Network Interface Appendix

Anybus® CompactCom PROFIBUS DP-V1

Doc.Id. HMSI-168-66
Rev. 2.22

HMS Industrial Networks
Mailing address: Box 4126, 300 04 Halmstad, Sweden
Visiting address: Stationsgatan 37, Halmstad, Sweden

E-mail: info@hms-networks.com
Web: www.anybus.com

2. Important User Information

This document is intended to provide a good understanding of the functionality offered by PROFIBUS DP-V1. The document only describes the features that are specific to the Anybus CompactCom PROFIBUS DP-V1. For general information regarding the Anybus CompactCom, consult the Anybus CompactCom design guides.

The reader of this document is expected to be familiar with high level software design, and communication systems in general. The use of advanced PROFIBUS DP-V1-specific functionality may require in-depth knowledge in PROFIBUS DP-V1 networking internal and/or information from the official PROFIBUS DP-V1 specifications. In such cases, the people responsible for the implementation of this product should either obtain the PROFIBUS DP-V1 specification to gain sufficient knowledge or limit their implementation in such a way that this is not necessary.

2.0.1 Liability

Every care has been taken in the preparation of this manual. Please inform HMS Industrial Networks AB of any inaccuracies or omissions. The data and illustrations found in this document are not binding. We, HMS Industrial Networks AB, reserve the right to modify our products in line with our policy of continuous product development. The information in this document is subject to change without notice and should not be considered as a commitment by HMS Industrial Networks AB. HMS Industrial Networks AB assumes no responsibility for any errors that may appear in this document.

There are many applications of this product. Those responsible for the use of this device must ensure that all the necessary steps have been taken to verify that the applications meet all performance and safety requirements including any applicable laws, regulations, codes, and standards.

HMS Industrial Networks AB will under no circumstances assume liability or responsibility for any problems that may arise as a result from the use of undocumented features, timing, or functional side effects found outside the documented scope of this 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.

The examples and illustrations in this document are included solely for illustrative purposes. Because of the many variables and requirements associated with any particular implementation, HMS Industrial Networks AB cannot assume responsibility for actual use based on these examples and illustrations.

2.0.2 Intellectual Property Rights

HMS Industrial Networks AB has intellectual property rights relating to technology embodied in the product described in this document. These intellectual property rights may include patents and pending patent applications in the US and other countries.

2.0.3 Trademark Acknowledgements

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

Warning:	This is a class A product. In a domestic environment this product may cause radio interference in which case the user may be required to take adequate measures.
ESD Note:	This product contains ESD (Electrostatic Discharge) sensitive parts that may be damaged if ESD control procedures are not followed. Static control precautions are required when handling the product. Failure to observe this may cause damage to the product.

Table of Contents

P. About This Document

For more information, documentation etc., please visit the HMS website, 'www.anybus.com'.

P.1 Related Documents

Document	Author
Anybus CompactCom 30 Software Design Guide	HMS
Anybus CompactCom 30 Hardware Design Guide	HMS
Anybus CompactCom Software Driver User Guide	HMS
Guideline Information & Maintenance Functions	PROFIBUS Nutzerorganisation e.V. (PNO)
Network Specification (IEC 61158)	IEC
Anybus CompactCom 30 Brick and withoiut Housing Design Guide	HMS

P.2 Document History

Summary of Recent Changes 2.21 ... 2.22)

Change	Page(s)
Updated support information	7
Moved front view and connector information to Technical Specification	76
Added recommendations to section E.2	78
Changed footnote in PROFIBUS connector description table	77
Clarified instance #1 of Network Configuration Object	32

Revision List

Revision	Date	Author(s)	Chapter(s)	Description
1.00	2005-10-06	PeP	-	First official version
1.01	2005-10-17	PeP	-	Minor modifications & corrections
1.10	2006-11-13	PeP	-	Updated for I&M and Alarm functionality
1.11	2007-04-03	PeP	-	Added notes regarding ADI vs. I/O module order
1.20	2007-04-20	PeP	3	Added information related to Call read/write functionality
1.21	2007-08-16	PeP	3, 4	Misc. minor corrections
1.30	2008-06-09	PeP	1, 3, 5, D	Major update
1.32	2008-07-02	PeP	-	Misc. minor corrections
2.00	2010-04-14	KeL	All	Change of concept
2.01	2011-02-10	KeL	P, 3, D, 4, 5	Minor update
2.02	2011-08-03	KaD	P, 2, 3, 5	Minor update
2.03	2011-12-08	KaD	P, 1, 4, 5	Minor update
2.04	2012-02-01	KaD	3, 5, D	Minor update
2.10	2012-09-12	KeL, KaD	1, 2	Added M12 connectors, minor updates
2.20	2012-12-04	KeL	1	Updated on brick version
2.21	2013-07-24	KeL	1	Info on pin placement for brick added
2.22	2015-05-12	KeL	P, E	Updates to tech spec

P.3 Conventions & Terminology

The following conventions are used throughout this manual:

- Numbered lists provide sequential steps
- Bulleted lists provide information, not procedural steps
- The terms 'Anybus' or 'module' refers to the Anybus CompactCom module.
- The terms 'host' or 'host application' refers to the device that hosts the Anybus module.
- Hexadecimal values are written in the format NNNNh or 0xNNNN, where NNNN is the hexadecimal value.
- A byte always consists of 8 bits.

P.4 Support

For general contact information and where to find support, please refer to the contact and support pages at www.anybus.com.

1. About the Anybus-CC PROFIBUS DP-V1

1.1 General

The Anybus CompactCom PROFIBUS DP-V1 communication module provides instant PROFIBUS connectivity via the patented Anybus CompactCom host interface. Any device that supports this standard can take advantage of the features provided by the module, allowing seamless network integration regardless of network type.

This product conforms to all aspects of the host interface for active modules defined in the Anybus CompactCom Hardware- and Software Design Guides, making it fully interchangeable with any other device following that specification. Generally, no additional network related software support is needed, however in order to take advantage of advanced network specific functionality, a certain degree of dedicated software support may be necessary.

1.2 Features

- PROFIBUS connector or M12 connectors
- Brick version
- Automatic baud rate detection
- Generic and PROFIBUS-specific diagnostic support
- User Parametrization Data support
- Set Slave Address support
- ADI access via DP-V1 read/write services
- Up to 368 bytes of I/O data
- Device identity customization
- Generic GSD file provided

2. Tutorial

2.1 Introduction

This chapter is a complement to the Anybus CompactCom Implementation Tutorial. The ABCC 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 certify for use on PROFIBUS DP-V1 networks.

2.2 Fieldbus Conformance Notes

- When using the default settings of all parameters, the module is precertified for network compliance. However, any parameter changes which require deviations from the standard GSD-file supplied by HMS will require recertification.
- As the name implies, the Anybus CompactCom PROFIBUS DP-V1 operates as a PROFIBUS DP-V1 device. It is not possible to certify this product as a PROFIBUS DP-V0 device.

  For further information, please contact HMS.

2.3 Certification

The following steps are necessary to perform to obtain a certification:

1. **Change PNO Ident Number:**

   The PNO Ident Number can be requested from PNO (PROFIBUS Nutzerorganisation e.V.). Replace the default PNO Ident Number with this. This is done by implementing the PROFIBUS DP-V1 object (FDh), instance 1, attribute 1, and returning the PNO Ident Number when receiving a Get_Attribute request.

2. **Add Node Address Information**

   If the host application does not set a valid node address by messaging the Network Configuration Object (04h), instance 1 ("Node Address"), the PROFIBUS Set Slave Address (SSA) service is enabled.

   If SSA functionality is enabled, it is mandatory to provide a mechanism for resetting the node address to its default value (126). This is because it is possible to lock the value from the network side. See "Set Slave Address" on page 20 for more information.

3. **Choose Buffer Mode:**

   Choose a Buffer Mode that suits the application. Only 5 - 7 should be used. Implement the PROFIBUS DP-V1 object (FDh), instance 1, attribute 6, and return the Buffer mode when receiving a Get_Attribute request.

4. **Change Manufacturer Id, Order Id, serial number and revision information (optional):**

   This is done by implementing the PROFIBUS DP-V1 object (FDh), instance 1, attributes 8 - 12, and returning the corresponding attributes when receiving a Get_Attribute request.

   The Manufacturer Id can be requested from PNO (PROFIBUS Nutzerorganisation e.V.).

5. **Modify the GSD file:**

   Modify the PROFIBUS DP-V1 GSD file so that it corresponds to the changes made above.

3. Basic Operation

3.1 General Information

3.1.1 Software Requirements

No additional network support code needs to be written in order to support the Anybus CompactCom PROFIBUS DP-V1, however certain restrictions must be taken into account:

- Due to the nature of the PROFIBUS networking system, at least one ADI must be mapped to Process Data.
- Only ADIs with instance numbers less than 65026 can be accessed acyclically from the network.
- By default, the first 68 bytes of an ADI may be accessed acyclically from the network. Larger sizes can be supported by changing the buffer mode, see "Buffer Modes" on page 11.
- By default, the module supports up to 152 bytes of Process Data. More data can be supported by changing the Buffer Mode, see "Buffer Modes" on page 11.
- The host application must be able to provide a response to an ADI request within the time period specified by the GSD-file ("DP-V1 related Keywords" on page 73, 'C1_Response_Timeout'), or the master will terminate the connection and reparameterize the slave. The default value for this parameter (i.e. the time specified by the generic GSD-file supplied by HMS) is 1 (one) second.
- The order in which ADIs are mapped to Process Data is significant and must be replicated in the PROFIBUS master when setting up the network communication (i.e. the I/O modules must be set up in the same order, and with the same size and direction, as the mapped ADIs). If not taken into account, the network connection establishment will fail and no communication will take place.
- The use of advanced PROFIBUS-specific functionality may require in-depth knowledge in PROFIBUS networking internals and/or information from the official PROFIBUS specification (IEC 61158). In such cases, the ones responsible for the implementation of this product should either obtain the PROFIBUS specification to gain sufficient knowledge or limit their implementation is such a way that this is not necessary.

IMPORTANT: *This document covers the functionality available in product firmware revision 2.10 or higher. For older firmware revisions, consult document revision 1.21.*

3.1.2 Electronic Data Sheet (GSD)

On PROFIBUS, the characteristics of a device is stored in an ASCII data file with the suffix GSD. This file is used by the PROFIBUS configuration tool when setting up the network.

HMS provides a generic GSD-file, which corresponds to the default settings in the module. However, due to the flexible nature of the Anybus CompactCom concept, it is possible to alter the behavior of the product in a way that invalidates the generic GSD-file.

See also...

- "Fieldbus Conformance Notes" on page 9
- "GSD file Customization" on page 59

3.2 Buffer Modes

The module features several internal memory buffers which are associated with certain network entities such as Process Data, Parametrization Data, Configuration Data etc. There are several different buffer configurations, a.k.a. Buffer Modes, which determines how the available memory shall be distributed across the internal buffers. This is an essential part of the module since it affects most aspects of its communication capabilities.

Buffer Mode	Process Data Buffer Size (bytes) (Input + Output)	Parametrization Data Buffer Size (bytes)	Configuration Data Buffer Size (bytes)	Class 1 Request Buffer Size (bytes)	Class 2 Request Buffer Size (bytes)	Max. no. of Diagnostic Events	Max. Channel Diagnostic Events	Identifier Related Diagnostics Supported	Alarm Supported	I&M Supported
Standard Mode (default)	152	160	152	68	68	5+1	5	Yes	No	Yes
Standard Mode 2	152	152	152	68	68	4+1	5	Yes	Yes	Yes
Parameter Data Mode 2	56	200	56	240	240	4+1	5	Yes	Yes	Yes
Process Data Mode 2	368[a]	16	16	68	68	1+1	0	No	Yes	Yes

a. On PROFIBUS, a maximum of 244 bytes can be mapped as either input or output data.

Note 1: The buffer size specifies the total memory area (both input and output data must be contained within its limit).

