校园一卡通实验

C0201 指导文档　用户登记

1. 实验目的

该实验的目的是让学生知道如何获取 ISO14443 的卡号并把卡号和用户信息一起登记到数据库中

2. 实验设备

软件：visualstudio2010 及以上版本，MicrosoftSQLServer2005 及以上版本
硬件：5V2A 电源，高频 14443 读写器，高频 14443 标签。串口线

3. 实验原理

使用高频 14443 读写器读取高频 14443 标签标签号，和用户填写的基本信息一起写入到数据库中，以便其他模块使用。如果标签号已经登记了用户信息，会提示标签号已登记。

4. 界面设计

控件名称	控件 Text 属性	控件 Name 属性	功能
Form 窗体	C0201	frmMain	
ComboBox 控件		cmbPortNum	获取计算机串口
Lable 标签	姓名	LblName	
TextBox 文本框		txtName	
Lable 标签	性别	LblSex	
TextBox 文本框		txtSex	
Lable 标签	身份证号	lblIDCard	
TextBox 文本框		txtIDCard	
Lable 标签	充值金额	lblMoney	
TextBox 文本框		txtMoney	
Button 按钮	打开串口	btnOpenSerialPort	打开串口
Button 按钮	关闭串口	btnCloseSerialPort	关闭串口
Button 按钮	获取卡型	btnGetCardType	获取卡型
Button 按钮	读取卡号	btnReadCard	读取卡号

TextBox 文本框		txtCardID	
Button 按钮	登记信息	btnRegister	在数据库中登记用户信息

5. 功能设计

首先添加下面两条引用命名空间的代码。

```
using KV_ISO14443;
using System.Threading;
```

(1):打开串口和关闭串口功能

A）定义变量

```
private FR102 Reader;
private String strSnr;
private Byte[] BArraySnr, ByteArrayKeyA;
KV_ISO14443.FR102 iso14443 = new FR102();
```

B)在主窗体 frmMain 的 Load 事件中调用 GetComList()方法使打开窗体时从注册表里获取所有可以使用的端口的端口号并将端口号绑定到端口号下拉框.

```
private void frmMain_Load(object sender, EventArgs e)
{
    GetComList();
    Reader = new FR102();
}

public void GetComList()
{
//RegistryKey keyCom =
Registry.LocalMachine.OpenSubKey("Hardware\\DeviceMap\\SerialComm");
```

```
string[] portNames = System.IO.Ports.SerialPort.GetPortNames();   //从注册表里获取所有可以使用的端口的端口号
if (portNames != null)                                             //如果有可用的串口
            {
                cmbPortNum.Items.AddRange(portNames);                          //把所有可用的串口项添加到下拉列表框
if (cmbPortNum.Items.Count > 0)                                    //如果下拉列表框不为空
                {
                    cmbPortNum.SelectedIndex = 0;                              //默认选定下拉列表框的第一项
                }
else
                {
MessageBox.Show(this.Owner, "没有找到可以使用的端口，请检查端口线是否已经连接到电脑上。", "提示", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                }

            }
else
            {
MessageBox.Show(this.Owner, "没有找到可以使用的端口，请检查端口线是否已经连接到电脑上。", "提示", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
        }
```

C)实例化 TagReader 类,通过对象调用 TagReader 类的 OpenSerialPort()方法,打开 cmbPortNum 列表中选定的串口.

```
private void btnOpenSerialPort_Click(object sender, EventArgs e)
        {
if (Reader.OpenSerialPort(cmbPortNum.Text.Trim()) != 0x00)
            {
MessageBox.Show(String.Format("打开端口:端口打开失败！\n请检查端口{0}是否被其它程序占用", cmbPortNum.Text.Trim()));
return;
            }
else
            {
if (Reader.TestReader() != 0x00)
                {
MessageBox.Show(String.Format("没有检测到连接到端口{0}的设备，请检查与设备连接的端口！", cmbPortNum.Text.Trim()));
if (Reader.CloseSerialPort() == 0x00)
                    {
MessageBox.Show(String.Format("关闭端口:端口{0}关闭成功！", cmbPortNum.Text.Trim()));
                    }
```

```
return;
                }

if (Reader.RestartReader() != 0x00)
                {
MessageBox.Show("设备启动失败！");
return;
                }
MessageBox.Show(String.Format("打开端口:端口{0}打开成功！", cmbPortNum.Text.Trim()));
return;

            }
        }
```

D)实例化 ISO14443Reader 类,通过该类的对象调用 CloseSerialPort()方法关闭 cmbPortNum 列表中选定的串口.

```
        private void btnCloseSerialPort_Click(object sender, EventArgs e)
        {
if (FR102.StatusCode.AllDone == iso14443.CloseSerialPort())
            {
MessageBox.Show("关闭成功");
return;
            }
else
            {
MessageBox.Show("关闭失败");
return;
            }
        }
```

注：ISO14443Reader 类和 TagReader 类都是接口 ISO14443.DLL 接口的中的类,使用之前需要添加该接口的引用,最好把该接口存放在项目的 bin 目录下.

