Getting started with STM32 MPU devices

Stable: 24.09.2019 - 10:30 / Revision: 24.09.2019 - 10:29

A quality version of this page, accepted on *24 September 2019*, was based off this revision.

Contents

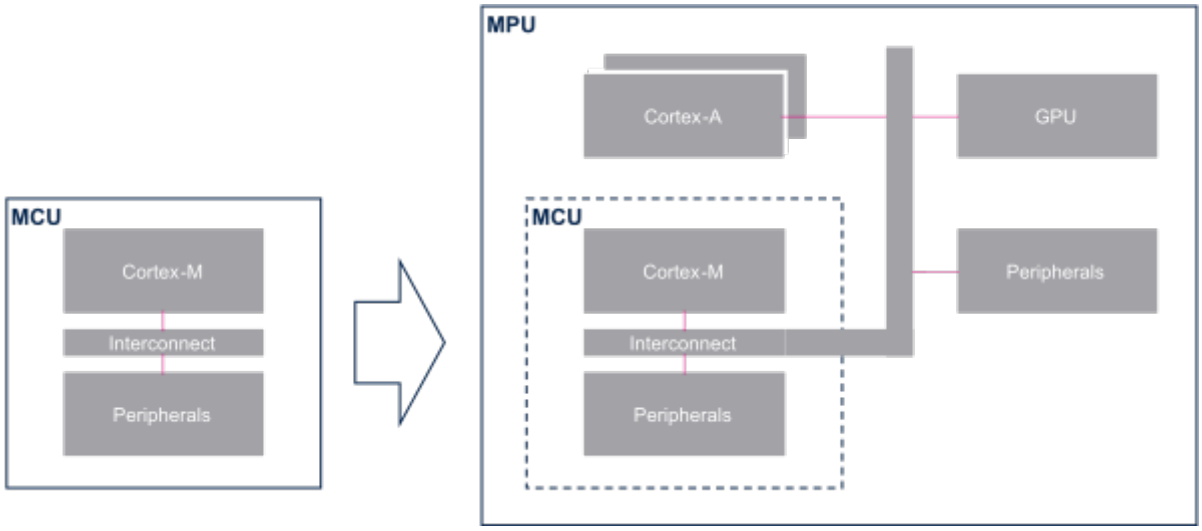
1 Extending the STM32 MCU family to the MPU world

Microcontroller units (MCUs) are built around MMU-less cores such as the Arm Cortex-M, which are very efficient for deterministic operations in a bare metal or real time operating system (RTOS) context. STMicroelectronics STM32 MCUs embed enough SRAM (static RAM) and Flash memory for many applications, and this can be completed with external memories.

Microprocessor units (MPUs) rely on cores such as the Arm Cortex-A, with memory management unit (MMU) to manage virtual memory spaces, opening the door to efficient support of a rich operating system (OS) such as Linux. A fast interconnect makes the bridge between the processing unit, high-bandwidth peripherals, external memories (RAM and NVM) and, usually, a graphical Processing Unit (GPU).

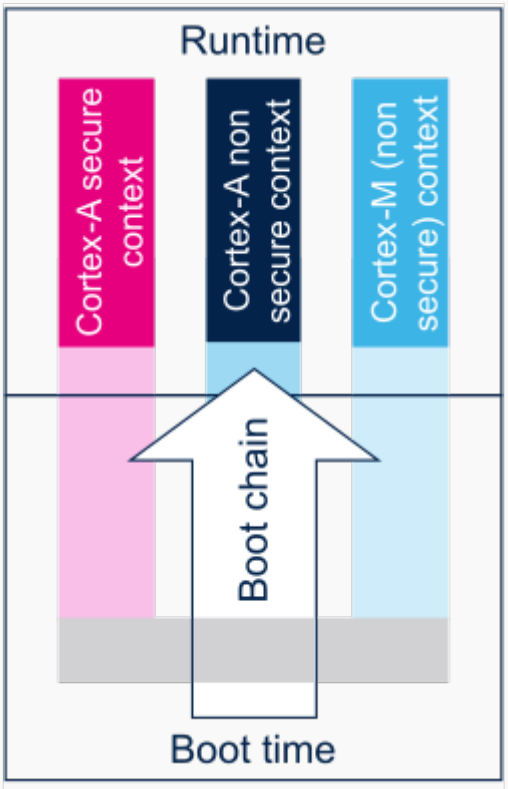
STMicroelectronics has a strong presence in MCU markets with STM32 family [1] and entered the MPU market with a first platform referenced as **STM32MP15**. This platform aims to address multiple market segments such as industrial, consumer, healthcare, home and building automation.

