

# VIKITEK 6100系列读写器SDK说明

动态库文件名：MR6100Api.dll

## 1.函数

### 1-1. 读写器管理函数

#### TcpConnectReader

函数描述	public int TcpConnectReader(string ip, int port)	
功能	基于网口模式下连接读写器设备	
参数	参数	参数意义、范围
	ip	读写器网口地址,是一个 IP 地址的地址的字符串格式的有效表示形式,必须和被连接的电脑处于同一个网段(默认:"192.168.1.200")
	port	设备端口号(范围 1~65535)
返回值	返回值：2001 为成功。	
应用实例	If(TcpConnectReader ("192.168.1.200",100)==2001)  MessageBox("成功"); Else MessageBox("失败");	

#### OpenCommPort

函数描述	public int OpenCommPort(string strPort, int nBoud)	
功能	初始化设备端口连接和配置 READER 波特率等参数	
参数	参数	参数意义、范围
	strPort	通讯串口号(COM1、COM2、COM3.....)
	nBoud	串口波特率(9600、19200、38400、57600、115200)
返回值	返回值：2001 为成功。	
应用实例	If(OpenCommPort ( "COM1" ,9600)==2001)  MessageBox("成功"); Else	

```
MessageBox("失败");
```

## TcpCloseConnect

函数描述	public int TcpCloseConnect()
功能	关闭设备端口，断开与设备的连接
参数	无
返回值	返回值：2001 为成功。
应用实例	<pre>If(TcpCloseConnect ()==2001)      MessageBox("成功"); Else     MessageBox("失败");</pre>

## CloseCommPort

函数描述	public void CloseCommPort()
功能	关闭设备端口，断开与设备的连接
参数	无
返回值	无
应用实例	<pre>CloseCommPort ();</pre>

## SetBaudRate

函数描述	public int SetBaudRate(int nReaderAddr, int nBaudRate)						
功能	设置读写器的波特率						
参数	<table><tr><th>参数</th><th>参数意义、范围</th></tr><tr><td>nBaudRate</td><td>串口通讯特定的波特率,分别为 9600、19200、38400、57600 和 115200 ( 或者 0,1,2,3,4 )</td></tr><tr><td>nReaderAddr</td><td>nReaderAddr：读写器地址，供固定式读写器组 RS485 网络使用，默认为 255 ( 手持机与模块为无效参数 )。</td></tr></table>	参数	参数意义、范围	nBaudRate	串口通讯特定的波特率,分别为 9600、19200、38400、57600 和 115200 ( 或者 0,1,2,3,4 )	nReaderAddr	nReaderAddr：读写器地址，供固定式读写器组 RS485 网络使用，默认为 255 ( 手持机与模块为无效参数 )。
参数	参数意义、范围						
nBaudRate	串口通讯特定的波特率,分别为 9600、19200、38400、57600 和 115200 ( 或者 0,1,2,3,4 )						
nReaderAddr	nReaderAddr：读写器地址，供固定式读写器组 RS485 网络使用，默认为 255 ( 手持机与模块为无效参数 )。						
返回值	8FD4 回值：2001 为成功。						

应用实例

```
If(SetBaudRate(255, 115200)==2001)
    MessageBox("成功");
Else
    MessageBox("失败");
```

## ResetReader

函数描述	public int ResetReader(int readerAddr)	
功能	读写器复位	
参数	readerAddr	读写器地址，供固定式读写器组 RS485 网络使用，默认为 255（手持机与模块为无效参数）。
返回值	返回值：2001 为成功。	
应用实例	<pre>If(ResetReader(255)==2001)     MessageBox("成功"); Else     MessageBox("失败");</pre>	

## ResetParameter

函数描述	public int ResetParameter (int readerAddr)	
功能	更新读写器的所有参数	
参数	readerAddr	读写器地址，供固定式读写器组 RS485 网络使用，默认为 255（手持机与模块为无效参数）。
返回值	返回值：2001 为成功。	
应用实例	<pre>If(ResetParameter (255)==2001)     MessageBox("成功"); Else     MessageBox("失败");</pre>	

## GetFirmwareVersion

函数描述	public int GetFirmwareVersion(int readerAddr,ref byte v1, ref byte v2)	
功能	查询读写器固件版本号	

参数	参数	参数意义、范围
	readerAddr	读写器地址
	v1	固件版本的高字节
	v2	固件版本的低字节
返回值	返回值：2001 为成功。	
应用实例	<pre>If(GetFirmwareVersion(255,ref v1,ref v2)==2001)     MessageBox("成功"); Else     MessageBox("失败");</pre>	

## SetRf

函数描述	<pre>public int SetRf(int readerAddr, int power1, int power2, int power3, int power4)</pre>	
功能	设置读写器的功率和频率参数	
参数	参数	参数意义、范围
	readerAddr	读写器地址
	Power1-	功率值，取值为 0~30，对应 0~30dBm.分别对应天线一到天线
	Power4	四
返回值	返回值：2001 为成功。	
应用实例	<pre>If(SetRf(255,0, 27,30,30) == 2001)     MessageBox("成功"); Else     MessageBox("失败");</pre>	

## GetRf

函数描述	<pre>public int GetRf(int readerAddr,ref int[] power)</pre>	
功能	查询读写器当前功率和频率参数	
参数	参数	参数意义、范围
	readerAddr	读写器地址
	power	查询到的天线对应功率值，取值为 0~30，对应 0~30dBm.
返回值	返回值：2001 为成功。	

应用实例

```
If(GetRf(255,ref power) == 2001)

    MessageBox("成功");
Else
    MessageBox("失败");
```

## SetAnt

函数描述	public int SetAnt(int readerAddr, byte Antenna)	
功能	设置读写器天线号的开放状态	
参数	参数	参数意义、范围
	readerAddr	读写器地址
	Antenna	表示工作的天线，用掩码的方式表示。低 4 位分别表示四个天线是否开通，1 表示开通，0 表示不开通；高 4 位没有意义。
返回值	返回值：2001 为成功。	
应用实例	<pre>If(SetAnt (ReaderAddr, ant) == 2001)      MessageBox("成功"); Else     MessageBox("失败");</pre>	

## GetAnt

函数描述	public int GetAnt(int ReaderAddr, ref byte workAnt, ref byte antState)	
功能	查询读写器当前处于开放状态的天线号和天线的连接情况	
参数	参数	参数意义、范围
	ReaderAddr	读写器地址，供固定式读写器组 RS485 网络使用，默认为 0xFF（手持机与模块为无效参数）。
	workAnt	表示当前开通的天线状态，用掩码表示。
	antState	表示当前实际可用的天线，1 表示可用，0 表示天线未接或不匹配。
返回值	返回值：2001 为成功。	
应用实例	<pre>If(GetAnt (ReaderAddr,ref workAnt , ref antState))== 2001)      MessageBox("成功");</pre>	

```
Else  
    MessageBox("失败");
```

## SetFrequency

函数描述	public int SetFrequency (int ReaderAddr, int freqNum, int[] points)	
功能	设置读写器射频参数	
参数	参数	参数意义、范围
	ReaderAddr	读写器地址,供固定式读写器组 RS485 网络使用,默认为 0XFF (手持机与模块为无效参数)。
	freqNum	频率点数,如果为非零,则频率为 Freq points 中各项频点;  如果 Freq num 为 0,则 Freq points 中一字节表示频率地区类型,分别为:  0: 中国  1: 北美  2: 欧洲
	points	自定义的频率范围是 900~930MHz 之间,以 250kHz 为步进的频率点索引。
返回值	返回值: 2001 为成功。	
应用实例	<pre>If(SetFrequency (ReaderAddr,freqNum , points))== 2001)      MessageBox("成功"); Else     MessageBox("失败");</pre>	

## GetFrequency

函数描述	public int GetFrequency (int ReaderAddr,ref int freqNum,ref int[] points)	
功能	查询读写器射频参数	
参数	参数	参数意义、范围
	ReaderAddr	读写器地址,供固定式读写器组 RS485 网络使用,默认为 0XFF (手持机与模块为无效参数)。
	freqNum	频率点数,如果为非零,则频率为 Freq points 中各项频点;  如果 Freq num 为 0,则 Freq points 中一字节表示频率地区类型,分别为:  0: 中国  1: 北美  2: 欧洲
	points	自定义的频率范围是 900~930MHz 之间,以 250kHz 为步进的频率点索引。

	区类型，分别为：
	0：中国
	1：北美
	2：欧洲
	points 自定义的频率范围是 900~930MHz 之间，以 250kHz 为步进的频率点索引。
返回值	返回值：2001 为成功。
应用实例	<pre>If(GetFrequency (ReaderAddr,ref freqNum , ref points))== 2001)     MessageBox("成功"); Else     MessageBox("失败");</pre>

## GetFastTagMode

函数描述	public int GetFastTagMode (int ReaderAddr, ref int mode)	
功能	获取读写器载波抵消策略	
参数	参数	参数意义、范围
	ReaderAddr	读写器地址，供固定式读写器组 RS485 网络使用，默认为 0XFF（手持机与模块为无效参数）。
	mode	Mode: 0 为单卡（含少量多卡）快速模式，非 0 为大量卡模式。
返回值	返回值：2001 为成功。	
应用实例	<pre>If(GetFastTagMode (addr,ref mode))== 2001)     MessageBox("成功"); Else     MessageBox("失败");</pre>	

## SetFastTagMode

函数描述	public int SetFastTagMode (int ReaderAddr, int mode)	
功能	设置读写器载波抵消策略	
参数	参数	参数意义、范围
	ReaderAddr	读写器地址，供固定式读写器组 RS485 网络使用，默认为 0XFF（手持机与模块为无效参数）。
	mode	Mode: 0 为单卡（含少量多卡）快速模式，非 0 为大量卡模式。

