**K22FSH开发包**

***编程手册***

版本历史

|  |  |  |
| --- | --- | --- |
| 版本 | 日期 | 描述 |
| 草稿 | 2014/12 | 草稿 |

目录

[1 介绍 3](#_Toc406490221)

[2 WIFI网络 4](#_Toc406490222)

[3 AllJoyn瘦客户端 4](#_Toc406490223)

[4 MQX RTOS 4](#_Toc406490224)

[5 FAT文件系统 4](#_Toc406490225)

[6 USB驱动 4](#_Toc406490226)

[7 74HC595 驱动 5](#_Toc406490227)

[7.1 概述 5](#_Toc406490228)

[7.2 配置宏定义 5](#_Toc406490229)

[7.3 代码路径 6](#_Toc406490230)

[7.4 头文件 6](#_Toc406490231)

[7.5 init/deinit 6](#_Toc406490232)

[7.5.1 mux\_74hc595\_init 6](#_Toc406490233)

[7.5.2 mux\_74hc595\_deinit 7](#_Toc406490234)

[7.6 Set/Clear 7](#_Toc406490235)

[7.6.1 mux\_74hc595\_set\_bit 7](#_Toc406490236)

[7.6.2 mux\_74hc595\_clear\_bit 8](#_Toc406490237)

[8 LCD 驱动 9](#_Toc406490238)

[8.2 概述 9](#_Toc406490239)

[8.2 代码路径 9](#_Toc406490240)

[8.3 头文件 10](#_Toc406490241)

[8.4 printf\_lcd 10](#_Toc406490242)

[9 音频输出适配层 10](#_Toc406490243)

[9.1 概述 10](#_Toc406490244)

[9.2 代码路径 10](#_Toc406490245)

[9.3 头文件 10](#_Toc406490246)

[9.4 init/deinit 10](#_Toc406490247)

[9.4.1 msi\_snd\_init 11](#_Toc406490248)

[9.4.2 msi\_snd\_init\_with\_periodbuffer 11](#_Toc406490249)

[9.4.3 msi\_snd\_deinit 11](#_Toc406490250)

[9.5 msi\_snd\_set\_format 12](#_Toc406490251)

[9.6 msi\_snd\_write 12](#_Toc406490252)

[9.7 mute/unmute 13](#_Toc406490253)

[9.7.1 msi\_snd\_mute 13](#_Toc406490254)

[9.7.2 msi\_snd\_unute 13](#_Toc406490255)

[9.8 vol control 13](#_Toc406490256)

[9.8.1 msi\_snd\_vol\_up 13](#_Toc406490257)

[9.8.2 msi\_snd\_vol\_down 13](#_Toc406490258)

[9.9 msi\_snd\_flush 14](#_Toc406490259)

[10 音频播放中间层 14](#_Toc406490260)

[10.1 概述 14](#_Toc406490261)

[10.2 代码路径 14](#_Toc406490262)

[10.3 头文件 14](#_Toc406490263)

[10.4 playback\_init 15](#_Toc406490264)

[10.5 playback\_setup 15](#_Toc406490265)

[10.6 playback\_cancel 15](#_Toc406490266)

[11 蓝牙协议 16](#_Toc406490267)

[11.1 概述 16](#_Toc406490268)

[11.2 库路径 16](#_Toc406490269)

[11.3 头文件 16](#_Toc406490270)

[11.4 协议栈初始化 16](#_Toc406490271)

[11.5 Bluetooth控制器的初始化 17](#_Toc406490272)

[11.6 Bluetooth 控制器的移除 18](#_Toc406490273)

[11.7 取得，设置Bluetooth 控制器MAC 地址 18](#_Toc406490274)

[11.7.1 ADK\_adkbtLocalMac 18](#_Toc406490275)

[11.7.2 ADKRDAChangeBDAddr 18](#_Toc406490276)

[11.8 ADK\_btEnable 19](#_Toc406490277)

[11.9 ADK\_btSetLocalName 19](#_Toc406490278)

[11.10 ADK\_btDiscoverable 19](#_Toc406490279)

[11.11 ADK\_btConnectable 20](#_Toc406490280)

[11.12 ADK\_btSetDeviceClass 20](#_Toc406490281)

[11.13 AVRCP Profile 21](#_Toc406490282)

[11.13.1 avrcpServicePassthroughHandle 21](#_Toc406490283)

[11.14 HFP Profile 22](#_Toc406490284)

[11.14.1 AT 命令 23](#_Toc406490285)

[11.4.2 Indicator 状态改变回调函数 24](#_Toc406490286)

[11.4.3 语音SCO 输出接口 25](#_Toc406490287)

