B0101 指导文档　ISO15693 串口操作

1. 实验目的

该实验主要是让学生熟悉 ISO15693 读写器的串口操作方法。

2. 实验设备

软件：visualstudio2010 及以上版本
硬件：ISO/IEC15693 RFID 读写器，九针串口线，电源（规格为电压 9V，电流 2A）

3. 实验原理

以下对 ISO15693 接口的相关介绍只是一部分，详情见其 ISO15693 接口说明文档和 ISO15693 接口源码。

3.1 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。

3.2 CloseSerialPort 方法

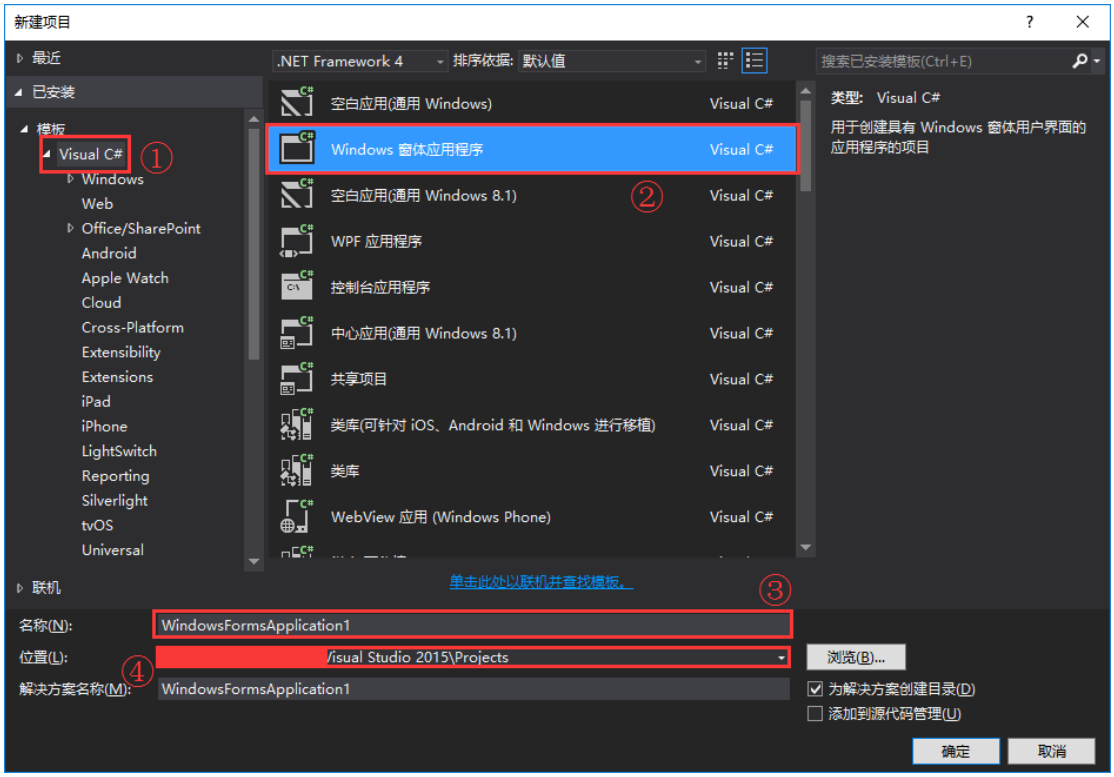
函数原型：public Byte CloseSerialPort()
描述：关闭串口；
参数：无.
返回值：关闭成功返回 0x00，关闭失败返回串口错误 0x01；
示例：Byte value = reader.CloseSerialPort();

