Application note CODESYS Control SL BACnet MS/TP

BACnet MS/TP via BACnet-MS/TP-Router MBS UBR-01 Mk II

CONTENT

1 Introduction

CODESYS is an automation platform based on a development interface under Windows and runtime environments for customer-specific or standard devices such as PCs or boards with ARM/Linux. The engineering of applications is carried out according to the international standard IEC 61131-3 and can also be used for tasks in building automation.

BACnet is one of the most important communication protocols in the field of building automation. The product CODESYS BACnet SL expands the functionality of a PLC by adding the possibility to implement applications in the field of building automation in compliance with ANSI/ASHRAE Standard 135-2012, directly integrated into the CODESYS Development System.

CODESYS BACnet SL can be licensed for PLCs with CODESYS Control SL and is available in the CODESYS Store: https://store.codesys.com/codesys-bacnet-sl.html.: https://store.codesys.com/codesys-bacnet-sl.html.

Although CODESYS BACnet generally allows the use of BACnet-MS/TP, the specific adaptation to a special hardware (UART) must be carried out by the controller manufacturer (who uses CODESYS Control). Therefore, CODESYS BACnet SL supports only BACnet-IP when used on CODESYS Control SL - but not BACnet-MS/TP.

The integration of BACnet-MS/TP operating devices with CODESYS Control SL can be implemented effortlessly using a BACnet-MS/TP router.

This example demonstrates such an integration with a product from MBS GmbH - the "UBR-01 Mk II" (https://www.mbs-solutions.de/ubr-01-mk-ii-bacnet-router - and BACnet MS/TP fieldbus modules from romutec Steuer- und Regelsysteme GmbH - RDC600 (https://www.romutec.de/Produkte/Kommunikation/RDC-BACnet.html).

2 Important features of the products selected for this example

UBR-01 Mk II

The UBR-01 universal BACnet router enables the implementation of the ISO 8802-2 (also known as BACnet/Ethernet), BACnet/IP, and MS/TP (RS485-based serial BACnet networks) BACnet network topologies.

Important features of the UBR-01:

- Integrated web server for configuration and analysis
- No moving parts like fans or the like
- Integrated and switchable network and bias resistors
- Slave proxy mode / auto slave detection
- Support of international language packets

Romutec RDC600

The RDC600 series includes fieldbus modules to control data points via RS485 protocol BACnet MS / TP. Important features of the RDC600 series:

- - Practical mix of fixed / configurable I/O
- - Integrated, local operating level in the control cabinet and optionally in the control cabinet door
- - Both centralized and decentralized use with cabling lengths up to 1200 m possible for BACnet-MS/TP
- - Commissioning of field devices even without bus communication and DDC system
- - Parameterization of all settings via BACnet objects

Other BACnet routers (such as Contemporary Controls BASrouter https://www.ccontrols.de/basautomation/basrouter.htm) allow for a similar approach to integration.

3 Basic information about BACnet-MS/TP

BACnet-MS/TP "Master-Slave/Token-Passing" is one of the BACnet data link protocols defined in the BACnet standard based on EIA-485 (as physical layer). The basic principle of BACnet-MS/TP is the so-called "token passing" - a "token" is transferred from one device to another. A device can only communicate after receiving this token. BACnet-MS/TP distinguishes between two types of devices - "master" and "slave".

BACnet-MS/TP is hardly ever used as a general BACnet backbone, since BACnet-MS/TP has too low a data rate for this purpose. As a rule, BACnet-IP is used as a general BACnet backbone.

BACnet-MS/TP supports cable lengths up to 1200 m without additional devices (such as repeaters, etc.), thus enabling distributed automation.

BACnet MS/TP devices are cost-effective alternatives for applications that have low input-output data rate requirements and where BACnet IP devices are usually not used for cost reasons (device costs, cabling costs, interference immunity without special measures).

4 Basic steps for integration

The following steps are necessary for an integration:

1. BACnet-MS/TP installation
2. Configuration and commissioning of BACnet-MS/TP
3. Integration of BACnet data points in the application (of BACnet-MS/TP fieldbus modules and/or of the BACnet-MS/TP router)

4.1 BACnet-MS/TP installation

The cabling must be carried out according to the specifications of the BACnet standard - see "ANSI/ASHRAE Standard 135-2012" - Chapter 9.2 "Physical Layer".

Important aspects:

- Two-wire "twisted pair
- Observe polarity (+/-)
- Termination resistors 120 Ohm on both sides of the network segment
- One or two pairs of bias resistors 510 Ohm within the network segment - preferably at the ends of the network segment
- If necessary, local bias resistors 47 kOhm on some devices
- Limited number of devices in the network segment (usually max. 32)

4.2 Configuration of BACnet-MS/TP

Master or Slave?

Slave devices offer an advantage: the more slave devices (i.e. the fewer masters) there are on the BACnet MS/TP bus, the fewer token transfers take place. This speeds up the BACnet MS/TP communication. On the downside, slave devices need a master on the network to establish communication. Therefore most of the BACnet MS/TP devices are executed as masters or at least configured as masters by default.

4.3 Choosing a device address for für BACnet-MS/TP

A master in the BACnet MS/TP bus cyclically sends out queries to the next potential masters between its own address and the next known master address to check whether a new master in this range has been added to the BACnet MS/TP bus. When doing this, it is necessary to wait until a timeout occurs to ensure that no new master has been added in this range. Therefore address gaps in the BACnet MS/TP device addresses should be avoided for master devices in order not to use part of the already low data rate for unnecessary queries. The setting "MaxMaster" allows a limitation of the active address range.

4.4 Integration of the BACnet data points into the application

Data points of the BACnet MS/TP fieldbus modules and the BACnet MS/TP router are represented by the respective device as BACnet (server) objects. The properties of these objects can be read or written with the CODESYS BACnet function blocks BACnetClientReadProperty and BACnetClientWriteProperty. BACnet COV/event subscriptions and any necessary device discovery are also possible using CODESYS BACnet function blocks, if required.

