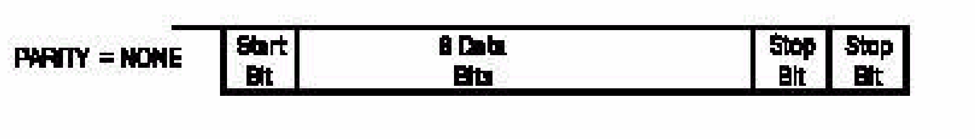
Communication Protocol for Power Supplies and PC

Power supplies may connect to RS-232 by the DB9 through the electrical couple transferred circuit. The following will help you to know how to control the outputs of the power supplies by the PC.

A. RS-232 Communication Setup

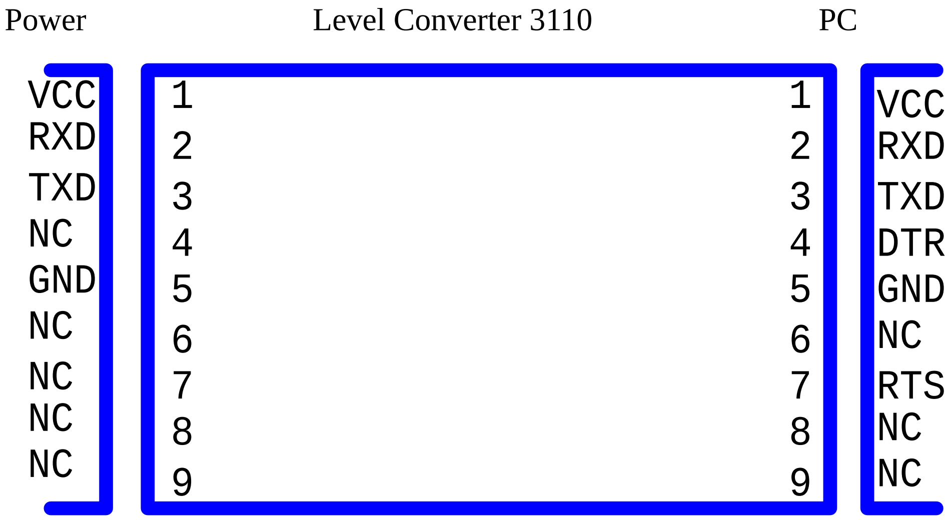
It may adjust the Communication Baud Rate and setup the address of the power supplies by the MENU key on the panel.

1□ Address:□0 --31□
2□ Baud Rate: 9600 (4800, 9600, 19200, 38400)
3□ Digital Bit: 8
4□ Stop Bit: 1
5□ Check Bit: None



B. Serial Interface DB9

The output of the DB9 interface on the back panel of the power supply is TTL electrical level. It must be electrical level transferred by the 3110 and then connected to the connector of the PC.



C. Frame Form

The length of the frame is 26 (compatible with FAB). And the form is as the below:

Lock Head	Power Address	Command Word	Byte 4 to Byte 25 are the contents of the related information	Check Code

Explanation:

1□ The lock head is AAH and occupies one byte.

2□ The power address range is from 0 to 31 and occupies one byte.

3□ The command word occupies one byte. The command contents is as the following:

a□ 80H------To set the max current, the max power and the voltage level of the power

b□ 81H------To read the current value, voltage value, power value and the power state, which includes the switch state of the power, the over current state, the over power state

c□ 82H------To control of the ON/OFF of the power

d□ 83H------To set the power calibration protection state

e□ 84H------To read the power calibration protection state

f□ 85H------Command to calibrate the voltage

g□ 86H------To return the power supply the current actual voltage

h□ 87H------Command to calibrate the current

i□ 88H------ To return the power supply the current actual current

j□ 89H------To set the calibration information of the power

k□ 8AH------To read the calibration information of the power

l□ 8BH------To set the serial No. of the power supplies

m□ 8CH-----To read the serial No., product type and software version No. of the power supplies

n□ 12H-------Check command

If you want to control the outputs of the power supplies by the PC, you must set the power supply to the PC controlling state first. And the command word is 82H. If it is needed to calibrate the output of the power, to set the power calibration information and the product serial No., it must to set the power calibration protection mode as OFF.

4□ From Byte 4 to Byte 25 is the information content.

5□ Byte 26 is the check code and is the adding sum of the front 25 bytes.

D. Using of the Command Byte

1. To setup the max current, the max power and the voltage level (80H)

Byte 1	Lock head (AAH)
Byte 2	Power Address (0-31)
Byte 3	Command Word (80H)
Byte 4	Low byte of the max current
Byte 5	High byte of the max current
Byte 6	Low byte of the front byte of the max voltage
Byte 7	High byte of the front byte of the max voltage
Byte 8	Low byte of the back byte of the max voltage

Byte 9	High byte of the back byte 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 front byte of the voltage set
Byte 13	High byte of the front byte of the voltage set
Byte 14	Low byte of the back byte of the voltage set
Byte 15	High byte of the Back byte of the voltage set
Byte 16	New address of the power
Byte 17 to Byte 25	Preserved by the system
Byte 26	Check code

Current, voltage and power are all expressed by two bytes. The low byte is in the front and the high byte is at the back. For example, the current value 3589H is expressed as the following:

89H	35H

The setup range of the current is: 0-3000mA.
The setup range of the voltage is: 0-3600mV.
The setup range of the power is: 0-108 (capable to be expanded 10 times).

