1 引言 3
2 命名空间介绍 3
2.1 ModuleTech.Gen2 名字空间 3
Session 枚举 3
MemBank 枚举 3
Gen2LockAct 枚举 3
Gen2LockAction 类 4
Gen2TagFilter 类 5
2.2 ModuleLibrary 名字空间 7
ModuleLibrary.ModuleException 类 7
ModuleLibrary.FatalInternalException 类 7
ModuleLibrary.OpFaidedException 类 7
ModuleLibrary.HardwareAlertException 类 7
2.3 ModuleTech 名字空间 7
ModuleTech.ByteFormat 类 7
ModuleTech.SimpleReadPlan 类 8
ModuleTech.Region 枚举 9
ModuleTech.ReaderIPInfo 类 9
ModuleTech .ReaderIPInfo_Ex.WifiSetting 类 10
ModuleTech .ReaderIPInfo_Ex 类 11
ModuleTech.TagData 类 11
ModuleTech.TagFilter 接口 12
ModuleTech.TagLockAction 类 12
ModuleTech.TagProtocol 枚举 12
ModuleTech.TagReadData 类 13
ModuleTech.EmbededCmdData 类 13
ModuleTech.EmdSecureReadData 类 ..... 错误!未定义书签。
ModuleTech.AntPower 结构 14
ModuleTech.ReaderConfiguration 类 14
ModuleTech.Reader 类 15
2.4 ModuleTech.CustomCmd 命名空间 30
CustomCmdType 枚举 30
CustomPara 类 31
CustomResult 类 31
NXP_ChangeEASPara 类 31
NXP_EASAlarmPara 类 31
NXP_SetReadProtectPara 类 32
NXP_ResetReadProtectPara 类 32
ALIEN_Higgs3_BlockReadLockPara 类 32
NXP_EASAlarmResult 类 33
IMPINJ_M4_QtPara 类 33
IMPINJ_M4_QtResult 类 34
2.5 ModuleTech.ISO180006B 命名空间 34

ISO180006bLockAction 类.....*34*
ISO180006bTagData 类.....*35*
MemBank 枚举.....*35*
3 读写器生命周期.....**35**
4 出错处理.....**36**
5 线程安全性.....**37**

Module API 开发文档 version 1. 3.140603

1 引言

ModuleAPI 用于.net 环境，帮助客户快速开发各种应用。

2 命名空间介绍

2.1 ModuleTech.Gen2 名字空间

对标签进行读，写，锁，销毁的时候需要用到很多和 Gen2 协议相关的类和枚举类型，主要都定义在这个命名空间中。

Session 枚举

功能：定义了 GEN2 协议中的 4 个 Session

成员	描述
Session0	对应 Gen2 的 Session 0
Session1	对应 Gen2 的 Session 1
Session2	对应 Gen2 的 Session 2
Session3	对应 Gen2 的 Session 3

MemBank 枚举

功能： 对应 GEN2 标签的四个 bank

成员	描述
RESERVED	保留区 bank
EPC	EPC bank
TID	TID bank
USER	用户区 bank

Gen2LockAct 枚举

功能：定义了锁定操作标签内存的区域和锁定的类型。

成员	描述
ACCESS_LOCK	暂时锁定访问密码
ACCESS_PERMALOCK	永久锁定访问密码
ACCESS_UNLOCK	解锁访问密码

EPC_LOCK	暂时锁定 EPC 区
EPC_PERMALOCK	永久锁定 EPC 区
EPC_UNLOCK	解锁 EPC 区
KILL_LOCK	暂时锁定销毁密码
KILL_PERMALOCK	永久锁定销毁密码
KILL_UNLOCK	解锁销毁密码
TID_LOCK	暂时锁定 TID 区（此区一般出厂就被永久锁定 ）
TID_PERMALOCK	永久锁定 TID 区
TID_UNLOCK	解锁 TID 区
USER_LOCK	暂时锁定 USER 区
USER_PERMALOCK	永久锁定 USER 区
USER_UNLOCK	解锁 USER 区

Gen2LockAction 类

功能：用于标签的锁定操作

构造函数：

public Gen2LockAction(ModuleTech.Gen2.Gen2LockAct[] acts)

参数	描述
acts	用于存放需要执行的锁定操作

例子：

可以同时对标签的多个区域进行三种类型（暂时锁定，永久锁定，解锁）的锁操作。例如：如果同时对标签的 EPC 区暂时锁定，解锁 USER 区，永久锁定访问密码，代码片段如下：

```
Gen2LockAct [] acts = new Gen2LockAct[3];
acts[0] = Gen2LockAct.EPC_LOCK;
acts[1] = Gen2LockAct.TID_UNLOCK;
acts[2] = Gen2LockAct.ACCESS_PERMALOCK;
Gen2LockAction lockact = new Gen2LockAction(acts)
```

Gen2TagFilter 类

功能：用于处理当需要对标签数据进行过滤的情况，在执行对标签的 Inventory, 读，写，锁，销毁等的操作中都可以添加过滤条件，使得只对符合过滤条件的标签进行操作。

对于不需要过滤条件的操作，Reader 类中的方法的此类型参数可传入 null。

构造函数：

public Gen2TagFilter(int filterlen, byte[] filterdata, MemBank filterbank, int filteraddr, bool isInvert)

参数	描述
filterlen	用于比较的数据的长度（bit 为单位）
filterdata	用于比较的数据，当 filterlen 不为 8 的整数倍的时候，filterdata 长度为（filterlen-1） /8 + 1
filterbank	表示要过滤哪个 bank 的数据
filteraddr	过滤的起始地址，以 bit 为单位
isInvert	过滤规则,false 表示筛选出匹配用于比较的数据的标签，true 表示筛选出不匹配用于比较的数据的标签

