Deutschmann

your ticket to all buses

Instruction Manual
Universal Fieldbus-Gateway
UNIGATE® CL - CANopen

Deutschmann Automation GmbH & Co. KG
www.deutschmann.com | wiki.deutschmann.de

Manual Art.-No.: V3608E

EDisclaimer 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.

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 harmonized 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-Straße 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 manufacturers

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.

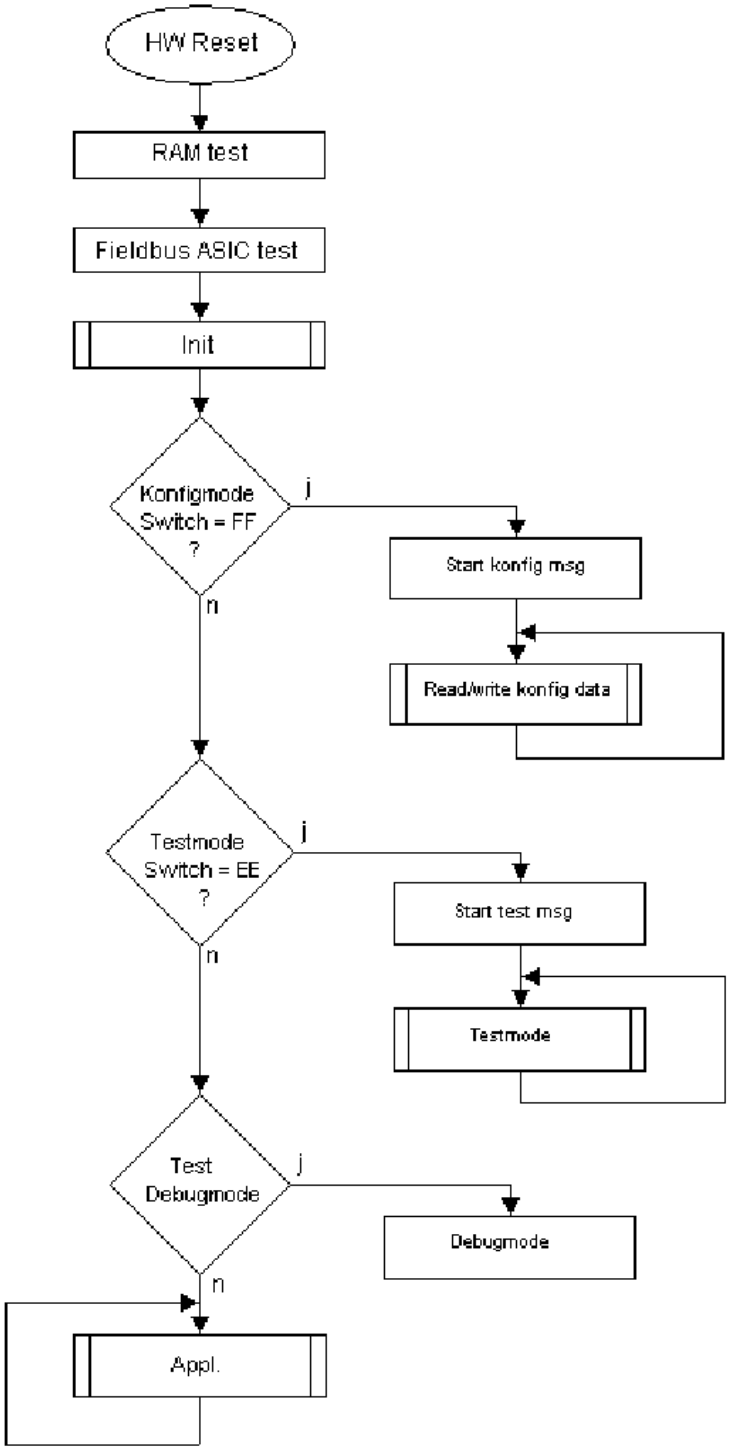
3 Introduction

The UNIGATE® CL-CANopen® module serves to adapt a serial port to CANopen®. In this application, it functions as a Gateway and operates as CANopen® Slave. It can be operated by any standard-compliant Master.

The module CL-CANopen® essentially consists of the following hardware components:

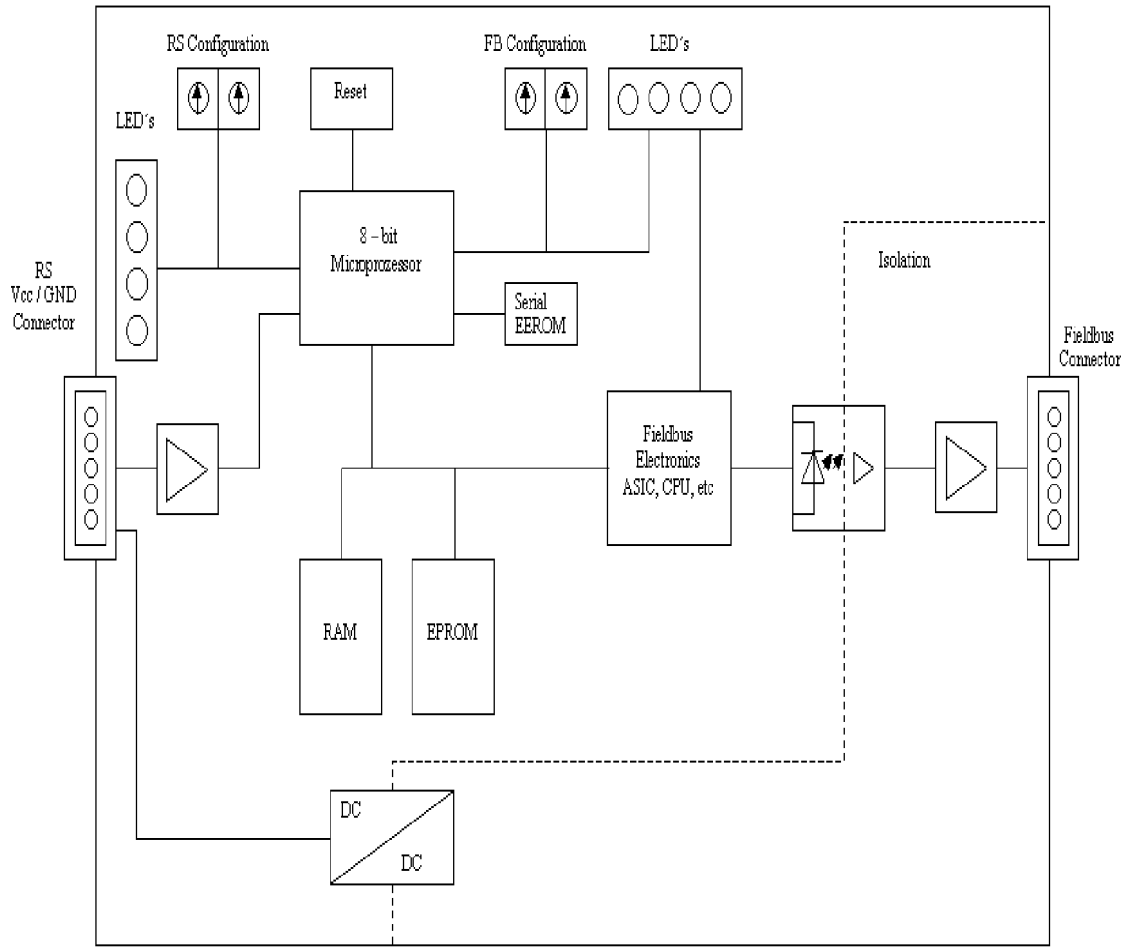
- Electrically isolated interface to CANopen®
- CAN-controller SJA 1000
- Microprocessor 89C51RD2
- RAM and EPROM
- Optionally electrically isolated RS-interface
- Serial interface (RS232, RS485 and RS422) to the device connected externally

3.1 UNIGATE® CL software flow-chart



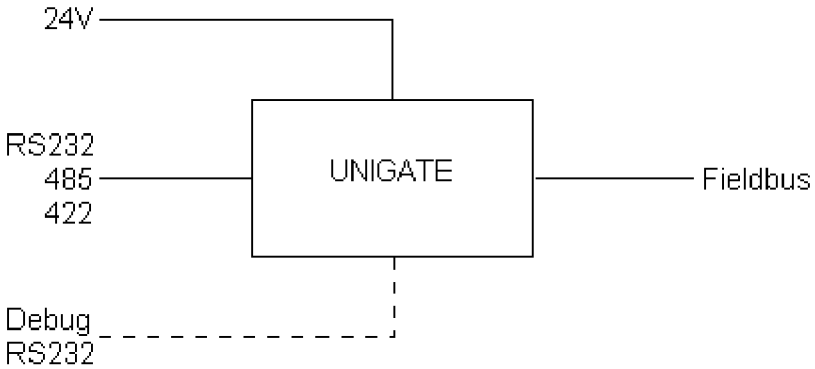
3.2 UNIGATE® block diagram

The following picture shows a typical UNIGATE®-module design.



3.3 UNIGATE® application diagram

The following graph shows a typical connection scheme.



4 Operation modes of the Gateway

4.1 Configuration mode (config mode)

The configuration mode serves to configure the Gateway. The following adjustments are possible in this mode.

- Loading a Script
- Updating the firmware
- Configuring the Gateway

The Gateway will be starting in this mode in case both switches S4 as well as S5 are set on position "F" when switching on the Gateway. Right after switching on the Gateway in the configuration mode it will be sending its starting message, that looks analog with the following message: "RS-COV4-CL (232/422/485) V2.0 [29] (c)dA Switch=0xC1FF Script(8k)="Leer" Author="Deutschmann Automation GmbH" Version="1.0" Date=21.08.2001 SN=47110001“.

