



《RFID原理及应用》

实 验 教 案

任课老师： 李 清 波

授课对象： 级物联网工程

授课学期： 20 学年 期

总 课 时： 24 课时

202 年 月

## 实验1 RFID程序开发基础实验（一）

**一、 实验目的**

理解并掌握RFID程序开发语言C#的下列基础控件：

Button控件, CheckBox控件, ContextMenuStrip控件, datetimepicker控件,

FlowLayoutPanel控件, ListView控件, MenuStrip控件, NumericUpDown控件,

**二、 实验内容**

1. 创建一个窗体应用程序。

2. 在窗体中添加以上控件。

3. Button 类

表示 Windows 按钮控件。

所属命名空间:System.Windows.Forms

关闭串口Button控件：Windows窗体Button 控件允许用户通过单击来执行操作。Button 控件既可以显示文本，又可以显示图像。当该按钮被单击时，它看起来像是被按下，然后被释放。

4. CheckBox类

表示 Windows CheckBox控件。

所属命名空间:System.Windows.Forms

CheckBox控件：复选框，也叫做CheckBox，是一种基础控件。.NET的工具箱里包含这个控件，它可以通过其属性和方法完成复选的操作。为了完成更多复杂的需求，也出了第三方控件的复选框。只需要将其dll添加到工具箱里，就可以使用更多功能的复选框控件。

5. ContextMenuStrip类

表示快捷菜单。

所属命名空间:System.Windows.Forms

ContextMenuStrip控件：要显示弹出菜单，或在用户右击鼠标时显示一个菜单，就应使用ContextMenuStrip类。与MenuStrip一样，ContextMenuStrip也是ToolStripMenuItems对象的容器，但它派生于ToolStripDropDownMenu。ContextMenu的创建与MenuStrip相同，也是添加ToolStripMenuItems，定义每一项的Click事件，执行某个任务。弹出菜单应赋予特定的控件，为此，要设置控件的ContextMenuStrip属性。在用户右击该控件时，就显示该菜单。

6. datetimepicker类

表示一个 Windows 控件，该控件用来让用户选择日期和时间并以指定的格式显示此日期和时间。

所属命名空间:System.Windows.Forms

datetimepicker控件：DateTimePicker控件，控件一般用于让用户可以从日期列表中选择单个值。运行时，单击控件边上的下拉箭头，会显示为两个部分：一个下拉列表，一个用于选择日期的。

7. FlowLayoutPanel类

表示一个面板，它可以进行动态水平或垂直布局其内容。

所属命名空间:System.Windows.Forms

FlowLayoutPanel控件：FlowLayoutPanel 控件在水平或垂直流方向排列其内容。可以将该控件的内容从一行换行至下一行，或者从一列换至下一列。还可以选择剪裁内容而不是换行。

可以通过设置 FlowDirection 属性的值来指定流方向。FlowLayoutPanel 控件在从右向左 (RTL) 的布局中正确地反转它的流方向。还可以通过设置 WrapContents 属性的值来指定是换行还是剪裁 FlowLayoutPanel 控件的内容。

将 AutoSize 属性设置为 true 时，FlowLayoutPanel 控件自动调整大小以容纳其内容。它还向其子控件提供了 FlowBreak 属性。将 FlowBreak 属性的值设置为 true 会使 FlowLayoutPanel 控件停止在当前流方向布局控件并换到下一行或下一列。

任何 Windows 窗体控件都可以是 FlowLayoutPanel 控件的子控件，包括 FlowLayoutPanel 的其他实例。利用此功能，可以在运行时构造适应窗体尺寸的复杂布局。

8. ListView类

表示一个面板，它可以进行动态水平或垂直布局其内容。

所属命名空间:System.Web.UI.WebControls

ListView控件： ListView 控件可使用四种不同视图显示项目。通过此控件，可将项目组成带有或不带有列标头的列，并显示伴随的图标和文本。 可使用 ListView 控件将称作 ListItem 对象的列表条目组织成下列四种不同的视图之一：1.大（标准）图标2.小图标3.列表4.报表 View 属性决定在列表中控件使用何种视图显示项目。还可用 LabelWrap 属性控制列表中与项目关联的标签是否可换行显示。另外，还可管理列表中项目的排序方法和选定项目的外观。

9. MenuStrip类

为窗体提供菜单系统。

所属命名空间:System.Windows.Forms

MenuStrip控件

在建立菜单系统时，要给MenuStrip添加ToolStripMenu对象。这可以在代码中完成，也可以在Visual Studio的设计器中进行。把一个MenuStrip控件拖放到设计器的一个窗体中，MenuStrip就允许直接在菜单项上输入菜单文本[1] 。

MenuStrip控件只有两个额外的属性。GripStyle使用ToolStripGripStyle枚举把栅格设置为可见或隐藏。

MdiWindowListItem属性提取或返回ToolStripMenuItem。这个ToolStripMenuItem是在MDI应用程序中显示所有已打开窗口的菜单。

10. NumericUpDowm类

表示显示数值的 Windows 数字显示框（也称作 up-down 控件）。

所属命名空间:System.Windows.Forms

NumericUpDowm控件: Windows 窗体 NumericUpDown 控件看起来像是一个文本框与一对箭头的组合，用户可以单击箭头来调整值。 该控件显示并设置选择列表中的单个数值。 用户可以通过单击向上和向下按钮、按向上键和向下键或键入一个数字来增大和减小数字。 单击向上键时，值沿最大值方向增加；单击向下键时，位置沿最小值方向移动。 说明此类控件很有用的一个示例是音乐播放器上的音量控件。 某些 Windows 控制面板应用程序中使用了数值 up-down 控件。

**三、 操作步骤**

（1）启动visualstudio，文件→新建→项目。

（2）选择Visual C#→Windows窗体应用程序，输入名称→选择存储路径。

（3）界面设计及控件属性

在工具箱中找到相应的控件按钮，双击或者拖拽都可以添加控件到窗体中。

**四、 思考与练习**

**五、实验结果及后记**

## 实验2 RFID程序开发基础实验（二）

**一、实验目的**

理解并掌握RFID程序开发语言C#的下列基础控件：

PictureBox控件, RadioButton控件, SplitContainer控件, TabContainer控件,

Timer控件, ToolStrip控件, TreeView控件, ComboBox控件, SeriaPort获取端口号

**二、 实验内容**

1. PictureBox类

表示用于显示图像的 Windows 图片框控件。所属命名空间:System.Windows.Forms

PictureBox 控件可以显示来自位图、图标或者元文件，以及来自增强的元文件、JPEG 或 GIF 文件的图形。如果控件不足以显示整幅图象，则裁剪图象以适应控件的大小。

