



USER'S GUIDE

Apollo4 Family Voice-on-SPOT

Ultra-Low Power Apollo SoC Family

A-SOCAP4-UGGA01EN v1.3



Legal Information and Disclaimers

AMBIQ MICRO INTENDS FOR THE CONTENT CONTAINED IN THE DOCUMENT TO BE ACCURATE AND RELIABLE. THIS CONTENT MAY, HOWEVER, CONTAIN TECHNICAL INACCURACIES, TYPOGRAPHICAL ERRORS OR OTHER MISTAKES. AMBIQ MICRO MAY MAKE CORRECTIONS OR OTHER CHANGES TO THIS CONTENT AT ANY TIME. AMBIQ MICRO AND ITS SUPPLIERS RESERVE THE RIGHT TO MAKE CORRECTIONS, MODIFICATIONS, ENHANCEMENTS, IMPROVEMENTS AND OTHER CHANGES TO ITS PRODUCTS, PROGRAMS AND SERVICES AT ANY TIME OR TO DISCONTINUE ANY PRODUCTS, PROGRAMS, OR SERVICES WITHOUT NOTICE.

THE CONTENT IN THIS DOCUMENT IS PROVIDED "AS IS". AMBIQ MICRO AND ITS RESPECTIVE SUPPLIERS MAKE NO REPRESENTATIONS ABOUT THE SUITABILITY OF THIS CONTENT FOR ANY PURPOSE AND DISCLAIM ALL WARRANTIES AND CONDITIONS WITH REGARD TO THIS CONTENT, INCLUDING BUT NOT LIMITED TO, ALL IMPLIED WARRANTIES AND CONDITIONS OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE AND NON-INFRINGEMENT OF ANY THIRD PARTY INTELLECTUAL PROPERTY RIGHT.

AMBIQ MICRO DOES NOT WARRANT OR REPRESENT THAT ANY LICENSE, EITHER EXPRESS OR IMPLIED, IS GRANTED UNDER ANY PATENT RIGHT, COPYRIGHT, MASK WORK RIGHT, OR OTHER INTELLECTUAL PROPERTY RIGHT OF AMBIQ MICRO COVERING OR RELATING TO THIS CONTENT OR ANY COMBINATION, MACHINE, OR PROCESS TO WHICH THIS CONTENT RELATE OR WITH WHICH THIS CONTENT MAY BE USED.

USE OF THE INFORMATION IN THIS DOCUMENT MAY REQUIRE A LICENSE FROM A THIRD PARTY UNDER THE PATENTS OR OTHER INTELLECTUAL PROPERTY OF THAT THIRD PARTY, OR A LICENSE FROM AMBIQ MICRO UNDER THE PATENTS OR OTHER INTELLECTUAL PROPERTY OF AMBIQ MICRO.

INFORMATION IN THIS DOCUMENT IS PROVIDED SOLELY TO ENABLE SYSTEM AND SOFTWARE IMPLEMENTERS TO USE AMBIQ MICRO PRODUCTS. THERE ARE NO EXPRESS OR IMPLIED COPYRIGHT LICENSES GRANTED HEREUNDER TO DESIGN OR FABRICATE ANY INTEGRATED CIRCUITS OR INTEGRATED CIRCUITS BASED ON THE INFORMATION IN THIS DOCUMENT. AMBIQ MICRO RESERVES THE RIGHT TO MAKE CHANGES WITHOUT FURTHER NOTICE TO ANY PRODUCTS HEREIN. AMBIQ MICRO MAKES NO WARRANTY, REPRESENTATION OR GUARANTEE REGARDING THE SUITABILITY OF ITS PRODUCTS FOR ANY PARTICULAR PURPOSE, NOR DOES AMBIQ MICRO ASSUME ANY LIABILITY ARISING OUT OF THE APPLICATION OR USE OF ANY PRODUCT OR CIRCUIT, AND SPECIFICALLY DISCLAIMS ANY AND ALL LIABILITY, INCLUDING WITHOUT LIMITATION CONSEQUENTIAL OR INCIDENTAL DAMAGES. "TYPICAL" PARAMETERS WHICH MAY BE PROVIDED IN AMBIQ MICRO DATA SHEETS AND/OR SPECIFICATIONS CAN AND DO VARY IN DIFFERENT APPLICATIONS AND ACTUAL PERFORMANCE MAY VARY OVER TIME. ALL OPERATING PARAMETERS, INCLUDING "TYPICALS" MUST BE VALIDATED FOR EACH CUSTOMER APPLICATION BY CUSTOMER'S TECHNICAL EXPERTS. AMBIQ MICRO DOES NOT CONVEY ANY LICENSE UNDER NEITHER ITS PATENT RIGHTS NOR THE RIGHTS OF OTHERS. AMBIQ MICRO PRODUCTS ARE NOT DESIGNED, INTENDED, OR AUTHORIZED FOR USE AS COMPONENTS IN SYSTEMS INTENDED FOR SURGICAL IMPLANT INTO THE BODY, OR OTHER APPLICATIONS INTENDED TO SUPPORT OR SUSTAIN LIFE, OR FOR ANY OTHER APPLICATION IN WHICH THE FAILURE OF THE AMBIQ MICRO PRODUCT COULD CREATE A SITUATION WHERE PERSONAL INJURY OR DEATH MAY OCCUR. SHOULD BUYER PURCHASE OR USE AMBIQ MICRO PRODUCTS FOR ANY SUCH UNINTENDED OR UNAUTHORIZED APPLICATION, BUYER SHALL INDEMNIFY AND HOLD AMBIQ MICRO AND ITS OFFICERS, EMPLOYEES, SUBSIDIARIES, AFFILIATES, AND DISTRIBUTORS HARMLESS AGAINST ALL CLAIMS, COSTS, DAMAGES, AND EXPENSES, AND REASONABLE ATTORNEY FEES ARISING OUT OF, DIRECTLY OR INDIRECTLY, ANY CLAIM OF PERSONAL INJURY OR DEATH ASSOCIATED WITH SUCH UNINTENDED OR UNAUTHORIZED USE, EVEN IF SUCH CLAIM ALLEGES THAT AMBIQ MICRO WAS NEGLIGENT REGARDING THE DESIGN OR MANUFACTURE OF THE PART.

Revision History

Revision	Date	Description
1.0	June 15, 2022	Initial release
1.1	August 29, 2022	<ul style="list-style-type: none">▪ Added new Table 2-1 Support Contact Information (page 11)▪ Updated Table 4-1 Project Directory Structure (page 16)▪ Updated content in section 5 Project Configuration to Select Target Board (page 17-20)▪ Updated content in section 6 Setting Other Project Options (page 21-23)
1.2	November 18, 2022	<ul style="list-style-type: none">▪ Updated section 2 Access and Licensing (page 11).▪ Updated section 4 Supported Hardware (page 13).
1.3	March 29, 2023	<ul style="list-style-type: none">▪ Updated section 1 Introduction.▪ Updated applicable references from "Light" to "Lite" (global)▪ Updated Figure 6-1 Audio Chain and Detection Engine Selection.▪ Update section 14. 1 Build OTA Firmware Image▪ Updated applicable file names (global)

Table of Contents

1. Introduction	10
2. Access and Licensing	11
2.1 Production IP Licensing	11
2.2 Support Contact Information	11
3. IDE Support	12
4. Supported Hardware	13
4.1 Apollo4 Blue Plus Evaluation Board	14
4.2 Apollo4 Plus Evaluation Board	14
4.3 MikroBUS Audio Shield Board	15
4.4 MEMSensing MIC MikroBUS Board (AM-STD-CL)	15
4.5 InvenSense Analog MIC (ISC40310)	16
5. Project Directory Structure	17
6. Project Configuration to Select Target Board	18
6.1 Top-Level Feature of the VoS SDK	18
6.2 Audio Signal Chain and Key Word Detection Library Selection in the Configuration File	19
6.3 Feature Selection in the Configuration File	20
6.4 Bluetooth Low Energy Protocol and Codec Selection in the Configuration File	20
6.5 Audio Feature Configuration in the am_vos_audio.h File	21
7. Setting Other Project Options	22
7.1 MIC Configuration	22
7.1.1 Single MIC	22
7.1.2 2x MICs	22
8. Building the SDK to Create a Custom Binary for Standalone Mode ..	23
9. VoS SDK Tuning Guide	24

9.1 VoS TalkTo/THF Flow Chart	24
9.2 VoS Lite Flow Chart	25
9.3 Voice Activity Detection (VAD)	25
9.3.1 Ambiq VAD	25
9.4 PDM Interface	25
9.5 Sound Pre-Processing (SPP) - TalkTo Beamformer/SCNR	26
9.6 Stereo to Mono (without SPP)	27
9.7 Key Word (Command Phrase) Engine	27
9.8 Audio Codec Encoder (OPUS or ADPCM)	28
9.9 Bluetooth Low Energy TX	28
10. Operation with the Alexa App	30
10.1 EVB LED Indication	30
10.2 Building VoS Firmware Binary for AMA Protocol	30
10.3 iOS App Installation	31
10.4 Android App Installation	31
10.5 Connecting VoS EVB Device with Mobile Phone App	31
11. Operation with Google ATVV	33
11.1 Building VoS Firmware Binary for ATVV Protocol	33
11.2 Connect VoS EVB Device with Android TV (or Set Top Box)	34
12. AMVoS Profile with Common BLE Test App	36
12.1 Build VoS Firmware	36
12.2 Download and Run General BLE Test App	36
12.3 Streaming Audio Data Through BLE	37
13. Sensory VoiceHub	39
13.1 Wake Word + Voice Command	39
13.2 Wake Word Only	39
13.3 Voice Command Only	39
13.4 VoiceHub Voice Detection Model in VoS	40
14. Building an OTA Image	42
14.1 Build OTA Firmware Image	42

14.2 Buildtools Installation	43
14.3 Python Installation	43
15. OTA Image Download	44
15.1 Download Firmware to the EVB	44
15.2 Update the EVB Firmware Via OTA	45
15.3 OTA App for Android	46
16. Audio Data Recording Using RTT	47
16.1 Build Image for Audio Data Recording	47
16.2 Install Software on PC	48
16.2.1 Install J-Link v6.47x or Later	48
16.2.2 Install Python 3.6 or Later	48
16.2.3 RTT PCM Recorder Script File Package	48
16.3 Audio (Voice) Data Recording	48
16.4 Convert Raw Data to WAV File Format	49
17. Debug Message Output	50
17.1 UART	50
17.2 SWO Output	52

List of Tables

Table 2-1 Support Contact Information	11
Table 5-1 Project Directory Structure	17
Table 6-1 Top-Level Feature of the VoS SDK	18
Table 6-2 Required Modules in the Four Base Examples	19
Table 6-3 Feature Selection in the Configuration File	20
Table 6-4 Bluetooth Low Energy Service/Protocol Selection in Configuration File	20
Table 6-5 Feature Configuration in the am_vos_audio.h File	21
Table 10-1 LED Indications of System Status	30