# 1 介绍

K22FSH 开发包软件基于Freescale RTOS MQX4.1.0， 此开发包包含了WIFI 网络，AllJoyn瘦客户端，音频播放，音频解码，蓝牙协议栈，FAT文件系统，Android Open Accessory 协议，LCD驱动， USB驱动等功能。

WIFI网络和AllJoyn瘦客户端的编程本文档直接参考Qualcomm开发套件中的文档。

Qualcomm 开发套件的下载请参考《K22FSH开发包-安装配置手册》 **2.2 安装MQX4.1.0 RTOS** **QCA4002驱动**

这个开发套件的下载文件名是pdk3.0.2-141.zip， 本文档引用了Qualcomm开发套件里的文档如下：

Software\SP140\_SP141\_Demo\_User\_Guide.pdf

Software\ QCA4002\_QCA4004\_PG.pdf

MQX,FAT文件系统,USB驱动开发编程文档本文档直接引用MQX4.1.0 开发包中的文档。这些文档在完成 《K22FSH开发包-安装配置手册》**2.2.1 安装Freescale MQX 4.1.0 操作系统** 后在MQX 4.1.0安装目录的doc目录下可以找到。

# 2 WIFI网络

请参考

QCA4002\_QCA4004\_PG.pdf

如果 mqx/source/io/enet/atheros\_wifi/custom\_src/include/a\_config.h

宏 ENABLE\_STACK\_OFFLOAD 定义成 0，如下

#define ENABLE\_STACK\_OFFLOAD 0

throughput\_demo 演示应用的IP网络协议栈会切换到 MQX相关的一个网络组件RTCS

关于RTCS请参考

rtcs\MQX\_RTCS\_User\_Guide.pdf

# 3 AllJoyn瘦客户端

请参考

SP140\_SP141\_Demo\_User\_Guide.pdf **5.1 AllJoyn Service frameworks**

# 4 MQX RTOS

请参考

mqx\MQX\_Reference\_Manual.pdf 介绍了 MQX 操作系统的数据结构和系统函数

mqx\MQX\_IO\_User\_Guide.pdf 介绍了 MQX IO 驱动的函数和用法

# 5 FAT文件系统

请参考 mfs\ MQX\_MFS\_User\_Guide.pdf

# 6 USB驱动

K22FSH开发套件的演示程序只包含了 USB host 方面的应用程序，而K22FSH开发板和驱动也支持 开发板作为USB device。

请参考

USB host

usb\ MQX\_USB\_Host\_User\_Guide.pdf

usb\ MQX\_USB\_Host\_Reference\_Manual.pdf

USB device

usb\ MQX\_USB\_Device\_Reference\_Manual.pdf

usb\ MQX\_USB\_Device\_User\_Guide.pdf

# 7 74HC595 驱动

## 7.1 概述

K22FSH开发板上的 WIFI模块一共47pin，引出的GPIO数目有限，开发板上加了两块74HC595串转并芯片，这样把WIFI模块上的3个GPIO引脚扩展出16个输出引脚。74HC595 驱动提供接口单独设置这16个引脚输出电平的高低。

