



by Schneider Electric

BACnet – ModBUS SL Gateway Application Note

Publisher: HVAC Solution Center, Alpago (BL) – Italy

Author: Pierpaolo Armeli, Federico Marcassa

Doc. Version: v0.8





Table of Contents

1. Application Note	3
1.1. Introduction	3
1.2. Gateway Architecture	3
1.3. Gateway Settings	4
1.4. Sync Value Type Objects	4
1.5. Retrieve the PV of Input Type Objects	6
1.6. Send the PV of a Schedule	6
1.7. Slave Servicing (Keep Alive / Reboot / Comm Status)	6
2. Appendix	7
2.1. Hardware Information	7
2.1.1. Usage of the EEPROM vs BACnet	7
2.2. Acronyms	7
3. Publisher's Info	8



Application Note

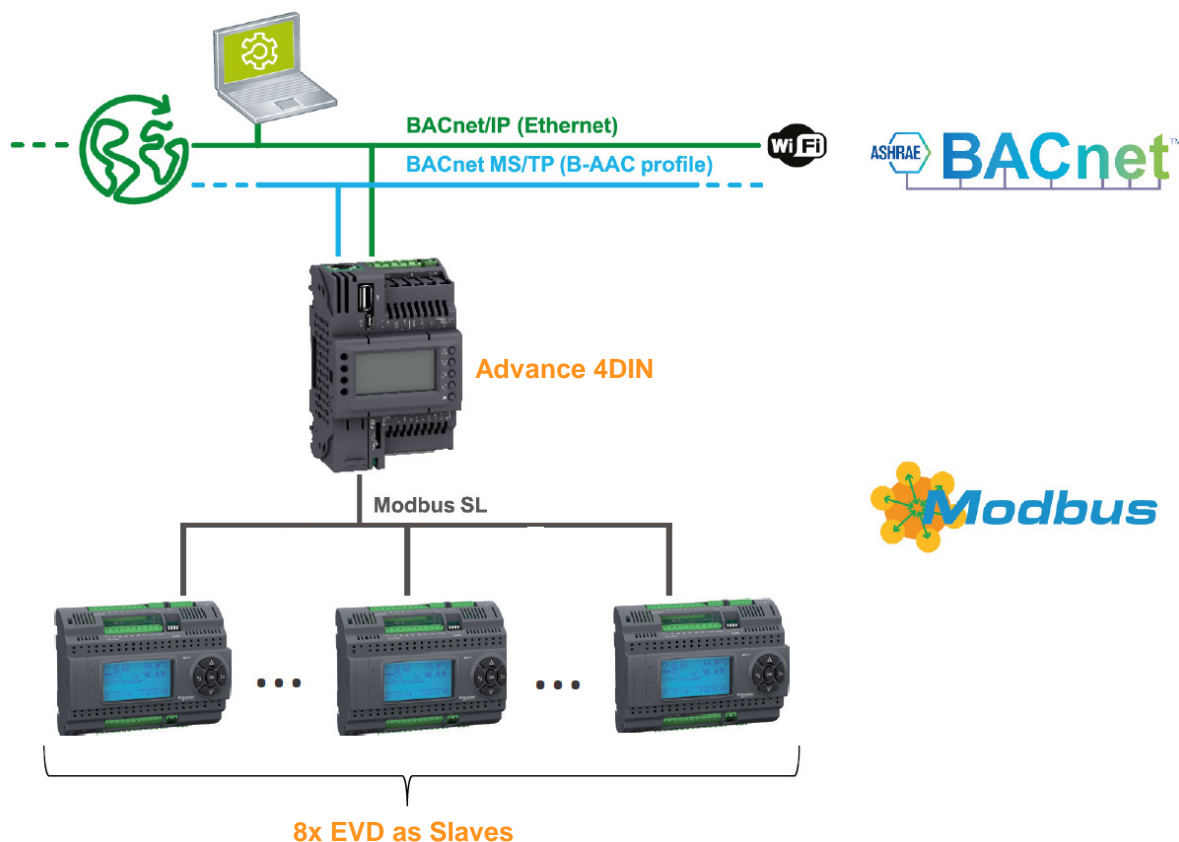
1.1. Introduction

The current documentation (v0.8) is related to the **BACnet to ModBUS SL Gateway Application Note v0.8**. The Application Note is tested/supported on FreeStudio 3.9.1 and greater version.

