Airoha IoT SDK Application Developers Guide

Version: 2.1.1

Release date: 29 September 2021

Document Revision History

Revision	Date	Description
1.0	31 July 2019	• Initial release
1.0.1	14 August 2019	• Added support for a different key action mapping table and configurable key action mapping table in key_remapper. • Add support for synchronizing the LED patterns from Agent to Partner.
2.0.0	01 July 2020	• Added support for 1565 and 1568 • Added the description for the Ear Detection APP and Smart Charger APP • Added the event APP_FIND_ME_EVENT_ID_TRIGGER to support triggering find me by other APPs.
2.1.0	23 November 2020	• Added the description for the MCSync share APP.
2.1.1	29 September 2021	• Added Race CMD of changing sidetone • Added power save app and auto power off when silence detection.

Table of Contents

Lists of Tables and Figures

1. Overview

This document helps developers to understand the design and code structure of Airoha IoT SDK application layer, and provides methods to do customizations.

This chapter introduces the system architecture of application layer of Airoha IoT SDK, include types of modules and the file structures. Based on this, the following chapters will illustrate the detail use of the modules and the customizations.

1.1. System architecture

As shown in Figure 1, the application layer of Airoha IoT SDK is composed of three kinds of modules: Event senders, APPs, and Config. Event senders send UI shell events to APPs. APPs receive the events and take actions to complete the applications. Configs manage the usage of keys, LEDs, and VP. Besides, there are also utilities such as the debugging tool and AWS sync senders which help fulfill modules.

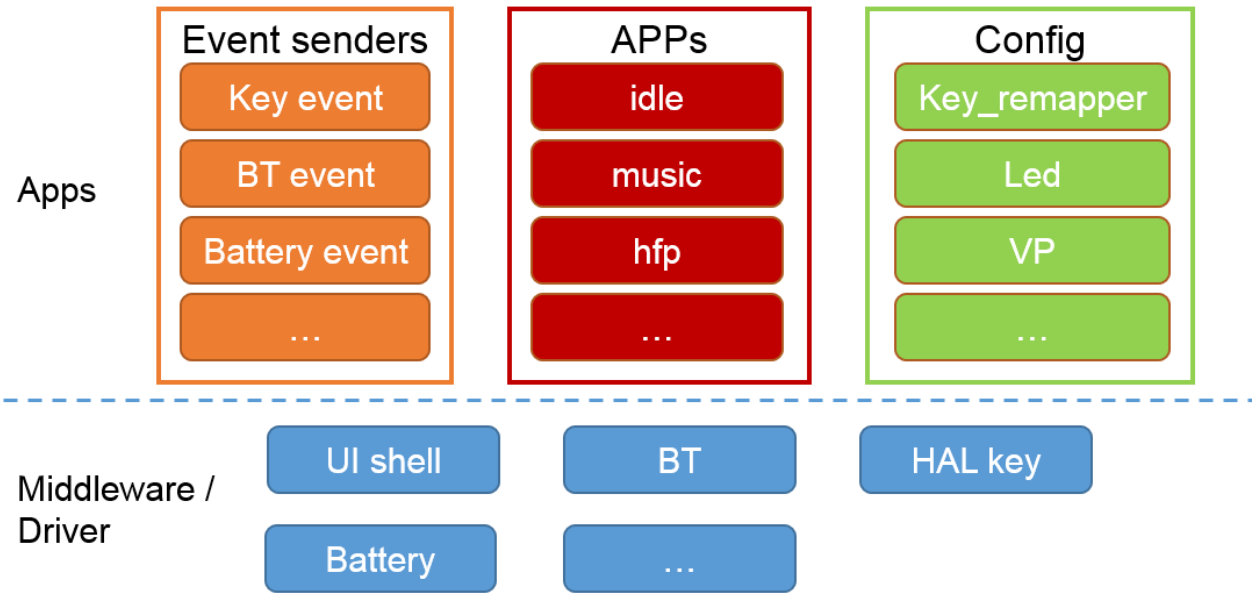


Figure 1. System architecture of 155X earbuds application layer

Event Senders include key events, BT events, battery events and others. These modules register callbacks in middleware or hardware abstraction layer modules. When the callbacks are executed, the events are sent to APPs. Take key event as an example:

- Key event sender registers callback in HAL key module. It sends an event to UI shell when called. After receiving the event sent by the key event sender, the UI shell sends it to APPs.

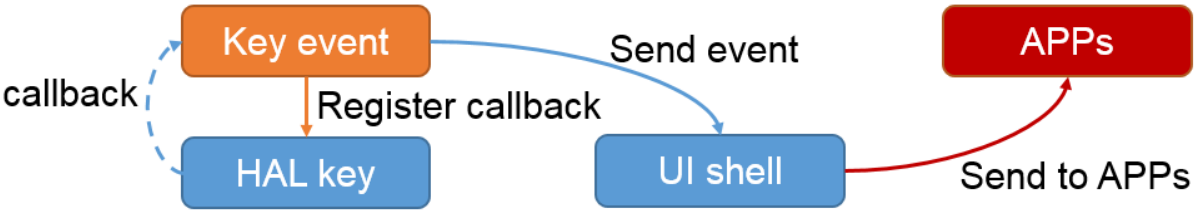


Figure 2. Key event sender

There are multiple folders named “app_***” as APPs, which implement UI logic. An APP is able to send events to another one for the request of taking some actions, or to notify the happening of some events.

Configs include Key Remappers, LED manager, and VP manager. Key Remappers provide a clear view of key usage in a table. For example, under connected status, short click is converted to KEY_AVRCP_PLAY.

1.2. Folder structure

The folders are listed below for your reference. Users who are interested in the detailed design of Airoha IoT SDK application can refer to the source code.

- APPs
  - /mcu/project/<board>/apps/<project>/src/apps/app_*
- Configs
  - /mcu/project/<board>/apps/<project>/src/apps/config
- Event sender
  - /mcu/project/<board>/apps/<project>/src/apps/events
- LED
  - /mcu/project/<board>/apps/<project>/src/apps/led
- VP
  - /mcu/project/<board>/apps/<project>/src/apps/vp
- Utils
  - /mcu/project/<board>/apps/<project>/src/apps/utils

2. Module introduction

Modules in event senders, APPs, and config are illustrated here.

2.1. Event senders

Event senders include three modules: battery event, BT event, and key event. These three modules are responsible for sending different events.

2.1.1. Battery event

This module sends EVENT_GROUP_UI_SHELL_BATTERY group events. The event IDs in this group are defined in /mcu/project/<board>/apps/<project>/inc/apps/events/app_events_battery_event.h as follow types:

- APPS_EVENTS_BATTERY_PERCENT_CHANGE
- APPS_EVENTS_BATTERY_CHARGER_STATE_CHANGE
- APPS_EVENTS_BATTERY_CHARGER_EXIST_CHANGE
- APPS_EVENTS_BATTERY_SHUTDOWN_STATE_CHANGE.

When the voltage of the device is less than the shutdown voltage, the battery event sends APPS_EVENTS_BATTERY_SHUT_DOWN_STATE_CHANGE to APPs.

2.1.2. BT event

This module sends three groups of events. The EVENT_GROUP_UI_SHELL_BT_SINK group is frequently used by APPs to get information about BT. The EVENT_GROUP_UI_SHELL_BT_CONN_MANAGER group and the EVENT_GROUP_UI_SHELL_BT group are helpful when the EVENT_GROUP_UI_SHELL_BT_SINK group cannot provide enough information.

- EVENT_GROUP_UI_SHELL_BT_SINK group events – The event IDs of the group is defined in sink module in middleware. The BT state change and AWS connection change are notified by these events.
- EVENT_GROUP_UI_SHELL_BT_CONN_MANAGER group events – The event IDs of the group are defined in BT connection manager module in middleware. This group currently only contains the BT_CONNECTION_MANAGER_EVENT_AIR_PAIRING_COMPLETE event which is used to notify that the air pairing process is complete.
- EVENT_GROUP_UI_SHELL_BT group events – The event IDs of the group are defined in Bluetooth module in middleware. The event sender callback is registered where the app needs the BT messages.

2.1.3. Key event

This module sends EVENT_GROUP_UI_SHELL_KEY group events. The event id of the group is composed of *key_id* and *airo_key_event*. For example, if the key_id is DEVICE_KEY_POWER (0x18) and the airo_key_event is AIRO_KEY_SHORT_CLICK (0x01), the event id is 0x0118. Please refer to /mcu/project/<board>/apps/<project>/src/apps/events/app_events_key_events.c.

2.2. APPs

APPs implement the actions of the applications layer, and are organized by features and functions. Each APP is composed of one or more activities. The activities are managed by UI shell. Every activity receives events of the EVENT_GROUP_UI_SHELL_SYSTEM group to follow the management to create, destroy, resume, pause, refresh

and result. Please refer to Chapter 2.3 of the "Airoha_IoT_SDK_UI_Framework_Developers_Guide" document to get more about EVENT_GROUP_UI_SHELL_SYSTEM.

Here we introduce the APPs in Airoha IoT SDK, customers can add others based on requirements.

2.2.1. Battery APP

This APP receives battery events and controls a state machine to support the features:

- Trigger system power off by sending REQUEST_POWER_OFF event when the battery is very low.
- Trigger RHO (Role Hand Over) when battery percentage of Agent is lower than Partner by sending TRIGGER_RHO event. To support this, Partner sends AWS data with the battery capacity/change of battery level or change of charger in status to Agent when AWS connected.
- Disable BT when charging in and enable BT when charging out by sending REQUEST_ON_OFF_BT events.
- Display LED pattern or play VP while in *Low capacity* or *charging* status.

The state definition of the battery APP is shown in Table 1. Battery APP Status Definition. The charger state 0 is CHARGER_STATE_CHR_OFF, 4 is CHARGER_STATE_EOC. APPS_BATTERY_LOW_THRESHOLD and APPS_BATTERY_FULL_THRESHOLD are defined in /mcu/project/<board>/apps/<project>/inc/apps*/apps_customer_config.h*.

Table 1. Battery APP Status Definition

Charger_exsit	Charger_state	Shut down state	Battery percent	Battery app Status
TRUE	= 0 or 4	N/A	N/A	Charging full
TRUE	!= 0 && != 4	N/A	N/A	Charging
FALSE	N/A	TRUE	N/A	Shut down
FALSE	N/A	FALSE	< low threshold	Low capacity
FALSE	N/A	FALSE	>= full threshold	Full
FALSE	N/A	FALSE	>= low threshold && < full threshold	Idle

Battery APP is composed of two activities: app_battery_idle_activity and app_battery_transient_activity. App_battery_idle_activity is created when the device is powered on, and app_battery_transient_activity is created by app_battery_idle_activity when a LED pattern is needed in a new state but not in old state.