4. 实验设计

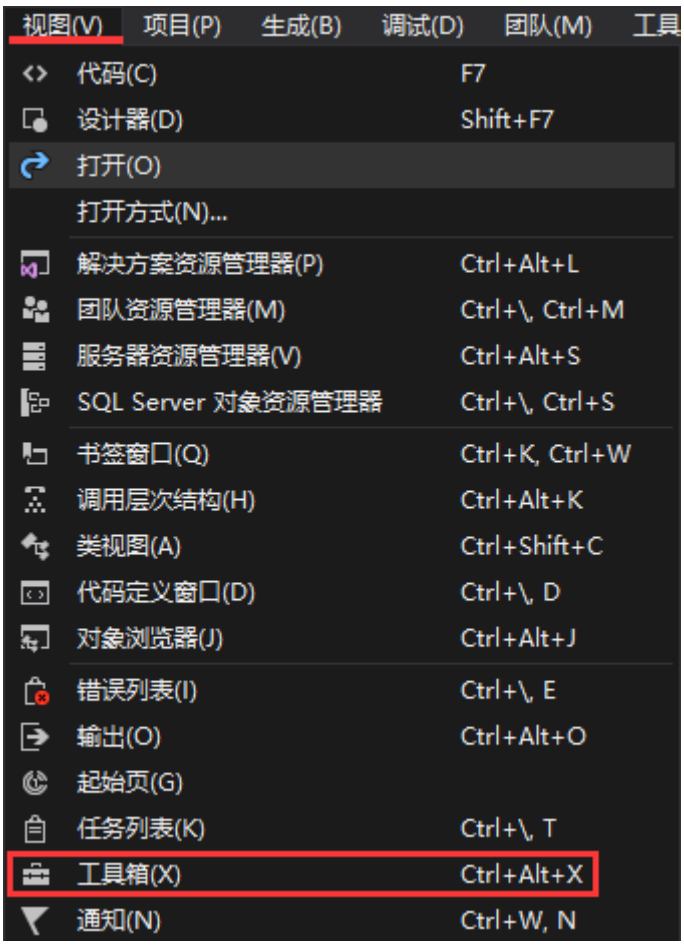
4.1 新建项目

启动 visualstudio，文件→新建→项目。

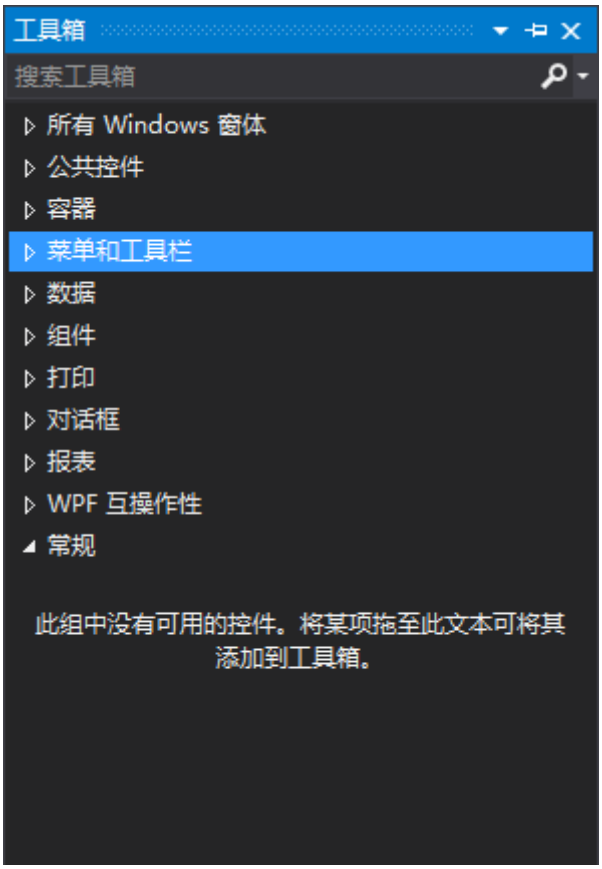
选择 VisualC#→Windows 窗体应用程序，输入名称→选择存储路径。



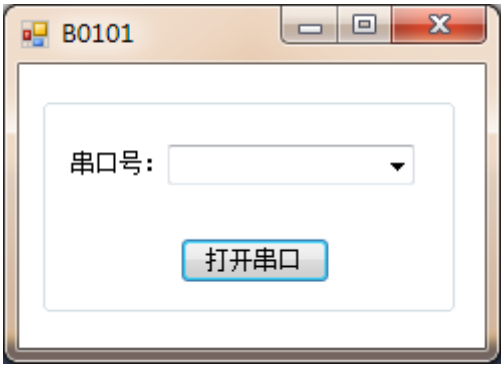
4.2 界面设计及控件属性



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。



控件名称	Text 属性	Name 属性	功能
From1 窗体	B0101		
Label 标签	串口号：		
comboBox 控件		cmbPortID	获取计算机串口
Button 控件	打开串口	btnOpenClosePort	
GroupBox 容器		groupBox1	



5. 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_ISO15693”，然后将类库引用到项目中。

5.1 窗体启动事件

在窗体启动事件中调用 SearchPort 方法。

```
private void Form1_Load(object sender, EventArgs e)
{
    SearchPort();
}
```

5.2 检测串口方法

使用 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。

```
private void SearchPort()
{
    //获取当前计算机串口数组
    string[] ports = SerialPort.GetPortNames();
    //把串口添加到comboBox控件中
    for (int i = 0; i < ports.Length; i++)
    {
        if (ports[i].Length < 7)
        {
            cmbPortID.Items.Add(ports[i]);
        }
    }
}
```

5.3 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。

```
private void btnOpenClosePort_Click(object sender, EventArgs e)
{
    if (btnOpenClosePort.Text == "打开串口")
    {
```

```
        try
        {
            //打开串口方法
            byte result = reader.OpenSerialPort(cmbPortID.Text);
            //判断返回值是否成功打开串口
            if (result == 0x00)
            {
                MessageBox.Show("串口成功打开！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                btnOpenClosePort.Text = "关闭串口";
            }
            else
            {
                MessageBox.Show("串口打开失败！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
    else
    {
        //判断串口是否打开
        if (reader.IsOpen)
        {
            //关闭串口方法
            Byte value = reader.CloseSerialPort();
            //判断返回值是否成功关闭串口
            if (value == 0x00)
            {
                MessageBox.Show("串口关闭成功！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                btnOpenClosePort.Text = "打开串口";
            }
            else
            {
                MessageBox.Show("串口关闭失败！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
        else
        {
```

```
            MessageBox.Show(String.Format("错误：串口已经处于关闭状态！"));
        }
    }
}
```

B0102 指导文档　ISO15693 寻卡操作

1. 实验目的

该实验主要是让学生熟悉读写器的读取卡号操作方法。

2. 实验设备

软件：visualstudio2010 及以上版本

硬件：ISO/IEC15693 RFID 读写器，ISO/IEC15693RFID 卡片，九针串口线，电源（规格为电压 9V，电流 2A）

3. 实验原理

以下对 ISO15693 接口的相关介绍只是一部分，详情见其 ISO15693 接口说明文档和 ISO15693 接口源码。

3.1 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。

3.2CloseSerialPort 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();

