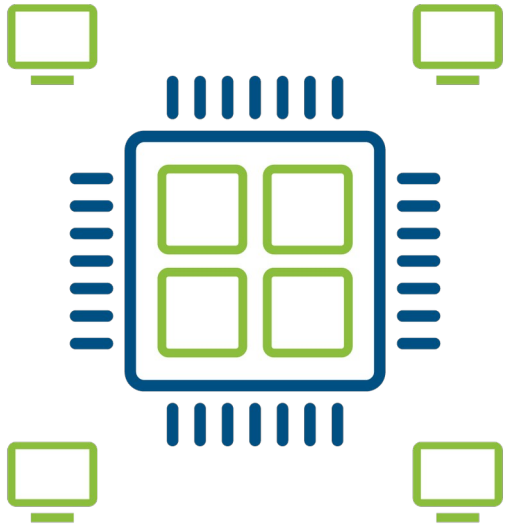
Toradex
Embedded. Computing.

EMBEDDED LINUX CONFERENCE

Asymmetric/Heterogeneous MultiProcessing (AMP/HMP): Mainline Linux and Zephyr in Unison



Presented by
Marcel Ziswiler
Software Team Lead - Embedded Linux BSP
Toradex

Zephyr™

WITH YOU TODAY…

- Joined Toradex 2011
- Spearheaded Embedded Linux Adoption
- Introduced Upstream First Policy
- Top 10 U-Boot Contributor
- Top 10 Linux Kernel ARM SoC Contributor
- Industrial Embedded Linux Platform Torizon Fully Based on Mainline Technology
  - Mainline U-Boot with Distroboot
  - KMS/DRM Graphics with Etnaviv & Nouveau
  - OTA with OSTree
  - Docker resp. Podman

Marcel Ziswiler

Software Team Lead - Embedded Linux BSP

Toradex

WE’RE LOCATED IN LUCERNE, SWITZERLAND

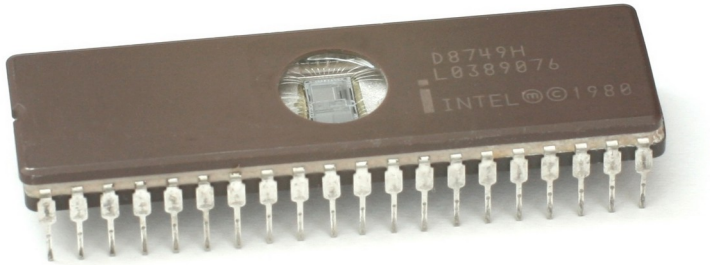
WHAT WE'LL COVER TODAY…

- Evolution of the Microcontroller
- Integration into the Linux Ecosystem
- Quick Overview of Open-Source Real-Time Operating Systems (RTOSes)
- AMP/HMP Lifecycle or How to Actually Launch Code
- Mainline Linux and Zephyr Working in Unison
  - Remote Processor Framework (remoteproc)
  - Remote Processor Messaging (rpmsg)
- Communication Libraries
  - OpenAMP
- Real-Life Demo Using NXP i.MX 7/8M Mini and 8M Plus

Evolution of the Microcontroller

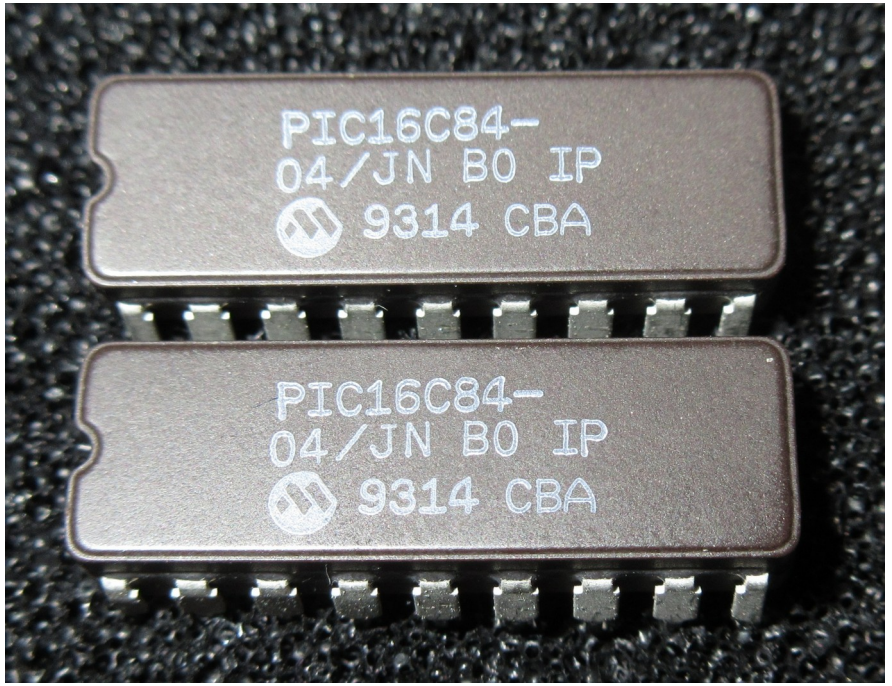
- **Texas Instruments TMS 1000**
  - 4-bit
  - 1971, commercially available 1974
  - Combines read-only memory, read/write memory, processor and clock
- **Intel 8048**

  - 8-bit
  - FCS 1977
  - EPROM for development (expensive ceramic package with quartz window allowing UV erasure)
  - OTP or pre-programmed mask for mass production
- **Intel 8086** (iAPX 86)
  - First 16-bit Microprocessor
  - FCS 1978
  - Required several additional ICs



Evolution of the Microcontroller (cont.)

