MOTINOVA Mid-Motor Communication Protocol (ECU)

[填入文件编号]

Compiled By: Dail

Audited By: _____

Approved By: _____

MOTINOVA Technology Co., LTD.

2020-01-10

Modification Record

Modification Date	Modified By	Content	Version
20200110	Dail	First published	V1.0

MOTINOVA Mid-Motor Communication Protocol (ECU)

1 System Composition

MC: Motor Controller

BMS: Battery Management System

CDL: CAN Dongle

ECU: Central Control Unit

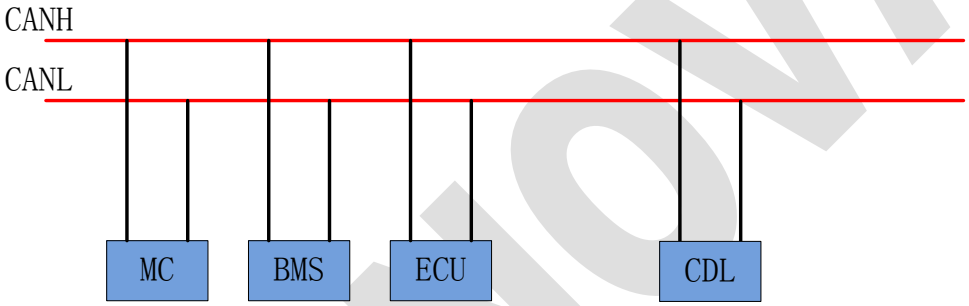

Figure 1 Communication Connector Graph

2 Communication Protocol Regulations

This protocol is mainly describing the format of data communication between the components of MOTINOVA mid-motor driving system which is only applicable to MOTINOVA.

2.1 Hardware Interface

Interface Type: CAN2.0A

Baud Rate: 125Kbps / 250Kbps

2.2 Data Frame Packing

2.2.1 Data frame format

This protocol is describing data content of each frame, including frame head, command segment length, command word, data segment, check bit and frame end.

Frame Format As Below:

Table 1 Data Frame Format

Head	Mode	Length	Command	Data Segment	CRC	End
55 AA	Read/Write/Report	LENGTH	COMMAND	DATA	CRC	F0

Including,

1) Frame head fixed to 0x55 0xAA, frame end fixed to 0xF0;
2) Frame modes include reading 0x11, writing 0x16, and reporting 0x0C. Any device needs to send generic feedback instructions based on data resources when receiving writing instruction;
3) The total length of command segment takes 1 byte and the virtual value is 0x02～0xFF;
4) The command word takes 2 bytes, first of which is the order number and the second one is the length of the data segment;
5) The length of the data segment is LENGTH - 2;
6) CRC, the check bit takes 4 bytes. CAN_ID inserted between frame head and frame mode from the beginning is calculating to the last byte of the data segment. The calculation method please see appendix. The result is high byte first, for example:

   When CAN_ID is 0x0712 ,the data frame is 55 AA 11 03 22 01 00 CRC1 CRC2 CRC3 CRC4 F0. CRC, the input data of calculation function is 55 AA 07 12 11 03 22 01 00, and the calculation result is written to CRC1、CRC2、CRC3、CRC4 in order from high to low;
7) Little-endian is adopted when the data segment is sent.

2.2.2 ID Distribution

Table 2 ID Distribution

MC	Target	Broadcast	MC	BMS	ECU	CDL
	CAN ID	0x710		0x712	0x713	0x715
BMS	Target	Broadcast	MC	BMS	ECU	CDL
	CAN ID	0x720	0x721		0x723	0x725
ECU	Target	Broadcast	MC	BMS	ECU	CDL
	CAN ID	0x730	0x731	0x732		0x735
CDL	Target	Broadcast	MC	BMS	ECU	CDL
	CAN ID	0x750	0x751	0x752	0x753	

2.2.3 Encapsulation Type

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

table:

Table 3 Encapsulation Way

Packet Order No.	1		……		N	
Content	ID	Byte1～Byte8	ID	Byte1～Byte8	ID	Byte1～ByteN

3 Communication Content

3.1 MC Command Words Definition

Table 4 MC Command Words Definition

ID	Mode	Command	Function	Data Segment	Remark
Broadcast Command					
0x710	0x0C	0x1020	MC Running information (Returned when receive control instruction from ECU)	**Bike Speed:** 2bytes	0.1km/h
				Motor Speed: 2bytes	1rpm
				Power Out: 2bytes	1W
				Bus Voltage: 2bytes	1mV
				Bus Current: 2bytes	1mA
				Cadence: 1byte	1rpm
				Padel Torque: 1byte	1N.m
				Padel Direction: 1byte	0-Forward, 1-Backward, 2-Stop
				Assist Level: 1byte	0x00:OFF 0x01:ECO 0x02:NORM 0x03:SPORT 0x04:TURBO 0x33:SMART
				Light Switch: 1byte	0xF0:OFF 0xF1:ON
				SOC: 1byte	1% Invalid 0xFF
				Range Distance: 2bytes	1km Invalid 0xFFFF
				ODO Distance: 2bytes	1km
				Power Consume: 1byte	0.01Ah/km Invalid 0xFF

