

# Ultrabeam RCU-06 – USB port description



## Abstract

The Ultrabeam RCU-06 is an electronic controller used in a variety of Ultrabeam antennas. The main task of the controller is to tune the antenna by moving the (radiating) elements, based on the frequency inputted manually by the user, obtained via serial communication using the built-in CAT interface to usual RF transceivers, or injected into the USB port by (typically) a computer.

This document describes the serial communications on the USB port of the Ultrabeam RCU-06 controller.

## Introduction

Via the USB port, the Ultrabeam RCU06 controller can communicate operating parameters (such as active radio band, frequency, direction), and also accepts commands to which it reacts.

The USB is treated as a serial line, and the same happens on the computer side: the USB port of the computer must be used as a serial port, with the typical communication parameters of an RS-232 interface. Then, the computer sends “packets” (described later), to which the Ultrabeam replies and, for some kind of packet, also some more work can be done by the controller, such as moving the elements of the antenna.

## About the USB interface

The USB interface was not designed to be highly reliable; compared to a normal RS-232 serial line, the USB line can be much faster but it is more sensitive to electrical noise. This means that, to obtain reliable communications, some precautions must be taken:

1. Use high quality and well shielded cables, as short as possible. Cheap cables usually are poorly shielded and so may easily pickup or radiate noise.
2. Avoid to run the USB cable near (electrically) noisy equipments.
3. Ensure that, on the computer side (master), good hardware, good operating system and good software drivers are used. The master side of a USB connection is in charge to recover from errors so, if the master is not performing well, disconnections can happen.

The protocol described here is **also available on the DATA IN connector** of the Ultrabeam controller, with the same format and baud rate, when the RS-232 “DTR” signal is asserted. The presence of that signal on **PIN 4 of DATA IN connector** disables the normal RTX behavior (CIV, CAT etc.) and enables the RCU-06 native protocol.

## Packet format

The used protocol has a fixed baud rate of 19200, 8 bit data length, no parity, and 1 (one) stop-bit. No other signal are used - no handshake like DTR, CTS etc. must be set or expected (except DTR must be asserted if using the **DATA IN** serial line).

The controller acts as a server – it waits for incoming data and sends back replies.

All the data traveling on the serial line is organized in packets, having the following format:

## Ultrabeam RCU-06 – USB port description

| Name | Length (bytes) | Description                                         |
|------|----------------|-----------------------------------------------------|
| STX  | 1              | fixed byte F5H (hex), 245 decimal                   |
| SEQ  | 1              | sequence number – one byte, from 0 to 127 decimal.  |
|      |                | <i>Values greater than 127 have special meaning</i> |
| COM  | 1              | command to perform, or response code from UB        |
| DAT  | 0 to 118       | bytes of data, comprising DLE escapes (see below)   |
| CHK  | 1              | checksum, see below                                 |
| ETX  | 1              | fixed byte FAH (hex), 250 decimal                   |

All the bytes between STX and ETX must not be an STX, ETX or DLE: they must be *escaped* with a DLE. The DLE is defined as F6H (hex), 246 decimal, and must precede a data byte when that byte is an STX, ETX, or DLE; then the byte to be transmitted must be stripped off of its MSB bit (*most significant*, the last one, bit number 7), and then sent to the serial line. For example, to send a decimal 245 (corresponding to STX) as a data byte, send a DLE and then a 117 decimal (which is 245 without the 7<sup>th</sup> bit, or 245-128=117). Please note that even SEQ, COM and CHK fields must comply to this requirement.

The correct algorithm to send a packet is as follows:

```
send an STX
send a "quoted" SEQ      (sequence number)
send a "quoted" COM      (command code)
send up to 59 "quoted" bytes
send the "quoted" CHK    (checksum)
send an ETX
```

The word "quoted" means: if the byte to send is equal to STX, ETX or DLE, then:

- send a DLE
- subtract 128 (decimal) from the datum, or do a logic AND with 127
- send the modified datum

If the datum is any other character, instead, send it as is.