例子：

在某个天线旁边有多个标签，其中有一个标签的EPC ID是以1101开头，其它标签都不是以1101开头，现在需要读出这个标签的TID,在读标签数据的方法中需要一个Gen2TagFilter类型的参数，来进行对标签的过滤，以保证读操作是作用在指定的标签上，则代码为：

```
int bitcnt = 0;
string binarystr = “1101”;
byte[] filterbytes = new byte[(binarystr.Length-1)/8+1];

for (int c = 0; c < filterbytes.Length; ++c)
     filterbytes[c] = 0;
```

```
foreach (Char ch in binarystr)
{
    if (ch == '1')
        filterbytes[bitcnt / 8] |= (byte)(0x01 << (7 - bitcnt % 8));
    bitcnt++;
}
Gen2TagFilter filter = new Gen2TagFilter(binarystr.Length, filterbytes,
ModuleTech.Gen2.MemBank.EPC, 32, false);
```

构造函数：此构造函数不再建议使用，仅仅是为了保持接口的兼容性

public Gen2TagFilter(byte[] filterdata, ModuleTech.Gen2.MemBank filterbank, int filteraddr, bool isInvert)

参数	描述
filterdata	用于比较的数据，长度必须是字（两个字节）的整数倍
filterbank	表示要过滤哪个 bank 的数据
filteraddr	过滤的起始地址，以 bit 为单位
isInvert	过滤规则,false 表示筛选出匹配用于比较的数据的标签，true 表示筛选出不匹配用于比较的数据的标签

例子：

在某个天线旁边有多个标签，其中有一个标签的 EPC ID 是以 0x1234 开头，其它标签都不是以 0x1234 开头，现在需要读出这个标签的 TID,在读标签数据的方法中需要一个 Gen2TagFilter 类型的参数，来进行对标签的过滤，以保证读操作是作用在指定的标签上，对于此例，应该这样构造 Gen2TagFilter：

```
Gen2TagFilter filter = new Gen2TagFilter(ByteFormat.FromHex(“1234”),
ModuleTech.Gen2.MemBank.EPC, 32, false);
```

其中 ByteFormat.FromHex 方法用于将 16 进制的字符串转化成二进制的 byte 数组。

注意：对于 PR_ONEANT 类型的读写器，除了 read 函数外的其它操作仅支持对 EPC 区的过滤，且过滤数据必须是完整的 pc 数据加 EPC 码数据。

2.2 ModuleLibrary 名字空间

ModuleLibrary.ModuleException 类

功能：是所有异常的基类

ModuleLibrary.FatalInternalException 类

功能：读写器出现不可恢复错误会抛出这个异常，这个时候需要析构读写器重新连接。

ModuleLibrary.OpFaidedException 类

功能：针对所有的标签操作在失败的时候都有可能产生的异常

ModuleLibrary.HardwareAlertException 类

功能：当硬件检测到有对硬件损害的条件发生时会抛出个异常，出现对硬件有可能造成损害的行为包括：

1， 在没有接天线的端口发射功率

2， 模块的温度过高，或者环境的温度过高

3， 读标签的时候天线近距离的面对金属面，如果发生功率很大，很容易使大功率的信号反射回天线，然后进入读写器内部，造成读写器功放的损伤

当检测到这个异常的时候应该立刻停止任何对读写器的操作，检查是否存在以上的错误操作。

2.3 ModuleTech 名字空间

ModuleTech.ByteFormat 类

功能：用于 16 进制字符串和 byte 数组以及 ushort 数组之间的转化。

FromHex 方法

public static byte[] FromHex(string hex)：

功能：将 16 进制字符串转化位 byte 数组

参数	描述
hex	需要被转化为 byte 数组的 16 进制字符串，字符串由 1…F 的字符组成

例子：

byte[] bepc = ByteFormat.FromHex(“1234abcd”);

ToHex 方法

public static string ToHex(ushort[] words)：

功能：将 ushort 数组转化为 16 进制字符串

参数	描述
words	需要被转化为 16 进制字符串的 ushort 数组

例子：

string readdata = ByteFormat.ToHex(new ushort[] {1,2,4});

ToHex

public static string ToHex(byte[] bytes)：

功能：将 byte 数组转化为 16 进制字符串

参数	描述
bytes	需要被转化为 16 进制字符串的 byte 数组

例子：

string epc = ByteFormat.ToHex(new byte[] {1,2,3,4});

ModuleTech.SimpleReadPlan 类

功能：用于指定操作执行的天线。对于 Read 操作可以指定多个天线，ModuleTech.ReadPlan

为基类，实际中不会使用这个基类。

构造函数:

public SimpleReadPlan(int[] antennas)

参数	描述
antennas	用于存放要指定的天线

例子：

针对某个 Read 操作，需要在天线 1，2，4 上执行

```
int [] ants = new int[]{1, 2, 4};
SimpleReadPlan plan = new SimpleReadPlan(ants);
```

ModuleTech.Region 枚举

功能：用于指定读写器工作的频率规则，一般指定为 ModuleTech.Region.NA 就行

成员	描述
UNSPEC	未指定区域
CN	加拿大
EU	欧洲，版本 1（LBT）
EU2	欧洲， 版本 2
EU3	欧洲，版本 3(no LBT)
IN	印度
KR	韩国
JP	日本
NA	北美
PRC	中国（920-925）
PRC2	中国（840-845）
OPEN	全频段（860-960）

ModuleTech.ReaderIPInfo 类

功能：用于表示读写器的 ip 信息，用于设置和获取读写器 ip 信息，使用

ReaderIPInfo.Create 方法创建 ReaderIPInfo 类。

public static ModuleTech.ReaderIPInfo Create(string ip, string subnet, string gateway)

参数	描述
ip	只读属性，表示读写器的 ip 地址
subnet	只读属性，写器的子网掩码
gateway	只读属性，网关

属性：

属性	描述
GATEWAY	网关
IP	IP 地址
SUBNET	子网掩码

ModuleTech .ReaderIPInfo_Ex.WifiSetting 类

功能：用于表示读写器 wifi 设置
构造函数：
WifiSetting(AuthMode auth, string ssid, KeyType ktype, string key)

