■■■■■■精益求精■■用心服务■■■■■■

CH372USB 模块调试说明

LENCHI V2.0 单片机开发系统配套教程■■■■■■

http://lenchimcu.taobao.com
蓝旗嵌入式系统工作室
产品配套手册

****

Date :2011/07/06

教程版本说明：

版本号	完成日期	主要修改内容	备注
Ver1.0	**2011.07.06**	初稿	

复杂源于简单　创新源于基础

http://lenchimcu.taobao.com ▪▪▪▪▪▪蓝旗嵌入式▪▪▪▪▪▪

目录

CH372 USB 模块调试说明

1.CH372 USB 模块简介

CH372 USB 模块是蓝旗嵌入式系统工作室开发的一款 USB 总线通用模块，它既可以直接应用到您实际的系统中，也可以作为学习模块来使用。

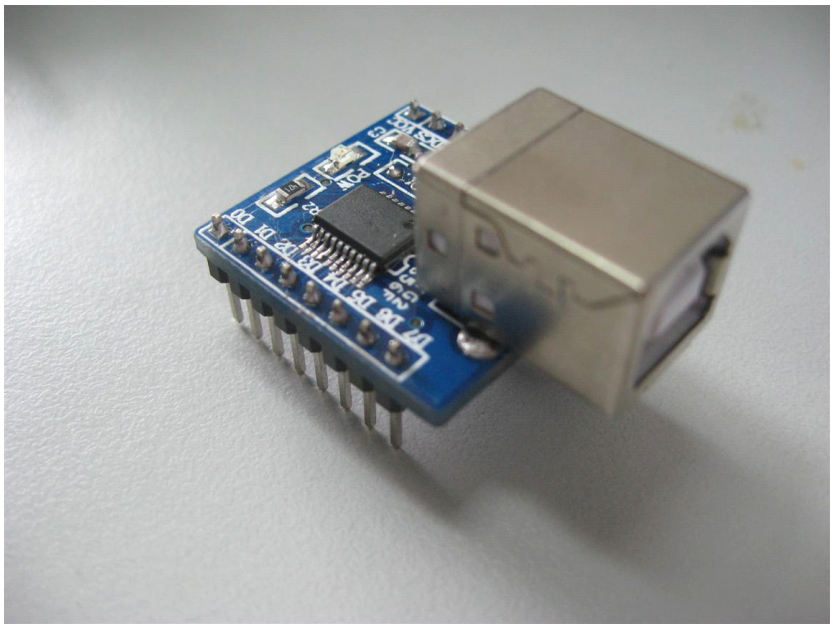

图 1 CH372 模块靓照

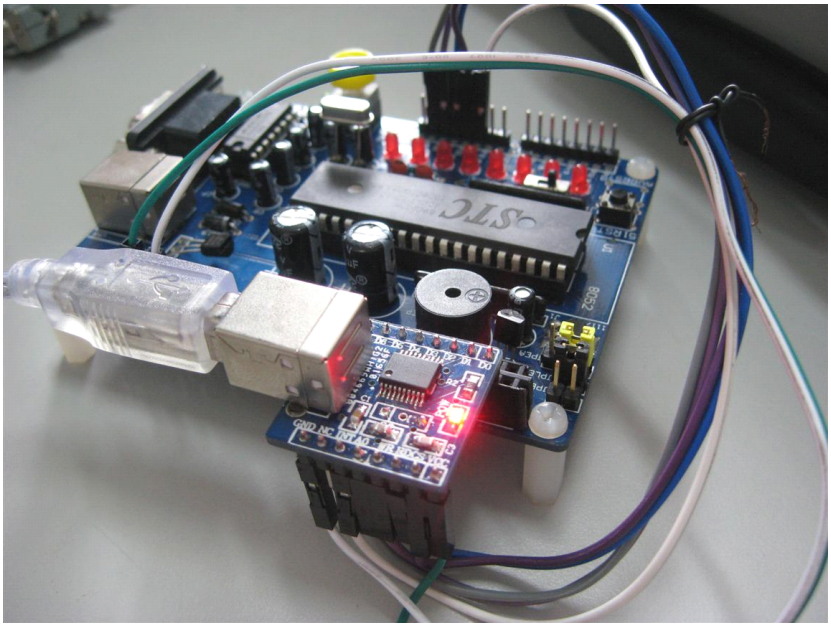

图 2 CH372 模块与单片机开发板相连

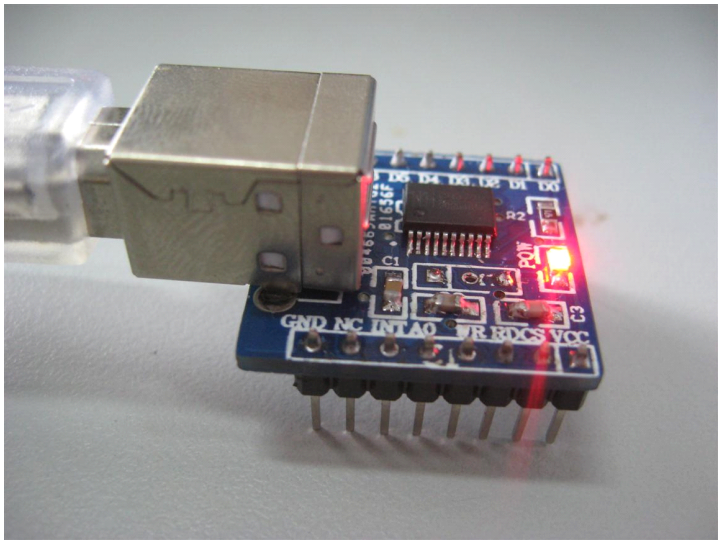

图 3 CH372 模块特写

CH372USB 模块用于上下位机之间通过 USB 接口进行通信，可以应用到单片机、DSP、ARM 等所有 MCU、MPU 与计算机的通信中。模块提供 8 位数据线及读、写、片选、中断线各一条，使用此模块可以快速的挂接到下位机系统，配合我们提供的上下位机实例，只需要几分钟即可完成连接和通信。

CH372 是南京沁恒生产的一款 USB 总线芯片。它内置 USB 通信底层协议，具有方便简单的内置固件模式和灵活的外置固件模式。内置固件模式下，芯片自动处理端点 0 的所有事务，本地单片机只需要负责处理数据交换即可，程序简洁、简单；外置固件模式下，由本地单片机负责处理各种 USB 请求，可以灵活的实现符合 USB 规范的设备。

