Anybus® CompactCom™ 30

BACnet MS/TP

NETWORK GUIDE

HMSI-168-73 2.3 en-US ENGLISH

Connecting Devices™

Important User Information

Disclaimer

The information in this document is for informational purposes only. Please inform HMS Industrial Networks of any inaccuracies or omissions found in this document. HMS Industrial Networks disclaims any responsibility or liability for any errors that may appear in this document.

HMS Industrial Networks reserves the right to modify its products in line with its policy of continuous product development. The information in this document shall therefore not be construed as a commitment on the part of HMS Industrial Networks and is subject to change without notice. HMS Industrial Networks makes no commitment to update or keep current the information in this document.

The data, examples and illustrations found in this document are included for illustrative purposes and are only intended to help improve understanding of the functionality and handling of the product. In view of the wide range of possible applications of the product, and because of the many variables and requirements associated with any particular implementation, HMS Industrial Networks cannot assume responsibility or liability for actual use based on the data, examples or illustrations included in this document nor for any damages incurred during installation of the product. Those responsible for the use of the product must acquire sufficient knowledge in order to ensure that the product is used correctly in their specific application and that the application meets all performance and safety requirements including any applicable laws, regulations, codes and standards. Further, HMS Industrial Networks will under no circumstances assume liability or responsibility for any problems that may arise as a result from the use of undocumented features or functional side effects found outside the documented scope of the product. The effects caused by any direct or indirect use of such aspects of the product are undefined and may include e.g. compatibility issues and stability issues.

Table of Contents

Page

1 Preface

1.1 About this document

This document is intended to provide a good understanding of the functionality offered by the Anybus CompactCom 30 BACnet MS/TP. The document describes the features that are specific to Anybus CompactCom 30 BACnet MS/TP. For general information regarding Anybus CompactCom 30, consult the Anybus CompactCom 30 design guides.

The reader of this document is expected to be familiar with high level software design and communication systems in general. The information in this network guide should normally be sufficient to implement a design. However if advanced BACnet MS/TP specific functionality is to be used, in-depth knowledge of BACnet MS/TP networking internals and/or information from the official BACnet MS/TP specifications may be required. In such cases, the persons responsible for the implementation of this product should either obtain the BACnet MS/TP specification to gain sufficient knowledge or limit their implementation in such a way that this is not necessary.

For additional related documentation and file downloads, please visit the support website at www.anybus.com/support.

1.2 Related Documents

Document	Author	Document ID
Anybus CompactCom 30 Software Design Guide	HMS	HMSI-216-125
Anybus CompactCom 30 Hardware Design Guide	HMS	HMSI-216-126
Anybus CompactCom B30 Design Guide	HMS	HMSI-27-230
Anybus CompactCom Host Application Implementation Guide	HMS	HMSI-27-334
BACnet specification	ASHRAE	Doc. Id. 135, 2008
Anybus CompactCom Implementation Tutorial	HMS	HMSI-168-106

1.3 Document History

Version	Date	Description
1.00	2011-09-15	First release
1.01	2011-10-26	Minor corrections and updates
1.02	2012-04-20	Minor corrections and updates
1.03	2012-09-14	Minor corrections
2.0	2017-07-06	Moved to DOX General update
2.1	2017-08-25	Minor update to descriptions of BACnet objects
2.2	2018-10-02	Correction to NC object Clarified description of bus connector
2.3	2019-06-13	Rebranding

1.4 Document Conventions

Ordered lists are used for instructions that must be carried out in sequence:

1. First do this
2. Then do this

Unordered (bulleted) lists are used for:

- Itemized information

- Instructions that can be carried out in any order

...and for action-result type instructions:

► This action...

→ leads to this result

Bold typeface indicates interactive parts such as connectors and switches on the hardware, or menus and buttons in a graphical user interface.

```
Monospaced text is used to indicate program code and other
kinds of data input/output such as configuration scripts.
```

This is a cross-reference within this document: *Document Conventions, p. 3*

This is an external link (URL): www.hms-networks.com

This is additional information which may facilitate installation and/or operation.

This instruction must be followed to avoid a risk of reduced functionality and/or damage to the equipment, or to avoid a network security risk.

Caution
This instruction must be followed to avoid a risk of personal injury.

WARNING
This instruction must be followed to avoid a risk of death or serious injury.

1.5 Document Specific Conventions

- The terms “Anybus” or “module” refers to the Anybus CompactCom module.
- The terms “host” or “host application” refer to the device that hosts the Anybus.
- Hexadecimal values are written in the format NNNNh or 0xNNNN, where NNNN is the hexadecimal value.
- A byte always consists of 8 bits.
- The terms “basic” and “extended” are used to classify objects, instances and attributes.

1.6 Trademarks

Anybus® is a registered trademark of HMS Industrial Networks. All other trademarks mentioned in this document are the property of their respective holders.

2 About the Anybus CompactCom 30 BACnet MS/TP

2.1 General

The Anybus CompactCom 30 BACnet MS/TP communication module provides instant BACnet and BACnet MS/TP connectivity via the patented Anybus CompactCom host interface. Most devices that support this standard can take advantage of the features provided by the module, allowing seamless network integration regardless of network type. The module supports linear network topology.

This product conforms to all aspects of the host interface for Anybus CompactCom 30 modules defined in the Anybus CompactCom 30 Hardware and Software Design Guides, making it fully interchangeable with any other device following that specification. However, BACNET, in some aspects, behaves differently than many other networks do. This must be taken into consideration when implementing an application. For more information see *Software Requirements, p. 7*..

2.2 Features

- Fulfills all requirements for a BACnet device
- Data sharing
- Linear network topology supported
- Supports baud rates 9600, 19200, 38400, and 76800 bps
- Customizable Identity Information, allowing the end product to appear as a vendor specific implementation.
- 256 ADIs available in simple mode for mapping to BACnet objects
- A total of 6120 ADIs (2040 per type) available in advanced mode for mapping to BACnet objects
- Change Of Value (COV) notification and Alarm/Event functionality supported (max 256 ADIs available)

3 Tutorial

3.1 Introduction

This chapter is a complement to the Anybus CompactCom Implementation Tutorial. The tutorial describes and explains a simple example of an implementation with Anybus CompactCom. This chapter includes network specific settings that are needed for a host application to be up and running and possible to test for use on BACnet MS/TP networks.

3.2 Implementation

It is recommended to enable attribute #7 (Support advanced mapping) in the BACnet Host Object, to fully take advantage of the functionality and flexibility of the module.

See also ...

- *Application Data (ADIs), p. 21*
- *Mapping of BACnet Objects to Anybus CompactCom, p. 22*
- *BACnet Host Object (EFh), p. 39*

3.3 Fieldbus Conformance Notes

- BACnet International does not require a certification for the use of BACnet products.
- Tests have been performed internally at HMS Industrial Networks to verify conformance with BACnet specifications.
- To enable the product to appear as a vendor specific implementation rather than a generic Anybus module, customize the information in the BACnet Host Object.

3.4 Certification

HMS Industrial Networks will not certify the Anybus CompactCom 30 BACnet MS/TP.

The module is implemented as a BACnet Application Specific Controller (B-ASC). Even though HMS Industrial Networks will not certify the module, the implementation fulfills the requirements for certification as a B-ASC. See *Implemented BACnet BIBBs, p. 47* for a complete list of BACnet BIBBs implemented in the module.

4 Basic Operation

4.1 General Information

4.1.1 Software Requirements

No additional network support code needs to be written in order to support the Anybus CompactCom 30 BACnet MS/TP, however due to the nature of the BACnet MS/TP networking system certain restrictions must be taken into account:

- There is no support for arrays of data elements in the ADIs as all data on BACnet is represented as single units without any possibility to access data in any other way.
- Data types UINT64 and SINT64 cannot be represented on BACnet.
- It is not possible to map read process data.

For in depth information regarding the Anybus CompactCom software interface, consult the general Anybus CompactCom 30 Software Design Guide.

See also ...

- Anybus-CompactCom 30 Software Design Guide, Application Data Object (FEh)
- *Network Object (03h), p. 33*

4.2 Device Customization

4.2.1 Network Identity

By default, the module uses the following identity settings:

Vendor Name:	“HMS Industrial Networks AB”
Vendor ID:	01E6h (HMS Industrial Networks)
Model Name	“Anybus-CompactCom”
Object Name	“Controller”
Network Type	0099h(“BACnet MS/TP”)
Product Name:	Anybus-CompactCom BACnet MS/TP

Optionally, it is possible to customize the identity of the module by implementing the corresponding instance attributes in the BACnet Host Object.

See also...

- *BACnet Host Object (EFh), p. 39* (Host Application Object)
- *Network Object (03h), p. 33*

4.3 BACnet MS/TP Implementation

It is recommended to enable attribute 7 (Support advanced mapping) in the BACnet Host Object, to fully take advantage of the functionality and flexibility of the module.

The module is implemented as a B-ASC (BACnet Application Specific Controller). It supports the following BACnet objects:

Object Name	Class
Device object	8
Analog Value object	2
Binary Value object	5
Multi-state Value object	19
Notification Class object	15

Each Anybus CompactCom 30 BACnet MS/TP contains one Device object and six Notification Class objects. These objects are fixed and can not be changed by the application.

The Analog Value, Binary Value, and Multi-State Value objects and their data are mapped against the ADIs in the Application Data object.

The BACnet Interoperability Building Blocks (BIBBs), that are implemented in the module, are listed in appendix *B*.

See also...

- *Network Data Exchange, p. 21*
- Application Data Object (see Anybus-CompactCom 30 Software Design Guide)
- *Implemented BACnet BIBBs, p. 47*

4.3.1 Device Object

The BACnet device object contains information about the module as a node on a BACnet network. Apart from the value of the Object_Identifier, the values of the properties in the object can not be changed by the application directly. Some values can be changed by setting the corresponding attributes in the BACnet Host object.

Properties that are stored in non volatile memory, keep their assigned values when the module is turned off.

