AN13970

Running Zephyr RTOS on Cadence Tensilica HiFi 4 DSP

Rev. 4.0 — 28 January 2025 **Application note**

Document information

Information	Content
Keywords	AN13970, i.MX 8M Plus, i.MX 8ULP, i.MX 8 QuadMax, i.MX 8 QuadXPlus, HiFi 4 DSP, Zephyr, IPC, OpenAMP, remoteproc
Abstract	This document explains how to harness the power processing of HiFi 4 DSP available in the NXP i.MX 8, i.MX 8M Plus,i.MX 8ULP, and i.MX 8X product family processor by running Zephyr RTOS on the DSP.

1 Introduction

Running Zephyr on Arm Cortex-A or Cortex-M cores is widely discussed and there are many examples on how to implement it. However, many Cortex-based microcontroller units (MCUs) and microprocessor units (MPUs) have on-chip digital signal processors (DSPs) incorporated to offload compute-intensive tasks.

The Cadence Tensilica HiFi 4 DSP is one such example of a high-performance embedded DSP optimized for audio, voice, or neural network processing. This application note explains how to harness the power processing of the HiFi 4 DSP available in the NXP i.MX 8, i.MX 8M Plus and i.MX 8X product family, by running Zephyr Real-Time Operating System (RTOS) on the DSP; while Linux Operating System (OS) runs on the main Cortex-A core.

Using example applications, this document explains:

- How to launch the applications on the HiFi 4 DSP
- How the HiFi 4 DSP and the main processor core communicate with each other
- How to get the output of the applications

In this document, all the examples are explained using existing drivers and/or frameworks from Linux OS and Zephyr RTOS.

2 Hardware platform

This document exemplifies on i.MX 8M Plus processor, but the same is available on other platforms: i.MX 8ULP, i.MX 8QuadMax, i.MX 8QuadXPlus.

The 8MPLUSLPD4-EVK board is based on the NXP i.MX 8M Plus applications processor, which is composed of the following:

- 4 × Arm Cortex-A53 up to 1.8 GHz
- 1 × Arm Cortex-M7 up to 800 MHz
- Cadence Tensilica HiFi 4 DSP up to 800 MHz

Figure 1 shows the top view of the 8MPLUSLPD4-EVK board.

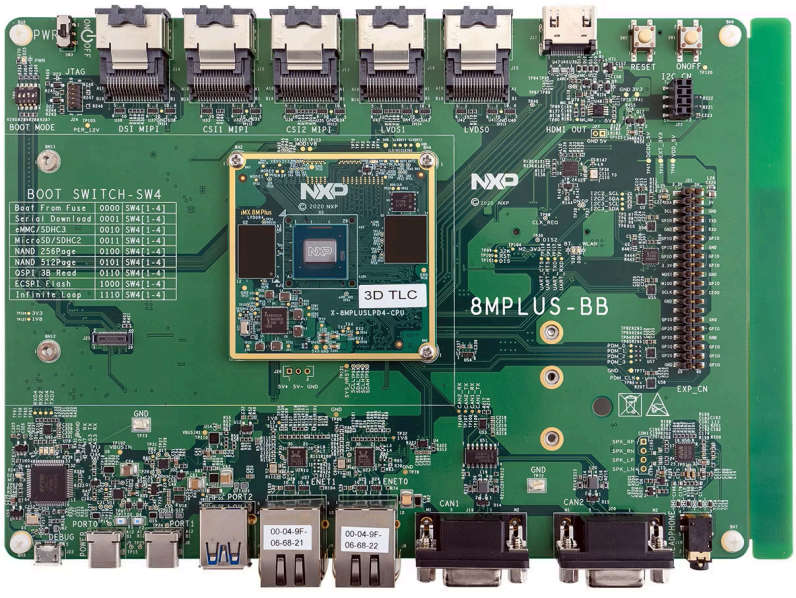

Figure 1. 8MPLUSLPD4-EVK board top view

For more details on the 8MPLUSLPD4-EVK board, see 8MPLUSLPD4-EVK.

3 Zephyr OS

The Zephyr Project is a scalable RTOS, which supports multiple hardware architectures, optimized for resource-constrained devices, and built with security in mind. It is based on a small-footprint kernel designed for use on resource-constrained systems.

NXP offers various evaluation and prototyping platforms that the Zephyr OS can support. Developers are able to tailor a solution easily to meet their needs using a true open source project with hardware, developer tools, and sensor and device drivers. Security enhancements with Zephyr OS enable easy implementation of device management, connectivity stacks, and file systems.

For more details on the Zephyr RTOS, visit www.zephyrproject.org/.

4 HiFi 4 audio DSP

The HiFi 4 audio engine is a highly optimized audio processor geared for efficient execution of audio and voice codecs and pre- and post-processing modules.

In Zephyr, the boards that support the audio DSP are as follows:

- For i.MX 8M Plus: `imx8mp_evk/mimx8ml8/adsp`
- For i.MX 8ULP: `imx8ulp_evk/mimx8ud7/adsp`
- For i.MX 8QuadMax: `imx8qm_mek/mimx8qm6/adsp`
- For i.MX 8QuadXPlus: `imx8qxp_mek/mimx8qx6/adsp`

4.1 Supported features

The `imx8mp_evk/mimx8ml8/adsp` board configuration supports the hardware features shown in Table 1.

Table 1. Supported hardware features

Interface	Controller	Driver/component
SYSTICK	On-chip	systick
CLOCK	On-chip	clock_control
PINMUX	On-chip	pinmux
UART	On-chip	serial port-polling

Note: *The port does not support other hardware features currently.*

The default configuration can be found in the defconfig file.

4.2 Connections and I/Os

The i.MX 8M Plus EVK board is tested with the pinmux controller configuration shown in Table 2.

Table 2. Connections

Board name	SoC name	Usage
UART4 RXD	UART4_TXD	UART console
UART4 TXD	UART4_RXD	UART console

4.3 System clock

The HiFi 4 DSP core is configured to run at 800 MHz clock speed.

4.4 Serial port

The i.MX 8M Plus SoC has four Universal Asynchronous Receiver/Transmitters (UARTs). Only `UART_4` is configured for the DSP console and the remaining UARTs are not used/tested.

