PLC TECOMAT
AS A STATION
IN BACnet NETWORK

PLC TECOMAT AS A STATION IN BACnet NETWORK

1st eddition - August 2009

CONTENT

1. BACnet

1.1 WHAT IS BACnet ?

BACnet is a communication protocol for automation and operator´s level of control of building equipment (BE). The keynote of the BACnet protocol is a formulation of a multi-purpose description of all possible objects, their functions and parameters used in BE. Specified objects, their functions and parameters are used for this purpose. BACnet systém is a worldwide norm, executive standard of building automation. It is used without any licence fees.

The description of the communication protocol exeeds the scope of this documentation, more can be found in the documet ASHARE 135-2004, www.bacnet.org)

It is presumed that the user is familiar with the basics of BACnet protocol. The tool VTS3 can be used for testing, freely accessable on the link http://sourceforge.net/projects/vts.

2. COMMUNICATION PROTOCOL BACnet IN PLC TECOMAT

2.1. REALIZATION OF STATION PLC TECOMAT IN BACnet

PLC TECOMAT can be connected to the BACnet network via the protocol:

- BACnet IP
- MS/TP (being prepared)

as a slave station.

Reqzured firmware versions:

CP-7004	sw 4.7 and higher.
FOXTROT	sw 4.7 and higher.

Required version of MOSAIC development environment 2.0.19.0 and higher.

Connection via the protocol BACnet IP is realized onto the ethernet interface of PLC TECOMAT. Configuration of interface and BACnet protocol service, see bellow.

After the connection of PLC TECOMAT into the BACnet network, defined objects with their parameters are published to other stations of BACnet network that can read and write into object parameters.

3. BACnet OBJECTS IN PLC TECOMAT

3.1. BACnet OBJECTS SUPPORTED BY PLC TECOMAT

BACnet objects and their features implemented in PLC TECOMAT.

Table 1: implemented objects and their features

	Device	Program	Binnary input	Binnary output	Binnary value	Analog input	Analog output	Analog value
Object Identifier	☑	☑	☑	☑	☑	☑	☑	☑
Object Name	☑	☑	☑	☑	☑	☑	☑	☑
Object Type	☑	☑	☑	☑	☑	☑	☑	☑
Description	☑	☑	☑	☑	☑	☑	☑	☑
System Status	☑	☑						
Vendor Name	☑							
Vendor Identifier	☑							
Model Name	☑							
Firmware Revision	☑							
Protocol Version	☑							
Location	☑							
Services Supported	☑							
Object Types Supported	☑							
Object List	☑							
Present Value			☑	☑	☑	☑	☑	☑
Units						☑	☑	☑
Status Flags		☑	☑	☑	☑	☑	☑	☑
Event State			☑	☑	☑	☑	☑	☑
Out of service		☑	☑	☑		☑	☑	
Description of halt		☑						
Polarity			☑	☑				

3.2. DEVICE

Device object (device) is used for identification of device in the BACnet network. Object contains features describing device parameters.

Table 2: device object (device) and its features

Device	Description
Object Identifier	Object type identifier: object ID vide infra *****
Object Name	Object name: TECOMAT BACnet server
Object Type	Object type: DEVICE = 8
Description	Object description: PLC TECOMAT with BACnet server
System Status	Object status: STATUS OPERATIONAL
Vendor Name	Producer name: TECO a.s. Czech Republic
Vendor Identifier	Producer identification number: 344
Model Name	PLC TECOMAT model name: TECOMAT FOXTROT / TECOMAT CP-7004
Firmware Revision	Firmware revision: PLC TECOMAT firmware version
Protocol Version	Protocol version: 1
Location	Location: CZ
Services Supported	List of supported services BACnet: see ASHARE 135-2004
Object Types Supported	List of supported objects
Object List	List of objects implemented and used within PLC TECOMAT

***** feature Object Identifier is set in the configuration dialogue of the MOSAIC development environment

3.3. PROGRAM

Object program is used for:

- identification of application program in PLC (program name, compiler version, PLC type, compiler used, time and date of compilation)
- program status RUN / HALT
- detection of status cause HALT – error message of PLC diagnostics is available

Table 3: object program and its features

Program	Description
Object Identifier	Object type identifier: 0 (only one PLC program instance)
Object Name	Object name: program name
Object Type	Object type: PROGRAM = 16
Description	Object description: program header (program name, version, PLC type, compiler used, date and time of compilation)
System Status	Object system status: RUN / HALT
Status Flags	Program run status: faultless, warning, error
Out of service	TRUE if the program is in HALT status, FALSE if it is in RUN status
Description of halt	Description of HALT status cause: e.g. E-80-04-0000 Invalid program at EEPROM

3.4. BINARY INPUT

Object binary input is used for:

- detection of PLC binary input status
- detection of binary input control error (set the user program)
- simulation of binary input status (manual setup to TRUE/FALSE value, (1/0), for testing)

Table 4.1: objekt binary input and its features

Binary input	Description
Object Identifier	Object type identifier: 0..maximum number of binary PLC inputs visible in BACnet (typically 32) see further, configuration of MOSAIC environment
Object Name	Object name: object name e.g. BINARY_INPUT_0
Object Type	Object type: BINARY_INPUT = 3
Description	Object description: description e.g. Description of BINARY_INPUT_0
Present Value	Binary input present value: TRUE / FALSE, (1/0)
Status Flags	Input status: OUT OF SERVICE=1 if there is a requirement on OUT OF SERVICE
Event State	Always EVENT_STATE_NORMAL
Out of service	1 = OUT OF SERVICE, disconnection of input physical clamp, valid is considered the value from Preset Value that can be edited as value TRUE / FALSE
Polarity	AlwaysPOLARITY_NORMAL

BACNET_TYP_FB_BI type declaration - BINARY INPUT

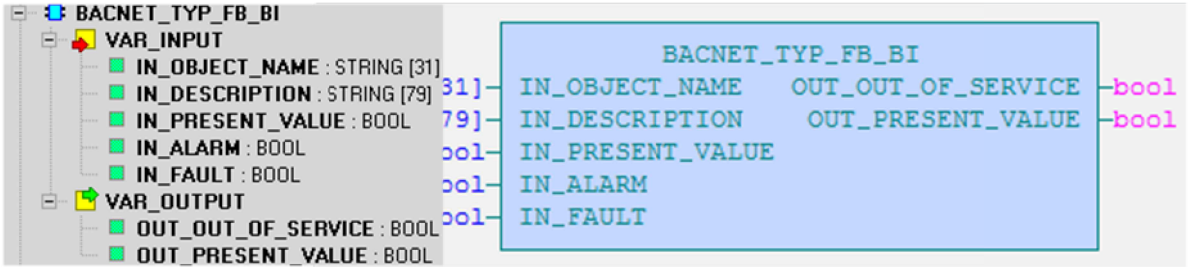


Table 4.2: The signification of BACNET_TYP_FB_BI data type items

Binary input	Description
IN_OBJECT_NAME	Object name: e.g. BINARY_INPUT_0, FIRE_007,…
IN_DESCRIPTION	Object description: e.g. Description of BINARY_INPUT_0, Fire alarm boiler room K12, etc.
IN_PRESENT_VALUE	Binary input present value: TRUE / FALSE, (1/0), during function block call, the value of particular binary input, optionable logic variable or constant TRUE / FALSE can be assigned .
IN_ALARM	Assignment of failure input (alarm) from optionable source, e.g. Ssystem variable, optionable logic variable etc.
IN_FAULT	Assignment of failure input (fault) from optionable source, e.g. Ssystem variable, optionable logic variable etc.
OUT_OUT_OF_SERVICE	OUT_OUT_OF_SERVICE = TRUE, disconnection of function block input IN_PRESENT_VALUE, on the function block output OUT_PRESENT_VALUE is value from internal variable *Preset_value*, which can be transcribed to the value TRUE / FALSE e.g. Because of application testing etc.
OUT_PRESENT_VALUE	If OUT_OUT_OF_SERVICE = TRUE, then OUT_PRESENT_VALUE = IN_PRESENT_VALUE. If OUT_OUT_OF_SERVICE = FALSE, then OUT_PRESENT_VALUE = internal variable *Preset_value.*

3.5. BINARY OUTPUT

Object binary output is used for:

- detection of PLC binary ouput
- detection of binary output control error (set by the user program)
- simulation of binary output status (setup manually to the value TRUE/FALSE, (1/0), for testing)

Table 5.1: object binary output and its features

Binary output	Description
Object Identifier	Object type identifier: 0..maximum number of PLC binary outputs visible in BACnet (typically 32) see further-configuration in MOSAIC environment
Object Name	Object name : object name, e.g. BINARY_OUTPUT_0
Object Type	Object type: BINARY_OUTPUT = 4
Description	Object description: e.g. Description of BINARY_OUTPUT_0
Present Value	Present value of binary output: TRUE / FALSE, (1/0)
Status Flags	Input status: OUT OF SERVICE=1 if the requirement is OUT OF SERVICE
Event State	Always EVENT_STATE_NORMAL
Out of service	1 = OUT OF SERVICE, disconnection of function block output to Preset Value, valid is the value from Preset Value, which can be transcribed to the value TRUE / FALSE
Polarity	Always POLARITY_NORMAL

