A picture containing night sky

Description automatically generated

*Chris Footner & Jerry Clark*

***WaterRower (UK) Ltd***

*Head Office – Unit 4, The Valley Center, Gordon Road, High Wycombe HP13 6EQ*

*R&D Office – Units A7 & A13, Aerodrome Studios, Airfield Way, Dorset, BH23 3TS*

*Software vs Hardware implementation of the OTF V3*

# Document Revisions:

* 1.0 – Document created 12/03/24
* 1.1 – Added Tablet information 19/03/24
* 1.2 – Added Force Pulley and Light Ring information 11/04/24
* 1.3 – Additional information 16/04/24

Table of Contents

[Document Revisions: 2](#_Toc164259659)

[Overview: 3](#_Toc164259660)

[Hardware: 3](#_Toc164259661)

[Power integration 3](#_Toc164259662)

[LightRing (LR000): 4](#_Toc164259663)

[ForcePulley (FP000): 5](#_Toc164259664)

[S6 USB (C601): 5](#_Toc164259665)

[15.6” RowActive Tablet: 6](#_Toc164259666)

[Software: 6](#_Toc164259667)

[LightRing (LR000) USB Protocol 6](#_Toc164259668)

[ForcePulley (FP000) USB Protocol 13](#_Toc164259669)

[S4/S6 Compatibility Mode 13](#_Toc164259670)

[*Strokes* 15](#_Toc164259671)

[*Rowing Information request packets* 16](#_Toc164259672)

[Force Pulley Data Mode 19](#_Toc164259673)

[Calculated Values Output 19](#_Toc164259674)

[Data output modes 19](#_Toc164259675)

[Method for detecting catch, end of drive, calculating stroke length and energy (force pulley mode only). 20](#_Toc164259676)

[S6 USB (C601) USB Protocol 21](#_Toc164259677)

[Supported Metrics 22](#_Toc164259678)

# Overview:

This document will outline the hardware of the WaterRower OTF V3 machine and how it will relate to the App within an Android Tablet on the machine, including the USB or Bluetooth Protocols.

# Hardware:

On the V3 there are 5 new main features that will have some interaction with the App side on the tablet:

## Power integration

For the V3, there will be a new DC chain power integration, using a 24V source. At the heart of this is a USB 2.0 Hub expander, similar to an off the shelf hub, but with custom power routing designed into our one unit. It creates a 12V supply for the tablet, but also has a USB‑A 2.0 Cable going directly to the tablet, this keeps the number of cables going to the tablet limited to just the 2.

Being a standard USB Hub, devices plugged into it appear native to the tablet, which creates com ports for them.

From the USB Hub:

1. A port specifically for the LightRing (LR000), this port is powered via the main power source and will be powered even if the tablet is not attached. Data is still attached through the Hub up to the tablet.
2. The other two ports, located within the forward riser and hidden from view, are the port for the S6 USB (C601) and ForcePulley (FP000). These ports take their power from the tablet’s 5V supply and so power-down with the tablet.

The unit has a RTC fuse on the main input and each output is also fused (whether on the hub or on the connecting device.)

## LightRing (LR000):

Firmware Version: 2.0.2

The Light Ring will be attached to the underside of the water flywheel tank in the centre. The unit will be connected to the USB port on the forward riser leg on the right-hand side of the machine.

This will be able to project custom colours from our 52 individually addressable RGB LEDs, including patterns into the water tank. As this is controlled via the app, any metric can be used to interact with the colours. The brightness/intensity is also adjustable.

A blue round object with a black wire

Description automatically generatedAs this unit is individually powered, separate from the tablet being turned on, the LightRing can be left powered on with a custom colour or pattern. For example, a ‘breathing’ green light to indicate the machine is free to use.

This device should be primarily tethered and communicate over USB.

The ‘always on’ color can be hardcoded to preference.

See the following USB Protocol section for details on the software interface.

## ForcePulley (FP000):

Firmware Version: 1.2.0

The new ForcePulley comprises the pulley itself and a new sensor module. The pulley is the same as V2 machines, but with an added sensor disc on the side. The FP sensor module will sit attached to the right-hand side Forward Riser, where the force of the row is measured. From here it is also able to measure the exact stroke length using the sensor disk. There is also the option to use Bluetooth, although it is recommended to stay tethered for Gym use.