5 Specific integration steps based on the example

5.1 BACnet-MS/TP installation based on the example

In the example, two BACnet-MS/TP fieldbus modules RDC601 are used.
The modules have:

- 8 digital inputs
- 6 digital outputs as relay outputs with configurable manual level
- 4 analog inputs for various sensors and voltage.

Diese werden als BACnet-BinaryInput, -BinaryOutput und –AnalogInput repräsentiert.

The UBR-01 Mk II BACnet MS/TP router is equipped with integrated termination and bias resistors that can be activated with DIP switches. Provided that the UBR-01 Mk II is placed at one end of the network segment, the last resistor can be used for termination. The other end of the network segment must be terminated in the wiring.

5.2 Configuration and commissioning of BACnet-MS/TP based on the example

The BACnet MS/TP router UBR-01 Mk II is basically operated as master.
The two BACnet MS/TP fieldbus modules RDC601 are also configured as master.
For this purpose, a BACnet MS/TP device address in the range of 1...127 is selected.

In the example, the BACnet MS/TP device addresses were defined as follows:

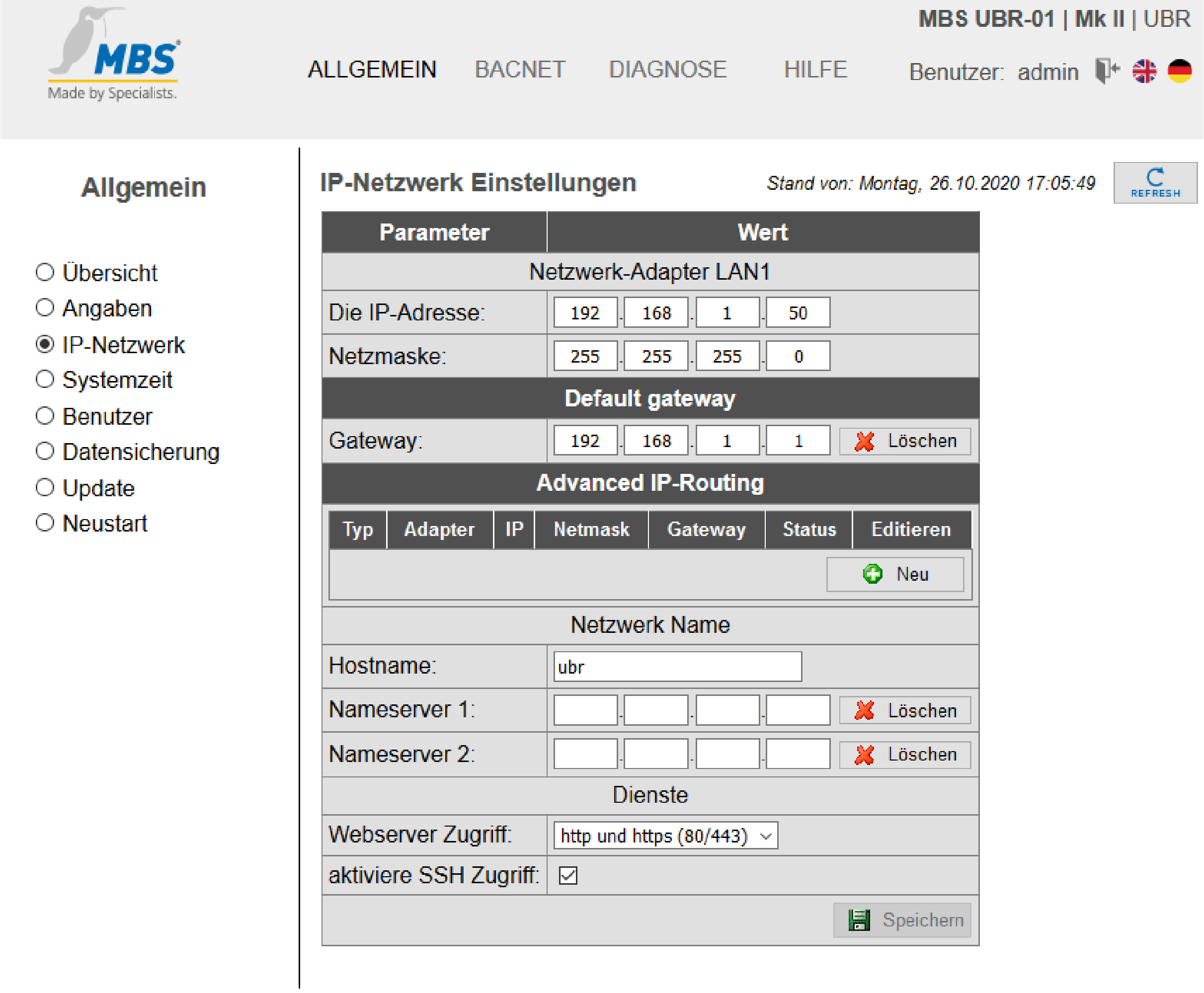
- UBR-01 – address 1
- RDC601 – addresses 2 and 3

The configuration of the BACnet MS/TP device address of the RDC601 is done via rotary coding switches on the device.

The configuration of the BACnet MS/TP device address of the UBR-01 is done via web browser.

For this purpose, an Ethernet connection to the UBR-01 must be established. Details can be found in the user manual.