参数	描述
auth	Wifi 认证模式，合法值为：AuthMode_Open 表示 open 模式,AuthMode_Open_Wep 表示 wep open 模式，AuthMode_Shared_Wep 表示 wep shared 模式，AuthMode_Wpa_Psk 表示 wpa psk 模式，AuthMode_Wpa2_Psk 表示 wpa2 psk 模式
ssid	Wifi 设置的 ssid
ktype	wifi 密码类型，合法值为：KeyType_HEX 表示 16 进制字符，KeyType_ASC2 表示 ASC2 字符

key	密码

属性：

属性	描述
Auth	只读，wifi 认证模式
SSID	只读，wifi ssid
KEY	只读，wifi 密码

ModuleTech .ReaderIPInfo_Ex 类

功能：用于表示读写器的 ip 信息，和 ReaderIPInfo 不同的是加入了 wifi 参数的设置。

构造函数：

ReaderIPInfo_Ex(ReaderIPInfo ip, NetType type, WifiSetting wifi)

参数	描述
ip	IP，掩码，网关设置
type	网络类型，合法值为 NetType_Ethernet（表示以太网）和 NetType_Wifi（表示 wifi）
wifi	Wifi 参数设置

属性：

属性	描述
IPInfo	基础 ip 信息
NType	只读,网络类型
Wifi	只读，wifi 配置参数

ModuleTech.TagData 类

功能：用于 Reader 类的 ModuleTech.Reader.WriteTag(ModuleTech.TagFilter, ModuleTech.TagData)方法的参数

构造函数：

TagData(byte[] epc)

参数	描述
epc	用 byte[]表示的二进制数据

构造函数：

public TagData(string epc)

参数	描述
epc	用 16 进制字符串表示的二进制数据

注意：要写入的 epc id 必须是以字（两个字节，4 个 16 进制字符串）为最小单位。

ModuleTech.TagFilter 接口

功能：ModuleTech.Gen2.Gen2TagFilter 实现了这个接口,实际中都是使用

ModuleTech.Gen2.Gen2TagFilter 类

ModuleTech.TagLockAction 类

功能：是 ModuleTech.Gen2.Gen2LockAction 的基类，实际中都是使用

ModuleTech.Gen2.Gen2LockAction

ModuleTech.TagProtocol 枚举

功能：表示空中接口协议

成员	描述
EPC0_IMPINJ	IMPINJ 的 EPC0
EPC0_MATRICS	MATRICS 的 EPC0
EPC1	EPC1 协议
GEN2	GEN2 协议
ISO18000_6B	ISO 18000-6B 协议
NONE	未设置
UCODE	UCODE 协议

IPX64	IPX64 协议
IPX256	IPX256 协议

ModuleTech.TagReadData 类

功能：作为 Read 方法返回的结果，是读写器 Inventory 操作的结果。

属性：

属性	描述
Antenna	只读属性，表示哪个天线读到了标签
EPC	只读属性，epc 的二进制表示形式
EPCString	只读属性，epc 的 16 进制字符串的表示形式
ReadCount	只读属性，读到的次数
Tag	只读属性，epc 用 TagData 的表示形式
Time	只读属性，标签被读到时的时间戳
EbdData	只读属性，如果 Inventory 操作中嵌入了读指令，则此属性返回读指令读到的数据，如果没有嵌入读指令或者嵌入的读指令在读此标签的数据的时候失败了，则为 null
Rssi	只读属性，标签信号强度

ModuleTech.EmbededCmdData 类

功能：在 Inventory 操作中除了返回标签的 EPC ID 外，还可以指定读某个 bank 的数据。

构造函数：

public EmbededCmdData(MemBank bank_, UInt32 addr, byte datacnt)

参数	描述
bank_	指定嵌入读指令要读的 bank
addr	在 bank 中的起始地址（以块为单位）
datacnt	要读的字节数

属性：

属性	描述

Bank	只读属性，表示要读取的 bank
StartAddr	只读属性，表示 bank 中的起始地址（以块为单位）
ByteCnt	只读属性，表示要读的字节数

ModuleTech.AntPower 结构

功能：表示单个天线的读写功率设置

构造函数：

public AntPower(byte aid, UInt16 rpwr, UInt16 wpwr)

参数	描述
aid	天线的 id 号，根据读写器类型不同，只能取值为读写器存在的天线端口号
rpwr	天线的读功率,单位为 centi-dBm
wpwr	天线的写功率，单位为 centi-dBm

属性：

属性	描述
AntId	读写属性，代表天线 id 号
ReadPower	读写属性，天线的读功率
WritePower	读写属性，天线的写功率

ModuleTech.ReaderConfiguration 类

功能：用于表示永久保存参数时的操作行为

构造函数:

public ReaderConfiguration(SaveConfCode sccode)

参数	描述
sccode	SaveConf_Save 表示永久保存配置， SaveConf_Erase 表示擦除永久保存的配置， 读写器上电后使用默认配置。

属性：

属性	描述
ReaderConfCode	读写属性，SaveConf_Save 表示永久保存配置，SaveConf_Erase 表示擦除永久保存的配置，读写器上电后使用默认配置

ModuleTech.Reader 类

功能：是最重要的类，封装所有对标签的操作，而且管理着读写器的生命周期。提供了对读写器参数进行配置的接口。方法 public object ParamGet(string key) 和方法 public void ParamSet(string key, object value)用于获取和设置读写器参数。在读写器内部有个表用于存储读写器的各个参数，每个参数都是使用一个键值来获取。键值是一个字符串。 有些参数是读写器自身的信息，这些参数一般在读写器的整个生命周期中不会变化。有些参数是针对某些标签操作的，这些参数需要在进行某些操作之前对参数进行设置。参数分为只读的，可读写的两类。

可读写的参数有：

参数含义	键值	值类型	说明
Ip 地址	IPAddress	ReaderIPInfo	设置了 ip 后会导致读写器重启，需要重新连接读写器
Gen2 协议 session	Gen2Session	ModuleTech.Gen2.Session	取值从 0 到 3，对于读少量但移动快速的标签应该使用 0，对于标签数量大但移动速度缓慢的情形使用 1
Gen2 协议 Q 值	Gen2Qvalue	int	取值从-1 到 15，-1 表示读写器自动调整 Q 值，从 0-15 表示使用静态的 Q 值
是否在发射信号前检查天线的物理连接	CheckAntConnection	bool	True 表示在发射信号前检查天线连接，如果天线没有连接，操作将失败。false 表示在发射信号前不检查天线连接。（不是所有的天线都可以被检测）

