# Coprocessor Application Manual

PHYTEC Messtechnik GmbH

Feb 27, 2025

# CONTENTS

**Warning**

This manual is a draft version and is currently **work in progress**. It will undergo significant changes over time.

We value your feedback, questions, and suggestions and encourage you to open an issue or pull request in the linked repository to get in contact.

| Coprocessor Application Manual | |
|---|---|
| Document Title | Coprocessor Application Manual |
| Document Type | Generic Application Guide |
| Release Date | XXXX/XX/XX |

This manual applies to all Phytec releases from kernel version x.

Most modern SoCs include one of more coprocessors beside an application processor. In most cases the application processor runs Linux, while the coprocessor may run an RTOS. This manuals goes into detail how to utilize the coprocessor efficiently for projects.

The manual explains generic principles and applies those principles in examples for a specific platform and tools. It gives in introduction to OpenAMP, Zephyr and Protocol Buffers (protobuf).

CHAPTER

# ONE

# USE CASES

CHAPTER

TWO

# OVERVIEW OF TECHNOLOGIES

## 2.1 OpenAMP

The OpenAMP Project which "seeks to standardize the interactions between operating environments in a heterogeneous embedded system through open source solutions for Asymmetric MultiProcessing (AMP)." This introduction explains the main components and terms, the OpenAMP documentation goes into further detail. OpenAMP is available in Linux as well as in RTOS (e.g. Zephyr) and Vendor SDKs (e.g. NXP MCUX, TI SDK, STM32Cube). The OpenAMP framework consists of several components:

### 2.1.1 Components

**Shared Memory**
: In SoCs, where coprocessors are integrated on the same die they typically communicate by exchanging data via sharing memory sections.

**Interrupts**
: Minimum set of one interrupt line per communicating core. This interrupt is often implemented in hardware blocks of the SoC, e.g. the "Messaging Unit (MU)" on the NXP i.MX8MP.

**remoteproc**
: Life Cycle Management (LCM) to manage and update the software running on coprocessors. The remoteproc documentation on Kernel.org goes into further technical details.

**RPMsg**
: Exchange of messages, api that enables Inter Processor Communications (IPC). The rpmsg documentation on Kernel.org goes into further technical details.

**VirtIO, Virtqueue, Vring**
: **VirtIO** provides a standardized framework for efficient message exchange between processing units using virtualization. **Virtqueue** is the abstract structure that manages the queues of messages and buffers for communication between these units. **Vring** is the specific circular buffer implementation used within a Virtqueue that organizes the descriptors for the messages being exchanged. There are ringbuffers for both directions (read, write). This set of infrastructure components can be used in combination with RPMsg to improve performance and reduce synchronization between Cores.

**RPC based on RPMsg**
: TBD

### 2.1.2 RPMsg Messaging Protocol

communication stack consisting of several protocol layers:

**Transport Layer:**
: RPMsg

**MAC Layer:**
VirtIO, Virtqueue, Vring

**Physical Layer:**
Shared Memory, Inter-core Interrupts e.g. via Messaging Unit (MU)

## 2.2 Libmetal

## 2.3 Protocol Buffers

## 2.4 Zephyr

## 2.5 NXP MCUX

CHAPTER

THREE

# APPLICATION ARCHITECTURES

## 3.1 VirtIO

## 3.2 RPmsg + Protobuf

CHAPTER

FOUR

# EXAMPLES AND RESOURCES

The Zephyr Inter-Processor Communication (IPC) Subsystem includes some samples:

**Note**

**Remoteproc:** The remoteproc framework is used to load and manage firmware on coprocessors. It also ensures to register the resource table and the RPMsg service. If RPMsg is used, flashing the firmware via a SWD debug adapter is not possible.

Furthermore, remoteproc only reads firmware files from the `/lib/firmware` directory. Loading firmware binaries from another location will result in errors.

**Resources:**

- NXP AN5317 - Loading code to Coprocessor

## 4.1 Hello World

### 4.1.1 Run the Sample

1. Make sure the devicetree overlay `imx8mp-phycore-rpmsg.dtbo` is activated, the BSP manual for your platform explains how to activate this.
2. Restart the target and execute in U-Boot:

```
u-boot=> run prepare_mcore
```

3. Save the environment in U-Boot in order to enable the m-core on every boot by default. Executing `saveenv` twice will save the environment to the redundant MMC partition as well.

```
u-boot=> saveenv
Saving Environment to MMC... Writing to MMC(1)... OK
u-boot=> saveenv
Saving Environment to MMC... Writing to redundant MMC(1)... OK
```

3. The target will now boot and you can build and flash the Zephyr application with:

```
host:zephyrproject/zephyr$ west build -b phyboard_pollux/mimx8ml8/m7 samples/hello_world -p
```

4. Zephyr should now boot with

```
target_m7:~$ *** Booting Zephyr OS build v3.7.0 ***
             Hello World! phyboard_pollux/mimx8ml8/m7
```

## 4.2 OpenAMP using resource table

The openamp_rsc_table sample "demonstrates how to use OpenAMP with Zephyr based on a resource table. It is designed to respond to [..]" the rpmsg client and rpmsg tty samples in the Linux Kernel. This sample demonstrates communication between Zephyr (coprocessor) and Linux (application processor) using OpenAMP. It creates the two RPMsg endpoints:

**rpmsg-client-sample**
: Demonstrates generic RPMsg message exchange (Ping-pong) between Zephyr and Linux.

**rpmsg-tty**
: A TTY service that virtualizes a serial connection at */dev/rpmsg-tty* in Linux, facilitating data exchange with Zephyr over this virtualized interface.

### 4.2.1 Prepare Linux

The example has been tested with the imx8mp and the BSP-Yocto-NXP-i.MX8MP-PD24.1.0. However, some modifications are necessary to be able to communicate in between Zephyr and Linux with RPMsg. The devicetree overlay that enables rpmsg has to be enabled. You can edit this line directly in bootenv.txt in the boot partition.

Listing 1: Changes in 'bootenv.txt'

```
+++ b/recipes-bsp/bootenv/phytec-bootenv/phyboard-pollux-imx8mp-3/bootenv.txt
@@ -1 +1 @@
-overlays=conf-imx8mp-phyboard-pollux-peb-av-10.dtbo
+overlays=conf-imx8mp-phyboard-pollux-peb-av-10.dtbo#conf-imx8mp-phycore-rpmsg.dtbo
```

Listing 2: Changes in the devicetree overlay 'imx8mp-phycore-rpmsg.dtbo'

```
+++ b/arch/arm64/boot/dts/freescale/imx8mp-phycore-rpmsg.dtso
@@ -14,11 +14,11 @@
        core-m7 {
                compatible = "fsl,imx8mn-cm7";
                clocks = <&clk IMX8MP_CLK_M7_DIV>;
-               mboxes = <&mu 0 1>,
-                        <&mu 1 1>,
-                        <&mu 3 1>;
+               mboxes = <&mu 0 0>,
+                        <&mu 1 0>,
+                        <&mu 3 0>;
                mbox-names = "tx", "rx", "rxdb";
-               memory-region = <&vdevbuffer>, <&vdev0vring0>, <&vdev0vring1>, <&rsc_table>;
+               memory-region = <&vdevbuffer>, <&vdev0vring0>, <&vdev0vring1>;
        };

        reserved-memory {
@@ -27,29 +27,31 @@ reserved-memory {
                #size-cells = <2>;

                vdev0vring0: vdev0vring0@55000000 {
-                       no-map;
+                       compatible = "shared-dma-pool";
```

(continues on next page)

(continued from previous page)

```
                        reg = <0 0x55000000 0 0x8000>;
+                       no-map;
                };

                vdev0vring1: vdev0vring1@55008000 {
-                       no-map;
+                       compatible = "shared-dma-pool";
                        reg = <0 0x55008000 0 0x8000>;
+                       no-map;
                };
```

### 4.2.2 Prepare Zephyr

The sample needs some board specific settings and a devicetree overlay for the phyBOARD Pollux. This will be upstreamed soon and maybe it is possible to make the Zephyr sample fully generic.

You can see a branch with the required changes here.

### 4.2.3 Run the Sample

1. Make sure the devicetree overlay `imx8mp-phycore-rpmsg.dtbo` is activated, the BSP manual for your platform explains how to activate this.
2. Restart the target and execute in U-Boot:

   ```
   u-boot=> run prepare_mcore
   ```

3. Build Zephyr and copy the firmware to `/lib/firmware` on the target:

   ```
   host:zephyrproject/zephyr$ west build -b phyboard_pollux/mimx8ml8/m7 samples/subsys/ipc/
   ↪openamp_rsc_table/ -p
   ```

4. Start the Zephyr application with remoteproc:

   ```
   root@phyboard-pollux-imx8mp-3:~# echo stop > /sys/class/remoteproc/remoteproc0/state
   root@phyboard-pollux-imx8mp-3:~# echo /lib/firmware/zephyr_openamp_rsc_table.elf > /sys/
   ↪class/remoteproc/remoteproc0/firmware
   root@phyboard-pollux-imx8mp-3:~# echo start > /sys/class/remoteproc/remoteproc0/state
   ```

4. Zephyr should now boot now. The kernel module `rpmsg_client_sample` should load automatically and respond to the running m-core.

   ```
   target_m7:~$ *** Booting Zephyr OS build v4.0.0-870-g6d87bd65aebf ***
                I: Starting application threads!
                I: OpenAMP[remote] Linux responder demo started
                D: mailbox_notify: msg received
                I: OpenAMP[remote] Linux sample client responder started
                D: mailbox_notify: msg received
                I: OpenAMP[remote] Linux TTY responder started
                D: mailbox_notify: msg received

                : platform_ipm_callback: msg received from mb 0
                I: [Linux sample client] incoming msg 1: hello world!
   ```

(continues on next page)

(continued from previous page)

```
            D: mailbox_notify: msg received
            D: platform_ipm_callback: msg received from mb 0
            I: [Linux sample client] incoming msg 1: hello world!
            D: mailbox_notify: msg received
```

5. If the the kernel Module does not load automatically, you can manually load it:

```
target:~$ modprobe rpmsg_client_sample
target:~$ dmesg | tail                          # Check module messages
target:~$ modprobe -u rpmsg_client_sample       # Unload Kernel module
```

#### Serial Communication

Once the demo is running, it opens two serial devices (`/dev/ttyRPMSG0`, `/dev/ttyRPMSG1`), one to send/receive any messages to Zephyr and one for the Zephyr shell backend.

```
# Open the tty channel
root@phyboard-pollux-imx8mp-3:~# cat /dev/ttyRPMSG1 &
[3] 504
root@phyboard-pollux-imx8mp-3:~# echo "Hello Zephyr" >/dev/ttyRPMSG1
root@phyboard-pollux-imx8mp-3:~# TTY 0x0402: Hello Zephyr
TTY 0x0402: Hello Zephyr

# Open the Zephyr shell with micocom
root@phyboard-pollux-imx8mp-3:~# microcom /dev/ttyRPMSG0

clear    device   devmem   help     history  kernel   rem      resize
retval   shell
ipc:~$
```

**Note**

Remoteproc ensures to register the resource table and the RPMsg service. Running firmware via debug adapter is not possible when using RPMsg.

**Warning**

Remoteproc only reads firmware files from the `/lib/firmware` directory! If you try to load a binary from another location errors will occur!

### 4.2.4 Console Output Linux

```
# Stop a running m-core
root@phyboard-pollux-imx8mp-3:~# echo stop > /sys/class/remoteproc/remoteproc0/state
[18375.572034] imx-rproc core-m7: Not in wfi, force stopped
[18375.577423] remoteproc remoteproc0: stopped remote processor imx-rproc

# Load the firmware
root@phyboard-pollux-imx8mp-3:~# echo /lib/firmware/zephyr_openamp_rsc_table.elf > /sys/class/
↪remoteproc/remoteproc0/firmware
```

(continues on next page)

(continued from previous page)

```
# Start the m-core
root@phyboard-pollux-imx8mp-3:~# echo start > /sys/class/remoteproc/remoteproc0/state
[18402.215721] remoteproc remoteproc0: powering up imx-rproc
[18402.221215] remoteproc remoteproc0: Direct firmware load for /lib/firmware/zephyr.elf failed␣
↪with error -2
[18402.230900] remoteproc remoteproc0: Falling back to sysfs fallback for: /lib/firmware/zephyr.
↪elf
[18402.243066] remoteproc remoteproc0: Booting fw image /lib/firmware/zephyr.elf, size 1402364
[18402.252283] rproc-virtio rproc-virtio.3.auto: assigned reserved memory node␣
↪vdevbuffer@55400000
[18402.262788] virtio_rpmsg_bus virtio0: rpmsg host is online
[18402.268484] rproc-virtio rproc-virtio.3.auto: registered virtio0 (type 7)
[18402.275367] virtio_rpmsg_bus virtio0: creating channel rpmsg-tty addr 0x400
[18402.276433] remoteproc remoteproc0: remote processor imx-rproc is now up
[18402.282735] virtio_rpmsg_bus virtio0: creating channel rpmsg-client-sample addr 0x401
[18402.297625] rpmsg_client_sample virtio0.rpmsg-client-sample.-1.1025: new channel: 0x401 ->␣
↪0x401!
[18402.308941] virtio_rpmsg_bus virtio0: creating channel rpmsg-tty addr 0x402
[18402.320915] rpmsg_client_sample virtio0.rpmsg-client-sample.-1.1025: incoming msg 1 (src:␣
↪0x401)
[18402.341810] rpmsg_client_sample virtio0.rpmsg-client-sample.-1.1025: incoming msg 2 (src:␣
↪0x401)
```

### 4.2.5 Debugging

Listing 3: Print resource table in Linux

```
root@phyboard-pollux-imx8mp-3:~# cat /sys/kernel/debug/remoteproc/remoteproc0/resource_table
Entry 0 is of type vdev
  ID 7
  Notify ID 0
  Device features 0x1
  Guest features 0x1
  Config length 0x0
  Status 0x7
  Number of vrings 2
  Reserved (should be zero) [0][0]

  Vring 0
    Device Address 0x55000000
    Alignment 16
    Number of buffers 8
    Notify ID 0
    Physical Address 0x0

  Vring 1
    Device Address 0x55008000
    Alignment 16
    Number of buffers 8
    Notify ID 1
    Physical Address 0x0
```

Listing 4: Print related memory areas in Linux:

```
root@phyboard-pollux-imx8mp-3:~# cat /sys/kernel/debug/remoteproc/remoteproc0/resource_table
Entry 0 is of type vdev
  ID 7
  Notify ID 0
  Device features 0x1
  Guest features 0x1
  Config length 0x0
  Status 0x7
  Number of vrings 2
  Reserved (should be zero) [0][0]

  Vring 0
    Device Address 0x55000000
    Alignment 16
    Number of buffers 8
    Notify ID 0
    Physical Address 0x0

  Vring 1
    Device Address 0x55008000
    Alignment 16
    Number of buffers 8
    Notify ID 1
    Physical Address 0x0
```

## 4.3 Other Examples

The following examples exist in Zephyr, however, they are specific to SoCs that have multiple instances of Zephyr running in the same SoC. They are partly related to Zephyrs ipc_service and not suitable for communication with Linux.

OpenAMP Sample

> sample builds different images for two targets running Zephyr. Both targets setup virtqueue and virtio and communicate with each other via RPMsg. This sample is mainly used to evaluate SoCs with two Cortex M devices and can not be used with Linux.

openamp-system-reference

> Several samples for both platforms, Linux and Zephyr that demonstrate different aspects of OpenAMP.

Samples in ipc_service/

> Examples related to Zephyr ipc_service subsystem. Note that not all of those examples may be applicable to heterogeneous systems with one core running Linux and the other Zephyr.

CHAPTER

FIVE

# CURRENT PROBLEMS

This section lists current problems that need work.

1. Shell not working in Zephyr for Linux SoCs.

   There may be a problem with interrupts and nxp deactivated the shell for the imx8qm boards. https://github.com/zephyrproject-rtos/zephyr/pull/79428