Protocol and Data Structure of Uwatec MemoMouse (Windows Version) for Aladin Dive Computers

---

1. Hardware, Baudrate, Bit Order, and Packets

MemoMouse (Windows version) and PC are connected by usual RS232C cable and protocol (9600 baud, 8 bits, No parity and 1 stop bit, No hardware flow control).

Each byte (8 bits) sent by MemoMouse is in reversed bit order, that is, if

```
transmitted bit order by MemoMouse: b_0 b_1 b_2 b_3 b_4 b_5 b_6 b_7
```

then

```
ordinary bit order of PC:           b_7 b_6 b_5 b_4 b_3 b_2 b_1 b_0
```

Thus, the first task for PC side software is to rearrange every byte before proceeding further data processing. Similarly, every byte needed to send to MemoMouse from PC must be in reversed bit order, too. So PC side software must rearrange every byte before sending any bytes.

It is very strange to know that every data from MemoMouse are doublely packeted/checksum'ed in the following way:

```
[Inner packet]
    1st and 2nd bytes:   Data length = [2] * 256 + [1], say N.
    3rd byte       --+
    ...              |   Original data MemoMouse wants to transmit
    N + 2nd byte   --+
    N + 3rd byte:        Data check sum
```

(Data check sum is calculated as the exclusive OR value of all the bytes from the first byte to N + 2nd byte.)
This "inner packet" is divided into smaller "outer packets" before sending to PC.

```
[Outer packet]
    1st byte:            Length included in this packet (1 -- 126), say n.
    2nd byte       --+
    ...              |   Divided data from inner packet
    n + 1st byte   --+
    n + 2nd byte:        Check sum
```

(The check sum is calculated as the exclusive OR value of all the bytes from the first byte

to n + 1st byte in this outer packet.)

For example, when MemoMouse tries to send its ID string "IFV1.00" it is firstly encoded as an inner packet

```
Original data MemoMouse wants to transmit:
                              I    F   V     1    .    0    0
                            0x49 0x46 0x56 0x31 0x2E 0x30 0x30

Inner packet:      0x07 0x00 0x49 0x46 0x56 0x31 0x2E 0x30 0x30 0x41
                   ~~~~~~~~~                                    ~~~~
                   length                                       check sum
```

Then this is encoded again to an outer packet as

```
Outer packet: 0x0A 0x07 0x00 0x49 0x46 0x56 0x31 0x2E 0x30 0x30 0x41 0x0A
              ~~~~                                                   ~~~~
              length                                                 check sum
```

and sent to PC with a bit-reversed string

```
              0x50 0xE0 0x00 0x92 0x62 0x6A 0x8C 0x74 0x0C 0x0C 0x82 0x50
```

---

2. Negotiation for Data Transfer

When the data transfer icon in DataTrak for Windows is clicked, the following negotiation is done:

```
 PC                                      MemoMouse
============================================================
 DTR = 1 ('space')
 RTS = 0 ('mark')
                          ---->
                                        Switch ON (Green LED is lit)
 (some delay)
 Send NAK (0x15) (See Note 1)
                          ---->
 (some delay)
 Send NAK (0x15)          ---->
 (repeat 4 or 5 times)
 ...
                                        Send ID string "IFV1.00"
                                               (See Note 2)
                          <----
 Send ACK (0x06)

 Send 0x07  (outer packet size)

 Send 0x05 -+- (inner packet size)
 Send 0x00 -+

 Send U (0x55)         (See Note 3)
 Send "time" (4 bytes) (See Note 4)
                       (See Note 5)
```

```
  Send outer check sum byte
                            ---->
                                        Send ACK (0x06)

                                        Send one outer packet consist of diving data.
                                        (Outer packet size, including the header and
                                         the check sum, is 128 bytes)
                                        (See Section 3 for more detail)
                            <----
  Send ACK (0x06)
                            ---->
                                        Send one outer packet consist of diving data.
                                        (Outer packet size, including the header and
                                         the check sum, is 128 bytes)
                            <----
  Send ACK (0x06)
                            ---->
  ...
  (repeat as many times as the whole data are transmitted)
  ...
                                        Send one outer packet consist of diving data.
                                        (Outer packet size, including the header and
                                         the check sum, is no more than 128 bytes)
                            <----
  Send ACK (0x06)
                            ---->
================================================================
```

