



**Deuschmann**

*your ticket to all buses*

**Instruction Manual  
Universal Fieldbus-Gateway  
UNIGATE® CX**





<b>1</b>	<b>Information on CE marking of the module</b>	<b>8</b>
1.1	EU Directive EMC	8
1.2	Scope of application	8
1.3	Note installation guidelines	8
1.4	Installation of the unit	8
1.5	Working on switch cabinets	8
<b>2</b>	<b>Information for the machine manufacturer</b>	<b>9</b>
2.1	Introduction	9
2.2	EU Machinery Directive	9
<b>3</b>	<b>Introduction</b>	<b>10</b>
<b>4</b>	<b>Hardware-design</b>	<b>11</b>
4.1	Available types	11
4.2	Hardware structure	12
4.3	UNIGATE® CX block diagram	13
4.4	UNIGATE® CX design	14
4.5	Operation modes	14
4.5.1	RUN-operation (data exchange mode)	14
4.5.2	Test mode	14
<b>5</b>	<b>Script-Programming</b>	<b>16</b>
5.1	What is a script?	16
5.1.1	Function of the scripts in UNIGATE® CL	16
<b>6</b>	<b>Protocol Configuration</b>	<b>17</b>
6.1	Overview of configurable buses	17
6.2	Configuration options of the available protocols	18
6.2.1	Configuration over view without Ethernet resp. ModbusTCP interface	18
6.2.2	Configuration overview with Ethernet resp. Modbus TCP interface	19
6.2.3	Configuration overview with two Ethernet resp. Modbus TCP interfaces	20
<b>7</b>	<b>Implemented protocols in UNIGATE® CX</b>	<b>21</b>
7.1	Protocol: Transparent	21
7.1.1	Data structure	21
7.2	Protocol: Universal 232	21
7.2.1	Data structure	21
7.2.2	Fieldbus parameters	22
7.2.3	RS232 parameter table	22
7.2.3.1	Start character (232 Start character)	22
7.2.3.2	Length 232 (232 Length)	22
7.2.3.3	Data area	22
7.2.3.4	Checksum	22
7.2.3.5	End character (232 End character)	22
7.2.4	Communication sequence	22
7.3	Protocol "CX (Pseudo)"	23
7.4	Protocol: 3964(R)	23
7.4.1	Data structure 3964R	23
7.4.2	Protocol definitions	23
7.4.3	Data communication	24

7.4.3.1	Initiation of data communication by the low-priority user . . . . .	24
7.4.3.2	Conflicts . . . . .	24
7.4.3.3	Timeout times . . . . .	24
7.4.3.4	Retries . . . . .	24
7.4.3.5	Initiation of data communication by the high-priority user . . . . .	24
7.4.4	Protocol type 3964 . . . . .	24
7.5	Protocol: MODBUS-RTU . . . . .	24
7.5.1	Notes . . . . .	24
7.5.2	UNIGATE® as MODBUS-Master . . . . .	24
7.5.2.1	Preparation . . . . .	24
7.5.2.2	Data structure . . . . .	25
7.5.2.3	Communication sequence . . . . .	25
7.5.3	UNIGATE® as MODBUS-Slave . . . . .	26
7.5.3.1	Preparation . . . . .	26
7.5.3.2	Data structure . . . . .	26
7.5.3.3	Communication sequence . . . . .	26
7.6	Protocol Modbus ASCII Master/Slave . . . . .	26
7.7	Protocol „Universal Modbus RTU Slave“ . . . . .	26
7.7.1	Data structure on the fieldbus side e.g.: PROFIBUS . . . . .	27
7.7.1.1	Example: FC1 + FC2 . . . . .	27
7.7.1.2	Example: FC3 (Read Holding Register) + FC4 (Read Input Register) . . . . .	28
7.7.1.3	Example: Write Single Coil FC5 . . . . .	29
7.7.1.4	Example: Write Single Register FC6 . . . . .	30
7.7.1.5	Example: Force multiple coils FC 15 . . . . .	31
7.7.1.6	Example: Preset multiple register FC16 . . . . .	31
7.8	Protocol „Universal Modbus RTU Master“ . . . . .	32
7.8.1	Data structure Fieldbus side (e.g. PROFIBUS): . . . . .	32
7.8.2	Data structure Application side: . . . . .	32
7.8.3	Configuration: via Wingate since wcf Datei Version 396 . . . . .	33
7.8.3.1	Example: Read coil status FC1 . . . . .	34
7.8.3.2	Example: Read input status FC2 . . . . .	35
7.8.3.3	Example: Read multiple register FC3 . . . . .	36
7.8.3.4	Example: Read input registers FC4 . . . . .	37
7.8.3.5	Example: Force single coil FC5 . . . . .	37
7.8.3.6	Example: Preset single register FC6 . . . . .	38
7.8.3.7	Example: Force multiple coils FC15 . . . . .	38
7.8.3.8	Example: Preset multiple register FC16 . . . . .	39
7.9	Protocol „Universal Modbus ASCII Master/Slave“ . . . . .	40
7.10	Protocol Modbus TCP client encapsulation . . . . .	41
7.10.1	Function . . . . .	41
7.10.1.1	UNIGATE® CL: . . . . .	41
7.10.1.2	UNIGATE® CX: . . . . .	42
<b>8</b>	<b>Optional bus parameter . . . . .</b>	<b>43</b>
8.1	The trigger byte . . . . .	43
8.2	The length byte . . . . .	43
8.3	Swap word . . . . .	43



<b>9</b>	<b>Fieldbus parameters / Ethernet parameters</b>	<b>44</b>
<b>10</b>	<b>Hardware connections, switches and LEDs</b>	<b>45</b>
10.1	Device label	45
10.2	Connectors	45
10.2.1	Connector supply voltage and DEBUG-interface 1	45
10.2.2	Connector output voltage and DEBUG-interface 2	46
10.3	Power supply	46
10.4	LEDs, switches, bus connection	46
10.5	UNIGATE® CX connection cable	46
<b>11</b>	<b>Installation guidelines</b>	<b>47</b>
11.1	Installation of the module	47
11.1.1	Mounting	47
11.1.2	Removal	47
11.2	Wiring	47
11.2.1	Connection systems	47
11.2.1.1	Power supply	47
11.2.1.2	Equipotential bonding connection	47
11.2.2	Communication interface	48
11.2.2.1	CANopen Slave, CANopen Master / CAN Layer 2	48
11.2.2.2	DeviceNet	48
11.2.2.3	EtherCAT	48
11.2.2.4	EtherNet/IP	48
11.2.2.5	Ethernet	48
11.2.2.6	LONWorks	49
11.2.2.7	MPI	49
11.2.2.8	PROFIBUS DP	49
11.2.2.9	PROFINET-IO	49
11.2.3	Line routing, shield and measures to combat interference voltage	49
11.2.4	General information on line routing	50
11.2.4.1	Shielding of lines	50
<b>12</b>	<b>Technical data</b>	<b>52</b>
12.1	Device data	52
<b>13</b>	<b>Commissioning guide</b>	<b>53</b>
13.1	Note	53
13.2	Components	53
13.3	Installation	53
13.4	Dimensional drawings	53
13.4.1	UNIGATE® CX (all versions without CANopen Slave, CANopen Master, CAN Layer 2, MPI or PROFIBUS DP)	53
13.4.2	UNIGATE® CX (all versions with CANopen Slave, CANopen Master, CAN Layer 2, MPI or PROFIBUS DP)	54
13.5	Commissioning	54
13.6	Fieldbus connection	54
13.7	Connecting the supply voltage	54
13.8	Shield connection	54

<b>14</b>	<b>Service Interface (RS232)</b>	<b>55</b>
14.1	Service interface (RS232) - Connection	55
14.2	Service interface (RS232) – Access	56
<b>15</b>	<b>Servicing</b>	<b>59</b>
15.1	Returning a device	59
15.2	Downloading PC software	59
<b>16</b>	<b>Annex</b>	<b>60</b>
16.1	Explanations of the abbreviations	60
16.2	Hexadecimal table	61

#### Disclaimer of liability

We have checked the contents of the document for conformity with the hardware and software described. Nevertheless, we are unable to preclude the possibility of deviations so that we are unable to assume warranty for full compliance. The information given in the publication is, however, reviewed regularly. Necessary amendments are incorporated in the following editions. We would be pleased to receive any improvement proposals which you may have.

#### Copyright

Copyright (C) Deutschmann Automation GmbH & Co. KG 1997 - 2021. All rights reserved. This document may not be passed on nor duplicated, nor may its contents be used or disclosed unless expressly permitted. Violations of this clause will necessarily lead to compensation in damages. All rights reserved, in particular rights of granting of patents or registration of utility-model patents.

# 1 Information on CE marking of the module

## 1.1 EU Directive EMC

The following applies to the module described in this User Manual:

Products which bear the CE mark comply with the requirements of EU Directive „Electromagnetic Compatibility“ and the harmonised European Standards (EN) listed therein.

The EU Declarations of Conformity are available at the following location for perusal by the responsible authorities in accordance with the EU Directive, Article 10:

Deutschmann Automation GmbH & Co. KG, Carl-Zeiss-Str. 8, 65520 Bad Camberg, Germany

## 1.2 Scope of application

The modules are designed for use in the industrial sector and comply with the following requirements

Scope of application	Requirement applicable to	
	Emitted interference	Interference immunity
Industry	EN 55011, cl. A (2007)	EN 61000-6-2 (2005)

## 1.3 Note installation guidelines

The module complies with the requirements if you

1. comply with the installation guidelines described in the User Manual when installing and operating the module.
2. also follow the rules below on installation of the equipment and on working on switch cabinets.

## 1.4 Installation of the unit

Modules must be installed in electrical equipment rooms/areas or in enclosed housings (e.g. switch boxes made of metal or plastic). Moreover, you must earth the unit and the switch box (metal box) or at least the top-hat rail (plastic box) onto which the module has been snapped.

## 1.5 Working on switch cabinets

In order to protect the modules against static electrical discharge, the personnel must discharge themselves electrostatically before opening switch cabinets or switch boxes.

## **2 Information for the machine manufacturer**

### **2.1 Introduction**

The UNIGATE® module does not constitute a machine as defined by the EU "Machinery" Directive. Consequently, the module does not have a Declaration of Conformity in relation to the EU Machinery Directive.

### **2.2 EU Machinery Directive**

The EU Machinery Directive stipulates the requirements applicable to a machine. The term "machine" is taken to mean a totality of connected parts or fixtures (see also EN 292-1, Paragraph 3.1).

The module is a part of the electrical equipment of the machine and must thus be included by the machine manufacturer in the Declaration of Conformity process.

### 3 Introduction

In the field of automatic control many different Fieldbuses and Industrial Ethernet became established worldwide. Again and again the task of interconnecting these incompatible networks comes up. The UNIGATE® CX-series was created exactly for that task. The series contains Fieldbus-Slave as well as Industrial Ethernet versions.

The UNIGATE® CX is designed as DIN-rail module and contains the selected Fieldbuses or Ethernet in the mechanical variant carried out in the respective standard. Internally the product is realized by using two UNIGATE® CL-modules. By this modular structure all Fieldbus- and Ethernet-versions can be supplied, provided that the respective CL-modules are available. The number of available versions is growing steadily by the continuous development of new CL-modules e. g. in the Industrial Ethernet field.

With the UNIGATE® CX series, the data exchange can be realized via the configuration of the implemented protocols or via the programming of a script. Another possibility is the combination of implemented protocol and a script. The implemented protocols can be configured with the configuration software WINGATE.

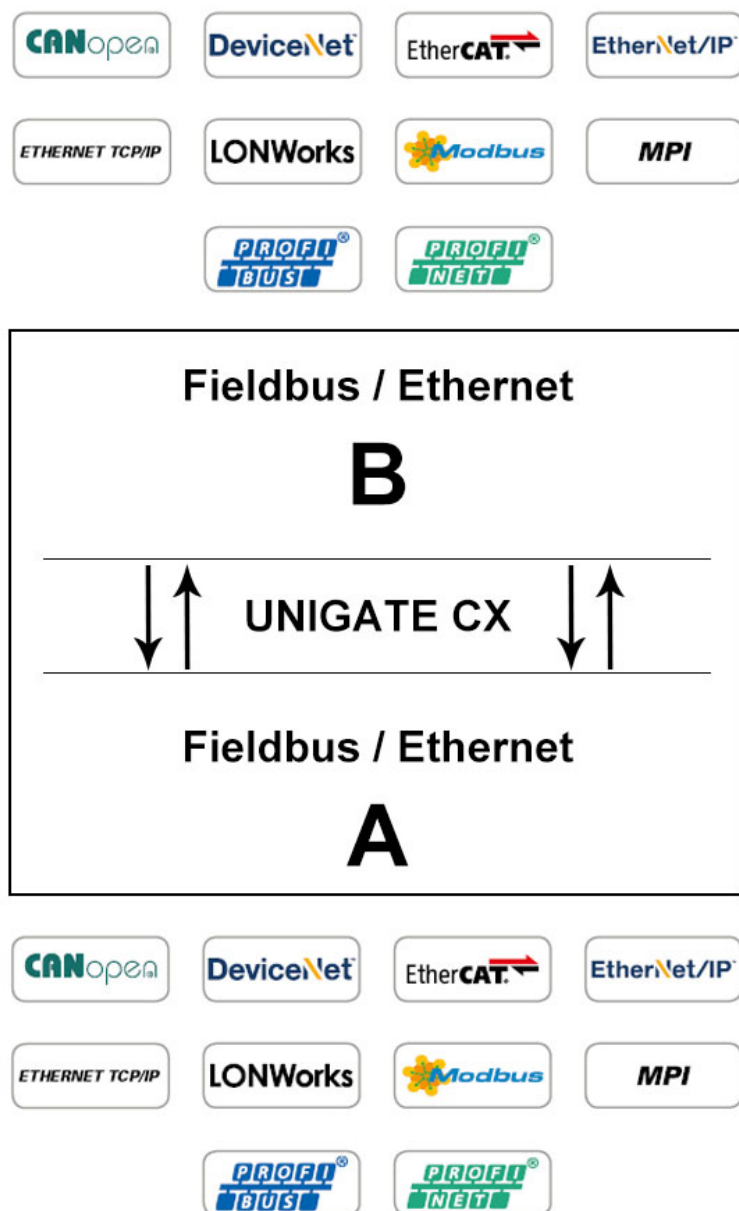
Scripts can be programmed with the software Protocol Developer.

