**BOOTLOADER IMPLEMENTATION STM32**

1. **Bootloader Basics.**



Figure 1 Bootloader.

1. **What is a bootloader?**

The bootloader is a very important component in any embedded 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/firmware like an application. The term bootloader is a shortened form of the words “bootstrap loader”.

1. **How does the bootloader Work?**

Bootloader will run when device is powered on, or the User presses the “Reset” button.

To do that, Bootloader program will be loaded into start area of Flash memory. When powered on, Program Counter will be set at reset handler of Bootloader Program and start Bootloader’s operation. If no Exception occurs, the Bootloader program will move Program Counter to reset handler of main Application program in another Flash memory address and starts that program.

1. **What is a Bootloader in Embedded System?**

Bootloader is the first piece of code that runs when the user presses the “reset” button. If we don’t have a bootloader, then directly an application will start running.

If we 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.

1. **What is the need for a Bootloader in Microcontroller?**
2. Firmware Update.

With a small project, we are not complicating by writing an extra bootloader. Only application is enough. But when we want to sell our products to the customers, what will do if we want to update the application/firmware in the device that we sold already? Every time go to the field and connect the JTAG/J-LINK and flash the firmware or application? It is not possible. So, if we have a bootloader, then we can update the firmware or application without connecting any debugger or flasher.

1. Security.

When we have the products which must be secured, then we can use the bootloader to check whether the firmware is valid or not. If it is valid, then only we give control permission to the firmware/application.

1. **Firmware Updating Process.**

We can divide Flash memory into 4 Area which are important for our purpose, as a follow:

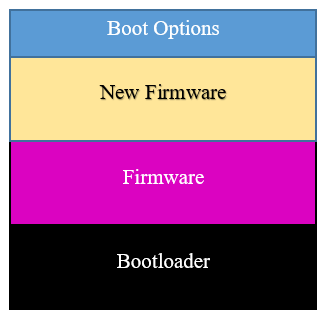


Figure 2 Flash memory.

1. **Boot Options.**

Boot Options is a small Flash memory (4 bytes – 8 bytes), it consists of some flags and some information, based on the Flags, Bootloader program will decide the next action such as running the main program or waiting for Firmware from User if a request occurs.

1. **New Firmware.**

“New Firmware” is a Flash memory area, it will be used as a temporary memory to store binary code of new firmware which is received from User. When an Update Firmware request occurs, Binary code will be divided into some small packets, the packets will be sent to device by some ways such as USB, Ethernet, LAN, RS232, … After that, bootloader will store that data to “New Firmware” area of Flash memory.

If the data transmission – reception and processing is successfully, “Boot Options” Area will set the Flags to notice the Bootloader Program that has a new firmware is ready to update.

1. **Firmware.**

“Firmware” is a Flash memory area. It will be used to store binary code of the main firmware/application. In normal mode, When Device is powered on, Bootloader program run, it will give control permission to firmware/application which stored in “Firmware” area of Flash memory, that program will operate util the device is powered off or an Update firmware Request occurs.

1. **Bootloader.**

“Bootloader” is a Flash memory area. It will be used to store binary code of the Bootloader program. When the device is powered on or after a “reset” signal by hardware (User presses RESET button) or software reset (Watchdog or create reset signal from source code by NVIC), it goes to Bootloader program. Based on the Flags of “Boot Options” Area in Flash memory to decide next action.

Boot Options Flag:

* + - * Firmware Update: Bootloader waiting for Firmware from User and store binary code of the new firmware to “New Firmware” Area in Flash memory.
* Firmware Request: Bootloader copy data from “New Firmware” Area to “Firmware” Area, the main firmware/application will be replaced by new firmware (Update Firmware process is already done).
* Boot Firmware: Bootloader set the PC (Program Counter) to reset handler of main firmware/application and run it.

1. **Data Integrity.**
2. **Software – Firmware verify.**

To ensure the integrity of the data transmitted – received between Software – Bootloader, we can modify a little. The protocol will be the original protocol, just modify the “Write firmware data” frame.