返回值            返回值：2001 为成功。

应用实例            If(SetFastTagMode (addr, mode)== 2001)

                          MessageBox("成功");

                          Else

                          MessageBox("失败");

## SetTestMode

函数描述	public int SetTestMode (int ReaderAddr, int mode)	
功能	设置读写器载波抵消策略	
参数	参数	参数意义、范围
	ReaderAddr	读写器地址 ,供固定式读写器组 RS485 网络使用 ,默认为 0XFF (手持机与模块为无效参数 )。
	mode	00 为打开功放 ;  01 为关闭功放 ;  02 为天线校准 ,天线校准在四个天线全部断开时使用
返回值	返回值：2001 为成功。	
应用实例	If(SetTestMode (addr, mode)== 2001)	
	MessageBox("成功");	
	Else	
	MessageBox("失败");	

## QueryIDCount

函数描述	public int QueryIDCount (int ReaderAddr, ref byte tagCount)	
功能	查询缓存区中的数据组数	
参数	参数	参数意义、范围
	ReaderAddr	读写器地址 ,供固定式读写器组 RS485 网络使用 ,默认为 0XFF (手持机与模块为无效参数 )。
	tagCount	查询到的数据组数
返回值	返回值：2001 为成功。	
应用实例	If(QueryIDCount (addr,ref count)== 2001)	
	MessageBox("成功");	
	Else	



MessageBox("失败");

SetOutPort

函数描述	public int SetOutPort(int ReaderAddr, byte port_num, byte level)	
功能	设置读写器输出端口的高低电平	
参数	参数	参数意义、范围
	ReaderAddr	读写器地址 ,供固定式读写器组 RS485 网络使用 ,默认为 0XFF (手持机与模块为无效参数 )。
	port_num	IO 口序号 : 0~1 ; 继电器 : 02
	level	输出电平 : 0 为低电平 , 1 为高电平
返回值	返回值 : 2001 为成功。	
应用实例	If(SetOutPort(255 , 0,1)==2001) MessageBox("成功"); Else MessageBox("失败");	

BuzzerLEDON

函数描述	public int BuzzerLEDON(int ReaderAddr)	
功能	开启声光	
参数	参数	参数意义、范围
	ReaderAddr	读写器地址 ,供固定式读写器组 RS485 网络使用 ,默认为 0XFF (手持机与模块为无效参数 )。
返回值	返回值 : 2001 为成功。	
应用实例	If(BuzzerLEDON (255)==2001) MessageBox("成功"); Else MessageBox("失败");	

BuzzerLEDOFF

函数描述	public int BuzzerLEDOFF(int ReaderAddr)	
功能	关闭声光	
参数	参数	参数意义、范围
	ReaderAddr	读写器地址 ,供固定式读写器组 RS485 网络使用 ,默认为 0XFF (手持机与模块为无效参数 )。

返回值            返回值：2001 为成功。

应用实例            If(BuzzerLEDOFF (255)==2001)  
                           MessageBox("成功");  
                           Else  
                           MessageBox("失败");

## GetBuzzerLED

函数描述	public int GetBuzzerLED(int ReaderAddr,ref byte state)	
功能	查询声光信息	
参数	参数	参数意义、范围
	ReaderAddr	读写器地址 ,供固定式读写器组 RS485 网络使用 ,默认为 0XFF (手持机与模块为无效参数 )。
	state	获取到的声光信息
返回值	返回值：2001 为成功。	
应用实例	If(GetBuzzerLED(255 , state)==2001) MessageBox("成功"); Else MessageBox("失败");	

## GetTcpParameter

函数描述	public int GetTcpParameter(int readerAddr,ref string strIP, ref string strMark, ref string strGate, ref int nTcpPort)	
功能	查询读写器 TCP 网络通讯端口的 IP 地址、子网掩码、缺省网关、TCP 端口号参数。	
参数	参数	参数意义、范围
	readerAddr	读写器地址 ,供固定式读写器组 RS485 网络使用 ,默认为 0XFF (手持机与模块为无效参数 )。
	strIP	读写器网口地址 ,是一个 IP 地址的地址的字符串格式的有效表示形式 ,必须和被连接的电脑处于同一个网段 (默认 : "192.168.1.200")
	strMark	子网掩码地址 ,是一个子网掩码的字符串格式的有效表示形式
	strGate	网关地址 ,是一个网关的字符串格式的有效表示形式
返回值	nTcpPort	设备端口号(范围 1~65535)
	返回值：2001 为成功。	

```

应用实例      If(GetTcpParameter (255 , ref strIP ,ref strMark ,ref strGate,ref
                nTcpPort)==2001)
                MessageBox( "成功" );
Else
                MessageBox( "失败" );

```

## SetTcpParameter

函数描述	<pre>public int SetTcpParameter(int readerAddr,string strIP, string strMark,                            string strGate, int nTcpPort)</pre>	
功能	设置读写器 TCP 网络通讯端口的 IP 地址、子网掩码、缺省网关、TCP 端口号参数。	
参数	参数	参数意义、范围
	readerAddr	读写器地址 ,供固定式读写器组 RS485 网络使用 ,默认为 0XFF (手持机与模块为无效参数)。
	strIP	读写器网口地址 , 是一个 IP 地址的地址的字符串格式的有效表示形式,必须和被连接的电脑处于同一个网段(默认 : "192.168.1.200")
	strMark	子网掩码地址 , 是一个子网掩码的字符串格式的有效表示形式
	strGate	网关地址 , 是一个网关的字符串格式的有效表示形式
	nTcpPort	设备端口号(范围 1~65535)
返回值	返回值 : 2001 为成功。	
应用实例	<pre> If(SetNetSetting(255 , strIP , strMark , strGate, nTcpPort)==2001)     MessageBox("成功"); else     MessageBox("失败"); </pre>	

## SetMacAddress

函数描述	<pre>public int SetMacAddress(int ReaderAddr, string[] strMacAddr)</pre>	
功能	设置读写器的 MAC 地址。	
参数	参数	参数意义、范围
	readerAddr	读写器地址 ,供固定式读写器组 RS485 网络使用 ,默认为 0XFF (手持机与模块为无效参数)。
	strMacAddr	需要设置的 MAC 地址

返回值            返回值：2001 为成功。

应用实例            If(SetMacAddress(255 , strMacAddr)==2001)

                      MessageBox("成功");

                      else

                      MessageBox("失败");

## GetMacAddress

函数描述	public int GetMacAddress(int ReaderAddr,ref string[] strMacAddr)	
功能	获取读写器的 MAC 地址。	
参数	参数	参数意义、范围
	readerAddr	读写器地址 ,供固定式读写器组 RS485 网络使用 ,默认为 0XFF ( 手持机与模块为无效参数 )。
	strMacAddr	获取到的 MAC 地址
返回值	返回值：2001 为成功。	
应用实例	If(GetMacAddress(255 , ref strMacAddr)==2001)	
	MessageBox("成功");	
	else	
	MessageBox("失败");	

## SetSerialNo

函数描述	public int SetSerialNo(int ReaderAddr, string[] strSerialNo)	
功能	设置读写器的序列号。	
参数	参数	参数意义、范围
	readerAddr	读写器地址 ,供固定式读写器组 RS485 网络使用 ,默认为 0XFF ( 手持机与模块为无效参数 )。
	strSerialNo	需要设置的序列号
返回值	返回值：2001 为成功。	
应用实例	If(SetSerialNo (255 , strSerialNo)==2001)	
	MessageBox("成功");	
	else	
	MessageBox("失败");	

## GetSerialNo

函数描述	public int GetSerialNo(int ReaderAddr, string[] strSerialNo)	
功能	设置读写器的序列号。	
参数	参数	参数意义、范围
	readerAddr	读写器地址 ,供固定式读写器组 RS485 网络使用 ,默认为 0XFF (手持机与模块为无效参数)。
	strSerialNo	获取到的序列号
返回值	返回值 : 2001 为成功。	
应用实例	<pre>If(GetSerialNo (255 , ref strSerialNo)==2001)     MessageBox("成功"); else     MessageBox("失败");</pre>	

## 1-2.ISO18000-6B 标签操作函数

### IsoMultiTagIdentify

函数描述	public int IsoMultiTagIdentify(int readerAddr,ref byte[] tag_buf, ref byte tag_cnt)	
功能	ISO18000-6B 多标签识别。	
参数	参数	参数意义、范围
	readerAddr	读写器地址 ,供固定式读写器组 RS485 网络使用 ,默认为 0XFF (手持机与模块为无效参数)。
	tag_buf	识别到的数据 (包括标签数据 ,天线号等)
	tag_cnt	标签数量
返回值	返回值 : 2001 为成功。	
应用实例	<pre>if(IsoMultiTagIdentify(255,ref tag_buf,ref tag_cnt)==2001)     MessageBox( "成功" ); else     MessageBox( "失败" );</pre>	

### IsoMultiTagRead

函数描述	public int IsoMultiTagRead(int ReaderAddr, int startAddr, ref byte[]
------	--

	tag_buf, ref int tag_cnt, ref int getCount)	
功能	ISO18000-6B 多标签用户数据读取。	
参数	参数	参数意义、范围
	readerAddr	读写器地址，供固定式读写器组 RS485 网络使用，默认为 0xFF（手持机与模块为无效参数）。
	startAddr	
	tag_buf	识别到的数据（包括标签数据，天线号等）
	tag_cnt	识别到的标签数量
	getCount	获取到的标签数量
返回值	返回值：2001 为成功。	
应用实例	<pre> if(IsoMultiTagRead (255,startAddr,ref tag_buf,ref tag_cnt,ref getCount)==2001)     MessageBox(“成功”); else     MessageBox(“失败”); </pre>	

