

# AsReader

## AsReaderDesktop

### SDK 用户开发指导手册（GPIO）

## 修订历史

版本	修订内容	日期
1.0	初始版本	2019/12/30

## 目录

1	操作系统 :	1
2	函数详单 :	1
2.1	通用函数 :	1
2.2	EPCC1-G2 协议函数 :	2
3	函数的描述 :	4
3.1	通用函数 :	4
3.1.1	AutoOpenComPort():自动连接串口	5
3.1.2	OpenComPort() : 连接到指定串口	6
3.1.3	CloseComPort():关闭串口连接	7
3.1.4	CloseSpecComPort() : 关闭指定串口	7
3.1.5	GetReaderInformation():获得读写器的信息	7
3.1.6	SetAddress() : 写入读写器地址	8
3.1.7	SetInventoryScanTime() : 设置查询命令最大响应时间	9
3.1.8	SetRfPower () : 设置读写器功率	9
3.1.9	SetRegion () : 设置读写器工作频率	10
3.1.10	SetBaudRate () : 设置串口波特率	11
3.1.11	SetBeepNotification():蜂鸣器设置	11
3.1.12	GetGPIOStatus():GPIO 控制命令	12
3.1.13	SetGPIO():设置 GPIO	12
3.1.14	GetSeriaNo ():获取序列号	13
3.1.15	SetSaveLen:设置缓存的 EPC/TID 长度	13
3.1.16	GetSaveLen:读取缓存的 EPC/TID 长度	14
3.1.17	ReadBuffer_G2:缓存数据获取	14
3.1.18	ClearBuffer_G2:清缓存	15
3.1.19	GetBufferCnt_G2:查询缓存区标签数量	15
3.1.20	SetTagCustomFunction ():标签自定义功能	16
3.2	EPCC1-G2 协议函数 :	16
3.2.1	Inventory_G2 () : G2 查询命令	17
3.2.2	ReadData_G2 () : G2 读取数据命令	18
3.2.3	WriteData_G2 () : G2 写命令	19
3.2.4	BlockErase_G2 () : G2 块擦除命令	20
3.2.5	Lock_G2 () : G2 设定存储区读写保护状态命令	22
3.2.6	KillTag_G2 () : G2 销毁标签命令	23
3.2.7	WriteEPC_G2 () : G2 写 EPC 号命令	24
3.2.8	SetPrivacyByEPC_G2 () : G2 单张读保护设置命令	25
3.2.9	SetPrivacyWithoutEPC_G2 () : G2 多张读保护设置命令	26

3.2.10	ResetPrivacy_G2 () : G2 解锁读保护命令 .....	26
3.2.11	CheckPrivacy_G2 () : G2 测试标签是否被读保护命令 .....	27
3.2.12	EASConfigure_G2 () : G2 EAS 报警设置命令 .....	28
3.2.13	EASAlarm_G2 () : G2 EAS 报警探测命令 .....	29
3.2.14	BlockWrite_G2 () : G2 块写命令 .....	29
3.2.15	ExtReadData_G2 () : G2 扩展读 .....	30
3.2.16	ExtWriteData_G2 () : G2 扩展写 .....	32
3.2.17	InventoryBuffer_G2 () : 带缓存查询命令 .....	33
3.2.18	GetMonza4QTWorkParamter_G2 () : 读取 Monza4QT 工作参数 .....	34
3.2.19	SetMonza4QTWorkParamter_G2 () : 设置 Monza4QT 工作参数 .....	35
4	其他返回值定义 .....	36
5	错误代码定义 .....	37

## 1 操作系统:

WINDOWS 2000/XP/7/8

## 2 函数详单:

**AsReaderDesktopGPIO.dll** 包括了如下的操作函数。

### 2.1 通用函数:

1. int AutoOpenComPort(int \*port,unsigned char \* ComAdr,unsigned char baud, int \*FrmHandle);
2. int OpenComPort(int Port, unsigned char \*ComAdr, unsigned char Baud, int \*FrmHandle);
3. int CloseComPort(void);
4. int CloseSpecComPort(int FrmHandle);
5. int GetReaderInformation(unsigned char \*ComAdr, unsigned char \*VersionInfo, unsigned char \*ReaderType, unsigned char \*TrType,unsigned char \* dmaxfre , unsigned char \*dminfre, unsigned char \*powerdBm,unsigned char \*ScanTime, unsigned char \*Ant, unsigned char \*BeepEn, unsigned char \*OutputRep, unsigned char \* CheckAnt,int FrmHandle);
6. int SetAddress(unsigned char \*ComAdr, unsigned char ComAdrData, int FrmHandle);
7. int SetInventoryScanTime(unsigned char \*ComAdr, unsigned char ScanTime, int FrmHandle);
8. int SetRfPower(unsigned char \*ComAdr, unsigned char powerDbm, int FrmHandle);
9. int SetRegion(unsigned char \*ComAdr,unsigned char dmaxfre, unsigned char dminfre,int FrmHandle);

- 10.int SetBaudRate (unsigned char \*ComAdr, unsigned char baud, int FrmHandle);
- 11.int SetBeepNotification(unsigned char \*ComAdr, unsigned char BeepEn, int FrmHandle);
- 12.int GetGPIOStatus(unsigned char \*ComAdr, unsigned char \*OutputPin, int FrmHandle);
- 13.int SetGPIO(unsigned char \*ComAdr, unsigned char OutputPin, int FrmHandle);
- 14.int GetSeriaNo(unsigned char \*ComAdr, unsigned char \*SeriaNo, int FrmHandle);
- 15.int SetSaveLen(unsigned char \*ComAdr, unsigned char SaveLen, int FrmHandle);
- 16.int GetSaveLen(unsigned char \*ComAdr,unsigned char \*SaveLen,int FrmHandle);
- 17.int ReadBuffer\_G2(unsigned char \*ComAdr, int \*Totalen, int \*CardNum, unsigned char \*pEPCList, int FrmHandle);
- 18.int ClearBuffer\_G2(unsigned char \*ComAdr, int FrmHandle);
- 19.int GetBufferCnt\_G2(unsigned char \*ComAdr, int \*Count, int FrmHandle);
- 20.int SetTagCustomFunction(unsigned char \*address, unsigned char \*InlayType, int FrmHandle);

## 2.2 EPCC1-G2 协议函数:

1. int Inventory\_G2(unsigned char \*ComAdr, unsigned char QValue, unsigned char Session,unsigned char MaskMem,unsigned char \*MaskAdr,unsigned char MaskLen,unsigned char \* MaskData, unsigned char MaskFlag, unsigned char AdrTID, unsigned char LenTID, unsigned char TIDFlag, unsigned char Target, unsigned char InAnt, unsigned char Scantime, unsigned char Fastflag,unsigned char \* EPCLenandEPC, unsigned char \*Ant, int \* Totalen, int \*CardNum,int FrmHandle);
2. int ReadData\_G2 (unsigned char \*ComAdr, unsigned char \* EPC, unsigned char Enum,unsigned char Mem, unsigned char WordPtr, unsigned char Num, unsigned char \* Password , unsigned char MaskMem, unsigned char\*MaskAdr, unsigned char MaskLen, unsigned char\*MaskData, unsigned char \* Data ,int \* errorcode,int FrmHandle);

3. int WriteData\_G2(unsigned char \*ComAdr, unsigned char \* EPC, unsigned char Wnum, unsigned char Enum,unsigned char Mem, unsigned char WordPtr,unsigned char \*Writedata,unsigned char \* Password, unsigned char MaskMem,unsigned char\*MaskAdr,unsigned char MaskLen,unsigned char \* MaskData,int \* errorcode,int FrmHandle);
4. int BlockErase\_G2 (unsigned char \*ComAdr, unsigned char \* EPC, unsigned char Enum,unsigned char Mem, unsigned char WordPtr, unsigned char Num, unsigned char \*Password,unsigned char MaskMem,unsigned char\* MaskAdr, unsigned char MaskLen,unsigned char \* MaskData, int \* errorcode,int FrmHandle);
5. int Lock\_G2 (unsigned char \*ComAdr, unsigned char \* EPC, unsigned char Enum, unsigned char select, unsigned char setprotect, unsigned char \* Password, unsigned char MaskMem,unsigned char\* MaskAdr, unsigned char MaskLen,unsigned char \* MaskData, int \* errorcode,int FrmHandle);
6. int KillTag\_G2 (unsigned char \*ComAdr, unsigned char \* EPC, unsigned char Enum,unsigned char \* Password, unsigned char MaskMem,unsigned char\* MaskAdr, unsigned char MaskLen,unsigned char \* MaskData, int \* errorcode,int FrmHandle);
7. int WriteEPC\_G2 (unsigned char \*ComAdr, unsigned char \* Password, unsigned char \* WriteEPC, unsigned char Enum, int \* errorcode,int FrmHandle);
8. int SetPrivacyByEPC\_G2 (unsigned char \*ComAdr, unsigned char \* EPC, unsigned char Enum,unsigned char \*Password, unsigned char MaskMem,unsigned char \*MaskAdr,unsigned char MaskLen,unsigned char \* MaskData,int \* errorcode,int FrmHandle);
9. int SetPrivacyWithoutEPC\_G2 (unsigned char \*ComAdr, unsigned char \* Password, int \* errorcode,int FrmHandle);
- 10.int ResetPrivacy\_G2 (unsigned char \*ComAdr, unsigned char \* Password, int \* errorcode,int FrmHandle);
- 11.int CheckPrivacy\_G2 (unsigned char \*ComAdr, unsigned char \*readpro, int \* errorcode,int FrmHandle);
- 12.int EASConfigure\_G2 (unsigned char \*ComAdr, unsigned char \* EPC, unsigned char Enum,unsigned char \* Password, unsigned char EAS, unsigned char MaskMem,unsigned char \*MaskAdr,unsigned char MaskLen,unsigned char \* MaskData, int \* errorcode,int FrmHandle);

