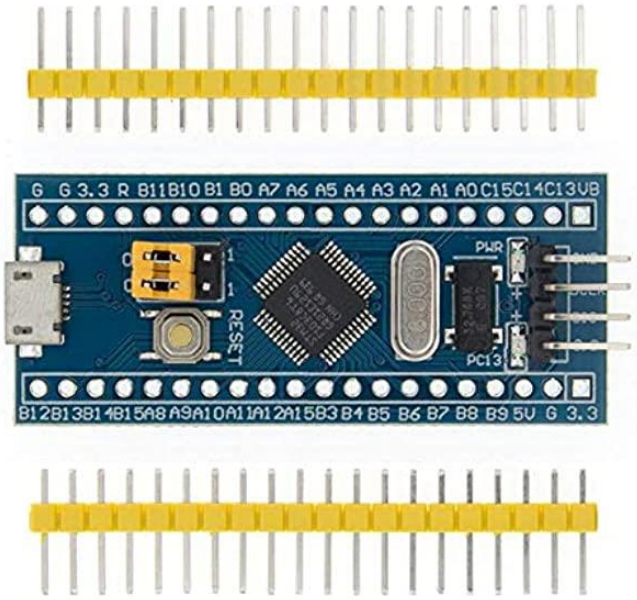
Lab2

ARM CORTEX-M3 STM32



Name: *Aya Sayed Abdellah*

In this lab I will write a bare-metal software to toggle led is connected to GPIO port C13, To make a GPIO toggling in STM32. After reading its specs I found that:

I need to work with two peripherals:

- *RCC (reset and clock control in **0x18 APB2ENR Register** in **IOPAEN Bit 2**) is necessary because the GPIO has disabled clock by default. Its base address is **0x40021000**.*
- *GPIO (general purpose input/output) so I connected it to GPIO Port A with address **0x40010800**, in GPIO_PA I have to read 2 in **0x04 CRH Register** to active mode pin 13 (from bit 20 to 24) and **in 0x0C ODR Register** (pin 13)*

Now I have all what I need to write application file I call it main.c

Next step it to read also its specs because Some part of startup code dependent on the target processor.

*When power is applied to the MCU the Program Counter (PC) value will be 0 which will mappe **to 0x08000000** and will therefore start at address **0x08000000**. This address is then copied to **the Stack Pointer (SP) register** for later use.*

The Program Counter then steps to the next address which is 0x0800 0004 and expects the address of the reset handler at this location then the next handler from the vector table handler.

*Also I need **.thumb_func** This directive specifies that the following symbol is the name of a Thumb encoded function. This information is necessary in order to allow the assembler and linker to generate correct code for interworking between Arm and Thumb instructions. Because there are 2 types of instructions are provided in this processor are: 16 bit instruction and 32 bit instruction.*

So I created a complex startup that consists of :

1. *Define Interrupt vectors Section*
2. *Copy Data from ROM to RAM*
3. *Initialize Data Area*
4. *Initialize Stack*
5. *Create a reset section and Call main( ).*

Before create it and because of SP initialized automated by the processor we can write startup.s and startup.c that feature is provided only for cortex-M3 and its family. So I wrote the 2 files.

And in the linker file I need to Aligned access memory to Efficiency fetch and execute so I used this command in linker script file (. = ALIGN(4)).

All the above information that I need it to write a bare-metal software to toggle the led let's start…

