Niagara IRM
Software License Advisory
This document supports software that is proprietary to Honeywell GmbH, Honeywell Control Systems Ltd. and/or to third party software vendors. Before software delivery, the end user must execute a software license agreement that governs software use. Software license agreement provisions include limiting use of the software to equipment furnished, limiting copying, preserving confidentiality, and prohibiting transfer to a third party. Disclosure, use, or reproduction beyond that permitted in the license agreement is prohibited.

Trademark Information
CentraLine and ‘close to you’ are trademarks of Honeywell Inc.

BACnet and ASHRAE are registered trademarks of American Society of Heating, Refrigerating and Air-Conditioning Engineers. Microsoft and Windows are registered trademarks, and Windows Internet Explorer are trademarks of Microsoft Corporation. Java and other Java-based names are trademarks of Sun Microsystems Inc. and refer to Sun's family of Java-branded technologies. Mozilla and Firefox are trademarks of the Mozilla Foundation. Echelon, LON, LonMark, LonTalk, and LonWorks are registered trademarks of Echelon Corporation.

Tridium, JACE, Niagara Framework, NiagaraNX Framework, Sedona Framework and Vykon are registered trademarks, and Workbench, WorkPlaceAX, and NXSupervisor, are trademarks of Tridium Inc. All other product names and services mentioned in this publication that is known to be trademarks, registered trademarks, or service marks are the property of their respective owners.
# CONTENTS

## VERSIONING

4

## REVISION HISTORY

4

## IRM_H_0001 APPLICATION

5

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unused Input Slots</td>
<td>6</td>
</tr>
<tr>
<td>Hidden Input Slots</td>
<td>6</td>
</tr>
<tr>
<td>Application Name</td>
<td>8</td>
</tr>
<tr>
<td>Application Versioning scheme</td>
<td>9</td>
</tr>
<tr>
<td>Application and Functionblock Version</td>
<td>10</td>
</tr>
<tr>
<td>Firmware Version</td>
<td>11</td>
</tr>
<tr>
<td>Firmware Update</td>
<td>12</td>
</tr>
<tr>
<td>Niagara IRM Engineering Tool Version</td>
<td>13</td>
</tr>
<tr>
<td>Bootloader Version</td>
<td>14</td>
</tr>
<tr>
<td>Setting a Password to the controller and Niagara Tool</td>
<td>16</td>
</tr>
<tr>
<td>Writing a new Application – Starting with the IOs</td>
<td>17</td>
</tr>
<tr>
<td>Copying the application from the palette h0nIRMAppl-rt.jar.sig file into a project</td>
<td>18</td>
</tr>
<tr>
<td>Copying the application from the palette file into a project</td>
<td>19</td>
</tr>
<tr>
<td>Hardware Compatibility Check – Adapt the I/Os</td>
<td>20</td>
</tr>
<tr>
<td>Entering Room Name and Application Name</td>
<td>25</td>
</tr>
<tr>
<td>Saving the Application as a Snapshot</td>
<td>27</td>
</tr>
<tr>
<td>Saving the Application as a Palette File</td>
<td>28</td>
</tr>
<tr>
<td>Short Description</td>
<td>30</td>
</tr>
<tr>
<td>Folder Structure of the Application</td>
<td>40</td>
</tr>
<tr>
<td>Data Flow</td>
<td>41</td>
</tr>
<tr>
<td>Monitoring</td>
<td>42</td>
</tr>
<tr>
<td>Enumeration and Causes</td>
<td>44</td>
</tr>
<tr>
<td>Causes</td>
<td>45</td>
</tr>
<tr>
<td>Connections</td>
<td>46</td>
</tr>
<tr>
<td>Configurations</td>
<td>47</td>
</tr>
<tr>
<td>Folder 0001_Occupancy</td>
<td>48</td>
</tr>
<tr>
<td>Folder 0050_Setpoint</td>
<td>49</td>
</tr>
<tr>
<td>Folder 0100_HvacMd</td>
<td>50</td>
</tr>
<tr>
<td>Selecting the Application HVAC Mode Bits</td>
<td>51</td>
</tr>
<tr>
<td>Folder 0150_Sensors_WmReset</td>
<td>53</td>
</tr>
<tr>
<td>Folder 0200_WM_PID_RmTemp</td>
<td>54</td>
</tr>
<tr>
<td>Folder 0800_Clg01_FCU_Wtr</td>
<td>56</td>
</tr>
<tr>
<td>Selection of Fan Type to be Staged or Variable Fanspeed Fan</td>
<td>58</td>
</tr>
<tr>
<td>Decoupling a FCU Sequence from the Fan</td>
<td>59</td>
</tr>
<tr>
<td>Changing from Relative to Absolute Setpoint</td>
<td>60</td>
</tr>
<tr>
<td>Changing between “C” and “F”</td>
<td>61</td>
</tr>
<tr>
<td>Wired Wallmodule</td>
<td>64</td>
</tr>
<tr>
<td>Hardware Availability</td>
<td>68</td>
</tr>
<tr>
<td>Sensor break/short recognition</td>
<td>69</td>
</tr>
<tr>
<td>Configuration Parameters</td>
<td>74</td>
</tr>
<tr>
<td>BACnet Communication Errors (Life Check, RefIn COV, RefIn Polling)</td>
<td>75</td>
</tr>
<tr>
<td>Changing the Default Value of the BACnet Functionblocks</td>
<td>78</td>
</tr>
<tr>
<td>Setting Out Of Service for BACnet Functionblocks</td>
<td>79</td>
</tr>
<tr>
<td>Reset the Controller to Factory Default</td>
<td>81</td>
</tr>
<tr>
<td>Functionblock Palette</td>
<td>82</td>
</tr>
<tr>
<td>Power Up Behavior / LED behavior</td>
<td>83</td>
</tr>
<tr>
<td>Niagara – Manual and Auto saving / Out of Sync Problem</td>
<td>84</td>
</tr>
<tr>
<td>Trouble Shooting / Technical Support</td>
<td>85</td>
</tr>
</tbody>
</table>
This application guide is applicable to the IRM_H_0001 application.

REVISION HISTORY

07 Jun 2019 - Remove Entering a defect into Jira
07 Jun 2019 - Add another way how to find out the installed Tools Version.
24 Jun 2019 - Added wired wallmodule
15 Nov 2019 - Added Output Type selection
Note:
The application already includes 2x cooling, 2x heating and 1x changeover sequence as usage for
FCU Changeover, Ceiling Cooling, Ceiling Heating, Ceiling Changeover, Radiator Heating or Underfloor Heating system.
Unused Input Slots

Unused Inputs Slots have the value Null. If a connection is removed, then write Null into the Input Slot if this is not done automatically. In below example. In C .. InF is not used (there is no connection) and therefore the value is Null.

<table>
<thead>
<tr>
<th>And1</th>
</tr>
</thead>
<tbody>
<tr>
<td>And</td>
</tr>
<tr>
<td>Execution</td>
</tr>
<tr>
<td>Out</td>
</tr>
<tr>
<td>In A</td>
</tr>
<tr>
<td>In B</td>
</tr>
<tr>
<td>In C</td>
</tr>
<tr>
<td>In D</td>
</tr>
<tr>
<td>In E</td>
</tr>
<tr>
<td>In F</td>
</tr>
</tbody>
</table>

Hidden Input Slots

Some Input Slots are available in addition as a Parameter (In A1 and In A1 Par in below function block). The advantage is that a constant value is entered directly in the function block. This means that fewer function blocks are required and the Wireshet is more clearly represented.

<table>
<thead>
<tr>
<th>BinarySelect</th>
</tr>
</thead>
<tbody>
<tr>
<td>Binary Select</td>
</tr>
<tr>
<td>Execution</td>
</tr>
<tr>
<td>Out</td>
</tr>
<tr>
<td>Select</td>
</tr>
</tbody>
</table>
About the menu “Pin Slots” the input slots can be made visible. Right click on Functionblock.

Note: Do not make any slots invisible because the information is not stored in the controller. When reading back from the device into a new project (Learn from controller), the slots are displayed again.
Description
The Application name starts with IRMN, then there are 1 or more letters describing the application (H=HVAC, L=Light, S=Sunblind). At the end, a unique consecutive application number is appended.

IRMN_H_0001

Application short description
H = HVAC, L=Light, S=Sunblind
Example: HLS includes HVAC, Light and Sunblind.
Example: L includes only Light

Product Family Name
IRMN = with Niagara (Nx) programable IRM controller

Application number
Unique number over ALL IRMN4 applications
Description
The version consists of the following 4 numbers.

2. 0. 0. 0
Build
Bug
Minor
Major

Major: This is a new Application
Minor: The BACnet Interface has changed, i.e. new parameters or a new feature is implemented in application (i.e. wind sensor added)
Bug: The Application has changed for Bug-Fixing without new features. Same BACnet Interface
Build: This is only used for internal testing. After successfully testing, the build number remains unchanged for the release (no relabeling). With every change in Major, Minor or Bug, the build restarts with 0. Not relevant for customer.

