Fieldbus Appendix

Anybus®-S ControlNet

Doc.Id. SCM-1200-009
Rev. 2.02

HMS Industrial Networks AB

	☎	✉
Germany	+49 - 721 - 96472 - 0	ge-sales@hms-networks.com
Japan	+81 - 45 - 478 -5340	jp-sales@hms-networks.com
Sweden	+46 - 35 - 17 29 20	sales@hms-networks.com
U.S.A.	+1 - 312 - 829 - 0601	us-sales@hms-networks.com
France	+33 - 3 89 32 76 76	fr-sales@hms-networks.com
Italy	+39 - 347 - 00894 - 70	it-sales@hms-networks.com
China	+86 - 10 - 8532 - 3183	cn-sales@hms-networks.com

Table of Contents

Chapter 4 Mailbox Interface

Chapter 5 Fieldbus Specific Registers

Appendix A Electrical Characteristics

Appendix B Connectors

Appendix C Mechanical Specification

Appendix D Environmental Specification

About This Manual

How To Use This Manual

This document is intended to be used in conjunction with the Anybus-S Parallel Design Guide. For general information about the Anybus-S platform, consult the general Anybus-S Parallel Design Guide.

The reader of this document is expected to be familiar with hardware and software design as well as to have basic knowledge in the ControlNet fieldbus, and communication systems in general.

Note: This document describes the functions available in the latest firmware release. Some features may be missing or behave slightly different in older releases. Please contact HMS for further information.

Important User Information

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 application meets all performance and safety requirements including any applicable laws, regulations, codes, and standards.

Related Documents

Document name	Author
Anybus-S Parallel Design Guide	HMS
ControlNet Specification	ControlNet International

Document History

Summary of Recent Changes (2.01 to 2.02)

Change	Page(s)
Added new mailboxes: SET_PRODUCT_INFO_ALL and SET_SERIAL_NUMBER	4-12, 4-11
Added new mailboxes: ENABLE_LARGE_IO, ENABLE_LARGE_UCMM_REQUEST, and LARGE_UCMM_REQUEST	4-19, 4-21, 4-22

Revision List

Revision	Date	Author	Chapter	Description
<2.00	-	-	All	See previous documents
2.00	2004-04-28	PeP	All	Second major revision
2.01	2004-06-17	ToT	Mailbox Messages	Removed obsolete message ‘ Set Product Code’
2.02	2009-08-19	KeL	4	Added new mailboxes

Conventions used in this manual

The following conventions are used throughout this manual:

- Numbered lists provide sequential steps
- Bulleted lists provide information, not procedural steps
- The term ‘module’ is used when referring to the Anybus-S ControlNet module
- The term ‘application’ is used when referring to the hardware that is connected to the Anybus Application Connector
- Hexadecimal values are written in the format NNNNh, where NNNN is the hexadecimal value.

Support

HMS Sweden (Head Office)

E-mail:	support@hms-networks.com
Phone:	+46 (0) 35 - 17 29 20
Fax:	+46 (0) 35 - 17 29 09
Online:	www.anybus.com

HMS North America

E-mail:	us-support@hms-networks.com
Phone:	+1-312-829-0601
Toll Free:	+1-888-8-Anybus
Fax:	+1-312-738-5873
Online:	www.anybus.com

HMS Germany

E-mail:	ge-support@hms-networks.com
Phone:	+49-721-96472-0
Fax:	+49-721-964-7210
Online:	www.anybus.com

HMS Japan

E-mail:	jp-support@hms-networks.com
Phone:	+81-45-478-5340
Fax:	+81-45-476-0315
Online:	www.anybus.com

HMS China

E-mail:	cn-support@hms-networks.com
Phone:	+86 10 8532 3023
Online:	www.anybus.com

HMS Italy

E-mail:	it-support@hms-networks.com
Phone:	+39 039 59662 27
Fax:	+39 039 59662 31
Online:	www.anybus.com

HMS France

E-mail:	mta@hms-networks.com
Phone:	+33 (0) 3 89 32 76 41
Fax:	+33 (0) 3 89 32 76 31
Online:	www.anybus.com

About the Anybus-S ControlNet

The Anybus-S module for ControlNet is classified as a ControlNet adapter, i.e., it can not originate connections on its own, but a scanner node can open a connection towards it. The module is implemented according to the ControlNet International specification for a Communication adapter (profile no. 12).

The Anybus-S ControlNet module can be accessed by other nodes via UCMM (Unscheduled) messages. Unscheduled messages are usually used for information such as configuration data.

The module is also equipped with a NAP (Network access port) for temporary connection of configuration tools, e.g. a PC card.

Features

- **ControlNet Adapter**
- **Network Access Port (NAP)**
- **Media redundancy support**
- **Conforms to Communications Adapter, profile 12**
- **Mac ID can be set via onboard switches or application interface**
- **Up to 512 bytes of I/O data in each direction[1].**
- **UCMM Client / Server support**
- **Galvanically isolated bus electronics**
- **UL & cUL conformance**

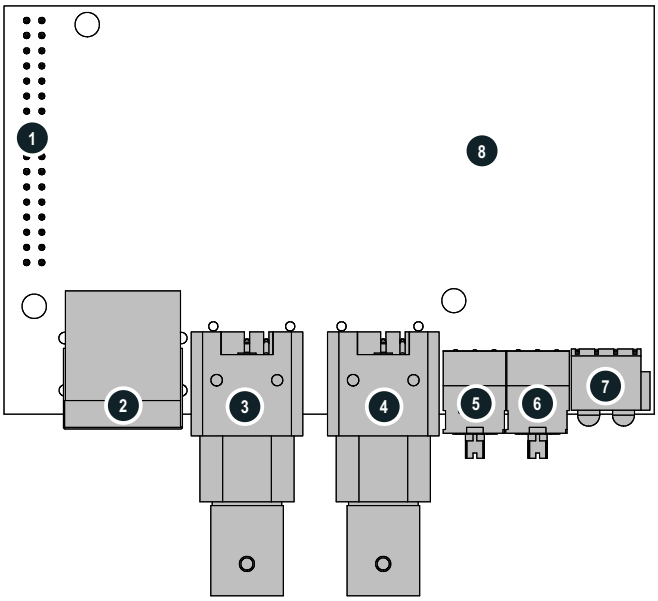
EDS File

Each device in a ControlNet network is associated with an EDS file, containing information needed for network configuration. This file is used by the network configuration tool during network configuration.

The latest version of EDS file can either be downloaded from HMS website (www.hms-networks.com) or obtained by contacting HMS.

1. If the command ENABLE_LARGE_IO is sent during initialization. Otherwise 450 bytes of I/O data in each direction will be available. Also see 4-19 “Enable Large I/O (ENABLE_LARGE_IO)”.

Overview



#	Description
1	Application Connector
2	Network Access Port (NAP)
3	ControlNet Channel A
4	ControlNet Channel B
5	MacID switch (x10)
6	MacID switch (x1)
7	ControlNet status indicators
8	Anybus-S Watchdog

Connectors

Application Connector

Standard Anybus-S DPM 2kbyte DPRAM interface. For further information, please consult the general Anybus-S Parallel Design Guide.

Network Access Port (NAP)

The NAP (Network Access Port) provides temporary access to the ControlNet network for diagnostic and configuration.

ControlNet Channels A & B

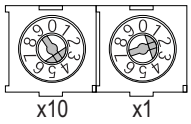
The module is equipped with two BNC contacts for connection to ControlNet. If redundant operation is desired, both connectors are used, otherwise connector A or B is used.

MacID Switches (x10 and x1)

On a ControlNet network, each node must be assigned it's own unique node address. The module features on board switches for MacID configuration, providing an address range of 1 - 99.

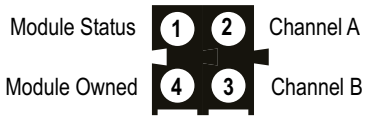
Example:

In this example, the MacID is set to 42 (4 x 10 + 2 x 1)



Note: The MacID can also be set via software. In this case, the switches should be set to 00.

Indicators



ControlNet Status Indicators

These LED:s indicate run time status and errors to the user.

