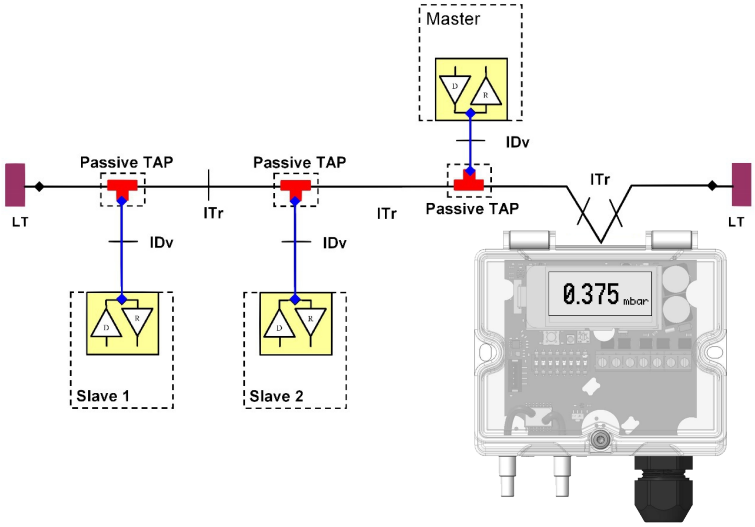
developing solutions

FISCHER
MESS- UND REGELTECHNIK



Handbuch

Modbus RTU

Protokollbeschreibung
für die Produktlinien DE80, DE85 und FT80

09015358 • HB_DE_Modbus_DE8x • Rev. ST4-A • 01/22

Impressum

Hersteller: **FISCHER Mess- und Regeltechnik GmbH**

Bielefelderstr. 37a
D-32107 Bad Salzuflen

Telefon: +49 5222 974 0
Telefax: +49 5222 7170

eMail: info@fischermesstechnik.de
web: www.fischermesstechnik.de

Technische Redaktion: Dokumentationsbeauftragter: T. Malischewski
Technischer Redakteur: R. Kleemann

Versionsgeschichte

Rev. ST4-A 01/22	Version 1 (Erstausgabe)

Inhaltsverzeichnis

1 Einleitung

Das Modbus Protokoll ist ein Kommunikationsprotokoll, dass auf einer Master/Slave Architektur basiert. Alle FISCHER Produkte arbeiten in der Betriebsart Modbus RTU.

Dieses Handbuch ist für einen Leser mit grundlegenden Kenntnissen des Modbus Protokolls verfasst. Hinweise auf einschlägige Fachliteratur zu diesem Thema finden Sie am Ende dieses Handbuchs.

1.1 Modbus Infrastruktur

Die Kommunikation mit den FISCHER Geräten erfordert einen seriellen zwei Draht Bus (2W) Bus gemäß dem EIA/TIA-485 Standard. Alle angeschlossenen Geräte müssen durch eine dritte Leitung (Common) auf ein gemeinsames Bezugspotenzial gelegt werden. Der Busabschluss erfolgt durch einen 120Ω 0,5W Widerstand. Die Pull up/down Widerstände werden gewöhnlich beim Master gesetzt. In der Regel können bis zu 32 Slaves ohne Repeater angeschlossen werden.

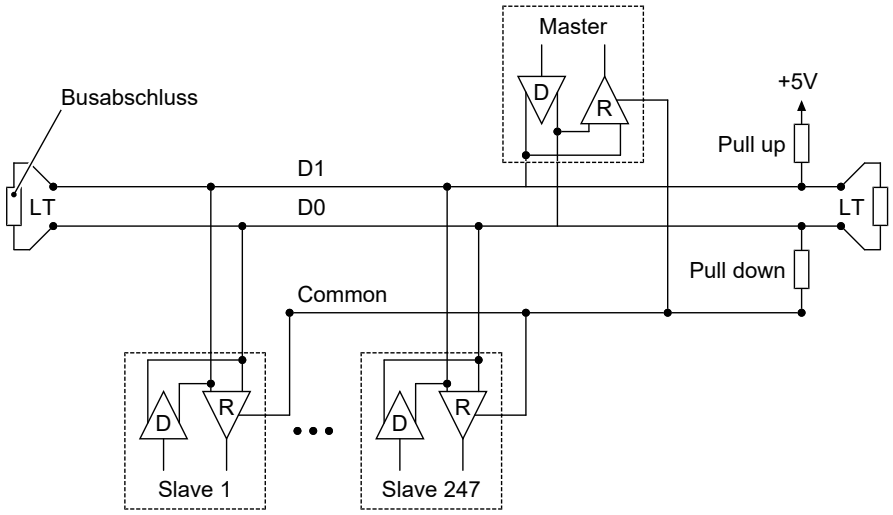


Abb. 1: Modbus Infrastruktur

HINWEIS

Passive TAP

Bei Anschluss der Geräte über einen Passive TAP (z.B. T-Anschlussadapter) können die Geräte vom BUS getrennt werden ohne den Bus zu unterbrechen.

