logical AND function will be implemented between the two "enables" (both enablers must be active) and they will not affect each other.

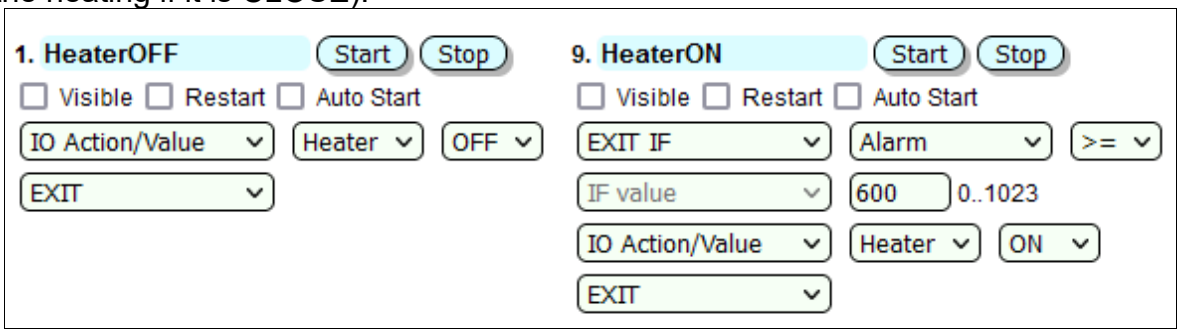
5. Better solution with new macros steps in v5.58

In this version, new steps have been added in the macros - "EXIT IF" and "Skip next step IF". They make it possible to change the execution of the macro depending on the value of the input-output circuits.

As you may have guessed, enabling the thermostat via an alarm input would be possible in this way. Time enabling (from the timers) remains possible only via an additional relay output connected in series.

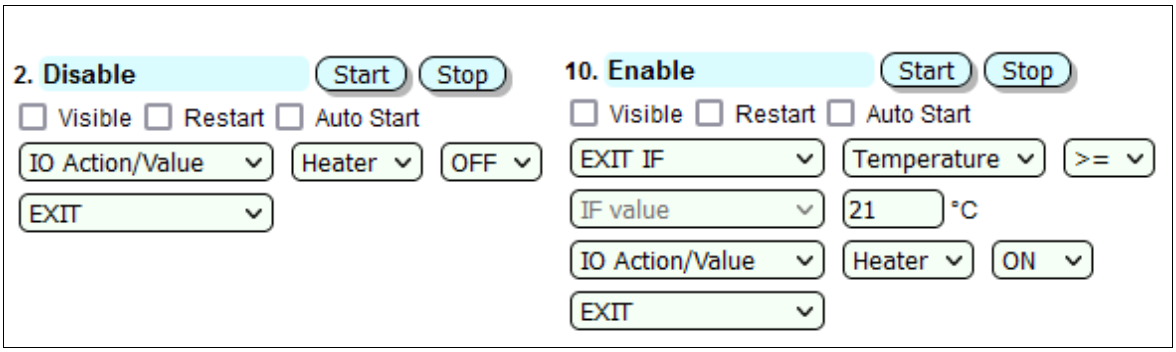
5.1. „Alarm" input resolution without additional relay

We use the Automation settings from the previous section without changes. Macro 1 remains unchanged, but in Macro 9 (which starts when a low temperature is detected and must turn on the heating) we add a check for the Alarm status (we only turn on the heating if it is CLOSE):



If the input is "OPEN" (when the alarm input is open, its value is about 1020, and when it is closed, about 10), the "EXIT IF" step ends the execution of the macro without turning on the heater.

In this way, we blocked the activation of the heating with the "Alarm" input. But what happens next time when "Alarm" takes state "CLOSE" and gives permission for heating? Nothing will happen because the Automation block has already started macro 9 and will not take any further action until the temperature drops below the lower threshold. To make the heating work when the input changes, it is necessary to make new settings in macro 10:



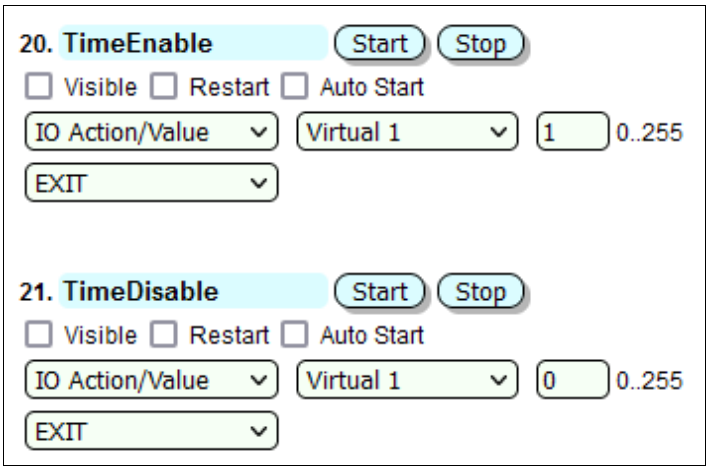
With these new settings, if Alarm becomes "CLOSE", then this causes "Enable" to start, which always turns on the heating if the temperature is <21° at that moment. The rule in this case is that we only modify the macros that are related to the active state of the load (in this case, heater on). Turning it off does not "interest" us in terms of the permission from the alarm input, so these macros remain unchanged.

