This document details the communication protocol used by the oXigen USB dongle.
The purpose of this communication is

- send back to the PC all the data that the controllers have collected from the cars, for 4 cars at most for each USB packet. This includes information on lap timing, pit lane, battery level, and more.
- transfer dongle-specific data, race-wide data (e.g. race status, max speed), and car-specific data (e.g. for example single car braking or limitations for fuel based strategies) from the PC RMS (Race Management Software) to the dongle and in turn to the controllers.

In an oXigen system, racing can be in one of four possible states, selected via RMS (Race Management System):

Started / Stopped / Paused / Flagged

Started:

Race is normally on. Lap counting is enabled. Max speed is set through the 'max speed knob'

Stopped:

Race is over.. Pressing 'Start' clears all Lap Counting data

Paused:

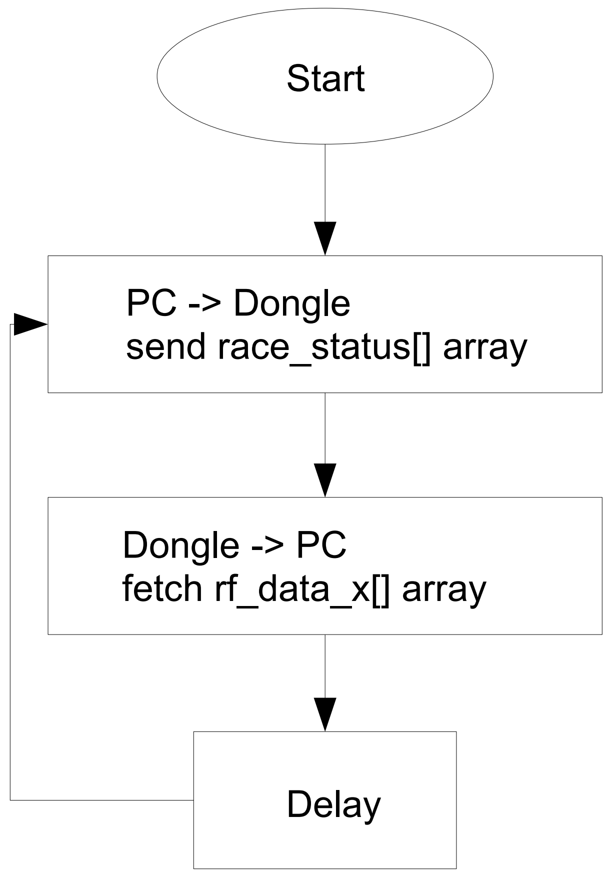
Race is, well, paused. It can be restarted or stopped. If restarted, lap counting is not affected.

Flagged:

This is the 'safety car' condition. Speed may be limited and / or lane changing disabled, or not.

NOTE: The PC recognises the dongle device as a virtual COM port. The ID vendor is 0x1FEE and the ID product is 0x0002.

The communication between PC and dongle is bidirectional as shown in the following flowchart.



The 'delay' value is strictly connected to the speed of the PC speed and its tasks' load, and this in turn may affect the speed of the RMS. Note that each controller sends its data back to the dongle with a 600ms cycle time (i.e. every 600 ms).

Dongle - PC communication protocol:

As described above, the PC sends the race status to the dongle before reading the information collected from the cars. Hence, the protocol consists of two parts: TX and RX.

TX (PC→ Dongle):

TX protocol: payload 10 bytes, *race_status[7]*.

race_status[0] and *race_status[1]* hold general information valid for all the controller/car pairs

race_status[2], *race_status[3]* and *race_status[4]* hold specific information only valid for one controller/car pair.

byte	bit	value (hex)	meaning
0	0..4	0	not used
		1	race **stopped**. (set race_status[1] to 0x00)
		2	not used
		3	race **running.**
		4	race **paused**.
		5	race **flagged,** LC **enabled**.
		6	dongle command. Refer to 'Dongle Commands' paragraph
		15h	race **flagged,** LC **disabled**.
	5	0/1	lap counting in pitlane: 0: enabled : lap counter is increased when car runs along the pitlane (lap trigger position is defined by bit 6) 1: disabled : lap counter is not increased when car runs along the pitlane (the value of bit 6 is not considered)
	6	0/1	pitlane lap trigger (valid only if bit 5 = 1): 0: lap counter is increased at the pitlane entry (lap time reference is the first magnet of the two ones used to define the pitlane entry) 1: lap counter is increased at the pitlane exit (lap time reference is the magnet used to define the pitlane exit)
	7		not used (*)

(*) When the chrono is started, the dongle receives the FFh command which means free race. The command is sent only when the chrono is started and in no other operating mode. The aim is to collect the number of controllers turns. The same can be achieved by starting the chrono in PAUSE mode.

byte	meaning	value (hex)	meaning
1	maximum speed	0-FFh	maximum global allowed speed (for all cars). This can be changed at will according to the status of the race. If the race is stopped or paused, this value is ignored by the controllers as the speed is forced to 0.