After logging in to the web interface of the UBR-01, it can be configured:



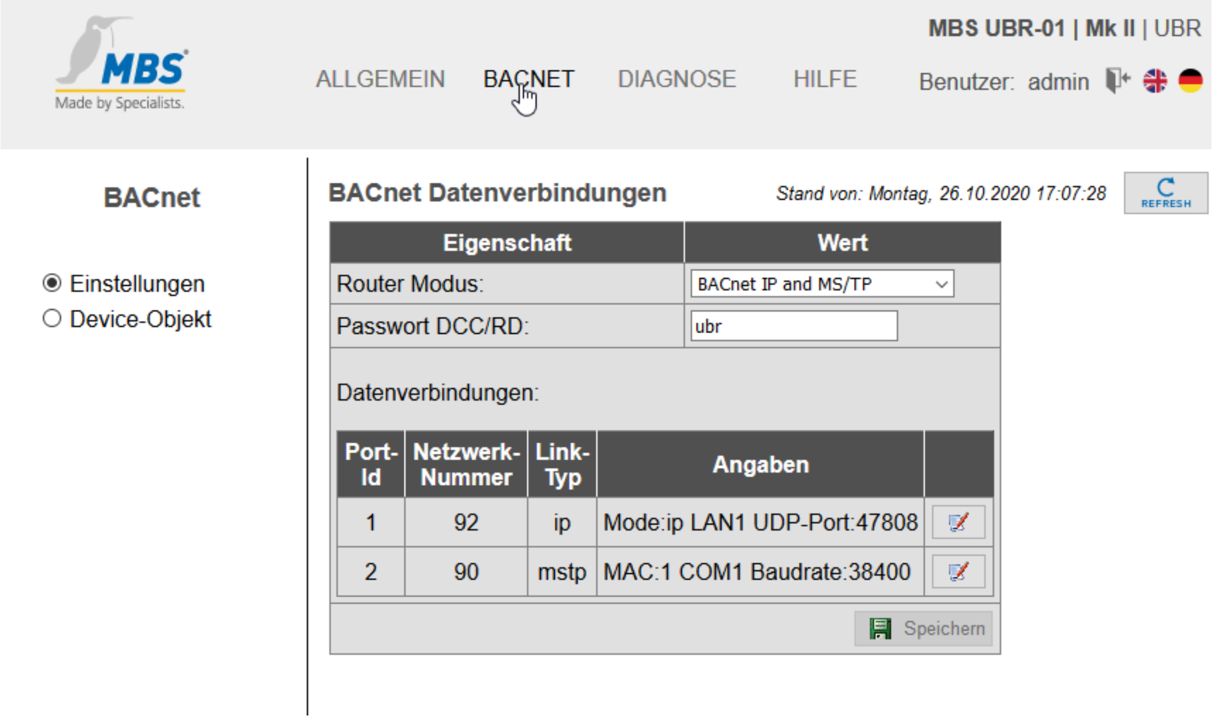
IP-Netzwerk Einstellungen

Stand von: Montag, 26.10.2020 17:05:49

Parameter	Wert
Netzwerk-Adapter LAN1	
Die IP-Adresse:	192 . 168 . 1 . 50
Netzmaske:	255 . 255 . 255 . 0
Default gateway	
Gateway:	192 . 168 . 1 . 1
Advanced IP-Routing	
Netzwerk Name	
Hostname:	ubr
Nameserver 1:	
Nameserver 2:	
Dienste	
Webserver Zugriff:	http und https (80/443)
aktiviere SSH Zugriff:	☑

Typ	Adapter	IP	Netmask	Gateway	Status	Editieren

Defining the router mode



BACnet Datenverbindungen

Stand von: Montag, 26.10.2020 17:07:28

Eigenschaft	Wert
Router Modus:	BACnet IP and MS/TP
Passwort DCC/RD:	ubr

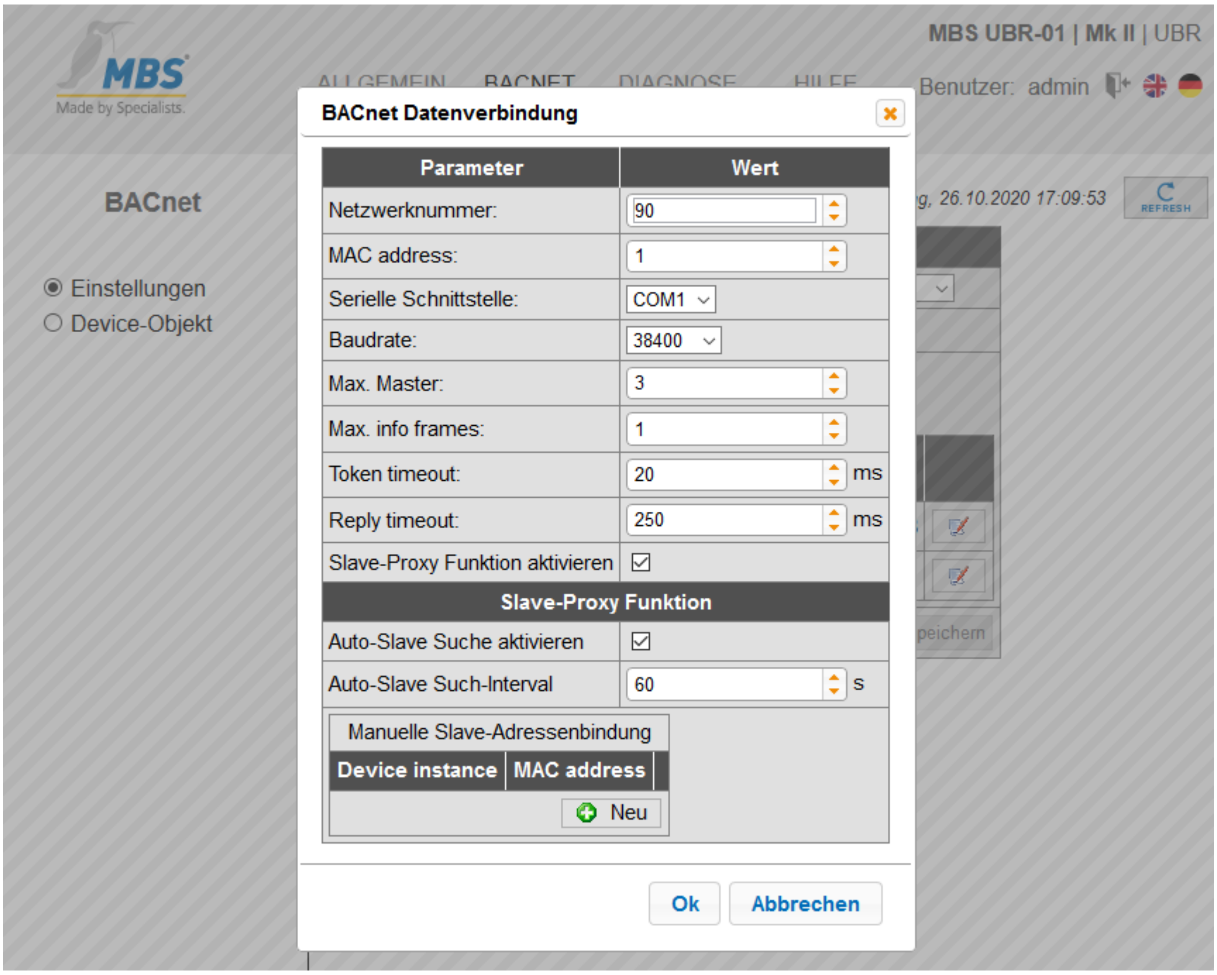
Datenverbindungen:

Port-Id	Netzwerk-Nummer	Link-Typ	Angaben	
1	92	ip	Mode:ip LAN1 UDP-Port:47808	
2	90	mstp	MAC:1 COM1 Baudrate:38400	

Configuring

- BACnet network number (of the BACnet-MS/TP segment)
- BACnet-MS/TP device address
- Interface parameters

- MaxMaster



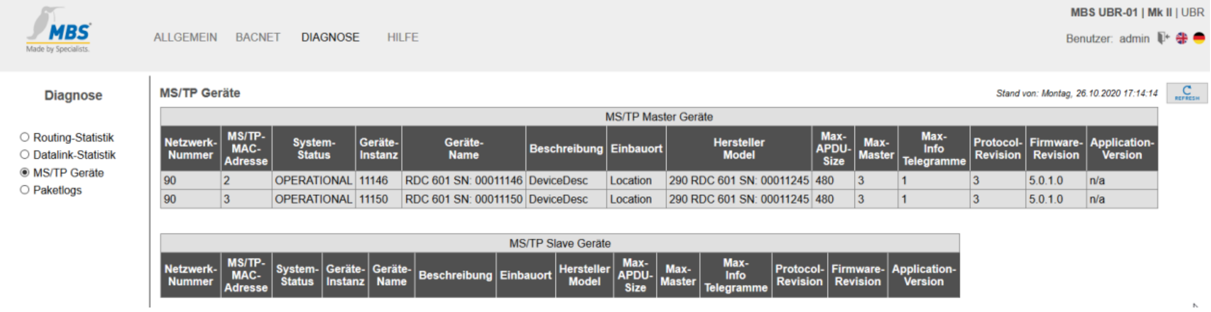
BACnet Datenverbindung

Parameter	Wert
Netzwerknummer:	90
MAC address:	1
Serielle Schnittstelle:	COM1
Baudrate:	38400
Max. Master:	3
Max. info frames:	1
Token timeout:	20 ms
Reply timeout:	250 ms
Slave-Proxy Funktion aktivieren	☑
Slave-Proxy Funktion	
Auto-Slave Suche aktivieren	☑
Auto-Slave Such-Interval	60 s

Manuelle Slave-Adressenbindung

Device instance	MAC address

Diagnosis of BACnet-MS/TP devices



MS/TP Geräte

Stand von: Montag, 26.10.2020 17:14:14

MS/TP Master Geräte

Netzwerk-Nummer	MS/TP-MAC-Adresse	System-Status	Geräte-Instanz	Geräte-Name	Beschreibung	Einbauort	Hersteller Model	Max-APDU-Size	Max-Master	Max-Info Telegramme	Protocol-Revision	Firmware-Revision	Application-Version
90	2	OPERATIONAL	11146	RDC 601 SN: 00011146	DeviceDesc	Location	290 RDC 601 SN: 00011245	480	3	1	3	5.0.1.0	n/a
90	3	OPERATIONAL	11150	RDC 601 SN: 00011150	DeviceDesc	Location	290 RDC 601 SN: 00011245	480	3	1	3	5.0.1.0	n/a

MS/TP Slave Geräte

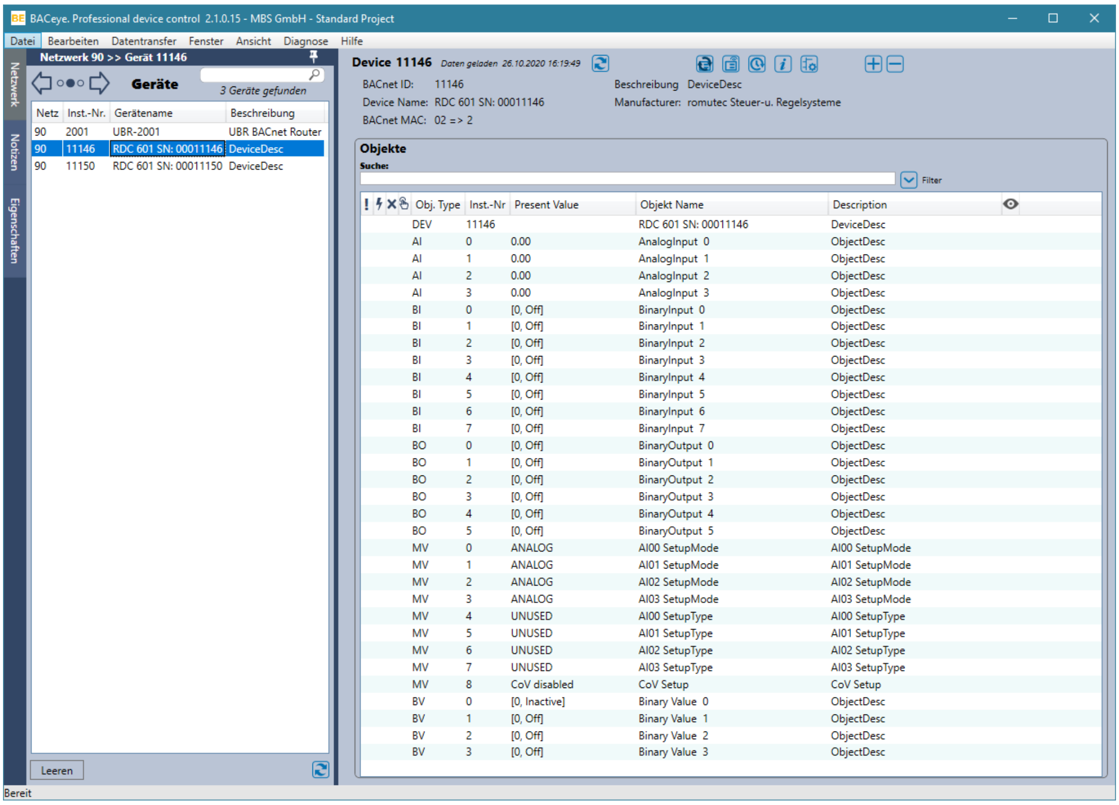
Netzwerk-Nummer	MS/TP-MAC-Adresse	System-Status	Geräte-Instanz	Geräte-Name	Beschreibung	Einbauort	Hersteller Model	Max-APDU-Size	Max-Master	Max-Info Telegramme	Protocol-Revision	Firmware-Revision	Application-Version