In the configuration mode the Gateway always operates with the settings 9600 Bauds, no Parity, 8 databits and 1 stopbit, the RS-State LED will always be flashing red, the "Error No/Select ID" LEDs are of no account for the user. All software revisions contain the configuration mode.

4.2 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, for instance to test the parameters of the RS-interfaces.

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 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.

4.3 Data exchange mode

The Gateway has to be in the data exchange mode, so that a data exchange between the RS-side of the Gateway and the fieldbus is possible. As long as the Gateway is not in the configuration mode or the test mode, the data exchange mode is active. In the data exchange mode the Gateway will execute the downloaded Script.

5 RS-interface

5.1 RS-interfaces at the UNIGATE® CL

The UNIGATE® CL - CANopen® has the interfaces RS232, RS422 and RS485 available. The hardware always features a DEBUG-interface, see chapter 7.

5.2 Buffer sizes at the UNIGATE® CL

UNIGATE® CL features at the serial side a buffer with the size of 1024 bytes for input data and output data each.
The FIFO of the application interface (RS-interface) can be changed in any Gateway form Script revision 26 on, that is capable for Script. For it please check in the Protocol Developer under "Device Control" - "Hardware".

5.3 Framing Check

The length of the stop bit received by the Gateway is checked through the function "Framing Check". Here the stop bit generated by the Gateway is always long enough, so that connected participants can evaluate the stop bit.

Please be aware that the function "Framing Check" becomes effective only in case of 8 data bit and the setting "No parity".

An error is detected and indicated by the Error LEDs in case the stop bit does not show the length 1 bit during the activated check.

The possible setting for this parameter can be controlled by the Script (see online help from Protocol Developer). The presetting for the "Stop Bit Framing Check" is "enabled".

6 SSI-interface

The UNIGATE® also supports the connection of applications or products, that communicate via SSI.

6.1 Initiation of the SSI-interface

The configuration of the SSI-interface is executed in the config mode with the WINGATE software, Protocol SSI. The encoder type and the sampling frequency are defined via the parameter "Resolution" (1 bit..15 bit, 24 bit...25 bit), "SSI Encoder Type" (Binary or Gray code) and "Clock stretch".

6.2 Parameter

6.2.1 Resolution

The range extends from 1 bit to 25 bits. This enables single-turn SSI encoders and multi-turn SSI encoders to be configured.

6.2.2 SSI Encoder Type

This can be selected between binary and gray code.

6.2.3 Parameter sample frequency (Clock stretch)

You can change the sampling frequency. For this purpose a "Stretch value" is passed that inserts a waiting period after each clock edge.
If a 0 is passed, there is no waiting time.
Thus the following SSI sample frequencies may vary slightly:

Waiting time = 0 → SSI-Clock ~ 333kHz (No Stretch)
Waiting time = 1 → SSI-Clock ~ 185kHz
Waiting time = 2 → SSI-Clock ~ 150kHz
Waiting time = 3 → SSI-Clock ~ 125kHz
Waiting time = 4 → SSI-Clock ~ 110kHz
Waiting time = 5 → SSI-Clock ~ 100kHz
Waiting time = 6 → SSI-Clock ~ 88kHz
Waiting time = 7 → SSI-Clock ~ 80kHz
Waiting time = 8 → SSI-Clock ~ 72kHz
Waiting time = 9 → SSI-Clock ~ 67kHz
Waiting time = A → SSI-Clock ~ 62kHz
Waiting time = B → SSI-Clock ~ 58kHz
Waiting time = C → SSI-Clock ~ 54kHz
Waiting time = D → SSI-Clock ~ 50kHz
Waiting time = E → SSI-Clock ~ 48kHz
Waiting time = F → SSI-Clock ~ 45kHz

The bit time from which these frequencies were derived, calculate as follows:
t = 3µs + (2* (+ 0.6µs (n* 0.6µs))), where n corresponds to the "Stretch value" (1.. F).
Without clock extension (n = 0) remains at 3µs → 333kHz!
The max. Bit length of 32 bits and the slowest clock this results in a total readout time of 32 * = 22µs ~ 700µs.

6.2.4 Parameter Encoder monitoring (Check Encoder)

An encoder monitoring can be activated via the parameter "Check encoder", as long as the used SSI-encoder supports this function. After the last read encoder bit it is verified if the data line is still at Low for at least one bit. If the UNIGATE® does NOT detect this bit on Low, error 12 is issued. For example it can detect a cable break or a not connected encoder. However, it can also be a misconfigured bit length, or a too slow read out clock.

6.3 Hardware-wiring

The clock wires of the SSI-interface are placed onto the Tx-wires of the RS422-interface and the data wires onto the Rx-wires at the UNIGATE®.

X1 (3pin + 4pin screw-plug-connector):

Pin no.	Name	Function at SSI
1	Rx 232	n. c.
2	Tx 232	n. c.
3	AP-GND	n. c.
4	Rx 422+	SSI DAT+
5	Rx 422-	SSI DAT-
6	Tx 422+	SSI CLK+
7	Tx 422-	SSI CLK-

7 The Debug-interface

7.1 Overview of the Debug-interface

The UNIGATE® features a Debug-interface, that allows a step-by-step processing of a Script. Normally this interface is only required for the development of a Script.

7.2 Starting in the Debug-mode

When applying power to the UNIGATE® (power up) the firmware will output the binary character 0 (0x00) after a self-test was carried out on this interface. If the UNIGATE® receives an acknowledgement via this interface within 500 ms, it is in the Debug-mode. The acknowledgement is the ASCII-character O (0x4F).
With the start in the Debug-mode the further execution of Script commands will be put to a stop.

7.3 Communication parameter for the Debug-interface

The Debug-interface is always operating with 9600 baud, no parity, 8 data bit, 1 stop bit. It is not possible to change this parameter in the Protocol Developer. Please consider the fact that these settings have to be in accordance with those of the PC-COM-interface and that the flow control (protocol) has to be set on „none“ there.

7.4 Possibilities with the Debug-interface

Usually the Protocol Developer is connected to the Debug-interface. With it a step-by-step processing of a Script, monitoring jumps and decisions and looking at memory areas is possible. Moreover breakpoints can be set. It basically possesses all characteristics a software-development tool is typically supposed to have. However, it is also possible to carry out a Scrip-update via this interface.
From Script version [27] on you can also output data with the Script command "`SerialOutputToDebugInterface`". Please also pay attention to the remark in the manual ’Protocol Developer’.

7.5 Commands of the Debug-interface

The commands for the use of the Debug-interface are described in the instruction manual Protocol Developer.

8 Mode of operation of the system

8.1 General explanation

Communication can be split into seven layers, Layer 1 to Layer 7, in accordance with the ISO/OSI model.
The Deutschmann Automation Gateways convert Layers 1 and 2 of the customized bus system (RS485 / RS232 / RS422) to the corresponding Fieldbus system. Layers 3 to 6 are blank, and Layer 7 is converted in accordance with chapter 8.3.

8.2 Interfaces

The Gateway features the RS232-, RS422- and RS485-interfaces.

8.3 Data exchange CANopen® V3

All data is transferred by the Gateway in dependence of the downloaded Script.
The following three objects are existing in the CANopen®-gateway for the data exchange on the CANopen®-side:

- Default setting as long as the Script command CO_Init_Channel is not carried out.
  - Adr. 2000H (Type DOMAIN):Data received by the gateway
  - Adr. 2001H (Type DOMAIN):Data sent by the gateway
  - Adr. 2002H (Type BYTE): Length of the data sent

The length of the receiving- and transmitting buffer (Obj. 2000 + 2001) is configured through WINGATE®.

8.3.1 SDO-access

Generally the data can always be exchanged through SDOs (Obj. 2000 - 2002).
Likewise an access to all Mandatory-objects according to CiA® DS 301 is possible through SDOs.

8.3.2 PDO-access

PDOs are supported according to the following table depending on the configured length and the PDO-length is set dynamically to the correct value:

Gateway receiving-data	**Gateway transmitting-data**	**Receiv.-PDO1 (Adr = 512 + ID)**	**Transm.-PDO1 (Adr = 384 + ID)**
Max. 8 Byte	Max. 8 Byte	Receiv. data	Transm. data
Max. 8 Byte	>8 Byte	Receiv. data	Length transmitting data
>8 Byte	Max. 8 Byte	-	Transm. data
>8 Byte	>8 Byte	-	Length transmitting data

8.4 Possible data lengths

The table below shows the maximum transferable data in CANopen®:.

Input data	max. 255 bytes	Variable: maximum value in this case
Output data	max. 255 bytes	Variable: maximum value in this case
Emergency data	1 byte	See chapter Error handling

9 Generating a Script

9.1 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.

9.2 Memory efficiency of the programs

A Script command can carry out e. g. a complex checksum like a CRC-16 calculation via data. For the coding of this command only 9 byte are required as memory space (for the command itself). This is only possible when these complex commands are contained in a library.
A further advantage of this library is, that the underlying functions have been in practical use for a couple of years and therefore can be described as 'void of errors'. As these commands are also present in the native code for the controller, at this point also the runtime performance of the Script is favorable.

9.3 What can you do with a Script device?

Our Script devices are in the position to process a lot of commands. In this case a command is always a small firmly outlined task. All commands can be put into classes or groups. A group of commands deals with the communication in general. This group's commands enable the Gateway to send and receive data on the serial side as well as on the bus-side.

9.4 Independence of buses

Basically the Scripts do not depend on the bus, they are supposed to operate on. It means that a Script which was developed on a PROFIBUS Gateway can also be operated on an Interbus without changes, since the functioning of these buses is very similar. In order to also process this Script on an Ethernet Gateway, perhaps further adjustments have to be made in the Script, so that the Script can be executed reasonably.
There are no fixed rules how which Scripts have to operate properly. When writing a Script you should take into account on which target hardware the Script is to be executed, so the necessary settings for the respective buses can be made.