## 4 Hardware-design

The UNIGATE<sup>®</sup> CX acts as gateway between two bus- respectively Ethernet systems. Both bus connections are usually realized as slave. Excluded are CANopen (master) as well as Ethernet or ModbusTCP (client). Each bus can be operated by a standard-compliant master.

### 4.1 Available types

UNIGATE<sup>®</sup> CX has a modular structure which makes any desired combination of fieldbuses possible. The following device types can be realized:



Fieldbus/Ethernet B											
Fieldbus/Ethernet A		CANopen (Slave), CANopen (Master), CAN Layer 2	DeviceNet	EtherCAT	Ethernet	EtherNet/IP	LONWorks	Modbus TCP	MPI	Profibus	Profinet
	CANopen (Slave), CANopen (Master), CAN Layer 2	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	DeviceNet	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	EtherCAT	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Ethernet	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	EtherNet/IP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	LONWorks	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Modbus TCP	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	MPI	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Profibus	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Profinet	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

From the above table you can see that all types can be combined, such as DeviceNet to DeviceNet in order to interconnect two autonomous DeviceNet-networks.

## 4.2 Hardware structure

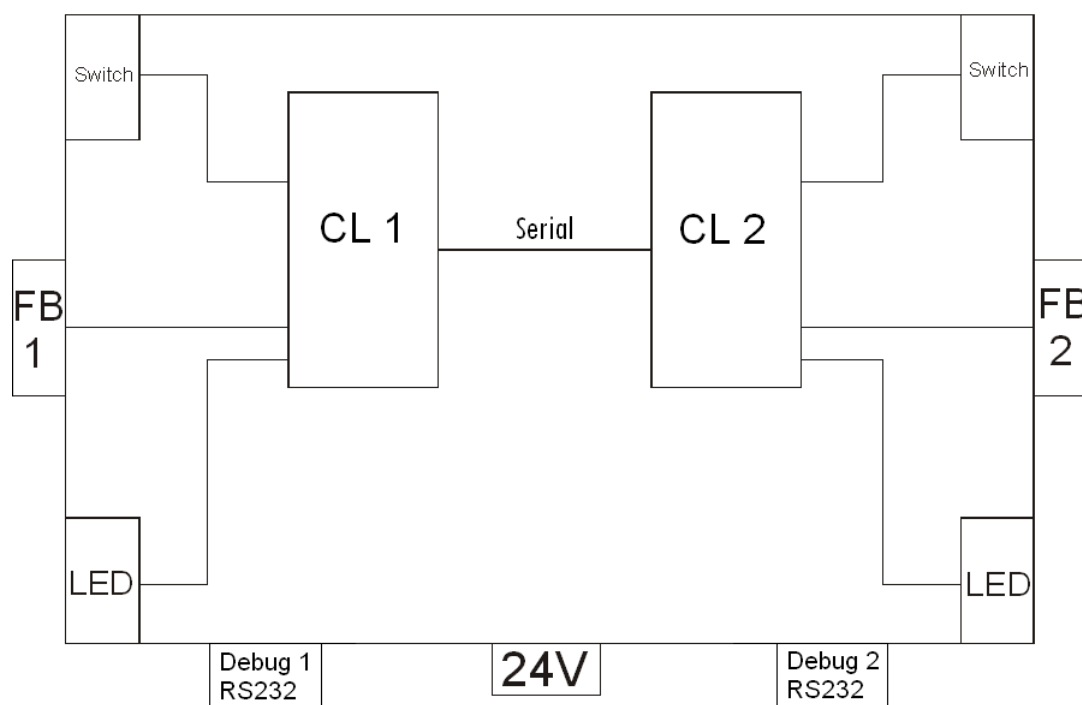
In order to have the possibility to interconnect different fieldbuses UNIGATE® CX consists of single components that can be combined with each other. These components are in detail:

- UNIGATE® CL 1 for fieldbus side A
- UNIGATE® CL 2 for fieldbus side B

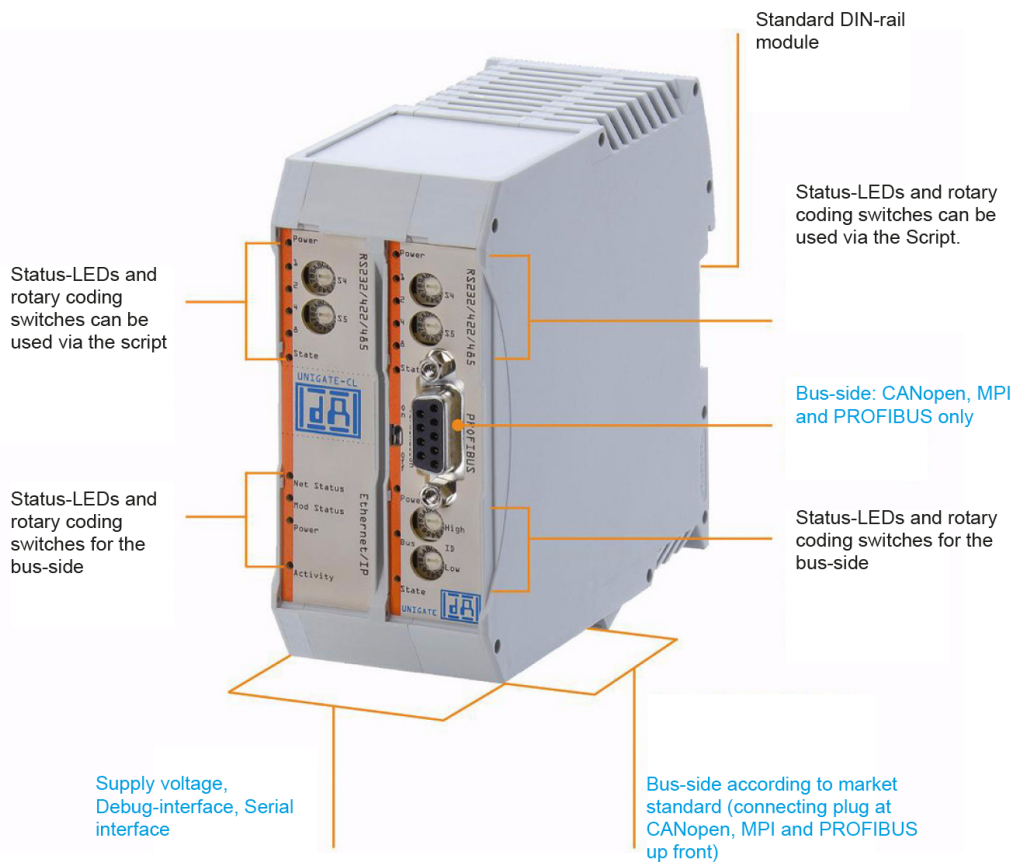
The exact internal structure is not relevant to the order since only the buses that are to be inter-connected have to be mentioned and you will receive a complete device.



### 4.3 UNIGATE® CX block diagram



## 4.4 UNIGATE® CX design



## 4.5 Operation modes

### 4.5.1 RUN-operation (data exchange mode)

In the ex-works condition the device is set to RUN-operation.

In RUN-operation it is possible to use the Debug-interfaces of the UNIGATE® CLs.

With these interfaces it is possible to execute a Script step-by-step and in a normal case they are only necessary for developing a Script. Furthermore the device can be configured through these interfaces. To do so, the software WINGATE is used ("Extras" -> "Upload\_Config\_Debug" or "Download\_Config\_Debug")

The Debug-interfaces are connected with X1 pin 3 + 4 or X2 pin 3 + 4.

Further information on how to use the Debug-interface can be found in the instruction manual UNIGATE® CL for the respective Fieldbus.

### 4.5.2 Test mode

#### Setting of the test mode

The test mode is set by bringing the switches S4 and S5 in position "E". All other switches will not be taken into consideration for the setting of the test mode. Now the Gateway has to be restarted with these settings (by a short disconnection from the power supply).

In the test mode the Gateway always operates with the settings 9600 baud, no parity, 8 databits and 1 stopbit.

The test mode may be helpful to integrate the Gateway in the relevant environment.

**Mode of operation of the test mode**

After the restart in the test mode the Gateway will be sending the values 0-15 in hexadecimal representation ("0".."F") in ASCII-coding on the serial side (Debug-interface) every second. Simultaneously the same values are issued binary on the fieldbus-interface.

In this mode the State-LED on the RS-side will be flashing red, the "Error No/Select ID" LEDs will be displaying the value in a binary way, that is issued that moment. Additionally each character that is received at one of the interfaces will also be output at the same interface as a local echo. On the fieldbus-side only the first byte will be used for the local echo, that means on receiving as well as on transmitting only the first byte of the bus data is looked at, the other bus data do not change compared to the last data.

## **5 Script-Programming**

### **5.1 What is a script?**

A script is a sequence of commands, that are executed in that exact order. Because of the fact that also mechanisms are given that control the program flow in the script it is also possible to assemble more complex processes from these simple commands.

The script is memory-oriented. It means that all variables always refer to one memory area. While developing a script you do not have to take care of the memory management though. The Protocol Developer takes on this responsibility for you.

#### **5.1.1 Function of the scripts in UNIGATE® CL**

The scripts handle the data exchange between the two UNIGATE® CLs. Here the data of fieldbus 1 is edited in the CL 1 and transmitted to the CL 2 via an RS-connection. On account of the field-buses' complexity and the multitude of resulting versions we are not going into details at this point. For further questions concerning scripts, please contact us directly.

## 6 Protocol Configuration

The UNIGATE® CX is delivered with the script "Universalscript Deutschmann". The delivery state is with transparent data exchange, so only the fieldbus-specific parameters must be configured. The fieldbus-specific parameters include, for example, data exchange ( byte), Fieldbus length byte (Chapter 7.11, The length byte), Swap word (Chapter 7.12, Swap word) and the settings for the IP address for the fieldbus variants.

The configuration takes place in the data exchange mode with the software WINGATE (Tools -> "Upload\_Config\_Debug" or "Download\_Config\_Debug").

Further configuration options can be found in the following tables.

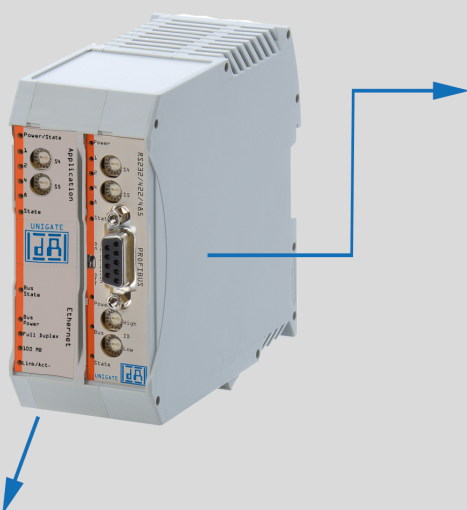
### 6.1 Overview of configurable buses

	Configuration	Programming (Script)
CANopen (Slave)	✓	✓
CANopen (Master)	✓	✓
CAN Layer 2	✓	✓
DeviceNet	✓	✓
EtherCAT	✓	✓
Ethernet	✓	✓
EtherNet/IP	✓	✓
LONWorks	✗	✓
ModbusTCP	✓	✓
MPI	✓	✓
PROFIBUS	✓	✓
PROFINET	✓	✓

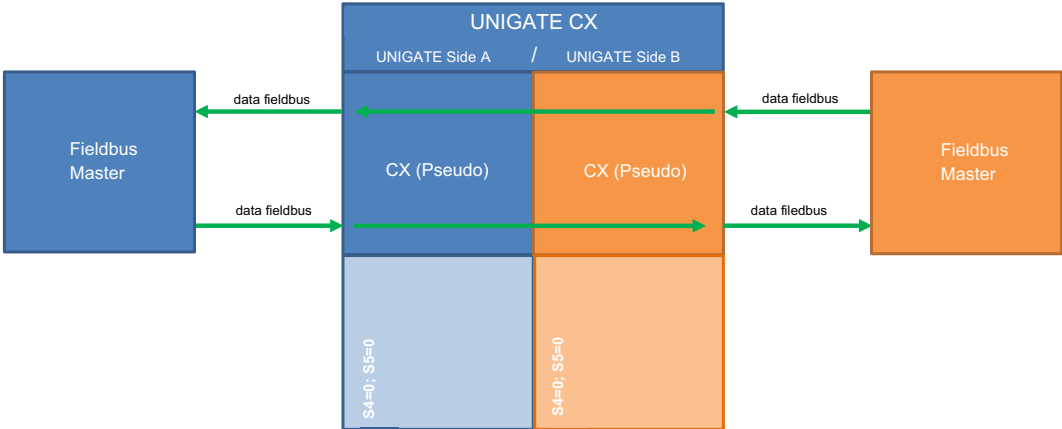
6.2 Configuration options of the available protocols

The configuration options of the protocols are highlighted with green checkmarks in the tables. For each application example, you will find a diagram for graphical illustration.

