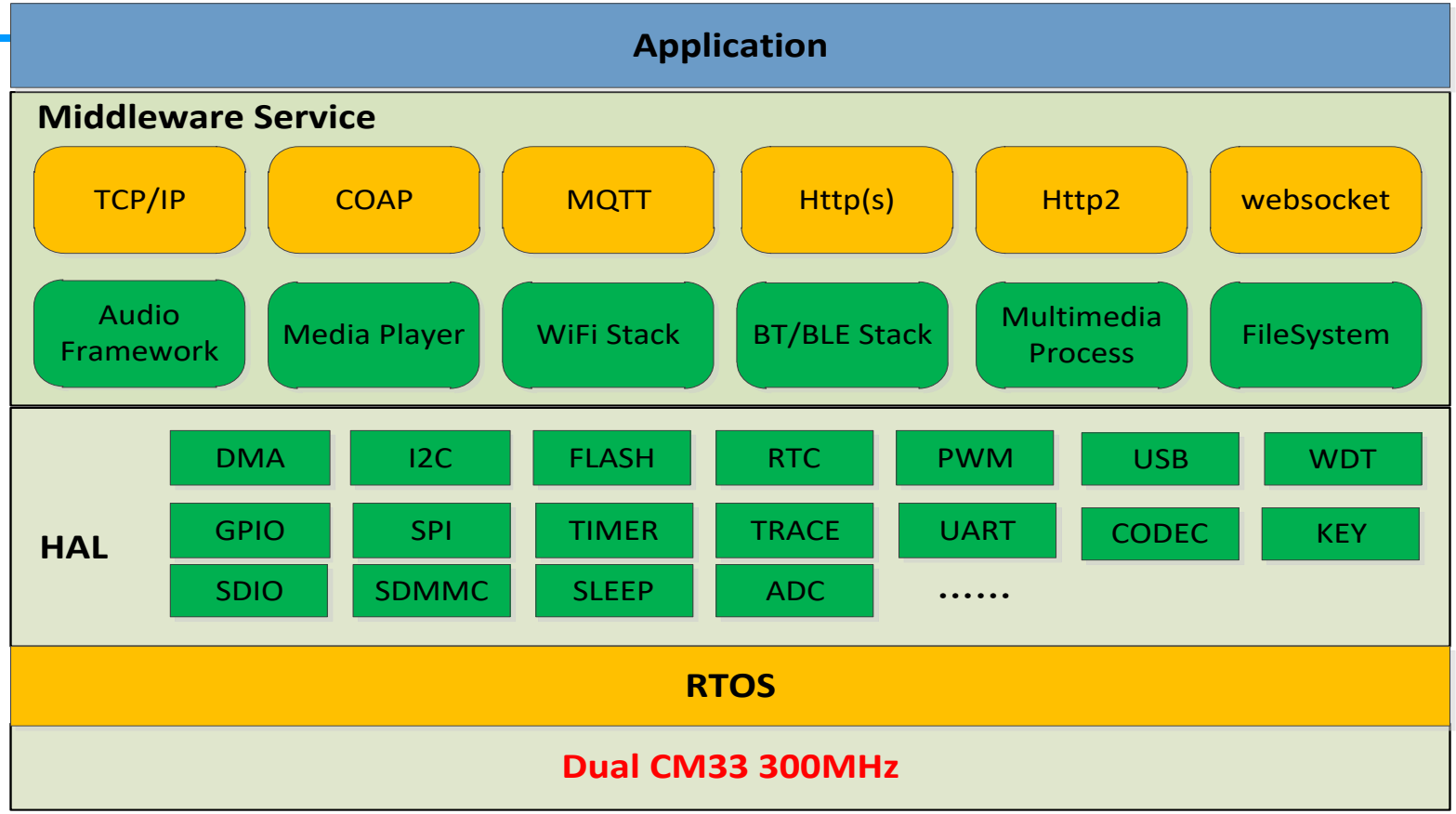
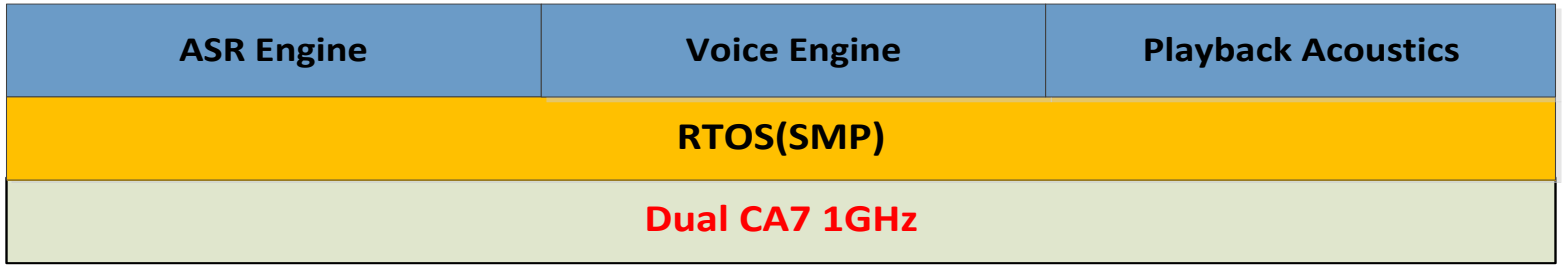
BES2600W SDK user guide