3.3 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：

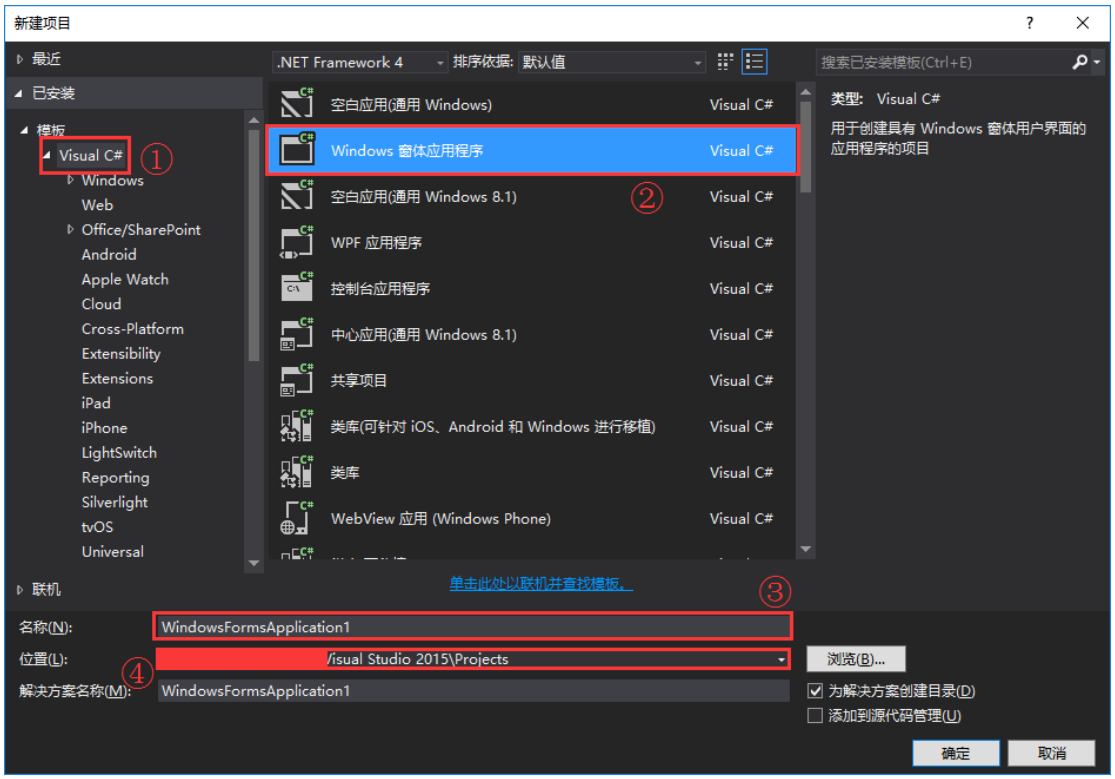
```
ModulateMethod mm=ModulateMethod.ASK;
InventoryModel im=InventoryModel.Multiple;
Int32 TagCount=0;
String[] TagNumber=new String[1];
Byte value = reader.Inventory(mm, im, ref TagCount, ref TagNumber);
```