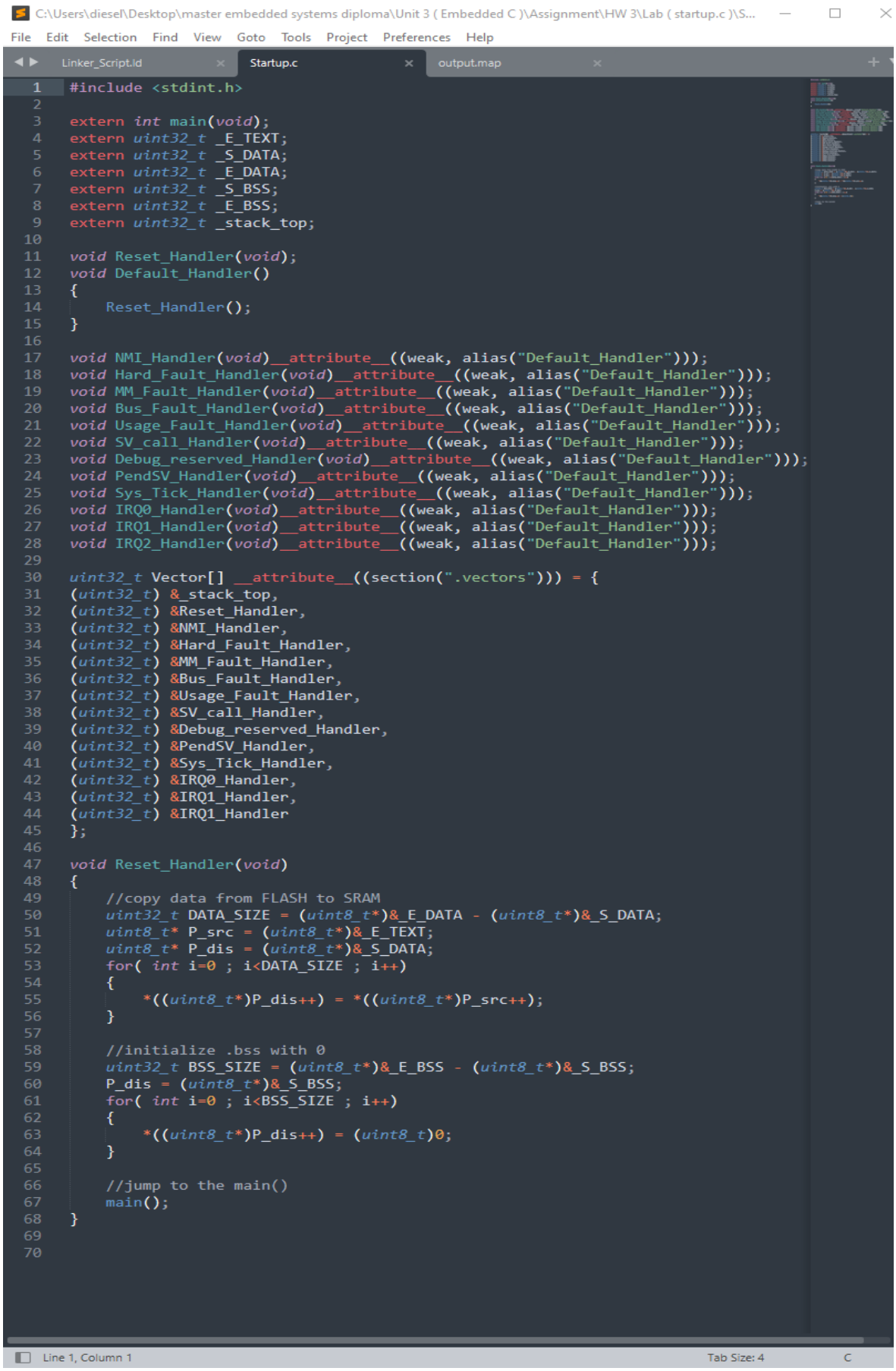
C:\Users\diesel\Desktop\master embedded systems diploma\Unit 3 (Embedded C)\Assignment\HW 3\Lab (startup.s)\...