6.2.1 Configuration over view without Ethernet resp. ModbusTCP interface

		Fieldbus or Industrial Ethernet												
		UNIGATE Side A												
		Available Protocols												
		Transparent	Universal 232	3964(R) low prior	CX(Pseudo)	Modbus RTU Master	Modbus RTU Slave	Modbus ASCII Master	Modbus ASCII Slave	Universal Modbus RTU Master	Universal Modbus RTU Slave	Universal Modbus ASCII Master	Universal Modbus ASCII Slave	
Fieldbus or Industrial Ethernet	UNIGATE Side B	Available Protocols	Transparent	Universal 232	3964(R) high prior	CX(Pseudo)	Modbus RTU Master	Modbus RTU Slave	Modbus ASCII Master	Modbus ASCII Slave	Universal Modbus RTU Master	Universal Modbus RTU Slave	Universal Modbus ASCII Master	Universal Modbus ASCII Slave
			✓	✓		✓								
			✓	✓		✓								
					✓									
			✓	✓		✓								
							✓	✓			✓			
										✓				
										✓			✓	
												✓		
													✓	
									✓					✓
										✓				

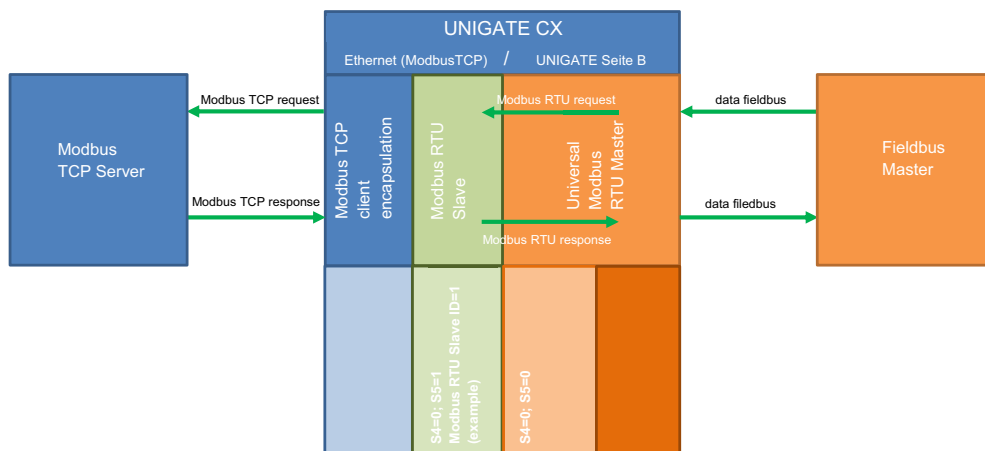
Application Example



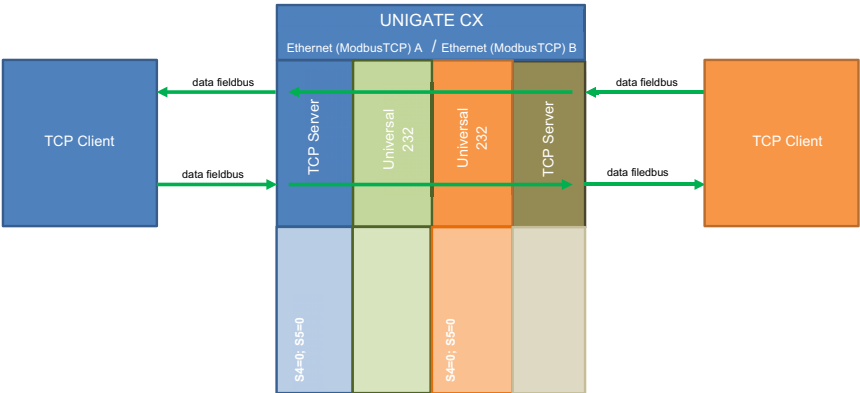
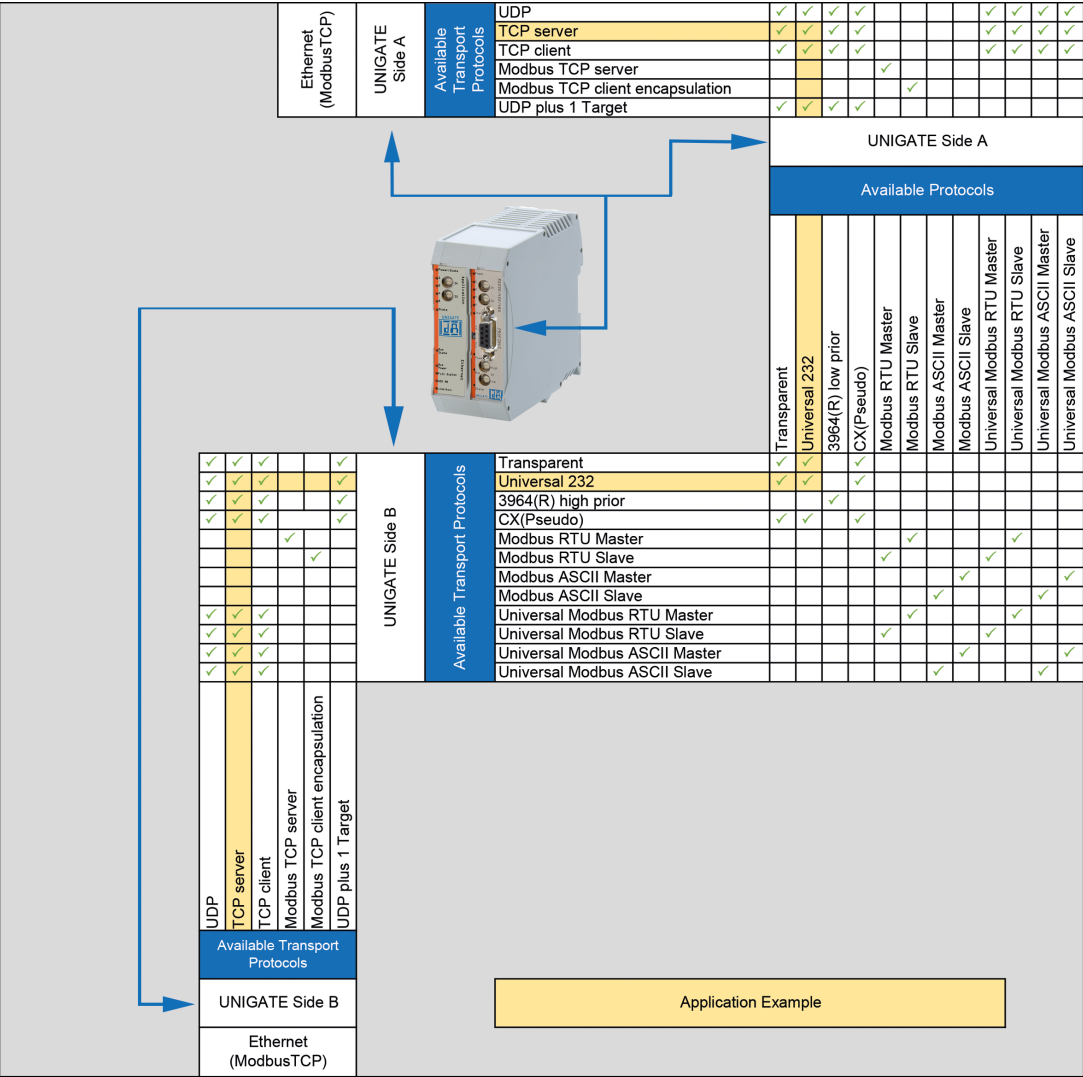
### 6.2.2 Configuration overview with Ethernet resp. Modbus TCP interface

Ethernet (Modbus TCP)	UNIGATE Side A	Available Transport Protocols	UDP																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																																				</
--------------------------	-------------------	-------------------------------------	-----	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--	----

#### Application Example



6.2.3 Configuration overview with two Ethernet resp. Modbus TCP interfaces





## 7 Implemented protocols in UNIGATE® CX

UNIGATE® CX is supplied with the Script "Universal Script Deutschmann". The configuration of the protocols is carried out in the configuration mode with the software WINGATE. See "Instructions UNIGATE® CL - Configuration with WINGATE". The PDF can also be found on our website under Support/Downloads/Manuals.



### Attention:

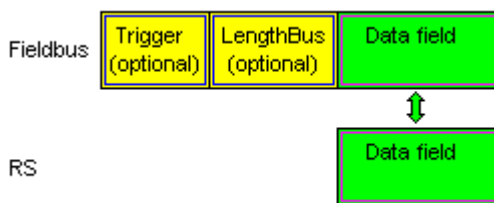
If a Reset Device is carried out it is possible (depending on the firmware version of the UNIGATE®) that the "Universal Script" will get lost and must be played in again.

The Script can be found on the Deutschmann Support-DVD in the folder \Software\ProtocolDeveloper\Example\Universal\.

### 7.1 Protocol: Transparent

The data is transferred bidirectional from the UNIGATE®.

#### 7.1.1 Data structure



On the RS-entry side the timeout time of 2 ms is firmly set. If no more data is received within the timeout period, then the data that has been received so far is transferred to the bus.

If less data is received through Rx then configured by the GSD-file (I/O-length), then the rest is complemented with ZERO.

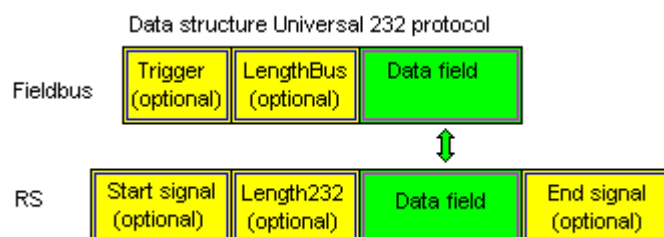
Too much data received will be cut off.

### 7.2 Protocol: Universal 232



The protocol designation "Universal 232" and the relation to the "RS232-interface" in the description have evolved over the years. The protocol also works with RS422 and RS485 though!

#### 7.2.1 Data structure



## 7.2.2 Fieldbus parameters

Trigger byte: See „The trigger byte“, chapter 8.1

Length byte: See „The length byte“, chapter 8.2

## 7.2.3 RS232 parameter table

### 7.2.3.1 Start character (232 Start character)

If this character is defined, the gateway evaluates only the data at the RS232 interface following this start character. Each transmission from the gateway via the RS232 interface is initiated with the start character in this case.

### 7.2.3.2 Length 232 (232 Length)

If this byte is activated, the gateway, at the receive end, awaits as many bytes of useful data as specified in this byte by the RS232 transmitter. At the transmission end, the gateway then sets this byte to the number of useful data items transmitted by it. If byte "Length232" is not defined, the gateway, on reception at the RS232 interface, waits for the end criterion if this is defined. If no end criterion is defined either, as many characters as can be transferred in the fieldbus transmit buffer are read in via the RS232 interface.

As a special case for this parameter also a length byte with additional Timeout monitoring can be set in WINGATE. In that case the received characters will be discarded at a Timeout.



**Attention:**

***If "Timeout" is selected as end character, then this byte has no significance.***

### 7.2.3.3 Data area

The user data is transferred in this field.

### 7.2.3.4 Checksum

At the universal 232 protocol the following checksums can be selected: XOR, bitwise sum, XOR with negated result and bitwise sum with negated result. The checksum is always generated on the basis of bytes "Length232", "ID" and "Data area" if present. The checksum is generated by the gateway at the transmit end independently. On reception from the RS232 interface, the gateway checks the checksum and then transfers the useful data (without checksum) to the fieldbus buffer if no checksum errors have been detected. Otherwise, a local error message issued.

### 7.2.3.5 End character (232 End character)

If this character is defined, the gateway receives data from the RS232 interface up to this character. The "Timeout" criterion can be defined as a special case. In this case, the gateway continues to receive characters until a defined pause occurs. In the special case "Timeout" the "Length 232-byte" has no significance. At the transmit end, the gateway inserts the end character, if defined, as the last character of a transmission.

## 7.2.4 Communication sequence

The useful data (data area) arriving via the fieldbus is copied in accordance with chapter 7.2.1 transparently into the RS232 data field and transferred via the RS interface, whereby the protocol is supplemented in accordance with the configuration (start character, end character...). NO acknowledgement is issued !

If the "Trigger byte" (see chapter 8) is active, data is sent only on a change of this byte. If the "Length byte" (see chapter 8.11) is active, only as many of the following bytes as specified there are transferred.

Receive data at the RS interface is evaluated in accordance with the configured protocol, and the data field (data area (see chapter 7.2.1)) is sent to the fieldbus Master. If more characters have been received than the fieldbus block length, the trailing bytes are truncated and an Rx Overrun is indicated. If less have been received, padding with 0 occurs. If the "Length byte" is active, the number of received useful data items is entered there. If the, "Trigger byte" is active, this is incremented by one after each complete reception operation at the RS interface.

### 7.3 Protocol "CX (Pseudo)"

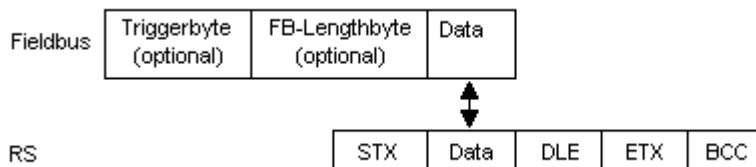
The protocol is based on the "Universal 232" protocol and has only preset parameters.

-----APPLICATION-----	
Protocol	<b>Universal 232</b>
232 Start character	<b>00</b>
232 Length	<b>No length byte</b>
232 End character	<b>FF</b>
232 RX Timeout (10ms)	<b>1</b>
232 Checksum	<b>No Checksum</b>
Start bits	<b>1</b>
Data bits	<b>8</b>
Stop bits	<b>1</b>
Parity	<b>Even</b>
Baudrate	<b>115200</b>
232 Interface	<b>232</b>

### 7.4 Protocol: 3964(R)

The 3964 protocol is used to transfer data between two serial devices. One partner must be a high-priority partner and the other must be a low-priority partner in order to resolve initialisation conflicts.

#### 7.4.1 Data structure 3964R



#### 7.4.2 Protocol definitions

The telegram format is as follows:

STX	Data	DLE	ETX	BCC
-----	------	-----	-----	-----

- The received net data is forwarded (transparently) in both directions unchanged.  
**Attention:** The DLE-doubling is excluded from it; that means one DLE (10H) on the bus-side is sent on the RS-side twice. A double DLE on the RS-side is only sent once to the bus-master.
- Data blocking is not scheduled.
- The net data length is restricted to 236 bytes per telegram.
- Communication always runs between high-priority and low-priority communication partners.

### 7.4.3 Data communication

#### 7.4.3.1 Initiation of data communication by the low-priority user

If the low-priority user also receives an STX in response to a transmitted STX, it interrupts its transmit request, reverts to Receive mode and acknowledges the received STX with DLE.

A DLE in the data string is duplicated and included in the checksum. The BCC is computed from XORing all characters.

#### 7.4.3.2 Conflicts

#### 7.4.3.3 Timeout times

The timeout times are preset by the definition of the 3964R protocol and cannot be overwritten !!!  
tq = acknowledgement timeout time (2 s).

The acknowledgement timeout time is started after transmission of control character STX. If no positive acknowledgement arrives within the acknowledgement timeout time, the job is repeated (max. 2 x). If it has not been possible to complete the job positively after two repetitions, the high-priority device nevertheless attempts to establish contact with the low-priority partner by transmitting STX (cycle corresponds to tq).

tz = character timeout time ( 200 ms)

If the 3964 R driver receives data, it monitors arrival of the individual characters within period tz. If no character is received within the timeout time, the protocol terminates transfer. No acknowledgement is sent to the coupling partner.

