目录

1. DeviceAPI.dll 的 API 说明

1.1 WIFI

1.1.1 WifiOn()

函数原型	BOOL WifiOn()
函数描述	加载 WIFI 驱动
参数	空
返回值	TRUE：WIFI 驱动加载成功
	FALSE：WIFI 驱动加载失败

1.1.2 WifiOff()

函数原型	BOOL WifiOff()
函数描述	卸载 WIFI 驱动
参数	空
返回值	TRUE：WIFI 驱动卸载成功
	FALSE：WIFI 驱动卸载失败

1.1.3 WifiDriver_whether_loaded()

函数原型	BOOL WifiDriver_whether_loaded()
函数描述	判断是否已经加载了 WIFI 驱动
参数	空
返回值	TRUE：已加载 WIFI 驱动
	FALSE：WIFI 驱动没有加载

1.1.4 WifiGet_signal()

函数原型	int WifiGet_signal()
函数描述	获取 WIFI 信号值
参数	空
返回值	1：无连接
	-100 到 0：信号值从-100 到 0 递增，值越大信号越强

1.2 GPRS

GPRS 部分开发注意事项：

⑴先对 GPRS 模块成功上电后再打开串口。

⑵先关闭串口再对 GPRS 模块下电。

1.2.1 GPRSPowerOn()

函数原型	BOOL GPRSPowerOn ()
函数描述	对 GPRS 模块上电
参数	空
返回值	TRUE：GPRS 模块上电成功
	FALSE：GPRS 模块上电失败

1.2.2 GPRSPowerOff()

函数原型	BOOL GPRSPowerOff()
函数描述	对 GPRS 模块下电
参数	空
返回值	TRUE：GPRS 模块下电成功
	FALSE：GPRS 模块下电失败

1.2.3 GPRSReset()

根据 GPRS 厂家的意见，不建议使用 GPRSReset 来复位 GPRS 模块，因为对 GPRS 模块可能会导致 GPRS 模块的程序混乱。

函数原型	BOOL GPRSReset()
函数描述	对 GPRS 模块复位
参数	空
返回值	TRUE：GPRS 模块复位成功
	FALSE：GPRS 模块复位失败

1.2.4 GPRS_judge_connection_status()

函数原型	BOOL GPRS_judge_connection_status(LPCTSTR GprsName)
函数描述	判断名为 GprsName 的 GPRS 连接的连接状态
参数	输入：GprsName，GPRS 拨号连接的名称
返回值	TRUE：名为 GprsName 的 GPRS 处于连接的状态
	FALSE：名为 GprsName 的 GPRS 处于断开的状态

1.2.5 GPRS_close_connection()

函数原型	void GPRS_close_connection(LPCTSTR GprsName)

函数描述	断开名为 GprsName 的 GPRS 连接
参数	输入：GprsName，GPRS 拨号连接的名称
返回值	TRUE：成功断开名为 GprsName 的 GPRS 连接
	FALSE：名为 GprsName 的 GPRS 断开失败

1.2.6 GPRS_judge_modem_status()

函数原型	BOOL GPRS_judge_modem_status ()
函数描述	判断 GPRS 模块是否已经上电
参数	空
返回值	TRUE：GPRS 模块已经上电
	FALSE：GPRS 模块还没有上电

1.2.7 GPRS_open_serial()

函数原型	UINT8 GPRS_open_serial()
函数描述	打开 GPRS 模块对应的串口
参数	空
返回值	0：打开串口失败
	1：打开串口成功
	2：串口已经打开

1.2.8 GPRS_close_serial()

函数原型	BOOL GPRS_close_serial()
函数描述	关闭 GPRS 模块对应的串口
参数	空
返回值	TRUE： 关闭串口成功
	FALSE： 关闭串口失败

1.2.9 GPRS_send_data()

函数原型	DWORD GPRS_send_data(UINT8* SendBuffer,DWORD dwLength)
函数描述	GPRS 发送数据
参数	输入：SendBuffer，发送数据的缓冲区指针
	输入：dwLength，发送数据的长度
返回值	实际发送的数据长度

1.2.10 GPRS_receive_data()

函数原型	DWORDGPRS_receive_data(UINT8*ReceiveBuffer,DWORD dwLength)
函数描述	GPRS 接收数据
参数	输出：ReceiveBuffer，接收数据的缓冲区指针
	输入：dwLength，要接收数据的长度
返回值	实际接收的数据长度

1.3 CDMA

1.3.1 CDMA_power_on()

函数原型	BOOL CDMA_power_on ()
函数描述	对 CDMA 模块上电
参数	空
返回值	TRUE：CDMA 模块上电成功
	FALSE：CDMA 模块上电失败

1.3.2 CDMA_power_off()

函数原型	BOOL CDMA_power_off()
函数描述	对 CDMA 模块下电
参数	空
返回值	TRUE： CDMA 模块下电成功
	FALSE： CDMA 模块下电失败

1.3.3 CDMA_reset()

函数原型	BOOL CDMA_reset()
函数描述	对 CDMA 模块复位
参数	空
返回值	TRUE： CDMA 模块复位成功
	FALSE： CDMA 模块复位失败

1.3.4 CDMA_online_data_to_command_status()

函数原型	BOOL CDMA_online_data_to_command_status()
函数描述	CDMA 模块从 online data 状态切换为 command status，在断开 CDMA 连接后，需要执行这个动作，否则下次不能正常拨号
参数	空
返回值	TRUE： CDMA 模块状态切换成功
	FALSE： CDMA 模块状态切换失败

1.3.5 CDMA_sync_dialing()

函数原型	BOOL CDMA_sync_dialing()
函数描述	对名为 CDMA 的连接进行拨号
参数	空
返回值	TRUE： 拨号成功
	FALSE： 拨号失败

1.4 GPS

1.4.1 Gps_init()

函数原型	void Gps_init()
函数描述	GPS 模块上电
参数	空
返回值	无

1.4.2 Gps_free()

函数原型	void Gps_free()
函数描述	GPS 模块断电
参数	空
返回值	无

1.4.3 BDS_init()

函数原型	BOOL BDS_init()
函数描述	北斗 GPS 模块上电
参数	空
返回值	TRUE：北斗 GPS 模块上电成功
	FALSE：北斗 GPS 模块上电失败

1.4.4 BDS_free()

函数原型	BOOL BDS_free()
函数描述	北斗 GPS 模块断电
参数	空
返回值	TRUE： 北斗 GPS 模块断电成功
	FALSE： 北斗 GPS 模块断电失败

1.4.5 Gps_open_serial()

函数原型	HANDLE Gps_open_serial()
函数描述	打开 GPS 串口
参数	空
返回值	返回 GPS 串口的句柄

1.4.6 Gps_close_serial()

函数原型	BOOL Gps_close_serial ()
函数描述	关闭 GPS 串口
参数	空
返回值	TRUE：成功关闭 GPS 串口
	FALSE：关闭 GPS 串口失败

1.4.7 Gps_send_data()

函数原型	DWORD Gps_send_data(UINT8* SendBuffer,DWORD dwLength)
函数描述	GPS 发送数据
参数	输入：SendBuffer，发送数据的缓冲区指针
	输入：dwLength，发送数据的长度
返回值	实际发送的数据长度

1.4.8 Gps_receive_data()

函数原型	DWORD Gps_receive_data(UINT8* ReceiveBuffer,DWORD dwLength)
函数描述	GPS 接收数据
参数	输出：ReceiveBuffer，接收数据的缓冲区指针

	输入：dwLength，接收数据的长度
返回值	实际接收的数据长度

1.5 1D

1.5.1 Barcode1D_init()

函数原型	BOOL Barcode1D_init()
函数描述	1D 模块上电
参数	空
返回值	TRUE：上电成功
	FALSE：上电失败

1.5.2 Barcode1D_free()

函数原型	void Barcode1D_free ()
函数描述	1D 模块下电
参数	空
返回值	无

1.5.3 Barcode1D_scan()

函数原型	int Barcode1D_scan(UINT8 *pszData)
函数描述	1D 模块扫描并获取条码
参数	输出：pszData，返回保存扫描条码的指针
返回值	返回条码长度

1.6 2D 硬解码

1.6.1 Barcode2D_init()

函数原型	BOOL Barcode2D_init()
函数描述	2D 硬解码模块上电
参数	空
返回值	TRUE：上电成功
	FALSE：上电失败

1.6.2 Barcode2D_free()

函数原型	BOOL Barcode2D_ free()
函数描述	2D 硬解码模块下电
参数	空
返回值	TRUE：下电成功
	FALSE：下电失败

1.6.3 Barcode2D_scan()

函数原型	BOOL Barcode2D_scan (UINT8 *pszData, int iBufferlen)
函数描述	2D 硬解码模块扫描并获取条码。
参数	输出：pszData，返回保存扫描条码的指针

	输入：iBufferlen，预扫描条码的长度
返回值	条码的实际长度

1.7 2D 软解码

1.7.1 SoftDecoding_Init()

函数原型	BOOL SoftDecoding_Init ()
函数描述	初始化二维软解码
参数	空
返回值	TRUE：初始化成功
	FALSE：初始化失败

1.7.2 SoftDecoding_Deinit()

函数原型	BOOL SoftDecoding_ Deinit()
函数描述	关闭二维软解码时用来释放资源
参数	空
返回值	TRUE：释放成功
	FALSE：释放失败

1.7.3 SoftDecoding_Select_ScanMode()

函数原型	BOOL SoftDecoding_ Select_ScanMode()
函数描述	设置二维头为解码模式，在 SoftDecoding_Scan 之前调用，连续扫描时只要调用一遍就可以了
参数	空
返回值	TRUE：设置成功
	FALSE：设置失败

1.7.4 SoftDecoding_Scan()

函数原型	BOOL SoftDecoding_Scan(UINT nTimeout, UCHAR *barcodeData, UINT bufSize, UINT *ReadDataSize)
函数描述	扫描条码并解码
参数	输入：nTimeout，扫描超时时间
	输入：barcodeData，条码数据存储区地址
	输入：bufSize，条码存储区大小,根据条码数据量设置大小，默认推荐 2048 字节
	输出: ReadDataSize， 读到的条码数据字节个数
返回值	TRUE：解码成功
	FALSE：解码失败

1.7.5 SoftDecoding_Select_SnapshotMode()

函数原型	BOOL SoftDecoding_Select_SnapshotMode()
函数描述	设置二维头为拍照模式，在 SoftDecoding_Snapshot 前调用，拍完照后要回到解码模式必须调用 SoftDecoding_Select_ScanMode

参数	空
返回值	TRUE：设置成功
	FALSE：设置失败

1.7.6 SoftDecoding_Snapshot()

函数原型	BOOL SoftDecoding_Snapshot(UINT nTimeout)
函数描述	拍照，拍完会在 wince 根目录生成 sdl.jpg 文件
参数	输入：nTimeout，超时时间
返回值	TRUE：拍照成功
	FALSE：拍照失败

1.7.7 SoftDecoding_BarcodeType_OnOff()

函数原型	BOOL SoftDecoding_BarcodeType_OnOff(UCHAR* parameter, UINT parameterSize)
函数描述	使能和关闭各类型的条码
参数	输入：parameter，指向要控制的条码类型
	输入：parameterSize，要控制的条码类型长度
返回值	TRUE：设置成功
	FALSE：设置失败

1.8 PSAM

注：1.8.1-18.6 小节内这部分函数针对 C5000 机器

1.8.1 Psam_init_Ex()

函数原型	BOOL Psam_init_Ex()
函数描述	初始化 PSAM 模块
参数	空
返回值	TRUE：初始化成功
	FALSE：初始化失败

1.8.2 Psam_free_Ex()

函数原型	BOOL Psam_free_Ex ()
函数描述	释放 PSAM 资源
参数	空
返回值	TRUE：释放成功
	FALSE：释放失败

1.8.3 Psam_powerOn_Ex()

函数原型	int Psam_powerOn_Ex(int iSlot, int iPowerUp, int iBaudrate, UINT8 *pszData)	
函数描述	设置或是选择 PSAM 卡槽，电压，波特率	
参数	输入	iSlot，对于 C5000 V2.0 主板(卡槽是白色)：值 1 对应设备的卡槽 2,值 2 对应设备的卡槽 1；对于 C5000 V6.0 及后面的主板(卡槽是

		黑色)：值 1 对应设备的卡槽 1,值 2 对应设备的卡槽 2。
		iPowerUp，0 - 5v 供电；1 - 3v 供电；2 - 1.8v 供电
		iBaudrate，取值为 0 到 6，0 - 9600；1 -19200；2 - 38400；3 - 43000；4 - 56000；5 - 57600；6 - 115200；
	输出	pszData，保存 PSAM 卡返回的复位信息
返回值	TRUE：设置或是选择成功	
	FALSE：设置或是选择失败	

1.8.4 Psam_powerOff_Ex()

函数原型	BOOL Psam_powerOff_Ex()
函数描述	对 PSAM 模块断电
参数	空
返回值	TRUE：断电成功
	FALSE：断电失败

1.8.5 Psam_select_Ex()

函数原型	int Psam_select_Ex(int iSlot)
函数描述	选择 PSAM 卡的卡槽
参数	输入：iSlot，对于 C5000 V2.0 主板(卡槽是白色)：值 1 对应设备的卡槽 2,值 2 对应设备的卡槽 1；对于 C5000 V6.0 及后面的主板(卡槽是黑色)：值 1 对应设备的卡槽 1,值 2 对应设备的卡槽 2。
返回值	TRUE：选择成功
	FALSE：选择失败

