Airoha IoT SDK for BT Audio Get Started Guide

Version: 2.4

Release date: 23 March 2023

Airoha Proprietary and Confidential. © 2019 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	1 October 2019	Initial release; Created from Airoha_IoT_SDK_Get_Started_Guide v4.15
1.1	11 November 2019	Fixed a typo Renamed a reference document
1.2	16 June 2020	Updated supported Linux versions
1.3	23 June 2020	Added support for AB1565/AB1568
1.4	16 September 2020	Revised project folder name
1.5	3 December 2020	Updated Figure 3. AB1565/AB1568 platform architecture layout
1.6	7 December 2021	Added the description of the relationship between the toolchain version and the chip series.
1.7	18 January 2022	Added support for AB158x
1.8	23 February 2022	Updated building the project using the SDK section
1.9	22 April 2022	Updated the DSP tool chain version
2.0	25 April 2022	Updated section 1.3 to show the new common folder in the SDK
2.1	30 May 2022	Updated the 1565/1568 DSP tool chain version
2.2	14 October 2022	Updated the 155x/156x/158x DSP JTAG debugging tool
2.3	28 December 2022	Refined language and formatting
2.4	23 March 2023	Added support for AB157x

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document,
in whole or in part, is strictly prohibited.

Table of Contents

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document,
in whole or in part, is strictly prohibited.

© 2019 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

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document,
in whole or in part, is strictly prohibited.

1. Overview

The Airoha IoT Software Development Kit (SDK) for Bluetooth (BT) Audio provides the software and tools for application development with the evaluation kit (EVK). 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 get started guide shows how to use the SDK and its supported features in the GNU Compiler Collection (GCC) environment.

1.1. Architecture of the platform

The three-layer architecture of the platform includes the board support package (BSP), Middleware, and Application layers with the related components as shown in Figure 2, Figure 3, Figure 4 and Figure 4.

