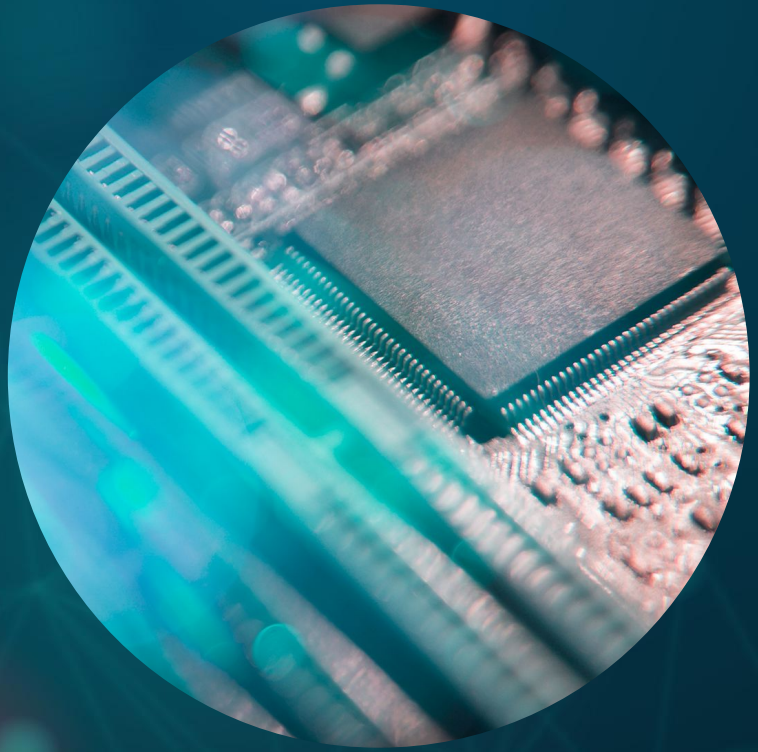
ARM Based Microcontroller

Bootloader

Lecture 21



Advanced RISC Machines

IMT SCHOOL

Flashing 01

Bootloader

Flashing Techniques

The flashing process is the process of updating the content of the flash memory, and since the flash is used to store the program code, the process is also called Programming, Burning. Recall that flash is a non-volatile memory which consists of floating gate mosfets, this kind of mosfets needs high power to change its state, this high power can't be provided by the processor, so that it is a must to use an extra hardware circuit that can provide this high power to access the flash memory, this circuit is called the flash driver. So now the process of programming is obvious, we prepare our executable file then send it to the flash driver, the flash driver in turn will write it to the flash. Depending on the position of the flash interface, different programming techniques are found:

Flashing Techniques

- **Off-Circuit Programming.**

This is the basic old technique, here the flash driver is external outside the microcontroller (MC) and the flash memory programming pins are connected to some (MC) pins to be programmed through. A device called burner is used, this burner is usually a hardware (HW) kit that sometimes has a socket on which the (MC) can be placed, or simply connected to the required pins with jumpers. This burner kit includes the flash driver, it also includes the communication port with the computer i.e., USB port, and uses the computer USB power.

Flashing Techniques

So, the task of the burner is to get the executable file from the computer via its communication port, then it applies the required high power on the flash memory as if it uploads the file to flash memory. When the burner finishes uploading the file, the microcontroller can now be removed and connected to the application circuit.

So, to summarize this technique, to program the microcontroller, the microcontroller shall be removed from its application circuit, then connected to the burner which in turn uploads the program to the flash memory, and when it finishes, the microcontroller can now go back to its application circuit and start working, hence called off circuit programming as it requires removing the microcontroller from its application circuit. But what if the process of removing the microcontroller from its application circuit for reprogramming is not that easy?! For example, in firmware update applications, where the new program shall be uploaded to a lot of microcontrollers inside their machines, this will be a big time and effort consuming task, so that a better technique has arisen. This technique is called In-Circuit Programming.