## 7.2 配置宏定义

74HC595 配置宏定义在文件 mqx/source/bsp/k22fsh/k22fsh.h 中 具体含义见下表

|  |  |  |
| --- | --- | --- |
| **宏** | **值** | **描述** |
| BSP\_74HC595\_0 | 0 | 74HC595 组的序号 |
| BSP\_74HC595\_0\_COUNT | 2 | 74HC595 组包含两块此芯片 |
| BSP\_74HC595\_SHCP  BSP\_74HC595\_SHCP\_MUX\_GPIO | (GPIO\_PORT\_C | GPIO\_PIN7)  (LWGPIO\_MUX\_C7\_GPIO) | 移位寄存器时钟输入为K22  的PORTC PIN7 |
| BSP\_74HC595\_STCP  BSP\_74HC595\_STCP\_MUX\_GPIO | (GPIO\_PORT\_B | GPIO\_PIN3)  (LWGPIO\_MUX\_B3\_GPIO) | 存储寄存器时钟输入为K22  的PORTB PIN3 |
| BSP\_74HC595\_DS  BSP\_74HC595\_DS\_MUX\_GPIO | (GPIO\_PORT\_B | GPIO\_PIN2)  (LWGPIO\_MUX\_B2\_GPIO) | 串行数据输入为K22  的PORTB PIN2 |
| BSP\_74HC595\_SPI\_S1 | 15 | 74HC4052D的 S1 输入 |
| BSP\_74HC595\_SPI\_S0 | 14 | 74HC4052D的 S0 输入 |
| BSP\_74HC595\_VBLE\_3V3 | 13 | BLE芯片EM9301 的电源 |
| BSP\_74HC595\_VIR\_RCV\_3V3 | 12 | 红外电源 |
| BSP\_74HC595\_BLT\_PWREN | 11 | LCD背光电源 |
| BSP\_74HC595\_RELAY\_CTRL | 10 | 继电器控制 |
| BSP\_74HC595\_GPIO | 9 | GPIO 输出 |
| BSP\_74HC595\_AUD\_LDO1\_EN | 8 | AUDIO DAC 芯片 LDO控制 |
| BSP\_74HC595\_AMB\_LED | 7 | D106 LED 开关 |
| BSP\_74HC595\_GPIO\_HOST\_EN | 6 | USB Host/Device 模式切换  “1” Device 模式  “0” Host 模式 |
| BSP\_74HC595\_5V\_PWREN | 5 | 5V\_PWREN |
| BSP\_74HC595\_LCD\_nRST | 4 | LCD Reset |
| BSP\_74HC595\_LCD\_PWREN | 3 | LCD 电源开关 |
| BSP\_74HC595\_AUD\_PWREN | 2 | AUDIO DAC 芯片 电源控制 |
| BSP\_74HC595\_BT\_PWREN | 1 | BT RDA5876 芯片 电源控制 |
| BSP\_74HC595\_BLE\_RST | 0 | BLE芯片EM9301 Reset |

## 7.3 代码路径

|  |  |
| --- | --- |
| 驱动 | 路径 |
| 74HC595驱动 | mux\source |

## 7.4 头文件

使用74HC595驱动，请包含下面的头文件，头文件位置在 mux\source

|  |  |
| --- | --- |
| 驱动 | 头文件 |
| 74HC595驱动 | 74hc595.h |

## 7.5 init/deinit

### 7.5.1 mux\_74hc595\_init

int **mux\_74hc595\_init**(uint32\_t index,

uint32\_t count,

uint32\_t shcppinid, uint32\_t shcppinmux,

uint32\_t stcppinid, uint32\_t stcppinmux,

uint32\_t dspinid, uint32\_t dspinmux

);

驱动初始化函数

74HC595驱动使用之前需要调用init函数，K22FSH开发的BSP会在mqx/source/bsp/k22fsh/init\_bsp.c 文件\_bsp\_init函数中调用此函数

初始化完成后，这组74HC595的所有输出引脚输出都设置为低电平。

**参数：**

**index** 这组74HC595 初始化序号

**count**  这组 74HC595 包含此芯片的数目

**shcppinid** 移位寄存器时钟输入 PIN

**shcppinmux** 移位寄存器时钟输入PIN MUX 配置

**stcppinid** 存储寄存器时钟输入 PIN

**stcppinmux** 存储寄存器时钟输入 PIN MUX 配置

**dspinid** 串行数据输入 PIN

**dspinmux** 串行数据输入 PIN MUX 配置

**返回值：**

**0：**  init　成功

**-1** init　失败

**例子：**

mux\_74hc595\_init(BSP\_74HC595\_0, BSP\_74HC595\_0\_COUNT,

BSP\_74HC595\_SHCP, BSP\_74HC595\_SHCP\_MUX\_GPIO,

BSP\_74HC595\_STCP, BSP\_74HC595\_STCP\_MUX\_GPIO,

BSP\_74HC595\_DS, BSP\_74HC595\_DS\_MUX\_GPIO);

### 7.5.2 mux\_74hc595\_deinit

int mux\_74hc595\_deinit(uint32\_t index);

**参数：**

**index:** 需要deinit的 74HC595 序号

**返回值：**

**0：** deinit 成功

**-1:** deinit 失败

## 7.6 Set/Clear

### 7.6.1 mux\_74hc595\_set\_bit

int mux\_74hc595\_set\_bit(uint32\_t index, uint32\_t bitindex);

这个接口可以单独设置一组74HC595某个引脚输出电平为高，对此输出的设置不影响这组74HC595其他引脚的输出。

**参数：**

**index:** 需要操作的74HC595组序号

**bitindex:** 需要设置的 74HC595输出引脚 对于K22FSH开发板来说，引脚的定义请参考

[**7.2 配置宏定义**](#_7.2_配置宏定义)

**返回值：**

**0：** 设置成功

**-1:** 设置 失败

**例子：**

mux\_74hc595\_set\_bit(BSP\_74HC595\_0, BSP\_74HC595\_AUD\_PWREN);

mux\_74hc595\_set\_bit(BSP\_74HC595\_0, BSP\_74HC595\_AUD\_LDO1\_EN);

