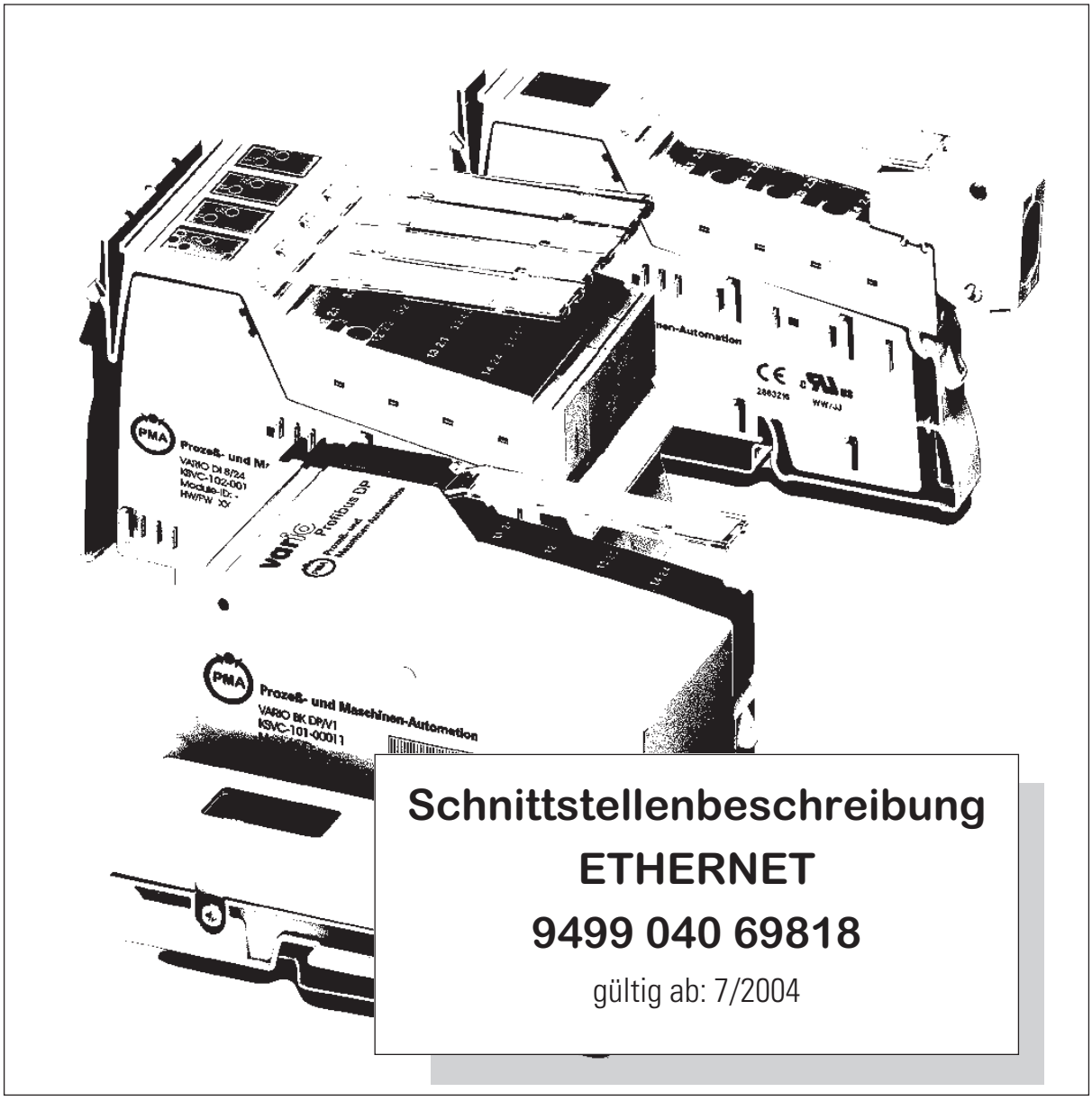
PMA Prozeß- und Maschinen-Automation GmbH

Modulares Reglersystem
KS vario



Schnittstellenbeschreibung
ETHERNET
9499 040 69818

gültig ab: 7/2004

BlueControl ® ist ein eingetragenes Warenzeichen der PMA Prozeß- und Maschinen-Automation GmbH

© PMA Prozeß- und Maschinen-Automation GmbH -
Printed in Germany
Alle Rechte vorbehalten.
Ohne vorhergehende schriftliche Genehmigung ist der Nachdruck oder die auszugsweise fotomechanische oder anderweitige Wiedergabe diese Dokumentes nicht gestattet.

Dies ist eine Publikation von PMA Prozeß- und Maschinen Automation
Postfach 310229
D-34058 Kassel
Germany

Inhalt

1. Allgemeines

Das modulare Reglersystem KS vario erlaubt den Anschluss von verschiedenen Feldbusschnittstellen. Hierzu wird der jeweilige Buskoppler als Kopfstation für ein Reglersystem benutzt.
Über einen dieser Buskoppler wird über eine frontseitige Schnittstelle (PJ45-Stecker) der ETHERNET (Modbus/TCP Protokoll) unterstützt. Hierüber wird eine Übertragung aller Prozeß-, Parameter- und Konfigurationsdaten ermöglicht. Diese Kommunikationsschnittstelle ermöglicht Verbindungen zu übergeordneten Steuerungen, Visualisierungstools etc..
Eine weitere, standardmäßig immer vorhandene Schnittstelle befindet sich auf den Reglerbausteinen KS vario. Diese vollwertige RS232 Schnittstelle dient dem Anschluß des 'BlueControl'-Tools, das auf einem PC abläuft.

Ubertragungs-rate

Der Ethernet-Koppler arbeitet als ModbusTCP-Server mit einer maximalen Übertragungsrate von 100Mbit.

Clients

Der Ethernet Buskoppler ermöglicht die Kommunikation mit bis zu 4 Clients über das TCP/IP-Protokoll.

2. Hinweise zum Betrieb

2.1. Anschluss der Schnittstelle

Der Ethernet wird an die frontseitige RJ45-Schnittstelle des Buskopplers angeschlossen.
Als Physical Layer wird 10BaseT bzw. 100BaseT verwendet.

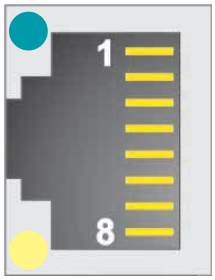
Die physikalische Anbindung erfolgt über Ethernet mit verdrillter Zweidrahtleitung (CAT5-Kabel, 8pol mit RJ-45 Verbindungstechnik).