2021.07

目录

总体架构





Legend:

- Property Software
- Open Source Software
- 3rd Party Software

SDK目录

apps	Application implementations
config	Customized configurations for different projects
mbed	MBED framework codes (now for File System only)
net	Including lwip、coap、libcurl、mbedtls、nghttp2、librws
platform	Plantform related including cmsis、driver、hal、dsp
rtos	ARM FreeRTOS & RTX codes (support posix)
scripts	Building framework scripts
services	Service codes
tests	Test programs
utils	Common utility codes
Makefile	

SDK目录对应的功能

应用程序入口
apps/main/apps.cpp ->app_init
->apps/bes_test/master_app.c

apps目录下主要是应用功能实现。
如：wifi，bt，audio等等的应用代码里面都会有一些参考。

config目录下是每一个目录对应的就是一个工程。
虽然是同一套SDK，但是不同的项目，可能有的功能需要，有的功能不需要，只需要根据项目的需要，配置对应的target.mk文件。

net目录下是wifi芯片上运行的网络协议。
如：802.11，lwip，libcurl，coap等等网络协议模块。

service目录下是可以给应用层提供使用的接口。
包括音频相关，网络相关，文件系统，flash相关等等。

编译环境设置:

主机系统： ubuntu 16.04/ubuntu 18.04

https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm

gcc-arm-none-eabi-9-2019-q4-major

1. 解压对应的工具链:

如：解压到~/toolchain/目录

2. 设置编译toolchain PATH：

echo export PATH=~/toolchain/gcc-arm-none-eabi-9-2019-q4-major/bin:$PATH >> ~/.bashrc