## IsoReadWithID

函数描述	public int IsoReadWithID(int ReaderAddr, byte[] byTagID, byte byAddress, ref byte[] byLabelData, ref byte byAntenna)	
功能	读取指定 ID 号的标签数据，每次读取指定地址开始的 8 字节数据	
参数	参数	参数意义、范围
	ReaderAddr	读写器地址
	byTagID	指定标签的 TID
	byAddress	读取起始地址
	byLabelData	读取的标签数据
	byAntenna	天线号
返回值	返回值：2001 为成功。	
应用实例	<pre> if(IsoReadWithID(255 , byTagID, byAddress, ref byLabelData, ref byAntenna)==2001) </pre>	

```

        MsgBox( "成功" );
    else
        MsgBox( "失败" );
    }

```

## IsoWriteWithID

函数描述	public int IsoWriteWithID(int readerAddr,byte[] byTagID, byte byAddress, byte byValue)	
功能	向指定 ID 号的标签中写入数据	
参数	参数	参数意义、范围
	readerAddr	读写器地址
	byTagID	指定标签的 TID
	byAddress	写入数据的起始地址
	byValue	写入标签的数据
返回值	返回值：2001 为成功。	
应用实例	if(IsoWriteWithID(255,byTagID, byAddress, byValue)==2001)                 MsgBox( "成功" );             else                 MsgBox( "失败" );	

## IsoLockWithID

函数描述	public int IsoLockWithID(int ReaderAddr, byte[] byTagID, byte byAddress)	
功能	给指定 ID 号的标签上锁	
参数	参数	参数意义、范围
	ReaderAddr	读写器地址
	byTagID	指定标签的 TID
	byAddress	写入数据的起始地址
返回值	返回值：2001 为成功。	
应用实例	if(IsoWriteWithID(255 , byTagID, byAddress)==2001)	

```

        MessageBox( "成功" );
else
    MessageBox( "失败" );

```

## IsoRead

函数描述	public int IsoRead(int readerAddr,byte addr, ref byte[] value)
------	--

功能 从 6B 标签中读取数据

参数	参数	参数意义、范围
	readerAddr	读写器地址
	addr	读取的起始地址
	value	读取到的数据

返回值 返回值：2001 为成功。

应用实例 if(IsoRead (255,addr, value)==2001)

```

        MessageBox( "成功" );
else
    MessageBox( "失败" );

```

## IsoWrite

函数描述	public int IsoWrite(int readerAddr,byte addr, byte value)
------	---

功能 向 6B 标签中写入数据

参数	参数	参数意义、范围
	readerAddr	读写器地址
	addr	写入数据的起始地址
	value	写入到标签的数据

返回值 返回值：2001 为成功。

应用实例 if(IsoWrite (255 , addr, value)==2001)

```

        MessageBox( "成功" );
else
    MessageBox( "失败" );

```



## IsoLock

函数描述	public int IsoLock((int readerAddr,byte addr)	
功能	对指定的 6B 标签地址进行锁定，一旦锁定，该地址不能被解锁	
参数	参数	参数意义、范围
	readerAddr	读写器地址
	addr	要锁定的地址
返回值	返回值：2001 为成功。	
应用实例	<pre>if(IsoLock (255 , addr))==2001)      MessageBox( "成功" ); else     MessageBox( "失败" );</pre>	

## IsoQueryLock

函数描述	public int IsoQueryLock(int ReaderAddr, byte addr, ref byte lstate)	
功能	Iso18000 锁定查询	
参数	参数	参数意义、范围
	readerAddr	读写器地址
	addr	要锁定的地址
	lstate	锁定状态
返回值	返回值：2001 为成功。	
应用实例	<pre>if(IsoQueryLock (255 , addr , ref lstate))==2001)      MessageBox( "成功" ); else     MessageBox( "失败" );</pre>	

## IsoQueryLockWithUID

函数描述	public int IsoQueryLock(int ReaderAddr, byte[] byTagID,byte addr, ref byte lstate)	
功能	Iso18000 对指定 ID 的标签锁定查询	
参数	参数	参数意义、范围
	readerAddr	读写器地址

	byTagID	指定标签的 ID
	addr	要锁定的地址
	lstate	锁定状态
返回值	返回值：2001 为成功。	
应用实例	<pre> if(IsoQueryLock (255 , id,addr,ref lstate))==2001)      MessageBox( "成功" ); else     MessageBox( "失败" ); </pre>	

## 1-3. EPC GEN2 标签操作函数

### EpcMultiTagIdentify

函数描述	<pre> public int EpcMultiTagIdentify(int readerAddr,ref byte[,] tag_buf, ref byte tag_cnt, ref byte tag_flag) </pre>	
功能	EPC GEN2 多标签识别	
参数	参数	参数意义、范围
	readerAddr	读写器地址
	tag_buf	识别到的数据（包括标签数据，天线号等）
	tag_cnt	标签数量
	tag_flag	状态标识，0 是读取成功,1 是读取失败
返回值	返回值：2001 为成功。	
应用实例	<pre> if(Gen2MultiTagIdentify(255 , ref tag_buf, ref tag_cnt, ref tag_flag)==2001)      MessageBox( "成功" ); else     MessageBox( "失败" ); </pre>	

### EpcRead

函数描述	<pre> public int EpcRead(int readerAddr,e membank, byte wordptr, byte wordcnt, ref byte[] value) </pre>
------	---

功能	从指定地址开始连续读取指定长度的数据	
参数	参数	参数意义、范围
	readerAddr	读写器地址
	membank	读取区域,销毁密码和访问密码区为 0 ,EPC 编码区为 1 ,USER 区为 3 区
	wordptr	起始地址
	wordcnt	读取数据的长度,单位为字
	value	读取到的标签数据
返回值	返回值: 2001 为成功。	
应用实例	<pre>if(EpcRead (255, membank, wordptr, wordcnt, ref value)==2001)     MessageBox( "成功" ); else     MessageBox( "失败" );</pre>	

## EpcWrite

函数描述	<pre>public int EpcWrite(int ReaderAddr, byte membank, byte wordptr,                     ushort value)</pre>
------	--

功能	从指定地址开始连续写入指定的数据,一次只能写一个字	
参数	参数	参数意义、范围
	ReaderAddr	读写器地址
	membank	读取区域,销毁密码和访问密码区为 0 ,EPC 编码区为 1 ,USER 区为 3 区
	wordptr	起始地址
	value	写入标签的数据
返回值	返回值: 2001 为成功。	
应用实例	<pre>if(EpcWrite (255 , membank, wordptr, value)==2001)     MessageBox( "成功" ); else     MessageBox( "失败" );</pre>	

## Gen2MultiTagWrite

函数描述	<pre>public int Gen2MultiTagWrite(int ReaderAddr, int membank, int wordaddr, int wordLen, string strValue, ref int writeCount)</pre>
------	--

功能	EPC Gen2 多标签写入	
参数	参数	参数意义、范围
	ReaderAddr	读写器地址
	membank	读取区域 ,销毁密码和访问密码区为 0 ,EPC 编码区为 1 ,USER 区为 3 区
	wordaddr	起始地址
	wordLen	写入数据的长度
	strValue	写入的字符串
	writeCount	写成功的标签数
返回值	返回值：2001 为成功。	
应用实例	<pre>if(EpcWrite (255 , membank, wordaddr , wordLen , strValue , ref writecount )==2001)     MessageBox( "成功" ); else     MessageBox( "失败" );</pre>	

## Gen2MultiTagRead

函数描述	<pre>public int Gen2MultiTagRead(int ReaderAddr, byte MembankMask, byte ResWordPtr, byte ResWordCnt, byte EpcWordPtr, byte EpcWordCnt, byte TidWordPtr, byte TidWordCnt, byte UserWordPtr, byte UserWordCnt, ref int ReadCnt)</pre>
------	---

功能	EPC Gen2 多标签读取	
参数	参数	参数意义、范围
	ReaderAddr	读写器地址
	MembankMask	存储区选择，掩码表示。从第 1 到 4 位分别表示保留区、EPC 区、TID 区和 USER 区

	ResWordPtr	保留区读取字地址
	ResWordCnt	读取字数
	EpcWordPtr	EPC 区读取首地址
	EpcWordCnt	读取字数
	TidWordPtr	TID 区读取首地址
	TidWordCnt	读取字数
	UserWordPtr	User 区读取首地址
	UserWordCnt	读取字数
	ReadCnt	读取到的数据组数
返回值	返回值：2001 为成功。	
应用实例	<pre> if(EpcWrite (255 , MembankMask, ResWordPtr, ResWordCnt, EpcWordPtr, EpcWordCnt, TidWordPtr, TidWordCnt, UserWordPtr, UserWordCnt, ref ReadCnt)==2001)      MessageBox( "成功" ); else     MessageBox( "失败" ); </pre>	

## Gen2SecLock

函数描述	<pre> public int Gen2SecLock(int ReaderAddr, uint AccPassWord, byte Membank, byte Level) </pre>
------	---

功能	EPC GEN2 标签加锁，使 EPC 标签只有提供正确密码才能安全写入	
参数	参数	参数意义、范围
	ReaderAddr	读写器地址
	AccPassWord	标签密码区数据
	Membank	锁写的区域，0 为保留区，1 为 EPC，2 为 TID，3 为用户区
	Level	安全锁级别（锁定等级，0 为不锁定，1 为永久不锁定，2 为安全锁定，3 为全锁定。）
返回值	返回值：2001 为成功。	
应用实例	<pre>if(Gen2SecLock(255, AccPassWord, Membank, Level)==2001)     MessageBox( "成功" ); else     MessageBox( "失败" );</pre>	