The UBR-01 provides the following functions that are important for commissioning:

- Routing statistics
- Datalink statistics
- Packet logs

After successful configuration and commissioning, the BACnet devices are now available in the BACnet IP network and can be edited with a BACnet explorer (in the example BACeye from MBS GmbH).

Template:



3 Geräte gefunden

Netz	Inst.-Nr.	Gerätename	Beschreibung
90	2001	UBR-2001	UBR BACnet Router
90	11146	RDC 601 SN: 00011146	DeviceDesc
90	11150	RDC 601 SN: 00011150	DeviceDesc

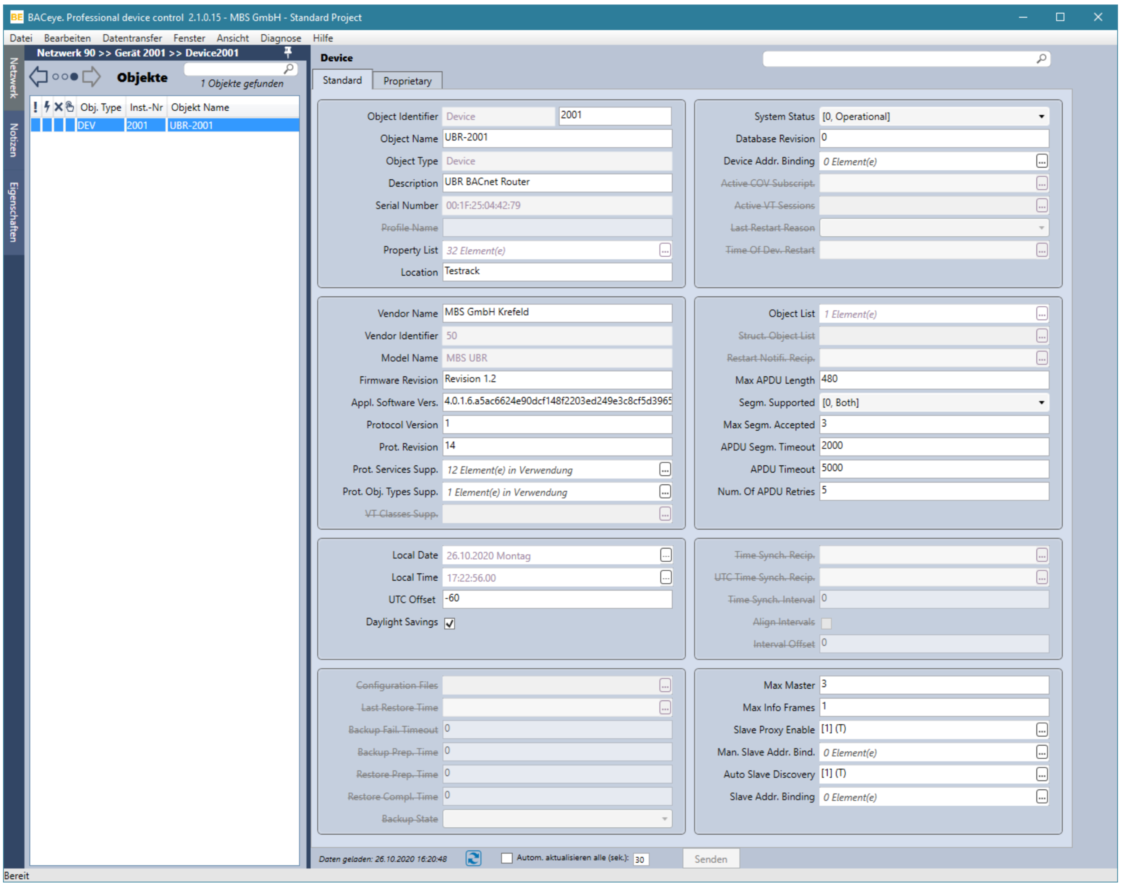
Device 11146 *Daten geladen 26.10.2020 16:19:49*

BACnet ID: 11146
Device Name: RDC 601 SN: 00011146
BACnet MAC: 02 => 2
Beschreibung DeviceDesc
Manufacturer: romutec Steuer-u. Regelsysteme

Objekte

Suche:

Obj. Type	Inst.-Nr	Present Value	Objekt Name	Description
DEV	11146		RDC 601 SN: 00011146	DeviceDesc
AI	0	0.00	AnalogInput 0	ObjectDesc
AI	1	0.00	AnalogInput 1	ObjectDesc
AI	2	0.00	AnalogInput 2	ObjectDesc
AI	3	0.00	AnalogInput 3	ObjectDesc
BI	0	[0, Off]	BinaryInput 0	ObjectDesc
BI	1	[0, Off]	BinaryInput 1	ObjectDesc
BI	2	[0, Off]	BinaryInput 2	ObjectDesc
BI	3	[0, Off]	BinaryInput 3	ObjectDesc
BI	4	[0, Off]	BinaryInput 4	ObjectDesc
BI	5	[0, Off]	BinaryInput 5	ObjectDesc
BI	6	[0, Off]	BinaryInput 6	ObjectDesc
BI	7	[0, Off]	BinaryInput 7	ObjectDesc
BO	0	[0, Off]	BinaryOutput 0	ObjectDesc
BO	1	[0, Off]	BinaryOutput 1	ObjectDesc
BO	2	[0, Off]	BinaryOutput 2	ObjectDesc
BO	3	[0, Off]	BinaryOutput 3	ObjectDesc
BO	4	[0, Off]	BinaryOutput 4	ObjectDesc
BO	5	[0, Off]	BinaryOutput 5	ObjectDesc
MV	0	ANALOG	AI00 SetupMode	AI00 SetupMode
MV	1	ANALOG	AI01 SetupMode	AI01 SetupMode
MV	2	ANALOG	AI02 SetupMode	AI02 SetupMode
MV	3	ANALOG	AI03 SetupMode	AI03 SetupMode
MV	4	UNUSED	AI00 SetupType	AI00 SetupType
MV	5	UNUSED	AI01 SetupType	AI01 SetupType
MV	6	UNUSED	AI02 SetupType	AI02 SetupType
MV	7	UNUSED	AI03 SetupType	AI03 SetupType
MV	8	CoV disabled	CoV Setup	CoV Setup
BV	0	[0, Inactive]	Binary Value 0	ObjectDesc
BV	1	[0, Off]	Binary Value 1	ObjectDesc
BV	2	[0, Off]	Binary Value 2	ObjectDesc
BV	3	[0, Off]	Binary Value 3	ObjectDesc



1 Objekte gefunden

Obj. Type	Inst.-Nr	Objekt Name
DEV	2001	UBR-2001

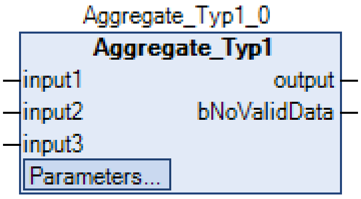
Object Identifier	Device 2001
Object Name	UBR-2001
Object Type	Device
Description	UBR BACnet Router
Serial Number	00:1F:25:04:42:79
~~Profile Name~~	
Property List	*32 Element(e)*
Location	Testrack
Vendor Name	MBS GmbH Krefeld
Vendor Identifier	50
Model Name	MBS UBR
Firmware Revision	Revision 1.2
Appl. Software Vers.	4.0.1.6.a5ac6624e90dcf148f2203ed249e3c8cf5d3965
Protocol Version	1
Prot. Revision	14
Prot. Services Supp.	*12 Element(e) in Verwendung*
Prot. Obj. Types Supp.	*1 Element(e) in Verwendung*
~~VT Classes Supp.~~	
Local Date	26.10.2020 Montag
Local Time	17:22:56.00
UTC Offset	-60
Daylight Savings	☑
~~Configuration Files~~	
~~Last Restore Time~~	
~~Backup Fail. Timeout~~	0
~~Backup Prep. Time~~	0
~~Restore Prep. Time~~	0
~~Restore Compl. Time~~	0
~~Backup State~~	

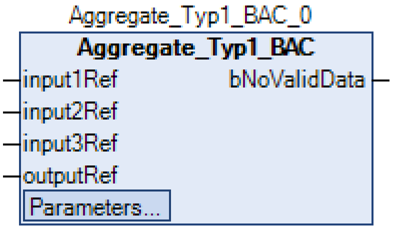
System Status	[0, Operational]
Database Revision	0
Device Addr. Binding	*0 Element(e)*
~~Active COV Subscript.~~	
~~Active VT Sessions~~	
~~Last Restart Reason~~	
~~Time Of Dev. Restart~~	
Object List	*1 Element(e)*
~~Struct. Object List~~	
~~Restart Notifi. Recip.~~	
Max APDU Length	480
Segm. Supported	[0, Both]
Max Segm. Accepted	3
APDU Segm. Timeout	2000
APDU Timeout	5000
Num. Of APDU Retries	5
~~Time Synch. Recip.~~	
~~UTC Time Synch. Recip.~~	
~~Time Synch. Interval~~	0
~~Align Intervals~~	□
~~Interval Offset~~	0
Max Master	3
Max Info Frames	1
Slave Proxy Enable	[1] (T)
Man. Slave Addr. Bind.	*0 Element(e)*
Auto Slave Discovery	[1] (T)
Slave Addr. Binding	*0 Element(e)*

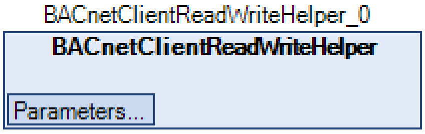
Daten geladen: 26.10.2020 16:20:48 □ Autom. aktualisieren alle (sek.): 30

5.3 Integration of BACnet data points into the application based on the example

In the example, a virtual HVAC unit is modeled, which can optionally be provided with a BACnet connection "wrapper". This approach to the design of CODESYS libraries is useful when aggregate implementations in libraries do not need to be connected directly to the inputs or outputs of process data of a specific communication protocol, but shall optionally be used with the process data of other inputs and outputs, for example with the process data of local I/Os, or different communication protocols, such as BACnet, MODBUS, or CANopen.