- **Motorola HC05** (MC68HC805B6, MC68HC805C4 and MC68HC11E2)
  - Serial Bootloader
  - EEPROM program storage
  - Late 1980
- **Microchip PIC16C84**
  - EEPROM
  - 1993
  - Rapid prototyping using in-system programming
- 1993 Atmel also introduced first microcontroller using flash memory
- 16-bit MCU dominant volume since 2011



Integration into the Linux Ecosystem

- **Independent Microcontrollers**
  - Interfacing:
    - I2C, memory mapped IO (MMIO) bus, SPI: all supported by in-kernel subsystems
    - Generic register map support
      (regmap: Documentation/devicetree/bindings/regmap, drivers/base/regmap)
  - Firmware loading using in-kernel firmware API
    - Introduced to update microcode for CPU errata
    - Device driver firmware, required to be loaded onto device microcontrollers
    - Device driver information data (calibration data, EEPROM overrides which may be completely optional)
    - Firmware requests
      - Synchronous

```
if(request_firmware(&fw_entry, $FIRMWARE, device) == 0)
        copy_fw_to_device(fw_entry->data, fw_entry->size);
release_firmware(fw_entry);
```

Integration into the Linux Ecosystem (cont.)

- Asynchronous
- Special optimization on reboot
  `firmware_request_cache()`

- Firmware upload API
  - Persistent sysfs nodes to enable users to initiate firmware updates for a device
  - E.g. Intel Stratix10 SoC service layer (FPGA programming)
- `Documentation/driver-api/firmware/introduction.rst`

- Multifunction miscellaneous devices/multifunction device drivers (mfd)
  - Heterogeneous hardware blocks containing more than one non-unique yet varying hardware functionality
  - External bus interfacing
  - Memory registers containing "miscellaneous system registers" also known as a system controller "syscon" or any other memory range containing a mix of unrelated hardware devices
  - `Documentation/devicetree/bindings/mfd/mfd.txt`

- **AMP/HMP integrated:** remoteproc, rpmsg, ...

Quick Overview of Open-Source Real-Time Operating Systems - eCos

Toradex
Embedded. Computing.

- Embedded Configurable Operating System (eCos)
- One process with multiple threads
- Customizable to precise application requirements
- Implemented in programming languages C and C++
- Portable Operating System Interface (POSIX) API
- Nucleus (µITRON) compatibility layer
- Initially developed in 1997 by Cygnus Solutions
- Later bought by Red Hat which ceased development early 2002
- 2004 Red Hat agreed to transfer eCos copyrights to Free Software Foundation
- eCosPro commercial fork of eCos created by eCosCentric incorporates proprietary software components
- License: Modified GNU GPL

FreeRTOS

- FreeRTOS kernel originally developed by Richard Barry around 2003
- Later developed and maintained by Barry's company Real Time Engineers Ltd.
- 2017 stewardship passed to Amazon Web Services (AWS)
- Designed to be small and simple
- Mostly written in the C programming language
- Easy to port and maintain (ported to 35 microcontroller platforms)
- Few assembly language functions (architecture-specific scheduler routines)
- License: MIT

Mbed OS

- Intended for Internet of Things (IoT) devices
- Based on Keil RTX5
- For 32-bit ARM Cortex-M microcontrollers
- Collaboratively developed by Arm and its technology partners
- Mbed online IDE (browser only), free online code editor and compiler run in the cloud
- Other development environments such as Keil µVision, IAR Embedded Workbench, and Eclipse with GCC ARM Embedded tools
- C/C++ software platform
- Components database provides driver libraries for components and services
- License: Apache 2.0

MicroC/OS-III

Toradex
Embedded. Computing.

- Micro-Controller Operating Systems (MicroC/OS, stylized as µC/OS)
- Designed by Jean J. Labrosse in 1991
- Published originally in three-part article in Embedded Systems Programmi
  Real-Time Kernel
- Priority-based pre-emptive real-time kernel
- Mostly written in programming language C (target microprocessor-specific code written in assembly language)
- Version 2 introduced as commercial product in 1998
- Manages up to 64 tasks, allows only 1 task at each of 255 priority levels
- Version 3 introduced in 2009
- Allows any number of application tasks, priority levels, and tasks per level, limited only by processor access to memory
- License: Apache 2.0

NuttX

Toradex
Embedded. Computing.