App_battery_idle_activity receives events of group EVENT_GROUP_UI_SHELL_BATTERY, EVENT_GROUP_UI_SHELL_BT_SINK and EVENT_GROUP_UI_SHELL_APP_INTERACTION, in addition to EVENT_GROUP_UI_SHELL_SYSTEM.

- EVENT_GROUP_UI_SHELL_BATTERY:
    - Contains states of battery percent change, states of battery charge. App_battery_idle_activity manage battery app status according to these events and complete the features listed above.

- EVENT_GROUP_UI_SHELL_BT_SINK:
    - Contains the state change of BT connection, AWS connection, and profile connection update, which helps app_battery_idle_activity manage the context and battery app status.
    - For Partner, if an AWS connection is changed to connected, app_battery_idle_activity will notify agent of its battery level. For agent, if receives a Partner's battery level report, app_battery_idle_activity should check whether the device should do RHO.
- EVENT_GROUP_UI_SHELL_APP_INTERACTION
    - Receives APPS_EVENTS_INTERACTION_RHO_END event and refreshes the local context and battery app status.

2.2.2. Homescreen APP

Homescreen APP works as settings. App_homescreen_idle_activity is the only activity of Homescreen APP, which is the priority when the system is in idle state.

- Homescreen APP processes the common key event requests, such as power_on, power_off, BT_discoverable, air_pairing, reconnect last device, reset paired device, and RHO to agent.
- Before disabling BT, power off or reboot Agent. Homescreen APP may trigger RHO by sending TRIGGER_RHO.
- If the device is disconnected from a smartphone and waits for the time APPS_TIMEOUT_OF_SLEEP_AFTER_NO_CONNECTION (defined in /mcu/project/<board>/apps/<project>/inc/apps*/apps_customer_config.h*), Homescreen APP automatically triggers sleep mode.

The homescreen_idle_activity contains a BT connection component, which helps Homescreen APP to process BT and BT sink events, and change the connection state of Homescreen APP. The relationship of BT state and connection state is shown in Table 2. Definitions of the BT State in the Homescreen APP:

- The connection state of Homescreen APP determines the LED patterns and voice prompt playback.
- On 1565/1568, the BT sink state cannot express the BT connection state or BT power state, so we use BT_CM_EVENT_REMOTE_INFO_UPDATE and BT_CM_EVENT_POWER_STATE_UPDATE of BT connection manager events to decide the BT state.
- On 155x, specifically for Partner, BT sink state changing to CONNECTED means AWS is connected instead of connected to the smartphone.

Table 2. Definitions of the BT State in the Homescreen APP

BT state	Profile connected	BT visible	Connection state
NONE	N/A	N/A	BT OFF
POWER_ON	N/A	TRUE	Discoverable (Only on Agent)
POWER_ON	N/A	FALSE	Non-discoverable
>= CONNECTED	FALSE	FALSE	Non-discoverable
>= CONNECTED	TRUE	FALSE	Connected to SP

2.2.3. RHO APP

This APP receives and processes TRIGGER_RHO request, and sends RHO_STARTED and RHO_END events to notify other APPs of the RHO status. The APP will start RHO only when the AWS connection is established as well as BT is connected to the smartphone. Otherwise, it immediately sends a RHO_END event.

App_rho_idle_activity process events of groups EVENT_GROUP_UI_SHELL_BT_SINK and EVENT_GROUP_UI_SHELL_APP_INTERACTION in addition to EVENT_GROUP_UI_SHELL_SYSTEM.

- EVENT_GROUP_UI_SHELL_BT_SINK
    - Contains state change of the BT connection, AWS connection and profile connection update, which helps app_rho_idle_activity manage local context. The context remembers the state of BT sink, and helps the APP to determine whether RHO can be started now.
- EVENT_GROUP_UI_SHELL_APP_INTERACTION
    - Contains TRIGGER_RHO and RHO_END. The APP checks the states and do RHO when it receives TRIGGER_RHO event.

2.2.4. HFP APP

This APP receives and processes events of BT_SINK, KEY, and BATTERY groups, and contains two activities: *app_hfp_idle_activity* and *app_hfp_activity*. The former is created when the device is powered on. The latter starts when BT sink state changes to the range of HFP, and is completes by itself when BT sink state changes to CONNECTED or POWER_OFF.

App_hfp_idle_activity processes events of groups EVENT_GROUP_UI_SHELL_KEY, EVENT_GROUP_UI_SHELL_BT_SINK and EVENT_GROUP_UI_SHELL_BATTERY in addition to EVENT_GROUP_UI_SHELL_SYSTEM.

- EVENT_GROUP_UI_SHELL_KEY
    - This group is used to notify that the voice assistant waked up and sets VP playback, or redials the last call.
- EVENT_GROUP_UI_SHELL_BT_SINK
    - When BT_SINK_SRV_EVENT_STATE_CHANGE event is received, app_hfp_idle_activity checks the state and starts app_hfp_activity.
    - When BT_SINK_SRV_EVENT_PROFILE_CONNECTION_UPDATE event is received and notify that the profile connected, app_hfp_idle_activity reports battery level to the remote device.
- EVENT_GROUP_UI_SHELL_BATTERY
    - When APPS_EVENTS_BATTERY_PERCENT_CHANGE is received, the APP reports the battery level to the smartphone if it is connected. The smartphone shows a battery icon in the status bar if it supports this feature.

App_hfp_activity processes events of groups EVENT_GROUP_UI_SHELL_KEY, EVENT_GROUP_UI_SHELL_BT_SINK and EVENT_GROUP_UI_SHELL_APP_INTERACTION besides EVENT_GROUP_UI_SHELL_SYSTEM.

- EVENT_GROUP_UI_SHELL_KEY

- When received events of EVENT_GROUP_UI_SHELL_KEY group, app_hfp_activity will execute actions by processing key events and the current state according to Table 3. Key Event Configurations. The configurations are in the file /mcu/project/<board>/apps/<project>/src/broads/<board>/customerized_key_config.c.

Table 3. Key Event Configurations

HFP State	Short key	Double key
BT_SINK_SRV_STATE_INCOMING	Accept Call	Reject Call
BT_SINK_SRV_STATE_OUTGOING	End Call	Hold Call
BT_SINK_SRV_STATE_ACTIVE	End Call	Hold Call
BT_SINK_SRV_STATE_TWC_INCOMING	Accept Call	Reject Call
BT_SINK_SRV_STATE_TWC_OUTGOING	N/A	Hold Call
BT_SINK_SRV_STATE_HELD_ACTIVE	End Call	Hold Call
BT_SINK_SRV_STATE_HELD_REMAINING	End Call	Hold Call
BT_SINK_SRV_STATE_MULTIPARTY	End Call	N/A

- EVENT_GROUP_UI_SHELL_BT_SINK
  - HFP state changes by BT sink state. App_hfp_activity sets different LED BG patterns and plays different VPs according to HFP state changes. For example, there are two methods to notify the user that there is an incoming call:
    - If the smartphone sends in-band ringtone to the device, the product plays in-band ringtone.
    - If smartphone doesn't send in-band ringtone, the device plays VP. This VP is special for it will loop until the state of the call is changed.
- EVENT_GROUP_UI_SHELL_APP_INTERACTION
  - When APPS_EVENTS_INTERACTION_UPDATE_LED_BG_PATTERN event is received, the APP will set background LED according to its state.
  - When APPS_EVENTS_INTERACTION_UPDATE_MMI_STATE event is received, the APP will set MMI state.
  - When APPS_EVENTS_INTERACTION_RHO_END or APPS_EVENTS_INTERACTION_PARTNER_SWITCH_TO_AGENT event is received, the APP will decide whether to set VP playback or not according to its state.

The app support use key or rotary to change the sidetone level. The function apps_config_audio_helper_set_sidetone_value() is used to change the sidetone value. The sidetone also can be enabled, disabled or change value by race CMD
The CMD for changing the sidetone enable or disable is "05 5A 05 00 82 2C 07 00 <00 or 01>" – The last byte 00 is disable, 01 is enable.
The CMD for changing the sidetone value is "05 5A 06 00 82 2C 06 00 <XX YY>" – The last 2 bytes means the target value is 0xYYXX.

2.2.5. FindMe APP

FindMe APP is developed for users to find the device if needed. While the device received a FindMe command, LED pattern and VP playback may be set by the APP for users to see the device soon.

This APP is composed of two activities: *app_fm_idle_activity* and *app_fm_activity*, and processes events of EVENT_GROUP_UI_SHELL_FINDME group and EVENT_GROUP_UI_SHELL_APP_INTERACTION group. The *app_fm_idle_activity* is created when the device is powered on. It starts *app_fm_activity* while receives a RACE_EVENT_TYPE_FIND_ME event which is one of FINDME group events.

App_fm_activity processes events of groups EVENT_GROUP_UI_SHELL_KEY, EVENT_GROUP_UI_SHELL_FINDME and EVENT_GROUP_UI_SHELL_APP_INTERACTION.

- EVENT_GROUP_UI_SHELL_KEY
    - The event will be sent when user presses a key on device, and received by the APP to stop the current FINDME request.
- EVENT_GROUP_UI_SHELL_FINDME
    - To trigger FINDME, the smartphone will send a RACE CMD to the device. After received the RACE CMD, the device will send an APP_FIND_ME_EVENT_ID_TRIGGER event which includes three parts' data*: is_blink*, *is_tone* and *duration_seconds.* When APP_FIND_ME_EVENT_ID_TRIGGER is received, the *app_fm_activity* will decide whether to set LED patterns and play VPs according to the parameters received.
        - If is_tone is 1, the activity will start playing FINDME VP and send a APPS_EVENTS_INTERACTION_PLAY_FIND_ME_TONE event with a delay time to itself through API ui_shell_send_event(). The activity will play the VP and send the same event with the delay time again while received the event.
        - The delay time is 4 seconds which is defined as FINDME_EVENT_TIMER_INTERVAL, and if the the *duration_seconds* is 0, total time for one FINDME request is 30 seconds which is defined as FINDME_TOTAL_TIME. The app_fm_activity will remember the current count of VP playback, and finish itself after VP played for a total 5 times.
        - A FINDME request should be stopped when the next FINDME request comes, even if the previous one has not been finished.
- EVENT_GROUP_UI_SHELL_APP_INTERACTION
    - When APPS_EVENTS_INTERACTION_UPDATE_LED_BG_PATTERN event is received, the APP will set background LED.
    - When APPS_EVENTS_INTERACTION_UPDATE_MMI_STATE event is received, the APP will set MMI state.
    - When APPS_EVENTS_INTERACTION_PLAY_FIND_ME_TONE event is received, the APP will set FindMe VP playback.