Property Identifier	Value	R/W	NV	Description/Comment
Object_Identifier	N/A	R/W		The instance number portion (device instance) of this property is affected when the value attribute of Instance 3 in the Network Configuration Object is changed.
Object_Name	“Controller”	R/W		This property can be written to only if the Object Name in the BACnet Host object can be set. This property can be changed by setting the corresponding attribute in the BACnet Host object
Object_Type	DEVICE	R		
System_Status	NON_OPERATIONAL OPERATIONAL	R		The status of the system is reported as NON_OPERATIONAL if the Anybus CompactCom module has entered the ERROR state. Otherwise the state is reported as OPERATIONAL.
Vendor_Name	“HMS Industrial Networks AB”	R		This property can be changed by setting the corresponding attribute in the BACnet Host object
Vendor_Identifier	486	R		HMS Industrial Networks This property can be changed by setting the corresponding attribute in the BACnet Host object
Model_Name	“Anybus-CompactCom”	R		This property can be changed by setting the corresponding attribute in the BACnet Host object
Firmware_Revision	N/A	R		Firmware revision of the Anybus CompactCom 30 BACnet MS/TP as a string. This property can be changed by setting the corresponding attribute in the BACnet Host object
Application_Software_Revision	N/A	R		
Protocol_Version	1	R		
Protocol_Revision	14	R		
Protocol_Service_Supported	1001 0100 0000 1011 1100 1000 0010 0000 1110 0001	R		Bit map stating what protocol services are supported in the object, see *Supported BACnet Services, p. 19*.
Protocol_Object_Types_ Supported	0010 0100 1000 0001 0001 0000	R		Bit map stating what object types are supported in the object (analog-value, binary-value, multi-state value, device, and notification-class objects).
Object_List		R		This list is filled based on the ADIs that are implemented in the application.
Max_APDU_Length_Accepted	480	R		
Segmentation_Supported	SUPPORTED_BOTH	R		Segmentation supported for both Rx and Tx APDU.

Property Identifier	Value	R/W	NV	Description/Comment
Max_Segments_Accepted	68	R		Supporting 68 segments Number of maximum length APDUs that can be received in a segmented message. This is the maximum APDU payload in a request after segmentation.
Local_Time	N/A	R		The local time is synchronized from the application at power up or set from the BACnet network via the TimeSynchronization service.
Local_Date	N/A	R		The local date is synchronized from the application at power up or set from the BACnet network via the TimeSynchronization service.
APDU_Timeout	10000 (default) Valid range: 0 - 65535	R/W	NV	APDU transaction timeout (ms). The value 0 is only valid if Number_Of_ APDU_Retries = 0.
Number_Of_APDU_Retries	3 (default) Valid range: 0 - 255	R/W	NV	Number of APDU transaction and/or segment retransmission retries.
Max_Master	127 (default) Valid range: 1 - 127	R/W	NV	Address range scanned for new master nodes.
Max_Info_Frames	1 (default) Valid range: 1 - 255	R/W	NV	Number of information frames node may send before it must pass the token.
Device_Address_Binding	N/A	R		Managed by the APL layer and contains the list of all device address bindings of active client processes inside the Anybus CompactCom BACnet/IP module.
Database_Revision	N/A	R	NV	Incremented by 1 each time an object identifier is changed or the name of a BACnet object is changed.
APDU_Segment_Timeout	5000 (default) Valid range: 0 - 65535	R/W	NV	APDU segment timeout (ms). The value 0 is only valid if Number_Of_ APDU_Retries = 0.
Property_List	[System_Status, Vendor_Name, Vendor_Identifier, Model_Name, Firmware_Revision, Application_ Software_Version, Protocol_Version, Protocol_Revision, Protocol_Service_ Supported, Protocol_ Object_Types_ Supported, Object_ List, Max_APDU_ Length_Accepted, Segmentation_ Supported, Max_ Segments_Accepted, Local_Time, Local_ Date, APDU_Timeout, Number_Of_APDU_ Retries, Device_ Address_Binding, Database_Revision, APDU_Segment_ Timeout, Active_COV_ Subscriptions]	R		Array containing all properties supported by the device object, except Object_Name, Object_Type, Object_Identifier and Property_List.

4.3.2 Analog Value Object

The analog value object is mapped to ADIs of data types that represent analog values, e.g. UINT16.

Properties that are stored in non volatile memory, keep their assigned values when the module is turned off. The properties are only available if the corresponding ADI is mapped on the write process data channel and will be set to default at a change in the write process data map.

See also...

- *Mapping of BACnet Objects to Anybus CompactCom, p. 22*
- *Communication Settings, p. 21*
- *Alarm/Event Functionality, p. 24*

Property Identifier	Value	R/W	NV	Description/Comment
Object_Identifier	N/A	R		
Object_Name	N/A	R/W		• In simple mode: Analog_Value_# (# = instance number) • In advanced mode: Corresponding ADI name, max name length 252 characters. If set support is implemented for the ADI name attribute (instance attribute #1, Application Object,FEh), the ADI name can be set from BACnet by sending a WriteProperty request to this property. If the host application for any reason returns an error code when the ADI name is read, the BACnet device class error code OPERATIONAL_PROBLEM is returned to the network.
Object_Type	ANALOG_VALUE	R		
Present_Value	N/A	R/W		Corresponding ADI value converted to Real. If the host application for any reason returns an error code, the BACnet device class error code OPERATIONAL_PROBLEM is returned to the network.
Status_Flags	Default: {0, 0, 0, 0} (0 = not set)	R		Bit string of Status flags indicating the status of the object. Bit 0: IN_ALARM Bits 1 - 3: not used. Set to 0.
Event_State	NORMAL (0)	R		Valid states: 0: NORMAL 3: HIGH_LIMIT 4: LOW_LIMIT
Out_Of_Service	FALSE	R		Always FALSE
Units	NO_UNITS	R		
COV_Increment	0	R/W	NV	Min. value: 0 Max. value: Corresponding ADI data type max value.
Time_Delay	0	R/W	NV	If an event occurs, this property gives the delay before an alarm is issued on the bus (s). If a value is outside a given limit for a brief period of time (less than the time delay), no alarm will be issued. Min. value: 0 Max. value: UINT32max/1000

Property Identifier	Value	R/W	NV	Description/Comment
Notification_Class	0	R/W	NV	Min. value: 0 Max. value: 5
High_Limit	0	R/W	NV	Min. value: Min. value of the ADI's data type Max. value: Max. value of the ADI's data type
Low_Limit	0	R/W	NV	
Deadband	0	R/W	NV	Min. value: 0 Max. value: Corresponding ADI data type max value
Limit_Enable	Default: {0, 0} (0 = not set)	R/W	NV	Bit string that determines what TO event limits are enabled Bit 0: LOW_LIMIT_ENABLE Bit 1: HIGH_LIMIT_ENABLE
Event_Enable	Default: {0, 0, 0} (0 = not set)	R/W	NV	Bit string that determines what TO events that are enabled Bit 0: TO-OFFNORMAL Bit 1: not used. Set to 0. Bit 2: TO-NORMAL
Notify_Type	Alarm	R/W	NV	Specifies the classification of an TO event that is sent by this object. 0: ALARM 1: EVENT
Acked_Transitions	T, T, T	R		Bit string that determines what TO events has been acknowledged by a BACnet recipient. Only available if the corresponding ADI is mapped on the write process data channel. Bit 0: TO-OFFNORMAL Bit 1: TO-FAULT Bit 2: TO-NORMAL
Event_Time_Stamps	N/A	R		Array of BACnetTimeStamp that specifies the last TO event stamp that was triggered. Only available if the corresponding ADI is mapped on the write process data channel. (ArrayIdx 0: Number of elements) ArrayIdx 1: TO-OFFNORMAL ArrayIdx 2: TO-FAULT ArrayIdx 3: TO-NORMAL
Event_Detection_Enable	TRUE	R/W	NV	This property specifies if alarm/event detection is enabled for the object. Note: Property is only available if the corresponding ADI is mapped on write process data. See *Setup of Alarm and Events, p. 25* for more details regarding this property.
Property_List	Corresponding ADI not mapped on write PD:[Present_Value, Status_Flags, Event_ State, Out_Of_ Service, Units]Corresponding ADI mapped on write PD:[Present_Value, Status_Flags, Event_ State, Out_Of_ Service, Units, COV_ Increment, Time_ Delay, Notification_ Class, High_Limit, Low_Limit, Deadband, Limit_Enable, Event_ Enable, Acked_ Transitions, Notify_ Type, Event_Time_ Stamps, Event_ Detection_Enable]	R		Array containing all properties supported by the analog value object, except Object_ Name, Object_Type, Object_Identifier and Property_List.

Non volatile properties are kept in non volatile memory until the write process data map changes. After a change to the write process data map, the BACnet object properties will be set to their default values. Non volatile properties are saved to non volatile memory immediately after they are changed.

The Present_Value property is linked to the Value attribute of the corresponding ADI. A successful read request from the network will return a value that will be converted to a BACnet Real value and returned to the network. If an error is returned from the application, the BACnet device class error code OPERATIONAL_PROBLEM is returned to the network.

When the Present_Value property is written from the network, the BACnet Real value is converted to the data type of the corresponding ADI. For all data types, except FLOAT, all decimal precision is lost. If error code Out of range or Attribute not settable is returned, the corresponding BACnet error code will be returned to the network. Any other error code will be translated to OPERATIONAL_PROBLEM.

4.3.3 Binary Value Object

The binary value object is mapped to ADIs of data type BOOL.

Properties that are stored in non volatile memory, keep their assigned values when the module is turned off. The properties are only available if the corresponding ADI is mapped on the write process data channel and will be set to default at a change in the write process data map.

See also...