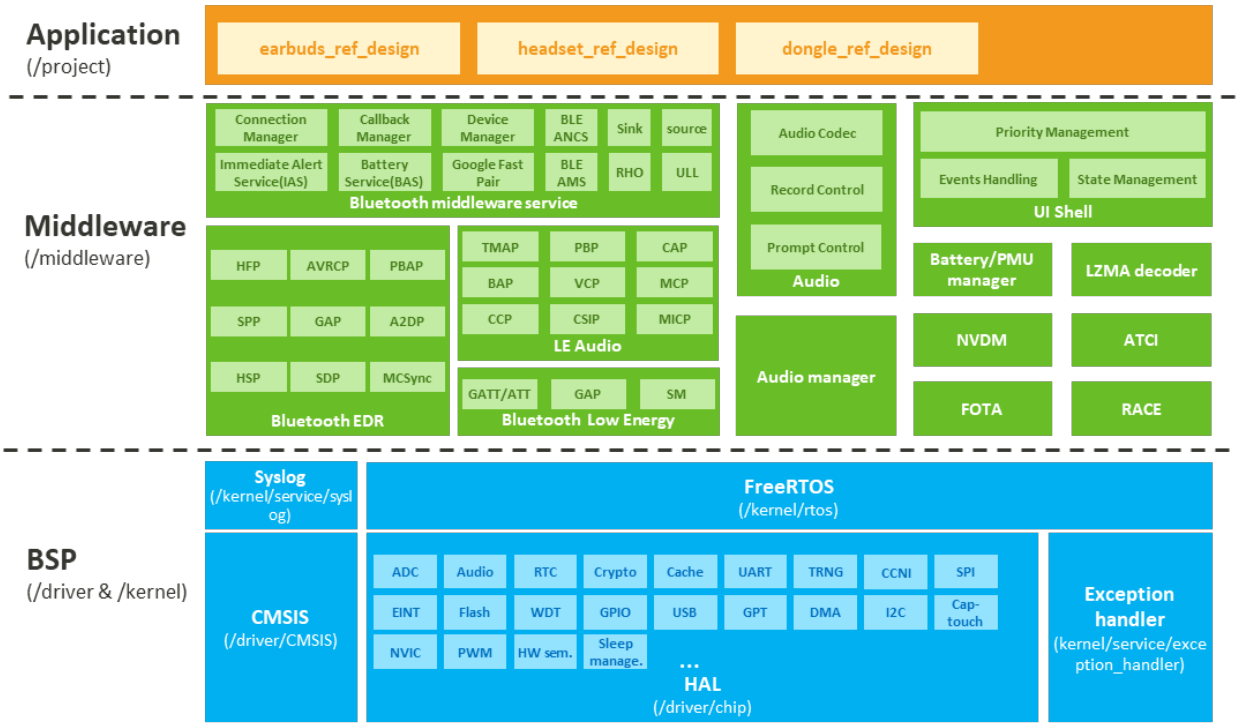

Figure 1. AB157x platform architecture layout

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

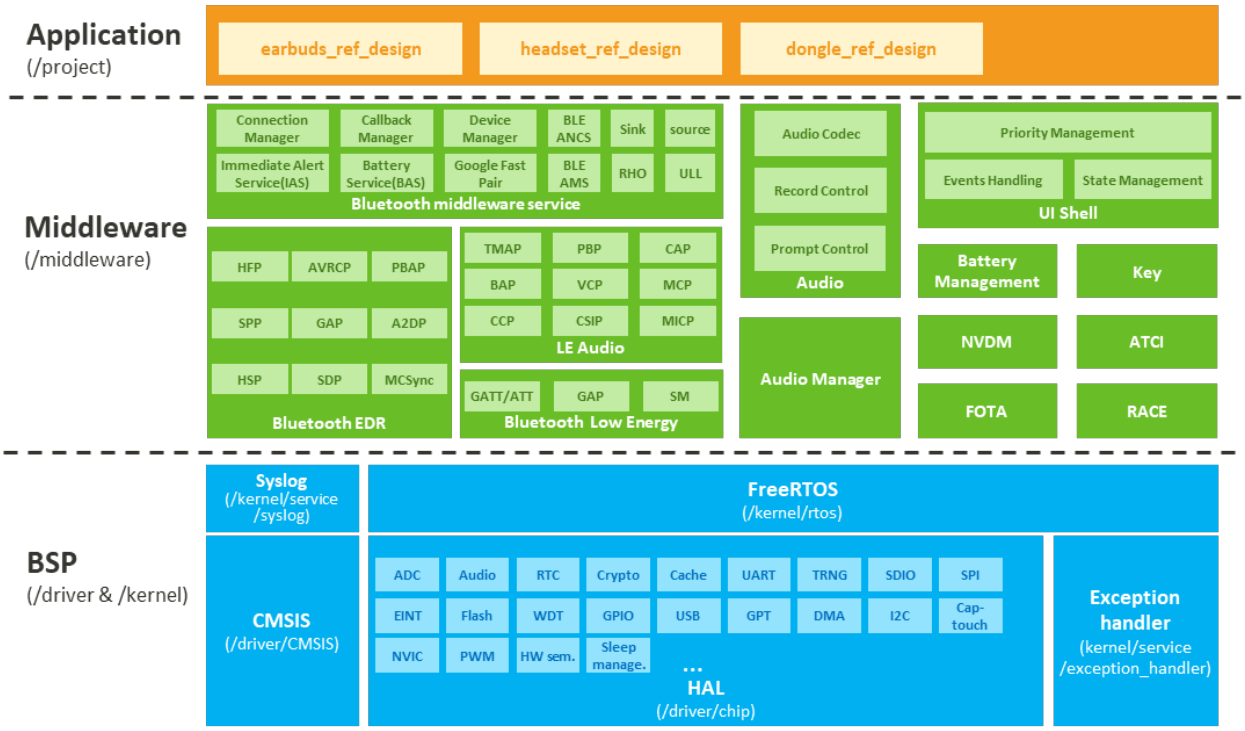

Figure 2. AB158x platform architecture layout

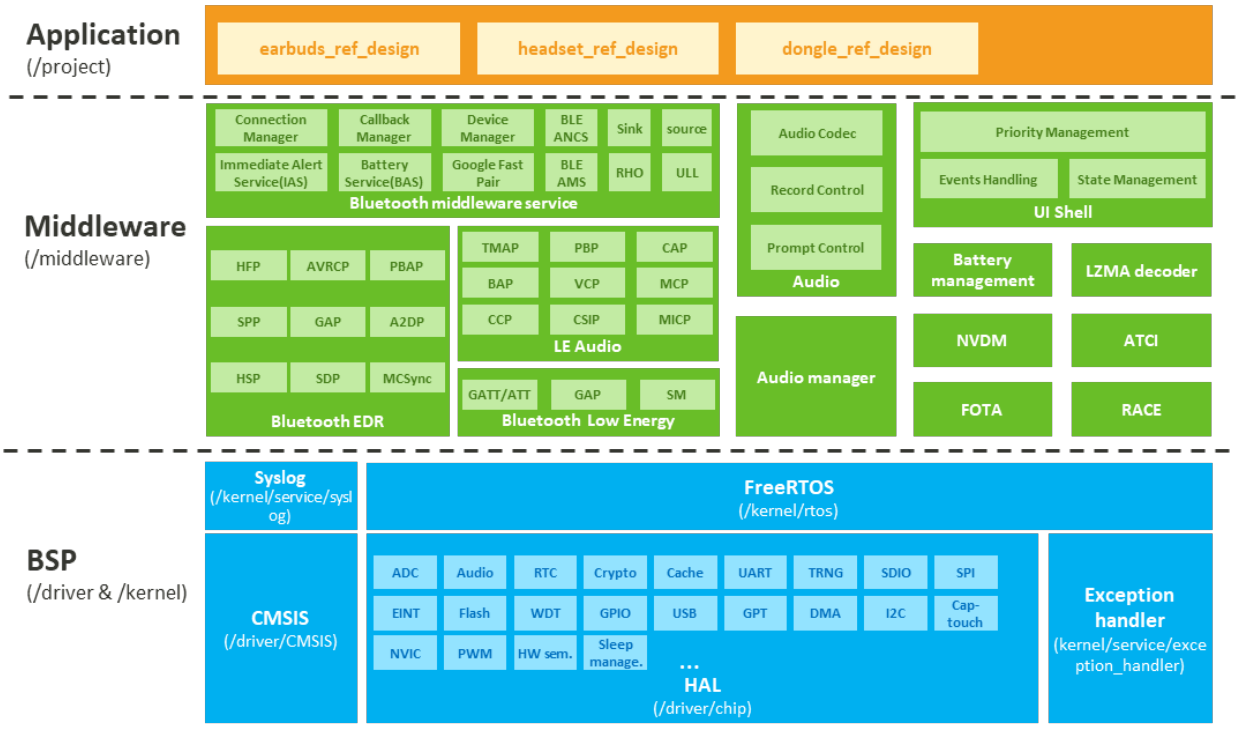

Figure 3. AB1565/AB1568 platform architecture layout

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

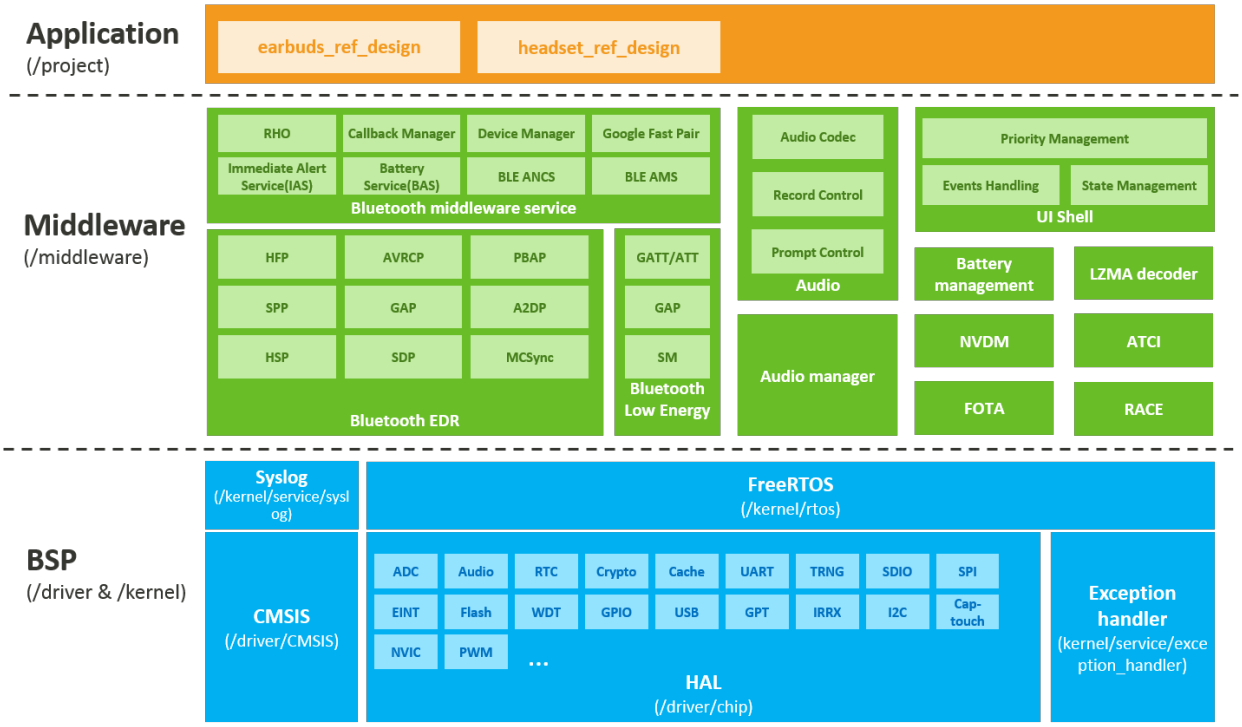

Figure 4. AB155x platform architecture layout

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.
- The hardware components at <sdk_root>\mcu\driver\board\component are used by the EVK (<sdk_root>\mcu\driver\board\xxx_evk). For example, the eint key driver is located under <sdk_root>\mcu\driver\board\component\eint_key folder and is available when you select the <chip> but the GPIO pins must be configured at <sdk_root>\mcu\driver\board\<chip>\eint_key. The source files must be included under the component folder.
- 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

- Bluetooth EDR/Bluetooth Low Energy – Provides stack and protocol-layer access profiles for data transfer and management control, such as Generic Access Profile (GAP), Serial Port Profile (SPP), Generic Attribute Profile (GATT), and Security Manager (SM).
- Bluetooth service – The implementation of Bluetooth common services.
- Audio – This module is for audio middleware implementation.
- Audio manager – This module is the Audio Manager control implementation including all main audio behavior management and most of the control for DSP.

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

- FOTA – Provides a mechanism to update the firmware.
- Battery management – Provides the charging flow control and precise information on battery.
- File system – Provides APIs to control data storage and retrieval in a file system.
- UI Shell – It is the UI framework which helps application developers to design and implement applications. Refer to *Airoha_IoT_SDK_UI_Framework_Developers_Guide* under `<sdk_root>/mcu/doc` for more information.
- Other features – Non-Volatile Data Management (NVDM), Race command, LZMA decoder, and other features that are dependent on HAL and FreeRTOS. The Airoha IoT SDK also supports AT command interface (ATCI) as an advanced feature.

Application

- Pre-configured projects using Middleware components, such as earbuds_ref_design and headset_ref_design.
- The application layer enables running projects that are based on Middleware, FreeRTOS, and HAL layers. These layers provide rich features for application development. For example, Middleware provides the Bluetooth and audio features and the OS provides the underlying real-time operating system.

Table 1 shows the HAL features that are supported on different chipsets.

To use the HAL features, enable the compile options of the corresponding modules:

1) Open the header file `hal_feature_config.h` which is located under the `inc` folder of each example project.
2) Modify and define the required compile options.
3) Add the related module header files (available via `<sdk_root>\mcu\driver\chip\inc`) to the project source files.

Table 1. HAL Features on Different Chipsets

Feature	Compile option
ADC	HAL_ADC_MODULE_ENABLED
AUDIO	HAL_AUDIO_MODULE_ENABLED
Cache	HAL_CACHE_MODULE_ENABLED
Crypto	HAL_AES_MODULE_ENABLED HAL_DES_MODULE_ENABLED HAL_MD5_MODULE_ENABLED HAL_SHA_MODULE_ENABLED
Clock	HAL_CLOCK_MODULE_ENABLED
CAP-TOUCH	HAL_CAPTOUCH_MODULE_ENABLED
EINT	HAL_EINT_MODULE_ENABLED
Flash	HAL_FLASH_MODULE_ENABLED
GDMA	HAL_GDMA_MODULE_ENABLED
GPIO	HAL_GPIO_MODULE_ENABLED

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

Feature	Compile option
GPT	HAL_GPT_MODULE_ENABLED
I2C Master	HAL_I2C_MASTER_MODULE_ENABLED
NVIC	HAL_NVIC_MODULE_ENABLED
PWM	HAL_PWM_MODULE_ENABLED
RTC	HAL_RTC_MODULE_ENABLED
SPI Master	HAL_SPI_MASTER_MODULE_ENABLED
SPI Slave	HAL_SPI_SLAVE_MODULE_ENABLED
SD	HAL_SD_MODULE_ENABLED
Sleep Manager	HAL_SLEEP_MANAGER_MODULE_ENABLED
TRNG	HAL_TRNG_MODULE_ENABLED
UART	HAL_UART_MODULE_ENABLED
USB	HAL_USB_MODULE_ENABLED
WDT	HAL_WDT_MODULE_ENABLED

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

Table 2 shows the middleware features that are supported on different chipsets. There is a `readme.txt` under the root directory of each middleware module. It provides a brief introduction and contains information about the module dependencies, feature options, notes, etc. To learn more about the usage of the middleware modules, refer to the `readme.txt` file under each of the module paths.

You must use `middleware\MTK` as <MIDDLEWARE_PROPRIETARY> for SDK v2.x.x and use `middleware\airoha` as <MIDDLEWARE_PROPRIETARY> for SDK v3.x.x.

Table 2. Middleware Features on Different Chipsets

Feature	Module path
Bluetooth stack	<MIDDLEWARE_PROPRIETARY>\bluetooth
Bluetooth profile	<MIDDLEWARE_PROPRIETARY>\bluetooth
Bluetooth Low Energy stack	<MIDDLEWARE_PROPRIETARY>\bluetooth
Bluetooth Low Energy profile	<MIDDLEWARE_PROPRIETARY>\bluetooth
Bluetooth AWS MCE	<MIDDLEWARE_PROPRIETARY>\bluetooth
Bluetooth AWS report	<MIDDLEWARE_PROPRIETARY>\bt_aws_mce_report
Bluetooth RHO	<MIDDLEWARE_PROPRIETARY>\bt_role_handover
Bluetooth sink service	<MIDDLEWARE_PROPRIETARY>\sink
Air pairing	<MIDDLEWARE_PROPRIETARY>\sink
Fast pairing	<MIDDLEWARE_PROPRIETARY>\bt_fast_pair
TLS	middleware\third_party\mbedtls
NVDM	<MIDDLEWARE_PROPRIETARY>\nvdm
Audio	<MIDDLEWARE_PROPRIETARY>\audio
ANC	<MIDDLEWARE_PROPRIETARY>\audio
Pass through	<MIDDLEWARE_PROPRIETARY>\audio
Mp3 codec	<MIDDLEWARE_PROPRIETARY>\audio
Prompt control	<MIDDLEWARE_PROPRIETARY>\audio
Record control	<MIDDLEWARE_PROPRIETARY>\audio
PEQ	<MIDDLEWARE_PROPRIETARY>\audio_manager
Audio management	<MIDDLEWARE_PROPRIETARY>\audio_manager
ATCI	<MIDDLEWARE_PROPRIETARY>\atci
Race Command	<MIDDLEWARE_PROPRIETARY>\race
Read-only file system	<MIDDLEWARE_PROPRIETARY>\rofs
File system	middleware\third_party\fatfs
Battery management	<MIDDLEWARE_PROPRIETARY>\battery_management
FOTA	<MIDDLEWARE_PROPRIETARY>\fota
LZMA decoder	middleware\third_party\lzma_decoder
UI framework	middleware\third_party\ui_shell

Note: the file system does not work without SD/eMMC.

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

1.2. Supported key components

The platform offers various options for connectivity, such as Bluetooth, peripheral drivers, and other advanced components. This section introduces each of these components.

1.2.1. Bluetooth and Bluetooth Low Energy

Bluetooth with Enhanced Data Rate (EDR) and Bluetooth Low Energy (LE) are key features of the Airoha IoT SDK. Table 3 shows the details. The API reference guide and Bluetooth developer's guides (available via `<sdk_root>\mcu\doc`) provide a description of the SDK API and module. Refer to `<sdk_root>\mcu\<MIDDLEWARE_PROPRIETARY>\bluetooth\readme.txt` for more details about the Bluetooth module.

Table 3. Bluetooth/Bluetooth Low Energy Features

Item	Features
EDR-A2DP	• Advanced Audio Distribution Profile
EDR-AVRCP	• Audio/Video Remote Control Profile (CT:v1.3/TG:v1.0)
EDR-HFP/HSP	• Hands-Free Profile v1.7 or Headset Profile
EDR-PBAP	• Phone Book Access Profile (PBAP) o Defines the procedures and protocols for exchanging Phonebook objects between devices.
EDR-SPP	• Serial Port Profile
EDR-GAP	• Generic Access Profile
BLE-GAP	• Generic Access Profile
BLE-GATT/ATT	• Generic Attribute Profile
BLE-SMP	• Low Energy Security Manager Protocol
Multipoint Support	• Supports multipoint Bluetooth access in EDR. o Two HFP (HF) o Two A2DP (Sink) o Two AVRCP (CT) o Two SPP server/client • Supports multipoint Bluetooth access in Bluetooth Low Energy. o Four Bluetooth Low Energy links.

1.2.2. FOTA

Table 4 provides a detailed list of FOTA features. The Airoha IoT SDK API Reference Manual and Airoha IoT SDK Firmware Update Developer's Guide (available via `<sdk_root>\mcu\doc` provide descriptions of the API and module. Refer to `<sdk_root>\mcu\<MIDDLEWARE_PROPRIETARY>\fota\readme.txt`. for more details about including this module.

Table 4. FOTA Features

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

Item	Features
FOTA	• Complete binary update mechanism • Package data compression • Integrity check • Power loss protection • FOTA packaging tool

1.2.3. Peripheral drivers

Table 5 shows a detailed list of peripheral drivers. The APIs for the drivers are in the Airoha IoT SDK API Reference Manual under `<sdk_root>\mcu\doc`. To include the HAL module, include `<sdk_root>\mcu\driver\chip\<chip>\module.mk` in the project makefile for the EVK.

Table 5. Supported Peripheral Drivers

Item	Features
ADC	• ADC module
CACHE	• The maximum size of the cache is 32kB
EINT	• External interrupt controller • Processes the interrupt request from an external source or a peripheral device
Flash	• Supports execute in place (XIP) and programming flash by software • Default 4MB system in package (SiP) flash on the AB155x and AB1565/AB1568EVK, and AB1568 have 8MB SiP flash • Supports external flash up to 16MB on Airoha IoT SDK platform HDKs
GPIO	• GPIO mode (in or out) • Set Pull Up/Down for GPIO IN mode
GPT	• General Purpose Timer • Supports 32kHz and 1MHz clock sources, repeat and one-shot modes for timing events and delays in µs or ms
PWM	• Range is 256 duty cycles • 32kHz, 2MHz, XTAL clock for PWM frequency reference
UART	• Three full set (TX/RX) UART support on the EVK • Baud rate of up to 3000000
I2C Master	• Two I2C interfaces • Supports 50/100/200/400kHz transmission rate
I2S Master	• I2S master is capable of servicing an external codec component • Supports 8/11.025/12/16/22.05/24/32/44.1/48 kHz audio sampling rates in stereo mode

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

Item	Features
ISINK	• Current sink • Adjustable backlight current
MPU	• Memory Protection Unit
IrDA	• RX (NEC, RC5, RC6, SIRC,RCMM)
WDT	• Supports hardware and software watchdog • Supports system reset
I2S-Slave	• Supports sample rates of 8, 12, 16, 24, 32, and 48 Kbits • Supports mono and stereo mode
SPI-Master	• Serial Peripheral Interface
RTC	• Real-Time Clock
GDMA	• General Purpose DMA
Security	• SHA1, SHA2 (256, 384, 512), and AES
TRNG	• Truly Random Number Generator • Generates a 32-bit random number
Charger	• Supports single-cell Li-Ion battery charging
Cap-touch	• Cap-touch key detect • Supports Cap-touch for key event

1.2.4. Battery management

Table 6 shows the battery management features. The battery management APIs are in the API Reference Manual under `<sdk_root>\mcu\doc`. Refer to `<sdk_root>\mcu\<MIDDLEWARE_PROPRIETARY>\battery_management\readme.txt` for more information about including this module.

