C#开发动态数据库函数描述

适用板卡型号：USB DAQ-580i 企业版

版本：ver190320-1

概述：

该板卡基于 USB 总线进行数据传输，该文件主要介绍 visual studio c#语言如何调用 USB 通信函数对采集器进行读写操作，用户也可直接阅读我司提供的范例源代码，可直接运行对采集器进行控制读取通道电压，相关程序用到以下三类动态链接库文件 dll

1：USB 通信数据传输函数
FTD2XX_NET.dll
主要功能：打开 USB 端口，发送字节，接收字节，关闭 USB 端口

2：数据解析函数（单通道，双通道，四通道，八通道）
DAQ_580i_VOLTDISPLAY_V5.dll
主要功能：将接收到的字节按照公式自动转换成电压值显示出来

3：采集卡指令控制函数
DAQ_580i_CMD_V5.dll
主要功能：控制采集卡工作模式，量程，采样速度等

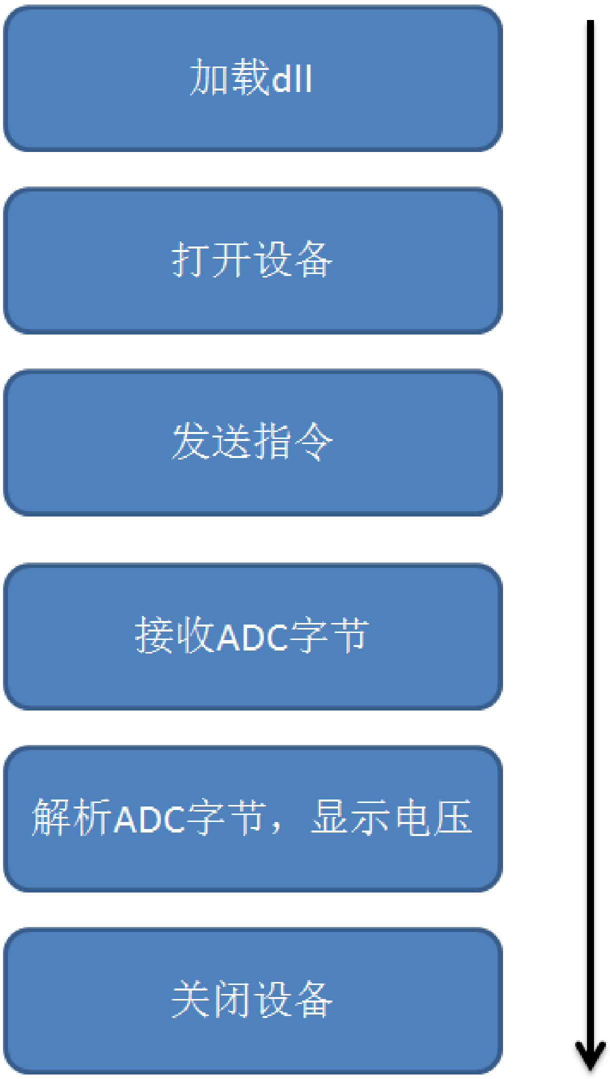
采集器的读写操作流程

一：单次测量模式

该模式即为查询模式，用户需要读一次电压的时候按以下流程操作一次即可获取电压，执行完成后自动停止

和串口操作极其相似，建议用户无需反复打开关闭 USB 设备，打开一次后可进行读写控制，直到测试任务完成，可执行关闭 USB 设备

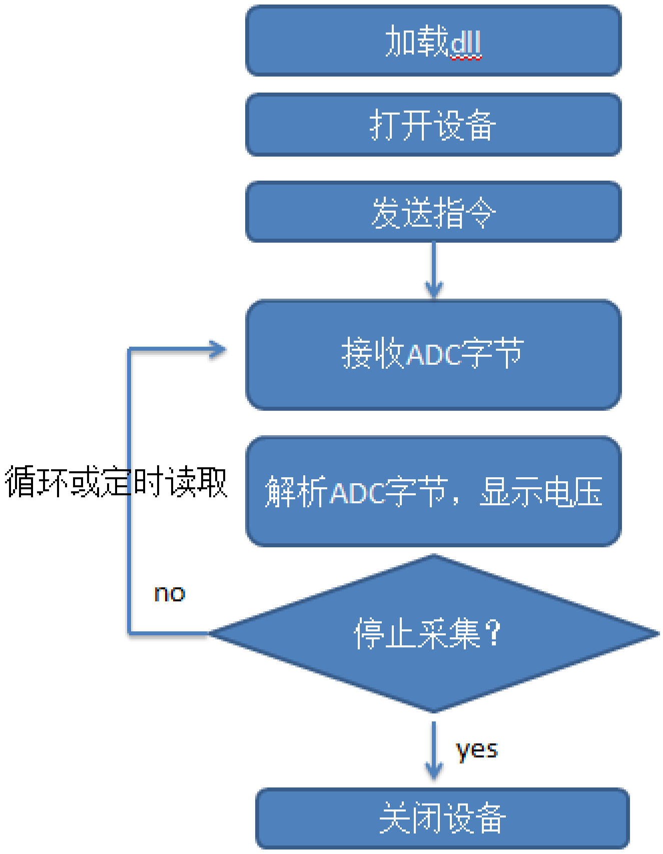
用户发送单次测量指令后，采集器根据接收到的指令决定返回字节数量的多少，然后处于待机状态，等待下一条指令



二：连续测量模式

用户需高速连续读取电压信号，或需要记录一个过程，可选用该模式，即连续采集模式

采集器接收的连续模式指令后，用户无需反复发送指令，采集器会根据设置的参数连续输出数据，用户需发送中断/停止采集指令才能中断采集器输出，操作流程如下



动态链接库介绍

USB 通信函数介绍 FTD2XX_NET.dll：

FTD2XX_NET.dll 集成以下函数；

1：//***** 采集器设备 USB 通信端口的打开时的相关操作************//

public FTD2XX_NET.FTDI.FT_STATUS **OpenByDescription**(string *description*)

描述：通过采集卡名称打开设备
参数：string *description*

例：**OpenByDescription**(USB DAQ-580i)；打开节点名称 USB DAQ-580i

返回：无

public FTD2XX_NET.FTDI.FT_STATUS **SetTimeouts**(uint *ReadTimeout*, uint *WriteTimeout*)

描述：设置 USB 通信发送和读超时时间
参数：uint *ReadTimeout*   uint *WriteTimeout*
