MT793X IoT SDK for ADSP VA Framework

User Guide

Version:	1.1
Release date:	2021-07-23

Use of this document and any information contained therein is subject to the terms and conditions set forth in Exhibit 1. This document is subject to change without notice.

Version History

Version	Date	Description
1.0	2021-03-16	Official release
1.1	2021-07-23	Update content

Table of Contents

List of Figures

1 Overview

This document introduces the VA (Voice Activity) Framework provided by MediaTek, as indicated by the the left region in the following figure.

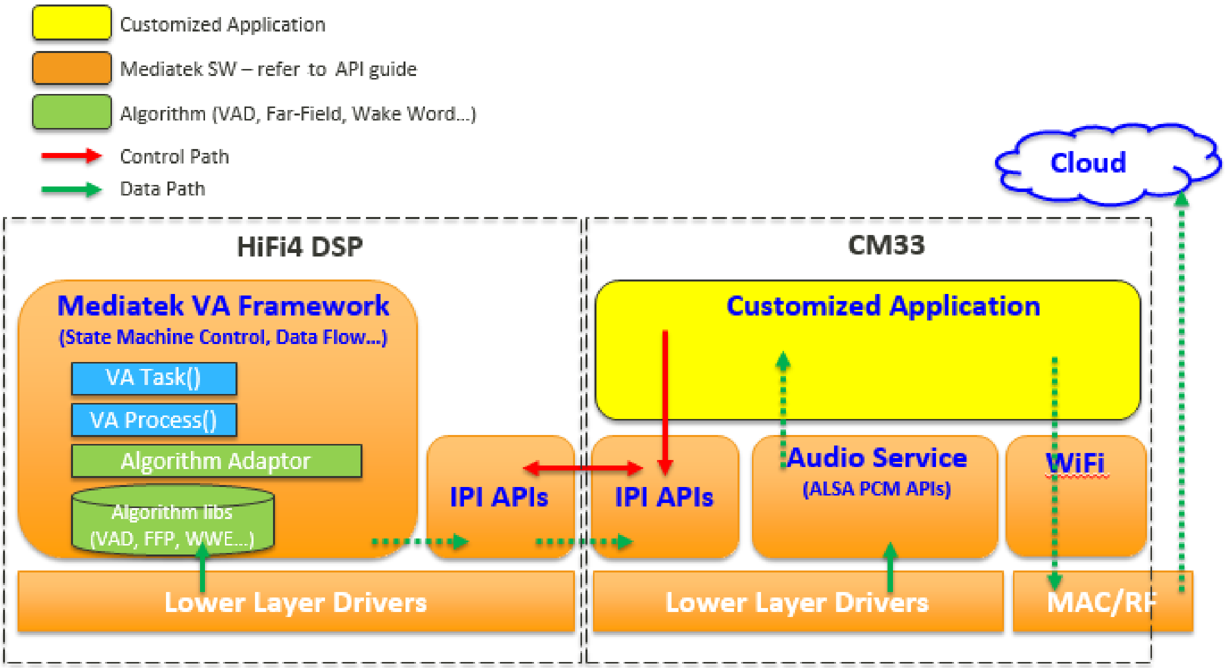


Figure 1 System Overview

2 How It Works

2.1 How VA Detection Works

The VA state machine has three stages for different system power requirements.

Phase1: VAD (Voice Activity Detection) works in low power settings, in which DSP runs at 26MHz XTAL CLK with 256KB SRAM in the active mode. The Cortex-M33 processor (CM33) and PSRAM are in the sleep mode.

Phase2: FFP and WWE operate in middle power settings, in which DSP runs at 600MHz PLL CLK with 256KB SRAM and 8MB PSRAM in the active mode. CM33 is in the sleep mode.

Phase3: There are two types of applications that can be customized. The 1st scenario is that CM33 handles the pre-defined local command. Once the local command is detected, DSP wakes up CM33 to process the corresponding actions, for example, flashing LEDs or playing local audio data. The 2nd scenario is that cloud voice services are integrated to handle voice commands online, for example, AIA (AVS Integration for AWS IoT).

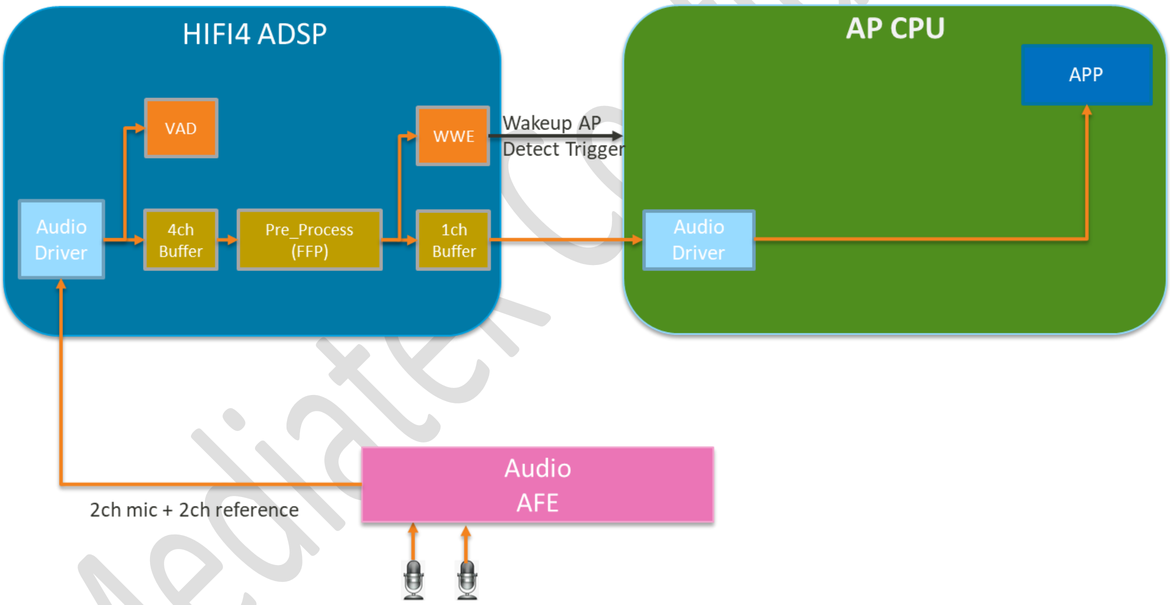


Figure 2 How VA Detection Works

2.2 Data Processing Path

This section introduces the data path of the VA Framework. The sound is recorded by microphone and stored in AFE DMA buffer. The period data is processed by the VAD algorithm, which is used to detect nearby sound. This data stream is stream 1. You can choose to replace the VAD algorithm according to your needs.

