Anybus CompactCom 40 Diagnostic Events for Modbus-TCP

APPLICATION NOTE

SCM-1202–039 1.1 ENGLISH

Important User Information

Liability

Every care has been taken in the preparation of this document. 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.

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 USA and other countries.

1 Preface

1.1 About this Document

This application note is intended to provide a description about how diagnostic events are presented in the engineering tool for the industrial network Modbus-TCP.

It is divided into two parts:

Part one provides a short overview of the Anybus CompactCom 40 Diagnostic Object (02h) and its diagnostic events. Part two is an example, showing how to get diagnostic messages displayed for Modbus-TCP using the PLC engineering tool Unity Pro XL V8.0.

1.1.1 Target Audience

This document is meant for trained and skilled personnel working with the equipment described.

You need electrical engineering skills for the installation and commissioning of electrical equipment.

You also need general knowledge of automation and programmable logic controllers, in particular about the Schneider automation software Unity Pro XL. Additionally knowledge on the Modbus-TCP Industrial Ethernet protocol and the Anybus CompactCom (ABCC) object model is necessary.

1.2 History

Revision	Date	Description	Responsible
1.0	2016-10-17	First version	OLB
1.1	2017-02-09	Converted to DOX	KaD

1.3 Referenced Documents

Description	Name / Type	Version
HMS Starter Kit	HMS Development Board 2 Rev 1.07	0314-1.1.1
Anybus CompactCom 40 module	ABCC-M40-EIT	V.1.06
Schneider Electric Modicon M340 Station Components	CPU: BMX P34 20302 CM: BMX NOC 0401.2	SV: 2.40 SV: 2.10
Anybus CompactCom 40 Modbus-TCP	Network Guide	V1.10
Anybus CompactCom Software Design guide	Software Guide	V3.0
PC with Schneider PLC programming software	Unity Pro XL	V8.0 - 131118
IDE	KEIL uVision 5	V5.20
Anybus CompactCom Driver	Anybus CompactCom Host Application Example Code	V2.01

1.4 More Information about Networks and Products

The latest manuals and EDS files can be found on the HMS website, www.anybus.com

1.5 Trademark Information

Anybus® is a registered trademark of HMS Industrial Networks AB.

All other trademarks are the property of their respective holders.

2 Introduction

The ability of an automation device to raise diagnostic events is an important benefit. It may not only reflect a higher level of quality setting it apart from competitors in the market, in the end it will also increase the device’s reliability, reduce downtimes by relieving preventive maintenance and protect the investment.

Implementing this feature set allows the device to signal significant incidents towards the PLC. This way operators will immediately be informed in case the device experiences errors or faults.

This document describes how developers of a field device implementing an Anybus Compact-Com 40 Modbus-TCP network interface can use its features to create diagnostics events.

3 The Diagnostic Object

The Anybus CompactCom concept is based on an object model.

For detailed information about this, please refer to the Anybus CompactCom 40 Software Design Guide.

For creating diagnostics (information from a field device to a PLC) the CompactCom concept contains the diagnostic object (02h) which is located inside the CompactCom. A diagnostic event is created by sending a **create *(03h)*** command to the **diagnostic object (02h)** of the CompactCom.

3.1 Create a Diagnostic Event

When the CompactCom device has been started and initialized, the diagnostic object (02h) is solely built up of its object instance #0 containing attributes common to all subsequent instances which may be added in the course of the device's operation. The process of reporting a diagnostic event to the network is initiated by the host application. To this end a ***create (03h)*** event command message is sent from the host application to the diagnostic object (02h).

The CompactCom will internally create a new instance inside the diagnostic object (02h). This new instance corresponds to a **diagnostic event**.

The ***create (03h)*** command must contain information for describing the diagnostic event: a **Severity** (CMDExt[0]) and an **Event Code** (CMDExt[1]). The **severity** indicates how critical the event is and if it will recover by itself or not. CMDExt[0] additionally contains a bit called ***Extended Diagnostic***, which informs the CompactCom if the event message contains additional user specific data. If the ***Extended Diagnostic*** bit is set, the additional user specific data will be found in MsgData[8..n], and the standard data in MsgData[0..7].

An **Event Code** informs about the nature of the event, i. e. what caused the device to react, like exceeding temperature or current limit.

Command Details: Create

Details

Command Code:	03h
Valid For:	Object

Description

Creates a new instance, in this case representing a new diagnostic event in the host application.

- Command details:

Field	Contents	Note
CMDExt[0]	Bit 0: Extended Diagnostic Bit 4–6: Severity Other bits Reserved. Set to zero.	
CMDExt[1]	Event Code, see previous page	
MsgData[0...1]	Slot number associated with the event Set to "0" if unknown or unsupported	These fields only exist if bit 0 (Extended Diag-nostic) is set
MsgData[2...3]	ADI associated with the event Set to "0" if unknown or unsupported	
MsgData[4]	Element associated with the event Set to "255" if unknown or unsupported	
MsgData[5]	Bit in element associated with the event Set to "255"if unknown or unsupported	
MsgData[6...7]	Reserved. Set to zero	
MsgData[0/8...n]	Network specific extension (optional, definition is net-work specific)	MsgData[8-n] if bit 0 in CmdExt[0] is set MsgData[0-n] if bit 0 in CmdExt[0] is not set

- Response details (Success):

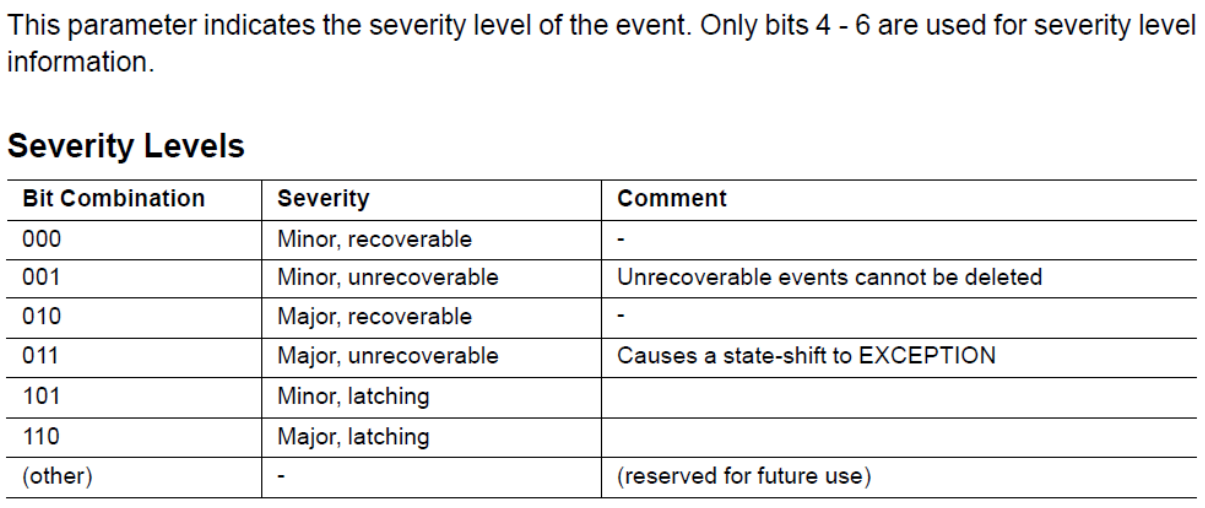
Field	Contents
MsgData[0...1]	The number of the created instance

- Response details (Error):

Error	Contents	Comment
Object Specific Error	MsgData[1] = 02h	Error code (Latching event not supported) The event could not be created since the module does not support latching events
	MsgData[1] = FFh	Error code (Network specific error) The event could not be created due to a network specific reason. Information about the event is found in response MsgData[2-n]

3.2 Severity Codes of Diagnostic Events

The following severity codes are defined for the CompactCom:



This parameter indicates the severity level of the event. Only bits 4 - 6 are used for severity level information.

Severity Levels

Bit Combination	Severity	Comment
000	Minor, recoverable	-
001	Minor, unrecoverable	Unrecoverable events cannot be deleted
010	Major, recoverable	-
011	Major, unrecoverable	Causes a state-shift to EXCEPTION
101	Minor, latching	
110	Major, latching	
(other)	-	(reserved for future use)

Typically, *recoverable* events are generated by the process e.g. if a temperature exceeds a limit value defined by the device manufacturer (e.g. internal temperature of the device exceeds 50° C). The character of this event is typically a warning when the event is defined as *minor*. The temperature of the device is recoverable as the device can cool down again when some heat producers are cut off.

The device manufacturer will add a *major recoverable* event when he wants to inform the PLC that the temperature has reached a critical high temperature which can damage the device.

An ***unrecoverable*** diagnostic event is typically created when the device detects that a sensor is broken. The sensor has to be replaced, it will - normally – not recover by itself. If the device has only one sensor and this sensor is broken the device will create a ***major unrecoverable*** event. The device has to be stopped and to be repaired (replace the broken sensor) before used again otherwise there will be a big risk that the device will be damaged after restart. If the device has some redundant sensors it will create a ***minor unrecoverable*** event informing the users that the broken sensor should be replaced by another one within the next scheduled inspection.

Minor latching* and *major latching allow the creation of *latching diagnostic event.*

In Modbus-TCP, latching events are not supported.

For more information, see the Software Design Guide and the Network Guide of the Anybus CompactCom 40 device.

> **!** The device manufacturer has to define which event will be reported as a diagnostic event to the PLC and which severity has to be used.

3.3 Anybus CompactCom Event Codes

The table below shows a list of the event codes applicable for Anybus CompactCom 40 device.