#### 7.4.3.4 Retries

In the event of negative acknowledgement or timeout, a telegram transmitted by the high-priority user is repeated twice. After this, the gateway signals communication as disturbed but still attempts to re-establish the connection.

#### 7.4.3.5 Initiation of data communication by the high-priority user

In the case of a negative acknowledgement or timeout, a telegram transmitted by the external device is repeated twice before a fault is signalled.

### 7.4.4 Protocol type 3964

The difference to protocol type 3964R is:

1. tq = acknowledge monitoring time
2. The checksum byte BCC is missing.

## 7.5 Protocol: MODBUS-RTU

### 7.5.1 Notes

- For reasons of simplicity, "MODBUS-RTU" is referred to as "MODBUS" in the text below.
- The terms "input" and "output" are always viewed from the gateway's point of view, i.e. fieldbus input data is the data sent by the fieldbus Master to the gateway.

### 7.5.2 UNIGATE® as MODBUS-Master

#### 7.5.2.1 Preparation

Before data exchange is commenced, the parameters "Baud rate", "Parity", "Start bits", "Stop bits" and "Data bits" and, if applicable, the "Trigger byte" and the "Length byte" must be set.

In addition, a "Response time" which corresponds to the maximum time up to which the Modbus Slave responds after a request must be set. UNIGATE® multiplies the value entered in WINGATE by 10 ms.

Since the Modbus operates with a variable data format - dependent on the required function and data length - but since the fieldbus requires a fixed data length, this must be preset by means of a selection in the GSD file (input and output are identical). This length should be selected by the user such that the longest Modbus request resp. response can be processed.

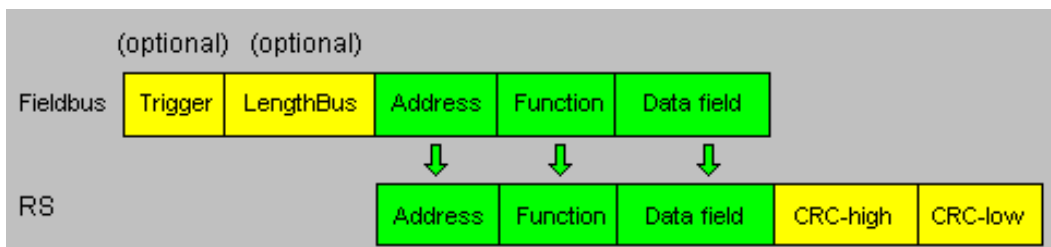
The user can choose whether the fieldbus requests are forwarded to the Modbus in case of a change (On Change) or on request (On Trigger).

In "Change" mode, detection of a change is based on the fact that the fieldbus data is compared with that of the last transmission, and a request is issued by the Modbus only in the case of a change.

The mode "Modbus request on demand" necessitates the first byte in the fieldbus containing a trigger byte (see chapter 8.1). This byte is not transferred to the Modbus and serves only to start a Modbus transmission. For this purpose, the gateway constantly monitors this trigger byte and sends data to the Modbus only when this byte has changed. In the reverse direction (to the fieldbus), the gateway transfers the number of received Modbus data records in this byte, i.e. this byte is incremented by the gateway after each data record.

If the "Length byte" is activated (see chapter 8.2), the gateway transfers only the number of bytes specified there. The number of received Modbus data items is saved in the direction of the fieldbus Master. The length always refers to bytes "Address" to "Data n" (inclusive in each case), always without CRC checksum.

#### 7.5.2.2 Data structure



#### 7.5.2.3 Communication sequence

The gateway always acts as the Slave with respect to the fieldbus and always acts as the Master at the Modbus end. Thus, data exchange must always be started by the fieldbus Master. The gateway fetches this data which must be structured in accordance with chapter "Data structure", from the fieldbus Master, determines the valid length of the Modbus data if the length byte is not activated, adds the CRC checksum and sends this data record as a request on the Modbus.

The response of the selected Slave is then sent to the fieldbus Master by the gateway - without CRC checksum. If no response occurs within the stipulated "Response time", the gateway signals a "TIMEOUT ERROR".

### 7.5.3 UNIGATE® as MODBUS-Slave

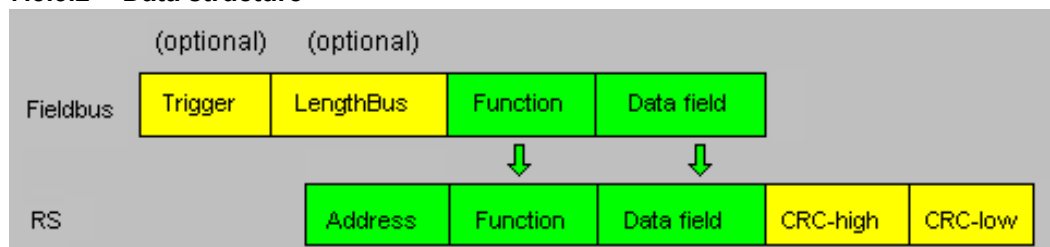
#### 7.5.3.1 Preparation

Before data exchange is commenced, the parameters "Trigger byte" and "Length byte", "Baud rate", "Parity", "Start bits", "Stop bits" and "Data bits" must be set.

At the rotary switch on the RS-side the MODBUS-ID has to be set, under which the gateway is addressed in the Modbus.

Since the Modbus operates with a variable data format - dependent on the required function and data length - but since the fieldbus requires a fixed data length, this must be preset by means of a selection in the GSD file. This length should be selected by the user such that the longest Modbus request resp. response can be processed.

#### 7.5.3.2 Data structure



#### 7.5.3.3 Communication sequence

The gateway always acts as the Slave with respect to the fieldbus and also acts as Slave at the Modbus end. A data exchange is always initiated by the MODBUS-Master via the RS-interface. If the Modbus-address (1st Byte) which is sent out by the Modbus-Master is identical with the address set on the gateway, the gateway sends the received data (without Modbus-address and CRC-check sum) to the fieldbus-master (look picture above). With it the gateway optionally completes as an introduction a Trigger byte and a Length byte.

The fieldbus-master detects when it has to analyse a record via the Trigger byte which is incremented by the gateway at every inquiry. The number of the following Modbus-data is to be found in the length byte.

Now the fieldbus-master has to analyse the Modbus-inquiry and it has to send back the answer in the same format (optionally with the leading Trigger byte and Length byte) via the fieldbus to the gateway.

The gateway then takes this answer and completes the Modbus-address and the CRC and sends the data to the Modbus-Master via the RS-interface. With it the data exchange is completed and the gateway waits for a new inquiry from the Modbus-Master.

## 7.6 Protocol Modbus ASCII Master/Slave

The fieldbus data exchange for Modbus ASCII is identical with RTU. On the serial site the UNIGATE automatically transfers the data in ASCII format.

-> For the description see chapter 7.5.2, UNIGATE® as MODBUS-Master respectively see chapter 7.5.3, UNIGATE® as MODBUS-Slave.

## 7.7 Protocol „Universal Modbus RTU Slave“

The UNIGATE® is a Modbus slave on the application side. The slave ID is set with the rotary coding switches S4 + S5 (S4 = High, S5 = Low).

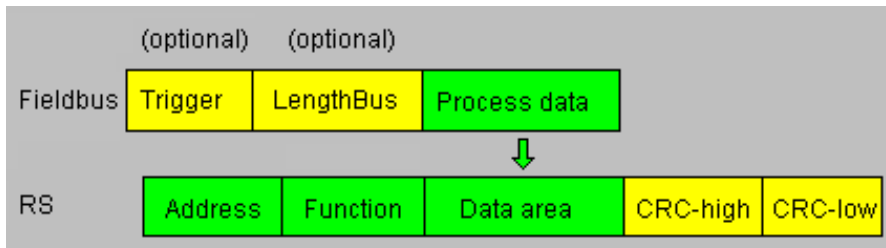
### 7.7.1 Data structure on the fieldbus side e.g.: PROFIBUS

Applies to In and Out

1. Byte: trigger byte, optional (see chapter 8.1 "The trigger byte")
2. Byte: fieldbus length byte, optional (see chapter 8.2, The length byte)
3. Byte: process data
4. Byte: process data

....

#### Data structure



#### 7.7.1.1 Example: FC1 + FC2

A Modbus Master (external device) sends a request with function code 1 or 2.

##### Note:

Modbus Master Request Address (High + Low)

Address request 01 .. 08 will always be on address 01.

Address request 09 .. 16 will always be on address 09.

Address request 17 .. 24 will always be on 17.

...

##### Configuration:

-----FIELDBUS-----	
Fieldbus ID	126
Data exchange	On Change
Fieldbus lengthbyte	active
-----APPLICATION-----	
Protocol	Universal Modbus RTU Slave

Fieldbus sends to UNIGATE®

08 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A...

**Note:** The 1. byte (0x08) is the fieldbus length byte. This means only the following 8 Bytes are stored in the UNIGATE®.

Connected Modbus Master sends request to the RS232/484 side of the UNIGATE®:

Start-Address 0001, Length 56 (38h), FC1 (-Read Coil Status)

[01] [01] [00] [00] [00] [38] [3d] [d8]

UNIGATE® sends response via RS232/485:

[01] [01] [07] [01] [02] [03] [04] [05] [06] [07] [6b] [c5]

Display of the data in the Modbus Master (FC1):

```
00001: <1> 00009: <0> 00017: <1> 00025: <0> 00033: <1> 00041: <0> 00049: <1>
00002: <0> 00010: <1> 00018: <1> 00026: <0> 00034: <0> 00042: <1> 00050: <1>
00003: <0> 00011: <0> 00019: <0> 00027: <1> 00035: <1> 00043: <1> 00051: <1>
00004: <0> 00012: <0> 00020: <0> 00028: <0> 00036: <0> 00044: <0> 00052: <0>
00005: <0> 00013: <0> 00021: <0> 00029: <0> 00037: <0> 00045: <0> 00053: <0>
00006: <0> 00014: <0> 00022: <0> 00030: <0> 00038: <0> 00046: <0> 00054: <0>
00007: <0> 00015: <0> 00023: <0> 00031: <0> 00039: <0> 00047: <0> 00055: <0>
00008: <0> 00016: <0> 00024: <0> 00032: <0> 00040: <0> 00048: <0> 00056: <0>
```

Example: StartAddress 0008, Length 80, FC2 (Read Input Status)  
[01] [02] [00] [07] [00] [50] [c9] [f7]

UNIGATE® sends response via RS232/485:

[01] [02] [0a] [02] [03] [04] [05] [06] [07] [08] [00] [00] [00] [8f] [7a]

#### 7.7.1.2 Example: FC3 (Read Holding Register) + FC4 (Read Input Register)

Fieldbus sends to the UNIGATE®

00 30 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 20 20 20...

(The configuration is "Data exchange = On Trigger", with an additional 1. control byte in the fieldbus data.)

„Fieldbus length byte = active“, in this example 30h (48d), the UNIGATE® copies the following 48 Byte from the fieldbus into the internal storage.

Connected Modbus Master sends request to the RS232/484 side of the UNIGATE®

[01] [03] [00] [00] [00] [14] [45] [c5]

UNIGATE® sends response via RS232/485:

[01] [03] [28] [02] [03] [04] [05] [06] [07] [08] [09] [0a] [0b] [0c] [0d] [0e] [0f] [10] [11] [12] [13] [14]...  
... [15] [16] [17] [18] [19] [1a]

Display of the process data in the Modbus Master:

```
40001: <0203H>
40002: <0405H>
40003: <0607H>
40004: <0809H>
40005: <0A0BH>
40006: <0C0DH>
40007: <0E0FH>
40008: <1011H>
40009: <1213H>
40010: <1415H>
40011: <1617H>
40012: <1819H>
40013: <1A20H>
40014: <2020H>
40015: <2020H>
40016: <0000H>
40017: <0000H>
40018: <0000H>
40019: <0000H>
40020: <0000H>
```

#### Functionality FC3 and FC4 in Protocol „Universal Modbus (RTU/ASCII) Slave:

From „Universalscript Deutschmann“ V1.5.1:

- FC3 (0x03): Read Holding Registers accesses Puffer Data to SPS.
- FC4 (0x04): Read Input Registers accesses Puffer Data From SPS.



### 7.7.1.3 Example: Write Single Coil FC5

The Fieldbus Master sent the following data to the UNIGATE® once:

07 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 17 18 19 1A 20 20 20...

1. Byte = Fieldbus length byte

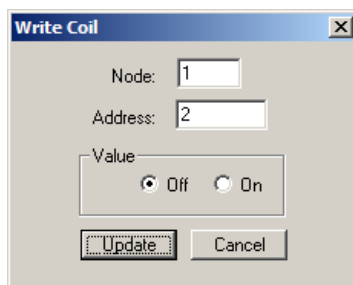
The following 7 byte are stored in the UNIGATE®, the rest is not overwritten.

With FC1 and the coil length = 80 (10 Bytes) a Modbus Master reads out the following data:

```
00001: <1> 00017: <1> 00033: <1> 00049: <1> 00065: <0>
00002: <0> 00018: <1> 00034: <0> 00050: <1> 00066: <0>
00003: <0> 00019: <0> 00035: <1> 00051: <1> 00067: <0>
00004: <0> 00020: <0> 00036: <0> 00052: <0> 00068: <0>
00005: <0> 00021: <0> 00037: <0> 00053: <0> 00069: <0>
00006: <0> 00022: <0> 00038: <0> 00054: <0> 00070: <0>
00007: <0> 00023: <0> 00039: <0> 00055: <0> 00071: <0>
00008: <0> 00024: <0> 00040: <0> 00056: <0> 00072: <0>
00009: <0> 00025: <0> 00041: <0> 00057: <0> 00073: <0>
00010: <1> 00026: <0> 00042: <1> 00058: <0> 00074: <0>
00011: <0> 00027: <1> 00043: <1> 00059: <0> 00075: <0>
00012: <0> 00028: <0> 00044: <0> 00060: <0> 00076: <0>
00013: <0> 00029: <0> 00045: <0> 00061: <0> 00077: <0>
00014: <0> 00030: <0> 00046: <0> 00062: <0> 00078: <0>
00015: <0> 00031: <0> 00047: <0> 00063: <0> 00079: <0>
00016: <0> 00032: <0> 00048: <0> 00064: <0> 00080: <0>
```

The fieldbus output data is only updated if it's triggered via a write command from the RS side.