2. RadioButton类

表示一个按钮，可以选择，但不是会清除，由用户。 IsChecked 属性 RadioButton 可以通过单击它，设置，但它可以只以编程方式清除。

所属命名空间:System.Windows.Forms

3. SplitContainer类

表示一个由可移动条组成的控件，该可移动条将容器的显示区域分成两个大小可调的面板。

所属命名空间:System.Windows.Forms

可以将 Windows 窗体 SplitContainer 控件看作是一个复合体，它是由一个可移动的拆分条分隔的两个面板。当鼠标指针悬停在该拆分条上时，指针将相应地改灰度校正状以显示该拆分条是可移动的。

使用 SplitContainer 控件可以创建复合的用户界面（通常，在一个面板中的选择决定了在另一个面板中显示哪些对象）。这种排列对于显示和浏览信息非常有用。拥有两个面板使您可以聚合不同区域中的信息，并且用户可以轻松地使用拆分条（也称为“拆分器”）调整面板的大小。

4. TabControl类

表示包含多个共享相同的空间在屏幕上的项的控件。所属命名空间:System.Windows.Forms

在 Windows 应用程序中，选项卡用于将相关的控件集中在一起，放在一个页面中用以显示多种综合信息。选项卡控件通常用于显示多个选项卡，其中每个选项卡均可包含图片和其他控件。选项卡相当于多窗体控件，可以通过设置多页面方式容纳其他控件。

5. Timer类

实现按用户定义的时间间隔引发事件的计时器。 此计时器最宜用于 Windows 窗体应用程序中，并且必须在窗口中使用。 所属命名空间:System.Windows.Forms

Timer 控件用于背景进程中，它是不可见的。

对于 Timer 控件以外的其它控件的多重选择，不能设置 Timer 的 Enabled 属性。

在运行于 Windows 95 或 Windows NT 下的 Visual Basic 5.0 中可以有多个活动的定时器控件，对此，实际上并没有什么限制。

补充:Timer控件通俗来说就是计时器,这是一个不可视控件.它的重要属性有Interval,Enabled.

它的Tick事件指的是每经过Interval属性指定的时间间隔时发生一次.

6. ToolStrip类

为 Windows 工具栏对象提供容器。所属命名空间:System.Windows.Forms

使用 ToolStrip 及其关联的类，可以创建具有 Windows XP、Microsoft Office、Microsoft Internet Explorer 或自定义的外观和行为的工具栏及其他用户界面元素。这些元素支持溢出及运行时项重新排序。ToolStrip 控件提供丰富的设计时体验，包括就地激活和编辑、自定义布局、漂浮（即工具栏共享水平或垂直空间的能力）。

尽管 ToolStrip 替换了早期版本的控件并添加了功能，但是仍可以在需要时选择保留 ToolBar 以备向后兼容和将来使用。

使用 ToolStrip 控件可以：

创建易于自定义的常用工具栏，让这些工具栏支持高级用户界面和布局功能，如停靠、漂浮、带文本和图像的按钮、下拉按钮和控件、“溢出”按钮和 ToolStrip 项的运行时重新排序。

支持操作系统的典型外观和行为。

对所有容器和包含的项进行事件的一致性处理，处理方式与其他控件的事件相同。

将项从一个 ToolStrip 拖到另一个 ToolStrip 内。

使用 ToolStripDropDown 中的高级布局创建下拉控件及用户界面类型编辑器。

通过使用 ToolStripControlHost 类来使用 ToolStrip 中的其他控件，并为它们获取 ToolStrip 功能。

通过使用 ToolStripRenderer、ToolStripProfessionalRenderer 和 ToolStripManager 以及 ToolStripRenderMode 枚举和 ToolStripManagerRenderMode 枚举，可以扩展此功能并修改外观和行为。

7. TreeView类

显示标记的每个表示项的分层集合 TreeNode。所属命名空间:System.Windows.Forms

TreeView控件用来显示信息的分级视图，如同Windows里的资源管理器的目录。TreeView控件中的各项信息都有一个与之相关的Node对象。TreeView显示Node对象的分层目录结构，每个Node对象均由一个Label对象和其相关的位图组成。在建立TreeView控件后，我们可以展开和折叠、显示或隐藏其中的节点。TreeView控件一般用来显示文件和目录结构、文档中的类层次、索引中的层次和其他具有分层目录结构的信息。

**8. ComboBox类**

表示 Windows 组合框控件。所属命名空间:System.Windows.Forms

Windows 窗体 ComboBox 控件用于在下拉组合框中显示数据。默认情况下，ComboBox 控件分两个部分显示：顶部是一个允许用户键入列表项的文本框。第二部分是一个列表框，它显示一个项列表，用户可从中选择一项。有关组合框的其他样式的更多信息，请参见何时使用 Windows 窗体 ComboBox 而非 ListBox。

SelectedIndex 属性返回一个整数值，该值与选择的列表项相对应。通过在代码中更改 SelectedIndex 值，可以编程方式更改选择项；列表中的相应项将出现在组合框的文本框部分。如果未选择任何项，则 SelectedIndex 值为 -1。如果选择列表中的第一项，则 SelectedIndex 值为 0。SelectedItem 属性与 SelectedIndex 类似，但它返回项本身，通常是一个字符串值。Count 属性反映列表的项数，由于 SelectedIndex 是从零开始的，所以 Count 属性的值通常比 SelectedIndex 的最大可能值大一。

若要在 ComboBox 控件中添加或删除项，请使用 Add、Insert、Clear 或 Remove 方法。或者，可以在设计器中使用 Items 属性向列表添加项。