Once the detection succeeds, the period data continues to be processed by the Pre-process algorithm such as Far-field. The output of Pre-process is sent to Wake Word and local command process. This data steam is stream 2.

Once Wake Word detection is successful, data is uploaded from the DSP side to the CM33 side. This stream is stream 3. If Local Command detection is successful, AP receives a notification.

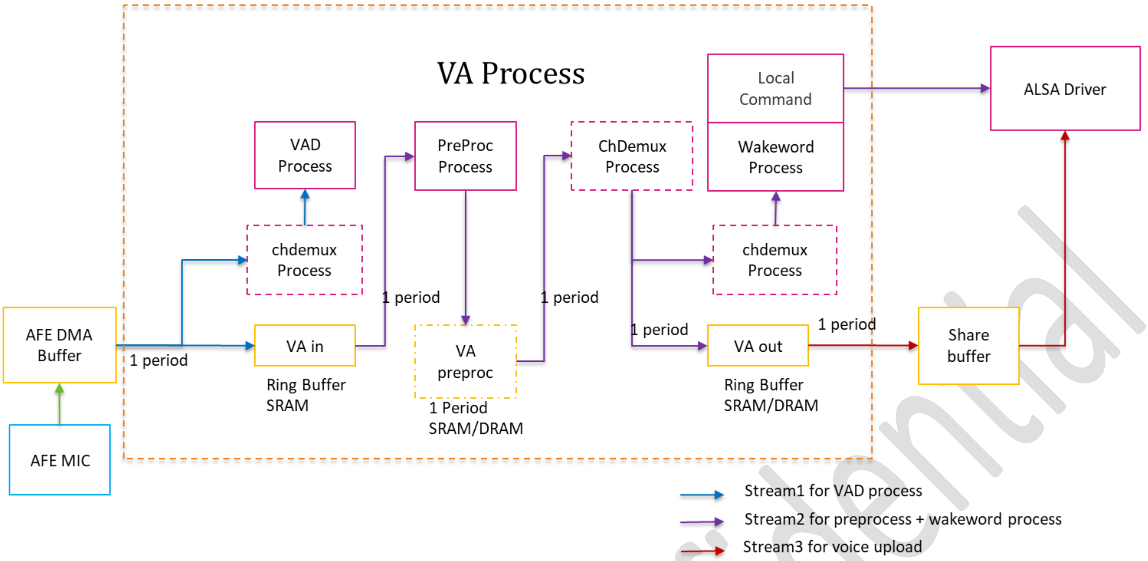


Figure 3 VA Process

3 VA User Guide

3.1 VA Detection Procedure

3.1.1 Steps to Start Detection

Step1. Set ADSP as enabled

Step2. Set audio route path for VA mic input

Step3. Start VA detection

Step4. Listen for the detection notification. If the audio is detected, voice recording starts. After the process is done, voice recording stops.

3.1.2 Steps to Stop Detection

Step1. Make sure voice recording is disabled

Step2. Stop VA detection

Step3. Set ADSP as disabled

3.2 Paths of VA Audio Routes

3.2.1 DAI

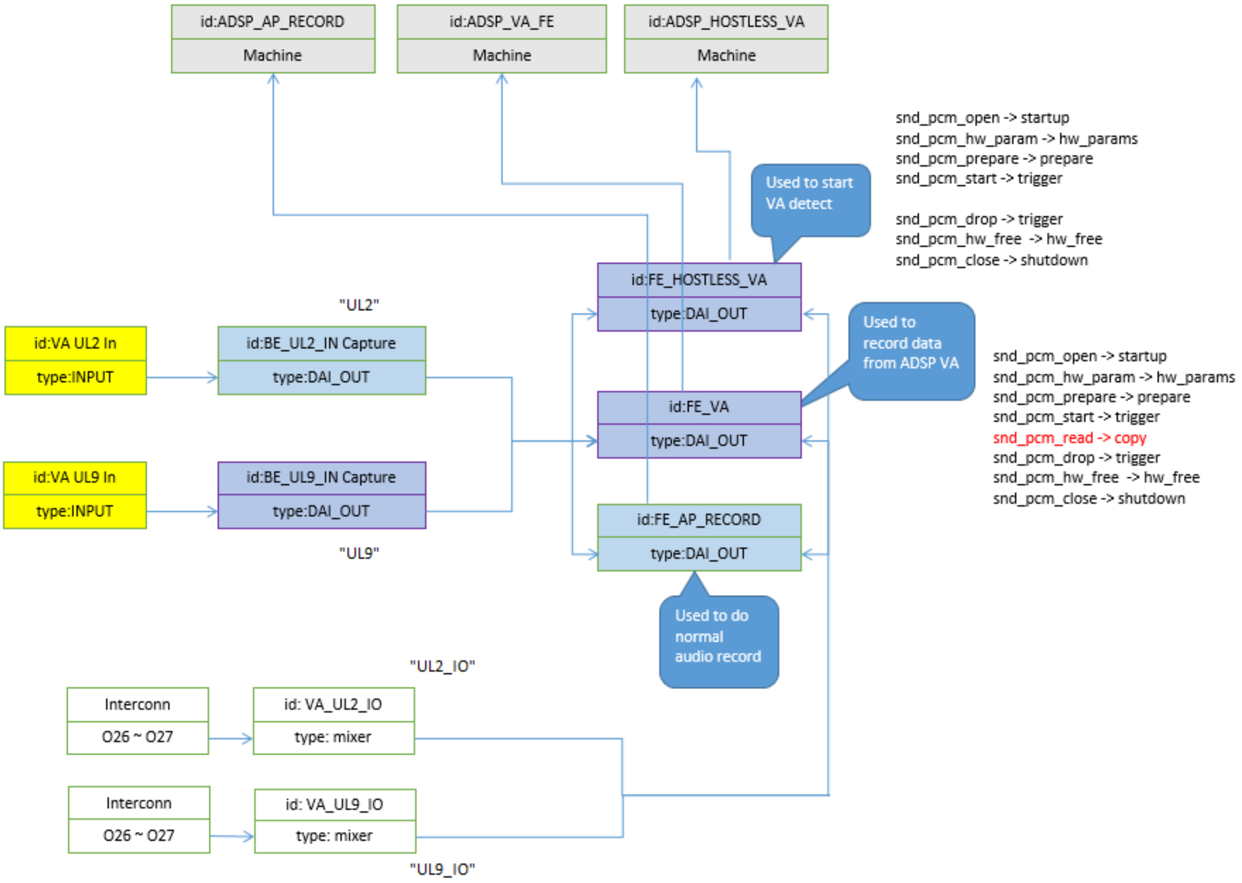
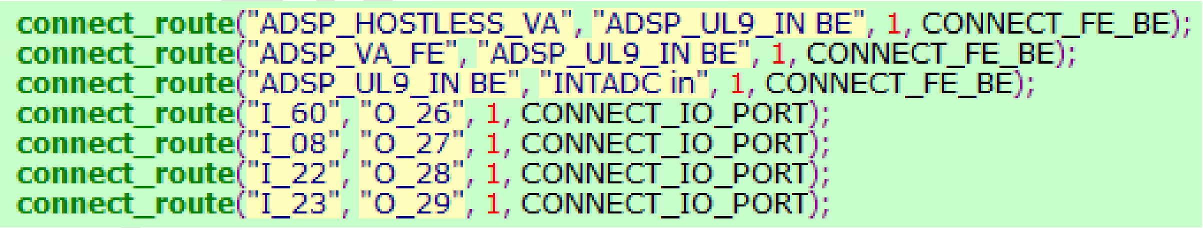