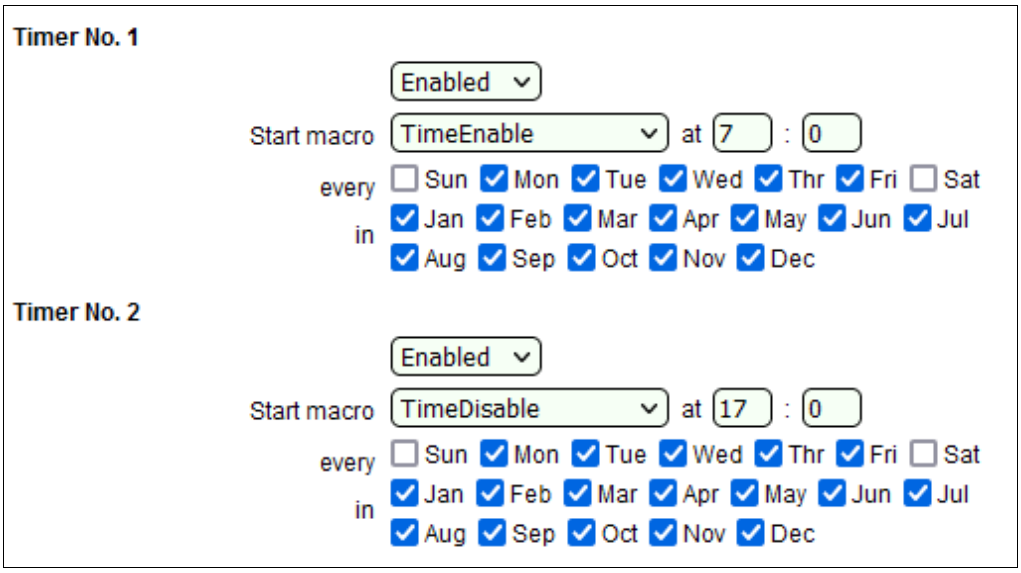
6. Add permission from 'Timers' and 'Virtual IO' from v5.59

Let's go back to the option discussed in the previous section, which allowed us to use an alarm input for permission, without an additional relay output.

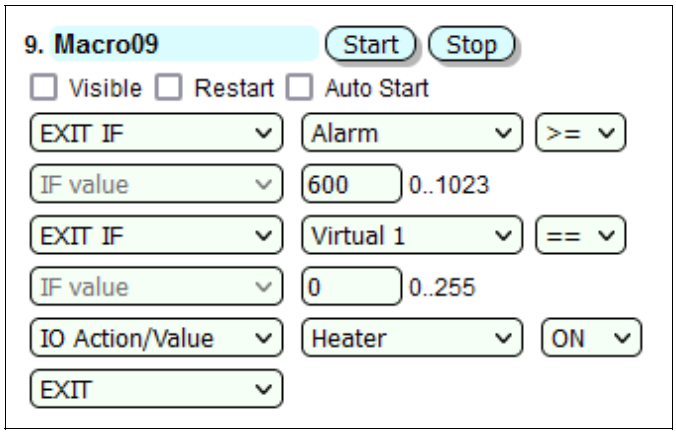
We will extend it with the ability to control the heating also by time (via Timers) again without using a second relay. Its role is taken by one of the 'Virtual IO' channels "Virtual 1".

For this purpose, we define two macros 20 and 21 to set value 1 and 0 to "Virtual 1". And the macros themselves are started by timers 1 and 2, respectively, at 7:00 and 17:00 every weekday (we simulate "working hours").



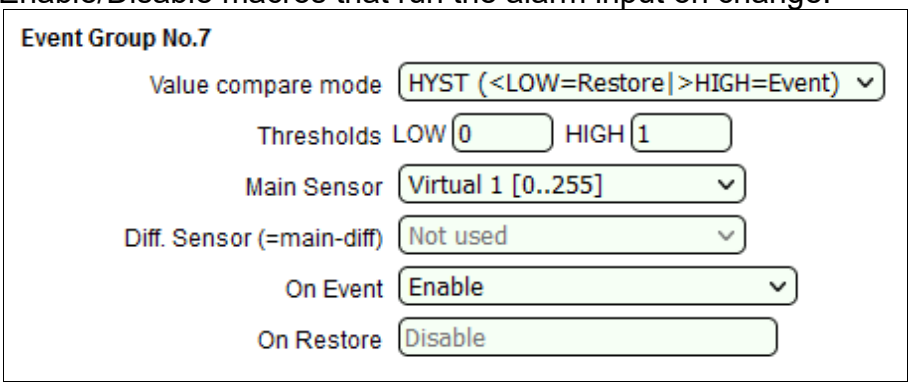


As mentioned in the previous section, we are interested in the macros that lead to an active action - turning on the load. First we need to add in Macro 9. "Heater ON" check if "Virtual 1" is 1 and give permission to run:



Then we also need to solve the problem with the state change to "Virtual 1" occurring after the Temperature Automation has run the corresponding macro for one of its two states.

We need to create a new Automation for the "Virtual 1" channel and have it run the same pair of Enable/Disable macros that run the alarm input on change.



Now at 7:00 "Virtual 1" becomes =1 and this causes the Enable macro to run - it will turn on the heating if the temperature is below the set threshold. At 17:00 via the Disable macro (since "Virtual 1" = 0) the heating is stopped directly.

Remember that comparisons with "Virtual IO" channels use the inequalities ">=HIGH" and "<=LOW".