UHF RFID API

Reference Manual

Net&Serial

V4.2

2017 年 12 月

目录

1. 引言

本文档适合于采用本公司超高频模块进行 UHF 开发和测试的软件开发人员和软件测试人员。API 接口只能工作在本公司生产的超高频模块上。

2. API 简介

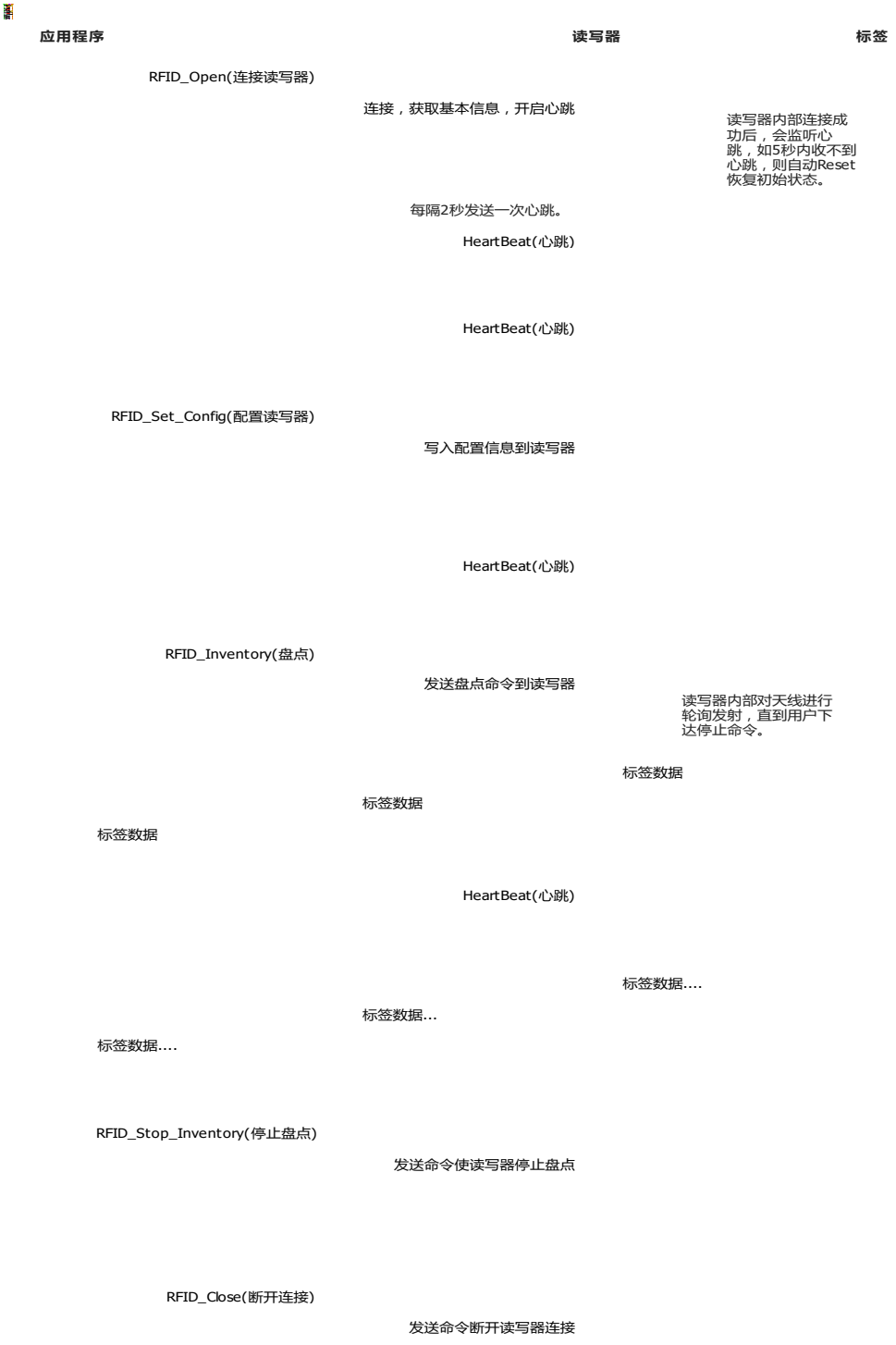
2.1 API 动态库列表

本公司提供一个 DLL 动态库。
JW.UHF.dll c#版本
SDK 内部日志打印采用 log4net，如需打印 sdk 通讯日志，建议加入该组件包，并启用日志。Sdk 通讯日志级别为 DEBUG.开发阶段和试运行阶段建议启用日志。

2.2 创建一个 C#工程

通过 VisioStudio 2005 以上版本创建一个 C#工程，
1 在工程中引用 JW.UHF.dll

2.3 API 调用时序图



3.模块初始化

模块初始化通过 new 一个 JWReader 对象，所有针对模块的调用均包含在该对象内部。

```
JWReader jwReader = new JWReader("COM3");
```

其中"COM3"为模块通过串口线连接 PC 后 PC 端对应的串口号。

```
JWReader jwReader = new JWReader("10.10.10.138",9761);
```

其中"10.10.10.138"为模块通过网口连接后的 IP 地址，9761 为端口号。

4.模块连接关闭

4.1 连接

4.1.1 定义

```
public Result RFID_Open()
public Result RFID_Open(bool enableHeartBeat)
```

建立与模块的连接,连接之前模块必须初始化。

4.1.2 参数

参数	描述
enableHeartBeat	是否启用上位机和下位机之间的心跳 True 启用 系统上线使用 False 禁用 开发调试阶段可禁用 默认 True

4.1.3 返回

Result.OK

其它 Result 参考 10.1 结果码。

4.1.4 例程

```
Result result = jwReader.RFID_Open();
if (result != Result.OK)
{
```

```
    //连接失败
}
//连接成功
```

4.2 断开

4.2.1 定义

```
public Result RFID_Close()
```

断开与模块的连接

4.2.2 参数

无

4.2.3 返回

Result.OK

其它 Result 参考 10.1 结果码。

4.2.4 例程

```
Result result =jwReader.RFID_Close();
if (result != Result.OK)
{
    //断开失败
}
//断开成功
```

5.模块配置

5.1 天线

5.1.1 定义

天线端口

```
public class AntennaPort
{
        public int AntennaIndex;
        public int Power;
        public bool Exist
}
```

5.1.2 参数

参数	描述
AntennaIndex	天线端口号
Power	发射功率
Exist	只读属性，天线是否存在

5.2 设置天线

5.2.1 定义

```
public Result RFID_Set_Antenna(List<AntennaPort> AntennaPortList)
```

5.2.2 参数

参数	描述
List<AntennaPort> AntennaPortList	天线端口列表,AntennaPort 定义参考 5.1.1

5.2.3 例程

```
//连接模块

List<AntennaPort> antennaPortList = new List<AntennaPort>();
AntennaPort ap = new AntennaPort();
ap.AntennaIndex = 1;//天线1
ap.Power = 27;//功率设为27
antennaPortList.Add(ap);
result = jwReader.RFID_Set_Antenna(antennaPortList);
if (result == Result.OK)
     Console.WriteLine("Antenna Set Success");
else
     Console.WriteLine("Antenna Set Failure");

//断开模块
```

5.3 SearchMode

读取模式

```
public enum SearchMode
{
     DUAL_TARGET = 0,// A-B和B-A状态标签均读取
     SINGLE_TARGET = 1,//仅读取A-B状态标签或仅读取B-A状态标签
     SINGLE_TARGET_WITH_SUPPRESSION = 2, //仅读取A-B状态标签或仅读取B-A状态标签,只读一次
}
```