Figure 4 Audio Route

Path: ADC => UL9 => ADSP



```
connect_route("ADSP_HOSTLESS_VA", "ADSP_UL9_IN BE", 1, CONNECT_FE_BE);
connect_route("ADSP_VA_FE", "ADSP_UL9_IN BE", 1, CONNECT_FE_BE);
connect_route("ADSP_UL9_IN BE", "INTADC in", 1, CONNECT_FE_BE);
connect_route("I_60", "O_26", 1, CONNECT_IO_PORT);
connect_route("I_08", "O_27", 1, CONNECT_IO_PORT);
connect_route("I_22", "O_28", 1, CONNECT_IO_PORT);
connect_route("I_23", "O_29", 1, CONNECT_IO_PORT);
```

Figure 5 Paths of Audio Routes

Note: Use the API “connect_route” to connect each BE/FE and IO port.

3.2.2 Enable ADSP

3.2.2.1 Enable or Disable ADSP

In the SDK, you can use the test CLI to enable or disable ADSP via the following commands.

Turn on and enable ADSP by using “aud_dbg cset ADSP_Enable 1 1”, and
Turn off and disable ADSP by using “aud_dbg cset ADSP_Enable 1 0”.
Note: 1. You can configure DSP_BOOT_RUN to set ADSP as automatically enabled when the system boots up.
2. After turning on ADSP, you should check if ADSP status is ON by using “aud_dbg cget ADSP_Enable 1”.

3.2.2.2 Start or Stop VA Detection

You can check the sample code for starting VA detection in the SDK.

```
driver/chip/mt7933/src/sound/test/va.c
```

Note: Use the Audio PCM interface to start or stop VA detection by using start pcm and stop pcm. Do not execute pcm read.

3.2.2.3 Notify Listener and Record Voice

To receive task notification from the audio driver, create the VA task and wait for task notification.
To record voice, use the VA record Audio PCM interface to control voice recording

```
pcm open + pcm hw param + pcm prepare + pcm read => start voice recording
pcm drop + pcm hw free + pcm close               => stop voice recording
```

Note: After voice recording stops, the ADSP system enters VA detection mode again.

4 VA Configuration

4.1 Configuration Position

The following factors can affect VA detection.

VA Hostless Setting Parameters

Sample rate, bit depths, channel number, period size, and period number

ADSP Project Configuration

```
+--- HIFI4
|
|  +--- build
|  +--- drivers
|  +--- project
```

project/mt7933/config/$project_name/HIFI4_A/AudioConfig.mk

project/mt7933/config/$project_name/HIFI4_A/ProjectConfig.mk

AudioConfig.mk is used for setting audio (VA) features.

ProjectConfig.mk is used for setting system-based features.

4.2 Hostless Setting

Use the following parameters for VA Hostless setting.

Sample rate, bit depths, channel number, period size, and period number

This setting will be applied to AFE DMA in the path “AFE DMA => ADSP VA Process”

And the period size is also triggered as the period size value in all VA process stages.

You can complete the above settings by calling the tinypcm API.

For example:

```
struct msd_hw_params *params = pvPortMalloc(sizeof(strust msd_hw_params));
params->format = MSD_PCM_FMT_S16_LE; /* bit depths */
params->channels = 4; /* channel number */
params->period_count = 12; /* period number */
params->rate = 16000; /* sample rate */
params->period_size = params->rate/100; /* period size */
```

4.3 Options for Audio Features

ProjectConfig.mk

CFG_AUDIO_SUPPORT = yes

ADSP Audio Framework Enable (AudioConfig.mk)

CFG_MTK_AUDIO_FRAMEWORK_SUPPORT = yes

Note: To disable all audio associated code, set the above 2 values to no

4.4 Options for Algorithm Features

AudioConfig.mk

CFG_AUDIO_VA_VAD_DUMMY
CFG_AUDIO_VA_AEC_DUMMY
CFG_AUDIO_VA_PREPROC_MTKFFP
CFG_AUDIO_VA_WW_DUMMY

For example,

CFG_AUDIO_VA_VAD_DUMMY = no
CFG_AUDIO_VA_AEC_DUMMY = no
CFG_AUDIO_VA_PREPROC_MTKFFP = no
CFG_AUDIO_VA_WW_DUMMY = no

Note: 1. Set all default features to no or remove unused ones.
2. Add the algorithm feature supported by your project.
3. The setting of these features determines whether your algorithm is built and used.