2.2.6. FOTA APP

This APP is completed by app_fota_idle_activity, who receives and processes FOTA_START event, FOTA_CANCEL event and FOTA_NEED_REBOOT event.

- When received FOTA_START, the app will play VP and set LED pattern.
- When received FOTA_CANCEL, the app will only set LED pattern.
- When received FOTA_NEED_REBOOT when FOTA finishes download, the APP will broadcast this event.

2.2.7. Music APP

Music APP controls the playback of music. This APP is composed of two activities: *app_music_idle_activity* and *app_music_activity*. The idle activity is created when the device is powered on, and the music activity will be started while streaming.

App_music_idle_activity processes events of groups EVENT_GROUP_UI_SHELL_KEY and EVENT_GROUP_UI_SHELL_BT_SINK.

- EVENT_GROUP_UI_SHELL_KEY
    - The key event handled by app_music_idle_activity is shown in Table 4. States and Keys Handled by app_music.
- EVENT_GROUP_UI_SHELL_BT_SINK
    - When BT_SINK_SRV_EVENT_STATE_CHANGE event is received, the APP will check whether the state is changed to streaming and start app_music_activity.
    - When BT_SINK_SRV_EVENT_AWS_MCE_STATE_CHANGE or BT_SINK_SRV_EVENT_PROFILE_CONNECTION_UPDATE event is received, the APP will update and check the states to do audio channel set.
        - When Agent and Partner are connected and Partner is not charging, both audio channels should be stereo.
        - Otherwise, Agent should be mono.

App_music_activity processes events of groups EVENT_GROUP_UI_SHELL_KEY, EVENT_GROUP_UI_SHELL_BT_SINK and EVENT_GROUP_UI_SHELL_APP_INTERACTION.

- EVENT_GROUP_UI_SHELL_KEY
    - The key event handled by app_music_activity is shown in Table 4. States and Keys Handled by app_music.
- EVENT_GROUP_UI_SHELL_BT_SINK
    - When BT_SINK_SRV_EVENT_STATE_CHANGE event is received, the APP will check whether the streaming is ended and finish itself.
- EVENT_GROUP_UI_SHELL_APP_INTERACTION
    - When APPS_EVENTS_INTERACTION_UPDATE_MMI_STATE event is received, the APP will set MMI state.

Table 4. States and Keys Handled by app_music

Activity	State	Key handle
Music idle activity	Power on; Connected	Play
Music activity	Streaming	Pause; Next / Forward;

Activity	State	Key handle
		Volume Up/Down;

2.2.8. Ear detection APP

This APP is used to detect the wearing status of earbuds and report it to other apps. This APP is completed by app_in_ear_idle_activity.

This APP does not control the playback of music, it is controlled by the music APP. The music APP receives event APPS_EVENTS_INTERACTION_IN_EAR_UPDATE_STA come from Ear detection APP and play or pause music according to current status. When status updated, if the Agent was taken off and the Partner is still wearing or only the Partner is put on, app_in_ear_idle_activity will trigger RHO.

App_in_ear_idle_activity process events of groups EVENT_GROUP_UI_SHELL_BT_SINK and EVENT_GROUP_UI_SHELL_APP_INTERACTION in addition to EVENT_GROUP_UI_SHELL_SYSTEM.

- EVENT_GROUP_UI_SHELL_BT_SINK
    - When BT_AWS_MCE_AGENT_STATE_INACTIVE event is received, app_in_ear_idle_activity will update status.
    - When BT_SINK_SRV_EVENT_PROFILE_CONNECTION_UPDATE event is received, the Partner will update status to Agent.
- EVENT_GROUP_UI_SHELL_APP_INTERACTION
    - The APPS_EVENTS_INTERACTION_UPDATE_IN_EAR_STA_EFFECT will be sent when the sensor detects that the earbuds is put on or taken off. When this event is received, for Agent, the APP will update status, for Partner, the APP will update status to Agent.

2.2.9. Smart Charger APP

This APP could receive event from Smart Charger middleware and control state machine.

The state machine is as below:

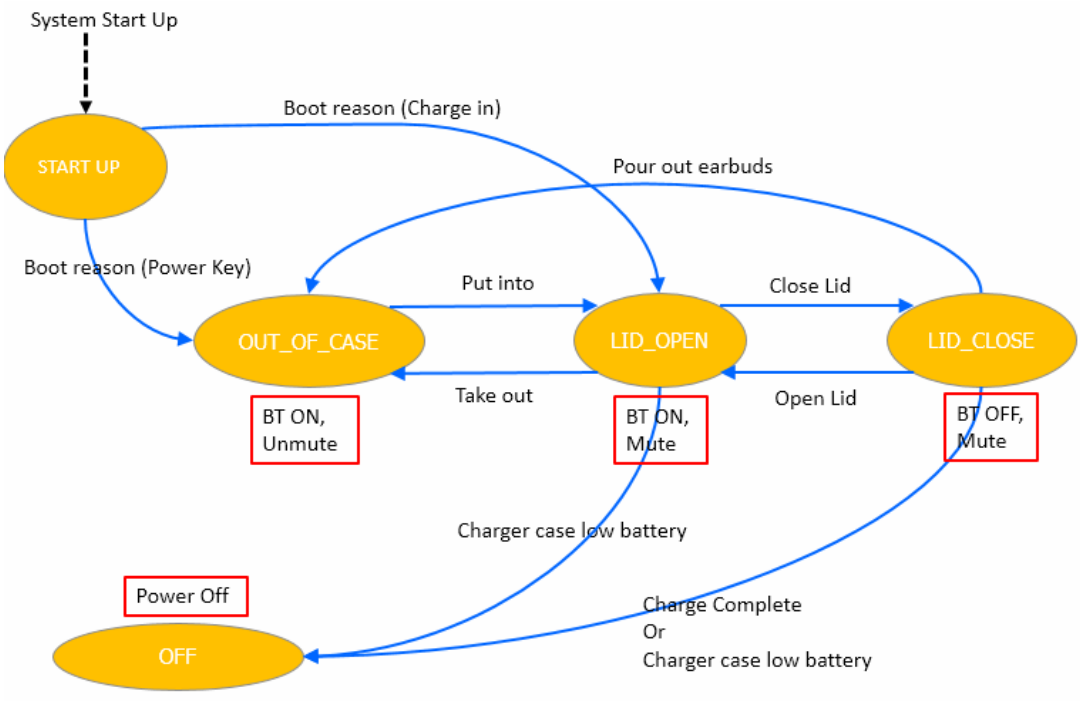


Figure 3. State machine in Smart Charger App

Smart Charger APP RHO scenario:

- Agent is in the charger case , but partner is out of case
- The battery percent of partner is 30% greater than agent

Other APP could get Smart Charger Case action and battery information by monitoring EVENT_ID_SMCHARGER_NOTIFY_PUBLIC_EVENT event of EVENT_GROUP_UI_SHELL_CHARGER_CASE group.

The design of Smart Charger Case should match with Smart Charger middleware, such as command/response payload, command sending delay etc. For more develop detail of Smart Charger Case, please refer to <One-Wire UART Smart Charger Case design.ppt>.

2.2.10. MCSync share APP

This APP provides a UI for entering or exiting MCSync share. It also handles RHO events for MCSync share. This APP is composed of two activities: *app_share_idle_activity* and *app_share_actvity*.

App_share_idle_activity is used to receive events to enter MCSync share. These events come from the Key and Race command. If the user sends the request of enter MCSync share through the Race command, the request is converted into the key event to enter MCSync share. The app_share_idle_activity handles the follow events:

- EVENT_GROUP_UI_SHELL_KEY
    - If the key action id is KEY_SHARE_MODE_SWITCH or KEY_SHARE_MODE_FOLLOWER_SWITCH, app_share_idle_activity will try to enter MCSync share and create app_share_activity to handle the events in sharing status. But if the current role is Partner, this event is sent to Agent.
- EVENT_GROUP_UI_SHELL_APP_INTERACTION

- When the APPS_EVENTS_INTERACTION_SHARE_MODE_STA_UPDATE event is received, the app checks the status of MCSync share. If the current status is already in sharing or preparing status, the app_share_idle_activity creates app_share_activity.

- EVENT_GROUP_UI_SHELL_BT_CONN_MANAGER
  - Before entering MCSync share, if there are two smart phones connected with earbuds, the APP disconnects the inactive smart phone and be pending the action of enter MCSync share until the earbuds are completely disconnected from the inactive phone. This activity will then handle the BT_CM_EVENT_REMOTE_INFO_UPDATE event to check the connection status and try to enter MCSync share again.

App_share_activity is used to receive events to exit MCSync share. It also manages the status of MCSync share. If the current status is preparing or sharing, the RHO is not allowed.

In addition to exiting from MCSync share by user, if the user puts any earbuds into the charger case or powers off any earbuds, all earbuds will exit MCSync share. In these cases, the RHO is allowed but will pending until the earbuds completely exit MCSync share.

The app_share_activity processes the follow events:

- EVENT_GROUP_UI_SHELL_KEY
  - If the key action id is KEY_SHARE_MODE_SWITCH or KEY_SHARE_MODE_FOLLOWER_SWITCH, app_share_activity will try to exit MCSync share. This activity will finish by itself when earbuds completely exit MCSync share.

- EVENT_GROUP_UI_SHELL_APP_INTERACTION
  - When the APPS_EVENTS_INTERACTION_SHARE_MODE_STA_UPDATE event is received, this activity checks the status MCSync status. If the current status is not preparing or sharing, app_share_activity will finish by itself.
  - When APPS_EVENTS_INTERACTION_POWER_OFF event received, this activity will try to exit MCSync share and be pending on the possible RHO.

- EVENT_GROUP_UI_SHELL_BATTERY
  - When APPS_EVENTS_BATTERY_CHARGER_EXIST_CHANGE event received, this activity will check the charging status. If currently charging, this activity will also try to exit MCSync share and be pending on the possible RHO.

2.2.11. Power save APP

Power save app support to do system power off or BT power off automatically to save power.