1.2 Modbus RTU Protokoll

Beim Modbus RTU werden Daten in binärer Form übertragen. Am seriellen Modbus dürfen gleichzeitig ein einziger Master und bis zu 247 Slaves angeschlossen werden.

Es gelten folgende grundsätzliche Regeln.

- Eine Modbus Transaktion wird ausschließlich vom Master initiiert.
- Zur gleichen Zeit findet stets nur eine einzige Modbus Transaktion statt.
- Ohne Request vom Master sendet ein Slave niemals Daten.
- Slaves können nicht miteinander kommunizieren.

1.3 Modbus Transaktion

Eine Modbus Transaktion besteht aus zwei Teilen. Einer Anfrage (Request) durch den Master und einer Antwort (Response) vom Slave.

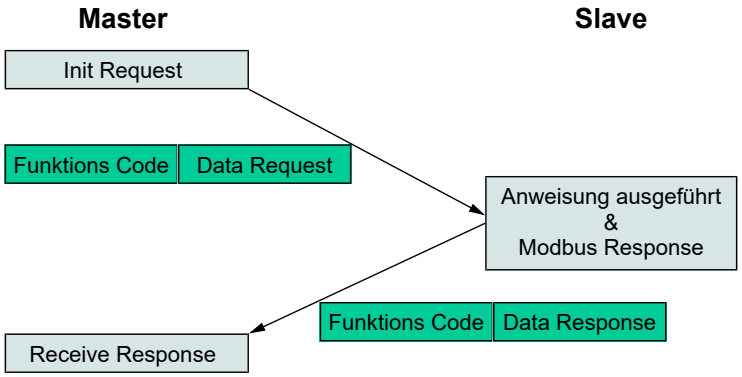


Abb. 2: Fehlerfreier Request/Response Zyklus

Tritt während einer Modbus Transaktion ein Fehler auf, so wird in der Modbus Response Nachricht der Funktionskode durch einen speziellen Funktionskode mit Fehlerindikator ersetzt und im Datenfeld eine nähere Beschreibung des Fehlers gesendet.

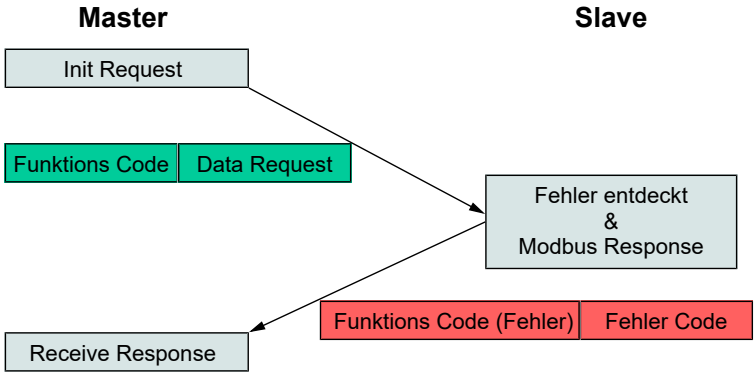


Abb. 3: Fehlerhafter Request/Response Zyklus

1.4 Modbus Frame

Ein Modbus Datenframe setzt sich aus zwei Komponenten zusammen.

- Protocoll Data Unit (PDU)
- Application Data Unit (ADU)

Die innere Datenstruktur ist die PDU und für die Kapselung des Frames in das jeweilige Protokoll der Datenübertragung kommen zusätzliche Datenfelder hinzu.

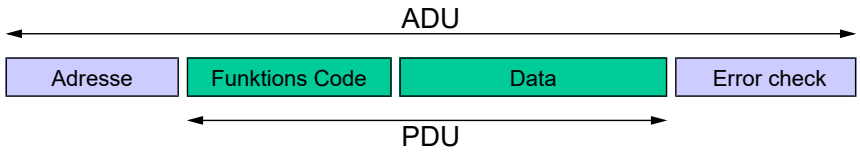


Abb. 4: MODBUS Frame

Beim Modbus RTU Protokoll enthält das Adressfeld die Slave-Adresse. Der Adressraum umfasst die Adressen 1 bis 255. Wenn der Slave eine Response sendet, platziert er seine eigene Adresse in das Adressfeld. Dadurch ‚weiss' der Master welcher Slave sendet. Der Funktions Code gibt an welche Aktion auszuführen ist. Im nachfolgenden Datenfeld sind Request und Response Parameter enthalten. Das Feld Error check enthält das Ergebnis einer CRC Prüfung des Inhalts der Sendung.

1.5 Modbus Datenübertragung

Im RTU Modus wird jede Nachricht als kontinuierlicher binärer Strom von Zeichen über den seriellen Bus gesendet.