- *Mapping of BACnet Objects to Anybus CompactCom, p. 22*
- *Communication Settings, p. 21*
- *Alarm/Event Functionality, p. 24*

Property Identifier	Value	R/W	NV	Description/Comment
Object_Identifier	N/A	R		
Object_Name	N/A	R/W		• In simple mode: Binary_Value_# (# = instance number) • In advanced mode: Corresponding ADI name, max name length 252 characters. If set support is implemented for the ADI name attribute (instance attribute #1, Application Object,FEh), the ADI name can be set from BACnet by sending a WriteProperty request to this property. If read request for any reason returns a error code from the application, the BACnet device class error code OPERATIONAL_ PROBLEM is returned to the network.
Object_Type	BINARY_VALUE	R		
Present_Value	N/A	R/W		Corresponding ADI value
Status_Flags	Default: {0, 0, 0, 0} (0 = not set)	R		Bit string of Status flags indicating the status of the object. Bit 0: IN_ALARM Bits 1 - 3: not used. Set to FALSE.
Event_State	NORMAL	R		Valid states: 0: NORMAL 2: OFF_NORMAL
Out_Of_Service	FALSE	R		Always FALSE
Time_Delay	0	R/W	NV	If an event occurs, this property gives the delay before an alarm is issued on the bus (s). If a value is outside a given limit for a brief period of time (less than the time delay), no alarm will be issued.
Notification_Class	0	R/W	NV	Min. value: 0 Max. value: 5
Alarm_Value	INACTIVE (0)	R/W	NV	
Event_Enable	N/A	R/W	NV	Bit string that determines what TO event is enabled Bit 0: TO-OFFNORMAL Bit 1: not used. Set to FALSE. Bit 2: TO-NORMAL
Notify_Type	Alarm	R/W	NV	Specifies the classification of an TO event that is sent by this object. 0: ALARM 1: EVENT

Property Identifier	Value	R/W	NV	Description/Comment
Acked_Transitions	Default: {1, 1, 1} (1 = set)	R		Bit string that determines what TO events have been acknowledged by a BACnet recipient. Only available if the corresponding ADI is mapped on the write process data channel. Bit 0: TO-OFFNORMAL Bit 1: TO-FAULT Bit 2: TO-NORMAL
Event_Time_Stamps	N/A	R		Array of BACnetTimeStamp that specifies the last TO event stamp that was triggered. (ArrayIdx 0: Number of elements) ArrayIdx 1: TO-OFFNORMAL ArrayIdx 2: TO-FAULT ArrayIdx 3: TO-NORMAL
Event_Detection_Enable	TRUE	R/W	NV	This property specifies if alarm/event detection is enabled for the object. Note: Property is only available if the corresponding ADI is mapped on write process data. See *Setup of Alarm and Events, p. 25* for more details regarding this property.
Property_List	Corresponding ADI not mapped on write PD:[Present_Value, Status_Flags, Event_ State, Out_Of_Service]Corresponding ADI mapped on write PD:[Present_Value, Status_Flags, Event_ State, Out_Of_ Service, Time_Delay, Notification_Class, Alarm_Value, Event_ Enable, Acked_ Transitions, Notify_ Type, Event_Time_ Stamps, Event_ Detection_Enable]	R		Array containing all properties supported by the binary value object, except Object_ Name, Object_Type, Object_Identifier and Property_List.

Non volatile properties are kept in non volatile memory until the write process data map changes. After a change to the write process data map, the BACnet object properties will be set to their default values. Non volatile properties are saved to non volatile memory immediately after they are changed.

The Present_Value property is linked to the Value attribute of the corresponding ADI. A successful read request from the network will return a value that will be converted to a BACnet BinaryPV value and returned to the network. If an error is returned from the application, the BACnet device class error code OPERATIONAL_PROBLEM is returned to the network.

When the Present_Value property is written from the network, the BACnet BinaryPV value is converted to Bool. If error code Out of range or Attribute not settable is returned, the corresponding BACnet error code will be returned to the network. Any other error code will be translated to OPERATIONAL_PROBLEM.

4.3.4 Multi-State Value Object

The multi-state value object is mapped to ADIs of data type ENUM.

Properties that are stored in non volatile memory, keep their assigned values when the module is turned off. The properties are only available if the corresponding ADI is mapped on the write process data channel and will be set to default at a change in the write process data map.

See also...

- *Mapping of BACnet Objects to Anybus CompactCom, p. 22*
- *Communication Settings, p. 21*
- *Alarm/Event Functionality, p. 24*

Property Identifier	Value	R/W	NV	Description/Comment
Object_Identifier	N/A	R		
Object_Name	N/A	R/W		• In simple mode: Multistate_Value_# (# = instance number) • In advanced mode: Corresponding ADI name, max name length 252 characters. If set support is implemented for the ADI name attribute (instance attribute #1, Application Object,FEh), the ADI name can be set from BACnet by sending a WriteProperty request to this property. If a read request returns an error code from the application, the BACnet device class error code OPERATIONAL_PROBLEM is returned to the network.
Object_Type	MULTISTATE_VALUE	R		
Present_Value	N/A	R/W		Corresponding ADI value
Status_Flags	Default: {0, 0, 0, 0} (0 = not set)	R		Bit string of Status flags indicating the status of the object. Bit 0: IN_ALARM Bit 1: FAULT Bits 2 - 3: not used. Set to FALSE.
Event_State	NORMAL	R		Valid states: 0: NORMAL 1: FAULT 2: OFF_NORMAL
Out_Of_Service	FALSE	R		Always FALSE
Number_Of_States	N/A	R		Corresponding ADI Max_Value + 1. If an error occurs when reading the Max_value, this property will be set to 256.
Time_Delay	0	R/W	NV	Time delay for an event to be triggered after occurrence (s)
Notification_Class	0	R/W	NV	Min. value: 0 Max. value: 5
Alarm_Values	Empty list	R/W	NV	
Fault_Values	Empty list	R/W	NV	
Event_Enable	N/A	R/W	NV	Bit string that determines what TO event is enabled Bit 0: TO-OFFNORMAL Bit 1: TO-FAULT Bit 2: TO-NORMAL

Property Identifier	Value	R/W	NV	Description/Comment
Notify_Type	Alarm	R/W	NV	Specifies the classification of an TO event that is sent by this object. 0: ALARM 1: EVENT
Acked_Transitions	Default: {1, 1, 1} (1 = set)	R		Bit string that determines the TO events that have been acknowledged by a BACnet recipient. Only available if the corresponding ADI is mapped on the write process data channel. Bit 0: TO-OFFNORMAL Bit 1: TO-FAULT Bit 2: TO-NORMAL
Event_Time_Stamps	N/A	R		Array of BACnetTimeStamp that specifies the last TO event stamp that was triggered. Only available if the corresponding ADI is mapped on the write process data channel. (ArrayIdx 0: Number of elements) ArrayIdx 1: TO-OFFNORMAL ArrayIdx 2: TO-FAULT ArrayIdx 3: TO-NORMAL
Event_Detection_Enable	TRUE	R/W	NV	This property specifies if alarm/event detection is enabled for the object. Note: Property is only available if the corresponding ADI is mapped on write process data. See *Setup of Alarm and Events, p. 25* for more details regarding this property.
State_Text	N/A	R		This property is mapped against the module's Get_Enum_String service Max length for each enum string is 252 characters
Reliability	NO_FAULT_DETECTED	R		Object reliability Only available if the corresponding ADI is mapped on the write process data channel. 0: NO_FAULT_DETECTED 9: MULTI_STATE_FAULT
Property_List	Corresponding ADI not mapped on write PD: [Present_Value, Status_Flags, Event_State, Out_Of_Service, Number_Of_States, State_Text, Reliability] Corresponding ADI mapped on write PD: [Present_Value, Status_Flags, Event_State, Out_Of_Service, Number_Of_States, Time_Delay, Notification_Class, Alarm_Values, Fault_Values, Event_Enable, Acked_Transitions, Notify_Type, Event_Time_Stamps, Event_Detection_Enable, State_Text, Reliability]	R		Array containing all properties supported by the multistate value object, except Object_Name, Object_Type, Object_Identifier and Property_List.

Non volatile properties are kept in non volatile memory until the write process data map changes. After a change to the write process data map, the BACnet object properties will be set to their default values. Non volatile properties are saved to non volatile memory immediately after they are changed.

The Present_Value property is linked to the Value attribute of the corresponding ADI. A successful read request from the network will return a value that will be converted to a BACnet Unsigned value, incremented by one and returned to the network. (The ENUM ADI is zero based in the Anybus CompactCom, but the first state of a multistate value object on BACnet is one.) If an error is returned from the application, a BACnet device class error code OPERATIONAL_ PROBLEM is returned to the network.

When the Present_Value property is written from the network, the BACnet Unsigned value is decremented by one. If error code Out of range or Attribute not settable is returned, the corresponding BACnet error code will be returned to the network. Any other error code will be translated to OPERATIONAL_PROBLEM.

The State_Text property is linked to the Get_Enum_String service of the ADI. The property is an array property, where array index 0 returns the number of states (see property Number_Of_ States). All other indices return the corresponding state text. Multistate states begin at 1, so the value is decremented by 1 and a Get_Enum_String command is sent to the corresponding ADI. If an error is returned from the application, a BACnet device class error code OPERATIONAL_ PROBLEM is returned to the network.

4.3.5 Notification Class Object

An Anybus CompactCom 30 BACnet MS/TP always contain 6 (0 - 5) instances of this object. Each instance contains a list of devices that need to be informed about certain events and alarms.

The default values of properties, stored in the non-volatile memory, are assigned by the Anybus CompactCom module the first time the module is started. When a BACnet user requests to write any of these properties the data is saved to non volatile memory directly after the validation of the service write request.

See also...

- *COV Notifications, Alarms and Events, p. 24*

Property Identifier	Value	R/W	NV	Description/Comment
Object_Identifier	N/A	R		
Object_Name	Default: Notification_ Class_# (# = instance number)	RW	NV	BACnet Char string (only ANSIX34 supported). Max. 30 characters
Object_Type	NOTIFICATION_CLASS	R		
Notification_Class	N/A	R		Equal to instance number
Priority	Default: {3, 0, 0, 0} Values 0 - 255 are allowed, a lower value has higher priority than a higher value.	Index 0: R Index 1-3: R/ W	NV	BACnet array of Unsigned values: 0: Number of Elements 1: TO-OFFNORMAL 2: TO-FAULT 3: TO-NORMAL
Ack_Required	Default: {0, 0, 0} (0 = not set)	R/W	NV	BACnet Bit string of 3 bits: Bit 0: TO-OFFNORMAL Bit 1: TO-FAULT Bit 2: TO-NORMAL
Recipient_List	N/A	R/W	NV	BACnet list of BACnetDestination primitives. There are 60 recipient list entries available to be configured among the 6 Notification Class instances.

Property Identifier	Value	R/W	NV	Description/Comment
Property_List	[Notification_Class, Priority, Ack_ Required, Recipient_ List, Recipient_Error_ Field]	R		Array containing all properties supported by the notification class object, except Object_ Name, Object_Type, Object_Identifier and Property_List.
Vendor property 600: Recipient_ Error_Field	Default: {0, 0, 0, ...} (0 = not set)	R		BACnet Bit string of a maximum of 60 bits. The string has one bit per element present in the Recipient_List. If a bit is set an event notification error is present for the corresponding recipient entry in the Recipient_List property.