Flashing Techniques

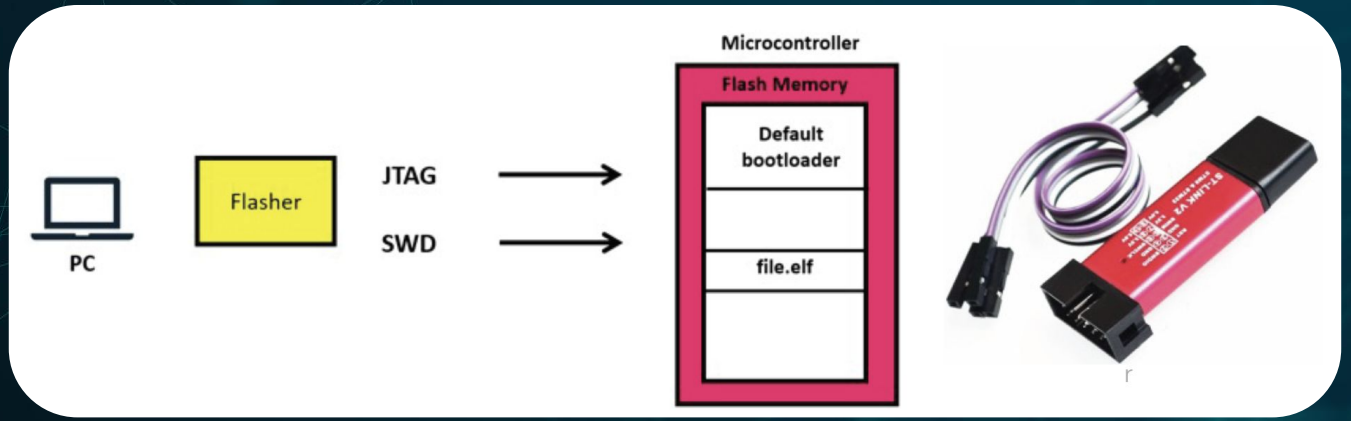
- **In-Circuit Programming.**

It's Also called In System Programming (ISP), here the microcontroller has an internal flash driver, that can generate any necessary programming power from the system's main power supply to provide the required high power for the flash programming which is inside the (MC) also, so the flashing process is done inside the (MC) which means there is no need to remove the (MC) from the system circuit for programming, hence its name.

Of course this feature has provided a lot of advantages as now the (MC) can be programmed while installed in a complete system, rather than requiring the chip to be programmed prior to installing it into the system, hence it allows firmware updates to be delivered to the on-chip flash memory of microcontrollers without requiring specialist programming circuitry on the circuit board, which simplifies design work, also it allows manufacturers of electronic devices to integrate programming and testing into a single production phase, and save money.

Flashing Techniques

But how to communicate with this on-chip flash driver I.e., how to send it the required executable file? Firstly, the flash driver is needed to communicate with the external world to get the required application code, this is done by serial communication, so the flash driver manufacturer defines one or more communication protocols through which the flash driver can interface, for our stm32f103 there are 2 protocols that are provided to communicate with the in-circuit flash programmer: JTAG and SWD (Serial Wire Debug).



Flashing Techniques

In our system the executable file will be sent to the in-circuit flash driver through one of these protocols then the flash driver will apply the required power to program the flash memory, but how to send the executable file from the computer using the USB protocol to the flash driver by the SWD? Again, the answer is that there must be a converter that translates the USB protocol to the SWD protocol, and this will be the only function of the external device that will be used during the programming process. This is the technique we use; we used the ST-Link programmer and debugger as shown in the last figure. One end connects into the computer. This allows us to transfer the executable file from the computer to the USBASP. The other end of the ST-LINK normally gets connected to 4 wires, which can then get hooked up easily to the programming pins on the KIT. Note that until this moment there is an extra device is used to implement the flashing process, as we needed a translator to send the executable file from the computer to the in-circuit flash driver, and this device is target-specific which means that changing the microcontroller to another one with different flash driver communication protocol will result in changing the translator device, this will be a big headache if it is required to reprogram a number of different microcontrollers in the same system, i.e., a car with 100 ECUS (electronic control units), this is one of the main reasons why the bootloader is used.