Table 1 Transfer Frame

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FA FB** | **Protocol Version** | **ERROR**  **Number** | **Message**  **Type** | **Data**  **Length** | **DATA** | **FC FD** |

Table 2 Write firmware data frame

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **FA FB** | **Protocol Version** | **ERROR**  **Number** | **Message**  **Type** | **Data**  **Length** | **DATA and CRC** | **FC FD** |

In the “DATA” field, we can divide it into 2 parts, the first part is the firmware data packet and the second part is the CRC, that’s enough. When the Bootloader receives the data packet, it will calculate the CRC and compare it with the CRC at the end of the “Data” field, if the CRC is correct, it will process it to write the firmware data to the Flash memory and return the result with a response packet, if the CRC is incorrect, a response packet with “failed” result will be sent without further processing.

1. **Firmware verify.**

To ensure that the data received from the User (Software) is the same as the data written to the Flash memory, after writing the Firmware data packet, the Bootloader will perform a verification process by reading and comparing the data in the Flash memory with the previously received data, if the data does not match, a "failed" response packet will be sent to the User, if they match, a "success" response packet will be sent.

* **Everything is original, just add 2 – 4 bytes CRC at the end of the “DATA” field.**

1. **Sumary.**
2. **Bootloader operation.**

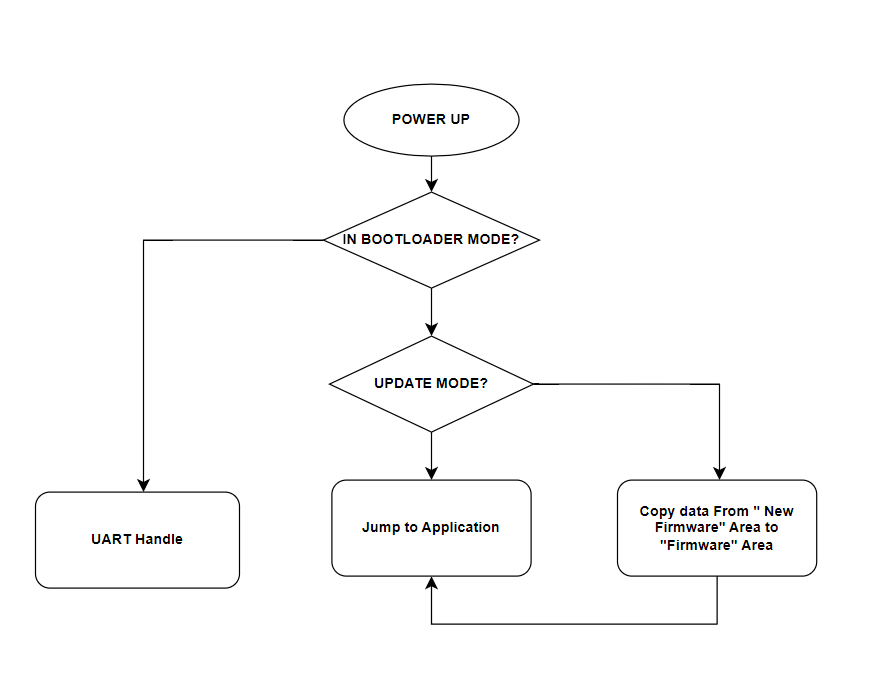


Figure 3 Bootloader Operation

1. **Advantages.**

Compared to the method of writing Binary code directly into the Firmware memory area, this method seems to be "safer". Because if an error occurs during the writing process or exceptional factors occur (such as power failure, battery failure, ...), the Binary code in the Firmware area in Flash memory is still guaranteed to be intact so that it can continue to operate.

1. **Disadvantages.**

Increase the number of read-write operations to the Flash memory area, the Flash memory area is a low-life memory area (about 10,000 write operations), so this method can reduce the life of the flash memory area if the Firmware Update request occurs at low cycles.