Table 6. Battery Management Features

Item	Features
Battery management	• Charging flow control mechanism • Algorithm for measuring the battery capacity • Precise information about the battery, including temperature and battery level

1.2.5. Advanced features and components

Table 7 shows the advanced features and components that are included in the platform.

Table 7. Advanced Features and Components

Item	Features
ATCI	• AT command parser
File system	• Windows compatible • Platform independent

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

Item	Features
	• Very small footprint for code and work area • Multiple volumes • Multiple ANSI/OEM code pages including DBCS • Support for long file names in ANSI/OEM or Unicode • FreeRTOS support for multitasking • Multiple sector size with support for up to 4 kB • Read-only, minimized API, I/O buffer, and more

1.3. Folder structure

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

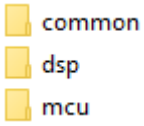

Figure 5. BT-Audio folder structure

The SDK package contains three folders: common; dsp; and mcu. A brief description is as follows:

- `common` – Includes the common declares of the subsequent modules which are referenced by both the mcu side and the dsp side. Refer to description of the MCU-side folder structure for more information about the common declare list.

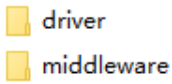

Figure 6. Common folder structure of bt-audio

- `dsp` – Includes the source and library files of the major components, the build configuration, related tools and the documentation. Refer to Airoha_IoT_SDK_DSP_Get_Started_Guide.pdf under <sdk_root>\dsp\doc for more information.
- `mcu` – Includes the source and library files of the major components, the build configuration, related tools and the documentation. This guide provides more information.

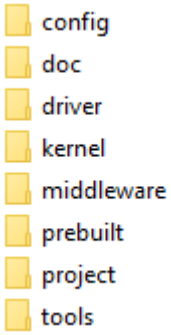

Figure 7. MCU folder structure of bt-audio

The mcu folder contains the source and library files for the major components, the build configuration, related, tools and the documentation as shown in Figure 7. A brief description on the layout of these files is as follows.

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

- `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 files such as board drivers, peripheral and CMSIS-CORE interface drivers. Note that some declares which are shared by both the mcu side and dsp side are defined under the `common\driver` folder.
- `kernel` – Includes the underlying RTOS and system services for handling exception and logging errors.
- `middleware` – Includes software features for HAL and OS such as network and advanced features.
- `prebuilt` – Contains binary files, libraries, header files, makefiles, and other pre-built files.
- `project` – Includes a 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.

The main middleware components are in the middleware folder:

MTK or airoha

- `atci` – Provides the interface for a target communication using AT commands though UART.
- `audio` – This module is for audio middleware implementation. Some of the declares which are shared by both mcu side and dsp side are defined under the `common\middleware\airoha\audio` folder.
- `audio_fft` – This module is the Fast Fourier Transform (FFT) implementation for analyzing data.
- `audio_manager` – This module is the Audio Manager control implementation including all of the main audio behavior management and most of the control for DSP.
- `battery_management` – Includes battery monitor, charging flow control, and battery capacity algorithms.
- `ble_ancs` – This module is the ble_ancs middleware implementation. Apple Notification Center Service (ANCS) allows access to notifications generated on iOS devices by a Bluetooth low-energy. This can only be connected with iOS devices.
- `ble_bas` – This module is the Battery Service (BAS) implementation. BAS shows the Battery State and Battery Level to the peer device through a Bluetooth low-energy link. This module supports Battery Level notifications to the peer device and manages Battery Level related read events from the peer device.
- `ble_dis` – This module is the Device Information Service (DIS) implementation. DIS shows the manufacturer and/or vendor information about a device. A control point uses Bluetooth low-energy link to allow a peer device to cause the local device to immediately send an alert. This module manages all of the user's registered callbacks and notifies all users when an Alert Level related write event occurs from the peer device.
- `ble_ias` – This module is the Immediate Alert Service (IAS) implementation. IAS exposes a control point to allow a peer device to cause the local device to immediately alert by a Bluetooth low-energy link. This module manages all of the user's registered callbacks and notifies all users when an Alert Level related write event occurs from the peer device.
- `bluetooth` – Bluetooth/Bluetooth Low Energy provides three profiles: GAP; GATT; and SM. These profiles discover and connect Bluetooth devices and allow the secure transfer and control of data through a Bluetooth connection.

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

- `bluetooth_service` – This module is the Bluetooth common services implementation. It includes two services: the Device Manager Service, which manages bonded peer device's security information; and the GATT Server Service, which supports the GAP service and the characteristics configuration feature.
- `bluetooth_service` – This module is the Bluetooth common services implementation. It includes two services: one is Device Manager Service, which manages the bonded peer device's security information; and the GATT Server Service, which supports the GAP service and the characteristics configuration feature.
- `bt_air` – This module is the BT AIR Service implementation. It includes three modules: BLE AIR; SPP AIR; and AirUpdate. It helps the AIR application communicate with peer device via a low-energy link or Bluetooth SPP profile or AirUpdate Fixed channel. This module manages all of the user's registered callbacks and notifies all users when a RX data event occurs from the peer device.
- `bt_aws_mce_report` – This module manages all Bluetooth aws mce report users. The user can register and deregister their callback functions to complete sending or receiving app report information.
- `bt_callback_manager` – This module manages the callback function of the Bluetooth stack. The user can register and deregister the EDR/BLE callback function to handle different callback functions.
- `bt_connection_manager` – This module manages all Bluetooth role handover users. The user can register and deregister their callback functions to complete the role handover procedure.
- `bt_fast_pair` – This module provides the google fast pair 2.0 feature. The user can call the API or send the action provided by this module to the fast pair 2.0 feature.
- `bt_role_handover` – This module is to manage all Bluetooth role handover users. User can register and deregister their callback functions to complete the role handover procedure.
- `fota` – FOTA provides firmware update functionality.
- `key` – Airo key is a common upper layer for different types of keys, including captouch_key, eint_key, gsensor key, and powerkey. This module has a common interface and common event type.
- `mfi_coprocessor` – This module is for middleware MFI coprocessor porting layer implementation.
- `module_log` – This module is the module_log implementation.
- `nvdm` – NVDM is a type of memory mechanism that retains its content when the system powers off.
- `port_service` – This module is the port service implementation. It is based on different serial device ports and offers user with unified interface to use.
- `race_cmd` – This module is the Race command interface.
- `rofs` – This module is the ROFS (read-only file system) interface.
- `serial_nand` – This module is the specialized flash-disk management implementation for the serial NAND flash device.
- `serial_nor` – This module is for accessing the serial NOR Flash by the SPI interface.
- `sink` – This module is the sink service which integrates HFP, A2DP, AVRCP and PBAPC profiles. It works as a Bluetooth headset and supports the headset features, such as, answering or rejecting incoming call, getting contact name of the incoming call, playing and pausing music, moving to previous song and next song, and connection reestablishing when power on or link lost.
- `smart_charger` – This module is the smart charger case interface.

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

- `ui_shell` – It is the UI framework which helps application developers to design and implement applications. Refer to *Airoha_IoT_SDK_UI_Framework_Developers_Guide* under `<sdk_root>/mcu/doc` folder for more information.
- `usb` – This module is the USB class interface.

third_party

- `mbedtls` – Transport Layer Security (TLS) and Secure Sockets Layer (SSL) are cryptographic protocols designed to provide communications security over a computer network. mbed TLS is an open source implementation for developers to include cryptographic and SSL/TLS capabilities in embedded products with a minimal coding footprint.
- `fatfs` – FatFs is generic FAT file system for small embedded systems. It is used to control data storage and retrieval in a file system.
- `lzma_decoder` – LZMA is the default and general compression method used to perform lossless data compression. LZMA is also suitable for embedded applications because it provides fast decompression and a high compression ratio.
- `micro_ecc` – A small and fast ECDH and ECDSA implementation for 8-bit, 32-bit, and 64-bit processors.

1.4. Project source structure

The SDK provides a set of reference applications. For example, projects with a single function showing how to use drivers or other module features and others with complex functionality demonstrating how to use the middleware components.

Example applications are in the `<sdk_root>\mcu\project\<chip>\apps` folder. They all have the same folder structure, as shown in Figure 8.

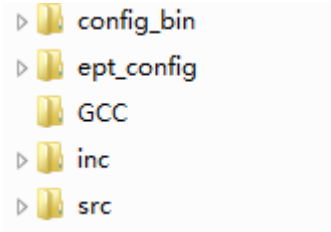

Figure 8. Project folder structure

- `config_bin` – The filesystem.bin and the nvkey.xml.
- `ept_config` – The *.ews files for setting GPIO of a board.
- GCC – GCC related project configuration files, such as a makefile.
- `inc` – Project header files.
- `src` – Project source files.
- `Readme.txt` – A brief introduction about project behavior and the required environment.

You can apply specific reference applications for your own development.

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

2. Getting Started with the Build Script

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

- Supported environments for development
- Configuring the EVK
- Building the project using the SDK
- Downloading and running the project via Microsoft Windows
- Debugging the project via Microsoft Windows
- Creating your own project
- Airoha IoT SDK chipsets for BT Audio product line
- Airoha IoT SDK for BT Audio with AB155x, AB1565, AB1568, and AB158x

2.1. Environment

You can use the SDK with Microsoft Windows 7, 8, and 10 and on Linux (e.g., Ubuntu 18.10 64-bit or Ubuntu 18.04 LTS/20.04 LTS/22.04 LTS). You must have compilers for ARM Cortex-M4/Cortex-M33 (gcc) and Cadence's Tensilica DSP HiFi Mini/HiFi 5 to build the project.

Download and extract the contents of the SDK all-in-one package on your local PC.

- Linux: IoT_SDK_For_BT_Audio_and_Linux_Tools_All_In_One
- Windows: IoT_SDK_For_BT_Audio_and_Windows_Tools_All_In_One

Refer to the instructions in the readme.txt file and run the installer script to install the SDK package.

install.sh installs the required system components (e.g., make, libc-i386, etc.), unpacks the SDK package and configures the toolchain. For most cases, install.sh can set up the complete environment. However, if you want to customize your environment, you can refer to Airoha_IoT_SDK_for_BT_Audio_Build_Environment_Guide under `<sdk_root>/mcu/doc` folder and Airoha_IoT_SDK_DSP_Get_Started_Guide under `<sdk_root>/dsp/doc` folder. The documents contain detailed instructions for the manual installation.

The installation process requires an internet connection to get the license for Cadence's Tensilica toolchain from the Airoha server. When the installation process is complete, the SDK can operate offline. If you cannot connect to the internet during the installation process, we provide an offline installation. Refer to the environment setup training video available via the Airoha eService documents for more information.

Note: The toolchain version used by different series of chips may also be different. Refer to the subsequent table for more specific information.

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

Table 8. Correspondence Between Toolchain Version and Chip Series

Chip series	Toolchain of MCU side	Toolchain of DSP side
AB155x	GCC 4.8.4	RG-2017.7 with Xplorer-7.0.7
AB156x (with SDK v2.x.x)	GCC 4.8.4	RG-2019.12 with Xplorer-8.0.9
AB156x (with SDK v3.x.x)	GCC 9.2.1	RI-2021.8 with Xplorer-9.0.18
AB158x		
AB157x		

2.2. Building the project using the SDK

Refer to Airoha_IoT_SDK_for_BT_Audio_Build_Environment_Guide.pdf under the `<sdk_root>/mcu/doc` for more information.

All you can change in SDK build flow are the feature options in feature*.mk and the project Makefile under `<sdk_root>/mcu/project/<chip>/apps/<project_name>/GCC` folder. Do not modified other .mk file, it may cause build flow broken.