这个例子应用在K22FSH开发板的Audio DAC驱动中，它打开了 Audio DAC的电源。

### 7.6.2 mux\_74hc595\_clear\_bit

int mux\_74hc595\_clear\_bit(uint32\_t index, uint32\_t bitindex);

这个接口可以单独设置一组74HC595某个引脚输出电平为低，对此输出的设置不影响这组74HC595其他引脚的输出。

**参数：**

**index:** 需要操作的74HC595组序号

**bitindex:** 需要设置的 74HC595输出引脚 对于K22FSH开发板来说，引脚的定义请参考

[**7.2 配置宏定义**](#_7.2_配置宏定义)

**返回值：**

**0：** 设置成功

**-1:** 设置 失败

**例子：**

mux\_74hc595\_clear\_bit(BSP\_74HC595\_0, BSP\_74HC595\_AUD\_PWREN);

mux\_74hc595\_clear\_bit(BSP\_74HC595\_0, BSP\_74HC595\_AUD\_LDO1\_EN);

这个例子应用在K22FSH开发板的Audio DAC驱动中，它关闭了 Audio DAC的电源。

# 8 LCD 驱动

## 8.2 概述

K22FSH 开发板J402接口支持SPI总线驱动芯片为Sitronix ST7567 的LCD。

目前K22FSH能够直接驱动的LCD为 POWERTIP 的 PE12864-055，可以参考如下网页:

<http://www.powertip.com.tw/products_2.php?product_id=1412267238&area_idbk=1228330941>

PE12864-055 一共可以显示 128x64个点，本LCD驱动只支持显示ASCII 字符，每个字符的大小固定为8x16个点，这样这块LCD上一共可以显示4行，每行最多显示16个 ASCII字符。



## 8.2 代码路径

|  |  |
| --- | --- |
| 驱动 | 路径 |
| LCD 驱动 | lcd\source |

## 8.3 头文件

使用74HC595驱动，请包含下面的头文件，头文件位置在 lcd\source

|  |  |
| --- | --- |
| 驱动 | 头文件 |
| LCD 驱动 | lcd.h |

## 8.4 printf\_lcd

void printf\_lcd(const char \*fmt\_ptr, ...);

打印字符串到LCD，只支持显示ASCII字符，这个函数执行时将独占K22的SPI1总线。LCD每行最多显示16个字符，超过16个字符，将分行显示，LCD最多显示4行。

# 9 音频输出适配层

## 9.1 概述

这层代码作为应用程序和MQX SAI同步音频接口驱动的适配层，还包含了K22FSH开发板上 AUDIO DAC 芯片的驱动。

## 9.2 代码路径

|  |  |
| --- | --- |
| 驱动 | 路径 |
| 音频输出适配层 | msi\source |

## 9.3 头文件

使用音频输出适配层，请包含下面的头文件，头文件位置在 msi\source

|  |  |
| --- | --- |
| 驱动 | 头文件 |
| 音频输出适配层 | msi.h |

## 9.4 init/deinit

K22 MQX SAI同步音频接口驱动内部会分配periodbuffersize x periodbuffercount 字节大小的内存，其内部每次通过DMA送到SAI接口的内存大小也是periodbuffersize，在SAI同步音频接口驱动初始化的时候得设置 periodbuffersize和periodbuffercount，同理此音频输出适配层在初始化的时候也要设置这两个值。

### 9.4.1 msi\_snd\_init

int32\_t msi\_snd\_init(void);

音频输出适配层初始化函数，periodbuffersize 设置为 512 bytes periodbuffercount 设置为 4个，此函数还将初始化 K22FSH开发板子上的AUDIO DAC.

**返回值：**

**0：**  init　成功

**-1** init　失败

### 9.4.2 msi\_snd\_init\_with\_periodbuffer

int32\_t msi\_snd\_init\_with\_periodbuffer(int32\_t periodbuffersize, int32\_t periodbuffercount);

音频输出适配层初始化函数，留有参数用户可以设置**periodbuffersize，periodbuffercount。**此函数还将初始化K22FSH开发板上的 AUDIO DAC.

