Bridge manual

Conversion DeviceNet Slave to Modbus RTU

Hilscher Gesellschaft für Systemautomation mbH
D-65795 Hattersheim
Rheinstraße 78
Germany

Tel. +49 (0) 6190/9907-0
Fax. +49 (0) 6190/9907-50

Sales: +49 (0) 6190/9907-0
Hotline and Support: +49 (0) 6190/9907-99

e-mail: hilscher@hilscher.com
Web: http://www.hilscher.com

Index	Date	Version	Chapter	Revision
1	25.02.99	V1.000	all	draw up
2	10.05.99	V1.002	all	Errors removed / Improvements

Although this protocol implementation has been developed with great care and intensively tested, Hilscher Gesellschaft für Systemautomation mbH cannot guarantee the suitability of this protocol implementation for any purpose not confirmed by us in writing.

Guarantee claims shall be limited to the right to require rectification. Liability for any damages which may have arisen from the use of this protocol implementation or its documentation shall be limited to cases of intent.

We reserve the right to modify our products and their specifications at any time in as far as this contributes to technical progress. The version of the manual supplied with the protocol implementation applies.

1 Introduction

This manual describes the linking of devices with the Modbus RTU and DeviceNet protocol based on the PKV30-DNS protocol converter.

The Modbus RTU protocol can be configured as master or slave. At the DeviceNet the protocol converter is a slave.

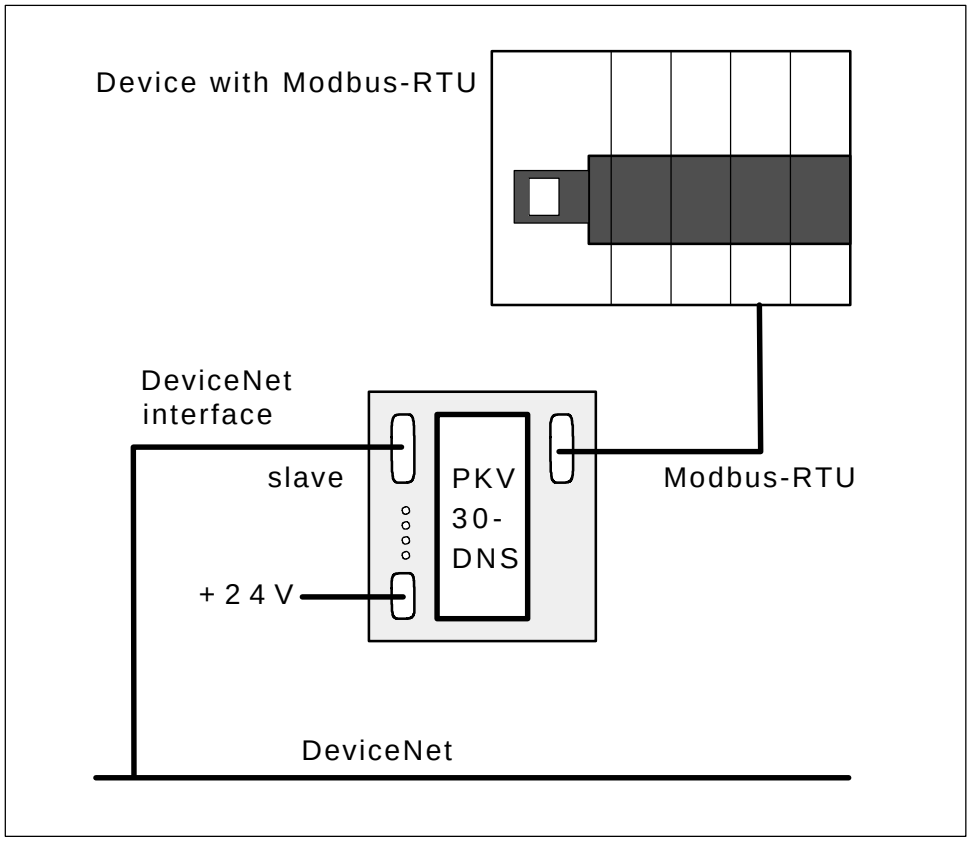
The data exchange is carried out by means of two common process images in the protocol converter. These process images correspond to the input and output data of the DeviceNet and are cyclically exchanged with the DeviceNet master.
The memories can be accessed via the Modbus RTU protocol. The access is different depending on whether the Modbus RTU protocol is operated as a master or a slave:

Modbus RTU as slave	The Modbus RTU master can access the process images directly by means of the function codes 1, 2, 3, 4, 5, 6, 15 and 16.
Modbus RTU as master	A job list is defined on the protocol converter, which the protocol converter processes cyclically, and thus exchanges data between connected Modbus slaves and its process image. For all function codes that write data to the slaves (FC 5, 6, 15 and 16), it can be additionally parametrized whether the data is to be written cyclically or only when the data is changed.

When the Modbus RTU protocol works as a master, then a **supervision** of the Modbus RTU slaves is possible. A bit-coded status field in the input data to the DeviceNet master then provides information on the current condition of the Modbus RTU slaves. The slave supervision is described in the *Status and error bits of the Modbus slave participants* chapter.

Furthermore, in the master operation of the Modbus RTU protocol, a **Watchdog supervision** of the Modbus RTU is possible. In this, the protocol converter reads a particular register (Watchdog register) of a Modbus RTU slave with a cycling time that can be parametrized. If this reading process is unsuccessful, then the failure of the whole Modbus RTU or all the devices connected to it, is assumed. In this case **all** input data to the DeviceNet master are set to **zero**. The inputs to the DeviceNet master are again served when the Watchdog register becomes available again. The Watchdog supervision is described in *The Watchdog supervision* chapter.

The parametrizing and diagnostic program ComPro is used for the configuration. This tool is described in its own manual.



Connection of the devices to the PKV 30-DNS

2 Data image of the data in the protocol converter

The data exchange is carried out via the process images in the protocol converter. This consists of a maximum of 128 input words and 128 output words. How much data is exchanged over the bus depends on the configuration of the DeviceNet and is independent of the accesses on the part of the Modbus RTU. Here is just tested whether the access is within the data region. It is up to the user to configure the DeviceNet such that relevant process data is available at the corresponding positions of the process image. Up to 255 bytes can be exchanged for input and output data at the DeviceNet.

2.1 Data formation from the viewpoint of the DeviceNet

The meaning of inputs and outputs must always be seen from the viewpoint of the DeviceNet master.

Input data	is always data that is received from the DeviceNet master.
Output data	is always data that is output from the DeviceNet master.

The input and output data of the common memory is exchanged right from the start. Only the amount of data is determined by the configuration. The arrangement for the function codes 3, 6 and 16 between the Modbus address (Mem.Adr. in the COMMAND table) and the process images looks as follows:

	DeviceNet - Bytes in the common memory		Mem.Adr.
Input data of the DeviceNet (Write from the Modbus side with FC 6 and 16)	1. Byte	2. Byte	40.001
	3. Byte	4. Byte	40.002
	5. Byte	6. Byte	40.003
	...	...	...
	...	Last byte	
	...	...	40.128
Output data of the DeviceNet (Read from the Modbus side with FC 3 and 4)	1. Byte	2. Byte	40.001
	3. Byte	4. Byte	40.002
	5. Byte	6. Byte	40.003
	...	...	...
	...	Last byte	
	...	...	40.128

Imageing of the DeviceNet data on the Modbus data

This means that there is always depicted the **lowest valid Modbus address** (here 40.001) **for the function code used** (here 3, 6 or 16) at the start of the process image! Correspondingly, the function code 4 in connection with the Modbus address 30.001 is **also** depicted at the start of the process image. The same is valid to the access to bit markers with the function codes 1, 2, 5 and 15.

The access to the process image is also possible bit-wise (bit marker, coils) with the function codes 1, 2, 5, and 15. The following picture shows how the individual bits are arranged in the common process image.