source ~/.bashrc

确认编译环境是否生效：which arm-none-eabi-gcc 和 arm-none-eabi-gcc --version

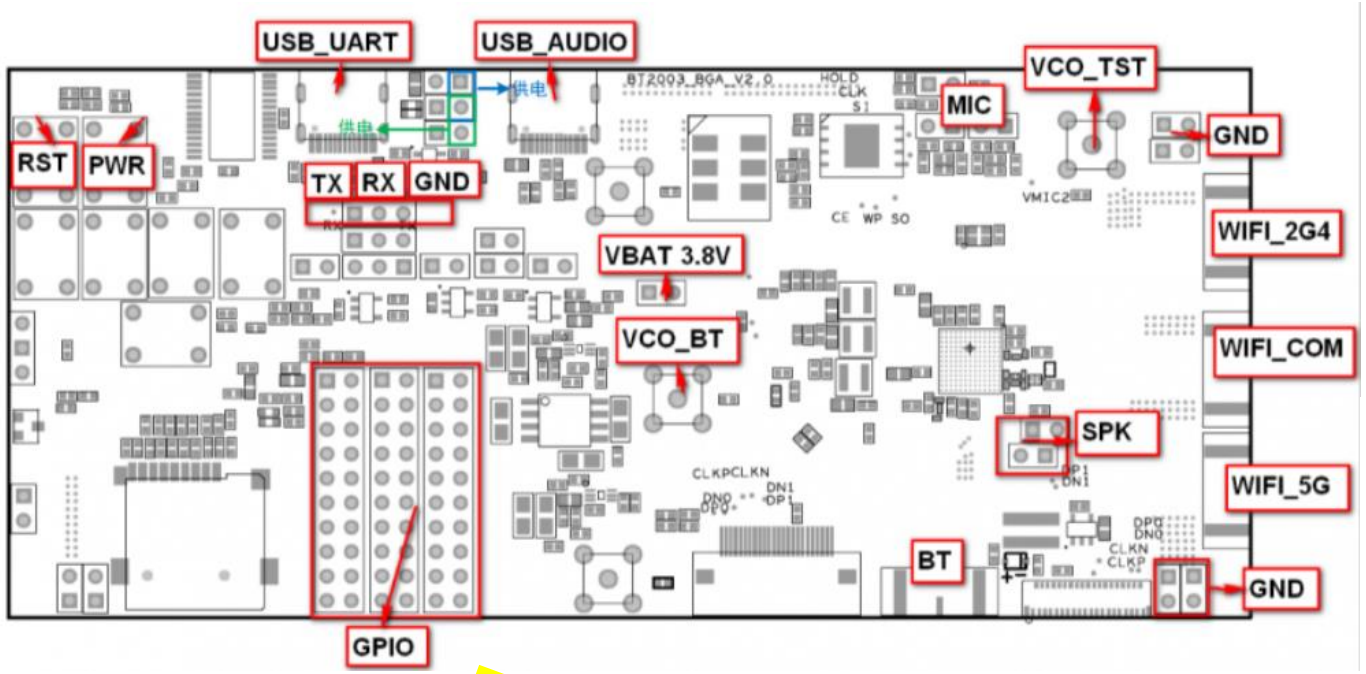
SDK编译

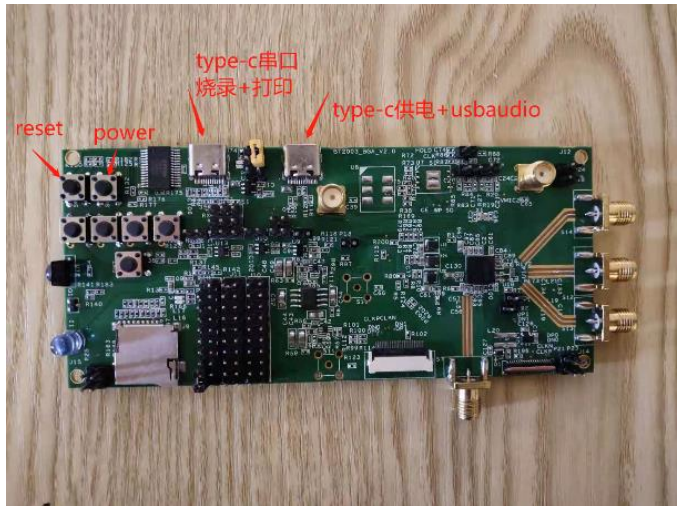
3. 设置编译环境后，在SDK目录下，运行对应的编译脚本：

编译脚本：tools/best2600w_aiot.sh SDK=1

输出文件：out/best2600w_aiot/best2600w_aiot.bin

EVB DEMO硬件连接





HAL_IOMUX_PIN_P0_0 对应图中gpio 00
HAL_IOMUX_PIN_P1_1 对应图中gpio 11
HAL_IOMUX_PIN_P2_2 对应图中gpio 22
依此类推

GPIO MAP

V_SSD	V_SSD	GND	02	12	26
RST	GND	GND	03	13	27
SSD_D0	GND	GND	04	14	30
SSD_D2	GND	GND	05	15	31
SSD_D1	GND	GND	06	20	32
–	GND	GND	07	21	33
SSD_D3	GND	GND	10	22	34
V_SSD	GDN	GND	11	23	–
–	GND	GND	00	24	36
–	GND	GND	01	25	37