**参数：**

**periodbuffersize:** SAI同步音频接口驱动DMA每次送到SAI接口内存大小

**periodbuffercount:** SAI同步音频接口驱动内部**periodbuffer** 数目

**返回值：**

**0：**  init　成功

**-1** init 失败

### 9.4.3 msi\_snd\_deinit

int32\_t msi\_snd\_deinit(void);

关闭和清理MQX SAI同步音频接口，释放资源，同时关闭和清理K22FSH开发板上的 AUDIO DAC。

**返回值：**

**0：**  deinit　成功

**-1** deinit 失败

## 9.5 msi\_snd\_set\_format

int32\_t msi\_snd\_set\_format(uint32\_t samplerate, uint8\_t bitwidth, uint8\_t chnum);

设置音频格式

**参数：**

**samplerate:** 音频采样率

**bitwidth:** 音频采样位宽

**chnum:** 音频声道数

**返回值：**

**0：**  设置成功

**-1** 设置失败

## 9.6 msi\_snd\_write

int32\_t msi\_snd\_write(uint8\_t \* pcmStream, uint32\_t pcmCnt);

写音频采样数据，此API为阻塞式。

**参数：**

**pcmStream:** 音频采样数据内存起始地址

**pcmCnt:** 音频采样数据个数，这个不是字节数

**返回值：**

**0：**  成功

**-1** 失败

## 9.7 mute/unmute

### 9.7.1 msi\_snd\_mute

int32\_t msi\_snd\_mute(void);

音频输出静音。

**返回值：**

**0：**  成功

**-1** 失败

### 9.7.2 msi\_snd\_unute

int32\_t msi\_snd\_umute(void);

音频输出恢复。

**返回值：**

**0：**  成功

**-1** 失败

## 9.8 vol control

### 9.8.1 msi\_snd\_vol\_up

int32\_t msi\_snd\_vol\_up(void)

提高音频输出音量。

**返回值：**

**0：**  成功

**-1** 失败

### 9.8.2 msi\_snd\_vol\_down

int32\_t msi\_snd\_vol\_down(void)

降低音频输出音量。

**返回值：**

**0：**  成功

**-1** 失败

## 9.9 msi\_snd\_flush

int32\_t msi\_snd\_flush(void);

把音频输出适配层缓存的内存写到SAI。

**返回值：**

**0：**  成功

**-1** 失败

# 10 音频播放中间层

## 10.1 概述

一个简易的本地音频文件播放中间层，输入为U盘或者SD卡在MQX 的MFS文件系统中的盘符，这个中间层会搜索此盘符下面所有的音频文件，根据音频文件的后缀决定其音频编码类型，音频编码格式分析器分析其音频编码格式的数据后交给音频解码器解码，解码后的数据交给音频输出适配层输出到Audio DAC。

## 10.2 代码路径

|  |  |
| --- | --- |
| 驱动 | 路径 |
| 音频播放中间层 | player\source |

## 10.3 头文件

使用音频播放中间层，请包含下面的头文件，头文件位置在 player\source

|  |  |
| --- | --- |
| 驱动 | 头文件 |
| 音频播放中间层 | play.h |

## 10.4 playback\_init

int playback\_init(void);

音频播放中间层初始化。

**返回值：**

**0：**  成功

**-1** 失败

## 10.5 playback\_setup

void playback\_setup(lp\_param\_t \* lpp\_param);

设置音频播放中间层，这个接口调用结束，中间层就会搜索音频文件并播放。

**参数：**

**lpp\_param:**

**typedef struct {**

**music\_play\_source\_type\_t lp\_type;**

**LWSEM\_STRUCT \* mfs\_io\_sem;**

**char path[16];** U盘或者SD在MQX的MFS文件系统中的盘符

**} lp\_param\_t;**

## 10.6 playback\_cancel

void playback\_cancel(\_mqx\_uint event\_type);

取消音频播放，这个接口通常在U盘或者SD拔出后调用。

# 11 蓝牙协议

## 11.1 概述

K22FSH包含的蓝牙协议栈基于 Google ADK2012 里的蓝牙协议栈，参见

<http://developer.android.com/tools/adk/adk2.html>，协议栈以library形式包含在开发包中，目前有GCC和IAR两个编译版本，支持的Profile 有 A2DP，AVRCP, HFP, 支持的HCI 传输接口有 USB，UART。

## 11.2 库路径

|  |  |
| --- | --- |
| 驱动 | 路径 |
| Bluetooth 协议栈 | externallibs\bluetooth\gcc  externallibs\bluetooth\iar |

## 11.3 头文件

使用蓝牙协议栈，请包含下面的头文件，头文件位置在 externallibs\bluetooth\include

|  |  |
| --- | --- |
| 驱动 | 头文件 |
| Bluetooth 协议栈 | ADK.h |

## 11.4 协议栈初始化

void ADK\_adkInit(struct FWKFUNCS fwkFuncs);

初始化协议栈本身，注册A2DP，AVRCP，HFP Profile。此函数返回后，Bluetooth协议栈内部task并不运行。 协议栈初始化不包含 Bluetooth 控制器的初始化，控制器初始化参考 [11.5 Bluetooth控制器的初始化](#_11.5__Bluetooth控制器的初始化)

