



Elektro-Automatik

Anleitung & Übersicht zur Befehlsliste UTA Interface

Inhaltsverzeichnis

1. Vorwort	3
2. Grundsätzliches	3
3. Struktur des 24 Byte Befehlsdatagramms.....	3
4. Befehlsliste	5
5. Antwortliste	6
6. Erklärung der Befehle	7
7. Beispiele	8

1. Vorwort

Um das UTA Interface von Applikationen aus anzusprechen, die in C, C++, Visual Basic oder anderen Programmiersprachen erstellt wurden, sind grundsätzlich nur zwei Dinge nötig: ein COM-Port und das unten beschriebene Protokoll. Der COM-Port wird als virtueller COM-Port vom Treiber des USB-Gerätes erzeugt. Sprich, es sind für jedes UTA Interface zwei Geräte in Windows vorhanden, ein „USB Serial converter“ und ein „USB Serial port (COMx)“. Nach der Installation des Treibers für ein neues UTA12 Gerät wird daher empfohlen, im Windows Gerätemanager nachzuprüfen, ob beide vorhanden sind.

Entgegen der Hinweise im UTA12 Handbuch, wo der virtuelle COM-Port (VCP) ausgeschaltet werden soll, damit die Steuerungssoftware UTA12 Control richtig funktioniert, ist der VCP hier nicht auszuschalten.

Wichtig! Auf der USB-Seite des UTA Interfaces arbeitet ein Chip, der eine Umsetzung von USB auf RS232 erledigt. Daher muß der COM-Port nach dem Öffnen konfiguriert werden.

Settings: 57600Bd, 1 Stopbit, 1 Startbit, keine Parität.

2. Grundsätzliches

Die Kommunikation mit dem UTA Interface erfolgt über Telegramme in Hexadezimalform. Ein Befehlszyklus besteht typischerweise aus BEFEHL SENDEN und ANTWORT EMPFANGEN.

Das Interface arbeitet also als Slave und wartet auf Eingaben.

Befehle und Antworten sind stets 24 Bytes lang, deren Struktur wird unten im Einzelnen erklärt.

Falls der Treiber des USB-Gerätes ordnungsgemäß installiert wurde, kann man den COM-Port öffnen.

Das UTA Interface kann generell nach der Reihenfolge OPEN-WRITE-READ-CLOSE angesprochen.

Es ist jedoch ratsam, nicht bei jedem Befehl den Port zu öffnen und hinterher wieder zu schließen, sondern solange geöffnet zu lassen, wie das Interface benutzt wird!

Ist das Gerät geöffnet, kann man einfach die Telegramme nach dem unten aufgeführten Protokoll an das UTA schicken.

Bevor dies geschieht, sollte der Port konfiguriert werden. Dies hat jedesmal nach dem Öffnen zu erfolgen. Siehe oben bei 1.

Wird das Gerät nicht mehr benötigt, sollte der COM-Port geschlossen werden.

3. Struktur des 24 Byte Befehlsdatagramms

Wird in C programmiert, empfehlen wir die Verwendung einer Struktur wie folgt: (dies ist nur ein Beispiel)

```
struct uta
{
    uint    uiHeader;
    uchar   ucIDnumber;
    uchar   ucCommand;
    char     ucDatafield[18];
    uint    uiChecksum;
};
```

Die Syntax der Datentypen variiert von Sprache zu Sprache und muß dementsprechend angepaßt werden.
Zur Erklärung der Elemente:

