SDK开发指南

本SDK 包是公司为方便用户开发应用程序而开发的软件开发包。SDK 以动态链接库文件形式提供给用户使用。

用户使用本公司系列读写器进行用户应用软件开发时，根据本公司提供的SDK 开发包，用户可以高效、正确地完成DRF 系列读写器应用软件的开发。SDK 支持VC、VB、C#、VB.NET、C++ Builder 和Delphi 的开发。

使用函数注意事项：

1.对标签操作时（包括读标签，写标签），最好在主从模式下使用，因为读写器 UART 通讯是半双工的，定时模式下，此时读卡器定时读标签并上传数据，这时容易跟上位机发送的命令冲突而造成读取成功率下降，如果要在定时模式下使用的话，最好先发送停读标签命令，再来操作。此时如若要恢复到定时模式，要发送复位读头命令以再次进入到定时模式。

2.如果要读单标签，使用我们的 DEMO 软件，到参数设置那里把读写器首先要设置到单标签模式。

3.如果要读多标签，使用我们的 DEMO 软件，到参数设置那里把读写器首先要设置到多标签模式。

1、OpenComm

函数原型： HANDLE OpenComm(int portNo);

功能说明： 打开串口。

返回值： 成功时返回串口句柄，失败时返回 INVALID_HANDLE_VALUE（-1）

参数： ●——portNo： 串口号

调用例程：

```
HANDLE hCom = OpenComm(1);  // 打开串口 1
if(hCom != INVALID_HANDLE_VALUE)
    AfxMessageBox(“打开串口成功!”);
else
    AfxMessageBox(“打开串口失败!”);
```

2、CloseComm

函数原型： void CloseComm(HANDLE hCom);

功能说明： 关闭串口。

返回值： 无

参数： ●——hCom： 串口句柄

调用例程：

```
CloseComm(hCom);
```

3、ReadFirmwareVersion

函数原型： BOOL ReadFirmwareVersion(HANDLE hCom, int* main, int* sub,BYTE ReaderAddr);

功能说明： 读取 Firmware 版本（读头版本）。

返回值： 成功时返回 TRUE（1），失败时返回 FALSE（0）

参数： ●——hCom： 串口句柄

●——main： 主版本号的变量地址（输出参数）

●——sub： 子版本号的变量地址（输出参数）

●——ReaderAddr： 读头地址，一台主机接多台读头时使用，接单台读头时置为 0；

调用例程：

```
int main = 0;
int sub = 0;
if(ReadFirmwareVersion(hCom, &main, &sub,0))
```

```
{
    CString str;
    str.Format(“Firmware Version: %d.%d”, main, sub);
    AfxMessageBox(str);
}
else
    AfxMessageBox(“读取失败!”);
```

4、StopReading

函数原型： BOOL StopReading(HANDLE hCom ,BYTE ReaderAddr);

功能说明： 使读卡器停止读取标签。通常是在定时模式下，要临时停止读标签，使用该函数，如果要恢复到定时模式，此时要发送 ResumeReading，即复位读头命令。

返回值： 成功时返回 TRUE（1），失败时返回 FALSE（0）

参数：

●——hCom： 串口句柄

●——ReaderAddr： 读头地址，一台主机接多台读头时使用，接单台读头时置为 0；

调用例程：

```
if(StopReading(hCom,0))
    AfxMessageBox(“停读标签成功！”);
else
    AfxMessageBox(“停读标签失败!”);
```

5、ResumeReading

函数原型： BOOL ResumeReading(HANDLE hCom,BYTE ReaderAddr);

功能说明： 复位读写器，相当于对读写器重新上电，用于定时模式下，临时停读标签之后，恢复到定时读取标签。

返回值： 成功时返回 TRUE（1），失败时返回 FALSE（0）

参数：

●——hCom： 串口句柄

●——ReaderAddr： 读头地址，一台主机接多台读头时使用，接单台读头时置为 0；

调用例程：

```
if(ResumeReading(hCom,0))
    AfxMessageBox(“复位读头成功！”);
else
    AfxMessageBox(“复位读头失败!”);
```