The Recipient_List specifies one or more recipients that a notification event should be sent to. In total there are 60 list entries available to be divided among the 6 Notification Class object instances. These are managed on a “pool” basis, and when all have been assigned to the Notification object instances, an error is issued (CLASS_SERVICES, CODE_NO_SPACE_TO_ADD_ LIST_ELEMENT).

4.3.6 Supported BACnet Services

The following services are supported by the Anybus CompactCom 30 BACnet MS/TP

Service	Description
ReadProperty	Reads a single property from a BACnet object
ReadPropertyMultiple	Reads multiple properties from a BACnet object
WriteProperty	Writes a single property to a BACnet object
WritePropertyMultiple	Writes multiple properties to a BACnet object
Who-Is	Upon receipt of a Who-Is request the module will return an I-Am response containing its BACnet device object instance number. The service is used to find out which devices are present on the network, and their addresses.
Who-Has	This service is used to find out what devices in a network are implementing a specific object. The service uses either the object identifier or the object name for lookup. Upon a receipt of a Who_Has request the module will return an I-Have response, if it implements the requested object.
I-Am	This service is sent by the module in response to a matching Who-Is request. It is used to locate BACnet devices based on the device object instance number.
DeviceCommunication-Control	Turns off the BACnet communication of the module. Once communication is disabled the module will only respond to DeviceCommunicationControl requests and ReinitializeDevice messages. If only initiation instead of all communication is disabled, the module will also respond to Who-Is requests.
ReinitializeDevice	Resets devices over BACnet. A cold start results in a hardware reset. A warm start results in a restart of the BACnet stack and data mapping will not be affected.
TimeSynchronization	Synchronizes the current time and date of the module.
SubscribeCOV	Subscribes for changes of value with a BACnet object. The ADI corresponding to the object has to be mapped on write process data.
GetAlarmSummary	Responds with a list containing information about all active alarms.
GetEventSummary	Responds with a list containing information about all active alarms and events.
AcknowledgeAlarm	Acknowledges an active alarm.

4.3.7 BACnet Error Codes

Device Class Error Codes

Error Code	Indicates
OPERATIONAL_PROBLEM	The application has generated a not expected response, e.g. reading the value attribute of an ADI failed when performing ReadProperty on a value object.

Property Class Error Codes

Error Code	Indicates
WRITE_ACCESS_DENIED	Trying to write a non-writable property.
READ_ACCESS_DENIED	Trying to read a non-readable property.
PROPERTY_IS_NOT_AN_ARRAY	Trying to read or write an array index of a property that is not an array.
INVALID_DATA_TYPE	Trying to write a property using the wrong data type.
VALUE_OUT_OF_RANGE	The value written to a property is either too large or too small.
UNKNOWN_PROPERTY	Trying to read a non-existent property.
INVALID_ARRAY_INDEX	Trying to read or write an array index that does not exist for the property.
CHARACTER_SET_NOT_SUPPORTED	Writing a char string property with a character set not supported by the Anybus CompactCom module.
DUPLICATE_NAME	The value written to an Object_Name property already exists as an object name for another object.
OTHER	The Anybus CompactCom module encountered a general error.

Service Class Error Codes

Error Code	Indicates
INCONSISTENT_PARAMETERS	Something went wrong while parsing the data in a WriteProperty request.
INVALID_EVENT_STATE	The event state supplied in an AcknowledgeAlarm request is not correct.
INVALID_TIME_STAMP	The time stamp supplied in an AcknowledgeAlarm request is not correct.
SERVICE_REQUEST_DENIED	Generic error occurred for an AcknowledgeAlarm request
CHARACTER_SET_NOT_SUPPORTED	Wrong character set used for the password supplied with a ReinitializeDevice or DeviceCommunicationControl request.
COMMUNICATION_DISABLED	Received an invalid request for the current DeviceCommunicationControl state.
COV_SUBSCRIPTION_FAILED	a SubscribeCOV request could not be completed due to lack of resources or it is not possible to set up the object for COV notification due to properties not being mapped on process-write data.

Object Class Error Codes

Error Code	Indicates
UNKNOWN_OBJECT	The requested object does not exist in the Anybus CompactCom module.
NO_ALARM_CONFIGURED	There is no alarm or event to acknowledge for the event state supplied with the AcknowlegeAlarm request.
OTHER	Generic error during object request
ABORT_BUFFER_OVERFLOW	Response APDU is too large.

Security Class Error Codes

Error Code	Indicates
PASSWORD_FAILURE	Wrong password for ReinitializeDevice or DeviceCommunicationControl service.

4.4 Communication Settings

As with other Anybus-CompactCom products, network related communication settings are grouped in the Network Configuration Object (04h).

In this case, this includes...

See also...

- *Network Configuration Object (04h), p. 35*

4.5 Diagnostics

Major unrecoverable faults are reported in the Diagnostic Object

See also...

- *Diagnostic Object (02h), p. 32*

4.6 Network Data Exchange

4.6.1 Application Data (ADIs)

ADIs are represented through the BACnet objects. The properties of the BACnet objects are mapped to instances in the Application Data Object on the host application side.

There are two mapping schemes, one simple and one advanced. The application decides what scheme to use by implementing an attribute (#7, Support advanced mapping) in the BACnet Host Object. Accessible range of ADIs for the simple mapping scheme is 1 to 256. In the advanced mapping scheme, there are 2040 analog value objects, 2040 binary value objects and 2040 multi-state value objects that can be mapped to ADIs in any order. It is recommended to use advanced mapping to fully take advantage of the functionality and flexibility of the module.

See also...

- *Mapping of BACnet Objects to Anybus CompactCom, p. 22*
- *BACnet Host Object (EFh), p. 39*

4.6.2 Translation of Data Types

The Anybus data types are translated to BACnet standard and vice versa as follows:

Anybus Data Type	BACnet Object Name	Comments
BOOL	Binary Value Object	Each ADI element of this type occupies one byte.
ENUM	Multi-state Value Object	
SINT8	Analog Value Object	
UINT8	Analog Value Object	
SINT16	Analog Value Object	Each ADI element of this type occupies two bytes.
UINT16	Analog Value Object	
SINT32	Analog Value Object	Each ADI element of this type occupies four bytes.
UINT32	Analog Value Object	
FLOAT	Analog Value Object	
CHAR	Analog Value Object	Each ADI element of this type occupies one byte.

If an ADI of a size that doesn't match the size of the Present Value property in a Value Object, is mapped to a Value object, it will result in FAULT or in OPERATIONAL_PROBLEM when accessed.

4.6.3 Mapping of BACnet Objects to Anybus CompactCom

Simple Mapping

In the simple mapping schema, that is implemented by default, the module will scan the first 256 implemented instances of the application data object during initialization. It will read the data type attribute and based on the data type BACnet objects will be created in a sequential order, starting with 0. Anybus CompactCom data types will be mapped to BACnet objects according to the table above. This mapping will stay fixed. For an example see the figure below:

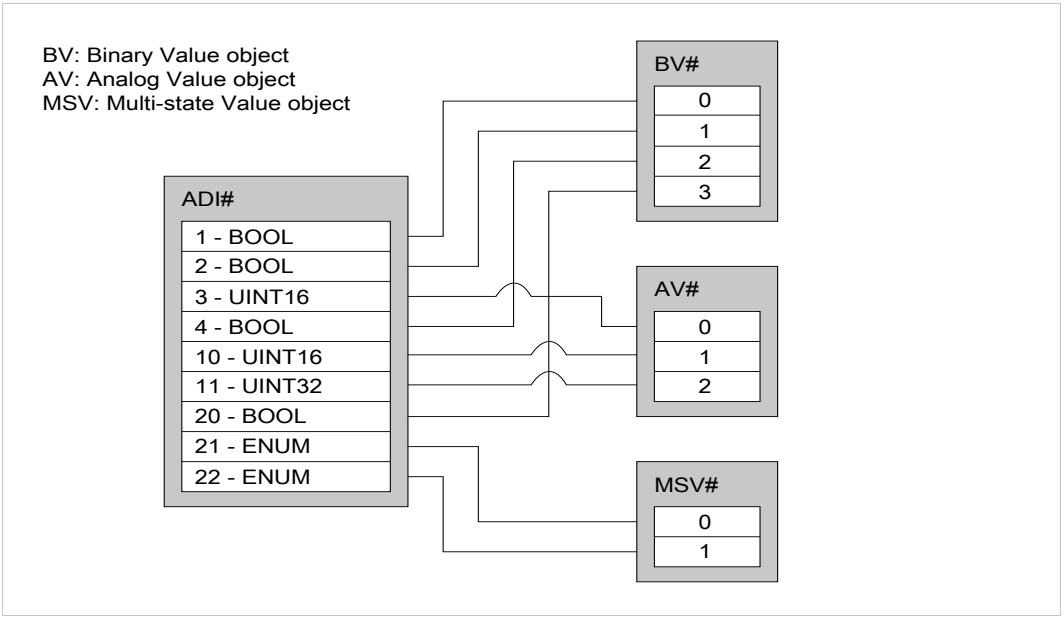


Fig. 1

Advanced Mapping

If the attribute “Advanced mapping supported” in the BACnet Host object is true, the user can create almost any type of mapping of BACnet objects to Anybus CompactCom ADIs. This mapping can be designed to closely match the application and it is up to the application to keep track of the ADI - BACnet object relationships. The services of the BACnet host object are then used to implement this mapping on BACnet. Valid range of object identifier number is 0 - 2039 for each value object type. For an example see the figure below:

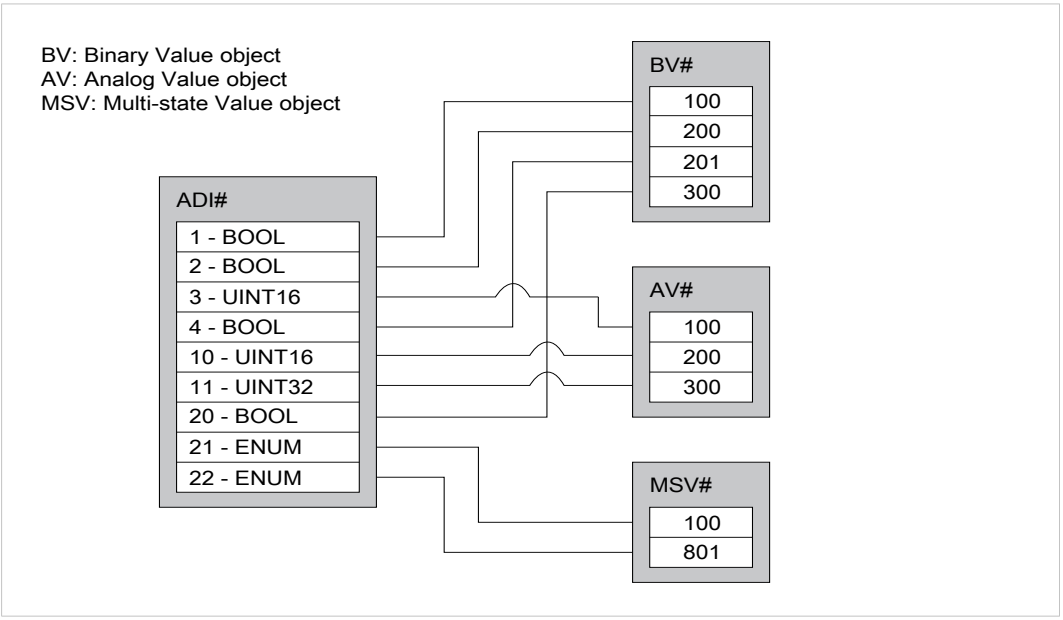


Fig. 2

4.6.4 Process Data

The Anybus CompactCom 30 BACnet MS/TP supports COV (Change Of Value) notification and Alarm/Event functionality, see *COV Notifications, Alarms and Events, p. 24*. To be able to use these features for a BACnet object, the corresponding ADI must be mapped on write process data. If not, the module has no way of detecting any changes in the value of the ADI. Also, if an ADI is mapped on write process data, the properties used for COV notifications and Alarms/ Events will be accessible to the corresponding BACnet object. If the mapping is changed, all COV and Alarm/Event information, that has been stored in non volatile memory, will be cleared.

No read process data is passed from the module to the application.

5 COV Notifications, Alarms and Events

5.1 General

The Anybus CompactCom 30 BACnet MS/TP supports COV (Change Of Value) notification and Alarm/Event functionality. These features can only be used for a BACnet object if the corresponding ADI is mapped on write process data.

If the mapping of the ADIs is changed, all COV and Alarm/Event information, that has been stored in non volatile memory, will be cleared.

5.2 COV (Change of Value) Notifications

A BACnet device can subscribe for a COV (Change of value) notification from another BACnet device on the network. The first device uses the service SubscribeCOV to send the identifier of the desired object to the second device to activate the notification. A change in the Present_ value property will then trigger a COV notification to the first BACnet device.

Each Value object is mapped to an ADI in the Anybus-CompactCom module, and the property Present_value corresponds to the Value attribute in the ADI. If the ADI is not mapped on the write process data area any change in the value will go unnoticed and a subscribeCOV service will return the error code COV_SUBSCRIPTION_FAILED.

In Binary Value and Multi-State Value objects any change in the present value will cause a notification to be sent. In Analog Value objects a property, COV_increment, is used to decide how much a present value needs to change in order for a notification to be sent.

5.3 Alarm/Event Functionality

A change in the present value of an object can also be used to trigger an alarm or an event.

Each value object is associated to a Notification Class object instance. Each instance contains a list of all recipients of a specific Alarm/Event notification. Alarms and events are handled in the same way by the value objects. If an alarm or an event is to be issued is decided by the application in the property Notify_Type in the value object and the choice indicates the severity of the notification to the recipient.

Alarms/events are issued every time a value object changes state. The reasons for changing states are specific to each of the three value objects as described below. The possibility to issue an alarm or an event has to be enabled in each object. It is also possible to delay an alarm/event using the Time_Delay property of the object. As previously mentioned this functionality is only available if the ADIs are mapped to write process data.

5.3.1 Notification Class Object Functionality

The Notification Class object is used to configure certain constraints in the functionality and the recipient list that is common to all objects that are associated to an instance of the Notification Class Object. The Anybus CompactCom 30 BACnet MS/TP module supports 6 instances of this object.

There are three object notifications (or events) that each indicate a transition from one state to another: TO-OFFNORMAL, TO-FAULT, and TO-NORMAL. Every alarm/event triggered by a value object, is represented as one of these. The interpretation depends on which value object was the origin of the notification.

5.3.2 Analog Value Object Alarm/Event Functionality

An Analog Value object has three possible states: NORMAL, LOW_LIMIT and HIGH_LIMIT. The LOW_LIMIT state is entered when the present value, object property Present_value, falls below the value in the object property Low_limit. The HIGH_LIMIT state is entered when the present value rises above the value in the property High_limit. When the present value returns above or below the given limit, the object returns to the NORMAL state.The OFF_NORMAL bit in the property Event_enable must be set, to allow transitions to other states than the NORMAL state. The LOW_LIMIT and HIGH_LIMIT bits in the property Limit_Enable must be set to allow transitions to the corresponding state.

A change of state will result in either a TO-NORMAL event or a TO-OFFNORMAL event issued to the Notification Class object instance. A TO-OFFNORMAL event indicates that the value of the property value has increased above the defined high limit or decreased below the defined low limit. A TO-NORMAL event indicates a return to the NORMAL state.

5.3.3 Binary Value Object Alarm/Event Functionality

A Binary Value Object has two states, NORMAL and OFF_NORMAL. An alarm value can be set and when the property Present_Value is changed to this value, the object changes from the NORMAL state to the OFF_NORMAL state. The corresponding bit(s) must be set in the property Event_Enable for this functionality to be available.

A change of state will result in either a TO-NORMAL event or a TO-OFFNORMAL event issued to the Notification Class object instance. A TO-OFFNORMAL event indicates that the value of the property value has changed to equal the alarm value. A TO-NORMAL event indicates a return to the NORMAL state.

5.3.4 Multi-State Value Object Alarm/Event Functionality

The Multi-State Value object contains one list for alarm values and one for fault values. When the property Present_Value is changed to a value that is present in the alarm value list, the object enters the OFF_NORMAL state and a TO-OFFNORMAL event is issued. If the value is present in the fault values list, the object enters the FAULT state and a TO-FAULT event is issued. A TO-NORMAL event is issued when the object returns to the NORMAL state from any of the other two states. The corresponding bit(s) must be set in the property Event_Enable for the functionality to be available.

5.3.5 Summary of States and Events for the Value Objects

Object	State	Possible events
Analog Value Object	NORMAL	TO-OFFNORMAL
	HIGH_LIMIT	TO-NORMAL
	LOW_LIMIT	
Binary Value Object	NORMAL	TO-OFFNORMAL
	OFF_NORMAL	TO-NORMAL
Multi State Value Object	NORMAL	TO-FAULT TO-OFFNORMAL
	FAULT	TO-NORMAL TO-OFFNORMAL
	OFF_NORMAL	TO-NORMAL TO-FAULT

5.4 Setup of Alarm and Events

This section gives a short guide to what properties in the BACnet objects need to be setup and monitored when implementing the BACnet alarm/event functionality.

5.4.1 Notification Class Object

BACnet object notifications are classified as either TO-OFFNORMAL, TO-FAULT, or TO-NORMAL. Any event triggered by a BACnet value object is interpreted as one of these three.

See also...

- *Notification Class Object, p. 18*

Priority

This property defines how to prioritize the three different kinds of notifications/events that can be issued by an object. A lower value has priority over a higher value, e.g. setting Priority arrayidx 2 to 1 and the other entries in the array to 5 and 76 will give TO-FAULT priority over the other notifications.

If all three events are given the same priority in the property, they will receive the following priority by default:

1. TO-NORMAL
2. TO-FAULT
3. TO-OFFNORMAL

I.e. TO-NORMAL has the highest priority.

Ack Required

This property specifies if an acknowledgement is required from the recipients of a specific TO event. Every TO event sent is flagged to tell the recipient whether an acknowledgement is required or not, based on the corresponding TO bit in the bit string in this property.

Recipient List

This list specifies the BACnet recipients that shall receive a notification of an alarm/event associated with the notification class. The list is a sequence of BACnetDestination primitives, where each entry consists of the following elements:

Element	Description
Transitions	Bit string that determines what TO events shall be sent to this recipient.
IssueConfirmedNotifications	Boolean that specifies if a ConfirmedEventNotification or UnConfirmedEventNotification request shall be sent to the recipient.
ProcessIdentifier	Unsigned32 value that specifies the process identifier that is linked to the notification event. If an acknowledgement is required for the specific event (see Ack_Required) the acknowledgement source has to have a valid value of a recipient process identifier to successfully acknowledge the alarm/event.
Recipient	BACnetRecipient primitive that can either be a DeviceInstanceNumber or a BACnetMAC address. The primitive will be added to the DAB (Device Address Binding) lookup mechanism of the BACnet APL layer and who-is/I-am BACnet UnConfirmedServices will be used to probe for the recipient device on BACnet. When a valid lookup is found the event notifications will be sent to that device. If no valid probe is found the corresponding bit in VendorProperty: 600, BACnetRecipientErrorField will be set in the notification class.
ToTime	BACNetTime primitive that specifies the ending time for when the recipient will accept event notifications. The contents must match the local RTC time or no event notifications will be sent to this recipient entry.
FromTime	BACNetTime primitive that specifies the starting time for when the recipient will accept event notifications. The contents must match the local RTC time or no event notifications will be sent to this recipient entry.
ValidDays	BACnetDaysOfWeek that specifies the valid days of the week for when the recipient will accept event notifications. The contents must match the local RTC time or no event notifications will be sent to this recipient entry.

Vendor Property: 600, Recipient_Error_Field

This vendor specific property, added by HMS Industrial Networks, holds a bit string that is used for diagnostics to determine if a recipient has notification errors present or not. Bit 0 in the bit string represents recipient 1 in the RecipientList property, bit 1 represents recipient 2, and so on.

If the bit for a recipient is set, one of the following error events are present:

- The recipient device was not found on the BACnet network.
- A ConfirmedEventNotification request sent to the recipient returns a negative acknowledge response or an internal timeout.

If a bit has been set, as a result of an error, the bit will be cleared when a successful notification event has been sent or when the Recipient_List property has been updated with a new list of recipients.