- Emphasis on technical standards compliance and small size
- Scalable from 8-bit to 64-bit microcontroller environments
- Portable Operating System Interface (POSIX)
- American National Standards Institute (ANSI)
- First released in 2007 by Gregory Nutt as free and open-source software under permissive BSD license
- Since December 2019 undergoing incubation at The Apache Software Foundation
- Written almost exclusively in programming language C
- Uses Kconfig to configure and generate GNU makefiles
- Program distribution combines kernel and substantial amount of middleware and code for board support and device drivers
- Gregory Nutt maintains source code exclusively, must approve all community contributions
- License: Apache 2.0

RIOT

Toradex
Embedded. Computing.

- Small operating system for networked, memory-constrained systems
- Focus on low-power wireless Internet of things (IoT) devices
- Initially developed by Free University of Berlin (FU Berlin), French Institute for Research in Computer Science and Automation (INRIA) and Hamburg University of Applied Sciences (HAW Hamburg)
- Kernel mostly inherited from FireKernel (originally developed for sensor networks)
- Based on microkernel architecture
- Allows applications written in C/C++ and Rust by an experimental API
- Full multithreading and real-time abilities
- Runs on 8-bit, 16-bit and 32-bit processors
- Partly POSIX compliant
- License: GNU LGPL

RT-Thread

- For embedded systems and Internet of Things (IoT)
- Developed by the RT-Thread Development Team based in China
- Aimed to change microcontroller RTOS landscape in China
- 2006 development of first variant named Standard
- 2017 development of second variant named Nano
- For resource-constrained microcontrollers (with as low as 3 KB of flash or ROM and 1.2 KB of RAM)
- Reported to be RTOS with 3rd largest number of contributors (behind Zephyr and Mbed)
- License: Apache 2.0

RTEMS

- Real-Time Executive for Multiprocessor Systems (RTEMS)
- Formerly Real-Time Executive for Missile Systems resp. Real-Time Executive for Military Systems
- Development began in late 1980s
- Early versions available via File Transfer Protocol (ftp) as early as 1993
- OAR Corporation currently managing the RTEMS project in cooperation with steering committee (includes user representatives)
- Support various open application programming interface (API) standards
- Portable Operating System Interface (POSIX) and µITRON
- Classic RTEMS API originally based on Real-Time Executive Interface Definition (RTEID) specification
- Includes port of FreeBSD Internet protocol suite (TCP/IP stack)
- Support for various file systems including Network File System (NFS) and File Allocation Table (FAT)
- Provides extensive multi-processing and memory-management services
- System-database alongside many other facilities
- Extensive documentation
- License: modified GNU GPL

Zephyr

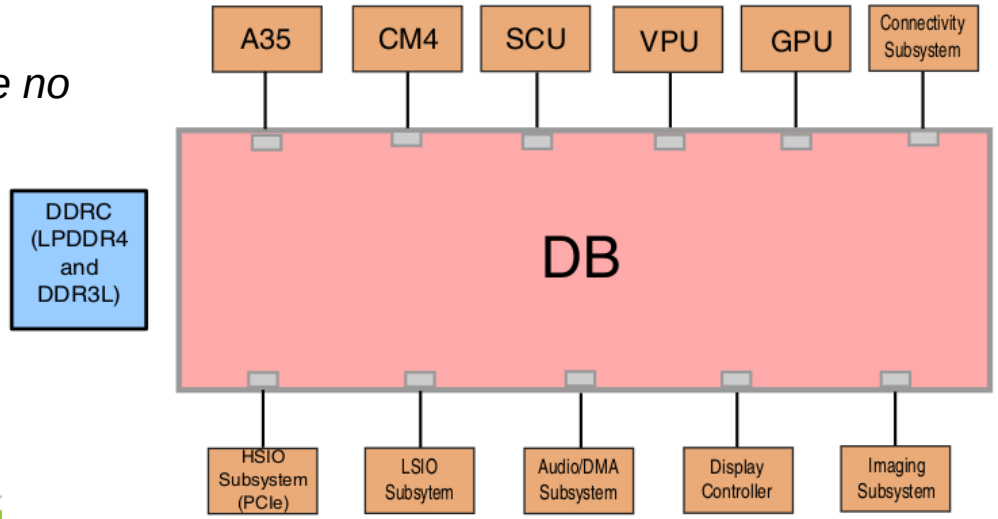
- Originated from Virtuoso RTOS for digital signal processors (DSPs)
- Wind River Systems acquired Belgian software company Eonic Systems in 2001
- Renamed to Rocket, made it open-source and royalty-free in November 2015
- Smaller memory needs (compared to VxWorks), suitable for sensors and single-function embedded devices
- Hosted collaborative project of the Linux Foundation under the name Zephyr since February 2016
- Early members and supporters of Zephyr include Intel, NXP Semiconductors, Synopsys, Linaro, Texas Instruments, DeviceTone, Nordic Semiconductor, Oticon, and Bose
- Small monolithic kernel
- Flexible configuration and build system for compile-time definition of required resources and modules
- Set of protocol stacks (IPv4 and IPv6, Constrained Application Protocol (CoAP), LwM2M, MQTT, 802.15.4, Thread, Bluetooth Low Energy, CAN)
- Virtual file system interface with several flash file systems for non-volatile storage (FATFS, LittleFS, NVS)
- Management and device firmware update mechanisms
- Uses Kconfig and devicetree (but implemented in Python for portability reason), build system based on CMake
- Largest number of contributors and commits compared to other RTOSes as of January 2022
- License: Apache 2.0

