Airoha IoT SDK DSP Get Started Guide

Version: 2.15
Release date: 26 May 2023

Document Revision History

Revision	Date	Description
1.0	20 September 2018	Initial release
1.1	12 December 2018	Added support for Ubuntu build environment
1.2	8 March 2019	Updated 2.4. Building a project using the SDK section
1.3	26 April 2019	Updated 1.1 Platform architecture and 1.2 Folder structure
2.0	13 May 2019	Added support for AM255x
2.1	4 June 2019	Added setting up Cadence license for an Xtensa toolchain
2.2	21 August 2019	Added note for Ubuntu Linux environment setup
2.3	24 September 2019	Added note for environment setup
2.4	15 November 2019	Revised instructions for using install.sh to automatically set up the environment
2.5	06 December 2019	Revised content for AM255x
2.6	16 June 2020	Revised supported Linux versions
2.7	30 June 2020	Added support for AB1565
2.8	24 July 2020	Added support for AB1568
2.9	20 January 2022	Added support for AB1585/AB1588
2.10	22 April 2022	Updated the tool chain version of 1585/1588 DSP
2.11	30 May 2022	Updated the tool chain version of 1565/1568 DSP
2.12	28 December 2022	Refined language and formatting
2.13	17 March 2023	Updated the tool chain version of 1571/1577 DSP
2.14	28 March 2023	Added the AB1571/AB1577 architecture diagram
2.15	26 May 2023	Added support for AB1627

Table of contents

Lists of tables and figures

1. Overview

The Airoha IoT Software Development Kit (SDK) provides the software and tools for application development on the AB155x/AM255x/AB1565 /AB1568/AB1571/AB1577/AB1585/AB1588 EVKs. The SDK includes drivers for the hardware abstraction layer, connectivity such as Bluetooth/Bluetooth Low Energy, peripherals, and other third-party features. It also provides battery management, firmware-over-the-air (FOTA) updates, and FreeRTOS.

This guide shows how to use the SDK and its supported features.

Note: In this guide, all supported chips use the same installation procedure and build method as AB155x. The subsequent examples use AB155x as a reference.