#	Indication	State	Description
1	Module Status	Green	Connection in Run State
		Green, Flashing	Connecting Connection Idle
		Red	Major fault
		Red, Flashing	Minor fault
2 *and* 3	Channel. A *and* Channel. B	Off	Module not initialized
		Red	Major fault
		Alternating red/green	Self test
		Red, flashing	Node configuration error; duplicate MAC ID etc.
2 *or* 3	Channel. A *or* Channel. B	Off	Channel disabled
		Green	Normal operation of channel
		Green, flashing	Temporary error (node will self correct) or not configured
		Red, flashing	No other nodes, or media fault
		Red & green, flashing	Network configuration error
4	Module Owned	Off	No connection has been opened
		Green	A connection has been opened towards the module

Anybus-S Watchdog

(Consult the general Anybus-S Design Guides for further information)

Software Overview

Initialization

The Input and Output I/O data sizes specified in ANYBUS_INIT specifies the maximum input/output size of a connection towards the module. The module supports up to 2kbyte of fieldbus data in each direction, out of which up to 450 bytes can be I/O data. If the command ENABLE_LARGE_IO is sent during initalization, up to 512 bytes can be I/O data, see 4-19 "Enable Large I/O (ENABLE_LARGE_IO)".

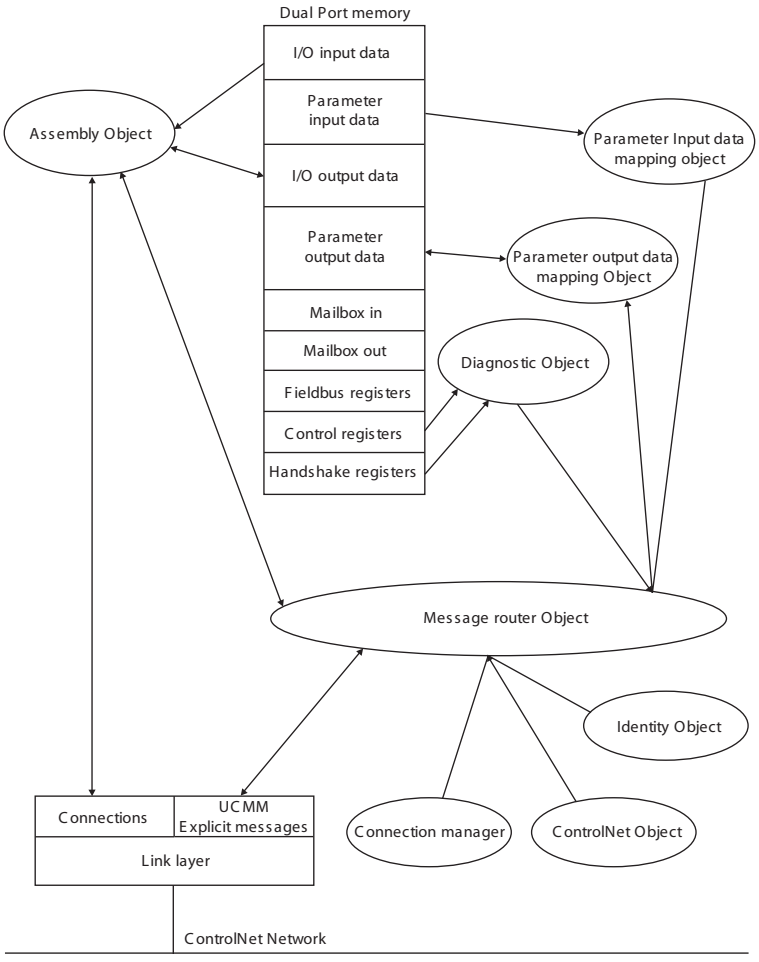
The actual size of a connection depends on the size that is specified in the bus configuration tool, and may thus be less than specified in ANYBUS_INIT. However, a connection towards the module can never be larger than the size specified during initialization.

Note that unlike most bus configuration tools, the data sizes in ANYBUS_INIT are specified in bytes.

Object Model

The interface from the fieldbus towards the module is based on the standard ControlNet objects and three vendor specific objects.

The vendor specific objects can be used to access different memory areas and for diagnostics during development. However, most applications requires only the standard ControlNet objects.



Data Exchange

I/O Data

The input/output data is read/written through I/O connection or Explicit messages from the assembly object.

Note: Unlike other registers in the dual port memory, the data in the Input and Output Data areas are stored in little endian (Intel) format.

Parameter Data

The input/output Parameter data can be accessed via Explicit messages through application objects mapped by mailbox telegrams during the initialization.

Note: Unlike other registers in the dual port memory, the data in the Input and Output Data areas are stored in little endian (Intel) format.

Implementation Note

Rockwell Automation PLCs uses the first four bytes produced/consumed by a device defined for status information.

By default, these four I/O bytes are parsed off and ignored. However, this behaviour can be changed using the COPY_IO_STATUS mailbox command (see 4-18 “Copy I/O Status (COPY_IO_STATUS)”).

ControlNet Object Implementation

General

ControlNet is based on the Control and Information protocol (CIP) which is also the application layer for DeviceNet and EtherNet/IP.

CIP makes use of abstract object modelling to describe the communications of a product. Objects are well defined subsets of the functionality of a device. This include functions, called 'Services' and data variables called 'Attributes'. If more than one copy of an object is needed, each copy is called an 'Instance'.

Implemented Objects

ControlNet requires some mandatory objects; these are implemented as well as some vendor specific objects. The mandatory objects are in the specification from ODVA.

ControlNet Objects

Vendor Specific Objects

Mandatory ControlNet Objects

Identity Object, Class 01h

Services

Class services: Get Attribute All

Instance services: Get Attribute All
Reset (See next page)

Class Attributes

#	Access	Name	Type	Value	Description
1	Get	Revision	UINT	0001h	Revision 1
2	Get	Max Instance	UINT	0001h	Maximum instance number is 1

Instance Attributes

#	Access	Name	Type	Value	Description
1	Get	Vendor ID	UINT	Default: 005Ah	HMS Industrial Networks AB[a]
2	Get	Device Type	UINT	Default: 000Ch	Communication Adapter[a]
3	Get	Product Code	UINT	Default: 000Eh	ABS-CNT[a]
4	Get	Revision	Struct of:		(Version XX.YY)
			USINT	XX	Major fieldbus version[a]
			USINT	YY	Minor fieldbus version[a]
5	Get	Status	WORD	-	Device status, see table below
6	Get	Serial Number	UDINT	Module serial number	Serial number of the module
7	Get	Product Name	SHORT_STRING	“Anybus-S CNT”	Name of product[a]

a. Can be customized using mailbox commands, see 4-3 “Mailbox Messages”.

Status Attribute

bit(s)	Name	Description
0	Module Owned	Set when at least one connection is configured
1	(reserved)	-
2	Configured	Always set to 1
3	(reserved)	-
4 - 7	Extended Device Status	(See table on the right)
8	Minor recoverable fault	Set = minor fault
9	Minor unrecoverable fault	Set = minor fault
10	Major recoverable fault	Set = major fault
11	Major unrecoverable fault	Set = major fault, fatal
12 - 15	(reserved)	-

Extended Device Status

Value	Meaning
0000b	Power-up/Self-test
0010b	Faulted I/O Connection
0011b	Awaiting connection
0100b	Non volatile configuration bad
0110b	Connection in Run mode
0111b	Connection in Idle mode

Reset Service

The Identity object provides a reset service. There are two different types of reset requests:

- **Type 0: 'Power Cycling Reset'**
  This service emulates a power cycling of the module.
- **Type 1: 'Out of box reset'**
  This service sets a "out of box" configuration and performs a reset.

The default behaviour when the module receives a reset command is to reset the module.

If the application should be notified about fieldbus reset requests, the feature must be enabled in the ANYBUS_INIT mailbox command during module initialization:

- In ANYBUS_INIT, set the RDR bit of the Operation Mode register
- In ANYBUS_INIT, set the RST bit of the Event Notification Config register

If enabled, the module will generate an Event Notification when a fieldbus reset request has been encountered. The type of reset request can then be retrieved using the mailbox command GET_ID_RESET_PARAM, see 4-10 "Get Reset Parameter (GET_ID_RESET_PARAM)".

It is then up to the application to reset itself and the module.

Message Router, Class 02h

Services

Class services: -
Instance services: -

Assembly Object, Class 04h

Services

Class services: Get Attribute Single

Instance services: Get Attribute Single
Set Attribute Single

Description

The Assembly Object binds all mapped I/O data. This data is used for I/O connections.

Default I/O Instances used are 64h and 96h.

