

SDK 接口函数说明文档

版本：V1.4.1

/**解码触发方式**/

typedef enum DB_TRIGGER_MODE_

{

DB_TRIGGER_M_LEVEL = 0x00, /**[被动](电平)按键保持**/ //默认

DB_TRIGGER_M_PULSE, /**[被动](脉冲)单次按键触发模式**/

DB_TRIGGER_M_CONTINUOUS, /**[主动]连续模式**/

DB_TRIGGER_M_AUTOSENS, /**[主动]自动感应模式**/

DB_TRIGGER_M_HOST, /**[被动]主机(通过指令控制)**/

DB_TRIGGER_M_KEY_CONTINUOUS, /**[被动]按键连续模式**/ /**按键控制下的连续模式,该模式下开启和结束扫描都由按键控制,解码成功和超时不结束扫描**/

DB_TRIGGER_M_KEY_AUTOSENS, /**[被动]按键自动感应模式**/ /**按键一直按下的情况下进入自动感应模式,直到按键释放结束扫描,特殊:按键按下定位灯亮,释放灭**/

DB_TRIGGER_M_MAX,

}DB_TRIGGER_MODE_E;

/**通信方式**/

typedef enum DB_COMM_MODE_

{

DB_COMM_M_UART = 0x00, /**00:串口**/

DB_COMM_M_USB_KBW, /**01:USB KBW(键盘)**/

DB_COMM_M_USB_VCOM, /**02:USB 转串口**/

DB_COMM_M_AUTO_UK, /**03:自动模式 UK,USB 和串口同时输出(USB 使用 KBW)**/

DB_COMM_M_AUTO_UV, /**04:自动模式 UV,USB 和串口同时输出(USB 使用 USB 串口)**/

DB_COMM_M_WG, /**05:韦根**/

DB_COMM_M_RS485, /**06:RS485**/

DB_COMM_M_AUTO_UW, /**07:自动模式 UW,USB 和韦根同时输出(USB 使用 USB 串口)**/

DB_COMM_M_AUTO_UR, /**08:自动模式 UR,USB 和 RS485 同时输出(USB 使用 USB 串口)**/

```

    DB_COMM_M_PS2_KBW,          /**09:PS2 键盘**/

    DB_COMM_M_TTDATA,          /**0A:自定义 usb 通信**/
    DB_COMM_M_AUTO_UT,         /**0B:自动模式 UT,USB 自定义通信和串口同时输出**/
    DB_COMM_M_AUTO_TW,         /**0C:自动模式 TW,USB 和韦根同事输出(USB 使用 usb
自定义通信)**/
    DB_COMM_M_AUTO_TR,         /**0D:自动模式 TR,USB 和 RS485 同时输出(USB 使用 usb
自定义通信)**/

    DB_COMM_M_HID_POS,         /**0E:KBW 与 USB HID 同时输出，HID 可用于交互**/

    DB_COMM_M_MAX,
}DB_COMM_MODE_E;

/**LED 灯的状态(定位灯和照明灯)**/
typedef enum DB_LED_STATE_
{
    DB_LED_S_NORMAL = 0x00, /**正常读码的时候亮**/
    DB_LED_S_ALWAYS_OFF,    /**一直关闭**/
    DB_LED_S_ALWAYS_ON,     /**一直开启**/

    DB_LED_S_MAX,
}DB_LED_STATE_E;

/**蜂鸣器提示声音类型**/
typedef enum MP_BEEP_TYPE_
{
    /**单个声音**/
    MP_BEEP_T_DEFAULT_PARAM = 0x00, /**使用默认参数**/
    MP_BEEP_T_SETUP_FAIL,           /**设置失败**/
    MP_BEEP_T_MULTI_SETUP,          /**多次设置,如需要用到数字设置**/
    MP_BEEP_T_DECODE_SUCC,          /**解码成功**/
    MP_BEEP_T_ACC_FAIL,             /**意外失败**/
    MP_BEEP_T_FACTORY_PARAM,        /**使用出厂参数**/
    MP_BEEP_T_CUSTOMER_USE_0,       /**客户使用 0**/
    MP_BEEP_T_CUSTOMER_USE_1,       /**客户使用 1**/

    /**对个声音**/
    MP_BEEP_T_START_UP,             /**开机**/
    MP_BEEP_T_SETUP_SUCC,           /**设置成功**/

    /**语音**/
    MP_BEEP_T_NOTSUPPORT_PAY_CODE, /**不支持的支付码,商米 2G 模块使用**/
    MP_BEEP_T_PAY_WEIXIN,           /**欢迎使用微信支付**/

