Spark karaoke SDK
方案移植指南

文档履历

版本号	日期	制/修订人	制/修订记录
V1.0	2022-05-16		初始版本

目录

1. 概述

1.1. 名词解释

Spark SDK，SmartLinkTech 公司的智能软件开发平台；

1.2. 编写目的

为了快速移植及配置 SmartLink 公司的 karaoke 产品方案，可以参照文档中的一些配置项进行简单的定制化修改，实现方案的快速移植。

1.3. 配置文件修改

客户在二次开发过程中，主要是修改用户层的配置文件，以 dongle_board 为例：
配置文件位于：spark_sdk\apps\karaoke\app_config：dongle_board_user_config.h

a. 修改 2.4G 默认地址：

```
#define G24_ADDR_DEFAULT                {0xF1,0xF1,0xF1,0xF1,0xF1,0xF1} //2.4G设备默认地址 [低地址-高地址]
#define G24_ADDR_USE_RANDOM             1       //2.4G设备地址使用随机值功能
```

b. 根据产品定义，配置使用场景：

```
#define MODE_BT_EN                      0       //蓝牙模式
#define MODE_G24_EN                     1       //2.4G模式
#define MODE_MUSIC_EN                   0       //本地音乐模式
#define MODE_RECORD_EN                  0       //录音模式
#define MODE_LINEIN_EN                  0       //音频输入模式
#define MODE_USBDEV_EN                  0       //USB从机模式
#define MODE_SPDIF_EN                   0       //光纤输入模式
#define MODE_POWEROFF_EN                1       //软关机模式
#define MODE_CHARGE_EN                  1       //充电模式
#define MODE_DISKUPDATE_EN              0       //设备升级固件模式
#define MODE_BTUPDATE_EN                0       //蓝牙升级固件模式
```

c. 根据原理图，选择麦克风是单端还是差分 mic：

```
/*
 * analog mic connection mode: 0:differential mode; 1:single-end mode
 */
#define AUDIO_AMIC0_SINGLE_END          1
#define AUDIO_AMIC1_SINGLE_END          1
#define AUDIO_AMIC2_SINGLE_END          1
```

d. 根据原理图，选择麦克风的音源输入：

```
/*
 * Default input source config
 * 0:AMIC0(analog microphone 0); 1:AMIC1; 2:AMIC2;
 * 3:DMIC0F(digital microphone 0 falling edge);
 * 4:DMIC0R(digital microphone 0 rising edge);
 * 5:AMIC0/1; 6:AMIC0/2; 7:AMIC1/2;
 * 8:DMIC0F_R(digital microphone 0 falling and rising edge)
 * 9:linein0, 10:linein1, 11:linein0/1
 */
#define AUDIO_REC_INPUT_SEL                 2       // record and call input select
```

e. 打印调试功能：

```
*                   uart config
*******************************************************************************/
#define UART0_EN                            1       //UART0模块 用于打印
#define UART0_TXBAUDRATE                    3000000 //波特率
#define UART0_RXBAUDRATE                    3000000
#define UART0_SCLK_FREQ                     48000000
#define UART0_TX_PIN_SEL                    11       //0:PA0(FPGA); 1:PB2; 2:
#define UART0_RX_PIN_SEL                    99      //0:PA1(FPGA); 1:PB1; 2:
```

选择打印口及波特率，默认选用 uart0 做打印。

在 spark_sdk\inc\sys\sys_config.h 打开打印总开关，就可以看到调试 log.

```
#define DEBUG_LOG_EN                1                   //UART打印总开关
```

