



## **CAN Protocol Specs**



## Introduction

This document outlines the data communication and fault propagation protocol followed by the CAN of Vecmocon's battery management systems. The protocol is based on SAEJ1939 with custom command/data messages on vendor reserved PGNs.

## Physical Layer

The physical layer conforms to the J1939 requirements and uses a CAN2.0B (non-FD) controller peripheral with a galvanically isolated CAN transceiver on the battery management system for safe and reliable communication at all times.

The bit-timing should be calculated for a bitrate of 500 Kbps with the sample point at 75% and segment jump width (SJW) equal to 1.

## Data Link Layer

The data link layer of the CAN communication on the battery management system conforms to the SAEJ1939 specifications. However, host communications may be implemented without conforming to the complete specification if the required message parsing outlined in the document is met.

## Network Management Layer

The battery management system series does not perform any kind of network management or address claiming. All addresses are hardcoded into the firmware and follow the addressing scheme described below.

Address of Node  $n = (\text{OFFSET} + (n-1))$

**where the offset is set to 0x23 and n is the number of nodes.**

**Note: All messages sent on the bus must be of little-endian or Intel byte format.**

## Synchronization

Synchronization with a time-master is necessary for all ECU's sampling vehicle data. When any Sync request is sent to the BMS, it syncs its ticks with that of the companion device and then starts transmitting data and each data frame has ticks within its packet which denote the time when the data was being captured. As long as the BMS is synced the BMS keeps on sending data packets unless it's DeSynced.

### **Device Synchronization Adapted from AUTOSAR\_SWS\_TimeSyncOverCAN.pdf Section 7.3**

The Time Synchronization over CAN is responsible for realizing the CAN-specific Time Synchronization protocol.

SYNC and FUP messages are assigned to a dedicated message type "TimeSync".

SYNC and FUP messages of the same Time Domain share the same CAN ID by using a multiplexed signal group. For different Time Domains, the same CAN ID may be used if TimeSync messages are sent by the same Time Master or Time Gateway.

The multiplexer is located at Byte 0, named as "Type".

Deviation From Spec: The byte order for time value signals in Time Sync messages is "Little Endian"

## **OFS and OFNS messages are not supported**

The DLC of SYNC, FUP messages is 8 bytes

Data Page: 0

Extended Data Page: 0

PGN: 0x00EF00 (61184)

PF: 239

PS: DA

### **SYNC**

Byte 0: Type = 0x10

Byte 1: User Byte 1, default: 0

Byte 2: D = Time Domain 0 to 15 (Bit 7 to Bit 4)

SC = Sequence Counter (Bit 3 to Bit 0)

Byte 3: User Byte 0, default: 0

Byte 4-7: SyncTimeMS

### **FUP**

Byte 0: Type = 0x18

Byte 1: User Byte 2, default: 0

Byte 2: D = Time Domain 0 to 15 (Bit 7 to Bit 4)

SC = Sequence Counter (Bit 3 to Bit 0)