Conversion of an existing application to a new firmware

The controller should always operate with the latest firmware. It may happen that an application has been created with an older firmware. Therefore it is planned to provide an update tool to update an existing application to make it running with the latest firmware.
<table>
<thead>
<tr>
<th>Property Sheet</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control Manager (Irm Control Manager)</td>
<td>Markus Latz</td>
</tr>
<tr>
<td>Description</td>
<td>IRMN_H_0001 1.0.1.0</td>
</tr>
<tr>
<td>Application Type</td>
<td>y4t.dbm</td>
</tr>
<tr>
<td>Function Block Family</td>
<td>IrmControl</td>
</tr>
<tr>
<td>Function Block Version</td>
<td>0.7.0.0</td>
</tr>
<tr>
<td>Number Of Folders</td>
<td>21</td>
</tr>
</tbody>
</table>

**Application Version**
- Jira Component = IRM N4 Application
- JiraVersion = IRMN_H_0001_x.x.x.x

**Functionblock Version**
- Jira Component = IRM N4 Tools
- Jira Version = IRMN_FB_Ver_x.x.x.x
<table>
<thead>
<tr>
<th>Name</th>
<th>Ext</th>
<th>Device ID</th>
<th>Status</th>
<th>Netwk</th>
<th>MAC Addr</th>
<th>Vendor</th>
<th>Model</th>
<th>Firmware Rev</th>
<th>App SW Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPO-RL4N</td>
<td></td>
<td>device:1001</td>
<td>[ok]</td>
<td>60000</td>
<td>59</td>
<td>Honeywell</td>
<td>RL4N</td>
<td>0.0.4.3</td>
<td>No application</td>
</tr>
<tr>
<td>CPO-RL6N</td>
<td></td>
<td>device:1000</td>
<td>[ok]</td>
<td>60000</td>
<td>45</td>
<td>Honeywell</td>
<td>RL6N</td>
<td>0.0.4.3</td>
<td>No application</td>
</tr>
</tbody>
</table>

**Firmware Version**

Jira Component = IRM N4 Firmware
Jira Version = IRMN_FW_Ver_x.x.x.x
Firmware Update

Before the firmware update, the application should be switched off via the wall module to avoid unwanted switching on/off of the relays/triacs during the download process.

To ensure a fast downloading, change the Apdu Timeout from 3000msec to 6000 msec. The Apdu Retry is used to set how often a message is to be repeated if the message has not been answered and the Apdu Timeout is used to set the time after which the message is to be sent again.

Apdu Timeout = 6000 ms [0 max]
maxMaster = 32

Before selecting “Download Firmware”, start “Learn from Controller” if the controller is new added to the Database before.

Note: If the R1/T1 LEDs blink very fast during Firmware Download, then the download process will take more than 30min. That can happen, if the “Learning from Controller” is not fully finished before downloading.
The Tools version depends on the station

In **JACE / Eagle Hawk**, the Tools Version is shown on Software Manager.

From local station, select Platform – Open Platform Services. Then Platform->Platform Administration->View Details.

The list includes all modules (.jar-Files) with versions.

Another way to find out the Tools Version is check the **Installed jar-Files**:

Goto the installation path of Niagara\Modules\honIrmConfig-wb.jar – Right click – 7-zip - open archive – META-INF\module.xml

```xml
<?xml version="1.0"?>
<module runtimeProfile="wb" moduleName="honIrmConfig" buildHost="IE3BVWMSLCVM" buildMillis="1571910094331" autoload="true" installable="true" nre="true" preferredSymbol="irmn" description="Programmable Unitary Controller" vendorVersion="1.1.0.6" vendor="Honeywell" bajaVersion="0" name="honIrmConfig-wb"/>
```

The function block version installed in the Niagara IRM Engineering Tool can be read from honIrmControl-rt.jar

```xml
<?xml version="1.0"?>
<module runtimeProfile="rt" moduleName="honIrmControl" buildHost="IE3BLTGJS7TC2" buildMillis="1572412652768" autoload="true" installable="true" nre="true" preferredSymbol="irmn" description="Library of IRM Control Components" vendorVersion="1.0.0.0" vendor="Honeywell" bajaVersion="0" name="honIrmControl-rt"/>
```
Bootloader Version

- CPO-RL4N
- Alarm Source Info
- Points
- Virtual
- Alarms
- Schedules
- Trend Logs
- Config
- Device Object
- IRM Program

| proprietary1026 | 1.0.0.0 |
Setting the Time to all Controllers

The alarms will then use the right date and time stamp.

Functionblocks

Currently Functionblock do not support the time. This will be implemented still.
After creating the controller in the database it is recommended to assign a Niagara and controller password before programming.

**Right Click on IRM Program**

**Select Actions – Set Controller Password**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>New Password</td>
<td>Enter the new password</td>
</tr>
<tr>
<td>Repeat New Password</td>
<td>Enter the new password again for verification</td>
</tr>
<tr>
<td>Set Password for</td>
<td>Select the password type (Tool and Controller)</td>
</tr>
</tbody>
</table>

**Password requirements:**
- Minimum length of password is 10
- Minimum number of lower case characters is 1
- Minimum number of upper case characters is 1
- Minimum number of digits is 1

Although the Tool-Password is entered under IRM Program, it is not stored in the IRM Program folder but in the device. The existing IRM program folder can be deleted and a new IRM program can be copied from a palette without entering the Tool-Password again. The Tool-Password is only lost when the device is deleted. Also Clear Project and Learn From Controller can be done.

The Database view shows

**Password Status**
- Default Password
- User Password
Writing a new Application – Starting with the IOs

Learn From Controller

After adding the controller to the database, the **On board IO** wiresheet is empty.

After right mouse click on the IRM Program, select Actions – Learn From Controller. Then the **On board IO** wiresheet with the right functionblocks are loaded.
Copying the application from the palette honIrmAppl-rt.jar.sig file into a project

1. Copy the application palette files honIrmAppl.jar and honIrmAppl-rt.jar.sig into the Niagara\modules installation directory if this has not already been done with the other .jar files. Then start the station.

2. In Niagara open the Palette honIrmAppl

   Enter honIrmAppl and press OK.

3. Drag and Drop IRM Program to the controller
Copying the application from the palette file into a project

1. Copy the palette file to any folder on your PC.
2. Open “My PC, “My Filesystem” and search the palette file
3. Select the palette file with the right Mouse click and select “Copy”
4. Then go to the controller to the IRM Program entry
5. With right mouse click on IRM Program and delete the existing IRM Program.

Note: Your logic on the wire sheet will be deleted then.

6. Select the controller and with right mouse click select “Paste”. Then a new IRM Program is inserted with the application from the palette.

In Future: For managing the applications, create “honIrmAppl” module which includes palette file. This makes it easy to manage the delivery of applications as part of setup.
Since the controllers are available in different I/O variants, the application needs to be adapted to the specific I/O variant. Connections to I/Os that are missing on the new hardware should be deleted.

Before Teaching the Application to the controller, make a “Check Hw Compatibility”. I/Os that are not available on the specific hardware are output as messages. The deletion does not take place automatically.

The deletion takes place on the On Board IO. Please note that existing connections will be lost due to deleting. Therefore, it is advisable to first adapt the connections to/from the application to the new hardware, and in the next step to delete the non-existing terminals. This is described on the next page.

Note: In general, the system should be switched off before teaching, e.g. via the wall module, in order to avoid undesirable side effects (e.g. when using an electric heater that requires a fan).

Note: When programming a sunblind application, the blind motor should only be connected when the application is ready so that the drive is not damaged during programming.
Note: Before deleting the I/Os, adjust the connections.

Suppose UI5 (Universal Input 5) is not present on the specific hardware, but the window contact is to be read by the application from another UI.

A double click on the connection takes you to the Wiresheet where the connection comes from.
The connection comes from Wireshell 0150_Sensors_WmReset.

The connection can be deleted with the right mouse button.

The new connection to another UI is then prepared with the right mouse button on Link Mark.
Now go to On board IO and select the new UI.

Select **Link To...** for hardware inputs and **Link From...** for hardware outputs.
Select Out and In

Then Rename the Function block accordingly, enter the parameters (characteristic...) and the right Annotation (compare with the old UI5).

Delete the old not supported IO Functionblock
Entering Room Name and Application Name

The **Object Name** is the name of the Controller shown on BACnet network in 3rd party or in the Honeywell SymmetrE building management system. The name should be unique in the building and the name should be the same as the Controller name in tree on the left side.

Enter a **location** where the controller is installed.

Enter a short **description** about the application.
Enter your name and the application description here.
Saving the Application as a Snapshot

Taks Snapshot makes a backup of the application to a local file. With Restore Snapshot, the backup can be loaded.
Saving the Application as a Palette File

Copy the IRM Program.

Then select a folder in your PC (C:\01_IRM in below example). In this folder, a new Palette File will be created.
The Paste the IRM Program into the Palette File on your PC.

Select OK

Note: Please select “Save” and “Close”.

First select “Save”

Then select “Close”
**FCU Application**

The Application IRMN_H_0001 is a FCU application consisting of a cooling sequence, a heating sequence and an electric heater.

The application is already implemented with 2 cooling and 2 heating sequences. A changeover sequence can be used instead of cooling sequence 1 and heating sequence 1. The free programming makes it easy to create a radiator or chilled ceiling application by decoupling the fan from the corresponding sequence.

**Sensors**

The application already supports a motion detector, a changeover temperature sensor measuring the pipe temperature, a window contact and an air flow contact to check the fan.
Frost and Overheat Protection

There are 2 frost and 2 overheat conditions implemented, which can both be evaluated in the cooling and heating outputs. Sequence 1 can evaluate the primary and sequence 2 the secondary condition or the sequence uses different levels for the primary and secondary condition.

HVAC Mode

The HVAC mode (Heating, Ventilation and Air Conditioning) covers various scenarios:

- Application HVAC Mode: This is used to configure the application to determine which equipment is available in the room. Example 1: The room is equipped with cooling via water and an additional electric heater. Example 2: The room is equipped with a 2-pipe changeover system for cooling and heating.

- Plant HVAC mode: The plant controller sends the information whether the cooling or heating primary system or both are switched on. Only when, for example, the primary cooling system is switched on, the controller will determine a cooling setpoint if the room temperature is too warm. In summer, the Plant controller should send 'cooling' when the outside temperature is high, in winter 'heating' when the outside temperature is cold and 'Cooling+Heating” during the transition period.

- Changeover Water Medium: The Plant controller sends the information which medium in the supply line for the 2-Pipe Changeover System is provided by the Plant controller, either Off, cooling or heating water. The water temperature can also be measured and evaluated by the room controller itself if the sensor is present.

