RFID 读写设备 PC
开发指南 C#

编写人：刘枭

V0.3.0.0

目录

1. 前言.....4
1.1. 概述.....4
1.2. 适用设备.....4
1.3. 版权说明.....4
1.4. 读写基本流程.....5
2. 快速上手.....5
3. 连接说明.....8
3.1. RS232 连接.....8
3.2. RS485 连接.....8
3.3. TCP 客户端连接.....9
3.4. USB-HID 连接.....9
3.5. TCP 服务器监听.....9
3.6. 关闭连接.....10
4. 事件说明.....10
4.1. ISO18000-6C 标签上报事件.....10
上报对象.....10
4.2. ISO18000-6C 标签上报结束事件.....11
4.3. ISO18000-6B 标签上报事件.....12
上报对象.....12
4.4. ISO18000-6B 标签上报结束事件.....13
4.5. 国标标签上报事件.....13
上报对象.....13
4.6. 国标标签上报结束事件.....14
4.7. GPI 触发开始事件.....15
上报对象.....15
4.8. GPI 触发结束事件.....15
上报对象.....15
4.9. TCP 连接断开事件.....16
4.10. TCP 连接事件.....16
上报对象.....17
5. 消息配置与查询说明.....17
5.1. 发送同步消息.....17
代码示例 1.....18
代码示例 2.....18
代码示例 3.....18
6. 消息说明.....19
6.1. 读写器配置与管理.....19
6.1.1. 重启读写器.....19
6.1.2. 配置与查询串口参数.....20
6.1.3. 配置 GPO 状态参数.....20
6.1.4. 查询 GPI 状态参数.....21
6.1.5. 配置与查询 GPI 触发参数.....21

6.1.6. 查询基带软件版本.....22
6.1.7. 查询读写器信息.....22
6.1.8. 查询读写器 RFID 能力.....22
6.1.9. 获取标签缓存数据.....23
6.1.10. 清空标签缓存数据.....24
6.2. RFID 配置与操作.....24
6.2.1. 停止指令.....24
6.2.2. 配置与查询读写器功率.....24
6.2.3. 配置与查询读写器工作频段.....25
6.2.4. 配置与查询 EPC 基带参数.....25
6.2.5. 配置与查询标签上传参数.....26
6.2.6. 读 EPC 标签.....26
6.2.7. 写 EPC 标签.....27
6.2.8. 锁 EPC 标签.....27
6.2.9. 灭活 EPC 标签.....28
6.2.10. 读 6B 标签.....28
6.2.11. 写 6B 标签.....28
6.2.12. 6B 标签锁定.....29
6.2.13. 6B 标签锁定查询.....29
6.2.14. 读 GB 标签.....30
6.2.15. 写 GB 标签.....30
6.2.16. 锁 GB 标签.....31
6.2.17. 灭活 GB 标签.....31
7. 参数说明.....32
7.1.1. 6C/GB 标签选择参数.....32
7.1.2. 6C 标签读取 TID 参数.....32
7.1.3. 6C 标签读取用户数据区参数.....33
7.1.4. 6B 标签读用户数据区参数.....33
7.1.5. GB 标签读用户数据区参数.....33
8. 附录 1.....34
读写器所支持的频段列表.....34
9. 附录 2.....34