(2)获取卡型和读取卡号功能

A)通过 FR102 类的对象调用 PcdRequest(Byte req_code, out Byte[] TagType)方法获取卡型

```
        private void btnGetCardType_Click(object sender, EventArgs e)
        {
byte[] TagType;
FR102.StatusCode ec = Reader.PcdRequest(0x52, out TagType);
if (ec == FR102.StatusCode.AllDone)
            {
MessageBox.Show("获取成功");
return;

            }
else if (ec == FR102.StatusCode.NoTagErr)
```

```
            {
MessageBox.Show("请求失败:请求码0x52，失败原因是在读写器天线场区内无标签");
return;
            }
else
            {
MessageBox.Show("请求失败:请求码0x52，失败原因不详");
return;
            }
        }
```

B)通过 FR102 类的对象调用 PcdAnticoll(out Byte[] TagNumber)方法读取读写器上的卡的卡号并把获取到的卡号显示在 txtCardID 文本框中

```
        private void btnReadCard_Click(object sender, EventArgs e)
        {
Byte[] data = new Byte[1];
FR102.StatusCode value = Reader.PcdAnticoll2(out data);
if (value == FR102.StatusCode.NoTagErr)
                {
Thread.Sleep(1000); //让程序停止一秒，而后尝试重新搜寻卡片。
                }

                strSnr = "";
                BArraySnr = new Byte[data.Length];
for (Byte i = 0; i < data.Length; i++)
                {
                    strSnr = strSnr + String.Format("{0:X2} ", data[i]);
                    BArraySnr[i] = data[i];
                }
if (strSnr != "")
                {
MessageBox.Show(String.Format("找到标签，其卡号为:{0}", strSnr));
                    txtCardID.Text = strSnr;
                }
if (data.Length < 4)
                {
MessageBox.Show(String.Format("找到标签，其卡号为:{0}，长度不足6字节！", strSnr));
return;
                }
        }
```

注：（1） 在程序中需要给 PcdRequest()方法传一个参数 0X52 即给 ISO14443 读写器发送寻天线区内所有卡命令,寻到卡返回“获取成功”

(2)PcdAnticoll()方法将会返回所寻到的卡的卡号,由于卡号是 Byte 型,所以需要用 Byte 型变量接收,再转化成 string 才能显示在控件中，为了保险起见读取卡号功能放在循环里，循环三次，即获取卡号三次

登记信息功能

输入姓名性别和身份证号后单击登记信息按钮通过调用 SqlHelper 类中的 Insert()方法在数据库中登记用户输入的信息,调用 IsSuccess()方法获取数据库中受影响的行数,如果返回的值大于 0,则提示登记信息成功!否则提示登录失败.

```
        private void btnRegister_Click(object sender, EventArgs e)
        {
if (txtCardID.Text == "" || txtName.Text == "" || txtSex.Text == "" || txtIDCard.Text
== ""||txtMoney.Text=="")
            {
MessageBox.Show("卡号、姓名、性别、身份证号、充值金额都不能为空!");
            }
else
            {
sqlHelper sh = new sqlHelper();
                sh.Insert(strSnr,txtName.Text.ToString(),
txtSex.Text.ToString(),txtIDCard.Text.ToString(),txtMoney.Text.ToString());
int i = sh.IsSuccess();
if (i > 0)
                {
MessageBox.Show("登记信息成功!");
                }
else
                {
MessageBox.Show("数据库错误!登记信息失败!");
                }

            }
        }
```

注:(1)登记信息功能只有成功完成打开串口,获取卡型,读取卡号操作,并且将所有信息填写完整才能使用.在 C0201.cs 中需要引用命名空间 using TagReader;using PracticeSystem;using System.Threading;在 sqlHelper.cs 中需要引用命名空间 using System.Data 和 usingSystem.Data.sqlClient,否则程序会出现错误.

(2) Insert()方法通过 SqlCommand 对象与数据库进行连接并对数据库执行特定的插入语句,并把数据库中受影响的行数保存在变量 i 中

(3) IsSuccese ()方法通过获取 Insert()方法中返回的 i 的值判断数据库中受影响的行数,返回值大于 0 则说明登记信息操作执行成功,不大于 0 则说明登记信息操作失败

C0202 指导文档　充值消费

1. 实验目的

该实验的目的是让学生知道如果使用 ISO14443 卡如何实现校园一卡通的充值和消费功能

2. 实验设备

软件：visualstudio2010 及以上版本，MicrosoftSQLServer2005 及以上版本
硬件：5V2A 电源，高频 14443 读写器，高频 14443 标签。串口线

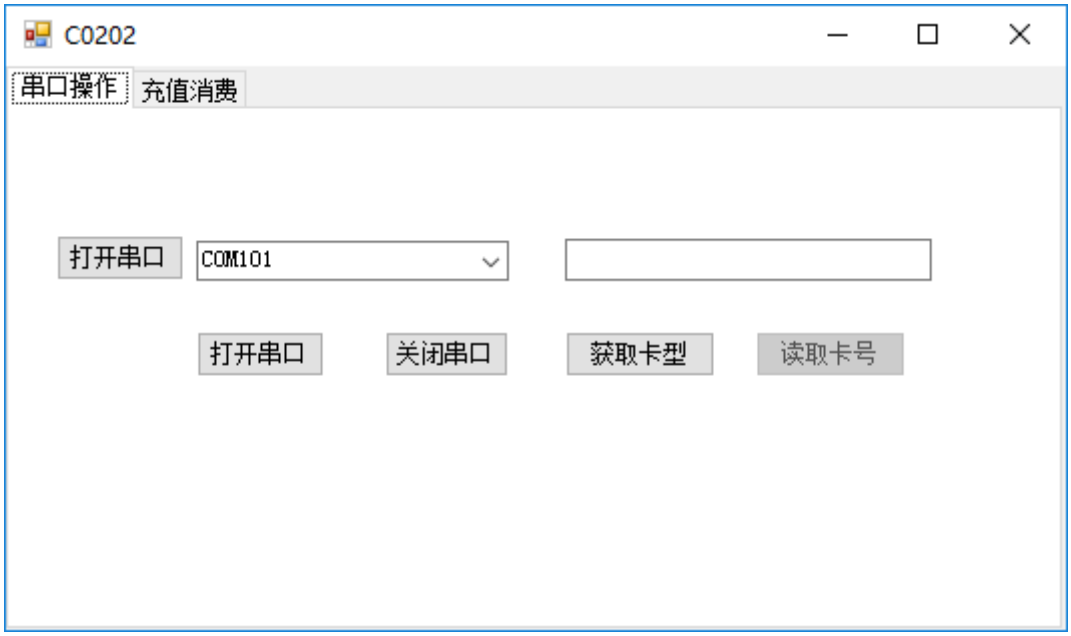
3. 实验原理

使用高频 14443 读写器读取需要充值的高频 14443 标签标签号，然后通过标签号读取数据库中存储的用户基本信息，消费和充值操作是将数据库中该用户存储的余额读取出来，经过消费和充值后，把余额信息重新写入到数据库中。

