Boot chains overview

Contents

1. Boot chains overview 3
2. Boot chain overview 3

Redirect to:

- Boot chain overview

Stable: 10.11.2020 - 07:59 / Revision: 10.11.2020 - 07:58

A quality version of this page, approved on *10 November 2020*, was based off this revision.

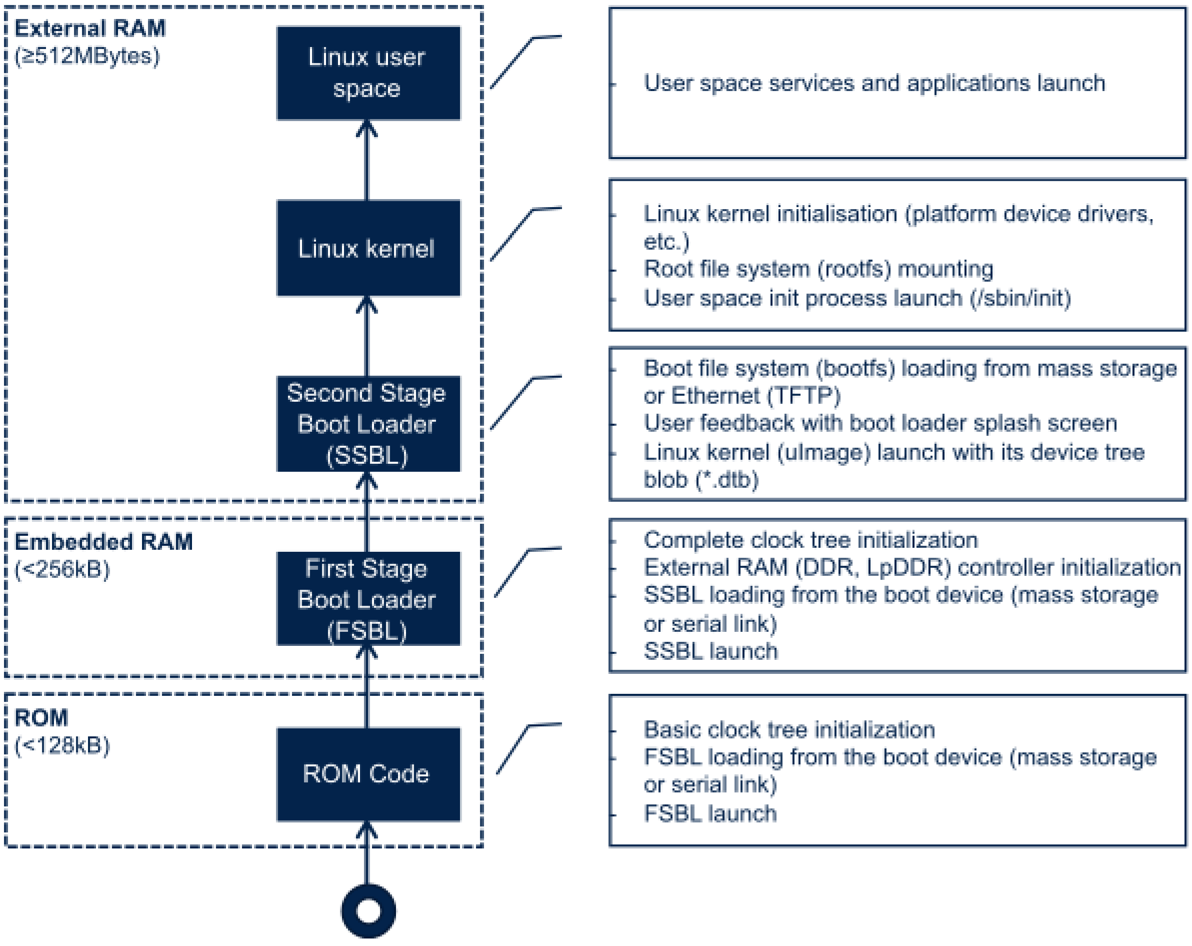
Contents

1 Generic boot sequence 4
1.1 Linux start-up 4
1.1.1 ROM code 4
1.1.2 First stage boot loader (FSBL) 4
1.1.3 Second-stage boot loader (SSBL) 5
1.1.4 Linux kernel space 5
1.1.5 Linux user space 5
1.2 Other services start-up 5
2 STM32MP boot sequence 6
2.1 Diagram frames and legend 6
2.2 STM32MP15 boot chain 6
2.2.1 Overview 6
2.2.2 ROM code 7
2.2.3 First stage boot loader (FSBL) 7
2.2.4 Second stage boot loader (SSBL) 7
2.2.5 Linux 7
2.2.6 Secure OS / Secure Monitor 8
2.2.7 Coprocessor firmware 8

1 Generic boot sequence

1.1 Linux start-up

Starting Linux® on a processor is done in several steps that progressively initialize the platform peripherals and memories. These steps are explained in the following paragraphs and illustrated by the diagram on the right, which also gives typical memory sizes for each stage.


External RAM
(≥512MBytes)
Linux user space
Linux kernel
Second Stage Boot Loader (SSBL)
Embedded RAM
(<256kB)
First Stage Boot Loader (FSBL)
ROM
(<128kB)
ROM Code
User space services and applications launch
Linux kernel initialisation (platform device drivers, etc.)
Root file system (rootfs) mounting
User space init process launch (/sbin/init)
Boot file system (bootfs) loading from mass storage or Ethernet (TFTP)
User feedback with boot loader splash screen
Linux kernel (uImage) launch with its device tree blob (*.dtb)
Complete clock tree initialization
External RAM (DDR, LpDDR) controller initialization
SSBL loading from the boot device (mass storage or serial link)
SSBL launch
Basic clock tree initialization
FSBL loading from the boot device (mass storage or serial link)
FSBL launch

1.1.1 ROM code