9.5 Further settings at the Gateway

Most devices require no further adjustments, except for those made in the Script itself. However, there are also exceptions to it. These settings are made by means of the software WINGATE. If you know our UNIGATE®-series, you are already familiar with the proceeding with it. An example is the adjustment of the IP-address and the net-mask of an Ethernet-Gateway. These values have to be known as fixed values and are not available for the runtime. Another reason for the

configuration of the values in WINGATE is the following: After an update of the Script these values remain untouched, i. e. the settings that were made once are still available after a change of the Script.
Only this way it is also possible that the same Script operates on different Ethernet-Gateways, that feature different IP-addresses.

9.6 The use of the Protocol Developer

The Protocol Developer is a tool for an easy generation of a Script for our Script Gateways. Its operation is exactly aimed at this use. After starting the program the Script that was loaded the last time is loaded again, provided that it is not the first start.
Typical for Windows Script commands can be added by means of the mouse or the keyboard. As far as defined and required for the corresponding command, the dialog to the corresponding command is displayed, and after entering the values the right text is automatically added to the Script. The insertion of new commands by the Protocol Developer is carried out in a way that existing commands will not be overwritten. Generally a new command is inserted in front of the one where the cursor is positioned. Of course the commands can also be written by means of the keyboard or already written commands can also be modified.

9.7 Accuracies of the baud rates

The baud rate of the serial interface is derived from the processor's crystal frequency.
Meanwhile all Script-Gateways are working with a crystal frequency of 40 MHz.
You can enter any desired integer baud rate into the Script. After that the firmware adjusts the baud rate, that can be derived the most precisely from the crystal frequency.
The baud rate the Gateway is actually working with (BaudIst) can be determined as follows:

BaudIst= (F32 / K)
F32 = Crystal frequency [Hz] / 32
K = Round (F32 / BaudSoll);
Round () is a commercial roundoff

Example:
The actual baud rate is to be calculated, when 9600 baud are pre-set, where the Gateway is operated with 40 MHz:

F32 = 40000000 / 32 = 1250000
K = Round(1250000 / 9600) = Round(130.208) = 130
BaudIst= 1250000 / 130 = 9615.38

I. e.: The baud rate actually adjusted by the Gateway is 9615.38 baud

The resulting error in per cent can be calculated as follows:

Error[%] = (abs(BaudIst - BaudSoll) / BaudSoll) * 100
In our example the following error results:

Error = (abs(9615.38 - 9600) / 9600) * 100 = 0.16%

In practise errors below 2% can be tolerated!

In the following please find a listing of baud rates at a 40 MHz-crystal frequency with the corresponding errors:

4800 baud:	0.16%
9600 baud:	0.16%
19200 baud:	0.16%
38400 baud:	1.35%
57600 baud:	1.35%
62500 baud:	0%
115200 baud:	1.35%
312500 baud:	0%
625000 baud:	0%

9.8 Script processing times

The Script is translated by the Protocol Developer and the consequently generated code is loaded into the Gateway. Now the processor in the Gateway interprets this code. In this case, there are commands that can be processed very fast (e. g. "`Set Parameter`"). There are also commands, however, that take longer (e. g. copying 1000 bytes). Consequently, for one thing the processing time differs due to the kind of Sript command. But the processing time of the Script commands is considerably more determined by the processor time that is available for this process. Since the processor has to carry out several tasks simultaneously (multitasking system) only a part of the processor's capacity is available for the Script processing. The following tasks - in the order of priority - are executed on the processor:

- Sending and receiving data at the Debug-interface (provided that the Protocol Developer has been started on the PC)
- Sending and receiving data at the RS-interface
- Sending and receiving data at the Fieldbus-interface
- Tasks controlled via internal clock (1 ms) (e. g. flashing of an LED)
- Processing of the Script

From experience approximately 0.5 ms can be calculated for each Script line. This value confirmed itself again and again in many projects as a standard value. He is always quite right if the processor has enough time available for the Script processing.
By means of the tasks mentioned above, the following recommendation can be formulated in order to receive a rather fast Script processing:

- Deactivate the Debug-interface (it is the normal case in the serial use)
- Keep the data length at the RS-interface as small as possible. The baud rate is not the problem here, but the amount of characters which are transfered per second.

- Do not unnecessarily extend the data length at the Fieldbus side. Especially at acyclical bus data, if possible do only send them when changes were made. The data length at buses that are configured to a fixed length (e. g. PROFIBUS) should not be longer than absolutely necessary.

If the processing time should be too large in spite of these measures, there is the possibility to generate a customized Script command, that executes several tasks in one Script command. Please contact our support department for this purpose.

10 Implemented protocols with Universal Script - Configuration

UNIGATE® CL is supplied with the Script “Universal Script Deutschmann“. The configuration of the protocols is carried out by means of the software WINGATE. See also "Instructions UNIGATE® CL - Configuration with WINGATE". The PDF can also be found on our website under Support/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.
If you no longer have the compiled script, a corresponding request must be sent to Deutschmann Support.
https://www.deutschmann.de/en/support/enquiry/

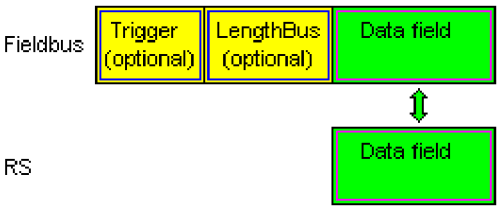
11 Implemented protocols with Universal Script - protocols application

Below you will find the descriptions of the implemented protocols that can be used for the application side.

11.1 Protocol: Transparent

The data is transferred bidirectional from the UNIGATE®.

11.1.1 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.

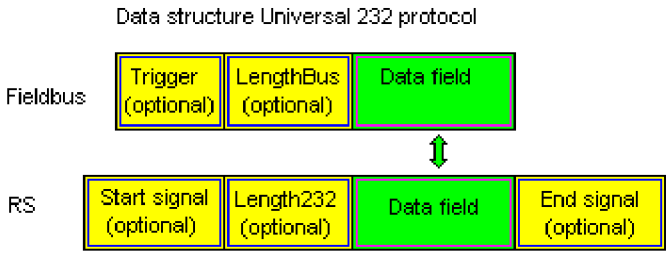
If less data is received through Rx then configured by the GSD-file (I/O-length), then the rest is complemented with ZERO.
Too much data received will be cut off.

Depending on the fieldbus, the required length of the input and output data (I / O length) can be set via the device configuration of the UNIGATE or via the device description file in the higher-level controller.

11.2 Protocol: Universal 232

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

11.2.1 Data structure



11.2.2 Fieldbus parameters

Trigger byte: See „The trigger byte“, chapter 14
Length byte: See „The length byte“, chapter 15

11.2.3 RS232 parameter table

11.2.3.1 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.

11.2.3.2 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.

11.2.3.3 Timeout

If the end character is set to "FF", the value that was set in the RX_Timeout parameter is activated and the time entered there is waited for with serial reception, or triggered for new characters. If the set time is exceeded without an event, the end criterion is reached and the characters are copied onto the bus.

11.2.3.4 Data domain

The operating data is transferred to this field.

11.2.3.5 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.

11.2.4 Communication sequence

The useful data (data area) arriving via the fieldbus is copied in accordance with chapter 11.2.1 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 !

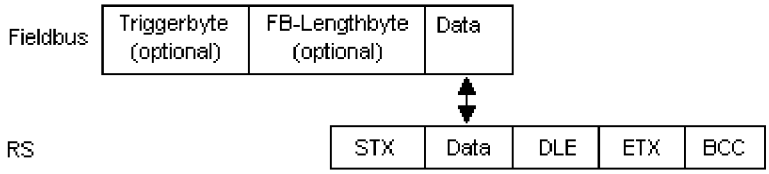
If the "Trigger byte“ (see chapter 14) is active, data is sent only on a change of this byte. If the "Length byte" (see chapter 15) is active, only as many of the following bytes as specified there are transferred.

Receive data at the RS interface is evaluated in accordance with the configured protocol, and the data field (data area (see chapter 11.2.1)) 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.

11.3 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.

11.3.1 Data structure 3964R



11.3.2 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.

11.3.3 Data communication

11.3.3.1 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.

11.3.4 Conflicts

11.3.4.1 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.

11.3.4.2 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.

11.3.4.3 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.

11.3.5 Protocol type 3964

The difference to protocol type 3964R is:
1. tq = acknowledge monitoring time
2. The checksum byte BCC is missing.

11.4 Protocol: MODBUS-RTU

11.4.1 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.

11.4.2 UNIGATE® as MODBUS-Master

11.4.2.1 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.

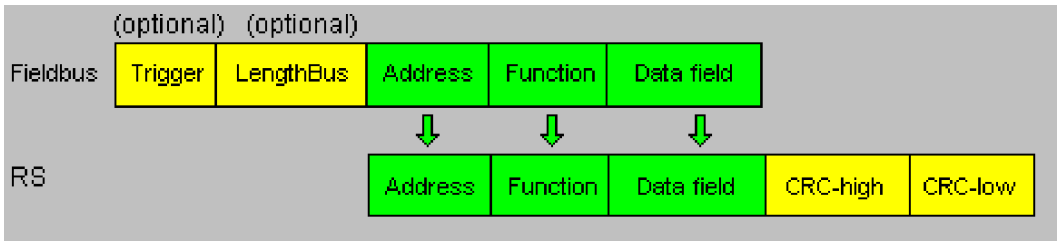
Since the Modbus operates with a variable data format - dependent on the required function and data length - but since the fieldbus requires a fixed data length, this must be preset via the configuration with WINGATE (blocklength fieldbus input and blocklength fieldbus output). This length should be 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 an event-driven way (On Event) or on request (On Trigger).

