

## 6 Implementation of Modbus TCP

<b>6.1</b>	<b>Common Modbus description.....</b>	<b>6-2</b>
6.1.1	Protocol description.....	6-3
6.1.2	Data model .....	6-4
<b>6.2</b>	<b>Implemented Modbus functions.....</b>	<b>6-6</b>
<b>6.3</b>	<b>Modbus registers.....</b>	<b>6-7</b>
6.3.1	Structure of the packed in-/ output process data .....	6-12
	– Packed input process data.....	6-12
	– Packed output process data.....	6-13
	– Data width of the I/O-modules in the modbus-register area .....	6-14
6.3.2	Register 0x100C: Gateway status.....	6-16
6.3.3	Register 0x1130h: Modbus-Connection-Mode .....	6-17
6.3.4	Register 0x1131: Modbus-Connection-Timeout.....	6-17
6.3.5	Register 0x113C and 0x113D: Restore Modbus-connection parameters.....	6-17
6.3.6	Register 0x113E and 0x113F: „Save Modbus-Connection-Parameters“ .....	6-18
6.3.7	Register 0x1140: Disable protocol .....	6-18
6.3.8	Register 0x1141: Active protocol.....	6-18
6.3.9	Register 0x2000 bis 0x207F: The Service-Object .....	6-19
	– Indirect reading of registers .....	6-20
	– Indirect writing of registers .....	6-21
<b>6.4</b>	<b>Bit areas: mapping of input-discrete- and coil-areas .....</b>	<b>6-22</b>
<b>6.5</b>	<b>Error behavior of outputs (watchdog).....</b>	<b>6-23</b>

6.1 Common Modbus description



**Note**  
The following description of the Modbus protocol is taken from the Modbus Application Protocol Specification V1.1 of Modbus-IDA.

Modbus is an application layer messaging protocol, positioned at level 7 of the OSI model, that provides client/server communication between devices connected on different types of buses or networks.

The industry’s serial de facto standard since 1979, Modbus continues to enable millions of automation devices to communicate. Today, support for the simple and elegant structure of Modbus continues to grow.

The Internet community can access Modbus at a reserved system port 502 on the TCP/IP stack.

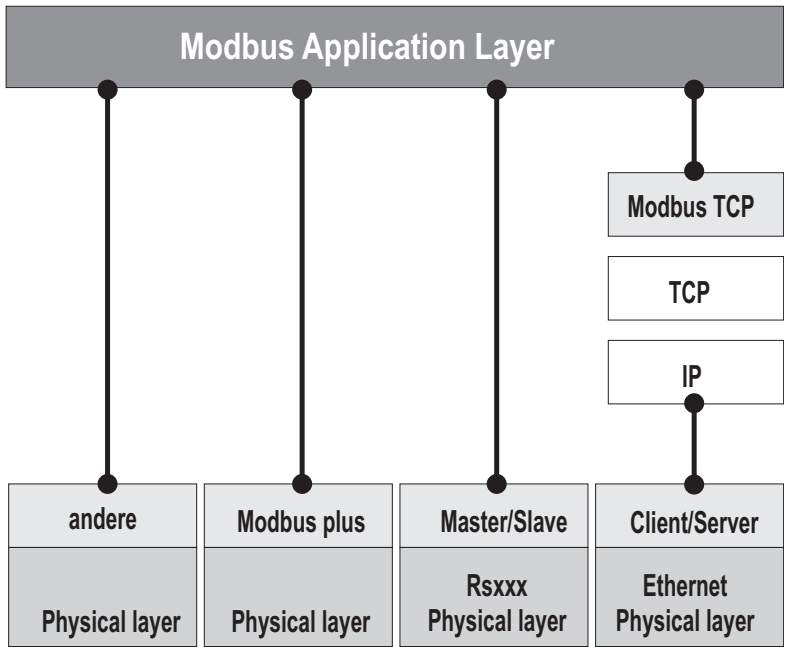
Modbus is a request/reply protocol and offers services specified by function codes. Modbus function codes are elements of Modbus request/reply PDUs (Protocol Data Unit).

It is currently implemented using:

- TCP/IP over Ethernet. (that is used for the BLxx-gateways for Modbus TCP and described in the following)
- Asynchronous serial transmission over a variety of media (wire: RS232, RS422, RS485, optical: fiber, radio, etc.)
- Modbus PLUS, a high speed token passing network.

