Protocol

Modbus/TCP

Meusburger Deutschland GmbH
Voltastrasse 2
68519 Viernheim
Germany
Phone +49 6204 6069-0
www.meusburger.com
office-de@meusburger.com

CE

Rev. 1.00.02
01/2020
Translation of original
operating instructions

1 Table of Contents

2 General Specification

Modbus/TCP protocol is available for the following PSG controllers

Multi loop controller
flexotemp ® MCU 128
flexotemp ® PCU 128
flexotemp ® PCU 048
flexotemp ® PCU 024

Hot Runner Controller
profiTEMP

The Modbus protocol is designed for a client/server application. For this reason all configuration parameters and also the status of the zones are transparent for the user.

Typographic Conventions

Symbols and conventions are used in this manual for faster orientation for you.

	Caution	With this symbol, references and information are displayed which are decisive for the operation of the device. In case of non-compliance with or inaccurate compliance there can result damage to the device or injuries to persons.
	Note	The symbol refers to additional information and declarations, which serve for improved understanding.
	Example	With the symbol, a function is explained by means of an example.
	Reference	With this symbol, information in another document is referred to.
	FAQ	Here FAQ (Frequently Asked Questions) are answered.
	`Equations`	Calculation specifications and examples are represented in this way.

2.1 Principle of Communication

The communication follows the client/server model. The client communicates with one or several servers. The server is only active, when the client calls it.

Modbus/TCP uses Standard Ethernet with TCP/IP. As transfer protocol TCP is used. Before the appropriate operations/commands will be transferred, a TCP connection with the server (MCU/PCU) must be established. For the connection the IP address of the controller and the destination port „502“ is used. On the controller at least one of the available ports must be set to „502“, so that the connection by the Modbus/TCP protocol can be established.

The establishment/termination of the connection is done by the client. When after establishment of connection no communication occurs in between 30 sec between client and server, the server automatically terminates the connection.

2.2 Addressing

See 3.2 Addressing

3 Protocol Frame

3.1 General Conditions

The client/server operations are transferred in the so called Modbus PDUs. These are independent from the subordinate communication system. For Modbus/TCP before the Modbus PDU a so called MBAP header (Modbus Application Protocol Header) is placed:

Data Part of the TCP Segments					
MBAP-Header				**Modbus-PDU**	
2 Bytes	2 Bytes	2 Bytes	1 Byte	1 Byte	0…253 Bytes
Transaction Identifier (TId)	Protocol Identifier (Pid)	Length	Unit Identifier (Zone)	Function Code (FC)	Data

The length of the user data of a Modbus PDU is limited to 253 bytes by compatibility reasons. Per TCP segment only one Modbus PDU is allowed. The MBAP header consists of the following fields:

Element	Description
Transaction Identifier (Tid)	Used for the allocation between request and response. The client allocates the field, the server copies the value into the response. By this identifier the client could assign the answer to its request.
Protocol Identifier (Pid)	Includes for Modbus/TCP always the value 0.
Length	Length of the following data in bytes (from Unit Identifier)
Unit Identifier	Includes the zone to address (1-n). The value 1 equates the 1. Zone on the controller.
Function Code (FU)	See Chapter4.1

The values are coded in the byte order of Big Endian.

3.2 Addressing

In Modbus/TCP protocol each zone has its own address. The first zone on the controller has always the address 1. The zone n has always the address n. The controller is specified by its IP address at establishing connection. Because only one byte is available for the address, maximal 256 zones can be addressed per controller. Because the address 00H is reserved for broadcast messages, the number of reachable zones by Modbus is reduced to 255.

3.3 Error Messages (Exception response)

The following error codes are supported

Error code	Name	Meaning
01	Illegal function	Function number out of allowed range
02	Illegal data address	Parameter not supported
03	Illegal data value	Data value incorrect / function not executed

4 Detailed Specification of Protocol Functions

More detailed information on referred **Bit**- / **Word tables** see controller specific parameter and object lists.

