Sentinel PMS5005 Protocol Reference Manual

With Support for Constellation System and Power Controller

Draft Version

SEP. 2006

Table of Contents

I.   Introduction

This reference manual provides detail information on the Sentinel motion controller (PM5005) and power controller communication protocol. This protocol is used to allow any processor, DSP, or PC to control the Sentinel through the UART communication interface with setting of 115200kbps, N, 8, 1.

Note that in using this protocol; please make sure that Dr Robot's Sentinel Gateway software on PC is closed.

II.   Packet Format

Each data packet consists of the following basic components:

       1. Start-of-transmission indicator (STX)

       2. Robot ID (RID)

       3. Reserved byte for future use

       4. Data ID (DID)

       5. A Length field to specify the total number of byte in the Data field

       6. Data, if any

       7. A checksum

       8. An end-of-transmission indicator (ETX)

The format of the frame is illustrated as follows:

STX	RID	Reserved	DID	LENGTH	DATA	CHECKSUM	ETX

II.1.   Start-of-Transmission (STX)

STX contains two bytes and is used as an indicator of the beginning of each packet. In the system, 94 | 02 (0x5E | 0x02) is used to represent STX.

II.2.   Robot ID (RID)

Each device will have an ID to represent itself. This one byte ID field is used to store the destination of the packet. Currently, we define as

               0 ---- Host (e.g. PC)

               1 ---- PMS5005 or Power controller

               8 ---- PMB5010 (not supporting in the current version)

Other values are reserved for future use.

II.3.   Data ID (DID), Length, and Data

The one byte data ID is used to identify the command with a one byte length field to specify the number of bytes in the Data field. As a result, the Data field can contain up to 255 bytes data. The following table is a list of pre-defined Data ID and DATA format sending from host to PMS5005 Motion & Sensing Controller or power controller

Table II.1 Detail Description of Each Data ID and DATA Field

DID	Definition	DATA Field Contents			
3	Move specified DC motors to a given target position *require setting of potentiometer or optical encoder	- [Motor channel (1 byte) + Target position (low 8bit + high 8 bit)+ ....] + FLAG + Time (2 bytes) - DC motor channel # is from 0 to 5 - Target position is from 0 to 32767 pulses for optical encoder, 0 to 4095 for single potentiometer, or 0 to 4428 for dual potentiometers - FLAG = 6 - Time is the numbers of control period and the movement would be under linear trajectory planning - If point control is preferred, user does not need to supply the FLAG and time values			
4	Move all 6 DC motors to a given target position *require setting of potentiometer or optical encoder	- MOT1 position (2 bytes) + MOT2 position (2 bytes) + MOT3 position (2 bytes) + MOT4 position (2 bytes) + MOT5 position (2 bytes) + MOT6 position (2 bytes) + Time (2 bytes) - If point control is preferred, user does not need to supply the time value - Please also refer to DID = 3			
5	Move specified DC motors with direct PWM control	-[Motor channel # (1 byte) + PWM (low 8bit + high 8 bit)+ ....] + FLAG + Time(2 bytes) - DC motor channel # is from 0 to 5 - PWM value is from 0 to 32767 to represent 0 to 100% pulse cycle - FLAG = 6 - Time is the numbers of control period and the movement would be under linear trajectory planning - If point control is preferred, user does not need to supply the FLAG and time values			
6	Move all 6 DC motors with direct PWM control	- MOT1 PWM (2 bytes) + MOT2 PWM (2 bytes) + MOT3 PWM (2 bytes) + MOT4 PWM (2 bytes) + MOT5 PWM (2 bytes) + MOT6 PWM (2 bytes) + Time (2 bytes) - If point control is preferred, user does not need to supply the time value - Please also refer to DID = 5			
7	System Parameters Settings Having many sub commands which can be single or compound	**Sub-Command**	**Sub-Command Parameters**		