The available Buffer Modes have been revised somewhat compared to older revisions of the product. While not recommended for new designs, the "old" Buffer Modes are still available for backwards compatibility. For new designs, only use the modes presented in the table above.

'Standard Mode' (default) and 'Standard Mode 2' comply to the generic GSD-file supplied by HMS. By implementing the 'Buffer Mode' attribute (#6) of the PROFIBUS DP-V1 Object (FDh) into the host application, it is possible to redistribute the available memory to better suit particular application requirements. Note however that in such case, a custom GSD-file must be created, and fieldbus recertification is necessary.

Note: The maximum number of Diagnostic Events are written as 'n+1', where 'n' signifies the maximum number of diagnostic instances that can be created. An extra instance ('+1') is reserved to allow a Major Unrecoverable Event to be reported at any time.

See also...

- "PROFIBUS DP-V1 Object (FDh)" on page 47 (Attribute #6, 'Buffer mode')
- "GSD file Customization" on page 59

IMPORTANT: *If the end product is going to be certified for fieldbus compliance, it is strongly recommended to use a mode other than the default one (i.e. Standard Mode 2, Parameter Mode 2 or Process Data Mode 2). For example, applications with long (i.e. >1 ms) response times (i.e. devices with an inherent latency either due to serial communications or other factors) may cause certification issues due to excessive response times if the default mode is used. Other modes use a different buffer scheme internally which prevents this issue.*

IMPORTANT: *The buffer modes described in this section are available in firmware revision 2.10 or higher only. For older firmware revisions, consult document revision 1.21.*

3.3 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...

- **Node Address**

  See also...

  - "Network Configuration Object (04h)" on page 31

- **Baud rate**

  The baud rate is detected automatically by the module. The following baud rates are supported:

  - 9.6 kbps
  - 19.2 kbps
  - 45.45 kbps
  - 93.75 kbps
  - 187.5 kbps
  - 500 kbps
  - 1.5 Mbps
  - 3 Mbps
  - 6 Mbps
  - 12 Mbps

3.4 Device Identity

By default, the Anybus module appears as a generic HMS device with the following network identity:

Vendor Name	"HMS Industrial Networks"
Model Name	"Anybus CompactCom DPV1"
Ident Number	1811h

It is possible to customize the network identity information so that the Anybus module appears as a vendor specific implementation rather than a generic HMS product.

The PROFIBUS I&M-functionality (Identification & Maintenance) provides a standard way of gathering information about an I/O device. The I&M information is accessed by the master by means of the Call State Machine using DP-V1 read/write services.

By default, Anybus module supports I&M records 0... 4 for slot #0 (which is the device itself). Optionally, the host application can implement the 'Get_IM_Record'- and 'Set_IM_Record'-commands (PROFIBUS DP-V1 Object (FDh)) to support all I&M records for all slots.

See also...

- PROFIBUS Profile Guidelines Part 1: Identification & Maintenance Functions
- "Buffer Modes" on page 11
- "Network Configuration Object (04h)" on page 31
- "PROFIBUS DP-V1 Object (FDh)" on page 47, (Attribute #1, 'PNO Ident Number')
- "Command Details: Get_IM_Record" on page 53
- "Command Details: Set_IM_Record" on page 54
- "Device Identification" on page 60
- "Identification & Maintenance-related Keywords" on page 71

3.5 Data Exchange

3.5.1 Application Data Instances (ADIs)

ADIs can be accessed acyclically from the network using DP-V1 read/write services. The module translates these services into object requests towards the Application Data Object. If the host application responds with an error to such a request, the error code in the response will be translated to DP-V1 standard.

ADIs are mapped to slots and indexes as follows:

Correlation:

ADI = slot • 255 + index + 1
slot = (ADI - 1) / 255
index = (ADI - 1) MOD 255

Examples:

ADI	Slot	Index
1	0	0
255	0	254
65025	254	254

The length parameter in the DP-V1 request specifies the number of bytes to read/write.

- When reading more data than the actual size of the ADI, the response will only contain the actual ADI data, i.e. no padding on the data is performed by the module.
- When reading less data than the actual size of the ADI, only the requested amount of data is returned by the module.
- The maximum ADI data size that can be accessed acyclically is determined by the size of the Class 1 & Class 2 Request Buffers (default is 68 bytes).
- If attempting to read more data than supported by the Class 1 & Class 2 Request Buffers, the error code 'ADI truncated' will be returned.
- When writing to an ADI, the length parameter is not checked by the module, i.e. the host application must respond with an error if the length differs from the actual size of the requested ADI.

Note: Due to technical reasons, it is generally not recommended to use ADI numbers 1... 256, since this may cause trouble with certain PROFIBUS configuration tools.

See also...

- "Buffer Modes" on page 11
- "Error Handling" on page 58

3.5.2 Process Data

Mapping an ADI to Write Process Data results in PROFIBUS input data, and mapping an ADI to Read Process Data results in PROFIBUS output data. The maximum number of bytes that can be mapped depends on the size of the Process Data Buffer. If the host application tries to map more data than the Buffer Mode permits, the module will go into the EXCEPTION-state (exception code 06h) after 'Setup Complete'.

To guarantee consistency over an entire ADI, the ADI must not contain more than one element of a specific data type, since each element results by default in one identifier (a.k.a. 'module'). If consistency over an entire ADI which contains an array of elements is required, there are two possibilities:

- Implement the 'Configuration Data'-attribute (#3) in the PROFIBUS DP-V1 Object (FDh) and specify the configuration data manually.
- Use the network-specific ADI mapping commands in the Network PROFIBUS DP-V1 Object (0Bh)

In either case, a custom GSD-file must be created to support the size of the array.

Note: In Process Data Mode 2, the maximum number of ADI elements for Process Data is limited to 16 (max. modules/identifiers = 16). If additional elements are needed, either implement the 'Configuration Data'-attribute in the PROFIBUS DP-V1 Object (FDh) or use the network-specific ADI mapping commands in the Network PROFIBUS DP-V1 Object (0Bh).

See also...

- "Buffer Modes" on page 11
- "Network PROFIBUS DP-V1 Object (0Bh)" on page 40
- "PROFIBUS DP-V1 Object (FDh)" on page 47

3.6 Diagnostics

3.6.1 Standard Diagnostics

The Standard Diagnostics is handled automatically, with the exception of the following flags:

- **Ext Diag Overflow**

  This flag can be controlled by the host application via the 'Ext diag overflow'-attribute in the Additional Diagnostic Object (05h), and indicates that there are pending diagnostic events which couldn't be reported.

- **Static Diag Flag**

  This flag can be controlled by the host application via the 'Static Diag Flag'-attribute in the Additional Diagnostic Object (05h), and indicates that data from the slave is invalid.

3.6.2 Extended Diagnostics

Optionally, Extended Diagnostics can be supported via the generic Diagnostic Object (02h) and the Additional Diagnostic Object (05h). The Extended diagnostics consists of one or several Protocol Data Units (a.k.a. PDUs):

- Status PDU (See "Status PDU Implementation Details" on page 15)
- Identifier PDU (See "Channel and Identifier PDU Implementation Details" on page 15)
- Channel PDU (See "Channel and Identifier PDU Implementation Details" on page 15)

Note: When the PROFIBUS-master reads diagnostic data from the module, all pending events are reported; not only the ones that where recently added/removed.

Status PDU Implementation Details

Each instance within the generic Diagnostic Object (02h) corresponds to a Status PDU, which consists of the following components:

- **Status Specifier**

  The Status Specifier reflects the current state of the diagnostic event. When an event is reported for the first time, the Status Specifier value is 'Status appears' and upon deletion the value is 'Status disappears'. In between, the value is 'Status not changed'.

- **Status Type**

  The Status Type is fixed to 'Status Message' (81h).

- **Slot Number and Application Specific Data (a.k.a. Diagnostic User Data)**

  See "Diagnostic Object (02h)" on page 27 (NW Specific Extension, instance attribute #3).

Channel and Identifier PDU Implementation Details

Each instance within the Additional Diagnostic Object (05h) corresponds to a Channel PDU. Modules with diagnostic events are marked (i.e. set to one) in the Identifier-PDU.

The generation of Identifier-related diagnostics is enabled by implementing the 'Size of identifier related diagnostics'-attribute in the PROFIBUS DP-V1 Object (FDh). If this attribute is not implemented or set to zero, no Identifier_PDU will be generated.

3.7 Parametrization Data Handling

3.7.1 General Information

The master identifies itself with the slaves by sending Parametrization Data, specifying how the slave shall operate (i.e. Master address, PNO-ID, Sync/Freeze capabilities etc.).

The Parametrization Data consists of three parts:

	DP Standard Parameters	DP-V1 Status Bytes	User Parametrization Data
Size	7 bytes	3 bytes	Dynamic
Defined by	IEC 61158-6	IEC 61158-6	Host application
Evaluated by	Anybus module	Anybus module	Host application
Supported in the Generic HMS GSD-file	Yes	Yes	No

As seen in the table above, User Parametrization Data is not supported by default. Optionally, User Parametrization Data can be supported by implementing the 'Parametrization Data'-attribute in the PROFIBUS Object (FDh). In such case, the generic GSD-file supplied by HMS cannot be used.

The maximum amount of User Parametrization Data that can be handled by the module is determined by the size of the Parametrization Data Buffer, see "Buffer Modes" on page 11.

See also...

- "Buffer Modes" on page 11
- "PROFIBUS DP-V1 Object (FDh)" on page 47 (Attribute #2, 'Parametrization Data')
- "Parametrization-related Keywords" on page 69

3.7.2 Validation

The DP Standard Parameters and the DP-V1 Status Bytes are always evaluated by the Anybus module, while the User Parametrization Data must be evaluated by the host application. This is handled through the 'Parametrization Data'-attribute in the PROFIBUS DP-V1 Object (FDh).

- **'Parametrization Data'-attribute not implemented**

  In order for the Parametrization Data to be accepted by the module, it must not contain any User Parametrization Data.

- **'Parametrization Data'-attribute implemented**

  The host application must evaluate the contents of the 'Parametrization Data'-attribute and provide a suitable response.

  - To accept the Parametrization Data, respond with no error code.

  - To reject the Parametrization Data, respond with one of the following error codes:

    NOT_ENOUGH_DATA
    TOO_MUCH_DATA
    OUT_OF_RANGE
    INVALID_STATE
    NO_RESOURCES

3.8 Configuration Data Handling

3.8.1 General Information

The Anybus module determines its Expected Configuration Data based on the ADI mapping process. Alternatively, it can be specified by the host application by implementing the Get service of the 'Configuration Data'-attribute in the PROFIBUS DP-V1 Object (FDh).

The maximum amount of configuration data that can be handled by the module is determined by the size of the Configuration Data Buffer, see "Buffer Modes" on page 11.

See also...

- "PROFIBUS DP-V1 Object (FDh)" on page 47 (Attribute #3, 'Configuration Data')
- "I/O-related Keywords" on page 66
- "Definition of Modules" on page 68

3.8.2 Validation

The 'Check config behaviour'-attribute in the PROFIBUS DP-V1 Object (FDh) controls the functionality of the Chk_Cfg service. It can take either the value 0 or the value 1.

For more information about the 'Check config behaviour'-attribute, see...

"PROFIBUS DP-V1 Object (FDh)" on page 47

"Advanced" on page 50

'Check config behaviour' equals 0 (default):

Using the Chk_Cfg service, the PROFIBUS master will send the Actual Configuration Data needed for the application to the module. The module will compare the Actual Configuration Data with the Expected Configuration Data. In case of a mismatch, the module will send the Actual Configuration Data to the host application for further evaluation, using the Set service of the 'Configuration Data'-attribute in the PROFIBUS DP-V1 Object (FDh).

Implementing the 'Configuration Data'-attribute in the PROFIBUS DP-V1 Object (FDh) in the host application is optional.

- **'Configuration Data'-attribute not implemented**

  In case of a mismatch, the Actual Configuration Data must be rejected.

