



SeaPower Battery Firmware Interface Control Document

Document Number: FW-ICD-001

The contents of this document are proprietary to Kraken Robotics and subject to change without notice. Contact Kraken Robotics Inc. for most recent information. Reproduction or transfer of the information contained herein is prohibited without written consent from Kraken Robotics.

© Copyright 2012 - 2025 Kraken Robotics Inc. All Rights Reserved.

Kraken Robotics (Kraken), the Kraken Robotics logo, KATFISH, SeaPower, Sub-Bottom Imager, and Acoustic Corer are among the trademarks or registered trademarks owned by Kraken Robotics Inc. These trademarks and registered trademarks should not be reproduced or used without express written permission from Kraken Robotics Inc. All other brand and product names are or may be trademarks of, and are used to identify products or services of, their respective owners. The elements of this presentation are protected by Canadian and international copyright laws. They should not be reproduced or used without express written permission from Kraken Robotics Inc.

Disclaimer

Kraken should be consulted regarding the proper integration, calibration and configuration of all parts of any Kraken designed system. This document is intended as supplementary information. Any changes to the integration, calibration or configuration must be discussed with Kraken.

Failure to operate this product in a safe and responsible manner could result in injury or damage to the product or other property. This document and companion documents contain instructions for safety and operation of the product. It is essential to read all documentation pertaining to the product and follow all instructions and warnings in said documentation, prior to setup or use, in order to operate the product correctly and avoid damage or injury. Kraken has made every effort to provide clear and accurate information in the documentation, which is provided solely for the user's knowledge. While thought to be accurate, the information in this document is provided strictly "as is" and Kraken will not be held responsible for issues arising from typographical errors or user's interpretation of the language used in this documentation that is different from that intended by Kraken. Kraken reserves the right to revise this document and make changes from time to time without obligation to notify any person of such revisions or changes. In no event shall Kraken, its employees or authorized agents be liable for any damages or losses, direct or indirect, arising from the use of any technical or operational information contained in this document.

Documentation Feedback

To offer suggestions or to report any incorrect or missing information, please contact the documentation department at: documentcontrol@krakenrobotics.com.

Customer Support Contact Information

To get customer support, please contact the Customer Support Group at csg@krakenrobotics.com. Please contact Kraken before sending back any equipment.

To learn more about Kraken Robotics Inc., please visit www.krakenrobotics.com.

1 Notices and Warnings

It is important to be aware that there may be warnings and notices throughout this and any other documents included in this package.



Indicates a potential hazard that may cause injury or caution that may cause damage to equipment.



Indicates a general note.

Table of Contents

1 Notices and Warnings	3
2 Introduction	7
3 Physical interface	8
4 Message Structure	9
4.1 Header	9
4.1.1 Message Length	10
4.1.1.1 Message Length Examples	10
4.2 Payload	10
4.3 Checksum	10
4.3.1 Checksum Examples	11
4.3.2 Checksum C Example	12
5 Request Messages	13
5.1 Handshake Request	13
5.1.1 Handshake Receiver Address	13
5.2 Expanded Handshake Request	14
5.3 Heater	14
5.4 Balancing	14
5.5 State Changes	15
6 Response Messages	16
6.1 Handshake	16
6.2 System Parameters	16

6.3 System Monitoring Data	21
7 System States	26
8 Error Values	27
9 Reading Examples in C	29
9.1 Reading Voltage from Cell 3	29
9.2 Calculating the Checksum	29
9.3 Balancer Flag Array	29
9.5 Firmware Version	30
10 Document Version History	31

Tables

Table 1: Byte Order Example	8
Table 2: Physical Interface Properties	8
Table 3: Request Message Structure	9
Table 4: Header Structure	9
Table 5: Header Byte Definitions	9
Table 6: Message length	10
Table 7: Checksum Format	10
Table 8: Checksum Example Calculation	11
Table 9: Checksum Example 2: Request for Heater Switch On	11
Table 10: Request Message IDs	13
Table 11: Handshake Request Receiver Address	14
Table 12: Expanded Handshake Request Payload	14
Table 13: Heater Request Payload	14
Table 14: Balancer Request Payload	14
Table 15: State Changes	15
Table 16: Response messages	16
Table 17: Response Handshake	16
Table 18: System Parameters	16
Table 19: System Monitoring Data	21
Table 20: System States	26
Table 21: Error Values 1	27
Table 22: Error Values 2	27