The ROM code is a piece of software that takes its name from the read only memory (ROM) where it is stored. It fits in a few tens of Kbytes and maps its data in embedded RAM. It is the first code executed by the processor, and it embeds all the logic needed to select the boot device (serial link or Flash) from which the first-stage boot loader (FSBL) is loaded to the embedded RAM.
Most products require to trust the application that is running on the device and the ROM code is the first link in the chain of trust that must be established across all started components: this trust is established by authenticating the FSBL before starting it. In turn, the FSBL and each following component will authenticate the next one, up to a level defined by the product manufacturer.

1.1.2 First stage boot loader (FSBL)

Among other things, the first stage boot loader (FSBL) initializes (part of) the clock tree and the external RAM controller. Finally, the FSBL loads the second-stage boot loader (SSBL) into the external RAM and jumps to it.

The Trusted Firmware-A (TF-A) and U-Boot secondary program loader (U-Boot SPL) are two possible FSBLs.

1.1.3 Second-stage boot loader (SSBL)

The second-stage boot loader (SSBL) runs in a wide RAM so it can implement complex features (USB, Ethernet, display, and so on), that are very useful to make Linux kernel loading more flexible (from a Flash device, a network, and so on), and user-friendly (by showing a splash screen to the user). U-Boot is commonly used as a Linux bootloader in embedded systems.

1.1.4 Linux kernel space

The Linux kernel is started in the external memory and it initializes all the peripheral drivers that are needed on the platform.

1.1.5 Linux user space

Finally, the Linux kernel hands control to the user space starting the init process that runs all initialization actions described in the root file system (rootfs), including the application framework that exposes the user interface (UI) to the user.

