TZA 401

Messrechner
MODBUS-Schnittstelle

Bedienungsanleitung

42/18-58 DE

Rev. 1.0

Inhaltsverzeichnis

Beschreibung

Die serielle Kommunikation des Meßrechners TZA 401 kann mit Hilfe des Programms TZAKON2 auf zwei unterschiedliche Protokollarten eingestellt werden:

TZAKONFI-Protokoll

Konfigurieren und Parametrieren des TZA 401 mit dem Programm TZAKON2.

MODBUS-Protokoll

Lesen von Meßdaten und Schreiben/Lesen von Konstanten gemäß der MODBUS-Protokoll-Spezifikation.

Die Meßrechner TZA 401 arbeiten dabei immer als „Slaves“, d.h. sie reagieren nur, wenn das überlagerte System, der „Master“ (z.B. ein PC), einen entsprechenden Befehl erteilt. Vom TZA 401 wird nur das RTU-Verfahren und daraus nur die für den TZA 401 wichtigen Funktionen unterstützt.

Nähere Informationen über das MODBUS-Protokoll siehe:

GOULD MODICON MODBUS PROTOCOL
Reference Guide
Gould Inc., Programmable Control Division
P.O Box 3083
Andover, Massachussetts, 01810
PI-MBUS -300 Rev A, November 93

Schnittstellenmodul

Technische Daten

Der TZA 401 ist über eine frontseitige 9polige D-Buchse für folgende Schnittstellen konfigurierbar:

RS-232 Schnittstelle zum Parametrieren mit Hilfe dees Programms TZAKON2

RS-232 zum Anschluß **eines** TZA 401 an einen MODBUS-Masters (z.B. PC)

RS-485 zum Anschluß von maximal 32 MODBUS-Teilnehmern (einschließlich Master)

Das Schnittstellenmodul ist vom TZA 401 galvanisch getrennt. Die Schnittstellenart wird über zwei Steckbrücken eingestellt (siehe Konfigurieranleitung 42/18-51).

Schnittstelle		Pinbelegung			
Art	Brücken	Signal A	Signal B	Null	Schirm
RS-232 Front	B522, B523	RXD Pin 3	TXD Pin 2	 Pin 5	St.- Gehäuse
RS-485 Front	B521, B531	+RX/TX Pin 3	−RX/TX Pin 8	 Pin 5	St.- Gehäuse
RS-485 Messerleiste Klemmleiste	B521, B531 B506, B501, B3 B506, B501, B3	+RX/TX Pin 26z Kl. 16	−RX/TX Pin 26d Kl. 17	 Pin 28z Kl. 18	 Pin 28d PE

Tab. 1 Technische Daten

RS-485-Schnittstelle

Die Schnittstelle kann alternativ zum Frontstecker auch auf die rückseitige Messerleiste (alternativ zum Grenzwertausgang GW2) geführt werden (siehe Bild 1). Es sind max. 32 Busteilnehmer (einschließlich PC) zugelassen. Der Bus (Linienstruktur, keine Abzweigungen, nicht länger als 1200 m) hat Stichleitungen (nicht länger als 30 cm) zu den einzelnen Teilnehmern.

Es ist mindestens ein dreiadriges, geschirmtes Buskabel mit einem verdrillten Adernpaar zur Datenübertragung und einem zusätzlichen isolierten Leiter zum Potentialausgleich zwischen den Anschlüssen „Modulnull“ aller galvanisch getrennten Busteilnehmer zu verwenden. Zum Betrieb von nicht galvanisch getrennten Busteilnehmern ist ein zusätzlicher getrennter Leiter mit großem Querschnitt parallel zum Datenkabel erforderlich.

Blockschaltbild TZA 401-Schnittstelle

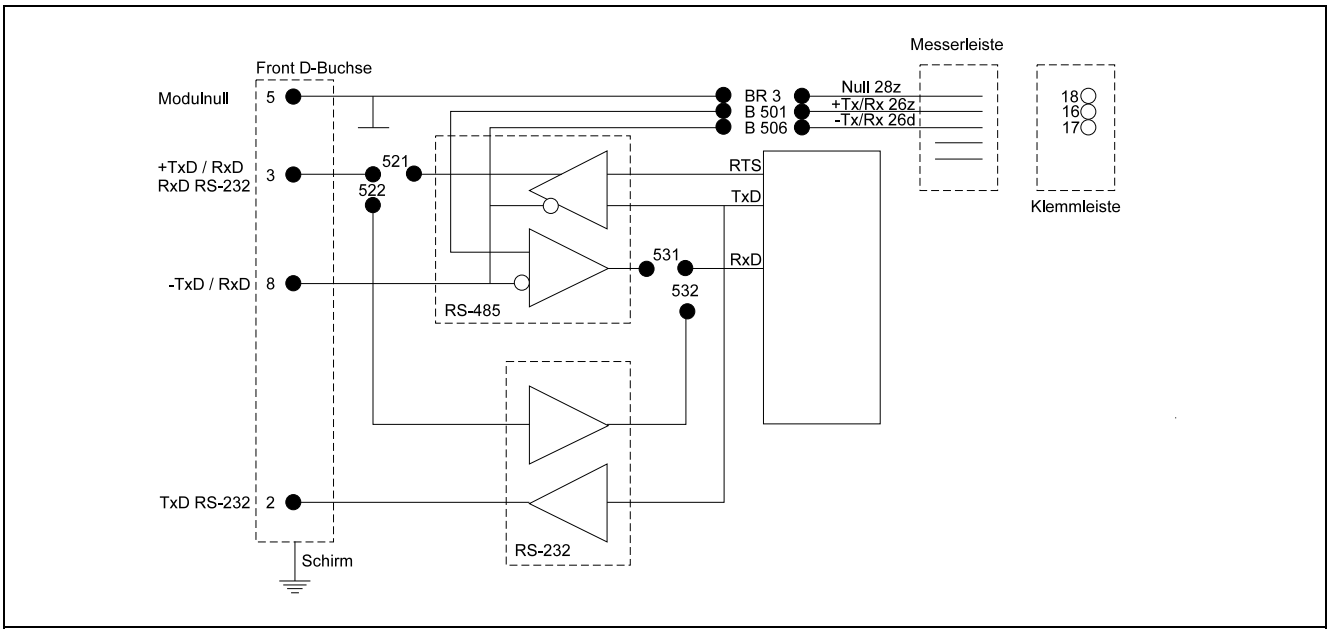

Bild 1 TZA 401-Schnittstelle (umschaltbar: RS-232 oder RS-485)
Z-18981 RS-232: Steckbrücken B 522, B 532
RS-485: Steckbrücken B 521, B 531
RS-485 (MODBUS) Anschluß an Steckerleiste (alternativ zu Grenzwert GW2):
Lötbrücken B 506, B 501, BR 3

MODBUS-Konfiguration über RS-485

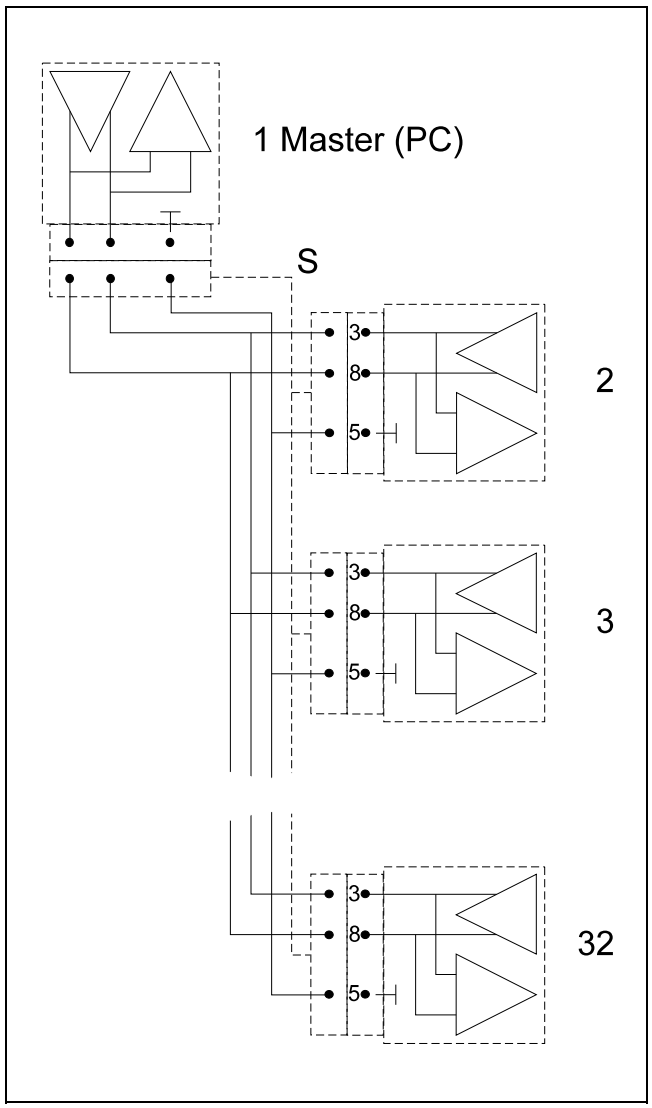

Bild 2 MODBUS-Konfiguration
Z-18981 S Schirm

MODBUS-Datentransfer

Allgemeines

Kommunikationsart

Die Kommunikationsart (TZAKONFI oder MODBUS) wird im Programm TZAKON2 (ab Version 3.2) gewählt (Menüpunkt „MODBUS / TZAKONFI KOMMUNIKATION“ des Hauptmenüs). Der TZA 401 kann nur im TZAKONFI-Mode konfiguriert oder parametriert werden. Vor der Umschaltung müssen folgende Bedingungen erfüllt sein:

- Die TZA 401-Schnittstelle muß auf RS-232 eingestellt sein (B522 und B532).
- Verbindung zwischen TZA 401 und PC nur mit RS232-Kabel.
- Im Meßrechner muß ein Programm geladen und durch einen „Power-On-Start“ gestartet worden sein.
- Das Programm darf keinen PRINT-Befehl ausführen (bei Hartmann & Braun-Rechenprogrammen werden die PRINT-Befehle nach einem Power-On-Start gesperrt).
- Bei eichfähigen Geräten muß die Steckbrücke XB6 auf der I/O-Erweiterungskarte entfernt werden.

Wird „MODBUS“ gewählt, muß eine Stationsadresse von 1...255 angegeben werden.

MODBUS

An einem Bus können beliebige Teilnehmer, die der MODBUS-Spezifikation entsprechen, betrieben werden. Die Anzahl der Teilnehmer richtet sich nach der verwendeten Übertragungstechnik:

RS-232 Ein Master und ein Slave.
RS-485 Ein Master und bis zu 31 Slaves.

Für die Datenübertragung wird eine Kombination von Telegrammzeichen zu einem oder mehreren Telegrammen zusammengefaßt. Diese Telegramme übernehmen auch die „Hand-Shake-Funktion“, indem jedes Telegramm vom Master zum Slave erst beantwortet werden muß, bevor ein neues Telegramm gesendet werden darf.

Im Master (z.B. PC) sind entsprechende Überwachungen notwendig, um nicht antwortende Busteilnehmer auszugrenzen („Time-Out-Überwachung“). Die Time-Out-Zeit richtet sich nach der verwendeten Baudrate und nach der Reaktionszeit der angeschlosenen Teilnehmer und sollte für die Kommunikation mit TZA 401 Meßrechnern mehr als 200 ms betragen.

Der TZA 401 arbeitet mit einer festen Datenübertragungsgeschwindigkeit von 9600 Baud.

Telegrammzeichen (Frame)

Die Telegramme bestehen aus einer Folge von I/0 Informationen. Die zu übertragenden Werte sind in Bytes (= 8 Bit) zerlegt. Jedes dieser Bytes wird ergänzt durch

1 Startbit
1 Stopbit

Es gibt kein Paritätsbit.

In der nachfolgenden Beschreibung wird der Begriff „Byte“ verwendet, auch wenn einschließlich der Start- , Stop-Bits eigentlich 10 Bits übertragen werden.

Übertragungsregeln

Der Ruhezustand (= Pause) der Datenleitung entspricht der logischen „1“.

Vor Beginn der Datenübertragung muß für die Zeit von mindestens 3 Byte der Ruhezustand auf der Datenleitung bestehen.

Zwischen den Bytes eines Telegrammes darf der Abstand nicht größer als 3 Byte werden, da ein Abstand von mehr als 3,5 Bytes als Trennung zwischen zwei Telegrammen definiert ist.

Telegramme

Die Modbus-Telegramme haben folgenden Aufbau:

Pause	Adresse	Funktion	Daten	Prüf-summe	Pause
	1 Byte	1 Byte	n Bytes	2 Bytes	

Zulässige Stationsadressen

Als Stationsadressen der Busteilnehmer sind die Ziffern 1 bis 255 zugelassen.

Die Adresse 0 ist die Globaladresse (Broadcast-Adresse). Diese Adresse wird vom TZA 401 nicht akzeptiert und es wird keine Bestätigung an den Master gegeben.

Prüfsumme CRC

Die Prüfsumme wird über alle Bytes eines Telegrammes (ohne Start-, Stopbits) berechnet.

Für die Ermittlung der Prüfsumme sind im Anhang Beispielprogramme aufgeführt (ab Seite). Details sind der Originaldokumentation über MODBUS zu entnehmen.

Funktionen

Der Meßrechner TZA 401 unterstützt folgende Funktionen:

Code	Bezeichnung	Funktion
01H	READ COIL STATUS	Binärausgangswerte und Statusbits lesen
02H	READ INPUT STATUS	Werte der Binäreingänge lesen
03H	READ HOLDING REGISTERS	REAL-Werte (IEEE-Format) lesen
04H	PRESET SINGLE REGISTER	Geräte-Adresse einschreiben
05H	LOOPBACK DIAGNOSTIC TEST	Testtelegramm zur Diagnose der Konmmunikationsfähigkeit des TZA 401
06H	PRESET MULTIPLE REGISTERS	Einschreiben von REAL Floting-Point-Konstanten

Tab. 2 Funktionen

Funktion 01

Diese Funktion dient zur Abfrage von mehreren aufeinanderfolgenden binären Werten aus den TZA401. Die Broadcast-Adresse 0 ist nicht zulässig.

Beispiel

1. Lesetelegramm vom Master an Slave Nr. 11H

Dieses Telegramm fordert den Binärstatus der TZA 401-Fehler-Status-Register PE_E1 (Register-Adresse 22H) bis PE_AX2 (Register-Adrese 2FH) an (insgesamt 14 Werte).

Adresse	Funktion	Startadresse		Anzahl		Prüfsumme CRC	
		HByte	LByte	HByte	LByte	LByte	HByte
11	01	00	22	00	0E		

(alle Angaben hexadezimal)

2. Antworttelegramm vom Slave Nr. 11H an Master

In der Antwort sind die binären Informationen in wenigen Bytes gepackt. Die erforderliche Anzahl der Bytes errechnet sich aus: Anzahl der Bytes = INT (Werte / 8) + 1 (in diesem Beispiel: Anzahl der Bytes = INT (14 / 8) + 1 = INT (1,75) + 1 = 1 + 1 = 2).

Adresse	Funktion	Anzahl der Bytes	Status 22H...29H	Status 2AH...2FH	Prüfsumme CRC	
					LByte	Hbyte
11	01	02	8 Bit	8 Bit		

(alle Angaben hexadezimal)

Die Anzahl der Bytes sagt aus, wieviel Datenbytes folgen. Im Beispiel sind es 2 Bytes:

Datenbyte Nr. 1
Status 22H bis 29H — Status der Fehlerstatus-Register PE_E1 bis PE_A1

Bit	7	6	5	4	3	2	1	0
Adresse	29	28	27	26	25	24	23	22
Reg. PE_	A1	EB	E6	E5	E4	E3	E2	E1
Status	x	x	x	x	x	x	x	x

(alle Angaben hexadezimal, der Status x kann 0 oder 1 sein)

Datenbyte Nr. 2
Status 2AH bis 2FH — Status der Binäreingänge PE_A2 bis PE_AX2

Da in diesem Byte nur 6 binäre Informationen übertragen werden, sind die Bits 6 und 7 mit „0“ belegt. Die Bits 0 bis 5 enthalten die gewünschten Daten.

Bit	7	6	5	4	3	2	1	0
Adresse			2F	2E	2D	2C	2B	2A
Reg. PE_			AX2	AX1	EX2	EX1	AB	A2
Status	0	0	x	x	x	x	x	x

(alle Angaben hexadezimal, der Status x kann 0 oder 1 sein)

Funktion 02

Diese Funktion dient zur Abfrage von mehreren aufeinanderfolgenden binären Eingangswerten aus den TZA 401. Die Broadcast-Adresse 0 ist nicht zulässig.

Beispiel

1. Lesetelegramm vom Master an Slave Nr. 11H

Dieses Telegramm fordert den Status der TZA 401-Binäreingänge EB1 (Register-Adresse 0) bis EB4 (Register-Adresse 3) an, insgesamt also 4 Werte.