1. 前言

1.1. 概述

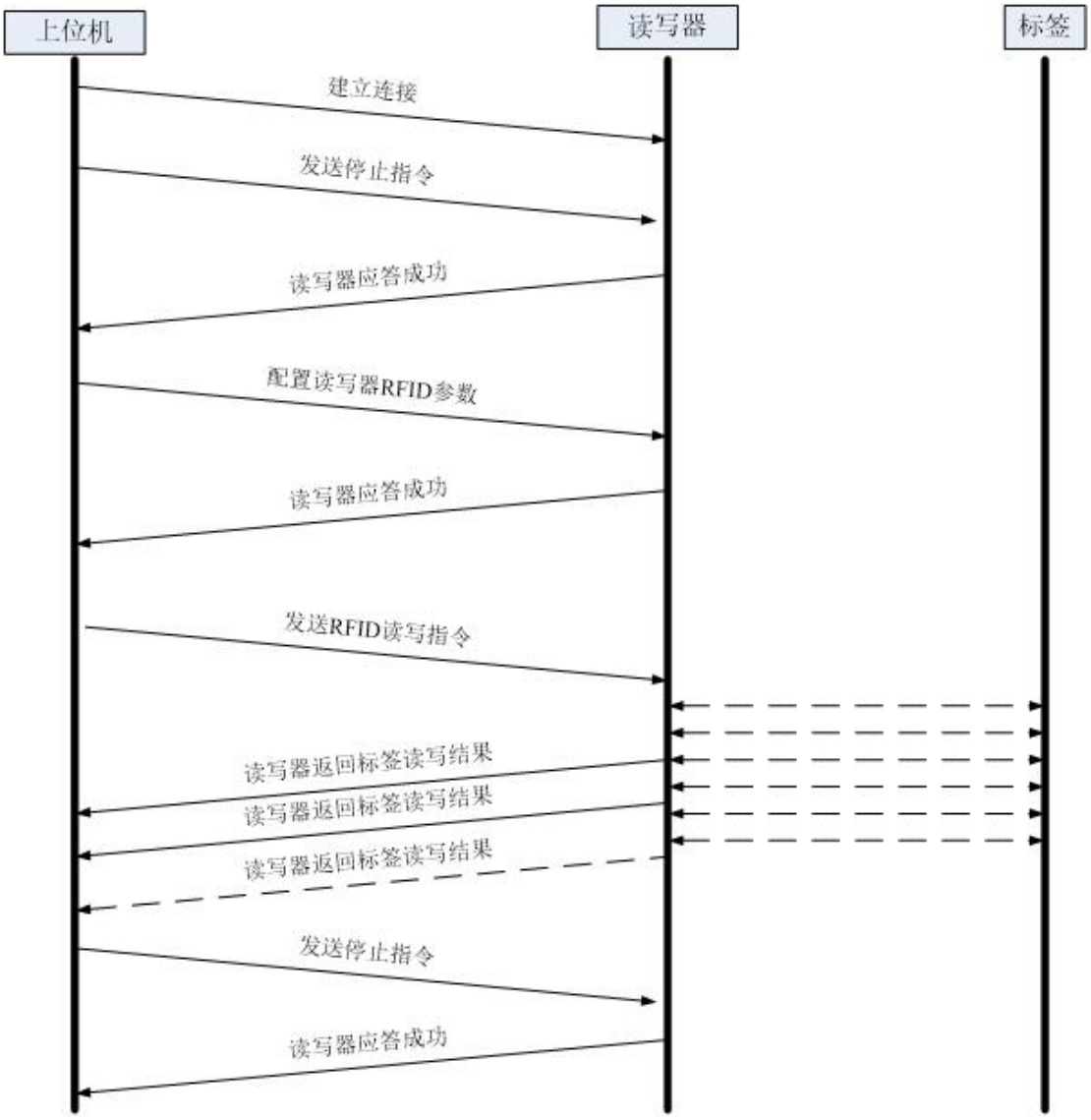
为了方便进行二次开发，我们提供了可以在.net 平台进行运行的函数库。该库采用 C#语言编写并封装成标准的 DLL 库“GReaderApi.dll”，支持.net framework 2.0 及以上版本。

1.2. 适用设备

功能模块	适用设备类型
读写器配置与管理	R8008、R8004
RFID 配置与操作	UHF 设备全型号

1.3. 版权说明

1.4.读写基本流程



2.快速上手

```
using GDotnet.Reader.Api.DAL;
using GDotnet.Reader.Api.Protocol.Gx;
using System;
using System.Collections.Generic;
using System.Text;

// ==========================================================
//   Copyright (C) 2019 SZGxwl Inc. All rights reserved.
//
```

```
//   Create by xiao.liu at 2019/1/11 11:22:51.
//
//   xiao.liu [mailto:fanqie0127@gmail.com]
// ============================================================
namespace GDotnet.Reader.Api
{
    static class Example
    {
        static void Main()
        {
            GClient clientConn = new GClient();
            eConnectionAttemptEventStatusType status;
            // clientConn.OpenTcp("192.168.1.168:6180", 3000, out status)
            if (clientConn.OpenSerial("COM16:115200", 3000, out status))
            {
                // 订阅标签上报事件
                clientConn.OnEncapedTagEpcLog += new delegateEncapedTagEpcLog(OnEncapedTagEpcLog);
                clientConn.OnEncapedTagEpcOver += new
delegateEncapedTagEpcOver(OnEncapedTagEpcOver);

                // 停止指令，空闲态
                MsgBaseStop msgBaseStop = new MsgBaseStop();
                clientConn.SendSynMsg(msgBaseStop);
                if (0 == msgBaseStop.RtCode)
                {
                    Console.WriteLine("Stop successful.");
                }
                else { Console.WriteLine("Stop error."); }

                // 功率配置，将 4 个天线功率都设置为 30dBm.
                MsgBaseSetPower msgBaseSetPower = new MsgBaseSetPower();
                msgBaseSetPower.DicPower = new Dictionary<byte, byte>()
                {
                    {1, 30},
                    {2, 30},
                    {3, 30},
                    {4, 30}
                };
                clientConn.SendSynMsg(msgBaseSetPower);
                if (0 == msgBaseSetPower.RtCode)
                {
                    Console.WriteLine("Power configuration successful.");
                }
                else { Console.WriteLine("Power configuration error."); }
```

```
            Console.WriteLine("Enter any character to start reading the tag.");
            Console.ReadKey();

            // 4 个天线读卡，读取 EPC 数据区以及 TID 数据区
            MsgBaseInventoryEpc msgBaseInventoryEpc = new MsgBaseInventoryEpc();
            msgBaseInventoryEpc.AntennaEnable = (ushort)(eAntennaNo._1 | eAntennaNo._2 |
eAntennaNo._3 | eAntennaNo._4);
            msgBaseInventoryEpc.InventoryMode = (byte)eInventoryMode.Inventory;
            msgBaseInventoryEpc.ReadTid = new ParamEpcReadTid();                // tid 参数
            msgBaseInventoryEpc.ReadTid.Mode = (byte)eParamTidMode.Auto;
            msgBaseInventoryEpc.ReadTid.Len = 6;
            clientConn.SendSynMsg(msgBaseInventoryEpc);
            if (0 == msgBaseInventoryEpc.RtCode)
            {
                Console.WriteLine("Inventory epc successful.");
            }
            else { Console.WriteLine("Inventory epc error."); }
            Console.ReadKey();

            // 停止读卡，空闲态
            clientConn.SendSynMsg(msgBaseStop);
            if (0 == msgBaseStop.RtCode)
            {
                Console.WriteLine("Stop successful.");
            }
            else { Console.WriteLine("Stop error."); }
        }
        else
        {
            Console.WriteLine("Connect failure.");
        }
        Console.ReadKey();
    }

    #region API 事件

    public static void OnEncapedTagEpcLog(EncapedLogBaseEpcInfo msg)
    {
        // 回调内部如有阻塞，会影响 API 正常使用
        // 标签回调数量较多，请将标签数据先缓存起来再作业务处理
        if (null != msg)
        {
            Console.WriteLine(msg.logBaseEpcInfo.ToString());
        }
```

```
        }

        public static void OnEncapedTagEpcOver(EncapedLogBaseEpcOver msg)
        {
            if (null != msg)
            {
                Console.WriteLine("Epc log over.");
            }
        }

        #endregion
    }
}
```

