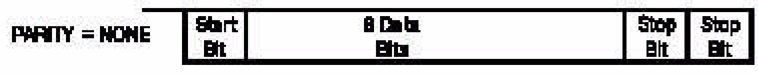
Control protocol for Power supply

Power supply can be connected to PC by the DB9 plug on the rear panel via 3311 or 3312 adapter. The following instructions can help you to know how to control the power supply by PC.

A. Default Serial Communications Port Settings

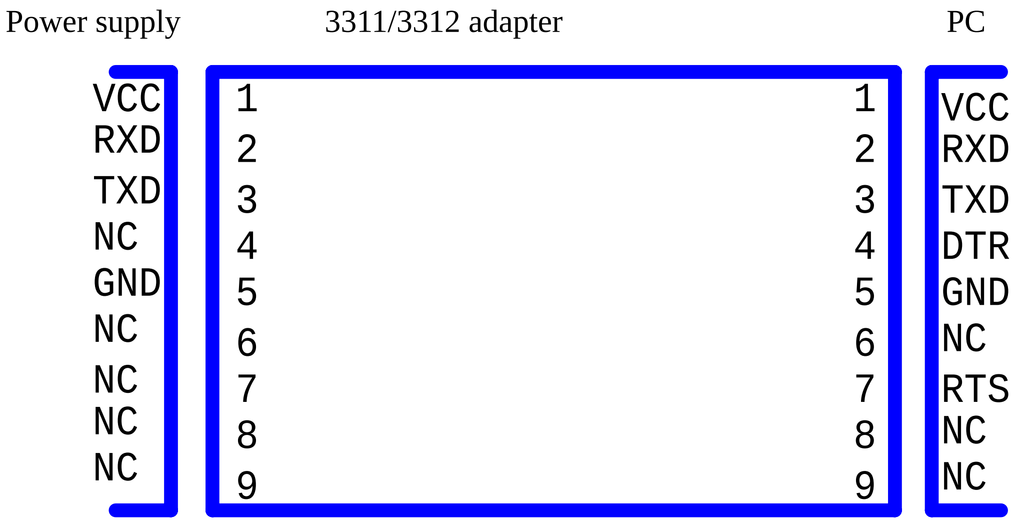
You can set the communication baudrate and the address of the power supply using the keyboard.

1. Address: (0-31) 00h-FEh
2. Baud rate: 9600 (4800, 9600, 19200, 38400)
3. Data bits: 8
4. Stop bitd: 1
5. handshake: None



B. DB9 Serial interface

The output of DB9 interface in the rear back of the unit is TTL, and you have to use 3311 or 3312 adapter to connect it to the PC Com port.



C. Frame format(applies to both transmitted and received date)

The frame length is 26 bytes with the following format:

AAh	Address	Command	Relative information :Byte 4-25	Checksum

Description of frame bytes:

1) The first byte of the frame is always AAh

2) The second byte is the power supply address(00h to FEh as set using front panel menu)

3) The third byte is the instrument control command

These are the possible commands:

a) 80h------Set max current, max power and set-value.

b) 81h------Read current, voltage, power and power supply’s state. The states include ON/OFF, over current and over power status of the power supply.

c) 82h------To control the ON/OFF state of the power supply.
d) 83h------Set the protection state of power supply.
e) 84h------Read the protection state of power supply.
f) 85h------Demarcate the power command.
g) 86h------Return the actual output voltage to power supply .
h) 87h------Demarcate the current command.
i) 88h------Return the actual output voltage to power supply.
j) 89h------Set the demarcating information of power supply.
k) 8Ah------Read the demarcating information.
l) 8Bh------Set the serial number of power supply.
m) 8Ch------Read the serial number, product model and software version of the power Supply.
n) 12h-------Check.

4)If you want to control the output of the power supply by PC, you have to set the power supply at PC control state, and the command is 82h. If you want to demarcate the output of the power supply and set the demarcating information and serial number of the power supply, you have to set the protection status to OFF first.

5)Byte26 is the checksum obtained by adding the values of the previous 25 bytes.