The function app_power_save_utils_get_target_mode() can return current power saving target mode. The target mode have 3 levels: NORMAL means doesn't need power off; BT_OFF means need power off BT but keep system is on; SYSTEM_OFF means need do system power off. Normally the target state is idle when BT is not connected or silence detected. But some other module can use app_power_save_utils_add_new_callback_func() to register callback and call app_power_save_utils_notify_mode_changed() to notify state changed. When power save app get target mode, the callback will be called and can return its mode. For example, when BT is not connected and USB audio is connected, the USB audio module don't want to do system off, so it can return BT_oFF.

Power save app have 3 states. Idle state means does not need power save. Waiting state means need do power save but must wait until timeout. Timeout state means need do power save and already timeout, so need power off immediately.

In AWS project, the partner keeps in idle state when AWS is connected. The agent will send power off request to partner to make sure both side can power off at the same time.

The waiting time are different for BT is not connected and silence detected, default is 5 minutes when BT is not connected and 20 min when silence detected.

The power save feature is enabled when APPS_SLEEP_AFTER_NO_CONNECTION is y. The power off when silence detected feature is enabled when AIR_SILENCE_DETECTION_ENABLE is y. And they also can be disabled dynamically by smart phone APP, you can review the function app_power_save_utils_set_cfg() and app_power_save_utils_set_cfg() to understand it.

2.2.12. Interaction events

The interaction events are sent by APPs. Here are three typical scenarios of interaction events:

- An APP sends an event to itself.
- An APP sends UPDATE_LED_BG_PATTERN event. The event is passed in activity stack from top to bottom. One of the activities set background LED pattern and return TRUE after processing, so that all the next activities will not receive the event.
- An APP can send a request event and another APP processes the request when it receives the event. The relationship is as below:

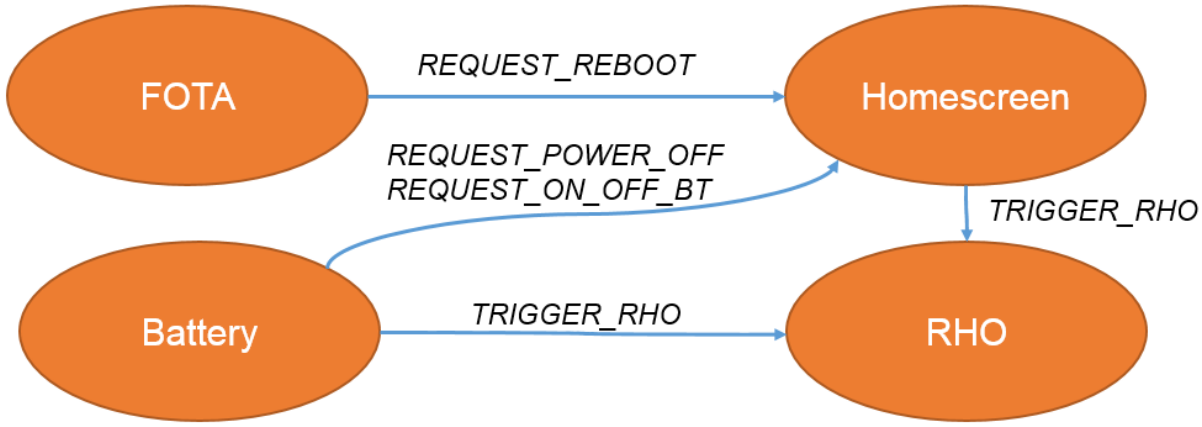


Figure 4. Interaction events sent between apps

2.3. Config

Config modules implement the management of what actions will be taken by the device when the user pressed a key, and what LED pattern and VP playback should be seen in some situations. The configurations of these modules are decided by project customization, which will be introduced in Chapter4.

2.3.1. Key_remapper

The actions to be taken when key pressed are decided by current states and key configs, the mapping method is provided in key_remapper. The mapping tables are defined in

/mcu/project/<board>/apps/<project>/src/<board_type>/customerized_key_config.c. Key_remapper module uses key_id, key_event and current MMI state as input to query action in the mapping tables.

There are three mapping tables in current design. Two tables are default (i.e. non-configurable) mapping tables; One is used when the device is configured as left side, and the other is used when the device is configured as right side or not yet configured to left nor right.

And, the last one of the three mapping tables is a configurable mapping table which has higher priority (please refer to Figure 5. The Flow for Getting the Key Action for reference). The configurable table can be configured by smartphone APP.

APPs call functions provided by this module to map key event. The input parameters are *key_id* and *airo_key_event* . The output is an action, which is an ENUM type named *apps_config_event_t*. Please refer to /mcu/project/<board>/apps/<project>/src/apps/config/apps_config_key_remapper.c.

Current state is defined in key_remapper module, and can be set by APPs through function *apps_config_key_set_mmi_state().* Normally MMI state is set by activities who receives an APPS_EVENTS_INTERACTION_UPDATE_MMI_STATE event. The activity should return TRUE after handling the event to avoid the next activity receiving it.

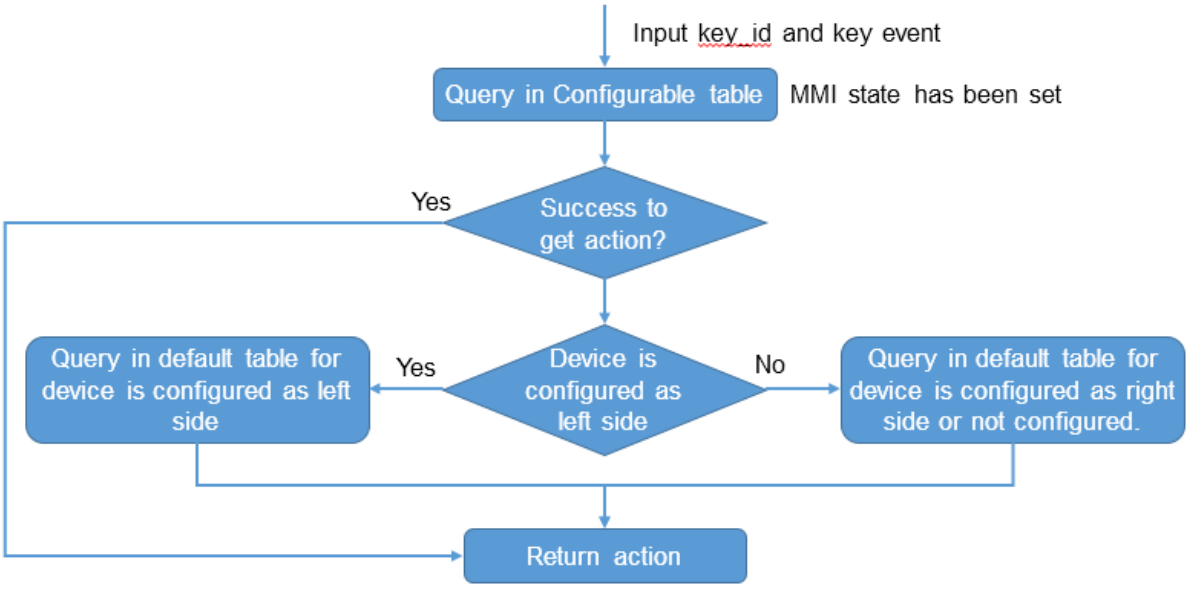


Figure 5. The Flow for Getting the Key Action

APIs:

- *apps_config_key_action_t apps_config_key_event_remapper_map_action(uint8_t key_id, airo_key_event_t key_event)*: get the key action in key_event map.
- *void apps_config_key_set_mmi_state(apps_config_state_t state)*: set the current state of the MMI.
- *apps_config_state_t apps_config_key_get_mmi_state(void)*: get the current state of the MMI.
- *void apps_config_key_remaper_init_configurable_table(void)*: initialize the configurable key

2.3.2. Led_manager

APPs call functions provided by this module to control LED patterns. Foreground and background LED patterns are supported, and the foreground LED patterns have higher priorities and need a timeout period as parameter. This module calls APIs provided by LED module to set LED patterns.

Normally background LED patterns are set by activities who receives an APPS_EVENTS_INTERACTION_UPDATE_LED_BG_PATTERN event. The activity should return TRUE after handling the event to avoid the next activity receiving it.

Both foreground pattern and background pattern can be synchronized from the agent to the partner. But there is a different point:

- Foreground patterns indicate event happens: Partner led pattern will show least led pattern no matter local or synchronized from agent.
- Background patterns indicate the states which may keep for a lone time: For background pattern, a parameter “priority” will be referred. If the “priority” of local background LED pattern on the partner side is higher than the pattern which is synchronized from agent, partner need show local background LED pattern and vice versa.

APIs:

- *void apps_config_led_manager_init(void)*: init led manager module when the system is started.
- *bool apps_config_set_backgroud_led_pattern(uint8_t index, bool need_sync, apps_config_led_manager_aws_sync_priority_t priority)*: set background LED pattern.
- bool apps_config_set_foreground_led_pattern(uint8_t index, uint16_t timeout, *bool need_sync*): set foreground LED pattern.

The foreground and background LED pattern list is as

Table 6 and

Table 7 in appendix. Normally background pattern indicate the current system states. To save power, *charging full*, *playing music* and *connected* state use LED_INDEX_IDLE LED pattern.

2.3.3. VP_manager

APPs call functions provided by this module to control the voice prompt playback.

There are 3 types of voice prompt and the APIs to start and stop them:

- normal voice prompt：
  Start: apps_config_set_vp()
  Stop: apps_config_stop_vp()

- voice number : phone number of incoming
  Start: app_config_set_voice()
  Stop: apps_config_stop_voice()

- ringtone: a voice prompt that can be played back.
  Start: app_config_set_voice()
  Stop: apps_config_stop_voice()

Language change is supported by VP_manager.

Table 5 in appendix shows the voice prompt priorities and indexes on different conditions. If a VP with higher priority are set when a VP is playing, it will be played immediately with the current one dropped, or it will be put in the right position of the VP playback queue according to priorities.

To get more details, please refer to description of the functions.

3. Feature options for MMI

Feature options can be controlled in feature.mk for some common scenarios of users. For example:

- APPS_DISABLE_BT_WHEN_CHARGING: Disable BT while the device is being charged.
- APPS_AUTO_TRIGGER_RHO: Trigger RHO before power off, disable BT and when the battery level of agent is lower than Partner.
- APPS_SLEEP_AFTER_NO_CONNECTION: Start a timer to sleep when the device isn't connected to the smartphone.
- AIR_SILENCE_DETECTION_ENABLE: Support auto power off when DSP detected audio is not playing or is a silence streaming. The option must be enable at CM4 side and DSP side simultaneously.

More features and details please refer to /mcu/project/<project_path>/src/apps/module.mk.

4. Customization

Customers can change configurations through ConfigTool or software code.