1.2 Other services start-up

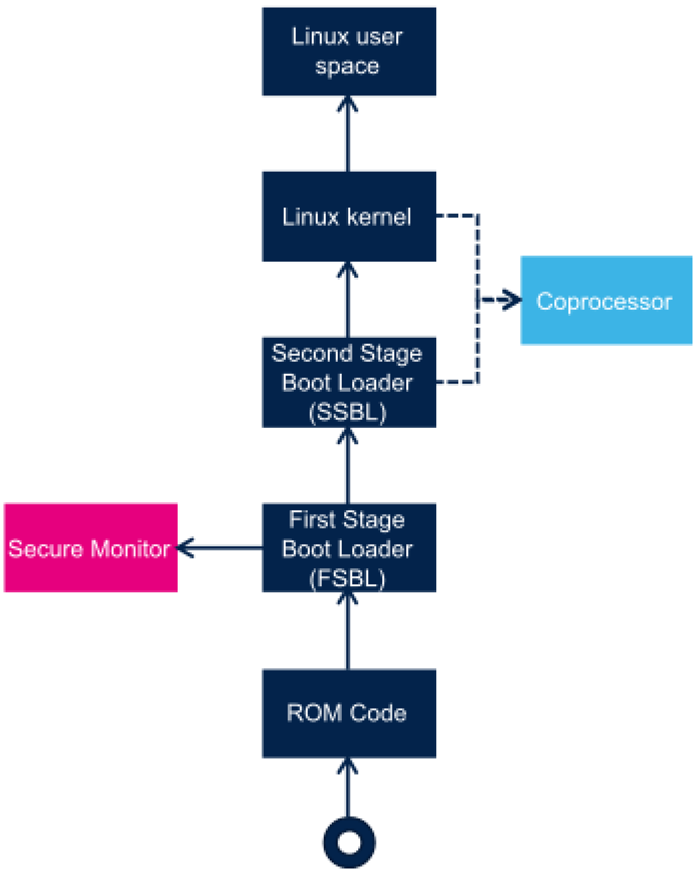
In addition to Linux startup, the boot chain also installs the secure monitor and may support coprocessor firmware loading.

For instance, for the STM32MP15, the boot chain starts:

- the secure monitor, supported by the Arm®Cortex®-A secure context (TrustZone). Examples of use of a secure monitor are: user authentication, key storage, and tampering management.
- the coprocessor firmware, running on the Arm Cortex-M core. This can be used to offload real-time or low-power services.

The dotted lines in the diagram on the right mean that:

- the coprocessor can be started by the **second stage boot loader (SSBL)**, known as “**early boot**”, or **Linux kernel**


Linux user space
Linux kernel
Coprocessor
Second Stage Boot Loader (SSBL)
Secure Monitor
First Stage Boot Loader (FSBL)
ROM Code