The mode "Modbus request on demand" necessitates the first byte in the fieldbus containing a trigger byte (see chapter 14). 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 15), 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 "Dat n" (inclusive in each case), always without CRC checksum.

11.4.2.2 Data structure



11.4.2.3 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".

11.4.3 UNIGATE® as MODBUS-Slave

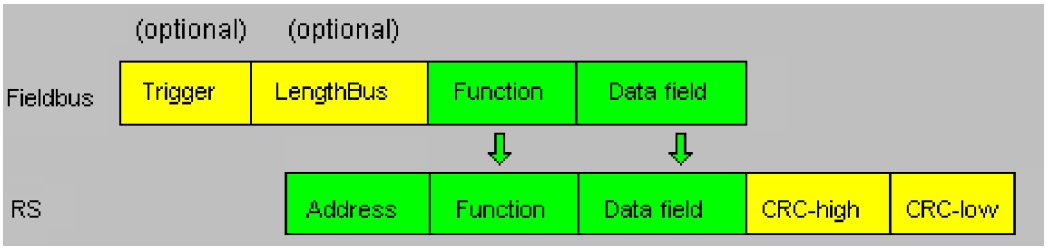
11.4.3.1 Preparation

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 via the configuration with WINGATE (blocklength fieldbus input and blocklength fieldbus output). This length should be selected by the user such that the longest Modbus request resp. response can be processed.

11.4.3.2 Data structure



11.4.3.3 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.

11.4.4 UNIGATE® as Modbus-ASCII Master

-> For the description see chapter 11.4.2 "UNIGATE® as MODBUS-Master"

11.5 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).

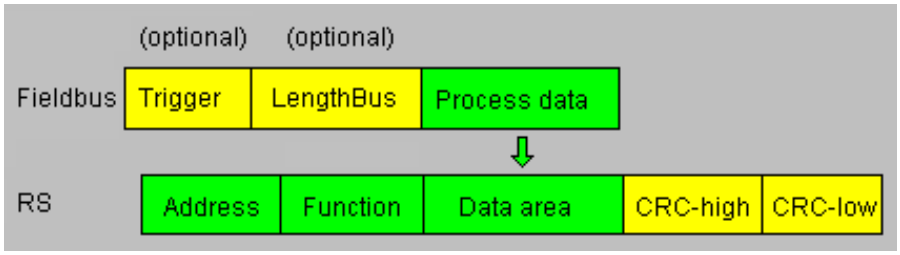
11.5.1 Data structure on the fieldbus side e.g.: PROFIBUS

Applies to In and Out

1. Byte: trigger byte, optional (see chapter 14, The trigger byte)
2. Byte: fieldbus length byte, optional (see chapter 15, The length byte)
3. Byte: process data
4. Byte: process data

....

Data structure



11.5.1.1 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]

11.5.1.2 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 additonal 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.

11.5.1.3 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...

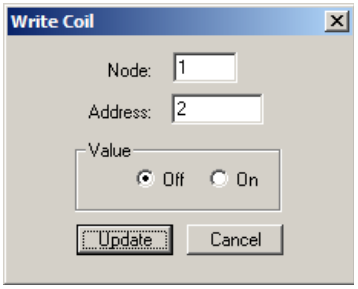
1. Byte = Fieldbus length byte

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 :

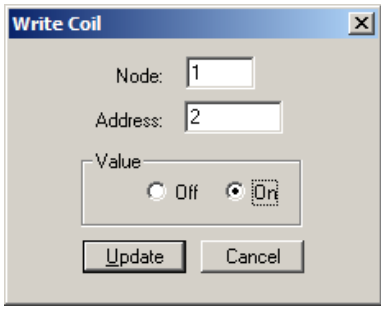


Address 0002 stays unchanged on 0, however, the fieldbus output data is updated.
After a reset they are NULL (1st row) at first and are then updated (2nd row):
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ...

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 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 than1024 byte.

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



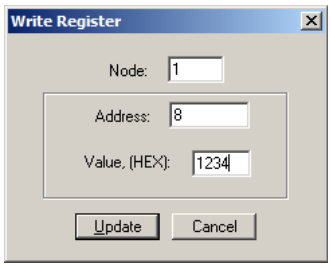
The fieldbus data is updated:
1F 03 02 03 04 05 06 07 00 00 00 00 00