D. Command Descriptions

1) 80h, Set power supply operating parameters and maximum limits

Byte 1	Frame start (AAh)
Byte 2	Addresses (00h-FEh)
Byte 3	Command (80h)
Byte 4	Low byte of the max current
Byte 5	High byte of the max current
Byte 6	Low byte of the low character of the Max voltage
Byte 7	High byte of the low character of the Max voltage
Byte 8	Low byte of the high character of the Max voltage
Byte 9	High byte of the high character of the Max voltage
Byte 10	Low byte of the Max power
Byte 11	High byte of the Max power
Byte 12	Low byte of the low character of the voltage set
Byte 13	High byte of the low character of the voltage set

Byte 14	Low byte of the high character of the voltage set
Byte 15	High byte of the high character of the voltage set
Byte 16	New address of the power supply
Byte 17~25	System Reserved
Byte 26	Checksum

The set-values for current, power are all expressed by two bytes.The set voltage is expressed by four bytes. The low byte is sent first.

For example: The current value 3589h is specified by the following sequence;

89h	35h

The voltage range of 0-36V is represented by an integer in the range of 0-36000mV

The current range of 0-3A is represented by an integer in the range of 0-3000mA

The power range of 0-108W is represented by an integer in the range of 0-108W

2)81h,Read the current, voltage, power value and the status of the power supply

Byte 1	Frame start (AAh)
Byte 2	Address (00h-FEh)
Byte 3	Command（81h）
Byte 4	Low byte of current
Byte 5	High byte of current
Byte 6	Low byte of the low character of the voltage
Byte 7	High byte of the low character of the voltage
Byte 8	Low byte of the high character of the voltage
Byte 9	High byte of the high character of the voltage
Byte 10	Low byte of power
Byte 11	High byte of power
Byte 12	Low byte of Max current
Byte 13	High byte of Max current
Byte 14	Low byte of the low character of the Max voltage

Byte 15	High byte of the low character of the Max voltage
Byte 16	Low byte of the High character of the Max voltage
Byte 17	High byte of the High character of the Max voltage
Byte 18	Low byte of Max power
Byte 19	High byte of Max power
Byte 20	Low byte of the low character of the voltage set
Byte 21	High byte of the low character of the voltage set
Byte 22	Low byte of the High character of the voltage set
Byte 23	High byte of the High character of the voltage set
Byte 24	Output state of the power supply
Byte 25	System reserved
Byte 26	Checksum

The output state of the power supply is revealed by the individual bits of byte 24:

From high to low

b7	b6	b5	b4	b3	b2	b1	b0

b0：0=output OFF;1=output ON

b1：0=current acceptable;1=excessive current

b2：0=power acceptable;1=excessive power

b3：0=local (front panel) control;1=remote (PC) control

3) 82h,Control the ON/OFF status of the power supply

Byte 1	Frame start (AAh)
Byte 2	Address (00h-FEh)
Byte 3	Command (82h)
Byte 4	The state of the power supply
Byte 5~25	System reserved
Byte 26	Checksum

The desired state of the power supply is specified by the individual bits of byte 4.

From high to low

b7	b6	b5	b4	b3	b2	b1	b0

b0： 0=output OFF;1=output ON

b1： 0=go to local mode (front panel control); 1=go to remote control (PC in control)

Notes：Only under the situation of PC controlling, you can set the parameters of the power .

4)83h, Set the power supply calibration protection state

Byte 1	Frame start (AAh)
Byte 2	Address (00h-FEh)
Byte 3	Command(83h)
Byte 4	calibration protection state
Byte 5	Calibration password (0X28H)
Byte 6	Calibration password (0X01H)
Byte 7 to Byte 25	System reserved
Byte 26	Checksum

Calibration protection state is specified by one byte. The definition of each bit unit is as the following:

From the high to the low

b7	b6	b5	b4	b3	b2	b1	b0

b0: 0 = protection enable;1= protection disable.

5) 84h, Read the power supply calibration protection state

Byte 1	Frame start (AAh)
Byte 2	Address (00h-FEh)
Byte 3	Command (84h)
Byte 4	Power supply calibration protection state
Byte 5 to Byte 25	System reserved
Byte 26	Checksum

Calibration protection state is specified by one byte. The definition of each bit unit is as the following:

From the high to the low

b7	b6	b5	b4	b3	b2	b1	b0

b0: 0=protection enable; 1= protection disable

6) 85h, Calibrate the voltage of the power supply

Byte 1	Frame start (AAh)
Byte 2	Address (00h-FEh)
Byte 3	Command (85h)
Byte 4	Voltage calibration point (1~4)