5.4.2 Analog Value Object

Available TO events are TO-NORMAL and TO-OFFNORMAL. The object has three states, NORMAL, LOW_LIMIT and HIGH_LIMIT, but the TO events only reflect transitions to and from a normal state to a not normal state.

Object Properties to Setup

Property	Description
Notification Class	Unsigned value that specifies which Notification Class the alarm/event notification of this object is associated to.
Event Enable	Bit string that defines what TO events will be enabled to send alarm/event notifications.
NotifyType	Enumeration that decides if a notification shall be an EVENT or an ALARM.
Limit Enable	Bit string that enables/disables HIGH_LIMIT and LOW_LIMIT guarding.
High Limit	BACnet real value (float) that specifies the upper limit of the normal value span where a TO_OFFNORMAL event will be issued.
Low Limit	BACnet real value (float) that specifies the lower limit of the normal value span where a TO_OFFNORMAL event will be issued.
Deadband	BACnet real value (float) that specifies the deadband for the high and low limits
Time Delay	Unsigned value that specifies the time (ms) that will elapse before an event notification is triggered to be sent to its recipient.
Event Detection Enable	Make sure this property is set to TRUE (default), otherwise alarms and events won't be triggered from this object

Object Properties to Monitor

Property	Description
Event State	BACnet enum of the event state that specifies the current state of the object.
Event Time Stamps	BACnet array of time stamps indicating when the last triggered events of each of the 3 different TO states, occurred.
Acked Transitions	Bit string that specifies the TO event transitions that have been acknowledged by a notification class recipient. It is enough if one recipient acknowledges an event for the corresponding TO bit to be set. If no acknowledgement is required for the notification class the corresponding TO bit will be set automatically.
Status Flags	Bit string of the status flags of the objects. If a TO-OFFNORMAL event is active the IN_ALARM bit will be set.

5.4.3 Binary Value Object

Available TO events are TO-NORMAL and TO-OFFNORMAL.

Object Properties to Setup

Property	Description
Notification Class	Unsigned value that specifies which Notification Class the alarm/event notification of this object is associated to.
Event Enable	Bit string that defines what TO events will be enabled to send alarm/event notifications.
NotifyType	Enumeration that decides if a notification shall be an EVENT or an ALARM.
Alarm Value	BACnet BinaryPV enumeration that specifies when Present_Value is OFF_NORMAL, either 0 (inactive) or 1 (active).
Time Delay	Unsigned value that specifies the time (ms) that will elapse before an event notification is triggered to be sent to its recipient.
Event Detection Enable	Make sure this property is set to TRUE (default), otherwise alarms and events won't be triggered from this object

Object Properties to Monitor

Property	Description
Event State	BACnet enum of the event state that specifies the current state of the object.
Event Time Stamps	BACnet array of time stamps indicating when the last triggered events of each of the2 different TO states, occurred.
Acked Transitions	Bit string that specifies the TO event transitions that have been acknowledged by a notification class recipient. It is enough if one recipient acknowledges an event for the corresponding TO bit to be set. If no acknowledgement is required for the notification class the corresponding TO bit will be set automatically.
Status Flags	Bit string of the status flags of the objects. If a TO-OFFNORMAL event is active the IN_ALARM bit will be set.

5.4.4 Multi-state Value Object

Available TO events are TO-NORMAL, TO-OFFNORMAL, and TO-FAULT.

Object Properties to Setup

If the same value is specified for the Alarm Values property as for the Fault Values property, both a TO_OFFNORMAL and a TO_FAULT will be triggered.

Property	Description
Notification Class	Unsigned value that specifies which Notification Class the alarm/event notification of this object is associated to.
Event Enable	Bit string that defines what TO events will be enabled to send alarm/event notifications.
NotifyType	Enumeration that decides if a notification shall be an EVENT or an ALARM.
Alarm Values	BACnet list of unsigned values that will trigger a TO-OFFNORMAL event.
Fault Values	BACnet list of unsigned values that will trigger a TO-FAULT event.
Event Detection Enable	Make sure this property is set to TRUE (default), otherwise alarms and events won't be triggered from this object
Time Delay	Unsigned value that specifies the time (ms) that will elapse before an event notification is triggered to be sent to its recipient.

Any valid value, that is not listed in the Alarm Values and Fault Values properties, can generate a TO-NORMAL event.

Object Properties to Monitor:

Property	Description
Event State	BACnet enum of the event state that specifies the current state of the object.
Event Time Stamps	BACnet array of time stamps indicating when the last triggered events of each of the 3 different TO states, occurred.
Acked Transitions	Bit string that specifies the TO event transitions that have been acknowledged by a notification class recipient. It is enough if one recipient acknowledges an event for the corresponding TO bit to be set. If no acknowledgement is required for the notification class the corresponding TO bit will be set automatically.
Status Flags	Bit string of the status flags of the objects. If a TO-OFFNORMAL event is active the IN_ALARM bit will be set. If a TO-FAULT event is active the FAULT bit will be set.
Reliability	BACnet enumeration that is set to 9 (multi_state_fault) if a TO-FAULT event is active and 0 (no_fault_detected) otherwise.

6 Anybus Module Objects

6.1 General Information

This chapter specifies the Anybus Module Object implementation and how they correspond to the functionality in the Anybus CompactCom 30 BACnet MS/TP.

Standard Objects:

6.2 Anybus Object (01h)

Category

Basic

Object Description

This object assembles all common Anybus data, and is described thoroughly in the general *Anybus CompactCom 40 Software Design Guide*.

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute
	Set_Attribute
	Get_Enum_String

Object Attributes (Instance #0)

(Consult the general *Anybus CompactCom 40 Software Design Guide* for further information.)

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Module type	Get	UINT16	0401h (Standard Anybus CompactCom 30)
2... 11	-	-	-	Consult the general Anybus CompactCom 30 Software Design Guide for further information.
12	LED colors	Get	struct of: UINT8 (LED1A) UINT8 (LED1B) UINT8 (LED2A) UINT8 (LED2B)	Value: Color: 01h Green 02h Red 01h Green 02h Red
13... 15	-	-	-	Consult the general Anybus CompactCom 30 Software Design Guide for further information.

6.3 Diagnostic Object (02h)

Category

Basic

Object Description

This object provides a standardized way of handling host application events & diagnostics, and is thoroughly described in the general Anybus CompactCom 30 Software Design Guide.

This module supports one instance of this object. This instance is reserved for major, unrecoverable events.

Supported Commands

Object:	Get_Attribute
	Create
	Delete
Instance:	Get_Attribute

Object Attributes (Instance #0)

#	Name	Access	Type	Value
1... 4	-	-	-	Consult the general Anybus CompactCom 30 Software Design Guide for further information.
11	Max no. of instances	Get	UINT16	1

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Severity	Get	UINT8	30h (Major, unrecoverable)
2	Event Code	Get	UINT8	Consult the general Anybus CompactCom 30 Software Design Guide for further information.

6.4 Network Object (03h)

Category

Basic

Object Description

For more information regarding this object, consult the general *Anybus CompactCom 30 Software Design Guide*.

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute
	Set_Attribute
	Get_Enum_String
	Map_ADI_Write_Area

It is not possible to map read process data in the Anybus CompactCom BACnet MS/TP, thus the standard command Map_ADI_Read_Area is not supported.

Object Attributes (Instance #0)

(Consult the general *Anybus CompactCom 30 Software Design Guide* for further information.)

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Network type	Get	UINT16	0099h
2	Network type string	Get	Array of CHAR	“BACnet MS/TP”
3	Data format	Get	ENUM	01h (MSB first)
4	Parameter data support	Get	BOOL	True
5	Write process data size	Get	UINT16	Current write process data size (in bytes) Updated on every successful Map_ADI_Write_Area. (Consult the general *Anybus CompactCom 30 Software Design Guide* for further information.)
6	Read process data size	Get	UINT16	0. (The Anybus CompactCom 30 BACnet MS/TP does not support read process data.)
7	Exception Information	Get	UINT8	Additional information available if the module has entered the EXCEPTION state. Value: Meaning: 00h No information available 01h The Get_All_BACnet_Object_Instances service request for analog value objects failed. 02h The Get_All_BACnet_Object_Instances service request for binary value objects failed. 03h The Get_All_BACnet_Object_Instances service request for multistate value objects failed. 04h An ADI mapped on process data could not be resolved as a BACnet object during start up
8... 10	-	-	-	Consult the general *Anybus CompactCom 30 Software Design Guide* for further information.

6.5 Network Configuration Object (04h)

Category

Extended

Object Description

This object holds network specific configuration parameters that may be set by the end user. A reset command (factory default) issued towards this object will result in all instances being set to their default values.

If the settings in this object do not match the configuration used, the Module Status LED will flash red to indicate a minor error.

See also...

- *Communication Settings, p. 21*

Supported Commands

Object:	Get_Attribute
	Reset
Instance:	Get_Attribute
	Set_Attribute
	Get_Enum_String

Object Attributes (Instance #0)

(Consult the general *Anybus CompactCom 40 Software Design Guide* for further information.)

Instance Attributes (Instance #1, MAC Address)

Advanced

Value is used after module reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“MAC Addr” (MS/TP node address) (Multilingual, see page *37*)
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	01h (one elements)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	UINT8	Min = 0, max =7Fh, default = 0 (in non volatile storage)

Instance Attributes (Instance #2, Baud Rate)

Advanced

Value is used after module reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Baud Rate" (MS/TP baud rate) (Multilingual, see page 37)
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	UINT8	Any change is valid after reset. Stored in non volatile memory Value — Meaning 0 — 9600 (default) 1 — 19200 2 — 38400 3 — 76800

Instance Attributes (Instance #3, Device Instance)

Advanced

Value is used after module reset.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Device inst" (Mapped to the Object_Identifier property in the Device Object) (Multilingual, see page 37)
2	Data type	Get	UINT8	06h (= UINT32)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	UINT32	Min = 0, max = 3FFFFEh, default = (module serial number & 3FFFFEh) (Stored in non volatile memory)

Instance Attributes (Instance #4, Process Active Timeout)

Advanced

Changes have immediate effect.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	"Process tmo" (Specifies the process active timeout in milliseconds) (Multilingual, see page 37)
2	Data type	Get	UINT8	05h (= UINT16)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	UINT16	Default = 0 (Stored in non volatile memory)

Instance Attributes (Instance #5, Max master)

Advanced

Changes have immediate effect.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“Max master” (Mapped to the Max_Master property of the Device Object) (Multilingual, see page 37)
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	UINT8	Min = 1, max = 7Fh, default = 7Fh

Instance Attributes (Instance #6, Max frames)

Advanced

Changes have immediate effect.

