



## USER'S GUIDE

# Apollo3 Blue Plus Voice-on-SPOT Kit

Ultra-Low Power Apollo SoC Family

A-SOCA3P-UGNA01EN v1.4

**(Voice-on SPOT is intended for demo purposes only)**



## 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	May 7, 2021	Initial release
1.1	December 6, 2022	Updated Section 2 Access and Licensing and other third party related information.
1.2	March 7, 2023	Removed Vesper, VM301x, VM10xx, WoS (Wake on Sound) related contents. Changed "AWE (or Audio Weaver)" words to "TalkTo". Update example and path to following latest VoS SDK structure.
1.3	August 1, 2023	<ul style="list-style-type: none"> <li>▪ Replaced "AmbiqSuite SDK v3.0.0" with "AmbiqSuite SDK v3.1.1"</li> <li>▪ Updated Table 5-1 Project Directory Structure</li> <li>▪ Replaced "vos_ble_light" with "vos_ble_lite"</li> <li>▪ Updated section 6.1 Top-Level Feature of the VoS SDK</li> </ul>
1.4	July 21, 2024	<ul style="list-style-type: none"> <li>▪ Removed DSP information (global)</li> <li>▪ Updated Arm version from 8.32.2 to 9.40.2 in section 3</li> <li>▪ Added new TDK MIC MikroBUS Board (TDK T5838) section</li> <li>▪ Updated Table 5.1 Project Directory Structure</li> <li>▪ Updated Figure 9-1 VoS THF Flow Chart</li> <li>▪ Added new section 17 Power Consumption Profile</li> <li>▪ Added new section 18 CPU Usage Profile</li> </ul>

# Table of Contents

<b>1. Introduction</b> .....	<b>10</b>
<b>2. Access and Licensing</b> .....	<b>11</b>
2.1 Production IP Licensing .....	11
2.2 Support Contact Information .....	11
<b>3. IDE Support</b> .....	<b>12</b>
<b>4. Supported Hardware</b> .....	<b>13</b>
4.1 Apollo3 Blue Plus EVB .....	13
4.2 MikroBUS Shield Board (AM_MBS_R20) .....	14
4.3 Memsensing MIC MikroBUS Board (AM-STD-CL) .....	14
4.4 TDK MIC MikroBUS Board (TDK T5838) .....	15
<b>5. Project Directory Structure</b> .....	<b>16</b>
<b>6. Project Configuration to Select Target Board</b> .....	<b>17</b>
6.1 Top-Level Feature of the VoS SDK .....	17
6.2 Audio Signal Chain and Key Word Detection Library Selection in the Configuration File .....	18
6.3 Feature Selection in the Configuration File .....	18
6.4 Bluetooth Low Energy Protocol and Codec Selection in the Configuration File .....	19
6.5 Audio Feature Configuration in the am_vos_audio.h File .....	19
<b>7. Building the SDK to Create a Custom Binary for Standalone Mode</b> ..	<b>20</b>
<b>8. VoS SDK Tuning Guide</b> .....	<b>21</b>
8.1 VoS/THF Flow Chart .....	21
8.2 VoS Lite Flow Chart .....	22
8.3 Voice Activity Detection (VAD) .....	22
8.3.1 Ambiq VAD .....	22
8.4 PDM Interface .....	22
8.5 Stereo to Mono (without SPP) .....	23

8.6 Key Word (Command Phrase) Engine .....	23
8.7 Audio Codec Encoder (OPUS or ADPCM) .....	24
8.8 Bluetooth Low Energy TX .....	25
<b>9. Operation with the Alexa App .....</b>	<b>26</b>
9.1 EVB LED Indication .....	26
9.2 Building VoS Firmware Binary for AMA Protocol .....	26
9.3 iOS App Installation .....	27
9.4 Android App Installation .....	27
9.5 Connecting VoS EVB Device with Mobile Phone App .....	27
<b>10. Operation with Google ATVV .....</b>	<b>29</b>
10.1 Building VoS Firmware Binary for ATVV Protocol .....	29
10.2 Connect VoS EVB Device with Android TV (or Set Top Box) .....	30
<b>11. AMVoS Profile with Common Bluetooth Low Energy Test App .....</b>	<b>32</b>
11.1 Build VoS Firmware .....	32
11.2 Download and Run General Bluetooth Low Energy Test App .....	32
11.3 Streaming Audio Data Through Bluetooth Low Energy .....	33
<b>12. Sensory VoiceHub .....</b>	<b>35</b>
12.1 Wake Word + Voice Command .....	35
12.2 Wake Word Only .....	35
12.3 Voice Command Only .....	35
12.4 VoiceHub Voice Detection Model in VoS .....	36
<b>13. Building an OTA Image .....</b>	<b>38</b>
13.1 Build OTA Firmware Image .....	38
13.2 Buildtools Installation .....	39
13.3 Python Installation .....	39
<b>14. OTA Image Download .....</b>	<b>40</b>
14.1 Download Firmware to the EVB .....	40
14.2 Update the EVB Firmware Via OTA .....	41
14.3 OTA App for Android .....	42

---

<b>15. Audio Data Recording Using RTT or AMU2S .....</b>	<b>43</b>
15.1 Build Image for Audio Data Recording .....	43
15.2 Install Software on PC .....	44
15.2.1 Install J-Link v6.47x or Later .....	44
15.2.2 Install Python 3.6 or Later .....	44
15.2.3 RTT and AMU2S PCM Recorder Script File Package .....	44
15.3 Audio (Voice) Data Recording .....	45
15.4 Convert Raw Data to WAV File Format .....	46
<b>16. Debug Message Output .....</b>	<b>47</b>
16.1 UART .....	47
16.2 SWO Output .....	49
<b>17. Power Consumption Profile .....</b>	<b>50</b>
17.1 Hardware Setup .....	50
17.2 Software Setup .....	50
17.3 Power Measurement .....	50
<b>18. CPU Usage Profile .....</b>	<b>51</b>
18.1 CPU Usage .....	51
18.2 Software Setup .....	51
18.3 Printing Result of CPU Utilization .....	52

## List of Tables

Table 2-1 Support Contact Information .....	11
Table 5-1 Project Directory Structure .....	16
Table 6-1 Top-Level Feature of the VoS SDK .....	17
Table 6-2 Feature Selection in the Configuration File .....	18
Table 6-3 Bluetooth Low Energy Service/Protocol Selection in Configuration File .....	19
Table 6-4 Feature Configuration in the am_vos_audio.h File .....	19
Table 9-1 LED Indications of System Status .....	26
Table 17-1 Power/CPU Profiling Cases Define Settings .....	50