4.1 List of commands

The following commands are supported

Function code	Meaning	Action
01_{DEC}	**Read n Bits**	Reads n bits
03_{DEC}	**Read n Words**	Reads n words
05_{DEC}	**Write 1 Bit**	Sets or deletes 1 bit
06_{DEC}	**Write 1 Word**	Writes 1 word
08_{DEC}	**Loop back test**	Checks the communication
15_{DEC}	**Write n Bits**	Deletes or sets n bits
16_{DEC}	**Write n Words**	Writes n words
17_{DEC}	**Report Slave-ID**	Shows the version number

4.1.1 Read n Bits (Function Code 01_{DEC})

This command allows to read a single or several bits.

For the sending command the following structure exists

Client	**Tid**		**Pid**		**Length**	
Byte number	1	2	3	4	5	6
HEX	HI	LO	HI	LO	HI	LO
Client	**Zone**	**Function**	**Address of 1. Bit**		**Number of Bits**	
Byte number	7	8	9	10	11	12
HEX		01	HI	LO	HI	LO

Thereby the parameter „Address of 1. Bit“ identifies the table index of a listed bit in the **Bit Table**. The parameter „Number of Bits“ identifies the number of bits, which should be read from the stated table index consecutively.

The controller returns the following response

Server	**Tid**		**Pid**		**Length**	
Byte number	1	2	3	4	5	6
HEX	HI	LO	HI	LO	HI	LO
Server	**Zone**	**Function**	**Byte Count**	**Bit 1…8**	**Bit …**	**Last Bit**
Byte number	7	8	9	10	?	?
HEX		01				

The parameter „Byte Count“ defines the number of transmitted data bytes.

The single bits are masked into single data bytes as follows

Bit of Data Byte No. …	Address of Bit in the Table
Bit 0 of data byte 1	x
Bit 1 of data byte 1	X + 1
Bit 2 of data byte 1	X + 2
Bit 3 of data byte 1	X + 3
Bit 4 of data byte 1	X + 4
Bit 5 of data byte 1	X + 5
Bit 6 of data byte 1	X + 6
Bit 7 of data byte 1	X + 7
Bit 0 of data byte 2	X + 8
Bit 1 of data byte 2	X + 9
(…)	(…)

where x = address of 1. Bit in the table

Per „Data Byte“ 8 bits are transferred in this way. If less than 8 bits are transferred in one „Data Byte“, the remaining bits are set to 0.

From table index 2 of channel 6 the status of 3 bits should be read.

The corresponding sending command looks as follows

Client	**Tid**		**Pid**		**Length**	
Byte number	1	2	3	4	5	6
HEX	00	9E	00	00	00	06
Client	**Zone**	**Function**	**Address of 1. Bit**		**Number of Bits**	
Byte number	7	8	9	10	11	12
HEX	06	01	00	02	00	03

Response of controller

Server	**Tid**		**Pid**		**Length**	
Byte number	1	2	3	4	5	6
HEX	00	9E	00	00	00	04
Server	**Zone**	**Function**	**Byte Count**	**Bit 1…8**		
Byte number	7	8	9	10		
HEX	06	01	01	05		

Byte-Count = 01H (exactly 1 data byte is transmitted)
Data field bit 1-8 = 05H

Bit of Data Field	Status of Bit	Address (Index) of Bit in the Table
Bit 0 of data byte 1	1	Table index 2
Bit 1 of data byte 1	0	Table index 2 + 1
Bit 2 of data byte 1	1	Table index 2 + 2
Bit 3 of data byte 1	0	Table index 2 + 3
Bit 4 of data byte 1	0	Table index 2 + 4
Bit 5 of data byte 1	0	Table index 2 + 5
Bit 6 of data byte 1	0	Table index 2 + 6
Bit 7 of data byte 1	0	Table index 2 + 7

4.1.2 Read n Words (Function Code 03 $_{DEC}$)

This command allows to read a single or several words.
For the sending command the following structure exists

Client	**Tid**		**Pid**		**Length**	
Byte number	1	2	3	4	5	6
HEX	HI	LO	HI	LO	HI	LO
Client	**Zone**	**Function**	**Address of 1. Word**		**Number of Words**	
Byte number	7	8	9	10	11	12
HEX		03	HI	LO	HI	LO

The parameter „Address of 1. Word“ identifies the table index of one in the **Word Table** listed word and the parameter „Number of Words“ the number of words, which should be read from the stated table index consecutively.

