SuperChrono Pro Bluetooth Communication and memory description

Method
The methods used for communicating with SuperChrono Pro, from now on referred to as the device, is standard Bluetooth SPP serial port.
Pair the devices using standard procedures with the code provided from the device marking on the device itself.
Turn the device on and push the bluetooth button on the device to put the device in bluetooth mode. The display of the device will show a small RF symbol, when activated. Connection is established interpreting the bluetooth as an SPP device. The communication device communicate at 115200bps 8b 1st, hardware flow control activated.

Protocol
After pairing the device as above, continue with communication procedures as follows.

| | |
|---|---|
| 1. Send out Following | -> 'C' 'O' 'M' |
| 2. Wait for loop back from microprocessor. (About 500 millisec) | <- 'C' |
| 3. Send out Following | -> 'C' 'O' 'M' |
| 4. Wait for loop back from microprocessor. (About 500 millisec) | <- 'C' |

The device is now considered connected and will send live data.

The device sends data string live data in the following format. <- '0' '4' '8' '3' '4' '5' '6' (sample)

The device will send this string as soon as new information is available. The string is divided into following sections;

| 0 | 4 | 8 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| 1=New hit detected<br>0=No new hit | Voltage first (0-9) | Voltage second(0-9) | Speed first (0-9) | Speed second(0-9) | Speed third(0-9) | Speed fourth(0-9) |

So basicly a string is sent out if voltage change or if new hit is detected. Application is listening for this data and update GUI accordingly.
The sample above gives No new hit, Voltage 4.8V, Speed 3456m/s

To be able to have fully control of the device its also possible to send commands to the device.
The different possible commands are as follows;

Put data to device memory
---
'Q' => Command prefix.
'P' => Request to put data into device memory.
'C' <= Receive Ackn.
Send data 10kB hit information all hits. 2500positions, every hit is 4 Bytes. Send 4 Bytes at a time. Like the following;
'HIT1' => (4 Bytes) And wait for Ackn. (1234 would be sent like '1' '2' '3' '4' characters)
'C' <= Receive Ackn.
'HIT2' => (4 Bytes) And wait for Ackn.
'C' <= Receive Ackn.
'HITn => (4 Bytes) And wait for Ackn.
'C' <= Receive Ackn.
Until HIT2500 is reached.
'X' <= Receive Ackn. Data received.

Receive data from device memory
---
'Q' => Command prefix.
'R' => Request to receive data from device memory.
'C' <= Receive Ackn.
Receive data 10kB hit information all hits. 2500positions, every hit is 4 Bytes. Received 4 Bytes at a time. Like the following;
'HIT1' <= (4 Bytes) And wait for Ackn. (1234 would be sent like '1' '2' '3' '4' characters)
'C' => Receive Ackn.
'HIT2' <= (4 Bytes) And wait for Ackn.
'C' => Send Ackn.
'HITn <= (4 Bytes) And wait for Ackn.
'C' => Send Ackn.
Until HIT2500 is reached.
'X' => Send Ackn. Data received.

Clear device memory

---

'Q' => Command prefix.
'D' => Force data
'C' <= Receive Ackn.
Force device to delete memory.
'X' <= Ackn. Memory deleted

Force data string from the device

---

'Q' => Command prefix.
'F' => Force data
'C' <= Receive Ackn.
Force device to send data string.
'DATA' <= (7 Bytes) Receive Data string, interpreted as live data in sample above.
'X' <= Ackn. Data string sent.

Total shots registered by the device

---

'Q' => Command prefix.
'T' => Force data
Device sends out number of shots detected.
'TOTAL' <= (7 Bytes) Receive Data string, format '9' '8' '1' '2' '3' '4' '5' -> 12345 hits total registred by device

Enter bootloader

---

'Q' => Command prefix.
'B' => Bootloader request
'C' <= Receive Ackn.
'O' => Bootloader request
'C' <= Receive Ackn.
Device goes into bootloader mode/complete reset.

Device Get Firmware Version

---

'Q' => Command prefix.
'I' => Get version number
Device sends out firmware version number
'Ver' <= (7 Bytes) Receive Data string, format '9' '7' '0' '0' '0' '0' '6' -> Firmware version is 6

Device cancel connection (Bluetooth mode is deactivated by user..)

---

'9' <= Dummy data.
'9' <= Dummy data.
'9' <= Dummy data.
'9' <= Dummy data.
'9' <= Dummy data.
'9' <= Dummy data.
'9' <= Dummy data.
Device drops connection.

These are the commands available. But because certain factory calibration modes and firmware upgrade commands, there will be more commands available. And it might also happened that the device sends out data which has to be neglected by the software if not recognised. The device also utilizes a time-out delay on all communication of 5 sec. This means that if the expected data don't arrive within 5 seconds the device will jump out of the procedure.

Memory description of the device.
The device contains 50 strings with 50 hits respectively. They are all sent over in one stream of data as stated above. To maintain the structure of the data when received, the application is supposed to divide the data into these 50 strings and 50 shots per string upon retrieval.
The memory data sends over the data of all positions in memory. That means that positions with no value will appear as 0 or zero, ('0000') 4Bytes.

Therefore the application software must handle those positions accordingly. A recommended layout would exclude these hits from the presentation in the software GUI.

Description for developer of firmware downloader.
The device firmware mode can be activated in two different matters. One is to powercycle the device, that is remove a battery and reposition it. The other one is by software. When connecting after a powercycle, wait until the display lights up, push connect and upgrade firmware via script. When connecting from firmware the device need to have the data in a string, that is the script must include the commands to switch to bootmode from regular mode. Its not supposed to disconnect/reconnect when doing this, just jump from regular mode. A good advice could be to rearrange the script to contain a check if a port is allready open or not. When jumping from regular mode the port will always be open, therefor it does not need to be reinitialized and shall not be due to bluetooth device sending corrupt information during initialization.