- 13.int EASAlarm\_G2 (unsigned char \*ComAdr, int \* errorcode,int FrmHandle);
- 14.int BlockWrite\_G2(unsigned char \*ComAdr, unsigned char \* EPC, unsigned char Wnum, unsigned char Enum,unsigned char Mem, unsigned char WordPtr,unsigned char \*Writedata,unsigned char \* Password, unsigned char MaskMem,unsigned char\*MaskAdr,unsigned char MaskLen,unsigned char \* MaskData,int \* errorcode,int FrmHandle);
- 15.int ExtReadData\_G2 (unsigned char \*ComAdr, unsigned char \* EPC, unsigned char Enum,unsigned char Mem, unsigned char \*WordPtr, unsigned char Num, unsigned char \* Password , unsigned char MaskMem, unsigned char\*MaskAdr, unsigned char MaskLen, unsigned char\*MaskData, unsigned char \* Data ,int \* errorcode,int FrmHandle);
- 16.int ExtWriteData\_G2(unsigned char \*ComAdr, unsigned char \* EPC, unsigned char Wnum, unsigned char Enum,unsigned char Mem, unsigned char \*WordPtr,unsigned char \*Writedata,unsigned char \* Password, unsigned char MaskMem,unsigned char\*MaskAdr,unsigned char MaskLen,unsigned char \* MaskData,int \* errorcode,int FrmHandle);
- 17.int InventoryBuffer\_G2(unsigned char \*ComAdr, unsigned char QValue, unsigned char Session,unsigned char MaskMem,unsigned char \*MaskAdr,unsigned char MaskLen,unsigned char \* MaskData, unsigned char MaskFlag, unsigned char AdrTID, unsigned char LenTID, unsigned char TIDFlag, unsigned char Target, unsigned char InAnt, unsigned char Scantime, unsigned char Fastflag, int \* BufferCount, int \*TagNum,int FrmHandle);
- 18.int GetMonza4QTWorkParamter\_G2(unsigned char \*address, unsigned char \*EPC, unsigned char ENum, unsigned char \*Password, unsigned char MaskMem, unsigned char \*MaskAdr, unsigned char MaskLen, unsigned char \*MaskData, unsigned char \*QTcontrol, int \*Errorcode, int FrmHandle);
- 19.int SetMonza4QTWorkParamter\_G2(unsigned char \*address, unsigned char \*EPC, unsigned char ENum, unsigned char QTcontrol, unsigned char \*Password, unsigned char MaskMem, unsigned char \*MaskAdr, unsigned char MaskLen, unsigned char \*MaskData, int \*Errorcode, int FrmHandle);

## 3 函数的描述:

### 3.1 通用函数:



## 3.1.1 AutoOpenComPort():自动连接串口

### 功能描述:

该函数用于自动识别与读写器连接的串口并且执行初始化操作，然后通过连接串口和读写器以创建通信连接。数据传输协议默认是波特率 57600bps，8 位数据，1 停止位，没有奇偶校验位。

在调用其它函数之前，您必须先连接串口和读写器。

### 应用:

```
int AutoOpenComPort(int* Port, unsigned char *ComAdr, unsigned char Baud,int *FrmHandle);
```

### 参数:

**Port:** 输出变量，COM1—COM9 与读写器连接的串口号。

**ComAdr:** 输入/输出变量，远距离读写器的地址。以广播地址（0xFF）调用此函数，函数将检测各个端口，并将检测到的连接有读写器的端口以及该端口上读写器的实际地址回写到指针 **Port** 和 **ComAdr** 所指变量中；以其它地址调用此函数，将在各个端口检测是否连接了具有指定 **ComAdr** 地址的读写器，并将检测到的端口号回写到指针 **Port** 所指变量中。

**Baud:** 输入变量，用该值设置或更改串口通讯控件的波特率。

baudrate	实际波特率
0	9600bps
1	19200 bps
2	38400 bps
5	57600 bps
6	115200 bps

**FrmHandle:** 输出变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

COM1-COM9 的含义如下:

```
#define COM1 1
#define COM2 2
#define COM3 3
#define COM4 4
#define COM5 5
#define COM6 6
#define COM7 7
#define COM8 8
#define COM9 9
```

### 返回:

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

## 3.1.2 OpenComPort(): 连接到指定串口

### 功能描述:

该函数用于指定串口初始化，并通过连接串口和读写器以创建通信连接。数据传输协议是默认波特率 57600bps，8 位数据，1 位停止位，没有奇偶校验位。在调用其它函数之前，您必须先连接串口和读写器。

### 应用:

```
int OpenComPort(int Port, unsigned char *ComAdr, unsigned char Baud,int *FrmHandle);
```

### 参数:

**Port:** 输入变量，COM1—COM9 常数。

**ComAdr:**输入/输出变量，远距离读写器的地址。以广播地址（0xFF）调用此函数，函数将检测指定端口，并将检测到的连接在此端口上的读写器的实际地址回写到指针 ComAdr 所指变量中；以其它地址调用此函数，将检测指定端口上是否连接了具有指定 ComAdr 地址的读写器。

**Baud:** 输入变量，用该值设置或更改串口通讯控件的波特率。

baudrate	实际波特率
0	9600bps
1	19200 bps
2	38400 bps
5	57600 bps
6	115200 bps

**FrmHandle:** 输出变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

COM1-COM9 的定义如下:

```
#define COM1 1  
#define COM2 2  
#define COM3 3  
#define COM4 4  
#define COM5 5  
#define COM6 6  
#define COM7 7  
#define COM8 8  
#define COM9 9
```

### 返回:

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

## 3.1.3 CloseComPort():关闭串口连接

### 功能描述:

该函数用于撤销串口和读写器的连接并释放相应资源。在一些开发环境里，串口资源必须在离开该程序前被释放，否则可能会造成系统不稳定。

### 应用:

```
int CloseComPort(void);
```

### 参数:

无

### 返回:

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

## 3.1.4 CloseSpecComPort(): 关闭指定串口

### 功能描述:

该函数用于关闭指定串口。

### 应用:

```
int CloseSpecComPort(int FrmHandle);
```

### 参数:

**FrmHandle:** 输入变量，与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。

### 返回:

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

## 3.1.5 GetReaderInformation():获得读写器的信息

### 功能描述:

执行该命令后，将获得读写器的信息，这其中包括读写器地址（**ComAdr**）和读写器软件版本（**VersionInfo**）的信息等多项信息。

应用:

```
int GetReaderInformation(unsigned char *ComAdr, unsigned char *VersionInfo,
unsigned char *ReaderType, unsigned char *TrType, unsigned char * dmaxfre ,
unsigned char *dminfre, unsigned char *powerdBm, unsigned char *ScanTime,
unsigned char *Ant, unsigned char *BeepEn, unsigned char *OutputRep, unsigned
char * CheckAnt, int FrmHandle);
```

参数:

**ComAdr:**输入/输出变量，远距离读写器的地址。以广播地址（0xFF）调用此函数，ComAdr 将返回读写器的实际地址，以其它地址调用此函数，将由 ComAdr 地址指定的读写器执行此函数命令。

**VersionInfo:** 指向输出数组变量（输出的是每字节都转化为字符的数据），远距离读写器版本信息，长度 2 个字节。第 1 个字节为版本号，第 2 个字节为子版本号。

**ReaderType:**输出变量，读写器类型代码，0x0F 代表 UHFREADER86。

**TrType:**输出变量读写器协议支持信息，具体定义请参见用户手册。(bit1 为 1 表示支持 18000-6c 协议，其它位保留。)

**dmaxfre:** 输出变量，Bit7-Bit6 用于频段设置用；Bit5-Bit0 表示当前读写器工作的最大频率，具体定义请参见用户手册。

**dminfre:**输出变量，输出变量，Bit7-Bit6 用于频段设置用；Bit5-Bit0 表示当前读写器工作的最小频率，具体定义请参见用户手册。

**PowerdBm:** 输出变量，读写器的输出功率。范围是 0 到 30。

**ScanTime:**输出变量，读写器询查命令最大响应时间。

**Ant:** 输出变量，保留。

**BeepEn:** 输出变量，蜂鸣器开关信息。

**OutputRep:** 输出变量，保留。

**CheckAnt:** 输出变量，保留

**FrmHandle:** 输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

返回:

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

### 3.1.6 SetAddress(): 写入读写器地址

功能描述:

执行该命令后，读写器将会把读写器地址改为用户给定的值，并把这个值写入 EEPROM 保存。出厂时默认值是 0x00。允许用户的修改范围是 0x00~0xfe。当用户写入的值是 0xff 时，读写器将会自动恢复成默认值 0x00。

应用:

```
int SetAddress(unsigned char *ComAdr, unsigned char ComAdrData, int FrmHandle);
```

参数:

**ComAdr**: 输入变量, 原先的读写器地址

**ComAdrData**: 输入变量, 一个字节, 待写入的读写器地址

**FrmHandle**: 输入变量, 返回与读写器连接端口对应的句柄, 应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功, 返回的句柄值为-1.

返回:

如果该函数调用成功, 返回一个零值。

否则, 返回非零值请查看其他返回值定义, 返回的错误代码请查看错误代码定义。

### 3.1.7 SetInventoryScanTime(): 设置查询命令最大响应时间

功能描述:

查询命令的最大响应时间范围是 3~255\*100ms, 默认值为 10\*100ms。

应用:

```
int SetInventoryScanTime(unsigned char *ComAdr, unsigned char ScanTime, int FrmHandle);
```

参数:

**ComAdr**: 输入变量, 读写器地址

**ScanTime**: 输入变量, 一个字节, 查询命令响应时间

**FrmHandle**: 输入变量, 返回与读写器连接端口对应的句柄, 应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功, 返回的句柄值为-1.

返回:

如果该函数调用成功, 返回一个零值。

否则, 返回非零值请查看其他返回值定义, 返回的错误代码请查看错误代码定义。

### 3.1.8 SetRfPower (): 设置读写器功率

功能描述:

本命令用来设置读写器功率。

应用:

```
int SetRfPower(unsigned char *ComAdr, unsigned char powerDbm, int FrmHandle);
```

参数:

**ComAdr**: 输入变量, 读写器地址

**Powerdbm**: 输入变量, 一个字节。读写器的输出功率。取值范围是 0~26。

**FrmHandle**: 输入变量, 返回与读写器连接端口对应的句柄, 应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功, 返回的句柄值为-1。

返回:

如果该函数调用成功, 返回一个零值。

否则, 返回非零值请查看其他返回值定义, 返回的错误代码请查看错误代码定义。

### 3.1.9 SetRegion (): 设置读写器工作频率

功能描述:

此命令设置读写器工作地上限频率, 下限频率。上限频率必须大于或等于下限频率。

应用:

```
int SetRegion (unsigned char *ComAdr, unsigned char dmaxfre, unsigned char dminfre,int FrmHandle);
```

参数:

**ComAdr**: 输入变量, 读写器地址

**dmaxfre**: 输入变量, Bit7-Bit6 用于频段设置用; Bit5-Bit0 表示当前读写器工作的最大频率, 具体定义请参见用户手册。

**Dminfre**: 输入变量, Bit7-Bit6 用于频段设置用; Bit5-Bit0 表示当前读写器工作的最小频率, 具体定义请参见用户手册。

**FrmHandle**: 输入变量, 返回与读写器连接端口对应的句柄, 应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功, 返回的句柄值为-1。

返回:

如果该函数调用成功, 返回一个零值。

否则, 返回非零值请查看其他返回值定义, 返回的错误代码请查看错误代码定义。

## 3.1.10 SetBaudRate (): 设置串口波特率

### 功能描述:

此命令用来更改读写器的串口波特率。

### 应用:

```
int SetBaudRate (unsigned char *ComAdr, unsigned char baud, int FrmHandle);
```

### 参数:

ComAdr : 输入变量, 读写器地址

baud: 输入变量, 一个字节。读写器上电后, 波特率默认为 57600bps。Baud 的范围是 0 ~ 6。其它值保留。其对应的波特率为:

baudrate	实际波特率
0	9600bps
1	19200 bps
2	38400 bps
5	57600 bps
6	115200 bps

FrmHandle: 输入变量, 返回与读写器连接端口对应的句柄, 应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功, 返回的句柄值为-1。

### 返回:

如果该函数调用成功, 返回一个零值。

否则, 返回非零值请查看其他返回值定义, 返回的错误代码请查看错误代码定义。

## 3.1.11 SetBeepNotification():蜂鸣器设置

### 功能描述:

该命令用于设置蜂鸣器开关。

### 应用:

```
Int SetBeepNotification(unsigned char *ComAdr, unsigned char BeepEn, int FrmHandle);
```

### 参数:

ComAdr: 输入变量, 读写器地址.

BeepEn: 输入变量, Bit0=0 时蜂鸣器关。

Bit0=1 时蜂鸣器开, 当读写器对标签操作成功有提示音。

Bit1~Bit7 位保留, 默认值 0。

FrmHandle: 输入变量, 返回与读写器连接端口对应的句柄, 应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功, 返回的句柄值为-1。

## 返回:

如果该函数调用成功, 返回一个零值。

否则, 返回非零值请查看其他返回值定义, 返回的错误代码请查看错误代码定义。

### 3.1.12 GetGPIOStatus():GPIO 控制命令

#### 功能描述:

该命令用于控制通用 GPIO 口输出, 默认输出为高电平。

#### 应用:

```
int GetGPIOStatus(unsigned char *ComAdr, unsigned char *OutputPin, int FrmHandle);
```

#### 参数:

**ComAdr:** 输入变量, 读写器地址.

**OutputPin:** 输出变量, GPIO 口输出状态。Bit0-Bit1 代表 IN1-IN2 引脚状态, Bit4-Bit5 分别代表 Out1-Out2 状态, 其他位保留。

**FrmHandle:** 输入变量, 返回与读写器连接端口对应的句柄, 应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功, 返回的句柄值为-1。

#### 返回:

如果该函数调用成功, 返回一个零值。

否则, 返回非零值请查看其他返回值定义, 返回的错误代码请查看错误代码定义

### 3.1.13 SetGPIO():设置 GPIO

#### 功能描述:

该命令用于设置 GPIO 口输出状态。

#### 应用:

```
int SetGPIO(unsigned char *ComAdr, unsigned char OutputPin, int FrmHandle);
```

#### 参数:

**ComAdr:** 输入变量, 读写器地址.

**OutputPin:** 输入变量, GPIO 口(Out1-Out2 引脚)输出状态。Bit4-Bit5 分别控制 Out1-Out2 引脚, 其余保留。

**FrmHandle:** 输入变量, 返回与读写器连接端口对应的句柄, 应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功, 返回的句柄值为-1。



## 返回:

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义

### 3.1.14 GetSeriaNo ():获取序列号

#### 功能描述:

该命令用于获取读写器的唯一序列号。

#### 应用:

```
Int GetSeriaNo(unsigned char *ComAdr, unsigned char *SeriaNo, int FrmHandle);
```

#### 参数:

**ComAdr**: 输入变量，读写器地址.

**SeriaNo**: 输出数组，4字节的唯一序列号。

**FrmHandle**: 输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

#### 返回:

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义

### 3.1.15 SetSaveLen:设置缓存的 EPC/TID 长度

#### 功能描述:

该命令用于设置读写器缓存存储 EPC 或者 TID 的长度。

#### 应用:

```
int SetSaveLen(unsigned char *ComAdr, unsigned char SaveLen,int FrmHandle);
```

#### 参数:

**ComAdr**: 输入变量，读写器地址.

**SaveLen**: 输入变量，1个字节，规定缓存 EPC/TID 的最大长度。取值 0 时为 128bit 长度，即 16 个字节。取值 1 时为 496bit 长度，即 62 个字节

**FrmHandle**: 输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

## 返回:

如果该函数调用成功，返回一个零值。  
否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

### 3.1.16 GetSaveLen:读取缓存的 EPC/TID 长度

#### 功能描述:

该命令用于读取读写器缓存存储 EPC 或者 TID 的长度。

#### 应用:

```
int GetSaveLen(unsigned char *ComAdr, unsigned char *SaveLen, int FrmHandle);
```

#### 参数:

ComAdr: 输入变量，读写器地址。

SaveLen: 输出变量，1 个字节，规定缓存 EPC/TID 的最大长度。取值 0 时为 128bit 长度，即 16 个字节。取值 1 时为 496bit 长度，即 62 个字节

FrmHandle: 输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

#### 返回:

如果该函数调用成功，返回一个零值。  
否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

### 3.1.17 ReadBuffer\_G2:缓存数据获取

#### 功能描述:

该命令用于获取读写器存储区中所有标签信息。响应完成后，缓存中的数据并不丢失，可以多次提取。

#### 应用:

```
int ReadBuffer_G2(unsigned char *ComAdr, int *TotalLen, int *CardNum, unsigned char *pEPCList, int FrmHandle);
```

#### 参数:

ComAdr: 输入变量，读写器地址;

TotalLen: 输出变量，返回的 pEPCList 数组字节大小;

CardNum: 输出变量，返回的标签张数;

pEPCList: 输出数组，大小是 TotalLen 个字节，读取的标签信息;

缓存中每个电子标签的 EPC/TID 数据。

pEPCList				
Ant	Len	EPC/TID	RSSI	Count
0xXX	0xXX	nBytes	0xXX	0xXX

**FrmHandle:** 输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

**返回:**

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

### 3.1.18 ClearBuffer\_G2:清缓存

**功能描述:**

该命令用于清空读写器存储区中所有标签信息。

**应用:**

`Int ClearBuffer_G2(unsigned char *ComAdr ,int FrmHandle);`

**参数:**

**ComAdr:** 输入变量，读写器地址。

**FrmHandle:** 输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

**返回:**

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

### 3.1.19 GetBufferCnt\_G2:查询缓存区标签数量

**功能描述:**

该命令用于获取缓存区中的标签数量。

**应用:**

`Int GetBufferCnt_G2(unsigned char *ComAdr , int *Count,int FrmHandle);`

## 参数:

ComAdr: 输入变量, 读写器地址.;

Count: 输出变量, 缓存区中的标签数量;

FrmHandle: 输入变量, 返回与读写器连接端口对应的句柄, 应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功, 返回的句柄值为-1。

## 返回:

如果该函数调用成功, 返回一个零值。

否则, 返回非零值请查看其他返回值定义, 返回的错误代码请查看错误代码定义。

### 3.1.20 SetTagCustomFunction ():标签自定义功能

#### 功能描述:

该命令用于设置读写器启动某些类型标签的自定义工作机制, 以实现标签特定功能。

#### 应用:

```
int SetTagCustomFunction(unsigned char *address, unsigned char *InlayType, int FrmHandle);
```

#### 参数:

ComAdr: 输入变量, 读写器地址.

