Concept2

Performance Monitor
CSAFE Communication Definition

Filename: Concept2 PM CSAFE Communication Definition.doc

Revision: 0.13
2/24/2021 9:42:00 AM

Concept2
105 Industrial Park Drive
Morrisville, VT 05661
802-888-5226 (Voice)
802-888-6331 (Fax)
rowing@concept2.com

Table of Contents

LIST OF FIGURES **4**

LIST OF TABLES **5**

PURPOSE AND SCOPE **6**

DOCUMENT HISTORY 6
RELATED DOCUMENTS 6

OVERVIEW **7**

INTERFACES **7**

CSAFE PROTOCOL DEFINITION **8**

FRAME STRUCTURE 8
FRAME CONTENTS 9
Command Format 9
Response Format 10
PM MANUFACTURER INFORMATION 11
PM EXTENSIONS 11

LINK LAYER DEFINITION **12**

USB 12
SMART BLUETOOTH 13

PUBLIC CSAFE **39**

FEATURES 39
Public CSAFE Default Configuration 39
Public CSAFE State Machine Operation 39
Public CSAFE Unsupported Features 40
Programmed Workout Parameter Limits 41
COMMAND LIST 42
Public Short Commands 42
Public Long Commands 43
C2 Proprietary Short Commands 45
C2 Proprietary Long Commands 46
SETTING UP AND PERFORMING WORKOUT 47
SPECIAL CONSIDERATION 49
ScreenType Commands 49
Maximum Block Size Commands 49
Fixed Block Size Command Responses 49
COMMAND LIST 49
C2 Proprietary Short Get Configuration Commands 49
C2 Proprietary Long Get Configuration Commands 52
C2 Proprietary Short Get Data Commands 54
C2 Proprietary Long Get Data Commands 57
C2 Proprietary Short Set Configuration Commands 59
C2 Proprietary Short Set Data Commands 59
C2 Proprietary Long Set Configuration Commands 60
C2 Proprietary Long Set Data Commands 63
SETTING UP AND PERFORMING WORKOUT 67

SAMPLE FUNCTIONALITY **69**

PUBLIC CSAFE WORKOUT CONFIGURATION 69

Fixed Distance....69
2000m/500m splits, power goal of 200 watts....69
Fixed Time....69
20:00/4:00 splits, power goal of 100 watts....69
Predefined....70
Standard List Workouut #3....70
PROPRIETARY CSAFE WORKOUT CONFIGURATION....70
JustRow....70
Fixed Distance....70
2000m/500m splits....71
Fixed Time....71
20:00/4:00 splits....71
Fixed Calories....72
100 Cals/20 Cal splits....72
Fixed Distance Interval....73
500m/:30 rest....73
Fixed Time Interval....73
2:00/:30 rest....73
Fixed Calorie Interval....74
25c/1:00 rest....74
Variable Interval....75
v500m/1:00r…4....75
Variable Interval Undefined Rest....77
v100m…2....77
Fixed Interval Undefined Rest....79
CSAFE MISCELLANEOUS....79
Terminate Workout....79

APPENDIX A....81

ENUMERATED VALUES....81
Operational State....81
Erg Machine Type....81
Workout Type....81
Interval Type....82
Workout State....82
Rowing State....82
Stroke State....82
Workout Duration Type....83
Display Units Type....83
Display Format Type....83
Workout Number....83
Workout Programming Mode....83
Stroke Rate State....84
Start Type....84
Race Operation Type....84
Race State....84
Race Type....84
Race Start State....85
Screen Type....86
Screen Value (Workout Type)....86
Screen Value (Race Type)....87
Screen Value (CSAFE Type)....87
Screen Status....88
Status Type....88
Display Update Rate....88
Wireless Channel Flags....88

Log Structure Identifiers 89
CPU Speed/Tick Rate 89
Game ID 90
GAME IDENTIFIER / VERIFIED INFORMATION 90
COMMUNICATING WITH THE PM USING CSAFE COMMANDS 91
Retrieving Heartrate Belt Information 91
Commanding the PM5 to Pair with a known Heartrate Belt 91

APPENDIX B 92

DATA REPRESENTATION 92
Time and Distance Displayed 92
Time and Distance Stored in Workout Log 92
DATA CALCULATION 92
Display 92
Workout Log 92
PACE CONVERSIONS 92
Watts <-> Pace 92
Calories/Hr <-> Pace 93
Pace <-> /500m Pace 93
DATA CONSTRUCTION 93
Two Byte Data 93
Three Byte Data 93
Four Byte Data 94
DATA DECONSTRUCTION 94
Two Byte Data 94
Three Byte Data 95
Four Byte Data 95

APPENDIX C 97

ROW/SKI ERG STANDARD LIST WORKOUTS 97
ROW/SKI ERG CUSTOM LIST WORKOUTS 97
BIKE ERG STANDARD LIST WORKOUTS 97
BIKE ERG CUSTOM LIST WORKOUTS 97

APPENDIX D 98

ERROR CODE LIST 98

APPENDIX E 148

PM STATE TRANSITIONS 148

List of Figures

FIGURE 1 - STANDARD FRAME FORMAT 8
FIGURE 2 - EXTENDED FRAME FORMAT 8
FIGURE 3 - LONG COMMAND FORMAT 9
FIGURE 4 - SHORT COMMAND FORMAT 10
FIGURE 5 - RESPONSE FRAME CONTENTS FORMAT 10
FIGURE 6 - INDIVIDUAL COMMAND RESPONSE FORMAT 10
FIGURE 7 – PUBLIC CSAFE STATE MACHINE DIAGRAM 39
FIGURE 8 – EXAMPLE PUBLIC CSAFE PM WORKOUT SETUP AND PROGRESS MONITORING 47
FIGURE 9 – EXAMPLE PUBLIC CSAFE PM SUCCESSIVE JUSTROW WORKOUTS 48
FIGURE 10 – MAXIMUM BLOCK SIZE COMMANDS 49
FIGURE 11 – FIXED BLOCK SIZED COMMAND RESPONSES 49
FIGURE 12 – EXAMPLE PROPRIETARY CSAFE PM WORKOUT SETUP AND PROGRESS MONITORING 67

List of Tables

TABLE 1 - DOCUMENT MODIFICATION HISTORY 6
TABLE 2 - RELATED DOCUMENTS 6
TABLE 3 – COMMUNICATION INTERFACE VERSUS FUNCTIONALITY 7
TABLE 4 - EXTENDED FRAME ADDRESSING 8
TABLE 5 - UNIQUE FRAME FLAGS 9
TABLE 6 - BYTE STUFFING VALUES 9
TABLE 7 - COMMAND FIELD TYPES 10
TABLE 8 - RESPONSE FIELD TYPES 10
TABLE 9 – RESPONSE STATUS BYTE BIT-MAPPING 11
TABLE 10 - CSAFE CONCEPT2 PM INFORMATION 11
TABLE 11 - PM-SPECIFIC CSAFE COMMAND WRAPPERS 11
TABLE 12 - PM PROPRIETARY CSAFE COMMAND WRAPPERS 12
TABLE 13 - PM USB DEFINITIONS 12
TABLE 14 – C2 PM BTS PERIPHERAL : ATTRIBUTE TABLE 13
TABLE 15 – C2 MULTIPLEXED INFORMATION: DATA DEFINITIONS 28
TABLE 16 – PM PUBLIC CSAFE PROTOCOL DEFAULTS 39
TABLE 17 - PM UNSUPPORTED PUBLIC CSAFE PROTOCOL FEATURES 40
TABLE 18 – PM3/PM4 WORKOUT CONFIGURATION PARAMETER LIMITS 41
TABLE 19 – PM5 WORKOUT CONFIGURATION PARAMETER LIMITS 41

Purpose and Scope

This document contains the CSAFE communications definition for applications communicating with Performance Monitor (PMs) using any of the available interfaces: 1. USB, 2. Blue Tooth Smart 3. RS485. Information in this document combined with the documents referred to in Table 2 should provide the developer with sufficient information to create applications that communicate with the PM over any communication interface.

Document History

Table 1 - Document Modification History

Edit Date	Engineer	Description of Modification
3/11/19	Mark Lyons	Initial outline created. V0.01
4/2/19	Mark Lyons	Numerous updates including list of errors. V0.02
4/5/19	Mark Lyons	Numerous updates including more sample functionality. V0.03
4/10/19	Mark Lyons	Numerous updates including more sample functionality. V0.04
4/10/19	Mark Lyons	Add appendix items on construction/desconstruction of multi-byte values. V0.05
4/16/19	Mark Lyons	More sample functionality. V0.06
8/16/19	Mark Lyons	More sample functionality (Public CSAFE). V0.07
6/8/20	Mark Lyons	Added definition for Fixed Interval Undefined Rest workouts V0.08
8/25/20	Mark Lyons	Added screen value definitions and new command V0.09
9/3/20	Mark Lyons	Added 2 additional Device Info characteristics V0.10
9/11/20	Mark Lyons	Added CPU speed/tick rate enumeration V0.11
2/15/21	Mark Lyons	Added explanation of setting SplitDurationDistance in Variable Interval Workouts with Undefined Rest V0.12
2/23/21	Mark Lyons	Added Appendix E with PM state transitions,V0.13

Related Documents

Table 2 - Related Documents

Document Title	Document Number - Date
CSAFE Protocol Technical Specification, V1.x	http://www.fitlinxx.com/csafe/
Concept2 PM Bluetooth Smart Communication Interface Definition.doc	

Overview

Communication with the Performance Monitor (PM) is based on the CSAFE prototocol. The CSAFE protocol was created to facilitate communication between fitness equipment and a host computer. The "public" CSAFE protocol implementation provides a basic framework for configuring workouts, and monitoring progress of those workouts, through a "state machine" style mechanism. So in order to be compatible with existing fitness equipment controllers, a public CSAFE implementation has been included with the PM.

Since the PM is substantially more programmable then the public CSAFE protocol can accommodate, a more expansive Concept2 proprietary CSAFE protocol implementation has also been included. It's important to understand that a developer must use either the public CSAFE protocol or the full proprietary CSAFE protocol (e.g., simultaneous use of both protocols is not supported). Note that the public CSAFE protocol does include some very limited proprietary commands deemed necessary for basic operation. The full proprietary protocol has limited availability on some interfaces without special authenticating information that is made available by Concept2 to qualified developers.

Interfaces

There are as many as three communications interfaces available depending on which generation Performance Monitor (PM). All performance monitor models (PM3/PM4/PM5) support a USB device interface, typically used for connecting to host computers. The PM4 and PM5 also support an RS485 interface typically used when interconnecting two or more monitors for racing or multi-machine workouts. The PM5 supports a Bluetooth Smart interface typically used when connecting to mobile device applications.

Table 3 – Communication Interface versus Functionality

Interface	PM3	PM4	PM5	Description	Application
USB Device	x	x	x	Access to public and full proprietary CSAFE if authenticated; otherwise, access to public and limited proprietary CSAFE	Connecting to host computer using Type A-B cable; connecting to mobile device using Type B-MicroB or custom
RS485		x	x	Access to public and full proprietary CSAFE if authenticated; otherwise, access to public and limited proprietary CSAFE	Connecting multiple PMs using RJ45/Ethernet cables
Bluetooth Smart			x	Access to public and full proprietary CSAFE	Connecting to mobile devices

All three interfaces utilize the CSAFE protocol to exchange commands and responses intended to configure and monitor PM operations. Each interface transports the CSAFE protocol using different link layer protocols. Adherence to these link layer protocols is equally as important as the CSAFE protocol in achieving successful communication with the PM.

CSAFE Protocol Definition

In the CSAFE protocol, communication between the primary and the secondary device is accomplished using two basic frame types: standard frame and extended frame. The standard frame provides no provisions for slave-to-slave communication or multi-drop network configurations, as device addressing is implicit. The PM application requires explicit device addressing for numerous scenarios (as provided by the extended frame format) so that both frame types will be handled for our implementation. In general, the secondary device only speaks when responding to a primary's request. Certain exceptions may be made in very specific circumstances.

The standard frame is defined as a stream of bytes with the structure shown in Figure 1. No explicit addressing information is present in the standard frame and its use is appropriate for a primary communcationing with a single secondary. The frame components (start flag, checksum, stop flag) provide a structure that allows unambi guously locating, validating, and interpreting a frame within a stream of bytes. The start flags and stop flag are unique values used to delineate the frame and, therefore, cannot appear in the frame contents or the checksum. A byte-stuffing technique is employed to ensure that these unique bytes do not occur elsewhere in the frame. A checksum is included in the frame to allow both the master and slave devices to verify the integrity of the "Frame Contents". Neither an acknowledgement (ACK) nor negative acknowledgement (NAK) at the frame level is an integral part of the protocol.

Figure 1 - Standard Frame Format

The extended frame is defined as stream of bytes with the structure shown in Figure 2. Note that the standard and extended frames are identical with the exception of the frame-unique start flag and the device address information. The extended frame is appropriate for a primary communicating with two or more secondarys.

Figure 2 - Extended Frame Format

Frame Structure

The frame structure is a stream of bytes with a unique start byte, optional addressing, frame contents (e.g., commands and responses), a checksum and a unique stop byte. The unique start and stop byte values are shown in Table 5. In order to ensure that these start and stop values do not appear anywhere in the frame, the primary and secondary devices perform "byte-stuffing" and "byte-unstuffing" on the byte stream (i.e., frame contents including extended frame addresses and checksum). This technique can be performed "on the fly" without impacting the data stream buffering requirements, since the extra bytes only exist on the data link.

The extended frame addressing rules are summarized in Table 4.

Table 4 - Extended Frame Addressing

Address	Description
0x00	PC Host (pirmary)
0x01 – 0xFC	<unassigned>
0xFD	Default secondary address
0xFE	Reserved for expansion
0xFF	"Broadcast" accepted by all

	secondarys

The "byte-stuffing" algorithm simply substitutes two bytes for each of the unique bytes listed in Table 5. The unique Byte Stuffing Flag is followed by a 0x00, 0x01, 0x02, or 0x03 as shown in Table 6 depending on the byte being replaced. The impact of this technique on the data link is that the frame size could increase in size by a factor of two in the worst case.

Table 5 - Unique Frame Flags

Description	Value
Extended Frame Start Flag	0xF0
Standard Frame Start Flag	0xF1
Stop Frame Flag	0xF2
Byte Stuffing Flag	0xF3

Table 6 - Byte Stuffing Values

Frame Byte Value	Byte-Stuffed Value
0xF0	0xF3, 0x00
0xF1	0xF3, 0x01
0xF2	0xF3, 0x02
0xF3	0xF3, 0x03

The frame beginning and end are designated by the unique Start and Stop bytes. If a Start or Stop byte is missed, the frame is discarded and frame resynchronization occurs at the beginning of the next frame. Once a full frame is received and all "byte-unstuffing" is performed, a one-byte checksum is computed with byte-by-byte XORing of the frame contents (e.g., excluding start/stop flags and addresses) to verify frame integrity. The frame definition does not explicitly place any limits on the frame length. Because the entire frame contents must be buffered before computing the checksum, memory resources on the secondary devices typically establish the restrictions on frame length. For CSAFE protocol compatibility, the following frame length restrictions are invoked for the PM physical link:

1. A maximum frame size of 120 bytes including start/stop flags, checksum and byte stuffing
2. All flow control handled natively as part of physical link

Frame Contents

The CSAFE protocol transports frame content data consisting of both commands and responses. The only restrictions on the frame contents relate to length of frame and the requirement that individual commands/ responses do not straddle a frame boundary (i.e., no partial commands/responses within a frame). The following sections detail the command and response formats.

Command Format

All commands have one of two basic formats: long command or short command. Long commands are those including command data while short commands are command only. The command is represented by a single byte with the command address space partitioned equally (i.e., long commands have MS bit clear and short commands have MS bit set). Figure 3 and Figure 4 illustrate the long and short command formats, respectively.

Figure 3 - Long Command Format

Long Command	Data Byte Count	Data

Figure 4 - Short Command Format

Short Command

In the long command format, the Long Command and Data Byte Count fields are single bytes. The Data Byte Count field determines the Data field size. The short command format consists solely of the single byte Short Command. Table 7 summarizes the command field types for both the long and short commands. Note that the command formats allows a long command with a Data Byte Count of 0 and no bytes in the Data field. The virtue of the Data Byte Count field in the long command is to allow slave devices to handle unrecognized commands by merely disregarding the command and its data, while continuing to process succeeding commands within the same frame.

Table 7 - Command Field Types

Description	Size (Bytes)	Value
Long Command	1	0x00 – 0x7F
Short Command	1	0x80 – 0xFF
Data Byte Count	1	0 - 255
Data	Variable	0 - 255

Multiple complete commands can be included in a single frame, but no partial commands or responses are allowed. When sending a frame consisting of multiple commands to a secondary device, the resulting response frame consists of multiple command responses.

Response Format

All responses have the same Frame Contents format as shown in Figure 5. The status byte is bit-mapped in order to indicate frame count, status and state machine state within the single byte. See Table 9 for status byte bit-mapping definitions.

Figure 5 - Response Frame Contents Format

Status	Command Response Data

Figure 6 - Individual Command Response Format

Command	Data Byte Count	Data

Table 8 - Response Field Types

Description	Size (Bytes)	Value
Status	1	0x00 – 0x7F
Command Response Data	Variable	0 - 255
Command	1	0x00 – 0xFF
Data Byte Count	1	1 - 255
Data	Variable	0 - 255

Table 9 – Response Status Byte Bit-Mapping

Description	Bit Mask	Notes
Frame Toggle	0x80	Toggles between 0 and 1 on alternate frames
Previous Frame Status	0x30	0x00: Ok 0x10: Reject 0x20: Bad 0x30: Not ready
State Machine State	0x0F	0x00: Error 0x01: Ready 0x02: Idle 0x03: Have ID 0x05: In Use 0x06: Pause 0x07: Finish 0x08: Manual 0x09: Off line

PM Manufacturer Information

Table 10 summarizes the Concept2 PM product-specific information CSAFE information.

Table 10 - CSAFE Concept2 PM Information

Product Information	Description
Manufacturer ID	22
Class Identifier	2
Model	PM3: 3, PM4: 4, PM5: 5
Maximum Frame Length	120 Bytes
Minimum Inter-frame Gap	50 msec.

PM Extensions

The PM extensions to the frame protocol involve utilizing one pre-defined custom command that serves as a "wrapper" for additional PM-specific commands. The one command is defined in Table 11. The one custom command wrapper is used to expand the CSAFE command set for additional configuration and data operations. See Public CSAFE section for a detailed explanation of the command wrapper implementation.

Table 11 - PM-Specific CSAFE Command Wrappers

Command Name	Command Identifier
CSAFE_SETUSERCFG1_CMD	0x1A

Additional PM proprietary extensions to the frame protocol involve utilizing four commands added to the existing public CSAFE protocol command set that serve as "wrappers" for the Concept2 proprietary command set. The four commands are defined in Table 12. The four command wrappers are used to functionally partition the PM command set space into "push" (i.e., set) and "pull" (i.e., get) operations for configuration and data. The use of these command wrappers allow the PM to support existing CSAFE protocol commands while introducing PM proprietary commands only accessible via the command set extension. Note that any wrapper can be used to access any proprietary command (e.g., there is no requirement to use the "set PM configuration" wrapper to access a configuration command).

Table 12 - PM Proprietary CSAFE Command Wrappers

Command Name	Command Identifier
CSAFE_SETPMCFG_CMD	0x76
CSAFE_SETPMDATA_CMD	0x77
CSAFE_GETPMCFG_CMD	0x7E
CSAFE_GETPMDATA_CMD	0x7F

Link Layer Definition

The CSAFE protocol transported over the USB or Smart Bluetooth link layer must still comply with the fundamental command and response behavior. Since both transport mechanisms can move data much more quickly than it takes the PM to respond, it is necessary for the "application" to not send additional commands until either a response has been received or the minimum frame spacing has elapsed.

USB

The PM support USB Version 1.10, operating at full speed (12 Mb/s). Specifically, the PM enumerates itself as a Human Interface Device (HID) with a control endpoint and two interrupt endpoints (IN/OUT).

Table 13 - PM USB Definitions

Parameter	Description
Bus Specification	USB 1.10
Bus Speed	Full-speed (12 Mbits/sec)
Control Endpoint Max Pkt Size	8 bytes
Device Description	Bus powered (98 mA max), 1 interface configuration (0)
Interface Description	Human Interface Device (HID)
Manufacturer string	"Concept2"
Product string	"Concept2 Performance Monitor 3 (PM3)" or "Concept2 Performance Monitor 4 (PM4)" "Concept2 Performance Monitor 5 (PM5)"
Endpoints	IN: Interrupt/EP3/polling rate: 8 msec. OUT: Interrupt/EP4/polling rate: 4 msec.
HID Reports	ID #1 – 20 bytes + 1 byte report ID ID #2 – 120 bytes + 1 byte report ID ID #4 – 62 bytes + 1 byte report ID

The report ID is always the first byte in the USB packet followed by the CSAFE frame.

Table 14 – C2 PM BTS Peripheral : Attribute Table

C2 PM BTS Peripheral : Attribute Table				
C2 PM Base UUID : CE06**XXXX**-43E5-11E4-916C-0800200C9A66				
UUID	**Type**	**Value**	**GATT Server Permissions**	**Notes**
0x1800	GAP primary service	GAP_SERVICE_UUID	READ	Start of GAP Service (Mandatory)
0x2A00	GAP device name characteristic	"PM5 430000000" where 430000000 is the actual PM5 serial number.	READ	Device name characteristic value
0x2A01	GAP appearance characteristic	0x0000	READ	Appearance characteristic value
0x2A02	GAP peripheral privacy characteristic	0x00 (GAP_PRIVACY_DISABLED)	READ/WRITE	Peripheral privacy characteristic value
0x2A03	GAP reconnect address characteristic	00:00:00:00:00:00	READ/WRITE	Reconnection address characteristic value
0x2A04	Peripheral preferred connection parameters characteristic	0x0018 (30ms preferred min connection interval) 0x0018 (30ms preferred max connection interval) 0x0000 (0 preferred slave latency) 0x03E8 (10000ms preferred supervision timeout)	READ	Peripheral preferred connection parameters characteristic value
0x1801	GATT primary service	GATT_SERVICE_UUID	READ	Start of GATT Service (Mandatory)
0x2A05	Service changed characteristic	(null)	(none)	Service changed characteristic value

C2 PM BTS Peripheral : Attribute Table				
C2 PM Base UUID : CE06**XXXX**-43E5-11E4-916C-0800200C9A66				
UUID	**Type**	**Value**	**GATT Server Permissions**	**Notes**
0x2902	GATT client configuration characteristic	00:00 (2 bytes)	READ/WRITE	Write 01:00 to enable notifications, 00:00 to disable
0x0010	C2 device information primary service	C2_DEVINFO_SERVICE_UUID	READ	Start of C2 Device Information Service
0x0011	C2 model number string characteristic	(Model Number, "PM5") (16 bytes)	READ	Model number string (Valid for PM5 V150 – V199.99 only) (Valid for PM5 V204 – V299.99 only)
0x0012	C2 serial number string characteristic	(Serial Number) (9 bytes)	READ	Serial number string
0x0013	C2 hardware revision string characteristic	(Hardware Revision) (3 bytes)	READ	Hardware revision string
0x0014	C2 firmware revision string characteristic	(Firmware Revision) (20 bytes)	READ	Firmware revision string
0x0015	C2 manufacturer name string characteristic	"Concept2" (16 bytes)	READ	Manufacturer name string
0x0016	Erg Machine Type characteristic	(Connected Erg Machine Type) (1 byte)	READ	Erg Machine Type enumerated value.[1] (Valid for PM5 V150 – V199.99 only) (Valid for PM5 V204 – V299.99 only)

[1] See Appendix for enumerated values

C2 PM BTS Peripheral : Attribute Table				
C2 PM Base UUID : CE06**XXXX**-43E5-11E4-916C-0800200C9A66				
UUID	**Type**	**Value**	**GATT Server Permissions**	**Notes**
0x0017	ATT MTU characteristic	(ATT Rx MTU) (2 bytes)	READ	23 – 512 bytes (Valid for PM5 V168.050 – V199.99 only) (Valid for PM5 V204.006 – V299.99 only)
0x0018	LL DLE characterstic	(LL Max Tx/Rx Bytes) (2 bytes)	READ	27 – 251 bytes (Valid for PM5 V168.050 – V199.99 only) (Valid for PM5 V204.006 – V299.99 only)
0x0020	C2 PM control primary service	C2_PM_CONTROL_SERVICE_UUID	READ	Start of C2 PM Control Primary Service
0x0021	C2 PM receive characteristic	(Up to 20 bytes)	WRITE	Control command in the form of a CSAFE frame sent to PM.[2]
0x0022	C2 PM transmit characteristic	(Up to 20 bytes)	READ	Response to command in the form of a CSAFE frame from the PM.
0x0030	C2 rowing primary service	C2_PM_CONTROL_SERVICE_UUID	READ	Start of C2 Rowing Service

[2] See Appendix for additional information on CSAFE commands

C2 PM BTS Peripheral : Attribute Table

C2 PM Base UUID : CE06**XXXX**-43E5-11E4-916C-0800200C9A66

UUID	Type	Value	GATT Server Permissions	Notes
0x0031	C2 rowing general status characteristic	(19 bytes) **Revision 0.13**	READ	*Data bytes packed as follows:* Elapsed Time Lo (0.01 sec lsb), Elapsed Time Mid, Elapsed Time High, Distance Lo (0.1 m lsb), Distance Mid, Distance High, Workout Type [3](enum), CSAFE_PM_GET_WORKOUTTYPE[4] typedef enum { *WORKOUTTYPE_JUSTROW_NOSPLITS*, /**< JustRow, no splits (0). */ WORKOUTTYPE_JUSTROW_SPLITS, /**< JustRow, splits (1). */ WORKOUTTYPE_FIXEDDIST_NOSPLITS, /**< Fixed distance, no splits (2). */ WORKOUTTYPE_FIXEDDIST_SPLITS, /**< Fixed distance, splits (3). */ WORKOUTTYPE_FIXEDTIME_NOSPLITS, /**< Fixed time, no splits (4). */ WORKOUTTYPE_FIXEDTIME_SPLITS, /**< Fixed time, splits (5). */ WORKOUTTYPE_FIXEDTIME_INTERVAL, /**< Fixed time interval (6). */ WORKOUTTYPE_FIXEDDIST_INTERVAL, /**< Fixed distance interval (7). */ WORKOUTTYPE_VARIABLE_INTERVAL, /**< Variable interval (8). */ WORKOUTTYPE_VARIABLE_UNDEFINEDREST_INTERVAL, /**< Variable interval, undefined rest (9). WORKOUTTYPE_FIXEDCALORIE_SPLITS, /**< Fixed calorie, splits (10). */ WORKOUTTYPE_FIXEDWATTMINUTE_SPLITS, /**< Fixed watt-minute, splits (11). */ WORKOUTTYPE_FIXEDCALS_INTERVAL, /**< Fixed calorie interval (12). */ WORKOUTTYPE_NUM /**< Number of workout types (13). */ } OBJ_WORKOUTTYPE_T; Interval Type[5] (enum), CSAFE_PM_GET_INTERVALTYPE Workout State (enum), CSAFE_PM_GET_WORKOUTSTATE Rowing State (enum), CSAFE_PM_GET_ROWINGSTATE Stroke State (enum), CSAFE_PM_GET_STROKESTATE Total Work Distance Lo, CSAFE_PM_GET_WORKDISTANCE

C2 PM BTS Peripheral : Attribute Table

C2 PM Base UUID : CE06**XXXX**-43E5-11E4-916C-0800200C9A66

UUID	Type	Value	GATT Server Permissions	Notes
0x0032	C2 rowing additional status 1 characteristic	(17 bytes)	READ	*Data bytes packed as follows:* Elapsed Time Lo (0.01 sec lsb), Elapsed Time Mid, Elapsed Time High, Speed Lo (0.001m/s lsb), CSAFE_GETSPEED_CMD[6] Speed Hi, Stroke Rate (strokes/min), CSAFE_PM_GET_STROKERATE Heartrate (bpm, 255=invalid), CSAFE_PM_GET_AVG_HEARTRATE Current Pace Lo (0.01 sec lsb), CSAFE_PM_GET_STROKE_500MPACE Current Pace Hi, Average Pace Lo (0.01 sec lsb), CSAFE_PM_GET_TOTAL_AVG_500MPACE Average Pace Hi, Rest Distance Lo, CSAFE_PM_GET_RESTDISTANCE Rest Distance Hi, Rest Time Lo, (0.01 sec lsb) CSAFE_PM_GET_RESTTIME Rest Time Mid, Rest Time Hi Erg Machine Type [7]

[3] See Appendix for enumerated values definitions
[4] For reference - The named CSAFE command returns the same value
[5] This value will change depending on where you are in the interval (work, rest, etc). Use workout type to determine whether the intervals are time or distance intervals.
[6] For reference - The named CSAFE command returns the same value
[7] See Appendix for enumerated values definitions. For MultiErg workouts, this will be the Machine Type of the current interval, which may not be the same as the connected Machine.

C2 PM BTS Peripheral : Attribute Table				
C2 PM Base UUID : CE06**XXXX**-43E5-11E4-916C-0800200C9A66				
UUID	**Type**	**Value**	**GATT Server Permissions**	**Notes**
0x0033	C2 rowing additional status 2 characteristic	(20 bytes)	READ	*Data bytes packed as follows:* Elapsed Time Lo (0.01 sec lsb), Elapsed Time Mid, Elapsed Time High, Interval Count, CSAFE_PM_GET_WORKOUTINTERVALCOUNT[8] Average Power Lo, CSAFE_PM_GET_TOTAL_AVG_POWER Average Power Hi, Total Calories Lo (cals), CSAFE_PM_GET_TOTAL_AVG_CALORIES Total Calories Hi, Split/Int Avg Pace Lo (0.01 sec lsb), CSAFE_PM_GET_SPLIT_AVG_500MPACE Split/Int Avg Pace Hi, Split/Int Avg Power Lo (watts), CSAFE_PM_GET_SPLIT_AVG_POWER Split/Int Avg Power Hi, Split/Int Avg Calories Lo (cals/hr), CSAFE_PM_GET_SPLIT_AVG_CALORIES Split/Interval Avg Calories Hi, Last Split Time Lo (0.1 sec lsb), CSAFE_PM_GET_LAST_SPLITTIME Last Split Time Mid, Last Split Time High, Last Split Distance Lo, CSAFE_PM_GET_LAST_SPLITDISTANCE (in meters) Last Split Distance Mid, Last Split Distance Hi

[8] For reference - The named CSAFE command returns the same value

C2 PM BTS Peripheral : Attribute Table				
C2 PM Base UUID : CE06**XXXX**-43E5-11E4-916C-0800200C9A66				
UUID	**Type**	**Value**	**GATT Server Permissions**	**Notes**
0x0034	C2 rowing general status and additional status sample rate characteristic	(1 byte)	WRITE/Read	Determines how often slave sends general status and additional status data as notifications. Set rate as follows: 0 – 1 sec 1 - 500ms (default if characteristic is not explicitly set by the app) 2 – 250ms 3 – 100ms

C2 PM BTS Peripheral : Attribute Table				
C2 PM Base UUID : CE06**XXXX**-43E5-11E4-916C-0800200C9A66				
UUID	**Type**	**Value**	**GATT Server Permissions**	**Notes**
0x0035	C2 rowing stroke data characteristic	(20 bytes)	READ	*Data bytes packed as follows:* Elapsed Time Lo (0.01 sec lsb), Elapsed Time Mid, Elapsed Time High, Distance Lo (0.1 m lsb), Distance Mid, Distance High, Drive Length (0.01 meters, max = 2.55m), CSAFE_PM_GET_STROKESTATS Drive Time (0.01 sec, max = 2.55 sec), Stroke Recovery Time Lo (0.01 sec, max = 655.35 sec), CSAFE_PM_GET_STROKESTATS Stroke Recovery Time Hi, CSAFE_PM_GET_STROKESTATS[9] Stroke Distance Lo (0.01 m, max=655.35m), CSAFE_PM_GET_STROKESTATS Stroke Distance Hi, Peak Drive Force Lo (0.1 lbs of force, max=6553.5m), CSAFE_PM_GET_STROKESTATS Peak Drive Force Hi, Average Drive Force Lo (0.1 lbs of force, max=6553.5m), CSAFE_PM_GET_STROKESTATS Average Drive Force Hi, Work Per Stroke Lo (0.1 Joules, max=6553.5 Joules), CSAFE_PM_GET_STROKESTATS Work Per Stroke Hi Stroke Count Lo, CSAFE_PM_GET_STROKESTATS Stroke Count Hi,

[9] For reference - The named CSAFE command returns the same value

C2 PM BTS Peripheral : Attribute Table

C2 PM Base UUID : CE06**XXXX**-43E5-11E4-916C-0800200C9A66

UUID	Type	Value	GATT Server Permissions	Notes
0x0036	C2 rowing additional stroke data characteristic	(15 bytes)	READ	*Data bytes packed as follows:* Elapsed Time Lo (0.01 sec lsb), Elapsed Time Mid, Elapsed Time High, Stroke Power Lo (watts), CSAFE_PM_GET_STROKE_POWER Stroke Power Hi, Stroke Calories Lo (cal/hr), CSAFE_PM_GET_STROKE_CALORICBURNRATE Stroke Calories Hi, Stroke Count Lo, CSAFE_PM_GET_STROKESTATS Stroke Count Hi, Projected Work Time Lo (secs), Projected Work Time Mid, Projected Work Time Hi, Projected Work Distance Lo (meters), Projected Work Distance Mid, Projected Work Distance Hi

C2 PM BTS Peripheral : Attribute Table				
C2 PM Base UUID : CE06**XXXX**-43E5-11E4-916C-0800200C9A66				
UUID	**Type**	**Value**	**GATT Server Permissions**	**Notes**
0x0037	C2 rowing split/interval data characteristic	(18 bytes)	READ	*Data bytes packed as follows:* Elapsed Time Lo (0.01 sec lsb), Elapsed Time Mid, Elapsed Time High, Distance Lo (0.1 m lsb), Distance Mid, Distance High, Split/Interval Time Lo (0.1 sec lsb), Split/Interval Time Mid, Split/Interval Time High, Split/Interval Distance Lo (1m lsb), Split/Interval Distance Mid, Split/Interval Distance High, Interval Rest Time Lo (1 sec lsb), Interval Rest Time Hi, Interval Rest Distance Lo (1m lsb), Interval Rest Distance Hi Split/Interval Type[10], Split/Interval Number,

[10] This value will change depending on where you are in the interval (work, rest, etc). Use workout type to determine whether the intervals are time or distance intervals

C2 PM BTS Peripheral : Attribute Table				
C2 PM Base UUID : CE06**XXXX**-43E5-11E4-916C-0800200C9A66				
UUID	**Type**	**Value**	**GATT Server Permissions**	**Notes**
0x0038	C2 rowing additional split/interval data characteristic	(19 bytes)	READ	*Data bytes packed as follows:* Elapsed Time Lo (0.01 sec lsb), Elapsed Time Mid, Elapsed Time High, Split/Interval Avg Stroke Rate, Split/Interval Work Heartrate, Split/Interval Rest Heartrate, Split/Interval Avg Pace Lo (0.1 sec lsb) Split/Interval Avg Pace Hi, Split/Interval Total Calories Lo (Cals), Split/Interval Total Calories Hi, Split/Interval Avg Calories Lo (Cals/Hr), Split/Interval Avg Calories Hi, Split/Interval Speed Lo (0.001 m/s, max=65.534 m/s) Split/Interval Speed Hi, Split/Interval Power Lo (Watts, max = 65.534 kW) Split/Interval Power Hi Split Avg Drag Factor, Split/Interval Number, Erg Machine Type[11]

[11] See Appendix for enumerated values definitions. For MultiErg workouts, this will be the Machine Type of the current interval, which may not be the same as the connected Machine.