```

```

        MP_BEEP_T_PAY_ZHIFUBAO,          /**欢迎使用支付宝**/

        MP_BEEP_T_MAX,
    }MP_BEEP_TYPE_E;

typedef enum DB_NFC_OUT_TYPE_
{
    DB_NFC_OUT_T_RAW = 0x00,    /**NFC 输出：0.十进制原始卡号：收到 0x7B,输出
123(0x31 0x32 0x33)**/
    DB_NFC_OUT_T_DEC_INV_DEC,    /**NFC 输出：1.十进制卡号倒置：123->321**/
    DB_NFC_OUT_T_HEX_INV_DEC,    /**NFC 输出：2.大小端翻转并以十进制卡号输出：
123456(0x1E240)->4252161(0x40E201)**/
    DB_NFC_OUT_T_HEX_CHAR,      /**NFC 输出：3.以十六进制字符串输出：
123456(0x1E240)->01E240**/
    DB_NFC_OUT_T_HEX_INV_HEX,    /**NFC 输出：4.以十六进制字符串倒置输出：
123456(0x1E240)->40E201**/

    DB_NFC_OUT_T_MAX,
}DB_NFC_OUT_TYPE_E;

/**解码数据和设备状态回调函数**/
public interface TsDataCallBack{
    int ts_get_data_fun(byte[] pParam,byte[] pbuf,int uiBufLen);
}

public interface TsStateCallBack{
    int ts_scan_state_fun(byte[] pParam,byte ucState);
}

/*****
*   函数名称: ts_scan_init
*   功能描述: 模块初始化
*   参数说明:
*       输入:
*       输出: 无
*   返回值:   int,成功:SUCC 失败:FAIL
*****/
public final int ts_scan_init();

/*****
*   函数名称: ts_scan_deinit
*   功能描述: 模块去初始化
*   参数说明:
*       输入:

```

```

*   输出: 无
*   返回值:  int,成功:Succ 失败:Fail
*****/

public final int ts_scan_deinit();

/*****
*   函数名称: ts_scan_get_version
*   功能描述: 获取该模块版本号
*   参数说明:
*       输入: 无
*       输出: 无
*   返回值:  String,版本字符串口
*****/

public final String ts_scan_get_version();

/*****
*   函数名称: ts_scan_get_product_type
*   功能描述: 获取模块类别
*   参数说明:
*       输入: 无
*       输出: 无
*   返回值:  byte,模块类别
*****/

public final byte ts_scan_get_product_type();

/*****
*   函数名称: ts_scan_uart_findPorts
*   功能描述: 获取串口号   uart sdk 才可以使用
*   参数说明:
*
*       输入: 无
*       输出: 无
*   返回值:  ArrayList<String>   串口链表
*****/

public final ArrayList<String> ts_scan_uart_findPorts()

/*****
*   函数名称: ts_scan_uart_open
*   功能描述: 打开串口   uart sdk 才可以使用
*   参数说明:  usUartNum  串口号
*               uiBaudrate 波特率
*       输入: 无
*       输出: 无
*   返回值:  int,成功:Succ 失败:Fail