Class Attributes

#	Access	Name	Type	Value	Description
1	Get	Revision	UINT	0002h	Revision 2

Instance Attributes - Instance/Connection Point 64h (Input Data Area)

Note: If the I/O input data size is set to 0 this instance will NOT be initialised.

#	Access	Name	Type	Value	Description
3	Get	Data	Array of USINT	-	Data produced by Anybus

Instance Attributes - Instance/Connection Point 96h (Output Data Area)

Note: If the I/O output data size is set to 0 this instance will NOT be initialised.

#	Access	Name	Type	Value	Description
3	Set	Data	Array of USINT	-	Data consumed by Anybus[a]

a. Rockwell Automation PLCs have the first four bytes consumed by a device defined as status information. Since all known PLCs have this implementation, the module strips off the first four bytes in the consumed data by default. However, this behaviour can be changed using the Copy I/O Status mailbox command, see 4-18 “Copy I/O Status (COPY_IO_STATUS)”.

Connection Manager Object, Class 06h

Services

Class services: Forward Open
Forward Close

Instance services: -

ControlNet Object, Class F0h

Services

Class services: Get Attribute All

Instance services: Get Attribute All
Get And Clear

Class Attributes

#	Access	Name	Type	Value	Description
1	Get	Revision	UINT	0001h	Revision 1
2	Get	Max Instance	UDINT	00000001h	Maximum instance number is 1.

Instance Attributes, Instance 01h

#	Access	Name	Type	Description
81h	Get	Current_link_config	Struct of:	
		Link_config	Struct of:	
		NUT_length	UINT	NUT length in 10 us ticks
		smax	USINT	0 to 99
		umax	USINT	1 to 99
		slotTime	USINT	In 1 us ticks
		blanking	USINT	In 1.6 us ticks
		gb_start	USINT	In 10 us ticks
		gb_center	USINT	In 10 us ticks
		reserved	UINT	Reserved
		modulus	USINT	127 (required)
		gb_prestart	USINT	In 10 us ticks
		TUI	Struct of:	
		unique_ID	UDINT	Keeper CRC
		status_flag	UINT	TUI flag
		reserved	USINT[16]	Reserved

#	Access	Name	Type	Description
82h	Get, Get and Clear	diagnostic_counters	Struct of:	
		buffer_errors	UINT	Buffer event counter
		error_log	BYTE[8]	Bad MAC frame log
		event_counters	Struct of:	
		good_frames_transmitted	BYTE[3]	Good MAC frames transmitted (LSB first)
		good_frames_received	BYTE[3]	Good MAC frames received (LSB first)
		selected_channel_frame_errors	USINT	Framing errors detected on active receive channel
		channel_A_frame_errors	USINT	Framing errors detected on channel A
		channel_B_frame_errors	USINT	Framing errors detected on channel B
		aborted_frames_transmitted	USINT	MAC frames aborted during transmission (transmit underflows)
		highwaters	USINT	LLC transmit underflow and LLC receive overflow
		NUT_overloads	USINT	No unscheduled time in NUT (All time used for scheduled transmissions)
		slot_overloads	USINT	More scheduled data queued for one NUT than allowed by sched_max_frame parameter
		blockages	USINT	Single Lpacket size exceeds sched_max_frame parameter
		non_concurrence	USINT	Two or more nodes could not agree whose turn it is to transmit
		aborted_frames_received	USINT	Incomplete MAC frames received
		lonely_counter	USINT	Number of times nothing heard on network for 8 or more NUTs
		duplicate_node	USINT	MAC frame received from node with local node's MAC ID
		noise_hits	USINT	Noise detected that locked the modem rx PLL
		collisions	USINT	Rx data heard just as we are going to transmit
		Mod_MAC_ID	USINT	MAC D of the current moderator node
		non_lowman_mods	USINT	Moderator frames heard from non-lowman nodes
		rouge_count	USINT	Rouge events detected
		unheard_moderator	USINT	MAC frames being heard but no moderators being heard
		vendor_specific	USINT	-
		reserved	BYTE[4]	Reserved
		vendor_specific	USINT	Not used (00h)
		vendor_specific	USINT	Not used (00h)
		reserved	BYTE	Reserved

#	Access	Name	Type	Description
83h	Get	station_status	Struct of:	
		smac_ver	USINT	MAC implementation (02h)
		vendor_specific	BYTE[4]	Vendor specific (00 18 00 00h)
		channel_state	BYTE	Channel state LEDs, redundancy warning, and active channel bits
84h	Get	MAC_ID	Struct of:	
		MAC_ID_current	USINT	Current MAC ID
		MAC_ID_switches	USINT	MAC ID switch settings
		MAC_ID_changed	BOOL	MAC ID switches changed since reset
		reserved	USINT	Reserved
86h	Get	error_log	Struct of:	
		buffer_errors	UINT	Buffer event counter
		error_log	BYTE[8]	Bad MAC frame log

Vendor Specific Objects

Diagnostic Object, Class AAh

Services

Class services: Get Attribute All

Instance services: Get Attribute All
Get Attribute Single

Description

This vendor specific object provides diagnostic information from the module. The information in this object corresponds to various parameters in the Control Register Area and Fieldbus Specific Area. Consult the Anybus-S Parallel Design Guide for further information.

Class Attributes

#	Access	Name	Type	Value	Description
1	Get	Revision	UINT	0001h	Revision 1

Instance Attributes, Instance 01h

#	Access	Name	Type	Description
01h	Get	Module serial number	UDINT	Serial number
02h	Get	Vendor ID	UINT	Manufacturer Vendor ID
03h	Get	Fieldbus Type	UINT	Fieldbus Type (In this case 0065h)
04h	Get	Module Software version	UINT	Module software version
05h	Get	Interrupt Count	UINT	Counter incremented each handshake interrupt
06h	Get	Watchdog counter in	UINT	(reserved, not implemented)
07h	Get	Watchdog counter out	UINT	Counter incremented every 1ms
09h	Get	LED	Struct of:	LED Indication:
			USINT	Top left
			USINT	Top right
			USINT	Bottom left
			USINT	Bottom right
			USINT	(reserved)
			USINT	(reserved)
0Ah	Get	Module Type	UINT	Module Type (In this case 0101h)
0Bh	Get	Module Status	WORD	Bit information (freeze, clear etc.)
0Ch	Get	New data field	LWORD	Array of new data flags for 8 bytes area
0Dh	Get	Event Cause	WORD	Event Cause register
0Eh	Get	Event Notification	WORD	Event Notification register
0Fh	Get	IN cyclic I/O length	UINT	Size of I/O Input area (in bytes)
10h	Get	IN DPRAM length	UINT	Number of valid IN bytes in DPRAM
11h	Get	IN total length	UINT	Total number of IN bytes supported
12h	Get	OUT cyclic I/O length	UINT	Size of I/O Output area (in bytes)
13h	Get	OUT DPRAM length	UINT	Number of valid OUT bytes in DPRAM
14h	Get	OUT total length	UINT	Total number of OUT bytes supported

Parameter Data Input Mapping Object, Class B0h

Services

Class services: Get Attribute All

Instance services: Get Attribute Single

Description

This vendor specific object is setup dynamically depending on how the module is initialized. The mapped attribute id:s corresponds to input parameter data. The mapped data is the data that is being sent *from* the module via connected- and unconnected messages.

Class Attributes

#	Access	Name	Type	Value	Description
1	Get	Revision	UINT	0001h	Revision 1

Instance Attributes, Instance 01h

#	Access	Name	Type	Description
01h	Get	Data	Array of USINT	Mapped data
02h	Get	Data	Array of USINT	Mapped data
...	...	...	...	...
32h	Get	Data	Array of USINT	Mapped data

The attributes above are created by the application when sending the mailbox command PARAMETER_INPUT_MAP (See 4-6 “Parameter Data Input Area Mapping (PARAMETER_INPUT_MAP)”)

Parameter Data Output Mapping Object, Class B1h

Services

Class services: Get Attribute All

Instance services: Get Attribute Single
Set Attribute Single

Description

This vendor specific object is setup dynamically depending on how the module is initialized. The mapped attribute id:s corresponds to output parameter data in the DPRAM. The mapped data is the data that is being *received* by the module via connected and unconnected messages.