1.1. Platform architecture

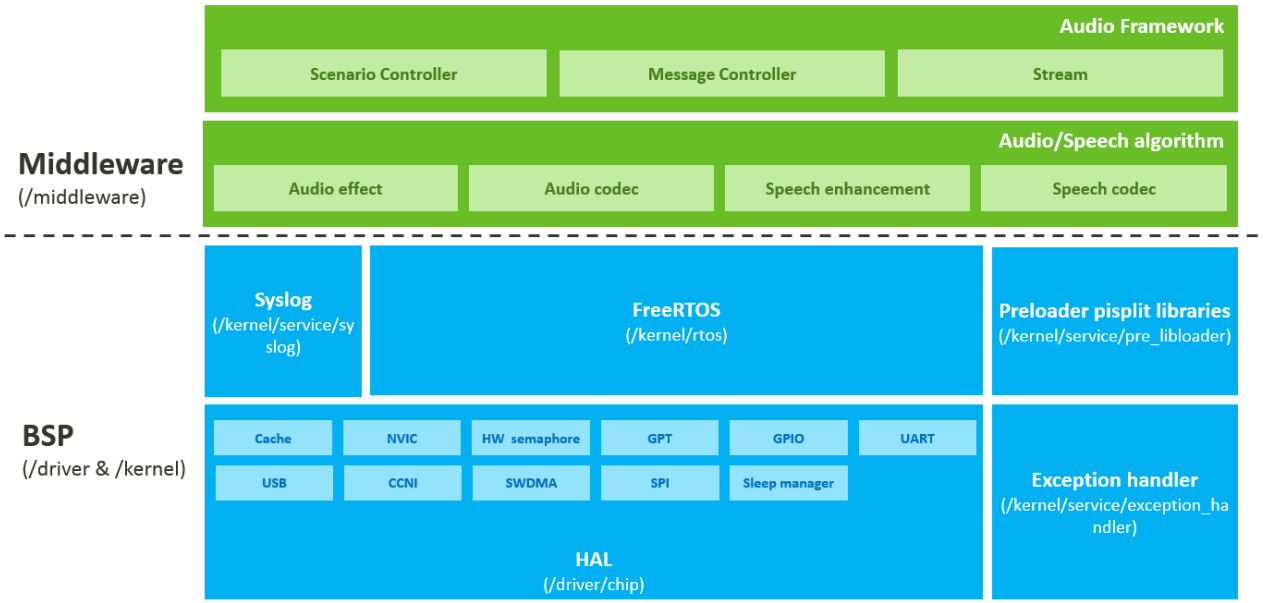

Figure 1. AB155x/AM255x platform architecture layout

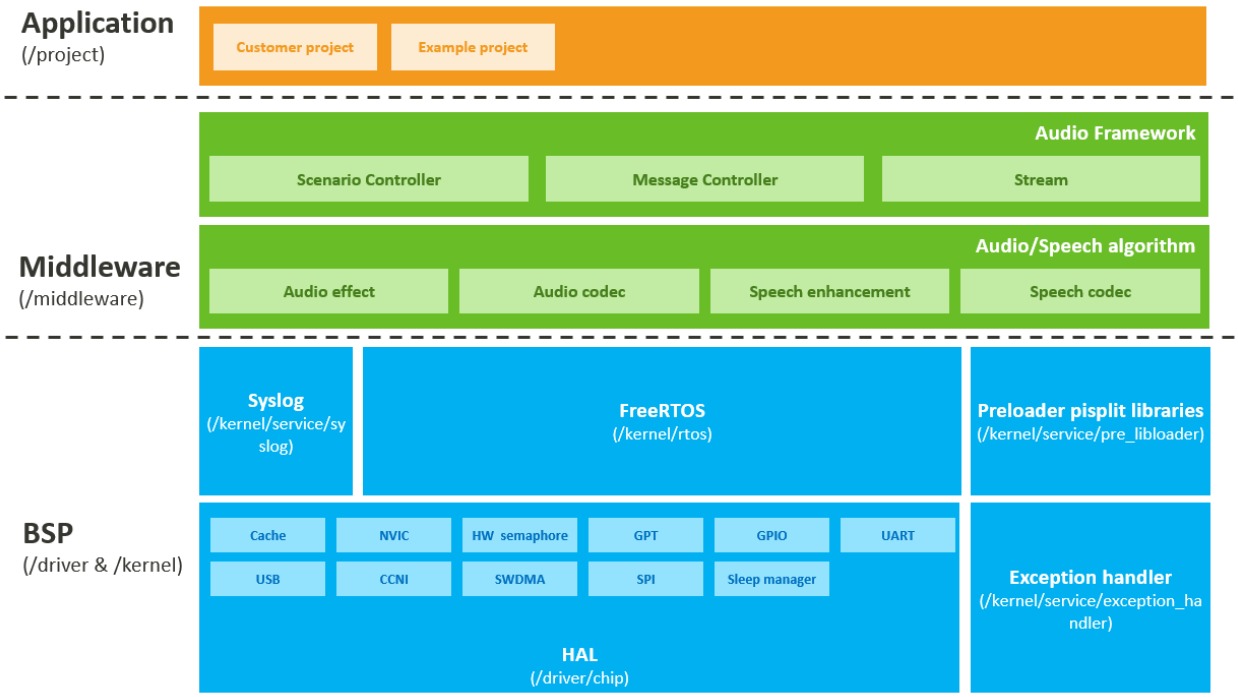

Figure 2. AB1565/AB1568 platform architecture layout

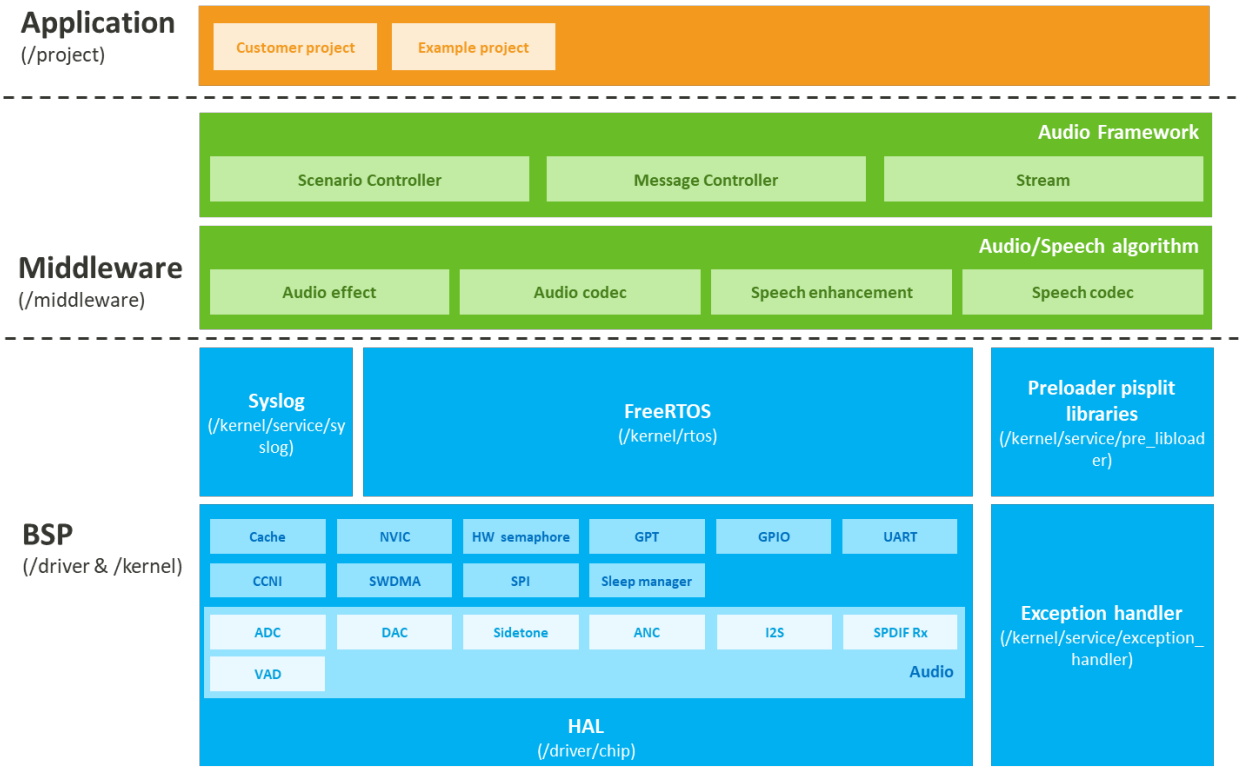

Figure 3. AB1585/AB1588 platform architecture layout

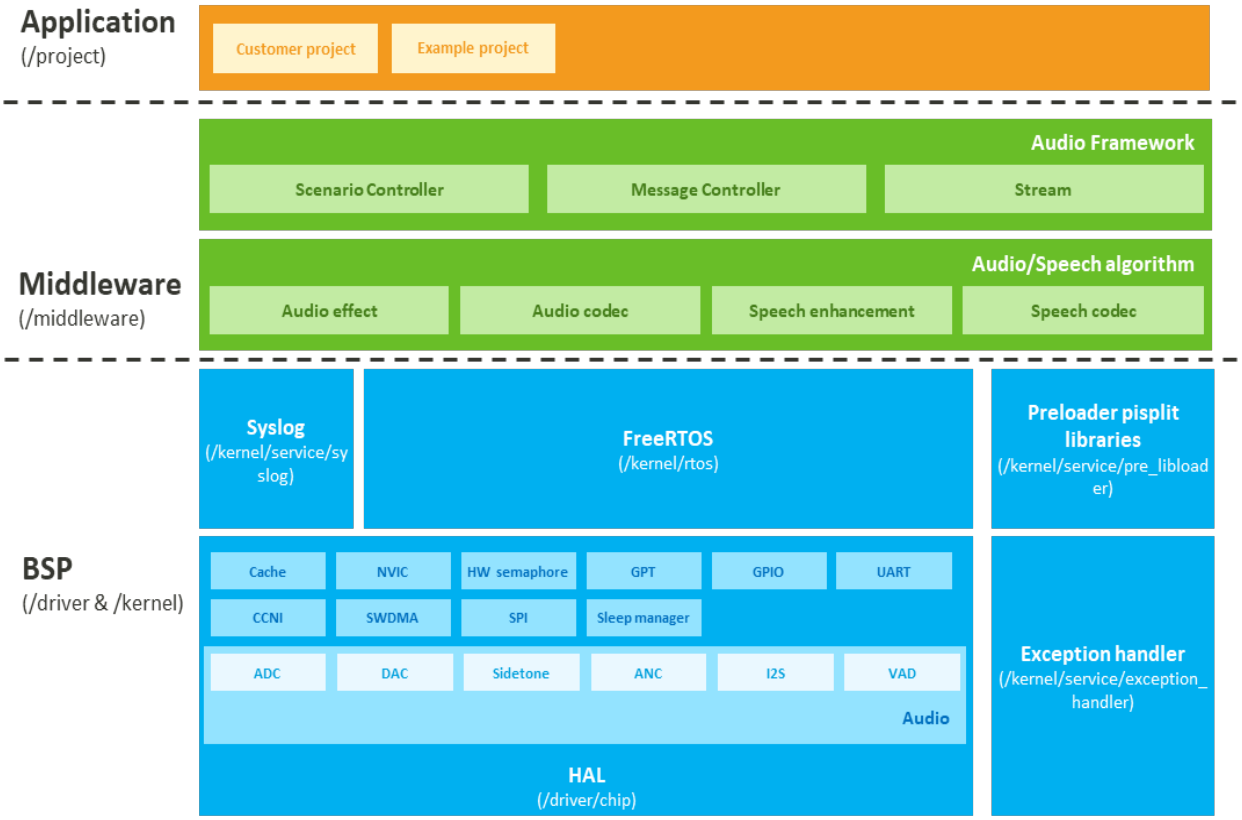

Figure 4. AB1571/AB1577/AB1627 platform architecture layout

A brief description of the layers is provided below.

BSP

- Hardware drivers – Provide peripheral drivers for the platform, such as ADC, I2S, I2C, SPI, RTC, GPIO, UART, Flash, Security Engine, TRNG, GDMA, PWM, WDT and IRDA TX/RX
- Hardware Abstraction Layer (HAL) – Provides the driver Application Programming Interface (API) encapsulating the low-level functions of peripheral drivers for the operating system (OS), Middleware features and Application.
- FreeRTOS – An OS with the open source software for the Middleware components and Application.
- Syslog – This module implements system logging for development and debugging.

Middleware

Middleware layer is composed of audio framework and audio/speech algorithm.

In the audio framework:

- Scenario controller – manages different types of scenarios, such as the eSCO scenario, A2DP scenario, and voice prompt scenario.
- Message controller – Manages message transmission between DSP and other processors such as CM4/CM33.
- Stream – Controls the data streaming and processes the audio/voice signal by using the resource of the audio/speech algorithm and hardware sample rate converter (SRC) according to different applications.

In the audio/speech algorithm:

- Audio codec – Includes SBC/AAC decoders for the A2DP application.
- Speech codec – Includes CVSD/mSBC decoders and encoders for the eSCO application.
- Audio effect – Includes equalizer (EQ) and dynamic range control (DRC) for variable and customized audio effect.
- Speech enhancement – Includes noise reduction (NR), echo cancellation (EC), and noise dependent volume control (NDVC) to enhance the signal performance for the listener.

