|  |
| --- |
| Baremetal app. from scratch without ide by a startup.c |
| Author: Eng. Mustafa Hafez | Date: July-7-2024 |

# Overview

## Description

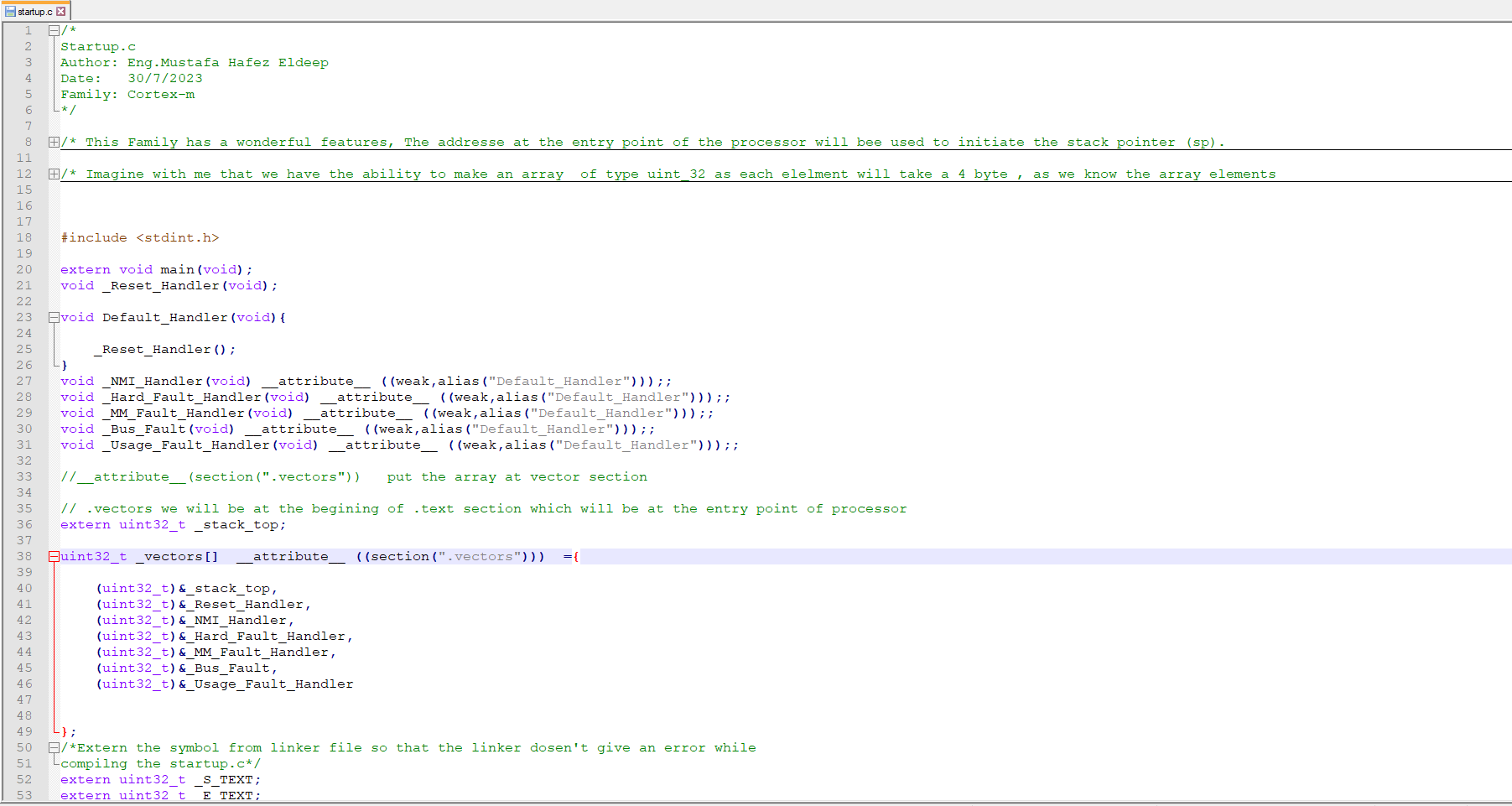
|  |  |
| --- | --- |
| Badge Tick1 with solid fill | The arm-cortex-m Family has a wonderful feature, The address at the entry point of the processor will be used to initiate the stack pointer (sp). so, we can write the startup file by C-language as the stack will be initiated. |