2 STM32MP boot sequence

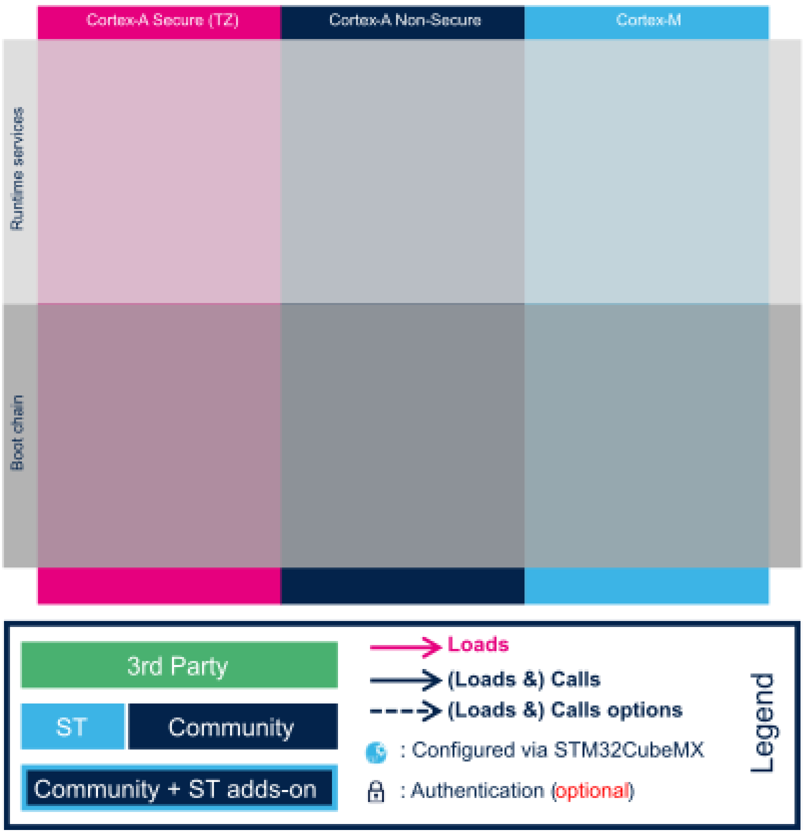
2.1 Diagram frames and legend

The hardware execution contexts are shown with vertical frames in the boot diagrams:

- the Arm Cortex-A secure context, in pink
- the Arm Cortex-A non-secure context, in dark blue
- the Arm Cortex-M context, in light blue

The horizontal frame in:

- the bottom part shows the **boot chain**
- the top part shows the **runtime** services, that are installed by the **boot chain**



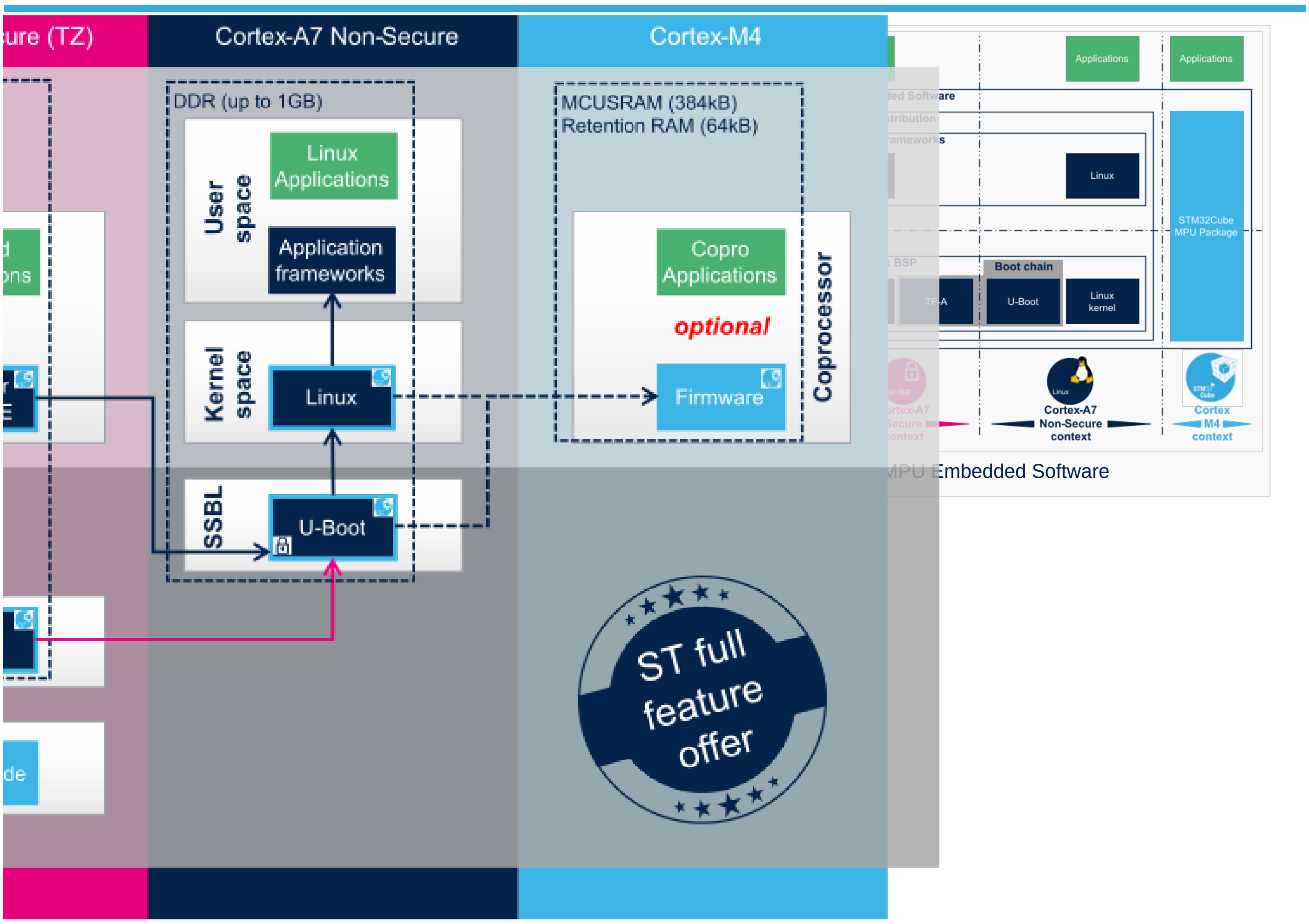
The legend on the right shows how information about the various components shown in the frames, and which are involved in the boot process, is highlighted:

- The box **color** shows the component source code origin
- The **arrows** show the loading and calling actions between the components
- The **Cube** logo is used on the top right corner of components that can be configured via STM32CubeMX
- The **lock** show the components that can be authenticated during the boot process

2.2 STM32MP15 boot chain

2.2.1 Overview

STM32MP15 boot chain uses Trusted Firmware-A (TF-A) as the FSBL in order to fulfill all the requirements for security-sensitive customers, and it uses U-Boot as the SSBL. Note that the authentication is optional with this boot chain, so it can run on any STM32MP15 device security variant (that is, with or without the Secure boot).
Refer to the security overview for an introduction of the secure features available on STM32MP15, from the secure boot up to trusted applications execution.



Note:

- The STM32MP15 coprocessor can be started at the SSBL level by the U-Boot early boot feature or, later, by the Linux remoteproc framework, depending on the application startup time-targets.

2.2.2 ROM code

The ROM code starts the processor in secure mode. It supports the FSBL authentication and offers authentication services to the FSBL.

2.2.3 First stage boot loader (FSBL)

The FSBL is executed from the SYSRAM.
Among other things, this boot loader initializes (part of) the clock tree and the DDR controller. Finally, the FSBL loads the second-stage boot loader (SSBL) into the DDR external RAM and jumps to it.
Trusted Firmware-A (TF-A) is the FSBL used on the STM32MP15.

2.2.4 Second stage boot loader (SSBL)

U-Boot is commonly used as a bootloader in embedded software and it is the one used on STM32MP15.

2.2.5 Linux

Linux$^{®}$OS is loaded in DDR by U-Boot and executed in the non-secure context.

2.2.6 Secure OS / Secure Monitor

The Cortex-A7 secure world can implement a minimal secure monitor (from TF-A or U-Boot) or a real secure OS, such as OP-TEE.

2.2.7 Coprocessor firmware

The coprocessor STM32Cube firmware can be started at the SSBL level by U-Boot with the remoteproc feature (rproc command) or, later, by Linux remoteproc framework, depending on the application startup time-targets.

Linux® is a registered trademark of Linus Torvalds.

Read Only Memory

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 memory shortened to gain space in titles, tables and block diagrams

First Stage Boot Loader

Second Stage Boot Loader

Das U-Boot -- the Universal Boot Loader (see U-Boot_overview)

Secondary Program Loader, *Also known as **U-Boot SPL***

User Interface

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

Cortex®

TrustZone®
Arm® and TrustZone® are registered trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere.

Doubledata rate (memory domain)

Operating System