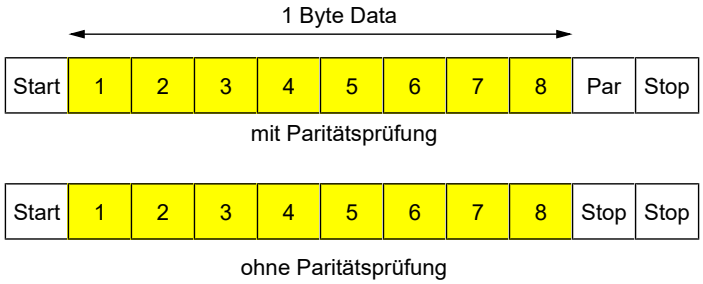


Abb. 5: Bit Sequence

Für das Paritätsbit ist Even Parity als Standardwert eingestellt. Es kann aber auch Odd Parity und No Parity verwendet werden. Wird No Parity verwendet, so wird ein weiteres Stopp-Bit eingefügt.

Eine Modbus Nachricht wird von dem übertragenden Gerät in einen sogenannten Frame gesetzt. Die maximale Größe einer Nachricht beträgt 256 Byte. Anfangs- und Endpunkt eines Frames sind wohldefiniert. Dies erlaubt dem empfangenden Gerät Beginn und Ende einer Nachricht zu erkennen.

Eine Übertragung startet mit einer Pause von mindestens 3,5 Zeichen (char). Dann werden die Frames gesendet. Nach jedem Frame muss ein Ruheintervall (t3,5) mit einer Länge von mindesten 3,5 Zeichen folgen, bevor das nächste Frame gesendet wird. Zwischen zwei Zeichen muss ein weiteres Ruheintervall (t1,5) mit einer Länge von maximal 1,5 Zeichen eingehalten werden. Die gesamte Sendung muss als kontinuierlicher Strom von Zeichen gesendet werden.

Werden die Ruheintervalle nicht eingehalten oder bricht der Zeichenstrom ab, so wird die Sendung für ungültig erklärt.

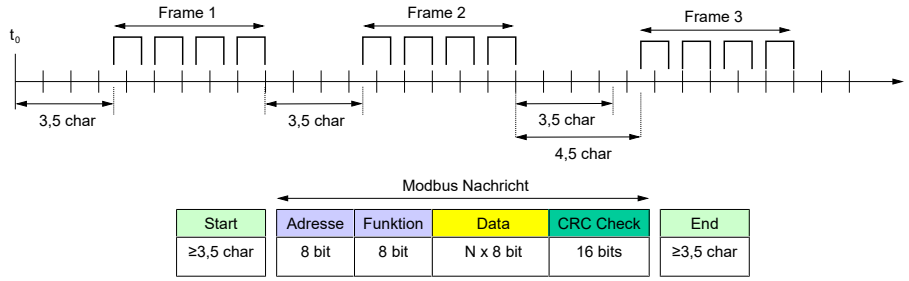


Abb. 6: Modbus Message Frame

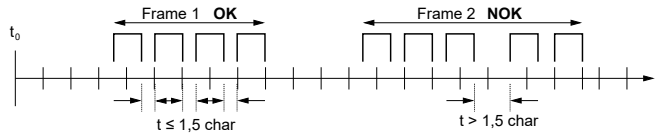


Abb. 7: Fehlerhafte Übertragung (Beispiel)

2 Funktionen

2.1 Allgemeines

Für den Zugriff auf Daten bietet das Modbus Protokoll eine Reihe von unterschiedlichen Möglichkeiten:

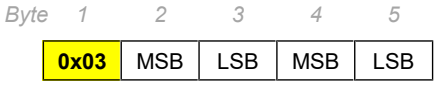
Typ	Zugriff	Name	Code
Daten	16 Bit	Read Holding Register	03
		Read Input Register	04
		Write Single Register	06
		Write Multiple Register	16
Diagnose		Report Server ID	17

2.2 Funktions Code [03] "Read Holding Register"

Dieser Funktions Code wird benutzt um Holding Register zu lesen. Die maximal mögliche Anzahl der Register, die in einer Nachricht adressiert werden können, beträgt 125.

Request

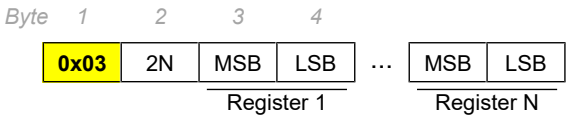
Die Anfrage enthält die Adresse des ersten zu lesenden Registers und die Anzahl der zu lesenden Register. Die Adressierung der Register beginnt bei 0 wohingegen die Nummerierung der Register bei 1 beginnt.



Byte	Feldname	Größe	Wertebereich
1	Funktions Code	1 Byte	0x03
2,3	Start Adresse	2 Byte	0x0000 to 0xFFFF
4,5	Anzahl Register	2 Byte	0x0001 to 0x007D (1…125)

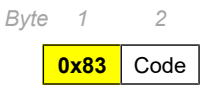
Response

Die Antwort enthält für jedes gelesene Register zwei Bytes, daher ist Anzahl der Bytes das 2fache der Anzahl der Register (N).