Declaration of **BACNET_TYP_FB_BO** type – BINARY OUTPUT

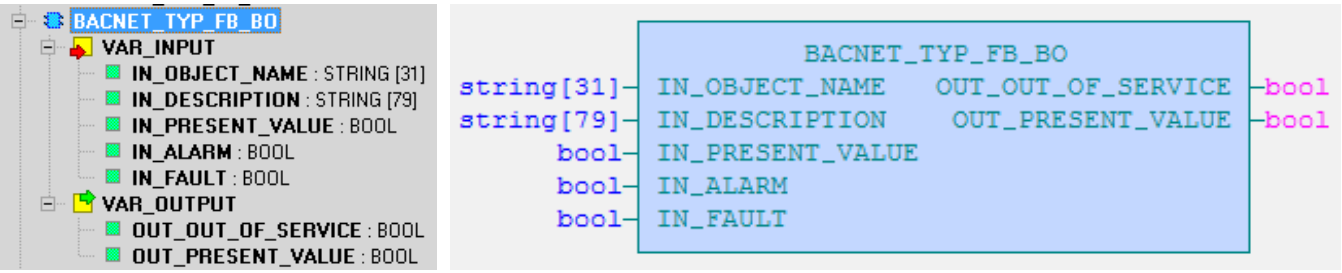


Table 5.2: Signification of BACNET_TYP_FB_BO data type items

Binary output	Description
IN_OBJECT_NAME	Object name: e.g. BINARY_OUTPUT_0, VENTIL_V387, …
IN_DESCRIPTION	Object description: e.g. Description of BINARY_OUTPUT_0, Ventil V378 add, Ventil V387 reduce, etc.
IN_PRESENT_VALUE	Present value added by the function block to the binary output: TRUE / FALSE, (1/0), during the function block call, there can be optionable logic variable or constant TRUE / FALSE.
IN_ALARM	Assignment of failure input (alarm) from optionable sourse, e.g. System variable, optionable logic variable, etc.
IN_FAULT	Assignment of failure input (fault) rom optionable sourse, e.g. System variable, optionable logic variable, etc.
OUT_OUT_OF_SERVICE	OUT_OUT_OF_SERVICE = TRUE, function block input disconnection IN_PRESENT_VALUE, on function block output OUT_PRESENT_VALUE jis the value from internal variable *Preset_value*, which can be transcribed to the value TRUE / FALSE e.g. Because of application testing, etc.
OUT_PRESENT_VALUE	If OUT_OUT_OF_SERVICE = TRUE, then OUT_PRESENT_VALUE = IN_PRESENT_VALUE. If OUT_OUT_OF_SERVICE = FALSE, then OUT_PRESENT_VALUE = internal variable *Preset_value.*

3.6. BINARY VALUE

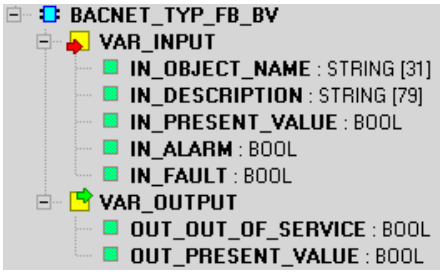
Object binary value is used for:

- detection or change of binary variable status in PLC
- detection of binary variable control error (set by the user program)
- simulation of binary variable status (setup manually to the value 1/0 for testing)

Table 6.1: object binary value and its features

Binary value	Description
Object Identifier	Object type identifier: 0..maximum number of PLC binary variables visible in BACnet (typically 32) see further – configuration in MOSAIC environment
Object Name	Object name: e.g. BINARY_VALUE_0
Object Type	Object type: BINARY_VALUE = 5
Description	Object description: e.g. Description of BINARY_OUTPUT_0
Present Value	Present value of binary variable: TRUE / FALSE, (1/0)
Status Flags	Input status: OUT OF SERVICE=1 if the requirement is on OUT OF SERVICE
Event State	Always EVENT_STATE_NORMAL
Out of service	1 = OUT OF SERVICE, disconnection of function block output to Preset Value, valid is the value from Preset Value, which can be transcribed to the value TRUE / FALSE

Declaration of **BACNET_TYP_FB_BV** type – BINARY VALUE



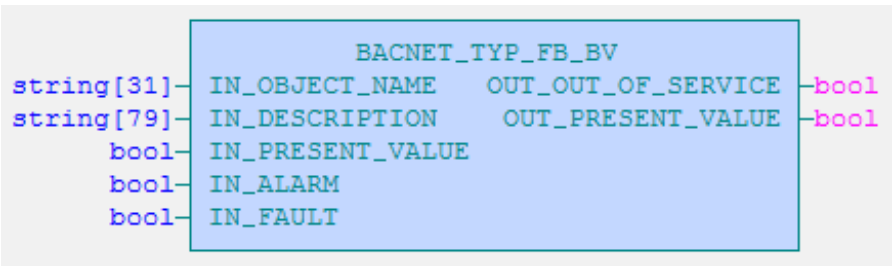


Table 6.2: Signification of BACNET_TYP_FB_BV data type items

Binary value	Description
IN_OBJECT_NAME	Object name: e.g. BINARY_VALUE_0, VENTIL_V387_MANUAL, …
IN_DESCRIPTION	Object description: e.g. Description of BINARY_VALUE_0, Ventil V378 manual control, etc.
IN_PRESENT_VALUE	Present value assigned by function block to the binary output: TRUE / FALSE, (1/0), during the function block call, there can be optionable logic variable or constant TRUE / FALSE.
IN_ALARM	Assignment of failure input (alarm) from optionable source, e.g. System variable, optionable logic variable, etc.
IN_FAULT	Assignment of failure input (fault) from optionable source, e.g. System variable, optionable logic variable, etc.
OUT_OUT_OF_SERVICE	OUT_OUT_OF_SERVICE = TRUE, function block input disconnection IN_PRESENT_VALUE, on function block output OUT_PRESENT_VALUE jis the value from internal variable *Preset_value*, which can be transcribed to the value TRUE / FALSE e.g. Because of application testing, etc.
OUT_PRESENT_VALUE	If OUT_OUT_OF_SERVICE = TRUE, then OUT_PRESENT_VALUE = IN_PRESENT_VALUE. If OUT_OUT_OF_SERVICE = FALSE, then OUT_PRESENT_VALUE = internal variable *Preset_value.*

3.7. ANALOG INPUT

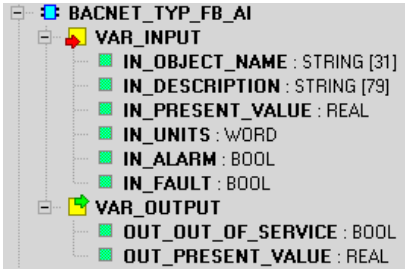
Object analog input is used for:

- detection of PLC analog input status
- detection of analog input control error (set by the user program)
- simulation of analog input statue (setup manually to the value in the range REAL for testing)

Table 7.1: analog input and its feature

Analog input	Description
Object Identifier	Object identifier: 0..maximum number of PLC analog inputs visible in BACnet (typically 32) see further - configuration in MOSAIC environment
Object Name	Object name: e.g. ANALOG_INPUT_0
Object Type	Object type: ANALOG_INPUT = 0
Description	Object description: e.g. Description of ANALOG_INPUT_0
Present Value	Present value of analog input.
Units	Used engineer units: e.g.[mA]
Status Flags	Input status: OUT OF SERVICE=1 if the requirement is on OUT OF SERVICE
Event State	Always EVENT_STATE_NORMAL
Out of service	1 = OUT OF SERVICE, disconnection of the physical input connector, valid is the value from Preset Value, which can be transcribed to the value in the range ???

Declaration of **BACNET_TYP_FB_AI** type – ANALOG INPUT



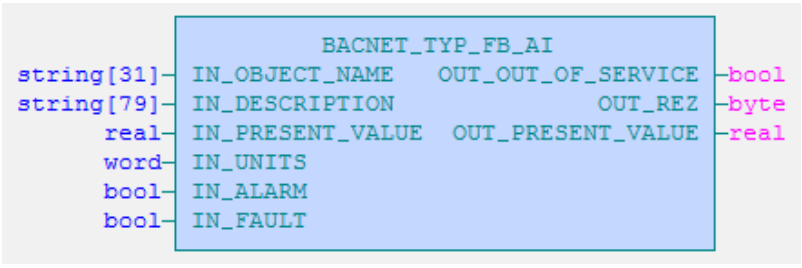


Table 7.2: signification of BACNET_TYP_FB_AI data type items

Analog input	Description
IN_OBJECT_NAME	Object name: e.g. ANALOG_INPUT_0, VENTIL_V387_POSITION, …
IN_DESCRIPTION	Object description: e.g. Description of ANALOG_INPUT_0, Ventil V378 actual possition, etc.
IN_PRESENT_VALUE	Present value assigned by function block from the analog input: range REAL, during the function block call, there can be optionable logic variable or constant of REAL type.
IN_ALARM	Assignment of failure input (alarm) from optionable source, e.g. System variable, optionable logic variable, etc.
IN_FAULT	Assignment of failure input (fault) from optionable source, e.g. System variable, optionable logic variable, etc.
OUT_OUT_OF_SERVICE	OUT_OUT_OF_SERVICE = TRUE, function block input disconnection IN_PRESENT_VALUE, on function block output OUT_PRESENT_VALUE jis the value from internal variable *Preset_value*, which can be transcribed to the value of REAL type e.g. Because of application testing, etc..
OUT_PRESENT_VALUE	If OUT_OUT_OF_SERVICE = TRUE, then OUT_PRESENT_VALUE = IN_PRESENT_VALUE. If OUT_OUT_OF_SERVICE = FALSE, then OUT_PRESENT_VALUE = internal variable *Preset_value.*

3.8. ANALOG OUTPUT

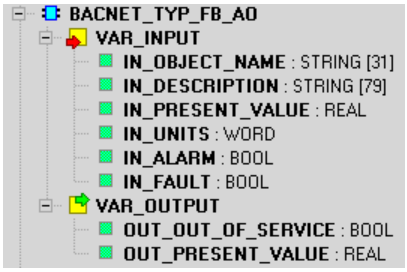
Object analog output is used for:

- detection of PLC analog output status
- detection of analog output control error (set by the user program)
- simulation of analog output status (setup manually to the value in range REAL for testing)

Table 8.1: object analog output and its features

Analog output	Description
Object Identifier	Object type identifier: 0..maximum number of PLC analog ouputs visible in BACnet (typically 32) see further – configuration in MOSAIC environment
Object Name	Object name: e.g. ANALOG_OUTPUT_0
Object Type	Object type: ANALOG_OUTPUT = 1
Description	Object description: e.g. Description of ANALOG_OUTPUT_0
Present Value	Present value of analog output
Units	Used engineer units: e.g.[℃]
Status Flags	Input status: OUT OF SERVICE=1 if the requirement is on OUT OF SERVICE
Event State	Always EVENT_STATE_NORMAL
Out of service	1 = OUT OF SERVICE, disconnection of the physical input connector, valid is the value from Preset Value, which can be transcribed to the value in the range ???

