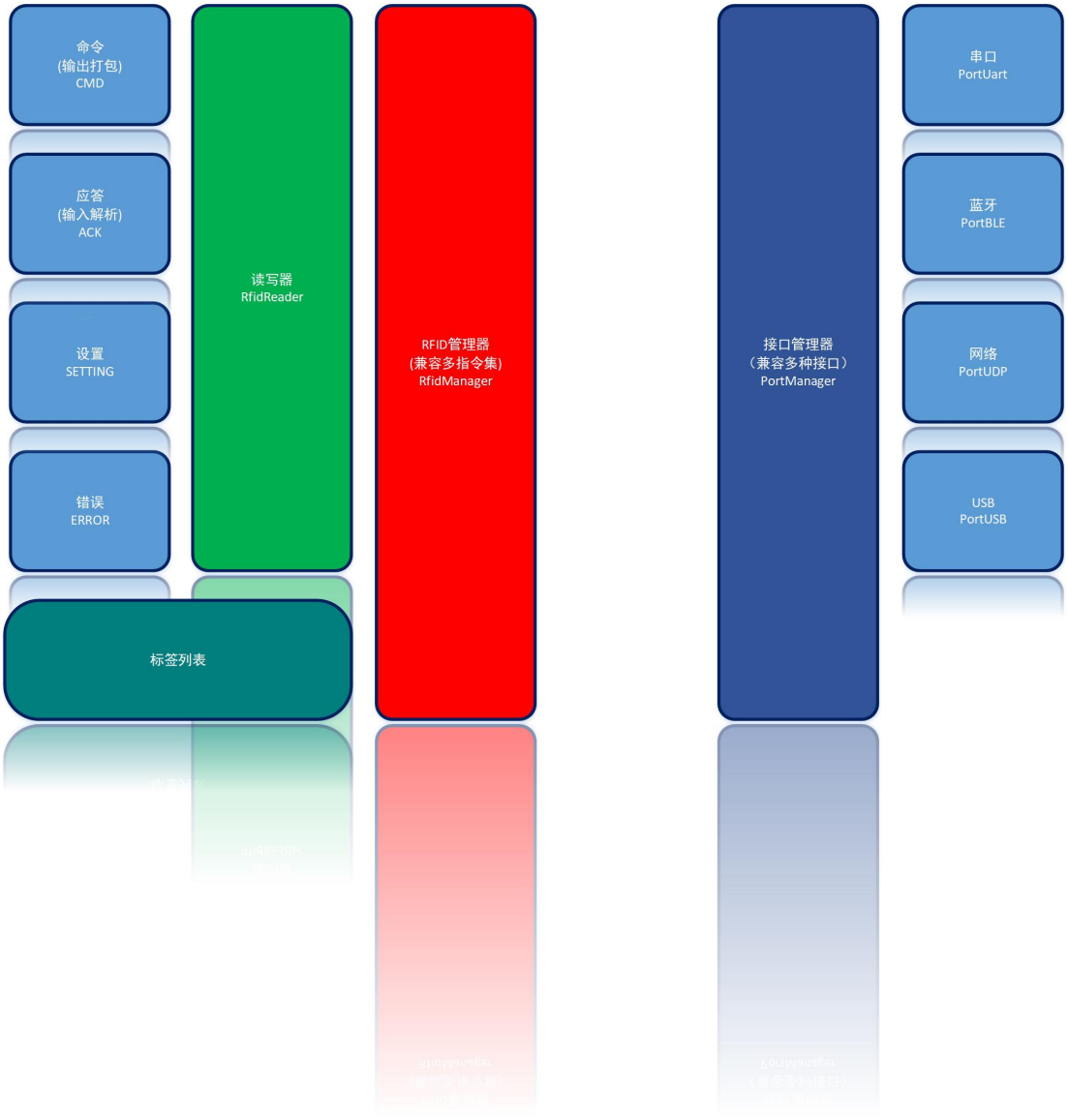
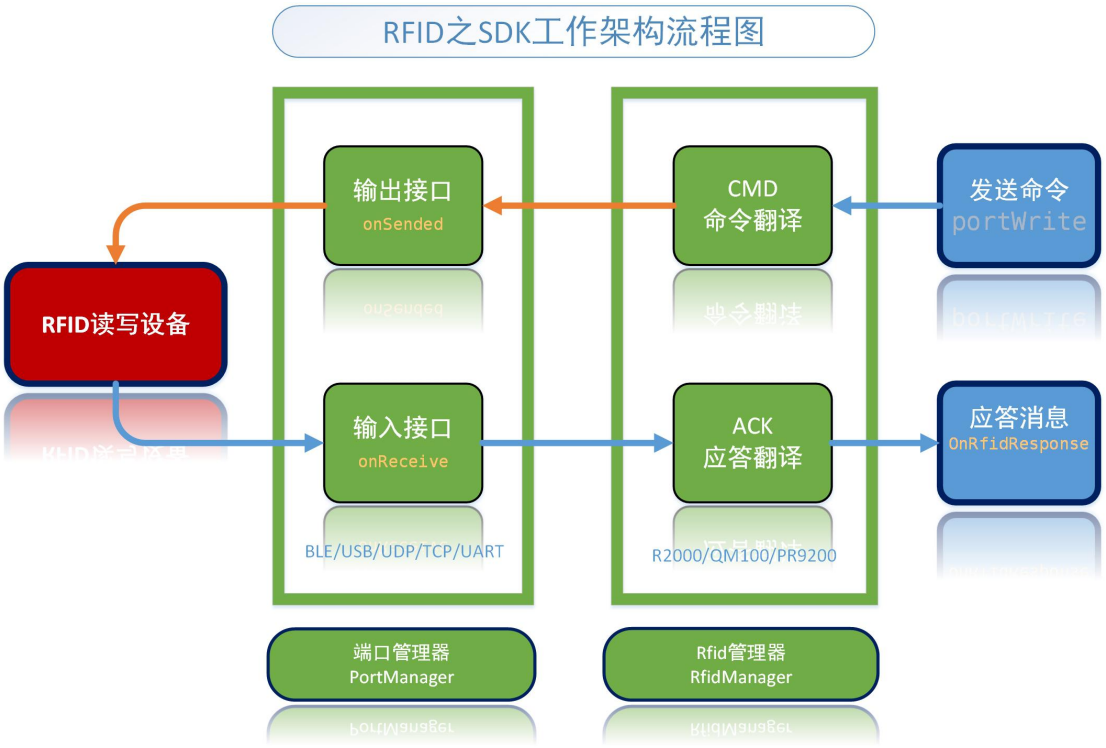
视窗版开发包使用说明