- **'Configuration Data'-attribute implemented**

  The host application must evaluate the contents of the 'Configuration Data'-attribute.

  - To accept the Configuration Data, respond with a no error code.

    Important: If the new configuration affects the Process Data mapping, it is important that the host application updates the Process Data before responding. Failure to observe this may cause erroneous data to be sent to the bus on the next state shift. Preferably, choose to reject the Actual Configuration Data and adapt to it by restarting the Anybus module and then revise the Process Data map and/or the Expected Configuration Data. Also note that the new configuration must exist in the GSD-file of the product.

- To reject the Configuration Data, respond with one of the following error codes:
  NOT_ENOUGH_DATA
  TOO_MUCH_DATA
  OUT_OF_RANGE
  INVALID_STATE
  NO_RESOURCES

‘Check config behaviour’ equals 1:

- **If the master has enabled Check_Cfg_Mode in the parameter data:**
  The Actual Configuration Data is accepted if its length equals the length of the Expected Configuration Data.
  All other configurations are rejected.
- **If the master has disabled Check_Cfg_Mode in the parameter data:**
  The Actual Configuration Data is accepted if it is exactly equal to the Expected Configuration Data.
  All other configurations are rejected.

3.9 Alarm Handling

3.9.1 General Information

To enable the alarm functionality, the host application must implement the PROFIBUS DP-V1 Object (FDh) and the 'Alarm settings'-attribute. The host application specifies which alarm types that shall be supported, and if some types is required to be enabled by the master. An application-specific GSD-file must be created which agrees to the 'Alarm settings'-attribute. The Anybus module supports Type Mode (only one alarm of each type may be pending for acknowledgement) and Sequence Mode (up to 32 alarms of the same or different type may be pending at the same time).

See also...

- "Buffer Modes" on page 11
- "Additional Diagnostic Object (05h)" on page 35
- "PROFIBUS DP-V1 Object (FDh)" on page 47

3.9.2 Alarm Notification Handling

Alarm notifications are issued by means of the 'Alarm_Notification'-command (Additional Diagnostic Object (05h)), which contains the properties of the alarm and up to 4 bytes of optional application-specific alarm data. The notification may be of different alarm types, not necessarily indicating an error, but for example a state change or plug of a module into a slot. The module responds to the command when the master has read the alarm, i.e. the response time depends on the PROFIBUS cycle time. The response is tagged with a Sequence Number which is used to identify the alarm later on.

Note: Alarm notifications may only be issued in PROCESS_ACTIVE and IDLE states.

See also...

- "Command Details: Alarm_Notification" on page 38

3.9.3 Alarm Acknowledge Handling

When the master acknowledges an alarm, the module forwards the acknowledgement by means of the 'Alarm_Acknowledge'-command (PROFIBUS DP-V1 Object (FDh)). This command holds details about the alarm (Slot number, Alarm type and Sequence number).

Note: The module issues this command for informational purposes only, i.e. the module will confirm the acknowledgement to the master as long as it matches a pending Alarm Notification, even if the command is rejected by the host application.

See also...

- "Command Details: Alarm_Acknowledge" on page 55

3.10 Set Slave Address

The module supports the 'Set Slave Address'-service, which enables a master or configuration tool to set the node address from the network.

This service features a flag which specifies whether or not it is allowed to change the device address from the network again at a later stage. If the service is accepted, the module saves the value of this flag in non-volatile memory; the only way to restore it again is by performing a Factory Default-reset on the Network Configuration Object (consult the general Anybus CompactCom Software Design Guide for more information). This behavior is mandatory for the application to pass PROFIBUS network certification.

The module will accept new settings received via this service under the following conditions:

- The 'Device Address'-attribute is set to a value higher than 125
- The 'SSA Enabled'-attribute (PROFIBUS DP-V1 Object (FDh)) is set to TRUE (or not implemented)
- The module is not in Data Exchange
- The module is addressed with the correct Ident Number
- No previous 'Set Slave Address'-request prevents the module from accepting the new settings

See also...

- "PROFIBUS DP-V1 Object (FDh)" on page 47 (Attribute #4, 'SSA Enabled')
- "Supported DP Features" on page 61

Note: It is possible to disable support for this service by implementing the 'SSA Enabled'-attribute in the PROFIBUS DP-V1 Object (FDh). In such a case, a new GSD-file must be created, and fieldbus re-certification is necessary.

3.11 Parameter Read/Write with Call

3.11.1 General Information

Parameter Read/Write with Call enables addressing of ADIs based on instance numbers rather than Slot and Index. This is useful if the ADI implementation is primarily designed for a linear addressing scheme as used on most other networks. It may also prove useful when using masters with limited addressing capabilities for slot 0, since such masters may otherwise have trouble accessing ADI instances 1... 256.

Unlike the standard DP-V1 Read/Write service, Parameter Read/Write with Call uses the 'Call' application service. On the PROFIBUS telegram level, the 'Parameter Read with Call' service request consists of a standard DP-V1 header, a Call header, and the ADI number. When received by the module, this is translated into a standard object request towards the application data object.

3.11.2 Parameter Read with Call Handling

The 'Parameter Read with Call'-service request looks as follows:

Byte #	Contents	Field Name	Value	Notes
1	DP-V1 Header	Function no.	5Fh	Indicates a DP-V1 Write service
2		Slot	00h	(must not be set to FFh)
3		Index	FFh	(fixed)
4		Length	06h	Size of telegram (Call Header + ADI number)
5	Call Header	Ext. Function no.	08h	Call service
6		(reserved)	00h	(reserved, set to zero)
7		Sub-index (high)	00h	Subindex 0002h, used when reading
8		Sub-index (low)	02h	
9	ADI number	ADI (high)	0000... FFFFh	Number of the ADI which shall be read
10		ADI (low)		

Upon reception, the module translates this into a 'Get_Attribute'-request towards the Application Data Object. In the same manner, the response from the host application will be transformed into an appropriate response telegram on PROFIBUS as follows:

'Parameter Read with Call'-response:

Byte #	Contents	Field Name	Value	Notes
1	DP-V1 Header	Function no.	5Eh	Indicates a DP-V1 Read service
2		Slot	00h	(must not be set to FFh)
3		Index	FFh	(fixed)
4		Length	06... F0h	Size of telegram (Call Header + ADI number + data)
5	Call Header	Ext. Function no.	08h	Call service
6		(reserved)	00h	(reserved, set to zero)
7		Sub-index (high)	00h	Subindex 0002h, used when reading
8		Sub-index (low)	02h	
9	ADI number	ADI (high)	0000... FFFFh	Number of the ADI which shall be read
10		ADI (low)		
11...n	Data	(actual data)	-	Data returned from the host application

3.11.3 Parameter Write with Call Handling

The 'Parameter Write with Call'-telegram looks as follows:

Byte #	Contents	Field Name	Value	Notes
1	DP-V1 Header	Function no.	5Fh	Indicates a DP-V1 Write service
2		Slot	00h	(must not be set to FFh)
3		Index	FFh	(fixed)
4		Length	06... F0h	Size of telegram (Call Header + ADI number + Data)
5	Call Header	Ext. Function no.	08h	Call service
6		(reserved)	00h	(reserved, set to zero)
7		Sub-index (high)	00h	Subindex 0001h, used when writing
8		Sub-index (low)	01h	
9	ADI number	ADI (high)	0000... FFFFh	Number of the ADI which shall be written
10		ADI (low)		
11... n	Data	(actual data)	-	Data which will be sent to the host application

Upon reception, the module translates this into a 'Set_Attribute'-request towards the Application Data Object. In the same manner, the response from the host application will be transformed into an appropriate response telegram on PROFIBUS as follows:

'Parameter Write with Call'-response:

Byte #	Contents	Field Name	Value	Notes
1	DP-V1 Header	Function no.	5Eh	Indicates a DP-V1 Read service
2		Slot	00h	(must not be set to FFh)
3		Index	FFh	(fixed)
4		Length	06h	Size of telegram (Call Header + ADI number)
5	Call Header	Ext. Function no.	08h	Call service
6		(reserved)	00h	(reserved, set to zero)
7		Sub-index (high)	00h	Subindex 0001h, used when writing
8		Sub-index (low)	01h	
9	ADI number	ADI (high)	0000... FFFFh	Number of the ADI which shall be read
10		ADI (low)		

3.12 Identification & Maintenance (I&M)

General Information

Identification & Maintenance (I&M) provides a standard way of gathering information about an I/O device. The I&M information can be accessed by the PROFIBUS Master by means of acyclic Record Data Read/Write services.

Default I&M0 Information:

IM Manufacturer ID	010Ch (HMS Industrial Networks)
IM Order ID	'ABCC-DPV1'
IM Serial Number	(unique serial number, set during manufacturing)
IM Hardware Revision	(Anybus hardware revision ID, set during manufacturing)
IM Software Revision	(Anybus software revision, set during manufacturing)
IM Revision Counter	(Revision counter)
IM Profile ID	F600h (Generic Device)
IM Profile Specific Type	0004h (Communication Module)
IM Version	0101h
IM Supported	001Eh (IM0..4 supported)

Optionally, the host application can customize the information for these I&M entries, or implement the support for the 'Get_IM_Record'- and 'Set_IM_Record'-commands to support all I&M record for all slots.

See also...

- "PROFIBUS DP-V1 Object (FDh)" on page 47
- "Command Details: Get_IM_Record" on page 53
- "Command Details: Set_IM_Record" on page 54
- "Network Configuration Object (04h)" on page 31

I&M Data Structures

The I&M records uses the following data structures.

Record	Content	Size	Description
I&M0	Manufacturer Id	2 bytes	PROFIBUS DP-V1 Object (FDh), attribute #8 ('Vendor ID/I&M Vendor ID')
	Order Id	20 bytes	PROFIBUS DP-V1 Object (FDh), attribute #9 ('I&M Order ID')
	Serial number	16 bytes	PROFIBUS DP-V1 Object (FDh), attribute #10 ('I&M Serial number')
	Hardware revision	2 bytes	PROFIBUS DP-V1 Object (FDh), attribute #11 ('I&M Hardware revision)'
	Software revision	4 bytes	PROFIBUS DP-V1 Object (FDh), attribute #12 ('I&M Software revision')
	Revision counter	2 bytes	PROFIBUS DP-V1 Object (FDh), attribute #13 ('I&M Revision counter')
	Profile Id	2 bytes	PROFIBUS DP-V1 Object (FDh), attribute #14 ('I&M Profile ID')
	Profile specific type	2 bytes	PROFIBUS DP-V1 Object (FDh), attribute #15 ('I&M Profile specific type')
	IM version	2 bytes	PROFIBUS DP-V1 Object (FDh), attribute #16 ('I&M Version')
	IM supported	2 bytes	PROFIBUS DP-V1 Object (FDh), attribute #17 ('I&M supported')
I&M1	Tag Function	32 bytes	Network Configuration Object (04h), instance #3 ('I&M Tag Function')
	Tag Location	22 bytes	Network Configuration Object (04h), instance #4 ('I&M Tag Location')
I&M2	Installation date	16 bytes	Network Configuration Object (04h), instance #5('I&M Installation Date')
I&M3	Descriptor	54 bytes	Network Configuration Object (04h), instance #16 ('I&M Descriptor')
I&M4[a]	Signature	54 bytes	Default: All bytes set to zero (00h)

a. Data of this field must only be accessed from the network by the IO Controller/Supervisor.

See also...

- "PROFIBUS DP-V1 Object (FDh)" on page 47

4. Anybus Module Objects

4.1 General Information

This chapter specifies the Anybus Module Object implementation and how they correspond to the functionality in the Anybus CompactCom PROFIBUS DP-V1.

Standard Objects:

Network Specific Objects:

4.2 Anybus Object (01h)

Category

Basic

Object Description

This object assembles all common Anybus data, and is described thoroughly in the general Anybus CompactCom 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 Software Design Guide for further information).