## List of Figures

Figure 4-1 Apollo3 Blue Plus EVB .....	13
Figure 4-2 MikroBUS Shield Board with 3 Slots .....	14
Figure 4-3 Top-Mount Microphone MikroBUS Board .....	14
Figure 4-4 Top-Mount Microphone MikroBUS Board .....	15
Figure 6-1 Audio Chain and Detection Engine Selection .....	18
Figure 8-1 VoS/THF Flow Chart .....	21
Figure 8-2 VoS Lite Flow Chart .....	22
Figure 8-3 am_pdm0_isr() in am_vos_isr.c File .....	23
Figure 8-4 am_vos_stereo_to_mono_proc() in am_vos_audio.c File .....	23
Figure 8-5 Keyword Detection Check in am_vos_engine_process() .....	24
Figure 8-6 Ring Buffer Status Check and Encode Audio Data in am_vos_codec_task() .....	24
Figure 8-7 Encoded Audio Data Size Check in am_vos_codec_task() .....	25
Figure 8-8 Define Example for AMA Protocol Enabled .....	25
Figure 9-1 Setting up Device in Alexa App .....	28
Figure 10-1 Android TV Settings Menu .....	30
Figure 10-2 Searching Result in Android TV .....	30
Figure 10-3 Google Voice Assistant Response Screen .....	31
Figure 11-1 Building VoS without AMA/ATV Protocol .....	32
Figure 11-2 LightBlue Explorer .....	32
Figure 11-3 UUID in LightBlue Properties Page .....	33
Figure 11-4 Listen for Notifications in LightBlue .....	33
Figure 11-5 Streaming Data in LightBlue .....	34
Figure 12-1 Sensory Module Configuration .....	36
Figure 12-2 Include Voice Detection Model Files at am_vos_thf.c File .....	37
Figure 12-3 Variable Name Modification of Wake Word Files .....	37
Figure 13-1 OTA Image Generated by Makefile for Apollo3 .....	38
Figure 13-2 Error Message Due to Non-existing make.exe .....	39
Figure 14-1 J-Flash Lite Address Setting for Apollo3 Blue Plus .....	40
Figure 14-2 iTunes File Manager .....	41
Figure 14-3 Ambiq OTA App Scan Result .....	41
Figure 15-1 RTT Recorder Definition .....	43
Figure 15-2 AMU2S Recorder Definition .....	43
Figure 15-3 Recorder Data Select .....	43
Figure 15-4 Disable Codec and Bluetooth Low Energy When Using RTT Recorder .....	44
Figure 15-5 System Path Environment Variable .....	44
Figure 15-6 Run RTT datalogger batch file .....	45
Figure 15-7 Run AMU2S audio recorder .....	45
Figure 16-1 configUSE_STDIO_PRINTF definition in am_vos_sys_config_h .....	47
Figure 16-2 configUSE_PRINTF_UART0 definition in am_vos_sys_config_h .....	47
Figure 16-3 Serial Port Setup .....	48
Figure 16-4 Terminal Setup .....	48
Figure 16-5 UART print out using Tera Term .....	48
Figure 16-6 configUSE_STDIO_PRINTF definition in am_vos_sys_config.h .....	49



Figure 16-7 configUSE_PRINTF_SWO definition in am_vos_sys_config.h .....	49
Figure 16-8 SWO device and clock configuration .....	49
Figure 16-9 SWO debugging message print out .....	49
Figure 18-1 SWO print enable in am_vos_sys_config.h file .....	51
Figure 18-2 System CPU usage measurement enable in am_vos_sys_config.h .....	51
Figure 18-3 System CPU usage measurement enable in am_vos_sys_config.h .....	52
Figure 18-4 CPU Usage Log Via SWO Debug Print .....	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 Apollo3 Blue Plus SoC. 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 Apollo SoC's stereo PDM 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 Apollo 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 Cortex-M4 core.

SECTION

2

## Access and Licensing

See [Section 6 Project Configuration to Select Target Board on page 17](#) for descriptions of audio processing modules.

### 2.1 Production IP Licensing

DSP Concepts, Sensory, and Ambiq have agreed to a third party software ecosystem licensing model to share IP.

### 2.2 Support Contact Information

Table 2-1: Support Contact Information

Company	Email
<b>Ambiq</b>	support@amiq.com
<b>Sensory</b>	techsupport@sensoryinc.com

SECTION

# 3

## IDE Support

Voice-on-SPOT SDK is developed based on AmbiqSuite SDK v3.1.1.

- **IAR Embedded Workbench IDE**
  - Developed and tested using IAR IDE version Arm 9.40.2
- **Keil uVision IDE**
  - uVision v5.27.1.0

## SECTION

# 4

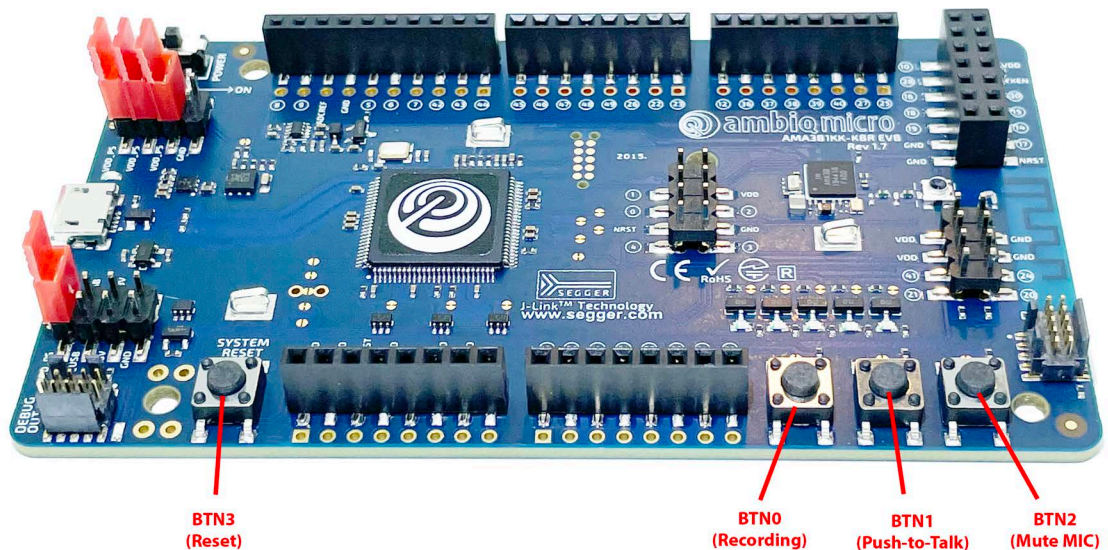
## Supported Hardware

This section describes the supported hardware for Apollo3 Blue Plus VoS.

### 4.1 Apollo3 Blue Plus EVB

- Apollo3 Blue Plus SoC
- Contains sockets for the Shield board

Figure 4-1: Apollo3 Blue Plus EVB

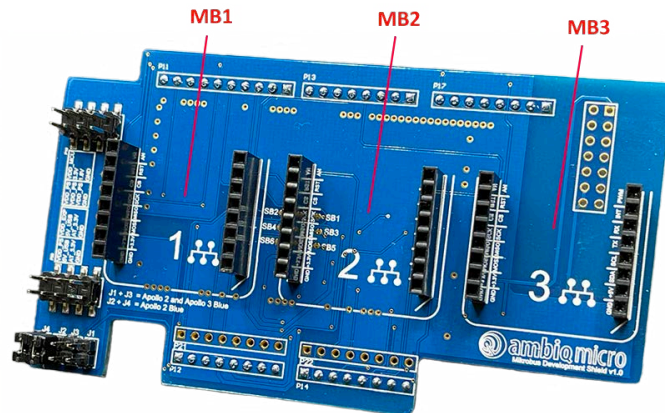


## 4.2 MikroBUS Shield Board (AM\_MBS\_R20)

- MikroBUS Shield Board (3 slots - MB1/MB2/MB3)

**NOTE:** Make sure J1 and J3 jumper on the shield board is shorted to support Apollo3 Blue Plus EVB.

Figure 4-2: MikroBUS Shield Board with 3 Slots



## 4.3 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 with 2cm spacing
- Configuration definition:  
**USE\_DMIC\_MB3**

Figure 4-3: Top-Mount Microphone MikroBUS Board



## 4.4 TDK MIC MikroBUS Board (TDK T5838)

- TDK T5838 digital MICs with selectable 1-mic or 2-mic configuration
- If 2 mics, 0.5cm, 1cm or 2cm spacing depending on SBx (solder bridge) configuration
- Default is 2 bottom-mounted MICs (C1, C5) with 2cm spacing
- Configuration definition: **USE\_DMIC\_MB3** or **USE\_DMIC\_MB3\_T5838**

Figure 4-4: Top-Mount Microphone MikroBUS Board



## 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 – MCU Dependent	ambiq_vos\am_vos_mcu
VoS Code - Sound Pre-processing Algorithms	ambiq_vos\am_vos_spp
VoS Code - Codec	ambiq_vos\codec
VoS Code – BLE Protocols	ambiq_vos\protocol
VoS Project	boards\apollo3p_evb\examples
Third-Party Add-ons (DSPC, Sensory)	third_party\



## SECTION

## 6

# 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
<b>AB</b>	Ambiq	Universal Audio Buffer (Raw, Encoded Audio data buffer)
<b>THF</b>	Sensory	Truly Hands Free Keyword Detection function - "Alexa" THF models are 64K v3c models
<b>OPUS</b>	IETF/Xiph Org	Audio codec that incorporates technology from Skype SILK codec and Xiph.Org's CELT.
<b>mSBC</b>	BlueZ	Modified Sub Band Coding Audio Encoder (Open Source)
<b>ADPCM</b>	SpanDSP	Adaptive differential pulse-code modulation encoder (Open Source)

The SDK consists of 2 base example projects:

- **vos\_ble\_thf** – contains Sensory Truly Hands Free features.
- **vos\_ble\_lite** – contains no keyword detection engine. This version supports manual trigger (e.g., "push to talk") and Bluetooth Low Energy transfer with AMA/ATVV protocol.

Base VoS SDK package (**AmbiqVoS\_R3.x.x.zip**) can build only one example - **vos\_ble\_lite**.

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

- **Sensory THF:** AmbiqVoS\_R3.x.x\_Sensory\_THF.zip

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\apollo3p\_evb\examples\vos\_ble\_xxx\src** folder of the VoS project.

## 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_DSPPC_TalkTo      1      // DSPPC 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 6-2: 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 / RTT / 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 OPUS / ADPCM encoding.	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

Table 6-2: Feature Selection in the Configuration File (*Continued*)

Field in Configuration File	1	0
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 6-3: 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	●	0	●
configUSE_ADPCM	0	1	●

## 6.5 Audio Feature Configuration in the am\_vos\_audio.h File

Table 6-4: 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

## 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.

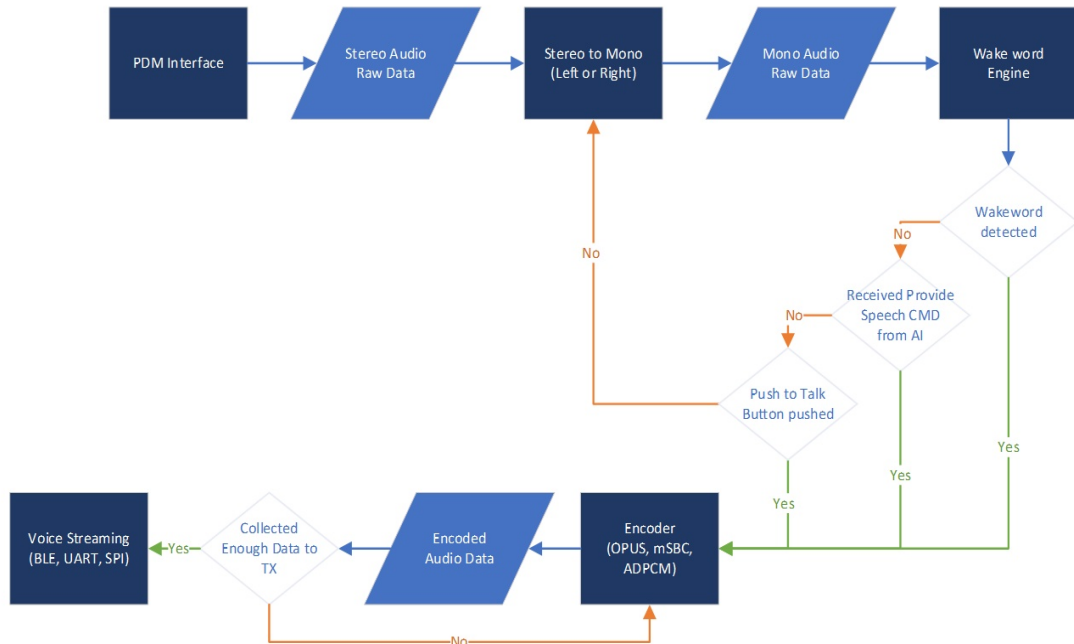
SECTION

8

# VoS SDK Tuning Guide

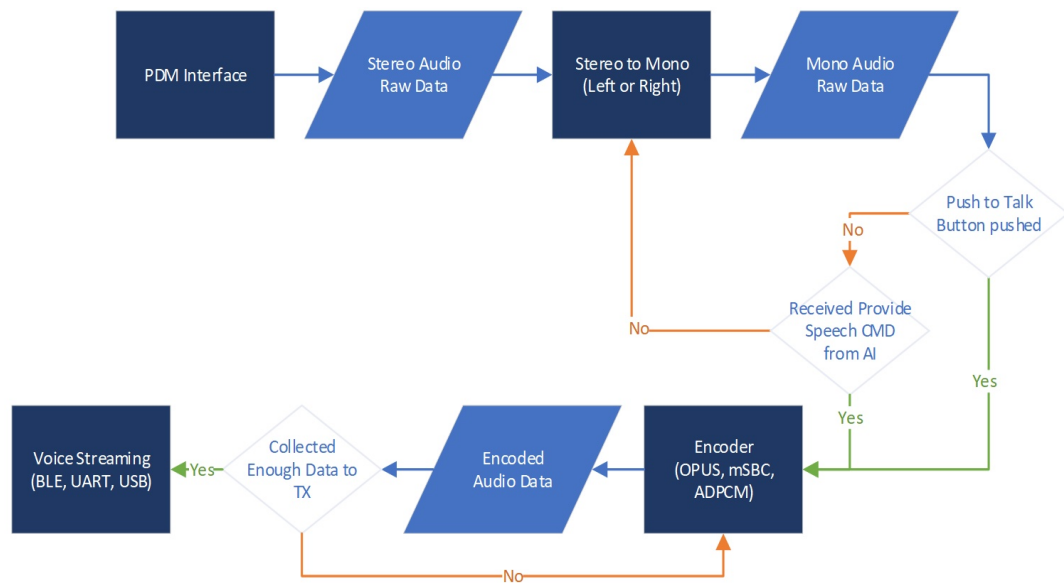
## 8.1 VoS/THF Flow Chart

Figure 8-1: VoS/THF Flow Chart



## 8.2 VoS Lite Flow Chart

Figure 8-2: VoS Lite Flow Chart



## 8.3 Voice Activity Detection (VAD)

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

### 8.3.1 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.

## 8.4 PDM Interface

- **am\_pdm0\_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 8-3: 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 // USE_OWP_DOUBLE_TAP

```

## 8.5 Stereo to Mono (without SPP)

- The **am\_vos\_stereo\_to\_mono\_proc()** function converts stereo data to mono audio data.

Figure 8-4: am\_vos\_stereo\_to\_mono\_proc() in am\_vos\_audio.c File

```

// This process only take 1-channel data into WVE
//
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)

## 8.6 Key Word (Command Phrase) Engine

- VoS supports several third-party keyword and command phrase detection engine.
  - Sensory THF (with VoiceHub): keyword and command phrase
- When mono audio buffer reaches 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 8-5: 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_wv_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.

## 8.7 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 8-6: 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_MONO]));

    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_BUFF_SIZE);
    #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, CODEC_OUT_RING_BUFF_SIZE);
    #endif // configUSE_OPTIM_OPUS

        am_audio_buffer_nested_push(AM_AUDIO_BUFFER_ENCODED, p_CodecOutBuf, CODEC_OUT_RING_BUFF_SIZE);

```



## 8.8 Bluetooth Low Energy TX

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

Figure 8-7: 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\_AMVOS\_AMA/configUSE\_AMVOS\_ATVV** to 0, then the VoS application is working as common GATT profile, and can read/write data using a Bluetooth Low Energy test app (e.g., LightBlue™).

Figure 8-8: 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

## 9

# Operation with the Alexa App

## 9.1 EVB LED Indication

Table 9-1: LED Indications of System Status

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

## 9.2 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 Table 6-3 on page 19)
  - 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.

## 9.3 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 or supported country's Apple Store account.

## 9.4 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 or supported country's Google Play Store account.

2. Connect KWD device with mobile phone app.

## 9.5 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:**

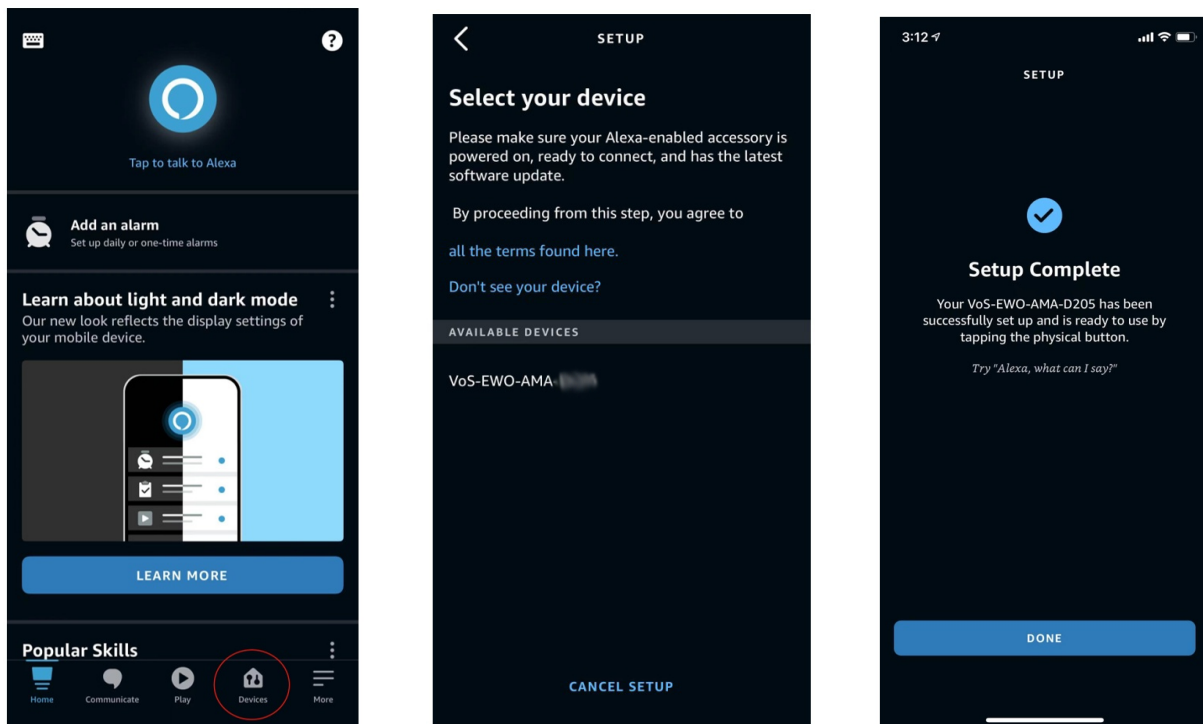
- The last four digits are the last four characters of the MAC address of the device.
- With the board Bluetooth Low Energy connection complete, the screen will change to display **Setup Complete** confirming connection to AVS.
- If you did not login to Alexa service, you need to login at this time.
- 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 Lite version, use the **Push to Talk** option (default is BTN1 on EVB) to trigger Alexa AI.

Figure 9-1: Setting up Device in Alexa App



SECTION

10

## Operation with Google ATV

### 10.1 Building VoS Firmware Binary for ATV Protocol

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

1. Configure the following defines for ATV support to **1** (refer to Table 6-2 on page 18):
  - **configUSE\_BLE**
  - **configUSE\_AMVOS\_ATV**
  - **configUSE\_AUDIO\_CODEC**
  - **configUSE\_WW\_Google**
  - **configUSE\_AMVOS\_HID**
  - **configUSE\_ADPCM**
2. Build and download the image to the EVB, then reset the board.

## 10.2 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 10-1: Android TV Settings Menu

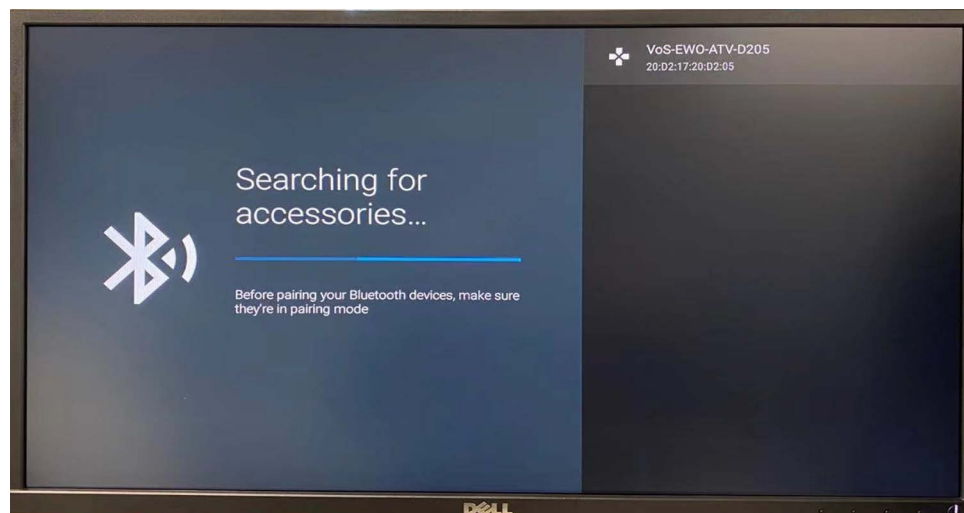


3. Select **Add accessories**, then select **Connect to VoS-Exx-ATV-xxxx**.

### NOTES:

- The last four digits are the last four characters of the MAC address of the device.
- With the board Bluetooth Low Energy 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 10-2: 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 10-3: Google Voice Assistant Response Screen



## SECTION

# 11

## AMVoS Profile with Common Bluetooth Low Energy Test App

### 11.1 Build VoS Firmware

Define `configUSE_AMVOS_AMA` and `configUSE_AMVOS_ATVV` to 0.

Figure 11-1: Building VoS without AMA/ATVV Protocol

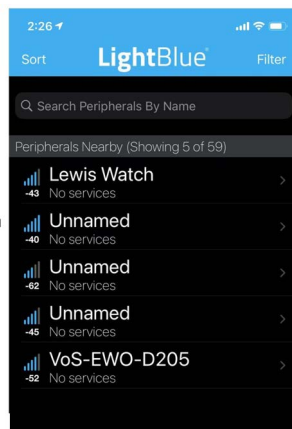
```
#if configUSE_BLE // If AMA,ATVV define all 0, then common GATT conf
#define configUSE_AMVOS_AMA 0 // Alexa mobile accessory protocol
#define configUSE_AMVOS_ATVV 0 // Android TV voice protocol

#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
```

### 11.2 Download and Run General Bluetooth Low Energy Test App

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

Figure 11-2: LightBlue Explorer



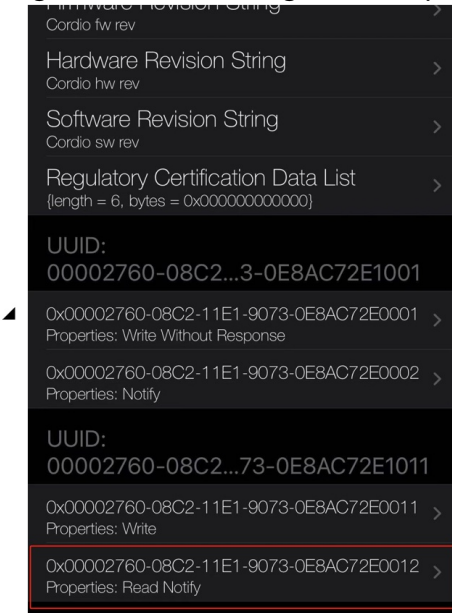


## 11.3 Streaming Audio Data Through Bluetooth Low Energy

Use the following procedure to stream audio data through Bluetooth Low Energy:

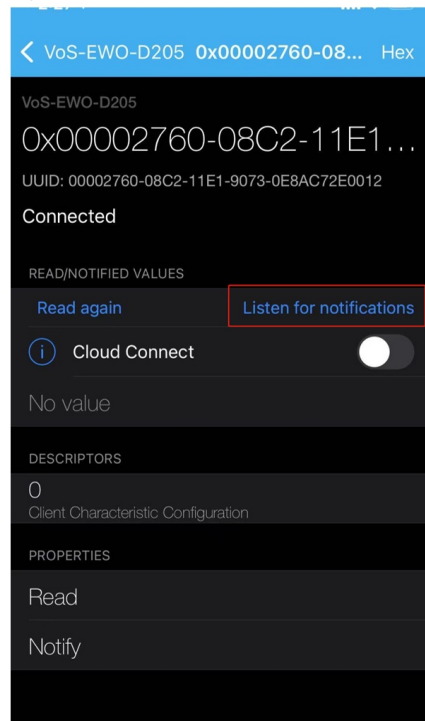
1. Select the UUID ending with **2E0012** and has **Read Notify** properties.

Figure 11-3: UUID in LightBlue Properties Page



2. Click **Listen for notifications**.

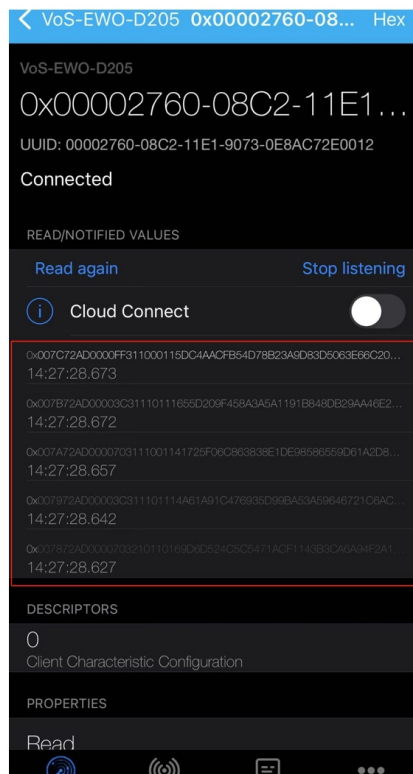
Figure 11-4: Listen for Notifications in LightBlue



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 11-5: Streaming Data in LightBlue



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

SECTION

12

## 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.

### 12.1 Wake Word + Voice Command

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

### 12.2 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?"

### 12.3 Voice Command Only

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

## 12.4 VoiceHub Voice Detection Model in VoS

**NOTE:** To use VoiceHub generate models, open:  
**vos\_ble\_thf, vos\_ble\_talkto\_thf**

Use the following 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 12-1 shows an example using "Hey, Ambiq" wake word and light control command phrase.

Figure 12-1: 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 **third\_party\Sensory\models** folder.
  - b. Include model files in **am\_vos\_thf.c** file as shown in Figure 12-2 on page 37.

Figure 12-2: 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 add 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 12-3: 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 Apollo3 Blue Plus EVB with microphone shield board.

## SECTION

# 13

## Building an OTA Image

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

### 13.1 Build OTA Firmware Image

**NOTE:** Apollo3 Blue Plus 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/apollo3\_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/apollo3\_amota/scripts/Makefile** for the bin file created above is correct. It should look like Figure 13-1.

Figure 13-1: OTA Image Generated by Makefile for Apollo3

```
# Apollo3-BLUE EVB
UPDATEBIN_APOLLO3_BLUE = ../../../../boards/apollo3_evb/examples/ble_freertos_amota/$(TOOL_CHAIN)
/bin/ble_freertos_amota.bin
APPBIN_APOLLO3_BLUE = ../../../../boards/apollo3_evb/examples/ble_freertos_amota/$(TOOL_CHAIN)/bin/ble_freertos_amota.bin

: $(APPBIN_APOLLO3_BLUE) $(UPDATEBIN_APOLLO3_BLUE) $(UPDATEBIN_APOLLO3_BLUE_ETHERMIND)
$(APPBIN_APOLLO3_BLUE_ETHERMIND)
# Apollo3 Cordio
or $(APPBIN_APOLLO3_BLUE) starter_binary_apollo3_blue.bin
```

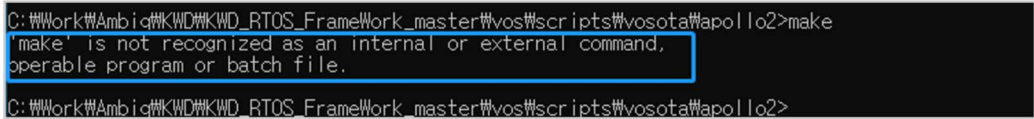
2. Run **Make** in command prompt, then the two binary files will be generated:
  - **starter\_binary\_apollo3\_blue.bin**: OTA bootloader + 1st firmware image
  - **update\_binary\_apollo3\_blue.bin**: 2nd firmware image that will be used for OTA reprogramming

## 13.2 Buildtools Installation

If the **make.exe** is not available to use **Makefile**, and the below error message (Figure 13-2) 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/96crrw2muzudqzae87bk1o0kjn4gu7v>
  - **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 13-2: Error Message Due to Non-existing make.exe



```
C:\Work\Ambiq\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\Ambiq\KWD\KWD_RTOS_Framework_master\vos\scripts\vosota\apollo2>
```

## 13.3 Python Installation

Python 3.6 or later needs to be installed. The **pycryptodome** package is required to build Apollo3 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

## 14

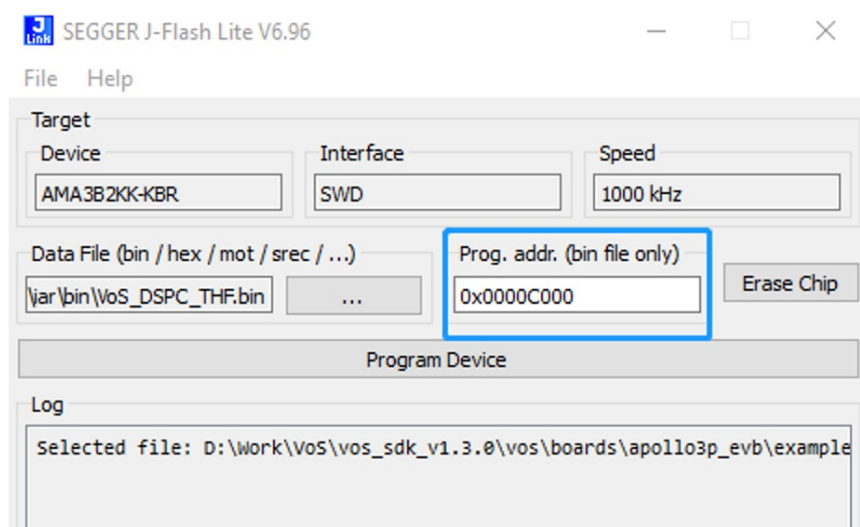
# OTA Image Download

## 14.1 Download Firmware to the EVB

- If the EVB has not been updated with the OTA supported image, the SoC 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\_apollo3\_blue.bin**.
- Start OTA binary can be programmed using the Segger J-Flash Lite programming utility. It needs to assign **Prog. addr.** to 0x0000C000.

**NOTE:** After installing the latest J-Link S/W (v6.47d or later), this address is updated automatically if Apollo3 Blue Plus (AMA3B2KK) is selected.

Figure 14-1: J-Flash Lite Address Setting for Apollo3 Blue Plus





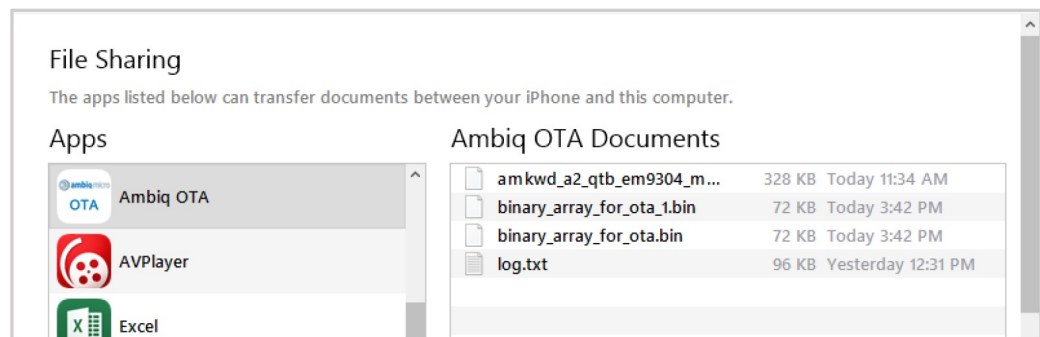
## 14.2 Update the EVB Firmware Via OTA

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:

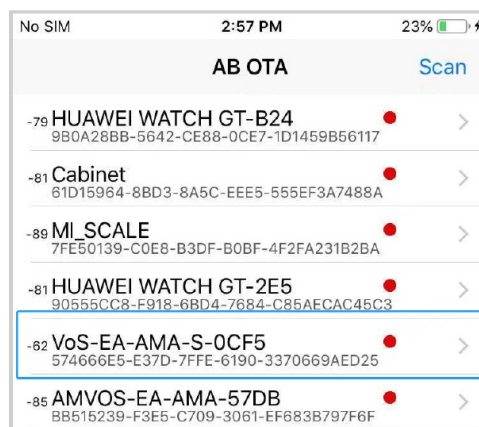
1. Install the Ambiq OTA app from the Apple store by searching "Ambiq":  
<https://apps.apple.com/kr/app/ambiq-ota-update/id1506027473>
2. Add KWD "update" image file (**update\_binary\_apollo3\_blue.bin**) to Ambiq OTA app's internal storage using iTunes (or iTools).
  - a. Connect the phone to the 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\_apollo3\_blue.bin** file to the **Documents** box.

Figure 14-2: 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 14-3: Ambiq OTA App Scan Result



5. Select **Load bin file**, and choose the update image file (e.g., **update\_binary\_apollo3\_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.

## 14.3 OTA App for Android

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

SECTION

15

# Audio Data Recording Using RTT or AMU2S

The VoS SDK supports RTT (J-Link) and AMU2S (SPI to USB) audio recording. AMU2S is supported on Apollo3 Blue Plus EVB only.

## 15.1 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 15-1: 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
```

- For AMU2S recording, set **configUSE\_AMU2S\_RECORDER** to 1.

Figure 15-2: AMU2S Recorder Definition

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

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

Figure 15-3: 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 15-4: Disable Codec and Bluetooth Low Energy When Using RTT Recorder

```
#define configUSE_BLE           0
#define configUSE_AUDIO_CODEC  0
#define configUSE_LEDS        1
```

4. Build and download image to board.

## 15.2 Install Software on PC

### 15.2.1 Install J-Link v6.47x or Later

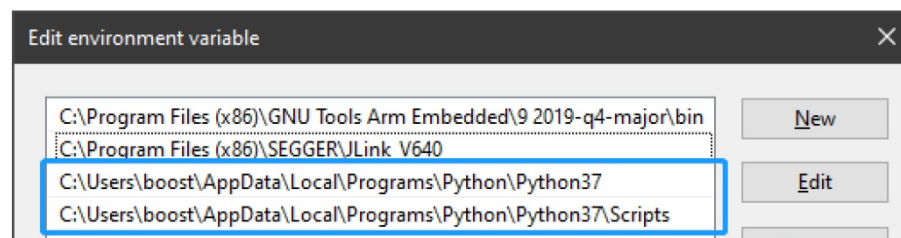
Use the following procedure to install J-Link v6.47x or later:

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.

### 15.2.2 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 15-5: System Path Environment Variable



### 15.2.3 RTT and AMU2S PCM Recorder Script File Package

- **vos\_audio\_recorder\_200714.7z**: [Box Cloud](#)
- **vos\_audio\_recorder\_200714.7z**: [Baidu Cloud \(for China\)](#) pw: 5brv

## 15.3 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 15-6: Run RTT datalogger batch file

```

C:\WINDOWS\system32\cmd.exe
LOCAL_APPDATA=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-SAMBU128 V3 compiled Jan 7 2019 14:06:26
S/N: 483071583

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
  
```

- AMU2S recording: run **vos\_amu2s\_recorder.bat**

Figure 15-7: Run AMU2S audio recorder

```

C:\Windows\system32\cmd.exe
C:\Work#\VoS#\Test\amu2s_tool_1122>python ftdi_bin_decoder.py -c logger
=====
AmU2S Receiver - Voice-On-SPOT, Ambiq Micro.
Version 0.6, 2019-11-22
www.ambiqmicro.com
=====

USB device opened successfully!
Received data will be saved to file: apollo_spi.bin
( Press anykey to stop recording )

start receiving spi data.....
Transfer Speed, 62.46 KBytes/S      File size, 0.15 MB

Processing data.....
already decoded 213.97 KB data. 100.00% completed...

All data decoded...
Get 0 bytes general data
Get 213440 bytes pcm data
Get 0 bytes spp/merged data
Get 0 bytes opus data

C:\Work#\VoS#\Test\amu2s_tool_1122>
C:\Work#\VoS#\Test\amu2s_tool_1122>
  
```

3. Press **BTNO** on Apollo3 Blue 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) or **apollo\_spi\_log\_pcm** (AMU2S).

## 15.4 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`
  - AMU2S: Command shell  
-> `python bin_to_wav.py pcm -i apollo_spi_log_pcm`

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

## SECTION

## 16

# Debug Message Output

## 16.1 UART

Use the following procedure for UART debugging:

1. Define **configUSE\_STDIO\_PRINTF** and **configUSE\_PRINTF\_UART0** to 1.

Figure 16-1: configUSE\_STDIO\_PRINTF definition in am\_vos\_sys\_config\_h

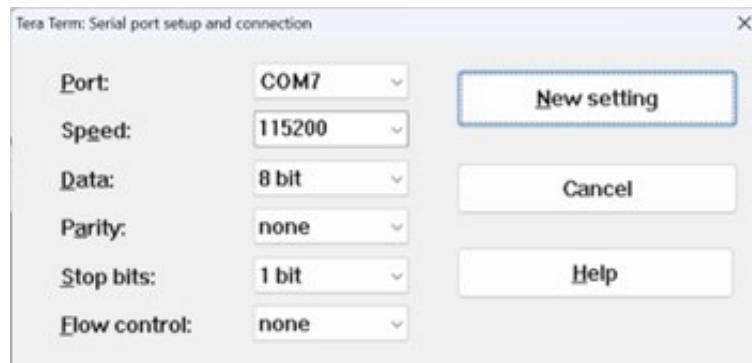
```
#define configUSE_SYSVIEWER           0
#define configUSE_SYS_LOG             0
#define configUSE_RTT_RECORDER       0
#define configUSE_STDIO_PRINTF       1
```

Figure 16-2: 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
```

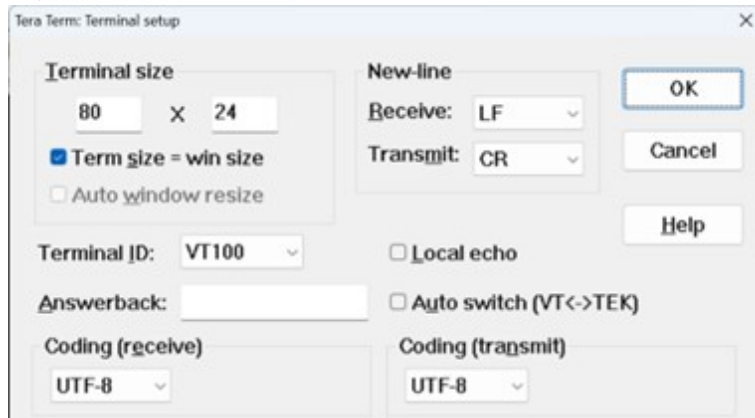
2. Complete the Serial port setup window as shown below:
  - a. Select **115200** in the **Speed** drop-down option.
  - b. Select **8 bit** in the **Data** drop-down option.
  - c. Select **none** in the **Parity** drop-down option.
  - d. Select **1 bit** in the **Stop bits** drop-down option.
  - e. Select **none** in the **Flow control** drop-down option.
  - f. Click **New setting**.

Figure 16-3: 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 16-4: Terminal Setup



UART debugging log is shown in Figure 16-5.

Figure 16-5: 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 rcv
[AMA Callback] VOS_AMA_EVT_GET_DEV_INFO

```



## 16.2 SWO Output

Use the following procedure for SWO debugging:

1. Define **configUSE\_STDIO\_PRINTF** and **configUSE\_PRINTF\_SWO** to 1.

Figure 16-6: configUSE\_STDIO\_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_STDIO_PRINTF  1

```

Figure 16-7: configUSE\_PRINTF\_SWO definition in am\_vos\_sys\_config.h

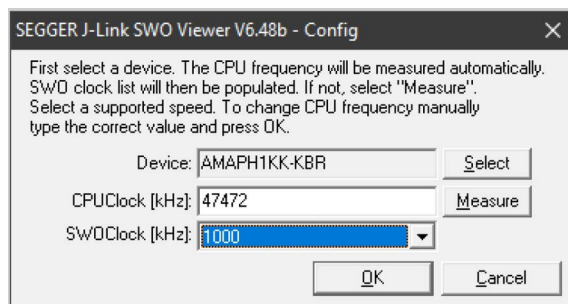
```

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

```

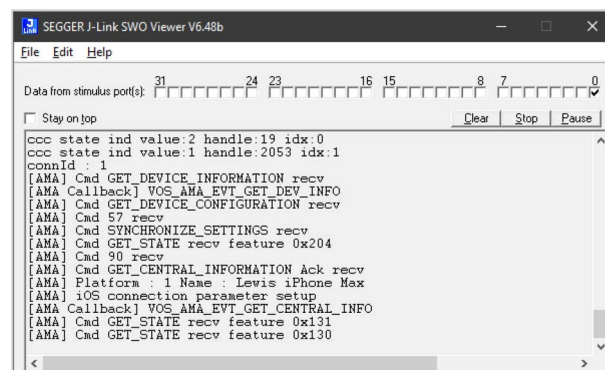
2. Run J-Link SWO Viewer and complete the following:
  - a. Select SoC type (e.g., Apollo3 Blue Plus: AMA3B2KK) in the **Device** box.
  - b. Click **Measure**.
  - c. Select **1000** in the **SWO Clock [kHz]** drop-down option.
  - d. Click **OK**.

Figure 16-8: SWO device and clock configuration



SWO debugging log message is shown in Figure 16-9.

Figure 16-9: SWO debugging message print out



## SECTION

## 17

# Power Consumption Profile

## 17.1 Hardware Setup

Use the following procedure to setup hardware:

- Measure current of **VDD\_MCU** using P19 pin on Apollo3 Plus EVB.

## 17.2 Software Setup

To check power consumption of MCU, SWO logging should be turned off (example: **configUSE\_STDIO\_PRINTF** to 0).

Table 17-1: Power/CPU Profiling Cases Define Settings

Measurement Cases	USE_DMIC_MB0	configUSE_AMBIQ_VAD	configUSE_Sensory_THF
1. Base	0	0	0
2. Base + PDM	1	0	0
3. Base + PDM + VAD	1	1	0
4. Base + PDM + VAD + THF	1	1	1

## 17.3 Power Measurement

- For getting a PDM power number, get the case #1 (Base) and #2 (Base + PDM) result. Then, calculate the delta of two numbers.
- $\text{PDM power (uW)} = \text{Case \#2 result} - \text{case \#1 result (uW)}$

SECTION

18

## CPU Usage Profile

### 18.1 CPU Usage

CPU usage can calculate the delta of CPU utilization at each case. CPU utilization (percent) is printed out by SWO log.

### 18.2 Software Setup

- To check overall CPU usage of each case, **configUSE\_STDIO\_PRINTF** and **VOS\_MEASURE\_SYS\_MIPS** should be set to **1** in **am\_vos\_sys\_config.h** file.

Figure 18-1: SWO print enable in am\_vos\_sys\_config.h file

```
54 #define configUSE_SYSVIEWER 0
55 #define configUSE_RTT_RECORDER 0
56 #define configUSE_STDIO_PRINTF 1
```

Figure 18-2: System CPU usage measurement enable in am\_vos\_sys\_config.h

```
133 #define VOS_MEASURE_AMSPP_MIPS 0
134 #define VOS_MEASURE_SPP_MIPS 0
135 #define VOS_MEASURE_WWE_MIPS 0
136 #define VOS_MEASURE_CODEC_MIPS 0
137 #define VOS_MEASURE_SYS_MIPS 1
138 #define VOS_MEASURE_VAD_WAKETIME 0
```

- Monitoring period can be configured by **AM\_VOS\_BENCHMARK\_TIME\_SEC** define in **am\_vos\_board\_setup.h** file (default is 10 secs)

Figure 18-3: System CPU usage measurement enable in am\_vos\_sys\_config.h

```

154 // #define AM_VOS_BENCHMARK_TIMER_CLK      AM_VOS_BENCH_TIMER_6MHZ
155 #define AM_VOS_BENCHMARK_TIMER_CLK      AM_VOS_BENCH_TIMER_32KHZ
156 // #define AM_VOS_BENCHMARK_TIMER_CLK      AM_VOS_BENCH_TIMER_DWT
157
158 #define AM_VOS_BENCHMARK_TIMER_FREQ      AM_VOS_BENCHMARK_TIMER_CLK
159
160 #define AM_VOS_BENCHMARK_TIME_SEC      10

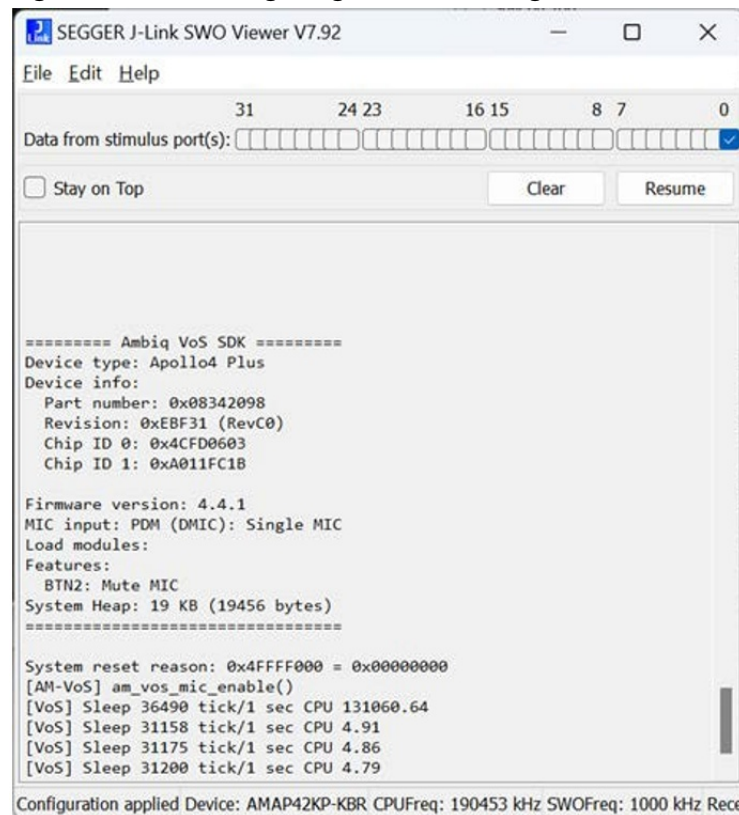
```

## 18.3 Printing Result of CPU Utilization

Use the following procedure to print result of CPU utilization:

1. Run SWO logger, setup the device as **AMA3B2KK-KBR** and speed as 1 MHz.

Figure 18-4: CPU Usage Log Via SWO Debug Print



2. Measure the timer counter and accumulate system sleep time for a specific period (e.g., 10 secs). From the above result, the 312K timer counted for a wake-up time with a 95.25% sleep time result. The timer is using 32768 Hz clock source.
3. To getting PDM CPU usage, get the case #1 (Base) and #2 (Base + PDM) result.
4. Calculate the delta of two results.  

$$\text{PDM CPU usage (\%)} = \text{Case \#2 CPU usage} - \text{Case \#1 CPU usage (\%)}$$



© 2024 Ambiq Micro, Inc. All rights reserved.

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

[www.ambiq.com](http://www.ambiq.com)

[sales@ambiq.com](mailto:sales@ambiq.com)

+1 (512) 879-2850

A-SOCA3P-UGNA01EN v1.4

July 2024