4. 实验设计

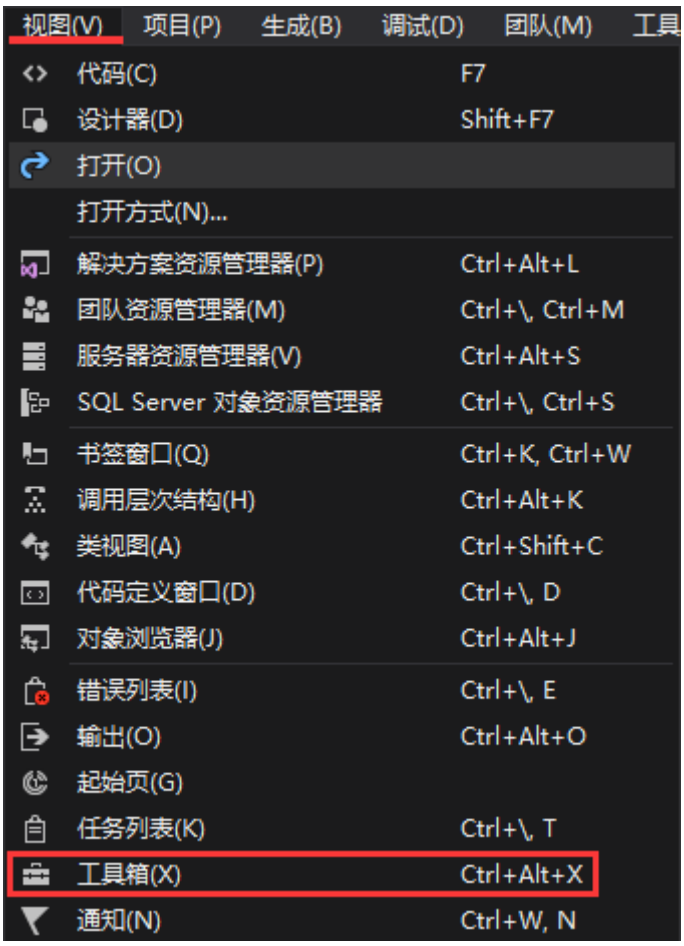
4.1 新建项目

启动 visualstudio，文件→新建→项目。

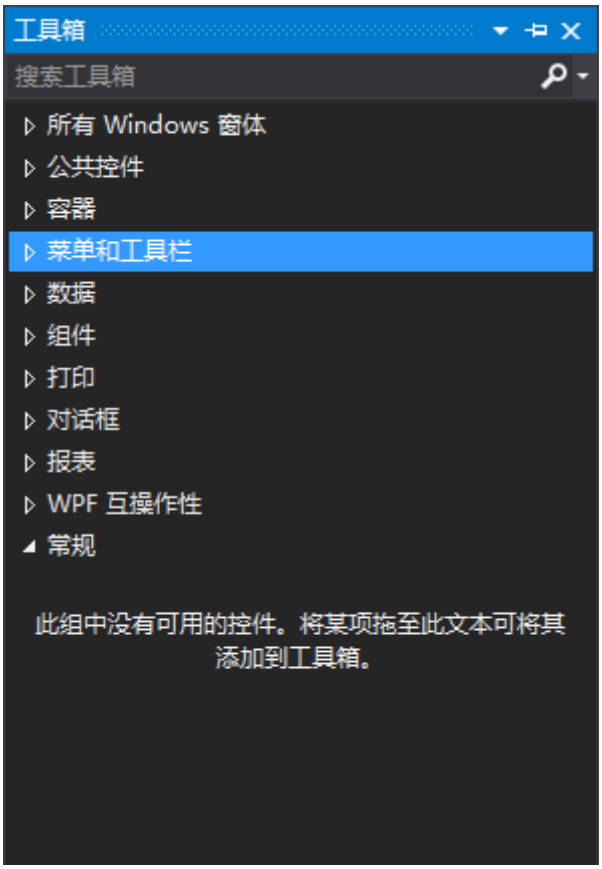
选择 VisualC#→Windows 窗体应用程序，输入名称→选择存储路径。



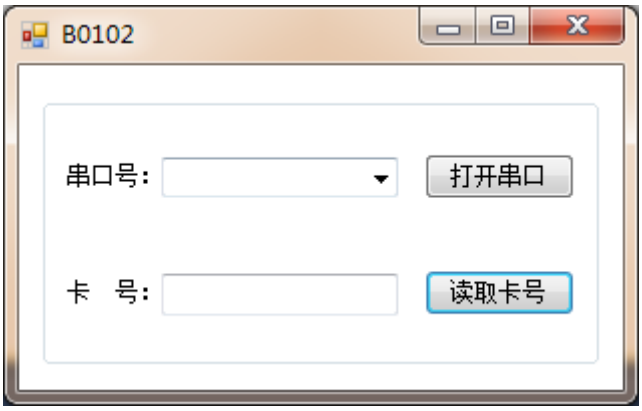
4.2 界面设计及控件属性



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。



控件名称	Text 属性	Name 属性	功能
From1 窗体	B0102		
Label 标签	串口号：		
comboBox 控件		cmbPortID	存放读取的串口号
Label 标签	卡号：		
TextBox 控件		txtCardID	存放读取到的卡号
Button 控件	打开串口	btnOpenClosePort	
Button 控件	读取卡号	btnReadPortID	
GroupBox 容器			



5. 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_ISO15693”，然后将类库引用到项目中。

5.1 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。

```
private void Form1_Load(object sender, EventArgs e)
{
    //获取当前计算机串口数组
    string[] ports = SerialPort.GetPortNames();
    //把串口添加到comboBox控件中
    for (int i = 0; i < ports.Length; i++)
    {
        if (ports[i].Length < 7)
        {
            cmbPortID.Items.Add(ports[i]);
        }
    }
}
```

5.2 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。

```
private void btnOpenClosePort_Click(object sender, EventArgs e)
{
    if (btnOpenClosePort.Text == "打开串口")
    {
        try
        {
            //打开串口方法
            byte result = reader.OpenSerialPort(cmbPortID.Text);
```

```
            //判断返回值是否成功打开串口
            if (result == 0x00)
            {
                MessageBox.Show("串口成功打开！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                btnOpenClosePort.Text = "关闭串口";
            }
            else
            {
                MessageBox.Show("串口打开失败！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
    else
    {
        //判断串口是否打开
        if (reader.IsOpen)
        {
            //关闭串口方法
            Byte value = reader.CloseSerialPort();
            //判断返回值是否成功关闭串口
            if (value == 0x00)
            {
                MessageBox.Show("串口关闭成功！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                btnOpenClosePort.Text = "打开串口";
            }
            else
            {
                MessageBox.Show("串口关闭失败！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
        else
        {
            MessageBox.Show(String.Format("错误：串口已经处于关闭状态！"));
        }
    }
}
```

5.3 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。

```
private void btnReadPortID_Click(object sender, EventArgs e)
{
    SearchCardID();
}

private void SearchCardID()
{
    try
    {
        int tagCount = 0;
        string[] tagNumber = null;
        byte result = reader.Inventory(ModulateMethod.FSK,
InventoryModel.Multiple, ref tagCount, ref tagNumber);
        //判断返回值是否成功读取到卡号
        if (result == 0x00)
        {
            txtCardID.Text = tagNumber[0];
        }
        else
        {
            MessageBox.Show("寻卡失败！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

B0103 指导文档　ISO15693 写数据操作

1. 实验目的

该实验主要是让学生熟悉读写器把数据写入卡片的操作。

2. 实验设备

软件：visualstudio2010 及以上版本

硬件：ISO/IEC15693 RFID 读写器，ISO/IEC15693RFID 卡片，九针串口线，电源（规格为电压 9V，电流 2A）

3. 实验原理

以下对 ISO15693 接口的相关介绍只是一部分，详情见其 ISO15693 接口说明文档和 ISO15693 接口源码。

3.1 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。

3.2CloseSerialPort 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();

3.3 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);
```

3.4 WriteSingleBlock 方法

函数原型：public Byte WriteSingleBlock(String TagNum, BlockLength bl, Byte BlockAddrss, Byte[] BlockData)