DID = 7 Sub-Commands

Sub-Command	Sub-Command Parameters			
0x06: Motor Polarity Direction	- Motor # (1 byte) + motor polarity direction (1 byte) - DC motor # is from 0 to 5 - Motor's polarity direction: 1 = positive, -1 (0xff) = negative - if the PWM cycle is greater 50% (16384 - 32768) and the sensor (potentiometer or encoder) value increases, this means positive. IF sensor value decreases, it means negative.			
0x07: PID Control Parameters Setting for Position Control Loop	1st Byte	2nd Byte	3rd Byte	4th Byte
	DC Motor Channel # (0 to 5)	0x01: Kp	Kp Low Byte	Kp High Byte
		0x02: Kd	Kd Low Byte	Kd High Byte
		0x03: KI Format: M+1/N	KI's Integer Part M	KI's Fraction Part N
0x08: PID Control Parameters Setting for Velocity Control Loop	1st Byte	2nd Byte	3rd Byte	4th Byte
	DC Motor Channel # (0 to 5)	0x01: Kp	Kp Low Byte	Kp High Byte
		0x02: Kd	Kd Low Byte	Kd High Byte
		0x03: KI Format: M+1/N	KI's Integer Part M	KI's Fraction Part N
0x0d: Sensor Usage Select for DC motor	1st Byte: DC Motor Channel # (0 to 5)	2nd Byte: 0x00 --- Single potentiometer 0x01 --- Dual potentiometer 0x02 --- Encoder		
0x0e: Control Method Select	1st Byte: DC Motor Channel # (0 to 5)	2nd Byte: 0x00 --- Direct PWM Control 0x01 --- Direct Position Close-loop Control 0x02 --- Direct Velocity Close-loop Control		
0x13: DRK6000/8000 Use Only	- 0: disable collision protect - 1: enable collision protect			

DID	Definition	DATA Field Contents
22	Set the custom GPIO output	- 1 byte data where MSB --- LSB (bit7---bit0) stands for GPIO7-----GPIO0 output value

22*	Send control command to power controller	Len =1 data = 1 byte command to power controller(in this version power1 is camera power, power2 is motion controller and PDA power, power3 is servo and servo controller power.)

0	Turn On/Off Power1 0 – turn Off, 1- Turn On
1	Reset Power2 0- 200ms reset
2	Turn On/Off Power3 0 – Turn Off, 1- Turn On
3	Select system powered by DCIN
4	Select system powered by Bat2
5	Select system powered by Bat1
6	Charge Bat2 1 – charge 0 – stop charge
7	Charge Bat1 1—charge 0 – stop charge

23	Control the display of the LCD screen LCD is operating in graphics mode, and user working in pixel writing mode	- 1st Byte : the serial Frame number (0 to 15) of LCD Display Packet; Even frame display in left side, odd number display in right side, from up to down. - 2nd Byte – 65th Byte : the LCD display packet of that frame - Every packet will contain data for 8 rows (total 64 bytes) - Every 16 packets represents one display frame (according to the whole LCD display 128*64)
26	Move specified DC motors with velocity control *require setting of potentiometer or optical encoder	- [Motor channel # (1 byte) + Target velocity (low 8bit + high 8 bit)+ ....] + FLAG + Time (2 bytes) - DC motor channel # is from 0 to 5 - Target velocity is from 0 to 32767 for pulse/s if optical encoder is used, actual range depends on what kind of encoder is used. If dual potentiometer is used, range from 0 to 4482, which is in sampling space. - FLAG = 6 - Time is the numbers of control period and the movement would be under linear trajectory planning - If point control is preferred, user does not need to supply the FLAG and time values
27	Move all DC motors with velocity control *require setting of potentiometer or optical encoder	- MOT1 velocity (2 bytes) + MOT2 velocity (2 bytes) + MOT3 velocity (2 bytes) + MOT4 velocity (2 bytes) + MOT5 velocity (2 bytes) + MOT6 velocity (2 bytes) + Time (2 bytes) - If point control is preferred, user does not need to supply the time value - Please also refer to DID = 26
28	Move specified servo motor	- [Servo # (1 byte) + Target position (low 8bit + high 8 bit)+ ....] + FLAG + Time (2 bytes) - Servo channel # is from 0 to 5 - Position is roughly from 1000 to 6000 since the range is servo dependent - FLAG = 6 - Time is the numbers of control period and the movement would be under linear trajectory planning - If point control is preferred, user does not need to supply the FLAG and time values
29	Move all 6 servo motors	- SERVO1 position (2 bytes) + SERVO2 position (2 bytes) + SERVO3 position (2 bytes) + SERVO4 position (2 bytes) + SERVO5 position (2 bytes) + SERVO6 position (2 bytes) + Time (2 bytes) - If point control is preferred, user does not need to supply the time value - Please also refer to DID = 28
30	Suspend/resume DC motors or enable/disable servos	- 0: suspend all DC and servo motors - 1: resume all DC and servo motors - mode + motor channel #: this is used to suspend/resume specified motors or enable/disable specified servo. Mode = 0 stands for suspend/disable and mode = 1 stands for resume/enable. Motor channel # 0---5 is for DC motor; 6---11 is for servo;
80	Set constellation sensor ID	Len = 4, Data = ID1, ID2,ID3,ID4, the PM5005 will scan all the four sensors to get the each distance between the transponder and car sensor.