2. To read the current, voltage, power and state of the power

Byte 1	Lock head (AAH)
Byte 2	Power Address (0-31)
Byte 3	Command Word (80H)
Byte 4	Low byte of the max current
Byte 5	High byte of the max current
Byte 6	Low byte of the front byte of the voltage
Byte 7	High byte of the front byte of the voltage
Byte 8	Low byte of the back byte of the voltage
Byte 9	High byte of the back byte of the voltage
Byte 10	Low byte of the power
Byte 11	High byte of the power
Byte 12	Low byte of the max current

Byte 13	High byte of the max current
Byte 14	Low byte of the front byte of the max voltage
Byte 15	High byte of the front byte of the max voltage
Byte 16	Low byte of the back byte of the max voltage
Byte 17	High byte of the back byte of the max voltage
Byte 18	Low byte of the max power
Byte 19	High byte of the max power
Byte 20	Low byte of the front byte of the voltage set
Byte 21	High byte of the front byte of the voltage set
Byte 22	Low byte of the back byte of the voltage set
Byte 23	High byte of the Back byte of the voltage set
Byte 24	State of the power supply
Byte 25	Preserved by the system
Byte 26	Check code

The current, voltage and power are all expressed by two bytes. The low byte is in the front and the high byte is at the back.

From the high to the low

7	6	5	4	3	2	1	0

The state of the power supply is expressed by one byte. The definition of each bit unit is as the following:

Bit 0: state of the power supply, 0 for OFF and 1 for ON.

Bit 1: over current state of the power supply, 0 for normal and 1 for abnormal.

Bit 2: over power state of the power supply, 0 for normal and 1 for abnormal.

Bit 3: operating state, 0 for keyboard and 1 for PC.

Notes: The frame form that power answering the PC is the same as the upper.

3□To control the ON/OFF of the Power (82H)

Byte 1	Lock head (AAH)
Byte 2	Power Address (0-31)
Byte 3	Command Word (80H)
Byte 4	State of the power supply
Byte 5 to Byte 25	Preserved by the system
Byte 26	Check code

The state of the power supply is expressed by one byte. The definition of each bit unit is as the following:

From the high to the low

7	6	5	4	3	2	1	0

Bit 0: State of the power supply, 0 for OFF and 1 for ON

Bit 1: Controlling of the PC over the power, 0 for the power self-controlling and 1 for the PC controlling over the power

Explanation: Only under the condition of PC controlling, the power parameter can be set.

4. To set the power calibration protection state (83H)

Byte 1	Lock head (AAH)
Byte 2	Power Address (0-31)
Byte 3	Command word□83H□
Byte 4	Power calibration protection state
Byte 5	Calibration password (0X28H)
Byte 6	Calibration password (0X01H)
Byte 7 to Byte 25	Preserved by the system
Byte 26	Check code

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

From the high to the low

7	6	5	4	3	2	1	0

Bit 0: protection state, 0 for protection making the ability and 1 for protection out of protection.

5. To read the power calibration protection state (84H)

Byte 1	Lock head (AAH)
Byte 2	Power Address (0-31)
Byte 3	Command Word (84H)
Byte 4	Power calibration protection state
Byte 5 to Byte 25	Preserved by the system
Byte 26	Check code

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

From the high to the low

7	6	5	4	3	2	1	0

Bit 0: protection state, 0 for protection making the ability and 1 for protection out of protection

6. To calibrate the voltage of the power (85H)

Byte 1	Lock head (AAH)
Byte 2	Power Address (0-31)
Byte 3	Command word (85H)
Byte 4	Voltage calibration point (1~4)
Byte 5 to Byte 25	Preserved by the system
Byte 26	Check code

7. To return the power supply the current actual output voltage (86H)

Byte 1	Lock head (AAH)
Byte 2	Power Address (0-31)
Byte 3	Command word (86H)
Byte 4	Low byte of the front byte of the actual voltage
Byte 5	High byte of the front byte of the actual voltage
Byte 6	Low byte of the back byte of the actual voltage
Byte 7	High byte of the back byte of the actual voltage
Byte 8 to Byte 25	Preserved by the system
Byte 26	Check code

8. To calibrate the current of the power (87H)

Byte 1	Lock head (AAH)
Byte 2	Power Address (0-31)
Byte 3	Command word (87H)
Byte 4	Current calibration point (1~2)
Byte 5 to Byte 25	Preserved by the system
Byte 26	Check code

9. To return the power supply the current actual output current (88H)

Byte 1	Lock head (AAH)
Byte 2	Power Address (0-31)
Byte 3	Command word (88H)
Byte 4	Low byte of the actual current
Byte 5	High byte of the actual current
Byte 5 to Byte 25	Preserved by the system
Byte 26	Check code