描述：写入单个数据块的数据。

参数 1：TagNum，卡号，字符串类型，用于指定需要进行数据写入的卡片；

参数 2：bl，BlockLength 枚举类型，用于指定卡片的数据块长度；

参数 3：BlockAddrss，字节类型，用于指定需要进行写入的数据块地址；

参数 4：BlockData，字节数组类型，用于存放需要写入的块数据，注意，该字节数组的长度必须大于等于参数 2 中的规定块长度的值；

返回值：成功返回 0x00，失败返回其它；

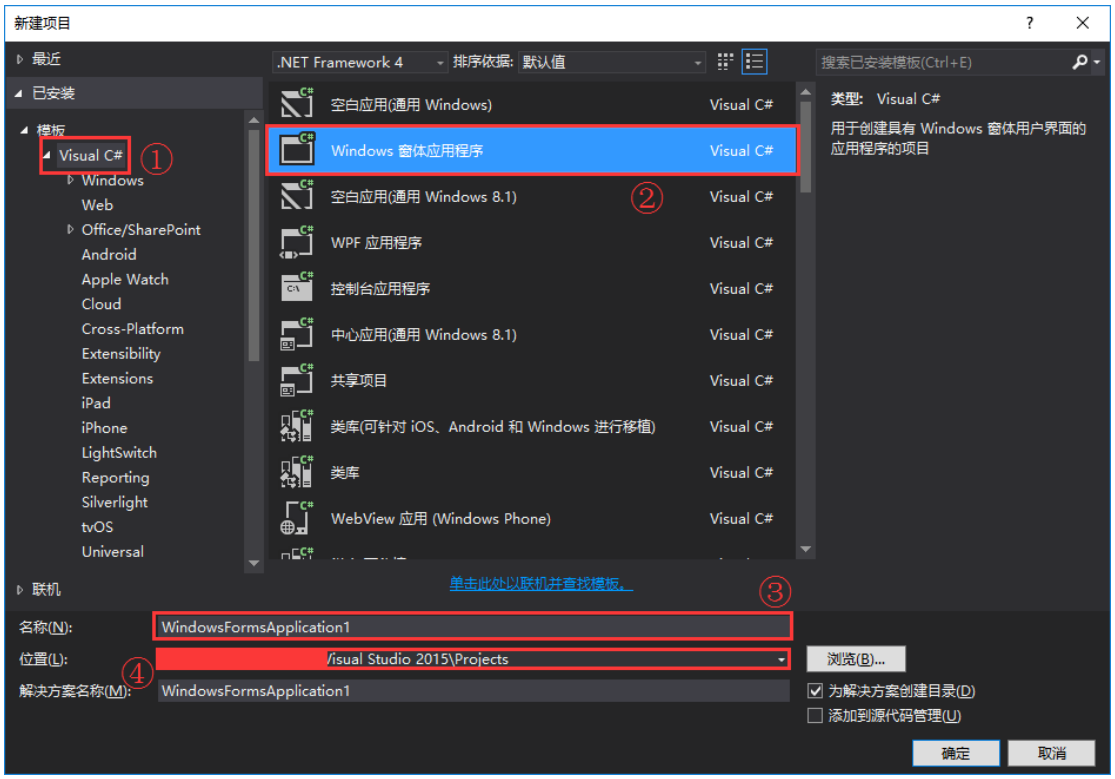
示例：
```
String TagNum="E004010012D99619";
BlockLength bl=BlockLength.ShortBlock4Byte;
Byte[] BlockData=new Byte[4] {0x00, 0x01, 0x02, 0x03};
Byte value = reader.WriteSingleBlock (TagNum, bl, 0x00, BlockData);
```

4. 实验设计

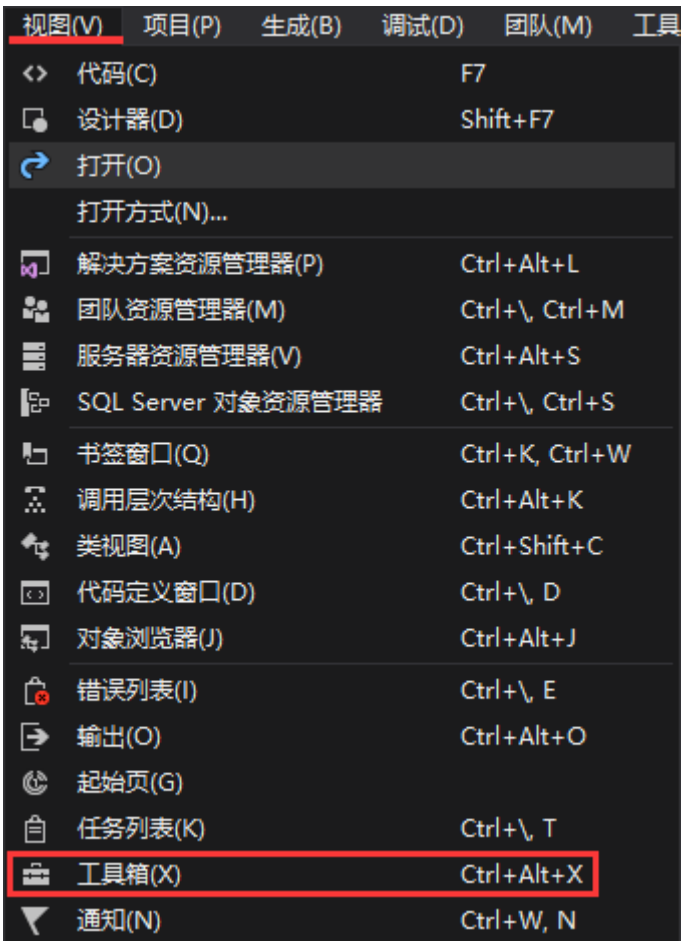
4.1 新建项目

启动 visualstudio，文件→新建→项目。

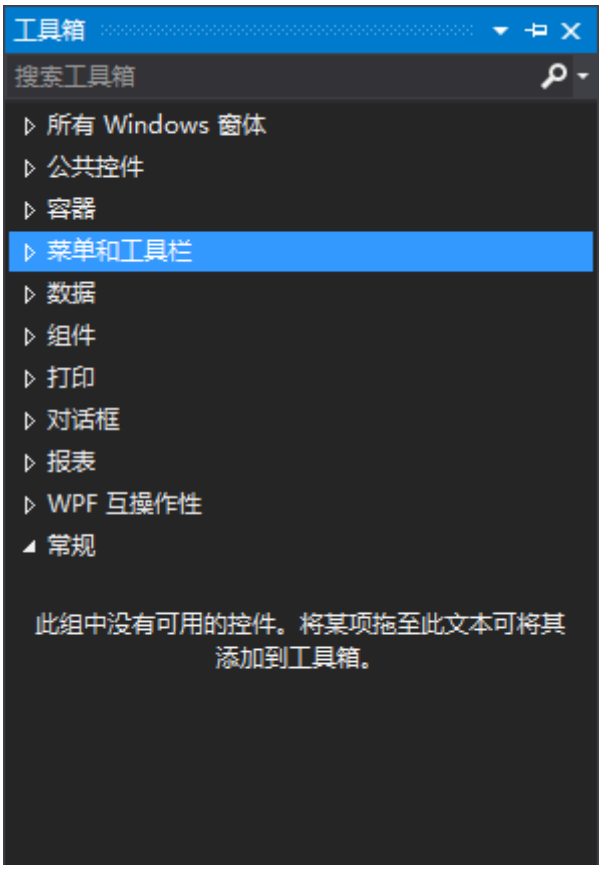
选择 VisualC#→Windows 窗体应用程序，输入名称→选择存储路径。



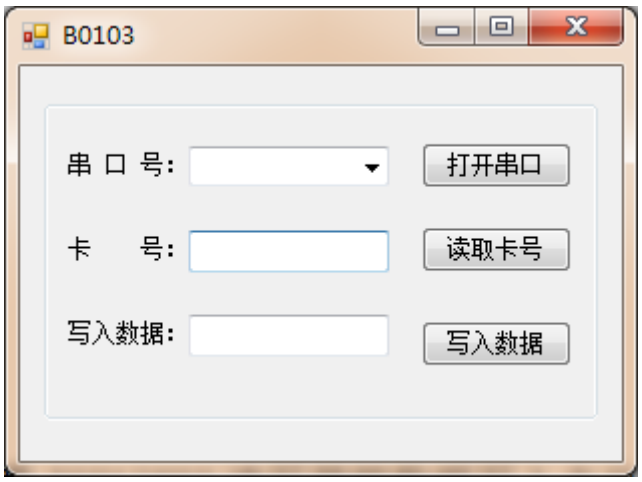
4.2 界面设计及控件属性



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。



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		txtWriteData	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			



