Standard Passive Reader C# Dynamic Link Library

User's Guide V1.0

1. Operation System Requirement: ----- *2*

2. Guide: ----- *2*

3. Using steps: ----- *2*

3.1 Create a PassiveRcp class object ----- *2*

3.2 Load event ----- *2*

3.3 Connect ----- *2*

3.4 Communication ----- *2*

3.5 Disconnect ----- *2*

4. Event List: ----- *2*

5. Function List: ----- *2*

5.1 General Function: ----- *2*

5.2 Base Function: ----- *2*

5.3 EPCC1-G2 Function: ----- *3*

5.4 ISO18000-6B Function: ----- *3*

5.5 Senior Function: ----- *3*

6. Function Explanation ----- *4*

6.1 General Function: ----- *4*

6.2 Base Function: ----- *5*

6.3 EPCC1-G2 Function: ----- *7*

7. Struct Information ----- *9*

1. Operation System Requirement:

WINDOWS

2. Guide:

1. Need include ADDevice.dll, ADDeviceReader.dll;
2. Maybe can include ADBase.dll, it is not necessary;
3. The ADDeviceReader.dll use asynchronous communication, so you need add Event to receive received data;

3. Using steps:

3.1 Create a PassiveRcp class object

```
ADRcp SystemPub.ADRcp = new PassiveRcp();
```

3.2 Load event

```
SystemPub.ADRcp.RxRspParsed += RxRspEventReceived;
SystemPub.ADRcp.StatusConnected += ADRcp_StatusConnected;
```

3.3 Connect

```
SystemPub.ADRcp.Connect(hostOrPort, portOrBaud, commType);
```

3.4 Communication

```
PassiveCommand.GetInformation(SystemPub.ADRcp);       //get base information
PassiveCommand.GetConfig(SystemPub.ADRcp);      //get base Parameters
```

3.5 Disconnect

```
SystemPub.ADRcp.DisConnect();
```

4. Event List:

```
public event EventHandler<StringEventArg> RcpLogEventReceived; //Communication log
public event EventHandler<ProtocolEventArg> RxRspParsed; //Asynchronous receive data(Hex)
public event EventHandler<ConnectEventArg> StatusConnected;   //Communication status
```

5. Function List:

ADDeviceReader.dll includes the following functions：

5.1 General Function:

5.1.1 bool SystemPub.ADRcp.Connect(string hostOrPort, int portOrBaud, int type);

5.1.2 void SystemPub.ADRcp.DisConnect();

5.2 Base Function:

5.2.1 bool PassiveCommand.GetInformation(ADRcp adr);

5.2.2 bool PassiveCommand.GetConfig(ADRcp adr);

5.2.3 bool PassiveCommand.SetConfig(ADRcp adr, RBasicParaStruct dt);

5.2.4 bool PassiveCommand.GetAddress(ADRcp adr);

5.2.5 bool PassiveCommand.SetAddress(ADRcp adr, int address);

5.2.6 bool PassiveCommand.Secret(ADRcp adr);

5.2.7 bool PassiveCommand.Reset(ADRcp adr);

5.2.8 bool PassiveCommand.InitSyris(ADRcp adr, byte[]iData);

5.2.9 bool PassiveCommand.Remote(ADRcp adr, byte position, byte state);

5.2.10 bool PassiveCommand.GetTime(ADRcp adr);

5.2.11 bool PassiveCommand.SetTime(ADRcp adr, byte[] iData);

5.3 EPCC1-G2 Function:

5.3.1 bool PassiveCommand.Identify6C (ADRcp adr);

5.3.2 bool PassiveCommand.Identify6CMult(ADRcp adr);

5.3.3 bool PassiveCommand.Read6C(ADRcp adr, int iMem, int iStartWord, int iLengthWord);

5.3.4 Bool PassiveCommand.Write6C(ADRcp adr, int iMem, int iStartWord, int iLengthWord, byte[] iData);

5.4 ISO18000-6B Function:

5.4.1 bool PassiveCommand.Identify6B (ADRcp adr);

5.4.2 bool PassiveCommand.Read6B(ADRcp adr , int iStart, int iLength)

