1. Overview

This standard specifies the communication protocol between the Motor Control Unit (MCU) and Vehicle Control Unit (VCU)

2. Communication Protocol Specifications

I. Data Link Layer

Bus communication rate: 250Kbps

Please refer to CAN2.0B and SAE J1939 standards for data link layer specifications, please refer to the below allocation table for 29-bit identifier of the CAN extension frame：

			IDENTIFIER 11BITS											SRR	IDE
			PRIORITY			R	DP	PDU FORMAT(PF)						SRR	IDE
			3	2	1	1	1	8	7	6	5	4	3		
31	30	29	28	27	26	25	24	23	22	21	20	19	18		
IDENTIFIER EXTENSION 18BITS															
PF		PDU SPECIFIC(PS)								SOURCE ADDRESS(SA)					
2	1	8	7	6	5	4	3	2	1	8	7	6	5	4	3	2	1
17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

PRIORITY is 3-bit and has up to 8 configurations.

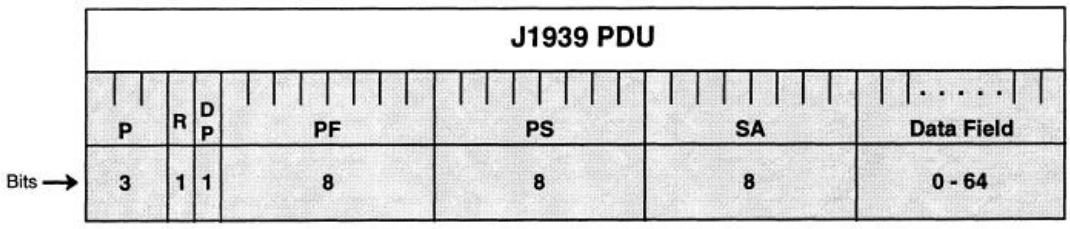
R is typically set at 0
DP current value set is 0
PF (8-bit) message
PS (8-bit) target address or group extension
SA (8-bit) source address from where the message was sent

›Each node that is connected to the network has a name and address, which is used to identify the function of the node and arbitrate the address, and the address is used for the data communication of the node

›Each node has at least one function. There may be more than one node with the same function, or one node may have more than one function

›For multibyte data, use a small-end approach, such as 4660=0x1234, sending 0x34 first, then 0x12

II. Protocol Data Unit (PDU)



PDU Specific (PS) protocol data unit details: PS is an 8-bit field description that is determined by the PDU FORMAT. It is up to the PF to decide whether the PS is a destination cell address or a group extension. If the value of the PF field is less than 240, the PS field represents the destination cell address. If the value of the PF field is within the range of 240 to 255, then the PS field represents the group expansion value.

	PDU Format (PF) Field	PDU Specified (PS) Field
PDU1 Format	0~239	Destination Address (DA)
PDU2 Format	240~255	Group Extension (GE)

III. CAN Network Address Assignment Table

If the CAN bus node address is already defined in J1939, try to use the address already defined in J1939. ECU with multiple functions can use multiple addresses or redefine new addresses; For newly defined addresses, 208~231, which are reserved addresses for road vehicles, should be used. The message number is the space allocated to each node for the purpose-addressing message number.

NODE	ADDRESS
Display Instrument (METER)	23(0x17)
Vehicle Control Unit (VCU)	208(0xD0)
Motor Control Unit (MCU)1	239(0xEF)
Motor Control Unit (MCU)2	240(0xF0)
Motor Control Unit (MCU)3	241(0xF1)
Motor Control Unit (MCU)4	242(0xF2)
Battery Management System (BMS)1	243(0xF3)
Battery Management System (BMS)2	244(0xF4)
Battery Management System (BMS)3	245(0xF5)
Battery Management System (BMS)4	246(0xF6)
GLOBAL (ANY NODE)	255(0xFF)

3. Messages

I. Vehicle Control Unit Sends

OUT	IN	ID(0x0C01EFD0)						Latency(ms)
VCU	MCU	P	R	DP	PF	PS	SA	50
		3	0	0	1(0x01)	239(0xEF) (N.B.1)	208(0xD0)	
DATA								
BYTE	BIT	DESCRIPTION			RESOLUTION		OFFSET	RANGE
0		Target Phase Current (Target Torque)			0.1A/bit		-3200A	-3200～3200A
1								
2		Target Speed			1rpm/bit		-32000rpm	-32000～32000rpm
3								
4	0	Command Controls					0	0：HALTED 1：RUNNING
	1							0：Torque Control Mode 1：Speed Control Mode
	7-2							Reserved
5		Reserved						
6		Reserved						
7		Life signal					0	0～0xFF