C2 PM BTS Peripheral : Attribute Table

C2 PM Base UUID : CE06**XXXX**-43E5-11E4-916C-0800200C9A66

UUID	Type	Value	GATT Server Permissions	Notes
0x0039	C2 rowing end of workout summary data characteristic	(20 bytes)	READ	*Data bytes packed as follows:* Log Entry Date Lo, Log Entry Date Hi, Log Entry Time Lo, Log Entry Time Hi, Elapsed Time Lo (0.01 sec lsb), Elapsed Time Mid, Elapsed Time High, Distance Lo (0.1 m lsb), Distance Mid, Distance High, Average Stroke Rate, Ending Heartrate, Average Heartrate, Min Heartrate, Max Heartrate, Drag Factor Average, Recovery Heart Rate, (zero = not valid data. After 1 minute of rest/recovery, PM5 sends this data as a revised End Of Workout summary data characteristic unless the monitor has been turned off or a new workout started) Workout Type, Avg Pace Lo (0.1 sec lsb) Avg Pace Hi

C2 PM BTS Peripheral : Attribute Table				
C2 PM Base UUID : CE06**XXXX**-43E5-11E4-916C-0800200C9A66				
UUID	**Type**	**Value**	**GATT Server Permissions**	**Notes**
0x003A	C2 rowing end of workout additional summary data characteristic	(19 bytes)	READ	*Data bytes packed as follows:* Log Entry Date Lo, Log Entry Date Hi, Log Entry Time Lo, Log Entry Time Hi, Split/Interval Type[12], Split/Interval Size Lo, (meters or seconds) Split/Interval Size Hi, Split/Interval Count, Total Calories Lo, Total Calories Hi, Watts Lo, Watts Hi, Total Rest Distance Lo (1 m lsb), Total Rest Distance Mid, Total Rest Distance High Interval Rest Time Lo (seconds), Interval Rest Time Hi, Avg Calories Lo, (cals/hr) Avg Calories Hi,
0x003B	C2 rowing heart rate belt information characteristic	(6 bytes)	WRITE/Read	Manufacturer ID, Device Type, Belt ID Lo, Belt ID Mid Lo, Betl ID Mid Hi, Belt ID Hi

[12] This value will change depending on where you are in the interval when the workout is terminated. Use workout type to determine whether the intervals are time or distance intervals.

C2 PM BTS Peripheral : Attribute Table				
C2 PM Base UUID : CE06**XXXX**-43E5-11E4-916C-0800200C9A66				
UUID	**Type**	**Value**	**GATT Server Permissions**	**Notes**
0x003D	C2 force curve data characteristic	(2 - 288 bytes separated into multiple successive notifications)	WRITE/Read	MS Nib = # characteristics, LS Nib = # words,[13] Sequence number, Data[n] (LS), Data[n+1] (MS), Data[n+2] (LS), Data[n+3] (MS), Data[n+4] (LS), Data[n+5] (MS), Data[n+6] (LS), Data[n+7] (MS), Data[n+8] (LS), Data[n+9] (MS), Data[n+10] (LS), Data[n+11] (MS), Data[n+12] (LS), Data[n+13] (MS), Data[n+14] (LS), Data[n+15] (MS), Data[n+16] (LS), Data[n+17] (MS)

[13] MS Nibble = Total number of characteristics for this force curve, LS Nibble = Number of 16-bit data points in the current characteristic

C2 PM BTS Peripheral : Attribute Table

C2 PM Base UUID : CE06**XXXX**-43E5-11E4-916C-0800200C9A66

UUID	Type	Value	GATT Server Permissions	Notes
0x0080	C2 multiplexed information characteristic	(Up to 20 bytes)	READ	The multiplexed information characteristic consists of an identification byte and up to 19 data bytes. The first byte identifies the payload as defined in the Data Definitions table in the following section. **Important note: The following identifiers will ONLY be multiplexed on this characteristic as long as the respective characteristic notification of the same ID is NOT enabled. Note: The byte length of the following multiplexed characteristics does not include the identifier byte. The total length of the data packet is N+1 bytes. 0x31 0x32 0x33 0x35 0x36 0x37 0x38 0x39 0x3A 0x3B 0x3C

Table 15 – C2 Multiplexed Information: Data Definitions

C2 Multiplexed Information: Data Definitions			
ID	**Name**	**Byte Length**	**Definitions**

C2 Multiplexed Information: Data Definitions

ID	Name	Byte Length	Definitions
0x0031	C2 rowing general status	(19 bytes)	*Data bytes packed as follows:* Elapsed Time Lo (0.01 sec lsb), Elapsed Time Mid, Elapsed Time High, Distance Lo (0.1 m lsb), Distance Mid, Distance High, Workout Type [14](enum), CSAFE_PM_GET_WORKOUTTYPE[15] typedef enum { *WORKOUTTYPE_JUSTROW_NOSPLITS*, /**< JustRow, no splits (0). */ WORKOUTTYPE_JUSTROW_SPLITS, /**< JustRow, splits (1). */ WORKOUTTYPE_FIXEDDIST_NOSPLITS, /**< Fixed distance, no splits (2). */ WORKOUTTYPE_FIXEDDIST_SPLITS, /**< Fixed distance, splits (3). */ WORKOUTTYPE_FIXEDTIME_NOSPLITS, /**< Fixed time, no splits (4). */ WORKOUTTYPE_FIXEDTIME_SPLITS, /**< Fixed time, splits (5). */ WORKOUTTYPE_FIXEDTIME_INTERVAL, /**< Fixed time interval (6). */ WORKOUTTYPE_FIXEDDIST_INTERVAL, /**< Fixed distance interval (7). */ WORKOUTTYPE_VARIABLE_INTERVAL, /**< Variable interval (8). */ WORKOUTTYPE_VARIABLE_UNDEFINEDREST_INTERVAL, /**< Variable interval, undefined rest (9). WORKOUTTYPE_FIXEDCALORIE_SPLITS, /**< Fixed calorie, splits (10). */ WORKOUTTYPE_FIXEDWATTMINUTE_SPLITS, /**< Fixed watt-minute, splits (11). */ WORKOUTTYPE_FIXEDCALS_INTERVAL, /**< Fixed calorie interval (12). */ WORKOUTTYPE_NUM /**< Number of workout types (13). */ } OBJ_WORKOUTTYPE_T; Interval Type[16] (enum), CSAFE_PM_GET_INTERVALTYPE Workout State (enum), CSAFE_PM_GET_WORKOUTSTATE Rowing State (enum), CSAFE_PM_GET_ROWINGSTATE Stroke State (enum), CSAFE_PM_GET_STROKESTATE Total Work Distance Lo, CSAFE_PM_GET_WORKDISTANCE Total Work Distance Mid, Total Work Distance Hi, Workout Duration Lo (if time, 0.01 sec lsb)

C2 Multiplexed Information: Data Definitions			
ID	**Name**	**Byte Length**	**Definitions**
0x0032	C2 rowing additional status 1	(19 bytes)	*Data bytes packed as follows:* Elapsed Time Lo (0.01 sec lsb), Elapsed Time Mid, Elapsed Time High, Speed Lo (0.001m/s lsb), CSAFE_GETSPEED_CMD[17] Speed Hi, Stroke Rate (strokes/min), CSAFE_PM_GET_STROKERATE Heartrate (bpm, 255=invalid), CSAFE_PM_GET_AVG_HEARTRATE Current Pace Lo (0.01 sec lsb), CSAFE_PM_GET_STROKE_500MPACE Current Pace Hi, Average Pace Lo (0.01 sec lsb), CSAFE_PM_GET_TOTAL_AVG_500MPACE Average Pace Hi, Rest Distance Lo, CSAFE_PM_GET_RESTDISTANCE Rest Distance Hi, Rest Time Lo, (0.01 sec lsb) CSAFE_PM_GET_RESTTIME Rest Time Mid, Rest Time Hi, Average Power Lo, CSAFE_PM_GET_TOTAL_AVG_POWER Average Power Hi Erg Machine Type

[14] See Appendix for enumerated values definitions
[15] For reference - The named CSAFE command returns the same value
[16] This value will change depending on where you are in the interval (work, rest, etc). Use workout type to determine whether the intervals are time or distance intervals.
[17] For reference - The named CSAFE command returns the same value

C2 Multiplexed Information: Data Definitions			
ID	**Name**	**Byte Length**	**Definitions**
0x0033	C2 rowing additional status 2	(18 bytes)	*Data bytes packed as follows:* Elapsed Time Lo (0.01 sec lsb), Elapsed Time Mid, Elapsed Time High, Interval Count, CSAFE_PM_GET_WORKOUTINTERVALCOUNT[18] Total Calories Lo (cals), CSAFE_PM_GET_TOTAL_AVG_CALORIES Total Calories Hi, Split/Int Avg Pace Lo (0.01 sec lsb), CSAFE_PM_GET_SPLIT_AVG_500MPACE Split/Int Avg Pace Hi, Split/Int Avg Power Lo (watts), CSAFE_PM_GET_SPLIT_AVG_POWER Split/Int Avg Power Hi, Split/Int Avg Calories Lo (cals/hr), CSAFE_PM_GET_SPLIT_AVG_CALORIES Split/Interval Avg Calories Hi, Last Split Time Lo (0.1 sec lsb), CSAFE_PM_GET_LAST_SPLITTIME Last Split Time Mid, Last Split Time High, Last Split Distance Lo, CSAFE_PM_GET_LAST_SPLITDISTANCE (in meters) Last Split Distance Mid, Last Split Distance Hi
0x0034	Not used		

[18] For reference - The named CSAFE command returns the same value

C2 Multiplexed Information: Data Definitions			
ID	**Name**	**Byte Length**	**Definitions**
0x0035	C2 rowing stroke data	(18 bytes)	*Data bytes packed as follows:* Elapsed Time Lo (0.01 sec lsb), Elapsed Time Mid, Elapsed Time High, Distance Lo (0.1 m lsb), Distance Mid, Distance High, Drive Length (0.01 meters, max = 2.55m), CSAFE_PM_GET_STROKESTATS Drive Time (0.01 sec, max = 2.55 sec), Stroke Recovery Time Lo (0.01 sec, max = 655.35 sec), CSAFE_PM_GET_STROKESTATS Stroke Recovery Time Hi, CSAFE_PM_GET_STROKESTATS[19] Stroke Distance Lo (0.01 m, max=655.35m), CSAFE_PM_GET_STROKESTATS Stroke Distance Hi, Peak Drive Force Lo (0.1 lbs of force, max=6553.5m), CSAFE_PM_GET_STROKESTATS Peak Drive Force Hi, Average Drive Force Lo (0.1 lbs of force, max=6553.5m), CSAFE_PM_GET_STROKESTATS Average Drive Force Hi, Stroke Count Lo, CSAFE_PM_GET_STROKESTATS Stroke Count Hi,

[19] For reference - The named CSAFE command returns the same value

C2 Multiplexed Information: Data Definitions

ID	Name	Byte Length	Definitions
0x0036	C2 rowing additional stroke data	(17 bytes)	*Data bytes packed as follows:* Elapsed Time Lo (0.01 sec lsb), Elapsed Time Mid, Elapsed Time High, Stroke Power Lo (watts), CSAFE_PM_GET_STROKE_POWER Stroke Power Hi, Stroke Calories Lo (cal/hr), CSAFE_PM_GET_STROKE_CALORICBURNRATE Stroke Calories Hi, Stroke Count Lo, CSAFE_PM_GET_STROKESTATS Stroke Count Hi, Projected Work Time Lo (secs), Projected Work Time Mid, Projected Work Time Hi, Projected Work Distance Lo (meters), Projected Work Distance Mid, Projected Work Distance Hi, Work Per Stroke Lo (0.1 Joules, max=6553.5 Joules), CSAFE_PM_GET_STROKESTATS Work Per Stroke Hi

C2 Multiplexed Information: Data Definitions			
ID	**Name**	**Byte Length**	**Definitions**
0x0037	C2 rowing split/interval data	(18 bytes)	*Data bytes packed as follows:* Elapsed Time Lo (0.01 sec lsb), Elapsed Time Mid, Elapsed Time High, Distance Lo (0.1 m lsb), Distance Mid, Distance High, Split/Interval Time Lo (0.1 sec lsb), Split/Interval Time Mid, Split/Interval Time High, Split/Interval Distance Lo (1m lsb), Split/Interval Distance Mid, Split/Interval Distance High, Interval Rest Time Lo (1 sec lsb), Interval Rest Time Hi, Interval Rest Distance Lo (1m lsb), Interval Rest Distance Hi Split/Interval Type[20], Split/Interval Number,

[20] This value will change depending on where you are in the interval (work, rest, etc). Use workout type to determine whether the intervals are time or distance intervals

C2 Multiplexed Information: Data Definitions			
ID	**Name**	**Byte Length**	**Definitions**
0x0038	C2 rowing additional split/interval data	(18 bytes)	*Data bytes packed as follows:* Elapsed Time Lo (0.01 sec lsb), Elapsed Time Mid, Elapsed Time High, Split/Interval Avg Stroke Rate, Split/Interval Work Heartrate, Split/Interval Rest Heartrate, Split/Interval Avg Pace Lo (0.1 sec lsb) Split/Interval Avg Pace Hi, Split/Interval Total Calories Lo (Cals), Split/Interval Total Calories Hi, Split/Interval Avg Calories Lo (Cals/Hr), Split/Interval Avg Calories Hi, Split/Interval Speed Lo (0.001 m/s, max=65.534 m/s) Split/Interval Speed Hi, Split/Interval Power Lo (Watts, max = 65.534 kW) Split/Interval Power Hi Split Avg Drag Factor, Split/Interval Number Erg Machine Type[21]

[21] See Appendix for enumerated values definitions. For MultiErg workouts, this will be the machine type of the current interval, which will not be the same as the connected Machine

C2 Multiplexed Information: Data Definitions

ID	Name	Byte Length	Definitions
0x0039	C2 rowing end of workout summary data characteristic	(18 bytes)	*Data bytes packed as follows:* Log Entry Date Lo, Log Entry Date Hi, Log Entry Time Lo, Log Entry Time Hi, Elapsed Time Lo (0.01 sec lsb), Elapsed Time Mid, Elapsed Time High, Distance Lo (0.1 m lsb), Distance Mid, Distance High, Average Stroke Rate, Ending Heartrate, Average Heartrate, Min Heartrate, Max Heartrate, Drag Factor Average, Recovery Heart Rate, (zero = not valid data. After 1 minute of rest/recovery, PM5 sends this data as a revised End Of Workout summary data characteristic unless the monitor has been turned off or a new workout started) Workout Type

C2 Multiplexed Information: Data Definitions

ID	Name	Byte Length	Definitions
0x003A	C2 rowing end of workout additional summary data characteristic 1	(18 bytes)	*Data bytes packed as follows:* Log Entry Date Lo, Log Entry Date Hi, Log Entry Time Lo, Log Entry Time Hi, Split/Interval Size Lo, (meters or seconds) Split/Interval Size Hi, Split/Interval Count, Total Calories Lo, Total Calories Hi, Watts Lo, Watts Hi, Total Rest Distance Lo (1 m lsb), Total Rest Distance Mid, Total Rest Distance High Interval Rest Time Lo (seconds), Interval Rest Time Hi, Avg Calories Lo, (cals/hr) Avg Calories Hi,
0x003B	C2 rowing heart rate belt information characteristic	(6 bytes)	Manufacturer ID, Device Type, Belt ID Lo, Belt ID Mid Lo, Betl ID Mid Hi, Belt ID Hi

C2 Multiplexed Information: Data Definitions			
ID	**Name**	**Byte Length**	**Definitions**
0x003C	C2 rowing end of workout additional summary data characteristic 2	(10 bytes)	*Data bytes packed as follows:* Log Entry Date Lo, Log Entry Date Hi, Log Entry Time Lo, Log Entry Time Hi, Avg Pace Lo (0.1 sec lsb) Avg Pace Hi, Game Identifier/ Workout Verified (see Appendix), Game Score Lo, Game Score Hi Erg Machine Type [22]

[22] See Appendix for enumerated values definitions. For MultiErg workouts, this will be the one of the MultiErg Machine Types, which may not be the same as the connected Machine.

Public CSAFE

Features

Public CSAFE Default Configuration

Individual manufacturers specify certain protocol parameters (e.g., timeouts, auto response behavior, etc.). Table 16 summarizes the protocol defaults for the PM. Note that certain parameters listed in Table 16 cannot be changed (refer to the section on Public CSAFE Unsupported Features for additional information).

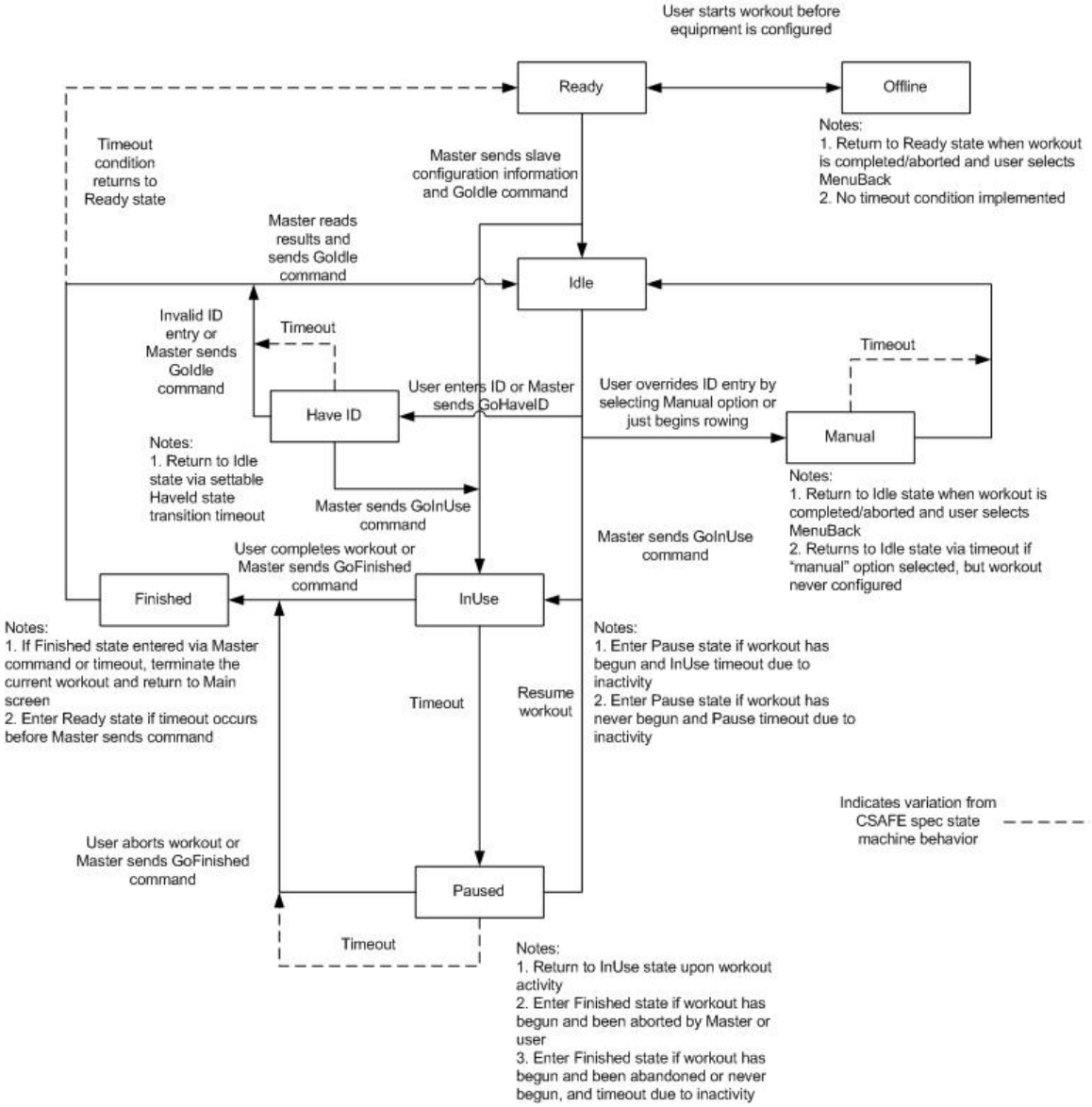
Table 16 – PM Public CSAFE Protocol Defaults

Parameter	Default Value	Comments
HaveID State Transition Timeout	10 seconds	This timeout (settable via the cmdSetTimeout command) defines the delay between entering the HaveID state and transitioning back to the Idle state
Inactivity During InUse State Timeout	6 seconds	This timeout defines the duration of inactivity during the InUse state (once the workout has begun) before entering the Paused state
Inactivity During Pause State Timeout	220 seconds	This timeout defines the duration of inactivity during the Paused state (once the workout has begun) before entering the Finished state
Unconfigured Workout During Manual State Timeout	220 seconds	This timeout is the same as the PAUSE state timeout and occurs if a user enters MANUAL mode and doesn't configure a workout
Inactivity During Finished State Timeout	220 seconds	This timeout is the same as the PAUSE state timeout and occurs if the workout has begun and been abandoned or the workout has never begun
Units Type	Metric	Metric units only
User ID Digits	5	Five-digit user ID (settable via the cmdIDDigits from 2 – 5 digits)
User ID	0 0 0 0 0	Default value
AutoUpload Byte	0x10	flgAutoStatus: Disabled (**cannot be changed**) flgUpStatus: Disabled (**cannot be changed**) flgUpList: Disabled (**cannot be changed**) flgAck : Enabled (**cannot be changed**) flgExternControl: Disabled (**cannot be changed**)
Serial Number Digits	9	Number of digits in serial number response
PM-specific Commands	All states	These commands are accessible in all slave states

Public CSAFE State Machine Operation

The state machine implementation is shown in
Figure 7 including variations to the behavior specific to the PM.

Figure 7 – Public CSAFE State Machine Diagram



Public CSAFE Unsupported Features

Individual manufacturers also determine which protocol features will not be supported by their equipment. Table 17 summarizes the unsupported protocol features and the deviations from other features.

Table 17 - PM Unsupported Public CSAFE Protocol Features

Feature	Comments
AutoStatus Enable	No unsolicited status uploads
UpList Enable	No unsolicited command list uploads
Ack Disable	All commands will be responded to by a least a status byte
Text Messaging	No text messaging functions
Set User Information	Not setting user weight, age and gender
Get User Information	User weight is fixed at 175 lbs, age and gender not supported

Finished State Timeout	No Finished state timeout is employed to cause a transition back to the Idle state; when a user hits the MENU/BACK to conclude viewing a finished workout result or terminate a workout in progress, the Ready state is entered instead of the Idle state. Instead a Finished state timeout is employed to return to the Ready state.
Paused State Timeout	A timeout is employed to enter the Finished state in the event a configured workout is never started or re-started.
Manual State Timeout	A timeout is employed to return to the Idle state in the event a manual user ID override is performed and a workout is never configured
InUse State Entry	In addition to allowing entry into the InUse state from the Idle and HaveID states, entry from the Ready state is also allowed
Set Calories Goal	Since the PM3/PM4 allows the user to select display units (either time/meters, watts or calories), setting the workout goal using power is sufficient to define a target pace for the pace boat display for all display units.

Programmed Workout Parameter Limits

There are several parameters which have minimum and maximum values when configuring a workout. These parameter limits are imposed by the user interface when configuring the workout during typical usage, but will be imposed somewhat differently when configuring workouts via the public CSAFE interface. When the SetProgramCmd is issued by the Master to program the previously configured workout, all pertinent workout parameters are checked against their respective limits. If any parameter violates its limits, the entire workout configuration operation is aborted resulting in a "PrevReject" frame status. The Master must issue a PM-specific GetErrorType command to determine the specific error information. Table 18 lists the workout configuration parameter limits which should be adhered to during programming.

Table 18 – PM3/PM4 Workout Configuration Parameter Limits

Command Name	Description	Minimum	Maximum
CSAFE_SETTWORK_CMD	Workout time goal	:20	9:59:59
CSAFE_SETHORIZONTAL_CMD	Horizontal distance goal	100m	50,000m
CSAFE_PM_SET_SPLITDURATION	Time/distance split duration	:20/100m[1]	N/A[2]

Notes:

1. The minimum split duration must not cause the total number of splits per workout to exceed the maximum of 30.
2. The maximum split duration cannot exceed the workout time goal or the horizontal distance goal.

Table 19 – PM5 Workout Configuration Parameter Limits

Command Name	Description	Minimum	Maximum
CSAFE_SETTWORK_CMD	Workout time goal	:20	9:59:59
CSAFE_SETHORIZONTAL_CMD	Horizontal distance goal	100m	50,000m[1]
CSAFE_PM_SET_SPLITDURATION	Time/distance split duration	:20/100m[2]	N/A[3]

Notes:

1. The maximum hoziontal distance for a BikeErg is 100,000m.
2. The minimum split duration must not cause the total number of splits per workout to exceed the maximum of 50.
3. The maximum split duration cannot exceed the workout time goal or the horizontal distance goal.

Command List

Public Short Commands

Command Name	Command Identifier	Response Data
CSAFE_GETSTATUS_CMD	0x80	Byte 0: Status
CSAFE_RESET_CMD	0x81	N/A[1]
CSAFE_GOIDLE_CMD	0x82	N/A[1]
CSAFE_GOHAVEID_CMD	0x83	N/A[1]
CSAFE_GOINUSE_CMD	0x85	N/A[1]
CSAFE_GOFINISHED_CMD	0x86	N/A[1]
CSAFE_GOREADY_CMD	0x87	N/A[1]
CSAFE_BADID_CMD	0x88	N/A[1]
CSAFE_GETVERSION_CMD	0x91	Byte 0: Mfg ID Byte 1: CID Byte 2: Model Byte 3: HW Version (LS) Byte 4: HW Version (MS) Byte 5: SW Version (LS) Byte 6: SW Version (MS)
CSAFE_GETID_CMD	0x92	Byte 0: ASCII Digit 0 (MS) Byte 1: ASCII Digit 1 Byte 2: ASCII Digit 2[2] Byte 3: ASCII Digit 3[2] Byte 4: ASCII Digit 4[2] (LS)
CSAFE_GETUNITS_CMD	0x93	Byte 0: Units Type
CSAFE_GETSERIAL_CMD	0x94	Byte 0: ASCII Serial # (MS) Byte 1: ASCII Serial # Byte 2: ASCII Serial # Byte 3: ASCII Serial # Byte 4: ASCII Serial # Byte 5: ASCII Serial # Byte 6: ASCII Serial # Byte 7: ASCII Serial # Byte 8: ASCII Serial # (LS)
CSAFE_GETLIST_CMD	0x98	<Not implemented>
CSAFE_GETUTILIZATION_CMD	0x99	<Not implemented>
CSAFE_GETMOTORCURRENT_CMD	0x9A	<Not implemented>
CSAFE_GETODOMETER_CMD	0x9B	Byte 0: Distance (LSB) Byte 1: Distance Byte 2: Distance Byte 3: Distance (MSB) Byte 4: Units Specifier
CSAFE_GETERRORCODE_CMD	0x9C	Byte 0: Error Code (LSB) Byte 1: Error Code Byte 2: Error Code (MSB)
CSAFE_GETSERVICECODE_CMD	0x9D	<Not implemented>
CSAFE_GETUSERCFG1_CMD	0x9E	<Not implemented>
CSAFE_GETUSERCFG2_CMD	0x9F	<Not implemented>
CSAFE_GETTWORK_CMD	0xA0	Byte 0: Hours Byte 1: Minutes Byte 2: Seconds
CSAFE_GETHORIZONTAL_CMD	0xA1	Byte 0: Horizontal Distance (LSB)

		Byte 1: Horizontal Distance (MSB) Byte 2: Units Specifier
CSAFE_GETVERTICAL_CMD	0xA2	<Not implemented>
CSAFE_GETCALORIES_CMD	0xA3	Byte 0: Total Calories (LSB) Byte 1: Total Calories (MSB)
CSAFE_GETPROGRAM_CMD	0xA4	Byte 0: Programmed/Pre-stored Workout Number
CSAFE_GETSPEED_CMD	0xA5	<Not implemented>
CSAFE_GETPACE_CMD	0xA6	Byte 0: Stroke Pace (LSB) Byte 1: Stroke Pace (MSB) Byte 2: Units Specifier
CSAFE_GETCADENCE_CMD	0xA7	Byte 0: Stroke Rate (LSB) Byte 1: Stroke Rate (MSB) Byte 2: Units Specifier
CSAFE_GETGRADE_CMD	0xA8	<Not implemented>
CSAFE_GETGEAR_CMD	0xA9	<Not implemented>
CSAFE_GETUPLIST_CMD	0xAA	<Not implemented>
CSAFE_GETUSERINFO_CMD	0xAB	Byte 0: Weight (LSB) Byte 1: Weight (MSB) Byte 2: Units Specifier Byte 3: Age Byte 4: Gender
CSAFE_GETTORQUE_CMD	0xAC	<Not implemented>
CSAFE_GETHRCUR_CMD	0xB0	Byte 0: Beats/Min
CSAFE_GETHRTZONE_CMD	0xB2	<Not implemented>
CSAFE_GETMETS_CMD	0xB3	<Not implemented>
CSAFE_GETPOWER_CMD	0xB4	Byte 0: Stroke Watts (LSB) Byte 1: Stroke Watts (MSB) Byte 2: Units Specifier
CSAFE_GETHRAVG_CMD	0xB5	<Not implemented>
CSAFE_GETHRMAX_CMD	0xB6	<Not implemented>
CSAFE_GETUSERDATA1_CMD	0xBE	<Not implemented>
CSAFE_GETUSERDATA2_CMD	0xBF	<Not implemented>
CSAFE_GETAUDIOCHANNEL_CMD	0xC0	<Not implemented>
CSAFE_GETAUDIOVOLUME_CMD	0xC1	<Not implemented>
CSAFE_GETAUDIOMUTE_CMD	0xC2	<Not implemented>
CSAFE_ENDTEXT_CMD	0xE0	<Not implemented>
CSAFE_DISPLAYPOPUP_CMD	0xE1	<Not implemented>
CSAFE_GETPOPUPSTATUS_CMD	0xE5	<Not implemented>

Notes:

1. No specific response data, but the status byte will be returned
2. Depends on # ID digits configuration

Public Long Commands

Command Name	Command Identifier	Command Data	Response Data
CSAFE_AUTOUPLOAD_CMD[2]	0x01	Byte 0: Configuration	N/A
CSAFE_UPLIST_CMD	0x02	<Not implemented>	N/A
CSAFE_UPSTATUSSEC_CMD	0x04	<Not implemented>	N/A
CSAFE_UPLISTSEC_CMD	0x05	<Not implemented>	N/A
CSAFE_IDDIGITS_CMD	0x10	Byte 0: # of Digits	N/A
CSAFE_SETTIME_CMD	0x11	Byte 0: Hour	N/A

		Byte 1: Minute Byte 2: Second	
CSAFE_SETDATE_CMD	0x12	Byte 0: Year Byte 1: Month Byte 2: Day	N/A
CSAFE_SETTIMEOUT_CMD	0x13	Byte 0: State Timeout	N/A
CSAFE_SETUSERCFG1_CMD[1]	0x1A	One or more PM-specific commands	<PM-specific command identifer(s)>
CSAFE_SETUSERCFG2_CMD	0x1B	<Not implemented>	N/A
CSAFE_SETTWORK_CMD	0x20	Byte 0: Hours Byte 1: Minutes Byte 2: Seconds	N/A
CSAFE_SETHORIZONTAL_CMD	0x21	Byte 0: Horizontal Distance (LSB) Byte 1: Horizontal Distance (MSB) Byte 2: Units Specifier	N/A
CSAFE_SETVERTICAL_CMD	0x22	<Not implemented>	N/A
CSAFE_SETCALORIES_CMD	0x23	Byte 0: Total Calories (LSB) Byte 1: Total Calories (MSB)	N/A
CSAFE_SETPROGRAM_CMD	0x24	Byte 0: Programmed or Pre-stored Workout Byte 1: <don't care>	N/A
CSAFE_SETSPEED_CMD	0x25	<Not implemented>	N/A
CSAFE_SETGRADE_CMD	0x28	<Not implemented>	N/A
CSAFE_SETGEAR_CMD	0x29	<Not implemented>	N/A
CSAFE_SETUSERINFO_CMD	0x2B	<Not implemented>	N/A
CSAFE_SETTORQUE_CMD	0x2C	<Not implemented>	N/A
CSAFE_SETLEVEL_CMD	0x2D	<Not implemented>	N/A
CSAFE_SETTARGETHR_CMD	0x30	<Not implemented>	N/A
CSAFE_SETMETS_CMD	0x33	<Not implemented>	N/A
CSAFE_SETPOWER_CMD	0x34	Byte 0: Stroke Watts (LSB) Byte 1: Stroke Watts (MSB) Byte 2: Units Specifier	N/A
CSAFE_SETHRZONE_CMD	0x35	<Not implemented>	N/A
CSAFE_SETHRMAX_CMD	0x36	<Not implemented>	N/A
CSAFE_SETCHANNELRANGE_CMD	0x40	<Not implemented>	N/A
CSAFE_SETVOLUMERANGE_CMD	0x41	<Not implemented>	N/A
CSAFE_SETAUDIOMUTE_CMD	0x42	<Not implemented>	N/A
CSAFE_SETAUDIOCHANNEL_CMD	0x43	<Not implemented>	N/A
CSAFE_SETAUDIOVOLUME_CMD	0x44	<Not implemented>	N/A
CSAFE_STARTTEXT_CMD	0x60	<Not implemented>	N/A
CSAFE_APPENDTEXT_CMD	0x61	<Not implemented>	N/A
CSAFE_GETTEXTSTATUS_CMD	0x65	<Not implemented>	N/A
CSAFE_GETCAPS_CMD	0x70	Byte 0: Capability Code	Capability Code 0x00: Byte 0: Max Rx Frame Byte 1: Max Tx Frame Byte 2: Min Interframe

			Capability Code 0x01: Byte 0: 0x00 Byte 1: 0x00 Capability Code 0x02: Byte 0: 0x00 Byte 1: 0x00 Byte 2: 0x00 Byte 3: 0x00 Byte 4: 0x00 Byte 5: 0x00 Byte 6: 0x00 Byte 7: 0x00 Byte 8: 0x00 Byte 9: 0x00 Byte 10: 0x00
CSAFE_SETPMCFG_CMD[1]	0x76	1 or more C2 proprietary CSAFE commands	See C2 proprietary commands
CSAFE_SETPMDATA_CMD[1]	0x77	1 or more C2 proprietary CSAFE commands	See C2 proprietary commands
CSAFE_GETPMCFG_CMD[1]	0x7E	1 or more C2 proprietary CSAFE commands	See C2 proprietary commands
CSAFE_GETPMDATA_CMD[1]	0x7F	1 or more C2 proprietary CSAFE commands	See C2 proprietary commands

Notes:
1. Added for PM-specific functionality as command wrappers. These are equivalent to the CSAFE_GETUSERCAPS1_CMD and CSAFE_GETUSERCAPS2_CMD commands defined in the Public CSAFE protocol documentation.

