alexruhmann@body-soft.de

PROTOCOL OF TRACER SEREIS AND MT-5 Ver 3
(REVISED ON 13, Dec.2011.)

1, instruction on communication date format.

PWL start up 6 byte, （AAH、55H、AAH、55H、AAH、55H）

+ communication synchronization, the first 6 byte （EBH、90H、EBH、90H、EBH、90H）

+ controller ID(1 byte)

+ command (1 byte) (see details in part3: instructions on command)

+ data packet length (1 byte, bytes of data content , maximumly don't exceeds 200 bytes)

+ data content (low byte in front)

+ CRC Cyclic redundancy check code (2 byte, high byte in front)

+7FH
e.g.: computer send a real time monitoring command string to E01 series controller(ID=16H) :
AA 55 AA 55 AA 55 EB 90 EB 90 EB 90 16 A0 00 B1 A7 7F

For example , responds is as following:

AA 55 AA 55 AA 55 EB 90 EB 90 EB 90 12 bytes to start the synchronization head

16 A0 18 ID、command、data length

CE 04 A4 06 00 00 E8 03 56 04 A0 05 01 00 00 50 00 00 00 00 37 E8 03 00 17 datas

9A 38 7F 2 byte check code, 1 byte exit code

二、Pre description as following:

1、 Controller cannot identify PWL start up 6 byte AA 55 AA 55 AA 55, in order to save , some device won't send , so the host computer is not sensible to such 6 byte AA 55

AA 55 AA 55.

2、 6 byte PWL start up + 6 byte communication synchronization is 12 byte, abbreviated as "synchronous head"
"

3、 Floating point data such as voltage, current , etc, are indicates as 100 times integer of the actual value, e.g., 12.5V, showed as 1250 (0x04E2 (hexadecimal 2-byte integer). Please switch by yourself.

4、 Order of integers of more than 2 byte are transferred in sequence : Low byte in front and high byte in behind.

5、 ID is 1 byte data, the present version won't deal with it now.

6、 The following commands added shall follow the above principals.

三、Command byte definitions are as following:

1、 Command A0H(read the real time collected data) : used for real time monitoring

Special delivery format and respond format:

Host send: synchronous head + ID + A0H + 00H + CRCH + CRCL + 7FH

Sub device send back: synchronous head + ID + A0H + 0DH + collected data strings + CRCH + CRCL + 7FH

The data strings content and sequence to be collected:

No.	items	bytes	instructions
1	Battery voltage	2 byte	e.g., 12.5V(1250 showed as 0x04E2.low byte is in front when sending.
2	Pv voltage	2 byte	same as above
3	Reserved	2 byte	"0" all time.
4	Load current	2 byte	e.g. 10.0A(1000 showed as 0x03E8, low byte is in front when sending.
5	over discharge voltage	2 byte	same as above
6	Battery full voltage	2 byte	same as above
7	Load on/off	1 byte	=0, load off; =1, load on
8	Over load	1	=0, normal ; =1, load current over load, over load

		byte	protection would happen.
9	Load short circuit	1 byte	=0, normal; =1, load short circuit, output would be stopped.
10	reserved	1 byte	Not used
11	Battery overload	1 byte	=0, normal; =1, battery voltage over voltage, controller would stop charging and discharging.
12	Over discharge	1 byte	=0, normal, =1, battery voltage too low and output would be stopped.
13	Full indicator	1 byte	=0, not fully charged; =1, battery voltage is full
14	Charging indicator	1 byte	=0, not charging; =1, being charged.
15	Battery temp.	1 byte	E.g, 25℃(55 showed as 0x37, there is 30 difference in value with the real value.
16	Charging current	2 byte	e.g. 10.0A(1000 showed as 0x03E8。Low byte in front when sending.
17	reserved	1byte	0

2. Command 0xAA: (manual control command):

used for controlling of load switch through remote meter or computer

Host send: synchronous head + ID + AAH + 01H + control switch command + CRCH + CRCL + 7FH

Sub device send back: synchronous head + ID + AAH + 01H + load switch state + CRCH + CRCL + 7FH

Control switch command:

=1, Load ON

=0, Load OFF

Load switch state:

=1, Load ON

=0, Load OFF

3. Command 0xAD: remote meter or computer send control data:

Format:

Host send: synchronous head + ID + ADH + 00H + control parameter data strings + CRCH + CRCL + 7FH

Sub device send back: synchronous head + ID + ADH + 00H + CRCH + CRCL + 7FH

Control parameter data strings:

No.	Items	bytes	instructions
1	Load type	1byte	unuse

2	Charging mode	1byte	unuse
3	Temperature compensation coefficienct	1byte	0~10
4	Battery type	1byte	0: gel，1: sealed，2: flooded
5	Battery capacity	1byte	Show 1/10 of normal battery capacity (AH), need recover process
6	Load control model 1	1byte	As below chart
7	Load control mode 2	1byte	As below chart

Load control mode 1 code

code	instruction
0	Dusk to Dawn
1~15	1~15 hours timer
16	Test
17	Manual
18	unable

Load mode 2 code only include 1~15、18。

CRC process example:

CRC_Buff –

The first address of data strings to be proceed

crc_len -- Byte numbers to be processed, from ID byte to final byte of CRC check result.

```
u16 Crc(u8 *CRC_Buff,u8 crc_len)
{
    u8 crc_i,crc_j,r1,r2,r3,r4;
    u16 crc_result;

    r1=*CRC_Buff;
    CRC_Buff++;
    r2=*CRC_Buff;
    CRC_Buff++;
    for (crc_i=0;crc_i<crc_len-2;crc_i++)
    {
        r3=*CRC_Buff;
        CRC_Buff++;
```

```
            for (crc_j=0;crc_j<8;crc_j++)
            {
                r4=r1;
                r1=(r1<<1);
                if ((r2&0x80)!=0)r1++;
                r2=r2<<1;
                if((r3&0x80)!=0)r2++;
                r3=(r3<<1);
                if ((r4&0x80)!=0)
                {
                   r1=r1^0x10;
                   r2=r2^0x41;
                }
            }
        }
        crc_result=r1;
        crc_result=crc_result<<8 | r2;
        return(crc_result);
}
```

example.:

```
Main()
{
 …
 //
calculate the CRC checkword before sending
 t_buf[t_buf[2]+3] = 0;        //   remove all command.
 t_buf[t_buf[2]+4] = 0;
 CRC_Result=Crc(t_buf,t_buf[2]+5);    //   calculate
 t_buf[t_buf[2]+3] = CRC_Result>>8;
 t_buf[t_buf[2]+4] = CRC_Result & 0xFF;
 t_buf[t_buf[2]+5] = 0x7F;        // exit code
 …
 …

}
```

Connector Pin definitions:

Pin number	definition
1	Power supply + 12V output
2	Power ground/signal ground
3	Power +12V output

4	Power ground/signal ground
5	TXD, the sender
6	RXD, the receiving end
7	Power ground/signal output
8	Power ground/signal output

RJ45crystal plus pin number as shown