# 第 4 章　ISO14443 读写操作

本章讲解如何在物联网虚拟仿真实验平台使用 C#语言对 ISO14443 进行读写操作，为后续课程打下坚实的基础。

## 4.1　ISO14443 的 API 参考手册

ISO14443 访问操作封装于动态链接库 FR102DLL.dll 中，下面对这个链接库的公共函数进行详细介绍。

1．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 0x01。

示例：Byte value= Reader.CloseSerialPort()

2．public Byte 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= Reader.TestReader ()

3．public Byte 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= Reader.RestartReader()

4．public Byte 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 RestartReader 命令）后进行，如果修改了串口的波特率，则在程序结束运行前一定要关闭串口，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 和 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= Reader. ChangeBaudRate (115200)

5．public Byte PcdRequest(Byte req_code, ref Byte[] TagType)

描述：请求命令，在每次寻卡之前必须运行 Request 命令，以便启动卡片上的 ARQ（请

求应答）模块，建立卡片与读写器的通信链路。

参数 1：req_code，请求模式，req_code=0x26，Request Idle，寻天线场区内未休眠的卡；req_code=0x52，Request All，寻天线场区内所有卡。

参数 2：TagType，卡片类型，该参数为引用参数，用于接收函数的返回值，正常情况下返回值（字节数组）的长度为 2Byte，如 0x00 02。特别注意在调用请求命令函数前，一定要先对该参数进行初始化，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：请求成功返回 0x00，请求失败返回 0x03，读写器天线场区无卡返回 0x02。

示例：Byte[] data = new Byte[2];

Byte value = Reader.PcdRequest(0x52, ref data);　//寻天线场区内所有卡

6. public Byte PcdAnticoll(ref Byte[] Snr)

描述：防冲突命令，用于获得天线场区内卡片的序列号，本书提供的版本的类库尚未实现读多卡的功能，因此在使用该命令时需保证场区内只有一张 Mifare 卡。

参数：Snr，卡片序列号，该参数为引用参数，正常情况下返回值的长度为 5Byte，其中前 4 个字节为卡片序列号，第 5 个字节为偶校验码，可用于验证序列号的正确性。

返回值：防冲突成功返回 0x00，防冲突失败返回 0x05。

示例：Byte[] data = new Byte[5];

Byte value = Reader.PcdAnticoll(ref data);

7. public Byte PcdSelect(Byte[] Snr)

描述：选择（激活）命令，用于选定指定参数的卡片，以便进行下一步的操作，如认证。

参数：Snr，卡片序列号，注意该参数不是引用参数，长度为 5Byte，其中前 4 个字节为卡片序列号，第 5 个字节为偶校验码。

返回值：激活成功返回 0x00，激活失败返回 0x06。

示例：Byte[] Snr = new Byte[5] {0x00,0x01,0x02,0x03, 0x00};　//序列号为 0x 00 01 02 03

Byte value = Reader.PcdSelect(Snr);

8. public Byte PcdAuthState(Byte auth_mode, Byte addr, Byte[] Key, Byte[] Snr)

描述：认证命令，用于对指定卡片的指定数据块进行认证，以便进行下一步的操作，如读/写操作必须通过认证后方可进行。Mifare 卡存储介质共 16 个扇区，每扇区 4 个块（Block），每块 16 个字节，其中每个扇区的最后一个块由密钥 A、存取控制和密钥 B 构成，用于对该扇区的数据进行存取控制，详见 Mifare 卡存储结构和存取控制。

参数 1：auth_mode，认证模式，auth_mode=0x60 为认证密钥 A；auth_mode=0x61 为认证密钥 B。

参数 2：addr，块地址。

参数 3：Key，密钥，长度为 6Byte。

参数 4：Snr，卡片序列号，长度 5Byte 或者 4Byte（可以没有第 5 个字节的偶校验码）。

返回值：认证成功返回 0x00，认证失败返回 0x08。

示例：Byte[] Snr = new Byte[5] {0x00,0x01,0x02,0x03}; //序列号为 0x 00 01 02 03

Byte[] KeyA = new Byte[6] {0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF }; //默认密钥