List of Figures

Figure 4-1 Apollo4 Blue Plus EVB	14
Figure 4-2 Apollo4 Plus EVB (not included in AMA4AUD)	14
Figure 4-3 MikroBUS Audio Shield Board with 4 Slots	15
Figure 4-4 Top-Mount Microphone MikroBUS Board	15
Figure 4-5 InvenSense Analog MIC	16
Figure 6-1 Audio Chain and Detection Engine Selection	19
Figure 7-1 Endfire Microphone Beamforming	22
Figure 9-1 VoS TalkTo/THF Flow Chart	24
Figure 9-2 VoS Lite Flow Chart	25
Figure 9-3 am_pdm0_isr() in am_vos_isr.c File	26
Figure 9-4 DSPC TalkTo Pre-Processing	26
Figure 9-5 Input Data to DSPC TalkTo (Beam Former/SCNR) at am_spp_input_push() in am_vos_talkto.c	26
Figure 9-6 Output Data from DSPC TalkTo at am_spp_output_pop() in am_vos_talkto.c	27
Figure 9-7 am_vos_stereo_to_mono_proc() in am_vos_audio.c File	27
Figure 9-8 Keyword Detection Check in am_vos_engine_process()	28
Figure 9-9 Ring Buffer Status Check and Encode Audio Data in am_vos_codec_task()	28
Figure 9-10 Encoded Audio Data Size Check in am_vos_codec_task()	29
Figure 9-11 Define Example for AMA Protocol Enabled	29
Figure 10-1 Setting up Device in Alexa App	32
Figure 11-1 Android TV Settings Menu	34
Figure 11-2 Searching Result in Android TV	34
Figure 11-3 Google Voice Assistant Response Screen	35
Figure 12-1 Building VoS without AMA/ATVV Protocol	36
Figure 12-2 LightBlue Explorer	36
Figure 12-3 UUID in LightBlue Properties Page	37
Figure 12-4 Listen for Notifications in LightBlue	37
Figure 12-5 Streaming Data in LightBlue	38
Figure 13-1 Sensory Module Configuration	40
Figure 13-2 Include Voice Detection Model Files at am_vos_thf.c File	41
Figure 13-3 Variable Name Modification of Wake Word Files	41
Figure 14-1 OTA Image Generated by Makefile for Apollo4	42
Figure 14-2 Error Message Due to Non-existing make.exe	43
Figure 15-1 J-Flash Lite Address Setting for the Apollo4 Plus	44
Figure 15-2 iTunes File Manager	45
Figure 15-3 Ambiq OTA App Scan Result	45
Figure 16-1 RTT Recorder Definition	47
Figure 16-2 Recorder Data Select	47
Figure 16-3 Disable Codec and BLE When Using RTT Recorder	47
Figure 16-4 System Path Environment Variable	48
Figure 16-5 Run RTT datalogger batch file	49
Figure 17-1 configUSE_STDIO_PRINTF definition in am_vos_sys_config_h	50

Figure 17-2 configUSE_PRINTF_UART0 definition in am_vos_sys_config_h	50
Figure 17-3 Serial Port Setup	51
Figure 17-4 Terminal Setup	51
Figure 17-5 UART print out using Tera Term	51
Figure 17-6 configUSE_STDIO_PRINTF definition in am_vos_sys_config.h	52
Figure 17-7 configUSE_PRINTF_SWO definition in am_vos_sys_config.h	52
Figure 17-8 SWO device and clock configuration	52
Figure 17-9 SWO debugging message print out	52

SECTION

1

Introduction

This document describes example builds in the Voice-on-SPOT® (VoS®) ultra-low power framework project that demonstrate Sound Pre-Processing (SPP), Key Word Detection (KWD) Engines and other features on the Ambiq® Apollo4 Plus SoCs. These examples support the following configurations and features:

- Inclusion or exclusion of audio and other functional features to create baseline and diagnostic builds.
- One or two PDM-driven digital microphones connected to the Apollo4 Plus SoC's stereo PDM interface.
- One or two analog microphones connected to the Apollo4 Plus SoC's AUDADC interface.
- Capture of an audio buffer output over Bluetooth® Low Energy (AMA protocol) to Amazon Alexa app for using Alexa Voice Service (AVS).
- Capture of an audio buffer output over Bluetooth Low Energy (ATV Voice Service protocol) to Android TV or set-top box for using Google Assistant.
- Transfer of an audio buffer between the Apollo4 family EVBs and over SEGGAR RTT interface to a PC for the generation of a WAV file.
- Ambiq's patented SPOT® technology provides up to 10x lower active power consumption than standard Arm® Cortex®-M4 core.

SECTION

2

Access and Licensing

2.1 Production IP Licensing

DSP Concepts, Sensory, and Ambiq have agreed to a third-party software ecosystem licensing model to share IP for demo purposes only.

2.2 Support Contact Information

Table 2-1: Support Contact Information

Company	Email
Ambiq	support@ambiq.com
DSP Concepts	info@dspconcepts.com
Sensory	techsupport@sensoryinc.com

SECTION

3

IDE Support

AmbiqVoS, also referred to VoS SDK in this document, is based on AmbiqSuite SDK R4.3.0.

- **IAR Embedded Workbench IDE**
 - Developed and tested using IAR IDE version Arm 8.32.2
- **Keil uVision IDE**
 - uVision v5.27.1.0

SECTION

4

Supported Hardware

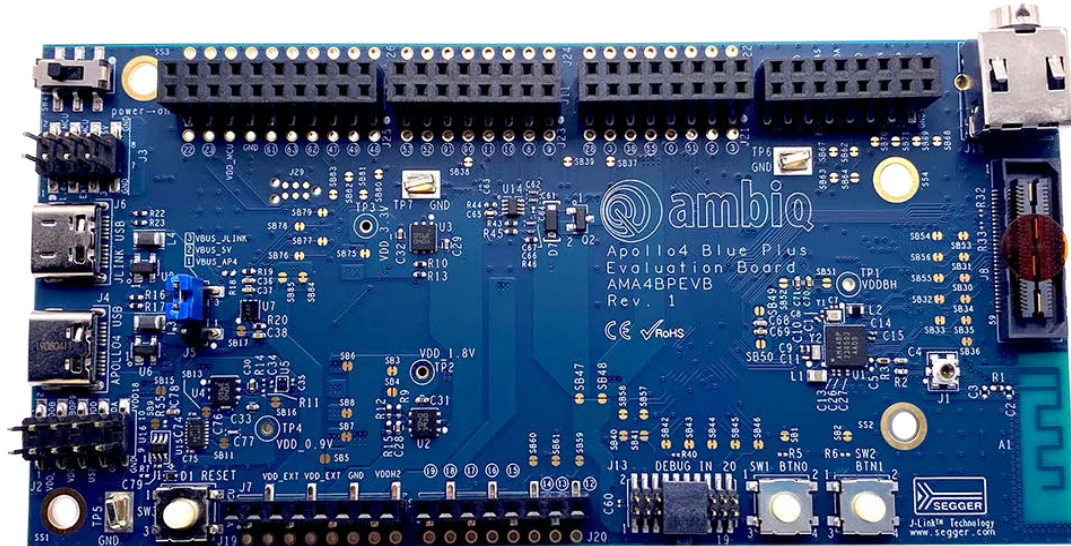
This section describes the hardware used to evaluate the Apollo4 Family VoS Kit

- An Apollo4 evaluation board
 - Apollo4 Plus EVB
 - Apollo4 Blue Plus EVB
- Apollo4 Audio Kit (AMA4AUD)
 - MikroBUS Audio Shield Board
 - MEMSensing Digital MIC Click Board
 - Invensense Analog MIC Click Board

4.1 Apollo4 Blue Plus Evaluation Board

- Apollo4 Blue Plus SoC
- Contains sockets for the **MikroBUS** Shield board

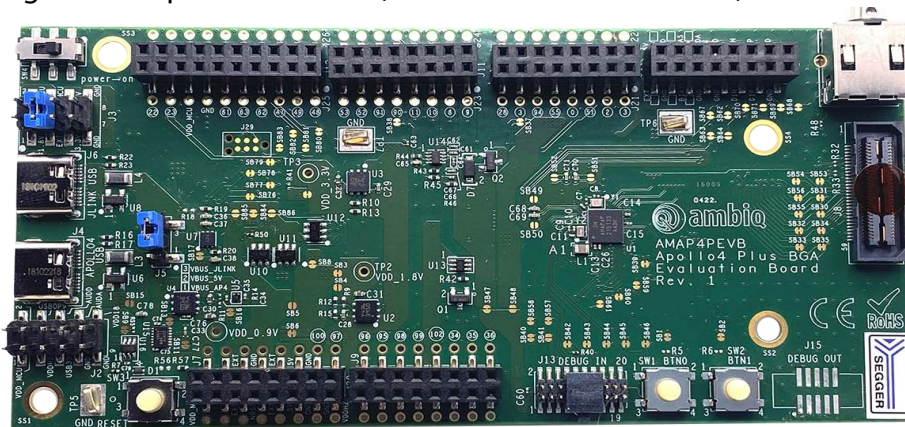
Figure 4-1: Apollo4 Blue Plus EVB



4.2 Apollo4 Plus Evaluation Board

- Apollo4 Plus SoC (without Bluetooth Low Energy Support)
- Contains sockets for the shield

Figure 4-2: Apollo4 Plus EVB (not included in AMA4AUD)

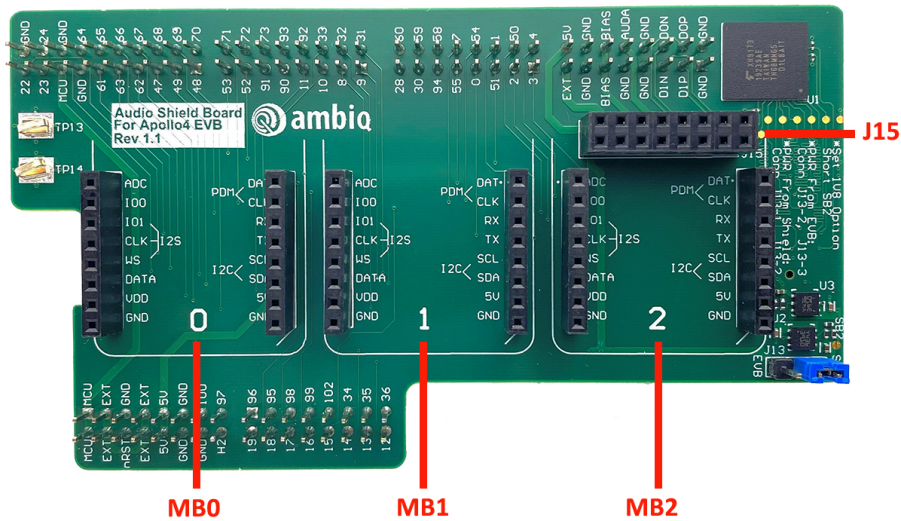


4.3 MikroBUS Audio Shield Board

- 3 digital MIC slots - MB0/MB1/MB2
- Analog MIC slot - J15

NOTE: Make sure J13 jumper on the shield board is shorted to select the MikroBUS shield's power supply.