Byte 3: Reserved, default: 0

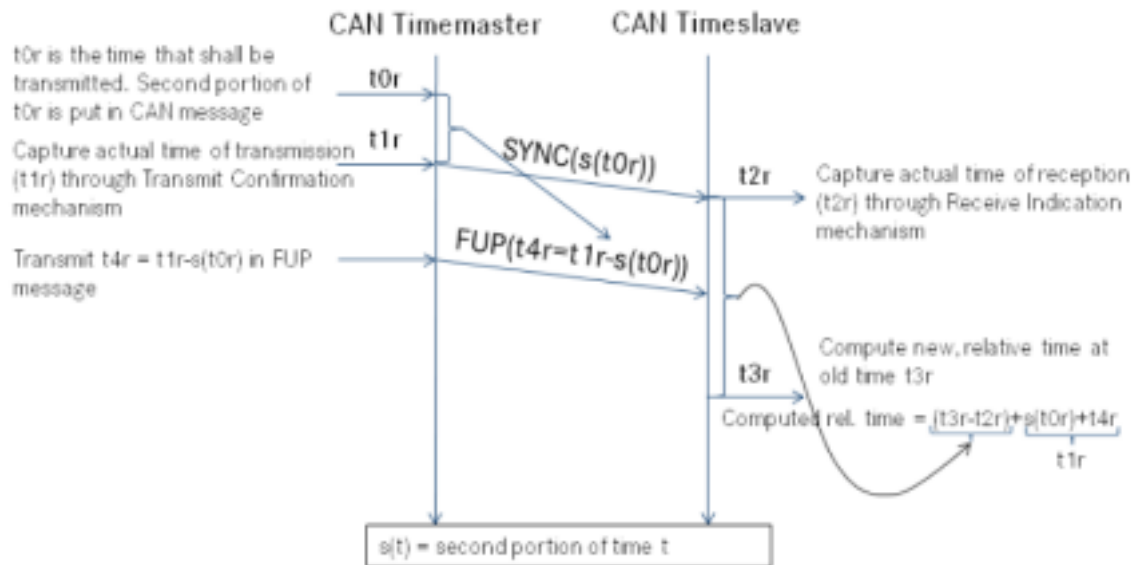
Byte 4-7: SyncTimeMS offset

A Time-Domain represents the Time-master and Time-slave groups in a CAN network. They exist so that every SYNC has a corresponding FUP.

A Sequence Counter of 4 bits is sent with every Time-Sync sequence. The value of SC must match in the SYNC and FUP messages

A Time Master shall start each Time Synchronization sequence for a Synchronized Time Base with a SYNC message

A Time Master shall finish each Time Synchronization sequence for a Synchronized Time Base with a FUP message.



Capture actual time of transmission via CAN message ACK

Capture actual time of reception by PG match

TimeSync timeout: 0.5 seconds, If no FUP is received within 500ms, time sync fails

### Example Packets

	ID	DLC	Data0	Data1	Data2	Data3	Data4	Data5	Data6	Data7
Sync Packet 1	0x14ef2330	8	10 0		11 0		0	0	0	0
Sync Packet 2	0x14ef2330	8	18 0		11 0		0	0	0	0

**Note: The Node ID is 0x30 in the above message.**

## Message Formats

### Requests

Requests are transmitted/received on the Request PGN ( $59904_{10}, 00EA00_{16}$ ). Every request expects either a response/acknowledgment as appropriate. This PGN provides the capability to request information globally or from a specific destination.

If the request is sent to a global address, then the response is sent to a global address.

A NACK is not permitted as a response to a global request

Data Length	3
Extended Data Page	0
Data Page	0
PDU Format	234
PDU Specific	Destination Address (0x23 Address of the BMS)
Default Priority	6

Parameter Group Number	$59904_{10}, 00EA00_{16}$
Data Byte 1 (PGN)	LSB of PGN (bit 8 first, bit 0 last)
Data Byte 2 (PGN)	2 <sup>nd</sup> byte of PGN (bit 8 first, bit 0 last)
Data Byte 3 (PGN)	MSB of PGN (bit 8 first, bit 0 last)
Data Byte 4-8 (Empty)	0xFF,0xFF,0xFF,0xFF,0xFF

**Table 1: Requests Format**

## Acknowledgement

The Acknowledgement PGN ( $59392_{10}, 00E800_{16}$ ) is used to provide a handshake mechanism between transmitting and receiving devices. There are 4 types of acknowledgments available based on the content of the control byte i.e. Positive Acknowledgement, Negative Acknowledgement, Cannot Respond, Authentication Denied.

Data Length	8
Extended Data Page	0
Data Page	0
PDU Format	232
PDU Specific	255
Default Priority	6
Parameter Group Number	$59392_{10}, 00E800_{16}$
Data Byte 1 (Control Byte)	0 = ACK, 1 = NACK, 2 = AD, 3 = CR
Data Byte 2 (Group Function Value)	First byte for proprietary PGN, otherwise 0xFF
Data Byte 3-4 (Reserved)	0xFF, 0xFF
Data Byte 5 (Address of Request)	Address
Data Byte 6-8 (PGN of request)	PGN of requested information (format in Table 1)