Declaration of **BACNET_TYP_FB_AO** type – ANALOG OUTPUT



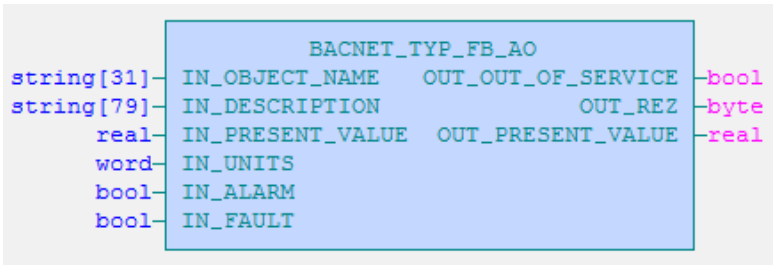


Table 8.2: Signification of BACNET_TYP_FB_AO data type items

Analog output	Description
IN_OBJECT_NAME	Object name: e.g. ANALOG_OUTPUT_0, VENTIL_V387_POSITION, …
IN_DESCRIPTION	Object description: e.g. Description of ANALOG_OUTPUT_0, Ventil V378 required possition, etc.
IN_PRESENT_VALUE	Present value assigned by function block from the analog output: range REAL, during the function block call, there can be optionable logic variable or constant of REAL type.
IN_ALARM	Assignment of failure input (alarm) from optionable source, e.g. System variable, optionable logic variable, etc.
IN_FAULT	Assignment of failure input (fault) from optionable source, e.g. System variable, optionable logic variable, etc.
OUT_OUT_OF_SERVICE	OUT_OUT_OF_SERVICE = TRUE, function block input disconnection IN_PRESENT_VALUE, on function block output OUT_PRESENT_VALUE jis the value from internal variable *Preset_value*, which can be transcribed to the value of REAL type e.g. Because of application testing, etc.
OUT_PRESENT_VALUE	If OUT_OUT_OF_SERVICE = TRUE, then OUT_PRESENT_VALUE = IN_PRESENT_VALUE. If OUT_OUT_OF_SERVICE = FALSE, then OUT_PRESENT_VALUE = internal variable *Preset_value.*

3.9. ANALOG VALUE

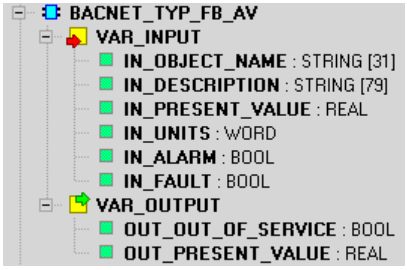
Object analog value is used for:

- detection or change of PLC analog variable status (type single)
- detection of analog variable control error (set by the user program)
- simulation of analog variable status (rsetup manually to the value in range of REAL type, for testing)

Table 8.1: object analog value and its features

Analog value	Description
Object Identifier	Object type identifier: 0..maximum number of PLC analog values visible in BACnet (typically 32) see further – configuration in MOSAIC environment
Object Name	Object name: e.g. ANALOG_VALUE_0
Object Type	Object type: ANALOG_VALUE = 2
Description	Object description: e.g. Description of ANALOG_VALUE_0
Present Value	Present value of analog value
Units	Used engineer units: e.g.[%]
Status Flags	Input status: OUT OF SERVICE=1 if the requirement is on OUT OF SERVICE
Event State	Always EVENT_STATE_NORMAL
Out of service	1 = OUT OF SERVICE, disconnection of the physical input connector, valid is the value from Preset Value, which can be transcribed to the value in the range ???

Declaration of **BACNET_TYP_FB_AV** type – ANALOG VALUE



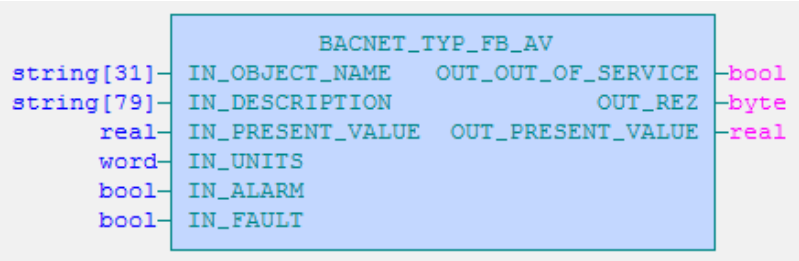


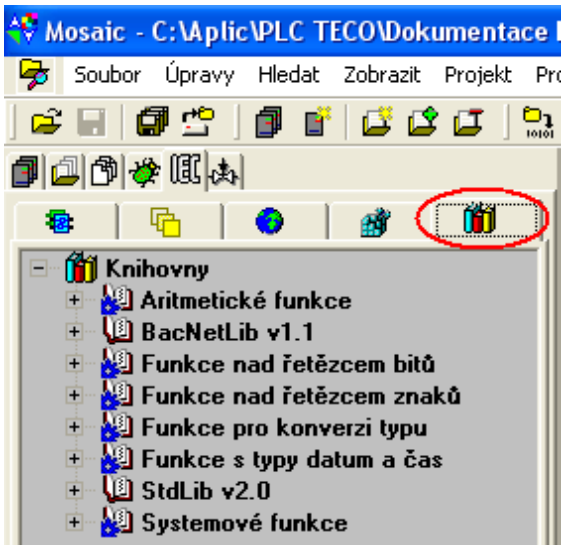
Table 9.2: Signification of BACNET_TYP_FB_AV data type items

Analog value	Description
IN_OBJECT_NAME	Object name: e.g. ANALOG_OBJECT_0, VENTIL_V387_MAX_POSITION, …
IN_DESCRIPTION	Object description: e.g. Description of ANALOG_VALUE_0, Ventil V378 maximum possition, etc.
IN_PRESENT_VALUE	Present value assigned by function block from the analog value: range REAL, during the function block call, there can be optionable logic variable or constant of REAL type.
IN_ALARM	Assignment of failure input (alarm) from optionable source, e.g. System variable, optionable logic variable, etc.
IN_FAULT	Assignment of failure input (fault) from optionable source, e.g. System variable, optionable logic variable, etc.
OUT_OUT_OF_SERVICE	OUT_OUT_OF_SERVICE = TRUE, function block input disconnection IN_PRESENT_VALUE, on function block output OUT_PRESENT_VALUE jis the value from internal variable *Preset_value*, which can be transcribed to the value of REAL type e.g. Because of application testing, etc.
OUT_PRESENT_VALUE	If OUT_OUT_OF_SERVICE = TRUE, then OUT_PRESENT_VALUE = IN_PRESENT_VALUE. If OUT_OUT_OF_SERVICE = FALSE, then OUT_PRESENT_VALUE = internal variable *Preset_value.*

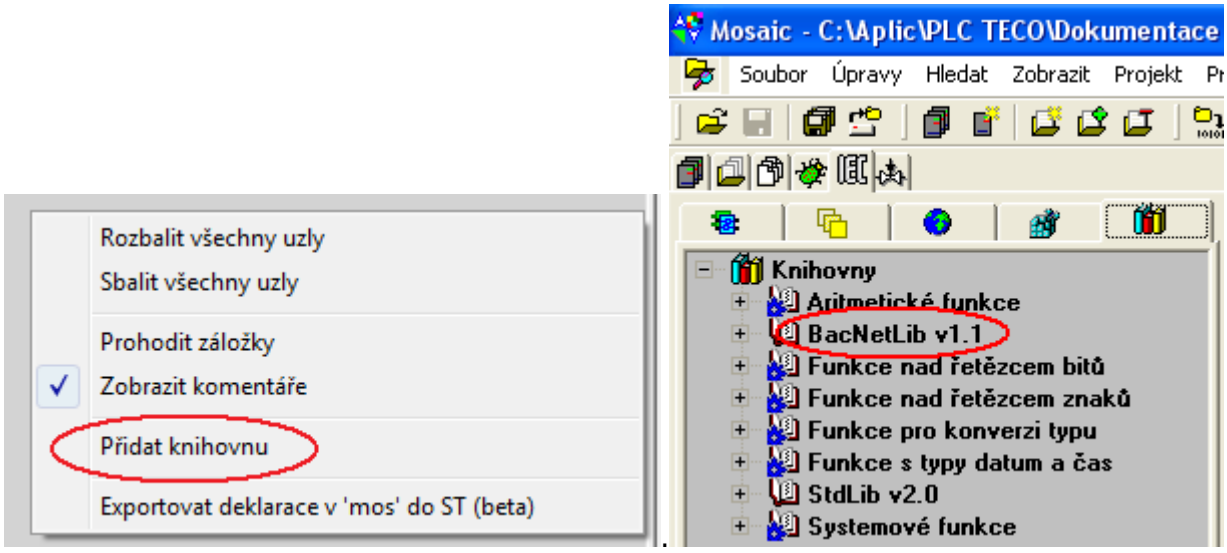
4. CONFIGURATION OF MOSAIC ENVIRONMENT AND BACnet SERVICES (from version 2.0.19.0)

4.1 CONFIGURATION OF MOSAIC ENVIRONMENT

BACnet service can be used for installation of the library BacNetLib v1.1 and higher. Installation is undertaken using the library manager.



