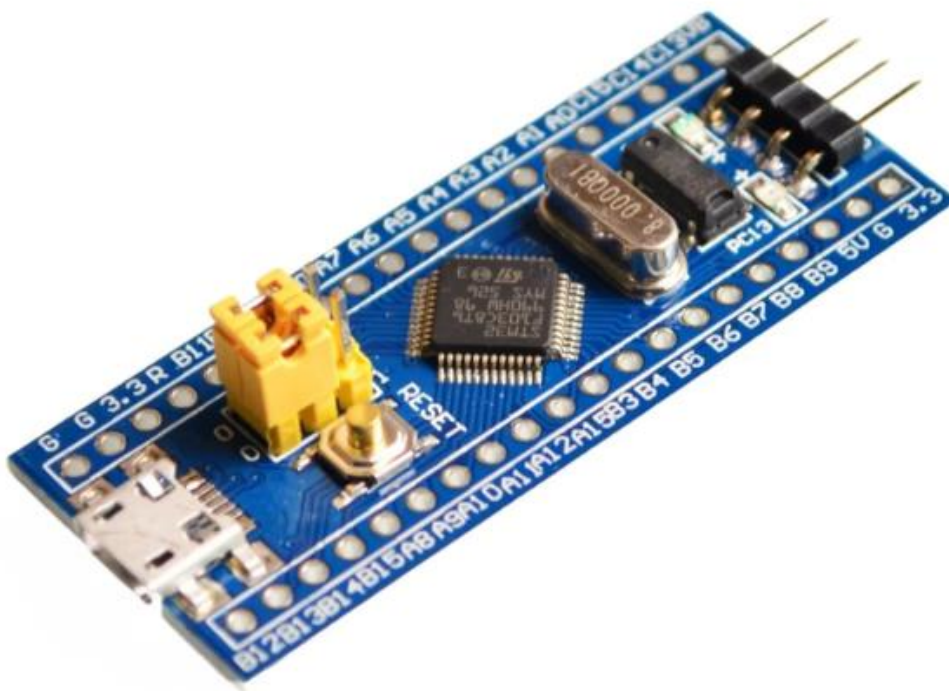


---

Embedded C Lab 2 Report : write bare-metal software to toggle a led connecting to GPIO portA13 on STM32F103C8T6 board.

ARM CORTEX-M3 STM32F103C8T6



---

By\ Eng . Belal Hani Abu Sabha

## Required Modules After Reading Datasheet:

1-RCC (Reset and Clock Control) Module :  
is necessary because GPIO has disabled clock by default .

Base address : 0x40021000

Registers : APB2ENR Register with offset 0x18

Enable bit : bit number 2

2-GPIO(general purpose input/output)  
Module :

I am working with GPIO portA

Base address : 0x40010800

Registers : A- CHR Register with offset 0x04  
and we need to write 2 on byte number 5 to  
configure PIN13 Mode

B- ODR Register with offset 0x0C to write 0/1  
on pin 13 .

## Startup Process:

when power is on to the MCU the PC value will be 0 which mapped to 0x08000000 address automatically and in this address contains a vector table and the first four bytes at address 0x08000000 contain address of Stack Pointer then address 0x08000004 contains Reset Handler Address then next address contains next handler from the vector table.

## Notes :-

- 1- SP is loaded automatically : this feature only for Cortex-M3 a
- 2- We can write a startup in C language (startup.c) not only in Assembly language (startup.s) .
- 3- in linker file , The ALIGN(4) command ensure memory alignment for more efficiency .
- 4- use -gdwarf-2 debug to simulate code on proteus