Belegung RJ-45

Der Anschluss erfolgt über eine RJ-45-Buchse, mit 2 integrierten LED´s.

Grüne LED an: Ethernet angeschlossen
Gelbe LED an: Traffic auf Ethernet

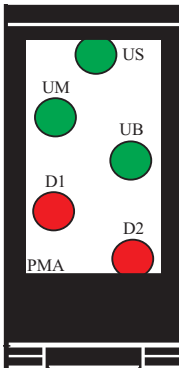
Kontakt	Signal	Beschreibung
1	TD +	Transmit +
2	TD -	Transmit -
3	RD +	Receive +
4	-	Nicht belegt
5	-	Nicht belegt
6	RD -	Receive -
7	-	Nicht belegt
8	-	Nicht belegt



2.2. Bedeutung der Anzeige-LEDs am Buskoppler

LEDs

LED-Nr.	LED-Farbe	Funktion
US	grün	Segmentspannung 24V vorhanden
UM	grün	Haupteinspeisung 24V vorhanden (z.Zt. nicht benuzt
UB	grün	Kopplerspannung 24V$_{\text{ss}}$ vorhanden
D1	rot	AN: Es besteht keine Verbindung zum Client
D2	rot	BLINKT: Kommunikation fehlerhaft AUS: Kommunikation fehlerfrei



2.3. Forcing

Eingänge

Alle physikalischen Eingänge können über ETHERNET überschrieben werden (konfigurierbar). Damit ist es z.B. möglich, den Istwert über Remote I/O (z.B. vario I/O-System) zu erfassen und über den Bus vorzugeben.

Ausgänge

Bei Forcing der Ausgänge, ist die Einstellung der Fail-safe Funktion zu beachten. Bei eingestelltem Fail-safe - Verhalten "zero" werden alle Ausgänge bei Busfehler oder Master-Stop auf null gesetzt, andernfalls behalten sie ihren alten Wert bei.

2.4. Fail-safe

Über die User-Parametrierung 'Fail-safe' wird das Verhalten des Gerätes bei Busausfall bzw. 'Bus-Stop' des Masters festgelegt.
Busausfall
Bei Busausfall arbeitet das Gerät nach folgenden Regeln.

Fail-safe	Reaktion bei Busausfall oder Master-Stop
Last value	Weiterarbeiten mit den zuletzt gesendeten Werten
	Geforcte analoge Eingänge werden auf FAIL gesetzt
zero	Geforcte analoge Eingänge werden auf FAIL 1) gesetzt
	geforcte digitale Eingänge werden auf null gesetzt
	Geforcte Ausgänge werden auf null gesetzt

3. Kommunikation über Ethernet

3.1. Physical Layer

Als Physical Layer wird 10BaseT bzw. 100BaseT verwendet.

3.2. Data Link Layer Ethernet / MAC-ID

Ethernet transportiert Ethernet Packets von einem Sender zu einem oder mehreren Empfängern ohne Quittung und ohne Wiederholung von verlorenen Packets.
Sender und Empfänger von Ethernet Packets werden über die MAC-ID adressiert. Die MAC-ID ist eine 6 Byte großer Ident Code, der eindeutig, d.h. für jedes Ethernet Gerät weltweit unterschiedlich ist. Die MAC-ID besteht aus zwei Teilen. Der erste Teil, d.h. die ersten 3 Byte ist eine Herstellerkennung. Die Firma PMA GmbH hat die Kennung 00 0E 0D. Die nächsten 3 Byte werden durch den Hersteller vergeben und entsprechen einer Seriennummer, sie sind eindeutig. Die MAC-ID kann zum Beispiel für das BOOTP Protokoll zum Einstellen der TCP/IP Nummer verwendet werden. Dafür wird ein Telegramm zum entsprechenden Knoten geschickt, der die Informationen wie Name oder TCP/IP Nummer beinhaltet.

3.3. Network Layer IP

Die Grundlage der Datenkommunikation ist das Internet Protokoll (IP). IP transportiert Datagramme von einem Teilnehmer zu einem anderen Teilnehmer im gleichen oder in einem anderen Netz und kümmerst sich dabei um das Adress-Management (Finden und Zuordnen der MAC-Ids), die Segmentierung und die Pfadsuche (Routing).

3.4. Transport Layer

Das auf IP aufsetzende Transmission Control Protocol TCP ist ein verbindungs-orientiertes Transport-Protokoll. Es umfasst Fehlererkennungs- und behandlungsmechanismen. Verlorengegangene Telegramme werden wiederholt.

UDP ist ein verbindungsloses Transport-Protokoll. Es gibt keine Kontrollmechanismen beim Datenaustausch zwischen Sender und Empfänger.
Dadurch resultiert eine schnellere Verarbeitungsgeschwindigkeit als zum Beispiel beim TCP. Eine Prüfung; ob das Telegramm angekommen ist; muss vom übergeordneten Protokoll durchgeführt werden.

3.5. Application Layer Modbus/TCP

Modbus/TCP ist eine Modbus Verbindung basierend auf den TCP/IP-Übertragungsprotokollen. Als Übertragungsstandard wird Ethernet genutzt. ModbusTCP folgt dem Client-Server-Model, wobei der hier vorliegende ModbusTCP-Server Dienste für Clients zur Verfügung stellt. Die Kommunikation wird durch einen 'Request' eines ModbusTCP-Client ausgelöst. Der Server beantwortet diesen 'Request' mit einer 'Indication'. Ist die Verarbeitung im Client beendet, schickt der Server eine 'Response' an den Client, die mit einer 'Confirmation' beantwortet wird.
Eine Kommunikation über Bridges, Router oder Gateways ist möglich.
Im Ethernet Buskoppler wird das Modbusprotokoll im RTU (Remote Terminal Unit)- Mode genutzt, d. h. jedes gesendete Nachrichtenbyte enthält zwei hexadezimale Zeichen (0..9, A..F).
Weitere Informationen sind im "Modicon Modbus Protocol Reference Guide" der Modicon, Inc., Industrial Automation Systems nachzulesen. Siehe auch Kapitel 5: "Informationen zum Modbus-Protokoll"