1.8.6 Psam_command_Ex()

函数原型	int Psam_command_Ex(UINT8 *pszCOS, int iLenCOS, UINT8 *pszData)	
函数描述	双 PSAM 模块发送 COS 指令	
参数		pszCOS，指令内容
	输入	iLenCOS，指令长度
	输出	pszData，接收 PSAM 卡反馈的数据的数组
返回值	TRUE：发送成功	
	FALSE：发送失败	

1.8.7 Psam_IO_PowerOn()

备注：此 API 用于 C2000 RFID 板中 PSAM 卡的上电，如果之前还没有对 RFID 板上电，就需要先调用 RF_ISO14443A_init()对 RFID 板上电，再调用 Psam_IO_PowerOn()才能对 PSAM 卡成功上电。

函数原型	int Psam_IO_PowerOn()
函数描述	PSAM 卡上电和初始化
参数	无
返回值	1：上电成功
	0：上电失败

1.8.8 Psam_IO_PowerOff()

备注：此 API 用于 C2000 RFID 板中 PSAM 模块的下电，在调用此 API 之后，如要对 RFID 板下电，还要调用 RF_ISO14443A_free()对 RFID 板下电。

函数原型	int Psam_IO_PowerOff()	
函数描述	PSAM 卡下电	
参数	无	
返回值	1：下电成功	
	0：下电失败	

1.8.9 Psam_IO_GetRandom()

备注：此 API 用于 C2000 RFID 板中 PSAM 模块获取随机数

函数原型	int Psam_IO_GetRandom(UINT8 type, UINT8 *recvBuf, UINT8 *recvLen)	
函数描述	获取随机数命令	
参数	输入	type，随机数长度，4 或者 8
	输出	recvBuf，接受存放随机数的数组
		recvLen，随机数字节长度(1 个字节)
返回值	1：获取随机数成功	
	0：获取随机数失败	

1.8.10 Psam_IO_CmdSendAndRecv()

备注：用于 C2000 RFID 板中 PSAM 模块透传

函数原型	intPsam_IO_CmdSendAndRecv(UINT8*sendBuf,UINT8sendLen,UINT8 *recvBuf, UINT8 *recvLen)	
函数描述	发送命令和接受 PSAM 卡返回值	
参数	输入	sendBuf,指令内容
		sendLen，指令长度
	输出	recvBuf，接收 PSAM 卡返回的数据
		recvLen,PSAM 卡返回数据的长度（1 个字节）
返回值	1：命令交换成功	
	0：命令交换失败	

1.9 EM125K

1.9.1 EM125k_init()

函数原型	bool EM125k_init();
函数描述	对低频 RFID 读卡模块初始化
参数	无
返回值	TURE 成功
	FAULE 失败

1.9.2 EM125k_free()

函数原型	void EM125k_free();
函数描述	释放低频 RFID 读卡资源
参数	无

返回值	无

1.9.3 EM125k_OpenAntenna()

函数原型	void EM125k_OpenAntenna()
函数描述	打开低频 RFID 读卡模块天线
参数	无
返回值	无

1.9.4 EM125k_CloseAntenna()

函数原型	void EM125k_CloseAntenna()
函数描述	关闭低频 RFID 读卡模块天线
参数	无
返回值	无

1.9.5 EM125k_read()

函数原型	int EM125k_read(int iMode, UINT8 *pszData);
函数描述	读取不同类型的标签数据
参数	输入：iMode 0 为 ID 卡；1 为全双工动物标签 ；2 为半双工动物 标签
	输出：pszData 标签信息，建议 50 字节
返回值	0 ： 成功
	其他： 失败

1.9.6 EM125k_UID_REQ()

函数原型	int EM125k_UID_REQ(int iMode, UINT8 *pszData);
函数描述	获取 hitag S 标签的 UID
参数	输入：iMode 参数没有调用，可以不赋值
	输出：pszData UID 信息，建议 50 字节
返回值	0 ： 成功
	其他： 失败

1.9.7 EM125k_ReadHitag()

函数原型	int EM125k_ReadHitag(UCHAR nPage, UCHAR *pszData);
函数描述	读取 hitag S 标签中某一页的数据
参数	输入：nPage 页码
	输出：pszData 此页信息，pszData[0]为长度，数据从 pszData[1]开始 建议 50 字节
返回值	0 ： 成功
	其他： 失败

1.9.8 EM125k_WriteHitagPage()

函数原型	int EM125k_WriteHitagPage(UCHAR nPage, UCHAR *pszData);
函数描述	写 Hitag S 卡中某一页的数据 （4 字节）

参数	输入	nPage0 页码
		pszData 要写入的 4 字节数据
返回值	0 ： 成功	
	其他： 失败	

1.9.9 EM125k_WriteHitagBlock()

函数原型	int EM125k_WriteHitagBlock(UCHAR nBlock, UCHAR *pszData);
函数描述	写 Hitag S 卡中的某一块数据，一块由 4 页组成
参数	输入：nBlock 块号
	输入：pszData 要写入的 16 字节数据
返回值	0 成功
	其他 失败

1.9.10 EM125k_Read4305()

函数原型	int EM125k_Read4305(UCHAR nPage, UCHAR *pszData);
函数描述	读 4305 卡的某一页（0~31）数据
参数	输入：nPage 页码
	输出：pszData 读出的 4 字节数据
返回值	0 ： 成功
	其他： 失败

1.9.11 EM125k_Write4305()

函数原型	int EM125k_Write4305(UCHAR nPage, UCHAR *pszData);
函数描述	写 4305 卡的某一页（0~31）数据 第 0、3、5~13 页可读可写，可供用户存储数据使用 第 1 页为 UID 存储区，只读 第 4 页为参数配置区域 第 14、15 页为保护区
参数	输入：nPage 页码
	输入：pszData 写入的 4 字节数据
返回值	0 ： 成功
	其他： 失败

1.10 ISO 14443A

1.10.1 RF_ISO14443A_init()

函数原型	bool RF_ISO14443A_init()
函数描述	对 HF 模块上电并打开对应的串口，ISO14443A 和 ISO14443B 都用此 API。
参数	TRUE:初始化成功，FALSE:初始化失败
返回值	TRUE：初始化成功
	FALSE：初始化失败

1.10.2 RF_ISO14443A_free()

函数原型	void RF_ISO14443A_free()
函数描述	对 HF 模块断电及关闭对应的串口，ISO14443A 和 ISO14443B 都用此 API
参数	无
返回值	无

1.10.3 RF_OpenAntenna()

函数原型	void RF_OpenAntenna();
函数描述	打开高频 RFID 模块天线
参数	无
返回值	无

1.10.4 RF_CloseAntenna()

函数原型	void RF_CloseAntenna();
函数描述	关闭高频 RFID 模块天线
参数	无
返回值	无

1.10.5 RF_ISO14443A_request()

备注：请注意此 API 仅适用于 C5000 项目，兼容 C2000、C3000 和 C5000 的需要调用下面的 RF_ISO14443A_request_Ex。所有 RFID 的 API 通过数组返回的参数，数组的第 0 项表明数组后的的数据大小

函数原型	int RF_ISO14443A_request(int iMode, UINT8 *pszATQA);
函数描述	呼叫天线感应区内的电子标签;
参数	输入：iMode 0 为呼叫未进入休眠状态电子标签；1 为呼叫所有状态电子标签
	输出：*pszATQA 标签返回的信息（建议 3 个字节）
返回值	0 ： 成功
	其他： 失败

1.10.6 RF_ISO14443A_request_Ex()

对于寻卡，我们建议只用这个函数即可，不需要分步调用唤醒，防冲突，选择

函数原型	int RF_ISO14443A_request_Ex(int iMode, UINT8 *pszData);
函数描述	呼叫天线感应区内的电子标签;
参数	输入：iMode 0 为呼叫未进入休眠状态电子标签；1 为呼叫所有状态电子标签
	输出：pszData 返回 ATQA（2 字节）＋ UID 长度（1 字节）＋ UID+SAK（UID 最长可达 10 个字节，pszATQA 建议 15 个字节） **注：判断卡片类型请勿根据 ATQA，应根据 SAK,SAK 值与卡类型对应关系** **S50(0x08),S70(0x18),PLUS_2K(0x10),PLUS_4K(0x11),DESFIRE(0x20)** **ULTRALIGHT(0x00)** **以上列举的只是部分，而且存在不同类型卡公用 ATQA 和 SAK 的情况，比如**

	S50 1K 和 PLUS CL2 2K 的 ATQA 都是 04 00，SAK 都是 08，推荐参考 NXP 官方文档 AN10833(MIFARE Type Identification Procedure)
返回值	0 ： 成功
	其他： 失败

1.10.7 RF_ISO14443A_anticoll()

函数原型	int RF_ISO14443A_anticoll(UINT8 *pszUID);	
函数描述	防冲突读 UID	
参数	输出：pszUID（建议 5 个字节）	
返回值	0 ： 成功	
	其他： 失败	

1.10.8 RF_ISO14443A_select()

函数原型	int RF_ISO14443A_select(UINT8 *pszUID,int iLenUID,UINT8 *pszSAK);	
函数描述	选择某一电子标签	
参数	输入：	*pszUID 标签 UID
		iLenUID 标签 UID 长度
	输出：pszSAK（2 字节）	
返回值	0 ： 成功	
	其他： 失败	

1.10.9 RF_ISO14443A_halt()

函数原型	int RF_ISO14443A_halt();
函数描述	标签休眠
参数	无
返回值	0 ： 成功
	其他： 失败

1.10.10 RF_ISO14443A_authentication()

函数原型	int RF_ISO14443A_authentication(int iMode,int iBlock, UINT8 *pszKey,int iLenKey);	
函数描述	认证卡片密钥	
参数	输入	int iMode 密钥验证模式 0 为验证 A 密钥；1 为验证 B 密钥；
		int iBlock 要验证密钥的绝对块号 块号范围 0~63；
		UINT8 *pszKey 密钥内容；（6 个字节）
		int iLenKey 密钥长度 最大长度为 6 字节；
返回值	0 ：	成功
	其他：	失败

1.10.11 RF_ISO14443A_read()

函数原型	int RF_ISO14443A_read(int iBlock, UINT8 *pszData);

函数描述	读取电子标签内容	
参数	输入	iBlock 要读取数据的绝对块号 块号范围 0~63；
	输出	*pszData 返回读取到的块数据信息； 一个块最大 16 个字节，建议 pszData 17 字节
返回值	0 ： 成功	
	其他： 失败	

1.10.12 RF_ISO14443A_write()

函数原型	int RF_ISO14443A_write(int iBlock, UINT8 *pszData, int iLenData);	
函数描述	将指定内容写入电子标签中；	
参数	输入	iBlock 要写入数据的绝对块号 块号范围 0~63；
		*pszData 要写入的数据内容； 一个块最大 16 字节，建议 pszData 16 字节
		iLenData 要写入的数据内容的长度；
返回值	0 ： 成功	
	其他： 失败	

1.10.13 RF_ISO14443A_initval()

函数原型	int RF_ISO14443A_initval(int iBlock, int iValue);	
函数描述	电子钱包初始化	
参数	输入	iBlock 要写入数据的绝对块号 块号范围 0~63；
		iValue 初始金额
返回值	0 ： 成功	
	其他： 失败	

1.10.14 RF_ISO14443A_readval()

函数原型	int RF_ISO14443A_readval(int iBlock, UINT8 *pszValue);	
函数描述	读取电子钱包余额	
参数	输入	iBlock 要写入数据的绝对块号 块号范围 0~63；
	输出	*pszValue 余额（4 字节） （pszValue 建议 5 个字节）
返回值	0 ： 成功	
	其他： 失败	

1.10.15 RF_ISO14443A_decrement()

函数原型	int RF_ISO14443A_decrement(int iBlockValue,int iBlockResult, int iValue);	
函数描述	电子钱包扣值；	
参数	输入	iBlockValue 当前金额所在块；
		iBlockResult 扣值后剩余金额保存的块；
		iValue 金额（4 字节）；
返回值	0 ： 成功	
	其他： 失败	

1.10.16 RF_ISO14443A_increment()

函数原型	int RF_ISO14443A_increment(int iBlockValue, int iBlockResult, int iValue);	
函数描述	电子钱包充值	
参数	输入	iBlockValue 当前金额所在块；
		iBlockResult 充值后剩余金额保存的块；
		iValue 金额（4 字节）；
返回值	0 ： 成功	
	其他： 失败	

1.10.17 RF_ISO14443A_restore()

函数原型	int RF_ISO14443A_restore(int iBlock);
函数描述	将 EEPROM 中的内容传入卡的内部寄存器；
参数	输入：int iBlock 值所在块；
返回值	0 ： 成功
	其他： 失败

1.10.18 RF_ISO14443A_transfer()

函数原型	int RF_ISO14443A_transfer(int iBlock);
函数描述	将寄存器的内容传送到 EEPROM 中；
参数	输入：int iBlock 值所在块；
返回值	0 ： 成功
	其他： 失败

1.10.19 RF_ISO14443A_ul_anticoll()

函数原型	int RF_ISO14443A_ul_anticoll(UINT8 *pszUID);
函数描述	Ultra light 防冲突读 UID
参数	输入：*pszUID 电子标签 UID 信息（7 字节）； pszUID 建议 8 字节
返回值	0 ： 成功
	其他： 失败

1.10.20 RF_ISO14443A_ul_write()

函数原型	int RF_ISO14443A_ul_write(int iBlock, UINT8 *pszData, int iLenData);	
函数描述	将指定内容写入电子标签	
参数	输入	IBlock 写入数据的绝对块号 （0-3 块不能写入数据）；
		*pszData 写入的数据信息；（建议 4 个字节）
		iLenData 写入数据信息的长度（每页 4 字节）；
返回值	0 ： 成功	
	其他： 失败	

1.10.21 RF_ISO14443A_cpu_rats()

函数原型	int RF_ISO14443A_cpu_rats(UINT8 *pszData);
函数描述	CPU 卡 RATS 操作指令（TYPE A 类型）
参数	输出： *pszData 返回信息； 返回的因卡片不同而不同，建议 50 个字节
返回值	0 ： 成功
	其他： 失败

1.10.22 RF_ISO14443A_cpu_reset()

函数原型	int RF_ISO14443A_cpu_reset(UINT8 *pszData);
函数描述	CPU 卡复位操作指令（TYPE A 类型）
参数	输出： *pszData 返回信息； 返回的因卡片不同而不同，建议 50 个字节
返回值	0 ： 成功
	其他： 失败

1.10.23 RF_FM1216_OnekeyReset()

函数原型	int RF_FM1216_OnekeyReset(UINT8 *pszData);
函数描述	FM1216 卡复位操作指令
参数	输出： *pszData 返回信息；
返回值	0 ： 成功
	其他： 失败

1.10.24 RF_ISO14443A_cpu_command()

在对 CPU 命令协议不熟悉的情况下不推荐使用该透传函数

函数原型	int RF_ISO14443A_cpu_command(UINT8 *pszCOS, int iLenCOS, UINT8 *pszData);	
函数描述	CPU 卡 T=CL 发送 COS 指令	
参数	输入：	UINT8 *pszCOS 传输的 COS 指令内容；
		int iLenCOS 传输的 COS 指令内容长度；
	输出：	UINT8 *pszData 返回的数据信息； pszData 建议 128 字节
返回值	0 ： 成功	
	其他： 失败	

1.10.25 RF_ModeSwitch()

函数原型	int RF_ModeSwitch(int iMode)	
函数描述	RFID 模式切换（需执行该命令之后才可进行相应卡操作，默认 ISO14443A 模式）	
参数	输入	iMode：0 表示 ISO14443A；1 表示 ISO14443B；2 表示 ISO15693
返回值	0：成功	
	其他：失败	

1.10.26 RF_ISO14443A_DESFIRE_Cpysel ()

函数原型	void RF_ISO14443A_DESFIRE_Cpysel(unsigned char cpyType)	
函数描述	DESFIRE 卡加解密方式选择	
参数	输入	cpyType：0x01 表示 AES 加密；0x02 表示 DES 加密，默认是 DES；
返回值	无	

1.10.27 RF_ISO14443A_DESFIRE_SelApp()

注：DESFire 操作函数只适用于 RC663 双协议机器，下面的 DESFire 函数同理

函数原型	int RF_ISO14443A_DESFIRE_SelApp(unsigned char *AppId)	
函数描述	Desfire 卡 选择应用	
参数	输入	AppId 应用 ID（3 字节）
返回值	0：成功	
	其他：失败	

1.10.28 RF_ISO14443A_DESFIRE_GetApps ()

函数原型	int RF_ISO14443A_DESFIRE_GetApps(unsigned char *AppNums,unsigned char *AppIDs)	
函数描述	获取 DESFire 卡内所有应用	
参数	输入	无
	输出	AppNums:获取到的应用数（1 字节） AppIDs:存放 ID 的缓存（3*AppNums 个字节）
返回值	0：成功	
	其他：失败	

1.10.29 RF_ISO14443A_DESFIRE_DelApp ()

函数原型	int RF_ISO14443A_DESFIRE_DelApp(unsigned char *AppId)	
函数描述	Desfire 卡 删除应用	
参数	输入	AppId 应用 ID（3 字节）
返回值	0：成功	
	其他：失败	

1.10.30 RF_ISO14443A_DESFIRE_AddApp ()

函数原型	int RF_ISO14443A_DESFIRE_AddApp(unsigned char *AppId,unsigned char KeySetting,unsigned char FileNums)	
函数描述	Desfire 卡 创建应用（最多 28 个应用，出去应用 0，可用 1~27 个）	
参数	输入	AppId 应用 ID（3 字节） FileNums 应用内最大文件数目 KeySetting 安全属性设置
	输出	无
返回值	0：成功	

	其他：失败

1.10.31 RF_ISO14443A_DESFIRE_GetFileIds ()

函数原型	int RF_ISO14443A_DESFIRE_GetFileIds(unsigned char *FileNums, unsigned char *FileBuf)	
函数描述	Desfire 卡 获取应用内所有文件号	
参数	输入	无
	输出	FileNums 共获取到的文件号数目，1 字节指针形式返回 FileBuf 存放文件号的缓存数组，一个文件号占 1 字节
返回值	0：成功	
	其他：失败	

1.10.32 RF_ISO14443A_DESFIRE_GetPiccInfo ()

函数原型	int RF_ISO14443A_DESFIRE_GetPiccInfo(unsigned char *PiccInfo)	
函数描述	Desfire 卡 获取卡片属性	
参数	输入	无
	输出	PiccInfo 卡片属性的存放缓存 卡片属性包括卡号，设备商信息
返回值	0：成功	
	其他：失败	

1.10.33 RF_ISO14443A_DESFIRE_AddStdFile ()

函数原型	int RF_ISO14443A_DESFIRE_AddStdFile(unsigned char FileNo,unsigned char CommSet,unsigned char *AccessRight,unsigned int FileSize);	
函数描述	Desfire 卡 创建标准数据文件	
参数	输入	FileNo 文件号 1 字节 CommSet 通讯设置（设置通讯加密方式） AccessRight 存取权限 FileSize 文件大小
	输出	无
返回值	0：成功	
	其他：失败	

1.10.34 RF_ISO14443A_DESFIRE_DelFile ()

函数原型	int RF_ISO14443A_DESFIRE_DelFile(unsigned char FileNo)	
函数描述	Desfire 卡 删除文件（标准数据文件和值文件都可以删除）	
参数	输入	FileNo 文件号
	输出	无
返回值	0：成功	
	其他：失败	

1.10.35 RF_ISO14443A_DESFIRE_GetFileSetting ()

函数原型	int RF_ISO14443A_DESFIRE_GetFileSetting(unsigned char FileNo, unsigned char *FileAttriBuf)	
函数描述	Desfire 卡 获取文件属性	
参数	输入	FileNo 文件号
	输出	FileAttriBuf 文件属性缓存
返回值	0：成功	
	其他：失败	

1.10.36 RF_ISO14443A_DESFIRE_ChangeFileSetting ()

函数原型	int RF_ISO14443A_DESFIRE_ChangeFileSetting(unsigned char FileNo, unsigned char CommSet,unsigned char *AccessRights)	
函数描述	Desfire 卡 修改文件属性	
参数	输入	FileNo 文件号 CommSet 通讯加密设置 AccessRights 存取权限
	输出	无
返回值	0：成功	
	其他：失败	

1.10.37 RF_ISO14443A_DESFIRE_Auth ()

函数原型	int RF_ISO14443A_DESFIRE_Auth(unsigned char KeyNo,unsigned char *KeyBuf,unsigned char KeyLen)	
函数描述	Desfire 卡 验证秘钥	
参数	输入	KeyNo 秘钥号 KeyBuf 秘钥缓存 KeyLen 秘钥长度
	输出	无
返回值	0：成功	
	其他：失败	

1.10.38 RF_ISO14443A_DESFIRE_ChangeKey ()

函数原型	int RF_ISO14443A_DESFIRE_ChangeKey(unsigned char KeyNo,unsigned char *KeyBuf,unsigned char KeyLen)	
函数描述	Desfire 卡 更改秘钥	
参数	输入	KeyNo 秘钥号 KeyBuf 新秘钥缓存 KeyLen 新秘钥长度
	输出	无
返回值	0：成功	
	其他：失败	

1.10.39 RF_ISO14443A_DESFIRE_GetKeySetting ()

函数原型	int RF_ISO14443A_DESFIRE_GetKeySetting(unsigned char *KeySetting, unsigned char *KeyNums);	
函数描述	Desfire 卡 获取秘钥设置	
参数	输入	无
	输出	KeySetting 安全设置缓存 1 字节 KeyNums 秘钥数目 1 字节
返回值	0：成功	
	其他：失败	

1.10.40 RF_ISO14443A_DESFIRE_ChangeKeySetting ()

函数原型	int RF_ISO14443A_DESFIRE_ChangeKeySetting(unsigned char KeySetting)	
函数描述	Desfire 卡 更改秘钥设置	
参数	输入	KeySetting 需要修改的的秘钥设置
返回值	0：成功	
	其他：失败	

1.10.41 RF_ISO14443A_DESFIRE_WriteStdFile ()

函数原型	int RF_ISO14443A_DESFIRE_WriteStdFile(unsigned char FileNo,unsigned int OffSet,unsigned int DataSize,unsigned char *DataBuf)	
函数描述	Desfire 卡 写标准数据集文件	
参数	输入	FileNo 文件号 OffSet 起始偏移位置 DataSize 写入数据大小 DataBuf 数据缓冲
	输出	无
返回值	0：成功	
	其他：失败	

1.10.42 RF_ISO14443A_DESFIRE_ReadStdFile ()

函数原型	int RF_ISO14443A_DESFIRE_ReadStdFile(unsigned char FileNo,unsigned int OffSet,unsigned int DataSize,unsigned char *DataBuf)	
函数描述	Desfire 卡读标准数据文件	
参数	输入	FileNo 文件号 OffSet 起始偏移位置 DataSize 读取数据大小
	输出	DataBuf 数据缓冲
返回值	0：成功	
	其他：失败	

1.10.43 RF_ISO14443A_DESFIRE_AddValueFile ()

函数原型	int RF_ISO14443A_DESFIRE_AddValueFile(unsigned char FileNo,unsigned char CommSet,unsigned char *AccessRights,int MinValue,

	int MaxValue,int InitValue)	
函数描述	Desfire 卡创建值文件	
参数	输入：	FileNo 文件号 CommSet 通讯加密设置 AccessRights 存取权限 MinValue 最小值 MaxValue 最大值 InitValue 初始值
	输出	无
返回值	0 ： 成功	
	其他： 失败	

1.10.44 RF_ISO14443A_DESFIRE_GetValueFile ()

函数原型	int RF_ISO14443A_DESFIRE_GetValueFile(unsigned char FileNo, int *Value)	
函数描述	Desfire 卡获取值文件数据（获取余额）	
参数	输入	FileNo 文件号
	输出	Value 文件数值
返回值	0 ： 成功	
	其他： 失败	

1.10.45 RF_ISO14443A_DESFIRE_CreditValueFile ()

函数原型	int RF_ISO14443A_DESFIRE_CreditValueFile(unsigned char FileNo, unsigned int CreValue)	
函数描述	Desfire 卡充值函数	
参数	输入	FileNo 文件号 CreValue 充值额（正整数）
	输出	无
返回值	0 ： 成功	
	其他： 失败	

1.10.46 RF_ISO14443A_DESFIRE_DebitValueFile ()

函数原型	int RF_ISO14443A_DESFIRE_DebitValueFile(unsigned char FileNo, unsigned int DeValue)	
函数描述	Desfire 卡扣费函数	
参数	输入	FileNo 文件号 DeValue 扣费额（正整数）
	输出	无
返回值	0 ： 成功	
	其他： 失败	

1.10.47 RF_ISO14443A_DESFIRE_FormatCard ()

函数原型	int RF_ISO14443A_DESFIRE_FormatCard()

函数描述	Desfire 卡格式化卡片	
参数	输入	无
	输出	无
返回值	0 ： 成功	
	其他： 失败	

1.11 ISO 14443B

1.11.1 RF_ISO14443B_cpu_reset()

函数原型	int RF_ISO14443B_cpu_reset(UINT8 *pszData);
函数描述	CPU 卡复位操作指令（TYPE B 类型）
参数	输出：UINT8 *pszData 返回的数据信息； pszData 建议 128 字节
返回值	0 ： 成功
	其他： 失败

1.11.2 RF_ISO14443B_cpu_command()

注：B 卡种类少，而且大都是加密的，应用也不那么广，每种卡内部是不同的，该透传函数不是针对每种卡都有效的，不建议使用，如有相关需求可先咨询相关工程师

函数原型	int RF_ISO14443B_cpu_command(UINT8 *pszCOS, int iLenCOS, UINT8 *pszData);	
函数描述	CPU 卡 T=CL 发送 COS 指令（TYPE B 类型）;	
参数	输入：	UINT8 *pszCOS 传输的 COS 指令内容；
		int iLenCOS 传输的 COS 指令内容长度；
	输出：UINT8 *pszData 返回的数据信息； pszData 建议 128 字节	
返回值	0 ： 成功	
	其他： 失败	

1.11.3 RF_ISO14443B_AT88SC6416_set_user_zone()

函数原型	int RF_ISO14443B_AT88SC6416_set_user_zone(UINT8 Data, UINT8 *pszData);	
函数描述	选择用户区	
参数	输入	Data 用户区号码，从 0 到 15；
	输出	pszData，返回的数据信息； pszData 建议 128 字节
返回值	0 ： 成功	
	其他： 失败	

1.11.4 RF_ISO14443B_AT88SC6416_read_user_zone()

函数原型	int RF_ISO14443B_AT88SC6416_read_user_zone(UINT8 addr_H,UINT8 addr_L, UINT8 number, UINT8 *pszData);	
函数描述	读用户区，每区共 512 字节，地址从 0 开始，注意读取数据不要超出范围！	
参数	输入	addr_H 用户区地址高 8 位

		addr_L 用户区地址低 8 位
		number 读取字节数
	输出	pszData，返回的数据信息； pszData 建议 128 字节
返回值	0 ： 成功	
	其他： 失败	

1.11.5 RF_ISO14443B_AT88SC6416_read_sys_zone()

函数原型	int RF_ISO14443B_AT88SC6416_read_sys_zone(UINT8 addr,UINT8 number, UINT8 *pszData)；	
函数描述	读系统区，系统区共 256 字节，地址从 0 开始，注意读取数据不要超出范围！	
参数	输入	addr 系统区地址
		number 读取字节数
	输出	pszData，返回的数据信息； pszData 建议 128 字节
返回值	0 ： 成功	
	其他： 失败	

1.11.6 RF_ISO14443B_AT88SC6416_write_user_zone()

函数原型	int RF_ISO14443B_AT88SC6416_write_user_zone(UINT8 addr_H,UINT8 addr_L,UINT8 number, UINT8 *pszData)；	
函数描述	写用户区，每区共 512 字节，地址从 0 开始，注意写入数据不要超出范围！	
参数	输入	addr_H 用户区地址高 8 位
		addr_L 用户区地址低 8 位
		number 写入字节数
	输出	pszData，返回的数据信息； pszData 建议 128 字节
返回值	0 ： 成功	
	其他： 失败	

1.11.7 RF_ISO14443B_AT88SC6416_write_sys_zone()

函数原型	int RF_ISO14443B_AT88SC6416_write_sys_zone(UINT8 addr,UINT8 number, UINT8 *pszData)；	
函数描述	写系统区，系统区共 256 字节，地址从 0 开始，注意写入数据不要超出范围！	
参数	输入	addr 系统区地址
		number 写入字节数
	输出	pszData，返回的数据信息； pszData 建议 128 字节
返回值	0 ： 成功	
	其他： 失败	

1.11.8 RF_ISO14443B_AT88SC6416_check_password()

函数原型	int RF_ISO14443B_AT88SC6416_check_password(UINT8 password_index,UINT8

	*password, UINT8 *pszData);		
函数描述	校验 password		
参数	输入	password_index	索引号，从 0 到 7，默认值应该设为 7
		*password	password 内容，共 3 个字节
	输出	pszData，返回的数据信息； pszData 建议 128 字节	
返回值	0 ： 成功		
	其他： 失败		

1.11.9 RF_ISO14443B_AT88SC6416_Verify_Crypto()

函数原型	int RF_ISO14443B_AT88SC6416_Verify_Crypto(UINT8 key_index,UINT8 *pQData, UINT8 *pCHData, UINT8 *pszData);		
函数描述	选择密钥		
参数	输入	key_index	00 Secret Seed G0
			01 Secret Seed G1
			02 Secret Seed G2
			03 Secret Seed G3
			10 Session Encryption Key S0
			11 Session Encryption Key S1
			12 Session Encryption Key S2
			13 Session Encryption Key S3
		*pQData 共 8 字节	
		*pCHData 共 8 字节	
	输出	pszData，返回的数据信息； pszData 建议 128 字节	
返回值	0 ： 成功		
	其他： 失败		

1.11.10 RF_ISO14443B_AT88SC6416_Send_Checksum()

函数原型	int RF_ISO14443B_AT88SC6416_Send_Checksum(UINT8 *pMAC, UINT8 *pszData);		
函数描述	发送用户加密后得到的 2 字节结果		
参数	输入	UINT8 *pMAC 共 2 字节	
	输出	pszData，返回的数据信息； pszData 建议 128 字节	
返回值	0 ： 成功		
	其他： 失败		

1.11.11 RF_ISO14443B_AT88SC6416_debug()

函数原型	int RF_ISO14443B_AT88SC6416_debug(UINT8 data_lenth,UINT8 *pdata_send, UINT8 *pszData);

函数描述	用于副板程序调试	
参数	输入	UINT8 data_lenth　　向副板发送的数据长度，最多 254 字节
		UINT8 *data_send　　向副板发送的数据
	输出	pszData，返回的数据信息； pszData 建议 128 字节
返回值	0 ： 成功	
	其他： 失败	

1.12　ISO15693

1.12.1 RF_ISO15693_init()

函数原型	bool RF_ISO15693_init();
函数描述	将 RFID 模块配置成操作符合 ISO15693 协议的卡的模式
参数	无
返回值	TRUE：配置成功
	FALSE：配置失败

1.12.2 RF_ISO15693_free()

函数原型	bool RF_ISO15693_free();
函数描述	释放模块资源
参数	无
返回值	TRUE：释放成功
	FALSE：释放失败

1.12.3 RF_ISO15693_inventory()

注：在操作卡之前请务必调用该函数以便底层硬件初始化 15693 相关配置

函数原型	int RF_ISO15693_inventory(int iMode, int iAFI, UINT8 *pszData);		
函数描述	读取卡片 UID		
参数	输入	iMode ：	0 多张卡呼叫 不带 AFI
			1 单张卡呼叫 不带 AFI
			2 多张卡呼叫 带 AFI
			3 单张卡呼叫 带 AFI
		iAFI ： AFI 值	
	输出：pszData：UID 内容 UID 为 8 字节，pszData 建议 11 个字节		
返回值	0： 成功		
	其他：失败		

1.12.4 RF_ISO15693_stayQuiet()

函数原型	int RF_ISO15693_stayQuiet(UINT8 *pszUID, int iLenUID);	
函数描述	指定电子标签进入静止状态	
参数	输入	pszUID：UID 内容，建议 8 字节

		iLenUID：UID 长度
返回值	0： 成功	
	其他：失败	

1.12.5 RF_ISO15693_read_sm()

该函数兼容 ICODEII/TI2048 等常规 15693 标签，同时也支持 LRIS64K 大容量标签的支持，区别在于 ICODEII 等 startblock 只是一个整形字节指明起始位置，而 LRIS64K,虽然也是通过 int 类型的 startblock 来传递位置，但是它是有 sector*62+block 组成的，在写应用程序时注意区分。

函数原型	int RF_ISO15693_read_sm(int iMode, UINT8 *pszUID, int iLenUID, int startblock, int blocknum, UINT8 *pszData);		
函数描述	读取卡片内部数据		
参数	输入	iMode：（值范围为 0-7）	0 非 SELECT 状态，不传 UID NXP I CODE SLI 标签
			1 SELECT 状态，不传 UID NXP I CODE SLI 标签
			2 非 SELECT 状态，传 UID NXP I CODE SLI 标签
			3 SELECT 状态，传 UID NXP I CODE SLI 标签
			4 非 SELECT 状态 不传 UID TI 标签
			5 SELECT 状态 不传 UID TI 标签
			6 非 SELECT 状态 传 UID TI 标签
			7 SELECT 状态 传 UID TI 标签
		pszUID：UID	
		iLenUID：UID 长度	
		Startblock：起始块号	
		Blocknum：一共读取多少块（1-10）	
	输出	pszData 返回数据信息 建议一次最多只读 10 个块，每个块有 4 个字节，pszData 建议 41 字节	
返回值	0： 成功		
	其他：失败		

1.12.6 RF_ISO15693_write_sm()

注意事项同 RF_ISO15693_read_sm 函数

函数原型	int RF_ISO15693_write_sm(int iMode, UINT8 *pszUID, int iLenUID, int startblock, int blocknum, UINT8 *pszData, int iWriteLen);		
函数描述	向卡片内部写入数据		
参数	输入	iMode:（值范围为 0-7）	0 非 SELECT 状态，不传 UID，NXP I CODE SLI 标签
			1 SELECT 状态，不传 UID NXP I CODE SLI 标签
			2 非 SELECT 状态，传 UID NXP I CODE SLI 标签
			3 SELECT 状态，传 UID NXP I CODE SLI 标签
			4 非 SELECT 状态 不传 UID TI 标签
			5 SELECT 状态 不传 UID TI 标签
			6 非 SELECT 状态 传 UID TI 标签

			7 SELECT 状态 传 UID TI 标签
		pszUID：UID	
		iLenUID：UID 长度	
		startblock：起始块号	
		blocknum：一共写入多少块（每块 4 字节）	
		pszData 写入的数据信息（建议一次最多 10 个块，40 个字节）	
		iWriteLen：写入的数据长度（4 字节的整数倍）	
返回值	0： 成功		
	其他：失败		

1.12.7 RF_ISO15693_lockBlock()

函数原型	int RF_ISO15693_lockBlock(int iMode, UINT8 *pszUID, int iLenUID, int startblock, int blocknum);		
函数描述	锁定数据块		
参数	输入	iMode:（值范围为 0-7）	0 非 SELECT 状态，不传 UID NXP I CODE SLI 标签
			1 SELECT 状态，不传 UID NXP I CODE SLI 标签
			2 非 SELECT 状态，传 UID NXP I CODE SLI 标签
			3 SELECT 状态，传 UID NXP I CODE SLI 标签
			4 非 SELECT 状态 不传 UID TI 标签
			5 SELECT 状态 不传 UID TI 标签
			6 非 SELECT 状态 传 UID TI 标签
			7 SELECT 状态 传 UID TI 标签
		pszUID：UID	
		iLenUID：UID 长度	
		startblock：起始块号	
		blocknum：一共锁定多少块	
返回值	0： 成功		
	其他：失败		

1.12.8 RF_ISO15693_select()

函数原型	int RF_ISO15693_select(UINT8 *pszUID, int iLenUID);	
函数描述	选择标签	
参数	输入	pszUID：UID
		iLenUID：UID 长度
返回值	0： 成功	
	其他：失败	

1.12.9 RF_ISO15693_resetToReady()

函数原型	int RF_ISO15693_resetToReady(int iMode, UINT8 *pszUID, int iLenUID);		
函数描述	ISO15693 电子标签复位指令 使状态变为 READY 状态		
参数	输入	iMode:（值范围为 0-3）	0 非 SELECT 状态，不传 UID NXP I CODE SLI 标签
			1 SELECT 状态，不传 UID NXP I CODE SLI 标签

			2 非 SELECT 状态，传 UID NXP I CODE SLI 标签
			3 SELECT 状态，传 UID NXP I CODE SLI 标签
		pszUID：UID	
		iLenUID：UID 长度	
返回值	0： 成功		
	其他：失败		

1.12.10 RF_ISO15693_writeAFI()

函数原型	int RF_ISO15693_writeAFI(int iMode, UINT8 *pszUID, int iLenUID, int iAFI);		
函数描述	写 AFI		
参数	输入	iMode:（值范围为 0-7）	0 非 SELECT 状态，不传 UID NXP I CODE SLI 标签
			1 SELECT 状态，不传 UID NXP I CODE SLI 标签
			2 非 SELECT 状态，传 UID NXP I CODE SLI 标签
			3 SELECT 状态，传 UID NXP I CODE SLI 标签
			4 非 SELECT 状态 不传 UID TI 标签
			5 SELECT 状态 不传 UID TI 标签
			6 非 SELECT 状态 传 UID TI 标签
			7 SELECT 状态 传 UID TI 标签
		pszUID：UID	
		iLenUID：UID 长度	
		iAFI：AFI 内容	
返回值	0： 成功		
	其他：失败		

1.12.11 RF_ISO15693_lockAFI()

函数原型	int RF_ISO15693_lockAFI(int iMode, UINT8 *pszUID, int iLenUID);		
函数描述	锁 AFI		
参数	输入	iMode：（值范围为 0-7）	0 非 SELECT 状态，不传 UID NXP I CODE SLI 标签
			1 SELECT 状态，不传 UID NXP I CODE SLI 标签
			2 非 SELECT 状态，传 UID NXP I CODE SLI 标签
			3 SELECT 状态，传 UID NXP I CODE SLI 标签
			4 非 SELECT 状态 不传 UID TI 标签
			5 SELECT 状态 不传 UID TI 标签
			6 非 SELECT 状态 传 UID TI 标签
			7 SELECT 状态 传 UID TI 标签
		pszUID：UID	
		iLenUID：UID 长度	
返回值	0： 成功		
	其他：失败		

1.12.12 RF_ISO15693_writeDSFID()

函数原型	int RF_ISO15693_writeDSFID(int iMode, UINT8 *pszUID, int iLenUID, int iDSFID);		
函数描述	写 DSFID		
参数	输入	iMode：（值范围为 0-7）	0 非 SELECT 状态，不传 UID NXP I CODE SLI 标签
			1 SELECT 状态，不传 UID NXP I CODE SLI 标签
			2 非 SELECT 状态，传 UID NXP I CODE SLI 标签
			3 SELECT 状态，传 UID NXP I CODE SLI 标签
			4 非 SELECT 状态 不传 UID TI 标签
			5 SELECT 状态 不传 UID TI 标签
			6 非 SELECT 状态 传 UID TI 标签
			7 SELECT 状态 传 UID TI 标签
		pszUID：UID	
		iLenUID：UID 长度	
		iDSFID：DSFID 内容	
返回值	0： 成功		
	其他：失败		

1.12.13 RF_ISO15693_lockDSFID()

函数原型	int RF_ISO15693_lockDSFID(int iMode, UINT8 *pszUID, int iLenUID);		
函数描述	锁 DSFID		
参数	输入	iMode：（值范围为 0-7）	0 非 SELECT 状态，不传 UID NXP I CODE SLI 标签
			1 SELECT 状态，不传 UID NXP I CODE SLI 标签
			2 非 SELECT 状态，传 UID NXP I CODE SLI 标签
			3 SELECT 状态，传 UID NXP I CODE SLI 标签
			4 非 SELECT 状态 不传 UID TI 标签
			5 SELECT 状态 不传 UID TI 标签
			6 非 SELECT 状态 传 UID TI 标签
			7 SELECT 状态 传 UID TI 标签
		pszUID：UID	
		iLenUID：UID 长度	
返回值	0： 成功		
	其他：失败		

1.12.14 RF_ISO15693_getSystemInformation()

函数原型	int RF_ISO15693_getSystemInformation(int iMode, UINT8 *pszUID, int iLenUID, UINT8 *pszData);		
函数描述	获取电子标签信息		
参数	输入	iMode：（值范围为 0-3）	0 非 SELECT 状态，不传 UID NXP I CODE SLI 标签
			1 SELECT 状态，不传 UID NXP I CODE SLI 标签
			2 非 SELECT 状态，传 UID NXP I CODE SLI 标签

			3 SELECT 状态，传 UID NXP I CODE SLI 标签
		pszUID：UID	
		iLenUID：UID 长度	
		pszData：标签信息，建议 15 个字节	
返回值	0： 成功		
	其他：失败		

1.12.15 RF_ISO15693_getMultipleBlockSecurityStatus()

函数原型	int RF_ISO15693_getMultipleBlockSecurityStatus(int iMode, UINT8 *pszUID, int iLenUID, int startblock, int blocknum, UINT8 *pszData);		
函数描述	获取电子标签信息		
参数	输入	iMode(值范围为 0-3)	0 非 SELECT 状态，不传 UID NXP I CODE SLI 标签
			1 SELECT 状态，不传 UID NXP I CODE SLI 标签
			2 非 SELECT 状态，传 UID NXP I CODE SLI 标签
			3 SELECT 状态，传 UID NXP I CODE SLI 标签
		pszUID：UID	
		iLenUID：UID 长度	
		Startblock：起始块号	
		blocknum：一共读取块数	
		pszData：标签信息，建议 50 个字节	
返回值	0： 成功		
	其他：失败		

1.13 UHF

1.13.1 UHFFlagCrcOn()

函数原型	void UHFFlagCrcOn ()
函数描述	开启 CRC 校验码（默认为不带校验码）
参数	空
返回值	空

1.13.2 UHFFlafCrcOff()

函数原型	void UHFFlafCrcOff ()
函数描述	关闭 CRC 校验位，默认为不带校验码
参数	空
返回值	空

1.13.3 UHFInit()

函数原型	void UHFInit ()
函数描述	UHF 模块上电
参数	空
返回值	空

1.13.4 UHFFree()

函数原型	void UHFFree ()
函数描述	UHF 下电
参数	空
返回值	空

1.13.5 UHFOpenAndConnect()

函数原型	bool UHFOpenAndConnect(UCHAR* status)
函数描述	连接 UHF
参数	status-- 获取到的 UHF 的工作状态（如果 bit0 等于 0 表示连接成功）
返回值	TRUE：连接成功
	FALSE：连接失败

1.13.6 UHFCloseAndDisconnect()

函数原型	bool UHFCloseAndDisconnect()
函数描述	断开连接
参数	空
返回值	TRUE：操作成功
	FALSE：操作失败

1.13.7 UHFGetPower ()

函数原型	bool UHFGetPower (UCHAR* uPower)
函数描述	获取 UHF 发射功率
参数	uPower——UHF 返回的功率值，范围为（10-30dBm）
返回值	TRUE：操作成功
	FALSE：操作失败

1.13.8 UHFSetPower ()

函数原型	bool UHFSetPower (UCHAR uPower)
函数描述	设置 UHF 发射功率
参数	uPower——设置功率参数（10 <= uPower <= 30）
返回值	TRUE：操作成功
	FALSE：操作失败

1.13.9 UHFGetFrequency()

函数原型	bool UHFGetFrequency(UCHAR* uFreMode, UCHAR* uFreBase, UCHAR* uBaseFre, UCHAR* uChannNum, UCHAR* uChannSpc, UCHAR* uFreHop)	
函数描述	获取 UHF 的工作频率参数设置	
参数	uFreMode	-频率工作模式（0x00-中国标准 920-925MHz、0x01-中国标准 840-845MHz、0x02-ETSI 标准、0x03-定频模式（915MHz）、0x04-用户自定义）

	uFreBase	-频率基数（0x00-50KHz、0x01-125KHZ),当频率基数为 50KHz,带宽不能大于 12MHz，当频率基数为 125KHz 带宽不能大于 32MHz
	uBaseFre-	起始频率（2 字节、bit14~5 为起始频率整数部分，bit3~0 为起始频率的小数部分
	uChanNum	频道数
	uChannSpc-	频道带宽（bit3~0-频道带宽的积数）
	uFreHop	跳频顺序（0x00-随机跳频、 0x01-从高往低方向跳频、 0x10-从低往高顺序跳频、 FALSE-随机跳频）
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.13.10 UHFSetFrequency_EX()

函数原型	bool UHFSetFrequency_EX(UCHAR FreMode)
函数描述	设置 UHF 工作频率默认参数
参数	输入：FreMode， 0x00 -- （中国标准、125KHz 基频、起始频率为 840.625MHz、频道数为 16、频道带宽积数为 2,、随机跳频） 0x01 -- （中国标准、125KHz 基频、起始频率为 920.625MHz、频道数为 16、频道带宽积数为 2,、随机跳频） 0x02 -- （ETSI 标准、50KHz 基频、 起始频率为 865.1MHz、频道数为 12、频道带宽积数为 4、随机跳频） 0x03 -- （顶频模式、125KHz 基频、起始频率 915MHz、频道数为 1、频道积数为 0、随机跳频）
返回值	TRUE：操作成功
	FALSE：操作失败

1.13.11 UHFSetFrequency()

函数原型	bool UHFSetFrequency (UCHAR* uFreMode, UCHAR* uFreBase, UCHAR* uBaseFre, UCHAR* uChannNum, UCHAR* uChannSpc, UCHAR* uFreHop)	
函数描述	设置 UHF 工作频率参数	
参数	uFreMode	-频率工作模式（0x00-中国标准 920-925MHz、0x01-中国标准 840-845MHz、0x02-ETSI 标准、0x03-定频模式（915MHz）、0x04-用户自定义）
	uFreBase	-频率基数（0x00-50KHz、0x01-125KHZ),当频率基数为 50KHz,带宽不能大于 12MHz，当频率基数为 125KHz 带宽不能大于 32MHz
	uBaseFre-	起始频率（2 字节、bit14~5 为起始频率整数部分，bit3~0 为起始频率的小数部分
	uChanNum	频道数
	uChannSpc-	频道带宽（bit3~0-频道带宽的积数）
	uFreHop	跳频顺序（0x00-随机跳频、 0x01-从高往低方向跳频、 0x10-从低往高顺序跳频、 FALSE-随机跳频）

返回值	TRUE：操作成功
	FALSE：操作失败

1.13.12 UHFReadCommand()

函数原型	bool UHFReadCommand (UCHAR* uCmd)
函数描述	获取当前命令
参数	uCmd: 命令代码
返回值	TRUE：操作成功
	FALSE：操作失败

1.13.13 UHFReadStatus()

函数原型	bool UHFReadStatus (UCHAR* uStatus)
函数描述	读取 UHF 工作状态
参数	uStatus：获取到的 UHF 的工作状态（如果 bit0 等于 0 表示连接成功）
返回值	TRUE：操作成功
	FALSE：操作失败

1.13.14 UHFInventory()

函数原型	bool UHFInventory (UCHAR flagAnti, UCHAR initQ)	
函数描述	发送 UHF 循环识别指令	
参数	flagAnti	0x01-使用防碰撞、 0x00-表示不使用防碰撞功能
	initQ	防碰撞个过程使用 Q 值（flagAnti=1 有效）、Q 值范围为 0~15、每次防碰撞需对目标标签连续扫描 2 的 Q 次方,每次防碰撞扫描时间与 Q 值的指数成正比.
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.13.15 UHFGetReceived()

函数原型	bool UHFGetReceived (int* uLenUii, UCHAR* uUii)	
函数描述	接收返回的标签 UII	
参数	uLenUii	uUii 的长度
	uUii	标签 UII 至少 66 字节
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.13.16 UHFStopGet()

函数原型	bool UHFStopGet()
函数描述	结束 UHF 扫描
参数	空
返回值	TRUE：操作成功
	FALSE：操作失败

1.13.17 UHFReadData()

函数原型	bool UHFReadData (UCHAR* uAccessPwd, UCHAR uBank, int uPtr, UCHAR uCnt, UCHAR* uUii, UCHAR* uReadData, UCHAR* uErrorCode)		
函数描述	读取标签数据（指定 UII）		
参数	输入	uAccessPwd	标签的 ACCESSPASSWORD（4 字节）()
		uBank	标签的数据段类型（0x00-RESERVED、0x01-EPC、0x02-TID、0x03-USER
		uPtr	起始地址的偏移量
		uCnt	读出的数据长度（两个字节为单位，不允许为 0）
		uUii	标签的 UII
	输出	uReadData	读取标签的数据（至少为 uCnt*2 字节）
		uErrorCode	标签返回的 Error Code（只有函数返回失败且 uErrorcode 不为 0xff 时有效）
返回值	TRUE：操作成功		
	FALSE：操作失败		

1.13.18 UHFWriteData()

函数原型	bool UHFWriteData (UCHAR* uAccessPwd, UCHAR uBank, int uPtr, UCHAR uCnt, UCHAR* uUii, UCHAR* uWriteData, UCHAR* uErrorCode)		
函数描述	写入标签数据（指定 UII）		
参数	输入	uAccessPwd	标签的 ACCESSPASSWORD（4 字节）()
		uBank	标签的数据段类型（0x00-RESERVED、0x01-EPC、0x02-TID、0x03-USER
		uPtr	起始地址的偏移量
		uCnt	写入的数据长度（两个字节为单位，不允许为 0）
		uUii	标签的 UII
		uWriteData	写入标签的数据(cnt*2 字节)
	输出	uErrorCode	标签返回的 Error Code（只有函数返回失败且 uErrorcode 不为 0xff 时有效）
返回值	TRUE：操作成功		
	FALSE：操作失败		

1.13.19 UHFEraseData()

函数原型	bool UHFEraseData (UCHAR* uAccessPwd, UCHAR uBank, int uPtr, UCHAR uCnt, UCHAR* uUii, UCHAR* uErrorCode)		
函数描述	擦除标签数据		
参数	输入	uAccessPwd	标签的 ACCESSPASSWORD（4 字节）()
		uBank	标签的数据段类型（0x00-RESERVED、0x01-EPC、0x02-TID、0x03-USER
		uPtr	起始地址的偏移量（EBV 格式）
		uCnt	读出的数据长度（两个字节为单位，不允许为 0）
		uUii	标签的 UII
	输	uErrorCode	标签返回的 Error Code（只有函数返回失败且

	出		uErrorcode 不为 0xff 时有效）
返回值	TRUE：操作成功		
	FALSE：操作失败		

1.13.20 UHFLockMem()

函数原型	bool UHFLockMem (UCHAR* uAccessPwd, UCHAR* uLockData, UCHAR* uUii, UCHAR* uErrorCode)		
函数描述	锁定标签指定的数据段		
参数	输入	uAccessPwd	标签的 ACCESSPASSWORD（4 字节）()
		uLockData	标签的数据段类型（0x00-RESERVED、0x01-EPC、0x02-TID、0x03-USER
		uUii	标签的 UII
	输出	uErrorCode	标签返回的 Error Code（只有函数返回失败且 uErrorcode 不为 0xff 时有效）
返回值	TRUE：操作成功;		
	FALSE：操作失败		

1.13.21 UHFKillTag()

函数原型	bool UHFKillTag (UCHAR* uKillPwd, UCHAR* uUii, UCHAR* uErrorCode)		
函数描述	销毁指定标签		
参数	输入	uKillPwd	标签 KillPassWord（4 字节）
		uUii	标签的 UII
	输出	uErrorCode	标签返回的 Error Code（只有函数返回失败且 uErrorcode 不为 0xff 时有效）
返回值	TRUE：操作成功;		
	FALSE：操作失败		

1.13.22 UHFGetVersion()

函数原型	bool UHFGetVersion (UCHAR* uSerial, UCHAR* uVersion)		
函数描述	读取 UHF 模块的硬件序列号和版本号		
参数	输出	uSerial	UHF 的硬件序列号（6 字节）
		uVersion	UHF 软件版本号（3 字节）
返回值	TRUE：操作成功;		
	FALSE：操作失败		

1.13.23 UHFInventorySingle()

函数原型	bool UHFInventorySingle (UCHAR* uLenUii, UCHAR* uUii)		
函数描述	单步识别标签（只对标签操作一次，无论失败或成功）		
参数	输出	uLenUii	标签的长度
		uUii	标签的 UII
返回值	TRUE：操作成功;		
	FALSE：操作失败		

1.13.24 UHFReadDataSingle()

函数原型	bool UHFReadDataSingle (UCHAR* uAccessPwd, UCHAR uBank, int uPtr, UCHAR uCnt, UCHAR* uReadData, UCHAR* uUii, UCHAR* uLenUii, UCHAR* uErrorCode)		
函数描述	读取标签数据（不指定 UII）		
参数	输入	uAccessPwd	标签的 ACCESSPASSWORD（4 字节）()
		uBank	标签的数据段类型（0x00-RESERVED、0x01-EPC、0x02-TID、0x03-USER
		uPtr	起始地址的偏移量（EBV 格式）
		uCnt	读出的数据长度（两个字节为单位，不允许为 0）
	输出	uReadData	读取标签的数据（至少为 uCnt*2 字节）
		uUii	标签的 UII
		uLenUii	UII 的长度（单位：字节）
		uErrorCode	标签返回的 Error Code（只有函数返回失败且 uErrorcode 不为 0xff 时有效）
返回值	TRUE：操作成功		
	FALSE：操作失败		

1.13.25 UHFWriteDataSingle()

函数原型	bool UHFWriteDataSingle (UCHAR* uAccessPwd, UCHAR uBank, int uPtr, UCHAR uCnt, UCHAR* uWriteData, UCHAR* uUii, UCHAR* uLenUii, UCHAR* uErrorCode)		
函数描述	写入标签数据（不指定 UII）		
参数	输入	uAccessPwd	标签的 ACCESSPASSWORD（4 字节）()
		uBank	标签的数据段类型（0x00-RESERVED、0x01-EPC、0x02-TID、0x03-USER
		uPtr	起始地址的偏移量（EBV 格式）
		uCnt	读出的数据长度（两个字节为单位，不允许为 0）
		uUii	标签的 UII
		uWriteData	要写入标签的数据
	输出	uUii	写入标签的 UII
		uLenUii	UII 的长度（单位：字节）
		uErrorCode	标签返回的 Error Code（只有函数返回失败且 uErrorcode 不为 0xff 时有效）
返回值	TRUE：操作成功		
	FALSE：操作失败		

1.13.26 UHFReadDataNoCnt()

函数原型	bool UHFReadDataNoCnt (UCHAR* uAccessPwd, UCHAR uBank, int uPtr, UCHAR* uUii, UCHAR* Data_len, UCHAR* uReadData, UCHAR* uErrorCode)		
函数描述	当 CNT 为 0 时的读取标签数据（指定 UII）操作		
参数	输	uAccessPwd	标签的 ACCESS PASSWORD（4 字节）
		uBank	标签数据类型（0x00-RESERVED、0x01-EPC、0x02-TID、

	入		0x03-USER）
		uPtr	起始地址偏移量（EBV 格式）
		uUii	UII--标签的 UII
	输出	Data_len	读取到的标签数据长度
		uReadData	读取到的标签数据
		uErrorCode	标签返回的 Error Code（只有函数返回失败且 uErrorcode 不为 0xff 时有效）
返回值	TRUE：操作成功;		
	FALSE：操作失败		

1.13.27 UHFReadDataSingleNoCnt()

函数原型	bool UHFReadDataSingleNoCnt (UCHAR* uAccessPwd, UCHAR uBank, int uPtr, UCHAR* Data_len, UCHAR* uReadData, UCHAR* uUii, UCHAR* uLenUii, UCHAR* uErrorCode)		
函数描述	当 CNT 为 0 时的度标签数据（不指定 UII）操作		
参数	输入	uAccessPwd	标签的 ACCESS PASSWORD（4 字节）
		uBank	标签数据类型（0x00-RESERVED、0x01-EPC、0x02-TID、0x03-USER）
		uPtr	起始地址偏移量（EBV 格式）
	输出	Data_len	读取到的标签数据长度
		uReadData	读取到的标签数据
		uUii	标签 UII
		uLenUii	UII 长度
		uErrorCode	标签返回的 Error Code（只有函数返回失败且 uErrorcode 不为 0xff 时有效）
返回值	TRUE：操作成功;		
	FALSE：操作失败		

1.13.28 UHFEraseDataSingle()

函数原型	bool UHFEraseDataSingle (UCHAR* uAccessPwd, UCHAR uBank, int uPtr, UCHAR uCnt, UCHAR* uUii, UCHAR* uErrorCode)		
函数描述	擦除标签数据（不指定 UII）		
参数	输入	uAccessPwd	标签的 ACCESSPASSWORD（4 字节）
		uBank	标签的数据段类型（0x00-RESERVED、0x01-EPC、0x02-TID、0x03-USER
		uPtr	起始地址的偏移量（EBV 格式）
		uCnt	需要擦除数据的长度（两字节为单位）
	输出	uUii	返回擦除数据的 UII
		uErrorCode	标签返回的 Error Code（只有函数返回失败且 uErrorcode 不为 0xff 时有效）
返回值	TRUE：操作成功		
	FALSE：操作失败		

1.13.29 UHFLockMemSingle()

函数原型	bool UHFLockMemSingle (UCHAR* uAccessPwd, UCHAR* uLockData, UCHAR* uUii, UCHAR* uErrorCode)		
函数描述	锁定标签指定的数据段（不指定 UII）		
参数	输入	uAccessPwd	标签的 ACCESSPASSWORD（4 字节）
		uLockData	命令的 LockData 数据段（3 字节）
	输出	uUii	返回标签的 UII
		uErrorCode	标签返回的 Error Code（只有函数返回失败且 uErrorcode 不为 0xff 时有效）
返回值	TRUE：操作成功;		
	FALSE：操作失败		

1.13.30 UHFKillTagSingle()

函数原型	bool UHFKillTagSingle (UCHAR* uKillPwd, UCHAR* uUii, UCHAR* uErrorCode)		
函数描述	销毁标签（不指定 UII）		
参数	输入	uKillPwd	标签的 Kill Password(4 字节)
	输出	uUii	返回标签的 UII
		uErrorCode	标签返回的 Error Code（只有函数返回失败且 uErrorcode 不为 0xff 时有效）
返回值	TRUE：操作成功;		
	FALSE：操作失败		

1.13.31 UHFAddSelect()

函数原型	bool UHFAddSelect (UCHAR* precord, UCHAR* STATUS)		
函数描述	添加 Select 记录		
参数	输入	precord	为 sRECORD 结构体的首地址（详情见本注释末端 SRECORD 定义）
	输出	STATUS	返回的操作状态信息（0x00-成功、0x01-SINDEX 超出范围、0x02-SINDEX=0 是只读的 SRECORD、0x03-未定义）
返回值	TRUE：操作成功;		
	FALSE：操作失败		

1.13.32 UHFDeleteSelect()

函数原型	bool UHFDeleteSelect (UCHAR SINDEX, UCHAR* STATUS)		
函数描述	删除 Select 记录		
参数	输入	SINDEX	所要删除的 Select 序列号
	输出	STATUS	返回操作状态信息（0x00-成功、0x01-SINDEX 超出范围、0x02-SINDEX=0 是只读的 SRECORD、0x03-未定义）
返回值	TRUE：操作成功;		
	FALSE：操作失败		

1.13.33 UHFGetSelect()

函数原型	bool UHFGetSelect (UCHAR SINDEX, UCHAR SNUM, UCHAR* STATUS)		
函数描述	读取 Select 记录		
参数	输入	SINDEX	所要读取的 Select 序列号
		SNUM	需要读取 SeLect 的个数
	输出	STATUS	返回操作状态信息（0x00-成功、0x01-SINDEX 超出范围、0x02-SINDEX=0 是只读的 SRECORD、0x03-未定义）
返回值	TRUE：操作成功;		
	FALSE：操作失败		

1.13.34 UHFGetSelectReceived()

函数原型	bool UHFGetSelectReceived(UCHAR* STATUS, UCHAR* precord)		
函数描述	获取 Select 记录		
参数	输出	STATUS	返回操作状态信息（0x00-成功、0x01-SINDEX 超出范围、0x02-SINDEX=0 是只读的 SRECORD、0x03-未定义）
		psRecord	返回指针指向 SRECORD 结构体（详情见末端 SRECORD 声明）
返回值	TRUE：操作成功;		
	FALSE：操作失败		

1.13.35 UHFChooseSelect()

函数原型	bool UHFChooseSelect (UCHAR SINDEX, UCHAR SNUM, UCHAR* STATUS)		
函数描述	选择发送 Select 命令参数		
参数	输入	SINDEX	需要选择的 SRECORD 的起始号（0~15）
		SNUM	SNUM--需要选择 SRECORD 的个数
	输出	STATUS	返回操作状态信息（0x00-成功、0x01-SINDEX 超出范围、0x02-SINDEX=0 是只读的 SRECORD、0x03-未定义）
返回值	TRUE：操作成功;		
	FALSE：操作失败		

1.13.36 UHFInventoryAdvanced()

函数原型	bool UHFInventoryAdvanced(UCHAR SINDEX, UCHAR SNUM, UCHAR SST, UCHAR Q)		
函数描述	高级识别命令		
参数	输入	SINDEX	-SRECORD 中的 Selec 命令起始号（0~15）
		SNUM	SNUM--发送 SRECORD 中的 Select 命令个数
		SST	发送 Sel，Session，Target 的参数（见本注释末端定义）
		Q	防碰撞 Q 值
返回值	TRUE：操作成功;		
	FALSE：操作失败		

1.13.37 UHFGetInventoryAdvancedStatus()

函数原型	bool UHFGetInventoryAdvancedStatus(UCHAR* STATUS, int* TNUM)		
函数描述	获取高级识别的返回状态值，所识别的标签数量		
参数	输入	STATUS	高级识别状态
	输出	TNUM	所识别的标签数量
返回值	TRUE：操作成功;		
	FALSE：操作失败		

1.13.38 UHFReadBuffer()

函数原型	bool UHFReadBuffer (int TINDEX, int TLEN, UCHAR* STATUS)		
函数描述	该函数用于读取标签缓冲区命令		
参数	输入	TINDEX	要选取缓冲区的起始号（2 字节）
		TLEN	返回标签信息个数（2 字节）
		TMODE	TIME 字段选项（1-响应 TRECORD 中包含 TIME，0-不包含 TIME）
	输出	STATUS	返回操作状态信息（0x00-有后续响应、0x01-没有后续响应,完成 TLEN 个 TRECORD 传输、0x03-没有后续响应，该序列不存在）
返回值	TRUE：操作成功;		
	FALSE：操作失败		

1.13.39 UHFGetBuffer()

函数原型	boolUHFGetBuffer (UCHAR* precord, UCHAR* STATUS)		
函数描述	获取标签缓存区数据		
参数	输入	precord	标签信息字段（指向 TRECORD 结构体的指针）
	输出	STATUS	返回操作状态信息(0x00-有后续响应、0x01-没有后续响应，完成 TLEN 个 TRECORD 传输、0x03-没有后续响应，该序列不存在）
返回值	TRUE：操作成功;		
	FALSE：操作失败		

1.14 指纹

1.14.1 EMFingerInit()

函数原型	bool EMFingerInit(int iPort, int iBaudRate)	
函数描述	初始化指纹模块	
参数	输入	iPort：串口号（数字 1、2、3..）
		iBaudRate:指纹模块的波特率，默认为 57600，可通过参数设置调整波特率

返回值	TRUE：操作成功	
	FALSE：操作失败	

1.14.2 EMFingerFree()

函数原型	void EMFingerFree (int iPort)	
函数描述	关闭指纹模块操作端口	
参数	输入	iPort：串口号（数字 1、2、3..）
返回值	无	

1.14.3 EMGetRandomData()

函数原型	bool EMGetRandomData(UCHAR *pszData)	
函数描述	获取随机数	
参数	输出	pszData：1 字节长度+数据内容
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.14.4 EMGetImage()

函数原型	bool EMGetImage()
函数描述	获取指纹图像信息并存于图像缓冲区
参数	无
返回值	TRUE：操作成功
	FALSE：操作失败

1.14.5 EMGenChar()

函数原型	bool EMGenChar(UCHAR GenID)	
函数描述	提取图像缓冲区的图像信息的特征文件并存于模板缓冲区 GenID	
参数	输入	GenID：取值为 1 或 2
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.14.6 EMMatch()

函数原型	bool EMMatch(int *MatchScore)	
函数描述	对比模板缓冲区与模板缓冲区的指纹特征文件	
参数	输出	MatchScore：指纹对比得分
返回值	TRUE：操作成功	
	FALSE：操作失	

1.14.7 EMSearch()

函数原型	bool EMSearch(UCHAR BuffID, int StartPage, int PageNum, int *PageID, int *MatchScore)	
函数描述	指纹对比，找出特征值相同的数据	
参数		BuffID：模板缓冲区地址（1 或 2）

	输入	StartPage：起始页
		PageNum：需要搜索的 ID 号个数
	输出	PageID：搜到的页码
		MatchScore：比对得分
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.14.8 EMRegModel()

函数原型	bool EMRegModel()1	
函数描述	将模板缓冲区与模板缓冲区中的模板文件合并生成模板，结果存于模板缓冲区	
参数	无	
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.14.9 EMStorChar()

函数原型	bool EMStorChar(UCHAR BuffID, int PageID)	
函数描述	将模板缓冲区 BuffID 中的模板文件存到 PageID 号所对应的指纹库位置	
参数	输入	BuffID：模板缓冲区（1 或 2）
		PageID：指纹库位置号（0~32768），ID 由用户自行分配
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.14.10 EMLoadChar()

函数原型	bool EMLoadChar(UCHAR BuffID, int PageID)	
函数描述	将指纹库中指定 PageID 号的指纹模板读入到模板缓冲区	
参数	输入	BuffID：模板缓冲区（1 或 2）
		PageID：指纹库位置号（0~32768），ID 由用户自行分配
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.14.11 EMUpChar()

函数原型	bool EMUpChar(UCHAR BuffID, UCHAR *pszData)	
函数描述	把模板缓冲区 BuffID 中的模板文件存放在 pszData 的数组中（512 字节模板文	
参数	输入	BuffID：模板缓冲区（1 或 2）
	输出	pszData--获得的特征文件指针
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.14.12 EMDownChar()

函数原型	bool EMDownChar(UCHAR BuffID, UCHAR *pszData)

函数描述	上位机下载特征文件到模块的一个特征缓冲区	
参数	输入	BuffID：模板缓冲区（1 或 2）
		pszData—特征文件首地址（512 字节）
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.14.13 EMDeletChar()

函数原型	bool EMDeletChar(int PageID, int nNum)	
函数描述	删除指纹库中指定 PageID 号开始的 nNum 个指纹模板	
参数	输入	PageID：指纹库中模板文件的起始页码
		nNum：连续删除的指纹页数
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.14.14 EMEmpty()

函数原型	bool EMEmpty()
函数描述	删除指纹库中所有指纹模板
参数	无
返回值	TRUE：操作成功
	FALSE：操作失败

1.14.15 EMSetReg()

函数原型	bool EMSetReg(UCHAR RegID, UCHAR nValue)	
函数描述	设置系统寄存器	
参数	输入	RegID：寄存器序号
		nValue：需设置的数值
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.14.16 EMAutoEnroll()

函数原型	bool EMAutoEnroll(UCHAR nTime, int UserID, int *RUserID)	
函数描述	刷两次指纹，模块自动完成合并指纹模板、注册功能	
参数	输入	nTime：自动录取指纹的次数
		IuserID：注册 ID
	输出	RuserID：用户 ID
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.14.17 EMAutoMatch()

函数原型	bool EMAutoMatch(UCHAR nFlag, int StartPage, int PageNum, int *PageID, int *MatchScore)
函数描述	录入一次指纹，自动完成比对功能

参数	输入	nFlag：模板缓冲区（1 或 2）
		StartPage：起始查找的页码
		PageNum：需要查找的页数
	输出	PageID：相匹配的页码
		MatchStore：比对得分
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.14.18 EMSetPSW()

函数原型	bool EMSetPSW(UCHAR *PassWord)	
函数描述	设置模块口令	
参数	输入	PassWord：4 字节口令
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.14.19 EMVfyPSW()

函数原型	bool EMVfyPSW (UCHAR *PassWord)	
函数描述	验证模块握手口令	
参数	输入	PassWord：4 字节口令
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.14.20 EMValidTempleteNum()

函数原型	bool EMValidTempleteNum(int *nValueModel)	
函数描述	读指纹库中有效模板个数	
参数	输出	nValueModel：有效的指纹模板文件个数
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.14.21 EMReadChipSN()

函数原型	bool EMReadChipSN(UCHAR *SerialNum)	
函数描述	读取模块序列号并返回给上位机（主要用于认证产品的唯一性）	
参数	输出	SerialNum：1 字节长度+序列号（采用 ASCII 码）
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.14.22 EMSetManuFacture()

函数原型	bool EMSetManuFacture(UCHAR *SerialNum)	
函数描述	设置产品序列号	
参数	输入	DeviceName：8 字节设备名称（ASCII 码表示）
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.14.23 EMSetDeviceName()

函数原型	bool EMSetDeviceName(UCHAR *DeviceName)	
函数描述	设置设备商名称	
参数	输入	DeviceName：8 字节设备名称（ASCII 码表示）
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.14.24 EMReadSysPara()

函数原型	bool EMReadSysPara(UCHAR *pszData)	
函数描述	获取系统参数	
参数	输出	pcData：35 字节系统信息：指纹模块版本号+型号（ASCII 码格式）
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.14.25 EMUpImage()

函数原型	bool EMUpImage(unsigned char Mode, LPCTSTR lpFileName, unsigned int *TotalLen)	
函数描述	下载指纹图片到上位机	
参数	输入	mode--是否带 bmp 格式头文件 0 表示不带、非零表示带
		pFile--需要进行操作的文件句柄
	输出	TotalLen：总共获取到的文件字节长度（字节为单位）
返回值	TRUE：操作成功	
	FALSE：操作失败	

1.15　红外

红外的功能目前只有 C5000 支持。

1.15.1 Infrared_init()

函数原型	BOOL Infrared_init()
函数描述	对红外模块上电并打开对应的串口
参数	空
返回值	TRUE：　上电和打开串口成功
	FALSE：　上电和打开串口失败

1.15.2 Infrared_free()

函数原型	BOOL Infrared_free()
函数描述	对红外模块下电并关闭对应的串口
参数	空
返回值	TRUE：　下电和关闭串口成功
	FALSE：　下电和关闭串口失败

1.15.3 Infrared_send()

函数原型	DWORD Infrared_send(UINT8* SendBuffer,DWORD dwLength)
函数描述	infrared 发送数据
参数	输入：SendBuffer，发送数据的缓冲区指针
	输入：dwLength，发送数据的长度
返回值	实际发送的数据长度

1.15.4 Infrared_receive()

函数原型	DWORD Infrared_receive(UINT8* ReceiveBuffer)
函数描述	infrared 接收数据
参数	输出：ReceiveBuffer，接收数据的缓冲区指针
返回值	实际接收的数据长度

1.16 背光

1.16.1 GetBackLightLevel()

函数原型	DWORD GetBackLightLevel()
函数描述	获取背光亮度等级
参数	空
返回值	背光亮度等级，取值范围为 1 到 10。

1.16.2 SetBackLightLevel()

函数原型	bool SetBackLightLevel(int iLevel)
函数描述	设置背光亮度等级
参数	输入：iLevel，背光亮度等级，范围为 1 到 10.
返回值	TRUE：设置成功。
	FALSE：设置失败

1.17 Power

1.17.1 module_power_control()

此 API 对于不同的产品其支持的功能有差别，具体如下：

C2000:

支持 HF 的上下电；支持 1D 的上下电、扫描头出光和不出光；支持 2D 硬解码的上下电、扫描头出光和不出光。

C3000：

支持 HF 上下电；支持 1D 的上下电、扫描头出光和不出光。

C5000：

支持 LF、HF、指纹和红外的上下电；支持 1D 的上下电、扫描头出光和不出光；支持 2D 硬解码的上下电、扫描头出光和不出光。

函数原型	BOOL module_power_control(UINT8 module,UINT8 action)
函数描述	控制具体模块的电源

参数	输入：module，表示不同的功能模块，功能模块的值如下： LF：0 HF:1 1D:3 2D 硬解码：4 指纹：8 红外：9
	输入：action，表示要控制模块的动作，比如关闭 1D 电源，开启 1D 电源，扫描头是否出光等，具体说明如下： LF、HF、1D、2D 硬解码、指纹和红外的下电对应此值为 0，上电对应此值为 1。 1D 和 2D 硬解码扫描头不出光对应此值为 2，扫描头出光对应此值为 3。
返回值	FALSE：控制的动作失败，TRUE：控制的动作成功。

1.17.2 system_power_down()

函数原型	BOOL system_power_down()
函数描述	关机
参数	空
返回值	TRUE：关机成功。
	FALSE：关机失败

1.17.3 power_key_enabled()

函数原型	bool power_key_enabled()
函数描述	开启 power 按键功能
参数	空
返回值	TRUE：开启成功。
	FALSE：开关机失败

1.17.4 power_key_disabled()

函数原型	BOOL power_key_disabled()
函数描述	关闭 power 按键功能
参数	空
返回值	TRUE：关闭成功。
	FALSE：关闭失败

1.17.5 system_restart()

函数原型	BOOL system_restart(void)
函数描述	重启系统
参数	空
返回值	TRUE：表示当前的系统支持重启功能，且能软件重启。
	FALSE：表示当前的系统不支持重启功能，且不能软件重启。

备注：对于 C2000 来说，2013.03.13 及之后版本的系统才支持此功能，对于 C5000 来说，2013.04 月份及之后的系统才支持此功能

1.18 触摸

1.18.1 touch_screen_enable()

函数原型	BOOL touch_screen_enable()
函数描述	开启触摸功能
参数	空
返回值	TRUE：开启成功。
	FALSE：开启失败

1.18.2 touch_screen_disable()

函数原型	BOOL touch_screen_disable()
函数描述	关闭触摸功能
参数	空
返回值	TRUE：关闭成功。
	FALSE：关闭失败

1.19 USB

1.19.1 usb_device_get_current_mode()

函数原型	UINT8usb_device_get_current_mode()
函数描述	获取 USB Slave 功能的当前工作模式
参数	空
返回值	0：USB Slave 的功能被禁用 1：存储卡作为 U 盘 2：Flash Disk 分区作为 U 盘 3：ActiveSync 功能

1.19.2 usb_device_close_usb_function()

函数原型	BOOL usb_device_close_usb_function()
函数描述	关闭 USB Slave 功能
参数	空
返回值	TRUE：关闭成功。
	FALSE：关闭失败

1.19.3 usb_device_storage_card_as_udisk()

函数原型	BOOL usb_device_storage_card_as_udisk()
函数描述	设置存储卡作为 U 盘模式
参数	空
返回值	TRUE：设置成功。

	FALSE：设置失败

1.19.4 usb_device_flash_as_udisk()

函数原型	BOOL usb_device_flash_as_udisk()
函数描述	设置 Flash Disk 作为 U 盘模式
参数	空
返回值	TRUE：设置成功。
	FALSE：设置失败

1.19.5 usb_device_set_as_activeSync()

函数原型	BOOL usb_device_set_as_activeSync()
函数描述	设置为 ActiveSync 同步模式
参数	空
返回值	TRUE：设置成功。
	FALSE：设置失败

1.20 LED

1.20.1 led_control_init()

函数原型	bool led_control_init()
函数描述	LED 灯控制的初始化
参数	空
返回值	TRUE：初始化成功。
	FALSE：初始化失败

1.20.2 led_control_switch()

函数原型	bool led_control_free()	
函数描述	控制指定 LED 点亮或是熄灭	
参数	输入	LedNum，对应的 LED 灯，比如 1 表示 LED1
		Switch，表示控制开关，TRUE 表示点亮，FALSE 表示熄灭
返回值	TRUE：控制成功。	
	FALSE：控制失败	

1.20.3 led_control_free()

函数原型	bool led_control_free()
函数描述	LED 灯控制资源的释放
参数	空
返回值	TRUE：释放资源成功。
	FALSE：释放资源失败

1.21 扩展串口

1.21.1 extended_serial_open ()

函数原型	BOOL extended_serial_open(UINT8 ComIndex,int iBaudRate)	
函数描述	以指定的波特率打开所选的扩展串口	
参数	ComIndex：ComIndex=0 表示 COM3，ComIndex=1 表示 COM4，目前仅支持 C5000。 iBaudRate：表示串口波特率，支持的波特率有 1200、2400、4800、9600、14400、19200、28800、38400、57600 和 115200。	
返回值	TRUE：打开串口成功。	
	FALSE：打开串口失败	

1.21.2 extended_serial_close ()

函数原型	BOOL extended_serial_close(void)	
函数描述	关闭扩展串口	
参数	空	
返回值	TRUE：关闭串口成功。	
	FALSE：关闭串口失败	

1.21.3 extended_serial_data_send ()

函数原型	int extended_serial_data_send (UINT8 *pszData, int iLength)	
函数描述	扩展串口发送数据	
参数	输入	pszData，指向要发送数据的指针
		iLength，发送数据的长度
返回值	返回实际发送的数据长度	

1.21.4 extended_serial_data_receive ()

函数原型	int extended_serial_data_receive (UINT8 *pszBuffer, int iLength)	
函数描述	扩展串口接收数据	
参数	输入	pszBuffer，指向要发送数据的指针
		iLength，要接收的数据长度
返回值	实际接收的数据长度	

1.22 扩展函数接口

1.22.1 READ_API_Version()

函数原型	void READ_API_Version(UINT8 *pszData)	
函数描述	获取 DeviceAPI.dll 的版本信息	
参数	输出	pszData，包含查询到的版本号信息，比如“DeviceAPI V2.21(2013.03.04)”
返回值	空	

1.22.2 HardwareVersion_Ex()

函数原型	int HardwareVersion_Ex(UINT8 *pszData)
函数描述	获取 RFID LF 和 HF 板单片机程序的版本信息

参数	输出	pszData，包含查询到的版本号信息
返回值	版本号长度(单位为字节)	

1.22.3 SerialPortSwitch_Ex()

备注：此 API 仅用于 C5000，此 API 和硬件控制关系紧密，如果硬件改变了，此 API 也需要跟着做相应的改变。对于新的客户必须要采用兼容三个项目的 API；对于开发过程及愿意修改的客户，尽量让他们采用兼容三个项目的 API，也便于他们在三个产品中使用，也可以避免以后兼容性问题，因为每个功能模块的初始化/上电函数(比如 RF_ISO14443A_init())都做包含了端口切换的动作,此 API 仅为确保客户不需要修改程序就可以在 C5000 新旧副板正常工作而保留。

函数原型	void SerialPortSwitch_Ex(UINT8 ComID)	
函数描述	切换为对应功能模块的串口	
参数	输入	ComID，指示要切换到功能模块对应的端口号 0：切换为 RFID LF/HF 的串口 1：切换为 UHF/红外/指纹/扩展 GPS 板(采用集成了 GPS 和 BD2 功能的模块) 2：1D/2D 硬解码 3：GPS
返回值	无	

1.22.4 SerialPortControl_Ex()

备注：此 API 仅用于 C5000，此 API 和硬件控制关系紧密，如果硬件改变了，此 API 也需要跟着做相应的改变。对于新的客户必须要采用兼容三个项目的 API；对于开发过程及愿意修改的客户，尽量让他们采用兼容三个项目的 API，也便于他们在三个产品中使用，也可以避免以后兼容性问题，可以使用每个功能模块电源控制 API(比如 RF_ISO14443A_init())，也可调用 module_power_control 函数控制某些模块的电源部分(不涉及打开串口和关闭串口),从 2013 年 9 月份版本开始的 DeviceAPI 增加了 extended_interface_power_control()，此 API 主要针对扩展口电源的控制，更好兼容目前所有的硬件版本，强烈建议客户采用这个 API，SerialPortControl_Ex()仅为确保客户不需要修改程序就可以在 C5000 新旧副板正常工作而保留。

函数原型	void SerialPortControl_Ex(UINT8 uPortID, UINT8 uValue)	
函数描述	控制对应端口的上下电	
参数	输入	uPortID：要控制的端口号
		uValue：0，表示要设置此端口为低电平；1 为高电平
返回值	无	

1.22.5 extended_interface_power_control()

函数原型	BOOL extended_interface_power_control(UINT8 PinInterface, UINT8 InterfaceControl)
函数描述	为扩展模块进行电源及 GPIO 控制。

参数	PinInterface，输入，表示要控制的引脚，比如要控制 5V 引脚： 1：3.3V 引脚。 2：RFID 接口的 5V，C5000 旧副板不支持。 3：RFID 接口引出的 VBAT (Q302 第 3 引脚，TP307，底面)，可控制---C5000 旧副板不支持。 4：RFID 接口的 GPIO1---TP311，C5000 旧副板不支持。 5 和 6：预留。 7：扩展接口的 5V。 8： 扩展接口引出的 VBAT(Q303 的第 3 引脚---TP301)，可控制---C5000 旧副板不支持。 9：扩展接口的 EN1---目前给客户引出的使能引脚就是 EN1。 10：扩展接口的 EN2。 11：对应于 IO2-OC，客户需要通过此引脚上拉连接到外部模块或是板的电源，此值对应的控制值为 0 时，此引脚输出 0V，对应的控制值为 1 时，此引脚输出的是外部模块或是板的电源电压。 其他：根据需要再扩展。 InterfaceControl，输入，表示要控制的引脚的动作，比如开或是关 5V。TRUE 为打开，FALSE 为关闭。
返回值	TRUE：操作成功;
	FALSE：操作失败

1.22.6 SerialPortOpen_Ex()

备注：此 API 仅用于 C5000，因为大部分功能模块都需要对应的波特率才能正常工作，比如 LF、HF 等，此 API 只能以指定的波特率打开 COM4，且我们绝大部分的功能模块的初始化/上电函数都已正确的波特率打开对应的 COM 口，故此 API 没有太多的应用场合，对于新的客户必须要采用兼容三个项目的 API；对于开发过程及愿意修改的客户，尽量让他们采用兼容三个项目的 API，也便于他们在三个产品中使用，也可以避免以后兼容性问题，此 API 仅为确保客户不需要修改程序就可以在 C5000 新旧副板正常工作而保留。

函数原型	bool SerialPortOpen_Ex(int iBaudRate)	
函数描述	以指定的波特率打开 COM4	
参数	输入	iBaudRate：波特率，支持的波特率有 1200、2400、4800、9600、14400、19200、28800、38400、57600 和 115200
返回值	TRUE:以指定波特率打开 COM4 成功，FALSE:打开失败	

1.22.7 SerialPortSetBaudRate_Ex()

备注：此 API 仅用于 C5000，因为大部分功能模块都需要对应的波特率才能正常工作，比如 LF、HF 等，此 API 只能以指定的波特率打开 COM4，且我们绝大部分的功能模块的初始化/上电函数都已正确的波特率打开对应的 COM 口，故此 API 没有太多的应用场合。对于新的客户必须要采用兼容三个项目的 API；对于开发过程及愿意修改的客户，尽量让他们采用兼容三个项目的 API，也便于他们在三个产品中使用，也可以避免以后兼容性问题，此 API 仅为确保客户不需要修改程序就可以在 C5000 新旧副板正常工作而保留。

函数原型	bool SerialPortSetBaudRate_Ex(int iBaudRate)	
函数描述	设置 COM4 串口的波特率	
参数	输入	iBaudRate：波特率，支持的波特率有 1200、2400、4800、9600、14400、19200、28800、38400、57600 和 115200
返回值	TRUE:设置波特率成功，FALSE:设置波特率失败	

1.22.8 SerialPortFunctionSwitch_Ex()

备注：此 API 仅用于 C5000，功能模块的初始化/上电函数中已经做了切换的动作(比如 RF_ISO14443A_init()),故此 API 没有太多的应用场合。对于新的客户必须要采用兼容三个项目的 API；对于开发过程及愿意修改的客户，尽量让他们采用兼容三个项目的 API，也便于他们在三个产品中使用，也可以避免以后兼容性问题，此 API 仅为确保客户不需要修改程序就可以在 C5000 新旧副板正常工作而保留。

函数原型	bool SerialPortFunctionSwitch_Ex(int iModule)	
函数描述	针对 RFID、条码 功能间相互切换 必须在上电之后使用该指令	
参数	输入	iModule：0 表示切到 RFID，1 表示切到条码
返回值	TRUE:切换成功，FALSE:切换失败	

1.23 操作系统功能接口

1.23.1 Uniscribe_disable ()

C2000 和 C5000 从 2012 年 11 月份开始的非中文系统增加了 Unicode Processor for Complex Scripts，此功能主要用于支持从右到左显示的语言的显示，比如阿拉伯语等，增加了此功能，系统的响应比没有增加会慢一点。此版本增加 Uniscribe_disable 函数，用于开启和关闭 Unicode Processor for Complex Scripts 的功能，用户可以根据需要来选择开启还是关闭此功能，调用此函数后需要重启才生效。

函数原型	BOOL Uniscribe_disable(BOOL Switch)
函数描述	是否开启 Uniscribe 功能
参数	输入：Switch=FALSE，表示开启 Uniscribe 功能;Switch=TRUE，表示关闭 Uniscribe 功能。
返回值	FALSE：控制的动作失败，TURE：控制的动作成功。

1.24 3G

3G 部分开发注意事项：

(1)PDA 开机后 3G 模块会自动上电

(2)休眠唤醒后 3G 模块至少需要等待 8 秒钟以后再去连接

1.24.1 GPRSPowerOn()

函数原型	BOOL GPRSPowerOn ()
函数描述	对 3G 模块上电
参数	空
返回值	TRUE：3G 模块上电成功
	FALSE：3G 模块上电失败

备注：目前系统不支持 GPRSPowerOn()函数。

1.24.2 GPRSPowerOff()

函数原型	BOOL GPRSPowerOff()
函数描述	对 3G 模块下电
参数	空
返回值	TRUE：3G 模块下电成功
	FALSE：3G 模块下电失败

备注：目前系统不支持 GPRSPowerOff()函数。

1.24.3 GPRSReset()

函数原型	BOOL GPRSReset()
函数描述	对 3G 模块复位
参数	空
返回值	TRUE：3G 模块复位成功
	FALSE：3G 模块复位失败

备注：目前系统不支持 GPRSReset()函数。

1.24.4 GPRS_judge_connection_status()

函数原型	BOOL GPRS_judge_connection_status(LPCTSTR GprsName)
函数描述	判断名为 GprsName 的 3G 连接的连接状态
参数	输入：GprsName，3G 拨号连接的名称
返回值	TRUE：名为 GprsName 的 3G 处于连接的状态
	FALSE：名为 GprsName 的 3G 处于断开的状态

备注：目前系统不支持 GPRS_judge_connection_status()函数。

1.24.5 GPRS_judge_modem_status()

函数原型	BOOL GPRS_judge_modem_status ()
函数描述	判断 3G 模块是否已经上电
参数	空
返回值	TRUE：3G 模块已经上电
	FALSE：3G 模块还没有上电

备注：目前系统不支持 GPRS_judge_modem_status()函数。

1.24.6 GPRS_open_serial()

函数原型	UINT8 GPRS_open_serial()
函数描述	打开 3G 模块对应的串口
参数	空
返回值	0：打开串口失败
	1：打开串口成功
	2：串口已经打开

1.24.7 GPRS_close_serial()

函数原型	BOOL GPRS_close_serial()
函数描述	关闭 3G 模块对应的串口

参数	空
返回值	TRUE： 关闭串口成功
	FALSE： 关闭串口失败

1.24.8 GPRS_send_data()

函数原型	DWORD GPRS_send_data(UINT8* SendBuffer,DWORD dwLength)
函数描述	3G 发送数据
参数	输入：SendBuffer，发送数据的缓冲区指针
	输入：dwLength，发送数据的长度
返回值	实际发送的数据长度

1.24.9 GPRS_receive_data()

函数原型	DWORDGPRS_receive_data(UINT8*ReceiveBuffer,DWORD dwLength)
函数描述	3G 接收数据
参数	输出：ReceiveBuffer，接收数据的缓冲区指针
	输入：dwLength，要接收数据的长度
返回值	实际接收的数据长度

1.25 蓝牙

1.25.1 bluetooth_load ()

函数原型	BOOL bluetooth_load()
函数描述	蓝牙上电和驱动加载
参数	空
返回值	TRUE：上电加载成功。
	FALSE：上电加载 s 失败

1.25.2 bluetooth_unload ()

函数原型	BOOL bluetooth_unload()
函数描述	蓝牙下电和驱动卸载
参数	空
返回值	TRUE：下电卸载成功。
	FALSE：下电卸载失败

备注：目前 C3000 不支持蓝牙上下电/加载卸载函数。

2. CameraAPI.dll 的 API 说明

2.1 GetCameraAPI_Version()

函数原型	void GetCameraAPI_Version(UINT8 *pszData)	
函数描述	获取 CameraAPI.dll 的版本信息，包括版 CameraAPI 名称、版本号和日期，比如"CameraAPI V1.0(2012.12.22)"	
参数	输出	pszData，指向获取的版本信息
返回值	空	

2.2 Preview()

函数原型	BOOL Preview(HWND hVideoWnd, LPCTSTR fileName,int resolution)	
函数描述	初始化 camera 并开始预览	
参数	输入	hVideoWnd：显示预览画面控件句柄，MFC 中对应的是 m_hWnd，C#中对应的是 Handle
		fileName：录像保存路径
		resolution：拍照时照片的分辨率；(0 是 320*240，1 是 640*480，2 是 1024*768，3 是 2048*1536)
返回值	TRUE：预览成功	
	FALSE：预览失败	

备注：此 API 包含了对 camera 模块的上电初始化及预览初始化，但这样设计并不是很合理，不推荐使用，现在改为用 CameraInit 和 CameraPreview 来实现对应的功能，见下面的描述。

2.3 CameraInit()

函数原型	void CameraInit(HWND hVideoWnd)	
函数描述	初始化 camera 模块，打开 camera 应用程序的时候应调用此函数	
参数	输入	hVideoWnd：显示预览画面控件句柄，MFC 中对应的是 m_hWnd，C#中对应的是 Handle
返回值	无	

2.4 CameraPreview()

函数原型	void CameraPreview(int resolution)	
函数描述	拍照预览	
参数	输入	resolution：拍照时照片的分辨率；(0 是 320*240，1 是 640*480，2 是 1024*768，3 是 2048*1536)
返回值	无	

2.5 TakePicture()

函数原型	BOOL TakePicture(LPCTSTR fileName)	
函数描述	拍照	
参数	输入	fileName：录像保存路径
返回值	TRUE：拍照成功	
	FALSE：拍照失败	

2.6 FreeCamera()

函数原型	void FreeCamera()
函数描述	释放 camera 资源
参数	空
返回值	无

2.7 Camera_Led_Ctrl()

函数原型	BOOL Camera_Led_Ctrl(BOOL swich)

函数描述	控制 C3000 摄像头拍照的闪光灯函数	
参数	输入	swich；TRUE 表示打开闪光灯，FALSE 表示关闭闪光灯。
返回值	TRUE：打开或者关闭闪光灯成功 FALSE：打开或者关闭闪光灯失败	

注：2014 年 1 月 23 号以后的 API 新增的函数