For example via FC 5 :



Address 0002 stays unchanged on 0, however, the fieldbus output data is updated.

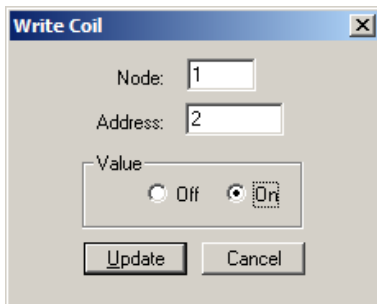
After a reset they are NULL (1st row) at first and are then updated (2nd row):

00 ...

1F 01 02 03 04 05 06 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00...

The 1. byte is the fieldbus length byte. It contains the number of usable characters, followed by the payload. The user data (internal buffer) is no bigger than 1024 byte.

In the following example the Bit (Coil) in Address 0002 is set to High (1):



The fieldbus data is updated:

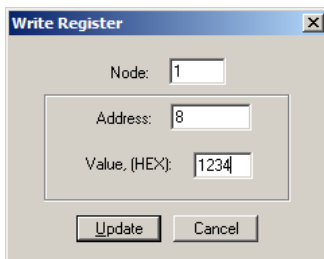
1F 03 02 03 04 05 06 07 00 00 00 00

The internal buffer reserves this value, which means it can be read back by the Master via FC1 Read Coil status:

```
00001: <1>
00002: <1>
00003: <0>
00004: <0>
00005: <0>
00006: <0>
00007: <0>
```

#### 7.7.1.4 Example: Write Single Register FC6

Modbus Master sends the value 1234H in Address 0008:



Der Modbus Master sends the request to the UNIGATE®:

[01] [06] [00] [07] [12] [34] [35] [7c]

The UNIGATE® sends a response:

[01] [06] [00] [07] [12] [34] [35] [7c]

The 1st row shows the fieldbus data BEFORE the write command:

1F 03 02 03 04 05 06 07 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00...

1F 03 02 03 04 05 06 07 00 00 00 00 00 00 12 34 00 00 00 00 00 00 00 00 00 00 00...

The 2nd row shows the fieldbus data AFTER the write command.

You can see that the value 00 07 is send as Address in the Modbus request. (As mentioned in the chapter Universal Modbus Master some Master pull System one as offset.)

This leads to the Byte-Offset for the fieldbus output data => 14. You start counting with the first process data value with Index NULL.

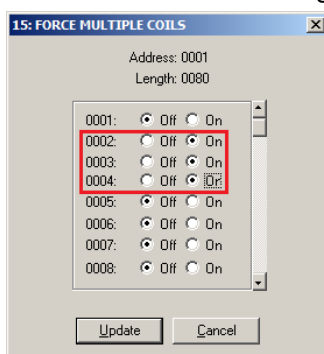
1F 03 02 ....

```
|  +---- 1. process value
+----- fieldbus length byte
```

#### 7.7.1.5 Example: Force multiple coils FC 15

**Note:** The address can only be passed in multiples of 8 incl. Null.  
Also 0, 8, 16, ... (Here you also have to keep in mind the offset of 1)

**Example:** Start address = 0001.  
Adr 0002 ... 004 was changed from Low to High



The 1st row shows the fieldbus BEFORE the request:

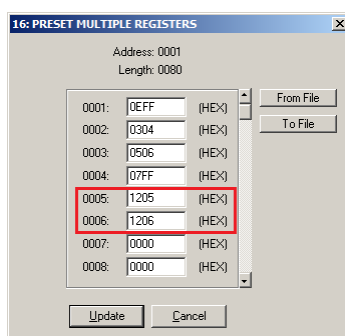
1F 00 FF 03 04 05 06 07 FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00...

```
1F 0E FF 03 04 05 06 07 FF 12 05 12 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00...
```

2nd row AFTER the request.

Therefor the 1. process data value changed from 00h to 0Eh.

#### 7.7.1.6 Example: Preset multiple register FC16



Only the content of the register address 0005 and 0006 was changed.

The 1st row shows the fieldbus BEFORE the request:

```
1F 0E FF 03 04 05 06 07 FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00...
```

```
1F 0E FF 03 04 05 06 07 FF 12 05 12 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00...
```

The 2nd row shows the fieldbus data content AFTER the update.

## 7.8 Protocol „Universal Modbus RTU Master“

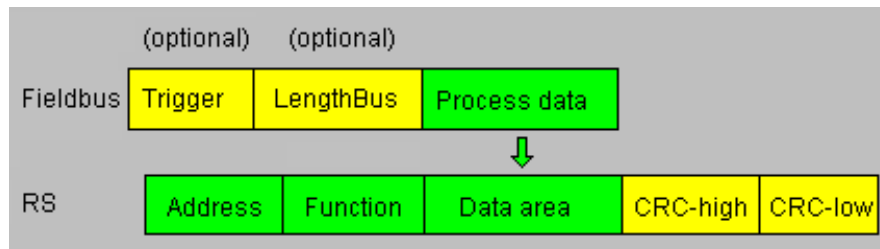
The UNIGATE® is Modbus-Master on the Application side.

### 7.8.1 Data structure Fieldbus side (e.g. PROFIBUS):

Applies to In and Out

1. Byte: Trigger-Byte, optional (see chapter 8.1, The trigger byte)
2. Byte: Fieldbus length byte, optional (see chapter 8.2, The length byte)
3. Process data

#### Data structure



### 7.8.2 Data structure Application side:

According to Modbus RTU Master definition.

#### Supported functions:

Read coil status FC1	(No. of Points = Bit)
Read input status FC2	(No. of Points = Bit)
Read multiple register FC3	(No. of Points = Word)
Read input registers FC4	(No. of Points = Word)
Force single coil FC5	(No. of Points – not used = fix 1 Bit)
Preset single register FC6	(No. of Points – not used = fix 1 Word)
Force multiple coils FC15	(No. of Points = Bit)
Preset multiple register FC16	(No. of Points = Word)

#### Note:

status and coil = 1 Bit, register = 16 Bit.

FC 1 + 2 as well as FC 3 + 4 are principally the same, the only difference is the definition of the start address.

At FC1 it starts at Null, at FC2 at 10 000.  
At FC3 it starts at 40 000, at FC4 at 30 000

### 7.8.3 Configuration: via Wingate since wcf Datei Version 396

Parameter Name	value range	Explanation
<b>Modbus Timeout (10ms)</b>	1 ... 255 (10ms ... 2550ms)	Max. Waiting time for the "Response" before an error 9 is generated by timeout. If "RX Poll Retry" > 0 an error is only generated after retries.
<b>RX Poll Retry</b>		Retry of the last, invalid replied "Request"
<b>RX Poll Delay (10ms)</b>		Pause before the next "Request"

#### Configurations parameter for a Modbus Request:

**Req. 1 Slave ID:** Slave ID of the Modbus slave participant

**Req. 1 Modbus Function:** see "supported functions"

**Req. 1 StartAdr (hex):** Start address (High / Low) of the Modbus register from which should be read/written

**Req. 1 No. of Points (dec):** Number of the to read/to write register/coils

**Req. 1 Fieldbus Map Adr(Byte):** Position of the to be copied process value from/to the fieldbus range, depending on the write/read-command. If the value is NULL the process data is automatically lined up behind the other.

Up to 24 requests can be configured.

#### Additional configuration possibilities in the setting „Req. ... Modbus Function“:

**jump to Req. 1:** jump to 1. request entry

**disable this Req.:** skip this request and perform the next request entry.

„(10ms)“ : adjustable in 10ms steps

„(hex)“: Enter in hexadecimal style.

„(dec)“: Enter in decimal style.

„(Byte)“: Counting in bytes, starting at the position Null. Attention: For read commands, e.g. FC3, after the trigger- and lengthbyte the first process value is the position null, which is copied to the fieldbus to the PLC.  
For write commands, e.g. FC16, the position Null is the trigger byte.

### 7.8.3.1 Example: Read coil status FC1

Configuration

Req. 3 Slave ID	1
Req. 3 Modbus Function	Read coil status FC1
Req. 3 StartAdr (hex)	0004
Req. 3 No. of Points (dec)	2
Req. 3 Fieldbus Map Adr(Byte)	6

Data content Modbus Slave

Device Id: <input type="text" value="1"/>	
Address: <input type="text" value="0001"/>	MODBUS Point Type
Length: <input type="text" value="24"/>	<input type="text" value="01: COIL STATUS"/>

00001: <0>	00009: <0>	00017: <0>
00002: <0>	00010: <0>	00018: <0>
00003: <0>	00011: <0>	00019: <0>
00004: <0>	00012: <0>	00020: <0>
00005: <1>	00013: <0>	00021: <0>
00006: <0>	00014: <0>	00022: <0>
00007: <0>	00015: <0>	00023: <0>
00008: <0>	00016: <0>	00024: <0>

UNIGATE® reads Address 5 + 6 and copies it into the 6. byte of the output buffer.

Fieldbus output data (UNIGATE® -> SPS)

66 07 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 0

1. Byte = Trigger byte (value = 0x66 )
2. Byte = Fieldbus length byte (value = 0x07)
3. Byte = Fieldbus Map Adr 0 (value = 0x00)
4. Byte = Fieldbus Map Adr 1 (value = 0x00)
5. Byte = Fieldbus Map Adr 2 (value = 0x00 )
6. Byte = Fieldbus Map Adr 3 (value = 0x00)
7. Byte = Fieldbus Map Adr 4 (value = 0x00)
8. Byte = Fieldbus Map Adr 5 (value = 0x00)
9. Byte = Fieldbus Map Adr 6 (value = 0x01) see configuration
10. Byte = Fieldbus Map Adr 7 (value = 0x00)
11. Byte ...

In the following example the value in address 6 in the Modbus Master is changed from 0 to 1.

00001: <0>
00002: <0>
00003: <0>
00004: <0>
00005: <1>
00006: <1>
00007: <0>
00008: <0>

```
AD 07 00 00 00 00 00 00 01 00 00 00 00 00 00 00
AE 07 00 00 00 00 00 00 03 00 00 00 00 00 00 00
```

The modification can be seen here:

9. Byte = Fieldbus Map Adr 6 (Wert = 0x01) => 0x03

A modification of address 7 in the Modbus slave has no consequences to the fieldbus output side because "No. Of Points = 2" is set in the configuration.

```
00001: <0>
00002: <0>
00003: <0>
00004: <0>
00005: <1>
00006: <1>
00007: <1>
00008: <0>
```

The value stays unchanged on 0x03:

```
1F 07 00 00 00 00 00 00 03 00 00 00 00 00 00
```

### 7.8.3.2 Example: Read input status FC2

The following example shows the content of address 10007 ... 10009 is mapped/copied into the 8. fieldbus output byte.

Req. 1 Slave ID	1
Req. 1 Modbus Function	Read input status FC2
Req. 1 StartAdr (hex)	0006
Req. 1 No. of Points (dec)	3
Req. 1 Fieldbus Map Adr(Byte)	8

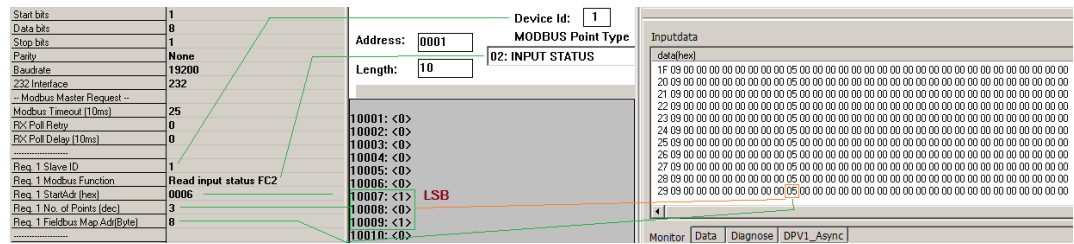
		Device Id: <input type="text" value="1"/>
Address: <input type="text" value="0001"/>	MODBUS Point Type	
Length: <input type="text" value="10"/>	<input type="text" value="02: INPUT STATUS"/>	

10001: <0>
10002: <0>
10003: <0>
10004: <0>
10005: <0>
10006: <0>
10007: <1>
10008: <0>
10009: <0>
10010: <0>

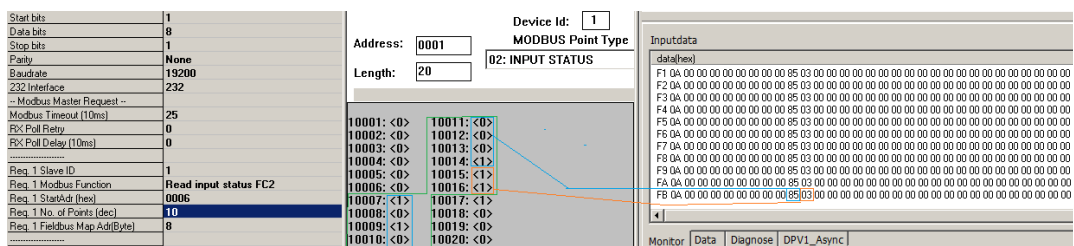
```
76 09 00 00 00 00 00 00 00 01 00 00 00 00 00
```

Here the content of the address 10009 is changed from 0 -> 1



In the following example only the "No. Of Points" is switched to 10.

Which means that now 10 Bits => 2 Byte are read out. This is also the reason why the fieldbus length byte (2. fieldbus byte) at 0x0A increases by 1 Byte.



### 7.8.3.3 Example: Read multiple register FC3

Protocol	Universal Modbus RTU Master
-- Modbus Master Request --	
Modbus Timeout (10ms)	25
RX Poll Retry	0
RX Poll Delay (10ms)	0
-----	
Req. 1 Slave ID	1
Req. 1 Modbus Function	Read multiple register FC3
Req. 1 StartAdr (hex)	0001
Req. 1 No. of Points (dec)	2
Req. 1 Fieldbus Map Adr(Byte)	0

RX Poll Delay = 0 is automatically set to 1 by the firmware.

Modbus-Request:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Slave ID	Modbus Function	StartAdr High	StartAdr Low	No. of Points High	No. of Points Low	CRC High	CRC Low
1	3	0x00	0x01	0	2	x	y

The CRC value is automatically calculated by the UNIGATE®

The UNIGATE® sends out the request (RX Poll Retry = 0) one time via the RS interface, and waits a maximum of 250 ms (Modbus Timeout = 25) on the response.