芯片的连接简图如下图所示。

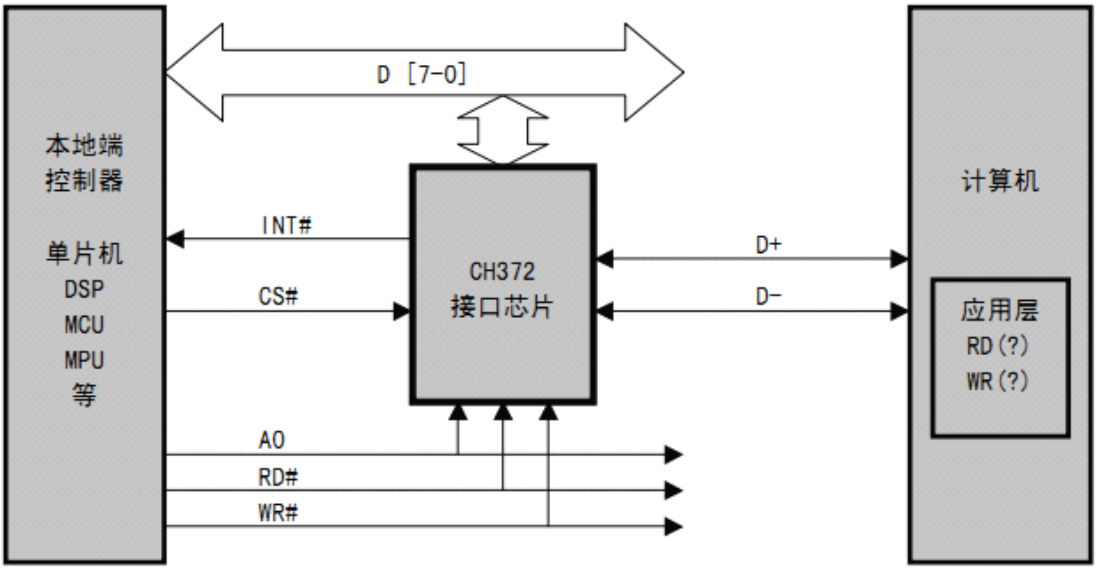

图 4 CH372 连接简图

特点：全速 USB 接口，兼容 USB2.0，即插即用，外围器件少；

支持控制传输、批量传输、中断传输；

内置固件模式、外置固件模式任你选择；
提供 Windows 驱动程序，提供 API 接口；
厂商可自定义厂商标示和产品 ID；
接口简单，通用 8 位数据线，5 位控制线；
主端点上下传输缓冲区各 64 字节，辅助端点 8 字节；
5V、3.3V 电源通吃，支持低功耗模式；

因为 CH372 与 CH375 资料通用，所以在文档中有些部分写的是 CH375。CH375 是 CH372 的增强版。

2.CH372 USB 模块硬件介绍

CH372 USB 模块提供了一个 USB-B 型接口，方便模块通过 USB 线与计算机连接；另外，模块将 8 根数据线，5 根控制线全部引出，方便模块与您的系统连接；模块还引出了电源线和地线，方便给模块供电或通过计算机 USB 接口给您的系统供电；模块带有一个 LED 电源指示灯，用于指示连接与否。

CH372 USB 模块的电路图如图 5 所示。

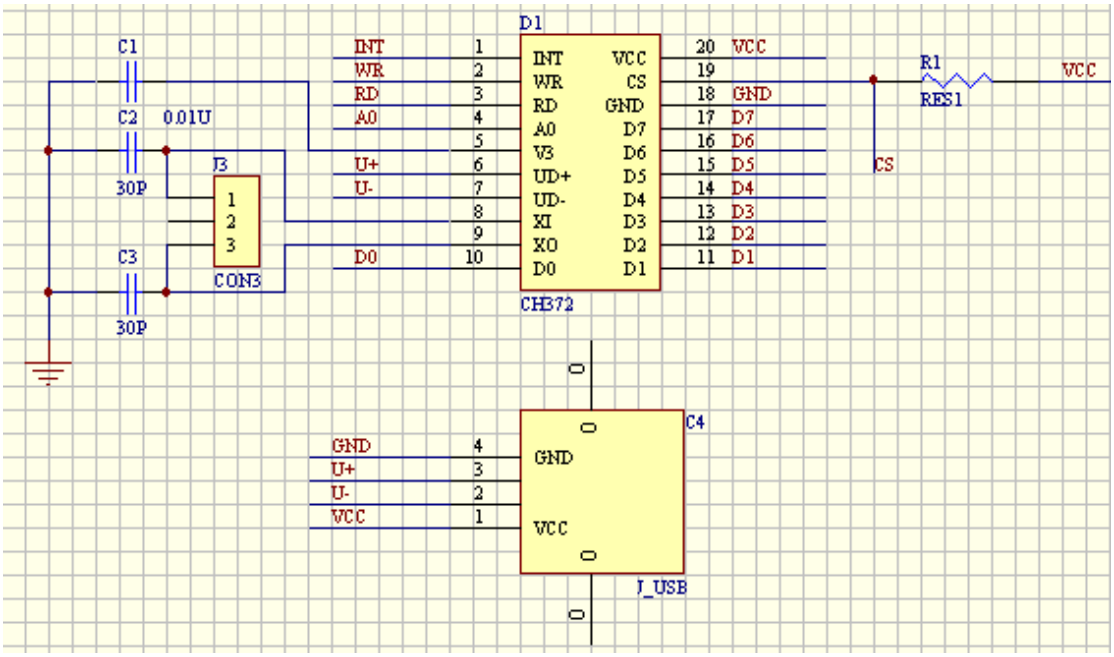

图 5 CH372 USB 模块电路图

图中 J3 为 12M 的晶振，用于给芯片提供时钟信号。晶振的两脚对地连接高频振荡电容，电容在电路图中的标号为 C2、C3，电容的值为 30p，当然电容的值可以是 15p、22p，如果电源使用的是 3.3V，晶振不容易起振时，可以将两个电容换为 6p，或干脆去掉，具体要在实际中进行调试。