byte	meaning	value (hex)	meaning
2	controller ID for car-specific information tx	0-27h	set this byte to the ID of the car that specific information (e.g. speed limitation) must be sent to

byte	bit	value (hex)	meaning
3	0..6	0-7Fh	command value. Refer to 'Command description' section for an explanation
	7	0/1	0: global command. (for all controllers) 1: car-specific command (car ID according to race_status[2]

byte	meaning	value (hex)	meaning
4	command argument	0-FFh	command argument. Refer to 'Command description' section for an explanation

Note: each instruction for the controller/car is made of a 'command', (race_status[3], bits 0..6, and a 'argument', race_status[4])

byte	meaning	value (hex)	meaning
5	not used		reserved

byte	meaning	value (hex)	meaning
6	not used		reserved

byte	meaning	value (hex)	meaning
7	not used		reserved

In general, race information has priority over controller/car specific information. Possible exceptions are, for example, the speed, which is limited by the lowest of the two: for example, if the global maximum speed is 90%, and the maximum speed for a specific car is 80%, that car will set its limit to 80%. If however the global top speed is turned down to 70%, then the car will set its speed limit to 70%.
This principle applies to most cases.

The following three bytes are used to communicate to the dongle the race timer value. The present race timer value is concatenated by PC to the USB packet when this latest one is ready to be transmitted to the dongle.
The saved race timer value will be forwarded, as is, to the PC by the dongle when the USB packet for PC is loaded. This way, the PC is able to calculate the time delay between the USB packet sending and the USB packet receiving. This time delay, added to *rf_data_x[4]* byte of the buffer sent to the PC by the dongle, is used to calculate the race time value of the lap trigger.

byte	meaning	value (hex)	meaning
8, 9, 10	race timer value	0-FFFFFFh	Value, in centiseconds, of the race timer saved at the moment the USB packet is sent to the dongle

RX (Dongle → PC):

RX protocol: for each controller-car pair, 13bytes, *rf_data_x[13]* are received. Nine bytes collect the status of the controller-car pair; four bytes, with the value of the race timer saved when the dongle receives the packet containing a new lap info from the controller, are concatenated to the packet. Payload max 52 bytes (max 4 controller-car pairs): in other words, data is fetched from each controller-car pair in round-robin fashion, with a maximum of 4 pairs each USB payload.

byte	bit	value (hex)	meaning
0	0	0/1	car reset event: 0: car power supply hasn't changed 1: car has just been powered up / reset (info available for 2 seconds)
	1	0/1	controller-car link check: 0: controller link with its paired car hasn't changed 1: controller has just got the link with its paired car (info available for 2 seconds) (e.g.:link dropped and restarted)
	2	0/1	controller battery level warning: 0: controller battery level is OK 1: controller battery level is low
	3	0/1	track call check: 0: no track call from controller 1: track call from controller (info available for 2 seconds). Round button on controller (ID equal to *rf_data_x [1]*) was pressed for more than 0,5 seconds. This information can be used to trigger a track call, if, for example, the relative car is off track AND this condition occurs. Race can then be paused; and restarted manually or automatically after a certain time delay
	4	0/1	sub software release carry: 0: no action 1: add 16 to sub software release value
	5	0/1	arrow up button status: 0: button not pressed 1: button pressed
	6	0/1	arrow down button status: 0: button not pressed 1: button pressed
	7	1	always equal to 1

byte	bit	value (hex)	meaning
1	0..7	0..27h	controller-car pair ID in hexadecimal format (it certifies the controller is powered up)

byte	bit	value (hex)	meaning
2	0..7	0..FFh	last lap time (high byte) in cs (hundredths of sec): see Note1

byte	bit	value (hex)	meaning
3	0..7	0..FFh	last lap time (low byte) in cs (hundredths of sec): see Note1

Note 1:
Get the last lap time in seconds with the following formula

last lap time [s] = [(rf_data_x [2]*256) + (rf_data_x [3])] / 99,25

byte	bit	value (hex)	meaning
4	0..7	0..FFh	time delay, in centiseconds, between the moment the lap trigger is detected by the car and the moment the new lap info is received by the dongle

byte	bit	value (hex)	meaning
5	0..7	0..FFh	total lap number (high byte): see Note2

byte	bit	value (hex)	meaning
6	0..7	0..FFh	total lap number (low byte): see Note2

Note 2:
Get the total lap number with the following formula

lap number = (rf_data_x [6] * 256) + (rf_data_x [5])

byte	bit	value (hex)	meaning
7	0..6	0..7Fh	trigger mean value: see Note3
	7	0/1	car fuel and feedback: 0: the car is not on the track 1: the car is on the track. Info available only if the paired controller is powered up

