

## AD14X 软件问题整理

链接：  
提取码：

以下是杰理客户问题反馈的二维码图片，可以通过微信扫二维码填写反馈！

也可以通过访问链接填写反馈：<https://www.wjx.cn/vj/O1EbvrN.aspx>

我们将第一时间安排对应工程协助解决！



杰理开源社区国内仓库 <https://gitee.com/Jieli-Tech>

<https://gitee.com/Jieli-Tech/fw-AD15N/issues> 这里有一些社区开发者的问题反馈，遇到问题可以先去这里看看。

1. 开发遇到任何问题先去这里找找.....	3
2. 串口设置好以后打印乱码，内部晶振没有校准，需要加下下面这段代码-210610hwx.....	3
3.DAC 输出方式设置，选择模拟输出和 PWM 输出-210610hwx.....	3
4.SD 卡 IO 设置-210610hwx.....	4
5.添加定时器中断-210610hwx.....	5
6.pwm 功能-210819hwx.....	6
7.无缝循环播放 210702hwx.....	7
8.AD 系列 MIDI 功能使用说明-210705hwx.....	8
9. MCU 工程一直喂狗会频繁开关中断导致定时器会经常出现几 uS 的误差-210713hwx.....	9
10.SDK V105 串口通信 串口接收 IO 需要设置数字功能-210715hwx.....	9
11.AD14 增加一个输入捕捉通道-210729hwx.....	9
12.SPI1 外挂 FLASH 录音修改-210729hwx.....	14
13.睡眠以后定时唤醒系统继续跑不复位-210730hwx.....	15
14.AD14 魔音变声功能参考-210809hwx.....	16
15.V106SDK 外挂 FLASH 音频下载和播放说明-210816hwx.....	18
16.V106 以前的 SDK 可能出现有些芯片音乐播放速度快-210902hwx.....	20
17.编译提示 FW 文件工具需要更新-210902hwx.....	20
18.解决以前的魔音功能参考代码开始的一小段声音没有了-210928hwx.....	20
19.无刷电机驱动程序参考-211015hwx.....	21
20.录音变调播放效果参数设置-211015hwx.....	21
21.烧录器升级以后旧版 SDK 没法烧录-211019hwx.....	21
22.电机驱动更新-211116hwx.....	22
23.实时变声效果说明-211118hwx.....	23

## 1.开发遇到任何问题先去这里找找

杰理开源社区国内仓库 <https://gitee.com/Jieli-Tech>

<https://gitee.com/Jieli-Tech/fw-AD15N/issues>

## 2.串口设置好以后打印乱码，内部晶振没有校准，需要加下下面这段代码-210610hwx

生产烧录的时候要屏蔽这段

```
int c_main(int cfg_addr)
{
#ifdef 1
    /*******debug code*****/
    JL_PLL->CON1 = 0XA403730; //me
    // JL_PLL->CON1 = 0XA40372d; //卓豪
    JL_PLL->CON1 = 0XA403724; //测试部
    JL_PLL->CON0 |= BIT(0); //EN
    volatile int i=0;
    for(i=0; i<0xfffff; i++);
    JL_PLL->CON0 |= BIT(1); //RST
    for(i=0; i<0xfffff; i++);
    /*******

```

## 3.DAC 输出方式设置，选择模拟输出和 PWM 输出-210610hwx

```
#define DAC_MODE_1_A ( DAC_PWM ) //PWM 数字输出 直接推 0.5W 喇叭
// 以下 DSM 模式暂时皆不可用
// #define DAC_MODE_1_B ( DAC_DSM | DAC_DSM11 )
// #define DAC_MODE_1_C ( DAC_DSM | DAC_DSM2 )
// #define DAC_MODE_1_D ( DAC_DSM | DAC_DSM3 )
// #define DAC_MODE_2_A ( DAC_DSM | DAC_DSM7 )
// #define DAC_MODE_3_A ( DAC_PWM )
// #define DAC_MODE_3_B ( DAC_DSM | DAC_DSM11 )

```

```
// #define DAC_MODE_3_C    ( DAC_DSM | DAC_DSM2 )
// #define DAC_MODE_3_D    ( DAC_DSM | DAC_DSM3 )
// #define DAC_MODE_4_A    ( DAC_DSM | DAC_DSM2 )
#define DAC_MODE_5_A    ( DAC_DSM | DAC_DSM7) //模拟信号输出 接功放
#define DAC_CURR_MODE    DAC_MODE_5_A
```

## 4.SD 卡 IO 设置-210610hwx

芯片内部只有一组卡口，不管选择那组卡口 SDMMC\_EN 不需要修改。  
在这里设置 sd 卡。

```
//SH54:
//SD A 组 IO:    CMD:PA02    CLK:PA01    DAT:PA03
//SD B 组 IO:    CMD:PA12    CLK:PA11    DAT:PA10
//SD C 组 IO:    CMD:PA05    CLK:PA04    DAT:USBDM
//SD D 组 IO:    CMD:PA05    CLK:PA04    DAT:PA06
//SH55:
//SD A 组 IO:    CMD:PA02    CLK:PA01    DAT:PA03
//SD B 组 IO:    CMD:PB01    CLK:PB00    DAT:PB02
//SD C 组 IO:    CMD:PB05    CLK:PB04    DAT:PB06
//SD D 组 IO:    CMD:PA07    CLK:PA06    DAT:PA08
#if TFG_SD_EN
SD0_PLATFORM_DATA_BEGIN(sd0_data)
    .port                = 'B',
    .data_width          = 1,
    .speed               = 8000000,
#if 0 //CMD 检测
    .detect_mode         = SD_CMD_DECT,
    .detect_func         = sdmmc_0_cmd_detect,
#endif
#if 1 //CLK 检测
    .detect_mode         = SD_CLK_DECT,
    .detect_func         = sdmmc_0_clk_detect,
    .detect_io_level     = 0, //0:低电平检测到卡 1:高电平检测到卡
#endif
#if 0 //IO 检测
    .detect_mode         = SD_IO_DECT,
    .detect_func         = sdmmc_0_io_detect,
    .detect_io           = IO_PORTx_xx, //用于检测的引脚
    .detect_io_level     = x, //0:低电平检测到卡 1:高电平检测到卡
#endif
    .power               = NULL,
    .priority             = 3,
```

```
SD0_PLATFORM_DATA_END()  
#endif
```