4.5 Process Enable Configuration

AudioConfig.mk

CFG_VA_VAD_DEFAULT_ON
CFG_VA_WW_DEFAULT_ON
CFG_VA_PREPROC_DEFAULT_ON
CFG_VA_VAD_WW_IN_ONE

For example,

CFG_VA_VAD_DEFAULT_ON = no
CFG_VA_WW_DEFAULT_ON = no
CFG_VA_PREPROC_DEFAULT_ON = no
CFG_VA_VAD_WW_IN_ONE = 0

Note: The setting of these features determines the VA process flow

4.5.1 Example 1

AudioConfig.mk

CFG_VA_VAD_DEFAULT_ON = yes
CFG_VA_WW_DEFAULT_ON = yes
CFG_VA_PREPROC_DEFAULT_ON = yes
CFG_VA_VAD_WW_IN_ONE = 0

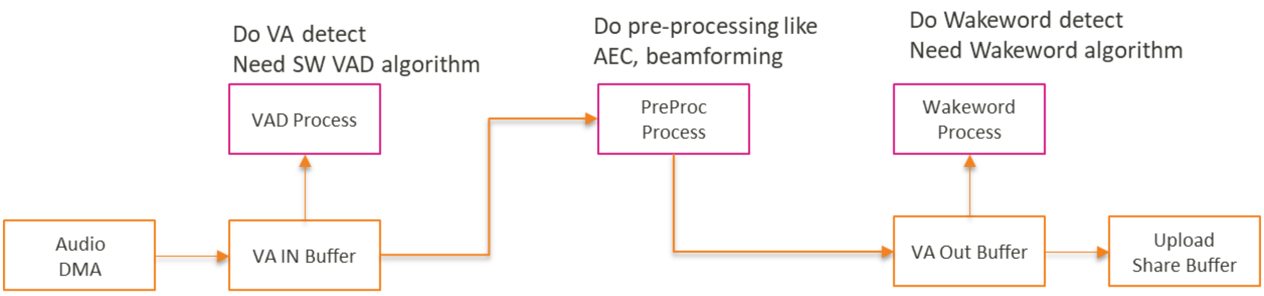


Figure 6 Example 1

4.5.2 Example 2

AudioConfig.mk

CFG_VA_VAD_DEFAULT_ON = yes

CFG_VA_WW_DEFAULT_ON = no

CFG_VA_PREPROC_DEFAULT_ON = no

CFG_VA_VAD_WW_IN_ONE = 1

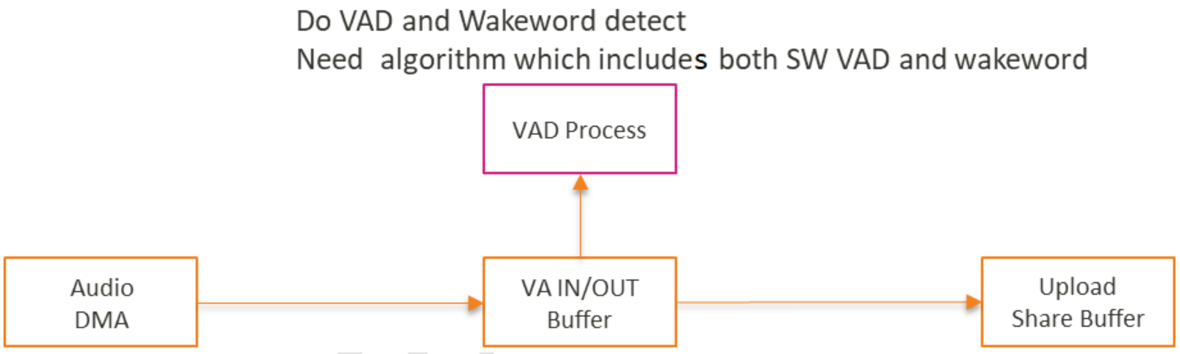

Do VAD and Wakeword detect
Need algorithm which includes both SW VAD and wakeword
VAD Process
Audio
DMA
VA IN/OUT
Buffer
Upload
Share Buffer