```

```

*****/
public final int ts_scan_uart_open(String usUartNum, int uiBaudrate);

/*****
*   函数名称: ts_scan_uart_close
*   功能描述: 关闭串口  uart sdk 才可以使用
*   参数说明:
*       输入: 无
*       输出: 无
*   返回值:  int,成功:SUCC 失败:FAIL
*****/
public final int ts_scan_uart_close();

/*****
*   函数名称: ts_scan_get_uart_state
*   功能描述: 获取串口状态  uart sdk 才可以使用
*   参数说明:
*       输入: 无
*       输出: 无
*   返回值:  int,打开:TRUE 关闭:FALSE
*****/
public final int ts_scan_get_uart_state();

/*****
*   函数名称: ts_scan_get_data_fun_register
*   功能描述: 注册扫描数据上报函数
*   参数说明:
*       输入: ts_scan_get_data_fun 解码数据回调函数
*       输出: 无
*   返回值: int,成功:SUCC 失败:FAIL
*****/
public final int ts_scan_get_data_fun_register(byte[] pParam, TsDataCallBack fGetDataFun)

/*****
*   函数名称: ts_scan_state_fun_register
*   功能描述: 注册状态上报函数
*   参数说明:
*       输入: ts_scan_state_fun 解码数据回调函数
*       输出: 无
*   返回值: int,成功:SUCC 失败:FAIL
*****/
public final int ts_scan_state_fun_register(byte[] pParam, TsStateCallBack fStateFun)

/*****

```

```

* 函数名称: ts_scan_decode_start
* 功能描述: 开始解码
* 参数说明:
*   输入:
*
*   输出: 无
* 返回值:int,成功:Succ   失败:FAIL
*****/
public final int ts_scan_decode_start();

/*****
* 函数名称: ts_scan_decode_stop
* 功能描述: 停止解码
* 参数说明:
*   输入:
*
*   输出: 无
* 返回值:int,成功:Succ   失败:FAIL
*****/
public final int ts_scan_decode_stop();

/*****
* 函数名称: ts_scan_set_rs485_net_mode
* 功能描述: 设置 RS485 组网模式 uart sdk 才可以使用
* 参数说明:      ucmode 1 为组网模式, 0 为普通模式
*   输入:
*
*   输出: 无
* 返回值:int,成功:Succ   失败:FAIL
*****/
public final int ts_scan_set_rs485_net_mode(byte ucmode);

/*****
* 函数名称: ts_scan_get_fixe_rn_add_decode_data
* 功能描述: 获取从机地址:xxxx 的解码数据 uart sdk 才可以使用
* 参数说明:      usrndevicadd 从机地址
*   输入:
*
*   输出: 无
* 返回值:int,成功:Succ   失败:FAIL
*****/
public final int ts_scan_get_fixe_rn_add_decode_data(int usrndevicadd);

```

```

/*****
*   函数名称: ts_scan_get_heartbeat_cnt
*   功能描述: 获取心跳包次数
*   参数说明:
*       输入:
*
*       输出: 无
*   返回值:int,心跳包次数
*****/
public final int ts_scan_get_heartbeat_cnt();

```

```

/*****
*   函数名称: ts_scan_get_en_sign
*   功能描述: 获取签名开/关
*   参数说明:
*       输入:
*
*       输出: 无
*   返回值: boolean,开:TRUE          关:FALSE
*****/
public final boolean ts_scan_get_en_sign();

```

```

/*****
*   函数名称: ts_scan_set_en_sign
*   功能描述: 设置签名开/关
*   参数说明:
*       输入: uiEnSign 开:TRUE          关:FALSE
*
*       输出: 无
*   返回值:Succ
*****/
public final int ts_scan_set_en_sign(boolean uiEnSign);

```

```

/*****
*   函数名称: ts_scan_get_sign_auth_is_succ
*   功能描述: 获取签名状态
*   参数说明:
*       输入:
*
*       输出: 无
*   返回值: true 已签名, false 未签名
*****/
public final boolean ts_scan_get_sign_auth_is_succ();