File Edit Selection Find View Goto Tools Project Preferences Help

Startup.s | output.map | Makefile | main.c | Linker_Script.ld

```c
/*******************************************************************************
Master Embedded System Diploma <Learn in depth>
file           : main.c
Author         : Aya Sayed
brief          : Main program body
*******************************************************************************/

#include "stdint.h"
typedef volatile uint32_t   vuint32_t;

/*******************************************************************************/
#define Set(bit)        |=bit                              //to set bit
#define AND(value)      &=value                           //to AND content of re
#define OR(value)       |=value                           //to OR content of reg

#define ON              1                                 //LED IS ON
#define OFF             0                                 //LED IS OFF

#define Delay           for( int i=0 ; i<5000 ; i++ );    //Delay Macro
/*******************************************************************************/

//Register addresses
#define RCC_BASE        0x40021000
#define GPIOA_BASE      0x40010800
#define RCC_APB2ENR     *( vuint32_t* )( RCC_BASE    + 0x18 )
#define GPIOA_CRH       *( vuint32_t* )( GPIOA_BASE  + 0x04 )
#define GPIOA_ODR       *( vuint32_t* )( GPIOA_BASE  + 0x0C )

//Bit fields
#define RCC_IOPAEN      ( 1<<2 )
#define GPIOA13         ( 1UL<<13 )


typedef union
{
    vuint32_t          all_fields;
    struct
    {
        vuint32_t     reversed:13;
        vuint32_t     pin_13:1;
    }Spin;
}R_ODR_t;

unsigned char g_variables[3] = {1,2,3};
unsigned char const const_variables[3]={1,2,3};
volatile R_ODR_t*  R_ODR =  ( volatile R_ODR_t* )( GPIOA_BASE + 0x0C );

int main(void)
{
    RCC_APB2ENR  Set(RCC_IOPAEN);
    GPIOA_CRH    AND(0xFF0FFFFF);
    GPIOA_CRH    OR(0x00200000);
    while(1)
    {
        R_ODR->Spin.pin_13 = ON ;   Delay
        R_ODR->Spin.pin_13 = OFF;   Delay
    }
}

```

Line 1, Column 1 Tab Size: 4 C


C:\Users\diesel\Desktop\master embedded systems diploma\Unit 3 (Embedded C)\Assignment\HW 3\Lab (startup.c)\S...
File Edit Selection Find View Goto Tools Project Preferences Help
Linker_Script.ld
Startup.c
output.map
#include <stdint.h>
extern int main(void);
extern uint32_t _E_TEXT;
extern uint32_t _S_DATA;
extern uint32_t _E_DATA;
extern uint32_t _S_BSS;
extern uint32_t _E_BSS;
extern uint32_t _stack_top;
void Reset_Handler(void);
void Default_Handler()
{
Reset_Handler();
}
void NMI_Handler(void)__attribute__((weak, alias("Default_Handler")));
void Hard_Fault_Handler(void)__attribute__((weak, alias("Default_Handler")));
void MM_Fault_Handler(void)__attribute__((weak, alias("Default_Handler")));
void Bus_Fault_Handler(void)__attribute__((weak, alias("Default_Handler")));
void Usage_Fault_Handler(void)__attribute__((weak, alias("Default_Handler")));
void SV_call_Handler(void)__attribute__((weak, alias("Default_Handler")));
void Debug_reserved_Handler(void)__attribute__((weak, alias("Default_Handler")));
void PendSV_Handler(void)__attribute__((weak, alias("Default_Handler")));
void Sys_Tick_Handler(void)__attribute__((weak, alias("Default_Handler")));
void IRQ0_Handler(void)__attribute__((weak, alias("Default_Handler")));
void IRQ1_Handler(void)__attribute__((weak, alias("Default_Handler")));
void IRQ2_Handler(void)__attribute__((weak, alias("Default_Handler")));
uint32_t Vector[] __attribute__((section(".vectors"))) = {
(uint32_t) &_stack_top,
(uint32_t) &Reset_Handler,
(uint32_t) &NMI_Handler,
(uint32_t) &Hard_Fault_Handler,
(uint32_t) &MM_Fault_Handler,
(uint32_t) &Bus_Fault_Handler,
(uint32_t) &Usage_Fault_Handler,
(uint32_t) &SV_call_Handler,
(uint32_t) &Debug_reserved_Handler,
(uint32_t) &PendSV_Handler,
(uint32_t) &Sys_Tick_Handler,
(uint32_t) &IRQ0_Handler,
(uint32_t) &IRQ1_Handler,
(uint32_t) &IRQ1_Handler
};
void Reset_Handler(void)
{
//copy data from FLASH to SRAM
uint32_t DATA_SIZE = (uint8_t*)&_E_DATA - (uint8_t*)&_S_DATA;
uint8_t* P_src = (uint8_t*)&_E_TEXT;
uint8_t* P_dis = (uint8_t*)&_S_DATA;
for(int i=0 ; i<DATA_SIZE ; i++)
{
((uint8_t)P_dis++) = *((uint8_t*)P_src++);
}
//initialize .bss with 0
uint32_t BSS_SIZE = (uint8_t*)&_E_BSS - (uint8_t*)&_S_BSS;
P_dis = (uint8_t*)&_S_BSS;
for(int i=0 ; i<BSS_SIZE ; i++)
{
((uint8_t)P_dis++) = (uint8_t)0;
}
//jump to the main()
main();
}
Line 1, Column 1
Tab Size: 4
C

C:\Users\diesel\Desktop\master embedded systems diploma\Unit 3 (Embedded C)\Assignment\HW 3\Lab (startup.c)\L...