C2 Proprietary Short Commands

Command Name	Command Identifier	Response Data
CSAFE_PM_GET_WORKOUTTYPE	0x89	Byte 0: Workout type
CSAFE_PM_GET_WORKOUTSTATE	0x8D	Byte 0: Workout State
CSAFE_PM_GET_INTERVALTYPE	0x8E	Byte 0: Interval Type
CSAFE_PM_GET_WORKOUTINTERVALCOUNT	0x9F	Byte 0: Workout Interval Count
CSAFE_PM_GET_WORKTIME	0xA0	Byte 0: Work Time (LSB) Byte 1: Work Time Byte 2: Work Time Byte 3: Work Time (MSB) Byte 4: Fractional Work Time
CSAFE_PM_GET_WORKDISTANCE	0xA3	Byte 0: Work Distance (LSB) Byte 1: Work Distance Byte 2: Work Distance Byte 3: Work Distance (MSB) Byte 4: Fractional Work Distance
CSAFE_PM_GET_ERRORVALUE[2]	0xC9	Byte 0: Error Value (LSB) Byte 1: Error Value (MSB)
CSAFE_PM_GET_RESTTIME	0xCF	Byte 0: Rest Time (LSB) Byte 1: Rest Time (MSB)

Notes:

1. The above commands are sent using the CSAFE_SETUSERCFG1_CMD command wrapper discussed in PM Extensions.
2. The ERRORVALUE command will serve to clear the latched error value in the PM3 when the Screen Error Display Mode is DISABLED

C2 Proprietary Long Commands

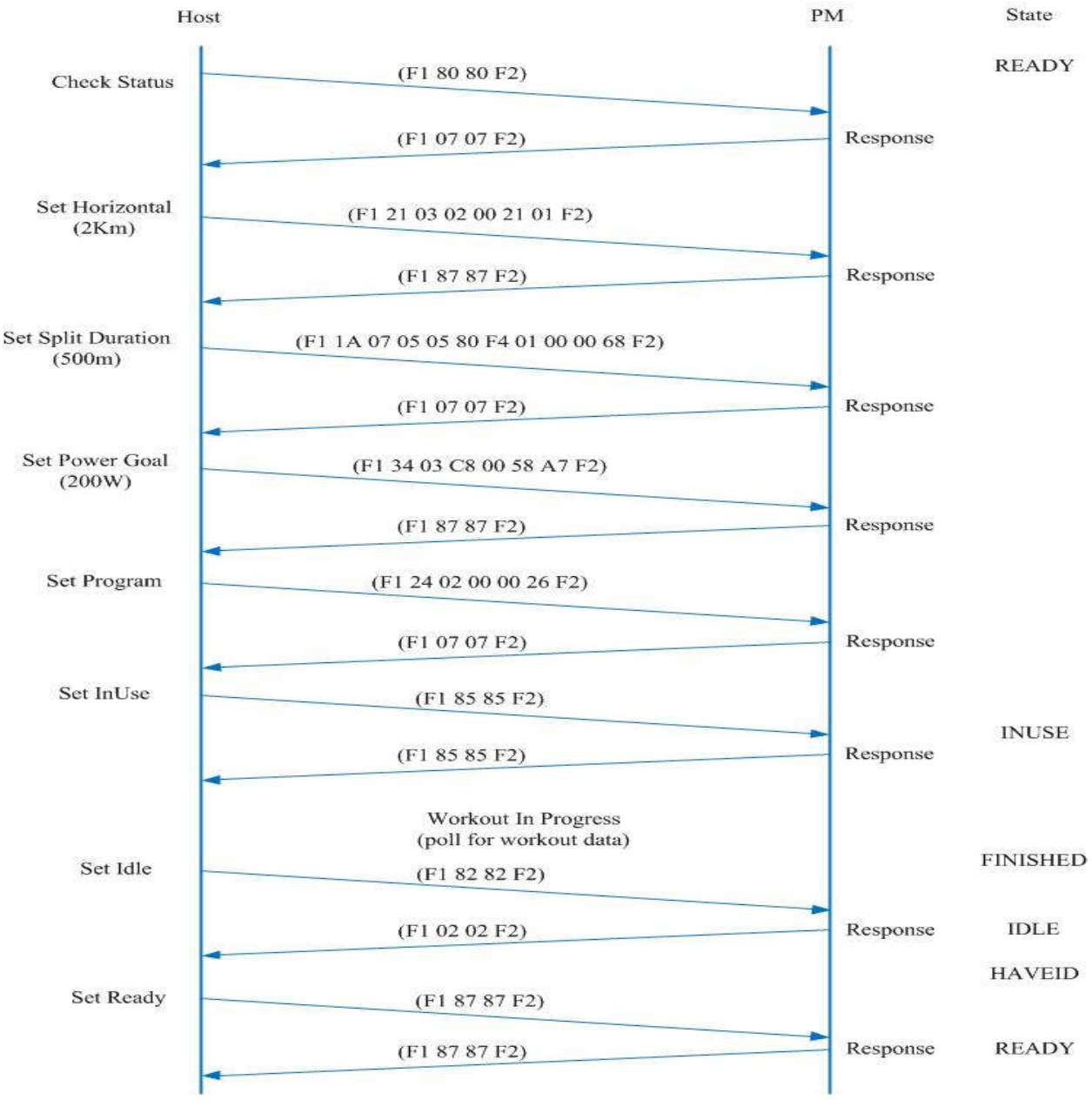
Command Name	Command Identifier	Command Data	Response Data
CSAFE_PM_SET_SPLITDURATION	0x05	Byte 0: Time/Distance duration (0: Time, 128: Distance) Byte 1: Duration (LSB) Byte 2: Duration Byte 3: Duration Byte 4: Duration (MSB)	N/A
CSAFE_PM_GET_FORCEPLOTDATA[2]	0x6B	Byte 0: Block length in bytes	Byte 0: Bytes read Byte 1: 1st data read (LSB) Byte 2: 1st data read (MSB) Byte 3: 2nd data read (LSB) . . . Byte 33: 16th data read (MSB)
CSAFE_PM_SET_SCREENERRORMODE	0x27	Byte 0: Mode (0: Disable, 1: Enable)	N/A
CSAFE_PM_GET_HEARTBEATDATA[3]	0x6C	Byte 0: Block length in bytes	Byte 0: Bytes read Byte 1: 1st data read (LSB) Byte 2: 1st data read (MSB) Byte 3: 2nd data read (LSB) . . . Byte 33: 16th data read (MSB)

Notes:
1. The above commands are sent using the CSAFE_SETUSERCFG1_CMD command wrapper discussed previously.
2. A maximum block length of 32 bytes (16 words) can be read. Fewer words can be read by specifying the block length accordingly, but a complete 33 bytes of response data will be returned. The first byte of the response will indicate how many valid data bytes are returned.
3. A maximum block length of 32 bytes (16 words) can be read. Fewer words can be read by specifying the block length accordingly, but a complete 33 bytes of response data will be returned. Only data samples recorded since the last read will be returned. The first byte of the response will indicate how many valid data bytes are returned.

Setting Up and Performing Workout

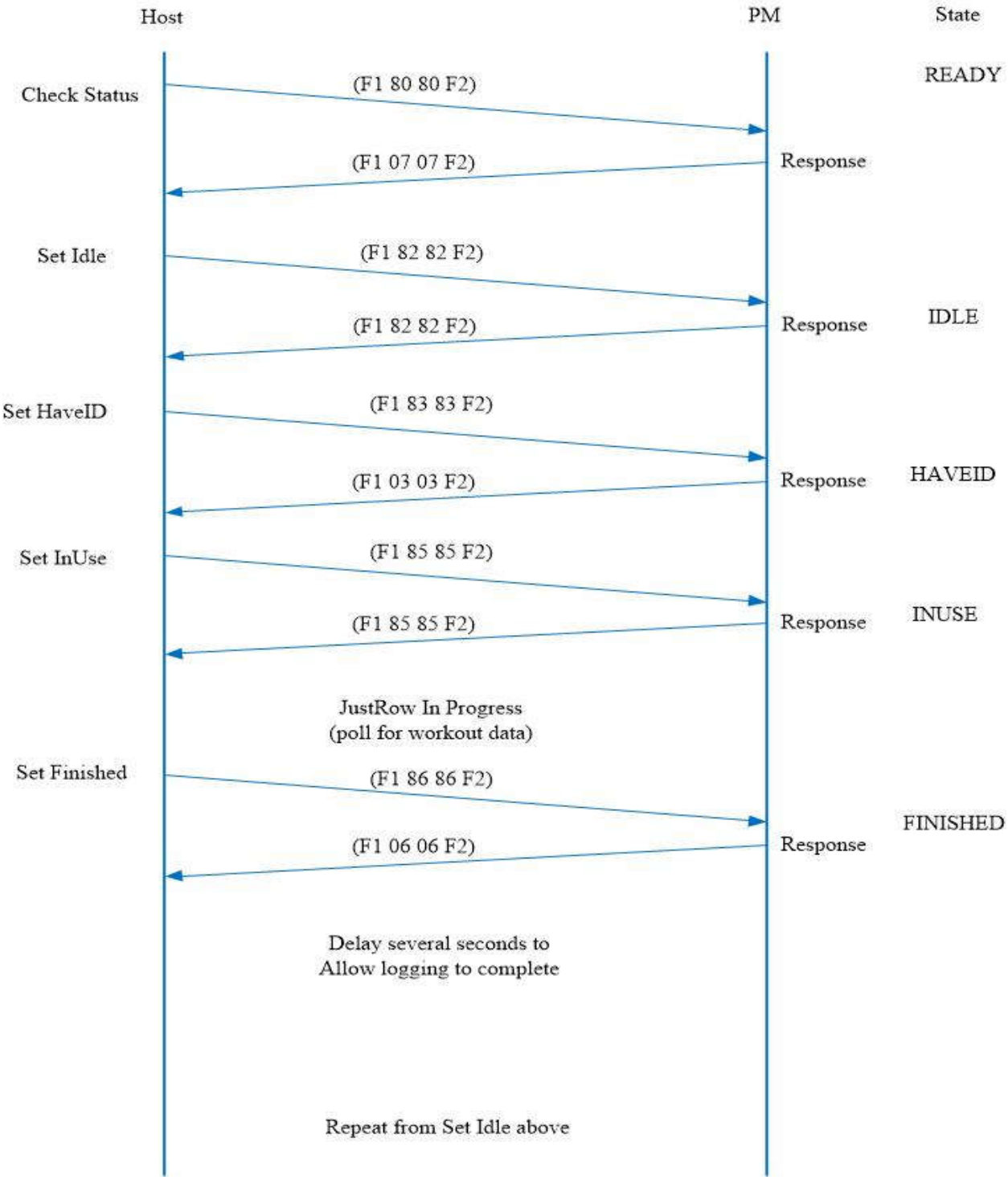
When the PM is turned on the CSAFE state machine is in the READY state (see Figure 7). From this state a workout can be configured and then the PM moved to the workout screen (INUSE state). The PM's progress throughout the workout can be monitored including the state machine, and out workout conclusion final results can be collected.

Figure 8 – Example Public CSAFE PM Workout Setup and Progress Monitoring



Performing a simple successive use of JustRow workouts to meet a series of workout goals can be achieved as illustrated in the following figure.

Figure 9 – Example Public CSAFE PM Successive JustRow Workouts



Proprietary CSAFE

Special Consideration

ScreenType Commands

The ScreenType command is unique in that it is initially processed by the communicaton task and "posted" for processing by the UI task. The CSAFE frame response is sent immediately by the communications task. Since the UI task only runs periodically (e.g., 2 - 5 Hz) there is some delay before the full affect of the command is realized.

The options are to delay sufficiently long for the command to complete (e.g., 1 second or more), or to poll for the status of ScreenType commands. Using the CSAFE_PM_GET_SCREENSTATESTATUS, the status will be set to APGLOBALS_SCREENPENDINGFLG_PENDING when the command is received. The status will change to APGLOBALS_SCREENPENDINGFLG_INPROGRESS while processing and set to APGLOBALS_SCREENPENDINGFLG_INACTIVE when complete. Note that depending on the polling rate, one or more status values may not be visible.

Maximum Block Size Commands

Certain commands have maximum block size limitations. An additional command parameter defines the number of valid bytes in the command block.

Figure 10 – Maximum Block Size Commands

Block Size, (Bytes)	Commands w/ Fixed Size Responses
64	CSAFE_PM_SET_DISPLAYBITMAP CSAFE_PM_SET_LOGCARDMEMORY

Fixed Block Size Command Responses

Certain commands have fixed size responses in if the requested data is less than the specified block size. An additional response parameter defines the number of valid bytes in the fixed size response.

Figure 11 – Fixed Block Sized Command Responses

Block Size, (Bytes)	Commands w/ Fixed Size Responses
64	CSAFE_PM_GET_MEMORY CSAFE_PM_GET_LOGCARDMEMORY CSAFE_PM_GET_INTERNALLOGMEMORY
32	CSAFE_PM_GET_FORCEPLOTDATA CSAFE_PM_GET_HEARTBEATDATA

Command List

C2 Proprietary Short Get Configuration Commands

Command Name	Command Identifier	Response Data
CSAFE_PM_GET_FW_VERSION	0x80	Byte 0: FW Exe Version # (MSB) Byte 1: FW Exe Version # Byte 2: FW Exe Version # .

		. . Byte 15: FW Exe Version # (LSB)
CSAFE_PM_GET_HW_VERSION	0x81	Byte 0: HW Version # (MSB) Byte 1: HW Version # Byte 2: HW Version # . . . Byte 15: HW Version # (LSB)
CSAFE_PM_GET_HW_ADDRESS	0x82	Byte 0: HW address (MSB) Byte 1: HW address Byte 2: HW address Byte 3: HW address (LSB)
CSAFE_PM_GET_TICK_TIMEBASE	0x83	Byte 0: Tick timebase (Float MSB) Byte 1: Tick timebase Byte 2: Tick timebase Byte 3: Tick timebase (Float LSB)
CSAFE_PM_GET_HRM	0x84	Byte 0: Channel Status 0 – Inactive 1 - Discovery 2 – Paired If paired then: Byte 1: Device Manufacture ID Byte 2: Device Type Byte 3: Device Num (MSB) Byte 4: Device Num (LSB) Else Bytes 1 - 4: 0
CSAFE_PM_GET_DATETIME	0x85	Byte 0: Time Hours (1 – 12) Byte 1: Time Minutes (0 – 59) Byte 2: Time Meridiem (0 – AM, 1 – PM) Byte 3: Date Month (1 – 12) Byte 4: Date Day (1 – 31) Byte 5: Date Year (MSB) Byte 6: Date Year (LSB)
CSAFE_PM_GET_SCREENSTATESTATUS	0x86	Byte 0: Screen type Byte 1: Screen value Byte 2: Screen status
CSAFE_PM_GET_RACE_LANE_REQUEST	0x87	Byte 0: Erg Physical Address
CSAFE_PM_GET_RACE_ENTRY_REQUEST	0x88	Byte 0: Erg Logical Address
CSAFE_PM_GET_WORKOUTTYPE	0x89	Byte 0: Workout type
CSAFE_PM_GET_DISPLAYTYPE	0x8A	Byte 0: Display type
CSAFE_PM_GET_DISPLAYUNITS	0x8B	Byte 0: Display units
CSAFE_PM_GET_LANGUAGETYPE	0x8C	Byte 0: Language type
CSAFE_PM_GET_WORKOUTSTATE	0x8D	Byte 0: Workout state
CSAFE_PM_GET_INTERVALTYPE	0x8E	Byte 0: Interval type
CSAFE_PM_GET_OPERATIONALSTATE	0x8F	Byte 0: Operational state
CSAFE_PM_GET_LOGCARDSTATE	0x90	Byte 0: Log card state
CSAFE_PM_GET_LOGCARDSTATUS	0x91	Byte 0: Log card status
CSAFE_PM_GET_POWERUPSTATE	0x92	Byte 0: Power-up state
CSAFE_PM_GET_ROWINGSTATE	0x93	Byte 0: Rowing state
CSAFE_PM_GET_SCREENCONTENT_VERSION	0x94	Byte 0: Screen Content Version # (MSB) Byte 1: Screen Content e Version # Byte 2: Screen Content Version #

		. . . Byte 15: Screen Content Version # (LSB)
CSAFE_PM_GET_COMMUNICATIONSTATE	0x95	Byte 0: Communication state
CSAFE_PM_GET_RACEPARTICIPANTCOUNT	0x96	Byte 0: Race Participant Count
CSAFE_PM_GET_BATTERYLEVELPERCENT	0x97	Byte 0: Battery Level Percent
CSAFE_PM_GET_RACEMODESTATUS	0x98	Byte 0: HW address (MSB) Byte 1: HW address Byte 2: HW address Byte 3: HW address (LSB) Byte 4: Race Operation Type Byte 5: Race State Byte 6: Race Start State Byte 7: Rowing State Byte 8: EPM Status Byte 9: Battery Level Percent PM3/PM4: Byte 10: Avg Flywheel RPM (MSB) Byte 11: Avg Flywheel RPM (LSB) PM5: Byte 10: Tach wire test status Byte 11: Tach Simulator status Byte 12: Workout State Byte 13: Workout Type Byte 14: Operational State
CSAFE_PM_GET_INTERNALLOGPARAMS	0x99	Byte 0: Log Start Address (MSB) Byte 1: Log Start Address Byte 2: Log Start Address Byte 3: Log Start Address (LSB) Byte 4: Last Log Entry Length (MSB) Byte 5: Last Log Entry Length (LSB)
CSAFE_PM_GET_PRODUCTCONFIGURATION	0x9A	Byte 0: PM Base HW Revision (MSB) Byte 1: PM Base HW Revision (LSB) Byte 2: PM Base SW Revision (MSB) Byte 3: PM Base SW Revision (LSB) Byte 4: Exp Card Status Byte 5: Exp Card CF Status Byte 6: Exp Card HW Revision (MSB) Byte 7: Exp Card HW Revision (LSB) Byte 8: Exp Card SW Revision (MSB) Byte 9: Exp Card SW Revision (LSB)
CSAFE_PM_GET_ERGSLAVEDISCOVERREQUESTSTATUS	0x9B	Byte 0: Status Byte 1: # of Erg slaves present
CSAFE_PM_GET_WIFICONFIG	0x9C	Byte 0: Configuration Index Byte 1: WEP Mode
CSAFE_PM_GET_CPUTICKRATE	0x9D	Byte 0: CPU/Tick Rate Enumeration
CSAFE_PM_GET_LOGCARDUSERCENSUS	0x9E	Byte 0: Number Users on Card Byte 1: Number of Current User
CSAFE_PM_GET_WORKOUTINTERVALCOUNT	0x9F	Byte 0: Workout Interval Count
CSAFE_PM_GET_WORKOUTDURATION	0xE8	Byte 0: Time/Distance duration (0: Time, 0x40: Calories, 0xC0: Watt-Min, 0x80: Distance) Byte 1: Duration (MSB) Byte 2: Duration

		Byte 3: Duration Byte 4: Duration (LSB)
CSAFE_PM_GET_WORKOTHER	0xE9	Byte 0: Work Other (MSB) Byte 1: Work Other Byte 2: Work Other Byte 3: Work Other (LSB)
CSAFE_PM_GET_EXTENDED_HRM	0xEA	Byte 0: HRM Channel Status Byte 1: HRM manufacturer ID Byte 2: HRM device type Byte 3: HRM device number (MSB) Byte 4: HRM device number Byte 5: HRM device number Byte 6: HRM device number (LSB)
CSAFE_PM_GET_DEFCALIBRATIONVERFIED	0xEB	Byte 0: DF Calibration Verified Status
CSAFE_PM_GET_FLYWHEELSPEED	0xEC	Byte 0: Flywheel speed, rpm (MSB) Byte 1: Flywheel speed, rpm (LSB)
CSAFE_PM_GET_ERGMACHINETYPE	0xED	Byte 0: Erg machine type
CSAFE_PM_GET_RACE_BEGINEND_TICKCOUNT	0xEE	Byte 0: Race begin tick time, (MSB) Byte 1: Race begin tick time Byte 2: Race begin tick time Byte 3: Race begin tick time (LSB) Byte 4: Race end tick time (MSB) Byte 5: Race end tick time Byte 6: Race end tick time Byte 7: Race end tick time (LSB)
CSAFE_PM_GET_PM5_FWUPDATESTATUS	0xEF	Byte 0: Update info type (MSB) Byte 1: Update info type (LSB) Byte 2: Update status (MSB) Byte 3: Update status (LSB)

C2 Proprietary Long Get Configuration Commands

Command Name	Command Identifier	Command Data	Response Data
CSAFE_PM_GET_ERG_NUMBER	0x50	Byte 0: HW address[1] (MSB) Byte 1: HW address Byte 2: HW address Byte 3: HW address (LSB)	Byte 0: Erg #
CSAFE_PM_GET_ERGNUMBERREQUEST	0x51	Byte 0: Logical Erg Number Requested Byte 1: Physical Erg Number Requested	Byte 0: Logical Erg # Byte 1: HW address[1] (MSB) Byte 2: HW address Byte 3: HW address Byte 4: HW address (LSB)s Byte 5: Physical Erg #
CSAFE_PM_GET_USERIDSTRING	0x52	Byte 0: User Number	Byte 0: User ID (MSB) Byte 1: User ID Byte 2: User ID . . . Byte 9: User ID (LSB)
CSAFE_PM_GET_LOCALRACEPARTICIP	0x53	Byte 0: Race Type	Byte 0: HW address

ANT		Byte 1: Race Length (MSB) Byte 2: Race Length Byte 3: Race Length Byte 4: Race Length (LSB) Byte 5: Race Participants Byte 6: Race State	(MSB) Byte 1: HW address Byte 2: HW address Byte 3: HW address (LSB) Byte 4: UserID String (MSB) Byte 5: UserID String Byte 6: UserID String Byte 7: UserID String Byte 8: UserID String Byte 9: UserID String (LSB) Byte 10: Machine type
CSAFE_PM_GET_USER_ID	0x54	Byte 0: User Number	Byte 0: User Number Byte 1: User ID (MSB) Byte 2: User ID Byte 3: User ID Byte 4: User ID (LSB)
CSAFE_PM_GET_USER_PROFILE	0x55	Byte 0: User Number	Byte 0: User Number Byte 1: User Weight (MSB) Byte 2: User Weight (LSB) Byte 3: User DOB Day Byte 4: User DOB Month Byte 5: User DOB Year (MSB) Byte 6: User DOB Year (LSB) Byte 7: User Gender
CSAFE_PM_GET_HRBELT_INFO	0x56	Byte 0: User Number	Byte 0: User Number Byte 1: Mfg ID Byte 2: Device Type Byte 3: Belt ID (MSB) Byte 4: Belt ID (LSB)
CSAFE_PM_GET_EXTENDED_HRBELT_INFO	0x57	Byte 0: User Number	Byte 0: User Number Byte 1: Mfg ID Byte 2: Device Type Byte 3: Belt ID (MSB) Byte 4: Belt ID Byte 5: Belt ID Byte 6: Belt ID (LSB)
CSAFE_PM_GET_CURRENT_LOG_STRUCTURE	0x58	Byte 0: Structure ID enumeration Byte 1: Split/interval number (1 – M)	Byte 0: Structure ID enumeration Byte 1: Split/interval number Byte 2: Bytes read Byte 3: 1st data read Byte 4: 2nd data read . . . Byte N + 2: Nth data read

Notes:

The hardware address is the unit serial # as stored in MFG EEPROM and accessible by command CSAFE_PM_GET_HW_ADDRESS.

The CSAFE_PM_GET_LOCALRACEPARTICIPANT command is only available for firmware that supports PCless racing.

C2 Proprietary Short Get Data Commands

Command Name	Command Identifier	Response Data
CSAFE_PM_GET_WORKTIME	0xA0	Byte 0: Work Time (MSB) Byte 1: Work Time Byte 2: Work Time Byte 3: Work Time (LSB)
CSAFE_PM_GET_PROJECTED_WORKTIME	0xA1	Byte 0: Projected Work Time (MSB) Byte 1: Projected Work Time Byte 2: Projected Work Time Byte 3: Projected Work Time (LSB)
CSAFE_PM_GET_TOTAL_RESTTIME	0xA2	Byte 0: Total Rest Time (MSB) Byte 1: Total Rest Time Byte 2: Total Rest Time Byte 3: Total Rest Time (LSB)
CSAFE_PM_GET_WORKDISTANCE	0xA3	Byte 0: Work Distance (MSB) Byte 1: Work Distance Byte 2: Work Distance Byte 3: Work Distance (LSB)
CSAFE_PM_GET_TOTAL_WORKDISTANCE	0xA4	Byte 0: Total Work Distance (MSB) Byte 1: Total Work Distance Byte 2: Total Work Distance Byte 3: Total Work Distance (LSB)
CSAFE_PM_GET_PROJECTED_WORKDISTANCE	0xA5	Byte 0: Projected Work Distance (MSB) Byte 1: Projected Work Distance Byte 2: Projected Work Distance Byte 3: Projected Work Distance (LSB)
CSAFE_PM_GET_RESTDISTANCE	0xA6	Byte 0: Rest Distance (MSB) Byte 1: Rest Distance (LSB)
CSAFE_PM_GET_TOTAL_RESTDISTANCE	0xA7	Byte 0: Total Rest Distance (MSB) Byte 1: Total Rest Distance Byte 2: Total Rest Distance Byte 3: Total Rest Distance (LSB)
CSAFE_PM_GET_STROKE_500M_PACE	0xA8	Byte 0: Pace / 500m (MSB) Byte 1: Pace / 500m Byte 2: Pace / 500m Byte 3: Pace / 500m (LSB)
CSAFE_PM_GET_STROKE_POWER	0xA9	Byte 0: Stroke Watts (MSB) Byte 1: Stroke Watts Byte 2: Stroke Watts Byte 3: Stroke Watts (LSB)
CSAFE_PM_GET_STROKE_CALORICBURNRATE	0xAA	Byte 0: Stroke Cals/Hr (MSB) Byte 1: Stroke Cals/Hr Byte 2: Stroke Cals/Hr Byte 3: Stroke Cals/Hr (LSB)
CSAFE_PM_GET_SPLIT_AVG_500M_PACE	0xAB	Byte 0: Split Avg Pace / 500m (MSB) Byte 1: Split Avg Pace / 500m Byte 2: Split Avg Pace / 500m

		Byte 3: Split Avg Pace / 500m (LSB)
CSAFE_PM_GET_SPLIT_AVG_POWER	0xAC	Byte 0: Split Avg Watts (MSB) Byte 1: Split Avg Watts Byte 2: Split Avg Watts Byte 3: Split Avg Watts (LSB)
CSAFE_PM_GET_SPLIT_AVG_CALORICBURN RATE	0xAD	Byte 0: Split Avg Cals/Hr (MSB) Byte 1: Split Avg Cals/Hr Byte 2: Split Avg Cals/Hr Byte 3: Split Avg Cals/Hr (LSB)
CSAFE_PM_GET_SPLIT_AVG_CALORIES	0xAE	Byte 0: Split Avg Cals (MSB) Byte 1: Split Avg Cals Byte 2: Split Avg Cals Byte 3: Split Avg Cals (LSB)
CSAFE_PM_GET_TOTAL_AVG_500MPACE	0xAF	Byte 0: Total Avg Pace / 500m (MSB) Byte 1: Total Avg Pace / 500m Byte 2: Total Avg Pace / 500m Byte 3: Total Avg Pace / 500m (LSB)
CSAFE_PM_GET_TOTAL_AVG_POWER	0xB0	Byte 0: Total Avg Watts (MSB) Byte 1: Total Avg Watts Byte 2: Total Avg Watts Byte 3: Total Avg Watts (LSB)
CSAFE_PM_GET_TOTAL_AVG_CALORICBUR NRATE	0xB1	Byte 0: Total Avg Cals/Hr (MSB) Byte 1: Total Avg Cals/Hr Byte 2: Total Avg Cals/Hr Byte 3: Total Avg Cals/Hr (LSB)
CSAFE_PM_GET_TOTAL_AVG_CALORIES	0xB2	Byte 0: Total Avg Calories (MSB) Byte 1: Total Avg Calories Byte 2: Total Avg Calories Byte 3: Total Avg Calories (LSB)
CSAFE_PM_GET_STROKE_RATE	0xB3	Byte 0: Strokes/Min
CSAFE_PM_GET_SPLIT_AVG_STROKERATE	0xB4	Byte 0: Split/Interval Avg Strokes/Min
CSAFE_PM_GET_TOTAL_AVG_STROKERATE	0xB5	Byte 0: Total Avg Strokes/Min
CSAFE_PM_GET_AVG_HEART_RATE	0xB6	Byte 0: Avg Beats/Min
CSAFE_PM_GET_ENDING_AVG_HEARTRATE	0xB7	Byte 0: Split/Interval Avg Beats/Min
CSAFE_PM_GET_REST_AVG_HEARTRATE	0xB8	Byte 0: Rest Interval Avg Beats/Min
CSAFE_PM_GET_SPLITTIME	0xB9	Byte 0: Elapsed Time / Split (MSB) Byte 1: Elapsed Time / Split Byte 2: Elapsed Time / Split Byte 3: Elapsed Time / Split (LSB)
CSAFE_PM_GET_LAST_SPLITTIME	0xBA	Byte 0: Last Elapsed Time / Split (MSB) Byte 1: Last Elapsed Time / Split Byte 2: Last Elapsed Time / Split Byte 3: Last Elapsed Time / Split (LSB)
CSAFE_PM_GET_SPLITDISTANCE	0xBB	Byte 0: Work Distance/Split (MSB) Byte 1: Work Distance/Split Byte 2: Work Distance/Split Byte 3: Work Distance/Split (LSB)
CSAFE_PM_GET_LAST_SPLITDISTANCE	0xBC	Byte 0: Last Work Distance/Split (MSB) Byte 1: Last Work Distance/Split Byte 2: Last Work Distance/Split Byte 3: Last Work Distance/Split (LSB)
CSAFE_PM_GET_LAST_RESTDISTANCE	0xBD	Byte 0: Last Rest Interval Distance (MSB) Byte 1: Last Rest Interval Distance Byte 2: Last Rest Interval Distance Byte 3: Last Rest Interval Distance (LSB)

CSAFE_PM_GET_TARGETPACETIME	0xBE	Byte 0: Target Pace Time (MSB) Byte 1: Target Pace Time Byte 2: Target Pace Time Byte 3: Target Pace Time (LSB)
CSAFE_PM_GET_STROKESTATE	0xBF	Byte 0: Stroke State
CSAFE_PM_GET_STROKERATESTATE	0xC0	Byte 0: Stroke Rate State
CSAFE_PM_GET_DRAGFACTOR	0xC1	Byte 0: Drag Factor
CSAFE_PM_GET_ENCODER_PERIOD	0xC2	Byte 0: Encoder Period (Float MSB) Byte 1: Encoder Period Byte 2: Encoder Period Byte 3: Encoder Period (Float LSB)
CSAFE_PM_GET_HEARTRATESTATE	0xC3	Byte 0: Heartrate State
CSAFE_PM_GET_SYNC_DATA	0xC4	Byte 0: Sync Data (Float MSB) Byte 1: Sync Data Byte 2: Sync Data Byte 3: Sync Data (Float LSB)
CSAFE_PM_GET_SYNCDATAALL	0xC5	Byte 0: Work Distance (Float MSB) Byte 1: Work Distance Byte 2: Work Distance Byte 3: Work Distance (Float LSB) Byte 4: Work Time (Float MSB) Byte 5: Work Time Byte 6: Work Time Byte 7: Work Time (Float LSB) Byte 8: Stroke Pace (Float MSB) Byte 9: Stroke Pace Byte 10: Stroke Pace Byte 11: Stroke Pace (Float LSB) Byte 12: Avg Heartrate (Float MSB) Byte 13: Avg Heartrate Byte 14: Avg Heartrate Byte 15: Avg Heartrate (Float LSB)
CSAFE_PM_GET_RACE_DATA	0xC6	Byte 0: Tick Time Stamp (MSB) Byte 1: Tick Time Stamp Byte 2: Tick Time Stamp Byte 3: Tick Time Stamp (LSB) Byte 4: Total Race Meters (MSB) Byte 5: Total Race Meters Byte 6: Total Race Meters Byte 7: Total Race Meters (LSB) Byte 8: 500m Pace (MSB) Byte 9: 500m Pace (LSB) Byte 10: Race Elapsed Time (MSB) Byte 11: Race Elapsed Time Byte 12: Race Elapsed Time Byte 13: Race Elapsed Time (LSB) Byte 14: Stroke Rate Byte 15: Race State Byte 16: Percent Battery Level Byte 17: Stroke State Byte 18: Rowing Byte 19: EPM Status Byte 20: Race Operation Type Byte 21: Race Start State
CSAFE_PM_GET_TICK_TIME	0xC7	Byte 0: Tick Time (MSB)

		Byte 1: Tick Time Byte 2: Tick Time Byte 3: Tick Time (LSB)
CSAFE_PM_GET_ERRORTYPE	0xC8	Byte 0: Error Type
CSAFE_PM_GET_ERRORVALUE	0xC9	Byte 0: Error Value (MSB) Byte 1: Error Value (LSB)
CSAFE_PM_GET_STATUSTYPE	0xCA	Byte 0: Status Type
CSAFE_PM_GET_STATUSVALUE	0xCB	Byte 0: Status Value
CSAFE_PM_GET_EPMSTATUS	0xCC	Byte 0: EPM Status
CSAFE_PM_GET_DISPLAYUPDATETIME	0xCD	Byte 0: Display Update Time (MSB) Byte 1: Display Update Time Byte 2: Display Update Time Byte 3: Display Update Time (LSB)
CSAFE_PM_GET_SYNCFRACTIONALTIME	0xCE	Byte 0: EPM Fractional Time
CSAFE_PM_GET_RESTTIME	0xCF	Byte 0: Rest Time (LSB) Byte 1: Rest Time (MSB)

C2 Proprietary Long Get Data Commands

Command Name	Command Identifier	Command Data	Response Data
CSAFE_PM_GET_MEMORY[1]	0x68	Byte 0: Device type (0: ESRAM 1: Ext SRAM 2: FLASH) Byte 1: Start address (MSB) Byte 2: Start address Byte 3: Start address Byte 4: Start address (LSB) Byte 5: Block length	Byte 0: Bytes read Byte 1: 1st data read Byte 2: 2nd data read . . . Byte 64:64nd data read
CSAFE_PM_GET_LOGCARD_MEMORY[1]	0x69	Byte 0: Start address (MSB) Byte 1: Start address Byte 2: Start address Byte 3: Start address (LSB) Byte 4: Block length	Byte 0: Bytes read Byte 1: 1st data read Byte 2: 2nd data read . . . Byte 64: 64nd data read
CSAFE_PM_GET_INTERNALLOGMEMORY[1]	0x6A	Byte 0: Start address (MSB) Byte 1: Start address Byte 2: Start address Byte 3: Start address (LSB) Byte 4: Block length	Byte 0: Bytes read Byte 1: 1st data read Byte 2: 2nd data read . . . Byte 64: 64nd data read
CSAFE_PM_GET_FORCEPLOTDATA[2]	0x6B	Byte 0: Block length	Byte 0: Bytes read Byte 1: 1st data read (MSB) Byte 2: 1st data read (LSB) Byte 3: 2nd data read (MSB) . . .