1.2. Folder structure

The SDK is delivered as a single package and is organized in the folder structure shown in Figure 5 and Figure 6.

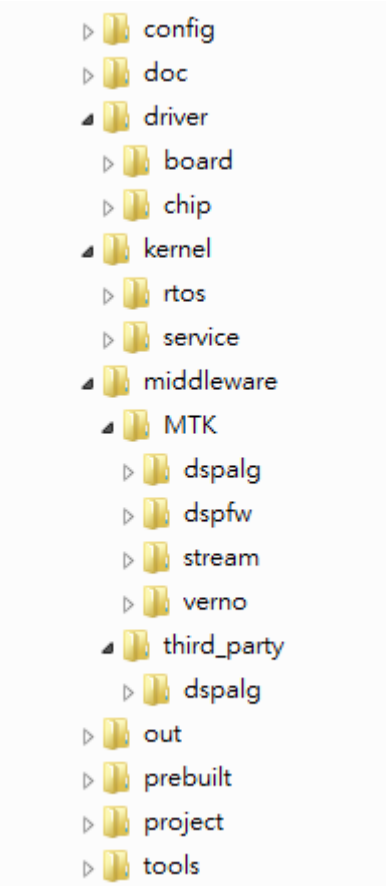

Figure 5. Folder structure for v2.x.x

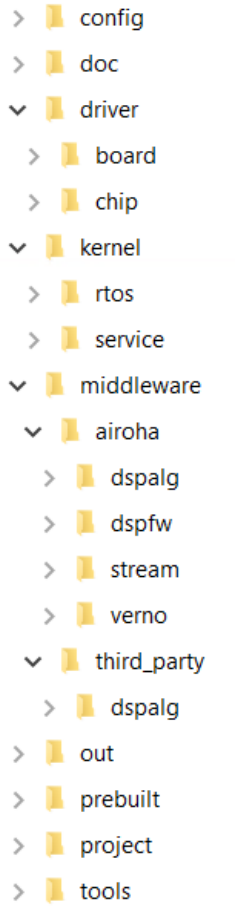

Figure 6. Folder structure for v3.x.x

This package contains the source and library files for the major components, the build configuration, related tools and documentation. A brief description of the layout of these files is as follows:

- `config` – Includes the make and compile configuration files for compiling a binary project.
- `doc` – Includes SDK related documentation, such as developer and SDK API reference guides.
- `driver` – Includes common driver

  - board – Includes driver files associated with the board.
  - chip – Includes common drivers for the modules associated with the chip, such as UART and I2C.
- kernel – Includes the underlying RTOS and system services for handling exception and logging errors.
- middleware
  - MTK or airoha – Includes middleware files created by Airoha.
    - dspalg – Includes the codecs and audio/voice processing swipe and the related driver wrapper.
    - dspfw – Includes the software framework for audio/voice processing and interactions with low-level driver interface.
    - stream – Handles the audio/voice data flow control.
  - third_party – Includes middleware files created by third parties.
    - dspalg – Includes the codecs and audio/voice processing swipe and the related driver wrapper.
- prebuilt – Contains the binary files, libraries, header files, makefiles and other pre-built files.
- project – The SDK includes example projects with pre-configured module features.
- tools – Includes tools to compile and build projects using the SDK.

1.3. Project source structure

The SDK provides a set of reference applications (i.e. projects with a single function showing how to use the drivers or other module features).

Example applications are located in <sdk_root>\dsp\project\<chip>\apps\<project name> and <sdk_root>\dsp\project\<chip>\templates\<project name>. The subsequent image shows the folder structure.

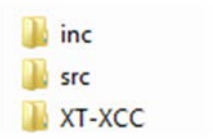

Figure 7. Project folder structure

- inc – Project header files
- src – Project source files
- XT-XCC – Xtensa Xplorer related project-configuration files, such as the makefile

You can apply the specific reference applications to your own development.

2. Setting up the DSP Development Tools

This section provides shows how to get started with the Airoha IoT development platform for DSP. It provides information about the subsequent items:

- Supported environments for development
- Running a project on the AB155x, AM255x, AB1565, AB1568, AB1571, A1577, AB1585, AB1588 EVKs
- Building a project with the SDK
- Creating your own project

2.1. Environment

A specific version of Ubuntu Linux is required, i.e. 18.10 64-bit or 18.04 LTS/20.04 LTS/22.04 LTS. You must install the Xtensa toolchain to build a DSP project. The subsequent software components are also required:

- XtensaTools RG_2017_7 for AB155x/AM255x
- XtensaTools RG_2019_12 for AB1565/AB1568 (With SDK v2.x.x)
- XtensaTools RI_2021_8 for AB1565/AB1568 (With SDK v3.x.x) and AB1571/AB1577/AB1585/AB1588/AB1627
- Xtensa License server daemon (The table shows the available function of the license provided by Airoha)

Functions supported by the license	Airoha license Support? (Y/N)
Compile source code	Y
Parse DSP side offline dump	N
GDB command online/offline debug	N
IDE online/offline debug by Xplorer	N
IDE simulation by Xplorer(ISS)	N

Complete the procedures shown in *Airoha_IoT_SDK_for_BT_Audio_Get_Started_guide* or *Airoha_IoT_SDK_for_Smart_MCU_Get_Started_Guide* to automatically set up the environment using a script.