Fieldbus Map Adr = 0 -> not activ



Thereby the addressed slave holds the following data in its registers.:

register	
address	value(hex)
40000	0x0000
40001	0x0202
40002	0x0303
40003	0x0000
40004	0x0000

register = 1 Word = 2 Byte



In the documentation of some applications, an Offset + 1 at the address is assumed. The notation for address "40000" stands for "holding register". But in actuality address 0x0000 is meant by it. This is not uniform in the Modbus-Slave documentations. (E.g. the PC simulation tool "ModSim32" has this offset).

If a valid response is received, the four byte (No. Of Points = 2) process value (Modbus-Data) will be copied to the fieldbus from "Fieldbus Map Adr(Byte)" = 0 on.

Fieldbus data from UNIGATE® -> SPS:

51 13 02 02 03 03 30 04 01 00 01 00 00 00 02 57 00 01 03 00 00 00 00 00 00 00 ...

Byte 0 = Trigger-Byte „0x51“

Byte 1 = Fieldbus length byte „0x13“

Byte 2 = Process value (High) from StartAdr „0x02“

Byte 3 = Process value (Low) from StartAdr „0x02“

Byte 4 = Process value (High) from StartAdr + 1 „0x03“

Byte 5 = Prozess value (Low) from StartAdr + 1 „0x03“

#### 7.8.3.4 Example: Read input registers FC4

(see chapter 7.8.3.3, Example: Read multiple register FC3)

#### 7.8.3.5 Example: Force single coil FC5

At FC5 a bit is set in the Modbus slave, if the mapped fieldbus byte is bigger (>) than NULL.

Configuration

Modbus Slave(impact)SPS sends  
Fieldbus data (reason)

Req. 1 Slave ID	1	00004: <0>	
Req. 1 Modbus Function	Force single coil FC5	00005: <0>	
Req. 1 StartAdr (hex)	0005	00006: <1>	
Req. 1 No. of Points (dec)	6	00007: <0>	
Req. 1 Fieldbus Map Adr(Byte)	7	00008: <0>	
		00009: <0>	

**Note:** No. of Points is not required

Another example for when a second request is configured:

Req. 1 Slave ID	1	00001: <0>	
Req. 1 Modbus Function	Force single coil FC5	00002: <0>	
Req. 1 StartAdr (hex)	0005	00003: <0>	
Req. 1 No. of Points (dec)	6	00004: <0>	
Req. 1 Fieldbus Map Adr(Byte)	7	00005: <0>	
		00006: <1>	
Req. 2 Slave ID	1	00007: <0>	
Req. 2 Modbus Function	Force single coil FC5	00008: <1>	
Req. 2 StartAdr (hex)	0007	00009: <0>	
Req. 2 No. of Points (dec)	6	00010: <0>	
Req. 2 Fieldbus Map Adr(Byte)	10		

### 7.8.3.6 Example: Preset single register FC6

Configuration

Req. 1 Slave ID	1
Req. 1 Modbus Function	Preset single register FC6
Req. 1 StartAdr (hex)	0005
Req. 1 Fieldbus Map Adr(Byte)	7

SPS sends to UNIGATE®

01 00 00 00 00 00 00 **FF23** 00 FF 00 00 00 00 00 FF 00 00 00 00 00 00 00 00 00 00 00 00 ...

UNIGATE® sends Modbus RTU request

[01] [06] [00] [05] [ff] [23] [99] [e2]

Modbus Slave sends response

[01] [06] [00] [05] [ff] [23] [99] [e2]

Storage content of Modbus Slave after Response:

```

40001: <0000H>
40002: <0000H>
40003: <0000H>
40004: <0000H>
40005: <0000H>
40006: <FF23-H>
40007: <0000H>
40008: <0000H>
40009: <0000H>
40010: <0000H>

```

### 7.8.3.7 Example: Force multiple coils FC15

Configuration

Req. 1 Slave ID	1
Req. 1 Modbus Function	Force multiple coils FC15
Req. 1 StartAdr (hex)	0002
Req. 1 No. of Points (dec)	10
Req. 1 Fieldbus Map Adr(Byte)	2

Fieldbus Master sends:

0E 00 FF 05 00...

UNIGATE® sends request:

[01] [0f] [00] [02] [00] [0a] [02] [ff] [05] [65] [29]

Modbus Slave sends response:

[01] [0f] [00] [02] [00] [0a] [74] [0c]

Storage content of Modbus Slave after response:

```

00001: <0>    00011: <1>
00002: <0>    00012: <0>
00003: <1>    00013: <0>
00004: <1>    00014: <0>
00005: <1>    00015: <0>
00006: <1>    00016: <0>
00007: <1>    00017: <0>
00008: <1>    00018: <0>
00009: <1>    00019: <0>
00010: <1>    00020: <0>

```

Hex	FF	05
Bin		00000101
Position	8 7 6 5 4 3 2 1	11 10 9

Please keep in mind that No. Of coils = 10, hence, only the lower bit in address 0011 is written at the value 0x05. Address 0013 would already be bit No. 11, which is not transmitted anymore.

### 7.8.3.8 Example: Preset multiple register FC16

Configuration

Req. 1 Slave ID	1
Req. 1 Modbus Function	Preset multiple register FC16
Req. 1 StartAdr (hex)	0002
Req. 1 No. of Points (dec)	10
Req. 1 Fieldbus Map Adr(Byte)	2

Fieldbus Master sends:

BA 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F 10 11 12 13 14 15 16 ...

UNIGATE® sends Request:

[01] [10] [00] [02] [00] [0a] [14] [01] [02] [03] [04] [05] [06] [07] [08] [09] [0a] [0b] [0c] [0d] [0e] [0f]...  
... [10] [11] [12] [13] [14] [3d] [e4]

Modbus Slave sends Response:

[01] [10] [00] [02] [00] [0a] [e1] [ce]

Storage content Modbus Slave to Response:

```

40001: <0000H>
40002: <0000H>
40003: <0102H>
40004: <0304H>
40005: <0506H>
40006: <0708H>
40007: <090AH>
40008: <0B0CH>
40009: <0D0EH>
40010: <0F10H>
40011: <1112H>
40012: <1314H>
40013: <0000H>

```

## 7.9 Protocol „Universal Modbus ASCII Master/Slave“

The fieldbus data exchange for Modbus ASCII is identical with RTU. The UNIGATE® automatically transmits the data in ASCII format on the serial side.

Protocol description: see chapter 7.7 "Protocol „Universal Modbus RTU Slave“" respectively see chapter 7.8 "Protocol „Universal Modbus RTU Master“".

## 7.10 Protocol Modbus TCP client encapsulation

The "Modbus TCP client encapsulation" fieldbus transport protocol can only be used in conjunction with the "Modbus RTU Slave" application transport protocol.

### 7.10.1 Function

#### 7.10.1.1 UNIGATE® CL:

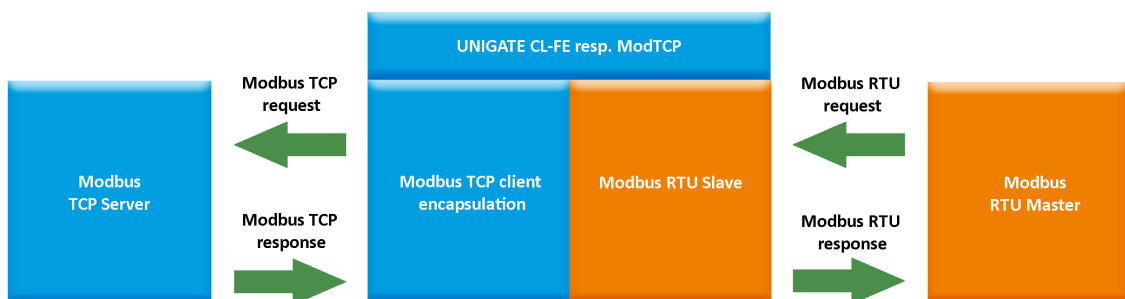
- Ethernet TCP/IP (Modbus TCP) transport protocol: Modbus TCP client encapsulation
- Application protocol: Modbus RTU Slave (Modbus RTU Slave ID adjusted via rotary coding switch S4+S5)

#### Description:

The Modbus requests of the Modbus RTU Master connected to the application side are forwarded device internally via the application protocol "Modbus RTU Slave" to the fieldbus transport protocol "Modbus TCP client encapsulation". So that the Modbus requests are transmitted to the Modbus TCP server (Target IP address). The Modbus Response of the Modbus TCP server is then transferred to the Modbus RTU Master in reverse order.

**Note:** A valid Modbus RTU slave ID (1 .. 247) must be set via the two rotary coding switches S4 and S5. The ID must match the one in the Modbus requests.

**Example:** UNIGATE CL-FE or ModTCP S4 + S5 = 01. Then "1" has to be displayed in the record in Modbus Request parameter "Req. ... Slave ID".



### 7.10.1.2 UNIGATE® CX:

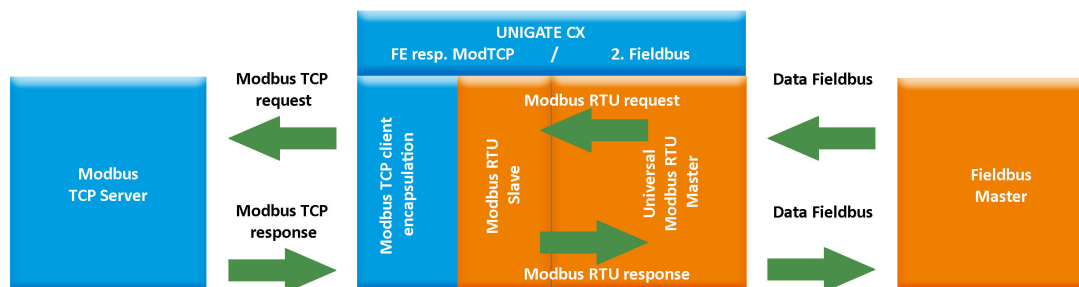
- FE or ModTCP side: Ethernet TCP/IP (Modbus TCP) Transport protocol: Modbus TCP client encapsulation
- FE or ModTCP side: Application protocol: Modbus RTU Slave (set Modbus RTU Slave ID via rotary coding switches S4+S5)
- 2. Fieldbus: Application protocol: Universal Modbus RTU Master (up to 24 Modbus Requests configurable)

#### Description:

The Modbus requests are configured in the "Universal Modbus RTU Master" application protocol. These are then passed on internally via the application protocol "Modbus RTU Slave" to the fieldbus transport protocol "Modbus TCP client encapsulation" so that the Modbus requests are transmitted to the Modbus TCP server (Target IP address). The Modbus Response of the Modbus TCP server is then transferred in reverse order to the "Universal Modbus RTU Master" application protocol. Depending on the function code (for example, FC3), the data is then written to the relevant fieldbus or read by the fieldbus.

**Note:** A valid Modbus RTU ID (1 ... 247) must be set via the two rotary coding switches S4 and S5. The ID must match the one in the configured Modbus requests.

**Example:** UNIGATE CL-FE or ModTCP S4 + S5 = 01. Then "1" has to be displayed in the record in Modbus Request parameter "Req. ... Slave ID".



## 8 Optional bus parameter

### 8.1 The trigger byte

If the data is always transmitted cyclically, the gateway must recognize when the user wants to send new data. This is usually done by having the gateway compare the data being transferred with the old data stored internally - data exchange on change (Fieldbus Data Exchange → On Change).

In some cases, this can not be used as a criterion, e.g. B. if always the same data should be sent. For this reason, the user can set that he wants to control the transmission via a trigger byte (Fieldbus Data exchange → On Trigger). In this mode, the gateway always (and only) sends when the trigger byte is changed.

If Trigger-Byte mode is activated, the gateway increments the trigger byte each time a telegram is received. The first byte in the fieldbus input / output data buffer is used as trigger byte if this mode is switched on.

### 8.2 The length byte

The user can configure whether the transmit length is also to be stored as a byte in the input/output data area (Fieldbus lengthbyte → active). In transmit direction, as many bytes as specified in this byte are sent. On reception of a telegram the gateway enters the number of characters received.

### 8.3 Swap word

With activated "Swap word" the data is word-swapped to and from the fieldbus. I.e. High and Low byte are word-swapped and transferred in a 16-bit word. It affects the entire fieldbus Buffer.

## 9 Fieldbus parameters / Ethernet parameters

The data of fieldbus A is passed on to fieldbus B depending on the handling of the scripts

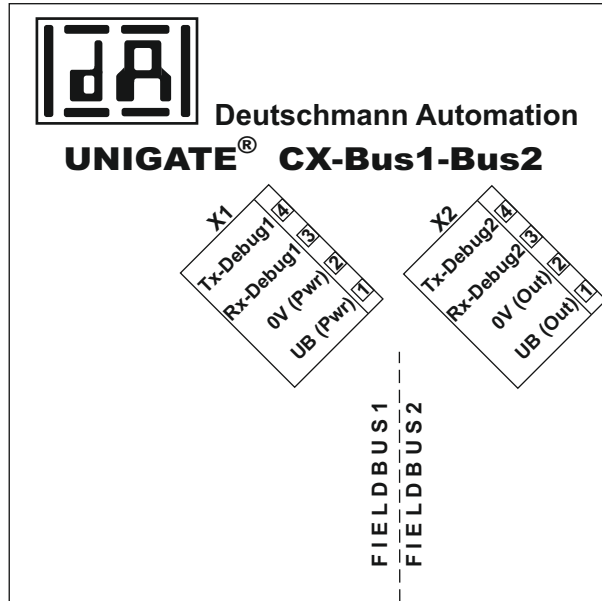
Here the following basic requirement have to be considered:

	Bus connection	Bus data	Bus baud raten	Bus ID
<b>CANopen Slave, CANopen Master, CAN Layer 2</b>	9 pin D-SUB connector	255 Byte I/O	125 kbit/s up to 1 Mbit/s adjustable via DIP-switch	Adjustable via DIP-switch
<b>DeviceNet</b>	5 pin screw-pug connector	255 Byte I/O	125, 250 and 500kbit/s adjustable via DIP-switch	Adjustable via DIP-switch
<b>EtherCAT</b>	2x RJ45 In and Out	512 Byte I/O	100 Mbit/s full duplex	Permanent MAC-address, is automatically assigned
<b>EtherNet/IP</b>	2x RJ45	1060 Byte I/O	100 Mbit/s full duplex	IP-address adjustable via WINGATE
<b>Fast Ethernet Modbus TCP</b>	RJ45	1024 Byte I/O	10/100 Mbit/s	IP-address via WINGATE or Script or UNIGATE Scan Tool
<b>LONWorks</b>	4 pin screw-plug connector	62 In and Out SNVTs, 512 Byte I/O	FTT-10A, 78 kBit/s	Permant Neuron ID
<b>MPI</b>	9 pin D-SUB socket	255 Byte I/O	Automatic detection (9600 bit/s - 12 Mbit/s)	Adjustable via rotary switch
<b>PROFIBUS</b>	9 pin D-SUB socket	244 Byte I/O	Automatic detection (9600 bit/s - 12 Mbit/s)	Adjustable via rotary switch or via PROFIBUS-Master
<b>PROFINET</b>	2x RJ45	1440 Byte I/O	100 Mbit/s full duplex	Adjustable or is assigned by the Master



## 10 Hardware connections, switches and LEDs

### 10.1 Device label



Picture 1: Connection labelling



**Note:**

X1 + X2 are always available.