			Byte 33: 16th data read (LSB)
CSAFE_PM_GET_HEARTBEATDATA[3]	0x6C	Byte 0: Block length in bytes	Byte 0: Bytes read Byte 1: 1st data read (LSB) Byte 2: 1st data read (MSB) Byte 3: 2nd data read (LSB) . . . Byte 33: 16th data read (MSB)
CSAFE_PM_GET_UI_EVENTS	0x6D	Byte 0: 0 (unused)	Byte 0: User I/F Events (MSB) Byte 1: User I/F Events (LSB)
CSAFE_PM_GET_STROKESTATS	0x6E	Byte: 0: 0 (unused)	Byte 0: Stroke Distance (MSB) Byte 1: Stroke Distance (LSB) Byte 2: Stroke Drive Time Byte 3: Stroke Recovery Time (MSB) Byte 4: Stroke Recovery Time (LSB) Byte 5: Stroke Length Byte 6: Drive Counter (MSB) Byte 7: Drive Counter (LSB) Byte 8: Peak Drive Force (MSB) Byte 9: Peak Drive Force (LSB) Byte 10: Impulse Drive Force (MSB) Byte 11: Impulse Drive Force (LSB) Byte 12: Avg Drive Force (MSB) Byte 13: Avg Drive Force (LSB) Byte 14: Work Per Stroke (MSB) Byte 15: Work Per Stroke (LSB)
CSAFE_PM_GET_DIAGLOG_RECORD_NUM	0x70	Byte 0: Record Type (Enum)	Byte 0: Record Type (Enum) Byte 1: Record Num (MSB) Byte 2: Record Num (LSB)
CSAFE_PM_GET_DIAGLOG_RECORD	0x71	Byte 0: Record Type	Byte 0: Record Type

		(Enum) Byte 1: Record Index (MSB) Byte 2: Record Index (LSB) Byte 3: Record Offset Bytes (MSB) Byte 4: Record Offset Bytes (LSB)	(Enum) Byte 1: Record Index (MSB) Byte 2: Record Index (LSB) Byte 3: Valid Record Bytes (MSB) Byte 4: Valid Record Bytes (LSB) Byte 5: 1st data read Byte 6: 2nd data read . . . Byte 72: 68nd data read
	0x74	Internal Use	
	0x75	Internal Use	
	0x76	Command Wrapper	
	0x77	Command Wrapper	
	0x7E	Command Wrapper	
	0x7F	Command Wrapper	

Notes:
1. A maximum block length of 64 bytes can be read. Fewer bytes can be read by specifying the block length accordingly, but a complete 65 bytes of response data will be returned.
2. A maximum block length of 32 bytes (16 words) can be read. Fewer words can be read by specifying the block length accordingly, but a complete 33 bytes of response data will be returned.
3. A maximum block length of 32 bytes (16 words) can be read. Fewer words can be read by specifying the block length accordingly, but a complete 33 bytes of response data will be returned. Only data samples recorded since the last read will be returned. The first byte of the response will indicate how many valid data bytes are returned.

C2 Proprietary Short Set Configuration Commands

Command Name	Command Identifier	Response Data
CSAFE_PM_SET_RESET_ALL	0xE0	<Not implemented>
CSAFE_PM_SET_RESET_ERG_NUMBER	0xE1	N/A

C2 Proprietary Short Set Data Commands

Command Name	Command Identifier	Response Data
CSAFE_PM_SET_SYNC_DISTANCE	0xD0	N/A
CSAFE_PM_SET_SYNC_STROKE_PACE	0xD1	N/A
CSAFE_PM_SET_SYNC_AVG_HEARTRATE	0xD2	N/A
CSAFE_PM_SET_SYNC_TIME	0xD3	N/A
CSAFE_PM_SET_SYNC_SPLIT_DATA	0xD4	<Not implemented>
CSAFE_PM_SET_SYNC_ENCODER_PERIOD	0xD5	<Not implemented>
CSAFE_PM_SET_SYNC_VERSION_INFO	0xD6	<Not implemented>
CSAFE_PM_SET_SYNC_RACETICKTIME	0xD7	N/A
CSAFE_PM_SET_SYNC_DATAALL	0xD8	N/A
CSAFE_PM_SET_SYNC_ROWINGACTIVE_TIME	0xD9	N/A

C2 Proprietary Long Set Configuration Commands

Command Name	Command Identifier	Command Data	Response Data
CSAFE_PM_SET_BAUDRATE	0x00	<Not implemented>	
CSAFE_PM_SET_WORKOUTTYPE	0x01	Byte 0: Workout Type	N/A
CSAFE_PM_SET_STARTTYPE	0x02	<Not implemented>	
CSAFE_PM_SET_WORKOUTDURATION	0x03	Byte 0: Time/Distance duration (0: Time, 0x40: Calories, 0x60: Watt-Min, 0x80: Distance) Byte 1: Duration (MSB) Byte 2: Duration Byte 3: Duration Byte 4: Duration (LSB)	N/A
CSAFE_PM_SET_RESTDURATION	0x04	Byte 0: Duration (MSB) Byte 1: Duration (LSB)	N/A
CSAFE_PM_SET_SPLITDURATION	0x05	Byte 0: Time/Distance duration (0: Time, 0x40: Calories, 0xC0: Watt-Min, 0x80: Distance) Byte 1: Duration (MSB) Byte 2: Duration Byte 3: Duration Byte 4: Duration (LSB)	N/A
CSAFE_PM_SET_TARGETPACETIME	0x06	Byte 0: Pace Time (MSB) Byte 1: Pace Time Byte 2: Pace Time Byte 3: Pace Time (LSB)	N/A
CSAFE_PM_SET_INTERVALIDENTIFIER	0x07	<Not implemented>	
CSAFE_PM_SET_OPERATIONALSTATE	0x08	<Not implemented>	
CSAFE_PM_SET_RACETYPE	0x09	Byte 0: Type	N/A
CSAFE_PM_SET_WARMUPDURATION	0x0A	<Not implemented>	
CSAFE_PM_SET_RACELANESETUP	0x0B	Byte 0: Erg Physical Address Byte 1: Race Lane Number	N/A
CSAFE_PM_SET_RACELANEVERIFY	0x0C	Byte 0: Erg Physical Address Byte 1: Race Lane Number	N/A
CSAFE_PM_SET_RACESTARTPARAMS	0x0D	Byte 0: Start Type (0: Random, 1: Countdown, 2: Random modified) Byte 1: Count Start Count/Race Start State Byte 2: Ready Tick Count (MSB) Byte 3: Ready Tick Count Byte 4: Ready Tick Count Byte 5: Ready Tick Count (LSB) Byte 6: Attention Tick Count/Countdown Ticks Per	N/A

		Number (MSB) Byte 7: Attention Tick Count/Countdown Ticks Per Number Byte 8: Attention Tick Count/Countdown Ticks Per Number Byte 9: Attention Tick Count/Countdown Ticks Per Number (LSB) Byte 10: Row Tick Count (MSB) Byte 11: Row Tick Count Byte 12: Row Tick Count Byte 13: Row Tick Count (LSB)	
CSAFE_PM_SET_ERGSLAVEDISCOVERYREQUEST	0x0E	Byte 0: Starting Erg Slave Address	N/A
CSAFE_PM_SET_BOATNUMBER	0x0F	Byte 0: Boat Number	N/A
CSAFE_PM_SET_ERGNUMBER	0x10	Byte 0: HW address[1] (MSB) Byte 1: HW address Byte 2: HW address Byte 3: HW address (LSB) Byte 4: Erg Number (Logical Address)	N/A
CSAFE_PM_SET_COMMUNICATIONSTATE	0x11	<Not implemented>	
CSAFE_PM_SET_CMDUPLIST	0x12	<Not implemented>	
CSAFE_PM_SET_SCREENSTATE	0x13	Byte 0: Screen Type Byte 1: Screen Value	N/A
CSAFE_PM_CONFIGURE_WORKOUT	0x14	Byte 0: Programming mode (0: Disable, 1: Enable)	N/A
CSAFE_PM_SET_TARGETAVGWATTS	0x15	Byte 0: Avg Watts (MSB) Byte 1: Avg Watts (LSB)	N/A
CSAFE_PM_SET_TARGETCALSPERHR	0x16	Byte 0: Cals/Hr (MSB) Byte 1: Cals/Hr (LSB)	N/A
CSAFE_PM_SET_INTERVALTYPE	0x17	Byte 0:Interval Type (0: Time, 1: Distance, 2: Rest, 3:Time w/ Undefined Rest 4: Distance w/ Undefined Rest, 5: Undefined Rest, 6: Calorie, 7: Calorie w/ Undefined Rest, 8: WattMinute, 9: WattMinute w/ Undefined Rest)	N/A
CSAFE_PM_SET_WORKOUTINTERVALCOUNT	0x18	Byte 0: Interval Count	N/A
CSAFE_PM_SET_DISPLAYUPDATERATE	0x19	Byte 0: Display Update Rate	N/A
CSAFE_PM_SET_AUTHENPASSWORD	0x1A	Byte 0: HW address[1] (MSB) Byte 1: HW address Byte 2: HW address Byte 3: HW address (LSB) Byte 4: Authen PW (MSB) Byte 5: Authen PW	Byte 0: Result

		Byte 6: Authen PW Byte 7: Authen PW Byte 8: Authen PW Byte 9: Authen PW Byte 10: Authen PW Byte 11: Authen PW (LSB)	
CSAFE_PM_SET_TICKTIME	0x1B	Byte 0: Tick Time (MSB) Byte 1: Tick Time Byte 2: Tick Time Byte 3: Tick Time (LSB)	N/A
CSAFE_PM_SET_TICKTIMEOFFSET	0x1C	Byte 0: Tick Time Offset (MSB) Byte 1: Tick Time Offset Byte 2: Tick Time Offset Byte 3: Tick Time Offset (LSB)	N/A
CSAFE_PM_SET_RACEDATASAMPLETICKS	0x1D	Byte 0: Sample Tick (MSB) Byte 1: Sample Tick Byte 2: Sample Tick Byte 3: Sample Tick (LSB)	N/A
CSAFE_PM_SET_RACEOPERATIONTYPE	0x1E	Byte 0: Type	N/A
CSAFE_PM_SET_RACESTATUSDISPLAYTICKS	0x1F	Byte 0:Display Tick (MSB) Byte 1: Display Tick Byte 2: Display Tick Byte 3: Display Tick (LSB)	N/A
CSAFE_PM_SET_RACESTATUSWARNINGTICKS	0x20	Byte 0: Warning Tick MSB) Byte 1: Warning Tick Byte 2: Warning Tick Byte 3: Warning Tick (LSB)	N/A
CSAFE_PM_SET_RACEIDLEMODEPARAMS	0x21	Byte 0: Doze Sec (MSB) Byte 1: Doze Sec (LSB) Byte 2: Sleep Sec (MSB) Byte 3: Sleep Sec (LSB) Byte 4: WIFI Sleep Duration Sec (MSB) Byte 5: WIFI Sleep Duration Sec (LSB) Byte 6: WIFI Holdover Duration Sec (MSB) Byte 7: WIFI Holdover Duration Sec (LSB)	N/A
CSAFE_PM_SET_DATETIME	0x22	Byte 0: Time Hours (1 – 12) Byte 1: Time Minutes (0 – 59) Byte 2: Time Meridiem (0 – AM, 1 – PM) Byte 3: Date Month (1 – 12) Byte 4: Date Day (1 – 31) Byte 5: Date Year (MSB) Byte 6: Date Year (LSB)	N/A
CSAFE_PM_SET_LANGUAGETYPE	0x23	Byte 0: Language Type	N/A
CSAFE_PM_SET_WIFICONFIG	0x24	Byte 0: Config Index Byte 1: WEP Mode	N/A
CSAFE_PM_SET_CPUTICKRATE	0x25	Byte 0: CPU/Tick Rate	N/A
CSAFE_PM_SET_LOGCARDUSER	0x26	Byte 0: Logcard User #	N/A

CSAFE_PM_SET_SCREENERRORMODE	0x27	Byte 0: Mode (disable/enable)	N/A
CSAFE_PM_SET_CABLETEST[2]	0x28	Byte 0: Mode (disable/enable) Byte 1: Dummy Data . . . Byte 79: Dummy Data	N/A
CSAFE_PM_SET_USER_ID	0x29	Byte 0: User Number Byte 1: User ID (MSB) Byte 2: User ID Byte 3: User ID Byte 4: User ID (LSB)	N/A
CSAFE_PM_SET_USER_PROFILE	0x2A	Byte 0: User Number Byte 1: User Weight (MSB) Byte 2: User Weight (LSB) Byte 3: User DOB Day Byte 4: User DOB Month Byte 5: User DOB Year (MSB) Byte 6: User DOB Year (LSB) Byte 7: User Gender	N/A
CSAFE_PM_SET_HRM	0x2B	Byte 0: Device Manufacture ID Byte 1: Device Type Byte 2: Device Num (MSB) Byte 3: Device Num (LSB)	N/A
CSAFE_PM_SET_RACESTARTINGPHYSCALADDRESS	0x2C	Byte 0: Physical Address of First Erg In Race	N/A
CSAFE_PM_SET_HRBELT_INFO	0x2D	Byte 0: User Number Byte 1: Mfg ID Byte 2: Device Type Byte 3: Belt ID (MSB) Byte 4: Belt ID (LSB)	N/A
CSAFE_PM_SET_SENSOR_CHANNEL	0x2F	Byte 0: RF Frequency Byte 1: RF Period Hz (MSB) Byte 2: RF Period Hz (LSB) Byte 3: Datapage Pattern Byte 4: Activity Timeout	N/A

Notes:

1. The hardware address is the unit serial # as stored in MFG EEPROM and accessible by command CSAFE_PM_GET_HW_ADDRESS.
2. This command is used by the PM3/PM4 only and not to be used by the PC

C2 Proprietary Long Set Data Commands

Command Name	Command Identifier	Command Data	Response Data
CSAFE_PM_SET_TEAM_DISTANCE	0x30	<Not implemented>	
CSAFE_PM_SET_TEAM_FINISH_TIME	0x31	<Not implemented>	

CSAFE_PM_SET_RACEPARTICIPANT[2]	0x32	Byte 0: Racer ID (Erg physical address) Byte 1: Racer Name (MSB) Byte 2: Racer Name Byte 3: Racer Name . . . Byte 17: Racer Name (LSB)	N/A
CSAFE_PM_SET_RACESTATUS	0x33	Byte 0: First Racer ID Byte 1: First Racer Position Byte 2: First Racer Delta Distance/Time (MSB) Byte 3: First Racer Delta Distance/Time Byte 4: First Racer Delta Distance/Time Byte 5: Lead Racer Delta Distance/Time (LSB) Byte 6: 2nd Racer ID Byte 7: 2nd Racer Position Byte 8: 2nd Racer Delta Distance/Time (MSB) Byte 9: 2nd Racer Delta Distance /Time Byte 10: 2nd Racer Delta Distance/Time Byte 11: 2nd Racer Delta Distance/Time (LSB) Byte 12: 3rd Racer ID Byte 13: 3rd Racer Position Byte 14: 3rd Racer Delta Distance (MSB) Byte 15: 3rd Racer Delta Distance /Time Byte 16: 3rd Racer Delta Distance/Time Byte 17: 3rd Racer Delta Distance/Time (LSB) Byte 18: 4th Racer ID Byte 19: 4th Racer Position Byte 20: 4th Racer Delta Distance (MSB) Byte 21: 4th Racer Delta Distance/Time Byte 22: 4th Racer Delta Distance/Time Byte 23: 4th Racer Delta Distance/Time (LSB) Byte 24: Team Distance (MSB) Byte 25: Team Distance Byte 26: Team Distance Byte 27: Team Distance	N/A

		(LSB) Byte 28: Mode	
CSAFE_PM_SET_LOGCARD_MEMORY[1]	0x34	Byte 0: Start address (MSB) Byte 1: Start address Byte 2: Start address Byte 3: Start address (LSB) Byte 4: Block length Byte 5: 1st data to be set Byte 6: 2nd data to be set . . . Byte 68: 64nd data to be set	Byte 0: Bytes written
CSAFE_PM_SET_DISPLAYSTRING	0x35	Byte 0: 1st Character Byte 1: 2nd Character Byte 2: 3rd Character . . . Byte 31: 32nd character	N/A
CSAFE_PM_SET_DISPLAYBITMAP	0x36	Byte 0: Bitmap index (MSB) Byte 1: Bitmap index (LSB) Byte 2: Block length Byte 3: Data Index + 0 Byte 4: Data Index + 1 . . . Byte 66: Data Index + 63	Byte 0: Total bitmap bytes (MSB) Byte 1: Total bitmap bytes (LSB)
CSAFE_PM_SET_LOCALRACEPARTICIPANT	0x37	Byte 0: Race Type Byte 1: Race Length (MSB) Byte 2: Race Length Byte 3: Race Length Byte 4: Race Length (LSB) Byte 5: Race Participants Byte 6: Race State Byte 7: Race Lane	N/A
CSAFE_PM_SET_GAMEPARAMS	0x38	Byte 0: Game Type ID Byte 1: Workout Duration Time (MSB) Byte 2: Workout Duration Time Byte 3: Workout Duration Time Byte 4: Workout Duration Time (LSB) Byte 5: Split Duration Time (MSB) Byte 6: Split Duration Time Byte 7: Split Duration Time Byte 8: Split Duration Time (LSB) Byte 9: Target Pace Time (MSB) Byte 10: Target Pace Time Byte 11: Target Pace Time	N/A

		Byte 12: Target Pace Time (LSB) Byte 13: Target Avg Watts (MSB) Byte 14: Target Avg Watts Byte 15: Target Avg Watts Byte 16: Target Avg Watts (LSB) Byte 17: Target Cals Per Hour (MSB) Byte 18: Target Cals Per Hour Byte 19: Target Cals Per Hour Byte 20: Target Cals Per Hour (LSB) Byte 21: Target Stroke Rate	
CSAFE_PM_SET_EXTENDED_HRBELT_INFO	0x39	Byte 0: <unused> Byte 1: HRM mfg id Byte 2: HRM device type Byte 3: HRM belt id (MSB) Byte 4: HRM belt id Byte 5: HRM belt id Byte 6: HRM belt id (LSB)	N/A
CSAFE_PM_SET_EXTENDED_HRM	0x3A	Byte 0: HRM mfg id Byte 1: HRM device type Byte 2: HRM belt id (MSB) Byte 3: HRM belt id Byte 4: HRM belt id Byte 5: HRM belt id (LSB)	N/A
CSAFE_PM_SET_LEDBACKLIGHT	0x3B	Byte 0: State (enable/disable) Byte 1: Intensity (0 – 100%)	N/A
CSAFE_PM_SET_DIAGLOG_RECORD_ARCHIVE	0x3C	Byte 0: Record Type (Enum) Byte 1: Record Index (MSB) Byte 2: Record Index (LSB) (65535 archives all)	N/A
CSAFE_PM_SET_WIRELESS_CHANNEL_CONFIG	0x3D	Byte 0: Wireless channel bit mask (MSB) Byte 1: Wireless channel bit mask Byte 2: Wireless channel bit mask Byte 3: Wireless channel bit mask (LSB)	N/A
CSAFE_PM_SET_RACECONTROLPARAMS	0x3E	Byte 0: Undefined rest to work transition time, 1sec LSB (MSB) Byte 1: Undefined rest to work transition time (LSB) Byte 2: Undefined rest interval, 1sec LSB (MSB) Byte 3: Undefined rest interval, (LSB) Byte 4: Race prompt bitmap	

		display duration, 1sec LSB (MSB) Byte 5: Race prompt bitmap display duration Byte 6: Race prompt bitmap display duration Byte 7: Race prompt bitmap display duration (LSB) Byte 8: Time Cap duration, 1 sec LSB (MSB) Byte 9: Time Cap duration, Byte 10: Time Cap duration, Byte 11: Time Cap duration (LSB)	

Notes:

A maximum block length of 64 bytes can be set. Fewer bytes can be set by specifying the block length accordingly, but a complete 69 bytes of command data will be sent.

The race participant name is a null-terminated string limited to 16 characters.

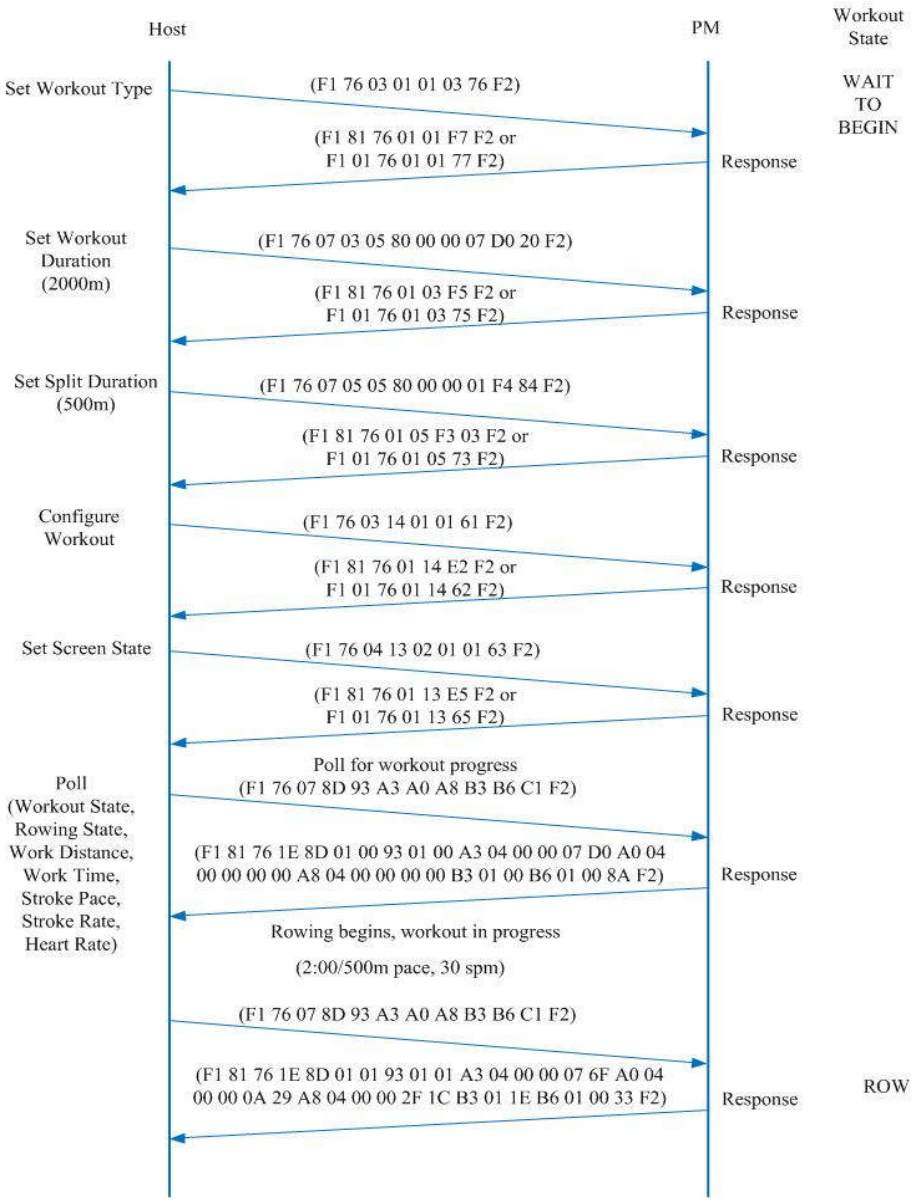
The CSAFE_PM_SET_LOCALRACEPARTICIPANT command is only available for firmware that supports PCless racing.

Setting Up and Performing Workout

The proprietary CSAFE interface does not use the public CSAFE state machine functionality. Generally, the proprietary and public operating modes should not be mixed as the resulting behavior will not be desirable. This example will demonstrate how to configure a workout and then transition to the workout screen, followed by one approach to monitor workout progress. Many different combinations of commands can be used to monitor workout progress depending on what parameters are being monitored and what precision is desired.

Note that the Smart Bluetooth notifications provide a good baseline for collecting general status data, stroke data, split data, and completed workout data.

Figure 12 – Example Proprietary CSAFE PM Workout Setup and Progress Monitoring


Host
PM
Workout State
Set Workout Type
(F1 76 03 01 01 03 76 F2)
WAIT TO BEGIN
(F1 81 76 01 01 F7 F2 or F1 01 76 01 01 77 F2)
Response
Set Workout Duration (2000m)
(F1 76 07 03 05 80 00 00 07 D0 20 F2)
(F1 81 76 01 03 F5 F2 or F1 01 76 01 03 75 F2)
Response
Set Split Duration (500m)
(F1 76 07 05 05 80 00 00 01 F4 84 F2)
(F1 81 76 01 05 F3 03 F2 or F1 01 76 01 05 73 F2)
Response
Configure Workout
(F1 76 03 14 01 01 61 F2)
(F1 81 76 01 14 E2 F2 or F1 01 76 01 14 62 F2)
Response
Set Screen State
(F1 76 04 13 02 01 01 63 F2)
(F1 81 76 01 13 E5 F2 or F1 01 76 01 13 65 F2)
Response
Poll for workout progress
(F1 76 07 8D 93 A3 A0 A8 B3 B6 C1 F2)
Poll (Workout State, Rowing State, Work Distance, Work Time, Stroke Pace, Stroke Rate, Heart Rate)
(F1 81 76 1E 8D 01 00 93 01 00 A3 04 00 00 07 D0 A0 04 00 00 00 00 A8 04 00 00 00 00 B3 01 00 B6 01 00 8A F2)
Response
Rowing begins, workout in progress
(2:00/500m pace, 30 spm)
(F1 76 07 8D 93 A3 A0 A8 B3 B6 C1 F2)
(F1 81 76 1E 8D 01 01 93 01 01 A3 04 00 00 07 6F A0 04 00 00 0A 29 A8 04 00 00 2F 1C B3 01 1E B6 01 00 33 F2)
Response
ROW

Sample Functionality

Public CSAFE Workout Configuration

Fixed Distance

2000m/500m splits, power goal of 200 watts

Command Frame	Description	Response Frame	Description
F1	Standard frame start flag	F1	Standard frame start flag
21	CSAFE_SETHORIZONTAL_CMD	81 or 01	Status
03	Command byte count	1A	CSAFE_SETUSERCFG1_CMD (wrapper)
02	2 Km (horizontal distance LS Byte)	00	Response byte count
00	2 Km (horizontal distance MS Byte)	9B or 1B	Checksum
21	Km, units specifier	F2	Standard frame stop flag
1A	CSAFE_SETUSERCFG1_CMD (wrapper)		
07	Wrapper byte count		
05	CSAFE_PM_SET_SPLITDURATION		
05	Command byte count		
80	WORKOUT_DURATION_IDENTIFIER_DISTANCE		
F4	500m (split duration distance LS Byte)		
01			
00			
00	500m (split duration distance MS Byte)		
34	CSAFE_SETPOWER_CMD		
03	Command byte count		
C8	200w (power goal LS Byte)		
00	200w (power goal MS Byte)		
58	Watt, units specifier		
24	CSAFE_SETPROGRAM_CMD		
02	Command byte count		
00	WORKOUTNUMBER_PROGRAMMED		
00	<unused>		
E8	Checksum		
F2	Standard frame stop flag		

Fixed Time

20:00/4:00 splits, power goal of 100 watts

Command Frame	Description	Response Frame	Description
F1	Standard frame start flag	F1	Standard frame start flag
20	CSAFE_SETTWORK_CMD	81 or 01	Status
03	Command byte count	20	CSAFE_SETTWORK_CMD
00	20:00 (time hours digit)	1A	CSAFE_SETUSERCFG1_CMD (wrapper)
14	20:00 (time minutes digit)	01	Wrapper command byte count
00	20:00 (time seconds digit)	05	CSAFE_PM_SET_SPLITDURATION
1A	CSAFE_SETUSERCFG1_CMD (wrapper)	34	CSAFE_SETPOWER_CMD
07	Wrapper byte count	24	CSAFE_SETPROGRAM_CMD
05	CSAFE_PM_SET_SPLITDURATION	2E	Checksum
05	Command byte count	F2	Standard frame stop flag

00	WORKOUT_DURATION_IDENTIFIER_TIME		
C0	4:00 (split duration time LS Byte, 0.01sec LSB)		
5D			
00			
00	4:00 (split duration time MS Byte, 0.01sec LSB)		
34	CSAFE_SETPOWER_CMD		
03	Command byte count		
64	100w (power goal LS Byte)		
00	100w (power goal MS Byte)		
58	Watt, units specifier		
24	CSAFE_SETPROGRAM_CMD		
02	Command byte count		
00	WORKOUTNUMBER_PROGRAMMED		
00	<unused>		
9A	Checksum		
F2	Standard frame stop flag		

Predefined

Standard List Workouut #3

Command Frame	**Description**	**Response Frame**	**Description**
F1	Standard frame start flag	F1	Standard frame start flag
24	CSAFE_SETPROGRAM_CMD	81 or 01	Status
02	Command byte count	24	CSAFE_SETPROGRAM_CMD
03	WORKOUTNUMBER_DEFAULT_3	24	Checksum
00	<unused>	F2	Standard frame stop flag
25	Checksum		
F2	Standard frame stop flag		

Proprietary CSAFE Workout Configuration

JustRow

Command Frame	**Description**	**Response Frame**	**Description**
F1	Standard frame start flag	F1	Standard frame start flag
76	C2 proprietary wrapper	81 or 01	Status
07	Wrapper command byte count	76	C2 proprietary wrapper
01	CSAFE_PM_SET_WORKOUTTYPE	02	Wrapper command byte count
01	Command byte count	01	CSAFE_PM_SET_WORKOUTTYPE
01	WORKOUTTYPE_JUSTROW_SPLITS	13	CSAFE_PM_SET_SCREENSTATE
13	CSAFE_PM_SET_SCREENSTATE	E7 or 67	Checksum
02	Command byte count	F2	Standard frame stop flag
01	SCREENTYPE_WORKOUT		
01	SCREENVALUEWORKOUT_PREPARETOROWWORKOUT		
61	Checksum		
F2	Standard frame stop flag		

F1 76 07 01 01 01 13 02 01 01 61 F2

Fixed Distance

2000m/500m splits

Command Frame	Description	Response Frame	Description
F1	Standard frame start flag	F1	Standard frame start flag
76	C2 proprietary wrapper	81 or 01	Status
18	Wrapper command byte count	76	C2 proprietary wrapper
01	CSAFE_PM_SET_WORKOUTTYPE	05	Wrapper command byte count
01	Command byte count	01	CSAFE_PM_SET_WORKOUTTYPE
03	WORKOUTTYPE_FIXEDDIST_SPLITS	03	CSAFE_PM_SET_WORKOUTDURATIO N
03	CSAFE_PM_SET_WORKOUTDURATION	05	CSAFE_PM_SET_SPLITDURATION
05	Command byte count	14	CSAFE_PM_CONFIGURE_WORKOUT
80	WORKOUT_DURATION_IDENTIFIER_DISTANCE	13	CSAFE_PM_SET_SCREENSTATE
00	2000m (duration distance MS Byte, 1meter LSB)	F3 or 72	Stuff byte flag (checksum = F2) or checksum
00		02 or F2	Stuff byte value or stop flag
07		F2	Standard frame stop flag or nothing
D0	(duration distance LS Byte)		
05	CSAFE_PM_SET_SPLITDURATION		
05	Command byte count		
80	WORKOUT_DURATION_IDENTIFIER_DISTANCE		
00	400m (split duration distance MS Byte)		
00			
01			
90	(split duration distance LS Byte)		
14	CSAFE_PM_CONFIGURE_WORKOUT		
01	Command byte count		
01	Programming mode enable		
13	CSAFE_PM_SET_SCREENSTATE		
02	Command byte count		
01	SCREENTYPE_WORKOUT		
01	SCREENVALUEWORKOUT_PREPARETOROWWORKOUT		
28	Checksum		
F2	Standard frame stop flag		

