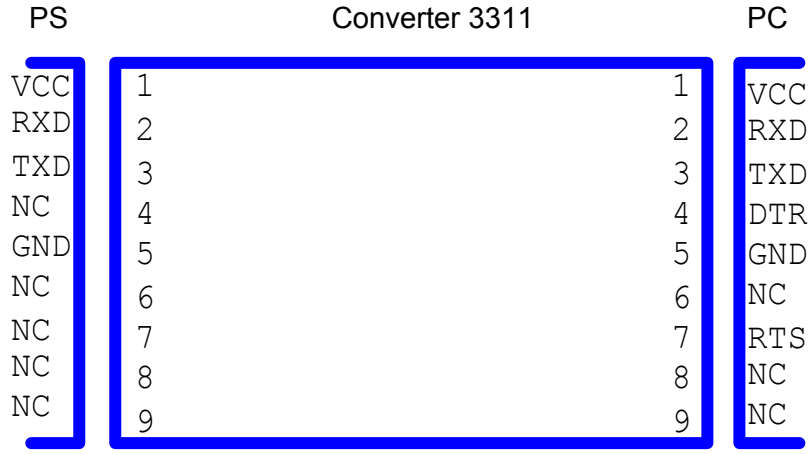
APPLICATION NOTE: *KONSTANTER LSP32K – Interface Protocol*

1 Interface Type

At the DB9 connector the LSP32K Power Supply device offers a serial data interface with TTL logic level for remote control and readout of set and measured parameters.

A Converter 3311 (RS232-to-TTL) is required to link a single Power Supply to the RS232 COM port of a PC.



A Converter 3312 (USB-to-TTL) is required to link a single Power Supply to the USB port of a PC.

Up to 32 devices can be linked to a PC using one Converter 3313 (RS232-to-RS485) at the RS232 COM port of a PC and one Converter 3314 (RS485-to-TTL) at each Power Supply. For identification each Power Supply in the system must be set to a different device address (0 – 31).

2 RS232 Communication Settings

- **Baud Rate**: 4800, 9600, 19200, 38400 bps
- **Data Bits**: 8
- **Stop Bit**: 1
- **Parity**: None

3 Data Frame

The length of the data frame is 26 bytes (compatible with FAB). The form is as follows.

1	2	3	4-25	26
AAh	**Address**	**Command Byte**	**Related Information Contents**	**Checking Code**

Directives:

1) **AAh** occupies one byte.
2) **Address** ranges from 0 to FEh and occupies one byte.
3) **Command Byte** ranges from 80h to 90h and occupies one byte.

   The content of the Command Byte is as follows:

   a) **80h** Setting the Max Current, the Max Power, the Voltage Level of the power supply.

   b) **81h** Reading the current value, the voltage value, the power value and the state of the power supply. The state of the power supply contains the ON/OFF state, the Over-current state and the Over-power state.

   c) **82h** Supervising the ON/OFF of the power supply.

d) **83h** Programming.

4) **Byte 4 to Byte 25** are the information contents.

5) **Byte 26** is the checking code and is the accumulating of the 25 previous bytes.

Use of the Command Bytes

6) Setup of the Max Current, the Max Power and the Voltage Level of the Power supply (80h)

Byte 1	**AAh**
Byte 2	**Address (0-FEh)**
Byte 3	**Command Byte (80h)**
Byte 4	**Low Byte of the Max Current**
Byte 5	**High Byte of the Max Current**
Byte 6	**Low Byte of the Max Voltage**
Byte 7	**High Byte of the Max Voltage**
Byte 8	**Low Byte of the Max Power**
Byte 9	**High Byte of the Max Power**
Byte 10	**Low Byte of the Voltage Setup**
Byte 11	**High Byte of the Voltage Setup**
Byte 12	**New Address of the Power Supply**
Byte 13 to 25	**Preserved by the System**
Byte 26	**Checking Code**

Current, Voltage and Power are all expressed by two bytes, with low byte in the front and the high byte behind.

