蓝牙 NDK 接口定义

版本	日期	内容说明	修改
V0.1	2014.02.11	初步版本	涂志广
V0.2	2014.07.28	添加 NDK_BTSetLocalMAC	陈镇江
V0.3	2014.9.9	取消 NDK_BTEnterCommand NDK_BTExitCommand 这 2 个函数的功能，为应用兼容，新蓝牙模块驱动这 2 个函数永远返回 NDK_OK；其他兼容	李家泉
V0.4	20140923	添加 NDK_BTSetLocalMAC 输入参数要求	陈镇江
V0.5	20141107	新增 2 个 NDK 接口： int NDK_BTGetPairingStatus(char *pszKey, int *pnStatus); int NDK_BTConfirmPairing (const char *pszKey, uint unConfirm);	李家泉
V0.6	20141110	统一蓝牙四种配对模式的命名如下： 0：JustWork 模式（原名：Random PIN，因该名字与第三种配对模式容易引起误解，故以后改名 Just Work）； 1： PINCode 模式（pos 有固定的 PIN Code,配对时手机输入该 PIN Code）； 2：SSP 模式（pos 和手机显示同样的随机的 6 位数字，两端都选择配对或取消）； 3: PassKey 模式（手机显示随机的 6 位数字，pos 输入对应的数字进行配对）。	陈镇江
V1.0	20141110	根据实际情况，修改完善各个 NDK 接口的说明 追加 NDK_BTDisconnect 接口说明	涂志广 陈镇江
V1.1	20141117	根据 20141112 评审修改文档内容	涂志广 陈镇江
V1.2	20141118	修改版本更新 V0.5 里写的名称	陈镇江
V1.3	20141127	1. 更改随机码配对模式部分错误的名称 2. Bm77 不受 NDK_PortOpen 的 NDK 接口追加 **NDK_BTDisconnect** 3. NDK_BTGetPairingStatus 返回值说明完善	陈镇江

版本	日期	内容说明	修改
V1.4	20150120	1. AP6210B 蓝牙方案的蓝牙初始化放在 NDK_PortOpen 里面，应用在开机后第一次使用蓝牙前需要先调用 NDK_PortOpen。 2. NDK_BTSetLocalName 如果设置中文名称要求使用 UTF8 格式，AP6210B 蓝牙模块支持名称长度改为最大 23 字节，bm77 超过 23 字节的设置则截取前面 15 字节进行设置。 3. 新增 NDK_BTSetDiscoverableStatus 的 NDK 接口	陈镇江

对于中端平台:

1. 所有蓝牙相关的 NDK 接口调用之前都要先调用 NDK_PortOpen，否则就返回 NDK_ERR_OPEN_DEV 错误。

2. 所有功能接口没有 NDK_ERR_IOCTL 和 NDK_ERR_TIMEOUT 这两个返回值

对于低端平台:

NDK_PortOpen 对不同方案的接口调用影响

BM77 方案:

a. 影响接口

1	int NDK_BTGetLocalName(char * name);
2	int NDK_BTGetLocalMAC(char * mac);
3	int NDK_BTGetPIN(char * pinstr);
4	int NDK_BTSetPIN(const char * pinstr);
5	int NDK_BTSetPairingMode(EM_PAIRING_MODE emMode);
6	int NDK_BTGetPairingStatus(char * pszKey, int *pnStatus)
7	int NDK_BTConfirmPairing (const char * pszKey, uint unConfirm);
8	NDK_Port 操作的相关函数 比如，NDK_PortWrite, NDK_PortRead 等

b. 不影响接口

1	int NDK_BTReset(void);
2	int NDK_BTStatus(int * status);
3	int NDK_BTEnterCommand(void);
4	int NDK_BTExitCommand(void);
5	int NDK_BTDisconnect(void)

