Water Rower S4 & S5 USB Protocol

Issue 1.04 – Aug 2009

PROVISIONAL COPY, ADDITIONS AND CHANGES OF THE MEMORY MAP ARE POSSIBLE DURING THE CODE CHANGES OF VERSION 2.00 FIRMWARE.

Overview

The Water Rower S4 and S5 rowing computers use a USB CDC interface to communicate with a PC or similar. A CDC is a virtual “com” port, that is once connected to a PC a new hardware “com” port will appear in the ports listing. (Installation requires the Microchip INF file).

This new “com” port is auto baud rate up to 115,200 baud (recommended 19,200 for slow update applications) and may be on any “com” number; typically the designer has found this to be COM9, but any number could be used. The port has product identification strings as follows, “Microchip Technology Inc.” and “CDC RS-232: WR-S4.2” (S5.0 for S5).

The micro-controller of the rowing computer require the packets sent to and from it to be no larger than 50 bytes of data, this includes ALL preceding and terminating characters, more on these later. Each packet is sent to do or give specific information to the PC or the rowing computer, it is recommended a PC sends data no faster than every 25mS (1 packet), this gives the rowing computer time to process and reply as well as send existing messages which maybe pending.

During rowing the rowing computer will be outputting a lot of packets, typically this could be as fast as once every 5-10mS, a PC will usually hold this in a buffer (dependant on language used) for the user’s application to process, each packet will be terminated in a form for easy “line” reading of the data, regular servicing of the incoming data will be essential to stop the port from stalling. Should a stall occur the rowing computer would need to be reconnected to the PC. Typically 10mS has been used to great success.

Packets

Each packet is formed using ASCII characters; initially the first character will be the command or data type that is following. After this will be either a subset indicator or the actual data, data will be in either decimal or hexadecimal.

Using both will allow less conversion needed for the micro-controller as well as keeping the packet size smaller when using hex, each packet will indicate what data type it is in this protocol and range of values it will cover maximum.

Finally after the data will be the terminating character, this will be 0x0D0A (hex), this tells most languages a CR LF (Carriage Return & Line Feed) were sent. In visual studio a complete packet can then be read with a single command from the buffer even with multiple packet’s received.

The micro-controller will also work with this format, as it requires the termination character to know when to process the message it just received.

Every packet sent from the PC to the rowing computer will result in a reply packet, however rowing computer packets do not need to be acknowledged in any way, this keeps the packets required sent to a minimum.

Packet format

The following format shows how all packets will be described, this gives a clear format in a repeated way to ensure the many current and future commands can be added without causing corruption of earlier defined packets.

Packet description (Error message)	From:	S4/5	Data:	None
E	RROR + 0x0D0A			
Comment for this packet.				

The above example explained:

Packet description would be "Error message"; this would be to tell the PC the last reply it received was of an unknown type.

From can be seen as either an S4 or S5 rowing computer, this can also be PC for packets to the rowing computer.

Data (type) in this case is none; there are no values inclusive of this packet. However, if there were any data to be included this would indicate if that data was in decimal (dec), hexadecimal (hex), Ascii coded decimal (ACD) or Ascii coded hexadecimal (ACH) format, correct conversion is required at the PC end to match the rowing computer.

E – This is the command letter for all "Error" type packets, other packets may also proceed with E, however there next character will be different, hence all error's will start "ER", where another command maybe "E2". An "E2" command could be used to then read or write to Eeprom should this be required, this ensures a vast amount of data and command messages available.

"+ 0x0D0A" This is the termination of the packet, sent as separate bytes it means CR LF (carriage return, Line feed). In decimal this is 13, 10 and sent with visual basic using the "chr" command. The "+" is NOT a character, this is used to break up the message for documentation only, additionally the 0x in 0x0D0A is also not used character's, this is to indicate the value is in hex.

Comment for the packet explains itself, some packets will require additional information to indicate the structure of that packet, typically when using hex values will include packing 0's for large values, this eliminates the need for packet length management.

Empty packet box.	From:		Data:	
	+ 0x0D0A			

Basic packets

These are basic packets for all versions of the USB rowing computer, there need is to give information to the PC of the device that is attached and if messages were accepted or rejected.

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

Application starting communication’s	From:	PC	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:	S4/5	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 switch’s off or the application issues an exit packet.				

To terminate communications use the following packet:

Application is exiting	From:	PC	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.				

Acceptance, error and ping packets are as follows:

Packet Accepted		From:	S4/5	Data:	None
O	K + 0x0D0A				
This packet will only be sent where no other reply to a PC would otherwise be given. If a packet response is required to the PC then that will take the place of the OK packet.					