Byte 5 to Byte 25	System reserved
Byte 26	Checksum

7) 86h, Read the current actual output voltage of the power supply

Byte 1	Frame start (AAh)
Byte 2	Address (00h-FEh)
Byte 3	Command (86h)
Byte 4	Low byte of the low character of the actual voltage
Byte 5	High byte of the low character of the actual voltage
Byte 6	Low byte of the high character of the actual voltage
Byte 7	High byte of the high character of the actual voltage
Byte 8 to Byte 25	System reserved
Byte 26	Checksum

8) 87h, Calibrate the current of the power supply

Byte 1	Frame start (AAh)
Byte 2	Address (00h-FEh)
Byte 3	Command (87h)
Byte 4	Current calibration point (1~2)
Byte 5 to Byte 25	System reserved
Byte 26	Checksum

9) 88h, Read the actual output current of the power supply

Byte 1	Frame start (AAh)
Byte 2	Address (00h-FEh)
Byte 3	Command (88h)
Byte 4	Low byte of the actual current
Byte 5	High byte of the actual current
Byte 5 to Byte 25	System reserved
Byte 26	Checksum

10) 89h, Set the calibration information of the power supply

Byte 1	Frame start (AAh)
Byte 2	Address (00h-FEh)
Byte 3	Command (89h)

Byte 4 to byte 23	Calibration information (ASCII Code)
Byte 24	System reserved
Byte 25	System reserved
Byte 26	Checksum

11) 8Ah, Read the calibration information of the power supply

Byte 1	Frame start (AAh)
Byte 2	Address (00h-FEh)
Byte 3	Command (8Ah)
Byte 4 to Byte 23	Calibration Information (ASCII Code)
Byte 24	System reserved
Byte 25	System reserved
Byte 26	Checksum

12) 8Bh, Set the serial No of the power supply

Byte 1	Frame start (AAh)
Byte 2	Address (00h-FEh)
Byte 3	Command (8Bh)
Byte 4 to Byte 23	Serial No. (ASCII Code)
Byte 24	System reserved
Byte 25	System reserved
Byte 26	Checksum

13) 8Ch, Read the serial No, product type and software version No of the power supply

Byte 1	Frame start (AAh)
Byte 2	Address (00h-FEh)
Byte 3	Command (8Ch)
Byte 4 to Byte 9	Product serial No (ASCII Code)
Byte 10 to Byte 14	Product type (ASCII Code)
Byte 15	Low byte of the software version
Byte 16	High byte of the software version
Byte 16 to Byte 25	System reserved
Byte 26	Checksum

Press the button”1” on the keyboard when you are switching on the power supply, you will see the serial number, product type and firmware version of this unit will show on the LCD.

For example:

If the product serial No is 000045, the product type is 3645A and the software version No is V2.03, the teturn data is as the following:

AA	00	30	30	30	30	30	34	35	33	36	35	41	CB	00	XX	XX	XX	XX	XX	XX	XX	XX	XX	XX	57

14) 12h, Return check information command

Byte 1	Frame start (AAh)
Byte 2	Address (00h-FEh)
Byte 3	Command (12h)
Byte 4	80h indicates correct, 90h indicates wrong
Byte 5 to byte 25	System reserved
Byte 26	Checksum

When the power supply receives a frame of set command, it will check this frame of command and return the relative checked result.

When the power supply receives a frame of reading command, it will check this frame of command. If it checks correctly, it will return the relative read data. And if it checks wrongly, it will return the check command (90h).

Examples:

1. Set the parameters:

3000mA,36000mV,10800mW(108W) ,3000mV

AA 00 80 B8 0B A0 8C 00 00 30 2A B8 0B 00 00 00 00 00 00 00 00 00 00 00 00 36

2. Read the parameters:

AA 00 81 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 2B

3. Set control status:

A: PC control, output ON

AA 00 82 03 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 2F

B: PC control, output OFF

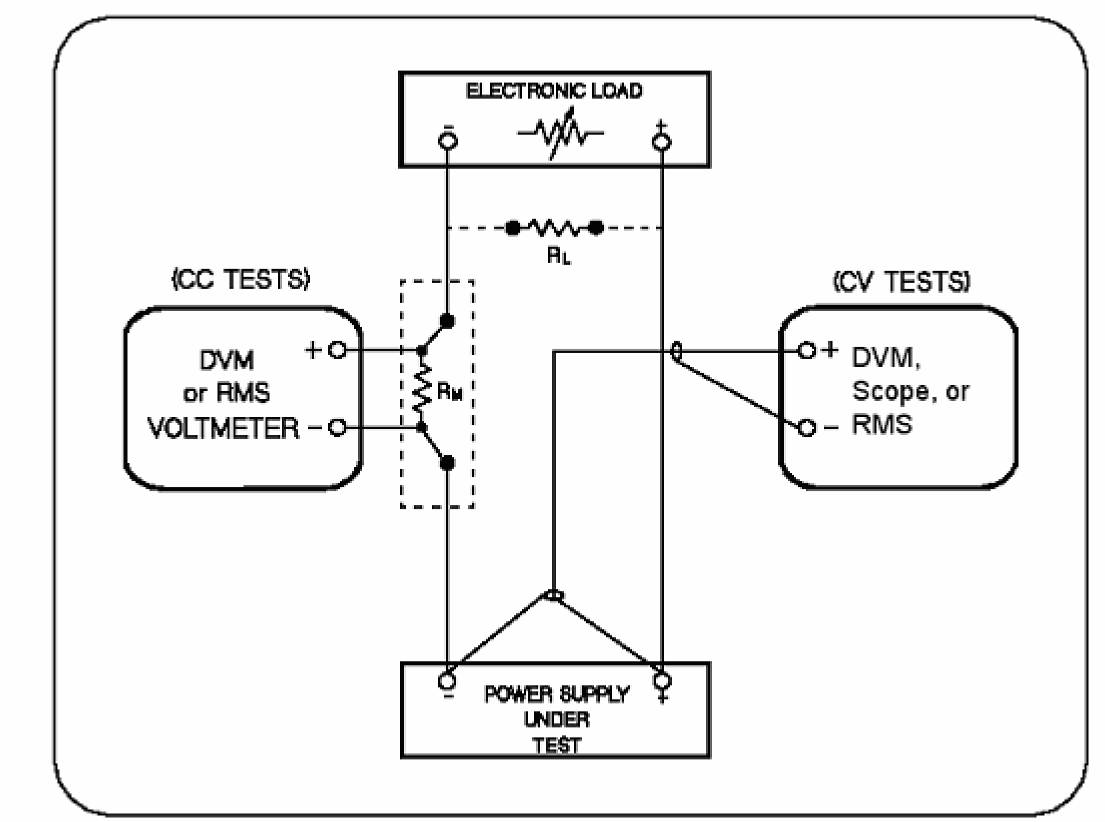
AA 00 82 02 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 2E

4.Self-controlled

AA 00 82 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 2C

E. Power supply Calibration

1. Structure of the system



2. Procedure of calibration

a. Disable the power calibration protection.

AA	00	83	01	28	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	57

b. Set load to CC mode and Load OFF

c. Calibrate the first point of the voltage

AA	00	85	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	30

d. Return the actual output voltage to the power supply unitl the output is stable

AA	00	86	XX	XX	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	XX

e. To calibrate the voltage of the second point

AA	00	85	02	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	31

f. To wait the outputs of the power to be stable and return to the power the current actual testing voltage value

AA	00	86	XX	XX	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	XX

g. To calibrate the voltage of the third point

AA	00	85	03	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	32

h. To wait the outputs of the power to be stable and return to the power the current actual testing voltage value

AA	00	86	XX	XX	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	XX

i. To calibrate the voltage of the fourth point

AA	00	85	04	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	33

j. To wait the outputs of the power to be stable and return to the power the current actual testing voltage value

AA	00	86	XX	XX	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	XX

k. To make the load be short circuit

l. To calibrate the current of the first point

AA	00	87	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	32

m. To wait the outputs of the power to be stable and return to the power the current actual testing voltage value

AA	00	88	XX	XX	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	XX

n. To calibrate the current of the second point

AA	00	87	02	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	33

o. To wait the outputs of the power to be stable and return to the power the current actual testing voltage value

AA	00	88	XX	XX	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	XX

p. To make the power calibration protection mode be ability

AA	00	83	00	28	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	56

q. To finish the power calibration

Example Program 1

Explanation:

The following program is edited and passed in Delphi5.0;

TComm32 control file can be downloaded from www.array.com;

The demonstration program can be downloaded from www.array.com;