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Module type	Get	UINT16	0401h (Standard Anybus CompactCom)
2... 11	-	-	-	Consult the general Anybus CompactCom 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 Software Design Guide for further information.

4.3 Diagnostic Object (02h)

Category

Extended, advanced

Object Description

This object provides a standardised way of handling host application events & diagnostics, and is thoroughly described in the general Anybus CompactCom Software Design Guide. In the case of PROFIBUS, each instance created in this object adds one Status PDU to the Extended Diagnostics.

Supported Commands

Object:	Get_Attribute Create Delete
Instance:	Get_Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1... 4	-	-	-	Consult the general Anybus CompactCom Software Design Guide for further information.
11	Max no. of instances	Get	UINT16	The maximum number of pending events/instances depends on the current Buffer Mode, see "Buffer Modes" on page 11.

Instance Attributes (Instance #1...n)

Extended

#	Name	Access	Type	Value
1	Severity	Get	UINT8	Consult the general Anybus CompactCom Software Design Guide for further information.
2	Event Code	Get	UINT8	

Advanced

#	Name	Access	Type	Value
3	NW specific extension[a]	Get	Array of UINT8: Element: Contents: 0 Slot Number 1 (reserved) 2...7 Application Specific Field	

a. The use of this attribute is optional; if not implemented, the module will use slot no. 0 (zero), and the 'Severity' and 'Event Code'-attributes will be reported as application specific data. If implemented, a custom GSD-file may be required.

See also...

- "Diagnostics" on page 15
- "Status Diagnostic Messages" on page 72

4.4 Network Object (03h)

Category

Basic

Object Description

This object contains network specific data for the module. It also controls the mapping of ADIs to the process data part of the telegrams. For more information, consult the general Anybus CompactCom Software Design Guide.

Note: The order in which ADIs are mapped to Process Data is significant and must be replicated in the PROFIBUS master when setting up the network communication.

Supported Commands

Object:	Get_Attribute
Instance:	Get_Attribute Set_Attribute Get_Enum_String Map_ADI_Write_Area Map_ADI_Read_Area

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Value
1	Network type	Get	UINT16	0005h
2	Network type string	Get	Array of CHAR	'PROFIBUS DP-V1'
3	Data format	Get	ENUM	0001h (MSB first)
4	Parameter Data support[a]	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[b] (and Map_ADI_Specified_Write_Area)
6	Read process data size	Get	UINT16	Current read process data size (in bytes) Updated on every successful Map_ADI_Read_Area[b] (and Map_ADI_Specified_Read_Area)
7	Exception Information	Get	UINT8	Additional PROFIBUS-specific exception information is presented here in case the Anybus module has shifted to the EXCEPTION-state.
8... 10	-	-	-	Consult the general Anybus CompactCom Software Design Guide for further information.

a. This attribute indicates if the network supports acyclic data services and must not be confused with PROFIBUS Parametrization Data.
b. Consult the general Anybus CompactCom Software Design Guide for further information.

Exception Information

This attribute holds additional information when the Anybus module shifts to the EXCEPTION state.

#	Value
00h	(no information available)
01h	Too much Configuration Data; the default Process Data map resulted in more Configuration Data than permitted by the current Buffer Mode.
02h	Too much Configuration Data; the 'Configuration Data'-attribute (PROFIBUS DP-V1 Object (FDh)) holds more data than permitted by the current Buffer Mode.
03h	Configuration error; The 'Configuration Data'-attribute does not match the actual Process Data map.
04h	Too much Process Data; the amount of mapped Process Data exceeds the capabilities of the current Buffer Mode.
05h	Implementation error; the host application has called Map_ADI_Specified_Read/Map_ADI_Specified_Write and specified Expected Configuration Data ('Configuration Data'-attribute, PROFIBUS DP-V1 Object (FDh))
06h	Implementation error; support for the Set Slave Address (SAS) telegram has been disabled ('SSA Enable'-attribute, PROFIBUS DP-V1 Object (FDh)), but no valid device address has been supplied by the application.
07h	Implementation error; invalid Buffer Mode specified ('Buffer Mode'-attribute, PROFIBUS DP-V1 Object (FDh))

4.5 Network Configuration Object (04h)

Category

Basic, advanced

Object Description

This object contains network specific configuration parameters that may be configured by the end user.

Note 1: A 'Reset'-command towards this object will cause the module to revert all instance values to their factory default values.

Supported Commands

Object:	Get_Attribute Reset
Instance:	Get_Attribute Set_Attribute Get_Enum_String

Object Attributes (Instance #0)

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

Instance Attributes (Instance #1, ‘Node Address’)

Basic

The module must be assigned a unique node address (a.k.a. device address) in order to be able to communicate on the PROFIBUS network. The application can assign the module a node address in the range from 0 to 125, or allow the device address to be set from the PROFIBUS master by setting the node address to a value of 126 or above.

Address 126 is reserved for SSA functionality, see “Set Slave Address” on page 20. This feature allows the device address to be set from the PROFIBUS master. Any value larger than 126 will be interpreted as 126.

The value set by the application is saved in non volatile memory, and will be accessible during setup, for example when the module is restarted. Attribute #5 will after setup show the actual node address assigned:

1. The application has written e.g. the value 13 to attribute #5. After a restart this value (13) will be returned both during setup and after setup is complete.

2. The application has written 126 to attribute #5 (SSA is enabled). A PROFIBUS master has assigned a valid address (0 - 125) from the network. After a restart the value 126 will be returned before setup is completed. After setup is completed, the address assigned by the master will be returned.

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	‘Node address’
2	Data type	Get	UINT8	04h (= UINT8)
3	Number of elements	Get	UINT8	01h (one element)
4	Descriptor	Get	UINT8	07h (get/set/shared access)
5	Value	Get/Set	UINT8	Value:: Meaning: 0... 125 Node address 126 ... 255 Get node address using SSA (default)[b]

a. Multilingual, see “Multilingual Strings” on page 34.
b. Support for SSA can be globally disabled by implementing the PROFIBUS object, see “PROFIBUS DP-V1 Object (FDh)” on page 47)

Instance Attributes (Instance #3, ‘Function Tag’)

Advanced

This instance holds the I&M parameter ‘Function Tag’ for slot 0 (i.e. the device itself).

Default after factory reset is ‘ ’.

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	‘Function tag’
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	20h (32 elements)
4	Descriptor	Get	UINT8	07h (get/set/shared access)
5	Value[b]	Get/Set	Array of CHAR	String describing the functionality or task of the device in the network. Pad with 0x20 (space) up to number of elements.

a. Multilingual, see “Multilingual Strings” on page 34.
b. Stored in nonvolatile memory.

Instance Attributes (Instance #4, ‘Location Tag’)

Advanced

This instance holds the I&M parameter ‘Location Tag’ for slot 0 (i.e. the device itself).

Default after factory reset is ‘ ’.

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	‘Location tag’
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	16h (22 elements)
4	Descriptor	Get	UINT8	07h (get/set/shared access)
5	Value[b]	Get/Set	Array of CHAR	String describing the location of the device in the network. Pad with 0x20 (space) up to number of elements.

a. Multilingual, see “Multilingual Strings” on page 34.
b. Stored in non-volatile memory.

Instance Attributes (Instance #5, ‘Installation Date’)

Advanced

This instance holds the I&M parameter ‘Installation Date’ for slot 0 (i.e. the device itself).

Default after factory reset is ‘ ’.

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	‘Install. date’
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	10h (16 elements)
4	Descriptor	Get	UINT8	07h (get/set/shared access)
5	Value[b]	Get/Set	Array of CHAR	String in the format ‘YYYY-MM-DD hh:mm’ (e.g. ‘2005-06-15 19:42’)

a. Multilingual, see “Multilingual Strings” on page 34.
b. Stored in nonvolatile memory.

Instance Attributes (Instance #6, ‘Description’)

Advanced

This instance holds the I&M parameter ‘Descriptor/Comment’ for slot 0 (i.e. the device itself).

Default after factory reset is ‘ ’.

#	Name	Access	Type	Description
1	Name[a]	Get	Array of CHAR	‘Description’
2	Data type	Get	UINT8	07h (= CHAR)
3	Number of elements	Get	UINT8	36h (54 elements)
4	Descriptor	Get	UINT8	07h (get/set/shared access)
5	Value[b]	Get/Set	Array of CHAR	String with a description of the device; free to be used as desired by the end-user. Pad with 0x20 (space) up to number of elements.

a. Multilingual, see "Multilingual Strings" on page 34.
b. Stored in non-volatile memory.

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	Node address	Geräteadresse	Direcc nodo	Indirizzo	Adresse
3	Function tag	Funktion	Función	Descr. Funz.	Fonction
4	Location tag	Position	Locación	Descr. Locaz.	Placement
5	Install. date	Install. Tag	Fecha Inst.	Data install.	Date Installé
6	Description	Beschreibung	Descripción	Descrizione	Description

4.6 Additional Diagnostic Object (05h)

Category

Extended

Object Description

This object provides advanced PROFIBUS-specific diagnostic support according to IEC 61158-6 (Channel related diagnostics, Static Diag and Ext Diag Overflow bits). It also handles alarms. Each instance created in this object adds one Channel PDU to the Extended Diagnostics. Note that the use of this object is optional.

Supported Commands

Object: Get_Attribute
Set_Attribute
Create (See "Command Details: Create" on page 37)
Delete (See "Command Details: Delete" on page 38)
Alarm_Notification (See "Command Details: Alarm_Notification" on page 38)

Instance: Get_Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'Additional diagnostics'
2	Revision	Get	UINT8	02h
3	Number of instances	Get	UINT16	Corresponds to no. of pending events
4	Highest instance no.	Get	UINT16	Highest created instance no.
11	Max. no of instances	Get	UINT8	The maximum number of pending events/instances depends on the current Buffer Mode, see "Buffer Modes" on page 11
12	Ext Diag Overflow	Get/Set	BOOL	Value: Meaning: False Ext Diag Overflow is cleared (default) True Ext Diag Overflow is set See also "Standard Diagnostics" on page 15
13	Static Diag	Get/Set	BOOL	Value: Meaning: False Static Diag is cleared (default) True Static Diag is set See also "Standard Diagnostics" on page 15

Instance Attributes (Instance #n)

Extended

#	Name	Access	Data Type	Description
1	Module number	Get	UINT8	Specifies the source of the diagnosis (range: 0... 63)
2	I/O type	Get	UINT8	Value:Meaning: 1: Input channel 2: Output channel 3: Bidirectional channel
3	Channel number	Get	UINT8	Specifies the source of the diagnosis (range: 0... 63)
4	Channel type	Get	UINT8	Value:Meaning: 0: Unspecific 1: 1 bit 2: 2 bit 3: 4 bit 4: Byte 5: Word 6: 2 words
5	Error type	Get	UINT8	Value:Meaning: 1: Short circuit 2: Undervoltage 3: Overvoltage 4: Overload 5: Overtemperature 6: Line break 7: Upper limit value exceeded 8: Lower limit value exceeded 9: Error 10...15: (reserved) 16...31: (manufacturer specific)

Command Details: Create

Category

Extended

Details

Command Code: 03h

Valid for: Object

Description

This command creates a new diagnostic event / instance.

- **Command**

Field	Contents	Description
MsgData[0]	Module number	Specifies the source of the diagnosis (range: 0... 63)
MsgData[1]	I/O type	Value:Meaning: 1: Input channel 2: Output channel 3: Bidirectional channel
MsgData[2]	Channel number	Specifies the source of the diagnosis (range: 0... 63)
MsgData[3]	Channel type	Value:Meaning: 0: Unspecific 1: 1 bit 2: 2 bit 3: 4 bit 4: Byte 5: Word 6: 2 words
MsgData[4]	Error type	Value:Meaning: 1: Short circuit 2: Undervoltage 3: Overvoltage 4: Overload 5: Overtemperature 6: Line break 7: Upper limit value exceeded 8: Lower limit value exceeded 9: Error 10... 15: (reserved) 16... 31: (manufacturer specific)

- **Response (Success)**

Field	Contents
MsgData[0]	The number of the created instance (low byte)
MsgData[1]	The number of the created instance (high byte)

- **Response (Error)**

Error Code	Meaning
01h	Invalid Module number
02h	Invalid I/O type

Error Code	Meaning
03h	Invalid Channel number
04h	Invalid Channel type
05h	Invalid Error type

Command Details: Delete

Category

Extended

Details

Command Code: 04h

Valid for: Instance

Description

This command deletes a previously created diagnostic event / instance.