Byte value=Reader.PcdAuthState(0x60, 0x02, KeyA, Snr); //认证密钥 A

9. public Byte PcdRead(Byte addr, ref Byte[] Read_data)
描述：读取命令，用于读取指定数据块的数据，必须在通过必要的认证后才能执行。
参数 1：addr，块地址。
参数 2：Read_data，引用类型，长度为 16Byte，用于存放读取到的数据。
返回值：读取成功返回 0x00，读取失败返回 0x09。
示例：Byte[] data_Read = new Byte[16];
Byte value = Reader.PcdRead(addr, ref data_Read);
10. public Byte PcdWrite(Byte addr, Byte[] Write_data)
描述：写入命令，用于向指定的数据块写入数据，必须在通过必要的认证后才能执行。
参数 1：addr，块地址。
参数 2：Write_data，长度为 16Byte，用于存放需要写入的数据。
返回值：写入成功返回 0x00，写入失败返回 0x0A。
示例：Byte[] Write_data = new Byte[16];
for (Byte i = 0; i < 16; i++) { Write_data [i]=(Byte)i;}
Byte value = Reader.PcdWrite(addr, ref Write_data);
11. public Byte PcdHalt()
描述：休眠命令，让当前选定的卡进入休眠状态，必须在通过必要的认证后才能执行。
参数：无。
返回值：休眠成功返回 0x00，休眠失败返回 0x07。
示例：Byte value = Reader. PcdHalt();
12. public void BuzzerEnable(Boolean flag)
描述：板载蜂鸣器使能（开启/关闭）命令。
参数：flag，布尔型标识，flag=true 表示开启蜂鸣器，flag=false 表示关闭蜂鸣器。
返回值：无。
示例：Reader.BuzzerEnable(checkBox1.Checked);
13. public void LEDActEnable (Boolean flag)
描述：板载 LED 灯动作使能（开启/关闭）命令。
参数：flag，布尔型标识，flag=true 表示开启 LED 灯，flag=false 表示关闭 LED 灯。
返回值：无。
示例：Reader.LEDActEnable(checkBox2.Checked);
14. public void LEDUserEnable (Boolean flag)
描述：板载 LED 灯用户使能（开启/关闭）命令。
参数：flag，布尔型标识，flag=true 表示开启 LED 灯，flag=false 表示关闭 LED 灯。
返回值：无。
示例：Reader. LEDUserEnable (checkBox2.Checked);

## 4.2 ISO14443 的读写示例

本实验主要是为了让学生了解打开和关闭、读取和写入 13.56MHz ISO/IEC14443A/B RFID

读写器的基本方法。

开发环境：Microsoft Visual Studio 2012

开发语言：C#

1. 界面设计

新建一个 Windows 应用程序，在项目中新建一个“Tools”文件夹，然后将本书所提供的“Converter.cs”“CRC16Rev.cs”“FR102.cs”三个类文件复制到“Tools”文件夹并添加到项目中。