5.4.3 Bool PassiveCommand.Write6B(ADRcp adr, int iStart, int iLength, byte[] iData);

5.5 Senior Function:

5.5.1 bool PassiveCommand.GetTcpip(ADRcp adr);

5.5.2 bool PassiveCommand.SetTcpip(ADRcp adr, byte[] iData);

6. Function Explanation

6.1 General Function:

6.1.1 Connect();

```
/// <summary>
/// <para>Create a communication connection</para>
/// </summary>
/// <param name="hostOrCom">
/// <para>IP Address or Comport</para>
/// <para>COM TYPE:"COM1"</para>
/// <para>NET TYPE:"192.168.2.115"</para>
/// <para>USB TYPE:"AD"</para>
/// </param>
/// <param name="baudOrPort">
/// <para>IP Port or Baudrate</para>
/// <para>COM TYPE:"9600"</para>
/// <para>NET TYPE:"49152"</para>
/// <para>USB TYPE:"0"</para>
/// </param>
/// <param name="type">
/// <para>Communication Type</para>
/// <para>0 - COM,1 - NET,2 - USB</para>
/// </param>
/// <returns></returns>
```

6.1.2 DisConnect();

```
/// <summary>
/// <para>DisConnect communication connection</para>
/// </summary>
```

6.2 Base Function:

6.2.1 GetInformation();

```
/// <summary>
/// Get base information -  获取基本信息
/// </summary>
/// <param name="adr">Communication Data Managed Base Class</param>
/// <returns></returns>
```

6.2.2 GetConfig();

```
/// <summary>
/// Get Basic Parameters -  获取基本参数
/// </summary>
/// <param name="adr">Communication Data Managed Base Class</param>
/// <returns></returns>
```

6.2.3 SetConfig();

```
/// <summary>
/// Set Basic Parameters -  设置基本参数
/// </summary>
/// <param name="adr">Communication Data Managed Base Class</param>
/// <param name="dt">base parameters struct</param>
/// <returns></returns>
```

6.2.4 GetAddress();

```
/// <summary>
/// Get Device Address
/// </summary>
/// <param name="adr">Communication Data Managed Base Class</param>
```

6.2.5 SetAddress();

```
/// <summary>
/// Set Device Address
/// </summary>
/// <param name="adr">Communication Data Managed Base Class</param>
/// <param name="address">New Address</param>
```

6.2.6 Secret();

```
/// <summary>
/// Secret EPC tag   -  加密卡号
/// </summary>
/// <param name="adr">Communication Data Managed Base Class</param>
```

6.2.7 Reset();

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

```
/// Reset Reader -  重启读卡器
/// </summary>
 /// <param name="adr">Communication Data Managed Base Class</param>
```

1.1.1 InitSyris();

```
 /// <summary>
 /// Init Syris SN And ID(nonstandard) -  初始化读卡器序列号和ID(非标准命令)
 /// </summary>
 /// <param name="adr">Communication Data Managed Base Class</param>
/// <param name="iData">
/// <para>Syris SN Info Array</para>
/// <para>Syris  序列号和ID字节数组</para>
/// </param>
```

1.1.2 Remote();

```
/// <summary>
/// Set IO Out[nonstandard] -  设置IO口输出(非标准命令)
/// </summary>
/// <param name="adr">Communication Data Managed Base Class</param>
/// <param name="position">输出位置</param>
 /// <param name="state">输出状态</param>
```

1.1.3 GetTime();

```
//// <summary>
/// Get Datetime Parameters(nonstandard) -  获取读卡器实时时间(非标准命令)
/// </summary>
 /// <param name="adr">Communication Data Managed Base Class</param>
```

1.1.4 SetTime();

```
/// <summary>
/// Set Datetime Parameters(nonstandard) -  设置读卡器实时时间(非标准命令)
/// </summary>
/// <param name="adr">Communication Data Managed Base Class</param>
 /// <param name="iData"></param>
```

1.2 EPCC1-G2 Function:

1.2.1 Identify6C();

```
/// <summary>
 /// Identify card from ISO18000-6C(EPC) tag -  识别单张ISO18000-6C(EPC)标签卡号
 /// </summary>
 /// <param name="adr">Communication Data Managed Base Class</param>
```