Flashing Techniques

- **Bootloader.**

In App Programming(IAP)

(IAP) allows the user to re-program the flash memory while the application is running. Nevertheless, part of the application has to be been programmed in the flash memory using (ICP) previously.

Steps:

- receive the file.
- Flash it (page by page).

There is two Applications APP1 and File.elf, We need a flash driver then the APP1 must be flashed ICP firstly, now we can use IAP.

Flashing Techniques

Booting (also known as booting up) is the initial set of operations that a computer system performs when electrical power is switched on. Bootloader Anyone who has turned on a computer might be familiar with the boot-up sequence as the computer flashes lines of text on screen before the Windows logo appears, what you are seeing is a bootloader in action, loading essential software to get the minimum running on the processor chip before higher-level software can run. As its name implies it can both “boot” the operating system or whatever application is to be run, and also as a second function provide a “loader” capability for developers.

Bootloader is a small OS, or application, designed to download firmware in MCU’s internal or external Flash Memory).

Why do we need a Bootloader?

- To fix Bugs.
- Updating system with new features.

Flashing Techniques

Where are they stored?

They reside In ROM, or in the Flash Memory of MCU. (write protected). So, you can't destroy them.

Bootloader Requirements:

- Needs Flash Driver (FPEC). "Responsible for program/erase flash".
- Selects a communication Protocol (CAN). "to download programming data into memory".
- Good design.

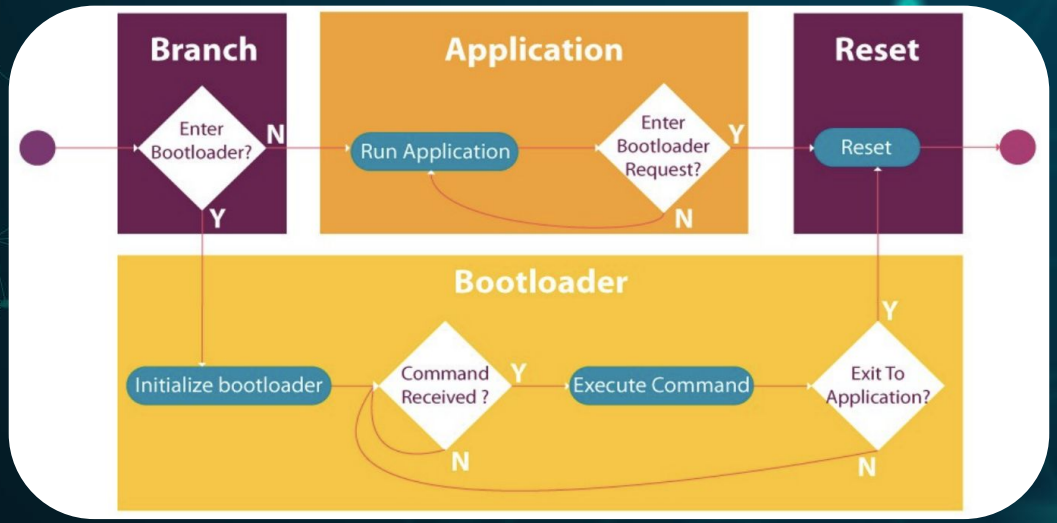
In embedded systems, This task can be simplified as the following:

1. The processor initializes a communication port in the microcontroller to make it ready to receive data.
2. A computer starts sending the application code through this communication port to our microcontroller.
3. the processor reads the data, then sends it to the in-circuit flash driver which in turn writes this data in the flash, eventually the target controller is updated with the code. (Flash the file page by page).