**Table 2: Acknowledgement Format**

## Broadcast Messages

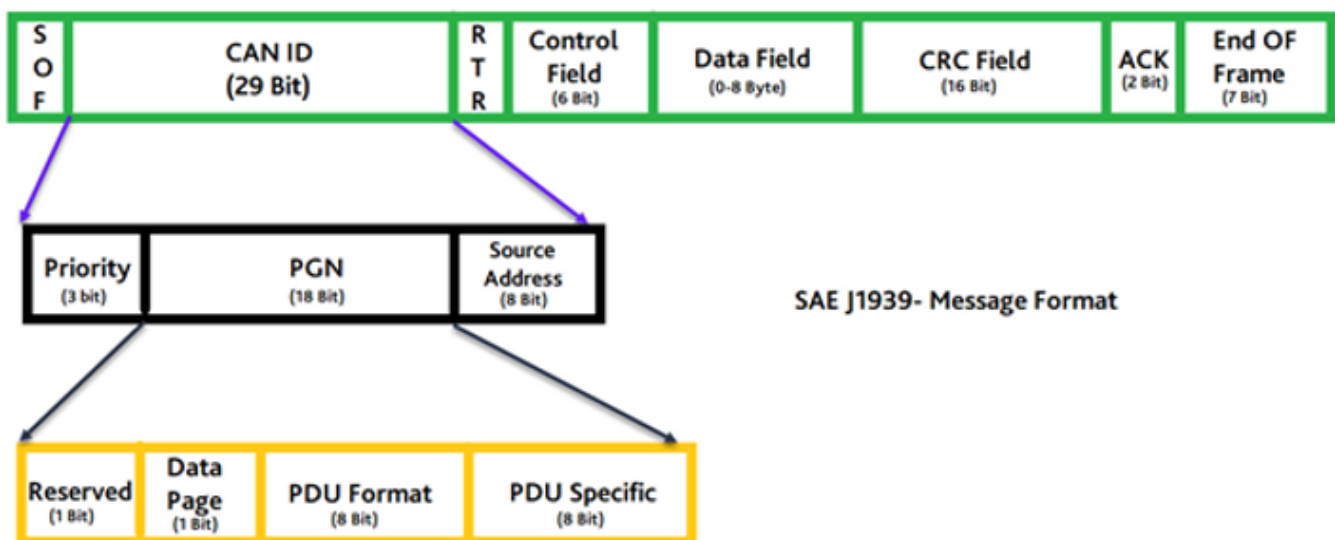
### 1. Single Packet Messages

SPMs are the simplest and most commonly used message format in J1939. They consist of a single data packet with a maximum length of 8 bytes, which can carry up to 64 bits of data.

SPMs are identified by their parameter group number (PGN), which is a 3-byte code that specifies the type of data being transmitted. The first byte of the PGN specifies the data priority, while the second and third bytes indicate the function or application of the data.

<b>Data Length</b>	8
<b>Extended Data Page</b>	0
<b>Data Page</b>	0
<b>PDU Format</b>	>240
<b>PDU Specific</b>	255
<b>Default Priority</b>	6

Following structure is used for identification of single packet messages.



- Priority:** Controls a message's priority during the arbitration process. A "0" value holds the highest priority, typically given to high-speed control messages. For example, the messages coming from brakes will always prioritize messages coming from the vehicle's ambient temperature. The Priority for broadcast messages is 6.



- **Extended Data Page and Data Page:** These bits are included in the PGN for counting as the most significant bit.
- **PDU format:** Determines whether the message is to be transmitted with a destination address or if always transmitted as a broadcast message.
- **PDU Specific:** The PDU specific field changes based on the PDU Format value:  
PDU format lies from 240-255: -
  - ❖ Then the message can only be broadcast (BAM).
  - ❖ PDU format and the Group Extension in the PDU specific field form the PGN of the transmitted parameter group.
  - ❖ The Group extension expands the number of possible broadcast Parameter Groups that the identifier can represent.
- **Source Address:** These contain the address of the device transmitting the message. The address is a specific label assigned to access a given device on the network uniquely. For any given network, every address must be unique. There are a total of 254 different addresses available. No two different devices (ECUs) can use the same address.

## 2. Multi-Packet Messages

MPMs are used when the data to be transmitted exceeds the maximum length of 8 bytes allowed by SPMs. MPMs can consist of multiple packets, each with a maximum length of 8 bytes, that are sent sequentially. The packets are numbered to indicate their order and to ensure that they are reassembled correctly by the receiving ECU.

MPMs use a special PGN format that includes a group extension (GE) byte in addition to the standard three-byte PGN. The GE byte identifies the specific segment of the MPM being transmitted and allows the receiving ECU to correctly assemble the complete message.

<b>Data Length</b>	<b>&gt;8</b>
<b>Extended Data Page</b>	<b>0</b>
<b>Data Page</b>	<b>0</b>
<b>PDU Format</b>	<b>&gt;240</b>
<b>PDU Specific</b>	<b>255</b>
<b>Default Priority</b>	<b>6</b>

MultiPacket Messages are sent using two CAN IDs -

**1cecff23** - Used to transmit Message information.

**1cebff23** - Used to transmit data packets.

## Commands

- **Sync Request command :**

- PGN : 0x00EF00UL
- CAN\_SYNC\_GF\_FUP 0x5

- **Desync Request command :**

- PGN : 0x00EF00UL (Note: For Desync PGN is sent in the request type message)

- **Data Messages :**

Note: Data Packets with DLC > 8 bytes are multi packet messages and need to be reconstructed at the receiving end

## BMS Payloads

### 1. Battery Cell Voltages

This Message contains battery cell voltages of each cell from the battery. Following is the format of the payload.