4. 实验设计

控件名称	控件 Text 属性	控件 Name 属性	功能
Form 窗体	C0202	frmMain	
TabControl 选项卡集合		TabControl1	
TabPage 选项卡	充值消费	Tab2	
Lable 标签	姓名	LblName	
TextBox 文本框		txtName	
Lable 标签	性别	LblSex	
TextBox 文本框		txtSex	
Lable 标签	身份证号	lblIDCard	
TextBox 文本框		txtIDcard	
Lable 标签	余额	lblBalance	
TextBox 文本框		txtBalance	
Lable 标签	充值/消费数额	lblChangeBalance	
TextBox 文本框		txtChangeBalanc	

		e	
Button 按钮	读取信息	btnReadInfo	
Button 按钮	充值	btnTopUp	
Button 按钮	消费	btnExpense	
TabPage 选项卡	串口及寻卡操作	Tab1	
ComboBox 控件		cmbPortNum	获取计算机串口
Button 按钮	打开串口	btnOpenSerialPort	打开串口
Button 按钮	关闭串口	btnCloseSerialPort	关闭串口
Button 按钮	获取卡型	btnGetCardType	获取卡型
Button 按钮	读取卡号	btnReadCard	读取卡号



5. 实验代码解析

定义系统使用变量：

```
        private FR102 Reader;
private string strSnr;
private Byte[] BArraySnr, ByteArrayKeyA;
sqlHelper sh = new sqlHelper();
        KV_ISO14443.FR102 iso14443 = new FR102();
```

打开串口和关闭串口功能

在主窗体 frmMain 的 Load 事件中调用 GetComList()方法使打开窗体时从注册表里获取所有可以使用的端口的端口号并将端口号绑定到端口号下拉框.

```
         public void GetComList()
        {
//RegistryKey keyCom =
Registry.LocalMachine.OpenSubKey("Hardware\\DeviceMap\\SerialComm");
string[] portNames = System.IO.Ports.SerialPort.GetPortNames();
if (portNames != null)
            {
                cmbPortNum.Items.AddRange(portNames);
if (cmbPortNum.Items.Count > 0)
                {
                    cmbPortNum.SelectedIndex = 0;
                }
else
                {
MessageBox.Show(this.Owner, "没有找到可以使用的端口，请检查端口线是否已经连接到电脑上。
```

```
", "提示", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                }

            }
else
            {
MessageBox.Show(this.Owner, "没有找到可以使用的端口，请检查端口线是否已经连接到电脑上。
", "提示", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
        }
```

实例化 TagReader 类，通过对象调用 TagReader 类的 OpenSerialPort()方法打开 cmbPortNum 列表中选定的串口.

```
        private void btnOpenSerialPort_Click(object sender, EventArgs e)
        {
if (Reader.OpenSerialPort(cmbPortNum.Text.Trim()) != 0x00)
            {
MessageBox.Show(String.Format("打开端口:端口打开失败！\n请检查端口{0}是否被其它程序占用
", cmbPortNum.Text.Trim()));
return;
            }
else
            {
if (Reader.TestReader() != 0x00)
                {
MessageBox.Show(String.Format("没有检测到连接到端口{0}的设备，请检查与设备连接的端口！
", cmbPortNum.Text.Trim()));
if (Reader.CloseSerialPort() == 0x00)
                    {
MessageBox.Show(String.Format("关闭端口:端口{0}关闭成功！", cmbPortNum.Text.Trim()));
                    }
return;
                }

if (Reader.RestartReader() != 0x00)
                {
MessageBox.Show("设备启动失败！");
return;
                }
MessageBox.Show(String.Format("打开端口:端口{0}打开成功！", cmbPortNum.Text.Trim()));
return;

            }
        }
```

实例化 ISO14443Reader 类，通过该类的对象调用 CloseSerialPort()方法关闭串口.

```
        private void btnCloseSerialPort_Click(object sender, EventArgs e)
        {
if (FR102.StatusCode.AllDone==iso14443.CloseSerialPort())
            {
MessageBox.Show("串口关闭成功");
return;
            }
else
            {
MessageBox.Show("串口关闭失败");
return;
            }
        }
```

注: ISO14443Reader 类和 TagReader 类都是接口 ISO14443.DLL 接口的中的类,使用之前需要添加该接口的引用,最好把该接口存放在项目的 bin 目录下.

获取卡型和读取卡号功能

通过 FR102 类的对象调用 PcdRequest(Byte req_code, out Byte[] TagType)方法获取卡型

```
private void btnGetCardType_Click(object sender, EventArgs e)
        {
byte[] TagType;
FR102.StatusCode ec = Reader.PcdRequest(0x52, out TagType);
if (ec == FR102.StatusCode.AllDone)
            {
MessageBox.Show("获取成功");
return;

            }
else if (ec == FR102.StatusCode.NoTagErr)
            {
MessageBox.Show("请求失败:请求码0x52，失败原因是在读写器天线场区内无标签");
return;
            }
else
            {
MessageBox.Show("请求失败:请求码0x52，失败原因不详");
return;
            }
        }
```