Unknown packet		From:	S4/5	Data:	None
E	RROR + 0x0D0A				
The last received packet from the PC was of an unknown time and caused a general ERROR reply to be issued.					

Ping		From:	S4/5	Data:	None
P	ING + 0x0D0A				
Sent once a second while NO rowing is occurring to indicate to the PC the rowing monitor is still operational but stopped.					

PC requested reset:

Request the rowing computer to reset		From:	PC	Data:	None
R	ESET + 0x0D0A				
Request the rowing computer to perform a reset; this will be identical to the user performing this with the power button. Used prior to configuring the rowing computer from a PC. Interactive mode will be disabled on a reset.					

Information request packets

Information request packets are to allow the connected PC to request data from a large area of memory in the rowing computer, these locations may change from version to version hence it is essential to first determine the firmware of the device connected and associate with the needed memory map.

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

Current Model Information		From:	S4/5	Data:	ACD
I	V + Model + Version High + Version Low + 0x0D0A				
Details of what unit is attached: Model - Sent as 4 or 5 to indicate if it is a Series 4 or series 5 rowing computer. Version high - 02 as an example for version 2.00 MSB of the firmware version. Version low - 00 as an example for version 2.00 LSB of the firmware version.					

The read packets will retrieve values from the rowing-computer memory, these locations are raw data which maybe a decimal, hexadecimal, binary or BCD format, each will be returned in ACH format in the packet. Correct conversion and usage will be needed for the PC application to use the values.

XXX is in ACH format and has a maximum range of 0x000 to 0xFFF, however not all locations are available (see Memory Map), errors will be replied for out of spec memory reads.

Read a single memory location		From:	PC	Data:	ACH
I	RS + XXX + 0x0D0A				
Requests the contents of a single location XXX, this will return a single byte in hex format.					

Value from a single memory location		From:	S4/5	Data:	ACH
I	DS + XXX + Y1 + 0x0D0A				
Returns the single byte of data Y1 from location XXX for the users application.					

Read double memory locations	From:	PC	Data:	ACH
I	RD + XXX + 0x0D0A			
Requests the contents of two location starting from XXX, this will return two bytes in hex format.				

Value from double memory location	From:	S4/5	Data:	ACH
I	DD + XXX + Y2 + Y1 + 0x0D0A			
Returns two bytes of data starting from the second location first (Y2) then location XXX (Y1). This is for reading 16bit values which have (H)igh and (L)ow pair in one go.				

Read triple memory location	From:	PC	Data:	ACH
I	RT + XXX + 0x0D0A			
Requests the contents of three locations starting from XXX, this will return three bytes in hex format.				

Value from a single memory location	From:	S4/5	Data:	ACH
I	DT + XXX + Y3 + Y2 + Y1 + 0x0D0A			
Returns three bytes of data starting from the third location first (Y3) then (Y2) to location XXX (Y1). This is for reading 24bit values like a clock, which has Hours, Minutes & Seconds.				

Strokes

These packets are auto transmitted by the rowing computer; they have 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 paddle, 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 application.

<table>
<tr><td colspan="2">Start of stroke</td><td>From:</td><td>S4</td><td>Data:</td><td>None</td></tr>
<tr><td>S</td><td colspan="5">S + 0x0D0A</td></tr>
<tr><td colspan="6">Start of stroke pull to show when the rowing computer determined acceleration occurring in the paddle.
This packet has the highest priority of transmission on the USB.</td></tr>
</table>

<table>
<tr><td colspan="2">End of stroke</td><td>From:</td><td>S4</td><td>Data:</td><td>None</td></tr>
<tr><td>S</td><td colspan="5">E + 0x0D0A</td></tr>
<tr><td colspan="6">End of stroke pull to show when the rowing computer determined deceleration occurring in the paddle. (Now entered the relax phase).
This packet has the second highest priority of transmission on the USB.</td></tr>
</table>

<table>
<tr><td colspan="2">Pulse Count in the last 25mS</td><td>From:</td><td>S4</td><td>Data:</td><td>ACH</td></tr>
<tr><td>P</td><td colspan="5">XX + 0x0D0A</td></tr>
<tr><td colspan="6">“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). Please refer to “Water Rower Series 4 Rowing Algorithm.doc” for in depth details on how to use this data. At this time the constant values are:
pins_per_xxcm 32 ; number of pin edges allowed to equal xxcm (dec)
distance_xxcm 35 ; number of cm per flagged xxcm no. of pins (dec)
This packet has the third highest priority of transmission on the USB.</td></tr>
</table>

Display Settings

The following settings allow the PC to set the required display parameters.

