Airoha IoT SDK DSP Get Started Guide

Version: 2.11
Release date: 30 May 2022

Document revision history

Revision	Date	Description
1.0	20 September 2018	Initial version
1.1	12 December 2018	• Added support for Ubuntu build environment
1.2	8 March 2019	• Update 2.4. Building a project using the SDK section
1.3	26 April 2019	• Update 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 Aug 2019	• Added note for Ubuntu Linux environment setup
2.3	24 Sep 2019	• Added note for environment setup
2.4	15 Nov 2019	• Revised for using install.sh to automatically set up the environment
2.5	06 Dec 2019	• Revised content for AM255x
2.6	16 Jun 2020	• Revise supported Linux versions
2.7	30 Jun 2020	• Added support for AB1565
2.8	24 Jul 2020	• Added support for AB1568
2.9	20 Jan 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

Table of contents

Lists of tables and figures

1. Overview

The Airoha IoT SDK provides the software and tools for your application development on AB155x/AM255x/AB1565 /AB1568/AB1585/AB1588 EVK. The platform supports hardware abstraction layers, peripheral drivers, and FreeRTOS.

This guide provides instructions on how to use the SDK and its supported features.

In this guide, all supported chips use the same installation flow and build method as AB155x. The following examples use AB155x as a reference.