通过 FR102 类的对象调用 PcdAnticoll(out Byte[] TagNumber)方法读取读写器上的卡的卡号并把获取到的卡号显示在 txtCardID 文本框中

```
private void btnReadCard_Click(object sender, EventArgs e)
        {
```

```
Byte[] data = new Byte[1];
FR102.StatusCode value = Reader.PcdAnticoll2(out data);
if (value == FR102.StatusCode.NoTagErr)
            {

Thread.Sleep(1000); //让程序停止一秒，而后尝试重新搜寻卡片。
            }

            strSnr = "";
            BArraySnr = new Byte[data.Length];
for (Byte i = 0; i < data.Length; i++)
            {
                strSnr = strSnr + String.Format("{0:X2} ", data[i]);
                BArraySnr[i] = data[i];
            }
if (strSnr != "")
            {
MessageBox.Show(String.Format("找到标签，其卡号为:{0}", strSnr));
                txtCardID.Text = strSnr;
            }
if (data.Length < 4)
            {
MessageBox.Show(String.Format("找到标签，其卡号为:{0}，长度不足6字节！", strSnr));
return;
            }

        }
```

注：（1） 在程序中需要给 PcdRequest()方法传一个参数 0X52 即给 ISO14443 读写器发送寻天线区内所有卡命令，寻到卡返回“获取成功”

（2）PcdAnticoll()方法将会返回所寻到的卡的卡号，由于卡号是 Byte 型，所以需要用 Byte 型变量接收，再转化成 string 才能显示在控件中，为了保险起见读取卡号功能放在循环里，循环三次，即获取卡号三次

读取信息功能

单击读取信息按钮通过该按钮的 Click 单击事件调用 SqlHelper 类中的 IsUsing()，方法判断该卡是否可用，如果可用再调用 SqlHelper 类中的 ReadInfo()方法读取用户信息并显示在 TextBox 文本框中，否则将不能读取信息.

```
        private void btnReadInfo_Click(object sender, EventArgs e)
        {
            sh.ReadInfo(strSnr);
            txtName.Text = sh.Name.ToString();
            txtSex.Text = sh.Sex.ToString();
            txtIDCard.Text = sh.IDCard.ToString();
```

```
            txtBalance.Text = sh.Balance.ToString();
        }
```

充值和消费功能

充值:输入充值的数额,单击充值按钮通过该按钮的 Click 单击事件调用 SqlHelper 类中的 IsUsing()方法判断卡是否可用,如果可用则调用 SqlHelper 类中的 ChgangeMoney()方法更新该卡的金额,成功后提示“充值成功!”否则提示“充值失败”.

```
    private void btnTopUp_Click(object sender, EventArgs e)
        {
float i = float.Parse(txtBalance.Text);
float j = float.Parse(txtChangeBalance.Text);
float k = i + j;
            sh.IsUsing(strSnr);        //验证该卡是否处于可用状态
string s = sh.IsSucced();
if (s != "使用中")
            {
MessageBox.Show("该卡已被挂失或被退卡,不能使用!");
            }
else
            {
                sh.ChangeMoney(k.ToString(), strSnr);
int a = sh.IsSuccess();
if (a > 0)
                {
MessageBox.Show("充值成功!");
                }
else
                {
MessageBox.Show("充值失败!");
                }
            }
        }
```

消费:输入消费的数额,单击消费按钮通过该按钮的 Click 单击事件调用 SqlHelper 类中的 IsUsing()方法判断卡是否可用,如果可用则调用 SqlHelper 类中的 ChgangeMoney()方法更新该卡的金额,成功后提示“消费成功!”,否则提示“消费失败!”

```
private void btnExpense_Click(object sender, EventArgs e)
        {
float i = 0;
float j = 0;
float k = 0;
            sh.IsUsing(strSnr);            //验证该卡是否处于可用状态
string s = sh.IsSucced();
if (s != "使用中")
            {
MessageBox.Show("该卡已被挂失或被退卡,不能使用!");
```

```
            }
else
            {

if (txtBalance.Text != "" || txtBalance.Text != "0") //确保有钱消费
                {
                    i = float.Parse(txtBalance.Text);
                    j = float.Parse(txtChangeBalance.Text);
if (i >= j)
                    {
                        k = i - j;
                        sh.ChangeMoney(k.ToString(), strSnr);
int a = sh.IsSuccess();
if (a > 0)
                        {
MessageBox.Show("消费成功!");
                        }
else
                        {
MessageBox.Show("消费失败!");
                        }
                    }
else
                    {
MessageBox.Show("卡上余额不足,无法进行消费!!");
                    }
                }
else
                {
MessageBox.Show("卡上余额不足,无法进行消费!");
                }
            }
        }
```

注： A）读取信息、充值、消费功能只有有成功完成打开串口，获取卡型，读取卡号操作后才能实现．在 CO202.cs 中需要引用命名空间 using TagReader;using PracticeSystem;using System.Threading;在 sqlHelper.cs 中需要引用命名空间 using System.Data 和 usingSystem.Data.sqlClient,否则程序会出现错误.

B） IsUsing()方法通过查询数据库中对应卡号的记录的状态(e_Status)是否为“使用中”来判断卡是否可用.

C)ReadInfo()方法通过 SqlDataAdapter 实例化的对象执行对数据库的查询操作,并把查询结果存储在 DateSet 缓存中,通过 fereach 遍历缓存查询结果并把查询到的信息存储在变量中,通过参数传递设置控件中的值

D）调用 ChgangeMoney()方法时将消费或充值后的结果（余额）传递过去,ChgangeMoney()方法将通过 SqlCommand 对象与数据库进行连接并对数据库执行更新余

额(e_Money)的操作

C0203 指导文档　考勤签到

1. 实验目的

该实验的目的是让学生知道如何使用 ISO14443 卡实现刷卡功能

2. 实验设备

软件：visualstudio2010 及以上版本，MicrosoftSQLServer2005 及以上版本
硬件：5V2A 电源，高频 14443 读写器，高频 14443 标签。串口线