InlayType: 标签类型, 取值范围 0~254。

默认值为 0, 表示不指定标签类型。

取值 1 为启动 Monza4QT 标签的 Peek 功能(标签状态临时从 public 转换到 private), 它将影响标签的读数据、写数据、块写数据、写保护字和写 EPC 号等操作。

FrmHandle: 输入变量, 返回与读写器连接端口对应的句柄, 应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功, 返回的句柄值为-1。

#### 返回:

如果该函数调用成功, 返回一个零值。

否则, 返回非零值请查看其他返回值定义, 返回的错误代码请查看错误代码定义

## 3.2 EPCC1-G2 协议函数:

## 3.2.1 Inventory\_G2 (): G2 查询命令

### 功能描述:

查询命令的作用是检查有效范围内是否有符合协议的电子标签存在。

### 应用:

```
int Inventory_G2(unsigned char *ComAdr, unsigned char QValue, unsigned char Session, unsigned char MaskMem, unsigned char *MaskAdr, unsigned char MaskLen, unsigned char *MaskData, unsigned char MaskFlag, unsigned char AdrTID, unsigned char LenTID, unsigned char TIDFlag, unsigned char Target, unsigned char InAnt, unsigned char Scantime, unsigned char Fastflag, unsigned char *EPClenandEPC, unsigned char *Ant, int *TotalLen, int *CardNum, int FrmHandle);
```

### 参数:

ComAdr: 输入变量, 读写器地址。

QValue: 输入变量, Q 值, 范围 0-15;

Session: 输入变量, Session 值, 范围 0-3;

MaskMem: 输入变量, 一个字节, 掩码区。0x01: EPC 存储区; 0x02: TID 存储区; 0x03: 用户存储区。

MaskAdr: 输入数组, 2 个字节, 掩码的起始位地址 (单位: Bits)。范围 0~16383。

MaskLen: 一个字节, 掩码的位长度 (单位: Bits)。

MaskData: 输入数组, 掩码数据。MaskData 数据字节长度是 MaskLen/8。如果 MaskLen 不是 8 的整数倍, 则 MaskData 数据字节长度为 [MaskLen/8] 取整再加 1。不够的在低位补 0。

MaskFlag: 输入变量, 一个字节, 掩码标记位,

MaskFlag=1 掩码, MaskFlag=0 不掩码。

AdrTID: 输入变量, 查询 TID 区的起始地址。

LenTID: 输入变量, 查询 TID 区的数据字数。LenTID 取值为 0~15。

TIDFlag: 输入变量,

TIDFlag=1: 表示查询 TID 区;

TIDFlag=0: 表示查询 EPC;

Target: 1 个字节, 查询 EPC 标签时使用的 Target 值

0x00: Target 值使用 A;

0x01: Target 值使用 B;

InAnt: 1 个字节, 本次要进行查询的天线号。本模块固定 0x80;

Scantime: 1 个字节, 本次命令查询的最大时间, 范围 3-255(\*100ms);

Fastflag: 快速查询的标志位, 为 0 时不启用, 为 1 时启用, 此时要指定 Target, InAnt, Scantime 3 个参数。

EPClenandEPC: 指向输出数组变量 (输出的是每字节都转化为字符的数据)。是读到的电子标签的 EPC 数据, 是一张标签的 EPC 长度+一张标签的 EPC 号+RSSI, 依此累加。每个电子标签 EPC 号高字在前, 每一个字的最高位在前。

Ant: 输出变量, 一个字节, 查询到标签的天线, 本模块固定 0x01。

TotalLen: 输出变量, EPClenandEPC 的字节数。

**CardNum:** 输出变量，电子标签的张数。

**FrmHandle:** 输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

## 返回:

如果该函数调用成功，返回值:

0x01 询查时间结束前返回

0x02 询查时间结束使得询查退出

0x03 如果读到的标签数量无法在一条消息内传送完，将分多次发送。如果 **Status** 为 0x0D，则表示这条数据结束后，还有数据。

0x04 还有电子标签未读取，电子标签数量太多，MCU 存储不了

返回其他值，请查看其他返回值定义，返回的错误代码请查看错误代码定义。

## 3.2.2 ReadData\_G2 (): G2 读取数据命令

### 功能描述:

这个命令读取标签的整个或部分保留区、EPC 存储器、TID 存储器或用户存储器中的数据。从指定的地址开始读，以字为单位。

### 应用:

```
int ReadData_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Enum,unsigned char Mem, unsigned char WordPtr, unsigned char Num,unsigned char * Password,unsigned char MaskMem,unsigned char *MaskAdr, unsigned char MaskLen, unsigned char*MaskData, unsigned char * Data ,int * errorcode,int FrmHandle);
```

### 参数:

**ComAdr:** 输入变量，读写器地址。

**EPC:** 指向输入数组变量（输入的是每字节都转化为字符的数据）。是电子标签的 EPC 号。

**Enum:** 在(0x00~0x0F)范围内表示 EPC 号长度，以字为单位。EPC 的长度在 15 个字以内。此时不掩码。ENum 为 0xFF 时掩码。如果为其它值将返回参数错误信息。

**Mem:** 输入变量，一个字节。选择要读取的存储区。

0x00: 保留区;

0x01: EPC 存储器;

0x02: TID 存储器;

0x03: 用户存储器。

其他值保留。若命令中出现了其它值，将返回参数出错的消息。

**WordPtr:** 输入变量，一个字节。指定要读取的字起始地址。0x00 表示从第一个字(第一个 16 位存储体)开始读，0x01 表示从第 2 个字开始读，依次类推。

**Num:** 输入变量，一个字节。要读取的字的个数。不能设置为 0x00，将返回参数错误信息。Num 不能超过 120，即最多读取 120 个字。若 Num 设置为 0 或者超过了 120，将返回参数出错的消息。

**Password:** 指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32 位的访问密码的最高位在 PassWord 的第一字节(从左往右)的最高位，访问密码最低位在 PassWord 第四字节的最低位，PassWord 的前两个字节放置访问密码的高字。

**MaskMem:** 输入变量，一个字节，掩码区。0x01: EPC 存储区；0x02: TID 存储区；0x03: 用户存储区。

**MaskAdr:** 输入数组，2 个字节，掩码的起始位地址（单位：Bits）。范围 0~16383。

**MaskLen:** 一个字节，掩码的位长度（单位：Bits）。

**MaskData:** 输入数组，掩码数据。MaskData 数据字节长度是 MaskLen/8。如果 MaskLen 不是 8 的整数倍，则 MaskData 数据字节长度为[MaskLen/8]取整再加 1。不够的在低位补 0

**Data:** 指向输出数组变量（输出的是每字节都转化为字符的数据），是从标签中读取的数据。

**Errorcode:** 输出变量，一个字节，读写器返回响应状态为 0xFC 时，返回错误代码。

**FrmHandle:** 输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

## 返回:

如果该函数调用成功，返回一个零值，读到的数据在 Data 中。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

### 3.2.3 WriteData\_G2 (): G2 写命令

#### 功能描述:

这个命令可以一次性往保留内存、EPC 存储器、TID 存储器或用户存储器中写入若干个字。

#### 应用:

```
int WriteData_G2(unsigned char *ComAdr, unsigned char * EPC, unsigned char Wnum, unsigned char Enum,unsigned char Mem, unsigned char WordPtr,unsigned char *Writedata,unsigned char * Password, unsigned char MaskMem,unsigned char*MaskAdr,unsigned char MaskLen,unsigned char * MaskData,int * errorcode,int FrmHandle);
```

#### 参数:

**ComAdr:** 输入变量，读写器地址。

**EPC:** 指向输入数组变量（输入的是每字节都转化为字符的数据）。是电子标签的 EPC 号。



**Wnum:** 输入变量，待写入的字个数，一个字为 2 个字节。这里字的个数必须和实际待写入的数据个数相等。

**Enum:** 在(0x00~0x0F)范围内表示 EPC 号长度，以字为单位。EPC 的长度在 15 个字以内。此时不掩码。**ENum** 为 0xFF 时掩码。如果为其它值将返回参数错误信息。

**Mem:** 输入变量，一个字节。选择要写的存储区。

0x00: 保留区；

0x01: EPC 存储器；

0x02: TID 存储器；

0x03: 用户存储器。

其他值保留。若命令中出现了其它值，将返回参数出错的消息。

**WordPtr:** 输入变量，一个字节。指定要写入的字起始地址。指定要写入数据的起始地址。如果写的是 EPC 区，则会忽略这个起始地址。EPC 区总是规定从 EPC 区 0x02 地址(EPC 号的第一个字节)开始写。

**Writedata:** 指向输入数组变量（输入的是每字节都转化为字符的数据）。待写入的字。这是要写入到存储区的数据。比如，**WordPtr** 等于 0x02，则输出变量 **Data** 中第一个字(从左边起)写在 **Mem** 指定的存储区的地址 0x02 中，第二个字写在 0x03 中，依次类推。

**Password:** 指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32 位的访问密码的最高位在 **PassWord** 的第一字节(从左往右)的最高位，访问密码最低位在 **PassWord** 第四字节的最低位，**PassWord** 的前两个字节放置访问密码的高字。

**MaskMem:** 输入变量，一个字节，掩码区。0x01: EPC 存储区；0x02: TID 存储区；0x03: 用户存储区。

**MaskAdr:** 输入数组，2 个字节，掩码的起始位地址（单位：Bits）。范围 0~16383。

**MaskLen:** 一个字节，掩码的位长度（单位：Bits）。

**MaskData:** 输入数组，掩码数据。**MaskData** 数据字节长度是 **MaskLen/8**。如果 **MaskLen** 不是 8 的整数倍，则 **MaskData** 数据字节长度为  $\lceil \text{MaskLen}/8 \rceil$  取整再加 1。不够的在低位补 0

**Errorcode:** 输出变量，一个字节，读写器返回响应状态为 0xFC 时，返回错误代码。

**FrmHandle:** 输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

## 返回:

如果该函数调用成功，返回一个零值，完全写入。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

### 3.2.4 BlockErase\_G2 (): G2 块擦除命令

#### 功能描述:

此命令可以擦除标签的保留内存、EPC 存储器、TID 存储器或用户存储器的若干字。



## 应用:

```
int BlockErase_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Enum,unsigned char Mem, unsigned char WordPtr, unsigned char Num, unsigned char *Password,unsigned char MaskMem,unsigned char* MaskAdr, unsigned char MaskLen,unsigned char * MaskData, int * errorcode,int FrmHandle);
```