## Requirements:

|  |  |
| --- | --- |
| Badge Tick1 with solid fill | main.c – Startup.c – Linkerscript.ld - Makefile ---🡪 arm-none-ebai cross toolchain. |

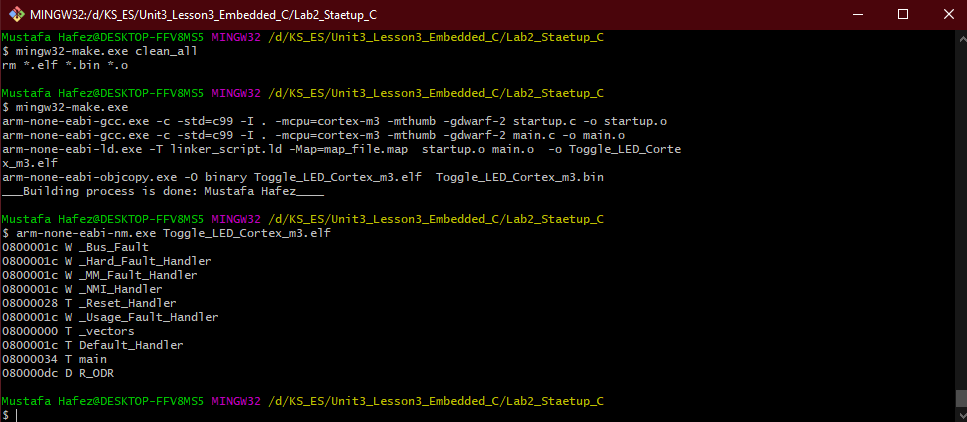
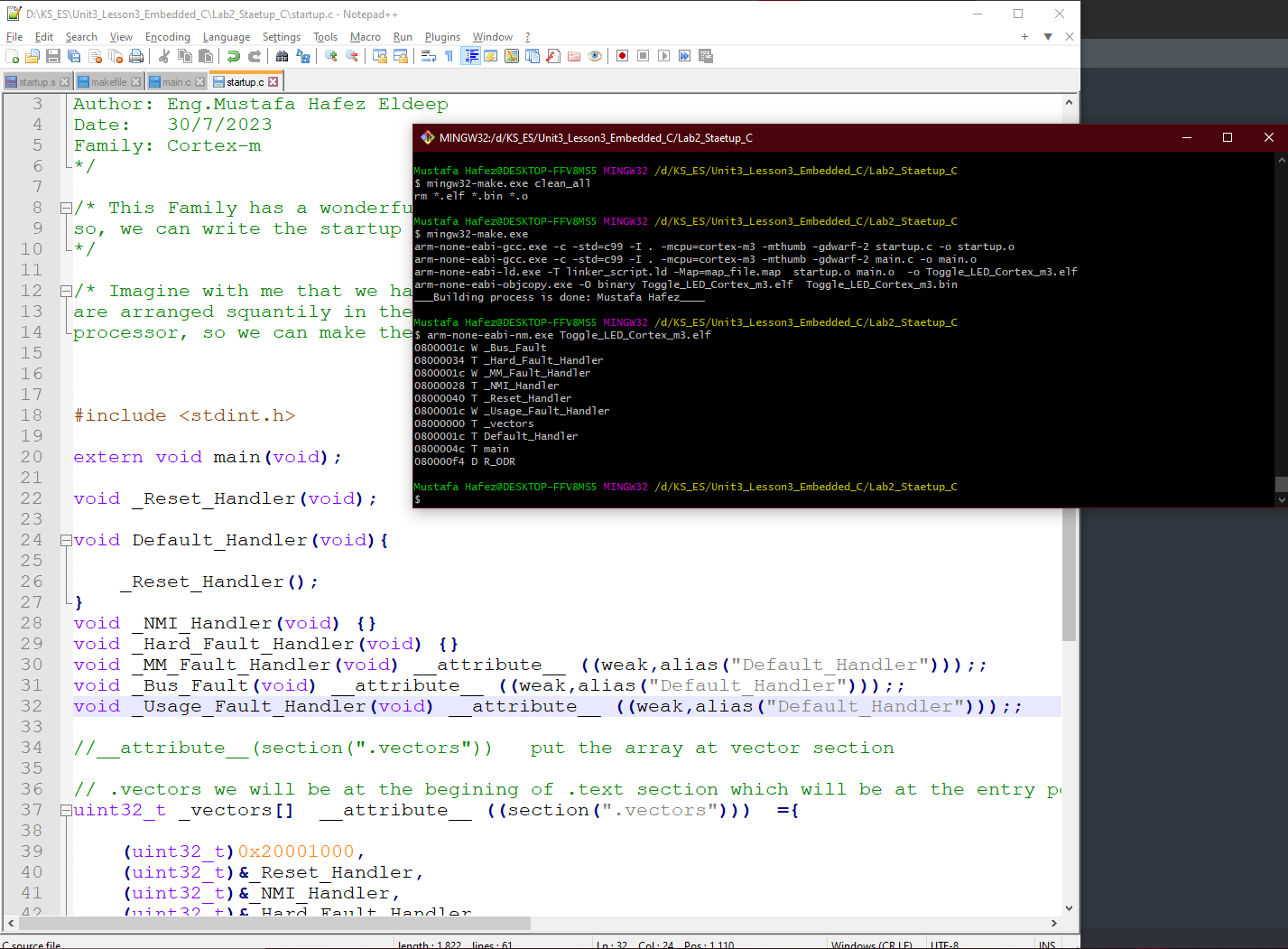
## Procedure