- **Command**

Field	Contents
CmdExt[0]	The number of the instance to delete (low byte)
CmdExt[1]	The number of the instance to delete (high byte)

- **Response**

  -

Command Details: Alarm_Notification

Category

Extended

Details

Command Code: 10h

Valid for: Object

Description

This command issues an Alarm Notification to the master.

Note: This command is only allowed in PROCESS_ACTIVE or IDLE state

- **Command**

Field	Contents	Description
CmdExt[0]	Alarm type	Value:Meaning: 1: Diagnostic Alarm 2: Process Alarm 3: Pull Alarm 4: Plug Alarm 5: Status Alarm 6: Update Alarm 32.. 126: Manufacturer-specific Alarm (other) (reserved)
MsgData[0]	Slot Number	The slot associated by the alarm.
MsgData[1]	Alarm Specifier	Value:Meaning: 0: No further differentiation 1: Alarm appears 2: Alarm disappears and slot is OK 3: Alarm disappears but the slot is still disturbed
MsgData[2]	Additional Acknowledge	Value:Meaning: TRUE: The 'Additional ack'-bit in the Alarm specifier byte will be set FALSE: The 'Additional ack'-bit in the Alarm specifier byte will be cleared
MsgData[3...]	User Data (up to 4 bytes)	Application specific alarm data

- **Response (Success)**

Field	Contents	Contents
MsgData[0]	Sequence number	Used to identify the alarm acknowledge.

Object Specific Error Codes

Error Code	Meaning
01h	Invalid module number
02h	Invalid I/O type
03h	Invalid channel number
04h	Invalid channel type
05h	Invalid error type
06h	Invalid alarm specifier
07h	Alarm type disabled
08h	Too many active alarms (Sequence mode enabled)
09h	Alarm type already active (Type mode enabled)

4.7 Network PROFIBUS DP-V1 Object (0Bh)

Category

Extended

Object Description

This object complements the Network Object (03h).

Supported Commands

Object: Get_Attribute

Instance: Get_Attribute
Map_ADI_Specified_Write_Area
(see "Command Details: Map_ADI_Specified_Write_Area" on page 41)

Map_ADI_Specified_Read_Area
(see "Command Details: Map_ADI_Specified_Read_Area" on page 42)

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	"Network PROFIBUS DP-V1"
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

Instance Attributes (Instance #1)

This instance has no attributes. It supports all instance commands.

Command Details: Map_ADI_Specified_Write_Area

Category

Extended

Details

Command Code: 10h

Valid for: Object

Description

This command is functionally equivalent to Map_ADI_Write_Area in the Network Object (03h), with the exception of certain additional PROFIBUS-specific parameters specifying custom module identifiers, which is particularly useful when mapping ADIs with multiple elements.

Note 1: Mixed calls to Map_ADI_Specified_Write_Area and Map_ADI_Write_Area are not permitted.

Note 2: The module definitions must be represented in the GSD-file.

Note 3: The order in which ADIs are mapped to Process Data is significant and must be replicated in the PROFIBUS master when setting up the network communication.

- **Command**

Field	Contents
CmdExt[0... 1]	(See specification for Map_ADI_Write_Area)
Msg_Data[0... 3]	
Msg_Data[4... n]	Module definitions; PROFIBUS identifier(s) which defines this particular ADI in the expected configuration data. The maximum number of bytes, along with module definitions for already mapped ADIs, must not exceed the size of the Configuration Data Buffer. Module definitions must agree with the size of the mapped ADI and the direction of the process data.

- **Response (Success)**

Field	Contents
Msg_Data[0]	Offset of the mapped ADI from the start of the Write Process Data.

See also...

- "Network Object (03h)" on page 29
- "Object Specific Error Codes" on page 43
- Map_ADI_Write_Area (consult the Anybus CompactCom Software Design Guide)
- "Definition of Modules" on page 68

Command Details: Map_ADI_Specified_Read_Area

Category

Extended

Details

Command Code: 11h

Valid for: Object

Description

This command is functionally equivalent to Map_ADI_Read_Area in the Network Object (03h), with the exception of certain additional PROFIBUS-specific parameters specifying custom module identifiers, which is particularly useful when mapping ADIs with multiple elements.

Note 1: Mixed calls to Map_ADI_Specified_Read_Area and Map_ADI_Read_Area are not permitted.

Note 2: The module definitions must be represented in the GSD-file.

Note 3: The order in which ADIs are mapped to Process Data is significant and must be replicated in the PROFIBUS master when setting up the network communication.

- **Command**

Field	Contents
CmdExt[0... 1]	(See specification for Map_ADI_Read_Area)
Msg_Data[0... 3]	
Msg_Data[4... n]	Module definitions; PROFIBUS identifier(s) which defines this particular ADI in the expected configuration data. The maximum number of bytes, along with module definitions for already mapped ADIs, must not exceed the size of the Configuration Data Buffer. Module definitions must agree with the size of the mapped ADI and the direction of the process data.

- **Response (Success)**

Field	Contents
Msg_Data[0]	Offset of the mapped ADI from the start of the Read Process Data.

See also...

- "Network Object (03h)" on page 29
- "Object Specific Error Codes" on page 43
- Map_ADI_Read_Area (consult the Anybus CompactCom Software Design Guide)
- "Definition of Modules" on page 68

Object Specific Error Codes

Code	Meaning
01h	Invalid data type
02h	Invalid number of elements
03h	Invalid total size
04h	Invalid order number
05h	Invalid command sequence
06h	Invalid module definition
07h	Total size of expected configuration data exceeds the size of the configuration data buffer.

4.8 PROFIBUS DP-V0 Diagnostic Object (10h)

Category

Advanced

Object Description

This object provides completely transparent extended diagnostic data.

Note: Instance #1 can not be used in conjunction with the standard Diagnostic Object (page 27).

Supported Commands

Object: Get_Attribute
Set_Attribute

Instance: Get_Attribute
Set_Attribute

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'PROFIBUS DP-V0 Diagnostic'
2	Revision	Get	UINT8	01h
3	Number of instances	Get	UINT16	1
4	Highest instance no.	Get	UINT16	1
12	Ext Diag Overflow	Get/Set	BOOL	Value: Meaning: False Ext Diag Overflow is cleared (default) True Ext Diag Overflow is set See also "Standard Diagnostics" on page 15
13	Static Diag	Get/Set	BOOL	Value: Meaning: False Static Diag is cleared (default) True Static Diag is set See also "Standard Diagnostics" on page 15

Instance Attributes (Instance #1)

Advanced

#	Name	Access	Data Type	Description
1	Diagnostic data	Get/Set	Array of UINT8	Array of bytes containing the diagnostic data. See "Diagnostic Data" on page 45.

Diagnostic Data

Byte #	Contents	Description
0	Extended Diagnostic Data Flag	Value: Meaning: 0 No extended diagnostic data 1 Diagnostic data is reported as extended diagnostic data
1 - n	Diagnostic Data	Amount depends on the initialized buffer mode, see "Buffer Modes" on page 11. The buffer mode sets the total length for the diagnostic data buffer. The user specific part can be set with this attribute and the length of the data is up to the length specified by the Buffer Mode.

5. Host Application Objects

5.1 General Information

This chapter specifies the host application object implementation in the module. The objects listed here may optionally be implemented within the host application firmware to expand the PROFIBUS implementation.

Standard Objects:

- Application Object (see Anybus CompactCom Software Design Guide)
- Application Data Object (see Anybus CompactCom Software Design Guide)

Network Specific Objects:

- "PROFIBUS DP-V1 Object (FDh)" on page 47

5.2 PROFIBUS DP-V1 Object (FDh)

Category

Basic, extended, advanced

Object Description

This object implements PROFIBUS-specific settings in the host 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 start-up; 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]").

Note 1: During operation, the host application must always be able to respond to requests from the module. Respond either with the requested data or an adequate error message. Never leave a request from the module unattended.

Note 2: Altering the default settings within this object may require a new GSD file, which in turn requires fieldbus recertification.

See also...

- "Generic GSD file provided" on page 8
- "Electronic Data Sheet (GSD)" on page 10
- "GSD file Customization" on page 59
- Guideline Information & Maintenance functions
- Anybus CompactCom Software Design Guide, "Error Codes"

Supported Commands

Object:	Get_Attribute Get_IM_Record (See "Command Details: Get_IM_Record" on page 53) Set_IM_Record (See "Command Details: Set_IM_Record" on page 54)
Instance:	Get_Attribute Set_Attribute Alarm_Acknowledge (See "Command Details: Alarm_Acknowledge" on page 55)

Object Attributes (Instance #0)

#	Name	Access	Data Type	Value
1	Name	Get	Array of CHAR	'PROFIBUS DP-V1'
2	Revision	Get	UINT8	02h
3	Number of instances	Get	UINT16	0001h
4	Highest instance no.	Get	UINT16	0001h

Instance Attributes (Instance #1)

Basic

#	Name	Access	Type	Comment
1	PNO Ident Number	Get	UINT16	Ident Number (default = 1811h) See also... - "Device Identity" on page 12 - "Device Identification" on page 60
6	Buffer mode	Get	UINT8	This attribute specifies the Buffer Mode: Value: Meaning: 0 Standard Mode (default) 1 (obsolete) Parameter Data Mode 2 (obsolete) Process Data Mode 3 (obsolete) Alarm Mode 4 (obsolete) Process Data I&M Mode 5 Standard Mode 2 6 Parameter Data Mode 2 7 Process Data Mode 2 **Note 1:** Modes 0... 4 are included for backwards compatibility only and are not recommended for new designs. **Note 2:** Other values will cause the module to enter the EXCEPTION state (exception code 07h). See also... - "Buffer Modes" on page 11

Extended

#	Name	Access	Type	Comment
3	Configuration Data	Get/Set	Array of UINT8	This attribute is used both to specify the Expected Configuration Data, and for evaluation of the Actual Configuration Data. See also... - "Configuration Data Handling" on page 17 - "Definition of Modules" on page 68 **Note:** If this attribute doesn't match the Process Data map, or if it doesn't fit into the Configuration Data Buffer, the module will enter the EXCEPTION state (exception code 06h)
4	SSA Enabled	Get	BOOL	This attribute enables/disables 'Set Slave Address'-support. Value: Meaning: True Enabled (default) False Disabled See also... - "Set Slave Address" on page 20 - "Supported DP Features" on page 61
5	Size of identifier related diagnostics[a]	Get	UINT8	This attribute specifies the length of the identifier-related diagnostics (excluding header byte) Value: Meaning: 0 No identifier related diagnostics (default) 1... 8 Size of identifier related diagnostics (each byte represents 8 modules) See also... - "Diagnostics" on page 15

#	Name	Access	Type	Comment
7	Alarm settings	Get	Struct of: UINT8 UINT8 BOOL	This attribute specifies alarm related information. If not implemented, the Alarm functionality will be disabled. See also... - "Buffer Modes" on page 11 - "Alarm Structure" on page 51 - "Alarm-related Keywords" on page 75
8	Manufacturer ID	Get	UINT16	Device Manufacturer ID (default = 010Ch) See also... - "Device Identity" on page 12 - Guideline Information & Maintenance functions
9	Order ID	Get	Array of CHAR	Device Order ID, up to 20 characters (default = 'ABCC-DPV1') See also... - "Device Identity" on page 12 - Guideline Information & Maintenance functions
10	Serial Number	Get	Array of CHAR	Serial number, up to 16 characters. If not implemented, the module defaults to its production assigned serial number.
11	Hardware Revision	Get	UINT16	Hardware revision of the device. Value: Meaning: 0... FFFFh Hardware revision FFFFh Indicates profile specific information See also... - Guideline Information & Maintenance functions
12	Software Revision	Get	Struct of: CHAR (Type) UINT8 (Major) UINT8 (Minor) UINT8 (Build)	Software revision of the device. If not implemented, the module defaults to the Anybus firmware revision. See also... - "Software Revision Structure" on page 52 - Guideline Information & Maintenance functions

a. In Process Data Mode (See "Buffer Modes" on page 11), identifier-related diagnostics is not supported.