Event Codes

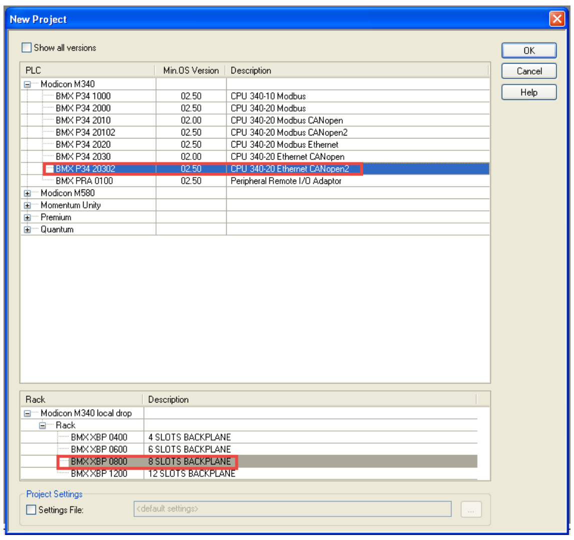
#	Meaning	Comment
10h	Generic Error	-
20h	Current	-
21h	Current, device input side	-
22h	Current, inside the device	-
23h	Current, device output side	-
30h	Voltage	-
31h	Mains Voltage	-
32h	Voltage inside the device	-
33h	Output Voltage	-
40h	Temperature	-
41h	Ambient Temperature	-
42h	Device Temperature	-
50h	Device Hardware	-
60h	Device Software	-
61h	Internal Software	-
62h	User Software	-
63h	Data Set	-
70h	Additional Modules	-
80h	Monitoring	-
81h	Communication	-
82h	Protocol Error	-
90h	External Error	-
F0h	Additional Functions	-
FFh	NW specific	Definition is network-specific; consult separate network guide for further information.

The event code **FFh** is used if network specific diagnostics are reported (not considered within this application note).

4 Modbus-TCP Diagnostics

To show the commissioning within the Schneider Modicon environment, we will be using the Unity Pro XL engineering tool and start with a blank project. In step one, we will configure the PLC, and in step two the Modbus TCP network and its devices.

4.1 PLC Configuration



New Project

☐ Show all versions

OK | Cancel | Help

PLC	Min.OS Version	Description
Modicon M340		
BMX P34 1000	02.50	CPU 340-10 Modbus
BMX P34 2000	02.50	CPU 340-20 Modbus
BMX P34 2010	02.00	CPU 340-20 Modbus CANopen
BMX P34 20102	02.50	CPU 340-20 Modbus CANopen2
BMX P34 2020	02.50	CPU 340-20 Modbus Ethernet
BMX P34 2030	02.00	CPU 340-20 Ethernet CANopen
BMX P34 20302	02.50	CPU 340-20 Ethernet CANopen2
BMX PRA 0100	02.50	Peripheral Remote I/O Adaptor
Modicon M580		
Momentum Unity		
Premium		
Quantum		

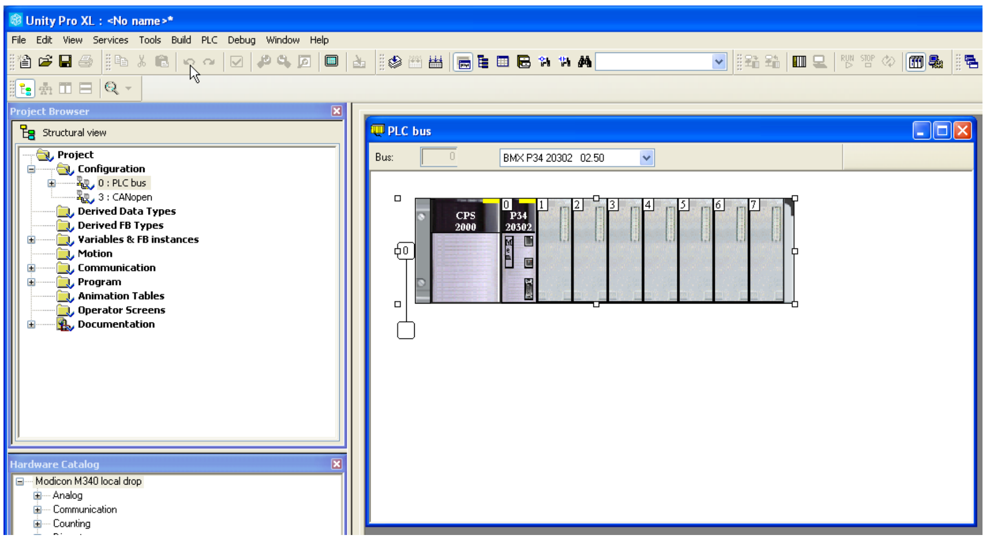
Rack	Description
Modicon M340 local drop	
Rack	
BMX XBP 0400	4 SLOTS BACKPLANE
BMX XBP 0600	6 SLOTS BACKPLANE
BMX XBP 0800	8 SLOTS BACKPLANE
BMX XBP 1200	12 SLOTS BACKPLANE

Project Settings

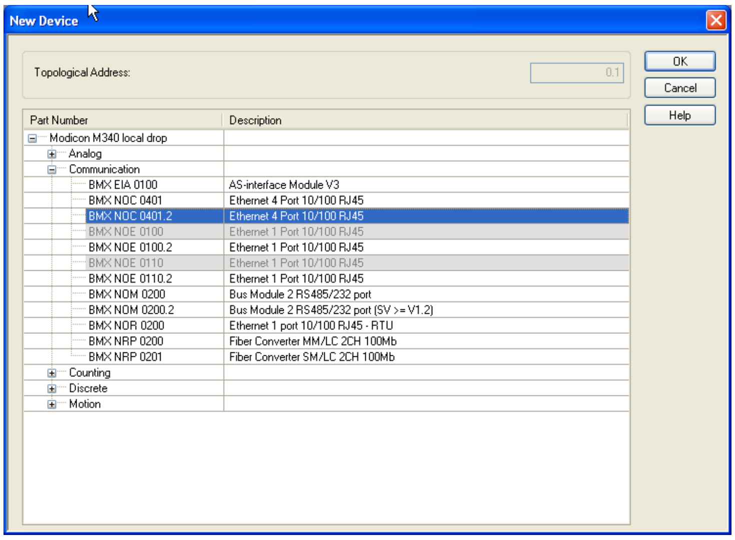
☐ Settings File: <default settings>

Start a Unity Pro XL project by clicking on menu File > New. From the Modicon M340 series of PLCs, select the BMX P34 20302 together with its corresponding rack, in our case an 8 slot backplane as shown in the picture below.

Press OK and right-click on "**PLC bus**" in the navigation list to the left and select **Open**. This will open a new window showing the rack with its power supply, the CPU and 7 empty slots.



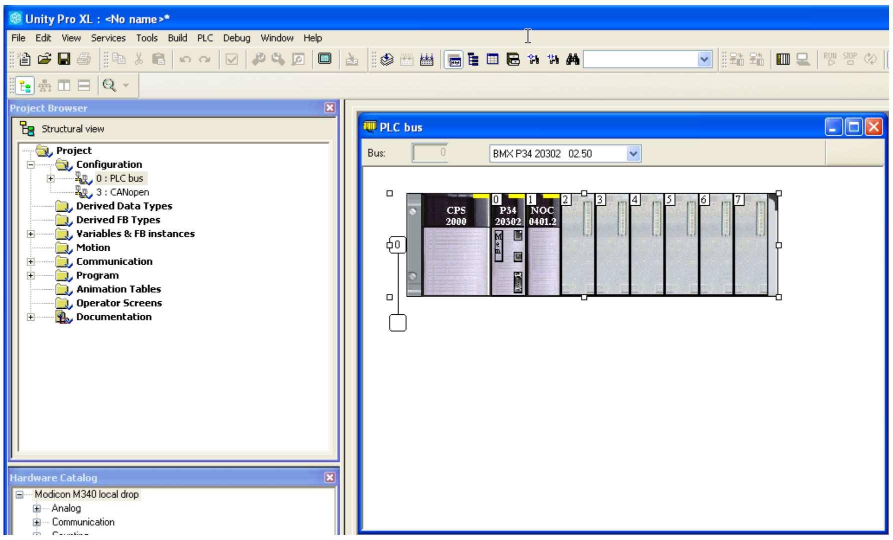
Now add a new communication module to the project by double-clicking on slot 1. This will open a window where you can select the communication module you want to use. In our case, we choose the Ethernet module BMX NOC 0401 from the list.


New Device

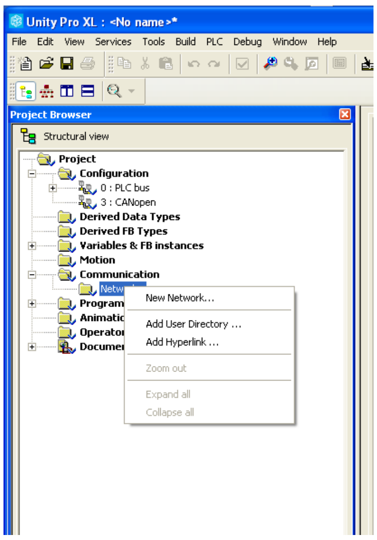
Topological Address: 0.1

Part Number	Description
Modicon M340 local drop	
Analog	
Communication	
BMX EIA 0100	AS-interface Module V3
BMX NOC 0401	Ethernet 4 Port 10/100 RJ45
BMX NOC 0401.2	Ethernet 4 Port 10/100 RJ45
BMX NOE 0100	Ethernet 1 Port 10/100 RJ45
BMX NOE 0100.2	Ethernet 1 Port 10/100 RJ45
BMX NOE 0110	Ethernet 1 Port 10/100 RJ45
BMX NOE 0110.2	Ethernet 1 Port 10/100 RJ45
BMX NOM 0200	Bus Module 2 RS485/232 port
BMX NOM 0200.2	Bus Module 2 RS485/232 port (SV >= V1.2)
BMX NOR 0200	Ethernet 1 port 10/100 RJ45 - RTU
BMX NRP 0200	Fiber Converter MM/LC 2CH 100Mb
BMX NRP 0201	Fiber Converter SM/LC 2CH 100Mb
Counting	
Discrete	
Motion	