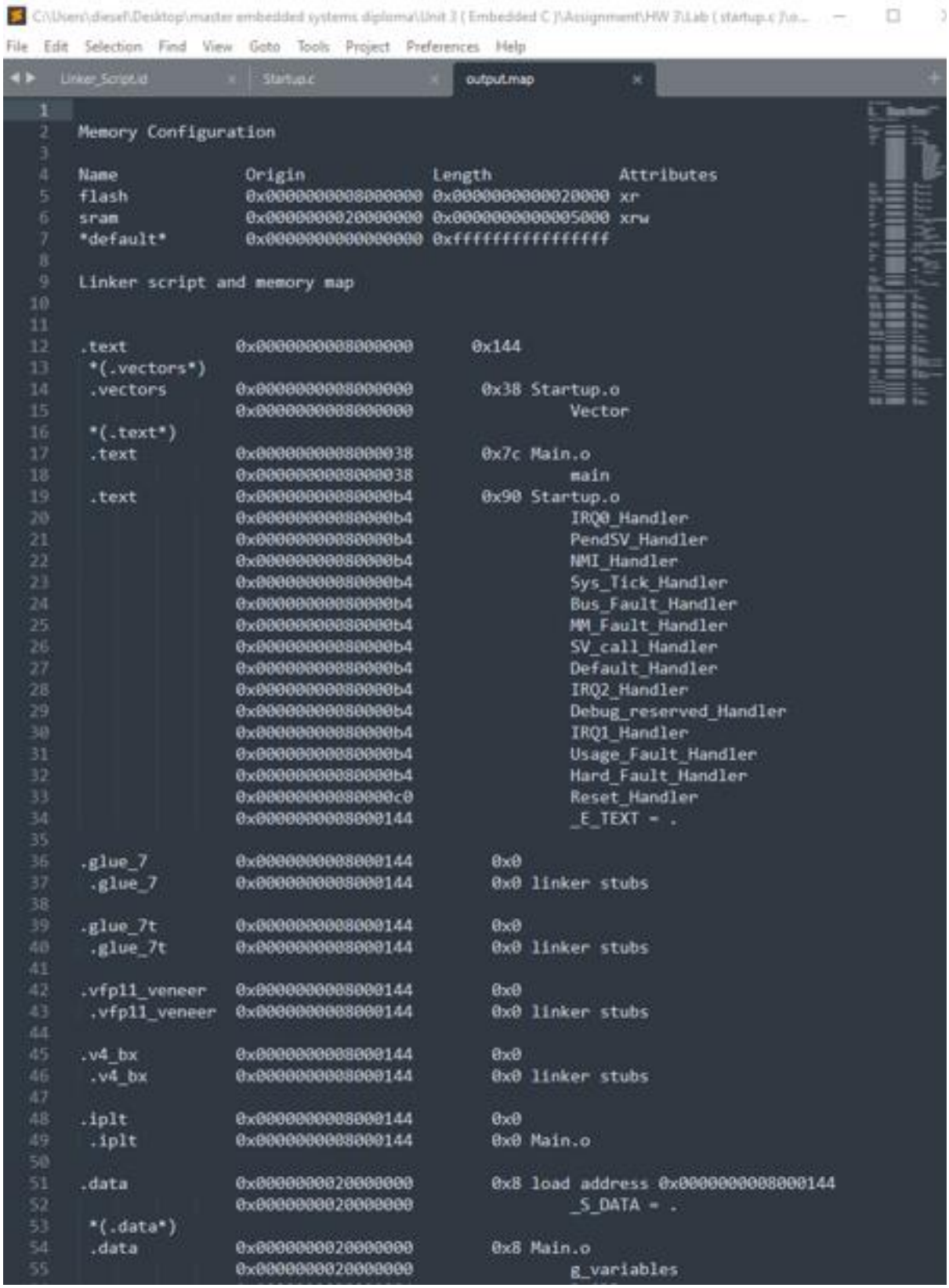
File Edit Selection Find View Goto Tools Project Preferences Help