AP6210B 方案

a. 影响接口:所有的蓝牙 NDK 接口。

强烈建议: 使用蓝牙模块前，首先调用 NDK_PortOpen()， 并且 NDK_PortOpen(), NDK_PortClose()成对使用。

```
/** @addtogroup  蓝牙
* @{
*/
/**
  *@brief 蓝牙模块软件复位
  *@return
  *@li     NDK_OK                          操作成功
  *@li     \ref NDK_ERR "NDK_ERR"        操作失败
  *@li     \ref NDK_ERR_OPEN_DEV " NDK_ERR_OPEN_DEV"         设备未打开
*/
int NDK_BTReset(void);

/**
```

```
 *@brief 设置本机蓝牙模块名称
 *@param     pszName 设定的名称(中文名称必需使用 UTF8 格式)，对于 bm77 方案如果名称长度大于 15 字节小于 23 字节，则截取前面 15 字节为有效名称。
 *@return
 *@li     NDK_OK                    操作成功
 *@li     \ref NDK_ERR_PARA "NDK_ERR_PARA"          参数非法(pszName 为 NULL 或者名称长度大于 23 个字节)
 *@li     \ref NDK_ERR "NDK_ERR"      操作失败
 *@li     \ref NDK_ERR_OPEN_DEV"NDK_ERR_OPEN_DEV"       设备未打开
 *@li     \ref NDK_ERR_TIMEOUT” NDK_ERR_TIMEOUT “     操作超时
*/
int NDK_BTSetLocalName(const char *pszName);

/**
 *@brief 获取本机蓝牙模块名称
 *@retval     pszName 获取的名称
 *@return
 *@li     NDK_OK                    操作成功
 *@li     \ref NDK_ERR_PARA "NDK_ERR_PARA"          参数非法(pszName 为 NULL)
 *@li     \ref NDK_ERR "NDK_ERR"      操作失败
 *@li     \ref NDK_ERR_OPEN_DEV"NDK_ERR_OPEN_DEV"       设备未打开
 *@li     \ref NDK_ERR_TIMEOUT” NDK_ERR_TIMEOUT “     操作超时
*/
int NDK_BTGetLocalName(char *pszName);

/**
 *@brief 设置本机蓝牙模块 PIN 码
 *@param     pszPinCode    设定的 PIN 码
 *@return
 *@li     NDK_OK                    操作成功
 *@li     \ref NDK_ERR_PARA "NDK_ERR_PARA"          参数非法(pszPinCode 为 NULL)
 *@li     \ref NDK_ERR "NDK_ERR"      操作失败
 *@li     \ref NDK_ERR_OPEN_DEV"NDK_ERR_OPEN_DEV"       设备未打开
 *@li     \ref NDK_ERR_TIMEOUT” NDK_ERR_TIMEOUT “     操作超时
*/
int NDK_BTSetPIN(const char *pszPinCode);

/**
 *@brief 获取本机蓝牙模块 PIN 码
 *@retval     pszPinCode    获取的 PIN 码
 *@return
 *@li     NDK_OK                    操作成功
 *@li     \ref NDK_ERR_PARA "NDK_ERR_PARA"          参数非法(pszPinCode 为 NULL)
 *@li     \ref NDK_ERR "NDK_ERR"      操作失败
```

```
 *@li     \ref NDK_ERR_OPEN_DEV"NDK_ERR_OPEN_DEV"          设备未打开
 *@li     \ref NDK_ERR_TIMEOUT” NDK_ERR_TIMEOUT “      操作超时
*/
int NDK_BTGetPIN(char *pszPinCode);


/**
 *@brief 获取本机蓝牙模块 MAC 地址
 *@retval      pszMac   获取的蓝牙的 MAC 地址
 *@return
 *@li     NDK_OK                          操作成功
 *@li     \ref NDK_ERR_PARA "NDK_ERR_PARA"            参数非法(pszMac 为 NULL)
 *@li     \ref NDK_ERR "NDK_ERR"       操作失败
 *@li     \ref NDK_ERR_OPEN_DEV"NDK_ERR_OPEN_DEV"          设备未打开
 *@li     \ref NDK_ERR_TIMEOUT” NDK_ERR_TIMEOUT “      操作超时
*/
int NDK_BTGetLocalMAC(char *pszMac);

/**
 *@brief 获取的蓝牙连接状态   0 ——连接状态              1——未连接状态
 *@retval      pnStatus  获取的蓝牙状态
 *@return
 *@li     NDK_OK                          操作成功
 *@li     \ref NDK_ERR_PARA "NDK_ERR_PARA"            参数非法(pnStatus 为 NULL)
 *@li     \ref NDK_ERR "NDK_ERR"       操作失败
 *@li     \ref NDK_ERR_OPEN_DEV"NDK_ERR_OPEN_DEV"          设备未打开
*/
int NDK_BTStatus(int *pnStatus);

/**
 *@brief 蓝牙模块进入命令模式(只有 bm77 方案有此功能，AP6210B 方案直接全部返回
NDK_OK)
 *@return
 *@li     NDK_OK                          操作成功
 *@li     \ref NDK_ERR "NDK_ERR"       操作失败
 *@li     \ref NDK_ERR_OPEN_DEV"NDK_ERR_OPEN_DEV"          设备未打开
*/
int NDK_BTEnterCommand(void);

/**
 *@brief 蓝牙模块退出命令模式(只有 bm77 方案有此功能，AP6210B 方案直接全部返回
NDK_OK)
*@return
 *@li     NDK_OK                          操作成功
```