# Main.c:

```
Embedded System\c_source\codes_github\Master_Embedded_Systems\Embedded_C\Assignment 3 - Lab 2(using startup.c\main.c) - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
main.c
/*
 * @file      main.c
 * @author    Eng - Belal Hani Abu Sabha
 * @brief     toggle led - This program toggles an LED connected to pin PA13 on an ARM Cortex-M3
 * @Created on Sep 23, 2025
 */

// Create a typedef for volatile unsigned 32-bit integer
// volatile tells compiler this value may change outside program execution (in hardware)

typedef volatile unsigned int vuint32_t;
typedef unsigned int uint32_t;

// Define register addresses for Reset and Clock Control (RCC) peripheral
#define RCC_BASE 0x40021000 //Base address of RCC peripheral
#define RCC_APB2ENR_REG *(vuint32_t*)(RCC_BASE+0x18) // APB2 peripheral clock enable register

// Define register addresses for GPIO Port A peripheral

#define GPIO_PORTA_BASE 0x40010800 // Base address of GPIO Port A
#define GPIO_PORTA_CRH_REG *(vuint32_t*)(GPIO_PORTA_BASE+0x04)
#define GPIO_PORTA_ODR_REG *(vuint32_t*)(GPIO_PORTA_BASE+0x0C) // Output data register

#define RCC_IO_PORTA_ENABLE (1<<2) //assign 1 to bit number 2
#define GPIO_PORT_A13 (1UL<<13) //assign 1 to bit number 13

uint32_t Global_Var[2] = {10,20};
uint32_t const Constant_Var [2] = {1,2};

// delay func
void delay(void);

extern void Usage_fault_handler(void){
}
extern void MM_fault_handler(void){
}
}
```

```

43 main
44
45 /* -----another solution to W/R on PortA13 using union and structure bit field-----
46
47 // Create a union typedef for accessing each bits of ODR register
48
49 typedef union{
50     vuint32_t _32_bits; // ensure we can access all 32 bits
51     struct // struct bit filed to access every bit
52     {
53         vuint32_t reserved:13; //reserved bit 0 to bit 12
54         vuint32_t PIN_no_13:1; // bit 13 control output state
55     }Spin;
56 }U_ODR;
57
58 // Create a pointer to the ODR register
59
60 volatile U_ODR* Ptr_ODR = (volatile U_ODR*)(GPIO_PORTA_BASE+0x0C);
61
62 //----- */
63
64 int main(void)
65 {
66     RCC_APB2ENR_REG |=RCC_IO_PORTA_ENABLE; // XOR with 0000 0000 0000 0010 (assign 1 to bit 2) to enable RCC Clock
67
68     // Configure PortA13 as output
69     GPIO_PORTA_CRH_REG &=0xf00ffff; // Clear bits 20-23
70     GPIO_PORTA_CRH_REG |=0x00200000; // assign 1 to bits to bit 21
71
72     // infinite loop
73     while(1){
74         Ptr_ODR->Spin.PIN_no_13=1; // OUTPUT =1 , led is on until end of loop
75
76         GPIO_PORTA_ODR_REG |= GPIO_PORTA13; // OUTPUT =1 , led is on until end of loop
77         delay(); // call delay func
78         Ptr_ODR->Spin.PIN_no_13=0; // OUTPUT =0 , led is off until end of loop
79
80         GPIO_PORTA_ODR_REG &= ~GPIO_PORTA13; // OUTPUT =0 , led is off until end of loop
81         delay(); // call delay func
82
83
84
85     }
86     return 0;
87 }
88
89 // delay function
90 void delay(void) {
91     for(vuint32_t i = 0; i < 500; i++);
92 }
93
94