X3 and maybe X4 depend on the combination of the Fieldbuses.

### 10.2 Connectors

#### 10.2.1 Connector supply voltage and DEBUG-interface 1

Pin assignment X1 (4-pole screw-plug connector, on the bottom side, right at the back)

Pin No.	Name	Function
1	UB (Pwr)	10..33 V supply voltage / DC
2	0 V (Pwr)	0 V supply voltage / DC
3	Rx-Debug 1	Receive signal Debug Fieldbus 1
4	Tx-Debug 1	Transmit signal Debug Fieldbus 1



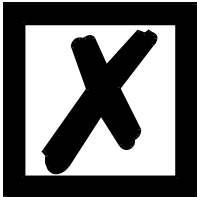
**Attention:**

Ground for the DEBUG-Interface must be connected with pin 2 0V (pwr)!

### 10.2.2 Connector output voltage and DEBUG-interface 2

Pin assignment X2 (4-pole screw-plug connector, on the bottom side, left at the back)

Pin No.	Name	Function
1	UB (Out)	10..33 V output (dependent on supply voltage X1)
2	0 V (Out)	0 V output
3	Rx-Debug 2	Receive signal Debug Fieldbus 2
4	Tx-Debug 2	Transmit signal Debug Fieldbus 2



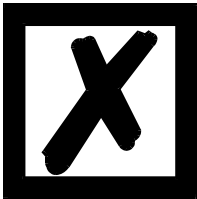
**Attention:**  
Ground for the DEBUG-Interface must be connected with pin 2 0V (pwr) of X1!

### 10.3 Power supply

The device must be powered with 10-33 VDC. The voltage supply is made through the 4-pole screw-plug connector X1 at the device's bottom side.

Please note that the devices of the series UNIGATE® should not be operated with AC voltage.

### 10.4 LEDs, switches, bus connection



For a description of the LEDs, switches and the Fieldbus- or Industrial Ethernet-connectors please take a look at the instruction manual UNIGATE® CL for the relevant Fieldbus.

### 10.5 UNIGATE® CX connection cable

A pre-assembled connection cable (Art.-No. V3791) is available as accessory. The cable connects the Gateway with the two Debug-interfaces and power supply.

## 11 Installation guidelines

### 11.1 Installation of the module

The module with the maximum dimension (46 x 100 x 117 mm W x H x D) has been developed for switch cabinet use (IP 20) and can thus be mounted only on a standard mounting channel (deep top-hat rail to EN 50022).

#### 11.1.1 Mounting

- Engage the module from the top in the top-hat rail and swivel it down so that the module engages in position.
- Other modules may be rowed up to the left and right of the module.
- There must be at least 5 cm clearance for heat dissipation above and below the module.
- The standard mounting channel must be connected to the equipotential bonding strip of the switch cabinet. The connection wire must feature a cross-section of at least 10 mm<sup>2</sup>.

#### 11.1.2 Removal

- First disconnect the power supply and signal lines.
- Then push the module up and swivel it out of the DIN-rail.

Vertical installation

The standard mounting channel may also be mounted vertically so that the module is mounted turned through 90°.

### 11.2 Wiring

#### 11.2.1 Connection systems

The following connection systems must resp. may be used when wiring the module:

- Standard screw-type/plug connection (power supply + RS)
  - 9-pin SUB-D plug connectors (CANopen and RS232 connection)
- a) In the case of standard screw-type terminals, one lead can be clamped per connection point. It is best to then use a screwdriver with a blade width of 3.5 mm to firmly tighten the screw.

Permitted cross-sections of the line:

- Flexible line with wire-end ferrule: 1 x 0.25 ... 1.5 mm<sup>2</sup>
- Solid conductor: 1 x 0.25 ... 1.5 mm<sup>2</sup>
- Tightening torque: 0.5 ... 0.8 Nm

- b) The plug-in connection terminal strip is a combination of standard screw-type terminal and plug connector. The plug connection section is coded and can thus not be plugged on the wrong way round.
- c) The 9-pin SUB-D plug connectors are secured with two screws with "4-40-UNC" thread. It is best to use a screwdriver with a blade width of 3.5 mm to screw the screw tight.  
Tightening torque: 0.2... 0.4 Nm

##### 11.2.1.1 Power supply

The device must be powered with 10..33 V DC.

- Connect the supply voltage to the 4-pole plug-in screw terminal in accordance with the labelling on the device.

##### 11.2.1.2 Equipotential bonding connection

The connection to the potential equalization automatically takes place it is put on the DIN-rail.

## 11.2.2 Communication interface

### 11.2.2.1 CANopen Slave, CANopen Master / CAN Layer 2

This interface is located on the module in the form of a 9-pin D-SUB plug on the front side of the housing.

- Plug the CANopen connector onto the SUB-D plug labelled "CANopen".
- Firmly screw the securing screws of the plug connector tight using a screwdriver.
- If the module is located at the start or end of the CANopen line, you must connect the bus terminating resistor integrated in the gateway. In order to do this, slide the slide switch to the position labelled ...on...
- If the module is not located at the start or at the end, you must set the slide switch to position "off".

### 11.2.2.2 DeviceNet

This interface is located on the module in the form of a 5-pin screw-plug-connector on the lower side of the housing.

- Plug the DeviceNet connecting plug onto the socket labelled "DeviceNet".
- If the module is located at the start or end of the DeviceNet line, you must switch on the bus terminating resistor integrated in the gateway. In order to do this, slide the sliding switch to the position labelled ...on...
- If the module is not located at the start or at the end, you must set the sliding switch to position "off".

### 11.2.2.3 EtherCAT

This interface is located on the module in the form of two 8-pin RJ45 sockets on the bottom side of the housing.

- Plug the EtherCAT-connector onto one of the RJ45 sockets labeled "In" (cable from the Master) or "Out" (further cable to the next EtherCAT-Slave).
- Please make sure that the length of the line to the adjacent Ethernet participants does not fall below 0.6 m.

### 11.2.2.4 EtherNet/IP

This interface is located on the module in the form of a 8-pin RJ45 socket on the bottom side of the housing.

- Plug the Ethernet/IP-connector onto the RJ45 socket labeled "RJ45 Ethernet/IP" until it snaps in.
- Please make sure that the length of the line to the adjacent Ethernet participants does not fall below 0.6 m.

### 11.2.2.5 Ethernet

This interface is located on the module in the form of a 8-pin RJ45 socket on the bottom side of the housing.

- Plug the Ethernet-connector onto the RJ45 socket labeled "RJ45 Ethernet" until it snaps in.
- Please make sure that the length of the line to the adjacent Ethernet participants does not fall below 0.6 m.

#### 11.2.2.6 LONWorks

This interface is located on the module in the form of a 4-pin screw-plug-connector on the lower side of the housing.

- Plug the LONWorks connecting plug onto the socket labelled "LONWorks".

#### 11.2.2.7 MPI

This interface is located on the module in the form of a 9-pin D-SUB socket on the front side of the housing.

- Plug the MPI-bus connector onto the D-SUB socket labelled "MPI-bus".
- Firmly screw the securing screws of the plug connector tight using a screwdriver.
- If the module is located at the start or end of the MPI-bus line, you must connect the bus terminating resistor integrated in the Gateway. In order to do this, slide the slide switch to the position labelled ...on...
- If the module is not located at the start or at the end, you must set the slide switch to position "off".

#### 11.2.2.8 PROFIBUS DP

This interface is located on the module in the form of a 9-pin D-SUB socket on the front side of the housing.

- Plug the PROFIBUS connector onto the SUB-D socket labelled "PROFIBUS DP".
- Firmly screw the securing screws of the plug connector tight using a screwdriver.
- If the module is located at the start or end of the PROFIBUS line, you must connect the bus terminating resistor integrated in the gateway. In order to do this, slide the slide switch to the position labelled ...on...
- If the module is not located at the start or at the end, you must set the slide switch to position "off".

#### 11.2.2.9 PROFINET-IO

This interface is located on the module in the form of a 8-pin RJ45 socket on the bottom side of the housing.

- Plug the Profinet-connector onto the RJ45 socket labelled "RJ45 PROFINET-IO" until it snaps in.
- Please make sure that the length of the line to the adjacent Ethernet participants does not fall below 0.6 m.

### 11.2.3 Line routing, shield and measures to combat interference voltage

This chapter deals with line routing in the case of bus, signal and power supply lines, with the aim of ensuring an EMC-compliant design of your system.

### 11.2.4 General information on line routing

- Inside and outside of cabinets

In order to achieve EMC-compliant routing of the lines, it is advisable to split the lines into the following line groups and to lay these groups separately.

- ⇒ Group A:
  - shielded bus and data lines (e.g. for CANopen, RS232C and printers etc.)
  - shielded analogue lines
  - unshielded lines for DC voltages  $\geq 60$  V
  - unshielded lines for AC voltage  $\geq 25$  V
  - coaxial lines for monitors
- ⇒ Group B:
  - unshielded lines for DC voltages  $\geq 60$  V and  $\geq 400$  V
  - unshielded lines for AC voltage  $\geq 24$  V and  $\geq 400$  V
- ⇒ Group C:
  - unshielded lines for DC voltages  $> 400$  V

The table below allows you to read off the conditions for laying the line groups on the basis of the combination of the individual groups.

	Group A	Group B	Group C
Group A	1	2	3
Group B	2	1	3
Group C	3	3	1

Table: Line laying instructions as a function of the combination of line groups

- 1) Lines may be laid in common bunches or cable ducts.
- 2) Lines must be laid in separate bunches or cable ducts (without minimum clearance).
- 3) Lines must be laid in separate bunches or cable ducts inside cabinets but on separate cable racks with at least 10 cm clearance outside of cabinets but inside buildings .

#### 11.2.4.1 Shielding of lines

Shielding is intended to weaken (attenuate) magnetic, electrical or electromagnetic interference fields.

Interference currents on cable shields are discharged to earth via the shielding bus which is connected conductively to the chassis or housing. A low-impedance connection to the PE wire is particularly important in order to prevent these interference currents themselves becoming an interference source.

Wherever possible, use only lines with braided shield. The coverage density of the shield should exceed 80 %. Avoid lines with foil shield since the foil can be damaged very easily as the result of tensile and compressive stress on attachment. The consequence is a reduction in the shielding effect.

In general, you should always connect the shields of cables at both ends. The only way of achieving good interference suppression in the higher frequency band is by connecting the shields at both ends.

The shield may also be connected at one end only in exceptional cases. However, this then achieves only an attenuation of the lower frequencies. Connecting the shield at one end may be more favourable if

- it is not possible to lay an equipotential bonding line
- analogue signals (a few mV resp. mA) are to be transmitted
- foil shields (static shields) are used.

In the case of data lines for serial couplings, always use metallic or metallised plugs and connectors. Attach the shield of the data line to the plug or connector housing.

If there are potential differences between the earthing points, a compensating current may flow via the shield connected at both ends. In this case, you should lay an additional equipotential bonding line.

Please note the following points when shielding:

- Use metal cable clips to secure the shield braiding. The clips must surround the shield over a large area and must have good contact.
- Downstream of the entry point of the line into the cabinet, connect the shield to a shielding bus. Continue the shield as far as the module, but do not connect it again at this point!

## 12 Technical data

### 12.1 Device data

The technical data of the module is given in the table below.

No.	Parameter	Data	Explanations
1	Location	Switch cabinet	DIN-rail mounting
2	Enclosure	IP20	Protection against foreign bodies and water to IEC 529 (DIN 40050)
3	Service life	10 years	
4	Housing size	46 x 100 x 117 mm (screw-plug connector not included) 46 x 106 x 117 mm (screw-plug-connector included)	W x H x D W x H x D (maximum including connector)
5	Installation position	Any	
6	Weight	Max. 260 g	min. 230 g up to max. 260 g (depending on the version)
7	Operating temperature	-40°C ... +85°C -25°C ... +85°C (all versions with RJ45)	The negative temperatures are only valid for the usual conditions (not condensing)
8	Storage/transport temperature	-40°C ... +85°C	
9	Atmospheric pressure during operation during transport	795 hPa ... 1080 hPa 660 hPa ... 1080 hPa	
10	Installation altitude	2000 m 4000 m	Unrestricted Restricted - Ambient temperature ≤ 40°C
11	Relative humidity	Max. 80 %	No condensation, no corrosive atmosphere
12	External power supply	10...33 V DC	Standard power supply unit to DIN 19240
13	Current consumption at 24 VDC	Typ: 60 mA up to 200 mA max: 80 mA up to 240 mA	At 10V: typ. 620 mA (depending on the version)
14	Reverse voltage protection	Yes	But device does not function!
15	Short-circuit protection	Yes	
16	Overload protection	Poly-switch	Thermal fuse
17	Undervoltage detection (USP)	≤ 9 V DC	
18	Emergency power supply	≥ 5 ms	Device fully operable

Table: Technical data of the module



**The interface data can be found in the instruction manual UNIGATE® CL for the relevant bus.**



## 13 Commissioning guide

### 13.1 Note

Only trained personnel following the safety regulations may commission the UNIGATE®.