- HVAC Mode Operator: The wall module operator can use the Sylk wall module to set whether he only wants heating or cooling or both. In summer, for example, he can say that he never wants to heat. This prevents heating in summer, for example, when the window is open in the morning and there is no window contact and the outside temperature is still quite fresh, e.g. 14°C. On the other hand, in summer the operator can select "Heating" to deactivate cooling, as cold air can be very unpleasant.

The application is already prepared in such a way that the Plant controller can also select between cooling with water or electrical cooling (same for heating). Thus, the most cost-effective method for cooling and heating can be determined by the Plant controller at the present time.
Occupancy Mode

The energy savings and comfort depend essentially on the Occupancy Mode. The following Occupancy Modes are available:

- Occupied: The room is occupied. In the office, for example, 8:00-17:00 a clock.
- Standby: The room is not occupied for a short time. The Occupancy Mode can go into standby via the motion detector in case of non-detection.
- Unoccupied: The room is unoccupied, for example during night and at the weekend.
- Holiday: The room is unoccupied for a longer time.
- Bypass: Temporary Occupied. The operator can temporarily set the Occupancy Mode to Bypass via the wall module.

The effective Occupancy Mode depends on:

- Time scheduler
- Occupancy sensor
- Occupancy override from the wall module operator

Example 1

The wallmodule is TR42. An occupancy sensor is not installed. The Occupancy Mode from the scheduler is “Standby”. The operator can override the Occupancy Mode for 2 hours into the Bypass Mode (Temporary Occupied) from the Wall Module TR42.
Out Eff Occ Md = Stby (-> 25°C Clg Setpt)

Out Occ Md Ovrd *1 = null (no ovrd)

Out Occ Md Ovrd Rem Time *1 = 0 (no timer running)

*1 Goto Views – AX Slot Sheet and select this points with the right mouse and select Config Flags. Then Select Summary and Deselect Hidden.
Now the user has selected Occupancy Override Mode
“Bypass” on the wall panel.

Out Eff Occ Md Ovrd = Bypass (\(\geq 23^\circ C\) Clg Setpt)

Out Occ Md Ovrd \(^*1\) = Bypass

Out Occ Md Ovrd Rem Time \(^*1\) = 2 (countdown timer is running, 2min pending)

If the timer is expired, then the EffOccMd is again Stby.

\(^*1\) Goto Views – AX Slot Sheet and select this points with the right mouse and select Config Flags. Then
Select Summary and Deselect Hidden.
The room temperature setpoint is determined by the Occupancy Mode. For each Occupancy Mode there is a cooling and a heating setpoint.

If the room temperature is above the cooling setpoint, the cooling setpoint is controlled and the setpoint mode is then "Cooling". If the room temperature is below the heating setpoint, then the heating setpoint is controlled and the setpoint mode is then "Heating".

If the room temperature is between Cooling and Heating setpoint, in the so-called Zero Energy Zone, the setpoint and mode remain the same. In the Zero Energy Zone, however, cooling or heating continues in principle. Only when the room temperature with closed cooling and heating outputs is between the cooling and heating setpoint then no more energy is consumed.

The operator can set a relative setpoint or an absolute setpoint via the wall module. The range can be limited depending on the Occupancy Mode. The relative setpoint is added to the base setpoint (one of the 6 setpoints). The absolute setpoint is based on the middle between the cooling and heating setpoints.

Before changing the setpoint mode, a stabilization time is started to ensure that the measured room temperature is stable above or below the new setpoint. If the change has taken place, another time is started during which the Setpoint mode is set to OFF. All cooling and heating outputs are closed or switched off.

This time should be considerably longer than the running time of the valves. This ensures, for example, that a cooling valve closes before a heating valve opens. Especially with a FCU, a lot of energy can still be stored in the water register.

**Room PID control**

The PID controller is used to control the room temperature to the setpoint. Depending on the setpoint mode (cooling, heating, off), the PID controller is operated in direct or inverse mode. The result is a controlled value of 0..100%.

**Cooling and Heating Sequences**

The cooling sequences take the PID output as base value if the setpoint mode is COOLING. For the heating sequence according to HEATING.

If several cooling or several heating sequences are present, it often makes sense not to control both at the same time, but first one sequence and then the other (0..50%, 50..100%). If the window is open, the level can be set (e.g. 0%). Next there are some queries which correct the level to 0%:

- The HVAC mode for this sequence is not enabled (no cooling comes from the plant controller).
- The setpoint mode is OFF.
- The setpoint mode does not match the sequence (cooling sequence requires setpoint mode cooling)
- The device was switched off by the fan speed selection of the wall module.
Then, in the Frost and Overheat cases, a level is entered to protect the registers or the room. The cooling sequences send a request to the sun protection system so that the blinds are closed when the sun shines. The same happens with the fan. The fan is switched on via the request and if there is feedback that the fan is running, the sequence will wait a while to 0% before opening. As the last step, the sequence can also be deactivated via BACnet. Now that all queries have been completed, the sequence is given a value of 0..100% and a cause from which you can read which of the many conditions has applied. The cause list can be found further down in the "Monitor" chapter.

**Fan control**

Following Fan types are supported:

- 1-stage fan
- 2-stage fan
- 3-stage fan
- Variable speed fan

In principle, the fan level depends on the levels of the cooling and heating outputs. The higher the cooling and heating levels, the higher the fan level. The cooling and heating outputs are opened or switched on if the fan has been switched on for a while. If the cooling and heating outputs have been closed or switched off, the fan will still run for a while. It is guaranteed that the fan always remains on when electrical outputs are switched on, independent of the MinOn/Off times for Staged Output and independent of the PWM period for PWM output. If the optional Air Flow Contact does not detect airflow, the cooling and heating outputs are turned off. However, the fan remains activated.

The following fan setting options are available:

- The fan characteristic is freely adjustable
- The fan level can be raised to a min level depending on the Occupancy Mode, e.g. min level 1 for Occupied.
- The fan level can be limited to a max level depending on the Occupancy Mode, e.g. max level 2 for Occupied.
- The fan is switched off when the window is open and there is no frost protection.
- The fan can be switched to any position via the wall-mounted module.
- The fan runs for an adjustable time after there is no request via the cooling and heating outputs.

**Terminal assignment**

The application is supplied with a preset terminal assignment that matches with an RL4N, RL5N, RL6N hardware variant. This can of course be adjusted. Also the outputs are preset with certain actuators (0..10V, Floating, PWM, 3-Staged). This serves as an example of how the programming for the individual actuators looks like.
In principle, the regulated value from the PID controller 0..100% is connected to **one** of the following output function blocks depending on your installation (see below picture):

- Floating for 3-Point Actuator (2 outputs for open and close). The outputs are defined in the function block to minimize possible runtime deviations. Therefore, no connection to the OnBoardIO is necessary.

- PWM for Thermal Actuator or Electrical Cooling or Heating

- Stg123Outp for staged Cooling or Heating

- For the Analog Output, the 0..100% value can immediately connected with to the OnBoardIO – AoTerminal functionblock.
Folder Structure of the Application

Each folder occupies memory, even if there is no logic in the Wireshell. Per Wireshell it is recommended to use only approx. 100 Funktionblocks. In addition, there is a limitation in the width of the wireshell.

Therefore, the application is structured in different folders as shown on the left side. Each folder usually contains some function blocks with configurable parameters, which can be adapted to the particular customer application. The names of the function blocks with adjustable configuration parameters start with Cfg followed by a 4-digit number. The 4-digit number matches the folder number.

Example: The folder 0050_Setpoint includes functionblocks with following names

The Excel File ‘IRMN_H_0001_Interface” includes the configurations.

The function blocks in the folders are arranged in such a way that the Action “Reorder - Sort by Position” can be executed at any time. If new function blocks are added, the order from left to right should be kept and after the change the Reorder-Action should be executed.
Data Flow

The data flows from the top Folder ‘0001_Occupancy’ to the lowest folder ‘1050_FanOutputs’ and repeats with the DDC cycle of 500msec in the Periodic program folder. The folders are arranged from left to right, that means in a way that the action ”Reorder - Sort by Position” can be executed at any time.

The function blocks are processed in the order of the execution number. The Logic data flow should run from left to right.

If a new function block is added to an existing logic, the new Functionblock is processed as the last one, because the last function block has the highest execution number.

Therefore, a ”Reorder - Sort by Position” must then be executed.

If the reorder is not executed, strange effects may occur, e.g. values / relays / triacs for a DDC cycle get wrong values.
There is a small overview page implemented showing the most important points for the inputs, the setpoint and HVAC mode calculation and the outputs including the output causes.
A very important feature is that the application for the sequences and the fan display the Cause for the current position. This makes the application very easy to understand.
# Enumeration and Causes

