Linux and Zephyr "talking" to each other in the same SoC

Diego Sueiro, Sepura / Embarcados
www.embarcados.com.br

Embedded Linux Conference Europe 2018

Agenda

Real-Time applications with HMP (Heterogeneous Multiprocessing)

OpenAMP Introduction

RPMsg Introduction

RPMsg-lite Introduction

Enabling RPMsg on Linux

Enabling RPMsg-lite on Zephyr

Linux and Zephyr communication setup

Demo

Future work

Preamble

- NXP i.MX7 processor is hardware reference platform;

- The "full" OpenAMP will not be used in this presentation;

- Up to now on Zephyr, only one SoC family (LPC54114) is using the "full" OpenAMP to communicate between Cortex M0 and M4 cores;

- This work is not in the mainline (kernel and zephyr) yet, but is open source ;-)

Agenda

Real-Time applications with HMP (Heterogeneous Multiprocessing)

OpenAMP Introduction

RPMsg Introduction

RPMsg-lite Introduction

Enabling RPMsg on Linux

Enabling RPMsg-lite on Zephyr

Linux and Zephyr communication setup

Demo

Future work

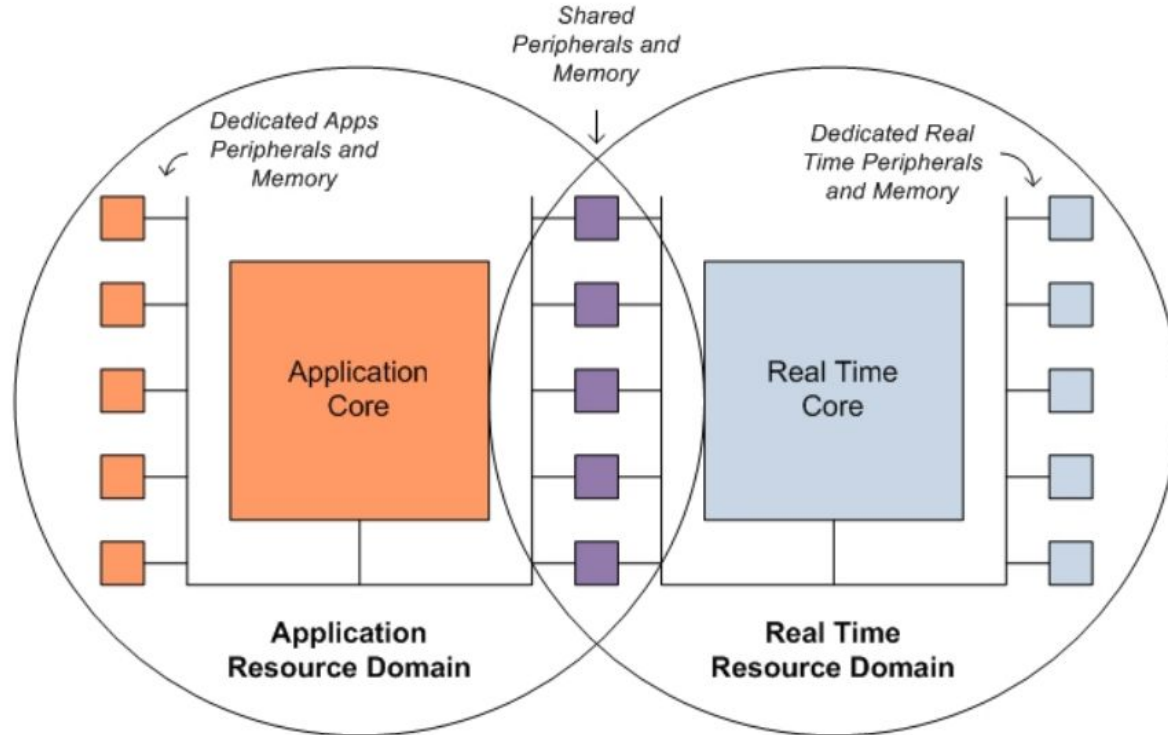# Real-Time applications with HMP

- a.k.a: AMP (Asymmetric Multiprocessing);

- In the same SoC we have different CPU architectures and combinations:
  - Application Core, e.g.: ARM Cortex A9;
  - Digital Signal Processing, e.g.: Texas Instruments DSP C6000;
  - Computing Power, e.g.: Xilinx FPGA Artix-7;
  - Low Power and Real-time performance: ARM Cortex M4;
  - ...

- Some applications may require:
  - Real-time performance;
  - Performance optimization;
  - Power consumption;
  - Fast boot;
  - System integrity;
  - System security;
  - Leverage hardened or certified software solutions;
  - Reuse of legacy software.

Real-Time applications with HMP

- Linux kernel (e.g. PREEMPT_RT) can meet some of these requirements but tuning, customising, extending, debugging, maintaining and updating is costly in terms of knowledge, time and money;

- A HMP is a possible solution that gives:
  - Software domains isolation and partition;
  - Sensors and Actuators HUB;
  - BOM cost reduction.

- But HMP has some challenges like:
  - Interprocessor Synchronization and Communication;
  - Efficient Power Management;
  - Shared resources Isolation and Protection;
  - Cache coherency management.