#	Name	Access	Data Type	Description
1	Name	Get	Array of CHAR	“Max frames” (Multilingual, see page 37)
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (read/write/shared access)
5	Value	Get/Set	UINT8	Min = 01h, max = FFh, default = 01h

Multilingual Strings

The instance names and enumeration strings in this object are multilingual, and are translated based on the current language settings as follows:

Instance	English	German	Spanish	Italian	French
1	MAC Addr	Geräteadresse	Dir. Dispos.	Indirizzo	Addr Produit
2	Baud rate	Baudrate	Vel. Comunica	Baud rate	Débit
3	Device inst	Geraetenumm-er	Istanc.Dispos	Dispositivo	Inst produit
4	Process tmo	Prozess Tmo	Tout Proceso	Tout Processo	Process tmo
5	Max master	Max Master	Max Maestro	Max. Master	Nb max maîtres
6	Max frames	Max Telegr.	Max Tramas	Max. messaggi	Nb Trames max

7 Host Application Objects

7.1 General Information

This chapter specifies the host application object implementation in the module. The objects listed here may be implemented within the host application firmware to expand the BACnet MS/TP implementation.

Standard Objects

- " Application Object (FFh)" (see Anybus CompactCom 30 Software Design Guide)
- "Application Data Object (FEh)" (see Anybus CompactCom 30 Software Design Guide)

Network Specific Objects:

- *BACnet Host Object (EFh), p. 39*

7.2 BACnet Host Object (EFh)

Object Description

This object implements BACnet specific features in the host application. If attribute #7 (Support Advanced Mapping) is enabled, the application can define the ADI mapping of the module to suit the application.

The implementation of this object is optional; the host application can support none, some, or all of the attributes specified below. The module will attempt to retrieve the values of these attributes during startup; if an attribute is not implemented in the host application, simply respond with an error message (06h, “Invalid CmdExt[0]”). In such cases, the module will use its default value.

If the module attempts to retrieve a value of an attribute not listed below, respond with an error message (06h, “Invalid CmdExt[0]”).

See also...

- Anybus CompactCom 40 Software Design Guide, “Error Codes”
- *Device Object, p. 9*

Supported Commands

Object:	Get_Attribute
	Get_ADI_By_BACnet_Object_Instance
	Get_ADI_By_BACnet_Name
	Get_All_BACnet_Object_Instances
	Get_BACnet_Object_Instance_By_ADI
Instance:	Get_Attribute
	Set_Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	“BACnet”
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

Instance Attributes (Instance #1)

#	Name	Access	Data Type	Default Value	Comment
1	Object Name	Get/Set	Array of CHAR	“CompactCom 40 BACnet/IP”	Changes the Object_Name property of the BACnet Device Object Max 64 bytes. Both Get and Set are optional. If Set is implemented Get is required.
2	Vendor Name	Get	Array of CHAR	“HMS Industrial Networks”	Changes the Vendor_Name property of the BACnet Device Object Max 64 bytes. Both Get and Set are optional. If Set is implemented Get is required.
3	Vendor Identifier	Get	UINT16	486	Changes the Vendor_Identifier property of the BACnet Device Object.
4	Model Name	Get	Array of CHAR	“CompactCom 40 BACnet/IP”	Changes the Model_Name property of the BACnet Device Object and the product name, displayed on the web pages and in HICP responses.
5	Firmware Revision	Get	Array of CHAR	The modules firmware revision	Changes the Firmware_Revision property of the BACnet Device Object Max 16 bytes.
6	Application_ Software_Version	Get	Array of CHAR	The modules firmware revision	Changes the Application_Software_Version property of the BACnet Device Object Max 16 bytes.
7	Support advanced mapping	Get	BOOL	False	If true, the application supports advanced BACnet to ADI mapping schema and must support all related commands.
8	Current date and time	Get/Set	Struct of:		Sanity checks will be done when this value is read. If invalid values are detected the whole struct will be set to 0. If the Anybus CompactCom 30 BACnet MS/TP receives a date/ time update from the network it will write updated date/time to this attribute.
			UINT16	0	Current year
			UINT8	0	Current month
			UINT8	0	Current day
			UINT8	0	Current hour
			UINT8	0	Current minute
			UINT8	0	Current second
9	Password	Get	Array of CHAR	“Admin”	Password used for the ReinitializeDevice and DeviceCommunicationControl commands. Max 20 bytes.

Command Details: Get_ADI_By_BACnet_Object_Instance

Category

Extended

Details

Command Code	10h
Valid for:	Object

Description

By setting the attribute Support advanced mapping (#7), all requests to BACnet data objects will be translated to ADIs using this service.

The service is used to translate from BACnet addressing to Anybus CompactCom addressing. Request to supported BACnet object classes are forwarded to the application, so that it can return the corresponding ADI.

For information about BACnet object classes, see *BACnet MS/TP Implementation, p. 8*.

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved)	
CmdExt[1]		
MsgData[0,1]	BACnet Object	BACnet object class
MsgData[2... 5]	BACnet Instance	-

- Response Details

Field	Contents	Comments
CmdExt[0]	(reserved)	
CmdExt[1]		
MsgData[0,1]	ADI	ADI that corresponds to the requested BACnet Object Instance

Command Details: Get_ADI_By_BACnet_Name

Category

Extended

Details

Command Code	11h
Valid for:	Object

Description

By setting the attribute Support advanced mapping (#7), all requests to BACnet data objects by name will be translated to ADIs using this service.

The service is used to translate from BACnet addressing to Anybus CompactCom addressing. Request to supported BACnet object classes are forwarded to the application, so that it can return the corresponding ADI.

For information about BACnet object classes, see *BACnet MS/TP Implementation, p. 8*.

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved)	
CmdExt[1]		
MsgData[0.... N]	BACnet Object_Name	This field holds a string containing the BACnet object name

- Response Details

Field	Contents	Comments
CmdExt[0]	(reserved)	
CmdExt[1]		
MsgData[0,1]	ADI	ADI that corresponds to the requested BACnet Object Name
MsgData[2,3]	BACnet object class	The BACnet object class that corresponds to the BACnet Object_Name requested
MsgData[4... 7]	BACnet Instance	The BACnet Instance that corresponds to the BACnet Object_ Name requested.

Command Details: Get_All_BACnet_Object_Instances

Category

Extended

Details

Command Code	12h
Valid for:	Object

Description

If the attribute Support advanced mapping (#7) is set, the Object_List attribute in the Device Object will be populated during initialization using this service. The response returns a bit array of 2040 entries. A set bit indicates that the corresponding instance is implemented within the application.

The example in the table below, shows the first two bytes in the application response. Instances 1, 4, 10, 12, and 13 are implemented in the application.

Byte	Value
0	12h (0001 0010b)
1	34h (0011 0100b)

For information about BACnet object classes, see *BACnet MS/TP Implementation, p. 8*.

By setting the attribute Support advanced mapping (#7), this service must be implemented. When the mapping of write process data has been performed, it is not possible for the module to know which ADI corresponds to which BACnet object identifier. This service finds that information by translating from an ADI to a BACnet object identifier.

For information about BACnet object classes, see *BACnet MS/TP Implementation, p. 8*.

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved)	
CmdExt[1]		
MsgData[0, 1]	BACnet Object	BACnet object class

- Response Details

Field	Contents	Comments
CmdExt[0]	(reserved)	
CmdExt[1]		
MsgData[0... 254]	Object List	Bit array indicating implemented BACnet objects corresponding to the requested BACnet object class.

Command Details: Get_BACnet_Object_Instance_By_ADI

Category

Extended

Details

Command Code	13h
Valid for:	Object

Description

If the attribute Support advanced mapping (#7) is set, this service must be implemented. When the mapping of write process data has been performed, it is not possible for the module to know which ADI corresponds to which BACnet object identifier. This service finds that information by translating from an ADI to a BACnet object identifier.

For information about BACnet object classes, see *BACnet MS/TP Implementation, p. 8*.

- Command Details

Field	Contents	Comments
CmdExt[0]	(reserved)	
CmdExt[1]		
MsgData[0, 1]	ADI	The ADI for which the application wants to find the BACnet object identifier.

- Response Details

Field	Contents	Comments
CmdExt[0]	(reserved)	
CmdExt[1]		
MsgData[0, 1]	BACnet object class	The BACnet object class used for the ADI supplied in the command. (UINT16)
MsgData[2... 5]	BACnet Instance	The BACnet instance corresponding to the ADI supplied in the command. (UINT32)

A Categorization of Functionality

The objects, including attributes and services, of the Anybus CompactCom and the application are divided into two categories: basic and extended.

A.1 Basic

This category includes objects, attributes and services that are mandatory to implement or to use. They will be enough for starting up the Anybus CompactCom and sending/receiving data with the chosen network protocol. The basic functions of the industrial network are used.

Additional objects etc, that will make it possible to certify the product also belong to this category.

A.2 Extended

Use of the objects in this category extends the functionality of the application. Access is given to the more specific characteristics of the industrial network, not only the basic moving of data to and from the network. Extra value is given to the application.

Some of the functionality offered may be specialized and/or seldom used. As most of the available network functionality is enabled and accessible, access to the specification of the industrial network may be required.

B Implementation Details

B.1 SUP-Bit Definition

The supervised bit (SUP) indicates that the network participation is supervised by another network device. In the case of BACnet MS/TP, this means that the SUP-bit is set when the time (ms) elapsed since the last BACnet request is less than the value of the parameter “Process Active Timeout”, if this parameter value is greater than zero.