AMP/HMP Lifecycle or How to Actually Launch Code - 1. From “boot container” by boot ROM

- Not supported on the i.MX 7Solo
  ”The Cortex-A7 is the primary boot on the i.MX7Solo. The Cortex-M4 core can be enabled during boot as a secondary core to handle timing-critical tasks, but it cannot be used as the boot core.” (i.MX 7Solo Applications Processor Reference Manual, Rev. 0, 05/2016, page 179)
- Not supported on the i.MX 7Dual
  ”The Cortex-A7 is the primary boot on the i.MX7Dual. The Cortex-M4 core can be enabled during boot as a secondary core to handle timing-critical tasks, but it cannot be used as the boot core.” (i.MX 7Dual Applications Processor Reference Manual, Rev. 1, 01/2018, page 233)
- Not supported on the i.MX 8M Mini
  ”The chip will always boot from the A53 core first, the M4 core will be held in reset during the A53 boot and won’t run until it is enabled by the A53 core. The image for the M4 core will be loaded into memory and authenticated by the A53 core.” (i.MX 8M Mini Applications Processor Reference Manual, Rev. 3, 11/2020, page 18)
- Not supported on the i.MX 8M Plus
  ”The chip will always boot from the A53 core first, the M7 core will be held in reset during the A53 boot and won’t run until it is enabled by the A53 core. The image for the M7 core will be loaded into memory and authenticated by the A53 core.” (i.MX 8M Plus Applications Processor Reference Manual, Rev. 1, 06/2021, page 21)

1. From “boot container” by boot ROM (cont.)

- Supported on the i.MX 8QuadMax
  ”The System Controller Unit (SCU) is made of a Cortex-M4 processor running at 266MHz with 256KB of TCM and a set of peripherals and interfaces to connect to external PMIC and to control internal subsystems. The SCU Cortex-M4 is the first processor to boot the chip (see System Boot).” (i.MX 8QuadMax Applications Processor Reference Manual, Rev. 0, 9/2021, page 16)
- Supported on the i.MX 8X
  ”The System Controller Unit (SCU) is made of a Cortex-M4 processor running at 266MHz with 256KB of TCM and a set of peripherals and interfaces to connect to external PMIC and to control internal subsystems. The SCU Cortex-M4 is the first processor to boot the chip (see System Boot).” (i.MX 8DualX/8DualXPlus/8QuadXPlus Applications Processor Reference Manual, Rev. 0, 05/2020, page 16)
- However, *“The Application Processors (AP) and Cortex-M4s (M4) have no direct access to those hardware mechanisms. The hardware mechanisms are abstracted from the AP and M4 by the SCU firmware.”*
- And this SCU firmware (SCFW) is highly proprietary closed-source!
- Other SoC vendors might support such use-cases better



2. From the U-Boot boot loader using bootaux command

- Support boot auxiliary core: CONFIG_IMX_BOOTAUX (arch/arm/mach-imx/Kconfig)
- Available on i.MX 6(SX)/7/8M(incl. Mini and Plus) and Vybrid VF610
- Implementation of SoC specifics (arch/arm/mach-imx/{mx7|imx8m}/soc.c)
- Implementation of actual command: bootaux [addr] (arch/arm/mach-imx/imx_bootaux.c)
- Optional memory reservation in Linux via device tree:
  CONFIG_BOOTAUX_RESERVED_MEM_{BASE|SIZE} (arch/arm/mach-imx/imx8/fdt.c)
- Supports M4 (resp. M7) firmware as raw binaries (e.g. on i.MX 7):
  setenv bootm4 'load mmc 0:1 ${loadaddr} zephyr.bin && \
  cp.b ${loadaddr} 0x007F8000 ${filesize} && dcache flush && bootaux 0x007F8000'
- Or elf files (elf header already contains address the binary has been linked to):
  setenv bootm4 'load mmc 0:1 ${loadaddr} zephyr.elf && bootaux ${loadaddr}'

3. From Linux using remote processor framework (remoteproc)

- Modern SoCs with heterogeneous remote processor devices in asymmetric multiprocessing (AMP) configurations
- Allows different platforms/architectures to control (power on, load firmware, power off) those remote processors while abstracting hardware differences (e.g. avoid driver code duplication)
- Also adds rpmsg virtio devices for remote processors that support this kind of communication
- User API
  - Boot remote processor (load its firmware, power it on, …)
    `int rproc_boot(struct rproc *rproc)`
  - Power off remote processor (previously booted with `rproc_boot()`)
    `int rproc_shutdown(struct rproc *rproc)`
    - Not decrementing rproc refcount, only power refcount
    - Users can still use it in subsequent `rproc_boot()` calls, if needed
  - Find rproc handle using device tree phandle
    `struct rproc *rproc_get_by_phandle(phandle phandle)`

