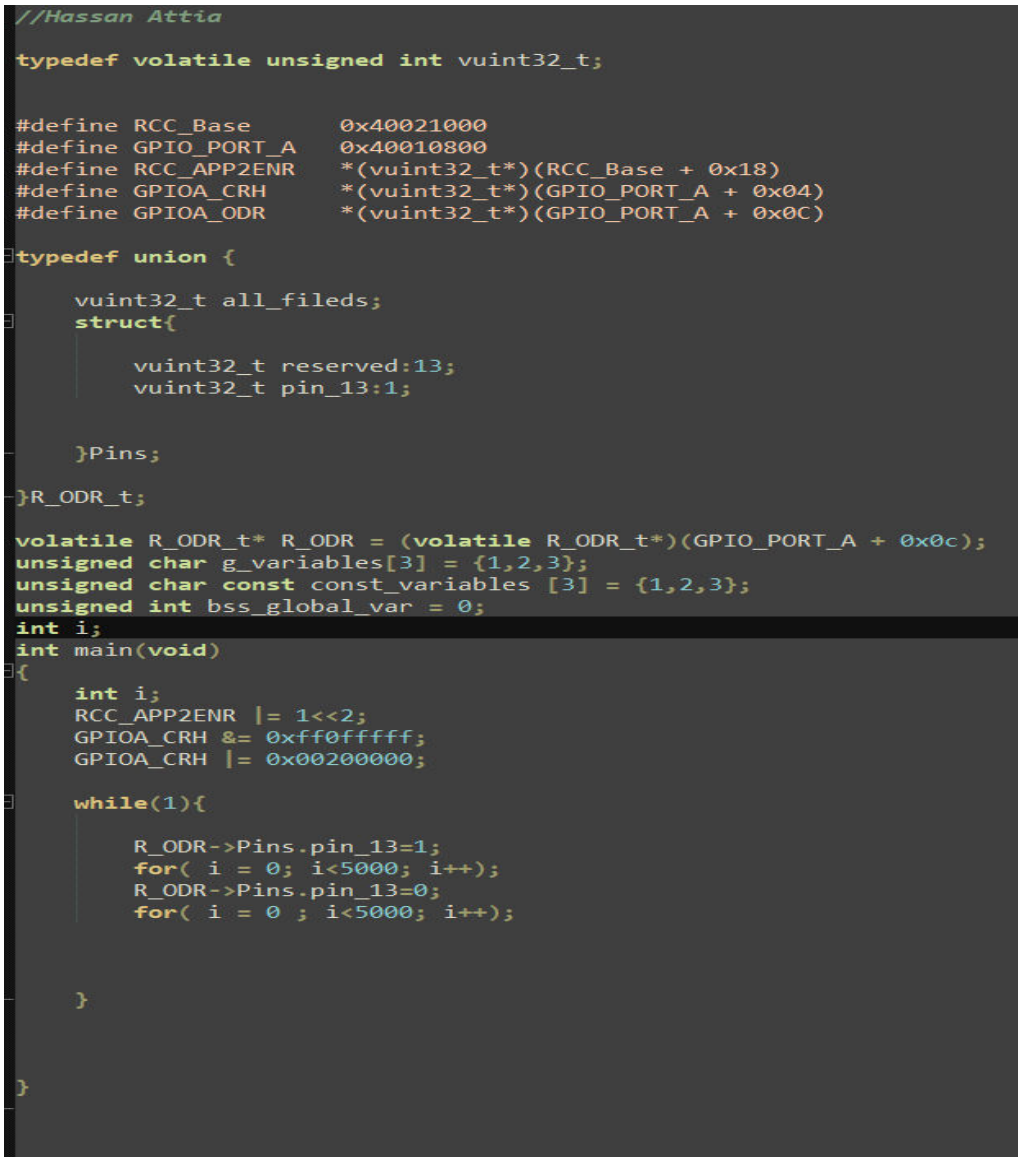
TOGGLING LED ON ARM CORTEX M3 PROCESSOR WITH STARTUP.C

AUTHOR: Hassan Attia

Brief:

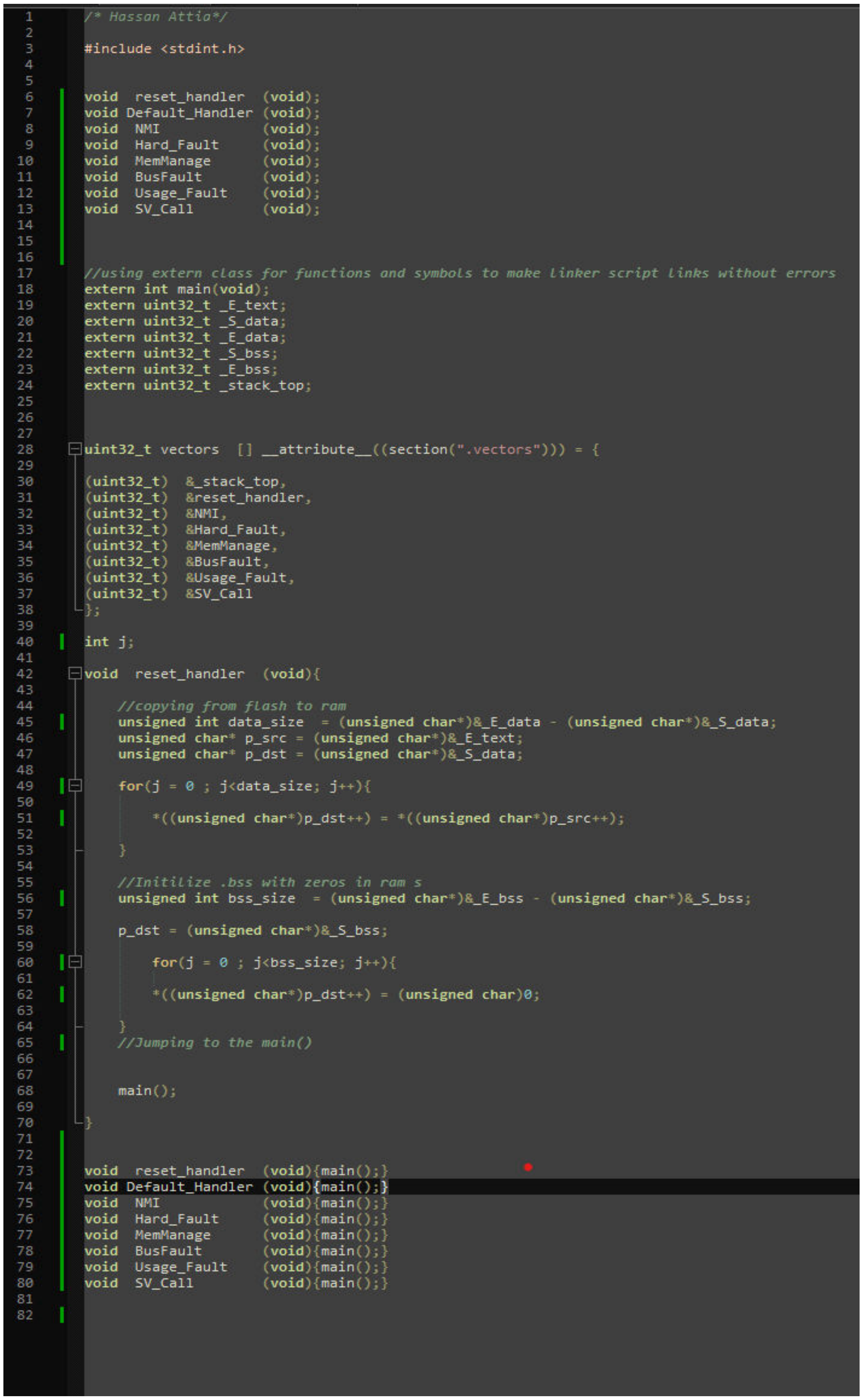
- I made a bare metal software that runs on STM32 microcontroller.
- STM 32 has ARM cortex M3 processor and it supports writing startup file with C code.
- I'll put some screenshots illustrating the whole processes which include(compiling the files, map file, proteus simulation, debugging, and weak and alias attributes).

1-Main.c



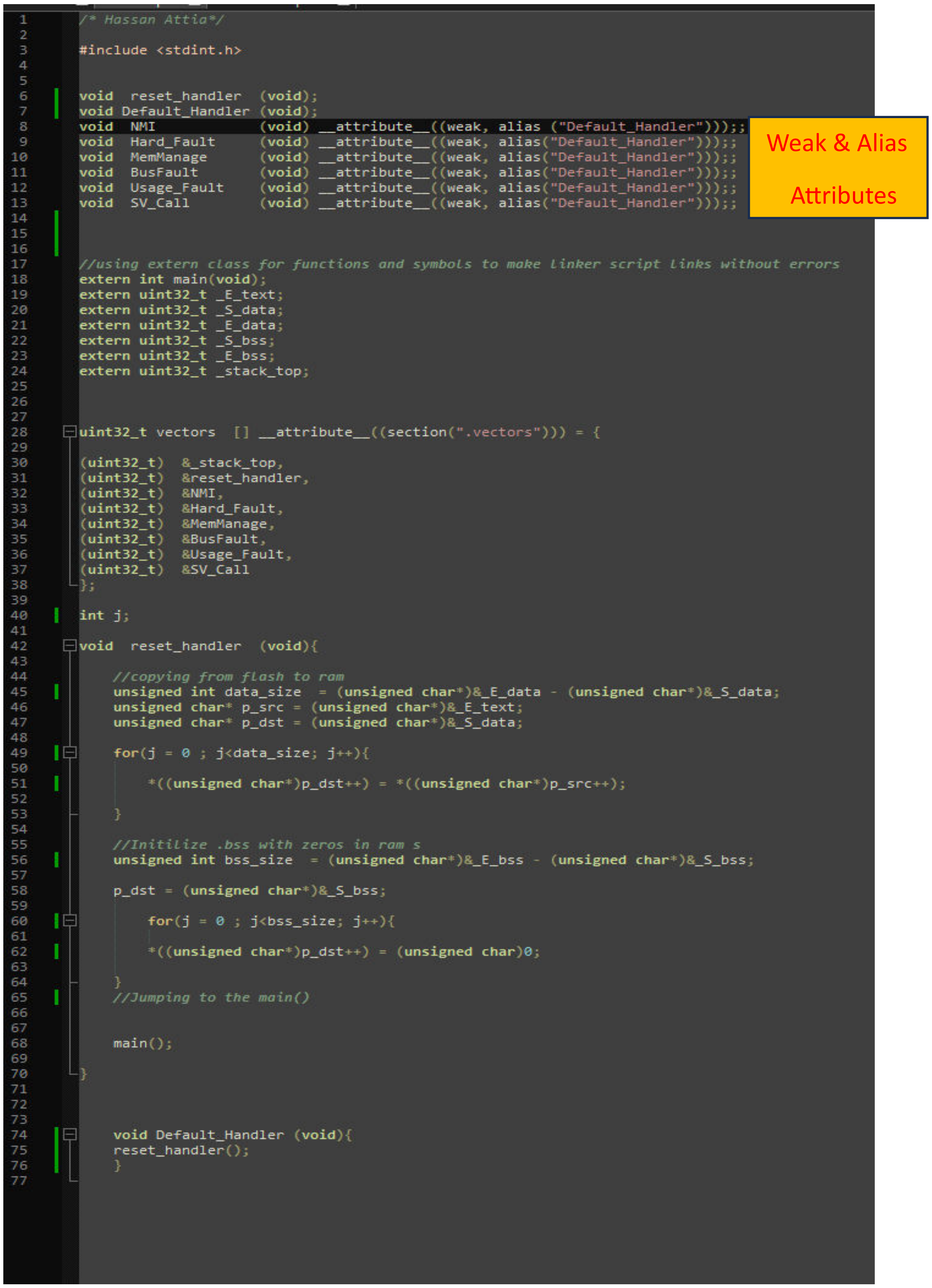
```c
//Hassan Attia

typedef volatile unsigned int vuint32_t;


#define RCC_Base        0x40021000
#define GPIO_PORT_A     0x40010800
#define RCC_APP2ENR     *(vuint32_t*)(RCC_Base + 0x18)
#define GPIOA_CRH       *(vuint32_t*)(GPIO_PORT_A + 0x04)
#define GPIOA_ODR       *(vuint32_t*)(GPIO_PORT_A + 0x0C)

typedef union {

    vuint32_t all_fileds;
    struct{

        vuint32_t reserved:13;
        vuint32_t pin_13:1;


    }Pins;

}R_ODR_t;

volatile R_ODR_t* R_ODR = (volatile R_ODR_t*)(GPIO_PORT_A + 0x0c);
unsigned char g_variables[3] = {1,2,3};
unsigned char const const_variables [3] = {1,2,3};
unsigned int bss_global_var = 0;
int i;
int main(void)
{
    int i;
    RCC_APP2ENR |= 1<<2;
    GPIOA_CRH &= 0xff0fffff;
    GPIOA_CRH |= 0x00200000;

    while(1){

        R_ODR->Pins.pin_13=1;
        for( i = 0; i<5000; i++);
        R_ODR->Pins.pin_13=0;
        for( i = 0 ; i<5000; i++);



    }



}
```