例：SetTimeouts(10000, 10000); 接收超时时间10秒，发送超时时间10秒
返回：无

public bool **IsOpen** { get; }
描述：检查 USB 是否打开设备
参数：
返回：bool 量，True 为打开，False 为未打开

关于 USB 设备端口打开的操作实例代码（详见范例）：

```
    string name = "USB DAQ-580i";
    // tbx_Status.Text = "DAQ OPEN";
    myFtdiDevice.OpenByDescription(name);
    myFtdiDevice.SetTimeouts(10000, 10000);
    tbx_Status.Text=myFtdiDevice.IsOpen.ToString();
    if (tbx_Status.Text == "True") {    tbx_Status.Text="SUCCESS"; }
    else { tbx_Status.Text = "FAILED"; MessageBox.Show("Device open failed, disconnect device
from PC then reconnect"); }
```

2：//*********** USB 发送指令至采集卡***********//

Public **FTD2XX_NET.FTDI.FT_STATUS Write**(**byte[]** *dataBuffer*, **uint** *numBytesToWrite*, ref **uint** *numBytesWritten*)

描述：发送数据数组

参数： *dataBuffer*，需要发送的字节数组

numBytesToWrite，写入字节长度

numBytesWritten 默认为 0

返回：无

3：//*********** USB 接收采集卡发出的数据 ***********//

public **FTD2XX_NET.FTDI.FT_STATUS Read**(**byte[]** *dataBuffer*, **uint** *numBytesToRead*, ref **uint** *numBytesRead*)

描述：接收数据

参数： *dataBuffer* 需要接收的数据存放的数组

numBytesToWrite，接收字节的长度

ref **uint** *numBytesRead* 默认为 0

返回：*dataBuffer 数组*

4：//*********** 关闭 USB 设备 ***********//

public **FTD2XX_NET.FTDI.FT_STATUS Close**()