123	Get Motor control signal feedback	- None: send back data to host continuously - N: send back data N times to host where N is from 1 to 255 - 0: stop sending back data to host
124	Request for custom I/O and A/D data feedback	- None: send back data to host continuously - N: send back data N times to host where N is from 1 to 255 - 0: stop sending back data to host
124*	Power controller information	Detail in II.3.4
125	Get sensor data feedback	- None: send back data to host continuously - N: send back data N times to host where N is from 1 to 255 - 0: stop sending back data to host
127	Get all sensor feedback	- None: send back data to host continuously - N: send back data N times to host where N is from 1 to 255 - 0: stop sending back data to host
255	Setup System communication	- Len = 1, data = 1: host to ping PMS5005
		- Len = 1, data = 0: PMS5005 ping PC periodically with this packet - Len = 1, data = 1: PMS5005 acknowledge data packet (including ping packet) from host with this packet

II.3.1.   Motor Control Signal Data Format

After a request from host for motor control signal data (DID = 123 or 127) is received, PMS5005 will send the host the data with the following data field format. For detail information for each return value, please refer to the WiRobot SDK Application Programming Interface Reference Manual.

Table II.2 Motor Control Signal Data Field Format

Data buffer position	Data range	Data meaning
Byte1	0x00~0xff	Potentiometer-1 A/D sample value
Byte2	0x00~0x0f	
Byte3	0x00~0xff	Potentiometer-2 A/D sample value
Byte4	0x00~0x0f	
Byte5	0x00~0xff	Potentiometer-3 A/D sample value
Byte6	0x00~0x0f	
Byte7	0x00~0xff	Potentiometer-4 A/D sample value
Byte8	0x00~0x0f	
Byte9	0x00~0xff	Potentiometer-5 A/D sample value
Byte10	0x00~0x0f	
Byte11	0x00~0xff	Potentiometer-6 A/D sample value
Byte12	0x00~0x0f	
Byte13	0x00~0xff	Motor-1 current feedback A/D sample value
Byte14	0x00~0x0f	
Byte15	0x00~0xff	Motor-2 current feedback A/D sample value
Byte16	0x00~0x0f	
Byte17	0x00~0xff	Motor-3 current feedback A/D sample value
Byte18	0x00~0x0f	
Byte19	0x00~0xff	Motor-4 current feedback A/D sample value
Byte20	0x00~0x0f	
Byte21	0x00~0xff	Motor-5 current feedback A/D sample value
Byte22	0x00~0x0f	
Byte23	0x00~0xff	Motor-6 current feedback A/D sample value
Byte24	0x00~0x0f	
Byte25	0x00~0xff	Encoder-1 pulse count value
Byte26	0x00~0xff	
Byte27	0x00~0xff	Encoder-1 speed
Byte28	0x00~0xff	

Byte29	0x00~0xff	Encoder-2 pulse count value	
Byte30	0x00~0xff		
Byte31	0x00~0xff	Encoder-2 speed	
Byte32	0x00~0xff		
Byte33	0x00~0x03	LSB-bit0: Encoder-1 direction	1 --positive
		LSB-bit1: Encoder-2 direction	0---negative

Note: Splitting one word to two bytes, we put LSB into first byte and MSB into second byte

II.3.2. Custom IO Port and A/D Channel Data Format

After a request from host for custom IO port and A/D channel format (DID = 124 or 127) is received, PMS5005 will send the host the data with the following data field format. For detail information for each return value, please refer to the WiRobot SDK Application Programming Interface Reference Manual.