Linker_Script.ld | Startup.c | output.map

```
/*****************************************************************************
Master Embedded System Diploma <Learn in depth>
file           : Linker_Script.ld
Author         : Aya Sayed
brief          : Linker Script program body
*****************************************************************************/

MEMORY
{
    flash(RX)  : ORIGIN = 0x08000000, LENGTH = 128k
    sram(RWX)  : ORIGIN = 0x20000000, LENGTH = 20k
}

SECTIONS
{
   .text :{
       *(.vectors*)
       *(.text*)
       _E_TEXT = . ;
   }> flash
   .data :{
       _S_DATA = . ;
       *(.data*)
       . = ALIGN(4);
       _E_DATA = . ;
   }> sram AT> flash
   .bss :{
       _S_BSS = . ;
       *(.bss)
       . = ALIGN(4);
       _E_BSS = . ;
       . = . + 0x1000;
       _stack_top = . ;
   }> sram
   .rodata :{
       *(.rodata* )
   }> flash
}
```

MINGW32:/c/Users/diesel/Desktop/master embedded systems diploma/Unit 3 (Embedded C)/Assignment/H...

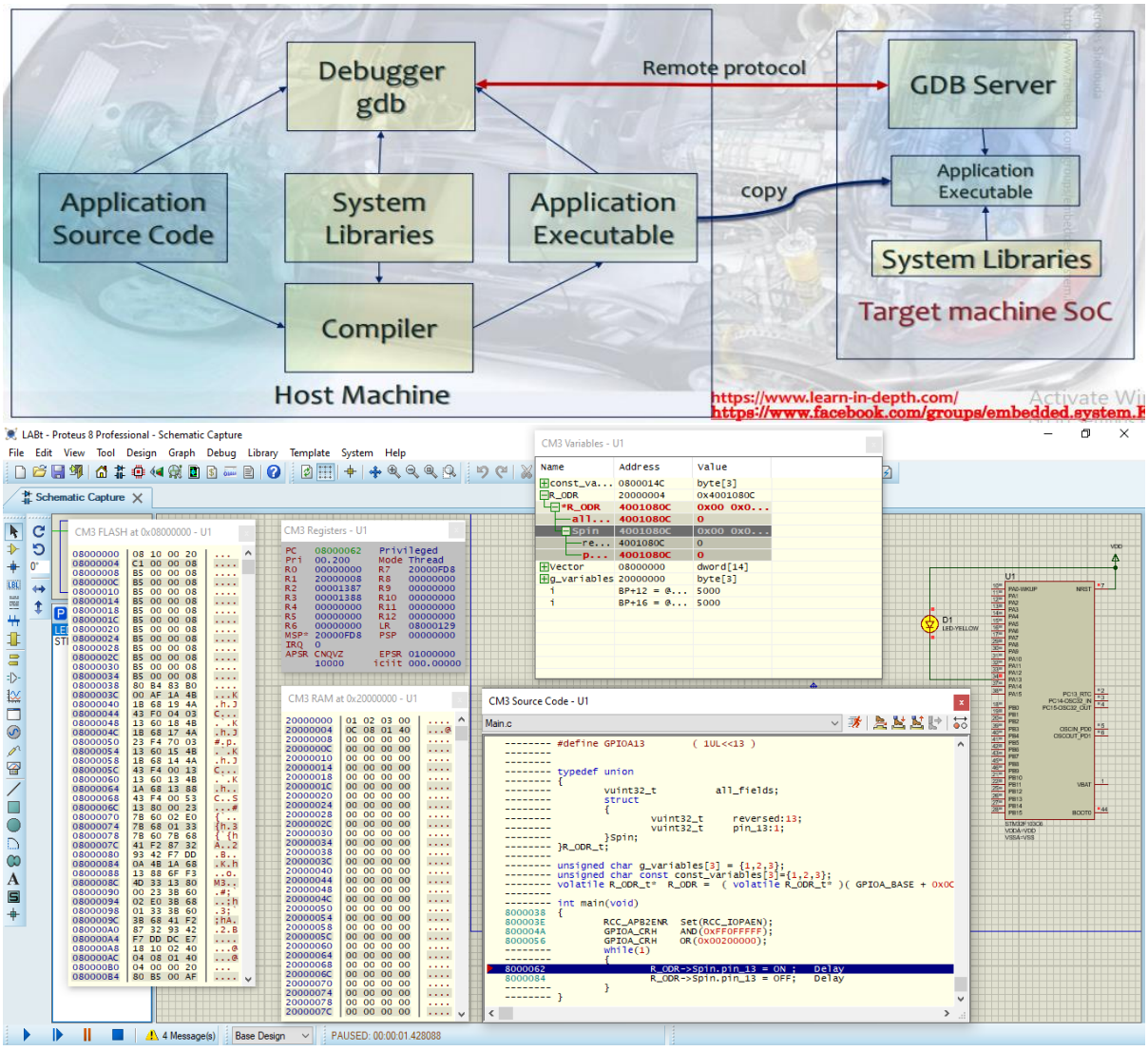
```
diesel@DESKTOP-ET889OF MINGW32 ~/Desktop/master embedded systems diploma/Unit 3 ( Embedded C )/Assignment/
HW 3/LAB2
$ export PATH=/c/ST/STM32CubeIDE_1.4.0/STM32CubeIDE/plugins/com.st.stm32cube.ide.mcu.externaltools.gnu-too
ls-for-stm32.7-2018-q2-update.win32_1.4.0.202007081208/tools/bin:$PATH

diesel@DESKTOP-ET889OF MINGW32 ~/Desktop/master embedded systems diploma/Unit 3 ( Embedded C )/Assignment/
HW 3/LAB2
$ make
arm-none-eabi-gcc.exe  -c -mcpu=cortex-m3 -gdwarf-2 -I . Main.c -o Main.o
arm-none-eabi-gcc.exe  -c -mcpu=cortex-m3 -gdwarf-2 -I . Startup.c -o Startup.o
arm-none-eabi-ld.exe   -T  Linker_Script.ld  -Map=output.map Main.o Startup.o  -o  LAB2_ARM_CORTEX-M3.elf

arm-none-eabi-objcopy.exe  -O binary LAB2_ARM_CORTEX-M3.elf LAB2_ARM_CORTEX-M3.bin
------------------Build is done------------------

diesel@DESKTOP-ET889OF MINGW32 ~/Desktop/master embedded systems diploma/Unit 3 ( Embedded C )/Assignment/
HW 3/LAB2
$
```