## 5.添加定时器中断-210610hwx

注意中断函数放在 RAM 里面，客户自己加的代码不要太多否则 RAM 不够。

```
__attribute__((weak))  
AT(.tick_timer_code)  
void user_timer2_ram_loop(void)  
{  
    // putchar('1');  
}
```

```
void user_timer2_loop(void){  
  
}
```

```
SET(interrupt(""))  
AT(.tick_timer_code)  
void user_timer2_isr()  
{  
    /*  
     *用户的 timer 函数不能加入到这里,加到 tick_timer_loop  
     * */  
    //bit_clr_swi(TIME0_INIT);  
    JL_TMR2->CON |= BIT(6);
```

```
    user_timer2_ram_loop();
```

```
    // user_timer2_loop();  
}
```

```
void user_timer2_init(void)  
{  
    tt_printf("----user_timer2_init \n");  
    HWI_Install IRQ_TIME2_IDX, (u32)user_timer2_isr, IRQ_IRTMR_IP); //timer0_isr  
    JL_TMR2->CNT = 0X00;  
    JL_TMR2->PRD = (sys_clock_get() / 1000) * 2;  
    JL_TMR2->CON |= BIT(0);
```

```
    tt_printf("user_timer2_init--end \n");
```

## 6.pwm 功能-210819hwxw

V106 版本 SDK 里面有写好 PWM 的初始化,一共有 6 路独立的 PWM。

1.timer 产生的 PWM

```
ms_api.c  mcpwm.c  timer_drv.c X  msg.c  app_record.c  encod
ad140-release_v1.0.6 > app > bsp > cpu > sh54 > C timer_drv.c > ...
62  /*
63  *timer_pwm
64  */
65  const u32 timer2_pwm_tab[] = {
66      IO_PORTA_03,
67      IO_PORTA_02,
68  };
69
70  #define _timer2_pwm_init(ch,fre,duty) \
71      TIMER_SFR(2)->PRD = clk_get("lsb")/fre; \
72      TIMER_SFR(2)->PWM#ch = TIMER_SFR(2)->PRD*duty/100; \
73      TIMER_SFR(2)->CON |= BIT(8 + 4*ch); \
74      gpio_set_direction(timer2_pwm_tab[ch], 0); \
75      gpio_set_die(timer2_pwm_tab[ch], 0); \
76      JL_IOMC->IOMC1 |= BIT(15 + ch); \
77      TIMER_SFR(2)->CON |= BIT(0)|BIT(6);
78
79  void timer2_pwm_init(u8 ch, u32 fre, u8 duty)
80  {
81      switch (ch) {
82      case 0:
83          _timer2_pwm_init(0, fre, duty);
84          break;
85      case 1:
86          _timer2_pwm_init(1, fre, duty);
87          break;
88      default:
89          break;
90      }
91  }
92
93
```

2.MCPWM 模块产生的 PWM



```
ms_api.c  mcpwm.c x timer_drv.c  msg.c  app_re
ad140-release_v1.0.6 > app > bsp > cpu > sh54 > C mcpwm.c > LOG_TAG
1  #include "sfr.h"
2  #include "cpu.h"
3  #include "config.h"
4  #include "gpio.h"
5  #include "clock.h"
6  #include "mcpwm.h"
7
8  #define LOG_TAG_CONST    NORM
9  #define LOG_TAG          "[normal]"
10 #include "debug.h"
11
12 #define pwm_frq_cnt(f)    (clk_get("lsb")/f)
13
14 const u32 mcpwm_tab[4] = {
15     IO_PORTA_04,
16     IO_PORTA_05,
17     IO_PORTA_08,
18     IO_PORTA_09,
19 };
20
```

## 7.无缝循环播放 210702hwx

最后一个参数设置 255 一直循环。

```
dec_obj *decoder_io(void *pfile, u32 dec_ctl, dp_buff *dbuff, u8 loop)
```

如果是 AD14 或者 AD15 还要在音频原文件要在转换的时候选上无缝循环的选项。



开了断点记忆和无限循环在上播断点的时候会从断点的位置开始到结束一直循环。需要按照下面修改。

```
#if 0
    clear_dp(dbuff);
    if (0 != loop) { // (dec_ctl & BIT_LOOP)
```

```
        p_dec->loop = loop;
        log_info("get loop dp\n");
        if (true == get_dp(p_dec, dbuff)) {
            log_info(" -loop save succ!\n");
            p_dec->p_dp_buf = check_dp(dbuff);
        } else {
            log_info(" -loop save fail!\n");
        }
    }
#else
    u8 MUSIC_PLAY_FROM_START = (check_dp(dbuff) == 0);
```

```
        clear_dp(dbuff);
        if (0 != loop) { // (dec_ctl & BIT_LOOP)
static dp_buff loop_dbuff;
            p_dec->loop = loop;
            if (MUSIC_PLAY_FROM_START)
            {
                log_info("*****get loop dp*****\n");
                if (true == get_dp(p_dec, &loop_dbuff)) {
                    vm_write(VM_LOOP_BP, (u8 *)&loop_dbuff, sizeof(dp_buff));
                    p_dec->p_dp_buf = check_dp(&loop_dbuff);
                } else {
                    log_info(" -loop save fail!\n");
                }
            }
        } else
        {
            log_info("*****get loop bp from vm *****\n");
            vm_read(VM_LOOP_BP, (u8 *)&loop_dbuff, sizeof(dp_buff));
            p_dec->p_dp_buf = check_dp(&loop_dbuff);
        }
    }
}
```

```
#endif
```