The unit will have a small RGB LED mounted in the top, though this will be hidden from general view.

The S6 USB will remain the main metric calculator to stay true to the familiar S5 values.

This will also update via Bluetooth, if necessary.

A grey rectangular object with holes

Description automatically generated

See the following USB Protocol section for details on the software interface.

## S6 USB (C601):

Firmware Version: 1.06 (13/03/24)

The S6 USB will be a USB only communication giving out the same readings that the S5 on V2 machines, only over a USB protocol detailed in the following USB protocols section. This S6 USB uses the same sensor that the S5 used, located within the Top deck.

Also updateable via USB, this may require a separate app.

See page 22 for a comparison.

## 15.6” RowActive Tablet:

The final addition to the V3 line up will be the 15.6” tablet. Powered via the 12V/1A output from the main power distribution through a 5.5 x 2.2mm DC Jack plug.

This 15.6” tablet features a 3mm toughened glass 10-point capacitive touchscreen Infront of a 1080P display.

Previous demos seen by OTF had been using Android 11, the 1.8GHz Quad Cortex™-A53 NXP i.MX 8M Mini with 2GB of RAM and 8GB on-board memory, expandable with up-to a 64GB Micro SD Card, internally. Features single band 2.4Ghz Wi-fi, Bluetooth 5.1 and two programmable ANT Modules.

Or an upgraded SoM with Android 11, 1.8GHz Quad Cortex™-A53 NXP’s iMX8M Plus processor with 800MHz Cortex™-M7 Real-time co-processor. 2GB Ram – 16 GB Memory. expandable with up-to a 64GB Micro SD Card, internally. Features single band 2.4Ghz Wi-fi, Bluetooth 5.1 and two programmable ANT Modules that communicate over RS232.

Configurable with 10W User facing speakers and custom logo button. One USB 2.0 port (used in this configuration), Audio out port.

WiFi 5gHz should be able to be added to the spec.

The external ANT modules, via a GFIT, also have additional Bluetooth.

# Software:

It was noted on the call on 12th April 24 that Total time (rowed and in use) machine firmware and error code were desired. It would be possible from early research that these could be added in several areas of the machine. Including interval set times, if a need is there.

All protocols can be provided in a SDK. Movelab offers continued support for all devices and a demo app so speed up integration of the products mentioned.

## LightRing (LR000) USB Protocol

The USB interface on the Light Ring presents a CDC-ACM (virtual comm port) to the host tablet and enumerates as a standard serial port with VID = 1915 and PID = 5702 descriptors when connected.

#### Protocol

A total number of 260 ‘frames’ can be stored in the frame buffer on the Light Ring, where a frame consists of 52 individual LED colours.

From these frames, 54 animations can be created. An animation consists of up to 51 frames selected from the frame buffer. Frames added to an animation can be rotated, and their intensity can be modified.

If an animation is playing, and one of its frames is updated while the animation is playing, the updated frame will be used next time it is accessed, i.e. there is no buffering waiting for the animation to be stopped.

The Light Ring Command Protocol data format comprises sending a single command-specifying byte followed by any data needed to support the command. Some commands consist of the single command byte only e.g. ‘Increase Brightness’ while others have, for example, the colour data for each of the 52 LEDs in the ring. The correct number of bytes must be sent for each command.

