MSE | MASTER OF SCIENCE IN ENGINEERING

TSM_AdvEmbSof

Bootloader

Serge Ayer | 04.12.2023 | Cours MSE

Good enough, soon enough

- How do we make software correct enough, without going bankrupt ?!
  - Follow a good plan
  - Develop and test efficiently
    - Perform design and code reviews
    - Developing without testing is not efficient
  - Everything is iterative:
    - Be able to improve the development process
    - Be able to improve the software

What is a good plan?

- Start with customer requirements
- Design the building blocks of the system
  - What are the system tasks and modules?
- Add missing requirements and constraints
- Apply a good development process
  - Integrate testing from the beginning
  - Plan unit and integration testing, preferably automated
  - Apply continuous integration and delivery (CI/CD)

Every software version has a lifecycle

- Everything is iterative !
- Requirements will evolve over time:
  - Design needs to be adapted
  - Test plan needs to be adapted
- Enhancements needs to be developed and deployed
  - Adaptations (new requirements)
  - Corrections (including preventive maintenance)

What about deploying updates to embedded systems?

- Billions of microcontrollers are shipped and deployed every year
- How to make software enhancements to these microcontrollers:
  - Cannot return the equipment to the manufacturer !
    - Would you return your computer to its manufacturer for updating it ?
  - While the update of some systems may be assisted by the users, it is often not the case on embedded systems
  - Very often, updates need to be deployed remotely
- Deploying updates on microcontrollers is done through a bootloader application

What is a bootloader?

- It is an application (software)
- Primary purpose: allow a software/firmware to be updated
- Without the use of specialized hardware
  - No JTAG programmer !
- Can use different protocols for receiving the software update

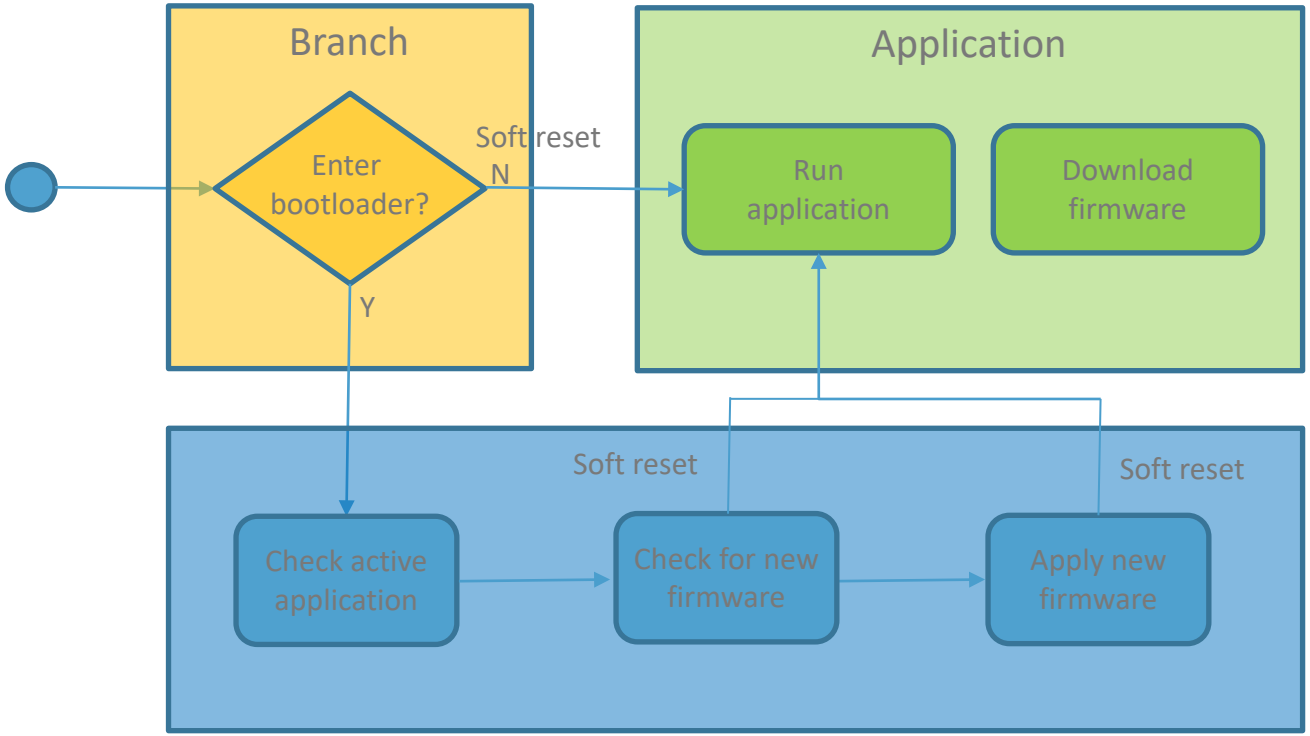
What does it take to develop a bootloader?