Figure 7 Example 2

4.5.3 Example 3

AudioConfig.mk

CFG_VA_VAD_DEFAULT_ON = no

CFG_VA_WW_DEFAULT_ON = yes

CFG_VA_PREPROC_DEFAULT_ON = no

CFG_VA_VAD_WW_IN_ONE = 0

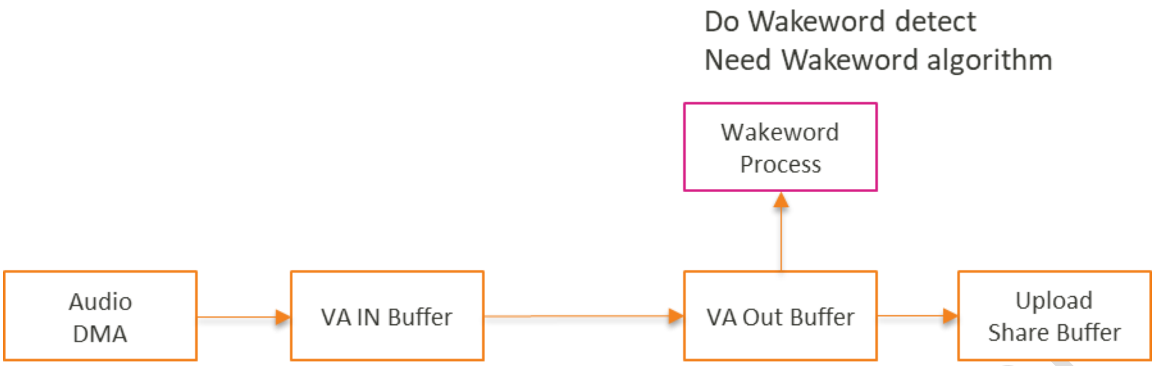


Figure 8 Example 3

4.5.4 Example 4

AudioConfig.mk

CFG_VA_VAD_DEFAULT_ON = no

CFG_VA_WW_DEFAULT_ON = no

CFG_VA_PREPROC_DEFAULT_ON = yes

CFG_VA_VAD_WW_IN_ONE = 0

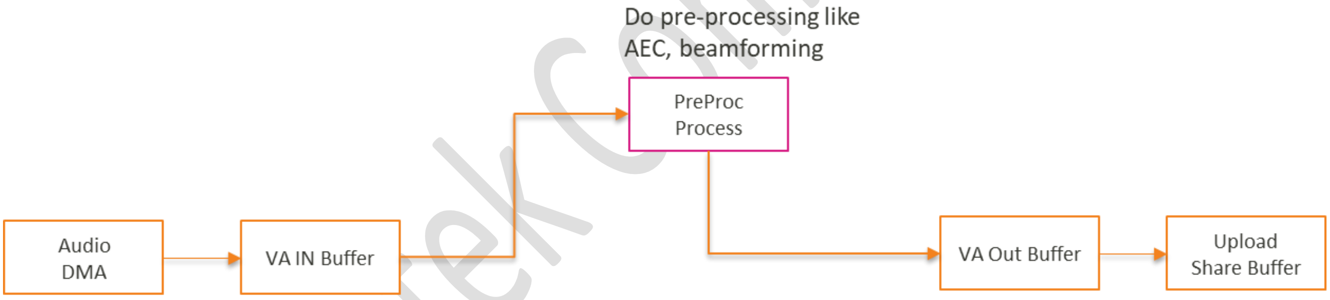


Figure 9 Example 4

4.6 VAD Configuration

AudioConfig.mk

CFG_VA_VAD_ALWAYS_ON

CFG_VA_VAD_BITWIDTH

CFG_VA_VAD_CHNUM

For example,

CFG_VA_VAD_ALWAYS_ON = 1

CFG_VA_VAD_BITWIDTH = 16

CFG_VA_VAD_CHNUM = 1

Note: 1. If you set CFG_VA_VAD_ALWAYS_ON to 1, VA detection works in the VAD, Wake Word and Voice upload stages. If you set CFG_VA_VAD_ALWAYS_ON to 0, VA detection works only in the VAD stage. The recommended value is 1

2. CFG_VA_VAD_BITWIDTH and CFG_VA_VAD_CHNUM are the algorithm input data's requests. If VA IN buffer format is different from algorithm's request, the framework converts the data format by default.

4.7 Wake Word Detection Configuration

4.7.1 Wake Word Data Format

AudioConfig.mk
CFG_VA_WW_ALWAYS_ON
CFG_VA_WW_BITWIDTH
CFG_VA_WW_CHNUM
CFG_VA_WW_PRE_ROLL_LEN

For example,

```
CFG_VA_WW_ALWAYS_ON = 1
CFG_VA_WW_BITWIDTH = 16
CFG_VA_WW_CHNUM = 1
CFG_VA_WW_PRE_ROLL_LEN = 500    #millisecond
```

Note: 1. If you set CFG_VA_WW_ALWAYS_ON to 1, Wake Word detection works in the Wake Word and Voice upload stages. If you set CFG_VA_WW_ALWAYS_ON to 0, Wake Word detection only works in the Wake Word stage.

2. CFG_VA_WW_BITWIDTH and CFG_VA_WW_CHNUM are the algorithm input data's requests. If VA OUT buffer format is different from algorithm's request, the framework converts the data format by default.

3. CFG_VA_WW_PRE_ROLL_LEN is Wake Word pre-roll length. It is requested by the voice process in the CM33 side. When ADSP's Wake Word detection is successful, the framework uploads the voice data from the position as illustrated in the following graphic.

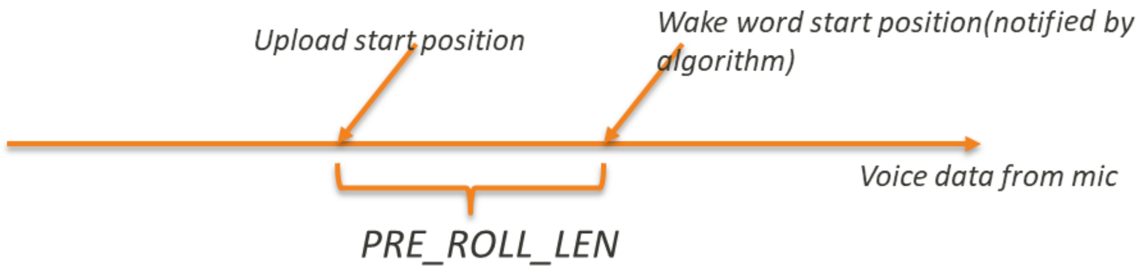


Figure 10 Upload Data

4.7.2 Wake Word Detection Configuration

AudioConfig.mk
CFG_VA_PROCESS_SPEED_UP_EN
CFG_VA_WW_TIMEOUT_EN
CFG_VA_WW_TIMEOUT_LEN

For example,
CFG_VA_PROCESS_SPEED_UP_EN = 1
CFG_VA_WW_TIMEOUT_EN = 1
CFG_VA_WW_TIMEOUT_LEN = 2000
Note: If the VAD and Wake Word processes are supported

1. Set CFG_VA_PROCESS_SPEED_UP_EN as enabled, and the input voice data is processed as soon as possible to reduce latency in VAD buffer.
2. Set CFG_VA_WW_TIMEOUT_EN as enabled, and a timer starts counting during the Wake Word state. If no Wake Word detection event happens during TIMEOUT_LEN, a timeout event occurs and the state changes to VAD state from Wake Word state.

4.8 In and Out Buffer Configuration

AudioConfig.mk
CFG_VA_VAD_BUF_LEN
CFG_VA_VAD_BUF_TYPE
CFG_VA_WW_BUF_LEN
CFG_VA_WW_BUF_TYPE

For example,
CFG_VA_VAD_BUF_LEN = 500
CFG_VA_VAD_BUF_TYPE = ADSP_MEM_TASK_RESERVE
CFG_VA_WW_BUF_LEN = 2500
CFG_VA_WW_BUF_TYPE = ADSP_MEM_TASK_RESERVE

Note: 1. The unit of BUF_LEN is one millisecond.
VAD IN buffer length is defined by CFG_VA_VAD_BUF_LEN.
If VAD process is not supported, the CFG_VA_VAD_BUF_LEN is not used.
VAD OUT buffer length is defined by CFG_VA_WW_BUF_LEN.
If Wake Word process is not supported, the CFG_VA_WW_BUF_LEN is not used.
2. BUF_TYPE specifies the following buffer memory types.
ADSP_MEM_TASK_RESERVE
ADSP_MEM_LP_CACHE
ADSP_MEM_NORMAL_CACHE

VA Task reserves 1 big memory pool from DTCM SRAM heap. TASK_RESERVE type allocates memory from this reserved memory pool.

ADSP_MEM_LP_CACHE is from DTCM heap.

ADSP_MEM_NORMAL_CACHE is from DRAM heap.