Schematic representation of the Modbus Communication Stack (according to Modbus Application Protocol Specification V1.1 of Modbus-IDA):

Figure 6-1:  
Schematic  
representation  
of the Modbus  
Communica-  
tion Stack

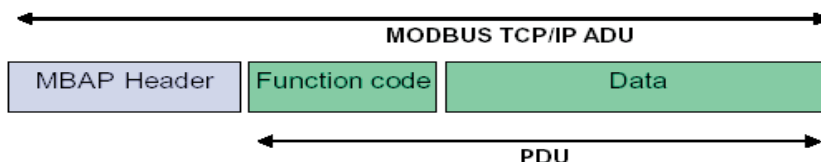


### 6.1.1 Protocol description

The Modbus protocol defines a simple protocol data unit (PDU) independent of the underlying communication layers.

The mapping of Modbus protocol on specific buses or network can introduce some additional fields on the application data unit (ADU).

Figure 6-2:  
Modbus tele-  
gram acc. to  
Modbus-IDA



The Modbus application data unit is built by the client that initiates a Modbus transaction.

The function code indicates to the server what kind of action to perform.

The Modbus application protocol establishes the format of a request initiated by a client.

The field function code of a Modbus data unit is coded in one byte. Valid codes are in the range of 1... 255 decimal (128 – 255 reserved for exception responses).

When a message is sent from a Client to a Server device the function code field tells the server what kind of action to perform. Function code "0" is not valid.

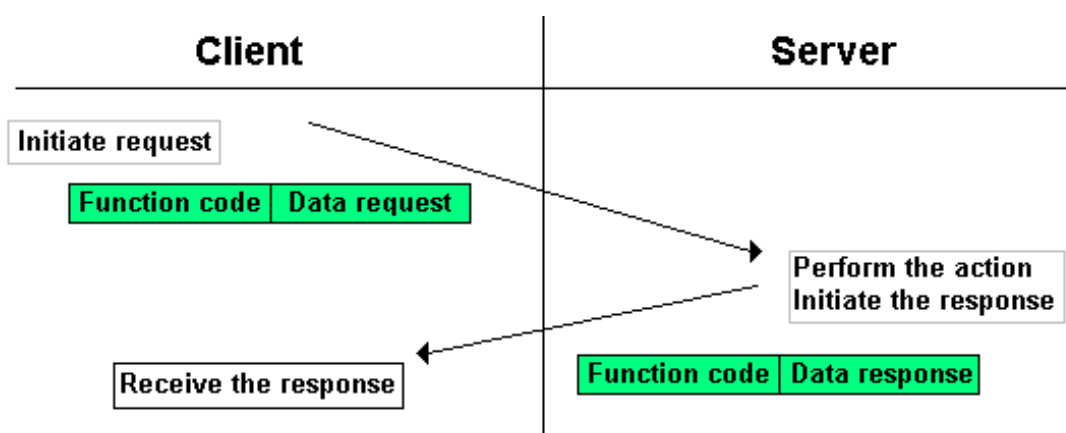
Sub-function codes are added to some function codes to define multiple actions.

The data field of messages sent from a client to server devices contains additional information that the server uses to take the action defined by the function code. This can include items like discrete and register addresses, the quantity of items to be handled, and the count of actual data bytes in the data field.

The data field may be non-existent (= 0) in certain kinds of requests, in this case the server does not require any additional information. The function code alone specifies the action.

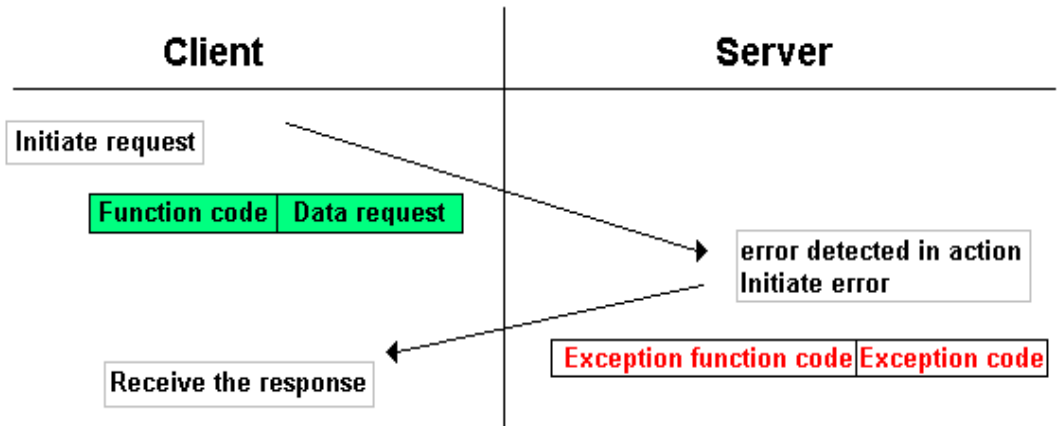
If no error occurs related to the Modbus function requested in a properly received Modbus ADU the data field of a response from a server to a client contains the data requested.