5 Building and running Zephyr samples on HiFi 4 DSP

This section describes how to build and run Zephyr samples on HiFi 4 DSP using the following three applications:

- Section 5.1 "hello_world application"
- Section 5.2 "openamp_rsc_table application"
- Section 5.3 "number_crunching application"

5.1 hello_world application

The Zephyr's hello_world application is a simple sample that can be used with one of the Supported boards and prints `Hello World` to the console.

5.1.1 Load hello_world application on DSP

To load the `hello_world` application on the DSP, use the Linux remoteproc driver.

In Linux, a generic i.MX remoteproc driver and a DSP-specific driver (imx_dsp_rproc) are already available.

Because the application is running on the DSP, use the `imx_dsp_rproc` driver. To use the driver, enable `CONFIG_IMX_DSP_REMOTEPROC` in the Linux kernel.

5.1.2 Compile hello_world application

Compile the `hello_world` application in Zephyr for the i.MX 8M Plus DSP, that is, compile the application for the `imx8mp_evk/mimx8ml8/adsp` board.

Go to the `zephyr/` folder from `zephyrproject` and run the following:

```
~/zephyrproject/zephyr$ west build -p always -b imx8mp_evk/mimx8ml8/adsp
 samples/hello_world/
~/zephyrproject/zephyr$
~/zephyrproject/zephyr$ ls -la build/zephyr
total 4288
drwxr-xr-x 14 user nxp   4096 Oct 17 17:05 .
drwxr-xr-x  7 user nxp   4096 Oct 17 17:05 ..
drwxr-xr-x  5 user nxp   4096 Oct 17 17:05 arch
drwxr-xr-x  3 user nxp   4096 Oct 17 17:05 boards
drwxr-xr-x  5 user nxp   4096 Oct 17 17:05 cmake
-rw-r--r--  1 user nxp     64 Oct 17 17:05 .cmake.dotconfig.checksum
drwxr-xr-x  6 user nxp   4096 Oct 17 17:05 CMakeFiles
-rw-r--r--  1 user nxp  12355 Oct 17 17:05 cmake_install.cmake
-rw-r--r--  1 user nxp  39648 Oct 17 17:05 .config
...
-rw-r--r--  1 user nxp   2275 Oct 17 17:05 zephyr.dts
-rw-r--r--  1 user nxp    619 Oct 17 17:05 zephyr.dts.d
-rw-r--r--  1 user nxp 124460 Oct 17 17:05 zephyr.dts.pre
-rwxr-xr-x  1 user nxp 715896 Oct 17 17:05 zephyr.elf
-rw-r--r--  1 user nxp 408374 Oct 17 17:05 zephyr_final.map
-rw-r--r--  1 user nxp 408374 Oct 17 17:05 zephyr.map
-rwxr-xr-x  1 user nxp 717052 Oct 17 17:05 zephyr_pre0.elf
-rw-r--r--  1 user nxp 408886 Oct 17 17:05 zephyr_pre0.map
-rw-r--r--  1 user nxp   7273 Oct 17 17:05 zephyr.stat
```

The `zephyr.elf` file is used as the firmware to be loaded on the DSP.

The same command is used to compile the `hello_world` sample on i.MX 8ULP, i.MX 8QuadMax, i.MX 8QuadXPlus, only the board name is different. For example, for i.MX 8QuadMax uses the following board with the same command:

```
west build -p always -b imx8qm_mek/mimx8qm6/adsp samples/hello_world/
```

5.1.3 Run hello_world application on DSP

To run the `hello_world` application on HiFi 4 DSP from i.MX 8M Plus, perform the following steps:

1. Start the board with a specific device tree source (DTS).
   For i.MX 8M Plus use `imx8mp-evk-dsp.dtb`, for i.MX 8ULP use `imx8ulp-evk.dtb`, for i.MX 8QuadMax use `imx8qm-mek-rpmsg.dtb`, and for i.MX 8QuadXPlus use `imx8qxp-mek-rpmsg.dtb`.
   After inserting `imx_dsp_rproc` kernel module, check in the remoteproc id `/sys/class/remoteproc/` for the DSP:

   ```
   root@imx8mpevk:~# ls -la /sys/class/remoteproc/
   total 0
   drwxr-xr-x  2 root root 0 Mar  3 09:49 .
   drwxr-xr-x 90 root root 0 Mar  3 09:49 ..
   lrwxrwxrwx  1 root root 0 Mar  3 09:54 remoteproc0 -> ../../devices/
   platform/3b6e8000.dsp/remoteproc/remoteproc0
   root@imx8mpevk:~# cat /sys/class/remoteproc/remoteproc0/firmware
   imx/dsp/hifi4.bin
   root@imx8mpevk:~#
   ```

   Here, `remoteproc0`, which is for DSP, is used.
2. Check the firmware image on the board:

   ```
   root@imx8mpevk:~# ls -la /lib/firmware/imx/zephyr/
   total 1000
   drwxr-xr-x  2 root root   4096 Mar  9  2018 .
   drwxr-xr-x 10 root root   4096 Mar  9  2018 ..
   -rwxr-xr-x  1 root root  65424 Mar  9  2018 imx8mp_evk-hello_world-zephyr.elf
   -rwxr-xr-x  1 root root 221072 Mar  9  2018 imx8mp_evk-
   number_crunching_cmsisDSP-zephyr.elf
   -rwxr-xr-x  1 root root 147740 Mar  9  2018 imx8mp_evk-openamp_rsc_table-
   zephyr.elf
   -rwxr-xr-x  1 root root  78792 Mar  9  2018 imx8mp_evk-philosophers-
   zephyr.elf
   -rwxr-xr-x  1 root root  67536 Mar  9  2018 imx8mp_evk-synchronization-
   zephyr.elf
   -rwxr-xr-x  1 root root  76688 Mar  9  2018 imx8qm_mek-hello_world-zephyr.elf
   -rwxr-xr-x  1 root root  97336 Mar  9  2018 imx8qm_mek-openamp_rsc_table-
   zephyr.elf
   -rwxr-xr-x  1 root root  76688 Mar  9  2018 imx8qxp_mek-hello_world-
   zephyr.elf
   -rwxr-xr-x  1 root root  97336 Mar  9  2018 imx8qxp_mek-openamp_rsc_table-
   zephyr.elf
   -rwxr-xr-x  1 root root  72536 Mar  9  2018 imx8ulp_evk-hello_world-
   zephyr.elf
   root@imx8mpevk:~#
   ```

