
# Instructions & overview of the command list of the UTA interface

## Index




State: 20/01/2011, M.Staberock

## 1. Prologue

**In order to access and control the UTA interface in environments and applications created with languages like C, C++, Visual Basic and other ones, only two things are needed: a COM port and the communication protocol, which is described below. The COM port is generated as virtual COM port (VCP) by the USB driver. Hence any UTA12 interface will display two devices in the Windows device manager, a "USB Serial converter" and a "USB Serial port". After adding a new UTA12 interface to a PC, it is advised to check the Windows device manager to show these two devices.**
**Contrary to what is described in the UTA12 user's guide when using the UTA12 Control software, you must not deactivate the VCP in this case.**

***Important! At the USB side of the UTA interface is a chip working which provides a USB-to-RS232 conversion. Some of the function calls therefore only relate to Open, Close etc. of the USB chip, not to transfer commands to the UTA interface.***

## 2. Basics

**The communication with the UTA interface is done with telegram in hexadecimal form. A command cycle typically consists of a command transfer and a reply transfer.**
**Hence, the interface works as a slave and waits for commands. Commands and answers are always 24 Bytes long, their structure will be explained in detail later on.**
**In case the driver has been installed successfully, the COM port is ready to be accessed.**

**If the device is opened, telegrams can be sent and read from/to the UTA interface according to the protocol described below.**

***Before this, the COM port should be configured. This has to be done everytime it is opened.***
***Settings: 57600Bd, 1 stop bit, 1 start bit, no parity.***

**If the device is not used anymore, it should be closed with the function FT_Close.**

## 3. Structure of the 24 Bytes sized command datagram

***If programming is in C we recommend to use a structure like this to build the datagram:***
***(this is just an example)***

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


State: 20/01/2011, M.Staberock

**The syntax of the data types varies from language to language and has to be adapted acoordingly.**
**Explanation of the elements:**

**uint uiHeader** = unsigned integer (2 Bytes), header word, always 0xAA55

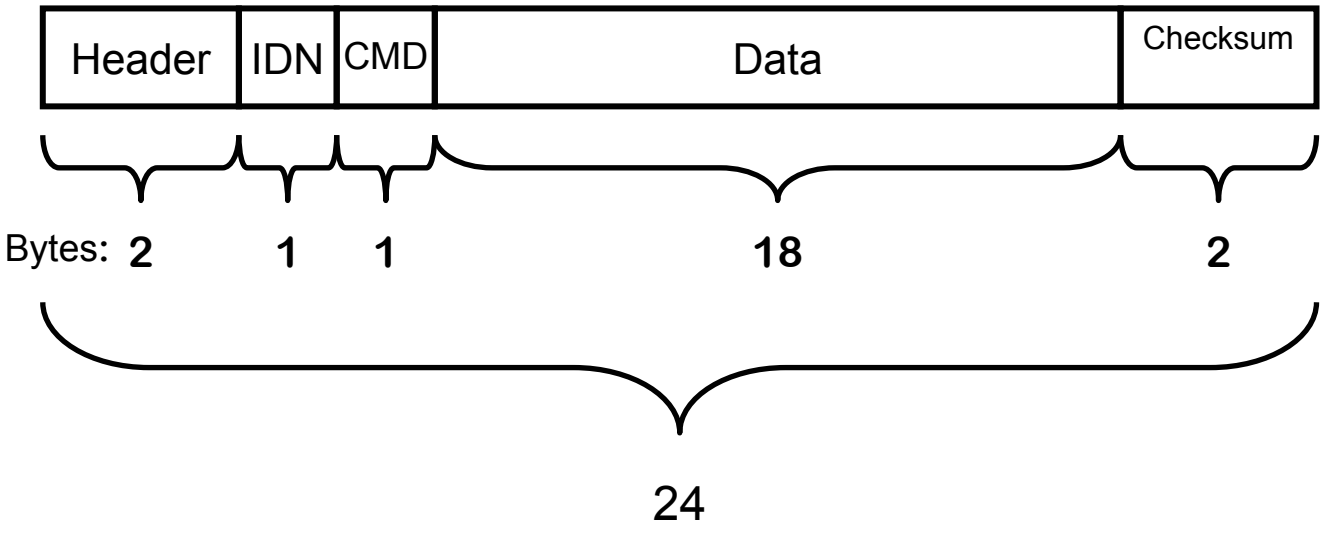
**uchar ucIDnumber** = unsigned char (1 Byte), ID number of the UTA
**The ID number should be set at least once, it is stored inside the UTA and has to be put into the structure, else any datagram containing a different ID than the one assigned to the interface will be ignored.**

**uchar ucCommand** = unsigned char (1 Byte), contains the command (hex byte, see table)

**char ucDatafield[18]** = char (18 Bytes), contains parameters or ASCII characters for the related command (see table)

**uint uiChecksum** = unsigned integer (2 Bytes), checksum of the first 22 Byte of the datagram, serves for a simple security of the data transfer and is calculated as follows:
**build a total sum of the first 22 bytes (results in a word), complement this and put this word as the checksum into the last two bytes of the datagram.**

**Figure of the datagram:**





State: 20/01/2011, M.Staberock

## 4. Command list

| datagram format sent by PC (command to UTA) | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| name | hex | header | id | command | value1 | | value2 | | | | | | | | | | | | | | | | check sum | notes |
| SetVoltage | 1F | AA55 | xx | 1F | xx | xx | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | xxxx | value1 ranges from 0x0000 to 0xffff |
| GetVoltage | 2F | AA55 | xx | 2F | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | xxxx | |
| SetCurrent | 3F | AA55 | xx | 3F | xx | xx | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | xxxx | value1 ranges from 0x0000 to 0xffff |
| GetCurrent | 4F | AA55 | xx | 4F | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | xxxx | |
| SetStatus | 5F | AA55 | xx | 5F | xx | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | xxxx | value1 ranges from 0x00 to 0xff |
| GetStatus | 6F | AA55 | xx | 6F | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 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 | xx | xx | xx | xx | xx | xx | xx | xx | xxxx | value1 & value2 MUST contain characters |
| GetIDString | 8F | AA55 | xx | 8F | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 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 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | xxxx | without ID |
| SetIDN | EF | AA55 | 00 | EF | xx | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | xxxx | value1 ranges from 0x00 to 0xff |
| GetIDN | FF | AA55 | 00 | FF | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | xxxx | without ID |
| GetMaxValues | 48 | AA55 | xx | 48 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | xxxx | |
| SetUserText | C8 | AA55 | xx | C8 | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xxxx | value1 & value2 MUST contain characters |
| GetUserText | D8 | AA55 | xx | D8 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 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 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | xxxx | does not cause a reply, only resets the UTA |

**Error codes of the reply AF:**

| Hex | Meaning |
|---|---|
| 0x0001: | check sum mismatch |
| 0x0002: | id string empty |
| 0x0003: | header mismatch |
| 0x0004: | eeprom write error |

**Bit assignment for command SetStatus (5F):**
**Bit 7: Stby (output off/on), value 0x80**
**Bit 6: REM (remote control on/off), value 0x40**
**Attention! The bits can not be set seperately!**

## 5. Reply list

| datagram format sent by UTA (answer to PC) | | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| name | hex | header | id | command | value1 | | value2 | | | | | | | | | | | | | | | | check sum | notes |
| SetVoltage | 1F | AA55 | xx | 1F | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | xxxx | |
| GetVoltage | 2F | AA55 | xx | 2F | xx | xx | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | xxxx | |
| SetCurrent | 3F | AA55 | xx | 3F | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 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 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | xxxx | |
| SetStatus | 5F | AA55 | xx | 5F | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | xxxx | |
| GetStatus | 6F | AA55 | xx | 6F | xx | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | xxxx | |
| SetIDString | 7F | AA55 | xx | 7F | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 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 | xx | xx | xx | xx | xx | xx | xx | xx | xxxx | value2 contains characters |
| Echo | 9F | AA55 | xx | 9F | 9F | 9F | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 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 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | xxxx | value1 contains error code |
| SetIDN | EF | AA55 | xx | EF | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | xxxx | |
| GetIDN | FF | AA55 | xx | FF | xx | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 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 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | xxxx | see legend |
| SetUserText | C8 | AA55 | xx | C8 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | 00 | xxxx | |
| GetUserText | D8 | AA55 | xx | D8 | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xx | xxxx | |

***Notes:***

GetIDN is 'broadcasted" to id number 00 since you can can't send this command to a certain id before one is given

Error (AF) is a only an answer, which may be caused by any command (instead of the expected answer)

Meaning of the bits of the returned byte of GetStatus corresponding to the inputs of the UTA

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | Stby | Rem | CC/CV | Error | | CP | OT | OVP |

***legend***

a - voltage out (Uset)
b - current out (Iset)
c - voltage in (Umon)
d - current in (Imon)
xx - various

# 6. Explanation of the commands

*The value ranges of the first 4 commands are limited by the so-called MaxValues. These are calibrated values, which correspond to 10.000V at the outputs Uset, Iset and the inputs Umon and Imon.*
*The values are different at every UTA, stored inside the UTA and have to be read out and copied into variables. See command GetMaxValues.*

## (1F) SetVoltage - sets the output voltage of the output Uset of the UTA

Example: if the max. value for 10.000V at Uset is 0xDCFF, the command would produce 10.000V with 0xDCFF as argument in value1.

## (2F) GetVoltage - returns a hex value corresponding to the voltage provided at input Umon

Important: the max. value for this input depicts the hex value, which is returned if 10.000V are put in.
At 5.000V exactly the half of this value would be returned.

## (3F) SetCurrent - analogous to (1F), but for output Iset (channel

## (4F) GetCurrent - analogous to (2F), but for input Imon

## (5F) SetStatus - sets the state of the outputs Standby and Remote of the UTA

The outputs correspond to bit 6 and 7 of the status byte (other bits are ignored) and can not be set seperately.
See command list.
Attention! Setting of those bits is related to the logical state of the related output pins at the UTA12. If bit 7 (Stby) is set to 1, then pin Stby is activated and LOW. The same applies for REM. Thus you need to check, if the device switches the output off with LOW (Stby=1) or with HIGH. Depending on the given situation, the logic of the bits might be inverted.

## (6F) GetStatus - returns a byte which contains a bit pattern displaying the state of the status signal inputs

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

A High (logical 1) means the particular input/output is set/activated.
Note, that the state of the status outputs Standby and Remote can be queried here.

## (7F) SetIDString - sets the ID string of the UTA

The string is stored inside the UTA and can be read out on demand.
Normally used to identify a UTA connected to a certain power supply by setting the ID string to the unit's label/type name.

## (8F) GetIDString - analogous to (7F), returns the ID string

## (9F) Echo - "pings" the UTA in expection for an instant reply

Simply used to check if the device is still responding.

## (EF) SetIDN - sets the ID number of the device

The ID number is used to assign a command to a certain, properly set up device. The ID number of the UTA and the one in the datagram have to be identical.

## (FF) GetIDN - analogous to (EF), returns the ID number

## (48) GetMaxValues - returns the maximum values, stored by default inside the UTA, that relate to 10.000V

These values have to be read at least once after opening the device and assigned to variables for further use in the application. Else, the nominal values will not produce the correct voltages.

## (C8) SetUserText - stores a freely choseable user text (max. 18 ASCII chars) inside the UTA

The text is stored as long as it is not overwritten. In common, it is used to store additional information about location or utilisation of the UTA.

## (D8) GetUserText - analogous to (C8), returns the user text

## 7. Examples

**These are some examples for the UTA commands (used IDN is always 1, check sum generation see page 4):**

**In order to get the voltage supplied at port *Umon* use this:**

| | |
|---|---|
| **command:** | **AA 55 01 2F 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FE D0** |
| **answer:** | **AA 55 01 2F 2F DE 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FD C3** |

**In order to set the user text to '*supply room 1*' (without the ') it has to be:**

| | |
|---|---|
| **command:** | **AA 55 01 C8 73 75 70 70 6C 79 20 72 6F 6F 6D 20 31 00 00 00 00 00 F9 5C** |
| **answer:** | **AA 55 01 C8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FD C3** |

**In order to reset the interface use this (the USB device itself is not resetted/closed!):**

| | |
|---|---|
| **command:** | **AA 55 01 E8 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FE 17** |
| **answer:** | ***no answer, because "Reset" can not reply a definite answer*** |

**In order to read out the "MaxValues":**

| | |
|---|---|
| **command:** | **AA 55 01 48 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 FE B7** |
| **answer:** | **AA 55 01 48 FC 05 FC 78 3D E4 3D 3C 00 00 00 00 00 00 00 00 00 FA A8** |