

Lab

❖ Toggle LED By STM32 and processor CortexM3 :

➤ Main.c :

■

```
/* By Abdelrahman Matarawy */
#include "Platform_Types.h"

/* Register Addresses */
#define RCC_Base_Address 0x40021000
#define GPIO_PortA 0x40010800

/* Bit in Register RCC */
#define RCC_APB2ENR *((vuint32_t*) (RCC_Base_Address + 0x18))
/* Bits In Register GPIO */
#define GPIO_CRH *((vuint32_t*) (GPIO_PortA + 0x04))
#define GPIO_ODR *((vuint32_t*) (GPIO_PortA + 0x0c))
/* To Write in all register bits or specific bit */
typedef union{
    vuint32_t all_fields;
    struct{
        vuint32_t reserved : 13;
        vuint32_t p_13 : 1;
    }Pin;
}R_ODR_T;

volatile R_ODR_T * R_ODR = ((volatile R_ODR_T *) (GPIO_PortA + 0x0c));
uint8_t g_variables[3] = {1,2,3}; /* Saved in .data section */
uint8_t const const_variables[3] = {1,2,3}; /* Saved in .rodata section */
uint8_t uint8_t_variable[3]; /* Saved in .bss section */

/* For check weak feature */
extern void NMI_Handler()
{
}
extern void Bus_Fault()
{
}

int main(void)
{
    int i;
    RCC_APB2ENR |= (1<<2); /* Write on bit 2 logic high
    GPIO_ODR |= (1<<13); /* Write on bit 13 logic high
    GPIO_CRH &= 0xFF0FFFFFFF;
    GPIO_CRH |= 0x00200000;
    while(1)
    {
        /* For Applying Logic 1 for seconds and toggle it with logic low */
        R_ODR->Pin.p_13 = 1;
        for(i = 0; i < 5000; i++);
        R_ODR->Pin.p_13 = 0;
        for(i = 0; i < 5000; i++);
    }
}
```

➤ StartUp.s :

```
/* By Abdelrahman Matarawy */
/* StartUp for CortexM3 */

/* Interrupt Vector Table */
.section .vectors

.word 0x20001000 /* Address added for stack top */
.word _reset /* 1 Reset Handler */
.word vector_handler /* 2 NMI */
.word vector_handler /* 3 Hard fault */
.word vector_handler /* 4 MM fault */
.word vector_handler /* 5 Bus fault */
.word vector_handler /* 6 usage fault */
.word vector_handler /* 7 Reserved */
.word vector_handler /* 8 Reserved */
.word vector_handler /* 9 Reserved */
.word vector_handler /* 10 Reserved */
.word vector_handler /* 11 SV call */
.word vector_handler /* 12 Debug reserved */
.word vector_handler /* 13 Reserved */
.word vector_handler /* 14 pendSV */
.word vector_handler /* 15 SysTick */
.word vector_handler /* 16 IRQ0 */
.word vector_handler /* 17 IRQ1 */
.word vector_handler /* 18 IRQ2 */
.word vector_handler /* 19 ... */

.section .text
_reset:
    bl main
    b .

/* For activate 16 bits mode */
.thumb_func

vector_handler:
    b _reset
```