Display: Set Intensity – Meters per second	From:	PC	Data:	None
D	I + MS + 0x0D0A			
Change the intensity display.				

Display: Set Intensity – MPH	From:	PC	Data:	None
D	I + MPH + 0x0D0A			
Change the intensity display.				

Display: Set Intensity – 500m	From:	PC	Data:	None
D	I + 500 + 0x0D0A			
Change the intensity display.				

Display: Set Intensity – 2Km	From:	PC	Data:	None
D	I + 2KM + 0x0D0A			
Change the intensity display.				

Display: Set Intensity – Watts	From:	PC	Data:	None
D	I + WA + 0x0D0A			
Change the intensity display.				

Display: Set Intensity – Cal/hr	From:	PC	Data:	None
D	I + CH + 0x0D0A			
Change the intensity display.				

These change the display to average intensity’s.

Display: Set Intensity – Average meters per second	From:	PC	Data:	None
D	A + MS + 0x0D0A			
Change the intensity display.				

Display: Set Intensity – Average MPH	From:	PC	Data:	None
D	A + MPH + 0x0D0A			
Change the intensity display.				

Display: Set Intensity – 500m	From:	PC	Data:	None
D	A + 500 + 0x0D0A			
Change the intensity display.				

Display: Set Intensity – 2Km	From:	PC	Data:	None
D	A + 2KM + 0x0D0A			
Change the intensity display.				

These change the distance display type.

Display: Set Distance – Meters	From:	PC	Data:	None
D	D + ME + 0x0D0A			
Change the distance display.				

Display: Set Distance – Miles	From:	PC	Data:	None
D	D + MI + 0x0D0A			
Change the distance display.				

Display: Set Distance – Km	From:	PC	Data:	None
D	D + KM + 0x0D0A			
Change the distance display.				

Display: Set Distance – Strokes	From:	PC	Data:	None
D	D + ST + 0x0D0A			
Change the distance display.				

Eeprom Setting

Some values are stored even while there is no power to the rowing computer; these are stored in Eeprom (known as E2). Where these need to be read, use the read commands to read the RAM location of the similar name. Writing is a one-time operation on request ONLY; do not send these commands to often.

Eeprom setting: pins_per_xxcm	From:	PC	Data:	ACH
E	2PX + YY + 0x0D0A			
Write the value of YY to the above location, this value has a limited range of 10 to 63 (decimal). An error will be sent if the value is outside of this range.				

Eeprom setting: distance_xxcm	From:	PC	Data:	ACH
E	2DX + YY + 0x0D0A			
Write the value of YY to the above location, this value has a limited range of 10 to 63 (decimal). An error will be sent if the value is outside of this range.				

Application Interactive controls.

These controls once enabled are to allow the rowing computer to navigate an application that understands the use of this system; this mode will be cancelled by a user or PC reset. These packets are sent as and when required based on the action of the user or PC application.

Start Interactive mode	From:	S4/5	Data:	None
A	IS + 0x0D0A			
The user of the rowing computer has requested the use of interactive controls; the PC application must acknowledge this for the interactive mode to be enabled. This is our negotiation handshake.				

Accepted start of Interactive mode	From:	PC	Data:	None
A	IA + 0x0D0A			
Sent to the rowing computer to acknowledge the start of interactive mode. The rowing computer will configure its keypad to be sent as commands to the PC instead of controlling the functions of it self.				

End Interactive mode	From:	PC	Data:	None
A	IE + 0x0D0A			
Sent to the rowing computer to cancel interactive mode, button controls of the rowing computer return to normal functionality.				

Keypad press detection sends the following message, label in “( )” is for overlay reference of which keypad button this packet will be referenced against.

Interactive keypad press: RESET	From:	S4	Data:	None
A	KR + 0x0D0A			
The rowing computer is about to reset its settings, this is the one function which can over ride and control the rowing computer, all other keypad interaction is sent as messages. Interactive mode will be cancelled once this message has been sent.				

Interactive keypad press: 1 (Units)	From:	S4	Data:	None
A	K1 + 0x0D0A			
Keypad entry detected for this button.				

Interactive keypad press: 2 (Zones)		From:	S4	Data:	None
A	K2 + 0x0D0A				
Keypad entry detected for this button.					

Interactive keypad press: 3 (Workout Programs)		From:	S4	Data:	None
A	K3 + 0x0D0A				
Keypad entry detected for this button.					

Interactive keypad press: 4 (Up arrow)		From:	S4	Data:	None
A	K4 + 0x0D0A				
Keypad entry detected for this button.					