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>
```

11.5.1.4 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 00 12 34 00 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 ….

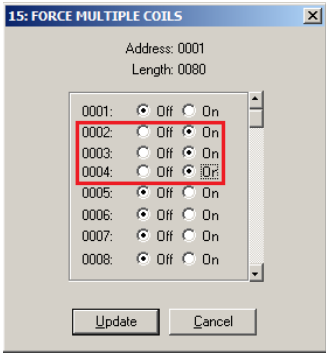
| +---- 1. process value
+-------- fieldbus length byte

11.5.1.5 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.

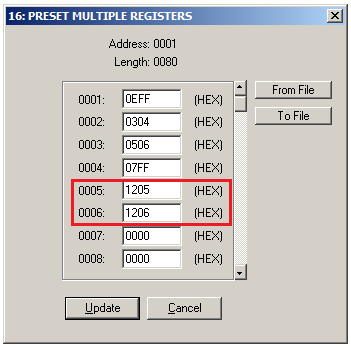
Adr 0002 ... 004 was changed from Low to High



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 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.

11.5.1.6 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 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...
The 2nd row shows the fieldbus data content AFTER the update.

11.6 Protocol „Universal Modbus RTU Master“

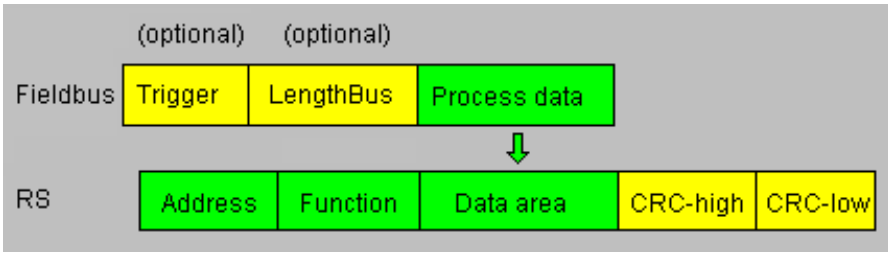
The UNIGATE® is Modbus-Master on the Application side.

11.6.1 Data structure Fieldbus side (e.g. PROFIBUS):

Applies to In and Out

1. Byte: Trigger-Byte, optional (see chapter 14, The trigger byte)
2. Byte: Fieldbus length byte, optional (see chapter 15, The length byte)
3. Process data

Data structure



11.6.2 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

11.6.3 Configuration: via Wingate since wcf Datei Version 396

Parameter Name	value range	Explanation
Modbus Timeout (10ms)	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.
RX Poll Retry		Retry of the last, invalid replied "Request"
RX Poll Delay (10ms)		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.

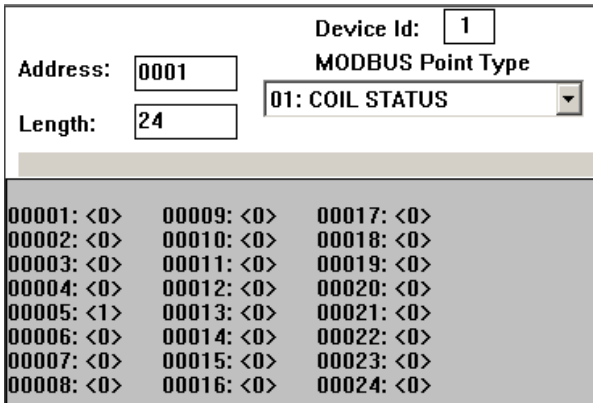
„(10ms)“ : adjustable in 10ms steps
„(hex)“: Enter in hexadecimal style.
„(dec)“: Enter in decimal style.
„(Byte)“: Counting in bytes, starting at the position Null. Attention: For read commands, e.g. FC3, after the trigger- and lenghtbyte the first process value is the postion nulll, which is copied to the fieldbus to the PLC.
For write commands, e.g. FC16, the position Null is the trigger byte.

11.6.3.1 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



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 01 00 00 00 00 00 00 00 00 00 00 00 0

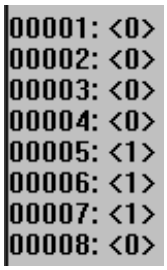
1. Byte = Trigger byte (value = 0x66)
2. Byte = Fieldbus length byte (value = 0x07)
3. Byte = Fieldbus Map Adr 0 (value = 0x00)
4. Byte = Fieldbus Map Adr 1 (value = 0x00)
5. Byte = Fieldbus Map Adr 2 (value = 0x00)
6. Byte = Fieldbus Map Adr 3 (value = 0x00)
7. Byte = Fieldbus Map Adr 4 (value = 0x00)
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.

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

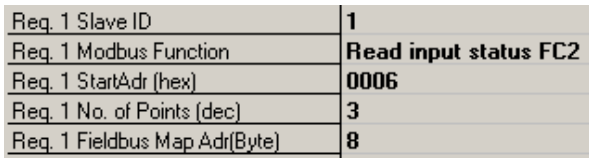
A modification of address 7 in the Modbus slave has no consequences to the fieldbus output side because "No. Of Points = 2" is set in the configuration.

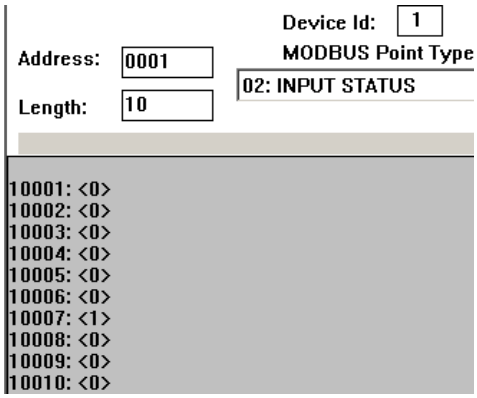


The value stays unchanged on 0x03:
1F 07 00 00 00 00 00 03 00 00 00 0

11.6.3.2 Example: Read input status FC2

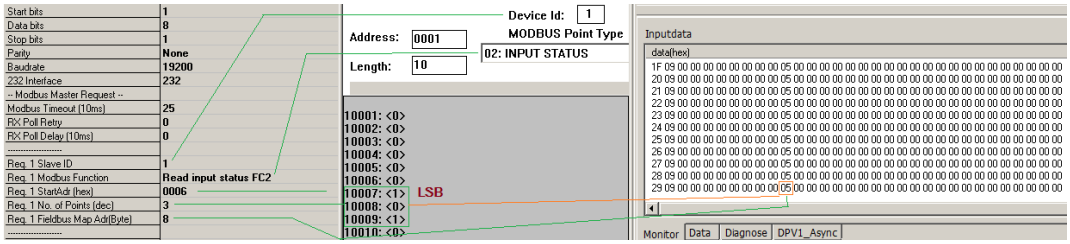
The following example shows the content of address 10007 ... 10009 is mapped/copied into the 8. fieldbus output byte.





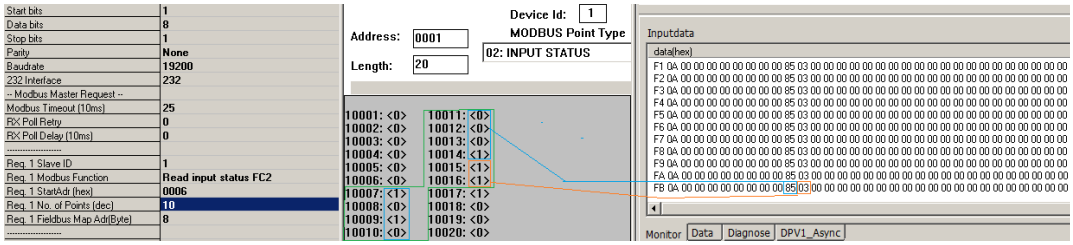
76 09 00 00 00 00 00 00 00 00 01 00 00 00 00

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.



11.6.3.3 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 "**4**0000" stands for "holding register". But in acutality 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“

11.6.3.4 Example: Read input registers FC4

(see chapter 11.6.3.3, Example: Read multiple register FC3)

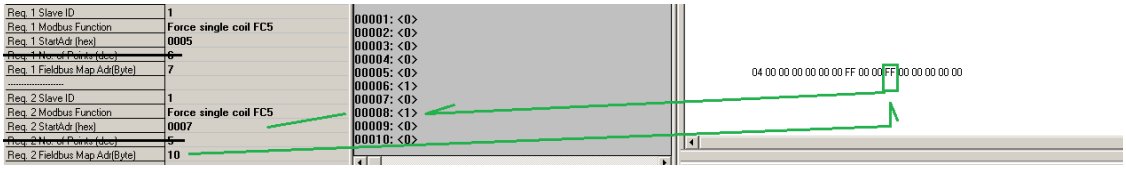
11.6.3.5 Example: Force single coil FC5

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)

Note: No. of Points is not required

Another example for when a second request is configured:



11.6.3.6 Example: Preset single register FC6

Configuration

1 Slave ID	1
1 Modbus Function	Preset single register FC6
1 StartAdr (hex)	0005
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: <FF23H>
40007: <0000H>
40008: <0000H>
40009: <0000H>
40010: <0000H>

11.6.3.7 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 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 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	||||||||	00000|0|
Position	8 7 6 5 4 3 2 1	11 10 9

Please keep in mind that No. Of coild = 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.

11.6.3.8 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>

11.7 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 11.5 "Protocol „Universal Modbus RTU Slave“" respectively see chapter 11.6 "Protocol „Universal Modbus RTU Master“".

11.8 Delta exchange protocol

The Delta exchange protocol can only be used since version 1.9.0 of the universal script Deutschmann.

With this protocol, data can be specifically copied to an area on the fieldbus output side, i.e. from the UNIGATE® to the PLC.
To do this, two control parameters, the address offset and the number of bytes, are required at the start of the transfer before the actual user data then follows. First comes the address offset (Word), then the number of bytes (Word), then the user data as a byte array.
The two parameters (address offset and number of bytes) must be sent with every data transfer.

1st byte address offset high
2nd byte address offset low
3rd byte number of bytes high
4th byte number of bates low
5th byte user data 1st byte
6th byte "2nd byte
7. "...

Example:

1st byte 0x00
2nd byte 0x02
3rd byte 0x00
4th byte 0x05
5th byte 0x01
6th byte 0x02
7th byte 0x03
8th byte 0x04
9th byte 0x05

In the fieldbus output data (e.g. PROFIBUS), the data is sent to the PLC as follows
sent: 0x00 0x00 0x01 0x02 0x03 0x04 0x05 0x00 ...

11.9 Protocol SSI

With the SSI protocol, e.g. SSI encoders are evaluated with the UNIGATE® and this information is forwarded to the higher-level controller. Parameters can be used to configure the encoder type, the encoder resolution, the clock frequency and an ERROR bit (if supported) according to the SSI encoder used. See also chapter 6 (SSI-interface).

12 Implemented Protocols with Universal Script - Protocols CANopen/CAN Layer 2

Below you will find the descriptions of the implemented protocols that can be used for the bus.

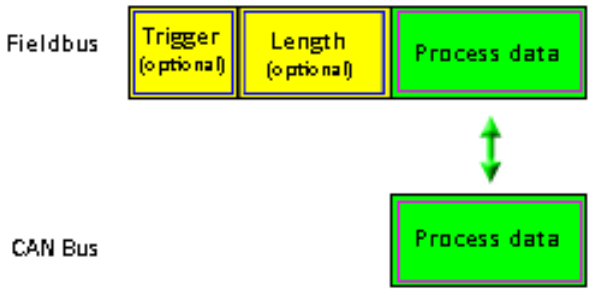
12.1 Protocol CANopen (default)

See „Mode of operation of the system“, chapter 8.

Note: The protocols for CAN Layer 2 can only be used from version 1.9.0 of the universal script Deutschmann.

12.2 Protocol CAN 2.0A

Data structure



Fieldbus trigger byte (i.e. fieldbus data exchange = On Trigger), fieldbus length byte and swap word.

Die Scriptdurchlaufzeit erhöht sich da zusätzliche Routinen durchlaufen werden.

12.3 Protocol CAN 2.0A with ID-filter

Additional configurations are possible:
Up to 16 COB-ID’s can be configured to which the CAN interface can react to.
All others are filtered (not processed).

If the value "0000" is in the first entry "Accepted ID 1", messages with COB-ID NULL are also reacted to.

If the value "NULL" is in one of the other entries "Accepted ID 2 ... 16)", this is the end of the list; all subsequent entries are not processed.

Example:

--------FIELDBUS--------	
CAN Transport protocol	**CAN 2.0A with ID filter**
Accepted ID 1	**0000**
Accepted ID 2	**0181**
Accepted ID 3	**0201**
Accepted ID 4	**0000**
Accepted ID 5	**0000**
Accepted ID 6	**0000**
Accepted ID 7	**0000**
Accepted ID 8	**0000**
Accepted ID 9	**0000**
Accepted ID 10	**0000**
Accepted ID 11	**0000**
Accepted ID 12	**0000**
Accepted ID 13	**0000**
Accepted ID 14	**0000**
Accepted ID 15	**0000**
Accepted ID 16	**0000**

In this example only the COB-ID´s „0000“ (hex), „0181“ (hex), „0201“ (hex) and „0080“ (hex) are processed, i.e. transferred to the fieldbus.

12.4 Protocol CAN 2.0B

A maximum of 13 Byte of data are displayed on the fieldbus side (1 Byte frame info, 4 Byte COB-ID, 8 Byte payload).
Additional configurations on the fieldbus side are possible:
Fieldbus trigger byte (i.e. data exchange = On Trigger), fieldbus length byte and Swap word.

Data display on the fieldbus side (SPS receives CAN) „11 Bit Mode“:

1. Byte number of received data from CAN (Low Nibble)
   MSB = 0 => 11Bit Frame received
2. Byte not used
3. Byte not used
4. Byte COB-ID High Byte
5. Byte COB-ID Low Byte
6. Byte Data n
7. Byte Data n+1
8. …

Example 1. Line: 8 Payload, COB-ID 181, Payload 11 … 88

Example 2. Line: 3 Payload, COB-ID 203, Payload 01 02 03

08 00 00 01 81 11 22 33 44 55 66 77 88
03 00 00 02 03 01 02 03 00 00 00 00 00

Data display on the fieldbus side (SPS receives from CAN) “29 Bit Mode”:

1. Byte number of received data from CAN (Low Nibble)
   Bit 7 (MSB) = 1 => 29Bit Frame received
   Bit 6 = RTR
   Bit 5 … 0 = not used
2. Byte COB-ID High Byte
3. Byte COB-ID
4. Byte COB-ID
5. Byte COB-ID Low Byte
6. Byte Data n
7. Byte Data n+1
8. …

Example 1. Line: MSB = 1 => 29 Bit Frame, 8 Payload, COB-ID 181, Payload 11 … 88

Example 2. Line: MSB = 1 => 29 Bit Frame, 3 Payload, COB-ID 203, Payload 01 02 03