- SoC vendors have being launching a vast variety of HMP for different market verticals.
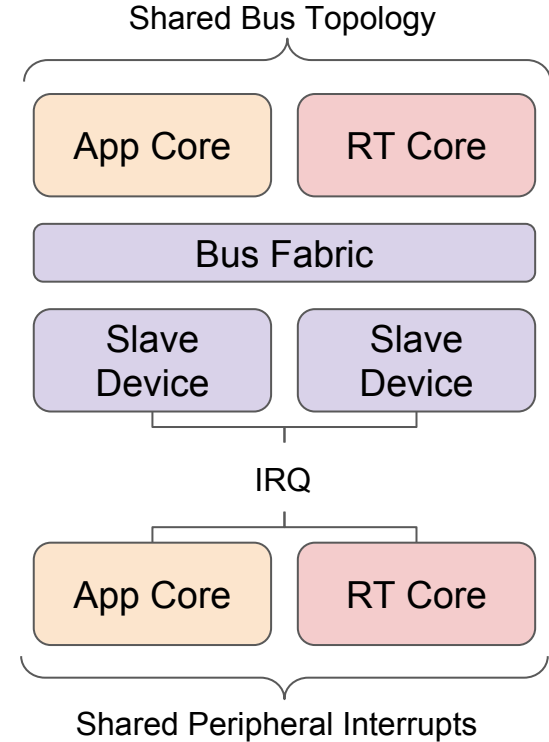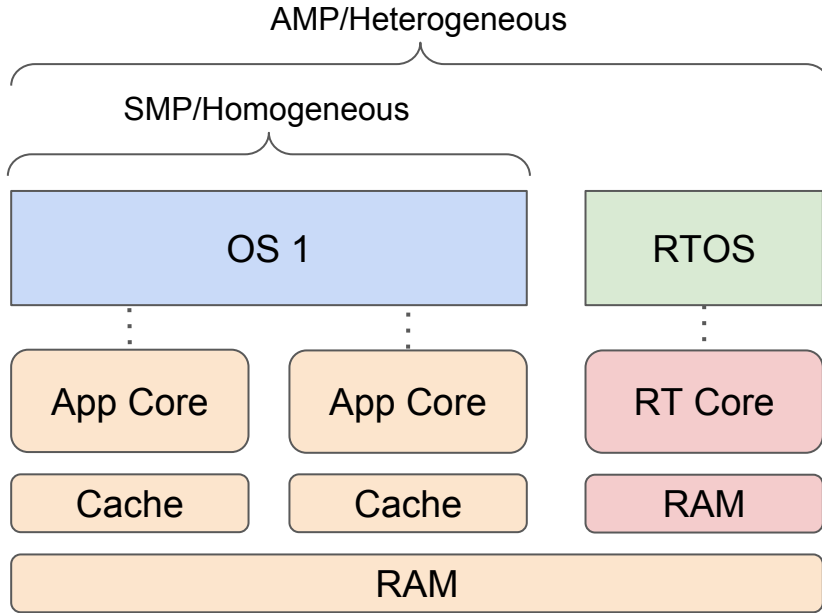
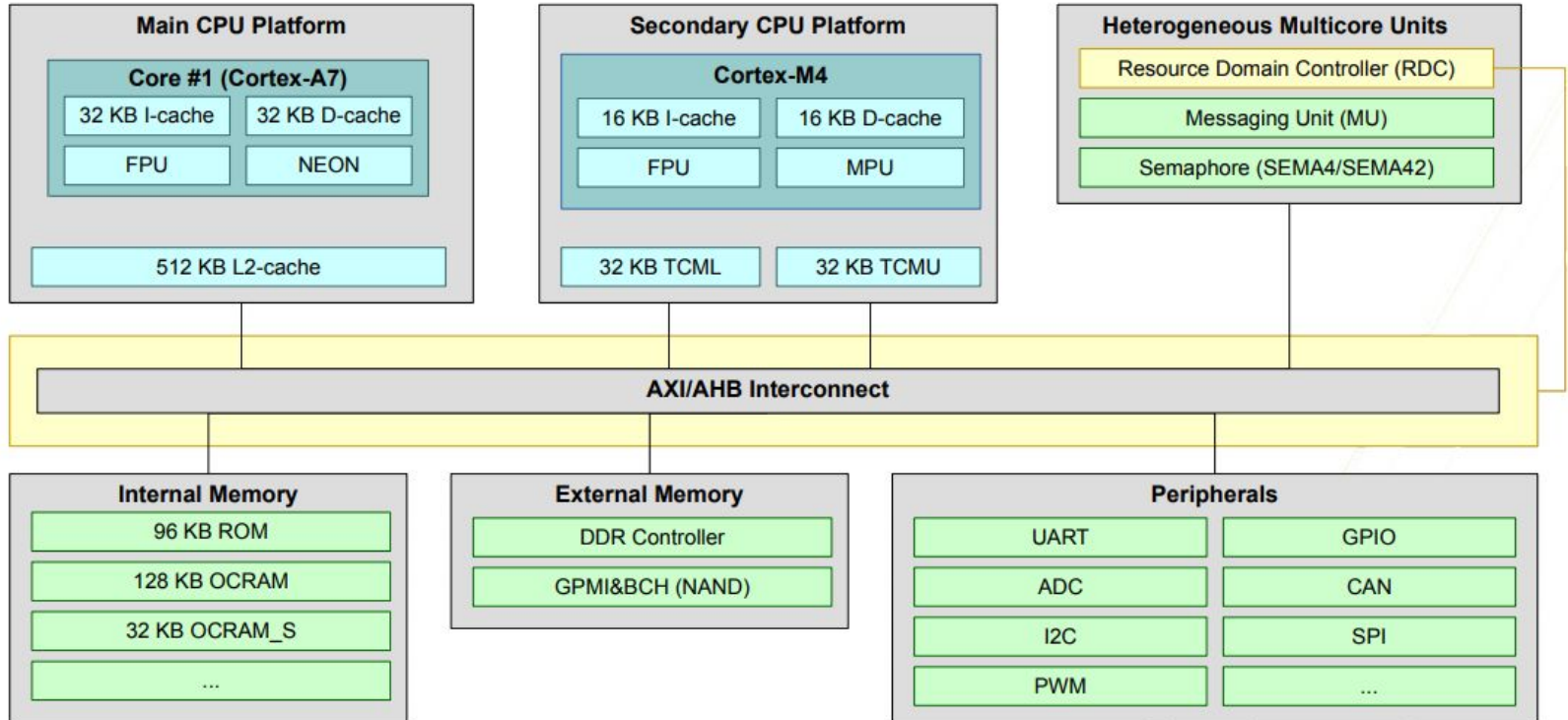# Real-Time applications with HMP

How a HMP looks like:

Real-Time applications with HMP

How a HMP looks like:

Real-Time applications with HMP

How a HMP looks like: e.g. NXP i.MX7S

Agenda

Real-Time applications with HMP (Heterogeneous Multiprocessing)

OpenAMP Introduction

RPMsg Introduction

RPMsg-lite Introduction

Enabling RPMsg on Linux

Enabling RPMsg-lite on Zephyr

Linux and Zephyr communication setup

Demo

Future work

OpenAMP Introduction

OpenAMP: standard from MCA (The Multicore Association) implemented in the Linux kernel and Zephyr mainline:

- Lifecycle operations via **Remoteproc** (Remote Processor): Framework that allows a master to control/manage remote processors (power on/off, reset, load firmware);

- Messaging via **RPMsg** (Remote Processor Messaging): Framework that provides inter-processor communication (IPC) using **VirtIO** (standard Linux virtualization component) for shared memory management when sending/receiving data from/to master/remote;

- Proxy operations: Remote access to systems services such as file system ("_open", "_close", "_read", and "_write"). A transparent interface to remote contexts from Linux user space applications running on the master processor;

OpenAMP Introduction

- Resource manager **rproc_srm**: composed by **system resources** shared between the master and remote cores like clocks, power, reset and memory, and **peripheral resources** assigned and controlled by the master and remote cores without conflicting with each other. Proposed by ST and still in discussion.

- Depends on libmetal acting as an OS environment and hardware abstraction layer;

- On going work to decouple Remoteproc and RPMsg so that they can be used independently;

OpenAMP Introduction

Remoteproc: a.k.a. The LCM (Life Cycle Management) component
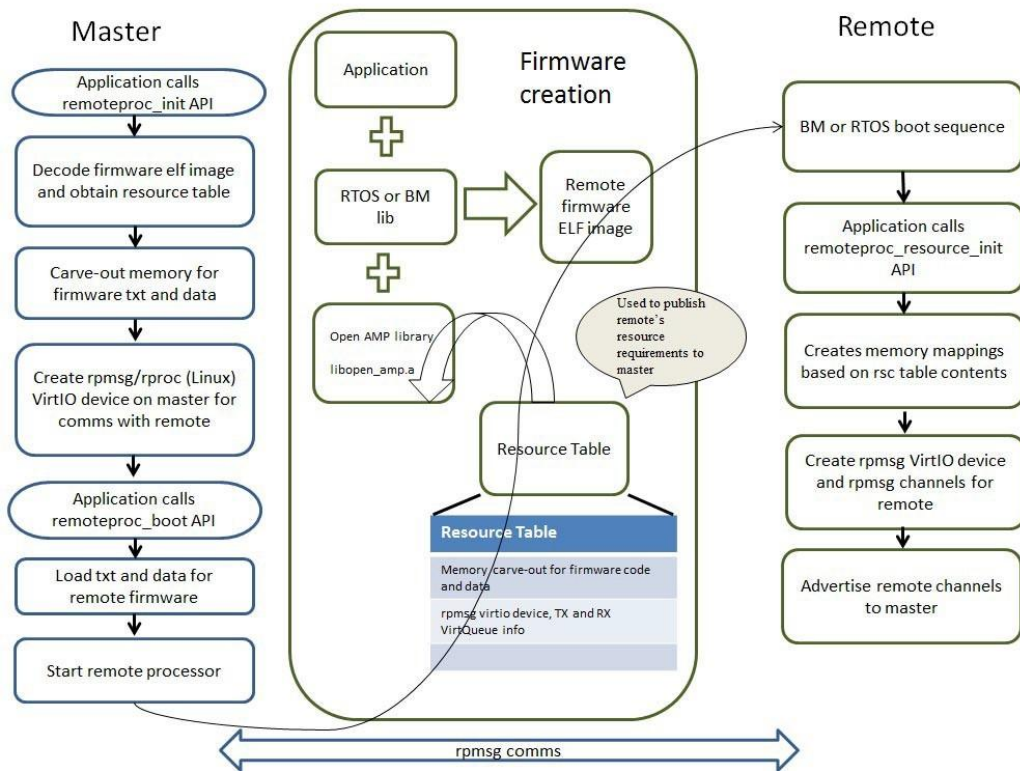
OpenAMP Introduction

Remoteproc:

- **Restriction:** Runtime VirtIO devices creation for RPMsg is not implemented for i.MX devices in the remoteproc driver using data from the resource table;

- Instead, NXP implemented the VirtIO devices, rings and queues creation using data from device tree directly in the rpmsg driver.

Agenda

Real-Time applications with HMP (Heterogeneous Multiprocessing)

OpenAMP Introduction

RPMsg Introduction

RPMsg-lite Introduction

Enabling RPMsg on Linux

Enabling RPMsg-lite on Zephyr

Linux and Zephyr communication setup

Demo

Future work

RPMsg Introduction

RPMsg Protocol Layers:

RPMsg Lite, OpenAMP RPMsg, ...	RPMsg	Transport Layer
VirtIO, Virtqueue, Vring	VirtIO / Virtqueue	MAC Layer
Shmem, MU, Mailbox	Shared Memory Inter-core Interrupts	Physical Layer
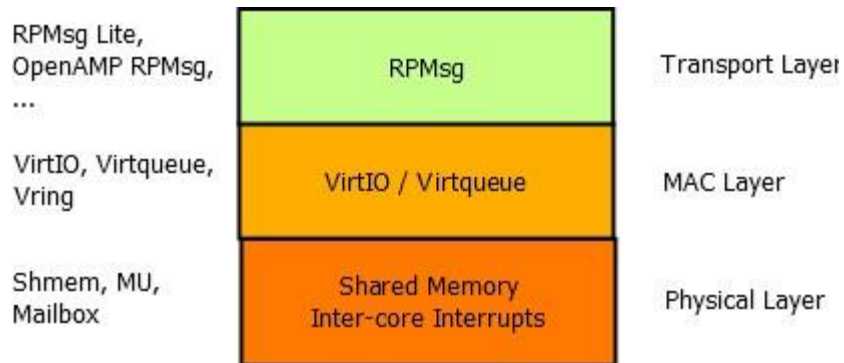
RPMsg Physical Layer – Shared Memory:
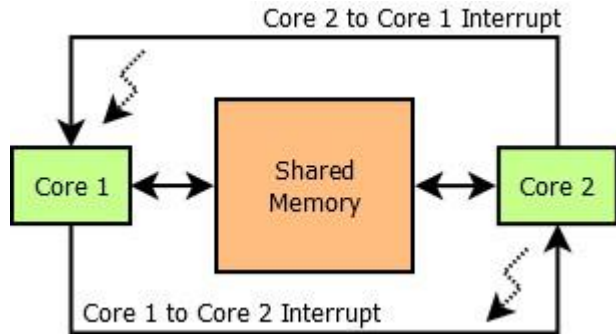
NXP i.MX7S/D/ULP and i.MX6SX
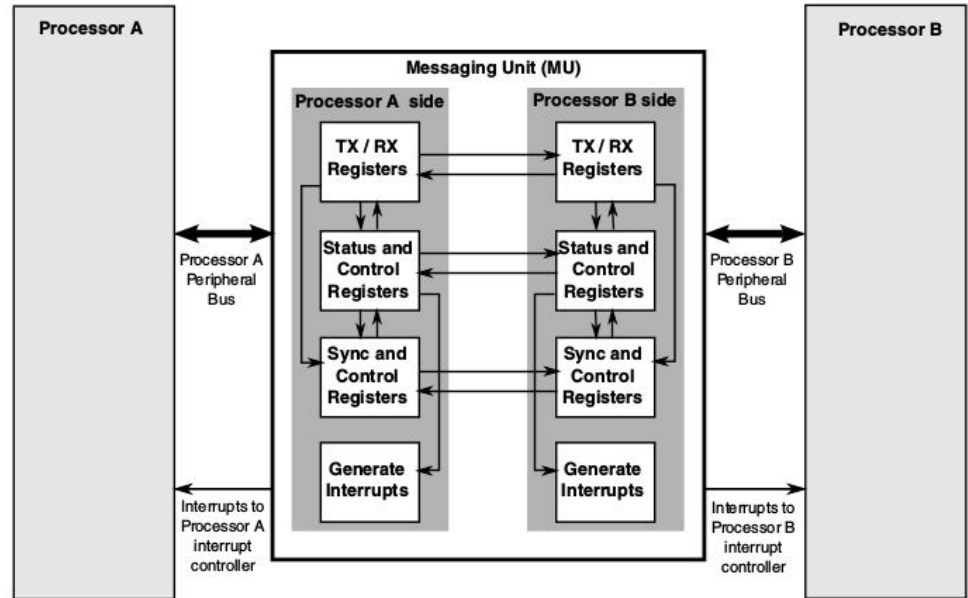
RPMsg Introduction

RPMsg Media Access Layer - VirtIO:

- Used to transfer the user data in shared memory in single-writer single-reader circular buffering technique;

- 2 ring buffers (Used/Available) for each direction (tx,rx);

- Ring buffers contains the addresses of the shared memory with RPMsg data;

- RPMsg Framework Virtio Implementation at OpenAMP wiki;

- More details about VirtIO and vrings structures in OpenAMP wiki page.

RPMsg Introduction

RPMsg Media Access Layer - VirtIO:

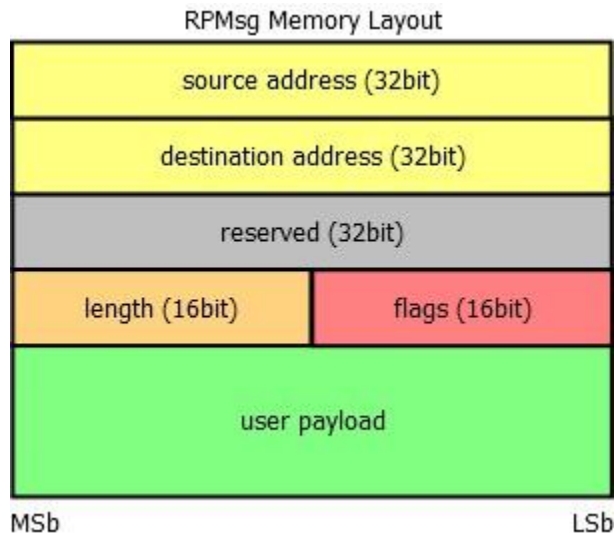- Related presentations with more details:

  - Implementation details of RPMsg on Linux: Asymmetric Multiprocessing and Embedded Linux - Marek Novak & Dušan Červenka, NXP Semiconductor - ELCE 2017 - video, slides;

  - An Introduction to Asymmetric Multiprocessing: When this Architecture can be a Game Changer and How to Survive It - Nicola La Gloria & Laura Nao, Kynetics - ELC 2018 - video, slides.

RPMsg Introduction

RPMsg Transport Layer:

- The RPMsg message is a buffer stored in the shared memory which address is stored in the vring descriptor poll.