Class Attributes

#	Access	Name	Type	Value	Description
1	Get	Revision	UINT	0001h	Revision 1

Instance Attributes, Instance 01h

#	Access	Name	Type	Description
01h	Get/Set	Data	Array of USINT	Mapped data
02h	Get/Set	Data	Array of USINT	Mapped data
...	...	...	...	...
32h	Get/Set	Data	Array of USINT	Mapped data

The attributes above are created by the application when sending the mailbox command PARAMETER_OUTPUT_MAP (See 4-8 “Parameter Data Output Area Mapping (PARAMETER_OUTPUT_MAP)”)

Mailbox Interface

This chapter describes the fieldbus specific mailbox commands in the module. Consult the Anybus-S Parallel Design Guide for more information regarding mailbox functionality.

Summary

Fault Codes

When a mailbox command for some reason can't be processed by the module, the Message Information register in the header of the response will indicate that an error has occurred. If the Error Code in the

Message Information register is Invalid Other (Fh), additional fault information can be found in the Fault Code register (Extended Word 8 in the response header).

The fault codes are:

Fault Code	Description
0006h	Command failed
0011h	Too many registered objects
0012h	Object already registered
0013h	Invalid class
0016h	Send UCMM command failed
001Bh	Mapping Failed

Mailbox Messages

Set Mac ID (SET_MAC_ID)

Description

This command enables the application to set the Mac ID from software instead of using the two on board rotary switches. Please note that when using this feature the rotary switches should be set to zero to indicate that the Mac ID is set via software.

Parameter	Description
Command initiator	Application
Command Name	SET_MAC_ID
Message type	02h
Command number	0001h
Extended Header data	-
Command data	Mac ID (Node address)
Response data	The response indicates if the Mac ID was accepted. The Mac ID is checked to be in the range 1-99. If the value is out of range the Anybus-S module automatically sets the Mac ID to 0, which causes the module to stay off-line.

Command and response layout:

	Command	Expected response	
Message ID	(ID)	(ID)	
Message information	4002h	0002h	
Command	0001h	0001h	*SET_MAC_ID*
Data size	0001h	0001h	
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	-	-	
Extended word 2	-	-	
Extended word 3	-	-	
Extended word 4	-	-	
Extended word 5	-	-	
Extended word 6	-	-	
Extended word 7	-	-	
Extended word 8	-	Fault Code	*(See 4-1 "Fault Codes")*
Message data byte 1	Mac ID	Mac ID	

- **Mac ID Value**
  Desired Mac ID.

Note: If the application features on board switches for Mac ID configuration, this mailbox must be sent each time the switch value has changed. In accordance with the ControlNet specification, this will cause a minor fault when the switches are changed during runtime.

Set Product Info (SET_PRODUCT_INFO)

Description

This command is used to customize the contents of the identity object, i.e. the Vendor ID, Product Code and Product Name. Note that in order to use this feature, a unique Vendor ID must be obtained from ODVA. Furthermore, the.EDS-file must be updated accordingly.

Note: This command can only be sent during module initialization.

Parameter	Description
Command initiator	Application
Command Name	SET_PRODUCT_INFO
Message type	02h
Command number	0002h
Extended Header data	-
Command data	Vendor ID, Product Code, Name length and name (ascii, max 32 characters)
Response data	The response indicates if the command was accepted.

Command and response layout:

	Command	Expected response	
Message ID	(ID)	(ID)	
Message information	4002h	0002h	
Command	0002h	0002h	*SET_PRODUCT_INFO*
Data size	(data size)	(data size)	*Message data size in bytes*
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	-	-	
Extended word 2	-	-	
Extended word 3	-	-	
Extended word 4	-	-	
Extended word 5	-	-	
Extended word 6	-	-	
Extended word 7	-	-	
Extended word 8	-	Fault Code	*(See 4-1 "Fault Codes")*
Message data byte 1	Vendor ID (msb)	Vendor ID (msb)	
Message data byte 2	Vendor ID (lsb)	Vendor ID (lsb)	
Message data byte 3	Product Code (msb)	Product Code (msb)	
Message data byte 4	Product Code (lsb)	Product Code (lsb)	
Message data byte 5	Product Name Length	Product Name Length	*Maximum 32 characters*
Message data byte 6	Product Name (1st char.)	Product Name (1st char.)	
Message data byte 7	Product Name (2nd char.)	Product Name (2nd char.)	
...	...	...	
Message data byte N	Product Name (last char.)	Product Name (last char.)	

- **Vendor ID**

  Unique vendor ID, must be obtained from ODVA.

- **Product Code**

  Unique product code.

- **Product Name Length**

  Length of the Product Name (See below) in bytes.

- **Product Name**

  ASCII string containing the name of the product, up to 32 characters long.

Parameter Data Input Area Mapping (PARAMETER_INPUT_MAP)

Description

This command makes it possible to set up blocks of data in the Input Parameter Data Area to be used with Class B0h 'Parameter Data Input Object' (See 3-9 "Parameter Data Input Mapping Object, Class B0h"). This way, a Get_Attribute_Single command from the fieldbus can be used to retrieve a specified block of data from the Input Parameter Data Area.

Attributes are mapped beginning with attribute 1, followed by attribute 2 through 50. Offset in the Input Data Area and number of bytes to map is specified for each attribute. If zero length is specified for an attribute, that attribute will not be mapped. This way, it is for example possible to map only attributes 1 and 10 by specifying zero length for attributes 2 through 9. It is only necessary to include information in the telegram up to the last mapped attribute; the remainder will not be mapped.

Note that the length or offset setting for an attribute is invalid, that attribute will not be mapped.

To access the mapped data from ControlNet, use Class attribute B0h, Instance 01h and an Attribute ID between 01h and 32h.

Note: This command can only be sent during module initialization after the ANYBUS_INIT command

Parameter	Description
Command initiator	Application
Command Name	PARAMETER_INPUT_MAP
Message type	02h
Command number	0004h
Extended Header data	Fault Code
Command data	Offset and length of the attributes to map.
Response data	The response indicates if the message was accepted.

Command and response layout:

	Command	Expected response	
Message ID	(ID)	(ID)	
Message information	4002h	0002h	
Command	0004h	0004h	*PARAMETER_INPUT_MAP*
Data size	0014h	0014h	
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	-	-	
Extended word 2	-	-	
Extended word 3	-	-	
Extended word 4	-	-	
Extended word 5	-	-	
Extended word 6	-	-	
Extended word 7	-	-	
Extended word 8	-	Fault Code	*(See 4-1 "Fault Codes")*
Message data word 1	Offset for Attribute 1	Offset for Attribute 1	
Message data word 2	Length for Attribute 1	Length for Attribute 1	
Message data word 3	Offset for Attribute 2	Offset for Attribute 2	
Message data word 4	Length for Attribute 2	Length for Attribute 2	
Message data word 5	Offset for Attribute 3	Offset for Attribute 3	
Message data word 6	Length for Attribute 3	Length for Attribute 3	
Message data word 7	Offset for Attribute 4	Offset for Attribute 4	
Message data word 8	Length for Attribute 4	Length for Attribute 4	
Message data word 9	Offset for Attribute 5	Offset for Attribute 5	
Message data word 10	Length for Attribute 5	Length for Attribute 5	

Note: Example when only setting attribute 1-5.

- **Fault Code**
  (See 4-1 "Fault Codes")
- **Offset for Attribute N**
  Offset of the mapped attribute from the start of the Input Parameter Data Area
- **Length for Attribute N**
  Length (in bytes) of the mapped attribute.

Parameter Data Output Area Mapping (PARAMETER_OUTPUT_MAP)

Description

This command makes it possible to set up blocks of data in the Output Parameter Data Area to be used with Class B1h 'Parameter Data Output Object' (See 3-10 "Parameter Data Output Mapping Object, Class B1h"). This way, a Get_Attribute_Single or Set_Attribute_Single command from the fieldbus can return/write a specified block of data from/to the Output Parameter Data Area.

Attributes are mapped beginning with attribute 1, followed by attribute 2 through 50. Offset in the Input Data Area and number of bytes to map is specified for each attribute. If zero length is specified for an attribute, that attribute will not be mapped. This way, it is for example possible to map only attributes 1 and 10 by specifying zero length for attributes 2 through 9. It is only necessary to include information in the telegram up to the last mapped attribute; the remainder will not be mapped.

Note that the length or offset setting for an attribute is invalid, that attribute will not be mapped.

To access the mapped data from ControlNet, use Class attribute B1h, Instance 01h and an Attribute ID between 01h and 32h.

Note: This command can only be sent during module initialization after the ANYBUS_INIT command.