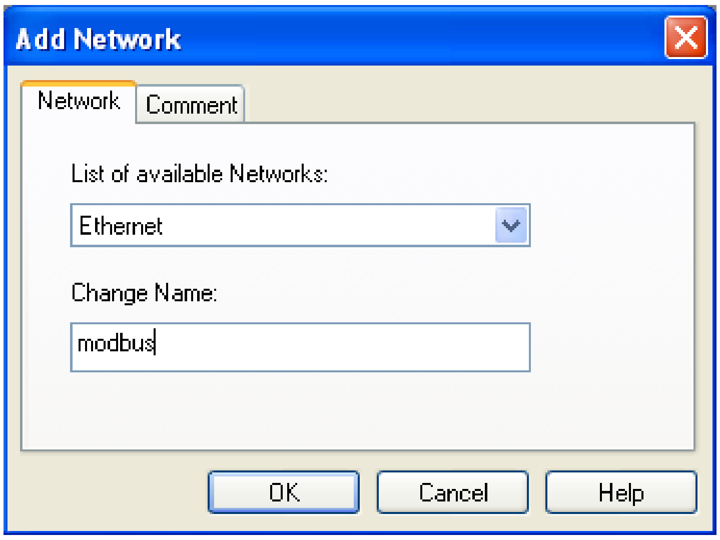
Press OK and the new device will be added as shown in the picture below.



At this stage it is possible to set the IP address of the PLC, which will be taken into effect once the configuration has been downloaded into the PLC. To do this in the project browser, expand the Communication folder then right-click on **Networks** and select **New Network**.



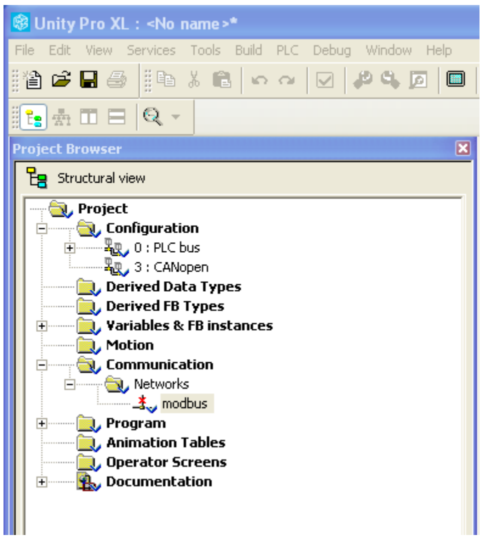
Select Ethernet and choose a name. In our case the network is named “modbus”. Click OK.



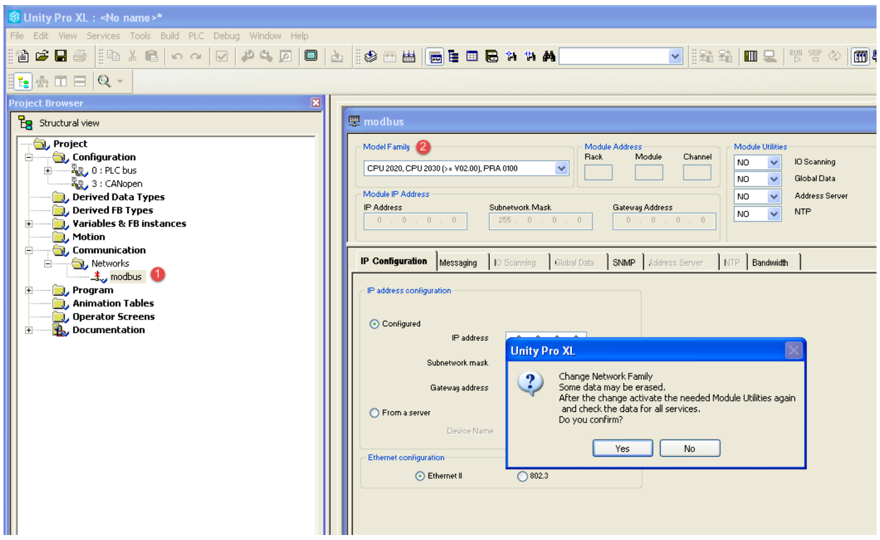
List of available Networks: Ethernet

Change Name: modbus

The red X symbol next to the network name means that the network is not linked to an Ethernet port.



Double-click or right-click on modbus and select **Open**. The picture below will appear



Model Family: CPU 2020, CPU 2030 (>= V02.00), PRA 0100

Module Address: Rack, Module, Channel

Module IP Address: IP Address 0 . 0 . 0 . 0; Subnetwork Mask 255 . 0 . 0 . 0; Gateway Address 0 . 0 . 0 . 0

Module Utilities: IO Scanning NO; Global Data NO; Address Server NO; NTP NO

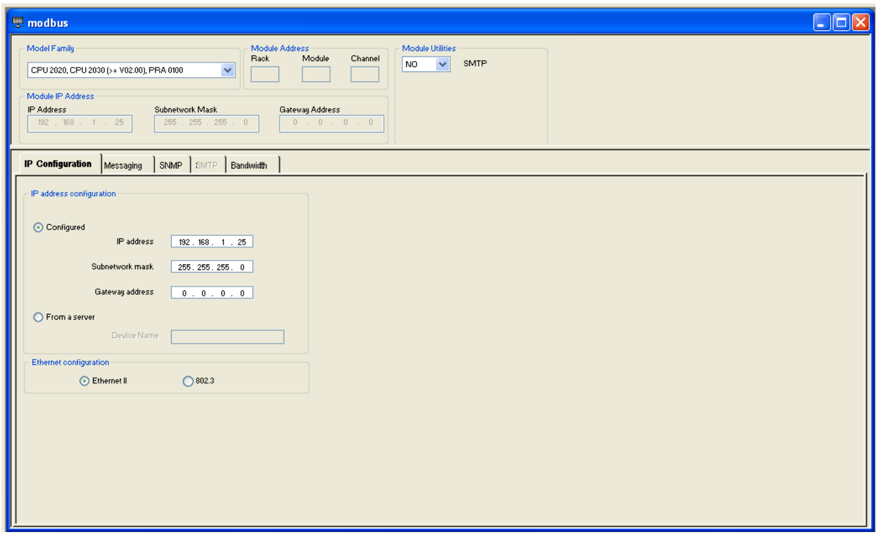
IP address configuration: [✓ Configured] IP address, Subnetwork mask, Gateway address; From a server, Device Name

Ethernet configuration: [✓ Ethernet II] 802.3

Unity Pro XL

Change Network Family
Some data may be erased.
After the change activate the needed Module Utilities again and check the data for all services.
Do you confirm?

Change the model family to **cpu 2020, cpu 2030 (>= V02,00), PRA 0100.** Click on **Yes**.



Model Family: CPU 2020, CPU 2030 (>= V02.00), PRA 0100

Module Utilities: NO

IP Address: 192 . 168 . 1 . 25
Subnetwork Mask: 255 . 255 . 255 . 0
Gateway Address: 0 . 0 . 0 . 0

Configured
IP address: 192 . 168 . 1 . 25
Subnetwork mask: 255 . 255 . 255 . 0
Gateway address: 0 . 0 . 0 . 0

Ethernet II

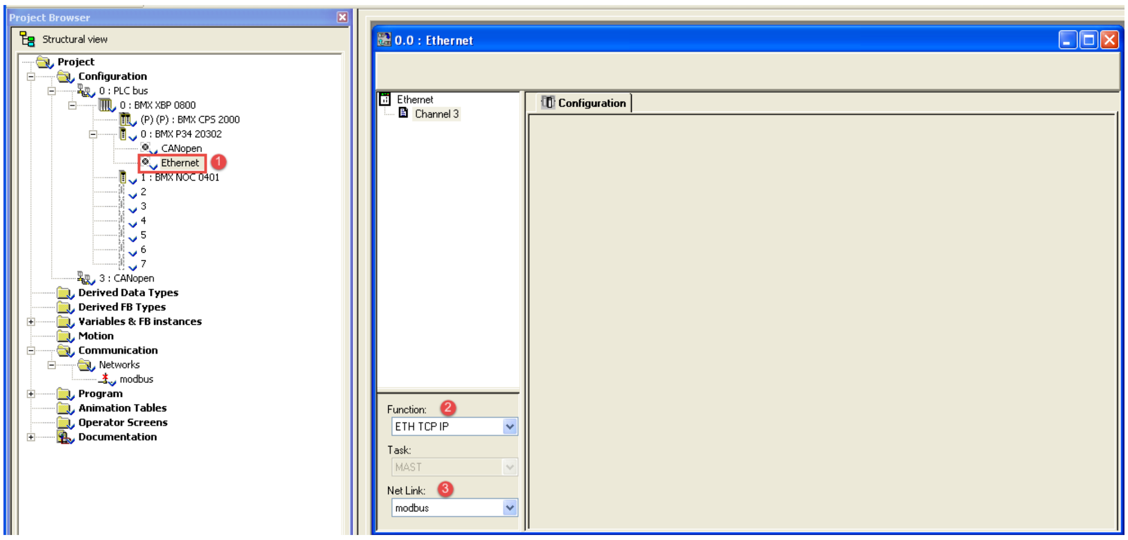
802.3

Enter the IP Address, Subnetwork mask and Gateway address (if needed) as shown in the picture above. Validate your settings in menu Edit > Validate.

In the next step, expand the “**PLC bus**” in the configuration folder and the CPU named **BMX P34 20302**. Double-click on Ethernet and configure the network the PLC module is to be connected to.

Open the configuration in the left navigation list and double-click on the Ethernet interface of the CPU.

The window below will be opened. In the window to the right of the navigation list, select channel 3 under Function *ETH TCP IP* and *modbus* under Net link. Validate the changes by clicking on the black check mark in the project toolbar or in menu Edit > Validate.