3. 连接说明

3.1. RS232 连接

命名空间	GDotnet.Reader.Api.DAL
类	GClient
方法	public bool OpenSerial(String readerName, int timeout, out eConnectionAttemptEventStatusType status)
说明	readerName: 连接字符串，如"COM1:115200"("串口号:波特率") timeout: 连接确认超时时间（毫秒），如 "1000" status: 连接状态枚举

3.2. RS485 连接

命名空间	GDotnet.Reader.Api.DAL
类	GClient
方法	public bool OpenSerial485(String readerName, int timeout, out eConnectionAttemptEventStatusType status)

说明	readerName: 连接字符串，如”COM1:115200:1”(“串口号:波特率:485 地址”) timeout: 连接确认超时时间（毫秒），如 “1000” status: 连接状态枚举 半双工模式下，偶尔的通讯失败属于正常现象，失败后请重试。

3.3. TCP 客户端连接

命名空间	GDotnet.Reader.Api.DAL
类	GClient
方法	public bool OpenTcp(String readerName, int timeout, out eConnectionAttemptEventStatusType status)
说明	readerName: 连接字符串(如“192.168.1.168:6180”)。读写器默认 IP ”192.168.1.168”，默认端口 “6180” timeout: 连接确认超时时间（毫秒），如 “1000” status: 连接状态枚举

3.4. USB-HID 连接

命名空间	GDotnet.Reader.Api.DAL
类	GClient
方法	public bool OpenUsbHid(String readerName,IntPtr handle, int timeout, out eConnectionAttemptEventStatusType status)
说明	readerName: 连接字符串 (通过 GetUsbHidList()方法获取可用 USB 列表)。 handle: 句柄，可以为 IntPtr.Zero。 timeout: 连接确认超时时间（毫秒），如 “1000” status: 连接状态枚举

3.5. TCP 服务器监听

命名空间	GDotnet.Reader.Api.DAL
类	GServer

方法	public bool Open(int port)
说明	port: 上位机监听的本地接口。 使用该方法监听，需要将 UHF 读写设备配置成“客户端模式”。 “客户端模式”配置方法详见《RFID 演示软件操作手册》。

3.6. 关闭连接

命名空间	GDotnet.Reader.Api.DAL
类	GClient
方法	public void Close()
说明	关闭并释放链接资源。 注：对于已经失效的链接对象，需要主动调用此方法释放资源。

4. 事件说明

4.1. ISO18000-6C 标签上报事件

命名空间	GDotnet.Reader.Api.DAL
类	GClient
事件	public delegateTagEpcLog OnTagEpcLog;
说明	public delegate void delegateTagEpcLog(LogBaseEpcInfo msg); 6C 标签主动上报事件：当读写器为读卡状态时，标签信息会通过此事件上报。 示例详见”**快速上手**”。 LogBaseEpcInfo：详见”**上报对象**”

上报对象

命名空间	GDotnet.Reader.Api.Protocol.Gx
对象	LogBaseEpcInfo
属性	Epc：16 进制 EPC 字符串 BEpc：EPC 字节数组 Pc：PC 值 AntId：天线编号 Rssi：信号强度 Result：标签读取结果，0 为读取成功，非 0 为失败 1，标签无响应 2，CRC 错误 3，数据区被锁定 4，数据区溢出 5，访问密码错误 6，其他标签错误 7，其他读写器错误 Tid：16 进制 TID 字符串 BTid：TID 字节数组 Userdata：16 进制 Userdata 字符串 BUser：Userdata 字节数组 Reserved：16 进制保留区字符串 BRes：保留区字节数组
说明	6C 标签主动上报参数。

