



RO-Serie

Serielles-Protokoll

1	EINLEITUNG	3
2	REGISTER-ZUGRIFFE	4
2.1	Was sind überhaupt Register ?	4
2.2	Registerzugriff mit 8/ 16/ 32 oder 64 Bit-Datenbreite	4
2.3	Registerbelegung	5
3	DAS ÜBERTRAGUNGSPROTOKOLL	6
3.1	Geschwindigkeit und Schnittstellenparameter	6
3.2	Aufbau des „Sende-Strings“	6
3.2.1	Anfangszeichen	6
3.2.2	MODUL-NR	6
3.2.3	JOB-ID	7
3.2.4	COMMAND	7
3.2.5	WIDTH	7
3.2.6	ADDRESS	7
3.2.7	DATA	7
3.2.8	CHECKSUM	7
3.2.9	END	8
3.3	Aufbau des „Antwort-Strings“	9
3.3.1	Antwort-String „O“ (OK)	9
3.3.2	Antwort-String „D“ (DATA)	9
3.3.3	Startzeichen mit Antwort „E“ (ERROR)	10
4	COPYRIGHTS AND TRADEMARKS	11

1 Einleitung

Die Kommunikation mit unseren Modulen erfolgt über eine serielle Schnittstelle. Die Module reagieren hierbei auf eine Anfrage (Sendestring), bearbeiten diese Anfrage und antworten mit einer Antwort (Empfangsstring).

Die Anfrage:

Der Sendestring besteht aus einem String, der mit einem <SOH>-Zeichen anfängt, der Inhalt besteht dann aus mehreren ASCII-Zeichen mit den Daten („0“ – „9“ und „A“ – „F“). Das Ende dieses Strings wird mit einem <CR> gekennzeichnet.

Die Antwort:

Nach Abarbeiten der Anfrage wird der Antwortstring zurückgesendet. Dieser kann die Anfrage einfach nur „bestätigen“, es können „Daten“ zurück übertragen werden oder es kann ein Fehlercode übermittelt werden.

Die genauere Zusammensetzung der Sende- und Empfangsstrings werden in den folgenden Kapiteln beschrieben.

2 Register-Zugriffe

2.1 Was sind überhaupt Register ?

Register sind kleine Zwischenspeicher, die die zu schreibenden oder zu lesenden Daten zwischenpuffern. Die Daten verbleiben solange gespeichert, bis sie überschrieben werden oder deren Stromversorgung abgeschaltet wird. Sie besitzen einen Adressbereich um sie anzusprechen. Mit Hilfe der Adresse kann man von Registern lesen oder auf ihnen schreiben. Der Registerzugriff erfolgt also adressiert.

Eine Besonderheit bilden die Register der Eingangszustandsänderungs-Merker. Werden diese ausgelesen, so werden deren Daten zugleich zurückgesetzt.

2.2 Registerzugriff mit 8/ 16/ 32 oder 64 Bit-Datenbreite

Der Registerzugriff kann in unterschiedlicher Datenbreite erfolgen. Die Datenbreite kann wahlweise 8 (Byte), 16 (Word), 32 (Long) oder 64 (eXtralong) Bit breit sein. Mit der Adresse wird der Bereich ausgewählt, auf dem man Zugriff erlangen möchte. Eine Adresse weist auf 8 Bits.

Erfolgt beispielsweise ein 32 Bit-Zugriff auf Adr. 0004, so werden 4 x 8 Byte (als ein Datenblock) beginnend mit Adresse 0004 bis Adresse 0007 ausgelesen bzw. beschrieben.

Tabelle mit Aufteilung in 8,16,32,64 Bit Registerzugriff

Zugriffsbreite	Adresse	00	01	02	03	04	05	06	07
8 Bit	0000 hex	X							
16 Bit	0000 hex	X	X						
32 Bit	0000 hex	X	X	X	X				
32 Bit	0004 hex					X	X	X	X
64 Bit	0000 hex	X	X	X	X	X	X	X	X

Beim Schreiben sowie beim Lesen werden die Daten Byte für Byte beschrieben bzw. ausgelesen. Dabei ist auf die Byte-Reihenfolge zu achten. Begonnen wird mit den niederwertigen Bytes (Byteorder: Little Endian).