## 8.AD 系列 MIDI 功能使用说明-210705hwx

链接: <https://pan.baidu.com/s/1aRlafU0LSYlxhKWw8GZbNA>

提取码: 8888



## 9. MCU 工程一直喂狗会频繁开关中断导致定时器会经常出现几 uS 的误差-210713hwx

用定时器做时间精准的翻转 io 需要关掉看门狗，因为再喂狗的函数里面有关总中断的操作，实测会对定时器中断造成大概 5us 的误差。改成定时喂狗或者关掉看门狗。

```
wdt_close();
```

## 10.SDK V105 串口通信 串口接收 IO 需要设置数字功能-210715hwx

```
659     }
660     u8 gpio_set_uart1(u32 ut_ch) //ch=-1; 0; 1; 2; 3
661     {
662         if (ut_ch == -1) {
663             return 0;
664         }
665         JL_IOMC->IOMC0 |= BIT(7);
666         SFR(JL_IOMC->IOMC0, 8, 2, ut_ch); //USB
667         if (ut_ch == 0) {
668             gpio_set_direction(IO_PORTA_05, 1); //in /*JL_PORTA->DIR |= BIT(5);*/
669             gpio_set_direction(IO_PORTA_04, 0); //out /*JL_PORTA->DIR &= ~BIT(4);*/
670             gpio_set_pull_up(IO_PORTA_04, 1);
671             gpio_set_pull_up(IO_PORTA_05, 1);
672             gpio_set_die(IO_PORTA_05, 1);
673             return 1;
674         }
675     }
```

## 11.AD14 增加一个输入捕捉通道-210729hwx

AC104 103 版本 SDK 的红外用到了 TIMER2 和 IRFLT 模块。IRFLT 只是相当于一个信号过滤的模块，对于脉冲宽度的测量实际还是通过 TIMER 来实现的。

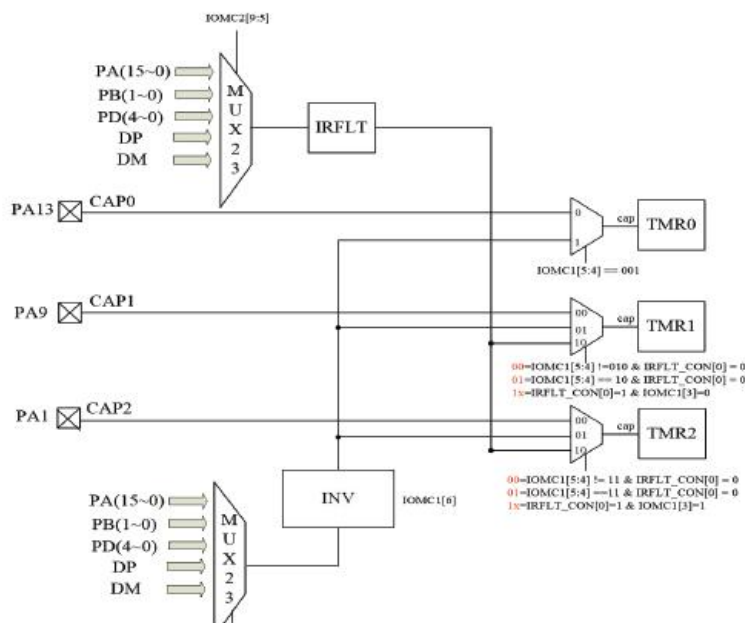
## 第 12 章 红外滤波模块 (IRFLT)

### 12.1 模块说明

IRFLT 是一个专用的硬件模块，用于去除掉红外接收头信号上的窄脉冲信号，提升红外接收解码的质量。

IRFLT 使用一个固定的时基对红外信号进行采样，必须连续 4 次采样均为 ‘1’ 时，输出信号才会变为 ‘1’，必须连续 4 次采样均为 ‘0’ 时，输出信号才会变为 ‘0’。换言之，脉宽小于 4 倍时基的窄脉冲将被滤除。改变该时基的产生可兼容不同的系统工作状态，也可在一定范围内调整对红外信号的过滤效果。

通过对 IOMC (IO re-mapping) 寄存器的配置，可以将 IRFLT 插入到系统 2 个 timer 中某一个的捕获引脚之前。例如通过 IOMC 寄存器选择了 IRFLT 对 timer1 有效，并且 IRFLT\_EN 被使能之后，则 IO 口的信号会先经过 IRFLT 进行滤波，然后再送至 timer1 中进行边沿捕获。