4.2. ISO18000-6C 标签上报结束事件

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	GClient
属性	`public delegateTagEpcOver OnTagEpcOver;`
说明	`public delegate void delegateTagEpcOver(LogBaseEpcOver msg);` 6C 标签主动上报结束参数，以确保异步消息得到同步状态。

4.3.ISO18000-6B 标签上报事件

命名空间	GDotnet.Reader.Api.DAL
类	GClient
事件	public delegateTag6bLog OnTag6bLog;
说明	public delegate void delegateTag6bLog(LogBase6bInfo msg); 6B 标签主动上报事件：当读写器为读卡状态时，标签信息会通过此事件上报。 示例详见"**快速上手**"。 LogBase6bInfo：详见"**上报对象**"

上报对象

命名空间	GDotnet.Reader.Api.Protocol.Gx
对象	LogBase6bInfo
属性	AntId：天线编号 Rssi：信号强度 Result：标签读取结果，0 为读取成功，非 0 为失败 1，标签无响应 2，CRC 错误 3，其他读写器错误 Tid：16 进制 TID 字符串 BTid：TID 字节数组 Userdata：16 进制 Userdata 字符串 BUser：Userdata 字节数组
说明	6C 标签主动上报参数。

4.4. ISO18000-6B 标签上报结束事件

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	GClient
属性	`public delegateTag6bOver OnTag6bOver;`
说明	`public delegate void delegateTag6bOver(LogBase6bOver msg);` 6B 标签主动上报结束参数，以确保异步消息得到同步状态。

4.5. 国标标签上报事件

命名空间	GDotnet.Reader.Api.DAL
类	GClient
事件	`public delegateTagGbLog OnTagGbLog;`
说明	`public delegate void delegateTagGbLog(LogBaseGbInfo msg);` GB 标签主动上报事件：当读写器为读卡状态时，标签信息会通过此事件上报。 示例详见"**快速上手**"。 LogBaseGbInfo：详见"**上报对象**"

上报对象

命名空间	GDotnet.Reader.Api.Protocol.Gx
对象	LogBaseGxInfo

属性	Epc：16 进制 EPC 字符串(标签编码区) BEpc：EPC 字节数组(标签编码区) Pc：PC 值 AntId：天线编号 Rssi：信号强度 Result：标签读取结果，0 为读取成功，非 0 为失败 1，标签无响应 2，CRC 错误 3，数据区被锁定 4，数据区溢出 5，读取密码错误 6，权限不足 7，鉴别失败 8，其他标签错误 9，其他读写器错误 Tid：16 进制 TID 字符串(标签信息区) BTid：TID 字节数组(标签信息区) Userdata：16 进制 Userdata 字符串 BUser：Userdata 字节数组
说明	6C 标签主动上报参数。

4.6. 国标标签上报结束事件

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	GClient
属性	`public delegateTagGbOver OnTagGbOver;`
说明	`public delegate void delegateTagGbOver(LogBaseGbOver msg);` GB 标签主动上报结束参数，以确保异步消息得到同步状态。

4.7. GPI 触发开始事件

命名空间	GDotnet.Reader.Api.DAL
类	GClient
事件	public delegateGpiStart OnGpiStart;
说明	public delegate void delegateGpiStart(LogBaseGpiStart msg); GPI 触发开始主动上报事件：当 GPI 达到配置的开始条件时，此事件会上报。 LogBaseGpiStart：详见"**上报对象**"

上报对象

命名空间	GDotnet.Reader.Api.Protocol.Gx
对象	LogBaseGpiStart
属性	GpiPort：GPI 端口号（索引从 0 开始） Level：电平状态，0 低电平 1 高电平 TriggerTime：触发时间
说明	GPI 触发开始主动上报参数。

4.8. GPI 触发结束事件

命名空间	GDotnet.Reader.Api.DAL
类	GClient
事件	public delegateGpiOver OnGpiOver;
说明	public delegate void delegateGpiOver(LogBaseGpiOver msg); GPI 触发开始主动上报事件：当 GPI 达到配置的结束条件时，此事件会上报。 LogBaseGpiOver：详见"**上报对象**"

上报对象

命名空间	GDotnet.Reader.Api.Protocol.Gx
对象	LogBaseGpiOver
属性	GpiPort：GPI 端口号（索引从 0 开始） Level：电平状态，0 低电平 1 高电平 TriggerTime：触发时间
说明	GPI 触发结束主动上报参数。

4.9. TCP 连接断开事件

命名空间	GDotnet.Reader.Api.DAL
类	GClient
事件	public delegateTcpDisconnected OnTcpDisconnected;
说明	public delegate void delegateTcpDisconnected(String readerName); 说明： ➢ 连接处于 TCP，当远程连接主动断开或者物理层异常时，此事件上报。 ➢ 此事件上报后，需由上位机（调用者）释放连接对象，否则事件会循环上报，直至连接对象被释放(Close)。 ➢ 为满足不同需求，是否重连该远程设备，由上位机（调用者）自主控制。 readerName：连接对象名称。