Figure 6-3:  
Modbus data  
transmission  
(acc. to  
Modbus-IDA)



If an error related to the Modbus function requested occurs, the field contains an exception code that the server application can use to determine the next action to be taken.

Figure 6-4:  
Modbus data  
transmission  
(acc. to  
Modbus-IDA)



6.1.2 Data model

The data model distinguishes 4 basic data types:

Table 6-1: Data types for Modbus	Data Type	Object type	Access	Comment
	Discrete Inputs	Bit	Read	This type of data can be provided by an I/O system.
	Coils	Bit	Read-Write	This type of data can be alterable by an application program.
	Input Registers	16-bit, (word)	Read	This type of data can be provided by an I/O system.
	Holding Registers	16-bit, (word)	Read-Write	This type of data can be alterable by an application program.

For each of these basic data types, the protocol allows individual selection of 65536 data items, and the operations of read or write of those items are designed to span multiple consecutive data items up to a data size limit which is dependent on the transaction function code.

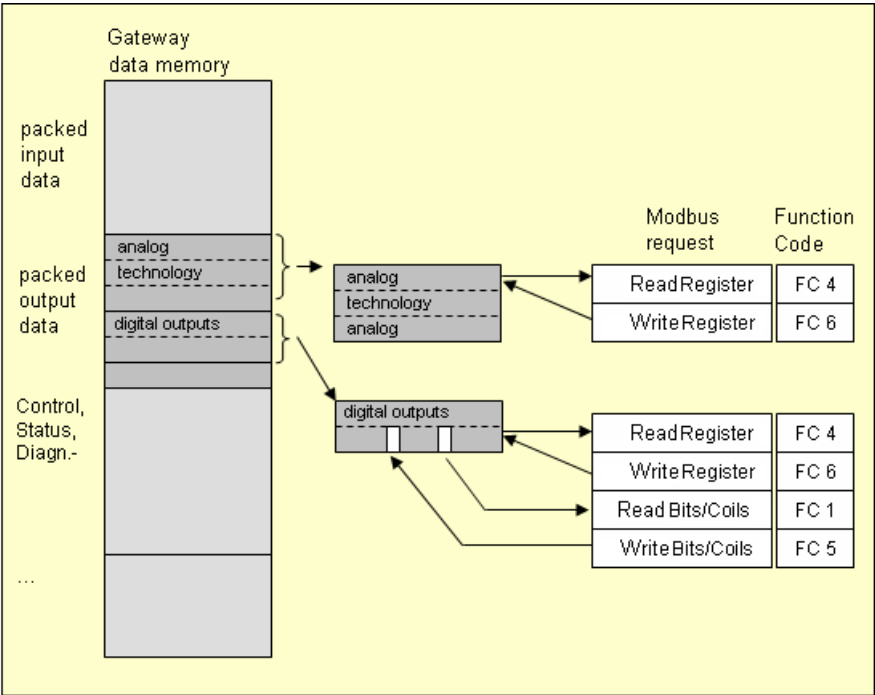
It's obvious that all the data handled via Modbus (bits, registers) must be located in device application memory.

Access to these data is done via defined access-addresses (see „Modbus registers“, [page 6-7](#)).

The example below shows the data structure in a device with digital and analog in- and outputs.

BL20 devices have only one data block, whose data can be accessed via different Modbus functions. The access can be carried out either via registers (16-bit-access) or, for some of them, via single-bit-access.

Figure 6-5:  
Picture of the  
data memory of  
the BL20  
modules



## 6.2 Implemented Modbus functions

The BL20-gateways for Modbus TCP support the following functions for accessing process data, parameters, diagnostics and other services.

