# BMS Communication Protocol Specification

| | |
|---|---|
| **Version** | 0.6 |
| **Status** | draft |
| **Date** | 24-Feb-2022 |

# TABLE OF CONTENTS

# 1. Document Details

## 1.1. Revision & Approval History

| Version | Author | | Reviewer | | Approver | |
|---|---|---|---|---|---|---|
| | Name | Date (DD-MM-YYYY) | Name | Date (DD-MMM-YYYY) | Name | Date (DD-MM-YYYY) |
| Draft 0.1 | Ashvin Ramani | 18-Nov-2021 | Tejendra Joshi | | | |
| Draft 0.2 | Srikanth Reddy Ramidi | 23-Nov-2021 | Tejendra Joshi | | | |
| Draft 0.3 | Srikanth Reddy Ramidi | 21-Dec-2021 | Tejendra Joshi | | | |
| Draft 0.4 | Srikanth Reddy Ramidi | 19-Jan-2022 | Tejendra Joshi | | | |
| Draft 0.5 | Srikanth Reddy Ramidi | 28-Jan-2022 | Tejendra Joshi | | | |
| Draft 0.6 | Ashvin Ramani | 24-Feb-2022 | Tejendra Joshi | | | |

| Version | Description of Change |
|---|---|
| Draft 0.1 | Newly Created |

| Version | Description of Change |
|---|---|
| Draft 0.2 | Changes made as per internal review comments |
| Draft 0.3 | Fault detection command and GUI start commands are added |
| Draft 0.4 | Updated with all the board level as well as System level faults in Fault detection message |
| Draft 0.5 | Updated document with connect and disconnect messages. |
| Draft 0.6 | Updated all commands to support maximum number of slaves up to 128 and re-arranged Opcode for all commands in sequential manner. |

## 1.2. Definition, Acronyms and Abbreviations

| Definition/Acronym/Abbreviation | Description |
|---|---|
| BMS | Battery Management System |
| GUI | Graphical User Interface |
| MCU | Microcontroller |
| UART | Universal Asynchronous Receiver-Transmitter |
| SPI | Serial Peripheral Interface |
| USB | Universal Serial Bus |
| PC | Personal Computer |

## 1.3. References

| No. | Document | Version | Remarks |
|---|---|---|---|
| 1 | BMS GUIs.pptx | 1.0 | Presentation shared by ADI team |
| 2 | adi_adbms-r_gui-rel2.4.exe | 2.4 | Referred by ADI |

# 2. Introduction

## 2.1. Purpose of the Document

The purpose of the document is to describe the Communication Protocol intended to be used in development of BMS Master board and GUI.

## 2.2. Intended Audience

This document is intended for

- Developers
- Project team members
- Testers

# 3. Protocol Description

## 3.1. General Principles

The protocol is designed to serve as a point-to-point single master, single slave communication channel between GUI and BMS, where master is GUI and slave is BMS.

The Master/Slave relationship implies that it is the Master that is responsible for initiating data transfers between the subsystems. The Slave can't initiate the transfer by itself.

The protocol doesn't support multi-master, master-master, slave-slave, or multi-slave relationship; there should always be a single master and single slave.

**Note:** The data transfer between slave (BMS) and master (GUI) is in big endian manner i.e.., MSB is sent/received first and LSB will be sent/received last.

## 3.2. Communication Principles

The Communication Protocol consists of the following three layers:

- Physical
- Transport
- Application

The Physical Layer consisting of electrical wiring, which are designed to support electrical signalling as per common UART protocol. This layer is context-agnostic.

The Transport layer is designed to ensure that GUI-BMS handshaking and package deliveries are done in correct manner so that the data corruption can be easily detected. This layer is context-agnostic.

The Application Layer is designed to ensure error-free messaging between GUI and BMS applications. This layer is context aware.

As this protocol is point-to-point one, there are no broadcasts or multipoint messages. The relationship between nodes is strictly Master-Slave one. The role reversal is not allowed. This means that the Slave node will only talk when requested to do so by Master node.

The master can request to the slave to continuous reading of the data, where slave will periodically transfer requested data to the master.

The slave can accept single command at a time, and it will reject any command from master if they are processing previously received command.

## 3.3. Physical Layer

The physical layer is implemented in hardware (UART/USB) and not subject for software implementation.

The settings for the physical layer shall be:

1. Speed: 115200 baud rate (UART) / 9600, 115200 baud rates (USB)
2. No parity
3. Data bits: 8
4. Stop bits: 1
5. Optional hardware flow control (may be disabled in software)

## 3.4. Transport Layer

### 3.4.1. General

The Transport Layer is responsible for low-level communication handshaking and link provision. At this layer the packets are analysed for correctness.

Note: The Transport Layer doesn't do any payload inspection; this is the function of Application Layer.

The Transport Layer processing software shall check the validity of the packet and respond with status whenever error occurs, this giving to Sender a clear reason why the packet will not be passed onto Application Layer of the receiver.