2-Startup.c before using weak and alias



```c
/* Hassan Attia*/

#include <stdint.h>


void  reset_handler  (void);
void Default_Handler (void);
void  NMI            (void);
void  Hard_Fault     (void);
void  MemManage      (void);
void  BusFault       (void);
void  Usage_Fault    (void);
void  SV_Call        (void);



//using extern class for functions and symbols to make linker script links without errors
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;



uint32_t vectors  [] __attribute__((section(".vectors"))) = {

(uint32_t)  &_stack_top,
(uint32_t)  &reset_handler,
(uint32_t)  &NMI,
(uint32_t)  &Hard_Fault,
(uint32_t)  &MemManage,
(uint32_t)  &BusFault,
(uint32_t)  &Usage_Fault,
(uint32_t)  &SV_Call
};

int j;

void  reset_handler  (void){

    //copying from flash to ram
    unsigned int data_size  = (unsigned char*)&_E_data - (unsigned char*)&_S_data;
    unsigned char* p_src = (unsigned char*)&_E_text;
    unsigned char* p_dst = (unsigned char*)&_S_data;

    for(j = 0 ; j<data_size; j++){

        *((unsigned char*)p_dst++) = *((unsigned char*)p_src++);

    }

    //Initilize .bss with zeros in ram s
    unsigned int bss_size  = (unsigned char*)&_E_bss - (unsigned char*)&_S_bss;

    p_dst = (unsigned char*)&_S_bss;

        for(j = 0 ; j<bss_size; j++){

        *((unsigned char*)p_dst++) = (unsigned char)0;

    }
    //Jumping to the main()


    main();

}


void  reset_handler  (void){main();}
void Default_Handler (void){main();}
void  NMI            (void){main();}
void  Hard_Fault     (void){main();}
void  MemManage      (void){main();}
void  BusFault       (void){main();}
void  Usage_Fault    (void){main();}
void  SV_Call        (void){main();}
```

3-Startup.c after using weak and alias:

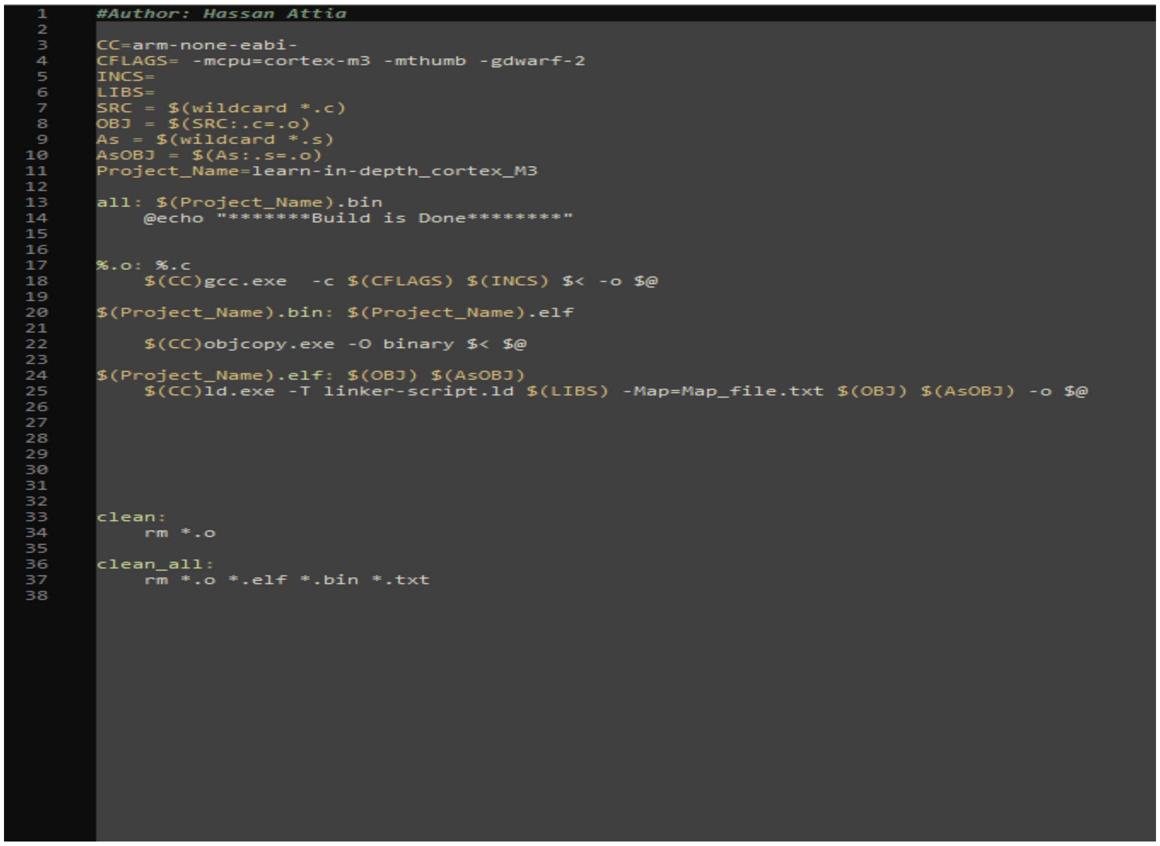


```c
/* Hassan Attia*/

#include <stdint.h>


void  reset_handler  (void);
void Default_Handler (void);
void  NMI            (void) __attribute__((weak, alias ("Default_Handler")));;
void  Hard_Fault     (void) __attribute__((weak, alias("Default_Handler")));;
void  MemManage      (void) __attribute__((weak, alias("Default_Handler")));;
void  BusFault       (void) __attribute__((weak, alias("Default_Handler")));;
void  Usage_Fault    (void) __attribute__((weak, alias("Default_Handler")));;
void  SV_Call        (void) __attribute__((weak, alias("Default_Handler")));;



//using extern class for functions and symbols to make linker script links without errors
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;



uint32_t vectors  [] __attribute__((section(".vectors"))) = {

(uint32_t)  &_stack_top,
(uint32_t)  &reset_handler,
(uint32_t)  &NMI,
(uint32_t)  &Hard_Fault,
(uint32_t)  &MemManage,
(uint32_t)  &BusFault,
(uint32_t)  &Usage_Fault,
(uint32_t)  &SV_Call
};

int j;

void  reset_handler  (void){

    //copying from flash to ram
    unsigned int data_size  = (unsigned char*)&_E_data - (unsigned char*)&_S_data;
    unsigned char* p_src = (unsigned char*)&_E_text;
    unsigned char* p_dst = (unsigned char*)&_S_data;

    for(j = 0 ; j<data_size; j++){

        *((unsigned char*)p_dst++) = *((unsigned char*)p_src++);

    }

    //Initilize .bss with zeros in ram s
    unsigned int bss_size  = (unsigned char*)&_E_bss - (unsigned char*)&_S_bss;

    p_dst = (unsigned char*)&_S_bss;

        for(j = 0 ; j<bss_size; j++){

        *((unsigned char*)p_dst++) = (unsigned char)0;

    }
    //Jumping to the main()


    main();

}



    void Default_Handler (void){
    reset_handler();
    }