Table 6-2:  
Implemented  
functions

Function codes		
No.	Function	Description
1	<b>Read Coils</b>	Serves for reading multiple output bits.
2	<b>Read Discrete Inputs</b>	Serves for reading multiple input bits.
3	<b>Read Holding Registers</b>	Serves for reading multiple output registers.
4	<b>Read Input Registers</b>	Serves for reading multiple input registers.
5	<b>Write Single Coil</b>	Serves for writing a single output bit.
6	<b>Write Single Register</b>	Serves for writing a single output register.
15	<b>Write Multiple Coils</b>	Serves for writing multiple output bits.
16	<b>Write Multiple Registers</b>	Serves for writing multiple output registers.
23	<b>Read/Write Multiple Registers</b>	Reading and writing of multiple registers.

### 6.3 Modbus registers


**Note**

The [Table 6-5](#), [page 6-14](#) shows the register mapping for the different Modbus addressing methods.

*Table 6-3:  
Modbus registers of the  
module*

**A** *ro* = read only  
*rw* = read/write

Address (hex.)	Access A	Description
0x0000 to 0x01FF	ro	packed process data of inputs (process data length of the modules → see <a href="#">Table 6-5: Data width of the I/O-modules</a> )
0x0800 to 0x09FF	rw	packed process data of outputs (process data length of the modules → see <a href="#">Table 6-5: Data width of the I/O-modules</a> )
0x1000 to 0x1006	ro	gateway identifier
0x100C	ro	Gateway status (see <a href="#">Table 6-6: Register 100Ch: Gateway status</a> )
0x1010	ro	process image length in bit for the intelligent output modules
0x1011	ro	process image length in bit for the intelligent input modules
0x1012	ro	process image length in bit for the intelligent output modules
0x1013	ro	process image length in bit for the intelligent input modules
0x1017	ro	Register-mapping-revision (always 1, if not, mapping is incompatible with this description)
0x1018 to 0x101A	ro	group diagnostics of I/O-modules 0 to 32 (1 bit per I/O module)
0x1020	ro	watchdog, actual time [ms]
0x1120	rw	watchdog predefined time [ms] (default: 0), see also <a href="#">Error behavior of outputs (watchdog) (page 6-23)</a> )
0x1121	rw	Watchdog reset register
0x1130	rw	Modbus connection mode register, <a href="#">page 6-17</a>
0x1131	rw	Modbus connection timeout in sec. (Def.: 0 = never), <a href="#">page 6-17</a>
0x113C to 0x113D	rw	Modbus parameter restore, <a href="#">page 6-17</a> (reset of parameters to default values)

Table 6-3:  
Modbus registers of the  
module

Address (hex.)	Access A	Description
0x113E to 0x113F	rw	Modbus parameter save, <a href="#">page 6-18</a> (permanent storing of parameters)
0x1140 (VN 03-00 and higher)	rw	Disable Protocol, <a href="#">page 6-18</a>
0x1141 (VN 03-00 and higher)	ro	Active Protocol, <a href="#">page 6-18</a>
0x2000 to 0x207F	rw	service-object, request-area, <a href="#">page 6-19</a>
0x2080 to 0x20FF	ro	service-object, response-area, <a href="#">page 6-19</a>
0x2400	ro	System voltage $U_{\text{SYS}}$ [mV]
0x2401	ro	Load voltage $U_L$ [mV]
0x2405	ro	load current $I_L$ [A]
0x27FE	ro	no. of entries in actual module list
0x27FF	rw	no. of entries in reference module list
0x2800 to 0x283F	rw	Reference module list (max. 32 modules per station $\times$ 2 registers for module-ID)
0x2A00 to 0x2A3F	ro	Actual module list (max. 32 modules per station $\times$ 2 registers for module-ID)
0x8000 to 0x8400	ro	process data inputs (max. 32 modules per station $\times$ 32 registers for module-ID)
0x9000 to 0x9400	rw	process data outputs (max. 32 modules per station $\times$ 32 registers for module-ID)
0xA000 to 0xA400	ro	Diagnosis (max. 32 modules per station $\times$ 32 registers for module-ID)
0xB000 to 0xB400	rw	Parameters (max. 32 modules per station $\times$ 32 registers for module-ID)



The following table shows the register mapping for the different Modbus addressing methods

Table 6-4:  
Mapping of  
BL20-E-GW-EN  
Modbus regis-  
ters (holding  
registers)