   The firmware must be present in `/lib/firmware` before the remoteproc driver is probed; however, it can also be given with an absolute path.
3. Insert `imx_dsp_rproc` kernel module:
   By default, the i.MX DSP remoteproc protocol waits for a "READY" reply from the remote processor. Because not all Zephyr sample applications (especially simple applications that do not use the mailbox)

send a "READY" reply, you must use the remoteproc module `imx_dsp_rproc` without waiting for a reply. The `imx_dsp_rproc` module is implemented using the kernel module parameter `no_maiboxes`, as shown below:

```
root@imx8mpevk:~# modinfo imx_dsp_rproc
filename:       /lib/modules/6.6.23-lts-next-06224-ga6e188bee815/kernel/
drivers/remoteproc/imx_dsp_rproc.ko
author:         Shengjiu Wang <shengjiu.wang@nxp.com>
description:    i.MX HiFi Core Remote Processor Control Driver
license:        GPL v2
...
depends:
intree:         Y
name:           imx_dsp_rproc
vermagic:       6.6.23-lts-next-06224-ga6e188bee815 SMP preempt mod_unload
 modversions aarch64
parm:           no_mailboxes:There is no mailbox between cores, so ignore
 remote proc reply after start, default is 0 (off). (int)
root@imx8mpevk:~#
```

By default, the `no_maiboxes` parameter is off; do not ignore the reply from the remote processor. Therefore, first check the `imx_dsp_rproc` parameter. If it is off, remove the module and insert it with the right parameter.

```
root@imx8mpevk:~# grep -H '' /sys/module/imx_dsp_rproc/parameters/*
/sys/module/imx_dsp_rproc/parameters/no_mailboxes:0      /* no_mailboxes
 param is off */
root@imx8mpevk:~#
root@imx8mpevk:~# rmmod imx_dsp_rproc    /* remove kernel module */
root@imx8mpevk:~#
root@imx8mpevk:~# modprobe imx_dsp_rproc no_mailboxes=1    /* insert kernel
 module with the right parameter */
root@imx8mpevk:~#
root@imx8mpevk:~# ls -la /sys/class/remoteproc/     /* insert kernel module
 with the right parameter */
total 0
drwxr-xr-x  2 root root 0 Feb 27 17:37 .
drwxr-xr-x 92 root root 0 Feb 27 17:27 ..
lrwxrwxrwx  1 root root 0 Feb 27 17:37 remoteproc0 -> ../../devices/
platform/3b6e8000.dsp/remoteproc/remoteproc0
root@imx8mpevk:~#
```

4. To load the firmware on DSP and run it, execute the following commands:

```
root@imx8mpevk:~# echo -n /lib/firmware/imx/zephyr/imx8mp_evk-hello_world-
zephyr.elf > /sys/class/remoteproc/remoteproc0/firmware
root@imx8mpevk:~# echo start > /sys/class/remoteproc/remoteproc0/state
```

5. To stop the firmware, use the following command:

```
root@imx8mpevk:~# echo stop > /sys/class/remoteproc/remoteproc0/state
root@imx8mpevk:~#
```

5.1.4 hello_world application output

To get the `hello_world` application output, perform the following steps:

1. Get the console through the UART.

2. Open a serial terminal on the fourth serial port:

```
user@developerpc:~# minicom -D /dev/ttyUSB3
```

Note: *Check the serial port number for the other targets.*

The terminal displays the following message (also shown in Figure 2):

```
Hello World! imx8mp_evk/mimx8ml8/adsp
*** Booting Zephyr OS build v4.0.0-3025-g52fd016274f3 ****
```

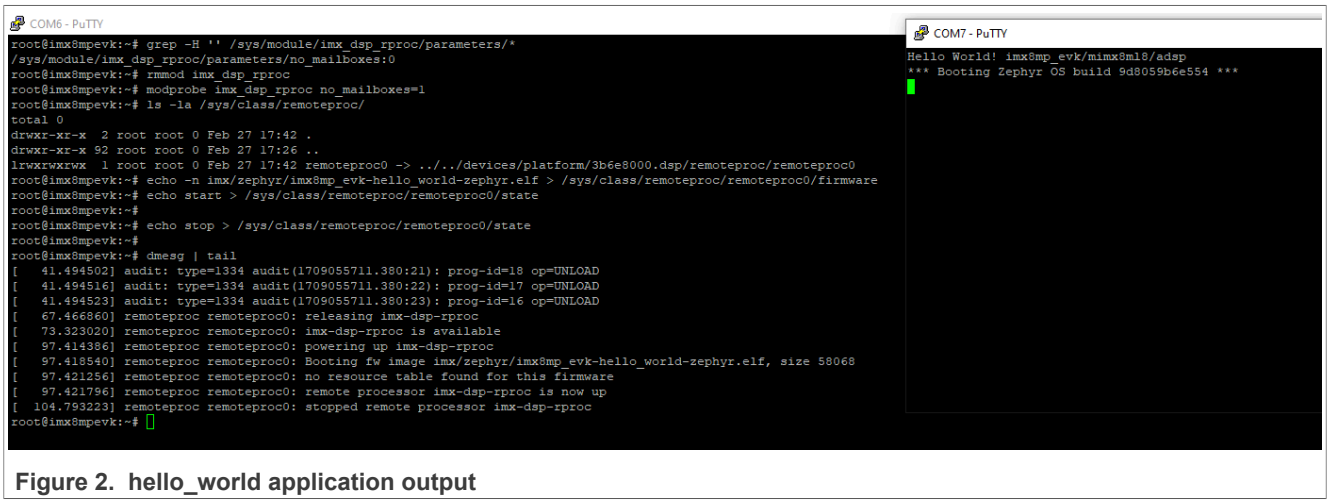

Figure 2. hello_world application output

To build and test any other sample, such as synchronization or philosophers, for any of the targets i.MX 8M Plus, i.MX 8ULP, i.MX 8QuadMax, and i.MX 8QuadXPlus, use the above steps.

You can also run more complex samples that demonstrate how Linux and Zephyr can work in unison.

The next section exemplifies the `openamp_rsc_table` application.

5.2 openamp_rsc_table application

The Zephyr's openamp_rsc_table application demonstrates how to use Open Asymmetric Multi-Processing (OpenAMP) with Zephyr based on a resource table. It is designed to respond to the:

- Linux rpmsg client sample
- Linux rpmsg tty driver

This sample implementation is compatible with platforms that embed a Linux kernel OS on the main processor and a Zephyr application on the coprocessor.