## 参数:

**ComAdr:** 输入变量，读写器地址。

**EPC:** 指向输入数组变量（输入的是每字节都转化为字符的数据）。是电子标签的 EPC 号。

**Enum:** 在(0x00~0x0F)范围内表示 EPC 号长度，以字为单位。EPC 的长度在 15 个字以内。此时不掩码。ENum 为 0xFF 时掩码。如果为其它值将返回参数错误信息。

**Mem:** 输入变量，一个字节。选择要擦除的存储区。

0x00: 保留区;

0x01: EPC 存储器;

0x02: TID 存储器;

0x03: 用户存储器。

其他值保留。若命令中出现了其它值，将返回参数出错的消息。

**WordPtr:** 输入变量，一个字节。指定要擦除的字起始地址 0x00 表示从第一个字(第一个 16 位存储体)开始擦除，0x01 表示从第 2 个字开始擦除，依次类推。当擦除 EPC 区时，WordPtr 必须大于等于 0x02，若小于 0x02，则返回参数错误消息。

**Num:** 输入变量，一个字节。指定要擦除的字的个数。从 WordPtr 指定的地址开始擦除，擦除 Num 指定个数的字。若 Num 为 0x00，则返回参数错误信息。

**Password:** 指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32 位的访问密码的最高位在 PassWord 的第一字节(从左往右)的最高位，访问密码最低位在 PassWord 第四字节的最低位，PassWord 的前两个字节放置访问密码的高字。

**MaskMem:** 输入变量，一个字节，掩码区。0x01: EPC 存储区; 0x02: TID 存储区; 0x03: 用户存储区。

**MaskAdr:** 输入数组，2 个字节，掩码的起始位地址（单位：Bits）。范围 0~16383。

**MaskLen:** 一个字节，掩码的位长度（单位：Bits）。

**MaskData:** 输入数组，掩码数据。MaskData 数据字节长度是 MaskLen/8。如果 MaskLen 不是 8 的整数倍，则 MaskData 数据字节长度为[MaskLen/8]取整再加 1。不够的在低位补 0。

**Errorcode:** 输出变量，一个字节，读写器返回响应状态为 0xFC 时，返回错误代码。

**FrmHandle:** 输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

## 返回:

如果该函数调用成功，返回一个零值，擦除成功。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

## 3.2.5 Lock\_G2 (): G2 设定存储区读写保护状态命令

### 功能描述:

这个命令可以设定保留区为可读写、永远可读写、带密码可读写、永远可读写，可以分别设定 EPC 存储器、TID 存储器和用户存储器为可写、永远可写、带密码可写、永远不可写。EPC 存储器、TID 存储器或用户存储器是永远可读的。而且，TID 存储器是只读的，永远都不可写。

### 应用:

```
int Lock_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Enum, unsigned char select, unsigned char setprotect, unsigned char * Password, unsigned char MaskMem,unsigned char* MaskAdr, unsigned char MaskLen,unsigned char * MaskData, int * errorcode,int FrmHandle);
```

### 参数:

**ComAdr:** 输入变量，读写器地址。

**EPC:** 指向输入数组变量（输入的是每字节都转化为字符的数据）。是电子标签的 EPC 号。

**Enum:** 在(0x00~0x0F)范围内表示 EPC 号长度，以字为单位。EPC 的长度在 15 个字以内。此时不掩码。ENum 为 0xFF 时掩码。如果为其它值将返回参数错误信息。

**Select:** 输入变量，一个字节。

0x00 时，控制 Kill 密码读写保护设定。

0x01 时，控制访问密码读写保护设定。

0x02 时，控制 EPC 存储器读写保护设定。

0x03 时，控制 TID 存储器读写保护设定。

0x04 时，控制用户存储器读写保护设定。

其它值保留，若出读写器接收到了其他值，将返回参数出错的消息。

**Setprotect:** 输入变量，一个字节。

当 Select 为 0x00 或 0x01，SetProtect 值代表的意义如下:

0x00: 设置为可读写

0x01: 设置为永远可读写

0x02: 设置为带密码可读写

0x03: 设置为永远不可读写

当 Select 为 0x02、0x03、0x04 时，SetProtect 值代表的意义如下:

0x00: 设置为可写

0x01: 设置为永远可写

0x02: 设置为带密码可写

0x03: 设置为永远不可写

**Password:** 指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32 位的访问密码的最高位在 PassWord 的第一字节(从左往右)的最高位，访问密码最低位在 PassWord 第四字节的最低位，PassWord 的前两个字节放置访问密码的高字。

**MaskMem:** 输入变量，一个字节，掩码区。0x01: EPC 存储区；0x02: TID 存储区；0x03: 用户存储区。

**MaskAdr:** 输入数组，2 个字节，掩码的起始位地址（单位：Bits）。范围 0~16383。

**MaskLen:** 一个字节，掩码的位长度（单位：Bits）。

**MaskData:** 输入数组，掩码数据。**MaskData** 数据字节长度是 **MaskLen/8**。如果 **MaskLen** 不是 8 的整数倍，则 **MaskData** 数据字节长度为  $[\text{MaskLen}/8]$  取整再加 1。不够的在低位补 0。

**Errorcode:** 输出变量，一个字节，读写器返回响应状态为 0xFC 时，返回错误代码。

**FrmHandle:** 输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

## 返回:

如果该函数调用成功，返回一个零值，设置成功。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

### 3.2.6 KillTag\_G2 (): G2 销毁标签命令

#### 功能描述:

这个命令用来销毁标签。标签销毁后，永远不会再处理读写器的命令。

#### 应用:

```
int KillTag_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Enum,unsigned char * Password, unsigned char MaskMem,unsigned char* MaskAdr, unsigned char MaskLen,unsigned char * MaskData, int * errorcode,int FrmHandle);
```

#### 参数:

**ComAdr:** 输入变量，读写器地址。

**EPC:** 指向输入数组变量（输入的是每字节都转化为字符的数据）。是电子标签的 EPC 号。

**Enum:** 在(0x00~0x0F)范围内表示 EPC 号长度，以字为单位。EPC 的长度在 15 个字以内。此时不掩码。ENum 为 0xFF 时掩码。如果为其它值将返回参数错误信息。

**Password:** 指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32 位的访问密码的最高位在 PassWord 的第一字节(从左往右)的最高位，访问密码最低位在 PassWord 第四字节的最低位，PassWord 的前两个字节放置访问密码的高字。

**MaskMem:** 输入变量，一个字节，掩码区。0x01: EPC 存储区；0x02: TID 存储区；0x03: 用户存储区。

**MaskAdr:** 输入数组，2 个字节，掩码的起始位地址（单位：Bits）。范围 0~16383。

**MaskLen:** 一个字节，掩码的位长度（单位：Bits）。

**MaskData:** 输入数组,掩码数据。**MaskData** 数据字节长度是 **MaskLen/8**。如果 **MaskLen** 不是 8 的整数倍, 则 **MaskData** 数据字节长度为  $[\text{MaskLen}/8]$ 取整再加 1。不够的在低位补 0。

**Errorcode:** 输出变量, 一个字节, 读写器返回响应状态为 **0xFC** 时, 返回错误代码。

**FrmHandle:** 输入变量, 返回与读写器连接端口对应的句柄, 应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功, 返回的句柄值为-1。

## 返回:

如果该函数调用成功, 返回一个零值, 销毁成功。

否则, 返回非零值请查看其他返回值定义, 返回的错误代码请查看错误代码定义。

### 3.2.7 WriteEPC\_G2 (): G2 写 EPC 号命令

#### 功能描述:

这个命令向电子标签写入 EPC 号。写入的时候, 天线有效范围内只能有一张电子标签。

#### 应用:

```
int WriteEPC_G2 (unsigned char *ComAdr, unsigned char * Password, unsigned char * WriteEPC, unsigned char Enum, int * errorcode,int FrmHandle);
```

#### 参数:

**ComAdr:** 输入变量, 读写器地址。

**Password:** 指向输入数组变量 (输入的是每字节都转化为字符的数据), 四个字节, 这四个字节是访问密码。**32** 位的访问密码的最高位在 **PassWord** 的第一字节(从左往右)的最高位, 访问密码最低位在 **PassWord** 第四字节的最低位, **PassWord** 的前两个字节放置访问密码的高字。

**WriteEPC:** 指向输入数组变量 (输入的是每字节都转化为字符的数据), 对电子标签写入的 EPC 号 (覆盖原 EPC 号)。

**Enum:** 在(1~31)范围内表示 EPC 号长度, 以字为单位。EPC 的长度在 31 个字以内。如果为其它值将返回参数错误信息。

**Errorcode:** 输出变量, 一个字节, 读写器返回响应状态为 **0xFC** 时, 返回错误代码。

**FrmHandle:** 输入变量, 返回与读写器连接端口对应的句柄, 应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功, 返回的句柄值为-1。

## 返回:

如果该函数调用成功, 返回一个零值, 写入成功。

否则, 返回非零值请查看其他返回值定义, 返回的错误代码请查看错误代码定义。

## 3.2.8 SetPrivacyByEPC\_G2 (): G2 单张读保护设置命令

### 功能描述:

这个命令根据电子标签的 EPC 号,对标签设置读保护,使得电子标签不能被任何命令读写,对标签进行查询操作,也无法得到电子标签的 EPC 号。仅对 NXP 的 UCODE EPC G2X 标签有效。

### 应用:

```
int SetPrivacyByEPC_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Enum,unsigned char *Password, unsigned char MaskMem,unsigned char *MaskAdr,unsigned char MaskLen,unsigned char * MaskData,int * errorcode,int FrmHandle);
```

### 参数:

**ComAdr:** 输入变量,读写器地址。

**EPC:** 指向输入数组变量(输入的是每字节都转化为字符的数据)。是电子标签的 EPC 号。

**Enum:** 在(0x00~0x0F)范围内表示 EPC 号长度,以字为单位。EPC 的长度在 15 个字以内。此时不掩码。ENum 为 0xFF 时掩码。如果为其它值将返回参数错误信息。

**Password:** 指向输入数组变量(输入的是每字节都转化为字符的数据),四个字节,这四个字节是访问密码。32 位的访问密码的最高位在 PassWord 的第一字节(从左往右)的最高位,访问密码最低位在 PassWord 第四字节的最低位,PassWord 的前两个字节放置访问密码的高字。