5. 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_ISO15693”，然后将类库引用到项目中。

5.1 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。

```
private void Form1_Load(object sender, EventArgs e)
{
    //获取当前计算机串口数组
    string[] ports = SerialPort.GetPortNames();
    //把串口添加到comboBox控件中
    for (int i = 0; i < ports.Length; i++)
    {
        if (ports[i].Length < 7)
        {
            cmbPortID.Items.Add(ports[i]);
        }
    }
}
```

5.2 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。

```
private void btnOpenClosePort_Click(object sender, EventArgs e)
{
    if (btnOpenClosePort.Text == "打开串口")
    {
        try
        {
            //打开串口方法
            byte result = reader.OpenSerialPort(cmbPortID.Text);
            //判断返回值是否成功打开串口
            if (result == 0x00)
            {
                MessageBox.Show("串口成功打开！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                btnOpenClosePort.Text = "关闭串口";
            }
            else
            {
                MessageBox.Show("串口打开失败！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
    else
    {
        //判断串口是否打开
        if (reader.IsOpen)
        {
            //关闭串口方法
            Byte value = reader.CloseSerialPort();
            //判断返回值是否成功关闭串口
            if (value == 0x00)
            {
                MessageBox.Show("串口关闭成功！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                btnOpenClosePort.Text = "打开串口";
```

```
            }
            else
            {
                MessageBox.Show("串口关闭失败！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
        else
        {
            MessageBox.Show(String.Format("错误：串口已经处于关闭状态！"));
        }
    }
}
```

5.3 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。

```
private void btnReadCardID_Click(object sender, EventArgs e)
{
    SearchCardID();
}

private void SearchCardID()
{
    try
    {
        int tagCount = 0;
        string[] tagNumber = null;
        byte result = reader.Inventory(ModulateMethod.FSK,
InventoryModel.Multiple, ref tagCount, ref tagNumber);
        //判断返回值是否成功读取到卡号
        if (result == 0x00)
        {
            txtCardID.Text = tagNumber[0];
        }
        else
        {
            MessageBox.Show("寻卡失败！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }
```

```
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

5.4 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。

```
private void btnWriteData_Click(object sender, EventArgs e)
{
    if (txtWriteData.Text.Length == 8)
    {
        try
        {
            byte[] strData =
reader.StringToByteArray(txtWriteData.Text.Trim());
            byte result = reader.WriteSingleBlock(txtCardID.Text.Trim(),
BlockLength.ShortBlock4Byte, 0x00, strData);
            if (result == 0x00)
            {
                MessageBox.Show("成功写卡！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Information);
            }
            else
            {
                MessageBox.Show("写卡失败！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
    else
    {
        MessageBox.Show("数据长度错误，请重新填写！", "提示",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        txtWriteData.Text = "";
```

```
        txtWriteData.Focus();
    }
}
```

B0104 指导文档　ISO15693 读数据操作

1. 实验目的

该实验主要是让学生熟悉读写器读取卡片内的数据。

2. 实验设备

软件：visualstudio2010 及以上版本

硬件：ISO/IEC15693 RFID 读写器，ISO/IEC15693RFID 卡片，九针串口线，电源（规格为电压 9V，电流 2A）

3. 实验原理

以下对 ISO15693 接口的相关介绍只是一部分，详情见其 ISO15693 接口说明文档和 ISO15693 接口源码。

3.1 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。

3.2CloseSerialPort 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();

3.3 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);
```

3.4 ReadSingleBlock 方法

函数原型：public Byte ReadSingleBlock(String TagNum, BlockLength bl, Byte BlockAddrss, ref Byte[] BlockData)