固件烧录

烧录工具：dldtool.exe

接口：串口，波特率1.5M

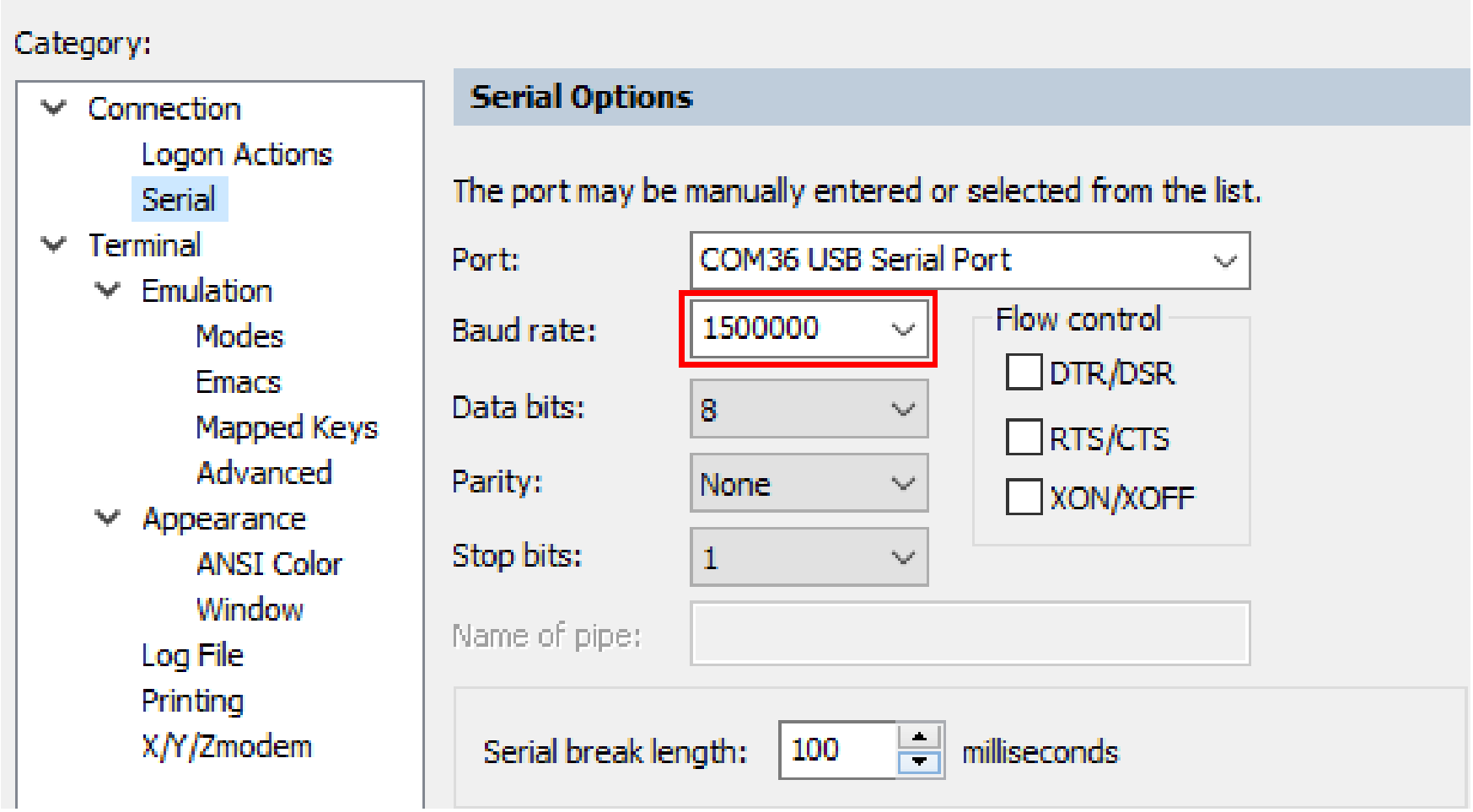
烧录命令：dldtool.exe {com_num} .\programmer2003.bin .\...\bes2600w_aiot.bin