Description	Hex	Decimal	5-digit	Modicon
packed input data	0x0000 to 0x01FF	0 to 511	40001 to 40512	400001 to 400512
packed output data	0x0800 to 0x09FF	2048 to 2549	42049 to 42560	402049 to 402560
gateway identifier	0x1000 to 0x1006	4096 to 4102	44097 to 44103	404097 to 404103
Gateway status	0x100C	4108	44109	404109
process image length in bit for the intelligent output modules	0x1010	4112	44113	404113
process image length in bit for the intelligent input modules	0x1011	4113	44114	404114
process image length in bit for the digital output modules	0x1012	4114	44115	404115
process image length in bit for the digital input modules	0x1013	4115	44116	404116
Register-mapping-revision	0x1017	4119	44120	404120
group diagnostics of I/O-modules 1 to 32 (1 bit per I/O module)	0x1018 to 0x1019	4120 to 4121	44121 to 44122	404121 to 404122
watchdog, actual time	0x1020	4128	44129	404129
watchdog, predefined time	0x1120	4384	44385	404385
Watchdog reset register	0x1121	4385	44386	404386
Modbus connection mode register	0x1130	4400	44401	404401
Modbus connection timeout in sec.	0x1131	4401	44402	404402
Modbus parameter restore	0x113C to 0x113D	4412 to 4413	44413 to 44414	404413 to 404414
Modbus parameter save	0x113E to 0x113F	4414 to 4415	44415 to 44416	404415 to 404416
service-object, request-area,	0x2000 to 0x207F	8192 to 8319	48193 to 48320	408193 to 408320

Table 6-4:  
Mapping of  
BL20-E-GW-EN  
Modbus regis-  
ters (holding  
registers)

Description	Hex	Decimal	5-digit	Modicon
Disable protocol (VN 03-00 and higher)	0x1140	4416	44417	404417
Active protocol (VN 03-00 and higher)	0x1141	4417	44418	404418
service-object, response-area, to 0x20FF	0x2080 to 0x20FF	8320 to 8447	48321 to 48448	408321 to 408448
System voltage $U_{\text{SYS}}$ [mV]	0x2400	9216	49217	409217
Load voltage $U_L$ [mV]	0x2401	9217	49218	409218
load current $I_L$ [A]	0x2405	9221	49222	409222
no. of entries in actual module list	0x27FE	10238	-	410239
no. of entries in reference module list	0x27FF	10239	-	410240
Reference module list (max. 32 modules per station $\times$ 2 registers for module-ID)	0x2800 to 0x283F	10240 to 10303	-	410241 to 410304
Actual module list (max. 32 modules per station $\times$ 2 registers for module-ID)	0x2A00 to 0x2A3F	10752 to 10815	-	410753 to 410816
<b>Slot-related address assignment</b>				
Process data inputs (max. 32 modules per station $\times$ 32 registers for module-ID)	0x8000 to 0x8400			
slot 1	0x8000	32768	-	432769
slot 2	0x8020	32800	-	432801
slot 3	0x8040	32832	-	432833
...	...	...	...	...
slot 32	0x83E0	33760		433761
Process data outputs (max. 32 modules per station $\times$ 32 registers for module-ID)	0x9000 to 0x9400			
slot 1	0x9000	36864	-	436865
slot 2	0x9020	36896	-	436897
slot 3	0x9040	36928	-	436929
...	...	...	...	...
slot 32	0x93E0	37856	-	437857

Table 6-4:  
Mapping of  
BL20-E-GW-EN  
Modbus regis-  
ters (holding  
registers)

Description	Hex	Decimal	5-digit	Modicon
Diagnostics (max. 32 modules per station × 32 registers for module-ID)	0xA000 to 0xA400			
slot 1	0xA000	40960	-	440961
slot 2	0xA020	40991	-	440992
slot 3	0xA040	41023	-	441024
...	...	...	...	...
slot 32	0xA3E0	41983	-	441984
Parameters (max. 32 modules per station × 32 registers for module-ID)	0xB000 to 0xB400			
slot 1	0xB000	45056	-	445057
slot 2	0xB020	45088	-	445089
slot 3	0xB040	45120	-	445121
...	...	...	...	...
slot 32	0xB3E0	46048	-	446049

6.3.1 Structure of the packed in-/ output process data

In order to assure a largely efficient access to the process data of a station, the module data are consistently packed and mapped to a coherent register area.

The I/O-modules are divided into digital and intelligent modules (analog modules, serial interfaces).



Note

For the data mapping, the BL20-1SWIRE-modules are not considered as intelligent modules. Their process data is mapped into the register area for the digital in- and output modules

Both module types are mapped in separate register ranges.