Message Type: MultiPacket Message

Transmission Type: Broadcast Announce Message

Transmission Period: 1s

#### Packet Structure:-

Group Function		Data Type	Length(Bit)	Offset(Bits)
ms time		Unsigned Integer	32	-
Stack voltage	max ~4 million Volts	Unsigned Integer	32	-
Stack number (n)	n > 0, max 4590 Series	Unsigned Integer	8	-
Cell (1 + (n-1)) Voltage	Stack n Cell Voltage m	Unsigned Integer	16	-
Cell (2 + (n-1)) Voltage	Stack n Cell Voltage m	Unsigned Integer	16	-
Cell (3 + (n-1)) Voltage	Stack n Cell Voltage m	Unsigned Integer	16	-
Cell (4 + (n-1)) Voltage	Stack n Cell Voltage m	Unsigned Integer	16	-
Cell (5 + (n-1)) Voltage	Stack n Cell Voltage m	Unsigned Integer	16	-
Cell (6 + (n-1)) Voltage	Stack n Cell Voltage m	Unsigned Integer	16	-
Cell (7 + (n-1)) Voltage	Stack n Cell Voltage m	Unsigned Integer	16	-
Cell (8 + (n-1)) Voltage	Stack n Cell Voltage m	Unsigned Integer	16	-
Cell (9 + (n-1)) Voltage	Stack n Cell Voltage m	Unsigned Integer	16	-
Cell (10 + (n-1)) Voltage	Stack n Cell Voltage m	Unsigned Integer	16	-
Cell (11 + (n-1)) Voltage	Stack n Cell Voltage m	Unsigned Integer	16	-
Cell (12 + (n-1)) Voltage	Stack n Cell Voltage m	Unsigned Integer	16	-
Cell (13 + (n-1)) Voltage	Stack n Cell Voltage m	Unsigned Integer	16	-
Cell (14 + (n-1)) Voltage	Stack n Cell Voltage m	Unsigned Integer	16	-
Cell (15 + (n-1)) Voltage	Stack n Cell Voltage m	Unsigned Integer	16	-
Cell (16 + (n-1)) Voltage	Stack n Cell Voltage m	Unsigned Integer	16	-
Cell (17 + (n-1)) Voltage	Stack n Cell Voltage m	Unsigned Integer	16	-
Cell (18 + (n-1)) Voltage	Stack n Cell Voltage m	Unsigned Integer	16	-

Example logs for the packet:-

ID: 1cecff23	X Rx	DL: 8	20 2d 00 07 ff c7 ff 00
ID: 1cebff23	X Rx	DL: 8	01 f4 53 01 00 8a c5 00
ID: 1cebff23	X Rx	DL: 8	02 00 01 1e 0b 74 0c ad
ID: 1cebff23	X Rx	DL: 8	03 0c 56 0c 80 0b 4b 0c
ID: 1cebff23	X Rx	DL: 8	04 9d 0c 01 0b c3 0c d0
ID: 1cebff23	X Rx	DL: 8	05 0c ff 0b fb 0c 68 0c
ID: 1cebff23	X Rx	DL: 8	06 e7 0c be 0d c9 0c ff
ID: 1cebff23	X Rx	DL: 8	07 00 ff 00 ff ff ff ff

If we number the packets, then we get,

```

Packet 0   -   DL:  8   20 2d 00 07 ff c7 ff 00
Packet 1   -   DL:  8   01 f4 53 01 00 8a c5 00
Packet 2   -   DL:  8   02 00 01 1e 0b 74 0c ad
Packet 3   -   DL:  8   03 0c 56 0c 80 0b 4b 0c
Packet 4   -   DL:  8   04 9d 0c 01 0b c3 0c d0
Packet 5   -   DL:  8   05 0c ff 0b fb 0c 68 0c
Packet 6   -   DL:  8   06 e7 0c be 0d c9 0c ff
Packet 7   -   DL:  8   07 00 ff 00 ff ff ff ff

```

Parsing these packets, We get,

```

Packet 0 Data:-
20      -   Indicating Start of Broadcast
2d 00    -   Length of the data(Least Significant Byte first)
07      -   Number of packets
ff      -   Reserved
c7 ff 00 -   PGN(Least Significant Byte first)

```

```

Packet 1-7 Data:-
First Byte -   Packet Number
Following Bytes -   Raw Data