The tables below list each command supported on the USB interface.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Description | | Format | | Reference |
|  | Command to set all the LEDs to a single colour defined by the first 3 data bytes GRB (24 Bpp) | | {00}{G}{R}{B} | |  |
|  |
| Identifier | Byte Index | Size (bytes) | Format | Value/Range | Description |
| COMMAND ID | 0 | 1 | Hex | 0x00 |  |
| G | 1 | 1 | Hex | 0x00 -> 0xAF | Green intensity |
| R | 2 | 1 | Hex | 0x00 -> 0xAF | Red intensity |
| B | 3 | 1 | Hex | 0x00 -> 0xAF | Blue intensity |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Description | | Format | | Reference |
|  | Command to set each LED to a specified colour defined by the next 156 bytes.  Total (52\* {G}{R}{B}) | | {01} {G1}{R1}{B1}  {G2}{R2}{B2} {G3}{R3}{B3}  {G4}{R4}{B4} ..etc | |  |
|  |
| Identifier | Byte Index | Size (bytes) | Format | Value/Range | Description |
| COMMAND ID | 0 | 1 | Hex | 0x01 |  |
| G1 | 1 | 1 | Hex | 0x00 -> 0xAF | Green intensity for 1st LED |
| R1 | 2 | 1 | Hex | 0x00 -> 0xAF | Red intensity for 1st LED |
| B1 | 3 | 1 | Hex | 0x00 -> 0xAF | Blue intensity for 1st LED |
| G2 | 4 | 1 | Hex | 0x00 -> 0xAF | Green intensity for 2nd LED |
| R2 | 5 | 1 | Hex | 0x00 -> 0xAF | Red intensity for 2nd LED |
| B2 | 6 | 1 | Hex | 0x00 -> 0xAF | Blue intensity for 2nd LED |
| Gx | 3(x-1)+1 | 1 | Hex | 0x00 -> 0xAF | Green intensity for xth LED |
| Rx | 3(x-1)+2 | 1 | Hex | 0x00 -> 0xAF | Red intensity for xth LED |
| Bx | 3(x-1)+3 | 1 | Hex | 0x00 -> 0xAF | Blue intensity for xth LED |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Description | | Format | | Reference |
|  | Add 1 frame to frame buffer.  Total (52\* {G}{R}{B}) | | {02}{FIFI}{G1}{R1}{B1}  {G2}{R2}{B2} {G3}{R3}{B3}  {G4}{R4}{B4} ..etc | |  |
|  |
| Identifier | Byte Index | Size (bytes) | Format | Value/Range | Description |
| COMMAND ID | 0 | 1 | Hex | 0x02 |  |
| F1 | 1 | 2 | Hex | 0x0000 -> 0x0103 | Frame index to  store |
| G1 | 1 | 1 | Hex | 0x00 -> 0xAF | Green intensity for 1st LED |
| R1 | 2 | 1 | Hex | 0x00 -> 0xAF | Red intensity for 1st LED |
| B1 | 3 | 1 | Hex | 0x00 -> 0xAF | Blue intensity for 1st LED |
| G2 | 4 | 1 | Hex | 0x00 -> 0xAF | Green intensity for 2nd LED |
| R2 | 5 | 1 | Hex | 0x00 -> 0xAF | Red intensity for 2nd LED |
| B2 | 6 | 1 | Hex | 0x00 -> 0xAF | Blue intensity for 2nd LED |
| Gx | 3(x-1)+3 | 1 | Hex | 0x00 -> 0xAF | Green intensity for xth LED |
| Rx | 3(x-1)+4 | 1 | Hex | 0x00 -> 0xAF | Red intensity for xth LED |
| Bx | 3(x-1)+5 | 1 | Hex | 0x00 -> 0xAF | Blue intensity for xth LED |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Description | | Format | | Reference |
|  | Command to Start animation | | {03}{AA}{TT}{DDDD} | |  |
|  |
| Identifier | Byte Index | Size (bytes) | Format | Value/Range | Description |
| COMMAND ID | 0 | 1 | Hex | 0x03 | command byte |
| AA | 1 | 1 | Hex | 0x00 -> 0x36 | animation slot index |
| TT | 2 | 1 | Hex | 00 -> 01 | animation type (00 - loop) (01 - one-shot) |
| DDDD | 3 | 2 | Hex | 0x0000 -> 0xFFFF | Msec to fade in animation. If looping, on first playback only |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Description | | Format | | Reference |
|  | Command to Stop the current animation | | {04}{DD} | |  |
|  |
| Identifier | Byte Index | Size (bytes) | Format | Value/Range | Description |
| COMMAND ID | 0 | 1 | Hex | 0x04 | command byte |
| DDDD | 1 | 2 | Hex | 0x0000 -> 0xFFFF | Msec to fade out animation |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Description | | Format | | Reference |
|  | Command to add selected frames from frame buffer to start of selected animation buffer. NB this  command will erase the second half of the animation on completion. | | {05}{NN}  {TS1}{BBBB1}{AA1}{RO1}{IF 1}  …  {TS26}{BBBB26}{AA26}{RO26}  {IF26} {AS} | |  |
|  |
| Identifier | Byte Index | Size (bytes) | Format | Value/Range | Description |
| COMMAND ID | 0 | 1 | Hex | 0x05 | command byte |
| NN | 1 | 1 | Hex | 0x01 -> 0x26 | Number of frames to add |
| TS1 | 2 | 2 | Hex | 0x0000 -> 0XFFFF | Byte-swapped timestamp (I.e. sending 0x12 0x34  for this will hold the frame for 0x3412  milliseconds) |
| BBBB1 | 4 | 2 | Hex | 0x0000 -> 0x0103 | Index of frame from frame buffer to add |