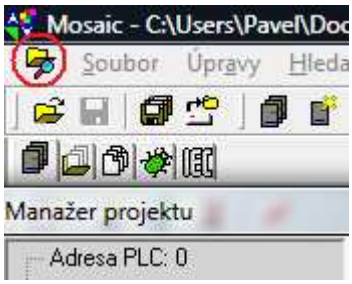
After right-mouse click, the dialogue will appear and we select the item Add library and undertake the installation itself by selection of library BacNetLib in appropriate folder. If the installation is succesful, the library list will contain the library BacNetLib of the corresponding version.

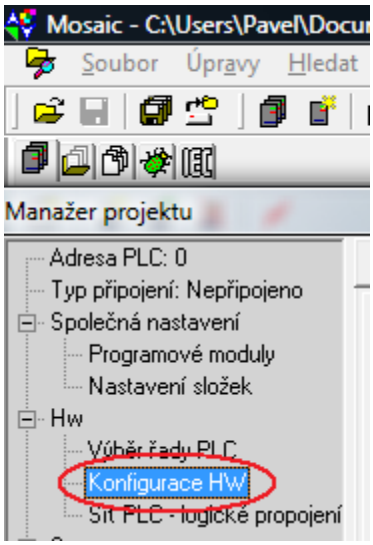


4.2 CONFIGURATION OF BACnet SERVICE WITHIN THE PLC TECOMAT

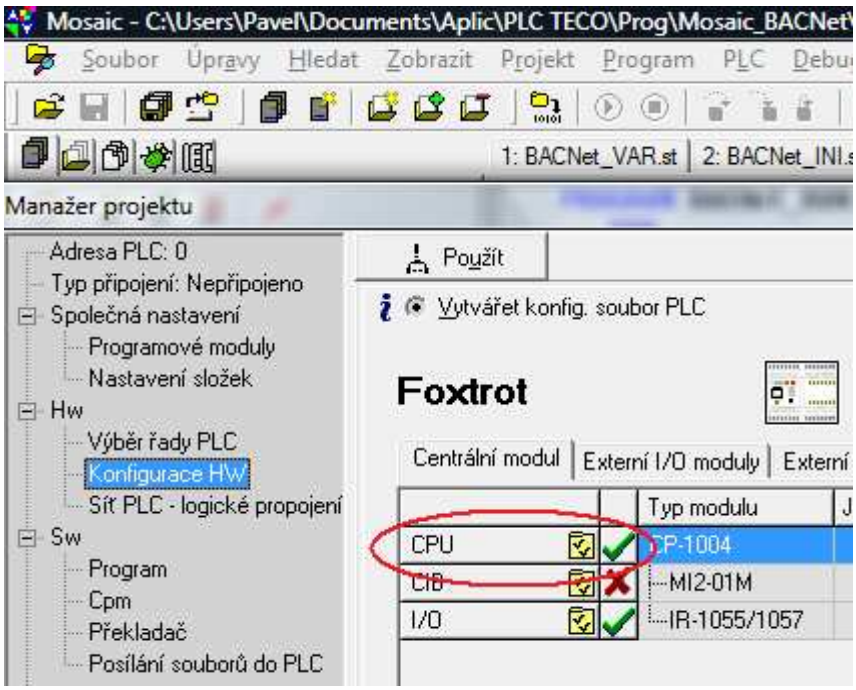
BACnet service is neccessary to be configured prior to its usage. The configuration is done in the MOSAIC environment (from version 2.0.19.0). BACnet service „is running “ above the communication channel of CPU PLC.

4.2.1 Select menu „Project manager“ and subsequently the item HW configuration

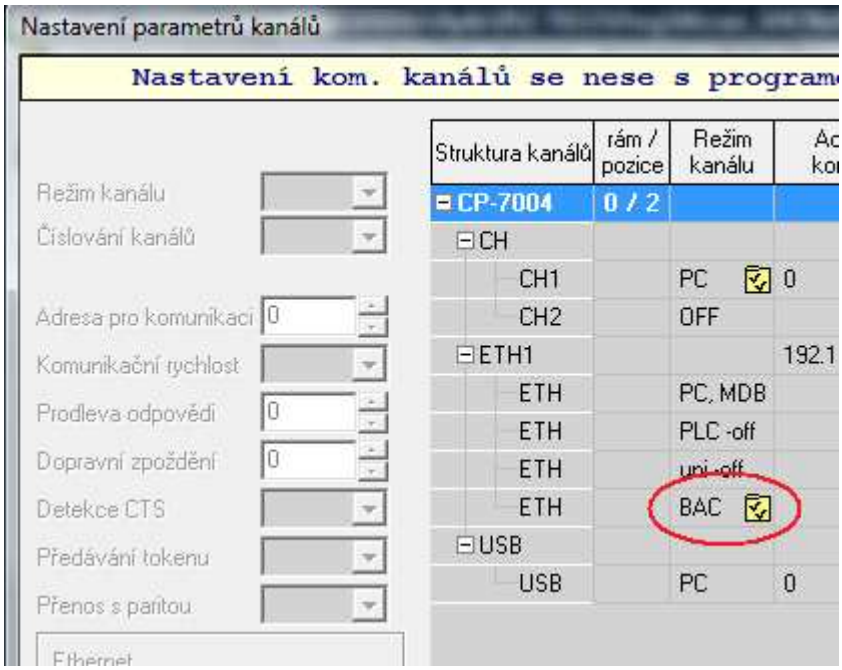




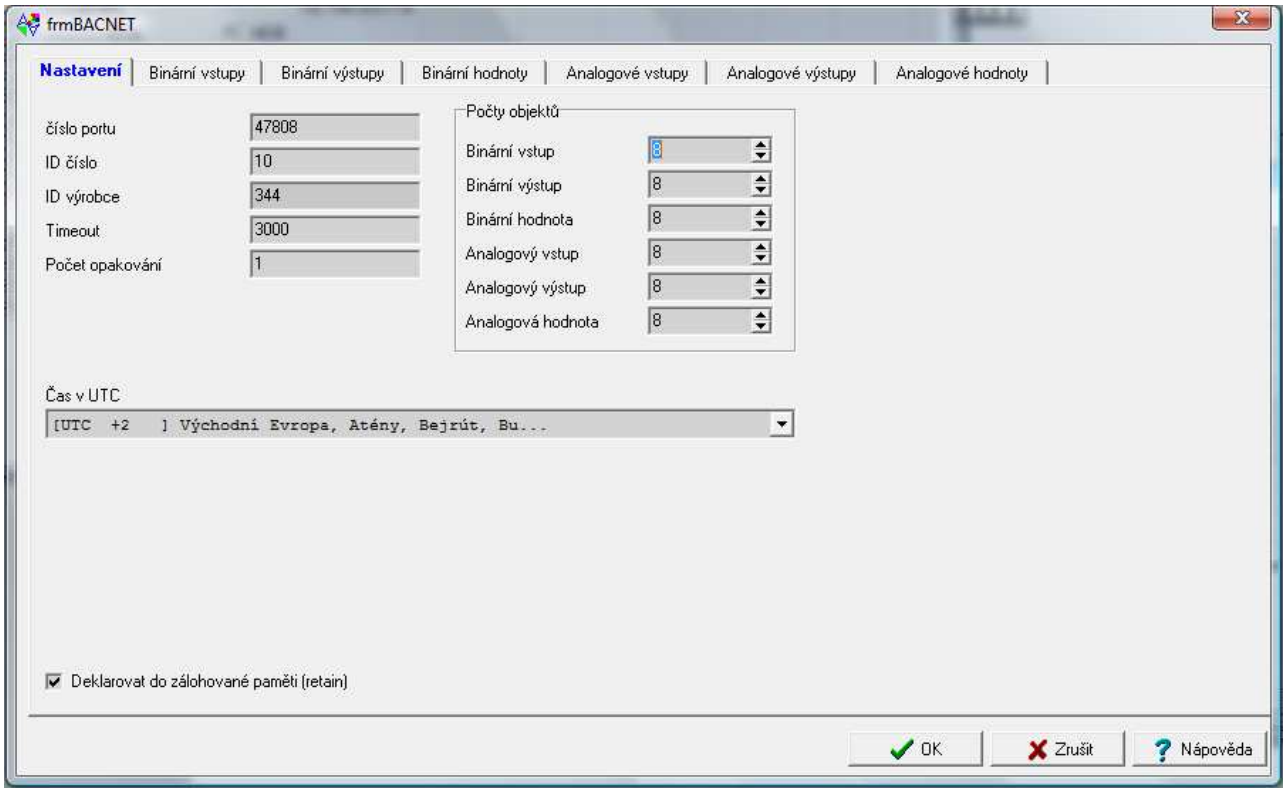
4.2.2 Select the item CPU in Central module bookmark



