Direct Flashing vs Flashing through MCU/MPU

Introduction

In-system programming (ISP) is a critical technique that enables the programming of devices—such as microcontrollers (MCUs), memories, CPLDs (Complex Programmable Logic Devices)…—without needing to physically remove the components from the circuit board. This approach simplifies the development and manufacturing processes, as it allows for programming, updating, and configuring devices while they are already installed in the system.

There are two primary methods for implementing in-system programming:

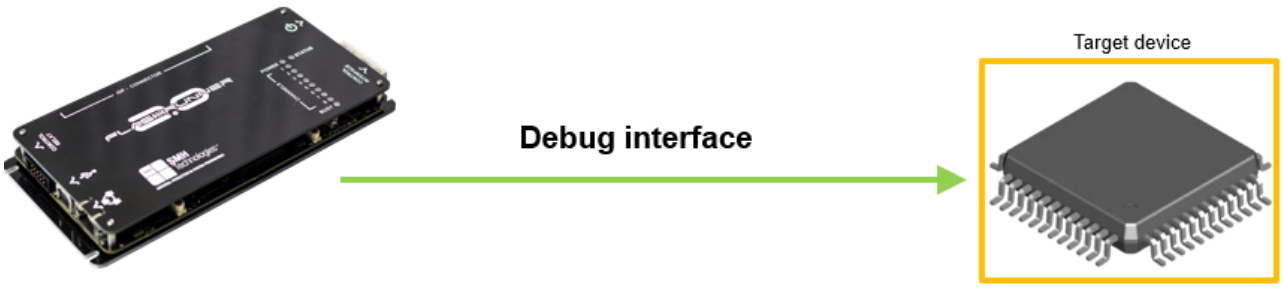
1. **Direct In-System Programming (Direct Flashing)**
2. **In-System Programming via MCU or Microprocessor (MPU)**

The choice between these two methods largely depends on various factors, including the project's specific requirements, the overall cost, performance expectations, and the complexity of the hardware design. Each method offers distinct advantages and trade-offs.

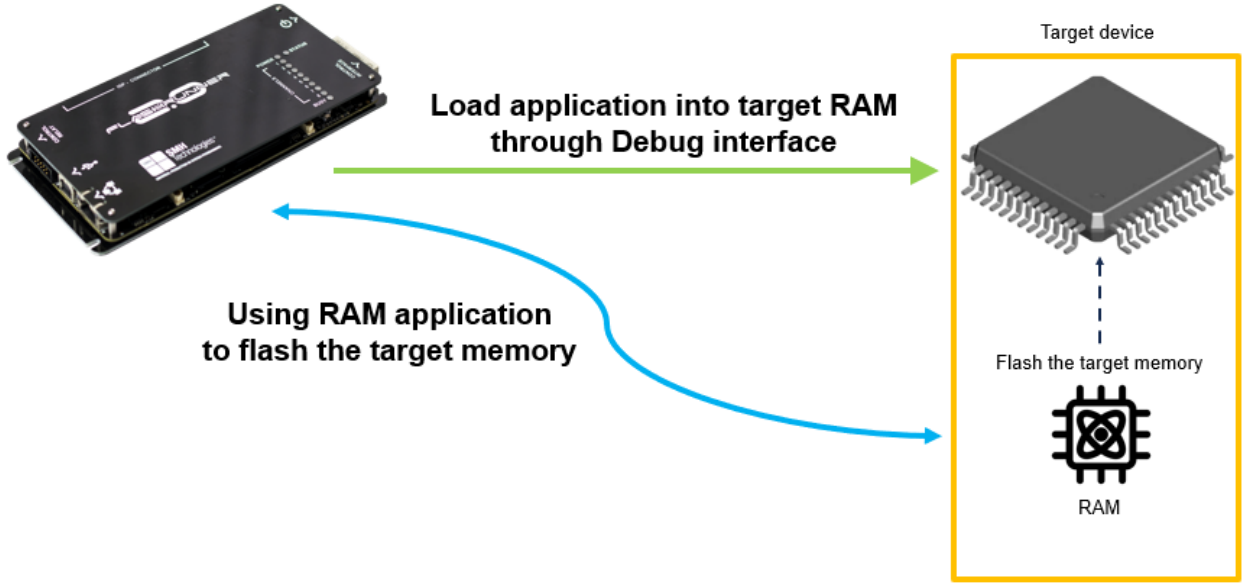
Direct in-system programming

Direct in-system programming, also known as **Direct Flashing**, is the most common approach for programming memory devices embedded on a circuit board. In this method, the target device (e.g., a microcontrollers, memories…) is directly programmed through a dedicated **debugging interface** such as **UART**, **SPI**, **I2C**, **JTAG**, **SWD** (Serial Wire Debug)... These interfaces allow a programming system to load the device's firmware directly into the device's memory. The flashing process can be performed in one of two main ways:

- **Direct Flashing via Debug Interface**: The firmware is directly transmitted through the debug interface into the target device's memory. This method is relatively straightforward and typically used when the target device has the necessary interfaces for direct communication.



- **Indirect Flashing via Application in RAM**: Alternatively, the programming system may load an application into the target device's **RAM**. This application, once executed, will execute the flashing process by communicating with the programming system through the debug interface. RAM serves as temporary storage for the executable code, allowing the programming system to perform necessary actions, such as transferring firmware to the device's non-volatile memories.