3.6. IP Adresse des KS vario Systems über BootP Protokoll

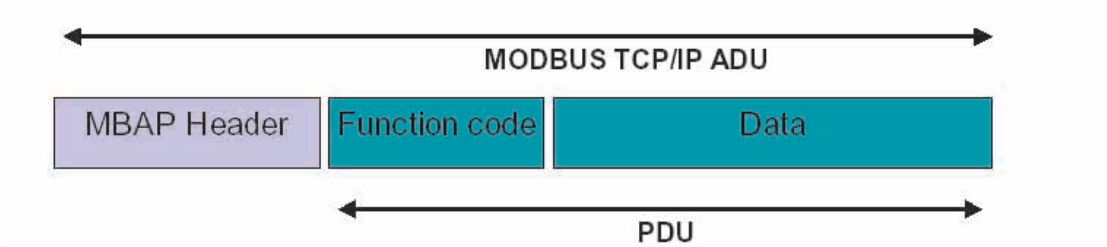
Es besteht die Möglichkeit, die IP-Adresse und Sub-Net-Maske mit dem BootP Protokoll anzufordern. Nach dem Spannungseinschalten wird das BootP Protokoll immer ausgesendet, wenn keine eigene IP-Adr. bekannt ist. Bekannt heißt, dass diese über Engineering Tool "BlueControl" vorgegeben wurde .

3.7. IP Adresse des KS vario Systemsüber Engineering Tool "BlueControl"

Die IP-Adesse (4 Byte) undSub-Net-Maske (4 Byte) kann durch das BlueControl-Tool in den KS-vario eingetragen werden, der diese Information an den Buskoppler während der Initialisierung überträgt. Wird die IP-Adresse über Tool als "0" vorgegeben, so erkennt der Buskoppler dies als ungültige Adresse und die Adressvorgabe über Boot P wird relevant.

3.8. Modbus TCP Meldungsformat (Application Data Unit)

Der Aufbau der ADU (Application Data Unit) ist in der folgenden Abbildung dargestellt:

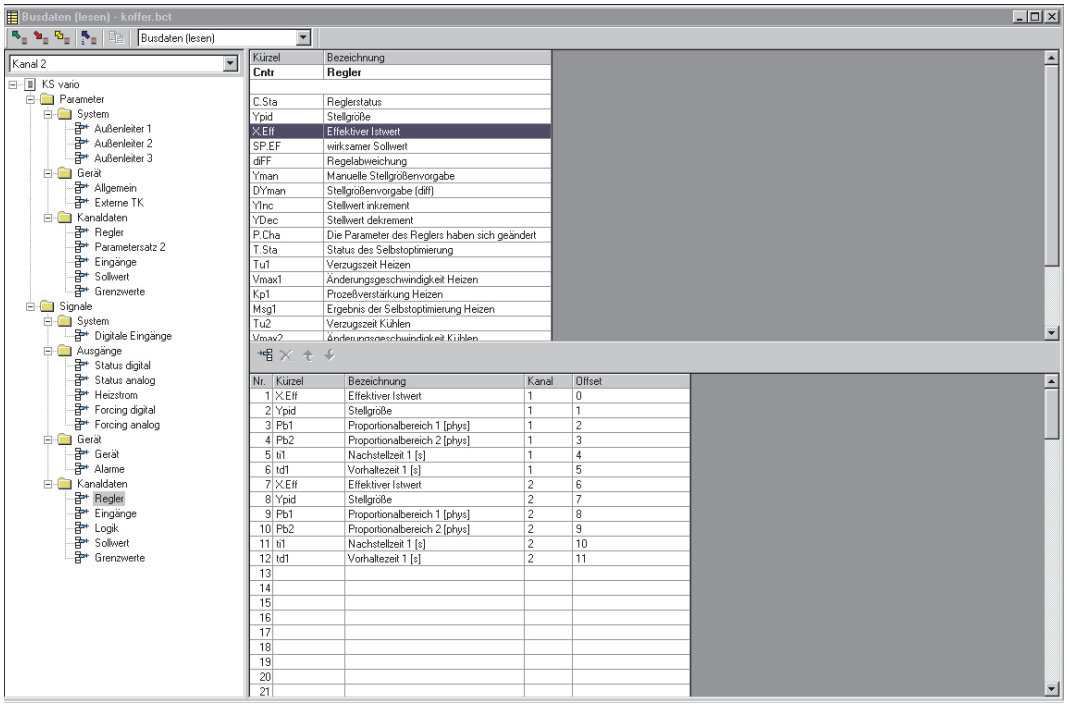


Aufbau der ADU

Fields	Length	Description -	Client	Server
Transaction Identifier	2 Bytes	Identification of a MODBUS Request / Response transaction.	Initialized by the Client	Recopied by the server from the received request
Protocol Identifier	2 Bytes	0 = MODBUS protocol	Initialized by the Client	Recopied by the server from the received request
Length	2 Bytes	Number of following Bytes	Initialized by the Client (request)	Initialized by the Server(response)
Unit Identifier	1 Byte	Identification of a remote slave connected on a serial line or on other buses.	Initialized by the Client	Recopied by the server from the received request
Function-Code	1 Byte	Modbus RTU Function-Code		
Data	x Bytes	Data		

3.9. Funktionscodes

Folgende Funktionscodes sind im KS vario realisiert:



Funktionscode	Bedeutung
0x03	Lesen von Prozessdaten-Daten,Parameter oder Konfigurationsdaten
0x04	Lesen von Prozessdaten-Daten,Parameter oder Konfigurationsdaten
0x06	Schreiben einer einzelnen Date (Prozessdaten, Parameter oder Konfiguration)
0x10	Schreiben mehrerer Daten (Prozessdaten, Parameter oder Konfiguration)
0x17	Read/Write Register Ausgänge wortweise schreiben mit Startadresse und Anzahl der Ausgänge Eingänge wortweise lesen mit Startadresse und Anzahl der Eingänge
0x2B	Read Device Identification Auslesen von Herstellername, Produktcode, und Softwareversion Details siehe Kapitel 5 “Informationen zum Modbus-Protokoll”

Anmerkung: Die Funktionscodes 03 und 04 werden nicht unterschieden, da nicht sichergestellt werden kann, dass alle Master beide Funktionscodes unterstützen. Die Funktionscodes 0x17 und 0x2B werden nur vom Buskoppler unterstützt, hiermit kann auf die Prozessdaten-Caches zugegriffen werden. Wahlfreie Zugriffe auf beliebige andere Daten des KS vario sind mit diesem Funktionscode nicht möglich.