### HVAC

<table>
<thead>
<tr>
<th>Enum</th>
<th>Fan causes</th>
<th>Sequence causes</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>NoCtgHtgDemand</td>
<td>AutoHtg (From PID)</td>
</tr>
<tr>
<td>03</td>
<td>AutoCtg (From PID)</td>
<td>AutoCtg (From PID)</td>
</tr>
<tr>
<td>06</td>
<td>AutoHtg (From PID)</td>
<td>AutoCtg (From PID)</td>
</tr>
<tr>
<td>09</td>
<td>FanOnForCtg01</td>
<td>FanOnForCtg01 (From PID)</td>
</tr>
<tr>
<td>12</td>
<td>FanOnForCtg02</td>
<td>FanOnForCtg02 (From PID)</td>
</tr>
<tr>
<td>15</td>
<td>FanOnForHtg01</td>
<td>FanOnForHtg01 (From PID)</td>
</tr>
<tr>
<td>18</td>
<td>FanOnForHtg02</td>
<td>FanOnForHtg02 (From PID)</td>
</tr>
<tr>
<td>21</td>
<td>PowerUp</td>
<td>PowerUp</td>
</tr>
<tr>
<td>24</td>
<td>OccMdNeedsMinSpd</td>
<td>OccMdNeedsMinSpd</td>
</tr>
<tr>
<td>27</td>
<td>OccMdPreventsMaxSpd</td>
<td>OccMdPreventsMaxSpd</td>
</tr>
<tr>
<td>30</td>
<td>WindowOpen</td>
<td>WindowOpen</td>
</tr>
<tr>
<td>33</td>
<td>NightPurge</td>
<td>NightPurge</td>
</tr>
<tr>
<td>36</td>
<td>OverhtPrim</td>
<td>OverhtPrim</td>
</tr>
<tr>
<td>39</td>
<td>OverhtSec</td>
<td>OverhtSec</td>
</tr>
<tr>
<td>42</td>
<td>FrostPrim</td>
<td>FrostPrim</td>
</tr>
<tr>
<td>45</td>
<td>FrostSec</td>
<td>FrostSec</td>
</tr>
<tr>
<td>48</td>
<td>DeviceOff (Wallmod)</td>
<td>DeviceOff</td>
</tr>
<tr>
<td>51</td>
<td>FanSelection (Wallmod)</td>
<td>FanSelection (Wallmod)</td>
</tr>
<tr>
<td>54</td>
<td>DripPan</td>
<td>DripPan</td>
</tr>
<tr>
<td>57</td>
<td>CondSwitch</td>
<td>CondSwitch</td>
</tr>
<tr>
<td>60</td>
<td>Dewpoint</td>
<td>Dewpoint</td>
</tr>
<tr>
<td>63</td>
<td>SwitchOver</td>
<td>SwitchOver</td>
</tr>
<tr>
<td>66</td>
<td>Fire</td>
<td>Fire</td>
</tr>
<tr>
<td>69</td>
<td>AirFlowFail</td>
<td>AirFlowFail</td>
</tr>
<tr>
<td>72</td>
<td>SetptMdOff</td>
<td>SetptMdOff (Htg or Clg, HVAC Mode Change)</td>
</tr>
<tr>
<td>75</td>
<td>SetptMdCtg</td>
<td>SetptMdCtg</td>
</tr>
<tr>
<td>78</td>
<td>SetptMdHtg</td>
<td>SetptMdHtg</td>
</tr>
<tr>
<td>81</td>
<td>HvacMdNotCtg1</td>
<td>HvacMdNotCtg1</td>
</tr>
<tr>
<td>84</td>
<td>HvacMdNotCtg2</td>
<td>HvacMdNotCtg2</td>
</tr>
<tr>
<td>86</td>
<td>SetptMdCtg</td>
<td>SetptMdCtg</td>
</tr>
<tr>
<td>87</td>
<td>HvacMdNotHtg1</td>
<td>HvacMdNotHtg1</td>
</tr>
<tr>
<td>90</td>
<td>HvacMdNotHtg2</td>
<td>HvacMdNotHtg2</td>
</tr>
<tr>
<td>93</td>
<td>FanOnDelayRuns</td>
<td>FanOnDelayRuns</td>
</tr>
<tr>
<td>94</td>
<td>FanOverrun</td>
<td>FanOverrun</td>
</tr>
<tr>
<td>96</td>
<td>MasterControlled</td>
<td>MasterControlled</td>
</tr>
<tr>
<td>97</td>
<td>WaitingForSlblndDn</td>
<td>WaitingForSlblndDn</td>
</tr>
<tr>
<td>101</td>
<td>FanOnAfterPowerUp</td>
<td>FanOnAfterPowerUp</td>
</tr>
<tr>
<td>103</td>
<td>FanOnForSeqOutputs</td>
<td>FanOnForSeqOutputs</td>
</tr>
<tr>
<td>105</td>
<td>OutOfService (From Service)</td>
<td>OutOfService (From Service)</td>
</tr>
</tbody>
</table>

### Window

<table>
<thead>
<tr>
<th>Enum</th>
<th>Blinds Commands</th>
<th>Emergency Cmd</th>
<th>Manual Ovrd Cmd</th>
<th>Application Cmd</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Stop</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>ToPositionAndAngle</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>StatsHorizontal</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>ToFullyOpenPosition</td>
<td>5</td>
<td>NA</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>ToFullyClosePosition</td>
<td>6</td>
<td>NA</td>
<td>6</td>
</tr>
<tr>
<td>NA</td>
<td>ManOpen</td>
<td>2</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>NA</td>
<td>ManClose</td>
<td>3</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>NA</td>
<td>ManAngleStepPositive</td>
<td>8</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>NA</td>
<td>ManAngleStepNegative</td>
<td>9</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>
Causes

Shown on first Wiresheet

Ctg Level = 0..100%

Ctg 01 Cause = 3 means Cooling Auto
**Example:** Connection from Pid -> Clg01

Connection from the Folder Pid

Connection to the Folder Clg01
Here we see a Functionblock starting with the name Cfg0903… including a Configuration Parameter. This Functionblock is located in the fold 0900… In this folder there could be other configurations until Cfg0949 because the next folder starts with 950. The numer is unique and will never change.

There is an excel sheet available showing the Configurations.
The plant controller has a time program which is communicated to the IRM room controller via the BACnet point `ExtOccSchedule` (Occ, Stby, Unocc). The time program serves as the basis for the effective Occupancy Mode `EffOccMd`. The point is connected to the wall module function block. An internal Scheduler is currently not supported.

If an occupancy sensor is connected to the UI terminal, `Cfg0002_OccSensorAvailable` must be set to true. After conversion to the F states (1=NotUsed, 2=Occupied, 5=Unoccupied) the motion detector is sent to BACnet. If the Out of Service Property of the BACnet Object `HwOccSensor` is set to true, then the Present Value can be overwritten over BACnet to any value (1=NotUsed, 2=Occupied, 5=Unoccupied).

The `HwOccSensor` is or'd with an external occupancy sensor coming over BACnet as `ExtOccSensor`. The result is the Effective Occupancy Sensor `EffOccSensor`, which is also available on BACnet. If the Out of Service Property of the BACnet Object `EffOccSensor` is set to true, then the Present Value can be overwritten over BACnet to any value (1=NotUsed, 2=Occupied, 5=Unoccupied).

In addition, there is a BACnet point `ExtWmOccCmd`. This is used to command the occupancy mode from an external wallmodule.

The Occupancy Mode logic is part of the wallmodule functionblock `WmConfigHvacA`. The entire Occupancy Logic is handled here, taking into account the manual override of the wallmodule operator. The result is the effective occupancy mode `EffOccMd`, the occupancy mode `OccWmOvrd` selected by the operator and a running timer `OccWmOvrdTimer`, if the operator has selected an Occupancy mode for a certain time only. All 3 outputs are available on BACnet.
The setpoint is calculated in the wall module function block WmConfigHvacA based on 6 base setpoints. In addition, the operator can select either a relative or an absolute setpoint via the wall module.

The 6 basic setpoints, which are available as parameters via BACnet (cooling/heating for Occupied, Standby, Unocc) go into the wall modules function block (implemented in 0200_WM_PID_RmTemp). The setpoint which can be selected by the operator is limited in the range according to the Occupancy Mode, see MinClgSetptSelection...MaxHtgSetptSelection. Default is -5...+5 °C in Occupied, Standby and Bypass Mode; 0°C in Off, Unoccupied and Holiday.

The Bacnet Point HwRmSetpt is used with a conventional wired wallmodule. Over Cfg0060_HwAiRmSetptRelAbsCelsFahrenht it can be selected how the wired setpoint is evaluated, either as relative or absolute in °C or °F.

The user selected setpoint is powerresistant.

TmpEffSpTemp [0002] [22°C]

MinClgSetptSelection [0023]
MaxClgSetptSelection [0024]
MinHtgSetptSelection [0025]
MaxHtgSetptSelection [0026]

MinMaxSetptSelection

EffOccMd

Occ Middle Setpoint between Clg and Htg Setpts, 22 °C,
-50..150 °C,
-9..9 °F or 55..85 °F)

UI[0102]BEV

EffRmTempSetptMd
3=Off (Off Timer running), 4=Clg, 5=Htg

UI[02]

HwAiRmSetpt
0..100%

For Occ/Off/Hol/Unocc/Stby/Byp

Cfg0060_HwAiRmSetptRelAbsCelsFahrenht
1=RelCelsius, 2=AbsCelsius, 3=RelFahrenh, 4=AbsFahrenh
The Cooling/Heating application installed in the room is set via Cfg0100_AppHvacMdBits (see chapter "Short Description – Hvac Mode"). Cfg0101_Wtr2Pipe should be set to true for a 2-pipe changeover application.

For the 2-pipe changeover application, the water medium must be known, whether cold or warm water is available. Otherwise the changeover output remains at 0%. The temperature is either measured in the room at the supply pipe over HwCngOvrWtrTemp or the controller receives the temperature over ExtCngOvrWtrTemp or the Cooling/heating mode ExtCngOvrWtrMedium via BACnet.

The BACnet point ExtPlantHvacMd informs the controller whether the cooling system or the heating or both are switched on. This tells the room controller whether it can go into cooling or heating mode.

Via the BACnet point HvacMdExternalWmBits the controller receives the information from an external wall module whether the operator wants both cooling and heating as needed or whether he just wants cooling or just heating.

The data are all evaluated in the wall module function block. The WallModule operator can also choose between cooling, heating and auto. The effective HVAC mode EffHvacMdBits is output as a bit value and converted into an enum, which is then sent via BACnet over EffHvacMd. EffHvacMdBits is evaluated by the cooling/heating sequences. If the mode does not match the sequence, the sequence receives the value 0%.
Selecting the Application HVAC Mode Bits

Water Clg + Water Htg + E-Htg = Application 769

The application very generally implemented to cover many different cooling/heating applications. A numeric value "AppHvacMdBits" tells the application that a cooling with water, a heating with water and additionally an electric heating is installed.

