Multicore Application Design Using Embedded Procedure Call (eRPC) Library

M. Princ, P. Lukáš, M. Chromec, D. Červenka, M. Novák, M. Cingel

ABSTRACT

With newly designed NXP multicore platforms there is a need for software components that would ensure efficient communication between individual cores and rational usage of the multicore computational power. The software for multicore plays a vital role and can essentially influence the overall performance of the multicore system. Several multicore components has be developed by NXP:

- Embedded Remote Procedure Call (eRPC)
- Remote Processor Messaging Lite (RPMsg-Lite)
- Multicore Manager (MCMGR)

eRPC

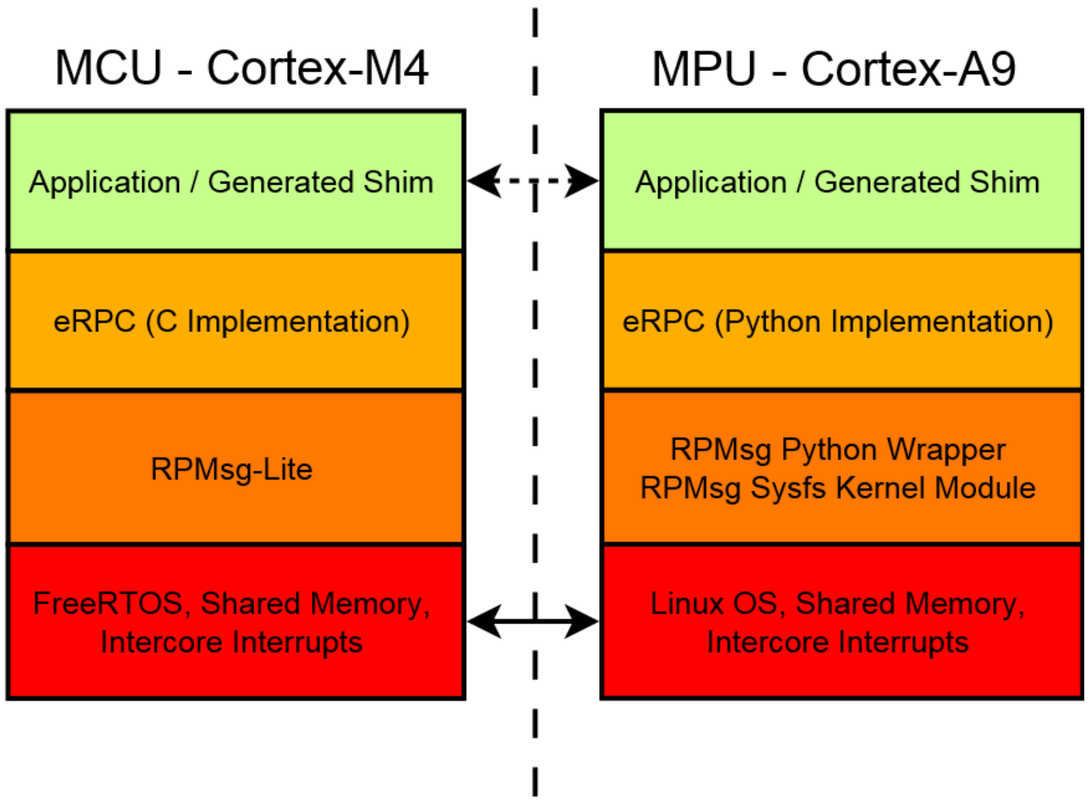
eRPC (Embedded Remote Procedure Call) is the RPC system created by NXP. The RPC is a mechanism used to invoke a software routine on a remote system via a simple local function call. When a remote function is called by the client, the function's parameters and an identifier for the called routine are marshalled (or serialized) into a stream of bytes. This byte stream is transported to the server through a communications channel (IPC, TPC/IP, UART, etc). The server unmarshalls the parameters, determines which function was invoked, and calls it. If the function returns a value, it is marshalled and sent back to the client. Main eRPC features:

- Scalable from bare metal to Linux® OS – configurable memory and threading policies
- Focus on embedded systems - intrinsic support for C, modular, and lightweight implementation.
- Abstracted transport interface – RPMsg-Lite is be the primary transport for multicore, UART or SPI-based solutions can be used for both multichip and multicore.