- Note 1: NAK is sent as 0xA8 because every byte in the communication is bit-reversed.
- Note 2: As we already see, the ID string "IFV1.00" is sent as the sequence

  `0x50 0xE0 0x00 0x92 0x62 0x6A 0x8C 0x74 0x0C 0x0C 0x82 0x50`

- Note 3: This byte is the command byte for MemoMouse:
  - command 0x47 (ASCII 'G') : Configure Aladin.
  - command 0x4c (ASCII 'L') : Load configuration of Aladin.
  - command 0x55 (ASCII 'U') : Send dive data.
  - command 0xe7 : Wake up Aladin.

  I'm not, however, interested in configuring Aladin through MemoMouse, the descriptions of the commands other than 'U', "Send dive data", are omitted in this document.
- Note 4: DataTrak wants to receive all the newer data than this "time". Time is represented as 4 bytes of Aladin's time (the time is basically from 00:00 1 Jan, 1994 GMT in unit of 0.5 seconds but may contain some discrepancy) = [1st byte] + [2nd byte]*2^8 + [3rd byte]*2^16 + [4th byte]*2^24. (Remark: Byte order is in reverse to the original Aladin timestamp). The value of "time" depends on whether "Only newer" or "All dives" is selected in the DataTrak dialog.
- Note 5: Strangely, in this point there is no inner packet check sum.

---

3. Structure of Data Transmitted by MemoMouse

As it has been already mentioned in Section 1, original data that MemoMouse wants to transmit is doublely packeted (and bit reversed) before sending. This section is devoted to explain the original data structure before encoding (i.e., the data structure before adding any trailing check sums or data length headers).

MemoMouse tries to transmit all the newer diving data than the time DataTrak requested. The original data structure is:

```
Offset       Description
=============================================================
  0          Constant U (0x55).
  1--4       Current time of Aladin (MemoMouse)
             = [1] + [2] * 2^8 + [3] * 2^16 + [4] * 2^24.
             (The time is basically from 00:00 1 Jan, 1994 GMT in unit of
              0.5 seconds but may contain some discrepancy)
             (Remark: Byte order is in reverse to the original Aladin timestamp)
  ------start of a diving record-----
  5--7      Serial ID of Aladin with which this dive was done
             = [5] * 2^16 + [6] * 2^8 + [7].
  8         Type code of Aladin with which this dive was done.
  9--20     Logbook data (See Note 1).
             9:    bit 7 -- high place diving flag (higher bit),
                   bit 6                          (lower bit).
                   bit 5 -- SOS mode.
                   bit 4 -- work too hard (Air series only).
                   bit 3 -- decompression violation.
                   bit 2 -- figure of hundreds of bottom time.
                   bit 1 -- repeated diving.
                   bit 0 -- ascent warning too long.
             10:   Bottom time (BCD).
             11-12: Maximum depth.
             13-14: Surface time (BCD).
             15:   Air consumption (Air series only).
             16-19: Entry time
                   = [16] + [17] * 2^8 + [18] * 2^16 + [19] * 2^24.
                   (The time is basically from 00:00 1 Jan, 1994 GMT in unit of
                    0.5 seconds but may contain some discrepancy.)
                   (Remark: Byte order is in reverse to the original Aladin
                    timestamp.)
             20:   Water temperature.
  21--22     Length of depth profile of this dive (See Note 2)
             = [21] + [22] * 2^8.
  23--       Depth profile data. (See Note 3)
  -------end of a diving record--------------
  (repeat the above as many dives that MemoMouse has.) (See Note 4)
=============================================================
```

- Note 1: These 12 bytes of logbook data are the same as the data received from Aladin, except the byte order of the four bytes for entry time.
- Note 2: If profile data are not available for this dive then the length is set to zero.
- Note 3: Depth profile data are the same as the data received from Aladin with the header byte 0xff omitted.

- Note 4: MemoMouse sends all the diving data newer than requested in the following order:

```
Oldest diving data
Second oldest diving data
...
Newest diving data
Newest diving data
Second newest diving data
...
Oldest diving data.
```

Yes, this is very nasty; MemoMouse sends all the data twice in old-new and new-old orders. I don't know the reason why MemoMouse do like this...