1.2.2 Identify6CMult()

```
/// <summary>
/// Identify cards from mult ISO18000-6C(EPC) tag -  识别多张ISO18000-6C(EPC)标签卡号
/// </summary>
/// <param name="adr">Communication Data Managed Base Class</param>
 /// <returns></returns>
```

1.2.3 Read6C();

```
/// <summary>
/// Read data from ISO18000-6C(EPC) tag -  读取ISO18000-6C(EPC)标签数据
/// </summary>
/// <param name="adr">Communication Data Managed Base Class</param>
/// <param name="iMem">Memory Bank, 0x00-RFU,0x01-EPC,0x02-TID,0x03-User  块地址</param>
/// <param name="iStartWord">Start Address(word)  块起始地址(word)</param>
/// <param name="iLengthWord">Data Length in Words(word)  数据长度(word)
/// <para>单次操作不超过16byte(8Words)</para>
/// </param>
 /// <returns></returns>
```

1.2.4 Write6C();

```
/// <summary>
/// Write Data to ISO18000-6C(EPC) tag -  写入ISO18000-6C(EPC)标签数据
/// </summary>
/// <param name="adr">Communication Data Managed Base Class</param>
/// <param name="iMem">Memory Bank, 0x00-RFU,0x01-EPC,0x02-TID,0x03-User  块地址</param>
/// <param name="iStartWord">Start Address(word)  块起始地址(word)</param>
/// <param name="iLengthWord">Data Length in Words(word)  数据长度(word)
/// <para>单次操作不超过16byte(8Words)</para>
/// </param>
/// <param name="iData">data(byte array)(16进制数组)</param>
 /// <returns></returns>
```

1.3 ISO18000-6B Function:

1.3.1 Identify6B();

```
/// <summary>
  /// Identify card from ISO18000-6B tag -  识别ISO18000-6B标签卡号
  /// </summary>
  /// <param name="adr">Communication Data Managed Base Class</param>
  /// <returns></returns>
```

1.3.2 Read6B();

```
/// <summary>
/// Read data from ISO18000-6B tag -  读取ISO18000-6B标签数据
/// </summary>
/// <param name="adr">Communication Data Managed Base Class</param>
/// <param name="iStart">Start Address(byte)  块起始地址(byte)</param>
/// <param name="iLength">Data length(byte)  数据长度(byte)</param>
/// <returns></returns>
```

1.3.3 Write6B();

```
/// <summary>
/// Write Data to ISO18000-6B tag -  写入ISO18000-6B标签数据
/// </summary>
/// <param name="adr">Communication Data Managed Base Class</param>
/// <param name="iStart">Start Address(byte)  块起始地址(byte)</param>
/// <param name="iLength">Data length(byte)  数据长度(byte)</param>
/// <param name="iData">data(byte array)(16进制数组)</param>
/// <returns></returns>
```

1.4 Senior Function:

1.4.1 GetTcpip();

```
/// <summary>
/// Get Tcpip Parameters(nonstandard) -  获取TCPIP参数(非标准命令)
/// </summary>
  /// <param name="adr">Communication Data Managed Base Class</param>
```

1.4.2 SetTcpip();

```
/// <summary>
/// Set Tcpip Parameters(nonstandard) -  设置TCPIP参数(非标准命令)
/// </summary>
/// <param name="adr">Communication Data Managed Base Class</param>
  /// <param name="iData"></param>
```

2. Struct Information

2.1 ProtocolStruct

```
[StructLayout(LayoutKind.Sequential, Size = 261, Pack = 1)]
public struct ProtocolStruct
{
  /// <summary>
  /// START OF INFORMATION
  /// </summary>
  public byte Preamble;//0
  /// <summary>
  /// Equip address（1～65534）,(65535 public address,0 reserve address)
  /// </summary>
  [MarshalAs(UnmanagedType.ByValArray, SizeConst = 2)]
  public byte[] Address;//1-2
  /// <summary>
  /// Control identification code (data type description)
  /// </summary>
  public byte Code;//3
  /// <summary>
  /// control identification code (action type description)
  /// </summary>
  public byte Type;//4
  /// <summary>
  /// INFO Data Length
  /// </summary>
  public byte Length;//5
  /// <summary>
  /// INFO Data Packet,（include Checksum byte)
  /// </summary>
  [MarshalAs(UnmanagedType.ByValArray, SizeConst = 255)]
  public byte[] Payload;//7
}
```