### 3.4.2. Transport Layer Packet Structure

The Transport Layer packet consists of the following fields:

| Field Name | Description | Length | Typical Value |
|---|---|---|---|
| SOF | Start of Frame.<br>*Uses unique pattern in Hex: Ascii Value of "BMS"* | 3 bytes | 0x42, 0x4D, 0x53 |
| ML | Message Length (Excluding SOF) | 2 bytes | 0x0000 – 0xFFFF |
| MT | Message Types | 1 byte | 0x00 – Reserved<br>0x01 – Command<br>0x02 - Response<br>0x03 - 0xFF - Reserved |
| MP | Message Payload<br>*This field is extracted from the message and passed to Application Layer. It includes the command or response payload based on MT. The Transport Layer does not inspect the contents of Payload* | Defined by CL or RL based on MT field | 0x00 – 0xFF per byte |

| | | | |
|---|---|---|---|
| Checksum | Checksum<br>*The checksum is used to ensure the correctness of the Transport Layer Packet*<br>*The algorithm is 16-bit 2's complement calculated on whole message*<br>*Refer section **4.3*** | 2 bytes | 0x0000 – 0xFFFF |

## 3.5. Application Layer

### 3.5.1. General

The Application Layer is responsible for parsing the Payload packet and pass it to Application. Once Application decided to respond to the packet, the Application Layer shall prepare the response packet and pass it onto Transport Layer.

### 3.5.2. Application Layer Packet Structure

The Application Layer consists of the following command or response packet fields:

#### *3.5.2.1. Command Packet Fields*

| Field Name | Description | Length | Typical Value |
|---|---|---|---|
| CL | Command length | 2 bytes | 0x0000 – 0xFFFF |
| Opcode | Operation code<br>*This code can be used to directly map the Application processing functions to a requested operation* | 1 byte | 0x00 – Reserved<br>0x01 – Connect<br>0x02 – Disconnect<br>0x03 – Configuration<br>0x04 – Fault detection<br>0x05 – Start Measurement<br>0x06 – 0x0A – Reserved<br>0x0B – Read<br>0x0C – Write<br>0x0D – 0xFF – Reserved |
| ICC | IC Count<br>*Note: This field is applicable for All Opcodes except connect and disconnect (0x01 and 0x02)* | 1 byte | 0x00 – Reserved<br>0x01 – 0x78<br>0x79 – 0xFF – Reserved |
| ICN | IC number | 16 bytes | 0x00 – 0xFF per byte |

| | | | |
|---|---|---|---|
| | *The size of this field is 16 bytes array, where each byte can hold up to 8 ICs request, where each bit represents requested operation need to perform or not on the respective IC in the accordance to IC Count*<br><br>*eg:*<br>*1)0x00000000000000000000000000000001 – Requested operation on IC number 1*<br>*2)0x00000000000000000000000000000003 – Requested operation on IC number 1 & 2*<br>*2)0x000000000000000000000000000004FF –Requested operation on IC number 1 to 8 and 11*<br><br>*Note: This field is applicable for All Opcodes except connect and disconnect (0x01 and 0x02)* | | |
| ICT | IC types<br>*The size of this field is 128 bytes array, where each byte is representing IC types according to IC number*<br><br>*Note: This field is only applicable for Opcode 0x03 (Configuration command)* | 128 bytes | Per byte:<br>0x00 – Reserved<br>0x01 – ADBMS1818<br>0x02 – ADBMS1816<br>0x03 – 0xFF – Reserved |
| Optypes | Operation types | 1 byte | 0x00 – Reserved<br>0x01 – One-Shot<br>0x02 – Continuous<br>0x03 – Stop<br>0x04 – 0xFF – Reserved |
| DL | Data length | 1 byte | 0x00 – 0xFF |
| Data | Data passed to Application Firmware by GUI.<br>*NOTE: All the ADBMS commands are handled through read/write (Opcode) command only* | Variable, see DL | 0x00 – 0xFF per byte |