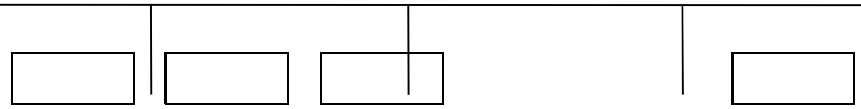
4.2.3 Select item mode of BAC channel



4.2.4 Now, the dialoque window appears for BACnet service configuration

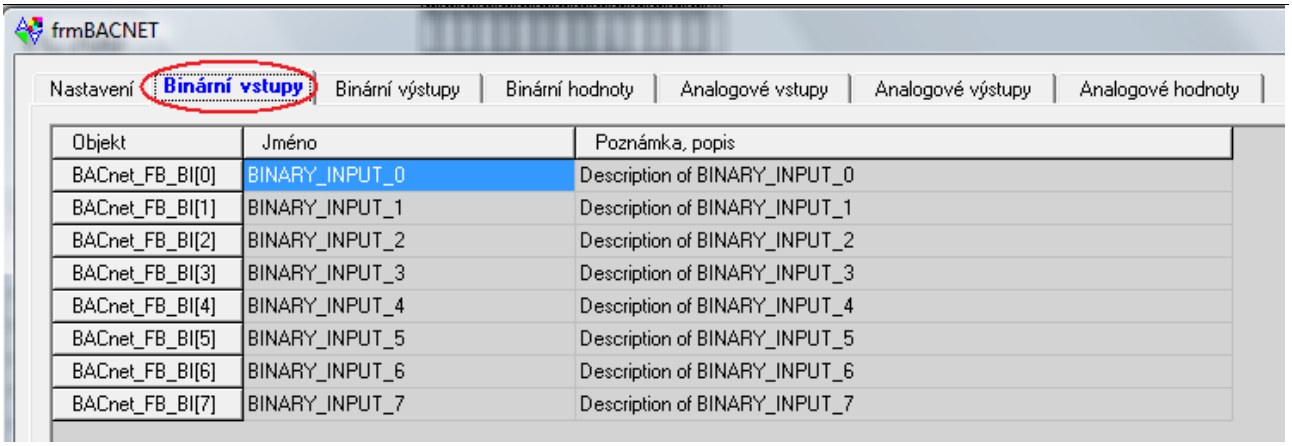


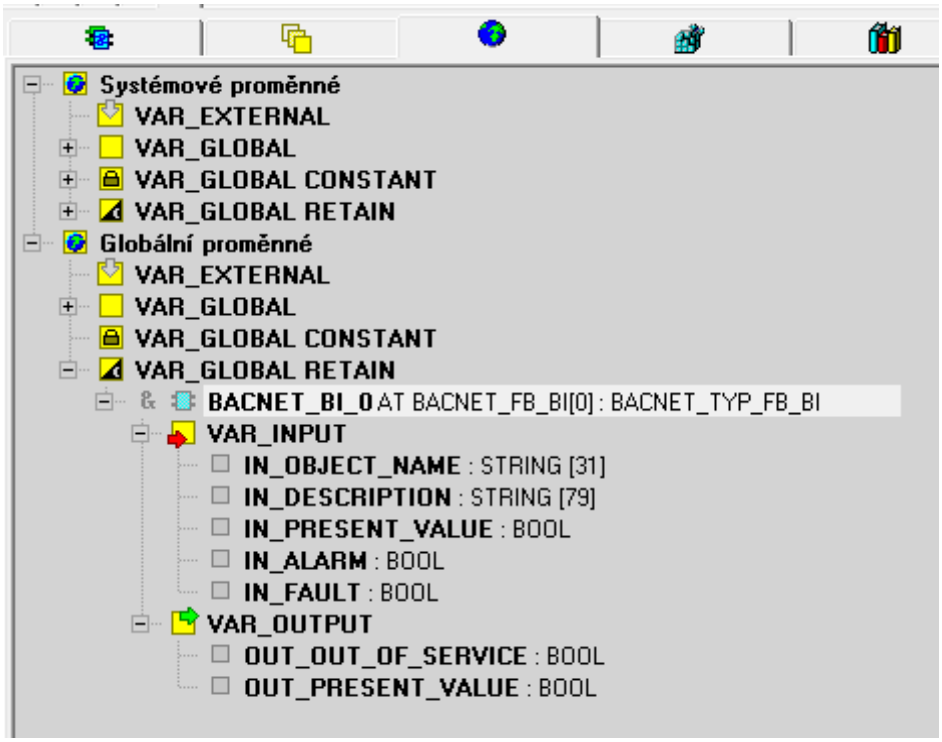
4.2.5 Setup of BACnet service parameters

Parameter	Description
Port number	Typically 47808 = 0xBAC0, basic port number for BACnet service, within own local network a different number can be used but it must be same for all units within the network !
ID number	Identification number of the unit in the network = address within BACnet network.
	
ID producer	344 – Teco a.s.
Timeout	Response idle period.
Cycling number	Number of repetitions of communication cycles when communication error occurs.
Time in UTC	Time zone toward UTC.
Declare to the backup memory (retain)	Declaration to the backup memory ensures the remembering of set parameters of BACnet objects also after PLC switch off/on. Such situation can occure when object parameters (their description for example) is set from the operator´s console and we want this change to be „remembered“ after PLC switch off/on, also.
Number of objects	Assign the number of BACnet objects available in the user program. The number of each object is limited according to the version of the CPU PLC firmware. Typically, the maximum number for each type of objects is 32.

4.2.6 Setup of individual object types parameters

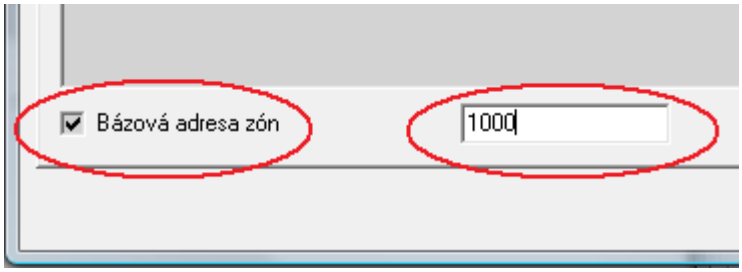
In objects Binary inputs, Binary outputs, Binary values, Analog inputs, Analog outputs, Analog values, we can set name, description and eventually physical units if the value is the physical unit. Each object type has its bookmark.



Item	Description
Name	Symbolic name of BACnet object which is presented by the function block in the user program. This name is further operated with in the user program as with the global variable. The variable is accessible via the Variable manager. 
Note, description	Detailed object description, accessible from the user program to other stations within the BACnet network
Units	Physical units, selection from the list available

4.2.7 Setup of zone base address

During the definition of parameters of objects Binary inputs, Binary outputs, Binary value, Analog inputs, Analog outputs, Analog value, their location in the notepad can be set. Select the option Zone base address and set the offset of the zone beginning (i.e. Address of the first object in the notepad).



4.2.8 Configuration check of BACnet objects

After the completion of configuration of BACnet service and objects, we compile the program. After faultless compilation, we can see and check in the variable manager window declared BACnet objects.

5. THE USE OF BACnet OBJECTS IN THE USER PROGRAM

5.1 BACnet OBJECT AS A FUNCTION BLOCK

After configuration of BACnet service and objects and successful program compilation, we can use these objects in the PLC user program. Objects are represented by function blocks. Declaration of function blocks is available in the library.

Example of declaration of function block FUNCTION_BLOCK BACNET_TYP_FB_BI that is represented by the object binary input in BACnet.

```
FUNCTION_BLOCK BACNET_TYP_FB_BI
  VAR_INPUT
    IN_OBJECT_NAME : string [31];
    IN_DESCRIPTION : string [79];
    IN_PRESENT_VALUE {ALIGNED} : bool;
    IN_ALARM {ALIGNED} : bool;
    IN_FAULT {ALIGNED} : bool;
  END_VAR
  VAR_OUTPUT
    OUT_OUT_OF_SERVICE {ALIGNED} : bool;
    OUT_PRESENT_VALUE {ALIGNED} : bool;
  END_VAR
  VAR
    BAC_OUT_OF_SERVICE {ALIGNED} : bool;
    BAC_NORMAL {ALIGNED} : bool;
    BAC_ALARM {ALIGNED} : bool;
    BAC_FAULT {ALIGNED} : bool;
    BAC_PRESENT_VALUE {ALIGNED} : bool;
    BAC_EVENT_STATE : byte;
  END_VAR
END_FUNCTION_BLOCK
```

5.2 AUTOGENERATION OF BACnet OBJECTS

When the compilation is successful, the MOSAIC environment autogenerates declarations in the user program, see the following extract.

```
VAR_GLOBAL CONSTANT
  BACNET_MAX_BINARY_INPUTS    : UINT := 0;
  BACNET_MAX_BINARY_OUTPUTS   : UINT := 0;
  BACNET_MAX_BINARY_VALUES    : UINT := 3;
  BACNET_MAX_ANALOG_INPUTS    : UINT := 0;
  BACNET_MAX_ANALOG_OUTPUTS   : UINT := 0;
  BACNET_MAX_ANALOG_VALUES    : UINT := 2;
END_VAR

VAR_GLOBAL
  BACNET_FB_BV : ARRAY[0..BACNET_MAX_BINARY_VALUES-1] OF T_BACNET_TYP_FB_BV := [
        ( IN_OBJECT_NAME := 'BACO_BV_CTUD1_Reset', IN_DESCRIPTION := 'Reset of the dual counter' ),
```

```
        ( IN_OBJECT_NAME := 'BACO_BV_CTUD1_Up', IN_DESCRIPTION := 'Counter output, preselection
reached, output = TRUE otherwise FALSE' ),
        ( IN_OBJECT_NAME := 'BACO_BV_CTUD1_Down', IN_DESCRIPTION := 'Counter output, real value = 0
=> output TRUE, otherwise FALSE' ) ];

  BACNET_FB_AV : ARRAY[0..BACNET_MAX_ANALOG_VALUES-1] OF T_BACNET_TYP_FB_AV := [
        ( IN_OBJECT_NAME := 'BACO_AV_CTUD1_PresentValue', IN_DESCRIPTION := 'Counter preselection',
IN_UNITS := BACNET_UNITS_NO_BACNET_UNITS ),
        ( IN_OBJECT_NAME := 'BACO_AV_CTUD1_CurrentValue', IN_DESCRIPTION := 'Real value of the
counter', IN_UNITS := BACNET_UNITS_NO_BACNET_UNITS ) ];

  BACO_BV_CTUD1_Reset         AT BACNET_FB_BV[0] : BACNET_TYP_FB_BV;
  BACO_AV_CTUD1_PresentValue  AT BACNET_FB_AV[0] : BACNET_TYP_FB_AV;
  BACO_BV_CTUD1_Up            AT BACNET_FB_BV[1] : BACNET_TYP_FB_BV;
  BACO_AV_CTUD1_CurrentValue  AT BACNET_FB_AV[1] : BACNET_TYP_FB_AV;
  BACO_BV_CTUD1_Down          AT BACNET_FB_BV[2] : BACNET_TYP_FB_BV;
END_VAR
```

5.3 BACnet OBJECTS CONTROL IN THE USER PROGRAM

BACnet objects control by the user program means the call of function blocks which represent these objects. This call is not generated by the development environment MOSAIC, it is fully under control of the application programmer.