ConfigTool can be used to change the parameters and data for fine-tuning and also some feature options for factory producing. They are:

- LED patterns
- VP file table
- BT name
- MP Test Mode and DUT Mode

A software configuration is a macro or a variable which is defined in a *.c or *.h file. Here are types:

- A configuration can be a parameter or a feature option that affects limited code in *.c files.
- The parameters of APP features are defined in /mcu/project/<board>/apps/<project>/inc/apps*/apps_customer_config.h*, such as
    - Air pairing key and air pairing duration.
    - Support of enabling BT discoverable mode when disconnected from smartphone.
- Some configurations for BT middleware features are defined in /mcu/project/<board>/apps/<project>/src/apps*/bt_customer_config.c*. For example,
    - The rule of deciding who is agent in air pairing
    - default volume
    - HFP feature

4.1. Project settings

BOARD_TYPE

- Defined in mcu/project/<project_path>/GCC/feature_xxx.mk.
- It's used to decide which folders in */config_bin*, */ept_config*, */inc* and */src* are included.
    - For example, if *GCC/feature_ab1552_evk.mk* is used and defines BOARD_TYPE = ab1552_evk, the folders */config_bin/ab1552_evk*, */ept_config/ab1552_evk* and */inc/ab1552_evk*, */src/ab1552_evk* will be included in the project.

Serial port configuration

- Can be changed through the macros defined in */inc/<board_type>/serial_port_assignment.h*.

nvdm.bin and filesystem.bin are in folder /ept_config/<board_type>.

- The *nvdm.bin* is based on *nvkey.xml* which is in the same folder.
- *nvdm.bin* and *filesystem.bin* can be changed by ConfigTool. Please refer to the document AB155x_AM255x_Config_Tool_Users_Guide.

4.2. Voice prompt

Customers can add voice prompt source and change the configurations of the VP playback, as well as select the language you use. Noted, only MP3 file is supported to be added as source and only Chinese and English are supported in language usage.

4.2.1. Change voice prompt

Use ConfigTool to change voice prompt, follow the steps.

1) Open AB155x ConfigTool and click *Open File* to load the firmware.

Figure 6. Step 1 of the Config Tool

2) Select VP section and VP language.

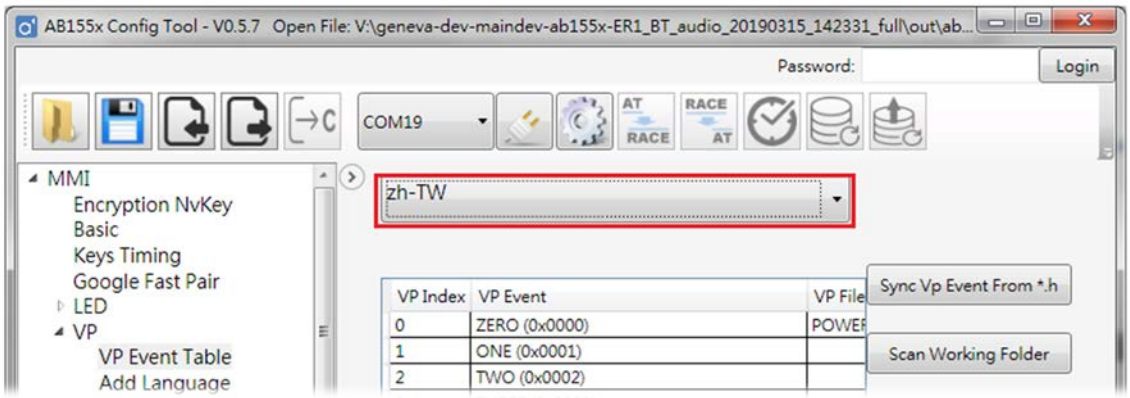


Figure 7. Step 2 of the Config Tool

3) Sync VP from *.h. The configuration file is in *config/mcu/project/<project_path>/inc/apps/config/apps_config_vp_index_list.h*. VP event will change as shown in Figure 8 and Figure 9.

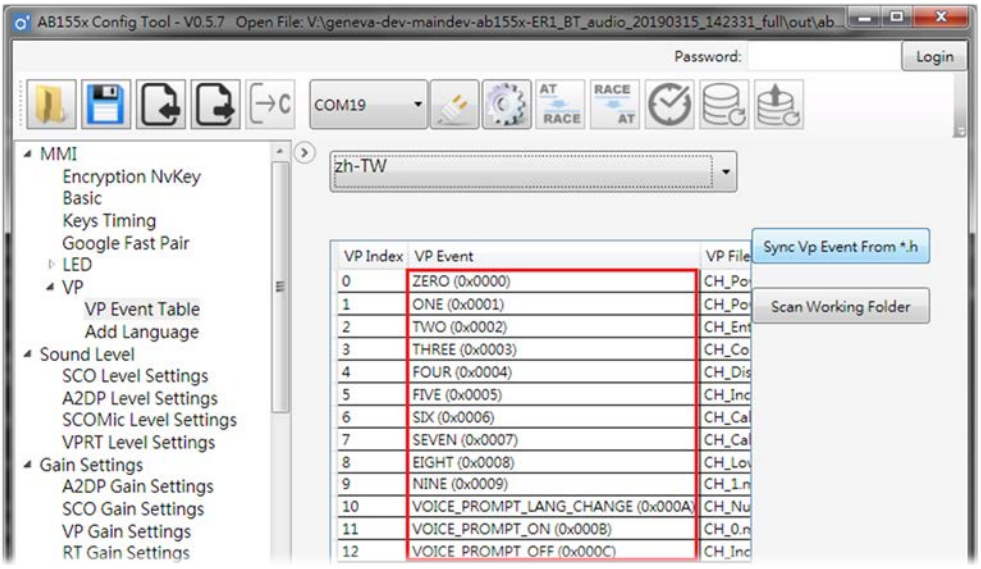


Figure 8. Step 3 of the Config Tool (before sync)

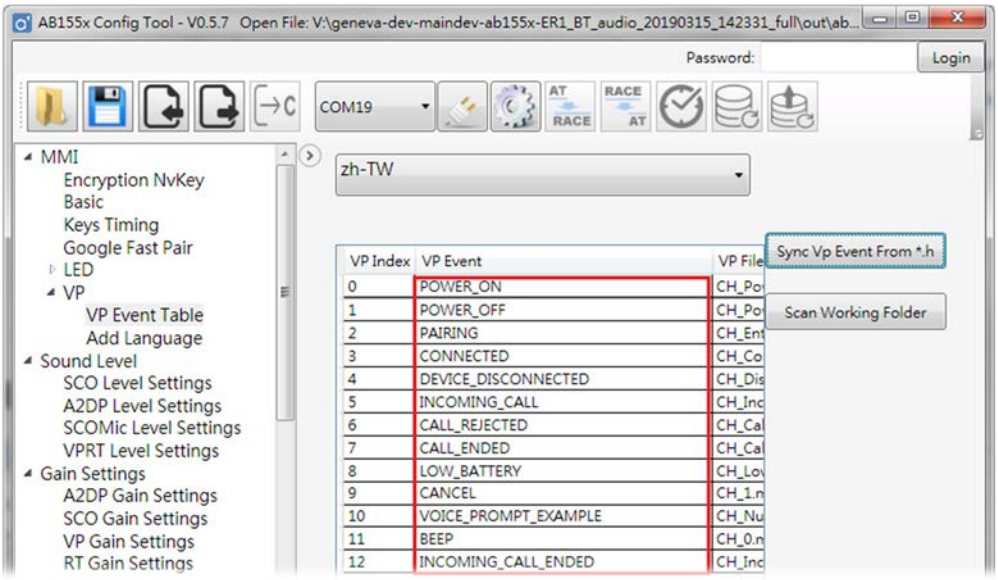


Figure 9. Step 3 of the Config Tool (after sync)

4) Sync VP from work folder. When open AB155x config, you can find working folder in ConfigTool folder as shown in Figure 10. The working folder contains VP mp3 files which are stored in AB155x SDK by default.

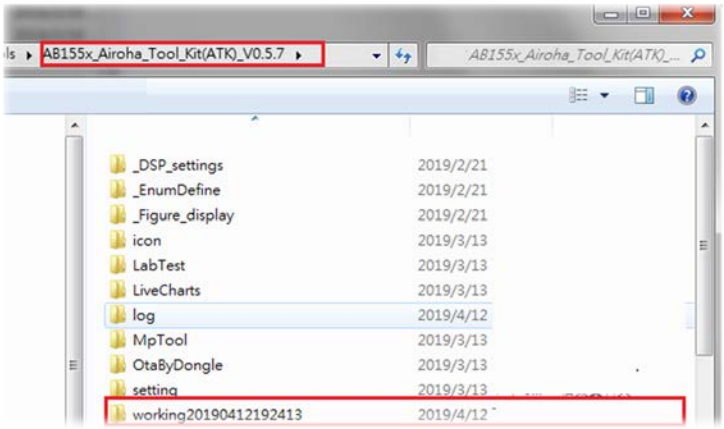


Figure 10. Step 4 of the Config Tool (default folder)

5) Copy customized VP files to working folder. Please be noticed that we only support mp3 files, and the file name should be rename according to ENUM type in /mcu/project/<board>/apps/<project>/inc/apps/config/apps_config_vp_index_list.h. For example, VP_INDEX_POWER_ON is defined as index name of VP of power on, so the file should be renamed as POWER_ON.mp3.

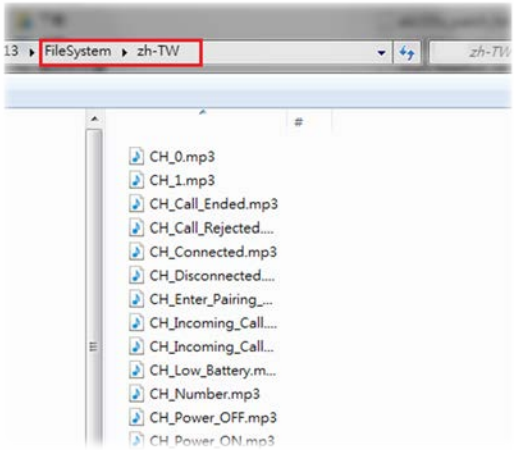


Figure 11. Step 5 of the Config Tool (user files)

6) Click sync working folder button, and double click VP index to add/remove VP files.