About this "quoting" mechanism, note the behavior of the receiver side:

- 1) An incoming STX always starts a new packet
- 2) An incoming ETX always ends a packet, and passes it to further analysis
- 3) An incoming DLE always sets the MSB of the next (following) incoming byte

### How to calculate the checksum (CHK)

The checksum is a single byte transmitted at the end of the data, and validates the packet. To calculate the checksum for a packet, a byte variable has to be allocated, and initialized to 55H (hex). Then for every data byte between STX and ETX, the following calculus must be done:

- set the checksum equal to the bit-wise XOR of the checksum and the datum
- increment the checksum (note that after incrementing 255, results 0)

WiMo Antennen und Elektronik GmbH, Am Gäxwald 14, 76863 Herxheim, Germany  
www.ultrabeam.it www.wimo.com info@wimo.com

## Ultrabeam RCU-06 – USB port description

Now, the final algorithm to send a packet to the Ultrabeam controller reads like this:

```
checksum := 55H
send STX
send SEQ:
    checksum := checksum XOR SEQ
    checksum := checksum + 1
    if SEQ=STX or SEQ=ETX or SEQ=DLE then "quote it" else "send it as is"
send COM:
    checksum := checksum XOR COM
    checksum := checksum + 1
    if SEQ=STX or ... (like before)
send up to 59 "quoted" bytes, with check-summing and quoting like above
send the "quoted" checksum:
    *** do NOT update checksum! ***
    if checksum=STX or checksum=ETX or ... (like before)
send an ETX
```

If the packet sent to the controller complies with the following requirements:

- begins with STX and ends with ETX
- contains at least three bytes: SEQ, COM, CHK
- the checksum (last byte before ETX) is correct

then the Ultrabeam will reply with a packet. If one of the above is not respected, the Ultrabeam will not reply. A missing STX makes the controller not aware of incoming data; a missing ETX will make the Ultrabeam wait for more data; a packet too short (not having at least SEQ and COM) can not be processed; a packet too long will be discarded; an incorrect checksum implies that a transmission error has occurred.

To receive a packet from the controller, the inverse procedure can be used; normally an interrupt-driven or event-driven approach is used. A method is described here:

*Allocate the following variables:*

|           |                                                                  |
|-----------|------------------------------------------------------------------|
| ub_in_ok  | <i>a boolean, indicating reception is ongoing (STX received)</i> |
| ub_in_dle | <i>a boolean, indicating a DLE was received</i>                  |
| ub_in_chk | <i>a byte, tracking the checksum</i>                             |
| ub_in_cnt | <i>counter for incoming bytes</i>                                |
| ub_buffer | <i>an array or pointer / somewhere to store data to</i>          |

For every byte received from the serial line, invoke a subroutine to process the datum.

## Ultrabeam RCU-06 – USB port description

The interrupt- or event-driven routine can be implemented like the following:

```
if received byte is STX:
    ub_in_ok := true
    ub_in_cnt := 0
    ub_in_dle := false
    ub_in_chk := 55H
    exit

if received byte is ETX:
    if ub_in_chk=1 and ub_in_ok=true and ub_in_cnt>2:
        // a correct packet has been received – post it to the main program
        ...
    // anyway disable reception
    ub_in_ok := false
    exit

in any other case:
    if ub_in_ok<>true:
        // no STX was received before - ignore byte
        exit

    if datum=DLE:
        ub_in_dle := true
        exit

    // process and store this received byte
    if ub_in_dle=true:
        datum := datum or 128
    ub_in_dle := false;
    ub_in_chk := ub_in_chk xor datum
    ub_in_chk := ub_in_chk + 1
    ub_in_cnt := ub_in_cnt + 1
    if ub_in_cnt > THE_MAXIMUM_ALLOWED
        // packet too long, discard it
        ub_in_ok := false
        exit

    // append this byte to the buffer
    ub_buffer[ub_in_cnt] = datum
    exit
```

The method described above is just one among other possible methods to implement the receiving software. In particular note that `THE_MAXIMUM_ALLOWED` does not need to be the same as Ultrabeam – in order to be compatible with future releases, this can be raised to 128 or 256 bytes.