```

```

/*****
* 函数名称: ts_scan_sw
* 功能描述: 扫描开关
* 参数说明:
*   输入: ucEn,0:禁止扫描,1:使能扫描
*   输出: 无
*   返回值: int,成功:Succ 失败:Fail
*****/

public final int ts_scan_sw(byte ucEn);

/*****
* 函数名称: ts_scan_get_device_state
* 功能描述: 查询设备设备连接正常
* 参数说明:
*
*   返回值:正常:Succ 断开:Fail
*****/

public final int ts_scan_get_device_state();

/*****
* 函数名称: ts_scan_get_product_name
* 功能描述: 获取产品名称
* 参数说明:
*   输入:usLen:产品名称长度
*   输出:pName:产品名称
*   返回值:int,实际获取的长度
*****/

public final int ts_scan_get_product_name(byte[] pName, int usLen);

/*****
* 函数名称: ts_scan_get_product_id
* 功能描述: 获取产品编号,范围:0-15
* 参数说明:
*   输入:usLen:产品编号长度
*   输出:byte:pSn
*   返回值:int,实际获取的长度
*****/

public final int ts_scan_get_product_id(byte[] pSn, int usLen);

/*****
* 函数名称: ts_scan_get_product_date
* 功能描述: 获取出厂日期
* 参数说明:
*   输入:usLen:出厂日期长度

```



```

*   输出:pdata:出厂日期
*   返回值:int,实际获取的长度
*****/
public final int ts_scan_get_product_date(byte[] pdata, int usLen);

/*****
*   函数名称: ts_scan_get_custom_name
*   功能描述: 获取定制名称
*   参数说明:
*       输入:usLen:定制名称长度
*       输出:pName:定制名称
*   返回值:int,实际获取的长度
*****/
public final int ts_scan_get_custom_name(byte[] pName, int usLen);

/*****
*   函数名称: ts_scan_get_custom_id
*   功能描述: 获取定制编号
*   参数说明:
*       输入:usLen:定制编号长度
*       输出:pSn:定制编号
*   返回值:int,实际获取的长度
*****/
public final int ts_scan_get_custom_id(byte[] pSn, int usLen);

/*****
*   函数名称: ts_scan_get_custom_date
*   功能描述: 获取定制日期
*   参数说明:
*       输入:usLen:定制日期长度
*       输出:pdata:定制日期
*   返回值:int,实际获取的长度
*****/
public final int ts_scan_get_custom_date(byte[] pdata, int usLen);

/*****
*   函数名称: ts_scan_get_firmware_version
*   功能描述: 获取软件版本
*   参数说明:
*       输入:usLen:软件版本长度
*       输出:pVer:软件版本
*   返回值:int,实际获取的长度
*****/
public final int ts_scan_get_firmware_version(byte[] pVer, int usLen);

```

```

/*****
* 函数名称: ts_scan_get_hardware_version
* 功能描述: 获取硬件版本
* 参数说明:
*     输入:usLen:硬件版本长度
*     输出:phver:硬件版本
* 返回值:int,实际获取的长度
*****/
public final int ts_scan_get_hardware_version(byte[] phver, int usLen);

```

```

/*****
* 函数名称: ts_scan_get_complex_sn
* 功能描述: 获取复合产品序列号,范围:0-30(产品名称+序列号)
* 参数说明:
*     输入:usLen:产品序列号长度
*     输出:byte:pCplSn
* 返回值:int,实际获取的长度 ,负数错误
*****/
public final int ts_scan_get_complex_sn(byte[] pCplSn, int usLen);

```

```

/*****
* 函数名称: ts_scan_set_trigger_mode
* 功能描述: 设置触发方式
* 参数说明:
*     输入:ucMode:DB_TRIGGER_MODE_E
*           isTemp:0:永久参数,1:临时参数
*     输出:
* 返回值:int,实际获取的长度 ,负数错误
*****/
public final int ts_scan_set_trigger_mode(byte ucMode, byte isTemp);