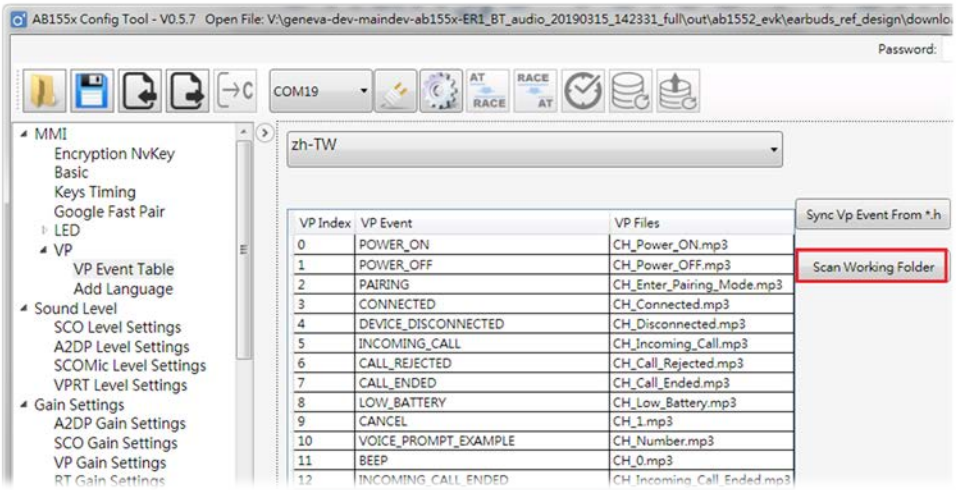


Figure 12. Step 6 of the Config Tool (scanning the working folder)

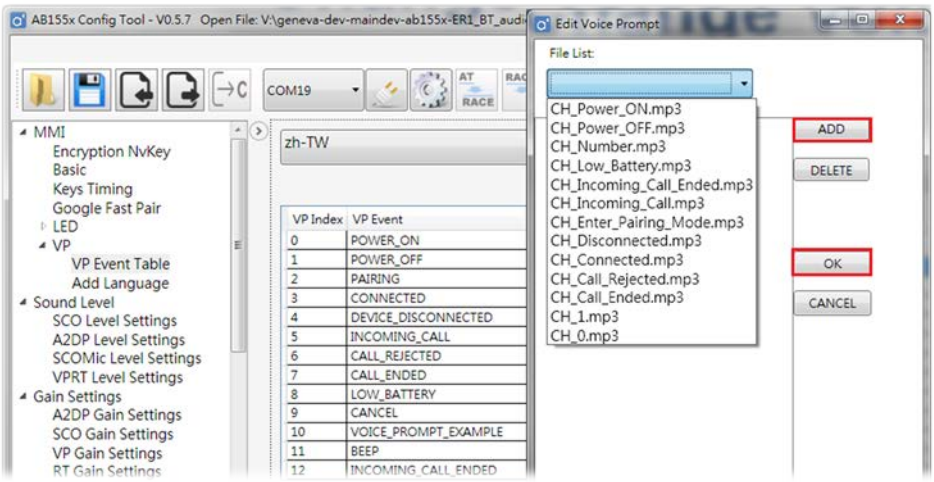


Figure 13. Step 6 of the Config Tool (adding VP)

7) When you finished add/remove VP files, you need click ConfigTool save button to update *nvdm.bin* and *filesystem.bin*. The *nvdm.bin* and *filesystem.bin* will be updated in the folder \mcu\project\ab155x_evk\apps\earbuds_ref_design\config_bin\<board_type>.

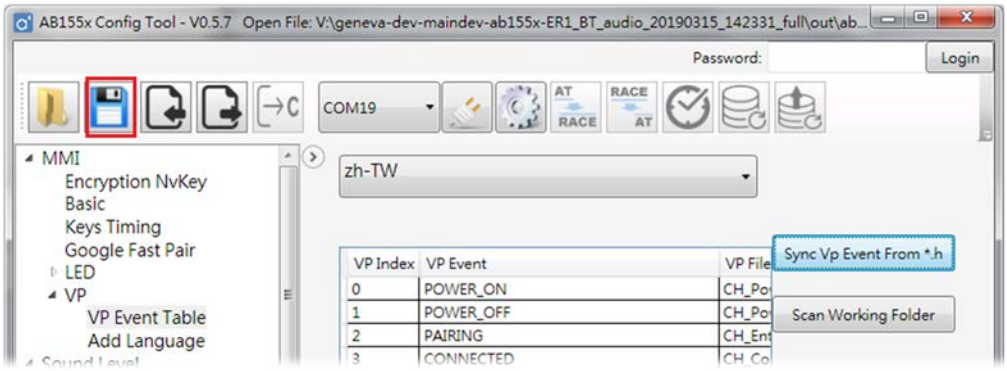


Figure 14. Step 7 of the Config Tool

Every time you build the load, the *nvdm.bin* and *filesystem.bin* will be copied to the out folder /out/<board_type>/earbuds_ref_design/download. If you don't want to build code after you have changed the *nvdm.bin* or *filesystem.bin*, you can copy the two files manually before download.

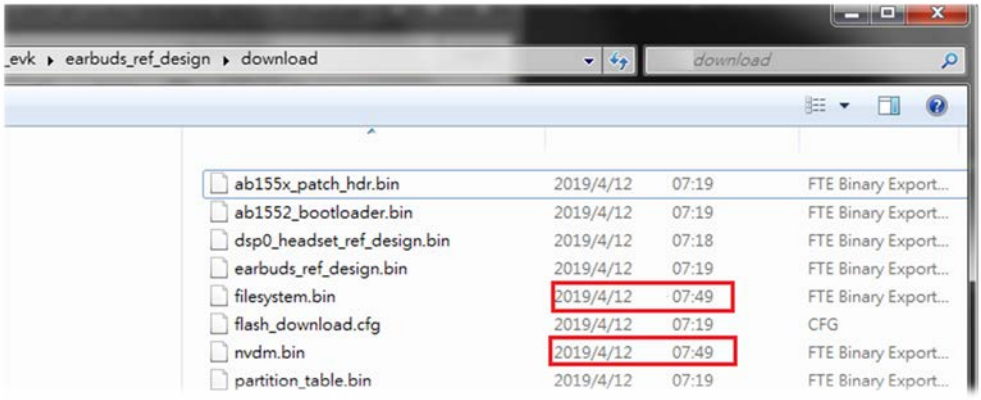


Figure 15. Binary files in build out folder

Please refer to the document AB155x_AM255x_Config_Tool_Users_Guide for more details.

4.2.2. Call voice prompt in Code

Here are functions to play a voice prompt.

- *uint16_t apps_config_set_vp(uint32_t vp_index, bool need_sync, uint32_t sync_delay_time, app_vp_prio_t level, bool cleanup)*: play a normal voice prompt. The recommended value of delay time is 200ms.
- *uint16_t apps_config_set_voice(uint32_t vp_index, bool need_sync, uint32_t sync_delay_time, app_vp_prio_t level, bool ringtone)*: play a ringtone or voice number. Normally the function is used to notify incoming call.

Please refer to the file /mcu/project/<project_path>/src/apps/config/apps_config_vp_manager.h to get more information.

4.3. LED pattern

LED pattern configurations are supported to change LEDs flash behavior, including ON/OFF, frequency, repeat time and others.

4.3.1. Change LED patterns

1) Use ConfigTool to open the folder which contains *nvdm.bin* and *filesystem.bin*.

2) Click the button and select the file mcu/project/ab155x_evk/apps/<project>/inc/apps/config/apps_config_led_index_list.h. The number of max LED style number is depends on the file.

Figure 16. Step 2 of the LED pattern config

3) Increase or decrease the ENUM type defined in /mcu/project/<board>/apps/<project>/src/apps/config/*apps_config_led_index_list.h* to add or remove LED pattern.

4) Click to fine tune the LED pattern of one line.

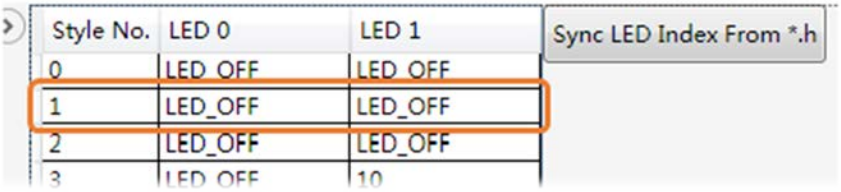


Figure 17. Step 4 of the LED pattern config (select item)

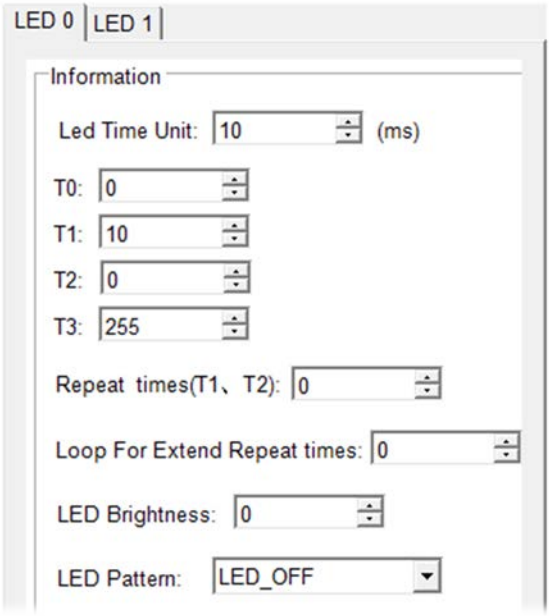


Figure 18. Step 4 of the LED pattern config (adjust parameters)

Figure 19 shows the parameters those should be set when LED patterns are changed.

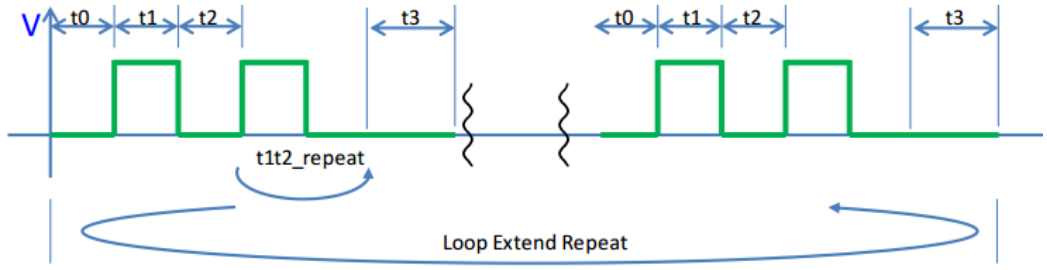


Figure 19. Parameters of LED patterns

Please refer to the document AB155x_AM255x_Config_Tool_Users_Guide for more details.