6、IdentifySingleTag

函数原型： BOOL IdentifySingleTag(HANDLE hCom, BYTE* tagID, BYTE* antennaNo, BYTE ReaderAddr);

功能说明： 识别单个标签，通常在“主从模式”使用。

返回值： 成功时返回 TRUE（1），失败时返回 FALSE（0）

参数：

●——hCom： 串口句柄

●——tagID： 接收标签 ID 的数组地址（输出参数），长度为 12

●——antennaNo： 接收天线号的变量地址（输出参数）。不需要时置为 NULL。

●——ReaderAddr： 读头地址，一台主机接多台读头时使用，接单台读头时置为 0；

调用例程：

```
BYTE tagID[12];
BYTE antennaNo;
if(IdentifySingleTag(hCom, tagID, &antennaNo,0))
{
    // Transfer tag ID to Hex Format
```

```
    CString strID;
    CString strTmp;
    for(int i = 0; i < 12; i++)
    {
        strTmp.Format("%02X ", id[i]);
        strID += strTmp;
    }
    // Antenna No
    CString strAntennaNo;
    strAntennaNo.Format("%d", antennaNo);
    // Display
    AfxMessageBox(strID + "    Device No: " + strAntennaNo);
}
```

7、IdentifyUploadedSingleTag

函数原型： BOOL IdentifyUploadedSingleTag(HANDLE hCom, BYTE* tagID, BYTE* devNo=NULL, BYTE* antennaNo);

功能说明： 识别读卡器上传的单个标签，在“定时模式”及“单卡模式”下获取识别到的标签时使用该函数。

返回值： 成功时返回 TRUE（1），失败时返回 FALSE（0）

参数：

- ●——hCom：　串口句柄
- ●——tagID：　接收标签 ID 的数组地址（输出参数），长度为 12
- ●——devNo：　接收设备号的变量地址（输出参数）。不需要时置为 NULL。
- ●——antennaNo：　接收天线号的变量地址（输出参数）。不需要时置为 NULL。

调用例程：

调用例程：

```
BYTE tagID[12];
BYTE devNo;
BYTE antennaNo;
if(IdentifyUploadedSingleTag(hCom, tagID, &devNo, &antennaNo))
//只需 TagID 时: if(IdentifyUploadedSingleTag(hCom, tagID))
{
    // Transfer tag ID to Hex Format
    CString strID;
    CString strTmp;
    for(int i = 0; i < 12; i++)
    {
        strTmp.Format("%02X ", id[i]);
        strID += strTmp;
    }
    // Device No
    CString strDevNo;
    strDevNo.Format("%d", devNo);
    // Antenna No
    CString strAntennaNo;
    strAntennaNo.Format("%d", antennaNo);
```

```
        // Display
        AfxMessageBox(strID + "    Device No: " + strDevNo
                              + "    Antenna No: " + strAntennaNo);
    }
```

注意点： 通常结合 Timer 使用，具体可参照示例程序。

8、IdentifyUploadedMultiTags

函数原型： BOOL IdentifyUploadedMultiTags(HANDLE hCom, BYTE* tagNum, BYTE* tagIDs, BYTE* devNos=NULL, BYTE* antennaNos);

功能说明： 识别读卡器上传的多个标签。在“定时模式”和“多卡模式”下，获取读写器读到的多标签数据时使用该函数。

返回值： 成功时返回 TRUE（1），失败时返回 FALSE（0）

参数：

- ●——hCom： 串口句柄
- ●——tagNum： 接收标签数的变量地址（输出参数）。一次可能读取的最大标签数是 200。
- ●——tagIDs： 接收标签 ID 的数组地址（输出参数），长度为 12 * tagNum
- ●——devNos： 接收设备号的数组地址（输出参数）。长度为 1 * tagNum，不需要时置为 NULL。
- ●——antennaNos： 接收天线号的数组地址（输出参数）。长度为 1 * tagNum，不需要时置为 NULL。