4. Adressierung im Ethernet Buskoppler

Neben dem Zugriff auf alle Parameter und Daten über Modbusadressen besteht außerdem die Möglichkeit, über BlueControl-Tool auf ausgewählte Daten (max. je 1080 pro Richtung) eines Caches zuzugreifen.

Die Adressierung erfolgt über das Datenfeld Unit Identifier des MBAP Headers.

Unit Identifier = 0 oder 1 --> Zugriff auf Daten des Cache-Speicher im Buskopplers
Unit Identifier = 2..255 --> Zugriff auf Daten des KS vario

4.1. Zugriff auf Daten des Cache-Speichers im Buskopplers

Zugriff mit Unit Identifier =0 oder 1.

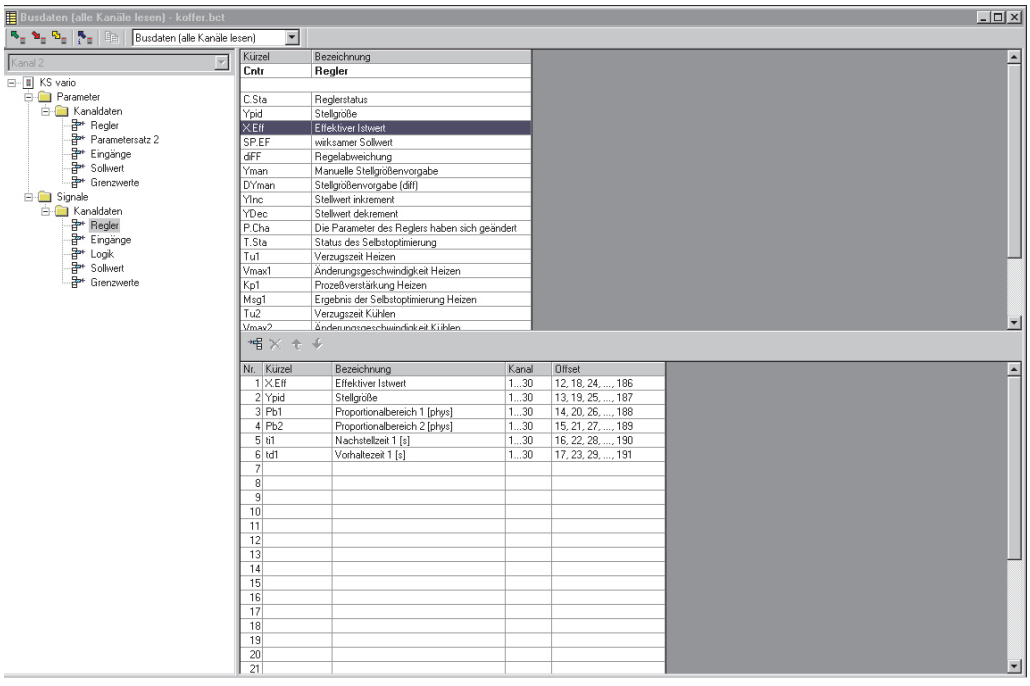
Beliebige Prozessdaten und Parameter können mit dem BlueControl Tool im KS vario selektiert werden. Diese Daten werden städig im Cache-Speicher des Buskopplers aktualisiert.

Die Prozessdaten umfassen einen Datenbereich von jeweils 1080 Worddaten im Write-Cache (Schreibbereich) und Read-Cache (Lesebereich).

4.1.1 Definition der zu übertragenden Werte im Engineering Tool "BlueControl"

Im BlueControl lassen sich die zu lesenden Daten auf 2 Arten auswählen (Schreibrichtung korrespondierend):

- ❑ Bis zu maximal 120 beliebige Parameter und Prozessdaten von beliebigen Kanälen zum Schreiben sowie max. 120 zum Lesen. Die Positionierung bestimmt die Reihenfolge in der Übertragung.
- ❑ Zusätzlich oder alternativ können - für alle Kanäle gemeinsam - bis zu jeweils 32 beliebige Parameter und Prozessdaten ausgewählt werden. So können mit der Auswahl einer Date z.B. die Istwerte von allen Kanälen (max. 30) übertragen werden. Ingesamt können somit bis zu 960 Schreib- und 960 Lesedaten definiert werden (32 Daten x 30 Kanäle).



Diese ausgewählten Daten (maximal 1080 Schreib- und 1080 Lesedaten) stehen im Buskoppler als Cache-Speicher in der im BlueControl definierten Reihenfolge zur VerfügungDie jeweiligen Indizes bzw. Offsets der einzelnen Daten werden über das BlueControl-Tool angezeigt bzw. können ausgedruckt werden.

4.1.2 Aufbau und Zugriff auf den Daten-Cache im Ethernet-Buskoppler

Die Prozessdaten umfassen einen Datenbereich von jeweils 1080 Worddaten im Write-Cache (Schreibbereich) und Read-Cache (Lesebereich). Der Zugriff erfolgt mit folgenden Adressen:

Modbusadresse	Datenbereich	Zugriffsart
1...1080	Read-Cache	Read
2001...3080	Write-Cache	Write

Index Read-Cache	**Inhalt**
1 **bis max.120**	beliebige Daten von beliebigen Kanälen
ab max. 121 **bis max. 1080**	Ausgewählte Daten (für alle Kanäle identisch): alle Daten Kanal 1 alle Daten Kanal 2 ... alle Daten Kanal 30

Index Write-Cache	**Inhalt**
1 **bis max.120**	beliebige Daten von beliebigen Kanälen
ab max. 121 **bis max. 1080**	Ausgewählte Daten (für alle Kanäle identisch): alle Daten Kanal 1 alle Daten Kanal 2 ... alle Daten Kanal 30

4.2. Wahlfreier Zugriff auf beliebige Daten des KS vario

Zugriff mit Unit Identifier = 2..255

Die Adresse wird in 2 Byte kodiert. Die höchstwertigsten 2 Bits (D15, D14) werden dazu herangezogen des Format in dem die Daten geschrieben oder gelesen werden zu definieren.

Das Modbusverzeichnis ist in gleich große Bereiche von jeweils 512 Worten aufgeteilt (Bit D13...D09). Über jeden dieser Bereiche lässt sich auf alle Daten für jeweils einen Regelkanal (1...30 Kanäle) zugreifen.