描述：关闭 USB 采集卡设备

参数：无

返回：无

动态链接库介绍

采集卡控制指令函数 DAQ_580i_CMD_V5.dll
DAQ_580i_CMD_V5.dll 集成以下函数；

1：差分单次单通道测量模式

输入参数：

string	*Samples*	**样本数量（启动 ADC 内部滤波器）**
string	*Range,*	**量程**
string	*Channel*	**通道**

输出参数

byte[]	*CMD*	*输出最终的控制指令数组*
uint	*len*	*单次接收 USB 缓冲区字节长度*
uint	*gain*	*量程参数，代入公式计算最终输出电压*

public static **void SINGLE_DIFF_ONECHS**(**string** *SampleRated*, **string** *Range*, **string** *Channel*, out **byte[]** *CMD*, out **uint** *len*, out **uint** *gain*)

```
        {
            byte bit1;
            byte bit2;
            byte bit3;
            byte bit4;
            byte bit5 = 0;

            double dt;

            switch (Channel)    //转换通道指令
            {

                default:          bit2 = 1; bit1 = 1; break;
                case "A1-A2": bit2 = 1; bit1 = 1; break;
```

```
    case "A3-A4": bit2 = 1; bit1 = 2; break;
    case "A5-A6": bit2 = 1; bit1 = 3; break;
    case "A7-A8": bit2 = 1; bit1 = 4; break;

}


switch (SampleRated)//采样速度指令
{
    default: bit3 = 240; dt = 1 / 30000d; len = 3; break;
    case "1": bit3 = 240; dt = 1 / 30000d; len = 3; break;
    case "2": bit3 = 224; dt = 1 / 15000d; len = 3; break;
    case "4": bit3 = 208; dt = 1 / 7500d; len = 3; break;
    case "8": bit3 = 192; dt = 1 / 3750d; len = 3; break;
    case "15": bit3 = 176; dt = 1 / 2000d; len = 3; break;
    case "30": bit3 = 161; dt = 1 / 1000d; len = 3; break;
    case "60": bit3 = 146; dt = 1 / 500d; len = 3; break;
    case "300": bit3 = 130; dt = 1 / 100d; len = 3; break;
    case "500": bit3 = 114; dt = 1 / 60d; len = 3; break;
    case "600": bit3 = 99; dt = 1 / 50d; len = 3; break;
    case "1000": bit3 = 83; dt = 1 / 30d; len = 3; break;
    case "1200": bit3 = 67; dt = 1 / 25d; len = 3; break;
    case "2000": bit3 = 51; dt = 1 / 15d; len = 3; break;
    case "3000": bit3 = 35; dt = 1 / 10d; len = 3; break;
    case "6000": bit3 = 19; dt = 1 / 5d; len = 3; break;
    case "12000": bit3 = 3; dt = 1 / 2.5d; len = 3; break;


}

switch (Range)    //量程指令
{
    default: bit4 = 1; gain = 2; break;
    case "±2.5V": bit4 = 1; gain = 2; break;
    case "±1.25V": bit4 = 2; gain = 4; break;
    case "±0.625V": bit4 = 3; gain = 8; break;
    case "±312.5mV": bit4 = 4; gain = 16; break;
    case "±156.25mV": bit4 = 5; gain = 32; break;
    case "±78.125mV": bit4 = 6; gain = 64; break;

}

byte[] buffer = { 170, bit1, bit2, bit3, bit4, bit5, 187 };
CMD = buffer;
```

```
        }
```