例如： dldtool.exe 8 programmer2003.bin E:/bin/best2600w_aiot.bin

注意运行烧录命令再给EVB上电或者按reset键启动烧录

调试串口设置

1. 调试串口默认采用**1.5M波特率**，清按下图进行配置



启动Log举例

```
KERNEL=FREERTOS
CRASH_DUMP_SIZE=0
AUD_SEC_SIZE=0
USER_SEC_SIZE=0x1000
FACT_SEC_SIZE=0x1000
NV_REC_DEV_VER=2
FLASH_BASE=0x28000000
FLASH_SIZE=0x1000000
OTA_CODE_OFFSET=0
CRC32_OF_IMAGE=0x00000000
BUILD_DATE=Jun  9 2021 14:57:00
REV_INFO=:best2600w_aiot


------
METAL_ID: 0
------

      963/I/MAIN  /  1 | FLASH_ID: C8-60-18
      963/I/NONE  /  1 | sram_text start 20005d80 size 3b020
      963/I/NONE  /  1 | fram start 2016a000 size 9880
      964/I/NONE  /  1 | Warning: Bad dc calib efuse L: 0x0000
      964/I/NONE  /  1 | Warning: Bad dc calib efuse R: 0x0000
      964/I/NONE  /  1 | ANA: DC CALIB L=0x0000/0 R=0x0000/0
      964/I/NONE  /  1 | please check all sections sizes and heads is correct
      965/I/NONE  /  1 | __coredump_section_start: 0x28ffb000 length: 0x0
      965/I/NONE  /  1 | __ota_upgrade_log_start: 0x28ffb000 length: 0x0
      965/I/NONE  /  1 | __log_dump_start: 0x28ffb000 length: 0x0
      965/I/NONE  /  1 | __crash_dump_start: 0x28ffb000 length: 0x0
      965/I/NONE  /  1 | __custom_parameter_start: 0x28ffb000 length: 0x1000
      965/I/NONE  /  1 | __lhdc_license_start: 0x28ffb000 length: 0x0
      965/I/NONE  /  1 | __userdata_start: 0x28ffc000 length: 0x2000
      965/I/NONE  /  1 | __aud_start: 0x28ffe000 length: 0x0
      965/I/NONE  /  1 | __factory_start: 0x28fff000 length: 0x1000
      965/I/NONE  /  1 | app_init

      966/I/NONE  /  1 | app_tws_ibrt_bandwidth_table_register 0x34002c68
```

AT命令行调试

SDK支持的AT命令列表可通过AT+HELP列出
参考：services/wifi_app/wifi_console/readme_en.txt

```
A7_TRACE:A7 capture cnt:15061
A7_TRACE:A7 capture cnt:15121
   970250/-16E | AT+A7DUMP        - a7 data to ftp server
   970250/-16E | AT+ECHO               -set at cmd echo disable/enable
   970251/-16E | AT+EPTA        - config epta params(WiFi/bt)
   970251/-16E | AT+FCC="FED2"    - config freq cal cfg
   970251/-16E | AT+FREEHEAPSIZE         - show RTOS/application free heap size
   970251/-16E | AT+HEAP       - show thread heap malloc-free summary
   970251/-16E | AT+HELP                - show AT CMD list
   970251/-16E | AT+MD                 - read mem or reg
   970251/-16E | AT+MSLEEP          - sleep for ms
   970251/-16E | AT+MW                 - write mem or reg
   970251/-16E | AT+NDNS                - do dns
   970251/-16E | AT+NIPERF          - start iperf test
   970251/-16E | AT+NPING           - do ping
   970251/-16E | AT+NTCPCLI         - start tcp client
   970251/-16E | AT+NTCPSER         - start tcp server
   970251/-16E | AT+NUDPCLI         - start udp client
   970251/-16E | AT+NUDPSER         - start udp server
   970251/-16E | AT+RBNVREC         - rebuild nvrecord
   970251/-16E | AT+RECDUMP        - mic record data to ftp server
   970251/-16E | AT+RUNTIME              - show run time statistics
   970251/-16E | AT+SETGPIO                     - set gpio output low/high
   970251/-16E | AT+SHOOTTSK=tsk_name       - trace dedicate thread heap malloc-free
   970251/-16E | AT+SNSEND           - send management frame
   970251/-16E | AT+SNSETCH           - set sniffer channel
   970251/-16E | AT+SNSTART           - start sniffer
   970251/-16E | AT+SNSTOP           - stop sniffer
   970251/-16E | AT+THREAD              - show thread information
   970251/-16E | AT+WSACONF         - add AP config
   970251/-16E | AT+WSACONN         - start wifi auto connect
   970251/-16E | AT+WSAIRKISS       - start airkiss
   970251/-16E | AT+WSCONN          - start wifi connect
   970251/-16E | AT+WSDCONF         - del AP config
   970251/-16E | AT+WSDISCONN           - disconnect AP
   970251/-16E | AT+WSGCCONF           - get current AP config
   970251/-16E | AT+WSGLINKSIG      - get current link signal (RSSI)
   970252/-16E | AT+WSGSCONF           - get saved AP config
   970252/-16E | AT+WSGSTA          - get connect state
   970252/-16E | AT+WSSCAN          - scan AP
   970252/-16E | AT+WSSCONF         - save AP config
   970252/-16E | AT+WSSETIP         - set static ip addr
   970252/-16E | AT+WSSETRECON      - set reconnect policy
   970961/-16E | CPU USAGE: busy=10 light=90 sys_deep=0 chip_deep=0
A7_TRACE:A7 capture cnt:15181
```