**9. SerialPort类**

serial port，即串行端口，现在大多数硬件设备均采用串口技术与计算机相连，因此串口的应用程序开发越来越普遍。例如，在计算机没有安装网卡的情况下，将本机上的一些信息数据 传输到另一台计算机上，那么利用串口通信就可以实现。

所属命名空间: System.IO.Ports

**三、 操作步骤**

（1）启动visualstudio，文件→新建→项目。

（2）选择Visual C#→Windows窗体应用程序，输入名称→选择存储路径。

（3）界面设计及控件属性

在工具箱中找到相应的控件按钮，双击或者拖拽都可以添加控件到窗体中。

**四、 思考与练习**

**五、实验结果及后记**

## 实验3 RFID程序开发基础实验（三）

**一、实验目的**

1. 理解并掌握数据库的连接和数据的编辑
2. 理解并掌握C#的IO操作
3. 理解并掌握C#中XML的使用
4. 理解并应用Socket类通信

**二、 实验内容**

1. System.Data.SqlClient 命名空间是 SQL Server 的 .NET Framework 数据提供程序。

SQL Server 的 .NET Framework 数据提供程序描述了一个类集合，这个类集合用于访问托管空间中的 SQL Server 数据库。使用 SqlDataAdapter 可以填充驻留在内存中的 DataSet，该数据集可用于查询和更新数据库。

2. System.Data.SqlClient 命名空间是 SQL Server 的 .NET Framework 数据提供程序。

SQL Server 的 .NET Framework 数据提供程序描述了一个类集合，这个类集合用于访问托管空间中的 SQL Server 数据库。使用 SqlDataAdapter 可以填充驻留在内存中的 DataSet，该数据集可用于查询和更新数据库。

3.System.IO 命名空间

System.IO 命名空间包含允许读写文件和数据流的类型以及提供基本文件和目录支持的类型。

在.NET中Stream 是所有流的抽象基类。流是字节序列的抽象概念，或者说是计算机在处理文件或数据时产生的二进制序列。例如文件、输入/输出设备、内部进程通信管道或者 TCP/IP 套接字。Stream 类及其派生类提供这些不同类型的输入和输出的一般视图，使程序员不必了解操作系统和基础设备的具体细节。简单的说流提供了不同介质之间的数据交互功能。

2. System.Threading命名空间：

Thread 类

创建并控制线程，设置其优先级并获取其状态。

System.Net 命名空间

System.Net 命名空间包含具有以下功能的类：提供适用于许多网络协议的简单编程接口，以编程方式访问和更新 System.Net 命名空间的配置设置，定义 Web 资源的缓存策略，撰写和发送电子邮件，代表多用途 Internet 邮件交换 (MIME) 标头，访问网络流量数据和网络地址信息，以及访问对等网络功能。 另外，其他子命名空间还能让你以受控方式实现 Windows 套接字 (Winsock) 接口，能让你访问网络流以实现主机之间的安全通信。

3. System.Net.Sockets 命名空间

为需要严密控制网络访问的开发人员提供了 Windows Sockets (Winsock) 接口的托管实现。

TcpClient、TcpListener 和 UdpClient 类封装有关创建到 Internet 的 TCP 和 UDP 连接的详细信息

线程，有时被称为轻量级进程(Lightweight Process，LWP），是程序执行流的最小单元。一个标准的线程由线程ID，当前指令指针(PC），寄存器集合和堆栈组成。另外，线程是进程中的一个实体，是被系统独立调度和分派的基本单位，线程自己不拥有系统资源，只拥有一点儿在运行中必不可少的资源，但它可与同属一个进程的其它线程共享进程所拥有的全部资源。一个线程可以创建和撤消另一个线程，同一进程中的多个线程之间可以并发执行。由于线程之间的相互制约，致使线程在运行中呈现出间断性。线程也有就绪、阻塞和运行三种基本状态。就绪状态是指线程具备运行的所有条件，逻辑上可以运行，在等待处理机；运行状态是指线程占有处理机正在运行；阻塞状态是指线程在等待一个事件（如某个信号量），逻辑上不可执行。每一个程序都至少有一个线程，若程序只有一个线程，那就是程序本身。

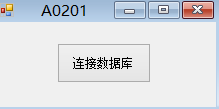
线程是程序中一个单一的顺序控制流程。进程内一个相对独立的、可调度的执行单元，是系统独立调度和分派CPU的基本单位指运行中的程序的调度单位。在单个程序中同时运行多个线程完成不同的工作，称为多线程。

**三、 操作步骤**

1、启动visualstudio，文件→新建→项目。

2、选择VisualC#→Windows窗体应用程序，输入名称→选择存储路径。

3、连接数据--界面设计及控件属性



|  |  |  |  |
| --- | --- | --- | --- |
| 控件名称 | 控件Text属性 | 控件Name属性 | 功能 |
| Form窗体 | A0201 | frmMain |  |
| Button按钮 | 连接数据库 | btnDatabaseConnection |  |

4、相关事件

添加连接数据库按钮单击事件btnDatebaseConnection\_Click

5、常见错误解析：

1）当我们没有创建数据库的情况下，程序会报异常，并且把异常信息显示到异常提示框中，学生可根据异常提示框中的信息定位到错误类型，

2）当我们使用sql账号登录，用户登录失败，是因为账号密码错误，请确认账号密码后重新登录密码。

注意：只要把错误信息复制到百度上，大部分异常都能在百度上找到解决办法

6、实验代码解析

注：在连接数据库之前必须先创建数据库文件。

添加命名空间

using System.Data.SqlClient;

连接数据库按钮单击事件。

privatevoid btnDatebaseConnection\_Click(object sender, EventArgs e)

{

//建立和数据库服务器的连接

SqlConnection conn = newSqlConnection("server = .;database = db\_UserInfo; uid=sa;pwd=123;");

//其中server = .;中的"."代表本地服务器,db\_BookManage代表需要连接的数据库名,Trusted\_Connection = true;中的"true'代表打开数据库连接,为"false"则表示关闭.

conn.Open(); //打开数据库连接

MessageBox.Show("连接成功");

