ST Restriced – Subject to Non-Disclosure Agreement – Do not copy

Coprocessor management overview

Stable: 16.02.2021 - 17:28 / Revision: 16.02.2021 - 17:01

ST Restriced – Subject to Non-Disclosure Agreement – Do not copy

Stable: 18.02.2021 - 17:29 / Revision: 18.02.2021 - 17:31

This article provides an overview of the management of the heterogeneous asymmetric architecture implemented in the STM32 MPU microprocessor family. It provides information on mechanisms put in place to help developers to design software in the multiprocessor system.

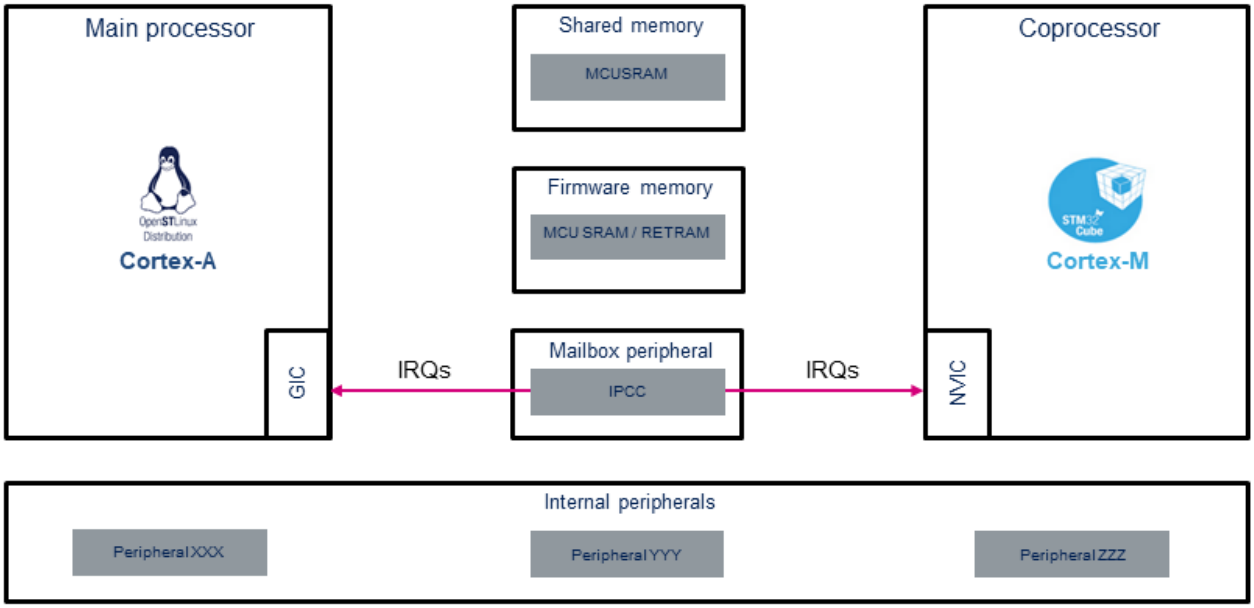
Contents

ST Restriced – Subject to Non-Disclosure Agreement – Do not copy

1 System overview

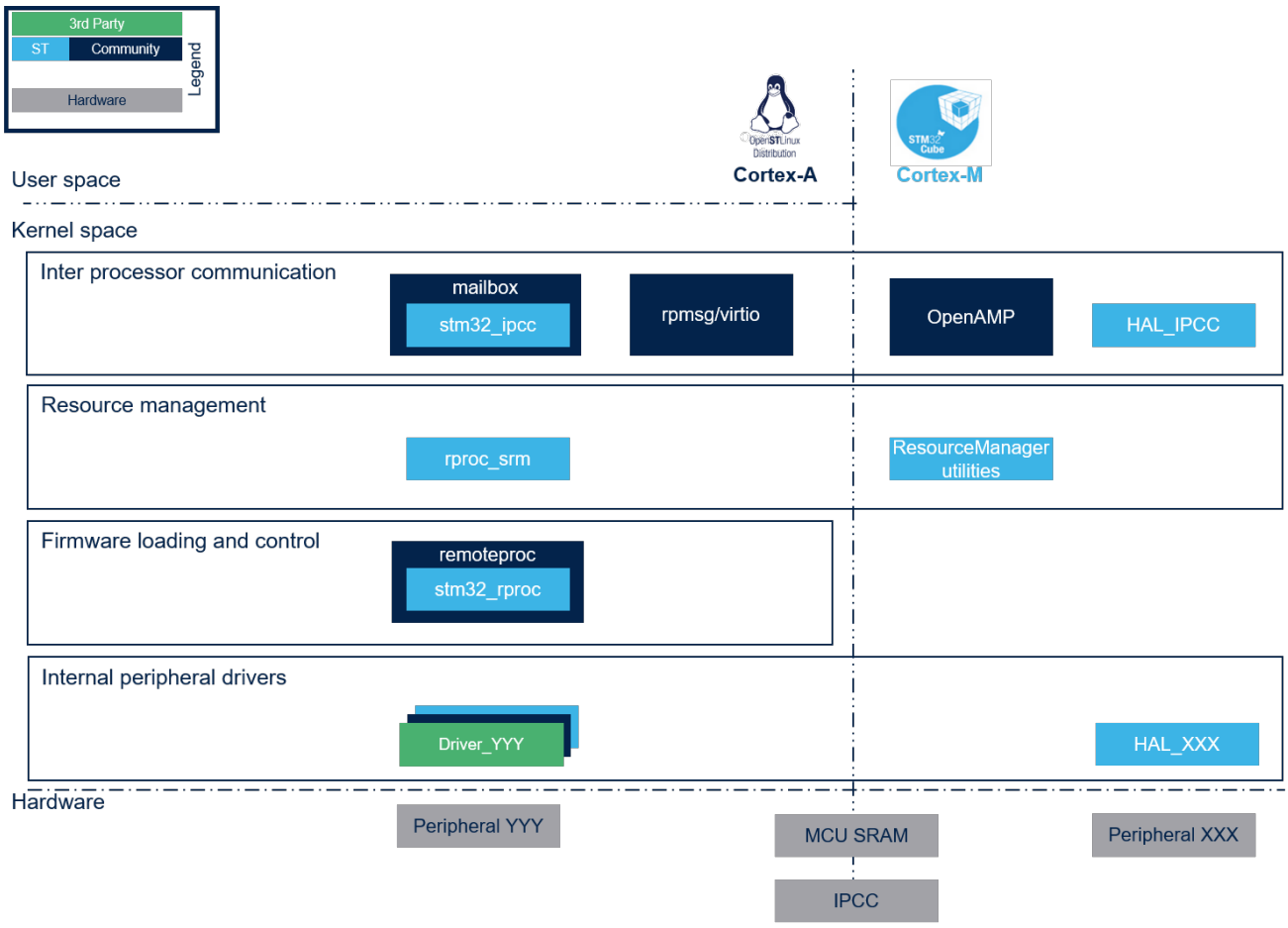
The STM32 MPU multiprocessor system allows to run independent firmwares on each CPU core. The below subsystems are involved in the management of the coexistence of the two CPU subsystems:

- A Arm®Cortex®-A acting as main processor and optimized to run the Linux® based OS.
- A Arm®Cortex®-M coprocessor which can run the RTOS optimized for microcontrollers or a bare-metal application.
- Internal memory regions with access granted for both the master and/or the slave processors:
  - To load and execute coprocessor firmware and define static common structures.
  - To share buffers for inter processing communication through a messaging infrastructure.
- An inter-processor communication controller peripheral allowing a signaling system by a dedicated mailbox.
- Internal peripheral resources that can be assigned to the master or the slave processor.



2 Functional features and design

In order to manage the coprocessor system, a list of services is proposed relying on the open-source **RemoteProc** and **RPMsg** frameworks. These frameworks are introduced in chapters below with links to dedicated articles for further explanation.



2.1 Load and control the Cortex-M firmware

The Linux OS integrates the RemoteProc framework that allows to load firmware and control remote processors.