**参数：**

**fwkfuncs:**

**struct FWKFUNCS{**

协议栈内部Task的MQX优先级，这个优先级必须和调用 [11.5 Bluetooth控制器的初始化](#_11.5__Bluetooth控制器的初始化) 的Task优先级一样。 如果有USB Stack运行，此优先级必须要低于 USB Stack内部Task的优先级。USB Stack 内部Task优先级参考

usb\host\source\include\host\_cnfg.h 宏 USBCFG\_KHCI\_TASK\_PRIORITY

**uint8\_t taskpriority;**

音频函数结构，可以参考 音频输出适配层 [9 音频输出适配层](#_9_音频输出适配层) 。

A2DP, HFP Profile 都会使用这个音频函数结构。

**struct{**

**int32\_t (\*init\_with\_periodbuffer)(int32\_t, int32\_t);**

**int32\_t (\*init)(void);**

**int32\_t (\*deinit)(void);**

**int32\_t (\*setSampleRate)(uint32\_t, uint8\_t, uint8\_t);**

**int32\_t (\*tryAddBuffer) (uint8\_t \*, uint32\_t);**

**int32\_t (\*vol\_up) (void);**

**int32\_t (\*vol\_down)(void);**

**int32\_t (\*mute) (void);**

**int32\_t (\*unmute)(void);**

**int32\_t (\*flush)(void);**

**}dacFuncs;**

**};**

## 11.5 Bluetooth控制器的初始化

int ADK\_btInit(MQX\_FILE\_PTR handle, uint8\_t type);

初始化Bluetooth 控制器: 启动内部的Task， reset 蓝牙控制器，获取控制器的一些特性等等。

**参数:**

**handle:** Bluetooth 控制器 HCI 接口的MQX文件指针，对于UART接口的控制器， handle 就是 此

UART口的文件指针，对于USB接口的控制器，handle就是 Bluetooth USB控制器的文

件指针。

**type：** 1 UART 接口

2 USB 接口

**返回值：**

**0：**  成功

**-1** 失败

## 11.6 Bluetooth 控制器的移除

void ADK\_btDeinit(void);

USB接口的Bluetooth控制器移除后，调用此函数清除协议栈内部分配的资源，销毁内部的Task.

## 11.7 取得，设置Bluetooth 控制器MAC 地址

### 11.7.1 ADK\_adkbtLocalMac

char ADK\_adkbtLocalMac(uint8\_t\* buf);

取得Bluetooth 控制器 MAC地址

**参数：**

**buf:** 6字节大小内存，用来存放mac地址。

**返回值：**

**0:** 失败

**1：** 成功

### 11.7.2 ADKRDAChangeBDAddr

char ADKRDAChangeBDAddr(const uint8\_t\* mac)

RDA5876 芯片改变控制器MAC地址，该地址在重启后失效。

**参数：**

**buf:** 6字节大小内存，用来存放mac地址。

**返回值：**

**0:** 失败

**1：** 成功

## 11.8 ADK\_btEnable

void ADK\_btEnable(adkBtConnectionRequestF crF,

adkBtLinkKeyRequestF krF,

adkBtLinkKeyCreatedF kcF,

adkBtPinRequestF prF,

adkBtDiscoveryResultF drF);

设置Bluetooth connection 建立过程中的一些回调函数。

## 11.9 ADK\_btSetLocalName

char ADK\_btSetLocalName(const char\* name);

设置Bluetooth 控制器的本地名字。

**参数：**

**name:** Bluetooth 控制器本地名字字符串首地址

**返回值：**

**0:** 失败

**1：** 成功

## 11.10 ADK\_btDiscoverable

char ADK\_btDiscoverable(char on);

设置Bluetooth控制器的被发现属性。

**参数：**

**on: 0** 不能被发现

**1** 能被发现

**返回值：**

**0:** 失败

**1：** 成功

## 11.11 ADK\_btConnectable

char ADK\_btConnectable(char on);

设置Bluetooth控制器的连接属性。

**参数：**

**on: 0** 不能被连接

**1** 能被连接

**返回值：**

**0:** 失败

**1：** 成功

## 11.12 ADK\_btSetDeviceClass

char ADK\_btSetDeviceClass(uint32\_t cls);