Flashing Techniques

Bootloaders are responsible for:

- Flash new application.
- Run Application.
- Flash new Bootloader.
- Select Application (if we have many applications to select).



FBEC 02

Bootloader

IMT SCHOOL

FBEC

FBEC:- Flash Program and Erase controller

How the flash write / erase operations are done:-

To Program 1 → 0 (become zero physically).
To Erase 0 → 1 (become one physically).

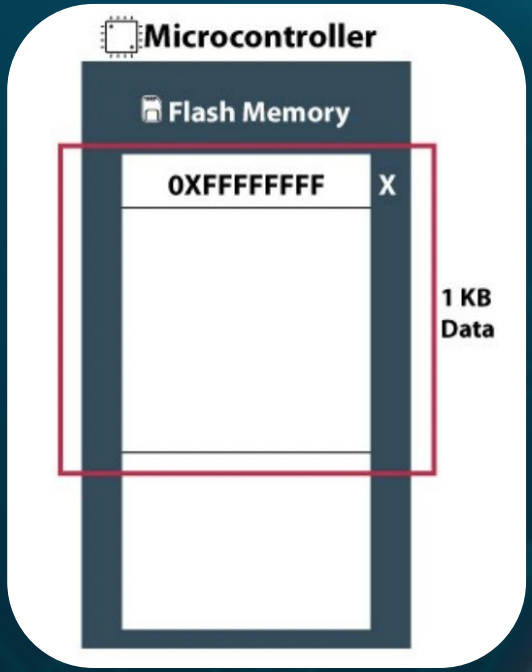
So the eras state of memory is 0XFF, Here we have a question what is the minimum unit that we can program/eras so we have two new concepts appeared.

Sector: which is the minimum unit (size) to eras.

Page: which is the minimum unit (size) to program.

Example 1

If we have a flash fully erased (0xFF) and we need to write 4B (u32 x = 0X01020304). The page is 2B.



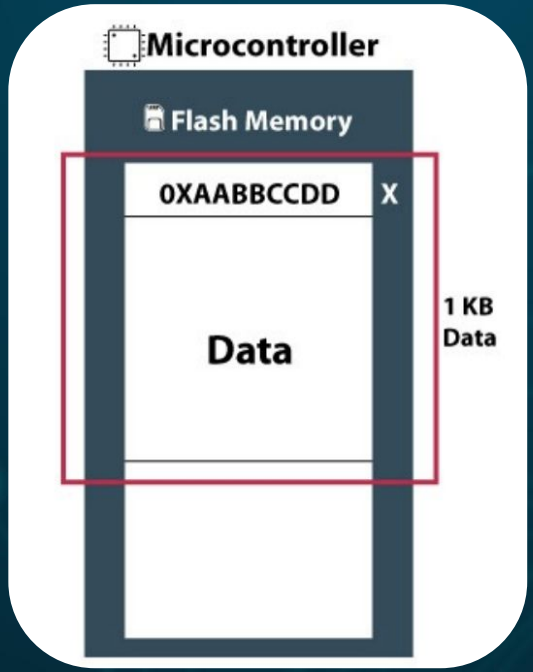
Example 2

If we have a flash sector is 1KB and page is 2B we need to write 4B (u32 x = 0X01020304).

we have two problems here:-

1- x is not erased.

2- 1KB beside x has Data and the sector is 1KB.



Example 2 Solution

The solution is:

1- copy the content of 1KB into the RAM.

2- update x in the RAM.

3- Erase the sector in the Flash (0XFF).

4- program the flash page by page (2B by 2B) until complete 1KB.

Flashing Sequence

If you want to write on the flash follow the sequence:

1- PG Bit = 1 in control register.
2- Write half a word.
3- Wait busy Bit in status register (you can not read when write).
4- Read (optional to verify).

Flashing Sequence

Problems:

Busy Flag Error: When you try to write in a specific location it is not erased.

Key locked: The registers are locked as a safety mechanism to write on, you must write on "Key" firstly. By default all keys are locked after reset.

