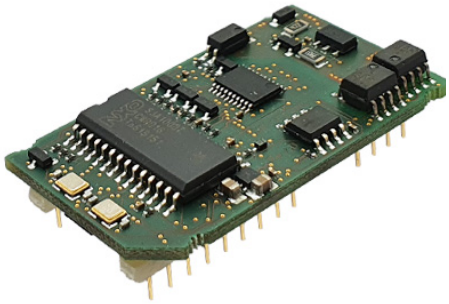
Deutschmann

*your ticket to all buses*

# Instruction Manual
# Universal Fieldbus-Gateway
# UNIGATE® IC - DeviceNet



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

Manual Art-No. V3289E

Disclaimer of liability

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

# 1 General introduction

In the past the integration of a fieldbus connection required an enormous effort from the progress engineers. On account of the large variety of communication systems it is not enough to compile the right combination of communication hardware; due to their standards and fundamentals different busses also require the corresponding skills of the engineers.

This does not apply in case of the UNIGATE® IC by Deutschmann Automation any more. All digital functions, software, stack and driver as well as optocoupler are integrated on a UNIGATE® IC in correspondence with the standard. In addition to the reduction of the required size, also different fieldbusses can easily be integrated.

**Through the flexible firmware of UNIGATE® IC no software-changes are required on the side of the customer!**

Since 1997 Deutschmann Automation has experience in the field of fieldbus gateways; this enormous experience results in the UNIGATE® IC as a consistent sequel of this successful product line.

**Terminology**
In the entire document and in all parts of the software that is to be used, the terms Input and Output are used. Input and Output are ambiguous, always depending on the viewpoint. We see the fieldbus as central interface and as integral component of your device; therefore in all places it is always referred to data from the viewpoint of the Slave, that is Input data, as data from the Master to the Slave - regardless of the used bus.

**Representation of numbers**
Numbers in decimal format are always represented without prefix and without suffix as well. Hexadecimal numbers are always marked with the prefix 0x.

# 2 The UNIGATE® IC

## 2.1 Technical introduction

The UNIGATE® IC by Deutschmann Automation contains all components that are required for the communication in a fieldbus in one single module. Therefore a developer does not have to take care for that detail any more, only a hardware redesign is necessary in order to integrate the UNIGATE® IC and the required plug connectors.

## 2.2 Availability

The module is available as DeviceNet. Further fieldbusses are either planned or being worked on. They will only differ in the connections of the busses. The meaning of the general pins 1 - 9 as well as 24 and 26 - 32 will remain unchanged also for further fieldbus implementations. An up-to-date list for all UNIGATE® ICs can be found at:
http://www.deutschmann.de

## 2.3 Firmware

UNIGATE® IC is programmed via scripts. On principle any script, that has been developed for a UNIGATE® SC, can also be operated on the UNIGATE® IC.

## 2.4 The serial standard interface

Intelligent devices, that already feature a micro controller or a microprocessor, are generally supplied with a serial asynchronous interface with a TTL-level. It is directly connected with the TTL-interface of the UNIGATE® IC. For more information on this serial interface see chapter 4 on page 23.

## 2.5 The synchronous seral interface

In addition to the standard interface there is also the possibility of the synchronous input and output. That way for instance digital IOs can be connected through shift register components or also analog IOs can be connected through a DA-converter with serial in-/output. For synchronous IOs 256 signals at the most can be used (256 bit). Wiring examples can be found in chapter 3.5 and software examples can be found in chapter 5. This interface can also be used to connect modules or devices with SPI-interface. It is also possible to build, for instance digital or analogous I/O-modules, with the customer's device not being equipped with an own controller. The fieldbus IC is also operable autonomously without that controller.

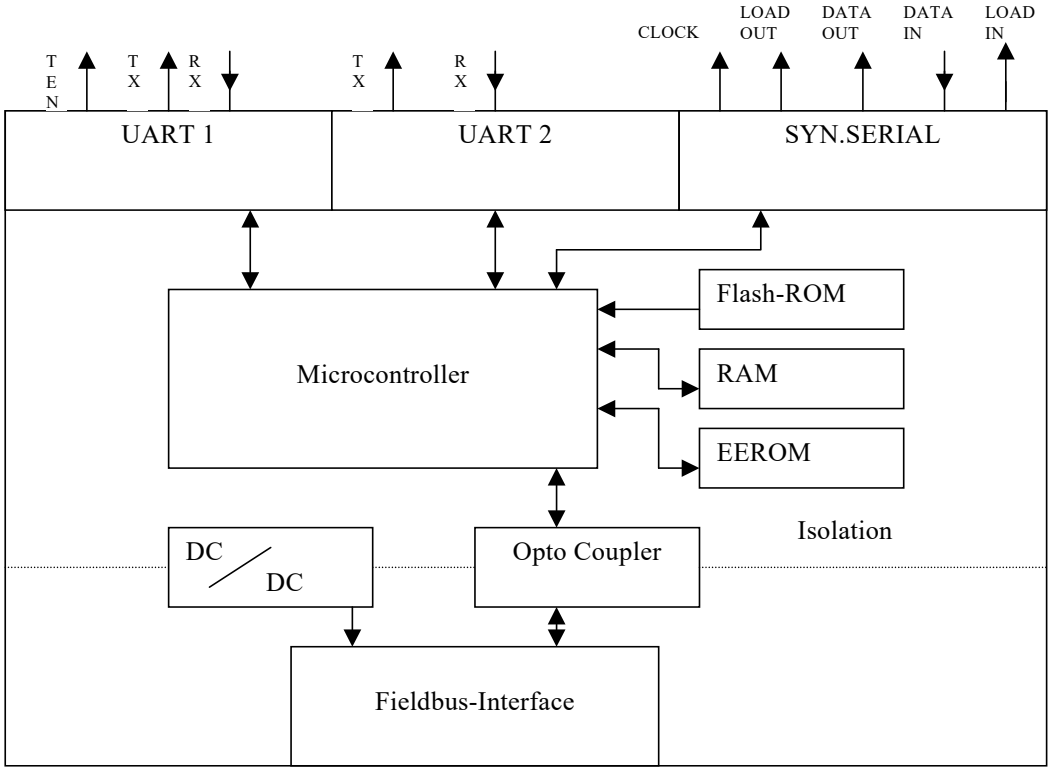
## 2.6 The Debug-interface

The UNIGATE® IC features a Debug-interface, which allows to process a script step by step and also to monitor or manipulate data. This is indispensable for the development of a script. Usually a script is developed with the software Protocol Developer. For more details take a look at the instruction manual Protocol Developer.

**All interfaces can independently be used at the same time.**

## 2.7 UNIGATE® IC hardware survey

The hardware of the UNIGATE® IC consists of some few standard components. The picture below shows the functional structure of the IC.

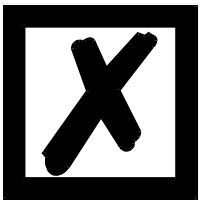


# 3 Hardware design

This chapter gives basic advise, that is required in order to load UNIGATE® IC into your own hardware designs. In the following all ports of UNIGATE® IC are described in detail.

## 3.1 Ports

UNIGATE® IC features 32 pins in its layout as a DIL 32 component. Pin 10 -12 and 21 - 23 as well are not wired due to the electrical isolation. The exact mechanical dimensions can taken from chapter 12.