}

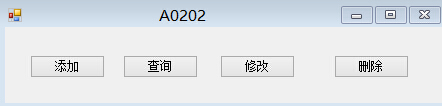
**编辑数据操作：**

1、启动visualstudio，文件→新建→项目。

2、选择VisualC#→Windows窗体应用程序，输入名称→选择存储路径。

3、添加类：右击项目→添加→类。选择VisualC#项→类，填写类名sqlhelper.cs。

4、界面设计及控件属性



|  |  |  |  |
| --- | --- | --- | --- |
| 控件名称 | 控件Text属性 | 控件Name属性 | 功能 |
| Form1窗体 | A0202 | frmMain |  |
| Button按钮 | 添加 | btnInsert | 往数据库中添加信息 |
| Button按钮 | 查询 | btnSelect | 查询数据库中的信息 |
| Button按钮 | 修改 | btnChange | 修改数据库中的信息 |
| Button按钮 | 删除 | btnDelete | 删除数据库中的信息 |

5、相关事件

添加添加按钮单击事件btnInsert\_Click

添加查询按钮单击事件btnSelect\_Click

添加修改按钮单击事件btnChange\_Click

添加删除按钮单击事件btnDelete\_Click

6. 实验代码解析

**IO操作：**

界面设计及控件属性

控件拖拽详见实验编号A01XX系列

|  |  |  |  |
| --- | --- | --- | --- |
| 控件名称 | Text属性 | Name属性 | 功能 |
| Form窗体 | A0203 | FrmMain |  |
| Label控件 | 文件： | label1 |  |
| TextBox控件 |  | textBox1 | 读取或写入文件的路径 |
| RichTextBox控件 | 参考：https://www.cnblogs.com/arxive/p/5725570.html | rtxtbox\_Content | 读取或写入文件的内容 |
| GroupBox控件 | 文件内容： | groupBox1 |  |
| Button | 读取文本 |  |  |
| 读取图片 |  |  |
| 写入文本 |  |  |



4、相关事件

添加读取文本按钮单击事件btn\_Read\_Click

添加写入文本按钮单击事件btn\_Write\_Click

添加读取图片按钮单击事件btn\_ReadImage\_Click

**四、 思考与练习**

**五、实验结果及后记**

## 实验4 低频125K RFID系统实验

**一、实验目的**

1.京胜实验台介绍

2.ISO 18000-2 125KHz串口操作

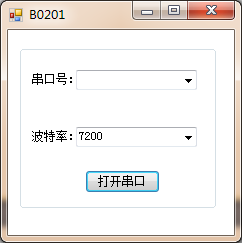
2. ISO 18000-2 125KHz数据读取操作

**二、 实验内容**

ISO/IEC 18000-2 信息技术－基于单品管理的射频识别－适用于中频125～134KHz，规定在标签和读写器之间通信的物理接口，读写器应具有与Type A(FDX)和Type B(HDX)标签通信的能力；规定协议和指令再加上多标签通信的防碰撞方法。

低频125K是通过串口线读取读写器串口缓存区的数据，然后经过进制转换获得十六进制数据，通过四则运算来获取韦根数据。

1、打开串口



|  |  |  |  |
| --- | --- | --- | --- |
| 控件名称 | Text属性 | Name属性 | 功能 |
| From1窗体 | B0201 |  |  |
| Label控件 | 串口号： |  |  |
| Label控件 | 波特率： |  |  |
| comboBox控件 |  | cmbPortID | 存放读取到的串口号 |
| comboBox控件 |  | cmbBaudRate | 存放波特率 |
| Button控件 | 打开串口： | btnOpenClose |  |
| GroupBox容器 |  |  |  |

2、读取数据



|  |  |  |  |
| --- | --- | --- | --- |
| 控件名称 | Text属性 | Name属性 | 功能 |
| From1窗体 | B0202 |  |  |
| Label标签 | 串口号： |  |  |
| Label标签 | 波特率： |  |  |
| ComboBox控件 |  | cmbPortID | 存放读取到的串口号 |
| ComboBox控件 |  | cmbBaudRate | 存放波特率 |
| Button按钮 | 打开串口 | btnOpenClose |  |
| Button按钮 | 开始读取 | btnStartStopRead | 让读写器读取数据 |
| GroupBox容器 | 串口操作 |  |  |
| GroupBox容器 | 十六进制数据 |  |  |
| GroupBox容器 | 韦根数据 |  |  |
| ListBox控件 |  | LstHexData | 存放返回的十六进制数据 |
| ListBox控件 |  | lstWGData | 存放返回的韦根数据 |

**三、 操作步骤**

1、打开串口/关闭串口

A)获取计算机可用串行资源，将获取到的串口号放置在数组中：

string[] PortName = System.IO.Ports.SerialPort.GetPortNames();

B)引用System.IO.Ports中的Open（）方法打开串口，利用IsOpen方法判断串口是否打开，如果打开，则提示串口打开成功，把Bool类型变量置为true，

C)利用变量判断串口是否打开，如果串口当前状态为打开，则提示关闭串口成功，把Bool类型变量置false。

2、读取数据

利用线程循环检测读写器是否检测到卡，如果检测到则显示到ListBox控件中显示出来。

检测方法：

Receive\_125K()能把检测到的十六进制数据和韦根数据分别存入数组，把十六进制数据通过AddHexData（）方法添加到LstHexData中显示，把韦根数据通过AddWGData（）方法添加到lstWGData中显示。

把十六进制数据填充进LstHexData控件中方法AddHexData()

把韦根数据填充进lstWGData控件中AddWGData()

**四、 思考与练习**

**五、实验结果及后记**

## 实验5 高频RFID系统ISO15693实验（一）

**一、实验目的**

1. 串口操作
2. 寻卡操作
3. 读/写数据操作
4. 多路复用器操作

**二、 实验内容**

1 OpenSerialPort方法

函数原型：public Byte OpenSerialPort(String portName, Int32 BaudRate)

public Byte OpenSerialPort(String portName)

描述：打开串口，提供了两个重载。

参数：portName为串口号，如"COM1"；在未指定波特率时，采用设备默认配置：波特率115200；数据位8；停止位1；奇偶校验无。

返回值：打开成功返回0x00，打开失败返回串口错误0x01。

示例：String PortName = "COM1";

Int32 BaudRate = 115200;

Byte value=reader.OpenSerialPort(PortName, BaudRate);

if (value == 0x00) { //串口打开成功 }

else{ //串口打开失败 }

注：示例中的reader为Reader类的一个实例，下同。

2 CloseSerialPort方法

函数原型：public Byte CloseSerialPort()

描述：关闭串口；

参数：无.