uint uiHeader = unsigned integer (2 Bytes), Headerwort, immer 0xAA55

uchar uclDnumber = unsigned char (1 Byte), ID Nummer des UTA

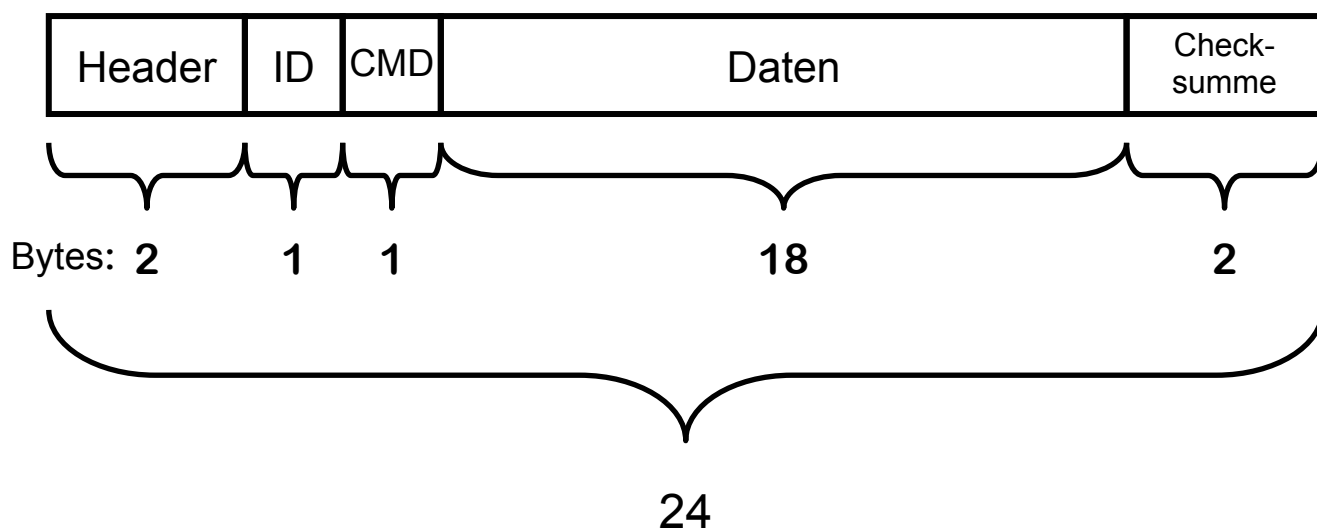
Die ID Nummer sollte wenigstens einmal gesetzt werden, wird im UTA gespeichert und muß zur Kommunikation mit dem UTA angegeben werden, da Befehle, die eine andere ID beinhalten, ignoriert werden.

uchar ucCommand = unsigned char (1 Byte), enthält den Befehl (Hexbyte, siehe Tabelle)

char ucDatafield[18] = char (18 Bytes), enthält Parameter oder ASCII Zeichen für den jeweiligen Befehl (siehe Tabelle)

uint uiChecksum = unsigned integer (2 Bytes), Checksumme über die ersten 22 Byte des Datagramms, dient zur einfachen Sicherheit der Datenübertragung und wird wie folgt gebildet: Summe über die ersten 22 Bytes bilden (ergibt ein Wort), davon ein Komplement, und diese Checksumme in den letzten zwei Bytes ablegen.

Bildliche Darstellung des Datagramms:



4. Befehlsliste

Datagrammformat gesendet vom PC (Befehl an das UTA)																
Name	hex	header	id	command	value1		value2								check sum	Hinweise
SetVoltage	1F	AA55	xx	1F	xx	xx	00	00	00	00	00	00	00	00	xxxx	value1 reicht von 0x0000 bis 0xffff
GetVoltage	2F	AA55	xx	2F	00	00	00	00	00	00	00	00	00	00	xxxx	value1 reicht von 0x0000 bis 0xffff
SetCurrent	3F	AA55	xx	3F	xx	xx	00	00	00	00	00	00	00	00	xxxx	
GetCurrent	4F	AA55	xx	4F	00	00	00	00	00	00	00	00	00	00	xxxx	value1 reicht von 0x00 bis 0xff
SetStatus	5F	AA55	xx	5F	xx	xx	00	00	00	00	00	00	00	00	xxxx	value1 & value2 MUSS ASCII Zeichen enthalten
GetStatus	6F	AA55	xx	6F	00	00	00	00	00	00	00	00	00	00	xxxx	
SetIDString	7F	AA55	xx	7F	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xxxx	ohne ID
GetIDString	8F	AA55	xx	8F	00	00	00	00	00	00	00	00	00	00	xxxx	
Echo	9F	AA55	00	9F	00	00	00	00	00	00	00	00	00	00	xxxx	value1 reicht von 0x00 bis 0xff
SetIDN	EF	AA55	00	EF	xx	xx	00	00	00	00	00	00	00	00	xxxx	ohne ID
GetIDN	FF	AA55	00	FF	00	00	00	00	00	00	00	00	00	00	xxxx	
GetMaxValues	48	AA55	xx	48	00	00	00	00	00	00	00	00	00	00	xxxx	value1 & value2 MUSS ASCII Zeichen enthalten
SetUserText	C8	AA55	xx	C8	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xxxx	erzeugt keine Antwort
GetUserText	D8	AA55	xx	D8	00	00	00	00	00	00	00	00	00	00	xxxx	
Reset	E8	AA55	xx	E8	00	00	00	00	00	00	00	00	00	00	xxxx	