2 Bereiche haben einen Sonderstatus. Im untersten Adressbereich (Modbusabdr. 0..512) sind alle Gerätedaten hinterlegt. Im darauffolgenden Bereich (Adr. 512...1023) sind die wichtigsten Prozessdaten aller 30 Kanäle zusätzlich nocheinmal angeordnet. Dieser Bereich ist für Zugriffe von Visualisierungseinrichtungen gedacht.

Die Bedeutung der einzelnen Adress-Bits ist wie folgt: s.folgende Seite

Die detaillierte Adressübersicht aller Daten finden Sie im Dokument:

Parametertabelle für KS vario (9499-040-72918)

INTEGER/ FIX-Point-Modbusadressen:

MSB		LSB
D15 - D14	**D13 - D09**	**D08 - D00**
Datenformat **00: Integer** **01: Fix Point 1** **1X: reserviert für Float**	**Gerät, Visualisier., Kanal X** **00000: Gerätedaten** **00001: Visualisierungsdaten** **00010: Daten Kanal 1** **00011: Daten Kanal 2** **....** **11111: Daten Kanal 30**	**jeweilige Date**

Modbus-Verzeichnis (Datenformat: Integer):
Für den Fix Point 1-Bereich sind für die Adressen 4000 hex zu addieren.

Adressen	Daten
0 511 (1FF hex)	**Gerätedaten**
512 (200 hex) 1023 (3FF hex)	**Visualisierungsbereich Kanal 1..30**
1024 (400 hex) 1535 (5FF hex)	**Daten Kanal 1**
1536 (600 hex) 2047 (7FF hex)	**Daten Kanal 2**
....	....
15872 (3E00 hex) 16383 (3FFF hex)	**Daten Kanal 30**

FLOAT-Modbusadressen:

MSB		LSB
D15	**D14 - D10**	**D09 - D00**
Datenformat **0: reserviert für Integer und Fix Point 1** **1: Float**	**Gerät, Visualisier., Kanal X** **00000: Gerätedaten** **00001: Visualisierungsdaten** **00010: Daten Kanal 1** **00011: Daten Kanal 2** **....** **11111: Daten Kanal 30**	**jeweilige Date (Offset 2)**

Modbus-Verzeichnis (Datenformat: FLOAT):

Adressen	**Daten**
32768 (8000 hex) 33791 (83FF hex)	**Gerätedaten**
33792 (8400 hex) 34815 (87FF hex)	**Visualisierungsbereich Kanal 1..30**
34816 (8800 hex) 35839 (8BFF hex)	**Daten Kanal 1**
35840 (8C00 hex) 36863 (8FFF hex)	**Daten Kanal 2**
....	....
64512 (FC00 hex) 65535 (FFFF hex)	**Daten Kanal 30**

Die Daten belegen jeweils 4 Byte.

Übertragbare Werte:

Integer:	-30000 ... +32000	(Auflösung: +/-1)
Fix Point 1:	-3000.0 ...+3200.0	(Auflösung: +/- 0,1)
Float:	-1.0 E+037...+1.0 E+037	(Auflösung: +/- 1.4E-045)

Folgende Sonderwerte sind bei der Übertragung im **Integerformat** definiert:

-31000	Diese Date ist nicht definiert. Dieser Wert wird vom Regler zurückgegeben, wenn bei einer Blockabfrage eine Date innerhalb des Blockes nicht definiert ist.
-32000	Die Funktion ist abgeschaltet.
-32768	Entspricht 0x8000 Hex. der zu übertragende Wert liegt außerhalb des übertragbaren Integerbereichs.

Folgende Sonderwerte sind bei der Übertragung im **Floatformat** definiert:

-1.5E37	Diese Date ist nicht definiert. Dieser Wert wird vom Regler zurückgegeben, wenn bei einer Blockabfrage eine Date innerhalb des Blockes nicht definiert ist.

In den Code-Tabellen (Parametertabelle für KS vario (9499-040-72918)) sind die Adressen jedes Parameters für das entsprechende Datenformat in dezimalen Werten angegeben (Adr. = Integer ohne Nachkommastelle; 1 dP = Integer mit 1 Nachkommastelle; real = Float (IEEE-Format)).

5. Informationen zum Modbusprotokoll RTU

5.1. Allgemeines

Das MODBUS - Potokoll wurde zur Kommunikation zwischen einem Leitsystem und der Modicon-Steuerung definiert.

Es wurden die Protokolle ASCII und RTU definiert. Das Gerät KS vario unterstützt das RTU Protokoll.

Der Aufbau zur Übertragung eines Bytes im RTU-Protokoll ist folgendermaßen:

Startbit	8 Datenbits	Parity/Stopbit	Stopbit

Das Paritybit kann gerade oder ungerade gewählt werden. Wird kein Paritybit gewählt, so wird ein zusätzliches Stoppbit übertragen.

5.1.1 Genereller Nachrichtenaufbau

Die Nachricht wir in einen Datenbuffer mit einer maximalen Länge von 250 Byte eingelesen. Ist die Nachricht länger, so wird sie nicht akzeptiert. Es erfolgt keine Antwort durch das Gerät.

Die Nachricht setzt sich aus folgenden Elementen zusammen:

Geräteadresse	Funktioncode	Data	CRC	Endekennung

- Geräteadresse (Addr)
  Die Geräteadresse spezifiziert das Gerät. Geräteadressen können im Bereich von 1 - 247 vergeben werden.
  Die Geräteadresse 0 wird als Broadcast-Message verwendet. Eine Broadcast-Message kann für Schreibaufträge vergeben werden. Sie werden von allen Geräten am Bus ausgeführt. Da alle Geräte den Auftrag ausführen erfolgt keine Antwort durch die Geräte.
- Funktioncode
  Der Funktionscode definiert den Typ einer Nachricht. Es gibt 17 definierten Nachrichten. Welche Nachrichten unterstützt werden ist im Kapitel "Daten und Funktionskontrolle" beschrieben.
- Data
  Der Datenblock beinhaltet die weitere Spezifikation der Aktion die mit dem Funktionscode definiert wird. Die Länge des Datenblocks ist abhängig vom Funktionscode. Weitere Informationen siehe Kapitel "MODBUS Funktionsformat" (Kapitel 4). Der interne Datenbuffer beträgt 256 Byte. Somit können maximal 120 Integer- bzw. 60 Realdaten in einer Nachricht vorgegeben oder angefordert werden.
- CRC
  Der CRC-Code stellt sicher, das Übertragungsfehler erkannt werden können. Weitere Informationen siehe Kapitel "CRC"
- Endekennung
  Das Ende einer Nachricht wird definiert durch eine Zeit von 3,5 Zeichen in der kein Datentransfer stattgefunden hat. Weitere Informationen siehe Kapitel "Endekennung"

5.1.2 Parität (PrtY)

Bei der Parität kann Gerade, Ungerade und keine Parity gewählt werden.