```

```

After Rearranging:-
Timestamp_BattTemp -> f4 53 01 00 -> 87028
StackVoltage -> 8a c5 00 00 ->
StackIndex_CellVoltages -> 01 -> 1
StacknCellVoltage_1 -> 1e 0b -> 0ble -> 2846
StacknCellVoltage_2 -> 74 0c -> 0c74 -> 3188
StacknCellVoltage_3 -> ad 0c -> 0cad -> 3245
StacknCellVoltage_4 -> 56 0c -> 0c56 -> 3158
StacknCellVoltage_5 -> 80 0b -> 0b80 -> 2944
StacknCellVoltage_6 -> 4b 0c -> 0c4b -> 3147
StacknCellVoltage_7 -> 9d 0c -> 0c9d -> 3229
StacknCellVoltage_8 -> 01 0b -> 0blc -> 2844
StacknCellVoltage_9 -> c3 0c -> 0cc3 -> 3267
StacknCellVoltage_10 -> d0 0c -> 0cd0 -> 3280
StacknCellVoltage_11 -> ff 0b -> 0bff -> 3071
StacknCellVoltage_12 -> fb 0c -> 0cfb -> 3323
StacknCellVoltage_13 -> 68 0c -> 0c68 -> 3176
StacknCellVoltage_14 -> e7 0c -> 0ce7 -> 3303
StacknCellVoltage_15 -> be 0d -> 0dbe -> 3518
StacknCellVoltage_16 -> c9 0c -> 0cc9 -> 3273

```

## 2. Battery Temperatures

This message contains the Battery Temperatures from NTC on the battery as well as onboard temperatures.

Message Type: MultiPacket Message

Transmission Type: Broadcast Announce Message

Transmission Period: 5s

### Packet Structure:-

When Stack number = 0, BMS temperatures are sent in the ID.

Group Function	Data Type	Length(Bit)	Offset(Bits)
ms time	Unsigned Integer	32	-
BMS temperatures (n=0)	Unsigned Integer	8	-
Precharge temperature	Signed Integer	16	-
Mosfet temperature 1	Signed Integer	16	-
Mosfet temperature 2	Signed Integer	16	-
BMS IC Temperature	Signed Integer	16	-
Reserved 0xFFFF	Signed Integer	16	-
Reserved 0xFFFF	Signed Integer	16	-
Reserved 0xFFFF	Signed Integer	16	-
Reserved 0xFFFF	Signed Integer	16	-
Reserved 0xFFFF	Signed Integer	16	-
Reserved 0xFF	Unsigned Integer	8	-

Example logs for the packet:-

ID: lcecff23	X Rx	DL: 8	20 18 00 04 ff 7e ff 00
ID: lcebff23	X Rx	DL: 8	01 90 63 01 00 00 fa 00
ID: lcebff23	X Rx	DL: 8	02 fe 00 00 00 ff 00 ff
ID: lcebff23	X Rx	DL: 8	03 00 ff 00 ff 00 ff 00
ID: lcebff23	X Rx	DL: 8	04 ff 00 ff ff ff ff ff

If we number the packets, then we get,

Packet 0	-	ID: lcecff23	X Rx	DL: 8	20 18 00 04 ff 7e ff 00
Packet 1	-	ID: lcebff23	X Rx	DL: 8	01 90 63 01 00 00 fa 00
Packet 2	-	ID: lcebff23	X Rx	DL: 8	02 fe 00 00 00 ff 00 ff
Packet 3	-	ID: lcebff23	X Rx	DL: 8	03 00 ff 00 ff 00 ff 00
Packet 4	-	ID: lcebff23	X Rx	DL: 8	04 ff 00 ff ff ff ff ff

Parsing these packets, We get,

#### Packet 0 Data:-

```

20      -   Indicating Start of Broadcast
18 00    -   Length of the data(Least Significant Byte first)
04      -   Number of packets
ff      -   Reserved
7E FF 00    -   PGN(Least Significant Byte first)

```

#### Packet 1-4 Data:-

```

First Byte -   Packet Number
Following Bytes -   Raw Data

```

#### After Rearranging:-

```

Timestamp_BattTemp -> 00016390 -> 91024
StackIndex_BattTemp -> 0
PrechargeTemperature -> 00FA -> 250
MOSFETTemperature_1 -> 00FE -> 254
MOSFETTemperature_2 -> 0000 -> 0
BMS_Board_Temperature -> 00ff -> 255

```

When Stack number = 1, Battery temperatures are being transmitted.