执行 Read 操作时的天线设置	ReadPlan	ReadPlan	切记不可在没有连接天线的端口执行 Read 操作。
地区设置	Region	Region	
执行除了 Read 操作外的其它标签访问操作的天线设置	TagopAntenna	int	只能选择一个天线，且必须选择一个天线。切记不要设置成未连接天线的端口。
执行标签访问操作时使用的协议	TagopProtocol	TagProtocol	在执行除了盘存外的标签操作前一定要设置操作协议
标签访问密码	AccessPassword	uint	当标签被锁定时，访问需要密码，操作前应设置这个参数
标签过滤条件	Singulation	TagFilter	执行 Read 操作时设置的过滤条件，如果不需要过滤条件应该设置为 null
除了 Read 操作外的其它针对标签操作的超时时间	OpTimeout	ushort	单位为毫秒，默认是 1000 毫秒，是标签操作方法会阻塞的最长时间，如果提前完成了操作则阻塞的时间少于此参数设置的时间
在 Inventory 操作中指	EmbededCmdOfInventory	EmbededCmdData	在执行 Read 操作时设置的嵌入读指令，如果不需要嵌入读指令则应该设置为 null，对于每次在 Read 操作中

定的附加读操作			发现的标签，读写器都将对其执行嵌入的读指令
在 Inventory 操作中指定的安全附加读操作	EmdSecureDataOfInventory	EmdSecureReadData	在执行 Read 操作时设置的嵌入读指令，如果不需要嵌入读指令则应该设置为 null，对于每次在 Read 操作中发现的标签，读写器都将对其执行嵌入的读指令，但此操作不同于 EmbededCmdOfInventory 操作，安全附加数据，是指在盘存的过程中可以使用固定密码或者推算密码的方法来读取标签的某个 bank 的数据。并可针对 Impinj Monza 4 IC 在处于公有视图的时候启用 QT peek 功能直接读取私有视图某个 bank 的数据
每个天线读写功率的具体设置	AntPowerConf	AntPower[]	建议同时对所有的天线进行设置，只有最后一次被设置的天线的功率值会生效，没有被设置的天线的功率值为由 ReadTxPower 和 WriteTxPower 代表的值，读写器启动后每个天线的默认读写功率为最大功率上限的三分之二（对于 slr1000 类型的读写器读功率和写功率必须相同）
设置读写器所支持的最大EPC码长度（单位为 bit）	MaxEPCLength	int	可以设置为 96 或者 496，读写器默认为 96，此配置 M6_A7_FOURANTS 读写器不支持，M6_A7_FOURANTS 读写器默认值为 496
设置读写器的跳频	FrequencyHopTable	uint[]	所设置的频点必须在读写器当前所工作的频段内，不同的区域选项工作

表			频段也不同。
Gen2 目标	Gen2Target	ModuleTech.Gen2.Target	对应于 gen2 空中接口的参数
Gen2 写模式	Gen2WriteMode	ModuleTech.Gen2.WriteMode	目前只支持WORD_ONLY（字写）和BLOCK_ONLY（块写）
Gen2 后向链路速率	gen2BLF	int	合法值为：40，250，400，640，单位为kHz
Gen2 编码	gen2tagEncoding	int	合法值为：0表示FM0，1表示miller2,2表示miller4,3表示miller8
Gen2Tari	Gen2Tari	ModuleTech.Gen2.Tari	
Iso180006 后向链路速率	Iso180006bBLF	int	合法值为：40，160，单位为kHz
数据项天线字段唯一性	IsTagDataUniqueByAnt	bool	标签缓存中的数据项是否使用天线id作为键值
数据项附加数据字段唯一性	IsTagDataUniqueByEmddata	bool	标签缓存中的数据项是否使用Inventory的附加数据作为键值
数据项 Rssi 记录行为	IsTagdataRecordHighestRssi	bool	标签缓存中的数据项是否只记录同一标签的最大Rssi值

注意：以上参数一旦设置了，如果没有再更改则会在 Reader 类的整个生命周期中起作用，如果针对某个操作需要使用不同的参数，应该在调用具体操作方法前，更改参数的设置，例如:在写第一个标签的 EPC 区时,需要提供访问密码,则应该在操作前先设置 AccessPassword 参数，写完这个标签后写下一个标签又不需要访问密码了，这时应该重新设置

AccessPassword 参数为 0。

只读的参数有：

参数含义	键值	值类型	说明
Bootloader 版本	BootloaderVersion	string	
天线端口数	AntennaPorts	Int []	读写器的所有天线端口
检测到的天线	ConnectedAntennas	int[]	不是所有的天线都可以被检测到
硬件版本	HardwareVersion	string	
Firmware 版本	SoftwareVersion	string	
Firmware 发布时间	SoftwareTimestamp	string	
功率最大值	RfPowerMax	int	读和写功率的上限，单位为 centi-dBm
功率最小值	RfPowerMin	int	读和写功率的下限，单位为 centi-dBm
支持的空中接口协议	SupportedProtocols	TagProtocol[]	
支持的地区	SupportedRegions	Region[]	