88 00 00 01 81 11 22 33 44 55 66 77 88
83 00 00 02 03 01 02 03 00 00 00 00 00

SPS sends a 29 Bit data frame to COB-ID 456, 5 payload, data 01 02 03 04 05

Example:

SPS (Out): 85 00 00 04 56 01 02 03 04 05

To CAN Bus:

ID (hex)	Daten (hex)
456	01 02 03 04 05 06 20 20

12.5 Protocol CAN 2.0B with ID-filter

Identical with protocol CAN 2.0B, (see chapter 12.4), with additional configuration possibility.
Up to 16 COB-ID’s can be configured to which the CAN interface can react to as described in chapter 12.3.

13 Delivery status (factory setting)

On delivery, the UNIGATE® is set to the CANopen protocol (default) on the fieldbus side and to the Transparent protocol on the application side.

14 The trigger byte

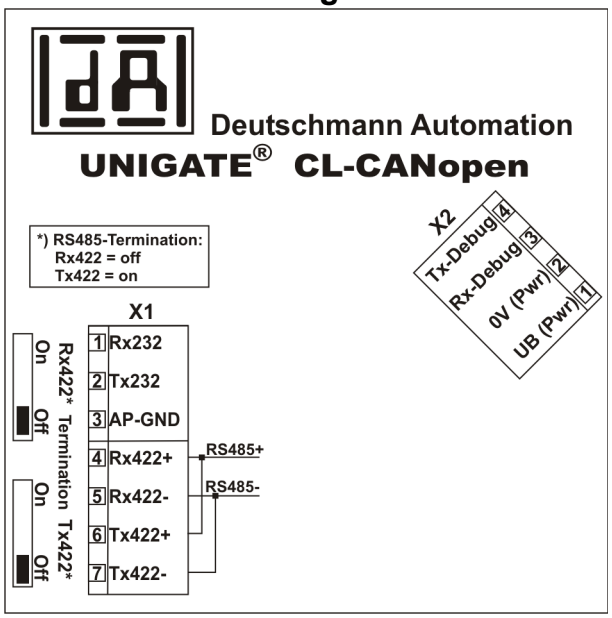
If data is transferred cyclically via the bus (which is usually not the case), the gateway must detect when the user wishes to send new data via the serial interface. For this reason, the user can set control of transmission via a trigger byte (data exchange -> On Trigger). In this mode, the gateway always sends (and only then) when the trigger byte is changed.
If the Trigger-Byte mode is activated, the gateway increments the trigger byte each time a telegram has been received.
The first byte in the bus input/output data buffer is used as the trigger byte if this mode is activated.

15 The length byte

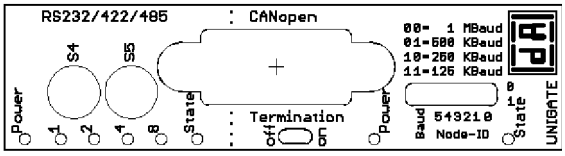
The user can configure whether the transmit length is also to be stored as a byte in the input/output data area (Fieldbus lengthbyte -> active). In transmit direction, as many bytes as specified in this byte are sent. On reception of a telegram the gateway enters the number of characters received.

16 Hardware ports, switches and LEDs

16.1 Device labeling



Picture 1: Terminal labeling and termination



Picture 2: Front panel: Rotary switches, DIP-switch, LEDs and termination CO

In case the front panel should pop out it does not affect the device's function or quality. It can be put in again without problems.

16.2 Connectors

16.2.1 Connector to the external device (RS-interface)

The serial interface is available at the plug accessible on the upper side of the device.

Pin assignment X1 (3-pole and 4-pole screw-type plug connector)

Pin No.	Name	Function
1	Rx 232	Receive signal
2	Tx 232	Transmit signal
3	AP-GND	Application Ground
4	Rx 422+ (485+)	Receive signal

5	Rx 422- (485-)	Receive signal
6	Tx 422+ (485+)	Transmit signal
7	Tx 422- (485-)	Transmit signal

For the operation at a 485-interface the two pins labeled "485-" have to be connected together.
Also the two pins "485+".

16.2.2 Connector supply voltage and DEBUG-interface

Pin assignment X2 (4-pin screw-plug connector, on the bottom side, 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	Receive signal Debug
4	Tx-Debug	Transmit signal Debug

Attention:
At isolated devices (option GT) Ground for the DEBUG-Interface must be connected with pin 3 (AP-GND) of the RS-interface!
At devices that are not isolated also the 0V (Pwr)-signal can be used as reference.

16.2.3 CANopen®-connector

The plug (labeled: CANopen®) for the connection to CANopen® is available on the front side of the device.

Pin assignment (9-pin D-SUB, plug)

Pin No.	Name	Function
1		
2	CAN-L	Dominant Low
3	CAN-GND	CAN Ground
4		
5		
6		
7	CAH-H	Dominant High
8		
9		

16.2.4 Power supply

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

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

16.3 LEDs

The Gateway UNIGATE® CL - CANopen® features 8 LEDs with the following significance:

LED (Bus) Power	green	Supply voltage CANopen®
LED (Bus) State	red/green	Interface state CANopen®
LED Power	green	Supply voltage serial interface
LED State	red/green	User-defined / general Gateway error
LEDs 1 / 2 / 4 / 8 (Error No. / Select ID)	green	User-defined / general Gateway error

16.3.1 LED "(Bus) Power"

This LED is connected directly to the electrically isolated supply voltage of the CANopen®-side.

16.3.2 LED "(Bus) State"

Indicator states and flash rates

The following Indicator states are distinguished:

LED on	constantly on
LED off	constantly off
LED flickering	iso-phase on and off with a frequency of approximately 10 Hz: on for approximately 50 ms and off for approximately 50 ms.
LED blinking	iso-phase on and off with a frequency of approximately 2,5 Hz: on for approximately 200 ms followed by off for approximately 200 ms.
LED single flash	one short flash (approximately 200ms) followed by a long off phase (approximately 1000 ms).
LED double flash	a sequence of two short flashes (approximately 200ms), separated by an off phase (approximately 200ms). The sequence is finished by a long off phase (approximately 1000 ms).
LED triple flash	a sequence of three short flashes (approximately 200ms), separated by an off phase (approximately 200ms). The sequence is finished by a long off phase (approximately 1000 ms).

If one bicolor Status LED is used instead of two single color LEDs, this LED shall indicate both the physical bus status and the status of the CANopen® state machine. This bicolor LED shall be red and green.

CANopen® ERROR LED (red)

The CANopen® Error LED indicates the status of the CAN physical layer and indicates errors due to missing CAN messages (SYNC, GUARD or HEARTBEAT).

ERROR LED	State	Description
Off	No error	The Device is in working condition.
Single flash	Warning limit reached	At least one of the error counters of the CAN controller has reached or exceeded the warning level (too many error frames).
Flickering	AutoBaud/LSS	Auto Baudrate detection in progress or LSS services in progress (alternately flickering with RUN LED).
Double flash	Error Control Event	A guard event (NMT-Slave or NMT-master) or a heartbeat event (Heartbeat consumer) has occurred.
Triple flash	Sync Error	The SYNC message has not been received within the configured communication cycle period time out (see Object Dictionary Entry 0x1006).
On	Bus Off	The CAN controller is'bus off

If at a given time several errors are present, the error with the highest number is indicated (e.g. if NMT Error and Sync Error occur, the SYNC error is indicated).

CANopen® RUN LED (green)

The CANopen® RUN LED indiates the status of the CANopen® network state machine.

CAN RUN LED	State	Description
Flickering	AutoBaud/LSS	Auto Baudrate detection in progress or LSS services in progress (alternately flickering with RUN LED).
Single flash	STOPPED	The device is in STOPPED state.
Blinking	PRE-OPERATIONAL	The device is in the PRE-OPERATIONAL state.
On	OPERATIONAL	The device is in the OPERATIONAL state.

Whilst the device is executing a reset the CANopen® RUN LED shall be off.

In case there is a conflict between turning the LED on green versus red, the LED may be turned on red. Apart from this situation, the bicolor status LED shall combine the behavior of the CAN Error LED.

Old hardware (with RD2-processor) - supplied until the end of February 2009:

Lights green	CAN-state = OPERATIONAL
Flashes green	CAN-state = PREOPERATIONAL or PREPARED
Flashes red	Guarding error
Lights red	CAN-bus error

16.3.3 LED "Power"

This LED is connected directly to the (optionally also electrically isolated) supply voltage of the serial interface (RS232/422/485).

16.3.4 LED "State"

Lights green	Data exchange active via RS422 / RS485 / RS232; controllable via script
Flashes green	RS422 / RS485 / RS232 OK, but no constant data exchange; controllable via script
Flashes green/red	No data exchange since switching on; controllable via script
Lights red	General Gateway error (see LEDs Error No.), controllable via Script
Flashes red	UNIGATE is in the configuration / test mode, controllable via Script

16.3.5 LEDs 1 / 2 / 4 / 8 (Error No. / Select ID)

If these 4 LEDs flash and LED “State“ simultaneously lights red, the error number is displayed in binary notation (conversion table, see Annex) in accordance with the table in chapter "Error handling". Additionally there LEDs are controllable via Script.

16.4 Switches

The Gateway features 6 switches with the following functions:

Termination Rx 422	switchable Rx 422-terminating resistor for the serial interface
Termination Tx 422	switchable Tx 422- or RS485-terminating resistor for the serial interface
Rotary coding switch S4	ID High for serial interface i. e. configmode
Rotary coding switch S5	ID Low for serial interface i. e. configmode
Termination (CANopen®)	switchable CANopen®-terminating resistor
DIP-switch	Node-ID and baud rate

16.4.1 Termination Rx 422 + Tx 422 (serial interface)

If the Gateway is operated as the physically first or last device in an RS485-bus or as 422, there must be a bus termination at this Gateway. In order to do this the termination switch is set to position ON. The resistor (150 Ω) integrated in the Gateway is activated. In all other cases, the switch remains in position OFF.