Notes:

- The process for setting up the environment shown here has been integrated in a shell script, i.e. `install.sh`. The script is located in the SDK all-in-one package. You must first install Ubuntu Linux, and then run `install.sh`.
- `install.sh` can complete the SDK installation and automatically configure the toolchain, including Xtensa development tools. Refer to the subsequent documents for more information about `install.sh`:
  Airoha_IoT_SDK_for_BT_Audio_Get_Started_guide or
  Airoha_IoT_SDK_for_Smart_MCU_Get_Started_Guide; and
  Airoha_IoT_SDK_for_BT_Auido_Build_Environment_Guide or
  Airoha_IoT_SDK_for_Smart_MCU_Build_Environment_Guide.
- We strongly recommend that you use `install.sh` to complete the installation process. Refer to *Appendix – Setting up DSP Toolchain Manually* for more information.

2.2. Building a project using the SDK

You must use the script under <sdk_root>\dsp\build.sh when building a project using the SDK. For more information about the script, go to the dsp directory and execute the subsequent command:

```
cd <sdk_root>/dsp
./build.sh
```

The output appears as follows:

```
============================================================
Build Project
============================================================
Usage: ./build.sh <board> <project> [clean]

Example:
       ./build.sh ab1555_evk dsp0_freertos_create_thread
       ./build.sh clean
        (clean folder: out)
       ./build.sh ab1555_evk clean
        (clean folder: out/ab1555_evk)
       ./build.sh ab1555_evk dsp0_freertos_create_thread clean
        (clean folder: out/ab1555_evk/dsp0_freertos_create_thread)

Argument:
       -f=<feature makefile> or --feature=<feature makefile>
           Replace feature.mk with other makefile. For example,
           the feature_example.mk is under project folder,
           -f=feature_example.mk will replace feature.mk with
           feature_example.mk.

       -o=<make option> or --option=<make option>
           Assign additional make option. For example,
           to compile module sequentially, use -o=-j1.
           to turn on specific feature in feature makefile,
           use -o=<feature_name>=y to assign more than one options,
           use -o=<option_1> -o=<option_2>.

============================================================
List Available Example Projects
============================================================
Usage: ./build.sh list
```

Execute the subsequent command to show all available boards and projects:

```
./build.sh list
```

The available boards and projects are shown in a list according to the related configuration files under <sdk_root>/dsp/project/<chip>/<apps or templates>/<project>. The output appears in the console as follows.

```
============================================================
Available DSP Build Projects:
============================================================
  ab155x_evk
    dsp0_headset_ref_design
        |-feature.mk
        |-feature_ab1552_asia.mk
        |-feature_ab1552_evb.mk
        |-feature_ab1552_evk.mk
```

```
        |-feature_ab1555_evk.mk
        |-feature_ab1555_mini_board.mk
        |-feature_ab1556_evk.mk
        |-feature_ab1558_evk.mk
    dsp0_freertos_create_thread
        |-feature.mk
        |-feature_ab1552_evk.mk
    dsp0_no_rtos_initialize_system
        |-feature.mk
        |-feature_ab1552_evk.mk
```

Execute the subsequent command to build a specific project:

```
./build.sh <board> <project>
```

The output files are put under <sdk_root>\dsp\out\<board>\<project>.

For example, to build a project on ab155x_evk, execute the subsequent build command:

```
./build.sh ab155x_evk dsp0_freertos_create_thread
```

The output appears in the console as follows:

```
$./build.sh ab155x_evk dsp0_freertos_create_thread
FEATURE = feature.mk
make -C project/ab155x_evk/templates/dsp0_freertos_create_thread/XT-XCC
OUTDIR=<sdk_root>/dsp/out/ab155x_evk/dsp0_freertos_create_thread  2>>
<sdk_root>/dsp/out/ab155x_evk/dsp0_freertos_create_thread/log/err.log
make: Entering directory
...
```

The generated files are put under `<sdk_root>\dsp\out\ab155x_evk\ dsp0_freertos_create_thread\`.

The build script `<sdk_root>\dsp\build.sh` provides options for removing the generated output and clearing the folder.

To clean the `<sdk_root>\dsp\out` folder:

```
./build.sh clean
```

To clean the `<sdk_root>\dsp\out\<board>` folder:

```
./build.sh <board> clean
```

To clean the `<sdk_root>\dsp\out\<board>\<project>` folder:

```
./build.sh <board> <project> clean
```

2.3. Downloading a project to the EVK

To download a project to the EVK:

1) Copy all of the binaries to the same folder as `flash_download.cfg`.
2) Modify the binary names in `flash_download.cfg` according to your specific project binaries.

You must download the corresponding binaries of CM4/N9 and the DSP binary. You cannot import only the DSP binary.