Advanced

#	Name	Access	Type	Comment
2	Parametrization Data	Set	Array of UINT8	The module attempts to forward the Parametrization Data to this attribute during network startup. See also... - "Parametrization Data Handling" on page 16 - "Diagnostic-related Keywords" on page 70
13	Revision Counter	Get	UINT16	Revision counter; a changed value of this counter marks a change of the hardware or of its parameters. If not implemented, the module defaults to 0 (zero). See also... - Guideline Information & Maintenance functions
14	Profile ID[a]	Get	UINT16	Specifies the Profile ID. Default:F600h (Generic Device) See also... - Guideline Information & Maintenance functions
15	Profile specific type[a]	Get	UINT16	Specifies the Profile specific type. Default:0004h (Communication Module) See also... - Guideline Information & Maintenance functions
16	IM version	Get	Struct of: UINT8 (Major) UINT8 (Minor)	Should only be used if the application shall support another I&M version than 1.1. In such case, it is mandatory to implement Get_IM_Record and Set_IM_Record. If not implemented, the module defaults to v1.1. See also... - Guideline Information & Maintenance functions
17	IM supported	Get	UINT16	Bit field specifying which I&M records that are supported for slot 0. See also... - "IM Supported Structure" on page 52
18	IM header	Get	Array of UINT8	This header will be sent to the master together with I&M0... 4 for slot 0. The master is then required to supply this information back with each write request to these records. If not implemented, the header will be filled with zeroes. The maximum number of elements in the array is 10.
19	Check config behaviour	Get	UINT8	Controls the functionality of the Check cfg service. 0 (Default): The actual cfg is accepted if it equals the expected cfg. Otherwise the actual cfg data will be sent to the application for further evaluation. 1: All configurations with the same data length as expected are accepted if the master has enabled Check_Cfg_Mode in the Prm data. Other configurations are rejected. If Check_Cfg_Mode is disabled only cfgs which equals the expected cfg are accepted. See "Configuration Data Handling" on page 17.

a. These attributes do not affect the behavior of the module, since it does not handle profiles.

Alarm Structure

Member	Contents	Comments
UINT8	Supported Alarm Types: Bit:Type: 0... 1 (reserved) 2 Update Alarm 3 Status Alarm 4 Manuf. Specific Alarm 5 Diagnostic Alarm 6 Process Alarm 7 Pull & Plug Alarm	This bit field specifies which alarm types that shall be supported, i.e. which alarm types that can be enabled by the master during parameterization. The host application may only issue alarms which have been defined as supported in this bit field. Default value: 0.
UINT8	Required Alarm Types: Bit:Type: 0... 1 (reserved) 2 Update Alarm 3 Status Alarm 4 Manuf. Specific Alarm 5 Diagnostic Alarm 6 Process Alarm 7 Pull & Plug Alarm	This bit field specifies which alarms that must be enabled by the master during parametrization. Default value: 0.
BOOL	Sequence Mode Supported: Value:Meaning: True Sequence Mode & Type Mode False Type Mode only	In Type Mode, only one of each supported alarm types can be unacknowledged at the same time. In Sequence Mode, multiple alarms of the same - or different - type can be unacknowledged simultaneously. The master sets the max. no. unacknowledged alarms during parametrization. Default value: False.

Note: The GSD file must be adjusted to match the settings specified in this structure.

See also...

- "Alarm-related Keywords" on page 75

Software Revision Structure

Member	Contents	Comments
CHAR	Letter:Meaning: 'V' Official release 'R' Revision 'P' Prototype 'U' Under test (field test) 'T' Test device	-
UINT8	Major version Range:0... 255	Functional enhancement
UINT8	Minor version Range:0... 255	Bug fix
UINT8	Internal change Range:0... 255	No impact on function

IM Supported Structure

This is a bit field where each bit corresponds to an I&M parameter that shall be supported for slot 0. A set bit means that the corresponding I&M parameter is supported.

Bit	I&M Record	Default
0	Profile Specific IM	Disabled
1	IM1	Enabled
2	IM2	Enabled
3	IM3	Enabled
4	IM4	Enabled
other	-	(reserved)

Note: I&M0 is mandatory and cannot be disabled.

Command Details: Get_IM_Record

Category

Advanced

Details

Command Code: 10h

Valid for: Object

Description

This command is sent to the host application when the master (Class 1 or Class 2) asks for an I&M Record other than I&M0... 4 for slot 0, and for all I&M Records for slots other than 0. If the command is rejected, the original I&M-request will be rejected as well.

Note: Implementation of this command is optional unless attribute #16 ('IM version') has been implemented.

- **Command**

Field	Contents
CmdExt[0]	I&M Record index (0... 199)
CmdExt[1]	(reserved, ignore)
Msg_Data[0]	Slot number (0... 254)
Msg_Data[1... 3]	(reserved, ignore)

- **Response (Success)**

Field	Contents
Msg_Data[0... 9]	I&M user specific header.
Msg_Data[10... 63]	I&M Record; I&M parameters associated with the request

See also...

- "Command Details: Set_IM_Record" on page 54
- "Instance Attributes (Instance #1)" on page 48 (Attribute #16, 'IM version')
- Guideline Information & Maintenance functions

Command Details: Set_IM_Record

Category

Advanced

Details

Command Code: 11h

Valid for: Object

Description

This command is sent to the host application when the master (Class 1 or Class 2) updates for an I&M Record other than I&M0... 4 for slot 0, and for all I&M Records for slots other than 0. If the command is rejected, the original I&M-request will be rejected as well.

Note: Implementation of this command is optional unless attribute #16 ('IM version') has been implemented.

- **Command**

Field	Contents
CmdExt[0]	I&M Record index (0... 199)
CmdExt[1]	(reserved, ignore)
Msg_Data[0]	Slot number (0... 254)
Msg_Data[1... 3]	(reserved, ignore)
Msg_Data[4... 13]	I&M user specific header.
Msg_Data[14... 67]	I&M Record; I&M parameters associated with the request

- **Response (Success)**

  -

See also...

- "Command Details: Get_IM_Record" on page 53
- "Instance Attributes (Instance #1)" on page 48 (Attribute #16, 'IM version')
- Guideline Information & Maintenance functions

Command Details: Alarm_Acknowledge

Category

Extended

Details

Command Code: 12h

Valid for: Object

Description

This command is issued when the master (Class 1) has acknowledged an Alarm Notification.

Note: Implementation of this command is optional.

- **Command**

Field	Contents	Comments
CmdExt[0]	Alarm type	Refer to IEC 61158-6, 'Alarm Notification'.
CmdExt[1]	Sequence number	Identifies the alarm together with the alarm type. Range 0... 31.
Msg_Data[0]	Slot number	Slot associated with the alarm.

- **Response (Success)**

  -

See also...

- "Instance Attributes (Instance #1)" on page 48 (Attribute #7, 'Alarm settings')
- IEC 61158-6 (Alarm Notification)
- Guideline Information & Maintenance functions

A. Categorization of Functionality

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

A.3 Advanced

The objects, attributes and services that belong to this group offer specialized and/or seldom used functionality. Most of the available network functionality is enabled and accessible. Access to the specification of the industrial network is normally required.

B. Implementation Details

B.1 SUP-Bit Definition

The supervised bit (SUP) indicates that the network participation is supervised by another network device. For PROFIBUS, this bit is set when any of the following conditions are fulfilled.

- Parametrization and Configuration Data has been accepted (i.e. MS0 connection established)
- An MS2 connection is open

B.2 Anybus State Machine

The table below describes how the Anybus state machine relates to the PROFIBUS network.

State	Description
WAIT_PROCESS	No MS0 connection DP state = Power-On/WaitPrm/WaitCfg MS2 connection may be open
ERROR	Not used
PROCESS_ACTIVE	Master Mode = Operate DP State = DataExchange MS0 connection established MS2 connection may be open
IDLE	Master Mode = Clear DP State = DataExchange MS0 connection established MS2 connection may be open
EXCEPTION	MS0, MS1 and MS2 connections will be closed. The module will enter this state in the following cases: - Invalid Device Address and 'SSA Enabled'=FALSE - Invalid Buffer Mode - Size of 'Configuration Data'-attribute is larger than the size of the Configuration Data Buffer - Major Unrecoverable event created in Diagnostic Object - Configuration Data does not match the mapped Process Data

B.3 Watchdog Behavior (Application Stopped)

If the application watchdog expires, the module will enter the 'EXCEPTION' state, terminate all open PROFIBUS connections (MS0, MS1 and MS2) and leave the network.

C. Error Handling

Translation of Anybus Error Codes

When a DP-V1 request is received from the network, the module translates this request into an object request to the Application Data Object. When such requests are rejected by the host application, the error code in the response is translated to DP-V1 standard as follows:

Anybus Error Code	Resulting DP-V1 Error Codes			
	Error Decode	Error Code 1		Error Code 2
		Error Class	Error Code	
'Unsupported Object'	DP-V1 (128)	Access	Invalid Area	Same as the Anybus error code
'Unsupported Instance'		Access	Invalid Index	
'Unsupported Command'		Access	Access Denied	
'Invalid CmdExt0'		Access	Invalid Parameter	
'Invalid CmdExt1'		Access	Invalid Parameter	
'Attribute not Set-able'		Access	Access Denied	
'Attribute not Get-able'		Access	Access Denied	
'Too Much Data'		Access	Write Error Length	
'Not Enough Data'		Access	Write Error Length	
'Out of range'		Access	Invalid Range	
'Invalid Sate'		Access	State Conflict	
'Out of Resources'		Resource	Resource Busy	
'Object Specific Error'		Application	User Specific (10)	
(All other Anybus error codes)		Application	User Specific (11)	

Other Errors

Error	Resulting DP-V1 Error Codes			
	Error Decode	Error Code 1		Error Code 2
		Error Class	Error Code	
ADI truncated	DP-V1 (128)	Application	User Specific (12)	0
No free source IDs		Resource	Resource busy	0

D. GSD file Customization

D.1 General

The GSD file specifies the characteristics of the device, and is used by the PROFIBUS configuration tool when setting up the network.

HMS provides a generic GSD file, which corresponds to the default settings in the module. However, due to the flexible nature of the Anybus CompactCom concept, it is possible to alter the behavior of the product in a way that invalidates the generic GSD file. In such case, a custom GSD file must be created, and fieldbus recertification is necessary.

This chapter is intended to provide a brief overview of the GSD entries that may need alteration, and how they correspond to settings within the Anybus module. Some of the entries should not be changed, and the others are divided in the same way as the objects and object attributes, into the groups Basic, Extended and Advanced.

For further information, consult the Specification for PROFIBUS Device Description and Device Integration Volume 1: GSD (order. no. 2.122).

Note: The user is expected to have sufficient knowledge in the PROFIBUS networking system to understand the concepts involved when performing the changes specified in this chapter. In case of uncertainties, send the customized GSD file to HMS for verification.

D.2 Device Identification

General

By default, the module will appear as a generic Anybus implementation ('Anybus CompactCom DPV1') from HMS Industrial Networks (PROFIBUS ident no. 1811h).

However, the identity of the module can be customized to appear as a vendor specific implementation by creating a custom GSD-file and implementing the 'PNO Ident Number'-attribute in the PROFIBUS DP-V1 Object (FDh).

Contact PNO to obtain a unique Ident Number.