Adresse	Funktion	Startadresse		Anzahl		Prüfsumme CRC	
		HByte	LByte	HByte	LByte	LByte	HByte
11	02	00	00	00	04		

(alle Angaben hexadezimal)

2. Antworttelegramm vom Slave Nr. 11H an Master

In der Antwort sind die binären Informationen in wenigen Bytes gepackt. Die erforderliche Anzahl der Bytes errechnet sich aus: Anzahl der Bytes = INT (Werte / 8) + 1 (in diesem Beispiel: Anzahl der Bytes = INT (4 / 8) + 1 = INT (0,5) + 1 = 0 + 1 = 1).

Adresse	Funktion	Anzahl der Bytes	Status 0 bis 4	Prüfsumme CRC	
				LByte	Hbyte
11	02	01	8 Bit		

(alle Angaben hexadezimal)

Die Anzahl der Bytes sagt aus, wieviel Datenbytes folgen.

Datenbyte
Status 0 bis 4 — Status der Binäreingänge EB1,EB2,EB3 und EB4

Da in diesem Byte nur 4 binäre Informationen übertragen werden, sind die Bits 4 bis 7 mit „0“ belegt. Die Bits 0 bis 3 enthalten die gewünschten Daten.

Bit	7	6	5	4	3	2	1	0
Adresse					3	2	1	0
Reg. PE_					EB4	EB3	EB2	EB1
Status	0	0	0	0	x	x	x	x

(alle Angaben hexadezimal, der Status x kann 0 oder 1 sein)

Funktion 03

Diese Funktion dient zur Abfrage eines Floating-Point-REAL-Wertes aus dem TZA 401. Die Broadcast-Adresse 0 ist nicht zulässig.

REAL-Werte

REAL-Werte sind im TZA 401 im BASIC-52-Floating-Point-Format (6 Byte gepacktes BCD-Format) abgelegt. Zum Lesen dieser Werte wird von der TZA 401-Kommunikations-Firmware eine Konvertierung in das IEEE-Floating-Point-Format vorgenommen.

Das IEEE-Floating-Point-Format setzt sich aus 32 Bits zusammen und belegt immer 2 Register (4 Byte siehe Abschnitt „Berechnung der Daten“). Deshalb müssen in der Spalte „Anzahl“ immer 2 Register angegeben werden.

Beispiel

Lesen des Analogeingangs-Signals „Sig_E3“ aus den Registern 42H und 43H:

Adresse	Funktion	Startadresse		Anzahl		Prüfsumme CRC	
		HByte	LByte	HByte	LByte	LByte	HByte
11	03	00	42	00	02		

(alle Angaben hexadezimal)

Die Antwort hat folgenden Aufbau:

Adresse	Funktion	Anzahl	IEEE-Floating-Point-Wert				Prüfsumme CRC	
		Byte	Date [0] Adresse 42H		Date [1] Adresse 43H			
11	03	4	HByte	LByte	HByte	LByte	LByte	HByte

(alle Angaben hexadezimal)

Die Umrechnung der 4 Bytes in REAL-Werte ist im Abschnitt „Berechnung der Daten“ beschrieben.

Funktion 06

Mit der Funktion 6 kann im TZA 401 nur die MODBUS-Geräteadresse eingestellt bzw. abgeändert werden. Das Überschreiben anderer Register ist nicht möglich. Die Zieladresse muß dabei immer mit 0000H angegeben werden, die Zieldaten beinhalten die neue Geräteadresse.

Beispiel

Geräteadresse 11H abändern in Adresse 12H:

Adresse	Funktion	Zieladresse		Zieldaten		Prüfsumme CRC	
		HByte	LByte	HByte	LByte	LByte	HByte
11	06	00	00	00	12		

(alle Angaben hexadezimal)

Die Antwort hat folgenden Aufbau:

Adresse	Funktion	Zieladresse		Zieldaten		Prüfsumme CRC	
		HByte	LByte	HByte	LByte	LByte	HByte
11	06	00	00	00	12		

(alle Angaben hexadezimal)

Nach diesem Antworttelegramm ist der TZA 401 nun über die Geräteadresse 12 ansprechbar.

Funktion 08

Diese Funktion dient zur Diagnose der Kommunikation. Zur Zeit ist im TZA 401 die „Loopback“-Testfunktion implementiert. Das empfangene Telegramm wird sofort wieder als Antworttelegramm vom TZA 401 zurückgesendet.

Beispiel

Dieses Telegramm ist an Busteilnehmer Nr. 11H gerichtet. Der Busteilnehmer Nr.11H interpretiert dieses Telegramm, prüft die CRC-Checksumme und gibt das gleiche Telegramm wieder an den Master zurück. Der Diagnose-Code muß immer 0000 sein. Andernfalls wird Fehler zurückgesendet. Die Daten können beliebig sein.

Adresse	Funktion	Diagnose-Code		Daten		Prüfsumme CRC	
		HByte	LByte	HByte	LByte	LByte	HByte
11	08	00	00	A5	37		

(alle Angaben hexadezimal)

Die Antwort ist identisch mit der Frage:

Adresse	Funktion	Diagnose-Code		Daten		Prüfsumme CRC	
		HByte	LByte	HByte	LByte	LByte	HByte
11	08	00	00	A5	37		

(alle Angaben hexadezimal)

Telegramm Fehlermeldungen

Jedes Telegramm des Masters an den Slave wird vom Slave auf Gültigkeit geprüft. Bei fehlerhaftem Telegramm wird, mit Ausnahme eines CRC-Fehlers, vom Slave ein Fehlertelegramm generiert und zum Master zurückgesendet. Dazu wird im Antworttelegramm das 7. Bit im Funktionsbyte auf „high“ gesetzt und eine Fehlercode-Nr. mitgeliefert.

Folgende Fehler werden von der Kommunikationseinheit erkannt:

- falsches Funktionsbyte — Code 1
- falsche Startadresse — Code 2
- falsche Anzahl von Werten — Code 2
- Zieldaten nicht interpretierbar — Code 3
- CRC-Prüfsummenfehler: — kein Antworttelegramm

Beispiel

Lese Status der Binäreingänge EB1 bis EB4. Angegeben ist eine falsche Startadresse = 0023H.

Adresse	Funktion	Startadresse		Anzahl		Prüfsumme CRC	
		HByte	LByte	HByte	LByte	LByte	HByte
11	02	00	23	00	04	x	x

(alle Angaben hexadezimal)

Die Antwort lautet:

Adresse	Funktion	Fehler-Code	Prüfsumme CRC	
			LByte	HByte
11	82	02	x	x

(alle Angaben hexadezimal)

Fehlerbit 7 im Funktionsbyte ist gesetzt!
Fehlercode 2: falsche Zieladresse wurde zurückgeliefert!

Funktion 16 (10H)

Mit der Funktion 16 können im TZA 401 REAL-Konstanten eingeschrieben bzw. abgeändert werden. Mit einem Telegramm kann immer nur ein REAL-Wert transferiert werden. Zur Übertragung eines Wertes werden 2 Register (4 Byte) benötigt. In der Spalte „Registerzahl" muß immer „2" und in der Spalte „Bytezahl" immer „4" eingetragen werden. Im TZA 401 stehen Speicherplätze für 71 REAL-Konstanten zur Verfügung. Sie belegen die Registeradressen 100....240 (64H...0F0H).

Beispiel

Geräteadresse 11H; Registeradresse = 64H; Floating-Point-REAL-Wert = 1.234567 → IEEE-Format = 3F9E064B

Adresse	Funktion	Registeradresse		Registerzahl		Anzahl Bytes	1. Daten-Word		2. Daten-Word		Prüfsumme CRC	
		HByte	LByte	Hbyte	LByte		Mantisse Bit 8...15	Mantisse Bit 0...7	Exponent Bit 24...31	Mantisse Bit 16...23	LByte	HByte
11	10	00	64	00	02	04	06	4B	3F	9E	40	72

(alle Angaben hexadezimal)

Die Antwort hat folgenden Aufbau:

Adresse	Funktion	Registeradresse		Registerzahl		Anzahl Bytes	Prüfsumme CRC	
		HByte	LByte	HByte	LByte		LByte	HByte
11	10	00	64	00	02	04	07	02

(alle Angaben hexadezimal)

Wertebereiche

REAL

− 1,175495 E−38 ... 0 ... 3,402823 E+38
(wird in zwei Registern (= 4 Byte) gespeichert)

BOOL

0 und 1

Berechnung der Daten

REAL-Werte

Das in dem TZA 401 verwendete BASIC-52-Floating-Point-Format wird in das IEEE-Format umgewandelt. Das IEEE-Format entspricht dem in den PCs verwendetem Format.

Das MODBUS-Protokoll sieht nur 16-Bit vorzeichenbehaftete Ganzzahlen als Übertragungswerte vor. Um auch Gleitkomma-Zahlen in der maximalen Genauigkeit übertragen zu können, ist auf dem TZA 401 ein Verfahren zum Übertragen eines 32-Bit Wertes im IEEE-Format implementiert worden: das „Pair of Register“-Verfahren.