(Windows SDK)

版本:1.10

1. 环境要求:Visual Studio 2019 以上或
2. 包要求.NET 4.0 以上.
3. 开发语言:C#
4. 系统要求:Windows 10 以上(BLE 要求 Windows 8.1 以上).
5. 系统架构:



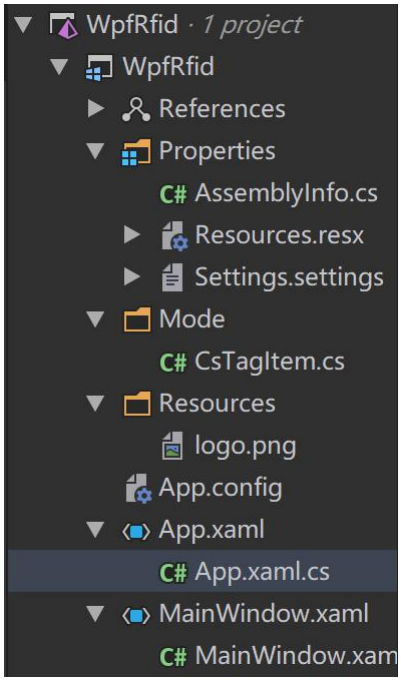


本开发包架构主要包含两大块：

1. 接口管理器(PortManager)
   目前支持：串口，网口，USB，蓝牙.

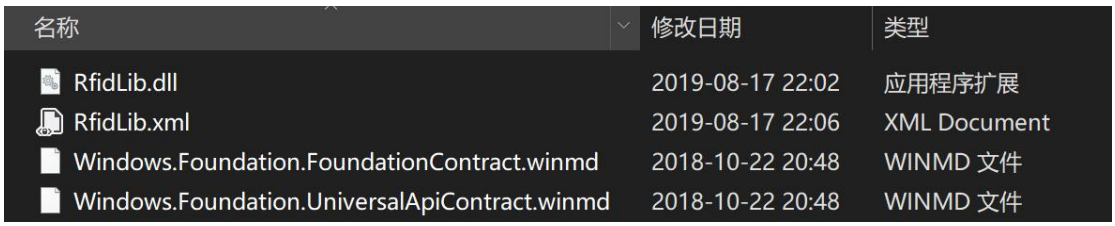
2. 电子标签管理器(RfidManager)
   目前支持：M100，R2000，J2000，...
   支持多种通道：1/2/4/8/16.

