MOTINOVA Mid drive system communication protocol of BMS

Drafted by: Dail.zhou

Checked by:

Approved by:

MOTINOVA TECHNOLOGY LIMITED

2020-3-12

Revision History

Date	Modifier	Content	Version
2020-3-12	Dail	First release.	V2.5.2

MOTINOVA Mid drive system communication protocol of BMS

1 System components

MC: Motor Controller

BMS: Battery Management System

PBU: Push Button Unit

HMI: Human Machine Interface

OBC: On Board Computer

ECU: Electronic Control Unit

CDL: CAN Dongle

APP: Application

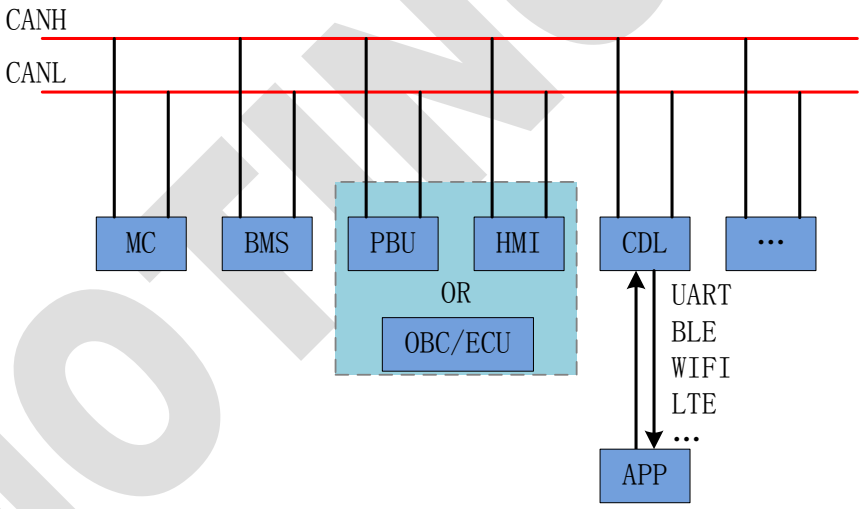

Figure1 Schematic diagram of system communication interface

2 Communication protocol rules

This protocol mainly describes the data communication format between the components of the MOTINOVA Mid drive system, which is only applicable to the communication between the internal components of the MOTINOVA Mid drive system.

2.1 Hardware interface

Type: CAN2.0A

Baud rate: 125kbps

2.2 Frame format

2.2.1 Frame format

The protocol describes the data content of each frame, including frame header, command segment length, command word, data segment, check bit and frame tail. The format of each frame is as follows:

Tabel1 Frame format

Header	Mode	Length	Command	Data	CRC	Tail
0x55 0xAA	Read/Write/Report	LENGTH	COMMAND	DATA	CRC	0xF0

Inside:

1) The fixed frame head is 0x55 0xAA and the fixed frame tail is 0xF0;
2) The frame mode includes reading 0x11, writing 0x16, and reporting 0x0C. When any device receives the write instruction, it needs to send general feedback instruction according to the data source;
3) The length is sum of command and data, one byte overhead for it;
4) Command indicates function of the frame, two bytes overhead for it;
5) Data includes the content of the frame, length is adjustable;
6) The CRC calculation result takes 4 bytes. When calculating, you need to insert CAN ID between frame header and frame mode. Such as: CAN ID is 0x712, and the frame content is 0x55 0xAA 0x11 0x03 0x22 0x01 0x00 CRC1 CRC2 CRC3 CRC4 0xF0, so you should input 0x55 0xAA 0x07 0x12 0x11 0x03 0x22 0x01 0x00 to the calculated function. The results are written into CRC1 CRC2 CRC3 and CRC4 from high to low;;
7) When fill the data segment, little-endian is adopt.