返回值：关闭成功返回0x00，关闭失败返回串口错误0x01；

示例：Byte value = reader.CloseSerialPort();

3 Inventory方法

函数原型：public Byte Inventory(ModulateMethod mm, InventoryModel im, ref Int32 TagCount, ref String[] TagNumber)

描述：寻卡，获取场区内卡片的卡号；

参数1：mm，枚举类型ModulateMethod，用于选择调制方式，ASK或FSK；

参数2：im，枚举类型InventoryModel，用于选择寻卡方式，寻单卡Single或寻多卡Multiple；

参数3：TagCount，整形，该参数为引用参数，用于返回读取到的卡片数量；特别注意，在调用该函数前，一定先对该参数进行申明和初始化，以便系统为之分配内存空间，在函数运行时存放返回值。

参数4：TagNumber，字符串数组类型，该参数为引用参数，用于返回读取到的所有卡片的卡号，正常情况下，该数组的长度等于TagCount；特别注意，在调用该函数前，一定先对该参数进行申明和初始化，以便系统为之分配内存空间，在函数运行时存放返回值。

返回值：寻卡成功返回0x00，启动失败返回其它；

示例：ModulateMethod mm=ModulateMethod.ASK;

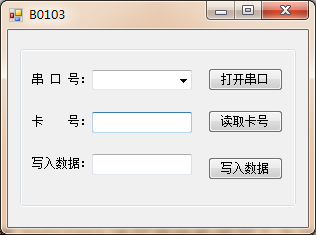
InventoryModel im=InventoryModel.Multiple;

Int32 TagCount=0;

String[] TagNumber=new String[1];

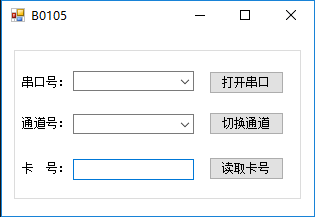
Byte value = reader.Inventory(mm, im, ref TagCount, ref TagNumber);

（一）、打开串口，读取卡号和写入数据操作



|  |  |  |  |
| --- | --- | --- | --- |
| 控件名称 | Text属性 | Name属性 | 功能 |
| Form1窗体 | B0103 |  |  |
| Label标签 | 串口号： |  |  |
| Lable标签 | 卡号： |  |  |
| Lable标签 | 写入数据： |  |  |
| comboBox控件 |  | cmbPortID | 存放读取到的串口号 |
| TextBox控件 |  | txtCardID | 存放读取到的卡号 |
| TextBox控件 |  | txtReadData | 要写入的数据 |
| Button控件 | 读取卡号 | btnReadCardID |  |
| Button控件 | 打开串口 | btnOpenClosePort |  |
| Button控件 | 写入数据 | btnWriteData |  |
| GroupBox容器 |  |  |  |

（二）、切换通道操作



|  |  |  |  |
| --- | --- | --- | --- |
| 控件名称 | Text属性 | Name属性 | 功能 |
| Form1窗体 | B0105 |  |  |
| Label标签 | 串口号： |  |  |
| Lable标签 | 卡号： |  |  |
| Lable标签 | 通道号： |  |  |
| comboBox控件 |  | cmbPortID | 存放读取到的串口号 |
| comboBox控件 |  | cmbGalleryID | 存放多路复用器的通道号 |
| TextBox控件 |  | txtCardID | 存放读取到的卡号 |
| Button控件 | 切换通道 | btnSwitchoverGallery |  |
| Button控件 | 打开串口 | btnOpenClosePort |  |
| Button控件 | 读取卡号 | btnReadCardID |  |
| GroupBox容器 |  |  |  |

**三、 操作步骤**

1. **窗体启动事件**

在窗体启动事件中调用System.IO.Ports.SerialPort.GetPortNames方法获取当前计算机串口号数组，使用循环将获取到的所有串口号添加到comPortID的项列表中。

1. **打开串口按钮事件**

利用KV\_ISO15693.Reader类中的OpenSerialPort方法打开串口，判断返回值，如果返回值为0x00则打开串口成功，否则打开串口失败。

利用KV\_ISO15693.Reader类中的CloseSerialPort方法关闭串口，判断返回值，如果返回值为0x00则关闭串口成功，否则关闭串口失败。

在打开串口成功后将“打开串口”按钮的Text属性修改为“关闭串口”，再次单击按钮就是执行关闭串口的代码。在关闭串口成功后将“关闭串口”按钮的Text属性修改为“打开串口”，再次单击按钮就是执行打开串口的代码。

1. **读取卡号按钮事件**

利用KV\_ISO15693.Reader类中的Inventory方法获取卡号。判断返回值是否为0x00，是则把卡号显示到文本框中，否则提示寻卡失败。

1. **写入数据按钮事件**

利用KV\_ISO15693.Reader类中的WriteSingleBlock方法将数据写入卡片。判断返回值是否为0x00，是则写卡成功，否则写卡失败。

因为WriteSingleBlock方法只能写单块，并且实验所用的RFID卡片单块存储长度为4个字节，所以B0103中一次只能写入4个字节长度的数据。

1. **切换通道按钮事件**

调用KV\_ISO15693.Reader类中的SetAntanna方法切换多路复用器的通道，判断返回值，如果返回值为0x00则切换通道成功，否则切换通道失败。

**四、 思考与练习**

**五、实验结果及后记**

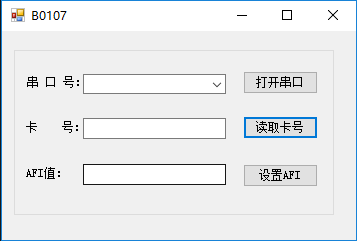
## 实验6 高频RFID系统ISO15693实验（二）

**一、实验目的**

1. 设置应用族标识
2. 设置数据存储格式标识
3. 了解锁定数据块、锁定应用族标识、锁定数据存储格式标识操作
4. 获取卡片系统信息
5. 获取多个安全块状态
6. 获取所有标签操作