5.2.1 Load openamp_rsc_table application on DSP

As mentioned in Section 5.1.1 "Load hello_world application on DSP", to load the `openamp_rsc_table` application on the DSP, use the `imx_dsp_rproc` driver, after enabling `CONFIG_IMX_DSP_REMOTEPROC` in the Linux kernel.

5.2.2 Compile openamp_rsc_table application in Zephyr

Compile the `openamp_rsc_table` application in Zephyr for the i.MX 8M Plus DSP, that is, compile the application for the `imx8mp_evk/mimx8ml8/adsp` board.

Go to the `zephyr/` folder from `zephyrproject` and run the following:

```
~/zephyrproject/zephyr$ west build -p always -b imx8mp_evk//adsp samples/subsys/
ipc/openamp_rsc_table
~/zephyrproject/zephyr$
~/zephyrproject/zephyr$ ls -la build/zephyr total 5284
drwxr-xr-x 14 nxa06898 nxp   4096 Sep 27 17:42 .
drwxr-xr-x  7 nxa06898 nxp   4096 Sep 27 17:42 ..
drwxr-xr-x  5 nxa06898 nxp   4096 Sep 27 17:42 arch
drwxr-xr-x  3 nxa06898 nxp   4096 Sep 27 17:42 boards
drwxr-xr-x  5 nxa06898 nxp   4096 Sep 27 17:42 cmake
-rw-r--r--  1 nxa06898 nxp     96 Sep 27 17:42 .cmake.dotconfig.checksum
drwxr-xr-x  6 nxa06898 nxp   4096 Sep 27 17:42 CMakeFiles
-rw-r--r--  1 nxa06898 nxp  13684 Sep 27 17:42 cmake_install.cmake
-rw-r--r--  1 nxa06898 nxp  41787 Sep 27 17:42 .config
drwxr-xr-x 16 nxa06898 nxp   4096 Sep 27 17:42 drivers
...
drwxr-xr-x  4 nxa06898 nxp   4096 Sep 27 17:42 soc
drwxr-xr-x 23 nxa06898 nxp   4096 Sep 27 17:42 subsys
-rw-r--r--  1 nxa06898 nxp   2421 Sep 27 17:42 zephyr.dts
-rw-r--r--  1 nxa06898 nxp    730 Sep 27 17:42 zephyr.dts.d
-rw-r--r--  1 nxa06898 nxp 124812 Sep 27 17:42 zephyr.dts.pre
-rw-r--r--  1 nxa06898 nxp 550701 Sep 27 17:42 zephyr_final.map
-rwxr-xr-x  1 nxa06898 nxp 998304 Sep 27 17:42 zephyr_openamp_rsc_table.elf
-rw-r--r--  1 nxa06898 nxp 550701 Sep 27 17:42 zephyr_openamp_rsc_table.map
-rw-r--r--  1 nxa06898 nxp   7463 Sep 27 17:42 zephyr_openamp_rsc_table.stat
-rwxr-xr-x  1 nxa06898 nxp 998476 Sep 27 17:42 zephyr_pre0.elf
-rw-r--r--  1 nxa06898 nxp 551293 Sep 27 17:42 zephyr_pre0.map
...
```

The `zephyr_openamp_rsc_table.elf` file is used as the firmware to be loaded on the DSP.

For i.MX 8ULP (`imx8ulp_evk/mimx8ud7/adsp` board), the above command can be used.

For i.MX 8QuadMax and i.MX 8QuadXPlus, to compile the sample, use the following command:

```
west build -p -b imx8qm_mek/mimx8qm6/adsp -S nxp-openamp-imx8-adsp -s samples/
subsys/ipc/openamp_rsc_table
```

For these two targets, use a snippet for the device tree and configuration overlay.

5.2.3 Run openamp_rsc_table application on DSP in Linux

To run the `openamp_rsc_table` application on HiFi 4 DSP from i.MX 8M Plus in Linux, perform the following steps:

Note: *The `rpmsg_client_sample` and `rpmsg_tty` modules are used to communicate with the `openamp_rsc_table` application that runs on the DSP. These modules are the sample modules that run on the main processor (Cortex A core).*

Note: *To build the `rpmsg_client_sample` and `rpmsg_tty` modules, enable the `CONFIG_SAMPLE_RPMSG_CLIENT` and `CONFIG_RPMSG_TTY` configurations, respectively, in the Linux kernel.*

1. Start the i.MX 8M Plus EVK board with a specific DTS.
   For i.MX 8M Plus use `imx8mp-evk-dsp.dtb`, for i.MX 8ULP use `imx8ulp-evk.dtb`, for i.MX 8QuadMax use `imx8qm-mek-rpmsg.dtb`, and for i.MX 8QuadXPlus use `imx8qxp-mek-rpmsg.dtb`.

After inserting `imx_dsp_rproc` kernel module, check in the remoteproc id `/sys/class/remoteproc/` for the DSP:

```
root@imx8mpevk:~# ls -la /sys/class/remoteproc/
total 0
drwxr-xr-x  2 root root 0 Mar  3 09:49 .
drwxr-xr-x 90 root root 0 Mar  3 09:49 ..
lrwxrwxrwx  1 root root 0 Mar  3 09:54 remoteproc0 -> ../../devices/
platform/3b6e8000.dsp/remoteproc/remoteproc0
root@imx8mpevk:~# cat /sys/class/remoteproc/remoteproc0/firmware
imx/dsp/hifi4.bin
root@imx8mpevk:~#
```

Here, `remoteproc0`, which is for DSP, is used.

2. Check the firmware image on the board:

```
root@imx8mpevk:~# ls -la /lib/firmware/imx/zephyr/
total 1000
drwxr-xr-x  2 root root   4096 Mar  9  2018 .
drwxr-xr-x 10 root root   4096 Mar  9  2018 ..
-rwxr-xr-x  1 root root  65424 Mar  9  2018 imx8mp_evk-hello_world-zephyr.elf
-rwxr-xr-x  1 root root 221072 Mar  9  2018 imx8mp_evk-
number_crunching_cmsisDSP-zephyr.elf
-rwxr-xr-x  1 root root 147740 Mar  9  2018 imx8mp_evk-openamp_rsc_table-
zephyr.elf
-rwxr-xr-x  1 root root  78792 Mar  9  2018 imx8mp_evk-philosophers-
zephyr.elf
-rwxr-xr-x  1 root root  67536 Mar  9  2018 imx8mp_evk-synchronization-
zephyr.elf
-rwxr-xr-x  1 root root  76688 Mar  9  2018 imx8qm_mek-hello_world-zephyr.elf
-rwxr-xr-x  1 root root  97336 Mar  9  2018 imx8qm_mek-openamp_rsc_table-
zephyr.elf
-rwxr-xr-x  1 root root  76688 Mar  9  2018 imx8qxp_mek-hello_world-
zephyr.elf
-rwxr-xr-x  1 root root  97336 Mar  9  2018 imx8qxp_mek-openamp_rsc_table-
zephyr.elf
-rwxr-xr-x  1 root root  72536 Mar  9  2018 imx8ulp_evk-hello_world-
zephyr.elf
root@imx8mpevk:~#
```

The firmware must be present in `/lib/firmware` before the remoteproc driver is probed; however, it can also be given with an absolute path.

