Airoha IoT SDK for BT Audio Build Environment Guide

Version: 1.16

Release date: 2 November 2022

Airoha Proprietary and Confidential. © 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

Document Revision History

Revision	Date	Description
1.0	31 March 2016	Initial release
1.2	30 June 2016	• Added support for build environment on Windows OS.
1.3	13 January 2017	Update providing: • A method to add a module. • Support for C++ source files.
1.4	5 May 2017	• Added details on the pre-built folder. • Added requirement to Install GNU Awk on LinkIt 2533 HDK.
1.5	19 July 2019	• Configuration for multi-processor build.
1.6	12 Nov 2019	• Renamed the document from “Airoha IoT SDK GCC Build Environment Guide” to “Airoha IoT SDK for BT Audio Build Environment Guide.” • Added 155x DSP build environment setup
1.7	21 Nov 2019	• Made a correction to the description of the install process
1.8	13 Dec 2019	• Revised examples and related description for AB155x.
1.9	16 Jun 2020	• Revise supported Linux versions
1.10	1 July 2020	• Added support for AB1565
1.11	24 July 2020	• Added support for AB1568
1.12	6 May 2021	• Remove unused description
1.13	21 Jan 2022	• Added support for AB1585/AB1588
1.14	23 Feb 2022	• Revised examples and related description to AB158x
1.15	10 March 2022	• Added feature option improvements in section 5 Makefiles
1.16	2 November 2022	• Refined the use of language and the formatting of text.

© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document,
in whole or in part, is strictly prohibited.

Table of contents

© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document,
in whole or in part, is strictly prohibited.

Lists of tables and figures

© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document,
in whole or in part, is strictly prohibited.

1. Overview

Airoha IoT Software Development Kit (SDK) for BT Audio build environment guide provides tools and utilities to install the supporting build environment and run your projects.

The information in this document guides you through:

- Setting up the build environment
- Building a project using the SDK
- Adding a module to the middleware
- Creating your own project

The build environment guide is applied to the Airoha IoT Development Platform including AB155x/AB1565/AB1568/AB1585/AB1588 EVK.

In this guide, all of the supported chips use the same installation procedure and build method as AB158x. The following examples use AB158x as a reference.

© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

2. Environment

This section provides detailed information on how to set up the SDK build environment with default GCC on Linux OS and on Microsoft Windows using MSYS2 cross-compilation tool.

2.1. Installing the SDK build environment on Linux

2.2. Preparing Linux build environment

The toolchain provided with the SDK is required to set up the build environment on Linux OS (Ubuntu 18.10 64bit or Ubuntu 18.04 LTS).

Table 1. Recommended build environment

Item	Description
OS	• Linux OS 18.10 or 18.04 LTS
Make	• GNU make 3.81

To install the SDK and build environment on Linux:

Download IoT_SDK_For_BT_Audio_and_Linux_Tools_All_In_One from Mediatek OnLine (MOL). The package contains ARM GCC toolchain, Cadence DSP toolchain for Linux OS, and Airoha IoT SDK for BT Audio. The package also contains an install script to setup the environment.

1) Select a folder as an installation working folder and unpack IoT_SDK_For_BT_Audio_and_Linux_Tools_All_In_One to the folder. The folder name should not contain any special characters such as space and brace.

2) Open install.sh in the install working folder for editing. Set a free network port (we strongly suggest using a port number between 1024 and 9999) for Cadence compiler license service daemon: change the value of DSP_LICSERV_PORT. Please do the setting especially if you are working on a Linux host sharing with multiple users.

```
#!/bin/bash

...
DSP_LICSERV_PORT=6677
...
```

3) Execute ./install.sh. The installation process requires Linux root permission and an internet connection. The script will:

   - install Cadence license tool chain;
   - get Cadence toolchain license; and
   - unzip the Airoha IoT SDK for BT Audio.

```
./install.sh
```


© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

The following messages appear when the installation process is complete:

```
...
...
Installation done.
!!!!! Please remember to run '~/airoha_sdk_toolchain/start_lic_server'
again if you reboot the system !!!!!
Now you can start the first build, please change directory to bta_sdk
and execute build command.
Example:
       cd bta_sdk
       ./build.sh ab1585_evk earbuds_ref_design
```

4) Build your project. You must wait before starting your first build because the Cadence license service daemon takes a few minutes to start after the installation process. Run the subsequent commands to build you project.

```
cd bta_sdk
./build.sh ab1585_evk earbuds_ref_design
```

The subsequent text appears on the screen when the build process is complete.

```
$./build.sh ab1585_evk earbuds_ref_design
cd /home/airoha/<All in one root>/bta_sdk/dsp
============================================================
Start DSP0 Build
============================================================
output folder change to:
...
...
trigger by build.sh, skip clean_log
Assembling...
../../../../../driver/chip/ab158x/dsp0/src_core/sleep_exit_prepare.S
...
...
```

The installation process puts the toolchain under ~/airoha_sdk_toolchain. You do not need to run the install script again for a new SDK release; Just get the SDK standalone package “IoT_SDK_For_BT_Audio” and unzip it to use the new SDK package. Do not remove or modify ~/airoha_sdk_toolchain because it causes the build environment broken.

2.3. Starting the Cadence toolchain license service daemon

The installation process starts the Cadence license service daemon. However, you must restart the server after the system reboots. You can use the subsequent command under the installation working folder to restart the server. The Cadence licenser service daemon takes approximately two minutes to start the service after launch.

```
~/airoha_sdk_toolchain/start_lic_server.sh
```

2.4. Offline installation

The installation process must connect to the internet to get the Cadence license. You must manually install the license if you are installing offline.

© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

- Submit a JIRA request to get a **floating** license.
- Revise the install.sh

    a. modify value of DSP_LIC_FILE to your license file

You can use the following Airoha eService Cadence License Request to get <license name>.lic:
Airoha eService website > Project > Your Project > Customer channels > Customer portal > Software Releated > License Request.

    b. remove the line CADENCE_LIC_HOST=xxxxx

```
#!/bin/bash

DSP_LIC_FILE="<license name>.lic"
...

#CADENCE_LIC_HOST=lic.airoha.com.tw
...
```

Run the install.sh for install SDK offline.

You can watch a video of the installation process (online and offline) here. Please refer to the "AB155x_Linux_Dev_Env_Setup.mp4"**.** The installation process for this build environment is the same.

2.5. Installing the SDK build environment on Microsoft Windows

To install the SDK and build environment on Windows, please download and extract the content of the IoT_SDK_For_BT_Audio_and_Windows_Tools_All_In_One on your host. The package contains MSYS, ARM GCC for Windows, and Cadence DSP toolchain for Windows, and the Airoha IoT SDK for BT Audio. The subsequent sections show the instructions for installing the SDK and build environment:

2.6. Preparing the build environment

1) Select a folder as an installation working folder and unpack IoT_SDK_For_BT_Audio_and_Windows_Tools_All_In_One to the folder. The folder name must not contain special characters.

2) Execute install_msys.bat to install msys2. You can skip this step if you have already installed msys. This step requires an internet connection. Refer to the MSYS2 page here for information about performing an offline installation.

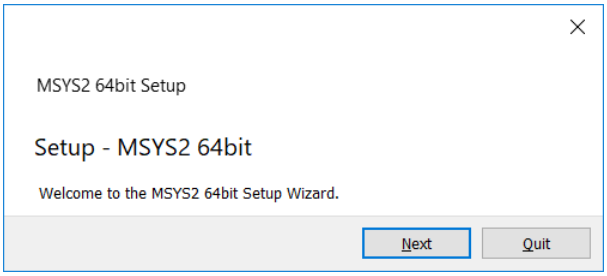


Figure 1. Installing MSYS2

3) Execute msys2.exe. An msys2 terminal window appears on the screen.

4) Key `./install.sh` into the msys2 terminal. The installation process requires an internet connection. The script performs the following actions:

© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

- Install the required MSYS software components (i.e. diffutils, make, p7zip., and tar)
- Install the Cadence license tool chain
- Get the Cadence toolchain license
- Unzip the Airoha IoT SDK for BT Audio

The following text appears on the screen when the installation process is complete:

```
$ ./install.sh
MSYS package installing
loading packages...
...
...
MSYS package install done
...
...
Extracting archive: IoT_SDK_for_BT_Audio_V3.0.0.7z
...
...

  0% 105 - mcu/tools/gcc/win/gcc-arm-none- ....
```

5) Build your first project

```
cd bta_sdk
./build.sh ab1585_evk earbuds_ref_design
```

The following text appears on the screen when the build process is complete:

```
$./build.sh ab1585_evk earbuds_ref_design
...
============================================================
Start DSP0 Build
============================================================
output folder change to:
...
...
trigger by build.sh, skip clean_log
Assembling...
../../../../../driver/chip/ab158x/dsp0/src_core/sleep_exit_prepare.S
...
...
```

The installation process puts the toolchain under ~/airoha_sdk_toolchain. You do not need to run the install script again for a subsequent SDK release, just get the SDK standalone package "IoT_SDK_For_BT_Audio" and unzip to use the new SDK package. Do not remove or modify ~/airoha_sdk_toolchain because it will cause the build environment to break.

2.7. Offline installation

The installation process must connect to the internet to get the Cadence license.

If you are installing the build environment offline, you must:

1) Manually install the license

© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

2) Submit a JIRA request to get a node-locked license.

3) Revise the install.sh

   a. modify value of DSP_LIC_FILE to your license file

You can get the <license name>.lic from the Airoha eService Cadence License Request:
https://eservice.airoha.com.tw/servicedesk/customer/portal/234

   b. remove the line CADENCE_LIC_HOST=lic.airoha.com.tw

```
#!/bin/bash

DSP_LIC_FILE="<license name>.lic"
...
#CADENCE_LIC_HOST=lic.airoha.com.tw
...
```

4) Run the install.sh for install SDK offline.

You can watch a video of the installation process (online and offline) here. Please refer to "AB155x_Windows_Dev_Env_Setup.mp4". The installation process for this build environment is the same.

2.8. Troubleshooting

Please note the following items when building your project with MSYS2.