Example of the call of function block BINARY_INPUT_0 that represents the object binary inputs BACnet.

```
VAR_GLOBAL
  BUT_BI0: BOOL;            // Tlačítko
END_VAR

BINARY_INPUT_0( IN_PRESENT_VALUE := BUT_BI0);
atd.
BINARY_OUTPUT_0 ( IN_PRESENT_VALUE := BUT_BO0 );
BINARY_VALUE_0  ( IN_PRESENT_VALUE := BUT_BV0 );
ANALOG_INPUT_0  ( IN_PRESENT_VALUE := BUT_AI0 );
ANALOG_OUTPUT_0 ( IN_PRESENT_VALUE := BUT_AO0 );
ANALOG_VALUE_0  ( IN_PRESENT_VALUE := BUT_AV0 );
```

Declaration of function block binary inputs and demonstration of usage of input parameter **IN_PRESENT_VALUE.**

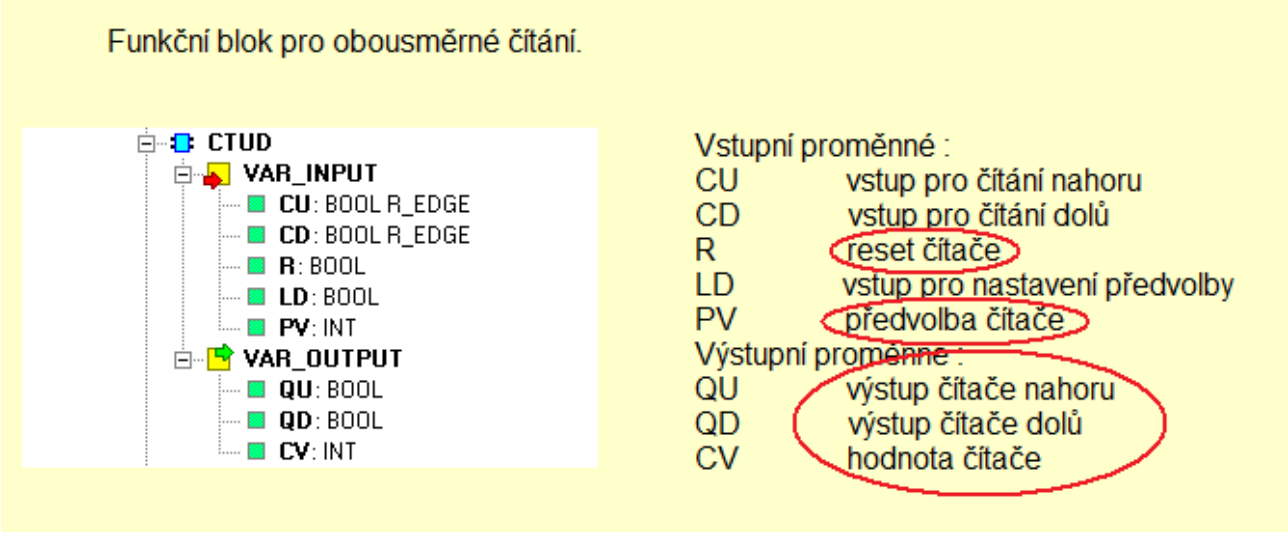
```
BINARY_INPUT_0( IN_PRESENT_VALUE := BUT_BI0);

FUNCTION_BLOCK BACNET_TYP_FB_BI
  VAR_INPUT
    IN_OBJECT_NAME : string [31];
    IN_DESCRIPTION : string [79];
    IN_PRESENT_VALUE {ALIGNED} : bool;
    IN_ALARM {ALIGNED} : bool;
    IN_FAULT {ALIGNED} : bool;
  END_VAR
  VAR_OUTPUT
    OUT_OUT_OF_SERVICE {ALIGNED} : bool;
    OUT_PRESENT_VALUE {ALIGNED} : bool;
  END_VAR
  VAR
    BAC_OUT_OF_SERVICE {ALIGNED} : bool;
    BAC_NORMAL {ALIGNED} : bool;
    BAC_ALARM {ALIGNED} : bool;
    BAC_FAULT {ALIGNED} : bool;
```

```
      BAC_PRESENT_VALUE {ALIGNED} : bool;
      BAC_EVENT_STATE : byte;
    END_VAR
  END_FUNCTION_BLOCK
```

5.4 EXAMPLES OF USER PROGRAMS WITH BACnet OBJECTS

5.4.1 Example of control of counter parameters using the BACnet objects

Lets suppose the task where we need to set the preselection in dual counter, reset the counter and watch the real value and ouput of the counter. We require these parameters to be controllable and visible in BACnet environment, meaning, that they can be operated and visible from the operator´s console or other control systems with the BACnet protocol. This situation is shown in the following picture.



Example of the program with the counter CTUD without „control“ via BACnet objects

```
VAR_GLOBAL
  pulseUP     : BOOL;
  pulseDOWN   : BOOL;
  resetCTUD   : BOOL;
  limitUP     : BOOL;
  limitDOWN   : BOOL;
  presentCTUD : INT;
  counterCTUD : CTUD;
END_VAR

PROGRAM Example_CTUD1

  counterCTUD(
    CU := pulseUP,
    CD := pulseDOWN,
    R  := resetCTUD,
    PV := presentCTUD,
    QU => limitUP,
    QD => limitDOWN);

END_PROGRAM
```

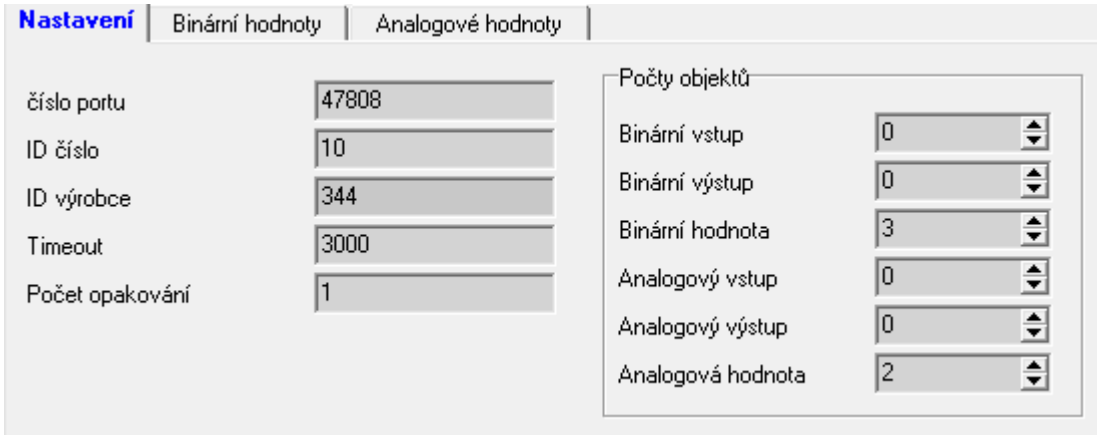
Now, we undertake declarations of BACnet objects.
Binary values:

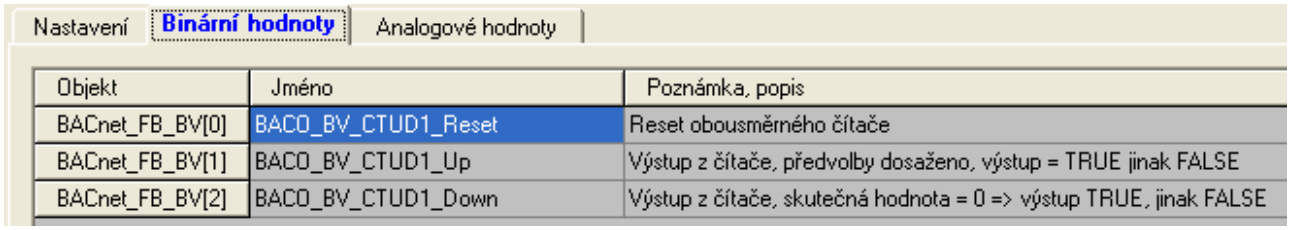
- counter reset
- counter output up
- counter output down
  total 3

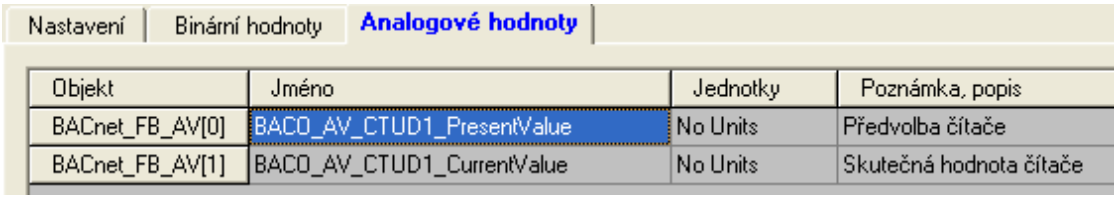
Analog values:

- counter preselection
- counter value
  total 2

Declaration is undertaken according to steps shown in chapter 4.2.