AT命令连接WIFI AP

AT+WSCONN=SSID,PASSWORD

如： AT+WSCONN= BES_SZ_CH11, BES2000S

连接上AP会显示成功和获取到的IP地址，如下图：

```
531853/-16E | [wsm_oper]wsm_set_pm 0x81.
531853/-16E | EAPOL: SUPP_BE entering state IDLE
531853/-16E | eap_peer_sm_step 1485.
531853/-16E | eap_peer_sm_step 1485.
531853/-16E | [WIFI] status CONNECTING, event 1
531853/-16E | wifi connecting in state 9
531853/-16E | [WIFI] status CONNECTING, event 2
531853/-16E | wifi mac connected to BES_offices
531853/-16E | [WIFI] change status:CONNECTING->CONNECTED
531853/-16E | [wsm_oper]wsm_epta_cmd 60000 40000 3.
531861/-16E | [wsm_oper]wsm_set_pm_indication.
531861/-16E | Set PS mode 0
531861/-16E | nl80211_set_power_save disabled
531861/-16E | ieee80211_recalc_channel 0 12773 11.
531861/-16E | cw1200_bss_info_changed=0x10000 hw_priv:0x20044af0 priv:0x2004589c

531861/-16E | [STA] BSS_CHANGED_PS Aid: 3, setbssparams_done: 1, Joined: yes(2), Powersave: WSM_PSM_ACTIVE

531862/-16E | [wsm_oper]wsm_set_pm 0x0.
531862/-16E | link up
531862/-16E | [WIFI] change status:CONNECTED->DHCPING
531862/-16E | config_dpd, ch=11
531864/-16E | [wsm_oper]wsm_set_pm_indication.
531887/-16E | bwifi send event, id:5
531887/-16E | [WIFI] status DHCPING, event 5
531887/-16E | get ip:10.75.30.21
531887/-16E | get mask:255.255.255.0
531887/-16E | get gw:10.75.30.254
531887/-16E | cw1200_bss_info_changed=0x1000 hw_priv:0x20044af0 priv:0x2004589c

531887/-16E | [STA] BSS_CHANGED_ARP_FILTER enabled: 1, cnt: 1

531887/-16E | set epta w:100000 hw:2 wc:60000 fbit:0 fw:0
531887/-16E | [wsm_oper]wsm_epta_cmd 100000 0 2.
531887/-16E | [WIFI] change status:DHCPING->GOT_IP
531887/-16E | connect complete, success
531887/-16E | +ok
531986/-16E | ieee80211_recalc_channel 0 12773 11.
A7_TRACE:A7 capture cnt:8341
A7_TRACE:A7 capture cnt:8401
540886/-16E | CPU USAGE: busy=17 light=83 sys_deep=0 chip_deep=0
```