GSD-file Entries

```
; Device identification
Vendor_Name         = "<vendor>"
Model_Name          = "<product>"
Revision            = "<prod_rev>"
Ident_Number        = <ident_no>
Protocol_Ident      = 0                 ; DP protocol
Station_Type        = 0                 ; Slave device
FMS_supp            = 0                 ; FMS not supported
Slave_Family        = 0                 ; General device
Hardware_Release    = "Version <hw_rev>"
Software_Release    = "Version <sw_rev>"
```

Basic

Setting	Description
<vendor>	Vendor name as text (e.g. "HMS Industrial Networks")
<product>	Product name as text (e.g. "Anybus CompactCom DPV1")
<prod_rev>	Product revision (major.minor) (e.g. "1.01")
<ident_no>	PNO Ident Number in HEX. Written as 0xNNNN, where NNNN is the hexadecimal value.
<hw_rev>	Hardware revision (major.minor) (e.g."Version 1.00")
<sw_rev>	Software revision (major.minor) (e.g."Version 1.00")

Related Information

Information	Page(s)
Device Identity	12
PROFIBUS DP-V1 Object (FDh) (Attribute #1)	47
-	-

D.3 Supported Hardware Features

General

Do not change the standard settings.

GSD file Entries

```
; Supported hardware features
Redundancy        = 0         ; not supported
Repeater_Ctrl_Sig = 2         ; TTL
24V_Pins          = 0         ; not connected
Implementation_Type= "NP30"
```

Setting	Unit	Description
-	-	-

Related Information

Information	Page(s)
-	-

D.4 Supported DP Features

General

-

D.4.1 GSD file Entries

```
; Supported DP features
Freeze_Mode_supp  = 1
Sync_Mode_supp    = 1
Auto_Baud_supp    = 1
Set_Slave_Add_supp= <SSA>
Fail_Safe         = 1
```

Extended

Setting	Description
<SSA>	This value must be set to match the 'SSA enable'-attribute in the PROFIBUS DP-V1 Object (FDh): 0: 'SSA enabled'-attribute set to FALSE 1: 'SSA enabled'-attribute set to TRUE (or not implemented)

Related Information

Information	Page(s)
Set Slave Address	20
PROFIBUS DP-V1 Object (FDh) (Attribute #4)	47
-	-

D.5 Supported Baud Rates

General

Do not change the standard settings.

GSD file Entries

```
; Supported baud rates
9.6_supp          = 1
19.2_supp         = 1
45.45_supp        = 1
93.75_supp        = 1
187.5_supp        = 1
500_supp          = 1
1.5M_supp         = 1
3M_supp           = 1
6M_supp           = 1
12M_supp          = 1
```

Setting	Unit	Description
-	-	-

Related Information

Information	Page(s)
-	-

D.6 Maximum Responder Time for Supported Baud Rates

General

Do not change the standard settings.

GSD file Entries

```
; Maximum responder time for supported baud rates
MaxTsdr_9.6       = 15
MaxTsdr_19.2      = 15
MaxTsdr_45.45     = 15
MaxTsdr_93.75     = 15
MaxTsdr_187.5     = 15
MaxTsdr_500       = 15
MaxTsdr_1.5M      = 25
MaxTsdr_3M        = 50
MaxTsdr_6M        = 100
MaxTsdr_12M       = 200
```

Setting	Unit	Description
-	-	-

Related Information

Information	Page(s)
-	-

D.7 Maximum Polling Frequency

General

If using the recommended buffer modes 5, 6 or 7, it is generally recommended to set a Min_Slave_Intervall of 1 (i.e. 0.1 ms). In all other modes, including the default mode (i.e. 'Standard Mode'), the GSD file default value (6) can be used.

GSD file Entries

```
; Maximum polling frequency
Min_Slave_Intervall= 6        ; 0.6 ms
```

Setting	Unit	Description
-	-	-

Related Information

Information	Page(s)
-	-

D.8 I/O-related Keywords

General

-

GSD file Entries

```
; I/O related keywords
Modular_Station   = 1
Max_Module        = <module>
Max_Input_Len     = <input>
Max_Output_Len    = <output>
Max_Data_Len      = <total>
Modul_Offset      = 1
```

Note: Buffer Modes marked as 'obsolete' in the table below shall be regarded as obsolete and are thus not recommended for new designs.

Basic

Setting	Unit	Description
<module>	bytes	This value must be set within the range imposed by the current Buffer Mode as follows: Buffer Mode: — Min: — Max: Standard Mode — 0 — 152 *(Parameter Data Mode - obsolete)* — 0 — 72 *(Process Data Mode - obsolete)* — 0 — 16 *(Alarm Mode - obsolete)* — 0 — 152 *(Process Data I&M Mode - obsolete)* — 0 — 16 Standard Mode 2 — 0 — 152 Parameter Data Mode 2 — 0 — 56 Process Data Mode 2 — 0 — 16
<input>	bytes	This value must be set within the range imposed by the current Buffer Mode as follows: Buffer Mode: — Min: — Max: Standard Mode — 0 — 152 *(Parameter Data Mode - obsolete)* — 0 — 72 *(Process Data Mode - obsolete)* — 0 — 244 *(Alarm Mode - obsolete)* — 0 — 152 *(Process Data I&M Mode - obsolete)* — 0 — 244 Standard Mode 2 — 0 — 152 Parameter Data Mode 2 — 0 — 56 Process Data Mode 2 — 0 — 244
<output>	bytes	This value must be set within the range imposed by the current Buffer Mode as follows: Buffer Mode: — Min: — Max: Standard Mode — 0 — 152 *(Parameter Data Mode - obsolete)* — 0 — 72 *(Process Data Mode - obsolete)* — 0 — 244 *(Alarm Mode - obsolete)* — 0 — 152 *(Process Data I&M Mode - obsolete)* — 0 — 244 Standard Mode 2 — 0 — 152 Parameter Data Mode 2 — 0 — 56 Process Data Mode 2 — 0 — 244
<total>	bytes	This value must be set within the range imposed by the current Buffer Mode as follows: Buffer Mode: — Min: — Max: Standard Mode — 1 — 152 *(Parameter Data Mode - obsolete)* — 1 — 72 *(Process Data Mode - obsolete)* — 1 — 400 *(Alarm Mode - obsolete)* — 1 — 152 *(Process Data I&M Mode - obsolete)* — 1 — 368 Standard Mode 2 — 1 — 152 Parameter Data Mode 2 — 1 — 56 Process Data Mode 2 — 1 — 368

Related Information

Information	Page(s)
Buffer Modes	11
-	-

D.9 Definition of Modules

General

These parameters generally only need to be altered if customizing module names, if using network-specific ADI mapping commands (Map_ADI_Specified_Read_Area and Map_ADI_Specified-Write_Area), or if the Configuration Data attribute ("PROFIBUS DP-V1 Object (FDh)" on page 47) has been implemented.

GSD file Entries

```
; Definition of modules
Module = "<name>" <identifier>
<module_id>
EndModule
```

Extended

Setting	Description
<name>	Name of module
<identifier>	Configuration Identifier; hexadecimal value (written as 0xNN where NN is the hexadecimal value) specifying the properties of the module (see below).
<module_id>	Decimal number, must be unique for each module.

Identifier Explanation

Extended

b7	b6	b5	b4	b3	b2	b1	b0	Contents	Usage
								Number of configured data units	Value: Meaning: 0000 1 unit 0001 2 units 0010 3 units ... . 1111 16 units
								Direction of data unit	Value: Meaning: 00 Special Format, see note. 01 Input 10 Output 11 Input and Output
								Unit type	Value: Meaning:, 0 Byte 1 Word
								Consistency	Value: Meaning: 0 Consistency over unit 1 Consistency over module

Note: Advanced users may want to specify modules using the special (extended) ID format. Exactly how this is done is beyond the scope of this document.

D.10 Parametrization-related Keywords

General

These parameters generally only need to be altered in advanced implementations which requires the use of User Parameterization Data. The details about such implementations are beyond the scope of this document and requires in-depth knowledge in the PROFIBUS networking system.

GSD file Entries

```
; Parametrization related keywords
Max_User_Prm_Data_Len         = <up_len>
Ext_User_Prm_Data_Const(0)    = <up_data>
```

Advanced

Setting	Unit	Description
<up_len>	bytes	Size of User Parametrization Data, must be set within the range imposed by the current Buffer Mode as follows: Buffer Mode: — Min: — Max. Standard Mode (default) 3 153 *(Parameter Data Mode - obsolete)* 3 237 *(Process Data Mode - obsolete)* 3 9 *(Alarm Mode - obsolete)* 3 137 *(Process Data I&M Mode - obsolete)* 3 25 Standard Mode 2 3 145 Parameter Data Mode 2 3 193 Process Data Mode 2 3 9
<up_data>	-	Actual User Parametrization Data as hexadecimal values. The first three bytes are fixed and should be set to C0h, 00h, 00h.

Note: Buffer Modes marked as 'obsolete' in the table above are considered obsolete and should not be used for new designs.

Related Information

Information	Page(s)
Buffer Modes	11
Parametrization Data Handling	16
PROFIBUS DP-V1 Object (FDh) (Attribute #2)	47

D.11 Diagnostic-related Keywords

General

-

GSD file Entries

```
; Diagnostic related keywords
Max_Diag_Data_Len = <diag_len>
```

Basic

Setting	Unit	Description
<diag_len>	bytes	This value must be set to match the current Buffer Mode as follows: Buffer Mode: — Value: Standard Mode — 80 *(Parameter Data Mode - obsolete)* — 88 *(Process Data Mode - obsolete)* — 16 *(Alarm Mode - obsolete)* — 88 *(Process Data I&M Mode - obsolete)* — 24 Standard Mode 2 — 88 Parameter Data Mode 2 — 88 Process Data Mode 2 — 24

Note: Buffer Modes marked as 'obsolete' in the table above are considered obsolete and should not be used for new designs.

Related Information

Information	Page(s)
Buffer Modes	11
Diagnostics	15
Diagnostic Object (02h)	27
Additional Diagnostic Object (05h)	35
PROFIBUS DP-V1 Object (FDh) (Attribute #5)	47

D.12 Identification & Maintenance-related Keywords

General

-

GSD file Entries

```
; Identification & Maintenance related keywords
Ident_Maintenance_supp= <supp>
```

Basic

Setting	Description
<supp>	0: I&M not supported (i.e. module is running in Process Data Mode) 1: I&M supported

Related Information

Information	Page(s)
Device Identity	12
Network Configuration Object (04h)	31
PROFIBUS DP-V1 Object (FDh)	47

D.13 Status Diagnostic Messages

General

These settings may need alteration when the 'NW specific extension' of the Diagnostic Object is used. It is generally recommended to remove Diagnostic Codes which are not used by the implementation (e.g. remove 'Value (48) = "Voltage"' if this code is not applicable for the end product).