- Full understanding of
  - How the microcontroller works
  - How the memory is organized
  - How the Flash memory is partitioned and can be written/erased

Bootloader requirements

- Ability to switch or select the operating mode (application or bootloader)
  - How to enter bootloader mode?
- Communication interface (USB, network, …)
- Format of the update (hex, bin, …)
- Flash/EEPROM requirements (erase, write, read, map/location)
- Corruption protection (checksum)
- Security (protecting the bootloader and the application)

Bootloader process



Bootloader process

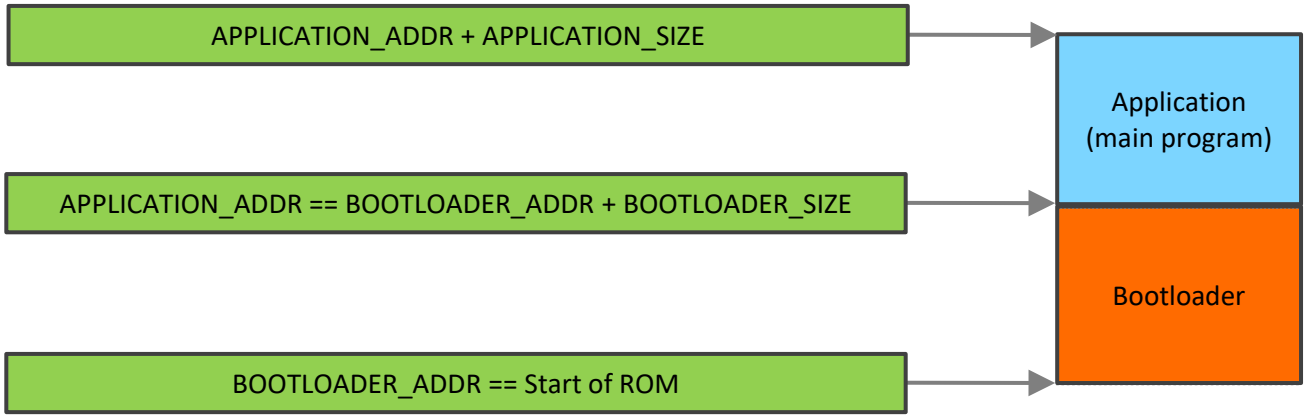
- The process can come in many different flavors
  - Because there are many different scenarios
- Branching code
  - In most cases, the bootloader is always executed for performing some basic system functions (e.g. system integrity check).
  - The branching code is thus included in the bootloader.
  - The branching code makes the decision as whether the bootloader checks for system integrity and for new firmware versions. It also makes the decision of installing a new firmware.
- Application code
  - Executed after the branching code and after integrity checks
  - May host a task for downloading a new firmware – this task could also be performed by the bootloader itself