### Sequence number and Ultrabeam replies

When the controller receives a correct packet, it issues the requested operation and then sends back a reply packet. If the originating (request) packet does not reach the controller, or if the reply packet does not reach the computer, the computer sees always the same thing (i.e. no response), without

WiMo Antennen und Elektronik GmbH, Am Gäxwald 14, 76863 Herxheim, Germany  
www.ultrabeam.it www.wimo.com info@wimo.com

## Ultrabeam RCU-06 – USB port description

knowing which packet got lost, the request or the reply. In this case, the computer should issue a retry. Ultrabeam replies are normally very fast, so a timeout of a second, or little more, should be sufficient. When requesting operations which write in the controller parameters, however, this time can, rarely, rise up to 20 or 30 seconds; this happens when the entire flash has been filled, and a brand new copy has to be generated. Giving this, a right strategy could be to have a timeout of one or two seconds for the first two or three tries, and then use a timeout of five or ten second for further two or three tries.

The sequence number in a request to the controller serves as a higher level protocol check. In the normal form, it is simply a number from 0 to 127; the Ultrabeam controller replies to a request using the same sequence number. Typically, the computer communicating with the controller increments this number on every new request, but this is not mandatory - it can be used if the communicating task uses some form of queuing.

If the MSB (*most significant bit*, bit number seven) is set, hence the sequence number is greater than 127, the controller behaves differently – it issues the command only if the sequence number is different than the previous one. This is intended to avoid to issue the same command twice, in the case that the reply packet is lost. This is called a “non repetition” command. If the controller receives a packet having sequence greater than 127, and this sequence number is the same as the previous one, then it replies with a simple “OK” (see later), without issuing the command again. If the command involves a reply containing some data, that data is not transmitted again.

### Command packets

As stated before, a communication is started by the computer, which sends a packet. The Ultrabeam replies with a packet having the same sequence number, and the field COM set to one of four possible values:

|        |   |                                                          |
|--------|---|----------------------------------------------------------|
| UB_OK  | 0 | normal execution, no errors                              |
| UB_BAD | 1 | invalid command – nothing done                           |
| UB_PAR | 2 | bad parameters or argument for the command, nothing done |
| UB_ERR | 3 | error while executing command                            |

Many command replies have some more data following the COM byte.

What follows is a list of commands, identified by the COM byte being 1, 2, 3, and so on.

# Ultrabeam RCU-06 – USB port description

## Command 1 – General status query

This command has no data – it is formed simply by the COM byte (equal to 1).

The Ultrabeam replies with COM=UB\_OK, and a buffer of data as follows:

byte 1: Firmware version, minor revision  
byte 2: Firmware version, major revision  
byte 3: Current interactive operation:  
    0 = normal operation  
    1 = adjusting of factory presets (not for normal user)  
    2 = adjusting band data (user presets)  
    3 = adjusting user settings (setup / preferences)  
byte 4: Current frequency, low byte  
byte 5: Current frequency, high byte  
    the frequency is expressed in Khz, in binary representation.  
byte 6: Current band, out of eleven. The first band is indicated as 0.  
byte 7: Antenna orientation (lower 4 bits): 0=normal; 1=180°, 2=bi-directional  
byte 8: General flags, part 1 (undocumented bits are reserved):  
    bit 1 = “Off” state (display off, no interaction)  
byte 9: General flags, part 2 (undocumented bits are reserved):  
byte 10: Some motor is moving: bit0=first motor; bit1=second; bit2=third...  
byte 11: Lowest acceptable frequency, in Mhz  
byte 12: Highest acceptable frequency, in Mhz  
Subsequent received bytes are reserved and must be ignored.