Das Pair of Register-Verfahren wird auch vom Hartmann & Braun-Kompaktautomatisierungssystem Digimatik untertützt. Dabei werden REAL-Werte (4 Byte IEEE-Format) in zwei aufeinander folgenden 16-Bit-Registern übertragen. Register mit gerader Adresse übertragen das niederwertige „Word“, Register mit ungerader Adresse (gerade + 1) das höherwertige „Word“ (16-Bit-Register). Um die Konsistenz der Darstellung zu erhalten, müssen bei der Übertragung eines 32-Bit-Wertes stets beide Register aufeinanderfolgend übertragen werden.

```
|-------WORD 2-------|-------WORD1-------|

31.30....23.22.....16|15.................0
+----------+----------------------------+
|s| 8 bits |msb    23 bit mantisse    lsb|
+----------+----------------------------+
|     |              |
|     |              +----------- Mantisse
|     +-------------------- Exponent (7fh)
+------------- Sign bit (0 = Pos, 1 = Neg)
```

Der wahre Exponent-Wert ist der Exponent minus 7Fh für das IEEE 4 Byte REAL-Format.

Je nach der im PC verwendeten Programmiersprache kann, direkt oder indirekt, auf die einzelnen Bytes der REAL-Werte zugegriffen werden. Für Rechnersysteme, die ein anderes Zahlenformat verwenden oder keinen Zugriff auf die einzelnen Bestandteile der REAL-Werte zulassen, sind im Anhang Beispielprogramme in C und Quickbasic zur Umwandlung von REAL-Werten in das Bytemuster des IEEE-Formates angegeben.

Lesen eines Pairs of Registers aus einem TZA 401

Mit einen Telegramm der Funktion 03 wird ein BASIC-52-Floating-Point-Wert adressiert, in das IEEE-Format umgewandelt und in zwei Registern (4 Byte) abgelegt. Das Antworttelegramm wird zusammengestellt und zum Master übertragen. Im Master muß dann aus den zwei Datenregistern der Floating-Point-Wert im IEEE-Format wieder zusammengesetzt werden.

Bildungsvorschrift für das Zusammenfügen von zwei Registerwerten mit je 16 Bit in einen (4 Byte IEEE) REAL-Wert, wobei in data[0] der gelesene Wert des geraden Registers, in data[1] der des ungeraden Registers enthalten ist:

```
float   *ptrReal
int     data[2]

ptrReal = (float *)&data[0]
```

Zuordnung der TZA 401-Variablen zu den MODBUS-Registern

Die Zuordnung der einzelnen Parameter und dynamischen Größen zu den Registern geht aus den nachfolgenden Modbus-Registertabellen hervor.

Alle Registernummern sind in dezimaler und hexadezimaler Schreibweise notiert.

REAL-Variablen

Für das Lesen von REAL-Variablen mit der MODBUS-Funktion 03 sind die Register 0 bis 96 reserviert. Zum Lesen der 32-Bit-Werte wird das Pair of Register-Verfahren angewandt. Die Variablen sind den TZA 401 Rechenprogrammen P731, P735, P751, P761 und P765 fest zugeordnet und sind unterteilt in physikalische- und elektrische Meß- bzw. Rechenwerte.

REAL-Konstanten

REAL-Konstanten können mit der Funktion 10H über den MODBUS in den TZA 401 eingeschrieben und mit der Funktion 03 ausgelesen werden. Hierbei wird ebenfall das Pair of Register-Verfahren angewandt. Im TZA 401 sind dafür die Register 100 (64H) bis 240 (0F0H) reserviert. Diese Konstanten dienen dem TZA 401 als Parameterdaten und werden vom jeweils geladenen TZA 401-BASIC-Programm verwaltet.

BOOLEAN-Variablen

Die BOOLEAN-Variablen sind in vier Gruppen geteilt:
- Status (high/low) der Binäreingänge
- Status (high/low) der Binärausgänge
- Fehlerstatus der Hard- und Firmware
- Status der physikalischen Min.-/Max.-Werte der Rechenprogramme P731, P735, P751, P761 und P765

Status Binäreingänge

Mit der MODBUS-Funktion 02 lassen sich alle binären Eingangszustände des TZA 401 auslesen. Siehe MODBUS-Registertabelle der Funktion 02: Eingangszustand. Jedem Eingangszustand wird im Quittungstelegramm ein Bit im Datenbyte zugeordnet. Siehe Datenübertragung Funktion 02.

Status Binärausgänge

Mit der MODBUS-Funktion 01 lassen sich alle binären Ausgangszustände des TZA 401 auslesen. Siehe MODBUS-Registertabelle der Funktion 01: Ausgangszustand. Hierfür sind die Adressen 0 bis 6 reserviert. Jedem Ausgangszustand wird im Quittungstelegramm ein Bit im Datenbyte zugeordnet. Siehe Datenübertragung Funktion 01.

Fehlerstatus der Hard- und Firmware

Mit der MODBUS-Funktion 01 lassen sich alle Fehlerregister des TZA 401 auslesen. Siehe MODBUS-Registertabelle der Funktion 01: Fehlerstatus. Hierfür sind die Adressen 7 bis 29 reserviert. Jedem Fehlerstatus wird im Quittungstelegramm ein Bit im Datenbyte zugeordnet. Siehe Datenübertragung Funktion 01.

Status physikalische Min.-/Max.-Werte

Mit der MODBUS-Funktion 01 lassen sich alle Statusregister der Min.-/Max.-Grenzwerte auslesen. Siehe MODBUS-Registertabelle der Funktion 01: Min.-Max.-Status. Hierfür sind die Adressen 34 bis 47 reserviert. Jedem Fehlerstatus wird im Quittungstelegramm ein Bit im Datenbyte zugeordnet. Siehe Datenübertragung Funktion 01.

MODBUS-Registertabellen der TZA 401-Telegramm-funktionen

Register der Funktion 01: „Lese Ausgangsstatus“ High/Low-Pegel der Binärausgänge
0 = Low-Pegel, 1 = High-Pegel

Registeradressen Dez	Hex	Bezeichnung	Datentyp	Beschreibung
0	0	ERR	BOOL	Fehlersignalausgang ERR (Grundkarte)
1	1	AB1	BOOL	Binärausgang aktiv AB1 5V TTL-Pegel (Grundkarte)
2	2	GW1	BOOL	Grenzwertausgang GW1 (Grundkarte 24 V / 100 mA)
3	3	GW2	BOOL	Grenzwertausgang GW2 (Grundkarte 24 V / 100 mA)
4	4	ABX1	BOOL	Binärausgang ABX1 (I/O-Erweiterung 24 V / 440 mA)
5	5	ABX2	BOOL	Binärausgang ABX2 (I/O-Erweiterung 24 V / 440 mA)
6	6	ABX3	BOOL	Binärausgang ABX3 (I/O-Erweiterung 24 V / 440 mA)

Register der Funktion 02: „Lese Eingangsstatus“ High/Low-Pegel der Binärausgänge
0 = Low-Pegel, 1 = High-Pegel

Registeradressen Dez	Hex	Bezeichnung	Datentyp	Beschreibung
0	0	EB1	BOOL	Binäreingang EB1 (Grundkarte: 5 V bzw. 24 V Signal-Pegel)
1	1	EB2	BOOL	Binäreingang EB2 (Grundkarte: 5 V bzw. 24 V Signal-Pegel)
2	2	EB3	BOOL	Binäreingang EB3 (Grundkarte: 5 V bzw. 24 V Signal-Pegel)
3	3	EB4	BOOL	Binäreingang EB4 (Grundkarte: 5 V bzw. 24 V Signal-Pegel)
4	4	ENI	BOOL	Binäreingang ENI (Grundkarte: NAMUR HF-Eingang)
5	5	EBX1	BOOL	Binäreingang EBX1 (I/O-Erweiterungskarte: 24 V Signal-Pegel galvanisch getrennt)
6	6	EBX2	BOOL	Binäreingang EBX2 (I/O-Erweiterungskarte: 24 V Signal-Pegel galvanisch getrennt)
7	7	EBX3	BOOL	Binäreingang EBX3 (I/O-Erweiterungskarte: 24 V Signal-Pegel galvanisch getrennt)
8	8	NN		

Register der Funktion 01: „Lese Fehlerstatus“ I/O-, Hard- und Firmware-Fehlermeldungen
0 = Low-Pegel, 1 = High-Pegel