2 Introduction

This document describes the Kraken SeaPower™ Battery Management System (BMS) firmware communications interface. The interface described here applies to kraken BMS firmware Version 1.3.0 and higher. If you require a custom interface, contact Kraken. Any requirements outside of those described within the document must be explicitly discussed with Kraken Power GmbH before implementation.

3 Physical interface

The following tables display the communication protocol of the Kraken SeaPower-batteries. Poll and response messages are sent via the RS485 serial connection, and all data is transmitted in little endian form.

Table 1: Byte Order Example

Order of Receiving	Byte n	Byte n+1
Byte Value in Hex (Dec)	0x03(3)	0xE2 (226)
Transmitted Value in Hex (Dec)	0xE203 (57859)	

The standard configuration for the serial communications protocol is as follows:

Table 2: Physical Interface Properties

Serial Com	RS485
Baud Rate	See specific battery ICD
Data bits	8
Parity	none
Stop bits	1

4 Message Structure

All messages follow the same structure. This structure consists of a header, the payload, and a two-byte checksum. The message structure is shown below.

Table 3: Request Message Structure

Byte	Byte 0	:	Byte 9	Byte 10	:	Byte n-3	Byte n-2	Byte n
Content	Header			Payload			Checksum	

4.1 Header

Every message is structured from 16-bit word elements (with some exceptions). Each message contains a header which consists of a start word, the message length, a unique message ID, sender address, receiver address, and the checksum in the end. Every header begins with the start bytes 0x16 0x00 or integer 22.

Table 4: Header Structure

	Start Word		Message Length		Receiver Address		Sender Address		Message ID	
Byte	0	1	2	3	4	5	6	7	8	9
Value	0x16	0x00	AA	BB	CC	DD	EE	FF	GG	HH

Where:

Table 5: Header Byte Definitions

Parameter	Type	Range	Description
AABB	[uint]	0-65535	The number of 16-bit words included between the start word and the checksum. See the Message Length section for further details.
CCDD	[uint]	0-65535	The communication address of the intended receiver of the message.
EEFF	[uint]	0-65535	The communication address of the sending device (8 is reserved and should not be used).
GGHH	[uint]	0-65535	The message ID of the sent message.

4.1.1 Message Length

The message length is an integer which is equal to the number of words to be found between the start byte and the checksum. For example, for a standard header including no payload the message length is equal to 4.

Table 6: Message length

Position		How to Calculate
Byte 2	Byte 3	$(\text{Message Length} * 2) + 4 = \text{number of bytes in message (CS and start byte included)}$ OR $\text{Message Length} = (\text{number of bytes in message (CS and start byte included)} - 4) / 2$

4.1.1.1 Message Length Examples

For a given message, the message length can be calculated as follows:

```
uint16_t messageLength = (sizeof(RXarray) - 4) / 2;
```

Read message length from incoming message can be extracted as follows:

```
uint16_t valueLength = (RXarray[3] << 8) | RXarray[2];
uint16_t RXbytes = (valueLength * 2) + 4;
```

4.2 Payload

Some messages include a payload as part of the message structure. If there is no payload, then the checksum would be found in bytes 10 and 11.

4.3 Checksum

All messages end with a two-byte checksum to ensure transmission validity. The checksum is an integer structured as below.