F1 76 18 01 01 03 03 05 80 00 00 07 D0 05 05 80 00 00 01 90 14 01 01 13 02 01 01 28 F2

Fixed Time

20:00/4:00 splits

Command Frame	Description	Response Frame	Description
F1	Standard frame start flag	F1	Standard frame start flag
76	C2 proprietary wrapper	81 or 01	Status
18	Wrapper command byte count	76	C2 proprietary wrapper
01	CSAFE_PM_SET_WORKOUTTYPE	05	Wrapper command byte count
01	Command byte count	01	CSAFE_PM_SET_WORKOUTTYPE
05	WORKOUTTYPE_FIXEDTIME_SPLITS	03	CSAFE_PM_SET_WORKOUTDURATIO N
03	CSAFE_PM_SET_WORKOUTDURATION	05	CSAFE_PM_SET_SPLITDURATION
05	Command byte count	14	CSAFE_PM_CONFIGURE_WORKOUT
00	WORKOUT_DURATION_IDENTIFIER_TIME	13	CSAFE_PM_SET_SCREENSTATE
00	20:00 (duration time MS Byte, 0.01sec LSB)	F3 or 72	Stuff byte flag (checksum = F2) or checksum

01		02 or F2	Stuff byte value or stop flag
D4		F2	Standard frame stop flag or nothing
C0	(duration time LS Byte)		
05	CSAFE_PM_SET_SPLITDURATION		
05	Command byte count		
00	WORKOUT_DURATION_IDENTIFIER_TIME		
00	400m (split duration time MS Byte, 0.01 sec LSB)		
00			
5D			
C0	(split duration time LS Byte)		
14	CSAFE_PM_CONFIGURE_WORKOUT		
01	Command byte count		
01	Programming mode enable		
13	CSAFE_PM_SET_SCREENSTATE		
02	Command byte count		
01	SCREENTYPE_WORKOUT		
01	SCREENVALUEWORKOUT_PREPARETOROWWORKOUT		
E0	Checksum		
F2	Standard frame stop flag		

F1 76 18 01 01 05 03 05 00 00 01 D4 C0 05 05 00 00 00 5D C0 14 01 01 13 02 01 01 E0 F2

Fixed Calories

100 Cals/20 Cal splits

Command Frame	**Description**	**Response Frame**	**Description**
F1	Standard frame start flag	F1	Standard frame start flag
76	C2 proprietary wrapper	81 or 01	Status
18	Wrapper command byte count	76	C2 proprietary wrapper
01	CSAFE_PM_SET_WORKOUTTYPE	05	Wrapper command byte count
01	Command byte count	01	CSAFE_PM_SET_WORKOUTTYPE
0A	WORKOUTTYPE_FIXEDCALORIE_SPLITS	03	CSAFE_PM_SET_WORKOUTDURATIO N
03	CSAFE_PM_SET_WORKOUTDURATION	05	CSAFE_PM_SET_SPLITDURATION
05	Command byte count	14	CSAFE_PM_CONFIGURE_WORKOUT
C0	WORKOUT_DURATION_IDENTIFIER_CALORIES	13	CSAFE_PM_SET_SCREENSTATE
00	100 Cals (duration calories MS Byte, 1Cal LSB)	F3	Stuff byte flag (checksum = F2)
00		02	Stuff byte value
00		F2	Standard frame stop flag
64	(duration calories LS Byte)		
05	CSAFE_PM_SET_SPLITDURATION		
05	Command byte count		
C0	WORKOUT_DURATION_IDENTIFIER_CALORIES		
00	20 Cals (split duration calories MS Byte, 1Cal LSB)		
00			
00			
14	(split duration time LS Byte)		
14	CSAFE_PM_CONFIGURE_WORKOUT		
01	Command byte count		
01	Programming mode enable		
13	CSAFE_PM_SET_SCREENSTATE		
02	Command byte count		
01	SCREENTYPE_WORKOUT		
01	SCREENVALUEWORKOUT_PREPARETOROWWORKOUT		
17	Checksum		
F2	Standard frame stop flag		

F1 76 18 01 01 0A 03 05 C0 00 00 00 64 05 05 0 00 00 00 14 14 01 01 13 02 01 01 17 F2

Fixed Distance Interval

500m/:30 rest

Command Frame	Description	Response Frame	Description
F1	Standard frame start flag	F1	Standard frame start flag
76	C2 proprietary wrapper	81 or 01	Status
15	Wrapper command byte count	76	C2 proprietary wrapper
01	CSAFE_PM_SET_WORKOUTTYPE	05	Wrapper command byte count
01	Command byte count	01	CSAFE_PM_SET_WORKOUTTYPE
07	WORKOUTTYPE_FIXEDDIST_INTERVAL	03	CSAFE_PM_SET_WORKOUTDURATIO N
03	CSAFE_PM_SET_WORKOUTDURATION	04	CSAFE_PM_SET_RESTDURATION
05	Command byte count	14	CSAFE_PM_CONFIGURE_WORKOUT
80	WORKOUT_DURATION_IDENTIFIER_DISTANCE	13	CSAFE_PM_SET_SCREENSTATE
00	500m (duration distance MS Byte, 1meter LSB)	F3 or 73	Stuff byte flag (checksum = F3) or checksum
00		03 or F2	Stuff byte value or stop flag
01		F2	Standard frame stop flag or nothing
F4	(duration distance LS Byte)		
04	CSAFE_PM_SET_RESTDURATION		
02	Command byte count		
00	:30 (rest duration time MSB, 1sec LSB)		
1E	(rest duration time LS Byte)		
14	CSAFE_PM_CONFIGURE_WORKOUT		
01	Command byte count		
01	Programming mode enable		
13	CSAFE_PM_SET_SCREENSTATE		
02	Command byte count		
01	SCREENTYPE_WORKOUT		
01	SCREENVALUEWORKOUT_PREPARETOROWWORKOUT		
0A	Checksum		
F2	Standard frame stop flag		

F1 76 15 01 01 07 03 05 80 00 00 01 F4 04 02 00 1E 14 01 01 13 02 01 01 0A F2

Fixed Time Interval

2:00/:30 rest

Command Frame	Description	Response Frame	Description
F1	Standard frame start flag	F1	Standard frame start flag
76	C2 proprietary wrapper	81 or 01	Status
15	Wrapper command byte count	76	C2 proprietary wrapper
01	CSAFE_PM_SET_WORKOUTTYPE	05	Wrapper command byte count
01	Command byte count	01	CSAFE_PM_SET_WORKOUTTYPE
06	WORKOUTTYPE_FIXEDTIME_INTERVAL	03	CSAFE_PM_SET_WORKOUTDURATIO N
03	CSAFE_PM_SET_WORKOUTDURATION	04	CSAFE_PM_SET_RESTDURATION
05	Command byte count	14	CSAFE_PM_CONFIGURE_WORKOUT
00	WORKOUT_DURATION_IDENTIFIER_TIME	13	CSAFE_PM_SET_SCREENSTATE
00	2:00 (duration time MS Byte, .01sec LSB)	F3 or 73	Stuff byte flag (checksum = F3) or

			checksum
00		03 or F2	Stuff byte value or stop flag
2E		F2	Standard frame stop flag or nothing
E0	(duration time LS Byte)		
04	CSAFE_PM_SET_RESTDURATION		
02	Command byte count		
00	:30 (rest duration time MSB, 1sec LSB)		
1E	(rest duration time LS Byte)		
14	CSAFE_PM_CONFIGURE_WORKOUT		
01	Command byte count		
01	Programming mode enable		
13	CSAFE_PM_SET_SCREENSTATE		
02	Command byte count		
01	SCREENTYPE_WORKOUT		
01	SCREENVALUEWORKOUT_PREPARETOROWWORKOUT		
0A	Checksum		
F2	Standard frame stop flag		

F1 76 15 01 01 06 03 05 00 00 00 2E E0 04 02 00 1E 14 01 01 13 02 01 01 0A F2

Fixed Calorie Interval

25c/1:00 rest

Command Frame	**Description**	**Response Frame**	**Description**
F1	Standard frame start flag	F1	Standard frame start flag
76	C2 proprietary wrapper	81 or 01	Status
15	Wrapper command byte count	76	C2 proprietary wrapper
01	CSAFE_PM_SET_WORKOUTTYPE	05	Wrapper command byte count
01	Command byte count	01	CSAFE_PM_SET_WORKOUTTYPE
0C	WORKOUTTYPE_FIXEDCALS_INTERVAL	03	CSAFE_PM_SET_WORKOUTDURATIO N
03	CSAFE_PM_SET_WORKOUTDURATION	04	CSAFE_PM_SET_RESTDURATION
05	Command byte count	14	CSAFE_PM_CONFIGURE_WORKOUT
40	WORKOUT_DURATION_IDENTIFIER_CALORIES	13	CSAFE_PM_SET_SCREENSTATE
00	25c (duration cals MS Byte, 1cal LSB)	F3 or 73	Stuff byte flag (checksum = F3) or checksum
00		03 or F2	Stuff byte value or stop flag
00		F2	Standard frame stop flag or nothing
19	(duration cals LS Byte)		
04	CSAFE_PM_SET_RESTDURATION		
02	Command byte count		
00	1:00 (rest duration time MSB, 1sec LSB)		
3C	(rest duration time LS Byte)		
14	CSAFE_PM_CONFIGURE_WORKOUT		
01	Command byte count		
01	Programming mode enable		
13	CSAFE_PM_SET_SCREENSTATE		
02	Command byte count		
01	SCREENTYPE_WORKOUT		
01	SCREENVALUEWORKOUT_PREPARETOROWWORKOUT		
0A	Checksum		
F2	Standard frame stop flag		

F1 76 15 01 01 0C 03 05 40 00 00 00 19 04 02 00 0C 14 01 01 13 02 01 01 3F F2

Variable Interval

v500m/1:00r…4

Interval 1: 500m/1:00r, target pace of 1:40
Interval 2: 3:00/0:00r, target pace of 1:40
Interval 3: 1000m/0:00r, target pace of 1:40
Interval 4: 5:00/2:00r , target pace of 1:40

Command Frame	Description	Response Frame	Description
F1	Standard frame start flag	F1	Standard frame start flag
76	C2 proprietary wrapper	81 or 01	Status
70	Wrapper command byte count	76	C2 proprietary wrapper
18	CSAFE_PM_WORKOUTINTERVALCOUNT	1A	Wrapper command byte count
01	Command byte count	18	CSAFE_PM_WORKOUTINTERVALCOUNT
00	Interval #1	01	CSAFE_PM_SET_WORKOUTTYPE
01	CSAFE_PM_SET_WORKOUTTYPE	17	CSAFE_PM_SET_INTERVALTYPE
01	Command byte count	03	CSAFE_PM_SET_WORKOUTDURATION
08	WORKOUTTYPE_VARIABLE_INTERVAL	04	CSAFE_PM_SET_RESTDURATION
17	CSAFE_PM_SET_INTERVALTYPE	06	CSAFE_PM_SET_TARGETPACETIME
01	Command byte count	14	CSAFE_PM_CONFIGURE_WORKOUT
01	INTERVALTYPE_DIST	18	CSAFE_PM_WORKOUTINTERVALCOUNT
03	CSAFE_PM_SET_WORKOUTDURATION	17	CSAFE_PM_SET_INTERVALTYPE
05	Command byte count	03	CSAFE_PM_SET_WORKOUTDURATION
80	WORKOUT_DURATION_IDENTIFIER_DIST	04	CSAFE_PM_SET_RESTDURATION
00	500m (duration distance MS Byte, 1meter LSB)	06	CSAFE_PM_SET_TARGETPACETIME
00		14	CSAFE_PM_CONFIGURE_WORKOUT
01		18	CSAFE_PM_WORKOUTINTERVALCOUNT
F4	(duration distance LS Byte)	17	CSAFE_PM_SET_INTERVALTYPE
04	CSAFE_PM_SET_RESTDURATION	03	CSAFE_PM_SET_WORKOUTDURATION
02	Command byte count	04	CSAFE_PM_SET_RESTDURATION
00	1:00 (rest duration time MSB, 1sec LSB)	06	CSAFE_PM_SET_TARGETPACETIME
3C	(rest duration time LS Byte)	14	CSAFE_PM_CONFIGURE_WORKOUT
06	CSAFE_PM_SET_TARGETPACETIME	18	CSAFE_PM_WORKOUTINTERVALCOUNT
04	Command byte count	17	CSAFE_PM_SET_INTERVALTYPE
00	1:40 (pace time MS Byte, .01sec LSB)	03	CSAFE_PM_SET_WORKOUTDURATION
00		04	CSAFE_PM_SET_RESTDURATION
27		06	CSAFE_PM_SET_TARGETPACETIME
10	(pace time LS Byte)	14	CSAFE_PM_CONFIGURE_WORKOUT
14	CSAFE_PM_CONFIGURE_WORKOUT	13	CSAFE_PM_SET_SCREENSTATE
01	Command byte count	FF or 7F	Checksum
01	Programming mode enable	F2	Standard frame stop flag
18	CSAFE_PM_WORKOUTINTERVALCOUNT		
01	Command byte count		
01	Interval #2		
17	CSAFE_PM_SET_INTERVALTYPE		
01	Command byte count		
00	INTERVALTYPE_TIME		
03	CSAFE_PM_SET_WORKOUTDURATION		
05	Command byte count		
00	WORKOUT_DURATION_IDENTIFIER_TIME		
00	3:00 (duration time MS Byte, .01sec LSB)		
00			

46			
50	(duration time LS Byte)		
04	CSAFE_PM_SET_RESTDURATION		
02	Command byte count		
00	:00 (rest duration time MSB, 1sec LSB)		
00	(rest duration time LS Byte)		
06	CSAFE_PM_SET_TARGETPACETIME		
04	Command byte count		
00	1:40 (pace time MS Byte, .01sec LSB)		
00			
27			
10	(pace time LS Byte)		
14	CSAFE_PM_CONFIGURE_WORKOUT		
01	Command byte count		
01	Programming mode enable		
18	CSAFE_PM_WORKOUTINTERVALCOUNT		
01	Command byte count		
02	Interval #3		
17	CSAFE_PM_SET_INTERVALTYPE		
01	Command byte count		
01	INTERVALTYPE_DIST		
03	CSAFE_PM_SET_WORKOUTDURATION		
05	Command byte count		
80	WORKOUT_DURATION_IDENTIFIER_DIST		
00	1000m (duration distance MS Byte, 1meter LSB)		
00			
03			
E8	(duration distance LS Byte)		
04	CSAFE_PM_SET_RESTDURATION		
02	Command byte count		
00	:00 (rest duration time MSB, 1sec LSB)		
00	(rest duration time LS Byte)		
06	CSAFE_PM_SET_TARGETPACETIME		
04	Command byte count		
00	1:40 (pace time MS Byte, .01sec LSB)		
00			
27			
10	(pace time LS Byte)		
14	CSAFE_PM_CONFIGURE_WORKOUT		
01	Command byte count		
01	Programming mode enable		
18	CSAFE_PM_WORKOUTINTERVALCOUNT		
01	Command byte count		
03	Interval #4		
17	CSAFE_PM_SET_INTERVALTYPE		
01	Command byte count		
00	INTERVALTYPE_TIME		
03	CSAFE_PM_SET_WORKOUTDURATION		
05	Command byte count		
00	WORKOUT_DURATION_IDENTIFIER_TIME		
00	5:00 (duration time MS Byte, .01sec LSB)		
00			
75			

30	(duration time LS Byte)		
04	CSAFE_PM_SET_RESTDURATION		
02	Command byte count		
00	2:00 (rest duration time MSB, 1sec LSB)		
78	(rest duration time LS Byte)		
06	CSAFE_PM_SET_TARGETPACETIME		
04	Command byte count		
00	1:40 (pace time MS Byte, .01sec LSB)		
00			
27			
10	(pace time LS Byte)		
14	CSAFE_PM_CONFIGURE_WORKOUT		
01	Command byte count		
01	Programming mode enable		
13	CSAFE_PM_SET_SCREENSTATE		
02	Command byte count		
01	SCREENTYPE_WORKOUT		
01	SCREENVALUEWORKOUT_PREPARETOROWWORKOUT		
09	Checksum		
F2	Standard frame stop flag		

F1 76 6F 18 01 00 01 01 08 17 01 01 03 05 80 00 00 01 F4 04 02 00 3C 06 04 00 00 27 10 14 01 01 18 01 01 17 01 00 03 05 00 00 00 46 50 04 02 00 00 06 04 00 00 27 10 14 01 01 18 01 02 17 01 01 03 05 80 00 00 03 E8 04 02 00 00 06 04 00 00 27 10 14 01 01 18 01 03 17 01 00 03 05 00 00 00 75 30 04 02 00 78 06 04 00 00 27 10 14 01 01 13 02 01 01 09 F2

Variable Interval Undefined Rest

The additional configuration command setting SplitDurationDistance to 0 is necessary so that "Biathlon" workout specific logic is not triggered. A Biathlon workout is a form of variable interval workout with undefined rest that uses a non-zero SplitDurationDistance to assess a penalty distance to force the user to perform extra work. This is a special case workout, and in order to "fit it into the workout paradigm" it was necessary to employ the SplitDurationDistance.

Setting the SplitDurationDistance to 0 is necessary when at least one undefined rest interval interval is configured in a variable interval workout.

v100m…2

Interval 1: 100m, target pace of 2:10
Interval 2: 2:00, target pace of 2:10

Command Frame	**Description**	**Response Frame**	**Description**
F1	Standard frame start flag	F1	Standard frame start flag
76	C2 proprietary wrapper	81 or 01	Status
45	Wrapper command byte count	76	C2 proprietary wrapper
18	CSAFE_PM_WORKOUTINTERVALCOUNT	10	Wrapper command byte count
01	Command byte count	18	CSAFE_PM_WORKOUTINTERVALCOUNT
00	Interval #1	01	CSAFE_PM_SET_WORKOUTTYPE
01	CSAFE_PM_SET_WORKOUTTYPE	17	CSAFE_PM_SET_INTERVALTYPE
01	Command byte count	03	CSAFE_PM_SET_WORKOUTDURATION
08	WORKOUTTYPE_VARIABLE_INTERVAL	04	CSAFE_PM_SET_RESTDURATION
17	CSAFE_PM_SET_INTERVALTYPE	06	CSAFE_PM_SET_TARGETPACETIME

01	Command byte count	14	CSAFE_PM_CONFIGURE_WORKOUT
04	INTERVALTYPE_DISTANCERESTUNDEFINED	18	CSAFE_PM_WORKOUTINTERVALCOUNT
03	CSAFE_PM_SET_WORKOUTDURATION	17	CSAFE_PM_SET_INTERVALTYPE
05	Command byte count	03	CSAFE_PM_SET_WORKOUTDURATION
80	WORKOUT_DURATION_IDENTIFIER_DIST	04	CSAFE_PM_SET_RESTDURATION
00	100m (duration distance MS Byte, 1meter LSB)	06	CSAFE_PM_SET_TARGETPACETIME
00		14	CSAFE_PM_CONFIGURE_WORKOUT
00		01	CSAFE_PM_SET_WORKOUTTYPE
64	(duration distance LS Byte)	05	CSAFE_PM_SET_SPLITDURATION
04	CSAFE_PM_SET_RESTDURATION	13	CSAFE_PM_SET_SCREENSTATE
02	Command byte count	F3 or 71	Stuff byte flag (checksum = F1) or checksum
		01 or F2	Stuff byte value or stop flag
00	:00 (rest duration time MSB, 1sec LSB)	F2	Standard frame stop flag or nothing
00	(rest duration time LS Byte)		
06	CSAFE_PM_SET_TARGETPACETIME		
04	Command byte count		
00	2:10 (pace time MS Byte, .01sec LSB)		
00			
32			
C8	(pace time LS Byte)		
14	CSAFE_PM_CONFIGURE_WORKOUT		
01	Command byte count		
01	Programming mode enable		
18	CSAFE_PM_WORKOUTINTERVALCOUNT		
01	Command byte count		
01	Interval #2		
17	CSAFE_PM_SET_INTERVALTYPE		
01	Command byte count		
03	INTERVALTYPE_TIMERESTUNDEFINED		
03	CSAFE_PM_SET_WORKOUTDURATION		
05	Command byte count		
00	WORKOUT_DURATION_IDENTIFIER_TIME		
00	2:00 (duration time MS Byte, .01sec LSB)		
00			
2E			
E0	(duration time LS Byte)		
04	CSAFE_PM_SET_RESTDURATION		
02	Command byte count		
00	:00 (rest duration time MSB, 1sec LSB)		
00	(rest duration time LS Byte)		
06	CSAFE_PM_SET_TARGETPACETIME		
04	Command byte count		
00	2:10 (pace time MS Byte, .01sec LSB)		
00			
32			
C8	(pace time LS Byte)		
14	CSAFE_PM_CONFIGURE_WORKOUT		
01	Command byte count		
01	Programming mode enable		
01	CSAFE_PM_SET_WORKOUTTYPE		
01	Command byte count		
09	WORKOUTTYPE_VARIABLE_UNDEFINEDREST_INTERVAL		
05	CSAFE_PM_SET_SPLITDURATION		
05	Command byte count		
80	WORKOUT_DURATION_IDENTIFIER_DISTANCE		
00	0m (split duration distance MS Byte)		
00			
00			

00	(split duration distance LS Byte)		
13	CSAFE_PM_SET_SCREENSTATE		
02	Command byte count		
01	SCREENTYPE_WORKOUT		
01	SCREENVALUEWORKOUT_PREPARETOROWWORKOUT		
46	Checksum		
F2	Standard frame stop flag		

F1 76 45 18 01 00 01 01 08 17 01 04 03 05 80 00 00 00 64 04 02 00 00 06 04 00 00 32 C8 14 01 01 18 01 01 17 01 03 03 05 00 00 00 2E E0 04 02 00 00 06 04 00 00 32 C8 14 01 01 01 01 09 05 05 80 00 00 00 00 13 02 01 01 8F F2

Fixed Interval Undefined Rest

All fixed interval workouts using undefined rest should be programmed as variable interval workouts with undefined rest up to a maximum of 50 intervals. When terminated the fixed interval workouts will be logged with only the intervals completed.

CSAFE Miscellaneous

Terminate Workout

Command Frame	Description	Response Frame	Description
F1	Standard frame start flag	F1	Standard frame start flag
76	C2 proprietary wrapper	81 or 01	Status
04	Wrapper command byte count	76	C2 proprietary wrapper
13	CSAFE_PM_SET_SCREENSTATE	01	Wrapper command byte count
02	Command byte count	13	CSAFE_PM_SET_SCREENSTATE
01	SCREENTYPE_WORKOUT	E5 or 65	Checksum
02	SCREENVALUEWORKOUT_TERMINTATEWORKOUT	F2	Standard frame stop flag
62	Checksum		
F2	Standard frame stop flag		

F1 76 04 13 02 01 02 62 F2

Appendix A

Enumerated Values

Operational State

```
typedef enum {

  OPERATIONALSTATE_RESET,                 /**< Reset state (0). */
  OPERATIONALSTATE_READY,                 /**< Ready state (1). */
  OPERATIONALSTATE_WORKOUT,               /**< Workout state (2). */
  OPERATIONALSTATE_WARMUP,                /**< Warm-up state (3). */
  OPERATIONALSTATE_RACE,                  /**< Race state (4). */
  OPERATIONALSTATE_POWEROFF,              /**< Power-off state (5). */
  OPERATIONALSTATE_PAUSE,                 /**< Pause state (6). */
  OPERATIONALSTATE_INVOKEBOOTLOADER,   /**< Invoke boot loader state (7). */
  OPERATIONALSTATE_POWEROFF_SHIP,   /**< Power-off ship state (8). */
  OPERATIONALSTATE_IDLE_CHARGE,       /**< Idle charge state (9). */
  OPERATIONALSTATE_IDLE,                  /**< Idle state (10). */
  OPERATIONALSTATE_MFGTEST,               /**< Manufacturing test state (11). */
  OPERATIONALSTATE_FWUPDATE,              /**< Firmware update state (12). */
  OPERATIONALSTATE_DRAGFACTOR,        /**< Drag factor state (13). */
  OPERATIONALSTATE_DFCALIBRATION = 100 /**< Drag factor calibration state (100). */
} OBJ_OPERATIONALSTATE_T;
```

Erg Machine Type

```
typedef enum {
  ERGMACHINE_TYPE_STATIC_D,                          /**< Model D, static type (0). */
  ERGMACHINE_TYPE_STATIC_C,                          /**< Model C, static type (1). */
  ERGMACHINE_TYPE_STATIC_A,                          /**< Model A, static type (2). */
  ERGMACHINE_TYPE_STATIC_B,                          /**< Model B, static type (3). */
  ERGMACHINE_TYPE_STATIC_E = 5,                      /**< Model E, static type (5). */
  ERGMACHINE_TYPE_STATIC_SIMULATOR = 7,              /**< Rower simulator type (7). */
  ERGMACHINE_TYPE_STATIC_DYNAMIC = 8,                /**< Dynamic, static type (8). */
  ERGMACHINE_TYPE_SLIDES_A = 16,                     /**< Model A, slides type (16). */
  ERGMACHINE_TYPE_SLIDES_B,                          /**< Model B, slides type (17). */
  ERGMACHINE_TYPE_SLIDES_C,                          /**< Model C, slides type (18). */
  ERGMACHINE_TYPE_SLIDES_D,                          /**< Model D, slides type (19). */
  ERGMACHINE_TYPE_SLIDES_E,                          /**< Model E, slides type (20). */
  ERGMACHINE_TYPE_LINKED_DYNAMIC = 32,               /**< Dynamic, linked type (32). */
  ERGMACHINE_TYPE_STATIC_DYNO = 64,                  /**< Dynomometer, static type (32). */
  ERGMACHINE_TYPE_STATIC_SKI = 128,                  /**< Ski Erg, static type (128). */
  ERGMACHINE_TYPE_STATIC_SKI_SIMULATOR = 143,        /**< Ski simulator type (143). */
  ERGMACHINE_TYPE_BIKE = 192,                        /**< Bike, no arms type (192). */
  ERGMACHINE_TYPE_BIKE_ARMS,                         /**< Bike, arms type (193). */
  ERGMACHINE_TYPE_BIKE_NOARMS,                       /**< Bike, no arms type (194). */
  ERGMACHINE_TYPE_BIKE_SIMULATOR = 207,              /**< Bike simulator type (207). */
  ERGMACHINE_TYPE_MULTIERG_ROW = 224,                /**< Multi-erg row type (224). */
  ERGMACHINE_TYPE_MULTIERG_SKI,                      /**< Multi-erg ski type (225). */
  ERGMACHINE_TYPE_MULTIERG_BIKE,                     /**< Multi-erg bike type (226). */
  ERGMACHINE_TYPE_NUM,                               /**< Number of machine types (227). */
} OBJ_ERGMACHINETYPE_T;
```

Workout Type

```
typedef enum {
   WORKOUTTYPE_JUSTROW_NOSPLITS,                     /**< JustRow, no splits (0). */
```

```
   WORKOUTTYPE_JUSTROW_SPLITS,                          /**< JustRow, splits (1). */
   WORKOUTTYPE_FIXEDDIST_NOSPLITS,                      /**< Fixed distance, no splits (2). */
   WORKOUTTYPE_FIXEDDIST_SPLITS,                        /**< Fixed distance, splits (3). */
   WORKOUTTYPE_FIXEDTIME_NOSPLITS,                      /**< Fixed time, no splits (4). */
   WORKOUTTYPE_FIXEDTIME_SPLITS,                        /**< Fixed time, splits (5). */
   WORKOUTTYPE_FIXEDTIME_INTERVAL,                      /**< Fixed time interval (6). */
   WORKOUTTYPE_FIXEDDIST_INTERVAL,                      /**< Fixed distance interval (7). */
   WORKOUTTYPE_VARIABLE_INTERVAL,                       /**< Variable interval (8). */
   WORKOUTTYPE_VARIABLE_UNDEFINEDREST_INTERVAL,         /**< Variable interval, undefined rest (9).
   WORKOUTTYPE_FIXEDCALORIE_SPLITS,                     /**< Fixed calorie, splits (10). */
   WORKOUTTYPE_FIXEDWATTMINUTE_SPLITS,                  /**< Fixed watt-minute, splits (11). */
   WORKOUTTYPE_FIXEDCALS_INTERVAL,                      /**< Fixed calorie interval (12). */
   WORKOUTTYPE_NUM                                      /**< Number of workout types (13). */
} OBJ_WORKOUTTYPE_T;
```

Interval Type

```
typedef enum {
   INTERVALTYPE_TIME,                                   /**< Time interval type (0). */
   INTERVALTYPE_DIST,                                   /**< Distance interval type (1). */
   INTERVALTYPE_REST,                                   /**< Rest interval type (2). */
   INTERVALTYPE_TIMERESTUNDEFINED,                      /**< Time undefined rest interval type (3). */
   INTERVALTYPE_DISTANCERESTUNDEFINED,                  /**< Distance undefined rest interval type (4). */
   INTERVALTYPE_RESTUNDEFINED,                          /**< Undefined rest interval type (5). */
   INTERVALTYPE_CALORIE,                                /**< Calorie interval type (6). */
   INTERVALTYPE_CALORIERESTUNDEFINED ,                  /**< Calorie undefined rest interval type (7). */
   INTERVALTYPE_WATTMINUTE,                             /**< Watt-minute interval type (8). */
   INTERVALTYPE_WATTMINUTERESTUNDEFINED,                /**< Watt-minute undefined rest interval type (9). */
   INTERVALTYPE_NONE = 255                              /**< No interval type (255 ). */
} OBJ_INTERVALTYPE_T;
```

Workout State

```
typedef enum {
   WORKOUTSTATE_WAITTOBEGIN,                            /**< Wait to begin state (0). */
   WORKOUTSTATE_WORKOUTROW,                             /**< Workout row state (1). */
   WORKOUTSTATE_COUNTDOWNPAUSE,                         /**< Countdown pause state (2). */
   WORKOUTSTATE_INTERVALREST,                           /**< Interval rest state (3). */
   WORKOUTSTATE_INTERVALWORKTIME,                       /**< Interval work time state (4). */
   WORKOUTSTATE_INTERVALWORKDISTANCE,                   /**< Interval work distance  state (5). */
   WORKOUTSTATE_INTERVALRESTENDTOWORKTIME,              /**< Interval rest end to work time state (6). */
   WORKOUTSTATE_INTERVALRESTENDTOWORKDISTANCE,          /**< Interval rest end to work distance state (7). */
   WORKOUTSTATE_INTERVALWORKTIMETOREST,                 /**< Interval work time to rest state (8). */
   WORKOUTSTATE_INTERVALWORKDISTANCETOREST,             /**< Interval work distance to rest state (9). */
   WORKOUTSTATE_WORKOUTEND,                             /**< Workout end state (10). */
   WORKOUTSTATE_TERMINATE,                              /**< Workout terminate state (11). */
   WORKOUTSTATE_WORKOUTLOGGED,                          /**< Workout logged state (12). */
   WORKOUTSTATE_REARM,                                  /**< Workout rearm state (13). */
} OBJ_WORKOUTSTATE_T;
```

Rowing State

```
typedef enum {
  ROWINGSTATE_INACTIVE,                 /**< Inactive (0). */
  ROWINGSTATE_ACTIVE,                   /**< Active (1). */
} OBJ_ROWINGSTATE_T;
```

Stroke State

```
typedef enum {
  STROKESTATE_WAITING_FOR_WHEEL_TO_REACH_MIN_SPEED_STATE,   /**< FW to reach min speed state (0). */
  STROKESTATE_WAITING_FOR_WHEEL_TO_ACCELERATE_STATE,        /**< FW to accelerate state (1). */
  STROKESTATE_DRIVING_STATE,                                /**< Driving state (2). */
```

```
 STROKESTATE_DWELLING_AFTER_DRIVE_STATE,        /**< Dwelling after drive state (3). */
 STROKESTATE_RECOVERY_STATE                     /**< Recovery state (4). */
} OBJ_STROKESTATE_T;
```

Workout Duration Type

```
enum DurationTypes {
  WORKOUT_DURATION_IDENTIFIER_TIME = 0,
  WORKOUT_DURATION_IDENTIFIER_CALORIES = 0x40,
  WORKOUT_DURATION_IDENTIFIER_DISTANCE  = 0x80,
  WORKOUT_DURATION_IDENTIFIER_WATTMIN = 0xC0
};
```

Display Units Type

```
typedef enum {
   DISPLAYUNITS_TIMEMETERS,          /**< Time/meters display units (0). */
   DISPLAYUNITS_PACE,                /**< Pace display units (1). */
   DISPLAYUNITS_WATTS,               /**< Watts display units (2). */
   DISPLAYUNITS_CALORICBURNRATE,     /**< Caloric burn rate display units (3). */
   DISPLAYUNITS_CALORIES             /**< Calorie display units (4). */
} OBJ_DISPLAYUNITS_T;
```

Display Format Type

```
typedef enum {
   DISPLAYTYPE_STANDARD,             /**< Standard display type (0). */
   DISPLAYTYPE_FORCEVELOCITY,        /**< Force curve display type (1). */
   DISPLAYTYPE_PACEBOAT,             /**< Pace boats display type (2). */
   DISPLAYTYPE_PERSTROKE,            /**< Store rate/heart rate display type (3). */
   DISPLAYTYPE_SIMPLE,               /**< Large format display type (4). */
   DISPLAYTYPE_TARGET                /**< Target display type (5). */
} OBJ_DISPLAYTYPE_T;
```

Workout Number

```
typedef enum {
   WORKOUTNUMBER_PROGRAMMED,         /**< Programmed (0). */
   WORKOUTNUMBER_DEFAULT_1,          /**< Standard list 1 (1). */
   WORKOUTNUMBER_DEFAULT_2,          /**< Standard list 2 (2). */
   WORKOUTNUMBER_DEFAULT_3,          /**< Standard list 3 (3). */
   WORKOUTNUMBER_DEFAULT_4,          /**< Standard list 4 (4). */
   WORKOUTNUMBER_DEFAULT_5,          /**< Standard list 5 (5). */
   WORKOUTNUMBER_CUSTOM_1,           /**< Custom list 1 (6). */
   WORKOUTNUMBER_CUSTOM_2,           /**< Custom list 2 (7). */
   WORKOUTNUMBER_CUSTOM_3,           /**< Custom list 3 (8). */
   WORKOUTNUMBER_CUSTOM_4,           /**< Custom list 4 (9). */
   WORKOUTNUMBER_CUSTOM_5,           /**< Custom list 5 (10). */
   WORKOUTNUMBER_MSD_1,              /**< Favorite list 1 (11). */
   WORKOUTNUMBER_MSD_2,              /**< Favorite list 2 (12). */
   WORKOUTNUMBER_MSD_3,              /**< Favorite list 3 (13). */
   WORKOUTNUMBER_MSD_4,              /**< Favorite list 4 (14). */
   WORKOUTNUMBER_MSD_5,              /**< Favorite list 5 (15). */
   WORKOUTNUMBER_NUM                 /**< Number of workouts (16). */
} OBJ_WORKOUTNUMBER_T;
```