The aim of the Gateway is to show how to implement the **BACnet ModBUS Gateway library**. The BACnet Addons and the ModBUS Addons library are also used, but there are other official project samples that are specifically designed to show how to implement them.

1.2. Gateway Architecture

The architecture of the Application Note includes one Advance 4DIN that acts as BACnet – ModBUS SL Gateway and up to 8 EVD that represent the Slaves. The slaves do not need to be the same type of device, in fact the Gateway can be modified to reflect the specific needs of each project, providing full flexibility of implementation.



In the Application Note, even if it is not a must, the registers which need **only to be read** from the Slaves are managed through **Connection**, while the **read/write** ones are managed in **Application** through the ModBUS on event functions. This is done because the messages executed from Connection are managed by the controller in a **more efficient** way.



The ModBUS messages on event do not support the wait before send option yet. Therefore, the communication with the Smart cannot yet be executed from Application. For this reason, an Evolution has been used to create the Application Note. Anyway, a gateway could also be created e.g. sending ModBUS messages only from Connection and therefore fully supporting the Smart.

The **scope of the Gateway** in the Application Note is to:

1. Read and publish on BACnet the Room Temperature, the Outdoor Temperature, the Alarm and the Working Mode related to each slave.
2. Read/write and publish on BACnet the Cooling Setpoint, the Heating Setpoint, the Temperature Differential, the Machine State, the Season Mode.
3. Write the Machine State based on a Schedule (different variable on the slave with respect to the above-mentioned Machine State).

The **additional functionalities not directly related to the Gateway**, but more in general to the ModBUS communication, that are used in the Application Note, are:

- Monitoring the ModBUS communication on the slave side.
- Rebooting the slave from the Master with a ModBUS command.
- Keeping the slave alive (minimum amount of ModBUS messages to be sent to the slave in a specified timeframe, else the slave enters an error state).
- Managing the slave nodes presence based on their real status.

1.3. Gateway Settings

The main setting of the gateway is the **Slaves_Number**, which disables the slaves that are not present. The maximum configurable slaves number is 8 and the following BACnet Objects are available per each slave:

- 3+1 Analog Values
- 2 Analog Inputs
- 1 Binary Value
- 1 Binary Input
- 1 Multi State Value
- Schedule (same for all the slaves)

The number of slaves and the number of objects available per slave can be changed by adding BACnet Objects and by instantiating new function blocks that make the link between BACnet and ModBUS. Of course, it is also possible to configure different types of slaves / of slaves groups.

1.4. Sync Value Type Objects

The Analog Value, Binary Value and Multi State Value BACnet Objects can be synchronized with the variables on the slaves by using the BACnet_ModBUS_SL_AV / BACnet_ModBUS_SL_BV / BACnet_ModBUS_SL_MV function blocks. These function blocks use the ModBUS Master on event functionalities and have to be assigned to a background task.

In the Application Note there is a program per type of object (BACnet_Sync_AV_Obj, BACnet_Sync_BV_Obj, BACnet_Sync_MV_Obj).



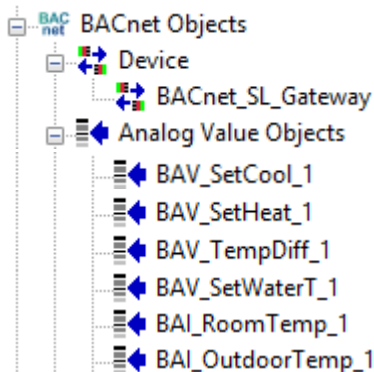
Only the structure of the program related to the Analog Value will be analyzed, as the ones related to the BV and the MV are analogous.

In the `BACnet_Sync_AV_Objs` program, the `BACnet_ModBUS_SL_AV` function block is instanced two times: **AV_link_3R** and **AV_link_1R**.

The **AV_link_3R** makes the following link:

Advance 4DIN (Gateway)

Evolution (Slave EEPROM)



#	Address	Name
1	16384	iSetpoint_Summer
6	16385	iSetpoint_Winter
2	16386	iDeltaT_Diff
3	16387	xMachineState_E2
5	16388	xWorkingMode_E2

It links three variables having contiguous addresses on the slave (16384, 16385, 16386) with three contiguous Analog Values (BAV_SetCool_1, BAV_SetHeat_1, BAV_TempDiff_1).

In order for the function block to properly execute the link, the information required are:

- The index of the first AV to link (in this case, it is 0)
- The number of registers to link (in this case 3)
- The address of the first variable to link on the slave (in this case, it is 16384)
- The scaler used for the specific variable (therefore, in case of contiguous registers that use different scalers, the same FB instance cannot be used)
- The information about whether the Relinquish Default of the AVs is set to be in EEPROM
- The structure containing the information about the slave (strModbusRTUslave type).