Table II.3 Custom IO Port and A/D Channel Data Field Format

Data buffer position	Data range	Data meaning
Byte1	0x00~0xff	Custom A/D channel –1 A/D value
Byte2	0x00~0x0f	
Byte3	0x00~0xff	Custom A/D channel –2 A/D value
Byte4	0x00~0x0f	
Byte5	0x00~0xff	Custom A/D channel –3 A/D value
Byte6	0x00~0x0f	
Byte7	0x00~0xff	Custom A/D channel –4 A/D value
Byte8	0x00~0x0f	
Byte9	0x00~0xff	Custom A/D channel –5 A/D value
Byte10	0x00~0x0f	
Byte11	0x00~0xff	Custom A/D channel –6 A/D value
Byte12	0x00~0x0f	
Byte13	0x00~0xff	Custom A/D channel –7 A/D value
Byte14	0x00~0x0f	
Byte15	0x00~0xff	Custom A/D channel –8 A/D value
Byte16	0x00~0x0f	
Byte17	0x00~0xff	- 1 byte data where MSB --- LSB (bit7---bit0) stands for input I/O port7-----input I/O port 0 input value
Byte18	0x00-0xff	The distance from the constellation left sensor to ID1 transponder.(unit: mm)
Byte19	0x00-0xff	
Byte20	0x00-0xff	The distance from the constellation left sensor to ID2 transponder. .(unit: mm)
Byte21	0x00-0xff	
Byte22	0x00-0xff	The distance from the constellation left sensor to ID3 transponder. .(unit: mm)
Byte23	0x00-0xff	
Byte24	0x00-0xff	The distance from the constellation left sensor to ID4 transponder. .(unit: mm)
Byte25	0x00-0xff	
Byte26	0x00-0xff	The distance from the constellation right sensor to ID1 transponder. .(unit: mm)
Byte27	0x00-0xff	
Byte28	0x00-0xff	The distance from the constellation right sensor to ID2 transponder. .(unit: mm)
Byte29	0x00-0xff	
Byte30	0x00-0xff	The distance from the constellation right sensor to ID3 transponder. .(unit: mm)
Byte31	0x00-0xff	
Byte32	0x00-0xff	The distance from the constellation right sensor to ID4 transponder. .(unit: mm)
Byte33	0x00-0xff	
Byte34	0x00-0xff	Get the ID of constellation transponder that related with distance of ID1
Byte 35	0x00-0xff	Get the ID of constellation transponder that related with distance of ID2

| Byte 36 | 0x00-oxff | Get the ID of constellation transponder that related with distance of ID3 |
| Byte 37 | 0x00-oxff | Get the ID of constellation transponder that related with distance of ID4 |

Note: Splitting one word to two bytes, we put LSB into first byte and MSB into second byte

II.3.3. Sensor Data Format

After a request from host for sensor data (DID = 125 or 127) is received, PMS5005 will send the host the data with the following data field format. For detail information for each return value, please refer to the WiRobot SDK Application Programming Interface Reference Manual.

Table II.4 Sensor Data Field Format

Data buffer position	Data range	Data meaning
Byte1	0x00~0xff	Ultrasonic Range-1 value (Unit cm)
Byte2	0x00~0xff	Ultrasonic Range-2 value (Unit cm)
Byte3	0x00~0xff	Ultrasonic Range-3 value (Unit cm)
Byte4	0x00~0xff	Ultrasonic Range-4 value (Unit cm)
Byte5	0x00~0xff	Ultrasonic Range-5 value (Unit cm)
Byte6	0x00~0xff	Ultrasonic Range-6 value (Unit cm)
Byte7	0x00~0xff	Human sensor-1 alarm channel A/D value
Byte8	0x00~0x0f	
Byte9	0x00~0xff	Human sensor-1 motion detect channel A/D value
Byte10	0x00~0x0f	
Byte11	0x00~0xff	Human sensor-2 alarm channel A/D value
Byte12	0x00~0x0f	
Byte13	0x00~0xff	Human sensor-2 motion detect channel A/D value
Byte14	0x00~0x0f	
Byte15	0x00~0xff	Tilting sensor X-axis A/D value
Byte16	0x00~0x0f	
Byte17	0x00~0xff	Tilting sensor Y-axis A/D value
Byte18	0x00~0x0f	
Byte19	0x00~0xff	Overheat sensor-1 A/D value
Byte20	0x00~0x0f	Motor-driver board –1
Byte21	0x00~0xff	Overheat sensor-2 A/D value
Byte22	0x00~0x0f	Motor-driver board –2
Byte23	0x00~0xff	Temperature sensor A/D value
Byte24	0x00~0x0f	
Byte25	0x00~0xff	IR Range sensor A/D value
Byte26	0x00~0x0f	
Byte27	0x00~0xff	Infrared command
Byte28	0x00~0xff	
Byte29	0x00~0xff	
Byte30	0x00~0xff	
Byte31	0x00~0xff	Main board battery power A/D value (0 – 9V)
Byte32	0x00~0x0f	
Byte33	0x00~0xff	DC motor battery power A/D value (0 – 24V)
Byte34	0x00~0x0f	
Byte35	0x00~0xff	Servo battery power A/D value (0 - 9V)
Byte36	0x00~0x0f	
Byte37	0x00~0xff	Reference voltage detect A/D value (Vcc)
Byte38	0x00~0x0f	