4.10. TCP 连接事件

命名空间	GDotnet.Reader.Api.DAL
类	GServer

事件	public delegateGClientConnected OnClientConnected;
说明	public delegate void delegateGClientConnected(GClient client); TCP 处于监听状态，远程读写设备主动连接上位机时，会触发此事件上报。 GClient：详见"上报对象"。

上报对象

命名空间	GDotnet.Reader.Api.DAL
对象	GClient
属性	无。
说明	说明：此连接对象与其他主动连接的对象相同，用法完全一致。

5. 消息配置与查询说明

5.1. 发送同步消息

命名空间	GDotnet.Reader.Api.DAL
类	GClient
方法	public void SendSynMsg(Message msg)
方法 1	public void SendSynMsg(Message msg, int timeout)
方法 2	public void SendSynMsgRetry(Message msg, int timeout, int retry)
返回值	msg.RtCode：消息返回码 0 为操作成功，非 0 操作失败。 msg.RtMsg：操作失败原因

说明	发送同步消息,详见代码示例。 提示：“读写器配置与管理”、“RFID 配置与操作”等消息，均通过此方法发送。

代码示例 1

```
// 停止指令，空闲态
MsgBaseStop msgBaseStop = new MsgBaseStop();
clientConn.SendSynMsg(msgBaseStop);
if (0 == msgBaseStop.RtCode)
{
    Console.WriteLine("Stop successful.");
}
else { Console.WriteLine("Stop error."); }
```

代码示例 2

```
// 功率配置，将 4 个天线功率都设置为 30dBm.
MsgBaseSetPower msgBaseSetPower = new MsgBaseSetPower();
msgBaseSetPower.DicPower = new Dictionary<byte, byte>()
{
    {1, 30},
    {2, 30},
    {3, 30},
    {4, 30}
};
clientConn.SendSynMsg(msgBaseSetPower);
if (0 == msgBaseSetPower.RtCode)
{
    Console.WriteLine("Power configuration successful.");
}
else { Console.WriteLine("Power configuration error."); }
```

代码示例 3

```
if (null != this.clientConn)
{
    // 查询天线功率
    MsgBaseGetPower msg = new MsgBaseGetPower();
    this.clientConn.SendSynMsg(msg);
    if (0 == msg.RtCode && null != msg.DicPower)
    {
        foreach (var item in msg.DicPower)
        {
```

```
                    switch (item.Key)
                    {
                        case 1:
                            {
                                cmbAnt1.SelectedIndex = item.Value;
                            }
                            break;
                        case 2:
                            {
                                cmbAnt2.SelectedIndex = item.Value;
                            }
                            break;
                        case 3:
                            {
                                cmbAnt3.SelectedIndex = item.Value;
                            }
                            break;
                        case 4:
                            {
                                cmbAnt4.SelectedIndex = item.Value;
                            }
                            break;
                        default:
                            break;
                    }
                }
            }
```

6. 消息说明

6.1. 读写器配置与管理

6.1.1. 重启读写器

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	MsgAppReset

属性	无
说明	设备重启消息，一般在修改需要重启生效的配置后执行。

6.1.2. 配置与查询串口参数

命名空间	GDotnet.Reader.Api.Protocol.Gx
配置类	MsgAppSetSerialParam
查询类	MsgAppGetSerialParam
属性	BaudrateIndex: 波特率索引（0，9600 bps； 1，19200 bps； 2，115200 bps； 3，230400 bps; 4， 460800bps）
说明	(持久化配置，断电保存)配置设备串口参数。 注：此配置需在设备空闲时修改（即循环读卡状态无法更改配置）。

6.1.3. 配置 GPO 状态参数

命名空间	GDotnet.Reader.Api.Protocol.Gx
配置类	MsgAppSetGpo
属性	Gpo1: 0 （低，继电器断开） 1 （高，继电器闭合） Gpo2: 0 （低，继电器断开） 1 （高，继电器闭合） Gpo3: 0 （低，继电器断开） 1 （高，继电器闭合） Gpo4: 0 （低，继电器断开） 1 （高，继电器闭合） ……
说明	(持久化配置，断电保存)配置设备 GPO 参数。 注：对于不需要控制状态的 GPO，无须赋值。

6.1.4. 查询 GPI 状态参数

命名空间	GDotnet.Reader.Api.Protocol.Gx
查询类	MsgAppGetGpiState
属性	DicGpiState: 对应 GPI 的电平状态(Dictionary<byte, byte> , key：GPI 索引号，value：电平状态（0 低，1 高）)
说明	查询设备 GPI 状态。 注：索引号从 1 开始。