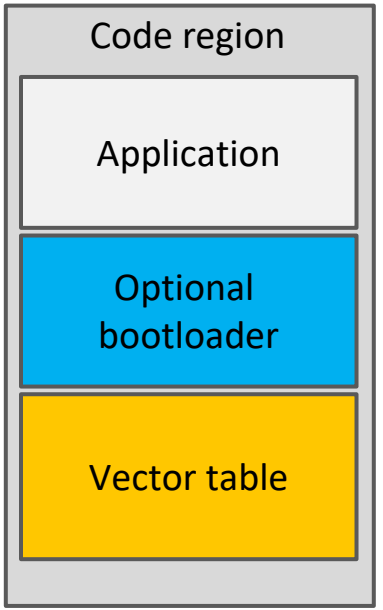
Bootloader application

- The bootloader is not different from a standard application
- It has the capability of erasing and programing a new application in its place by supporting the following commands
  - Erase the flash
  - Write the flash
  - Exit / Restart (reboot while jumping to the application code).
- Depending on the scenario, it may need to access peripherals for carrying out the bootloading functions
- Downloading a new firmware:
  - May happen anytime over a network or based on a specific event (e.g. inserting a SD card or connecting over a serial port).
  - Is usually done as a task in the main application (in particular if the downloads happens through a network).
  - May also be part of the bootloader application.

Memory model with a bootloader

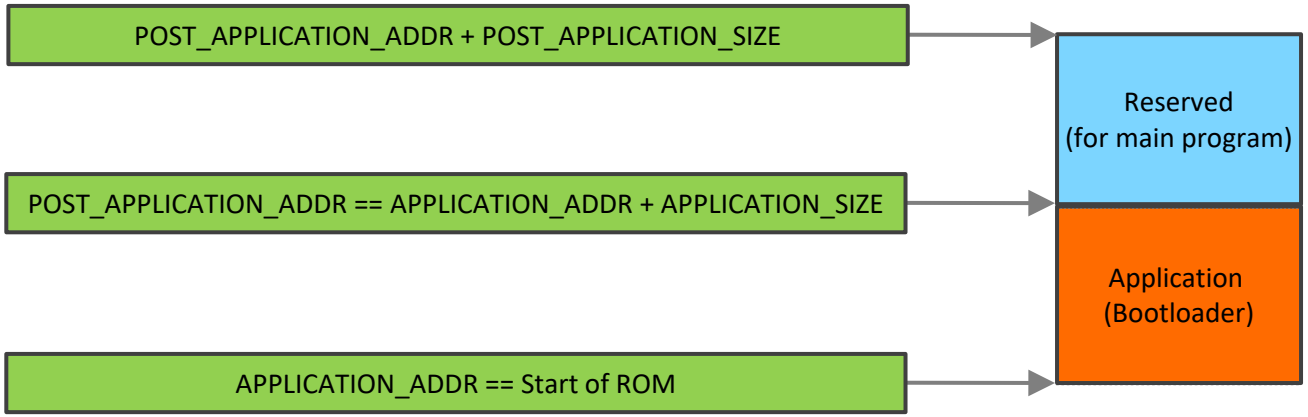
- Mbed OS tools supports the following memory model
- When building the **main application**, it defines symbols that can be used by the main application

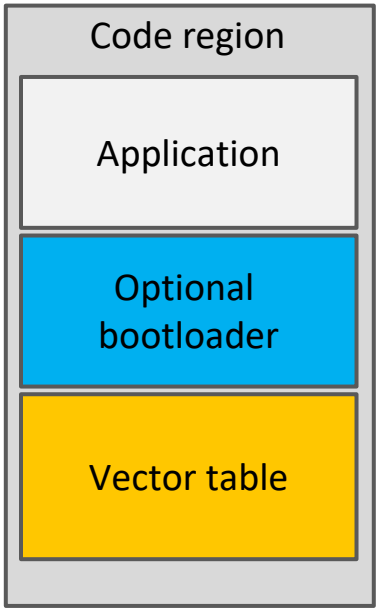




Memory model with a bootloader

- Mbed OS tools supports the following memory model
- When building the **bootloader application**, it defines symbols that can be used by the bootloader application