Interactive keypad press: 5 (OK)		From:	S4	Data:	None
A	K5 + 0x0D0A				
Keypad entry detected for this button.					

Interactive keypad press: 6 (Down arrow)		From:	S4	Data:	None
A	K6 + 0x0D0A				
Keypad entry detected for this button.					

Interactive keypad press: 7 (Advanced)		From:	S4	Data:	None
A	K7 + 0x0D0A				
Keypad entry detected for this button.					

Interactive keypad press: 8 (Stored Programs)		From:	S4	Data:	None
A	K8 + 0x0D0A				
Keypad entry detected for this button.					

Interactive keypad press: 9 (Hold/Cancel)		From:	S4	Data:	None
A	K9 + 0x0D0A				
Keypad entry detected for this button.					

Workouts

Workouts are configured with at least 1 packet for single distance and duration workouts while interval workouts require multiple messages to define the total number of intervals. Should any message be incorrectly formatted then an ERROR will be issued, failing to complete an interval workout will result in it being scrapped at the next PING packet.

This means the application will have a second to download and confirm the whole of a interval workout, the PING transmit timer will be set to 0 at the start of the interval workout programming. Because of the need to use the PING and rowing is NOT recommended during workout programming the application must warn the user to stop all rowing and wait for the PING messages before attempting to load a workout program.

It is also recommended that the rowing computer is RESET prior to downloading any workout, a PING after a reset will indicate the rowing computer is ready again for data.

Single workouts

Define a distance workout		From:	PC	Data:	ACH
W	SI + X + YYYY + 0x0D0A				
This packet will configure a single distance workout. X is the units of the workout: 1 –Meters 2 – Miles 3 – Km’s 4 – Strokes YYYY is a 16bit ACH value. (This value is limited to 64,000 meters, which equates to 39.8 miles or 5,000 strokes, using values outside these will cause an ERROR reply, 0000 is also invalid). When X = 1, 2 or 3: this value is in Meters, the display value for miles is a conversion and valid values are 0001 to FA00. Note: Km’s & Miles are internally formatted to 3 decimal places, correct by a factor of 1,000 prior to sending data, e.g. 1mile =1608meters. (The S4 performs the conversion as (YYYY*250)/402, this is simplified floating type maths and far quicker to perform for a micro controller while keeping accurate, < 0.1% error). When X = 4 this value is the number of strokes and valid values are 0001 to 1388.					

Define a duration workout		From:	PC	Data:	ACH
W	SU + YYYY + 0x0D0A				
This packet will configure a single distance workout. YYYY is a 16bit ACH value in seconds. (0001 to 4650). (This value is limited to 5 Hours, which is 18,000 seconds, using values outside this range will cause an ERROR reply, 0000 is also invalid).					

Interval workouts

Start a interval distance workout		From:	PC	Data:	ACH
W	II + X + YYYY + 0x0D0A				

This packet will configure the first interval distance workout; additional intervals or a completion of interval packet must be sent to complete the workout.

X is the units of the workout:
1 –Meters 2 – Miles 3 – Km's 4 – Strokes

YYYY is a 16bit ACH value.
(This value is limited to 64,000 meters, which equates to 39.8 miles or 5,000 strokes, using values outside these will cause an ERROR reply, 0000 is also invalid).

When X = 1, 2 or 3: this value is in Meters, the display value for miles is a conversion and valid values are 0001 to FA00. Note: Km's & Miles are internally formatted to 3 decimal places, correct by a factor of 1,000 prior to sending data, e.g. 1mile =1608meters.
(The S4 performs the conversion as (YYYY*250)/402, this is simplified maths and far quicker to perform for a micro controller while keeping accurate, < 0.1% error).

When X = 4 this value is the number of strokes and valid values are 0001 to 1388.

Start a interval duration workout		From:	PC	Data:	ACH
W	IU + YYYY + 0x0D0A				

This packet will configure the first interval distance workout; additional intervals or a completion of interval packet must be sent to complete the workout.

YYYY is a 16bit ACH value in seconds. (0001 to 4650).
(This value is limited to 5 Hours, which is 18,000 seconds, using values outside this range will cause an ERROR reply, 0000 is also invalid).

Add/End an interval to a workout		From:	PC	Data:	ACH
W	IN + XXXX + YYYY + 0x0D0A				

This packet will add an interval to a workout already started or end the workout.

XXXX is the rest interval and is a 16bit ACH value in seconds. (0001 to 0E10).
(This value is limited to 60 minutes, which is 3,600 seconds; using values outside this range will cause an ERROR reply, 0000 is also invalid).

YYYY is dependent on the workout type and follows the same format as in WII and WIU.