10. To set the calibration information of the power (89H)

Byte 1	Lock head (AAH)
Byte 2	Power Address (0-31)
Byte 3	Command word (89H)
Byte 4 to byte 23	Calibration information (ASCII Code)
Byte 24	Preserved by the system
Byte 25	Preserved by the system
Byte 26	Check code

11. To read the calibration information of the power (8AH)

Byte 1	Lock head (AAH)
Byte 2	Power Address (0-31)
Byte 3	Command Word (8AH)
Byte 4 to Byte 23	Calibration Information (ASCII Code)
Byte 24	Preserved by the system
Byte 25	Preserved by the system
Byte 26	Check code

12. To read the product serial No, product type and software version No of the power (8CH)

Byte 1	Lock head (AAH)
Byte 2	Power Address (0-31)
Byte 3	Command word (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	Preserved by the system
Byte 26	Check code

Press the 1 key while starting the machine, then the LCD will display the product serial No, product type and software version No of the power.

For example:

If the product serial No is 000045, the product type is 3645A and the software version No is V2.03, the data the power returning 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

13. Check command (12H)

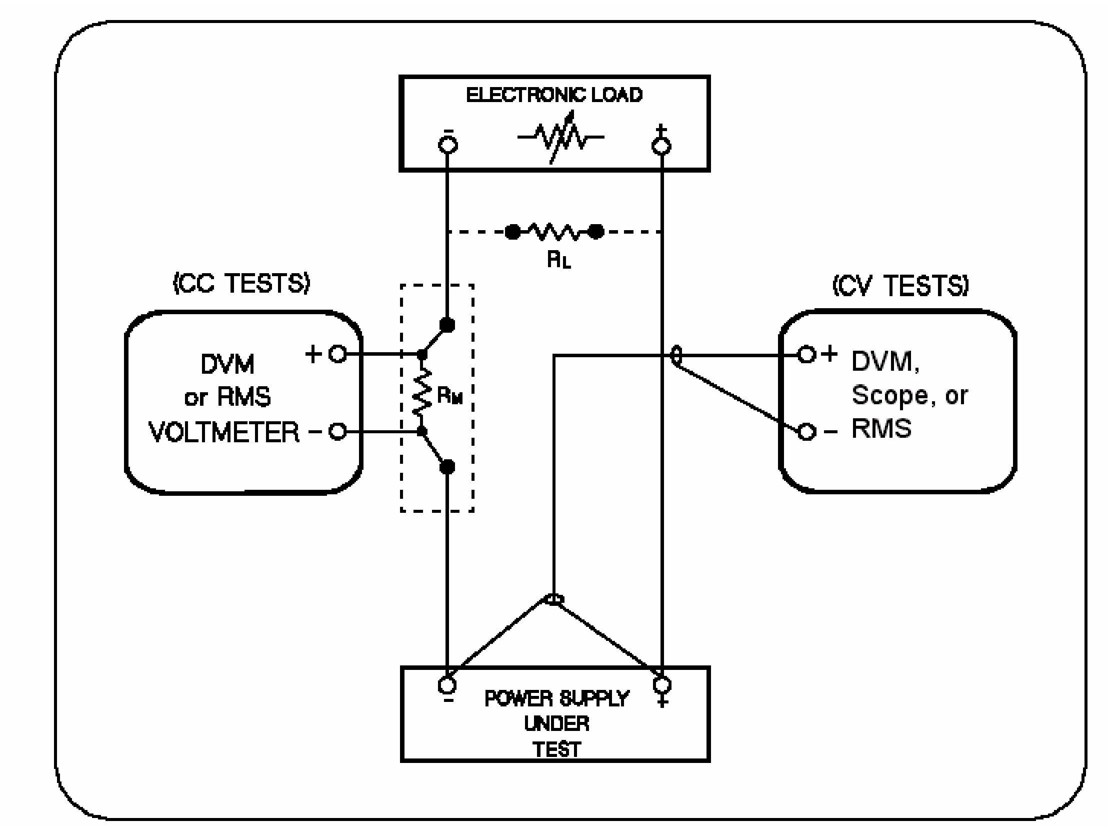
Byte 1	Lock head (AAH)
Byte 2	Power Address (0-31)
Byte 3	Command word (12H)
Byte 4	80H is correct. 90H is wrong.
Byte 5 to byte 25	Preserved by the system
Byte 26	Check code

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

When the power 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).

E. Power Calibration

1. Structure of the system



3. Procedure of calibration

a. To make the power calibration mode be out of ability

A	0	8	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5
A	0	3	1	8	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7

b. Load being constant current mode and the output being OFF

c. To calibrate the voltage of the first point

A	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
A	0	5	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

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

A	0	8	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X
A	0	6	X	X	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	X

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:

1□ The following program is edited and passed in Delphi5.0;
2□ TComm32 control file can be downloaded from www.array.com;
3□ The demonstration program can be downloaded from www.array.com;
4□ 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:

1□ **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
```

2. 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;
}
```

4. 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;
        }
    }
```