StartUp File

**High Level Design & Low-Level Design**

The purpose of this document is to provide a template for documenting both HLD & LLD.

Contents

[**2.** Expected Outcomes 3](#_Toc162348240)

[3. Deliverables 3](#_Toc162348241)

[4. Startup File Structure and Flow 3](#_Toc162348242)

[5. Reset Vector 4](#_Toc162348243)

[6. Reset Handler 4](#_Toc162348244)

[7. System Initialization 4](#_Toc162348245)

[8. System Configuration 4](#_Toc162348246)

[9. Interrupt Handlers 4](#_Toc162348247)

[10. Linker Script 4](#_Toc162348248)

[ MEMORY command 4](#_Toc162348249)

[ SECTIONS command 4](#_Toc162348250)

[ ENTRY command 4](#_Toc162348251)

[11. Code Snippets 5](#_Toc162348252)

[12. Conclusion 5](#_Toc162348253)

**1.** Introduction

The STM32F407 is a high-performance microcontroller from STMicroelectronics. It is part of the STM32 family of microcontrollers based on the ARM Cortex-M4F processor core.   
This document provides a combined high-level and low-level design of the startup file for the STM32F407 microcontroller.

# **2.** Expected Outcomes

The goal is to develop a startup file for the STM32F series controller with possible configurations supported by the hardware.

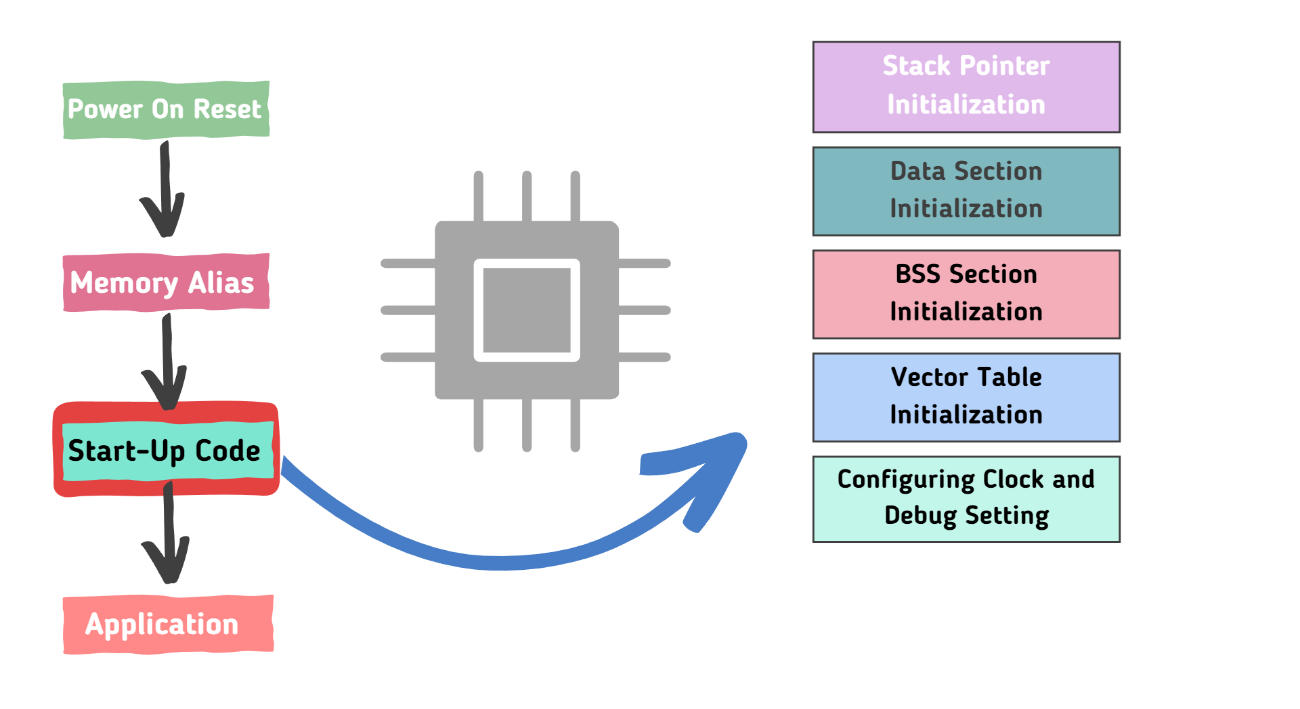
# 3. Deliverables

The project folder for STM32 with the bare minimum APIs mentioned below will be delivered.