B.2 Anybus State Machine

The table below describes how the Anybus state machine relates to the BACnet MS/TP network

Anybus State	Implementation	Comment
WAIT_PROCESS	The module stays in this state until a BACnet request arrives.	-
ERROR	N/A	-
PROCESS_ACTIVE	BACnet request(s) addressed to this module have been received within the last “Process Active Timeout” time.	• If no process active timeout value is specified (i.e. the parameter is set to 0), the module will remain in this state after the first BACnet request has been received. • The supervised bit is set when the module is in this state.
IDLE	N/A	-
EXCEPTION	Unexpected error, e.g. watchdog timeout etc.	MS LED turns red (to indicate a major fault) NS LED is off

B.2.1 Process Active Timeout

Whenever a read or write BACnet request is received, the Anybus CompactCom module will go from state WAIT_PROCESS to PROCESS_ACTIVE. If no new request is received during the specified process active timeout, the module will return to WAIT_PROCESS to wait for the next request. If the process active timeout is set to zero (default), the module will never time out, but stay in PROCESS_ACTIVE.

The timeout timer is disabled during data collection and enabled when the BACnet response is generated.

B.3 Application Watchdog Timeout Handling

Upon detection of an application watchdog timeout, the module will cease network participation and shift to state EXCEPTION. No other network specific actions are performed.

B.4 Implemented BACnet BIBBs

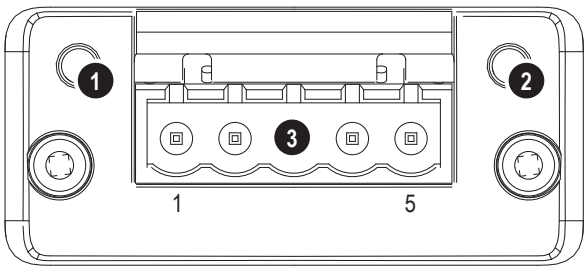
The Anybus CompactCom 30 BACnet MS/TP is implemented as a BACnet Application Specific Controller (B-ASC). To make the module eligible for certification as a B-ASC, the following BIBBs (BACnet Interoperability Building Blocks) are implemented:

BIBB	**Corresponding BACnet Service(s)**
Data Sharing-ReadProperty-B (DS-RP-B)	ReadProperty (Execute)
Data Sharing-ReadPropertyMultiple-B (DS-RPM-B)	ReadPropertyMultiple (Execute)
Data Sharing-WriteProperty-B (DS-WP-B)	WriteProperty (Execute)
Data Sharing-WritePropertyMultiple-B (DS-WPM-B)	WritePropertyMultiple (Execute)
Data Sharing-COV-B (DS-COV-B)	SubscribeCOV (Execute) ConfirmedCOVNotification (Initiate) UnConfirmedCOVNotification (Initiate)
Alarm and Event-Notification Internal-B (AE-N-I-B)	ConfirmedEventNotification (Initiate) UnConfirmedEventNotification (Initiate)
Alarm and Event-ACK-B (AE-ACK-B)	AcknowledgeAlarm (Execute)
Alarm and Event-Alarm Summary-B (AE-ASUM-B)	GetAlarmSummary (Execute)
Alarm and Event-Information-B (AE-INFO-B)	GetEventInformation (Execute)
Device Management-Dynamic Device Binding-A (DM-DDB-A)	Who-Is (Initiate) I-Am (Execute)
Device Management-Dynamic Device Binding-B (DM-DDB-B)	Who-Is (Execute) I-Am (Initiate)
Device Management-Dynamic Object Binding-B (DM-DDB-B)	Who-Has (Execute) I-Have (Initiate)
Device Management-Device Communication Control-B (DM-DCC-B)	DeviceCommunicationControl (Execute)
Device Management-TimeSynchronization-B (DM-TS-B)	TimeSynchronization (Execute)
Device Management-ReinitializeDevice-B (DM-RD-B)	ReinitializeDevice (Execute)

C Technical Specification

C.1 Front View

#	Item
1	Network Status LED
2	Module Status LED
3	BACnet connector



C.1.1 Network Status LED

LED State	Indication/Description
Off	No power
Green	• On-line, one or more BACnet messages have arrived • Module has active COV subscriptions • At least one value object has one or more events enabled
Green, flashing	On-line, waiting for first BACnet message.
Red	FATAL error
Red, flashing	• Connection timeout. No BACnet message has been received within the configured “process active timeout” time. • A COV or Alarm/Event notification could not be sent to its recipient.

A test sequence is performed on this LED during startup.

C.1.2 Module Status LED

LED State	Indication/Description
Off	No power
Green	Normal operation
Red	Major fault (state EXCEPTION, FATAL error etc.)
Red, flashing	Recoverable fault(s)

A test sequence is performed on this LED during startup.

C.1.3 BACnet Connector

Pin Number	Name	Description
1	Common	Signal common
2	Data-	Negative RS485 RxD/TxD
3	Shield	Cable shield
4	Data+	Positive RS485 RxD/TxD
5	Reserved, do not connect	

C.2 Functional Earth (FE) Requirements

In order to ensure proper EMC behavior, the Anybus CompactCom 30 BACnet MS/TP must be properly connected to functional earth via the FE pad / FE mechanism described in the general *Anybus CompactCom 30 Hardware Design Guide*. If the brick version is used, please make sure that the hardware is properly connected to FE.

HMS Industrial Networks does not guarantee proper EMC behavior unless these FE requirements are fulfilled.

C.3 Power Supply

C.3.1 Supply Voltage

The module requires a regulated 3.3V power source as specified in the general *Anybus CompactCom 30 Hardware Design Guide*.

C.3.2 Power Consumption

The Anybus CompactCom 30 BACnet MS/TP is designed to fulfil the requirements of a Class B module. For more information about the power consumption classification used on the Anybus CompactCom platform, consult the general *Anybus CompactCom M30 Hardware Design Guide*.

The current hardware design consumes up to 317 mA (RMS).

It is strongly advised to design the power supply in the host application based on the power consumption classifications described in the general Anybus CompactCom M40 Hardware Design Guide, and not on the exact power requirements of a single product.

In line with HMS policy of continuous product development, we reserve the right to change the exact power requirements of this product without prior notification. Note however that in any case, the Anybus CompactCom 30 BACnet MS/TP will remain a Class B module.

C.4 Environmental Specification

Consult the Anybus CompactCom 30 Hardware Design Guide for further information.

C.5 EMC Compliance

Consult the Anybus CompactCom 30 Hardware Design Guide for further information.

D Timing & Performance

D.1 General Information

This chapter specifies timing and performance parameters that are verified and documented for the Anybus CompactCom 30 BACnet MS/TP.

Category	Parameters	Comment
Startup Delay	T1, T2	Please consult the Anybus CompactCom Software 30 Design Guide, App. B.
NW_INIT Delay	T3	
Telegram Delay	T4	
Command Delay	T5	
Anybus Read Process Data Delay (Anybus Delay)	T6, T7, T8	
Anybus Write Process Data Delay (Anybus Delay)	T12, T13, T14	
Network System Read Process Data Delay (Network System Delay)	T9, T10, T11	Not supported by Anybus CompactCom 30 BACnet MS/TP.
Network System Write Process Data Delay (Network System Delay)	T15, T16, T17	

For further information, please consult the Anybus CompactCom 30 Software Design Guide.

D.2 Process Data

D.2.1 Overview

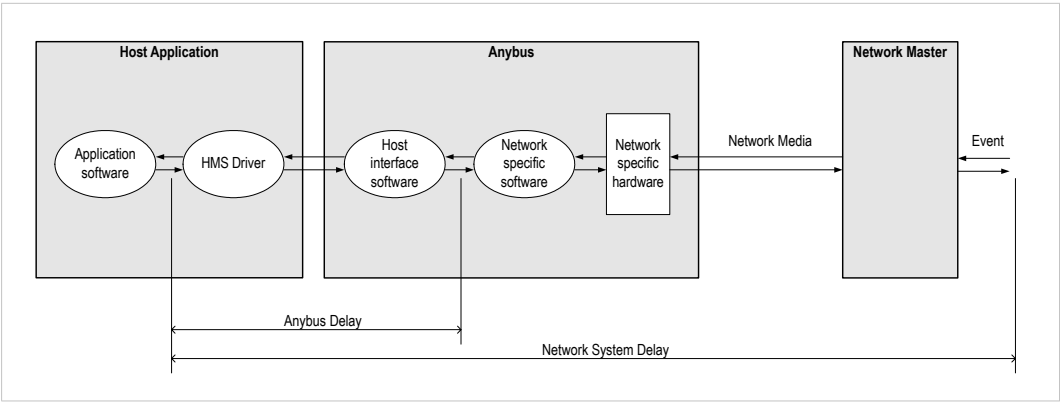


Fig. 3

D.2.2 Anybus Read Process Data Delay (Anybus Delay)

The Read Process Data Delay (labelled “Anybus delay” in the figure above) is defined as the time measured from just before new data is buffered and available to the Anybus host interface software, to when the data is available to the host application (just after the new data has been read from the driver).

Please consult the Anybus CompactCom Software 30 Design Guide, Appendix B, for more information.

D.2.3 Anybus Write Process Data Delay (Anybus Delay)

The Write Process Data Delay (labelled “Anybus delay” in the figure) is defined as the time measured from the point the data is available from the host application (just before the data is

written from the host application to the driver), to the point where the new data has been forwarded to the network buffer by the Anybus host interface software.

Please consult the Anybus CompactCom Software Design Guide, Appendix B, for more information.

D.2.4 Network System Read Process Data Delay (Network System Delay)

Read process data delay is not tested for Anybus CompactCom 30 BACnet MS/TP.

D.2.5 Network System Write Process Data Delay (Network System Delay)

The Network System Write Process Data Delay (labelled “Network System Delay” in the figure), is defined as the time measured from the time after the new data is available from the host application (just before the data is written to the driver) to when this data generates a corresponding event at the network master.

Parameter	Description	Typ.	Avg.	Max.	Unit.
T15	Network System Write Process Data delay, 8 ADIs (single UINT8)	39.2	37.5	63.2	ms
T16	Network System Write Process Data delay, 16 ADIs (single UINT8)	38.8	38.3	65.6	ms
T17	Network System Write Process Data delay, 32 ADIs (single UINT8)	36.8	39.4	86.0	ms

Conditions:

Parameter	Conditions
Application CPU	-
Timer system call interval	1 ms
Driver call interval	0.2... 0.3 ms
No.of ADIs (single UINT8) mapped to Process Data in each direction.	8, 16 and 32
Communication	Parallel
Telegram types during measurement period	Process Data only
Bus load, no. of nodes, baud rate etc.	Normal

Box 4126
300 04 Halmstad, Sweden

info@hms.se