描述：读取单个数据块的数据。

参数 1：TagNum，卡号，字符串类型，用于指定需要进行数据读取的卡片；

参数 2：bl，BlockLength 枚举类型，用于指定卡片的数据块长度；

参数 3：BlockAddrss，字节类型，用于指定需要进行读取的数据块地址；

参数 4：BlockData，引用参数，字节数组类型，用于存放读取到的块数据；

返回值：成功返回 0x00，失败返回其它；

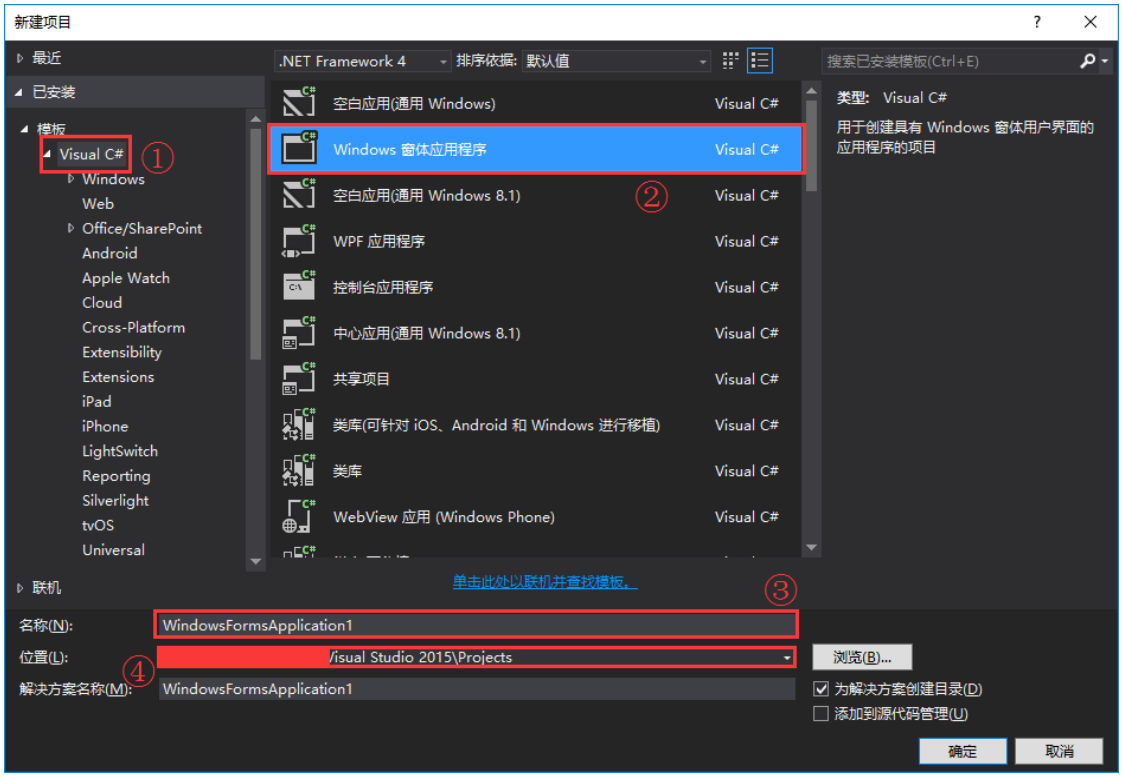
示例：
```
String TagNum="E004010012D99619";
BlockLength bl = BlockLength.ShortBlock4Byte;
Byte[] BlockData=new Byte[4];
Byte value = reader.ReadSingleBlock(TagNum, bl, 0x00, ref BlockData);
```