这里增加了 TIMER0 用默认的 PA13 来检测脉宽。

```
// 根据封装修改 IO 和 timer
// 注意红外默认用的是 TIMER2 的映射通道
// TIMER0 默认输入捕捉 IO PA13
// TIMER1 默认输入捕捉 IO PA09
// TIMER2 默认输入捕捉 IO PA01
```

```
#define USER_CAP_IO          IO_PORTA_13 //timer 默认的 io 需要映射可以参考红外按键
#define USER_CAP_TMR         JL_TMR0
#define IRQ_USER_CAP_TMR     IRQ_TIME0_IDX
```

```
#define INPUT_CHANNLE0_SRC_SEL(x)      SFR(JL_IOMC->IOMC2, 0, 5, x)
#define CAP_TIMER_SEL(x)               SFR(JL_IOMC->IOMC1, 4, 2, x)
```

```
u16 user_cap_prd;
```

```
static const u16 timer_div[] = {
    /*0000*/    1,
    /*0001*/    4,
    /*0010*/    16,
    /*0011*/    64,
    /*0100*/    2,
    /*0101*/    8,
    /*0110*/    32,
    /*0111*/    128,
    /*1000*/    256,
    /*1001*/    4 * 256,
    /*1010*/    16 * 256,
    /*1011*/    64 * 256,
    /*1100*/    2 * 256,
    /*1101*/    8 * 256,
    /*1110*/    32 * 256,
    /*1111*/    128 * 256,
};
```

```
/*-----*/
/**@brief    time1 红外中断服务函数
  @param    void
  @param    void
  @return   void
  @note     void timer1_ir_isr(void)
*/
/*-----*/
static u32 data_temp = 0;
static u8 cnt_temp = 0;
__interrupt
static void user_cap_isr(void)
{ //采样中不能在中断里面加打印
    u16 bCap1;
    u8 cap = 0;
```

```
static u8 cnt = 0;
```

```
USER_CAP_TMR->CON |= BIT(6);
```

```
bCap1 = USER_CAP_TMR->PRD;  
USER_CAP_TMR->CNT = 0;  
cap = bCap1 / user_cap_prd;
```

```
if (cap <= 1) {  
    data_temp >>= 1;  
    cnt_temp++;  
} else if (cap <= 3){  
    data_temp >>= 1;  
    cnt_temp++;  
    data_temp |= 0x8000;  
}else if ((cap == 13) && (cnt_temp < 8)){  
    data_temp = 0;  
    cnt_temp= 0;  
}  
if (cnt_temp == 32) {  
    log_info("data_temp:%x\n",data_temp );  
}  
}
```

```
void user_cap_outtime(void)  
{  
    static u8 wait_cnt = 0;  
    if(cnt_temp){  
        wait_cnt++;  
        if(wait_cnt > 20){  
            wait_cnt = 0;  
            data_temp = 0;  
            cnt_temp= 0;  
        }  
    }  
}
```

```
/*-----*/  
/**@brief    ir 按键初始化  
  @param    void  
  @param    void  
  @return   void  
  @note     void set_ir_clk(void)
```

```
((cnt - 1)* 分频数)/lsb_clk = 1ms
*/
/*-----*/
#define OSC_Hz          12000000L
```

```
// 根据实际要捕捉的信号频率调整 TIMER 的分频和 prd_cnt
/* #define MAX_TIME_CNT 0x07ff //分频准确范围, 更具实际情况调整 */
/* #define MIN_TIME_CNT 0x0030 */
void user_cap_timer_init(void)
{
    u32 clk;
    u32 prd_cnt;
    /* u8 index; */
    clk = OSC_Hz;//clock_get_lsb_freq();
```

```
    clk /= (1000 * 64);
    clk *= 1; //1ms for cnt
    prd_cnt = clk;
    user_cap_prd = prd_cnt;
```

```
    USER_CAP_TMR->CON = BIT(6);
    USER_CAP_TMR->PRD = 0;
    USER_CAP_TMR->CNT = 0;
    USER_CAP_TMR->CON = ((3 << 4) | (2 << 2) | (3 << 0)); //htc + falling edge
}
```

```
static void user_cap_io_init(void){
    gpio_set_direction(USER_CAP_IO, 1);
    gpio_set_die(USER_CAP_IO, 1);
    gpio_set_pull_up(USER_CAP_IO, 1);
}
```

```
/*-----*/
/**@brief    ir 按键初始化
 * @param    void
 * @param    void
 * @return    void
 * @note     void ir_key_init(void)
 */
/*-----*/
```

```
int user_cap_init(void )
```

```
{  
    //timer1  
    r_printf("ir key init >>>\n");  
  
    request_irq(IRQ_USER_CAP_TMR, IRQ_IRTMR_IP, user_cap_isr, 0);  
  
    user_cap_io_init();    //初始化输入捕捉的 IO  
    user_cap_timer_init();    //设置定时器  
  
    return 0;  
}
```

## 12.SPI1 外挂 FLASH 录音修改-210729hwx

注意这里说的外挂 FLASH 是指除了跑代码的那个 FLASH 另外增加的那个。比如 AD142A0 外挂跑代码的 FLASH 不属于这里说的。

1.修改录音初始化打开外挂的 FLASH。

```
67  
68     static int encode_start(void)  
69     {  
70         char c;  
71         u32 err;  
72         void *device = 0;  
73         e_status = 0;  
74  
75         //采样率支持8k,12k,16k,24k  
76         err = audio_adc_init_api(24000, ADC_MIC, 0);  
77         if (0 != err) {  
78             log_info(" audio adc init fail : 0x%x\n");  
79             return -1;  
80         }  
81  
82         #if(EXT_FLASH_REC == 0)  
83             device = dev_open(__SFC_NAME, 0);  
84         #else  
85             device = dev_open(__EXT_FLASH_NAME, 0);  
86         #endif
```



3.外挂 FLASH 需要自己定义录音区域大小（跑代码的 FLASH 会自己计算把所有可以用区域全部作为录音去不需要用户自己设定）

```
39 //如果是外挂FLASH 需要修改这里的录音起始地址和结束地址
40 void norfs_init_api(void)
41 {
42     log_info("norfs_init_api !!!\n");
43     u32 sector_start;
44     u32 sector_end;
45     u32 sector_size;
46     u32 sector_bit = 12;
47
48     sector_size = (1 << sector_bit);
49     sector_start = (boot_info.vm.vm_saddr + sector_size - 1) / sector_size;
50     sector_end = (boot_info.vm.vm_size + boot_info.vm.vm_saddr) / sector_size;
51     sector_end--;
52     norfs_init(sector_start, sector_end, sector_bit); //1<<12 = 4K
53 }
```