To send an end of workout packet, XXXX must equal FFFF; this is the only accepted completion of the workout setup. (YYYY maybe omitted or sent as FFFF as well)

8 additional rest/work intervals can be added then the end interval packet must be sent.

Memory Map

These memory maps are unique to each version of code, attempting to read outside a given range will result in an error, some locations will be included which are of no use to the PC application. In this way it is possible to determine just about everything the rowing computer is doing, calculating or displaying. It then leads for the end designer to choose how much or little of this information is worth reading, all locations are 8bit, multiple locations used for 16, 24 and 32bit values.

Series 4, Version 2.00

The following memory locations are available to the user for reading; this is in a mono space font to ensure clarity. Other locations not specified are unavailable for reading. Please refer to “Water Rower Series 4 Rowing Algorithm.doc” and “WaterRower series 4 Maths.xls” for additional information on some of these variables. A lot of the timers count 1bit per 25mS of actual time, remember this for ALL maths, otherwise things can be confusing of how much time say a stroke was done IN.

PLEASE NOTE: This memory map is subject to change during programming of the protocol, this maybe due to simplification of the memory structure or removal/addition of variables found to be of no use or able to give additional information.

These registers can be read to determine the current state of the display; they are formed of 8 separate true or false flags. These flags are defined at the end of the memory map.

```
;
; these flags are backed up when in menus
;
fcycle                  03d     ; cycling, alternating and other flags.
fextended               03e     ; working and workout control flags
ffunc_menu              03f     ; menu functions, cleared accordingly


;
; Display flags registers, there will be alot of these.
;
fint_flags1             040     ; intensity
fint_flags2             041     ;
fdist_flags             042     ; distance
fdist_flags2            043     ; distance
fprog_flags             044     ; program
fdur_flags              045     ; duration
fhr_flags               046     ; heartrate
fsr_flags               047     ; stroke rate
fzone_flags             048     ; zones (z1-z5 lo and hi)
fmisc_flags             049     ; zone words and misc
```

```
;
; Screen mode variable
; --------------------
;
; Mode--\ /--Sub screens (byte makeup "xx"h)
;       | |
;       0 x     Nomral Display, setup by flags accordingly
;
;       1 *     Unit Screen
;       + 0             Intensity Units Settings Window
;       + 1             Average Intensity Settings Window
;       + 2             Distance Settings Window
;
;       2 *     Zone Screen
;       + 0             Heartrate Zone Settings
;       + 1             Intensity Zone Settings
;       + 2             Strokerate Zone settings
;
;       3 *     Workout Screen
;       + 0             Distance Workout Settings
;       + 1             Duration Workout Settings
;       + 2             Distance Interval Workout Settings
;       + 3             Duration Interval Workout Settings
;
;       4 *     Advanced Screen
;       + 0             1) Store Workout
;       + 1             2) Load Workout
;       + 2             3) Projected Duration
;       + 3             4) Projected Distance
;       + 4             5) Ratio
;       + 5             6) Advanced Heart Rate Analysis
;       + 6             7) Prognostics
;       + 7             8) Tank Volume Input
;       + 8             9) Total Distance Rowed / Issue No.
;
;       5 *     Stored Screen (count through stores 1-9, roll around)
;
;       F *     System control function holds
;       + 0             Interval hold, at end of workout for cancel
;       + 1             Interval hold, in rest period, lock system
;       + E             Interactive mode in operation, all button presses send to USB
;       + F             TEST DISPLAY MODE - required to get routines
;                       back to the test routines
;
;               x - dont care           + - same screen * - multiple options
;
screen_mode             00d     ; see above
screen_sub_mode 00e     ; for extra sub menu selections (as required)
screen_interval 00f     ; current number of intervals remaining


;
; Distance variables
;
ms_distance_dec 054     ; 0.1m count  (only counts up from 0-9)
ms_distance_low 055     ; low byte of meters
ms_distance_hi          056     ; hi byte of meters and km (65535meters max)

;
; this is the displayed distance
;
distance_low            057     ; low byte of meters
distance_hi             058     ; hi byte of meters and km (65535meters max)
test_count              059     ; used by test routines

;
; CLock count down, this is 16bit value.
;
clock_down_dec          05a     ; seconds 0.9-0.0
clock_down_low          05b     ; low byte clock count down
clock_down_hi           05c     ; hi byte clock count down
```

```
;
; total distance meter counter - this is stored at switch off
;
total_dis_dec           080     ; dec byte of meters
total_dis_low           081     ; low byte of meters
total_dis_hi            082     ; hi byte of meters and km (65535meters max)
;
pins_per_xxcm           083     ; number of pin edges allowed to equal xxcm
distance_xxcm           084     ; number of cm per flaged xxcm no. of pins
;

; Locations between these are not used and should read as 0, these maybe used if space is required

;
kcal_watts_low          088
kcal_watts_hi           089
total_kcal_low          08a
total_kcal_hi           08b
total_kcal_up           08c




;
; zone values, this are kept even during a reset, but not on a power on
;
zone_hr_hi              090     ; hi setting for the heartrate
zone_hr_low             091     ; low setting for the heartrate

z_int_m_s_hmsb          092     ; hi setting for the intensity
z_int_m_s_hlsb          093     ; low setting for the intensity
z_int_m_s_lmsb          094     ; hi setting for the intensity
z_int_m_s_llsb          095     ; low setting for the intensity

z_int_mph_hmsb          096     ; hi setting for the intensity
z_int_mph_hlsb          097     ; low setting for the intensity
z_int_mph_lmsb          098     ; hi setting for the intensity
z_int_mph_llsb          099     ; low setting for the intensity

z_int_500m_hmsb 09a     ; hi setting for the intensity
z_int_500m_hlsb 09b     ; low setting for the intensity
z_int_500m_lmsb 09c     ; hi setting for the intensity
z_int_500m_llsb 09d     ; low setting for the intensity

z_int_2km_hmsb          09e     ; hi setting for the intensity
z_int_2km_hlsb          09f     ; low setting for the intensity
z_int_2km_lmsb          0a0     ; hi setting for the intensity
z_int_2km_llsb          0a1     ; low setting for the intensity

zone_sr_hi              0a2     ; hi setting for the strokerate
zone_sr_low             0a3     ; low setting for the strokerate


;
; advanced workout prognostic variables
;
prognostic_sech 0a4     ; prognostic seconds hi and low
prognostic_secl 0a5     ; 16 bit value
prognostic_cmsu 0a6     ; prognostic cm/s (multiplied by 100)
prognostic_cmsh 0a7     ; this is the constance for the maths.
prognostic_cmsl 0a8     ; results is a 24bit value

;
; tank volume in liters
;
tank_volume             0a9     ; volume of water in tank
```

```
;
; BANK 1
; ======
;

;
; Stroke counter
;
strokes_cnt_low 140     ; low byte count
strokes_cnt_hi          141      ; high byte count
stroke_average          142      ; average time for a whole stroke
stroke_pull             143      ; average time for a pull (acc to dec)
```