2.2 PBasicParameters

```
public class PBasicParameters
{
    #region Attribute
    /// <summary>
    /// 功率大小,可调节读卡器距离(0~30)
    /// </summary>
    public byte PowerSize { set; get; }
    /// <summary>
    /// 跳频使能 1-定频，2,-跳频
    /// </summary>
```

```
public byte HoppingEnable { set; get; }
/// <summary>
///  定频值  default: 84(902MHz)  值域:[0~200](860MHz ~ 960MHz)
/// </summary>
public byte FixedFreq { set; get; }
/// <summary>
///  跳频值1 default: 84(902MHz)  值域:[0~200]
/// </summary>
public byte Hopping1 { set; get; }
/// <summary>
///  跳频值2 default: 93(906.5MHz)  值域:[0~200]
/// </summary>
public byte Hopping2 { set; get; }
/// <summary>
///  跳频值3 default: 102(911MHz)  值域:[0~200]
/// </summary>
public byte Hopping3 { set; get; }
/// <summary>
///  跳频值4 default: 110(915MHz)  值域:[0~200]
/// </summary>
public byte Hopping4 { set; get; }
/// <summary>
///  跳频值5 default: 119(919.5MHz)  值域:[0~200]
/// </summary>
public byte Hopping5 { set; get; }
/// <summary>
///  跳频值6 default: 130(925MHz)  值域:[0~200]
/// </summary>
public byte Hopping6 { set; get; }
/// <summary>
///  工作模式,1-应答,2-主动,3-被动
/// </summary>
public byte WorkMode { set; get; }
/// <summary>
///  读卡周期ms, default : 10  值域:[5~255]
/// </summary>
public byte ReadInterval { set; get; }
/// <summary>
///  触发模式,0-关闭,2-低电平触发
/// </summary>
public byte Trigger { set; get; }
/// <summary>
///  输出类型,1-232,2-485,3-TCPIP,4-CANBUS,5-SYRIS,6-WG26,7-WG34
/// </summary>
public byte OutputMode { set; get; }
/// <summary>
///  数据偏移(韦根参数) 0~20
```

```
/// </summary>
public byte ByteOffset { set; get; }
/// <summary>
/// 输出周期(韦根参数) default: 30 值域:[0~255]
/// </summary>
public byte OutInterval { set; get; }
/// <summary>
/// 脉冲宽度(韦根参数) default: 10 值域:[0~255]
/// </summary>
public byte PulseWidth { set; get; }
/// <summary>
/// 脉冲周期(韦根参数) default: 15 值域:[0~255]
/// </summary>
public byte PulsePeriod { set; get; }
/// <summary>
/// 天线
/// </summary>
public byte Antenna { set; get; }
/// <summary>
/// 读卡类别,1-[6B单卡],16-[6C单卡],17-[6B+6C],32-[6C多卡],64-[6C+其他区域]
/// </summary>
public byte CardType { set; get; }
/// <summary>
/// 相同ID输出间隔 default: 1 值域:[0~255]
/// </summary>
public byte SameIDinterval { set; get; }
/// <summary>
/// 嗡鸣器使能 default: 0-禁止 1-使能
/// </summary>
public byte Buzzer { set; get; }
/// <summary>
/// 其他区域选择 default: 1 值域:[1~2] 1-TID,2-USER
/// </summary>
public byte Area { set; get; }
/// <summary>
/// 其他区域起始位置 default: 0 值域:[0~31]
/// </summary>
public byte StartWord { set; get; }
/// <summary>
/// 其他区域数据长度 default: 2 值域:[1~12]
/// </summary>
public byte Length { set; get; }
/// <summary>
/// 加密使能 0-禁止 1-使能
/// </summary>
public byte Encrypt { set; get; }
/// <summary>
```

```
        /// 加密密码 default: 0 值域:[0~9999]
        /// </summary>
        public int Password { set; get; }
        /// <summary>
        /// 多卡最大值 default: 32 值域:[10~64]
        /// </summary>
        public byte MaxTag { set; get; }
}
```