The data mapping always starts with the mapping of the intelligent modules. Each module occupies as many Modbus registers as necessary, depending on it's data width. At least one register is occupied. A RS232-module, for example, occupies 4 consecutive registers (8 bytes) in the input and in the output area.

The data byte arrangement is done according to the physical order in the station, from the left to the right.

The data of the intelligent modules are followed by the data of the digital modules, also structured according to their physical appearance in the station. The Modbus registers for the digital data are filled up to 16 bit. This means on the one hand that one Modbus register can contain data of different digital modules and on the other hand that the data of one digital module can be distributed over multiple registers. Bit 0 of a digital module is thus not necessarily located on a word limit.



Note

An example in [chapter 7, page 7-16ff.](#) describes the data mapping.

Additionally, the software I/O-ASSISTANT offers the possibility to create a mapping table for every station.

Packed input process data

■ input register area: **0x0000** to **0x01FF**

<b>0x0000</b>			<b>0x01FF</b>
intelligent modules, input data	digital Input modules	status/ diagnosis	free



Note

Independent of the I/O-configuration, an access to all 512 registers is always possible. Registers that are not used send "0".

### Status/ diagnosis

The area "status/diagnosis" comprises a maximum of 9 registers.

The first register contains a common gateway-/station-status.

The following registers (max. 8) contain a group diagnostic bit for each I/O-module which shows whether a diagnostic message is pending for the relevant module or not.

Status/ diagnosis	
n + 0x0000	n + 0x0008
Gateway status (reg. 100Ch)	group diagnosis I/O-modules 0...127 (register 0x1018 to 0x101F)

### Packed output process data

■ output register area: **0x0800** to **0x09FF**

0x0800	0x09FF
intelligent modules, output data	free



#### Note

Independent of the I/O-configuration, an access to all 512 registers is always possible. Registers that are not used send "0" answering a read access, write accesses are ignored.

## Data width of the I/O-modules in the modbus-register area

The following table shows the data width of the BL20-I/O-modules within the modbus register area and the type of data alignment.

Table 6-5:  
Data width of  
the I/O-modules

Module	Process input	Process output	Alignment
<b>– digital inputs</b>			
BL20-2DI-x	2 Bit	-	bit by bit
BL20-4DI-x	4 Bit	-	bit by bit
BL20-E-8DI-x	8 Bit	-	bit by bit
BL20-16DI-x	16 Bit	-	bit by bit
BL20-E-16DI-x	16 Bit	-	bit by bit
BL20-32DI-x	32 Bit	-	bit by bit
<b>– digital outputs</b>			
BL20-2DO-x	-	2 Bit	bit by bit
BL20-4DO-x	-	4 Bit	bit by bit
BL20-E-8DO-x	-	8 Bit	bit by bit
BL20-16DO-x	-	16 Bit	bit by bit
BL20-E-16DO-x	-	16 Bit	bit by bit
BL20-32DO-x	-	32 Bit	bit by bit
<b>– Analog input modules</b>			
BL20-1AI-x	1 word		word by word
BL20-2AI-x	2 word		word by word
BL20-2AIH-I	12 word		word by word
BL20-4AI-x	4 word		word by word
BL20-E-4AI-TC	4 word		word by word
BL20-E-8AI-U/I-4AI-PT/NI	8 word		word by word
<b>– Analog outputs</b>			
BL20-1AO-x		1 word	word by word
BL20-2AO-x		2 word	word by word
BL20-2AOH-I	8 word	2 word	word by word
BL20-E-4AO-U/I		4 word	word by word

Table 6-5:  
Data width of  
the I/O-modules

	Module	Process input	Process output	Alignment
<b>A</b> The process data of the SWIRE-modules is mapped into the register area for the digital in- and output modules.	<b>– Technology modules</b>			
	BL20-1RSxxx	4 word	4 word	word by word
	BL20-1SSI	4 word	4 word	word by word
	BL20-E-2CNT-2PWM	12 word	12 word	word by word
	BL20-E-SWIRE <b>A</b>	4 word	4 word	word by word
	BL20-2RFID-S	12 word	12 word	word by word
	<b>– Power distribution modules</b>			
	BL20-BR-x	-		
	BL20-PF-x	-		

## 6.3.2 Register 0x100C: Gateway status

This register contains a general gateway/ station status.