Byte-Reihenfolge der Daten im Register

Zugriffsbreite	Adresse	Wert	00	01	02	03	04	05	06	07
8 Bit	04	1a					1a			
16 Bit	06	1a1b							1b	1a
32 Bit	00	01020304	04	03	02	01				
32 Bit	04	01020304					04	03	02	01
64 Bit	00	0102030405060708	08	07	06	05	04	03	02	01

2.3 Registerbelegung

Dies finden Sie im Dokument RO-Registerbelegung.

3 Das Übertragungsprotokoll

3.1 Geschwindigkeit und Schnittstellenparameter

Die Serielle Schnittstelle ist mit 115200 Baud, 8 Datenbits, 1 Stoppbit und keine Parität konfiguriert (8,n,1).

3.2 Aufbau des „Sende-Strings“

Der Beginn eines Pakets wird mit dem Zeichen <SOH> („Start of Header“, 01 hex) gekennzeichnet und das Ende mit dem Zeichen <CR> („Carriage-Return“, 0D hex).

Mit Ausnahme vom Anfangs- und Endzeichen, sowie COMMAND und WIDTH werden hexadezimalen Daten ASCII-kodiert übertragen. Wird beispielsweise die Adresse 0012 hex angesprochen, so werden von links nach rechts die 4 ASCII-Zeichen „0012“ nacheinander übertragen. Die ASCII-kodierten hexadezimalen Werte können nur aus Zahlen von 0-9 und aus Buchstaben von A-F gebildet werden.

Beispiel für ein Schreibbefehl:

Im folgenden Beispiel wird auf die Adresse 0012 hex das Datenwort 0F hex geschrieben (der Sendestring besteht aus 16 ASCII-Zeichen).

Zeichen Nr.	Bedeutung	ASCII-Zeichen	Hexadezimal
1	Start Of Header (Markiert start eines Packets)	<SOH>	01
2, 3	MODUL-NR (=34 hex)	34	33, 34
4, 5	JOB-ID (= 12 hex)	12	31, 32
6	COMMAND	W	57
7	WIDTH (Datenbreite)	B	42
8, 9, 10, 11	ADDRESS (= 0012 hex)	0012	30, 30, 31, 32
12, 13	DATA (= 0F hex)	0F	30, 46
14, 15	CHECKSUM (= 29D hex, es werden aber nur die unteren 8 Bit übertragen)	9D	39, 44
16	END (Carriage Return)	<CR>	0D

3.2.1 Anfangszeichen

Mit dem Zeichen <SOH> wird der Beginn einer Datenübertragung gekennzeichnet.

3.2.2 MODUL-NR

Die Modul-Nr kennzeichnet das ausgewählte Modul. (Die Modulnummern können auf den Modulen mit Hilfe von DIP-Schaltern eingestellt werden.)

3.2.3 JOB-ID

Die JOB-ID kennzeichnet den aktuellen Sende-String. Zwei hintereinanderliegende Sendestrings dürfen hierbei nicht die gleiche JOB-ID erhalten.

TIP:

Die Sende-Routine sollte die JOB-ID nach erfolgreicher Übertragung eines Sendestrings um „1“ erhöhen. So werden nacheinander die Nummern „0“ dann „1“ ... „255“ und dann wieder „0“ verwendet.

3.2.4 COMMAND

Mit COMMAND wird das Zeichen „W“ (write) zum Beschreiben von Register oder „R“ (read) für das Auslesen von Register gekennzeichnet.

3.2.5 WIDTH

Abhängig vom verwendeten „COMMAND“ und der benutzten Datenbreite „WIDTH“ werden unterschiedlich lange Daten versendet.

COMMAND	WIDTH	Datenbreite (Bits)	Anzahl ASCII Zeichen
W	B	8	2
W	W	16	4
W	L	32	8
W	X	64	16
R		0	0

Beim COMMAND = „R“ hat die das WIDTH Zeichen keinen Einfluss auf den Sendestring. Es steuert hierbei die Daten-Länge des Empfangs-Strings.