Stroke_pull is first subtracted from stroke_average then a modifier of 1.25 multiplied by the result to generate the ratio value for display.

```
;
; Meters per second registers
;
m_s_low_total           148      ; total distance per second in cm low byte
m_s_hi_total            149      ; total distance per second in cm hi byte
m_s_low_average 14a     ; instant average distance in cm low byte
m_s_hi_average          14b      ; instant average distance in cm hi byte
m_s_stored              14c      ; no. of the stored values.
m_s_projl_avg           14d      ; all average for projected distance/duration maths
m_s_projh_avg           14e      ; all average for projected distance/duration maths


;
; Zone maths - these are used each time the routine is ran
;
zone_hi_msb             190      ; high byte msb
zone_hi_lsb             191      ; high byte lsb
zone_low_msb            192      ; low byte msb
zone_low_lsb            193      ; low byte lsb
zone_sec_msb            194      ; sector size to perform scaling msb
zone_sec_lsb            195      ; sector size to perform scaling lsb
zone_val_msb            196      ; value of operation msb
zone_val_lsb            197      ; value of operation lsb
zone_range_mhi          198      ; range scaled for hi byte msb
zone_range_lhi          199      ; range scaled for hi byte lsb
zone_range_mlow 19a     ; range scaled for low byte msb
zone_range_llow 19b     ; range scaled for low byte lsb
zone_range_mval 19c     ; range scaled for the input value byte msb
zone_range_lval 19d     ; range scaled for the input value byte lsb

;
; stored values for the zone maths above (these are pre display values)
;
zone_hr_val             1a0      ; heart rate stored value
zone_m_s_hval           1a1      ; m/s hi stored value (cm/s)
zone_m_s_lval           1a2      ; m/s low stored value (cm/s)
zone_mph_hval           1a3      ; mph hi stored value (xx.x)
zone_mph_lval           1a4      ; mph low stored value (xx.x)
zone_500m_hval          1a5      ; 500m hi stored value (sec's)
zone_500m_lval          1a6      ; 500m low stored value (sec's
zone_2km_hval           1a7      ; 2km hi stored value (sec's)
zone_2km_lval           1a8      ; 2km low stored value (sec's)
zone_sr_val             1a9      ; stroke rate stored value
```

```
;
; Interval's
; ----------
;
; These are the interval timing's in use or being programmed.
;
workout_work1_l 1b0
workout_work1_h 1b1
workout_rest1_l 1b2
workout_rest1_h 1b3
workout_work2_l 1b4
workout_work2_h 1b5
workout_rest2_l 1b6
workout_rest2_h 1b7
workout_work3_l 1b8
workout_work3_h 1b9
workout_rest3_l 1ba
workout_rest3_h 1bb
workout_work4_l 1bc
workout_work4_h 1bd
workout_rest4_l 1be
workout_rest4_h 1bf
workout_work5_l 1c0
workout_work5_h 1c1
workout_rest5_l 1c2
workout_rest5_h 1c3
workout_work6_l 1c4
workout_work6_h 1c5
workout_rest6_l 1c6
workout_rest6_h 1c7
workout_work7_l 1c8
workout_work7_h 1c9
workout_rest7_l 1ca
workout_rest7_h 1cb
workout_work8_l 1cc
workout_work8_h 1cd
workout_rest8_l 1ce
workout_rest8_h 1cf
workout_work9_l 1d0
workout_work9_h 1d1

workout_inter           1d9     ; No work workout intervals

;
; used to generate the display clock
;
display_sec_dec 1e0     ; seconds 0.0-0.9
display_sec             1e1     ; seconds 0-59
display_min             1e2     ; minutes 0-59
display_hr              1e3     ; hours 0-9 only

;
; workout total times/distances/limits
;
workout_timel           1e8     ; total workout time
workout_timeh           1e9
workout_ms_l            1ea     ; total workout m/s
workout_ms_h            1eb
workout_strokel 1ec     ; total workout strokes
workout_strokeh 1ed
workout_limit_h 1ee     ; this is the limit value for workouts
workout_limit_l 1ef

;
; heart rate analysis variables
;
hr_above_tenths 1f0     ; time above heart zone
hr_above_low            1f1
hr_above_hi             1f2
hr_in_tenths            1f3     ; time in heart zone
hr_in_low               1f4
hr_in_hi        1f5
hr_below_tenths 1f6     ; time below heart zone
hr_below_low            1f7
hr_below_hi             1f8
hr_peak                 1f9     ; peak heartrate (always)
```