2.2.2 CAN ID allocation

Tabel2 CAN ID allocation

MC	Target	Broadcast	MC	BMS	PBU/OBC/ECU	HMI	CDL
	CAN ID	N/A		N/A	N/A	N/A	N/A
BMS	Target	Broadcast	MC	BMS	PBU	HMI	CDL
	CAN ID	0x720	N/A		N/A	N/A	N/A
PBU/OBC/ ECU	Target	Broadcast	MC	BMS	PBU	HMI	CDL
	CAN ID	N/A	N/A	0x732		N/A	N/A
HMI	Target	Broadcast	MC	BMS	PBU	HMI	CDL

	CAN ID	N/A	N/A	0x742	N/A		N/A
CDL	Target	Broadcast	MC	BMS	PBU	HMI	CDL
	CAN ID	N/A	N/A	0x752	N/A	N/A	

2.2.3 Packaging method

For data frames with a length of more than 8 bytes, subcontract in the way of 8 + N, and fill in the same ID number for each data packet, as shown in the following table:

Tabel3 Packaging method

Index	1		.....		N	
Content	ID	Byte1～Byte8	ID	Byte1～Byte8	ID	Byte1～ByteN

3 Communication content

3.1 Sent by BMS

Tabel4 Command list Sent by BMS

ID	Mode	Command	Function	Data
0x720	0x0C	0x1010	When BMS received the command 0x5000 from PBU/OBC/ECU, or the command 0x3400 form CDL, this command will be sent by BMS. This data package contains the running information of the BMS.	**Byte1-Byte2:** Bus voltage, unit is mV **Byte3-Byte4:** Bus current, unit is mA **Byte5-Byte6:** Remain capacity, unit is mAh **Byte7-Byte8:** Full charged capacity, unit is mAh **Byte9:** Cell temperature , need to cut 40, unit is Centigrade **Byte10:** SOC, from 0% to 100% **Byte11-Byte16:** 0x00
0x720	0x0C	0x1204	When the BMS goes into fault, the BMS sends fault code in 500ms period. After the fault disappears, it stop send fault code.	By bit or logical calculation , 0-normal, 1-abnormal. Four bytes can represent 32 types of faults at most. High 16bit: 0x00000000 Low 16bit: 0x0001: Discharge over current protection 2st 0x0002: Charge over current

				protection 0x0004:Short circuit protection 0x0008: Over discharge protection 0x0010: Over charge protection 0x0020: Discharge low temperature protection 0x0040: Discharge over temperature protection 0x0080: Charge low temperature protection 0x0100: Charge Over temperature protection 0x0200:D-MOS fault 0x0400: C-MOS fault 0x0800:NTC fault 0x1000: Discharge over current alarm 1st 0x2000: Discharge over current protection 1st 0x4000:AFE fault 0x8000:N/A
0x720	0x0C	0x1308	When the BMS goes to sleep, or turns off the D-MOS, and send it 1s in advance. According to the application requirements, BMS can be designed into three types: manual sleep(Press and hold the key for 3s), automatic sleep(Bus current is less than 500mA and CAN-Bus is idle) and no sleep(Keep awake unless there is a failure).	Fixed as “SHUTDOWN”, encoded in ASCII.
0x720	0x0C	0x1410	When BMS received the command 0x5200 from	**Byte1-Byte2:** Design capacity, unit is mAh

			PBU/OBC/ECU, or the command 0x5100 from HMI, or command 0x3600 from CDL, this command will be sent by BMS. This data package contains the design information of the BMS.	**Byte3:** Design voltage, unit is V **Byte4-Byte11:** Cell Model, encoded in ASCII, Extra bytes are filled as 0x2E. **Byte12-Byte16:** 0x00
0x720	0x0C	0x1540	When BMS received the command 0x5100 from PBU/OBC/ECU, or the command 0x5000 from HMI, or command 0x3300 from CDL, this command will be sent by BMS. This data package contains the version information of BMS.	The following information is encoded in ASCII, Extra bytes are filled as 0x2E. **Byte1-Byte16:** The model of BMS. **Byte17-Byte32:** The serial number of BMS. **Byte33-Byte48:** The hardware version of BMS. **Byte49-Byte64:** The software version of BMS.