Function: ETH TCP IP

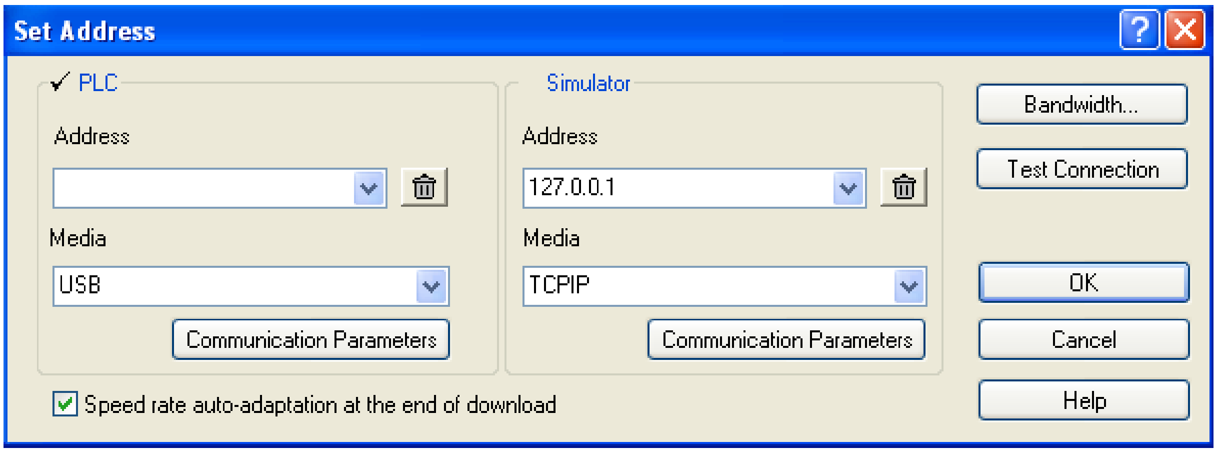
Task: MAST

Net Link: modbus

The connection will be established.

Now you can download your configuration to the PLC. To do this link the PLC to the Unity Pro software through USB (cable), Ethernet, or Modbus.

In the menu PLC select **Set Address.** The following screen appears:



PLC Media: USB

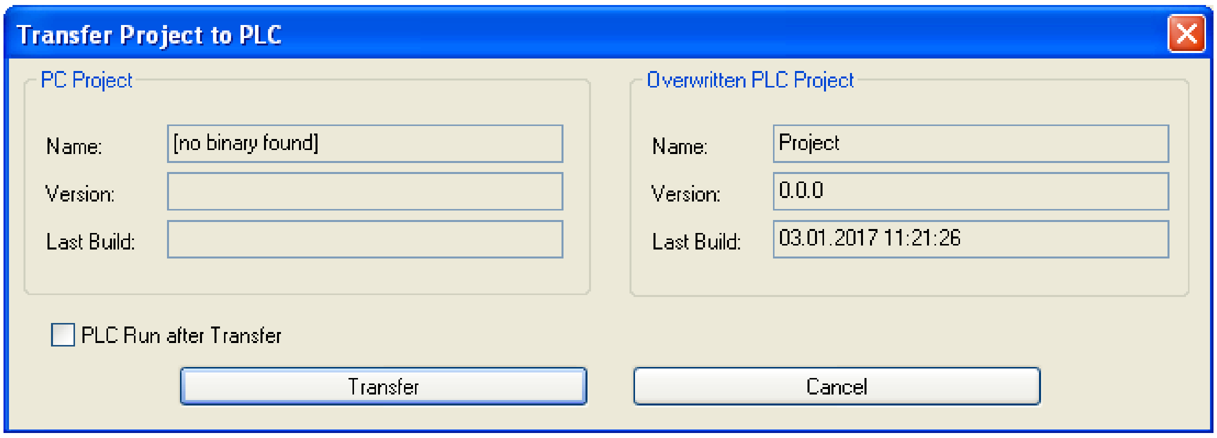
Simulator Address: 127.0.0.1

Simulator Media: TCPIP

☑ Speed rate auto-adaptation at the end of download

In media menu, select USB, as shown in the above figure. Press OK.

On the Unity Pro XL tab, select **PLC** > **Connect** to link the M340 system. Open the transfer project to PLC by selecting **PLC** > **Transfer Project to PLC**



PC Project

Name: [no binary found]

Version:

Last Build:

Overwritten PLC Project

Name: Project

Version: 0.0.0

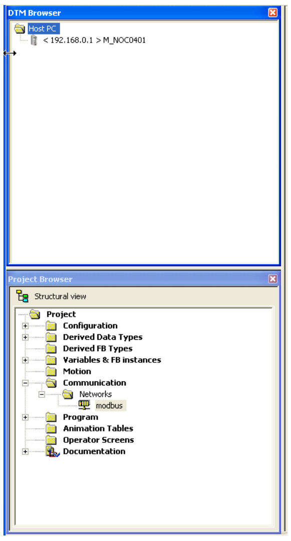
Last Build: 03.01.2017 11:21:26

☐ PLC Run after Transfer

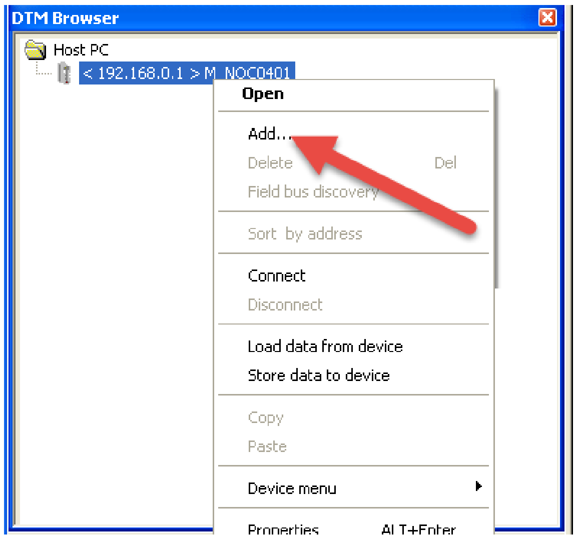
After that, disconnect the PLC by clicking **PLC** > **Disconnect.** Set the IP address at **PLC** > **Set Address** in menu media, select TCPIP and enter the IP address you have configured earlier and then press OK.

4.2 Modbus-TCP Network Configuration

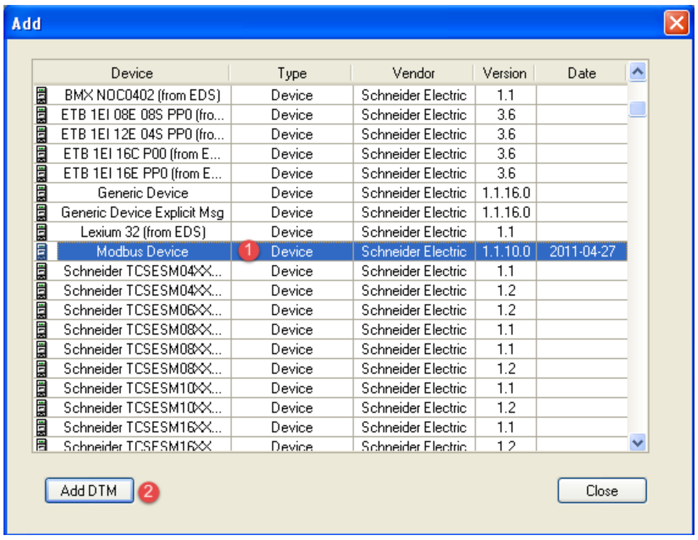
The communication module (BMX NOC401.2) is configured via the DTM browser. DTM stands for device type manager, to integrate remote devices, in this case Anybus CompactCom 40 Modbus TCP. The Anybus CompactCom 40 module will be added to the hardware catalog by means of a DTM.



Adding a device to the configuration means adding the DTM device to the Unity Pro’s DTM browser. To do this, open the DTM browser by clicking on the menu **Tools** and select **DTM browser.**

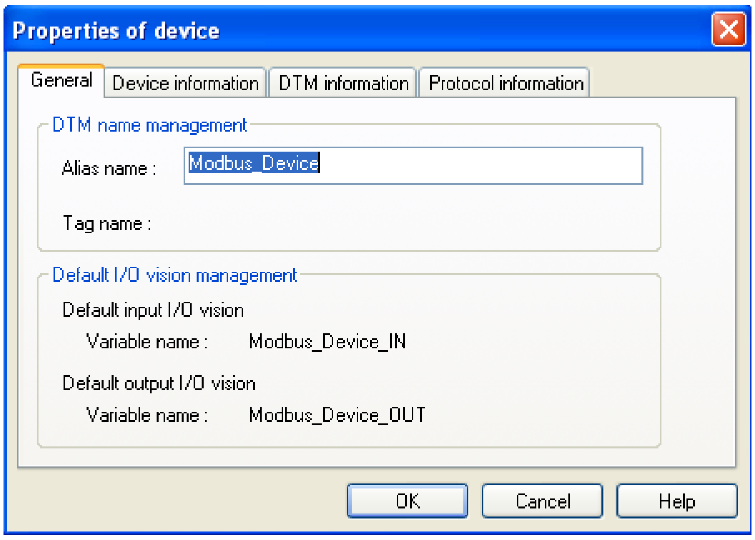


In the DTM browser, select the communication module in order to add the modbus device. Right click on **BMX NOC401.2** then select **Add** . The picture below will appear.