*@li \ref NDK_ERR "NDK_ERR"操作失败
*@li \ref NDK_ERR_OPEN_DEV"NDK_ERR_OPEN_DEV" 设备未打开
*/
int NDK_BTExitCommand(void);

/**
*@brief 设置本机蓝牙模块 MAC 地址
*@param pszMac 设定的蓝牙 MAC 地址，MAC 地址长度必需为 6 个字节。如参数字节数偏大或偏小会导致错误的 MAC 地址被设置。
*@return
*@li NDK_OK 操作成功
*@li \ref NDK_ERR_PARA "NDK_ERR_PARA" 参数非法(pszMac 为 NULL)
*@li \ref NDK_ERR "NDK_ERR"操作失败
*@li \ref NDK_ERR_OPEN_DEV "NDK_ERR_OPEN_DEV" 设备未打开
对于 bm77 方案：调用此接口总是返回 NDK_ERR_NOT_SUPPORT。
对于 AP6210B 方案：此接口用于生产时设置产品的 MAC 地址。
*/
int NDK_BTSetLocalMAC(const char *pszMac);

/**
*@brief 断开当前连接
*@return
*@li NDK_OK 操作成功
*@li \ref NDK_ERR "NDK_ERR" 操作失败
*@li \ref NDK_ERR_OPEN_DEV "NDK_ERR_OPEN_DEV" 设备未打开
*/
int NDK_BTDisconnect(void)

/*
*设置蓝牙配对模式：
* PAIRING_MODE 及解释 ：
typedef enum{
PAIRING_MODE_JUSTWORK = 0, /**<Just Work 模式*/
PAIRING_MODE_PINCODE= 1, /**<PinCode 模式*/
PAIRING_MODE_SSP= 2, /**<SSP 模式*/
PAIRING_MODE_PASSKEY= 3, /**<PassKey 模式*/
}EM_PAIRING_MODE;
0：JustWork 模式（原名：Random PIN，因该名字与第三种配对模式容易引起误解，故以后改名 JustWork）；
1：PINCode 模式（pos 有固定的 PIN Code,配对时手机输入该 PIN Code）；
2：SSP 模式（pos 和手机显示同样的随机的 6 位数字，两端都选择配对或取消）；
3: PassKey 模式（手机显示随机的 6 位数字，pos 输入对应的数字进行配对）。
*@return
*@liNDK_OK 操作成功

*@ li \ref NDK_ERR_PARA 传入非 EM_PAIRING_MODE 的参数，返回 "NDK_ERR_PARA"
*@li \ref NDK_ERR "NDK_ERR" 操作失败
*@ li \ref NDK_ERR_IOCTL "NDK_ERR_IOCTL" 驱动接口调用错误
*@li\ref NDK_ERR_OPEN_DEV"NDK_ERR_OPEN_DEV" 设备未打开
*/
NDK_BTSetPairingMode(EM_PAIRING_MODE emMode)

/**
*@brief 获取蓝牙配对状态(只对配对模式 SSP PIN 和 PassKey 有效)
说明： 目前 bm77 方案暂不支持此功能，AP6210B 方案支持此功能
*@retval pszKey: a)SSP 模式：pszKey 返回手机上显示的配对码；
* b)PassKey 模式：pszKey[0]返回'\0'，表明收到手机配对请求。
*@retval pnStatus : 1:收到手机配对请求；2: 配对成功；3 配对失败；0：其他状态
*@return
*@li NDK_OK 操作成功
*@li \ref NDK_ERR_PARA pszKey 或者 pnStatus 为 NULL 返回 "NDK_ERR_PARA"
*@li \ref NDK_ERR "NDK_ERR" 操作失败
*@li \ref NDK_ERR_IOCTL "NDK_ERR_IOCTL" 驱动接口调用错误
*@li \ref NDK_ERR_OPEN_DEV"NDK_ERR_OPEN_DEV" 设备未打开
对于 bm77 方案：调用此接口总是返回 NDK_ERR_NOT_SUPPORT。
*/
int NDK_BTGetPairingStatus(char * pszKey, int *pnStatus);

/**
*@brief 蓝牙配对确认
*@param pszKey： a)SSP 模式：pszKey 设置为 NDK_BTGetParingStatus()获取到的 key；
* b)PassKey 模式：pszKey 为键盘输入的 key。
*@param unConfirm : 0:取消配对；1：接受配对
*@return
*@li NDK_OK 操作成功
*@li \ref NDK_ERR_PARA pszKey 为 NULL 返回 "NDK_ERR_PARA"
*@li \ref NDK_ERR "NDK_ERR" 操作失败
*@li \ref NDK_ERR_IOCTL "NDK_ERR_IOCTL" 驱动接口调用错误
*@li \ref NDK_ERR_OPEN_DEV"NDK_ERR_OPEN_DEV" 设备未打开
对于 bm77 方案：调用此接口总是返回 NDK_ERR_NOT_SUPPORT。
*/
int NDK_BTConfirmPairing(const char * pszKey, uint unConfirm);