3. 实验原理

学生通过高频 14443 读写器读取高频 14443 标签标签号，然后将卡号和当前的刷卡时间存储到数据库中，完成签到考勤。

4. 实验设计

控件名称	控件 Text 属性	控件 Name 属性	功能
Form 窗体	C0203	frmMain	
ComboBox 控件		cmbPortNum	获取计算机串口
Button 按钮	打开串口	btnOpenSerialPort	打开串口
Button 按钮	关闭串口	btnCloseSerialPort	关闭串口
Button 按钮	获取卡型	btnGetCardType	获取卡型
Button 按钮	读取卡号	btnReadCard	读取卡号
TextBox 文本框		txtCardID	
Button 按钮	刷卡	btnSwipingCard	在数据库中登记卡号和刷卡时间

5. 实验代码解析

定义系统使用变量：

```
        private FR102 Reader;                                   //实例化ISO14443接口中的类
private string strSnr;
private Byte[] BArraySnr, ByteArrayKeyA;
sqlHelper sh = new sqlHelper();
FR102 iso14443 = new FR102();//实例化 ISO14443 接口中的类
```

(1)打开串口和关闭串口功能

A)在主窗体 frmMain 的 Load 事件中调用 GetComList()方法使打开窗体时从注册表里获取所有可以使用的端口的端口号并将端口号绑定到端口号下拉框.

```
public void GetComList()
        {
string[] portNames = System.IO.Ports.SerialPort.GetPortNames();
if (portNames != null)
            {
                cmbPortNum.Items.AddRange(portNames);
if (cmbPortNum.Items.Count > 0)
                {
                    cmbPortNum.SelectedIndex = 0;
                }
else
                {
MessageBox.Show(this.Owner, "没有找到可以使用的端口，请检查端口线是否已经连接到电脑上。
", "提示", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                }

            }
else
            {
MessageBox.Show(this.Owner, "没有找到可以使用的端口，请检查端口线是否已经连接到电脑上。
", "提示", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
        }
```

B)实例化 TagReader 类,通过对象调用 TagReader 类的 OpenSerialPort()方法打开 cmbPortNum 列表中选定的串口.

```
        private void btnOpenSerialPort_Click(object sender, EventArgs e)
        {
if (Reader.OpenSerialPort(cmbPortNum.Text.Trim()) != 0x00)
            {
MessageBox.Show(String.Format("打开端口:端口打开失败！ \n请检查端口{0}是否被其它程序占用
", cmbPortNum.Text.Trim()));
```

```
return;
            }
else
            {
if (Reader.TestReader() != 0x00)
                {
MessageBox.Show(String.Format("没有检测到连接到端口{0}的设备，请检查与设备连接的端口！
", cmbPortNum.Text.Trim()));
if (Reader.CloseSerialPort() == 0x00)
                    {
MessageBox.Show(String.Format("关闭端口:端口{0}关闭成功！", cmbPortNum.Text.Trim()));
                    }
return;
                }

if (Reader.RestartReader() != 0x00)
                {
MessageBox.Show("设备启动失败！");
return;
                }
MessageBox.Show(String.Format("打开端口:端口{0}打开成功！", cmbPortNum.Text.Trim()));
return;

            }
        }
```

C） 实例化 ISO14443Reader 类,通过该类的对象调用 CloseSerialPort()方法关闭串口.

```
        private void btnCloseSerialPort_Click(object sender, EventArgs e)
        {

if (FR102.StatusCode.AllDone== iso14443.CloseSerialPort())
            {
MessageBox.Show("串口关闭成功");
return;
            }
else
            {
MessageBox.Show("串口关闭失败");
return;
            }
        }
```

注：ISO14443Reader 类和 TagReader 类都是接口 ISO14443.DLL 接口的中的类,使用之前需要添加该接口的引用,最好把该接口存放在项目的 bin 目录下.

(2)获取卡型和读取卡号功能

A)通过 FR102 类的对象调用 PcdRequest(Byte req_code, out Byte[] TagType)方法获取卡型

```
        private void btnGetCardType_Click(object sender, EventArgs e)
        {
byte[] TagType;
FR102.StatusCode ec = Reader.PcdRequest(0x52, out TagType);
if (ec == FR102.StatusCode.AllDone)
            {
MessageBox.Show("获取成功");
return;

            }
else if (ec == FR102.StatusCode.NoTagErr)
            {
MessageBox.Show("请求失败:请求码0x52，失败原因是在读写器天线场区内无标签");
return;
            }
else
            {
MessageBox.Show("请求失败:请求码0x52，失败原因不详");
return;
            }
        }
```

B)通过 FR102 类的对象调用 PcdAnticoll(out Byte[] TagNumber)方法读取读写器上的卡的卡号并把获取到的卡号显示在 txtCardID 文本框中

```
        private void btnReadCard_Click(object sender, EventArgs e)
        {
Byte[] data = new Byte[1];
FR102.StatusCode value = Reader.PcdAnticoll2(out data);
if (value == FR102.StatusCode.NoTagErr)
                {
Thread.Sleep(1000); //让程序停止一秒，而后尝试重新搜寻卡片。
                }

                strSnr = "";
                BArraySnr = new Byte[data.Length];
for (Byte i = 0; i < data.Length; i++)
                {
                    strSnr = strSnr + String.Format("{0:X2} ", data[i]);
                    BArraySnr[i] = data[i];
                }
if (strSnr != "")
                {
```

```
MessageBox.Show(String.Format("找到标签，其卡号为:{0}", strSnr));
                    txtCardID.Text = strSnr;
                }
if (data.Length < 4)
                {
MessageBox.Show(String.Format("找到标签，其卡号为:{0}，长度不足6字节！", strSnr));
return;
                }
        }
```

注：（1） 在程序中需要给 PcdRequest()方法传一个参数 0X52 即给 ISO14443 读写器发送寻天线区内所有卡命令，寻到卡返回“获取成功”