1.1. Platform architecture

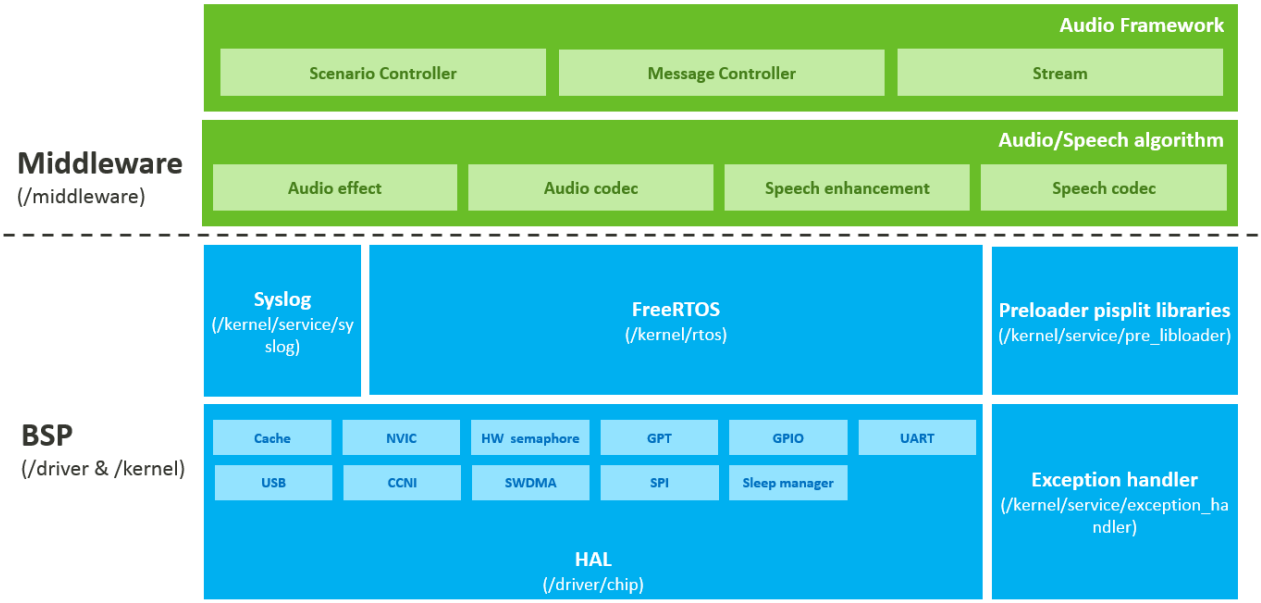

Figure 1. Architecture layout of the platform of AB155x/AM255x

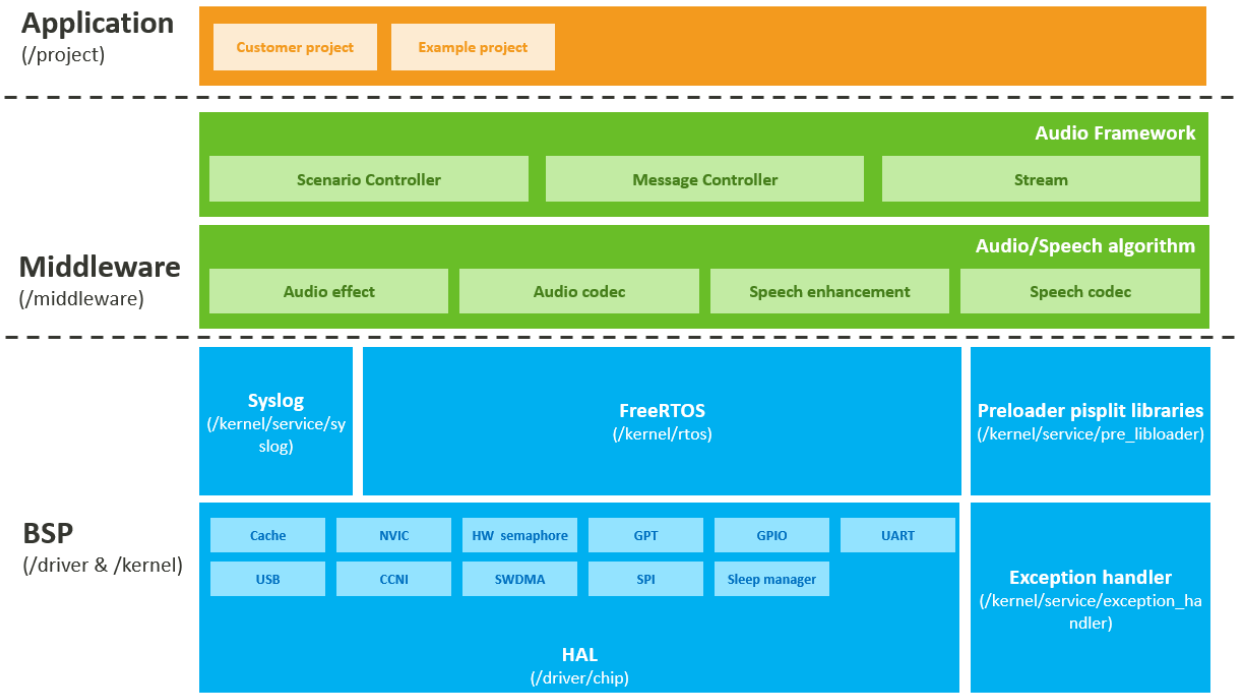

Figure 2. Architecture layout of the platform of AB1565/AB1568

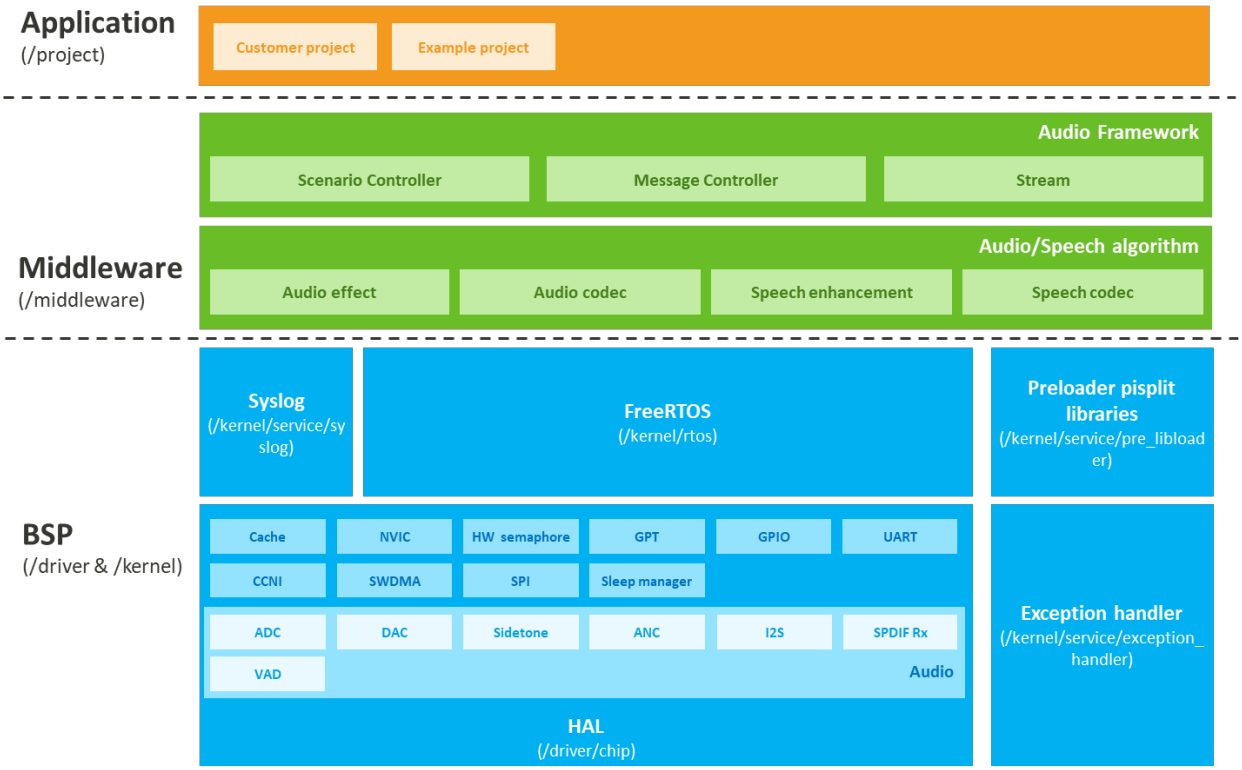

Figure 3. Architecture layout of the platform of AB1585/AB1588

A brief description of the layers is provided below:

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

- In audio framework,
  - The scenario controller handles several kinds of scenarios, such as the eSCO scenario, A2DP scenario, and voice prompt scenario.
  - The message controller handles message transmission between DSP and other processors, such as CM4/CM33.
  - The stream module 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 audio/speech algorithm,
  - The audio codec module includes SBC/AAC decoders for the A2DP application.
  - The speech codec module includes CVSD/mSBC decoders and encoders for the eSCO application.
  - The audio effect module includes equalizer (EQ) and dynamic range control (DRC) for variable and customized audio effect.
  - The speech enhancement module includes noise reduction (NR), echo cancellation (EC), and noise dependent volume control (NDVC) to enhance the signal performance for the listener.

- **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 **Middleware** components and **Application**.
  - Syslog – This module implements system logging for development and debugging.
  - Pre_loader – This module implements a position-independent library which can be loaded and run at any address.

1.2. Folder structure

The SDK is delivered as a single package organized in a folder structure, as shown in Figure 4 and Figure 5.