Example: The current value 3589h is expressed as

89h	**35h**

7) Read the current, the voltage, the power and the state of the power supply

Byte 1	**AAh**
Byte 2	**Address (0-FEh)**
Byte 3	**Command Byte (81h)**
Byte 4	**Low Byte of the Current**
Byte 5	**High Byte of the Current**
Byte 6	**Low Byte of the Voltage**
Byte 7	**High Byte of the Voltage**
Byte 8	**Low Byte of the Power**
Byte 9	**High Byte of the Power**
Byte 10	**Low Byte of the Max Current**
Byte 11	**High Byte of the Max Current**
Byte 12	**Low Byte of the Max Voltage**
Byte 13	**High Byte of the Max Voltage**
Byte 14	**Low Byte of the Max Power**
Byte 15	**High Byte of the Max Power**
Byte 16	**Low Byte of the Voltage Setup**
Byte 17	**High Byte of the Voltage Setup**
Byte 18	**The State of the Power Supply**
Byte 19 to Byte 25	**Preserved by the System**
Byte26	**Checking Code**

Current, Voltage and Power are all expressed by two bytes, with low byte in the front and the high byte behind.

The State of the Power Supply is expressed by one byte. Each unit is defined as:

7	6	5	4	3	2	1	0

Position 0: State of the power supply output. 0 is OFF and 1 is ON.

Position 1: Over-current State of the power supply. 0 is normal and 1 is abnormal.

Position 2: Over-power state of the power supply. 0 is normal and 1 is abnormal.

Position 3: Operating State. 0 is for keyboard and 1 is for PC.

Note: The frame of power supply answering the PC is the same as the above.

8) Control the Output ON/OFF of the power supply

Byte 1	**AAh**
Byte 2	**Address**
Byte 3	**Command Byte**
Byte 4	**State of the Power Supply**
Byte 5 to Byte 25	**Preserved by the System**
Byte 26	**Checking Code**

The state of the power supply is expressed by one byte. Each unit is defined as:

7	6	5	4	3	2	1	0

Position 0: State of the power supply. 0 is OFF and 1 is ON.

Position 1: PC control over the power supply. 0 is the power supply self-control and 1 is the PC control over the power supply.

9) Power supply automatically transmits the max current, the max power and the voltage levels to the PC.

The frame is the same as frame of the setup of the max current, the max power and the voltage levels.

Samples 1

Note: The following procedure could be translated and edited in Delphi5.0.

```
unit Main;

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

// definition of constant
const
  POWER_ADDRESS  = $00; //The address of power supply
  ORDER_WRITE    = $80; //setting instruction
  ORDER_READ     = $81; //read data
  ORDER_CONTROL  = $82; //control instruction
  PC_CONTROL     = $02; //PC control
  SELF_CONTROL   = $00; //self control of power supply
  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;   //send the data delay saving instruction
    ReceBuf:array[0..25] of Byte;   //receive the data delay saving instruction
    Order   :Byte;              //instruction
    AddOrder:Byte;              //accessory instruction
    procedure TotalBytes;         //checking the total bytes
    procedure ShowSendBuf;      //show the data send out
    procedure ShowReceBuf;      //show the data receive in
  public
    { Public declarations }
  end;

var
  Form1: TForm1;

implementation

{$R *.DFM}
//open COM
procedure TForm1.btnOpenClick(Sender: TObject);
begin
  Comm232.StopComm;
  Comm232.CommPort:=cbComm.Text;
  Comm232.BaudRate:=StrToInt(cbBaud.Text);
  Comm232.ByteSize:=8;  //8 byte size
  Comm232.Parity:=0;    // Parity is none
```

```
  Comm232.StopBits:=0;  //1 byte stop
  try
    Comm232.StartComm;
    btnSend.Enabled:=True;
  except
    ShowMessage(Format('Falied to open %s',[cbComm.Text]));
    btnSend.Enabled:=False;
  end;
end;

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

//chose instruction
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;

//send the data
procedure TForm1.btnSendClick(Sender: TObject);
var
  CurrentMax:Word; //the max of current (proportion coefficien 1000)
  VoltageMax:Word; //the max of voltage (proportion coefficien 1000)
  PowerMax  :Word; //the max of power(proportion coefficien 100)
  CurVoltage:Word; // current voltage value (proportion coefficien 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 //setting 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;

//COM hang
procedure TForm1.Comm232RequestHangup(Sender: TObject);
begin
  Comm232.StopComm;
  Comm232.StartComm;
end;

//receive the byte
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;

//caculate the total byte
procedure TForm1.TotalBytes;
var
  i:Byte;
begin
  SendBuf[25]:=0;
  for i:=0 to 24 do
    SendBuf[25]:=SendBuf[25]+SendBuf[i];
end;

//show the date send out
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;

//show the data receive in
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.
```

Samples 2

Note: The following program could be edited and translated in VC6.0.