#### *3.5.2.2. Response Packet Fields*

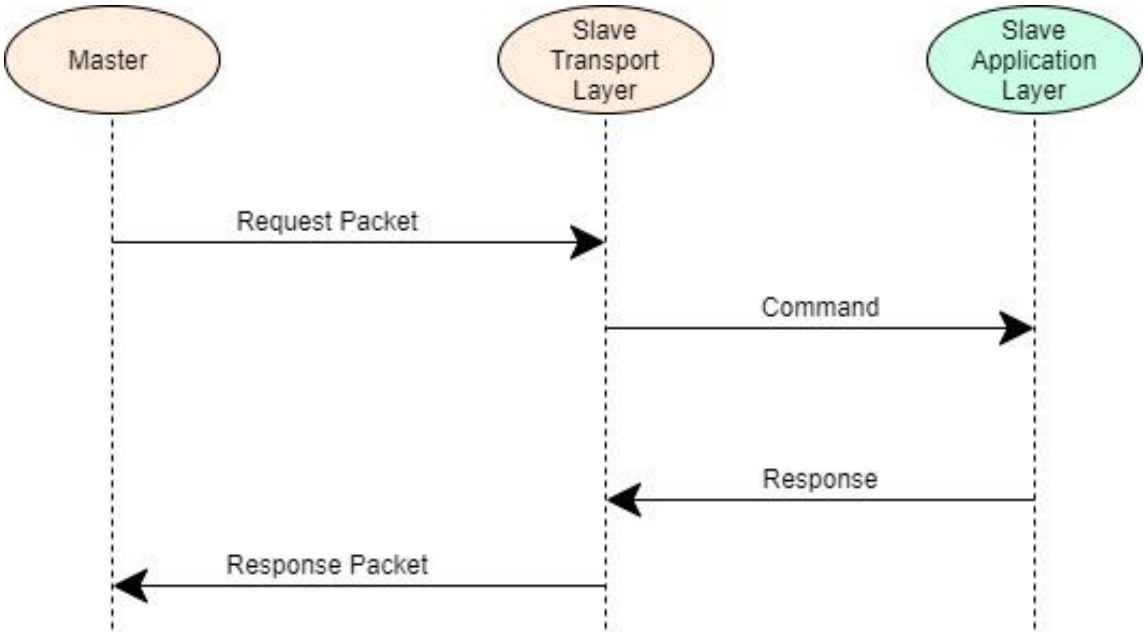
| Field Name | Description | Length | Typical Value |
|---|---|---|---|
| RL | Response length | 2 bytes | 0x0000 – 0xFFFF |
| Opcode | Operation code<br>*This field must be same as command`s Opcode* | 1 byte | 0x00 – Reserved<br>0x01 – Connect<br>0x02 – Disconnect<br>0x03 – Configuration<br>0x04 – Fault detection<br>0x05 – Start Measurement<br>0x06 – 0x0A – Reserved<br>0x0B – Read<br>0x0C – Write<br>0x0D – 0xFF – Reserved |
| ICN | IC number<br>*Each bit represents the response of which ICs*<br>*Note: At a time only one IC`s response can be received for Opcode: 0x05(GUI Start), 0x0B (Read) & 0x0C (Write).*<br>*eg :*<br>*1)0x00000000000000000000000000000001 – Response of IC number 1*<br>*2)0x00000000000000000000000000000004 – Response of IC number 3*<br>*3)0x00000000000000000000000000000800 – Response of IC number 12* | 16 bytes | 0x01 – 0x80 per byte |
| Status | Response status. | 1 byte | See section 3.8 |
| DL | Data length | 1 byte | 0x00 – 0xFF |
| Data | Data passed to GUI by Application Firmware | Variable, see DL | 0x00-0xFF per byte |

## 3.6. Communication Flow

The protocol is based on Request - Response relationship. This means that all packets sent over Transport Layer shall be responded with the response packet.