|  |  |
| --- | --- |
|  | Imagine that we have the ability to make an array of type uint\_32 as each elements will take a 4 byte , as we know the array elements are arranged sequentially in the memory, so if we make the first element of our array equal to the address we want to give to the SP, and by someway but this array at the entry point of processor, so we can make the startup file and the next element in array will be next to the SP address.    The other Handler is arranged according to the vendor vector table.  The command \_\_attribute\_\_ ((section(".vectors"))) is used to put some lines at a desired spcified section. so we put our arrays at a. vector section  **Here we used an array of pointer to functions to hold the handlers addresses** |



when we define the vector table of our MC, it may be having a huge number of interrupts, so it's not sensible to define all of them, in addition, the user itself may not use all of them so why occupies an address although the user doesn’t use them?!!

The weak and alias help to avoid that, as we give the function a weak attribute to make it overridden and an alias to a default handler definition by this way, a user can override the function detention and if it not defined, it takes the default aliased definition.



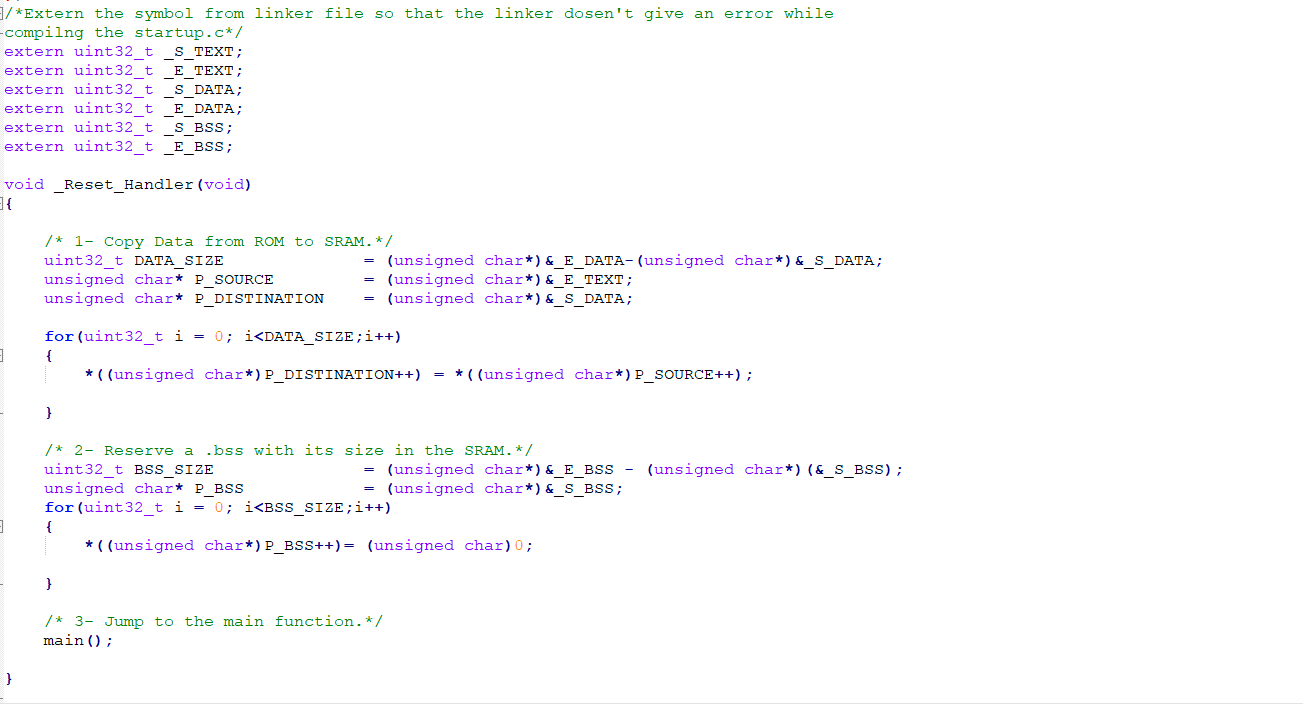
**Weak and alias to the Default \_Handler**

Bus\_Fault and Usage\_Fault\_Handler

Have a weak and alias to Default handler so that each one of them has the same address.

NMI\_Handler and Hard\_Fault\_Handler

Have not a weak neither alias so that each one of them has its own address.