RPMsg-Lite

RPMsg-Lite is a lightweight implementation of the RPMsg protocol. The RPMsg protocol defines a standardized binary interface used to communicate between multiple cores in a heterogeneous multicore system. Compared to the OpenAMP implementation, RPMsg-Lite offers a code size reduction, API simplification, and improved modularity. Main RPMsg protocol features:

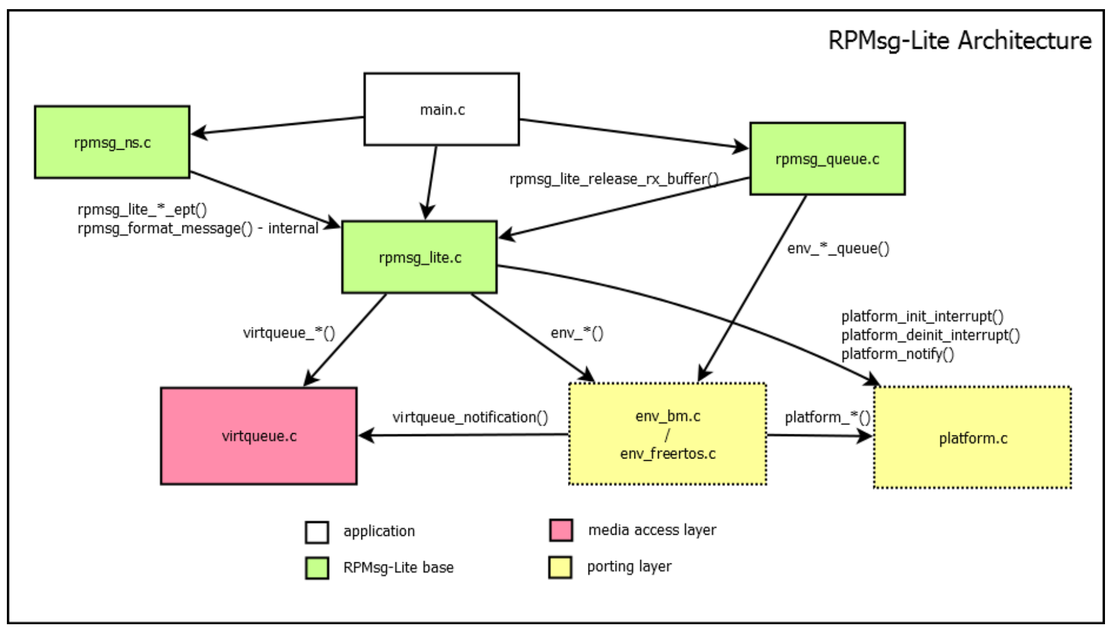
- Shared memory interprocessor communication
- Virtio-based messaging bus
- Application-defined messages sent between endpoints
- Portable to different environments/platforms
- RPMsg protocol available in upstream Linux OS