3. Insert the `imx_dsp_rproc` kernel module as shown below:

```
root@imx8mpevk:~# modprobe imx_dsp_rproc
root@imx8mpevk:~# ls -la /sys/class/remoteproc
total 0
drwxr-xr-x  2 root root 0 Feb 27 17:54 .
drwxr-xr-x 92 root root 0 Feb 27 17:27 ..
lrwxrwxrwx  1 root root 0 Feb 27 17:54 remoteproc0 -> ../../devices/
platform/3b6e8000.dsp/remoteproc/remoteproc0
root@imx8mpevk:~#
```

4. Insert `rpmsg` Linux client samples as follows:

```
root@imx8mpevk:~# modprobe rpmsg_client_sample /* rpmsg client sample driver
 used to communicate with remote processor over the rpmsg bus */
root@imx8mpevk:~# modprobe rpmsg_tty /* export rpmsg endpoints as tty
 devices, usually found as /dev/ttyRPMSGx */
```

5. Load firmware on DSP and run it:

```
root@imx8mpevk:~# echo -n imx/zephyr/imx8mp_evk-openamp_rsc_table-zephyr.elf
 > /sys/class/remoteproc/remoteproc0/firmware
root@imx8mpevk:~# echo start > /sys/class/remoteproc/remoteproc0/state
root@imx8mpevk:~# dmesg
...
[  979.299090] remoteproc remoteproc0: powering up imx-dsp-rproc
[  979.306290] remoteproc remoteproc0: Direct firmware load for /lib/
firmware/imx/zephyr/imx8mp_evk-hello_world-zephyr.elf failed with error -2
[  979.306308] remoteproc remoteproc0: Falling back to sysfs fallback for: /
lib/firmware/imx/zephyr/imx8mp_evk-hello_world-zephyr.elf
[  979.329548] remoteproc remoteproc0: Booting fw image /lib/firmware/imx/
zephyr/imx8mp_evk-hello_world-zephyr.elf, size 58068
[  979.332297] remoteproc remoteproc0: no resource table found for this
 firmware
[  979.332841] remoteproc remoteproc0: remote processor imx-dsp-rproc is now
 up
[ 1033.463449] remoteproc remoteproc0: stopped remote processor imx-dsp-rproc
[ 1675.323618] remoteproc remoteproc0: releasing imx-dsp-rproc
[ 1698.645763] remoteproc remoteproc0: imx-dsp-rproc is available
[ 2198.083754] remoteproc remoteproc0: powering up imx-dsp-rproc
[ 2198.095763] remoteproc remoteproc0: Booting fw image imx/zephyr/
imx8mp_evk-openamp_rsc_table-zephyr.elf, size 139300
[ 2198.101399] rproc-virtio rproc-virtio.2.auto: assigned reserved memory
 node vdev0buffer@94300000
[ 2198.104977] virtio_rpmsg_bus virtio0: rpmsg host is online
[ 2198.105061] rproc-virtio rproc-virtio.2.auto: registered virtio0 (type 7)
[ 2198.105068] remoteproc remoteproc0: remote processor imx-dsp-rproc is now
 up
[ 2198.105117] virtio_rpmsg_bus virtio0: creating channel rpmsg-tty addr
 0x400
[ 2198.105489] virtio_rpmsg_bus virtio0: creating channel rpmsg-client-sample
 addr 0x401
[ 2198.105731] rpmsg_client_sample virtio0.rpmsg-client-sample.-1.1025: new
 channel: 0x401 -> 0x401!
[ 2198.105815] virtio_rpmsg_bus virtio0: creating channel rpmsg-tty addr
 0x402
[ 2198.106153] rpmsg_client_sample virtio0.rpmsg-client-sample.-1.1025:
 incoming msg 1 (src: 0x401)
[ 2198.106207] rpmsg_client_sample virtio0.rpmsg-client-sample.-1.1025:
 incoming msg 2 (src: 0x401)
[ 2198.106246] rpmsg_client_sample virtio0.rpmsg-client-sample.-1.1025:
 incoming msg 3 (src: 0x401)
...
[ 2198.111203] rpmsg_client_sample virtio0.rpmsg-client-sample.-1.1025:
 incoming msg 97 (src: 0x401)
[ 2198.111242] rpmsg_client_sample virtio0.rpmsg-client-sample.-1.1025:
 incoming msg 98 (src: 0x401)
[ 2198.111282] rpmsg_client_sample virtio0.rpmsg-client-sample.-1.1025:
 incoming msg 99 (src: 0x401)
[ 2198.111323] rpmsg_client_sample virtio0.rpmsg-client-sample.-1.1025:
 incoming msg 100 (src: 0x401)
[ 2198.111329] rpmsg_client_sample virtio0.rpmsg-client-sample.-1.1025:
 goodbye!
[ 2198.111341] virtio_rpmsg_bus virtio0: destroying channel rpmsg-client-
sample addr 0x401
[ 2198.111413] rpmsg_client_sample virtio0.rpmsg-client-sample.-1.1025: rpmsg
 sample client driver is removed
root@imx8mpevk:~#
```

By default, the Zephyr console only prints:

```
*** Booting Zephyr OS build v4.0.0-3025-g52fd016274f3 ***
```