Byte	Feldname	Größe	Wertebereich
1	Funktions Code	1 Byte	0x03
2	Anzahl der Bytes	2 Byte	2N
3,4	Holding Register	N x 2 Byte	16 Bit Wert

Error



Byte	Feldname	Größe	Wertebereich
1	Funktions Code (Fehler)	1 Byte	0x83
2	Fehlercode	1 Byte	Code s. Tabelle

Folgende Fehlercodes sind möglich:

0x01	Die Funktion wird nicht unterstützt
0x02	Eine ungültige Adresse wird referenziert
0x03	Die Anfrage entspricht nicht dem erwarteten Format; die Anzahl der angefragten Register ist größer als 125

Beispiel:

- Holding Register 108 bis 110 auslesen
- Inhalt Register 108= 0x000A
- Inhalt Register 109= 0x000B
- Inhalt Register 110= 0x000C

Byte	1	2	3	4	5	6	7	8
Request	**0x03**	0x00	0x6B	0x00	0x03			
Response	**0x03**	0x06	0x00	0x0A	0x00	0x0B	0x00	0x0C

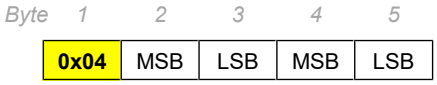
Request		**Response**	
Feldname	**Wert**	**Feldname**	**Wert**
Funktions Code	0x03	Funktions Code	0x03
Start Adresse MSB	0x00	Anzahl der Bytes	0x06
Start Adresse LSB	0x6B	Holding Register 108 MSB	0x00
Anzahl der Register MSB	0x00	Holding Register 108 LSB	0x0A
Anzahl der Register LSB	0x03	Holding Register 109 MSB	0x00
		Holding Register 109 LSB	0x0B
		Holding Register 110 MSB	0x00
		Holding Register 110 LSB	0x0C

2.3 Funktions Code [04] "Read Input Register"

Dieser Funktions Code wird benutzt um Input Register zu lesen. Die maximal mögliche Anzahl der Register, die in einer Nachricht adressiert werden können, beträgt 125.

Request

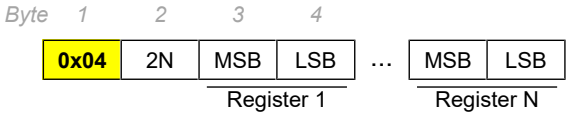
Die Anfrage enthält die Adresse des ersten zu lesenden Registers und die Anzahl der zu lesenden Register. Die Adressierung der Register beginnt bei 0 wohingegen die Nummerierung der Register bei 1 beginnt.



Byte	Feldname	Größe	Wertebereich
1	Funktions Code	1 Byte	0x04
2,3	Start Adresse	2 Byte	0x0000 to 0xFFFF
4,5	Anzahl Register	2 Byte	0x0001 to 0x007D (1…125)

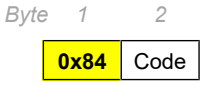
Response

Die Antwort enthält für jedes gelesene Register zwei Bytes, daher ist Anzahl der Bytes das 2fache der Anzahl der Register.



Byte	Feldname	Größe	Wertebereich
1	Funktions Code	1 Byte	0x04
2	Anzahl der Bytes	2 Byte	2N
3,4	Input Register	N x 2 Byte	16 Bit Wert

Error



Byte		Größe	Wertebereich
1	Funktions Code (Fehler)	1 Byte	0x84
2	Fehlercode	1 Byte	Code s. Tabelle

Folgende Fehlercodes sind möglich:

0x01	Die Funktion wird nicht unterstützt
0x02	Eine ungültige Adresse wird referenziert
0x03	Die Anfrage entspricht nicht dem erwarteten Format; die Anzahl der angefragten Register ist größer als 125

Beispiel:

- Input Register 9 auslesen
- Inhalt von Register 9 = 0x000A

Byte	1	2	3	4	5
Request	**0x04**	0x00	0x08	0x00	0x01
Response	**0x04**	0x02	0x00	0x0A	

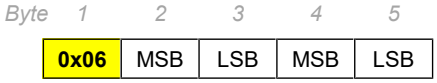
Request		Response	
Feldname	**Wert**	**Feldname**	**Wert**
Funktions Code	0x04	Funktions Code	0x04
Start Adresse MSB	0x00	Anzahl der Bytes	0x02
Start Adresse LSB	0x08	Input Register 9 MSB	0x00
Anzahl der Register MSB	0x00	Input Register 9 LSB	0x0A
Anzahl der Register LSB	0x01		

2.4 Funktions Code [06] "Write Single Register"

Dieser Funktions Code wird benutzt um ein einzelnes Holding-Register zu schreiben.

Request

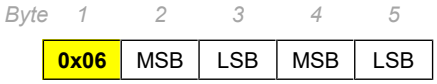
Die Anfrage enthält die Adresse des zu schreibenden Registers und den zu schreibenden Wert.