```

Explaining Alias & Weak cross-tool chain attributes:

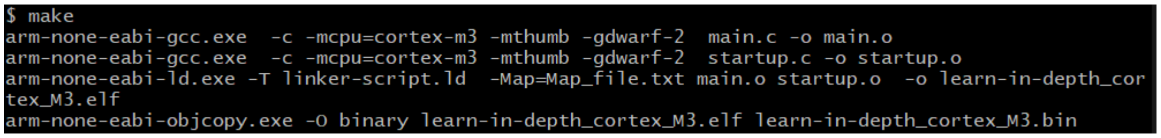
- We use weak to override a function already defined but not used by the manufacturer which is in our case the STM32 Microcontroller.
- We use Alias attribute to alias or to make unused defined function to point or to have same address as used function.
- From manufacturer perspective, Aliasing helps in decreasing the code size and only write many prototype functions without defining them.
- From developer perspective who writes embedded C code on Microcontrollers, weak and alias allow you to re-use the same functions that had defined by the manufacturer.

4-Makefile:



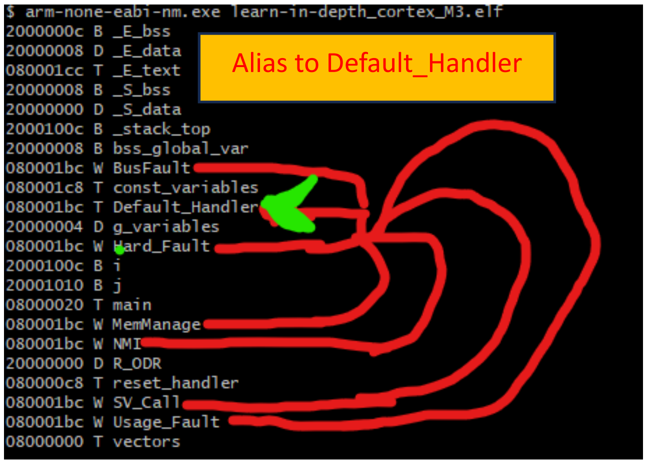
```makefile
#Author: Hassan Attia

CC=arm-none-eabi-
CFLAGS= -mcpu=cortex-m3 -mthumb -gdwarf-2
INCS=
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: %.c
	$(CC)gcc.exe  -c $(CFLAGS) $(INCS) $< -o $@

$(Project_Name).bin: $(Project_Name).elf

	$(CC)objcopy.exe -O binary $< $@

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








clean:
	rm *.o

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

```

5-Compilling using make:

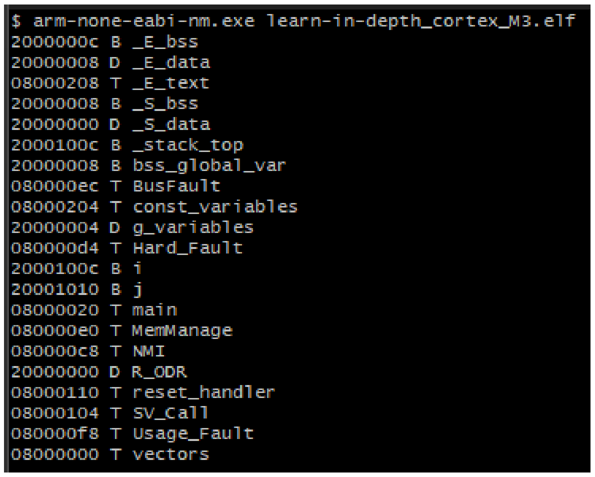


```
$ make
arm-none-eabi-gcc.exe  -c -mcpu=cortex-m3 -mthumb -gdwarf-2  main.c -o main.o
arm-none-eabi-gcc.exe  -c -mcpu=cortex-m3 -mthumb -gdwarf-2  startup.c -o startup.o
arm-none-eabi-ld.exe -T linker-script.ld  -Map=Map_file.txt main.o startup.o  -o learn-in-depth_cor
tex_M3.elf
arm-none-eabi-objcopy.exe -O binary learn-in-depth_cortex_M3.elf learn-in-depth_cortex_M3.bin
```