3.2 Sent by PBU/OBC/ECU

Tabel5 Command list of sent by PBU/OBC/ECU

ID	Mode	Command	Function	Data
0x732	0x11	0x5000	Read BMS running information	No content
0x732	0x11	0x5100	Read BMS version information	No content
0x732	0x11	0x5200	Read BMS design information	No content

3.3 Sent by HMI

Tabel6 Command list of sent by HMI

ID	Mode	Command	Function	Data
0x742	0x11	0x5000	Read BMS version information	No content
0x742	0x11	0x5100	Read BMS design information	No content

3.4 Sent by CDL

Tabel7 Command list of sent by CDL

ID	Mode	Command	Function	Data
0x752	0x11	0x3300	Read BMS version	No content

			information	
0x752	0x11	0x3400	Read BMS running information	No content
0x752	0x11	0x3600	Read BMS design information	No content

4 Appendix 1: CRC32 calculation method

4.1 Calculation polynomial table

```
uint32_t Crc32Table[ 256 ] =
{
  0x00000000, 0x04C11DB7, 0x09823B6E, 0x0D4326D9, 0x130476DC, 0x17C56B6B,
  0x1A864DB2, 0x1E475005, 0x2608EDB8, 0x22C9F00F, 0x2F8AD6D6, 0x2B4BCB61,
  0x350C9B64, 0x31CD86D3, 0x3C8EA00A, 0x384FBDBD, 0x4C11DB70, 0x48D0C6C7,
  0x4593E01E, 0x4152FDA9, 0x5F15ADAC, 0x5BD4B01B, 0x569796C2, 0x52568B75,
  0x6A1936C8, 0x6ED82B7F, 0x639B0DA6, 0x675A1011, 0x791D4014, 0x7DDC5DA3,
  0x709F7B7A, 0x745E66CD, 0x9823B6E0, 0x9CE2AB57, 0x91A18D8E, 0x95609039,
  0x8B27C03C, 0x8FE6DD8B, 0x82A5FB52, 0x8664E6E5, 0xBE2B5B58, 0xBAEA46EF,
  0xB7A96036, 0xB3687D81, 0xAD2F2D84, 0xA9EE3033, 0xA4AD16EA, 0xA06C0B5D,
  0xD4326D90, 0xD0F37027, 0xDDB056FE, 0xD9714B49, 0xC7361B4C, 0xC3F706FB,
  0xCEB42022, 0xCA753D95, 0xF23A8028, 0xF6FB9D9F, 0xFBB8BB46, 0xFF79A6F1,
  0xE13EF6F4, 0xE5FFEB43, 0xE8BCCD9A, 0xEC7DD02D, 0x34867077, 0x30476DC0,
  0x3D044B19, 0x39C556AE, 0x278206AB, 0x23431B1C, 0x2E003DC5, 0x2AC12072,
  0x128E9DCF, 0x164F8078, 0x1B0CA6A1, 0x1FCDBB16, 0x018AEB13, 0x054BF6A4,
  0x0808D07D, 0x0CC9CDCA, 0x7897AB07, 0x7C56B6B0, 0x71159069, 0x75D48DDE,
  0x6B93DDDB, 0x6F52C06C, 0x6211E6B5, 0x66D0FB02, 0x5E9F46BF, 0x5A5E5B08,
  0x571D7DD1, 0x53DC6066, 0x4D9B3063, 0x495A2DD4, 0x44190B0D, 0x40D816BA,
  0xACA5C697, 0xA864DB20, 0xA527FDF9, 0xA1E6E04E, 0xBFA1B04B, 0xBB60ADFC,
  0xB6238B25, 0xB2E29692, 0x8AAD2B2F, 0x8E6C3698, 0x832F1041, 0x87EE0DF6,