Table 6-6:  
Register 100Ch:  
Gateway status

Bit	Name	Description
<b>Gateway</b>		
15	reserved	-
14	Force Mode Active Error	The Force Mode is activated, which means, the actual output values may no match the ones defined and sent by the field bus.
13	reserved	-
12	Modbus Wdog Error	A timeout occurred in the modbus-communication.
<b>Module bus</b>		
11	I/O Cfg Modified Error	The I/O-configuration has be changed and is no longer compatible.
10	I/O Communication Lost Error	No Communication on the module bus.
<b>Voltage errors</b>		
9	$U_{sys}$ too low	System supply voltage too low (< 18 V DC).
8	$U_{sys}$ too high	System supply voltage too high (> 30 V DC).
7	$U_L$ too low	Load voltage too low (< 18 V DC).
6	reserved	-
5	reserved	-
4	reserved	-
<b>Warnings</b>		
3	I/O Cfg Modified Warning	The station configuration has changed.
0	I/O Diags Active Warning	At least one I/O-module sends active diagnosis.



### 6.3.3 Register 0x1130h: Modbus-Connection-Mode

This register defines the behavior of the Modbus connections:

Table 6-7: Register 0x1130h: Modbus- Connection- Mode  <b>A</b> default setting	<b>Bit</b>	<b>Name</b>
		– Description
	15 to 2	reserved
	1	<b>MB_ImmediateWritePermission</b> – <b>0</b> : With the first write access, a write authorization for the respective Modbus-connection is requested. If this request fails, an exception response with exception-code 01h is generated. If the request is accepted, the write access is executed and the write authorization remains active until the connection is closed. <b>A</b> – <b>1</b> : The write authorization for the respective Modbus-connection is already opened during the establishment of the connection. The first Modbus-connection thus receives the write authorization, all following connections don't (only if bit 0 = 1).
	0	<b>MB_OnlyOneWritePermission</b> – <b>0</b> : all Modbus-connections receive the write authorization <b>A</b> – <b>1</b> : only one Modbus-connection can receive the write permission. A write permission is opened until a Disconnect. After the Disconnect the next connection which requests a write access receives the write authorization.

### 6.3.4 Register 0x1131: Modbus-Connection-Timeout

This register defines after which time of inactivity a Modbus-connection is closed through a Disconnect.

### 6.3.5 Register 0x113C and 0x113D: Restore Modbus-connection parameters

Register 0x113C and 0x113D are used to reset the parameter registers 0x1120 and 0x1130 to 0x113B to default.

For this purpose, write 0x6C6F to register 0x113E. To activate the reset of the registers, write 0x6164 ("load") within 30 seconds in register 0x113D.

Both registers can also be written with one single request using the function codes FC16 and FC23.

The service resets the parameters without saving them. This can be achieved by using a following "save" service.

### 6.3.6 Register 0x113E and 0x113F: „Save Modbus-Connection-Parameters“

Registers 0x113E and 0x113F are used for permanent storing the parameters in registers 0x1120 and 0x1130 to 0x113B.

For this purpose, write 0x7361 to register 0x113E. To activate the saving of the registers, write 0x7665 ("save") within 30 seconds in register 0x113F.

Both registers can also be written with one single request using the function codes FC16 and FC23.

### 6.3.7 Register 0x1140: Disable protocol



#### Note

This register is only valid for BL20-E-GW-EN with multiprotocol-functionality, meaning, for gateways with **VN 03-00** and higher.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	-	-	-	-	PROFINET deactivate	reserved	EtherNet/IP deactivate
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Web-Server deactivate	-	-	-	-	-	-	-

### 6.3.8 Register 0x1141: Active protocol



#### Note

This register is only valid for BL20-E-GW-EN with multiprotocol-functionality, meaning, for gateways with **VN 03-00** and higher.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	-	-	-	-	PROFINET active	Modbus TCP active	EtherNet/IP active
Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
Web-Server active	-	-	-	-	-	-	-

### 6.3.9 Register 0x2000 bis 0x207F: The Service-Object

The service-object is used to execute one-time or acyclic services. It is an acknowledge service which may serve, for example, to parameterize an I/O-module.

0x2000	0x2080	0x20FF
service request area	service response area	

The service request area allows write access, the service response area only read access.

- service request area

0x2000	0x2001	0x2002	0x2003	0x2004	0x2005	0x207F
Service-number	reserved	Service-Code	Index/ addr	Data-Reg-Count	optional data (0...122 registers)	

The register **service no.** in the request area can contain a user defined value which is deleted after the execution of the service.