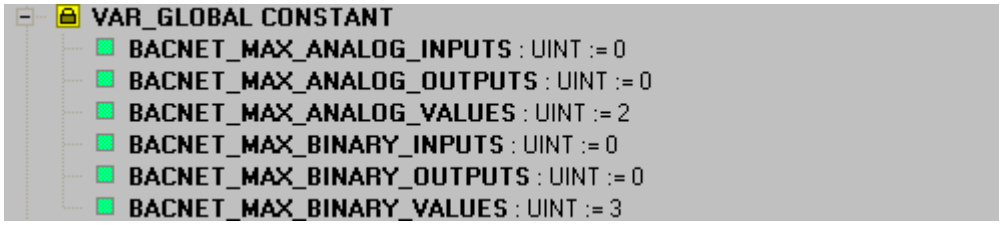


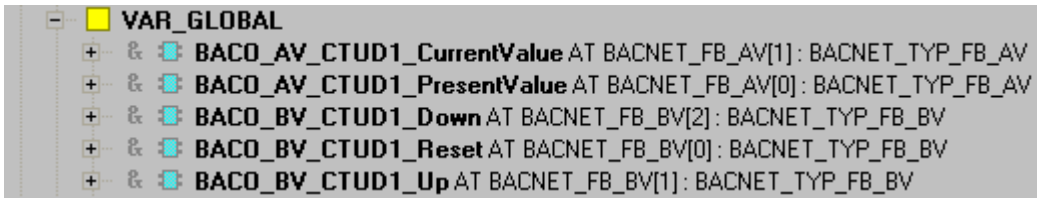


Note:
It is recommended to pay attention to BACnet object names. Select names so, that it is evident that they are BACnet objects, which function block in your program do they belong to and what do they affect.

E.g. BACO_BV_CTUD1_Reset

After successful compilation, we can see, in the variable inspector, our declared BACnet objects.





Furthermore, it is autocreated by the MOSAIC environment the file HWConfig.ST which contains necessary declarations, see bellow. Interesting is the last part of the declaration that declares our defined objects which can be used in the user program.

```
VAR_GLOBAL CONSTANT
  BACNET_MAX_BINARY_INPUTS    : UINT := 0;
  BACNET_MAX_BINARY_OUTPUTS   : UINT := 0;
  BACNET_MAX_BINARY_VALUES    : UINT := 3;
  BACNET_MAX_ANALOG_INPUTS    : UINT := 0;
  BACNET_MAX_ANALOG_OUTPUTS   : UINT := 0;
  BACNET_MAX_ANALOG_VALUES    : UINT := 2;
END_VAR

VAR_GLOBAL
  BACNET_FB_BV : ARRAY[0..BACNET_MAX_BINARY_VALUES-1] OF T_BACNET_TYP_FB_BV := [
        ( IN_OBJECT_NAME := 'BACO_BV_CTUD1_Reset', IN_DESCRIPTION := 'Reset of dual counter' ),
        ( IN_OBJECT_NAME := 'BACO_BV_CTUD1_Up', IN_DESCRIPTION := 'Counter output, preselection
reached, output = TRUE otherwise FALSE' ),
        ( IN_OBJECT_NAME := 'BACO_BV_CTUD1_Down', IN_DESCRIPTION := 'Counter output, real value = 0
=> output TRUE, otherwise FALSE' ) ];

  BACNET_FB_AV : ARRAY[0..BACNET_MAX_ANALOG_VALUES-1] OF T_BACNET_TYP_FB_AV := [
        ( IN_OBJECT_NAME := 'BACO_AV_CTUD1_PresentValue', IN_DESCRIPTION := 'Counter preselection',
IN_UNITS := BACNET_UNITS_NO_BACNET_UNITS ),
        ( IN_OBJECT_NAME := 'BACO_AV_CTUD1_CurrentValue', IN_DESCRIPTION := 'Real counter value',
IN_UNITS := BACNET_UNITS_NO_BACNET_UNITS ) ];

BACO_BV_CTUD1_Reset          AT BACNET_FB_BV[0] : BACNET_TYP_FB_BV;
BACO_AV_CTUD1_PresentValue   AT BACNET_FB_AV[0] : BACNET_TYP_FB_AV;
BACO_BV_CTUD1_Up             AT BACNET_FB_BV[1] : BACNET_TYP_FB_BV;
BACO_AV_CTUD1_CurrentValue   AT BACNET_FB_AV[1] : BACNET_TYP_FB_AV;
BACO_BV_CTUD1_Down           AT BACNET_FB_BV[2] : BACNET_TYP_FB_BV;

END_VAR
```

Example of program modification with the counter CTUD with „contol“ via BACnet objects

```
VAR_GLOBAL
  pulseUP     : BOOL;
  pulseDOWN   : BOOL;
  resetCTUD   : BOOL;
  limitUP     : BOOL;
  limitDOWN   : BOOL;
  presentCTUD : INT;

  counterCTUD : CTUD;
END_VAR
```

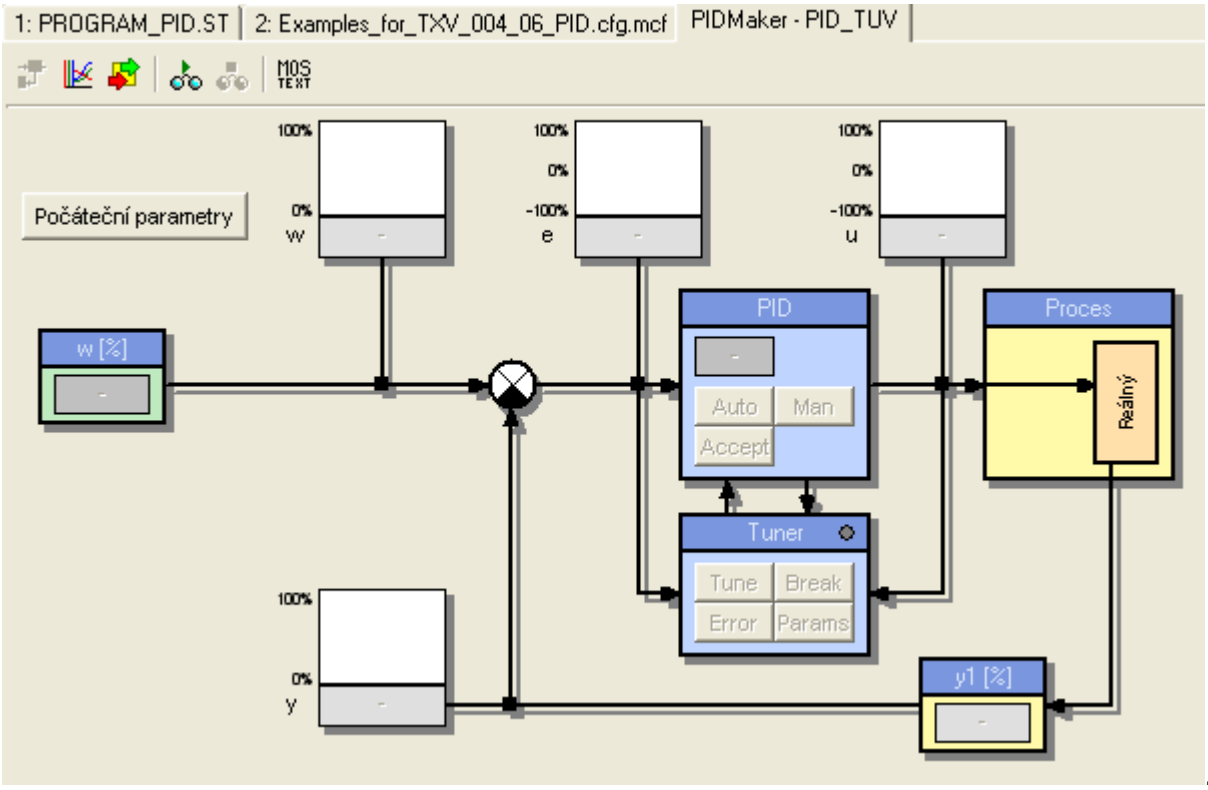
```
PROGRAM Example_CTUD1
  BACO_AV_CTUD1_CurrentValue();
  BACO_AV_CTUD1_PresentValue();
  BACO_BV_CTUD1_Down();
  BACO_BV_CTUD1_Reset();
  BACO_BV_CTUD1_Up();

  // Impletion of parameters from BACnet objects
  resetCTUD    := BACO_BV_CTUD1_Reset.OUT_PRESENT_VALUE;
  presentCTUD  := REAL_TO_INT(BACO_AV_CTUD1_PresentValue.OUT_PRESENT_VALUE);

  counterCTUD(
    CU := pulseUP,
    CD := pulseDOWN,
    R  := resetCTUD,
    PV := presentCTUD,
    QU => limitUP,
    QD => limitDOWN);

  // acceptance of counter outputs to BACnet objects
  BACO_AV_CTUD1_CurrentValue.OUT_PRESENT_VALUE := INT_TO_REAL(counterCTUD.CV);
  BACO_BV_CTUD1_UP.OUT_PRESENT_VALUE           := limitUP;
  BACO_BV_CTUD1_DOWN.OUT_PRESENT_VALUE         := limitDOWN;

END_PROGRAM
```

Alternatively, we can the above shown code simlify so, that BACnet objects are used directly as parameters of function block counterCTUD.

```
VAR_GLOBAL
  pulseUP     : BOOL;
  pulseDOWN   : BOOL;
  counterCTUD : CTUD;
END_VAR

PROGRAM Example_CTUD1
  BACO_AV_CTUD1_CurrentValue();
  BACO_AV_CTUD1_PresentValue();
  BACO_BV_CTUD1_Down();
  BACO_BV_CTUD1_Reset();
  BACO_BV_CTUD1_Up();

  // the use of BACnet objects as function block parameters directly

  counterCTUD(
    CU := pulseUP,
    CD := pulseDOWN,
    R  := BACO_BV_CTUD1_Reset.OUT_PRESENT_VALUE,
    PV := REAL_TO_INT(BACO_AV_CTUD1_PresentValue.OUT_PRESENT_VALUE),
    QU => BACO_BV_CTUD1_UP.OUT_PRESENT_VALUE,
    QD => BACO_BV_CTUD1_DOWN.OUT_PRESENT_VALUE);

  // acceptance of counter output to BACnet object
  BACO_AV_CTUD1_CurrentValue.OUT_PRESENT_VALUE := INT_TO_REAL(counterCTUD.CV);

END_PROGRAM
```