1. The definition of Variable and Function

```
public:
BYTE Cur_Order;        //commend character
BYTE Add_Order;       //accessory character
int  Rece_Count;       //the byte size of receive in
CByteArray SendBuf;    //send the instruction of delay saving
CByteArray ReceBuf;    //receive the instruction of delay saving
void InitData();       //initial dealy saving
void CalDataTotal();   //check and caculate the total data
void ShowSendData();   //show the data send out
void ShowReceData();   //show the data receive in
```

2. Definition of constant

```
const BufferMax      = 26;  //the max value of data delay saving
const POWER_ADDRESS  = 0x00; // The address of power supply
const ORDER_WRITE    = 0x80; //setting instruction.
const ORDER_READ     = 0x81; //read parameter
const ORDER_CONTROL  = 0x82; //control instruction.
const PC_CONTROL     = 0x02; //PC control
const SELF_CONTROL   = 0x00; //self-control of power supply
const POWER_ON       = 0x03; //open the output
const POWER_OFF      = 0x02; //close the output
```

3. Function -part

3.1 Calculate the total data

```
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 Initialization of dealy saving data

```
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 Show the data be send out

```
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 Show the data be received in

```
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 Convert to hexadecimal data

```
int Str2Hex(CString str,CByteArray &data)
    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 convert the data into initialization data.

```
int Str2Hex(CString str,CByteArray &data)

{   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. Dealing

4.1 Receiving the 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)//receive the data of buffer.
  {
    m_input1=m_Comm.GetInput();//read the data of buffer
    m_input2=m_input1;//convert the variable of VARIANT to variable ColeSafeArray
    length=m_input2.GetOneDimSize();//confirm the length of data
    for(i=0;i<length;i++)
      m_input2.GetElement(&i,data+i);//convert the data to BYTE type .
    for(i=0;i<length;i++)//convert the data to variable of Cstring.
{
      BYTE a=* (char *)(data+i);
      if(m_hex.GetCheck())
      {
        str.Format("%02X ",a);
       if ((a==0xAA) && (Rece_Count>=26))
         Rece_Count=0;
       //Saving data to ReceBuf
       ReceBuf.SetAt(Rece_Count+i,a);
      }
      else
       str.Format("%c",a);
}
    Rece_Count=Rece_Count+length;
     UpdateData(FALSE);//update of content of edition.
    //dealing the received data
    if (Rece_Count == 26)
    {
        //1. check the AAh
        if (ReceBuf.GetAt(0) != 0xAA)
            exit(0);
        //2.Check the address
         if (ReceBuf.GetAt(1) != POWER_ADDRESS)
            exit(0);
        //3.Check the instruction
         if (ReceBuf.GetAt(2) < 0x80)
            exit(0);
        if (ReceBuf.GetAt(2) > 0x82)
            exit(0);
        //4.check the code
        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 data dealing part
        ShowReceData();
    ...
     }
   }
}
```

4.2 Initialization dialogue frame

```
BOOL CCommDlg::OnInitDialog()
{
    …
    // TODO: Add extra initialization here
    //initialization control part and amortize.
    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 Open COM

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

4.4 Clear the receiving data

```
void CCommDlg::OnButton2()
{
  m_ReceiveData.Empty();// delete the data of receiving dialogue frame
  //m_SendData.Empty();// delete the data of sendin-g dialogue frame
  UpdateData(FALSE);
}
```

4.5 Chose COM

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

4.6 Setting 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 Receiving data

```
void CCommDlg::OnSend()
{
    // TODO: Add your control notification handler code here
    //setting data which will be send out
    SendBuf.SetAt(2,Cur_Order);
   SendBuf.SetAt(3,Add_Order);
    if (m_Order.GetCurSel()==1)
    {
        //setting current is 3A□ coeffcient of proportionality□1000
        SendBuf.SetAt(3,3000 % 256);
        SendBuf.SetAt(4,3000 / 256);
        //Setting voltage of 36V□coeffcient of proportionality□1000
        SendBuf.SetAt(5,36000 % 256);
        SendBuf.SetAt(6,36000 / 256);
        //Setting power is 08W□coeffcient of proportionality□100
        SendBuf.SetAt(7,10800 % 256);
        SendBuf.SetAt(8,10800 / 256);
        //Setting the output voltage of 3V□coeffcient of proportionality□1000
        SendBuf.SetAt(9,3000 % 256);
        SendBuf.SetAt(10,3000 / 256);
        //Setting the address□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));//send the data
       }
       else
         m_Comm.SetOutput(COleVariant(m_SendData));//send the data
    }
    else
        MessageBox("□□□□COM□!",NULL,MB_OK);
}
```

4.8 Commend choice

```
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;
    }
}
```