演示代码结构如下：



主要内容就在 App.xaml.cs 和 MainWindow.xaml.cs

demo 程序使用库如下：



名称	修改日期	类型
RfidLib.dll	2019-08-17 22:02	应用程序扩展
RfidLib.xml	2019-08-17 22:06	XML Document
Windows.Foundation.FoundationContract.winmd	2018-10-22 20:48	WINMD 文件
Windows.Foundation.UniversalApiContract.winmd	2018-10-22 20:48	WINMD 文件

基本用法只需要 **RfidLib.dll** 一个库，另外两个 **winmd** 文件，是蓝牙驱动的系统引用库，根据需要决定是否添加.

使用方式：

1. 创建 RfidSystem 对象.（指定端口 UART，读写器类型:R2000，以及使用 4 通道）

```
/// 标签管理器
private RfidSystem rfidSystem;
rfidSystem = new RfidSystem(this, PortType.PORT_TYPE_UART, ReaderType.READER_TYPE_R2000, 4);
```

获取读写器管理器

```
public static RfidManager GetRfidManager(){
    return GetInstance().rfidSystem.GetRfidManager();
}
```

```
// 获取端口管理器
public static PortManager GetPortManager()
{
    return GetInstance().rfidSystem.GetPortManager();
}
```

2. 设置消息接口

```
//  设置标签消息监听
GetRfidManager().SetRfidEvent(this); //对应 OnRfidResponse,响应标签消息
//  设置端口消息监听
GetPortManager().SetPortEvent(this); //对应 OnPortEvent,响应接口消息
```

3. 开始或停止扫描.

```
if (App.GetSystem().IsRunning())
{
    App.GetSystem().Stop();
    Log_d(TAG, "停止");
}
else
{
    App.GetSystem().Start();
    Log_d(TAG, "开始");
}
```

4. 收到标签, 或是其他消息, 进入如下回调接口:

```
public void OnRfidResponse(int type, int cmd, byte[] param, object obj)
{
   //  管理器
   var manager = App.GetRfidManager();
   //  读写器
   var reader = manager.GetReader();

   if (obj is string s)
   {
      Debug.Print("{0:X2} - {1}", cmd, s);
      Log_d(TAG, s);
   }

    //======================
   //  正常消息
   //======================
   switch (cmd)
```

```
    {
        //==================
        //  收到标签
        //==================
        case BaseCmd.RFID_CMD_INVENTORY:
            if (type == BaseAck.RESPONSE_TYPE_ERROR)
            {
                return;
            }

            //  收到标签
            if (obj is TagItem tag)
            {
               //  标签数量
                var text = "" + manager.GetList().Count;
                //  异步刷新
                Action<Label, DataGrid, string, List<TagItem>> updateAction = Update_Total;
                //  调用异步更新
                Dispatcher.BeginInvoke(updateAction, label_total, DataGrid_Inventory, text, manager.GetList());
            }
            break;
        default:
            break;
    }
}
```

以上响应接口会收到 **obj** 对象，就是**解析之后**的对象.（不同指令，返回不同对象）

TagItem 对象对应的就是一个标签. 基本结构如下：

```
public class TagItem
{
    /// * 序号
    public int index;

    /// * 信号强弱(1 byte)
    public int rssi;

    /// * PC-通讯协议(2 bytes)
    public ushort pc;

    /// * 校验码(2 bytes)
```

```
    public uint crc;

    /// * EPC-电子标签码(一般是12 bytes,最大60bytes)
    public byte[] epc;

    /// * RFU 区域
    public byte[] rfu;

    /// * TID 区域-唯一码
    public byte[] tid;

    /// * USER 区域-用户区
    public byte[] user;

    /// * 天线 (1byte，32bits)
    public int ant;

    /// * 累计
    public int count;

    /// * 激活时间
    public long time;

    /// * 首次时间
    public long first;

    /// * 关联信息
    public object tag;

}
```

rfidManager 直接获取标签列表调用如下接口

```
public List<TagItem> GetList()
{
    return listTags;
}
```

读写器对象有 4 个模块组成,对应常见的处理.

```
/// * 应答
public BaseAck ack;

/// * 命令
public BaseCmd cmd;

/// * 错误
public BaseErr err;

/// * 设置
public BaseSet set;
```

发送命令使用 cmd,
接收解析使用 ack,自动完成.
错误消息使用 err,自动完成.
设置使用 set.

注意多设备接口时,需要选择设备号,比如 COM1,COM2,COM3 都有效时,需要指定
举例:端口 2,波特率 115200.

```
UartPort.Config(2,115200);
```

开发包架构所有系统版本基本相同,只是一些大小写会有差别!

作者:施探宇
微信:18680399436
邮箱:Alecksty@163.com
网站:http://www.ts-rfid.com
深圳探索智能科技有限公司
2019 年 9 月 20 日