（2）PcdAnticoll()方法将会返回所寻到的卡的卡号，由于卡号是 Byte 型，所以需要用 Byte 型变量接收，再转化成 string 才能显示在控件中，为了保险起见读取卡号功能放在循环里，循环三次，即获取卡号三次。

刷卡功能

单击刷卡按钮通过单击事件调用 SqlHelper 类中的 IsUsing()方法判断卡是否可用，如果可用则调用 SqlHelper 中的 insertSignIn()方法将刷卡的卡号和刷卡时间保存在数据库中，成功则提示“刷卡成功”．否则提示“刷卡失败”．

```
        private void btnSwipingCard_Click(object sender, EventArgs e)
        {
            sh.IsUsing(strSnr);
string j = sh.IsSucced();
if (j != "使用中")
            {
MessageBox.Show("该卡已被挂失或被退卡,不能使用!");
            }
else
            {
                sh.insertSignIInCard(strSnr);
int i = sh.IsSuccess();
if (i > 0)
                {
MessageBox.Show("刷卡成功!");
                }
else
                {
MessageBox.Show("刷卡失败!");
                }
            }
        }
```

注：(1)刷卡功能只有成功完成打开串口，获取卡型，读取卡号操作，才能使用．在

C0203.cs 中需要引用命名空间 using TagReader;using PracticeSystem;using System.Threading;在 sqlHelper.cs 中需要引用命名空间 using System.Data 和 usingSystem.Data.sqlClient,否则程序会出现错误.

(2)insertSignIn()方法通过 SqlCommand 对象与数据库进行连接并对数据库执行更新操作将刷卡卡号和刷卡时间保存在数据库中

C0204 指导文档　挂失和取消挂失

1. 实验目的

该实验的目的是让学生知道如何使用 ISO14443 卡如何实现挂失和取消挂失功能

2. 实验设备

软件：visualstudio2010 及以上版本，MicrosoftSQLServer2005 及以上版本
硬件：5V2A 电源，高频 14443 读写器，高频 14443 标签。串口线

3. 实验原理

使用高频 14443 读写器读取高频 14443 标签标签号。当点击挂失和取消挂失时，改变数据库中该卡片对应的状态，

4. 实验设计

控件名称	控件 Text 属性	控件 Name 属性	功能
Form 窗体	C0204	frmMain	
ComboBox 控件		cmbPortNum	获取计算机串口
Button 按钮	打开串口	btnOpenSerialPort	打开串口
Button 按钮	关闭串口	btnCloseSerialPort	关闭串口
Button 按钮	获取卡型	btnGetCardType	获取卡型
Button 按钮	读取卡号	btnReadCard	读取卡号
TextBox 文本框		txtCardID	
Button 按钮	挂失	btnReportTheLoss	
Button 按钮	取消挂失	btnCancelReportTheLoss	

5. 实验代码解析

定义系统使用变量

```
        private FR102 Reader;                                   //实例化ISO14443接口中的类
private string strSnr;
private Byte[] BArraySnr, ByteArrayKeyA;
sqlHelper sh = new sqlHelper();
FR102 iso14443 = new FR102();//实例化 ISO14443 接口中的类
```

(1)打开串口和关闭串口功能

A)在主窗体 frmMain 的 Load 事件中调用 GetComList()方法使打开窗体时从注册表里获取所有可以使用的端口的端口号并将端口号绑定到端口号下拉框.

```
        public void GetComList()
        {
string[] portNames = System.IO.Ports.SerialPort.GetPortNames();
if (portNames != null)
            {
                cmbPortNum.Items.AddRange(portNames);
if (cmbPortNum.Items.Count > 0)
                {
                    cmbPortNum.SelectedIndex = 0;
                }
else
                {
MessageBox.Show(this.Owner, "没有找到可以使用的端口，请检查端口线是否已经连接到电脑上。
", "提示", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                }

            }
else
            {
MessageBox.Show(this.Owner, "没有找到可以使用的端口，请检查端口线是否已经连接到电脑上。
", "提示", MessageBoxButtons.OK, MessageBoxIcon.Warning);
```

```
            }
        }
```

B)实例化 TagReader 类,通过对象调用 TagReader 类的 OpenSerialPort()方法打开 cmbPortNum 列表中选定的串口.

```
        private void btnOpenSerialPort_Click(object sender, EventArgs e)
        {
if (Reader.OpenSerialPort(cmbPortNum.Text.Trim()) != 0x00)
            {
MessageBox.Show(String.Format("打开端口:端口打开失败！\n请检查端口{0}是否被其它程序占用
", cmbPortNum.Text.Trim()));
return;
            }
else
            {
if (Reader.TestReader() != 0x00)
                {
MessageBox.Show(String.Format("没有检测到连接到端口{0}的设备，请检查与设备连接的端口！
", cmbPortNum.Text.Trim()));
if (Reader.CloseSerialPort() == 0x00)
                    {
MessageBox.Show(String.Format("关闭端口:端口{0}关闭成功！", cmbPortNum.Text.Trim()));
                    }
return;
                }

if (Reader.RestartReader() != 0x00)
                {
MessageBox.Show("设备启动失败！");
return;
                }
MessageBox.Show(String.Format("打开端口:端口{0}打开成功！", cmbPortNum.Text.Trim()));
return;

            }
        }
```

C) 实例化 ISO14443Reader 类,通过该类的对象调用 CloseSerialPort()方法关闭串口.

```
        private void btnCloseSerialPort_Click(object sender, EventArgs e)
        {
if (FR102.StatusCode.AllDone== iso14443.CloseSerialPort())
            {
MessageBox.Show("串口关闭成功");
return;
            }
```

```
else
            {
MessageBox.Show("串口关闭失败");
return;
            }
        }
```

注：ISO14443Reader 类和 TagReader 类都是接口 ISO14443.DLL 接口的中的类,使用之前需要添加该接口的引用,最好把该接口存放在项目的 bin 目录下.