Workout Programming Mode

```
typedef enum {
  WORKOUTPROGRAMMINMODE_DISABLE,     /**< Disable (0). */
```

```
  WORKOUTPROGRAMMINMODE_ENABLE,      /**< Enable (1). */
} OBJ_WORKOUTPROGRAMMINGMODE_T;
```

Stroke Rate State

```
typedef enum {
   STROKERATESTATE_IDLE,                  /**< Idle state (0). */
   STROKERATESTATE_STEADY,                /**< Steady state (1). */
   STROKERATESTATE_INCREASING,            /**< Increasing state (2). */
   STROKERATESTATE_DECREASING             /**< Decreasing state (3). */
} OBJ_STROKERATESTATE_T;
```

Start Type

```
typedef enum {
  STARTTYPE_RANDOM,                 /**< Random type (0). */
  STARTTYPE_COUNTDOWN,              /**< Countdown type (1). */
  STARTTYPE_RANDOMMODIFIED,         /**< Random modified type (2). */
  STARTTYPE_IMMEDIATE,              /**< Immediate type (3). */
  STARTTYPE_WAITFORFLYWHEEL         /**< Wait for flywheel type (4). */
} OBJ_STARTTYPE_T;
```

Race Operation Type

```
typedef enum {
  RACEOPERATIONTYPE_DISABLE,                    /**< Disable type (0). */
  RACEOPERATIONTYPE_PARTICIPATIONREQUEST,       /**< Participation request type (1). */
  RACEOPERATIONTYPE_SLEEP,                      /**< Sleep type (2). */
  RACEOPERATIONTYPE_ERGINIT,                    /**< Erg initialization type (3). */
  RACEOPERATIONTYPE_PHYADDRINIT,                /**< Physical address/lane initialization type (4). */
  RACEOPERATIONTYPE_RACEWARMUP,                 /**< Race warmup type (5). */
  RACEOPERATIONTYPE_RACEINIT,                   /**< Race initialization type (6). */
  RACEOPERATIONTYPE_TIMESYNC,                   /**< Time synchronization type (7). */
  RACEOPERATIONTYPE_RACEWAITTOSTART,            /**< Race wait to start type (8). */
  RACEOPERATIONTYPE_START,                      /**< Race start type (9). */
  RACEOPERATIONTYPE_FALSESTART,                 /**< Race false start type (10). */
  RACEOPERATIONTYPE_TERMINATE,                  /**< Race terminate type (11). */
  RACEOPERATIONTYPE_IDLE,                       /**< Race idle type (12). */
  RACEOPERATIONTYPE_TACHSIMENABLE,              /**< Tach simulator enable type (13). */
  RACEOPERATIONTYPE_TACHSIMDISABLE,             /**< Tach simulator disable type (14). */
} OBJ_RACEOPERATIONTYPE_T;
```

Race State

```
typedef enum {
  RACESTATE_IDLE,                   /**< Race idle state (0). */
  RACESTATE_COUNTDOWN,              /**< Race countdown state (1). */
  RACESTATE_ROWING,                 /**< Race rowing state (2). */
  RACESTATE_INTERVAL_REST,          /**< Race interval rest state (3). */
  RACESTATE_END_INTERVAL,           /**< Race end interval state (4). */
  RACESTATE_END_WORKOUT_RACE,       /**< Race end workout state (5). */
  RACESTATE_TERMINATE_WORKOUT_RACE, /**< Race terminate workout state (6). */
  RACESTATE_FALSESTART,             /**< Race false start state (7). */
  RACESTATE_INACTIVE,               /**< Race inactive state (8). */
} OBJ_RACESTATE_T;
```

Race Type

```
typedef enum {
  RACETYPE_FIXEDDIST_SINGLEERG,             /**< Fixed distance, individual type (0). */
```

```
  RACETYPE_FIXEDTIME_SINGLEERG,                          /**< Fixed time, individual type (1). */
  RACETYPE_FIXEDDIST_TEAMERG,                           /**< Fixed distance, team type (2). */
  RACETYPE_FIXEDTIME_TEAMERG,                            /**< Fixed time, team type (3). */
  RACETYPE_WORKOUTRACESTART,                             /**< Workout race start type (4). */
  RACETYPE_FIXEDCAL_SINGLEERG,                           /**< Fixed calorie, individual type (5). */
  RACETYPE_FIXEDCAL_TEAMERG,                             /**< Fixed calorie, team type (6). */
  RACETYPE_FIXEDDIST_RELAY_SINGLEERG,                    /**< Fixed distance, relay individual type (7). */
  RACETYPE_FIXEDTIME_RELAY_SINGLEERG,                   /**< Fixed time, relay individual type (8). */
  RACETYPE_FIXEDCAL_RELAY_SINGLEERG,                     /**< Fixed calorie, relay individual type (9). */
  RACETYPE_FIXEDDIST_RELAY_TEAMERG,                      /**< Fixed distance, relay team type (10). */
  RACETYPE_FIXEDTIME_RELAY_TEAMERG,                     /**< Fixed time, relay team type (11). */
  RACETYPE_FIXEDCAL_RELAY_TEAMERG,                      /**< Fixed calorie, relay team type (12). */
  RACETYPE_FIXEDDIST_MULTIACTIVITY_SEQUENTIAL_SINGLEERG,          /**< Fixed distance, multiactivity
individual type, sequential use (13). */
  RACETYPE_FIXEDTIME_MULTIACTIVITY_SEQUENTIAL_SINGLEERG,          /**< Fixed time, multiactivity
individual type, sequential use (14). */
  RACETYPE_FIXEDCAL_MULTIACTIVITY_SEQUENTIAL_SINGLEERG,           /**< Fixed calorie, multiactivity
individual type, sequential use (15). */
  RACETYPE_FIXEDDIST_MULTIACTIVITY_SEQUENTIAL_TEAMERG,            /**< Fixed distance, multiactivity team
type, sequential use (16). */
  RACETYPE_FIXEDTIME_MULTIACTIVITY_SEQUENTIAL_TEAMERG,            /**< Fixed time, multiactivity team
type, sequential use (17). */
  RACETYPE_FIXEDCAL_MULTIACTIVITY_SEQUENTIAL_TEAMERG,             /**< Fixed calorie, multiactivity team
type, sequential use (18). */
  RACETYPE_FIXEDDIST_ERGATHLON,                          /**< Fixed distance, Ergathlon type (19). */
  RACETYPE_FIXEDTIME_ERGATHLON,                         /**< Fixed time, Ergathlon type (20). */
  RACETYPE_FIXEDCAL_ERGATHLON,                           /**< Fixed calorie, Ergathlon type (21). */
  RACETYPE_FIXEDDIST_MULTIACTIVITY_SIMULTANEOUS_SINGLEERG,        /**< Fixed distance, multiactivity
individual type, simultaneous use (22). */
  RACETYPE_FIXEDTIME_MULTIACTIVITY_SIMULTANEOUS_SINGLEERG,  /**< Fixed time, multiactivity individual
type, simultaneous use (23). */
  RACETYPE_FIXEDCAL_MULTIACTIVITY_SIMULTANEOUS_SINGLEERG,   /**< Fixed calorie, multiactivity
individual type, simultaneous use (24). */
  RACETYPE_FIXEDDIST_MULTIACTIVITY_SIMULTANEOUS_TEAMERG,    /**< Fixed distance, multiactivity team
type, simultanous use (25). */
  RACETYPE_FIXEDTIME_MULTIACTIVITY_SIMULTANEOUS_TEAMERG,    /**< Fixed time, multiactivity team type,
simultaneous use (26). */
  RACETYPE_FIXEDCAL_MULTIACTIVITY_SIMULTANEOUS_TEAMERG     /**< Fixed calorie, multiactivity team type,
simultaneous use (27). */
  RACETYPE_FIXEDDIST_BIATHLON,              /**< Fixed distance, Biathlon type (28). */
  RACETYPE_FIXEDCAL_BIATHLON,               /**< Fixed calorie, Biathlon type (29). */
  RACETYPE_FIXEDDIST_RELAY_NOCHANGE_SINGLEERG,  /**< Fixed distance, no change prompt, relay individual
type (30). */
  RACETYPE_FIXEDTIME_RELAY_NOCHANGE_SINGLEERG,  /**< Fixed time, no change prompt, relay individual type
(31). */
  RACETYPE_FIXEDCAL_RELAY_NOCHANGE_SINGLEERG,   /**< Fixed calorie, no change prompt, relay individual type
(32). */
  RACETYPE_FIXEDTIME_CALSCORE_SINGLEERG,       /**< Fixed time, calorie score, individual type (33). */
  RACETYPE_FIXEDTIME_CALSCORE_TEAMERG          /**< Fixed time, calorie score, team type (34). */
  RACETYPE_FIXEDDIST_TIMECAP_SINGLEERG,        /**< Fixed time, calorie score, individual type (35). */
  RACETYPE_FIXEDCAL_TIMECAP_SINGLEERG          /**< Fixed time, calorie score, team type (36). */}
OBJ_RACETYPE_T;
```

Race Start State

```
typedef enum {
   RACESTARTSTATE_INIT,                          /**< Init state (0). */
   RACESTARTSTATE_PREPARE,                       /**< Prepare state (1). */
   RACESTARTSTATE_WAITREADY,                     /**< Wait ready state (2). */
   RACESTARTSTATE_WAITATTENTION,                 /**< Wait attention state (3). */
   RACESTARTSTATE_WAITROW,                       /**< Wait row state (4). */
```

```
   RACESTARTSTATE_COUNTDOWN,          /**< Countdown state (5). */
   RACESTARTSTATE_ROW,                /**< Row state (6). */
   RACESTARTSTATE_FALSESTART          /**< False start state (7). */
} OBJ_RACESTARTSTATE_T;
```

Screen Type

```
typedef enum {
  SCREENTYPE_NONE,
  SCREENTYPE_WORKOUT,          /**< Workout type (0). */
  SCREENTYPE_RACE,             /**< Race type (1). */
  SCREENTYPE_CSAFE,            /**< CSAFE type (2). */
  SCREENTYPE_DIAG,             /**< Diagnostic type (3). */
  SCREENTYPE_MFG,              /**< Manufacturing type (4). */
} OBJ_SCREENTYPE_T;
```

Screen Value (Workout Type)

```
typedef enum {
  SCREENVALUEWORKOUT_NONE,                          /**< None value (0). */
  SCREENVALUEWORKOUT_PREPARETOROWWORKOUT,           /**< Prepare to workout type (1). */
  SCREENVALUEWORKOUT_TERMINATEWORKOUT,              /**< Terminate workout type (2). */
  SCREENVALUEWORKOUT_REARMWORKOUT,                  /**< Rearm workout type (3). */
  SCREENVALUEWORKOUT_REFRESHLOGCARD,                /**< Refresh local copies of logcard structures(4). */
  SCREENVALUEWORKOUT_PREPARETORACESTART,            /**<  Prepare to race start (5). */
  SCREENVALUEWORKOUT_GOTOMAINSCREEN,                /**< Goto to main screen (6). */
  SCREENVALUEWORKOUT_LOGCARDBUSYWARNING,            /**< Log device busy warning (7). */
  SCREENVALUEWORKOUT_LOGCARDSELECTUSER,             /**< Log device select user (8). */
  SCREENVALUEWORKOUT_RESETRACEPARAMS,               /**< Reset race parameters (9). */
  SCREENVALUEWORKOUT_CABLETESTSLAVE,                /**< Cable test slave indication(10). */
  SCREENVALUEWORKOUT_FISHGAME,                      /**< Fish game (11). */
  SCREENVALUEWORKOUT_DISPLAYPARTICIPANTINFO,        /**< Display participant info (12). */
  SCREENVALUEWORKOUT_DISPLAYPARTICIPANTINFOCONFIRM, /**<  Display participant info w/ confirmation
(13). */
  SCREENVALUEWORKOUT_CHANGEDISPLAYTYPETARGET = 20,      /**< Display type set to target (20). */
  SCREENVALUEWORKOUT_CHANGEDISPLAYTYPESTANDARD,         /**< Display type set to standard (21). */
  SCREENVALUEWORKOUT_CHANGEDISPLAYTYPEFORCEVELOCITY, /**< Display type set to forcevelocity (22). */
  SCREENVALUEWORKOUT_CHANGEDISPLAYTYPEPACEBOAT,         /**< Display type set to Paceboat (23). */
  SCREENVALUEWORKOUT_CHANGEDISPLAYTYPEPERSTROKE,        /**< Display type set to perstroke (24). */
  SCREENVALUEWORKOUT_CHANGEDISPLAYTYPESIMPLE,           /**< Display type set to simple (25). */
  SCREENVALUEWORKOUT_CHANGEUNITSTYPETIMEMETERS = 30, /**< Units type set to timemeters (30). */
  SCREENVALUEWORKOUT_CHANGEUNITSTYPEPACE,               /**< Units type set to pace (31). */
  SCREENVALUEWORKOUT_CHANGEUNITSTYPEWATTS,              /**< Units type set to watts (32). */
  SCREENVALUEWORKOUT_CHANGEUNITSTYPECALORICBURNRATE, /**< Units type set to caloric burn rate(33). */
  SCREENVALUEWORKOUT_TARGETGAMEBASIC,                   /**< Gasic target game (34). */
  SCREENVALUEWORKOUT_TARGETGAMEADVANCED,                /**< Advanced target game (35). */
  SCREENVALUEWORKOUT_DARTGAME,                          /**< Dart game (36). */
  SCREENVALUEWORKOUT_GOTOUSBWAITREADY,                  /**< USB wait ready (37). */
  SCREENVALUEWORKOUT_TACHCABLETESTDISABLE,              /**< Tach cable test disable (38). */
  SCREENVALUEWORKOUT_TACHSIMDISABLE,                    /**< Tach simulator disable (39). */
  SCREENVALUEWORKOUT_TACHSIMENABLERATE1,                /**< Tach simulator enable, rate = 1:12 (40). */
  SCREENVALUEWORKOUT_TACHSIMENABLERATE2,                /**< Tach simulator enable, rate = 1:35 (41). */
  SCREENVALUEWORKOUT_TACHSIMENABLERATE3,                /**< Tach simulator enable, rate = 1:42 (42). */
  SCREENVALUEWORKOUT_TACHSIMENABLERATE4,                /**< Tach simulator enable, rate = 3:04 (43). */
  SCREENVALUEWORKOUT_TACHSIMENABLERATE5,                /**< Tach simulator enable, rate = 3:14 (44). */
  SCREENVALUEWORKOUT_TACHCABLETESTENABLE,               /**< Tach cable test enable (45). */
  SCREENVALUEWORKOUT_CHANGEUNITSTYPECALORIES,           /**< Units type set to calories(46). */
   SCREENVALUEWORKOUT_VIRTUALKEY_A,                     /**< Virtual key select A (47). */
  SCREENVALUEWORKOUT_VIRTUALKEY_B,                      /**< Virtual key select B (48). */
  SCREENVALUEWORKOUT_VIRTUALKEY_C,                      /**< Virtual key select C (49). */
```

```
  SCREENVALUEWORKOUT_VIRTUALKEY_D,                    /**< Virtual key select D (50). */
  SCREENVALUEWORKOUT_VIRTUALKEY_E,                    /**< Virtual key select E (51). */
  SCREENVALUEWORKOUT_VIRTUALKEY_UNITS,                /**< Virtual key select Units (52). */
  SCREENVALUEWORKOUT_VIRTUALKEY_DISPLAY,              /**< Virtual key select Display (53). */
  SCREENVALUEWORKOUT_VIRTUALKEY_MENU,                 /**< Virtual key select Menu (54). */
  SCREENVALUEWORKOUT_TACHSIMENABLERATERANDOM,         **< Tach simulator enable, rate = random (55). */
  SCREENVALUEWORKOUT_SCREENREDRAW = 255               /**< Screen redraw (255). */
} OBJ_SCREENVALUEWORKOUT_T;
```

Screen Value (Race Type)

```
typedef enum {
  SCREENVALUERACE_NONE,                               /**< None value (0). */
  SCREENVALUERACE_SETPHYSICALADDR,                    /**< Set physical address (1). */
  SCREENVALUERACE_CONFIRMPHYSICALADDR,                /**< Confirm physical address (2). */
  SCREENVALUERACE_WARMUPFORRACE,                      /**< Warmup for race (3). */
  SCREENVALUERACE_PREPARETORACE,                      /**< Prepare to race (4). */
  SCREENVALUERACE_FALSESTARTRACE,                     /**< False start race (5). */
  SCREENVALUERACE_TERMINATERACE,                      /**< Terminate race (6). */
  SCREENVALUERACE_AUTOSETPHYSADDR,                    /**< Automatically set physical address (7). */
  SCREENVALUERACE_SETPARTICIPANTLIST,                 /**< Indication that participant list is being set (8). */
  SCREENVALUERACE_SYNCRACETIME,                       /**< Indication that race time sync is occuring (9). */
  SCREENVALUERACE_PREPARETOSLEEP,                     /**< Preparation for sleeping erg (10). */
  SCREENVALUERACE_RESETRACEPARAMS,                    /**< Reset race parameters (11). */
  SCREENVALUERACE_SETDEFAULTCOMMPARAMS,               /**< Set default communication parameters (12). */
  SCREENVALUERACE_RACEIDLE,                           /**< Enter race idle (13). */
  SCREENVALUERACE_ERGADDRESSSTATUS,                   /**< Display current erg physical address (14). */
  SCREENVALUERACE_RACEIDLEROW,                        /**< Enter race idle row (15). */
  SCREENVALUERACE_DISPLAYRACEBITMAP,                  /**< Display race bitmap (16). */
  SCREENVALUERACE_DISPLAYRACETEXTSTRING,              /**< Display race text string (17). */
  SCREENVALUERACE_SETLOGICALADDR,                     /**< Set logical address (18). */
  SCREENVALUERACE_CONFIRMLOGICALADDR,                 /**< Confirm logical address (19). */
  SCREENVALUERACE_ERGSLAVEDISCOVERY,                  /**< Discover secondary Ergs (20). */
  SCREENVALUERACE_GOTOMAINSCREEN,                     /**< Goto to main screen (21). */
  SCREENVALUERACE_RESETERG,                           /**< Reset Erg (22). */
  SCREENVALUERACE_SETUNITSTYPEDEFAULT,                /**< Set units type to default (23). */
  SCREENVALUERACE_TACHSIMDISABLE = 39,                /**< Tach simulator disable (39). */
  SCREENVALUERACE_TACHSIMENABLERATE1,                 /**< Tach simulator enable, rate = 1:12 (40). */
  SCREENVALUERACE_TACHSIMENABLERATE2,                 /**< Tach simulator enable, rate = 1:35 (41). */
  SCREENVALUERACE_TACHSIMENABLERATE3,                 /**< Tach simulator enable, rate = 1:42 (42). */
  SCREENVALUERACE_TACHSIMENABLERATE4,                 /**< Tach simulator enable, rate = 3:04 (43). */
  SCREENVALUERACE_TACHSIMENABLERATE5,                 /**< Tach simulator enable, rate = 3:14 (44). */
  SCREENVALUERACE_TACHCABLETESTENABLE,                /**< Tach cable test enable (45). */
  SCREENVALUERACE_ERGATHLONMODEDISABLE,               /**< Ergathlon mode disable (46). */
  SCREENVALUERACE_RS485FIRMWAREUPDATEPROGRESS, /**< RS-485 firmware update in progress (47). */
  SCREENVALUERACE_TERMINATERACEANDPRESERVERESULTS, /**< Terminate race and preserve results (48). */
  SCREENVALUERACE_TACHSIMENABLERATERANDOM,     /**< Tach simulator enable, rate = random (49). */
  SCREENVALUERACE_SCREENREDRAW = 255                  /**< Screen redraw (255). */
} OBJ_SCREENVALUERACE_T;
```

Screen Value (CSAFE Type)

```
typedef enum {
  SCREENVALUECSAFE_NONE,                              /**< None value (0). */
  SCREENVALUECSAFE_USERID,                            /**< Enter user ID (1). */
  SCREENVALUECSAFE_PREPARETOROWWORKOUT,               /**< Prepare to workout (2). */
  SCREENVALUECSAFE_GOTOMAINSCREEN,                    /**< Goto to main screen (3). */
  SCREENVALUECSAFE_CUSTOM,                            /**< Goto custom screen (4). */
  SCREENVALUECSAFE_RACECHANOPEN = 250,                /**< Open racing channel (250). */
  SCREENVALUECSAFE_RACECHANCLOSE = 251,               /**< Close racing channel (251). */
```

```
  SCREENVALUECSAFE_SCREENREDRAW = 255                    /**< Screen redraw (255). */
} OBJ_SCREENVALUECSAFE_T;
```

Screen Status

```
enum
{
   APGLOBALS_SCREENPENDINGFLG_INACTIVE = 0,
   APGLOBALS_SCREENPENDINGFLG_PENDING,
   APGLOBALS_SCREENPENDINGFLG_INPROGRESS,
};
```

Status Type

```
typedef enum {

  STATUSTYPE_NONE,                                    /**< None (0). */
  STATUSTYPE_BATTERY_LEVEL1_WARNING,          /**< Battery level 1 warning, status value = (current battery
                                                          level/max battery value) * 100 (1). */
  STATUSTYPE_BATTERY_LEVEL2_WARNING,          /**< Battery level 2 warning, status value = (current battery
                                                          level/max battery value) * 100 (2). */
  STATUSTYPE_LOGDEVICE_STATE,                  /**< Log device state, status value = log device status (3). */
  STATUSTYPE_LOGCARD_STATE = STATUSTYPE_LOGDEVICE_STATE,  /**< Log device state, status value = log
                                                                                  device status (3). */
  STATUSTYPE_POWERSOURCE_STATE,               /**< Power source, status value = power source status (4). */
  STATUSTYPE_LOGCARD_WORKOUTLOGGED_STATUS,   /**< Log device workout logged, status value = workout
                                                                  logged status (5). */
  STATUSTYPE_FLYWHEEL_STATE,                   /**< Flywheel, status value = not turning, turning (6). */
  STATUSTYPE_BAD_UTILITY_STATE,                /**< Bad utility, status value = correct utilty, wrong utility (7). */
  STATUSTYPE_FWUPDATE_STATUS,                 /**< Firmware update, status value = no update pending, update
                                                          pending, update complete (8). */
  STATUSTYPE_UNSUPPORTEDUSBHOSTDEVICE,      /**< Unsupported USB host device, status value = unused (9). */
  STATUSTYPE_USBDRIVE_STATE,                   /**< USB host drive, status value = uninitialized, initialized (10). */
  STATUSTYPE_LOADCONTROL_STATUS,             /**< Load control, status value = all loads allowed, usb host not
                                                          allowed, backlight not allowed, neither allowed (11). */
  STATUSTYPE_USBLOGBOOK_STATUS,              /**< USB log book, status value = directory missing/corrupt, file
                                                          missing/corrupt, validated (12). */
  STATUSTYPE_LOGSTORAGECAPACTYWARNING_STATUS,  /**< Log storage capacity warning, status value = current
                                                                  used capacity (13). */
  STATUSTYPE_FACTORYCALIBRATION_WARNING,    /**< Full calibration warning, status value = unused (14). */
  STATUSTYPE_VERIFYCALIBRATION_WARNING,      /**< Verify calibration warning, status value = unused (15). */
  STATUSTYPE_SERVICECALIBRATION_WARNING,     /**< Service calibration warning, status value = unused (16). */
} OBJ_STATUSTYPE_T;
```

Display Update Rate

```
typedef enum {
   DISPLAY_UPDATERATE_5HZ,                             /**< 5Hz (0). */
   DISPLAY_UPDATERATE_4HZ,                             /**< 4Hz (1). */
   DISPLAY_UPDATERATE_2HZ                              /**< 2Hz (2). */
} OBJ_DISPLAYUPDATERATE_T;
```

Wireless Channel Flags

```
typedef enum {
  WIRELESSCHANNELFLG_NFC = 0x00000001,                 /**< NFC channel (1). */
  WIRELESSCHANNELFLG_BLEHRM = 0x00000002,              /**< BLE HRM channel (2). */
  WIRELESSCHANNELFLG_BLEMOBILE = 0x00000004,           /**< BLE mobile channel (4). */
  WIRELESSCHANNELFLG_ANTHRM = 0x00000008,              /**< ANT+ HRM channel (8). */
  WIRELESSCHANNELFLG_ANTRACING = 0x00000010,           /**< ANT+ racing channel (16). */
  WIRELESSCHANNELFLG_ANTFE = 0x000000020,              /**< ANT+ FE channel (32). */
  WIRELESSCHANNELFLG_ANTFEC = 0x00000040,              /**< ANT+ FEC channel (64). */
```

```
  WIRELESSCHANNELFLG_ANTSPDCAD = 0x00000080,      /**< ANT+ speed/cadence channel (128). */
  WIRELESSCHANNELFLG_ANTBPWR = 0x00000100,        /**< ANT+ cycling power channel (256). */
  WIRELESSCHANNELFLG_ANTFECGRP = 0x00000200,      /**< ANT+ FEC group channel (512). */
  WIRELESSCHANNELFLG_UNUSED = 0xFFFFFFFF,         /**< Unused channel (0xFFFFFFFF). */
} OBJ_WIRELESSCHANNELFLG_T;
```

Log Structure Identifiers

LOGMAP_RECIDENT_PM5_LOGHEADER = 21
LOGMAP_RECIDENT_PM5_LOGFIXEDHEADERDATA = 22
LOGMAP_RECIDENT_PM5_LOGSPLITDATA = 23
LOGMAP_RECIDENT_PM5_LOGFIXEDINTERVALHEADER = 24
LOGMAP_RECIDENT_PM5_LOGFIXEDINTERVALDATA = 25
LOGMAP_RECIDENT_PM5_LOGVARIABLEINTERVALHEADER = 26
LOGMAP_RECIDENT_PM5_LOGVARIABLEINTERVALDATA = 27
LOGMAP_RECIDENT_PM5_COMBINED_LOGHEADER_LOGFIXEDHEADERDATA = 128
LOGMAP_RECIDENT_PM5_COMBINED_LOGHEADER_LOGFIXEDINTERVALHEADER = 129
LOGMAP_RECIDENT_PM5_COMBINED_LOGHEADER_LOGVARIABLEINTERVALHEADER = 130

CPU Speed/Tick Rate

Low Speed = 4 (128 ticks/sec)
Med Speed = 2 (256 ticks/sec)
High Speed = 1 (512 ticks/sec)

Game ID

```
enum {
        APGLOBALS_GAMEID_NONE,
        APGLOBALS_GAMEID_FISH,
        APGLOBALS_GAMEID_DART,
        APGLOBALS_GAMEID_TARGET_BASIC,
        APGLOBALS_GAMEID_TARGET_ADVANCED,
        APGLOBALS_GAMEID_CROSSTRAINING};
```

Game Identifier / Verified Information

The Game Identifier/Workout Verified byte in the *C2 rowing end of workout additional summary data characteristic 2* contains two independent data. The Game Identifer is contained in the lower nibble with the enumeration as defined above. The Workout Verified flag is contained in the upper nibble. See the additional definitions below.

```
#define LOGMAP_GAMETYPEIDENT_PM5_MSK                0x0F
#define LOGMAP_LOGHEADER_STRUCT_VERIFIED_MSK        0xF0

#define LOGMAP_GET_GAMETYPEIDENT_M(gameid)             \
   ((UINT8_T)(gameid & LOGMAP_GAMETYPEIDENT_PM5_MSK))

#define LOGMAP_GET_WORKOUTVERIFIED_M(gameid)             \
    ((UINT8_T)(( gameid & LOGMAP_LOGHEADER_STRUCT_VERIFIED_MSK) >> 4))
```

Communicating with the PM using CSAFE Commands

The C2 PM Receive Characteristic and C2 PM Transmit Characteristic can be used to send and receive CSAFE frames. In general refer to the PM communications specification and the CSAFE protocol specification for information on how to do this. The following are some additional notes to supplement these specifications.

Retrieving Heartrate Belt Information

The PM Heart Rate Belt Information Characteristic will send data whenever it changes. You can also get this data using a CSAFE command. As the PM5 now supports the Polar H7 and similar Bluetooth Smart heart rate belts with 32-bit belt IDs, use this new CSAFE command: CSAFE_PM_GET_EXTENDED_HBELT_INFO – 0x57
This command returns a 1 byte user number, 1 byte manufacturer ID, 1 byte device type and 4-byte belt id.

Commanding the PM5 to Pair with a known Heartrate Belt

If your application saves the heart rate belt information then you can command the PM to automatically pair with the belt each time you connect with the PM. This will save a step for the user, as typically he had to pair the PM to a belt using the PM front panel menus. To do this use the CSAFE command CSAFE_PM_SET_EXTENDED_HRM – 0x39. This command uses the same parameters as the GET function in the previous paragraph.

Appendix B

Data Representation

Time and Distance Displayed

1. Meters: 1m resolution (no rounding/truncating)
2. Time: 1sec resolution for elapsed time/pace, 0.1sec resolution for avg pace(rounded from 0.01sec)
3. Stroke rate: 1spm resolution

Time and Distance Stored in Workout Log

1. Meters: 1m resolution (no rounding/truncating)
2. Time: 0.1sec resolution (rounded from 0.01sec)
3. Stroke rate: 1spm resolution

Data Calculation

Display

1. Workout Summary:
    a. Floating point elapsed time is truncated to 0.1sec resolution.
    b. Total distance at 1m resolution.
2. Stroke Pace:
    a. Floating point distance and time used to compute stroke pace, result is rounded to 1sec resolution.
3. Avg Pace:
    a. Floating point distance and time used to compute avg pace, result is rounded to 0.1sec resolution.
4. Split Pace:
    a. Floating point distance and time used to compute split pace, result is rounded to 0.1sec resolution.
5. Stroke Rate:
    a. Floating point stroke duration (time) rounded to 1spm resolution.
6. Avg Stroke Rate:
    a. Stroke count and floating point elapsed time truncated to 1spm resolution.

Workout Log

1. Workout summary:
    a. Stroke rate for each interval/split is added (1spm resolution) and average taken. Result is truncated to 1spm resolution.
    b. Floating point elapsed time rounded from 0.01sec resolution to 0.1sec resolution.
    c. Total distance at 1m resolution.
    d. Average pace is computed using b. and c. above and truncated to 0.1sec resolution.

Pace Conversions

Watts <-> Pace

Pace is in sec/meter:

Watts = (2.8 / (pace * pace * pace))

Calories/Hr <-> Pace

Pace is in sec/meter:

Calories/Hr = (((2.8 / (pace * pace * pace)) * (4.0 * 0.8604)) + 300.0)

Pace <-> /500m Pace

Pace is in sec/meter:

Pace/500m = (pace * 500)

Data Construction

Multi-byte data ordering varies between Smarth Bluetooth notifications and CSAFE command so be aware. Note that all calculations are integer.

Two Byte Data

Constructing two-byte data from an array of single byte values is done by combining the single byte data in pairs. If the array of single byte data is represented as Data[10] = {data0, data1, data2, data3, data4, data5, data6, data7, data8, data9} and ordered as [Hi, Lo] or [MSB, LSB]. The the five two-byte values are produced as follows:

value0 = (data0 * 256) + data1
value1 = (data2 * 256) + data3
value2 = (data4 * 256) + data5
value3 = (data6 * 256) + data7
value4 = (data8 * 256) + data9

A numeric example would be:

Data[10] = {0, 200, 1, 150, 8, 0, 50, 15, 100, 23}

value0 = (0 * 256) + 200 = 200
value1 = (1 * 256) + 150 = 406
value2 = (8 * 256) + 0 = 2048
value3 = (50 * 256) + 15 = 12,815
value4 = (100 * 256) + 23 = 25,623

Three Byte Data

Constructing three-byte data from an array of single byte values is done by combining the single byte data in sets. If the array of single byte data is represented as Data[9] = {data0, data1, data2, data3, data4, data5, data6, data7, data8} and ordered as [Lo, Mid, Hi]. The the three three-byte values are produced as follows:

value0 = (data2 * 65536) + (data1 * 256) + data0
value1 = (data5 * 65536) + (data4 * 256) + data3
value2 = (data8 * 65536) + (data7 * 256) + data6

A numeric example would be:

Data[9] = {33, 3, 0, 150, 8, 4, 50, 30, 10}

value0 = (0 * 65536) + (3 * 256) + 33 = 801

value1 = (4 * 65536) + (8 * 256) + 150 = 264,342
value2 = (10 * 65536) + (30 * 256) + 50 = 663,106

Four Byte Data

Constructing four-byte data from an array of single byte values is done by combining the single byte data in sets. If the array of single byte data is represented as Data[8] = {data0, data1, data2, data3, data4, data5, data6, data7} and ordered as [Lo, Mid Lo, Mid Hi, Hi] or [MSB, Lo MSB, Hi LSB, LSB]. The the four-byte values are produced as follows:

value0 = (data3 * 16777216) + (data2 * 65536) + (data1 * 256) + data0
value1 = (data7 * 16777216) + (data6 * 65536) + (data5 * 256) + data4

A numeric example would be:

Data[8] = {4, 3, 2, 1, 1, 2, 3, 4}

value0 = (1 * 16777216) + (2 * 65536) + (3 * 256) + 4 = 16,909,060
value1 = (4 * 16777216) + (3 * 65536) + (2 * 256) + 1 = 67,305,985

Data Deconstruction

Multi-byte data ordering varies between Smarth Bluetooth notifications and CSAFE command so be aware. Note that all calculations are integer.