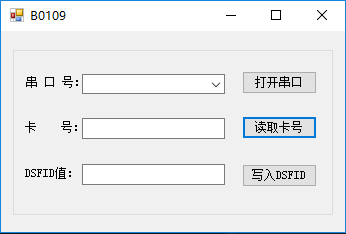
**二、 实验内容**

1. **设置应用族标识**



|  |  |  |  |
| --- | --- | --- | --- |
| 控件名称 | Text属性 | Name属性 | 功能 |
| Form1窗体 | B0107 |  |  |
| Label标签 | 串口号： |  |  |
| Lable标签 | 卡号： |  |  |
| Lable标签 | AFI值： |  |  |
| comboBox控件 |  | cmbPortID | 存放读取到的串口号 |
| TextBox控件 |  | txtCardID | 存放卡号 |
| TextBox控件 |  | txtAFI | AFI值 |
| Button控件 | 设置AFI | btnSetAFI |  |
| Button控件 | 打开串口 | btnOpenClosePort |  |
| Button控件 | 读取卡号 | btnCardID |  |
| GroupBox容器 |  |  |  |

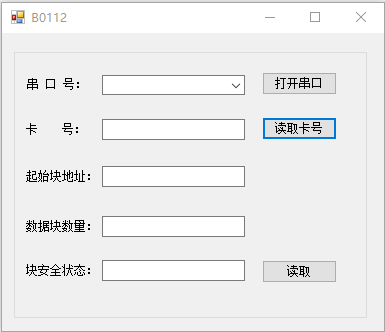
1. **设置数据存储格式标识**



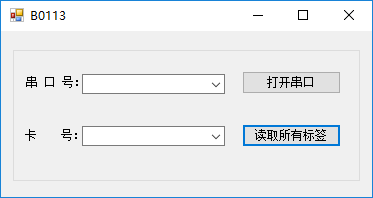
1. **获取卡片系统信息**



1. **获取多个安全块状态**



1. **获取所有标签操作**



**三、 操作步骤**

1. **窗体启动事件**

在窗体启动事件中调用System.IO.Ports.SerialPort.GetPortNames方法获取当前计算机串口号数组，使用循环将获取到的所有串口号添加到comPortID的项列表中。

1. **打开串口按钮事件**

利用KV\_ISO15693.Reader类中的OpenSerialPort方法打开串口，判断返回值，如果返回值为0x00则打开串口成功，否则打开串口失败。

利用KV\_ISO15693.Reader类中的CloseSerialPort方法关闭串口，判断返回值，如果返回值为0x00则关闭串口成功，否则关闭串口失败。

在打开串口成功后将“打开串口”按钮的Text属性修改为“关闭串口”，再次单击按钮就是执行关闭串口的代码。在关闭串口成功后将“关闭串口”按钮的Text属性修改为“打开串口”，再次单击按钮就是执行打开串口的代码。

1. **读取卡号按钮事件**

利用KV\_ISO15693.Reader类中的Inventory方法获取卡号。判断返回值是否为0x00，是则把卡号显示到文本框中，否则提示寻卡失败。

1. **设置AFI按钮事件**

调用KV\_ISO15693.Reader类中的WriteAFI方法写入AFI，判断返回值是否为0x00，是则提示写入成功，否则提示写入失败。

1. **写入DSFID按钮事件**

调用KV\_ISO15693.Reader类中的WriteDSFID方法写入DSFID，判断返回值是否为0x00，是则提示写入成功，否则提示写入失败。

1. **获取系统信息按钮事件**

调用KV\_ISO15693.Reader类中的GetSystemInfo方法读取系统信息，判断返回值是否为0x00，是则提示读取数据成功，否则提示读取数据失败。

1. **读取多个安全块状态**

调用KV\_ISO15693.Reader类中的GetMultiBlockSec方法读取数据块安全状态，判断返回值是否为0x00，是则提示读取数据成功，否则提示读取数据失败。

1. **读取所有标签**

调用KV\_ISO15693.Reader类中的GetAllTagsNum方法读取所有标签，判断返回值是否为0x00，是则提示读取到标签的数量，否则提示未读取到标签。

**四、 思考与练习**

**五、实验结果及后记**

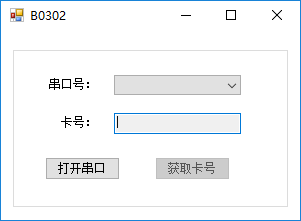
## 实验7 高频RFID系统ISO 14443实验

**一、实验目的**

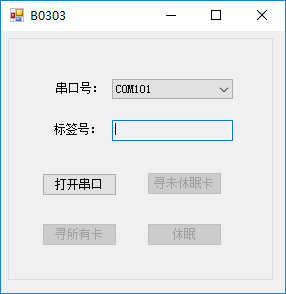
1. 寻卡操作，读取标签号。
2. 休眠标签
3. 验证密钥
4. 卡数据读写
5. 电子钱包

**二、 实验内容**

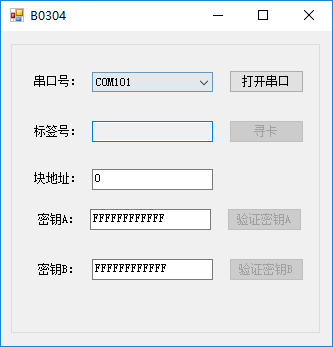
1. 寻卡操作，读取标签号。



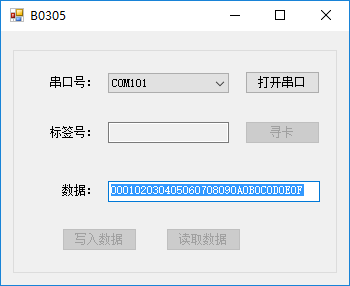
1. 休眠标签



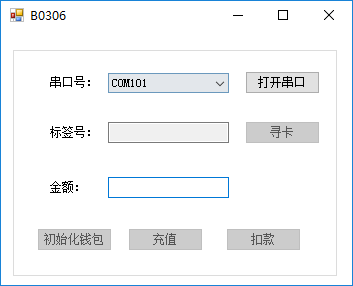
1. 验证密钥



1. 卡数据读写



1. 电子钱包



**三、 操作步骤**

1. public static StatusCode OpenSerialPort(String portName)

描述：打开串口；

参数：portName为串口号,如"COM1";串口打开时采用设备默认配置:波特率9600;数据位8;停止位1;奇偶校验无.