6.1.5. 配置与查询 GPI 触发参数

命名空间	GDotnet.Reader.Api.Protocol.Gx
配置类	MsgAppSetGpiTrigger
查询类	MsgAppGetGpiTrigger
属性	GpiPort: GPI 端口号，索引从 0 开始 TriggerStart: 触发开始（0 触发关闭，1 低电平触发，2 高电平触发，3 上升沿触发，4 下降沿触发，5 任意边沿触发） TriggerCommand:触发绑定命令（Hex，可为空） BtriggerCommand:触发绑定命令（Byte[]，可以空） TriggerOver:触发停止（0 不停止，1 低电平触发，2 高电平触发，3 上升沿触发，4 下降沿触发，5 任意边沿触发，6 延时停止） OverDelayTime:延时停止时间（仅当停止条件为“延时停止”生效） LevelUploadSwitch:触发不停止时 IO 电平变化上传开关（0 不上传，1 上传）
说明	(持久化配置，断电保存)配置设备 GPI 触发参数。 注：此配置需在设备空闲时修改（即循环读卡状态无法更改配置）。

6.1.6. 查询基带软件版本

命名空间	GDotnet.Reader.Api.Protocol.Gx
查询类	MsgAppGetBaseVersion
属性	Version: 基带软件版本号（如：“0.1.0.0”）
说明	无。

6.1.7. 查询读写器信息

命名空间	GDotnet.Reader.Api.Protocol.Gx
查询类	MsgAppGetReaderInfo
属性	Imei: 读写器流水号 PowerOnTime：上电时间 BaseBuildDate：基带编译时间 AppVersion：应用软件版本（如：“0.1.0.0”） AppBuildDate：应用编译时间 SystemVersion：操作系统版本
说明	无。

6.1.8. 查询读写器 RFID 能力

命名空间	GDotnet.Reader.Api.Protocol.Gx
查询类	MsgBaseGetCapabilities

属性	MaxPower：最大支持功率 MinPower：最小支持功率 AntennaCount：天线数量 FrequencyArray：支持的频段列表， 0， 国标 920~925MHz 1， 国标 840~845MHz 2， 国标 840~845MHz 和 920~925MHz 3， FCC，902~928MHz 4，ETSI，866~868MHz ProtocolArray：支持的协议列表， 0， ISO18000-6C/EPC C1G2 1， ISO18000-6B 2， 国标 GB/T 29768-2013 3， 国军标 GJB 7383.1-2011
说明	无。

6.1.9. 获取标签缓存数据

命名空间	GDotnet.Reader.Api.Protocol.Gx
查询类	MsgAppGetResume
属性	RtCode 为 0 时有缓存，为 1 时无缓存。
说明	➢ 获取缓存标签数据，需要在”读写器管理软件”->”设备配置”，开启”TCP 断点续传”功能。 ➢ 离线缓存的数据，需要通过该消息主动获取，标签数据通过标签上报事件上报，同时附带标签实际读取到的时间。

6.1.10. 清空标签缓存数据

命名空间	GDotnet.Reader.Api.Protocol.Gx
查询类	MsgAppClearCache
属性	无。
说明	清空离线缓存的标签数据。

6.2. RFID 配置与操作

6.2.1. 停止指令

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	MsgBaseStop
属性	无
说明	停止读写器所有的 RFID 操作，并使读写器进入到空闲状态。 提示：当读写器处于读卡状态时，所有配置消息将无法发送，必须发送停止指令。

6.2.2. 配置与查询读写器功率

命名空间	GDotnet.Reader.Api.Protocol.Gx
配置类	MsgBaseSetPower
查询类	MsgBaseGetPower
属性	DicPower：读写器对应天线功率(Dictionary<byte, byte> , key：天线索引号，value：天线功率值)

说明	(持久化配置，断电保存)配置各个天线端口读写器功率。

6.2.3. 配置与查询读写器工作频段

命名空间	GDotnet.Reader.Api.Protocol.Gx
配置类	MsgBaseSetFreqRange
查询类	MsgBaseGetFreqRange
属性	FreqRangeIndex: 频段索引，具体对应关系，详见附录 1。
说明	(持久化配置，断电保存)用于对读写器当前的工作频段进行配置。

6.2.4. 配置与查询 EPC 基带参数

命名空间	GDotnet.Reader.Api.Protocol.Gx
配置类	MsgBaseSetBaseband
查询类	MsgBaseGetBaseband
属性	BaseSpeed: EPC 基带速率（可选）。 QValue: 默认 Q 值（可选）(0~15)。 Session: （可选）(0,Session0; 1,Session1; 2,Session2; 3,Session3)。 InventoryFlag: 盘存标志参数(可选)(0,仅用 Flag A 盘存;1,仅用 Flag B 盘存;2,轮流使用 Flag A 和 Flag B)。
说明	(持久化配置，断电保存)用于配置读写器使用的基带参数。

6.2.5. 配置与查询标签上传参数

命名空间	GDotnet.Reader.Api.Protocol.Gx
配置类	MsgBaseSetFreqRange
查询类	MsgBaseGetFreqRange
属性	RepeatedTime：重复标签过滤时间（可选）（表示在一个读卡指令执行周期内，在指定的重复过滤时间内相同的标签内容只上传一次，0~65535，时间单位：10ms）。 RssiTV: RSSI 阈值（可选）（标签 RSSI 值低于阈值时标签数据将不上传并丢弃）。
说明	(持久化配置，断电保存)配置标签主动上传参数。