5.4.2 Example of control of controller parameters via BACnet objects

Lets suppose the task where we need in the controller to be set the required value, watch real value, switch between manual and automatic control and during the manual control the controller output to be controlled. We require these parameters to be operated and visible in BACnet environment, meaning that they can be operated and visible form the operator´s console or from other control systems. This situation is demonstrated in the following picture (output form the tool PIDMaker which is a part of MOSAIC environment)



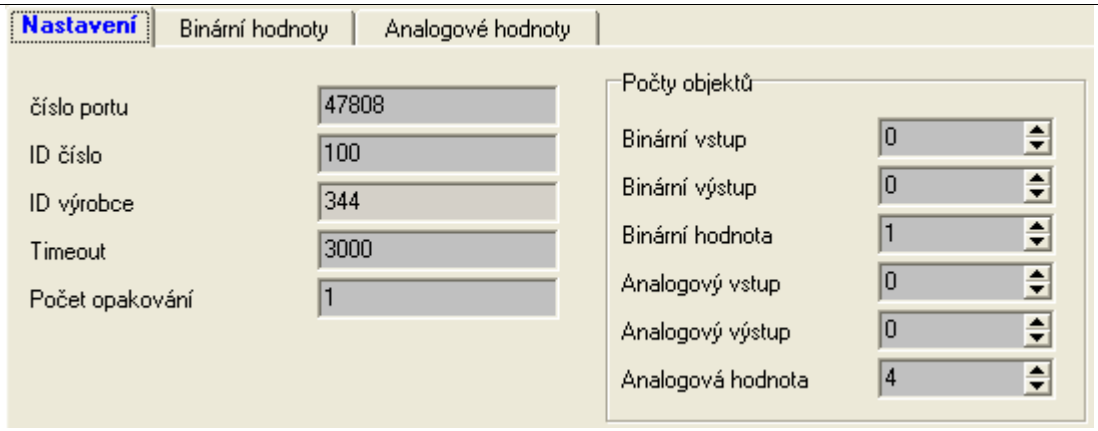
Now, we undertake declarations of BACnet objects.
Binary values:

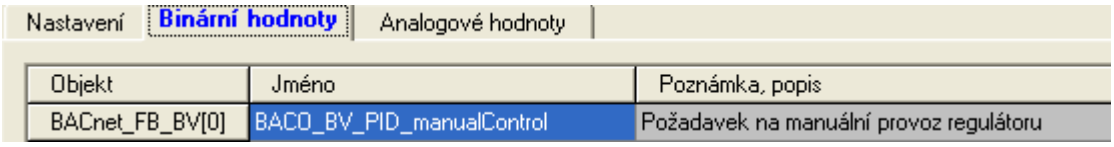
- requirement on manual operation of the controller (controller input)
  total 1

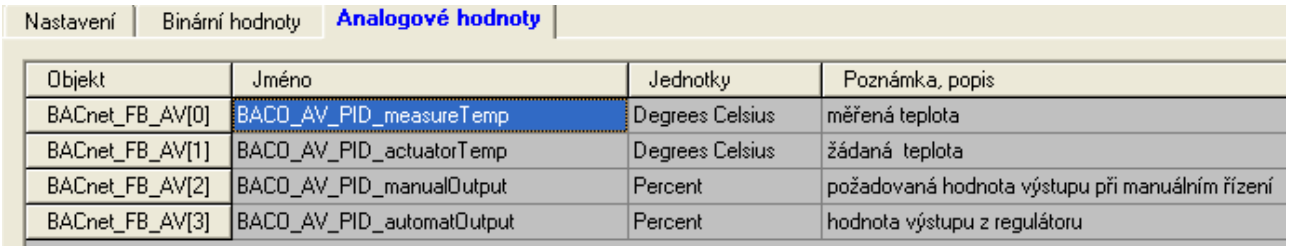
Analog values:

- measured temperature (controller input)
- required temperature (controller input)
- action intervention (controller output)
- required action intervention within the manual controller operation
  total 4

Declaration is undertaken according to steps described in chapter 4.2.





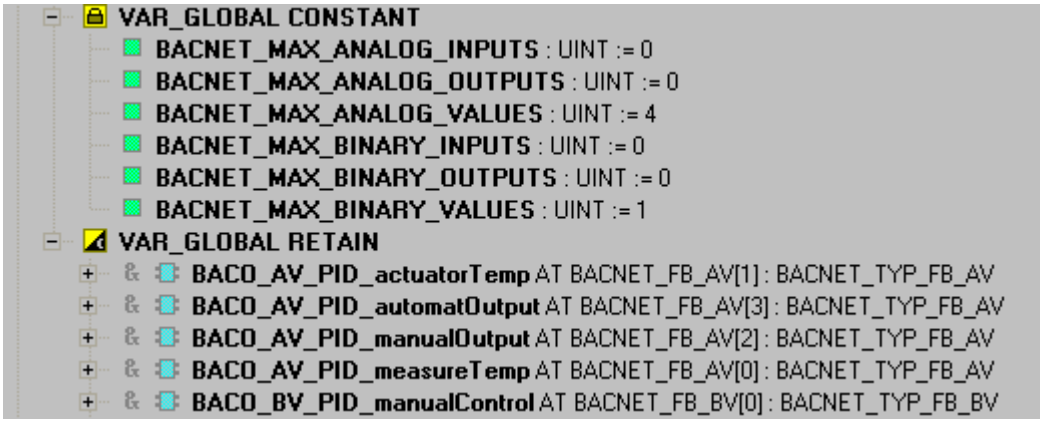


Note:

It is recommended to pay attention to BACnet object names. Select names so, that it is evident that they are BACnet objects, which function block in your program do they belong to and what do they affect.

E.g. BACO_BV_PID_manualControl

After successful compilation, we can see, in the variable inspector, our declared BACnet objects.



Furthermore, it is autocreated by the MOSAIC environment the file HWConfig.ST which contains necessary declarations, see bellow. Interesting is the last part of the declaration that declares our defined objects which can be used in the user program.

```
VAR_GLOBAL CONSTANT
  BACNET_MAX_BINARY_INPUTS    : UINT := 0;
  BACNET_MAX_BINARY_OUTPUTS   : UINT := 0;
  BACNET_MAX_BINARY_VALUES    : UINT := 1;
  BACNET_MAX_ANALOG_INPUTS    : UINT := 0;
  BACNET_MAX_ANALOG_OUTPUTS   : UINT := 0;
  BACNET_MAX_ANALOG_VALUES    : UINT := 4;
END_VAR

VAR_GLOBAL RETAIN
  BACNET_FB_BV : ARRAY[0..BACNET_MAX_BINARY_VALUES-1] OF T_BACNET_TYP_FB_BV := [
  ( IN_OBJECT_NAME := 'BACO_BV_PID_manualControl', IN_DESCRIPTION := 'Manual operation of the
controller required' ) ];

  BACNET_FB_AV : ARRAY[0..BACNET_MAX_ANALOG_VALUES-1] OF T_BACNET_TYP_FB_AV := [
  ( IN_OBJECT_NAME := 'BACO_AV_PID_measureTemp', IN_DESCRIPTION := 'measured temperature',
IN_UNITS := BACNET_UNITS_DEGREES_CELSIUS ),
  ( IN_OBJECT_NAME := 'BACO_AV_PID_actuatorTemp', IN_DESCRIPTION := 'required  temperature',
IN_UNITS := BACNET_UNITS_DEGREES_CELSIUS ),
  ( IN_OBJECT_NAME := 'BACO_AV_PID_manualOutput', IN_DESCRIPTION := 'required value of ouput during
the manual operation', IN_UNITS := BACNET_UNITS_PERCENT ),
  ( IN_OBJECT_NAME := 'BACO_AV_PID_automatOutput', IN_DESCRIPTION := 'output value from the
controller', IN_UNITS := BACNET_UNITS_PERCENT ) ];

  BACO_BV_PID_manualControl   AT BACNET_FB_BV[0] : BACNET_TYP_FB_BV;
  BACO_AV_PID_measureTemp     AT BACNET_FB_AV[0] : BACNET_TYP_FB_AV;
  BACO_AV_PID_actuatorTemp    AT BACNET_FB_AV[1] : BACNET_TYP_FB_AV;
  BACO_AV_PID_manualOutput    AT BACNET_FB_AV[2] : BACNET_TYP_FB_AV;
  BACO_AV_PID_automatOutput   AT BACNET_FB_AV[3] : BACNET_TYP_FB_AV;
END_VAR
```

Example of program modification with PID and „control“ via BACnet objects

```
PROGRAM PROGRAM_PID

  // BACnet objects call
  BACO_AV_PID_actuatorTemp();   // required temperature
  BACO_AV_PID_automatOutput();  // controller output
  BACO_AV_PID_manualOutput();   // required controller output during manual
control
  BACO_AV_PID_measureTemp();    // measured temperature
  BACO_BV_PID_manualControl();  // manual control requirement

  // assignment of BACnet objects to controller parameters
  PID_TUV_MAN.0 := BACO_BV_PID_manualControl.OUT_PRESENT_VALUE;
  PID_TUV_sp    := BACO_AV_PID_actuatorTemp.OUT_PRESENT_VALUE;

  // assignment of variables to BACnet objects
  // measured temperature
  BACO_AV_PID_measureTemp.IN_PRESENT_VALUE   := measureTemp;
  // controller output = action intervention
  BACO_AV_PID_automatOutput.IN_PRESENT_VALUE := automatOutput;

  // for control of the controller output during the manual control
  PID_TUV_hv    := BACO_AV_PID_manualOutput.OUT_PRESENT_VALUE;

END_PROGRAM
```

Objednávky a informace:
Teco a. s. Havlíčkova 260, 280 58 Kolín 4, tel. 321 737 611, fax 321 737 633