(2)获取卡型和读取卡号功能

A)通过 FR102 类的对象调用 PcdRequest(Byte req_code, out Byte[] TagType)方法获取卡型

```
        private void btnGetCardType_Click(object sender, EventArgs e)
        {
byte[] TagType;
FR102.StatusCode ec = Reader.PcdRequest(0x52, out TagType);
if (ec == FR102.StatusCode.AllDone)
            {
MessageBox.Show("获取成功");
return;

            }
else if (ec == FR102.StatusCode.NoTagErr)
            {
MessageBox.Show("请求失败:请求码0x52，失败原因是在读写器天线场区内无标签");
return;
            }
else
            {
MessageBox.Show("请求失败:请求码0x52，失败原因不详");
return;
            }
        }
```

B)通过 FR102 类的对象调用 PcdAnticoll(out Byte[] TagNumber)方法读取读写器上的卡的卡号并把获取到的卡号显示在 txtCardID 文本框中

```
        private void btnReadCard_Click(object sender, EventArgs e)
        {
Byte[] data = new Byte[1];
FR102.StatusCode value = Reader.PcdAnticoll2(out data);
if (value == FR102.StatusCode.NoTagErr)
            {
Thread.Sleep(1000); //让程序停止一秒，而后尝试重新搜寻卡片。
            }
```

```
            strSnr = "";
            BArraySnr = new Byte[data.Length];
for (Byte i = 0; i < data.Length; i++)
            {
                strSnr = strSnr + String.Format("{0:X2} ", data[i]);
                BArraySnr[i] = data[i];
            }
if (strSnr != "")
            {
MessageBox.Show(String.Format("找到标签，其卡号为:{0}", strSnr));
                txtCardID.Text = strSnr;
            }
if (data.Length < 4)
            {
MessageBox.Show(String.Format("找到标签，其卡号为:{0}，长度不足6字节！", strSnr));
return;
            }
        }
```

注：(1) 在程序中需要给 PcdRequest()方法传一个参数 0X52 即给 ISO14443 读写器发送寻天线区内所有卡命令,寻到卡返回“获取成功”

(2)PcdAnticoll()方法将会返回所寻到的卡的卡号,由于卡号是 Byte 型,所以需要用 Byte 型变量接收,再转化成 string 才能显示在控件中，为了保险起见读取卡号功能放在循环里，循环三次，即获取卡号三次

挂失和取消挂失功能

A)单击挂失按钮通过单击事件调用 SqlHelper 类中的 ReportTheLoss()方法将该身份证号对应的卡的状态修改为”挂失”,挂失状态的卡将不能修改金额和刷卡

```
    private void btnReportTheLoss_Click(object sender, EventArgs e)
        {
            sh.ReportTheLoss(strSnr);
int i = sh.IsSuccess();
if (i > 0)
            {
MessageBox.Show("挂失成功!");
            }
else
            {
MessageBox.Show("挂失失败!");
            }
        }
```

B)挂失后如果需要取消挂失则单击取消挂失通过单击事件调用 SqlHelper 类中的

CancelReportTheLoss()方法将该身份证号对应的卡的状态从“挂失”修改为“使用中”，恢复正常使用.

```
private void btnCancelReportTheLoss_Click(object sender, EventArgs e)
        {
            sh.CancelReportTheLoss(strSnr);
int i = sh.IsSuccess();
if (i > 0)
            {
MessageBox.Show("取消挂失成功!");
            }
else
            {
MessageBox.Show("取消挂失失败!");
            }
        }
```

注：(1)挂失和取消挂失功能只有成功完成打开串口，获取卡型，读取卡号操作,才能使用.在 C0204.cs 中需要引用命名空间 using TagReader;using PracticeSystem;using System.Threading;在 sqlHelper.cs 中需要引用命名空间 using System.Data 和 usingSystem.Data.sqlClient,否则程序会出现错误.

(2)CancelReportTheLoss()方法通过 SqlCommand 对象与数据库进行连接并对数据库执行更新操作完成挂失/取消挂失功能

C0205 指导文档　销卡

1. 实验目的

该实验的目的是让学生知道如何使用 ISO14443 卡如何进行退卡

2. 实验设备

软件：visualstudio2010 及以上版本，MicrosoftSQLServer2005 及以上版本
硬件：5V2A 电源，高频 14443 读写器，高频 14443 标签。串口线

3. 实验原理

使用高频 14443 读写器读取高频 14443 标签标签号，根据标签号，删除该用户存储在数据库中的所有信息，以便卡片重复利用。节约成本，提高使用率。

4. 实验设计

控件名称	控件 Text 属性	控件 Name 属性	功能
Form 窗体	C0205	frmMain	
ComboBox 控件		cmbPortNum	获取计算机串口
Button 按钮	打开串口	btnOpenSerialPort	打开串口
Button 按钮	关闭串口	btnCloseSerialPort	关闭串口
Button 按钮	获取卡型	btnGetCardType	获取卡型
Button 按钮	读取卡号	btnReadCard	读取卡号
TextBox 文本框		txtCardID	
Button 按钮	退卡	btnAbsentCard	在数据库中删除卡的记录

5. 实验代码解析

定义系统使用变量：

```
        private FR102 Reader;                                   //实例化ISO14443接口中的类
private string strSnr;
private Byte[] BArraySnr, ByteArrayKeyA;
sqlHelper sh = new sqlHelper();
FR102 iso14443 = new FR102();//实例化 ISO14443 接口中的类
```