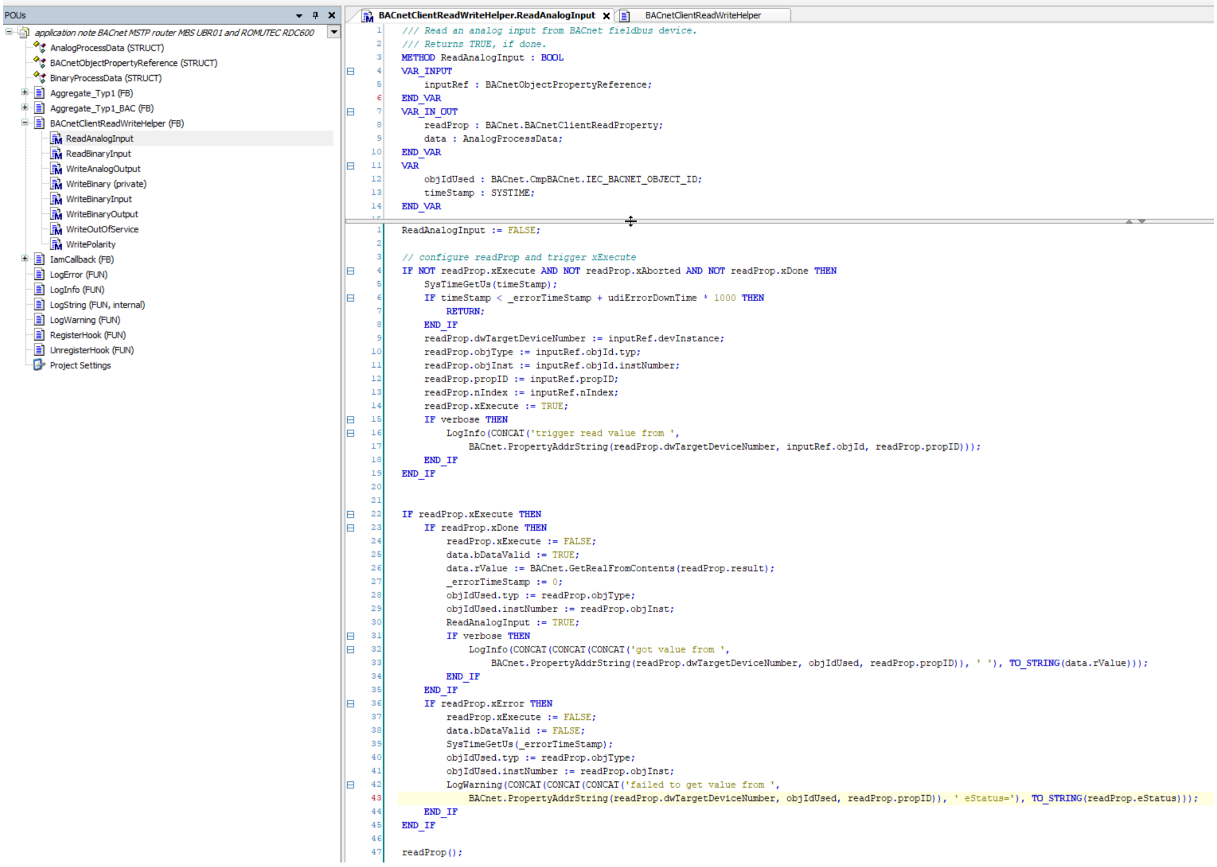




The function block Aggregate_Typ1 represents the aggregate implementation, the inputs and outputs of the process data are declared as VAR_INPUT and VAR_OUTPUT.

The function block Aggregate_Typ1_BAC represents a "BACnet wrapper" for an Aggregate_Typ1, the references to concrete BACnet data points are declared as VAR_INPUT.

The BACnetClientReadWriteHelper function block simplifies the reading / writing of BACnet object properties by means of BACnetClientReadProperty and BACnetClientWriteProperty.



```
/// Read an analog input from BACnet fieldbus device.
/// Returns TRUE, if done.
METHOD ReadAnalogInput : BOOL
VAR_INPUT
    inputRef : BACnetObjectPropertyReference;
END_VAR
VAR_IN_OUT
    readProp : BACnet.BACnetClientReadProperty;
    data : AnalogProcessData;
END_VAR
VAR
    objIdUsed : BACnet.CmpBACnet.IEC_BACNET_OBJECT_ID;
    timeStamp : SYSTIME;
END_VAR
```