| R01 | 6 | 1 | Hex | 0x00 -> 0x33 | The offset to shift the animation frame with. 1 step shifts the frame 1 LED clockwise. 00 for no rotation |
| IFG1 | 7 | 1 | Hex | 0x00 -> 0xFF | Factor to change intensity of all Green LEDs. Value is fraction (/255) to be used to multiply Green value in base animation frame. |
| IFR1 | 8 | 1 | Hex | 0x00 -> 0xFF | Factor to change intensity of all Red LEDs. Value is fraction (/255) to be used to multiply Red value in base animation frame. |
| IFB1 | 9 | 1 | Hex | 0x00 -> 0xFF | Factor to change intensity of all Blue LEDs. Value is fraction (/255) to be used to multiply Blue value in base animation frame. |
| …  Up to 25 more {TS}{BBBB}{AA}{RO}{IFG}{IFR}{IFB} sequences | | | | | |
| AS | (NN x 8) +2 | 1 | Hex | 0x00 -> 0x36 | Animation slot to use |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Description | | Format | | Reference |
|  | Command to clear the frame buffer of all frames | | {06} | |  |
|  |
| Identifier | Byte Index | Size (bytes) | Format | Value/Range | Description |
| COMMAND ID | 0 | 1 | Hex | 0x06 | command byte |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Description | | Format | | Reference |
|  | Command to delete an animation | | {07}{AA} | |  |
|  |
| Identifier | Byte Index | Size (bytes) | Format | Value/Range | Description |
| COMMAND ID | 0 | 1 | Hex | 0x07 | command byte |
| AA | 1 | 1 | Hex | 0x00-> 0x36 | animation slot index |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Description | | Format | | Reference |
|  | Command to increase brightness by one increment | | {08} | |  |
|  |
| Identifier | Byte Index | Size (bytes) | Format | Value/Range | Description |
| COMMAND ID | 0 | 1 | Hex | 0x08 | command byte |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Description | | Format | | Reference |
|  | Command to decrease brightness by one increment | | {09} | |  |
|  |
| Identifier | Byte Index | Size (bytes) | Format | Value/Range | Description |
| COMMAND ID | 0 | 1 | Hex | 0x09 | command byte |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Description | | Format | | Reference |
|  | Command to set all LEDs to a preset colour at 50% brightness.  -00 blue  -01 red  -02 green  -03 cyan  -04 yellow  -05 magenta  -06 orange  -07 white | | {0A}{CC} | |  |
|  |
| Identifier | Byte Index | Size (bytes) | Format | Value/Range | Description |
| COMMAND ID | 0 | 1 | Hex | 0x0A | command byte |
| CC | 1 | 1 | Hex | 0x00-> 0x07 | colour index |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Description | | Format | | Reference |
|  | Command to disable auto power down | | {0C} | |  |
|  |
| Identifier | Byte Index | Size (bytes) | Format | Value/Range | Description |
| COMMAND ID | 0 | 1 | Hex | 0x0C | command byte |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Description | | Format | | Reference |
|  | Command to add selected frames from frame buffer the end of selected animation buffer. | | {0D}{NN}  {TS1}{BBBB1}{AA1}{RO1}{IF 1}  …  {TS26}{BBBB26}{AA26}{RO26}  {IF26} {AS} | |  |
|  |
| Identifier | Byte Index | Size (bytes) | Format | Value/Range | Description |
| COMMAND ID | 0 | 1 | Hex | 0x0D | command byte |
| NN | 1 | 1 | Hex | 0x01 -> 0x26 | Number of frames to add |
| TS1 | 2 | 2 | Hex | 0x0000 -> 0XFFFF | Byte-swapped timestamp (I.e. sending 0x12 0x34  for this will hold the frame for 0x3412  milliseconds) |
| BBBB1 | 4 | 2 | Hex | 0x0000 -> 0x0103 | Index of frame from frame buffer to add |
| R01 | 6 | 1 | Hex | 0x00 -> 0x33 | The offset to shift the animation frame with. 1 step shifts the frame 1 LED clockwise. 00 for no rotation |
| IFG1 | 7 | 1 | Hex | 0x00 -> 0xFF | Factor to change intensity of all Green LEDs. Value is fraction (/255) to be used to multiply Green value in base animation frame. |
| IFR1 | 8 | 1 | Hex | 0x00 -> 0xFF | Factor to change intensity of all Red LEDs. Value is fraction (/255) to be used to multiply Red value in base animation frame. |
| IFB1 | 9 | 1 | Hex | 0x00 -> 0xFF | Factor to change intensity of all Blue LEDs. Value is fraction (/255) to be used to multiply Blue value in base animation frame. |
| …  Up to 25 more {TS}{BBBB}{AA}{RO}{IFG}{IFR}{IFB} sequences | | | | | |
| AS | (NN x 8) +2 | 1 | Hex | 0x00 -> 0x36 | Animation slot to use |