Mit dem Paritätsbit kann überprüft werden, ob ein einzelner Fehler innerhalb eines Bytes bei der Übertragung aufgetreten ist.

Bei gerader Parität wird das Paritybit so eingestellt, dass die Summe der gesetzten Bits in den 8-Datenbits und dem Paritybit eine gerade Zahl ergibt. Entsprechendes gilt für die ungerade Parität.

Wird beim Empfang ein Paritätsfehler erkannt, so wird keine Antwortnachricht generiert.

Bei keine Parität können 1 bzw. 2 Stopbits ausgegeben werden (Festlegung über Konfiguration).

5.1.3 CRC

Bei dem CRC handelt es sich um ein 16-Bit Wert der der Nachricht angehängt wird. Er dient dazu feststellen zu können, ob die Übertragung einer Nachricht fehlerfrei erkannt wurde. Zusammen mit der Paritätskontrolle sollten alle möglichen Übertragungsfehler erkannt werden.

Wird beim Empfang ein Paritätsfehler erkannt, so wird keine Antwortnachricht generiert.

Der Algorithmus zur Erzeugung des CRC ist folgendermaßen:

- CRC-Register mit FFFF laden
- Exklusive ODER Verknüpfung der Sende/Empfangsbyte mit dem High-Teil des CRC-Registers
- CRC-Register um 1 Bit nach rechts schieben
- Wenn das hinausgeschobene Bit eine 1 ist, dass CRC-Register mit dem Wert A001 exklusive ODER verknüpfen.
- Schritt 3 und 4 für die anderen 7 Datenbits wiederholen.
- Schritt 2 bis 5 für alle weiteren Sende/Empfangsbyte wiederholen.
- Ergebnis des CRC-Registers an die Nachricht anhängen. Zuerst den High-Teil. Bei der Kontrolle einer Empfangsnachricht ergibt sich im CRC-Register eine 0 wenn die Nachricht inklusive des CRC bearbeitet wird.

5.1.4 Endekennung

Die Endekennung einer Nachricht ist spezifiziert als Ruhesituation auf dem Modbus mit einer Länge von 3,5 Zeichen. Nach dem Verstreichen dieser Zeit darf ein Slave frühestens mit seiner Antwort beginnen oder ein Master frühestens eine neue Nachricht aussenden.

Die Auswertung einer Nachricht darf bereits beginnen, wenn erkannt wird, dass die Ruhebedingung auf den Modbus für mehr als 1,5 Zeichen aufgetreten ist. Eine Antwort wird jedoch frühestens nach 3,5 Zeichen gestartet.

5.1.5 Modbus Funktionsformat

Je nach Funktionscode unterscheidet sich die Bedeutung des Datenbereiches. Das Modbusprotokoll definiert 17 unterschiedliche Funktionscodes.

Um die Abfrage und Vorgabe von Prozessdaten, Parameter und Konfigurationsdaten mit möglichst wenig Zugriffen zu ermöglichen, werden die entsprechenden Bereiche zusammen gruppiert.
Die Prozessdaten sind dabei in unterschiedlicher Zusammenfassung mehrfach definiert.

Beispiel für eine Übertragung

Anfrage:

Feldname	Wert (Hex)	Bedeutung
Adresse	11	Adresse 17
Funktion	04	Lesen Parameter/Konfiguration
Startadresse High Startadresse Low	03 EC	Anfangsadresse 1004
Anzahl der Werte	00 03	Anzahl der Werte 03
CRC	CRC-Byte1 CRC-Byte2	

Antwort:

Feldname	Wert	Bedeutung
Adresse	11	Adresse 17
Funktion	04	Lesen Parameter/Konfiguration
Anzahl der Bytes	06	Es werden 6 Datenbytes geschickt
Wert1	04 2A	Wert1 = 1066
Wert2	00 8C	Wert2 = 140
Wert3	10 3E	Wert3 = 4158
CRC	CRC-Byte1 CRC-Byte2	

5.1.6 Funktionscodes

Codetabelle siehe Kapitel 3.9

Details zum Funktionscode 43 (0x2B), Read Device Identification:

Object ID	Object Name / Description	Type	M/O	category
0x00	VendorName	ASCII String	Mandatory	Basic
0x01	ProductCode	ASCII String	Mandatory	
0x02	MajorMinorRevision	ASCII String	Mandatory	
0x03	VendorUrl	ASCII String	Optional	Regular
0x04	ProductName	ASCII String	Optional	
0x05	ModelName	ASCII String	Optional	
0x06	UserApplicationName	ASCII String	Optional	
0x07 ... 0x7F	Reserved		Optional	
0x80 ... 0xFF	Private objects may be optionally defined. The range (0x080 - 0xFF) is product depandant	Device dependant	Optional	Extended

Unterstützt werden die Zugriffe auf die Basic- und die Regular-Objekte. Als Conformity Level wird Level 2 [regular identification (stream access only)] unterstützt.