Registeradressen Dez	Hex	Bezeichnung	Datentyp	Beschreibung
7	07	ERR_ALL	BOOL	Globaler Fehlerstatus, wird gesetzt, wenn der TZA 401 einen Fehler feststellt
8	08	ERR_E1	BOOL	Analogeingang E1 außerhalb der zulässigen Signalgrenzen
9	09	ERR_E2	BOOL	Analogeingang E2 außerhalb der zulässigen Signalgrenzen
10	0A	ERR_E3	BOOL	Analogeingang E3 außerhalb der zulässigen Signalgrenzen
11	0B	ERR_E4	BOOL	Analogeingang E4 außerhalb der zulässigen Sianalgrenzen
12	0C	ERR_E5	BOOL	Analogeingang E5 außerhalb der zulässigen Signalgrenzen
13	0D	ERR_E6	BOOL	Analogeingang E6 außerhalb der zulässigen Signalgrenzen
14	0E	ERR_EB	BOOL	Binäreingänge EB1, EB4 oder ENI: Signalfrequenz außerhalb der zulässigen Grenzen
15	0F	ERR_A1	BOOL	Analogausgang A1 außerhalb der zulässigen Signalgrenzen
16	10	ERR_A2	BOOL	Analogausgang A2 außerhalb der zulässigen Signalgrenzen
17	11	ERR_AB	BOOL	Binärausgang AB1 außerhalb der zulässigen Grenzen (wird noch nicht unterstützt)
18	12	ERR_EX1	BOOL	Analogeingang EX1 (I/O-Erweiterung) außerhalb der zulässigen Signalgrenzen
19	13	ERR_EX2	BOOL	Analogeingang EX2 (I/O-Erweiterung) außerhalb der zulässigen Signalgrenzen
20	14	ERR_AX1	BOOL	Analogausgang AX1 (I/O-Erweiterung) außerhalb der zulässigen Signalgrenzen
21	15	ERR_AX2	BOOL	Analogausgang AX2 (I/O-Erweiterung) außerhalb der zulässigen Signalgrenzen
22	16	ERR_ABX	BOOL	Kurzschluß bzw. Überlastung von Binärausgang ABX1, ABX2 oder ABX3
23	17	ERR_US1	BOOL	wird noch nicht unterstüzt
24	18	ERR_US2	BOOL	wird noch nicht unterstüzt
25	19	ERR_US3	BOOL	wird noch nicht unterstüzt
26	1A	ERR_US4	BOOL	wird noch nicht unterstüzt
27	1B	ERR_EP1	BOOL	EPROM1-Fehler Summencheckfehler Betriebssystem-EPROM
28	1C	ERR_EP2	BOOL	EPROM2-Fehler Summencheckfehler BASIC-EPROM (wird noch nicht unterstützt)
29	1D	ERR_CNT	BOOL	Redundanzfehler des Hauptzählers
30	1E	NN		
31	1F	NN		
32	20	NN		
33	21	NN		

Register der Funktion 01: „Lese Fehlerstatus“ Physikalische Meßwertgrenzen verletzt!
Rechenprogramme: P731 und P735
0 = OK, 1 = Fehler

Registeradressen Dez	Hex	Bezeichnung	Datentyp	physikalische Meßwertbezeichnungen Programm P731: Durchfluß, Gas trocken/feucht	Programm P735: Wärmeleistung, Gas trocken/feucht
34	22	PE_E1	BOOL	p	p
35	23	PE_E2	BOOL	T	T
36	24	PE_E3	BOOL	Dichte	Hu
37	25	PE_E4	BOOL	Feuchte	Feuchte
38	26	PE_E5	BOOL	dp1, Qv	dp1, Qv
39	27	PE_E6	BOOL	dp2	dp2
40	28	PE_EB	BOOL	-	-
41	29	PE_A1	BOOL	Q	P
42	2A	PE_A2	BOOL	Q, T, Feuchte, p	Q, P, T, Feuchte, p
43	2B	PE_AB	BOOL	-	-
44	2C	PE_EX1	BOOL	T	T
45	2D	PE_EX2	BOOL	-	-
46	2E	PE_AX1	BOOL	Q	Q, P
47	2F	PE_AX2	BOOL	Q, T, Feuchte, p	Q, P, T, Feuchte, p

Register der Funktion 01: „Lese Fehlerstatus“ Physikalische Meßwertgrenzen verletzt!
Rechenprogramme P751, P761 und P765
0 = OK, 1 = Fehler

Registeradressen Dez	Hex	Bezeichnung	Datentyp	physikalische Meßwertbezeichnungen Programm P751: Durchfluß / Wärme-Leistung, Wasser	Programm P761: Durchfluß / Wärme-Leistung, Dampf	Programm P765: Durchfluß / Wärme-Leistung, Dampf minus Wasser
34	22	PE_E1	BOOL	p	p	p
35	23	PE_E2	BOOL	Tw	T	T
36	24	PE_E3	BOOL	Tk	-	Tw
37	25	PE_E4	BOOL	-	-	dpw, Qvw
38	26	PE_E5	BOOL	dp1, Qv	dp1, Qv	dp1, Qvd
39	27	PE_E6	BOOL	dp2	dp2	dp2
40	28	PE_EB	BOOL	-	-	-
41	29	PE_A1	BOOL	Q, P,	Q, P	Qd, Qw, Pd, Pw, dP
42	2A	PE_A2	BOOL	Q, P, Tw, Tk, dT, T	Q, P, T, p	Qd, Qw, Pd, Pw, dP, T, Tw, p
43	2B	PE_AB	BOOL	-	-	-
44	2C	PE_EX1	BOOL	Tw	T	T
45	2D	PE_EX2	BOOL	Tk	-	Tw
46	2E	PE_AX1	BOOL	Q, P, Tw, Tk, dT, T	Q, P, T, p	Qd, Qw, Pd, Pw, dP, T, Tw, p
47	2F	PE_AX2	BOOL	Q, P, Tw, Tk, dT, T	Q, P, T, p	Qd, Qw, Pd, Pw, dP, T, Tw, p

Register der Funktion 03: „Lese Floating-Point-Werte“ physikalische Meßwerte
Rechenprogramme P731 und P 735

Registeradressen Dez	Hex	Bezeichnung	Datentyp	Physikalische Meßgrößen (Dimensionen je nach Parametrierung) Programm P731: Durchfluß, Gas trocken/feucht	Programm P735: Wärmeleistung, Gas trocken/feucht
0	00	Count1	REAL	-	P
2	02	Count2	REAL	Q	Q
4	04	W1	REAL	-	-
6	06	W2	REAL	-	-
8	08	W3	REAL	-	-
10	0A	W4	REAL	-	-
12	0C	W5	REAL	-	-
14	0E	W6	REAL	-	-
16	10	W7	REAL	-	-
18	12	W8	REAL	-	-
20	14	W9	REAL	-	-
22	16	W10	REAL	-	-
24	18	W11	REAL	-	-
26	1A	W12	REAL	-	-
28	1C	W13	REAL	-	-
30	1E	W14	REAL	-	-
32	20	Phy_E1	REAL	p	p
34	22	Phy_E2	REAL	T	T
36	24	Phy_E3	REAL	Norm-Dichte	Hu
38	26	Phy_E4	REAL	Feuchte	Feuchte
40	28	Phy_E5	REAL	dp1, Qv	dp1, Qv
42	2A	Phy_E6	REAL	dp2	dp2
44	2C	Phy_EB	REAL	f(Qv)	f(Qv)
46	2E	Phy_A1	REAL	Q	P, Q
48	30	Phy_A2	REAL	Q, T, Feuchte, p	P, Q, T, Feuchte, p
50	32	Phy_AB	REAL	-	-
52	34	Phy_EX1	REAL	T	T
54	36	Phy_EX2	REAL	-	-
56	38	Phy_AX1	REAL	Q	P, Q
58	3A	Phy_AX2	REAL	Q, T, Feuchte, p	P, Q, T, Feuchte, p
60	3C	Phy_ABX	REAL	-	-

Register der Funktion 03: „Lese Floating-Point-Werte“ Ein-/Ausgangs-Signalmeßwerte
Programme P731 und P735

Registeradressen Dez	Hex	Bezeichnung	Datentyp	Signalmeßgröße in mA, V oder Ω je nach Parametrierung Programm P731: Durchfluß, Gas trocken/feucht	Programm P735: Wärmeleistung, Gas trocken/feucht
62	3E	Sig_E1	REAL	p	p
64	40	Sig_E2	REAL	T	T
66	42	Sig_E3	REAL	Norm-Dichte	Hu
68	44	Sig_E4	REAL	Feuchte	Feuchte
70	46	Sig_E5	REAL	dp1, Qv	dp1, Qv
72	48	Sig_E6	REAL	dp2	dp2
74	4A	Sig_EB	REAL	f(Qv)	f(Qv)
76	4C	Sig_A1	REAL	Q	P, Q
78	4E	Sig_A2	REAL	Q, T, Feuchte, p	P, Q, T, Feuchte, p
80	50	Sig_AB	REAL	-	-
82	52	Sig_EX1	REAL	T	T
84	54	Sig_EX2	REAL	-	-
86	56	Sig_AX1	REAL	Q	P, Q
88	58	Sig_AX2	REAL	Q, T, Feuchte, p	P, Q, T, Feuchte, p
90	5A	Sig_ABX	REAL		
92	5C	NN			
94	5E	NN			
96	60	NN			

