BMS Can Protocol

Contents page

1. Revision of history-------------------------------------------2
2. Preface-------------------------------------------------------2
3. Scope---------------------------------------------------------2
4. Term----------------------------------------------------------2
5. Reference documentation---------------------------------------2
6. Protocol definitions------ -----------------------------------2

6.1. E-UAVCAN Protocol------------------------------------------2

6.1.1. ID field------------------------------------------------3

6.1.2. Can Payload---------------------------------------------3

6.2. BMS communication message----------------------------------5

6.2.1 Info (1092)6S----------------------------------------------6

6.3. BMS Default NodeID-----------------------------------------7

7. Version-------------------------------------------------------7

8. Appendix------------------------------------------------------7

1. Revision of history

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Comment | 作者 |
| V-A/0 | 2020-12-04 | First Publish |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

2. Preface

3. Scope

This paper mainly describes the protocol of intelligent battery (BMS) and external communication.

All applications require license from Greep.

4. Term

Table 3-1 Document Terms

|  |  |
| --- | --- |
| Term | Explanation |
| BMS | Battery Management System |
| CAN | Controller Area Network |

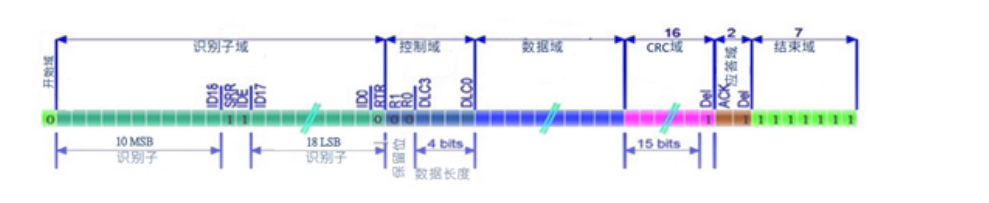
5. Reference documentation

《CANBUS规范v2.0+中文版.pdf》

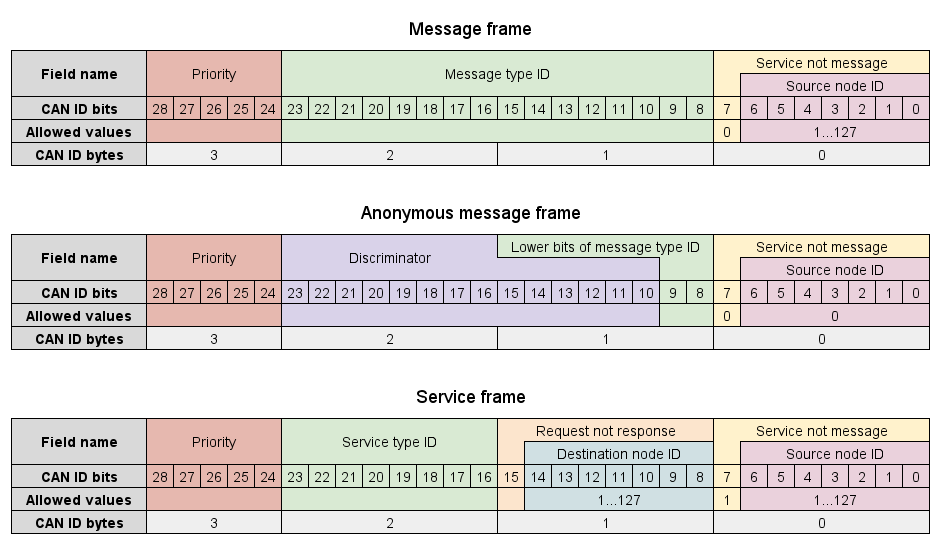
6. Protocol definitions

6.1. E-UAVCAN protocol

E -UAVCAN protocol is based on standard CANBus 2.0B protocol. extended frames based on 29 bit. Communication rate 1000 Kbps.