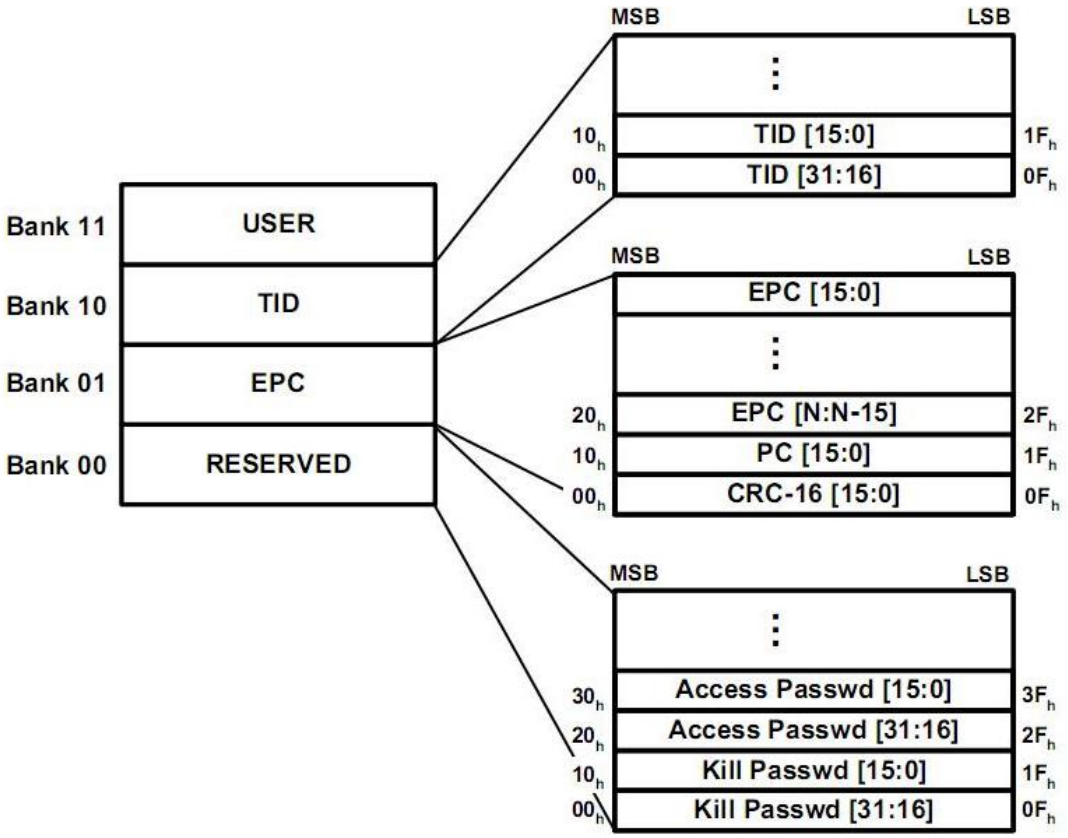
只写的参数有：

参数含	键值	值类型	说明

义			
永久保存配置	SaveConfiguration	ReaderConfiguration	用于保存读写器当前的参数配置，由 slr1000 和 slr1100 模块构建的纯串口/usb 口设备暂不支持保存参数配置。且不同读写器能保存的当前配置参数也略有差异，详见下表。

配置参数\读写器类型	M5e 系列	M6e 系列	slr1100 网口读写器
Gen2Session	Yes	Yes	Yes
Gen2Target	Yes	Yes	Yes
Gen2Qvalue	Yes	Yes	Yes
工作区域	Yes	Yes	Yes
gen2 编码	Yes（不支持 FM0）	Yes	No
天线检测	Yes	Yes	No
发射功率	Yes	Yes	Yes
数据天线唯一性	Yes	Yes	Yes
数据附件数据唯一性	Yes	Yes	Yes
记录最高 Rssi	Yes	Yes	Yes
跳频表	Yes	Yes	Yes
省电模式	Yes	Yes	No
最大 EPC 长度	Yes	No	No
Gen2 写模式	Yes	Yes	Yes
Gen2Trai	No	Yes	No
Gen2 后向散射速率	No	Yes	No
清点模式	No	Yes	No
Iso180006b 后向散射速率	No	Yes	No

所有的操作方法都是针对 Gen2 标签，Gen2 标签的内存被分成四个 bank，分别是 bank0，bank1，bank2，bank3，bank0 又称为保留区，存放着访问密码和销毁密码，每个密码都有 32bit。bank1 又称为 EPC 区，其中又含有 CRC 字段（16bit），PC 字段（16bit），EPC 码（最长可到达 496bit，一般为 96bit），bank2 又称为 TID 区，含有全球唯一的序列号，共 64bit。bank3 又称为 USER 区，容量是变长，不同的标签品牌容量不同，也有很多标签没有 bank3。



不同的 bank 中，标签操作函数操作的最小内存单元为块，每个块是 16 个 bit，从第 0 块开始编址，例如对于销毁密码而言，它所占有的内存区域为 bank0 的第 0 块和第 1 块。

所有的标签操作函数都可以指定一个超时时间，如果操作在超市时间内完成，函数就会提前于超市时间返回，否则函数最坏会阻塞直到超时。对于 Inventory 操作可以指定多个天线，对于其它的标签读，写，锁，销毁只能指定一个天线。

对于除了 Inventory 外的其它标签读，写，锁，销毁操作，如果天线场中有多个标签，第一个应答读写器的标签将会是被操作的标签。因此如果想要准确将一个操作作用到指定的标签中，应保证在天线的场区中只有一个标，或者通过设置过滤条件。

ParamGet 方法

public object ParamGet(string key)：

功能：获取读写器参数

参数	描述
key	参数的键值

放回值：

返回 object 类型的对象，需要用户强制转换为正确的参数类型。

例子：

获取读写器的 firmware 版本号, rdr 是 reader 类实例（以下所有例子都假设 rdr 是 reader 类实例）：

```
string fvir =  (string) rdr.ParamGet(“SoftwareVersion”);
```

ParamSet 方法

```
public void ParamSet(string key, object value)
```

功能：设置读写器参数

参数	描述
key	参数键值
value	具体的参数设置

例子：

设置读写器读功率为 27dbm：

```
rdr.ParamSet(“ReadTxPower”, 27*100);
```

Create 方法

```
public static Reader Create(string uriString, Region region, int antsnum)
```

功能：用于创建 Reader 类，但不用指定具体的读写器类型，只需要指定设备的天线端口数。

参数	描述
uriString	读写器 ip 地址或串口号
region	读写器工作区域
antsnum	读写器天线端口数量

例子：

```
try
{
    Reader rdr = Reader.Create(“192.168.0.250”, Region.NA, 4);
}
catch (Exception ex)
```

```
{
    MessageBox.Show(“连接失败”);
}
```