2：差分单次四通道测量模式

输入参数：

string *Samples* **样本数量（启动 ADC 内部滤波器）**

string *Range*, **量程**

string *Channel* **通道**

输出参数

byte[] *CMD* *输出最终的控制指令数组*

uint *len* *单次接收 USB 缓冲区字节长度*

uint *gain* *量程参数，代入公式计算最终输出电压*

double *dt* 时间间隔 采样速度的倒数

```
public static void SINGLE_DIFF_FOURCHS(string SampleRated, string Range, string
Channel, out byte[] CMD, out uint len, out uint gain, out double dt)
{
            byte bit1;
            byte bit2;
            byte bit3;
            byte bit4;
            byte bit5 = 0;


            switch (Channel)    //转换通道指令
            {

                default:                    bit2 = 1; bit1 = 21; break;
```

```
        case "A1-A2 A3-A4 A5-A6 A7-A8": bit2 = 1; bit1 = 21; break;

    }


    switch (SampleRated)//采样速度指令
    {
        default: bit3 = 240; dt = 1 / 30000d; len = 12; break;
        case "1": bit3 = 240; dt = 1 / 30000d; len = 12; break;
        case "2": bit3 = 224; dt = 1 / 15000d; len = 12; break;
        case "4": bit3 = 208; dt = 1 / 7500d; len = 12; break;
        case "8": bit3 = 192; dt = 1 / 3750d; len = 12; break;
        case "15": bit3 = 176; dt = 1 / 2000d; len = 12; break;
        case "30": bit3 = 161; dt = 1 / 1000d; len = 12; break;
        case "60": bit3 = 146; dt = 1 / 500d; len = 12; break;
        case "300": bit3 = 130; dt = 1 / 100d; len = 12; break;
        case "500": bit3 = 114; dt = 1 / 60d; len = 12; break;
        case "600": bit3 = 99; dt = 1 / 50d; len = 12; break;
        case "1000": bit3 = 83; dt = 1 / 30d; len = 12; break;
        case "1200": bit3 = 67; dt = 1 / 25d; len = 12; break;
        case "2000": bit3 = 51; dt = 1 / 15d; len = 12; break;
        case "3000": bit3 = 35; dt = 1 / 10d; len = 12; break;
        case "6000": bit3 = 19; dt = 1 / 5d; len = 12; break;
        case "12000": bit3 = 3; dt = 1 / 2.5d; len = 12; break;
    }

    switch (Range)    //量程指令
    {
        default: bit4 = 1; gain = 2; break;
        case "±2.5V": bit4 = 1; gain = 2; break;
        case "±1.25V": bit4 = 2; gain = 4; break;
        case "±0.625V": bit4 = 3; gain = 8; break;
        case "±312.5mV": bit4 = 4; gain = 16; break;
        case "±156.25mV": bit4 = 5; gain = 32; break;
        case "±78.125mV": bit4 = 6; gain = 64; break;

    }

    byte[] buffer = { 170, bit1, bit2, bit3, bit4, bit5, 187 };
    CMD = buffer;


}
```