The other programming tool is the similar;

```
unit Main;

interface

uses
  Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs,
  StdCtrls, Comm32, ExtCtrls;

//To definite the constant
const
  POWER_ADDRESS    = $00; //Power Address
  ORDER_WRITE      = $80; //To set order
  ORDER_READ       = $81; //To read the parameter
  ORDER_CONTROL    = $82; //Control order
  PC_CONTROL       = $02; //PC controlling
  SELF_CONTROL     = $00; //Power self-control
  POWER_ON         = $03; //Open the output
  POWER_OFF        = $02; //Close the output

type
  TForm1 = class(TForm)
    Comm232: TComm32;
    cbCOMM: TComboBox;
    cbBaud: TComboBox;
    Label1: TLabel;
    Label2: TLabel;
    btnOpen: TButton;
    Memo1: TMemo;
    cbOrder: TComboBox;
    btnSend: TButton;
    Bevel1: TBevel;
    procedure btnOpenClick(Sender: TObject);
    procedure FormCreate(Sender: TObject);
    procedure cbOrderClick(Sender: TObject);
    procedure btnSendClick(Sender: TObject);
    procedure Comm232RequestHangup(Sender: TObject);
    procedure Comm232ReceiveData(Buffer: Pointer; BufferLength: Word);
  private
    { Private declarations }
    SendBuf:array[0..25] of Byte;     //Sending data buffer definition
    ReceBuf:array[0..25] of Byte;     //Receiving data buffer fefinition
    Order     :Byte;                  //Order
    AddOrder:Byte;                    //Attached order
    procedure TotalBytes;             //Counting the checking sum
    procedure ShowSendBuf;            //Displaying the sending data
    procedure ShowReceBuf;            //Displaying the receiving data
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}
//Event of Opening the COM
procedure TForm1.btnOpenClick(Sender: TObject);
begin
  Comm232.StopComm;
  Comm232.CommPort:=cbComm.Text;
  Comm232.BaudRate:=StrToInt(cbBaud.Text);
  Comm232.ByteSize:=8;    //8-digit data bit
  Comm232.Parity:=0;      //Non-checking
  Comm232.StopBits:=0;    //Bit 1; the stopping bit
  try
    Comm232.StartComm;
    btnSend.Enabled:=True;
  except
    ShowMessage(Format('Falied to open %s',[cbComm.Text]));
    btnSend.Enabled:=False;
  end;
end;

//To initialize the parameter
procedure TForm1.FormCreate(Sender: TObject);
begin
  cbOrder.ItemIndex:=0;
  Order:=Order_Read;
end;

//To select the order
procedure TForm1.cbOrderClick(Sender: TObject);
begin
  case cbOrder.ItemIndex of
    0 :   Order:=Order_Read;       //Read Params

    1 :   Order:=Order_Write;      //Params Setting
    2 :   begin            //PC Control
            Order:=Order_Control;
```

```
                AddOrder:=Pc_Control;
             end;
        3 :   begin             //Control By Self
              Order:=Order_Control;
                AddOrder:=Self_Control;
              end;
        4 :   begin             //Power Off
               Order:=Order_Control;
                 AddOrder:=Power_Off;
              end;
        5 :   begin             //Power On
              Order:=Order_Control;
                AddOrder:=Power_On;
              end;
   end;
end;

//Event of sending data
procedure TForm1.btnSendClick(Sender: TObject);
var
   CurrentMax:Word; //The max current (proportion coefficient being 1000)
   VoltageMax:Word; // The max voltage (proportion coefficient being 1000)
   PowerMax    :Word; // The max power (proportion coefficient being 1000)
   CurVoltage:Word; // The current voltage value (proportion coefficient being 1000)
begin
   CurrentMax:=3000;    //3A
   VoltageMax:=36000; //36V
   PowerMax    :=10800; //108W
   CurVoltage:=10000; //10V
   FillChar(SendBuf,26,0);
   SendBuf[0]:=$AA;
   SendBuf[1]:=Power_Address;
   SendBuf[2]:=Order;
   if Order = Order_Write then //To set the parameter
   begin
      SendBuf[3]:=CurrentMax mod 256;
      SendBuf[4]:=CurrentMax div 256;
      SendBuf[5]:=VoltageMax mod 256;
      SendBuf[6]:=VoltageMax div 256;
      SendBuf[7]:=PowerMax mod 256;
      SendBuf[8]:=PowerMax div 256;
      SendBuf[9]:=CurVoltage mod 256;
      SendBuf[10]:=CurVoltage div 256;
      SendBuf[11]:=Power_Address;
   end;
   if Order = Order_Control then
      SendBuf[3]:=AddOrder;
   TotalBytes;
```

```
  ShowSendBuf;
  Comm232.WriteCommData(@SendBuf,26);
end;

//Event of hanging the COM port
procedure TForm1.Comm232RequestHangup(Sender: TObject);
begin
  Comm232.StopComm;
  Comm232.StartComm;
end;

//Event of receiving data
procedure TForm1.Comm232ReceiveData(Buffer: Pointer; BufferLength: Word);
var
  i:Byte;
  Byte25:Byte;
begin
  if BufferLength <> 26 then Exit;
  CopyMemory(@ReceBuf,Buffer,26);
  if ReceBuf[0] <> $AA then Exit;
  if not (ReceBuf[2] in [Order_Write..Order_Control]) then Exit;
  Byte25:=0;
  for i:=0 to 24 do
    Byte25:=Byte25+ReceBuf[i];
  if Byte25 <> ReceBuf[25] then Exit;
  ShowReceBuf;
end;

//To count the checking sum
procedure TForm1.TotalBytes;
var
  i:Byte;
begin
  SendBuf[25]:=0;
  for i:=0 to 24 do
    SendBuf[25]:=SendBuf[25]+SendBuf[i];
end;

//To display the sending data
procedure TForm1.ShowSendBuf;
var
  i:Byte;
  Str:String;
begin
  for i:=0 to 25 do
    Str:=Str+' '+IntToHex(SendBuf[i],2);
  Memo1.Lines.Add('Send :'+Str);
end;
```

```
//To display the receiving data
procedure TForm1.ShowReceBuf;
var
  i:Byte;
  Str:String;
begin
  for i:=0 to 25 do
    Str:=Str+' '+IntToHex(ReceBuf[i],2);
  Memo1.Lines.Add('Rece :'+Str);
end;

end.
```

