# Introduction to Bootloader and Secure Boot in Embedded System

Table of Contents

[1. Introduction 3](#_Toc120368752)

[2. Boot ROM 3](#_Toc120368753)

[3. Bootloader in Embedded System 4](#_Toc120368754)

[4. Need of Bootloader in Embedded System 6](#_Toc120368755)

[1. Firmware update 6](#_Toc120368756)

[2. Security 6](#_Toc120368757)

[5. Secure Boot 7](#_Toc120368758)

[6. References 7](#_Toc120368761)

# Introduction

This document is intended to describe about the bootloader and secure boot in Embedded System

# Boot ROM

Boot Rom is a small piece of mask ROM or write-protected flash embedded inside the processor chip. It contains the very first code which is executed by the processor on power-on or reset. Depending on the configuration of some strap pins or internal fuses it may decide from where to load the next part of the code to be executed and how or whether to verify it for correctness or validity.

To provide more flexibility in booting, many processors include a multi-Kbyte “Boot ROM” on chip that includes code that the processor vendor develops and burns into the ROM. As we'll see, the ROM code can perform many different functions.

# Bootloader in Embedded System

Bootloader is responsible for finding and loading the final OS or firmware which is supposed to run on the chip.

The bootloader is a very important component in any operating system. A bootloader, also known as a boot program or bootstrap loader. It is special operating system software that loads into the working memory of a computer after start-up. So, you must be clear that this is also software like an application. The term bootloader is a shortened form of the words “bootstrap loader”.

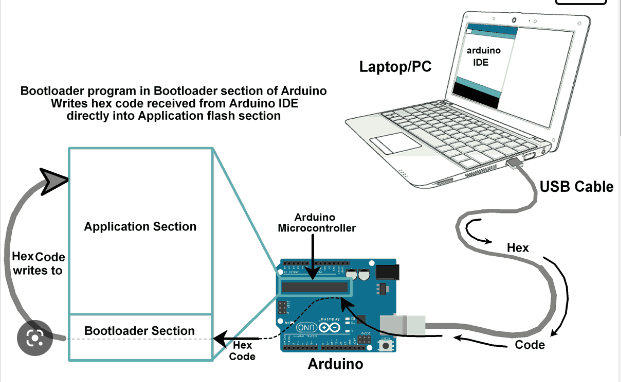


Figure 1: Booting process

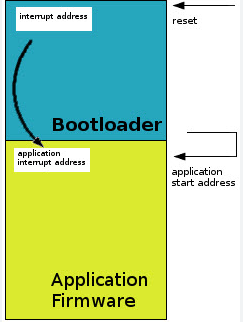


Figure 2: Loading application

Like a normal OS, the bootloader in a microcontroller also serves the same purpose. This is the first piece of code that runs when you press the reset button if you have a bootloader. If you don’t have a bootloader, then directly an application will start running.

If you have a bootloader, then before the application, this bootloader starts running and does some process. Once it is done with the operations, the bootloader job is done. So, it gives control to the application. Now, the application does its job based on our product or project.

**Note:**

A **bootloader**is an application whose primary objective is to upgrade/modify **system software**without the intervention of specialized firmware upgrade tools.

# Need of Bootloader in Embedded System

## Firmware update

Only application is enough. But when you are planning to sell your products to the customers, what will do if you want to update the application/firmware in the device that you sold already?

Every time go to the field and connect the JTAG/J-LINK and flash the firmware or application? It is not possible, right? So, If you have your bootloader, then you don’t need to worry about that. You can update the firmware or application without connecting any debugging

## Security

When you have the product which has to be secured, then what will you do when someone overwrites the application or firmware with their customized firmware to hack your product? How do you find it? In this case, we can use the bootloader to check whether the firmware is valid or not. If it is valid, then only we give control to the firmware or application.

# Secure Boot

# Secure Boot is a process that ensures only authenticated software runs on the device and it is achieved by verifying digital signatures of the software before executing that code. The hardware is pre-configured to authenticate code using trusted security credentials

# To achieve Secure Boot, support from processor/SoC is required. In precise, it assures that the bootloader and OS software is the deliberate manufacturer version and hasn’t been tampered with by any malicious party or process.

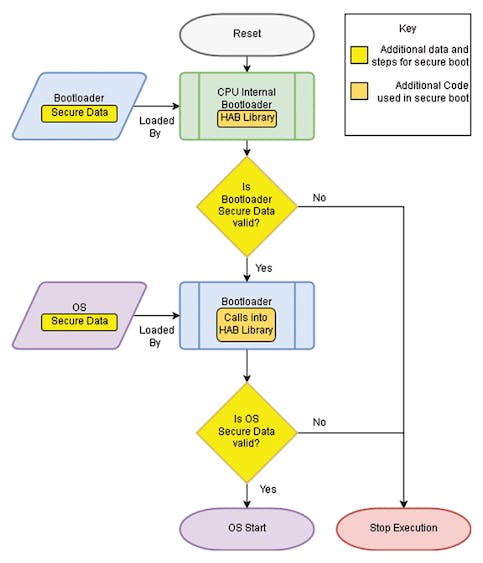


Figure 3: Secure Boot process

# References

1. <https://www.embedded.com/fundamentals-of-booting-for-embedded-processors/>
2. <https://www.microcontrollertips.com/what-is-an-embedded-bootloader-faq/>
3. <https://embetronicx.com/tutorials/microcontrollers/stm32/bootloader/bootloader-basics/>
4. <https://embetronicx.com/tutorials/microcontrollers/stm32/reset-sequence-in-arm-cortex-m4/>
5. <https://www.emergingdefense.com/blog/2019/5/20/bootrom-versus-bootloader>

# 