The controller returns the following response

Server	**Tid**		**Pid**			**Length**			
Byte number	1	2	3	4		5		6	
HEX	HI	LO	HI	LO		HI		LO	
Server	**Zone**	**Function**	**Byte Count**	**Word 1**		**Word …**		**Last Word**	
Byte number	7	8	9	10	11	?	?	?	?
HEX		03		HI	LO	HI	LO	HI	LO

The parameter „Byte Count“ defines the number of transmitted data bytes, that is exactly the double value of the requested word. In the response of the controller first the High byte and then the Low byte of the word is transferred.

From table index 2 of channel 3 (SOLL = 100°C) one word should be read.

The corresponding sending command looks as follows

Client	**Tid**		**Pid**		**Length**	
Byte number	1	2	3	4	5	6
HEX	02	83	00	00	00	06
Client	**Zone**	**Function**	**Address of 1. Word**		**Number of Words**	
Byte number	7	8	9	10	11	12
HEX	03	03	00	02	00	01

Response of controller

Server	**Tid**		**Pid**		**Length**	
Byte number	1	2	3	4	5	6
HEX	02	83	00	00	00	05
Server	**Zone**	**Function**	**Byte Count**	**Word 1**		
Byte number	7	8	9	10	11	
HEX	03	03	02	03	E8	

Byte-Count = 02H (exactly 1 data word is transmitted)
Word 1 = 03E8H = 1000_{DEC} (1000, because the setpoint value is specified in 0.1°C)

4.1.3 Write 1 Bit (Function Code 05 $_{DEC}$)

This command allows to set or delete a single bit.

For the sending command the following structure exists

Client	**Tid**		**Pid**		**Length**	
Byte number	1	2	3	4	5	6
HEX	HI	LO	HI	LO	HI	LO
Client	**Zone**	**Function**	**Address of Bit**		**Set / Reset Data Index**	
Byte number	7	8	9	10	11	12
HEX		05	HI	LO	HI	LO

The parameter „Address of Bit“ identifies the table index of a listed bit in the **Bit Table** and the parameter „Set / Reset Data Index“ indicates, whether the mentioned bit should be set or deleted. Is „Set / Reset Data Index“ set to FF00H, the bit is set to logical 1, is it set to 0000H, the bit is set to logical 0.

The controller repeats as response the received command after its execution.

Bit FBA of channel 32 should be set to „ON“ (logical 1). The table index of bit FBA is 6.

The command and the response are

Client	**Tid**		**Pid**		**Length**	
Byte number	1	2	3	4	5	6
HEX	03	50	00	00	00	06
Client	**Zone**	**Function**	**Address of Bit**		**Set / Reset Data Index**	
Byte number	7	8	9	10	11	12
HEX	20	05	00	06	FF	00

4.1.4 Write 1 Word (Function Code 06 $_{DEC}$)

This command allows to write a value in one word parameter.

For the sending command the following structure exists

Client	**Tid**		**Pid**		**Length**	
Byte number	1	2	3	4	5	6
HEX	HI	LO	HI	LO	HI	LO
Client	**Zone**	**Function**	**Address of Word**		**Data value**	
Byte number	7	8	9	10	11	12
HEX		06	HI	LO	HI	LO

The parameter „Address of Word“ identifies the table index of a listed control parameter in the **Word Table**. The „Data Value“ contains the value, where the control parameter should be set to. Consider the value range of the respective control parameter, when setting the „Data Value“.

The controller repeats as response the received command after its execution.

Setpoint value of channel 1 should be set to 10°C. The setpoint value has the table index 1 and is specified in 0.1°C.

The command and the response are

Client	**Tid**		**Pid**		**Length**	
Byte number	1	2	3	4	5	6
HEX	03	76	00	00	00	06
Client	**Zone**	**Function**	**Address of Word**		**Data value**	
Byte number	7	8	9	10	11	12
HEX	01	06	00	01	00	64

4.1.5 Loop Back Test (Function Code 08 $_{DEC}$)

This command allows a simple test of the communication.