Example program 2

Explanation:

1. The following program is edited and passed in VC6.0;

2. The demonstration program can be downloaded from www.array.com;

Procedure:

To definite the variable and the function:

```
public:
BYTE Cur_Order;            //Command word
BYTE Add_Order;           //Attached command word
int    Rece_Count;           //The total sum of the received characters
CByteArray SendBuf;       //To sending the buffer
CByteArray ReceBuf;       //To receiving the buffer
void InitData();                //To buffer storage the initial data
void CalDataTotal();         //To account the checking sum
void ShowSendData();      //To display the sending data
void ShowReceData();      //To display the receiving data
```

To definite the constant:

```
const BufferMax                   = 26;    //The max data buffer
const POWER_ADDRESS    = 0x00; //Power address
const ORDER_WRITE         = 0x80; //To set the order
const ORDER_READ           = 0x81; //To read the parameter
const ORDER_CONTROL    = 0x82; //The control order
const PC_CONTROL           = 0x02; //PC controlling
const SELF_CONTROL       = 0x00; //Powe self-controlling
const POWER_ON               = 0x03; //To open the output
const POWER_OFF             = 0x02; //To close the output
```

3. Function Part:

3．1//To account the checking sum

```
void CCommDlg::CalDataTotal()
```

```
{
    BYTE i;
    BYTE Value1;
    Value1=0;
  for (i=0;i<=BufferMax-2;i++)
    {
        Value1=Value1+SendBuf.GetAt(i);
    }
    SendBuf.SetAt(BufferMax-1,Value1);
}
```

3．2 //To initialize the dada buffer

```
void CCommDlg::InitData()
{
    Rece_Count=0;
  SendBuf.SetSize(26);
  ReceBuf.SetSize(26);
    SendBuf.RemoveAll();
  ReceBuf.RemoveAll();
    for (BYTE i=0;i<=BufferMax-1;i++)
  {
     SendBuf.Add(0);
     ReceBuf.Add(0);
  }
    SendBuf.SetAt(0,0xAA);
    SendBuf.SetAt(1,POWER_ADDRESS);
  SendBuf.SetAt(2,ORDER_READ);
    Cur_Order=ORDER_CONTROL;
    Add_Order=POWER_OFF;
  CalDataTotal();
  ShowSendData();
}
```