返回值：打开成功返回0x00，打开失败返回0x01；

示例：Byte value = FR102.OpenSerialPort("COM1")

注：示例中的FR102是接口函数所在类的类名，下同。

1. public static StatusCode CloseSerialPort()

描述：关闭串口；

参数：无.

返回值：关闭成功返回0x00，关闭失败返回0x01；

示例：Byte value = FR102.CloseSerialPort()

1. public static StatusCode TestReader()

描述：检测设备是否已经连接到当前打开的串口;建议在每次打开串口后立即进行检查；

参数：无.

返回值：连接成功返回0x00，连接失败返回0x01；

示例：Byte value = FR102.TestReader()

1. public static StatusCode RestartReader()

描述：重新启动FR102读写器设备;建议在设备连接成功后，进行读写器相关操作前重启设备；

参数：无.

返回值：启动成功返回0x00，启动失败返回0x01；

示例：Byte value = FR102.RestartReader()

1. public static StatusCode ChangeBaudRate(Int32 BaudRate )

描述：修改串口波特率；缺省的波特率为9600，如果需要修改波特率，建议在重启设备（调用完RestartReader命令）后进行。如果修改了串口的波特率，则特别注意在程序结束运行之前一定要关闭串口，否则可能需要重新插拔读写器才能让其正常运行;

参数：BaudRate，波特率，读写器支持的波特率有：7200, 9600(缺省值), 14400, 19200, 38400, 57600, 115200, 128000, 230400, 460800, 921600, 1228800，因此在调用函数时传递的参数不要超出该范围。

返回值：修改成功返回0x00，修改失败返回0x0B；

示例：Byte value = FR102. ChangeBaudRate(115200)

1. 启动visualstudio，文件→新建→项目。

选择VisualC#→Windows窗体应用程序，输入名称→选择存储路径。

界面设计及控件属性

在工具箱中找到所需控件，然后双击或者拖拽都可以添加控件到窗体中。

注意：在编写程序之前，先要在项目中新建一个“Tools”文件夹，然后将“文档”文件夹下的“Tools”文件夹下的所有类文件复制到项目中的“Tools”文件夹下。接着将这些类文件添加到项目中。

**四、 思考与练习**

**五、实验结果及后记**

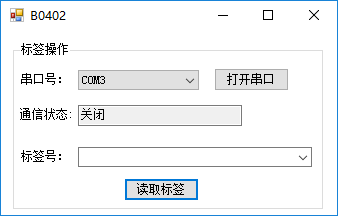
## 实验8 超高频RFID系统实验

**一、实验目的**

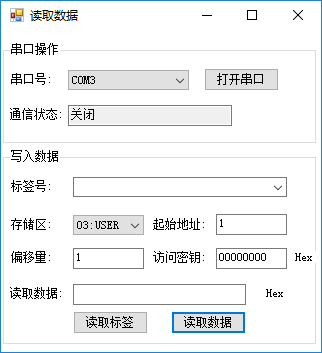
1. UHF寻卡
2. 数据写入 / 数据读取
3. 功率设置

**二、 实验内容**

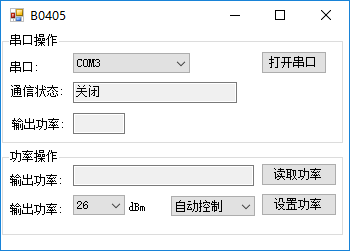
1. UHF寻卡



1. 数据写入 / 数据读取



1. 功率设置



**三、 操作步骤**

9.1 OpenAndConnect方法

函数原型：public static ErrorCode OpenAndConnect (String PortName)

public static ErrorCode OpenAndConnect (String PortName, Int32 BaudRate)

函数说明：用于打开与读写器连接的串口。

String类型的参数PortName表示要打开的串口号，如"COM1"；

Int32类型的参数BaudRate表示在串口上使用的波特率，缺省为57600；

返回值：ErrorCode枚举类型。

9.2 CloseAndDisconnect方法

函数原型：public static Byte CloseAndDisconnect ()

函数说明：用于关闭与读写器连接的串口。

返回值：ErrorCode枚举类型。

9.3 GetPaStatus方法

函数原型：public static ErrorCode GetPaStatus(out Boolean PaStatus, Boolean CRCFlag)

函数说明：获取与读写器的连接状态。

输出布尔型参数PaStatus，用于指示是否与读写器设备连接成功；

布尔型参数CRCFlag用于指示在与读写器交互过程中是否使用CRC校验。

返回值：ErrorCode枚举类型。

9.4 InventorySingle方法

函数原型：public static ErrorCode InventorySingle(out Byte[] UII, Boolean CRCFlag)

函数说明：单步识别单个标签，并读取标签UII。该方法只能用于识别单个标签，它不启动识别循环，也不会触发TagDetected事件。

输出字节数组型参数UII用于存放读写器返回的标签UII。

布尔型参数CRCFlag用于指示在与读写器交互过程中是否使用CRC校验。

返回值：ErrorCode枚举类型。

9.5 StopInventory方法

函数原型：public static ErrorCode StopInventory(Boolean CRCFlag)

函数说明：停止读写器的识别循环，即Inventory操作

布尔型参数CRCFlag用于指示在与读写器交互过程中是否使用CRC校验。

返回值：ErrorCode枚举类型。

9.6 WriteData方法

函数原型：public static ErrorCode WriteData(Byte[] AccessPassword, Byte Bank, Int32 Pointer, Byte Count, Byte[] UII, Byte[] Data, Boolean CRCFlag)