**In the layout boreholes for ALL 32 pins have to be planned.**

## 3.2 Pinout

| Pin | Technical specifications | Name | Description | Remark |
|---|---|---|---|---|
| 1* | 5V ± 5% < 75mA | Vcc | + 5 V voltage supply | optionally 3,3V available |
| 2 | $IN_{Logic}$ | -BE | -boot enable | |
| 3 | $OUT_{Driver}$ | Load out (SS0-) | strobe signal for synchronous, serial interface | |
| 4 | $OUT_{Driver}$ | Data out (SS1-) | output data for synchronous, serial interface | |
| 5 | $IN_{Logic}$ | Data in (MISO) | input data of the synchronous, serial interface | internally pulled up with 10 kΩ |
| 6 | $OUT_{Logic}$ | Load in (MOSI) | input data of the synchronous, serial interface; alternatively strobe signal of the output data | |
| 7 | $OUT_{Driver}$ | Clock (SCK) | clock pulse signal for synchronous, serial interface | |
| 8 | $IN_{Reset}$ | -Reset in[1)] | reset-input of the IC | internally pulled up with 100 kΩ |
| 9* | connected to pin 1 | Vcc | + 5 V voltage supply | |
| 10-12 | nc | nc | no pin available | |
| 13 | according to norm | CANL | DeviceNet-signal according to standard | galvanically isolated insulation voltage 1000 Vrms |
| 14 | according to norm | CANH | DeviceNet-signal according to standard | galvanically isolated insulation voltage 1000 Vrms |
| 15 | according to norm | nc | not connected | |
| 16 | according to norm | nc | | |
| 17 | according to norm | V+ | Voltage supply DeviceNet 24V | |
| 18 | according to norm | V- | Voltage supply DeviceNet 0V | |
| 19 | according to norm | | not available, pin internally connected | |
| 20 | according to norm | nc | not connected | |
| 21-23 | nc | nc | no pin available | |
| 24* | connected to pin 32 | GND | ground supply voltage of the IC | |
| 25 | OUT | LED-DN | LED DeviceNet | anode of the green LED |
| 26 | $IN_{Logic}$ | -Config Mode | signal to start the configuration mode | internally pulled up with 10 kΩ |
| 27 | $OUT_{Logic}$ | DbgTX | serial Debug TX | |
| 28 | $IN_{Logic}$ | DbgRX | serial Debug RX | internally pulled up with 10 kΩ |
| 29 | $IN_{Logic}$ | RX | serial data RX | internally pulled up with 10 kΩ |
| 30 | $OUT_{Logic}$ | TX | serial data TX | |
| 31 | $OUT_{Logic}$ | TE | transmit enable | |
| 32* | GND | GND | ground supply voltage of the IC | |

* The supply voltage is 5 V +/- 5 %, max. 75 mA DC (optionally 3,3V).
The DeviceNet signals are galvanically isolated. The insulation voltage is 1000 Vrms.

| | $V_{IL}$ | $V_{IH}$ |
|---|---|---|
| $IN_{Reset}$ | < 0.3V / 5mA | > 1.95V / 10µA |
| $IN_{Logic}$ | < 0.8V / 0.5mA | > 1.95V / 10µA |

| | $V_{OL}$ | $V_{OH}$ |
|---|---|---|
| $OUT_{Logic}$ | < 0.6V / 1mA | > 3.8V / 0.1mA |
| $OUT_{Driver}$ | < 0.33V / 4mA | > 3.8V / 4mA |

### 3.2.1 -Boot enable

The IC is started in the firmware update mode with the level GND during the power up process. See also chapter 11.

### 3.2.2 Load out (SPI-Master: SS0-)

Strobe signal for the synchronous serial interface. With the positive edge at this output data is taken from the connected shift registers to the physical outputs.
In SPI mode this Pin serves as a low-active Slave-Select-Signal.

### 3.2.3 Data out (SPI-Master: SS1-)

On this line data is output on the synchronous serial interface. The most significant bit of the data is output first.
In SPI mode this Pin serves as a low-active Slave-Select-Signal.

### 3.2.4 Data In (SPI: MISO)

Data is read in on the synchronous serial interface via this signal. The most significant bit of the data is expected first.
In SPI mode this Pin serves as data transfer from Slave to Master.

### 3.2.5 Load In (SPI: MOSI)

This pin is the strobe signal for the input data of the synchronous serial interface.
In SPI mode this Pin serves as data transfer Master to Slave.

### 3.2.6 Clock (SPI: SCK)

This signal is the clock line for the synchronous serial interface. That signal is equally valid for data input and data output.

### 3.2.7 -Reset In

- A reset generator (Max 809) is on board; with it in the normal case the reset input is not required. In this case the reset input has to be connected with VCC, in order to avoid interferences (see chapter 3.6).
- If the the customer's application has to initiate a reset of the UNIGATE® IC, then the reset input can also be connected with a reset output of the customer's application instead of connecting it with VCC. Here all specifications of the reset signal, mentioned in chapter 3.2 have to be kept. The reset-impulse is supposed to last at least 10 ms.

### 3.2.8 LED-DN

A green LED can be connected to this wire from hardware revision C (lower board) or Rev. - (DNL-hardware / single-board solution) (see chapter 3.6). This LED flashes in the state "Bus ok, not allocated" and shines shines in the state "Allocated".
-> This LED can not be used in case of RS485-operation.

### 3.2.9 -Config Mode

If the pin has the level GND, then the IC starts in the configuration mode.

### 3.2.10 DbgTX, DbgRX

They are transmission line (Tx) and receive line (Rx) as well of the IC's Debug-interface. For the function description of the Debug-interface see chapter 6.

### 3.2.11 TE

The transmit enable signal allows the connection of RS485 drivers to the IC's serial interface. The signal is set to High whenever the IC sends via the line TX.

### 3.2.12 TX, RX

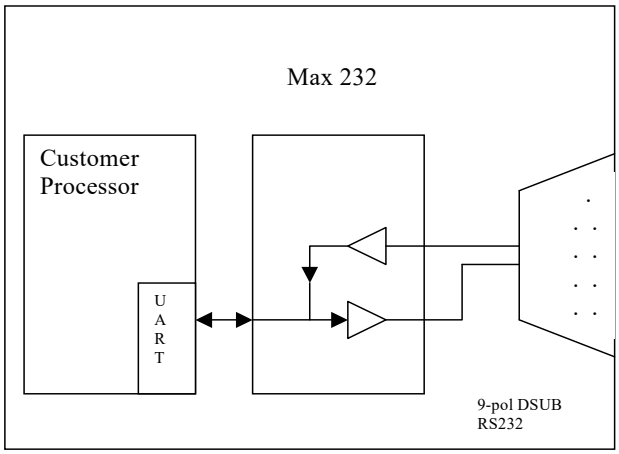
Transmission (Tx) and receive lines (Rx) of the serial interface. This interface is programmable in accordance with the description in chapter 4.

## 3.3 Software

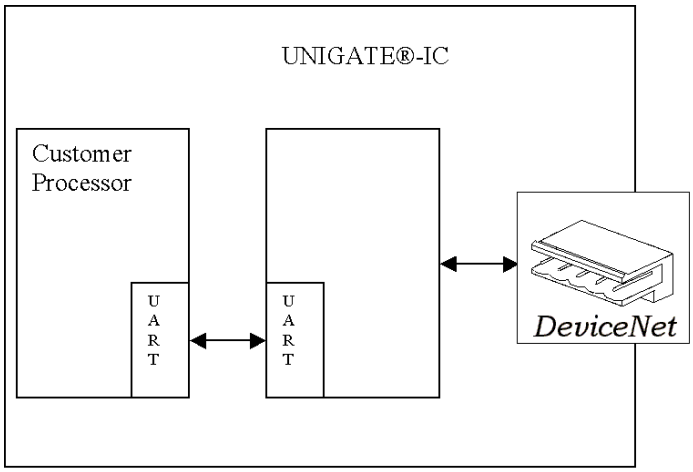
The software executes script-commands, which in turn control the IC's hardware and they process their complete protocol by software. The script itself can be generated by the company Deutschmann Automation or with the software Protocol Developer by yourself. For a detailed description of the script.commands of the Protocol Developer see the instruction manual Protocol Developer and the online documentation concerning script-commands.

## 3.4 Basic line of proceeding

In theory it is enough to replace the RS232-driver that is included in your application by the UNIGATE® IC.



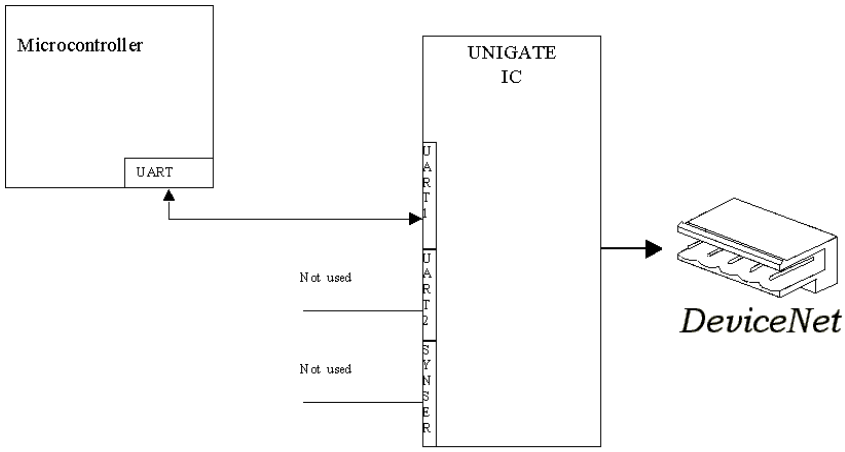
Your device, which on the whole is supposed to be assembled as shown above, will now be modified in a way that the DeviceNet is available at the 5-pol. socket. However, a hardware redesign is necessary in order to keep the assignment in standard form.