II. Motor Control Unit Sends Part I

OUT	IN	ID (0x1801D0EF)						Polling period (ms)
MCU	VCU	P	R	DP	PF	PS	SA	50
		6	0	0	1(0x01)	208(0xD0)	239(0xEF) (N.B.1)	
DATA								
BYTE	BIT	ITEM			RESOLUTION		OFFSET	RANGE
0		BUS Voltage			0.1V/bit		0	0～300V
1								
2		BUS Current			0.1A/bit		-3200A	-3200～3200A
3								
4		Phase Current			0.1A/bit		-3200A	-3200～3200A
5								
6		Speed			1rpm/bit		-32000rpm	-32000～32000rpm
7								

III. Motor Control Unit Sends Part II

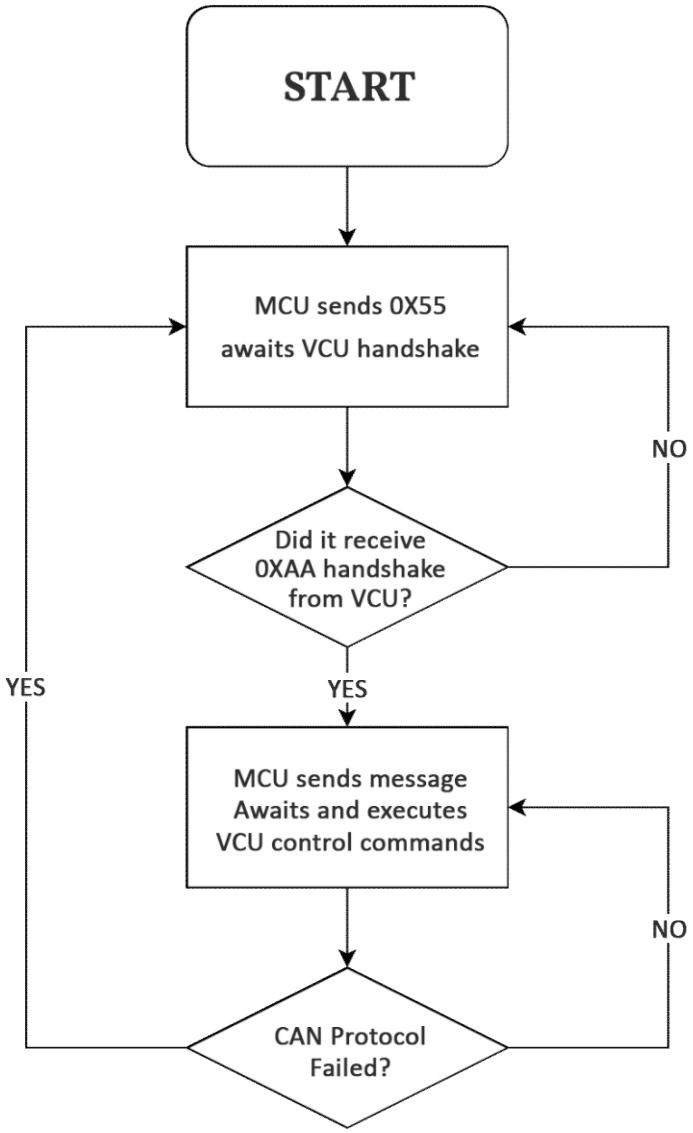
OUT	IN	ID (0x1802D0EF)						Polling period (ms)
MCU	VCU	P	R	DP	PF	PS	SA	50
		6	0	0	2(0x02)	208(0xD0)	239(0xEF) (N.B.1)	
DATA								
BYTE	BIT	ITEM			RESOLUTION		OFFSET	RANGE
0		Controller Temperature			1℃/bit		-40℃	-40～210℃
1		Motor Temperature			1℃/bit		-40℃	-40～210℃
2	0	STATUS					0	0：HALTED 1：RUNNING
	1							0：Torque Control Mode 1：Speed Control Mode
	7-2							Reserved
3	0	ERROR			Overcurrent		0	0：NORMAL 1：ERROR
	1				Overload			
	2				Overvoltage			
	3				Undervoltage			
	4				Controller Overheat			
	5				Motor Overheat			
	6				Motor Stalled			
	7				Motor Out of phase			
4	0				Motor Sensor			
	1				Motor AUX Sensor			
	2				Encoder Misaligned			
	3				Anti-Runaway Engaged			
	4				Main Accelerator			
	5				AUX Accelerator			
	6				Pre-charge			
	7				DC Contactor			
5	0				Power valve			
	1				Current Sensor			
	2				Auto-tune			
	3				RS485			
	4				CAN			
	5				Software			
	7-6				Reserved			
6					Reserved			
7		Life signal					0	0～0xFF