在 Debug 目录下新建一个名为“serial.ini”的文件并打开，在文件中加入如下两行代码：

```
COM101
9600
```

按照如图 4.1 所示的标注摆放控件。

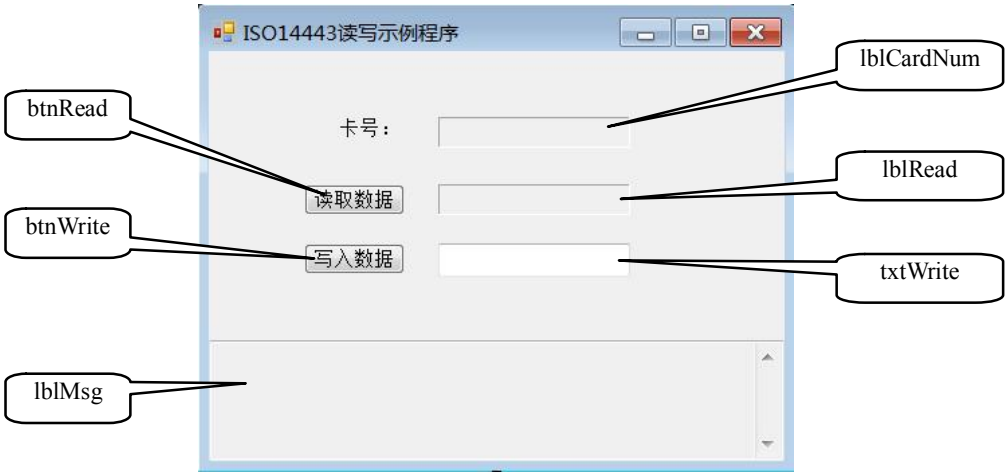


图 4.1　界面设计图

2. 代码编写

```
Tools.FR102 reader = new Tools.FR102();
Converter converter = new Converter();
string[] strConfig = File.ReadAllLines("serial.ini");
Thread myThread;
private void Form1_Load(object sender, EventArgs e)
{
    if (strConfig.Length < 2)
    {
        MessageBox.Show("请先进行串口号和波特率的配置", "提示", MessageBoxButtons.OK,
                MessageBoxIcon.Information);
        return;
    }

    if (reader.OpenSerialPort(strConfig[0]) == 0x00)
    {
        txtMsg.Text = String.Format("◆串口{0}打开成功！", strConfig[0]);
```

```
        }
        else
        {
            txtMsg.Text = String.Format("◆串口{0}打开失败！", strConfig[0]);
            return;
        }
        //检测并连接设备
        if (reader.TestReader() == 0x00)
        {
            txtMsg.Text += "\r\n" + String.Format("◆检测到连接到串口{0}的读卡器！", strConfig[0]);
        }
        else
        {
            txtMsg.Text += "\r\n" + String.Format("◆没有检测到连接到串口{0}的设备，请检查与设备连接
                    的串口！", strConfig[0]);
            ClosePort();
            return;
        }
        if (reader.RestartReader() == 0x00)
        {
            txtMsg.Text += "\r\n" + "◆设备重启成功！";
        }
        else
        {
            txtMsg.Text += "\r\n" + "◆设备重启失败！";
            ClosePort();
            return;
        }
        if (reader.ChangeBaudRate(Int32.Parse("9600")) == 0x00)
        {
            txtMsg.Text += "\r\n" + "◆修改串口波特率成功！";
        }
        else
        {
            txtMsg.Text += "\r\n" + "◆修改串口波特率失败！";
            ClosePort();
            return;
        }

        myThread = new Thread(new ThreadStart(CrossThreadInvoke));
        myThread.IsBackground = true;
        myThread.Start();
    }
    private void ClosePort()
    {
        if (reader.CloseSerialPort() == 0x00)
```

```
        {
            txtMsg.Text += "\r\n" + String.Format("◆串口{0}关闭成功！", strConfig[0]);
        }
    }

    /// <summary>
    /// 跨线程调用方法
    /// </summary>
    private void CrossThreadInvoke()
    {
        //将无线循环和 Sleep 放在等待异步外面
        while (true)
        {
            GetCardNum();
            Thread.Sleep(500);
        }
    }

    private void UpdateLblCardNum(string str)
    {
        if (lblCardNum.InvokeRequired)
        {
            Action<string> actionDelegate = delegate(string txt)
                {
                    if (txt != lblCardNum.Text)
                    {
                        lblCardNum.Text = txt;
                    }
                };
            lblCardNum.Invoke(actionDelegate, str);
        }
    }

    private void AddMsg(string str)
    {
        if (txtMsg.InvokeRequired)
        {
            Action<string> actionDelegate = (x) =>
            {
                txtMsg.Text += "◆" + str;
            };
            txtMsg.Invoke(actionDelegate, str);
        }
    }

    public void GetCardNum()
```

```
{
    string cardID;
    byte[] TagType;
    byte[] TagNumber;

    //寻天线区场内所有卡，得到卡类型
    if (reader.PcdRequest(0x52, out TagType) != FR102.StatusCode.AllDone)
    {
        UpdateLblCardNum("无卡");
        return;
    }
    //防冲撞，得到卡号
    FR102.StatusCode ec2 = reader.PcdAnticoll2(out TagNumber);
    //将数组 TagNumber 转换为字符串
    cardID = converter.ArrayToHexStr(TagNumber);

    if (cardID == "00000000")
    {
        cardID = "";
    }

    //设置蜂鸣器
    reader.EnableBuzzer(true);
    //设置 LED 灯
    reader.EnableLEDAct(true);
    Thread.Sleep(1000);
    reader.EnableBuzzer(false);
    reader.EnableLEDAct(false);

    //将字符串 cardID 转换为数组
    byte[] TagNumber2 = converter.HexStrToArray(cardID);
    //选定卡片
    FR102.StatusCode ec3 = reader.PcdSelect(TagNumber2);
    if (cardID != lblCardNum.Text)
    {
        UpdateLblCardNum(cardID);
    }
}

/// <summary>
///  字符串数据转换为 List 类型
  /// </summary>
  /// <param name="str"></param>
  /// <returns></returns>
  public List<byte[]> To16(string str)
  {
      byte[] buffer = System.Text.Encoding.UTF8.GetBytes(str);
```

```
            List<byte[]> myArr = new List<byte[]>();
            byte[] arr;
            int a = buffer.Length / 16;
            int b = buffer.Length % 16;
            for (int k = 0; k < a; k++)
            {
                arr = new byte[16];
                for (int m = 0; m < 16; m++)
                {
                    arr[m] = buffer[k * 16 + m];
                }
                myArr.Add(arr);
            }
            if (b != 0)
            {
                byte[] arr1 = new byte[16];
                for (int n = 0; n < 16; n++)
                {
                    if (n < b)
                    {
                        arr1[n] = buffer[a * 16 + n];
                    }
                    else
                    {
                        arr1[n] = System.Text.Encoding.UTF8.GetBytes("@")[0];
                    }
                }
                myArr.Add(arr1);
            }
            byte[] allByte = new byte[myArr.Count * 16];
            for (int k = 5; k < myArr.Count + 5; k++)
            {
                string temp1 = "";
                byte[] byteArr = myArr[k - 5];
                for (int x = 0; x < byteArr.Length; x++)
                {
                    temp1 = byteArr[x].ToString();
                }
             }
             return myArr;
    }

    /// <summary>
    ///  认证密钥的方法
    /// </summary>
    /// <param name="strCardNum">卡号</param>
    private void AuthenticateKey(string strCardNum)
    {
        byte addr = byte.Parse("4");
        //将密钥转换成字节数组
```

```
        byte[] keyA = converter.HexStrToArray("FFFFFFFFFFFF");
        //将卡号转换成字节数组
        byte[] tagNum = converter.HexStrToArray(strCardNum);
        //调用 PcdAuthState 方法验证密钥
        FR102.StatusCode ec = reader.PcdAuthState(0x60, addr, keyA, tagNum);
    }

    private void btnWrite_Click(object sender, EventArgs e)
    {
        myThread.Suspend();
        AuthenticateKey(lblCardNum.Text);
        string strData = txtData.Text.Trim();
        List<byte[]> myArr = To16(strData);
        byte[] allByte = new byte[myArr.Count * 16];
        FR102.StatusCode ec = FR102.StatusCode.WriteErr;

        for (int i = 5; i < myArr.Count + 5; i++)
        {
            byte[] byteArr = myArr[i - 5];
            ec = reader.PcdWrite(byte.Parse(i.ToString()), byteArr);
        }
        if (ec == 0x00)
        {
            txtData.Text = "";
            txtMsg.Text += "\r\n◆数据写入成功！";
        }
        lblRead.Text = ReadCard(lblCardNum.Text);
        myThread.Resume();
    }

    private string ReadCard(string strCardNum)
    {
        if (strCardNum == "")
        {
            txtMsg.Text += "\r\n◆未读到卡片，请确定卡片已放在读卡器上！";
        }
        string strInformation = "";
        byte[] data;

        //验证密钥
        AuthenticateKey(strCardNum);
        FR102.StatusCode ec = FR102.StatusCode.ComErr;

        for (int i = 5; i < 7; i++)
        {
            ec = reader.PcdRead(Convert.ToByte(i), out data);
            strInformation += System.Text.Encoding.UTF8.GetString(data);
        }

        if (ec == 0x00)
```

```
        {
            return strInformation.Substring(0, strInformation.IndexOf('@'));
        }
        else
        {
            return "";
        }
    }

    private void btnRead_Click(object sender, EventArgs e)
    {
        myThread.Suspend();
        lblRead.Text = ReadCard(lblCardNum.Text);
        myThread.Resume();
    }
```