Group Function		Data Type	Length(Bit)	Offset(Bits)
ms time		Unsigned Integer	32	-
Stack number (n > 0)	n > 0, max 2295 thermistors	Unsigned Integer	8	-
Stack (n-1) Temp 1		Signed Integer	16	-
Stack (n-1) Temp 2		Signed Integer	16	-
Stack (n-1) Temp 3		Signed Integer	16	-
Stack (n-1) Temp 4		Signed Integer	16	-
Stack (n-1) Temp 5		Signed Integer	16	-
Stack (n-1) Temp 6		Signed Integer	16	-
Stack (n-1) Temp 7		Signed Integer	16	-
Stack (n-1) Temp 8		Signed Integer	16	-
Stack (n-1) Temp 9		Signed Integer	16	-
0xFF		Unsigned Integer	8	-

#### Example logs for the packet:-

```

ID: 1cecff23    X Rx          DL: 8    20 18 00 04 ff 7e ff 00
ID: 1cebff23    X Rx          DL: 8    01 8c 63 01 00 01 ff 00
ID: 1cebff23    X Rx          DL: 8    02 fe 00 f6 00 f6 00 f6
ID: 1cebff23    X Rx          DL: 8    03 00 f6 00 fe 00 00 00
ID: 1cebff23    X Rx          DL: 8    04 00 00 ff ff ff ff ff

```

If we number the packets, then we get,

```

Packet 0    -   ID: 1cecff23    X Rx          DL: 8    20 18 00 04 ff 7e ff 00
Packet 1    -   ID: 1cebff23    X Rx          DL: 8    01 8c 63 01 00 01 ff 00
Packet 2    -   ID: 1cebff23    X Rx          DL: 8    02 fe 00 f6 00 f6 00 f6
Packet 3    -   ID: 1cebff23    X Rx          DL: 8    03 00 f6 00 fe 00 00 00
Packet 4    -   ID: 1cebff23    X Rx          DL: 8    04 00 00 ff ff ff ff ff

```

Parsing these packets, We get,

Packet 0 Data:-

```
20      -   Indicating Start of Broadcast
18 00    -   Length of the data(Least Significant Byte first)
04      -   Number of packets
ff      -   Reserved
7E FF 00 -   PGN(Least Significant Byte first)
```

Packet 1-4 Data:-

```
First Byte -   Packet Number
Following Bytes -   Raw Data
```

After Rearranging:-

```
Timestamp_BattTemp -> 00016390 -> 91024
StackIndex_BattTemp -> 0
StackTemperature_1 -> 00FF -> 255
StackTemperature_2 -> 00FE -> 254
StackTemperature_3 -> 00F6 -> 250
StackTemperature_4 -> 00F6 -> 246
StackTemperature_5 -> 00F6 -> 246
StackTemperature_6 -> 00F6 -> 246
StackTemperature_7 -> 00FE -> 254
```

### 3. BMS Faults Status and Balancing Status

This message contains fault statuses from the BMS and balancing status of each cell.

Message Type: Single Packet Message

Transmission Type: Broadcast Announce Message

Transmission Period: 1s

#### Packet Structure:-

When Multiplexor = 0, the message contains BMS faults.

Group Function	Data Type	Length(Bit)	Offset(Bits)
Multiplexor=0		8	0
Charging MOSFET Status	Default Range, Table 1	Measured	2
Discharging MOSFET Status	Default Range, Table 1	Measured	2
Precharge MOSFET Status	Default Range, Table 1	Measured	2
Over Voltage Fault	Boolean	1	14
Under Voltage Fault	Boolean	1	15
Over Temperature Fault	Boolean	1	16
Under Temperature Fault	Boolean	1	17
Over Current Discharge Fault	Boolean	1	18
Over Current Charge Fault	Boolean	1	19
Short Circuit Fault	Boolean	1	20
MOSFET Over Temperature	Boolean	1	21
MOSFET Under Temperature	Boolean	1	22
AFE Unresponsive	Boolean	1	23
BMS Over Temperature	Boolean	1	24
BMS Under Temperature	Boolean	1	25
0x3F	-	6	26

Example logs for the packet:-

ID: 18fbf523 X Rx DL: 4 00 05 00 fc

Parsing these packets, We get,

ID: 18fbf523 X Rx DL: 4 00 05 00 fc

```
StackInex_FaultStat -> 00 -> 0
MosfetStatusCharge -> (05>>0)&0x03 = 0x01 -> "Enabled( on-active )" (Check its descriptor of value from DBC)
MosfetStatusDischarge -> (05>>2)&0x03 = 0x01 -> "Enabled( on-active )" (Check its descriptor of value from DBC)
MosfetStatusPrecharge -> (05>>4)&0x03 = 0x00 -> "Disabled( on-passive )" (Check its descriptor of value from DBC)
OverVoltageFault -> (05>>6) = 0
UnderVoltageFault -> (05>>7) = 0

OverTemperatureFault -> (0>>0) = 0
UnderTemperatureFault -> (0>>1) = 0
OverCurrentDischarge -> (0>>2) = 0
OverCurrentCharge -> (0>>3) = 0
ShortCircuitFault -> (0>>4) = 0
MosfetOverTemperature -> (0>>5) = 0
MosfetUnderTemperature -> (0>>6) = 0
BCCUnresponsiveFault -> (0>>7) = 0

BoardOverTemperature -> (fc>>0) = 0
BoardUnderTemperature -> (fc>>1) = 0
```

When Multiplexor = 1, the message contains BMS balancing status.