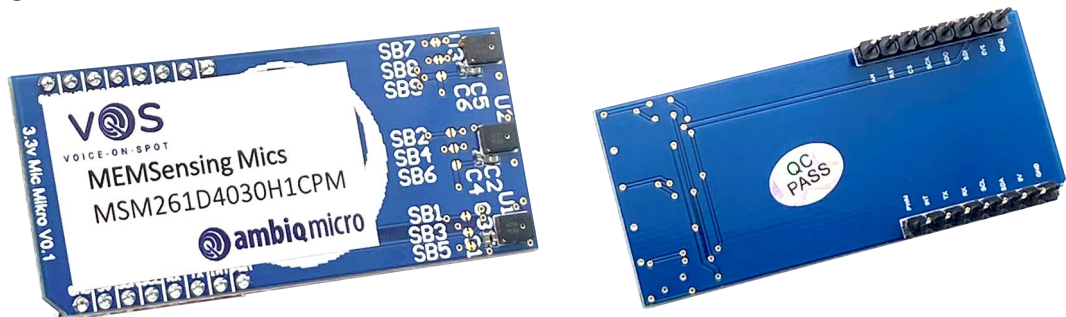
Figure 4-3: MikroBUS Audio Shield Board with 4 Slots



4.4 MEMSensing MIC MikroBUS Board (AM-STD-CL)

- MEMSensing MSM261D4030H1CPM digital MICS with selectable 1-mic or 2-mic configuration
- If 2 mics, 1cm or 2cm spacing depending on SBx (solder bridge) configuration
- Default is 2 top-mounted mics (U1, U3) with 2cm spacing
- Configuration definition:
USE_DMIC_MB0, USE_DMIC_MB1 or USE_DMIC_MB2

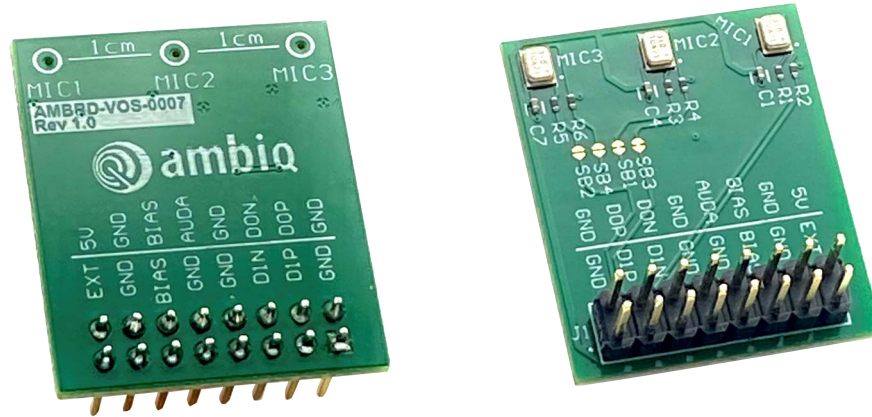
Figure 4-4: Top-Mount Microphone MikroBUS Board



4.5 InvenSense Analog MIC (ISC40310)

- Selectable 1-mic or 2-mic configuration
- If 2 mics, 1cm or 2cm spacing depending on SBx (solder bridge) configuration
- Default is 2 bottom-mounted mics (MIC1, MIC2) with 1cm spacing
- Configuration definition: **USE_AMIC_SINGLE, USE_AMIC_DUAL**

Figure 4-5: InvenSense Analog MIC



SECTION

5

Project Directory Structure

Table 5-1: Project Directory Structure

Project	Directory
VoS Code - Common	ambiq_vos\am_vos
VoS Code – BLE Common	ambiq_vos\am_vos_ble
VoS Code – Codec	ambiq_vos\codec
VoS Code – BLE Protocols	ambiq_vos\protocol
VoS Project	boards\apollo4p_evb\examples boards\apollo4p_blue_evb\examples
Third-Party Add-ons (DSPC, Sensory)	third_party\

Project Configuration to Select Target Board

6.1 Top-Level Feature of the VoS SDK

The VoS SDK include the following top-level features:

Table 6-1: Top-Level Feature of the VoS SDK

IP	IP Provider	Description
AB	Ambiq	Universal Audio Buffer (Raw, Encoded Audio data buffer)
TalkTo	DSPC	Audio Weaver Engine audio processing. Includes beam forming (BF) and single-channel noise reduction (SCNR)
THF	Sensory	Truly Hands Free Keyword Detection function - “Alexa” THF models are 64K v3c models
OPUS	IETF/Xiph Org	Audio codec that incorporates technology from Skype SILK codec and Xiph.Org’s CELT.
ADPCM	SpanDSP	Adaptive differential pulse-code modulation encoder (Open Source).

The SDK consists of 4 base example projects:

- **vos_talkto_thf / vos_ble_talkto_thf**– contains both DSPC audio processing and Sensory Truly Hands Free wake word detection features
- **vos_talkto / vos_ble_talkto** – contains DSPC audio processing features
- **vos_thf / vos_ble_thf** – contains Sensory Truly Hands Free features
- **vos_lite/ vos_ble_lite** – contains neither audio pre-processing or keyword detection engine. This version support manual trigger (e.g., “push to talk”) and Bluetooth Low Energy transfer with AMA/ATVV protocol.

Table 6-2 on page 19 summarizes the modules (Add-on) needed for each example project that are in addition to the base set of files common to all projects.

Table 6-2: Required Modules in the Four Base Examples

Examples	DSPC TalkTo	Sensory THF
vos_talkto_thf	●	●
vos_talkto	●	
vos_thf		●
vos_lite		
vos_ble_talkto_thf	●	●
vos_ble_talkto	●	
vos_ble_thf		●
vos_ble_lite		

Base SDK package (**AmbiqVoS_R4.x.x.zip**) can build only two examples - **vos_lite**, **vos_ble_lite**.

To build TalkTo, THF examples, it needs additional files and libraries. They are going to be provided as separate packages.

- **DSP Concepts TalkTo + Sensory THF:** AmbiqVoS_R4.x.x_TalkTo_THF.zip
- **DSP Concepts TalkTo:** AmbiqVoS_R4.x.x_TalkTo.zip
- **Sensory THF:** AmbiqVoS_R4.x.x_THF.zip

To build example **vos_talkto_thf**, it needs the DSPC + Sensory package (**AmbiqVoS_R4.x.x_TalkTo_THF.zip**). To build example **vos_talkto**, it needs the DSPC (**AmbiqVoS_R4.x.x_TalkTo.zip**) package.

The subsections that follow describe the settings in the configuration file and the selection of specific files for building each of the 4 base projects, or variations of those projects. Target MIC board and feature selections are selected/enabled by #define's in **am_vos_sys_config.h**. These selections are followed by the target selection in the **boards\evb_name\examples\vos\vos_xxx\src** folder of the VoS project. (e.g., **evb_name** for the Apollo4 Blue Plus EVB is **apollo4p_blue_evb**).

6.2 Audio Signal Chain and Key Word Detection Library Selection in the Configuration File

Selected by project target.

Figure 6-1: Audio Chain and Detection Engine Selection

```
#if defined (AM_VOS_TALKTO)
    #define configUSE_DSPC_TalkTo      1      // DSPC AWE frame work switch
#endif // AM_VOS_TALKTO

#if defined (AM_VOS_THF)
    #define configUSE_Sensory_THF      1      // Sensory detection Lib
#endif // AM_VOS_THF

#if defined (AM_VOS_FLUENT)
    #define configUSE_Fluent           1      // Fluent detection Lib
#endif // AM_VOS_FLUENT
```

6.3 Feature Selection in the Configuration File

Table 6-3: Feature Selection in the Configuration File

Field in Configuration File	1	0
configUSE_RTT_RECORDER	RTT audio recorder enabled (SWO)	Disabled
configUSE_STDIO_PRINTF	Debug print log enabled (UART / SWO)	No debug log printing.
configUSE_AMBIQ_VAD	Enable Ambiq VAD (Voice Activity Detection)	Disabled
configUSE_BLE	Enable audio data transfer via Bluetooth Low Energy.	Disabled
configUSE_AUDIO_CODEC	During audio streaming, enable mSBC/OPUS/ADPCM encoding.	Disabled
configUSE_OVVP_DOUBLE_TAP	Enable OVVP feature and double tap detection using G-sensor.	Disabled
configUSE_PAIRING_MODE_BTN	Enable removing pairing information button switch	Disabled
configUSE_PUTH_TO_TALK	Enable push to talk using on-board button switch.	Disabled
configUSE_MUTE_MIC	Enable mute MIC using button switch.	Disabled
configUSE_PREROLL	Buffer pre-roll feature for key word verification.	No pre-roll
configUSE_AMVOS_AMA	Use AMA protocol to support Alexa app.	Common GATT profile support Read/Write notification.
configUSE_AMVOS_ATVV	Use ATVV protocol to support Android TV connecting as a remote controller.	Common GATT profile support Read/Write notification.

6.4 Bluetooth Low Energy Protocol and Codec Selection in the Configuration File

- 1 : Should be 1 when use it with specific service/protocol
- 0 : Should be 0 (not supported)
- ● : This could be selected as the user's design requires

Table 6-4: Bluetooth Low Energy Service/Protocol Selection in Configuration File

Field in Configuration File	AMA (Alexa)	ATVV (Google)	Common GATT
configUSE_BLE	1	1	1
configUSE_AUDIO_CODEC	1	1	1
configUSE_AMVOS_AMA	1	0	0
configUSE_AMVOS_ATVV	0	1	0
configUSE_AMVOS_HID	0	1	0
configUSE_PREROLL	1	0	0
configUSE_OPTIM_OPUS	1	0	●
configUSE_ADPCM	0	1	●

6.5 Audio Feature Configuration in the am_vos_audio.h File

Table 6-5: Feature Configuration in the am_vos_audio.h File

Field in Configuration	Description
AUDIO_KWD_TIMEOUT_S	Timeout of streaming in seconds after wake word detected. Default is 8 seconds.
AUDIO_PREROLL_TIME_MS	If audio buffer transfer is enabled for UART or Bluetooth Low Energy, this defines ms of pre-roll data before key word (wake word) is detected from pre-buffer to transfer. Current value is 500 ms.

SECTION

7

Setting Other Project Options

7.1 MIC Configuration

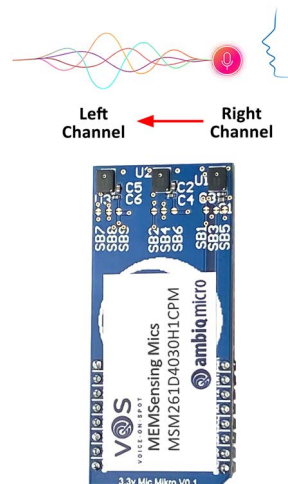
7.1.1 Single MIC

- VoS SDK support one digital MIC without audio pre-processing. Supports THF (vos_thf, vos_ble_thf) and Lite example (vos_lite, vos_ble_lite).
- Using left channel setting of DMIC/AMIC as a default.

7.1.2 2x MICs

- VoS SDK supports two analog/digital microphones with 2cm separation (distance) as a default.
- These stereo audio data is passed to SPP – Sound pre-processing module. That is DSPC TalkTo.
- Default beamforming direction is right to left channel.

Figure 7-1: Endfire Microphone Beamforming



SECTION

8

Building the SDK to Create a Custom Binary for Standalone Mode

Use the following procedure to build the SDK to create a custom binary for standalone mode:

1. In IAR, build the project, complete the following:
 - a. Select **Project**, then select **Clean**.
 - b. Select **Project**, then select **Rebuild All**.
2. Confirm the board is connected by USB, and powered.
3. In IAR, download the bin file by selecting **Project**, then select **Download**, and then select **Download active application**.
4. Cycle power on EVB after Flashing MCU to initiate operation.