Notes:

- `partition_table.bin` and `ab155x_bootloader.bin` (`am255x_bootloader.bin`) are mandatory bin files. The names are fixed for all projects.
- Binaries for N9/CM4/DSP0/DSP1, and the binary for CM4 is mandatory. Update the binary names according to your project.
- You must delete the corresponding ROM section if there is no image to download.

```
main_region:
    address_type: physical
    rom_list:
...
...
        - rom:
            file: ab155x_patch_hdr.bin
            name: N9
            begin_address: 0x08012000
        - rom:
            file: freertos_create_thread.bin
            name: CM4
            begin_address: 0x08032000
        - rom:
            file: dsp0_freertos_create_thread.bin
            name: DSP0
            begin_address: 0x0812C000
        - rom:
            file: dsp1_no_rtos_initialize_system.bin
            name: DSP1
            begin_address: 0x081D3000
```

Finally, download the binaries with IoT Flash Tool by using the previously modified `flash_download.cfg`.

Refer to *Airoha_IoT_SDK_Flash_Tool_Users_Guide.pdf* for more information and the instructions for using the flash tool.

2.4. Creating your own project

This section shows how to use an existing project to create your own project.

1) Copy an existing template project to the destination folder of the new project to be created.
2) Path of templates projects: `<SDK version>\dsp\project\<chip>\templates`
3) Rename the template project name to your project name.
4) Modify the ROOTDIR in the XT-XCC/Makefile of the new project. Make sure that ROOTDIR is mapping to the `dsp` folder.

```
…
################################################################################
########
# Project Configuration
################################################################################
########
PWD           :=  $(shell pwd)
ROOTDIR       :=  ../../../../..
```

```
PROJ_PATH    :=  $(patsubst $(abspath $(PWD)/$(ROOTDIR))/%,%,$(abspath
$(dir $(PWD))))
…
```

5) Make any necessary changes to the source code and add your application files to the new project.

3. Appendix – Setting up DSP Toolchain Manually

3.1. Installing system components

Complete the subsequent procedure to install the libraries that are required for the Xtensa toolchain:

1) $ sudo apt-get install build-essential
2) $ sudo dpkg --add-architecture i386
3) $ sudo apt-get update
4) $ sudo apt-get install libc6-i386 lsb

3.2. Installing the Xtensa toolchain

You must install the Xtensa toolchain package and DSP configuration package to build a DSP image. Get the Airoha IoT SDK for BT audio and tools all-in-one package for the required toolchain.

Platform	Package name	Chip	Package files
Linux	Xtensa toolchain package and license server	AB155x/AM255x	XtensaTools_RG_2017_7_linux.tgz
		AB1565/AB1568 (with SDK v2.x.x)	XtensaTools_RG_2019_12_linux.tgz
		AB1565/AB1568 (with SDK v3.x.x)	XtensaTools_RI_2021_8_linux.tgz
		AB1585/AB1588	
		AB1571/AB1577	
		AB1627	
		all chips	licserv_linux_x64_v11_13.tgz
	DSP configuration package	AB155x/AM255x	dsp0_core_winabi_xtensac_linux_redist.tgz dsp1_core_winabi_xtensac_linux_redist.tgz
		AB1565/AB1568	AB1568_i64B_d32B_512K_linux_redist.tgz
		AB1585/AB1588	AIR_PREMIUM_G3_HIFI5_linux_redist.tgz
		AB1571/AB1577	AIR_STEREO_HIGH_G3_MINI_A_linux_redist.tgz
		AB1627	AIR_STEREO_HIGH_G3_MINI_A_linux_redist.tgz

3.2.1. Installing the Xtensa toolchain package

`<sdk_root>` appears as follows. The `mcu` and `dsp` folders are available under `<sdk_root>` when you extract `IoT_SDK_for_BT_Audio_Vx.x.x.7z` to `/home/ab155x`. Your `<sdk_root>` is then `/home/ab155x/IoT_SDK_for_BT_Audio_V1.0.0`

Notes:

You must copy the subsequent files to <sdk_root>. AB155x_AM255x_DSP_Toolchain_Packages on MOL includes three files for AB155x/AM255x:

1) XtensaTools_RG_2017_7_linux.tgz
2) dsp0_core_winabi_xtensac_linux_redist.tgz
3) dsp1_core_winabi_xtensac_linux_redist.tgz

AB1565_AB1568_DSP_Toolchain_Packages on MOL includes four files for AB1565/AB1568:

1) XtensaTools_RI_2021_8_linux.tgz (For SDK v2.x.x)
2) AB1568_i64B_d32B_512K_linux_redist.tgz (For SDK v2.x.x)
3) XtensaTools_RI_2021_8_linux.tgz (For SDK v3.x.x)
4) AB1568_i64B_d32B_512K_linux_redist.tgz (For SDK v3.x.x)

AB1585_AB1588_DSP_Toolchain_Packages on MOL includes two files for AB1585/AB1588:

1) XtensaTools_RI_2021_8_linux.tgz
2) AIR_PREMIUM_G3_HIFI5_linux_redist.tgz

AB1571_AB1577_DSP_Toolchain_Packages on MOL includes two files for AB1571/AB1577:

1) XtensaTools_RI_2021_8_linux.tgz
2) AIR_STEREO_HIGH_G3_MINI_A_linux_redist.tgz

AB1627_DSP_Toolchain_Packages on MOL includes two files for AB1627:

1) XtensaTools_RI_2021_8_linux.tgz
2) AIR_STEREO_HIGH_G3_MINI_A_linux_redist.tgz

First, execute the subsequent command:

```
cd <sdk_root>/dsp/tools
mkdir xtensa
mv <sdk_root>/XtensaTools_RG_2017_7_linux.tgz xtensa/
mv <sdk_root>/dsp0_core_winabi_xtensac_linux_redist.tgz xtensa/
mv <sdk_root>/dsp1_core_winabi_xtensac_linux_redist.tgz xtensa/
```

To install the toolchain package, extract the contents of XtensaTools_RG_2017_7_linux.tgz

Execute the subsequent command to extract the file in a Linux environment:

```
cd xtensa
tar xvfz XtensaTools_RG_2017_7_linux.tgz
```

The toolchains are extracted to <sdk_root>/dsp/tools/xtensa/RG-2017.7-linux/XtensaTools/.

3.2.2. Installing the DSP configuration package

To install the DSP configuration package in a Linux environment, execute the subsequent command to extract two files into the xtensa folder.

```
cd <sdk_root>/dsp/tools/xtensa
tar xvfz dsp0_core_winabi_xtensac_linux_redist.tgz
```

```
tar xvfz dsp1_core_winabi_xtensac_linux_redist.tgz
```

The files are extracted to <sdk_root>/dsp/tools/xtensa/RG-2017.7-linux/dsp0_core_winabi_xtensac and <sdk_root>/dsp/tools/xtensa/RG-2017.7-linux/dsp1_core_winabi_xtensac.

Go to the dsp0_core_winabi_xtensac/ and dsp1_core_winabi_xtensac/ directories and execute the subsequent commands to launch the installer:

Note: <sdk_root> shown in the subsequent example must be the complete file path.

```
cd RG-2017.7-linux/dsp0_core_winabi_xtensac/
./install --xtensa-tools <sdk_root>/dsp/tools/xtensa/RG-2017.7-linux/XtensaTools --no-default
cd ../dsp1_core_winabi_xtensac/
./install --xtensa-tools <sdk_root>/dsp/tools/xtensa/RG-2017.7-linux/XtensaTools --no-default
```

The output appears in the console as follows when you successfully install the DSP configuration packages.

```
Non-interactive mode
  Xtensa Tools location:     /home/ab155x/Share/IoT_SDK_for_BT_Audio_V1.0.0.ER2/dsp/tools/xtensa/RG-2017.7-linux/XtensaTools/
  Xtensa Core registry:      /home/ab155x/Share/IoT_SDK_for_BT_Audio_V1.0.0.ER2/dsp/tools/xtensa/RG-2017.7-linux/XtensaTools//config
  Register as default:       no
  Replace same-named config: yes
```

Figure 8. Successful installation

3.2.3. Setting up Cadence license server for an Xtensa toolchain

You must first extract the contents of CDNSLICREQ-xxx.zip to set up a license server. The license package is available via the Airoha eService portal. The license file (i.e. AB155x_AM255x_DSP_XXXXXXXXXXXX.lic) is contained within the new directory.

Refer to section *3.2.3.2. Configuring environment variables* for more information about the setup procedure.

Note:

- Go to the Airoha eService portal and go to Customer Portal > Software Related > License. Submit a request for the Cadence license. Select the **floating license** license type on the application form. Download CDNSLICREQ-xxx.zip (attached to the JIRA item) after approval.

3.2.3.1. Setting up the license server on Linux

To set up a license server in a Linux environment:

1) Go to <sdk_root>.
2) Create a folder (xt_server) for the Xtensa toolchain Cadence license server.
3) Copy licserv_linux_x64_v11_13.tgz to the new folder.

```
mkdir xt_server
cp licserv_linux_x64_v11_13.tgz xt_server/
```

4) Extract the contents of licserv_linux_x64_v11_13.tgz.

5) Copy AB155x_AM255x_DSP_XXXXXXXXXXXX.lic to x64_lsb so the license file and lmgrd file are in the same folder.

```
cd  xt_server
tar -zxvf licserv_linux_x64_v11_13.tgz
cp <sdk_root>/AB155x_AM255x_DSP_XXXXXXXXXXXX.lic x64_lsb/
cd x64_lsb/
```

6) Check the Linux Hostname by executing one of the subsequent commands.

a) Method 1. Execute the subsequent command to get the hostname:

```
cat /etc/hostname
```

The output appears as follows:

```
airoha@/Projects/AB155x/xt_server/x64_lsb# cat /etc/hostname
AIROHA-XTS
```

b) Method 2. Execute the subsequent command to get host information:

```
cat /etc/hosts
```

The output appears as follows:

```
airoha@/Projects/AB155x/xt_server/x64_lsb# cat /etc/hosts
127.0.0.1       localhost
127.0.1.1       AIROHA-XTS
```

7) Open the license file (e.g., AB155x_AM255x_DSP_XXXXXXXXXXXX.lic)

8) Modify the host name, TCP port, and path for xtensad in the license file

   a) Host name: <SERVERNAME>

   b) TCP port: <PORT> (choose an un-used TCP port)

   c) Path of xtensad: </PATH/TO/XTENSAD>

Original license file:

```
SERVER <SERVERNAME> F0DEF1ABCFA0 <PORT>
#
USE_SERVER
VENDOR xtensad </PATH/TO/XTENSAD>
#
...
```