图中的 D0-D7 为芯片的数据 I/O 口，用于上下位机间的数据交换，可以与下位机的 I/O 直接相连，无需外接器件。

CH372 现在在产型号为 CH372B，这种型号的芯片支持 5V 和 3.3V 的电源。当使用 3.3V 电源时，芯片的 V3 脚应该与 VCC 脚相连，并外接 3.3V 外部电源，同时与芯片其它管脚相连的电压不能超过 3.3V；当使用 5V 电源时，VCC 管脚外接 5V 电源，V3 管脚需接退耦电容，电容的值为 1000p-0.01u，推荐使用 0.01u。退偶电容在电路图中的标号为 C1。

图中的 WR 和 RD 为写选通和读选通引脚，它们可以与下位机的任意 I/O 相连；WR 为高电平，并且 CS、A0、RD 都为低电平时，芯片中的数据通过 D0-D7 输出；RD 为高电平，并且 CS、A0、WR 都为低电平时，D0-D7 的数据写入芯片。

A0 为地址线输入，用于区分命令与数据，芯片在此管脚内置上拉电阻，当 A0 为低电平时，读写的是数据；当 A0 为高电平时，读写的是命令。

CS 为片选脚，当下位机总线挂载多个器件时，通过此管脚输入低电平选择芯片；当下位机只有 CH372 一个器件时，可以直接将其接地。

INT 为中断信号输出脚，当输出低电平时代表中断请求信号。可以连接到下位机的中断脚或 I/O 脚，通过中断或查询的方式获取中断信号。一次中断请求，代表一次数据传输完毕。

图中的 J_USB 代表 USB-B 型母口，1 脚为 VCC，是计算机方传输过来的电源，4 脚为 GND。3 脚为 U+，2 脚为 U-，分别与芯片的 U+和 U-管脚相连，无需外接器件，如果为了保护芯片而串接保险丝、电感、ESD 保护器件，那么其等效串联电阻应小于 5 欧姆。

3.CH372 USB 模块软件介绍

使用 CH372 芯片用于上下位机间的数据通信时的软件包含两个部分。一部分为下位机程序，另一部分为上位机软件。

3.1 单片机端的程序

先来说一下下位机的程序，以 51 单片、内置固件模式、中断传输为例。

用 Keil 或其它类似软件新建一个工程，选择芯片型号，例子中使用的是 52 子系列，开始编写程序。

```
#include<reg51.h>
#include "CH375INC.H"
```

程序的开头除了包含单片机头文件之外，还需要包含 CH372 的 C 语言头文件，具体请见例子的文件夹之内。当然，CH372 的头文件也可以不用包含，但相关的命令值需要自己在程序中定义。

然后，需要定义 CH372 模块的相关控制线的脚位。

```
sbit CH375_INT=P3^2;   //CH375 中断信号输出端，低电平输出
sbit CH375_CS=P3^3;    //CH375 片选控制端，低电平有效，芯片内置上拉电阻
sbit CH375_RD=P3^4;    //CH375 读控制端，低电平有效，芯片内置上拉电阻
sbit CH375_WR=P3^5;    //CH375 写控制端，低电平有效，芯片内置上拉电阻
sbit CH375_A0=P3^6;    //CH375 命令/数据选择控制端，低电平时写数据，高电平时写命令
```

CH372 的数据线在例子中与单片机的 P0 口对应序号相连。

接下来需要写两个延时函数，因为 CH372 的时序需要对持续时间有保证，特别是在主控芯片处理速度比较块的情况之下，必须需要延时。

```
/*******************************************************************************
****函数名称：不准确的延时 2US 的函数
****函数作用：
****函数描述：
*******************************************************************************/
void delay2us(void)
{
 unsigned char i;
 for(i=0;i<2;i++);
}
/*******************************************************************************
****函数名称：不准确的延时 50mS 的函数
****函数作用：
****函数描述：
*******************************************************************************/
void delay50ms(void)
{
 unsigned char m,n;
 for(m=0;m<200;m++)
        for(n=0;n<250;n++);
}
```