M33程序入口

在apps\bes_test\master_app.c为m33测试程序主文件

master_app.c

2600w_aiot_sdk > apps > bes_test > master_app.c > user_app_init(void)

```c
#define MAIN_SSID "bes_test"
#define MAIN_PASSWD "12345678"

void  master_main(const void *argument)
{
    TRACE(0, "enter %s %d ",__func__,__LINE__);
    app_sysfreq_req(APP_SYSFREQ_USER_APP_15, APP_SYSFREQ_390M);
    TRACE(0, "sys freq calc: %d", hal_sys_timer_calc_cpu_freq(5, 0));

```

A7程序入口

A7主要用于集成前端音频算法及唤醒模型

主要文件位于services/a7_dsp目录内，与主m33的通信框架，录音控制等功能已经实现

可参考a7_dummy_main.c以及mcu_dummy_main.c中的接口快速集成第三方算法

若第三方算法需要编译成静态库，必现采用如下编译选项：

-marm -march=armv7-a -mfpu=neon-vfpv4 -mfloat-abi=hard -mtune=cortex-a7 -mlittle-endian

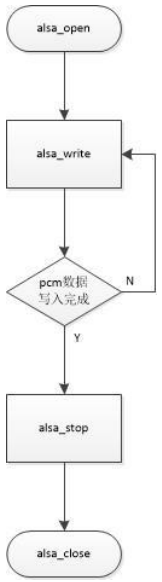
```
a7_dsp/
├── common
│   ├── a7_audio.c
│   ├── a7_cmd.c
│   ├── a7_cmd.h
│   ├── a7_main.c
│   ├── aud_dump.c
│   ├── aud_dump.h
│   ├── audio_dump_single_channel.c
│   ├── audio_dump_single_channel.h
│   ├── data_dump.c
│   ├── data_dump.h
│   ├── decom
│   │   ├── dsp_code_decom.c
│   │   ├── lzma_decode.c
│   │   ├── lzma_decode.h
│   │   └── Makefile
│   ├── heart_beat.c
│   ├── Makefile
│   ├── mcu_audio.c
│   ├── mcu_audio.h
│   ├── mcu_cmd.c
│   ├── mcu_slave_code.S
│   ├── mcu_slave_init.c
│   └── my_a7_test.c
├── Makefile
└── prj
    └── dummy
        ├── a7_dummy_main.c
        ├── dummy
        │   ├── dummy_main.c
        │   ├── lib
        │   │   └── lib_a7_dummy.a
        │   └── Makefile
        ├── Makefile
        └── mcu_dummy_main.c
```

alsa 接口说明

接口头文件位置：services/wifi_app/alsa_adapter/alsa.h

```
alsa_handle_t * alsa_open(alsa_mode_t mode, int sample_rate, int channels, alsa_pcm_format_t format);
int alsa_write(alsa_handle_t * h, uint8_t *buf, uint32_t size);
int alsa_start(alsa_handle_t * h);
int alsa_stop(alsa_handle_t * h);
int alsa_close(alsa_handle_t * h);
void alsa_register_pcm_state_callback(alsa_handle_t * h, alsa_pcm_state_callback_t cb, void * arg);

void alsa_mute_set(void);
void alsa_mute_cancel(void);
void alsa_volume_set(uint8_t vol_dac);
uint8_t alsa_volume_get(void);
```



alsa_open	创建播放实例，返回实例句柄。目前最大支持3个播放实例。需配置PCM数据采用率及声道参数： • 如果非48k音频，内部会重采样至48k采样率 • 如果是双声道音频，内部会转换为单声道音频
alsa_write	写入需要播放的pcm数据。该接口为阻塞接口，完成内部音频处理并在全部写入数据进入播放队列后会返回。写入满足起播条件的数据量后自动起播（故无需调用alsa_start）
alsa_stop	通知内部停止该实例播放，调整内部状态机。该接口不会阻塞。 需要在alsa_close前调用，以保证内部状态机正常。
alsa_close	销毁一个播放实例。会阻塞至已写入的数据全部播放后返回。
alsa_mute_set	设置静音。全部播放实例将静音。
alsa_volume_set	设置音量。参数为tgt_hardware.c中配置的dac增益数组的序号。全部播放实例将按照该设定音量输出。

THANK YOU