4.3.2. Call LED in code

Here are functions to set foreground and background LED patterns.

- *bool apps_config_set_foreground_led_pattern(uint8_t index, uint16_t timeout):* *Index* is the LED style number in ConfigTool. *Timeout* is the active timeout of the foreground pattern.
- bool apps_config_set_backgroud_led_pattern(uint8_t index): Index is the LED style number in ConfigTool.

Please refer to the file /mcu/project/<project_path>/src/apps/config/apps_config_led_manager.h to get more information.

4.4. Modify AB15xx MMI key events

To modify MMI Key Events, you have to do things list as below.

(1) Use EPT Tool to generate the Pinmux configurations you need to add a key action.

(2) Use Airoha Tool Kit to help you config the key timings of your key actions such as how long is a long press.

(3) Finally you need to change/add mapping table of key action as the description of this chapter.

4.4.1. EPT tool

Airoha Easy PinMux Tool (EPT) is a convenient and user-friendly graphical user interface (GUI) to configure pin multiplexor (PinMux) and supported driver settings for Airoha chipsets. The tool provides modes and options for each PinMux and enables customized settings for input and output (I/O) characteristics according to design requirements. When configured, all settings can be saved as a workspace file that can be reloaded to apply the preconfigured tool settings. The results can also be output as C header and source files.

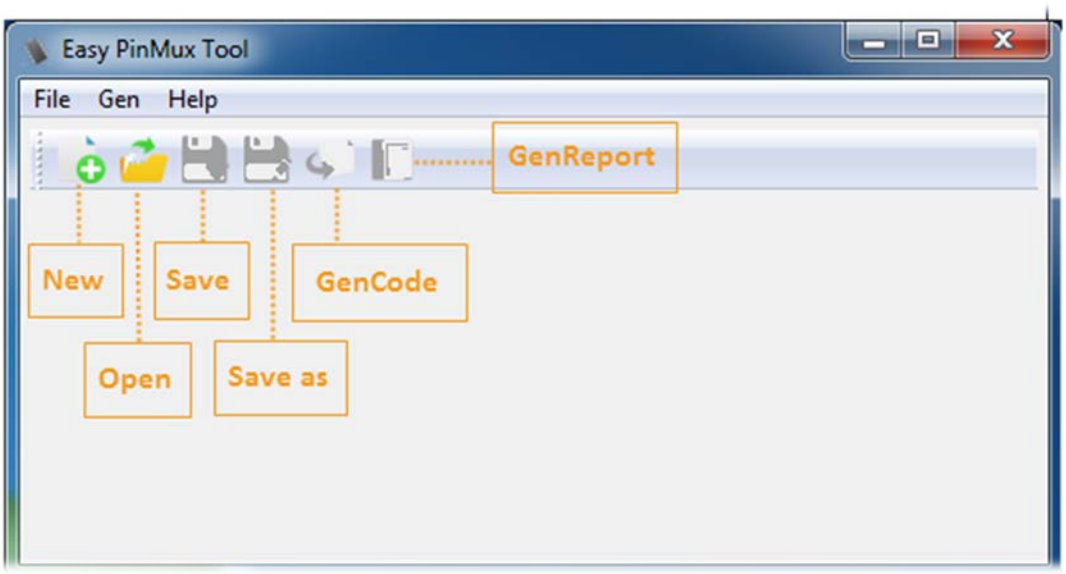
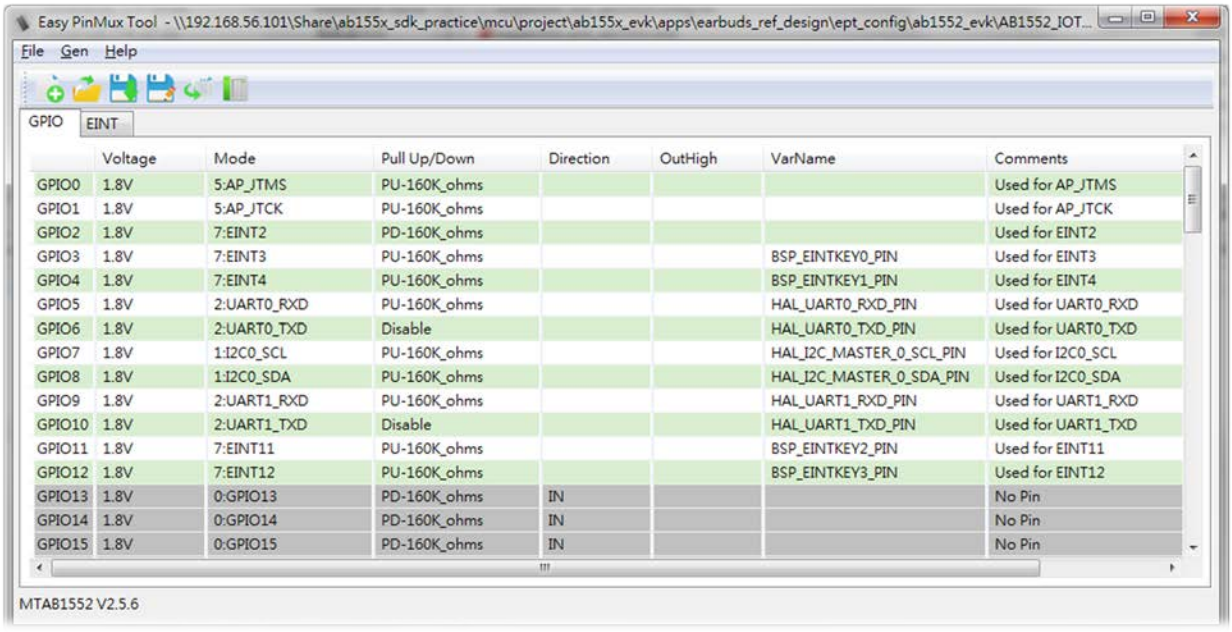


Figure 20. EPT tool (tool button)



	Voltage	Mode	Pull Up/Down	Direction	OutHigh	VarName	Comments
GPIO0	1.8V	5:AP_JTMS	PU-160K_ohms				Used for AP_JTMS
GPIO1	1.8V	5:AP_JTCK	PU-160K_ohms				Used for AP_JTCK
GPIO2	1.8V	7:EINT2	PD-160K_ohms				Used for EINT2
GPIO3	1.8V	7:EINT3	PU-160K_ohms			BSP_EINTKEY0_PIN	Used for EINT3
GPIO4	1.8V	7:EINT4	PU-160K_ohms			BSP_EINTKEY1_PIN	Used for EINT4
GPIO5	1.8V	2:UART0_RXD	PU-160K_ohms			HAL_UART0_RXD_PIN	Used for UART0_RXD
GPIO6	1.8V	2:UART0_TXD	Disable			HAL_UART0_TXD_PIN	Used for UART0_TXD
GPIO7	1.8V	1:I2C0_SCL	PU-160K_ohms			HAL_I2C_MASTER_0_SCL_PIN	Used for I2C0_SCL
GPIO8	1.8V	1:I2C0_SDA	PU-160K_ohms			HAL_I2C_MASTER_0_SDA_PIN	Used for I2C0_SDA
GPIO9	1.8V	2:UART1_RXD	PU-160K_ohms			HAL_UART1_RXD_PIN	Used for UART1_RXD
GPIO10	1.8V	2:UART1_TXD	Disable			HAL_UART1_TXD_PIN	Used for UART1_TXD
GPIO11	1.8V	7:EINT11	PU-160K_ohms			BSP_EINTKEY2_PIN	Used for EINT11
GPIO12	1.8V	7:EINT12	PU-160K_ohms			BSP_EINTKEY3_PIN	Used for EINT12
GPIO13	1.8V	0:GPIO13	PD-160K_ohms	IN			No Pin
GPIO14	1.8V	0:GPIO14	PD-160K_ohms	IN			No Pin
GPIO15	1.8V	0:GPIO15	PD-160K_ohms	IN			No Pin

Figure 21. EPT tool (config)

Generate file and copy to your project. The copy destination is ept\output\<chip>\inc => <sdk_root>mcu\project\<board>\apps\<project>\inc and ept\output\<chip>\src => <sdk_root>mcu\project\<board>\apps\<project>\src. The work steps is as Figure 22.

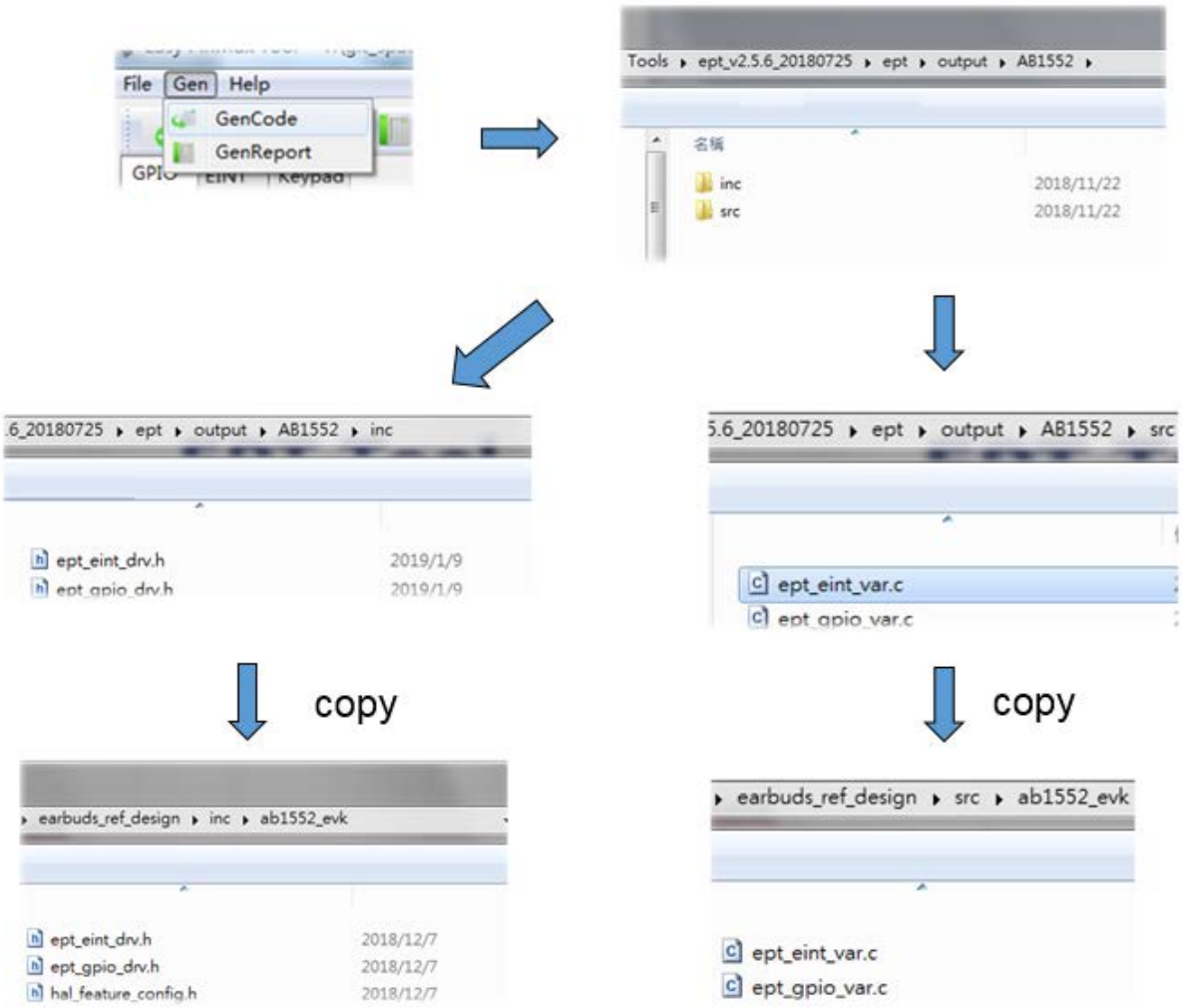


Figure 22. Work flow for the EPT tool

Please refer to the document Easy_PinMux_Tool_Users_Guide for more details.