紧接着我们需要编写 CH372 的写命令函数、写数据函数、读数据函数，相关函数按照手册的时序编写即可，相关注意事项请见函数的注释。

```
/*******************************************************************************
****函数名称：CH375 写命令函数
****函数作用：向 375 写指定的命令
****函数描述：具体命令请见 375 的头文件中的定义
*******************************************************************************/
void CH375_WRCMD(unsigned char cmd)
{
 delay2us();
 P0=cmd;
 CH375_A0=1;
 CH375_RD=1;
 CH375_CS=0;
 CH375_WR=0;
 CH375_WR=1;
 CH375_CS=1;
 CH375_A0=0;
```

```
 P0=0xFF;              //单片机的端口全部拉高
 delay2us();
}
/*******************************************************************************
****函数名称：CH375 写数据函数
****函数作用：向 375 写指定的数据
****函数描述：
*******************************************************************************/
void CH375_WRDAT(unsigned char dat)
{
 P0=dat;

 CH375_A0=0;
 CH375_CS=0;
 CH375_WR=0;
 //CH375_CS=0;       //延时用，因为单片机运行比较慢，可省略，运算快的处理器不可省略
 CH375_WR=1;
 CH375_CS=1;

 P0=0xFF;            //单片机的端口全部拉高

}
/*******************************************************************************
****函数名称：从 CH375 读数据函数
****函数作用：从 375 的缓冲区中读取收到的数据
****函数描述：返回读到的数据
*******************************************************************************/
unsigned char CH375_REDAT(void)
{
 unsigned char temp;
 P0=0xFF;            //单片机的端口全部拉高

 CH375_A0=0;
 CH375_CS=0;
 CH375_RD=0;
 temp=P0;
 CH375_RD=1;
 CH375_CS=1;

 P0=0xFF;            //单片机的端口全部拉高

 return temp;
}
```

注意各个程序中在函数的结尾或开头处都有将单片机的数据端口拉高的程序语句，这一句不可以省略，笔者在编写程序的初期就曾因为没有将端口拉高而走了很多弯路。

因为我们采用的是中断处理数据，所以还需要编写单片机的中断处理函数。我们占用的是单片机的外部中断 0，因为 CH372 芯片的中断脚有效时输出低电平，所以需要将单片机外部中断 0 设置为低电平触发。

先设置单片机的中断：

```
IT0=0;                                          //设置外部中断 0 为低电平出发方式
EX0=1;                                          //使能外部中断 0
EA=1;
```

然后编写外部中断 0 的中断处理函数：

```
/*******************************************************************************
****函数名称：外部中断 0 的响应函数
****函数作用：通过判断 375 的中断类型值进行相应的处理
****函数描述：
*******************************************************************************/
void int0(void) interrupt 0 using 0
{
 unsigned char i,temp,length;
 unsigned char data buf[64];                    //接收到的数据存放的缓冲数组，64 个位置

 CH375_WRCMD(CMD_GET_STATUS);   //发获取 375 的中断类型（状态）命令，并取消中断信号

 temp=CH375_REDAT();                            //读取中断类型（状态）值

 switch(temp)                                   //判断中断类型（状态），具体区分请见 375 头文件中的定义
 {
  case  USB_INT_EP2_OUT:            //如果是 0x02，则是端点 2 的 OUT 事务（接收到数据，OUT 成
                                    功），OUT 是相对于主机端计算机来说的
        CH375_WRCMD(CMD_RD_USB_DATA);           //发读取 USB 缓冲区数据命令，并释放缓
                                                冲区
        length=CH375_REDAT();                   //先读取数据长度
        for(i=0;i<length;i++)                   //根据数据长度，读取所有值
        {
            buf[i]=CH375_REDAT();               //读取的数据放入缓冲数组
        }

        /*以下为向主机端计算机发送接收到的数据*/
        CH375_WRCMD(CMD_WR_USB_DATA7);          //发向端点 2 的发送缓冲区写数据命令
        CH375_WRDAT(length);                    //先写入数据的长度
        for(i=0;i<length;i++)                   //根据长度，依次发送要发送的数据，数据取反
            CH375_WRDAT(~buf[i]);
        break;                                  //跳出

    case  USB_INT_EP2_IN:       //如果是 0x0A，则是端点 2 的 IN 事务（发送完数据，IN 成功），
                                IN 是相对于主机端计算机来说的

        CH375_WRCMD(CMD_UNLOCK_USB);            //发送释放缓冲区命令
        break;                                  //跳出
    default:
```

```
            CH375_WRCMD(CMD_UNLOCK_USB);
            break;
    }
    }
```