For the sending command the following structure exists

Client	**Tid**		**Pid**		**Length**	
Byte number	1	2	3	4	5	6
HEX	HI	LO	HI	LO	HI	LO
Client	**Zone**	**Function**	**Diagnostics code**		**Data**	
Byte number	7	8	9	10	11	12
HEX		08	00	00	HI	LO

By the parameter „Diagnostics Code“ is determined, which data the controller should return. The controller only supports „Diagnostics Code“ 0000H. This means, that the data is returned 1:1 in the data field. Any word could be inserted as „Data“.

The controller should repeat the whole request without any further action.

Data value 1234H of channel 5 should be returned.

The command and the response are

Client	**Tid**		**Pid**		**Length**	
Byte number	1	2	3	4	5	6
HEX	03	89	00	00	00	06
Client	**Address**	**Function**	**Diagnostics code**		**Data**	
Byte number	7	8	9	10	11	12
HEX	05	08	00	00	12	34

4.1.6 Write n Bits (Function Code 15 $_{DEC}$)

This command allows to set or delete several bits. The bits must be in the **Bit Table** in consecutive order.

For the sending command the following structure exists

Client	**Tid**		**Pid**				**Length**		
Byte number	1	2	3		4		5		6
HEX	HI	LO	HI		LO		HI		LO
Client	**Zone**	**Function**	**Address of 1. Bit**		**Quantity**		**Byte Count**	**Data Byte1**	**Data Byte …**
Byte number	7	8	9	10	11	12	13	14	?
HEX		0F	HI	LO	HI	LO			

The parameter „Address of 1. Bit“ matches the index of the Bit Table, from where bits should be set / deleted.
The parameter „Quantity“ shows the number of bits, which should be set / deleted consecutively from the above mentioned index in the Bit Table.
„Byte Count“ defines the number of transmitted “Data Bytes”.
In the „Data Bytes“ the information is transferred, whether a bit should be set or deleted. The bit is deleted, when a logical 0 is set in the corresponding data byte. By a 1 it is set.

The allocation of data byte information and the bits in the Bit Table is as follows

Bit of Data Byte No. …	**Address of Bit in the Table**
Bit 0 of data byte 1	x
Bit 1 of data byte 1	X + 1
Bit 2 of data byte 1	X + 2
Bit 3 of data byte 1	X + 3
(…)	(…)
Bit 14 of data byte 1	X + 14
Bit 15 of data byte 1	X + 15
Bit 0 of data byte 2	X + 16
Bit 1 of data byte 2	X + 17
(…)	(…)

where x = address of 1. Bit in the table

Per „Data Byte“ 8 bits are transferred in this way. If less than 8 bits are transferred in one „Data Byte“, the remaining bits are set to 0.

The controller returns the following response

Server	**Tid**		**Pid**				**Length**		
Byte number	1	2	3		4		5		6
HEX	HI	LO	HI		LO		HI		LO
Server	**Zone**	**Function**	**Address of 1. Bit**		**Quantity**				
Byte number	7	8	9	10	11	12			
HEX		0F	HI	LO	HI	LO			

The controller repeats the request except for the „Data Bytes“.

8 bits from table index 2 of channel 2 should be deleted independent from their status before and the bit 9 should be set.

The corresponding sending command looks as follows

Client	**Tid**		**Pid**				**Length**		
Byte number	1	2	3		4		5		6
HEX	03	E8	00		00		00		09
Client	**Zone**	**Function**	**Address of 1. Bit**		**Quantity**		**Byte Count**	**Data Byte1**	**Data Byte 2**
Byte number	7	8	9	10	11	12	13	14	15
HEX	02	0F	00	02	00	09	02	00	01

Response of controller

Server	**Tid**		**Pid**				**Length**		
Byte number	1	2	3		4		5		6
HEX	03	E8	00		00		00		06
Server	**Zone**	**Function**	**Address of 1. Bit**		**Quantity**				
Byte number	7	8	9	10	11	12			
HEX	02	0F	00	02	00	09			

4.1.7 Write n Words (Function Code 16 $_{DEC}$)

This command allows to write several words with different data values.