Fehlercodes der Antwort AF:

Hex	Ascii	Bedeutung
0x0001:	check sum mismatch	Checksumme stimmt nicht
0x0002:	id string empty	leerer Bezeichner
0x0003:	header mismatch	Datagrammkopf nicht AA55
0x0004:	eprom write error	EEPROM Daten nicht übereinstimmend

Bitzuordnung für Befehl SetStatus (5F):

Bit 7: Stby (Ausgang aus/ein), Wertigkeit 0x80

Bit 6: REM (Fernsteuerung ein/aus), Wertigkeit 0x40

Achtung! Es müssen immer beide Bits gesetzt werden!

5. Antwortliste

Datagrammformat gesendet vom UTA (Antwort zum PC)																		
	hex	header	id	command	value1		value2								check sum	Hinweise		
Name	SetVoltage	1F	AA55	xx	1F	00	00	00	00	00	00	00	00	00	00	xxxx	value2 enthält ASCII Zeichen	
	GetVoltage	2F	AA55	xx	2F	xx	xx	00	00	00	00	00	00	00	00	xxxx		
	SetCurrent	3F	AA55	xx	3F	00	00	00	00	00	00	00	00	00	00	xxxx		
	GetCurrent	4F	AA55	xx	4F	xx	xx	00	00	00	00	00	00	00	00	xxxx		
	SetStatus	5F	AA55	xx	5F	00	00	00	00	00	00	00	00	00	00	xxxx		
	GetStatus	6F	AA55	xx	6F	xx	xx	00	00	00	00	00	00	00	00	xxxx		
	SetIDString	7F	AA55	xx	7F	00	00	00	00	00	00	00	00	00	00	xxxx		
	GetIDString	8F	AA55	xx	8F	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xxxx		
	Echo	9F	AA55	xx	9F	9F	9F	00	00	00	00	00	00	00	00	00		xxxx
	Error	AF	AA55	xx	AF	xx	xx	00	00	00	00	00	00	00	00	00		xxxx
SetIDN	EF	AA55	xx	EF	00	00	00	00	00	00	00	00	00	00	00	xxxx	value1 enthält Fehlercode	
GetIDN	FF	AA55	xx	FF	xx	xx	00	00	00	00	00	00	00	00	00	xxxx		
GetMaxValues	48	AA55	xx	48	aa	aa	bb	bb	cc	cc	dd	dd	00	00	00	xxxx		
SetUserText	C8	AA55	xx	C8	00	00	00	00	00	00	00	00	00	00	00	00	xxxx	siehe Legende
GetUserText	D8	AA55	xx	D8	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xx	xxxx	
Legende:																		
a - voltage out (Usoll)																		
b - current out (Isoll)																		
c - voltage in (Umon)																		
d - current in (Imon)																		
xx - variiert																		
Beachten:																		
GetIDN wird an die ID Nummer 00 "gesendet" weil der Befehl nicht an eine bestimmte ID gesendet werden kann, bevor sie nicht vergeben wurde.																		
Error (AF) ist nur eine Antwort und kann bei jedem Befehl statt der erwarteten kommen.																		
Bedeutung der Bits im Byte, das mit GetStatus zurückgegeben wird:																		
Bit							7	6	5	4	3	2	1	0				
							Stbv	Rem	CC/CV	Error	CP	OT	OVP					

6. Erklärung der Befehle

Die Wertebereiche der ersten 4 Befehle sind begrenzt durch die sogenannten MaxValues. Die MaxValues sind ermittelte Werte, die 10,000V an den Ausgängen Usoll, Isoll und den Eingängen Umon und Imon entsprechen. Die Werte sind bei jedem UTA verschieden, im UTA gespeichert und müssen zur Verwendung ausgelesen und in Variablen zwischengespeichert werden. Siehe Befehl GetMaxValues.

(1F) SetVoltage - setzt die Ausgangsspannung des Ausganges Usoll des UTA

Beispiel: wenn der max. Wert für 10,000V an Usoll 0xDCFF wäre, dann würde der Befehl mit 0xDCFF als Argument in value1 genau 10,000V erzeugen,

(2F) GetVoltage - gibt einen Hexwert zurück, der der am Eingang Umon angelegten Spannung entspricht

Wichtig: der max. Wert (MaxValue) für diesen Eingang stellt den Hexwert dar, der beim Anlegen von 10,000V zurückgegeben wird. Bei 5,000V würde dann genau die Hälfte dieses Wertes zurückgegeben werden.