After the RS232-driver has been replaced by the UNIGATE® IC, the DeviceNet is available at the 5-pol. screw-plug connector.
Deutschmann Automation is also offering an appropriate adapter board. With it existing devices can be adapted without re-design; see chapter 13.

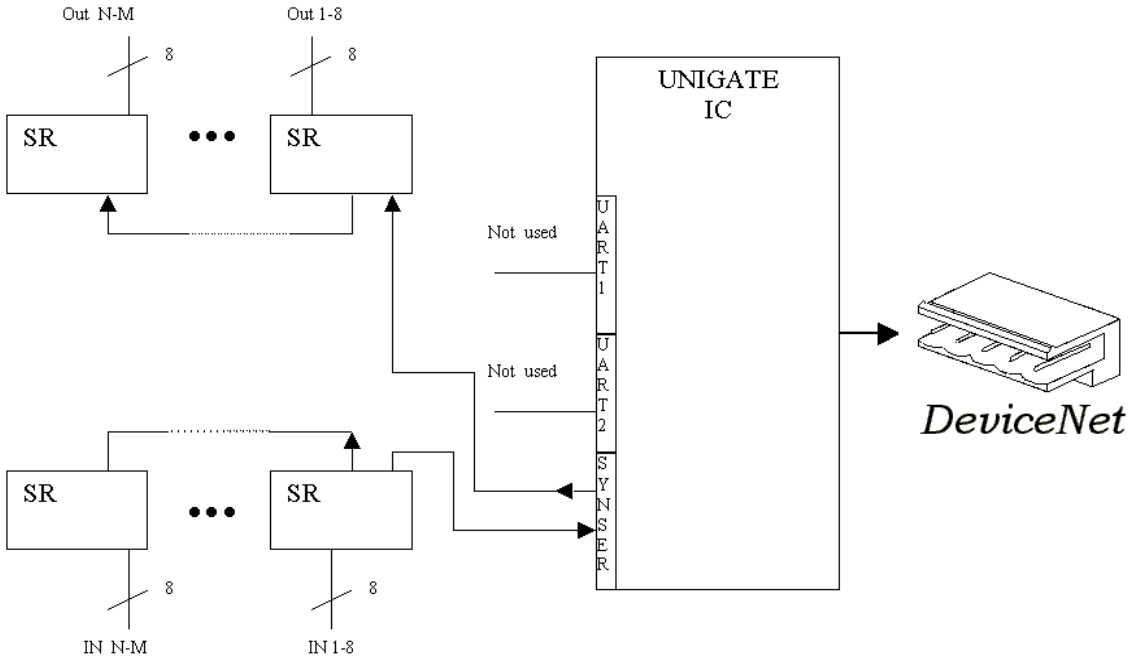
## 3.5 Connection examples

Here you will find some advise that offers help for a re-design. In the following several versions are listed, that should make it easier for you to decide.
Version 1: Use as a pure link module for the bus




The UNIGATE® IC independently processes the communication with the customer's device via the TTL-interface.

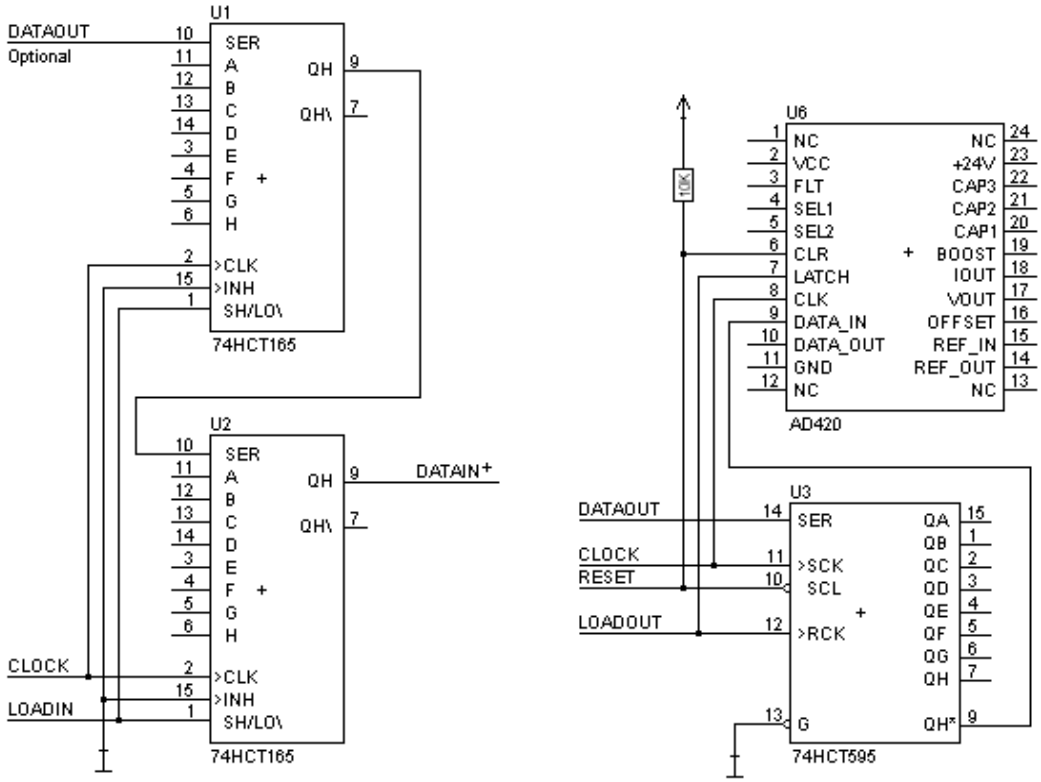
Version 2: Use of UNIGATE® IC for digital or analog I/O-modules




Here only the synchronous serial interface is used, the asynchronous serial interface is basically of no account. If you want to program the script in your completed application, then the use of a connector for the asynchronous interface is recommended. With it you can carry out the ISP-programming.

**For this operating mode no additional controller is required on your application!**

The following circuit diagram is an example for how shift register components can be connected to the IC.

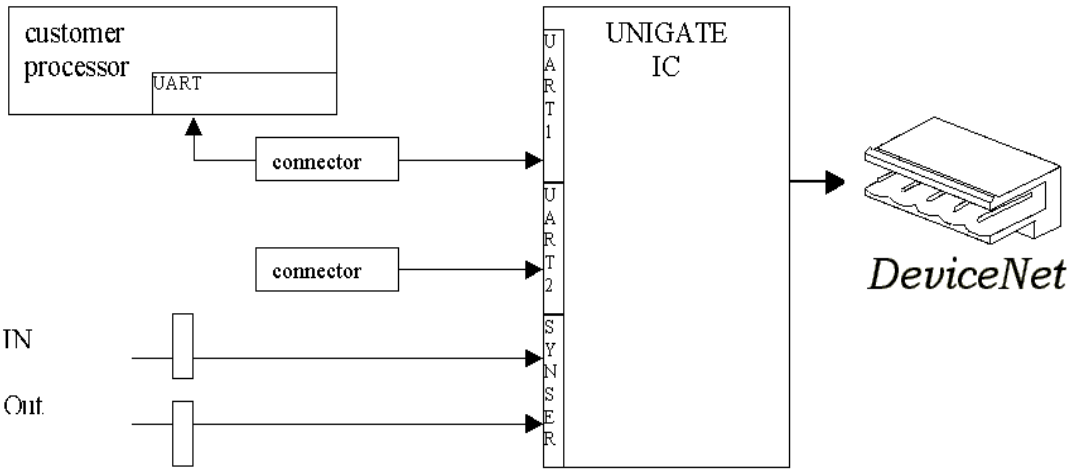



Version 3: Example for digital I/Os

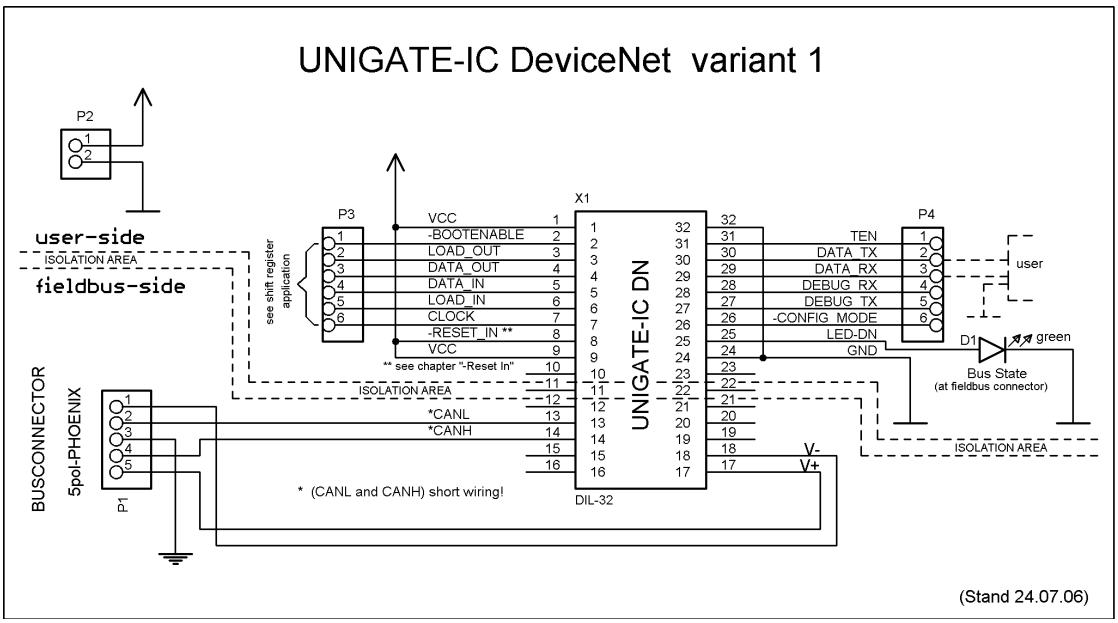
The serial synchronous and the asynchronous interface as well can be operated by UNIGATE® IC at the same time. Here the possibility results that an existing application can be extended by additional digital or analog I/Os.

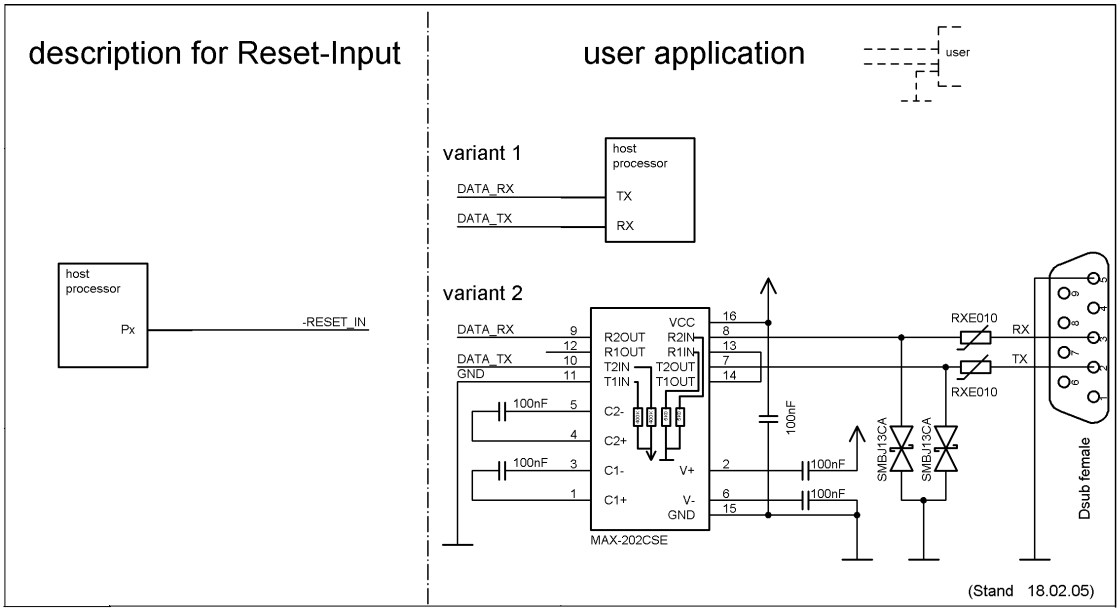
In chapter 5.2 you find an example for a script, that operates these I/Os.

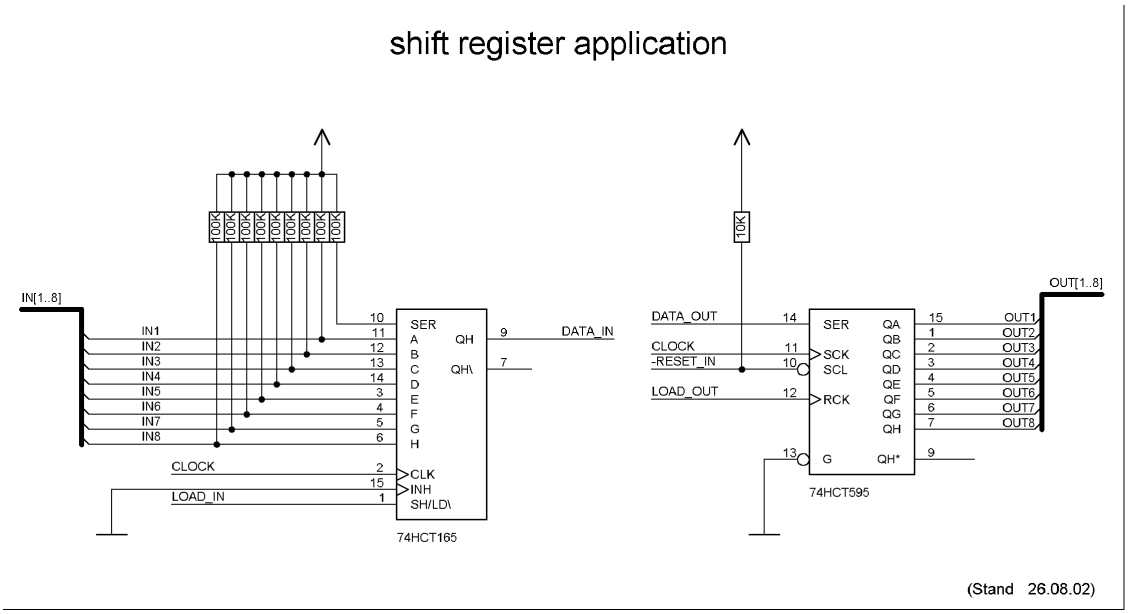
Valid for all versions: A planed plug connection of the serial interface in the application offers the possibility of an update of the firmware or the software via an external connection.

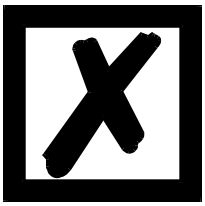


## 3.6 Layout examples

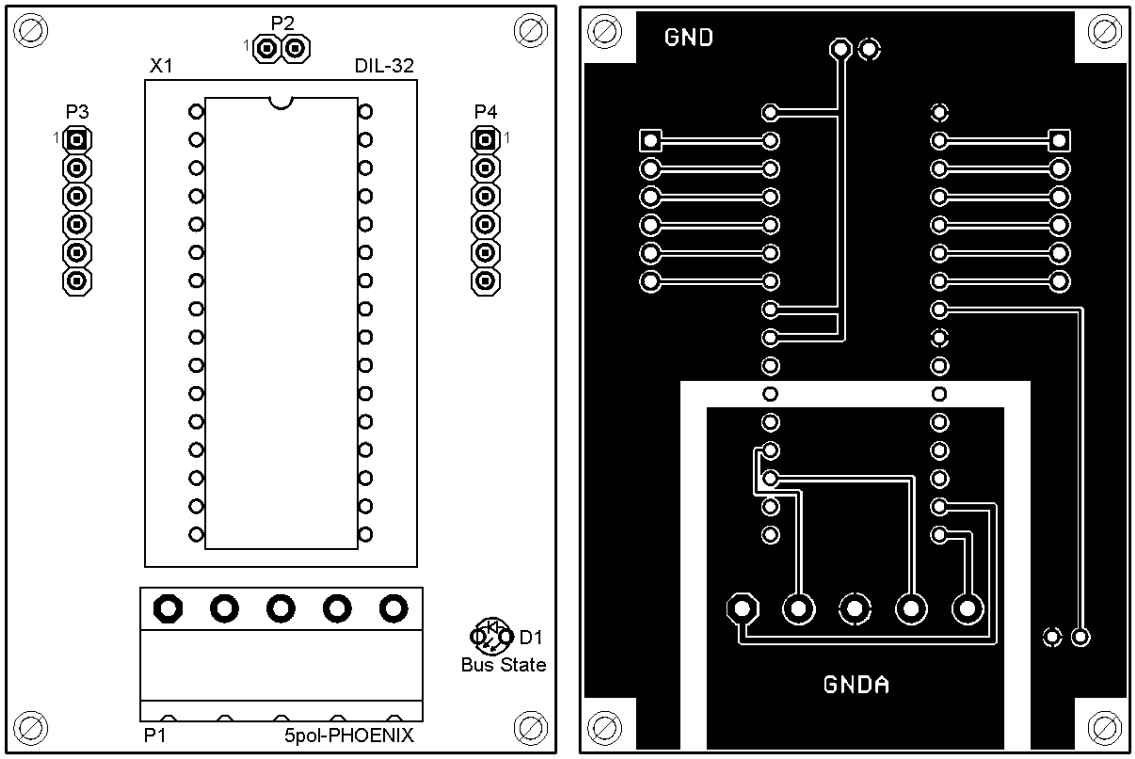




**The 74HCT595 used in this example has an undefined on-position, but therefor can set the outputs to the tri-state condition via the OutEnable-Pin 13. If it is more important to have a defined on-position in an application, and the OutEnable-pin is not necessary, the 74HCT594 can be used here.**


P2
1
X1
DIL-32
P3
1
P4
1
D1
Bus State
P1
5pol-PHOENIX
GND
GNDA

## 3.7 Handling (mounting the UNIGATE® IC on the carrier board)

Depending on the application and the expected shock- and vibration-conditions you can choose from the following possibilities for the UNIGATE® IC's installation on the carrier board:

- Mounting on a socket in the carrier board. If necessary solder the UNIGATE® IC to 2 or 4 pins in the socket. Normally the IC can easily be pulled out after the soldering points have been removed.
- Make arrangements for two holes next to the socket in the layout. After the UNIGATE® IC was plugged in the socket pull an isolated wire over the IC and solder it on the carrier board at the specified holes.
- Fasten the UNIGATE® IC With a wire or a tie wrap on the socket.
- Manual soldering directly on the carrier board.
- Automatic soldering directly on the carrier board, whereas „selective" soldering is essential (no wave soldering)

The advantage of the socketed variant is the easy download of Script- and Firmware-updates, if the carrier board is not designed for it. Besides, that way the Fieldbus can be changed easily by changing the UNIGATE® IC if the corresponding plug connectors are provided on the carrier board. Another advantage is, that - normally - only a reflow soldering of the carrier board is necessary.

The advantage of the soldered variant is, that the installation height is lower and a higher shock- and vibration-safety is provided.

# 4 The serial interface

## 4.1 Overview

The serial interface is the most important connection between the UNIGATE® IC and the micro controller of your application. The interface is designed in a way so that your application at least does not have to be changed on the software-side. The wide range of services of the UNIGATE® IC's serial interface constitutes the basis for it. The UNIGATE® IC allows to connect controllers with a baudrate of 110 baud to 625 kbaud. The baudrate for the communication itself is permanently stored in the module. The maximum size for IO-data can be read-out with the Script command "`Get RSOutBufFree16`".
Depending on the downloaded script of the UNIGATE® IC, the module carries out actions independently, in order to identify data from the connected device. For customers who already have a software-adaptation at he company Deutschmann Automation, this protocol or script as well can be processed by the IC after an adaptation.
Anyway, the IC will take over the communication with the fieldbus independently.

## 4.2 Initialization of the serial interface

The initialization of the interface is carried out by script-commands, such as "`Set baudrate`", "`Set databits`", "`Set parity`". For a detailed description of these commands see the online documentation for the Protocol Developer or the instruction manual for the Protocol Developer.

## 4.3 Use of the serial interface

The serial interface can freely be programmed by the user. Efficient script-commands for sending and receiving data are available; just to mention some possibilities: such as waiting with timeout for a character, waiting for a fixed number of characters or also sending and receiving data in the Modbus RTU. A reference to these commands is offered in the online documentation for the Protocol Developer as well as in the instruction manual for the Protocol Developer.

## 4.4 Further operation modes

In the modes configuration mode and firmware-update mode the serial interface also serves to configure the standard software or to carry out a firmware-update. More details can be found in chapter 11.5.

# 5 Synchronous serial interface

The synchronous serial interface of the UNIGATE® IC is used to connect clocked shift registers or components that have a Serial Peripheral Interface (SPI).

It allows
- the expansion of the IC for digital inputs and outputs (for example for driving LEDS or for reading switch positions)
- communicate with microcontrollers or
- the control of DA and AD converters.

Connection examples are give in chapter 3.
By using the synchronous serial interface can realize products that can work without another microcontroller (stand-alone mode). Examples are sensor products or digital IO modules

## 5.1 Shift register operation

Before the interface can be used, it has to be initialized by setting various Script parameters. (see chapter 5.1.1)
The parameters „`ShiftRegisterInputType`“ and „`ShiftRegisterOutputType`“ allow the use of different shift register types, which differ in the polarity of the shift register signals. To use the shift register types 74595 and 74165, for example, the values „`RiseClk_RiseLoad`“ and „`RiseClk_LowLoad`“ can be set.

The shift register width is set by the parameter „`ShiftRegisterInputBitLength`“ and „`ShiftRegisterOutputBitLength`“ The maximum width is 256 bits.

The data exchange with the connected shift registers ensues with the commands „`WriteShiftRegister`“, „`ReadShiftRegister`“ or bidirectional with the command „`ShiftRegisterDataExchange`“. The clock rate is between 280 kHz and 320 kHz.

Further information on the commands and parameter values can be found in the Help section of the Protocol Developer Software. On request, the Deutschmann Script language can be complemented by additional parameter values in order to support other types of shift registers.

### 5.1.1 Example-Script

Note: The script example refers to the circuit example in chapter 3.5.

```
var InBuffer: Buffer[2];
var OutBuffer: Buffer[2];
MoveConst( OutBuffer[0], #0x58#0x21 );
Set( ShiftRegisterInputType, RiseClock_FallLoad );
Set( ShiftRegisterOutputType, RiseClock_RiseLoad );
```

```
Set( ShiftRegisterInputBitLength, 16 );
Set( ShiftRegisterOutputBitLength, 16 );

WriteShiftRegister( OutBuffer[0] );
ReadShiftRegister( InBuffer[0] );

// Input data is now in the INBuffer
// 0x58 is applied to the outputs of the analog converter
// 0x21 at the shift register's outputs
```

## 5.2 SPI mode

Before the interface can be used in SPI mode, this must be initialized. The command `InitSPI` sets the operating type, the mode (signal polarity and phase) and the clock frequency.
The data exchange ensues with the command `ExchangeSPI`. The maximum clock frequency is between 1 and 5 MHz, depending on the hardware. For details please see the IC-Pinout list in the download area of our website.
Please also refer to the script commands documentation in the online help of the Protocol Developer.

### 5.2.1 Example-Script

```
var   L_Freq     : long;
var   b_Channel  : byte;
var   w_Len      : word;
var   a_BufOut   : buffer[100];
var   a_BufIn    : buffer[100];


moveconst( L_Freq, 1000000); // 1 MHz
InitSPI( 1 , 0 , L_Freq );

moveconst( b_Chanel, 0 );
moveconst( w_Len, 11 );
moveconst( a_BufOut[0], "Hello World" );
ExchangeSPI( b_Channel , w_Len , a_BufOut[0] , a_BufIn[0] );
```

# 6 The Debug-interface

## 6.1 Overview of the Debug-interface

The UNIGATE® IC 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.

## 6.2 Starting in the Debug-mode

When applying power to the UNIGATE® IC (power up) the firmware will output the binary character 0 (0x00) after a self-test was carried out on this interface. If the IC 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.

## 6.3 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 make sure that these settings match those of the PC-COM-interface and that the flow control (protocol) is set to “none“ there.

## 6.4 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.

## 6.5 Commands of the Debug-interface

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

# 7 Script and configuration

## 7.1 Overview

The script stored in the UNIGATE® IC, as well as the configuration, can be replaced or updated via the serial interface (application) in the configuration mode.

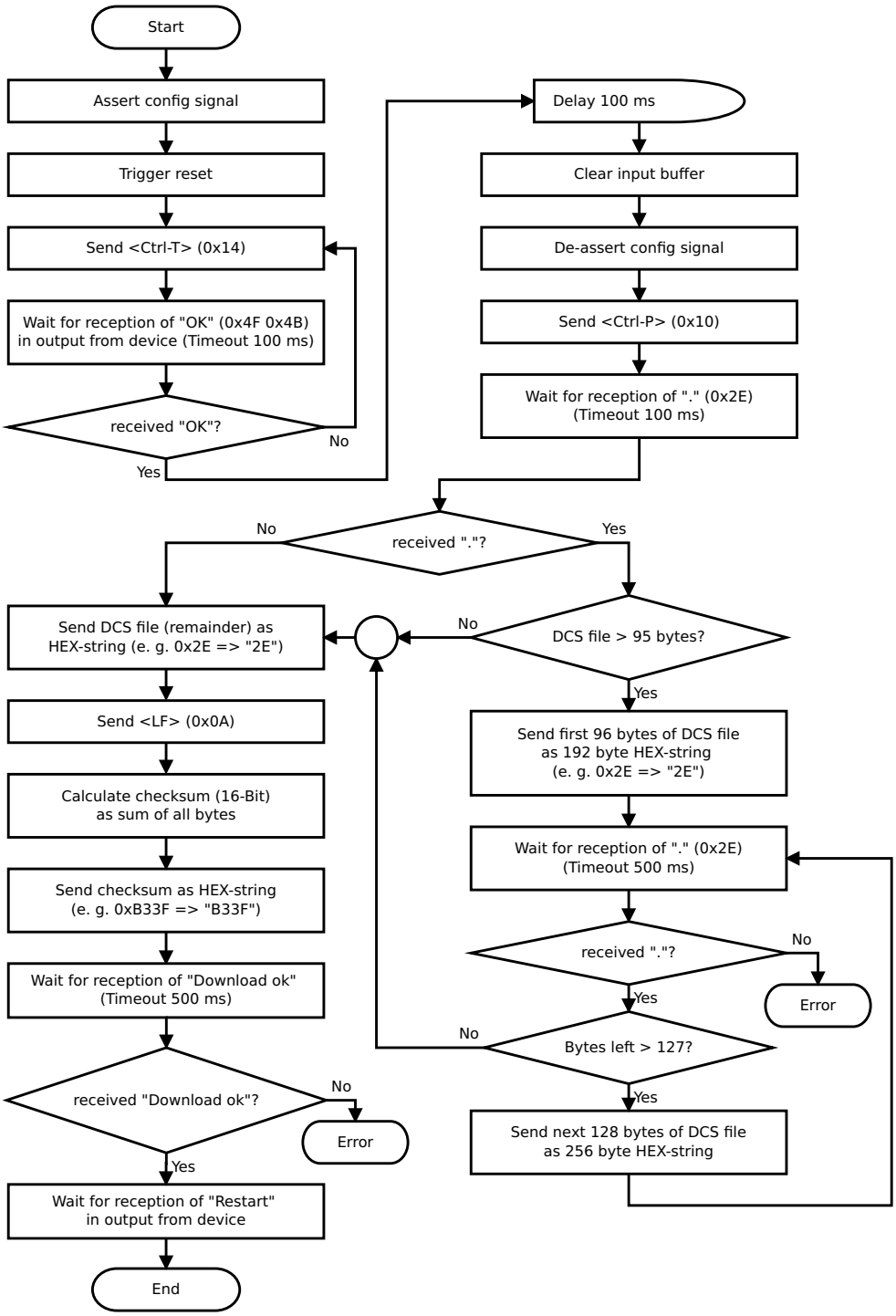
## 7.2 The configuration mode

If the pin „ConfigMode“ pulled to GND during the PowerUp or Reset, then the UNIGATE® IC starts in the configuration mode. In this mode it is possible to communicate with the IC without processing the regular software. In this mode it is possible to change the UNIGATE® IC’s settings of the standard software or to write a new script in the UNIGATE® IC. It shows its start in the configuration mode by issuing a status message, which might look as follows:
"IC-DN-SC V4.03[26] (c)dA Script(8k)="Leer" Author="Deutschmann Automation GmbH" Version="1.0" Date=21.08.2001 SN=47110001“.

## 7.3 Update the script

- The preferred option is to insert the UNIGATE® IC into the base board from Deutschmann (Developer Board UNIGATE® IC-AB) and use the Deutschmann tools (software WINGATE with „Write Script“ under „File“ or with the software ScriptProgramTool)
- You can also automatically replace the script in your application from your host (see flowchart)

## Script Download



**The operational sequence is as follows:**

The Gateway has to be in the config-mode.
The script-download is initiated with "Ctrl-P (=0x10)".
After that the data follows byte by byte as ASCII-hex-characters.
The download is terminated with a "LF (=0x0A)".
Afterwards the word-checksum follows as ASCII-hex-characters.

The Gateway responds with a clear text reply to that download and carries out a warm start.

Example:
The following 4-bytes script is supposed to be downloaded: 0x01 0x12 0x5A 0x23
The sum of the bytes is 0x0090 as checksum.
Then the following sequence is to be sent:

1. 0x10 Ctrl-P
2. 0x30 '0'
3. 0x31 '1'
4. 0x31 '1'
5. 0x32 '2'
6. 0x35 '5'
7. 0x41 'A'
8. 0x32 '2'
9. 0x33 '3'
10. 0x0A LF
11. 0x30 '0'
12. 0x30 '0'
13. 0x39 '9'
14. 0x30 '0'

Gateway's reply: "Download ok"

## 7.4 Configuration of the UNIGATE® IC

UNIGATE® IC is delivered with an empty script.
The configuration of the UNIGATE® IC - DeviceNet is restricted to the setting of the fieldbus address.

### 7.4.1 DeviceNet

- Vendor-ID: 272
- Device-type: 12 (communication adapter)
- MAC-ID: in accordance script setting or configuration data
- Poll-data length: In accordance with script
- Product code: In accordance with script

### 7.4.2 RS232/RS485/RS422

- RS-type: RS232
- Start bit: 1
- Data bits: 8
- Stop bit: 1

- Parity: None
- Baud rate: 9600 Baud

  Default setting This configuration can be changed via the Script.

# 8 Generating a script

## 8.1 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.

## 8.2 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.

## 8.3 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.

## 8.4 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.

## 8.5 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.

## 8.6 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.

## 8.7 Accuracies of the baud rates at UNIGATE® IC

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% |

## 8.8 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.

# 9 DeviceNet

The UNIGATE® CL-DeviceNet module serves to adapt a serial port to DeviceNet in accordance with "DeviceNet Specification Release 2.0". In this application, it functions as a Gateway and operates as DeviceNet "Group 2 Only Slave". It can be operated by any standard-compliant Master.

At present UNIGATE® IC-DeviceNet supports the data exchange in the mode “polling“. The other possible modes “bit-strobe“ and “change of state“ are in preparation. In the mode “polling“ the UNIGATE® IC is at present restricted to up to 255 bytes input and output. Every combination of input- and output-size is possible. However, the EDS-file will not fit any more. Then it becomes necessary either to work without the EDS-file, which is possible for most systems or the EDS-file that is available from our website has to be modified.

## 9.1 Mode of operation of the system

### 9.1.1 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 (UART) to the corresponding Fieldbus system. Layers 3 to 6 are blank, and Layer 7 is converted in accordance with chapter 9.1.3.

### 9.1.2 Interfaces

The gateway is equipped with UART interfaces and an SPI or synchronous serial interface.

### 9.1.3 Data exchange DeviceNet

All data is transferred by the Gateway in dependence of the downloaded Script.
In the DeviceNet the gateway acts as a "Group 2 Only Slave". The access method "Polling", is supported, which is described below.

### 9.1.4 Polling

In the event of Polling the DeviceNet-master sends a telegram according to the configured length to the gateway. The gateway sends out these data via the RS-interface according to the selected protocol. As an answer the DeviceNet-master receives the latest data received by the RS-interface.
The maximum amount of input- out output data to be configured is 255 Byte.

### 9.1.5 Possible data lengths

**The table below shows the maximum transferable data in DeviceNet:**.

| | | |
|---|---|---|
| Input data (Consumed) | max. 255 bytes | Variable: maximum value in this case |
| Output data (Produced) | max. 255 bytes | Variable: maximum value in this case |

## 9.2 Setting the DeviceNet-address

There are different possibilities to set the IC's DeviceNet-address.

1. Setting the address through the configuration
   The UNIGATE® IC has to be in the configuration mode chapter 7.2. With WINGATE it is now possible to set the address. This address is preserved until it is changed again.

2. Setting the address through the Script
   The address can be stored in the Script as well. This proceeding, however, is likely to be of interest for a few applications only, since it is necessary to change the Script in order to also adjust the DeviceNet-address (see also the following Script example).

3. Setting the address through the serial interface
   The address can also be transmitted to the IC through the serial interface. Then the address can be set with the Script command "`SetByVar`". This possibility should be used in case your device has a control front available and the menu of the front can be extended by the setting "*DeviceNet-address*". The adjustment of the PROFIBUS-address through the serial interface is the most convenient possibility for those applications.
   However, as usual for DeviceNet also the baudrate of the DeviceNet should be set through the Dip-switch.

4. Connecting the rotary switches to the shift registers
   Rotary switches can be connected to a shift register as well as to our basis board. Now it is possible for the Script to read those switches and to set them as fieldbus-address. Basically the following Script can be used for it.

## Script example for the initialization of the DeviceNet

```
var InSize: word;
var OutSize: word;

Set (FieldbusID, 4) ;
// this parameter can also be set by the command SetByVar
// var DNAddress: long;
// MoveConst( DNAddress, 4) ; or from the shift registers
// SetByVar();

// Define the baudrate the bus is supposed to be operated with.
// here exemplary 125 kBaud
// This has to take place before the bus-start
Set ( BusBaudrate, 125000 );

// Before the bus-start the participant has to be configured.
// Most important is the setting of the data width,
// Here the values are exemplary.
MoveConst ( Insize, 10) ;
MoveConst ( OutSize, 12 );
SetByVar ( BusInputSize,  InSize );
SetByVar ( BusOutputSize, OutSize );
// Insize and OutSize are from the IC's point of view!!
// Here the values can be set with Set.

// The ProductCode of the participant can also be determined.
// This has to take place before the bus-start
// ! It is not important to set the command Productcode.
// It is possible to set a fixed ProductCode for a script gateway.
// If this value is set to 0 the gateway calculates its product code by
// 256 * consumed size + produced size
// !!! If you like to set a special ProductCode, you MUST set this command after
"BusInPutSize" and "BusOutputSize"

BusStart;
// The DeviceNet is ready. From now the Master CAN configure
// the participant.
// However, this does not mean that the Master has already
// opened up a Poll-connection with the Slave.

wait (Bus_Active);
// The Poll-connection has now been set up by the DeviceNet
// Scanner.
// This command might take a very long time and it cannot
// be interrupted!

// Data can be read out from the bus
// As many bytes as available should be read.
var InBuffer: Buffer[100];
Readbus ( InBuffer[0], InSize) ;

// Now it is possible to write data.
// You must not write more bytes than available.
var OutBuffer: Buffer[100];
WriteBus ( OutBuffer[0], OutSize );
```

# 10 Error handling at UNIGATE® IC

A distinction can be made between two categories of system-errors:
Serious errors (1-4): 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.
The system-error can be read-out via the Script.

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

Table 1: Error handling at UNIGATE® IC

Flashing frequency 2 times per second (system error)

*) The system-error 10 is always displayed when one of the following errors appears:

| Error | Error value |
|---|---|
| DUP_MAC_ERROR | 0x0001 |
| RX_QUEUE_OVERRUN | 0x0002 |
| TX_QUEUE_OVERRUN | 0x0004 |
| IO_SEND_ERROR | 0x0008 |
| CAN_BUS_OFF | 0x0010 |
| CAN_OVERRUN | 0x0020 |
| EXPL_CNXN_TIMEOUT 1) | 0x0040 |
| IO_CNXN_TIMEOUT | 0x0080 |
| IO_CNXN_DELETE | 0x0100 |
| DNS_RESET | 0x0200 |
| DNS_BUS_SENSE_ERROR | 0x0400 |

1) Timeout = Expected Packed Rate * 4!

With the Script command "`Get (DetailErrorCode, w_Error)`" the above value can be read out and with it it can be exactly determined, what error was the actuator.

# 11 Firmware-update

## 11.1 Overview

UNIGATE® IC has a 64 kbyte flash memory for the firmware. In the firmware-update-mode the firmware can be replaced via the UNIGATE® IC's serial interface.

## 11.2 Adjusting the firmware-update-mode

### 11.2.1 Adjustment by hardware

UNIGATE® IC can be brought to the firmware-update-mode by the hardware. For it the signal -BE (boot enable) has to be pulled to the potential GND during the Power-up-process.

### 11.2.2 Adjustment by software

If the UNIGATE® IC is in the configuration mode (see chapter 7.2 on page 27) it can be brought to the firmware-update-mode interactively through the command CTRL-F (0x06). After sending the command a security query follows, that has to be answered with J or N (J = Yes, N = No). After a positive confirmation the IC is re-started in the firmware-update-mode.

## 11.3 Execution of the firmware-update

The safest way for the firmware-update is the use of the basis board combined with the software "FDT.EXE" (firmware-download-tool). These tools are available from Deutschmann Automation (see chapter 13 on page 42).

## 11.4 Note on safety

The firmware-update should only be carried out when there is no other possibility left. A firmware-update-process that has already been started CANNOT be undone. With it the previously used firmware is permanently unusable.

## 11.5 Operation mode of the IC

**Standard-operation mode**
This mode is required for the regular use of the IC. In this mode the IC will process all script-commands and normally exchange the corresponding user data. The bus as well is operated in this mode through the IC.

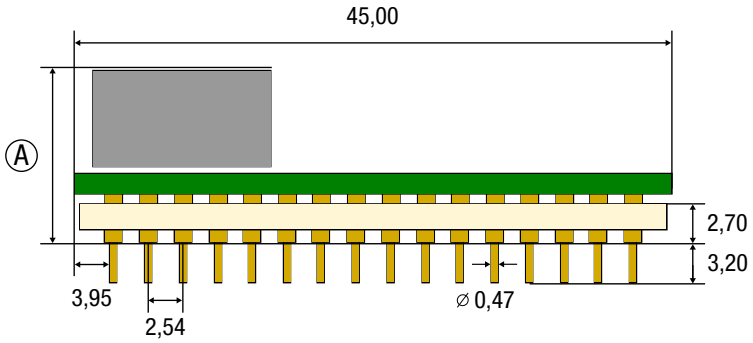
**Configuration mode**
In the configuration mode the UNIGATE® IC will carry out a self-test after the start (or after a reset). After a successful self-test it will wait for further commands. Here it is possible to load a trnslated script into the device or to initialize the firmware-download-mode.

# 12 Technical data

In this chapter you will find all necessary technical data on UNIGATE® IC.
All measurements in mm.

## 12.1 Mechanics of the UNIGATE® IC

### 12.1.1 General dimensions of UNIGATE® IC



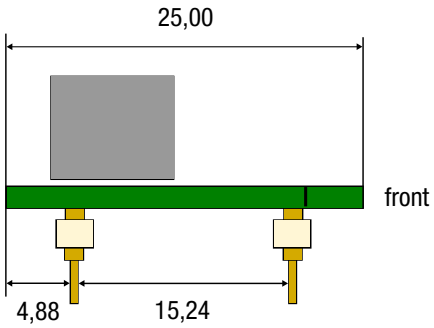

**UNIGATE® IC**

| | CANopen | DeviceNet | EtherCAT | EtherCAT (wt*) | EtherNet/IP 1Port | EtherNet/IP 2Port (wt*) | Ethernet TCP/IP + Modbus TCP | Ethernet TCP/IP (wt*) + Modbus TCP (wt*) | LONWorks | MPI | PROFIBUS DP-DPV1 | PROFIBUS DP (PBL/PBX/PBY) | PROFINET 1Port | PRROFINET 2Port (wt*) | RS |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A= total height (mm) | 12 | 9,5 | 11 | 8 | 11 | 8 | 11 | 8 | 13 | 12 | 9,5 | 12 | 11 | 8 | 12 |

*wt = without magnetics

**Note:** The total height of all UNIGATE® ICs can be found in the "Pinout list for UNIGATE® IC and UNIGATE® IC2". (Download PDF)

The pins of the UNIGATE® ICs are arranged in a grid dimension of 2.54 mm




DIP-Spacing Code 6

In case you intend to use other fieldbus ICs, the maximum overall height of ≤ 20 mm (including pins) has to be taken into consideration.