For the sending command the following structure exists

Client	**Tid**		**Pid**				**Length**				
Byte number	1	2	3		4		5		6		
HEX	HI	LO	HI		LO		HI		LO		
Client	**Zone**	**Function**	**Address of 1. Word**		**Quantity**		**Byte Count**	**Data Word1**		**Data Word …**	
Byte number	7	8	9	10	11	12	13	14	15	?	?
HEX		10	HI	LO	HI	LO		HI	LO	HI	LO

The parameter „Address of 1. Word“ identifies the index of a listed control parameter in the **Word Table**.

The parameter „Quantity“ shows the number of words, which should be written from the above mentioned index into the Word Table.

In the parameter „Byte Count“ the number of transmitted “Data Bytes” is defined.

The information, which value is written to a control parameter, is transferred in the „Data Words“; Consider the value range of the respective control parameter.

The controller returns the following response

Server	**Tid**		**Pid**		**Length**	
Byte number	1	2	3	4	5	6
HEX	HI	LO	HI	LO	HI	LO
Server	**Zone**	**Function**	**Address of 1. Word**		**Quantity**	
Byte number	7	8	9	10	?	?
HEX		10	HI	LO	HI	LO

The controller repeats the request except for the „Data Words“.

Exactly 2 words from table index 11 of channel 4 (SPLO = 10.0°C and SPHI = 200.0°C) should be written.

The corresponding sending command looks as follows

Client	**Tid**		**Pid**				**Length**				
Byte number	1	2	3		4		5		6		
HEX	05	21	00		00		00		0B		
Client	**Zone**	**Function**	**Address of 1. Word**		**Quantity**		**Byte Count**	**Data Word1**		**Data Word …**	
Byte number	7	8	9	10	11	12	13	14	15	16	17
HEX	04	10	00	0B	00	02	04	00	64	07	D0

Response of controller

Server	**Tid**		**Pid**		**Length**	
Byte number	1	2	3	4	5	6
HEX	05	21	00	00	00	06
Server	**Zone**	**Function**	**Address of 1. Word**		**Quantity**	
Byte number	7	8	9	10	11	12
HEX	04	10	00	0B	00	02

4.1.8 Report Slave ID (Function Code 17 $_{DEC}$)

With this command the version number is read. Additionally the system status can be requested.

For the sending command the following structure exists

Client	**Tid**		**Pid**		**Length**	
Byte number	1	2	3	4	5	6
HEX	HI	LO	HI	LO	HI	LO
Client	**Zone**	**Function**				
Byte number	7	8				
HEX		11				

The controller returns the following response

Server	**Tid**		**Pid**		**Length**	
Byte number	1	2	3	4	5	6
HEX	HI	LO	HI	LO	HI	LO

Server	**Zone**	**Function**	**Byte Count**	**Identification**		**1. Word**		**2. Word**		**3. Word**	
Byte number	7	8	9	10	11	12	13	14	15	16	17
HEX		11	06	10	FF	HI	LO	HI	LO	HI	LO

The parameter „Byte Count“ defines the number of transmitted bytes. Always 6 data bytes are transferred with this command.

In the High byte of the 1. Word for a PCU ‘P’ and for a MCU ‘M’ is written in ASCII value. In the Low byte of the 1. Word the maximal zone number, available on the controller, is listed. For a PCU048 48 is itemized.
In the „second” and the “third word“ the software version number of the controller is transferred.

The version number should be read.

The corresponding sending command looks as follows

Client	**Tid**		**Pid**		**Length**	
Byte number	1	2	3	4	5	6
HEX	05	34	00	00	00	02
Client	**Zone**	**Function**				
Byte number	7	8				
HEX	01	11				

Response of controller

Server	**Tid**		**Pid**		**Length**	
Byte number	1	2	3	4	5	6
HEX	05	34	00	00	00	0B

Server	**Zone**	**Function**	**Byte Count**	**Identification**		**1. Word**		**2. Word**		**3. Word**	
Byte number	7	8	9	10	11	12	13	14	15	16	17
HEX	01	11	08	10	FF	50 ('P')	80 (128)	01 (01)	2B (43)	09 (09)	0A (A)

5 Appendix

5.1 Version History

Version	Date	Changes
1.00.02	03-04-2014	Hot runner controller profiTEMP added
1.00.01	01-29-2010	Character adjustment (pdf) corrected
1.00.00	11-23-2009	First publication