在单片机的中断处理函数中，先通过向 CH372 发送命令，读取 CH372 的中断值，并取消中断。然后判断中断值，并根据不同的值来做相应的处理。注意，在程序中使用到的 IN 和 OUT 都是相对于计算机端来说的。

例子的程序作用是接收计算机发送来的数据，并将它们取反后再发回给计算机。

再来看一下主函数。

```
/*******************************************************************************
****函数名称：主函数
****函数作用：
****函数描述：
*******************************************************************************/
void main(void)
{
 unsigned char i;
 delay50ms();                                    //延时 50ms

 //注释掉的为测试硬件是否正确，实际应用中可以不用
 //CH375_WRCMD(CMD_CHECK_EXIST);
 //CH375_WRDAT(0x01);
 //i=~0x01;
 //if(CH375_REDAT()==i)
 //P1=i;
 //else
 //P1=0;

 CH375_WRCMD(CMD_SET_USB_MODE);                  //设置工作模式
 CH375_WRDAT(2);                                 //设置为内置固件模式
 IT0=0;                                          //设置外部中断 0 为低电平出发方式
 EX0=1;                                          //使能外部中断 0
 EA=1;                                           //使能所有中断
 while(1);                                       //等待
}
```

程序的开头先延时 50ms 左右，等待芯片完成复位。然后先看一下程序段中注释掉的部分，这一部分的作用是测试 CH372 的硬件连接是否正确，这一步在模块钢焊接完时非常有用，可以用来检验模块的硬件连接是否正确或芯片是否损坏，当然，正式程序中可以将它们注释掉。测试的方法是向芯片发送一个 CMD _CHECK_EXIST 命令，然后向芯片发送任意数值，如果硬件部分完好，则芯片会返回这个数值的取反以后的值。

测试通过之后需要设置芯片的工作模式，例子中使用的是芯片的内置固件模式。设置好芯片工作模式之后就是初始化中断并等待中断发生了。

由上述内容可见，单片机段的软件在使用内置固件时是非常简介明了的。读者可以根据这个例子来编写适合自己需要的程序。

3.2 计算机端软件的编写

了解了单片机端的 CH372 的程序之后，我们再来了解一下计算机端软件的编写方法。

CH372 的厂家提供芯片的 Windows 环境下的驱动程序，并以 API 的形式提供编程接口。来看一下计算机端软件是如何编写的，我们以 XP 环境、VS2005 C# 为例来讲一下。

首先，按照 3.1 编写好单片机端的程序之后，烧写进单片机运行，将 CH372 与单片机按照电路图连接，并经过测试没有问题。这时候将 CH372 通过 USB 线与计算机相连，连接之后会提示发现新硬件，并要求安装驱动。我们可以选择手动安装，并将目录定位到“...CH372usb 资料\官方资料\计算机上安装的驱动\CH372DRV\DRIVER”，这时候会自动安装，安装完成之后，USB 模块就可以使用了。

我们可以先用官方提供的工具来测试一下，测试工具在“CH372usb 资料\调试工具”中，打开之后，如果 USB 模块连接无误，则软件界面会提示设备已插入，具体如图 6 左下角所示。