Parameter	Description
Command initiator	Application
Command Name	PARAMETER_OUTPUT_MAP
Message type	02h
Command number	0005h
Extended Header data	Fault Code
Command data	Offset and length of the attributes to map.
Response data	The response indicates if the message was accepted.

Command and response layout:

	Command	Expected response	
Message ID	(ID)	(ID)	
Message information	4002h	0002h	
Command	0005h	0005h	*PARAMETER_OUTPUT_MAP*
Data size	0014h	0014h	
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	-	-	
Extended word 2	-	-	
Extended word 3	-	-	
Extended word 4	-	-	
Extended word 5	-	-	
Extended word 6	-	-	
Extended word 7	-	-	
Extended word 8	-	Fault Code	*(See 4-1 "Fault Codes")*
Message data word 1	Offset for Attribute 1	Offset for Attribute 1	
Message data word 2	Length for Attribute 1	Length for Attribute 1	
Message data word 3	Offset for Attribute 2	Offset for Attribute 2	
Message data word 4	Length for Attribute 2	Length for Attribute 2	
Message data word 5	Offset for Attribute 3	Offset for Attribute 3	
Message data word 6	Length for Attribute 3	Length for Attribute 3	
Message data word 7	Offset for Attribute 4	Offset for Attribute 4	
Message data word 8	Length for Attribute 4	Length for Attribute 4	
Message data word 9	Offset for Attribute 5	Offset for Attribute 5	
Message data word 10	Length for Attribute 5	Length for Attribute 5	

Note: Example when only setting attribute 1-5.

- **Fault Code**
  (See 4-1 "Fault Codes")
- **Offset for Attribute N**
  Offset of the mapped attribute from the start of the Output Parameter Data Area.
- **Length for Attribute N**
  Length (in bytes) of the mapped attribute.

Get Reset Parameter (GET_ID_RESET_PARAM)

Description

Using this command, it is possible to determine what type of reset that has been received from ControlNet.

If the application should be notified about fieldbus reset requests, the feature must be enabled in the ANYBUS_INIT mailbox command during module initialization:

- In ANYBUS_INIT, set the RDR bit of the Operation Mode register
- In ANYBUS_INIT, set the RST bit of the Event Notification Config register

If enabled, the module will generate an Event Notification when a fieldbus reset request has been encountered.

Note: This functionality requires firmware release 1.50 or higher.

Parameter	Description
Command initiator	Application
Command Name	GET_ID_RESET_PARAM
Message type	02h
Command number	0095h
Extended Header data	-
Command data	-
Response data	Reset Type

Command and response layout:

	Command	Expected response	
Message ID	(ID)	(ID)	
Message information	4002h	0002h	
Command	0095h	0095h	*GET_ID_RESET_PARAM*
Data size	0000h	0001h	
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	-	-	
Extended word 2	-	-	
Extended word 3	-	-	
Extended word 4	-	-	
Extended word 5	-	-	
Extended word 6	-	-	
Extended word 7	-	-	
Extended word 8	-	Fault Code	*(See 4-1 "Fault Codes")*
		Reset Type	**Response data byte 1**

Reset Type	Action
00h - "Power On Reset"	Application shall emulate a power on or restart
01h - "Out of Box Config."	Application shall reset all settings to their factory default value and reset the module
FFh - (No reset)	Do nothing

Set Serial Number (SET_SERIAL_NUMBER)

Description

This command enables the application to set a custom serial number in the product.

Note 1: Each CIP module from a vendor must have a unique serial number.

Note 2: This command may only be issued during the initialization.

Parameter	Description
Command initiator	Application
Command Name	SET_SERIAL_NUMBER
Message type	02h
Command number	0081h
Extended Header data	-
Command data	Contains the custom serial number to be set in the product.
Response data	The response indicates if the data was accepted.

Command and response layout:

	Command	Expected response	
Message ID	(ID)	(ID)	
Message information	4002h	0002h	
Command	0081h	0081h	*SET_SERIAL_NUMBER*
Data size	0004h	0004h	*4 bytes of data*
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	-	-	
Extended word 2	-	-	
Extended word 3	-	-	
Extended word 4	-	-	
Extended word 5	-	-	
Extended word 6	-	-	
Extended word 7	-	-	
Extended word 8	-	-	
Message data byte 1	Serial number (msb)	Serial number (msb)	*Serial number (msb)*
Message data byte 2	Serial number	Serial number	*Serial number*
Message data byte 3	Serial number	Serial number	*Serial number*
Message data byte 4	Serial number (lsb)	Serial number (lsb)	*Serial number (lsb)*

Set All Product Info (SET_PRODUCT_INFO_ALL)

Description

This command enables the application to change Vendor ID, Device Type, Product Code, Major Revision, Minor Revision, and Product Name in the Identity object, see 3-2 "Identity Object, Class 01h", to customize the developed product.

Note 1: The EDS file needs to be modified to fit the product, see 1-1 "EDS File".

Note 2: This command may only be issued during the initialization.

Parameter	Description
Command initiator	Application
Command Name	SET_PRODUCT_INFO_ALL
Message type	02h
Command number	0089h
Extended Header data	-
Command data	Vendor ID, Device Type, Product Code, Major revision, Minor Revision, Name length, Name (ascii values)
Response data	The response indicates if the data was accepted.

Command and response layout:

	Command	Expected response	
Message ID	(ID)	(ID)	
Message information	4002h	0002h	
Command	0089h	0089h	*SET_PRODUCT_INFO_ALL*
Data size	(data size)	(data size)	*Message data size in bytes*
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	-	-	
Extended word 2	-	-	
Extended word 3	-	-	
Extended word 4	-	-	
Extended word 5	-	-	
Extended word 6	-	-	
Extended word 7	-	-	
Extended word 8	-	-	
Message data byte 1	Vendor ID (msb)	Vendor ID (msb)	*Vendor ID (msb)*
Message data byte 2	Vendor ID (lsb)	Vendor ID (lsb)	*Vendor ID (lsb)*
Message data byte 3	Device type (msb)	Device type (msb)	*Device type (msb)*
Message data byte 4	Device type (lsb)	Device type (lsb)	*Device type (lsb)*
Message data byte 5	Product code (msb)	Product code (msb)	*Product code (msb)*
Message data byte 6	Product code (lsb)	Product code (lsb)	*Product code (lsb)*
Message data byte 7	Major revision	Major revision	*Major revision*
Message data byte 8	Minor revision	Minor revision	*Minor revision*
Message data byte 9	Product name length	Product name length	*Length of product name (hexadecimal) max 32 characters*
Message data byte 10	Product name1st character	Product name1st character	*Product name,1st character*

Message data byte 11	Product name 2nd character	Product name 2nd character	*Product name, 2nd character*
.	-	-	-
Message data byte n	Product name last character	Product name last character	*Product name, last character*

Register Class (REGISTER_CLASS)

Description

This command enables the application to register objects inside the message router object. If there is a node in the network that sends an explicit message request to the module, addressed to the registered class, the explicit message will cause the module to spontaneously issue the mailbox message 'UCMM Request', see 4-16 "UCMM Request (UCMM_REQUEST)".

Note: This functionality requires firmware release 1.50 or higher.

Parameter	Description
Command initiator	Application
Command Name	REGISTER_CLASS
Message type	02h
Command number	008Bh
Extended Header data	Fault Code
Command data	Contains the class id of the class to be registered in the Anybus module.
Response data	The response indicates if the data was accepted.

Command and response layout:

	Command	Expected response	
Message ID	(ID)	(ID)	
Message information	4002h	0002h	
Command	008Bh	008Bh	*REGISTER_CLASS*
Data size	0002h	0002h	*2 bytes of data*
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	-	-	
Extended word 2	-	-	
Extended word 3	-	-	
Extended word 4	-	-	
Extended word 5	-	-	
Extended word 6	-	-	
Extended word 7	-	-	
Extended word 8	-	Fault Code	*(See 4-1 "Fault Codes")*
Message data byte 1	Class ID (msb)	Class ID (msb)	
Message data byte 2	Class ID (lsb)	Class ID (lsb)	

- **Class ID**

  ID of the class to be registered in the Anybus module.

Deregister Class (DEREGISTER_CLASS)

Description

This command makes it possible de-register objects inside the message router object.

The following classes cannot be de-registered:

- class 02h, Message router
- class 04h, Assembly object

Note: This functionality requires firmware release 1.50 or higher.