## EpcLockTag

函数描述	public int EpcLockTag(int readerAddr,byte MemBank)	
功能	EPC GEN2 标签加锁，使 EPC 标签只有提供正确密码才能安全写入	
参数	参数	参数意义、范围
	readerAddr	读写器地址
	Membank	byMemBank 锁写的区域，0 为保留区，1 为 EPC，2 为 TID，3 为用户区
返回值	返回值：2001 为成功。	
应用实例	<pre>if(EpcLockTag (255 , MemBank)==2001)      MessageBox( "成功" ); else     MessageBox( "失败" );</pre>	

## EpcInitEpc

函数描述	public int EpcInitEpc(int readerAddr,byte bit_cnt)	
功能	初始化标签的 EPC 编码区长度，默认初始化为 96 位（6 个字），并赋值为全 0	
参数	参数	参数意义、范围
	readerAddr	读写器地址
	bit_cnt	初始化数据长度
返回值	返回值：2001 为成功。	
应用实例	<pre>if(EpcInitEpc (255 , bit_cnt)==2001)      MessageBox( "成功" ); else     MessageBox( "失败" );</pre>	

## Gen2SecWrite

函数描述	public int Gen2SecWrite(int ReaderAddr, uint AccPassWord, byte
------	--

	Membank, byte WordAddr, ushort Value)	
功能	EPC GEN2 标签数据安全写	
参数	参数	参数意义、范围
	readerAddr	读写器地址
	AccPassWord	标签密码区数据
	Membank	byMemBank 锁写的区域，0 为保留区，1 为 EPC，2 为 TID，3 为用户区
	WordAddr	安全写的起始地址
	Value	需要安全写入的数据
返回值	返回值：2001 为成功。	
应用实例	<pre> if(Gen2SecWrite(255, AccPassWord, Membank, WordAddr, Value)==2001)     MessageBox( "成功" ); else     MessageBox( "失败" ); </pre>	

### Gen2SecRead

函数描述	public int Gen2SecRead(int ReaderAddr, uint AccPassWord, byte Membank, byte WordAddr, byte WordCnt, ref byte[] value)	
功能	EPC GEN2 标签数据安全读	
参数	参数	参数意义、范围
	readerAddr	读写器地址
	AccPassWord	标签密码区数据
	Membank	byMemBank 锁写的区域，0 为保留区，1 为 EPC，2 为 TID，3 为用户区
	WordAddr	安全读的起始地址
	WordCnt	安全读取数据的长度
	value	读取到的标签数据
返回值	返回值：2001 为成功。	
应用实例	<pre> if(Gen2SecRead(255, AccPassWord, Membank, WordAddr, WordCnt, ref value)==2001)     MessageBox( "成功" ); else     MessageBox( "失败" ); </pre>	

## Gen2SelectConfig

函数描述	<pre>public int Gen2SelectConfig(int ReaderAddr, int Action, int Membank,                              int wordAddr, int wordCnt, string[] words)</pre>
------	---

**功能** GEN2 配置标签筛选功能参数

<b>参数</b>	<table><tr><th>参数</th><th>参数意义、范围</th></tr><tr><td>readerAddr</td><td>读写器地址</td></tr><tr><td>Action</td><td>0 表示选择匹配的，1 表示选择不匹配的</td></tr><tr><td>Membank</td><td>byMemBank 匹配的区域，0 为保留区，1 为 EPC，2 为 TID，3 为用户区</td></tr><tr><td>WordAddr</td><td>筛选数据的起始地址</td></tr><tr><td>WordCnt</td><td>需要筛选的数据长度</td></tr><tr><td>words</td><td>筛选数据</td></tr></table>	参数	参数意义、范围	readerAddr	读写器地址	Action	0 表示选择匹配的，1 表示选择不匹配的	Membank	byMemBank 匹配的区域，0 为保留区，1 为 EPC，2 为 TID，3 为用户区	WordAddr	筛选数据的起始地址	WordCnt	需要筛选的数据长度	words	筛选数据
参数	参数意义、范围														
readerAddr	读写器地址														
Action	0 表示选择匹配的，1 表示选择不匹配的														
Membank	byMemBank 匹配的区域，0 为保留区，1 为 EPC，2 为 TID，3 为用户区														
WordAddr	筛选数据的起始地址														
WordCnt	需要筛选的数据长度														
words	筛选数据														

**返回值** 返回值：2001 为成功。

**应用实例**

```
if(Gen2SelectConfig(255, action, Membank, WordAddr, WordCnt,
words)==2001)
    MessageBox( "成功" );
else
    MessageBox( "失败" );
```

## Gen2SetAccPwd

函数描述	<pre>public int Gen2SetAccPwd(uint AccPassWord)</pre>
------	---

**功能** EPC GEN2 标签访问密码设置

<b>参数</b>	<table><tr><th>参数</th><th>参数意义、范围</th></tr><tr><td>AccPassWord</td><td>要设定的密码区数据</td></tr></table>	参数	参数意义、范围	AccPassWord	要设定的密码区数据
参数	参数意义、范围				
AccPassWord	要设定的密码区数据				

**返回值** 返回值：2001 为成功。

**应用实例**

```
if(Gen2SecRead(AccPassWord)==2001)
    MessageBox( "成功" );
else
    MessageBox( "失败" );
```



## Gen2KillTag

函数描述	public int Gen2KillTag(int ReaderAddr, uint AccPassWord)	
功能	EPC GEN2 标签销毁	
参数	参数	参数意义、范围
	ReaderAddr	读写器地址
	AccPassWord	标签销毁区密码数据
返回值	返回值：2001 为成功。	
应用实例	<pre>if(Gen2KillTag (255 , AccPassWord)==2001)      MessageBox( "成功" ); else     MessageBox( "失败" );</pre>	