➤ StartUp.c :

```
/* StartUp.c For CortexM3
By Eng.Abdelrahman Mataravy
*/

#include "Platform_Types.h"
int i;
extern unsigned int stack_top;
extern int main(void);

void Reset_Handler() ;

void Default_Handler()
{
    Reset_Handler();
}

void NMI_Handler() __attribute__((weak, alias("Default_Handler")));

void H_Fault_Handler() __attribute__((weak, alias("Default_Handler")));

void MM_Fault_Handler() __attribute__((weak, alias("Default_Handler")));

void Bus_Fault() __attribute__((weak, alias("Default_Handler")));

void Usage_Fault_Handler() __attribute__((weak, alias("Default_Handler")));
/* Array for IVT */

uint32_t vectors[] __attribute__((section(".vectors")))={
    (uint32_t) &stack_top,
    (uint32_t) &Reset_Handler,
    (uint32_t) &NMI_Handler,
    (uint32_t) &H_Fault_Handler,
    (uint32_t) &MM_Fault_Handler,
    (uint32_t) &Bus_Fault,
    (uint32_t) &Usage_Fault_Handler,
};

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;
void Reset_Handler()
{
    /* Copy data section from flash to ram */
    uint32_t Data_Size = (uint8_t*)&_E_data - (uint8_t*)&_S_data;
    uint8_t* source = (uint8_t*)&_E_text;
    uint8_t* destination = (uint8_t*)&_S_data;
    for(i = 0 ; i < Data_Size ; i++ )
    {
        *((uint8_t*)source++) = *((uint8_t*)destination++);
    }

    /* Init bss section with zeros in ram */
    uint32_t Bss_Size = (uint8_t*)&_E_bss - (uint8_t*)&_S_bss;
    destination = (uint8_t*)&_S_bss;
    for(i = 0 ; i < Data_Size ; i++ )
    {
        *((uint8_t*)source++) = (uint8_t)0;
    }

    /* jump to main */
    main();
}
```

➤ Linker Script:

```
/* Linker_Script for CortexM3 */
/* By Abdelrahman Matarawy */

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

SECTIONS
{
    .text : {
        *(.vectors*)
        *(.text*)
        *(.rodata)
        . = ALIGN(4);
        _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
}
```

➤ Make file:

■

```
##@Copy@riter: Matarawy

CC=arm-none-eabi-
CFLAGS=-mcpu=cortex-m3 -gdwarf-2
INCS=-I .
LIBS=
SRC=$(wildcard *.c)
OBJ=$(SRC:.c=.o)
As=$(wildcard *.s)
AsOBJ=$(As:.s=.o)
Project_Name= learn-in-depth-cortex-m3

all: $(Project_Name).bin
    @echo "===== Build is Done ====="

%.o: %.s
    $(CC) as.exe $(CFLAGS) -mthumb $< -o $@

%.o: %.c
    $(CC) gcc.exe -c $(INCS) $(CFLAGS) -mthumb $< -o $@

$(Project_Name).elf: $(OBJ) $(AsOBJ)
    $(CC) ld.exe -T linker_script.ld $(LIBS) $(OBJ) $(AsOBJ) -o $@ -Map=Map_file.map

$(Project_Name).bin: $(Project_Name).elf
    $(CC) objcopy.exe -O binary $< $@

clean_all:
    rm *.o *.elf *.bin

clean:
    rm *.elf *.bin
```

Before Weak and alias :

```
/* Bit in Register RCC */
#define RCC_APB2ENR      *((vuint32_t*) (RCC_Base_Address + 0x18))
/* Bits In Register GPIO */
#define GPIO_CRH          *((vuint32_t*) (GPIO_PortA + 0x04))
#define GPIO_ODR          *((vuint32_t*) (GPIO_PortA + 0x0c))
/* To Write in all register bits or specific bit */
typedef union{
    vuint32_t all_fields;
    struct{
        vuint32_t reserved : 13;
        vuint32_t p_13 : 1;
    }Pin;
}R_ODR_T;

volatile R_ODR_T * R_ODR = ((volatile R_ODR_T *) (GPIO_PortA + 0x0c));
uint8_t g_variables[3] = {1,2,3}; /* Saved in .data section */
uint8_t const const_variables[3] = {1,2,3}; /* Saved in .rodata section */
uint8_t uninit_Gvariable[3]; /* Saved in .bss section */

int main(void)
{
    int i;
    RCC_APB2ENR |= (1<<2); /* Write on bit 2 logic high
    GPIO_ODR |= (1<<13); /* Write on bit 13 logic high
    GPIO_CRH |= 0xFF0FFFFFFF;
    GPIO_CRH |= 0x00200000;
    while(1)
    {
        /* For Applying Logic 1 for seconds and toggle it with logic low */
        R_ODR->Pin.p_13 = 1;
        for(i = 0; i < 5000; i++);
        R_ODR->Pin.p_13 = 0;
        for(i = 0; i < 5000; i++);
    }
}
```

```
MINGW32/c/Users/abdel/Downloads/Mastering_embedded_system/github_repo/...
arm-none-eabi-objcopy.exe -O binary learn-in-depth-cortex-m3.elf learn-in-depth-cortex-m3.in
===== Build is Done =====

abdel@LAPTOP-3GVNE8GO MINGW32 ~/Downloads/Mastering_embedded_system/github_repo/Unit3_Embedded_C/Unit3/Lesson_3/Lesson3 (main)
$ arm-none-eabi-nm.exe learn-in-depth-cortex-m3.elf
08000028 W Bus_Fault
080000f4 T const_variables
08000028 T Default_Handler
080000fc D g_variables
08000028 W H_Fault_Handler
08000034 T main
08000028 W MM_Fault_Handler
08000028 W NMI_Handler
080000fc D R_ODR
0800001c T Reset_Handler
20000000 B uninit_Gvariable
08000028 W Usage_Fault_Handler
08000000 T vectors

abdel@LAPTOP-3GVNE8GO MINGW32 ~/Downloads/Mastering_embedded_system/github_repo/Unit3_Embedded_C/Unit3/Lesson_3/Lesson3 (main)
$ mingw32-make clean_all
```