Device	Type	Vendor	Version	Date
BMX NOC0402 (from EDS)	Device	Schneider Electric	1.1	
ETB 1EI 08E 08S PP0 (fro...	Device	Schneider Electric	3.6	
ETB 1EI 12E 04S PP0 (fro...	Device	Schneider Electric	3.6	
ETB 1EI 16C P00 (from E...	Device	Schneider Electric	3.6	
ETB 1EI 16E PP0 (from E...	Device	Schneider Electric	3.6	
Generic Device	Device	Schneider Electric	1.1.16.0	
Generic Device Explicit Msg	Device	Schneider Electric	1.1.16.0	
Lexium 32 (from EDS)	Device	Schneider Electric	1.1	
Modbus Device	Device	Schneider Electric	1.1.10.0	2011-04-27
Schneider TCSESM04XX...	Device	Schneider Electric	1.1	
Schneider TCSESM04XX...	Device	Schneider Electric	1.2	
Schneider TCSESM06XX...	Device	Schneider Electric	1.2	
Schneider TCSESM08XX...	Device	Schneider Electric	1.1	
Schneider TCSESM08XX...	Device	Schneider Electric	1.1	
Schneider TCSESM08XX...	Device	Schneider Electric	1.2	
Schneider TCSESM10XX...	Device	Schneider Electric	1.1	
Schneider TCSESM10XX...	Device	Schneider Electric	1.2	
Schneider TCSESM16XX...	Device	Schneider Electric	1.1	
Schneider TCSESM16XX	Device	Schneider Electric	1.2	

Select **Modbus Device** (1) then Press **Add DTM** (2) as shown above.



DTM name management

Alias name : Modbus_Device

Tag name :

Default I/O vision management

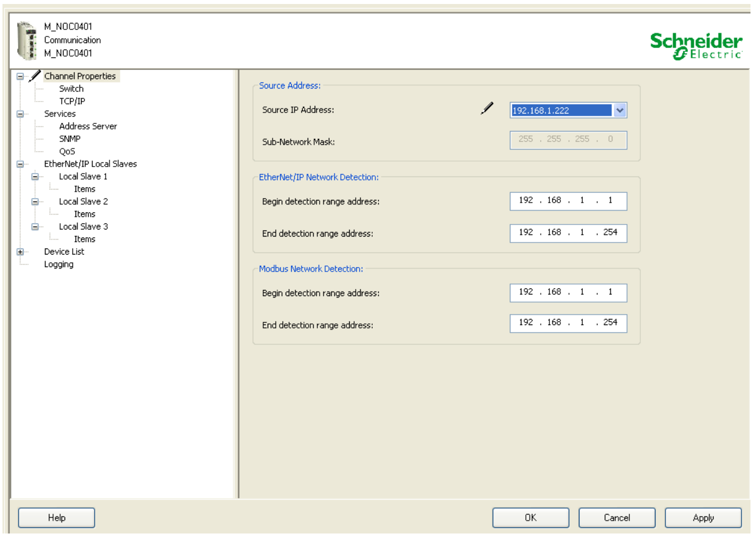
Default input I/O vision

Variable name : Modbus_Device_IN

Default output I/O vision

Variable name : Modbus_Device_OUT

Click OK.

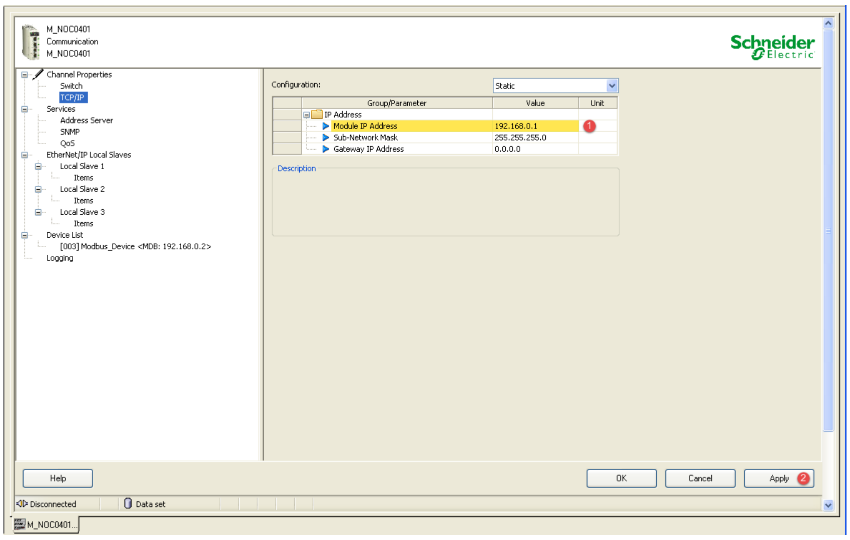

M_NOC0401
Communication
M_NOC0401

Source Address:
Source IP Address: 192.168.1.222
Sub-Network Mask: 255 . 255 . 255 . 0

EtherNet/IP Network Detection:
Begin detection range address: 192 . 168 . 1 . 1
End detection range address: 192 . 168 . 1 . 254

Modbus Network Detection:
Begin detection range address: 192 . 168 . 1 . 1
End detection range address: 192 . 168 . 1 . 254

Double-click on the communication module (<192.168.0.1> M_NOC0401.2). A window to the right of the navigation list will be opened. Set up the IP address of your host PC under **Channel properties** as shown above.


M_NOC0401
Communication
M_NOC0401

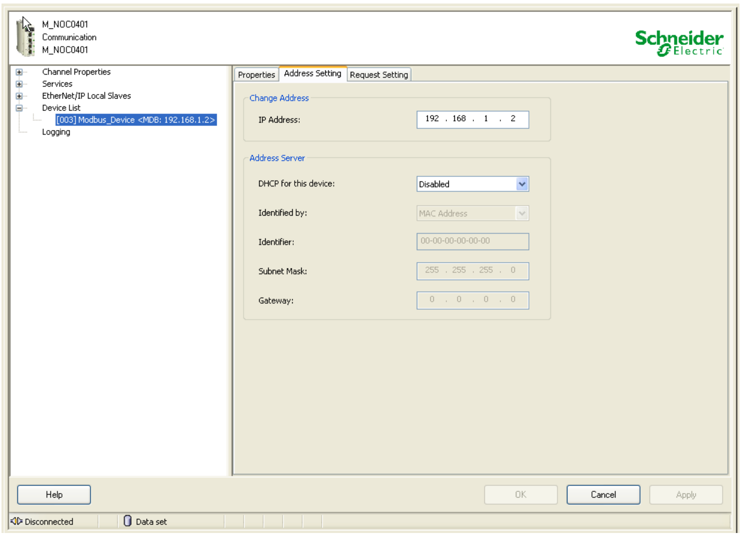
Configuration: Static

Group/Parameter	Value	Unit
IP Address		
Module IP Address	192.168.0.1	
Sub-Network Mask	255.255.255.0	
Gateway IP Address	0.0.0.0	

Description

In TCP/IP set the configuration to **static.** Change the IP address of the communication module M_NOC0401. In our case, set the IP address to 192.168.1.1.

Then under **Device List**, select your modbus device. In the tab **Address Setting**, change the IP address corresponding to the IP address of the Anybus CompactCom Modbus-TCP module.



Change Address

IP Address:	192 . 168 . 1 . 2

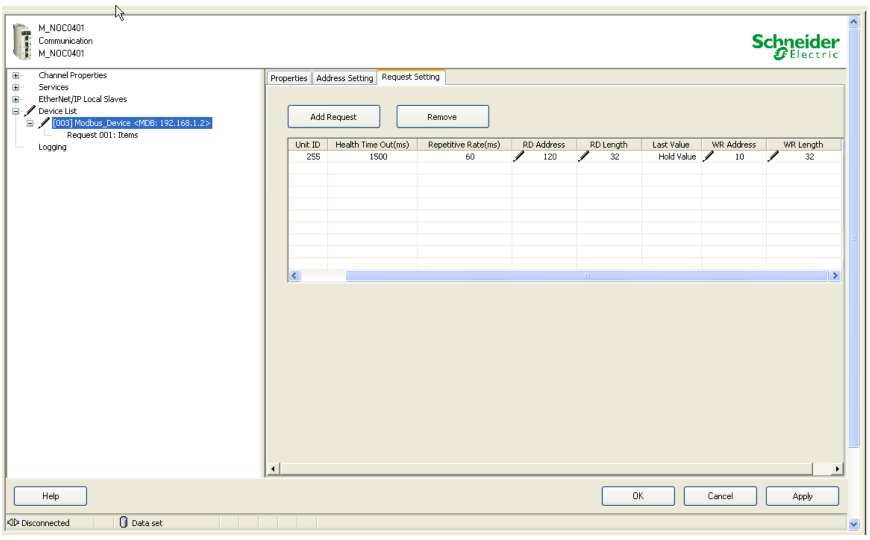
Address Server

DHCP for this device:	Disabled
Identified by:	MAC Address
Identifier:	00-00-00-00-00-00
Subnet Mask:	255 . 255 . 255 . 0
Gateway:	0 . 0 . 0 . 0

Move to the next tab **Request Setting** and click on **Add Request**. Default values will be filled out in the first row.

Then, set the new length of the I/O data in **RD length** and **WR length** .

RD address indicates the address of the PLC memory where from each device the information is read and stored and **WR address** indicates the address of PLC memory from where data is written into the slave device.