The following example shows how to calculate the value for this application. Since this application is a water cooling (1) + water heating (256) + electric heating (512), the sum is 1+256+512=769.

A pure changeover application for cooling and heating would give the value water cooling (1) + water heating (256) = 257 as AppHvacMdBits. In addition, Wtr2Pipe should have the value True.

The following figure shows the corresponding values.
--- High Byte for Heating (256..65280) ---

<table>
<thead>
<tr>
<th>32768</th>
<th>16384</th>
<th>8192</th>
<th>4096</th>
<th>2048</th>
<th>1024</th>
<th>512</th>
<th>256</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Simply build the sum of the values to define your application.

--- Low Byte for Cooling (1..255) ---

<table>
<thead>
<tr>
<th>128</th>
<th>64</th>
<th>32</th>
<th>16</th>
<th>8</th>
<th>4</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Htg Water and Clg Water can also be used for a change over application. The changeover application has the same Application value.

**Example Applications**

There is an E-Heating + Cooling only over Fan installed: CfgAppHvacMdBits = 512 + 4 = 516. CfgWtr2Pipe = False.

There is an E-Heating + Water Heating + Cooling only over Fan installed: CfgAppHvacMdBits = 512 + 256 + 4 = 772. CfgWtr2Pipe = False.

There is a Changeover Heating/Cooling installed: CfgAppHvacMdBits = 256 + 1 = 257. CfgWtr2Pipe = True.

There is a Changeover Heating/Cooling + Electric Cooling installed: CfgAppHvacMdBits = 256 + 1 + 2 = 259. CfgWtr2Pipe = True.

There is a E-Htg + Water Htg + DX-Clg + Water Clg installed: CfgAppHvacMdBits = 512 + 256 + 2 + 1 = 771. CfgWtr2Pipe = False.
This Wireshet is used to read the sensors from a conventional wall module such as room temperature, room humidity, air quality (CO2). Whenever a sensor is not configured (because a Sylk- or external wall module is used), then the points HwRmTemp, HwRmHum and HwRmCO2 get a Null value. HwRmTemp, HwRmHum and HwRmCO2 are send to BACnet.

The effective room temperature EffRmTemp, effective room humidity EffRmHum and the effective air quality EffRmCO2, which come out of the FB wall module in the folder 0200_WM_PID_RmTemp, are sent to BACnet and distributed to the application. If no sensors are present and also the Sylk wall module is not connected or has a communication error, then an EffRmTemp = 8°C is set (can also be changed to Null) to ensure frost protection; EffRmHum and EffRmCO2 are set to Null to inform the further Logic that no sensor is present.

The binary states for the frost like RmTempFrostPrim, RmTempFrostSec and the overheat protection like RmTempOvrhtPrim, RmTempOvrhtSec are derived from the effective room temperature (as output from the wall module). Room Frost and Overheat is used as input for the wallmodule functionblock for ignoring the user selected hvac mode and fanspeed.

The Window contact is read from the terminal and send to BACnet as HwWindow for displaying or for evaluation on a Master-Controller. The wired window contact is evaluated with a BACnet sensor contact. As soon as one of the contacts reports "Window open", the effective window contact EffWindow will be "Window Open".

The wired Air Flow contact is read in. If the contact reports "Air Flow", a stabilization time Cfg0190_AirFlowStableTimeOnDelay is started. After the time has expired, the "Air Flow" is considered to be constant. Before the "Air Flow" changes to "No Air Flow", the time Cfg0191_AirFlowUnstableTimeOffDelay must have elapsed. EffAirFlow is sent on BACnet and used for the Sequence and fan logic.

ExtWmReset is received by the BACnet and converted into bits before it is connected to the wall module. ExtWmReset is used to reset the wallmodule.
The wall module function block WmConfigHvacA is necessary in every HVAC room application to determine the effective values of Occupancy Mode, Setpoint, Setpoint Mode and HVAC Mode. Additionally the Fanspeed Selection and the sensors are being evaluated. All these values depend on each other. All values can be determined or partly changed via the Sylk wall module, the wired wall module or via an external wall module. The entire dependency takes place in the wall module.

There are 6 base setpoints Cfg0200_RmTempSetptClgUnocc, Cfg0201_RmTempSetptClgStby, Cfg0202_RmTempSetptClgOcc, Cfg0203_RmTempSetptHtgOcc, Cfg0204_RmTempSetptHtgStby, Cfg0205_RmTempSetptHtgUnocc which can be written over BACnet. Note that these values are saved into the flash memory of the controller and therefore they may not be overwritten very often (not periodically!).

On the output side of the wallmodule, there are the effective sensor values which are also available on BACnet like EffRmTemp, EffRmHumidity, EffOccMd, EffRmTempSetpt, EffRmTempSetptMd, EffRmTempSetptMidOcc, EffFanSelStg, EffFacSelStg. Very often is used the EffFanSelSpdOff. The EffHvacMdBits is converted into an enum EffHvacMd which is send on BACnet.

The EffRmTemp, EffRmTempSetpt, EffRmTempSetptMd are used as an input for the PID function block which controls the room temperature. The corresponding Cfg0220_PidRmTempClgXp, Cfg0221_PidRmTempClgTi, Cfg0222_PidRmTempClgTd, Cfg0223_PidRmTempHtgXp, Cfg0224_PidRmHtgTi, Cfg0225_PidRmTempHtgTd can be changed over BACnet. Note that these parameters are saved in the Flash memory and therefore they may not be written periodically over BACnet. Otherwise the flash memory will be damaged.
If the EffRmTempSetptMd has the value Cooling, then the control output 0..100% is used as basis as cooling level. If several cooling sequences are present, it is often recommended not to control both sequences in parallel with the same level, but to first fully open one sequence before the other sequence starts opening (0..50%, 50..100%). The Clg01Cause is assigned the value 3, which stands for the automatic mode, which means that the level comes directly from the PID.

The next step is to check whether the window is open. If the EffWindow says “Window Open”, the level is corrected to 0% (changeable) and the Clg01Cause is changed to 30.

The next step is to check whether the HVAC mode applies to exactly this sequence. Since this sequence is to represent a cooling sequence with water, it is checked whether the bit for cooling with water is set in the EffHvacMdBits. If the bit is not set (e.g. because the Plant controller says that the primary cooling system is switched off or the operator did not want cooling via the wall module), the level is set to 0% and the Clg01Cause is set to 81. If the sequence is not to be used for water, but for electrical cooling, only another bit must be checked. Therefore the Cfg0802_Clg01EffHvacMdBits needs to be changed.

The next step is to check whether the setpoint mode EffRmTempSetptMd is set to Off. This happens with every change between cooling and heating and vice versa or when the effective HVAC mode EffHvacMdBits is set to 0, which means it can’t be heated or cooled. During a cooling/heating change there is an adjustable time (WmConfigHvacA.Setpoint.SetptOffTimePar) during which the OFF mode is valid. As a result, all valves close and there is no mixing between cold and hot water.

If the window is open, the level can be set (e.g. 0%). Next there are some queries which correct the level to 0%:

- The HVAC mode for this sequence is not enabled (no cooling comes from the plant controller).
- The setpoint mode is OFF.
- The setpoint mode does not match the sequence (cooling sequence requires setpoint mode cooling)
- The device was switched off by the fan speed selection of the wall module.

Then, in the Frost and Overheat cases, a level is entered to protect the registers or the room. The cooling sequences send a request to the sun protection system so that the blinds are closed when the sun shines. The same happens with the fan. The fan is switched on via the request and if there is feedback that the fan is running, the sequence will wait a while to 0% before opening. As the last step, the sequence can also be deactivated via BACnet. Now that all queries have been completed, the sequence is given a value of 0..100% and a cause from which you can read which of the many conditions has applied. The cause list can be found further down in the "Monitor" chapter.
NightPurge
Night Purge is active
DeviceOff
The device is switched OFF via the wall module.
OverhtSec 100%
The RmTemp has exceeded the sec overheat limit.
See Note 1.
DripPan
Drip pan is full

Dewpoint
The cold water temperature is below the calculated dew point.

FrostPrim
The room temp has fallen below the first frost limit.

OverhtPrim 100%
The RmTemp has exceeded the first overheat limit.
See Note 1.

FanOnDelayRuns
0% if Fan ON Delay is running

WaitingForFan 0%
until the fan is running

WaitingForSblndDn 0%
until the sunblind is down during brightness in htg period

Clg01_Cause
EffRmTempSetptMd 3=Off, 4=Clg, 5=Htg

Note 1: Independent of the EffHvacMdBits, the output opens and the fan is started with the Stg/Spd for this level (i.e., 100%). This is the safest behavior in case of ExtPltMd is wrong or the user has deactivated the sequence or Clg/Htg via the wallmodule.
Selection of Fan Type to be Staged or Variable Fanspeed Fan

The interface description contains a column "Stg/Spd Fan". All configurations marked with "Fan" are set for a 3-stage fan. These configurations have to be adapted for a 1- or 2-speed fan or for a variable fan. Using the “View - AX-Property Sheet” of a folder makes it easy to find the right function blocks.