For example…

```
SERVER NB12010014 F0DEF1ABCFA0 5678
#
USE_SERVER
VENDOR xtensad xtensad
#
...
```

9) Execute the subsequent command to start the license server.

```
./lmgrd -c <license_file> -l <debug_log>
```

o e.g., ./lmgrd -c AB155x_AM255x_DSP_382C4A76B2CE.lic -l log.txt

10) Execute the subsequent command to check whether the license server has started.

```
ps aux | grep lmgrd
```

The output appears as follows if the license server has started:

```
airoha/Projects/AB155x/xt_server/x64_lsb$ ps aux | grep lmgrd
airoha     26850  0.0  0.0  17636  3136 pts/21   S    16:43   0:00
./lmgrd -c AB155x_DSP_382C4A76B2CE.lic -l log.txt
airoha     26851  0.1  0.1 141436 10084 ?        Ssl  16:43   0:00
xtensad -T alanlai-test02 11.13 3 -c :AB155x_DSP_382C4A76B2CE.lic:
-srv

DX9KLTvgVqwA7yhN5R1WZPWLh0sfp6S6GcJmybQF66RVfu4xfkkfmxWgWA5HdmO --
lmgrd_start 5c0f7897 -vdrestart 0
```

Note:

1. Make a note of the license server IP address that you use and change the value of the environment variable (i.e. LM_LICENSE_FILE) in <sdk_root>\dsp\.rule.mk.

2. It is only used to build load for our cadence license. Customers could not do other operation by cadence tool, such as the simulation in Xplorer.

3.2.3.2. Configuring environment variables

You must make changes to these three environment variables in <sdk_root>\dsp\.rule.mk to complete setting up the Cadence license server for an Xtensa toolchain...

1) LM_LICENSE_FILE
2) PATH
3) XTENSA_SYSTEM

Notes:

- <sdk_root> must be the complete path.
- The LM_LICENSE_FILE variable contains the TCP port and IP of your license server. The format is <tcp_port>@<ip>.

Use the subsequent example to set the PATH and XTENSA_SYSTEM variables:

```
export LM_LICENSE_FILE := 5280@127.0.0.1
export PATH := <sdk_root>/dsp/tools/xtensa/RG-2017.7-
linux/XtensaTools/bin:$(PATH)
export XTENSA_SYSTEM := <sdk_root>/dsp/tools/xtensa/RG-2017.7-
linux/XtensaTools/config
```

4. Appendix - Troubleshooting

This section shows how to fix common build errors.

4.1. Error message – License checkout failed

When a "License checkout failed" error occurs (as shown in the subsequent image), complete the subsequent procedure to resolve the issue. Refer to section *3.2.3.1. Setting up the license server on Linux* for more information.

Figure 9. License checkout failed error

1) Make sure the MAC address in the license file is the same as the Ethernet MAC address on your PC.
2) Make sure the hostname in the license file is the same as the hostname of your Ubuntu PC.
3) Make sure the TCP port in the license file is not used by a different network service on your PC.
4) Make sure the license server manager (lmgrd) has started. Execute the subsequent command to start the license server:

```
./lmgrd -c <license_file> -l <debug_log>
```

e.g., `./lmgrd -c AB155x_AM255x_DSP_382C4A76B2CE.lic -l log.txt`

5) Execute the subsequent command to check if the license server has started.

```
ps aux | grep lmgrd
```

4.2. Error message – xt-xcc: not found

When an "xt-xcc: not found" error occurs (as shown in the subsequent image), refer to Section 3.2.3.2. Configuring environment variables to make sure the settings for the `<sdk_root>\dsp\.rule.mk` environment variables are correct.

Figure 10. xt-xcc: not found error

Make sure the environment variables are the same as shown in the red frame in the subsequent image.

```
################################################################################
# Compiler Toolchain Settings
################################################################################
#Xtensa tool chain path & license setting,
#These settings can be configured either in a project's Makefile for the specific
#project or setting here for all the projets.
XTENSA_ROOT ?= /mtkeda/xtensa/Xplorer-7.0.7
XTENSA_VER ?= RG-2017.7-linux
ifeq ($(shell domainname), mcdswrd)
XTENSA_LICENSE_FILE ?= 7400@172.26.66.32
else
XTENSA_LICENSE_FILE ?= 7400@mtklc17
endif

export LM_LICENSE_FILE := 5280@127.0.0.1
export PATH := /home/ab155x/IoT_SDK_for_BT_Audio_V1.0.0/dsp/tools/xtensa/RG-2017.7-linux/XtensaTools/b
in:$(PATH)
export XTENSA_SYSTEM := /home/ab155x/IoT_SDK_for_BT_Audio_V1.0.0/dsp/tools/xtensa/RG-2017.7-linux/Xten
saTools/config
export XTENSA_CORE := $(XTENSA_CORE)
LM_LICENSE_FILE := $(strip $(LM_LICENSE_FILE))
XTENSA_CORE     := $(strip $(XTENSA_CORE))
XTENSA_SYSTEM   := $(strip $(XTENSA_SYSTEM))
```

Figure 11. Environment variables in .rule.mk