Disconnect()方法

ModuleTech.Reader.Disconnect()：

功能：断开连接，在程序结束的时候调用。

GPIGet 方法

public abstract bool GPIGet(int pin)

功能：获取 GPI 状态

参数	描述
pin	GPI 编号

返回值：

true 为高电平，false 为低电平。

例子：获取 GPI1 的状态

bool gpi1 = rdr. GPIGet(1);

GPOSet 方法

public abstract void GPOSet(int pin, bool state)：

功能：设置 GPO 的状态

参数	描述
pin	GPO 编号
state	要设置的状态， true 为高电平，false 为低电平

例子：

设置 GPO1 的状态

rdr. GPOSet(1, true);

KillTag 方法

public abstract void KillTag(ModuleTech.TagFilter target, uint password)

功能：销毁标签。

参数	描述
target	过滤条件，如果不需要过滤设为 null
password	销毁密码

例子：

在一堆标签中，有一个标签的 epc 码是以 0x1234 开头，它的销毁密码是 0x11112222。用天线 1 执行销毁此标签的操作

```
string kpasswdstr = “11112222”;
uint kpasswd = uint.Parse(kpasswdstr,
System.Globalization.NumberStyles.AllowHexSpecifier);
Gen2TagFilter filter = new Gen2TagFilter(ByteFormat.FromHex(“1234”),
ModuleTech.Gen2.MemBank.EPC, 32, false);
rdr.ParamSet("TagopAntenna", 1);
rdr.KillTag(filter, kpasswd);
```

LockTag 方法

public abstract void LockTag(ModuleTech.TagFilter target,

ModuleTech.TagLockAction action)：

功能：对标签执行锁定操作，可同时对多个锁定区域进行暂时锁，永久锁，解锁操作

参数说明：

参数	描述
target	过滤条件，如果不需要过滤设为 null
action	具体的锁定动作

例子：

在一个堆标签中，有一个标签的 TID 区是以 0x43211234 开头，它的访问密码是 0x33332222。对标签的 EPC 区暂时锁定，解锁 USER 区，永久锁定访问密码，使用天线 2 执行操作

```
Gen2TagFilter filter = new Gen2TagFilter(ByteFormat.FromHex(“43211234”),
ModuleTech.Gen2.MemBank.TID, 0, false);
string accesspwdstr = “33332222”;
uint passwd = uint.Parse(accesspwdstr,
System.Globalization.NumberStyles.AllowHexSpecifier);
rdr.ParamSet("AccessPassword", passwd);
Gen2LockAct [] acts = new Gen2LockAct[3];
acts[0] = Gen2LockAct.EPC_LOCK;
acts[1] = Gen2LockAct.TID_UNLOCK;
acts[2] = Gen2LockAct.ACCESS_PERMALOCK;
Gen2LockAction lockact = new Gen2LockAction(acts)
rdr.ParamSet("TagopAntenna", 2);
rdr.LockTag(filter, lockact);
```

注意：

在进行锁定之前必须先要将访问密码置为非 0

Read 方法

public abstract ModuleTech.TagReadData[] Read(int milliseconds)：

功能： 多标签识读，也即 Inventory 操作。

参数	描述
milliseconds	读操作的超时时间，单位为毫秒，方法至少会阻塞 milliseconds 毫秒，往往更长一点。

例子：

在天线 1，3，4 上执行 Inventory 操作,且只读 EPC 码以 0x1234 开头的标签，而且嵌入读指令，读 TID 区从第二个块开始的两个块的数据。

```
EmbededCmdData ecd = new EmbededCmdData(MemBank.TID, 2, 4);
rdr.ParamSet("EmbededCmdOfInventory", ecd);
Gen2TagFilter filter = new Gen2TagFilter(ByteFormat.FromHex(“1234”),MemBank.EPC,
32, false);
```

```
rdr.ParamSet("Singulation", filter);
int [] ants = new int[]{1, 3, 4};
SimpleReadPlan searchPlan = new SimpleReadPlan(ants);
rdr.ParamSet("ReadPlan", searchPlan);
TagReadData[] tags = rdr.Read(500);
```

注意：

对于在 n 个天线上读标签的操作，读写器会按照 ReadPlan 中的天线顺序逐个进行标签的搜索，当前一个天线搜索不到更多标签的时候就会自动切换到下一个天线上执行搜索。如果嵌入读指令需要读的是保留区的数据，而且访问密码或者销毁密码被锁定了，为了保证嵌入读指令执行成功还需要在调用 Read 方法前设定 AccessPassword 参数。

ReadTagMemWords 方法

public abstract ushort[] ReadTagMemWords(ModuleTech.TagFilter target, ModuleTech.Gen2.MemBank bank, int wordAddress, int wordCount)

功能：读标签不同内存区域的数据

参数	描述
target	过滤条件
bank	要读取的 bank 区域
wordAddress	在对应 bank 内的起始地址（以字为单位）
wordCount	要读取的块数(以字为单位)

返回值：读到的数据

例子：

要读取标签 TID 区从第 2 个块开始的连续两个块的数据

```
ushort[] readdata = rdr.ReadTagMemWords(null, 2, MemBank.TID, 2);
```

注意：

此例中未设置过滤条件和访问密码，在需要的时候可以做相应设置，如果天线场内有多个标签，则第一个响应读写器的标签的数据被读到，对于设置了过滤条件的情况，如果有多个标签满足过滤条件也是第一个响应读写器的标签的数据被读到。这个原则同样适用于写，销毁和锁操作。

ReadTagMemBytes 方法

public abstract byte[] ReadTagMemBytes(TagFilter target, ISO180006b.MemBank bank, int byteAddress, int byteCount)

功能：读取 180006b 标签的内存数据

参数	描述
target	必须指定为 ISO180006bTagData 类型，表示要操作的标签
bank	只能是 ISO180006b.MemBank .ISO180006BMEM
byteAddress	块起始地址
byteCount	要读的块数