If `CONFIG_SHELL` is disabled from the project configuration, we get:

```
*** Booting Zephyr OS build v4.0.0-3025-g52fd016274f3 ***
[00:00:00.007,000] <inf> openamp_rsc_table: Starting application threads!
[00:00:00.012,000] <dbg> openamp_rsc_table: mailbox_notify: mailbox_notify:
 msg received
--- 287 messages dropped ---
[00:00:00.012,000] <dbg> openamp_rsc_table: platform_ipm_callback:
 platform_ipm_callback: msg received from mb 0
[00:00:00.012,000] <inf> openamp_rsc_table: [Linux sample client] incoming
 msg 95: hello world!
[00:00:00.012,000] <dbg> openamp_rsc_table: mailbox_notify: mailbox_notify:
 msg received
[00:00:00.012,000] <dbg> openamp_rsc_table: platform_ipm_callback:
 platform_ipm_callback: msg received from mb 0
[00:00:00.012,000] <inf> openamp_rsc_table: [Linux sample client] incoming
 msg 96: hello world!
[00:00:00.012,000] <dbg> openamp_rsc_table: mailbox_notify: mailbox_notify:
 msg received
[00:00:00.012,000] <dbg> openamp_rsc_table: platform_ipm_callback:
 platform_ipm_callback: msg received from mb 0
[00:00:00.012,000] <inf> openamp_rsc_table: [Linux sample client] incoming
 msg 97: hello world!
[00:00:00.012,000] <dbg> openamp_rsc_table: mailbox_notify: mailbox_notify:
 msg received
[00:00:00.012,000] <dbg> openamp_rsc_table: platform_ipm_callback:
 platform_ipm_callback: msg received from mb 0
[00:00:00.012,000] <inf> openamp_rsc_table: [Linux sample client] incoming
 msg 98: hello world!
[00:00:00.012,000] <dbg> openamp_rsc_table: mailbox_notify: mailbox_notify:
 msg received
[00:00:00.012,000] <dbg> openamp_rsc_table: platform_ipm_callback:
 platform_ipm_callback: msg received from mb 0
[00:00:00.012,000] <inf> openamp_rsc_table: [Linux sample client] incoming
 msg 99: hello world!
[00:00:00.012,000] <dbg> openamp_rsc_table: mailbox_notify: mailbox_notify:
 msg received
[00:00:00.012,000] <dbg> openamp_rsc_table: platform_ipm_callback:
 platform_ipm_callback: msg received from mb 0
[00:00:00.012,000] <inf> openamp_rsc_table: [Linux sample client] incoming
 msg 100: hello world!
[00:00:00.012,000] <dbg> openamp_rsc_table: mailbox_notify: mailbox_notify:
 msg received
[00:00:00.012,000] <dbg> openamp_rsc_table: mailbox_notify: mailbox_notify:
 msg received
[00:00:00.012,000] <inf> openamp_rsc_table: OpenAMP Linux sample client
 responder ended
```

6. On the Linux console, send a message to Zephyr, which replies with the "TTY <add>" prefix. `<addr>` corresponds to the Zephyr rpmsg-tty endpoint address.
   Linux console:

```
root@imx8mpevk:~# cat /dev/ttyRPMSG1 &
[1] 1657
root@imx8mpevk:~# echo "Hello Zephyr" >/dev/ttyRPMSG1
TTY 0x0402: Hello Zephyr
```

If `CONFIG_SHELL` option is enabled use the `ttyRPMSG1` for rpmsg TTY demo, as `ttyRPMSG0` is used for shell.
Otherwise, use `ttyRPMSG0`.

7. To stop the firmware, use the following command:

```
root@imx8mpevk:~# echo stop > /sys/class/remoteproc/remoteproc0/state
root@imx8mpevk:~#
root@imx8mpevk:~# dmesg | tail
...
[ 2758.000796] remoteproc remoteproc0: stopped remote processor imx-dsp-rproc
root@imx8mpevk:~#
```

5.2.4 openamp_rsc_table application output

To get the `openamp_rsc_table` application output, perform the following steps:

1. Get the console through the UART.
2. Open a serial terminal on the fourth serial port, for i.MX 8M Plus:

```
user@developerpc:~# minicom -D /dev/ttyUSB3
```

Note: *Check the serial port number for the other targets.*

The terminal displays the following messages (also shown in Figure 3):

```
*** Booting Zephyr OS build v4.0.0-3025-g52fd016274f3 ***
```

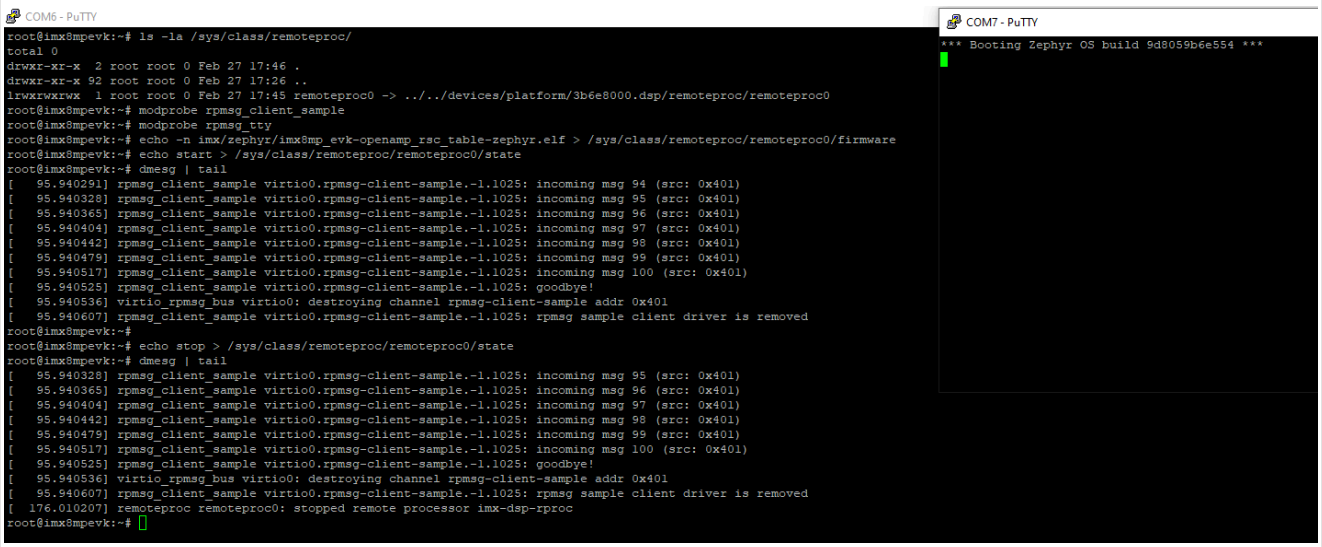

Figure 3. openamp_rsc_table application output

5.3 number_crunching application

This section demonstrates how to include a proprietary static library into the Zephyr build system. The library is in an out-of-tree location.