假设读到 10 个标签时：

*tagNum 的值为 10；

tagIDs[0~11]、tagIDs[12~23]……、tagIDs[108~119]分别为第 1 个、第 2 个……第 10 个标签的 ID；

devNos[0]、devNos[1]……devNos[9]分别为第 1 个、第 2 个……第 10 标签的设备号；

antennaNos[0]、antennaNos[1]……antennaNos[9]为第 1 个、第 2 个……第 10 个标签的天线号；

调用例程：

```
    BYTE tagNum;
    BYTE tagIDs[12 * 200];
    BYTE devNos[200];
    BYTE antennaNos[200];
    if(IdentifyUploadedMultiTags(hCom, &tagNum, tagIDs, devNos, antennaNos))
//只需 TagID 时: if(IdentifyUploadedMultiTags(hCom, &tagNum, tagIDs,))
    {
        CString str;
        CString strTmp;
        for(int j = 0; j < tagNum; j++)
        {
            // Line No.
            strTmp.Format("%d ", j);
            str += ("No." + strTmp + ": ");
            // Transfer tag ID to Hex Format
            for(int i = 0; i < 12; i++)
            {
                strTmp.Format("%02X ", ids[12 * j + i]);
                str += strTmp;
            }
            // Device No
```

```
            CString strDevNo;
            strDevNo.Format("%d", devNos[j]);
            str += ("    Device No: " + strDevNo);
            // Antenna No
            CString strAntennaNo;
            strAntennaNo.Format("%d", antennaNos[j]);
            str += ("    Antenna No: " + strAntennaNo);
            // Change Line
            str += "\r\n";
        }
        // Display
        AfxMessageBox(str);
    }
```

注意点： 通常结合 Timer 使用，具体可参照示例程序。

9、ReadTag

函数原型： BOOL ReadTag(HANDLE hCom, BYTE memBank, BYTE address, BYTE length, BYTE* data, BYTE ReaderAddr);

功能说明： 读标签内容，最好在“主从模式”使用。可以读取保留区，EPC 区，TID 区，及用户区数据，此函数跟 IdentifySingleTag 函数的区别在于，

1. IdentifySingleTag 函数只能读取 EPC 区的标签 ID 号。而本函数可以读 4 个区，范围更广。
2. IdentifySingleTag 标签识别的距离更远，效果更好，而本函数读取标签的距离要近些，效果不如 IdentifySingleTag 好。

返回值： 成功时返回 TRUE（1），失败时返回 FALSE（0）

参数：

- ●——hCom： 串口句柄
- ●——memBank： 要读的区域。各值的意义如下：
  - 0x00——保留区
  - 0x01——EPC 区
  - 0x02——TID 区
  - 0x03——用户区
- ●——address： 要读区域中的地址，取值为范围 0-7。
- ●——length： 要读取的长度，取值范围是 1 到 8（单位是 Word，1Word = 2Byte）。
- ●——data： 要写入内容的地址（输出参数）
- ●——ReaderAddr: 读头地址，一台主机接多台读头时使用，接单台读头时置为 0；

调用例程：

```
BYTE data[2];
if(::ReadTag(hCom, 0x01, 0x03, 1, data,0))     // 读 EPC 区地址 02 开始的 1 个字(2 字节)内容//
{
    CString strData;
    strData.Format("%02X %02X ", data[0], data[1]);
    AfxMessageBox("读 EPC 区地址 02 开始的 1 个字(2 字节)内容成功！ "+strData);
}
```

```
else
    AfxMessageBox("读标签内容失败！");
```

10、WriteTagSingleWord

函数原型： BOOL WriteTagSingleWord(HANDLE hCom, BYTE memBank, BYTE address, BYTE data1, BYTE data2, BYTE ReaderAddr);

功能说明： 向标签写入 1 个字（2 字节）的内容。（注：EPC 区的地址 0、1 不可写入）最好在主从模式下使用。

返回值： 成功时返回 TRUE（1），失败时返回 FALSE（0）

参数：

- ●——hCom：串口句柄
- ●——memBank：要写的区域。各值的意义如下：
  - 0x00——保留区
  - 0x01——EPC 区
  - 0x02——TID 区
  - 0x03——用户区
- ●——address：要写区域中的地址，取值为范围 0-7（memBank 为 EPC 区时， 0、 1 不可取）。
- ●——data1：要写入内容的第 1 个字节
- ●——data2：要写入内容的第 2 个字节
- ●——ReaderAddr：读头地址，一台主机接多台读头时使用，接单台读头时置为 0；

调用例程：

```
// 往 EPC 区地址 02 处写 1 个字(2 字节)内容，2 个字节值均为 0xFF
if(::WriteTagSingleWord(hCom, 0x01, 0x02, 0xFF, 0xFF,0))
    AfxMessageBox("往 EPC 区地址 02 处写 1 个字(2 字节)内容成功！");
else
    AfxMessageBox("写标签失败！");
```

11、FastWriteTagID

函数原型： BOOL FastWriteTagID(HANDLE hCom, int bytesNum, const BYTE* bytes, BYTE ReaderAddr);

功能说明： 快写标签 ID。只写标签的 EPC 区，即通常所说的标签号码，最好在主从模式下使用。

返回值： 成功时返回 TRUE（1），失败时返回 FALSE（0）

参数：

- ●——hCom：串口句柄
- ●——bytesNum：要写入内容的字节数，必须为 2，4，6，8，10，或 12。
- ●——bytes：要写入内容的地址
- ●——ReaderAddr：读头地址，一台主机接多台读头时使用，接单台读头时置为 0；

调用例程：

```
BYTE id[4] = {0x00, 0x11, 0x22, 0x33};
if(::FastWriteTagID(hCom, 4, id,0)) // 写入 4 字节
    AfxMessageBox("写卡成功！");
else
    AfxMessageBox("写卡失败！");
```

注意点： 一次写入的字节数越多，成功率越低，所以请只写必要数量的字节。

12. FastWriteTag

函数原型： BOOL FastWriteTag(HANDLE hCom, BYTE memBank, BYTE address, BYTE WordCount,const BYTE* bytes,BYTE ReaderAddr)

功能说明： 快写标签,包括 EPC 区，数据区和保留区，最好在主从模式下使用。

参数：

●——hCom: 串口句柄

●——memBank: 要写的区域。各值的意义如下：

0x00——保留区

0x01——EPC 区

0x02——TID 区

0x03——用户区

●——address: 要写区域中的地址，

memBank 为 EPC 区时，地址范围为 2-7，最大 WordCount 为 6；

为保留区时，地址范围为 0-3，最大 WordCount 为 4；

为数据区时，址址范围为 0-31，最大 WordCount 为 8；

●——WordCount: 要写入内容的长度，以字为单位，1 个字为 2 个字节

●——bytes: 要写入内容的地址

●——ReaderAddr: 读头地址，一台主机接多台读头时使用，接单台读头时置为 0；

调用例程：

```
BYTE id[8] = {0x00, 0x11, 0x22, 0x33，0x44,0x55,0x66,0x77};
if(::FastWriteTag(hCom, 3, 0，4，id,0)) // 写入 4 个字，即 8 个字节到用户区
    AfxMessageBox("写卡成功！");
else
    AfxMessageBox("写卡失败！");
```

注意点： 一次写入的字节数越多，成功率越低，所以请只写必要数量的字节。

13、InitializeTag

函数原型： BOOL InitializeTag(HANDLE hCom, BYTE ReaderAddr);

功能说明： 初始化标签，初始化标签的协议控制字，通常不需使用该函数。

返回值： 成功时返回 TRUE（1），失败时返回 FALSE（0）

参数：

●——hCom: 串口句柄

●——ReaderAddr: 读头地址，一台主机接多台读头时使用，接单台读头时置为 0；

调用例程：

```
if(InitializeTag(hCom,0))
    AfxMessageBox("初始化标签成功！");
else
    AfxMessageBox("初始化标签失败！");
```

14、LockPassWordTag

函数原型 LockPassWordTag (HANDLE hCom,BYTE passwd1,BYTE passwd2,BYTE passwd3,BYTE passwd4,BYTE lockType, BYTE ReaderAddr);

功能说明：通过访问密码锁定标签，通常在主从模式下使用该函数。

参数：

●——hCom: 串口句柄

●——passwd1: 访问密码 1

●——passwd2: 访问密码 2

●——passwd3: 访问密码 3

●——passwd4: 访问密码 4

●——lockType: 锁定类型。各值的意义如下：

值	锁定类型
00	LOCK USER

01	LOCK TID
02	LOCK EPC
03	LOCK ACCESS
04	LOCK KILL
05	LOCK ALL
其他值	DO NOT LOCK

●——ReaderAddr:　读头地址，一台主机接多台读头时使用，接单台读头时置为 0；

调用例程：

```
if(LockPassWordTag (hCom, 0X12,0X34,0X56,0X78, 02,0))   // LOCK EPC
    AfxMessageBox("锁定标签成功！");
else
    AfxMessageBox("锁定标签失败！");
```

15、 UnlockPassWordTag

函数原型 UnlockPassWordTag(HANDLE hCom,BYTE passwd1,BYTE passwd2,BYTE passwd3,BYTE passwd4,BYTE lockType, BYTE ReaderAddr);

功能说明：通过访问密码解锁标签，通常在主从模式下使用该函数。

参数：

●——hCom:　串口句柄

●——passwd1:　访问密码 1

●——passwd2:　访问密码 2

●——passwd3:　访问密码 3

●——passwd4:　访问密码 4

●——lockType:　锁定类型。各值的意义如下：

值	解锁类型
00	UNLOCK USER
01	UNLOCK TID
02	UNLOCK EPC
03	UNLOCK ACCESS
04	UNLOCK KILL
05	UNLOCK ALL
其他值	DO NOT UNLOCK

●——ReaderAddr:　读头地址，一台主机接多台读头时使用，接单台读头时置为 0；

调用例程：

```
if(UnlockPassWordTag (hCom, 0X12,0X34,0X56,0X78, 02,0)) //UNLOCK EPC
    AfxMessageBox("解锁标签成功！");
else
    AfxMessageBox("解锁标签失败！");
```

16、KillTag

函数原型： BOOL KillTag(HANDLE hCom, BYTE passwd1, BYTE passwd2, BYTE passwd3, BYTE passwd4, BYTE ReaderAddr);

功能说明： 销毁标签。通常在主从模式下使用该函数。

返回值： 成功时返回 TRUE（1），失败时返回 FALSE（0）

参数： ●——hCom:　串口句柄

●——passwd1：　　密码 1
●——passwd2：　　密码 2
●——passwd3：　　密码 3
●——passwd4：　　密码 4
●——ReaderAddr:　读头地址，一台主机接多台读头时使用，接单台读头时置为 0；

调用例程：

```
if(KillTag(hCom,0))
    AfxMessageBox("销毁标签成功！");
else
    AfxMessageBox("销毁标签失败！");
```

17、GetReaderParameters

函数原型： BOOL GetReaderParameters(HANDLE hCom, int addr, int paramNum, BYTE* params, BYTE ReaderAddr);

功能说明： 获取多个读写器参数。

返回值： 成功时返回 TRUE（1），失败时返回 FALSE（0）

参数：
●——hCom:　　串口句柄
●——addr:　　要查询的读写器参数的起始地址（各参数的对应地址参考附表 1-11）
●——paramNum: 要查询的读写器参数的数量
●——params:　接收读写器参数的数组地址（输出参数）
●——ReaderAddr:　读头地址，一台主机接多台读头时使用，接单台读头时置为 0；

调用例程：

```
BYTE params[2];
// 获取用户标志码和发射功率（对应的地址分别为 0x64、0x65）
if(GetReaderParameters(hCom, 0x64, 2, params,0))
{
    CString str;
    str.Format(“用户标志码: %d. 功率: %d”, params[0], params[1]);
    AfxMessageBox(str);
}
else
    AfxMessageBox(“获取读写器参数失败!”);
```

18、SetReaderParameters

函数原型： BOOL SetReaderParameters(HANDLE hCom, int addr, int paramNum,
const BYTE* params, BYTE ReaderAddr);

功能说明： 设置多个读写器参数，通常在主从模式下使用该函数。

返回值： 成功时返回 TRUE（1），失败时返回 FALSE（0）

参数：
●——hCom:　　串口句柄
●——addr:　　要查询的读写器参数的起始地址（各参数的对应地址参考附表 1-11）
●——paramNum: 要查询的读写器参数的数量
●——params:　读写器参数的数组地址
●——ReaderAddr:　读头地址，一台主机接多台读头时使用，接单台读头时置为 0；

调用例程：

```
BYTE params[2] = { 0, 140 };
// 用户标志码 = 0，发射功率 = 140（对应的地址分别为 0x64、0x65）
if(SetReaderParameters(hCom, 0x64, 2, params,0))
    AfxMessageBox(“获取读写器参数成功!”);
```

```
else
    AfxMessageBox(“获取读写器参数失败!”);
```

19、ReadTIDByEpcID

函数原型： BOOL ReadTIDByEpcID(HANDLE hCom, const BYTE* bytes,BYTE* data,BYTE ReaderAddr);

功能说明： 指定标签的 EPC 区号码（12 个字节）读取对应标签的 TID 区（8 个字节）
通常在主从模式下使用该函数。

返回值： 成功时返回 TRUE（1），失败时返回 FALSE（0）

参数：

- ●——hCom： 串口句柄
- ●——bytes： EPC 号码指针，包含 12 个字节的 EPC 数据
- ●——data 返回的 TID 数据指针，包含 8 个字节的 TID 数据
- ●——ReaderAddr: 读头地址，一台主机接多台读头时使用，接单台读头时置为 0；

调用例程：

```
if ReadTIDByEpcID(hCom, bytes, data,0);
    AfxMessageBox(“指定 EPC 区，读取 TID 成功!”);
else
    AfxMessageBox(“指定 EPC 区，读取 TID 失败!”);
```

20、ReadByEpcID

函数原型： BOOL ReadByEpcID(HANDLE hCom, BYTE memBank, BYTE address, BYTE WordCount,const BYTE* EpcID, BYTE* data,BYTE ReaderAddr)

功能说明： 指定标签的 EPC 区号码（12 个字节）读取对应标签，通常在主从模式下使用该函数。

返回值： 成功时返回 TRUE（1），失败时返回 FALSE（0）

参数：

- ●——hCom： 串口句柄
- ●——memBank： 要读标签的区域。各值的意义如下：
  0x00——保留区
  0x01——EPC 区
  0x02——TID 区
  0x03——用户区
- ●——address： 要读区域中的地址，
  memBank 为 EPC 区时，地址范围为 2-7，最大 WordCount 为 6；
  为保留区时，地址范围为 0-3，最大 WordCount 为 4；
  为数据区时，址址范围为 0-31，最大 WordCount 为 8；
- ●——WordCount： 要读入内容的长度，以字为单位，1 个字为 2 个字节
- ●——EpcID： EPC 号码指针，包含 12 个字节的 EPC 数据
- ●——data： 读出的数据存入的指针
- ●——ReaderAddr: 读头地址，一台主机接多台读头时使用，接单台读头时置为 0；

调用例程：

```
BYTE data[8];
BYTE EPCID[12]={11,22,33,44,55,66,77,88,99,00,11,22};
  if ReadByEpcID (hCom, 03,00,4，EPCID, data,0);//指定 EPC 读用户区
    AfxMessageBox(“指定 EPC 区，读用户区成功!”);
  else
    AfxMessageBox(“指定 EPC 区，读用户区失败!”);
```

21、WriteByEpcID

函数原型： BOOL WriteByEpcID(HANDLE hCom, BYTE memBank, BYTE address, BYTE WordCount,const BYTE* EpcID,BYTE* data,BYTE ReaderAddr)

功能说明： 指定标签的 EPC 区号码（12 个字节）写对应标签

返回值： 成功时返回 TRUE（1），失败时返回 FALSE（0）

参数： ●——hCom：串口句柄

●——memBank：要写入标签的区域。各值的意义如下：

0x00——保留区

0x01——EPC 区

0x02——TID 区

0x03——用户区

●——address：要写入区域中的地址，

memBank 为 EPC 区时，地址范围为 2-7，最大 WordCount 为 6；

为保留区时，地址范围为 0-3，最大 WordCount 为 4；

为数据区时，址址范围为 0-31，最大 WordCount 为 8；

●——WordCount：要写入内容的长度，以字为单位，1 个字为 2 个字节

●——EpcID：EPC 号码指针，包含 12 个字节的 EPC 数据

●——data：要写入的数据的指针

●——ReaderAddr：读头地址，一台主机接多台读头时使用，接单台读头时置为 0；

调用例程：

```
BYTE data[8];
BYTE EPCID[12]={11,22,33,44,55,66,77,88,99,00,11,22};
  if WriteByEpcID (hCom, 03,00,4, EPCID, data,0);//指定 EPC 写用户区
      AfxMessageBox(“指定 EPC 区，写用户区成功!”);
  else
      AfxMessageBox(“指定 EPC 区，写用户区失败!”);
```

22、BeepCtrl

函数原型： BOOL BeepCtrl(HANDLE hCom,BYTE BeepStatus,BYTE ReaderAddr)

功能说明： 控制读写器 BUZZER 。

返回值： 成功时返回 TRUE（1），失败时返回 FALSE（0）

参数： ●——hCom：串口句柄

●——BeepStatus：控制 BUZZER 动作参数，

BeepStatus=0：读到卡时不响 BEEP 声；

BeepStatus=1：读到卡时响 BEEP 声；

BeepStatus>=2：单独响一次 BEEP 声；

●——ReaderAddr：读头地址，一台主机接多台读头时使用，接单台读头时置为 0；

```
If (BeepCtrl(hCom,0,0))
  AfxMessageBox(“关闭 BEEP 声成功!”);
```

23、RelayCtrl

函数原型： BOOL RelayCtrl(HANDLE hCom,BYTE RelayOnOff,BYTE ReaderAddr)

功能说明： 控制继电器。

返回值： 成功时返回 TRUE（1），失败时返回 FALSE（0）

参数： ●——hCom：串口句柄

●——RelayOnOff：控制继电器参数，

RelayOnOff =0：关闭继电器；

RelayOnOff =1：打开继电器；

●——ReaderAddr：读头地址，一台主机接多台读头时使用，接单台读头时置为 0；

```
If (RelayCtrl(hCom,0,0))
  AfxMessageBox(“关闭继电器成功!”);
```

附表 1 ：

参数在EEPROM中地址（十六进制）	项目含义	设置操作的有效值（十进制）	数值含义解释	其他
0x64	设备号	0 - 255	默认为 0	发送或接受命令时有些命令带有设备号,此时命令的设备号要与读头的设备号一致.
0x65	发射功率	0 - 150	功率模拟量	
0x70	读写器读卡操作发生模式	1，2，3	1：主从工作模式 2：定时工作模式 3：触发工作模式	**注意：工作在模式 2，3 时，主从模式仍然有效**

附表 2 ：

参数在EEPROM中地址（十六进制）	项目含义	设置操作的有效值（十进制）	数值含义解释	其他
0x71	读卡时间间隔	N	数值单位为：(N*10)毫秒 其中 N 为 10 – 100;	读写器读卡操作为模式 2，3 时有效
0x72	读卡器读取到数据后主动发送数据的链路选择	1，2，3	1：RS485 链路 2：wiegand 链路 3：RS232 链路	

附表 3:

参数在EEPROM中地址（十六进制）	项目含义	设置操作的有效值（十进制）	数值含义解释	其他
0x73	Wiegand 协议选择	1，2，3	1：wiegand26 2：wiegand34 3：wiegand32	对 wiegand 方式有效
0x74	Wiegand 输出数据脉冲宽度	1 - 255	读卡器内部转换为时间，时间=这个值*10(微秒)。	
0x75	Wiegand 输出数据脉冲周期	1 - 255	读卡器内部转换为时间，时间=这个值*100(微秒)。	
0x76	Wiegand 输出重复次数	1，2，3	暂不支持	
0x77	Wiegand 重复输出的间隔时间	1，2，3，4，5，6，7，8，9，10	读卡器内部转换为时间，时间=这个值*10(毫秒)。（暂不支持）	

附表 4:

参数在 EEPROM 中地址（十六进制）	项目含义	设置操作的有效值（十进制）	数值含义解释	其他
0x80	触发管脚启用触发工作模式设置	数值低四比特设置为 0 或者 1，表示工作不触发或者触发	Bit0：对应触发管脚 0 Bit1：对应触发管脚 1（暂不支持） Bit2：对应触发管脚 2（暂不支持） Bit3：对应触发管脚 3（暂不支持）	读写器读卡操作为模式 3 时有效
0x81	触发管脚触发模式	数值低四比特设置为 0 或者 1，表示低电平触发或者高电平触发	Bit0：对应触发管脚 0（支持高电平触发） Bit1：对应触发管脚 1（暂不支持） Bit2：对应触发管脚 2（暂不支持） Bit3：对应触发管脚 3（暂不支持）	
0x84	延迟关闭时间	0 - 240	读卡器内部转换为时间，时间=这个值*100(毫秒)。	

附表 5：

参数在 EEPROM 中地址（十六进制）	项目含义	设置操作的有效值（十进制）	数值含义解释	其他
0x90	跳频设置	0 - 50	0：跳频工作方式。 1--50：固定频率工作方式，频率值由此数值决定	
0x92~0x98	跳频-频率参数	比特设置为 0 或者 1，表示不选定该频率或者选定该频率	从 0x92 的 BIT0（第 1 个频率） - BIT7(第 7 个频率)，依此类推，可设置 50 个频率循环工作	

附表 6：

参数在 EEPROM 中地址（十六进制）	项目含义	设置操作的有效值（十进制）	数值含义解释	其他
0x87	单标签和多标签识别	0，1	0：EPC 单标签识别 1：EPC 多标签识别	

附表 7：

参数在 EEPROM 中地址（十六进制）	项目含义	设置操作的有效值（十进制）	数值含义解释	其他
0x89	天线工作方式	1 ，4	1：单一天线工作	

			4：多天线循环工作	
0x8A	选择工作天线	数值低四比特设置为 0 或者 1，表示不选择或者选择相应天线工作。	0：不选任何天线工作 1：天线 1 工作 2：天线 2 工作 4：天线 3 工作 8：天线 4 工作 15：全部天线都工作	

附表 8：

参数在 EEPROM 中地址（十六进制）	**项目含义**	**设置操作的有效值（十进制）**	**数值含义解释**	**其他**
0x7B	ID 相邻判别	1 ，2	1：ID 相邻判别启动 2:不启动(实时发数据有效)	
0x7A	ID 相邻判别时间	1 - 255	读卡器内部转换为时间，时间=这个值*1（秒)	注:ID 相邻判别启动时，时间值不能为 0,否则自动转为不启动。