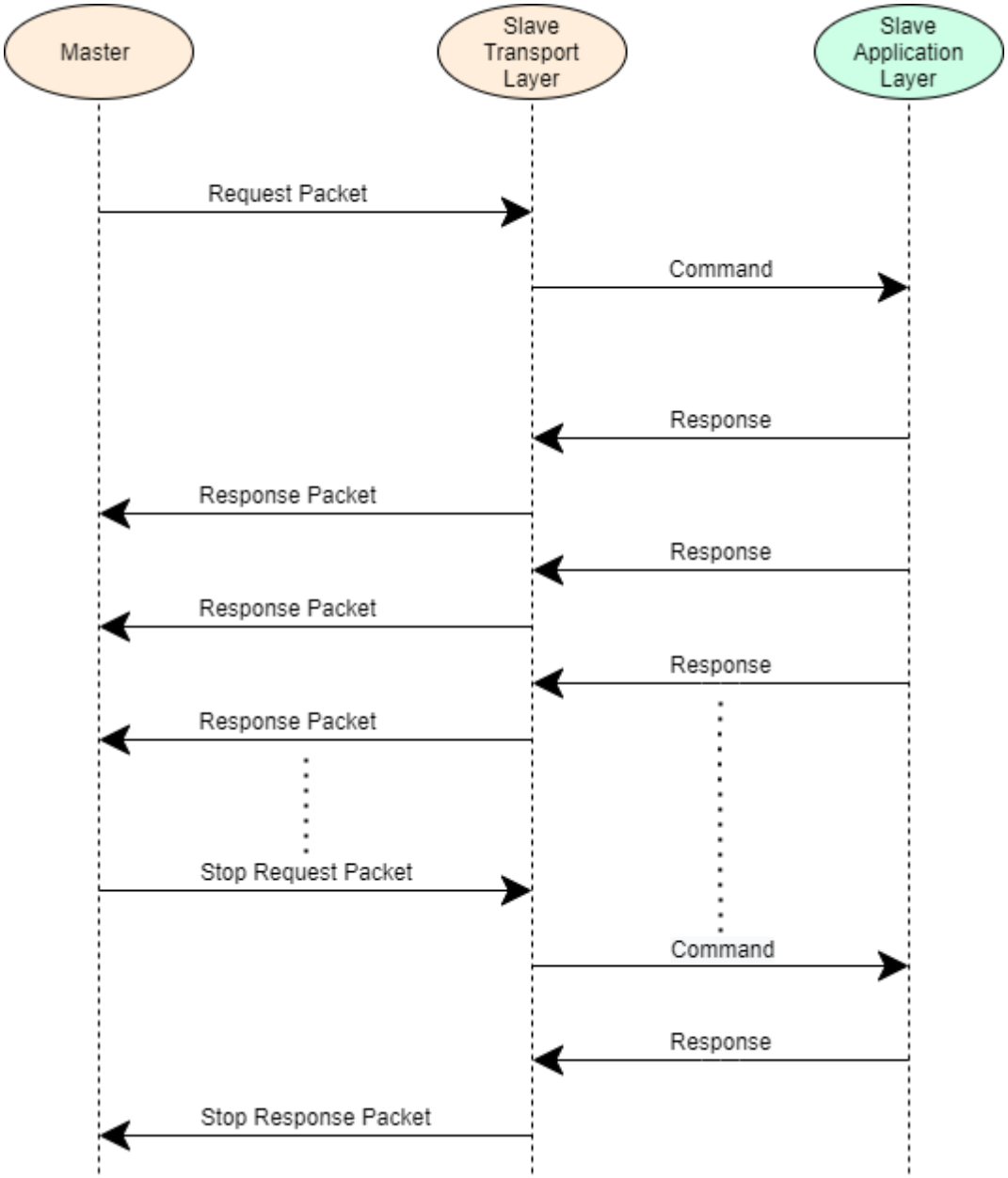
### 3.6.1. One Shot Communication

Below is the example of successful transaction, in which Master issues a command to Slave, then Slave responds with an answer.




### 3.6.2. Continuous Communication

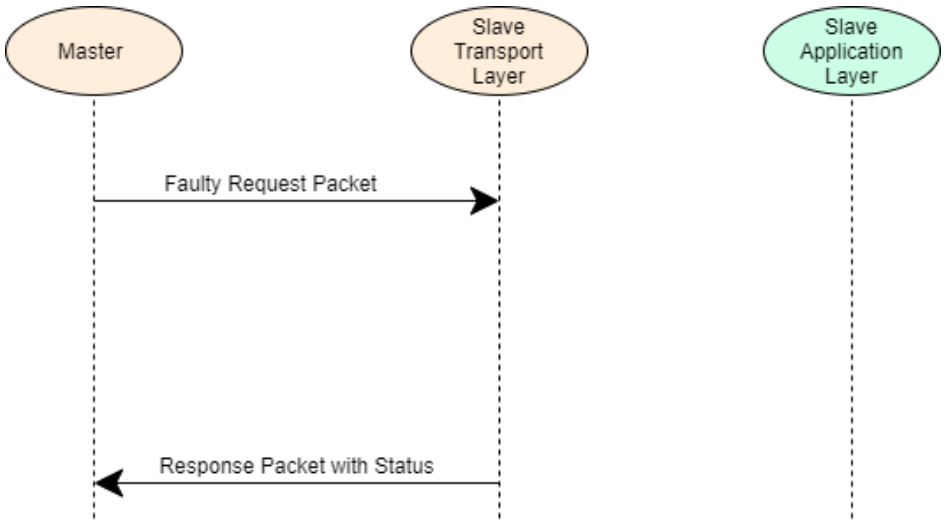
In the example below, Master issues a single command to take multiple reading and then Slave responds with an answer with multiple response packets periodically based on predefined interval. To stop this, Master will again issue a stop command.




### 3.6.3. Faulty Communication

In the example below, the Slave's Transport Layer rejected the packet with appropriate response code because of having some invalids contain inside the packet.

If a packet is rejected by Transport Layer, the Application Layer won't be aware of it.




## 3.7. List of Messages

### 3.7.1. General

Below is the complete list of Messages:

| Opcode | Message Description | Section |
|---|---|---|
| 0x00 | Reserved | N/A |
| 0x01 | Connect | **3.7.2** |
| 0x02 | Disconnect | **3.7.2** |
| 0x03 | Configuration | **3.7.3** |
| 0x04 | Fault Detection | **3.7.4** |
| 0x05 | Start Measurement | **3.7.5** |
| 0x06 – 0x0A | Reserved for Special Command | N/A |
| 0x0B | Read (RAW Command) | **3.7.6** |
| 0x0C | Write (RAW Command) | **3.7.7** |
| 0x0D – 0xFF | Reserved | N/A |

### 3.7.2. Connect/Disconnect Message

#### *3.7.2.1. General*

The Connect/Disconnect message shall be used by GUI to BMS. This command must be issued before any other commands.