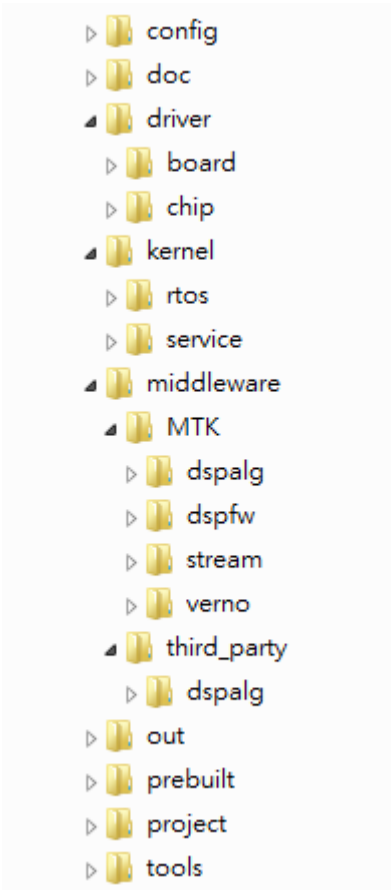

Figure 4. Folder structure for v2.x.x

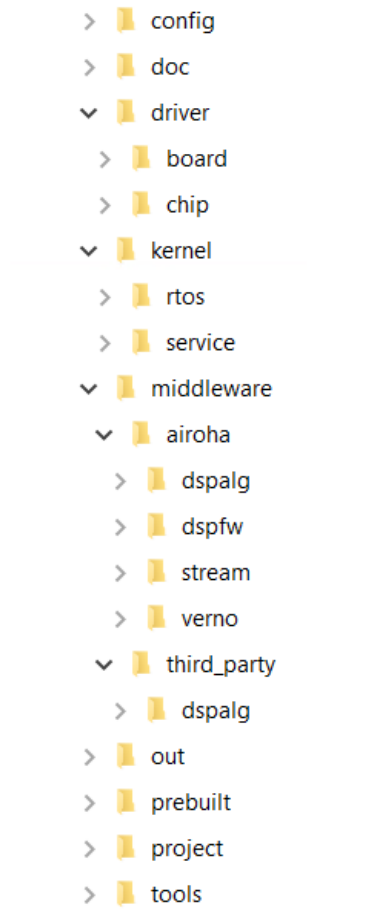

Figure 5. Folder structure for v3.x.x

This package contains the source and library files of the major components, 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
  - 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
  - rtos – Includes third party open source FreeRTOS.
  - service – Includes system service, such as the system log and exception handler.
- middleware
  - MTK or airoha – Includes middleware files created by Airoha.
    - dspalg – Includes codecs and audio/voice processing swipe and related driver wrapper.
    - dspfw – Includes the SW 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 codecs and audio/voice processing swipe and related driver wrapper.
- prebuilt – Contains 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 (e.g., projects with a single function showing how to use the drivers or other module features).

Example applications are under <sdk_root>\dsp\project\<evk>\apps\<project name> and <sdk_root>\dsp\project\<evk>\templates\<project name>. The following figure shows the folder structure.

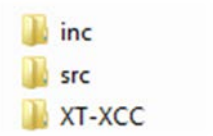

Figure 6. Project folder structure

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

You can apply the necessary reference applications to your development.

2. Setting up the DSP Development Tools

This section provides a guide to getting started with the Airoha IoT development platform for DSP and covers the following items:

- The supported environments for development
- Running the project on AB155x/AM255x/AB1565/AB1568/AB1585/AB1588 EVK
- Building the project using the SDK
- Creating your own project

2.1. Environment

A **specific version 18.10 64-bit or 18.04 LTS** of Ubuntu Linux environment is necessary. You must install the Xtensa toolchain to build a DSP project. The following software components are required:

- XtensaTools RG_2017_7
- XtensaTools RG_2019_12
- XtensaTools RI_2021_8
- Xtensa License server daemon

Complete the procedure shown in Airoha_IoT_SDK_for_BT_Audio_Get_Started_guide or Airoha_IoT_SDK_for_Smart_MCU_Get_Started_Guide, Chapter 2 to automatically set up the environment using a script.

Note:

The environment setup process herein has been integrated in a shell script, i.e. install.sh, under SDK all in one package. Please install Ubuntu first, the execute install.sh. The install.sh can complete the whole SDK installation and toolchain configuration automatically, including Xtensa development tools. For more information about install.sh, please refer to Airoha_IoT_SDK_for_BT_Audio_Get_Started_guide or Airoha_IoT_SDK_for_Smart_MCU_Get_Started_Guide, Airoha_IoT_SDK_for_BT_Auido_Build_Environment_Guide or Airoha_IoT_SDK_for_Smart_MCU_Build_Environment_Guide. We strongly recommend using install.sh to complete the installation process. If you want to understand more about the DSP environment setup, please refer to the appendix “Setup DSP toolchain manually”.

2.2. Building a project using the SDK

When building a project using the SDK, you need to use the script in <sdk_root>\dsp\build.sh.