3.2.6 ADDRESS

Die Register sind 16 Bit breit und werden mit 4 hexadezimalen Zeichen dargestellt. Da die Übertragung jedes hexadizimalen Zeichens mit einem ASCII-Zeichen repräsentiert wird, werden 4 ASCII-Zeichen benötigt.

3.2.7 DATA

Das Datenfeld ist nur bei einem Schreibkommando vorhanden. Bei einem Lesekommando fällt dieses weg.

3.2.8 CHECKSUM

CHECKSUM ist eine Prüfsumme. Es ist eine hexadezimale Aufsummierung aller Zeichen (einschliesslich <SOH>) bis zur Prüfsumme. Der Empfänger vergleicht seine Aufsummierung mit der übertragenen Prüfsumme. Sind diese gleich, gilt die Übertragung als fehlerfrei.

In hexadezimal:

$$01 + 33 + 34 + 31 + 32 + 57 + 42 + 30 + 30 + 31 + 32 + 30 + 46 = 29D$$

→ CHECKSUM = 9D hex (2 ist der Überlauf und fällt somit weg).

3.2.9 END

Das Ende des Sende-Strings wird mit einem <CR> gekennzeichnet (Carriage Return, 0D hex).

3.3 Aufbau des „Antwort-Strings“

Der Antwort String startet mit dem Zeichen („O“, „E“ oder „D“) gefolgt von ASCII Zeichen und wird mit einem <CR> (Carriage Return, 0d hex) beendet.

3.3.1 Antwort-String „O“ (OK)

Dies bedeutet, dass die Antwort „OK“ war und teilt dem Sender mit, dass der Schreibbefehl erfolgreich ausgeführt wurde.

„O“-Format

Zeichen Nr.	Bedeutung
1	ASCII-„O“ für OK
2	JOB-ID (Higher Byte)
3	JOB-ID (Lower Byte)
4	CHECKSUM (Higher Byte)
5	CHECKSUM (Lower Byte)
6	END (Carriage Return)

OK-Antwort zum Sendestring Beispiel aus Kapitel 3.2

Zeichen Nr.	Bedeutung	ASCII-Zeichen	Hexadezimal
1	OK	O	4F
2, 3	JOB-ID (= 12 hex)	12	31, 32
4, 5	CHECKSUM (= B2 hex)	B2	42, 32
6	END (Carriage Return)	<CR>	0D

3.3.2 Antwort-String „D“ (DATA)

Ein Antwort-String beginnend mit einem „D“ kennzeichnet Antwortdaten, die aus einem Lesebefehl resultieren.

„D“-Format

Zeichen Nr.	Bedeutung
1	„D“ für DATA (Antwort des Lesebefehls)
2	JOB-ID (Higher Byte)
3	JOB-ID (Lower Byte)
4,5 ...	n-Daten
4 + n	CHECKSUM (Higher Byte)
5 + n	CHECKSUM (Lower Byte)
6 + n	END (Carriage Return)

Mit n = 2, 4, 8 oder 16 Zeichen (abhängig von WIDTH des Sendestrings)

Kommando + WIDTH vom Sende-String und die Antwort-Daten

Lese-Anfrage		Antwort-Daten	
Command	WIDTH	Datenbreite (Bits)	Anzahl ASCII-Zeichen
R	B	8	2
R	W	16	4
R	L	32	8
R	X	64	16

3.3.3 Startzeichen mit Antwort „E“ (ERROR)

Der zuletzt gesendete „Sende-String“ wurde fehlerhaft empfangen.

„E“-Format

Zeichen Nr.	Bedeutung
1	„E“ für Error
2	Fehlercode
3	END (Carriage Return)

Fehlercode

Hexadezimal	Bedeutung
31	Kommando ungültig; es wurde ein ungültiges COMMAND benutzt
32	Ungültige Datenlänge; die Datenlänge des Sende-Strings stimmt nicht
33	Checksum error; Prüfsummenfehler

4 Copyrights and trademarks

Linux is registered trademark of Linus Torvalds.

Windows CE is registered trademark of Microsoft Corporation.

USB is registered trademark of USB Implementers Forum Inc.