## 1-4.SO18000-6B 标签数据处理函数

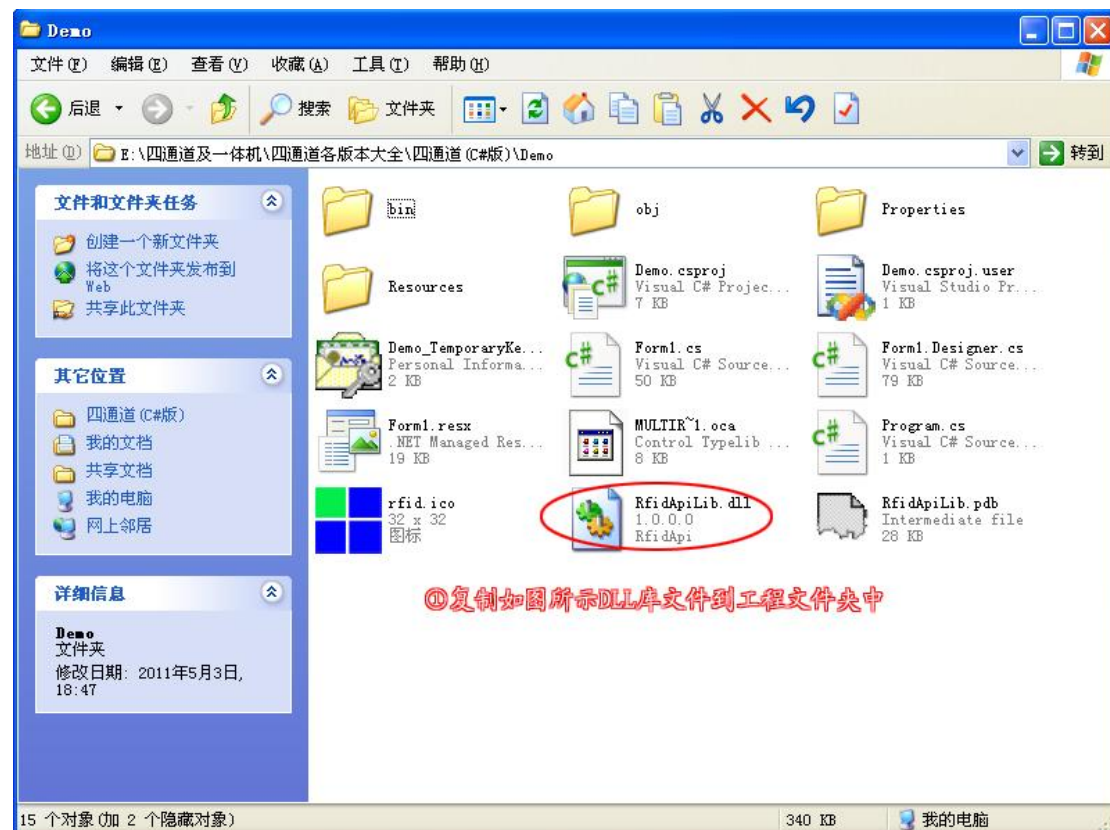
### ClearIdBuf

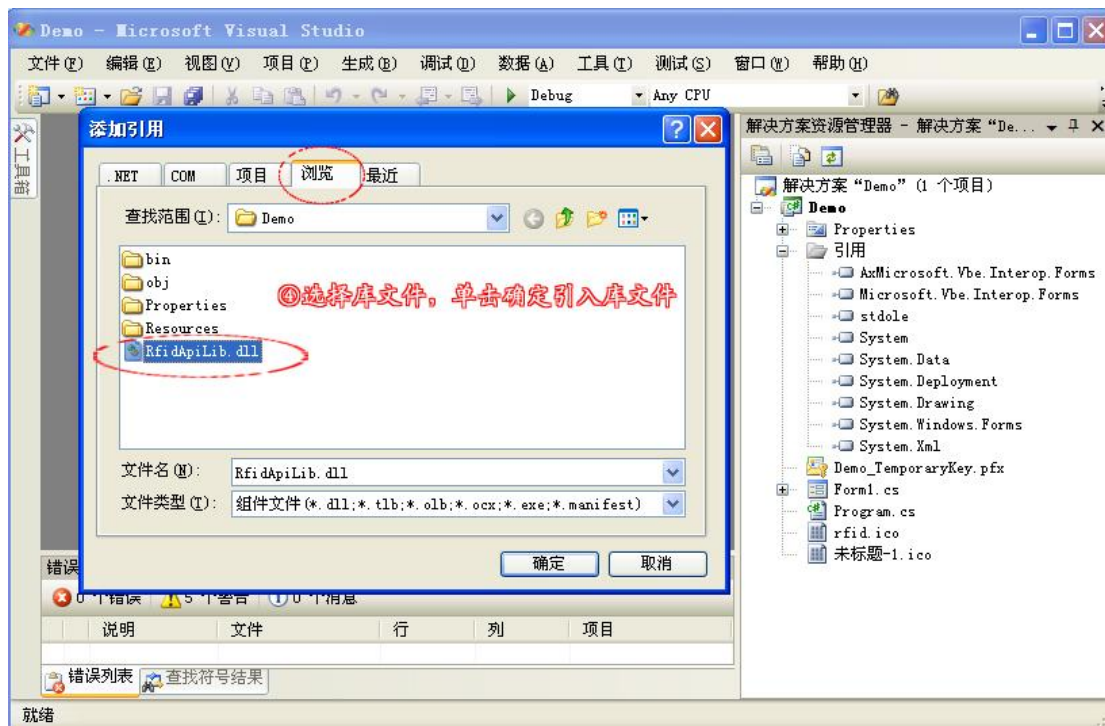
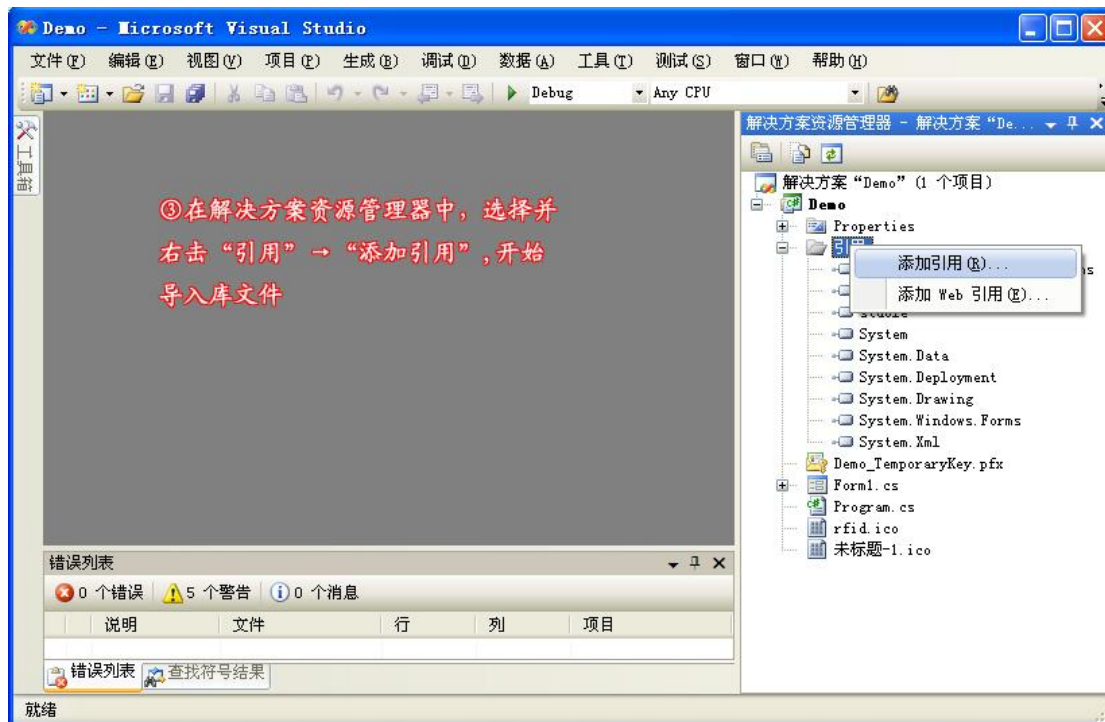
函数描述	public int ClearIdBuf(int readerAddr)	
功能	清空读写器标签数据缓存区，在每次多标签识别前可使用	
参数	readerAddr	读写器地址
返回值	返回值：2001 为成功。	
应用实例	<pre>if(ClearIdBuf (255)==2001)     MessageBox( "成功" ); else     MessageBox( "失败" );</pre>	