- The folder name and file name in the Airoha IoT SDK must not contain ' ', ,'(', ')', '[' or ']' characters.
- The sum of the path length SDK installed folder and project name should less than 35 characters in length. Otherwise, a build error similar to the one shown below may occur.

```
Arm-none-eabi-gcc.exe: error:
../../../../../out/ab158x_evk/i2c_communication_with_EEPROM_dma/obj/proj
ect/ab158x/hal_examples/i2c_communication_with_EEPROM_dma/src/system_ab1
58x.o: No such file or directory
```

- The makefile in your project must not use any platform dependent commands or files, such as `stat` or `/proc/cpuinfo`.
- By default, the parallel build feature is enabled in `build.sh` to make the compilation process faster. Disable the parallel build feature if there are any irregular build error or system exceptions.
- To disable the parallel build feature for all modules, change the value "-j" of EXTRA_VAR to "-j1" in <SDK_ROOT>/mcu/build.sh and <SDK_ROOT>/dsp/build.sh as shown below.

```
platform=$(uname)
if [[ "$platform" =~ "MINGW" || "$platform" =~ "MSYS" ]]; then
    max_jobs=$(( $(WMIC CPU Get NumberOfLogicalProcessors|tail -2|awk
'{print $1}') - 1))
else
    max_jobs=`cat /proc/cpuinfo |grep ^processor|wc -l`
fi
```


© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

```
export EXTRA_VAR=-j$max_jobs
```

- If the parallel build of a specific module causes build errors, set the value <module>_EXTRA as "-j1" in <SDK_ROOT>/mcu/.rule.mk to disable the parallel build for that specific module as shown below.

```
OS_VERSION := $(shell uname)
ifneq ($(filter MINGW% MSYS%,$(OS_VERSION)),)
  $(DRV_CHIP_PATH)_EXTRA    := -j1
  $(MID_MBEDTLS_PATH)_EXTRA := -j1
  $(<module>)_EXTRA         := -j1
endif...
```

Then, rebuild your project.

```
./build.sh ab1585_evk earbuds_ref_design clean
./build.sh ab1585_evk earbuds_ref_design
```


© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

3. Building the Project Using the SDK

3.1. Building projects

Build MCU and DSP projects using the script in <sdk_root>/build.sh. To find more information about the script, navigate to the SDK root directory and execute the following command:

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

The following text appears on the screen:

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

Example:
       ./build.sh ab158x earbuds_ref_design
       ./build.sh ab158x earbuds_ref_design -fm=feature_ab1585_evk.mk
       ./build.sh clean
        (clean folder: out)
       ./build.sh ab158x clean
        (clean folder: out/ab158x_evk)
       ./build.sh ab158x earbuds_ref_design clean
        (clean folder: out/ab158x /earbuds_ref_design)

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

       -fd0=<feature makefile>
           Replace feature.mk with other makefile for dsp0. For example,
           the feature_example.mk is under project folder,
           -fd0=feature_example.mk will replace feature.mk with
           feature_example.mk.

       -fd1=<feature makefile>
           Replace feature.mk with other makefile for dsp1. For example,
           the feature_example.mk is under project folder,
           -fd1=feature_example.mk will replace feature.mk with
           feature_example.mk.

       -mcu
           Build MCU only. (Use default dsp bin at 'dsp/prebuilt/dsp0/'
           instead of build dsp bin)."

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

- List all available boards and projects.

Run the command to show all available boards and projects:

© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

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

The available boards and projects are listed below.

```
============================================================
Available Build Projects:
============================================================
...
ab158x_evk
  earbuds_ref_design
      MCU: earbuds_ref_design
      dsp0: dsp0_headset_ref_design
ab158x_evk
  headset_ref_design
      MCU: headset_ref_design
      dsp0: dsp0_headset_ref_design
...
```

3.1. Build the project.

To build a specific project, simply run the following command:

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

The generated files are put in the <sdk_root>/out/<board>/<project> folder.

NOTE: Use the **–mcu** option to skip the DSP build.

For example, to build a project in the AB158x EVK, run the following build command:

```
./build.sh ab158x earbuds_ref_design
```

The standard output in the terminal window appears as follows:

```
$./build.sh ab158x earbuds_ref_design
cd /d/131/bta_sdk/dsp
============================================================
Start DSP0 Build
============================================================
output folder change to:
/d/131/bta_sdk/dsp/out/ab158x_evk/dsp0_headset_ref_design/feature
make -C project/ab158x_evk/apps/dsp0_headset_ref_design/XT-XCC
OUTDIR=/d/131/bta_sdk/dsp/out/ab158x_evk/dsp0_headset_ref_design/feature
-j5 FEATURE=feature.mk
OUT=out/ab158x_evk/dsp0_headset_ref_design/feature 2>>
/d/131/bta_sdk/dsp/out/ab158x_evk/dsp0_headset_ref_design/feature/log/er
r.log
make: Entering directory
'/d/131/bta_sdk/dsp/project/ab158x_evk/apps/dsp0_headset_ref_design/XT-
XCC'
trigger by build.sh, skip clean_log
Assembling... ../../../../../driver/chip/ab158x/dsp0/src_core/sleep_exit
_prepare.S
Assembling... ../../../../../kernel/service/exception_handler/src/except
ion.S
```

The generated files are added to the <sdk_root>/out/ab158x_evk/earbuds_ref_design/ folder.

© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

3.2. Cleaning the output folder

The build script <sdk_root>/build.sh provides options for removing the generated files as shown below.

Clean the <sdk_root>/out folder.

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

Clean the <sdk_root>/out/<board> folder.

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

Clean the <sdk_root>/out/<board>/<projecct> folder.

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

The output folder is defined under variable BUILD_DIR in the Makefile in <sdk_root>/mcu/project/ab158x_evk/apps/earbuds_ref_design/GCC:

```
BUILD_DIR = $(PWD)/Build
PROJ_NAME = $(shell basename $(dir $(PWD)))
```

A project image earbuds_ref_design.bin is generated under <sdk_root>/mcu/project/ab158x_evk/apps/earbuds_ref_design/GCC/Build.