For more information about the script, navigate to the dsp directory and execute the following command:

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

The outcome is 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
```

Run the following command to show all available boards and projects:

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

The available boards and projects are listed according to the related configuration files under the <sdk_root>/dsp /project/<board>/<apps or templates>/<project> folder. The console output is shown below.

```
================================================================
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
```

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

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

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

For example, to build a project on the ab155x_evk, run the following build command:

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

The standard output in the terminal window is 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 output files are put under the <sdk_root>\dsp\out\ab155x_evk\ dsp0_freertos_create_thread\ folder.

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

Clean the <sdk_root>\dsp\out folder.

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

Clean the <sdk_root>\dsp\out\<board> folder

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

Clean the <sdk_root>\dsp\out\<board>\<project> folder.

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

2.3. Downloading the project on EVK

To download the project on EVK:

1) Copy all 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, along with the DSP binary. You cannot import only the DSP binary.

Notes:

- The 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 of CM4 is mandatory. Update the binary names according to your project.
- You must delete the corresponding ROM section if there is no image for downloading.

```
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.

Please refer to <Airoha_IoT_SDK_Flash_Tool_Users_Guide.pdf> for more information and the guidelines 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 existed template project to the destination folder of the new project to be created.
2) Path of templates projects: <SDK version>\dsp\project\<board>\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 – Setup DSP Toolchain Manually

3.1. Install system components

- Install the necessary components in Ubuntu.

Please complete the following procedure to install the libraries that are necessary 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. Installation Xtensa toolchain

You must install the Xtensa toolchain package and DSP configuration package to build a DSP image. Please get the Airoha IoT SDK for BT audio and tools all-in-one package for the necessary 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	XtensaTools_RI_2021_8_linux.tgz
		common	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	Hifi5_MT2833_FDI_FINAL_linux_redist.tgz

3.2.1. Install Xtensa toolchain package

The <sdk_root> appears as follows. The “mcu” and “dsp” folders are available in <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”/

Note: You must first copy the following files to <sdk_root> folder.For AB155x/AM255x, AB155x_AM255x_DSP_Toolchain_Packages on MOL includes these files.

- XtensaTools_RG_2017_7_linux.tgz
- dsp0_core_winabi_xtensac_linux_redist.tgz
- dsp1_core_winabi_xtensac_linux_redist.tgz

For AB1565/AB1568, AB1565_AB1568_DSP_Toolchain_Packages on MOL includes these files.

- XtensaTools_RI_2021_8_linux.tgz (For SDK v2.x.x)
- AB1568_i64B_d32B_512K_linux_redist.tgz (For SDK v2.x.x)
- XtensaTools_RI_2021_8_linux.tgz (For SDK v3.x.x)
- AB1568_i64B_d32B_512K_linux_redist.tgz (For SDK v3.x.x)

For AB1585/AB1588, AB1585_AB1588_DSP_Toolchain_Packages on MOL includes these files.

- XtensaTools_RI_2021_8_linux.tgz
- Hifi5_MT2833_FDI_FINAL_linux_redist.tgz

First, execute the following 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, simply unpack the XtensaTools_RG_2017_7_linux.tgz.

Use the following command to extract the file in a Linux environment:

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

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

3.2.2. Install DSP configuration package

To install the DSP configuration package on a Linux platform, use the following command to **extract these two files in the xtensa folder**. The files are extracted to the <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 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
```

Change to the “dsp0_core_winabi_xtensac/”and “dsp1_core_winabi_xtensac/” directory respectively and run the installer program using following commands:

Note: **<sdk_root>** in the following commands **MUST** be the complete filepath".