				PCB Temperature: 1bytes **Coil Temperature:** 1bytes **MCU Temperature:** 1bytes **Ride Distance:** 2bytes **Ride Time:** 2byte **Reserved:** 4bytes	+40℃ +40℃ +40℃ 0.1km 1s Fill 0x00
0x710	0x0C	0x1104	MC Fault Code (Auto sent at 200ms if there is a fault, and stop sending after the faul t disappears)	**double byte 32bit** **low 16 bit:** **0x0001:** OCP **0x0002:** UVP **0x0004:** OVP **0x0008:** Rotor Lock **0x0010:** OTP **0x0020:** SPS Fault **0x0040:** TQS Fault **0x0080:** HALL Fault **0x0100:** Open Phase **0x0200:** NTC Fault **0x2000:** MCU Fault **0x4000:** Cadence Fault **High 16 bit:** **0x0001:** MOS SC **0x0002:** Voltage Abnormal **0x0004:**	By bit or output, 0- normal, 1- fault

				Circuit Failure	
0x710	0x0C	0x1240	Motor version information (Return instruction)	ASCII	Sequence: MODE、SN、HW、FW; Each message is 16 bytes in length, ends with '.', and is invalid filled with 0x20. The FW are named in the format of "Vxrxrx_YYYYMMDD"
Send To ECU					
0x713	0x0C	0x5303	Generic feed back instruction (Return instruction)	ASCII	ACK
0x713	0x0C	0x5410	User Params (Return instruction)	**Wheel Size:** 1byte **Speed Limit:** 1byte **UVP:** 2bytes **Reserved:** 12bytes	1cm 1km/h 1mV Fill 0x00

3.2 BMS Command Word Definition

Table 5 BMS Command Word Definition

ID	Mode	Command	Function	Data Segment	Remark
Broadcast Command					
0x720	0x0C	0x1010	Battery Operating information(Return instruction)	**Voltage:** 2bytes **Current:** 2bytes **RC:** 2bytes **FCC:** 2bytes **Cell Temperature:** 1byte **SOC:** 1byte **Status:**	1mV 1mA, int16, Negative for discharge, positive for charge 1mAh 1mAh +40℃ 0～100% 0x00:Sleep

				1byte (By bit) **Reserved:** 5bytes	0x01:In Charging 0x02:RS 0x04: RS 0x08: RS 0x10: RS 0x20: RS 0x40: RS 0x80: RS Fill 0x00
0x720	0x0C	0x1204	BMS Fault Code (Auto sent at 200ms if there is a fault, and stop sending after the fault disappears)	**Double byte 32bit** **low 16 bit:** **High 16 Bits:** **0x0001:** Over voltage warning while in charge **0x0002:** Under voltage warning while in discharge **0x0004:** Over current warning while in charge **0x0008:** Over current warning while in discharge **0x0010:** High temperature warning while in charge **0x0020:** Low temperature warning while in charge **0x0040:** High temperature warning while in discharge **0x0080:** Low temperature warning while in	By bit or output, 0- normal, 1- fault

				discharge **0x0100:** MOS high temperature warning	
0x720	0x0C	0x1308	Shutdown instructions (Actively send, after 1s stop)	ASCII	Send before go into sleep mode, delay for 1s, then shutdown.
0x720	0x0C	0x1410	Battery design information (Return instruction)	Design Capacity: 2bytes Design Voltage: 1byte Cell Mode: 8bytes **Reserved:** 5bytes	1mAh 1V ASCII, ends with '.', and is invalid filled with 0x20. Fill 0x00
0x720	0x0C	0x1540	Battery version information (Return instruction)	ASCII	Sequence: MODE、SN、HW、FW; Each message is 16 bytes in length, ends with '.', and is invalid filled with 0x20. The FW are named in the format of "Vxrxrx_YYYYMMDD"

3.3 ECU Command Word Definition

Table 6 ECU Command Word Definition

ID	Mode	Command	Function	Data Segment	Remark
Broadcast Command					
0x730	0x0C	0x1008	Shutdown instructions (Actively send, after 1s stop)	ASCII	Send before turn off the system, delay for 1s, then turn off the system.
Send to MC					
0x731	0x0C	0x3708	Motor control instruction (Timed upload)	**Assist Level:** 1byte	0x00:OFF 0x01:ECO 0x02:NORM

				Light Switch: 1byte **Reserved:** 6bytes	0x03:SPORT 0x04:TURBO 0x33:SMART 0xF0:OFF 0xF1:ON Fill 0x00
0x731	0x11	0x3300	Read motor Parameters (Actively send, receive return or timeout stop)	/	/
0x731	0x16	0x3810	Write motor Parameters (Actively send, receive ack or timeout stop)	**Wheel Size:** 1byte **Speed Limit:** 1byte **UVP:** 2bytes **Reserved:** 12bytes	1cm 1km/h 1mV Fill 0x00
Send to BMS					
0x732	0x11	0x5000	Read BMS running information (Actively send, receive return or timeout stop)	/	/
0x732	0x11	0x5100	Read BMS version information (Actively send, receive return or timeout stop)	/	/
0x732	0x11	0x5200	Read BMS design information (Actively send, receive return or timeout stop)	/	/

4 Appendix 1: CRC32 Calculating Method

4.1 CRC32 Table of calculating polynomials

```
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 CRC32 Calculating 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;
}
```