<table>
<thead>
<tr>
<th>Folder</th>
<th>Function Block</th>
<th>Default value(s)</th>
<th>Function Block</th>
<th>Stg/Spd Fan</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000_Fan_Seq_EffOccMd</td>
<td>Cfg1000_FanType</td>
<td>Value=True for Staged Fan, Enter True for Variable Fanspeed</td>
<td>Boolean</td>
<td>Fan</td>
</tr>
<tr>
<td>1000_Fan_Seq_EffOccMd</td>
<td>Cfg10001_FanLevelRequestFromCtg01Ctg02</td>
<td>InA1Par=1 for Staged Fan, InA2Par=20% for Variable Fanspeed</td>
<td>BinarySelect</td>
<td>Fan</td>
</tr>
<tr>
<td>1000_Fan_Seq_EffOccMd</td>
<td>Cfg10002_FanLevelRequestFromHtg01Htg02</td>
<td>InA1Par=1 for Staged Fan, InA2Par=20% for Variable Fanspeed</td>
<td>BinarySelect</td>
<td>Fan</td>
</tr>
<tr>
<td>1000_Fan_Seq_EffOccMd</td>
<td>Cfg10003_FanLevelAfterPowerUp</td>
<td>InA1Par=1 for Staged Fan, InA2Par=20% for Variable Fanspeed</td>
<td>BinarySelect</td>
<td>Fan</td>
</tr>
<tr>
<td>1000_Fan_Seq_EffOccMd</td>
<td>Cfg1031_FanMaxLevelEffOccMd</td>
<td>In1.6Par=3/3/3/3/3 for Ccc/Off/Hol/Unocc/Stby/Byp. Change to 100% for Variable Fan</td>
<td>NumericSelect</td>
<td>Fan</td>
</tr>
<tr>
<td>1025_Fan_Window_RmTemp</td>
<td>36_OverhtPrim</td>
<td>InA2Par=3. Change to 100% for Variable Fan</td>
<td>BinarySelectMulti</td>
<td>Fan</td>
</tr>
<tr>
<td>1025_Fan_Window_RmTemp</td>
<td>39_OverhtSec</td>
<td>InA2Par=3. Change to 100% for Variable Fan</td>
<td>BinarySelectMulti</td>
<td>Fan</td>
</tr>
<tr>
<td>1025_Fan_Window_RmTemp</td>
<td>42_FrostPrim</td>
<td>InA2Par=1. Change to 20% for Variable Fan</td>
<td>BinarySelectMulti</td>
<td>Fan</td>
</tr>
<tr>
<td>1025_Fan_Window_RmTemp</td>
<td>45_FrostSec</td>
<td>InA2Par=1. Change to 20% for Variable Fan</td>
<td>BinarySelectMulti</td>
<td>Fan</td>
</tr>
</tbody>
</table>

Please use as reference the Excel file "IRMN_H_0001_Interface.xlsx".
Decoupling a FCU Sequence from the Fan

The Clg and Htg sequences are coupled with the fan. This ensures that first the fan runs and then the sequence is output. If the sequence has a value > 0%, a request to the fan logic is generated (RmSeqClg01FanReq = True). The fan is then switched to a Min level / Min position. If the feedback from the fan is present (ResFanRunning = True), then the sequence level is output.

To decouple the sequence simply set the request RmSeqClg01FanReq = False to keep the fan off for this sequence. Simulate that the fan is running so that the sequence level is not set to 0%.

=> Set the Request to False

=> Set the ResFanRunning to True
The interface description contains a column "Setpt Rel/Abs". All configurations marked with "RelAbs" needs to be changed. Using the "View - AX-Property Sheet" of a folder makes it easy to find the right function blocks.

<table>
<thead>
<tr>
<th>Folder</th>
<th>Function Block</th>
<th>Default value(s)</th>
<th>Functionblock</th>
<th>Setpt Rel/Abs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0050_Setpoint</td>
<td>Clg0050_MinClgSetptSelection</td>
<td>In1.6 Par = -5/0/0/0/-5/5 for Occ/Off/Hot/Unocc/Stby/Byp; -9 for °F; 12°C/55°F Abs</td>
<td>Const5Numeric</td>
<td>RelAbs</td>
</tr>
<tr>
<td>0050_Setpoint</td>
<td>Clg0051_MaxClgSetptSelection</td>
<td>In1.6 Par = +5/0/0/0/+5/+5 for Occ/Off/Hot/Unocc/Stby/Byp; +9 for °F; 30°C/85°F Abs</td>
<td>Const5Numeric</td>
<td>RelAbs</td>
</tr>
<tr>
<td>0050_Setpoint</td>
<td>Clg0052_MinHtgSetptSelection</td>
<td>In1.6 Par = -5/0/0/0/-5/5 for Occ/Off/Hot/Unocc/Stby/Byp; -9 for °F; 12°C/55°F Abs</td>
<td>Const5Numeric</td>
<td>RelAbs</td>
</tr>
<tr>
<td>0050_Setpoint</td>
<td>Clg0053_MaxHtgSetptSelection</td>
<td>In1.6 Par = +5/0/0/0/+5/+5 for Occ/Off/Hot/Unocc/Stby/Byp; +9 for °F; 30°C/85°F Abs</td>
<td>Const5Numeric</td>
<td>RelAbs</td>
</tr>
<tr>
<td>0050_Setpoint</td>
<td>Clg0060_HwAirSetptRelAbsCelsFahrenheit</td>
<td>Value = 1=RelCelsius. Also possible: 1=Rel °C, 2=Abs °C, 3=Rel °F, 4=Abs °F</td>
<td>Const1Numeric</td>
<td>RelAbs</td>
</tr>
<tr>
<td>0200_WM_PID_RmTemp</td>
<td>Sylk_Conv_Ext_WallModule</td>
<td>SetpointOvrd to -5/5/-5/5°C or -9/9/-9/9°F or 12/30/12/30°C or 55/85/55/85°F</td>
<td>WmConfigHvacA</td>
<td>RelAbs</td>
</tr>
<tr>
<td>0200_WM_PID_RmTemp</td>
<td>Sylk_Conv_Ext_WallModule</td>
<td>SetptOvrdTypePar = Relative / Absolute</td>
<td>WmConfigHvacA</td>
<td>RelAbs</td>
</tr>
</tbody>
</table>

Please use as reference the Excel file "IRMN_H_0001_Interface.xlsx".
Changing between °C and °F

Select Measurement Type

All the parameters from the wallmodule Function block need to set to °F values. See below example.
Using the IRM_H_0001 application requires some more changes. Using the "View - AX-Property Sheet" of a folder makes it easy to find the right function blocks.

Please use as reference the Excel file "IRMN_H_0001_Interface.xlsx".
1. Setpoint

With a wired wallmodule, the setpoint is read in via the "On board IO" wiresheet. The "UI Terminal" function block is read in with the Characteristic "Setpt10kCharacteristic". The corresponding Out value is then 0..100%. Then the 0..100% are converted into the corresponding setpoint (e.g. -5..5°C or -9..9°F or 12..30°C / 55..85°F) via a LinearGraph function block. Below values are used for the LinearGraph function block. The Out from linear graph is connected to the wallmodule functionblock WmConfigHvacA to the input Setpt Adjust Wired Wm. The wallmodule functionblock “WmConfigHvacA” needs to be configured with WmModel = WiredWmTempSetpt...

<table>
<thead>
<tr>
<th>WmModel</th>
<th>WiredWmTempSetpt</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Relative °C</th>
<th>Absolute °C</th>
<th>Relative °F</th>
<th>Absolute °F</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1</td>
<td>95.74</td>
<td>99.58</td>
<td>95.77</td>
</tr>
<tr>
<td>X2</td>
<td>14.26</td>
<td>11.5</td>
<td>14.23</td>
</tr>
<tr>
<td>Y1</td>
<td>-5</td>
<td>12</td>
<td>-9</td>
</tr>
<tr>
<td>Y2</td>
<td>5</td>
<td>30</td>
<td>9</td>
</tr>
</tbody>
</table>

Used parameters for the linear graph functionblock
On board IO

Periodic program with example of Relative Fahrenheit -9..9K
2. Push Button

To enable the Push Button on the wallmodule, select in GeneralSettings - WmModel a wired wallmodule with a button. Select in GeneralSettings - OccOvrdSelection = Enabled. After “Save”, a new menu called OccupancyOvrd is shown.

The OccOvrdTypeBitsPar determines whether an Occupancy Override via the button is possible and which modes should be possible. If no override is to be possible, 0 is entered here. If several overrides should be possible, then add the numbers. See example below.

0 = No manual Override at all.
2 = Overwrite to Holiday mode is possible by long press.
4 = Overwrite to Unoccupied is possible by short press.
32 = Overwrite to Bypass is possible by short press.

Example: The button should change the occupancy mode to Unoccupied and Bypass -> OccOvrdTypeBitsPar = 4 + 32 = 36

Note: The OccOvrdTypeBits are also available as Input Slot, selecting the slots over the View – AX Slot Sheet – OccOvrdTypeBits – Select Summary and deselect Hidden
Note: If the button is pressed briefly and released again, this is recognized easily as a short keystroke.

If the key is pressed and not released, the LED status is first kept at the current status when the key is pressed, and the LED is briefly switched on or off after 3 seconds to signal a short press. After a total of 6 seconds, a long press is signaled by switching the LED again. The button can then be released.

Example:


Each override to a certain occupancy mode can be released by a short press of the button. The override Button has no effect, if the scheduler says “Off” or “Holiday” or “Bypass”.
The interface description contains a column "Hw Availability". These points must be adapted to the connected sensors/actuators. If, for example, a sensor is not connected, the corresponding "...Available" parameter must be set to "false".