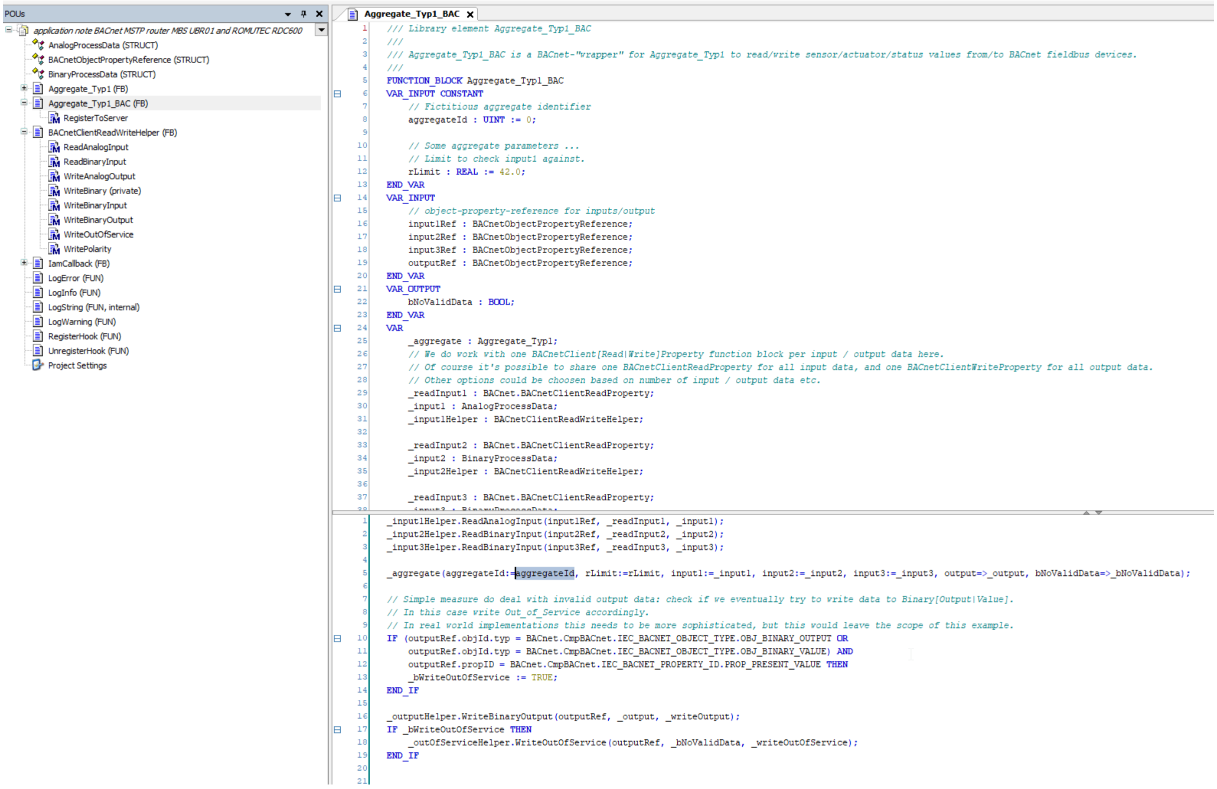
```
ReadAnalogInput := FALSE;

// configure readProp and trigger xExecute
IF NOT readProp.xExecute AND NOT readProp.xAborted AND NOT readProp.xDone THEN
    SysTimeGetUs(timeStamp);
    IF timeStamp < _errorTimeStamp + udiErrorDownTime * 1000 THEN
        RETURN;
    END_IF
    readProp.dwTargetDeviceNumber := inputRef.devInstance;
    readProp.objType := inputRef.objId.typ;
    readProp.objInst := inputRef.objId.instNumber;
    readProp.propID := inputRef.propID;
    readProp.nIndex := inputRef.nIndex;
    readProp.xExecute := TRUE;
    IF verbose THEN
        LogInfo(CONCAT('trigger read value from ',
            BACnet.PropertyAddrString(readProp.dwTargetDeviceNumber, inputRef.objId, readProp.propID)));
    END_IF
END_IF


IF readProp.xExecute THEN
    IF readProp.xDone THEN
        readProp.xExecute := FALSE;
        data.bDataValid := TRUE;
        data.rValue := BACnet.GetRealFromContents(readProp.result);
        _errorTimeStamp := 0;
        objIdUsed.typ := readProp.objType;
        objIdUsed.instNumber := readProp.objInst;
        ReadAnalogInput := TRUE;
        IF verbose THEN
            LogInfo(CONCAT(CONCAT(CONCAT('got value from ',
                BACnet.PropertyAddrString(readProp.dwTargetDeviceNumber, objIdUsed, readProp.propID)), ' '), TO_STRING(data.rValue)));
        END_IF
    END_IF
    IF readProp.xError THEN
        readProp.xExecute := FALSE;
        data.bDataValid := FALSE;
        SysTimeGetUs(_errorTimeStamp);
        objIdUsed.typ := readProp.objType;
        objIdUsed.instNumber := readProp.objInst;
        LogWarning(CONCAT(CONCAT(CONCAT('failed to get value from ',
            BACnet.PropertyAddrString(readProp.dwTargetDeviceNumber, objIdUsed, readProp.propID)), ' eStatus='), TO_STRING(readProp.eStatus)));
    END_IF
END_IF

readProp();
```

The implementation of the "BACnet-Wrapper" for Aggregate_Type1_BAC is thus very easy to implement.



```
/// Library element Aggregate_Typ1_BAC
///
/// Aggregate_Typ1_BAC is a BACnet-"wrapper" for Aggregate_Typ1 to read/write sensor/actuator/status values from/to BACnet fieldbus devices.
///
FUNCTION_BLOCK Aggregate_Typ1_BAC
VAR_INPUT CONSTANT
    // Fictitious aggregate identifier
    aggregateId : UINT := 0;

    // Some aggregate parameters ...
    // Limit to check input1 against.
    rLimit : REAL := 42.0;
END_VAR
VAR_INPUT
    // object-property-reference for inputs/output
    input1Ref : BACnetObjectPropertyReference;
    input2Ref : BACnetObjectPropertyReference;
    input3Ref : BACnetObjectPropertyReference;
    outputRef : BACnetObjectPropertyReference;
END_VAR
VAR_OUTPUT
    bNoValidData : BOOL;
END_VAR
VAR
    _aggregate : Aggregate_Typ1;
    // We do work with one BACnetClient[Read|Write]Property function block per input / output data here.
    // Of course it's possible to share one BACnetClientReadProperty for all input data, and one BACnetClientWriteProperty for all output data.
    // Other options could be choosen based on number of input / output data etc.
    _readInput1 : BACnet.BACnetClientReadProperty;
    _input1 : AnalogProcessData;
    _input1Helper : BACnetClientReadWriteHelper;

    _readInput2 : BACnet.BACnetClientReadProperty;
    _input2 : BinaryProcessData;
    _input2Helper : BACnetClientReadWriteHelper;

    _readInput3 : BACnet.BACnetClientReadProperty;
    _input3 : BinaryProcessData;
```

```
_input1Helper.ReadAnalogInput(input1Ref, _readInput1, _input1);
_input2Helper.ReadBinaryInput(input2Ref, _readInput2, _input2);
_input3Helper.ReadBinaryInput(input3Ref, _readInput3, _input3);

_aggregate(aggregateId:=aggregateId, rLimit:=rLimit, input1:=_input1, input2:=_input2, input3:=_input3, output=>_output, bNoValidData=>_bNoValidData);

// Simple measure do deal with invalid output data: check if we eventually try to write data to Binary[Output|Value].
// In this case write Out_of_Service accordingly.
// In real world implementations this needs to be more sophisticated, but this would leave the scope of this example.
IF (outputRef.objId.typ = BACnet.CmpBACnet.IEC_BACNET_OBJECT_TYPE.OBJ_BINARY_OUTPUT OR
    outputRef.objId.typ = BACnet.CmpBACnet.IEC_BACNET_OBJECT_TYPE.OBJ_BINARY_VALUE) AND
    outputRef.propID = BACnet.CmpBACnet.IEC_BACNET_PROPERTY_ID.PROP_PRESENT_VALUE THEN
    _bWriteOutOfService := TRUE;
END_IF

_outputHelper.WriteBinaryOutput(outputRef, _output, _writeOutput);
IF _bWriteOutOfService THEN
    _outOfServiceHelper.WriteOutOfService(outputRef, _bNoValidData, _writeOutOfService);
END_IF
```

Further details can be found in the attached example which could be found at https://forge.codesys.com/prj/codesys-example/appnotebacnmstp/home/Home/.