Category	File naming	Description	Location	Attribute
Feature	feaure_*.mk	feature config for project	<sdk_root>/<mcu or dsp>/project/<chip>/apps/<project_name>/<GCC or XT-XCC>	user configuration
Chip	chip.mk	chip rule	<sdk_root>/<mcu or dsp>/config/chip/<chip>	read only
Board	module.mk	board setting	<sdk_root>/<mcu or dsp>/config/chip	read only
Module	module.mk	module makefile	under each module folder	read only
Project makefile	Makefile	main makefile for project build	<sdk_root>/<mcu or dsp>/project/<chip>/apps/<project_name>/<GCC or XT-XCC>	user configuration

2.3. Developing on AB157x EVK

The following sections show the configure, install, flash, run, and debugging processes when working with the EVK.

2.3.1. Configuring the AB157x EVK

The AB157x EVK has two separate major boards. One board is the EVK main board. The other board is an adaptor board for each chipset. There are four types of adaptors available: AB1577S, AB1571, AB1571D, and AB1577AM. Figure 15 shows the front view of the AB157x EVK with the AB1577S adaptor board.

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

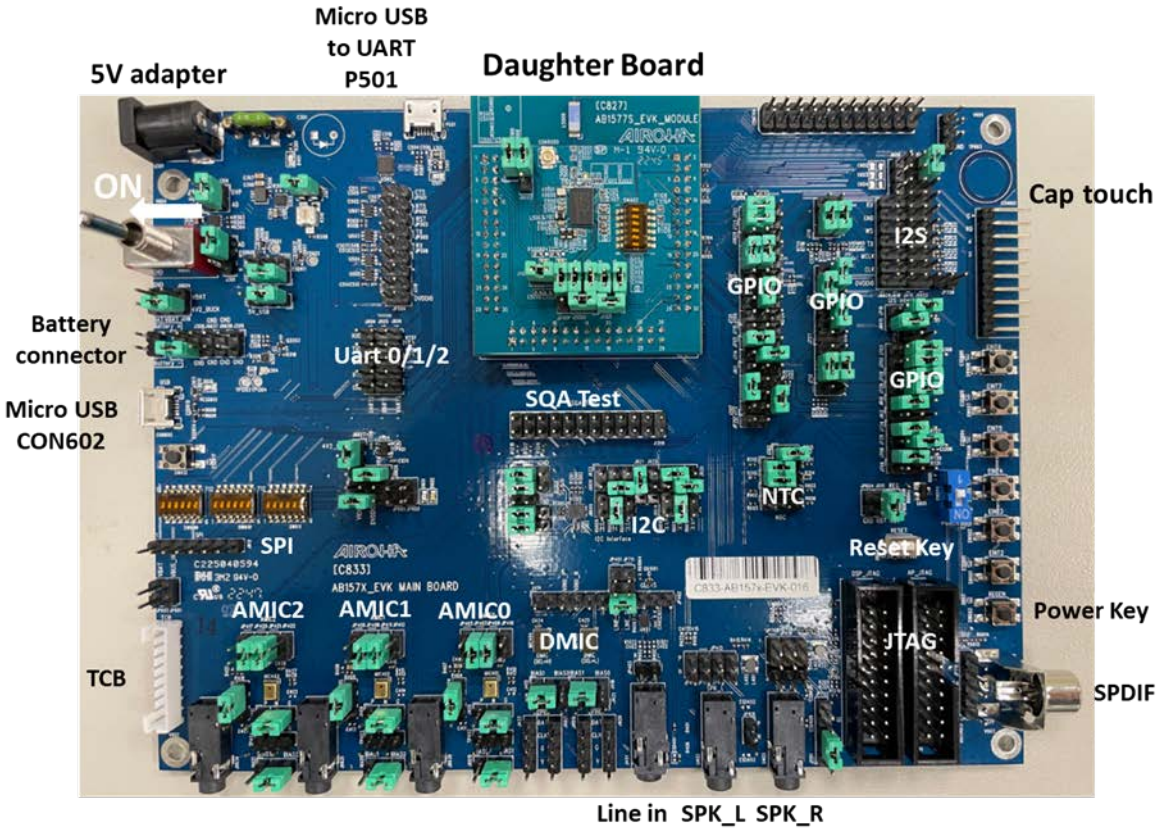

Figure 9. Front view of the AB158x EVK

The EVK can be powered by either USB VBUS or the 5V DC jack. Set the jumpers J301 1-2 to on for a 5V DC jack.

There is a micro-USB port on the left-side of the EVK. It can also be used as the firmware download port. Set jumpers J301 2-3 to on, jumpers J1323 2-3 to on and jumpers J313 1-2 to on. The VBUS power source is from USB VBUS.

AB157x has reserved two JTAG debugging interfaces. AP_JTAG is used for embedded CM33 and DSP_JTAG is used for the embedded DSP. You can directly connect JTAG JIG with these connectors for debugging.

2.3.2. Installing AB157x Flash Tool for AB157x EVK

Airoha Flash Tool is a flexible device flashing tool for application development on the AB157x EVK. It is under `<sdk_root>/mcu/tools/pc_tool/IOT_Flash_Tool`.

You can run the AIROHA_`Flash_Tool.exe` inside the folder to start the Flash Tool.

2.3.3. Installing the AB157x EVK drivers on Microsoft Windows

This section shows how to install the AB157x EVK USB drivers on a Microsoft Windows PC.

To install the MediaTek USB Port driver on the EVK in Microsoft Windows:

1) Install the MediaTek USB Port driver from MS_USB_ComPort_Driver/v3.16.46.1 in the IOT_Flash_Tool folder.

2) Run InstallDriver.exe to install the driver.

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

3) Use a micro-USB cable to connect the AB157x EVK to your computer.

2.3.4. Flashing the image to AB157x EVK

You must use a pre-built project file (.cfg) or build your own project to get one (refer to Section *2.2*) before using the Airoha Flash Tool.

To download the firmware to the target device via UART:

1) Plug in the **UART** cable.
2) Click **Download** button on the left panel of the main GUI.
3) Select right **UART** from the **COM Port** drop down menu list.
4) Click **Open** button to provide the configuration file.
5) Click **Start** button to start downloading.
6) Power off then power on the target or press reboot button within target, if reboot ok, then the process will start automatically.

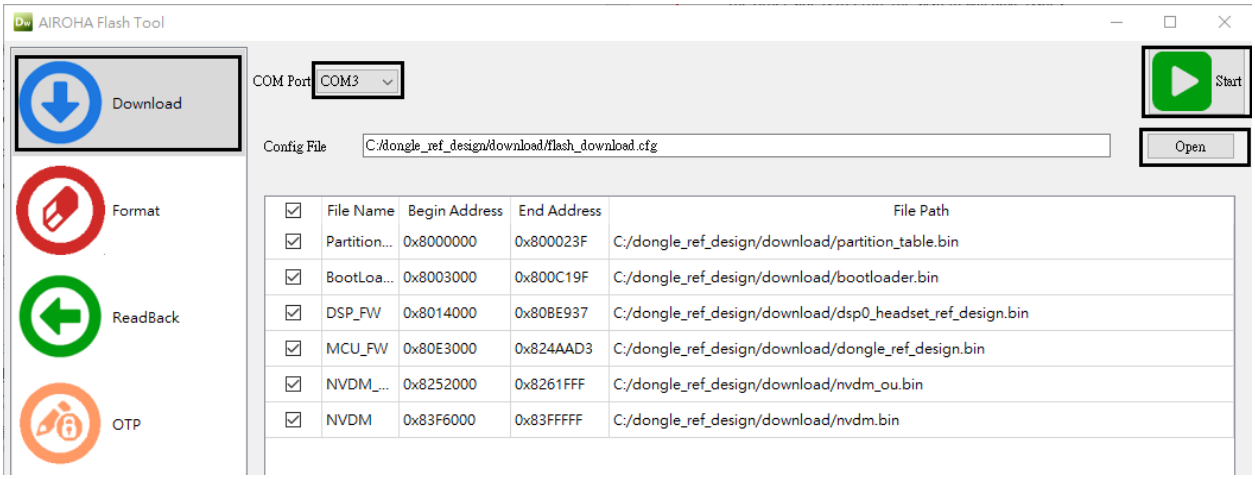

Figure 10 Download the firmware to a target device using UART connection

2.3.5. Running the project on AB157x EVK

To use the logging tool:

1) Launch Logging tool.
2) Click the **log binary file** button as shown in Figure 17.
3) Set the **Baud Rate** as shown in Figure 18.
4) Select your serial port on the dropdown list and click the **Connect** button as shown in Figure 19.
5) Click the **Wireshark** button as shown in Figure 20.
6) Reset the target. The log is shown in the log window.