<table>
<thead>
<tr>
<th>Folder</th>
<th>Function Block</th>
<th>Default value(s)</th>
<th>FunctionBlock</th>
<th>Hw Availability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0001_Occupancy</td>
<td>Cfg0002_HWOccSensorAvailable</td>
<td>Value=False; Sensor not available; true=Sensor available</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01S0_Sensors_WmReset</td>
<td>Cfg0150_HwRmTempSensorAvailable</td>
<td>Value=False</td>
<td>Const1Boolean</td>
<td>Hw</td>
</tr>
<tr>
<td>01S0_Sensors_WmReset</td>
<td>Cfg0151_HwRmHumSensorAvailable</td>
<td>Value=False</td>
<td>Const1Boolean</td>
<td>Hw</td>
</tr>
<tr>
<td>01S0_Sensors_WmReset</td>
<td>Cfg0152_HwRmCO2SensorAvailable</td>
<td>Value=False</td>
<td>Const1Boolean</td>
<td>Hw</td>
</tr>
<tr>
<td>01S0_Sensors_WmReset</td>
<td>Cfg0153_HwWindowContactAvailable</td>
<td>Value=False</td>
<td>Const1Boolean</td>
<td>Hw</td>
</tr>
<tr>
<td>01S0_Sensors_WmReset</td>
<td>Cfg0154_HwAirFlowContactAvailable</td>
<td>Value=False</td>
<td>Const1Boolean</td>
<td>Hw</td>
</tr>
<tr>
<td>01S0_Sensors_WmReset</td>
<td>Cfg0155_CO2_SensorType</td>
<td>Y2Par=2000 ppm</td>
<td>LinearGraph</td>
<td>Hw</td>
</tr>
<tr>
<td>0200_WM_PID_RmTemp</td>
<td>Syik_Conv_Ext_WallModule</td>
<td>.Setpoint.SetptOffTimePar = 3sec</td>
<td>WmConfigHvacA</td>
<td>Hw</td>
</tr>
</tbody>
</table>

Please use as reference the Excel file "IRMN_H_0001_Interface.xlsx".
Sensor break/short recognition

The **BACnet Numeric Input** function block represents the sensor value on BACnet. The function block is set to activate an alarm if the sensor is not connected (open loop), short-circuited (short loop) or out of the valid range. If the sensor is not used at all in the application, then the **BACnet Numeric Input** function block should be deleted on the wiresheet and the value **Null** (nan) should be assigned as sensor value. The other application logic evaluates **Null** accordingly as a non-existing sensor. By deleting, the number of BACnet points is reduced. Below some recommendations for different sensor types.
1. Relative Humidity

The sensor value is in the normal range.

- Status Flags: Normal (All bits are 0)
- Event State: Normal
- Reliability: No Fault Detected

The sensor value is too high / low. An alarm is created.

- Status Flags: InAlarm
- Event State: HighLimit or LowLimit
- Reliability: No Fault Detected

Note: Event Enable To Fault is set to Fault, because the Status Flags and Event State are never set to Fault because there is no sensor break detection for 0-10V.

Note: Property 353 = Event Detection Enable
2. Temperatures

<table>
<thead>
<tr>
<th>Status Flags</th>
<th>Event State</th>
<th>Reliability</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal (all bits 0)</td>
<td>Normal (Enum=0)</td>
<td>No Fault Detected (Enum=0)</td>
</tr>
</tbody>
</table>

The sensor value is in the normal range.

The sensor value is too high or too low.

-115 / <110°C, < -35 / > -30°C

Status Flags:
- Bit1=InAlarm
- Bit2=Fault
- Bit3=Overridden
- Bit4=OutOfService

Event State:
- HighLimit (Enum=3)
- LowLimit (Enum=4)

Reliability:
- No Fault Detected (Enum=0)

The sensor is open or shorted.

Status Flags:
- InAlarm + Fault
- Bit1=InAlarm
- Bit2=Fault
- Bit4=OutOfService

Event State:
- Fault (Enum=1)

Reliability:
- Open Loop (Enum 4)
- Shorted Loop (Enum 5)

Note: Property 353 = Event Detection Enable

View from BACnet analyzer BACShark

---

Note: Property 353 = Event Detection Enable

View from BACnet analyzer BACShark

---

Note: Property 353 = Event Detection Enable

View from BACnet analyzer BACShark
3. Cooling / Heating Outputs

The actuator is enabled and controlled from the PID controller.

Out_Of_Service can be set to Out Of Service. Here is an example, which means that the Present_Value coming from the Priority_Array is used as Out value.

The function block determines that another entry with a higher priority exists via the Priority_Array. Only the Out value differs from the In value.

Note: The Status_Flag: Overridden list has a special meaning as BACnet, e.g. it could be used to invalidate on BACnet or a repair task has been executed on site (that means there is no longer a connection to the device).

Status Flags: Normal (true value)
- Bit flags: BACnet, Bit-Overridden, Bit-OutOfService

Event State: Normal (true value)
- Event State Array: Normal, Out-Of-Service

Reliability: No Fault Detected (true value)
- Event with 15 entries like over-range, short-loop, ...

Status Flags: OutOfService
- Bit flags: BACnet, Bit-Overridden, Bit-OutOfService

Event State: Out Of Service (true value)
- Event State Array: Normal, Fault, Out-Of-Service

Reliability: No Fault Detected (true value)
- Event with 15 entries like over-range, short-loop, ...

View from BACnet analyzer BACShark

If a priority value would be in the Priority_Array, this value would be output as Present_Value and as Out value.

By setting Out_Of_Service = True, the Priority_Array no longer has any influence on Out Present_Value. The value with the highest priority from the Priority_Array, which is Out-Of-Service, is the last Present_Value that was present when the Out_Of_Service was set to True.

Note: Property 353 = Event Detection Enable
The points marked with "Flash" are parameters that can also be written via BACnet. Note that these parameters are saved in the Flash memory and therefore they may not be written periodically over BACnet. Otherwise the flash memory will be damaged.

Please use as reference the Excel file "IRMN_H_0001_Interface.xlsx".

<table>
<thead>
<tr>
<th>Folder</th>
<th>function Block</th>
<th>Default value(s)</th>
<th>Function block</th>
<th>Flash</th>
</tr>
</thead>
<tbody>
<tr>
<td>0200_WM_PID</td>
<td>Cfg0200_RmTempSetptClgUnocc</td>
<td>Default Value=28°C; with Imperial eng. unit, enter 82°F and then make a Power Up.</td>
<td>BacnetNumericValue</td>
<td>Flash</td>
</tr>
<tr>
<td>0200_WM_PID</td>
<td>Cfg0201_RmTempSetptClgStby</td>
<td>Default Value=25°C; with Imperial eng. unit enter 77°F and then make a Power Up.</td>
<td>BacnetNumericValue</td>
<td>Flash</td>
</tr>
<tr>
<td>0200_WM_PID</td>
<td>Cfg0202_RmTempSetptClgOcc</td>
<td>Default Value=23°C; with Imperial eng. unit enter 73°F and then make a Power Up.</td>
<td>BacnetNumericValue</td>
<td>Flash</td>
</tr>
<tr>
<td>0200_WM_PID</td>
<td>Cfg0203_RmTempSetptHtgOcc</td>
<td>Default Value=21°C; with Imperial eng. unit enter 70°F and then make a Power Up.</td>
<td>BacnetNumericValue</td>
<td>Flash</td>
</tr>
<tr>
<td>0200_WM_PID</td>
<td>Cfg0204_RmTempSetptHtgStby</td>
<td>Default Value=19°C; with Imperial eng. unit enter 68°F and then make a Power Up.</td>
<td>BacnetNumericValue</td>
<td>Flash</td>
</tr>
<tr>
<td>0200_WM_PID</td>
<td>Cfg0205_RmTempSetptHtgUnocc</td>
<td>Default Value=16°C; with Imperial eng. unit enter 61°F and then make a Power Up.</td>
<td>BacnetNumericValue</td>
<td>Flash</td>
</tr>
<tr>
<td>0200_WM_PID</td>
<td>Cfg0220_PIDFlmTempClgxP</td>
<td>Default Value=8; with Imperial eng. unit enter 14°F and then make a Power Up.</td>
<td>BacnetNumericValue</td>
<td>Flash</td>
</tr>
<tr>
<td>0200_WM_PID</td>
<td>Cfg0221_PIDFlmTempClgTl</td>
<td>Default Value=1200s. Make a Power Up after a change.</td>
<td>BacnetNumericValue</td>
<td>Flash</td>
</tr>
<tr>
<td>0200_WM_PID</td>
<td>Cfg0222_PIDFlmTempClgTd</td>
<td>Default Value=0s. Make a Power Up after a change.</td>
<td>BacnetNumericValue</td>
<td>Flash</td>
</tr>
<tr>
<td>0200_WM_PID</td>
<td>Cfg0223_PIDFlmTempHtgXp</td>
<td>Default Value=8; with Imperial eng. unit enter 14°F and then make a Power Up.</td>
<td>BacnetNumericValue</td>
<td>Flash</td>
</tr>
<tr>
<td>0200_WM_PID</td>
<td>Cfg0224_PIDFlmTempHtgTl</td>
<td>Default Value=1200s. Make a Power Up after a change.</td>
<td>BacnetNumericValue</td>
<td>Flash</td>
</tr>
<tr>
<td>0200_WM_PID</td>
<td>Cfg0225_PIDFlmTempHtgTd</td>
<td>Default Value=0s. Make a Power Up after a change.</td>
<td>BacnetNumericValue</td>
<td>Flash</td>
</tr>
</tbody>
</table>
BACnet Communication Errors (Life Check, Refln COV, Refln Polling)

Life Check using Standard BACnet Mechanism

During normal standard BACnet communication, the room controller only receives values from the plant controller. The room controller does not request any values, so it does not act itself, but only receives the values.

Normally, the plant controller sends a value to the room controller when the value changes according the COV (Change of value) mechanism. If the plant controller is defective or the bus is interrupted, the room controller does not notice this and remains at its old values.

In order to find out such a communication error, the plant controller can send a new data point to the room controller at intervals of 5 minutes with an alternately value between 0 and 1. Example: 0, 1, 0, 1, ...

The room controller checks whether the 0 value or the 1 value remains stable for more than 20 minutes and if it is stable, then it knows that there is a communication error. This is 3 times the sending interval of the plant controller and therefore robust enough to say for sure that the plant controller is no longer communicating. With this information, the room controller can then set i.e. the Occupancy Scheduler to Occupied or set the outdoor temperature to null.
Reference Input Honeywell proprietary Mechanism

The Reference Input Mechanism is a Honeywell proprietary solution where the room controller itself becomes active and either subscribes for the COV mechanism or polls a value at periodic intervals from the Plant controller.