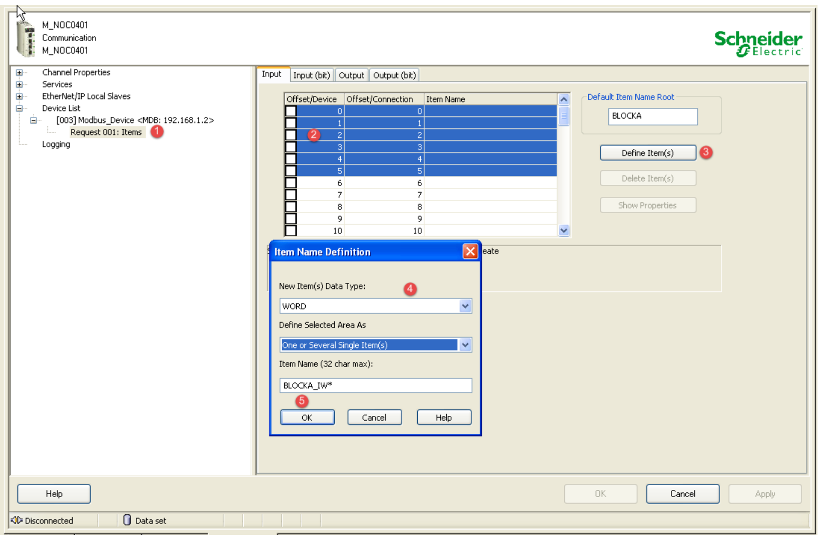
Unit ID	Health Time Out(ms)	Repetitive Rate(ms)	RD Address	RD Length	Last Value	WR Address	WR Length
255	1500	60	120	32	Hold Value	10	32

As example, set **RD length** to 32 words and **WR length** to 32 words as shown above… and click on apply.

Click on **Request 001: Items**. The elements are applied to the PLC as bytes, but the presentation of the data can be modified depending on how you want to access the data (e.g. as 16-bit registers).

In this example the first eight elements are presented as words at the outputs, leaving the rest as bytes.

To do this, select, with the mouse, the number of bytes you want to present as words then click on **Define Item(s)**. We can do the same for inputs. Confirm by clicking on apply.


M_NOC0401
Communication
M_NOC0401

Channel Properties
Services
EtherNet/IP Local Slaves
Device List
[003] Modbus_Device <MDB: 192.168.1.2>
Request 001: Items
Logging

Offset/Device	Offset/Connection	Item Name
0	0	
1	1	
2	2	
3	3	
4	4	
5	5	
6	6	
7	7	
8	8	
9	9	
10	10	

Default Item Name Root: BLOCKA

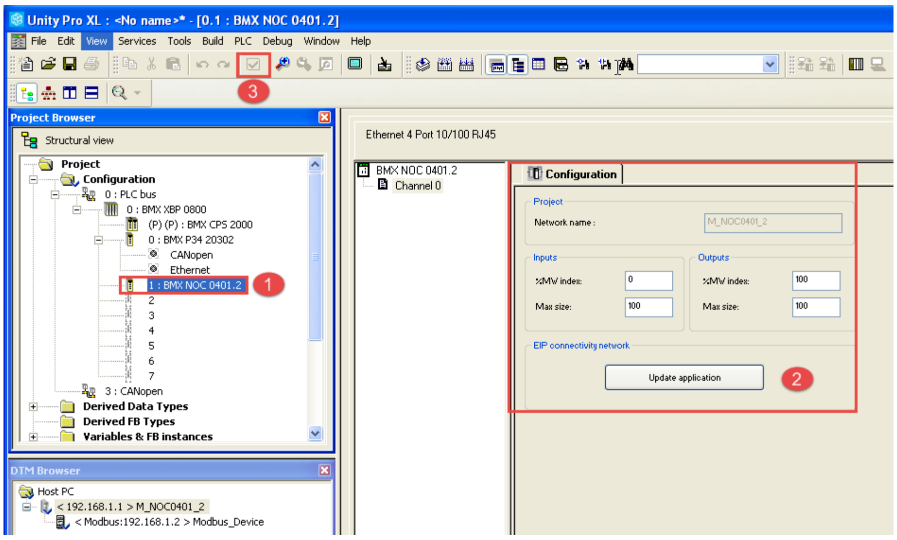
Item Name Definition

New Item(s) Data Type: WORD

Define Selected Area As: One or Several Single Item(s)

Item Name (32 char max): BLOCKA_IW*

Configure I/O word range of BMX NOC 0401.2. In the Project browser window double-click on BMX NOC 0401.2, then select the configuration tab. Type in the starting address and maximum number of 16 bits words dedicated for inputs/outputs as shown below. Validate the settings by clicking on the black check mark.


Unity Pro XL : <No name>* - [0.1 : BMX NOC 0401.2]

Project Browser

Structural view
Project
Configuration
0 : PLC bus
0 : BMX XBP 0800
(P) (P) : BMX CPS 2000
0 : BMX P34 20302
CANopen
Ethernet
1 : BMX NOC 0401.2
2
3
4
5
6
7
3 : CANopen
Derived Data Types
Derived FB Types
Variables & FB instances

DTM Browser
Host PC
< 192.168.1.1 > M_NOC0401_2
< Modbus:192.168.1.2 > Modbus_Device

Ethernet 4 Port 10/100 RJ45
BMX NOC 0401.2
Channel 0

Configuration

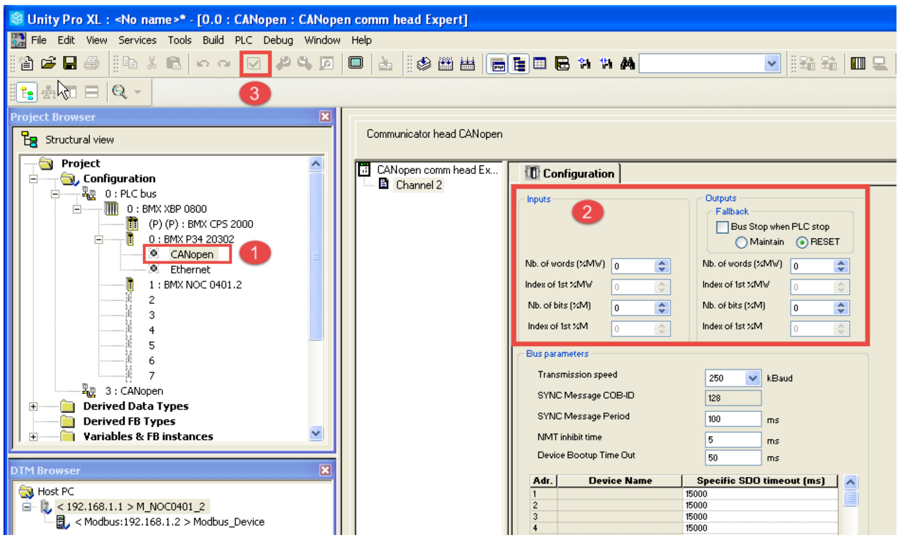
Project
Network name : M_NOC0401_2

Inputs
%MW index: 0
Max size: 100

Outputs
%MW index: 100
Max size: 100

EIP connectivity network

Before you build your project, the CANopen inputs/outputs values should be set to zero as described in the following picture. Then validated, built and saved.



Inputs

Nb. of words (%MW)	0
Index of 1st %MW	0
Nb. of bits (%M)	0
Index of 1st %M	0

Outputs

Fallback: Bus Stop when PLC stop [ ] — Maintain [✓ RESET]

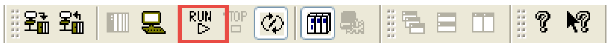
Nb. of words (%MW)	0
Index of 1st %MW	0
Nb. of bits (%M)	0
Index of 1st %M	0

Bus parameters

Transmission speed	250	kBaud
SYNC Message COB-ID	128	
SYNC Message Period	100	ms
NMT inhibit time	5	ms
Device Bootup Time Out	50	ms

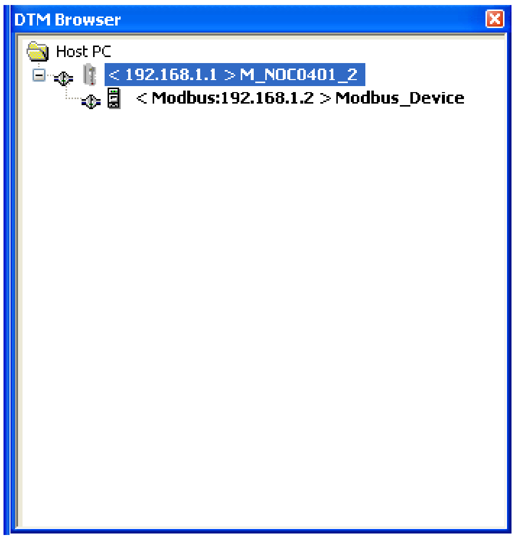
Adr.	Device Name	Specific SDO timeout (ms)
1		15000
2		15000
3		15000
4		15000

Download your project to the PLC. Set up the standard mode in the PLC menu. Then select *transfer project to the PLC* and click on transfer to finish the operation. In the menu toolbar, click on the **Run** icon.



The process data exchange can start between the PLC and the Anybus CompactCom device.

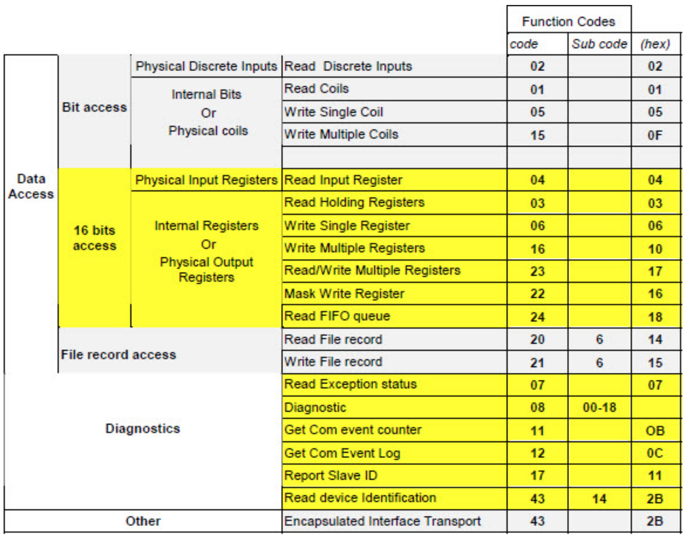
Select the DTM you want to connect to remote device (in this case, the CompactCom Modbus-TCP device). Click the right mouse button and select **Connect**. If the device name appears in bold text it is connected as shown below.