#### 3.7.2.2. Master (GUI) to Slave (BMS) Command packet

| Byte | Field | Value |
|---|---|---|
| 0 - 1 | CL (Excluding this field) | 0x0003 |
| 2 | Opcode | 0x01 – Connect<br>0x02 – Disconnect |
| 3 | Optypes | 0x01 – One-Shot |
| 4 | DL | 0x00 |

#### 3.7.2.3. Slave (BMS) to Master (GUI) Response packet

| Byte | Field | Value |
|---|---|---|
| 0 - 1 | RL (Excluding this field) | 0x0003 |
| 2 | Opcode | 0x01 – Connect<br>0x02 – Disconnect |
| 3 | Response Status | Response Code |
| 4 | DL | 0x00 |

### 3.7.3. Configuration Message

#### 3.7.3.1. General

The Configuration message shall be used by GUI to Configure the important settings on BMS. This command must be issued before any other commands.

#### 3.7.3.2. Master (GUI) to Slave (BMS) Command packet

| Byte | Field | Value |
|---|---|---|
| 0 - 1 | CL (Excluding this field) | 0x009B |
| 2 | Opcode | 0x03 – Configuration |
| 3 | IC Count<br>*Total Number of ICs Connected with BMS* | 0x01 – 0x80 |

| | | |
|---|---|---|
| 4 – 19 | IC number<br>*The size of this field is 16 bytes array, where each byte can hold up to 8 ICs request, where each bit represents requested operation need to perform or not on the respective IC in the accordance to IC Count* | 0x00000000000000000000000000000001 – 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF |
| 20 – 147 | IC types<br>*The size of this field is 128 bytes array, where each byte is representing IC types according to IC number* | Per byte:<br>0x01 – ADBMS1818<br>0x02 – ADBMS1816 |
| 148 | Optypes | 0x01 – One-Shot |
| 149 | DL | 0x07 |
| 150-156 | Data (7 bytes)<br>Byte-0,1: Time interval for sending voltage reading response to GUI.<br>➔Time interval is in MS:<br>Ex: 10 ms -> 0x000A,<br>1 sec -> 0x03EB<br><br>Byte-2,3: Cell Under Voltage Threshold.<br>Byte-4,5: Cell Over Voltage Threshold.<br>➔Voltage Threshold should be like value multiplied by 10000:<br>Ex: 3.1V -> 31000 -> 0x7918,<br>5V -> 50000 ->0xC350<br><br>Byte-6: Fault Group (Default: 0x1F)<br>➔Each bit represents respective fault group is enable or disable as per following:<br>BIT 0: Cell UV/OV,<br>BIT 1: GPIO UV/OV, | 0x00-0xFF per byte |

| | | |
|---|---|---|
| | BIT 2: Other UV/OV,<br>BIT 3: Cell OW,<br>BIT 4: System Fault,<br>BIT 5 – BIT 7: Reserved<br>Where, 0: Disabled, 1: Enabled | |

#### *3.7.3.3. Slave (BMS) to Master (GUI) Response packet*

| Byte | Field | Value |
|---|---|---|
| 0 - 1 | RL (Excluding this field) | 0x0013 |
| 2 | Opcode | 0x03 – Configuration |
| 3 – 18 | IC number | Same value IC number<br>will be return from command |
| 19 | Response Status | Response Code |
| 20 | DL | 0x00 |

### 3.7.4. Fault Detection Message

#### *3.7.4.1. General*

The Fault Detection message shall be used by GUI to get the faults of any board that are connected to BMS.

#### *3.7.4.2. Master (GUI) to Slave (BMS) Command packet*

| Byte | Field | Value |
|---|---|---|
| 0 - 1 | CL | 0x0016 |
| 2 | Opcode | 0x04 - Fault Detection |
| 3 | IC Count<br>*Total Number of ICs want to write* | 0x01 – 0x80 |
| 4 – 19 | IC number<br>*The size of this field is 16 bytes array, where each byte can hold up to 8 ICs request, where each bit represents requested operation need to perform or not on the respective IC in the* | 0x00000000000000000000000000000001 – 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF |

| | | |
|---|---|---|
| | *accordance to IC Count* | |
| 20 | Optypes | 0x01 – One-Shot<br>0x02 – Continuous<br>0x03 – Stop |
| 21 | DL | 0x02 |
| 22 – 23 | DATA<br>Byte-0,1: Time interval (in MS) for sending Fault detection response to GUI. | 0x00-0xFF per byte |

#### *3.7.4.3. Slave (BMS) to Master (GUI) Response packet*

| Byte | Field | Value |
|---|---|---|
| 0 - 1 | RL | 0x003B |
| 2 | Opcode | 0x04 – Fault Detection |
| 3 – 18 | IC number<br>*Each bit represents the response of which ICs*<br>*Note: At a time only one IC`s response can be received* | 0x00000000000000000000000000000001 –<br>0x80000000000000000000000000000000 |
| 19 | Status | Response Code |
| 20 | DL | 0x28 |
| 21 – 60 | DATA:<br>Data structure will be as shown in section **3.7.4.4** | 0x00-0xFF per byte |