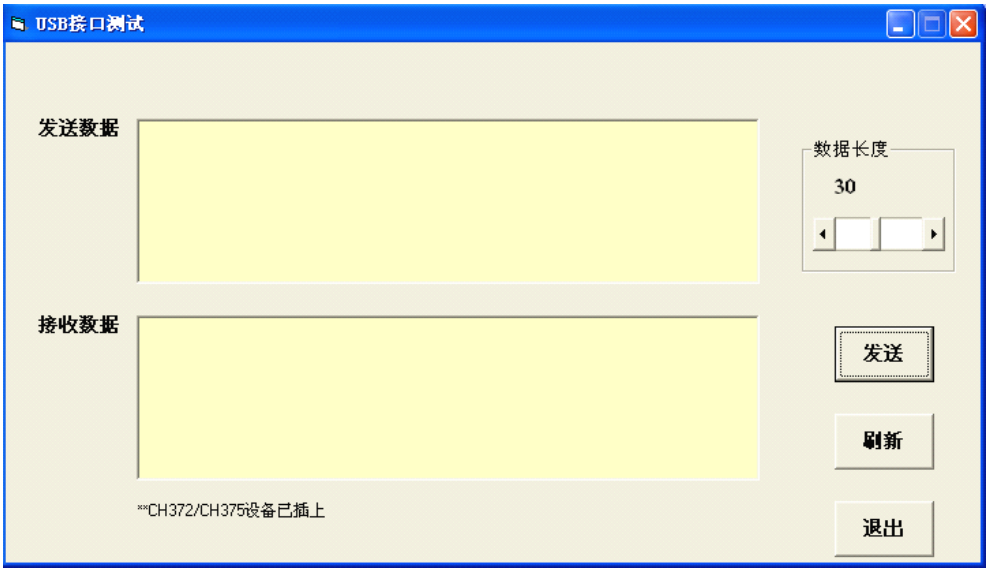

图 6 USB 调试工具界面

软件刚打开之后默认的数据长度是 30，我们可以点击发送按钮，软件会向模块发送随机的 30 个字符，并接收模块返回的数值，之后会判断发送和接受的数值是否相同。具体如图 7 所示。