Number crunching sample does vector operations, Fast Fourier transformation (FFT), and filtering.

To demonstrate how to integrate a proprietary code, the NatureDSP library from Cadence has been used, which is in an out-of-tree location as it has a proprietary license.

By default, the sample is using the CMSIS-DSP backend which is in Zephyr as a module.

To use this sample, with an out-of-tree library, define an environment variable `LIB_LOCATION`. In the location, ensure to have a `CMakeLists.txt` that defines the following:

```
# Link with the external 3rd party library.
set(LIB_DIR         "lib"                   CACHE STRING "")
set(INCLUDE_DIR     "include"               CACHE STRING "")
set(LIB_NAME        "proprietary_lib.a"     CACHE STRING "")
```

If the environment variable `LIB_LOCATION` is not defined, a default Zephyr API is used instead.

This sample executes some mathematical functions that can be used for audio processing like filtering (FFT) or echo cancellation (Least Mean Square (LMS) filter algorithm).

The sample has the following:

- `main.c`: Calls the generic math functions;
- `math_ops.c`: Executes the math functions, computes the cycles it took to execute, and checks the output.
- `cmsis_dsp_wrapper.c`: Calls the exact math functions from CMSIS-DSP when CONFIG_CMSIS_DSP is defined and LIB_LOCATION is not defined.
- `nature_dsp_wrapper`: If LIB_LOCATION is defined and points to an out-of-tree location where the NatureDSP library and headers can be found, calls the exact math functions from the NatureDSP library.

To include a new backend, create a new wrapper for that library or backend.

5.3.1 Load number_crunching application on DSP

To load the `number_crunching` application on the DSP, use the Linux DSP-specific remoteproc driver.

To use the driver, enable `CONFIG_IMX_DSP_REMOTEPROC` in the Linux kernel.

5.3.2 Compile number_crunching application

CMSIS-DSP is an optional module and must be added explicitly to the Zephyr workspace:

```
west config manifest.project-filter -- +cmsis-dsp
west update cmsis-dsp
```

NatureDSP is available on GitHub: https://github.com/foss-xtensa/ndsplib-hifi4/tree/main. To compile it, use the steps described in the repository.

5.3.2.1 Compilation

Go to the `zephyr/` folder from `zephyrproject` and run the following:

```
~/zephyrproject/zephyr$ west build -p -b imx8mp_evk/mimx8ml8/adsp -s samples/
boards/nxp/adsp/number_crunching/
```

This links the application with the default backend, CMSIS_DSP.

If you have a proprietary library, define an environment variable `LIB_LOCATION` and export it, before compiling the application:

```
~/zephyrproject/zephyr$ export LIB_LOCATION=/home/external_lib_location
~/zephyrproject/zephyr$
```

```
~/zephyrproject/zephyr$ west build -p always -b imx8mp_evk/mimx8ml8/adsp -s
 samples/boards/nxp/adsp/number_crunching/
~/zephyrproject/zephyr$
```

Or use `LIB_LOCATION` as simple variable and compile with the following:

```
~/zephyrproject/zephyr$ west build -p always -b imx8mp_evk/mimx8ml8/
adsp -s samples/boards/nxp/adsp/number_crunching/ -DLIB_LOCATION=/home/
external_lib_location
```

After compilation you have the following result:

```
~/zephyrproject/zephyr$
~/zephyrproject/zephyr$ ls -la build/zephyr
total 8216
drwxr-xr-x 14 user nxp    4096 Jan 10 13:56 .
drwxr-xr-x  7 user nxp    4096 Jan 10 13:56 ..
drwxr-xr-x  5 user nxp    4096 Jan 10 13:56 arch
drwxr-xr-x  5 user nxp    4096 Jan 10 13:56 boards
drwxr-xr-x  5 user nxp    4096 Jan 10 13:56 cmake
-rw-r--r--  1 user nxp      64 Jan 10 13:56 .cmake.dotconfig.checksum
...
-rw-r--r--  1 user nxp  125981 Jan 10 13:56 zephyr.dts.pre
-rwxr-xr-x  1 user nxp 1087796 Jan 10 13:56 zephyr.elf
-rw-r--r--  1 user nxp 1446150 Jan 10 13:56 zephyr_final.map
-rw-r--r--  1 user nxp 1446150 Jan 10 13:56 zephyr.map
-rwxr-xr-x  1 user nxp 1088088 Jan 10 13:56 zephyr_pre0.elf
-rw-r--r--  1 user nxp 1446801 Jan 10 13:56 zephyr_pre0.map
-rw-r--r--  1 user nxp    7584 Jan 10 13:56 zephyr.stat
```

The `zephyr.elf` file is used as the firmware to be loaded on the DSP.

5.3.3 Run number_crunching application on DSP

To run the `number_crunching` application on HiFi 4 DSP from i.MX 8M Plus, perform the following steps:

1. Start the board with a specific DTS, see Section 5.1.3 "Run hello_world application on DSP".
2. Check the firmware image on the board:

```
root@imx8mpevk:~# ls -la /lib/firmware/imx/zephyr/
total 1000
drwxr-xr-x  2 root root   4096 Mar  9  2018 .
drwxr-xr-x 10 root root   4096 Mar  9  2018 ..
-rwxr-xr-x  1 root root  65424 Mar  9  2018 imx8mp_evk-hello_world-zephyr.elf
-rwxr-xr-x  1 root root 221072 Mar  9  2018 imx8mp_evk-
number_crunching_cmsisDSP-zephyr.elf
-rwxr-xr-x  1 root root 147740 Mar  9  2018 imx8mp_evk-openamp_rsc_table-
zephyr.elf
-rwxr-xr-x  1 root root  78792 Mar  9  2018 imx8mp_evk-philosophers-
zephyr.elf
-rwxr-xr-x  1 root root  67536 Mar  9  2018 imx8mp_evk-synchronization-
zephyr.elf
-rwxr-xr-x  1 root root  76688 Mar  9  2018 imx8qm_mek-hello_world-zephyr.elf
-rwxr-xr-x  1 root root  97336 Mar  9  2018 imx8qm_mek-openamp_rsc_table-
zephyr.elf
-rwxr-xr-x  1 root root  76688 Mar  9  2018 imx8qxp_mek-hello_world-
zephyr.elf
-rwxr-xr-x  1 root root  97336 Mar  9  2018 imx8qxp_mek-openamp_rsc_table-
zephyr.elf
```

```
-rwxr-xr-x  1 root root  72536 Mar  9  2018 imx8ulp_evk-hello_world-
zephyr.elf
root@imx8mpevk:~#
```

The firmware must be present in `/lib/firmware` before the remoteproc driver is probed; however, it can also be given with an absolute path.