6-Symbols before and after using weak and alias attributes:

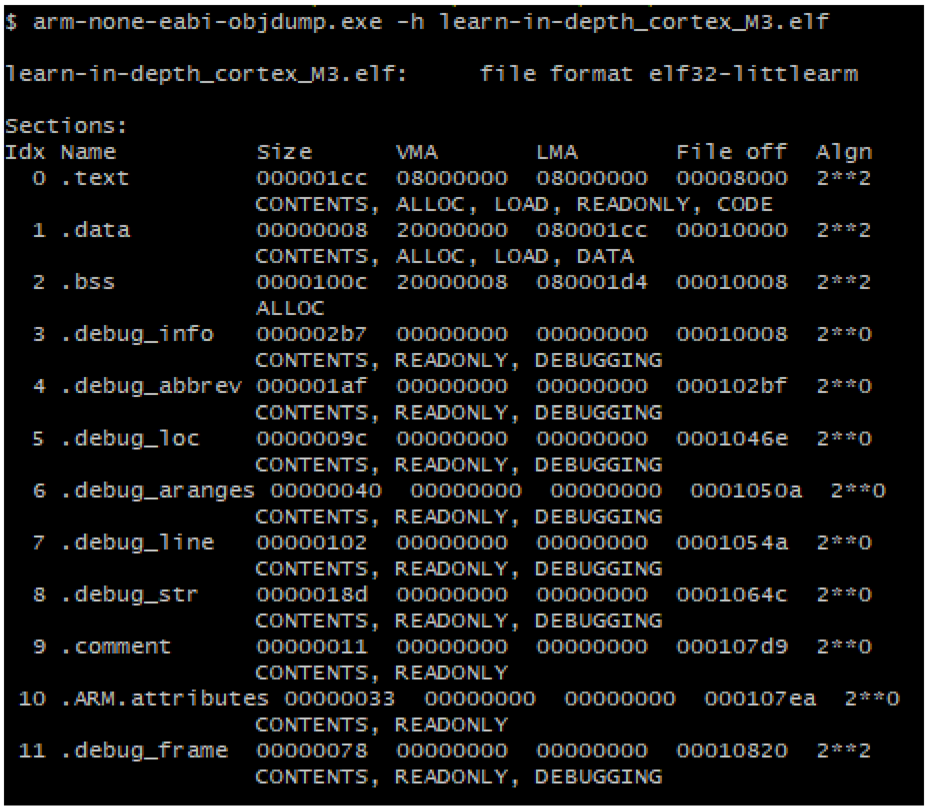


```
$ arm-none-eabi-nm.exe learn-in-depth_cortex_M3.elf
2000000c B _E_bss
20000008 D _E_data
080001cc T _E_text
20000008 B _S_bss
20000000 D _S_data
2000100c B _stack_top
20000008 B bss_global_var
080001bc W BusFault
080001c8 T const_variables
080001bc T Default_Handler
20000004 D g_variables
080001bc W Hard_Fault
2000100c B i
20001010 B j
08000020 T main
080001bc W MemManage
080001bc W NMI
20000000 D R_ODR
080000c8 T reset_handler
080001bc W SV_Call
080001bc W Usage_Fault
08000000 T vectors
```



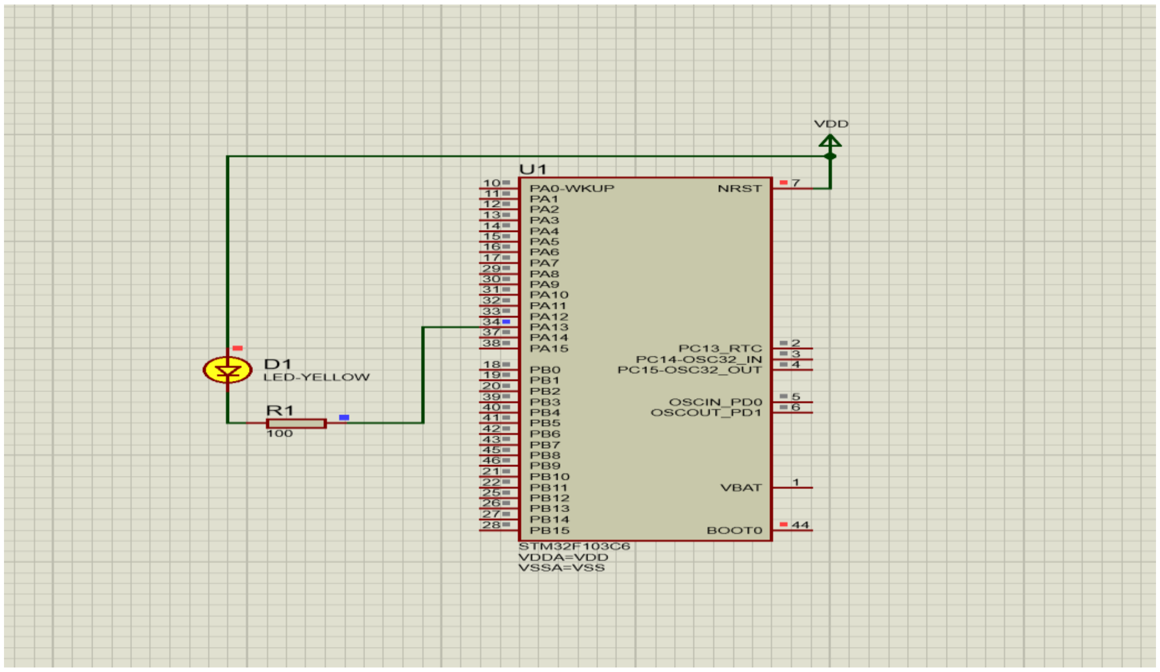
```
$ arm-none-eabi-nm.exe learn-in-depth_cortex_M3.elf
2000000c B _E_bss
20000008 D _E_data
08000208 T _E_text
20000008 B _S_bss
20000000 D _S_data
2000100c B _stack_top
20000008 B bss_global_var
080000ec T BusFault
08000204 T const_variables
20000004 D g_variables
080000d4 T Hard_Fault
2000100c B i
20001010 B j
08000020 T main
080000e0 T MemManage
080000c8 T NMI
20000000 D R_ODR
08000110 T reset_handler
08000104 T SV_Call
080000f8 T Usage_Fault
08000000 T vectors
```