Table 7: Checksum Format

Position		How to calculate
Byte n-1 XX	Byte n YY	(0xFFFF + 1) – last two lower value (least significant) bytes of $\sum words$
0xYYXX		

4.3.1 Checksum Examples

Table 8: Checksum Example Calculation

	Start Word		Message Length		Receiver		Sender		Message ID	
Byte	0	1	2	3	4	5	6	7	8	9
HEX-Value	0x16	0x00	0x04	0x00	0x00	0x00	0x06	0x00	0x083	0x00
Right Order	0x0016		0x0004		0x0000		0x0006		0x0083	
Sum	0x83									
	Check sum = (0xFFFF + 1) – 0x00A3									
Check Sum	0xFF5D									

Table 9: Checksum Example 2: Request for Heater Switch On

Byte	Values	Words
0	0x16	0x0016
1	0x00	
2	0x05	0x0005
3	0x00	
4	0x00F	0x000F
5	0x00	
6	0x04	0x0004
7	0x00	
8	0x39	0x0039
9	0x00	
10	0xFF	0xFFFF
11	0xFF	

Byte	Values	Words
12	0x9A	0xFF9A
13	0xFF	
	Sum of words (without check sum)	0x10070
	Least significant two bytes of sum of words	0x0070
	Check sum = (FFFF + 1) - 0070	0xFF9A

4.3.2 Checksum C Example

Calculate the checksum as follows:

```
uint32_t sum = 0;
uint16_t j = 0;
for (uint16_t i = 0; i <= (sizeof(RXarray) - 2) / 2; i++) {
    j = i * 2;
    sum = sum + ((RXarray[j + 1] << 8) | RXarray[j]);
}
uint16_t Checksum = (0xFFFF + 1) - sum;
```

5 Request Messages

The host system can request several responses from the battery units. Each request must be made individually. A full request includes the header + the payload + the checksum as below. Some requests require data fields to be present. These requests are outlined in the table below and further specified in later sections of the document.

Table 10: Request Message IDs

Message Sent		Description	Requires Data Field?
Byte 8	Byte 9		
0x04	0x00	Handshake	No
0x68	0x00	Charge request	No
0x69	0x00	Standby request	No
0x70	0x00	Discharge request	No
0x39	0x00	Switch Heater	Yes
0x43	0x00	Switch auto balancing	Yes
0x62	0x00	System reset	No
0x66	0x00	System monitoring data	No
0x67	0x00	System parameters	No

5.1 Handshake Request

When initializing the batteries, the first step is to determine which batteries are available on the communications bus. This is accomplished through the Handshake Request. The host system issues the request and waits for the batteries on the communications bus to respond. The host of the system must wait at least 500ms after issuing a handshake request to the system. This is needed to give all batteries enough time to response to the request.

5.1.1 Handshake Receiver Address

When completing the handshake request the host system must send the receiver value of 0xFFFF to ensure that the battery units on the communications bus respond.

Table 11: Handshake Request Receiver Address

Byte 4	Byte 5
0xFFFF	

5.2 Expanded Handshake Request

If the host system keeps requesting for undetected batteries connected, the already detected batteries should be included in the handshake request. Since the detected batteries are already known to the host, they will not answer again on the handshake request.

Table 12: Expanded Handshake Request Payload

Byte 4	Byte 5	Byte 10	Byte 11	Byte 12	Byte n-4	Byte n-3	Byte n-2
0xFFFF	Communication address of first detected battery		Communication address N x subsequent detected batteries		Communication address of last detected battery		

5.3 Heater

Control the heater for the battery unit by sending the appropriate header with the Heater Request message ID, the receiver address, which is the communication address of the battery being controlled, and the payload as displayed below.

Table 13: Heater Request Payload

Byte 10	Byte 11	Desired State
0xFFFF	0xFFFF	on
0x0000	0x0000	off

5.4 Balancing

Control the balancers for the battery unit by sending the appropriate header with the Balancer Request message ID, the receiver address, which is the communication address of the battery being controlled, and the payload as displayed below.

Table 14: Balancer Request Payload

Byte 10	Byte 11	Desired State
0xFFFF	0xFFFF	on
0x0000	0x0000	off

5.5 State Changes

Put the batteries into various states by sending the appropriate header with the desired state change message ID, and the receiver address, which is the communication address of the battery being controlled. A payload is not required for the state change request message. The available states are listed below for reference.