3.3. Building the MCU project with the "bl" option

By default, the pre-built bootloader image file is copied to the <sdk_root>/mcu/out/<board>/<project>/ folder when the project is built. The primary purpose of the bootloader image is to download the Flash Tool.

Apply the "bl" option to rebuild the bootloader and use the generated bootloader image file instead of the pre-built file as shown below.

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

To build the project on the AB158x EVK:

```
cd <sdk_root>/mcu
./build.sh ab158x earbuds_ref_design bl
```

The generated image file of the project and the bootloader and the merged image file flash.bin are put under <sdk_root>/mcu/out/ab158x_evk/earbuds_ref_design folder.

3.4. Building the project from the MCU and DSP project configuration directory

To build the project:

1) Change the current directory to the project source directory where the SDK is located.

   There are makefiles provided for the project build configuration. For example, the MCU project earbuds_ref_design is built by the project makefile under <sdk_root>/mcu/project/ab158x_evk/apps/earbuds_ref_design/GCC.

   The DSP project dsp0_headset_ref_design is built by the project makefile under

   <sdk_root>/dsp/project/ab158x_evk/apps/dsp0_headset_ref_design/XT-XCC.

Navigate to the example project's location.

For MCU project earbuds_ref_design:

```
cd <sdk_root>/mcu/project/ab158x_evk/apps/earbuds_ref_design/GCC
```

For DSP project dsp0_headset_ref_design:

© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

```
cd <sdk_root>/dsp/project/ab158x_evk/apps/dsp0_headset_ref_design/XT-XCC
```

2) Run the `make` command.

```
Make
```

The MCU project output folder is defined under variable `BUILD_DIR` in the `Makefile` located at `<sdk_root>/mcu/project/ab158x_evk/apps/earbuds_ref_design/GCC`:

```
BUILD_DIR = $(PWD)/build
PROJ_NAME = earbuds_ref_design
```

A project image `earbuds_ref_design.bin` is generated under `<sdk_root>/mcu/project/ab158x_evk/apps/earbuds_ref_design/GCC/build`.

The DSP project output folder is defined under variable `OUT_DIR` in the `Makefile` located at `<sdk_root>/dsp/project/ab158x_evk/apps/dsp0_headset_ref_design/XT-XCC`:

```
OUTDIR       :=  $(abspath $(strip $(ROOTDIR))/out/$(strip
$(BOARD))/$(strip $(PROJ_NAME))/$(strip $(FEATURE_BASENAME)))
PROJ_NAME :=  $(notdir $(shell cd ../ ; pwd))
```

A project image `dsp0_headset_ref_design.bin` is generated under `<sdk_root>/dsp/out/ab158x/dsp0_headset_ref_design/feature`.

© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

4. Folder Structure

This section shows the structure of the SDK and introduces the content of each folder. The directory of the SDK package is organized into the folder structure shown in Figure 2.

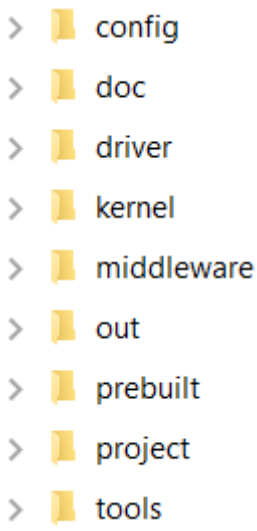


Figure 2. SDK package folder structure

This package contains the source and library files of the main features, the build configuration, related tools, and documentation. A brief description of the layout of these files is provided below:

- config — includes 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 files, such as board drivers, peripheral and CMSIS-CORE interface drivers.
- kernel — includes the underlying RTOS and system services for exception handling and error logging.
- middleware — includes software features for HAL and OS, such as network and advanced features.
- out — contains binary files, libraries, objects and build logs.
- prebuilt — contains binary files, libraries, header files, makefiles and other pre-built files.project — includes pre-configured example and demo projects using Wi-Fi, HTTP, HAL, and more.
- tools — includes tools to compile, download and debug projects using the SDK.

To enable building several different projects simultaneously, each project's generated files are placed under the corresponding `<sdk_root>/out/<board>/<project>/` folder. The dependency of mcu/dsp side projects can be used to build the bin in the `<sdk_root>/out/<board>/<project>/` folder. Please refer to Section 7.6 "Configuring multi-processor products" for more information.

The target folder path for AB158x is the `<sdk_root>/out/ab158x_evk/earbuds_ref_design/` folder.

The following list provides a brief description of the binary files:

© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

project image — the naming rule of the image file is PROJ_NAME.bin, e.g., earbuds_ref_design.bin. The variable PROJ_NAME is defined in Makefile, see section 5, “Makefiles”.

- bootloader image — the naming rule of the image file is bootloader.bin.
- elf file — contains information about the executable, object code, shared libraries and core dumps.
- map file — contains the link information of the project libraries.
- lib folder — contains module libraries.
- log folder — contains build log including build information, timestamp and error messages.
- obj folder — contains object and dependency files.

© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

5. Makefiles

The SDK package contains three makefiles, including the following:

Project Makefile

1) Makefile - Mainly used to generate the project image.

Configuration makefiles

2) feature.mk - Feature configuration file
3) chip.mk - Chip configuration file

We have made the following improvements to the AB158x series chipset:

- Simplified feature_*.mk - Removed unused feature options and rules, synchronized option names with other Airoha chips, and made the optional names consistent with the same functionality between MCU and DSP sides. Adjustments use one feature option for only one feature whenever possible.
- Comments - Added comments to describe the usage and notes about the feature options in feature_*.mk to make it easier for you to understand.
- Dependency check - Sets the dependency rules for the build flow of the project to check feature option settings.