3：差分连续单通道测量模式

输入参数：

string *Samples* **样本数量（启动 ADC 内部滤波器）**

string *Range*, **量程**

string *Channel* **通道**

输出参数

byte[] *CMD* *输出最终的控制指令数组*

uint *len* *单次接收 USB 缓冲区字节长度*

uint *gain* *量程参数，代入公式计算最终输出电压*

double *dt* 时间间隔 采样速度的倒数

```
public static void CONTINUE_DIFF_ONECHS(string SampleRated, string Range, string
Channel, out byte[] CMD, out uint len, out uint gain, out double dt)
{
                        byte bit1;
            byte bit2;
            byte bit3;
            byte bit4;
            byte bit5=0;

            switch (Channel)    //转换通道指令
            {

                default:            bit2 = 2; bit1 = 9; break;
                case "A1-A2": bit2 = 2; bit1 = 9; break;
                case "A3-A4": bit2 = 2; bit1 = 13; break;
                case "A5-A6": bit2 = 2; bit1 = 11; break;
                case "A7-A8": bit2 = 2; bit1 = 12; break;

            }
            switch (SampleRated)//采样速度指令
            {
                default: bit3 = 240; dt = 1 / 30000d; len = 6000 * 3; break;
                case "30000": bit3 = 240; dt = 1 / 30000d; len = 6000 * 3; break;
                case "15000": bit3 = 224; dt = 1 / 15000d; len = 3000 * 3; break;
                case "7500": bit3 = 208; dt = 1 / 7500d; len = 1500 * 3; break;
                case "3750": bit3 = 192; dt = 1 / 3750d; len = 700 * 3; break;
                case "2000": bit3 = 176; dt = 1 / 2000d; len = 400 * 3; break;
                case "1000": bit3 = 161; dt = 1 / 1000d; len = 300 * 3; break;
                case "500": bit3 = 146; dt = 1 / 500d; len = 100 * 3; break;
                case "100": bit3 = 130; dt = 1 / 100d; len = 20 * 3; break;
                case "60": bit3 = 114; dt = 1 / 60d; len = 12 * 3; break;
                case "50": bit3 = 99; dt = 1 / 50d; len = 10 * 3; break;
                case "30": bit3 = 83; dt = 1 / 30d; len = 6 * 3; break;
                case "25": bit3 = 67; dt = 1 / 25d; len = 5 * 3; break;
                case "15": bit3 = 51; dt = 1 / 15d; len = 3 * 3; break;
                case "10": bit3 = 35; dt = 1 / 10d; len = 2 * 3; break;
                case "5": bit3 = 19; dt = 1 / 5d; len = 1 * 3; break;
                case "2.5": bit3 = 3; dt = 1 / 2.5d; len = 1 * 3; break;
            }

            switch (Range)    //量程指令
            {
                default:                  bit4 = 1; gain = 2; break;
                case "±2.5V":          bit4 = 1; gain = 2; break;
                case "±1.25V":         bit4 = 2; gain = 4; break;
                case "±0.625V":        bit4 = 3; gain = 8; break;
                case "±312.5mV":    bit4 = 4; gain = 16; break;
                case "±156.25mV": bit4 = 5; gain = 32; break;
                case "±78.125mV": bit4 = 6; gain = 64; break;

            }

            byte[] buffer = { 170, bit1, bit2, bit3, bit4, bit5, 187 };
            CMD = buffer;


        }
```

4：差分连续双通道测量模式

输入参数：

string *Samples* **样本数量（启动 ADC 内部滤波器）**

string *Range*, **量程**

string *Channel* **通道**

输出参数

byte[] *CMD* *输出最终的控制指令数组*

uint *len* *单次接收 USB 缓冲区字节长度*

uint *gain* *量程参数，代入公式计算最终输出电压*

double *dt* 时间间隔 采样速度的倒数

public static **void CONTINUE_DIFF_TWOCHS**(string *SampleRated*, string *Range*, string *Channel*, out byte[] *CMD*, out uint *len*, out uint *gain*, out double *dt*)

```
{
                         byte bit1;
        byte bit2;
        byte bit3;
        byte bit4;
        byte bit5=0;


        switch (Channel)   //转换通道指令
        {

            default:              bit2 = 2; bit1 = 18; break;

            case "A1-A2 A3-A4": bit2 = 2; bit1 = 18; break;
            case "A3-A4 A5-A6": bit2 = 2; bit1 = 35; break;
            case "A5-A6 A7-A8": bit2 = 2; bit1 = 52; break;


        }


        switch (SampleRated)//采样速度指令
        {
```

```
        default: bit3 = 240; dt = 1 / 2114.25d; len = 500 * 6; break;
        case "2114.25": bit3 = 240; dt = 1 / 2114.25d; len = 500 * 6; break;
        case "1853.28": bit3 = 224; dt = 1 / 1853.28d; len = 300 * 6; break;
        case "1485.89": bit3 = 208; dt = 1 / 1485.89d; len = 300 * 6; break;
        case "1064.34": bit3 = 192; dt = 1 / 1064.34d; len = 200 * 6; break;
        case "711.162": bit3 = 176; dt = 1 / 711.162d; len = 200 * 6; break;
        case "415.602": bit3 = 161; dt = 1 / 415.602d; len = 90 * 6; break;
        case "226.963": bit3 = 146; dt = 1 / 226.963d; len = 40 * 6; break;
        case "49.0064": bit3 = 130; dt = 1 / 49.0064d; len = 10 * 6; break;
        case "29.6424": bit3 = 114; dt = 1 / 29.6424d; len = 6 * 6; break;
        case "24.7546": bit3 = 99; dt = 1 / 24.7546d; len = 5 * 6; break;
        case "14.7059": bit3 = 83; dt = 1 / 14.7059d; len = 3 * 6; break;
        case "12.4378": bit3 = 67; dt = 1 / 12.4378d; len = 2 * 6; break;
        case "7.48839": bit3 = 51; dt = 1 / 7.48839d; len = 1 * 6; break;
        case "4.8": bit3 = 35; dt = 1 / 4.8d; len = 1 * 6; break;
        case "2.4": bit3 = 19; dt = 1 / 2.4d; len = 1 * 6; break;
        case "1.15": bit3 = 3; dt = 1 / 1.15d; len = 1 * 6; break;

    }

    switch (Range)    //量程指令
    {
        default: bit4 = 1; gain = 2; break;
        case "±2.5V": bit4 = 1; gain = 2; break;
        case "±1.25V": bit4 = 2; gain = 4; break;
        case "±0.625V": bit4 = 3; gain = 8; break;
        case "±312.5mV": bit4 = 4; gain = 16; break;
        case "±156.25mV": bit4 = 5; gain = 32; break;
        case "±78.125mV": bit4 = 6; gain = 64; break;

    }

    byte[] buffer = { 170, bit1, bit2, bit3, bit4, bit5, 187 };
    CMD = buffer;


}
```