Advantages of Direct In-System Programming:

- **Simplicity**: This method is often simpler to implement because it directly leverages standard debug interfaces.
- **Speed**: Flashing through dedicated interfaces tends to be faster and more efficient, especially when using high-speed protocols like JTAG/SWD or SPI.
- **Widely Supported**: Many MCUs and programmable devices come with built-in support for direct programming through these common debug interfaces, making it a widely accepted and accessible solution in embedded development.

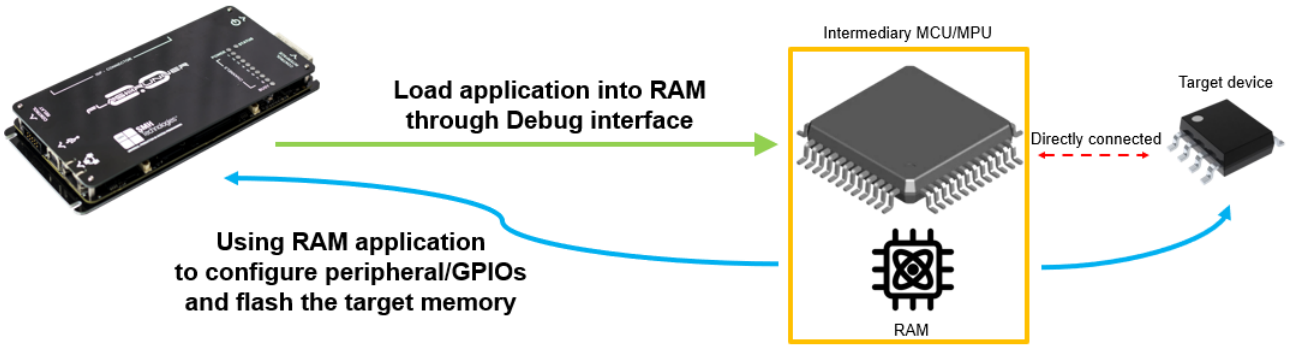
Disadvantages of Direct In-System Programming:

- **Limited Flexibility in Complex Systems**: In systems where multiple devices are present some of the devices debug interfaces might be not available or accessible, so this method may not be feasible at all or without additional hardware intervention.

In-system programming via MCU/MPU

In contrast to direct flashing, **in-system programming via MCU or MPU** involves using an intermediary **microcontroller (MCU)** or **microprocessor (MPU)** to program the target device. This method is used when the target device cannot be directly programmed through a debug interface, or when the system design doesn't provide access to such interfaces.

In this approach, the target device is connected to a peripheral of the MCU/MPU, such as **QSPI** (Quad SPI) or specific **GPIO pins**. When the system's programming environment does not allow direct access to the target device's debug interface, the MCU/MPU's debug interface is used (JTAG, USB…) to load an application into the **RAM** of the intermediary MCU/MPU. Once the application is loaded and executed, it takes control of the system's peripherals (e.g., GPIOs or other interfaces) by configuring and initializing them and communicates with the programming system through the debug interface to flash the target device directly connected to this intermediary MCU/MPU.



Advantages of In-System Programming via MCU/MPU:

- **Greater Flexibility**: This method is highly flexible, particularly in complex systems where the target device does not have a direct debugging interface or where additional control over peripheral interfaces is needed.

Disadvantages:

- **Increased Complexity**: The system design becomes more complex since additional software is required to control the flashing process.
- **Slower Flashing Speed**: The flashing process may be slower compared to direct programming methods, as it relies on the performances of the intermediary MCU/MPU, rather than the direct debug interface of the target device.

Choosing Between Direct Flashing and Flashing via MCU/MPU

The decision between direct flashing and flashing through an intermediary MCU/MPU depends on several factors:

1. **Project Requirements**: If the target device has accessible debug interfaces and the flashing process needs to be fast and simple, **direct flashing** is the preferred method. However, if the system is more complex or the target device lacks a debug interface, **flashing via MCU/MPU** becomes necessary.
2. **Cost and Hardware Design**: Direct flashing typically requires fewer resources and simpler software. On the other hand, flashing via MCU/MPU adds more software complexity, potentially increasing costs.
3. **Performance Needs**: Direct flashing offers faster programming times, making it ideal for mass production or high-performance applications. Flashing via MCU/MPU, while more flexible, may have slower programming speeds and additional overhead due to the intermediary MCU/MPU.
4. **System Complexity**: For simpler systems with limited devices, direct flashing is usually sufficient. However, for larger, multi-device systems or those that require more specialized control over peripheral interfaces, flashing through an MCU/MPU is often the better choice.

Conclusion

Both **direct in-system programming** and **in-system programming via MCU/MPU** have their respective strengths and are suitable for different types of projects. **Direct flashing** is generally preferred for its simplicity and speed when direct debug interfaces are available. On the other hand, **flashing through an intermediary MCU/MPU** is advantageous when more flexibility is needed, such as in complex systems or where the target device lacks a direct programming interface. The choice depends on the specific needs of the project, including the complexity of the hardware design, performance requirements, and cost considerations.