GSD file Entries

```
;Status diagnostic messages
Unit_Diag_Area=16-17
Value(0)          = "Status not changed"
Value(1)          = "Status appears"
Value(2)          = "Status disappears"
Unit_Diag_Area_End

Unit_Diag_Area    = <start>-<end>
Value(<val>)      = "<text>"
Value(<val>)      = "<text>"
Value(<val>)      = "<text>"
 ...
Value(<val>)      = "<text>"
Unit_Diag_Area_End

Unit_Diag_Area    = <start>-<end>
Value(<val>)      = "<text>"
Value(<val>)      = "<text>"
Value(<val>)      = "<text>"
 ...
Value(<val>)      = "<text>"
Unit_Diag_Area_End

 ...

Unit_Diag_Area    = <start>-<end>
Value(<val>)      = "<text>"
Value(<val>)      = "<text>"
Value(<val>)      = "<text>"
 ...
Value(<val>)      = "<text>"
Unit_Diag_Area_End
```

Advanced

Setting	Description
<start>	These settings specify the bit range to associated with <val> and <text>.
<end>	
<val>	Value
<text>	String associated with the value of <val>

Related Information

Information	Page(s)
Diagnostics	15
Diagnostic Object (02h)	27

D.14 DP-V1 related Keywords

General

-

GSD file Entries

```
; DPV1 related keywords
DPV1_Slave        = 1
Check_Cfg_Mode    = 1

C1_Read_Write_Supp= 1
C1_Max_Data_Len   = <C1_L>
C1_Response_Timeout= <C1_T>

C2_Read_Write_Supp= 1
C2_Max_Data_Len   = <C2_L>
C2_Response_Timeout= <C1_T>
C2_Max_Count_Channels= 1

Max_Initiate_PDU_Length= 52
```

Basic

Setting	Unit	Description
<C1_L>	bytes	This value must be set to match the current Buffer Mode as follows: Buffer Mode: — Value: Standard Mode — 68 *(Parameter Data Mode - obsolete)* — 240 *(Process Data Mode - obsolete)* — 16 *(Alarm Mode - obsolete)* — 68 *(Process Data I&M Mode - obsolete)* — 68 Standard Mode 2 — 68 Parameter Data Mode 2 — 240 Process Data Mode 2 — 68
<C1_T>	10ms	The host application must be able to provide a response to an ADI request within this time period, or the master may terminate the connection (default = 100)
<C2_L>	bytes	This value must be set to match the current Buffer Mode as follows: Buffer Mode: — Value: Standard Mode — 68 *(Parameter Data Mode - obsolete)* — 240 *(Process Data Mode - obsolete)* — 48 *(Alarm Mode - obsolete)* — 68 *(Process Data I&M Mode - obsolete)* — 68 Standard Mode 2 — 68 Parameter Data Mode 2 — 240 Process Data Mode 2 — 68

Note: Buffer Modes marked as 'obsolete' in the table above are considered obsolete and should not be used for new designs.

Related Information

Information	Page(s)
Buffer Modes	11
-	-

D.15 Alarm-related Keywords

General

These keywords are only applicable if the alarm functionality has been enabled in the PROFIBUS DP-V1 Object (FDh) (instance attribute #7).

GSD file Entries

```
; Alarm related keywords
Diagnostic_Alarm_supp         = <AT_S>
Process_Alarm_supp            = <AT_S>
Pull_Plug_Alarm_supp          = <AT_S>
Status_Alarm_supp             = <AT_S>
Update_Alarm_supp             = <AT_S>
Manufacturer_Specific_Alarm_supp= <AT_S>

Extra_Alarm_SAP_supp          = 0   ; Do not change
Alarm_Sequence_Mode_Count     = 32  ; Max. allowed value
Alarm_Type_Mode_supp          = 1

Diagnostic_Alarm_required     = <AT_R>
Process_Alarm_required        = <AT_R>
Pull_Plug_Alarm_required      = <AT_R>
Status_Alarm_required         = <AT_R>
Update_Alarm_required         = <AT_R>
Manufacturer_Specific_Alarm_required= <AT_R>
```

Extended

Setting	Description
<AT_S>	These settings must match the implementation as follows: Implementation: — Value Alarm type not supported — 0 Alarm type supported — 1 See also... - "Alarm Structure" on page 51 ('Supported Alarm Types')
<AT_R>	These settings must match the implementation as follows: Implementation: — Value Master is not required to enable alarm type — 0 Master is required to enable alarm type — 1 See also... - "Alarm Structure" on page 51 ('Required Alarm Types')

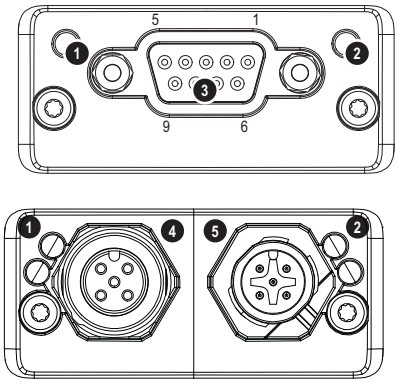
Related Information

Information	Page(s)
Buffer Modes	11
Alarm Handling	19
PROFIBUS DP-V1 Object (FDh)	47

E. Technical Specification

E.1 Front View

#	Item
1	Operation Mode
2	Status
3	PROFIBUS Connector
4	M12 Female Connector
5	M12 Male Connector



Operation Mode

State	Indication	Comments
Off	Not online / No power	-
Green	Data exchange	-
Flashing Green	Clear	-
Flashing Red (1 flash)	Parametrization error	See “Parametrization Data Handling” on page 16
Flashing Red (2 flashes)	PROFIBUS Configuration error	See “Configuration Data Handling” on page 17

Status

State	Indication	Comments
Off	Not initialized	Anybus state = ‘SETUP’’ or ‘NW_INIT’
Green	Initialized	Anybus module has left the ‘NW_INIT’ state
Flashing Green	Initialized, diagnostic event(s) present	Extended diagnostic bit is set
Red	Exception error	Anybus state = ‘EXCEPTION’

PROFIBUS Connector (DB9F)

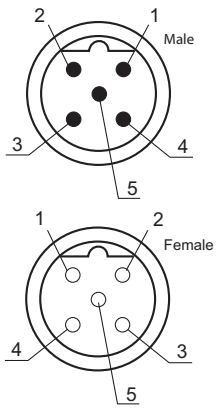
Pin	Signal	Description
1	-	-
2	-	-
3	B Line	Positive RxD/TxD, RS485 level
4	RTS	Request to send
5	GND Bus	ground (isolated)
6	+5 V Bus Output[a]	+5 V termination power (isolated, short-circuit protected)
7	-	-
8	A Line	Negative RxD/TxD, RS485 level
9	-	-
Housing	Cable Shield	Internally connected to the Anybus protective earth via cable shield filters according to the PROFIBUS standard.

a. The current drawn from this pin will affect the total power consumption. The output can supply 10 mA for one output. See also "Power Consumption" on page 79.

M12 Connectors, Code B

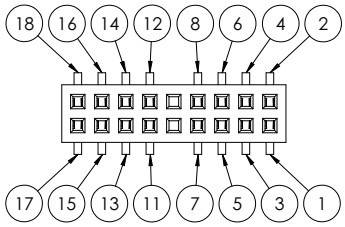
The female M12 connector is used when modules are used in a daisy-chain topology.

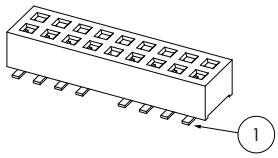
Pin	Name (Male / Female)	Male	Female
1	NC / P5V	Not connected	Power Supply, 5V DC
2	RxD/TxD-N (A) / RxD/TxD-N (A)	Receive/Transmit negative	Receive/Transmit negative
3	NC / DGND	Not Connected	Power ground, 0 V
4	RxD/TxD-P (B) / RxD/TxD-P (B)	Receive/Transmit positive	Receive/Transmit positive
5 (Thread)	Shield	Shield	Shield

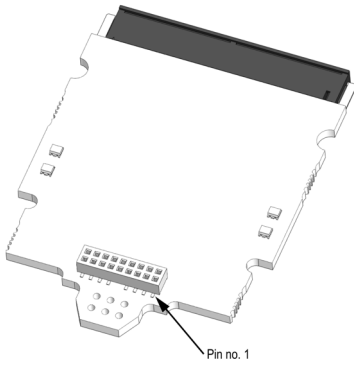


E.2 Network Connector, Brick Version

The Anybus CompactCom PROFIBUS DP-V1 can also be acquired in a brick version, without a fieldbus connector, but instead a pin connector to the carrier board (the host device). The concept and assembly are described in the Anybus CompactCom Brick and without Housing Design Guide (Doc. Id. HMSI-168-30).







Pin no.	Signal	Comments
1	Shield	Input when daisy-chaining and/or using M12 connectors for access to the network. Otherwise not used.
2	A	
3	B	
4	Shield	
5	NC	
6	NC	
7	NC	
8	Shield	
11	Shield	Output when daisy-chaining and/or using M12 connectors for access to network. Used for network access when designing for a DSUB connector. Pins 12 and 13 are used for termination.
12	+5 V	
13	DGND	
14	Shield	
15	NC	
16	B	
17	A	
18	Shield	

Keep the A and B signals as short as possible. Avoid unnecessary stubs. Use a single, non-broken reference plane for A and B signals, preferably DGND.

All shield pins are internally connected. Ensure that shield is properly connected, see "Protective Earth (PE) Requirements" on page 78.

E.3 Protective Earth (PE) Requirements

In order to ensure proper EMC behavior, the module must be properly connected to protective earth via the PE pad / PE mechanism described in the general Anybus CompactCom 30 Hardware Design Guide.

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

E.4 Power Supply

Supply Voltage

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

Power Consumption

The Anybus CompactCom PROFIBUS DP-V1 is designed to fulfill the requirements of a Class A module. For more information about the power consumption classification used on the Anybus CompactCom platform, consult the general Anybus CompactCom 30 Hardware Design Guide.

The current hardware design consumes up to 230 mA[1,2].

Note: 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 30 Hardware Design Guide, and not on the exact power requirements of a single product.

E.5 Environmental Specification

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

E.6 EMC Compliance

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

1. Note that 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 PROFIBUS DP-V1 will remain a Class A module.
2. This value is valid under the condition that no current is being drawn from bus connector pin 6 (+5 V termination power; see "PROFIBUS Connector (DB9F)" on page 77).

F. Timing & Performance

F.1 General Information

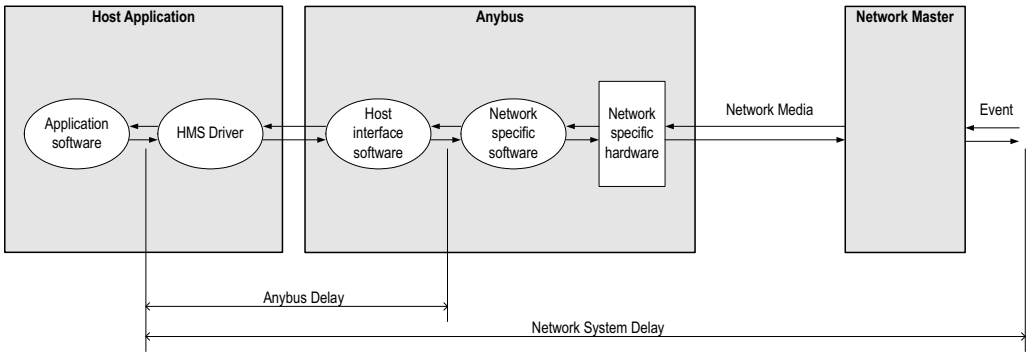
This chapter specifies timing and performance parameters that are verified and documented for the Anybus CompactCom PROFIBUS DP-V1.

The following timing aspects are measured:

Category	Parameters	Page
Startup Delay	T1, T2	Please consult the Anybus CompactCom Software 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	82
Network System Write Process Data Delay (Network System Delay)	T15, T16, T17	82

F.2 Process Data

F.2.1 Overview



F.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 Design Guide, Appendix B, for more information.

F.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.

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

The Network System Read Process Data Delay (labelled 'Network System Delay in the figure), is defined as the time measured from the point where an event is generated at the network master to when the corresponding data is available to the host application (just after the corresponding data has been read from the driver).

Parameter	Description	Typ.	Max.	Unit.
T9	Network System Read Process Data delay, 8 ADIs (single UINT8)	8	9	ms
T10	Network System Read Process Data delay, 16 ADIs (single UINT8)	10	12	ms
T11	Network System Read Process Data delay, 32 ADIs (single UINT8)	11	12	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
Profibus baud rate	12 Mbit/s
Bus load, no. of nodes etc.	Normal

F.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.	Max.	Unit.
T15	Network System Write Process Data delay, 8 ADIs (single UINT8)	4.5	5.5	ms
T16	Network System Write Process Data delay, 16 ADIs (single UINT8)	5.5	6	ms
T17	Network System Write Process Data delay, 32 ADIs (single UINT8)	8	9	ms

Conditions: as in "Network System Read Process Data Delay (Network System Delay)" on page 82.