7-Executable file Sections:

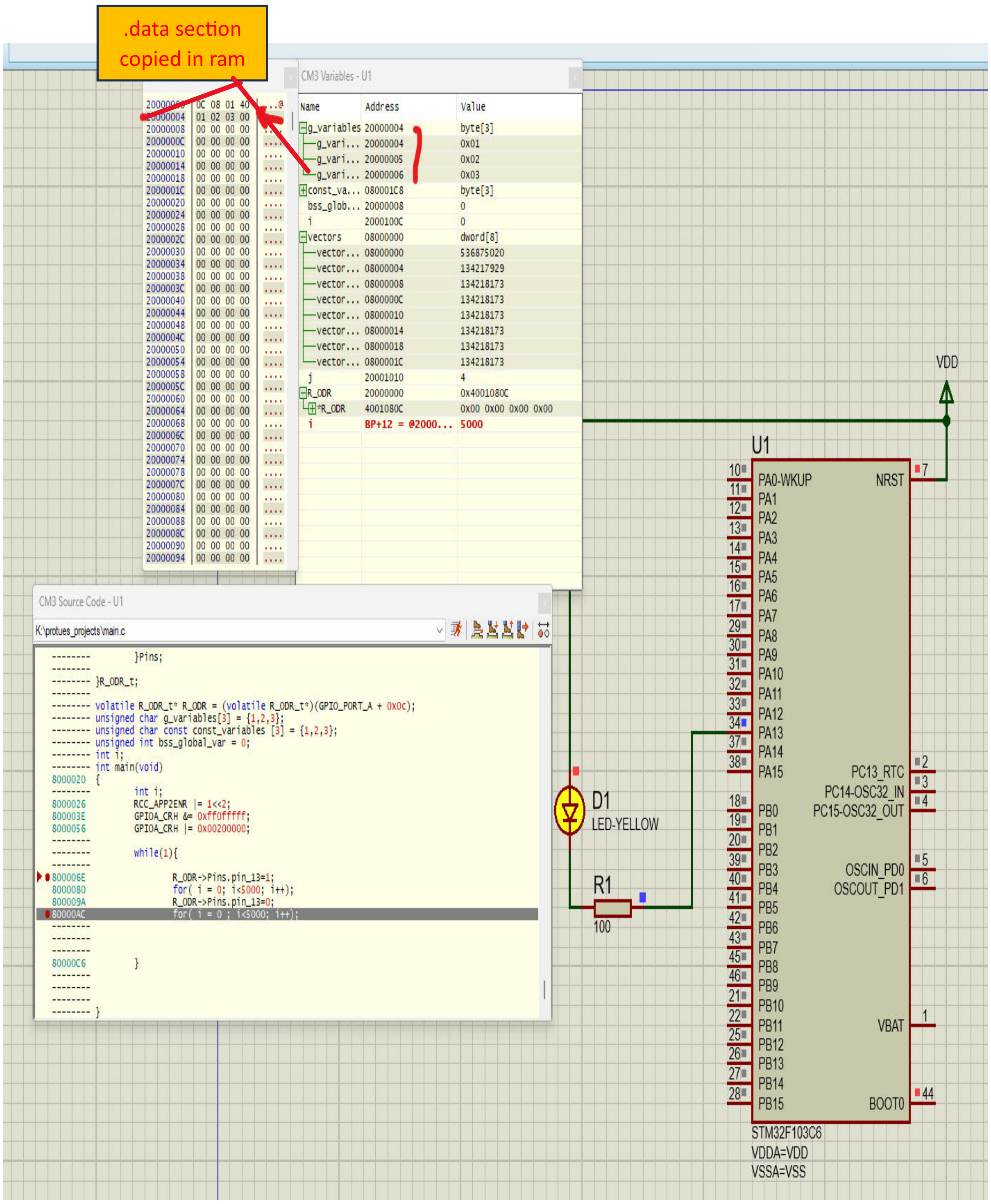


```
$ arm-none-eabi-objdump.exe -h learn-in-depth_cortex_M3.elf

learn-in-depth_cortex_M3.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000001cc  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000008  20000000  080001cc  00010000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          0000100c  20000008  080001d4  00010008  2**2
                  ALLOC
  3 .debug_info   000002b7  00000000  00000000  00010008  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev 000001af  00000000  00000000  000102bf  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    0000009c  00000000  00000000  0001046e  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000040  00000000  00000000  0001050a  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   00000102  00000000  00000000  0001054a  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_str    0000018d  00000000  00000000  0001064c  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000011  00000000  00000000  000107d9  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  000107ea  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000078  00000000  00000000  00010820  2**2
                  CONTENTS, READONLY, DEBUGGING
```

8-Simualtion on proteus:



9-Debugging(Ram, Flash, and values in variables):



```
20000000 | 0C 08 01 40 | ...@
20000004 | 01 02 03 00 | ....
20000008 | 00 00 00 00 | ....
2000000C | 00 00 00 00 | ....
20000010 | 00 00 00 00 | ....
20000014 | 00 00 00 00 | ....
20000018 | 00 00 00 00 | ....
2000001C | 00 00 00 00 | ....
20000020 | 00 00 00 00 | ....
20000024 | 00 00 00 00 | ....
20000028 | 00 00 00 00 | ....
2000002C | 00 00 00 00 | ....
20000030 | 00 00 00 00 | ....
20000034 | 00 00 00 00 | ....
20000038 | 00 00 00 00 | ....
2000003C | 00 00 00 00 | ....
20000040 | 00 00 00 00 | ....
20000044 | 00 00 00 00 | ....
20000048 | 00 00 00 00 | ....
2000004C | 00 00 00 00 | ....
20000050 | 00 00 00 00 | ....
20000054 | 00 00 00 00 | ....
20000058 | 00 00 00 00 | ....
2000005C | 00 00 00 00 | ....
20000060 | 00 00 00 00 | ....
20000064 | 00 00 00 00 | ....
20000068 | 00 00 00 00 | ....
2000006C | 00 00 00 00 | ....
20000070 | 00 00 00 00 | ....
20000074 | 00 00 00 00 | ....
20000078 | 00 00 00 00 | ....
2000007C | 00 00 00 00 | ....
20000080 | 00 00 00 00 | ....
20000084 | 00 00 00 00 | ....
20000088 | 00 00 00 00 | ....
2000008C | 00 00 00 00 | ....
20000090 | 00 00 00 00 | ....
20000094 | 00 00 00 00 | ....
```

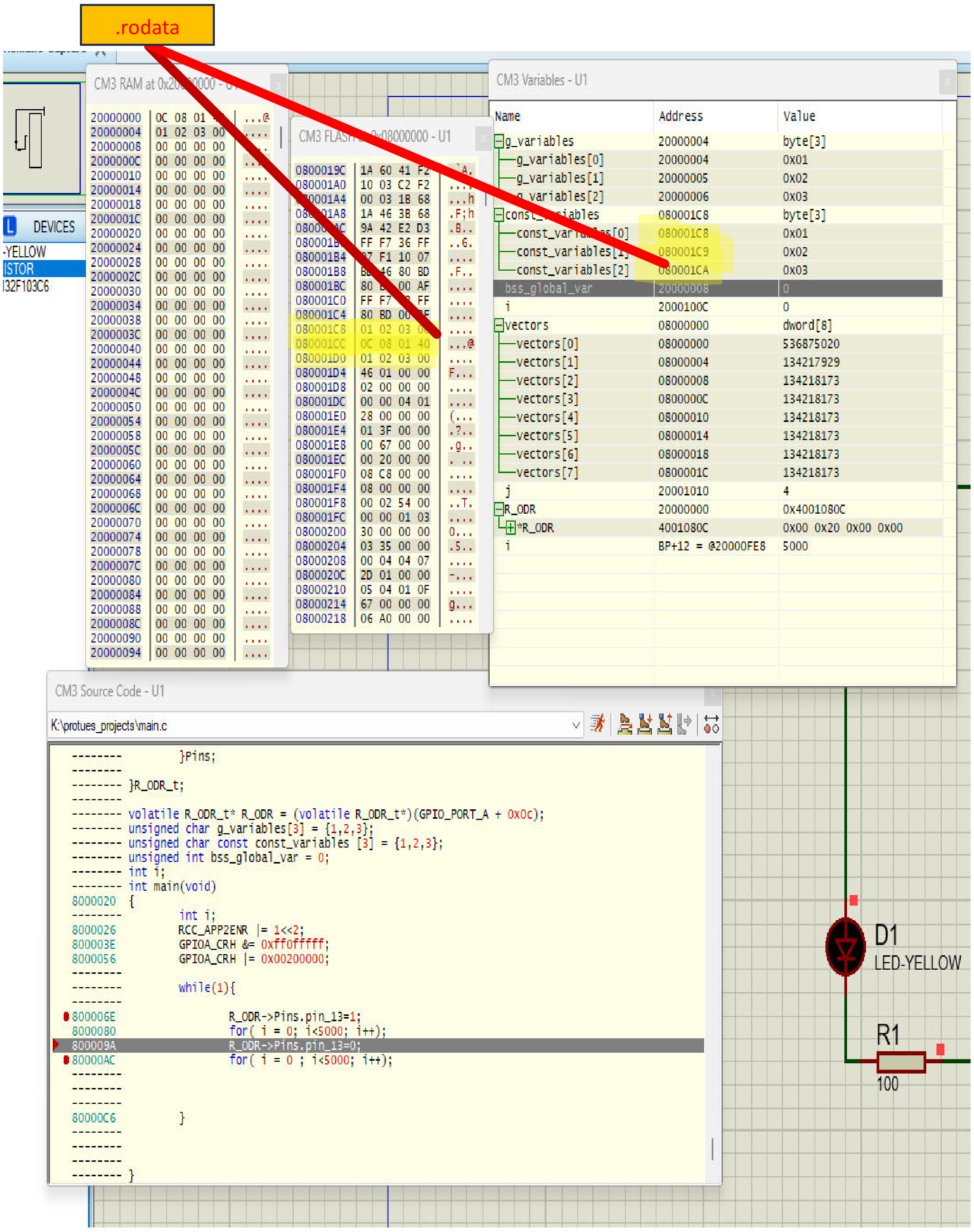
CM3 Variables - U1

Name	Address	Value
g_variables	20000004	byte[3]
g_vari...	20000004	0x01
g_vari...	20000005	0x02
g_vari...	20000006	0x03
const_va...	080001C8	byte[3]
bss_glob...	20000008	0
i	2000100C	0
vectors	08000000	dword[8]
vector...	08000000	536875020
vector...	08000004	134217929
vector...	08000008	134218173
vector...	0800000C	134218173
vector...	08000010	134218173
vector...	08000014	134218173
vector...	08000018	134218173
vector...	0800001C	134218173
j	20001010	4
R_ODR	20000000	0x4001080C
*R_ODR	4001080C	0x00 0x00 0x00 0x00
i	BP+12 = @2000...	5000