RPMsg Memory Layout
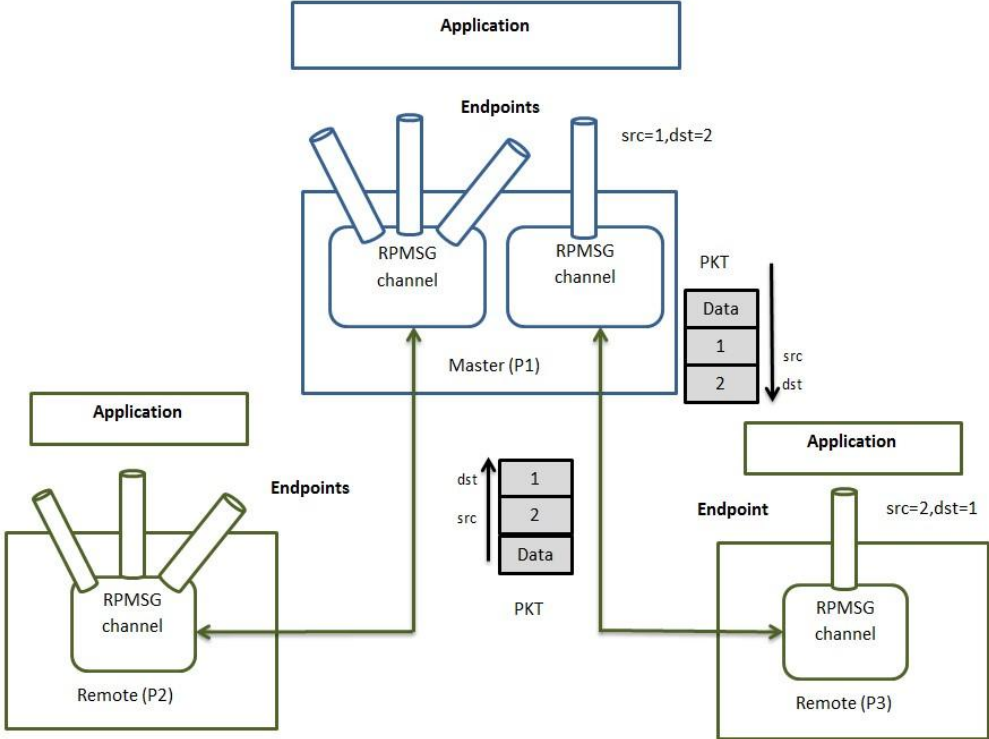
source address (32bit)

destination address (32bit)

reserved (32bit)

length (16bit) | flags (16bit)

user payload

MSb   LSb

RPMsg Introduction

RPMsg Transport Layer:

Agenda

Real-Time applications with HMP (Heterogeneous Multiprocessing)

OpenAMP Introduction

RPMsg Introduction

RPMsg-lite Introduction

Enabling RPMsg on Linux

Enabling RPMsg-lite on Zephyr

Linux and Zephyr communication setup

Demo

Future work

RPMsg-lite Introduction

RPMsg-lite: https://github.com/NXPmicro/rpmsg-lite

- Authored and maintained by Marek Novak;

- Simplification of extensive API of OpenAMP RPMsg implementation;

- Smaller footprint compared to OpenAMP RPMsg implementation;

- Option to use static API (no mallocs) to reduce code size;

- Decoupled from remoteproc;

- Provides no-copy-send no-copy-receive which eliminates the cost of copying data from/to the application to/from the RPMsg/VirtIO buffer;

RPMsg-lite Introduction

RPMsg-lite:

- Two optional sub-components:

  - **Queue**: blocking receive API which is common in RTOS-environments and requires an implementation in the environment adaptation layer;

  - **Name Service**: which is present in the Linux Kernel implementation of RPMsg. It allows both communicating nodes to send announcements about "named" endpoint (a.k.a channel) creation or deletion and to receive these announcements taking any user-defined action in an application callback.

RPMsg-lite Introduction

RPMsg-lite: Architecture
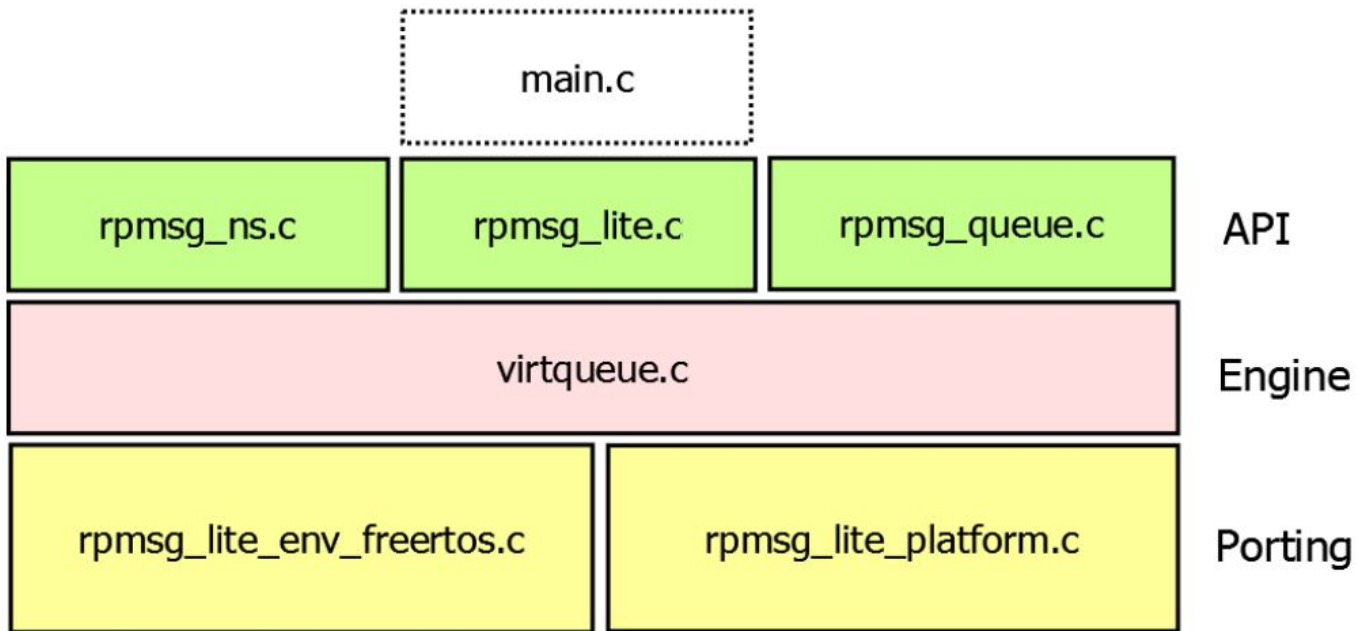
RPMsg-lite Introduction

RPMsg-lite: Implementation

Agenda

Enabling RPMsg on Linux

NXP Linux kernel source tree in 4.9-1.0.x-imx branch for i.MX7 Soc:

- Kconfis automatically set when defconfig `CONFIG_SOC_IMX7` is processed:
  - `CONFIG_HAVE_IMX_MU=y`
  - `CONFIG_HAVE_IMX_RPMSG=y`
  - `CONFIG_RPMSG=y`
  - `CONFIG_RPMSG_VIRTIO=y`
  - `CONFIG_IMX_RPMSG_PINGPONG=m`
  - `CONFIG_IMX_RPMSG_TTY=m`

- i.MX MU driver at: `arch/arm/mach-imx/mu.c`

- i.MX RPMsg driver at: `drivers/rpmsg/imx_rpmsg.c`

- i.MX RPMsg tty driver at: `drivers/rpmsg/imx_rpmsg_tty.c`

Enabling RPMsg on Linux

NXP Linux kernel source tree in 4.9-1.0.x-imx branch for i.MX7 Soc:

- Devicetree entries:

    `arch/arm/boot/dts/imx7s.dtsi:`

```
mu: mu@30aa0000 {
        compatible = "fsl,imx7d-mu", "fsl,imx6sx-mu";
        reg = <0x30aa0000 0x10000>;
        interrupts = <GIC_SPI 88 IRQ_TYPE_LEVEL_HIGH>;
        clocks = <&clks IMX7D_MU_ROOT_CLK>;
        clock-names = "mu";
        status = "okay";
};

rpmsg: rpmsg{
        compatible = "fsl,imx7d-rpmsg";
        status = "disabled";
};
```

Enabling RPMsg on Linux

NXP Linux kernel source tree in 4.9-1.0.x-imx branch for i.MX7 Soc:

- Devicetree entries (cont):

```
arch/arm/boot/dts/imx7s-warp.dts:

    reserved-memory {
        #address-cells = <1>;
        #size-cells = <1>;
        ranges;

        rpmsg_reserved: rpmsg@8fff0000 {
            No-map;
            reg = <0x8fff0000 0x10000>;
        };
    };

    &rpmsg {
        vdev-nums = <1>;
        reg = <0x8fff0000 0x10000>;
        status = "okay";
    };
```

```
arch/arm/boot/dts/imx7s-warp.dts (cont):

    &uart2 {
        status = "disabled";
    };
```

Agenda

Real-Time applications with HMP (Heterogeneous Multiprocessing)

OpenAMP Introduction

RPMsg Introduction

RPMsg-lite Introduction

Enabling RPMsg on Linux

Enabling RPMsg-lite on Zephyr

Linux and Zephyr communication setup

Demo

Future work

Enabling RPMsg-lite on Zephyr

- i.MX MU driver PR 8527 is still in review in Zephyr:
  - Zephyr fork created at github.com/diegosueiro/zephyr/tree/rpmsglite-imx:
    - Includes i.MX MU driver, soc and board changes to support i.MX7 and WaRP7

- Attempt to include RPMsg-lite on Zephyr: PRs 6153 and 5960. But Zephyr TSC chose to only support OpenAMP as the IPC mechanism;

- RPMsg-lite fork to support Zephyr at github.com/diegosueiro/rpmsg-lite/tree/zephyr-support:
  - Support added for i.MX7 processors as well;
  - Added a sample remote echo app.

Enabling RPMsg-lite on Zephyr

i.MX MU driver on Zephyr

- Impletments API defined at: `zephyr/include/ipm.h`

- Driver source code at: `zephyr/drivers/ipm/ipm_imx.c`
  - Option to configured data size: 4, 8 or 16 bytes;
  - Aligned with Linux side, RPMsg uses 4 bytes (MU register index 1) for message direction control (bit 16).

- Device tree defines registers map, interrupt number and RDC permissions
  - Binding at `zephyr/dts/bindings/arm/nxp,imx-mu.yaml`
  - Defined at `zephyr/dts/arm/nxp/nxp_imx7d_m4.dtsi`
  - Enabled at `zephyr/boards/arm/warp7_m4/warp7_m4.dts`

Enabling RPMsg-lite on Zephyr

RPMsg-lite Zephyr environment porting layer:

- Environment API defined at: `rpmsg-lite/lib/include/rpmsg_env.h`;

- Implemented at:
  `rpmsg-lite/lib/rpmsg_lite/porting/environment/rpmsg_env_zephyr.c`;

RPMsg-lite Zephyr platform porting layer for i.MX7:

- Platform global definitions and API defined at:
  `rpmsg-lite/lib/include/platform/imx7d_m4/rpmsg_platform.h`;

- Implemented at:
  `rpmsg-lite/lib/rpmsg_lite/porting/platform/imx7d_m4/rpmsg_platform_zephyr_ipm.c`;

Enabling RPMsg-lite on Zephyr

Zephyr build with RPMsg-lite

- Needs the Inter-processor mailbox subsystem (`CONFIG_IPM`) and the low-level driver implementation (e.g. `CONFIG_IPM_IMX`) selections;

- RPMsg-lite build config selection: `CONFIG_IPC_RPMSG_LITE`;

- Compiled alongside with the application using `Kconfig`, `prj.conf` and `CMakeLists.txt` build settings files:

```
rpmsg-lite/zephyr/samples/subsys/ipc/rpmsg_lite/remote_echo/
├── CMakeLists.txt
├── Kconfig
├── prj.conf
├── prj_warp7_m4.conf
├── sample.yaml
└── src
    └── main_remote_echo.c
```

Agenda

Real-Time applications with HMP (Heterogeneous Multiprocessing)

OpenAMP Introduction

RPMsg Introduction

RPMsg-lite Introduction

Enabling RPMsg on Linux

Enabling RPMsg-lite on Zephyr

Linux and Zephyr communication setup

Demo

Future work

Linux and Zephyr communication setup

Linux - Master Domain	Zephyr - Remote Domain
U-boot loads and starts Zephyr Image and Kernel	
RPMsg driver creates virtqueues and endpoints	RPMsg app creates virtqueues
Notifies remote processor	Waits for link being up
RPMsg driver waits for name service announcement	App creates endpoint and sends name service announcement
Send/Receive messages	Send/Receive messages

Agenda

Real-Time applications with HMP (Heterogeneous Multiprocessing)