3. 运行程序

打开物联网虚拟仿真实验平台，按照图 4.2 所示的方式创建 ISO14443 读写器及卡片。

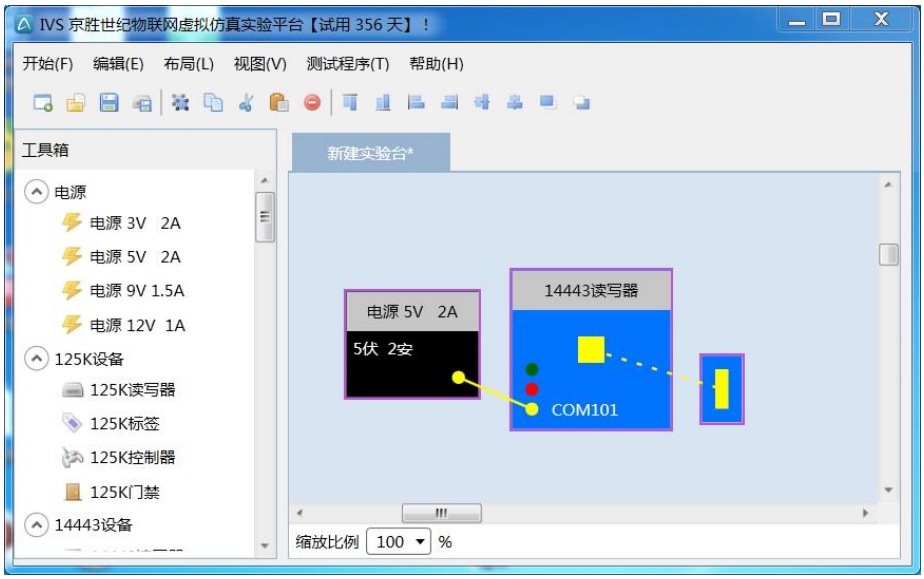


图 4.2 仿真组件摆放

运行程序，程序自动读取 ISO14443 卡号，写入内容并查看效果。

本程序使用了一个后台线程，每隔 1 秒读取是否有新卡片，并在写入数据后，自动读取一次卡片里的数据。