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

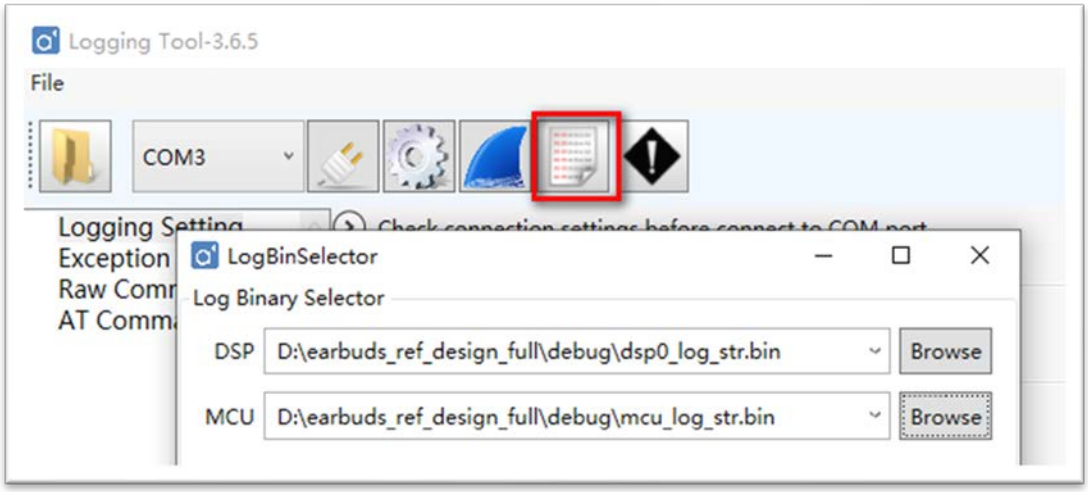

Figure 11. Selecting log binary

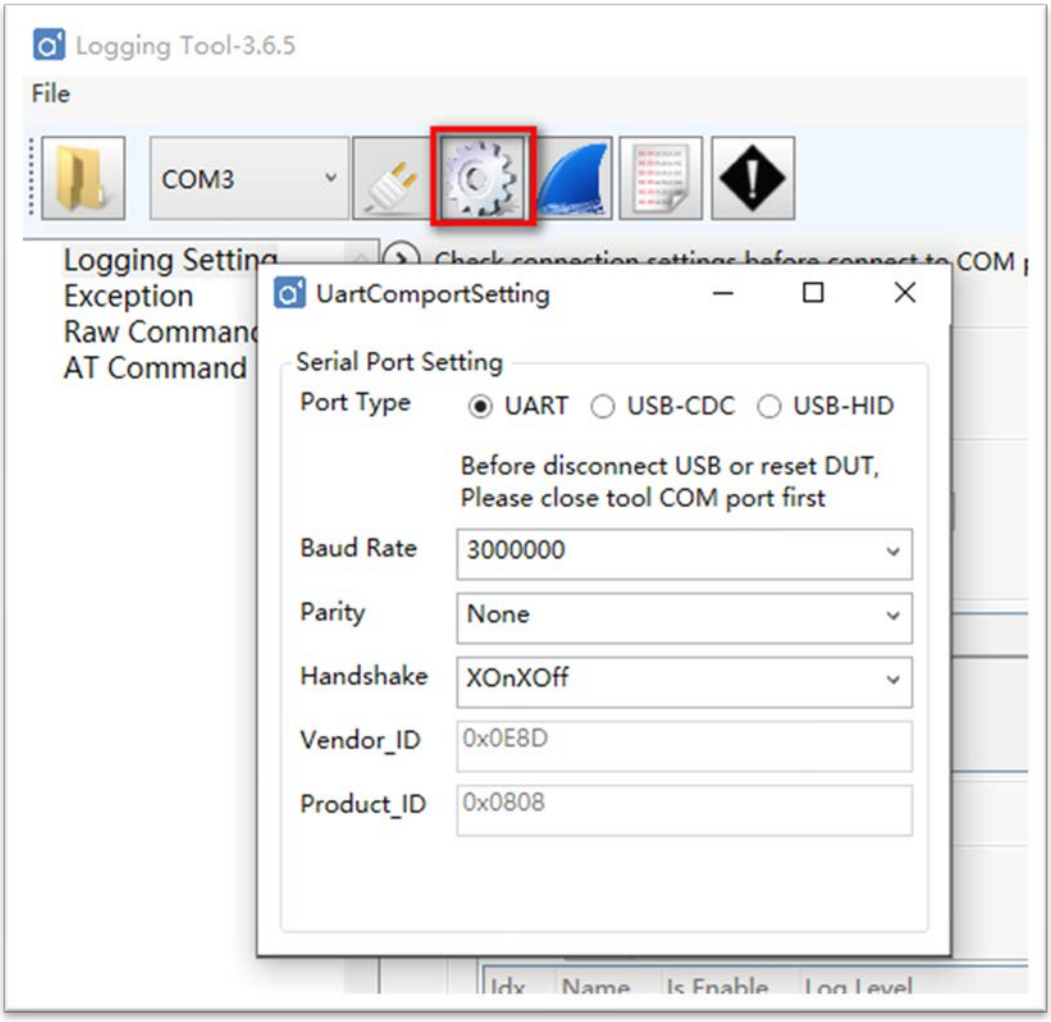

Figure 12. Serial Port Configuration window

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

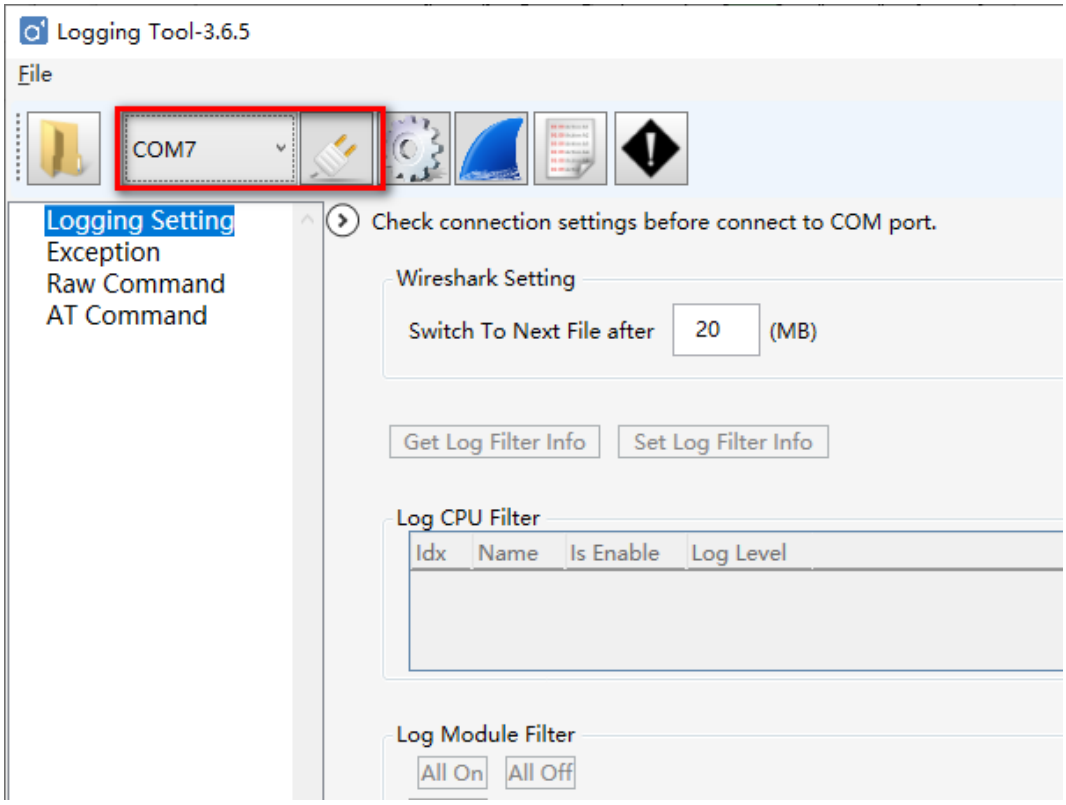

Figure 13. Serial Port Connect button

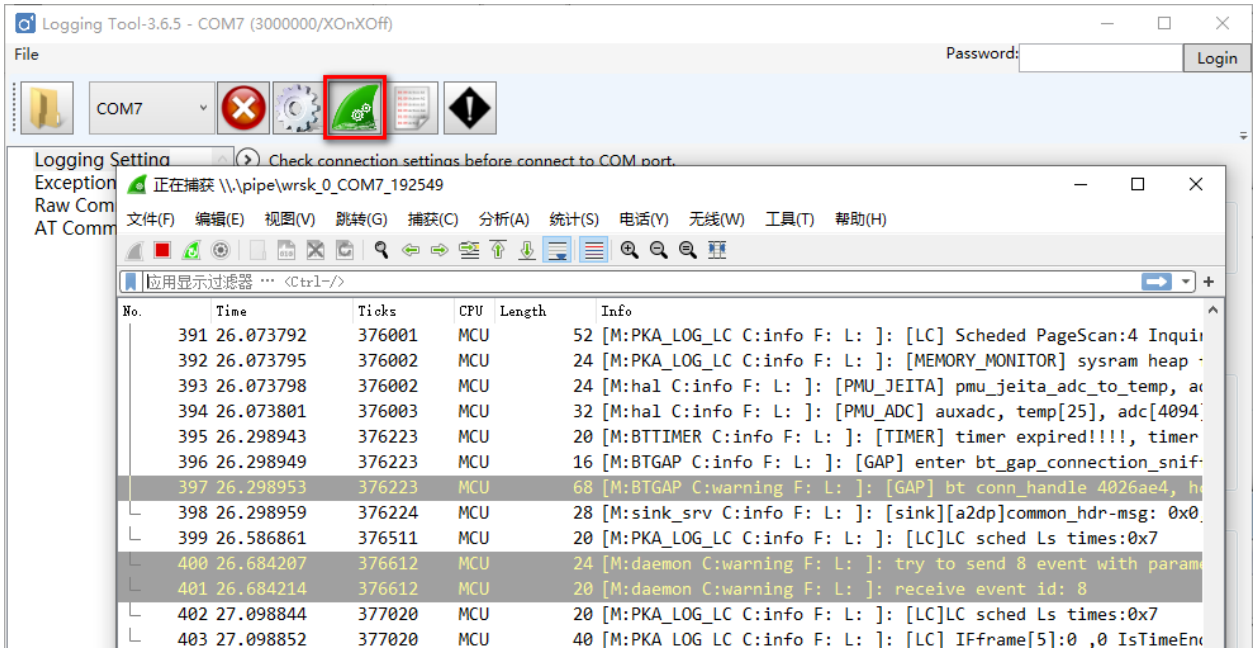

Figure 14. Start Wireshark button

2.3.6. Debugging MCU with the AB157x EVK via Microsoft Windows

The process for debugging on AB157x is the same as 155x. Refer to Section *2.6.6* for more information.

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

2.3.7. Debugging DSP with the AB157x EVK on Microsoft Windows

This section shows the tools and versions for debugging DSP.

- xplorer-9.0.18 and xt-ocd-14.0.8 are the 157x dsp debugging tools.
- You can download the tools from here. You must have an account to download the tools.
- You must have a license to use xplorer.

Although we support JTAG debugging, we do not recommend customers to debug using this method. The reasons are as follows:

- Our syslog and memory dump are better than JTAG debug, and easier to debug.
- Our system is a collaboration between MCU and DSP. If JTAG is connected to one side and this side stop while and the other side is freerun, the system will have issues.

2.4. Developing on AB158x EVK

The following sections show the configure, install, flash, run, and debugging processes when working with the EVK.

2.4.1. Configuring the AB158x EVK

The AB158x EVK has two separate major boards. One board is the EVK main board. The other board is an adaptor board for each chipset. There are two types of adaptors available: AB1585 and AB1588. Figure 15 shows the front view of the AB158x EVK with the AB1588 adaptor board.

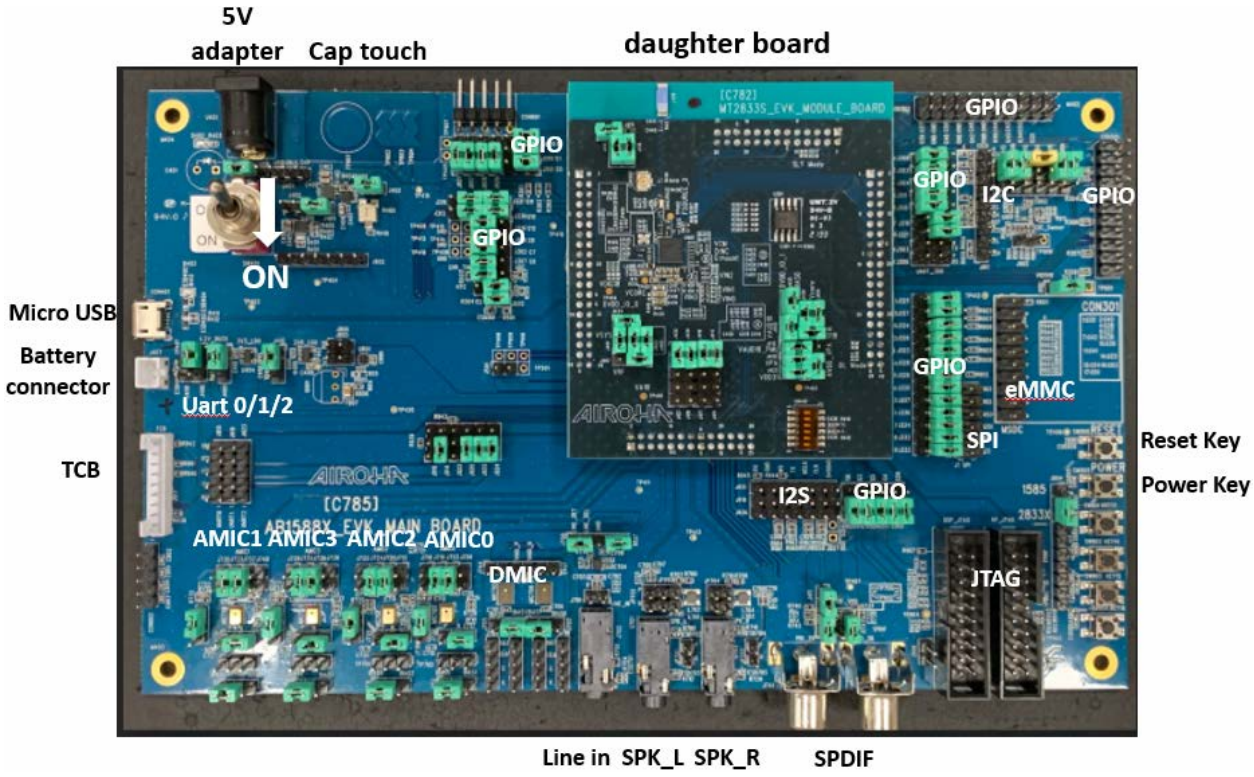

Figure 15. Front view of the AB158x EVK

The EVK can be powered by either USB VBUS or the 5V DC jack. Set the jumpers J403 1-2 to on for a 5V DC jack.

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

There is a micro-USB port on the left-side of the EVK. It can also be used as the firmware download port. Set jumpers J403 2-3 to on and jumpers J401 1-2 to on. The VBUS power source is from USB VBUS.

AB158x has reserved two JTAG debugging interfaces. AP_JTAG is used for embedded CM33 and DSP_JTAG is used for the embedded DSP. You can directly connect JTAG JIG with these connectors for debugging.

2.4.2. Installing AB158x Flash Tool for AB158x EVK

The installation process is the same as 155x. Refer to Section *2.6.2* and complete the same process for 158x.