Table 15: State Changes

Message ID		Representation
Byte 8	Byte 9	
0x71	0x00	Discharge request received
0x72	0x00	Charge request received
0x73	0x00	Standby request received

6 Response Messages

When the battery unit is polled with one of the request messages, it provides a corresponding response message. These messages are structured in the same way as the request.

Table 16: Response messages

Response Message ID		Representation
Byte 8	Byte 9	
0x83	0x00	Handshake
0x72	0x00	Charge ack.
0x73	0x00	Standby ack.
0x71	0x00	Discharge ack.
0xB4	0x00	System monitoring data
0xB1	0x00	System parameters

6.1 Handshake

Table 17: Response Handshake

	Start Word		Message Length		Receiver		Sender		Message Id		Checksum	
Byte	0	1	2	3	4	5	6	7	8	9	10	11
Value	0x16	0x00	0x04	0x00	0x00	0x00	XX	YY	0x83	0x00	XX	YY

Information about the answering battery is stored in the sender word.

6.2 System Parameters

Table 18: System Parameters

Byte	Explanation	Unit
0	Start Byte	
1		

Byte	Explanation	Unit
2	message length	
3		
4	receiver	
5		
6	Communication address	
7		
8	message ID	
9		
10	FW version	major.minor.revision (see example below)
11		
12	Manuf Date: month	
13	Manuf Date: day	
14	Manuf Date: year	
15		
16	Depth rating	{uint}[m]
17		
18	Capacity	{uint}[cAh]
19		
20	Nominal Voltage	{uint}[cV]
21		

Byte	Explanation	Unit
22	Part number	
23	- convert to ASCII - ends with “.”	
24	Example:	
25	BM450V45. Is 0x42 0x4D 0x34	
26	0x35 0x30 0x56 0x34 0x35 0x2E	
27		
28		
29		
30		
31		
32		
33		
34	Critical Low Cell Voltage	{uint}[cV]
35		
36	Low Cell Voltage	{uint}[cV]
37		
38	Critical High Cell Voltage	{uint}[cV]
39		
40	High Cell Voltage	{uint}[cV]
41		
42	Charger voltage low (system	{uint}[cV]
43	voltage)	

Byte	Explanation	Unit
44	Charger voltage high (system voltage)	{uint}[cV]
45		
46	High load current	{uint}[cA]
47		
48	Critical high load current	{uint}[cA]
49		
50	High charge current	{uint}[cA]
51		
52	Critical high charge current	{uint}[cA]
53		
54	Low cell temperature	{uint}[dC°]
55		
56	High cell temperature	{uint}[dC°]
57		
58	Critical high cell temperature	{uint}[dC°]
59		
60	High BMU temperature	{uint}[dC°]
61		
62	Critical high BMU temperature	{uint}[dC°]
63		
64	Serial number	
65		

Byte	Explanation	Unit
66	Nominal charge current	{uint}[cA]
67		
68	Charge complete voltage (battery voltage)	{uint}[cV]
69		
70	Charge complete current	{uint}[cA]
71		
72	Minimum cell temperature for charging	{uint}[dC°]
73		
74	Minimum cell voltage for charging	{uint}[cV]
75		
76	Whether "Maximum Power Stage Temperature" is populated	{bool}
77	Reserved	
78	Lower cell voltage threshold for cell balancing	{uint}[cV]
79		
80	Minimum active discharge level (30% SOC of transport voltage)	{uint}[cV]
81		
82	Reserved	
83		
84	Reserved	
85		
86	Number of cells (Nc)	
87		

Byte	Explanation	Unit
88	Number of modules (Nm)	
89		
90	Number of cell temperature sensors used (Nt)	
91		
92	Number of BMS temperature sensors	
93		
94	Number of onboard temperature sensors per module	
95		
96	Reserved	
97		
98	Check Sum	
99		

6.3 System Monitoring Data

The system monitoring data message length (n) is dependent on the model and configuration of the battery. Specifically, the addresses where the user can find the appropriate values for voltage, temperature, and error are dependent on the number of cells (Nc), the number of modules (Nm) and the number of temperature sensors (Nt).