(3F) SetCurrent - analog wie (1F), aber für den Ausgang Isoll

(4F) GetCurrent - analog wie (2F), aber für den Eingang Imon

(5F) SetStatus - setzt den Zustand der Schaltausgänge Standby und Remote des UTA

Diese Ausgänge sind bezogen auf die Zustände von Bit 6 (REM) und Bit 7 (Stby) des Statusbytes (andere Bits werden ignoriert) und müssen immer zusammen gesetzt werden. Siehe oben bei Befehlsliste.

Achtung! Das Setzen dieser Bits bezieht sich auf den logischen Zustand der zugehörigen Ausgangspins am UTA12. Wenn also Bit 7 (Stby) gesetzt ist, dann ist Pin Stby aktiviert und LOW. Das Gleiche gilt für REM. Prüfen Sie daher, ob das zu steuernde Gerät bei LOW, also Stby=1, den Ausgang einschaltet oder ausschaltet. Ggf. muß die Logik der Bits dann invertiert angewendet werden.

(6F) GetStatus – gibt ein Byte zurück, dessen Bitmuster enthält den Zustand der Statussignaleingänge widerspiegelt

Bit	7	6	5	4	3	2	1	0
	STBY	REM	CC/CV	Error		CP	OT	OVP

Hinweis: der Zustand der Statusausgänge Standby und Remote kann hiermit abgefragt werden.

(7F) SetIDString - setzt den "ID String" (ASCII-Zeichenkette)

Der String wird im UTA gespeichert und kann bei Bedarf ausgelesen werden.

Der ID String wird normalerweise benutzt, um ein UTA zu identifizieren, das an ein bestimmtes Netzgerät angeschlossen ist, indem er den Namen/die Bezeichnung des Netzgerätes enthält.

(8F) GetIDString - analog wie (7F), gibt den ID String zurück, falls vorhanden

(9F) Echo - „pingt“ das UTA an

Dient dazu festzustellen, ob ein UTA noch antwortet bzw. ob es vorhanden ist, also richtig vom Treiber eingebunden.

(EF) SetIDN - setzt die ID Nummer des UTA

Die ID Nummer wird benutzt, um ein Befehl an ein bestimmtes UTA zu schicken. Die ID Nummer des UTA und die im Datagramm verwendete müssen übereinstimmen.

(FF) GetIDN - analog zu (EF), gibt die ID Nummer zurück

(48) GetMaxValues - gibt die im UTA werkseitig gespeicherten Maximumwerte zurück, die 10,000V entsprechen

Diese Werte müssen nach dem Öffnen des Gerätes wenigstens einmal ausgelesen und Variablen in der Software zur weiteren Verwendung zugewiesen werden, da sonst die Sollwerte nicht die gewünschten Spannungen erzeugen.

(C8) SetUserText - schreibt einen frei wählbaren Usertext (max. 18 ASCII-Zeichen) in den internen Speicher

Der Text wird solange gespeichert bis er geändert wird. Im Allgemeinen dient dieser Text dazu, um z.B. Standortinformationen für das UTA festzuhalten oder andere, dem UTA zugewiesene, Informationen zu speichern.

(D8) GetUserText - analog zu (C8), gibt den Usertext zurück

7. Beispiele

Dies sind einige Beispiele für UTA Befehle (die benutzte IDN ist immer 1, Checksummenbildung siehe Seite 4):

Um die am Eingang *Umon* angelegte Spannung zu messen und den Meßwert zu erhalten:

Befehl: AA 55 01 2F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FE D0
 Beispielanwort: AA 55 01 2F 2F DE 00 00 00 00 00 00 00 00 00 00 00 00 FD C3

Um den Usertext '*Lageraum 1*' (ohne '"') zu schreiben:

Befehl: AA 55 01 C8 4C 61 67 65 72 72 61 75 6D 20 31 00 00 00 00 00 00 FA 46
 Antwort: AA 55 01 C8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FD C3

Um das Interface zu resettten (das USB-Gerät an sich wird nicht resettet/geschlossen!):

Befehl: AA 55 01 E8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FE 17
 Antwort: *keine Antwort, weil ein "Reset" keine definierte Antwort erzeugt*

Um die "[MaxValues](#)" auszulesen:

Befehl: AA 55 01 48 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FE B7
 Beispielanwort: AA 55 01 48 FC 05 FC 78 3D E4 3D 3C 00 00 00 00 00 00 00 00 FA A8