3. From Linux using remote processor framework (remoteproc, cont.)

- Implementors API
  - Allocate new remote processor handle
    struct rproc *rproc_alloc(struct device *dev, const char *name, const struct rproc_ops *ops, const char *firmware, int len)
  - Free rproc handle
    void rproc_free(struct rproc *rproc)
  - Register @rproc with remoteproc framework
    int rproc_add(struct rproc *rproc)
  - Unroll rproc_add()
    int rproc_del(struct rproc *rproc)
  - Report crash in remoteproc
    void rproc_report_crash(struct rproc *rproc, enum rproc_crash_type type)

3. From Linux using remote processor framework (remoteproc, cont.)

- Implementation callbacks

```
/**
 * struct rproc_ops - platform-specific device handlers
 * @start:    power on the device and boot it
 * @stop:     power off the device
 * @kick:     kick a virtqueue (virtqueue id given as a parameter)
 */
struct rproc_ops {
      int (*start)(struct rproc *rproc);
      int (*stop)(struct rproc *rproc);
      void (*kick)(struct rproc *rproc, int vqid);
};
```

- Kick: interrupt remote processor to let it know it has pending messages in a particular virtqueue

- Binary Firmware Structure
- Supports ELF32 and ELF64 firmware binaries

3. From Linux using remote processor framework (remoteproc, cont.)

- Linux kernel configuration
  CONFIG_REMOTEPROC
  CONFIG_IMX_REMOTEPROC
- Device tree

```
imx7d-cm4 {
        compatible = "fsl,imx7d-cm4";
        clocks =  <&clks IMX7D_ARM_M4_ROOT_CLK>;
        fsl,auto-boot;
        memory-region = <&cm4tcmcode>, <&cm4sramcode>;
        syscon = <&src>;
};
```

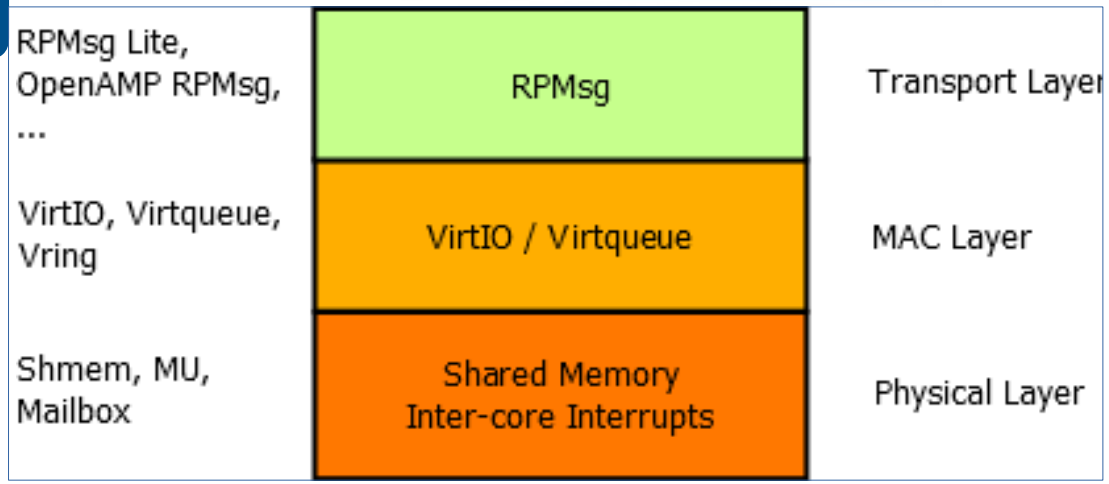
```
reserved-memory {
    #address-cells = <1>;
    #size-cells = <1>;
    ranges;
    cm4tcmcode: cm4tcmcode@7f8000 {
            compatible = "shared-dma-pool";
            reg = <0x007f8000 0x8000>;
            no-map;
    };
    cm4sramcode: cm4sramcode@900000 {
            compatible = "shared-dma-pool";
            reg = <0x00900000 0x40000>;
            no-map;
    };

};
```