Group Function	Data Type	Length(Bit)	Offset(Bits)
Multiplexor=n (n>0)		8	0
Balancing at stack n cell 1	Boolean	1	8
Balancing at stack n cell 2	Boolean	1	9
Balancing at stack n cell 3	Boolean	1	10
Balancing at stack n cell 4	Boolean	1	11
Balancing at stack n cell 5	Boolean	1	12
Balancing at stack n cell 6	Boolean	1	13
Balancing at stack n cell 7	Boolean	1	14
Balancing at stack n cell 8	Boolean	1	15
Balancing at stack n cell 9	Boolean	1	16
Balancing at stack n cell 10	Boolean	1	17
Balancing at stack n cell 11	Boolean	1	18
Balancing at stack n cell 12	Boolean	1	19
Balancing at stack n cell 13	Boolean	1	20
Balancing at stack n cell 14	Boolean	1	21
Balancing at stack n cell 15	Boolean	1	22
Balancing at stack n cell 16	Boolean	1	23
Balancing at stack n cell 17	Boolean	1	24
Balancing at stack n cell 18	Boolean	1	25
0x3F	-	6	26

Example logs for the packet:-

ID: 18fbf523 X Rx DL: 4 01 00 00 00



Parsing these packets, We get,

ID: 18fbf523 X Rx DL: 4 01 00 00 00

```
StackInex_FaultStat -> 01 -> 1
StacknBalancingStatus1 -> 0>>0 = 0
StacknBalancingStatus1 -> 0>>0 = 0
StacknBalancingStatus2 -> 0>>0 = 0
StacknBalancingStatus3 -> 0>>0 = 0
StacknBalancingStatus4 -> 0>>0 = 0
StacknBalancingStatus5 -> 0>>0 = 0
StacknBalancingStatus6 -> 0>>0 = 0
StacknBalancingStatus7 -> 0>>0 = 0
StacknBalancingStatus8 -> 0>>0 = 0
StacknBalancingStatus9 -> 0>>0 = 0
StacknBalancingStatus10 -> 0>>0 = 0
StacknBalancingStatus11 -> 0>>0 = 0
StacknBalancingStatus12 -> 0>>0 = 0
StacknBalancingStatus13 -> 0>>0 = 0
StacknBalancingStatus14 -> 0>>0 = 0
StacknBalancingStatus15 -> 0>>0 = 0
StacknBalancingStatus16 -> 0>>0 = 0
```

## 4. Battery Output

This message shows discharging/charging current and Remaining Capacity.

Message Type: MultiPacket Message

Transmission Type: Broadcast Announce Message

Transmission Period: 0.5s

### Packet Structure:-

Group Function	Data Type	Length(Bit)	Offset(Bits)
ms timestamp	Unsigned Integer	32	-
Current	Signed Integer	32	-
Remaining Capacity	Signed Integer	32	-

Example logs for the packet:-

```
ID: 1cecff23 X Rx DL: 8 20 0c 00 02 ff b0 ff 00
ID: 1cebff23 X Rx DL: 8 01 0b 52 01 00 0c 00 00
ID: 1cebff23 X Rx DL: 8 02 00 1c 4d 04 00 ff ff
```

If we number the packets, then we get,

```

Packet0 -> ID: 1cecff23    X Rx          DL:  8    20 0c 00 02 ff b0 ff 00
Packet1 -> ID: 1cebff23    X Rx          DL:  8    01 0b 52 01 00 0c 00 00
Packet2 -> ID: 1cebff23    X Rx          DL:  8    02 00 1c 4d 04 00 ff ff

```

Parsing these packets, We get,

```

Packet 0 Data:-
20      -   Indicating Start of Broadcast
0c 00    -   Length of the data(Least Significant Byte first)
02      -   Number of packets
ff      -   Reserved
b0 ff 00 -   PGN(Least Significant Byte first)

Packet 1-2 Data:-
First Byte -   Packet Number
Following Bytes -   Raw Data

Timestamp_BattOutput -> 0b 52 01 00 -> 0001520b -> 86539
PackCurrent -> 00 0c 00 00 -> 0c -> 12 -> 3072
RemainingCapacity -> 1c 4d 04 00 -> 44dlc -> 281884

```

## 5. Battery State

This Message shows SOC and SOH of the battery.