设置 Bluetooth的 The Class of Device/Service field。

**参数：**

**cls:** The Class of Device/Service field

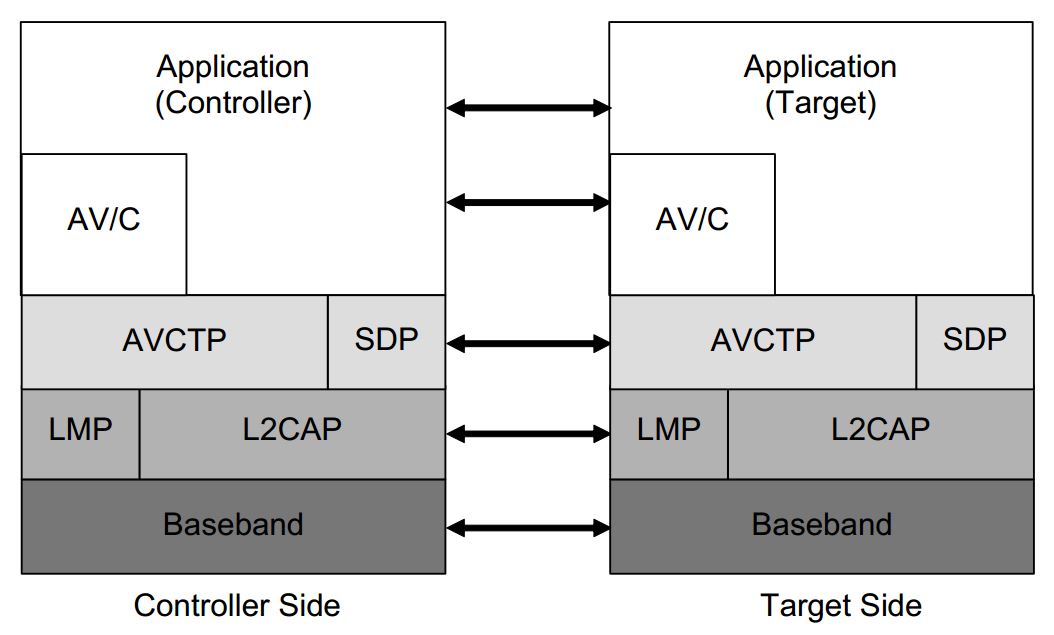
**返回值：**

**0:** 失败

**1：** 成功

## 11.13 AVRCP Profile

此蓝牙协议栈实现的是AVRCP Profile的 控制器一侧。



### 11.13.1 avrcpServicePassthroughHandle

int avrcpServicePassthroughHandle(BTAVRCPCTKEY key, BTAVRCPCTKEYACTION action);

发送控制命令,详细见下面的参数。

**参数：**

**key:**

**typedef enum btAvrcpCTKey {**

**AVRCPCTPLAY,** 播放

**AVRCPCTPAUSE,** 暂停

**AVRCPCTSTOP,** 停止

**AVRCPCTNEXT,** 下一首

**AVRCPCTPREVIOUS,** 上一首

**AVRCPCTVOLUP,** 提高音量

**AVRCPCTVOLDOWN,** 降低音量

**AVRCPCTUNKNOW,**

**} BTAVRCPCTKEY;**

**action:**

**typedef enum btAvrcpCTKeyAction {**

**AVRCPCTPRESS,** 按键按下

**AVRCPCTRELEASE,** 按键松开

**AVRCPCTUNKNOWACTION,**

**} BTAVRCPCTKEYACTION;**

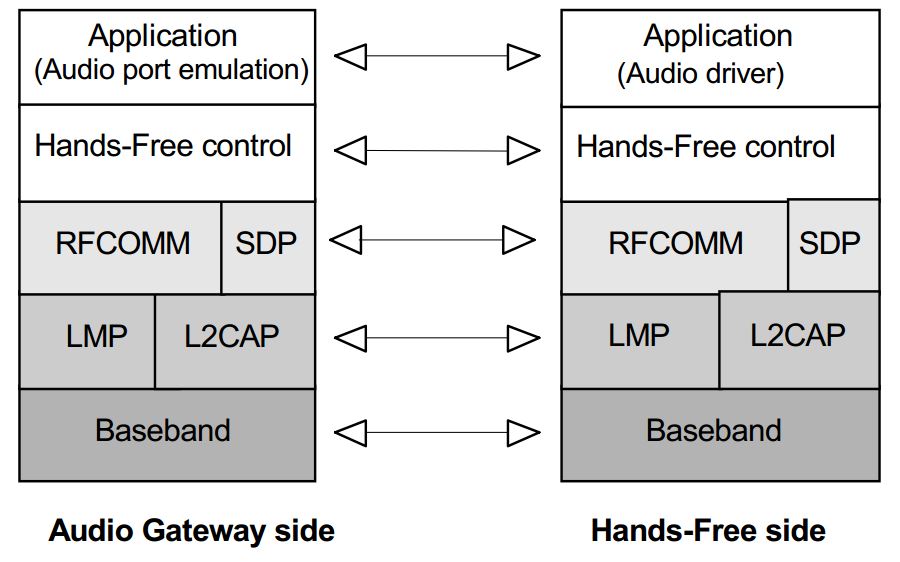
**返回值：**

**0：**  成功

**-1** 失败

## 11.14 HFP Profile

此蓝牙协议栈实现的是HFP Profile 的 Hands-Free一侧。



### 11.14.1 AT 命令

HFP AT 命令接口都是非阻塞式的，发出AT命令后不等待对方的响应马上返回，每个 AT命令接口都有一个回调函数的参数，这个回调函数会在AT命令得到对方响应后调用。