Important Note:

1-We use VMA (Virtual memory address) while copying the data to SRAM while LMA (Load Memory Address) is the physical address at burning.

2-Don't forget to **check the map file** to make sure everything is correct, and the memory is aligned (no odd address as its effect on performance).

3-Use ALIGN (4) to align your memory with 4 Bytes if you need.

2-Intiate the. bss with zero at the SRAM:

As we know the. bss section doesn’t exist at the flash but with an information of its size we initialize the SRAM with it.

we can do that if we know:

- what's the start address of .bss sectoin

- what's the end address of .bss sectoin.

- As a result the size of .data is known.

1-Copying data from ROM to SRAM:

As we know the .data section is burned on the flash but during the startup, it's copied to the SRAM ,we can do that if we know:

- what's the start address of .data section

- what's the end address of .data section.

- As a result, the size of .data is known.

A screen shot of a computer

Description automatically generatedA screen shot of a computer

Description automatically generated

**All sections are aligned so that no need to a ALIGN(4). You can put it for safety.**

**The \_S\_DATA at a VMA = 0x2000000 which is the beginning of the SRAM while the LMA is 0x080000dc the same of \_E\_TEXT**

**The Handlers have the same addresses (Alias to Default Handler)**

**. vectors section at the beginning of .text at address 0x08000000**

**The start address of the SRAM at 0x20000000**

**The start address of the flash at 0x08000000**