Register der Funktion 03: „Lese Floating-Point-Werte“ physikalische Meßwerte
Programme P751, P761 und P765

Registeradressen Dez	Hex	Bezeichnung	Datentyp	Physikalische Meßgrößen (Dimensionen je nach Parametrierung) Programm P751: Durchfluß / Wärme-Leistung, Wasser	Programm P761: Durchfluß / Wärme-Leistung, Dampf	Programm P765 Durchfluß / Wärme-Leistung, Dampf minus Wasser
0	00	Count1	REAL	P	P	P
2	02	Count2	REAL	Q	Q	Q
4	04	W1	REAL	H	H	Rho_d
6	06	W2	REAL	Rho	Rho	Rho_w
8	08	W3	REAL	Hw	-	Hd
10	0A	W4	REAL	Hk	-	Hw
12	0C	W5	REAL	-	-	-
14	0E	W6	REAL	-	-	-
16	10	W7	REAL	-	-	-
18	12	W8	REAL	-	-	-
20	14	W9	REAL	-	-	-
22	16	W10	REAL	-	-	-
24	18	W11	REAL	-	-	-
26	1A	W12	REAL	-	-	-
28	1C	W13	REAL	-	-	-
30	1E	W14	REAL	-	-	-
32	20	Phy_E1	REAL	p	p	p
34	22	Phy_E2	REAL	Tw, T	T	T
36	24	Phy_E3	REAL	Tk	-	Tw
38	26	Phy_E4	REAL	-	-	dpw; Qvw
40	28	Phy_E5	REAL	dp1, Qv	dp1, Qv	dp1, Qvd
42	2A	Phy_E6	REAL	dp2	dp2	dp2
44	2C	Phy_EB	REAL	f(Qv)	f(Qv)	f(Qv)
46	2E	Phy_A1	REAL	P, Q	P, Q	Pd, Pw, dP, Qd, Qw
48	30	Phy_A2	REAL	P, Q, , Tw, Tk, dT, T	P, Q, T, p	Pd, Pw, dP, Qd, Qw, T, Tw, p
50	32	Phy_AB	REAL	-	-	-
52	34	Phy_EX1	REAL	Tw	T	T
54	36	Phy_EX2	REAL	Tk	-	Tw
56	38	Phy_AX1	REAL	P, Q, , Tw, Tk, dT, T	P, Q, T, p	Pd, Pw, dP, Qd, Qw, T, Tw, p
58	3A	Phy_AX2	REAL	P, Q, , Tw, Tk, dT, T	P, Q, T, p	Pd, Pw, dP, Qd, Qw, T, Tw, p
60	3C	Phy_ABX	REAL	-	-	-

Register der Funktion 03: „Lese Floating-Point-Werte“ Ein-/Ausgangs-Signal-Meßwerte
Programme P751, P761 und P765

Registeradressen Dez	Hex	Bezeichnung	Datentyp	Signal-Meßgrößen in mA, V oder Ω je nach Parametrierung Programm P751: Durchfluß Wärme-Leistung-Wasser	Programm P761: Durchfluß-Wärme-Leistung, Dampf	Programm P765: Durchfluß / Wärmeleistung, Dampf minus Wasser
62	3E	Sig_E1	REAL	p	p	p
64	40	Sig_E2	REAL	Tw	T	T
66	42	Sig_E3	REAL	Tk	-	Tw
68	44	Sig_E4	REAL	-	-	dpw; Qvw
70	46	Sig_E5	REAL	dp1, Qv	dp1, Qv	dp1, Qvd
72	48	Sig_E6	REAL	dp2	dp2	dp2
74	4A	Sig_EB	REAL	f(Qv)	f(Qv)	f(Qv)
76	4C	Sig_A1	REAL	P, Q	P, Q	Pd, Pw, dP, Qd, Qw
78	4E	Sig_A2	REAL	P, Q, ,Tw, Tk, dT, T	P, Q, T, p	Pd, Pw, dP, Qd, Qw, T, Tw, p
80	50	Sig_AB	REAL	-	-	-
82	52	Sig_EX1	REAL	Tw	T	T
84	54	Sig_EX2	REAL	Tk	-	Tw
86	56	Sig_AX1	REAL	P, Q, , Tw, Tk, dT, T	P, Q, T, p	Pd, Pw, dP, Qd, Qw, T, Tw, p
88	58	Sig_AX2	REAL	P, Q, , Tw, Tk, dT, T	P, Q, T, p	Pd, Pw, dP, Qd, Qw, T, Tw, p
90	5A	Sig_ABX	REAL			
92	5C	NN				
94	5E	NN				
96	60	NN				

Register der Funktion 16 (10H): „Schreibe Floating-Point-Konstanten“					
Registeradressen		Bezeichnung	Datentyp	TZA 401 - RAM	Konstantenbezeichnungen sind BASIC-Programmabhängig und können frei definiert werden
Dez	Hex			Speicher-Adresse	
100	0064H	Konst01	REAL	6400H	
102	0066H	Konst02	REAL	6406H	
104	0068H	Konst03	REAL	640CH	
106	006AH	Konst04	REAL	6412H	
108	006CH	Konst05	REAL	6418H	
110	006EH	Konst06	REAL	641EH	
112	0070H	Konst07	REAL	6424H	
114	0072H	Konst08	REAL	642AH	
116	0074H	Konst09	REAL	6430H	
118	0076H	Konst10	REAL	6436H	
120	0078H	Konst11	REAL	643CH	
122	007AH	Konst12	REAL	6442H	
124	007CH	Konst13	REAL	6448H	
126	007EH	Konst14	REAL	644EH	
128	0080H	Konst15	REAL	6454H	
130	0082H	Konst16	REAL	645AH	
132	0084H	Konst17	REAL	6460H	
134	0086H	Konst18	REAL	6466H	
136	0088H	Konst19	REAL	646CH	
138	008AH	Konst20	REAL	6472H	
140	008CH	Konst21	REAL	6478H	
142	008EH	Konst22	REAL	647EH	
144	0090H	Konst23	REAL	6484H	
146	0092H	Konst24	REAL	648AH	
148	0094H	Konst25	REAL	6490H	
150	0096H	Konst26	REAL	6496H	
152	0098H	Konst27	REAL	649CH	
154	009AH	Konst28	REAL	64A2H	
156	009CH	Konst29	REAL	64A8H	
158	009EH	Konst30	REAL	64AEH	
160	00A0H	Konst31	REAL	64B4H	
162	00A2H	Konst32	REAL	64BAH	
164	00A4H	Konst33	REAL	64B0H	
166	00A6H	Konst34	REAL	64C6H	
168	00A8H	Konst35	REAL	64CCH	
170	00AAH	Konst36	REAL	64D2H	
172	00ACH	Konst37	REAL	64D8H	
174	00AEH	Konst38	REAL	64DEH	
176	00B0H	Konst39	REAL	64E4H	
178	00B2H	Konst40	REAL	64EAH	
180	00B4H	Konst41	REAL	64F0H	
182	00B6H	Konst42	REAL	64F6H	
184	00B8H	Konst43	REAL	64FCH	
186	00BAH	Konst44	REAL	6502H	
188	00BCH	Konst45	REAL	6508H	
190	00BEH	Konst46	REAL	650EH	
192	00C0H	Konst47	REAL	6514H	
194	00C2H	Konst48	REAL	651AH	
196	00C4H	Konst49	REAL	6520H	
198	00C6H	Konst50	REAL	6526H	
200	00C8H	Konst51	REAL	652CH	
202	00CAH	Konst52	REAL	6532H	
204	00CCH	Konst53	REAL	6538H	
206	00CEH	Konst54	REAL	653EH	
208	00D0H	Konst55	REAL	6544H	
210	00D2H	Konst56	REAL	654AH	
212	00D4H	Konst57	REAL	6550H	
214	00D6H	Konst58	REAL	6556H	
216	00D8H	Konst59	REAL	655CH	
218	00DAH	Konst60	REAL	6562H	
220	00DCH	Konst61	REAL	6568H	
222	00DEH	Konst62	REAL	656EH	
224	00D0H	Konst63	REAL	6574H	
226	00D2H	Konst64	REAL	657AH	
228	00E4H	Konst65	REAL	6580H	
230	00E6H	Konst66	REAL	6486H	
232	00E8H	Konst67	REAL	648CH	
234	00EAH	Konst68	REAL	6592H	
236	00ECH	Konst69	REAL	6598H	
238	00EEH	Konst70	REAL	659EH	
240	00F0H	Konst71	REAL	65A4H	

Programmierbeispiele in C

Alle nachfolgenden Beispiele für die Zugriffe auf Register sind in C, um ein exaktes und fehlerfreies Beispiel zu gewährleisten. Die Übertragung findet hier im RTU-Protokoll statt. Die Funktionen *modbus_read()* und *modbus_write()* zeigen, wie ein Telgramm aufgebaut wird, alle anderen Funktionen zeigen den Umgang mit den verschiedenen Datenformaten.

