Shinwa BMS Communication Protocol

1Communication Electrical Standard

RS485(Half-Duplex Transmission) for Shinwa PC Software

RS485(Half-Duplex Transmission) for Terminal User's Monitor Module

2 Communication Parameters

BaudRate 9600bit/s，NoParity，8 Data Bits，1 Stop Bit

COM(9600 N 8 1)

3 Communication Process

PC/Monitor→Board(request)

Board→PC/Monitor(reply information of board)

Information include Cell Voltage、Current、Alarm Info、Protection Info。

4 Communication Command Format

Head	**Address**	**CID**	**Data length**	**Data**	**Check**	**Tail**
0x7E	0x00~0x0e	0x01	—	—	—	0x0D

Description：

1. PC Command and Board Command follow the same format
2. Head:Command Head(1byte）
3. Address:Board Address(1byte) (Dip Code Setting)
4. CID: Command ID(1byte)
5. Data length: (1byte)
6. Data: Data Content(not fixed length)
7. Check:Check Code(1byte)
8. Tail: Command Tail(1byte)

5 Board Command data description

- **Analog Value Content：**

Content	Child CMD	Description
Cell Vol	1	2Bytes per cell vol, Unit(mv)[Data0/Data1] Data0: Bit7:balance flag Bit6:Over Voltage flag Bit5:Under Voltage flag Data0[Bit4..Bit0]Data1[Bit7..Bit0]:value info
Current	2	Unit(0.01A)，Charging is Positive Value，Discharging is Negative Value. Offset Value:300A. `Current_value = (30000 - (Data0 * 256 + Data1)) / 100;` eg: Transfer:30101 Current_value=(30000-30101)/100=-1.01 Current is Discharge current, value is 1.01A
SOC	3	SOC（0-100）2Bytes
Capacity	4	Full Capacity, Unit(0.01AH) 2Bytes. (1AH~600AH)
Temperature	5	12Bytes Unit(℃) Offset -50 Eg: Act value：-50℃ After Offset：0℃ Value of CMD：0℃
Alarm/Protection Info	6	Read the state of the battery pack Return data 10 bytes (Data0..Data9)

		Data0: BIT0: (Reserved) BIT1: (Reserved) BIT2: (Reserved) BIT3: (Reserved) BIT4: (Reserved) BIT5: Charge_MOS_Error BIT6: Discharge_MOS_Error BIT7: Voltage_Module_Error Data1: BIT0: NTC_Line_Disconnected BIT1: Current_Module_Error BIT2: Charge_Source_Reversed BIT3: (Reserved) BIT4: (Reserved) BIT5: (Reserved) BIT6: (Reserved) BIT7: (Reserved) Data2: Bit0: Discharge_OT_Protect Bit1: Discharge_UT_Protect Bit2: (Reserved) Bit3: (Reserved) Bit4: (Reserved) Bit5: (Reserved) Bit6: (Reserved) Bit7: (Reserved)

		Data3: Bit0: Charging Bit1: Discharging Bit2: Short_Current_Protect Bit3: Over_Current_Protect Bit4: Over_Voltage_Protect Bit5: Under_Voltage_Protect Bit6: Charge_OT_Protect Bit7: Charge_UT_Protect Data4…Data9：(Reserved)
Cycle_count	7	Cycle_count 2Bytes
Pack_voltage	8	Unit:10mV, 2Bytes
SOH	9	SOH（0-100）2Bytes
Reserved	10	Reserved

6 Command Example(15S)

PC Command:

```
buf[0] = 0x7E;          //head
buf[1] = 0x00;          //addr
buf[2] = 0x01;          //CID
buf[3] = 0x00;          //data length
buf[4] = 0x00;          //Check Code
buf[5] = 0x0D;          //tail
```

BMS Return：

```
buf[0] = 0x7E;          //head
buf[1] = 0x00;          //addr
buf[2] = 0x01;          //CID
```