6.1.1. ID field

In the E -UAVCAN protocol, we only use the data frames defined in the CANBus, and all the data are transmitted through the data frames. We define data frames in the following format: 

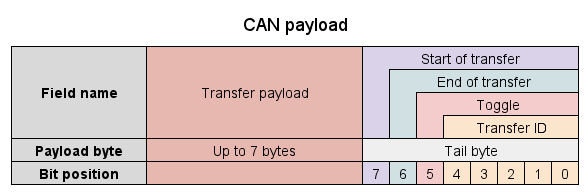
In the case of a message broadcast transfer, the CAN ID field of every frame of the transfer will contain the following fields:

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Bits | Allowed values | Description |
| Priority | 5 | 0-31 | Default highest priority :0 |
| Message type ID | 16 | 0x1092 | Query Mode |
| Service or message | 1 | 0 | 0x 1092----This bit is 0； |
| Source node ID | 7 | 1…127 | 0 is reserved and represents an unknown node;  Id of its own nodes |

※ BMS default Source node ID is 0 x16.

6.1.2.

CANBus2.0B CAN bus transmits 8 bits per frame As shown below:



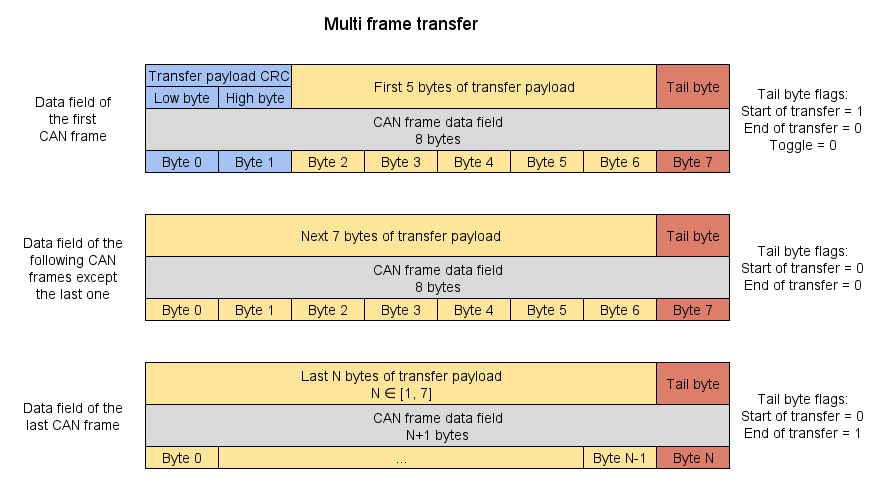
As E-UAVCAN protocol Provisions，Payload 8 Byte divided into two parts

|  |  |
| --- | --- |
| Field | Field Description |
| Transfer payload | Actual payload of the transfer |
| Tail byte | The last byte of the CAN frame data field that contains auxiliary fields of the transport layer |

Where Transfer payload is the payload data actually transmitted, the Tail byte contains the following information:

|  |  |  |
| --- | --- | --- |
| Field | Bits | Description |
| Start of transfer | 1 | See below |
| End of transfer | 1 | See below |
| Toggle bit | 1 | See below |
| Transfer ID | 5 | The transfer ID value |

Multi-frame transfer:

[](file:///F:\04--UAVCAN\4.%20CAN%20bus%20transport%20layer%20-%20UAVCAN_files\multi_frame_transfer.png)

6.1.2.1. Start of transfer

For multi-frame transfers, the value of this field is 1 if the current frame is the first frame of the transfer, and 0 otherwise.

6.1.2.2. End of transfer

For multi-frame transfers, the value of this field is 1 if the current frame is the last frame of the transfer, and 0 otherwise.

6.1.2.3. Toggle bit

For multi-frame transfers, this field contains the value of the toggle bit, which is specified above.

6.1.2.4. Transfer ID

For all kinds of transfers, this field contains the transfer ID value of the current transfer.

The value is 5 bits wide, therefore the allowed values range from 0 to 31, inclusively.