2. 模块配置与修改

2.1. 提示音

客户在二次开发过程中，替换提示音文件即可，以 dongle 为例：

提示音文件位于：spark_sdk\apps\karaoke\app_config\dongle_board\tone，

a. 替换系统自带提示音文件

客户把需要替换的提示音文件，覆盖系统自带的文件，并命名为一样的名字就可以。

系统自带的提示音文件如下：

```
#define SYS_TONE_WELCOME_NAME                "welcome.mp3"
#define SYS_TONE_BTMODE_NAME                 "bluetooth_mode.mp3"
#define SYS_TONE_CONNECTED_NAME              "connected.mp3"
#define SYS_TONE_DISCONNECTED_NAME           "disconnected.mp3"
#define SYS_TONE_PEER_CONNECTED_NAME         "peer_connected.mp3"
#define SYS_TONE_PEER_DISCONNECTED_NAME      "peer_disconnected.mp3"
#define SYS_TONE_POWER_OFF_NAME              "power_off.mp3"
#define SYS_TONE_CALL_NAME                   "call.mp3"
#define SYS_TONE_LOW_POWER_NAME              "low_power.mp3"
#define SYS_TONE_GAME_MODE_OPEN_NAME         "game_mode_open.mp3"
#define SYS_TONE_GAME_MODE_CLOSE_NAME        "game_mode_close.mp3"
#define SYS_TONE_KEY_NAME                    "key.mp3"
#define SYS_TONE_KEY_UP_NAME                 "key_up.mp3"
#define SYS_TONE_NOISE_NAME                  "noise.mp3"
#define SYS_TONE_CLOSE_NAME                  "close.mp3"
#define SYS_TONE_PASS_THOSE_NAME             "pass_those.mp3"
#define SYS_TONE_SPACE_MODE_OPEN_NAME        "space_open.mp3"
#define SYS_TONE_SPACE_MODE_CLOSE_NAME       "space_close.mp3"
```

b. 根据方案的实际需要，配置各种场景下，是否需要播放提示音；

```
#define TONE_EN                                      1 //提示音总开关
#define TONE_VOLUME                                 50 //默认的提示音音量，如果某个提示音没有指定
#define TONE_M_CONNECTED_NOTIFY_S_PLAY_CONNECTED_TONE 1 //单耳连接手机然后对耳连接上，则副耳是否需

#define TONE_WELCOME_EN                  1       //开机提示音
#define TONE_BT_MODE_EN                  1       //蓝牙模式提示音
#define TONE_MUSIC_MODE_EN               0       //音乐模式提示音
#define TONE_CLOCK_MODE_EN               0       //clock模式提示音
#define TONE_FM_MODE_EN                  0       //FM模式提示音
#define TONE_LINEIN_MODE_EN              0       //linein模式提示音
#define TONE_PC_MODE_EN                  0       //PC模式提示音
#define TONE_POWER_OFF_EN                1       //关机提示音
#define TONE_RECORDING_MODE_EN           0       //录音模式提示音
#define TONE_CONNECTED_EN                1       //蓝牙连接提示音
#define TONE_DISCONNECTED_EN             1       //蓝牙断开连接提示音
#define TONE_PEER_CONNECTED_EN           0       //对耳连接提示音：0:disable; 1:使用对耳连接提
#define TONE_PEER_DISCONNECTED_EN        0       //对耳断开连接提示音：0:disable; 1:使用对耳断
#define TONE_INCOMING_NUM_EN             0       //来电报号提示音
#define TONE_CALL_EN                     2       //来电铃声：0:disable; 1:使用本地来电提示音;
#define TONE_ALARM_EN                    0       //闹钟提示音
#define TONE_MAX_VOLUME_EN               0       //最大音量提示音
#define TONE_MIN_VOLUME_EN               0       //最小音量提示音
#define TONE_LOW_POWER_EN                1       //低电量提示音(必须为wav(linear pcm))
#define TONE_BT_NEXT_EN                  0       //蓝牙模式下一首提示音
#define TONE_BT_PREV_EN                  0       //蓝牙模式上一首提示音
```

c. 配置各场景下提示音的音量：

```
//提示音音量
#define TONE_BLUETOOTH_MODE_VOL          60
#define TONE_CALL_VOL                    60
#define TONE_CONNECTED_VOL               60
#define TONE_DISCONNECTED_VOL            60
#define TONE_LOW_POWER_VOL               40
#define TONE_PEER_CONNECTED_VOL          60
#define TONE_PEER_DISCONNECTED_VOL       60
#define TONE_POWER_OFF_VOL               60
#define TONE_WELCOME_VOL                 60
#define TONE_KEY_VOL                     100
#define TONE_KEY_UP_VOL                  60
#define TONE_CLOSE_VOL                   60
#define TONE_NOISE_VOL                   60
#define TONE_PASS_THOSE_VOL              60
#define TONE_GAME_MODE_OPEN_VOL          60
#define TONE_GAME_MODE_CLOSE_VOL         60
```

d. 特别提醒：音频文件的相关参数建议如下：

格式： mp3
码率模式：CBR
码率： 24kb/s
采样率： 16kHz
声道: 1 声道

2.2. Key 的配置与修改

2.2.1. IO 按键的配置

a. 根据方案规格和原理图，选择合适的按键方式，例如：使用的是 IO 按键；

```
/*****************************************************************************
 *                key config
 *****************************************************************************/
#define KEY_AD_EN                               0       //AD按键
#define KEY_IO_EN                               1       //IO按键
#define KEY_IR_EN                               0       //红外遥控按键
#define KEY_TOUCH_EN                            0       //触摸按键
#define KEY_CODING_EN                           0       //编码开关旋钮
#define KEY_UART_EN                             0       //从UART接收按键
#define KEY_ONOFF_EN                            0       //ONOFF按键
#define KEY_DIG_EN                              0       //数字按键
```

b. 选择 IO key 使用的 GPIO 组，配置 pin 脚；

```
/*****************************************************************************
 *              keyio config
 *****************************************************************************/
#define KIO_GPIOB                       0       //使用PB组GPIO
#define KIO_GPIOC                       0       //使用PC组GPIO
#define KIO_GPIOD                       1       //使用PD组GPIO

#define KIO_M_EN                        1       //耳机上1个按键实现多功能（包括单击、多击、长按）

#define PIN_KIO1                        (PIN_D0)
```

c. 根据方案规格，配置按键功能；

```
/*
* 按键功能
*/
#define KEY_FUNC_S_EN                           1       //short
#define KEY_FUNC_U_EN                           1       //up
#define KEY_FUNC_SU_EN                          1       //short up
#define KEY_FUNC_L_EN                           0       //long
#define KEY_FUNC_LU_EN                          0       //long up
#define KEY_FUNC_H_EN                           0       //hold
#define KEY_FUNC_LL_EN                          1       //long long (1s/2s/3s/...)
#define KEY_FUNC_M_EN                           1       //multiple click
```

d. 单个 IO Key 用作多功能键，使能 KIO_M_EN；

```
/*****************************************************************************
 *              keyio config
 *****************************************************************************/
#define KIO_GPIOB                       0       //使用PB组GPIO
#define KIO_GPIOC                       0       //使用PC组GPIO
#define KIO_GPIOD                       1       //使用PD组GPIO

#define KIO_M_EN                        1       //耳机上1个按键实现多功能（包括单击、多击、长按）

#define PIN_KIO1                        (PIN_D0)

#if KIO_M_EN
#undef KEY_FUNC_L_EN
#undef KEY_FUNC_LU_EN
#undef KEY_FUNC_H_EN
#undef KEY_FUNC_LL_EN
#undef KEY_FUNC_M_EN
#define KEY_FUNC_L_EN                   0
#define KEY_FUNC_LU_EN                  0
#define KEY_FUNC_H_EN                   0
#define KEY_FUNC_LL_EN                  1
#define KEY_FUNC_M_EN                   1
#endif
```

2.2.2.ADC 按键的配置

a. 根据方案规格和原理图，选择合适的按键方式，使能 ADC key；

```
/*****************************************************************************
 *               key config
 *****************************************************************************/
#define KEY_AD_EN                               1       //AD按键
#define KEY_IO_EN                               0       //IO按键
#define KEY_IR_EN                               0       //红外遥控按键
#define KEY_TOUCH_EN                            0       //触摸按键
#define KEY_CODING_EN                           0       //编码开关旋钮
#define KEY_UART_EN                             0       //从UART接收按键
#define KEY_ONOFF_EN                            0       //ONOFF按键
#define KEY_DIG_EN                              0       //数字按键
```

b. 根据原理图，配置相应 ADC 按键；

```
/*****************************************************************************
 *               keyadc config
 *****************************************************************************/
#define KADC0_EN                                1       //PD1
#define KADC1_EN                                0       //PD0
#define KADC2_EN                                0       //PB11
#define KADC3_EN                                0       //PB12
#define KADC4_EN                                0       //PB13
#define KADC5_EN                                0       //PB8
```

c. 选择 ADC key 的工作方式，查询或者中断

```
/*
* 0: 查询方式（硬件上需确保adc输入电压小于VCCIO，不可以直接上拉到VCCIO）；
* 1: 中断方式；
*/
#define KADC0_IRQ                               0
#define KADC1_IRQ                               0
#define KADC2_IRQ                               0
#define KADC3_IRQ                               0
#define KADC4_IRQ                               0
#define KADC5_IRQ                               0
```

2.2.3.ON/OFF 键的配置

a. 根据方案规格和原理图，选择合适的按键方式，例如：

```
/*****************************************************************************
 *               key config
 *****************************************************************************/
#define KEY_AD_EN                               0       //AD按键
#define KEY_IO_EN                               0       //IO按键
#define KEY_IR_EN                               0       //红外遥控按键
#define KEY_TOUCH_EN                            0       //触摸按键
#define KEY_CODING_EN                           0       //编码开关旋钮
#define KEY_UART_EN                             0       //从UART接收按键
#define KEY_ONOFF_EN                            1       //ONOFF按键
#define KEY_DIG_EN                              0       //数字按键
```

b. 根据方案规格，配置按键功能；

```
/*
* 按键功能
*/
#define KEY_FUNC_S_EN                       1       //short
#define KEY_FUNC_U_EN                       1       //up
#define KEY_FUNC_SU_EN                      1       //short up
#define KEY_FUNC_L_EN                       0       //long
#define KEY_FUNC_LU_EN                      0       //long up
#define KEY_FUNC_H_EN                       0       //hold
#define KEY_FUNC_LL_EN                      1       //long long (1s/2s/3s/...)
#define KEY_FUNC_M_EN                       1       //multiple click
```

2.3. 显示的配置与修改

a. 根据方案规格，选择合适的显示方式；

```
/*****************************************************************************
 *                disp config
 *****************************************************************************/
#define DISP_EN                             0       //显示
#define LED_DIODE_EN                        1       //单个二极管
```

b. 根据原理图，配置二极管的 pin 脚和输出方式；

```
//LED_DIODE:
#define LED_RED_EN                      1
#define LED_RED_PIN_NUM                 (PIN_B22)
#define LED_RED_PIN_ACTIVE_STATE        3       //0-输出低电平；1-输出高电平；2-下拉2K；3-上拉2K

#define LED_BLUE_EN                     1
#define LED_BLUE_PIN_NUM                (PIN_B23)
#define LED_BLUE_PIN_ACTIVE_STATE       3       //0-输出低电平；1-输出高电平；2-下拉2K；3-上拉2K
```

c. 控制灯光各种行为的间隔：

```
#define DISP_INTERVAL_TIME      100                         //interval time, unit:ms
#define DISP_FLASH_TIME         (200/DISP_INTERVAL_TIME)    //闪烁间隔：200ms
#define DISP_FLASH_FAST_TIME    (200/DISP_INTERVAL_TIME)    //快闪间隔：200ms
#define DISP_FLASH_SLOW_TIME    (5000/DISP_INTERVAL_TIME)   //慢闪间隔：5000ms
#define DISP_DELAY_OFF_TIME     (3000/DISP_INTERVAL_TIME)   //延时关闭：3000ms
```

2.4. Accelerometer 的配置与修改

2.5. TWI 模块配置

a. 根据原理图，打开或关闭 TWI 模块，并配置相应的 pin 脚：

```
/*****************************************************************************
 *              twi config
 *****************************************************************************/
#define TWI1_EN                             0       //TWI1模块
#define TWI1_SCL_PIN_SEL                    0       //0:PB0; 1:PD0;
#define TWI1_SDA_PIN_SEL                    2       //0:PB2; 1:PD2;2:PB15;
#define TWI1_SD_SHARE_EN                    0       //TWI1与SD共用PB0/PB2
#if !TWI1_EN
#undef TWI1_SD_SHARE_EN
#define TWI1_SD_SHARE_EN                    0
#endif

#define TWI2_EN                             0       //TWI2模块
#define TWI2_SCL_PIN_SEL                    8       //0:PB3; 1:PB12; 2:PD1; 3:--; 4:PB6; 8:PD4
#define TWI2_SDA_PIN_SEL                    9       //0:PB4; 1:PB13; 2:PD0; 3:--; 4:PB7; 9:PD5
#define TWI2_SD_SHARE_EN                    0       //TWI2与SD共用PB3/PB4
#define TWI2_DM_SHARE_EN                    0       //TWI2_SCL与USB_DM共用PB3
#if !TWI2_EN
#undef TWI2_SD_SHARE_EN
#undef TWI2_DM_SHARE_EN
#define TWI2_SD_SHARE_EN                    0
#define TWI2_DM_SHARE_EN                    0
#endif

#define TWI_IO_EN                           1       //软件利用gpio模拟twi
#define TWI_IO_SCL_PIN                      PIN_D5
#define TWI_IO_SDA_PIN                      PIN_D6
```

2.6. UART 模块配置

a. 根据原理图，打开或关闭相应的 uart 模块，并配置相应的 pin 脚：

```
/*****************************************************************************
 *              uart config
 *****************************************************************************/
#define UART0_EN                            0       //UART0模块 用于打印
#define UART0_TXBAUDRATE                    3000000 //波特率
#define UART0_RXBAUDRATE                    3000000
#define UART0_SCLK_FREQ                     48000000
#define UART0_TX_PIN_SEL                    1       //0:PA0(FPGA); 1:PB2; 2:PB3; 3:PB4;
#define UART0_RX_PIN_SEL                    99      //0:PA1(FPGA); 1:PB1; 2:PB3; 3:PB5;
```

2.7. PMU 及充电相关配置

a. 根据电池规格，设置满电电压 PMU_CHARGE_VOLT 和充电电流 PMU_CHARGE_CURRENT；

```
#define PMU_POWEROFF_WHEN_BAT_FIRST_IN      1       //使能第一次插入电池时关机
#define PMU_ULTRA_LONG_PRESS_TIME           8000    //设置超长按时长，死机时超长按可以复位，单位:ms；只能
#define PMU_LONG_PRESS_TIME                 1000    //设置长按时长，长按开机或者长按关机，单位:ms；只能选
#define PMU_HYPERLONG_PRESS_TIME            3000    //设置二级长按时长，二级长按时长是基于长按，假如长按3
#define PMU_CHARGE_MIN_EN                   1       //充满后使用充电电流最小档充电
#define PMU_BAT_FULL_VOLT_TUNING            2       //满电电压微调，满电电压一般跟恒压值一致，可以在恒压值
#define PMU_CHARGE_VOLT                     0       //设置充电恒压值，0:4.2V，1:4.3V，2:4.35V，3:4.4V
#define PMU_CHARGE_VOLT_TUNING              2       //充电恒压值微调，在上面恒压值的基础上叠加，0:0mV，1:
#define PMU_CHARGE_CURRENT                  17500   //设置充电电流，是充电时芯片耗电和充到电池的电流总和，
#define PMU_SUBCHARGE_CURRENT               10      //设置涓流充电电流，单位:mA;
#define PMU_POWEROFF_CURRENT                2       //设置关机电流，没有电池的方案关机电流要设置为2，有电
#define PMU_VBUSIN_RESTART                  0       //使能VBUS插入后重启，常用于耳机方案
#define PMU_VCC_RTC_ALWAYS_ON               0       //设置RTC是否常在电，如果需要RTC时间在关机时继续走，
#define PMU_VCC_IO_VOLT                     3000    //设置VCC-IO电压，单位:mV，2700~3400mV，步进:100mV，
#define PMU_AVCC_VOLT                       3000    //设置AVCC电压，单位:mV，  2700~3400mV，  步进:100mV，
```

b. 设置电池曲线；

```
#define BAT_LEVEL_0PERCENT_VOLT             3300 //[ 0%~9%]:(3000~3300]  //300
#define BAT_LEVEL_10PERCENT_VOLT            3541 //[10%~19%]:(3300~3541] //241
#define BAT_LEVEL_20PERCENT_VOLT            3608 //[20%~29%]:(3541~3608] //67
#define BAT_LEVEL_30PERCENT_VOLT            3643 //[30%~39%]:(3608~3643] //35
#define BAT_LEVEL_40PERCENT_VOLT            3675 //[40%~49%]:(3643~3675] //32
#define BAT_LEVEL_50PERCENT_VOLT            3710 //[50%~59%]:(3675~3710] //35
#define BAT_LEVEL_60PERCENT_VOLT            3754 //[60%~69%]:(3710~3754] //44
#define BAT_LEVEL_70PERCENT_VOLT            3807 //[70%~79%]:(3754~3807] //53
#define BAT_LEVEL_80PERCENT_VOLT            3870 //[80%~89%]:(3807~3870] //63
#define BAT_LEVEL_90PERCENT_VOLT            3960 //[90%~99%]:(3870~3960] //90
#define BAT_LEVEL_100PERCENT_VOLT           4120 //   100%  :(3960~4200] //240
```

c. 设置低电电压和低电提示周期；

```
#define BAT_HW_LB_SHUT_EN                   1       //使能硬件低电关机
#define BAT_HW_LB_SHUT_VOLT                 2700    //设置硬件低电关机电压，只能选择2700/2800/2900/3100mV
#define BAT_HW_OK_VOLT                      2900    //设置硬件允许开机电压，只能选择2900/3000/3100/3300mV
#define BAT_SW_LB_WRN_VOLT                  3300    //设置软件低电提醒电压，单位:mV，2900~4400mV，步进:100mV
#define BAT_SW_LB_SHUT_VOLT                 3200    //设置软件低电关机电压，单位:mV，2900~4400mV，步进:100mV
#define BAT_SW_LB_WARNING_PERIOD            30000   //设置软件低电提醒周期，单位:ms，步进:1000ms
```

3. 用户 UI 操作定制

根据方案规格书和用户操作手册，自定义实现用户操作，以 dongle 为例：

a. UI 定制文件在 spark_sdk\apps\karaoke\app_config\dongle_board\config\dongle_board_ui.c

4. Declaration