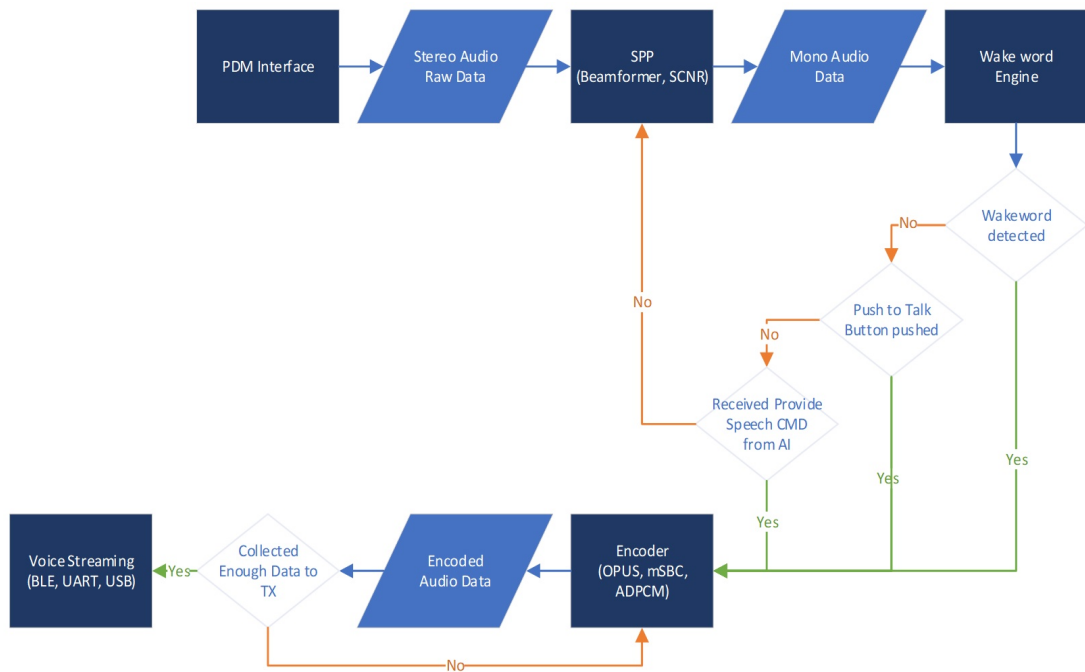
SECTION

9

VoS SDK Tuning Guide

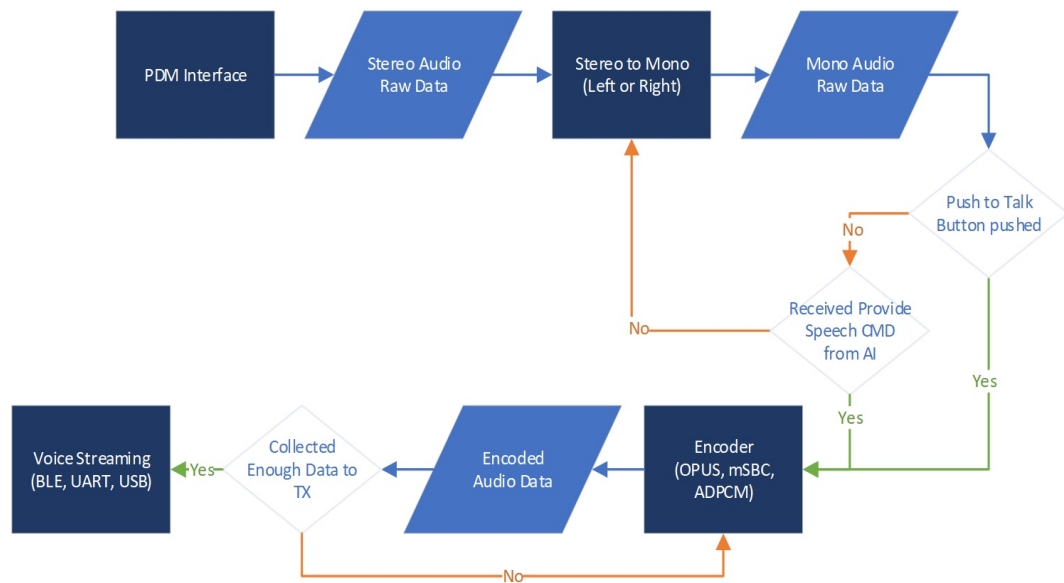
9.1 VoS TalkTo/THF Flow Chart

Figure 9-1: VoS TalkTo/THF Flow Chart



9.2 VoS Lite Flow Chart

Figure 9-2: VoS Lite Flow Chart



9.3 Voice Activity Detection (VAD)

VAD is designed for reducing CPU runtime for wake word detection engine and save power consumption.

9.3.1 Ambiq VAD

The Ambiq VAD algorithms, as part of the Voice-on-SPOT software suite, provides two low-complexity, low-power algorithms that provide reasonable separation of voiced speech from background noise for always-on listening.

The audio frame, typically 16-160 samples at 16 kHz sampling rate, is passed to the VAD algorithm, which provides a binary classification of speech/no-speech. These VAD algorithms can operate sample-by-sample as well.

- Check the number of wake word detected and WoS triggered for specific period, then determine next WoS trigger level threshold (Sensitivity).

9.4 PDM Interface

- **am_pdm_isr()** function is called each time the PDM interrupt is triggered.
- Puts stereo data in a ring buffer and sends notification to the audio processing task.

Figure 9-3: am_pdm0_isr() in am_vos_isr.c File

```

// Test code for PDM wakeup time measurement
//am_hal_gpio_state_write(LED_SYSTEM, AM_HAL_GPIO_OUTPUT_TOGGLE);
//
// Once our DMA transaction completes, we will disable the PDM and send a
// flag back down to the main routine. Disabling the PDM is only necessary
// because this example only implemented a single buffer for storing FFT
// data. More complex programs could use a system of multiple buffers to
// allow the CPU to run the FFT in one buffer while the DMA pulls PCM data
// into another buffer.
//
//
if (ui32Status & AM_HAL_PDM_INT_DCMP)
{
    // trigger next traction
    PDMn(0)->DMATOTCOUNT = AM_SPP_FRAME_SAMPLES * SAMPLE_32BIT; // FIFO unit in bytes

    am_audio_buffer_push(AM_AUDIO_BUFFER_STEREO, g_ui32PDMDataBuffer, PCM_FRAME_SIZE * SAMPLE_32BIT);
}
#endif // configUSE_OWP_DOUBLE_TAP

```

9.5 Sound Pre-Processing (SPP) - TalkTo Beamformer/SCNR

- VoS uses TalkTo beam former to convert stereo audio to mono output data with directional audio gain control. It is one beam direction and set up for maximum SNR.
- SCNR is processed in DSPC TalkTo modules after beam forming.

Figure 9-4: DSPC TalkTo Pre-Processing



- Two audio input and one output can be accessed with the code below. It can be reviewed with define **configUSE_TalkTo**.

Figure 9-5: Input Data to DSPC TalkTo (Beam Former/SCNR) at am_spp_input_push() in am_vos_talkto.c

```

//
// Data IO of Layout
// Get Current AWE Layout number of channels
// Layout has 1 input and 1 output
//
awe_layoutGetChannelCount(&g_AWEInstance, 0, &fwInCount, &fwOutCount);
if(fwInCount == spp_input->ui32SppInChNum)
{
    for(uint32_t ch_indx=0; ch_indx<fwInCount; ch_indx++)
    {
        awe_audioImportSamples(&g_AWEInstance, (void*)spp_input->SppInputArray[ch_indx], 1, ch_indx, (Sar
    }
}
else
{
    AM_SPP_LOG("Input channel number is not aligned with loading AWB...\n");
    return;
}

```

Figure 9-6: Output Data from DSPC TalkTo at am_spp_output_pop() in am_vos_talkto.c

```

//
// Data IO of Layout
// Get Current AWE Layout number of channels
// Layout has 1 input and 1 output
//
awe_layoutGetChannelCount(&g_AWEInstance, 0, &fwInCount, &fwOutCount);
if(fwOutCount == spp_output->ui32SppOutChNum)
{
    for(uint32_t ch_indx=0; ch_indx<fwOutCount; ch_indx++)
    {
        awe_audioExportSamples(&g_AWEInstance, (void*)spp_output->SppOutputArray[ch_indx], 1, ch_indx, ($
#ifdef AM_configUSE_LOG_SPP
        amu2s_send(Amu2s_spp, spp_output->SppOutputArray[ch_indx], AM_SPP_FRAME_SAMPLES*AM_SPP_OUT_SAMPLE
#endif // AM_configUSE_LOG_SPP
    }
}
else
{
    AM_SPP_LOG("Output channel number is not aligned with loading AWB...\n");
    return;
}

```

9.6 Stereo to Mono (without SPP)

- The **am_vos_stereo_to_mono_proc()** function converts stereo data to mono audio data.

Figure 9-7: am_vos_stereo_to_mono_proc() in am_vos_audio.c File

```

// This process only take 1-channel data into WWE
//
void am_vos_stereo_to_mono_proc(int32_t *nLRSample, int16_t *nMonoSample)
{
    int32_t nSample;

    for (nSample = 0; nSample < AM_SPP_FRAME_SAMPLES; nSample++)
    {
        // Without voice pre-processing(e.g. Beamforming), using right MIC's data as a default.
        nMonoSample[nSample] = nLRSample[nSample] >> 16; // Right channel
        //nMonoSample[nSample] = nLRSample[nSample] & 0xFFFF; // Left channel
    }
}

```

- Other audio pre-processing functions can be added here (e.g., Beam Former, Noise Reduction)

9.7 Key Word (Command Phrase) Engine

- VoS support several third-party keyword and command phrase detection engine.
 - Sensory THF (with VoiceHub): keyword and command phrase
- When mono audio buffer reach the specific size, call **am_vos_engine_process()** function to checking trigger of key word (command phrase).
- **am_vos_engine_process()** function is implemented for each key word engine's usage. Sensory THF engine example is as below.

Figure 9-8: Keyword Detection Check in am_vos_engine_process()

```

case STATE_TRIGGER:
    result = SensoryProcessBrick(frame, ap);
#if configUSE_OVVP_DOUBLE_TAP
    if(result == ERR_OK && (s_flag > 0))
#else // configUSE_OVVP_DOUBLE_TAP
    if(result == ERR_OK)
#endif // configUSE_OVVP_DOUBLE_TAP
    {
        am_vos_reset_detected_flag();
        g_sVosSys.ui8KwdDetectedFlag = 1;

        xTimerReset(am_KWD_timers[AM_APP_TIMER_HEART_BEAT], 0); // reset heart beat timer to

#if USE_DMIC_MB3_VM3011 && configUSE_WOS
        g_ui32DetectionCount++;
#endif // USE_DMIC_MB3_VM3011

#if configUSE_OVVP_DOUBLE_TAP
        AM_APP_LOG_INFO("\n[AM-VoS] Keyword Detected! s_flag = %d\n", s_flag);
#else // configUSE_OVVP_DOUBLE_TAP
        AM_APP_LOG_INFO("\n[AM-VoS] Keyword Detected! [%d]", t->wordID);
#endif // configUSE_OVVP_DOUBLE_TAP

        am_vos_WW_LED_display();

#if (configUSE_AMVOS_AMA && configUSE_PREROLL)
        SensoryFindStartpoint(ap, &stIndex);
        SensoryFindEndpoint(ap, &epIndex, &tailCount);

```

- If THF engine detects the key word, the **SensoryFindStartpoint()** and **SensoryFindEndpoint()** functions are called to calculating the start/end index of the key word.

9.8 Audio Codec Encoder (OPUS or ADPCM)

- After the key word is detected (or **Push to Talk/Provided Speech** command is received), the codec task checks the status of the ring buffer, and then runs the OPUS, or ADPCM codec.