```
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
```

When you successfully complete installing the DSP configuration packages, it show similar messages as below.

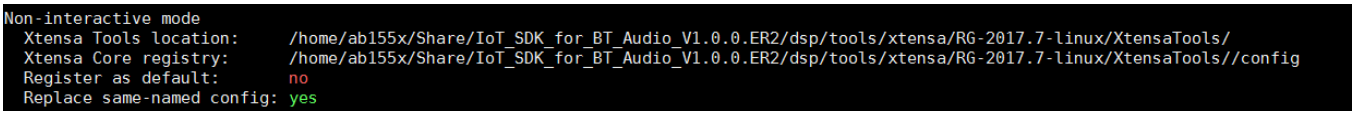

Figure 7. Install completed

3.2.3. Setting up Cadence license server for an Xtensa toolchain

Please make sure to read **Section 3.2.3.2. Configuring environment variables** to complete the final step for the setup.

To set up a license server, you must first extract CDNSLICREQ-xxx.zip (the license package is available from the Airoha eService Portal). You will see the license file: AB155x_AM255x_DSP_XXXXXXXXXXXX.lic.

Note:

- Visit the Airoha eService portal to apply for the Cadence license (in [Customer Portal] -> [Software Related] -> [License Request]). Please choose "floating license" as 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

Please follow below steps to set up license server on Linux.

1) **You need to go back to <sdk_root> first**, generate a folder (xt_server) for Xtensa toolchain Cadence license server and copy licserv_linux_x64_v11_13.tgz in it.

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

2) Untar the licserv_linux_x64_v11_13.tgz and copy AB155x_AM255x_DSP_XXXXXXXXXXXX.lic to the x64_lsb, to let license file and lmgrd file 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/
```

3) After execute these commands, you need to check your Linux Hostname.

Method 1. Execute the following command to get hostname:

```
cat /etc/hostname
```

The outcome is as follows:

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

Method 2. Execute the following command to get hosts information:

```
cat /etc/hosts
```

The outcome is as follows:

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

4) Open AB155x_AM255x_DSP_XXXXXXXXXXXX.lic license file to modify host name, tcp port and path of xtensad.

   a. Host name: <SERVERNAME>

   b. tcp port: <PORT>
      choose an un-used tcp port

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

o Original license file

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

o Example

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

5) Use the following command to launch license server.

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

o eg: ./lmgrd -c AB155x_AM255x_DSP_382C4A76B2CE.lic -l log.txt

6) Execute this command to check if license server launched.

```
ps aux | grep lmgrd
```

The output will appear as follows if it launches successfully:

```
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:

- Please make a note of what license server IP you use, and modify the value of environment variable (LM_LICENSE_FILE) in <sdk_root>\dsp\.rule.mk.

3.2.3.2. Configuring environment variables

To finish setting up the Cadence license server for an Xtensa toolchain, you must make changes to the three environment variables in <sdk_root>\dsp\.rule.mk, as shown below.

a. LM_LICENSE_FILE

b. PATH

c. XTENSA_SYSTEM

Note:

- **LM_LICENSE_FILE** variable stands for the TCP port and IP of your license server (format is <tcp_port>@<ip>)
- Use the following example to set the **PATH** and **XTENSA_SYSTEM** variables
- <sdk_root> **MUST** be full path

For example:

- 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

4.1. How to troubleshoot build error?

This section will shows how to fix common build error messages, please see below sections for detail.

4.1.1. Error message “License checkout failed”

When you occurred an error as following picture, please do these check lists to fix the problem. For more detail, you can refer **to section 3.2.3.1. Setting up the license server on Linux.**

Figure 8. License checkout failed

1) Check whether the MAC address in license file is identical with the Ethernet MAC address on your PC.
2) Check whether the hostname in license file is identical with hostname of your Ubuntu PC.
3) Check TCP port setting in license file is not used by other network services on your PC.
4) Make sure your license server manager (lmgrd) has been started. Execute this command to start the license server:

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

eg: ./lmgrd -c AB155x_AM255x_DSP_382C4A76B2CE.lic -l log.txt

5) Execute this command to check if license server launched.

```
ps aux | grep lmgrd
```

4.1.2. Error message “xt-xcc: not found”

When you occurred an error as following picture, to fix the problem please refer **to section 3.2.3.2. Configuring environment variables** to make sure your <sdk_root>\dsp\.rule.mk environment variables setting is correct.

Figure 9. xt-xcc: not found

You can reference to the following screenshot to double check environment variables inside the red frame.

```
################################################################################
# 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 10. Screenshot for .rule.mk