**MaskMem:** 输入变量,一个字节,掩码区。0x01: EPC 存储区; 0x02: TID 存储区; 0x03: 用户存储区。

**MaskAdr:** 输入数组,2 个字节,掩码的起始位地址(单位: Bits)。范围 0~16383。

**MaskLen:** 一个字节,掩码的位长度(单位: Bits)。

**MaskData:** 输入数组,掩码数据。MaskData 数据字节长度是 MaskLen/8。如果 MaskLen 不是 8 的整数倍,则 MaskData 数据字节长度为[MaskLen/8]取整再加 1。不够的在低位补 0。

**Errorcode:** 输出变量,一个字节,读写器返回响应状态为 0xFC 时,返回错误代码。

**FrmHandle:** 输出变量,返回与读写器连接端口对应的句柄,应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功,返回的句柄值为-1。

### 返回:

如果该函数调用成功,返回一个零值,读保护设置成功。

否则,返回非零值请查看其他返回值定义,返回的错误代码请查看错误代码定义。

## 3.2.9 SetPrivacyWithoutEPC\_G2 (): G2 多张读保护设置命令

### 功能描述:

这个命令可以为有效范围内的电子标签设定读保护。这个命令与前面一个任务的区别是，当有效范围内存在多张标签的时候，无法知道这个命令操作的是哪一张电子标签。如果要对多张标签进行操作，则标签的访问密码最好是相同的。仅对 NXP 的 UCODE EPC G2X 标签有效。

### 应用:

```
int SetPrivacyWithoutEPC_G2 (unsigned char *ComAdr, unsigned char * Password, int * errorcode,int FrmHandle);
```

### 参数:

**ComAdr:** 输入变量，读写器地址。

**Password:** 指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32位的访问密码的最高位在 **PassWord** 的第一字节(从左往右)的最高位，访问密码最低位在 **PassWord** 第四字节的最低位，**PassWord** 的前两个字节放置访问密码的高字。

**Errorcode:** 输出变量，一个字节，读写器返回响应状态为 0xFC 时，返回错误代码。

**FrmHandle:** 输出变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

### 返回:

如果该函数调用成功，返回一个零值，多张读保护设置成功。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

## 3.2.10 ResetPrivacy\_G2 (): G2 解锁读保护命令

### 功能描述:

这个命令用来给设置了读保护的标签解锁。用这个命令时，天线有效范围内只能放置一张要被解锁的电子标签。仅对 NXP 的 UCODE EPC G2X 标签有效。

### 应用:

```
int ResetPrivacy_G2 (unsigned char *ComAdr, unsigned char * Password, int * errorcode,int FrmHandle);
```

### 参数:

**ComAdr:** 输入变量，读写器地址。

**Password:** 指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32位的访问密码的最高位在 **PassWord** 的第一字节(从左往右)



的最高位，访问密码最低位在 PassWord 第四字节的最低位，PassWord 的前两个字节放置访问密码的高字。

**Errorcode:** 输出变量，一个字节，读写器返回响应状态为 0xFC 时，返回错误代码。

**FrmHandle:** 输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

## 返回:

如果该函数调用成功，返回一个零值，解锁读保护成功。说明: 对于不支持读保护设定的标签，认为没有被解锁。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

### 3.2.11 CheckPrivacy\_G2 (): G2 测试标签是否被读保护命令

#### 功能描述:

这个命令不能测试标签是否支持读保护锁定，只能测试标签是否被读保护锁定。对于不支持读保护锁定的电子标签，一致认为没有被锁定。

这个命令只能对单张电子标签进行操作，确保天线有效范围内只存在一张电子标签。仅对 NXP 的 UCODE EPC G2X 标签有效。

#### 应用:

```
int CheckPrivacy_G2 (unsigned char *ComAdr, unsigned char *Readpro , int *  
errorcode,int FrmHandle);
```

#### 参数:

**ComAdr:** 输入变量，读写器地址。

**Readpro:** 输出变量，一个字节。**Readpro** 为 00，电子标签没有被设置为读保护。

**Readpro** 为 01，电子标签被设置读保护。说明: 对于不支持读保护设定的标签，认为没有被设置读保护。

**Errorcode:** 输出变量，一个字节，保留。

**FrmHandle:** 输出变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

#### 返回:

如果该函数调用成功，返回一个零值。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

### 3.2.12 EASConfigure\_G2 (): G2 EAS 报警设置命令

#### 功能描述:

对电子标签的 EAS 状态位进行设置或复位。仅对 NXP 的 UCODE EPC G2 标签有效。

#### 应用:

```
int EASConfigure_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Enum,unsigned char * Password, unsigned char EAS, unsigned char MaskMem,unsigned char *MaskAdr,unsigned char MaskLen,unsigned char * MaskData, int * errorcode,int FrmHandle);
```

#### 参数:

**ComAdr:** 输入变量, 读写器地址。

**EPC:** 指向输入数组变量(输入的是每字节都转化为字符的数据)。是电子标签的 EPC 号。

**Enum:** 在(0x00~0x0F)范围内表示 EPC 号长度, 以字为单位。EPC 的长度在 15 个字以内。此时不掩码。ENum 为 0xFF 时掩码。如果为其它值将返回参数错误信息。

**Password:** 指向输入数组变量(输入的是每字节都转化为字符的数据), 四个字节, 这四个字节是访问密码。32 位的访问密码的最高位在 PassWord 的第一字节(从左往右)的最高位, 访问密码最低位在 PassWord 第四字节的最低位, PassWord 的前两个字节放置访问密码的高字。

**EAS:** 输入变量, 1 个字节。bit0 位为 0, 表示设置为关闭 EAS 报警; 为 1, 表示设置为打开 EAS 报警。bit1 – bit7 位默认为 0, 保留。

**MaskMem:** 输入变量, 一个字节, 掩码区。0x01: EPC 存储区; 0x02: TID 存储区; 0x03: 用户存储区。

**MaskAdr:** 输入数组, 2 个字节, 掩码的起始位地址(单位: Bits)。范围 0~16383。

**MaskLen:** 一个字节, 掩码的位长度(单位: Bits)。

**MaskData:** 输入数组, 掩码数据。MaskData 数据字节长度是 MaskLen/8。如果 MaskLen 不是 8 的整数倍, 则 MaskData 数据字节长度为[MaskLen/8]取整再加 1。不够的在低位补 0。

**Errorcode:** 输出变量, 一个字节, 读写器返回响应状态为 0xFC 时, 返回错误代码。

**FrmHandle:** 输出变量, 返回与读写器连接端口对应的句柄, 应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功, 返回的句柄值为-1。

#### 返回:

如果该函数调用成功, 返回一个零值, EAS 报警设置成功。

否则, 返回非零值请查看其他返回值定义, 返回的错误代码请查看错误代码定义。



## 3.2.13 EASAlarm\_G2 (): G2 EAS 报警探测命令

### 功能描述:

该命令检测电子标签的 EAS 报警。仅对 NXP 的 UCODE EPC G2 标签有效。

### 应用:

```
int EASAlarm_G2 (unsigned char *ComAdr, int * errorcode,int FrmHandle);
```

### 参数:

ComAdr: 输入变量, 读写器地址。

Errorcode: 输出变量, 一个字节, 保留。

FrmHandle: 输出变量, 返回与读写器连接端口对应的句柄, 应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功, 返回的句柄值为-1。

### 返回:

EAS 报警, 返回零值。

无 EAS 报警, 返回值为 0xFB。

否则, 返回其他值请查看其他返回值定义, 返回的错误代码请查看错误代码定义。

## 3.2.14 BlockWrite\_G2 (): G2 块写命令

### 功能描述:

这个命令可以一次性往保留内存、EPC 存储器、TID 存储器或用户存储器中写入若干个字。

### 应用:

```
int BlockWrite_G2(unsigned char *ComAdr, unsigned char * EPC, unsigned char Wnum, unsigned char Enum,unsigned char Mem, unsigned char WordPtr,unsigned char *Writedata,unsigned char * Password, unsigned char MaskMem,unsigned char*MaskAdr,unsigned char MaskLen,unsigned char * MaskData,int * errorcode,int FrmHandle);
```

### 参数:

ComAdr: 输入变量, 读写器地址。

EPC: 指向输入数组变量 (输入的是每字节都转化为字符的数据)。是电子标签的 EPC 号。

Wnum: 输入变量, 待写入的字个数, 一个字为 2 个字节。这里字的个数必须和实际待写入的数据个数相等。

Enum: 在(0x00~0x0F)范围内表示 EPC 号长度, 以字为单位。EPC 的长度在 15 个字以内。此时不掩码。ENum 为 0xFF 时掩码。如果为其它值将返回参数错误信息。

**Mem:** 输入变量，一个字节。选择要读取的存储区。

**0x00:** 保留区；

**0x01:** EPC 存储器；

**0x02:** TID 存储器；

**0x03:** 用户存储器。

其他值保留。若命令中出现了其它值，将返回参数出错的消息。

**WordPtr:** 输入变量，一个字节。指定要写入的字起始地址。指定要写入数据的起始地址。如果写的是 EPC 区，则会忽略这个起始地址。EPC 区总是规定从 EPC 区 0x02 地址(EPC 号的第一个字节)开始写。

**Writedata:** 指向输入数组变量（输入的是每字节都转化为字符的数据）。待写入的字。这是要写入到存储区的数据。比如，WordPtr 等于 0x02，则输出变量 Data 中第一个字(从左边起)写在 Mem 指定的存储区的地址 0x02 中，第二个字写在 0x03 中，依次类推。