The register **service code** specifies which service is requested.

The register **index/addr** is optional and the meaning depends on the particular service.

The register **data-reg-count** contains, depending on the service, the number (0 to 122) of the transferred or of the requested data registers.

Depending on the service, the **optional data area** can contain additional parameters and/or other data to be written.

- Service-response-area

0x2080	0x2081	0x2082	0x2083	0x2084	0x2085	0x20FF
Service-number	result	Service-Code	Index/ Addr	Data-Reg-Count	optional data (0...122 registers)	

After the execution of a request, the registers **service-no.**, **service code** and **index/addr** in the response area contain a copy of the values in the request area.



#### Note

The service no. is thus used for a simple handshake on the application level. The application increases the service no. with every request. The service is blocked, until the service number in the request area matches the service number in the response area.

The register **result** shows whether the execution was successful or not.

The register **data-reg-count** contains the number of data registers (0 to 122).

The **optional data area** can contain, depending on the service, the requested data.

Supported service numbers:

Table 6-8:  
Supported  
service numbers

Service-Code	Meaning
0x0000	no function
0x0003	indirect reading of registers
0x0010	indirect writing of registers

A service request may have the following results:

Table 6-9:  
results of the  
service request

Service-Code	Meaning
0x0000	error free execution of service
0xFFFE	service parameters incorrect/ inconsistent
0xFFFF	service code unknown



### Note

The services "indirect reading of registers" and "indirect writing of registers" offer an additional possibility to access any Modbus register.

Current Modbus-masters support only a limited number of register-areas that can be read or written during the communication with a Modbus-server. These areas can not be changed during operation.

In this case, the services mentioned above enables non-cyclic access to registers.

### Indirect reading of registers

1...122 (Count) Modbus-registers are read starting with address x (Addr).

#### ■ service-request

0x2000	0x2001	0x2002	0x2003	0x2004	0x2005	0x207F
Service-number	0x0000	0x0003	Addr	Count	no meaning	

#### ■ service response

0x2080	0x2081	0x2082	0x2083	0x2084	0x2085	0x20FF
Service-number	result	0x0003	Addr	Count	register contents	

**Indirect writing of registers**

1 to 122 ( Count) Modbus-registers are read, starting with address Addr.)

■ service-request

<b>0x2000</b>	<b>0x2001</b>	<b>0x2002</b>	<b>0x2003</b>	<b>0x2004</b>	<b>0x2005</b>	<b>0x207F</b>
Service-number	0x0000	0x0010	Addr	Count	register contents	

■ service response

<b>0x2080</b>	<b>0x2081</b>	<b>0x2082</b>	<b>0x2083</b>	<b>0x2084</b>	<b>0x2085</b>	<b>0x20FF</b>
Service-number	result	0x0010	Addr	Count	no meaning	

### 6.4 Bit areas: mapping of input-discrete- and coil-areas

The digital in- and outputs can be read and written (for outputs) as registers in the data area of the packed in- and output process data.



#### Note

In the packed process data, the digital I/O data are stored following the variable in- and output data area of the intelligent modules, which means they are stored with a variable offset, depending on the station's I/O-configuration.

---

In order to set for example a single output (single coil), the following functions are available for reading and writing single bits:

- FC1 („Read Coils“)
- FC2 („Read Discrete Inputs“)
- FC5 („Write Single Coil“)
- FC15 („Write Multiple Coils“)

#### Data mapping in the input-discrete- and coil-areas:

- Mapping Mapping: input-discrete-area  
All digital inputs are stored in this area (offset "0").
- Mapping Mapping: Coil-area  
All digital outputs are stored in this area (offset "0").

### 6.5 Error behavior of outputs (watchdog)

In case of a failure of the Modbus communication, the outputs' behavior is as follows, depending on the defined time for the Watchdog ([register 0x1120 \(page 6-7\)](#)):

- watchdog = 0 ms (default)  
→ outputs hold the momentary value
- watchdog > 0 ms  
→ outputs switch to **0** after the watchdog time has expired



#### Note

Please observe that changes in the watchdog time have to be saved per save-command (see [Register 0x113E and 0x113F: „Save Modbus-Connection-Parameters“ \(page 6-18\)](#)).

---



#### Note

Setting the outputs to predefined substitute values is not possible in Modbus TCP. Eventually parameterized substitute values will not be used.

---