5.4 SessionTarget

目标标签状态

```
public enum SessionTarget
{
     A = 0,//盘点A状态标签
     B = 1//盘点B状态标签
}
```

5.5 Session

会话形式

```
public enum Session
```

```
{
        S0 = 0,//标签在被盘点到后立即改变状态 如A状态到B状态后立即转为A状态
        S1 = 1,//标签在被盘点到停留0.5-5秒改变状态, 如A状态到B状态停留0.5-5秒后转为A状态
        S2 = 2,//标签在被盘点到停留5-60秒改变状态, 如A状态到B状态停留5-60秒后转为A状态
        S3 = 3 //与S2相同
}
```

5.6 标签会话参数

标签会话参数

5.6.1 定义

```
public class TagGroup
{
        public Session Session;
        public SessionTarget SessionTarget;
        public SearchMode SearchMode;
}
```

5.6.2 参数

参数	描述
SearchMode	读取模式,参考 5.3
SessionTarget	目标 Tag, 参考 5.4
Session	会话模式,参考 5.5

5.7 设置标签会话

5.7.1 定义

```
public Result RFID_Set_Tag_Group(TagGroup tagGroup)
```

5.7.2 参数

参数	描述

tagGroup	标签会话参数,参考 5.6

5.7.3 例程

```
//连接模块
#region 设置标签会话参数
TagGroup tg = new TagGroup();
tg.SearchMode = SearchMode.DUAL_TARGET;
tg.Session = Session.S0;
tg.SessionTarget = SessionTarget.A;
result = jwReader.RFID_Set_Tag_Group(tg);
if (result == Result.OK)
     Console.WriteLine("TagGroup Set Success");
else
     Console.WriteLine("TagGroup Set Failure");
#endregion

//断开模块
```

5.8 读取速率

读取速率

```
public enum SpeedMode
{
     SPEED_FASTEST = 0,//最快速度读取
      SPEED_NORMAL = 1, //正常速度读取
      SPEED_POWERSAVE = 2//省电模式读取
     SPEED_FULL_POWER=3 //满电工作 不建议长时间工作状态下使用
}
```

5.9 设置读取速率

5.9.1 定义

```
public Result RFID_Set_SpeedMode(SpeedMode speedMode)
```

5.9.2 参数

参数	描述

speedMode	读取速率参数,参考 5.8

5.9.3 例程

```
//连接模块
#region 设置读取速率
SpeedMode sm = SpeedMode.SPEED_FASTEST;//快速模式
result = jwReader.RFID_Set_SpeedMode(sm);
if (result == Result.OK)
    Console.WriteLine("Speed  Mode Set Success");
else
    Console.WriteLine("Speed Mode Set Failure");
#endregion
//断开模块
```

5.10 工作频段

工作频段

```
public enum RegionList
{
     FCC = 0,//欧洲频段902-928
     CCC = 1,//中国频段920-925
     NCC = 2,//台湾频段920-928
     OPTIMAL=3 //最优频段
}
```

5.11 设置工作频段

5.11.1 定义

```
public Result RFID_Set_RegionList(RegionList region)
```

5.11.2 参数

参数	描述
region	工作频段,参考 5.10

5.11.3 例程

```
//连接模块
#region 设置工作频段
RegionList rl = RegionList.CCC;//CCC频段
result = jwReader.RFID_Set_RegionList(rl);
if (result == Result.OK)
    Console.WriteLine("RegionList Set Success");
else
    Console.WriteLine("RegionList Set Failure");
#endregion
//断开模块
```

5.12 设置盘点时间

5.12.1 定义

```
public Result RFID_Set_Inventory_Time(int invTime)
```

5.12.2 参数

参数	描述
invTime	盘点时间(MS),在盘点 invTime 毫秒后就停止。 0 代表持续盘点

5.12.3 例程

```
//连接模块
#region 设置盘点时间
result = jwReader.RFID_Set_Inventory_Time(1000);//1秒停止盘点
if (result == Result.OK)
    Console.WriteLine("Inventory Time Set Success");
else
{
     Console.WriteLine("Inventory Time Set Failure");
     goto Exit;
}
#endregion
//断开模块
```

5.13 RSSI 过滤参数

5.13.1 定义

```
public class RSSIFilter
{
    public bool Enable;
    public float RSSIValue;
}
```

5.13.2 参数

参数	描述
Enable	是否启用过滤
RSSIValue	RSSI 值

5.14 设置 RSSI 过滤参数

5.14.1 定义

```
public Result RFID_Set_RSSIFilter(RSSIFilter rssiFilter)
```

5.14.2 参数

参数	描述
rssiFilter	Rssi 过滤参数，参考 5.17

5.14.3 例程

```
//连接模块
#region 设置RSSI 过滤
RSSIFilter rssiFilter=new RSSIFilter();
rssiFilter.Enable =true;
rssiFilter.RSSIValue = (float)-50.7;
```

```
result = jwReader.RFID_Set_RSSIFilter(rssiFilter);
if (result == Result.OK)
    Console.WriteLine("RSSI Filter Set Success");
else
    Console.WriteLine("RSSI Filter Set Failure");
#endregion
//断开模块
```

5.15 GPIO 事件

5.15.1 定义

```
public enum GPITriggerValue
{
      None=0,//不做动作
      Inventory=1,//盘点
      Input=2, //输入信号
      Hign=3, //将其转变为GPO 高电平输出
      Low=4   //将其转变为GPO 低电平输出
}

public enum GPOTriggerValue
{
      Low=0, //低电平
      Hign=1 //高电平
}

public enum GPIEventType{
        Start_Inventory=0,//启动盘点事件
        Stop_Inventory=1,//停止盘点事件
        Hign_Slow=2,//由高电平到低电平
        Slow_Hign=3 //由低电平到高电平
}

public class GPIEvent
{
        public int Port; //GPI 口
        public GPIEventType EventType; //GPI事件类型
        public GPITriggerValue TriggerValue;//GPI 触发器当前值
}
```

5.15.2 参数

参数	描述
Port	GPI 口
EventType	GPI事件类型
TriggerValue	GPI 触发器当前值

5.16 GPIO 配置

5.16.1 定义

```
public class GPIOConfig
{
        public GPITriggerValue GPI0_VALUE; //GPI0 触发器值
        public GPITriggerValue GPI1_VALUE; //GPI1 触发器值
        public GPOTriggerValue GPO0_VALUE; //GPO0 触发器值
        public GPOTriggerValue GPO1_VALUE; //GPO1 触发器值
}
public Result RFID_Set_GPIO(GPIOConfig gpioConfig)
```

5.16.2 参数

参数	描述
gpioConfig	GPIO 配置事件

5.16.3 例程

```
//连接模块
#region 设置 GPIO Trigger
GPIOConfig gpioConfig = new GPIOConfig();
gpioConfig.GPI0_VALUE = GPITriggerValue.Input;
gpioConfig.GPI1_VALUE = GPITriggerValue.None;
gpioConfig.GPO0_VALUE = GPOTriggerValue.Hign;
gpioConfig.GPO1_VALUE = GPOTriggerValue.Hign;
result = jwReader.RFID_Set_GPIO(gpioConfig);
#endregion
//断开模块
```

5.16.4 注意事项

必须停止盘点任务

5.17 模块参数

5.17.1 定义

```
public class RfidSetting
{
     public List<AntennaPort> AntennaPort_List;
     public List<GPIOTrigger> GPIO_Trigger_List;
     public TagGroup Tag_Group;
     public RSSIFilter RSSI_Filter;
     public InventoryMode Inventory_Mode;
     public SpeedMode Speed_Mode;
     public int Inventory_Time;
     public RegionList Region_List;
     public WorkMode Work_Mode;
}
```