2.4.3. Installing the AB158x EVK drivers on Microsoft Windows

The installation process is the same as 155x. Refer to Section *2.6.3* and complete the same process for 158x.

2.4.4. Flashing the image to AB158x EVK

You must use a pre-built project file (.cfg) or build your own project to get one (refer to Section *2.2*) before using the IoT Flash Tool.

The process is the same as 155x. Refer to Section *2.6.4* and complete the same process for 158x.

2.4.5. Running the project on AB158x EVK

The process is the same as 1565/1568. Refer to Section *2.5.5* and complete the same process for 158x.

2.4.6. Debugging MCU with the AB158x EVK via Microsoft Windows

This section shows how to debug a project that is built with the GCC compiler using the openOCD debugger tool.

You must install the supporting software on Windows OS before starting a project debugging.

1) Download openocd-0.11.0 from here and extract it into the `<openocd_root>` folder.

   Download the GCC toolchain for your specific version of Windows from here, and extract it into the `<gcc_root>` folder.

2) Install the mbed serial port driver. If the mbed serial port driver is not installed. Refer to Section *2.5.3* for more information.

3) Create a board configuration file and name it "`ab158x.cfg`". Copy the subsequent text to the file:

```
puts "Load AB158x configuration"

#source [find interface/cmsis-dap.cfg]
source [find interface/jlink.cfg]
transport select swd
source [find target/swj-dp.tcl]

set _CHIPNAME AB158x
set _TARGETNAME $_CHIPNAME.CM33
set _CPUTAPID 0x3ba02477

swj_newdap $_CHIPNAME cpu -irlen 4 -expected-id $_CPUTAPID
dap create $_CHIPNAME.dap -chain-position $_CHIPNAME.cpu
target create $_TARGETNAME cortex_m -dap $_CHIPNAME.dap


adapter_khz 1000
reset_config srst_only
```


© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

```
$_TARGETNAME configure -event gdb-attach {
    global _TARGETNAME
      targets $_TARGETNAME
      halt
}

$_TARGETNAME configure -event gdb-detach {
    global _TARGETNAME
      targets $_TARGETNAME
      resume
}

puts "AB158x configuration done"
```

4) Put the board configuration file under `<openocd_root>\share\openocd\scripts\board.`

Start debugging with AB158x EVK:

5) Copy the project `.elf` file from the project `Build` folder to the `<gcc_root>` (refer to step 1). For example, `<sdk_root>\out\ab158x_evk\ earbuds_ref_design\debug\ earbuds_ref_design.elf`.)
6) Open the command window for openOCD.
7) Change the directory in the command window to the openOCD tool folder, e.g., `<openocd_root>\bin`.
8) Disconnect the micro-USB cable from the board to completely power off the board.
9) Use a simulator (e.g., J-Link) to connect the JTAG pin (TMS, TCLK, VCC and GND) or SWD pins (SWDIO, SWCLK, VCC and GND) of AB158x.
10) Reconnect the micro-USB cable to the board to power on the board.
11) Run the subsequent command to start openOCD.

```
openocd.exe -s ..\share\openocd\scripts -f board\ab158x.cfg
```

12) Open the command window for GNU project debugger (GDB).
13) Change the directory in the command window to the tool folder, such as `<gcc_root>\bin`.
14) Run the command to start the GDB.

```
arm-none-eabi-gdb.exe <gcc_root>\earbuds_ref_design.elf
(gdb) target extended-remote:3333
(gdb) monitor reset init
(gdb) load
(gdb) info registers
(gdb) x/10i $pc
```

You now have the openOCD debugging software running on your system.

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

Note:

- openOCD is a free third-party debugging tool (GPL license). To resolve any issues or perform troubleshooting. Refer to the openOCD official forum for more information.
- openOCD debugging cannot work if the system goes into sleep mode. For more detailed information, refer to the *Airoha IoT SDK Power Mode Developers Guide* under `<sdk_root>/mcu/doc`.
- Although we support JTAG debu, we do not recommend clients use this debugging method. The reasons are as follows:
    - our syslog and memory dump are better than JTAG debug, and easier to debug; and
    - our system is a collaboration between MCU and DSP. If JTAG is connected to one side. If this side stops and the other side is freerun, the system will have problems.

2.4.7. Debugging DSP with the AB158x EVK on Microsoft Windows

This section shows the tools and versions for debugging DSP.

- xplorer-9.0.18 and xt-ocd-14.0.8 are the 158x dsp debugging tools.
- You can download the tools from here. You must have an account to download the tools.
- You must have a license to use xplorer.

Although we support JTAG debugging, we do not recommend customers to debug using this method. The reasons are as follows:

- Our syslog and memory dump are better than JTAG debug, and easier to debug; and
- Our system is a collaboration between MCU and DSP. If JTAG is connected to one side and this side stop while and the other side is freerun, the system will have issues.

2.5. Developing on AB1565/AB1568 EVK

2.5.1. Configuring the AB1565/AB1568 EVK

There are two boards for the AB1565/1568 EVK. One board is the main board and the other board is an adaptor for each specific chipset. There are two types of adaptors available: AB1565 and AB1568. Figure 16 shows the front view of the AB1565/AB1568 EVK with the AB1568 adaptor.

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

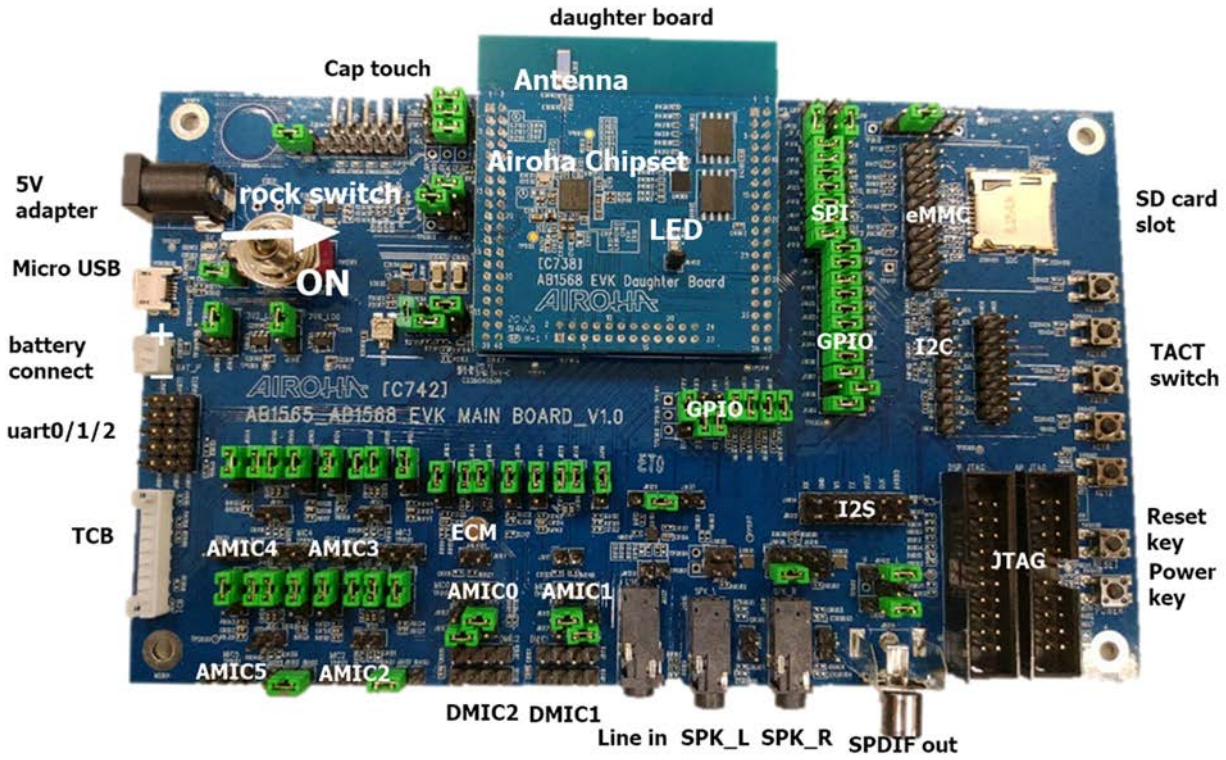

Figure 16. Front view of the AB1565/AB1568 EVK

The EVK can be powered by either USB VBUS or the 5V DC jack. Set the jumpers J2001 1-2 to on to provide power to the board with 5V DC jack.

A micro-USB connector is reserved on the left-side of the EVK. Set jumpers J2001 1-2 to on to provide power to the board via the USB connector. The VBUS power source is from USB VBUS. You can also use the micro-USB connector to download firmware to the board.

AB1565/8 has reserved two JTAG debugging interfaces. AP_JTAG is used for embedded CM4 and DSP_JTAG is used for the embedded DSP. You can directly connect JTAG JIG with these connectors for debugging.

2.5.2. Installing AB1565/AB1568 Flash Tool for AB1565/AB1568 EVK

The process for installing the Flash Tool on AB1565/AB1568 is the same as 155x. Refer to Section *2.6.2* for detailed instructions.

2.5.3. Installing the AB1565/AB1568 EVK drivers on Microsoft Windows

The process for installing the EVK drivers on AB1565/AB1568 is the same as 155x. Refer to Section *2.6.3* for detailed instructions.

2.5.4. Flashing the image to AB1565/AB1568 EVK

The process for flashing the image on AB1565/AB1568 is the same as 155x. Refer to Section *2.6.4* for detailed instructions.

2.5.5. Running the project on AB1565/AB1568 EVK

To use the logging tool:

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

1) Run Airoha.Tool.Kit.exe to start the logging tool.
2) Click the **log binary file** button as shown in Figure 17.
3) Set the **Baud Rate** as shown in Figure 18.
4) Select your serial port on the dropdown list and click the **Connect** button as shown in Figure 19.
5) Click the **Wireshark** button as shown in Figure 20.
6) Reset the target. The log is shown in the log window.

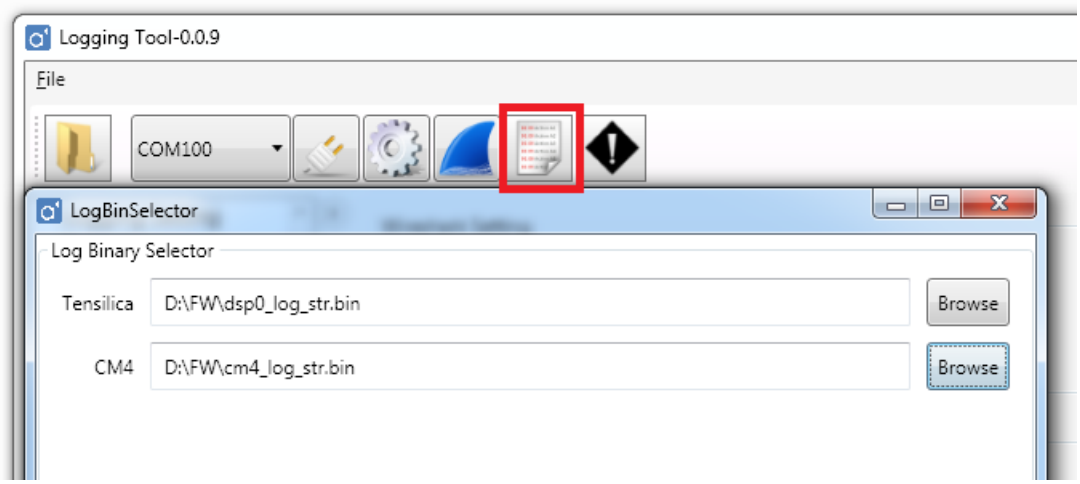

Figure 17. Selecting log binary

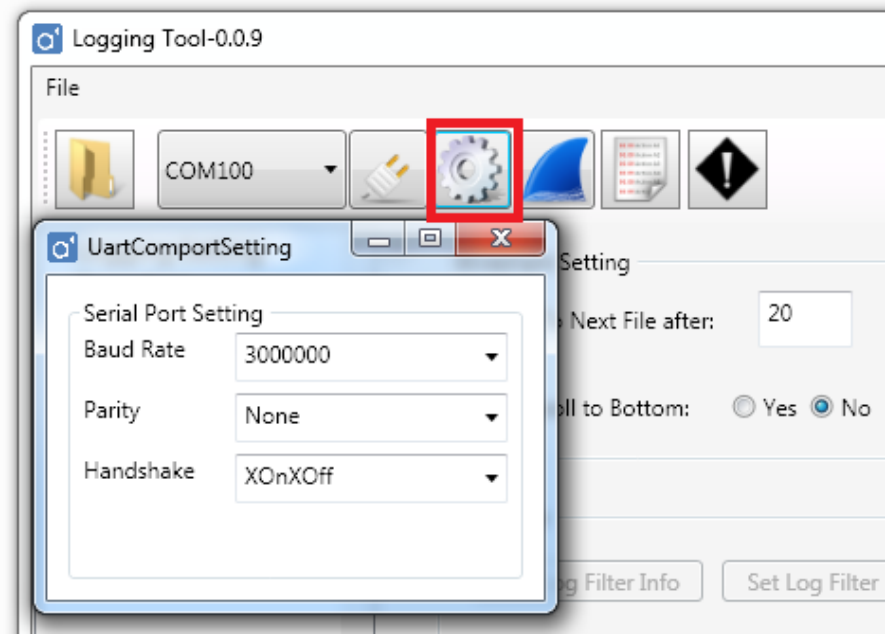

Figure 18. Serial Port Configuration window

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

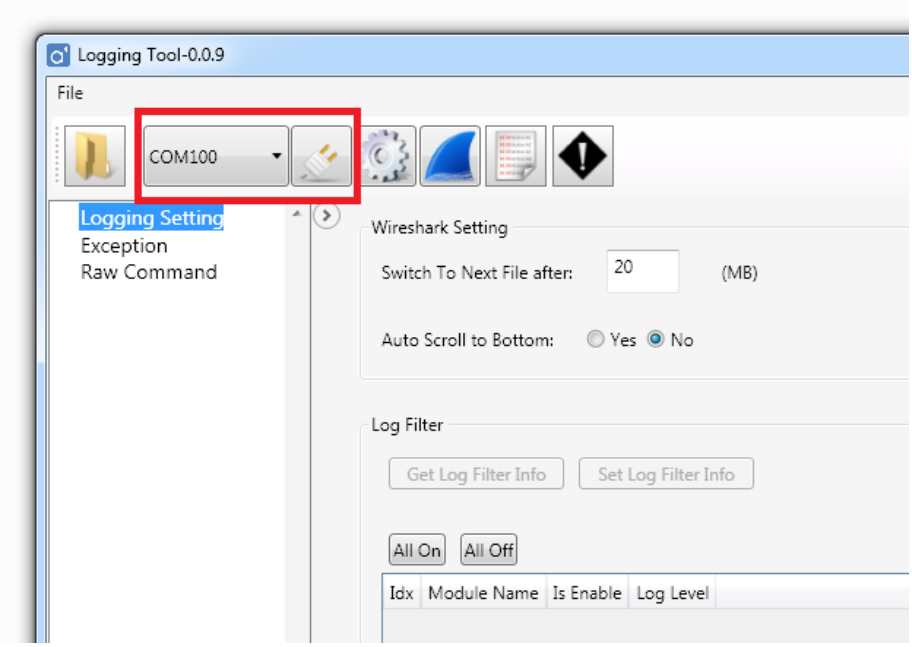

Figure 19. Serial Port Connect button

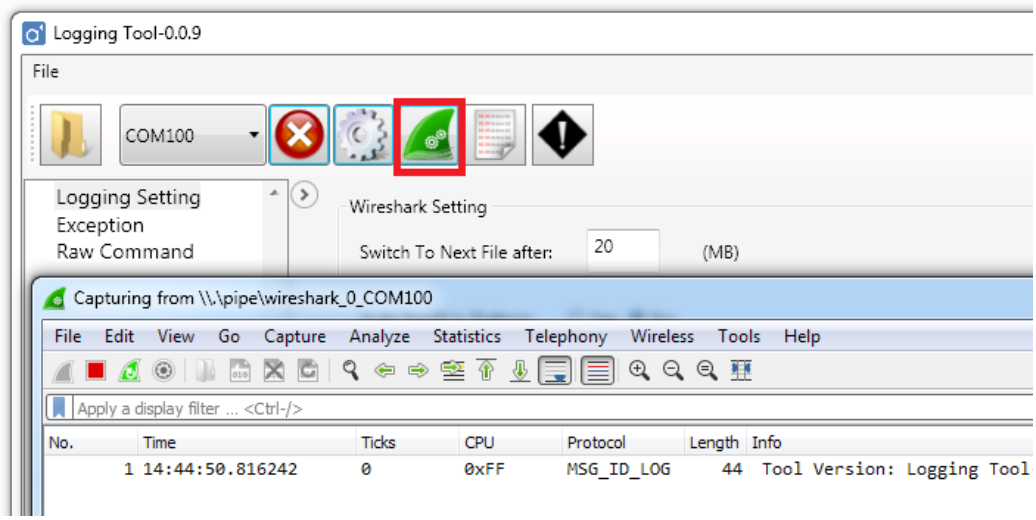

Figure 20. Start Wireshark button

2.5.6. Debugging with the AB1565/AB1568 EVK on Microsoft Windows

The process for debugging on AB1565/AB1568 is the same as 155x. Refer to Section *2.6.6* for detailed instructions.

2.6. Developing on AB155x EVK

2.6.1. Configuring the AB155x EVK

There are two boards for the AB155x EVK. One board is the main board and the other board is an adaptor for each specific chipset. There are two types of adaptors available: AB1558/6 and AB1555. Figure 21 shows the front view of the AB155x EVK with the AB1555 adaptor.

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

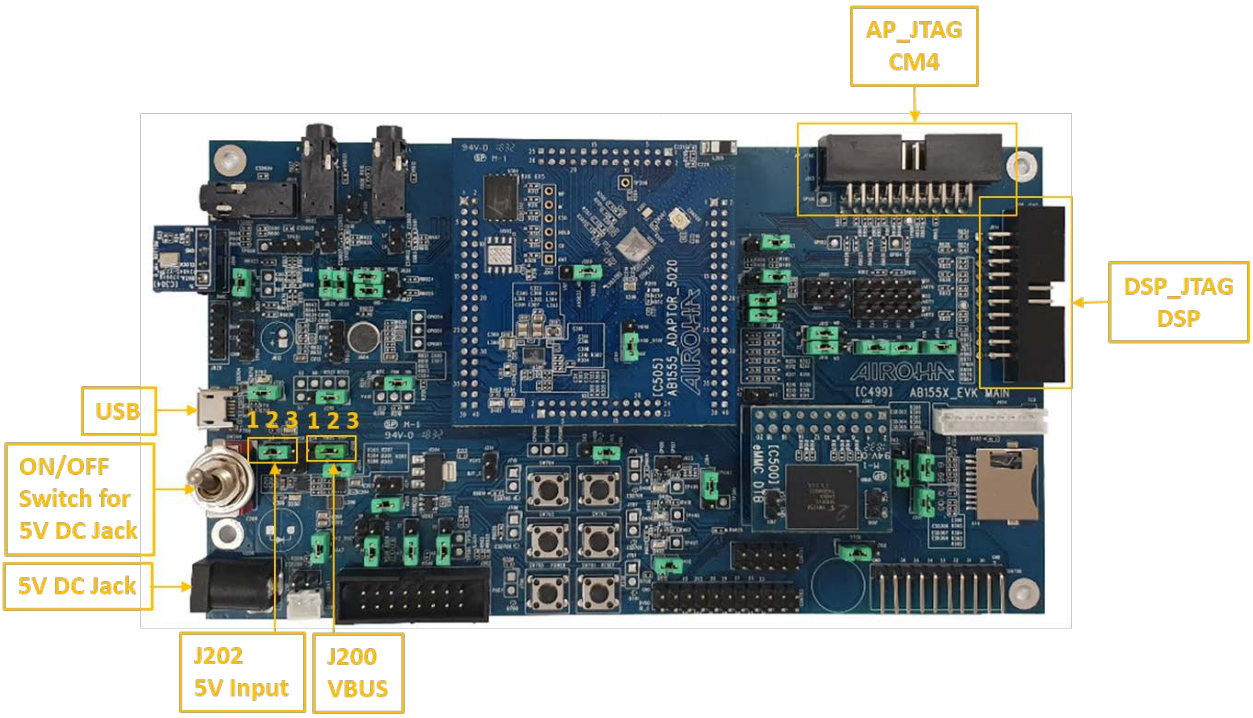

Figure 21. Front view of the AB155x EVK

The EVK can be powered by either USB VBUS or the 5V DC jack. Set the jumpers J202 2-3 on for USB VBUS (5V_USB).

A micro-USB connector is reserved on the left-side of the EVK, this port can also be used as the firmware download port. Set jumpers J200 1-2 on, the VBUS power source is from USB VBUS.

AB155x has reserved two JTAG debugging interfaces, AP_JTAG is used for embedded CM4 and DSP_JTAG is used for the embedded DSP. You can directly connect JTAG JIG with these connectors for debugging.

2.6.2. Installing AB155x Flash Tool for AB155x EVK

IoT Flash Tool is a flexible device flashing tool for application development on the AB155x EVK. It is under `<sdk_root>/mcu/tools/pc_tool/IOT_Flash_Tool`. You can also download it via the MediaTek MOL website. Search for “IoT_Flash_Tool” in “Tool Name” to find the latest version of IoT Flash Tool.

You can run the `FlashTool.exe` inside the folder to start the Flash Tool.

2.6.3. Installing the AB155x EVK drivers on Microsoft Windows

This section shows how to install the AB155x EVK USB drivers on a Microsoft Windows PC.

To install the MediaTek USB Port driver on the EVK in Microsoft Windows:

1) Install the MediaTek USB Port driver from `MS_USB_ComPort_Driver/v3.16.46.1` in the `AB155x_FlashTool` folder.
2) Run `InstallDriver.exe` to install the driver.
3) Use a micro-USB cable to connect the AB155x EVK to your computer.

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

2.6.4. Flashing the image to AB155x EVK

You must use a pre-built project file (.cfg) or build your own project to get one (refer to Section *2.2*) before using the IoT Flash Tool.

To download the firmware to the target device via USB:

1) Disconnect the USB cable to power off the target device.
2) Start IoT Flash Tool
3) Click the **Download** button in the left panel of the Flash Tool GUI.
4) Select **USB** on the **COM Port** dropdown menu. If you do not have the adapter or battery, select the **Enable Download without Battery** option.
5) Click **Open** to load the configuration file. The configuration file is usually named as `flash_download.cfg` and is generated after build process. If it loads successfully, the Flash Tool shows **Download Information**, including the **Name**, **Length** and **File Path** of the firmware binary file.
6) Click the **Start** button to start the download process.
7) Connect the micro-USB cable to the target device and the AB155x to power on the target device. The flash process automatically starts.

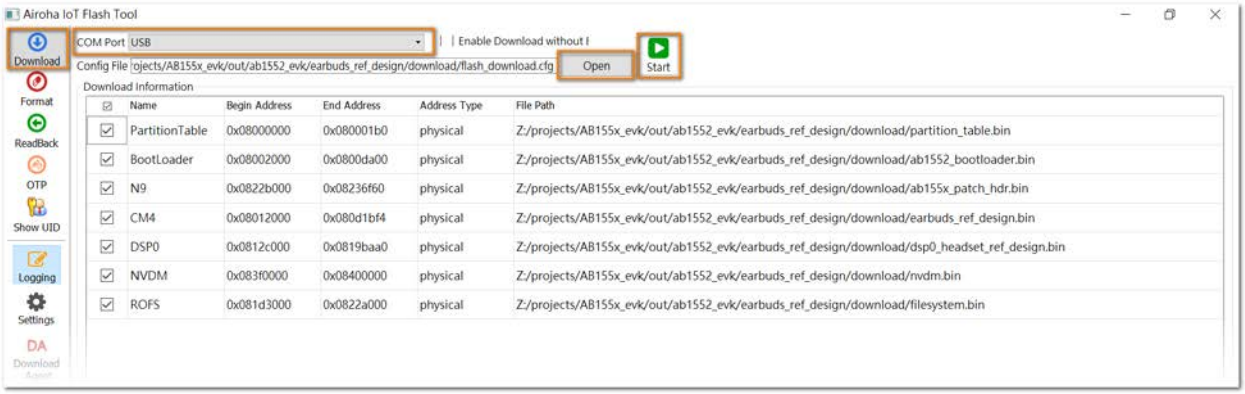

Figure 22. Downloading firmware to a target device via USB

2.6.5. Running the project on AB155x EVK

You must complete the subsequent procedure to use Logging tool:

1) Launch Logging tool.
2) Click the **Serial port** button as shown in Figure 23.
3) Select log binary file as shown in Figure 24
4) Set the **Serial port** and **Baud Rate** is shown in Figure 25.
5) Click the **Start** button as shown in Figure 26.
6) Reset the target. The log appears in the log window.