## Command 2 – Retract elements

The controller retracts its element, and replies UB\_OK.

## Command 3 – Change frequency

After the COM byte, the command must specify the new frequency to go to, in Khz, with two bytes: first the low order and then the high order. If the frequency requested is reachable, the Ultrabeam will start to move motors and send UB\_OK; otherwise a UB\_PAR will be sent back.

After the two bytes for frequency, an optional third byte can be sent to specify the directional mode:

0=normal direction  
1=180°  
2=bidirectional

Append the direction byte after the two frequency bytes, for a total of three bytes following the command. If the antenna does not support direction, or a wrong value is specified, or a wrong number of bytes is sent, the direction specification is ignored (but the frequency is always honored).

Sending a frequency of 0 Khz, a *retract axes* is performed.

## Ultrabeam RCU-06 – USB port description

### Command 4 – Calibrate axes (from firmware v4.41)

The controller retracts all the axes (elements), in such a way they surely are fully retracted. This is different from simply retract them: in retract operation, the position of an element is assumed to be correct, and the element is retracted by just *slightly more than its current length*. When calibrating instead, the elements get retracted *as if they were fully extended*; this operation stresses the mechanical components of the antenna, and should not be performed if not necessary.

*This command is not implemented in firmware versions prior to 4.41.*

### Command 9 – Read user current band user adjustments

The Ultrabeam replies with 6 words (12 bytes), which are the current lengths of the antenna elements, in millimeters. Every word is split in lo-order and hi-order bytes.

### Command 10 – Read status of progress bar (moving motors)

The Ultrabeam replies with 2 words (4 bytes), which are the total distance to travel, in mm, and its completion status in units/60. If the first word is different than 0, then the antenna is moving; the second word starts with 0, and goes up to 60.

## **Warning**

***The Ultrabeam controller implements other commands which are not described here. Those commands are used by the Ultrabeam factory, and should not be used: an improper use can permanently damage the controller.***

# Ultrabeam RCU-06 – USB port description

## Index of contents

|                                                               |   |
|---------------------------------------------------------------|---|
| Abstract .....                                                | 1 |
| Introduction .....                                            | 1 |
| About the USB interface .....                                 | 1 |
| Packet format .....                                           | 1 |
| How to calculate the checksum (CHK).....                      | 2 |
| Sequence number and Ultrabeam replies .....                   | 4 |
| Command packets.....                                          | 5 |
| Command 1 – General status query.....                         | 6 |
| Command 2 – Retract elements.....                             | 6 |
| Command 3 – Change frequency .....                            | 6 |
| Command 4 – Calibrate axes (from firmware v4.41).....         | 7 |
| Command 9 – Read user current band user adjustments .....     | 7 |
| Command 10 – Read status of progress bar (moving motors)..... | 7 |
| Warning .....                                                 | 7 |

- **The information in this documents is provided “as is”, and believed to be true, but no warranty is provided.**
- **It is forbidden to spread or otherwise divulge the information found here, without express, written permission of WiMo / Ultrabeam.**
- **No responsibility can be taken by WiMo / Ultrabeam for any usage of this document.**

© **WiMo Antennen und Elektronik GmbH**

Am Gäxwald 14  
76863 Herxheim  
Germany  
+49 7276 96680

[www.ultrabeam.it](http://www.ultrabeam.it)  
[www.wimo.com](http://www.wimo.com)  
[info@wimo.com](mailto:info@wimo.com)

WiMo Antennen und Elektronik GmbH, Am Gäxwald 14, 76863 Herxheim, Germany  
[www.ultrabeam.it](http://www.ultrabeam.it) [www.wimo.com](http://www.wimo.com) [info@wimo.com](mailto:info@wimo.com)