**Password:** 指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32 位的访问密码的最高位在 PassWord 的第一字节(从左往右)的最高位，访问密码最低位在 PassWord 第四字节的最低位，PassWord 的前两个字节放置访问密码的高字。

**MaskMem:** 输入变量，一个字节，掩码区。0x01: EPC 存储区；0x02: TID 存储区；0x03: 用户存储区。

**MaskAdr:** 输入数组，2 个字节，掩码的起始位地址（单位：Bits）。范围 0~16383。

**MaskLen:** 一个字节，掩码的位长度（单位：Bits）。

**MaskData:** 输入数组，掩码数据。MaskData 数据字节长度是 MaskLen/8。如果 MaskLen 不是 8 的整数倍，则 MaskData 数据字节长度为[MaskLen/8]取整再加 1。不够的在低位补 0

**Errorcode:** 输出变量，一个字节，读写器返回响应状态为 0xFC 时，返回错误代码。

**FrmHandle:** 输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

## 返回:

如果该函数调用成功，返回一个零值，完全写入。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

### 3.2.15 ExtReadData\_G2 (): G2 扩展读

#### 功能描述:

这个命令读取标签的整个或部分保留区、EPC 存储器、TID 存储器或用户存储器中的数据。从指定的地址开始读，以字为单位。

#### 应用:

```
int ExtReadData_G2 (unsigned char *ComAdr, unsigned char * EPC, unsigned char Enum,unsigned char Mem, unsigned char WordPtr, unsigned char Num,unsigned char * Password,unsigned char MaskMem,unsigned char *MaskAdr, unsigned char
```

MaskLen, unsigned char\*MaskData, unsigned char \* Data ,int \* errorcode,int FrmHandle);

## 参数:

**ComAdr:** 输入变量, 读写器地址。

**EPC:** 指向输入数组变量(输入的是每字节都转化为字符的数据)。是电子标签的 EPC 号。

**Enum:** 在(0x00~0x0F)范围内表示 EPC 号长度, 以字为单位。EPC 的长度在 15 个字以内。此时不掩码。ENum 为 0xFF 时掩码。如果为其它值将返回参数错误信息。

**Mem:** 输入变量, 一个字节。选择要读取的存储区。

0x00: 保留区;

0x01: EPC 存储器;

0x02: TID 存储器;

0x03: 用户存储器。

其他值保留。若命令中出现了其它值, 将返回参数出错的消息。

**WordPtr:** 输入数组, 2 个字节。指定要读取的字起始地址。0x0000 表示从第一个字(第一个 16 位存储体)开始读, 0x0001 表示从第 2 个字开始读, 依次类推。

**Num:** 输入变量, 一个字节。要读取的字的个数。不能设置为 0x00, 将返回参数错误信息。Num 不能超过 120, 即最多读取 120 个字。若 Num 设置为 0 或者超过了 120, 将返回参数出错的消息。

**Password:** 指向输入数组变量(输入的是每字节都转化为字符的数据), 四个字节, 这四个字节是访问密码。32 位的访问密码的最高位在 Password 的第一字节(从左往右)的最高位, 访问密码最低位在 Password 第四字节的最低位, Password 的前两个字节放置访问密码的高字。

**MaskMem:** 输入变量, 一个字节, 掩码区。0x01: EPC 存储区; 0x02: TID 存储区; 0x03: 用户存储区。

**MaskAdr:** 输入数组, 2 个字节, 掩码的起始位地址(单位: Bits)。范围 0~16383。

**MaskLen:** 一个字节, 掩码的位长度(单位: Bits)。

**MaskData:** 输入数组, 掩码数据。MaskData 数据字节长度是 MaskLen/8。如果 MaskLen 不是 8 的整数倍, 则 MaskData 数据字节长度为[MaskLen/8]取整再加 1。不够的在低位补 0

**Data:** 指向输出数组变量(输出的是每字节都转化为字符的数据), 是从标签中读取的数据。

**Errorcode:** 输出变量, 一个字节, 读写器返回响应状态为 0xFC 时, 返回错误代码。

**FrmHandle:** 输入变量, 返回与读写器连接端口对应的句柄, 应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功, 返回的句柄值为-1。

## 返回:

如果该函数调用成功, 返回一个零值, 读到的数据在 Data 中。

否则, 返回非零值请查看其他返回值定义, 返回的错误代码请查看错误代码定义。

## 3.2.16 ExtWriteData\_G2 (): G2 扩展写

### 功能描述:

这个命令可以一次性往保留内存、EPC 存储器、TID 存储器或用户存储器中写入若干个字。

### 应用:

```
int WriteData_G2(unsigned char *ComAdr, unsigned char * EPC, unsigned char Wnum, unsigned char Enum,unsigned char Mem, unsigned char WordPtr,unsigned char *Writedata,unsigned char * Password, unsigned char MaskMem,unsigned char*MaskAdr,unsigned char MaskLen,unsigned char * MaskData,int * errorcode,int FrmHandle);
```

### 参数:

**ComAdr:** 输入变量，读写器地址。

**EPC:** 指向输入数组变量（输入的是每字节都转化为字符的数据）。是电子标签的 EPC 号。

**Wnum:** 输入变量，待写入的字个数，一个字为 2 个字节。这里字的个数必须和实际待写入的数据个数相等。

**Enum:** 在(0x00~0x0F)范围内表示 EPC 号长度，以字为单位。EPC 的长度在 15 个字以内。此时不掩码。ENum 为 0xFF 时掩码。如果为其它值将返回参数错误信息。

**Mem:** 输入变量，一个字节。选择要写的存储区。

**0x00:** 保留区;

**0x01:** EPC 存储器;

**0x02:** TID 存储器;

**0x03:** 用户存储器。

其他值保留。若命令中出现了其它值，将返回参数出错的消息。

**WordPtr:** 输入数组，2 个字节。指定要写入的字起始地址。指定要写入数据的起始地址。如果写的是 EPC 区，则会忽略这个起始地址。EPC 区总是规定从 EPC 区 0x0002 地址(EPC 号的第 2 个字节)开始写。

**Writedata:** 指向输入数组变量（输入的是每字节都转化为字符的数据）。待写入的字。这是要写入到存储区的数据。比如，WordPtr 等于 0x02，则输出变量 Data 中第一个字(从左边起)写在 Mem 指定的存储区的地址 0x02 中，第二个字写在 0x03 中，依次类推。

**Password:** 指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32 位的访问密码的最高位在 PassWord 的第一字节(从左往右)的最高位，访问密码最低位在 PassWord 第四字节的最低位，PassWord 的前两个字节放置访问密码的高字。

**MaskMem:** 输入变量，一个字节，掩码区。0x01: EPC 存储区; 0x02: TID 存储区; 0x03: 用户存储区。

**MaskAdr:** 输入数组，2 个字节，掩码的起始位地址（单位：Bits）。范围 0~16383。

**MaskLen:** 一个字节，掩码的位长度（单位：Bits）。

**MaskData:** 输入数组,掩码数据。**MaskData** 数据字节长度是 **MaskLen/8**。如果 **MaskLen** 不是 8 的整数倍, 则 **MaskData** 数据字节长度为  $[\text{MaskLen}/8]$  取整再加 1。不够的在低位补 0

**Errorcode:** 输出变量, 一个字节, 读写器返回响应状态为 0xFC 时, 返回错误代码。

**FrmHandle:** 输入变量, 返回与读写器连接端口对应的句柄, 应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功, 返回的句柄值为-1。

**返回:**

如果该函数调用成功, 返回一个零值, 完全写入。

否则, 返回非零值请查看其他返回值定义, 返回的错误代码请查看错误代码定义。

### 3.2.17 InventoryBuffer\_G2 (): 带缓存查询命令

**功能描述:**

查询命令的作用是检查有效范围内是否有符合协议的电子标签存在。

**应用:**

```
int InventoryBuffer_G2(unsigned char *ComAdr, unsigned char QValue, unsigned char Session, unsigned char MaskMem, unsigned char *MaskAdr, unsigned char MaskLen, unsigned char *MaskData, unsigned char MaskFlag, unsigned char AdrTID, unsigned char LenTID, unsigned char TIDFlag, unsigned char Target, unsigned char InAnt, unsigned char Scantime, unsigned char Fastflag, int *BufferCount, int *TagNum, int FrmHandle);
```

**参数:**

**ComAdr:** 输入变量, 读写器地址。

**QValue:** 输入变量, Q 值, 范围 0-15;

**Session:** 输入变量, Session 值, 范围 0-3;

**MaskMem:** 输入变量, 一个字节, 掩码区。0x01: EPC 存储区; 0x02: TID 存储区; 0x03: 用户存储区。

**MaskAdr:** 输入数组, 2 个字节, 掩码的起始位地址 (单位: Bits)。范围 0~16383。

**MaskLen:** 一个字节, 掩码的位长度 (单位: Bits)。

**MaskData:** 输入数组, 掩码数据。**MaskData** 数据字节长度是 **MaskLen/8**。如果 **MaskLen** 不是 8 的整数倍, 则 **MaskData** 数据字节长度为  $[\text{MaskLen}/8]$  取整再加 1。不够的在低位补 0。

**MaskFlag:** 输入变量, 一个字节, 掩码标记位,

**MaskFlag=1** 掩码, **MaskFlag=0** 不掩码。

**AdrTID:** 输入变量, 查询 TID 区的起始字地址。

**LenTID:** 输入变量, 查询 TID 区的数据字数。LenTID 取值为 0~15。

**TIDFlag:** 输入变量,

**TIDFlag=1:**表示查询 TID 区;

TIDFlag=0:表示查询 EPC;

Target:1 个字节, 查询 EPC 标签时使用的 Target 值

0x00:Target 值使用 A;

0x01:Target 值使用 B;

InAnt:1 个字节, 本次要进行查询的天线号, 本模块固定 0x80;

Scantime:1 个字节, 本次命令查询的最大时间, 范围 3-255(\*100ms);