返回值：读出的标签内存数据

例子：

读取 id 号为 E002345BE3256111 的标签从第 8 块开始的 9 个块的数据

byte[] data = rdr.ReadTagMemBytes(new ISO180006bTagData(ByteFormat.FromHex(E002345BE3256111)), ISO180006b.MemBank .ISO180006BMEM, 8, 9);

WriteTag 方法

public abstract void WriteTag(ModuleTech.TagFilter target, ModuleTech.TagData epc)

功能：写标签的 EPC 码

参数	描述
target	此方法不支持设置过滤条件，必须为 null
epc	要写入的 EPC 码

例子：

向标签中写入 EPC 码‘008600040000410000000280’

TagData epccode = new TagData(“008600040000410000000280”);

rdr.WriteTag(null, epccode);

注意：

对于天线场中有多个标签的情形，第一个应答读写器的标签将被写入。因此如果想要准确将一个 epc 码写到指定的标签中，应保证在天线的场区中只有一个标签。WriteTagMemWords 方法同样可以写 epc 码，只需要指定写入区域为 EPC 区，再设定起始地址为了 2，但和 WriteTag 的区别是，WriteTag 可以在写 epc 码的同时改变 EPC 区的 PC 字段，PC 字段保存着 epc 码长度的信息。且此方法不支持过滤条件，也不能写 EPC 被锁定的标签，此方法一般用于初始化标签。

WriteTagMemWords 方法

public abstract void WriteTagMemWords(ModuleTech.TagFilter target, ModuleTech.Gen2.MemBank bank, int address, ushort[] data)

功能：向标签的不同内存区域中写入数据

参数	描述
target	过滤条件
bank	要写的 bank
address	对应 bank 中的起始地址（以字为单位）
data	要写入的数据

例子：

向标签的 USER 区从第 3 个块开始的地址写入两个块的数据”0x44445555”

```
string datastr = ”0x44445555”;
ushort[] writedata = new ushort[2];
for (int a = 0; a < writedata.Length; ++a)
writedata[a] = ushort.Parse(datastr.Substring(a * 4, 4), System.Globalization.NumberStyles.AllowHexSpecifier);
rdr.WriteTagMemWords(null, MemBank.USER, 3, writedata);
```

注意：

在向标签中写数据的时候，一定要遵从 Gen2 协议对标签内存的规定，否则会导致写失败。

WriteTagMemBytes 方法

public abstract void WriteTagMemBytes(TagFilter target, ISO180006b.MemBank bank, int address, byte[] data)

功能：向 180006b 标签内存中写入数据

参数	描述
target	必须指定为 ISO180006bTagData 类型，表示要操作的标签
bank	只能是 ISO180006b.MemBank .ISO180006BMEM
address	块起始地址
data	要写入的数据

例子：

向 id 号为 E002345BE3256111 的标签从第 20 块开始写入 10 个字节的数据

```
byte[] wdata = new byte[10];
for (int i = 0; i < 10; ++i)
    wdata[i] = (byte)i;
rdr.WriteTagMemBytes(new
ISO180006bTagData(ByteFormat.FromHex(E002345BE3256111)),
ISO180006b.MemBank .ISO180006BMEM, 20, wdata);
```

CustomCmd 方法

public abstract CustomResult CustomCmd(CustomCmdType type, CustomPara para);

功能：执行标签的私有指令（slr1000 和 slr1100 模块构建的读写器暂时不支持此方法）

参数	描述
type	私有指令类型
para	私有指令参数，不同的私有指令其此参数的类型不同，详见下表。

返回值：不同私有指令返回值类型不同，详见下表

私有指令参数和返回值对照表

命令类型	参数类型	返回值类型

NXP_SetReadProtect	NXP_SetReadProtectPara	null
NXP_ResetReadProtect	NXP_ResetReadProtectPara	null
NXP_ChangeEAS	NXP_ChangeEASPara	null
NXP_EASAlarm	NXP_EASAlarmPara	NXP_EASAlarmResult
NXP_Calibrate		
ALIEN_Higgs2_PartialLoadImage		
ALIEN_Higgs2_FullLoadImage		
ALIEN_Higgs3_FastLoadImage		
ALIEN_Higgs3_LoadImage		
ALIEN_Higgs3_BlockReadLock	ALIEN_Higgs3_BlockReadLockPara	null
IMPINJ_M4_Qt	IMPINJ_M4_QtPara	IMPINJ_M4_QtResult or null

(注：表格中参数类型和返回值类型都未填的指令还暂为实现)

例子：

将 NXP 标签的 EAS 状态置为 set 状态。

```
NXP_ChangeEASPara ChEasPara = new NXP_ChangeEASPara(new byte[]{0,0,0,1}, true);
rdr.CustomCmd(CustomCmdType.NXP_ChangeEAS, ChEasPara);
```

2.4 ModuleTech.CustomCmd 命名空间

对标签的非 Gen2 的私有指令的支持。

CustomCmdType 枚举

功能：定义不同标签的私有指令

成员	描述
NXP_SetReadProtect	NXP 的 SetReadProtect 指令

NXP_ResetReadProtect	NXP 的 ResetReadProtect 指令
NXP_ChangeEAS	NXP 的 ChangeEAS 指令
NXP_EASAlarm	NXP 的 EASAlarm 指令
NXP_Calibrate	NXP 的 Calibrate 指令
ALIEN_Higgs2_PartialLoadImage	ALIEN_Higgs2 的 PartialLoadImage 指令
ALIEN_Higgs2_FullLoadImage	ALIEN_Higgs2 的 FullLoadImage 指令
ALIEN_Higgs3_FastLoadImage	ALIEN_Higgs3 的 FastLoadImage 指令
ALIEN_Higgs3_LoadImage	ALIEN_Higgs3 的 LoadImage 指令
ALIEN_Higgs3_BlockReadLock	ALIEN_Higgs3 的 BlockReadLock 指令
IMPINJ_M4_Qt	Impinj Monza4 的指令