Byte39	0x00~0xff	Potentiometer power detect A/D value (Vref)
Byte40	0x00~0x0f	

Note: When splitting one word to two bytes, we put LSB into first byte and MSB into second byte

II.3.4. Power controller Information data package

After a request from host for power information (DID = 124) is receives by power controller, it will send the information data package (DID =124) to host with following data field format, and by default the power controller will send back this data package every 100ms.

Data buffer position	Data meaning
Byte 1	Battery 1 Voltage lower 8 bit
Byte 1	Battery 1 Voltage higher 8 bit
Byte 1	Battery 1 Temperature lower 8 bit
Byte 4	Battery 1 Temperature higher 8 bit
Byte 5	Battery 2 Voltage lower 8 bit
Byte 6	Battery 2 Voltage higher 8 bit
Byte 7	Battery 2 Temperature lower 8 bit
Byte 8	Battery 2 Temperature higher 8 bit
Byte 9	DCIN Voltage lower 8 bit
Byte 10	DCIN Voltage higher 8 bit
Byte 11	reserve
Byte 12	reserve
Byte 13	reserve
Byte 14	reserve
Byte 15	reserve
Byte 16	reserve
Byte 17	Power controller Status*
Byte 18	reserve
Byte 19	reserve
Byte 20	Power controller A/D reference voltage lower 8bit
Byte 21	Power controller A/D reference voltage higher 8bit
Byte 22	reserve
Byte 23	reserve
Byte 24	reserve
Byte 25	reserve
Byte 26	reserve
Byte 27	reserve
Byte 28	Power controller select power path*
Byte 29	X
Byte 30	Power controller select charge path*
Byte 31	X

Note:

a. Power controller status - 1 byte

Bit	Description
0	X
1	X
2 - CH	1 -- Charging 0 – no charge
3 – PF	Power fail
4 -DC	DCDIV comparator output
5 – LP	Lower power

6—FA	Fault(on reset, short circuit, shut down by DCDIV
7	X

b. Power path – only 1 byte

Bit	Description
0	X
1	X
2	X
3	X
4	X
5	Powered by DCIN
6	Powered by Battery 2
7	Powered by Battery 1

c. Charge path – only 1 byte

Bit	Description
0	X
1	X
2	X
3	X
4	X
5	X
6	Charge Battery 2
7	Charge Battery 1

II.4. Checksum

Checksum is calculated over all the fields of each data packet, except the STX and ETX. This checksum is used to determine any packet transmission error. The sender is required to append the checksum in each packet and the receiver is required to verify the checksum to see if any transmission error. Note that any value can be used in the reserved byte for calculation purpose but please make sure that this value will be stored in the reserved field of the packet.

The following is a C sample code that is used in PMS5005 for calculating the checksum. The variable lpBuffer stores the content of a data packet excluding the STX and ETX in an array and the variable nSize stores the size of the data (from RID to DATA fields) in the lpBuffer array.

Figure II.1 Sample C Code of Checksum Calculation

```
BYTE CalculateCRC(char *lpBuffer, int nSize)
{
        BYTE shift_reg, sr_lsb, data_bit, v;
        int i, j;
        BYTE fb_bit;

        shift_reg = 0; // initialize the shift register

        for (i = 0 ; i < nSize ; i++)
        {
                v = (BYTE) (lpBuffer[i] & 0x0000FFFF);
                for (j = 0 ; j < 8 ; j++) // for each bit
                {
                        data_bit = v & 0x01; // isolate least sign bit
                        sr_lsb = shift_reg & 0x01;
                        fb_bit = (data_bit & sr_lsb) & 0x01; // calculate the feed back bit
                        shift_reg = shift_reg >> 1;
```

```
                            if (fb_bit == 1)
                                        shift_reg = shift_reg & 0x8C;
                            v = v >> 1;
                }
        }
        return shift_reg;
}
```

II.5.   End-of-Transmission (ETX)

STX contains two bytes and is used as an indicator of the end of each packet. In the system, 94 | 13 (0x5E | 0x0D) is used to represent ETX.