```

Line 41, Column 1

# Startup.c:

```
D:\Embedded System\C source\codes_github\Master_Embedded_Systems\Embedded C\Assignment 3 - Lab 2(using startup.c\startup.c - Sublime Text (UNREGISTERED))
File Edit Selection Find View Goto Tools Project Preferences Help

main.c startup.c

1 /*
2  * @file startup.c
3  * @author Eng - Belal Hani Abu Sabha
4  * @Created on Sep 23, 2025
5  */
6
7 typedef unsigned int uint32_t;
8 typedef unsigned char uint8_t;
9
10 #define Start_Stack_SP 0x20001000
11
12 extern int main(void);
13 void reset_handler(void);
14 void default_handler(void);
15 void WFI_handler(void) __attribute__((weak, alias("default_handler")));
16 void H_fault_handler(void) __attribute__((weak, alias("default_handler")));
17 void MMIO_fault_handler(void) __attribute__((weak, alias("default_handler")));
18 void Bus_fault_handler(void) __attribute__((weak, alias("default_handler")));
19 void Usage_fault_handler(void) __attribute__((weak, alias("default_handler")));
20
21 uint32_t vectors[] __attribute__((section(".vectors"))) = {
22     Start_Stack_SP,
23     (uint32_t)&reset_handler,
24     (uint32_t)&default_handler,
25     (uint32_t)&WFI_handler,
26     (uint32_t)&H_fault_handler,
27     (uint32_t)&MMIO_fault_handler,
28     (uint32_t)&Bus_fault_handler,
29     (uint32_t)&Usage_fault_handler,
30 };
31
32 extern uint32_t End_Text;
33 extern uint32_t Start_Data;
34 extern uint32_t End_Data;
35 extern uint32_t Start_Bss;
36 extern uint32_t End_Bss;
37 extern uint32_t Stack_Top;
38
39 void reset_handler(void) {
40     // copying data from flash to sram
41     uint32_t Data_Size = (uint8_t*)&End_Data - (uint8_t*)&Start_Data;
42     uint8_t *Src = (uint8_t*)&End_Text;
43     uint8_t *Des = (uint8_t*)&Start_Data;
44     for (int i = 0; i < Data_Size; ++i)
45     {
46         *((uint8_t*)Des++) = *((uint8_t*)Src++);
47     }
48
49     // create .bss section and init with zero
50     uint32_t Bss_Size = (uint8_t*)&End_Bss - (uint8_t*)&Start_Bss;
51
52     Des = (uint8_t*)&Start_Bss;
53     for (int i = 0; i < Bss_Size; ++i)
54     {
55         *Des++ = 0;
56     }
57
58     // go to main in main.c file
59     main();
60 }
61
62 void default_handler(void) {
63     reset_handler();
64 }
65
66 }
```

# Startup.s:

```
D:\Embedded System\C course\codes_github\Master_Embedded_Systems\Embedded C\Assignment 3 - Lab 2\using startup.s\startup.s - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
main.c startup.c startups
1  /*
2  * @file      startup.s
3  * @author    Eng - Belal Hani Abu Sabha
4  * @Created on Sep 22, 2025
5  */
6
7
8  // telling assembler to create a section with name vectors
9  .section .vectors
10
11  .word 0x20001000 // stack top address
12  .word _reset    // Reset
13  .word vector_handler //NMI
14  .word vector_handler //Hard Fault
15  .word vector_handler //MM Fault
16  .word vector_handler //Bus Fault
17  .word vector_handler //Usage Fault
18  .word vector_handler //Reserved
19  .word vector_handler //Reserved
20  .word vector_handler //Reserved
21  .word vector_handler //Reserved
22  .word vector_handler //SV Call
23  .word vector_handler //Debug Reserved
24  .word vector_handler //Reserved
25  .word vector_handler //PendSv
26  .word vector_handler //SysTick
27  .word vector_handler //IRQ0
28  .word vector_handler //IRQ1
29  .word vector_handler //IRQ2
30  .word vector_handler //..... to IRQ67
31
32  .section .text
33  _reset:
34  bl main // main function is called using branch
35  b .    // branch itself
36
37  .thumb_func //use 16 bits and 32 bits instruction if available
38
39  vector_handler: // every time we called a vector we called _reset which called main function again
40  b _reset
41
```