### 13.2 Components

You will require the following components to commission the UNIGATE®:

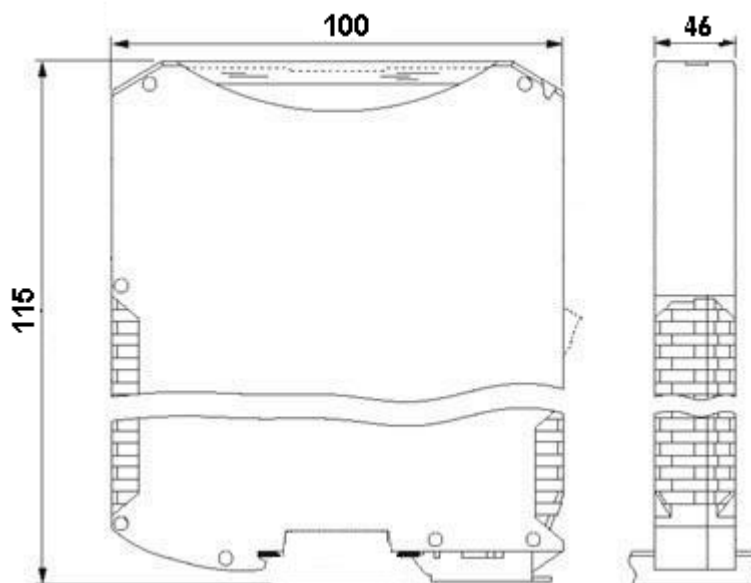
- UNIGATE®
- Connection cable from gateway to the process (Debug-interface)
- Connector for Fieldbus connection to the gateway
- Fieldbus cable (this cable is generally already installed on site!)
- 18..30 V DC power supply (DIN 19240)
- Device description file (e. g. GSD-file at PB) and operating instruction user manual, the instruction manual can be ordered separately or downloaded free of charge from our homepage at [www.deutschmann.de](http://www.deutschmann.de).

### 13.3 Installation

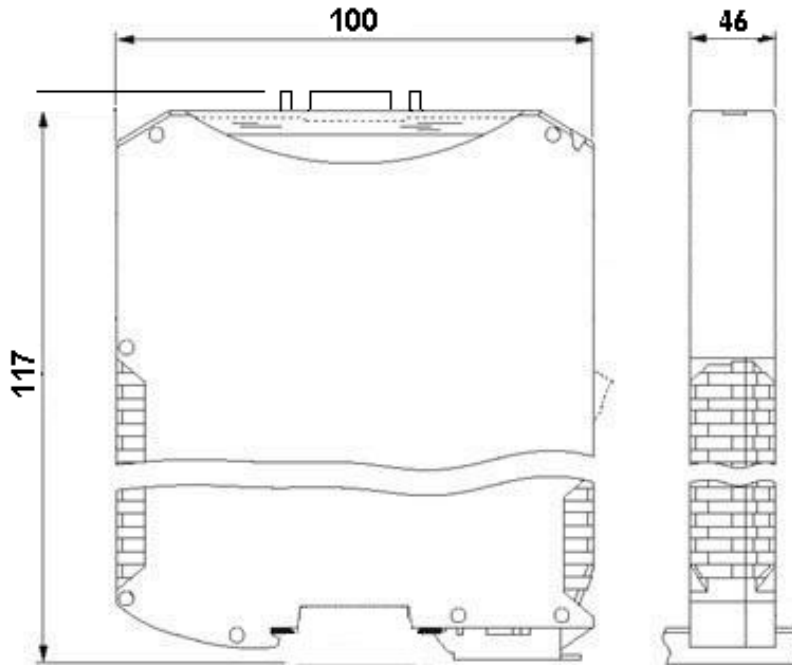
The UNIGATE® CX features protection class IP 20 and is thus suitable for switch cabinet use. The device is designed for snapping onto a 35 mm DIN-rail.

### 13.4 Dimensional drawings

#### 13.4.1 UNIGATE® CX (all versions without CANopen Slave, CANopen Master, CAN Layer 2, MPI or PROFIBUS DP)



### 13.4.2 UNIGATE® CX (all versions with CANopen Slave, CANopen Master, CAN Layer 2, MPI or PROFIBUS DP)



## 13.5 Commissioning

It is essential that you perform the following steps during commissioning in order to ensure that the module operates correctly:



#### **Attention:**

**The set Fieldbus-address must correspond to the planned address!**  
**For more information on how to adjust the Fieldbus-address please take a look at the instruction manual UNIGATE® CL for the relevant Fieldbus!**

## 13.6 Fieldbus connection

Connect the device to the Fieldbuses at the corresponding interfaces.

## 13.7 Connecting the supply voltage

Please connect 10.8...30 DC voltage to the terminals provided for this.

## 13.8 Shield connection

Earth the DIN-rail onto which the module has been snapped.

## 14 Service Interface (RS232)

Das UNIGATE® CX has 2 Service interface (RS232). These may only be used as follows:

1. Update firmware (\*.hex) file
2. Reset device

A service interface is available for each of the 2 bus sides. In order to use these, the device must be opened at the top. Of the two 7 pin. power strips located under the upper cover plate only pins 1 to 3 can be used for serial RS232 communication. The bus side where the service interface (RS232) is to be used must be started in configuration mode. (Rotary coding switch S4 + S5 = FF) The other bus side must be started in data exchange mode.



**Attention:**

*For UNIGATE CX with CANopen mapping: when updating the firmware on the CANopen side, the set protocol "Delta exchange" must first be reconfigured to the protocol "Transparent" on the 2nd Fieldbus interface. Only then the "Delta exchange" protocol is to be set again.*

- The update of a firmware (\*.hex) file must be done with the software 'Firmware Download Tool (FDT)'. Further information can be found in the help and the manual for the FDT.
- A 'Reset device' must be performed with the configuration software WINGATE®. Further information can be found in the WINGATE® manual.

**Note:** Only one of the two service interfaces (RS232) can be used at the same time.

### 14.1 Service interface (RS232) - Connection

For the wiring between the UNIGATE® and the PC COM port (RS232-USB converter), the following pin assignment must be observed.

**Attention:** Apart from the terminal assignment listed below nothing should be connected. The exception is the power supply for the UNIGATE®.

UNIGATE®	PC COM-Port (9pin. D-Sub connector)
Rx232 / RX	COM-Port Pin 3 = Tx
Tx232 / Tx	COM-Port Pin 2 = Rx
APGND / GND / 0V (RS)	COM-Port Pin 5 = GND

**Attention:** The connection between GND or 0V and the GND of the PC COM port is mandatory!

Pin assignment of the two service interfaces:

UNIGATE® CX	Service interface (RS232) - Bus A
Pin 1 = Rx232	
Pin 2 = Tx232	
Pin 3 = AP-GND	
Pin 4 = n.c.	
Pin 5 = n.c.	
Pin 6 = n.c.	
Pin 7 = n.c.	

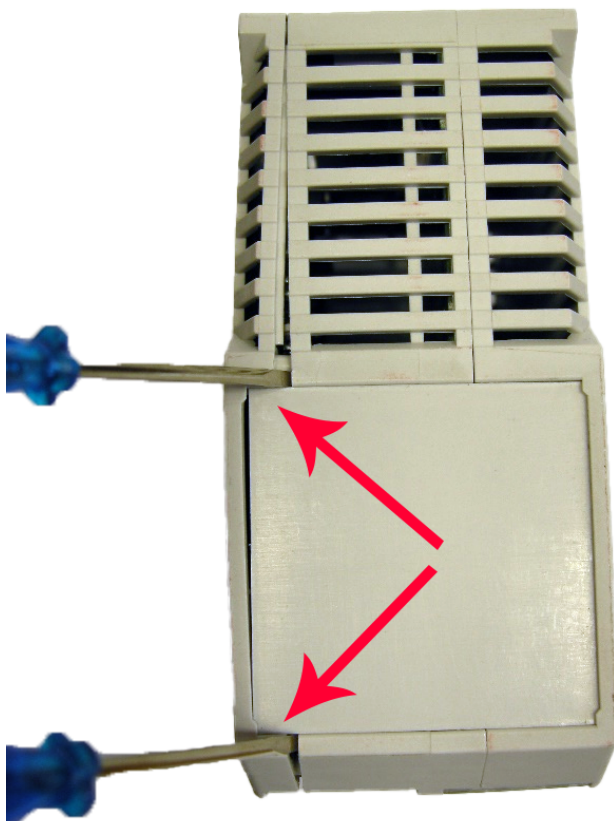
UNIGATE® CX	Service interface (RS232) - Bus B
Pin 1 = Rx232	
Pin 2 = Tx232	
Pin 3 = AP-GND	
Pin 4 = n.c.	
Pin 5 = n.c.	
Pin 6 = n.c.	
Pin 7 = n.c.	

## 14.2 Service interface (RS232) – Access

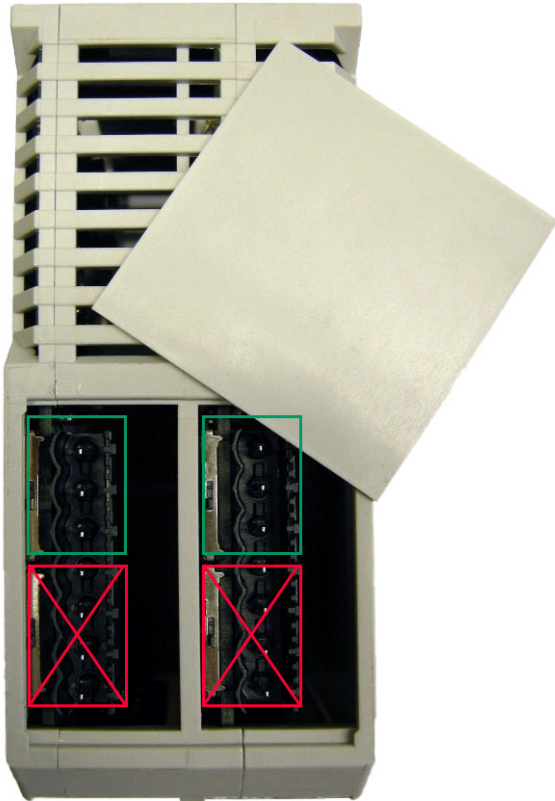
1. Top of UNIGATE® CX



2. The housing can be opened by hand or with a screwdriver. Then the upper cover plate can be removed.



3. There are two 7-pin power strips located under the upper cover plate. Of those only pins 1 to 3 can be used for serial RS232 communication. A service interface (RS232) is available for each bus side.



4. Located on the left side (seen in installation position) you find the service interface of the bus first mentioned in the device name.

The bus side of the connected service interface must be started in configuration mode.  
The other bus side must be started in data exchange mode.

5. Located on the right side (seen in installation position) you find the service interface of the bus mentioned second in the device name.

The bus side of the connected service interface must be started in configuration mode.  
The other bus side must be started in data exchange mode.

## 15 Servicing

Should questions arise that are not covered in this manual you can find further information in our

- FAQ/Wiki area on our homepage [www.deutschmann.com](http://www.deutschmann.com) or directly in our Wiki on [www.wiki.deutschmann.de](http://www.wiki.deutschmann.de)

If your questions are still unanswered please contact us directly.

**Please note down the following information before calling:**

- Device designation
- Serial number (S/N)
- Article number
- Error number and error description

Your request will be recorded in the Support center and will be processed by our Support Team as quickly as possible (Usually in 1 working day, rarely more than 3 working days.).

The technical support hours:

Monday to Thursday from 8 am to midday and from 1 pm to 4 pm, Friday from 8 am to midday. (CET)

Deutschmann Automation GmbH & Co. KG  
Carl-Zeiss-Straße 8  
D-65520 Bad-Camberg  
Germany

Central office & sales	+49 6434 9433-0
Technical support	+49 6434 9433-33

Fax sales department	+49 6434 9433-40
Fax technical support	+49 6434 9433-44

E-mail technical support	<a href="mailto:support@deutschmann.de">support@deutschmann.de</a>
--------------------------	--

### 15.1 Returning a device

If you return a device, we require as comprehensive a fault/error description as possible. We require the following information in particular:

- What error number was displayed?
- What is the supply voltage ( $\pm 0.5$  V) with gateway connected?
- What were you last doing or what last happened on the device (programming, error on power-up, ...)?

The more precise information a fault/error description you provide, the more exactly we will be able to pinpoint the possible causes.

### 15.2 Downloading PC software

You can download current information and software free of charge from our Internet server.  
<http://www.deutschmann.com>

## 16 Annex

### 16.1 Explanations of the abbreviations

#### General

CL	=	Product group CL (Compact Line)
CX	=	Product group CX
GT	=	Galvanic separation RS-side
GY	=	Housing color gray
RS	=	Product group RS
SC	=	Product group SC (Script)
232/485	=	Interface RS232 and RS485 switchable
232/422	=	Interface RS232 and RS422 switchable
DB	=	Additional RS232 DEBUG-interface
D9	=	Connection of the RS through 9-pin D-SUB instead of 5-pin screw-plug connector
PL	=	Board only without DIN-rail module and without housing cover
PD	=	Board only without DIN-rail module and with housing cover
AG	=	Gateway installed in a die-cast aluminum housing
EG	=	Gateway installed in a stainless steel housing
IC	=	Product group IC (IC-design DIL32)
IC2	=	Product group IC2 (IC-design DIL32)
16	=	Script memory expanded to 16KB
5V	=	Operating voltage 5V
3,3V	=	Operating voltage 3.3V

#### Fieldbus

CO	=	CANopen
C4	=	CANopen V4
C4X	=	CANopen V4-version X (see comparison table UNIGATE® IC for the respective product)
DN	=	DeviceNet
EC	=	EtherCAT
EI	=	Ethernet/IP
FE	=	Ethernet 10/100 MBit
FEX	=	Ethernet 10/100 MBit-version X (see comparison table UNIGATE® IC for the respective product)
IB	=	Interbus
IBL	=	Interbus
LN62	=	LONWorks62
LN512	=	LONWorks512
MPI	=	Siemens MPI®
PN	=	PROFINET-IO
PBDP	=	PROFIBUS DP
PBDPL	=	PROFIBUS DP-version L (see comparison table UNIGATE® IC for the respective product)
PBDPX	=	PROFIBUS DP-version X (see comparison table UNIGATE® IC for the respective product)
PBDPV0	=	PROFIBUS DPV0
PBDPV1	=	PROFIBUS DPV1
RS	=	Serial RS232/485/422



## 16.2 Hexadecimal table

Hex	Decimal	Binary
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	5	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
A	10	1010
B	11	1011
C	12	1100
D	13	1101
E	14	1110
F	15	1111