4.4.2. Airoha Tool Kit

Use AB155x Airoha Tool Kit to configure key timing as Figure 23 shows.

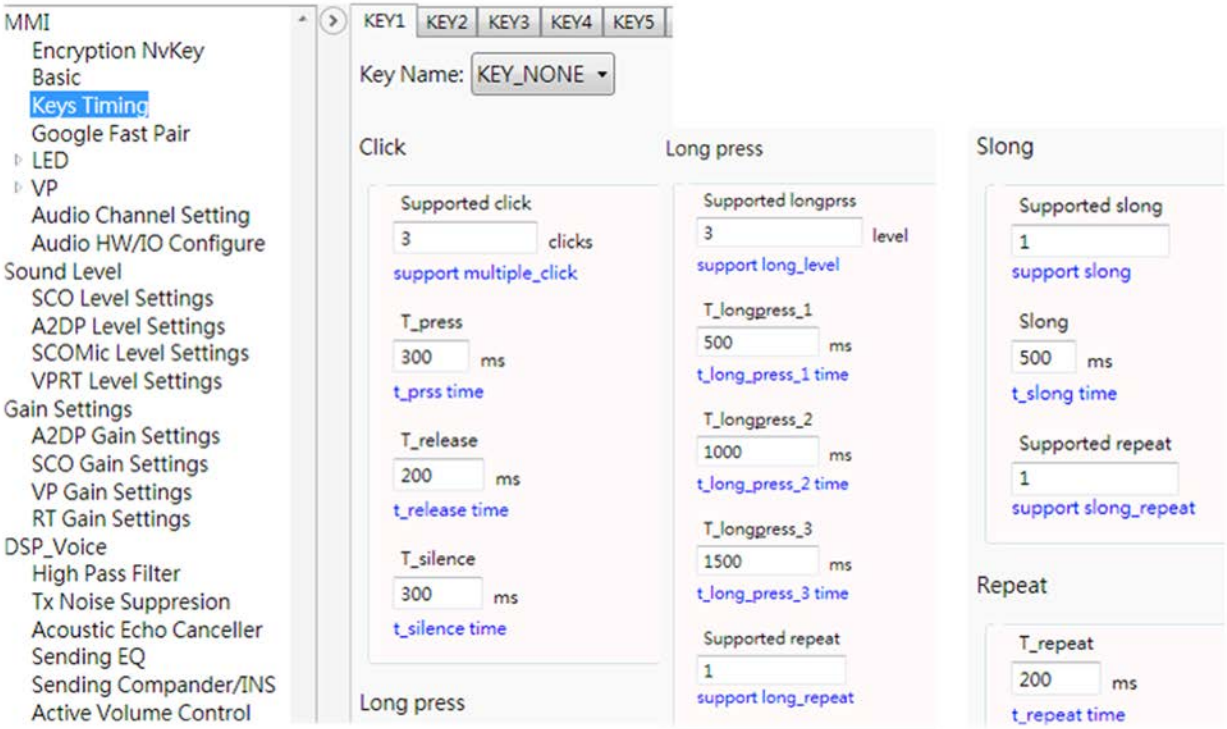


Figure 23. Airoha Tool Kit timing config

Please refer to the document of Airoha Tool Kit for more details.

4.4.3. AB15xx key event

Customers are able to change/add mapping table of Key Action. The mapping table is declared in /mcu/project/<project_path>/src/<board_type>/customerized_key_config.c as showed in Figure 24. The logic of key mapping can be simply described as the action is decided by Key ID, Key Event and current status.

```
const apps_config_key_event_map_t temp_key_short_click_configs[] = {
        {
                DEVICE_KEY_POWER,
                KEY_ACPCALL,
                (1 << APP_HFP_INCOMMING) | (1 << APP_HFP_CAIMG)
        },
        {       DEVICE_KEY_POWER,
                KEY_AVRCP_PLAY,
                (1 << APP_CONNECTED)
        }
```

Figure 24. Mapping table for the key action

Add new action means add new logic. The new key actions can be added in file /mcu/project/<project_path>/inc/apps/config/apps_config_event_list.h. And the app code must be changed to support the new feature.

MMI state is defined in mcu/project/ab155x_evk/apps/earbuds_ref_design/inc/apps/config/apps_config_state_list.h. Customers must

study UI shell and the usage of APPS_EVENTS_INTERACTION_UPDATE_MMI_STATE event in APPs in order to add MMI states.

5. FAQ

5.1.1. How to change BLE advertising data?

The BLE device name is defined in function *bt_customer_config_get_ble_device_name()* in file mcu/project/<project>/src/bt_customer_config.c. And you need to read the *bt_app_common_advtising_start()* in file mcu/project/<board>/apps/<project>/src/bt_app_common.c to get the information of the other data in BLE advertising.

5.1.2. How do Agent and Partner communicate?

Agent and Partner communicate with each other by MCSync packets.

- Function *bt_sink_srv_send_action()* is an SDK API for sending MCSync packet to each other.
- Function *apps_aws_sync_event_send()* is a shortcut of the API *bt_sink_srv_send_action()* to send simple data.

For example, if you want to do role handover with key triggered, you should

1. Add KEY_ROLE_SWITCH_TO_AGENT in *apps_config_event_list.h.*
2. Parse BT_SINK_SRV_EVENT_AWS_MCE_PACKET_RECEIVED_IND in function *homescreen_app_bt_sink_event_proc()*.
3. Decode event group with EVENT_GROUP_UI_SHELL_KEY and *event_id* with KEY_ROLE_SWITCH_TO_AGENT.

5.1.3. What is a NVDM reserve list?

There are two NVDM reserve lists in “Airoha IoT SDK for BT Audio” in path mcu/project/<board>/apps/<project>/inc/. The list in nvdm_config.h is for FOTA upgrade, and the other in nvdm_config_factory_reset.h is for factory reset. After FOTA upgraded or doing factory reset, all NVDM items except those in the reserve lists will be cleared.

nvdm.bin can be modified according to document AB155x_AM255x_Config_Tool_Users_Guide, and is not upgraded when user does FOTA because some important information (e.g. BT address) is stored in *nvdm.bin*. After setting the BT address in factory, the *nvdm.bin* should not be upgraded.

6. Appendix

Table 5. Voice Prompt Priority

Condition	VP index	Priority
System power on and device is not charging	VP_INDEX_POWER_ON	HIGH
Take out from charger case	VP_INDEX_POWER_ON	HIGH
Battery is too low to power off	VP_INDEX_POWER_OFF	EXTREME
Press key to power off	VP_INDEX_POWER_OFF	EXTREME
Wait connection too long to sleep	VP_INDEX_POWER_OFF	EXTREME
Device is enter pairing mode or air pairing	VP_INDEX_PAIRING	MEDIUM
Device is connected by smartphone	VP_INDEX_CONNECTED	MEDIUM
Device is disconnected from smartphone	VP_INDEX_DEVICE_DISCONNECTED	MEDIUM
Incoming call	VP_INDEX_INCOMING_CALL	ULTRA
Reject call by pressing key on earbuds	VP_INDEX_CALL_REJECTED	MEDIUM
Active call is ended	VP_INDEX_CALL_ENDED	MEDIUM
Battery capacity is low	VP_INDEX_LOW_BATTERY	MEDIUM
Fail to response a key action	VP_INDEX_FAILED	MEDIUM
Press key long enough to trigger voice assistant	VP_INDEX_PRESS	MEDIUM
FOTA finish successfully	VP_INDEX_SUCCESSED	MEDIUM
Received find me request from smartphone	VP_INDEX_DOORBELL	MEDIUM

Table 6. Foreground LED Patterns

Event	LED index	Timeout
System power on and device is not charging	LED_INDEX_POWER_ON	3s
Take out from charger case	LED_INDEX_POWER_ON	3s
Battery is too low to power off	LED_INDEX_POWER_OFF	3s
Press key to power off	LED_INDEX_POWER_OFF	3s
Wait connection too long to sleep	LED_INDEX_POWER_OFF	3s
Charging full	LED_INDEX_CHARGING_FULL	5s
Start FOTA	LED_INDEX_FOTA_START	3s

Event	LED index	Timeout
FOTA is interrupted	LED_INDEX_FOTA_CANCELLED	3s
Start air pairing	LED_INDEX_AIR_PAIRING	30s
Air pairing is success	LED_INDEX_AIR_PAIRING_SUCCESS	3s
Air pairing is failed	LED_INDEX_AIR_PAIRING_FAIL	3s

Table 7. Background LED Patterns

State	LED index
Battery charging but not charging full	LED_INDEX_CHARGING
Battery is low	LED_INDEX_LOW_BATTERY
Find me	LED_INDEX_FIND_ME
Incoming call	LED_INDEX_INCOMMING_CALL
Outgoing call	LED_INDEX_OUTGOING_CALL
Call active	LED_INDEX_CALL_ACTIVE
Hold call	LED_INDEX_HOLD_CALL
Pairing mode	LED_INDEX_CONNECTABLE
Disconnected and not in pairing mode	LED_INDEX_DISCONNECTED