Please refer to the general RS485 literature for further information on the subject of bus terminations.
If the integrated resistor is used, please allow for the fact that this also activates a pull-down resistor (390 Ω) to ground and a pull-up resistor (390 Ω) to VCC.

At RS485 only the Tx 422-switch must be set to ON.
The Rx 422-switch has to be set to OFF.

16.4.2 Rotary coding switches S4 + S5 (serial interface)

These two switches can be read out through the Script command "`Get (RS_Switch, Destination)`" and the value can be used for further functions. This value is read in when the Gateway is switched on or always after a Script command has been executed. The switch positions "EE" (testmode) and "FF" (config mode) are not possible for RS422- or RS485-operation.

Note: The switch position "DD" (i.e., S4 and S5 in position "D") is reserved for internal purposes.

16.4.3 Termination (CANopen®)

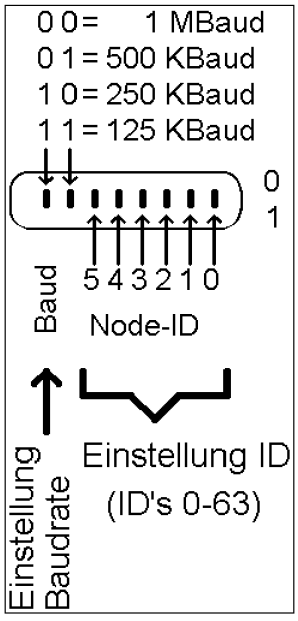
If the Gateway is operated as the first or last physical device in the CANopen®, there must be a bus termination at this Gateway. In order to do this, either a bus terminating resistor must be activated in the connector or the resistor (220 Ω) integrated in the Gateway must be activated. In order to do this, slide the slide switch to position ON. In all other cases, the slide switch must remain in position OFF. Please refer to the general Fieldbus literature for further information on the subject of bus termination.
Note: To activate or deactivate the bus termination, please remove the BUS-connector and carefully set the switch to the desired position.

16.4.4 DIP-switch

The DIP-switch is used to set the Node-ID and Baud rate according to picture 3.

If the Node-ID 0 (which is not allowed in CANopen®) is set on the DIP-Switch, then the Node-ID, which is stored in the EEROM via the Script or WINGATE is used in this case. That way it is also possible to set Node-IDs > 63 and there is no unclarity if the Node-ID of the DIP-Switch or the Node-ID of the Script is valid.

In the initial state the Node-ID 1 is stored in the EEROM.



Picture 3: DIP-switch

16.5 The Debug cable for UNIGATE® CL

As accessory a pre-configured Debug cable is available. The Debug cable connects the Gateway to Debug and RS.

17 Error handling

17.1 Error handling at UNIGATE® CL

If the Gateway detects an error, the error is signalled by the “State“ LED lighting red and, simultaneously, the error number being indicated by means of LEDs “Error No.“ as shown in the table below. In the default state this error numer is additionally sent as emergency telegram via CANopen®. The code 61xx (hex), that indicates an internal Firmware error according to DS301 is used for it. The current error code is entered for “xx“ in accordance with the below table. A detail error, that is used for internal purposes may still be included in byte 3 and 4 of the emergency message.
A distinction can be made between two error categories:
Serious errors (1-5): In this case, the Gateway must be switched off and switched back on again. If the error occurs again, the Gateway must be exchanged and returned for repair.

Warnings (6-15): These warnings are displayed for one minute simply for information purposes and are then automatically reset. If such warnings occur frequently, please inform After-Sales Service.

Error 10 is additionally activated in case of the heartbeat-error.

In the configuration mode these displays are not valid and only meant for internal use.

LED8	LED4	LED2	LED1	Error no. resp. ID	Error description
0	0	0	0	0	Reserved
0	0	0	1	1	Hardware fault
0	0	1	0	2	EEROM error
0	0	1	1	3	Internal memory error
0	1	0	0	4	Fieldbus hardware error or wrong Fieldbus-ID
0	1	0	1	5	Script error
0	1	1	0	6	Reserved
0	1	1	1	7	RS-transmit buffer overflow
1	0	0	0	8	RS-receive buffer overflow
1	0	0	1	9	RS timeout
1	0	1	0	10	General fieldbus error
1	0	1	1	11	Parity-or frame-check-error
1	1	0	0	12	Reserved
1	1	0	1	13	Fieldbus configuration error
1	1	1	0	14	Fieldbus data buffer overflow
1	1	1	1	15	Reserved

Table 1: Error handling at UNIGATE® CL - system error

Flashing frequency 2 times per second (system error)

LED8	LED4	LED2	LED1	Error-No.	Protocol	Error description
0	0	1	1	3	all Protocols	No Universal script support
0	1	0	1	5	all Protocols	Unknown Protocols
1	0	0	1	9	Modbus RTU Master Modbus ASCII Master	Timeout-Modbus Slave Participant didn't response in set time frame (response time).
					Modbus RTU Slave Universal Modbus RTU Slave	Timeout at Response-transmission
					3964(R)	Timeout - no response from Participant
1	0	1	1	11	Universal 232 (with 232 Checksum)	Checksum of reception does not match the calculated one.
					Modbus RTU Slave Universal Modbus RTU Slave	Unknown error after response transmission.
					3964(R)	Error at data exchange (e.g. Checksum error)
1	1	0	0	12	Universal Modbus RTU Master	Error in Response of Functioncode
1	1	0	0	12	SSI	Error at the SSI communication
1	1	1	1	15	Modbus RTU Master Modbus ASCII Master	General reception error at Modbus (ASCII) Exchange, e.g. Checksum error
1	1	1	0	14	Modbus RTU Slave	Exception Response
1	1	1	1	15	all Protocols	internal error at process data measurement

Table 2: Protocol based errors

Flashing frequency once per second (user-defined errors or protocol-related errors)

Note: The error is displayed as long as is defined with "Set Warning Time".

18 Installation guidelines

18.1 Installation of the module

The module with the dimensions 23 x 117 x 111 mm (W x D x H) has been developed for switch cabinet use (IP 20) and can thus be mounted only on a standard mounting channel (deep DIN-rail to EN 50022).

18.1.1 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 mm².

18.1.2 Removal

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

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

18.2 Wiring

18.2.1 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 D-SUB plug connector (CANopen®)

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 mm²
- Solid conductor: 1 x 0.25 ... 1.5 mm²
- 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 D-SUB plug connector is 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

18.2.1.1 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.

18.2.1.2 Equipotential bonding connection

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

18.2.2 CANopen® communication interface

18.2.2.1 Bus line with copper cable

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 labeled ...on...
- If the module is not located at the start or at the end, you must set the slide switch to position "off".

18.2.3 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.

18.2.4 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 PROFIBUS DP, RS232C and printers etc.)
- shielded analogue lines
- unshielded lines for DC voltages ≥ 60 V
- unshielded lines for AC voltage ≥ 25 V
- coaxial lines for monitors

⇒ Group B:
- unshielded lines for DC voltages ≥ 60 V and ≥ 400 V
- unshielded lines for AC voltage ≥ 24 V and ≥ 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 3: 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.

18.2.4.1 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 favorable 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 metallized 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!

19 CANopen®

19.1 Description CANopen®

This specification is based on the CiA® Draft Standard 301 (DS301).
CANopen® supports the Standard CAN-frame with 11-bit Identifier.
It is not required to support the extended frame with 29-bit Identifier.

19.1.1 CANopen® V3

Only for used products. Currently we provide CANopen V4.

A CANopen V3 script can be adapted to CANopen V4 with only minor changes.

Syntax
CO_InitChannel (vw_Channel , Direction , vw_len , vw_Obj_Nr , vw_COBId)

Description
From Script rev. 25 on and higher the CAN Firmware allows the definition of user objects and the mapping of up to 16 Rx and 16 Tx PDOs. For Script rev. between 22 and 25 only up to 5 Rx and 5 Tx PDOs can be used.

Predefined Communication
For some applications one Rx and one Tx PDO is sufficient.
It is possible to use CANopen® without the definition of communication channels. In this case the data is mapped as follows:

Data width	Direction	Object	Mapping
1..8 byte	Rx	2000	Default Rx-PDO1 (COB-ID 200 + Node ID)
1..8 byte	Tx	2001	Default Tx-PDO1 (COB-ID 180 + Node ID)
9..255 byte	Rx	2000	Data not mapped (could be read by SDO) no Rx-PDO available
9..255	Tx	2001 (Tx-Data) 2002 (Tx-Length)	Data not mapped (write by SDO) Data width in Tx PDO 1 (COB-ID 180 + Node ID)

It is possible to use ReadBus and WriteBus for the data exchange. This standard behavior is no longer active if you call CO_InitChannel at least once.

User-defined communication
This mode is necessary if you want to use more sophisticated CANopen® functions. You have to initialize a CANopen® channel for every PDO or object.

Syntax