Figure 9-9: Ring Buffer Status Check and Encode Audio Data in am_vos_codec_task()

```

AM_CRITICAL_BEGIN_VOS;
ui32StreamLen = am_app_utils_get_ring_buffer_status(&(g_sAmUtil.sRingBuf[AM_AUDIO_BUFFER_MON

AM_CRITICAL_END_VOS;

while(ui32StreamLen)
{
    //
    // Attempt to clear mono buffer
    //
    if(ui32StreamLen > codecInBufRemaining)
    {
        ...

#if configUSE_MSBC_BLUEZ || configUSE_SBC_BLUEZ
        am_vos_codec_encode(&(g_sVosSys.sBluezSBCInstance), p_CodecInBuf,
            CODEC_IN_RING_BUFF_SIZE, p_CodecOutBuf, CODEC_OUT_RING_B
#endif // configUSE_MSBC_BLUEZ || configUSE_SBC_BLUEZ

#if configUSE_OPTIM_OPUS
        am_vos_codec_encode(NULL, p_CodecInBuf, CODEC_IN_RING_BUFF_SIZE, p_CodecOutBuf, CODE
#endif // configUSE_OPTIM_OPUS
        am_audio_buffer_nested_push(AM_AUDIO_BUFFER_ENCODED, p_CodecOutBuf, CODEC_OUT_RING_B

```

9.9 Bluetooth Low Energy TX

- Encoded audio data is collected and sent to the Alexa app or Android TV through BLE when it is at least the designated size.

Figure 9-10: Encoded Audio Data Size Check in am_vos_codec_task()

```

        if(am_app_utils_get_ring_buffer_status(&(g_sAmUtil.sRingBuf[AM_AUDIO_BUFFER_ENCODED]
        {
    #if configUSE_BLE
        am_vos_ble_stream_send();
    #else // configUSE_BLE
        am_vos_audio_flush_ring_buffer();
    #endif // configUSE_BLE
    }

```

- User can disable AMA protocol by defining the below **configUSE_AM-VOS_AMA/configUSE_AMVOS_ATVV** to 0, then the VoS application is working as common GATT profile, and can read/write data using a BLE test app (e.g., LightBlue™).

Figure 9-11: Define Example for AMA Protocol Enabled

```

    #if configUSE_BLE
    #define configUSE_AMVOS_AMA          1      // If AMA,ATVV define all 0, then common GATT confi
    #define configUSE_AMVOS_ATVV        0      // Alexa mobile accessory protocol
                                           // Android TV voice protocol

    #define configUSE_BLE_WATCHDOG      1      // Using watchdog timeout feature to recover connec
    #define configUSE_BLE_SECURE_CONNECTION 1  // Using BLE with secured connection.
    #define configUSE_BLE_BURST_MODE    0      // Enable burst mode at BLE operation.
    #endif // configUSE_BLE

```

SECTION

10

Operation with the Alexa App

10.1 EVB LED Indication

Table 10-1: LED Indications of System Status

Operation	LED	Description
Boot up	All LEDs	LEDs swirl twice
Advertising (not connected to App) or BLE disabled	LED D5	Blinks twice every second.
Connected with App	LED D5	Blinks once every second.
Keyword detected	All LEDs	Swirls one time
CMD phrase detected	LED pattern indicates each command.	e.g., D5 "Louder", D6 "Pause Music"...
RTT recording started	LED D5	Blinks every 300 ms.

10.2 Building VoS Firmware Binary for AMA Protocol

Use the following procedure to build VoS firmware binary for AMA protocol:

1. Configure defines for Alexa app support. (refer to Table 6-4 on page 20)
 - a. **configUSE_BLE, configUSE_AUDIO_CODEEC, configUSE_AM-VOS_AMA, configUSE_PREROLL** to 1.
 - b. Codec selection: OPUS (**configUSE_OPTIM_OPUS**). OPUS codec has better audio quality that is recommended by Amazon.
2. Build and download image to EVB, then reset the board.

10.3 iOS App Installation

Download the Alexa app from the Apple App Store by searching for **Alexa**:
<https://itunes.apple.com/us/app/amazon-alexa/id944011620?mt=8>

NOTE: This will require a US Apple Store account.

10.4 Android App Installation

Use the following procedure to install the Alexa app from Google Play:

1. Download the Alexa app from Google Play Store by searching for **Alexa**:
<https://play.google.com/store/apps/details?id=com.amazon.dee.app&hl=en>

NOTE: This will require a US Google Play Store account.

2. Connect KWD device with mobile phone app.

10.5 Connecting VoS EVB Device with Mobile Phone App

Use the following procedure to connect a VoS EVB device with mobile phone app:

1. With the device programmed and powered up, open the **Amazon Alexa App**.
2. Tap **Device** icon on bottom of screen and complete the following:
 - a. Tap + icon on top right of screen.
 - b. Select **Add Device**, and then select **Choose Headphones**.
3. Connect to **VoS-xx-AMA-xxxx**.

NOTES:

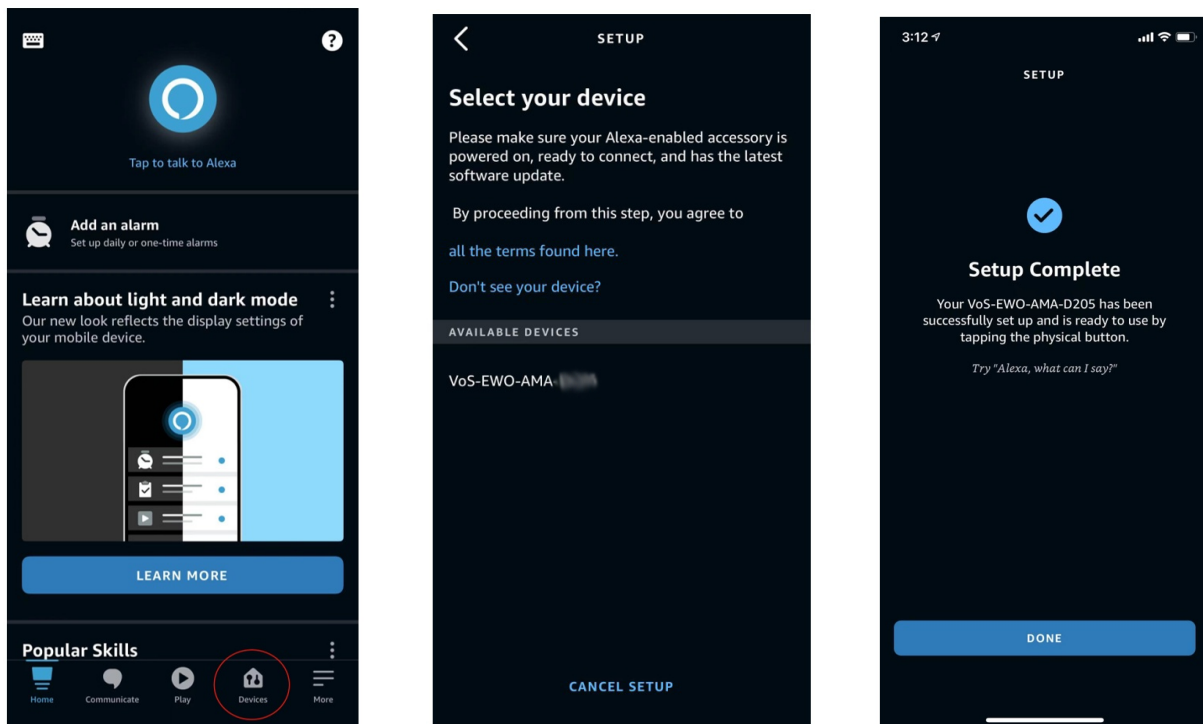
- The last four digits are the last four characters of the MAC address of the device.
- With the board BLE connection complete, the screen will change to display **Setup Complete** confirming connection to AVS.
- If you did not login to Alexa service, you need to login at this time.
- Alexa may not always work in China, so you may need to enable VPN to use this service if you have any network issues.

4. Say "Alexa, what's the weather?"

NOTES:

- The screen will display the activity of the app that transfers data to the AVS. The app will then respond with the weather forecast. You can ask Alexa for any information, and the full host of queries offered by AVS.
- If using DSPC only version, or Light version, use the **Push to Talk** option (default is BTN1 on EVB) to trigger Alexa AI.

Figure 10-1: Setting up Device in Alexa App



SECTION

11

Operation with Google ATVV

11.1 Building VoS Firmware Binary for ATVV Protocol

Use the following procedure to build VoS firmware binary for ATVV (Android TV Voice Service) protocol:

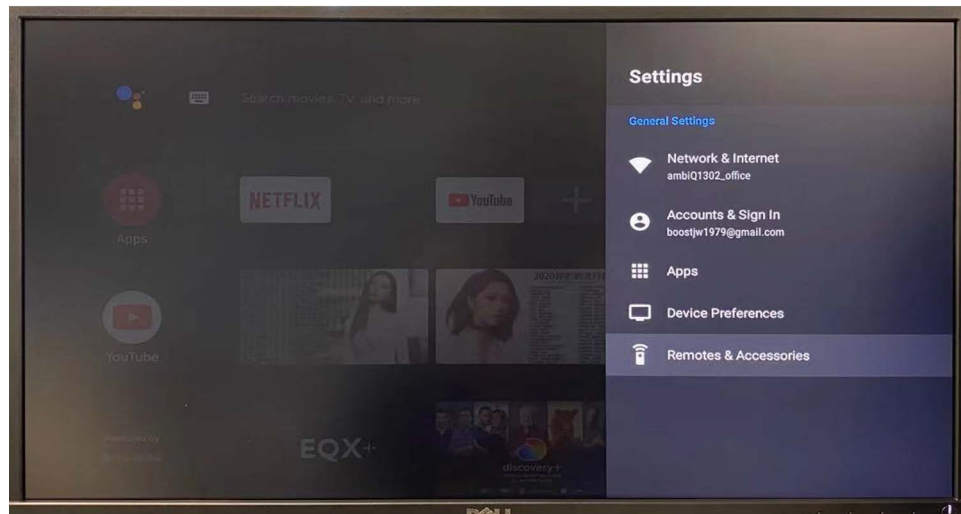
1. Configure the following defines for ATVV support to **1** (refer to Table 6-3 on page 20):
 - **configUSE_BLE**
 - **configUSE_AMVOS_ATVV**
 - **configUSE_AUDIO_CODEC**
 - **configUSE_WW_Google**
 - **configUSE_AMVOS_HID**
 - **configUSE_ADPCM**
2. Build and download image to EVB, then reset the board.

11.2 Connect VoS EVB Device with Android TV (or Set Top Box)

Use the following procedure to connect VoS EVB device with Android TV:

1. With the device programmed and powered up, turn on the Android TV.
2. Go to **Settings**, then select **Remotes & Accessories**.

Figure 11-1: Android TV Settings Menu

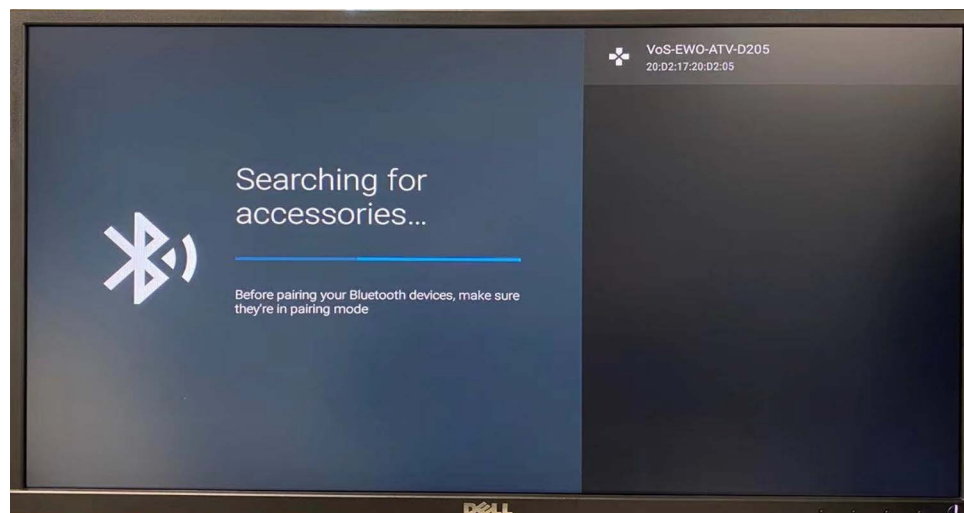


3. Select **Add accessories**, then select **Connect to VoS-Exx-ATV-xxxx**.

NOTES:

- The last four digits are the last four characters of the MAC address of the device.
- With the board BLE connection complete, the **VoS EVB** should now be available in the **Remotes & Accessories** list.
- The VoS EVB should now be connected to Google Assistant.