		Register	Arrangement of the bit markers / coils in the common memory															
Input data of the DeviceNet (Write with FC 5 and 15)	1st word	40.001	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
	2nd word	40.002	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
	3rd word	40.003	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
	Last word	40.128	2048	2047	2046	2045	2044	2043	2042	2041	2040	2039	2038	2037	2036	2035	2034	2033
Output data of the DeviceNet (Read with FC 1)	1st word	40.001	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
	2nd word	40.002	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
	3rd word	40.003	48	47	46	45	44	43	42	41	40	39	38	37	36	35	34	33
	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
	Last word	40.128	2048	2047	2046	2045	2044	2043	2042	2041	2040	2039	2038	2037	2036	2035	2034	2033

Arrangement of the bit markers in the common process images

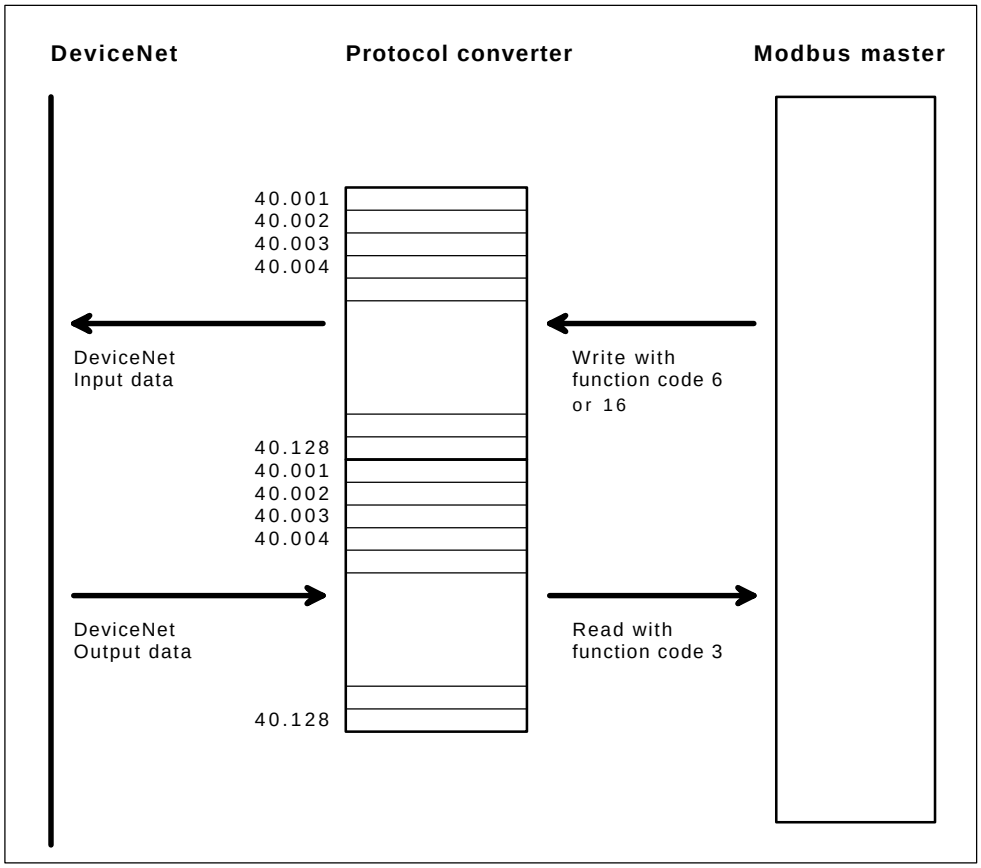
The access to the output data of the DeviceNet via the function code 2 thus begins at the Modbus address 10.001.

In contrast to controllers from the Modicon family, the registers and the coils (bit markers) in the protocol converter are situated above each other in the process image. Thus, word- or bit-wise access of the same data can be selected!

2.2 Data image from the viewpoint of the Modbus RTU Slave

If the protocol converter is a slave (server) at the Modbus RTU, then reading and writing can occur on the common memory with the functions of the Modbus RTU.