# Linker.ld:

```
D:\Embedded System\C course\codes_github\Master_Embedded_Systems\Embedded C\Assignment 3 - Lab 2\using startup.c\linker.ld - Sublime Text (UNREGI
File Edit Selection Find View Goto Tools Project Preferences Help
main.c startup.c startup.s linker.ld
1 /*
2  * @file    linker.ld
3  * @author  Eng - Belal Hani Abu Sabha
4  * @Created on Sep 23, 2025
5  */
6
7 MEMORY
8 {
9     flash(rx) : ORIGIN = 0x08000000, LENGTH = 128K
10    sram(rwx) : ORIGIN = 0x20000000, LENGTH = 20K
11 }
12
13 SECTIONS
14 {
15     .text : {
16         *(.vectors*)
17         *(.text*)
18         *(.rodata*)
19         End_Text = . ;
20     }>flash
21
22     .data : {
23         Start_Data = . ;
24
25         *(.data*)
26         End_Data = . ;
27     }>sram AT>flash
28
29     .bss : {
30         Start_Bss = . ;
31         *(.bss*)
32         . = ALIGN(4) ;
33         End_Bss = . ;
34         . = ALIGN(4) ;
35         . = . + 0x1000 ;
36         Stack_Top = . ;
37     }>sram
38 }
39
```



# Compile and link using MakeFile:

```
MINGW64:/d/Embedded System/C course/codes_github/Master_Embedded_Systems/Embedded C/Assignment 3 - Lab 2/using startup.c (main)
TUF@Belal MINGW64 /d/Embedded System/C course/codes_github/Master_Embedded_Systems/Embedded C/Assignment 3 - Lab 2/using startup.c (main)
$ make
arm-none-eabi-gcc.exe -c -gdwarf-2 -mcpu=cortex-m3 -I . main.c -o main.o
arm-none-eabi-gcc.exe -c -gdwarf-2 -mcpu=cortex-m3 -I . startup.c -o startup.o
arm-none-eabi-ld.exe -T linker.ld main.o startup.o -o cortex-M3.elf -Map=mab_file.map
arm-none-eabi-objcopy.exe -O binary cortex-M3.elf cortex-M3.bin
build is done

TUF@Belal MINGW64 /d/Embedded System/C course/codes_github/Master_Embedded_Systems/Embedded C/Assignment 3 - Lab 2/using startup.c (main)
$
```

# Symbols for \*.o file and .elf file:

```
TUF@Belal MINGW64 /d/Embedded System/C course/codes_github/Master_Embedded_Systems/Embedded C/Assignment 3 - Lab 2/using startup.c (main)
$ arm-none-eabi-nm.exe startup.o
00000088 W Bus_fault_handler
00000088 T default_handler
          U End_Bss
          U End_Data
          U End_Text
00000088 W H_fault_handler
          U main
00000088 W MM_fault_handler
00000088 W NMI_handler
00000000 T reset_handler
          U Start_Bss
          U Start_Data
00000088 W Usage_fault_handler
00000000 D vectors

TUF@Belal MINGW64 /d/Embedded System/C course/codes_github/Master_Embedded_Systems/Embedded C/Assignment 3 - Lab 2/using startup.c (main)
$ arm-none-eabi-nm.exe main.o
00000000 R Constant_Var
00000070 T delay
00000000 D Global_Var
00000018 T main
0000000c T MM_fault_handler
00000000 T Usage_fault_handler

TUF@Belal MINGW64 /d/Embedded System/C course/codes_github/Master_Embedded_Systems/Embedded C/Assignment 3 - Lab 2/using startup.c (main)
$ arm-none-eabi-nm.exe cortex-M3.elf
08000140 W Bus_fault_handler
0800014c T Constant_Var
08000140 T default_handler
08000090 T delay
20000008 B End_Bss
20000008 D End_Data
08000154 T End_Text
20000000 D Global_Var
08000140 W H_fault_handler
08000038 T main
0800002c T MM_fault_handler
08000140 W NMI_handler
080000b8 T reset_handler
20001008 B Stack_Top
20000008 B Start_Bss
20000000 D Start_Data
08000020 T Usage_fault_handler
08000000 T vectors
```