When receiving a value, the detection time of a communication error depends on the type of transmission.

In the case of a COV (Change of Value) in which only changes are sent, a new (repeated) COV subscription is sent to the sender every 15 minutes (Time not changeable). If the subscription is successful, the value is transmitted, even if there is no value change.

If the subscription was not successful, the value is polled and if the sender does not reply, a communication error occurs immediately. The poll interval can be set via Niagara.

Thus, it can be said that a communication error is detected either after 15 min or after the poll interval. The communication error is reported in the application via the Status Flag Fault, which is true then.

When sending over Reference Output, the communication error occurs after sending without an acknowledge. This happens immediately after a COV (Change of Value). The communication error is reported in the application via the Status Flag Fault, which is true then.

Reference Input Honeywell proprietary Polling Mechanism for critical points like Windspeed

The wind speed coming from a plant controller to control sun blinds should not be requested from the plant controller via COV subscription, but via the polling mechanism, because 15 min is too long time to detect a communication failure.

COV subscription means that the plant controller is informed that it should send the Windspeed when values are changed. The COV subscription is only valid for 15 minutes and then the COV subscription must be repeated, which means that the plant controller must be informed again to send value changes. If the plant controller has failed, in the worst case the room controller will only detect this after 15 minutes. This is the case where the repeated COV subscription request is not answered by the plant controller. However, 15 minutes usually takes too long to react to wind peaks.

If the polling mechanism is used, the wind speed is requested from the plant controller according to the poll rate, e.g. every minute. If the plant controller fails, the message is repeated maximal 2 times every 3 sec (can be set for the entire BACnet communication via Config - Device Object - Apdu Timeout and Number of APDU Retries). This means that with a poll rate of 1 min, after 1 min + 3*3 sec = 1:09 min it is detected that the Plant controller has failed. The Fault bit of the Status Flag is True then.

If it turns out in practice that the polling solution is not robust enough against the communication failure, the fault detection can be delayed in the application for further minutes with an OnDelay function block, so that the plant controller can be switched off for several minutes before the room controller is supposed to react.
In COV mode, it is possible that the status flags are not changed with a COV update and therefore they cannot be interpreted at RefIn. Therefore, for critical points, the polling mode should be selected!

Fault is also set to true, if there is a BACnet communication failure. In this case, Reliability is set to "Communication failure".

OutOfService shows whether this RefIn was set to Out of Service via BACnet.

--- Parameters ---

**Initial Value**
Use nan as initial value.

**Poll Rate Mode**
Use Polling as Mode instead of COV for critical points.

**Poll Rate**
Use a short time for the windspeed like 1min

--- Parameters ---

**Evaluate Bacnet Status Flags**
Evaluate Bacnet Status Flags

<table>
<thead>
<tr>
<th>Execution</th>
<th>InAlarm</th>
<th>Fault</th>
<th>Overridden</th>
<th>OutOfService</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>false</td>
</tr>
</tbody>
</table>

--- Parameters ---

**Null if Windspeed is Fault**
Constit Numeric

<table>
<thead>
<tr>
<th>Execution</th>
<th>Out</th>
<th>Null</th>
<th>InA1</th>
<th>InA2</th>
</tr>
</thead>
<tbody>
<tr>
<td>false</td>
<td>false</td>
<td>true</td>
<td>false</td>
<td>true</td>
</tr>
</tbody>
</table>

--- Parameters ---

**Windspedd**

<table>
<thead>
<tr>
<th>RefIn</th>
<th>Execution</th>
<th>Out</th>
<th>Status Flags</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>null</td>
<td>2.00</td>
<td>2.00</td>
<td>No Units</td>
<td># Windspeed polled from Plant controller any minute</td>
</tr>
</tbody>
</table>
### Changing the Default Value of the BACnet Functionblocks

When changing the **Default Value** of a BACnet FB, then the value is used on the Output only after the next power up of the controller.

<table>
<thead>
<tr>
<th>Property Sheet</th>
<th>Configuration Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cfg0200_RmTempSetptCigUnoc</td>
<td>Execution</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>Out</td>
<td>25.60 [ok]</td>
</tr>
<tr>
<td></td>
<td>Status Flags</td>
<td>6.99 [ok]</td>
</tr>
<tr>
<td></td>
<td>Event State</td>
<td>Normal</td>
</tr>
<tr>
<td></td>
<td>Out Of Service</td>
<td>- [mill]</td>
</tr>
<tr>
<td></td>
<td>In</td>
<td>25.60 [ok]</td>
</tr>
<tr>
<td></td>
<td>Units</td>
<td>No Units</td>
</tr>
<tr>
<td></td>
<td>Cov Increment</td>
<td>0.10</td>
</tr>
<tr>
<td></td>
<td>Default Value</td>
<td>25.00</td>
</tr>
<tr>
<td></td>
<td>BACnet Object Instance</td>
<td>2</td>
</tr>
</tbody>
</table>

Note: Do not write periodically.
Setting Out Of Service for BACnet Functionblocks

Drag and drop a Bacnet Functionblock, i.e. a BacnetEnumValue to the wiresheet. Then goto the Points menu and start Discover. Add the BacnetEnumValue point to the Database while changing the Type to “Enum Writable”. Under Virtual, the point appears as a multistateValue with the property outOfService. Make a double click on outOfService and then a right click on the current value. Changing to True means, that outOfService is set to true.

After setting outOfService to True
The pin slot “Out Of Service” can be selected to be visible and can be used in the wiresheet to decouple the BACnet value (Out) to an actuator.
Reset the Controller to Factory Default

The controller can be completely reset to the factory default if the Service Pin button is pressed before switching on the supply voltage and remains pressed until the green Power LED lights up for a second time. Then the BACnet Instance ID, Mac Addr, Max Masters... are set to default and the Application will be deleted. All details can be found in the “Product Installation Instruction”.

In the same way an action **Reset Controller** can be made visible by right mouse clicking on **IRM Program**. Then select **Views - Ax Slot Sheet**. Find entry with Display Name = **Reset Controller**. Right mouse click on **Reset Controller** and select **Config Flags**. Uncheck the Hidden Box.

There is a new Action “**Reset Controller**”, which deletes the controller completely.
The Functionblocks are in a separate honIrmControl Palette.
**Auto-Baud-Rate-Detection**

After power up, the controller tries to communicate over BACnet with different Baud rates. It starts with the default baud rate of 38400, then 19200, 9600, 115200, 57600, 76800 and then it repeats with 38400. After 3 min without detection of the baud rate, the controller uses the default baud rate of 38400. After that only a communication with the baud rate of 38400 is possible. To enable the controller to communicate at a different baud rate, a power-up is required to restart autobaud detection.

**Green Power LED Behavior**

The green Power LED is turned on during booting. The LED is switched off during the following firmware start. As soon as the firmware has been started successfully, the LED flashes ON every 2sec and OFF every 2sec.

**Yellow Alarm LED showing that there is no Application loaded**

The yellow Alarm LED Power remains off during booting. During the following firmware start, the LED is briefly switched on and then switched off again. As soon as the firmware has been successfully started, the LED flashes ON every 4sec and OFF every 4sec if no application is loaded.

**Recognition of Power Up in the Wiresheet-Logic**

If the controller is switched on after a power supply interruption, the function block "SystemA" outputs the value "True" one time in the very first DDC cycle. For this purpose, the function block has to be configured as operation "PowerUp".

Note: Please have also a look Niagara IRM Engineering Tool, where the LEDs are described in detail.
Niagara – Manual and Auto saving / Out of Sync Problem

**Manual Saving**
Perform a manual Saving before closing the Niagara software, otherwise the changes on the Application on the Wireshet could be lost.

Right Click on Station – Save Station

**Auto-Saving**
It is recommended to enable the Auto-Save feature and to enter a short Auto-Save Frequency during programming the Application on Wireshet.

Goto Station – Config – Services - PlatformServices

<table>
<thead>
<tr>
<th>Enable Station Auto-Save</th>
<th>Enable</th>
</tr>
</thead>
<tbody>
<tr>
<td>Station Auto-Save Frequency</td>
<td>00:00h 05m 30s [1min--inf]</td>
</tr>
<tr>
<td>Station Auto-Save Versions to Keep</td>
<td>3 [1-10]</td>
</tr>
</tbody>
</table>

**Out of Sync Problem**
If Niagara is terminated without saving the station (either manually or via Auto-Saving) and after the PC has been booted, the last changes to the application are lost. The next time Niagara is started, a difference (Out of Sync) will be detected between the controller and the project. The application can be retrieved via a Clear Project and Learn from Controller if it has previously been loaded into the controller via "Teach to Controller" or via "Immediate" Teaching Mode.
Trouble shooting

If you have made changes to the application, a "Reorder - Sort by Position" in each changed wire sheet is necessary to ensure a data flow from the left to the right function blocks. Otherwise, toggle effects occur, i.e. a value is only passed on in the logic chain in the next DDC cycle and thus the logic first determines an incorrect value.

Technical Support

If you have any problems, please feel free to contact our technical support. Please let us know:

- Niagara workbench version (4.4, 4.7 etc.)
- Router or plant controller (EAGLEHAWKNX, HAWK8000 etc.) with firmware version
- The 4-digit numeric date code as printed on the sticker on the cover. The first two numeric digits represent the year and the next two numeric digits represent the week of production date.
- Firmware version (see Firmware Version).
- If possible the tools version (see Niagara IRM Engineering Tool Version).
- Wall module type (Sylk TRxx / Conventional / External from Modbus or MS/TP...).
- 3rd-Party devices on same MS/TP bus

If you have problems with the application IRMN_H_0001, you are welcome to send us a palette with your modified IRMN_H_0001 application. Please let us know which application version was used as base (see Application and Functionblock Version).