Flags

Flags are Boolean conditions, these can change very fast and hence some will be of no use to the PC application and should be filtered out, however many are useful to determine the current settings of the screen and rowing computer.

These flags not only control the display but much of the maths equations that are performed for the display values. Bit 7 is the MSB of the byte and bit 0 is the LSB of the byte.

```
;
; Cycling flags for the display - backup when in menu
;
fcycle_3_7              fcycle,0; set = 2 seconds display, clear = 8 sec.
fcycle_dur              fcycle,1; \ when set use fcycle_3_7 for correct display,
fcycle_dis              fcycle,2; | when fcycle_3_7 is set, display alternate
fcycle_ratio            fcycle,3; / when clear display standard value
fcycle_next_set fcycle,4; set on every flash on and off routine
fcycle_setup            fcycle,5; step through the current setup
fcycle_analysis fcycle,6; display cycle for heartrate analysis


;
; buzzer and other functions to use - backup when in menu
;
fbuzzer_2hz             ffunc_menu,0    ; sound the buzzer @ 2hz
fbuzzer_1hz             ffunc_menu,1    ; sound the buzzer @ 1hz
fzone_up        ffunc_menu,2    ; zone scroll up routine
fzone_down              ffunc_menu,3    ; zone scroll down routine
flast_z_or_w            ffunc_menu,4    ; which was configured last, zone (0) or workout (1)
fclock_down             ffunc_menu,5    ; set to display alt mm:ss for counting down


;
; for extended zones and workout modes.
;
fzone_hr        fextended,0     ; working in heartrate zone
fzone_int               fextended,1     ; working in intensity zone
fzone_sr        fextended,2     ; working in strokerate zone
fprognostics            fextended,3     ; prognostics active.
fworkout_dis            fextended,4     ; workout distance mode
fworkout_dur            fextended,5     ; workout duration mode
fworkout_dis_i          fextended,6     ; workout distance interval mode
fworkout_dur_i          fextended,7     ; workout duration interval mode
```