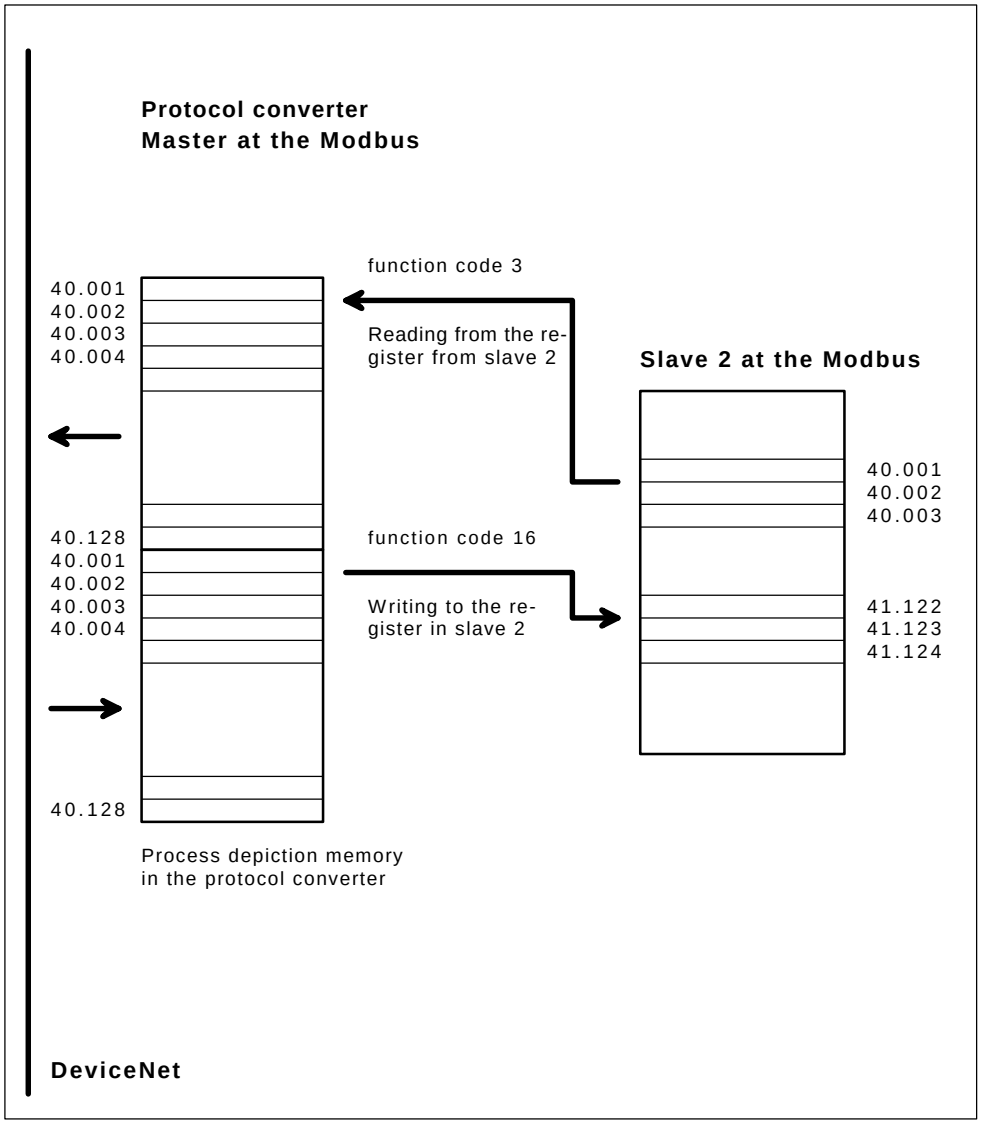
The data image between the addresses on the DeviceNet and the Modbus RTU is described with the following picture as an example of the function codes 3, 6 and 16 for the process image.



Data transfer between protocol converter as slave at the Modbus RTU and Modbus master

2.3 Data image from the viewpoint of the Modbus RTU master

If the protocol converter is a master at the Modbus RTU, then, on the basis of the commands defined in the COMMAND list, it independently generates read and write commands between the process image and the slaves connected to it.