Message Type: MultiPacket Message

Transmission Type: Broadcast Announce Message

Transmission Period: 5s

### Packet Structure:-

Group Function	Data Type	Length(Bit)	Offset(Bits)
ms timestamp	Unsigned Integer	32	0
SOC	Float	32	32
SOH	Float	32	48

Example logs for the packet:-

```

ID: 1cecff23    X Rx          DL:  8    20 0C 00 02 FF B5 FF 00
ID: 1cebff23    X Rx          DL:  8    01 51 0D 23 00 07 60 43
ID: 1cebff23    X Rx          DL:  8    02 41 00 00 C8 42 FF FF

```

If we number the packets, then we get,

Packet 0	-	ID: 1cecfff23	X Rx	DL: 8	20 0C 00 02 FF B5 FF 00
Packet 1	-	ID: 1cebfff23	X Rx	DL: 8	01 51 0D 23 00 07 60 43
Packet 2	-	ID: 1cebfff23	X Rx	DL: 8	02 41 00 00 C8 42 FF FF

Parsing these packets, We get,

```

Packet 0 Data:-
20      -   Indicating Start of Broadcast
0c 00   -   Length of the data(Least Significant Byte first)
02      -   Number of packets
ff      -   Reserved
b5 ff 00 -   PGN(Least Significant Byte first)

Packet 1-2 Data:-
First Byte -   Packet Number
Following Bytes -   Raw Data
Unused Data bytes is set to 0xFF

Timestamp_BattOutput -> 93 63 01 00 -> 00016393 -> 91027
SOC -> 07 60 43 41 -> 41436007 -> 12.2109
SOH -> 00 00 C8 42 -> 42c80000 -> 100

```

## 6. Internal Data

This Message shows the threshold of the battery. These parameters are required by the charger to set the current and voltage on the basis of the message.

**Packet Structure:-**

Group Function	Data Type	Length(Bit)	Offset(Bits)
Over Voltage Threshold	Unsigned Scaled	16	-
Under Voltage Threshold	Unsigned Scaled	16	-
Over Current Charge Threshold	Signed Scaled	32	-
Nominal Voltage (chemistry)	Unsigned Scaled	16	-
Over Temperature Threshold	Signed Scaled	16	-
Under Temperature Threshold	Signed Scaled	16	-
Full Charge Capacity	Unsigned Scaled	32	-
Design Capacity	Unsigned Scaled	16	-
Series Configuration	Unsigned Unscaled	8	-
Parallel Configuration	Unsigned Unscaled	8	-
ThermCount	Unsigned Unscaled	8	-



## **Copyright**

**Copyright © 2020  
Vecmocon Technologies Pvt. Ltd.  
All rights reserved.**

## **Disclaimer**

**This data sheet is provided in association with and is subject to, the terms and conditions contained in the “Standard Terms and Conditions of Sale for Vecmocon Technologies Pvt. Ltd.” for the companies Products. This includes the “Limited Warranty and Remedy”, “Limitations of Liability”, and “Disclaimer and Release” sections of those Standard Terms and Conditions.**

**The information contained throughout this manual is thought to be up to date at the time of publication. Vecmocon will endeavor to maintain this manual to reflect the most current information about the purchased products but has no obligation to do so. Vecmocon reserves the right to change the contents of this manual at any time without notice and assumes no liability for its accuracy.**

**In the preparation of this datasheet, Vecmocon has incorporated and/or compiled service information and maintenance procedures sourced from manufacturers and vendors of parts and components used in the manufacturing of this product. Vecmocon has relied on those manufacturers and vendors for that information and has not verified it. The content of that third-party information is not within Vecmocon’s control, and Vecmocon cannot and will not take responsibility for that information.**

**Vecmocon shall not be liable for errors, omissions, missing data, or any other matter, in respect of that information. It is not the intention of this data sheet to instruct personnel in using common sense, basic skills, and rules of, installation, service, or repair. The customer is solely responsible for installing the products and for undertaking the installation in a safe environment, using appropriate equipment, by experienced personnel.**

## **Confidentiality**

**The information contained in this document is the intellectual property of Vecmocon Technologies Pvt. Ltd. and is Commercially Confidential, subject to the confidentiality provisions of the “Standard Terms and Conditions of Sale for Vecmocon Technologies Pvt. Ltd.” No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Vecmocon Technologies Pvt. Ltd.**