DTM Browser
Host PC
< 192.168.1.1 > M_NOC0401_2
< Modbus:192.168.1.2 > Modbus_Device

4.3 Diagnostics in Modbus-TCP

This section shows how Modbus-TCP handles diagnostic data.

Modbus-TCP provides only one way for reporting diagnosis messages. This is done through the acyclic data channel.

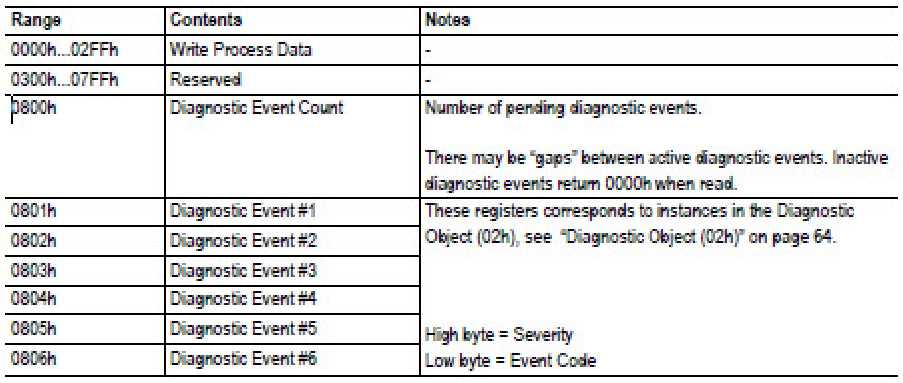


				Function Codes		
				code	Sub code	(hex)
Data Access	Bit access	Physical Discrete Inputs	Read Discrete Inputs	02		02
		Internal Bits Or Physical coils	Read Coils	01		01
			Write Single Coil	05		05
			Write Multiple Coils	15		0F
	16 bits access	Physical Input Registers	Read Input Register	04		04
		Internal Registers Or Physical Output Registers	Read Holding Registers	03		03
			Write Single Register	06		06
			Write Multiple Registers	16		10
			Read/Write Multiple Registers	23		17
			Mask Write Register	22		16
			Read FIFO queue	24		18
	File record access		Read File record	20	6	14
			Write File record	21	6	15
Diagnostics			Read Exception status	07		07
			Diagnostic	08	00-18	
			Get Com event counter	11		OB
			Get Com Event Log	12		0C
			Report Slave ID	17		11
			Read device Identification	43	14	2B
Other			Encapsulated Interface Transport	43		2B

In the Modbus protocol, function codes are used to access data. They determine whether data is to be read or written, and what kind of data is involved. The table above shows public function codes that give access to 16 bit registers.

The Modbus function code dedicated to diagnosis is 08. It is applicable **only** for serial line devices.

Therefore, the Anybus CompactCom 40 Modbus-TCP is devised to use function code **04** to read out specific Input Registers dedicated to diagnostics events.

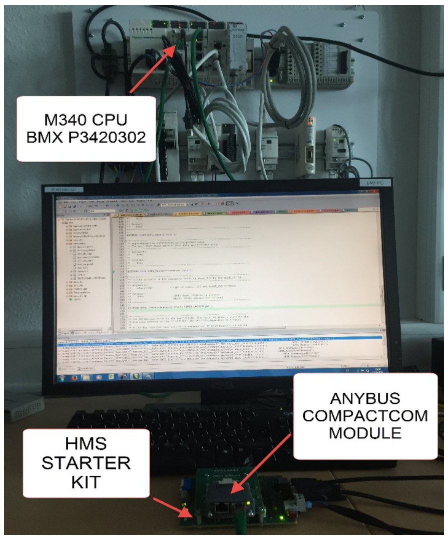


Range	Contents	Notes
0000h...02FFh	Write Process Data	-
0300h...07FFh	Reserved	-
0800h	Diagnostic Event Count	Number of pending diagnostic events. There may be "gaps" between active diagnostic events. Inactive diagnostic events return 0000h when read.
0801h	Diagnostic Event #1	These registers corresponds to instances in the Diagnostic Object (02h), see "Diagnostic Object (02h)" on page 64. High byte = Severity Low byte = Event Code
0802h	Diagnostic Event #2	
0803h	Diagnostic Event #3	
0804h	Diagnostic Event #4	
0805h	Diagnostic Event #5	
0806h	Diagnostic Event #6	

Extended diagnostic information is not supported by the Anybus CompactCom 40 Modbus-TCP since **network specific** information is not **implemented.**

5 Application Example

In this example, we are using the Anybus CompactCom 40 Starter Kit in which we integrate the Anybus CompactCom 40 device. We use the host application example code for the windows platform to simulate the host application. We also run the automation engineering tool Unity Pro XL by Schneider, for configuring the PLC. In this example, we employ Modicon 340 with -CPU BMX P34 20302- used as PLC-controller and -BMX NOC 0401.2- used as communication module. The engineering tool will also report the diagnostic events sent by the CompactCom device to the PLC.

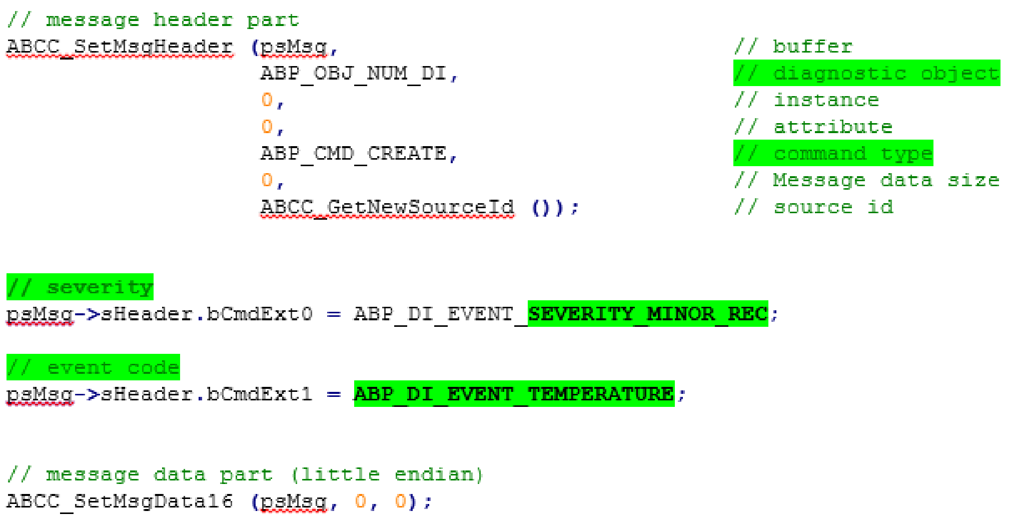


The following section will explain what has to be done from the application side to make the Anybus CompactCom 40 Modbus-TCP device send ***field device diagnosis*** to the network master.

5.1 Sample for Creating a Diagnostic Event

This sample is intended to show how to create a diagnostic event in the CompactCom using the Anybus CompactCom 40 host application example code. The example below shows the structure of a ***create (03h) event command message***, that generates a diagnostic event defined as *minor recoverable* for the event code *temperature*.

Section 6.1 will show how this event is displayed on Unity Pro XL.



```
// message header part
ABCC_SetMsgHeader (psMsg,                              // buffer
                   ABP_OBJ_NUM_DI,                     // diagnostic object
                   0,                                  // instance
                   0,                                  // attribute
                   ABP_CMD_CREATE,                     // command type
                   0,                                  // Message data size
                   ABCC_GetNewSourceId ());            // source id


// severity
psMsg->sHeader.bCmdExt0 = ABP_DI_EVENT_SEVERITY_MINOR_REC;

// event code
psMsg->sHeader.bCmdExt1 = ABP_DI_EVENT_TEMPERATURE;


// message data part (little endian)
ABCC_SetMsgData16 (psMsg, 0, 0);
```

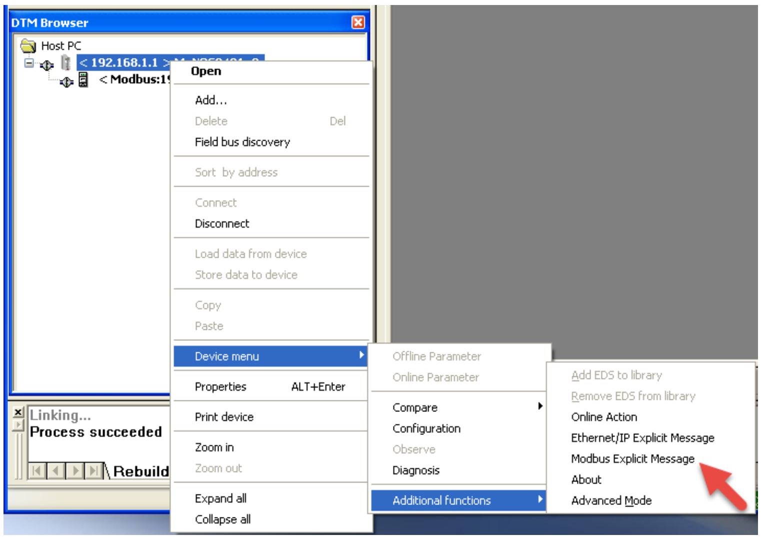
The *create (03h)* event command above has created a diagnostic event including the severity code (***minor recoverable (0x00)***) and the event code (***temperature (0x40)***). There is no data included in the message in this case.

The following section will show the appropriate diagnostic message for each diagnostic event when a diagnostic event is created and deleted from the Modbus-TCP slave device.

6 Readout of Diagnostics in Unity Pro XL

This section shows how Unity Pro XL handles diagnostic data.

Start your project in Unity Pro XL with all the settings complete, including your Anybus CompactCom 40 Modbus-TCP device. Since the Modbus protocol only allows acyclic access to registers for reading diagnostic messages, to be able to read out diagnostic information from a Modbus-TCP slave device we use the Modbus Explicit Message window. This is done to send an explicit message from the Unity Pro XL to a Modbus-TCP device -Here the CompactCom 40 Modbus-TCP module.