Codelabs

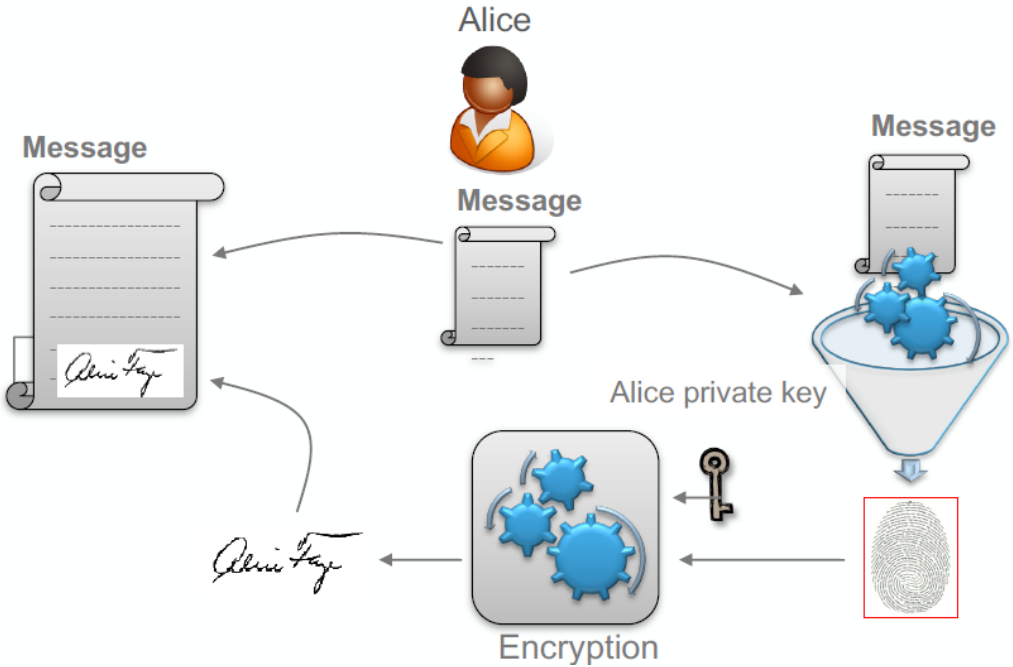
- The bootloader principle

  The bootloader principle
- Host a bootloader into your BikeComputer program

  Modifying the BikeComputer program
- Creating your first Booloader Application

  Your first Bootloader

Checking application integrity

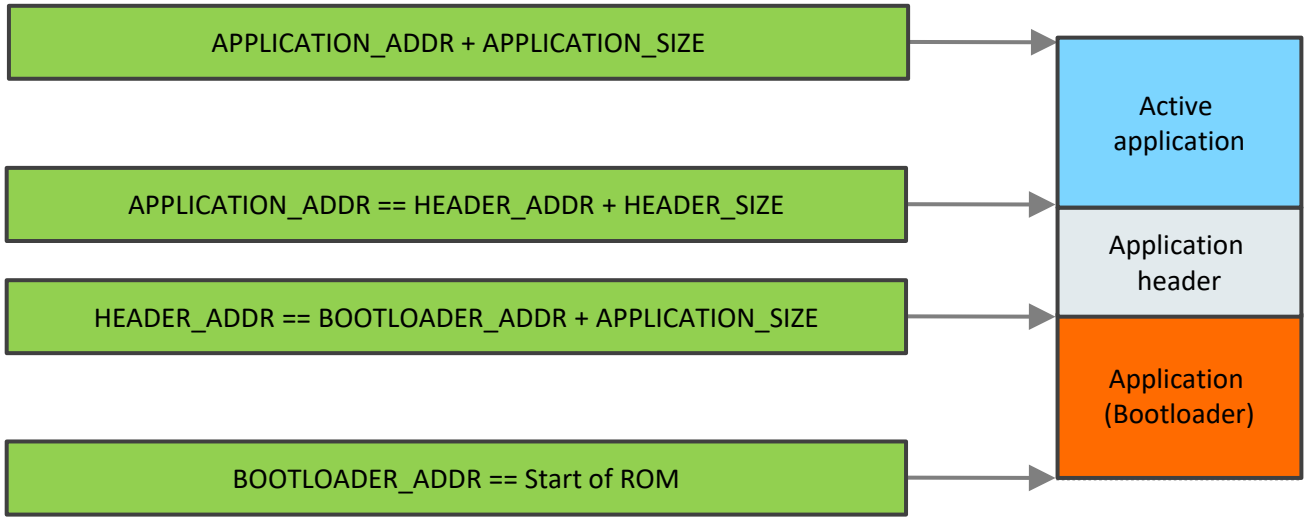


Adding metadata application header

- Metadata created and added at build time, including
  - Header versioning
  - Firmware versioning and size
  - Firmware hash (signature, not encrypted)
- Metadata used by the bootloader
  - For checking application integrity
  - For installing candidate applications