Data exchange between the protocol converter and slave devices at the Modbus RTU

2.4 Status and error bits of the Modbus slave participants

If the protocol converter is working as a **master** at the Modbus, a status or error field can be utilized in the process image (Supervision). This must be configured in the SUPERVIS table. The following options are available:

Off	No status/error field is utilized. This means that all slave participants must work without error, otherwise the protocol converter closes down the communication at the DeviceNet.
CommandStatus	A separate status bit is utilized for each command of the COMMAND table. The bit is set when the command has been carried out without error. The Bit is deleted in case of error.
CommandError	An error bit is utilized for each command of the COMMAND table. The bit is set when an error has occurred in carrying out the command. The bit is deleted again when the next error-free command has been carried out.
SlaveStatus	A separate status bit is utilized for each slave. The bit is set when a command to this slave is carried out without error. The bit is deleted in the case of error.
SlaveError	An error bit is utilized for each slave. The bit is set when an error occurs during a command to this slave. The bit is deleted again when the next command has been carried out without error.

This defines whether a bit that is set in the process image denotes an error-free or an error transmission. Furthermore, the bits can represent alternatively a slave or a command. Thus the lowest value bit is allocated to the slave with the address 1 or to the first command in the COMMAND table.
The allocation of the bits is taken from the interpretation in the process image.

The starting address of the bit field in the process image is defined by means of the *Start Register* parameter. In the *Quantity Register* parameter is indicated how many registers are included in the status field. If a bit lies outside the defined field, then it is not considered. If the **quantity is set to 0,** then no status field is utilized.

The following interpretation shows which data is shown in the bit field with the various configurations.

As an example, the following configuration is assumed:

1st slave	Address 2
2nd slave	Address 3
3rd slave	Address 4
4th slave	Address 20

1st command	Read from 1st slave
2nd command	Read from 2nd slave
3rd command	Read from 3rd slave
4th command	Write to 3rd slave
5th command	Read from 4th slave

The first register for the bit field given is 40.010 (*Start Register* = 40.010) and two register is given as the length of the bit field (*Register Quantity* = 2); the 2nd slave is switched off:

	Register	Arrangement of the bit field in the process image														
Input data	40.009	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1st word	**40.010**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**1**	**0**	**1**
2nd word	**40.011**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	40.012	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Outputs in the bit field during supervision mode = CommandStatus

	Register															
Input data	40.009	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1st word	**40.010**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
2nd word	**40.011**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	40.012	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Outputs in the bit field during supervision mode = CommandError

	Register															
Input data	40.009	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1st word	**40.010**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**1**	**0**	**0**
2nd word	**40.011**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**
	40.012	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Outputs in the bit field during supervision mode = SlaveStatus

	Register															
Input data	40.009	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1st word	**40.010**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**1**	**0**
2nd word	**40.011**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
	40.012	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Outputs in the bit field during supervision mode = SlaveError

2.5 The Watchdog supervision

A Watchdog supervision is possible in the **master operation** of the Modbus RTU protocol. Here, the protocol converter reads out a particular register (Watchdog register) of a Modbus RTU slave with a cycling time that can be parametrized. If this read process is flawed, then the failure of the Modbus RTU or its connected devices is assumed. In this case **all** input data (with the exception for Supervision, see below) to the DeviceNet master are set to **zero.** When the Watchdog register is available again, then the inputs to the DeviceNet master are served again.

Remarks:

- If registers are reserved for the Slave supervision (status and error bits), then these are **not** altered!
- The Watchdog supervision is carried out cyclically with the cycling time parametrized in the *Watchdog time* in the WATCHDOG table. However the cycling time cannot be guaranteed for the following reasons:
    - It can occur that the Modbus slave used for the Watchdog supervision answers too late. The protocol converter waits, according to its setting (MODBUS table *Timeout* parameter), for example, for one second for the answer from the Modbus slave. This fact must be considered in the choice of the *Watchdog time*. Normally, the Watchdog supervision requires approx. 20..50ms for a baud rate of 9,600 baud.
    - A priority switching guarantees that, **at least** one command of the COMMAND table is processed after every Watchdog supervision. This means at the same time that the Watchdog Supervision cannot completely block the data transfer. However, the Modbus side data throughput can be greatly reduced if the *Watchdog time* is selected too be too short (e.g. 100 ms) as a Watchdog supervision is inserted between every command of the COMMAND table.

3 Configuration

The transfer parameters of the two protocols are defined by the configuration:

Modbus RTU	The parameters of the MODBUS table must be entered. This determines the whole behavior of the Modbus RTU.
DeviceNet	The configuration is carried out by means of the ComPro. Thus, only the data model needs to be defined in the BUS_DNS table. The Bus address is set on the address switches on the protocol converter.