Two Byte Data

Deconstructing two-byte data into an array of single byte values is done by separating according to byte ordering. If the array of source values is represented as Source[4] = {src0, src1, src2, src3}, and the single byte data array wants to be ordered as [Hi, Lo] or [MSB, LSB]. The eight single-byte values are produced as follows:

data0 = src0/256
data1= src0 – (data0 * 256)
data2 = (src1/256)
data3 = src1 – (data1 * 256)
data4 = (src2/256)
data5 = src2 – (data2 * 256)
data6 = (src3/256)
data7 = src3 – (data3 * 256)

Data[8] = {data0, data1, data2, data3, data4, data5, data6, data7}

A numeric example would be:

Source[4] = {257, 32767, 63, 60000}

data0 = 257/256 = 1
data1= 257 – (1 * 256) = 1
data2 = (32767/256) = 127
data3 = 32767– (127 * 256) = 255
data4 = (63/256) = 0
data5 = 63– (0 * 256) = 63
data6 = (60000/256) = 234
data7 = 60000– (234 * 256) = 96

Data[8] = {1, 1, 127, 255, 0, 63, 234, 96}

Three Byte Data

Deconstructing three-byte data into an array of single byte values is done by separating according to byte ordering. If the array of source values is represented as Source[3] = {src0, src1, src2}, and the single byte data array wants to be ordered as [Lo, Mid, Hi]. The nine single-byte values are produced as follows (notice the order of calculation):

data2 = (src0/65536)
data1= (src0 – (data2 * 65536))/256
data0= src0 – (data2 * 65536) – (data1 * 256)

data5 = (src1/65536)
data4= (src1 – (data5 * 65536))/256
data3= src1– (data5 * 65536) – (data4 * 256)

data8 = (src2/65536)
data7= (src2 – (data8 * 65536))/256
data6= src2– (data8 * 65536) – (data7 * 256)

Data[9] = {data0, data1, data2, data3, data4, data5, data6, data7, data8}

A numeric example would be:

Source[4] = {65537, 150000, 57}

data2 = (65537/65536) = 1
data1= (65537– (1 * 65536))/256 = 0
data0= 65537– (1 * 65536) – (0 * 256) = 1

data5 = (150000/65536) = 2
data4= (150000– (2 * 65536))/256 = 73
data3= 150000– (2 * 65536) – (73 * 256) = 240

data8 = (57/65536) = 0
data7= (src2 – (0 * 65536))/256 = 0
data6= src2– (0 * 65536) – (0 * 256) = 57

Data[8] = {1, 0, 1, 240, 73, 2, 57, 0, 0}

Four Byte Data

Deconstructing four-byte data into an array of single byte values is done by separating according to byte ordering. If the array of source values is represented as Source[2] = {src0, src1}, and the single byte data array wants to be ordered as [Lo, Mid, Hi]. The eight single-byte values are produced as follows (notice the order of calculation):

data3 = (src0/16777216)
data2 = (src0 – (data3 * 16777216))/65536
data1= (src0 – (data3 * 16777216) – (data2 * 65536))/256
data0= src0 – (data3 * 16777216) – (data2 * 65536) - (data1 * 256)

Data[8] = {data0, data1, data2, data3, data4, data5, data6, data7}

A numeric example would be:

Source[2] = {16909060, 67305985}

data3 = (16909060/16777216) = 1
data2 = (16909060– (1 * 16777216))/65536 = 2
data1= (16909060– (1 * 16777216) – (2 * 65536))/256 = 3
data0= 16909060– (1 * 16777216) – (2 * 65536) - (3 * 256) = 4

data3 = (67305985/16777216) = 4
data2 = (67305985– (4 * 16777216))/65536 = 3
data1= (67305985– (4 * 16777216) – (3 * 65536))/256 = 2
data0= 67305985– (4 * 16777216) – (3 * 65536) - (2 * 256) = 1

Data[8] = {4, 3, 2, 1, 1, 2, 3, 4}

Appendix C

Pre-programmed workout definitions for standard list and custom list are defined below. Note that the "Custom List" and "Favorites" workouts can vary from PM to PM depending on actions taken by the user.

Row/Ski Erg Standard List Workouts

Program # / Description
1 - 2000m Fixed Distance with 500m splits
2 - 5000m Fixed Distance with 1000m splits
3 - 10000m Fixed Distance with 2000m splits
4 - 30:00 Fixed Time w/ 6:00 splits
5 - 500m Fixed Distance Interval with 1:00 rest between intervals (500m/1:00r)

Row/Ski Erg Custom List Workouts

Program # / Description
6 - 00:30 Fixed Time Interval w/ 00:30 rest between intervals (:30/:30r)
7 - 7 Interval Variable (1:00/1:00r, 2:00/2:00r, 3:00/3:00r, 4:00/4:00r, 3:00/3:00r, 2:00/2:00r, 1:00/1:00r)
8 - 4 Interval Variable (2000m/3:00r, 1500m/3:00r, 1000m/3:00r, 500m/3:00r)
9 - 9 Interval Variable (1:40/:20r, 1:40/:20r, 1:40/:20r, 1:40/:20r, 1:40/2:00r, 1:40/:20r, 1:40/:20r, 1:40/:20r, 1:40/:20r)
10 - 42195 Fixed Distance with 2000m splits

Bike Erg Standard List Workouts

Program # / Description
1 - 1000m Fixed Distance with 250m splits
2 - 5000m Fixed Distance with 500m splits
3 - 30:00 Fixed Time w/ 5:00 splits
4 – 50cal Fixed Calorie with 10cal splits
5 – 1:00 Fixed Time Interval with 1:00 rest between intervals (1:00/1:00r)

Bike Erg Custom List Workouts

Program # / Description
6 - 00:20 Fixed Time Interval w/ 00:10 rest between intervals (:20/:10r)
7 – 1:00:00 Fixed Time w/ 10:00 splits
8 – 40,000m Fixed Distance w/ 5000m splits
9 - 9 Interval Variable (1:40/:20r, 1:40/:20r, 1:40/:20r, 1:40/:20r, 1:40/2:00r, 1:40/:20r, 1:40/:20r, 1:40/:20r, 1:40/:20r)
10 – 100,000m Fixed Distance w/ 10,000m splits

Appendix D

Error Code List

The PM error display format is a combination of error code and screen number as defined below:

<Error Code> - <Screen Number>

Internal Name	Value	Description
APMAIN_TASKCREATE_ERR	1	
APMAIN_TASKDELETE_ERR	2	
APMAIN_VOLTSUPPLY_ERR	3	
APMAIN_USERKEY_STUCK_ERR A button is stuck in the 'down' (pressed) position, or possibly corrosion or liquids on the circuit board. *Proposed Error Text:"Button Stuck? Did you hold the button down while resetting or putting batteries in? Is the PM wet or damaged?"*	4	
APMAIN_TASK_INVALID_ERR	5	
APMAIN_MFGINFO_INVALID_ERR	6	
APMAIN_CIPHERKEY_INVALID_ERR	7	
APMAIN_FAILEDFLASHVERIFY_ERR	8	

Internal Name	Value	Description
APCOMM_INIT_ERR	10	
APCOMM_INVALIDPW_ERR	11	
APLOG_INIT_ERR	20	
APLOG_INVALIDUSER_ERR	21	
APLOG_USERSTATINFO_STORAGE_ERR	22	
APLOG_USERSTATINFO_RETRIEVE_ERR	23	
APLOG_USERDELETE_ERR	24	
APLOG_USERDYNAMINFO_STORAGE_ERR	25	
APLOG_USERDYNAMINFO_RETRIEVE_ERR	26	
APLOG_CUSTOMWORKOUT_STORAGE_ERR	27	
APLOG_CUSTOMWORKOUT_RETRIEVE_ERR	28	
APLOG_CUSTOMWORKOUT_INSUFFMEM_ERR	29	
APLOG_CUSTOMWORKOUT_INVALID_ERR	30	
APLOG_INVALIDCARDOPERATION_ERR	31	
APLOG_COPYTOCARD_INSUFFMEM_ERR	32	

Internal Name	Value	Description
APLOG_INVALIDCUSTOMWORKOUT_ERR	33	
APLOG_INVALIDWORKOUTIDENT_ERR	34	
APLOG_INVALIDLISTLENGTH_ERR	35	
APLOG_INVALIDINPUTPARAM_ERR	36	
APLOG_INVALIDWORKOUTNUM_ERR	37	
APLOG_CARDNOTPRESENT_ERR	38	
APLOG_INVALIDINTLOGADDR_ERR	39	
APLOG_INVALIDLOGHDRPTR_ERR	40	
APLOG_MAXSPLITSEXCEEDED_ERR	41	
APLOG_NODATAAVAILABLE_ERR	42	42 - Some kind of internal problem has occurred - specifically some data is missing when trying to display help (textbox) or various types of selections such as listbox, textbox, listbycategory, etc. Please report to support@c2vt.com or scotth@concept2.com the full error message (ie 42-147); the firmware version; and the steps to get to the error message.
APLOG_INVALIDCARDSTRUCTREV_ERR	43	
APLOG_CARDOPERATIONTIMEOUT_ERR	44	

Internal Name	Value	Description
APLOG_INVALIDLOGSIZE_ERR	45	
APLOG_LOGENTRYVALIDATE_ERR	46	
APLOG_USERDYNAMICVALIDATE_ERR	47	
APLOG_CARDINFOVALIDATE_ERR	48	
APLOG_CARDACCESS_ERR	49	
APLOG_CORRUPT_INTERNALLOGMEM_ERR	95	
APROWXX_INVALID_DF_ERR Bad drag factor (generic)	50	
APROWXX_INVALID_DF_TACHDURATION_ERR Bad drag factor (specific)	51	
APROWXX_INVALID_DF_STROKEDURATION_ERR Bad drag factor (specific)	52	
APROWXX_INVALID_DF_RECOVERYDURATION_ERR Bad drag factor (specific)	53	

Internal Name	Value	Description
APROWXX_INVALID_DF_STROKEDURATION1_ERR Bad drag factor (specific)	54	
APPM3_INVALIDWORKOUTNUM_ERR	60	
APPM3_NOPLOTDATA_ERR	61	
APPM3_INVALIDMFGINFO_ERR	62	
APPM3_INVALIDCALINFO_ERR	63	
APPM3_INVALIDWORKOUTDURATION_ERR *See 65 for similar situation*	64	
APPM3_INVALIDSPLITDURATION_ERR *See 65 for similar situation*	65	
APPM3_INVALIDRESTDURATION_ERR' *More than likely, a software application like RowPro etc. has created a workout that is causing the splits to be too small or too large; or too many splits (more than 30). This shouldn't happen with a PM5 just by itself.*	66	
APPM3_INVALIDINTERVALCNT_ERR *See 65 for similar situation*	67	

Internal Name	Value	Description
APPM3_INVALIDWORKOUTTYPE_ERR *See 65 for similar situation*	68	
APPM3_INVALIDDBMFGINFO_ERR	69	
APMEM_CALEEPROM_ERR	75	
APMEM_MFGEEPROM_ERR	76	
APMEM_USREEPROM_ERR	77	
APMEM_DBEEPROM_ERR	78	
APHEADER_INVALIDFONTHDR_ERR	80	
APHEADER_INVALIDSCRNHDR_ERR	81	
APHEADER_INVALIDLDRHDR_ERR	82	
APHEADER_INVALIDAPPHDR_ERR	83	
APHEADER_INVALIDMFGHDR_ERR	84	
APHEADER_INVALIDEXPMFGHDR_ERR	85	
APHEADER_INVALIDDBMFGHDR_ERR	86	
APHEADER_INVALID_UPDATETYPE_ERR	87	

Internal Name	Value	Description
APHEADER_IMAGE_PROGRAMMING_ERR	88	
APHEADER_UPDATEINFO_PROGRAMMING_ERR	89	
APRACE_INVALIDWORKOUTTYPE_ERR	1700	
APRACE_INVALID_RACECONFIG_ERR	1701	
APRACE_FLYWHEELS_MOVING_ERR	1702	
APFILE_FWUPDBUNDLE_INVALID_ERR	1960	
APFILE_INTERNALBUNDLE_INVALID_ERR	1961	
APFILE_SPIIMAGE_INVALID_ERR	1962	
APFILE_SPECIALTOKEN_INVALID_ERR	1963	
APFILE_FWUPDATE_ABORTED_ERR	1964	
TKCMDPR_INVALID_RSP_ERR	120	Changed from 130 which was a duplicate (TKCMDPR_INVALID_MSGTYPE_ERR)
TKCMDPR_INVALID_CMD_ERR	121	
TKCMDPR_INVALID_CMD_ADDR_ERR	122	
TKCMDPR_INVALID_DEST_ADDR_ERR	123	
TKCMDPR_INVALID_DEST_INTF_ERR	124	

Internal Name	Value	Description
TKCMDPR_INVALID_INTF_ERR	125	
TKCMDPR_ROUTE_TABLE_FULL_ERR	126	
TKCMDPR_NO_DATA_AVAILABLE_ERR	127	
TKCMDPR_UNAUTHORIZED_CMD_ERR	128	
TKCMDPR_REFUSE_CMD_ERR	129	
TKDATALOG_INIT_ERR	130 **DUPLICATE ERR CODE**	
TKDATALOG_DEVICE_INVALID_ERR	131	
TKDATALOG_CARD_INIT_ERR	132	
TKDATALOG_DEVICE_SIZE_ERR	133	
TKDATALOG_MULTI_STRUCT_ERR	134	
TKDATALOG_READ_ERR	135	
TKDATALOG_WRITE_ERR	136	
TKDATALOG_RECORDIDENTIFIER_ERR	137	
TKDATALOG_INSUFFMEM_ERR	138	

Internal Name	Value	Description
TKDATALOG_CARD_CORRUPT_ERR	139	
TKDISP_INVALID_CHAR_ERR	140	
TKDISP_INVALIDPARAM_ERR	141	
TKDISP_STRING_TOO_LONG_ERR	142	
TKDISP_STRING_TOO_HIGH_ERR	143	
TKDISP_INVALID_LANG_ERR	144	
TKDISP_UPDATE_TIMEOUT_ERR	145	
TKEEPROM_INIT_ERR	150	
TKEEPROM_ACK_ERR	151	
TKEEPROM_STOP_ERR	152	
TKEEPROM_INVALID_END_ADDR	153	
TKEEPROM_WRITE_TIMEOUT_ERR	154	
TKEEPROM_WRITE_READ_ERR	155	
TKEEPROM_WRITE_VERIFY_ERR	156	
TKEEPROM_CHKSM_READ_ERR	157	

Internal Name	Value	Description
TKFRAME_CSAFE_FRAME_STUFF_ERR	160	
TKFRAME_CSAFE_FRAME_CHKSM_ERR Some device plugged into the USB or RJ45 Jack is not sending data correctly... ref case 3317 and also the PM3/4 error code wiki. (ONLY IF NOT ALREAY TRAPPED): *Proposed Error Text, No Translations:"Checksum Error in USB connection to computer. Contact tech support for the software."*	161	
TKFRAME_NO_SCI_FRAME_ERR	162	
TKFRAME_NO_USB_FRAME_ERR (ONLY IF NOT ALREAY TRAPPED): *Proposed Error Text, No Translations:"Checksum Error in USB connection to computer. Contact tech support for the software."*	163	
TKFRAME_CSAFE_INVALID_SHORT_CMD_ERR (ONLY IF NOT ALREAY TRAPPED): *Proposed Error Text, No Translations:"Invalid Short Command Error in USB connection to computer. Contact tech support for the software."*	164	
TKFRAME_CSAFE_INVALID_LONG_CMD_ERR (ONLY IF NOT ALREAY TRAPPED):	165	

Internal Name	Value	Description
Proposed Error Text, No Translations:"Invalid Long Command Error in USB connection to computer. Contact tech support for the software."		
TKFRAME_CSAFE_FRAME_TOO_LONG_ERR (ONLY IF NOT ALREAY TRAPPED): *Proposed Error Text, No Translations:"Frame To Long Error in USB connection to computer. Contact tech support for the software."*	166	
TKFRAME_NO_EXPRF_FRAME_ERR	167	
TKFRAME_CSAFE_INVALID_LONG_RSP_ERR (ONLY IF NOT ALREAY TRAPPED): *Proposed Error Text, No Translations:"Invalid Long Response Error in USB connection to computer. Contact tech support for the software."*	168	
TKFRAME_NO_LPBCK_FRAME_ERR	169	
TKHDW_EVENT_BURST_STACK_OVF_ERR	170	
TKHDW_EVENT_BURST_STACK_UNF_ERR	171	
TKHDW_INVALIDLEDCOLOR_ERR	172	
TKHDW_INVALIDLEDMODE_ERR	173	
TKHDW_WORKOUT_LOG_ERR	174	

Internal Name	Value	Description
TKHDW_FLYWHEEL_SPINDOWN_ERR	175	
TKHDW_BATTFILT_INIT_ERR	176	
TKHRTMON_INVALID_NUM_MEAS_ERR	180	
TKHRTMON_TOO_FEW_MEAS_ERR	181	
TKMEM_INVALID_MEMTYPE_ERR	200	
TKMEM_INVALID_START_ADDR_ERR	201	
TKMEM_INVALID_END_ADDR_ERR	202	
TKMEM_FLASH_WRITE_ERR	203	
TKMEM_FLASH_ERASE_ERR	204	
TKRTTIMER_INVALID_MONTH_ERR	210	
TKRTTIMER_INVALID_DAY_ERR	211	
TKRTTIMER_INVALID_TIMER_NUM_ERR	212	
TKRTTIMER_INVALID_TIMER_MODE_ERR	213	
TKSCI_INVALID_PORT_ERR	220	
TKSCI_TX_SEND_ERR	221	

Internal Name	Value	Description
TKSCI_RX_TIMEOUT_ERR	222	
TKSCRN_INVALID_SPECFUNCTYPE	230	
TKSCRN_ILLEGAL_SPLITDURATION	231	
TKSMCD_ACK_ERR	240	
TKSMCD_STOP_ERR	241	
TKSMCD_INVALID_END_ADDR	242	
TKSMCD_WRITE_TIMEOUT_ERR	243	
TKSMCD_WRITE_READ_ERR	244	
TKSMCD_WRITE_VERIFY_ERR	245	
TKSMCD_CHKSM_READ_ERR	246	
TKSMCD_ACK_ERR_WRITE	247	
TKTACH_INVALID_NUM_MEAS_ERR	250	
TKTACH_TOO_FEW_MEAS_ERR	251	
TKTIME_INVALID_MONTH_ERR	260	
TKTIME_INVALID_DAY_ERR	261	

Internal Name	Value	Description
TKUSER_INIT_ERR	260	.
TKCRC_ERR	300	
TKCRC_LENGTH_ERR	301	
TKCIPHER_NOT_BLOCK_MULT_ERR	320	
TKUSB_BAD_DESC_RQT_ERR	330	
TKUSB_INVALID_EPNUM_ERR	331	
TKUSB_RX_TIMEOUT_ERR	332	
TKUSB_EPNUM_RX_OVERRUN	333	
TKUSB_INIT_EPNUM_ERR	334	
TKUSB_GET_RX_CHAR_ERR	335	
TKUSB_BUS_DISABLE_ERR	336	
TKUSB_BUS_RESET_ERR	337	
TKUSB_NO_FEATURE_REPORT_ERR	338	
TKUSB_INVALID_STRING_ID_ERR	339	
TKUSB_EP_TX_OVERRUN_ERR	340	

Internal Name	Value	Description
TKUSB_INVALID_TX_LEN_ERR	341	
TKUSB_HOST_CURRFAULT_ERR	342	
TKUSB_HOST_UNSUPPORTEDDEV_ERR	343	
TKUSB_DEVICE_VOLTFAULT_ERR	344	
TKUSB_HOST_INVALIDPWRCTLMODE_ERR	345	
TKCSAFE_INVALID_CMD_ERR	380	
TKCSAFE_INVALID_WORKOUTNUM_ERR	381	
TKCSAFE_INVALID_PRECISION_ERR	382	
TKCSAFE_INVALID_FRAMETYPE_ERR	383	
TKCSAFE_INVALID_FRAMESIZE_ERR	384	
TKRF_ILLEGAL_MSG_ERR	390	
TKRF_INVALID_CHAN_OPEN_ERR	391	
TKRF_MSG_OVERRUN_ERR	392	
TKRF_CHKSUM_ERR	393	
TKRF_ANT_RESPONSE_ERR	394	

Internal Name	Value	Description
TKRF_NO_FRAME_ERR	395	
TKRF_FRAME_OVERRUN_ERR	396	
TKRF_INVALID_CNT_ERR	397	
TKRF_BROADCAST_ERR	398	
TKRF_MSG_NOT_COMPLETE_ERR	399	
TKRF_NEXT_SENSOR_PACKET_ERR	400	
TKRF_ILLEGAL_PACKET_ENTRY_ERR	401	
TKLCD_WRITE_SPI_TIMEOUT_ERR	410	
TKSLIP_INVALID_PORT_ERR	430	
TKSLIP_NOT_FOUND_ERR	431	
TKSLIP_INVALID_ADDR_ERR	432	
TKHCI_INVALID_ADDR_ERR	440	
TKHCI_NOT_FOUND_ERR	442	
TKNRF422_NOT_FOUND_ERR	443	
TKNRF422_RSP_TIMEOUT_ERR	444	
TKNRF422_RADIO_SOC_ERR - **DUPLICATE ENTRY**	445	

Internal Name	Value	Description
TKNRF422_RADIO_BASE_ERR	446	
TKNRF422_RADIO_SDM_ERR	447	
TKNRF422_RADIO_SOC_ERR	448	
TKNRF422_RADIO_STK_ERR ALL of the above 443-449 errors indicate some problem with one of the wireless chips. Either this is an annoyance (comes up once) or indicates a hardware problem (replace monitor). Updating firmware won't hurt but not known to fix this.	449	
TKNRF422_NOT_FOUND_ERR	450	
TKNRF422_RSP_TIMEOUT_ERR	451	
TKNRF422_INIT_ERR	452	
TKNRF422_RADIO_BASE_ERR	453	
TKNRF422_RADIO_SDM_ERR	454	
TKNRF422_RADIO_SOC_ERR	455	
TKNRF422_RADIO_STK_ERR	456	
TKNRF422_RADIO_ANT_ERR	457	
TKNRF422_LDR_VER_ERR	458	
TKNRF422_APP_VER_ERR	459	

Internal Name	Value	Description
TKNRF822_NOT_FOUND_ERR	460	
TKNRF822_RSP_TIMEOUT_ERR	461	
TKNRF822_INIT_ERR	462	
TKNRF822_RADIO_BASE_ERR	463	
TKNRF822_RADIO_SDM_ERR	464	
TKNRF822_RADIO_SOC_ERR	465	
TKNRF822_RADIO_STK_ERR	466	
TKNRF822_RADIO_ANT_ERR	467	
TKNRF822_LDR_VER_ERR	468	
TKNRF822_APP_VER_ERR	469	
TKNRF822_LIST_FULL_ERR	470	
TKNRF822_CONNECT_TIMEOUT_ERR	471	
TKNRF822_DISCONNECT_ERR	472	
TKRFPKT_NOT_FOUND_ERR	480	
TKRFPKT_RSP_TIMEOUT_ERR	481	

Internal Name	Value	Description
TKRFPKT_INVALID_INTF_ERR	482	
TKRFPKT_RXOVERRUN_ERR	483	
TKRFPKT_INVALID_CNT_ERR	484	
TKRFPKT_INIT_PORT_ERR	485	
TKRFPKT_ANT422_RSP_TIMEOUT_ERR	486	
TKRFPKT_BLE422_RSP_TIMEOUT_ERR	487	
TKRFPKT_BLE822_RSP_TIMEOUT_ERR	488	
TKDIAG_DIAGFAIL_ERR	500	
TKDIAG_FLSHFONTDIAG_BADHDRCRC_ERR	501	
TKDIAG_FLSHFONTDIAG_CRCCALC_ERR	502	
TKDIAG_FLSHFONTDIAG_BADFONTCRC_ERR	503	
TKDIAG_FLSHSCRNDIAG_BADHDRCRC_ERR	510	
TKDIAG_FLSHSCRNDIAG_CRCCALC_ERR	511	
TKDIAG_FLSHSCRNDIAG_BADSCRNCRC_ERR	512	
TKDIAG_FLSHAPPDIAG_BADHDRCRC_ERR	520	

Internal Name	Value	Description
TKDIAG_FLSHAPPDIAG_CRCCALC_ERR	521	
TKDIAG_FLSHAPPDIAG_BADAPPCRC_ERR	522	
TKDIAG_UARTDIAG_UART1_INIT_ERR	530	
TKDIAG_UARTDIAG_UART1_WRITE_ERR	531	
TKDIAG_UARTDIAG_UART1_READ_ERR	532	
TKDIAG_UARTDIAG_UART2_INIT_ERR	533	
TKDIAG_UARTDIAG_UART2_WRITE_ERR	534	
TKDIAG_UARTDIAG_UART2_READ_ERR	535	
TKDIAG_ADCONVDIAG_INIT_ERR	540	
TKDIAG_ADCONVDIAG_NOTREADY_ERR	541	
TKDIAG_ADCONVDIAG_ADCINPUT_ERR	542	
TKDIAG_SWUSERCONFIRM_ERR	550	
TKDIAG_SWSHORT_ERR	551	
TKDIAG_SW0_ERR	552	
TKDIAG_SW1_ERR	553	

Internal Name	Value	Description
TKDIAG_SW2_ERR	554	
TKDIAG_SW3_ERR	555	
TKDIAG_SW4_ERR	556	
TKDIAG_SW5_ERR	557	
TKDIAG_SW6_ERR	558	
TKDIAG_SW7_ERR	559	
TKDIAG_AMUXDIAG_NOTREADY_ERR	560	
TKDIAG_AMUXDIAG_ANALOGVREFCHAN_ERR	561	
TKDIAG_AMUXDIAG_ANALOGGNDCHAN_ERR	562	
TKDIAG_VSUPPLYDIAG_VEXPDIAG_ERR	570	PM3 only
TKDIAG_VSUPPLYDIAG_GENINDIAG_ERR	571	PM3 only
TKDIAG_VSUPPLYDIAG_VBATEXPDIAG_ERR	572	PM3 only
TKDIAG_VSUPPLYDIAG_VBATPROTDIAG_ERR	573	PM3 only
TKDIAG_VSUPPLYDIAG_VUSBDIAG_ERR	574	PM3 only
TKDIAG_VSUPPLYDIAG_VREFDIAG_ERR	575	PM3 only

Internal Name	Value	Description
TKDIAG_VSUPPLYDIAG_VBIASDIAG_ERR	576	PM3 only
TKDIAG_VSUPPLYDIAG_VBATDIAG_ERR	570	PM4 only
TKDIAG_VSUPPLYDIAG_VNIMHDIAG_ERR	571	PM4 only
TKDIAG_VSUPPLYDIAG_GENINDIAG_ERR	572	PM4 only
TKDIAG_VSUPPLYDIAG_VEXPDIAG_ERR	573	PM4 only
TKDIAG_VSUPPLYDIAG_VREFDIAG_ERR	574	PM4 only
TKDIAG_VSUPPLYDIAG_EXPDIAG_ERR	575	PM4 only
TKDIAG_VSUPPLYDIAG_VBIASDIAG_ERR	576	PM4 only
TKDIAG_VSUPPLYDIAG_VBATDIAG_ERR No batteries present or VERY low. Install some batteries.	570	PM5 only
TKDIAG_VSUPPLYDIAG_GENINDIAG_ERR	571	PM5 only
TKDIAG_VSUPPLYDIAG_VDDDIAG_ERR	572	PM5 only
TKDIAG_VSUPPLYDIAG_VDUSBPROTDIAG_ERR	573	PM5 only
TKDIAG_VSUPPLYDIAG_VINPROTDIAG_ERR	574	PM5 only
TKDIAG_VSUPPLYDIAG_LCDBLADIAG_ERR Backlight voltage not proper. Firmware v15 may report this just after test rowing exits, in this case this can be ignored. If	575	PM5 only

Internal Name	Value	Description
backlight is operating fine, this could be ignored.		
TKDIAG_VSUPPLYDIAG_VSW_ERR	576	PM5 only
TKDIAG_VSUPPLYDIAG_VHUSBCLDIAG_ERR See case 3513. USB Flash Drive (host) port has experienced a Current Limit (overload) condition. - Is the usb device a flash drive? If it is, maybe it draws too much current, try another - Is the user conecting ANYTHING other than a USB flash drive to the "A" (rectangular) port on the monitor? Using an A-A cable to connect to a computer is WRONG.	577	PM5 only
TKDIAG_VSUPPLYDIAG_VUSBDIAG_ERR The monitor is trying to make 5V for the USB Flash Drive port, and the 5v is not within specifications.	578	PM5 only
TKDIAG_EXTEEDIAG_RDDATA1_ERR	580	PM3 only
TKDIAG_EXTEEDIAG_INVALIDCRC1_ERR	581	PM3 only
TKDIAG_EXTEEDIAG_RDDATA2_ERR	582	PM3 only
TKDIAG_EXTEEDIAG_INVALIDCRC2_ERR	583	PM3 only
TKDIAG_EXTEEDIAG_WRDATA1_ERR	584	PM3 only
TKDIAG_EXTEEDIAG_WRDATA2_ERR	585	PM3 only
TKDIAG_EXTEEDIAG_DATA1_ERR	586	PM3 only

Internal Name	Value	Description
TKDIAG_EXTEEDIAG_DATA2_ERR	587	PM3 only
TKDIAG_EXTEEDIAG_RDDATA1_ERR	580	PM4/PM5 only
TKDIAG_EXTEEDIAG_INVALIDCRC1_ERR	581	PM4/PM5 only
TKDIAG_EXTEEDIAG_RDDATA2_ERR	582	PM4/PM5 only
TKDIAG_EXTEEDIAG_INVALIDCRC2_ERR	583	PM4/PM5 only
TKDIAG_EXTEEDIAG_RDDATA3_ERR	584	PM4 only
TKDIAG_EXTEEDIAG_INVALIDCRC3_ERR	585	PM4 only
TKDIAG_EXTEEDIAG_RDDATA4_ERR	586	PM4 only
TKDIAG_EXTEEDIAG_WRDATA1_ERR	587	PM4 only
TKDIAG_EXTEEDIAG_WRDATA2_ERR	588	PM4 only
TKDIAG_EXTEEDIAG_WRDATA3_ERR	589	PM4 only
TKDIAG_EXTEEDIAG_WRDATA4_ERR	590	PM4 only
TKDIAG_EXTEEDIAG_DATA1_ERR	591	PM4 only
TKDIAG_EXTEEDIAG_DATA2_ERR	592	PM4 only
TKDIAG_EXTEEDIAG_DATA3_ERR	593	PM4 only

Internal Name	Value	Description
TKDIAG_EXTEEDIAG_DATA4_ERR	594	PM4 only
TKDIAG_TACHDIAG_USERCONFIRM_ERR	590	PM3 only
TKDIAG_TACHDIAG_TACHUNPLUG_ERR	591	PM3 only
TKDIAG_TACHDIAG_TACHPLUG_ERR	592	PM3 only
TKDIAG_TACHDIAG_TACHSPINNING_ERR	593	PM3 only
TKDIAG_TACHDIAG_USERABORT_ERR	594	PM3 only
TKDIAG_TACHDIAG_USERCONFIRM_ERR	595	PM4/PM5 only
TKDIAG_TACHDIAG_TACHUNPLUG_ERR	596	PM4/PM5 only
TKDIAG_TACHDIAG_TACHPLUG_ERR	597	PM4/PM5 only
TKDIAG_TACHDIAG_TACHSPINNING_ERR	598	PM4/PM5 only
TKDIAG_TACHDIAG_USERABORT_ERR	599	PM4/PM5 only
TKDIAG_HRTMONDIAG_USERCONFIRM_ERR	600	
TKDIAG_HRTMONDIAG_HRTUNPLUG_ERR	601	
TKDIAG_HRTMONDIAG_HRTPLUG_ERR	602	
TKDIAG_HRTMONDIAG_HRTACTIVE_ERR	603	

Internal Name	Value	Description
TKDIAG_HRTMONDIAG_USERABORT_ERR	604	
TKDIAG_GENINPUTDIAG_USERCONFIRM_ERR	610	
TKDIAG_GENINPUTDIAG_THRESHMAX_ERR	611	
TKDIAG_GENINPUTDIAG_THRESHMIN_ERR	612	
TKDIAG_GENINPUTDIAG_USERABORT_ERR	613	
TKDIAG_SCDIAG_USERCONFIRM_ERR	620	PM3/PM4 only
TKDIAG_SCDIAG_ILLEGALDETECT_ERR	621	PM3/PM4 only
TKDIAG_SCDIAG_DETECT_ERR	622	PM3/PM4 only
TKDIAG_SCDIAG_COMM_ERR	623	PM3/PM4 only
TKDIAG_SCDIAG_USERABORT_ERR	624	PM3/PM4 only
TKDIAG_EXPCFREG_NOTPRESENT_ERR	660	PM3 only
TKDIAG_EXPCFREG_LO_ERR	661	PM3 only
TKDIAG_EXPCFREG_HI_ERR	662	PM3 only
TKDIAG_EXPSTSLED_NOTPRESENT_ERR	670	PM3 only
TKDIAG_EXPFLASH_NOTPRESENT_ERR	680	PM3 only

Internal Name	Value	Description
TKDIAG_EXPFLASH_FILLNORMALDATA_ERR	681	PM3 only
TKDIAG_EXPFLASH_NORMALDATA_ERR	682	PM3 only
TKDIAG_EXPFLASH_FILLINVERTEDDATA_ERR	683	PM3 only
TKDIAG_EXPFLASH_INVERTEDDATA_ERR	684	PM3 only
TKDIAG_EXPSRAM_NOTPRESENT_ERR	690	PM3 only
TKDIAG_EXPSRAM_NORMALDATA_ERR	691	PM3 only
TKDIAG_EXPSRAM_INVERTEDDATA_ERR	692	PM3 only
TKDIAG_EXPEEDIAG_NOTPRESENT_ERR	700	PM3 only
TKDIAG_EXPEEDIAG_INVALIDCRC_ERR	701	PM3 only
TKDIAG_EXPEEDIAG_RDDATA1_ERR	702	PM3 only
TKDIAG_EXPEEDIAG_WRDATA1_ERR	703	PM3 only
TKDIAG_EXPEEDIAG_DATA1_ERR	704	PM3 only
TKDIAG_EXP232DIAG_NOTPRESENT_ERR	710	PM3 only
TKDIAG_EXP232DIAG_CONFIG_ERR	711	PM3 only
TKDIAG_EXP232DIAG_TXCHAR_ERR	712	PM3 only