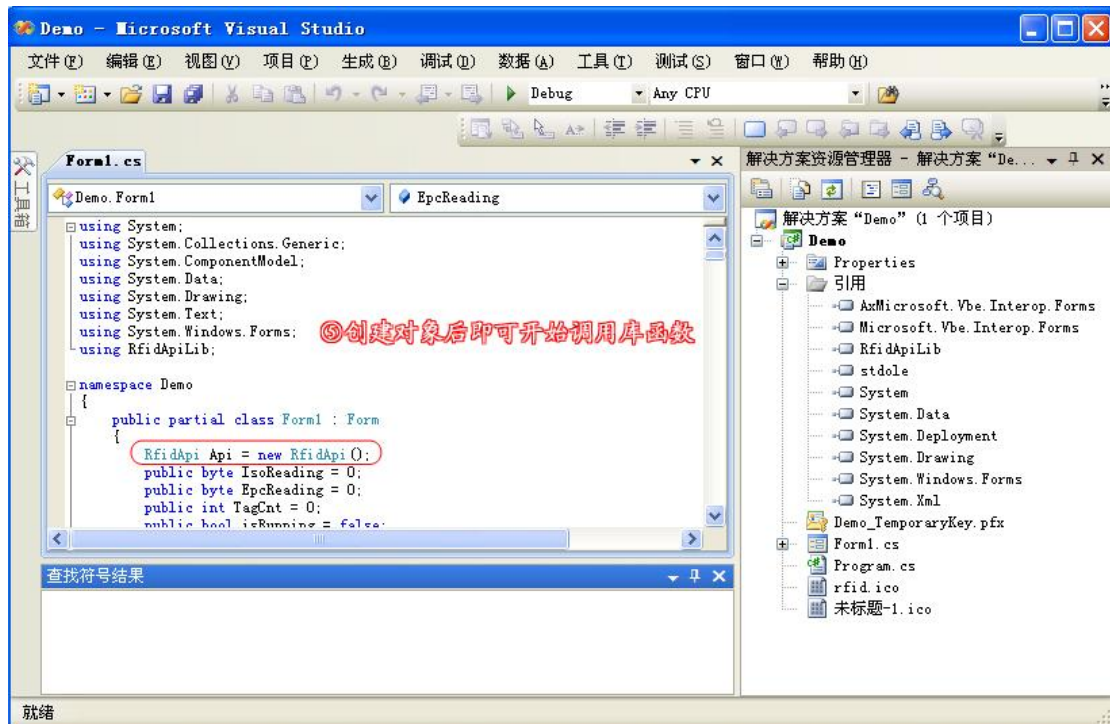
## 2.源程序样例

### 2-1 . 建立工程

## 2-2 . 添加动态库 RfidApiLib.dll 引用到工程项目中。( 如图所示 )



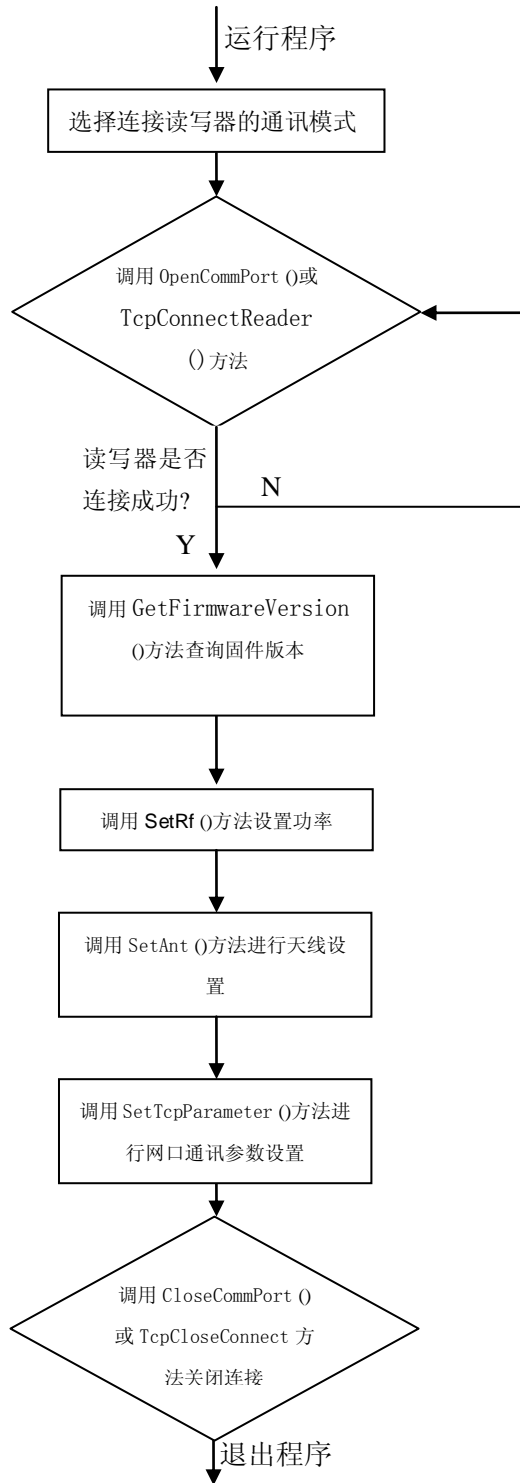




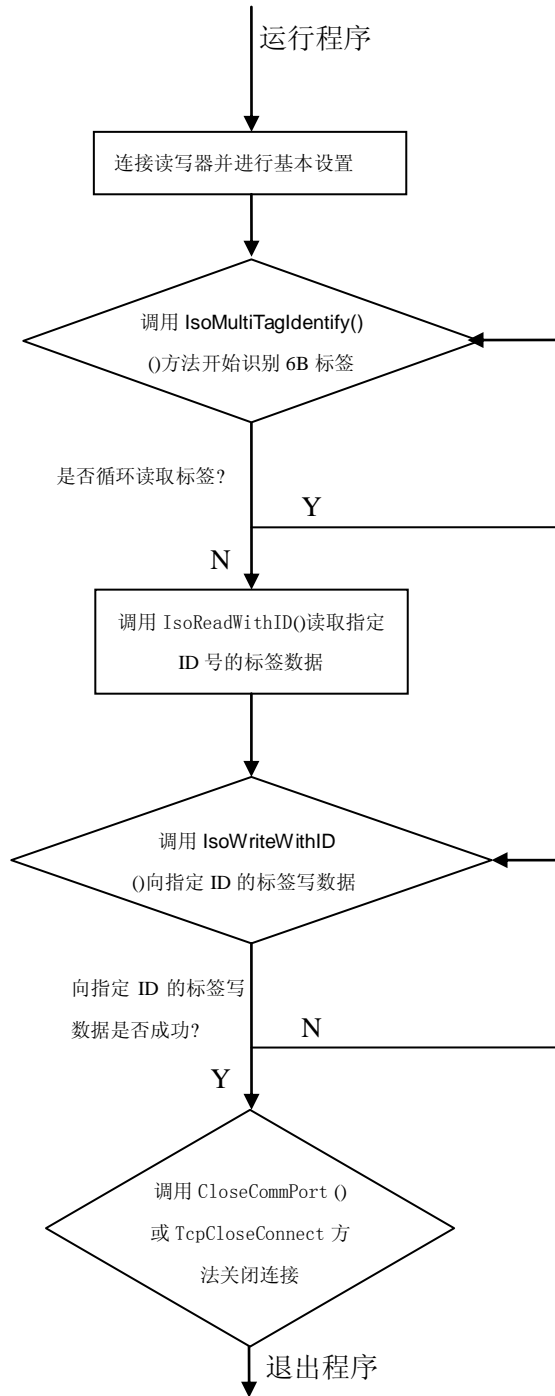
添加了动态库 RfidApiLib.dll 引用到工程项目并创建对象后 ,就可以直接调用 dll 里面的库函数了。

## 2-3 . 调用库函数实现相应的功能

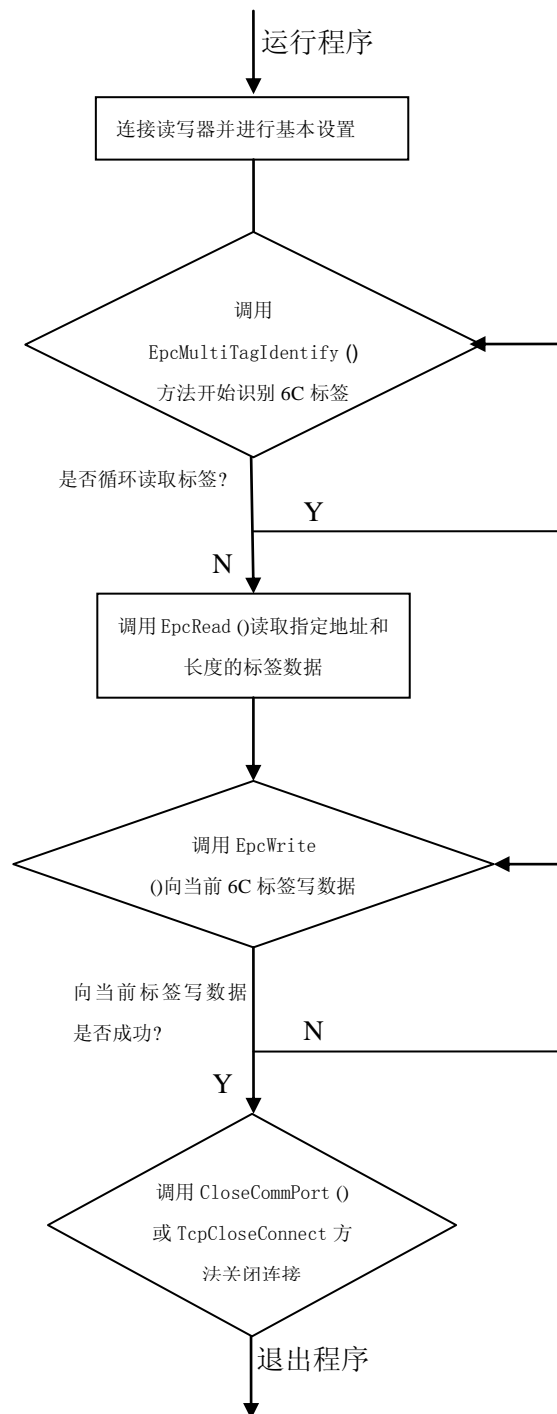
读写器基本设置与管理流程图如下:



**ISO18000-6B 标签操作流程图如下:**



**EPC GEN2 标签操作流程图如下:**



(1). 调用 OpenCommPort ()方法连接读写器。

```

int status;
byte v1 = 0;
byte v2 = 0;
string s = "";
status =
Api.OpenCommPort(cCommPort.SelectedItem.ToString(), Convert.ToInt32(cBaudrate.Text));
if (status != RfidApi.SUCCESS_RETURN)

```

```

{
    lInfo.Items.Add("Open Comm Port Failed! ");
    return;
}
status = Api.GetFirmwareVersion(ref v1, ref v2);
if (status != RfidApi.SUCCESS_RETURN)
{
    lInfo.Items.Add("Can not connect with the reader! ");
    Api.CloseCommPort();
    return;
}
lInfo.Items.Add("Connect the reader success! ");
s = string.Format("The reader's firmware version is:V{0:d2}.{1:d2}", v1, v2);
s = s + " ";
lInfo.Items.Add(s);

```

(2). 调用 SetRf ()方法设置功率。

```

byte pwr = 0;
byte freq = 0;

int status;
pwr = (byte)(tRfPwr.Value);
freq = (byte)(cRfFreq.SelectedIndex);
status = Api.SetRf(pwr, freq);
if (status != RfidApi.SUCCESS_RETURN)
{
    lInfo.Items.Add("Set Rf settings failed! ");
    return;
}
lInfo.Items.Add("Set Rf settings success! ");

```

(3). 调用 SetAnt ()方法进行天线设置。

```

byte ant_sel = 0;
byte antH = 0;
int status;

if (ant1.Checked)
    ant_sel |= 0x01;
if (ant2.Checked)
    ant_sel |= 0x02;
if (ant3.Checked)
    ant_sel |= 0x04;
if (ant4.Checked)
    ant_sel |= 0x08;

```



```

status = Api.SetAnt(ant_sel, antH);
if (status != RfidApi.SUCCESS_RETURN)
{
    lInfo.Items.Add("Set ant failed! ");
    return;
}
lInfo.Items.Add("Set ant success! ");

```

( 4 ). 调用 SetTcpParameter ()方法进行网口通讯参数设置。

```

int status=0;
string strIp="";
string strMark = "";
string strGate = "";
int nTcpPort =0;
try {
    strIp = txtNetIP.Text;
    strMark = txtSubNet.Text;
    strGate = txtDefaultGate.Text;
    nTcpPort = int.Parse(txtTcpPort.Text);
}
catch (Exception)
{
    lInfo.Items.Add("Please input all the parameter ! ");
    return;
}
status=Api.SetTcpParameter(strIp, strMark, strGate, nTcpPort);
if (status != RfidApi.SUCCESS_RETURN) {
    lInfo.Items.Add("Setting the TcpParameter Fail,please try again. ");
    return;
}
lInfo.Items.Add("Setting the TcpParameter successful. ");

```

( 5 ). 调用 IsoMultiTagIdentify() ()方法开始识别 6B 标签。

```

int status;
int i, j;
byte[, ] IsoBuf = new byte[100, 14];
byte tag_cnt = 0;
string s = "";
string sl = "";

status = Api.IsoMultiTagIdentify(ref IsoBuf, ref tag_cnt);
if (tag_cnt > 0)
{

```

```

for (i = 0; i < tag_cnt; i++)
{
    s1 = string.Format("NO. {0:D}:", TagCnt);
    s1 += string.Format("[ANT{0:D}]", IsoBuf[i, 1] + 1);
    for (j = 2; j < 10; j++)
    {
        s = string.Format("{0:X2} ", IsoBuf[i, j]);
        s1 += s;
    }
    s1 = s1.Substring(0, s1.Length - 1);
    if (lInfo.Items.Count > 1000)
        lInfo.Items.Clear();
    lInfo.Items.Add(s1);
    TagCnt++;
}
}

```

(6). 调用 IsoReadWithID()读取指定 ID 号的标签数据。

```

int addr;
int len;
int i = 0;
int status = 0;
byte byAntenna = 0;
byte[] TagID = new byte[16];
byte[] value = new byte[32];
string s = "The data is:";
string s1 = "";
try
{
    addr = int.Parse(tIsoAddr.Text);
    len = int.Parse(tIsoCnt.Text);
}
catch (Exception)
{
    lInfo.Items.Add("Please input ByteAddr and ByteCnt ");
    return;
}
string hexValues = txtTagID.Text;
string[] hexValuesSplit = hexValues.Split(' ');
try
{
    foreach (String hex in hexValuesSplit)
    {

```

```

        int x = Convert.ToInt32(hex, 16);
        TagID[i++] = (byte)x;
    }
}
catch (Exception)
{
    lInfo.Items.Add("Please input Tag ID needed ");
    return;
}

if (i != 8)
{
    lInfo.Items.Add("Please input Tag ID needed ");
    return;
}

for (i = 0; i < len; )
{
    status = Api.IsoReadWithID(TagID, (byte)addr, ref value, ref byAntenna);
    if (status != RfidApi.SUCCESS_RETURN)
    {
        lInfo.Items.Add("Read failed! ");
        return;
    }

    for (int j = 0; j < 8; j++)
    {
        s1 = string.Format("{0:X2}", value[j]);
        s += s1;
        if (i + j >= len - 1)
            break;
    }
    i += 8;
}
if (status == 2001)
{
    s += " ";
    lInfo.Items.Add("Read success! ");
    lInfo.Items.Add(s);
}
}

```

( 7 ). IsoWriteWithID ()向指定 ID 的标签写数据。

```

int addr;
int len;

```

```

int i = 0;
int status = 0;
byte byAntenna = 0;
byte[] TagID = new byte[16];
byte[] value = new byte[32];
string s = "The data is:";
string s1 = "";
try
{
    addr = int.Parse(tIsoAddr.Text);
    len = int.Parse(tIsoCnt.Text);
}
catch (Exception)
{
    lInfo.Items.Add("Please input ByteAddr and ByteCnt ");
    return;
}
string hexID = txtTagID.Text;
string[] hexIDSplit = hexID.Split(' ');
try
{
    foreach (String hex in hexIDSplit)
    {
        int x = Convert.ToInt32(hex, 16);
        TagID[i++] = (byte)x;
    }
}
catch (Exception)
{
    lInfo.Items.Add("Please input Tag ID needed ");
    return;
}
string hexValues = tIsoData.Text;
string[] hexValuesSplit = hexValues.Split(' ');
try
{
    i = 0;
    foreach (String hex in hexValuesSplit)
    {
        if (hex != "")
        {
            int x = Convert.ToInt32(hex, 16);
            value[i++] = (byte)x;
        }
    }
}

```

```

    }
}
}
catch (Exception)
{
    lInfo.Items.Add("Please input data needed ");
    return;
}
if (i != len)
{
    lInfo.Items.Add("Please input data needed ");
    return;
}
for (i = 0; i < len; i++)
{
    status = Api.IsoWriteWithID(TagID, (byte)(addr + i), value[i]);
    if (status != RfidApi.SUCCESS_RETURN)
    {
        lInfo.Items.Add("Write failed! ");
        return;
    }
}
lInfo.Items.Add("Write success! ");

```

(8). 调用 EpcMultiTagIdentify () 方法开始识别 6C 标签。

```

int status;
int i, j;
byte[, ] IsoBuf = new byte[100, 14];
byte tag_cnt = 0;
string s = "";
string s1 = "";
byte tag_flag = 0;

if (!isNetConnect)
    return;
try
{
    status = Api.EpcMultiTagIdentify(ref IsoBuf, ref tag_cnt, ref tag_flag);
    if (status == 2009)
    {
        isNetConnect = false;
        return;
    }
}

```

```

    }
    catch (Exception ex)
    {
        System.Diagnostics.Debug.WriteLine(ex.ToString());
        isNetConnect = false;
        return;
    }
    if (tag_flag == 1)
        this.BackColor = Color.MediumBlue;
    else
        this.BackColor = Color.MidnightBlue;
    if (tag_cnt >= 100)
        return;
    if (tag_cnt > 0)
    {
        try
        {
            for (i = 0; i < tag_cnt; i++)
            {
                s1 = string.Format("NO. {0:D} : ", TagCnt);
                s1 += string.Format("[ANT{0:D}]", IsoBuf[i, 1]+1);
                for (j = 2; j < 14; j++)
                {
                    s = string.Format("{0:X2} ", IsoBuf[i, j]);
                    s1 += s;
                }
                if (lInfo.Items.Count >= 1000)
                    lInfo.Items.Clear();

                s1 = s1.Substring(0, s1.Length - 1);
                lInfo.Items.Add(s1);
                TagCnt++;
            }
        }
        catch
        {
        }
    }
}

```

(9). 调用 EpcRead ()读取指定地址和长度的标签数据。

```

int membank;
int wordptr;
int wordcnt;

```

```

int status = 0;
byte[] value = new byte[16];
string s = "The data is: ";
string s1 = "";
membank = cEpcMembank.SelectedIndex;
wordptr = cEpcWordptr.SelectedIndex;
wordcnt = cEpcWordcnt.SelectedIndex + 1;
status = Api.EpcRead((byte)membank, (byte)wordptr, (byte)wordcnt, ref value);
if (status != RfidApi.SUCCESS_RETURN)
{
    lInfo.Items.Add("Read failed! ");
    return;
}
else
{
    for (int i = 0; i < wordcnt * 2; i++)
    {
        s1 = string.Format("{0:X2}", value[i]);
        s += s1;
    }
    lInfo.Items.Add("Read success! ");
    s += " ";
    lInfo.Items.Add(s);
}

```

( 10 ). 调用 EpcWrite ()向当前 6C 标签写数据。

```

ushort[] value = new ushort[16];
int i = 0;
byte membank;
byte wordptr;
byte wordcnt;
int status;
string hexValues;

membank = (byte) (cEpcMembank.SelectedIndex);
wordptr = (byte) (cEpcWordptr.SelectedIndex);
wordcnt = (byte) (cEpcWordcnt.SelectedIndex+1);

hexValues = tEpcData.Text;
string[] hexValuesSplit = hexValues.Split(' ');
{
    foreach (String hex in hexValuesSplit)
    {
        if (hex != "")

```

```

        {
            int x = Convert.ToInt32(hex, 16);
            value[i++] = (ushort)x;
        }
    }
}

if (i != wordcnt)
{
    lInfo.Items.Add("Please input data needed ");
    return;
}

for(byte j = 0; j < wordcnt; j++)
{
    status = Api.EpcWrite(membank, (byte)(wordptr+j), value[j]);
    if (status != RfidApi.SUCCESS_RETURN)
    {
        lInfo.Items.Add("Write failed! ");
        return;
    }
}

lInfo.Items.Add("Write success! ");

```

( 11 ). 调用 Gen2SecLock ()方法进行 EPC GEN2 标签加锁，使 EPC 标签只有提供正确密码才能安全写入。

```

byte membank;
byte pwdLevel;

int status = 0;
byte[] value = new byte[16];

string s = "";
if (tEpcAccess.TextLength != 8)
{
    lInfo.Items.Add("Access Password length not enough ");
    return;
}

uint unAccPwd;
switch(cEpcMembank.SelectedIndex)
{
    case 0:
        membank=3;
        break;
    case 1:

```



```

        membank=2;
        break;
    case 2:
        membank=1;
        break;
    case 3:
        membank=0;
        break;
    default:
        membank=2;
        break;
}
pwdLevel = (byte)(cmbLevel.SelectedIndex);

unAccPwd = Convert.ToInt32(tEpcAccess.Text, 16);
status = Api.Gen2SecLock(unAccPwd, membank, pwdLevel);
if (status != RfidApi.SUCCESS_RETURN)
{
    lInfo.Items.Add("Lock EPC tag failed! ");
    return;
}
else
{
    lInfo.Items.Add("Lock EPC tag success! ");
    lInfo.Items.Add(s);
}

```

( 12 ). 调用 Gen2SecWrite()方法进行 EPC GEN2 标签数据安全写入。

```

ushort[] value = new ushort[16];
int i = 0;
byte membank;
byte wordptr;
byte wordcnt;
int status;
string hexValues;

membank = (byte)(cEpcMembank.SelectedIndex);
wordptr = (byte)(cEpcWordptr.SelectedIndex);
wordcnt = (byte)(cEpcWordcnt.SelectedIndex + 1);
if (tEpcAccess.TextLength != 8)
{
    lInfo.Items.Add("Access Password length not enough ");
    return;
}

```

```

uint unAccPwd;
unAccPwd = Convert.ToInt32(tEpcAccess.Text, 16);

hexValues = tEpcData.Text;
string[] hexValuesSplit = hexValues.Split(' ');
foreach (String hex in hexValuesSplit)
{
    // Convert the number expressed in base-16 to an integer.
    if (hex != "")
    {
        int x = Convert.ToInt32(hex, 16);
        value[i++] = (ushort)x;
    }
}
if (i != wordcnt)
{
    lInfo.Items.Add("Please input data needed ");
    return;
}
for (byte j = 0; j < wordcnt; j++)
{
    status = Api.Gen2SecWrite(unAccPwd, membank, (byte)(wordptr + j), value[j]);
    if (status != RfidApi.SUCCESS_RETURN)
    {
        lInfo.Items.Add("Write failed! ");
        return;
    }
}
lInfo.Items.Add("Write success! ");

```

(13). 调用 Gen2KillTag ()方法进行标签销毁。

```

int status = 0;
byte[] value = new byte[16];

string s = "";
if (tEpcAccess.TextLength != 8)
{
    lInfo.Items.Add("Access Password length not enough ");
    return;
}
uint unAccPwd;
unAccPwd = Convert.ToInt32(tEpcAccess.Text, 16);
status = Api.Gen2KillTag(unAccPwd);
if (status != RfidApi.SUCCESS_RETURN)

```


```

{
    lInfo.Items.Add("Set Password failed! ");
    return;
}
else
{
    lInfo.Items.Add("Set Password success! ");
    lInfo.Items.Add(s);
}
}

```

## 3. 演示软件操作方法、步骤

### 3-1 . 启动“ 演示软件”

单击本软件图标  , 启动软件，单击 “Connect” 连接读写器，若成功提示如图 3-1，点击【查询】查询当前的天线参数，点击【设置】，用户可根据实际需要设定天线的发射功率，用户可以根据实际需求更改天线的频率和发射功率（0~30dBm）。

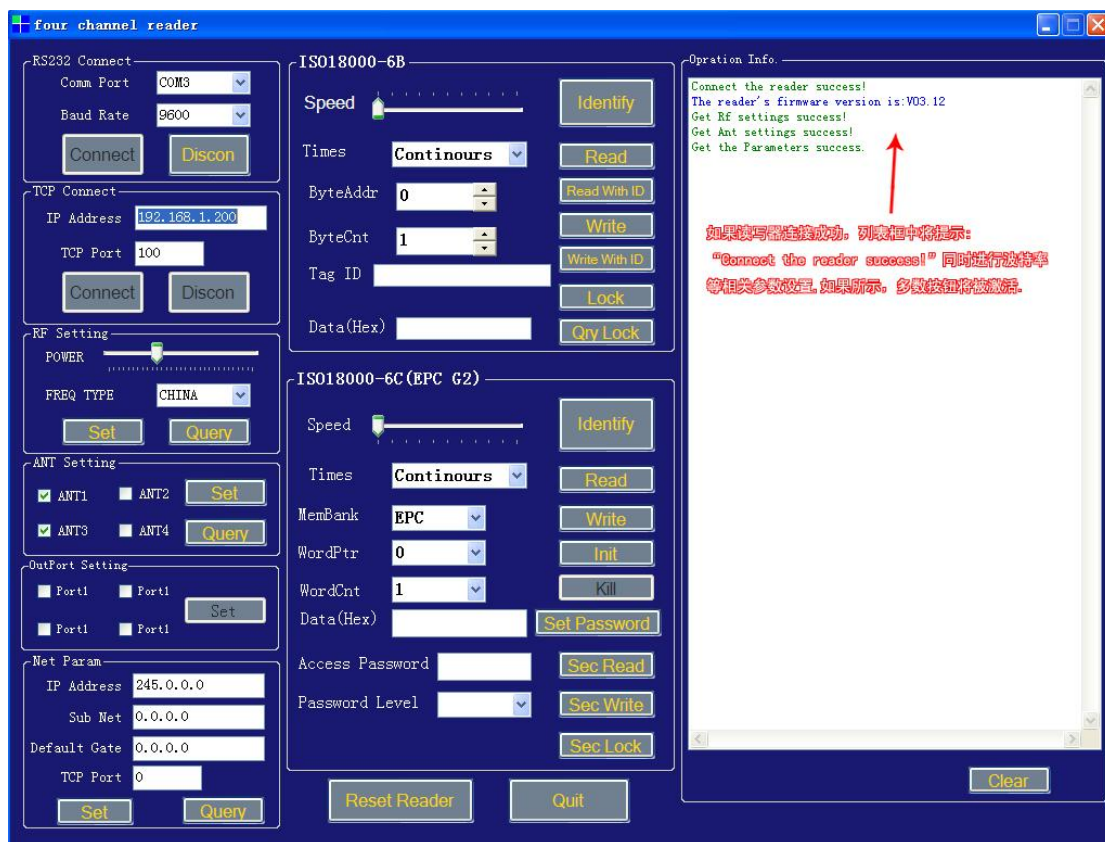


图 3-1

### 3-2 . ISO18000-6B 标签扫描



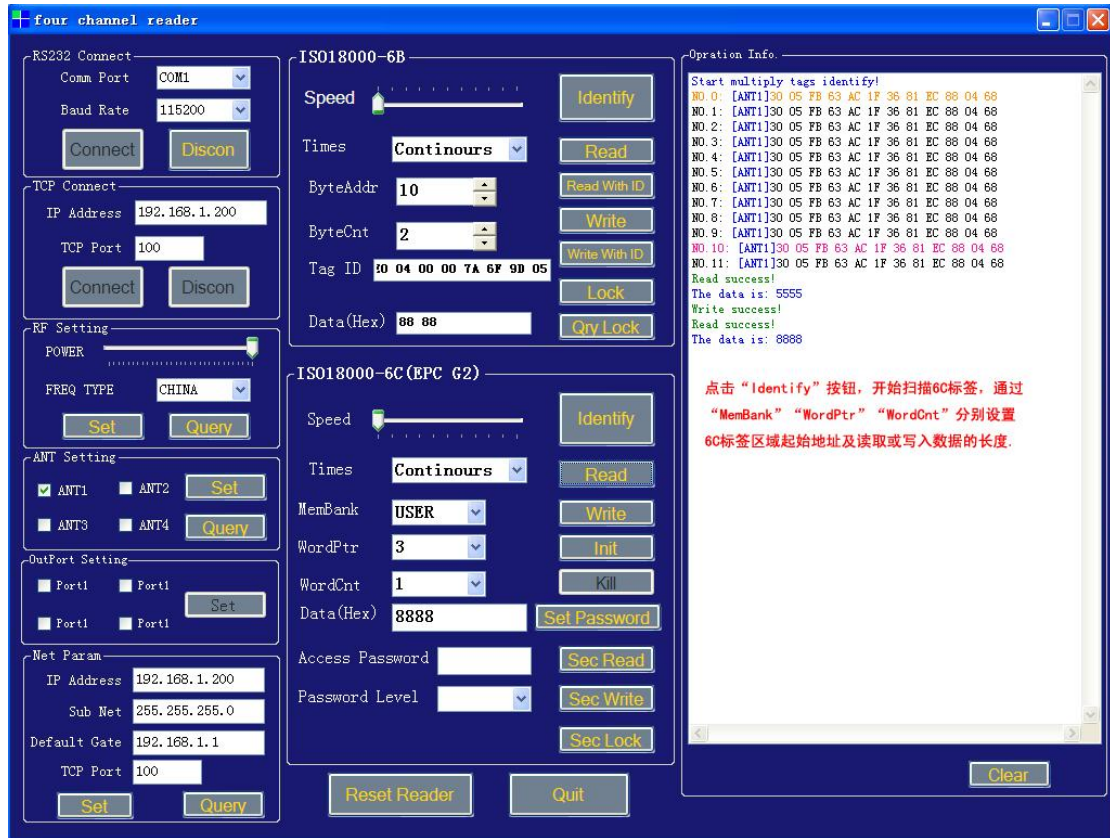


图 3-3

【SecLock】是给标签设置 Access Pwd 后，对 EPC 标签进行安全锁定;

【SecWrite】标签锁定级别为 10 后，标签只能勾选“Secured Password”，并输入正确的访问密码，才能写入数据;如图 3-4;

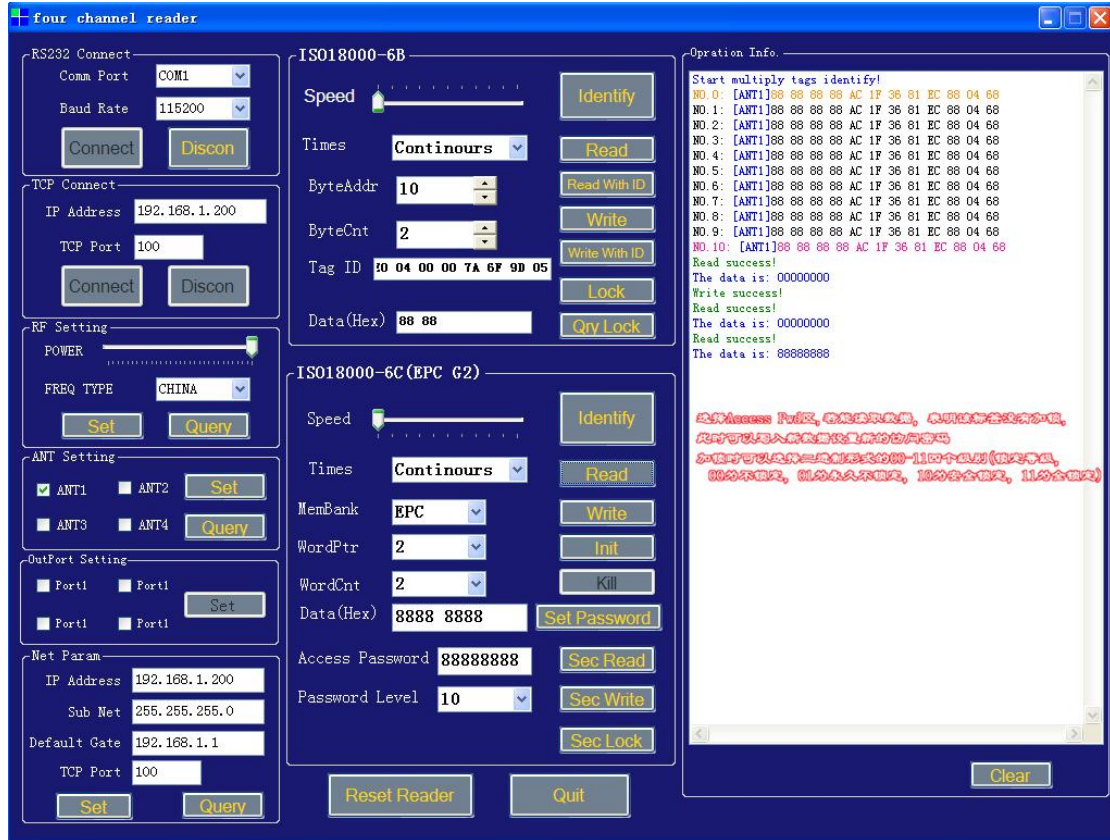


图 3-4