##### *Example:*

|  |  |
| --- | --- |
| Bytes sent via USB I/F (hex) | Light Ring Response |
|  |  |
| 00 80 00 00 | All LEDs green |
|  |  |
|  |  |
| 00 00 80 00 | All LEDs red |
|  |  |
| 08 | Increase brightness by 1 step |
| 09 | Decrease brightness by 1 step |
|  |  |
| 0A 03 | Set LEDs to cyan at 50% brightness |
|  |  |
|  |  |
| 03 00 00 03 E8 | Loop animation from slot 0 with 1000mS fade in |
|  |  |

## ForcePulley (FP000) USB Protocol

The ForcePulley is a new development, which as of 13th March 2024, is still in a beta phase of testing, so some of the following protocol may possibly change. The “[Product characterisation](Product_Characterisation%201.12_OTFV3%20Highlighted.docx)” document, currently version 1.12, still details all the current protocol surrounding the product. See “Tethered Force Pulley” subheading for the beginning of the documentation. It starts off with the Bluetooth protocols used, which are also available, however for the OTF V3, the USB option will be the preferred route, this documentation starts on Page 41.

The USB interface on the Tethered Force Pulley presents a CDC-ACM (virtual comm port) to the host tablet and enumerates as a standard serial port with VID = 1915 and PID = 5700 descriptors when connected.

There are three main requests to be done for the USB output stream, the first is the general rowing metrics which may or may not be used for the OTF V3. Due to the use of a different sensor in an alternative location to the S5, the pulses that come from it are different, although the actual calculated metrics are correct sent from the module. The main idea is for the S6 USB to be used as the main metric calculator. However, if preferred or required as a fall-back scenario where the S6USB sensor is damaged or faulty, the rowing metrics can be obtained using the following S5 style query packets in S4/S5/S6 Compatibility Mode.

### S4/S6 Compatibility Mode

Host (tablet or PC) sends USB<crlf> to enable S4/S5/S6 compatibility output - pulse data is output if detected within each 25ms timeslot in the format PXX<crlf>. XX is the ascii hex representation of the number of pulses detected in the last 25mS. Pulse data is only transmitted if detected. If no pulses are detected for one second or more PING<crlf> is sent for each second during which there were no pulses. Internal algorithms detect start and end of stroke and output SS<crlf> and SE<crlf> between pulse data to indicate. Firmware revision, session timing and calculated rowing metrics are available through the original S4/S5/S6 Information request packet commands. *Please note that the PXX pulse data is only compatible in its format and timing, but not in content. Due to the higher resolution of the Force Pulley sensor disk and its location on the pulley instead of the flywheel, the pattern and number of pulses per stroke will be different.*