Byte	Feldname	Größe	Wertebereich
1	Funktions Code	1 Byte	0x06
2,3	Register Adresse	2 Byte	0x0000 to 0xFFFF
4,5	Register Wert	2 Byte	0x0000 to 0xFFFF

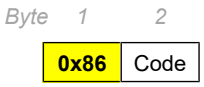
Response

Die Antwort enthält die Register Adresse und den geschriebenen Wert.



Byte	Feldname	Größe	Wertebereich
1	Funktions Code	1 Byte	0x06
2	Register Adresse	2 Byte	0x0000 to 0xFFFF
3,4	Register Wert	2 Byte	0x0000 to 0xFFFF

Error



Byte		Größe	Wertebereich
1	Funktions Code (Fehler)	1 Byte	0x86
2	Fehlercode	1 Byte	Code s. Tabelle

Folgende Fehlercodes sind möglich:

0x01	Die Funktion wird nicht unterstützt
0x02	Eine ungültige Adresse wird referenziert
0x03	Die Anfrage entspricht nicht dem erwarteten Format

Beispiel:

- Register 2 schreiben
- Zu schreibender Wert = 0x0003

Byte	1	2	3	4	5
Request	**0x06**	0x00	0x01	0x00	0x03
Response	**0x06**	0x00	0x01	0x00	0x03

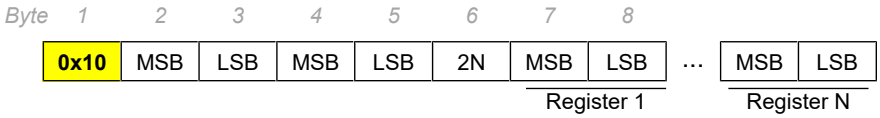
Request		Response	
Feldname	**Wert**	**Feldname**	**Wert**
Funktions Code	0x06	Funktions Code	0x06
Register Adresse MSB	0x00	Register Adresse MSB	0x00
Register Adresse LSB	0x01	Register Adresse LSB	0x01
Register Wert MSB	0x00	Register Wert MSB	0x00
Register Wert LSB	0x03	Register Wert LSB	0x03

2.5 Funktions Code [16] "Write Multiple Registers"

Dieser Funktions Code wird benutzt um einen Block aufeinanderfolgender Register zu schreiben. Die maximal mögliche Anzahl der Register, die in einer Nachricht adressiert werden können, beträgt 123.

Request

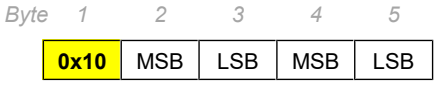
Die Anfrage enthält die Adresse des ersten zu schreibenden Registers und die Anzahl der zu schreibenden Register. Die Adressierung der Register beginnt bei 0 wohingegen die Nummerierung der Register bei 1 beginnt.



Byte	Feldname	Größe	Wertebereich
1	Funktions Code	1 Byte	0x10
2,3	Start Adresse	2 Byte	0x0000 to 0xFFFF
4,5	Anzahl Register	2 Byte	0x0001 to 0x007B (1…123)
6	Anzahl der Bytes	1 Byte	2 x N
7,8	Register Wert	N x 2 Byte	Wert

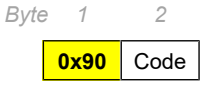
Response

Die Antwort enthält die Startadresse und die Anzahl der geschriebenen Register.



Byte	Feldname	Größe	Wertebereich
1	Funktions Code	1 Byte	0x10
2,3	Start Adresse	2 Byte	0x0000 to 0xFFFF
4,5	Anzahl der Register	2 Byte	0x0001 to 0x007B (1…123)

Error



Byte		Größe	Wertebereich
1	Funktions Code (Fehler)	1 Byte	0x90
2	Fehlercode	1 Byte	Code s. Tabelle

Folgende Fehlercodes sind möglich:

0x01	Die Funktion wird nicht unterstützt
0x02	Eine ungültige Adresse wird referenziert
0x03	Die Anfrage entspricht nicht dem erwarteten Format; die Anzahl der angefragten Register ist größer als 123; die Anzahl Datenbyte passt nicht zur Registeranzahl

Beispiel:

- 2 Register schreiben
- Start Adresse = 0x0001
- Inhalt von Register 2 = 0x000A
- Inhalt von Register 3 = 0x0102

Byte	1	2	3	4	5	6	7	8	9	10
Request	**0x10**	0x00	0x01	0x00	0x02	0x04	0x00	0x0A	0x01	0x02
Response	**0x10**	0x00	0x01	0x00	0x02					

Request		Response	
Feldname	**Wert**	**Feldname**	**Wert**
Funktions Code	0x10	Funktions Code	0x10
Start Adresse MSB	0x00	Start Adresse MSB	0x00
Start Adresse LSB	0x01	Start Adresse LSB	0x01
Anzahl der Register MSB	0x00	Anzahl der Register MSB	0x00
Anzahl der Register LSB	0x02	Anzahl der Register LSB	0x02
Anzahl der Bytes	0x04		
Register Wert MSB	0x00		
Register Wert LSB	0x0A		
Register Wert MSB	0x01		
Register Wert LSB	0x02		

2.6 Funktions Code [17] "Report Server ID"

Dieser Funktions Code erlaubt die Abfrage gerätespezifischer Daten.