3.设置播放的设备。

```
145
146     switch (msg[0]) {
147     case MSG_REC_MODE_SWITCH:
148         if (e_status == ENC_ING) {
149             ///开始播放录音
150             encode_stop();
151             decoder_init();
152
153             #if(EXT_FLASH_REC == 0)
154                 log_info("INNER_FLASH record \n");
155                 void *device = dev_open(__SFC_NAME, 0);
156             #else
157                 log_info("EXT_FLASH_REC-record \n");
158                 void *device = dev_open(__EXT_FLASH_NAME, 0);
159             #endif
```

## 13.睡眠以后定时唤醒系统继续跑不复位-210730hwx

早期 SDK 做不了要这些版本才可以用

AC104 V104 和以上版本

AD14 V106 和以上版本

AD15V106 和以上版本

版权所有，侵权必究

15



## sys\_power\_down(nuS)

调用这个函数单位是微秒。注意如果开了看门狗最大只能设置看门狗复位时长的一半，如果传入的参数超过看门狗复位时间的一半不会生效。如果需要设置更长可以关闭看门狗。

注意 AD14 v106 A0 版本外挂 FLASH 默认 SDK 进入睡眠以后无法唤醒需要做以下修改

1.换库 <https://pan.baidu.com/s/1iCQwlyyKY2GZ623m9XxoDg> 提取码 8888

2.改这个宏

```
28
29 *****power_param*****
30
31 #define TCFG_LOWPOWER_POWER_SEL          PWR_LDO15
32 #define TCFG_LOWPOWER_BTOSC_DISABLE      0
33 #define TCFG_LOWPOWER_LOWPOWER_SEL       SLEEP_EN
34 #define TCFG_LOWPOWER_VDDIOM_LEVEL        VDDIOM_VOL_32V
35 #define TCFG_LOWPOWER_VDDIOW_LEVEL        VDDIOW_VOL_28V
36 #define TCFG_KEEP_FLASH_POWER_GATE       1
37 #define TCFG_LOWPOWER_DAC_OPEN            1
38 #define TCFG_NORFLASH_4BYTE_MODE         0
39
40
41
42 struct low_power_param power_param = {
43     .config          = TCFG_LOWPOWER_LOWPOWER_SEL, //0: sniff
44     .btosc_hz         = 16000000, //外接晶振
```

## 14.AD14 魔音变声功能参考-210809hwx

参考代码位置 <https://pan.baidu.com/s/1zPoLCQt9rD-KqJHvDPZkGA> 提取码 8888

- 1.增加了一个 ms 模式，开机默认进入 app\_ms.
- 2.检测能量自动开始录音，检测到没有说话声音以后自动播放刚才的录音。
- 3.上述代码只是一个参考，用来告诉客户 LADC 采样的数据怎样计算，怎样控制录音和播放录音。具体开始和结束的阈值需要客户自己调整，或者客户可以重写整个判断开始结束的函数。

```

89
90 AT(.adc_oput_code) ← 必须定义到RAM 而且这个函数里面不能随便调用
91 int mic_energy_check(u8 *buff, u32 len) ← 这个不是放到RAM的函数
92 #define START_THRESHOLD_VALUE 400000
93 #define STOP_THRESHOLD_VALUE 100000
94 #define AD_DATA_CHECK_TIMES 20 //计算多少次AD数据
95 #define RECORD_DATA_CHECK_CNT 30 //计算多少次AD数据
96 s16 *buff_temp = (s16 *)buf;
97 static long int buff_total=0;
98 static long int data_total=START_THRESHOLD_VALUE/2;
99 static u8 data_cnt=0;
100 static u16 data_rec_cnt=0;
101 u8 cnt = len/2;
102
103 // 过滤掉最开始开MIC可能会导致的PO声
104 if((jiffies - start_time) < 10){
105     return 0;
106 }
107 if(ms_start_flag){
108     while(cnt--){
109         if(buff_temp[cnt] < 0){
110             buff_total -= buff_temp[cnt];
111         }else{
112             buff_total += buff_temp[cnt];
113         }
114     }

```

这些参数自己调整

```

62 }
63
64 int record_play(void){
65     u32 decoder_type = BIT_A | BIT_UMP3 | BIT_SPEED; ← 这一位控制播放是否变调
66     // u32 decoder_type = BIT_A | BIT_UMP3 ;
67     ms_record_status = 2;
68     encode_stop();
69     decoder_init();
70     void *device = dev_open(__SFC_NANE, 0);
71     if (music_play(&music_obj, NULL, findex, decoder_type, device, MUSIC_MODE_
72 // log_info("music play succ \n");
73 }
74     return 0;
75 }
76
77 AT(.adc_oput_code)
78 u8 get_ms_record_status(void)