(1)打开串口和关闭串口功能

A)在主窗体 frmMain 的 Load 事件中调用 GetComList()方法使打开窗体时从注册表里获取所有可以使用的端口的端口号并将端口号绑定到端口号下拉框.

```
        public void GetComList()
        {
string[] portNames = System.IO.Ports.SerialPort.GetPortNames();
if (portNames != null)
            {
                cmbPortNum.Items.AddRange(portNames);
if (cmbPortNum.Items.Count > 0)
                {
                    cmbPortNum.SelectedIndex = 0;
```

```
                }
else
                {
MessageBox.Show(this.Owner, "没有找到可以使用的端口，请检查端口线是否已经连接到电脑上。
", "提示", MessageBoxButtons.OK, MessageBoxIcon.Warning);
                }

            }
else
            {
MessageBox.Show(this.Owner, "没有找到可以使用的端口，请检查端口线是否已经连接到电脑上。
", "提示", MessageBoxButtons.OK, MessageBoxIcon.Warning);
            }
        }
```

B)实例化 TagReader 类,通过对象调用 TagReader 类的 OpenSerialPort()方法打开 cmbPortNum 列表中选定的串口.

```
private void btnOpenSerialPort_Click(object sender, EventArgs e)
        {
if (Reader.OpenSerialPort(cmbPortNum.Text.Trim()) != 0x00)
            {
MessageBox.Show(String.Format("打开端口:端口打开失败！\n请检查端口{0}是否被其它程序占用
", cmbPortNum.Text.Trim()));
return;
            }
else
            {
if (Reader.TestReader() != 0x00)
                {
MessageBox.Show(String.Format("没有检测到连接到端口{0}的设备，请检查与设备连接的端口！
", cmbPortNum.Text.Trim()));
if (Reader.CloseSerialPort() == 0x00)
                    {
MessageBox.Show(String.Format("关闭端口:端口{0}关闭成功！", cmbPortNum.Text.Trim()));
                    }
return;
                }

if (Reader.RestartReader() != 0x00)
                {
MessageBox.Show("设备启动失败！");
return;
                }
MessageBox.Show(String.Format("打开端口:端口{0}打开成功！", cmbPortNum.Text.Trim()));
return;
```

```
            }
        }
```

C) 实例化 ISO14443Reader 类,通过该类的对象调用 CloseSerialPort()方法关闭串口.

```
        private void btnCloseSerialPort_Click(object sender, EventArgs e)
        {
if (FR102.StatusCode.AllDone== iso14443.CloseSerialPort())
            {
MessageBox.Show("串口关闭成功");
return;
            }
else
            {
MessageBox.Show("串口关闭失败");
return;
            }
        }
```

注：ISO14443Reader 类和 TagReader 类都是接口 ISO14443.DLL 接口的中的类,使用之前需要添加该接口的引用,最好把该接口存放在项目的 bin 目录下.

(2)获取卡型和读取卡号功能

A)通过 FR102 类的对象调用 PcdRequest(Byte req_code, out Byte[] TagType)方法获取卡型

```
        private void btnGetCardType_Click(object sender, EventArgs e)
        {
byte[] TagType;
FR102.StatusCode ec = Reader.PcdRequest(0x52, out TagType);
if (ec == FR102.StatusCode.AllDone)
            {
MessageBox.Show("获取成功");
return;

            }
else if (ec == FR102.StatusCode.NoTagErr)
            {
MessageBox.Show("请求失败:请求码0x52，失败原因是在读写器天线场区内无标签");
return;
            }
else
            {
MessageBox.Show("请求失败:请求码0x52，失败原因不详");
return;
            }
```

```
        }
```

B)通过 FR102 类的对象调用 PcdAnticoll(out Byte[] TagNumber)方法读取读写器上的卡的卡号并把获取到的卡号显示在 txtCardID 文本框中

```
        private void btnReadCard_Click(object sender, EventArgs e)
        {
Byte[] data = newByte[1];
FR102.StatusCode value = Reader.PcdAnticoll2(out data);
if (value == FR102.StatusCode.NoTagErr)
            {
Thread.Sleep(1000); //让程序停止一秒，而后尝试重新搜寻卡片。
            }

            strSnr = "";
            BArraySnr = new Byte[data.Length];
for (Byte i = 0; i < data.Length; i++)
            {
                strSnr = strSnr + String.Format("{0:X2} ", data[i]);
                BArraySnr[i] = data[i];
            }
if (strSnr != "")
            {
MessageBox.Show(String.Format("找到标签，其卡号为:{0}", strSnr));
                txtCardID.Text = strSnr;
            }
if (data.Length < 4)
            {
MessageBox.Show(String.Format("找到标签，其卡号为:{0}，长度不足6字节！", strSnr));
return;
            }
        }
```

注：（1） 在程序中需要给 PcdRequest()方法传一个参数 0X52 即给 ISO14443 读写器发送寻天线区内所有卡命令，寻到卡返回“获取成功”

（2）PcdAnticoll()方法将会返回所寻到的卡的卡号，由于卡号是 Byte 型，所以需要用 Byte 型变量接收，再转化成 string 才能显示在控件中，为了保险起见读取卡号功能放在循环里，循环三次，即获取卡号三次

退卡功能

单击退卡通过单击事件调用 SqlHelper 类中的 AbsentCard()方法删除数据库中所检测到的卡的记录

```
        private void btnAbsentCard_Click(object sender, EventArgs e)
        {
            sh.AbsentCard(strSnr);
int i = sh.IsSuccess();
```

```
if (i > 0)
            {
MessageBox.Show("退卡成功!");
            }
else
            {
MessageBox.Show("退卡失败!");
            }
        }
```

注：(1)退卡功能只有成功完成打开串口，获取卡型，读取卡号操作,才能使用.在C0205.cs 中需要引用命名空间 using TagReader;using PracticeSystem;using System.Threading;在 sqlHelper.cs 中需要引用命名空间 using System.Data 和 usingSystem.Data.sqlClient,否则程序会出现错误.

(2)AbsentCard()通过 SqlCommand 对象与数据库进行连接并对数据库执行删除操作,将数据库中所对应卡号的记录删除