The sequence to unlock:

Key Register = key1;

Key Register = key2;

As a flash programming manual:

Key1 = 0x45670123.

Key2 = 0xCDEF89AB.

Note: If you write the key wrongly it will be locked to the next reset and the is a lock Bit (LCKK Bit) to lock the Key.

Erasing Sequence

If you want to erase on the flash follow the sequence:

1- PER Bit = 1.
2- ARR in address register = Sector Address.
3- STRT = 1 (Start erase).
4- Wait BSY (Busy Flag).
5- Check 0xFF is readed (optional to verify).

If you want to mass erase (fully memory erase) on the flash follow the sequence:

MER = 1 (Mass erase).
STRT = 1 (Start erase).

Note: Debugging option in Eclipse makes a mass erase.

Bootloader 03

Bootloader

IMT SCHOOL

Bootloader Requirements

1- Ability to select the operating mode (Application or boot-loader).

2- Communication requirements (USB, CAN, I2C, USART, etc).

3- Record parsing requirement (S-Record, hex, intel, toeff, etc).

4- Flash system requirements (erase, write, read, location).

5- EEPROM requirements (partition, erase, read, write).

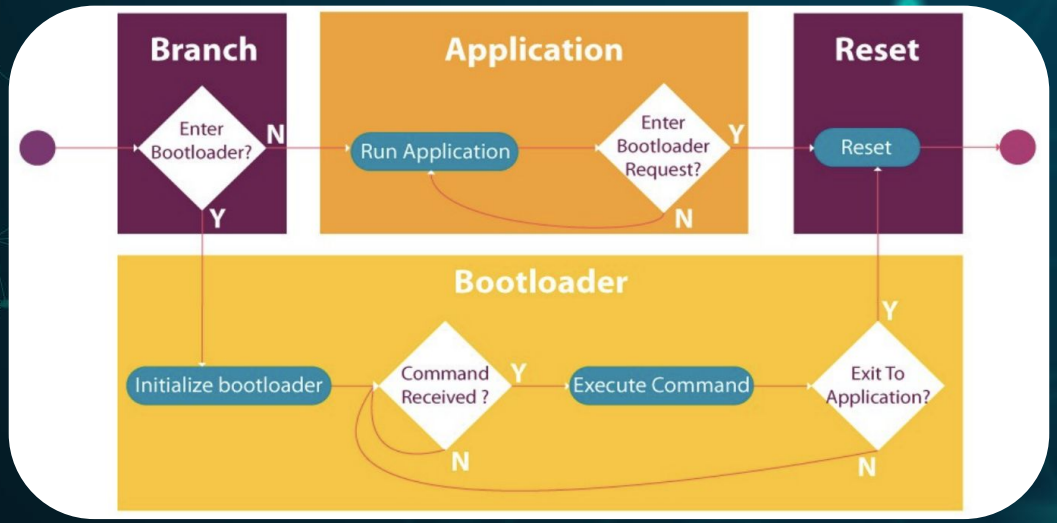
6- Application checksum (verifying the app is not corrupt).

7- Code Security (Protecting the bootloader and the application).

Bootloader Design

Boot Loaders are responsible for :

1- Flash new application.
2- Run Application.
3- Flash new Bootloader.
4- Select Application (if we have many applications to select).



Basic Hardware Initialization

- Mask all interrupts
- Set CPU speed and clock rate (RCC)
- Initialize RAM
- Initialize GPIO
- Disable CPU internal Instruction/Data Cache

Minimum Bootloader Commands

- Erase the flash – Removal of the application image from memory
- Write flash – Addition of a new application image to memory
- Exit / Restart – reboot with the intention of entering the application code