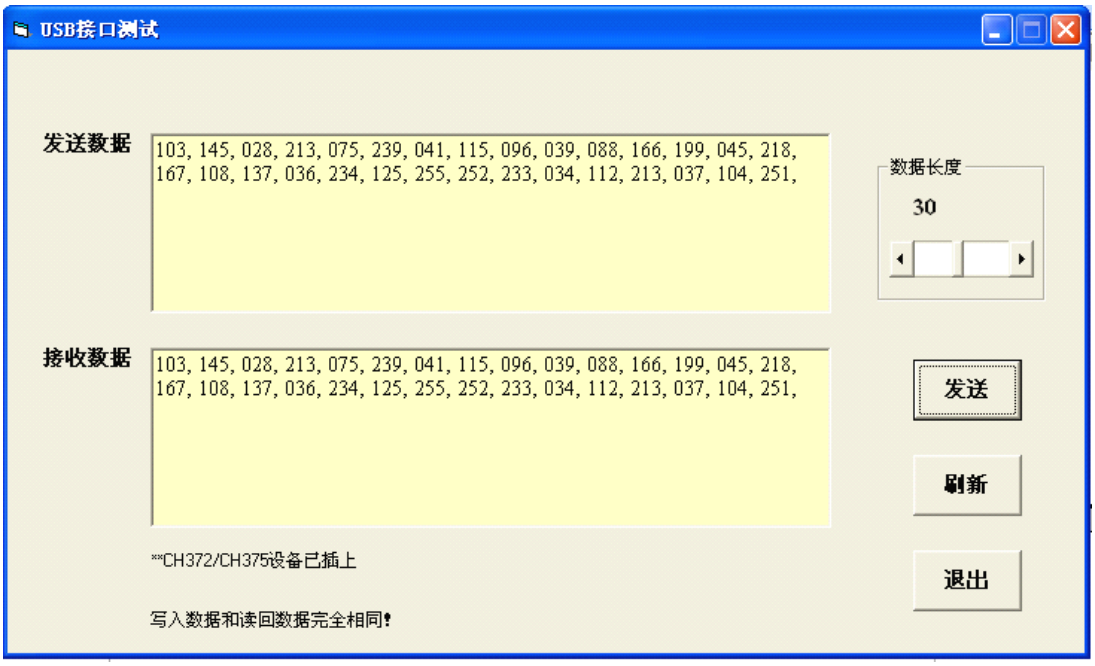

图 7 调试

如果读者测试的界面同 7 相似，则恭喜您，您的模块正确无误，可以编写上位机软件了。

接下来使用 VS2005 新建一个 C#的工程。

然后添加 CH375DLL.DLL，它提供给我们编程使用的 API。因为官方提供的 DLL 不是专门为 C#而写，所以不能在“引用”中添加，而应该在程序中添加。

为了调用方便，我们可以专门新建一个类用来编写 CH372 的相关函数，这个类我们不妨命名为“ClassUSBCOM”。

编写 CH372 的相关函数时我们可以参考 API 的函数头文件，头文件位置在“CH372usb 资料\官方资料\计算机上安装的驱动\CH372DRV\LIB\C\CH375DLL.H”。举例说明函数的引用方法，如果想要引用 API 中的写数据函数，则在程序中如下编写：

```
[DllImport("CH375DLL.DLL")]
public static extern bool CH375WriteData(int iIndex, [MarshalAs(UnmanagedType.LPArray)] int[] iBuffer,
                                          [MarshalAs(UnmanagedType.LPArray)] int[] ioLength);
```