10：差分连续四通道测量模式

输入参数：

string *Samples* **样本数量（启动 ADC 内部滤波器）**

string *Range*, **量程**

string *Channel* **通道**

输出参数

byte[] *CMD* *输出最终的控制指令数组*

uint *len* *单次接收 USB 缓冲区字节长度*

uint *gain* *量程参数，代入公式计算最终输出电压*

double *dt* 时间间隔 采样速度的倒数

```
public static void CONTINUE_DIFF_FOURCHS(string SampleRated, string Range,
string Channel, out byte[] CMD, out uint len, out uint gain, out double dt)
{
        byte bit1;
        byte bit2;
        byte bit3;
        byte bit4
        byte bit5 = 0;


        switch (Channel)   //转换通道指令
        {

            default:                    bit2 = 2; bit1 = 15; break;

            case "A1-A2 A3-A4 A5-A6 A7-A8": bit2 = 2; bit1 = 15; break;

        }


        switch (SampleRated)//采样速度指令
        {
            default:         bit3 = 240; dt = 1 / 1057.26d; len = 200 * 12; break;
            case "1057.26": bit3 = 240; dt = 1 / 1057.26d; len = 200 * 12; break;
            case "926.638": bit3 = 224; dt = 1 / 926.638d; len = 150 * 12; break;
            case "742.948": bit3 = 208; dt = 1 / 742.948d; len = 100 * 12; break;
            case "532.17":   bit3 = 192; dt = 1 / 532.17d;   len = 100 * 12; break;
            case "355.581": bit3 = 176; dt = 1 / 355.581d; len = 70 * 12; break;
            case "207.801": bit3 = 161; dt = 1 / 207.801d; len = 40 * 12; break;
            case "113.481": bit3 = 146; dt = 1 / 113.481d; len = 20 * 12; break;
            case "24.5032": bit3 = 130; dt = 1 / 24.5032d; len = 5 * 12; break;
            case "14.8212": bit3 = 114; dt = 1 / 14.8212d; len = 3 * 12; break;
            case "12.3773": bit3 = 99;   dt = 1 / 12.3773d; len = 3 * 12; break;
            case "7.35295": bit3 = 83;   dt = 1 / 7.35295d; len = 2 * 12; break;
            case "6.2189":   bit3 = 67;   dt = 1 / 6.2189d;   len = 1 * 12; break;
            case "3.74419": bit3 = 51;   dt = 1 / 3.74419d; len = 1 * 12; break;
            case "2.4":        bit3 = 35;   dt = 1 / 2.4d;        len = 1 * 12; break;
            case "1.2":        bit3 = 19;   dt = 1 / 1.2d;        len = 1 * 12; break;
            case "0.575":     bit3 = 3;     dt = 1 / 0.575d;     len = 1 * 12; break;
        }

        switch (Range)   //量程指令
        {
            default: bit4 = 1; gain = 2; break;
            case "±2.5V": bit4 = 1; gain = 2; break;
            case "±1.25V": bit4 = 2; gain = 4; break;
            case "±0.625V": bit4 = 3; gain = 8; break;
            case "±312.5mV": bit4 = 4; gain = 16; break;
            case "±156.25mV": bit4 = 5; gain = 32; break;
            case "±78.125mV": bit4 = 6; gain = 64; break;

        }

        byte[] buffer = { 170, bit1, bit2, bit3, bit4, bit5, 187 };
        CMD = buffer;


    }
```