Linking...
Process succeeded

In the DTM browser, select the communication module and right click. In the dropdown menu, select **Device menu** → Additional functions → **Modbus Explicit Message** as shown below.

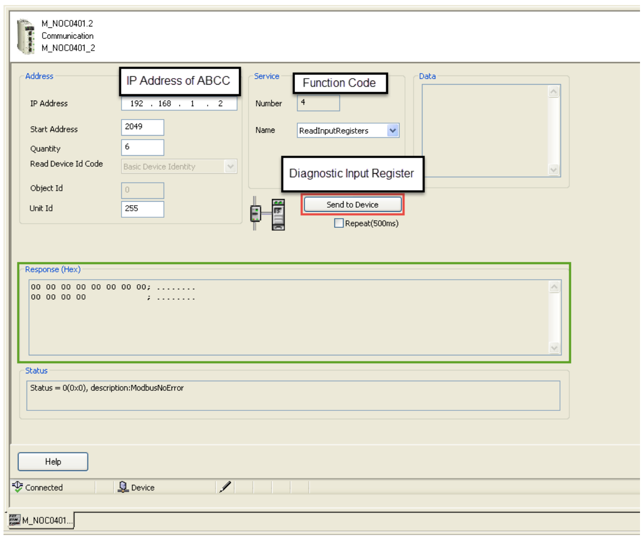
The Modbus Explicit Message window will be opened.

Before performing explicit messaging, connect the DTM for the communication module to the CompactCom 40 device.

To do this, select the module node in the DTM Browser, then select **Edit** → **Connect**.

The Modbus-TCP Explicit Message window, below, presents an example of both the configuration of a Modbus-TCP explicit message, and the response. In this example, the explicit message is used to read the six input registers in the CompactCom Modbus-TCP device dedicated for diagnostic events, starting at offset 2049.

To configure the explicit message means, here, to type the IP address of the CompactCom device, select the function code **04** including the action to perform **ReadInputRegisters**, set the number of registers to read to the value 6 and the start address to 2049 (0801h). After the configuration of the explicit message is done, click on **Send to Device** in the red frame. The diagnostic information will be displayed in the response area, in hexadecimal format, in the green frame.



Address
IP Address 192 . 168 . 1 . 2
Start Address 2049
Quantity 6
Read Device Id Code Basic Device Identity
Object Id 0
Unit Id 255

Service
Number 4
Name ReadInputRegisters

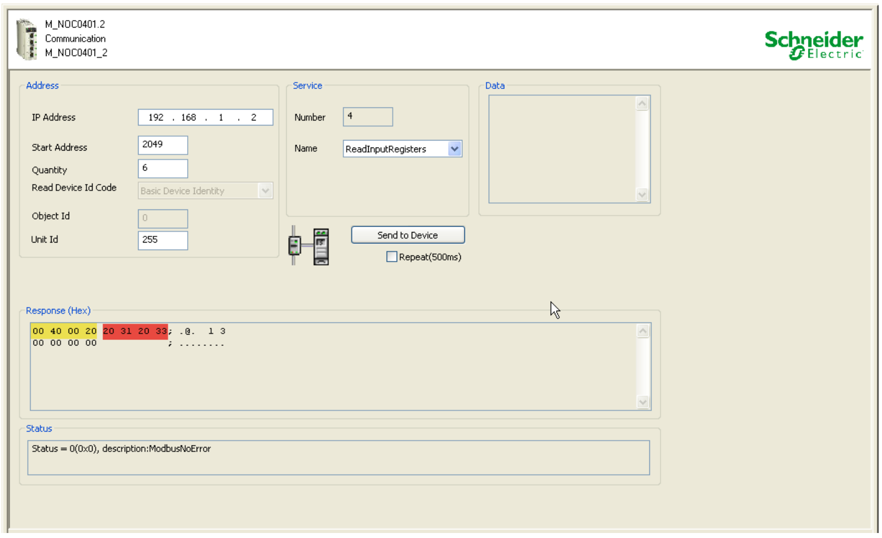
Repeat(500ms)

Response (Hex)
```
00 00 00 00 00 00 00 00; ........
00 00 00 00            ; ........
```

Status
Status = 0(0x0), description:ModbusNoError

6.1 Severity: Minor Recoverable and Major Recoverable

The picture below presents two active diagnostic events defined as **minor recoverable (00h)** highlighted in yellow and two other active diagnostic events highlighted in red defined **as major recoverable (20h).** In this case four input registers are read. Each input register contains 2 bytes. The first byte is the high byte returning the severity code and the second byte is the low byte associated with the event code.



Address
IP Address 192 . 168 . 1 . 2
Start Address 2049
Quantity 6
Read Device Id Code Basic Device Identity
Object Id 0
Unit Id 255

Service
Number 4
Name ReadInputRegisters

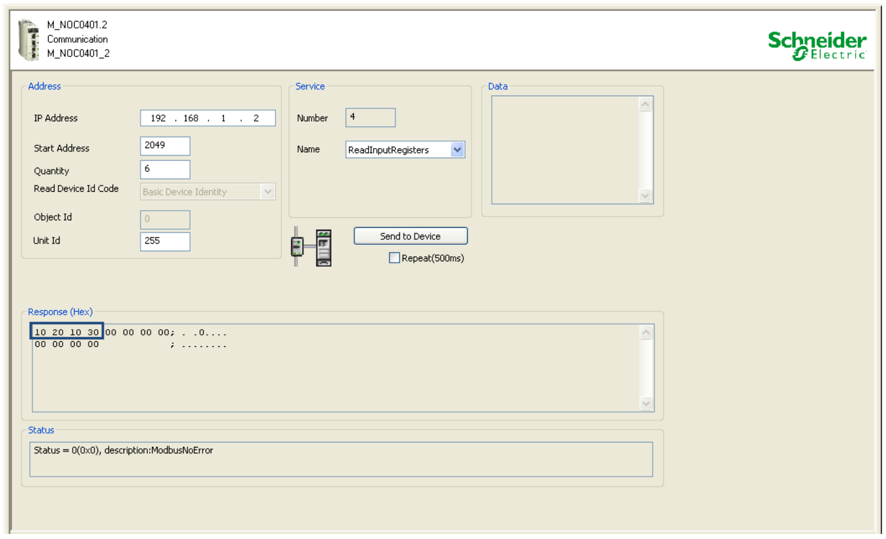
Repeat(500ms)

Response (Hex)
```
00 40 00 20 20 31 20 33; .@.  1 3
00 00 00 00            ; ........
```

Status
Status = 0(0x0), description:ModbusNoError

6.2 Severity: Minor Unrecoverable

The picture below shows two active diagnostic events defined as *minor unrecoverable*. From the response area, two input registers can be read out. Each input register provides the information about the severity of the event and the nature of the event. In the first input register the high byte (0x10) represents the severity code (*minor unrecoverable event*) and the low byte (20h, “Current”) the nature of the event which represents the event code. This diagnostic event cannot be deleted. A reset of the CompactCom device will delete this diagnostic event.


M_NOC0401.2
Communication
M_NOC0401_2

Address
IP Address: 192 . 168 . 1 . 2
Start Address: 2049
Quantity: 6
Read Device Id Code: Basic Device Identity
Object Id: 0
Unit Id: 255

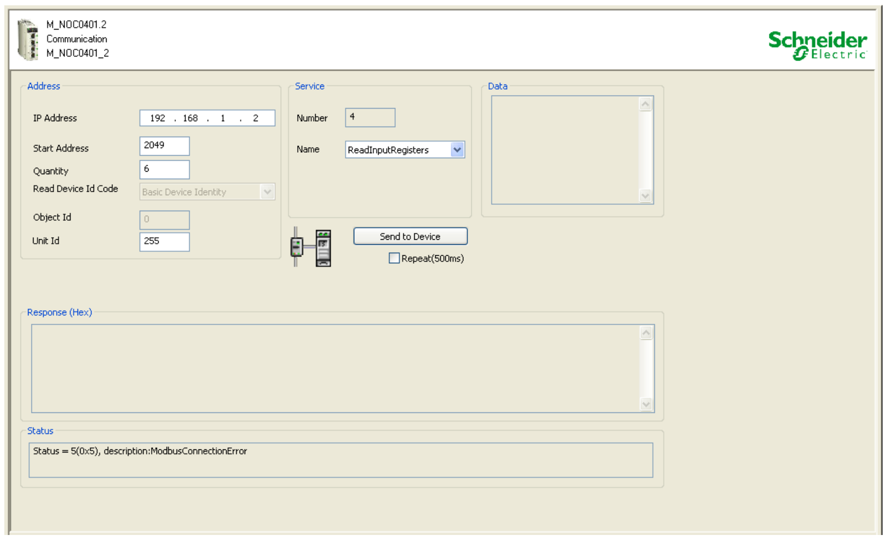
Service
Number: 4
Name: ReadInputRegisters

Response (Hex)
```
10 20 10 30 00 00 00 00; . .0....
00 00 00 00            ; ........
```

Status
Status = 0(0x0), description:ModbusNoError

6.3 Severity: Major Unrecoverable

The creation of a *major unrecoverable* diagnostic event causes the Anybus CompactCom 40 device to enter EXCEPTION state. This results in a disconnection of the device from the network. It is not possible to report any diagnostic. This is confirmed by the picture below.


M_NOC0401.2
Communication
M_NOC0401_2

Address
IP Address: 192 . 168 . 1 . 2
Start Address: 2049
Quantity: 6
Read Device Id Code: Basic Device Identity
Object Id: 0
Unit Id: 255

Service
Number: 4
Name: ReadInputRegisters

Response (Hex)

Status
Status = 5(0x5), description:ModbusConnectionError