Request

Die Anfrage enthält nur den Funktionscode.

Byte 1

0x11

Byte	Feldname	Größe	Wertebereich
1	Funktions Code	1 Byte	0x11

Response

Die Antwort besteht aus zwei Datenbytes. Das letzte Byte ist immer 0xFF.

Byte 1 2 3 4

0x11	0x02	Byte1	0xFF

Byte	Feldname	Größe	Wertebereich
1	Funktions Code	1 Byte	0x11
2	Anzahl der Bytes	1 Byte	0x02
3	Byte 1	1 Byte	Code s. Tabelle
4	Ende	1 Byte	0xFF

Gerät	Byte 1
DE80	0x00
DE85	0x01
FT80	0x02

Error

Byte 1 2

0x91	Code

Byte	Feldname	Größe	Wertebereich
1	Funktions Code (Fehler)	1 Byte	0x91
2	Fehlercode	1 Byte	Code s. Tabelle

Folgende Fehlercodes sind möglich:

0x01	Die Funktion wird nicht unterstützt
0x03	Die Anfrage entspricht nicht dem erwarteten Format

3 Datentypen

3.1 Integer (16 Bit)

- Standard-Format für Register
- Besteht aus zwei Bytes in einer Modbus-Nachricht
- Das höherwertige Byte (Bits 8 bis 15) wird immer zuerst gesendet
- Für vorzeichenbehaftete Ganzzahlen wird das Zweierkomplement-Format verwendet.

	Wertebereich
unsigned Integer	0 … 65535
signed Integer	-32768 … +32767

3.2 Float

- Fließkommazahlen werden im IEEE-475 Single Precision-Format übertragen.
- Sie bestehen aus zwei Registern (vier Bytes).
- Die Bytereihenfolge kann zwischen dem Big Endian- (das höherwertigste Byte zuerst) und dem Little Endian-Format (das niederwertigste Byte zuerst) geändert werden.

Beispiel:

$1234{,}56_{10} = 449A51EC_{16}$

Format	Reg. 1 MSB	Reg. 1 LSB	Reg. 2 MSB	Reg. 2 LSB
Big Endian	0x44	0x9A	0x51	0xEC
Little Endian	0xEC	0x51	0x9A	0x44

3.3 Character

- Zeichenketten werden mit je zwei Zeichen pro Register übertragen
- Das vorangehende Zeichen wird dabei im höherwertigen Byte (MSB) und das nachfolgende Zeichen im niederwertigen Byte (LSB) des Registers gespeichert
- Für Zeichenketten mit ungerader Länge ist das letzte Zeichen immer ein Nullzeichen (0x00).

Beispiel:

Zeichenkette = "FISCHER"

Reg. 1 MSB	Reg. 1 LSB	Reg. 2 MSB	Reg. 2 LSB	Reg. 3 MSB	Reg. 3 LSB	Reg. 4 MSB	Reg. 4 LSB
'F'	'I'	'S'	'C'	'H'	'E'	'R'	'\0'
0x46	0x49	0x53	0x43	0x48	0x45	0x52	0x00

4 Adressen

Folgende Abkürzungen werden verwendet:

Datentyp	Abkürzung	Beschreibung
Float	Float	Fließkommazahl
Unsigned Integrer	UInt	Ganzzahl ohne Vorzeichen
Signed Integer	SInt	Ganzzahl mit Vorzeichen
Character	Char	Zeichenkette

4.1 DE80 Differenzdrucktransmitter

4.1.1 Messwerte

Reg.	Adresse		Länge	Format	Beschreibung	Messwerte	Zugriff	
	Dez.	Hex.					Read	Write
1	0	0x0000	2	Float	Druck	**Messwert Kanal 1**	x	
2	1	0x0001						
					Einheit	*Pa oder mbar*		
5	4	0x0004	2	Float	Druck	**Messwert Rechenkanal 1**	x	
6	5	0x0006			(radiziert)			
					Einheit	*%*		
11	10	0x000A	1	UInt		**Sensor Fehlersignal**	x	
					Wert	*0: kein Fehler*		
					Wert	*1: Fehler*		

4.1.2 Gerätedaten

Reg.	Adresse		Länge	Format	Beschreibung	Messwerte	Zugriff	
	Dez.	Hex.					Read	Write
1001	1000	0x03E8	1	UInt		**Gerätetyp**	x	
					Wert	*0: DE80*		
1002	1001	0x03E9	3	Char		**Firmware Version**	x	
1003	1002	0x03EA						
1004	1003	0x03EB						
					Wert	*6 Zeichen*		