6.2.6. 读 EPC 标签

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	MsgBaseInventoryEpc
属性	AntennaEnable：天线端口（使用天线枚举，详见快速上手） InventoryMode：连续/单次读取（0：单次读取模式，读写器尽在各个使能的天线上进行一轮读卡操作便结束读卡操作并自动进入空闲状态; 1：连续读取模式，读写器一直进行读卡操作直到读写器收到停止指令后结束读卡) Filter：选择读取参数（可选）（详见参数说明） ReadTid: TID 读取参数（可选）（详见参数说明） ReadUserdata：用户数据区读取参数（可选）（详见参数说明） ReadReserved：保留区读取参数（可选）（详见参数说明） HexPassword：访问密码（可选）
说明	用于配置读写器的标签读取参数并启动读卡操作，任何读取标签数据操作都需要先获取到标签 EPC 码，所以任何读卡操作都会得到 EPC 码。

6.2.7. 写 EPC 标签

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	MsgBaseWriteEpc
属性	AntennaEnable：天线端口（使用天线枚举，详见快速上手） Area：待写入的标签数据区(0，保留区；1，EPC 区；2，TID 区；3，用户数据区) Start：待写入标签数据区的字起始地址 HexWriteData：待写入的数据内容（可选）(16 进制) BwriteData：待写入的数据内容 Filter：选择读取参数（可选）（详见参数说明） HexPassword：访问密码（可选）
说明	➢ 读写器对 EPC 标签进行写操作，本指令定义的写操作为单次操作 ➢ ISO18000-6C 协议规定读写操作最小数据单为字。 ➢ EPC 区由 CRC-16(第 0 个字) + PC(第 1 个字) + EPC 组成： CRC16：第 0 个字，不可写 。 PC：第 1 个字，前 5 个 bit 位表示 EPC 字长度，即 PC 计算方法为 EPC 的字长度左移 11 位。

6.2.8. 锁 EPC 标签

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	MsgBaseLockEpc
属性	AntennaEnable：天线端口（使用天线枚举，详见快速上手） Area：待锁定的标签数据区(0，灭活密码区；1，访问密码区；2，EPC 区；3，TID 区；4，用户数据区) Mode：锁操作类型(0，解锁；1，锁定；2，永久解锁；3，永久锁定) Filter：选择读取参数（可选）（详见参数说明） HexPassword：访问密码（可选）
说明	对标签进行锁或解锁操作，本指令定义的操作为单次操作

6.2.9. 灭活 EPC 标签

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	MsgBaseDestoryEpc
属性	AntennaEnable：天线端口（使用天线枚举，详见快速上手） HexPassword：销毁密码 Filter：选择读取参数（可选）（详见参数说明）
说明	对标签进灭活操作，进行灭活后的标签将永久失效，此操作为不可逆操作。本指令定义的操作为单次操作

6.2.10. 读 6B 标签

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	MsgBaseInventory6b
属性	AntennaEnable：天线端口（使用天线枚举，详见快速上手） InventoryMode：连续/单次读取（0：单次读取模式，读写器尽在各个使能的天线上进行一轮读卡操作便结束读卡操作并自动进入空闲状态; 1：连续读取模式，读写器一直进行读卡操作直到读写器收到停止指令后结束读卡) Area：读取内容(0，仅读取 6B TID；1，读取 6B TID+用户数据；2，仅读取用户数据) ReadUserdata：用户数据区读取参数（可选）（详见参数说明） HexMatchTid：待匹配 6B 标签的 TID 码（可选）(16 进制) BMatchTid：待匹配 6B 标签的 TID 码（可选）
说明	用于 ISO18000-6B 标签的数据读取操作

6.2.11. 写 6B 标签

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	MsgBaseWrite6b

属性	AntennaEnable: 天线端口（使用天线枚举，详见快速上手） HexMatchTid：待匹配 6B 标签的 TID 码（可选）(16 进制) BMatchTid：待匹配 6B 标签的 TID 码 Start：待写入标签数据区的字节起始地址 HexWriteData：待写入的数据内容（可选）(16 进制) BwriteData：待写入的数据内容
说明	对 6B 标签进行写操作，本指令定义的写操作为单次操作

6.2.12. 6B 标签锁定

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	MsgBaseLock6b
属性	AntennaEnable: 天线端口（使用天线枚举，详见快速上手） HexMatchTid：待匹配 6B 标签的 TID 码（可选）(16 进制) BMatchTid：待匹配 6B 标签的 TID 码 LockIndex：待锁定数据的字节地址
说明	对 6B 标签数据进行锁定操作，该操作不可撤销和逆转，本指令定义的锁定操作为单次操作

