# AN13201

## i.MX 8M Mini Heterogenous Low Power Voice Control Solution

Rev. 0 — 04/2021 Application Note

Contents



# 1 Introduction

## 1.1 Purpose

This application note shows how to deploy low-power local voice on an NXP SoC. Our partner RetuneDSP have been working for years in the field of algorithms for audio processing, voice recognition, and keyword spotting using machine learning to deliver the best-in-class solution ready to be deployed in a substantial number of applications and products. This application note demonstrates how to integrate this technology using NXP products while minimizing power consumption. By leveraging the i.MX heterogenous computing architecture, the following use case enables you to continuously spot for a keyword when the main system is set into the Deep Sleep Mode (DSM) in the power consumption constraints of home appliances.

The solution presented in this document targets the i.MX 8M Mini platforms using both the Cortex-A53 and Cortex-M4 cores. The solution can be applied to other i.MX platforms, but it must be adapted, depending on which software component manages the power states and handles the access to the resources shared between the Cortex-A53 and Cortex-M4 cores. For this platform, power management and resource access are managed in the Arm Trusted Firmware (ATF), Linux BSP, and Cortex-M4 app.

## 1.2 Definitions, acronyms, and abbreviations

Table 1. Acronyms and abbreviations

| Acronym | Meaning |
|---|---|
| SoC | System on Chip |
| BSP | Board Support Package |
| DSM | Deep Sleep Mode |
| I2S | Integrated Interchip Sound |
| RTOS | Real-Time Operating System |
| GPC | General Power Controller |
| ATF | Arm Trusted Firmware |
| MU | Messaging Unit |
| SAI | Synchronous Audio Interface |
| KWS | Keyword Spotting |

*Table continues on the next page...*

Table 1. Acronyms and abbreviations (continued)

| Acronym | Meaning |
|---|---|
| DTS | Device Tree Source |
| TCM | Tightly Coupled Memory |
| HMI | Human Machine Interface |

# 2 Solution overview

## 2.1 Context

This application explains how to deploy an application using voice command to return from the Suspend-to-Ram mode of i.MX 8M Mini. It focuses on waking up the Linux OS running on the Cortex-A53 core when a keyword is detected by the Cortex-M4 core.

The i.MX 8M Mini can be scaled to use one, two, or four Arm® Cortex®-A53 and Cortex-M4 cores for the Heterogeneous Multicore Processing (HMP). The Arm Cortex-A53 core runs at up to 1.8 GHz per core, delivering outstanding system performance. Delivered in an advanced low-power process, the core complex is optimized for fanless operation, low thermal system cost, and long battery life. The Cortex-A cores can be in the clock-gating, low-voltage mode while the Cortex-M4 subsystem performs low-power, real-time system monitoring. Thereby, the following solution is implemented:

- The Cortex-M core processing with the Cortex-A core in the DSM. The Cortex-M core processing implies recording with an I2S microphone and performing keyword spotting using a voice recognition library under the FreeRTOS.
- The Cortex-M core should be able to wake up the Cortex-A core when a keyword is detected.

## 2.2 Solution overview

The implemented solution is shown in Figure 1.

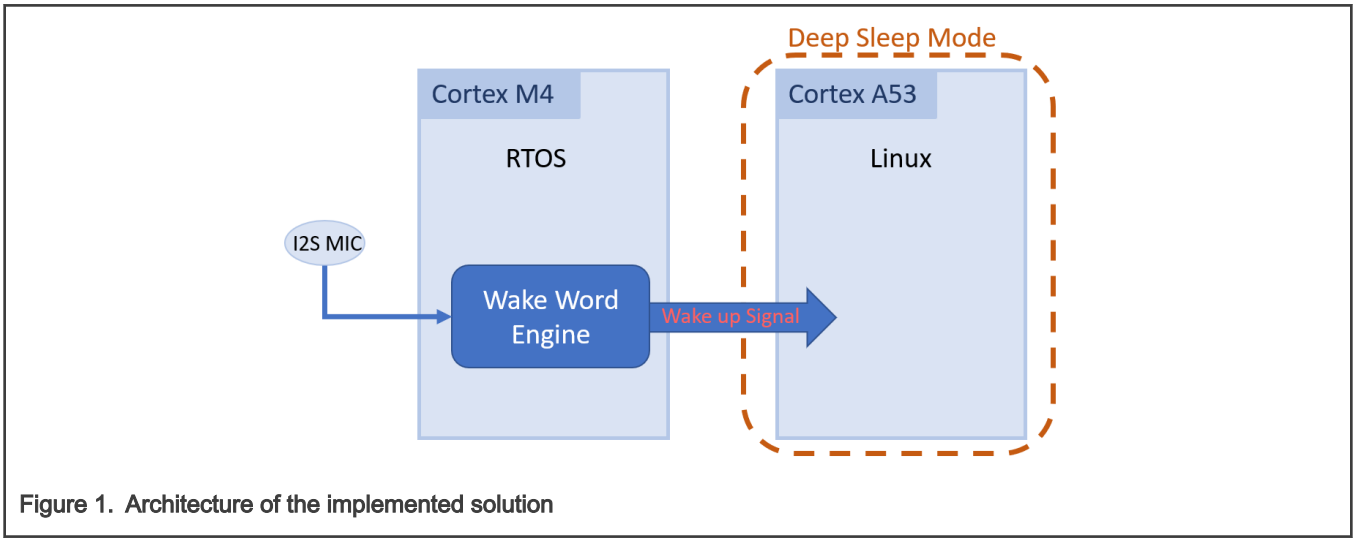


Figure 1. Architecture of the implemented solution

As shown below, the application runs on the Cortex-M core. This application integrates a wake-word engine that recognizes the keyword captured by the I2S microphone. When a keyword is detected, the Cortex-M core sends a wake-up signal to the Cortex-A core to exit from the DSM.

### 2.2.1 Entering CPU DSM

On the Cortex-A side, the Linux OS is put into the DSM (also called suspend-to-ram) using the following command:

```
"$: echo mem > /sys/power/state"
```

For more information about the power states and their management, see the "General Power Controller (GPC)" section in the *i.MX 8M Mini Applications Processor Reference Manual* (document IMX8MMRM).

### 2.2.2 Cortex-M resource management

For this application, the Cortex-M core uses resources like the SAI and UART, which are shared with the Cortex-A core. The Cortex-A core must release its control over these IPs, so that they are not impacted by the DSM state and to allow the Cortex-M core to fully manage them.

On the Linux OS side, this is achieved by updating the *imx8mm-evk-rpmsg.dts* file. This DTB file is specially designed for heterogenous use cases. The M4 core peripherals used by the current application were added to the list of IPs needed by the Cortex-M core that are disabled on the Cortex-A core (for example, sound-ak4458, sound-ak5558, sound-spdif, sound-spdif, sound-spdif, and spdif1).

The settings required on the M4 side are as follows:

- Register the resources to be used by the M4 core with the Resource Domain Controller (RDC). See the M4 core project source file *freertos_hello.c*.

```
void Peripheral_RdcSetting(void)
{
    rdc_domain_assignment_t assignment = {0};
    rdc_periph_access_config_t periphConfig;
    assignment.domainId = BOARD_DOMAIN_ID;

    RDC_GetDefaultPeriphAccessConfig(&periphConfig);
    /* Do not allow the A53 domain(domain0) to access the following peripherals. */
    periphConfig.policy = RDC_DISABLE_A53_ACCESS;

    periphConfig.periph = kRDC_Periph_UART4;
    RDC_SetPeriphAccessConfig(RDC, &periphConfig);

    periphConfig.periph = kRDC_Periph_SAI3_ACCESS;
    RDC_SetPeriphAccessConfig(RDC, &periphConfig);
…
```

- Keep the clocks active in the DSM. See the M4 core project source file *freertos_hello.c*.

```
/*
     * In order to wakeup M4 from LPM, all PLLCTRLs need to be set to "NeededRun"
     */
    for (i = 0; i < 39; i++)
    {
        CCM->PLL_CTRL[i].PLL_CTRL = kCLOCK_ClockNeededRun;
    }
```

### 2.2.3 Enabling M4 low-power audio with Cortex-A in the DSM

For the i.MX 8M Mini, the Low Power Audio (LPA) feature allows the Cortex-M core to be active while the Cortex-A core is in the DSM. The Arm Trusted Firmware (ATF) handles the entry to and exit from the low-power mode, as well as resource access (see gpc_common.c and imx8mm/ gpc.c). Reserved register SRC_GPR9 is used to allow for external components (that means, the M4 application) to specify the low-power mode. In this way, the ATF is informed that the Cortex-M core must be kept active while the Cortex-A is in the DSM.

- In the proposed implementation, the ‘ServiceBusy’ flag set in the M4 app indicates to the ATF that the M4 low-power audio mode is active (see the M4 project source file *freertos_hello.c*).

**Table 2. M4 application and ATF**

| M4 application | ATF |
| --- | --- |
| `#define ServiceFlagAddr SRC->GPR9`<br>`#define ServiceBusy (0x5555U)`<br>`#define ServiceIdle (0x0U)`<br>…<br>`void app_task(void *param)`<br>`{`<br>`ServiceFlagAddr = ServiceBusy;`<br>… | `#define M4_LPA_ACTIVE 0x5555`<br>`#define DSP_LPA_ACTIVE 0xD`<br>`#define DSP_LPA_DRAM_ACTIVE 0x1D`<br>`#define M4_LPA_IDLE 0x0`<br>…<br>`bool imx_m4_lpa_active(void)`<br>`{`<br>`uint32_t lpa_status;`<br>`lpa_status = mmio_read_32(IMX_SRC_BASE +`<br>`LPA_STATUS);`<br>`return (lpa_status == M4_LPA_ACTIVE ||`<br>`lpa_status == DSP_LPA_ACTIVE || lpa_status ==`<br>`DSP_LPA_DRAM_ACTIVE);`<br>`}`<br>… |

### 2.2.4 Optimizations for further power saving

Using the LPA mechanism, the DDR and the list of predefined peripherals are kept active. For the list of peripherals, see the "struct pll_override" in the gpc_common.c file.

To further reduce power consumption, the ATF implementation is updated to decrease the LPDDR frequency and lower the bus frequency of unused peripherals in the low-power mode.

See the updates made in the *plat/imx/imx8m/imx8m_psci_common.c - imx_domain_suspend and imx_domain_suspend_finish* and *plat/imx/imx8m/ddr/clock.c - bus_freq_dvfs* functions.

### 2.2.5 Wake-up mechanism

One option to trigger the wake-up signal for the Cortex-A core is to generate the MU interrupt via RPMSG (see the "rpmsg-ping-pong" demo). The drawback of this approach is that the RPMSG needs the DDR to be active. To keep it functioning at a low frequency to save power, the solution described in this document directly uses a platform-interrupt signal triggered from the M4 side, which bypasses the DDR.

- Enable the MU interrupt as the wake trigger in the ATF (see gpc_common.c):

```
/* enable the MU wakeup */
if (imx_is_m4_enabled())
    mmio_clrbits_32(IMX_GPC_BASE + gpc_imr_offset[last_core] + 0x8, BIT(24));
```

- The Cortex-M core generates the MU interrupt when the keyword is detected (see the M4 project source file *freertos_hello.c*):

```
MU_SendMsg(MUB, 1, 2); //wake up Cortex A
```

### 2.2.6 Cortex-M implementation

On the Cortex-M side, the keyword spotting application is running. It is divided into two parts: the part performing the microphone recording and the part performing the keyword detection.

To perform the recording, the Synchronous Audio Interface (SAI) is used. This module is highly configurable and allows users to process audio in different audio formats, such as I2S, codec/DSP, and TDM. In this case, the audio data is coming from the microphone which sends the I2S data through the board pins. The sample rate for the recording is set to 16 kHz. Using the SAI driver, the audio data is recovered into a buffer and then placed into a circular buffer that is used for the KWS processing. It also allows to pass the audio data to the Wolfson audio codec, which allows the users to use a headphone and hear the recording of the microphone through the jack connector of the i.MX 8M Mini.

The advantage of using a circular buffer to store audio data is that it allows the user to have the data ready for any type of the KWS engine. The SAI constrains the user to get a chunk of audio data in a restricted buffer size. In the presented case, the SAI buffer has 256 bytes and the KWS engine used takes an input of 800 bytes. Using a circular buffer enables us to pass the audio to the KWS engine without losing data.

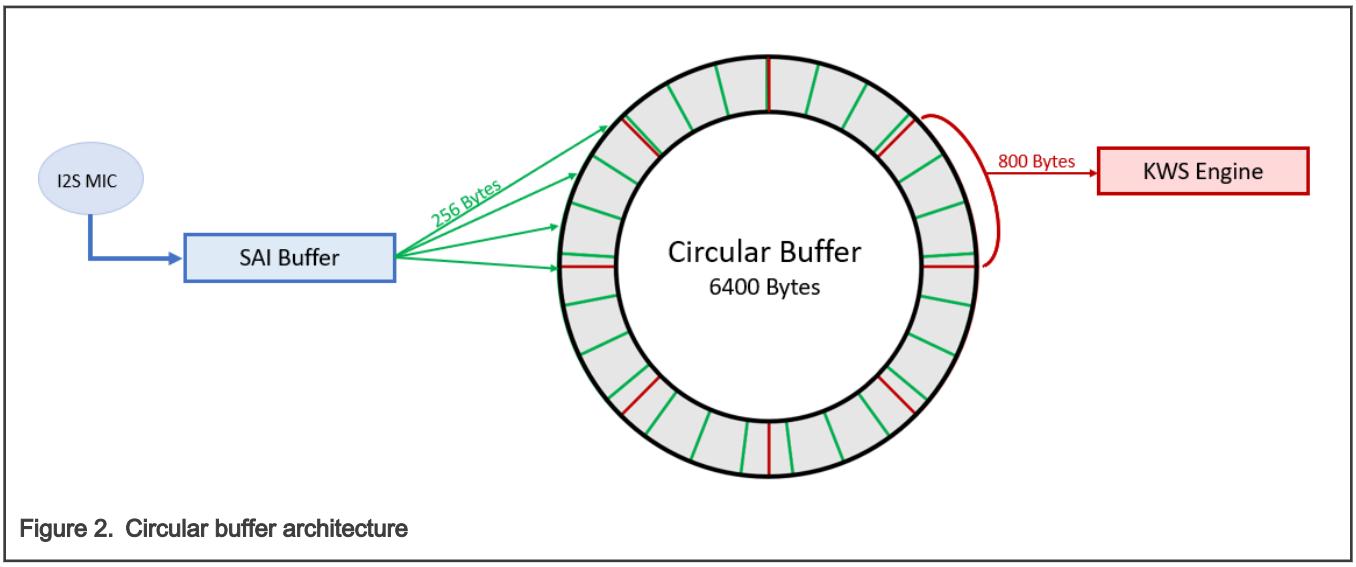


**Figure 2. Circular buffer architecture**

This solution integrates the Voicespot™ wake-word engine from NXP's partner RetuneDSP. RetuneDSP specialize in algorithms for audio signal processing.

The main features of VoiceSpot™ are:

- Small memory footprint
- Low computational complexity
- Low latency
- High precision and quality

The small memory footprint is a decisive criterion to meet the lowest power consumption. The Retune DSP VoiceSpot library has a total memory consumption of around 90 kB (code + data + model) when using a standard model size of 33 kB.

When the data is processed by the KWS engine and a keyword is detected, a wake-up signal is sent to get the Linux OS out of the suspend-to-RAM mode.

The execution time to fill the KWS engine buffer and process it is 11,86 ms.

To put everything together, the following is the list of files that need modifications:

In the ATF:

- *plat/imx/imx8m/ddr/clock.c*
- *plat/imx/imx8m/ddr/dram.c*
- *plat/imx/imx8m/gpc_common.c*
- *plat/imx/imx8m/imx8m_psci_common.c*
- *plat/imx/imx8m/imx8mm/include/platform_def.h*

- *plat/imx/imx8m/include/dram.h*

In the kernel of the Linux OS:

- *arch/arm64/boot/dts/freescale/imx8mm-evk-rpmsg.dts*

# 3 Hardware setup

## 3.1 Hardware materials needed

For this application, the board used is the i.MX 8M Mini EVK platform (8MMINILPD4-EVK). To run this demo, you need:

- 1 i.MX 8MMini Kit.
- 1 ribbon, 4 female-female wires, and a 60-pin connector to connect the microphone to the board.
- 1 omnidirectional microphone with an I2S digital output. In the presented solution, https://invensense.tdk.com/wp-content/uploads/2015/02/INMP441.pdf is used.
- Headphones (optional, used to test the recording on the M4 - everything recorded by the microphone is played through the headphones).

## 3.2 Hardware setup

The microphone has 6 pins that must be connected to the board.

**Table 3. Microphone pin function and description**

| Name | Function |
|---|---|
| SCK | Continuous serial clock. |
| SD | Serial data. |
| WS | Word select. |
| L/R | Left/right channel selection. When set to low, it is the left channel. When set to high, it is the right channel. |
| GND | Ground. |
| VDD | Power, from 1.8 V to 3.3 V. |

SAI5 is used for record and playback on the Cortex-M core. Connect the I2S microphone to the board using the following pins:

On jumper J1003:

- Pin 40 (connector) <-> SCK (microphone)
- Pin 38 (connector) <-> SD (microphone)
- Pin 39 (connector) <-> L/R (microphone)
- Pin 1 (connector) <-> VDD (microphone)
- Pin 9 (connector) <-> GND (microphone)

On jumper J1001 (using a 60-pin connector with a ribbon):

- Pin 33 (connector) <-> WS (microphone)

Figure 3 shows the schematic of the i.MX 8M Mini and the connections that must be made.

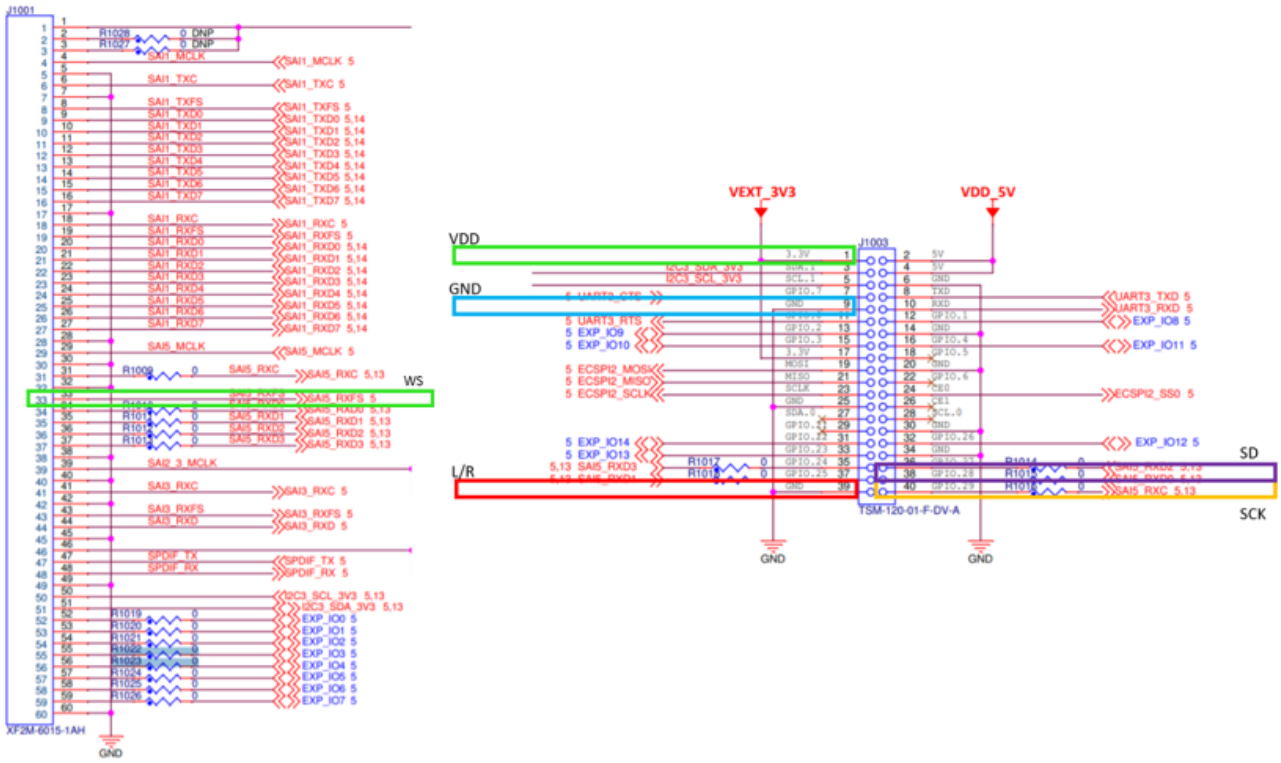


Figure 3. Microphone connection schematics

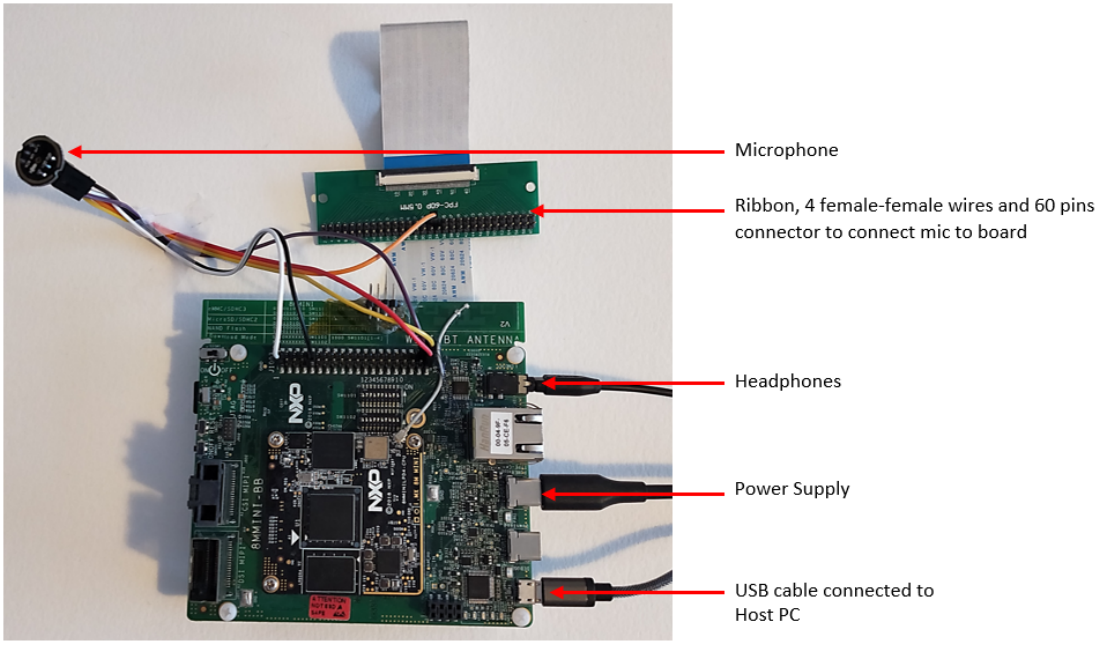


Figure 4. Hardware setup

Optionally, you can plug headphones to the jack connector. They are used to test the recording.

## 3.3 Board settings

For information on how to set up the i.MX 8M Mini and how to get started with the EVK, see the *i.MX 8M Mini EVK Quick Start Guide* (document 8MMINIEVKQSG).

# 4 Software environment

## 4.1 Software for Cortex-A core

- Create a successful build environment for i.MX 8M Mini 5.4.47_2.2.0. Any type of image is fine and the changes are done in the kernel drivers and in the ATF (Arm Trusted Firmware).
- Apply the ATF patch. Get the “atf_wake_word_low_power_demo.patch” ATF patch from https://source.codeaurora.org/external/imxsupport/low-power-voice-control-application-note/tree/wake_word_low_power_demo.

```
$: cd ${build_8MM_5_4_47_2_2_0} /tmp/work/aarch64-mx8mm-poky-linux/imx-atf/2.2+gitAUTOINC+c949a888e9-r0/git
$: git apply atf_wake_word_low_power_demo.patch
```

- Apply the kernel patch. Get the “kernel_wake_word_low_power_demo.patch” kernel patch from https://source.codeaurora.org/external/imxsupport/low-power-voice-control-application-note/tree/wake_word_low_power_demo.

```
$: cd ${build_8MM_5_4_47_2_2_0}/tmp/work/imx8mmevk-poky-linux/linux-imx/5.4-r0/git
$ git apply kernel_wake_word_low_power_demo.patch
```

- Rebuild the image (make sure to enable the Yocto build environment (i.e., source setup-environment build-xwayland):

```
$: bitbake -c cleanall imx-image-multimedia
$: bitbake -c cleanall imx-boot
$: bitbake -c cleanall kernel-pcitest
$: bitbake -f -c compile imx-atf
$: bitbake -f -c compile linux-imx
$: bitbake imx-image-multimedia
```

- Get the image in the *{build_8MM_5_4_47_2_2_0}/build_xwayland/tmp/deploy/images/imx8mmevk/ named imx-image-multimedia-imx8mmevk.rootfs.wic.bz2* file and flash it to an SD card:

```
$: bunzip2 -c imx-image-multimedia-imx8mmevk.wic.bz2 |sudo dd of=/dev/sdb bs=1M && sync
```

## 4.2 Software for Cortex-M core

### 4.2.1 Downloading MCUxpresso SDK IDE

Download the MCUXpresso SDK IDE for the i.MX 8M Mini SDK_2.8.0_EVK-MIMX8MM. Have at least the GCC Arm toolchain and include the FreeRTOS:

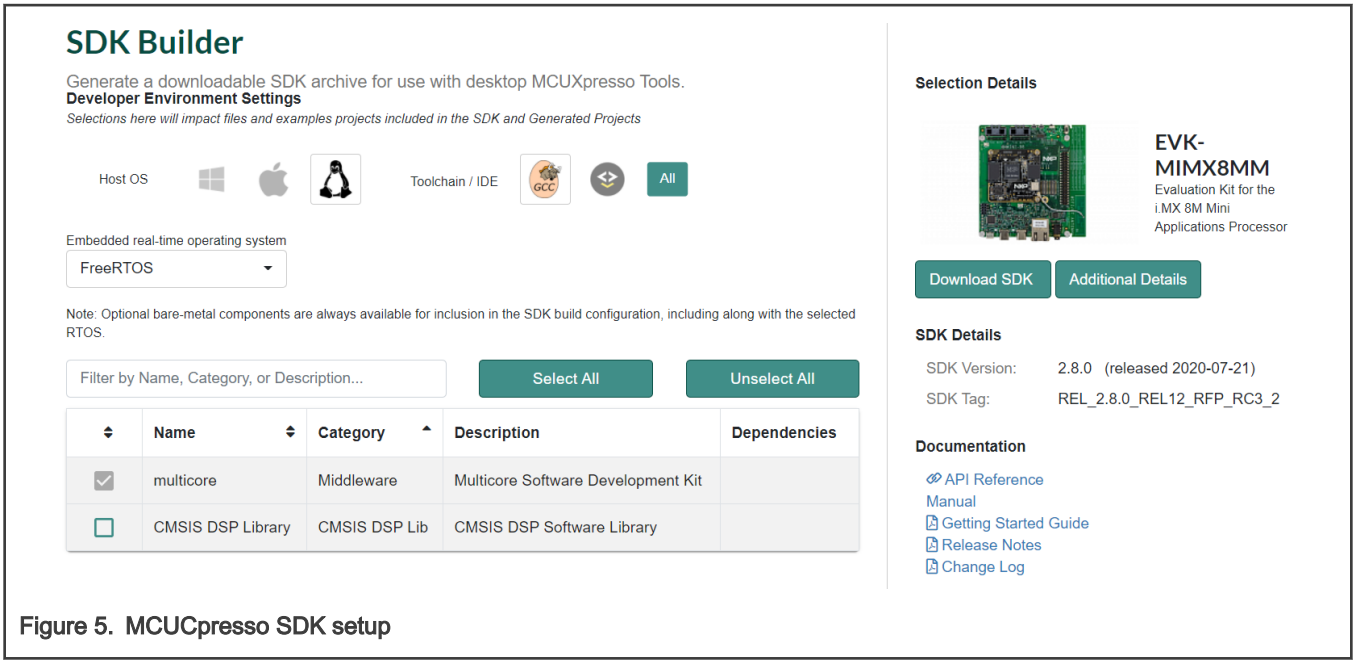


**Figure 5. MCUCpresso SDK setup**

Download the GNU Tools Arm Embedded/8 2018-q4-major to compile from https://developer.arm.com/tools-and-software/open-source-software/developer-tools/gnu-toolchain/gnu-rm/downloads/8-2018-q4-major.

### 4.2.2 Building the Cortex-M4 application

Download the application sources in the *wake_word_low_power_demo* folder from https://source.codeaurora.org/external/imxsupport/low-power-voice-control-application-note/tree/wake_word_low_power_demo.

Copy the application to the *$MCUXpressoSDK_ROOT\boards\evkmimx8mm\rtos_examples* folder.

The wake-word low-power project shows how to deploy the low-power local voice on a Cortex-M core, integrating the Voicespot™ wake-word engine from NXP's partner RetuneDSP. This application shows an example of the integration code. The VoiceSpot library is not linked to this application, so the build application cannot run as it is. The provided application is a template ready for the RetuneDSP Library integration.

The build instructions for the Linux OS are as follows:

- Go to the *${MCUXpressoSDK_ROOT}\boards\evkmimx8mm\rtos_examples\ wake_word_low_power_demo\armgcc* folder.
- Export the toolchain:

```
$ export ARMGCC_DIR=$(toolchain_dir) /gcc-arm-none-eabi-8-2019-q4-major
$ export PATH= $PATH: $(toolchain_dir) /gcc-arm-none-eabi-8-2018-q4-major/bin
```

- Call *build_debug.sh* (or *build_release.sh*) file to build. Make sure to use these files to have the M4 application in the TCM and not in the DDR or flash. This creates the executable in the *wake_word_low_power_demo\armgcc\debug\wake_word_low_power_demo.bin* (or *wake_word_low_power_demo\armgcc\release\ wake_word_low_power_demo.bin*) file.
- Copy the binary file to the boot partition of your board's SD card.

# 5 Running the application

After preparing the hardware setup, creating the Linux OS image, flashing it to an SD card, creating the M4 application binary, and copying it to the boot partition of the SD card, perform the following steps:

- Connect a 12-V power supply to the board and switch SW101 to power on the board.

- Connect a USB cable between the host PC and the J901 USB port on the target board.
- Open two serial terminals for the A53 and M4 cores with the following settings:
  - 115200 baud rate
  - 8 data bits
  - No parity
  - One stop bit
  - No flow control
- Insert an SD card into the board.
- Start serial consoles for the Cortex-A and Cortex-M cores.
- Power on the board and make sure to go into the u-boot in the Cortex-A console. Then follow the instructions displayed in the console:

```
U-BOOT
u-boot=> edit fdt_file
edit: fsl-imx8mm-evk-rpmsg.dtb
u-boot=> saveenv
u-boot=> fatload mmc 1:1 0x48000000 wake_word_low_power_demo.bin; cp.b 0x48000000 0x7e0000 20000;
u-boot=> bootaux 0x7e0000
u-boot=> boot

The text below should be displayed in the M4 console when the application starts successfully:

#################### WAKE WORD LOW POWER DEMO ####################
Build Time: Jan 27 2021--16:41:31
rdspVoiceSpot_CreateControl: voicespot_status = 0
rdspVoiceSpot_CreateInstance: voicespot_status = 0
rdspVoiceSpot_EnableAdaptiveThreshold: voicespot_status = 0
rdspVoiceSpot_OpenInstance: voicespot_status = 0
rdspVoiceSpot_SetParametersFromBlob: voicespot_status = 0
VoiceSpot library version: 0.15.1.1600757651
VoiceSpot model version: 0.3.0
VoiceSpot model string: Alexa EN-US
Say Alexa to wake up the Cortex-A
```

On the Cortex-A side, use the following commands:

```
$: modprobe imx-rpmsg-pingpong
$: echo mem > /sys/power/state
```

The “echo mem > /sys/power/state” command puts the Cortex-A core into the DSM.

On the M4 side, you can hear the recording from the microphone if headphones are plugged into the jack connector. To wake up the Cortex-A core, say the “Alexa” keyword. When the keyword is detected, the Cortex-A wakes up and the following log is displayed in the M4 terminal:

```
Trigger event found: class_string = Alexa, Score = 195
Set mu interrupt MU_SendMsg
```

# 6 Power measurement

## 6.1 Hardware setup

In Table 4, you can see all the voltage supply rails used on the EVK board. When some modules are not enabled, the power supplies might be shut down by software.

**Table 4. Power rails**

| SEQ | Power rail | Regulator | Value/V |
|---|---|---|---|
| 0 | DCDC_5V | Discrete | 5 |
| 1 | NVCC_SNVS_1V8 | PCA9450A LDO1 | 1.8 |
| 2 | VDD_SNVS_0V8 | PCA9450A LDO2 | 0.8 |
| 3 | RTC_RESET_B | PCA9450A | - |
| 4 | CLK_32K_OUT | PCA9450A | - |
| **5** | **VDD_SOC_0V8** | **PCA9450A BUCK1** | **0.85** |
| **6** | **VDD_DRAM_PU_0V9** | **PCA9450A BUCK3** | **0.85/0.9/0.95**[1] |
| 7 | VDD_PHY_0V9 | PCA9450A LDO4 | 1.2 |
| **8** | **VDD_ARM_0V9** | **PCA9450A BUCK2** | **0.85/0.95/1.0**[2] |
| 9 | VDDA_1V8 | PCA9450A LDO3 | 1.8 |
| 10 | VDD_1V8/NVCC_1V8/NVCC_ENET | PCA9450A BUCK5 | 1.8 |
| **11** | **NVCC_DRAM_1V1** | **PCA9450A BUCK6** | **1.1**[3] |
| 12 | NVCC_3V3 | PCA9450A BUCK4 | 3.3 |
| 13 | VDD_PHY_1V2 | Discrete | 1.2 |
| 14 | NVCC_SD2 | PCA9450A LDO5 | 3.3/1.8 |
| 15 | POR_B | PCA9450A | - |

1. *The PCA9450A BUCK3 default output voltage is 0.85 V for the DDRC at 1 GHz. The software changes it to 0.9 V for the DDRC at 1.2 GHz and 0.95 V for the DDRC at 1.5 GHz in the SPL before the DDR initialization.*
2. *The PCA9450A BUCK2 default output voltage is 0.85 V for the A53 at 1.2 GHz. The software changes it to 0.95 V for the A53 at 1.6 GHz and 1.0 V for the A53 at 1.8 GHz.*

The measurements are taken for the following power-supply domains:

- **VDD_ARM**: Arm® Cortex®-A53 Mini cores' supply
- **VDD_SOC**: SoC power supply
- **VDD _GPU_DRAM**: GPU, DRAM controller and PHY digital logic, and PLL power supply
- **NVCC_DRAM**: DRAM IO power supply (including an external DDR device)

The DDR I/O is supplied from the NVCC_DRAM, which provides power for the DDR I/O pads. The target voltage for this supply is 1.1 V (LPDDR4).

The measurements were performed using the 34470A 6½ digital multimeter.

## 6.2 Power results

The application has a total average power consumption of ~301 mW. Figure 6 shows the detailed power consumption for the example application.

| Supply Domain | Voltage(V) | | I(A) | | P(W) | |
|---|---|---|---|---|---|---|
| | peak | avg | peak | avg | peak | avg |
| VDD_ARM(L6) | 1.007941 | 1.00771 | 0.058735 | 0.058657 | 0.059201 | 0.059109 |
| VDD_SOC(L5) | 0.857184 | 0.856866 | 0.138296 | 0.13567 | 0.118545 | 0.116251 |
| VDD_GPU_VPU_DRAM(L10) | 0.984602 | 0.984463 | 0.069819 | 0.069474 | 0.068744 | 0.068394 |
| NVCC_DRAM(L15) | 1.100269 | 1.100068 | 0.055445 | 0.052134 | 0.061005 | 0.057351 |
| Total | | | | | 0.307495 | 0.301105 |

Figure 6. Power measurement

# 7 Conclusion

Using the best-in-class RetuneDSP VoiceSpot technology and leveraging the i.MX 8MM multicore heterogenous architecture, this solution enables any customer to implement a power efficient mechanism to keep the voice HMI active when the processor is in the deep-sleep mode. This is the first cornerstone of a proper voice implementation on an application processor. From there, the i.MX architecture is flexible enough to enable a complete voice solution by adding further and more advanced voice processing software like RetuneDSP's VoiceSeeker to be run on the A53 core when the Linux OS is up and running, getting a full local voice support solution without sacrificing neither power consumption nor quality.

The i.MX 8M product family is an optimal device to enable voice on any Linux OS-based products.

# 8 References

- *i.MX 8M Mini Power Consumption Measurement* (document AN12410)
- *Implement Low-Power Audio on i.MX8M* (document AN12195)
- The "Low-Power Audio on i.MX8M" application in the MXUXpresso SDK IDE in the *${MCUXpressoSDK_ROOT}\boards\evkmimx8mm\demo_projects\sai_low_power_audio* folder
- RetuneDSP Contact:

  Chris Welsh

  Director of Business Development

  Partner, Retune DSP

  +1 317-506-3881

  www.retune-dsp.com

# 9 Revision history

Table 5. Revision history

| Revision number | Date | Substantive changes |
|---|---|---|
| 0 | 04/2021 | Initial release |

*How To Reach Us*

**Home Page:**

nxp.com

**Web Support:**

nxp.com/support



For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

**Date of release: 04/2021**

**Document identifier: AN13201**

arm