C:\Users\diesel\Desktop\master embedded systems diploma\Unit 3 (Embedded C)\Assignment\HW 3\Lab (startup.c)\o...
File Edit Selection Find View Goto Tools Project Preferences Help
Linker_Script.ld
Startup.c
output.map
Memory Configuration
Name Origin Length Attributes
flash 0x0000000008000000 0x0000000000020000 xr
sram 0x0000000020000000 0x0000000000005000 xrw
default 0x0000000000000000 0xffffffffffffffff
Linker script and memory map
.text 0x0000000008000000 0x144
(.vectors)
.vectors 0x0000000008000000 0x38 Startup.o
0x0000000008000000 Vector
(.text)
.text 0x0000000008000038 0x7c Main.o
0x0000000008000038 main
.text 0x00000000080000b4 0x90 Startup.o
0x00000000080000b4 IRQ0_Handler
0x00000000080000b4 PendSV_Handler
0x00000000080000b4 NMI_Handler
0x00000000080000b4 Sys_Tick_Handler
0x00000000080000b4 Bus_Fault_Handler
0x00000000080000b4 MM_Fault_Handler
0x00000000080000b4 SV_call_Handler
0x00000000080000b4 Default_Handler
0x00000000080000b4 IRQ2_Handler
0x00000000080000b4 Debug_reserved_Handler
0x00000000080000b4 IRQ1_Handler
0x00000000080000b4 Usage_Fault_Handler
0x00000000080000b4 Hard_Fault_Handler
0x00000000080000c0 Reset_Handler
0x0000000008000144 _E_TEXT = .
.glue_7 0x0000000008000144 0x0
.glue_7 0x0000000008000144 0x0 linker stubs
.glue_7t 0x0000000008000144 0x0
.glue_7t 0x0000000008000144 0x0 linker stubs
.vfp11_veneer 0x0000000008000144 0x0
.vfp11_veneer 0x0000000008000144 0x0 linker stubs
.v4_bx 0x0000000008000144 0x0
.v4_bx 0x0000000008000144 0x0 linker stubs
.iplt 0x0000000008000144 0x0
.iplt 0x0000000008000144 0x0 Main.o
.data 0x0000000020000000 0x8 load address 0x0000000008000144
0x0000000020000000 _S_DATA = .
(.data)
.data 0x0000000020000000 0x8 Main.o
0x0000000020000000 g_variables

```
                0x0000000020000000                g_variables
                0x0000000020000004                R_ODR
 .data          0x0000000020000008        0x0 Startup.o
                0x0000000020000008                . = ALIGN (0x4)
                0x0000000020000008                _E_DATA = .

.igot.plt       0x0000000020000008        0x0 load address 0x000000000800014c
 .igot.plt      0x0000000020000008        0x0 Main.o

.bss            0x0000000020000008     0x1000 load address 0x000000000800014c
                0x0000000020000008                _S_BSS = .
 *(.bss)
 .bss           0x0000000020000008        0x0 Main.o
 .bss           0x0000000020000008        0x0 Startup.o
                0x0000000020000008                . = ALIGN (0x4)
                0x0000000020000008                _E_BSS = .
                0x0000000020001008                . = (. + 0x1000)
 *fill*         0x0000000020000008     0x1000
                0x0000000020001008                _stack_top = .

.rodata         0x000000000800014c        0x3
 *(.rodata*)
 .rodata        0x000000000800014c        0x3 Main.o
                0x000000000800014c                const_variables
LOAD Main.o
LOAD Startup.o
OUTPUT(LAB2_ARM_CORTEX-M3.elf elf32-littlearm)

.rel.dyn        0x0000000008000150        0x0
 .rel.iplt      0x0000000008000150        0x0 Main.o

.debug_info     0x0000000000000000      0x343
 .debug_info    0x0000000000000000      0x19d Main.o
 .debug_info    0x000000000000019d      0x1a6 Startup.o

.debug_abbrev   0x0000000000000000      0x1c0
 .debug_abbrev  0x0000000000000000       0xea Main.o
 .debug_abbrev  0x00000000000000ea       0xd6 Startup.o

.debug_loc      0x0000000000000000       0xb4
 .debug_loc     0x0000000000000000       0x38 Main.o
 .debug_loc     0x0000000000000038       0x7c Startup.o

.debug_aranges  0x0000000000000000       0x40
 .debug_aranges
                0x0000000000000000       0x20 Main.o
 .debug_aranges
                0x0000000000000020       0x20 Startup.o

.debug_line     0x0000000000000000      0x3d3
 .debug_line    0x0000000000000000      0x1df Main.o
 .debug_line    0x00000000000001df      0x1f4 Startup.o

.debug_str      0x0000000000000000      0x21f
 .debug_str     0x0000000000000000      0x196 Main.o
                                        0x1d3 (size before relaxing)
 .debug_str     0x0000000000000196       0x89 Startup.o
                                        0x204 (size before relaxing)
```

```
113
114   .comment        0x0000000000000000      0x7b
115    .comment       0x0000000000000000      0x7b Main.o
116                                           0x7c (size before relaxing)
117    .comment       0x000000000000007b      0x7c Startup.o
118
119   .ARM.attributes
120                   0x0000000000000000      0x33
121    .ARM.attributes
122                   0x0000000000000000      0x33 Main.o
123    .ARM.attributes
124                   0x0000000000000033      0x33 Startup.o
125
126   .debug_frame    0x0000000000000000      0x7c
127    .debug_frame   0x0000000000000000      0x2c Main.o
128    .debug_frame   0x000000000000002c      0x50 Startup.o
129
```

now using –gdwarf-2 to debug on Proteus accordding to the following concept.



The previous lab was by startup.c and I repeat it but with startup.s and here it but without copying .data from FLASH to SRAM and initializing .bss with 0.