Fastflag:快速查询的标志位, 为 0 时不启用, 为 1 时启用, 此时要指定 Target,Inant,Scantime 3 个参数。

BufferCount: 输出变量, 缓存中记录的标签总数, 相同 EPC/TID 数据的标签将被视为同一张标签。若未清空缓存, 标签数量为多次查询操作的数量累加。

TagNum: 输出变量, 本次查询中读取标签的次数, 不区分是否多次读取同一张标签。

FrmHandle: 输入变量, 返回与读写器连接端口对应的句柄, 应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功, 返回的句柄值为-1。

## 返回:

如果该函数调用成功, 返回值:

0x01 查询时间结束前返回

0x02 查询时间结束使得查询退出

0x03 如果读到的标签数量无法在一条消息内传送完, 将分多次发送。如果 Status 为 0x0D, 则表示这条数据结束后, 还有数据。

0x04 还有电子标签未读取, 电子标签数量太多, MCU 存储不了

返回其他值, 请查看其他返回值定义, 返回的错误代码请查看错误代码定义。

### 3.2.18 GetMonza4QTWorkParamter\_G2 (): 读取 Monza4QT 工作参数

#### 功能描述:

这个命令用于读取标签当前设置的工作参数。仅对 Impinj 的 Monza 4QT 标签有效。

#### 应用:

```
int GetMonza4QTWorkParamter_G2(unsigned char *address, unsigned char *EPC, unsigned char ENum, unsigned char *Password, unsigned char MaskMem, unsigned char *MaskAdr, unsigned char MaskLen, unsigned char *MaskData, unsigned char *QTcontrol, int *Errorcode, int FrmHandle);
```

#### 参数:

ComAdr: 输入变量, 读写器地址。

EPC: 指向输入数组变量(输入的是每字节都转化为字符的数据)。是电子标签的 EPC 号。

Enum: 在(0x00~0x0F)范围内表示 EPC 号长度, 以字为单位。EPC 的长度在 15 个字以内。此时不掩码。ENum 为 0xFF 时掩码。如果为其它值将返回参数错误信息。



**Password:** 指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。32位的访问密码的最高位在 **PassWord** 的第一字节(从左往右)的最高位，访问密码最低位在 **PassWord** 第四字节的最低位，**PassWord** 的前两个字节放置访问密码的高字。

**MaskMem:** 输入变量，一个字节，掩码区。0x01: EPC 存储区；0x02: TID 存储区；0x03: 用户存储区。

**MaskAdr:** 输入数组，2 个字节，掩码的起始位地址（单位：Bits）。范围 0~16383。

**MaskLen:** 一个字节，掩码的位长度（单位：Bits）。

**MaskData:** 输入数组，掩码数据。**MaskData** 数据字节长度是 **MaskLen/8**。如果 **MaskLen** 不是 8 的整数倍，则 **MaskData** 数据字节长度为  $[\text{MaskLen}/8]$  取整再加 1。不够的在低位补 0。

**QTcontrol:** 标签工作参数。

**Bit0:** 标签当前使用的镜像页。**Bit0 = 0** 时表示 **private**；**Bit0 = 1** 时表示 **public**。

**Bit1:** 标签是否使能距离保护。**Bit1 = 0** 时表示不使能；**Bit1 = 1** 时表示使能。

其他值保留。

**Errorcode:** 输出变量，一个字节，读写器返回响应状态为 0xFC 时，返回错误代码。

**FrmHandle:** 输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为-1。

**返回:**

如果该函数调用成功，返回一个零值，读到的数据在 **Data** 中。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

### 3.2.19 SetMonza4QTWorkParamter\_G2 (): 设置 Monza4QT 工作参数

**功能描述:**

这个命令用于读取标签当前设置的工作参数。仅对 **Impinj** 的 **Monza 4QT** 标签有效。

**应用:**

```
int SetMonza4QTWorkParamter_G2(unsigned char *address, unsigned char *EPC, unsigned char ENum, unsigned char QTcontrol, unsigned char *Password, unsigned char MaskMem, unsigned char *MaskAdr, unsigned char MaskLen, unsigned char *MaskData, int *Errorcode, int FrmHandle);
```

**参数:**

**ComAdr:** 输入变量，读写器地址。

**EPC:** 指向输入数组变量（输入的是每字节都转化为字符的数据）。是电子标签的 EPC 号。

**Enum:** 在(0x00~0x0F)范围内表示 EPC 号长度，以字为单位。EPC 的长度在 15 个字以内。此时不掩码。ENum 为 0xFF 时掩码。如果为其它值将返回参数错误信息。

**QTcontrol:** 标签工作参数。

**Bit0:** 标签当前使用的镜像页。**Bit0 = 0** 时表示 **private**；**Bit0 = 1** 时表示 **public**。

**Bit1:** 标签是否使能距离保护。**Bit1 = 0** 时表示不使能；**Bit1 = 1** 时表示使能。

其他值保留。

**Password:** 指向输入数组变量（输入的是每字节都转化为字符的数据），四个字节，这四个字节是访问密码。**32** 位的访问密码的最高位在 **PassWord** 的第一字节(从左往右)的最高位，访问密码最低位在 **PassWord** 第四字节的最低位，**PassWord** 的前两个字节放置访问密码的高字。

**MaskMem:** 输入变量，一个字节，掩码区。**0x01:** EPC 存储区；**0x02:** TID 存储区；**0x03:** 用户存储区。

**MaskAdr:** 输入数组，2 个字节，掩码的起始位地址（单位：**Bits**）。范围 **0~16383**。

**MaskLen:** 一个字节，掩码的位长度（单位：**Bits**）。

**MaskData:** 输入数组，掩码数据。**MaskData** 数据字节长度是 **MaskLen/8**。如果 **MaskLen** 不是 **8** 的整数倍，则 **MaskData** 数据字节长度为 **[MaskLen/8]** 取整再加 **1**。不够的在低位补 **0**

**Errorcode:** 输出变量，一个字节，读写器返回响应状态为 **0xFC** 时，返回错误代码。

**FrmHandle:** 输入变量，返回与读写器连接端口对应的句柄，应用程序通过该句柄可以操作连接在相应端口的读写器。如果打开不成功，返回的句柄值为 **-1**。

**返回:**

如果该函数调用成功，返回一个零值，读到的数据在 **Data** 中。

否则，返回非零值请查看其他返回值定义，返回的错误代码请查看错误代码定义。

## 4 其他返回值定义

<b>#define InventoryReturnEarly_G2</b>	0x01	查询时间结束前返回
<b>#define InventoryTimeOut_G2</b>	0x02	指定的查询时间溢出
<b>#define InventoryMoreData_G2</b>	0x03	本条消息之后，还有消息
<b>#define ReadermoduleMCUFull_G2</b>	0x04	读写模块存储空间已满
<b>#define AccessPasswordError</b>	0x05	访问密码错误
<b>#define DestroyPasswordError</b>	0x09	销毁密码错误
<b>#define DestroyPasswordCannotZero</b>	0x0a	销毁密码不能为全 0
<b>#define TagNotSupportCMD</b>	0x0b	电子标签不支持该命令
<b>#define AccessPasswordCannotZero</b>	0x0c	对该命令，访问密码不能为 0
<b>#define TagProtectedCannotSetAgain</b>	0x0d	电子标签已经被设置了读保护，不能再次设置
<b>#define TagNoProtectedDonotNeedUnlock</b>	0x0e	电子标签没有被设置读保护，不需要解锁
<b>#define ByteLockedWriteFail</b>	0x10	有字节空间被锁定，写入失败
<b>#define CannotLock</b>	0x11	不能锁定
<b>#define LockedCannotLockAgain</b>	0x12	已经锁定，不能再次锁定



#define ParameterSaveFailCanUseBeforeNoPower	0x13	参数保存失败，但设置的值在读写模块断电前有效
#define CannotAdjust	0x14	无法调整
#define InventoryReturnEarly_6B	0x15	查询时间结束前返回
#define InventoryTimeOut_6B	0x16	指定的查询时间溢出
#define InventoryMoreData_6B	0x17	本条消息之后，还有消息
#define ReadermoduleMCUFull_6B	0x18	读写模块存储空间已满
#define NotSupportCMDOrAccessPasswordCannotZero	0x19	电子不支持该命令或者访问密码不能为 0
#define ANTCheckError	0xF8	天线检查错误
#define CMDExecuteErr	0xF9	命令执行出错
#define GetTagPoorCommunicationCannotOperation	0xFA	有电子标签，但通信不畅，无法操作
#define NoTagOperation	0xFB	无电子标签可操作
#define TagReturnErrorCode	0xFC	电子标签返回错误代码
#define CMDLengthWrong	0xFD	命令长度错误
#define IllegalCMD	0xFE	不合法的命令
#define ParameterError	0xFF	参数错误
#define CommunicationErr	0x30	通讯错误
#define RetCRCErr	0x31	CRC 校验错误
#define RetDataErr	0x32	返回数据长度有错误
#define CommunicationBusy	0x33	通讯繁忙，设备正在执行其他指令
#define ExecuteCmdBusy	0x34	繁忙，指令正在执行
#define ComPortOpened	0x35	端口已打开
#define ComPortClose	0x36	端口已关闭
#define InvalidHandle	0x37	无效句柄
#define InvalidPort	0x38	无效端口
#define RecmdErr	0xEE	返回指令错误

## 5 错误代码定义

#define OtherError	0x00	其它错误，全部捕捉未被其它代码覆盖的错误
#define MemoryOutPcNotSupport	0x03	存储器超限或不被支持的 PC 值，规定存储位置不存在或标签不支持 PC 值
#define MemoryLocked	0x04	存储器锁定，存储位置锁定或永久锁定，且不可写入
#define NoPower	0x0b	电源不足，标签电源不足，无法执行存储写入操作
#define NotSpecialError	0x0f	非特定错误，标签不支持特定错误代码