回调函数的原型

typedef void\*(\*ATCOMMANDCALLBACK)(void \*userparam, void \*param);

AT命令接口函数列表

|  |  |  |
| --- | --- | --- |
| 函数 | 参数 | 描述 |
| btHfpCall | uint8\_t \*number,  ATCOMMANDCALLBACK atcommandcallback,  void \*atcommandcbparam | Place a Call With the Phone Number Supplied by the HF  number:Phone Number String. |
| btHfpAnswer | ATCOMMANDCALLBACK atcommandcallback,  void \*atcommandcbparam | Answer an Incoming Call |
| btHfpReject | ATCOMMANDCALLBACK atcommandcallback,  void \*atcommandcbparam | Reject an Incoming Call |
| btHfpLastNumReDial | ATCOMMANDCALLBACK atcommandcallback,  void \*atcommandcbparam | Last Number Re-Dial from the HF |
| btHfpDTMFCode | ATCOMMANDCALLBACK atcommandcallback,  void \*atcommandcbparam | Transmit DTMF Codes |
| btHfpSpeakerVolUpSync | ATCOMMANDCALLBACK atcommandcallback,  void \*atcommandcbparam | HF Voice Vol Up and Info AG |
| btHfpSpeakerVolDownSync | ATCOMMANDCALLBACK atcommandcallback,  void \*atcommandcbparam | HF Voice Vol Down and Info AG |
| btHfpMicVolUpSync | ATCOMMANDCALLBACK atcommandcallback,  void \*atcommandcbparam | HF Microphone Vol Up and Info AG |
| btHfpMicVolDownSync | ATCOMMANDCALLBACK atcommandcallback,  void \*atcommandcbparam | HF Microphone Vol Down and Info AG |
| btHfpNRECDisable | ATCOMMANDCALLBACK atcommandcallback,  void \*atcommandcbparamc | The HF Requests Turning Off the AG’s EC and NR |
| btHfpVoiceRecognitionActive | ATCOMMANDCALLBACK atcommandcallback,  void \*atcommandcbparamcc | Voice Recognition Activation Disable |
| btHfpVoiceRecognitionDeActive | ATCOMMANDCALLBACK atcommandcallback,  void \*atcommandcbparamccc | Voice Recognition Activation Enable |
| btHfpSimpleAT | uint8\_t \*atcommand,  ATCOMMANDCALLBACK atcommandcallback,  void \*atcommandcbparamccc | Directly Send AT Command  atcommnd: AT Command String |

### 11.4.2 Indicator 状态改变回调函数

char btHfpIndicatorRegister(INDICATORCALLBACK indicatorcb, void \*indicatorcbparam);

注册Indicator 状态改变回调函数，这个函数会在AG送出Indicator 改变时调用。

Indicatorcb 的原型是：

typedef void\*(\*INDICATORCALLBACK)(void \*userparam, HFP\_INDICATOR\_TYPE type, void \*param);

目前支持的 Indicator type如下：

**typedef enum** { /\* value \*/

**HFP\_I\_RING**, /\* no value \*/

**HFP\_I\_CLIP**, /\* string: after cb this string memory may not exist \*/

**HFP\_I\_CALL\_ACTIVE,** /\* bool \*/

**HFP\_I\_VGS**, /\* int \*/

**HFP\_I\_VGM,** /\* int \*/

**HFP\_I\_VRA,** /\* bool \*/

**HFP\_I\_UNKNOW**, /\* ? \*/

**}HFP\_INDICATOR\_TYPE;**

### 11.4.3 语音SCO 输出接口

HFP连接建立后 语音从AG端到HF端走的是SCO通道，语音进入HF端后直接输入给了音频输出适配层。 而HF端到AG端的语音需要通过此接口走SCO通道传给AG端。

char btHfpSendData(uint8\_t \*data, int len);

data 是输出的数据开始指针,len 是数据的长度，单位是字节。此函数成功返回0，否则返回-1.

data的长度需要和 btHFP.h 里的宏 SCOVOICEPAYLOADSIZE 的值保持对齐。