After weak and alias :

```
/* Bit in Register RCC */
#define RCC_APB2ENR      *((vuint32_t*) (RCC_Base_Address + 0x18))
/* Bits In Register GPIO */
#define GPIO_CRH          *((vuint32_t*) (GPIO_PortA + 0x04))
#define GPIO_ODR          *((vuint32_t*) (GPIO_PortA + 0x0c))
/* To Write in all register bits or specific bit */
typedef union{
    vuint32_t all_fields;
    struct{
        vuint32_t reserved : 13;
        vuint32_t p_13 : 1;
    }Pin;
}R_ODR_T;

volatile R_ODR_T * R_ODR = ((volatile R_ODR_T *) (GPIO_PortA + 0x0c));
uint8_t g_variables[3] = {1,2,3}; /* Saved in .data section */
uint8_t const const_variables[3] = {1,2,3}; /* Saved in .rodata section */
uint8_t uninit_Gvariable[3]; /* Saved in .bss section */

/* For check weak feature */
extern void NMI_Handler()
{
}
extern void Bus_Fault()
{
}

int main(void)
{
    int i;
    RCC_APB2ENR |= (1<<2); /* Write on bit 2 logic high
    GPIO_ODR |= (1<<13); /* Write on bit 13 logic high
    GPIO_CRH |= 0xFF0FFFFFFF;
    GPIO_CRH |= 0x00200000;
    while(1)
    {
        /* For Applying Logic 1 for seconds and toggle it with logic low */
        R_ODR->Pin.p_13 = 1;
    }
}
```

```
MINGW32/c/Users/abdel/Downloads/Mastering_embedded_system/github_repo/...
arm-none-eabi-objcopy.exe -O binary learn-in-depth-cortex-m3.elf learn-in-depth-cortex-m3.in
===== Build is Done =====

abdel@LAPTOP-3GVNE8GO MINGW32 ~/Downloads/Mastering_embedded_system/github_repo/Unit3_Embedded_C/Unit3/Lesson_3/Lesson3 (main)
$ arm-none-eabi-nm.exe learn-in-depth-cortex-m3.elf
08000040 T Bus_Fault
0800010c T const_variables
08000028 T Default_Handler
08000114 D g_variables
08000028 W H_Fault_Handler
0800004c T main
08000028 W MM_Fault_Handler
08000034 T NMI_Handler
08000110 D R_ODR
0800001c T Reset_Handler
20000000 B uninit_Gvariable
08000028 W Usage_Fault_Handler
08000000 T vectors

abdel@LAPTOP-3GVNE8GO MINGW32 ~/Downloads/Mastering_embedded_system/github_repo/Unit3_Embedded_C/Unit3/Lesson_3/Lesson3 (main)
$
```