To establish communications with the connected rowing computer the following packets are required:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Application starting communications | | From: | PC/Tablet | Data: | None |
| U | SB + 0x0D0A | | | | |
| This is the very first packet sent by an application once the COM port is opened, this will tell the rowing computer to reply with its hardware type packet. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Hardware Type | | From: | Force Pulley | Data: | None |
| \_ | WR\_ + 0x0D0A | | | | |
| The Water Rower will reply with this packet when it receives a “USB” packet and will then proceed to send other packets accordingly until it switches off or the application issues an exit packet. | | | | | |

To terminate communications, use the following packet:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Application is exiting | | From: | PC/Tablet | Data: | None |
| E | XIT + 0x0D0A | | | | |
| Any application wishing to normally terminate (close) is required to send this packet to stop the automatic packets being sent to the PC/Tablet. | | | | | |

Acceptance, error and ping packets are as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Packet Accepted | | From: | Force Pulley | Data: | None |
| O | K + 0x0D0A | | | | |
| This packet will only be sent where no other reply to a PC/Tablet would otherwise be given. If a packet response is required to the PC/Tablet, then that will take the place of the OK packet. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Unknown packet | | From: | Force Pulley | Data: | None |
| E | RROR + 0x0D0A | | | | |
| The last received packet from the PC/Tablet was of an unknown time and caused a general ERROR reply to be issued. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Ping | | From: | Force Pulley | Data: | None |
| P | ING + 0x0D0A | | | | |
| Sent once a second while NO rowing is occurring to indicate to the PC/Tablet the rowing monitor is still operational but stopped. | | | | | |

PC/Tablet requested reset:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Request the rowing computer to reset | | From: | PC/Tablet | Data: | None |
| R | ESET + 0x0D0A | | | | |
| Rowing metrics are reset. | | | | | |

### 

### Strokes

These packets are **auto transmitted** by the rowing computer and are sent once communications has been established. PING will be sent until rowing has commenced.

The stroke data has a priority order in which they will be sent ahead of ALL other pending information requests. During very low activity in rowing or minor movement of the pulley, either of these packets could be sent to indicate activity even though actual rowing is not occurring. Filtering of these is required by the PC / Tablet application.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Start of stroke | | From: | Force Pulley | Data: | None |
| S | S + 0x0D0A | | | | |
| Start of stroke pull to show when the rowing computer determined acceleration occurring in the pulley. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| End of stroke | | From: | Force Pulley | Data: | None |
| S | E + 0x0D0A | | | | |
| End of stroke pull to show when the rowing computer determined deceleration occurring in the pulley (now entered the recovery phase). | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Pulse Count in the last 25mS | | From: | Force Pulley | Data: | ACH |
| P | XX + 0x0D0A | | | | |
| “XX” is an ACH value representing the number of pulse’s counted during the last 25mS period; this value can range from 1 to 50 typically. (Zero values will not be transmitted).  There is 2.6mm of belt travel per pulse. | | | | | |