```

## 15.V106SDK 外挂 FLASH 音频下载和播放说明-210816hwx

1.在 CODEBLOCK 工程配置选项打开外挂 flash EXT\_FLASH\_EN=1

2.批处理里面增加外挂 FLASH 下载的命令

```
cd toy
isd_download.exe -tonorflash -dev sh54 -boot 0xb00 -div8 -wait 300 -uboot uboot.boot -uboot_compress -app app.bin
0x20000 -res dir_mio dir_eng -wflash dir_test 0 [PA05_1B_NULL]
::key AD14N.lkey
@rem isd_download.exe -tonorflash -dev sh54 -boot 0xb00 -div8 -wait 300 -uboot uboot.boot -uboot_compress -app
app.bin 0x20000 -res midi_cfg dir_midi dir_a dir_song dir_eng dir_poetry dir_story dir_notice -wflash dir_song 0
[PA05_1B_NULL]

@REM
@rem -format vm
@rem -format all
@rem -reboot 500

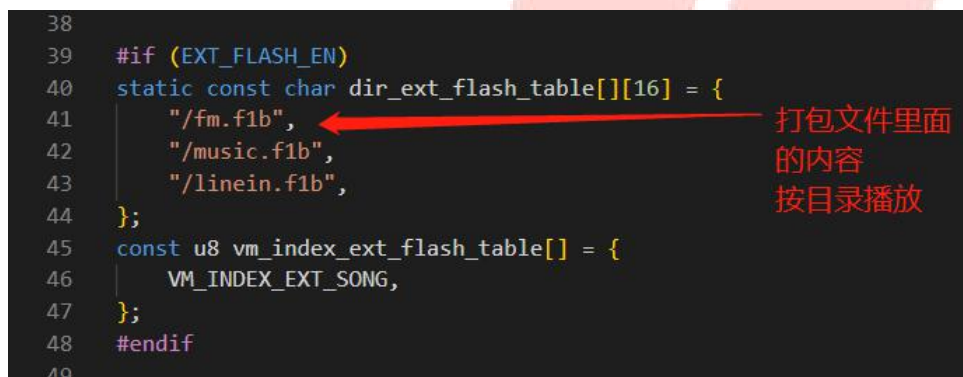
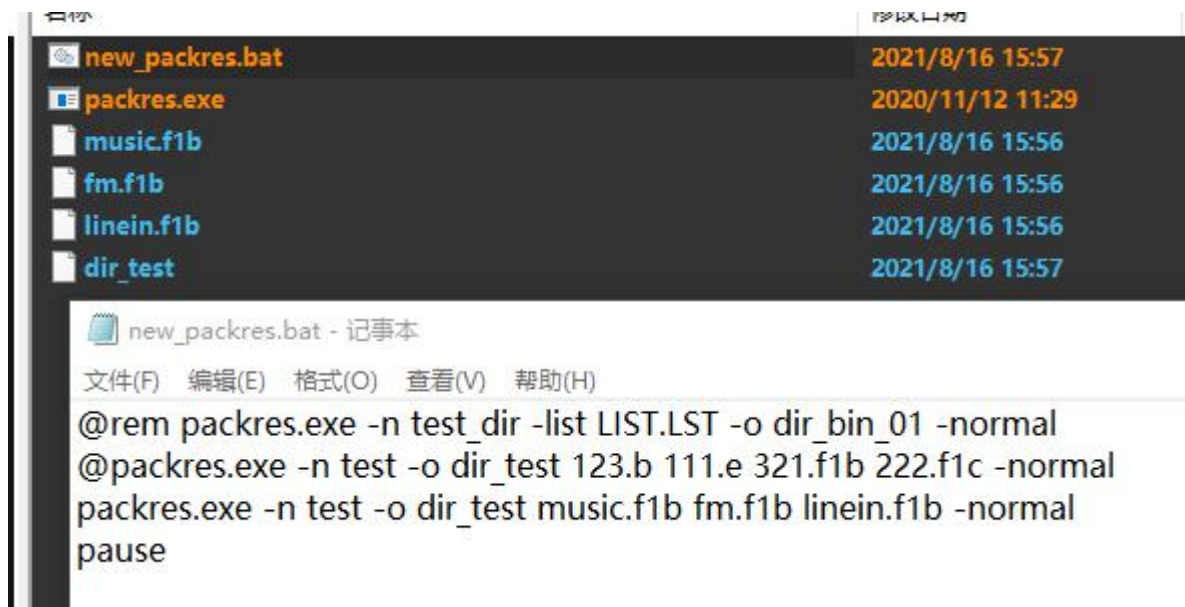
@REM //烧写外置flash 命令说明:
@rem -wflash dir_song 0 [PA05_1B_NULL]
@rem // dir_song : 要烧写的文件名 (文件需在download.bat文件夹下)
@rem // 0 : 文件烧录到外置flash的起始地址
@rem // [PA05_1B_NULL]: PA05: 外置flash片选引脚 (注意: 不能选USBDP/USBDM)
@rem // 1B : spi1 ,B端口
@rem // NULL: power_io & spi1_data_width, power_io连接到外置flash vcc引脚 可控制flash电源;spi1_data_width:0:
单线; 1: 双向
@rem // 例:NULL/PA00: power_io:无/pa0; spi1: 双向模式 (注: power_io不能选USBDP/USBDM)
@rem // NUL0/B010: power_io:无/pb1; spi1: 单线模式
@rem // NUL1/A081: power_io:无/pa8; spi1: 双向模式
@rem //注意: spi端口只能选B/C (B:PA11,PA12,PA10[CLK,DO,DI]; C:PA4,PA5,PA6[CLK,DO,DI])
```