Figure 11-2: Searching Result in Android TV



4. Say "OK Google, what's the weather?"

NOTE: The screen will display the activity of the app that transfers voice command to GVA. The TV will respond with the weather forecast. Ask Google Assistant for any information, and the full host of queries offered by GVA.

Figure 11-3: Google Voice Assistant Response Screen



SECTION

12

AMVoS Profile with Common BLE Test App

12.1 Build VoS Firmware

Define `configUSE_AMVOS_AMA` and `configUSE_AMVOS_ATVV` to 0.

Figure 12-1: Building VoS without AMA/ATVV Protocol

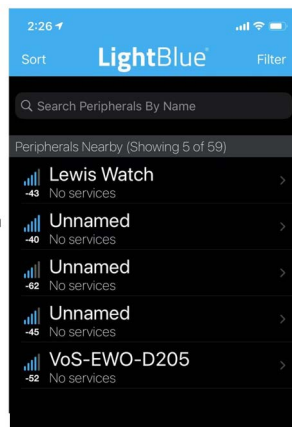
```
#if configUSE_BLE // If AMA,ATVV define all 0, then common GATT conf
#define configUSE_AMVOS_AMA 0 // Alexa mobile accessory protocol
#define configUSE_AMVOS_ATVV 0 // Android TV voice protocol

#define configUSE_BLE_WATCHDOG 1 // Using watchdog timeout feature to recover connec
#define configUSE_BLE_SECURE_CONNECTION 1 // Using BLE with secured connection.
#define configUSE_BLE_BURST_MODE 0 // Enable burst mode at BLE operation.
#endif // configUSE_BLE
```

12.2 Download and Run General BLE Test App

- Recommended app is LightBlue™
- Run LightBlue and connect **VoS-E**-xxxx** device.

Figure 12-2: LightBlue Explorer

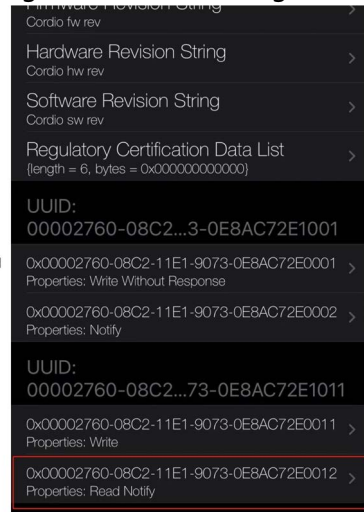


12.3 Streaming Audio Data Through BLE

Use the following procedure to stream audio data through BLE:

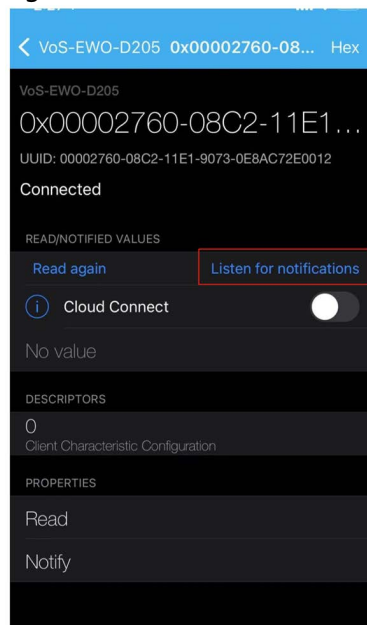
1. Select the UUID ending with **2E0012** and has **Read Notify** properties.

Figure 12-3: UUID in LightBlue Properties Page



2. Click **Listen for notifications**.

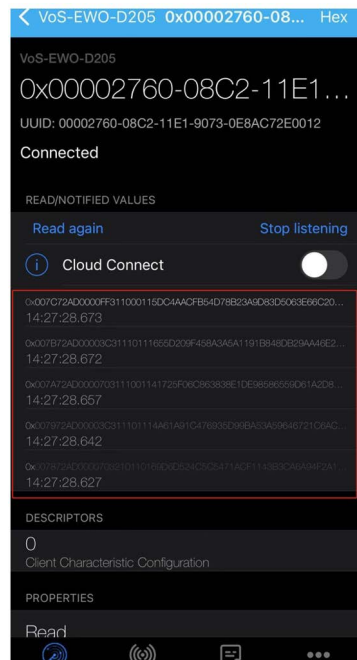
Figure 12-4: Listen for Notifications in LightBlue



3. Send an audio data using a key word trigger (e.g., saying "Alexa, ...") or use the BTN1 (push to talk) option. Then you can see streaming data.

NOTE: You should now see streaming data.

Figure 12-5: Streaming Data in LightBlue



NOTE: To get and verify this dumped data, use **Log** function of this app.

SECTION

13

Sensory VoiceHub

Sensory VoiceHub is a customized wake word and voice command solution. VoiceHub can generate a Sensory THF detection model file, and can be used at the VoS SDK.

13.1 Wake Word + Voice Command

- Voice command is followed by specific keyword.
- For example: "Hey, Ambiq. Kitchen lights on"

13.2 Wake Word Only

- Only wake word is detected, command or user utterance is transferred to AI assistant cloud (e.g., Alexa, Google voice assistant).
- For example: "Alexa, What's the weather?"

13.3 Voice Command Only

- Voice command detected by Sensory engine (without wake word).
- For example: "Kitchen lights on", "All lights on"

13.4 VoiceHub Voice Detection Model in VoS

NOTE: To see an example VoiceHub project, open:
vos_thf, vos_ble_thf, vos_talkto_thf, vos_ble_talkto_thf examples.

Use the following to configure the VoiceHub voice detection model in VoS:

1. Use the following procedure to define the configuration:
 - a. Enable the following:
 - **configUSE_THF_WW** (wake word)
 - **configUSE_THF_CMD** (voice command)
 - b. Choose target wake word / command phrase or add generated model.

NOTE: Figure 13-1 shows an example using "Hey, Ambiq" wake word and light control command phrase.

Figure 13-1: Sensory Module Configuration

```

//*****
// Sensory module configuration
//*****
#if configUSE_Sensory_THF
  #define configUSE_THF_WW      1
  #define configUSE_THF_CMD    1

  #define configUSE_THF_LPSD    0

  #if configUSE_THF_WW
    #define configUSE_WW_Alexa  0    // "Alexa"
    #define configUSE_WW_Google 0    // "Hey, Google"
    #define configUSE_WW_Ambiq 1    // "Hey, Ambiq", "OK, Sensory"
  #endif

  #if configUSE_THF_CMD
    #define configUSE_CMD_VoiceHubDemo 1    // Light control demo.
    #define configUSE_CMD_LightDemoCN 0    // Light control demo at Chinese.

    #define configUSE_THF_Static_Alloc 1    // If CMD phrase is enabled, Static mem allocation is
  #endif // configUSE_THF_CMD
#endif // configUSE_Sensory_THF

```

2. Use the following procedure to add vocabulary model files:
 - a. Place generated voice command vocabulary model files in the **vos\third_party\Sensory\models** folder.
 - b. Include model files in **am_vos_thf.c** file as shown in Figure 13-2 on page 41.

Figure 13-2: Include Voice Detection Model Files at am_vos_thf.c File

```

#if configUSE_WW_Alexa
#include "thfft_alex_a_enus_v3c_dsp_64kb_pc40\thfft_alex_a_enus_v3c_dsp_64kb_search_2_pc40.c"
#include "thfft_alex_a_enus_v3c_dsp_64kb_pc40\thfft_alex_a_enus_v3c_dsp_64kb_am_pc40.c"
#endif

#if configUSE_WW_Google
#include "sensory-thf-enUS-heygoogle-c5d48d402-pc60\sensory-thf-enUS-heygoogle-c5d48d402-pc60-6.0.0-c"
#include "sensory-thf-enUS-heygoogle-c5d48d402-pc60\sensory-thf-enUS-heygoogle-c5d48d402-pc60-6.0.0-c"
#include "sensory-thf-enUS-heygoogle-c5d48d402-pc60\sensory-thf-enUS-heygoogle-c5d48d402-pc60-6.0.0-c"
#endif // configUSE_WW_Google

#if configUSE_WW_Ambiq
// VoiceHub "OK, Sensory", "Hey, Ambiq" Wakeword grammar files
#include "trece_Command_Set_Test_wakeword_pc60\trece_Command_Set_Test_wakeword_pc60_6.3.1-op05-search"
#include "trece_Command_Set_Test_wakeword_pc60\trece_Command_Set_Test_wakeword_pc60_6.3.1-op05-search"
#include "trece_Command_Set_Test_wakeword_pc60\trece_Command_Set_Test_wakeword_pc60_6.3.1-op05-net.c"
#endif

#if configUSE_CMD_VoiceHubDemo
// VoiceHub demo commands files (Light control)
#include "trece_Command_Set_Test_command_pc62\trece_Command_Set_Test_command_pc62_6.1.0-op05-search.h"
#include "trece_Command_Set_Test_command_pc62\trece_Command_Set_Test_command_pc62_6.1.0-op05-search.c"
#include "trece_Command_Set_Test_command_pc62\trece_Command_Set_Test_command_pc62_6.1.0-op05-net.c"
#endif // configUSE_CMD_VoiceHubDemo

```

NOTE: For distinguishing the wake word and command vocabulary, change Sensory model's variable name as below when adding search and net files. (vocabulary file)

- **Wake Word:**
gs_grammarLabel -> gs_ww_grammarLabel
dnn_netLabel -> dnn_ww_netLabel
g_grammarLabel_xxx -> **WW**_grammarLabel_xxx (xxx: wake word)
- **Command Phrase:**
gs_grammarLabel -> gs_cmd_grammarLabel
dnn_netLabel -> dnn_cmd_netLabel
g_grammarLabel_yyy -> **CMD**_grammarLabel_yyy (yyy: command phrase).

Figure 13-3: Variable Name Modification of Wake Word Files

```

dnn_t dnn_ww_netLabel[] = {
    20, // 0x0014
    3, // 0x0003
    23173, // 0x5a85
};

// #ifndef __gs_t
// typedef const u16 __gs_t;
// #endif

__gs_t gs_ww_grammarLabel[] = {
    60, // 0x003c
};

extern u32 gs_grammarLabel;
#ifndef NETLABEL
#define NETLABEL
extern u32 dnn_netLabel;
#endif
#define WW_grammarLabel_SILENCE (0)
#define WW_grammarLabel_ok_sensory (1)
#define WW_grammarLabel_hay_ambiq (2)
#define WW_grammarLabel_nota (3)

```

3. Build and run application at the Apollo4 Plus EVB with microphone shield board.

SECTION

14

Building an OTA Image

The OTA feature is supported in the IAR and Keil projects.