The STMicroelectronics approach for a smooth transition to the MPU world consists of putting both worlds in a single device, seeing the MCU as a subsystem of the MPU: Template:ProductMarketingReviewsComments



2 Multiple-core architecture concepts

As seen above, the MPU is a multiple-core architecture that can interact with a wide number of peripherals. Some **new concepts** need to be introduced for a good understanding of the system: these concepts are explained below and are illustrated in the figure on the right.



Runtime context initialization at boot time

Template:WarningImageMapOverlay

2.1 Hardware execution contexts

Each core can run in a non-secure and - eventually - a secure (Arm Trustzone[2]) mode.
A **hardware execution context** corresponds to a core and security mode.
The three hardware execution contexts available on STM32 MPU devices are:

- Arm Cortex-A secure (Trustzone)
- Arm Cortex-A non secure
- Arm Cortex-M (non-secure)

Each hardware execution context can host different firmware, depending on the platform state. The following contexts can be distinguished:

- the **boot time** context, corresponding to a transitory firmware execution, when the device is booting up
- the **runtime** context, corresponding to an established firmware execution, when the device is up-and-running

2.2 Firmwares executed in the runtime contexts

Each runtime context executes a given piece of **firmware**:

- Arm Cortex-A secure (Trustzone), executes **OP-TEE**[3]

- Arm Cortex-A non secure, executes **Linux**[3]
- Arm Cortex-M (non-secure), executes **STM32Cube**[3]

OP-TEE, Linux and STM32Cube are **STM32MPU Embedded Software**[3] components.

2.3 Peripheral assignment to the runtime contexts

The term **peripheral assignment** is used to identify the action to assign a set of peripherals to a runtime context. This is a user choice that can be realized via STM32CubeMX[4] or manually, in order to properly configure the boot chain[5] and the several pieces of firmware that run on the platform.

Each microprocessor peripheral-overview article shows the assignment capabilities for each peripheral, with a table similar to the example below:

Domain	Peripheral	Runtime allocation				Comment
		Instance	**Cortex-A S** (OP-TEE)	**Cortex-A NS** (Linux)	**Cortex-M** (STM32Cube)	
X	Y	YYY1				YYY1 can be assigned (single choice) to whether Cortex-A non-secure or Cortex-M
		YYY2				YYY2 can only be

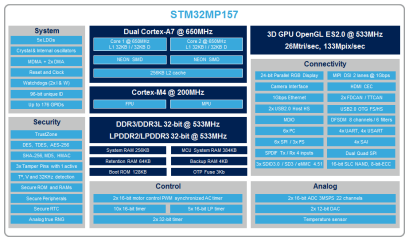
X X	Y Y					assigned to Cortex-A secure
		YYY3				YYY3 is shared accross all contexts: this is typically the case for system peripherals

Refer to How to assign an internal peripheral to a runtime context for detailed instructions.

3 STM32MP1 family microprocessors

What are the main features of an STM32 microprocessor device?
How to program STM32 microprocessor device-internal peripherals?
How to configure internal peripherals for new boards?

Click on the links in the frame below and be guided!



STM32MP15 microprocessor

4 References

- http://www.st.com/en/microcontrollers/stm32-32-bit-arm-cortex-mcus.html
- https://www.arm.com/products/security-on-arm/trustzone
- 3.03.13.23.3 STM32MPU Embedded Software
- STM32CubeMX
- Boot_chains_overview

Template:ToBeReviewedByProductMarketing

Microcontroller Unit (MCUs have internal flash memory and are intended to operate with a minimum amount of external support ICs. They commonly are a self-contained, system-on-chip (SoC) designs.)

Microprocessor Unit

Memory Management Unit. (A hardware device or circuit that supports virtual memory and paging by translating virtual addresses into physical addresses.)

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

Cortex®

Real Time Operating System

Random Access Memory (Early computer memories generally hadserial access. Memories where any given address can be accessed when desired were then called "random access" to distinguish them from the memories where contents can only be accessed in a fixed order. The term is used today for volatile random-acces ssemiconductor memories.)

Flash memories combine high density and cost effectiveness of EPROMs with the electrical erasability of EEPROMs. For this reason, the Flash memory market is one of the most exciting areas of the semiconductor industry today and new applications requiring in system reprogramming, such as cellular telephones, automotive engine management systems, hard disk drives, PC BIOS software for Plug & Play, digital TV, set top boxes, fax and other modems, PC cards and multimedia CD-ROMs, offer the prospect of very high volume demand.

Operating System

Linux® is a registered trademark of Linus Torvalds.

Non Volatile Memory, like a flash memory

Graphics Processing Units

Open Portable Trusted Execution Environment