Modbus_read

Daten von anderen Modbusteilnehmern lesen (Read-Output-Register: Funktion 03)

```
void modbus_read(unsigned regnr, int anzahl, int *recdata)
{
    int           i,anz;
    unsigned      crc;

    sendbuf[0] = mod_adr;   /* Modbus Zieladresse   */
    sendbuf[1] = 3;         /* Funktion „Read  Real-Wert aus TZA 401“ */
    sendbuf[2] = regnr>>8;  /* Hi Register Adresse  */
    sendbuf[3] = regnr;     /* Lo Register Adresse  */
    sendbuf[4] = 0;         /* Hi Anzahl Register (beim TZA 401 immer 0)  */
    sendbuf[5] = 2          /* Lo Anzahl Register  beim TZA 401 immer 2)  */
    crc        = CRC16(sendbuf,6);
    sendbuf[6] = crc;
    sendbuf[7] = crc>>8;

    ComWrite(sendbuf,8);    /* 8 Zeichen senden */
    ComRead(receivebuf);    /* Daten Empfangen */

    // receivebuf[0];   enth. die Modbus Zieladresse
    // receivebuf[1];   enth. den Functions-Code

    anz = receivebuf[2];    /*  Anzahl Datenbytes (beim TZA 401 immer 4) */

    // receivebuf[3+anz];   enth.AElt Adresse CRC
    // receivebuf[4+anz];   enth.AElt Adresse CRC

    for (i=0; i < anz; i+=2) {
        recdata[i+0] = receivebuf[4+i];
        recdata[i+1] = receivebuf[3+i];
    }
}
```

Modbus_write

REAL-Daten zu anderen Modbusteilnehmern senden (write Konstanten in TZA 401: Funktion 16)

```
void modbus_write(unsigned regnr, int data[0], int data[1])
{
   unsigned            crc;

sendbuf[00] = Stat_adr       ;     /* Modbus Stationsadresse  */
sendbuf[01] = 16             ;     /* Funktion write Real-Wert */
sendbuf[02] = regnr>>8             ;     /* Hi Register Nummer  */
sendbuf[03] = regnr          ;     /* Lo Register Nummer */
sendbuf[04] = 00             ;     /* Hi Register Anzahl (bei TZA 401 immer 0) */
sendbuf[05] = 02             ;     /* Lo Register Anzahl (bei TZA 401 immer 2) */
sendbuf[06] = 04             ;     /* Anzahl Byte   (bei TZA 401 immer 4)  */
sendbuf[07] = data[0]>>8     ;     /* Hi Byte  Register 0 laden */
sendbuf[08] = data[0]        ;     /* Lo Byte Register 0 laden */
sendbuf[09] = data[1]>>8     ;     /* Hi  Byte Register 1 laden */
sendbuf[10] = data[1]        ;     /* Lo Byte Register  1 laden */
crc         = CRC16(sendbuf,11);         /* CRC-Check für 11 Byte generieren */
sendbuf[11] = crc            ;     /* Lo Byte CRC laden */
sendbuf[12] = crc>>8         ;     /* Hi Byte  CRC laden */
comWrite(sendbuf,13)         ;     /* 13 Zeichen senden */
comRead(receivebuf)          ;     /* Quittung empfangen */

}
```

Programmierbeispiel zum Ermitteln der CRC-Summe des Modbus-RTU Telegramms

```
unsigned short CRC16(
    void *data_p    /* Datenbereich */,
    unsigned len    /* Datenlänge*/
)
/* 16 Bit CRC (Modbus-RTU) von data_p berechnen */
{

#   define POLYNOM            0x0A001

    int             i,j;
    unsigned short  crc = 0xffff;
    unsigned char   *p = data_p;

    for (j=0; j < len; j++) {                  /* für gesamten Puffer */
        for (crc ^=  *p++,i=0;  i < 8; i++) {  /* für ein Byte */
            if ((crc & 0x0001))
                crc = (crc >> 1) ^ POLYNOM;
            else
                crc >>= 1;
        }
    }
    return (crc);
}
```

Zählerstand „Wärmeleistung P“ (Programm P761) mit Pair of Register ermitteln

(W, Register 0..1)

```
void read_float_split_merge()
{
    float   *fval;
    int     recdata[30];

    modbus_read(0, 2, &recdata[0]);
    fval = (void *)&recdata[0];
    printf("Float-Register 0/1 :  float =%6.3f", *fval);
}
```

```
 Signalmeßwert dp1-Eingang (Programm P761) mit Pair of Register ermitteln
(dp1, Register 74..75)

void read_float_split_merge()
{
    float   *fval;
    int     recdata[30];

    modbus_read(74, 2, &recdata[0]);
    fval = (void *)&recdata[0];
    printf("Float-Register 74/75 :  float =%6.3f", *fval);
}
```

REAL-Kostante1 in Register-Adresse 100 einschreiben

(Register 100/101)

```
void write_float_split_merge()
{
    float             wert;
    unsigned long    *pdata;
    int               data[2];

    wert    = 123.4567;
    pdata   = (void *)&wert;
    data[0] = (unsigned) (*pdata & 0xFFFF);
    data[1] = (unsigned) (*pdata >>16);
    modbus_write(100,data[0],data[1]);
}
```

Programmierbeispiele in Quickbasic 4.5

IEEE-Werteberechnung mittels MKS$ und CSV-Funktion

```
’Demoprogramm zur Bearbeitung von IEEE-Werte-Darstellung
’in Quick-Basic 4.5
’benutzt die Quick-Basic-Funktionen MKS$ und CVS
’-------------------------------------------------------
DECLARE FUNCTION BIN.AER$ (z$)
DECLARE FUNCTION HEX2$ (x)
CLS
DO
  INPUT "Realwert (E = Ende) "; Realwert$
  IF UCASE$(Realwert$) = "E" THEN END
  Realwert! = VAL(Realwert$)
’-------------------------------------------------------
’Aufbereiten:
’-------------------------------------------------------
’ Realwert in IEEE-Darstellung
  IEEE$ = MKS$(Realwert!)             ’4 Byte-String
  FOR I = 0 TO 3
    Byte(I) = ASC(MID$(IEEE$, I + 1, 1))
  NEXT
  Date0& = Byte(1) * 256 + Byte(0)
  Date1& = Byte(3) * 256 + Byte(2)
  ’Diese 2 Worte muessen richtig in das Sendetelegramm
  ’eingebaut werden.
’-------------------------------------------------------
’Kontroll-Darstellungen
  IEEE$ = HEX2$(Byte(3)) + HEX2$(Byte(2))
  IEEE$ = IEEE$ + HEX2$(Byte(1)) + HEX2$(Byte(0))
  PRINT IEEE$; " ="; BIN.AER$(IEEE$)
’=======================================================
’Zurückrechnen
’-------------------------------------------------------
’es sind empfangen worden die Bytes(0) bis Byte(3)
’-------------------------------------------------------
  IEEEHEX$ = ""
  FOR I = 0 TO 3
    IEEEHEX$ = IEEEHEX$ + CHR$(Byte(I))
  NEXT
  Realwert! = CVS(IEEEHEX$)
  PRINT "Rueckrechnung = "; Realwert!
LOOP
’---------------------------------------
’Umwandlung einer Hex-Ziffer in Binärdarstellung
’-------------------------------------------------
FUNCTION BIN.AER$ (z$)
DEFINT A-Z
FOR I = 1 TO LEN(z$)
  x1$ = ""
  x% = VAL("&H" + MID$(z$, I, 1))
  DO UNTIL x% = 0
    Y$ = LTRIM$(STR$(x% MOD 2))
    x% = x% \ 2
    x1$ = Y$ + x1$
  LOOP
  x1$ = RIGHT$("0000" + x1$, 4)
  x$ = x$ + " " + x1$
NEXT
BIN.AER$ = x$
END FUNCTION
’---------------------------------------
DEFSNG A-Z
’Stellt Hex-Ziffern zweistellig dar
’---------------------------------------
FUNCTION HEX2$ (x)
  HEX2$ = RIGHT$("00" + HEX$(x), 2)
END FUNCTION
```