5.17.2 参数

参数	描述
AntennaPort_List	天线列表，AntennaPort 定义参考 5.1
GPIO_Trigger_List	GPIO 触发器列表,GPIO 触发器参考 5.19
Tag_Group	标签会话参数,参考 5.6
RSSI_Filter	RSSI 过滤参数，参考 5.17
Inventory_Mode	盘点模式，参考 5.29
Inventory_Time	盘点时间(S),盘点多长时间停止，0 代表持续盘点
Speed_Mode	读取速率,参考 5.8
Region_List	工作频段，参考 5.10
Work_Mode	工作模式,参考 5.12

5.18 获取模块配置

5.18.1 定义

```
Result RFID_Get_Config(RfidSetting setting)
```

5.18.2 参数

参数	描述
setting	模块配置参数,参考 5.21

5.18.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

5.18.4 例程

```
#region 获取模块配置
Console.WriteLine("Get RFID Config");
RfidSetting rs = null;
result = jwReader.RFID_Get_Config(out rs);
if (result == Result.OK)
{
     foreach( AntennaPort apPort in rs.AntennaPort_List){
          Console.WriteLine(String.Format("AntennaPort={0},Power={1},Exist={2}",
apPort.AntennaIndex, apPort.Power, apPort.Exist));
     }
     foreach (GPIOTrigger gpioTrigger in rs.GPIO_Trigger_List){
          Console.WriteLine(String.Format("GPIOPort={0},GPIOValue={1}",
gpioTrigger.GPIO_PORT, gpioTrigger.GPIO_VALUE));
     }

      Console.WriteLine("SearchMode={0},Session={1},SessionTarget={2}",
rs.Tag_Group.SearchMode, rs.Tag_Group.Session, rs.Tag_Group.SessionTarget);
      Console.WriteLine("InventoryMode={0},InventoryTime={1}", rs.Inventory_Mode,
rs.Inventory_Time);
      Console.WriteLine("RSSIEnable={0},RSSIValue={1}", rs.RSSI_Filter.Enable,
rs.RSSI_Filter.RSSIValue);
```

```
        Console.WriteLine("SpeedMode={0},RegionList={1},WorkMode={2}", rs.Speed_Mode,
rs.Region_List,rs.Work_Mode);
}
else
        Console.WriteLine("RFID Config Set Failure");
#endregion
```

5.19 更新模块配置

5.19.1 定义

```
Result RFID_Set_Config(RfidSetting setting)
```

5.19.2 参数

参数	描述
setting	模块配置参数，参考 5.21

5.19.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

5.19.4 例程

```
//连接模块
#region 配置模块
RfidSetting rs=new RfidSetting();
rs.AntennaPort_List =new List<AntennaPort>();

AntennaPort ap = new AntennaPort();
ap.AntennaIndex = 1;//天线1
ap.Power = 27;//功率设为27
rs.AntennaPort_List.Add(ap);

rs.GPIO_Trigger_List = null;

rs.Inventory_Time = 10;
```

```
rs.Region_List = RegionList.CCC;

rs.RSSI_Filter = new RSSIFilter();
rs.RSSI_Filter.Enable = true;
rs.RSSI_Filter.RSSIValue = (float)-50.7;

rs.Speed_Mode = SpeedMode.SPEED_FASTEST;

rs.Tag_Group = new TagGroup();
rs.Tag_Group.SessionTarget = SessionTarget.A;
rs.Tag_Group.SearchMode = SearchMode.DUAL_TARGET;
rs.Tag_Group.Session = Session.S0;

result = jwReader.RFID_Set_Config(rs);
if (result == Result.OK)
    Console.WriteLine("RFID Config Set Success");
else
    Console.WriteLine("RFID Config Set Failure");
#endregion

//断开连接
```

5.20 设置定频

5.20.1 定义

```
public Result RFID_Set_Fix_Frequency(double frequency)//仅支持 902-928 频段的频率
public Result RFID_Set_Fix_Frequency(int port, int power, double frequency)//支持 840-960 任意频率
```

5.20.2 参数

参数	描述
port	天线口
power	天线输出功率 0-30
frequency	频率值 如 922.25 922.75 等

5.20.3 例程

```
//连接模块
#region 设置定频;
result = jwReader.RFID_Set_Fix_Frequency(922.25);
if (result == Result.OK)
    Console.WriteLine("Frequency Set Success");
else
    Console.WriteLine("Frequency Set Failure");
#endregion
//断开模块
```

5.21 测试最优频道参数

5.21.1 定义

```
/// <summary>
/// 频道性能
/// </summary>
public class ChannelPerformance
{
        /// <summary>
        /// 天线端口
        /// </summary>
        public int Antenna_Port;

        /// <summary>
        /// 天线功率
        /// </summary>
        public int Antenna_Power;

        /// <summary>
        /// 反向功率值
        /// </summary>
        public int Reverse_Power_Value;
}
```

5.21.2 参数

参数	描述

Antenna_Port	天线口
Antenna_Power	天线功率
Reverse_Power_Value	反向功率限定值

5.22 测试最优频道

5.22.1 定义

```
public Result RFID_Test_Channel(ChannelPerformance cpf,out List<int> channelList)
```

5.22.2 参数

参数	描述
cpf	最优频道参数，参考 5.21
channelList	最优频道列表

5.22.3 例程

```
//连接模块
List<int> channelList = null;
ChannelPerformance cpf = new ChannelPerformance();
cpf.Antenna_Port = 0;
cpf.Antenna_Power = 30;
cpf.Reverse_Power_Value = 10;
Result result = jwReader.RFID_Test_Channel(cpf,out channelList);
if (result == Result.OK)//测试完成
{

}
//断开模块
```

5.23 获取最优频道

5.23.1 定义

```
public Result RFID_Get_Optimal_Channel(out List<int> channelList)
```

5.23.2 参数

参数	描述
channelList	最优频道列表

5.23.3 例程

```
//连接模块
List<int> channelList = null;
Result result = jwReader.RFID_Get_Optimal_Channel(out channelList);;
if (result == Result.OK)//最优频道列表获取完成
{

}
//断开模块
```

5.24 GPO 配置

5.24.1 定义

```
public Result RFID_Set_GPO(GPIOConfig gpioConfig)
```

5.24.2 参数

参数	描述
gpioConfig	GPIO 配置事件

5.24.3 例程

```
//连接模块
#region 设置 GPO Trigger
GPIOConfig gpioConfig = new GPIOConfig();
gpioConfig.GPO0_VALUE = GPOTriggerValue.Hign;
gpioConfig.GPO1_VALUE = GPOTriggerValue.Hign;
result = jwReader.RFID_Set_GPO(gpioConfig);
#endregion
//断开模块
```

5.25 GPI 配置

5.25.1 定义

```
public Result RFID_Set_GPI(GPIOConfig gpioConfig)
```

5.25.2 参数

参数	描述
gpioConfig	GPIO 配置事件

5.25.3 例程

```
//连接模块
#region 设置 GPI Trigger
GPIOConfig gpioConfig = new GPIOConfig();
gpioConfig.GPIO_VALUE = GPITriggerValue.Input;
gpioConfig.GPI1_VALUE = GPITriggerValue.None;
result = jwReader.RFID_Set_GPI(gpioConfig);
#endregion
//断开模块
```