/*
*@param status： status 设置为 1 表示蓝牙可以被搜索到，系统默认 status 为 1；status 设置为 0 表示蓝牙不可以被搜索到，主要用于休眠模式；
*说明： 该函数必需在非连接情况下调用，在连接状态下调用会返回 NDK_ERR_IOCTL。
*@return

*@li NDK_OK   操作成功
*@ li   \ref NDK_ERR_PARA"NDK_ERR_PARA"   传入非 0 且非 1 的参数返回此错误
*@ li   \ref NDK_ERR_IOCTL "NDK_ERR_IOCTL"   驱动接口调用错误
*@li\ref NDK_ERR_OPEN_DEV"NDK_ERR_OPEN_DEV"   设备未打开
*@li\ref NDK_ERR_NOT_SUPPORT" NDK_ERR_NOT_SUPPORT "   不支持此接口
*/
NDK_BTSetDiscoverableStatus (const char status)
/** @} */ //蓝牙模块结束

使用 Demo

SSP 配对模式及 PassKey 配对模式使用用例

```
static void Bt_Pairing_Test(void)
{
    char PassKey[7] = { '\0' };
    int ret, y, key, link_status, flag;
    uint len, accept = 1;
    char *pKey = PassKey;

    NDK_ScrClrs();
    DrawTitles(0, 0, lcd_w, font_h + 2, bt_prompt[12][g_language], 1);
    y = font_h + 4;

    NDK_BTExitCommand();
    RETRY: NDK_ScrDispString(0, y, "打开手机蓝牙和 POS 配对！", 0);
    while (1) {
        NDK_BTStatus(&link_status);
        if (!link_status)
            NDK_ScrDispString(0, y, "蓝牙已连接，请先断开蓝牙！", 0);
        else {
            ret = NDK_BTGetPairingStatus(pKey, &flag);
            if(ret == NDK_ERR_NOT_SUPPORT){
                NDK_ScrClrLine(y, lcd_h);
                NDK_ScrDispString(0, y, "蓝牙模块不支持！", 0);
                NDK_KbGetCode(0, &key);
                goto OUT;
            }
            if (ret == NDK_ERR) {
                NDK_ScrDispString(0, y, "配对模式错误!请修改为 SSP PIN 或者 PassKey!
", 0);
                NDK_KbGetCode(0, &key);
                if (key == K_ESC)
                    goto OUT;
                NDK_ScrClrLine(y, lcd_h);
                goto RETRY;
            }
            if (ret == NDK_OK) {
                if (flag == 1) {
                    NDK_ScrClrLine(y, lcd_h);
                    if (*pKey == '\0') {          // PassKey mode
                        NDK_ScrDispString(0, y, "输入手机显示的 PIN：", 0);
                        NDK_ScrGotoxy(0, y + font_h);
```

```
                    if ((NDK_KbGetInput(PassKey, 0, 6, &len,
                            INPUTDISP_PASSWD,                    0,
INPUT_CONTRL_LIMIT_ERETURN)) < 0) {
                        accept = 0;
                    }
                } else {          //ssp mode
                    NDK_ScrDispString(0, y, "确认配对？ ", 0);
                    NDK_ScrDispString(0, y + font_h, PassKey, 0);
                    NDK_KbGetCode(0, &key);
                    if (key == K_ESC){
                        accept = 0;
                    }
                }
                NDK_ScrClrLine(y, lcd_h);
                if (accept)
                    NDK_ScrDispString(0, y, "正在配对...", 0);
                else
                    NDK_ScrDispString(0, y, "配对取消!", 0);
                y += font_h;
                NDK_BTConfirmPairing (PassKey, accept);
            }
            if (flag == 2) {
                NDK_ScrDispString(0, y, "paring success!", 0);
                NDK_KbGetCode(10, &key);
                goto OUT;
            }
            if (flag == 3) {
                NDK_ScrDispString(0, y, "paring failed!", 0);
                NDK_KbGetCode(10, &key);
                goto OUT;
            }
        }
    }
    NDK_KbGetCode(1, &key);
    if (key == K_ESC)
        goto OUT;
  }
OUT:
  NDK_BTEnterCommand();
  return;
}
```