#### *3.7.4.4. Fault detection response's data structure*

| Cell OV/UV Fault | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **BYTE** | **bit7** | **bit6** | **bit5** | **bit4** | **bit3** | **bit2** | **bit1** | **bit0** |
| 0 | $C_4OV$ | $C_4UV$ | $C_3OV$ | $C_3UV$ | $C_2OV$ | $C_2UV$ | $C_1OV$ | $C_1UV$ |
| 1 | $C_8OV$ | $C_8UV$ | $C_7OV$ | $C_7UV$ | $C_6OV$ | $C_6UV$ | $C_5OV$ | $C_5UV$ |
| 2 | $C_{12}OV$ | $C_{12}UV$ | $C_{11}OV$ | $C_{11}UV$ | $C_{10}OV$ | $C_{10}UV$ | $C_9OV$ | $C_9UV$ |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 3 | $C_{16}OV$ | $C_{16}UV$ | $C_{15}OV$ | $C_{15}UV$ | $C_{14}OV$ | $C_{14}UV$ | $C_{13}OV$ | $C_{13}UV$ |
| 4 | X | X | X | X | $C_{18}UV$ | $C_{18}OV$ | $C_{17}UV$ | $C_{17}OV$ |
| 5 | X | X | X | X | X | X | X | X |
| 6 | X | X | X | X | X | X | X | X |
| 7 | X | X | X | X | X | X | X | X |

| GPIO OV/UV Fault | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **BYTE** | **bit7** | **bit6** | **bit5** | **bit4** | **bit3** | **bit2** | **bit1** | **bit0** |
| 8 | $G_4OV$ | $G_4UV$ | $G_3OV$ | $G_3UV$ | $G_2OV$ | $G_2UV$ | $G_1OV$ | $G_1UV$ |
| 9 | $G_8OV$ | $G_8UV$ | $G_7OV$ | $G_7UV$ | $G_6OV$ | $G_6UV$ | $G_5OV$ | $G_5UV$ |
| 10 | X | X | X | X | X | X | $G_9OV$ | $G_9UV$ |
| 11 | X | X | X | X | X | X | X | X |
| 12 | X | X | X | X | X | X | X | X |
| 13 | X | X | X | X | X | X | X | X |
| 14 | X | X | X | X | X | X | X | X |
| 15 | X | X | X | X | X | X | X | X |

| Other OV/UV Fault | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **BYTE** | **bit7** | **bit6** | **bit5** | **bit4** | **bit3** | **bit2** | **bit1** | **bit0** |
| 16 | DIE UT | DIE OT | STK OV | STK UV | $V_D$ OV | $V_D$ UV | $V_A$ OV | $V_A$ UV |
| 17 | X | X | X | X | X | X | X | X |
| 18 | X | X | X | X | X | X | X | X |
| 19 | X | X | X | X | X | X | X | X |
| 20 | X | X | X | X | X | X | X | X |
| 21 | X | X | X | X | X | X | X | X |
| 22 | X | X | X | X | X | X | X | X |
| 23 | X | X | X | X | X | X | X | X |

| Cell OW Fault | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **BYTE** | **bit7** | **bit6** | **bit5** | **bit4** | **bit3** | **bit2** | **bit1** | **bit0** |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 24 | $C_8OW$ | $C_7OW$ | $C_6OW$ | $C_5OW$ | $C_4OW$ | $C_3OW$ | $C_2OW$ | $C_1OW$ |
| 25 | $C_{16}OW$ | $C_{15}OW$ | $C_{14}OW$ | $C_{13}OW$ | $C_{12}OW$ | $C_{11}OW$ | $C_{10}OW$ | $C_9OW$ |
| 26 | X | X | X | X | X | X | $C_{18}OW$ | $C_{17}OW$ |
| 27 | X | X | X | X | X | X | X | X |
| 28 | X | X | X | X | X | X | X | X |
| 29 | X | X | X | X | X | X | X | X |
| 30 | X | X | X | X | X | X | X | X |
| 31 | X | X | X | X | X | X | X | X |

| System Faults | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **BYTE** | **bit7** | **bit6** | **bit5** | **bit4** | **bit3** | **bit2** | **bit1** | **bit0** |
| 32 | X | X | X | X | X | X | AFE COM | SPI fail |
| 33 | X | X | X | X | X | X | X | X |
| 34 | X | X | X | X | X | X | X | X |
| 35 | X | X | X | X | X | X | X | X |
| 36 | X | X | X | X | X | X | X | X |
| 37 | X | X | X | X | X | X | X | X |
| 38 | X | X | X | X | X | X | X | X |
| 39 | X | X | X | X | X | X | X | X |