### Rowing Information request packets

Information request packets are to allow the connected PC or Tablet to request data from the rowing computer. Rowing data is updated at the start of the recovery phase of each stoke.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Request Model Information | | From: | PC/Tablet | Data: | None |
| I | V? + 0x0D0A | | | | |
| Request details from the rowing computer on what it is and firmware version | | | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Current Model Information | From: | Force Pulley | Data: | ASCII |
| “TFP1.5 FW v1.1.5” + 0x0D0A | | | | |
| Details of what unit is attached:  Indicates Tethered Force Pulley with v1.5 hardware and v1.1.5 firmware | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Read Stroke Rate | | From: | PC/Tablet | Data: | ACH |
| I | RS + 1A9 + 0x0D0A | | | | |
| Requests the stroke rate in single byte hex format. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Stroke Rate Value | | From: | Force Pulley | Data: | ACH |
| I | DS + 1A9 + Y1 + 0x0D0A | | | | |
| Returns the single byte Y1 stroke rate data in hex format. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Read Total Distance | | From: | PC/Tablet | Data: | ACH |
| I | RD + 057 + 0x0D0A | | | | |
| Requests the total distance in two-byte hex format. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Total Distance Value | | From: | Force Pulley | Data: | ACH |
| I | DD + 057 + Y2 + Y1 + 0x0D0A | | | | |
| Returns the high byte Y2 and the low byte Y1 of the total distance data in hex format. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Read Stroke Count | | From: | PC/Tablet | Data: | ACH |
| I | RD + 140 + 0x0D0A | | | | |
| Requests the stroke count in two-byte hex format. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Total Stroke Count Value | | From: | Force Pulley | Data: | ACH |
| I | DD + 140+ Y2 + Y1 + 0x0D0A | | | | |
| Returns the high byte Y2 and the low byte Y1 of the stroke count data in hex format. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Read Speed | | From: | PC/Tablet | Data: | ACH |
| I | RD + 14A + 0x0D0A | | | | |
| Requests the speed (cm/s) in two-byte hex format. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Speed Value | | From: | Force Pulley | Data: | ACH |
| I | DD + 14A + Y2 + Y1 + 0x0D0A | | | | |
| Returns the high byte Y2 and the low byte Y1 of the speed (cm/s) data in hex format. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Read Power | | From: | PC/Tablet | Data: | ACH |
| I | RD + 088 + 0x0D0A | | | | |
| Requests the Power (W) in two-byte hex format. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Power Value | | From: | Force Pulley | Data: | ACH |
| I | DD + 088 + Y2 + Y1 + 0x0D0A | | | | |
| Returns the high byte Y2 and the low byte Y1 of the power data in hex format. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Read Calories | | From: | PC/Tablet | Data: | ACH |
| I | RT + 08A + 0x0D0A | | | | |
| Requests the Power (W) in two-byte hex format. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Calories Value | | From: | Force Pulley | Data: | ACH |
| I | DT + 08A + Y3 + Y2 + Y1 + 0x0D0A | | | | |
| Returns the high order byte Y3, the middle order byte Y2 and the low byte Y1 of the calories data in hex format. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Read Elapsed Time H:M:S | | From: | PC/Tablet | Data: | ACH |
| I | RT + 1E1 + 0x0D0A | | | | |
| Requests the elapsed time in three-byte bcd format. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Elapsed Time Value | | From: | Force Pulley | Data: | BCD |
| I | DT + 1E1 + Y3 + Y2 + Y1 + 0x0D0A | | | | |
| Returns the hours (0-9) Y3, minutes (0-59) Y2 and the seconds(0-59) Y1 of the elapsed time data in bcd format. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Read Elapsed Time (hundredths) | | From: | PC/Tablet | Data: | ACH |
| I | RS + 1E0 + 0x0D0A | | | | |
| Requests the hundredths of a second portion of elapsed time in single byte bcd format. | | | | | |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Stroke Rate Value | | From: | Force Pulley | Data: | BCD |
| I | DS + 1E0 + Y1 + 0x0D0A | | | | |
| Returns the single byte Y1 elapsed time hundredths of a second data in bcd format. | | | | | |

### Force Pulley Data Mode

The second request type that can be done is for the Force data stream. This is real-time force and pulley rotation data that can be used to draw force and power curves.

Sending FORCE\_ON<crlf> enables the output of 16-bit ascii hex encoded signed force data in the format FXXXX<crlf> every 12.5milliseconds. For V1.06, the output value is 134 x the force in kg; for V1.10 and later it is 111.1 x the force in kg. Before the force values, OK<crlf> is sent to acknowledge the command.

### Calculated Values Output

The third request is to output stroke length and drive energy at the end of the drive phase.

Sending CALCS\_ON<crlf> enables the output of calculated values in the USB output. These are sent out at the end of the rowing drive phase in the format LXXXX<crlf>WYYYY<crlf>.

XXXX is the ascii hex representation of the stroke length in mm and YYYY is the amount of energy in the drive phase in Joules. Sending CALCS\_OFF<crlf> or EXIT<crlf> disables this output.

### Data output modes

If USB<crlf> had been sent previously and the device was in S4/S6 compatibility mode, then SS, SE and PING are suppressed and pulse data now includes P00<crlf> when no pulse data is detected in the 12.5ms time slot. If the device was not in S4/S6 compatibility mode before the FORCE\_ON command, then the USB command will need to be sent to get pulse data.