```

```

/*****
* 函数名称: ts_scan_get_trigger_mode
* 功能描述: 获取触发方式
* 参数说明:
*     输入:
*
*     输出:
* 返回值:失败:FAIL 成功:DB_TRIGGER_MODE_E
*****/
public final int ts_scan_get_trigger_mode();

```

```

/*****

```

```

* 函数名称: ts_scan_set_comm_mode
* 功能描述: 设置通信方式
* 参数说明:
*   输入:ucMode:DB_COMM_MODE_E
*         isTemp:0:永久参数,1:临时参数
*   输出:
*   返回值:int,实际获取的长度 ,负数错误
*****/
public final int ts_scan_set_comm_mode(byte ucMode, byte isTemp);

/*****
* 函数名称: ts_scan_get_comm_mode
* 功能描述: 获取通信方式
* 参数说明:
*   输入:
*
*   输出:
*   返回值:失败:FAIL 成功:DB_COMM_MODE_E
*****/
public final int ts_scan_get_comm_mode();

/*****
* 函数名称: ts_scan_kbw_enable
* 功能描述: 使能 KBW
* 参数说明:
*   输入: ucEn 0:禁止,1:使能[单次有效]
*
*   输出:
*   返回值:int,实际获取的长度 ,负数错误
*****/
public final int ts_scan_kbw_enable(byte ucEn);

/*****
* 函数名称: ts_scan_custom_data_write
* 功能描述: 写入存储用户自定义数据(最大支持 2K)
* 参数说明:
*   输入: uiBufLen:写入的长度
*         pBuf:写入的数据
*   输出:
*   返回值:int,实际获取的长度 ,负数错误
*****/
public final int ts_scan_custom_data_write(byte[] pBuf, int uiBufLen);

/*****

```

```

* 函数名称: ts_scan_custom_data_read
* 功能描述: 读出存储用户自定义数据(最大支持 2K)
* 参数说明:
*   输入: uiBufLen:读出的长度
*
*   输出: pBuf:读出的数据
* 返回值:int,实际获取的长度 ,负数错误
*****/
public final int ts_scan_custom_data_read(byte[] pBuf, int uiBufLen);

/*****
* 函数名称: ts_scan_set_light_mode
* 功能描述: 设置照明灯
* 参数说明:
*   输入: ucMode:DB_LED_STATE_E
*         isTemp:0:永久参数,1:临时参数
*   输出:
* 返回值:int,实际获取的长度 ,负数错误
*****/
public final int ts_scan_set_light_mode(byte ucMode, byte isTemp);

/*****
* 函数名称: ts_scan_get_light_mode
* 功能描述: 获取照明灯控制
* 参数说明:
*   输入:
*
*   输出:
* 返回值:成功:DB_LED_STATE_E 失败:FAIL
*****/
public final int ts_scan_get_light_mode();

/*****
* 函数名称: ts_scan_beep_play
* 功能描述: 设置蜂鸣器
* 参数说明:
*   输入: ucMode:0:客户使用 0,1:客户使用 1
*
*   输出:
* 返回值:int,实际获取的长度 ,负数错误
*****/
public final int ts_scan_beep_play(byte ucMode);

```

```

/*****
* 函数名称: ts_scan_play_voice
* 功能描述: 播放语音,格式 WAV,采样频率 16Khz,位宽 16 位,单声道,由于指令的限制最大播放语音 62KB
* 参数说明:
*     输入: pBuf,播放的语音源
*           usLen,长度
*     输出: 无
*     返回值: int,成功:SUCC 失败:FAIL
*****/

public final int ts_scan_play_voice(byte[] pBuf, int16 usLen);

/*****
* 函数名称: ts_scan_set_one_scan_time
* 功能描述: 设置单次扫码时间
* 参数说明:
*     输入: uctime:设置时间 大于 5 小于 255
*           isTemp:0:永久参数,1:临时参数
*     输出:
*     返回值: int,成功:SUCC 失败:FAIL
*****/

public final int ts_scan_set_one_scan_time(byte uctime,byte isTemp);

/*****
* 函数名称: ts_scan_get_one_scan_time
* 功能描述: 获取单次扫码时间
* 参数说明:
*     输入:
*
*     输出:
*     返回值:成功:当前单次扫码时间 失败:FAIL
*****/

public final int ts_scan_get_one_scan_time();

/*****
* 函数名称: ts_scan_set_Prefix
* 功能描述: 设置多个前缀
* 参数说明:
*     输入: pBuf:设置的前缀
*           usLen:长度
*           isTemp:0:永久参数,1:临时参数
*     输出:
*     返回值: int,成功:SUCC 失败:FAIL
*****/