函数说明：用于往指定标签写入数据。

字节数组型参数AccessPassword为访问密钥，长度为4个字节。当四个字节不全为零时，读写器将利用访问密钥以确保在安全状态下访问标签。

字节型参数Bank为数据区类型，00:Reserved；01:UII；02:TID；03:USER。

整型参数Pointer为起始地址的偏移量；

字节型参数Count为写入的数据长度，以字(2Byte)为单位；

字节数组型参数UII为标签的 UII，用于指定欲写入数据的标签；

字节数组型参数Data，用于存储读欲写入的数据，长度应为Count\*2个Byte；

布尔型参数CRCFlag用于指示在与读写器交互过程中是否使用CRC校验。

返回值：ErrorCode枚举类型。

9.7 ReadData方法

函数原型：public static ErrorCode ReadData(Byte[] AccessPassword,Byte Bank,Int32 Pointer,Byte Count,Byte[] UII,out Byte[] Data, Boolean CRCFlag)

函数说明：用于读取指定标签的数据。

字节数组型参数AccessPassword为访问密钥，长度为4个字节。当四个字节不全为零时，读写器将利用访问密钥以确保在安全状态下访问标签。

字节型参数Bank为数据区类型，00:Reserved；01:UII；02:TID；03:USER。

整型参数Pointer为起始地址的偏移量；

字节型参数Count为读取的数据长度，以字(2Byte)为单位；

字节数组型参数UII为标签的 UII，用于指定欲读取的标签；

输出字节数组型参数Data，用于存储读取到的数据，长度为Count\*2个Byte；

布尔型参数CRCFlag用于指示在与读写器交互过程中是否使用CRC校验。

返回值：ErrorCode枚举类型。

9.8 StopInventory方法

函数原型：public static ErrorCode StopInventory(Boolean CRCFlag)

函数说明：停止读写器的识别循环，即Inventory操作

布尔型参数CRCFlag用于指示在与读写器交互过程中是否使用CRC校验。

返回值：ErrorCode枚举类型。

9.9 GetPower命令

函数原型：public static ErrorCode GetPower(out Byte Power, Boolean CRCFlag)

函数说明：获取读写器功率设置

输出字节型参数Power，表示读写器的功率字节，其中最高位表示功率控制方式：0代表手动，1代表自动，低7位为输出功率，单位dBm。

布尔型参数CRCFlag用于指示在与读写器交互过程中是否使用CRC校验。

返回值：ErrorCode枚举类型。

9.10 SetPower命令

函数原型：public static ErrorCode SetPower(Byte Power,Byte Option, Boolean CRCFlag)

函数说明：设置读写器输出功率；

字节型参数Power为读写器的功率字节，最高位表示功率控制方式：0代表手动，1代表自动；低7位为输出功率，单位dBm；

字节型参数Option为功率设置控制字节。最低位为输出功率控制位：1表示POWER字节的Bit6~0有效，0表示POWER字节的Bit6~0无效；次低位为功率控制模式控制位：1表示POWER字节的Bit7有效，0表示POWER字节的Bit7无效。

布尔型参数CRCFlag用于指示在与读写器交互过程中是否使用CRC校验。

返回值：ErrorCode枚举类型。

9.11 实验操作

1新建项目

启动visualstudio，文件→新建→项目。

选择VisualC#→Windows窗体应用程序，输入名称→选择存储路径。

2界面设计及控件属性

3实验代码解析

注意：在编写程序之前，先要在解决方案中添加类库“KV\_ISO18000-6”，然后将类库引用到项目中。

4窗体启动事件及各界面控件

在窗体启动事件中调用System.IO.Ports.SerialPort.GetPortNames方法获取当前计算机串口号数组，使用循环将获取到的所有串口号添加到cmbPortsName的项列表中。

**四、 思考与练习**

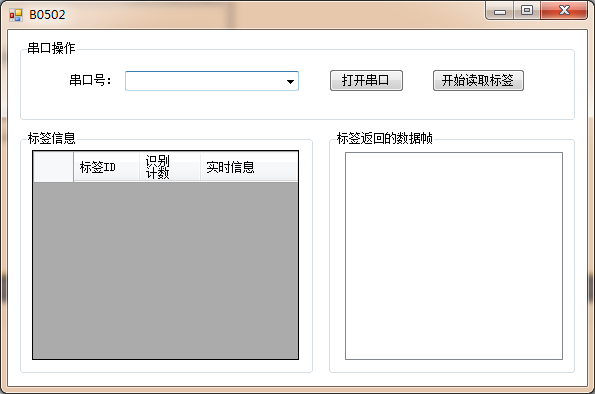
## 实验9 有源2.4GHz开发实验

**一、实验目的**

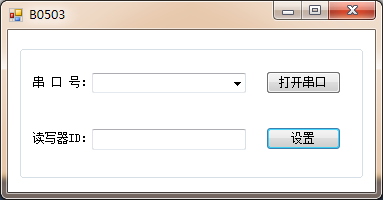
1. 标签读取
2. 设置标签ID
3. 设置读写器ID

**二、 实验内容**

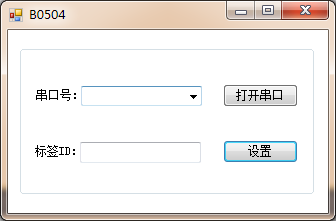
1. 标签读取



1. 设置读写器ID



1. 设置标签ID



**三、 操作步骤**

（1）、在项目中添加文档目录下的SHFDLL.CS类，在所需窗体引用命名空间SHF。在所需使用串口的窗体中添加命名空间System.IO.Ports。

using SHF;

using System.IO.Ports;

（2）、实例化SHFDLL类

SHFDLL read = newSHFDLL();

窗体启动事件

（3）、打开串口/关闭串口

A)利用SHFDLL类中的OpenSerial（）方法打开串口，

B)利用SHFDLL类中的CloseSerial（）方法关闭串口，

（4）、标签读取

利用SHFDLL类中的Inventory（）方法读取标签信息，当读取到标签信息的时候借助Read\_TagDetected（）方法把从读写器接收到的数据帧的标签信息用AddRow（）方法显示到DataGridView控件中，用AddUII（）把数据帧显示到ListBox控件中

注：标签上报的数据帧格式为251 16 TagID 00 SSI 00 ReaderID 01

其中，TagID 为标签ID号；SSI 为标签发射信号强度。

（5） 设置读写器ID

利用SHFDLL类中的SetReadID（）方法设置读写器的ID号。

（6）设置标签ID

利用SHFDLL类中的SetTagID（）方法设置标签ID。

**四、 思考与练习**

**五、实验结果及后记**