3. From Linux using remote processor framework (remoteproc, cont.)

- Hands-on using sysfs interface
  - After having started M4 via bootaux
    root@colibri-imx7-emmc:~# cat /sys/devices/platform/imx7d-cm4/remoteproc/remoteproc0/state
    attached
  - Stopping M4
    root@colibri-imx7:~# echo stop > /sys/devices/platform/imx7d-cm4/remoteproc/remoteproc0/state
    [5251.311146] remoteproc remoteproc0: stopped remote processor imx-rproc
    root@colibri-imx7-emmc:~# cat /sys/devices/platform/imx7d-cm4/remoteproc/remoteproc0/state
    offline
  - Using new firmware
    root@colibri-imx7-emmc-02997126:~# cp zephyr.elf /lib/firmware/rproc-imx-rproc-fw
  - And starting it again using that new firmware
    root@colibri-imx7:~# echo start > /sys/devices/platform/imx7d-cm4/remoteproc/remoteproc0/state
    [212.759189] remoteproc remoteproc0: powering up imx-rproc
    [212.761746] remoteproc remoteproc0: Booting fw image rproc-imx-rproc-fw, size 427580
    [212.761880] remoteproc remoteproc0: No resource table in elf
    [212.762092] remoteproc remoteproc0: remote processor imx-rproc is now up
    root@colibri-imx7-emmc:~# cat /sys/devices/platform/imx7d-cm4/remoteproc/remoteproc0/state
    running

Mainline Linux and Zephyr Working in Unison - Rpmsg



- **Remoteproc** (just covered during lifecycle discussion before)
- **Rpmsg**
  - Virtio-based messaging bus
  - Allows kernel drivers to communicate with remote processors
  - Security implications (e.g. remote processors might have full or at least partial access to memory, peripherals etc.)
  - Rpmsg device: communication channel identified by name, local *source* and remote *destination* address
  - Listening on channel means rx callback is bound with a unique rpmsg local address
  - Rpmsg core will dispatch incoming messages according to destination address
  - Implemented using virtio:
    - Mailbox style synchronization: tx, rx, and rxdb (doorbell)
    - Shared vring buffers

Rpmsg (cont.)

- **Linux**
  - Device tree bindings
    Documentation/devicetree/bindings/mailbox/fsl,mu.yaml
    Documentation/devicetree/bindings/virtio/mmio.yaml
    Documentation/devicetree/bindings/virtio/virtio-device.yaml
  - Documentation/staging/rpmsg.rst
    CONFIG_IMX_MBOX
    drivers/mailbox/imx-mailbox.c
    CONFIG_VIRTIO, CONFIG_VIRTIO_MMIO
    drivers/rpmsg/virtio_rpmsg_bus.c
  - Don't forget to disable peripherals used by Zephyr on M4/M7 (e.g. GPIOs, UART)
  - Device tree

```
imx7d-cm4 {
        compatible = "fsl,imx7d-cm4";
        mbox-names = "tx", "rx", "rxdb";
        mboxes = <&mu0b 0 1
                  &mu0b 1 1
                  &mu0b 3 1>;
        memory-region = <&rpmsg_vrings>, <&cm4tcmcode>, <&cm4sramcode>;
        syscon = <&src>;
};
```

Rpmsg (cont.)

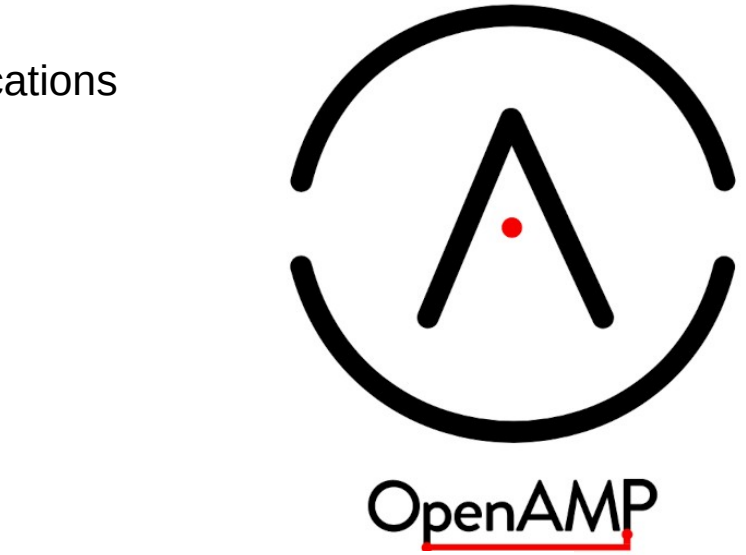
- Device tree (cont.)

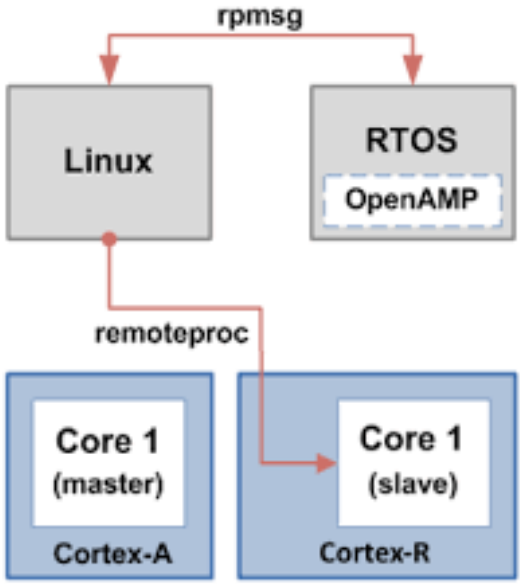
```
reserved-memory {
#address-cells = <1>;
#size-cells = <1>;
ranges;
rpmsg_vrings: vrings0@8ff00000 {
                reg = <0x8ff00000 0x100000>;
                no-map;
        };
};
...
&mu0b {
        status = "enabled";
};
```

- **Zephyr**
  - That’s where OpenAMP comes into play...