14.1 Build OTA Firmware Image

NOTE: The Apollo4 project's OTA feature is enabled by default.

Use the following procedure to build OTA firmware image:

1. Use a Python script located at **tools/apollo4_amota/scripts/** to make two images:
 - Starter (OTA bootloader + binary image)
 - Update (binary image only)

NOTE: Before running the script, make sure that the path listed in the **tools/apollo4_amota/scripts/Makefile** for the bin file created above is correct. It should look like Figure 14-1.

Figure 14-1: OTA Image Generated by Makefile for Apollo4

```
TOOL_CHAIN?=gcc
APOLLO4_BOARD = apollo4p_blue_kbr_evb
UPDATEBIN_APOLLO4_BLUE = ../../boards/$(APOLLO4_BOARD)/examples/vos/vos_ble_lite/$(TOOL_CHAIN)/bin/vos_ble_lite.bin
APPBIN_APOLLO4_BLUE = ../../boards/$(APOLLO4_BOARD)/examples/vos/vos_ble_lite/$(TOOL_CHAIN)/bin/vos_ble_lite.bin

all: $(APPBIN_APOLLO4_BLUE) $(UPDATEBIN_APOLLO4_BLUE) $(UPDATEBIN_APOLLO4_BLUE_ETHERMIND) $(APPBIN_APOLLO4_BLUE_ETHERMIND)
# Apollo4 Cordio
cp $(APPBIN_APOLLO4_BLUE) starter_binary_apollo4_blue.bin
cp -r ../../apollo4b_scripts/oem_tools_pkg* oem_tools_pkg
cp -r ../../apollo4b_scripts/arm_utils* arm_utils
cp ../../apollo4b_scripts/am_defines.py am_defines.py
cp ../../apollo4b_scripts/apollo4b_keys.py apollo4b_keys.py
cp ../../apollo4b_scripts/create_cust_image_blob.py create_cust_image_blob.py
cp ../../apollo4b_scripts/key_table.py key_table.py
cp ../../apollo4b_scripts/sample/keys.ini keys.ini
python create_cust_image_blob.py -c firmware.ini
python ota_binary_converter.py --appbin temp_binary_apollo4_blue.bin -o update_binary_apollo4_blue
@rm -rf temp_binary_apollo4_blue.bin
@rm -rf key_table.py
@rm -rf keys.ini
@rm -rf am_defines.py
@rm -rf create_cust_image_blob.py
@rm -rf oem_tools_pkg
@rm -rf arm_utils
@rm -rf apollo4b_keys.py

$(APPBIN_APOLLO4_BLUE):
$(MAKE) -C ../../boards/$(APOLLO4_BOARD)/examples/vos/vos_ble_lite/$(TOOL_CHAIN) $(MAKECMDGOALS)
```

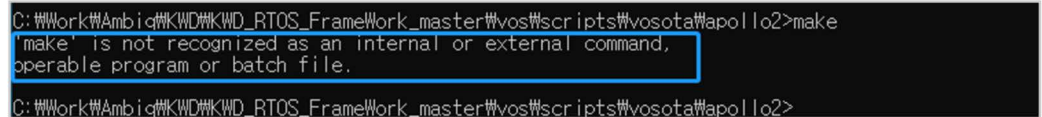
2. Run Make in command prompt, then the two binary files will be generated:
 - **starter_binary_apollo4_blue.bin**: OTA bootloader + 1st firmware image
 - **update_binary_apollo4_blue.bin**: 2nd firmware image that will be used for OTA reprogramming

14.2 Buildtools Installation

If the **make.exe** is not available to use **Makefile**, and the below error message (Figure 14-2) showed up, install **buildtools** for Windows using the following procedure:

1. Download **msys-1.0.7z** package from below link.
 - **Box Cloud**:
<https://app.box.com/s/96crrw2muzudqazae87bk1o0kjin4gu7v>
 - **Baidu Cloud** (for China):
<https://pan.baidu.com/s/1cSRxlzvCayd1pPRgXcpLrg> (pwd: tyew)
2. Extract and add bin folder (e.g., C:\DevUtils\msys\1.0\bin) to the system path.

Figure 14-2: Error Message Due to Non-existing make.exe



```
C:\Work#\Ambic#\KWD#\KWD_RTOS_Framework_master#\vos#\scripts#\vosota#\apollo2>make
'make' is not recognized as an internal or external command,
operable program or batch file.
C:\Work#\Ambic#\KWD#\KWD_RTOS_Framework_master#\vos#\scripts#\vosota#\apollo2>
```

14.3 Python Installation

Python 3.6 or later needs to be installed. The **pycryptodome** package is required to build the Apollo4 OTA image.

Use the following procedure to install pycryptodome:

- Install with **pip install pycryptodome** command.
- If you have any pre-installed crypto related package, uninstall those first.

```
pip uninstall crypto
pip uninstall pycrypto
```

SECTION

15

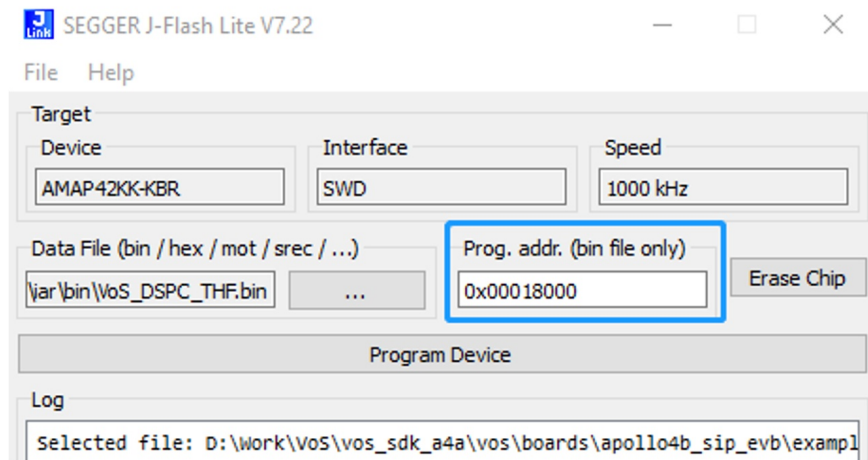
OTA Image Download

15.1 Download Firmware to the EVB

- If the EVB has not been updated with the OTA supported image, the MCU must be programmed with a binary image (.bin) file that contains both the application code, and the code that enables OTA reprogramming. The previous section showed that and, for this demo, that bin file is called **starter_binary_apollo4_blue.bin**.
- Start OTA binary can be programmed using the Segger J-Flash Lite programming utility. It needs to assign **Prog. addr.** to 0x00018000.

NOTE: After installing the latest J-Link S/W (v6.47d or later), this address is updated automatically if the Apollo4 Plus (AMAP42KK) is selected.

Figure 15-1: J-Flash Lite Address Setting for the Apollo4 Plus



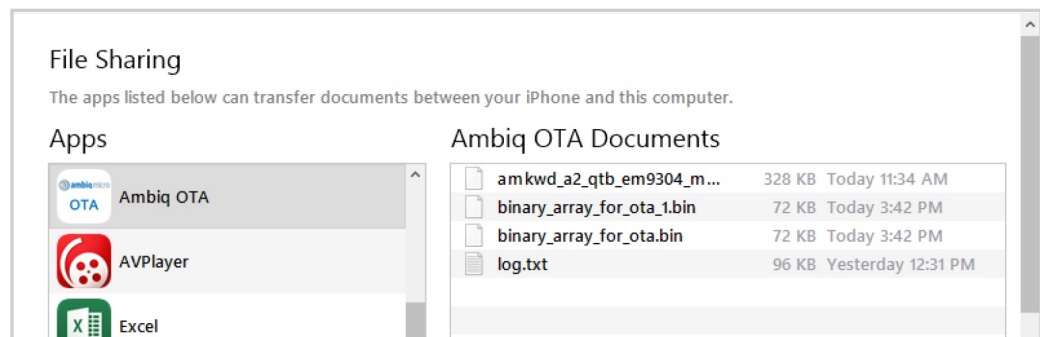
15.2 Update the EVB Firmware Via OTA

The EVB must already have a pre-programmed OTA upgradable image as described above.

Use the following procedure to update the EVB firmware via OTA:

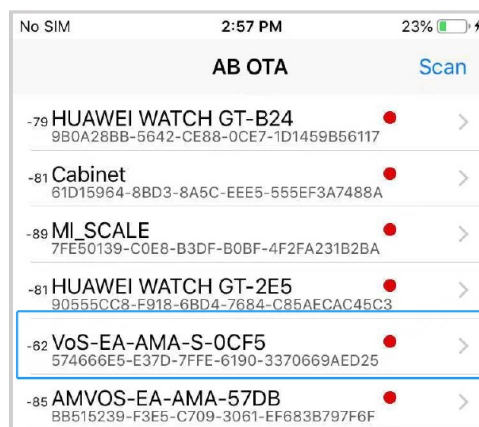
1. Install the Ambiq OTA app from the Apple store by searching "Ambiq":
<https://apps.apple.com/kr/app/ambiq-ota-update/id1506027473>
2. Add KWD "update" image file (**update_binary_apollo4_blue.bin**) to Ambiq OTA app's internal storage using iTunes (or iTools).
 - a. Connect the phone to the PC.
 - b. Open iTunes, then click the phone icon under the top menu.
 - c. Select **File Sharing**, then select **Ambiq OTA** app from the app list.
 - d. Drag and drop the **update_binary_apollo4_blue.bin** file to the **Documents** box.

Figure 15-2: iTunes File Manager



3. Open **Ambiq OTA** on the iPhone, and skip the short tutorial.
4. Wait for connections to show, and select the EVB (e.g., VoS-xx-AMA-xxxx).

Figure 15-3: Ambiq OTA App Scan Result



5. Select **Load bin file**, and choose the update image file (e.g., **update_binary_apollo4_blue.bin**).
6. Select **Send to device**. This starts firmware update via OTA.

NOTE: This starts firmware update via OTA.

7. Close the app after the update.

15.3 OTA App for Android

- Ambiq OTA **apk** file is provided in the AmbiqSuite SDK.
- File path: AmbiqSuite-KWD\tools\amota**Application-debug.apk**

SECTION

16

Audio Data Recording Using RTT

The VoS SDK supports RTT (J-Link) audio recording.

16.1 Build Image for Audio Data Recording

Use the following procedure to build image for audio data recording:

1. Complete one of the following:
 - For RTT recording, set **configUSE_RTT_RECORDER** to 1.

Figure 16-1: RTT Recorder Definition

```
#define configUSE_SYSVIEWER      0
#define configUSE_SYS_LOG        0
#define configUSE_RTT_RECORDER   1
#define configUSE_AMU2S_RECORDER 0
#define configUSE_STDIO_PRINTF   0
```

2. Select dump data type (e.g., RAW data or SPP processed data).

Figure 16-2: Recorder Data Select

```
*****
#if configUSE_RTT_RECORDER
  #define configUSE_RECORD_RAW_PCM      1      // Select which data be recorded
  #define configUSE_RECORD_FULL_FILTER  0
```

3. Set **configUSE_BLE** and **configUSE_AUDIO_CODEC** to 0.

Figure 16-3: Disable Codec and BLE When Using RTT Recorder

```
#define configUSE_BLE      0
#define configUSE_AUDIO_CODEC 0
#define configUSE_LEDs    1
```

4. Build and download image to board.

16.2 Install Software on PC

16.2.1 Install J-Link v6.47x or Later

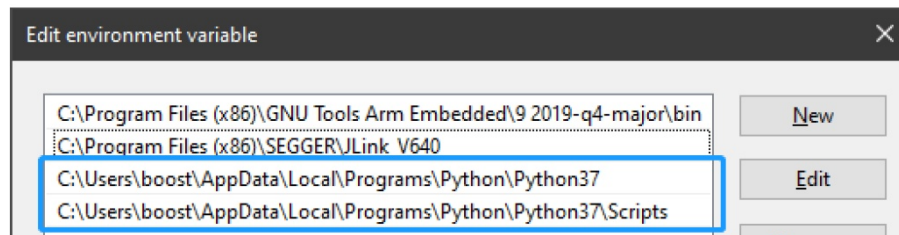
Use the following procedure to install J-Link v6.47x or later:

1. Install J-Link v6.47x or later:
<https://www.segger.com/downloads/jlink/#J-LinkSoftwareAndDocumentationPack>
2. Add J-Link directory to the system path variable by completing the following:
 - a. Click **Start**, then type **env**.
 - b. Select **Edit system environment variable**, and then select **Environment Variable**.
 - c. Add path directory of J-Link to current path variable.

16.2.2 Install Python 3.6 or Later

- Add the python directory path to the system path variable.
- Install necessary modules (e.g., docopt, soundfile, numpy, pandas) using the command shell: -> **pip install docopt soundfile numpy pandas**

Figure 16-4: System Path Environment Variable



16.2.3 RTT PCM Recorder Script File Package

- **vos_audio_recorder_200714.7z**: [Box Cloud](#)
- **vos_audio_recorder_200714.7z**: [Baidu Cloud \(for China\)](#) pw: 5brv

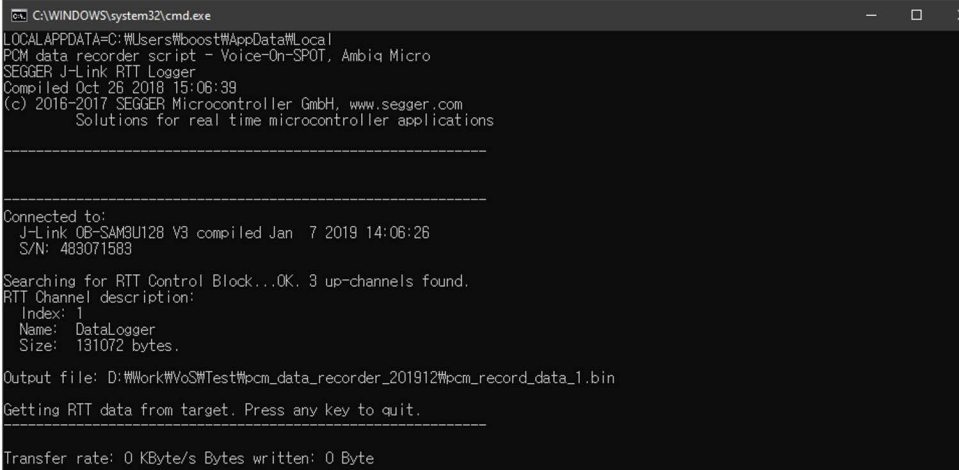
16.3 Audio (Voice) Data Recording

Use the following procedure to record audio data:

1. Connect the board (VoS firmware programmed with above way) to PC. Or reset the board after being programmed.

2. Complete the appropriate recording option:
 - RTT recording: run **vos_rtt_recorder.bat**

Figure 16-5: Run RTT datalogger batch file



```
CA\WINDOWS\system32\cmd.exe
LOCALAPPDATA=C:\Users\boost\AppData\Local
PCM data recorder script - Voice-On-SPOT, Ambiq Micro
SEGGER J-Link RTT Logger
Compiled Oct 26 2018 15:06:39
(c) 2016-2017 SEGGER Microcontroller GmbH, www.segger.com
Solutions for real time microcontroller applications

-----

Connected to:
J-Link OB-SAM9L128 V3 compiled Jan 7 2019 14:06:26
S/N: 489071583

-----

Searching for RTT Control Block...OK, 3 up-channels found.
RTT Channel description:
Index: 1
Name: DataLogger
Size: 131072 bytes.

Output file: D:\Work\VoS\Test\pcm_data_recorder_201912\pcm_record_data_1.bin
Getting RTT data from target. Press any key to quit.

-----

Transfer rate: 0 KByte/s Bytes written: 0 Byte
```

3. Press **BTNO** on the Apollo4 Plus EVB to start recording.
4. Press any key to stop recording.

NOTE: This generated PCM format file named **pcm_record_data_x.bin** (RTT).

16.4 Convert Raw Data to WAV File Format

Use the following procedure to convert raw data to wav file format:

- Run **pcm_to_wav.py** to convert it to WAV file format.
 - RTT: Command shell
 - > `python bin_to_wav.py pcm -i pcm_record_data_1.bin`

NOTE: This generates 2 types of wav files (e.g., before and after normalization).

SECTION

17

Debug Message Output

17.1 UART

Use the following procedure for UART debugging:

1. Define **configUSE_STDIO_PRINTF** and **configUSE_PRINTF_UART0** to 1.

Figure 17-1: configUSE_STDIO_PRINTF definition in am_vos_sys_config_h

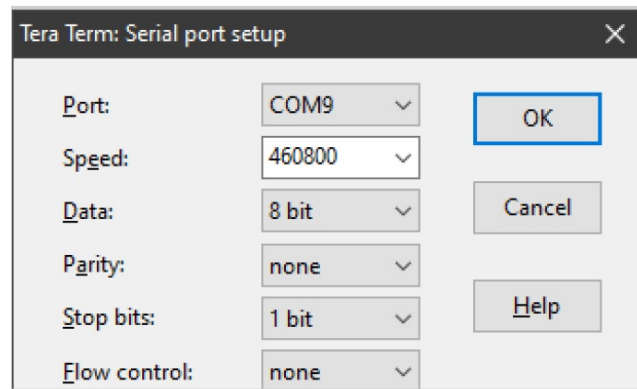
```
#define configUSE_SYSVIEWER           0
#define configUSE_SYS_LOG             0
#define configUSE_RTT_RECORDER        0
#define configUSE_STDIO_PRINTF        1
```

Figure 17-2: configUSE_PRINTF_UART0 definition in am_vos_sys_config_h

```
/*
#define configUSE_PRINTF_UART0         1
#define configUSE_PRINTF_RTT          0
#define configUSE_PRINTF_SWO          0
```

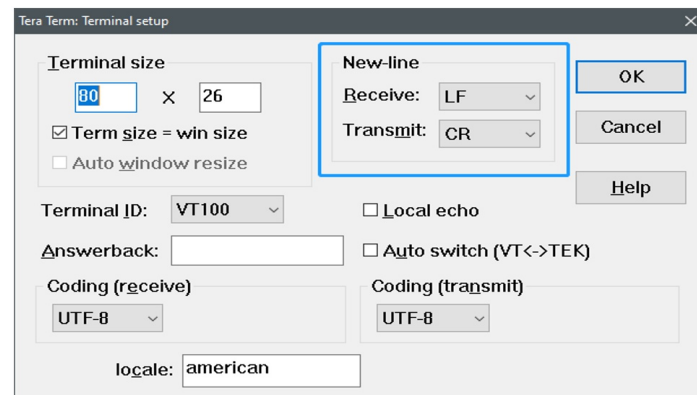
2. Complete the Serial port setup window as shown below:
 - a. Select **460800** in the **Speed** drop-down option.
 - b. Select **8 bit** in the **Data** drop-down option.
 - c. Select **none** in the **Parity** drop-down option.
 - d. Select **1 bit** in the **Stop bits** drop-down option.
 - e. Select **none** in the **Flow control** drop-down option.
 - f. Click **OK**.

Figure 17-3: Serial Port Setup



3. Complete the following **New-line** section in the Terminal setup window:
 - a. Select **LF** in the **Receive** drop-down option.
 - b. Select **CR** in the **Transmit** drop-down option.
 - c. Click **OK**.

Figure 17-4: Terminal Setup



UART debugging log is shown in Figure 17-5.

Figure 17-5: UART print out using Tera Term

```

VT COM5 - Tera Term VT
File Edit Setup Control Window Help
connInterval = 12 x 1.25 ms
connLatency = 0
supTimeout = 4000 ms
ATT_MTU_UPDATE_IND AttGetMtu(), return = 247 pMsg->att.mtu = 247
ccc state ind value:2 handle:19 idx:0
ccc state ind value:1 handle:2053 idx:1
connId : 1

[AMA] Cmd GET_DEVICE_INFORMATION recv
[AMA Callback] VOS_AMA_EVT_GET_DEV_INFO
  
```

17.2 SWO Output

Use the following procedure for SWO debugging:

1. Define **configUSE_STUDIO_PRINTF** and **configUSE_PRINTF_SWO** to 1.

Figure 17-6: configUSE_STUDIO_PRINTF definition in am_vos_sys_config.h

```

//*****
// System level functional module selection
//*****
#define configUSE_SYSVIEWER      0
#define configUSE_SYS_LOG       0
#define configUSE_RTT_RECORDER  0
#define configUSE_STUDIO_PRINTF 1

```

Figure 17-7: configUSE_PRINTF_SWO definition in am_vos_sys_config.h

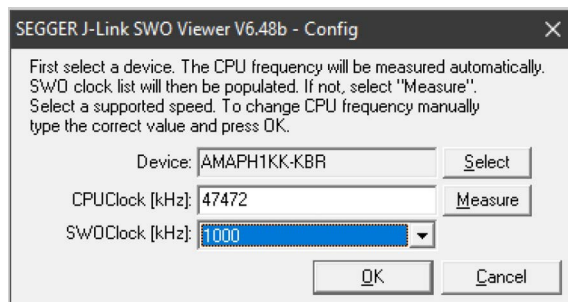
```

//*****
// std IO sub module configuration
//*****
#if configUSE_STUDIO_PRINTF
#define configUSE_PRINTF_UART0    0
#define configUSE_PRINTF_RTT     0
#define configUSE_PRINTF_SWO     1

```

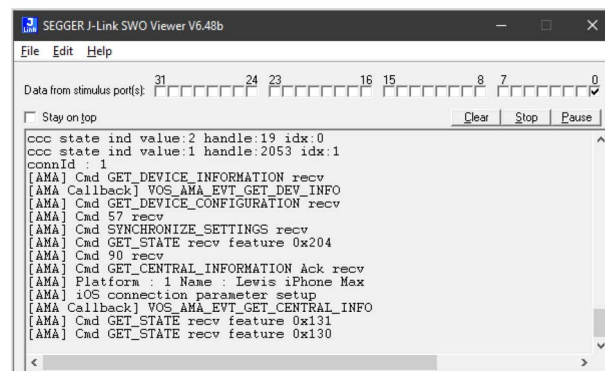
2. Run J-Link SWO Viewer and complete the following:
 - a. Select the MCU type (e.g., Apollo4 Plus: AMAP42KK-KBR) in the **Device** box.
 - b. Click **Measure**.
 - c. Select **1000** in the **SWO Clock [kHz]** drop-down option.
 - d. Click **OK**.

Figure 17-8: SWO device and clock configuration



SWO debugging log message is shown in Figure 17-9.

Figure 17-9: SWO debugging message print out





© 2023 Ambiq Micro, Inc. All rights reserved.

6500 River Place Boulevard, Building 7, Suite 200, Austin, TX 78730

www.ambiq.com

sales@ambiq.com

+1 (512) 879-2850

A-SOCAP4-UGGA01EN v1.3

March 2023