Table 19: System Monitoring Data

Byte	Explanation	Units
0	start byte	
1		
2	message length	
3		

Byte	Explanation	Units
4	host ID (who did the request)	
5		
6	Communication address	
7		
8	message ID	
9		
10	Reserved	
11		
12	System state	
13		
14	General error	{bool}
15		
16	Error Value	Bit field (see table 21)
17		
18	Error Value2	Bit field (see table 22)
19		
20	System voltage	{uint}[cV]
21		
22	Battery voltage	{uint}[cV]
23		
24	Current (signed)	{int}[cA]
25		

Byte	Explanation	Units
26	Heater status (0 = off; 1 = on)	
27		
28	Auto balancing status (0 = off; 1 = on)	
29		
30	Reserved	
31		
32	BMU temperature	{int}[dC°]
33		
34	Maximum Power Stage Temperature Note: See byte 76 of the System Parameter Response to determine if the value is present. If no sensors are present, the value of this field will be 0.	{int}[C°]
35		
36	State of charge (SOC)	{uint} [c%]
37		
38	Coulomb counter	{int} [mAh] (32 bit)
39		
40		
41		
42	Reserved	
43		
44	Reserved	
45		
46	Reserved	
47		

Byte	Explanation	Units
48	Reserved	
49		
50	Voltage cell: 1	{uint} [mV]
51		
to		
$50 + (2 * N_c) - 2$	Voltage cell: Number of cells	{uint} [mV]
$50 + (2 * N_c) - 1$		
$50 + (2 * N_c)$	Cell temperature: 1 (signed)	{int}[C°]
$50 + (2 * N_c) + 1$		
to		
$50 + (2 * N_c) + (2 * N_t) - 2$	Cell temperature: Value of byte 90 and 91 in parameter msg	{int}[C°]
$50 + (2 * N_c) + (2 * N_t) - 1$		
$50 + (2 * N_c) + (2 * N_t)$	Balancer statuses from cells of the first module (binary flag array): The Number of cells least significant bits symbolise the balancers. Where 0 is disabled and 1 enabled.	
$50 + (2 * N_c) + (2 * N_t) + 1$		
to (Only if Number of modules > 1)		
$50 + (2 * N_c) + (2 * N_t) + (N_m * 2) - 2$	Balancer statuses from cells of the n-th module (binary flag array): The Number of cells least significant bits symbolize the balancers. Where 0 is disabled and 1 enabled.	
$50 + (2 * N_c) + (2 * N_t) + (N_m * 2) - 1$		
$50 + (2 * N_c) + (2 * N_t) + (N_m * 2)$	Check sum	
$50 + (2 * N_c) + (2 * N_t) + (N_m * 2) + 1$		

The length of the message depends on the number of cells and the number of modules (slave boards). Cell voltages start at byte 50 and cell temperatures are listed directly after the voltages, while the balancer status comes last after the cell temperatures (6 temperatures per module).

Example

Number of cells: 36

Number of modules: 3

Byte #50 to #121 contain the cell voltages

Byte #122 to #157 contain the cell temperatures

Byte #158 to #163 contain the balancer statuses

Byte #164 to #165 contain the check sum

7 System States

Table 20: System States

Values		System State
Byte 12	Byte 13	
0x00	0x00	UNINITIALIZED
0x01	0x00	INITIALIZATION
0x02	0x00	INITIALIZED
0x03	0x00	IDLE
0x04	0x00	STANDBY
0x05	0x00	PRECHARGE
0x06	0x00	DISCHARGE
0x07	0x00	CHARGE_PRECHARGE
0x08	0x00	CHARGE
0x14	0x00	UNDEFINED
0xF0	0x00	ERROR

The battery goes automatically into standby mode after startup. From standby mode the battery can be set to charge mode or discharge mode. That can be done via a state request. From both states the battery can be set back into standby mode on request.