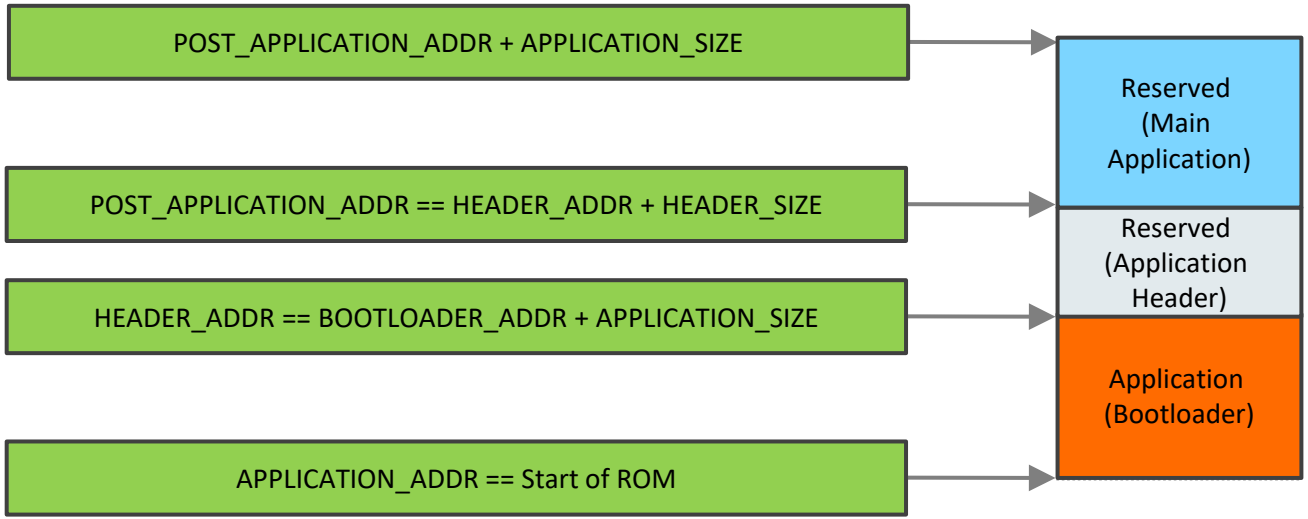
Memory model with application header

- Mbed OS tools supports the following memory model
- When building the **main application**, it defines symbols that can be used by the main application



Memory model with application header

- Mbed OS tools supports the following memory model
- When building the **bootloader application**, it defines symbols that can be used by the bootloader application



Codelabs

- NOTE: you may need to copy the links below and paste them in your browser (rather than simply open it by clicking on it).
- Checking the active application integrity