5.25.4 注意事项

必须停止盘点任务

5.26 获取模块温度

5.26.1 定义

```
public Result RFID_Get_Temperature(out int temperature)
```

5.26.2 参数

参数	描述
temperature	模块温度

5.26.3 例程

```
//连接模块
#region 获取模块温度
int temperature=0;
result = jwReader.RFID_Get_Temperature(out temperature);
#endregion
//断开模块
```

5.26.4 注意事项

必须停止盘点任务

5.27 检查天线

5.27.1 定义

```
public Result RFID_Check_Antenna(int antennaIndex)
```

5.27.2 参数

参数	描述
antennaIndex	天线索引

5.27.3 例程

```
//连接模块
#region 检查天线0状况
result = jwReader.RFID_Check_Antenna(0);
#endregion
//断开模块
```

5.28 设置天线驻留时间

5.28.1 定义

```
public Result RFID_Set_DWellTime(int dwellTime)
```

5.28.2 参数

参数	描述
dwellTime	天线驻留时间(Ms)

5.28.3 例程

```
//连接模块
#region 设置天线驻留300ms
result = jwReader. RFID_Set_DWellTime(300);
#endregion
//断开模块
```

5.29 设置盘点模式

5.29.1 定义

```
/// <summary>
/// 盘点模式
/// </summary>
public enum InventoryMode
{
        /// <summary>
        /// 持续盘点
        /// </summary>
        Continue = 0,//持续盘点

        /// <summary>
        /// 仅一个
        /// </summary>
```

```
        OnlyOne = 1//盘点到一个标签即停止
}

public Result RFID_Set_Inventory_Mode(InventoryMode mode)
```

5.29.2 参数

参数	描述
mode	盘点模式

5.29.3 例程

```
//连接模块
#region 设置读写器为持续盘点模式
result = jwReader.RFID_Set_Inventory_Mode(InventoryMode.Continue);
#endregion
//断开模块
```

5.30 获取模块 Firmware 信息

5.30.1 定义

```
/// <summary>
/// 分位信息
/// </summary>
public class FirmwareInfo
{
        /// <summary>
        /// 主版本号
        /// </summary>
        public int Main_Version//主版本号
        /// <summary>
        /// 副版本号
        /// </summary>
        public int Sub_Version //副版本号
}
public Result RFID_Get_Firmware_Info(out FirmwareInfo firmwareInfo)
```

5.30.2 参数

参数	描述
firmwareInfo	分位信息

5.30.3 例程

```
//连接模块
#region
FirmwareInfo firmwareInfo = new FirmwareInfo();
result = jwReader.RFID_Get_Firmware_Info(out firmwareInfo);
#endregion
//断开模块
```

5.31 设置天线（必须调用了 Save_Config 之后）

5.31.1 定义

```
/// <summary>
/// 设置天线 天线功率为上次调用Save_Config后保存的值
/// </summary>
public Result RFID_Set_Antenna(params int[] ports)
```

5.31.2 参数

参数	描述
ports	欲启用的天线 0,1,2,3

5.31.3 例程

```
//连接模块
#region 设置天线0,2启用
result = jwReader.RFID_Set_Antenna(0,2);
#endregion
//断开模块
```

5.31.4 注意事项

该方法为启用相应天线，天线功率为读写器上次保存到设备的功率值。
即必须先通过 RFID_Set_Config 去设定，然后通过 Save_Config 保存设定,再调用 RFID_Reset 重启令设备生效。
下次启用天线端口，不修改功率的话 就可以只提供欲启用的天线端口即可。

5.32 设置 Profile

5.32.1 定义

```
/// <summary>
/// 设置Profile 射频参数相关 一般默认即可
/// </summary>
public Result RFID_Set_Profile(int profile)
```

5.32.2 参数

参数	描述
profile	Profile 值可设值为 0,1,2,3 默认为 1

5.32.3 例程

```
//连接模块
#region 设置
result = jwReader.RFID_Set_Profile(1);
#endregion
//断开模块
```

5.33 获取 Profile

5.33.1 定义

```
/// <summary>
```

```
/// 获取Profile 射频参数相关
/// </summary>
public Result RFID_Get_Profile(out int profile)
```

5.33.2 参数

参数	描述
profile	Profile 值

5.33.3 例程

```
//连接模块
#region 设置
int profile = 0;
result = jwReader.RFID_Get_Profile(out profile);
#endregion
//断开模块
```

5.34 获取天线驻留时间

5.34.1 定义

```
/// <summary>
/// 获取天线驻留时间
/// </summary>
public Result RFID_Get_DWellTime(out int dwellTime)
```

5.34.2 参数

参数	描述
dwellTime	天线驻留时间

5.34.3 例程

```
//连接模块
#region 设置
int dwelltime = 0;
result = jwReader.RFID_Get_DWellTime (out dwelltime);
```

```
#endregion
//断开模块
```

5.35 设置是否启用 Antenna Hub

5.35.1 定义

```
/// <summary>
/// 设置是否启用Antenna Hub
/// </summary>
public Result RFID_Set_AntennaHub(int status)
```

5.35.2 参数

参数	描述
status	Antenna Hub 是否启用 0 禁用 1 启用

5.35.3 例程

```
//连接模块
#region 设置
result = jwReader.RFID_Set_AntennaHub(1);//启用
#endregion
//断开模块
```

注意事项:
仅 E 方

5.36 获取是否启用 Antenna Hub

5.36.1 定义

```
/// <summary>
/// 获取是否启用Antenna Hub
/// </summary>
public Result RFID_Get_AntennaHub(out int status)
```

5.36.2 参数

参数	描述
status	Antenna Hub 启用状态 0 禁用 1 启用

5.36.3 例程

```
//连接模块
#region 设置
int status=-1
result = jwReader.RFID_Get_AntennaHub(out status);
#endregion
//断开模块
```

6. 标签操作

6.1 标签响应模式

标签响应模式

```
public enum TagMode
{
   More=0,//一次盘点可以响应多次
   One=1  //一次盘点仅响应一次  目前仅支持EPC 96bit 1000个标签以内。
}
```

6.2 标签定义

6.2.1 定义

```
public class Tag
{
     public String DATA;
     public String EPC;
     public float RSSI;
     public int PORT;
}
```

6.2.2 参数

参数	描述
DATA	读到数据（快读 FAST_READ 返回）
EPC	当前 EPC 值
PORT	天线端口号
RSSI	标签返回的 RSSI 值

6.3 标签响应事件

6.3.1 定义

```
public class TagsEventArgs:EventArgs
{

    public Tag tag;//标签信息
    public String errorCode;//错误码
}
```

6.3.2 参数

参数	描述
tag	当前 Tag,参考 6.2

6.4 盘点回调函数

6.4.1 定义

```
public delegate void TagsEventHandler(object sender, TagsEventArgs args)
```

6.4.2 参数

参数	描述
sender	上下文句柄对象
args	标签响应事件，参考 6.3

6.4.3 注意事项

回调函数内部不能包含耗时操作，如更新界面，写数据库等。所有耗时操作请通过其它线程完成

6.5 内存区域

内存区域

```
public enum MemoryBank
{
        RESERVED = 0,// RESERVED区
        EPC = 1,//EPC区
        TID = 2,//TID区
        USER = 3//USER区
}
```

6.6 存取参数

6.6.1 定义

```
public class AccessParam
{
    public string AccessPassword;
    public MemoryBank Bank;
    public int OffSet;
    public int Count;
}
```

6.6.2 参数

参数	描述
Bank	内存区域
Count	读取字节数（Byte）以偶数字节数(Word)形式读
AccessPassword	存取密码
Offset	偏移量（Byte）