| Bit Field Descriptions | |
|---|---|
| **Bit** | **Description** |
| $C_NOV$ | $N^{th}$ cell Over Voltage flag |
| $C_NUV$ | $N^{th}$ cell Under Voltage flag |
| $G_NOV$ | $N^{th}$ GPIO Over Voltage flag |
| $G_NUV$ | $N^{th}$ GPIO Under Voltage flag |
| $V_A$ UV | Analog Power supply Under Voltage flag |
| $V_A$ OV | Analog Power supply Over Voltage flag |
| $V_D$ UV | Digital Power supply Under Voltage flag |
| $V_D$ OV | Digital Power supply Over Voltage flag |
| STK UV | Stack Under Voltage flag |

| | |
|---|---|
| STK OV | Stack Over Voltage flag |
| DIE UT | Die Under Temperature |
| DIE OT | Die Over Temperature |
| CxOW | X numbered cell Open Wire flag |
| AFE COM | AFE Communication Fault flag |
| SPI fail | SPI Communication Failure flag |
| X | Reserved bit |

## 3.7.5. GUI Start Message

### *3.7.5.1. General*

The GUI Start message shall be used by GUI to get the real time voltage reading(cell/gpio/stat) of any board that are connected to BMS.

### *3.7.5.2. Master (GUI) to Slave (BMS) Command packet*

| Byte | Field | Value |
|---|---|---|
| 0 - 1 | CL | 0x0014 |
| 2 | Opcode | 0x05 – Start Measurement |
| 3 | IC Count<br>*Total Number of ICs want to write* | 0x01 – 0x80 |
| 4 – 19 | IC number<br>*The size of this field is 16 bytes array, where each byte can hold up to 8 ICs request, where each bit represents requested operation need to perform or not on the respective IC in the accordance to IC Count* | 0x00000000000000000000000000000001 – 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF |
| 20 | Optypes | 0x01 – One Shot |
| 21 | DL | 0x00 |

### *3.7.5.3. Slave (BMS) to Master (GUI) Response packet*

The following Response (Voltage reading) will be sent to GUI at a frequency of the time as mentioned in configuration Command, along with this response, Fault Detection response is also sent to GUI w.r.t time interval mentioned in configuration Command.

| Byte | Field | Value |
|---|---|---|
| 0 - 1 | RL | 0x0079 |
| 2 | Opcode | 0x05 – Start Measurement |
| 3 - 18 | IC number<br>*Each bit represents the response of which ICs*<br>*Note: At a time only one IC`s response can be received* | 0x00000000000000000000000000000001 - 0x80000000000000000000000000000000 |
| 19 | Status | Response Code |
| 20 | DL | 0x66 |
| 21 - 122 | DATA:<br>Response data will consist of all the required voltages in the format<br>Byte-1: Data (indicating which voltage)<br>Byte-2: Length of the required voltage data<br>Byte-3-variable: Voltage data.<br>Repeats the same sequence in continuation if multiple types of voltages are requested.<br>Ex:<br>1. *If Data in Command is 0x01 Then response's Data would be 0130-------0 Here 01 indicates cell voltage, 30 indicates size, following --------0 assume it as cell voltage data.*<br>2. *If Data in Command is 0x03 Then response's Data would be 0130-------00224-------0 Here 01 indicates cell voltage, 30 indicates size, following --------0 assume it as cell voltage data, 02 indicates GPIO voltage, 24 indicates it's size, following --------* | 0x00-0xFF per byte |

| | | |
|---|---|---|
| | *0 assume it as gpio voltage data*.<br>*Note*:<br>1. For Cell Voltages, Cell voltage register group A – F responses from ADBMS are directly sent to GUI in Data field.<br>2. For GPIO Voltages, Auxiliary Register Group A– D responses from ADBMS are directly sent to GUI in Data field.<br>3. For Stat Voltages, Status Register Group A– B responses from ADBMS are directly sent to GUI in Data field. | |

### 3.7.6. Read Message

#### *3.7.6.1. General*

The Read message shall be used by GUI to read registers of any cell are connected to BMS.

#### *3.7.6.2. Master (GUI) to Slave (BMS) Command packet*

| Byte | Field | Value |
|---|---|---|
| 0 - 1 | CL (Excluding this field) | Variable |
| 2 | Opcode | 0x0B – Read |
| 3 | IC Count<br>*Total Number of ICs want to read* | 0x01 – 0x80 |
| 4 – 19 | IC number<br>*The size of this field is 16 bytes array, where each byte can hold up to 8 ICs request, where each bit represents requested operation need to perform or not on the respective IC in the accordance to IC Count* | 0x00000000000000000000000000000001 – 0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF |
| 20 | Optypes | 0x01 – One-Shot<br>0x02 – Continuous<br>0x03 – Stop |
| 21 | DL | Variable |

| | | |
|---|---|---|
| 22 - Variable | DATA | Variable depends on DL |

#### *3.7.6.3. Slave (BMS) to Master (GUI) Response packet*

| Byte | Field | Value |
|---|---|---|
| 0 - 1 | RL | Variable |
| 2 | Opcode | 0x0B – Read |
| 3 – 18 | IC number<br>*Each bit represents the response of which ICs*<br>*Note: At a time only one IC`s response can be received* | 0x00000000000000000000000000000001 –<br>0x80000000000000000000000000000000 |
| 19 | Status | Response Code |
| 20 | DL | Variable |
| 21 - Variable | DATA | Variable depends on DL |

### 3.7.7. Write Message

#### *3.7.7.1. General*

The Write message shall be used by GUI to write registers of any cell are connected to BMS.