4.1.3 Konfiguration

Reg.	Adresse		Länge	Format	Beschreibung	Messwerte	Zugriff	
	Dez.	Hex.					Read	Write
1101	1100	0x044C	2	Float		**Messbereich Kanal 1 Anfang**	x	
1102	1101	0x044D						
					Einheit	*Pa oder mbar*		
1103	1102	0x044E	2	Float		**Messbereich Kanal 1 Ende**	x	
1104	1103	0x044F						
					Einheit	*Pa oder mbar*		
1105	1104	0x0450	1	UInt		**Einheit Kanal 1**	x	x
					Wert	*0: mbar*		
					Wert	*1: Pa*		
1106	1105	0x0451	1	UInt		**Messwertanzeige Kanal 1**	x	x
					Wert	*0: normal*		
					Wert	*1: radiziert*		
1107	1106	0x0452	2	Float		**Druckoffset Kanal 1**	x	x
1108	1107	0x0451						
					Einheit	*Pa oder mbar*		
					Wert	*≤ 33% des Grundmessbereichs*		
1109	1108	0x0454	1	UInt		**Dämpfung Kanal 1**		
					Einheit	*ms*		
					Wert	*< 500: keine Dämpfung*		
					Wert	*500 … 5000*		
1110	1109	0x0455	1	UInt		**Messbereichs-Spreizung Kanal 1**	x	x
					Wert	0: Bereich 1 (Grundmessbereich)		
					Wert	1: Bereich 2		
					Wert	2: Bereich 3		
					Wert	3: Bereich 4		

4.2 DE85 Differenzdrucktransmitter

4.2.1 Messwerte

Reg.	Adresse		Länge	Format	Beschreibung	Messwerte	Zugriff	
	Dez.	Hex.					Read	Write
1	0	0x0000	2	Float	Druck	**Messwert Kanal 1**	x	
2	1	0x0001						
					Einheit	*Pa oder mbar*		
5	4	0x0004	2	Float	Druck	**Messwert Rechenkanal 1**	x	
6	5	0x0006			(radiziert)			
					Einheit	*%*		
11	10	0x000A	1	UInt		**Sensor Fehlersignal**	x	
					Wert	*0: kein Fehler*		
					Wert	*1: Fehler*		

4.2.2 Gerätedaten

Reg.	Adresse		Länge	Format	Beschreibung	Messwerte	Zugriff	
	Dez.	Hex.					Read	Write
1001	1000	0x03E8	1	UInt		**Gerätetyp**	x	
					Wert	*1: DE85*		
1002	1001	0x03E9	3	Char		**Firmware Version**	x	
1003	1002	0x03EA						
1004	1003	0x03EB						
					Wert	*6 Zeichen*		

4.2.3 Konfiguration

Reg.	Adresse		Länge	Format	Beschreibung	Messwerte	Zugriff	
	Dez.	Hex.					Read	Write
1101	1100	0x044C	2	Float		**Messbereich Kanal 1 Anfang**	x	
1102	1101	0x044D						
					Einheit	*Pa oder mbar*		
1103	1102	0x044E	2	Float		**Messbereich Kanal 1 Ende**	x	
1104	1103	0x044F						
					Einheit	*Pa oder mbar*		
1105	1104	0x0450	1	UInt		**Einheit Kanal 1**	x	x
					Wert	*0: mbar*		
					Wert	*1: Pa*		
1106	1105	0x0451	1	UInt		**Messwertanzeige Kanal 1**	x	x
					Wert	*0: normal*		
					Wert	*1: radiziert*		
1107	1106	0x0452	2	Float		**Druckoffset Kanal 1**	x	x
1108	1107	0x0451						
					Einheit	*Pa oder mbar*		
					Wert	*≤ 33% des Grundmessbereichs*		
1109	1108	0x0454	1	UInt		**Dämpfung Kanal 1**		
					Einheit	*ms*		
					Wert	*< 500: keine Dämpfung*		
					Wert	*500 … 5000*		

4.3 FT80 Feuchte- und Temperatur Messgerät

4.3.1 Messwerte

Reg.	Adresse		Länge	Format	Beschreibung	Messwerte	Zugriff	
	Dez.	Hex.					Read	Write
1	0	0x0000	2	Float	Temperatur	**Messwert Kanal 1**	x	
2	1	0x0001						
					Einheit	*°C oder °F*		
3	2	0x0002	2	Float	Feuchte	**Messwert Kanal 2**	x	
4	3	0x0003						
					Einheit	*%rH*		
5	4	0x0004	2	Float	Enthalpie (Luft)	**Messwert Rechenkanal 1**	x	
6	5	0x0005						
					Einheit	*kJ/kg*		
7	6	0x0006	2	Float	Absolute Feuchte	**Messwert Rechenkanal 2**	x	
8	7	0x0007						
					Einheit	*g/m* ^{3}		
9	8	0x0008	2	Float	Taupunkt Temperatur	**Messwert Rechenkanal 3**	x	
10	9	0x0009						
					Einheit	*°C oder °F*		
11	10	0x000A	1	UInt		**Sensor Fehlersignal**	x	
					Wert	*0: kein Fehler*		
					Wert	*1: Fehler*		