III. Packet Exchange between Host and PMS5005

When a receiver received a packet, it will first open the packet and use the checksum to confirm that there is no packet transmission error. Then, it will inspect the RID field of the packet to determine if they are the intended receiver as well as to double check that the length of the data field is the same as the number specified in the LENGTH field. If all these criteria are met, the receiver can act now to the received command.

As well, when PMS5005 receive a data packet, it will respond an acknowledgement packet (as long as the reserved byte is not 255) to signal the sender that that packet is received correctly. The acknowledge packet is shown as follows (assume the packet sender use 0 in the reserved field):

STX = 94 2, RID = 0, Reserved = 0, DID = 255, LENGTH = 1, DATA = 1, CHECKSUM = 72, ETX = 94 13

Note that users should not send any other DID value, except those defined above. As well, when PMS5005 boots up, it will keep sending all sensor data to the host for processing.

III.1. Example

The followings are few examples in demonstrating how to control the PMS5005 through a host controller (e.g. PC). Note that we use 0 in the reserved field.

III.1.1. Control Servo 3 to Position 2048 with No Time Control

In order to move servo 3 (channel 2) to position 2048 (00001000 00000000 in binary), the host (PC) has to send the following command:

STX = 94 2, RID = 1, Reserved = 0, DID = 28, LENGTH = 3, DATA = 2 0 8, CHECKSUM = 101, ETX = 94 13

III.1.2. Suspend Servo 2's Movement

The following command will suspend the movement of servo 2 (channel 1):

STX = 94 2, RID = 1, Reserved = 0, DID = 30, LENGTH = 2, DATA = 0 7, CHECKSUM = 214, ETX = 94 13

III.1.3. Control DC Motor 5 with PWM Value 4000

The following command will control the movement of DC motor 5 (channel 4) with PWM value 4000 (00001111 10100000):

STX = 94 2, RID = 1, Reserved = 0, DID = 5, LENGTH = 3, DATA = 4 160 15, CHECKSUM = 86, ETX = 94 13

III.1.4. Request Motor Control Signal Feedback 3 Times

The following command will request the motor control signal feedback 3 times:

STX = 94 2, RID = 1, Reserved = 0, DID = 123, LENGTH = 1, DATA = 3, CHECKSUM = 197, ETX = 94 13

After the PMS5005 received this command, it will send back the motor control signal 3 times to the host with the following format

STX = 94 2, RID = 0, Reserved = 255, DID = 123, LENGTH = 33, DATA, CHECKSUM, ETX = 94 13

where the format of DATA is shown in Table II.2 and the CHECKSUM depends on the actual values in the DATA field.

III.1.5. Control DC Motor 1 to Position 6000 by Using a Quadrature Encoder

In order to control the DC motor 1 (channel 0) with position feedback, the host has to first send the following commands to setup the sensor usage to "encoder" and the DC motor control to "position" method:

STX = 94 2, RID = 1, Reserved = 0, DID = 7, LENGTH = 3, DATA = 13 0 2, CHECKSUM = 72, ETX = 94 13

STX = 94 2, RID = 1, Reserved = 0, DID = 7, LENGTH = 3, DATA = 14 0 1, CHECKSUM = 78, ETX = 94 13

After the setup, the host can now send another command in asking the DC Motor 1 to the position 6000 (00010111 01110000) with no time control:

STX = 94 2, RID = 1, Reserved = 0, DID = 3, LENGTH = 3, DATA = 0 112 23, CHECKSUM = 83, ETX = 94 13

III.1.6. Send an image to the LCD display

Sending a complete image to LCD display requires several packets from host to PMS5005. Since the LCD display MGL5128 is 128x64, sending a complete image will require a delivery of 8192 bits (1024 bytes). In our system, each LCD image is represented by 16 packets.

III.1.7. Send constellation transponder ID to PM5005

Host can send 4 transponder sensor ID to PM50005. Here is the example to set sensor ID to 1,2,3,4.

STX 94 2, RID = 1, Reserved = 0, DID = 80, Length = 3, DATA = 1,2,3,4,

CHECKSUM = 134, EXT = 94,13

III.1.8. Send command to power controller

Host can send command to power controller, for selecting power source, turning on, turning off or resetting some modules.

Here is the example to select Battery1, Batter2 as power source, and turn on WiFi, PM5010, PM5005 module, and do not charge battery1, battery2.

STX 94 2, RID = 1, Reserved = 0, DID = 22, Length = 1, DATA=55,

CHECKSUM = 175, ETX = 94, 13

The controller information will send back in data package with DID =124.