动态链接库介绍

数据转电压显示函数 DAQ_580i_VOLTDISPLAY_V5.dll
DAQ_580i_VOLTDISPLAY_V5.dll 集成以下函数；

1：单通道数据转换

输入参数：

uint *len*	*单次接收 USB 缓冲区字节长度*
uint *gain*	*量程参数，代入公式计算最终输出电压*
byte[] *data*	*从 USB 缓冲区读取到的字节数组*

输出参数

double[] *CH1*	*电压值 数组*

```
public static void ONECHS(byte[] data, uint len, uint gain, double[] CH1)
{
            string a1;
            string a2;
            string a3;
            double b;
            double y;
            uint j;
            for (j = 0; j < len / 3; j++)
            {

                a1 = Convert.ToString(data[3* j], 16);
                if (a1.Length == 1) { a1 = "0" + a1; }

                a2 = Convert.ToString(data[3 * j + 1], 16);
                if (a2.Length == 1) { a2 = "0" + a2; }

                a3 = Convert.ToString(data[3 * j + 2], 16);
                if (a3.Length == 1) { a3 = "0" + a3; }


                a3 = "0x" + a1 + a2 + a3;//合并hex 0xffffFF

                b = System.Convert.ToUInt32(a3, 16);//将16进制转十进制数值

                if (b >= 8388607)
                {
                    y = (16777215 - b) / 16777215 / gain;
                    y = y * (-1) * 10;
```

```
        }
        else
        {
            y = (b / 16777215)/gain *10; ;

        }
        CH1[j] = y;


    }

}
```

2：双通道数据转换

输入参数：

uint *len*	*单次接收 USB 缓冲区字节长度*
uint *gain*	*量程参数，代入公式计算最终输出电压*
byte[] *data*	*从 USB 缓冲区读取到的字节数组*

输出参数

double[] *CH1*	*电压值 数组*
double[] *CH2*	*电压值 数组*

```
public static void TWOCHS(byte[] data, uint len, uint gain, double[] CH1, double[]
CH2)
{
            string a1;
            string a2;
            string a3;
            double b;
            double y;
            uint j;
            for (j = 0; j < len / 6; j++)
```

```
{

    a1 = Convert.ToString(data[6 * j], 16);
    if (a1.Length == 1) { a1 = "0" + a1; }

    a2 = Convert.ToString(data[6 * j + 1], 16);
    if (a2.Length == 1) { a2 = "0" + a2; }

    a3 = Convert.ToString(data[6 * j + 2], 16);
    if (a3.Length == 1) { a3 = "0" + a3; }


    a3 = "0x" + a1 + a2 + a3;//合并hex 0xffffFF

    b = System.Convert.ToUInt32(a3, 16);//将16进制转十进制数值

    if (b >= 8388607)
    {
        y = (16777215 - b) / 16777215 / gain;
        y = y * (-1) * 10;
    }
    else
    {
        y = (b / 16777215) / gain * 10; ;

    }
    CH2[j] = y;




    a1 = Convert.ToString(data[6 * j+3], 16);
    if (a1.Length == 1) { a1 = "0" + a1; }

    a2 = Convert.ToString(data[6 * j + 4], 16);
    if (a2.Length == 1) { a2 = "0" + a2; }

    a3 = Convert.ToString(data[6 * j + 5], 16);
    if (a3.Length == 1) { a3 = "0" + a3; }


    a3 = "0x" + a1 + a2 + a3;//合并hex 0xffffFF

    b = System.Convert.ToUInt32(a3, 16);//将16进制转十进制数值
```

```
        if (b >= 8388607)
        {
            y = (16777215 - b) / 16777215 / gain;
            y = y * (-1) * 10;
        }
        else
        {
            y = (b / 16777215) / gain * 10; ;

        }
        CH1[j] = y;


    }
}
```