Please use the **earbuds_ref_design** project as a reference for more information about the usage and relationship of each makefile.

5.1. Project Makefile

The project makefile is under the <sdk_root>/mcu/project/<board>/apps/<project_name>/GCC/ folder. The purpose of the project makefile is as follows:

Configure project settings, including the root directory, project name, project path, and more.

Include other makefiles for the configuration, such as `feature.mk`, `chip.mk` and `module.mk`.

Set the file path of the project's source code.

Set the include path of the project's header files.

Set the dependency rules for the build flow of the project.

- Add a make rule to check for feature options conflict and stop the build if there is a conflict. (Error message tag "**[Conflict feature option**]")
  - For example, environment detection (ED) and ANC are dependent; If you want to enable ED but forget to enable the ANC feature, the following message appears to tell you to enable ANC: "[Conflict feature option] To enable AIR_ANC_ENVIRONMENT_DETECTION_ENABLE must support AIR_ANC_ENABLE."
- Add a make rule to check for add-on feature options. (Error message tag "**[Addon feature option fail]**") In this case, please contact the Airoha Customer Program Management (CPM) team to request the add-on package.

© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

Set the module libraries to link when creating the image file.

Trigger to make a command for each module to create a module library.

5.2. Configuration makefiles

This section provides more information about the configuration makefiles, `feature.mk` and `chip.mk`.

The `feature_*.mk` is located in the <sdk_root>/mcu/project/<board>/apps/earbuds_ref_design/GCC/ folder. You can turn set a value for a specific feature or turn it on or off by simply making a change in the feature_*.mk. The option comments provide more information about using the feature. The template for option comments is as follows.

```
# Give a brief introduction to this feature option.
# (By case) It must be turned on/off for both DSP and MCU, otherwise, it
will not work.
# (By case) Dependency description
# (By case) For value case, describe each value of the feature option.
<feature_option> = y/n or value
```

The `IC_CONFIG` and `BOARD_CONFIG` are also defined in the `feature.mk`.

```
IC_CONFIG                               = <chip>
...
BOARD_CONFIG                            = <board>
...

# This option is used to enable/disable User Unaware adaptive ANC.
# It must be turned on/off for both DSP and MCU, otherwise, it will not
work.
# Dependency: AIR_ANC_ENABLE must be enabled when this option is set to
y.
AIR_ANC_USER_UNAWARE_ENABLE = n
...

# This option is to choose the uplink rate. Default setting is none.
# It must be set to the same value for both DSP and MCU, otherwise, it
will not work.
# Up Link Rate : none, 48k
#                none : uplink rate will be handled by scenario itself.
#                48k  : uplink rate will be fixed in 48k Hz.
AIR_UPLINK_RATE   = none
...
```

The `chip.mk makefile` is located at <sdk_root>/mcu/config/chip/<board>/chip.mk and defines the common settings, compiler configuration, and the path and middleware module path of the chip. The major functions of the chip.mk are as follows:

- Configure the common settings of the chip.
- Define the CFLAGS macro.
- Set the include path of the kernel and the driver header file.
- Set the module folder path that contains the makefile.

The `chip.mk` makefile includes the `CFLAGS`, including the paths and module folder paths, as shown below.

```
...
MTK_SYSLOG_VERSION_2                    ?= y
MTK_SYSLOG_SUB_FEATURE_STRING_LOG_SUPPORT = y
```


© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

```
MTK_SYSLOG_SUB_FEATURE_BINARY_LOG_SUPPORT = y
MTK_SYSLOG_SUB_FEATURE_USB_ACTIVE_MODE = y
MTK_SYSLOG_SUB_FEATURE_OFFLINE_DUMP_ACTIVE_MODE = y
MTK_CPU_NUMBER_0                        ?= y
FPGA_ENV                                ?= n
...

AR      = $(BINPATH)/arm-none-eabi-ar
CC      = $(BINPATH)/arm-none-eabi-gcc
CXX     = $(BINPATH)/arm-none-eabi-g++
OBJCOPY = $(BINPATH)/arm-none-eabi-objcopy
SIZE    = $(BINPATH)/arm-none-eabi-size
OBJDUMP = $(BINPATH)/arm-none-eabi-objdump

...

COM_CFLAGS += $(ALLFLAGS) $(FPUFLAGS) -ffunction-sections -fdata-
sections -fno-builtin -Wimplicit-function-declaration
COM_CFLAGS += -gdwarf-2 -Os -Wall -fno-strict-aliasing -fno-common
COM_CFLAGS += -Wall -Wimplicit-function-declaration -
Werror=uninitialized -Wno-error=maybe-uninitialized -Werror=return-type
COM_CFLAGS += -DPCFG_OS=2 -D_REENT_SMALL -Wno-error -Wno-switch
COM_CFLAGS += -DPRODUCT_VERSION=$(PRODUCT_VERSION)
COM_CFLAGS += -D$(TARGET)_BOOTING

...

#Incldue Path
COM_CFLAGS += -I$(SOURCE_DIR)/middleware/third_party/mbedtls/include
COM_CFLAGS += -I$(SOURCE_DIR)/middleware/third_party/mbedtls/configs

CFLAGS     += -std=gnu99 $(COM_CFLAGS)
CXXFLAGS   += -std=c++11 $(COM_CFLAGS)

...
```


© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

6. Adding a Module to the Middleware

This section provides detailed information about adding a module or a custom defined feature to an existing project. The added module is compiled, archived, and linked with other libraries to create the final image file during the project build. The following example shows how to add a module named "mymodule" to the earbuds_ref_design project on the AB158x EVK development board.