4.9 Upload Buffer Configuration

AudioConfig.mk
CFG_VA_VOICE_UPLOAD_CH_NUM
CFG_VA_VOICE_UPLOAD_BITWIDTH
CFG_VA_UPLOAD_BUF_LEN
CFG_VA_UPLOAD_SPEED_UP_EN

For example,
CFG_VA_VOICE_UPLOAD_CH_NUM = 2
CFG_VA_VOICE_UPLOAD_BITWIDTH = 16
CFG_VA_UPLOAD_BUF_LEN = 2000
CFG_VA_UPLOAD_SPEED_UP_EN = 1

Note: 1. CFG_VA_VOICE_UPLOAD_CH_NUM and CFG_VA_VOICE_UPLOAD_BITWIDTH specify the upload voice data format

2. CFG_VA_UPLOAD_BUF_LEN specifies ADSP to AP shared upload buffer size (in millisecond). This buffer is fixed to DRAM

3. Set 'CFG_VA_UPLOAD_SPEED_UP_EN = 1', and the ADSP uploads data to AP as soon as possible to reduce latency in Wake Word buffer.

4.10 Low Power and SRAM Heap Configuration

platform.mk
configTOTAL_HEAP_SIZE = ((size_t) (123* 1024))
AudioConfig.mk
CFG_VA_TASK_RESERVE_SRAM = 0x10000

Note: 1. configTOTAL_HEAP_SIZE defines the size of the Low Power heap and SRAM heap.

2. TASK RESERVE SRAM is the size of SRAM reserved by VA TASK from SRAM heap.

3. TASK RESERVE SRAM should be bigger than the size of MIC IN AFE DMA buffer + VA IN/OUT (if the type is reserved type).

4. The total size of SRAM is 256KB.

4.11 Power Control Configuration

AudioConfig.mk
CFG_VA_IDLE_DSP_CLK = DSP_CLK_13M
CFG_VA_VAD_DSP_CLK = DSP_CLK_26M

```
CFG_VA_WW_DSP_CLK   = DSP_CLK_26M
CFG_VA_VOICE_UPLOAD_DSP_CLK = DSP_CLK_PLL

CFG_VA_IDLE_SYS_HW = DSP_PSRAM_NONEED
CFG_VA_VAD_SYS_HW  = DSP_PSRAM_NONEED
CFG_VA_WW_SYS_HW   = DSP_PSRAM_NEED
CFG_VA_VOICE_UPLOAD_SYS_HW = DSP_PSRAM_NEED
```

Note: 1. DSP_CLK supports the setting of DSP_CLK_13M, DSP_CLK_26M, DSP_CLK_PLL, DSP_CLK_PLL_D_8, DSP_CLK_PLL_D_4, and DSP_CLK_PLL_D_2 to select the DSP clock frequency you need.

2. SYS_HW supports the setting of DSP_PSRAM_NONEED, and DSP_PSRAM_NEED to select the PSRAM state you need.

3. For each VA state, the setting of DSP_CLK and SYS_HW is required. They must be set correctly according to the needs of the VA state.

5 Integrating Third Party Algorithm

5.1 Process Block Diagram

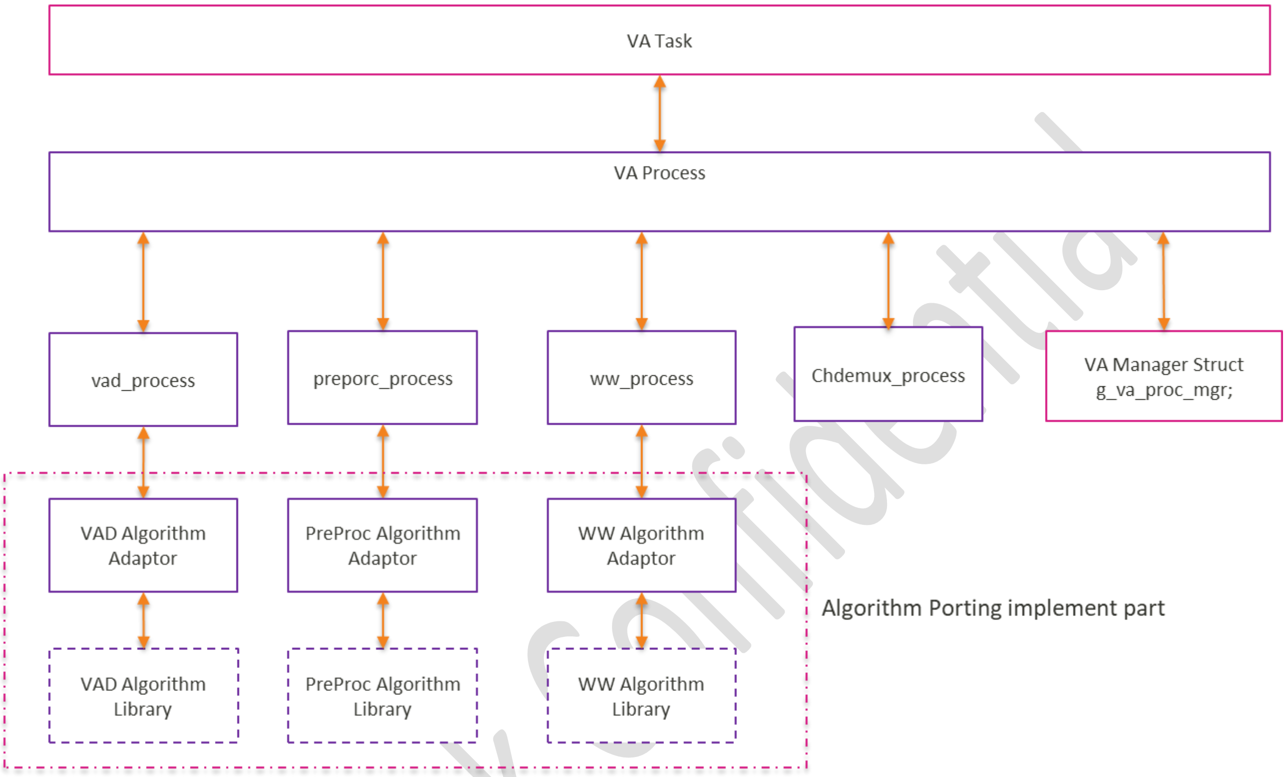


Figure 11 VA Process Block Diagram

5.2 Adaptor Implement

5.2.1 Implement Position

```
+--- HIFI4
   |
   |   +--- build
   |   +--- drivers
   |   |  +--- audio
   drivers/audio/tasks/voiceassistant/va_thirdparty/xxx/
    Note: xxx is the third party algorithm’s name or mark string.
```

5.2.2 Create Adaptor Folder

Create the algorithm adaptor folder in the above position, for example “vad” “wwe” “wwe_lite” “ffp” or other specific names.