3.注意代码里面的外挂 FLASH 设置要设置正确

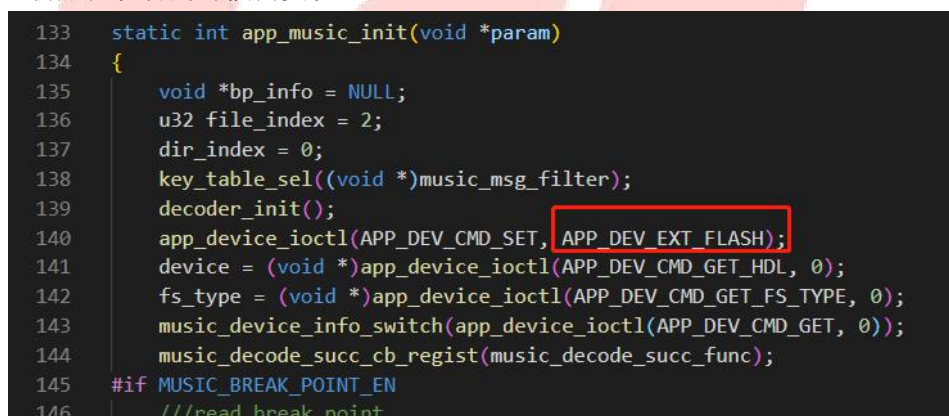
```
14
15 // *INDENT-OFF*
16 //sh54: SPI1: CLK , DO , DI
17 //SPI1: A组IO: IO_PORT_DP, IO_PORT_DM, IO_PORTA_03,
18 //SPI1: B组IO: IO_PORTA_11, IO_PORTA_12, IO_PORTA_10,
19 //SPI1: C组IO: IO_PORTA_04, IO_PORTA_05, IO_PORTA_06
20 //sh55: SPI1: CLK , DO , DI
21 //SPI1: A组IO: IO_PORTB_00, IO_PORTB_01, IO_PORTB_02,
22 //SPI1: B组IO: IO_PORTA_14, IO_PORTA_15, IO_PORTA_13,
23 //SPI1: C组IO: IO_PORTA_06, IO_PORTA_07, IO_PORTA_08,
24 //SPI1: D组IO: IO_PORTB_08, IO_PORTB_09, IO_PORTB_07
25 #if (EXT_FLASH_EN)
26 const struct spi_platform_data spi1_p_data = {
27     .port = {
28         SPI1_GROUPB_IO
29     },
30     .mode = SPI_MODE_BIDIR_1BIT,
31     .clk = 10000000,
32     .role = SPI_ROLE_MASTER,
33 };
34 //norflash
35 NORFLASH_DEV_PLATFORM_DATA_BEGIN(norflash_data)
36 .spi_hw_num = 1,
37 .spi_cs_port = IO_PORTA_05,
38 .spi_read_width = SPI_MODE_BIDIR_1BIT,
39 .spi_pdata = &spi1_p_data,
```



4.注意下载的打包文件是没有文件夹的，这里测试的 dir\_test 里面有这样三个文件，下载的时候工具会自动把打包文件里面的音乐文件一个一个存放到外挂 FLASH 的根目录下（内置 flash 是按打包的文件夹）。在程序里面可以用文件完整路径来播放，控制播放序号即可找到对应的文件。



5.初始化设备的时候需要设置 EXFLASH



## 16.V106 以前的 SDK 可能出现有些芯片音乐播放速度快-210902hwx

换掉 uboot 文件重新编译或者升级到 V106 版本 SDK。

压缩包里面有一个 EXE 工具如果只有 FW 文件找不到代码了可以用这个工具直接生成新的 FW 文件。

链接: <https://pan.baidu.com/s/1aoAkZLo0t0XqYgyEZiXp0w> 提取码: n2wy

## 17.编译提示 FW 文件工具需要更新-210902hwx

```
-----  
E:\UJM\任务\新任务\21083172 (AC104移植SDK104) \app\post_bu  
错误: fw文件格式比工具新, 请使用最新版本工具.  
E:\UJM\任务\新任务\21083172 (AC104移植SDK104) \app\post_bu  
找不到 E:\UJM\任务\新任务\21083172 (AC104移植SDK104) \app\p  
错误: 无效的FW文件[jl_isd.fw], 请重新选择
```

双击这个下载新的工具即可



## 18.解决以前的魔音功能参考代码开始的一小段声音没有了 -210928hwx

ReadMe.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

在原来版本上解决了开始的前面一点点声音没有了的问题。

- 1.一直开启编码功能。
- 2.用能量来控制要不要写入FLASH。
- 3.判断需要写入FLASH的时候会先把前面保存的一段声音写入。
- 4.播放完后马上打开编码。
- 5.SAVE\_DATA\_LEN 这个参数修改前面保存数据的长度。

具体代码修改请跟前面版本对比。

具体修改对比第 14 点下载的代码。

链接: <https://pan.baidu.com/s/12ys9myAv0i-HBXQJN6PLxg>

提取码: twsh

## 19.无刷电机驱动程序参考-211015hwx

链接: [https://pan.baidu.com/s/1kDnnIc\\_CWhBMNGCmUCk9jg](https://pan.baidu.com/s/1kDnnIc_CWhBMNGCmUCk9jg)

提取码: vw21

## 20.录音变调播放效果参数设置-211015hwx

可以用来做汤姆猫功能, 具体效果参考注释调整参数

```
void *speed_api(void *obuf, u32 insample, void **ppsound)
{
    log_info("speed_api\n");

    sp_parm.insample = insample;           //输入音频采样率
    sp_parm.pitchrate = 100;               //变调比例, 128:为不变调, <128:音调变高, >128:音调
    变低, 声音更沉
    //speedout/speedin 的结果大于 0.6 小于 1.8.变快变慢最好都不要超过 2 倍
    sp_parm.speedin = 80;
    sp_parm.speedout = 80;                 //sp_parm.speedin: sp_parm.speedout 为变速比例, 例如
    speedin=1;speedout=2;则为变慢 1 倍
    sp_parm.quality = 4;                   //变调运算的运算质量, 跟速度成正比, 配置范围 3-8
    sp_parm.pitchflag = 1;                 //如果 pitchflag=1,就是输入输出速度只由 speedin: speedout
    决定,调节音调的时候,速度比例保持不变.例如设置 speedout:speedin=1:2 ,这时候,去调节 pitchrate,
    都是只变调, 速度一直是保持放慢 1 倍。
    //如果 pitchflag=0, 就是变速变调比例分开控制 , 这样实际变速的比例
    是 (speedout/speedin)*pitchrate 了。

    return speed_phy(obuf, &sp_parm, ppsound);
}
```