For configuration, the protcol converter must be switched into the configuration and diagnostic mode.
This is described in the device manual of the PKV30-DNS.

Only a configuration as such is necessary for the protocol conversion, if the Modbus RTU protocol is to be operated in the master mode. Then the cyclical commands and commands of the protocol converter generated by the changes in the data must be defined in the COMMAND table. If a status or error field is to be utilized, then the SUPERVIS table must be defined. If the Watchdog supervision is required, it must be defined in the WATCHDOG table.

3.1 Editing the MODBUS table

Parameter	Meaning	Value range
Communication line	Gives the device interface that is served by the protocol.	**1**
RTS Control	If the RTS control is switched on, then the control lines RTS and CTS are served by an RS232C interface or by an RS485/RS422 interface that switches through the data drivers only while sending. This does not alter the protocol sequence.	**no** yes
Baud rate	Determines the rate of data transfer.	50 Baud 100 Baud 110 Baud 150 Baud 200 Baud 300 Baud 600 Baud 1200 Baud 2400 Baud 4800 Baud **9600 Baud** 19200 Baud
Stop bits	Defines the number of stop bits.	**1**, 2
Parity	Defines the parity bit.	none **even** odd
Mode	Defines the operating mode.	slave **master**
Modbus address	Gives the own address at the Modbus.	**1**,2...247
Timeout	Master mode: Gives the maximum waiting time in milliseconds for an answering telegram from a slave. Slave mode: Gives the maximum waiting time for an answer from the application.	10...**1000**...10000
Retries	Defines the number of retries of the telegram repeats in an error case. Relevant only in the Master operation.	0...**3**...10
Error-LED	This defines the operating mode of the error LEDs	**set/clear** only set

Parametrizing the Modbus RTU protocol in the MODBUS table.

3.2 Editing the BUS_DNS table

Parameter	Meaning	Value range
Slave MAC ID	Defines the DeviceNet address in the network.	**0** - 63
Vendor ID	DeviceNet specific manufacturer number, allocated to every manufacturer. Here: 283 for Hilscher; not changeable.	**283**
Prod.size	Defines the number of the input data in bytes.	0 - **8** - 255
Cons.size	Defines the number of output data in bytes.	0 - **8** - 255

Parametrizing the DeviceNet protocol in the BUS_DNS table

3.3 Editing the COMMAND table

This table is only relevant when the protocol converter operates as master on the Modbus RTU. A maximum of 300 commands can be defined. Each command consists of the following parameters. The default values are shown in bold and are underlined:

Parameter	Meaning	Value range
Slave	Gives the slave address from which, or into which the protocol converter conveys the data.	0 - **2** - 255
Function	Gives the function code for this access.	1 / 2 / **3** / 4 / 5 / 6 / 15 / 16
Address	Gives the address in the slave device.	1..9.999 10.001..19.999 30.001...39.999 **40.001**..49.999
Quantity	Gives the amount of data to be transmitted.	1 - **2** - 255
Mem.Addr.	Gives the address in the process image of the protocol converter in which the protocol converter stores the data or from which the protocol converter takes the data. The respective starting address (e.g. 40.001) is then always given at the start of the process image.	1..128 10.001..10.128 30.001..30.128 **40.001**..40.128
Write	Defines for all writing function codes (FC 5, 6, 15 and 16) whether the command is to be triggered cyclically or only by a change of data. This parameter has no significance for the remaining function codes.	**Cyclic** Change

Attention:
Addresses in accordance with Modbus convention are valid.

Defining the Master commands in the COMMAND table

When entering the parameters, care must be taken that these, in fact, address valid registers.

A delay time can be parametrized between the individual commands. This can be necessary in order to prevent the loading becoming too high on the connected Modbus slaves through uninterrupted communication.
The parametrizing is carried out in the SUPERVIS table.

3.4 Editing the SUPERVIS table

In this table, there is defined whether a bit field is utilized and if so, its function.