Flashing flags are not include due to them being just a screen effect, all displaying flags are set first by these regardless of them flashing or not.

```
;
; Display flags:
; -------------
;
; Intensity window   (C2_ HAS BEEN ADJUSTED FROM 5 TO 4 FOR TESTING)
;
fint_fg_ratio           fint_flags1,0   ; set when to turn on or if flashed is clear and flash is set
fint_fg_adj             fint_flags1,1   ;  "
fint_fg_kgs             fint_flags1,2   ;  "
fint_fg_int             fint_flags1,3   ;  "
fint_fg_c1              fint_flags1,4   ;  "
fint_fg_c2_             fint_flags1,5   ;  "
fint_fg_d1              fint_flags1,6   ;  "
fint_fg_dig_off  fint_flags1,7   ;  "

fint_fg_m_s             fint_flags2,0   ; set when to turn on or if flashed is clear and flash is set
fint_fg_mph             fint_flags2,1   ;  "
fint_fg_500m            fint_flags2,2   ;  "
fint_fg_2km             fint_flags2,3   ;  "
fint_fg_watts           fint_flags2,4   ;  "
fint_fg_cal_hr          fint_flags2,5   ;  "
fint_fg_litres          fint_flags2,6   ;  "
fint_fg_avg             fint_flags2,7   ;  "


;
; Distance window
;
fdist_fg_proj           fdist_flags,0   ; set when to turn on or if flashed is clear and flash is set
fdist_fg_dist           fdist_flags,1   ;  "
fdist_fg_meters  fdist_flags,2   ;  "
fdist_fg_miles          fdist_flags,3   ;  "
fdist_fg_km             fdist_flags,4   ;  "
fdist_fg_stks           fdist_flags,5   ;  "

fdist_fg_dig_offfdist_flags,7   ;  "

fdist_fg_d3             fdist_flags2,6  ; set when to turn on or if flashed is clear and flash is set


;
; Program Window
;
fprog_fg_load           fprog_flags,0   ; set when to turn on or if flashed is clear and flash is set
fprog_fg_wkout          fprog_flags,1   ;  "
fprog_fg_store          fprog_flags,2   ;  "
fprog_fg_inter          fprog_flags,3   ;  "
fprog_fg_prog           fprog_flags,4   ;  "
fprog_fg_adv            fprog_flags,5   ;  "
fprog_fg_dig_offfprog_flags,6   ;  "
fprog_fg_digits  fprog_flags,7   ;  "


;
; Duration Window
;
fdur_fg_dur             fdur_flags,0    ; set when to turn on or if flashed is clear and flash is set
fdur_fg_proj            fdur_flags,1    ;  "
fdur_fg_c2              fdur_flags,2    ;  "
fdur_fg_c3              fdur_flags,3    ;  "
fdur_fg_d2              fdur_flags,4    ;  "
fdur_fg_dig_off  fdur_flags,7    ;  "
```

```
;
; Heart Rate Window
;
fhr_fg_hr               fhr_flags,0      ; set when to turn on or if flashed is clear and flash is set
fhr_fg_progn            fhr_flags,1      ;  "
fhr_fg_bs_m             fhr_flags,2      ;  "
fhr_fg_percent          fhr_flags,3      ;  "
fhr_fg_hrt_sym          fhr_flags,4      ;  "
fhr_fg_dig_off          fhr_flags,7      ;  "


;
; Stroke Rate Window
;
fsr_fg_sr               fsr_flags,0      ; set when to turn on or if flashed is clear and flash is set
fsr_fg_srk_m            fsr_flags,1      ;  "
fsr_fg_half             fsr_flags,2      ;  "
fsr_fg_dig_off          fsr_flags,7      ;  "


;
; Zone Window (z1-z5, hi and lo)
;
fzone_fg_lo             fzone_flags,0    ; set when to turn on or if flashed is clear and flash is set
fzone_fg_z1             fzone_flags,1    ;  "
fzone_fg_z2             fzone_flags,2    ;  "
fzone_fg_z3             fzone_flags,3    ;  "
fzone_fg_z4             fzone_flags,4    ;  "
fzone_fg_z5             fzone_flags,5    ;  "
fzone_fg_hi             fzone_flags,6    ;  "


;
; Zone words and Misc Windows
;
fzone_fg_work           fmisc_flags,0    ; set when to turn on or if flashed is clear and flash is set
fzone_fg_rest           fmisc_flags,1    ;  "
fmisc_fg_lowbat fmisc_flags,2    ;  "
fmisc_fg_pc             fmisc_flags,3    ;  "
fmisc_fg_line           fmisc_flags,4    ;  "
fmisc_fg_mmc_cd fmisc_flags,5    ;  "
fmisc_fg_mmc_up fmisc_flags,6    ;  "
fmisc_fg_mmc_dn fmisc_flags,7    ;  "
```