6.2. BMS communication message

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| name | Data Type ID | message type | description | note |
| info | 0x1092 | Message | Battery active reporting information, including: manufacturer number, battery model code, battery voltage, etc., as shown in Table 6.2.1 | 4HZ  Active reporting |

All messages or data are in the data field in frame format, transmitting 7 valid data at a time.

6.2.1 Info (1092)

Data field：

|  |  |  |  |
| --- | --- | --- | --- |
| Field | Instruction | Size(bytes) | Comment |
| Manufacturer's Number | Short type | 2 |  |
| Battery Model Code | Short type | 2 |  |
| Total battery voltage | Unsigned short type | 2 |  |
| Charge/discharge current | Short type，unit（10mA）  Note: Positive number indicates charge, negative number indicates discharge | 2 |  |
| Battery temperature 1 | Short type,unit（1℃） | 2 |  |
| Battery temperature 2 | Short type,unit（1℃） | 2 |  |
| Battery temperature … | Short type,unit（1℃） | 2 |  |
| Battery temperature n | Short type,unit（1℃） | 2 |  |
| State of charge | Unsigned short type, unit（1%） | 2 |  |
| cycle count | Unsigned short type unit（1） | 2 |  |
| State of health | Short type，unit（1%） | 2 | Based on the chemical characteristics of the battery |
| Cell 1 voltage | Unsigned short type ，unit（mv） | 2 |  |
| Cell 2 voltage | Unsigned short type，unit（mv） | 2 |  |
| Cell 3 voltage | Unsigned short type，unit（mv） | 2 |  |
| Cell … voltage | Unsigned short type，unit（mv） | 2 |  |
| Cell n-1 voltage | Unsigned short type，unit（mv） | 2 |  |
| Cell n voltage | Unsigned short type，unit（mv） | 2 |  |
| Design capacity | Unsigned short type，unit（mAh） | 2 | 0 |
| Remain capacity | Unsigned short type，unit（mAh） | 2 | 0 |
| Error message | Uint32 | 4 | Each bit represents the state of an error type |
| Serial number | Type: Char | 16 |  |

Note: Low byte is before high byte

6.3. Error List

|  |  |  |
| --- | --- | --- |
| Bit ID | description | Note |
| Bit0 | Battery temperature is too low | 1--- error occurred  0---No errors |
| Bit1 | Batterytemperature is too high |
| Bit2 | Charging overcurrent |
| Bit3 | Discharge Overcurrent |
| Bit4 | Total voltage undervoltage |
| Bit5 | Total voltage overvoltage |
| Bit6 | Too much voltage difference between single cell |
| Bit7 | Single cell overvoltage |
| Bit8 | Single cell undervoltage |
| Bit9 | Charging Short circuit |
| Bit10 | Discharge Short circuit |
| Bit11 | Low Capacity Alarm |
| Bit12 | Non-original charger |
| Bit13….Bit31 | reserved |

Note: Low byte is before high byte

7. Version

8.Appendix

CRC algorithm

#define CRC\_CCITT\_INIT 0xFFFF

#define CRC\_CCITT\_POLY 0x1021U

void CCITT\_CRC16Init(uint8\_t const \* bytes, uint16\_t len)

{

CCITT\_CRC16 =CRC\_CCITT\_INIT;

CCITT\_CRC\_ARRAY(bytes,len);

}

void CCITT\_CRCStep(uint8\_t byte)

{

uint32\_t j;

CCITT\_CRC16 ^= ((uint16\_t)byte << 8); for (j = 0; j < 8; j++)

{

CCITT\_CRC16=(CCITT\_CRC16 & 0x8000U)?((CCITT\_CRC16 << 1) ^ CRC\_CCITT\_POLY):(CCITT\_CRC16 << 1);

}

}

void CCITT\_CRC\_ARRAY(uint8\_t const \* bytes, uint16\_t len)

{

while (len--) CCITT\_CRCStep(\*bytes++);

}