# 4. Startup File Structure and Flow

The startup file typically includes sections such as:

* Reset Vector: This points to the initial address of the program.
* Reset Handler: This initializes the processor, copies data from Flash to RAM, and jumps to the main application.
* System Initialization: This initializes system peripherals such as clocks, GPIOs, and other relevant hardware.
* System Configuration: This configures memory mapping, exception vectors, and other system-level settings.
* Interrupt Handlers: These are defined for handling interrupts.



# 5. Reset Vector

The reset vector is a crucial part of the startup file in the microcontroller. The reset vector points to the Reset\_Handler function, which is executed after the microcontroller is powered on or reset.

6. Reset Handler

In the Reset\_Handler function, the first instruction loads the address of \_estack (which is the top of the stack) into register r0. The second instruction moves the value in r0 to the stack pointer sp. This sets the stack pointer to the top of the stack.

This process is crucial because it prepares the stack memory for the execution of the program. After the Reset\_Handler function is executed, the control is transferred to the main function of your program.

# 7. System Initialization

System initialization includes initializing system peripherals such as clocks, GPIOs, and other relevant hardware.

# 8. System Configuration

System configuration includes configuring memory mapping, exception vectors, and other system-level settings.

# 9. Interrupt Handlers

Interrupt An interrupt handler, also known as an interrupt service routine (ISR), is a special function in microcontroller firmware that is executed in response to an interrupt1. Interrupts are signals sent to the CPU that indicate that a particular event has occurred, which requires immediate attention.

In the context of the STM32F407 microcontroller, each interrupt or exception has a corresponding handler function defined in the startup file1. The addresses of these handler functions are stored in the interrupt vector table1. When an interrupt is triggered, the microcontroller jumps to the address of the corresponding handler function and executes it.

# 10. Linker Script

The A linker script is a description of the memory layout of an embedded system that is used by the linker to control the placement of code and data into different memory regions. It is a crucial component in the build process of an embedded system.

* MEMORY command: This command defines the memory layout of the microcontroller.
* SECTIONS command**:** This command is used to define the output sections. These sections include the vector table, program code, and data. The SECTIONS command controls where these sections are placed in memory.
* ENTRY command: This command specifies the entry point of the program. For the STM32F407, the entry point is usually the Reset\_Handler function.  
  application.

# 11. Code Snippets

Here are some code snippets from the startup file to illustrate key points:

.section .isr\_vector,"a",%progbits

.long \_estack /\* Initial stack pointer value \*/

.long Reset\_Handler /\* Reset handler entry point \*/

* Reset\_Handler:

/\* Initialize stack pointer \*/

ldr r0, =\_estack

mov sp, r0

/\* Copy data from Flash to RAM \*/

ldr r0, =\_sidata /\* Load data address \*/

ldr r1, =\_sdata /\* Load RAM address \*/

ldr r2, =\_edata /\* End of data section \*/

copy\_loop:

cmp r1, r2 /\* Check if end of data section is reached \*/

bge init\_bss /\* If yes, initialize BSS section \*/

ldr r3, [r0], #4 /\* Load word from Flash, increment source pointer \*/

str r3, [r1], #4 /\* Store word to RAM, increment destination pointer \*/

b copy\_loop /\* Repeat until end of data section \*/

init\_bss:

ldr r0, =\_sbss /\* Load BSS start address \*/

ldr r1, =\_ebss /\* Load BSS end address \*/

mov r2, #0 /\* Clear value for BSS \*/

zero\_loop:

cmp r0, r1 /\* Check if end of BSS section is reached \*/

bge main /\* If yes, jump to main \*/

str r2, [r0], #4 /\* Store zero to BSS, increment pointer \*/

b zero\_loop /\* Repeat until end of BSS section \*/

main:

/\* Call main application \*/

bl main

# 12. Conclusion

This document provides a combined high-level and low-level design of the startup file for the STM32F407 microcontroller. The startup file is a crucial component of any embedded system as it contains the code that runs immediately after the system is powered on or reset. It initializes critical system components, sets up the initial stack pointer, and defines the vector table.