OpenAMP Introduction

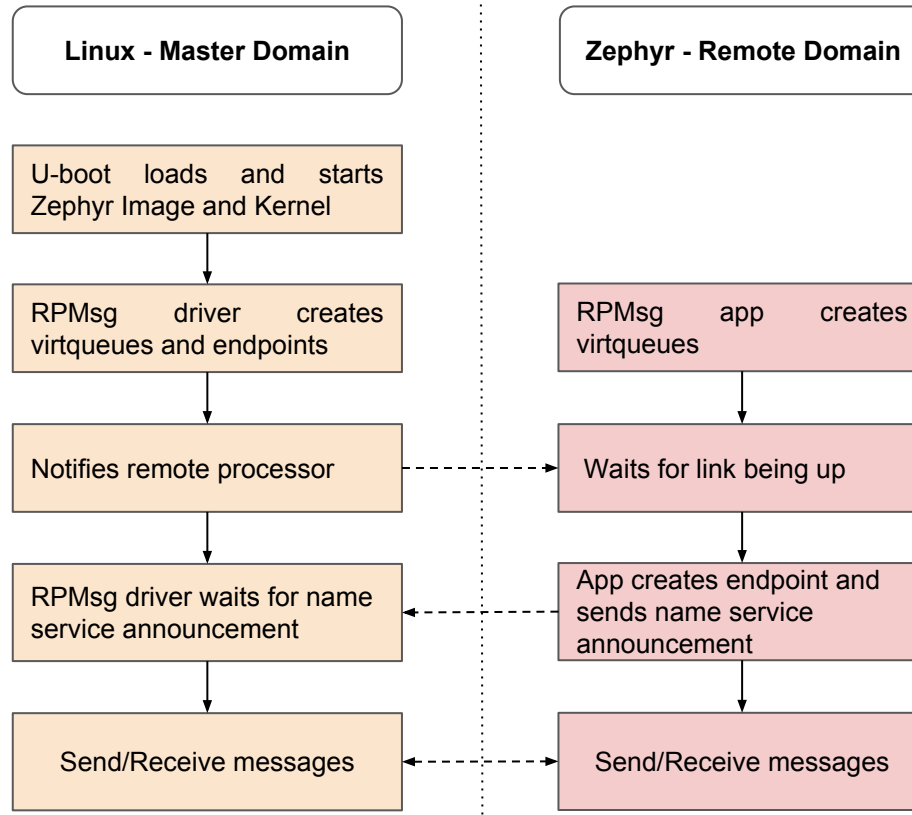RPMsg Introduction

RPMsg-lite Introduction
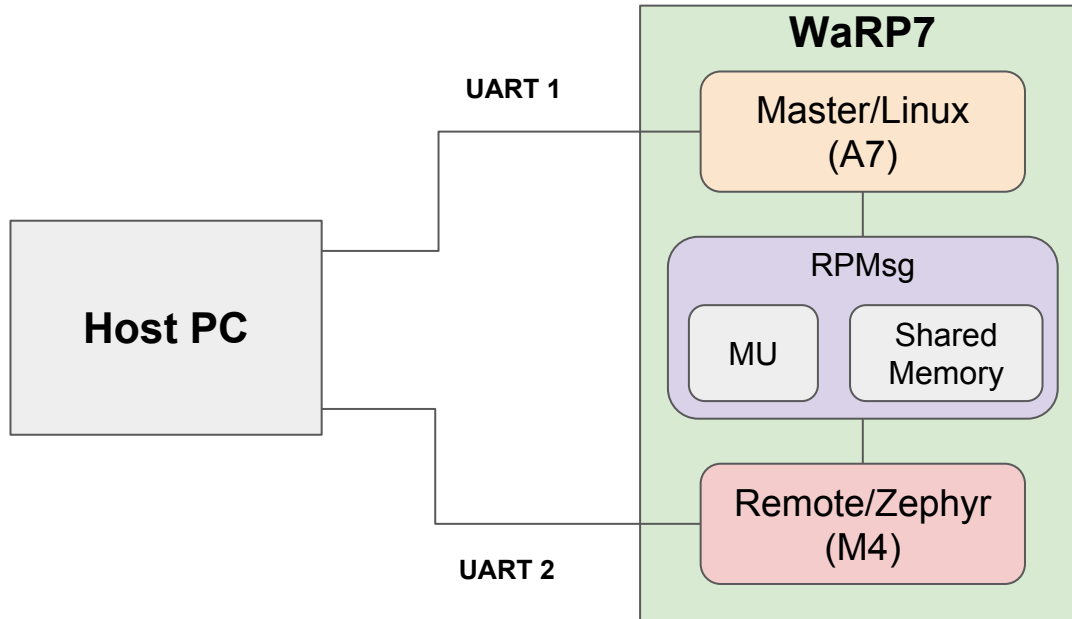
Enabling RPMsg-lite on Zephyr

Enabling RPMsg on Linux

Enabling RPMsg-lite on Zephyr

Demo

Future work

Demo

- Simplified Diagram:

Demo

- Sources M4 Side:
  - Zephyr: github.com/diegosueiro/zephyr/tree/rpmsglite-imx
  - RPMsg-lite: github.com/diegosueiro/rpmsg-lite/tree/zephyr-support
    Demo source location: `rpmsg-lite/zephyr/samples/subsys/ipc/rpmsg_lite/remote_echo`

- Sources A7 Side:
  - Linux Distribution (Yocto based): github.com/diegosueiro/meta-ipc/tree/sumo
  - Linux Kernel: github.com/diegosueiro/linux-fslc/tree/4.9-1.0.x-imx
  - U-boot: github.com/diegosueiro/u-boot-fslc/commits/2018.07+fslc

- Build and flash instructions:
  - Linux Distro: github.com/diegosueiro/meta-ipc/wiki/Yocto-Distribution-Build-and-Flash-Instructions
  - Zephyr Image: github.com/diegosueiro/rpmsg-lite/wiki