REQUEST

Function Code	1 Byte	0x2B
MEI Type *	1 Byte	0x0E
Read Device ID code	1 Byte	01 / 02 / 03 / 04
Object ID	1 Byte	0x00 to 0xFF

MEI= Modbus Encapsulated Interface

RESPONSE

Function Code	1 Byte	0x2B
MEI Type	1 Byte	0x0E
Read Device ID code	1 Byte	01 / 02 / 03 / 04
Conformity level	1 Byte	
More Folows	1 Byte	00 / FF
Next Object Id	1 Byte	Object ID number
Number of objects	1 Byte	
List of		
Object ID	1 Byte	
Object length	1 Byte	
Object Value	Object length	Depending on the object ID

ERROR

Function Code	1 Byte	0xAB: Fc 0x2B + 0x80
MEI Type	1 Byte	14
Exception Code	1 Byte	01, 02, 03, 04

Object Inhalte:

Object: 0 = PMA Prozeß- und Maschinen-Automation GmbH
Object: 1 = KS VARIO BK ETH
Object: 2 = V1.00 (aktuelle Softwareversion)
Object: 3 = pma-online.de
Object: 4 = KSvario
Object: 5 = Ethernet
Object: 6 = KSVC-101-00131

5.2. Beispiele

5.2.1 Lesen von Prozessdaten, Parametern oder Konfigurationsdaten

Der Aufbau einer Nachricht ist folgendermaßen:

Anfrage:

Feldname	Wert	Bedeutung
Adresse	11	Adresse 17
Funktion	03 oder 04	Lesen von Prozessdaten-Daten, Parameter oder Konfigurationsdaten
Startadresse High Startadresse Low	04 98	Anfangsadresse 0498 (ti1 / Kanal 1)
Anzahl der Werte	00 02	2 Daten
CRC	CRC-Byte1 CRC-Byte2	

Antwort:

Feldname	Wert	Bedeutung
Adresse	11	Adresse 17
Funktion	03 oder 04	Lesen von Prozessdaten-Daten, Parameter oder Konfigurationsdaten
Anzahl der Bytes	04	Es werden 4 Datenbytes geschickt
Parameter1	00 B4	Prozessdaten-Daten, Parameter/Konfigurationsdate 0498= 180
Parameter ti2	01 4D	Prozessdaten-Daten, Parameter/Konfigurationsdate 0499= 333
CRC	CRC-Byte1 CRC-Byte2	

Broadcast ist nicht möglich.

Ist der 1. Parameter/Konfigurationsdate nicht definiert, so wird eine Fehlermeldung "ILLEGAL DATA ADDRESS" erzeugt.

Sind in dem auszulesenden Bereich nach dem 1. Wert andere nicht definiert, so werden diese mit dem Wert "NOT DEFINED VALUE" eingetragen. Dieses dient dazu Bereiche mit Lücken mit einer Nachricht auslesen zu können.

5.2.2 Schreiben einer einzelnen Prozessdate, Parameter o. Konfiguration

Der Aufbau einer Nachricht ist folgendermaßen:

Anfrage:

Feldname	Wert	Bedeutung
Adresse	11	Adresse 17
Funktion	06	Schreiben einer einzelnen Date (Prozessdaten, Parameter oder Konfiguration)
Schreibadr. High Schreibadr. Low	0D 57	Schreibadresse 15990 (SetpInterface von Kanal 30)
Wert = 123	00 7B	
CRC	CRC-Byte1 CRC-Byte2	

Antwort:

Feldname	Wert	Bedeutung
Adresse	11	Adresse 17
Funktion	06	Schreiben einer einzelnen Date (Prozessdaten, Parameter oder Konfiguration)
Schreibadr. High Schreibadr. Low	3E 76	Schreibadresse 15990 (SetpInterface von Kanal 30)
Wert = 123	00 7B	
CRC	CRC-Byte1 CRC-Byte2	

Die Antwortnachricht entspricht bei Fehlerfreiheit exakt der Vorgabe.
Broadcast ist möglich.

Eine Vorgabe im Datenformat Real ist nicht möglich, da als Wert nur 2 Byte übergeben werden können.

Ist der Werte ausserhalb des einstellbaren Bereichs, so wird die Fehlermeldung "ILLEGAL DATA VALUE" erzeugt. Die Date bleibt unverändert.

Kann die Date nicht beschrieben werden (z.B. Konfigurationsdate und das Gerät befindet sich in Online), so wird eine Fehlermeldung "ILLEGAL DATA VALUE" erzeugt.

5.2.3 Schreiben mehrerer Prozessdaten, Parameter + Konfigurationsdaten

Der Aufbau einer Nachricht ist wie folgendermaßen:

Vorgabe:

Feldname	Wert	Bedeutung
Adresse	11	Adresse 17
Funktion	10	Schreiben mehrerer Prozessdaten-Daten, Parameter oder Konfigurationsdaten
Startadresse High Startadresse Low	0D 57	Schreibadresse 3415
Anzahl der Werte	00 02	2 Werte
Anzahl der Bytes	04	Es werden 4 Datenbytes geschickt
Parameter/ Konfigurationsdate 15	00 DE	Prozessdaten-Date, Parameter oder Konfigurationsdate 3415 = 222
Parameter/ Konfigurationsdate 16	01 4D	Prozessdaten-Date, Parameter oder Konfigurationsdate 3416 = 333
CRC	CRC-Byte1 CRC-Byte2	

Antwort:

Feldname	Wert	Bedeutung
Adresse	11	Adresse 17
Funktion	10	Schreiben mehrerer Prozessdaten-Daten, Parameter oder Konfigurationsdaten
Startadresse High Startadresse Low	0D 57	Schreibadresse 3415
Anzahl der Werte	00 02	2 Prozessdaten-Daten, Parameter/Konfigurationsdaten
CRC	CRC-Byte1 CRC-Byte2	

Broadcast ist möglich.

Ist der 1. Wert nicht definiert, so wird eine Fehlermeldung "ILLEGAL DATA ADDRESS" erzeugt.

Kann der 1. Wert nicht beschrieben werden (Konfiguration und Gerät ist in Online), so wird eine Fehlermeldung "ILLEGAL DATA VALUE" erzeugt.

Sind in dem vorgegebenen Bereich nach dem 1 Wert andere nicht definiert oder momentan nicht beschreibbar, so werden diese überlesen. Daten werden an diesen Stellen nicht verändert. Dieses dient dazu Bereiche mit Lücken bzw. momentan nicht beschreibbaren Daten mit einer Nachricht verändern zu können. Es wird keine Fehlermeldung ausgegeben.

Sind Werte ausserhalb der einstellbaren Grenzen, so wird die Fehlermeldung "ILLEGAL DATA VALUE" erzeugt. Die Auswertung der nachfolgenden Daten wird nicht durchgeführt. Bereits fehlerfrei übernommenen Daten sind aktiv.

Der Modbus sieht in seinem Fehlerprotokoll keine Information bezüglich der Position des Fehlers vor. Wenn dies gewünscht wird, so muß eine Date definiert werden, die die Position des letzten Fehlers beinhaltet. Diese kann im Fehlerfall vom Master ausgelesen werden.