  Checking active application integrity

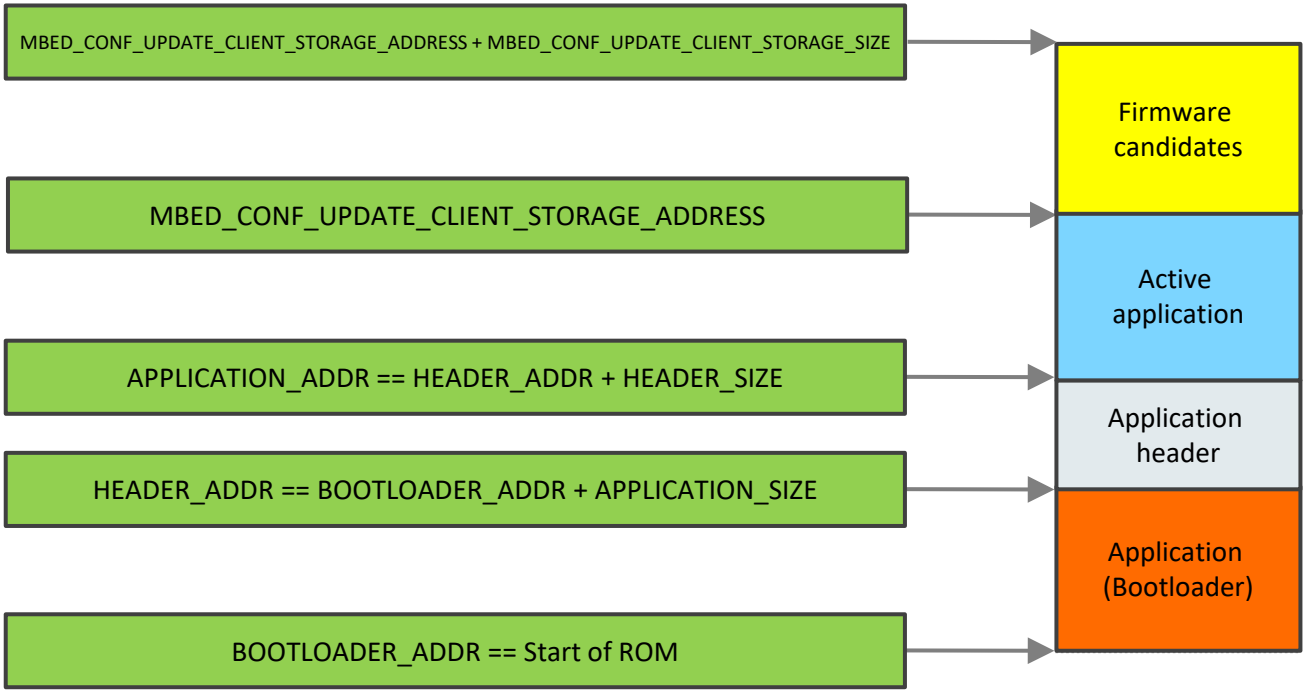
Downloading firmware candidates

- Can be done using various protocols
  - Over an IP network
    - Can be operated from a centralized management system
  - Over a BLE connection
    - Through an IP gateway or through a local BLE connection
  - With physical access
    - For instance, over serial
- In all cases, an update client must be running in the application
  - The client must watch for available updates
  - It must download the firmware candidate and store it at an appropriate location
  - Usually, several candidates can be stored on the target device
  - Does NOT install the firmware – this is the job of the bootloader

Storing the firmware candidates

- Firmware candidates must be stored on non volatile memory.
- Non volatile memory (Flash) may be
  - The Internal Flash
    - If large enough
    - Using the FlashIAP API on Mbed OS)
  - Another Flash Memory
    - Like the 64-Mbit Quad-SPI Flash memory of the DISCO target device
    - Using the QuadSPI API on Mbed OS

Memory model with firmware candidates (Internal Flash)



Codelabs

- NOTE: you may need to copy the links below and paste them in your browser (rather than simply open it by clicking on it).
- Downloading firmware candidates from the BikeComputer program

  Downloading and storing firmware candidates
- Checking for new firmware candidates and installing one

  Installing a Firmware Candidate