Demo

RPMsg-lite remote echo sample

```
rpmsg-lite/zephyr/samples/subsys/ipc/rpmsg_lite/remote_echo/src/main_remote_echo.c:
    <...>

    #define APP_TASK_STACK_SIZE (1024)
    #define LOCAL_EPT_ADDR (30)

    #ifdef CONFIG_SOC_SERIES_IMX7_M4
    /* Settings bellow aligned with Linux i.MX RPMsg side */
    #define RPMSG_MAX_SIZE  256
    #define RPMSG_LITE_LINK_ID (RL_PLATFORM_IMX7D_M4_LINK_ID)
    #define RPMSG_LITE_SHMEM_BASE (0x8FFF0000)
    #define RPMSG_LITE_NS_USED (1)
    #define RPMSG_LITE_NS_ANNOUNCE_STRING "rpmsg-openamp-demo-channel"
    #endif
```

Demo

RPMsg-lite remote echo sample

```
rpmsg-lite/zephyr/samples/subsys/ipc/rpmsg_lite/remote_echo/src/main_remote_echo.c (cont):
    <...>

    void app_task(void *arg1, void *arg2, void *arg3)
    {
            <...>
            char buf[RPMSG_MAX_SIZE];
            char rsp[RPMSG_MAX_SIZE];
            int len;
            volatile unsigned long remote_addr;
            struct rpmsg_lite_endpoint *volatile rl_endpoint;
            volatile rpmsg_queue_handle rl_queue;
            struct rpmsg_lite_instance *volatile rl_instance;
    #ifdef RPMSG_LITE_NS_USED
            volatile rpmsg_ns_handle ns_handle;
    #endif /*RPMSG_LITE_NS_USED*/
```

Demo

RPMsg-lite remote echo sample

```
rpmsg-lite/zephyr/samples/subsys/ipc/rpmsg_lite/remote_echo/src/main_remote_echo.c (cont):

        /* Initialize RPMsg Core and create virtqueues */
         rl_instance = rpmsg_lite_remote_init((void *)RPMSG_LITE_SHMEM_BASE,
                                              RPMSG_LITE_LINK_ID, RL_NO_FLAGS);

         printk("Waiting for Master.\r\n");
         while (!rpmsg_lite_is_link_up(rl_instance)){}

         rl_queue = rpmsg_queue_create(rl_instance);
         rl_endpoint = rpmsg_lite_create_ept(rl_instance, LOCAL_EPT_ADDR, rpmsg_queue_rx_cb,
                                              rl_queue);

    #ifdef RPMSG_LITE_NS_USED
         ns_handle = rpmsg_ns_bind(rl_instance, app_nameservice_isr_cb, NULL);
         rpmsg_ns_announce(rl_instance, rl_endpoint, RPMSG_LITE_NS_ANNOUNCE_STRING,
                           RL_NS_CREATE);
         printk("Nameservice announce sent.\r\n");
    #endif /*RPMSG_LITE_NS_USED*/
```

Demo

RPMsg-lite remote echo sample

```
rpmsg-lite/zephyr/samples/subsys/ipc/rpmsg_lite/remote_echo/src/main_remote_echo.c (cont):
        while(1)
         {
                rpmsg_queue_recv(rl_instance, rl_queue, (unsigned long*)&remote_addr,
                                (char*)buf, sizeof(buf), &recved, RL_BLOCK);

                printk("\nFrom endpoint 0x%X received %d bytes:\n",
                        (unsigned int)remote_addr, recved);
                buf[recved] = '\0';
                printk("%s\n",buf);

                /* Format the echo response */
                len = snprintf(rsp, sizeof(rsp), "echo: %s\r\n", buf);
                printk("Sending %d bytes to endpoint 0x%X:\n", len, (int)remote_addr);
                printk("%s",rsp);
                rpmsg_lite_send(rl_instance, rl_endpoint, remote_addr, rsp, len,
                                RL_BLOCK);
        }
    }
    <...>
```

Agenda

Real-Time applications with HMP (Heterogeneous Multiprocessing)

OpenAMP Introduction

RPMsg Introduction

RPMsg-lite Introduction

Enabling RPMsg on Linux

Enabling RPMsg-lite on Zephyr

Linux and Zephyr communication setup

Demo

Future work

Future Work

- Needs to be upstreamed
  - Linux kernel:
    - i.MX remoteproc to create VirtIO queues for RPMsg from firmware resource table;
    - i.MX MU (mailbox) driver - patchwork;
    - i.MX RPMsg drivers - patchwork;

  - OpenAMP:
    - Remoteproc and RPMsg decoupling - issue.

  - Zephyr:
    - i.MX MU driver - issue;
    - RPMsg-lite as a OpenAMP alternative.

- Message transmission latency measurement by varying:
  - Shared Memory type (internal and external)
  - Static and Dynamic Memory allocation
  - Copy and no-copy mechanisms
  - Message buffer size
  - Number of buffers

References

- Asymmetric Multiprocessing and Embedded Linux - Marek Novak & Dušan Červenka, NXP Semiconductor - ELCE 2017 - video, slides.

- An Introduction to Asymmetric Multiprocessing: When this Architecture can be a Game Changer and How to Survive It - Nicola La Gloria & Laura Nao, Kynetics - ELC 2018 - video, slides.

- Heterogeneous Software Architecture with OpenAMP - Shaun Purvis, Xilinx - ESC Boston 2017 - Slides.

- An Introduction to Heterogeneous Multiprocessing (ARM® Cortex®-A + CortexM) on Next-Generation i.MX Applications Processors - Glen Wienecke, NXP - FTF 2014 - Slides.

- i.MX 7Solo Applications Processor Reference Manual - IMX7SRM.

References

- OpenAMP Github Wiki:
  - OpenAMP Components and Capabilities
  - OpenAMP Life Cycle Management
  - RPMsg Messaging Protocol
  - OpenAMP RPMsg Virtio Implementation

THANK YOU !!!!

Questions?

Diego Sueiro, Embarcados
www.embarcados.com.br
diego.sueiro@gmail.com
linkedin.com/in/diegosueiro/

Embedded Linux Conference Europe 2018