Parameter	Meaning	Value range
Supervision Mode	No bit field Status bit field for commands Error bit field for commands Status bit field for Slaves Error bit field for Slaves	**off** CommandStatus CommandError SlaveStatus SlaveError
Start Register	Start address of the bit field. The first 16 bits of the bit field are entered into this register	**40.001**..40.128
Quantity Register	Number of Registers of the Bit field. 0 = No bit field is entered.	**0**..128
Delay [msec]	Delay time between the individual commands in the Modbus master mode in milliseconds.	**0**..65535

Parameter list

If the Supervision mode is set to *off*, then, the DeviceNet master has **no possibility** to check the data exchange with the Modbus slaves. In the other modes, the bit field is transfered to the DeviceNet master. This is then responsible for the evaluation and display of the error condition at the Modbus.

3.5 Editing the WATCHDOG table

In this table the Watchdog supervision is parametrized.

Parameter	Meaning	Value range
Watchdog	off = No Watchdog supervision The following parameters then have no significance. on = Watchdog supervision active.	**off** on
Watchdog address	Address of the Modbus slave.	1..**2**..247
Watchdog Register	Modbus address of the Watchdog register	**40001**..49999
Watchdog time	Cycling time of the Watchdog supervision in milliseconds.	10..**1000**..60000

Parameter list

4 Error messages

The following tables show the error messages of the individual protocols. These can be displayed with the aid of the ComPro program. Errors are also displayed by means of two error LEDs on the protocol converter (see chapter *Setting up the converter as a Slave at the DeviceNet*).

4.1 Error messages from the Modbus RTU

Error number	Error
10	Serial interface occupied Interface occupied The serial interface has already been initialized by another task. Error in the “Interface” parameter.
11	Sum of all baudrates to high Sum of all baud rates has been exceeded. The sum of all baud rates on all initialized interfaces is too high.
12	Error 'Communication line' Error “Interface” Parametrized interface on the device is not available.
13	Error 'Baudrate' Error “Baudrate”. Invalid value for the “Baud rate” inititializing parameter.
14	Error 'Parity' Error “Parity” Invalid value for the “Parity” inititializing parameter.
15	Error 'Databits' Error “Data Bits” Invalid value for the “Data Bits” inititializing parameter.
16	Error 'Stopbits' Error “Stop Bits” Invalid value for the “Stop Bits” inititializing parameter.
17	Error 'RTS-Control' Error “RTS control” Invalid value for the "RTS-Control" inititializing parameter.
50	Error 'Mode' Error “Mode” Invalid value for the "Mode" inititializing parameter.
51	Error 'Modbus address' Error “Modbus address” Invalid value for the "Modbus address" inititializing parameter.
52	Error 'Timeout' Error “Timeout” Invalid value for the "Timeout" inititializing parameter.
53	Error 'Retries' Error “Retries” Invalid value for the "Retries" inititializing parameter.
54	Error 'Error-LED' Error “LED Operation”' Invalid value for the "Error-LED" inititializing parameter.

Initializing errors

Error number	Error
100	Parity error Parity error The interface controller has detected a parity error.
101	Framing error Character frame error The interface controller has detected a parity error.
102	Overrun error Loss of received data The interface controller has detected an “overrun” error..
103	To much/less data received Too much/too little data received. More data has been received than can be accommodated in the receiving buffer or too little to make a proper telegram testing possible.
104	CRC error CRC error A checksum error has been recognized in the receiving telegram.
105	Timeout telegram Time monitoring error. A time monitoring error has occurred as no answer has been received from the Slave within the defined Slave waiting time.
110	Unknown exception received Unknown exception received. a non-defined “exception response code”, e.g. 0 or greater than 9 has been received.
111 ... 119	Exception 1 ... 9 received Exception 1 ... 9 received The accessed Slave has answered with an “exception response code”.
120	Invalid 'slave address' received Invalid Slave address received The Slave address in the answering telegram is not that of the addressed Slave.
121	Invalid 'function code' received Invalid function code received. The function information in the received Modbus telegram does not correspond with the issued function.
122	Invalid 'bytecount' received Incorrect “bytecount” received. The displayed amount of data in the received Modbus telegram (bytecount) does not correspond with the amount of data transferred.
123	Too much/less data received Too little/ too much data received. Incorrect amount of data in the received Modbus telegram.
124	Invalid data address received Invalid data address received. Incorrect data address in the Modbus telegram.
125	Invalid answer data received Invalid answer data received. The answering telegram from the Slave has been correctly received but does not correspond with the command telegram.
126	Invalid diagnostic code received Invalid diagnostic code received. Only the diagnostic code 0, i.e. “loopback test” is permissible.

Communication errors

Error number	Meaning
151	Invalid length of message Invalid length of message The transferred amount of useful data for output as Modbus telegram is incorrect.
152	Unknown message command Unknown message command. The message command Msg.B is invalid.
154	Message error received Message error is set. In Slave mode, the coupling partner returns an error instead of the data asked for or a confirmation.
155	Timeout message Time monitoring error message. A time monitoring error has occurred as no answer has arrived from the application program within the configured Timeout period.
160	Invalid telegram header in acknowledge message A different telegram header in the answering message. The application Program has returned a different telegram head in the command confirmation.
161	Error 'device address' Error “device address” The Slave address in Msg.DeviceAdr was entered incorrectly by the application program.
162	Error 'data area' Error “data area”. The data area in Msg.DataArea was entered incorrectly by the application program.
163	Error 'data address' Error “data address”,
165	Error 'data quantity' Error “data quantity” The data address in Msg.DataAdr was entered incorrectly by the application program.
166	Error 'data type' Error “data type” The data type in Msg.DataType was entered incorrectly by the application program.
167	Error 'function' Error ”function” The function code in Msg.Function was entered incorrectly by the application program.

Protocol and message errors

4.2 Error messages of the DNSMBR bridge

Error number	Error
10	Serial interface occupied Interface occupied The serial interface has already been initialized by another task. Error in the “interface” parameter.
11	Sum of all baudrates to high The sum of all baud rates has been exceeded. The sum of all baud rates on all initialized interfaces is too high.
50	Modbus data base missing Error in opening the data base - table MODBUS.
51	Command error 'Function' Incorrect Modbus function code in COMMAND table.
52	Command error 'Quantity' Incorrect amount of data in COMMAND table.
53	Command error 'Range of Address/Quantity' COMMAND table: the combination of Modbus address “address” and the amount of data “quantity” has exceeded the permitted area.
54	Command error 'Mem.Add.' Incorrect address in the process image of the protocol converter in the COMMAND table.
55	Command error 'Address' Incorrect Modbus address in COMMAND table.
60	SUPERVIS data base missing Error when opening the database - SUPERVIS table.
61	WATCHDOG data base missing Error when opening the database - WATCHDOG table.

Initializing errors

Error number	Error
200	Task not initialized The task could not be initialized.
210	Error at opening the data base The parameter database is not available.
212	Error at reading the data base The parameter database is inconsistent.
213	System error 'RcsPutStructure' Internal error.

Internal system errors

5 Setting up the converter as a Slave at the DeviceNet

The following sequence must be adhered to in the set up of the protocol converter as a slave:

- A valid configuration database must be saved on the protocol converter. In the supply, the indicated default parameters have been set.
- The bus address must be set on the protocol converter at the address switches.
- The coupling partner on the Modbus-RTU must be connected. Please consult the device manual for configuring the physical interface and fitting the cable.
- Configuring the DeviceNet master with the aid of the included EDS file. In the master configuration, exactly the same information must be input as in the BUS_DNS table.
- Connecting the DeviceNet cable and the supply voltage.
- The LED RDY and RUN must light up and must not blink.
- The LEDs signal the following conditions:

Display	Color	Condition	Meaning
RDY	yellow	on blinks cyclically blinks irregularly off	PKV ready Bootstraploader active Hardware or system error Hardware defective
RUN	green	on blinks irregularly blinks regularly off	Communication running Parametrizing error Ready for communication No communication
NET	red	on	Critical link failure
		blinking	Connection time out
		off	Device not powered
	green	on	On-line, link OK
		blinking	On-line, not connected
		off	Device not powered
MOD	red	on	Unrecoverable fault
		blinking	Minor fault
		off	No power
	green	on	
		blinking	Configuration failure
		off	No Power

LED indications of the PKV30-DNS