IEEE- Werteberechnung ohne spezielle Funktionen

```
'Demoprogramm zur Bearbeitung von IEEE-Werte-Darstellung
'in Quick-Basic 4.5                          Version 1.0
'-----------------------------------------------------
DECLARE FUNCTION BIN.AER$ (z$)
CLS
DO UNTIL i = 127
  INPUT "Realwert (e = ende) "; RealWert$
  IF UCASE$(RealWert$) = "E" THEN END
  RealWert! = VAL(RealWert$)
'-----------------------------------------------------------
'Aufbereitung:
'===========================================================
'Vorzeichen separieren
  Vorzeichen = 0
  IF RealWert! < 0 THEN
    RealWert! = RealWert! * (-1)
    Vorzeichen = -1
  END IF
'-----------------------------------------------------------
'Exponenten bestimmen
  Exponent% = 0
  X! = RealWert!
  IF X! > 1 THEN
    DO UNTIL X! < 1
      X! = X! / 2
      Exponent% = Exponent% + 1
    LOOP
    Exponent% = Exponent% - 1
  ELSE
    DO UNTIL X! > 1
      X! = X! * 2
      Exponent% = Exponent% - 1
    LOOP
    PRINT Exponent%

  END IF
'-----------------------------------------------------------
'Mantisse bestimmen
  Mantisse = RealWert! * (2 ^ (23 - Exponent%))
  Mantisse = Mantisse AND &H7FFFFF
'-----------------------------------------------------------
'Worte und Bytes für Telegramm bestimmen
  Exponent% = (Exponent% + &H7F) * 128
  Date0 = Mantisse MOD &H10000
  Date1 = Mantisse \ &H10000 + Exponent%
  Byte(0) = Date0 MOD 256
  Byte(1) = Date0 \ 256
  Byte(2) = Date1 MOD 256
  Byte(3) = Date1 \ 256 + ((-1) * Vorzeichen) * &H80
'-----------------------------------------------------
'Kontrolldarstellung
  PRINT "IEEE-Wert: ";
  FOR i = 3 TO 0 STEP -1
    PRINT BIN.AER$(HEX$(Byte(i)));
  NEXT
  PRINT
'=====================================================
'Zurückrechnen
'-----------------------------------------------------
'es sind empfangen worden die Bytes(0) bis Byte(3)
'-----------------------------------------------------
'Vorzeichn ist in Bit 7 von Byte(3) kodiert
Vorzeichen = 1
IF (Byte(3) AND &H80) = &H80 THEN Vorzeichen = -1
'-----------------------------------------------------------
'Exponent ermitteln aus Bit 6 bis 0 aus Byte(3)
'und Bit 8 aus Byte(2)
Exponent = (Byte(3) AND &H7F) * 2 + (Byte(2) \ 128)
'-----------------------------------------------------
'Mantisse ermitteln:
'Bit 7 von Byte(3) setzen,
```

```
’Mantisse aus Byte(0) bis Byte(3) errechnen
Mantisse = (Byte(2) OR &H80) * &H10000
Mantisse = Mantisse + Byte(1) * &H100 + Byte(0)
’-----------------------------------------------------
RealWert! = Vorzeichen * Mantisse / (2 ^ (23 - (Exponent - &H7F)))
PRINT "Rückrechnung = "; RealWert!
LOOP
’-------------------------------------------------
’Umwandlung einer Hex-Ziffer in Binärdarstellung
’-------------------------------------------------
FUNCTION BIN.AER$ (z$)
DEFINT A-Z
FOR i = 1 TO LEN(z$)
  x1$ = ""
  X% = VAL("&H" + MID$(z$, i, 1))
  DO UNTIL X% = 0
    Y$ = LTRIM$(STR$(X% MOD 2))
    X% = X% \ 2
    x1$ = Y$ + x1$
  LOOP
  x1$ = RIGHT$("0000" + x1$, 4)
  X$ = X$ + " " + x1$
NEXT
BIN.AER$ = X$
END FUNCTION
```

Berechnete Beispiele

Exponent auf Basis 2 wird durch mehrfache Multiplikation mit 2 oder Division durch 2 so berechnet, daß sich ein 24-stelliger binärer Wert mit einer 1 an der höchsten (linken) Stelle ergibt. In der IEEE-Darstellung wird diese 1 unterdrückt.

```
dezim. hexadezimal binär
                  s/Exponent /Wert                       /
-1,0  BF 80 00 00 1011 1111 1000 0000 0000 0000 0000 0000
-0,5  BF 00 00 00 1011 1111 0000 0000 0000 0000 0000 0000
-0,4  BE CC CC CD 1011 1110 1100 1100 1100 1100 1100 1101
-0,3  BE 99 99 9A 1011 1110 1001 1001 1001 1001 1001 1010
-0,2  BE 4C CC CD 1011 1110 0100 1100 1100 1100 1100 1101
-0,1  BD CC CC CD 1011 1101 1100 1100 1100 1100 1100 1101
 0,0  00 00 00 00 0000 0000 0000 0000 0000 0000 0000 0000
 0,1  3D CC CC CD 0011 1101 1100 1100 1100 1100 1100 1101
 0,2  3E 4C CC CD 0011 1110 0100 1100 1100 1100 1100 1101
 0,3  3E 99 99 9A 0011 1110 1001 1001 1001 1001 1001 1010
 0,4  3E CC CC CD 0011 1110 1100 1100 1100 1100 1100 1101
 0,5  3F 00 00 00 0011 1111 0000 0000 0000 0000 0000 0000
 1,0  3F 80 00 00 0011 1111 1000 0000 0000 0000 0000 0000
10,0  41 20 00 00 0100 0001 0010 0000 0000 0000 0000 0000
```

Berechnung der Prüfsumme CRC

```
’Basic-Programm zur ermittlung der Prüfsumme CRC für Modbus-RTU Telegramme
’Quickbasic 4.5                                  Version 1.0
’-----------------------------------------------------------
DECLARE FUNCTION HEX2$ (x!)
CLS
MaxI = 2
PRINT "Eingabe der Telegrammbytes in Hex 05H oder dezimal 5"
PRINT "getrennt einheitlich durch Leerzeichen oder Kommata"
DO
INPUT Tel$
i = 1
L = LEN(Tel$)
Tel$ = UCASE$(Tel$)
  DO UNTIL Tel$ = ""
    Tel$ = LTRIM$(Tel$)
```

```
    x = INSTR(Tel$, " ") + INSTR(Tel$, ",")
    IF x > 4 THEN Fehler = 1: EXIT DO
    IF x > 0 THEN
      Byte$(i) = LEFT$(Tel$, x)
      Tel$ = RIGHT$(Tel$, LEN(Tel$) - x + 1)
    ELSE
      Byte$(i) = Tel$
      Tel$ = ""
    END IF
    Byte$(i) = RTRIM$(Byte$(i))
    IF RIGHT$(Byte$(i), 1) <> "H" THEN
      Byte$(i) = HEX2$(VAL(Byte$(i)))
    ELSE
      Byte$(i) = LEFT$(Byte$(i), 2)
    END IF
    IF HEX2$(VAL("&H" + Byte$(i))) <> Byte$(i) THEN Fehler = 1: EXIT DO
    i = i + 1
  LOOP
  IF Fehler = 0 THEN EXIT DO
  SOUND 1000, .03
LOOP
MaxI = i - 1
x& = 65535
FOR i = 1 TO MaxI
  y& = (VAL("&H" + Byte$(i)) XOR x&)
  n = 1
  DO
    DO
      r = y& MOD 2
      y& = y& - r
      y& = y& / 2
      IF ABS(r) = 1 THEN EXIT DO
      n = n + 1
      IF (n = 9) AND (i = MaxI) THEN EXIT FOR
      IF n = 9 THEN EXIT DO
    LOOP
    IF n < 9 THEN
      y& = y& XOR 40961
      n = n + 1
    END IF
    IF n = 9 THEN
      IF (i = MaxI) THEN EXIT FOR
      EXIT DO
    END IF
  LOOP
  x& = y&
NEXT
PRINT "CRC ="; HEX$(y&); " Hex"
PRINT " muss in der Reihenfolge "; HEX2$(y& MOD 256); " "; HEX2$(y& \ 256);
PRINT " in das Telegramm aufgenommen werden !"
FUNCTION HEX2$ (x)
HEX2$ = RIGHT$("00" + HEX$(x), 2)
END FUNCTION
```

Technische Änderungen vorbehalten.

ABB Automation Products GmbH
Borsigstraße 2
D-63755 Alzenau
Tel. (0 60 23) 92 - 0
Fax (0 60 23) 92 - 33 00
http://www.abb.de/durchfluss

Technische Änderungen vorbehalten
Printed in the Fed. R. of Germany
42/18-58 DE Rev. 1.0
Ausgabe 04.01