  0x99A95DF3, 0x9D684044, 0x902B669D, 0x94EA7B2A, 0xE0B41DE7, 0xE4750050,
  0xE9362689, 0xEDF73B3E, 0xF3B06B3B, 0xF771768C, 0xFA325055, 0xFEF34DE2,
  0xC6BCF05F, 0xC27DEDE8, 0xCF3ECB31, 0xCBFFD686, 0xD5B88683, 0xD1799B34,
  0xDC3ABDED, 0xD8FBA05A, 0x690CE0EE, 0x6DCDFD59, 0x608EDB80, 0x644FC637,
  0x7A089632, 0x7EC98B85, 0x738AAD5C, 0x774BB0EB, 0x4F040D56, 0x4BC510E1,
  0x46863638, 0x42472B8F, 0x5C007B8A, 0x58C1663D, 0x558240E4, 0x51435D53,
  0x251D3B9E, 0x21DC2629, 0x2C9F00F0, 0x285E1D47, 0x36194D42, 0x32D850F5,
  0x3F9B762C, 0x3B5A6B9B, 0x0315D626, 0x07D4CB91, 0x0A97ED48, 0x0E56F0FF,
  0x1011A0FA, 0x14D0BD4D, 0x19939B94, 0x1D528623, 0xF12F560E, 0xF5EE4BB9,
  0xF8AD6D60, 0xFC6C70D7, 0xE22B20D2, 0xE6EA3D65, 0xEBA91BBC, 0xEF68060B,
  0xD727BBB6, 0xD3E6A601, 0xDEA580D8, 0xDA649D6F, 0xC423CD6A, 0xC0E2D0DD,
  0xCDA1F604, 0xC960EBB3, 0xBD3E8D7E, 0xB9FF90C9, 0xB4BCB610, 0xB07DABA7,
  0xAE3AFBA2, 0xAAFBE615, 0xA7B8C0CC, 0xA379DD7B, 0x9B3660C6, 0x9FF77D71,
  0x92B45BA8, 0x9675461F, 0x8832161A, 0x8CF30BAD, 0x81B02D74, 0x857130C3,
  0x5D8A9099, 0x594B8D2E, 0x5408ABF7, 0x50C9B640, 0x4E8EE645, 0x4A4FFBF2,
  0x470CDD2B, 0x43CDC09C, 0x7B827D21, 0x7F436096, 0x7200464F, 0x76C15BF8,
```

```
0x68860BFD, 0x6C47164A, 0x61043093, 0x65C52D24, 0x119B4BE9, 0x155A565E,
0x18197087, 0x1CD86D30, 0x029F3D35, 0x065E2082, 0x0B1D065B, 0x0FDC1BEC,
0x3793A651, 0x3352BBE6, 0x3E119D3F, 0x3AD08088, 0x2497D08D, 0x2056CD3A,
0x2D15EBE3, 0x29D4F654, 0xC5A92679, 0xC1683BCE, 0xCC2B1D17, 0xC8EA00A0,
0xD6AD50A5, 0xD26C4D12, 0xDF2F6BCB, 0xDBEE767C, 0xE3A1CBC1, 0xE760D676,
0xEA23F0AF, 0xEEE2ED18, 0xF0A5BD1D, 0xF464A0AA, 0xF9278673, 0xFDE69BC4,
0x89B8FD09, 0x8D79E0BE, 0x803AC667, 0x84FBDBD0, 0x9ABC8BD5, 0x9E7D9662,
0x933EB0BB, 0x97FFAD0C, 0xAFB010B1, 0xAB710D06, 0xA6322BDF, 0xA2F33668,
0xBCB4666D, 0xB8757BDA, 0xB5365D03, 0xB1F740B4 };
```

4.2 Calculation method

```
uint32_t CRC32_Calculate( uint8_t *pData, uint16_t Length )
{
  uint32_t nReg;
  uint32_t nTemp = 0;
  uint16_t i, n;

  nReg = 0xFFFFFFFF;
  for ( n = 0; n < Length; n++ )
  {
    nReg ^= (uint32_t) pData[ n ];
    for ( i = 0; i < 4; i++ )
    {
      nTemp = Crc32Table[ ( uint8_t )( ( nReg >> 24 ) & 0xFF ) ];
      nReg <<= 8;
      nReg ^= nTemp;
    }
  }
  return nReg;
}
```