## 12.2 Technical data UNIGATE® IC-DeviceNet

| Characteristics | Explanation |
|---|---|
| Supply voltage | 5 V ± 5 %, max. 75 mA DC (optionally 3,3V) |
| Reverse polarity protection power supply | No |
| Reverse polarity protection DeviceNet interface | No |
| Interface | 2 UART interfaces, 1 synchronous serial interface |
| Physical separation -fieldbus-side | Standard |
| Fieldbus-ID<br>Fieldbus-baud rate | Adjustable via script<br>Up to 500 Kbaud (adjustable via script) |
| Product code | Adjustable via script |
| I/O-sizes | Adjustable via script at present (0...255 bytes) |
| UART-baud rate | Up to 625 Kbaud (adjustable via script) |
| Fieldbus data format | Group 2 slave |
| Technology | SJA1000 |
| Others | Polling<br>E.g. digital I/Os, analogue signals, shift registers<br>LEDs, switches tec. can be connected externally |
| Dimensions | 45 x 25 x 9,5 mm (WxDxH) |
| Installation | 32 DIL |
| Weight | Approx. 8g |
| Ambient temperature* | -40°C ..+70°C |
| Storage / transport temperature | -40°C..+125°C |
| Built-in position | Any |

* When implementing a UNIGATE® IC in your electronics, not only the mechanical dimensions but also the thermal management must be taken into account. The ambient temperature specified in the manual does not take into account the specific installation situation (e.g. closed housing). External factors have a major influence on how well the heat is dissipated from the heat sources. When using the UNIGATE® IC in your electronics, adequate heat dissipation must therefore be ensured.

# 13 Accessory

The following tools are available from Deutschmann Automation.

## 13.1 FirmwareDownloadTool (FDT)

The FirmwareDownloadTool is available for download from our homepage: it is required for an update of the firmware. Condition for it is, that a PC can be connected to the serial of the IC. The software describes the procedure of an update itself.

## 13.2 Protocol Developer

The Protocol Developer is the development environment for scripts, that also contain the Debugger. This software package also contains the documentation to all script-commands. This software is available for download from our homepage at http://www.deutschmann.de. The instruction manual for the Protocol Developer, which is available in pdf-format, gives further advise on how to use the software.

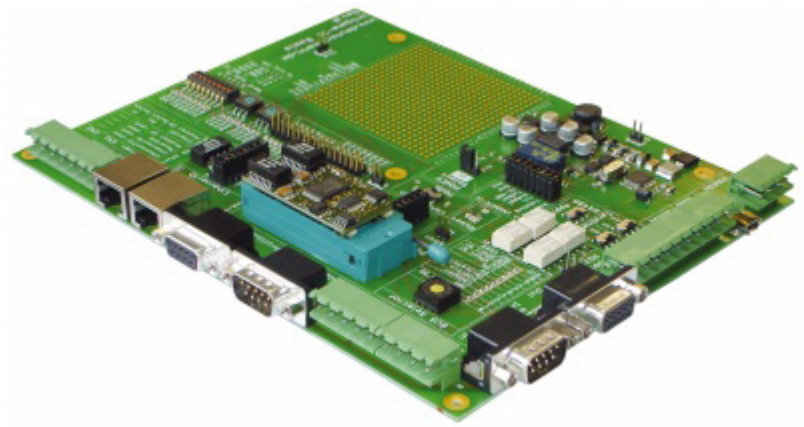
## 13.3 Developer Kit UNIGATE® IC-AB IC

**The Developer Kit IC contains**

- a Developer Board UNIGATE® IC (see chapter 13.3.1)
- a plug-in power pack to supply the Developer Board
- connection cables for appl. RS232, Debug RS232 and appl. RS422/485
- USB-cable
- Software and documentation to complete the packet.

### 13.3.1 Developer Board UNIGATE® IC-AB

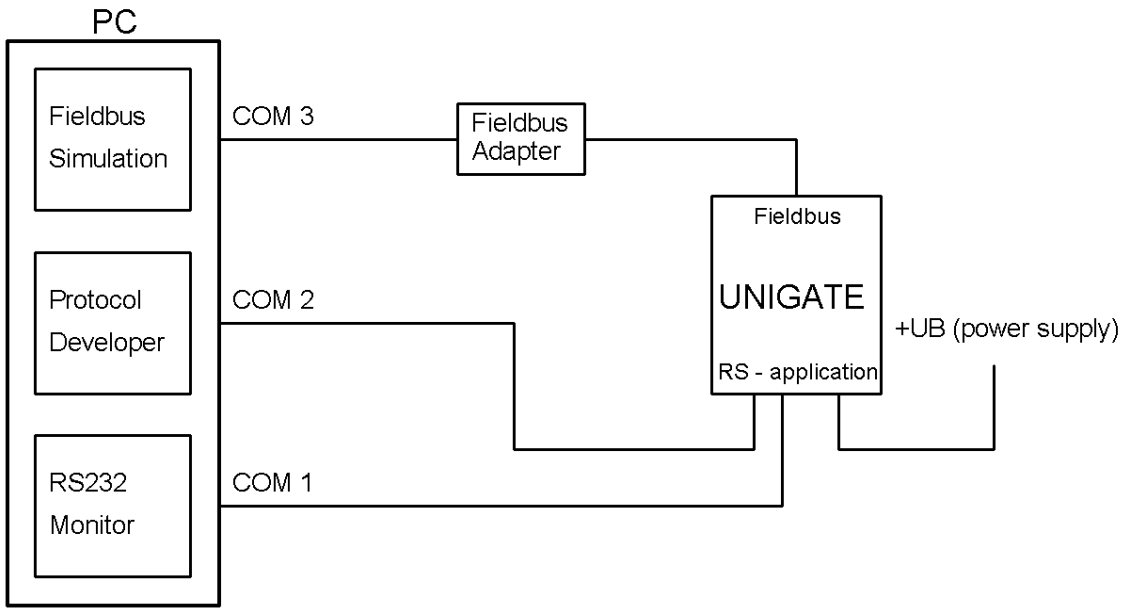
The Developer Board was developed so that the fast implementation of the Deutschmann All-in-one bus node UNIGATE® IC into your electronic system can be guaranteed. The board is suitable for all Fieldbuses and Industrial Ethernet Buses supported by Deutschmann Automation.



The required UNIGATE® IC / ICs are ordered separately. The required voltage (5V or 3.3V, depending on the version) can be adjusted. An RS232-interface or a USB-connection is available for the connection to the PC (Debug-interface).
The application can be connected either through the USB, RS232, RS485 or the RS422.

The bus-connections according to standard or market standard are available to test the respective bus-side. Optionally Deutschmann Add-on packages (bus-master simulation) are available. The board contains 32 bit input and 16 bit output, provided with one LED each. Different connectors allow an easy coupling to your processor. A hole matrix field with the most important signals (voltage, IOs) allows a customized hardware extension (e. g. to connect a D/A converter).

### 13.3.2 Quick start




For a transparent data exchange you will find example scripts for the respective Fieldus under "File->New" in the Protocol Developer.

# 14 Appendix

## 14.1 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 | = | PROFIBUS DP |
| PBDPL | = | PROFIBUS DP-version L (see comparison table UNIGATE® IC for the respective |

product)

PBDPX = PROFIBUS DP-version X (see comparison table UNIGATE® IC for the respective product)
PBDPV0 = PROFIBUS DPV0
PBDPV1 = PROFIBUS DPV1
RS = Serial RS232/485/422

# 15 Servicing

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

- FAQ/Wiki area on our homepage www.deutschmann.com 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 |


## 15.1 Returning a device

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

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

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

## 15.2 Downloading PC software

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