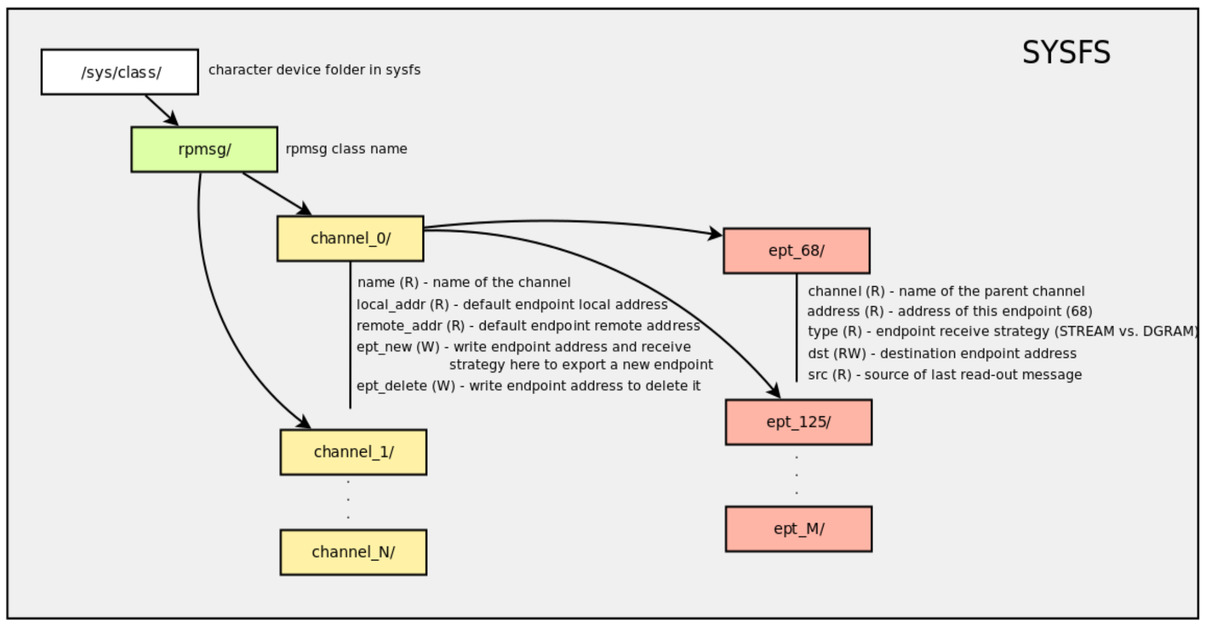


RPMsg multiendpoint Linux sysfs kernel driver

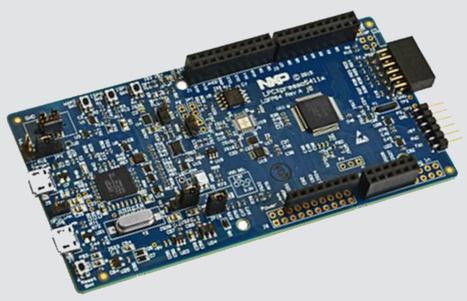
Newly designed and implemented Linux kernel module that is capable of exporting multiple channels and multiple endpoints to the user space. To export a new endpoint to the user space, a way similar to one used in GPIO was adopted - user has to write in a file the address of the new endpoint and its receiving strategy (datagram=message based OR stream). Then UDEV creates a new device in /dev and endpoint attributes appear in /sys/class/rpmsg/channel_X/rpmsg_eptY/.

- Multiple processes can use RPMsg to communicate with RTOS environment at the “MCU” side
- Patch proposed to the community, not upstreamed yet.
- Python / C wrappers done for ease of use



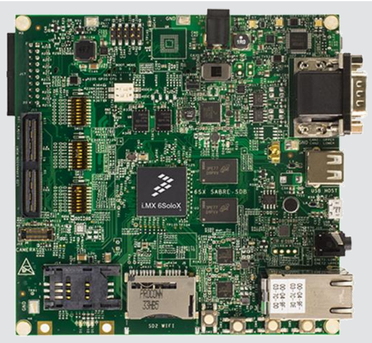


PLATFORMS



LPCXpresso5411x Board

- Cortex-M4 to Cortex-M0+
- Microcontroller platform:
- 256kB flash, 192kB SRAM
- Components:
  eRPC, RPMsg-Lite



i.MX6SX SABRE SDB Board

- Cortex-A9 to Cortex-M4
- CM4 can run from on chip SRAM
- CA9 running Yocto Linux
- Components:
  eRPC, RPMsg-Lite, RPMsg sysfs

RESULTS & BENEFITS

The solution can serve as a base enablement for virtualization of services provided and consumed by different cores in multi-core and multi-processor applications. Secondary core managed by the MCSDK can run following types of applications:

- Communication stacks (USB, Thread, BLE, Zigbee)
- Sensor aggregation/fusion apps.
- Encryption algorithms
- Virtual peripherals

REFERENCES

eRPC on GitHub:
https://github.com/EmbeddedRPC/erpc
RPMsg-Lite on GitHub:
https://github.com/NXPmicro/rpmsg-lite
OpenAMP on GitHub:
https://github.com/OpenAMP
Kinetis SDK at NXP Kinetis Expert webpage:
https://kex.nxp.com
Lockless Shared Memory Based Multicore Communication Protocol IEEE article
http://www.radio.feec.vutbr.cz/ieee/userfiles/downloads/archive/2016-Blansko/Sbornik_Blansko_2016.pdf

ACKNOWLEDGEMENT

Research leading to these results has received funding from the EU ARTEMIS Joint Undertaking under grant agreement no 621429 (project EMC^2) and from the respective national funding authorities.