Copy data section and init bss section :

```

16 void NMI_Handler() __attribute__((weak, alias("Default_Handler")));
17
18 void H_Fault_Handler() __attribute__((weak, alias("Default_Handler")));
19
20 void MM_Fault_Handler() __attribute__((weak, alias("Default_Handler")));
21
22 void Bus_Fault_Handler() __attribute__((weak, alias("Default_Handler")));
23
24 void Usage_Fault_Handler() __attribute__((weak, alias("Default_Handler")));
25 /* Array for IVT */
26
27 uint32_t vectors[] __attribute__((section(".vectors")))={
28     (uint32_t) &stack_top,
29     (uint32_t) &Reset_Handler,
30     (uint32_t) &NMI_Handler,
31     (uint32_t) &H_Fault_Handler,
32     (uint32_t) &MM_Fault_Handler,
33     (uint32_t) &Bus_Fault_Handler,
34     (uint32_t) &Usage_Fault_Handler,
35 };
36
37 extern uint32_t _E_text;
38 extern uint32_t _S_data;
39 extern uint32_t _E_data;
40 extern uint32_t _S_bss;
41 extern uint32_t _E_bss;
42 void Reset_Handler()
43 {
44     /* Copy data section from flash to ram */
45     uint32_t Data_Size = (uint8_t*)&_E_data - (uint8_t*)&_S_data;
46     uint8_t* source = (uint8_t*)&_E_text;
47     uint8_t* destination = (uint8_t*)&_S_data;
48 }

```

Linker Script: learn-in-depth-cortex-m3.elf: file format elf32-littlearm

Idx	Name	Size	VMA	LMA	File off	Align
0	.text	000001f8	08000000	08000000	00000000	2**2
1	.data	00000008	20000000	080001f8	00010000	2**2
2	.bss	00001007	20000008	08000200	00010008	2**2
3	.debug_info	00000332	00000000	00000000	00010008	2**0
4	.debug_abbrev	000001bd	00000000	00000000	0001033a	2**0
5	.debug_loc	000000f4	00000000	00000000	000104f7	2**0
6	.debug_aranges	00000040	00000000	00000000	000105eb	2**0
7	.debug_line	000000ec	00000000	00000000	0001062b	2**0
8	.debug_str	000001ed	00000000	00000000	00010717	2**0
9	.comment	00000011	00000000	00000000	00010904	2**0
10	.ARM.attributes	00000033	00000000	00000000	00010915	2**0
11	.debug_frame	000000a8	00000000	00000000	00010948	2**2

When led on :

CM3 Source Code - U1

```

main.c
----- volatile R_ODR_T * R_ODR = ((volatile R_ODR_T *) (GPIO_PortA + 0x
uint8_t g_variables[3] = {1,2,3}; /* Saved in .data section */
uint8_t const const_variables[3] = {1,2,3}; /* Saved in .rodata sect
uint8_t uninit_gvariable[3]; /* Saved in .bss section */
-----
/* For check weak feature */
extern void NMI_Handler()
{
    800011C
    8000120
}
extern void Bus_Fault_Handler()
{
    8000128
    800012C
}
int main(void)
{
    8000134
    int i;
    RCC_APB2ENR |= (1<<2); // Write on bit 2 logic high
    GPIO_ODR |= (1<<13); // Write on bit 13 logic high
    GPIO_CRH |= 0xFFFFF;
    GPIO_CRH |= 0x00200000;
    while(1)
    {
        /* For Applying Logic 1 for seconds and toggle it wi
        R_ODR -> PPin.p_13 = 1;
        for(i = 0; i < 2000; i++);
        R_ODR -> PPin.p_13 = 0;
        for(i = 0; i < 2000; i++);
    }
}

```

CM3 Variables - U1

Name	Address	Value
vectors	08000000	dw0rd[7]
R_ODR	080001f8	0x4001080C
g_variables	080001fc	byte[3]
const_va...	080001f4	byte[3]
uninit_g...	20001004	byte[3]
i	20001000	8
i	BP+12 = ...	981