Internal Name	Value	Description
TKDIAG_EXP232DIAG_LOOPBACK_TO_ERR	713	PM3 only
TKDIAG_EXP232DIAG_LOOPBACK_DATA_ERR	714	PM3 only
TKDIAG_EXP232DIAG_USERCONFIRM_ERR	715	PM3 only
TKDIAG_EXP232DIAGDIAG_USERABORT_ERR	716	PM3 only
TKDIAG_EXP485DIAG_NOTPRESENT_ERR	720	PM3 only
TKDIAG_EXP485DIAG_CONFIG_ERR	721	PM3 only
TKDIAG_EXP485DIAG_FORMATFRAME_ERR	722	PM3 only
TKDIAG_EXP485DIAG_SENDCMD_ERR	723	PM3 only
TKDIAG_EXP485DIAG_UNFORMATFRAME_ERR	724	PM3 only
TKDIAG_EXP485DIAG_USERCONFIRM_ERR	725	PM3 only
TKDIAG_EXP485DIAGDIAG_USERABORT_ERR	726	PM3 only
TKDIAG_EXP485DIAG_NORESPONSE_ERR	727	PM3 only
TKDIAG_EXPWIFIDIAG_NOTPRESENT_ERR	730	PM3 only
TKDIAG_EXPWIFIDIAG_CFINIT_ERR	731	PM3 only
TKDIAG_EXPWIFIDIAG_RECONFIG_ERR	732	PM3 only

Internal Name	Value	Description
TKDIAG_EXPWIFIDIAG_USERABORT_ERR	733	PM3 only
TKDIAG_EXPWIFIDIAG_DHCP_TO_ERR	734	PM3 only
TKDIAG_EXPWIFIDIAG_CFNOTPRESENT_ERR	735	PM3 only
TKDIAG_EXPWIFIDIAG_NOTAVAILABLE_ERR	736	PM3 only
TKDIAG_GPIOCFREG_NOTPRESENT_ERR	740	PM4 only
TKDIAG_GPIOCFREG_LO_ERR	741	PM4 only
TKDIAG_GPIOCFREG_HI_ERR	742	PM4 only
TKDIAG_STSLED_NOTPRESENT_ERR	750	
TKDIAG_ANTRFDIAG_NOTPRESENT_ERR	760	PM4 only
TKDIAG_ANTRFDIAG_CONFIG_ERR	761	PM4 only
TKDIAG_ANTRFDIAG_FORMATFRAME_ERR	762	PM4 only
TKDIAG_ANTRFDIAG_SENDCMD_ERR	763	PM4 only
TKDIAG_ANTRFDIAG_UNFORMATFRAME_ERR	764	PM4 only
TKDIAG_ANTRFDIAG_USERCONFIRM_ERR	765	PM4 only
TKDIAG_ANTRFDIAG_USERABORT_ERR	766	PM4 only

Internal Name	Value	Description
TKDIAG_ANTRFDIAG_NORESPONSE_ERR	767	PM4 only
TKDIAG_RFDIAG_422CONFIG_ERR	760	PM5 only
TKDIAG_RFDIAG_822CONFIG_ERR	761	PM5 only
TKDIAG_RFDIAG_USERABORT_ERR	762	PM5 only
TKDIAG_RFDIAG_422XMIT_822RECV_EXCESSERRORRATE_ERR	763	PM5 only
TKDIAG_RFDIAG_822XMIT_422RECV_EXCESSERRORRATE_ERR	764	PM5 only
TKDIAG_RS485DIAG_NOTPRESENT_ERR	770	
TKDIAG_RS485DIAG_CONFIG_ERR	771	
TKDIAG_RS485DIAG_FORMATFRAME_ERR	772	
TKDIAG_RS485DIAG_SENDCMD_ERR	773	
TKDIAG_RS485DIAG_UNFORMATFRAME_ERR	774	
TKDIAG_RS485DIAG_USERCONFIRM_ERR	775	
TKDIAG_RS485DIAG_USERABORT_ERR	776	
TKDIAG_RS485DIAG_NORESPONSE_ERR	777	
TKDIAG_RS485DIAG_TXCHAR_ERR	778	

Internal Name	Value	Description
TKDIAG_RS485DIAG_LOOPBACK_TO_ERR Some of the pins in one or more of the RJ45 jacks are bent and touching each other. Inspect with a flashlight, and carefully separate the pins with tiny screwdriver or blade. Recommend that this PM5 not be used for racing! UNSURE if this error message is even active.	779	
TKDIAG_RS4852DIAG_LOOPBACK_DATA_ERR	780	
TKDIAG_RS4852DIAG_NOCABLECONNECT_ERR	781	
TKDIAG_SPIDFLASHDIAG_ERASE_ERR	785	PM5 only
TKDIAG_SPIDFLASHDIAG_WRITE_ERR	786	PM5 only
TKDIAG_SPIDFLASHDIAG_READ_ERR	787	PM5 only
TKDIAG_SPIDFLASHDIAG_READVERIFY_ERR	788	PM5 only
TKDIAG_SPIBFLASHDIAG_ERASE_ERR	790	PM5 only
TKDIAG_SPIBFLASHDIAG_WRITE_ERR	791	PM5 only
TKDIAG_SPIBFLASHDIAG_READ_ERR	792	PM5 only
TKDIAG_SPIBFLASHDIAG_READVERIFY_ERR	793	PM5 only
TKDIAG_TACHDIAG_MODELCSELECT_ERR	800	PM4/PM5 only
TKDIAG_TACHDIAG_MODELDSELECT_ERR	801	PM4/PM5 only

Internal Name	Value	Description
TKDIAG_TACHDIAG_PULSEEMULATE_ERR	802	PM4/PM5 only
TKEXP_RS232_INVALID_ERR	1000	PM3 only
TKEXP_CF_NOTPRESENT_ERR	1001	PM3 only
TKEXP_CF_CIRQINVALID_ERR	1002	PM3 only
TKEXP_CF_CARDNOTREADY_ERR	1003	PM3 only
TKEXP_CF_MEMTEST_ERR	1004	PM3 only
TKEXP_CF_INVALIDSTATE_ERR	1005	PM3 only
TKEXP_CF_RFVENDORSTRING_ERR	1006	PM3 only
TKEXP_INVALIDLEDMODE_ERR	1007	PM3 only
TKEXP_INVALIDLEDCOLOR_ERR	1008	PM3 only
TKSPIFLASH_INVALID_ID_ERR	1820	PM5 only
TKSPIFLASH_PAGE_BOUNDARY_ERR	1821	PM5 only
TKSPIFLASH_ERASE_MODE_ERR	1822	PM5 only
TKSPIFLASH_WRITEVERIFY_ERR	1823	PM5 only
TKFILE_FILESYSTEM_ERR	1840	PM5 only

Internal Name	Value	Description
TKFILE_CREATE_SUBDIR_ERR Cannot create folders on the USB Flash drive. Try another USB Stick. See if the drive is 'write protected' by using it in a PC and checking properties (mac: info)	1841	PM5 only
TKFILE_FILE_OPEN_ERR	1842	PM5 only
TKFILE_FILE_SEEK_ERR	1843	PM5 only
TKFILE_GET_CURR_DIR_ERR	1844	PM5 only
TKFILE_GET_FREE_ERR	1845	PM5 only
TKFILE_CHANGE_DIR_ERR *Proposed Error Text:"Change Folder.").*	1846	PM5 only
TKFILE_WRITEFILE_ERR See below; or the USB flash drive could be 'write protected'. *Proposed Error Text:"File Write Error. Is Flash Drive write protected?").*	1847	PM5 only
TKFILE_READFILE_ERR PM5 is trying to access a file on the USB flash driveand for some reason cannot read it. Could be bad flash drive; bad connection; low batteries; or any number of things. Try another stick; fresh batteries; reset the monitor; or update firmware. Or there could be a hardware problem.	1848	PM5 only

Internal Name	Value	Description
Proposed Error Text:"File Read Error.").		
TKFILE_DELETE_ERR *Proposed Error Text:"File Delete error.").*	1849	PM5 only
TKFILE_OPEN_DIR_ERR Could occur if the files or folders on the USB Flash Drive are corrupted. Scott has seen this once when the Concept2/Logbook folder was converted to a file (so the Logbook entry is now a 'file' instead of the expected 'folder'). In this case, send a zip file with the contents of the /Concept2 folder to scotth@concept2.com along with the firmware version in use. Update the firmware; delete the /Concept2/Logbook filereinsert into the PM5 and it will create new Logbook files. Yes, the data is probably gone... *Proposed Error Text:"Error opening folder.").*	1850	PM5 only
TKFILE_FILESYSTEM_SEARCH_ERR	1851	PM5 only
TKFILE_FILE_CLOSE_ERR	1852	PM5 only
TKFILE_FILE_TOOSMALL_ERR	1853	PM5 only
TKFILE_DEVICE_WRITEPROTECTED_ERR	1854	PM5 only
TKSPILOG_TBL_SEARCH_INVALID_ERR	1880	PM5 only
TKSPILOG_WDACCTBLREC_INVALID_ERR Something is messed up in the internal "memory", so you can try the C2 Utility to 'transfer memory to logbook' and it may	1881	PM5 only

Internal Name	Value	Description
offer to repair it. If this does not work and solve the problem, advise Factory Reset.		
TKSPILOG_WDACCTBLREC_CRC_ERR	1882	PM5 only
TKSPILOG_NOTREADY_ERR	1883	PM5 only
TKSPILOG_LOGSTROKEHDR_INVALID_ERR	1884	PM5 only
TKSPILOG_LOGACCTBL_INVALID_ERR	1885	PM5 only
TKSPILOG_LOGDATASTORAGE_FULL_ERR	1886	PM5 only
TKMSDLOG_TBL_SEARCH_INVALID_ERR	1900	PM5 only
TKMSDLOG_WDACCTBLREC_INVALID_ERR	1901	PM5 only
TKMSDLOG_WDACCTBLREC_CRC_ERR	1902	PM5 only
TKMSDLOG_DEVICE_NOTREADY_ERR	1903	PM5 only
TKMSDLOG_VIRTUALADDR_ERR	1904	PM5 only
TKMSDLOG_LOGSTROKEHDR_INVALID_ERR	1905	PM5 only
TKMSDLOG_LOGACCTBL_INVALID_ERR	1906	PM5 only
TKMSDLOG_DEVICELOGHDR_INVALID_ERR It means that the Concept2\Logbook\DeviceLogInfo.bin file header structure is invalid. You should just be able to delete that single file (using a PC), and it should be replaced by the default values (which is fine) and no other log files should be	1907	PM5 only

Internal Name	Value	Description
affected.		
TKMSDLOG_LOGACCTBL_RECORDNUM_ERR	1908	
APFILE_FWUPDATE_ABORTED_ERR Firmware update failed to copy files from USB Stick to the internal memory. Try again, or use USB Cable to udpate the monitor.	1964	PM5 only
APFILE_FILEBACKUP_INVALIDID_ERR	1965	
APFILE_FILEIMAGE_INVALID_ERR	1966	
APFILE_FAVORITES_INVALID_ERR	1967	
APFWUPDATE_INTBUNDLE_IMAGECNT_ERR	1980	
APFWUPDATE_FONT_PROGRAMMING_ERR	1981	
APFWUPDATE_LOADER_PROGRAMMING_ERR	1982	
APFWUPDATE_LDRUPD_PROGRAMMING_ERR	1983	
APFWUPDATE_SCREEN_PROGRAMMING_ERR	1984	
APFWUPDATE_APPINT_PROGRAMMING_ERR	1985	
APFWUPDATE_APPEXT_PROGRAMMING_ERR	1986	
APFWUPDATE_BUNDLE_PROGRAMMING_ERR	1987	
APFWUPDATE_APPANT_PROGRAMMING_ERR	1989	
APFWUPDATE_APPBLE_PROGRAMMING_ERR	1990	
APFWUPDATE_INTBUNDLE_INVALID_ERR	1991	
APFWUPDATE_UPDATEINFO_INVALID_ERR	1992	
APFWUPDATE_UPDATEINFOAPP_INVALID_ERR	1993	
APFWUPDATE_APPNRF_PROGRAMMING_ERR	1994	
APFWUPDATE_SOFTDEVNRF_PROGRAMMING_ERR	1995	
APFWUPDATE_LDRNRF_PROGRAMMING_ERR	1996	
APFWUPDATE_LUPNRF_PROGRAMMING_ERR	1997	
APFWUPDATE_LDRSFE_PROGRAMMING_ERR	1998	

Internal Name	Value	Description
APFWUPDATE_LUPSFE_PROGRAMMING_ERR	1999	
APFWUPDATE_APPSFE_PROGRAMMING_ERR	2000	
APFWUPDATE_DFDATA_PROGRAMMING_ERR	2001	
TKDEBUG_INIT_ERR	2020	
TKDIAGLOG_VIRTUALADDR_ERR	2100	
TKDIAG_TACHDIAG_ADCRESULT_NOTREADY_ERR	2500	
TKDIAG_TACHDIAG_TACHDETECT_ERR	2501	
TKDIAG_TACHDIAG_TACHTEST_ERR	2502	
IOADCONV_BG_TIMEOUT_ERR	810	
IOADCONV_RESET_TIMEOUT_ERR	811	
IOADCONV_INVALID_CHAN_ERR	812	
IOADCONV_NOT_RDY_ERR	813	
IOADCONV_INVALID_REF_ERR	814	
IOADCONV_INIT_ADC_ERR	815	
IODMA_INVALID_MEM_CHAN_ERR	820	
IODMA_INVALID_IO_RQST_CHAN_ERR	821	
IODMA_INIT_DMA_ERR	822	
IODMA_QUEUE_FULL_ERR	823	

Internal Name	Value	Description
IODMA_INVALID_DMA_TYPE	824	
IODMA_DISABLE_TIMEOUT_ERR	825	
IOHDW_MEM_INVALID_CS_ERR	830	
IOHDW_INVALID_DMACLK_ERR	831	
IOHDW_INVALID_SYSCLK_ERR	832	
IOHDW_INVALID_PWRMODE_ERR	833	
IOHDW_MCUCLK_STARTUP_ERR	834	
IOHDW_BKUP_REG_ON_ERR	835	
IOI2C_NOACK_ERR	840	
IOI2C_INIT_WDR_TIMOUT_ERR	841	
IOI2C_INIT_XMIT_TIMOUT_ERR	842	
IOI2C_SEND_XMIT_TIMOUT_ERR	843	
IOI2C_GET_RECV_TIMOUT_ERR	844	
IOI2C_STOP_TIMEOUT_ERR	845	
IOI2C_WDR_TIMOUT_ERR	846	

Internal Name	Value	Description
IOI2C_INVALID_BAUD	847	
IOI2C_INVALID_CHANNEL_ERR	848	
IOI2C_BUSY_ERR	849	
IOLCD_DISPINIT_ERR	860	
IOLCD_INVALIDPARAM_ERR	861	
IOLCD_WRITE_SPI_TIMEOUT_ERR	862	
IOLCD_INVALID_ID_ERR	863	
IOMEM_FLASH_ERASE_TIMEOUT_ERR	870	
IOMEM_FLASH_WRITE_TIMEOUT_ERR	871	
IORTCLOCK_WRITE_TIME_ERR	880	
IORTCLOCK_CRC_ERR	881	
IORTCLOCK_OSC_TIMEOUT_ERR	882	
IORTCLOCK_INIT_ERR	883	
IORTCLOCK_INIT_TIME_ERR	884	
IORTCLOCK_INIT_DATE_ERR	885	
IORTCLOCK_RTCSTOPPED_ERR	886	

Internal Name	Value	Description
IORTCLOCK_LSIOSC_TIMEOUT_ERR	887	
IORTCLOCK_HSEOSC_ERR	888	
IORTCLOCK_RTCUPDATE_TIMEOUT_ERR	889	
IOSCI_INVALID_PORT_ERR	890	
IOSCI_INVALID_BAUD_ERR	891	
IOSCI_INVALID_CNT_ERR	892	
IOSCI_INIT_PORT_ERR	893	
IOSCI_TXOVERRUN_ERR	894	
IOSCI_RXOVERRUN_ERR	895	
IOSCI_RXFRAME_ERR	896	
IOSCI_RXPARITY_ERR	897	
IOSCI_RXBREAK_ERR	898	
IOSCI_PDC_OVERRUN_ERR	899	
IOSCI_INVALID_MODE_ERR	900	
IOTIMER_INVALID_TIMERID_ERR	910	
IOTIMER_INVALID_TIMERRATE_ERR	911	
IOUSER_SEMAPHORE_PEND_ERR	920	

Internal Name	Value	Description
IOUSER_SEMAPHORE_POST_ERR	921	
IOUSB_RST_TIMOUT_ERR	930	
IOUSB_CFG_TIMOUT_ERR	931	
IOUSB_CFG_ENDPT_ERR	932	
IOUSB_SETUP_ERR	933	
IOUSB_FIFO_RD_ERR	934	
IOUSB_NULL_PTR_ERR	935	
IOUSB_BUS_INIT_ERR	936	
IOUSB_TX_BUFFER_ERR	937	
IOUSB_EP_BUSY_ERR	938	
IOUSB_EP_INVALID_ERR	939	
IOUSB_WAKEUP_DISABLE_ERR	940	
IOUSB_BAD_FRAMENUM_ERR	941	
IOUSB_CFG_DEV_ERR	942	
IOUSB_BAD_IFCNUM_ERR	943	

Internal Name	Value	Description
IODIG_INVALID_IN_ERR	950	
IOSPI_WRITE_TIMEOUT_ERR	960	
IOSPI_WRITE_FULL_TIMEOUT_ERR	961	
IOSPI_INVALID_CHANNEL_ERR	962	
IONORFLASH_INIT_ERR	970	
IONORFLASH_WRITE_ERR	971	
IONORFLASH_ERASE_ERR	972	
IONORFLASH_QUERY_ERR	973	
APSMGEN_BUNDLE_STRUCT_INVALID	2040	PM5 BikeErg only
APSMGEN_SFE_DETECT_ERR	2041	PM5 BikeErg only
APSMGEN_INIT_ERR	2042	PM5 BikeErg only
TKDIAGLOG_VIRTUALADDR_ERR	2100	
TKDIAGLOG_TBL_SEARCH_INVALID_ERR	2101	
TKDIAGLOG_LOGACCTBL_INVALID_ERR	2102	
TKDIAGLOG_LOGSTORAGE_FULL_ERR	2103	
TKDIAGLOG_WDACCTBLREC_INVALID_ERR	2104	
TKDIAGLOG_WDACCTBLREC_CRC_ERR	2105	
TKDIAGLOG_INVALID_LOGACCESSTBLINPTR_ERR	2106	
TKDIAGLOG_INVALIDLOGSIZE_ERR	2107	
TKDIAGLOG_LOGENTRYVALIDATE_ERR	2108	
TKDIAGLOG_INSUFFMEMORY_ERR	2109	
TKNRF422DM_NOT_FOUND_ERR	2140	

Internal Name	Value	Description
TKNRF422DM_PEER_LIST_FULL_ERR	2141	
TKNRF422DM_INVALID_DEVINDEX_ERR	2142	
TKNRF422DM_BOND_DATA_READ_ERR	2143	
TKNRF422DM_BOND_MODE_INVALID_ERR	2144	
TKNRF422DM_CLEAR_BOND_TYPE_INVALID_ERR	2145	
TKNRF422DM_WRITE_BOND_DATA_FULL_ERR	2146	
TKNRF422DM_CLEAR_BOND_DATA_ERR	2147	
TKANTBLE_NOT_FOUND_ERR	2200	PM5v2 only:
TKNRF52_ANT_INIT_ERR	2220	PM5v2 only : firmware UPDATING ERROR: After updating some monitors today (and also changing batteries), I found a few monitors that every time they turned on they would for a few moments go into the 'flash loader mode' and then reboot to a 2220 error. Solution: Take batteries out for 1 minute. Put them back in. It will finish a previously unfinished firmware update process and then the problem will go away. Takes another 3-4 minutes to finish.
TKNRF52_INVALID_CMD_ERR	2221	PM5v2 only
TKNRF52_NOT_FOUND_ERR	2222	PM5v2 only
TKNRF52_RSP_TIMEOUT_ERR	2223	PM5v2 only
TKNRF52_INIT_ERR	2224	PM5v2 only
TKNRF52_RADIO_BASE_ERR	2225	PM5v2 only
TKNRF52_RADIO_SDM_ERR	2226	PM5v2 only
TKNRF52_RADIO_SOC_ERR	2227	PM5v2 only

Internal Name	Value	Description
TKNRF52_RADIO_STK_ERR	2228	PM5v2 only
TKNRF52_RADIO_ANT_ERR	2229	PM5v2 only
TKNRF52_LDR_VER_ERR	2230	PM5v2 only
TKNRF52_APP_VER_ERR	2231	PM5v2 only
TKNRF52_NO_CONNECT_ERR	2232	PM5v2 only
APDIAGLOG_LOGRECORD_PENDING_OVERRUN_ERR	2600	
APDIAGLOG_INVALID_LOGRECORDTYPE_ERR	2601	
APDIAGLOG_INVALID_LOGEVENT_ERR	2602	
*** Errors returned by Radios in PM5v1***		
NRF_ERROR_SVC_HANDLER_MISSING	10001	nRF422 only
NRF_ERROR_SOFTDEVICE_NOT_ENABLED	10002	nRF422 only
NRF_ERROR_INTERNAL	10003	nRF422 only
NRF_ERROR_NO_MEM	10004	nRF422 only
NRF_ERROR_NOT_FOUND	10005	nRF422 only
NRF_ERROR_NOT_SUPPORTED	10006	nRF422 only
NRF_ERROR_INVALID_PARAM	10007	nRF422 only
NRF_ERROR_INVALID_STATE	10008	nRF422 only
NRF_ERROR_INVALID_LENGTH	10009	nRF422 only
NRF_ERROR_INVALID_FLAGS	10010	nRF422 only
NRF_ERROR_INVALID_DATA	10011	nRF422 only
NRF_ERROR_DATA_SIZE	10012	nRF422 only
NRF_ERROR_TIMEOUT	10013	nRF422 only
NRF_ERROR_NULL	10014	nRF422 only
NRF_ERROR_FORBIDDEN	10015	nRF422 only

Internal Name	Value	Description
NRF_ERROR_INVALID_ADDR	10016	nRF422 only
NRF_ERROR_BUSY	10017	nRF422 only
NRF_ERROR_SDM_LFCLK_SOURCE_UNKNOWN	14096	nRF422 only
NRF_ERROR_SDM_INCORRECT_INTERRUPT_CONFIGURATION	14097	nRF422 only
NRF_ERROR_SDM_INCORRECT_CLENR0	14098	nRF422 only
NRF_ERROR_SOC_MUTEX_ALREADY_TAKEN	18192	nRF422 only
NRF_ERROR_SOC_NVIC_INTERRUPT_NOT_AVAILABLE	18193	nRF422 only
NRF_ERROR_SOC_NVIC_INTERRUPT_PRIORITY_NOT_ALLOWED	18194	nRF422 only
NRF_ERROR_SOC_NVIC_SHOULD_NOT_RETURN	18195	nRF422 only
NRF_ERROR_SOC_POWER_MODE_UNKNOWN	18196	nRF422 only
NRF_ERROR_SOC_POWER_POF_THRESHOLD_UNKNOWN	18197	nRF422 only
NRF_ERROR_SOC_POWER_OFF_SHOULD_NOT_RETURN	18198	nRF422 only
NRF_ERROR_SOC_RAND_NOT_ENOUGH_VALUES	18199	nRF422 only
NRF_ERROR_SOC_PPI_INVALID_CHANNEL	18200	nRF422 only
NRF_ERROR_SOC_PPI_INVALID_GROUP	18201	nRF422 only
BLE_ERROR_INVALID_CONN_HANDLE	22289	nRF422 only
BLE_ERROR_INVALID_ATTR_HANDLE	22290	nRF422 only
BLE_ERROR_NO_TX_BUFFERS	22291	nRF422 only
NRF_L2CAP_ERR_BASE	22544	nRF422 only
NRF_GAP_ERR_BASE	22800	nRF422 only
NRF_GATTC_ERR_BASE	23056	nRF422 only
NRF_GATTS_ERR_BASE	23312	nRF422 only
*** Errors returned by Radio in PM5v2***		
APRFANT_INVALID_DISCOVER_DEVICE_ERR	10050	PM5v2 only

Internal Name	Value	Description
APRFANT_TX_POWER_ERR	10051	PM5v2 only
APRFANT_HRM_DISCONNECT_ERR	10052	PM5v2 only
APRFANT_FEC_DISCONNECT_ERR	10053	PM5v2 only
APRFANT_C2RACE_DISCONNECT_ERR	10054	PM5v2 only
APRFANT_FE_DISCONNECT_ERR	10055	PM5v2 only
APRFANT_CONNECT_PARAM_ERR	10056	PM5v2 only
APRFANT_INVALID_DEVICETYPE_ERR	10057	PM5v2 only
APRFANT_INVALID_CONNECT_DEVICE_ERR	10058	PM5v2 only
APRFBLE_NOT_FOUND_ERR	10075	PM5v2 only
APRFBLE_LIST_FULL_ERR	10076	PM5v2 only
APRFBLE_INVALID_PARAM_ERR	10077	PM5v2 only
APRFBLE_TX_POWER_ERR	10078	PM5v2 only
APRFBLE_HRM_DISCONNECT_ERR	10079	PM5v2 only
APUTIL_INVALID_IMAGETYPE_ERR	10100	PM5v2 only
APHEADER_INVALIDLDRHDR_ERR	10127	PM5v2 only
APHEADER_INVALIDAPPHDR_ERR	10128	PM5v2 only
TKCMDPR_INVALID_CMD_ERR	10525	PM5v2 only
TKCMDPR_INVALID_CMD_ADDR_ERR	10526	PM5v2 only
TKCMDPR_INVALID_INTF_ERR	10527	PM5v2 only
TKCMDPR_INVALID_DEVTYPE_ERR	10528	PM5v2 only
TKFRAME_CSAFE_FRAME_TOO_LONG_ERR	10550	PM5v2 only
TKFRAME_NOT_FOUND_ERR	10551	PM5v2 only
TKSCI_INVALID_PORT_ERR	10600	PM5v2 only
TKSCI_TX_SEND_ERR	10601	PM5v2 only
TKSCI_RX_TIMEOUT_ERR	10602	PM5v2 only
TKCMDSET_UNKNOWN_CMD_ERR	10650	PM5v2 only
TKCMDSET_NULL_ERR	10651	PM5v2 only
TKCMDSET_INVALID_CMD_PARAM_ERR	10652	PM5v2 only
TKSLIP_INVALID_PORT_ERR	10700	PM5v2 only

Internal Name	Value	Description
TKSLIP_NOT_FOUND_ERR	10701	PM5v2 only
TKSLIP_INVALID_ADDR_ERR	10702	PM5v2 only
TKHCI_INVALID_ADDR_ERR	10725	PM5v2 only
TKHCI_NOT_FOUND_ERR	10726	PM5v2 only
TKERR_ECODE_DISCARDED_ERR	10775	PM5v2 only
TKRF_INVALID_DEVTYPE_ERR	10800	PM5v2 only
TKRF_UNSUPPORTED_DEVTYPE_ERR	10801	PM5v2 only
TKRF_INVALID_SCANMODE_ERR	10802	PM5v2 only
TKRF_INVALID_DISCOVER_DEVICE_ERR	10803	PM5v2 only
TKRF_MSG_OVERRUN_ERR	10804	PM5v2 only
TKRF_FRAME_OVERRUN_ERR	10805	PM5v2 only
TKRF_TX_POWER_ERR	10806	PM5v2 only
TKRF_INVALID_ADVERTISINGMODE_ERR	10807	PM5v2 only
TKRF_NULL_CMDDATA_ERR	10808	PM5v2 only
TKRF_NODATA_AVAIL_ERR	10809	PM5v2 only
TKRFANT_FEC_INVALID_PAGE_ERR	10850	PM5v2 only
TKRFBLEC_HRM_NOT_FOUND_ERR	10860	PM5v2 only
TKRFBLEP_ROW_ADV_LEN_ERR	10870	PM5v2 only
TKRFBLEP_ROW_INVALID_CNT_ERR	10871	PM5v2 only
TKRFBLEP_ROW_RX_OVERRUN_ERR	10872	PM5v2 only
TKRFBLEP_ROW_TX_OVERRUN_ERR	10873	PM5v2 only
TKRFBLEP_ROW_INVALID_STATE_ERR	10874	PM5v2 only
TKRFBLEP_ROW_INVALID_CHARACTERISTIC_ERR	10875	PM5v2 only
TKRFBLEP_ROW_ROWINGDATA_SIZE_ERR	10876	PM5v2 only
TKRFBLEP_ROW_ROWDATA_TX_OVERRUN_ERR	10877	PM5v2 only This error is reported only if an Ergdata connection is active. Can occur if the BLE is being

Internal Name	Value	Description
		compromised by other Wi-Fi interference or the mobile device is going out-of-range.
TKDFU_DATA_SIZE_ERR	10890	PM5v2 only
TKDFU_NOT_SUPPORTED_ERR	10891	PM5v2 only
TKDFU_INVALID_STATE_ERR	10892	PM5v2 only
TKDFU_INVALID_IMAGE_HDR_ERR	10893	PM5v2 only
TKDFU_NULL_ERR	10894	PM5v2 only
TKDFU_INVALID_APP_HDR_ERR	10895	PM5v2 only
TKDFU_INVALID_IMAGE_TYPE_ERR	10896	PM5v2 only
TKDFU_INVALID_IMAGE_INFO_ERR	10897	PM5v2 only
TKRFANT_FE_INVALID_PAGE_ERR	10900	PM5v2 only
TKMEM_INVALID_START_ADDR_ERR	10925	PM5v2 only
TKSTRUCT_MULTI_STRUCT_ERR	10950	PM5v2 only
TKNRF_NOT_FOUND_ERR	10960	PM5v2 only
TKNRF_PEER_LIST_FULL_ERR	10961	PM5v2 only
TKNRF_INVALID_DEVINDEX_ERR	10962	PM5v2 only
TKNRFP_BOND_DATA_READ_ERR	10970	PM5v2 only
TKNRFP_BOND_MODE_INVALID_ERR	10971	PM5v2 only
TKNRFP_CLEAR_BOND_TYPE_INVALID_ERR	10972	PM5v2 only
TKNRFP_WRITE_BOND_DATA_FULL_ERR	10973	PM5v2 only
TKNRFP_CLEAR_BOND_DATA_ERR	10974	PM5v2 only
IOSCI_INVALID_PORT_ERR	11575	PM5v2 only
IOSCI_INVALID_BAUD_ERR	11576	PM5v2 only
IOSCI_INVALID_CNT_ERR	11577	PM5v2 only
IOSCI_INIT_PORT_ERR	11578	PM5v2 only
IOSCI_TXOVERRUN_ERR	11579	PM5v2 only
IOSCI_RXOVERRUN_ERR	11580	PM5v2 only

Internal Name	Value	Description
IOSCI_RXFRAME_ERR	11581	PM5v2 only
IOSCI_RXPARITY_ERR	11582	PM5v2 only
IOSCI_RXBREAK_ERR	11583	PM5v2 only
IOSCI_PDC_OVERRUN_ERR	11584	PM5v2 only
IOSCI_INVALID_MODE_ERR	11585	PM5v2 only
IODIG_INVALID_IN_ERR	11625	PM5v2 only
IOTIMER_INVALID_TIMERID_ERR	11650	PM5v2 only
IOFLASH_ERASE_ERR	11700	PM5v2 only
These errors are related to the radio 3rd party softdevice stack and are unlikely to occur.	>= 12000	PM5v2 only
NRF_ERROR_SVC_HANDLER_MISSING	12001	SVC handler is missing
NRF_ERROR_SOFTDEVICE_NOT_ENABLED	12002	PM5v2 only 3rd party radio stack cannot be enabled. Possible oscillator problem
NRF_ERROR_INTERNAL	12003	Internal Error
NRF_ERROR_NO_MEM	12004	No Memory for operation
NRF_ERROR_NOT_FOUND	12005	Not found
NRF_ERROR_NOT_SUPPORTED	12006	Not supported
NRF_ERROR_INVALID_PARAM	12007	Invalid Parameter
NRF_ERROR_INVALID_STATE	12008	Invalid state
NRF_ERROR_INVALID_LENGTH	12009	Invalid Length
NRF_ERROR_INVALID_FLAGS	12010	Invalid Flags
NRF_ERROR_INVALID_DATA	12011	Invalid Data
NRF_ERROR_DATA_SIZE	12012	Invalid Data size
NRF_ERROR_TIMEOUT	12013	Operation timed out

Internal Name	Value	Description
NRF_ERROR_NULL	12014	Null Pointer
NRF_ERROR_FORBIDDEN	12015	Forbidden Operation
NRF_ERROR_INVALID_ADDR	12016	Bad Memory Address
NRF_ERROR_BUSY	12017	Busy
NRF_ERROR_CONN_COUNT	12018	Maximum connection count exceeded
NRF_ERROR_RESOURCES	12019	Not enough resources for operation

Appendix E

PM State Transitions

For any fixed duration workout or JustRow (no defined end) that is terminated prior to reaching its defined end:

WaitToBegin->WorkoutRow->Terminate (user or command)->Rearm->WaitToBegin

For any fixed duration workout (defined end) that reaches its defined end:

WaitToBegin->WorkoutRow->WorkoutEnd->WorkoutLogged->[Menu button]->WorkoutRearm->WaitToBegin

WaitToBegin->WorkoutRow->WorkoutEnd->WorkoutLogged->[Terminate command]->WaitToBegin

For a fixed distance or fixed calorie interval workout (no defined end) when terminated:

WaitToBegin->IntervalWorkDistance->IntervalWorkDistanceToRest (may not see this state)->IntervalRest->IntervalRestEndToWorkDistance (may not see this state)->IntervalWorkDistance->IntervalWorkDistanceToRest (may not see this state)->IntervalRest->Terminate->Rearm->WaitToBegin

For a fixed time interval workout (no defined end) when terminated:

WaitToBegin->IntervalWorkTime->IntervalWorkTimeToRest (may not see this state)->IntervalRest->IntervalRestEndToWorkTime (may not see this state)->IntervalWorkTime->IntervalWorkTimeToRest (may not see this state)->IntervalRest->Terminate->Rearm->WaitToBegin

For a variable interval workout, with distance and time intervals (defined end), that reaches its defined end:

WaitToBegin->IntervalWorkDistance->IntervalWorkDistanceToRest (may not see this state)->IntervalRest->IntervalRestEndToWorkTime (may not see this state)->IntervalWorkTime->IntervalWorkTimeToRest (may not see this state)->IntervalRest->WorkoutEnd>WorkoutLogged->[Menu button]->WorkoutRearm->WaitToBegin