3：四通道数据转换

输入参数：

uint *len*	*单次接收 USB 缓冲区字节长度*
uint *gain*	*量程参数，代入公式计算最终输出电压*
byte[] *data*	*从 USB 缓冲区读取到的字节数组*

输出参数

double[] *CH1*	*电压值 数组*
double[] *CH2*	*电压值 数组*
double[] *CH3*	*电压值 数组*
double[] *CH4*	*电压值 数组*

public static **void FOURCHS**(**byte[]** *data*, **uint** *len*, **uint** *gain*, **double[]** *CH1*,

```
double[] CH2, double[] CH3, double[] CH4)
{
            string a1;
            string a2;
            string a3;
            double b;
            double y;
            uint j;
            for (j = 0; j < len / 12; j++)
            {

                a1 = Convert.ToString(data[12 * j], 16);
                if (a1.Length == 1) { a1 = "0" + a1; }

                a2 = Convert.ToString(data[12 * j + 1], 16);
                if (a2.Length == 1) { a2 = "0" + a2; }

                a3 = Convert.ToString(data[12 * j + 2], 16);
                if (a3.Length == 1) { a3 = "0" + a3; }



                a3 = "0x" + a1 + a2 + a3;//合并hex 0xffffFF

                b = System.Convert.ToUInt32(a3, 16);//将16进制转十进制数值

                if (b >= 8388607)
                {
                    y = (16777215 - b) / 16777215 / gain;
                    y = y * (-1) * 10;
                }
                else
                {
                    y = (b / 16777215) / gain * 10; ;

                }
                CH4[j] = y;




                a1 = Convert.ToString(data[12 * j+3], 16);
                if (a1.Length == 1) { a1 = "0" + a1; }
```

```
a2 = Convert.ToString(data[12 * j + 4], 16);
if (a2.Length == 1) { a2 = "0" + a2; }

a3 = Convert.ToString(data[12 * j + 5], 16);
if (a3.Length == 1) { a3 = "0" + a3; }


a3 = "0x" + a1 + a2 + a3;//合并hex 0xffffFF

b = System.Convert.ToUInt32(a3, 16);//将16进制转十进制数值

if (b >= 8388607)
{
    y = (16777215 - b) / 16777215 / gain;
    y = y * (-1) * 10;
}
else
{
    y = (b / 16777215) / gain * 10; ;

}
CH1[j] = y;




a1 = Convert.ToString(data[12 * j+6], 16);
if (a1.Length == 1) { a1 = "0" + a1; }

a2 = Convert.ToString(data[12 * j + 7], 16);
if (a2.Length == 1) { a2 = "0" + a2; }

a3 = Convert.ToString(data[12 * j + 8], 16);
if (a3.Length == 1) { a3 = "0" + a3; }


a3 = "0x" + a1 + a2 + a3;//合并hex 0xffffFF

b = System.Convert.ToUInt32(a3, 16);//将16进制转十进制数值

if (b >= 8388607)
{
    y = (16777215 - b) / 16777215 / gain;
    y = y * (-1) * 10;
    }
    else
    {
        y = (b / 16777215) / gain * 10; ;

    }
    CH2[j] = y;



    a1 = Convert.ToString(data[12 * j+9], 16);
    if (a1.Length == 1) { a1 = "0" + a1; }

    a2 = Convert.ToString(data[12 * j + 10], 16);
    if (a2.Length == 1) { a2 = "0" + a2; }

    a3 = Convert.ToString(data[12 * j + 11], 16);
    if (a3.Length == 1) { a3 = "0" + a3; }



    a3 = "0x" + a1 + a2 + a3;//合并hex 0xffffFF

    b = System.Convert.ToUInt32(a3, 16);//将16进制转十进制数值

    if (b >= 8388607)
    {
        y = (16777215 - b) / 16777215 / gain;
        y = y * (-1) * 10;
    }
    else
    {
        y = (b / 16777215) / gain * 10; ;

    }
    CH3[j] = y;



}
```