5.2.3 Create Adaptor File

xxx_yyy_adaptor.c   xxx_yyy_adaptor.h

Note: xxx is the algorithm vendor's name or mark

yyy is the algorithm's name

For example, mtk_vad_adaptor   mtk_effp_adaptor

5.2.4 Implement xxx_yyy_adaptor.h Header File

void xxx_yyy_adaptor_register(struct algo_ops *ops);

Declare the algorithm register function, which is used in the vad_process.c, ww_processs.c or preproc_process.c

5.2.5 Implement xxx_yyy_adaptor_register() in xxx_yyy_adaptor.c

```
void xxx_yyy_adaptor_register(struct algo_ops *ops)
    {
        ops->init =  yyy_adaptor_init;
        ops->uninit = yyy_adaptor_init;
        ops->set_params = NULL;
        ops->get_params = NULL;
        ops->reset = NULL;
        ops->process = yyy_adaptor_process;
    }
```

Note: 1. There are 6 callbacks, init, uninit, set_params, get_params, reset, process in algo_ops. These callbacks are called from the init, uninit, set_params, get_params, reset, process functions from the vad_process   ww_process or preproc_process files.

2. If the callback is not implemented in the adaptor, set the callback to NULL in the register function.

5.2.6 Implement yyy_adaptor_init()

static int yyy_adaptor_init(struct va_proc_obj *obj, struct va_pcm_format in, struct va_pcm_format * out, int frames)

Note: 1. This function is used to perform algorithm initialization and setup. After the implementation is complete, you can use the algorithm to process audio data.

2. obj: can be converted to target process private manager structure.

For VAD process   : struct vad_private   *priv = (struct vad_private *)(obj->priv)

For Wakeword process : struct ww_private   *priv = (struct ww_private *)(obj->priv)

For PreProc process   : struct preproc_private * priv = (struct preproc_private)(obj->priv)

and the priv->priv can be used to store the algorithm handler.

in   : input audio data format

out: output audio data format; if there is no output, just ignore this parameter, for example, for VAD or Wake Word type of algorithm

frames: the audio sample number for every process unit.

5.2.7 Implement yyy_adaptor_uninit()

static int yyy_adaptor_uninit(struct va_proc_obj *obj)

Note: This function is used to perform algorithm uninitialization and release all the resources allocated in the init callback.

5.2.8 Implement yyy_adaptor_process()

static int yyy_adaptor_process(struct va_proc_obj *obj, char *inbuf, char *outbuf, int frames)

Note: 1. This function is used to send data to algorithm and return the results to upper layer.

2. obj: can be converted to target process private manager structure.

For VAD process : struct vad_private *priv = (struct vad_private *)(obj->priv)

For Wakeword process : struct ww_private *priv = (struct ww_private *)(obj->priv)

For PreProc process : struct preproc_private * priv = (struct preproc_private)(obj->priv)

inbuf : input audio data buffer

outbuf: output audio data buffer; if there is no output, just ignore this parameter, for example, for VAD or Wake Word type of algorithm

frames: the audio sample number for every process unit. It should be same as the frames parameter in the init callback.

3. How to return result for VAD and Wake Word.

[For VAD process algorithm]

yyy_adaptor_process() return VA_VAD_SUCESS means VA detection succeeds

yyy_adaptor_process() return VA_VAD_FAIL means VA detection fails

[For VAD and Wake Word in one algorithm]

yyy_adaptor_process() return VA_VAD_SUCESS means VA detection succeeds

yyy_adaptor_process() return VA_VAD_FAIL means VA detection fails in VAD stage

means Wake Word detection fails in Wake Word stage

yyy_adaptor_process() return VA_WW_SUCESS means Wake Word detection succeeds

yyy_adaptor_process() return VA_WW_CONTINUE means Wake Word detection continues

struct vad_private *pirv = (struct ww_private *) (obj->priv);

priv->state can be used to manage the VAD and Wake Word stage internal. 0 means VAD; 1 means Wake Word.

[Wake Word process algorithm]

struct ww_private *pirv = (struct ww_private *) (obj->priv);

priv->detect_result: 1 means detection succeeds; 0 means detection continues; 2 means detection fails

priv->wakeword: Wake Word string

priv->begin_idx : Wake Word start position if detection succeeds

priv->end_idx : Wake Word stop position if detection succeeds

priv->confidence: algorithm detection confidence

5.2.9 Implement reset/get_params/set_params

static int yyy_adaptor_reset(struct va_proc_obj *obj)

static int yyy_adaptor_set_params(struct va_proc_obj *obj, int cmd, void *data) static int yyy_adaptor_get_params(struct va_proc_obj *obj, int cmd, void *data)

Note: 1. If the algorithm implements the special operation, you can implement the above callbacks, and connect the process layer and algorithm layer.

2. For example, for FFP algorithm process, you can implement the CMD_GET_BEAMFORMING_RESULT in the get_params callback and supply beamforming result.

5.3 Editing the Build System

5.3.1 Makefile Modification

```
+--- HIFI4
|  +--- project
```

position: project/mtxxxx/platform/build/platform.mk

If code is built into SRAM space

```
Ifeq ($(CFG_XXX_YYY_SUPPORT),yes)
   INCLUDEDS +=\
        $(DRIVERS_COMMON_DIR)/audio/tasks/voiceassistant/va_thirdparty/xxx/yyy
   C_FILES += \
         $(DRIVERS_COMMON_DIR)/audio/tasks/voiceassistant/va_thirdparty/xxx/yyy/xxx_yyy_adaptor.c
   LISBFLAGS +=  algorithm library path
endif  # CFG_XXX_YYY_SUPPORT
```

If code is built into DRAM space

```
Ifeq ($(CFG_XXX_YYY_SUPPORT),yes)
   INCLUDEDS +=\
        $(DRIVERS_COMMON_DIR)/audio/tasks/voiceassistant/va_thirdparty/xxx/yyy
   NORMAL_C_FILES += \
         $(DRIVERS_COMMON_DIR)/audio/tasks/voiceassistant/va_thirdparty/xxx/yyy/xxx_yyy_adaptor.c
   NORMAL_SECTION_LISBS +=  algorithm library path
endif  # CFG_XXX_YYY_SUPPORT
```

Note: 1. XXX means vendor name or mark; YYY means the algorithm type.