CustomPara 类

功能：是 ModuleTech.Reader.CustomCmd 函数中所有私有指令参数类的基类，是纯虚类。

CustomResult 类

功能：是 ModuleTech.Reader.CustomCmd 函数返回结果类的基类，是纯虚类。

NXP_ChangeEASPara 类

功能：ModuleTech.Reader.CustomCmd 函数执行 NXP 的 ChangeEAS 指令时的输入参数。

构造函数：

public NXP_ChangeEASPara(byte [] pwd, bool isset)

参数	描述
pwd	标签的访问密码，4 个字节，注意：必须非零
isset	标签的 EAS 状态，true 为 set 状态，false 为 reset 状态

NXP_EASAlarmPara 类

功能：ModuleTech.Reader.CustomCmd 函数执行 NXP 的 EASAlarm 指令时的输入参数。

构造函数：

public NXP_EASAlarmPara(byte dr, byte mc, byte tre)

参数	描述
dr	Divide Ratio as Per Gen2
mc	Miller Cycles
tre	TrExt as Per Gen2

NXP_SetReadProtectPara 类

功能：ModuleTech.Reader.CustomCmd 函数执行 NXP 的 SetReadProtect 指令时的输入参数。

构造函数：

public NXP_SetReadProtectPara(byte[] pwd)

参数	描述
pwd	标签的访问密码，4 个字节，注意：必须非零

NXP_ResetReadProtectPara 类

功能：ModuleTech.Reader.CustomCmd 函数执行 NXP 的 ResetReadProtect 指令时的输入参数。

构造函数：

public NXP_ResetReadProtectPara(byte[] pwd)

参数	描述
pwd	标签的访问密码，4 个字节，注意：必须非零

ALIEN_Higgs3_BlockReadLockPara 类

功能：ModuleTech.Reader.CustomCmd 函数执行 ALIEN_Higgs3 的 BlockReadLock 指令时的输入参数。

构造函数:

public ALIEN_Higgs3_BlockReadLockPara(byte blkbitmap, byte[] pwd)

参数	描述
blkbitmap	八个 bit 分别代表了 512bits 中的八个区域，每个区域 64bits。最高的一个 bit 对应着

	0-63bit，剩下的 bit 依次排列对应关系，当对应 bit 为 1 的时候，所以代表区域将施加读保护，为 0 将取消读保护。
pwd	标签的访问密码，4 个字节，注意：必须非零

NXP_EASAlarmResult 类

功能：ModuleTech.Reader.CustomCmd 函数执行 NXP 的 EASAlarm 指令时的实际返回结果类。

属性

属性	描述
EASAlarmData	标签的 EAS 数据

IMPINJ_M4_QtPara 类

功能：ModuleTech.Reader.CustomCmd 函数执行 Impinj Monza4 芯片的 QT 指令时代输入参数

构造函数:

public IMPINJ_M4_QtPara(byte[] acspwd)

执行读 QT 状态时使用的构造函数

参数	描述
acspwd	标签的访问密码

public IMPINJ_M4_QtPara(byte[] acspwd, IMPINJ_Qt_Persist_Type persisttype, IMPINJ_Qt_Range_Type rangetype, IMPINJ_Qt_Mem_Type memtype)

执行写QT状态时使用的构造函数

参数	描述
acspwd	标签的访问密码
persisttype	写 QT 状态的存储属性，IMPINJ_Qt_Persist_Temp 表示暂时转换，IMPINJ_Qt_Persist_Perm 表示永久转换
rangetype	标签被识别时是否减小识读距离，IMPINJ_Qt_Range_FarField 表示不减小标签识读距离,IMPINJ_Qt_Range_NearField 表示

	使标签处于近场状态。
memtype	标签处于哪个视图上，IMPINJ_Qt_Mem_Private 表示私有视图，IMPINJ_Qt_Mem_Public 表示公有视图

IMPINJ_M4_QtResult 类

功能：ModuleTech.Reader.CustomCmd 函数执行 Impinj Monza4 芯片的 QT 读指令时实际返回的结果类

属性	描述
RangeType	表示标签的近远场属性
MemType	表示标签的存储视图属性

2.5 ModuleTech.ISO180006b 命名空间

ISO180006bLockAction 类

功能：用于 180006b 标签的锁定操作

构造函数：

public ISO180006bLockAction(int startaddr)

功能：用于锁定单块

参数	描述
startaddr	锁定块的地址

public ISO180006bLockAction(int startaddr, int blkcnt)

功能：用于锁定多块，但必须是连续的块

参数	描述
startaddr	块起始地址
blkcnt	要锁定的块数

ISO180006bTagData 类

功能：用于指定 180006b 标签操作时候的标签 ID 号

构造函数：

public ISO180006bTagData(byte[] uidBytes)

参数	描述
uidBytes	字节描述的 180006b 标签的 id 号

MemBank 枚举

功能：表示 180006b 标签的存储空间

成员	描述
ISO180006BMEM	表示 180006b 标签的存储空间

3 读写器生命周期

首先调用 Create 方法创建 Reader 类，然后就可以调用各个标签操作方法，最后调用 Disconnect 方法断开与读写器的连接。

4 出错处理

Reader.Create 操作过程中抛出的任何异常都导致读写器创建失败，必须对这个操作捕获所有的异常。

Reader 类中的所有方法都应该配有捕获异常的代码，凡是捕获到 System.IO.IOException 异常，都必须重新创建 Reader 类。

对于 Read，KillTag，LockTag，ReadTagMemWords，WriteTag，WriteTagMemWords，

ParamGet，ParamSet 这几个操作，如果捕获了 FatalInternalException 异常则意味着读写器内部出现错误，应该重新创建读写器。如果捕获到 OpFaidedException 异常则仅仅意味着此次操作失败。

如果捕获到 HardwareAlertException 异常往往是由于误操作导致产生了可能损坏模块的条件，这些误操作包括：在没有接天线的端口发射功率，使用了质量不合格的天线， 环境温度过高，天线发射的功率被近距离的金属板反射到天线上。所以这个时候最好关闭读写器，检查读写器的工作状态和工作环境。

5 线程安全性

当前版本开发包的所有函数在针对同一个读写器操作时不支持多线程环境，在多线程运行环境中需要用户自己确保不发生竞争条件。对于不同读写器没有这个限制。