6.7 过滤条件参数

6.7.1 定义

```
public class RfidCriteria
{
    public MemoryBank Bank;
    public int Count;
    public byte[] Mask;
    public int Offset;
    public bool Match;
}
```

6.7.2 参数

参数	描述
Bank	内存区域，参考 6.5,其中的 RESERVED 不是有效值
Count	匹配字节数（Byte）0 到 31 之间
Offset	偏移量(Byte)
Mask	匹配数据 举例：如果你想获取从 EPC 值第 2 个 Byte 开始的 2 个 Byte 值为”ABCD”的标签。则对应值为: bank = MemoryBank.EPC offset = 2(前面32bit为长度等信息) count = 2 mask[0] = 0xAB mask[1] = 0xCD
Match	选择匹配的数据还是不匹配的数据 True :选择匹配数据 False:选择不匹配数据

6.8 增加过滤条件

6.8.1 定义

```
public Result RFID_Set_Criteria(RfidCriteria criteria)
```

6.8.2 参数

参数	描述
criteria	过滤条件参数，参考 6.7

6.8.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

6.8.4 例程

```
//连接模块
#region 设置Criteria
RfidCriteria criteria = new RfidCriteria();
criteria.Bank = MemoryBank.EPC;
criteria.OffSet = 0;
criteria.Mask = Util.ToHexByte("3008");
criteria.Count = 2;
criteria.Match = true;
result=jwReader.RFID_Set_Criteria(criteria);
if (result == Result.OK)
   Console.WriteLine("Set Criteria Success");
else
   Console.WriteLine("Set Criteria Failure");
#endregion
//断开连接
```

以上代码作用为仅选取 EPC 值从 0 开始，以”3008”(16 进制)开头的标签。

6.8.5 注意事项

过滤条件全局有效，包括盘点，读写锁灭活等。

6.9 移除过滤条件

6.9.1 定义

```
public Result RFID_Clear_Criteria()
```

6.9.2 参数

无

6.9.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

6.9.4 例程

```
//连接模块
jwReader.RFID_Clear_Criteria()
//断开连接
```

6.10 启动盘点

6.10.1定义

```
public Result RFID_Start_Inventory()
public void RFID_Start_Inventory(bool async)
```

6.10.2参数

参数	描述
async	是否异步盘点 True:异步 False:同步(默认)

6.10.3返回

同步模式下有返回值：
Result.OK
其它 Result 参考 10.1 结果码。

异步模式无返回值

6.10.4例程

参考附录10.2

6.11 停止盘点

6.11.1 定义

```
public Result RFID_Stop_Inventory()
```
终止盘点

6.11.2 参数

无

6.11.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

6.11.4 例程

```
//连接模块
jwReader.RFID_Stop_Inventory();
//断开连接
```

6.12 读

6.12.1 定义

```
public Result RFID_Read(AccessParam ap);
```

6.12.2 参数

参数	描述
ap	存取参数,参考 6.6, 最大支持 128 Bytes

6.12.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

6.12.4 例程

参考附录 10.2

6.13 写

6.13.1 定义

```
public Result RFID_Write(AccessParam ap, ushort[] writedata
```

6.13.2 参数

参数	描述
ap	存取参数,参考 6.6
writedata	写入数据最大支持 128 Bytes

6.13.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

6.13.4 例程

参考附录 10.3

6.13.5 注意事项

写模式下,修改 EPC 内容同时会修改 EPC 长度。

6.14 锁

6.14.1 定义

```
public Result RFID_Lock(string lockPwd)
```

6.14.2 参数

参数	描述
lockPwd	锁密码

6.14.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

6.14.4 例程

```
//连接模块
jwReader.RFID_Lock(0x11111111);
//断开连接
```

6.15 解锁

6.15.1 定义

```
public Result RFID_UnLock(string lockPwd);
```

6.15.2 参数

参数	描述
lockPwd	解锁密码

6.15.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

6.15.4 例程

```
//连接模块
jwReader.RFID_Unlock (0x11111111);
//断开连接
```

6.16 灭活

6.16.1 定义

```
public Result RFID_Kill(string accessPwd, string killPwd);
```

6.16.2 参数

参数	描述
accessPwd	存取密码
killPwd	灭活密码

6.16.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

6.16.4 例程

```
//连接模块
jwReader.RFID_Kill(0x11111111,0x11111111);
//断开连接
```

6.17 块写

6.17.1 定义

```
public Result RFID_BlockWrite(AccessParam ap, ushort[] writedata
```

6.17.2 参数

参数	描述
ap	存取参数,参考 6.6
writedata	写入数据 最大支持 128Byte

6.17.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

6.17.4 例程

参考附录 10.3

6.17.5 注意事项

块写模式下,仅修改 EPC 内容不修改 EPC 长度。

6.18 永久锁区域

```
public enum LockMemory
{
```

```
        EPC = 0,
        User = 1,
        All = 2
}
```

6.19 永久锁

6.19.1 定义

```
public Result RFID_PermLock(LockMemory lockMemory)
```

标签必须处于解锁状态下，才可操作

6.19.2 参数

参数	描述
lockMemory	永久锁区域

6.19.3 返回

Result.OK

其它 Result 参考 10.1 结果码。

6.19.4 例程

```
//连接模块
jwReader.RFID_PermLock(LockMemory.All);
//断开连接
```

6.20 永久解锁

6.20.1 定义

```
public Result RFID_PermUnLock(LockMemory lockMemory);
```

6.20.2 参数

参数	描述
lockMemory	永久解锁区域

6.20.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

6.20.4 例程

```
//连接模块
jwReader.RFID_PermUnLock(LockMemory.All);
//断开连接
```

6.21 群读

6.21.1定义

```
public Result RFID_GroupRead (AccessParam ap)
public Result RFID_GroupRead (AccessParam ap ,bool async)
```

6.21.2参数

参数	描述
ap	存取参数,参考 6.6
async	是否异步群读 True:异步 False:同步(默认)

6.21.3返回

同步模式下有返回值：
Result.OK
其它 Result 参考 10.1 结果码。

异步模式无返回值

6.21.4例程

参考附录10.2盘点例程

6.22 快读

6.22.1 定义

```
public Result RFID_Fast_Read(AccessParam ap, out Tag tag);
```

6.22.2 参数

参数	描述
ap	存取参数,参考 6.6
tag	返回标签数据

6.22.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

6.22.4 例程

参考附录 10.2 读例程

6.23 QT

```
public enum QT
{
        /// <summary>
        /// 公有区
        /// </summary>
```

```
        PUBLIC=0,
        /// <summary>
        /// 私有区
        /// </summary>
        PRIVATE=1
}
```

6.24 设置 QT 功能

6.24.1 定义

```
public Result RFID_Set_QT(QT qt);
```

6.24.2 参数

参数	描述
qt	QT 参数,参考 6.23

6.24.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

6.24.4 例程

```
#region 设置QT Public
Console.WriteLine("Set QT Public");
result = jwReader.RFID_Set_QT(QT.PUBLIC);
if (result == Result.OK)
    Console.WriteLine("Set QT Public Success");
else
  {
      Console.WriteLine("Set QT Public Failure");
  }
  #endregion
```

6.25 写 EPC(带偏移量不修改 EPC 长度)

6.25.1 定义

```
public Result RFID_WriteEPC_With_OffSet(AccessParam ap, ushort[] writedata
```

6.25.2 参数

参数	描述
Ap	存取参数,参考 6.6
Writedata	写入数据最大支持 128 Bytes

6.25.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

6.25.4 例程

参考附录 10.3

6.25.5 注意事项

按照偏移量修改 EPC,不修改 EPC 长度。

7. IP 相关操作

7.1 设置 IP 地址(仅 E 方可用)

7.1.1 定义

```
public Result IP_Set_Address(string ipAddress)
```

7.1.2 参数

参数	描述
ipAddress	Ip 地址 如 10.10.10.101

7.1.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

7.1.4 例程

```
//连接模块
jwReader.IP_Set_Address("10.10.10.101");
//断开连接
```

注意事项:
通过 9.3 保存 UHF 和 IP 相关信息 重启生效

7.2 设置 IP Port(仅 E 方可用)

7.2.1 定义

```
public Result IP_Set_Port(int port)
```

7.2.2 参数

参数	描述
port	网口端口 如 9761

7.2.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

7.2.4 例程

```
//连接模块
jwReader.IP_Set_Port (9761);
//断开连接
```

注意事项:
通过 9.3 保存 UHF 和 IP 相关信息 重启生效

7.3 设置子网掩码 SubNetMask(仅 E 方可用)

7.3.1 定义

```
public Result IP_Set_SubNet_Mask(string ipAddress)
```

7.3.2 参数

参数	描述
ipAddress	Ip 地址 如 255.255.255.0

7.3.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

7.3.4 例程

```
//连接模块
jwReader.IP_Set_SubNet_Mask("255.255.255.0");
//断开连接
```

注意事项:
通过 9.3 保存 UHF 和 IP 相关信息 重启生效

7.4 设置网关 GateWay(仅 E 方可用)

7.4.1 定义

```
public Result IP_Set_GateWay(string ipAddress)
```

7.4.2 参数

参数	描述
ipAddress	Ip 地址 如 10.10.10.1

7.4.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

7.4.4 例程

```
//连接模块
jwReader.IP_Set_GateWay("10.10.10.1");
//断开连接
```

注意事项:
通过 9.3 保存 UHF 和 IP 相关信息 重启生效

7.5 设置 DHCP(仅 E 方可用)

7.5.1 定义

```
public Result IP_Set_DHCP(bool enable)
```

7.5.2 参数

参数	描述
enable	启用 DHCP 还是禁用 DHCP True Or False

7.5.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

7.5.4 例程

```
//连接模块
jwReader.IP_Set_DHCP (true);
//断开连接
```

注意事项:
通过 9.3 保存 UHF 和 IP 相关信息 重启生效

7.6 IP 配置信息(仅 E 方可用)

7.6.1 定义

```
public class IPConfiguration
{
        /// <summary>
        /// 是否启用DHCP
        /// </summary>
        public bool Dhcp;

        /// <summary>
        /// IP地址
        /// </summary>
        public string IP;


        /// <summary>
        /// 端口
        /// </summary>
        public int Port;

        /// <summary>
        /// 子网掩码
        /// </summary>
```

```
        public string SubNet_Mask;

        /// <summary>
        /// 网关
        /// </summary>
        public string Gateway;
}
```

7.6.2 参数

参数	描述
Dhcp	是否启用 DHCP
IP	IP 地址
Port	天线端口
SubNet_Mask	子网掩码
Gateway	网关

注意事项:
通过 9.3 保存 UHF 和 IP 相关信息 重启生效

7.7 获取 IP 配置信息(仅 E 方可用)

7.7.1 定义

```
public Result IP_Get_Configuration(out IPConfiguration ipConfiguration)
```

7.7.2 参数

参数	描述
ipConfiguration	IP 配置信息

7.7.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

7.7.4 例程

```
//连接模块
IPConfiguration ip;
Result result = jwReader.IP_Get_Configuration(out ip);
if (result == Result.OK)
{
        //ip.IP;
        //ip.Port.ToString();
        //ip.SubNet_Mask;
        //ip.Gateway;
        //ip.Dhcp;
}
//断开连接
```

注意事项:
通过 9.3 保存 UHF 和 IP 相关信息 重启生效

7.8 更新 IP 配置信息(仅 E 方可用)

7.8.1 定义

```
public Result IP_Set_Configuration(IPConfiguration ipConfiguration)
```

7.8.2 参数

参数	描述
ipConfiguration	IP 配置信息

7.8.3 返回

Result.OK
其它 Result 参考 10.1 结果码。

7.8.4 例程

```
//连接模块
IPConfiguration ip = new IPConfiguration();
```

```
ip.IP = “10.10.10.121”;
ip.Port = 9761;
ip.SubNet_Mask = “255.255.255.0”;
ip.Gateway = “10.10.10.1”;
ip.Dhcp = true;

Result result = jwReader.IP_Set_Configuration(ip);
//断开连接
```

注意事项:
通过 9.3 保存 UHF 和 IP 相关信息 重启生效

8.跨网段搜索更新 IP

8.1 设备信息

8.1.1 定义

```
public class TcpInfo
{
        /// <summary>
        /// 产品类型 E L W
        /// </summary>
        public ProductType ProductType;

        /// <summary>
        /// 模块IP地址
        /// </summary>
        public string ModelIP;


        /// <summary>
        /// 模块端口
        /// </summary>
        public int ModelTcpPort;

        /// <summary>
        /// 子网掩码
        /// </summary>
```

```
        public string ModelSubNetMask;

        /// <summary>
        /// 网关
        /// </summary>
        public string ModelGateway;

        /// <summary>
        /// 网关
        /// </summary>
        public string ModelMacAddress;

        /// <summary>
        /// 是否启用
        /// </summary>
        public bool ModelDHCPEnable;

        /// <summary>
        /// 主机名
        /// </summary>
        public string HostName;

}
```

8.1.2 参数

参数	描述
ProductType	产品类型
ModelIP	模块 IP 地址
ModelTcpPort	模块 TCP 端口
ModelSubNetMask	模块子网掩码
ModelGateway	模块网关
ModelMacAddress	模块 MAC 地址
ModelDHCPEnable	模块 DHCP 是否启用
HostName	主机名

8.2 开始搜索设备

8.2.1 定义

```
public static void StartAllDiscovery()
```

8.2.2 参数

8.2.3 返回

8.2.4 例程

```
TcpDiscoveryFactory.StartAllDiscovery();
```

8.3 停止搜索设备

8.3.1 定义

```
public static void StopAllDiscovery()
```

8.3.2 参数

8.3.3 返回

8.3.4 例程

```
TcpDiscoveryFactory.StopAllDiscovery();
```

8.4 更新 IP 地址信息

8.4.1 定义

```
public bool UpdateIP(TcpInfo tcpInfo)
```

8.4.2 参数

参数	描述
tcpInfo	参考 8.1.1

8.4.3 返回

True 更新成功 False 更新失败

8.4.4 例程

```
TcpInfo tcpInfo = new TcpInfo();
tcpInfo.HostName = this.hostNameTb.Text;
tcpInfo.ModelDHCPEnable = this.enableRb.Checked;
tcpInfo.ModelGateWay = this.modelGateWayTb.Text;
tcpInfo.ModelMacAddress = this.macAddressTb.Text;
tcpInfo.ModelSubNetMask = this.modelSubNetMaskTb.Text;
tcpInfo.ModelTcpPort = Int32.Parse(this.modelPortTb.Text);
tcpInfo.ModelIP = this.modelIPTb.Text;
tcpInfo.ModelVersion = this.modelVersionTb.Text;
tcpInfo.ProductType = ProductType.L;
TcpDiscoveryFactory.UpdateIP(tcpInfo);
```

9.其它操作

9.1 产品信息

9.1.1 定义

```
public class ProductInfo
{
    public string COMPANY_NO;//公司编号
    public string PRODUCT_DATE;//生产日期
    public string MODEL_TYPE;//模块型号
    public int MODEL_VERSION;//模块版本
    public string MODEL_SEQUENCE_NUMBER;//模块序列号
    public int ANTENNA_NUMBER;//天线个数
```

```
    public List<AntennaPort> ANTENNA_PORT_EXIST_LIST; //天线存在列表
    public ProductType PRODUCT_TYPE;
}
```

9.1.2 参数

参数	描述
ANTENNA_NUMBER	天线个数
ANTENNA_PORT_EXIST_LIST	描述天线是否可用。利用 AntennaPort 的 Exist 属性
COMPANY_NO	公司编号
PRODUCT_DATE	生产日期
MODEL_TYPE	模块类型
MODEL_VERSION	模块版本
MODEL_SEQUENCE_NUMBER	模块序列号
PRODUCT_TYPE	产品类型

9.2 获取产品信息

9.2.1 定义

```
public Result RFID_Get_Product_Info(out ProductInfo productInfo)
```

获取当前连接模块的相关信息，序列号，天线个数，模块类型等。

9.2.2 参数

参数	描述
productInfo	产品信息类，参考 7.1

9.2.3 返回

Result.OK

其它 Result 参考 10.1 结果码

9.2.4 例程

```
//连接模块
Module_Info info = new Module_Info();
Result result = jwReader.RFID_Get_Product_Info(ref info);
//断开连接
```

9.3 保存配置

9.3.1 定义

```
public Result Save_Config()
```

保存 UHF 和 IP 等相关信息。

9.3.2 参数

参数	描述
productInfo	产品信息类，参考 7.1

9.3.3 返回

Result.OK

其它 Result 参考 10.1 结果码

9.3.4 例程

```
//连接模块
Result result = jwReader.Save_Config();
//断开连接
```

9.4 设置运行模式

9.4.1 定义

```
public enum RunningMode
{
        /// <summary>
        /// API模式 数据会以API解析后形式返回
        /// </summary>
        API=0,

        /// <summary>
        /// 命令模式 原始数据返回
        /// </summary>
        COMMAND=1
}
public void Set_Running_Mode(RunningMode _runMode)
```

设置运行模式

9.4.2 参数

参数	描述
_runMode	运行模式

9.4.3 返回

9.4.4 例程

```
//连接模块
jwReader.Set_Running_Mode(RunningMode.COMMAND);
//断开连接
```

9.5 原始响应数据回调函数

9.5.1 定义

```
public delegate void CommandResponseEventHandler(JWReader reader, byte[] responseData);
```

9.5.2 参数

参数	描述
reader	读写器对象
responseData	响应数据

9.5.3 返回

9.5.4 例程

```
//连接模块
jwReader.commandResponseEventReported += CommandResponseEventReport;
//断开模块

private void CommandResponseEventReport(object sender, byte[] data)
{
      //处理响应数据
}
```

9.6 打开蜂鸣器

9.6.1 定义

```
public Result Open_Buzzer();
```

9.6.2 参数

9.6.3 返回

9.6.4 例程

```
//连接模块
jwReader.Open_Buzzer();//蜂鸣器响一声
//断开模块
```

10. 附录

10.1 结果码 Result

```
public enum Result
{
        /// <summary>
        /// OK
        /// </summary>
        OK = 0,

        /// <summary>
        /// 设备忙
        /// </summary>
        Module_Is_Busy = 1,//模块Busy

        /// <summary>
        /// 设备已经关闭
        /// </summary>
        Module_Is_Closed = 2,//模块是关闭的

        /// <summary>
        /// 设备已经打开
        /// </summary>
        Module_Is_Already_Opened = 3,//模块已经打开

        /// <summary>
        /// 设备无响应
        /// </summary>
        Model_Not_Response = 4,//模块无响应

        /// <summary>
        /// 发送指令失败
        /// </summary>
        Send_Instruct_Failure = 5,

        /// <summary>
        /// 响应失败
```

```
        /// </summary>
        Response_TimeOut = 6,

        /// <summary>
        /// 连接失败
        /// </summary>
        Connect_Failure = 7,

        /// <summary>
        /// 断开连接失败
        /// </summary>
        Disconnect_Failure = 8,

        /// <summary>
        /// 反响功率过高
        /// </summary>
        Reverse_Power_Too_Hign = 9,

        /// <summary>
        /// SDK忙
        /// </summary>
        Device_Is_Busy = 10,

        /// <summary>
        /// 写数据失败
        /// </summary>
        Write_Data_Is_Null = 11,

        /// <summary>
        /// 密码是空的
        /// </summary>
        Pwd_Is_Null = 12,

        /// <summary>
        /// 密码长度错误
        /// </summary>
        Pwd_Length_Is_Error = 13,

        /// <summary>
        /// 天线不存在
        /// </summary>
        Antenna_Not_Exists = 14,

        /// <summary>
```

```
        /// SDK等待超时
        /// </summary>
        Wait_TimeOut = 15,

        /// <summary>
        /// 读到数据为空
        /// </summary>
        Read_Data_Is_Empty = 16,

        /// <summary>
        /// SDK连接超时
        /// </summary>
        Sdk_Connect_TimeOut=17,

        /// <summary>
        /// 接口不可用
        /// </summary>
        Interface_Not_Avaliable=18,

        /// <summary>
        /// 天线未配置
        /// </summary>
        Antenna_Not_Configure=19,


        /// <summary>
        /// 频道不可用
        /// </summary>
        Channel_Not_Supported = 20,


        /// <summary>
        /// 写数据太长
        /// </summary>
        Write_Data_Too_Long=21,

        /// <summary>
        /// 网络异常
        /// </summary>
        Network_Exception=22,

        /// <summary>
        /// 接收缓冲区溢出
        /// </summary>
```

```
        Receive_Buffer_OverFlow=23,

        /// <summary>
        /// 解析盘点包错误
        /// </summary>
        Parse_Packet_Data_Error=24,

        /// <summary>
        /// 串口异常
        /// </summary>
        Serial_Exception=25,

        /// <summary>
        /// 天线不存在
        /// </summary>
        Antenna_Not_Connected = 26,

        /// <summary>
        /// 前向功率不足
        /// </summary>
        Forward_Power_InSufficient=27,

        /// <summary>
        /// 未知异常
        /// </summary>
        Unknown_Exception = 99

}
```

10.2 盘点操作例程完整代码

```
using System;
using System.Collections.Generic;
using System.Text;
using JW.UHF;
namespace Inventory
{
    class Program
    {
        /// <summary>
        /// 数据上报
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="args"></param>
        private static void TagsReport(object sender, TagsEventArgs args)
        {
            Tag tag=args.tag;
            Console.WriteLine(string.Format("EPC={0},Port={1},RSSI={2}", tag.EPC, tag.PORT,
tag.RSSI));
        }

        static void Main(string[] args)
        {
            JWReader jwReader = new JWReader("COM3");
            //JWReader jwReader = new JWReader("10.10.10.121",9761);
            #region 打开模块
            Result result = jwReader.RFID_Open();
            if (result == Result.OK)
                Console.WriteLine("Open Module Success");
            else
            {
                Console.WriteLine("Open Module Failure");
                goto Exit;
            }
            #endregion

            #region 配置模块
            RfidSetting rs = new RfidSetting();
            rs.AntennaPort_List = new List<AntennaPort>();
            AntennaPort ap = new AntennaPort();
            ap.AntennaIndex = 1;//天线1
            ap.Power = 22;//功率设为27
            rs.AntennaPort_List.Add(ap);
```

```
            rs.GPIO_Trigger_List = null;
            rs.Inventory_Time = 5000;
            rs.Region_List = RegionList.CCC;
            rs.RSSI_Filter = new RSSIFilter();
            rs.RSSI_Filter.Enable = true;
            rs.RSSI_Filter.RSSIValue = (float)-70;
            rs.Speed_Mode = SpeedMode.SPEED_FASTEST;
            rs.Tag_Group = new TagGroup();
            rs.Tag_Group.SessionTarget = SessionTarget.A;
            rs.Tag_Group.SearchMode = SearchMode.DUAL_TARGET;
            rs.Tag_Group.Session = Session.S0;
            result = jwReader.RFID_Set_Config(rs);
            if (result == Result.OK)
                Console.WriteLine("RFID Config Set Success");
            else
            {
                Console.WriteLine("RFID Config Set Failure");
                goto Exit;
            }
            #endregion

            #region 盘点
            Console.WriteLine("Start Inventory");
             jwReader.TagsReported += TagsReport;
            jwReader.RFID_Start_Inventory();
            Console.WriteLine("Stop Inventory");
            #endregion

        Exit:
            #region 关闭模块
            result = jwReader.RFID_Close();
            if (result == Result.OK)
                Console.WriteLine("Close Module Success");
            else
            {
                Console.WriteLine("Close Module Failure");
            }
            #endregion
            Console.ReadLine();
        }
    }
}
```

10.3 读操作例程完整代码

```
using System;
using System.Collections.Generic;
using System.Text;
using JW.UHF;
namespace Read
{
    class Program
    {
        static void Main(string[] args)
        {
            JWReader jwReader = new JWReader("COM3");
            //JWReader jwReader = new JWReader("10.10.10.121",9761);
            #region 打开模块
            Result result = jwReader.RFID_Open();
            if (result == Result.OK)
                Console.WriteLine("Open Module Success");
            else
            {
                Console.WriteLine("Open Module Failure");
                goto Exit;
            }
            #endregion

            #region 配置模块
            RfidSetting rs = new RfidSetting();
            rs.AntennaPort_List = new List<AntennaPort>();
            AntennaPort ap = new AntennaPort();
            ap.AntennaIndex = 1;//天线1
            ap.Power = 22;//功率设为27
            rs.AntennaPort_List.Add(ap);
            rs.GPIO_Trigger_List = null;

            rs.Inventory_Time = 5;
            rs.Region_List = RegionList.CCC;
            rs.RSSI_Filter = new RSSIFilter();
            rs.RSSI_Filter.Enable = true;
            rs.RSSI_Filter.RSSIValue = (float)-70;
            rs.Speed_Mode = SpeedMode.SPEED_FASTEST;
                    rs.Tag_Group = new TagGroup();
            rs.Tag_Group.SessionTarget = SessionTarget.A;
            rs.Tag_Group.SearchMode = SearchMode.DUAL_TARGET;
            rs.Tag_Group.Session = Session.S0;
```

```
            result = jwReader.RFID_Set_Config(rs);
            if (result == Result.OK)
                Console.WriteLine("RFID Config Set Success");
            else
            {
                Console.WriteLine("RFID Config Set Failure");
                goto Exit;
            }
            #endregion

            #region 读
            Console.WriteLine("Start Read");
            AccessParam accessParam = new AccessParam();
            accessParam.Bank = MemoryBank.EPC;
            accessParam.OffSet = 0;
            accessParam.Count = 12;
            for (int i = 1; i <= 10; i++)
            {
                string tagData = "";
                result = jwReader.RFID_Read(accessParam, out tagData);
                if (result == Result.OK)
                    Console.WriteLine(string.Format("Count={0},EPC={1}",i,tagData));
                else
                    Console.WriteLine("Read Failure");
            }
            Console.WriteLine("Stop Read");
            #endregion

        Exit:
            #region 关闭模块
            result = jwReader.RFID_Close();
            if (result == Result.OK)
                Console.WriteLine("Close Module Success");
            else
            {
                Console.WriteLine("Close Module Failure");
            }
            #endregion
            Console.ReadLine();
        }
    }
}
```

10.4 写操作例程完整代码

```
using System;
using System.Collections.Generic;
using System.Text;
using JW.UHF;
namespace Write
{
    class Program
    {
        static void Main(string[] args)
        {
            JWReader jwReader = new JWReader("COM3");
            //JWReader jwReader = new JWReader("10.10.10.121",9761);

            #region 打开模块
            Result result = jwReader.RFID_Open();
            if (result == Result.OK)
                Console.WriteLine("Open Module Success");
            else
            {
                Console.WriteLine("Open Module Failure");
                goto Exit;
            }
            #endregion


            #region 配置模块
            RfidSetting rs = new RfidSetting();
            rs.AntennaPort_List = new List<AntennaPort>();
            AntennaPort ap = new AntennaPort();
            ap.AntennaIndex = 1;//天线1
            ap.Power = 10;//功率设为27
            rs.AntennaPort_List.Add(ap);
            rs.GPIO_Trigger_List = null;

            rs.Inventory_Time = 5;
            rs.Region_List = RegionList.CCC;
            rs.RSSI_Filter = new RSSIFilter();
            rs.RSSI_Filter.Enable = true;
            rs.RSSI_Filter.RSSIValue = (float)-70;
            rs.Speed_Mode = SpeedMode.SPEED_FASTEST;

            rs.Tag_Group = new TagGroup();
```

```
rs.Tag_Group.SessionTarget = SessionTarget.A;
rs.Tag_Group.SearchMode = SearchMode.DUAL_TARGET;
rs.Tag_Group.Session = Session.S0;

result = jwReader.RFID_Set_Config(rs);
if (result == Result.OK)
    Console.WriteLine("RFID Config Set Success");
else
{
    Console.WriteLine("RFID Config Set Failure");
    goto Exit;
}
#endregion

#region 设置Criteria
RfidCriteria criteria = new RfidCriteria();
criteria.Bank = MemoryBank.EPC;
criteria.OffSet = 0;
criteria.Mask = Util.ToHexByte("3008");
criteria.Count = 2;
criteria.Match = true;
result = jwReader.RFID_Set_Criteria(criteria);
if (result == Result.OK)
    Console.WriteLine("Set Criteria Success");
else
{
    Console.WriteLine("Set Criteria Failure");
    goto Exit;
}
#endregion

#region 写
Console.WriteLine("Start Write");

AccessParam accessParam = new AccessParam();
accessParam.Bank = MemoryBank.USER;
accessParam.OffSet = 0;
accessParam.Count = 2;


string writeData = "DCBA";
result = jwReader.RFID_Write(accessParam, writeData);
if (result == Result.OK)
    Console.WriteLine("Write Success");
```

```
            else
                Console.WriteLine("Write Failure");

            #endregion

            #region 读
            Console.WriteLine("Start Read");
            string readData = "";
            result = jwReader.RFID_Read(accessParam, out readData);
            if (result == Result.OK)
                Console.WriteLine(string.Format("Read Data User={0}",readData));
            else
                Console.WriteLine("Read Failure");

            #endregion

            #region 清除 Criteria
            result = jwReader.RFID_Clear_Criteria();
            if (result == Result.OK)
                Console.WriteLine("Clear Criteria Success");
            else
            {
                Console.WriteLine("Clear Criteria Failure");
                goto Exit;
            }
            #endregion

        Exit:
            #region 关闭模块
            result = jwReader.RFID_Close();
            if (result == Result.OK)
                Console.WriteLine("Close Module Success");
            else
            {
                Console.WriteLine("Close Module Failure");
            }
            #endregion
            Console.ReadLine();
        }
    }
}
```