2. You can add other associate header file, c file, library to the above position.

3. For all code built to SRAM space, you can use NORMAL_SECTION_FUNC to make init/uninit callback to DRAM space to reduce SRAM code cost.

5.3.2 Configuration Modification

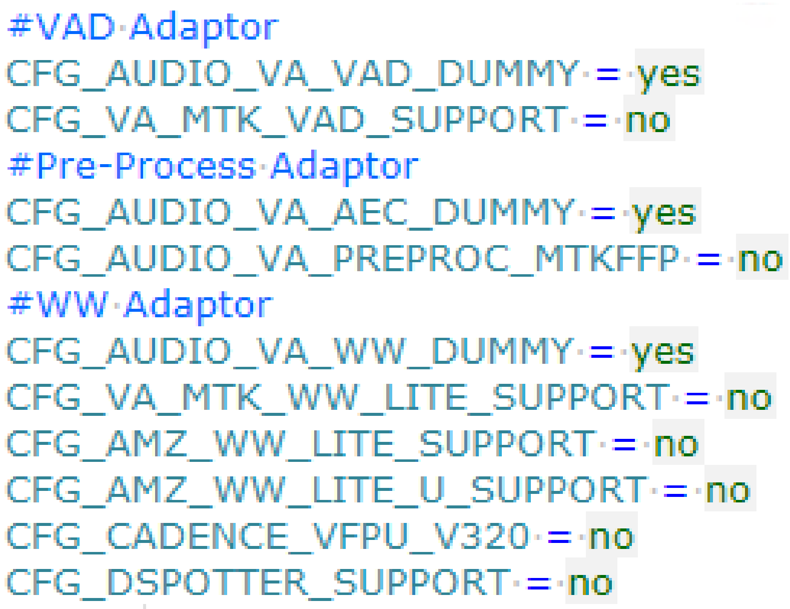
```
+--- HIFI4
|  +--- project
```

position: project/mtxxxx/config/$project/HIFI4_A/AudioConfig.mk

Add CFG_XXX_YYY_SUPPORT = yes to the above position.

6 VA Algorithm Checking

Voice Assistant algorithm related configurations are defined in AudioConfig.mk under the path: tinysys/adsp/HIFi4/project/mt7933/config/iot7933bga-hadron/AudioConfig.mk



```
#VAD Adaptor
CFG_AUDIO_VA_VAD_DUMMY = yes
CFG_VA_MTK_VAD_SUPPORT = no
#Pre-Process Adaptor
CFG_AUDIO_VA_AEC_DUMMY = yes
CFG_AUDIO_VA_PREPROC_MTKFFP = no
#WW Adaptor
CFG_AUDIO_VA_WW_DUMMY = yes
CFG_VA_MTK_WW_LITE_SUPPORT = no
CFG_AMZ_WW_LITE_SUPPORT = no
CFG_AMZ_WW_LITE_U_SUPPORT = no
CFG_CADENCE_VFPU_V320 = no
CFG_DSPOTTER_SUPPORT = no
```

Figure 12 Dummy Algorithm

To avoid build failure, if you do not have the license for MediaTek or other third-party vendor’s algorithm , you should enable the dummy process for voice assistant processing in AudioConfig.mk, and set the following values to no:

CFG_AUDIO_VA_VAD_DUMMY = yes: enable dummy VAD

CFG_AUDIO_VA_AEC_DUMMY= yes: enable dummy pre-processing

CFG_AUDIO_VA_WW_ DUMMY = yes: enable dummy Wake Word detection

Exhibit 1 Terms and Conditions

Your access to and use of this document and the information contained herein (collectively this “Document”) is subject to your (including the corporation or other legal entity you represent, collectively “You”) acceptance of the terms and conditions set forth below (“T&C”). By using, accessing or downloading this Document, You are accepting the T&C and agree to be bound by the T&C. If You don’t agree to the T&C, You may not use this Document and shall immediately destroy any copy thereof.

This Document contains information that is confidential and proprietary to MediaTek Inc. and/or its affiliates (collectively “MediaTek”) or its licensors and is provided solely for Your internal use with MediaTek’s chipset(s) described in this Document and shall not be used for any other purposes (including but not limited to identifying or providing evidence to support any potential patent infringement claim against MediaTek or any of MediaTek’s suppliers and/or direct or indirect customers). Unauthorized use or disclosure of the information contained herein is prohibited. You agree to indemnify MediaTek for any loss or damages suffered by MediaTek for Your unauthorized use or disclosure of this Document, in whole or in part.

MediaTek and its licensors retain titles and all ownership rights in and to this Document and no license (express or implied, by estoppels or otherwise) to any intellectual propriety rights is granted hereunder. This Document is subject to change without further notification. MediaTek does not assume any responsibility arising out of or in connection with any use of, or reliance on, this Document, and specifically disclaims any and all liability, including, without limitation, consequential or incidental damages.

THIS DOCUMENT AND ANY OTHER MATERIALS OR TECHNICAL SUPPORT PROVIDED BY MEDIATEK IN CONNECTION WITH THIS DOCUMENT, IF ANY, ARE PROVIDED “AS IS” WITHOUT WARRANTY OF ANY KIND, WHETHER EXPRESS, IMPLIED, STATUTORY, OR OTHERWISE. MEDIATEK SPECIFICALLY DISCLAIMS ALL WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, FITNESS FOR A PARTICULAR PURPOSE, COMPLETENESS OR ACCURACY AND ALL WARRANTIES ARISING OUT OF TRADE USAGE OR OUT OF A COURSE OF DEALING OR COURSE OF PERFORMANCE. MEDIATEK SHALL NOT BE RESPONSIBLE FOR ANY MEDIATEK DELIVERABLES MADE TO MEET YOUR SPECIFICATIONS OR TO CONFORM TO A PARTICULAR STANDARD OR OPEN FORUM.

Without limiting the generality of the foregoing, MediaTek makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does MediaTek assume any liability arising out of the application or use of any product, circuit or software. You agree that You are solely responsible for the designing, validating and testing Your product incorporating MediaTek’s product and ensure such product meets applicable standards and any safety, security or other requirements.

The above T&C and all acts in connection with the T&C or this Document shall be governed, construed and interpreted in accordance with the laws of Taiwan, without giving effect to the principles of conflicts of law.