# Mabfile.mab:

```
1
2 Memory Configuration
3
4 Name          Origin          Length          Attributes
5 flash         0x08000000      0x00020000      xr
6 sram          0x20000000      0x00005000      xrw
7 *default*     0x00000000      0xffffffff
8
9 Linker script and memory map
10
11
12 .text         0x08000000      0x154
13 *(.vectors*)
14 .vectors      0x08000000      0x20 startup.o
15              0x08000000      vectors
16 *(.text*)
17 .text         0x08000020      0x98 main.o
18              0x08000020      Usage_fault_handler
19              0x0800002c      MM_fault_handler
20              0x08000038      main
21              0x08000090      delay
22 .text         0x080000b8      0x94 startup.o
23              0x080000b8      reset_handler
24              0x08000140      Bus_fault_
25              0x08000140      deafuld_handler
26              0x08000140      H_fault_handler
27              0x08000140      NMI_handler
28 *(.rodata*)
29 .rodata       0x0800014c      0x8 main.o
30              0x0800014c      Constant_Var
31              0x08000154      End_Text = .
32
33 .glue_7       0x08000154      0x0
34 .glue_7       0x08000154      0x0 linker stubs
35
36 .glue_7t      0x08000154      0x0
37 .glue_7t      0x08000154      0x0 linker stubs
38
39 .vfp11_veneer 0x08000154      0x0
40 .vfp11_veneer 0x08000154      0x0 linker stubs
41
42 .v4_bx        0x08000154      0x0
43 .v4_bx        0x08000154      0x0 linker stubs
44
45 .iplt         0x08000154      0x0
46 .iplt         0x08000154      0x0 main.o
47
48 .rel.dyn      0x08000154      0x0
49 .rel.iplt     0x08000154      0x0 main.o
```

Line 5, Column 52

**CHM3 Registers - UI**

Register	Value	Privileged	Write Thread
R0	00000000	0	0
R1	00000000	0	0
R2	00000000	0	0
R3	00000000	0	0
R4	00000000	0	0
R5	00000000	0	0
R6	00000000	0	0
R7	00000000	0	0
R8	00000000	0	0
R9	00000000	0	0
R10	00000000	0	0
R11	00000000	0	0
R12	00000000	0	0
R13	00000000	0	0
R14	00000000	0	0
R15	00000000	0	0
R16	00000000	0	0
R17	00000000	0	0
R18	00000000	0	0
R19	00000000	0	0
R20	00000000	0	0
R21	00000000	0	0
R22	00000000	0	0
R23	00000000	0	0
R24	00000000	0	0
R25	00000000	0	0
R26	00000000	0	0
R27	00000000	0	0
R28	00000000	0	0
R29	00000000	0	0
R30	00000000	0	0
R31	00000000	0	0
R32	00000000	0	0
R33	00000000	0	0
R34	00000000	0	0
R35	00000000	0	0
R36	00000000	0	0
R37	00000000	0	0
R38	00000000	0	0
R39	00000000	0	0
R40	00000000	0	0
R41	00000000	0	0
R42	00000000	0	0
R43	00000000	0	0
R44	00000000	0	0
R45	00000000	0	0
R46	00000000	0	0
R47	00000000	0	0
R48	00000000	0	0
R49	00000000	0	0
R50	00000000	0	0
R51	00000000	0	0
R52	00000000	0	0
R53	00000000	0	0
R54	00000000	0	0
R55	00000000	0	0
R56	00000000	0	0
R57	00000000	0	0
R58	00000000	0	0
R59	00000000	0	0
R60	00000000	0	0
R61	00000000	0	0
R62	00000000	0	0
R63	00000000	0	0
R64	00000000	0	0
R65	00000000	0	0
R66	00000000	0	0
R67	00000000	0	0
R68	00000000	0	0
R69	00000000	0	0
R70	00000000	0	0
R71	00000000	0	0
R72	00000000	0	0
R73	00000000	0	0
R74	00000000	0	0
R75	00000000	0	0
R76	00000000	0	0
R77	00000000	0	0
R78	00000000	0	0
R79	00000000	0	0
R80	00000000	0	0
R81	00000000	0	0
R82	00000000	0	0
R83	00000000	0	0
R84	00000000	0	0
R85	00000000	0	0
R86	00000000	0	0
R87	00000000	0	0
R88	00000000	0	0
R89	00000000	0	0

Thank You