6.1. Files to add

6.1.1. Source and header files

Create a module folder with the module name under <sdk_root>/mcu/middleware/third_party/ folder. This folder will contain the module files. The module source and header files must be placed under the "src" and the "inc" folders, respectively, as shown in Figure 3.

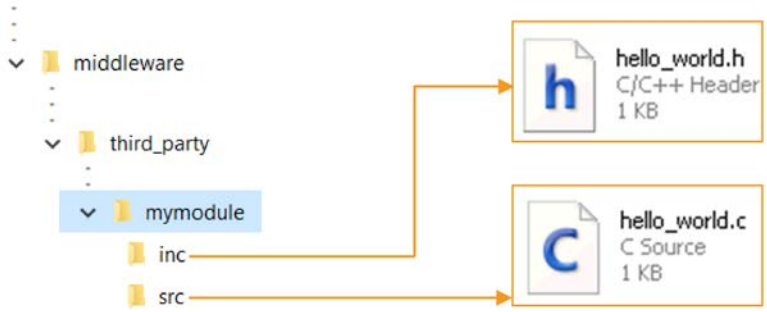


Figure 3. Module source and header files under the module folder

The sample source code hello_world.c and header file hello_world.h with their corresponding locations are shown below:

<sdk_root>/mcu/middleware/third_party/mymodule/src/hello_world.c

```
#include "hello_world.h"

void myFunc(void)
{
    printf("%s", "hello world\n");
}
```

<sdk_root>/mcu/middleware/third_party/mymodule/inc/hello_world.h

```
#ifndef __HELLO_WORLD__
#define __HELLO_WORLD__

#include <stdio.h>

void myFunc(void);

#endif
```


© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

6.1.2. Makefiles for the module

Create a makefile under the mymodule folder (as shown in Figure 4) named `module.mk`. It defines the module path and module sources that must be compiled and it must include the path that the compiler uses to find the header files during compilation.

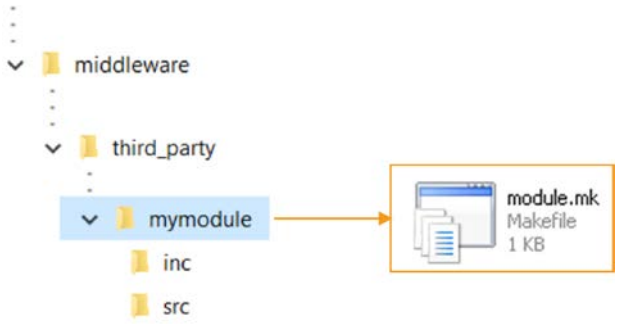


Figure 4. Creating a module.mk under the module folder

In this example, `module.mk` is at `<sdk_root>/mcu/middleware/third_party/mymodule/module.mk`. `C_FILES` and `CFLAGS` are built-in variables that store the module's .c source files and the paths. The corresponding built-in variables `CXX_FILES` and `CXXFLAGS` support compiling the source files (e.g., `.cpp`) of the module.

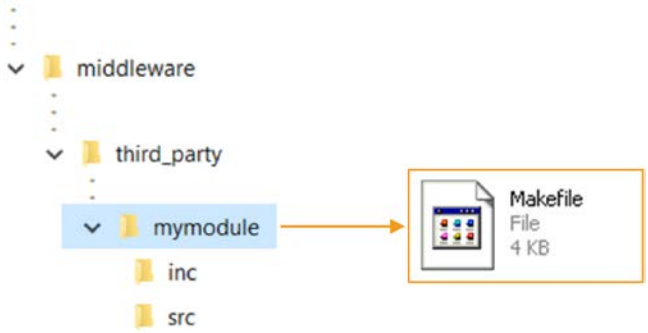
```
#module path
MYMODULE_SRC = middleware/third_party/mymodule

#source file
C_FILES  +=  $(MYMODULE_SRC)/src/hello_world.c
CXX_FILES+=

#include path
CFLAGS   += -I$(SOURCE_DIR)/middleware/third_party/mymodule/inc
CXXFLAGS +=
```

In addition to the module.mk makefile, another makefile under mymodule folder named Makefile (<sdk_root>/mcu/middleware/third_party/mymodule/Makefile) is required to generate a module library, as shown in Figure 5.

Most of the dependency rules and definitions in the makefile are written for a common use. You can simply copy the subsequent code and make any necessary changes to the values of the `PROJ_PATH` and `TARGET_LIB` variables. The `PROJ_PATH` variable is the path to the project folder that contains the makefile. The `TARGET_LIB` variable is the library for the added module.



© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

Figure 5. Creating a makefile under the mymodule folder

```
SOURCE_DIR  = ../../..
BINPATH     = ~/gcc-arm-none-eabi/bin
PROJ_PATH   = ../../../project/ab158x_evk/apps/earbuds_ref_design/GCC
CONFIG_PATH ?= .

CFLAGS += -I$(PROJ_PATH)/../inc
CFLAGS += -I$(SOURCE_DIR)/$(CONFIG_PATH)


FEATURE    ?= feature.mk
include $(PROJ_PATH)/$(FEATURE)

# Global Config
-include $(SOURCE_DIR)/.config
# IC Config
-include $(SOURCE_DIR)/config/chip/$(IC_CONFIG)/chip.mk
# Board Config
-include $(SOURCE_DIR)/config/board/$(BOARD_CONFIG)/board.mk

# Project name
TARGET_LIB=libmymodule

BUILD_DIR = Build
OUTPATH   = Build

# Sources
include module.mk

C_OBJS   = $(C_FILES:%.c=$(BUILD_DIR)/%.o)
  CXX_OBJS = $(CXX_FILES:%.cpp=$(BUILD_DIR)/%.o)

.PHONY: $(TARGET_LIB).a

all: $(TARGET_LIB).a
    @echo Build $< Done

include $(SOURCE_DIR)/.rule.mk

clean:
    rm -rf $(OUTPATH)/$(TARGET_LIB).a
    rm -rf $(BUILD_DIR)
```


© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

6.2. Adding a module to the build flow of the project

The rules for compiling module sources to a single library are now complete and the module is ready to build. To add the module into the project's build flow, modify the `makefile` under the project folder. In this example, it is at `<sdk_root>/mcu/project/ab158x_evk/apps/earbuds_ref_design/GCC/Makefile`.

There is a section in the XXXX/module.mk makefile that defines the rules. This section also defines the modules required by the project when creating an image file. Add a new line into the section to include the module in the project.

Include your module's `module.mk` path in the `makefile` as shown below.

```
…
################################################################################
######
#
# SDK source files
#
################################################################################
######
#include cjson
include $(SOURCE_DIR)/middleware/third_party/cjson/module.mk

#include xml
include $(SOURCE_DIR)/middleware/third_party/xml/module.mk
#include mymodule
include $(SOURCE_DIR)/middleware/third_party/mymodule/module.mk
…
```

When the module is successfully built, the object and the dependency files of the added module are under <sdk_root>/mcu/out/<board>/<project>/obj/middleware/third_party/mymodule folder. In this example the file path is <sdk_root>/mcu/out/<board>/<project>/obj/middleware/third_party/mymodule/hello_world.o and <sdk_root>/mcu/out/<board>/<project>/obj/middleware/third_party/mymodule/hello_world.d.

You have now successfully added a module to an existing project. The next section shows how to create a project.

© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

7. Creating a Project

This section provides detailed information about creating your own project from an existing project. The following example shows how to create a new project named `my_project` on AB158x EVK using MCU earbuds_`ref_design` project as a reference.

7.1. Using an existing project

Apply an existing project as a reference design for your own project development.

Copy the folder `<sdk_root>/mcu/project/ab158x_evk/apps/earbuds_ref_design` to a new directory `<sdk_root>/mcu/project/ab158x_evk/apps/` and rename `earbuds_ref_design` to the new project name `my_project`.

7.2. Removing a module

When you use an existing project to create a new project, the new project keeps the same features. You can remove the modules to have a clean start for your project development.

To remove a module:

1) Open the project `makefile` from

   `<sdk_root>/mcu/project/ab158x_evk/apps/my_project/GCC/Makefile.`

2) Locate the module include list of the project and remove any unwanted modules by removing or commenting out the corresponding statements.

```
############################################################
...
# Bluetooth module
include $(SOURCE_DIR)/middleware/airoha/bluetooth/module.mk

# BT callback manager
include $(SOURCE_DIR)/middleware/airoha/bt_callback_manager/module.mk

# BT connection manager
include $(SOURCE_DIR)/middleware/airoha/bt_connection_manager/module.mk
...
```

7.3. User-defined source and header files

User defined project source and header files must be put under the `src` and the `inc` folder as shown in Figure 6.

© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

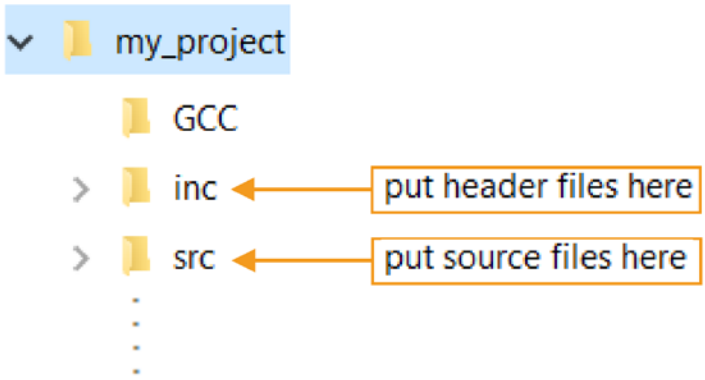


Figure 6. Project source and header files under the project folder

To compile the added source code, simply add the .c source files to variable "C_FILES" and the header search path to variable "CFLAGS" in the project makefile as shown below. The CXX_FILES and CXXFLAGS variables provide support compiling the source files (.cpp) of the module.

In the current makefile, there are two intermediate definitions, i.e. “APP_FILES” and “SYS_FILES”. Both of them are added in C_FILES. The line “include $(SOURCE_DIR)/$(APP_PATH_SRC)/apps/module.mk” in the makefile includes the C files in folder <my_projet>/src/apps

<sdk_root>/mcu/project/ab158x_evk/apps/my_project/GCC/Makefile

```
...
APP_FILES       += $(APP_PATH_SRC)/main.c
APP_FILES       += $(APP_PATH)/GCC/syscalls.c
APP_FILES       += $(APP_PATH_SRC)/regions_init.c
...
SYS_FILES       += $(APP_PATH_SRC)/system_ab158x.c
...
CXX_FILES       += ...
...
C_FILES         += $(APP_FILES) $(SYS_FILES)
...
```


© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

7.4. Testing and verification

When the MCU project is successfully built, the final image file is in the `<sdk_root>/mcu/out/<board>/<project>/` folder.
In this example, it is `<sdk_root>/mcu/out/ab158x_evk/my_project/`.
The object and the dependency files of your project are under `<sdk_root>/mcu/out/<board>/<project>/obj/project/<board>/apps/<project>/src/` folder. In this example, they are under `<sdk_root>/mcu/out/ab158x_evk/my_project/obj/project/ab158x_evk/apps/my_project/src/`.
Figure 7 shows the location of the image file, object and dependency files when the example project is built.