CO_InitChannel (vw_Channel , Direction , vw_len , vw_ObjAddress , vw_COBId)

Use the following values for the parameters:

<table>
<tr><th>Parameter</th><th>Type</th><th>Meaning</th></tr>
<tr><td>vw_Channel</td><td>Word</td><td>
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>Using CAN Layer2, we have no PDO and SDO data access</td></tr>
<tr><td>1..8</td><td>Define PDO 1..8</td></tr>
</table>
</td></tr>
<tr><td>Direction</td><td>-</td><td>RX or Tx depending of the data direction. It is seen from the devices view, this means Rx-data is incoming data.</td></tr>
<tr><td>vw_ByteLen</td><td>Byte</td><td>Length of the object to use. If the length is > 8 byte only the first 8 bytes are used to transmit by the PDO</td></tr>
<tr><td>vw_ObjAddress</td><td>Word</td><td>Allowed values are 0x2000 to 0x5FFF, which is the range of objects to be defined by the user</td></tr>
<tr><td>vw_COB_ID</td><td>Word</td><td>
<table>
<tr><th>Value</th><th>Meaning</th></tr>
<tr><td>0</td><td>PDO is not active, data is defined to be used by a SDO transfer only.</td></tr>
<tr><td>0x181..0x57F</td><td>Allowed range for normal Rx and Tx - PDO's</td></tr>
<tr><td>0xFFFE</td><td>Is to be used, if the master defines the COB-ID's when the CANopen network is started by the master (no predefined COB-IDs).
The resulting objects used by the device are 0x1800 + (PDO-Nr - 1). Sub-Index 1 of this object contains the COB-ID. After writing a valid value to this object with subindex the requested PDO becomes active. Valid values must be in the range from 0x181 to 0x57F.</td></tr>
<tr><td>0xFFFF</td><td>Is to be used for PDO1 and PDO2. The COB-ID is as defined by the predefined connection set
Tx-PDO1: 0x180 + Node-ID
Rx-PDO1: 0x200 + Node-ID
Tx-PDO2: 0x280 + Node-ID
Rx-PDO2: 0x300 + Node-ID</td></tr>
</table>
</td></tr>
</table>

CAN Layer 2

If you want to use CAN Layer 2, you can set a special Script initialization to access every CAN message without any Filter. Please note that the data format for ReadBus and WriteBus differs from other functionalities in this case. From now on the COB-ID of the message is to be read or sent in the data area's first 2 bytes.

The following examples can be found in the file folder "example" after the installation of the software "Protocol Developer":

- Example CANopen® 2 PDOs
- Example CAN Layer 2

With it please take a look at the fololwing Script commands from the "Protocol Developer" as well:

- CO_Read PDO
- CO_WriteEmergency
- CO_WritePDO

19.1.2 CANopen® V4

Additional supported functions:
- Heartbeat
- Dynamic mapping
- Onswich message

The following example can be found in the file folder “example“ after the installation of the software “Protocol Developer“ (this example gives a detailed description of the initialization):
- Example_CO_V4.dss

At CANopen® V4 the following fieldbus-specific Scripts are supported:
- Init Object Table
- Create Object
- Set PDO Communiction
- Set PDO Mapping
- Write Object
- Read New CANopen® Object Data
- Emergency Message

The software does not support default objects, as described at CANopen® V3.

20 Technical data

20.1 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	23 x 117 x 111 mm (screw-plug-connector included) 23 x 117 x 100 mm (screw-plug connector not included)	W x D x H
5	Installation position	Any	
6	Weight	123g	
7	Operating temperature	-40 ºC ... +85 ºC	The negative temperatures are only valid for the usual conditions (not condensing)
8	Storage/transport temperature	-40 ºC ... +85 ºC	
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 ≤ 40ºC
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. 50 mA max 60 mA	At 10V. Max. 150 mA
14	Reverse voltage protection	Yes	But 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

20.1.1 Interface data

The table below lists the technical data of the interfaces and ports on the device. The data has been taken from the corresponding Standards.

No.	Interface designation Physical interface	CANopen® RS485	RS232-C RS232-C	RS485/RS422 RS485/RS422
1	Standard	CiA® DS 102	DIN 66020	EIA Standard
2	Transmission mode	Symmetrical asynchronous serial half-duplex → Difference signal	Asymmetrical asynchronous serial full duplex → Level	Symmetrical asynchronous serial half-duplex full duplex at RS422 → Difference signal
3	Transmission method	Master / slave	Master / slave	Master / slave
4	Number of users : - Transmitters - Receivers	32 32	1 1	32 32
5	Cable length: - Maximum - Baud rate-dependent	1300 m 50 kBd → 1300 m 100 kBd → 640 m 200 kBd → 310 m 500 kBd → 112 m 1 MBd → 40 m	15 m no	1200 m <93.75 kBd → 1200 m 312, kBd → 500 m 625 kBd → 250 m
6	Bus topology	Line	Point-to-point	Line
7	Data rate: - Maximum - Standard values	1Mbit/s 125 kB 250 kB 500 kB 1 MB	120 kBit/s 2.4 k/B 4.8 k/B 9.6 kBit/s 19.2 kBit/s 38.4 kBit/s	625 kBaud 2.4 kBit/s 4.8 kBit/s 9.6 kBit/s 19.2 kBit/s 57.6 kB 312.5 kB 625 kB
8	Transmitter: - Load - Maximum voltage - Signal, unloaded - Signal, loaded	54 Ω - 7 V ... 12 V ± 5 V ± 1.5 V	3 ... 7 kΩ ± 25 V ± 15 V ± 5 V	54 Ω - 7 V ... 12 V ± 5 V ± 1.5 V
9	Receiver: - Input resistance - Max. input signal - Sensitivity	12 Ω - 7 V ... 12 V ± 0.2 V	3 ... 7 Ω ± 15 V ± 3 V	12 Ω - 7 V ... 12 V ± 0.2 V
10	Transmit range (SPACE): - Voltage level - Logic level	 - 0.5 ... + 0.05 V 0	 + 3 ... + 15 V 0	 - 0.2 ... + 0.2 V 0
11	Transmit pause (MARK): - Voltage level - Logic level	 + 1.5 ... +3 V 1	 - 3 ... –15 V 1	 + 1.5 ... +5 V 1

Table: Technical data of the interfaces and ports on the module

21 Commissioning guide

21.1 Note

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

21.2 Components

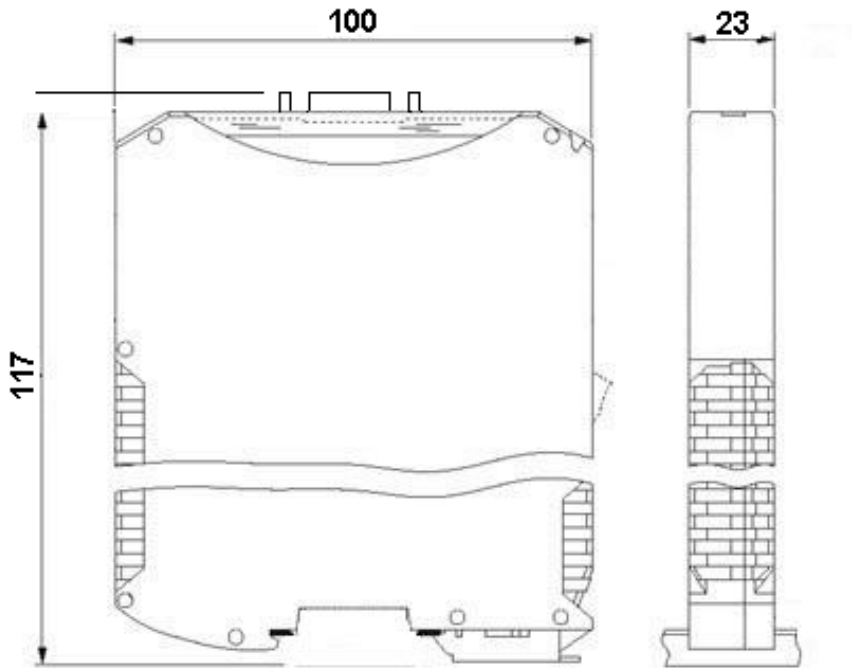
You will require the following components to commission the UNIGATE®:

- UNIGATE®
- Connection cable from gateway to the process
- Connector for CANopen® connection to the Gateway
- CANopen® cable (generally this cable is already installed on site!)
- 10..33 V DC power supply (DIN 19240)
- Type file or EDS file and user manual (a sample EDS file as well as the user manual can be ordered separately or downloaded free of charge from our homepage at www.deutschmann.de).

21.3 Installation

The UNIGATE® CL - CO module features protection type IP20 and is thus suitable for switch cabinet use. The device is designed for snapping onto a 35 mm DIN-rail.

21.4 Dimensional drawing UNIGATE® CL - CANopen®



21.5 Commissioning

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

21.6 Setting the CANopen® address and baud rate

Set the CANopen®-Node-ID and the baud rate at the fieldbus end of the module on the DIP-switch (see chapter 16.4.4).

Attention:
The CANopen® address set must correspond to the planned address!
All users in the CANopen® have to use the same Baud rate!
These values are read in only on power-up of the gateway!

21.7 CANopen® connection

Connect the device to the CANopen® at the interface labelled "CANopen®".

21.8 Connection to the process device

Please also read the manual for the process device when commissioning the process device.

21.9 Connecting the supply voltage

Please connect 10..33 DC voltage to the terminals provided for this.

21.10 Shield connection

Earth the top-hat rail onto which the module has been snapped.

21.11 Project planning

Use any project planning tool for project planning.
If the required EDS file was not supplied with your project planning tool, a sample file can be found on the Internet (www.deutschmann.de).

22 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 or directly in our Wiki on 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.).

Technical Support hours are as follows:
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 and sales department	+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	support@deutschmann.de

22.1 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 (±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.

22.2 Downloading PC software

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

23 Annex

23.1 Explanations of the abbreviations

General

CL	=	Product group CL (Compact Line)
CM	=	Product group CM (CANopen Line)
CX	=	Product group CX
EL	=	Product group EL (Ethernet Line)
FC	=	Product group FC (Fast Connect)
GT	=	Galvanic separation RS-side
GY	=	Housing color gray
MB	=	Product group MB
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)
IO8	=	Option I/O8
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/s
FEX	=	Ethernet 10/100 MBit/s-version X (see comparison table UNIGATE® IC for the respective product)
IB	=	Interbus
IBL	=	Interbus
LN62	=	LONWorks62
LN512	=	LONWorks512
ModTCP	=	ModbusTCP
MPI	=	Siemens MPI®
PL	=	Powerlink
PN	=	Profinet-IO
PBDP	=	ProfibusDP
PBDPL	=	ProfibusDP-version L (see comparison table UNIGATE® IC for the respective

product)

PBDPX = ProfibusDP-version X (see comparison table UNIGATE® IC for the respective product)
PBDPV0 = ProfibusDPV0
PBDPV1 = ProfibusDPV1
RS = Serial RS232/485/422

23.2 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