CM3 Source Code - U1

K:\protues_projects\main.c

```
--------        }Pins;
--------
-------- }R_ODR_t;
--------
-------- volatile R_ODR_t* R_ODR = (volatile R_ODR_t*)(GPIO_PORT_A + 0x0c);
-------- unsigned char g_variables[3] = {1,2,3};
-------- unsigned char const const_variables [3] = {1,2,3};
-------- unsigned int bss_global_var = 0;
-------- int i;
-------- int main(void)
8000020  {
--------        int i;
8000026         RCC_APP2ENR |= 1<<2;
800003E         GPIOA_CRH &= 0xff0fffff;
8000056         GPIOA_CRH |= 0x00200000;
--------
--------        while(1){
--------
800006E                 R_ODR->Pins.pin_13=1;
8000080                 for( i = 0; i<5000; i++);
800009A                 R_ODR->Pins.pin_13=0;
80000AC                 for( i = 0 ; i<5000; i++);
--------
--------
--------
80000C6         }
--------
--------
--------
-------- }
```



CM3 RAM at 0x20000000 - U1

```
20000000 | 0C 08 01 40 | ...@
20000004 | 01 02 03 00 | ....
20000008 | 00 00 00 00 | ....
2000000C | 00 00 00 00 | ....
20000010 | 00 00 00 00 | ....
20000014 | 00 00 00 00 | ....
20000018 | 00 00 00 00 | ....
2000001C | 00 00 00 00 | ....
20000020 | 00 00 00 00 | ....
20000024 | 00 00 00 00 | ....
20000028 | 00 00 00 00 | ....
2000002C | 00 00 00 00 | ....
20000030 | 00 00 00 00 | ....
20000034 | 00 00 00 00 | ....
20000038 | 00 00 00 00 | ....
2000003C | 00 00 00 00 | ....
20000040 | 00 00 00 00 | ....
20000044 | 00 00 00 00 | ....
20000048 | 00 00 00 00 | ....
2000004C | 00 00 00 00 | ....
20000050 | 00 00 00 00 | ....
20000054 | 00 00 00 00 | ....
20000058 | 00 00 00 00 | ....
2000005C | 00 00 00 00 | ....
20000060 | 00 00 00 00 | ....
20000064 | 00 00 00 00 | ....
20000068 | 00 00 00 00 | ....
2000006C | 00 00 00 00 | ....
20000070 | 00 00 00 00 | ....
20000074 | 00 00 00 00 | ....
20000078 | 00 00 00 00 | ....
2000007C | 00 00 00 00 | ....
20000080 | 00 00 00 00 | ....
20000084 | 00 00 00 00 | ....
20000088 | 00 00 00 00 | ....
2000008C | 00 00 00 00 | ....
20000090 | 00 00 00 00 | ....
20000094 | 00 00 00 00 | ....
```

CM3 FLASH at 0x08000000 - U1

```
0800019C | 1A 60 41 F2 | .`A.
080001A0 | 10 03 C2 F2 | ....
080001A4 | 00 03 1B 68 | ...h
080001A8 | 1A 46 3B 68 | .F;h
080001AC | 9A 42 E2 D3 | .B..
080001B0 | FF F7 36 FF | ..6.
080001B4 | 07 F1 10 07 | ....
080001B8 | BD 46 80 BD | .F..
080001BC | 80 [illegible] 00 AF | ....
080001C0 | FF F7 [illegible] FF | ....
080001C4 | 80 BD 00 [illegible] | ....
080001C8 | 01 02 03 00 | ....
080001CC | 0C 08 01 40 | ...@
080001D0 | 01 02 03 00 | ....
080001D4 | 46 01 00 00 | F...
080001D8 | 02 00 00 00 | ....
080001DC | 00 00 04 01 | ....
080001E0 | 28 00 00 00 | (...
080001E4 | 01 3F 00 00 | .?..
080001E8 | 00 67 00 00 | .g..
080001EC | 00 20 00 00 | . ..
080001F0 | 08 C8 00 00 | ....
080001F4 | 08 00 00 00 | ....
080001F8 | 00 02 54 00 | ..T.
080001FC | 00 00 01 03 | ....
08000200 | 30 00 00 00 | 0...
08000204 | 03 35 00 00 | .5..
08000208 | 00 04 04 07 | ....
0800020C | 2D 01 00 00 | -...
08000210 | 05 04 01 0F | ....
08000214 | 67 00 00 00 | g...
08000218 | 06 A0 00 00 | ....
```

CM3 Variables - U1

Name	Address	Value
g_variables	20000004	byte[3]
g_variables[0]	20000004	0x01
g_variables[1]	20000005	0x02
g_variables[2]	20000006	0x03
const_variables	080001C8	byte[3]
const_variables[0]	080001C8	0x01
const_variables[1]	080001C9	0x02
const_variables[2]	080001CA	0x03
bss_global_var	20000008	0
i	2000100C	0
vectors	08000000	dword[8]
vectors[0]	08000000	536875020
vectors[1]	08000004	134217929
vectors[2]	08000008	134218173
vectors[3]	0800000C	134218173
vectors[4]	08000010	134218173
vectors[5]	08000014	134218173
vectors[6]	08000018	134218173
vectors[7]	0800001C	134218173
j	20001010	4
R_ODR	20000000	0x4001080C
*R_ODR	4001080C	0x00 0x20 0x00 0x00
i	BP+12 = @20000FE8	5000

CM3 Source Code - U1

K:\protues_projects\main.c

```
--------         }Pins;
--------
-------- }R_ODR_t;
--------
-------- volatile R_ODR_t* R_ODR = (volatile R_ODR_t*)(GPIO_PORT_A + 0x0c);
-------- unsigned char g_variables[3] = {1,2,3};
-------- unsigned char const const_variables [3] = {1,2,3};
-------- unsigned int bss_global_var = 0;
-------- int i;
-------- int main(void)
8000020  {
--------         int i;
8000026          RCC_APP2ENR |= 1<<2;
800003E          GPIOA_CRH &= 0xff0fffff;
8000056          GPIOA_CRH |= 0x00200000;
--------
--------         while(1){
--------
800006E                  R_ODR->Pins.pin_13=1;
8000080                  for( i = 0; i<5000; i++);
800009A                  R_ODR->Pins.pin_13=0;
80000AC                  for( i = 0 ; i<5000; i++);
--------
--------
--------
80000C6          }
--------
--------
--------
-------- }
```