原函数中参数是有指针的，C#中是没有指针这个定义，所以我们用数组来传递参数，具体的 C#相关的编程理论请大家参阅有关书籍文献，在此不赘述。DLL 中其它函数的调用也与此相似。

在 C#编程中使用到的函数都可以使用此方法调用。具体的函数可以与控件关联。实例的作用是打开 USB、关闭 USB、发送数据、接受数据等常规操作，读者可以在此基础上改进，以实现自己的功能。

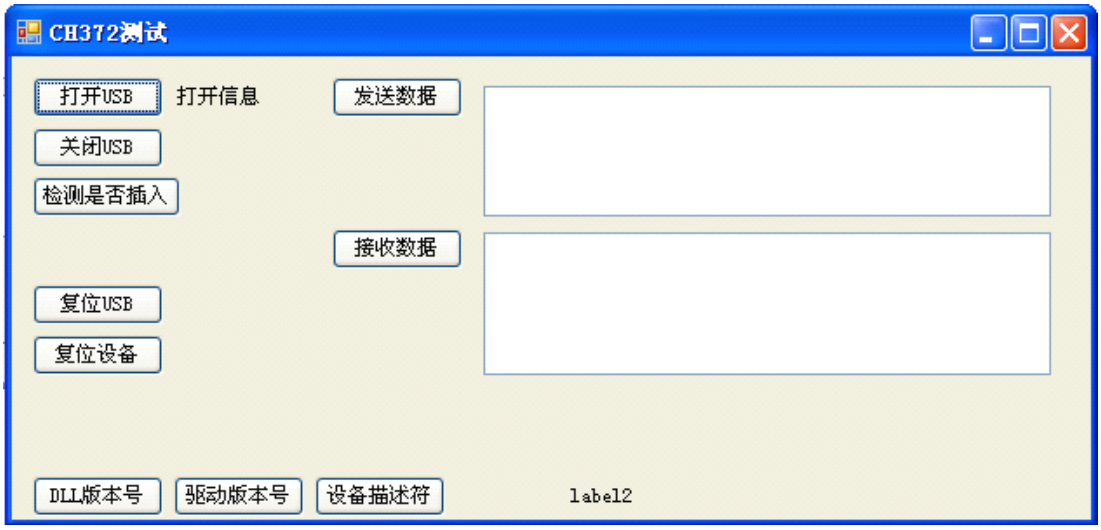

图 8 C#程序界面

另外，我们提供的资料包中还有官方提供的 VC、VB、Delphi 等不同语言版的程序例子。

4.模块保修事宜

模块在发货之前我们都会进行测试，确保用户拿到手上的产品合格率为 100%。

保修政策为：

1 周内包换，如非产品质量原因，邮费须由用户负责；

1 年内免费保修，如非产品质量原因，邮费及器件费用由用户负责；

1 年之后终生保修，邮费及器件成本费由用户负责。

5.联系方式

我们的联系方式为：

地址：江苏省无锡市蠡湖大道 1800 号 江南大学物联网工程学院 D508 室

邮编：214122

电话：13961733341

QQ：412917499

Email:lenchimcu@qq.com

淘宝店：http://lenchimcu.taobao.com

如果您对我们的产品满意，请大声告诉您的朋友！

如果您对我们的产品不满意，请悄悄的告诉我！