Understanding What is a Flash Bootloader

Every Electronic Control Unit (ECU) in an automobile is tasked with executing a specific application. An **Anti-Lock Braking System ECU**, for instance, ensures that brakes don't get locked during braking.
How this works? – the ABS software application ported into the ECU hardware, is able to fetch the vehicle speed as an input and is designed to reduce the brake on the wheels, based on this input.
What role does a Flash Bootloader software play in all this? You may ask!
Well, as the names suggests, Bootloader is the software algorithm that is executed during the booting of the system.
Let's delve deeper to understand why an Automotive ECU/Control Unit, should have software re-programming capability.
Automotive ECUs (Control Units) support host of functionalities. These features and functionalities have grown to become more advanced and complex.
It has become imperative for the Automotive OEMs' and Suppliers to ensure that these software-driven control units are always operating in a secure & efficient environment.
This can be ensured only if the ECUs' (control units), within your vehicle, have the latest and updated version of the software and security patches.
Hence, the application software designed and ported on **the MCU platform needs to be updated** quite frequently, either through a remote location or at the service station.
The Bootloader software, which occupies the ROM of the Electronic Control Unit, has been entrusted with this responsibility of facilitating the **ECU software update.**

How Bootloader Software Makes Re-programming of an Automotive ECU, an Efficient Process?

An F-35 fighter aircraft has approximately **5.7 million lines of code** and a Boeing 787 stands at a little higher than **6.5 million lines of code**.
But when on examining a luxury car, one would observe that it has close to **100 million lines of software code.**
Thus, the firmware update of an Automotive ECU (Electronic Control Unit) becomes a tedious job, due to such huge volumes of software code.
A Bootloader Software is designed to automate this process of flash re-programming and managing the firmware update.
At every Boot-up of the System, the Bootloader software checks if the latest/updated version of the ECU software is available.
If yes, then before the booting of the system, Bootloader software downloads and stores the new updated version of the firmware. Post this the system boot-up is executed and system now runs on the new version of the software in a completely secure environment.

Understanding how Firmware is Updated, using a Flash Bootloader: a Step-by-Step Process Guide

A Flash Bootloader module is designed to update the firmware without the use of any specialized hardware such as JTAG.
Following is the flowchart of the steps involved in ECU reprogramming using Flash Bootloader:

- The Flash Bootloader module is the first software module that gets activated during the booting of the system (after the power supply of the device is switched on).

- The control is transferred to the Flash Bootloader, which checks for the updated version of the firmware. If a new version is available, Flash Bootloader analyzes the update to authenticate the source and cross-check all the pre-defined system security parameters. If the authentication is successful then Bootloader Module writes the new updated on the flash memory, at the target address.
- Next, the updated program is again verified to check the integrity and fidelity of the software .If no anomaly is found, the control is transferred from the Flash Bootloader to the Application.

All these steps, involved in the process of an ECU reprogramming, are performed by two different parts of the Bootloader- the Primary and the Secondary Bootloader.
In the next section, we explain both of them.
Primary Bootloader: Microcontroller Unit Setup
After the microcontroller is reset, the control first comes to Primary Bootloader. This is the part of the Bootloader software that initializes and sets up the MCU resources.
In the context of a CAN based Bootloader, the resources include the CAN protocol and the CAN controller. The CAN controller is initiated to download the update, with the correct CAN speed. Any update transmitted over CAN BUS is identified by the CAN protocol and subsequent actions are taken.
Secondary Bootloader: Microcontroller Reprogramming
Similar to the primary Bootloader that initializes various functionalities, there is another part of the Bootloader software that gets active upon receiving the firmware update via the communication medium. It is known as the Secondary Bootloader.
Essentially, this secondary Bootloader downloads the updated related to the target application and reprograms the flash memory.
It has all the necessary flash routines and the UDS functions (if the Bootloader is UDS based) that are necessary for storing the updated data on the flash memory.
The Primary Bootloader transfers the control to the Secondary Bootloader, once the application update is received.
Next, the secondary Bootloader initializes **Flash Bus Interface Unit (FBUI)** on the RAM memory**.** This enables the capability to read/write on the flash memory.
Immediately after this step, the communication over CAN is initialized again, so that the Bootloader is ready to receive the UDS request.
Post this, the update is stored in the flash memory. This is when the Secondary Bootloader is erased from the RAM and control once again goes to the Primary Bootloader.

Role of UDS Software Stack in an Automotive ECU Reprogramming

According to the **AUTOSAR Standard**, **Unified Diagnostics Services (UDS)** is the most suitable protocol for implementing Bootloader, for the purpose of ECU reprogramming.
There are multiple reasons for that and we would discuss them in detail in other UDS blogs. For now, we will focus on the role played by UDS in a Bootloader operation.
Following are the key responsibilities of UDS in an ECU flashing operation:

1. UDS sets the server into a reprogramming mode and start the reprogramming sequence
2. It handles the initiation and termination of the data transfer
3. UDS takes care of size and order of data blocks to be sent/received and memory blocks where the data will be stored
4. The UDS services allow the client to start or stop a routine, which may be running on the server
5. It allows the client to initiate a software reset event on the server

UDS performs all these responsibilities with the help of certain software services. Some of these services and their use in flashing in explained in the table below.

How Remote ECU Flashing works, using a Specialized Flash Bootloader

In order to perform automotive control unit reprogramming, the vehicle has to be brought to the garage where the service engineer will do the needful.
However, as the software used in vehicles is growing in the volume and complexity, numbers of the software upgrades have also become more frequent.
In such scenarios, the OEMs cannot expect the vehicle owners to bring their car to the garage frequently, for the ECU flashing. A Bootloader software enabled with remote update capabilities over Ethernet is the key here.
With **DoIP (Diagnostics over Internet Protocol),** remote ECU update has become possible.
While the rest of the components and services are similar, usage of Ethernet as the medium for communication, has made the remote update possible.
In simple terms, the firmware update that is transmitted over CAN, is sent over Ethernet using a DoIP software stack.
The DoIP stack inside the Bootloader software identifies the message and the entire process of ECU reprogramming is performed.
Whether based on DoIP or UDS, Flash Bootloader is a component that is indispensable, given the importance of regular ECU updates.
And with the remote ECU update capabilities, it is going to be easy for the OEMs and other automotive suppliers to ensure the robustness and safety of the software-driven Electronic Control Units (ECU) in our