Parameter	Description
Command initiator	Application
Command Name	DEREGISTER_CLASS
Message type	02h
Command number	008Eh
Extended Header data	Fault Code
Command data	Contains the class id of the class to be deregistered in the Anybus module.
Response data	The response indicates if the data was accepted.

Command and response layout:

	Command	Expected response	
Message ID	(ID)	(ID)	
Message information	4002h	0002h	
Command	008Eh	008Eh	*DEREGISTER_CLASS*
Data size	0002h	0002h	*2 bytes of data*
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	-	-	
Extended word 2	-	-	
Extended word 3	-	-	
Extended word 4	-	-	
Extended word 5	-	-	
Extended word 6	-	-	
Extended word 7	-	-	
Extended word 8	-	Fault Code	*(See 4-1 "Fault Codes")*
Message data byte 1	Class ID (msb)	Class ID (msb)	
Message data byte 2	Class ID (lsb)	Class ID (lsb)	

- **Class ID**

  ID of the class to be de-registered in the Anybus module.

UCMM Request (UCMM_REQUEST)

Description

This message is spontaneously sent by the module when it has received an explicit message request to a class previously registered using the mailbox command 'Register Class', see 4-12 "Set All Product Info (SET_PRODUCT_INFO_ALL)". The format of the message is the router/request format (See ControlNet specification).

The application must process this message, and provide a response to the module containing the data necessary to generate a response to the explicit message request.

Note 1: Unlike the 'Send UCMM' command, the EPATH in the message data of this command is stored in big endian format.

Note 2: This functionality requires firmware release 1.50 or higher.

Parameter	Description
Command initiator	Anybus
Command Name	UCMM_REQUEST
Message type	02h
Command number	008Dh
Extended Header data	-
Command data	Explicit message data. EPATH in big endian order.
Response data	The response is the data that the explicit request message asks for (or error code)

Command and response layout:

	Command	Expected response	
Message ID	(ID)	(ID)	
Message information	4002h	0002h	
Command	008Dh	008Dh	*UCMM_REQUEST*
Data size	(data size)	(data size)	*N bytes of data*
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	-	-	
Extended word 2	-	-	
Extended word 3	-	-	
Extended word 4	-	-	
Extended word 5	-	-	
Extended word 6	-	-	
Extended word 7	-	-	
Extended word 8	-	-	
Message data byte 1	Service Request	Reply Service	
Message data byte 2	Request Path Size (in words)	Reserved (00h)	
Message data byte 3	Padded EPATH[a]	General Status	
...	(Big endian order)	Size of Additional Status	*Value = 0 or 1 word*
...	(optional service data)	Additional Status	
Message data byte N		Response Data	

a. Consult the ControlNet specification for further information.

Send UCMM (SEND_UCMM)

Description

This command is used to send an explicit unconnected message from the application, directly to a node on the network. The module will respond to the application when the addressed node has replied to the request, or when the message request times out.

Note 1: Unlike the UCMM Request message, the EPATH in the message data of this command is stored in little endian format.

Note 2: This functionality requires firmware release 1.50 or higher.

Parameter	Description
Command initiator	Application
Command Name	SEND_UCMM
Message type	02h
Command number	008Ah
Extended Header data	Destination Mac ID, possible fault information
Command data	Explicit message data. EPATH in little endian format.
Response data	The response is the data that the explicit request message asks for (or error code)

Command and response layout:

	Command	Expected response	
Message ID	(ID)	(ID)	
Message information	4002h	0002h	
Command	008Ah	008Ah	*SEND_UCMM*
Data size	(data size)	(data size)	*N bytes of data*
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	Destination MAC ID	Destination MAC ID	
Extended word 2	-	-	
Extended word 3	-	-	
Extended word 4	-	-	
Extended word 5	-	-	
Extended word 6	-	-	
Extended word 7	-	Extended Fault Code	*(See below)*
Extended word 8	-	Fault Code	*(See 4-1 "Fault Codes")*
Message data byte 1	Service Request	(Service Dependant)	
Message data byte 2	Request Path Size (in words)		
Message data byte 3	Padded EPATH[a] (Little endian order)		
...			
...	(optional service data)		
Message data byte N			

a. Consult the ControlNet Specification for further information.

- **Extended Fault Code**

  0000h: No additional fault information
  0B89h: Timeout

Copy I/O Status (COPY_IO_STATUS)

Description

If this command is sent to the module, the first four bytes in the connection (a.k.a. the Run/Idle header) will not be stripped off, and no 32-bit status header will be inserted in front of the data sent to the master.

Note 1: This message can only be sent during module initialization.

Note 2: This functionality requires firmware release 1.50 or higher.

Parameter	Description
Command initiator	Application
Command Name	COPY_IO_STATUS
Message type	02h
Command number	0094h
Extended Header data	-
Command data	-
Response data	-

Command and response layout:

	Command	Expected response	
Message ID	(ID)	(ID)	
Message information	4002h	0002h	
Command	0094h	0094h	*COPY_IO_STATUS*
Data size	0000h	0000h	
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	-	-	
Extended word 2	-	-	
Extended word 3	-	-	
Extended word 4	-	-	
Extended word 5	-	-	
Extended word 6	-	-	
Extended word 7	-	-	
Extended word 8	-	-	

Enable Large I/O (ENABLE_LARGE_IO)

Description

This command enables the module to use up to 512 bytes of I/O data. If the command is not sent, the module will only be able to use 450 bytes.

Note: This message can only be sent during module initialization.

Parameter	Description
Command initiator	Application
Command Name	ENABLE_LARGE_IO
Message type	02h
Command number	009Bh
Extended Header data	-
Command data	-
Response data	The response indicates if the data was accepted.

Command and response layout:

	Command	Expected response	
Message ID	(ID)	(ID)	
Message information	4002h	0002h	
Command	009Bh	009Bh	*ENABLE_LARGE_IO*
Data size	0000h	0000h	
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	-	-	
Extended word 2	-	-	
Extended word 3	-	-	
Extended word 4	-	-	
Extended word 5	-	-	
Extended word 6	-	-	
Extended word 7	-	-	
Extended word 8	-	-	

Enable Large UCMM Requests (ENABLE_LARGE_UCMM_REQUEST)

Description

This command enables the module to send UCMM requests larger than 256 bytes to application registered CIP classes. If the command has been sent to the module, the UCMM_REQUEST commands will be replaced by LARGE_UCMM_REQUEST commands. All requests to application registered classes will then be performed using LARGE_UCMM_REQUEST commands.

Note: This message can only be sent during module initialization.

Parameter	Description
Command initiator	Application
Command Name	ENABLE_LARGE_UCMM_REQUEST
Message type	02h
Command number	00D0h
Extended Header data	-
Command data	-
Response data	The response indicates if the data was accepted.

Command and response layout:

	Command	Expected response	
Message ID	(ID)	(ID)	
Message information	4002h	0002h	
Command	00D0h	00D0h	*ENABLE_LARGE_UCMM_REQUEST*
Data size	0000h	0000h	
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	-	-	
Extended word 2	-	-	
Extended word 3	-	-	
Extended word 4	-	-	
Extended word 5	-	-	
Extended word 6	-	-	
Extended word 7	-	-	
Extended word 8	-	-	

Large UCMM Request (LARGE_UCMM_REQUEST)

Description

This message is used when the application has registered a CIP class (See 4-14 "Register Class (REGISTER_CLASS)" in the Anybus module, and an explicit message request has been sent to this class from a ControlNet client. The application will have to process the message, and respond to the Anybus module with the data necessary to generate a response to the explicit message request for the class.

Parameter	Description
Command initiator	Application
Command Name	LARGE_UCMM_REQUEST
Message type	02h
Command number	00D1h
Extended Header data	-
Command data	-
Response data	The response indicates if the data was accepted.

Command and response layout

	Command	Expected response	
Message ID	(ID)	(ID)	
Message information	4002h	0002h	*Fieldbus Specific Message*
Command	00D1h	00D1h	*LARGE_UCMM_REQUEST*
Data size	(size)	(size)	*Data size*
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	Fragment type	Fragment type	*Fragmentation information*
Extended word 2	-	-	"
Extended word 3	-	-	"
Extended word 4	-	-	"
Extended word 5	-	-	"
Extended word 6	-	-	"
Extended word 7	-	-	"
Extended word 8	-	-	"
Message data byte 1	Service Request	Service Reply	
Message data byte 2	Request Path Size (in words)	Reserved (00h)	
Message data byte 3	Padded EPATH[a]	General Status	
...		Size of Additional Status	*0 or 1 word*
...	(optional service data)	Additional Status	
Message data byte n		Optional data	

a. EPATH is in big endian format

Fragment Type

Fragment Type	Description
0000h	First fragment of new message
0001h	Subsequent fragment
0002h	Last fragment. Signals the end of the fragmented data.

Example 1

This example mailbox is from a CIP UCMM request to the application registered class CCh, instance 1, attribute 2. The object specific service code 54h is used. 6 bytes of data are sent (11h, 22h, 33h, 44h, 55h and 66h) in 3 mailbox fragments. The application returns 6 bytes of payload data (12h, 34h, 56h, 78h, 9Ah and BCh) in 3 mailbox fragments.

Please note that this is an example. The actual fragmentation from the module will be in chunks of 256 bytes.

	Command	Expected response	
Message ID	0001h	0001h	
Message information	4002h	0002h	
Command	00D1h	00D1h	
Data size	000Ah	0000h	*Number of bytes*
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	0000h	0000h	*Fragmentation information*
Extended word 2	-	-	"
Extended word 3	-	-	"
Extended word 4	-	-	"
Extended word 5	-	-	"
Extended word 6	-	-	"
Extended word 7	-	-	"
Extended word 8	-	-	"
Message data byte 1	54h		
Message data byte 2	03h		
Message data byte 3	20h		
Message data byte 4	CCh		
Message data byte 5	24h		
Message data byte 6	01h		
Message data byte 7	30h		
Message data byte 8	02h		
Message data byte 9	11h		
Message data byte 10	22h		

	Command	Expected response	
Message ID	0001h	0001h	
Message information	4002h	0002h	
Command	00D1h	00D1h	
Data size	0002h	0000h	*Number of bytes*
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	00001h	0000h	*Fragmentation information*
Extended word 2	-	-	"
Extended word 3	-	-	"
Extended word 4	-	-	"
Extended word 5	-	-	"
Extended word 6	-	-	"
Extended word 7	-	-	"
Extended word 8	-	-	"
Message data byte 1	33h		
Message data byte 2	44h		

	Command	Expected response	
Message ID	0001h	0001h	
Message information	4002h	0002h	
Command	00D1h	00D1h	
Data size	0002h	0006h	*Number of bytes*
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	0002h	0000h	*Fragmentation information*
Extended word 2	-	-	"
Extended word 3	-	-	"
Extended word 4	-	-	"
Extended word 5	-	-	"
Extended word 6	-	-	"
Extended word 7	-	-	"
Extended word 8	-	-	"
Message data byte 1	55h	D4h	
Message data byte 2	66h	00h	
Message data byte 3		00h	
Message data byte 4		00h	
Message data byte 5		12h	
Message data byte 6		34h	

	Command	Expected response	
Message ID	0001h	0001h	
Message information	4002h	0002h	
Command	00D1h	00D1h	
Data size	0000h	0002h	*Number of bytes*
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	0000h	0001h	*Fragmentation information*
Extended word 2	-	-	"
Extended word 3	-	-	"
Extended word 4	-	-	"
Extended word 5	-	-	"
Extended word 6	-	-	"
Extended word 7	-	-	"
Extended word 8	-	-	"
Message data byte 1		56h	
Message data byte 2		78h	

	Command	Expected response	
Message ID	0001h	0001h	
Message information	4002h	0002h	
Command	00D1h	00D1h	
Data size	000Ah	0000h	*Number of bytes*
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	0000h	0002h	*Fragmentation information*
Extended word 2	-	-	"
Extended word 3	-	-	"
Extended word 4	-	-	"
Extended word 5	-	-	"
Extended word 6	-	-	"
Extended word 7	-	-	"
Extended word 8	-	-	"
Message data byte 1		9Ah	
Message data byte 2		BCh	

Example 2

This example is from a CIP UCMM request to the application registered class CCh, instance 1, attribute 3. The service code 0Eh is used. The application returns 6 bytes of payload data (AAh, BBh, CCh, DDh, EEh and FFh).

	Command	Expected response	
Message ID	0001h	0001h	
Message information	4002h	0002h	
Command	00D1h	00D1h	
Data size	0008h	000Ah	*Number of bytes*
Frame count	0001h	0001h	
Frame number	0001h	0001h	
Offset high	0000h	0000h	
Offset low	0000h	0000h	
Extended word 1	0002h	0002h	*Fragmentation information*
Extended word 2	-	-	“
Extended word 3	-	-	“
Extended word 4	-	-	“
Extended word 5	-	-	“
Extended word 6	-	-	“
Extended word 7	-	-	“
Extended word 8	-	-	“
Message data byte 1	0Eh	8Eh	
Message data byte 2	03h	00h	
Message data byte 3	20h	00h	
Message data byte 4	CCh	00h	
Message data byte 5	24h	AAh	
Message data byte 6	01h	BBh	
Message data byte 7	30h	CCh	
Message data byte 8	03h	DDh	
Message data byte 9		EEh	
Message data byte 10		FFh	

Fieldbus Specific Registers

Fieldbus Specific Area

This area holds runtime status information from the module and the ControlNet network. The information that can be read from this area is specified below:

640h	Net Mode	(see below)
641h	MacID	(see below)
642h	Connection State	(see below)
643h	Run/Idle	(see below)
644h - 7BFh	(reserved)	-

Net Mode

This register holds information about the state of the module.

Value	Description
00h	Invalid
01h	Start-up of bus controller
02h	Check for cable
03h	Waiting to rogue
04h	Check for moderator (listen only)
05h	I'm alive (Module identifies itself on the network)
06h	Attached
07h	Forced listen only
08h	Duplicate (listen only)

MacID

This register holds the currently used MacID.

Connection State

This register indicates if a connection has been opened towards the module.

- **Value**

  00h: Module is not owned
  01h: Module is owned (a connection has been opened towards the module)

Run/Idle

This register indicates the operating state of the module.

- **Value**

  00h: Connection in Idle state
  01h: Connection in Run state

Module Identification

The following registers are located in the Control Register Area, and can be used by the application to identify the module.

- **Module Type Value**
  0101h: Anybus-S Slave
- **Fieldbus Type Value**
  0065h: ControlNet

Electrical Characteristics

Power Supply

Supply Voltage

Both the module electronics and the fieldbus interface requires a regulated 5V DC power supply. For more information regarding power requirements, consult the Anybus-S Parallel Design Guide.

Power Consumption

Symbol	Description	Min.	Typ.	Max	Unit
I_{INMOD}	Supply current, module electronics	-	100	100	mA
I_{INBUS}	Supply current, bus interface	-	210	230	mA

Protective Earth (PE)

A PE-connection is included on one of the mounting holes according to the Anybus-S specification.

Galvanic Isolation

ControlNet channels A & B are galvanically isolated from the module electronics.

Connectors

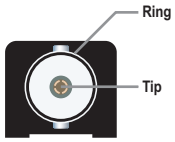
Application Interface

The module features a standard Anybus-S dual port memory interface. Consult the Anybus-S Parallel Design Guide for further information.

ControlNet Connector Pinout, Channel A & B (BNC)

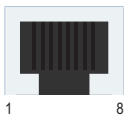
ControlNet connectivity is provided through two BNC coaxial connectors.

#	Signal
Tip	ControlNet
Ring	Shield



Network Access Port Connector Pinout (RJ45)

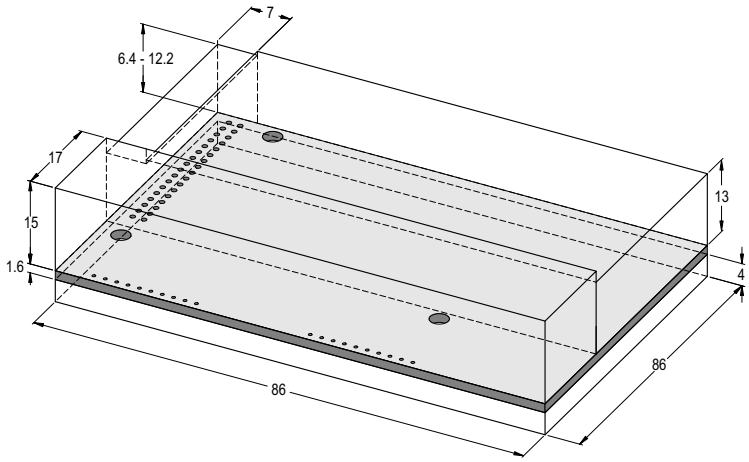
#	Signal	Description
1	GND	Signal Ground
2	-	(not connected)
3	Tx_H	Transmit Data, positive
4	Tx_L	Transmit Data, negative
5	Rx_L	Receive Data, negative
6	Rx_H	Receive Data, positive
7	-	(not connected)
8	Shield	Connected to PE



Mechanical Specification

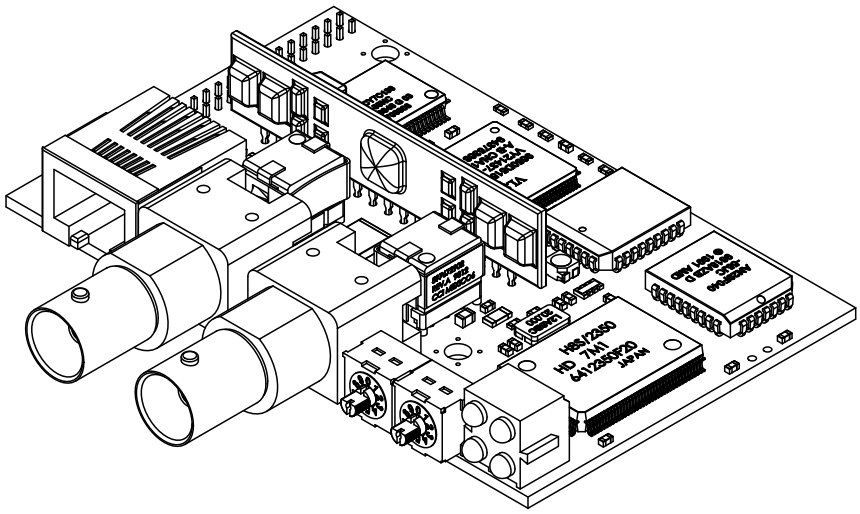
Height Limitations

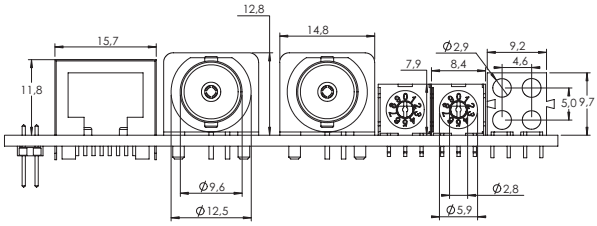
Due to component restrictions, the Anybus-S ControlNet does not fully conform to the Anybus-S height limit specification, see figure below.

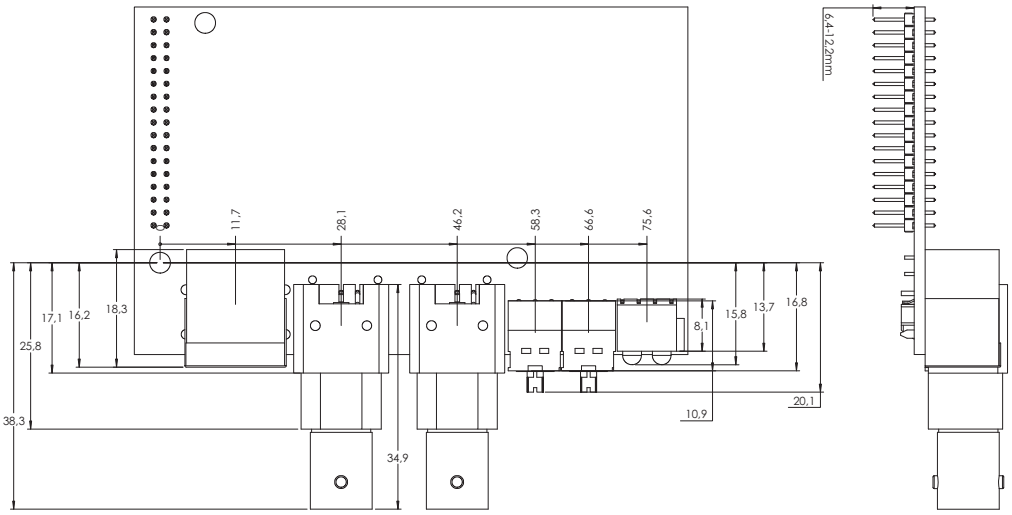


Consult the general Anybus-S Parallel Design Guide for further information.

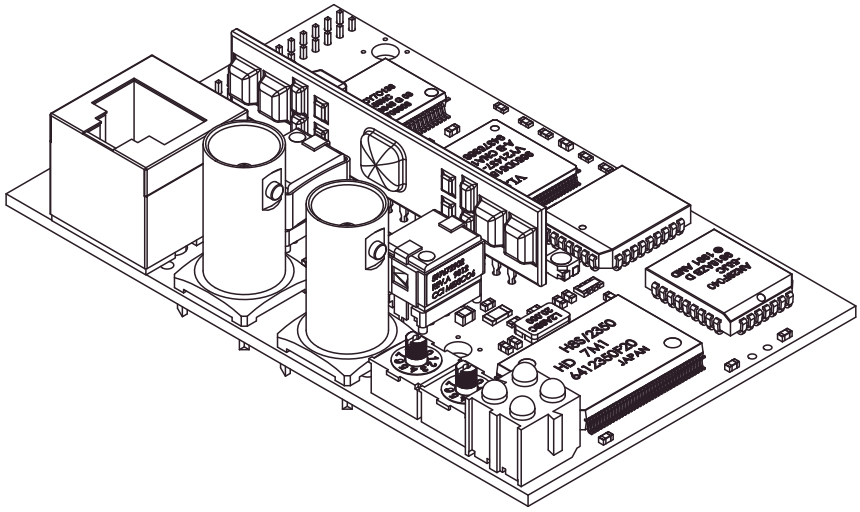
Angled Switches & Connectors

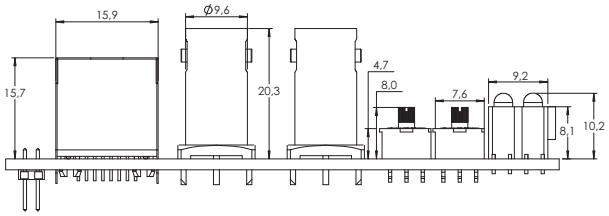


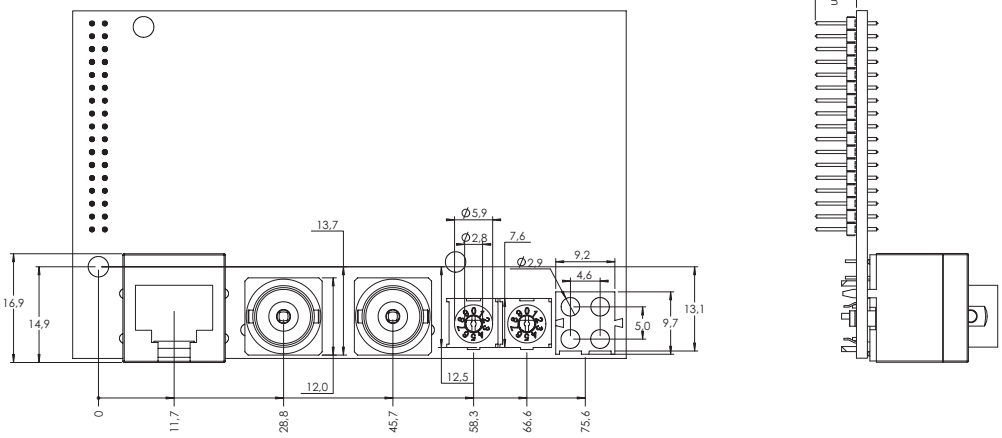




Straight Switches & Connectors







Environmental Specification

Temperature

Operating

+0 to +70 degrees Celsius

Test performed according to IEC-68-2-1 and IEC 68-2-2.

Non Operating

-15 to +85 degrees Celsius

Test performed according to IEC-68-2-1 and IEC 68-2-2.

Relative Humidity

The product is designed for a relative humidity of 5 to 95% non-condensing.

Test performed according to IEC 68-2-30.

EMC compliance

Emission

According to EN 50 081-2:1993

Tested per 55011:1990, class A, radiated

Immunity

According to EN 61000-6-2:1999

Tested per EN 61000-4-2:1995

EN 61000-4-3:1996

EN 61000-4-4:1995

EN 61000-4-5:1995

EN 61000-4-6:1996