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

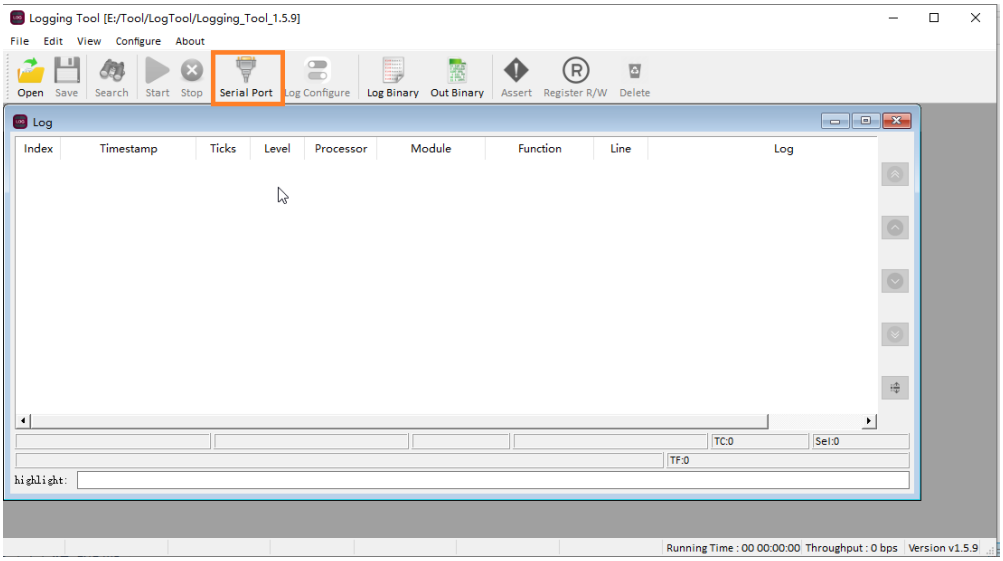

Figure 23. Serial Port button

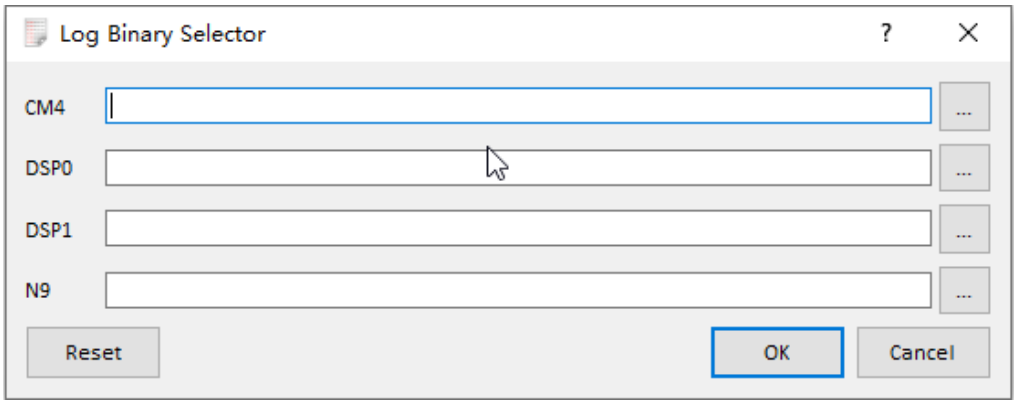

Figure 24. Selecting log binary

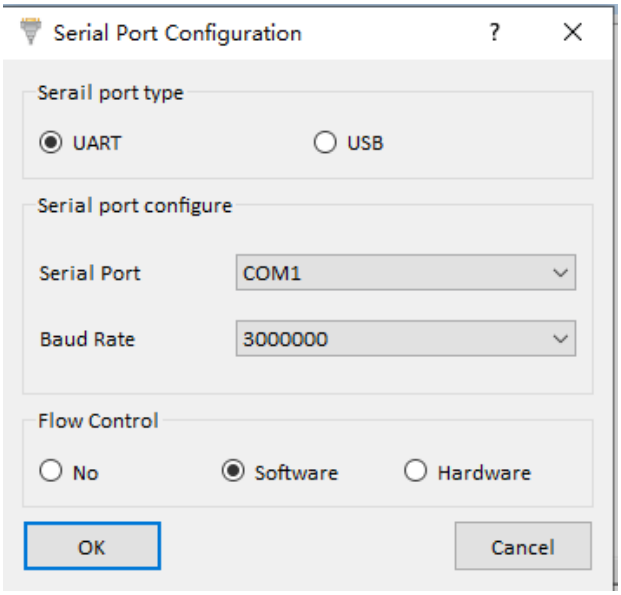

Figure 25. Serial Port Configuration window

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

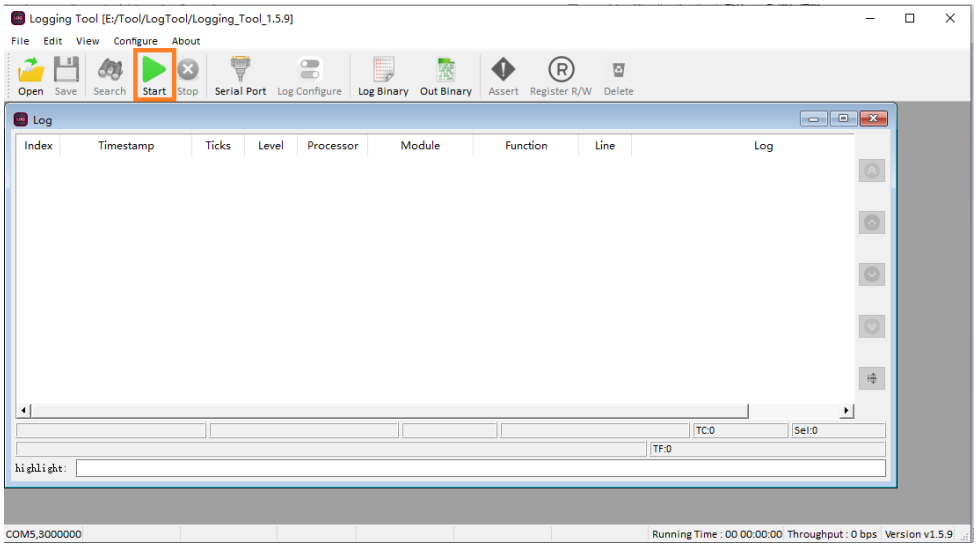

Figure 26. Start button

2.6.6. Debugging with the AB155x EVK on Microsoft Windows

This section shows how to use the openOCD debugger tool to debug a project that is built with the GCC compiler. You must first install the debugging software.

To install the software on Windows:

1) Download openocd-0.10.0 from here and extract the contents into the <openocd_root> folder.

   Download the GCC toolchain for your specific version of Windows from here, and extract the contents into the <gcc_root> folder.

2) Install the mbed serial port driver if the mbed serial port driver is not installed. Refer to Section *2.6.3* for more information.

3) Create a board configuration file and name it "ab155x.cfg". Copy the subsequent content to the new configuration file:

```
puts "Load AB155x configuration"

#source [find interface/cmsis-dap.cfg]
source [find interface/jlink.cfg]
transport select swd
source [find target/swj-dp.tcl]

set _CHIPNAME AB155x
set _CPUTAPID 0x3ba02477

swj_newdap $_CHIPNAME DAP -irlen 4 -expected-id $_CPUTAPID
target create $_TARGETNAME cortex_m -chain-position $_CHIPNAME.DAP

adapter_khz 1000
reset_config srst_only

$_TARGETNAME configure -event gdb-attach {
    global _TARGETNAME
      targets $_TARGETNAME
      halt
}
```


© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

```
$_TARGETNAME configure -event gdb-detach {
    global _TARGETNAME
      targets $_TARGETNAME
      resume
}

puts "AB155x configuration done"
```

4) Put the configuration file under <openocd_root>\share\openocd\scripts\board.

To start debugging with AB155x EVK:

5) Copy the project .elf file from the project Build folder to the <gcc_root> (refer to the step 2). For example, <sdk_root>\out\ab1552_evk\ earbuds_ref_design\debug\ earbuds_ref_design.elf.)
6) Open the command window for openOCD.
7) Change the directory in the command window to the openOCD tool folder (e.g., <openocd_root>\bin).
8) Disconnect the micro-USB cable from the board to completely power off the board.
9) Use a simulator (such as J-Link) to connect the JTAG pin (TMS, TCLK, VCC and GND) or SWD pin (SWDIO, SWCLK, VCC and GND) of AB155x.
10) Reconnect the micro-USB cable to the board to power on the board.
11) Run the command to start openOCD.

```
openocd.exe -s ..\share\openocd\scripts -f board\ab155x.cfg
```

12) Open the command window for GNU project debugger (GDB).
13) Change the directory in the command window to the tool folder, e.g., <gcc_root>\bin.
14) Run the subsequent command to start the GDB.

```
arm-none-eabi-gdb.exe <gcc_root>\earbuds_ref_design.elf
(gdb) target remote localhost:3333
(gdb) monitor reset init
(gdb) load
(gdb) info registers
(gdb) x/10i $pc
```

The openOCD debugging software is now running on your system.

2.7. Creating your own project

This section shows how to use an existing project to create your own project with the AB1565/AB1568 EVK. In this example, a new project named my_project is created from the earbuds_ref_design project.

2.7.1. Using an existing project

To create your own project from an existing project:

1) Copy the folder <sdk_root>/mcu/project/ab1565_ab1568_evk/apps/earbuds_ref_design to a new directory under <sdk_root>/mcu/project/ab1565_ab1568_evk/apps/.
2) Rename earbuds_ref_design to the new project name my_project.

This new project has the same features as the original project. You can build this project to generate an image file that is the same as the original project.

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.

2.7.2. Removing a module

You can remove modules from the new project so that you have a clean start for project development.

To remove a module:

1) Open the project makefile via `<sdk_root>/mcu/project/ab1565_ab1568_evk/apps/my_project/GCC/Makefile`.
2) Locate the module include list of the project.
3) Remove any unwanted modules by deleting or commenting out the related include statement. Finally, the modules are removed from your new project.

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

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

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

2.7.3. Add the source and header files

User-defined project source and header files must be under the `src` and the `inc` folders.

To compile the added source code:

1) Add the .c source files to the "`C_FILES`" variable to the project makefile.
2) Add the header search path to the "`CFLAGS`" variable in the project makefile.

The related `CXX_FILES` and `CXXFLAGS` variables provide support for compiling the source files (i.e. `.cpp`).

There are two intermediate defines (i.e. APP_FILES and SYS_FILES) in the original makefile. Both of them are added to C_FILES.

The line `include $(SOURCE_DIR)/$(APP_PATH_SRC)/apps/module.mk` in the makefile includes the C files in under `<my_projet>/src/apps` and `<sdk_root>/mcu/project/ab1565_ab1568_evk/apps/my_project/GCC/Makefile`

```
...
APP_FILES       += $(APP_PATH_SRC)/main.c
APP_FILES       += $(APP_PATH)/GCC/syscalls.c
...
SYS_FILES       += $(APP_PATH_SRC)/system_ab155x.c
...
CXX_FILES       += ...
...
C_FILES         += $(APP_FILES) $(SYS_FILES)
...
```

Airoha Proprietary and Confidential

© 2019 Airoha Technology Corp. All rights reserved.
Unauthorized reproduction or disclosure of this document, in whole or in part, is strictly prohibited.