* Whenever the USB command is sent, \_WR\_<crlf> is sent to acknowledge it.
* Force data is output first followed by pulse data, so a pair of points is a force reading followed by its distance (pulse) reading.
* Sending FORCE\_OFF disables the force data output and reverts the pulse data to S4/S6 compatibility mode if pulse data had been enabled.
* Sending CALCS\_ON / CALCS\_OFF will enable / disable calculated values regardless of data mode.
* Sending EXIT disables all output. EXIT<crlf> is sent back in acknowledgement.

##### *Example:*

|  |  |  |
| --- | --- | --- |
| Android/PC USB I/F | Tethered Force Pulley USB I/F |  |
|  | *No Output* |  |
| FORCE\_ON <crlf> |  |
|  | OK<crlf> |  |
|  | F0001  FFFFD  F002E  F0055  F00C6 | 12.5ms per force reading |
| USB<crlf> | F010A  F0131  … |  |
|  | \_WR\_ |  |
|  | F0184  P02 | 12.5ms per force/pulse reading  pair |
|  | F0C0B  P03 |
|  | … |
| FORCE\_OFF<crlf> | … |
|  | OK<crlf> |  |
|  | P03  P02  P01 | 25ms per pulse reading |
|  | PING  PING  PING | 1 second between PINGs |
| EXIT<crlf> | … |  |
|  | EXIT<crlf> |  |
|  | *No Output* |  |

### Method for detecting catch, end of drive, calculating stroke length and energy (force pulley mode only).

* Continuously sum pulse count and the product of pulse count and force for each pair of force and pulse data that is output.
* Continuously maintain a record of the stroke force above 10N.
* Zero the pulse count sum and the pulse count x force sum if the pulse data is 0 (P00<crlf).
* If the pulse count sum exceeds 120 and the force value is greater than 20N then the catch point was the last zero reading of the pulse data. The continued summation of the pulse counts until the next zero reading or P01,P02 pair (end of drive phase) will give stroke-length when multiplied by 2.6. The summation of the pulse count and force product pairs gives the energy per stroke in the drive phase when multiplied by 2.6.

## S6 USB (C601) USB Protocol

For the S6 USB, the USB protocol was derived from our original Series IV V2 monitor. Within the “[Product characterisation](Product_Characterisation%201.12_OTFV3%20Highlighted.docx)” document, currently version 1.12, page 5-13 outlines the structure of the protocol we use. However, some updates were done, these are referenced under “S6-USB” on page 35. The difference between the two are that the S6 USB was able to calculate the average faster with results being transmitted over USB within the first stroke, Averages are based on 4 strokes.

The USB interface on the S6USB presents a CDC-ACM (virtual comm port) to the host tablet and enumerates as a standard serial port with VID = 04D8 and PID = 000A descriptors when connected.

The protocol has the same command set and structure as the Force Pulley in S4/S5/S6 compatibility mode detailed earlier and can be used to provide the same metrics. The response to the identity command (IV?<crlf>) in the case of the S6USB is “10106<crlf>”

This is to be the main metric calculator that resembles the data received from the previous generation S5 display.

## Supported Metrics

|  |  |  |  |
| --- | --- | --- | --- |
|  | S5 | S6USB | Force Pulley |
| Distance |  |  |  |
| Speed(m/s) |  |  |  |
| Speed(mph) |  |  |  |
| Speed(s/500m) |  |  |  |
| Speed(s/2km) |  |  |  |
| Average Speed (m) |  |  |  |
| Average Speed(mph) |  |  |  |
| Average Speed(s/500m) |  |  |  |
| Average Speed(s/2km) |  |  |  |
| Stroke Rate |  |  |  |
| Average Stroke Rate |  |  |  |
| Accumulated Power |  |  |  |
| Instantaneous Power |  |  |  |
| Average Power |  |  |  |
| Calories |  |  |  |
| Calories/hr |  |  |  |
| Calories/min |  |  |  |

|  |
| --- |
|  |

Supported

|  |
| --- |
|  |

Unsupported