Note 3:
This data can be used to calculate the fuel consumption
Currently the Chrono RMS uses the following formula to scale it to 10:

tmpvalue = ((rf_data_x [6] AND 7Fh) / 127 * 10)

byte	bit	value (hex)	meaning
8	0..3	0/1	sub software release of the device specified by bit 7
	4,5	0..3h	main software release of the device specified by bit 7
	6	0/1	car pit-lane status: 0: car is not in pit-lane; 1: car is in pit-lane
	7	0/1	device software release owner: 0: controller software release; 1: car software release

Note 4: if the SCP controller is reset, when the car crosses the finish line the dongle sends the correct lap number to the PC with a lap time value of zero. This to avoid a wrong lap time after a controller reset since the lap time is calculated by SCP, and can be used to detect the reset of the controller itself. This may be especially important in case of a controller reset while the car is in pit lane, while at the same time the car is removed from track, a condition which may lead to one extra lap being added upon pit lane exit.

byte	meaning	value (hex)	meaning
9..12	race timer value	0-FFFFFFFFh	Value, in centiseconds, of the race timer saved at the moment the dongle receives the packet containing a new lap info from the controller.

Note 5: The dongle counts the race timer, in centiseconds. It is cleared when the dongle receives the START command from the PC. Value, in centiseconds, of the race timer saved at the moment the dongle receives the packet containing a new lap info from the controller is concatenated, split in four bytes, to the packet.

Get the race timer value, in centiseconds, corresponding to the last lap recorded with the following formula

race timer (last lap) = (rf_data_x [9] * 16777216) + (rf_data_x [10] * 65536) + (rf_data_x [11] * 256) +
+ (rf_data_x [12]) - (rf_data_x [4])

This value is used to manage the race table in case of multiples cars with the same lap counter value.

Device commands

Two command types are currently available:

- commands for one or all controllers;
- commands for the dongle.

Controller(s) commands

Command description	race_status[3] [0..6]	race_status[4]
No Action	0000000 b	00h
Set Pit Lane Speed	0000001 b	New pit-lane speed value
Set Maximum Speed	0000010 b	Maximum speed value
Set Minimum Speed	0000011 b	bit 7: 1/0 force exchange such as arrow ↑ SCP button bit 6: 1/0 force exchange such as arrow ↓ SCP button bit 5..0: minimum speed limitation divided by 2
Set RF Tx Power Level	0000100 b	Transmission power value 0 : -18dBm 1 : -12dBm 2 : - 6dBm 3 : 0dBm
Set Maximum Brake	0000101 b	Maximum brake value

The MSB of race_status[3] discriminates between a *global* commmand (all controllers) (set MSB to '0') or a *specific* controller command (set MSB to '1'). Hence, the value in brackets in column 2 describe the command for the specific controller. Refer to TX (PC→ Dongle) section, byte 2,3,4 explanation

Examples:

- Set Maximum Speed to 7Fh, for controller-car pair with ID 3:
  - race_status[2] = 03h; → controller/car number
  - race_status[3] = 82h; → command type
  - race_status[4] = 7Fh; → upper speed limit value
- Set Minimum Speed to 40h, for all controller-car pairs:
  - race_status[2] = not important in this case type;
  - race_status[3] = 03h; → command type
  - race_status[4] = 20h; → minimum speed value. The final value will be 40h.

To cancel a previously sent command, resend the commad with a different value.
Examples:

- Emable top speed 100% for controller-car pair with ID 3:
  - race_status[2] = 03h;   → controller/car number
  - race_status[3] = 82h; → command type
  - race_status[4] = ffh; → upper speed limit value

- Remove minimum speed limitat for all controller-car pairs:
  - race_status[2] = not important in this case type;
  - race_status[3] = 03h; → command type
  - race_status[4] = 00h; → minimum speed default value (speed 0 and no lane exchange forced).

Dongle Commands

Currently, we have developed only the dongle software release query, with the following command:

race_status[0]: 06h;
race_status[1]: 06h;
race_status[2]: 06h;
race_status[3]: 06h;
race_status[4]: XXh (any value);
race_status[5]: XXh (any value);
race_status[6]: XXh (any value);

The dongle will reply with five bytes: main sw release in first one, sub sw release in second one. The others three are ‘zero’ value bytes.

Communication Reset

Using the Slot.it chrono with the O204a dongle, the interruption of the radio communication between the dongle and the controllers was sometime detected. This is evidenced by the turning off of the green light that distinguishes the active connection of the controllers.
It is possible to try to restore the radio communication by sending the string 'SPISTART' to the dongle via USB. The dongle restarts the radio communication without clearing the lap count. This is valid from firmware version 3.13.

race_status[0]: ‘S’;
race_status[1]: ‘P’;
race_status[2]: ‘I’;
race_status[3]: ‘S’;
race_status[4]: ‘T’;
race_status[5]: ‘A’;
race_status[6]: ‘R’;
race_status[6]: ‘T’;