3．3//To display the sending data

```
void CCommDlg::ShowSendData()
{
   BYTE i;
   BYTE Value;
   CString Temp;
   m_SendData="";
   for (i=0;i<=BufferMax-1;i++)
   {
       Value=SendBuf.GetAt(i);
       Temp.Format("%2x",Value);
       if (Value < 16)
           Temp.SetAt(0,'0');
      m_SendData+=Temp;
       m_SendData+=" ";
```

```
        }
    m_SendData.MakeUpper();
    UpdateData(FALSE);
}
```

3．4//To display the receiving data

```
void CCommDlg::ShowReceData()
{
   BYTE i;
   BYTE Value;
   CString Temp;
   for (i=0;i<=BufferMax-1;i++)
   {
        Value=ReceBuf.GetAt(i);
        Temp.Format("%2x",Value);
        if (Value < 16)
             Temp.SetAt(0,'0');
       m_ReceiveData+=Temp;
        m_ReceiveData+=" ";

   }
   m_ReceiveData+="\r\n";
   m_ReceiveData.MakeUpper();
   UpdateData(FALSE);
}
```

3．5//To convert the characters into hexadecimal number

```
int Str2Hex(CString str,CByteArray &data)
{//To convert a character string as a hexadecimal string into a byte group. The bytes can be divided by spaces. The length of the converted byte group will be returned. Simultaniously the length of the byte group will be set automatically.
     int t,t1;
     int rlen=0,len=str.GetLength();
     data.SetSize(len/2);
     for(int i=0;i<len;)
     {
        char l,h=str[i];
        if(h==' ')
        {
            i++;
           continue;
        }
        i++;
        if(i>=len)break;
        l=str[i];
        t=HexChar(h);
        t1=HexChar(l);
        if((t==16)||(t1==16))
          break;
```

```
                else
                      t=t*16+t1;
                i++;
                data[rlen]=(char)t;
                rlen++;
          }
          data.SetSize(rlen);
          return rlen;
}
```

3．6//To test a character be a hexademical character or not. If it is, it will return the relative value. And if it is not, it will return 0x10;

```
char HexChar(char c)
{         if((c>='0')&&(c<='9'))
          return c-0x30;
          else if((c>='A')&&(c<='F'))
          return c-'A'+10;
          else if((c>='a')&&(c<='f'))
          return c-'a'+10;
          else return 0x10;
}
```

Event Processing Part

4． 1 Event of receiving data

```
void CCommDlg::OnComm()
{
   if(stop)return;
   VARIANT m_input1;
   COleSafeArray m_input2;
   long length,i;
   BYTE data[1024];
   CString str;
   if(m_Comm.GetCommEvent()==2)//Receiving the characters in buffer zone
   {
      m_input1=m_Comm.GetInput();//Readubg the data in the buffer zone
      m_input2=m_input1;//Convert the VARIANT variable into the ColeSafeArray variable
      length=m_input2.GetOneDimSize();//Defining the length of the data
      for(i=0;i<length;i++)
         m_input2.GetElement(&i,data+i);//Convert the data into BYTE array
      for(i=0;i<length;i++)//Convert the array into Cstring variable
{
         BYTE a=* (char *)(data+i);
         if(m_hex.GetCheck())
           {
            str.Format("%02X ",a);
                if ((a==0xAA) && (Rece_Count>=26))
                   Rece_Count=0;
                //Save the data to ReceBuf
          ReceBuf.SetAt(Rece_Count+i,a);
      }
     else
        str.Format("%c",a);
}
   Rece_Count=Rece_Count+length;
     UpdateData(FALSE);//Renew the contents of the editing frame
    //To process the receiving data
    if (Rece_Count == 26)
    {
        //1. To check the correct of the lock head
        if (ReceBuf.GetAt(0) != 0xAA)
            exit(0);
        //2. To check the correct of the address
      if (ReceBuf.GetAt(1) != POWER_ADDRESS)
            exit(0);
        //3. To check the command word
      if (ReceBuf.GetAt(2) < 0x80)
            exit(0);
        if (ReceBuf.GetAt(2) > 0x82)
            exit(0);
        //4. To check the checking sum
        BYTE Total,i;
        Total=0;
        for (i=0;i<=BufferMax-2;i++)
            Total=Total+ReceBuf.GetAt(i);
        if (Total != ReceBuf.GetAt(BufferMax-1))
            exit(0);
        //Correct part of data processing
        ShowReceData();
  ...
    }
  }
}
```