5.2.4 Auslesen und Vorgabe von Daten im Float-Format

Es können Ebene-1-Daten, Parameter und Konfigurationsdaten im Float-Format angefordert und vorgegeben werden. (Funktionscodes 03, 04, 16)
Eine einzelne Vorgabe einer Date im Float-Format mit dem Schlüssel 06 ist nicht möglich, da mit dieser Funktion nur 2 Byte für den Wert der Date übertragen werden können.

Werden Daten im Float-Format gewünscht, so muß die Adresse der gewünschten Date folgendermaßen berechnet werden:
Adresse der Date im Integer-Format multipliziert mit dem Faktor 2
Addition eines Versatzes von 8000H.

Bei der Angabe für die "Anzahl der Werte" ist der doppelte Wert wie bei einer Nachricht für Daten im Integer-Format notwendig.
Entsprechend verdoppelt sich auch der Wert im Feld "Anzahl der Daten-Bytes".

Es werden grundsätzlich alle Daten in Float umgerechnet. Dieses gilt auch für Status - bzw. Steuerworte.
Die Daten werden im Motorola-Format übertragen (Exponent zuerst gefolgt von Mantisse)
Der Aufbau einer Nachricht, wie unter dem vorhergehenden Kapitel beschrieben, sieht im Float-Format folgendermassen aus:

Vorgabe:

Feldname	Wert	Bedeutung
Adresse	11	Adresse 17
Funktion	10	Schreiben mehrerer Prozessdaten-Daten, Parameter oder Konfigurationsdaten
Startadresse High Startadresse Low	9A AE	Schreibadresse 2 * 3415 + 8000H für Float-Format
Anzahl der Werte	00 04	2 Werte im Float-Format
Anzahl der Bytes	08	Es werden 8 Datenbytes geschickt
Parameter/ Konfigurationsdate 15	43 5E 00 00	Prozessdaten-Date, Parameter oder Konfigurationsdate 3415 = 222
Parameter/ Konfigurationsdate 16	43 A6 80 00	Prozessdaten-Date, Parameter oder Konfigurationsdate 3416 = 333
CRC	CRC-Byte1 CRC-Byte2	

Antwort:

Feldname	Wert	Bedeutung
Adresse	11	Adresse 17
Funktion	10	Schreiben mehrerer Prozessdaten-Daten, Parameter oder Konfigurationsdaten
Startadresse High Startadresse Low	9A AE	Schreibadresse 2 * 3415 + 8000H für Float-Format
Anzahl der Werte	00 04	2 Prozessdaten-Daten, Parameter oder Konfigurationsdaten im Float-Format
CRC	CRC-Byte1 CRC-Byte2	

5.3. Fehlerprotokoll

Das Fehlerprotokoll wird erzeugt, wenn eine Nachricht fehlerfrei empfangen wurde, die Interpretation der Nachricht oder die Änderung einer Date jedoch nicht möglich ist.
Wird ein Übertragungsfehler festgestellt, so wird keine Antwort erstellt. Der Master muß die Nachricht erneut abzusenden.
Erkannte Übertragungsfehler sind:

- Paritätsfehler
- Framing-Fehler (Kein Stopbit empfangen)
- Overrun-Fehler (Empfangsbuffer ist übergelaufen oder Daten konnten nicht schnell genug vom UART abgeholt werden)
- CRC-Fehler

Der Datenaufbau des Fehlerprotokolls ist folgendermassen:

Feldname	Wert	Bedeutung
Adresse	11	Adresse 17
Funktion	90	Fehlerprotokoll für die Nachricht Schreiben mehrerer Parameter/Konfigurationsdaten
Fehlercode	02	ILLEGAL DATA ADDRESS
CRC	CRC-Byte1 CRC-Byte2	

Im Feld Funktion wird das höchstwertigste Bit gesetzt.
Im darauf folgenden Byte wird der Fehlercode übertragen.

Folgende Fehlercodes sind definiert:

Code	Name	Bedeutung
01	ILLEGAL FUNCTION	Der empfangen Funktionscode ist im Gerät nicht definiert.
02	ILLEGAL DATA ADDRESS	Die empfangene Adresse ist im Gerät nicht definiert. Werden mehrere Daten gleichzeitig gelesen (Funktionscode 01, 03, 04) oder geschrieben (Funktionscode 0F, 10), so wird dieser Fehler nur erzeugt, wenn die erste Date nicht definiert ist.
03	ILLEGAL DATA VALUE	Der empfangene Wert liegt ausserhalb der Einstellgrenzen oder kann momentan nicht beschrieben werden (Gerät befindet sich nicht im Konfigurationsmode). Werden mehrere Daten gleichzeitig geschrieben (Funktionscode 0F, 10), so wird dieser Fehler nur erzeugt, wenn die erste Date nicht beschrieben werden kann.
06	SLAVE DEVICE BUSY	Wird zurückgesendet, wenn kein Kommunikationskanal mehr zur Verfügung steht. Maximal unterstützt der Koppler 16 Kommunikationskanäle.
0B	GATEWAY TARGET DEVICE FAILED TO RESPOND	Wird zurückgesendet, wenn keine Kommunikation mit dem KS vario möglich ist.

Es gibt im Modbusprotokoll weitere definierte Fehlercodes die momentan aber nicht unterstützt werden:

Code	Name	Bedeutung
04	SLAVE DEVICE FAILURE	Ein nicht reproduzierbarer Fehler ist bei der Verarbeitung der Nachricht aufgetreten
05	ACKNOWLEDGE	Das Gerät hat eine Anforderung empfangen und verarbeitet diese. Da die Bearbeitung sehr lange dauert wird diese Antwort ausgegeben, um den Timeout der Schnittstelle nicht ablaufen zu lassen. Der Master kann über die Diagnose ein Polling ausführen um festzustellen ob die Bearbeitung beendet ist.
07	NEGATIVE ACKNOWLEDGE	Das Gerät kann den geforderten Auftrag nicht ausführen. Eventuell kann diese Fehlermeldung ausgegeben werden wenn eine Konfigurationsdate geändert werden soll, das Gerät sich aber nicht im Konfigurationsmode befindet.
08	MEMORY PARITY ERROR	Paritätsfehler beim Lesen des Speichers gefunden.

Subject to alterations without notice.
Bei Änderungen erfolgt keine Mitteilung.
Modifications sans avertissement réservées.

© PMA Prozeß- und Maschinen-Automation GmbH
Postfach 310 229, D - 34058 Kassel
Printed in Germany 9499 040 69718 (06/2008)

A4