8 Error Values

Table 21: Error Values 1

Bit	Errors Value 1
0	Current sensor not responding
1	Charge relay defect
2	Discharge relay defect
3	Reserved
4	Slave module SPI error
5	Precharge error
6	Dead short error
7	Reserved
8	Over charge current detected
9	Over discharge current detected
10	Over cell voltage detected
11	Under cell voltage detected
12	Over discharge temperature detected
13	Under discharge temperature detected
14	Over charge temperature detected
15	Under charge temperature detected

Table 22: Error Values 2

Bit	Errors Value 2
0	Reserved
1	Reserved
2	Reserved

Bit	Errors Value 2
3	Reserved
4	Reserved
5	Hardware Setup Error
6	Reserved
7	Reserved
8	Reserved
9	Reserved
10	Reserved
11	Reserved
12	Reserved
13	Reserved
14	Reserved
15	Reserved

9 Reading Examples in C

9.1 Reading Voltage from Cell 3

```
uint16_t voltageCell3 = (RXarray[55] << 8) | RXarray[54];
```

9.2 Calculating the Checksum

```
uint32_t sum = 0;
uint16_t j = 0;
for (uint16_t i = 0; i <= (sizeof(RXarray) - 2) / 2; i++) {
    j = i * 2;
    sum = sum + ((RXarray[j + 1] << 8) | RXarray[j]);
}
uint16_t Checksum = (0xFFFF + 1) - sum;
```

9.3 Balancer Flag Array

```
uint8_t i;
uint16_t balancerValue = (RXarray[87] << 8) | RXarray[86];

for ( i = 0; i <= 11; i++) {

    uint8_t bit = (balancerValue >> i) & 0x01;

    if (bit == 1)
    {
        printf("balancer %d is set \n", i + 1);
    }
    else
    {
        printf("balancer not set \n");
    }
}
```

9.5 Firmware Version

```
uint16_t SW_version = (RXarray[11] << 8) | RXarray[10];

uint16_t Major = SW_Version >> 11;
uint16_t Minor = SW_Version >> 6 & 0x001F;
uint16_t Revision = SW_Version & 0x003F;

printf("%d \n", Major);
printf("%d \n", Minor);
printf("%d \n", Revision);
```

10 Document Version History

Document: FW-ICD-001		
V13	August 2025	Updated the following bytes in the System Parameters table: • Byte 76: Explanation field updated to "Whether maximum power stage temperature is populated" Unit field updated to "{bool}." • Bytes 92-93: Explanation field updated to "Number of BMS temperature sensors". • Bytes 94-95: Explanation field updated to " Number of on board slave temperature sensors per module". Updated bytes 34-35 of the System Monitoring Data table to "Maximum power stage sensors temperature" in the Explanation field, and {int}[C°] in the Unit field.
V12	March 2025	Updated firmware content descriptions to include communications address.
V11	February 2025	Updated bytes 64-65 in System Parameters table from "Reserved" to "Serial Number".
V10	October 2024	Corrections made to the System Parameters Table
V9	July 2024	• Typographical error corrected in Table 16 • Corrected Error Values Tables
V8	August 2023	• Added error for incorrect hardware assembly • Added error for relay temperature error • Removed unused error codes • Updated code example snippets
V7	May 2023	• Added detail to Table 5: Header Byte Definitions • Added detail to: Handshake Request • Added a field in requests table to indicate if data field is required • Added error flag to Table 22: IVTS connection error • Updated Introduction
V6	November 2021	• Precise Error description in table 19 bit 14 to 19 • Updated copyright and contact information

V5	June 2021	• Added coulomb counter value in byte 38 – 41 of system monitoring data • Added message ID for CC reset in “Request Messages” table • Valid from FW Version 1.1.0 onwards
V4	May 2021	• Changed byte 80/81 of parameter message from % to cV • New parameter in system parameter message, byte 90/91 give amount of temperature sensors. This involves changes in the composition of the system monitoring data message. • Changed unit of SOC in monitoring message to [c%] • Updated copyright and contact information
V3	June 2020	• The PRECHARGE error was added to the System States and Error Values sections.
V2	April 2020	Updates were made to the following sections: • Checksum calculation examples • Balancer information • Monitoring messaging to correct byte assigning for cell voltages, cell temperatures and balancers
V1	March 2020	Initial release