4．2 To initializing the dialogue frame

```
BOOL CCommDlg::OnInitDialog()
{
    …
    // TODO: Add extra initialization here
    //To initialize the control file and buffer storage the data
    m_com.SetCurSel(0);
    m_speed.SetCurSel(4);
   m_Order.SetCurSel(0);
   m_hexsend.SetCheck(1);
    m_hex.SetCheck(1);
    UpdateData(TRUE);
    InitData();
    return TRUE;    // return TRUE unless you set the focus to a control
}
```

4．3 To open the COM port

```
void CCommDlg::OnButton1()
{
    if( !m_Comm.GetPortOpen())
        m_Comm.SetPortOpen(TRUE);//To open the Com
    else
    {
          m_Comm.SetPortOpen(FALSE);
          m_Comm.SetPortOpen(TRUE);//To open the Com
    }
    UpdateData(TRUE);
}
```

4．4 To delete the receiving data

```
void CCommDlg::OnButton2()
{
    m_ReceiveData.Empty();//To delete the data in the receiving dialogue frame
    //m_SendData.Empty();//To delete the data in the sending dialogue frame
    UpdateData(FALSE);
}
```

4．5 To select the COM port

```
void CCommDlg::OnComselect()
{
    if(m_Comm.GetPortOpen())
        m_Comm.SetPortOpen(FALSE);
    m_Comm.SetCommPort(m_com.GetCurSel()+1);
}
```

4．6 To set the Baud Rate

```
void CCommDlg::OnComspeed()
{
          CString temp;
          int i=m_speed.GetCurSel();
          switch(i)
          {
          case 0:
                    i=2400;
                    break;
          case 1:
                    i=4800;
                    break;
          case 2:
                    i=9600;
                    break;
          case 3:
```

```
        i=19200;
        break;
    case 4:
        i=38400;
        break;
    }
    temp.Format("%d,n,8,1",i);
    m_Comm.SetSettings(temp);
}
```

4．7 The receiving data event

```
void CCommDlg::OnSend()
{
    // TODO: Add your control notification handler code here
    //To set the sending data
    SendBuf.SetAt(2,Cur_Order);
  SendBuf.SetAt(3,Add_Order);
    if (m_Order.GetCurSel()==1)
    {
        // Set the output current as 3A and the proportion coefficient as 1000
        SendBuf.SetAt(3,3000 % 256);
     SendBuf.SetAt(4,3000 / 256);
        // Set the output voltage as 36V and the proportion coefficient as 1000
     SendBuf.SetAt(5,36000 % 256);
     SendBuf.SetAt(6,36000 / 256);
        // Set the power as 108W and the proportion coefficient as 1000
        SendBuf.SetAt(7,10800 % 256);
     SendBuf.SetAt(8,10800 / 256);
        //Set the output voltage as 3V and the proportion coefficient as 1000
        SendBuf.SetAt(9,3000 % 256);
     SendBuf.SetAt(10,3000 / 256);
        //Set the address as: POWER_ADDRESS
        SendBuf.SetAt(11,POWER_ADDRESS);
    }
    if( m_Comm.GetPortOpen())
    {
      CalDataTotal();
      ShowSendData();
     if(m_hexsend.GetCheck())
      {
       int len=Str2Hex(m_SendData,SendBuf);
       m_Comm.SetOutput(COleVariant(SendBuf));//Sending data
      }
     else
        m_Comm.SetOutput(COleVariant(m_SendData));//Sending data
    }
    else
        MessageBox("Please open the COM connector first!", NULL, MB_OK);
}
```

4．8 Command Selection

```
void CCommDlg::OnSelendokOrder()
{    int i=m_Order.GetCurSel();
        switch(i)
        {
        case 0:
                Cur_Order=ORDER_READ;
                break;
        case 1:
                Cur_Order=ORDER_WRITE;
                break;
        case 2:
                Cur_Order=ORDER_CONTROL;
                Add_Order=PC_CONTROL;
                break;
        case 3:
                Cur_Order=ORDER_CONTROL;
                Add_Order=SELF_CONTROL;
                break;
        case 4:
                Cur_Order=ORDER_CONTROL;
                Add_Order=POWER_ON;
                break;
        case 5:
                Cur_Order=ORDER_CONTROL;
                Add_Order=POWER_OFF;
                break;
        }
}
```