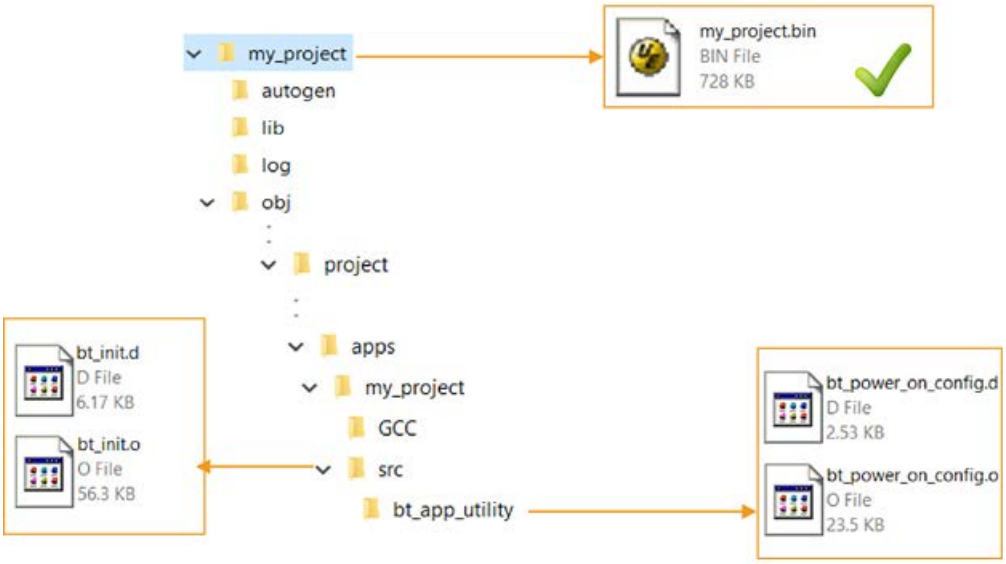


Figure 7. Location of the generated files when a project is built

7.5. Troubleshooting

When a build process fails as shown below, the error messages are written to the `err.log` file under `<sdk_root>/out/<board>/<project>/log/` folder. Please refer to the `err.log` file for more information.

```
…
TOTAL BUILD: FAIL
```

7.6. Configuring multi-processor products

Some Airoha products, such as AB1585, may have embedded digital signal processors as well as an MCU processor. Those processors have independent project files and build processes. We have integrated the build processes of the processors into one build.sh. You must configure the build mapping relation of each processor in advance to make the build.sh work correctly.

Section 7.7 Creating a new build target in mapping_proj.cfg shows how to add a new build target for multi-processor projects.

© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

NOTE: The multi-processors build script for AB158x series is in <sdk_root>/build.sh. It can build both the MCU and DSP projects.

7.7. Creating a new build target in mapping_proj.cfg

To create a new build target, you must edit the config file at <sdk_root>/mcu/tools/scripts/build/co_build_script/mapping_proj.cfg.

Add a new shell index array at the end of mapping_proj.cfg. The name of the array must be map__**<target _board>__<target_project>**. The <target_board> and <target_project> should be the expected first and second parameters while executing build.sh. The value of the array elements define the mapping relation of physical board, project and feature mapping of each processor. The array element definition is as follows:

0: borad_folder - folder name of board folder (under <sdk_root>/<processor>/project/)

1: MCU_project_folder – folder name of MCU project (under <sdk_root>/MCU/project/<board_folder>/)

2: dsp0_project_folder – folder name of dep0 project (under <sdk_root>/dsp0/project/<board_folder>/)

3: dsp1_project_folder – folder name of dep1 project (under <sdk_root>/dsp1/project/<board_folder>/)

4: MCU_project_feature_mk – Make file name of feautre definition for MCU project

5: dsp0_project_feature_mk – Make file name of feautre definition for dsp0 project

6: dsp1_project_feature_mk – Make file name of feautre definition for dsp1 project

Example:

```
map__ab1585_evk__earbuds_ref_design=( \
[0]="ab158x_evk" \
[1]="earbuds_ref_design" \
[2]="dsp0_headset_ref_design" \
[4]="feature_ab1585_evk.mk" \
[5]="feature_ab1585_evk.mk" \
)
```

To build the newly added target project, run the following command:

```
cd <sdk_root>
./build.sh <target_board> <target_project>
```

For example, run the following command under <sdk_root>:

```
./build.sh my_board my_project
```

This command causes the following command to run in both the MCU and DSP part.

For the MCU part:

```
cd <sdk_root>/mcu/
```


© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

```
./build.sh ab158x earbuds_ref_design –f=feature_ab1585_evk.mk
```

For the DSP part:

```
cd <sdk_root>/dsp/
  ./build.sh ab158x dsp0_headset_ref_design
-f=feature_ab1585_evk.mk
```


© 2016 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.