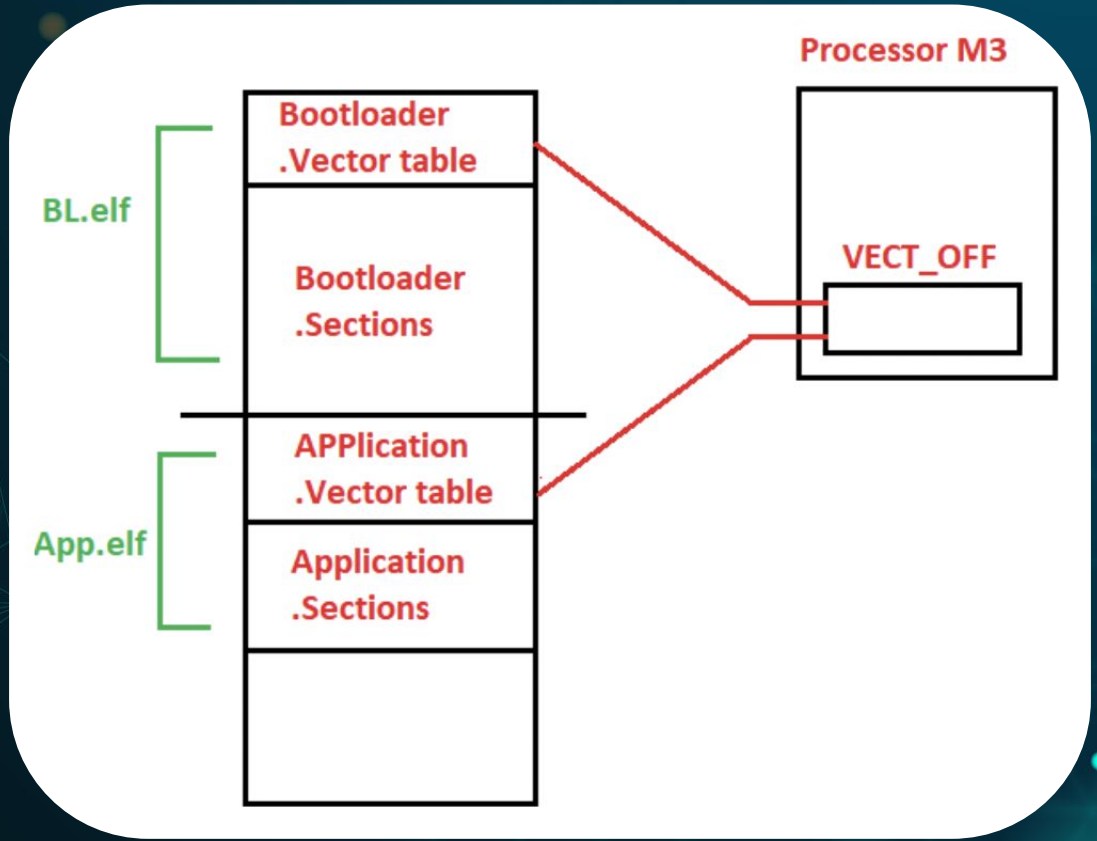
Commands Basic Functions

- Unlock Key – enter a security key to allow flash to be erased and then written
- Erase FLASH – erase configuration data stored on the system
- Write FLASH – write configuration data to the system
- Read FLASH – verify that the application image is correct
- Checksum – calculate an checksum that can be checked for corrupted application
- Lock Key – enter security mode that prevents flash from being written

Bootloader Considerations

- Unlock Key – enter a security key to allow flash to be erased and then written
- Erase FLASH – erase configuration data stored on the system
- Write FLASH – write configuration data to the system
- Read FLASH – verify that the application image is correct
- Checksum – calculate an checksum that can be checked for corrupted application
- Lock Key – enter security mode that prevents flash from being written

Reset and Vectores Reallocation



Bootloader Marker

Value	Mode
0xFFFFFFFF	No APP
0xAAAAAAAA	App1
0xBBBBBBBB	App2

STM32 Is AWESOME

Session LAb

IMT SCHOOL

THANKS!

Do you have any questions?

www.imtschool.com

www.facebook.com/imaketechnologyschool/