6.2.13. 6B 标签锁定查询

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	MsgBaseLock6bGet
属性	AntennaEnable: 天线端口（使用天线枚举，详见快速上手） HexMatchTid：待匹配 6B 标签的 TID 码（可选）(16 进制) BMatchTid：待匹配 6B 标签的 TID 码 LockIndex：待锁定查询数据的字节地址

说明	对 6B 标签数据锁定状态进行查询，本指令定义的锁定查询操作为单次操作

6.2.14. 读 GB 标签

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	MsgBaseInventoryGb
属性	AntennaEnable: 天线端口（使用天线枚举，详见快速上手） InventoryMode: 连续/单次读取 (0: 单次读取模式，读写器尽在各个使能的天线上进行一轮读卡操作便结束读卡操作并自动进入空闲状态; 1: 连续读取模式，读写器一直进行读卡操作直到读写器收到停止指令后结束读卡) Filter: 选择读取参数（可选）（详见参数说明） ReadTid: TID 读取参数（可选）（详见参数说明） ReadUserdata: 用户数据区读取参数（可选）（详见参数说明） HexPassword: 访问密码（可选）
说明	用于配置读写器的标签读取参数并启动读卡操作，任何读取标签数据操作都需要先获取到标签编码，所以任何读卡操作都会得到标签编码（EPC）。

6.2.15. 写 GB 标签

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	MsgBaseWriteGb
属性	AntennaEnable: 天线端口（使用天线枚举，详见快速上手） Area：待写入的标签数据区(0x10，标签编码区 0x20，标签安全区 0x30~0x3F，用户子区 0~15 区) Start：待写入标签数据区的字起始地址 HexWriteData：待写入的数据内容（可选）(16 进制) BwriteData：待写入的数据内容 Filter: 选择读取参数（可选）（详见参数说明） HexPassword: 访问密码（可选）

说明	➢ 读写器对 GB 标签进行写操作，本指令定义的写操作为单次操作 ➢ GB 协议规定读写操作最小数据单为字。

6.2.16. 锁 GB 标签

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	MsgBaseLockGb
属性	AntennaEnable：天线端口（使用天线枚举，详见快速上手） Area：待锁定的标签数据区(0x00 标签信息区，0x10 标签编码区，0x20 标签安全区，0x30~0x3F 用户子区 0~15) Mode：锁操作类型 0x00，可读可写。 0x01，可读不可写。 0x02，不可读可写。 0x03，不可读不可写。 0x11，安全模式设置为不需要鉴别；此操作区域必须为标签安全区。 0x12，安全模式设置为需要鉴别，不需要安全通信；此操作区域必须为标签安全区。 0x13，安全模式设置为需要鉴别，需要安全通信；此操作区域必须为标签安全区。 Filter：选择读取参数（可选）（详见参数说明） HexPassword：访问密码（可选）
说明	对标签进行锁或解锁操作，本指令定义的操作为单次操作

6.2.17. 灭活 GB 标签

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	MsgBaseDestoryGb

属性	AntennaEnable：天线端口（使用天线枚举，详见快速上手） HexPassword：销毁密码 Filter：选择读取参数（可选）（详见参数说明）
说明	对标签进灭活操作，进行灭活后的标签将永久失效，此操作为不可逆操作。本指令定义的操作为单次操作

7. 参数说明

7.1.1. 6C/GB 标签选择参数

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	ParamEpcFilter
属性	Area：要匹配的数据区(1，EPC 区；2，TID 区；3，用户数据区) (GB 标签，0x00 标签信息区，0x10 标签编码区，0x20 标签安全区， 0x30~0x3F 用户子区 0~15) BitStart：匹配数据起始位地址 BitLength：需要匹配的数据位长度 HexData：需要匹配的数据内容（可选）(16 进制) BData：需要匹配的数据内容
说明	可选参数 EPC 区前 32 位为 PC 值，故区别 EPC 时的起始位地址通常为 32。

7.1.2. 6C 标签读取 TID 参数

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	ParamEpcReadTid
属性	Mode：TID 读取模式配置，(0，TID 读取长度自适应，但最大长度不超过字节 1 定义的长度；1，按照字节 1 定义的长度读取 TID) Len：读写器需要读取 TID 数据的字(word，16bits，下同)长度

说明	可选参数

7.1.3. 6C 标签读取用户数据区参数

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	ParamEpcReadUserdata
属性	Start：起始字地址 Len：读写器需要读取的用户数据的字长度
说明	可选参数

7.1.4. 6B 标签读用户数据区参数

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	Param6bReadUserdata
属性	Start：用户数据起始字节地址 Len：用户数据字节长度
说明	可选参数

7.1.5. GB 标签读用户数据区参数

命名空间	GDotnet.Reader.Api.Protocol.Gx
类	ParamGbReadUserdata
属性	ChildArea：用户子区 Start：用户数据起始字节地址 Len：用户数据字节长度
说明	可选参数

8. 附录 1

读写器所支持的频段列表

索引	说明
0	国标 920~925MHz
1	国标 840~845MHz
2	国标 840~845MHz 和 920~925MHz
3	FCC，902~928MHz
4	ETSI，866~868MHz

9. 附录 2