Open Asymmetric Multi-Processing (OpenAMP)

- Framework providing software components to enable development of AMP applications
- Linaro Community Project
- Allows operating systems to interact within broad range of complex heterogeneous architectures
- Allows asymmetric multiprocessing applications to leverage parallelism offered by multicore configuration.
- Life cycle management
- Inter processor communication (IPC)
- Provides stand alone library usable with RTOS and Baremetal
- Compatibility with upstream Linux remoteproc and rpmsg components
- Supported AMP configurations:
  - Linux host/Generic(Baremetal) remote
  - Generic(Baremetal) host/Linux remote
- Proxy infrastructure and supplied demos showcase ability to handle printf, scanf, open, close, read, write calls from Bare metal based remote contexts





Open Asymmetric Multi-Processing (OpenAMP, cont.)

- OpenAMP source structure:

```
|- lib/
|  |- virtio/     # virtio implementation
|  |- rpmsg/      # rpmsg implementation
|  |- remoteproc/ # remoteproc implementation
|  |- proxy/      # implement one processor access device on the
|  |              # other processor with file operations
|- apps/        # demonstration/testing applications
|  |- examples/ # Application samples using the OpenAMP framework.
|  |- machine/  # common files for machine can be shared by applications
|  |            # It is up to each app to decide whether to use these files.
|  |- system/   # common files for system can be shared by applications
|               # It is up to each app to decide whether to use these files.
|- cmake        # CMake files
|- script       # helper scripts (such as checkpatch) for contributors.
```