## 21.烧录器升级以后旧版 SDK 没法烧录-211019hwx

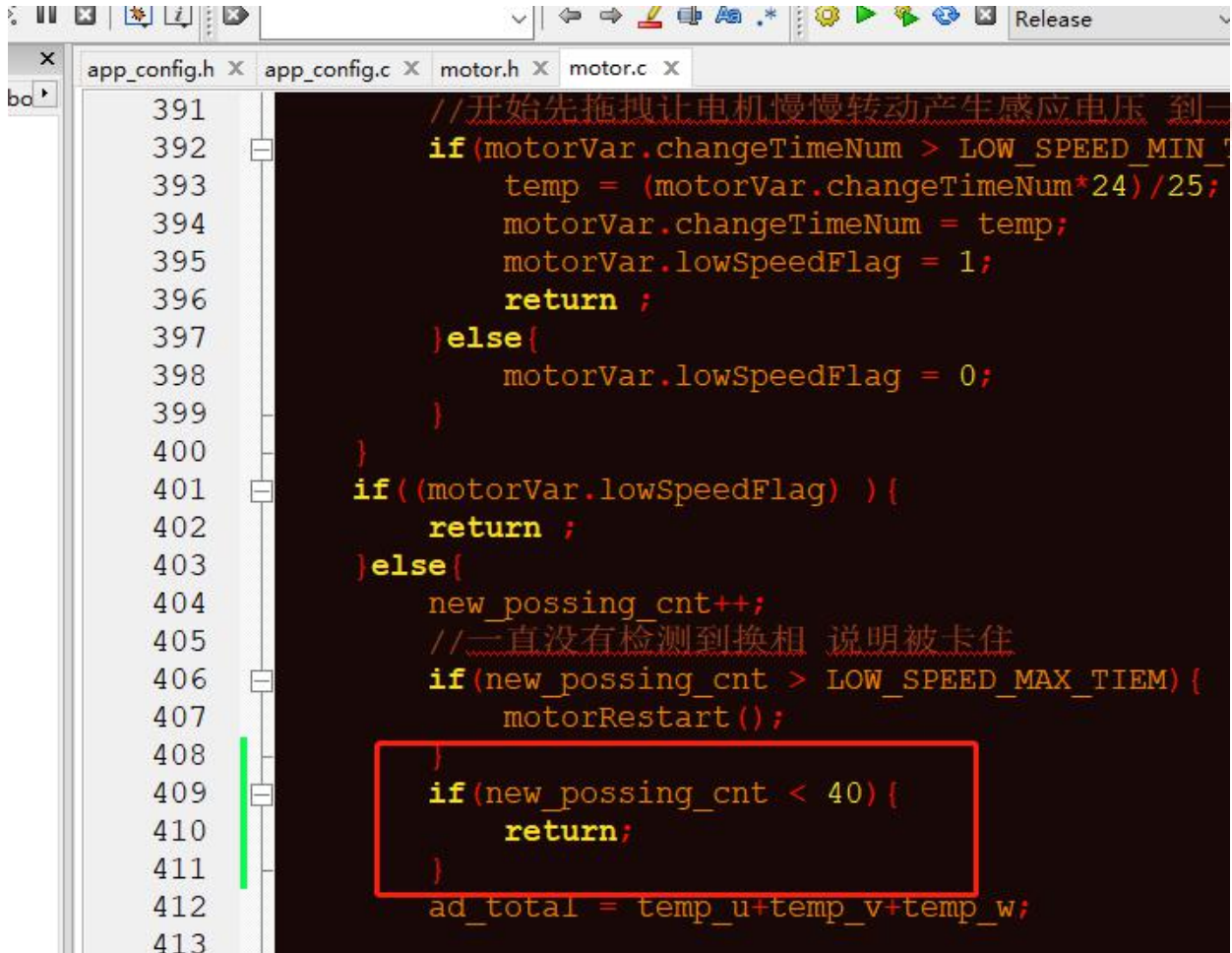
参考最新的 SDK 在 INI 文件里面加上芯片版本控制

[CHIP\_VERSION]

SUPPORTED\_LIST=A, B, C

## 22.电机驱动更新-211116hwx

增加一个换相时间的判断，解决在电机阻力突然变大的时候会有换相检测不准导致电机扭矩太小容易被卡停的问题。这个参数根据电机额定转速 KV 值电池电压等参数实际调整，总之就是电机最快的转速也不可能有这么快的换相速度。



```
391 //开始先拖拽让电机慢慢转动产生感应电压 到一
392 if(motorVar.changeTimeNum > LOW_SPEED_MIN_T
393     temp = (motorVar.changeTimeNum*24)/25;
394     motorVar.changeTimeNum = temp;
395     motorVar.lowSpeedFlag = 1;
396     return ;
397 }else{
398     motorVar.lowSpeedFlag = 0;
399 }
400 }
401 if((motorVar.lowSpeedFlag) ){
402     return ;
403 }else{
404     new_posing_cnt++;
405     //一直没有检测到换相 说明被卡住
406     if(new_posing_cnt > LOW_SPEED_MAX_TIEM) {
407         motorRestart();
408     }
409     if(new_posing_cnt < 40) {
410         return;
411     }
412     ad_total = temp_u+temp_v+temp_w;
413 }
```



## 23.实时变声效果说明-211118hwx

- 1.仔细阅读 SDK 文件夹下的 SDK 手册关于音效部分说明文档。
- 2.V110 版本 SDK 的 SIMPLE 工程开机默认就是在变声状态，只需要设置变声的模式即可。
- 3.变声效果可以自己调整 vo\_pitch\_api.c 里面的每个音效对应的参数设置。
- 4.添加函数切换变声效果

```
void user_change_vp(u8 cmd){
    VOICE_PITCH_PARA_STRUCT vc_parm;
    VOICEPITCH_STUCT_API *ops;
    r_printf("user_change_vp:%d ",cmd);
    vp_cmd_case(cmd, &vc_parm);
    ops = get_vopitch_context();           //获取变采样函数接口
    OS_ENTER_CRITICAL();
    ops->open(&VP_BUFLen[0], &vc_parm, NULL);
    OS_EXIT_CRITICAL();
}
```