N.B. 1 SA values can be set by MCU host. Setting parameters: controller number, default SA = controller number = 239 (0xEF)

4. Handshake Protocol

After the MCU is powered on, it sends 8 bytes 0x55, ID=0x1801D0EF (N.B.1) in polling periods of 50ms (20Hz), prompting the VCU to comply. After receiving the awaiting handshake command sent by MCU, VCU returns 8 bytes 0xAA, ID=0x0C01EFD0 (N.B.1) to successfully shake hands with MCU. After MCU receives the handshake command sent by VCU, the handshake protocol is established. MCU starts to send messages (MCU sends message I and II). Awaiting and execute VCU control commands (VCU sends message I).

CAN communication protocol is determined to have failed if the MCU fails to receive VCU control command for 10 consecutive times (vehicle controller sends message I) or receives life signal failure 5 consecutive times. In such a case the MCU will shut down and attempt to restart handshake process.



5. Example script/code

This example is based on STM32F4 HAL library (VCU design, for your reference)

VCU sending Handshake Protocol

ID=0x0C01EFD0(N.B.1)

```
void VCU_SendHandshake(void)
{
    CAN_HandleTypeDef *CANxHandle = &hcan1;
    CAN_TxHeaderTypeDef CAN_TxHeader;
    uint32_t TxMailbox;
    uint8_t TxBuf[8];
    uint8_t Index;

    for (Index = 0; Index < 8; Index++)
    {
        TxBuf[Index] = 0xAA;
    }

    CAN_TxHeader.ExtId = 0x0C01EFD0;
    CAN_TxHeader.IDE = CAN_ID_EXT;
    CAN_TxHeader.RTR = CAN_RTR_DATA;
    CAN_TxHeader.DLC = 8;
    CAN_TxHeader.TransmitGlobalTime = DISABLE;
    HAL_CAN_AddTxMessage(CANxHandle, &CAN_TxHeader, TxBuf, &TxMailbox);
}
```

VCU sending control commands（VCU sends message I）

Send polling rate 50ms

ID=0x0C01EFD0(N.B.1)

```
uint16_t TargetPhaseCurrent01A;
uint16_t TargetSpeedRPM;
uint8_t ControlCmd;
uint8_t LiveCounter;
void VCU_SendCommand(void)
{
    CAN_HandleTypeDef *CANxHandle = &hcan1;
    CAN_TxHeaderTypeDef CAN_TxHeader;
    uint32_t TxMailbox;
    uint8_t TxBuf[8];
```

```
        TxBuf[0] = (uint8_t)(TargetPhaseCurrent01A);
        TxBuf[1] = (uint8_t)(TargetPhaseCurrent01A >> 8);
        TxBuf[2] = (uint8_t)(TargetSpeedRPM);
        TxBuf[3] = (uint8_t)(TargetSpeedRPM >> 8);
        TxBuf[4] = ControlCmd;
        TxBuf[5] = 0;//Reserved
        TxBuf[6] = 0;//Reserved
        TxBuf[7] = LiveCounter++;

        CAN_TxHeader.ExtId = 0x0C01EFD0;
        CAN_TxHeader.IDE = CAN_ID_EXT;
        CAN_TxHeader.RTR = CAN_RTR_DATA;
        CAN_TxHeader.DLC = 8;
        CAN_TxHeader.TransmitGlobalTime = DISABLE;
        HAL_CAN_AddTxMessage(CANxHandle, &CAN_TxHeader, TxBuf, &TxMailbox);
}
```