#### *3.7.7.2. Master (GUI) to Slave (BMS) Command packet*

| Byte | Field | Value |
|---|---|---|
| 0 - 1 | CL | Variable |
| 2 | Opcode | 0x0C - Write |
| 3 | IC Count<br>*Total Number of ICs want to write* | 0x01 – 0x80 |
| 4 – 19 | IC number<br>*The size of this field is 16 bytes array, where each byte can hold up to 8 ICs request, where each bit represents requested operation need to perform or not on the respective IC in the accordance to IC Count* | 0x00000000000000000000000000000001 –<br>0xFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF |
| 20 | Optypes | 0x01 – One-Shot |

| | | |
|---|---|---|
| | | 0x02 – Continuous<br>0x03 – Stop |
| 21 | DL | Variable |
| 22 - Variable | DATA | Variable depends on DL |

#### *3.7.7.3. Slave (BMS) to Master (GUI) Response packet*

| Byte | Field | Value |
|---|---|---|
| 0 - 1 | RL | Variable |
| 2 | Opcode | 0x0C - Write |
| 3 – 18 | IC number<br>*Each bit represents the response of which ICs*<br>*Note: At a time only one IC`s response can be received* | 0x00000000000000000000000000000001 –<br>0x80000000000000000000000000000000 |
| 19 | Status | Response Code |
| 20 | DL | Variable |
| 21 - Variable | DATA | Variable depends on DL |

## 3.8. List of Slave (BMS) to Master (GUI) Response code

The BMS shall reply with one of the following response codes:

| Response Code | Meaning |
|---|---|
| 0x00 | Reserved |
| 0x01 | Command Accepted |
| 0x02 | Unrecognised SOF |
| 0x03 | Invalid message length |
| 0x04 | Invalid message type |
| 0x05 | Invalid command length |
| 0x06 | Unrecognised Opcode |
| 0x07 | Invalid command type |
| 0x08 | Invalid Number of ICs |

| | |
|---|---|
| 0x09 | Invalid or Mismatched IC number with IC Count |
| 0x0A | Invalid or Mismatched IC type with IC Count |
| 0x0B | Invalid operation type |
| 0x0C | Invalid command data length |
| 0x0D | Requested IC Count not detected |
| 0x0E | Configuration not yet done |
| 0x0F | Previous command in progress |
| 0x10 | Checksum mismatched |
| 0x11 | ADBMS1818 CRC mismatched |
| 0x12 | SPI/IsoSPI Communication error |
| 0x13 | GUI not yet connected |
| 0x14 - 0xFF | Reserved |

# 4. Appendix

## 4.1. Write Message Example

The following is the message to write CFGRA (Configuration register group A) on ADBMS:

Here, IC count is 0x01

Read Message:

424D5300250100200C0100000000000000000000000000000001010C00013D6EE05227A00050B628FAEA

Message Parsing:

SOF - 424D53
ML - 0025
MT - 01
CL - 0020
Opcode - 0C
ICC - 01
ICN - 00000000000000000000000000000001

Optypes - 01
DL - 0C
Data - 00013D6E E0 52 27 A0 00 50 B628
Checksum – FAEA

Response Message: 424D5300180200130C00000000000000000000000000000000010100FEE3
Message Parsing:

SOF - 424D53
ML - 0018
MT - 02
CL - 0013
Opcode - 0C
ICN - 00000000000000000000000000000001
Status - 01
DL - 00
Checksum - FEE3

## 4.2. Read Message Example

The following is the message to read CFGRA (Configuration register group A) from ADBMS:

Here, IC count is 0x01

Write Message: 424D53001D0100180B0100000000000000000000000000000001010400022B0AFE9F

Message Parsing:

SOF - 424D53
ML - 001D
MT - 01
CL - 0018
Opcode - 0B
ICC - 01
ICN - 00000000000000000000000000000001
Optypes - 01
DL - 04
Data - 00022B0A
Checksum – FE9F

Response Message:
424D53002002001B0B0000000000000000000000000000000010108DA5227A00040035AFC3C
Message Parsing:

SOF - 424D53
ML - 0020
MT - 02
CL - 001B
Opcode - 0B
ICN - 00000000000000000000000000000001
Status - 01
DL - 08
Data - DA 52 27 A0 00 40 035A
Checksum – FC3C

## 4.3. Checksum Calculation Method

The 16-bit 2's complement algorithm will be used to calculate the checksum on whole message.

**CheckSum16 2`s Complement:** 0x10000 - Sum of Bytes