4. 实验设计

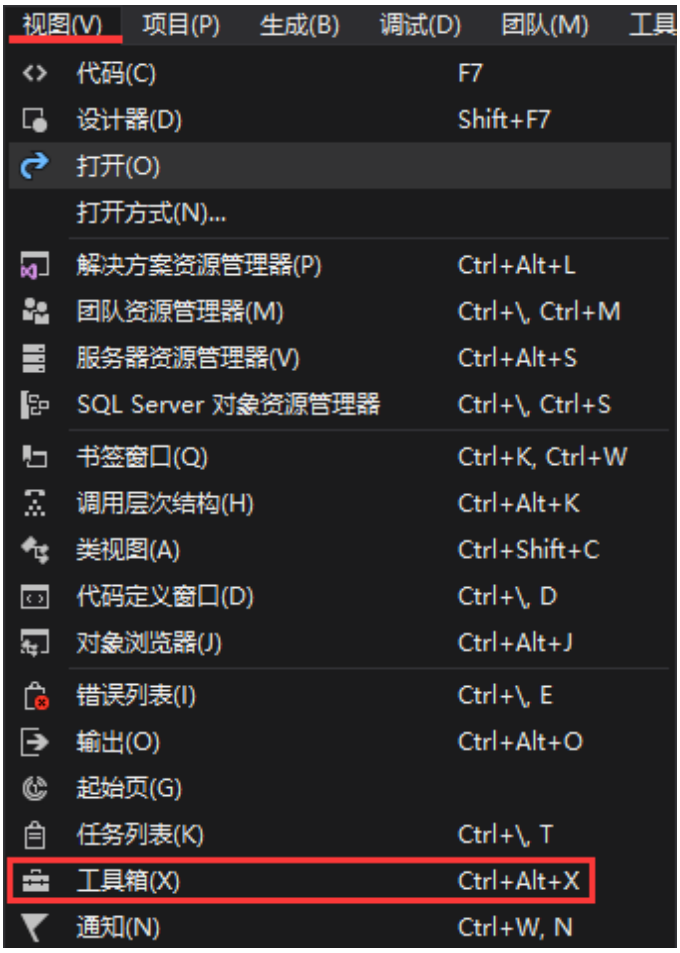
4.1 新建项目

启动 visualstudio，文件→新建→项目。

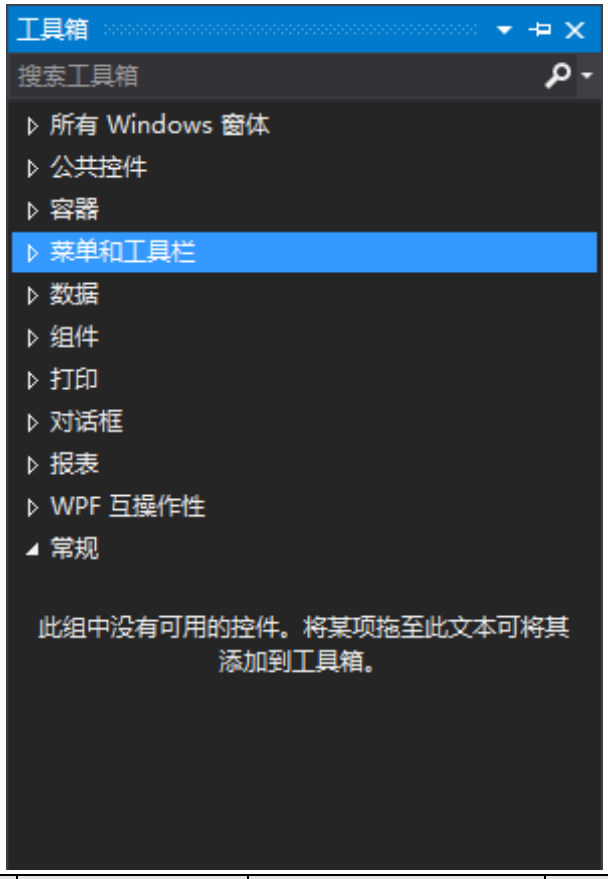
选择 VisualC#→Windows 窗体应用程序，输入名称→选择存储路径。


新建项目
最近
已安装
模板
Visual C#
①
Windows
Web
Office/SharePoint
Android
Apple Watch
Cloud
Cross-Platform
Extensibility
Extensions
iPad
iPhone
LightSwitch
Reporting
Silverlight
tvOS
Universal
联机
.NET Framework 4
排序依据: 默认值
空白应用(通用 Windows)
Visual C#
Windows 窗体应用程序
Visual C#
空白应用(通用 Windows 8.1)
②
Visual C#
WPF 应用程序
Visual C#
控制台应用程序
Visual C#
中心应用(通用 Windows 8.1)
Visual C#
共享项目
Visual C#
类库(可针对 iOS、Android 和 Windows 进行移植)
Visual C#
类库
Visual C#
WebView 应用 (Windows Phone)
Visual C#
单击此处以联机并查找模板。
③
搜索已安装模板(Ctrl+E)
类型: Visual C#
用于创建具有 Windows 窗体用户界面的应用程序的项目
名称(N):
WindowsFormsApplication1
位置(L):
④
Visual Studio 2015\Projects
浏览(B)...
解决方案名称(M):
WindowsFormsApplication1
为解决方案创建目录(D)
添加到源代码管理(U)
确定
取消

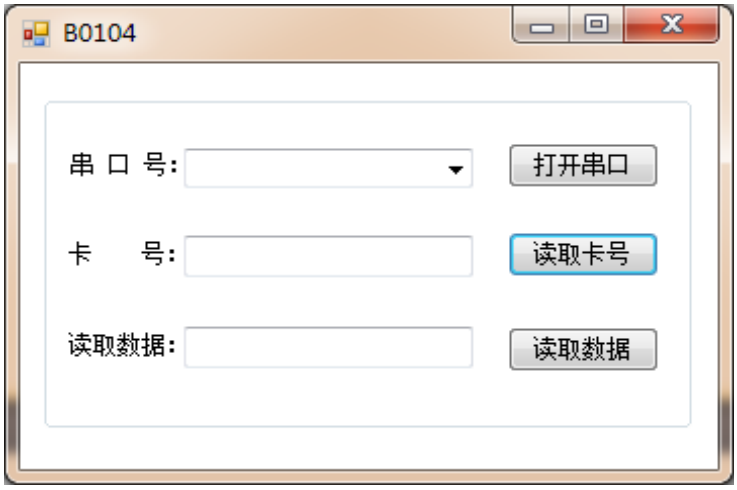
4.2 界面设计及控件属性



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。



控件名称	Text 属性	Name 属性	功能
From1 窗体	B0104		
Label 标签	串口号：		
Label 标签	卡号：		
Label 标签	读取数据：		
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	存放读取到的数据
Button 控件	打开串口	btnOpenClosePort	
Button 控件	读取卡号	btnCardID	
Button 控件	读取数据	btnReadData	
GroupBox 容器			



5. 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_ISO15693”，然后将类库引用到项目中。

5.1 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。

```
private void Form1_Load(object sender, EventArgs e)
{
    //获取当前计算机串口数组
    string[] ports = SerialPort.GetPortNames();
    //把串口添加到comboBox控件中
    for (int i = 0; i < ports.Length; i++)
    {
        if (ports[i].Length < 7)
        {
            cmbPortID.Items.Add(ports[i]);
        }
    }
}
```

5.2 打开串口按钮事件

调用 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。

调用 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。

```
private void btnOpenClosePort_Click(object sender, EventArgs e)
{
    if (btnOpenClosePort.Text == "打开串口")
    {
        try
        {
            //打开串口方法
            byte result = reader.OpenSerialPort(cmbPortID.Text);
            //判断返回值是否成功打开串口
            if (result == 0x00)
            {
                MessageBox.Show("串口成功打开！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                btnOpenClosePort.Text = "关闭串口";
            }
            else
            {
                MessageBox.Show("串口打开失败！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
        catch (Exception ex)
        {
            MessageBox.Show(ex.Message);
        }
    }
    else
    {
        //判断串口是否打开
        if (reader.IsOpen)
        {
            //关闭串口方法
            Byte value = reader.CloseSerialPort();
            //判断返回值是否成功关闭串口
            if (value == 0x00)
            {
                MessageBox.Show("串口关闭成功！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Information);
                btnOpenClosePort.Text = "打开串口";
            }
            else
```

```
            {
                MessageBox.Show("串口关闭失败！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
        }
        else
        {
            MessageBox.Show(String.Format("错误：串口已经处于关闭状态！"));
        }
    }
}
```

5.3 读取卡号按钮事件

调用 KV_IS015693.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。

```
private void btnReadCardID_Click (object sender, EventArgs e)
{
    SearchCardID();
}

private void SearchCardID()
{
    try
    {
        int tagCount = 0;
        string[] tagNumber = null;
        byte result = reader.Inventory(ModulateMethod.FSK,
InventoryModel.Multiple, ref tagCount, ref tagNumber);
        //判断返回值是否成功读取到卡号
        if (result == 0x00)
        {
            txtCardID.Text = tagNumber[0];
        }
        else
        {
            MessageBox.Show("寻卡失败！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Error);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

5.3 读取数据按钮事件

调用 KV_ISO15693.Reader 类中的 ReadSingleBlock 方法将数据写入卡片。判断返回值是否为 0x00，是则读卡成功，否则读卡失败。

因为 ReadSingleBlock 方法只能读单块，并且实验所用的 RFID 卡片单块存储长度为 4 个字节，所以 B0104 中一次只能读取 4 个字节长度的数据。

```
private void btnReadData_Click(object sender, EventArgs e)
{
    try
    {
        byte[] blockData={};
        byte result = reader.ReadSingleBlock(txtCardID.Text.Trim(),
BlockLength.ShortBlock4Byte, 0, ref blockData);
        if (result==0x00)
        {
            txtReadData.Text = reader.ByteArrayToString(blockData);
            MessageBox.Show("读取数据成功！","提示", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        }
        else
        {
            MessageBox.Show("读取到数据失败！", "提示", MessageBoxButtons.OK,
MessageBoxIcon.Information);
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```