4.3.2 Gerätedaten

Reg.	Adresse		Länge	Format	Beschreibung	Messwerte	Zugriff	
	Dez.	Hex.					Read	Write
1001	1000	0x03E8	1	UInt		**Gerätetyp**	x	
					Wert	*2: FT80*		
1002	1001	0x03E9	3	Char		**Firmware Version**	x	
1003	1002	0x03EA						
1004	1003	0x03EB						
					Wert	*6 Zeichen*		

4.3.3 Konfiguration

Reg.	Adresse		Länge	Format	Beschreibung	Messwerte	Zugriff	
	Dez.	Hex.					Read	Write
1101	1100	0x044C	2	Float		**Messbereich Kanal 1 Anfang**	x	
1102	1101	0x044D						
					Einheit	*°C oder °F*		
1103	1102	0x044E	2	Float		**Messbereich Kanal 1 Ende**	x	
1104	1103	0x044F						
					Einheit	*°C oder °F*		
1105	1104	0x0450	1	UInt		**Einheit Kanal 1**	x	x
					Wert	*0: °C*		
					Wert	*1: °F*		
1201	1200	0x04B0	2	Float		**Messbereich Kanal 2 Anfang**	x	
1202	1201	0x04B1						
					Einheit	*%rH*		
1203	1202	0x04B2	2	Float		**Messbereich Kanal 2 Ende**	x	
1204	1203	0x04B3						
					Einheit	*%rH*		
1205	1204	0x04B4	1	UInt		**Messwertanzeige Kanal 2**	x	x
					Wert	*0: relative Feuchte*		
					Wert	*1: Enthalpie*		
					Wert	*2: absolute Feuchte*		
					Wert	*3: Taupunkt-Temperatur*		

5 Anhang

5.1 Literatur

„IEEE Standard for Floating-Point Arithmetic.“ 29. 08 2008. <http://ieeexplore.ieee.org/document/4610935/>.

„Modbus Application Protocol v1.1b3.“ 26. 04 2012. <http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b3.pdf>.

Glossar

ADU

Die Application Data Unit (ADU) ist der vollständige Kommando-/Datenblock des Kommunikationsprotokolls.

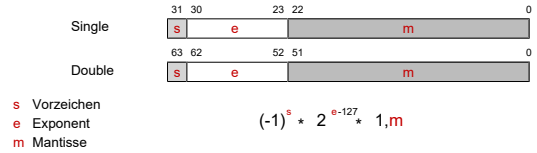
char

Abkürzung für character (engl.). Char als Datentyp legt fest, dass die einzelnen Zeichen eines Speicherbereichs aus je (i. d. R.) 8 Bits bestehen, die je ein darstellbares Zeichen (Buchstabe, Ziffer, Sonderzeichen ...) repräsentieren. Welches Zeichen dies ist, ergibt sich aus dem Inhalt der Speicherstelle

EIA-485

EIA-485, auch als RS-485 bezeichnet, ist ein Industriestandard für eine Schnittstelle mit asynchroner serieller Datenübertragung.

IEEE-475



Der IEEE754-Standard schreibt mehrere Datenformate vor. Die wichtigsten sind das Single- und das Double-Format . Diese Formate bestehen aus einem Vorzeichenbit s, dem Exponenten e und der Mantisse m.

Master/Slave

Master/Slave ist eine Form der hierarchischen Verwaltung des Zugriffs auf eine gemeinsame Ressource meist in Form eines gemeinsamen Datenkanals. Ein Teilnehmer ist der Master, alle anderen sind die Slaves. Der Master hat als einziger das Recht, unaufgefordert auf die gemeinsame Ressource zuzugreifen. Der Slave kann von sich aus nicht auf die gemeinsame Ressource zugreifen; er muss warten, bis er vom Master gefragt wird.

Nachricht

Prozess der Übertragung von Daten zwischen einem Sender und einem oder mehreren Empfängern.

PDU

Die Protocoll Data Unit (PDU) ist der Datenblock einer Nachricht.

Request

Die Anfoderung des Master an einen Slave, den in der Sendung enthaltenen Fuktions Code auszuführen.

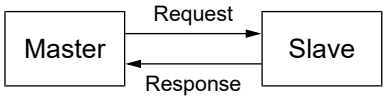
Response

Antwort des Slave an den Master auf einen Request.

RTU

Remote Terminal Unit

Transaktion



Eine Transaktion besteht aus einer Anforderung (Request) vom Master und einer Antwort (Response) vom Slave.

Notizen

FISCHER Mess- und Regeltechnik GmbH
Bielefelder Str. 37a
D-32107 Bad Salzuflen

Tel. +49 5222 974-0
Fax +49 5222 7170
www.fischermesstechnik.de
info@fischermesstechnik.de