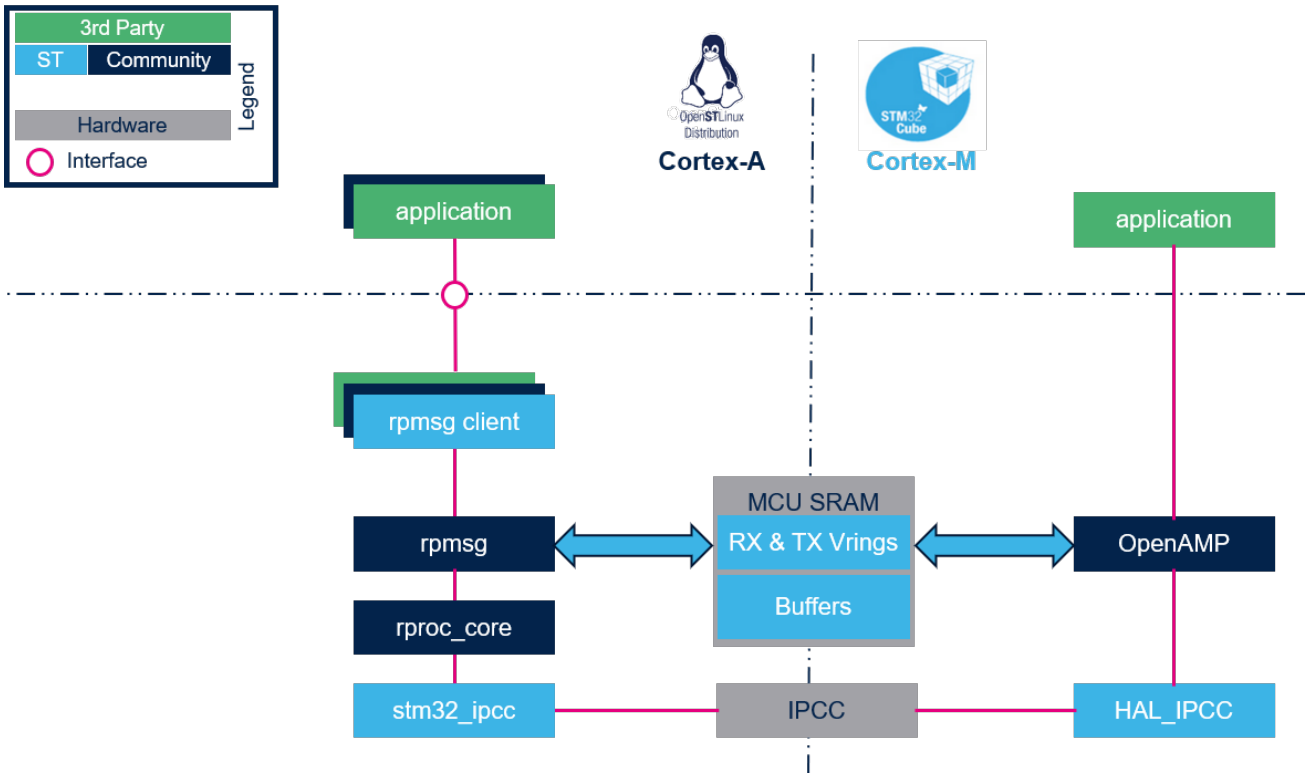
2.2 Resources management (for shared peripheral, clocks, GPIOs...)

The resource manager proposes services to manage common resources and avoid any conflict.

- **Peripheral assignment request**: the mechanism used to ensure that a peripheral is reserved for a processor usage. The principle is that a firmware requests the peripheral before starting to use it, relying on the ETZPC table.
  - On Cortex-A: At boot time, the ETZPC and Linux device tree are configured according to the TF-A® device tree (refer to How to assign an internal peripheral to a runtime context for details).
  - On Cortex-M: the service is implemented by the **Resource manager** utilities.
- **Coprocessor resource configuration set**: services available in the main processor (Cortex-A running Linux) to configure the system resources needed to operate the peripheral on the coprocessor. The service is implemented by **rproc_srm** driver.

2.3 Inter processor communication

Inter processor communication is based on **RPMsg** framework and **Mailbox** mechanisms.



- On Cortex-A:
  - The RemoteProc framework is in charge of enabling the IPC on Linux side, based on information available in the firmware resource table.
  - The RPMsg service is implemented by the RPMsg framework.
  - The Mailbox service is implemented by the stm32_ipcc mailbox driver.
- On Cortex-M:
  - The RPMsg service is implemented by the OpenAMP library .
  - The Mailbox service is implemented by the HAL_IPCC driver.

Microprocessor Unit

Central processing unit

Arm® is a registered trademark of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Cortex®

Linux® is a registered trademark of Linus Torvalds.

Operating System

Real Time Operating System

Remote Processor Messaging

Extended TrustZone Protection Controller

ST Restriced – Subject to Non-Disclosure Agreement – Do not copy

Inter-Processor Communication

Hardware Abstraction Layer

Inter-Processor Communication Controller