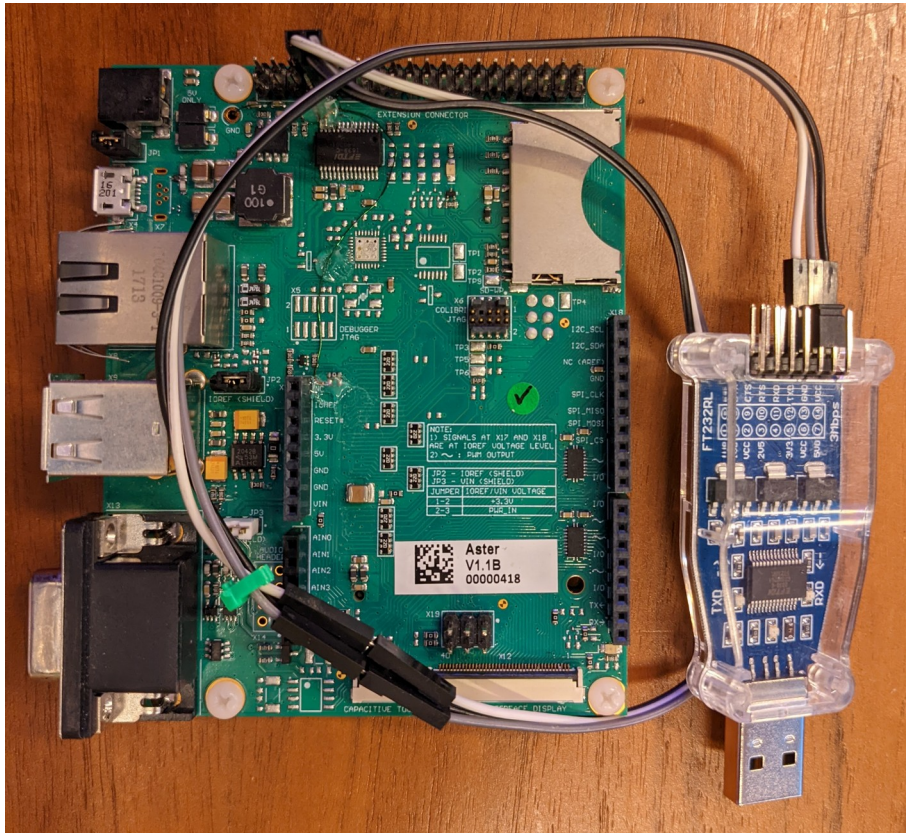
- zephyrproject/modules/lib/open-amp

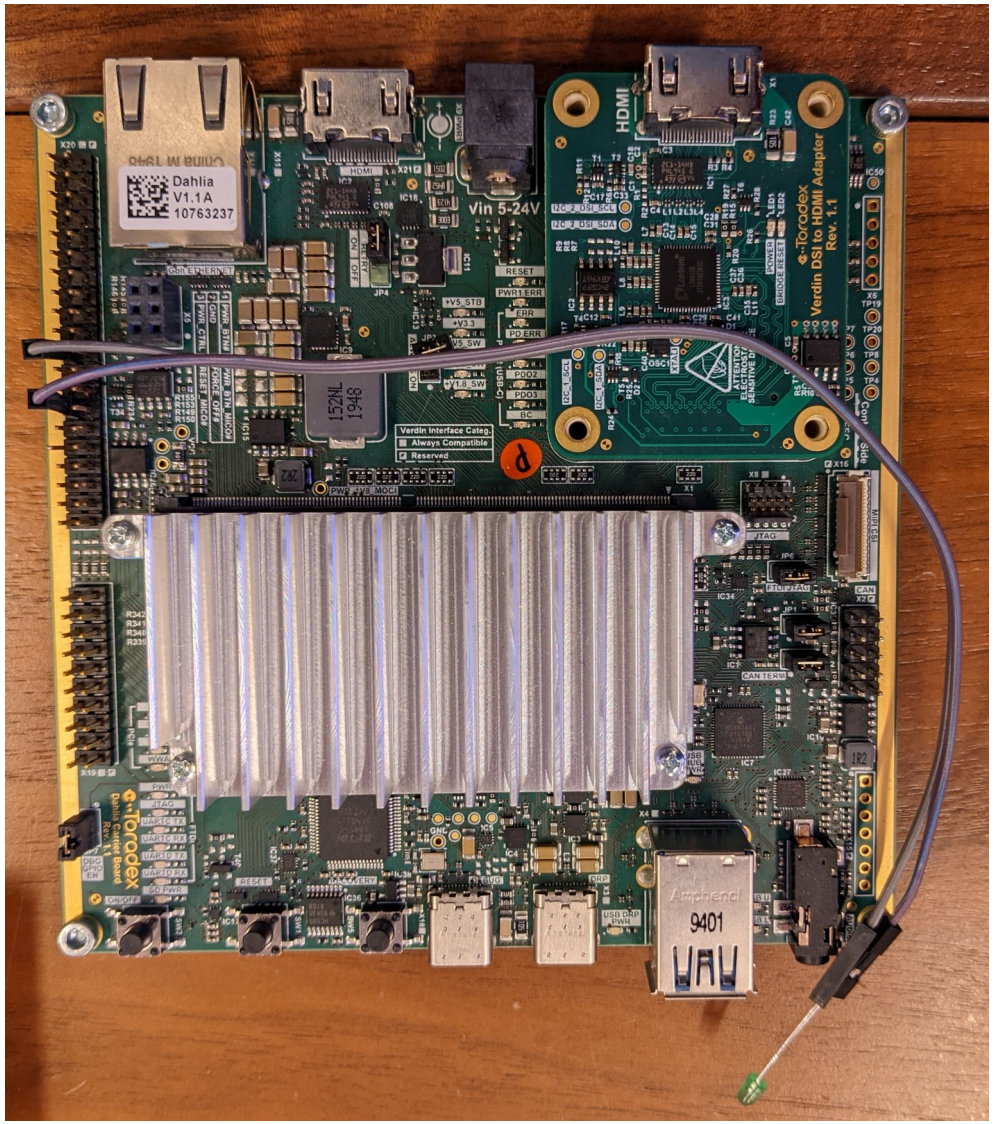
Other Communication Libraries

- **RPMSG-Lite**
  - Lightweight implementation of remote processor messaging (RPMsg) protocol
  - Compared to RPMsg implementation of OpenAMP it offers code size reduction, API simplification, and improved modularity
  - Recommended on smaller Cortex-M0+ based systems
  - Developed by NXP Semiconductors
  - Released under BSD-compatible license
- **eRPC** (Embedded RPC)
  - Remote procedure call (RPC) system for multichip embedded systems and heterogeneous multicore SoCs
  - Designed for tightly coupled systems
  - Using plain C for remote functions
  - Small code size (< 5 kB)
  - Among others can use RPMsg-Lite as transport
  - Unrestrictive BSD 3-clause license

Real-Life Demo Using NXP i.MX 7/8M Mini and 8M Plus

- AMP/HMP Lifecycle
- ...





Toradex
Embedded. Computing.

Q&A

References

- Real-Time Operating Systems (RTOSes)
  https://en.wikipedia.org/wiki/Comparison_of_real-time_operating_systems
- Zephyr on Colibri iMX7
  https://docs.zephyrproject.org/2.6.0/boards/arm/colibri_imx7d_m4/doc/index.html
- Remoteproc
  https://docs.kernel.org/staging/remoteproc.html
- Rpmsg
  https://docs.kernel.org/staging/rpmsg.html
- OpenAMP
  https://github.com/OpenAMP/open-amp
- OpenAMP on Zephyr
  https://docs.zephyrproject.org/2.6.0/samples/subsys/ipc/openamp/README.html
- RPMsg-Lite
  https://github.com/NXPmicro/rpmsg-lite
- eRPC (Embedded RPC)
  https://github.com/EmbeddedRPC/erpc

Toradex
Embedded. Computing.

EMBEDDED LINUX CONFERENCE

THANK YOU
FOR YOUR INTEREST

www.**toradex**.com
developer.toradex.com
community.toradex.com
labs.toradex.com

Torizon™