```

```
public final int ts_scan_set_Prefix(byte[] pBuf, int usLen, byte isTemp);
```

```
/******
```

```
* 函数名称: ts_scan_get_Prefix
```

```
* 功能描述: 获取多个前缀
```

```
* 参数说明:
```

```
* 输入: usLen:长度
```

```
*
```

```
*
```

```
* 输出: pBuf:获取的前缀
```

```
* 返回值: int,成功:前缀长度 失败:FAIL
```

```
*****/
```

```
public final int ts_scan_get_Prefix(byte[] pBuf, int usLen);
```

```
/******
```

```
* 函数名称: ts_scan_set_Suffix
```

```
* 功能描述: 设置多个后缀
```

```
* 参数说明:
```

```
* 输入: pBuf:设置的后缀
```

```
* usLen:长度
```

```
* isTemp:0:永久参数,1:临时参数
```

```
* 输出:
```

```
* 返回值: int,成功:Succ 失败:FAIL
```

```
*****/
```

```
public final int ts_scan_set_Suffix(byte[] pBuf, int usLen, byte isTemp);
```

```
/******
```

```
* 函数名称: ts_scan_get_Suffix
```

```
* 功能描述: 获取多个后缀
```

```
* 参数说明:
```

```
* 输入: usLen:长度
```

```
*
```

```
*
```

```
* 输出: pBuf:获取的后缀
```

```
* 返回值: int,成功:后缀长度 失败:FAIL
```

```
*****/
```

```
public final int ts_scan_get_Suffix(byte[] pBuf, int usLen);
```

```
/******
```

```
* 函数名称: ts_scan_set_packet_format
```

```
* 功能描述: 设置包格式
```

```
* 参数说明:
```

```
* 输入: ucMode:0:原始数据,1:数据包
```

```
* isTemp:0:永久参数,1:临时参数
```

```

*   输出:
*   返回值: int32,成功:SUCC(0) 失败:FAIL(-1)
*****/

public final int ts_scan_set_packet_format(byte ucFormat, byte isTemp);

/*****
*   函数名称: ts_scan_get_packet_format
*   功能描述: 获取包格式
*   参数说明:
*   输入:
*
*   输出:
*   返回值: 0:原始数据,1:数据包 失败:FAIL
*****/

public final int ts_scan_get_packet_format();

/*****NFC Start*****/
/*****
*   函数名称: ts_scan_set_nfc_work_mode
*   功能描述: NFC 工作模式
*   参数说明:
*   输入: ucMode:0:自动模式,1:指令模式
*          isTemp:0:永久参数,1:临时参数
*   输出:
*   返回值: int,成功:SUCC(0) 失败:FAIL(-1)
*****/

public final int ts_scan_set_nfc_work_mode(byte ucMode, byte isTemp);

/*****
*   函数名称: ts_scan_get_nfc_work_mode
*   功能描述: 获取 NFC 工作模式
*   参数说明:
*   输入:
*
*   输出:
*   返回值: 0:自动模式,1:指令模式 失败:FAIL
*****/

public final int ts_scan_get_nfc_work_mode();

/*****
*   函数名称: ts_scan_set_nfc_id_format
*   功能描述: NFC 卡号(ID)输出格式
*   参数说明:

```

```

*   输入: ucMode:DB_NFC_OUT_TYPE_E
*           isTemp:0:永久参数,1:临时参数
*   输出:
*   返回值: int,成功:SUC(0) 失败:FAIL(-1)
*****/

public final int ts_scan_set_nfc_id_format(byte ucFormat, byte isTemp);

/*****
*   函数名称: ts_scan_get_nfc_id_format
*   功能描述: 获取 NFC 卡号(ID)输出格式
*   参数说明:
*   输入:
*
*   输出:
*   返回值: DB_NFC_OUT_TYPE_E 失败:FAIL
*****/

public final int ts_scan_get_nfc_id_format();

/*****
*   函数名称: ts_scan_set_nfc_data_format
*   功能描述: NFC 扇区/页数据输出格式
*   参数说明:
*   输入: ucMode:0:以 ASCII 方式输出,1:输出原始数据(HEX)
*           isTemp:0:永久参数,1:临时参数
*   输出:
*   返回值: int,成功:SUC(0) 失败:FAIL(-1)
*****/

public final int ts_scan_set_nfc_data_format(byte ucFormat, byte isTemp);

/*****
*   函数名称: ts_scan_get_nfc_data_format
*   功能描述: 获取 NFC 扇区/页数据输出格式
*   参数说明:
*   输入:
*
*   输出:
*   返回值: 0:以 ASCII 方式输出,1:输出原始数据(HEX), 失败:FAIL
*****/

public final int ts_scan_get_nfc_data_format();

/*****
*   函数名称: ts_scan_set_nfc_block_num
*   功能描述: 设置 NFC 块号
*   参数说明:

```



```

*   输入: ucMode:0:块号
*           isTemp:0:永久参数,1:临时参数
*   输出:
*   返回值: int,成功:Succ(0) 失败:Fail(-1)
*****/

public final int ts_scan_set_nfc_block_num(byte ucBlock, byte isTemp);

/*****
*   函数名称: ts_scan_get_nfc_block_num
*   功能描述: 获取 NFC 块号
*   参数说明:
*   输入:
*
*   输出:
*   返回值: NFC 块号 失败:Fail
*****/

public final int ts_scan_get_nfc_block_num();

/*****
*   函数名称: ts_scan_set_nfc_key_type
*   功能描述: NFC 密钥类型
*   参数说明:
*   输入: ucMode: 0:密钥 A,1:密钥 B
*           isTemp:0:永久参数,1:临时参数
*   输出:
*   返回值: int,成功:Succ(0) 失败:Fail(-1)
*****/

public final int ts_scan_set_nfc_key_type(byte ucKeyType, byte isTemp);

/*****
*   函数名称: ts_scan_get_nfc_key_type
*   功能描述: 获取 NFC 密钥类型
*   参数说明:
*   输入:
*
*   输出:
*   返回值: 0:密钥 A,1:密钥 B, 失败:Fail
*****/

public final int ts_scan_get_nfc_key_type();

/*****
*   函数名称: ts_scan_set_nfc_key_a
*   功能描述: 设置 NFC 密钥 A
*   参数说明:

```

```

*   输入: pBuf:设置 NFC 密钥 A
*           usLen:长度(固定设置 6 字节)
*           isTemp:0:永久参数,1:临时参数
*   输出:
*   返回值:  int,成功:SUCC(0)  失败:FAIL(-1)
*****/
public final int ts_scan_set_nfc_key_a(byte[] pBuf, int usLen, byte isTemp);

/*****
*   函数名称: ts_scan_get_nfc_key_a
*   功能描述: 获取 NFC 密钥 A
*   参数说明:
*       输入: usLen:长度(6 字节)
*
*
*   输出:    pBuf:获取的 NFC 密钥 A
*   返回值:  int,成功:NFC 密钥 A 长度  失败:FAIL
*****/
public final int ts_scan_get_nfc_key_a(byte[] pBuf, int usLen);

/*****
*   函数名称: ts_scan_set_nfc_key_b
*   功能描述: 设置 NFC 密钥 B
*   参数说明:
*       输入: pBuf:设置 NFC 密钥 B
*           usLen:长度(固定设置 6 字节)
*           isTemp:0:永久参数,1:临时参数
*   输出:
*   返回值:  int,成功:SUCC(0)  失败:FAIL(-1)
*****/
public final int ts_scan_set_nfc_key_b(byte[] pBuf, int usLen, byte isTemp);

/*****
*   函数名称: ts_scan_get_nfc_key_b
*   功能描述: 获取 NFC 密钥 B
*   参数说明:
*       输入: usLen:长度(6 字节)
*
*
*   输出:    pBuf:获取的 NFC 密钥 B
*   返回值:  int,成功:NFC 密钥 B 长度  失败:FAIL
*****/
public final int ts_scan_get_nfc_key_b(byte[] pBuf, int usLen);

```

```

/*****
* 函数名称: ts_scan_nfc_cmd_read_data
* 功能描述: NFC 指令读 ID/扇区
* 参数说明:
*   输入: ucReadType: 0:只读 ID,1:只读扇区,2:读 ID+扇区
*           pBuf: 读到的数据,格式[Type]+[IdLen]+IdData+[BlockLen]+BlockData
*           uiBufLen: 缓存区长度/期望读出数据的长度,
*   输出:
*   返回值: int,实际获取的长度 ,负数错误
*****/

public final int ts_scan_nfc_cmd_read_data(byte ucReadType, byte[] pBuf, int uiBufLen);

/**NFC 写扇区数据:pBuf:写入的数据,uiBufLen:写入数据的长度(16 字节) **/
/*****
* 函数名称: ts_scan_nfc_block_data_write
* 功能描述: 设置 扇区(块)数据
* 参数说明:
*   输入: pBuf:设置扇区(块)数据
*           usLen:长度(固定设置 16 字节)
*   输出:
*   返回值: int,成功:SUCC(0) 失败:FAIL(-1)
*****/

public final int ts_scan_nfc_block_data_write(byte[] pBuf, int uiBufLen);

/*****
* 函数名称: ts_scan_set_nfc_en_write_ctrl
* 功能描述: 设置 NFC 对控制块的写入权限
* 参数说明:
*   输入: ucMode: 0:禁止写入,1:允许写入
*           isTemp:0:永久参数,1:临时参数
*   输出:
*   返回值: int,成功:SUCC(0) 失败:FAIL(-1)
*****/

public final int ts_scan_set_nfc_en_write_ctrl(byte ucEnable, byte isTemp);

/*****
* 函数名称: ts_scan_get_nfc_en_write_ctrl
* 功能描述: 获取 NFC 对控制块的写入权限
* 参数说明:
*   输入:
*   输出:
*   返回值: 0:禁止写入,1:允许写入, 失败:-1 FAIL
*****/

```

```

public final int ts_scan_get_nfc_en_write_ctrl();

/*****
* 函数名称: ts_scan_nfc_change_mf1_key
* 功能描述: 修改 Mifare S50/S70 卡密钥
* 参数说明:
*   输入: ucKeyType: 0:修改密钥 A, 1:修改密钥 B 2:同时修改密钥 AB
*         pBuf: 修改后的密钥
*         uiBufLen: 密钥的长度(固定是 6|12),
*   输出:
*   返回值: int,成功:SUCC(0) 失败:FAIL(-1)
*****/
public final int ts_scan_nfc_change_mf1_key(byte ucKeyType, byte[] pBuf, int uiBufLen);

/*****NFC End*****/

/*****
* 函数名称: ts_scan_decode_status_request
* 功能描述: 请求扫码器当前状态
* 参数说明:
*   输入: void
*
*   输出:
*   返回值: int, -1 失败,0:空闲, 1:解码中
*****/
public final int ts_scan_decode_status_request();

```