In order to reduce the load on the background task, the code is structured in cases that therefore run only one link and only one slave per background cycle execution.

The BACnet Objects related to the Slaves that are not connected are set to OUT OF SERVICE in a separate FOR cycle. The number of connected slaves is defined by the **Slaves_Number** EEPROM parameter. This is true and possible only because the base hypothesis is that all the slaves are of the same type. In case there are groups of different types of slaves, the code structure has to be modified accordingly.

In a similar way, the link for the iSetWaterT parameter is established (**AV_link_1R**).



The programs related to AVs and MVs show how to make the link using ST code, while the BVs link is made in FBD.



1.5. Retrieve the PV of Input Type Objects

The Present Value of the Input Type BACnet Objects is set by using the BACnet_ModBUS_SL_AI and BACnet_ModBUS_SL_BI function blocks.

In the related BACnet_Sync_AI_Objs and BACnet_Sync_BI_Objs programs, the values read from the Slaves by using the *ModBUS* messages in Connection are used as inputs for the function blocks.

1.6. Send the PV of a Schedule

The BACnet_Sync_Sc_Objs program uses the BACnet_ModBUS_SL_Sc_REAL to send the present value (in REAL) of a Schedule BACnet Object to a specific REAL variable of the slave. In case the value on the Slave is changed / not aligned with the Present Value of the Schedule, the function block writes the correct value again on the slave.

The function block requires the address of the variable on the slave, the slave info structure and the index of BACnet Schedule.

1.7. Slave Servicing (Keep Alive / Reboot / Comm Status)

The gateway includes the following functionalities:

- Use of the **strModbusRTUslave structure** to keep all the slave information available (e.g status) at each ModBUS message sent. This optimizes the communication (messages are not sent to slaves that are not present).
This allows to keep the slave alive by reading a register on it (not required for the Evolution, but required e.g. for the ATV drives). The nodes defined in Connection are disabled in case the slave is not PRESENT, in order to speed up the ModBUS connection.
- **Slave Keep Alive and Connection nodes management** (RTU_NodeMgt program).
This allows to keep the slave alive by reading a register on it (not required for the Evolution, but required e.g. for the ATV drives). The nodes defined in Connection are disabled in case the slave is not PRESENT, in order to speed up the ModBUS connection.
- **Slave Reboot command from the Gateway** (ModBUS_slvComErr_Reboot program):
By setting the xReboot_slvX variable to TRUE, the X slave will be rebooted. Of course, this works only if the slave has such a functionality setup.
- **Communication Status** (ModBUS_slvComErr_Reboot program):
The w_SL_Status_KA WORD is sent to the Slave, by changing its value alternatively at each cycle. A procedure is implemented on the slave so that, in case there is no change for 60s, a communication error is triggered on the slave side.



Appendix

2.1. *Hardware Information*

2.1.1. *Usage of the EEPROM vs BACnet*

On the M17x platform controllers, on a single EEPROM location it is possible to write only 100.000 times (this is related to the controller hardware specs). Afterwards, it is not anymore guaranteed that the value will be retained correctly.

To each EEPROM parameter one or more EEPROM locations are assigned, depending on the data type.

The above-mentioned limitation has to be taken into account when using the FBs of this library that write on EEPROM. These include:

- The “link” FBs, and in this case what especially has to be taken under consideration is the **use of the Mode 2**. In Mode 2, the Present Value is written on the EEPROM, while in Mode 0 and 1 it is the Relinquish Default that gets written on the EEPROM instead. Normally, the Present Value is more likely to be subject to frequent changes than the Relinquish Default, therefore the risk to have more frequent writes is higher in Mode 2 than in Mode 0 or 1.
- The “BACnet_ModBUS_SL” FBs, when the Relinquish Default is set to be stored in EEPROM.

2.2. *Acronyms*

- When talking about the IEC side/code in this document, the reference is to the PLC side/code.



Publisher's Info

The **publisher** of this application note is the **HVAC Solution Center** based in Alpago (BL), Italy. Its main goal is to work on machine architecture solutions, software libraries and application notes. Its **members** are:

- **Pierpaolo Armeli**
pierpaolo.armeli@schneider-electric.com
- **Federico Marcassa**
federico.marcassa@schneider-electric.com