buf[3] = 0x3D; //data length

buf[4] = 0x01; // subcommand to read voltage

buf[5] = 0x0F; //number of cells

buf[6]….buf[35] //voltage ,15S, two bytes per cell, totally 30 bytes

buf[36] = 0x02; // subcommand to read current

buf[37] = 0x01; //number of current:1

buf[38]buf[39] //current value, 2bytes

buf[40] = 0x03; // subcommand to read SOC

buf[41] = 0x01; //number of SOC

buf[42]buf[43] //SOC，2bytes

buf[44] = 0x04; //subcommand to read full capacity

buf[45] = 0x01; //number of full capacity

buf[46]buf[47] //CAPACITY，2bytes

buf[48] = 0x05; //subcommand to read temperature

buf[49] = 0x06; //number of temperature

buf[50]…buf[61] //6 temperature, 12bytes

buf[62] = 0x06; //subcommand to read alarm

buf[63] = 0x05; //number of alarm

buf[64]…buf[73] //bytes

buf[74]=CHK; //check code

buf[75] = 0x0D;

```
TX:(2013-07-15 03:12:32)，发送6字节
7E 00 01 00 00 0D
RX:(2013-07-15 03:12:33)，接收76字节
7E 00 01 46 01 0F 0D 21 0D 22 0D 22 0D 25 0D 23 0D 1D 0D 23 0D 23 0D 26 0D 23 0D 25 0D 22 0D 22 0D 24 0D 24
02 01 73 E3 03 01 1D 63 04 01 07 D0 05 06 00 4C 00 4C 00 4D 00 4D 00 4D 00 4C 06 05 00 00 10 01 00 00 00 00
00 00 40 0D
```

Check_function:

```
//CHECK FUNCTION

byte check(byte[] buf, byte len)

{

    byte i, chk = 0;

    int sum = 0;

    for (i = 0; i < len; i++)

    {

        chk ^= buf[i];

        sum += buf[i];

    }

    return (byte)((chk ^ sum) & 0xFF);

}
```

7 Example code

Rxbuf array is receiver buffer:

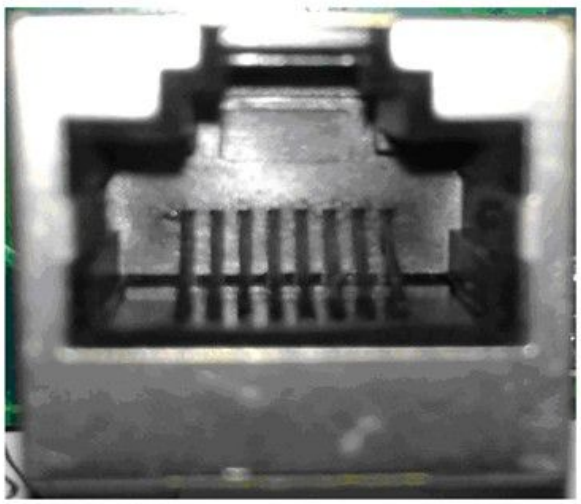
```
//cell vol flag:
public byte MSK_V_BAL   =       0x80;    //Balance flag
public byte MSK_V_OV    =       0x40;    //Over charge Voltage flag
public byte MSK_V_UV    =       0x20;    //Under discharge Voltage flag
for(int i = 0; i < CELL_COUNT; i++)
{
    //get balance flag,then clear it
    if ((rxbuf[p] & o.MSK_V_BAL) > 0)
    {
        rxbuf[p] &= (byte)(~o.MSK_V_BAL);
    }
    //get OV flag,then clear it
    if ((rxbuf[p] & o.MSK_V_OV) > 0)
```

```
        {
            rxbuf[p] &= (byte)(~o.MSK_V_OV);
        }
        //get UV flag,then clear it
        if ((rxbuf[p] & o.MSK_V_UV) > 0)
        {
            rxbuf[p] &= (byte)(~o.MSK_V_UV);
        }
        Vcell[i] = (int)(rxbuf[p++] * 256 + rxbuf[p++]);
    }
```

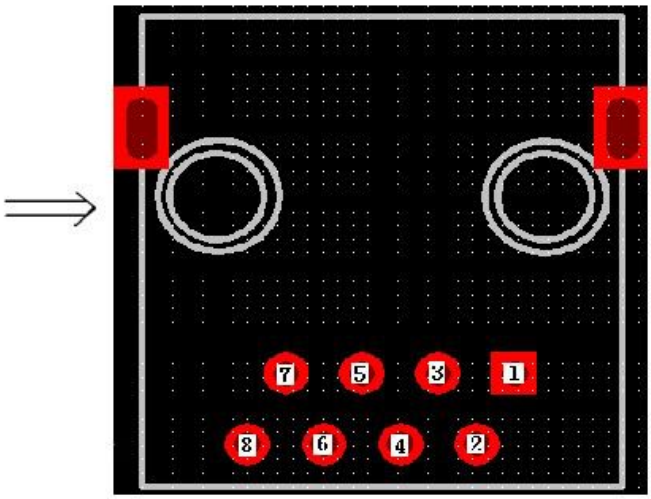
8 Attention:

485 communication to wakeup dormant can not be effective by one command, it need continuously send command(any command) until wakeup. Normally, it need 20 command to wakeup MCU. Please note to be continuous, don't stop or have gap.

9 Connection Description:



正面视图



正面视图

1脚 GND 连 地
2脚 RS485 A 连T/R+
3脚 RS485 B 连T/R-