3. Insert the `imx_dsp_rproc` kernel module with default parameters as shown below:

```
root@imx8mpevk:~# modprobe imx_dsp_rproc
```

4. To load the firmware on DSP and run it, execute the following commands:

```
root@imx8mpevk:~# echo -n /lib/firmware/imx/zephyr/imx8mp_evk-
number_crunching_cmsisDSP-zephyr.elf > /sys/class/remoteproc/remoteproc0/
firmware
root@imx8mpevk:~# echo start > /sys/class/remoteproc/remoteproc0/state
```

5. To stop the firmware, use the following command:

```
root@imx8mpevk:~# echo stop > /sys/class/remoteproc/remoteproc0/state
```

5.3.4 number_crunching application output

To get the number_crunching application output, perform the following steps:

1. Get the console through the UART.
2. Open a serial terminal on the fourth serial port:

```
user@developerpc:~# minicom -D /dev/ttyUSB3
```

The terminal displays the following message:

```
*** Booting Zephyr OS build v4.0.0-3164-g5faf471ce00d ***

Number crunching example!

[Library Test] == Vector Sum test  ==
[Backend] CMSIS-DSP module
[Library Test] Vector Sum takes 6614 cycles
[Library Test] == Vector Sum test end with 1 ==

[Library Test] == Vector power sum test  ==
[Backend] CMSIS-DSP module
[Library Test] Vector power sum takes 6669 cycles
[Library Test] == Vector power sum test end with 1 ==

[Library Test] == Vector power sum test  ==
[Backend] CMSIS-DSP module
[Library Test] Vector power sum takes 3762 cycles
[Library Test] == Vector power sum test end ==

[Library Test] == Fast Fourier Transform on Real Data test  ==
[Backend] CMSIS-DSP module
[Library Test] Fast Fourier Transform on Real Data takes 66870 cycles
[Library Test] == Fast Fourier Transform on Real Data test end ==

[Library Test] == Bi-quad Real Block IIR test  ==
[Backend] CMSIS-DSP module
[Library Test] Bi-quad Real Block IIR takes 506486 cycles
[Library Test] == Bi-quad Real Block IIR end ==
```

```
[Library Test] == Least Mean Square (LMS) Filter for Real Data test  ==
[Backend] CMSIS-DSP module
[Library Test] Least Mean Square (LMS) Filter for Real Data test takes 184792
 cycles
[Library Test] == Least Mean Square (LMS) Filter for Real Data test end ==
```

For NatureDSP, the output looks as follows:

```
*** Booting Zephyr OS build v4.0.0-3164-g5faf471ce00d ***

Proprietary library example!

[Library Test] == Vector Sum test  ==
[Backend] NatureDSP library
[Library Test] Vector Sum takes 3829 cycles
[Library Test] == Vector Sum test end with 1 ==

[Library Test] == Vector power sum test  ==
[Backend] NatureDSP library
[Library Test] Vector power sum takes 2432 cycles
[Library Test] == Vector power sum test end with 1 ==

[Library Test] == Vector power sum test  ==
[Backend] NatureDSP library
[Library Test] Vector power sum takes 2594 cycles
[Library Test] == Vector power sum test end ==

[Library Test] == Fast Fourier Transform on Real Data test  ==
[Backend] NatureDSP library
[Library Test] Fast Fourier Transform on Real Data takes 3338 cycles
[Library Test] == Fast Fourier Transform on Real Data test end ==

[Library Test] == Bi-quad Real Block IIR test  ==
[Backend] NatureDSP library
[Library Test] Bi-quad Real Block IIR takes 13501 cycles
[Library Test] == Bi-quad Real Block IIR end ==

[Library Test] == Least Mean Square (LMS) Filter for Real Data test  ==
[Backend] NatureDSP library
[Backend] NatureDSP library
[Library Test] Least Mean Square (LMS) Filter for Real Data test takes 7724
 cycles
[Library Test] == Least Mean Square (LMS) Filter for Real Data test end ==
```

6 Acronyms

Table 3 lists the acronyms used in this document.

Table 3. Acronyms

Acronym	Description
DTS	Device tree source
IPC	Inter-process communication
MCU	Microcontroller unit
MPU	Microprocessor unit

Table 3. Acronyms...*continued*

Acronym	Description
OpenAMP	Open Asymmetric Multi-Processing
OS	Operating system
rproc	Remote processor
rsc_table	Resource table
RTOS	Real-time operating system
UART	Universal Asynchronous Receiver/Transmitter

7 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2025 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

8 Revision history

Table 4 summarizes the revisions to this document.

Table 4. Revision history

Document ID	Release date	Description
AN13970 v.4.0	28 January 2025	• Updated the document for i.MX 8, i.MX 8M Plus, i.MX 8ULP, and i.MX 8X product family • Updated Section 5.1 "hello_world application" • Updated Section 5.2 "openamp_rsc_table application" • Added Section 5.3 "number_crunching application" • Updated Section 6 "Acronyms"
AN13970 v.3.0	05 June 2024	• Changed `nxp_adsp_imx8m board` to `imx8mp_evk/mimx8ml8/adsp` • Updated these images: – Figure 2

Table 4. Revision history...*continued*

Document ID	Release date	Description
		– Figure 3 • Updated the code in these sections: – Check firmware on board – Load firmware on DSP and run it – Stop firmware – Section 5.1.4 – Start i.MX 8M Plus EVK board – Check firmware on board – Insert imx_dsp_rproc kernel module – Insert rpmsg Linux client samples – rp – rpmsg TTY demo – Stop firmware – Section 5.2.4
AN13970 v.2.0	28 November 2023	• Updated these sections: – Section 5.1.2 – Start i.MX 8M Plus EVK board – Check firmware on board – Load firmware on DSP and run it – Stop firmware – Section 5.1.4 • Added Section 5.2
AN13970 v.1.0	1 June 2023	Initial public release

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents