Pressure Controller

Mastering Embedded Systems Diploma
www.learnindepth.com
First Term (Final Project 1)

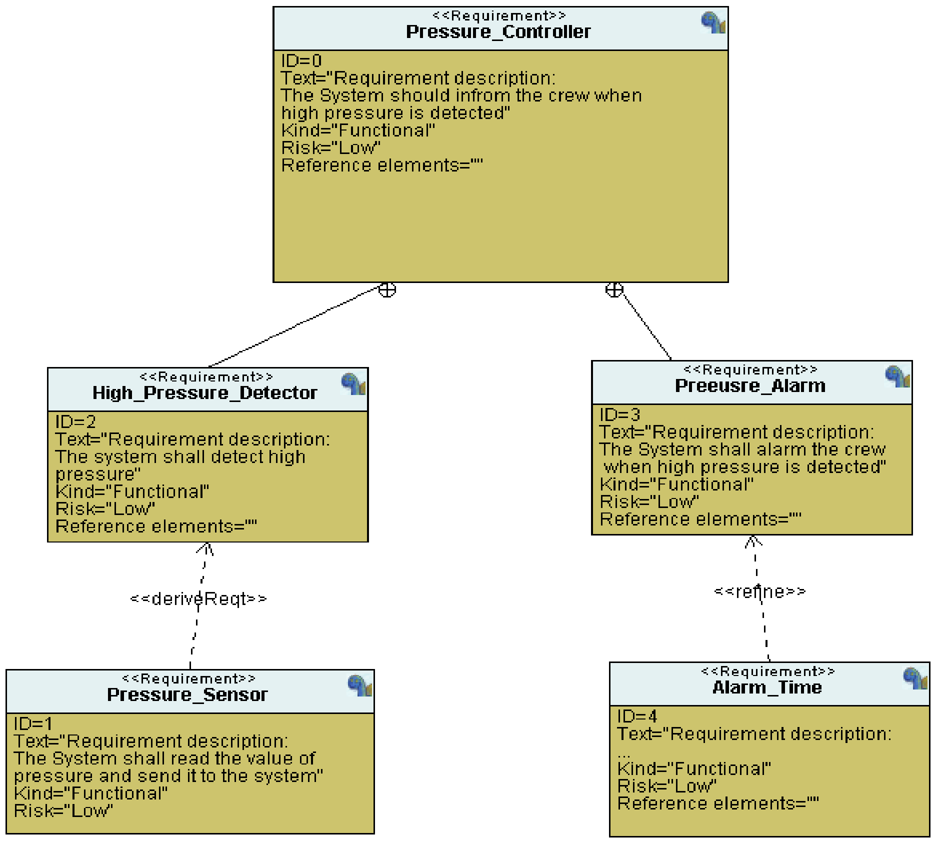
Eng. Michel Adel Fouad
My Profile: https://www.learn-in-depth.com/online-diploma/meshoadel2018%40gmail.com

Case Study

A" client" expects you to deliver the software of the following system:
Specification (from the client)
- *A pressure controller informs the crew of a cabin with an alarm when the pressure exceeds 20 bars in the cabin.*
- *The alarm duration equals 60 seconds.*

Assumptions:
- The system doesn.t need a starting system or shut down system.
- The system will always work efficiently and will never fail.

Requirement Diagram



<<Requirement>>
Pressure_Controller
ID=0
Text="Requirement description:
The System should infrom the crew when high pressure is detected"
Kind="Functional"
Risk="Low"
Reference elements=""

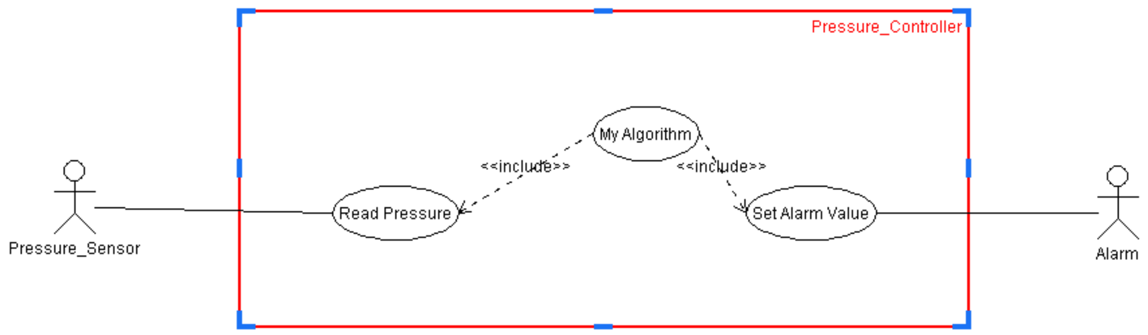
<<Requirement>>
High_Pressure_Detector
ID=2
Text="Requirement description:
The system shall detect high pressure"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Preeusre_Alarm
ID=3
Text="Requirement description:
The System shall alarm the crew when high pressure is detected"
Kind="Functional"
Risk="Low"
Reference elements=""

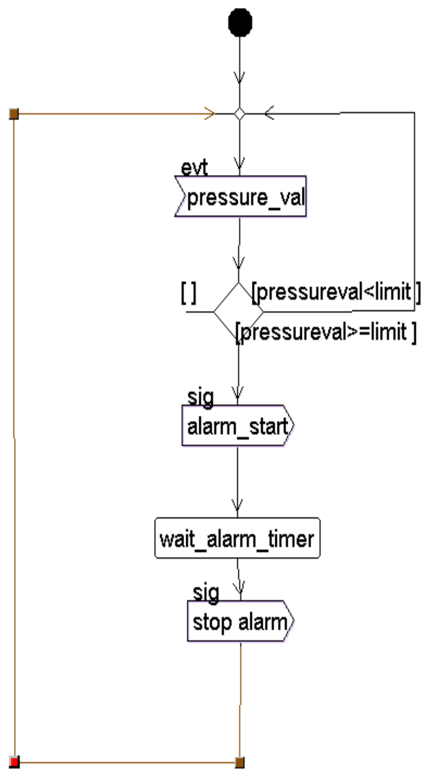
<<Requirement>>
Pressure_Sensor
ID=1
Text="Requirement description:
The System shall read the value of pressure and send it to the system"
Kind="Functional"
Risk="Low"

<<Requirement>>
Alarm_Time
ID=4
Text="Requirement description:
..."
Kind="Functional"
Risk="Low"
Reference elements=""

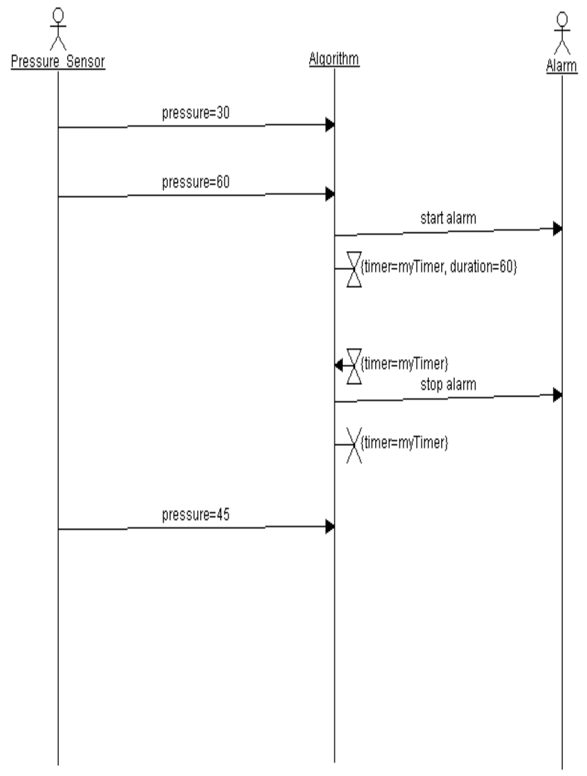
Use Case Diagram



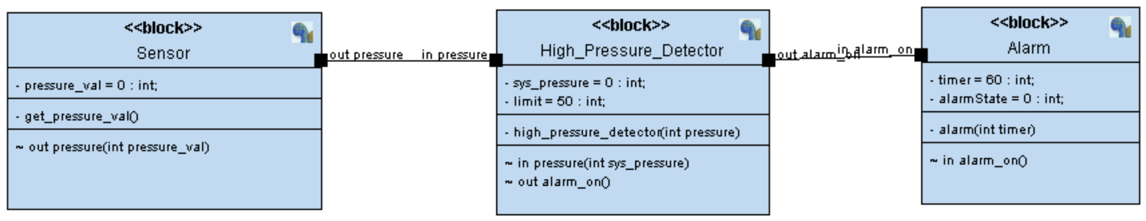
Activity Diagram



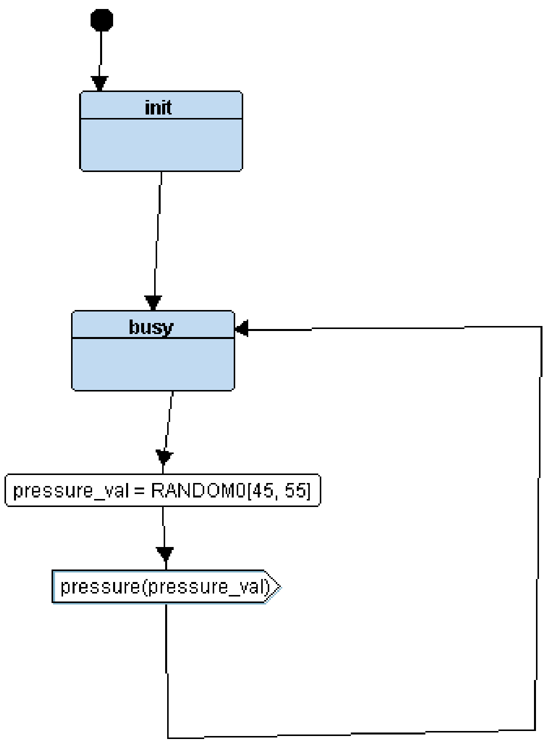
Sequence Diagram

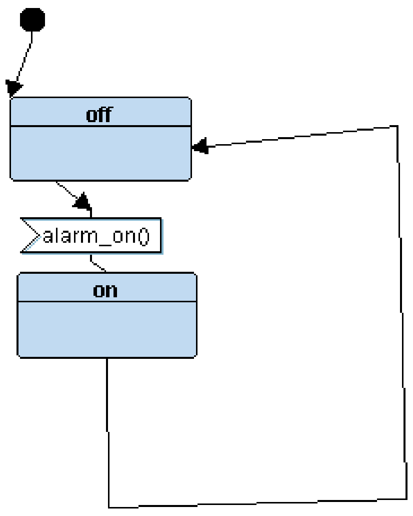


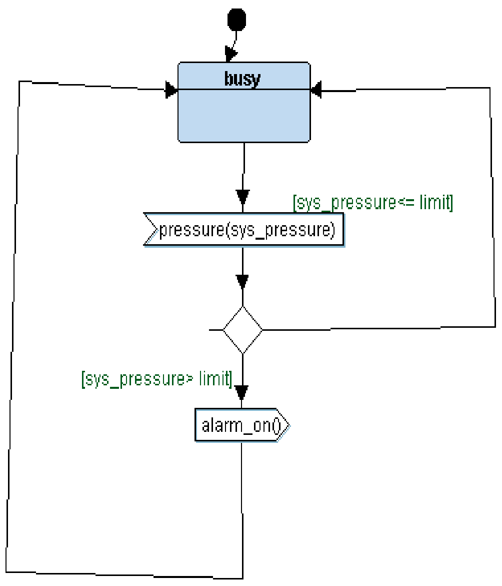
System Design



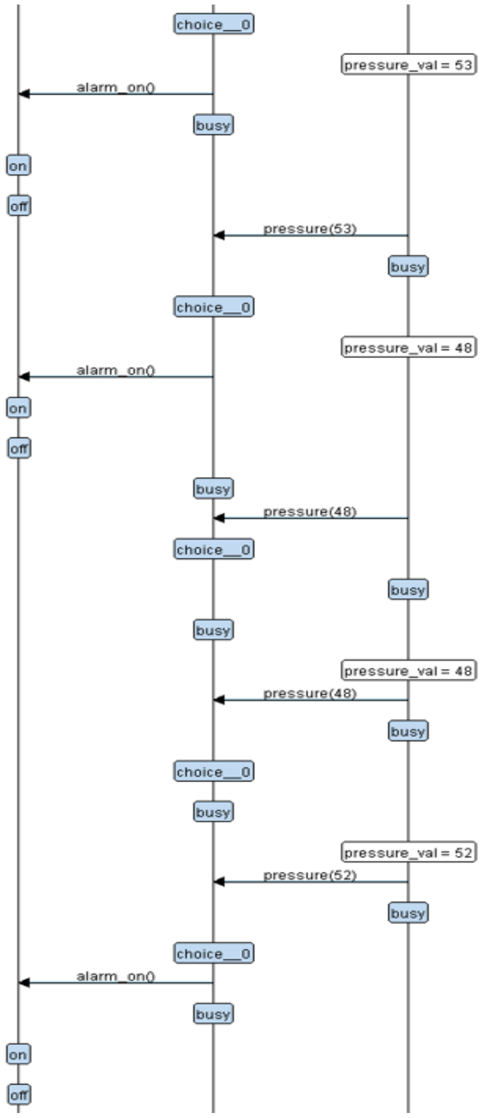
State Machines







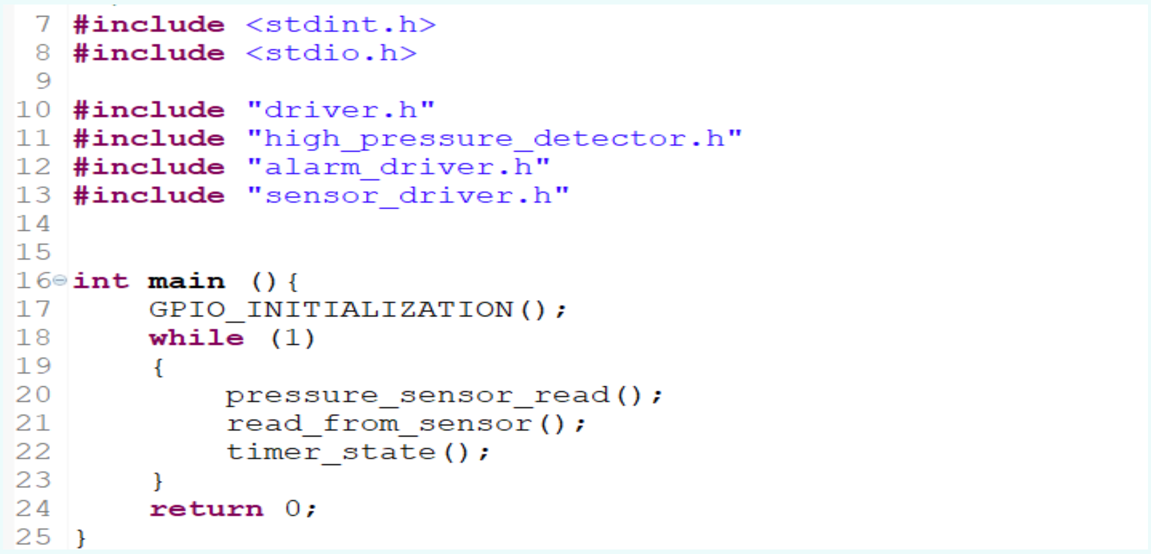
System Design Simulation



Code:

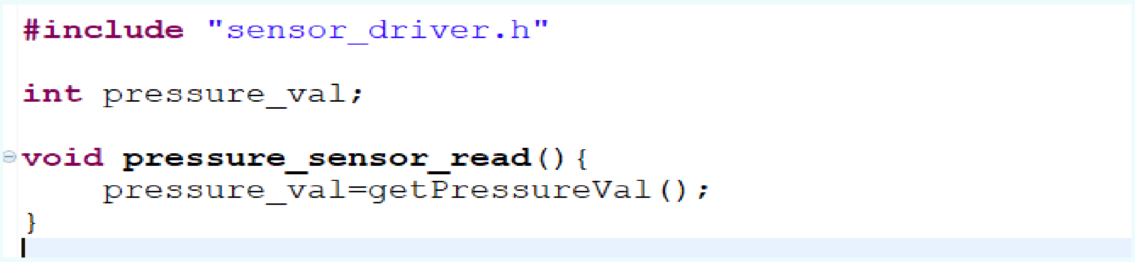
This project was bult on three modules, one for reading from sensor and another one for detecting high pressure and another one to configure the alarm.

main.c file:



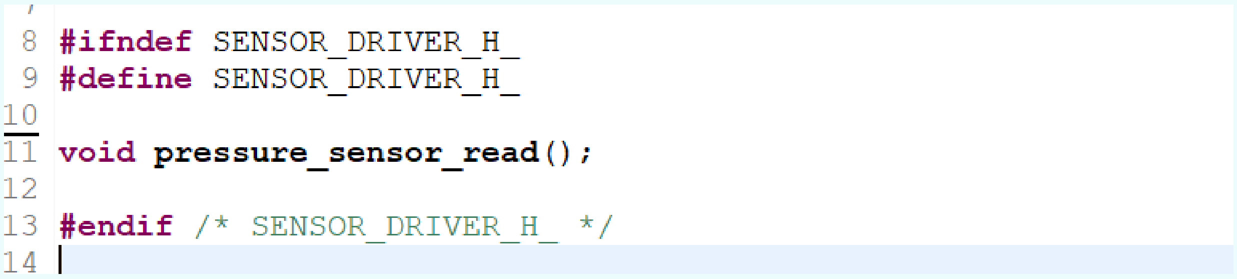
```c
#include <stdint.h>
#include <stdio.h>

#include "driver.h"
#include "high_pressure_detector.h"
#include "alarm_driver.h"
#include "sensor_driver.h"


int main (){
    GPIO_INITIALIZATION();
    while (1)
    {
        pressure_sensor_read();
        read_from_sensor();
        timer_state();
    }
    return 0;
}
```

sensor_driver.c file:



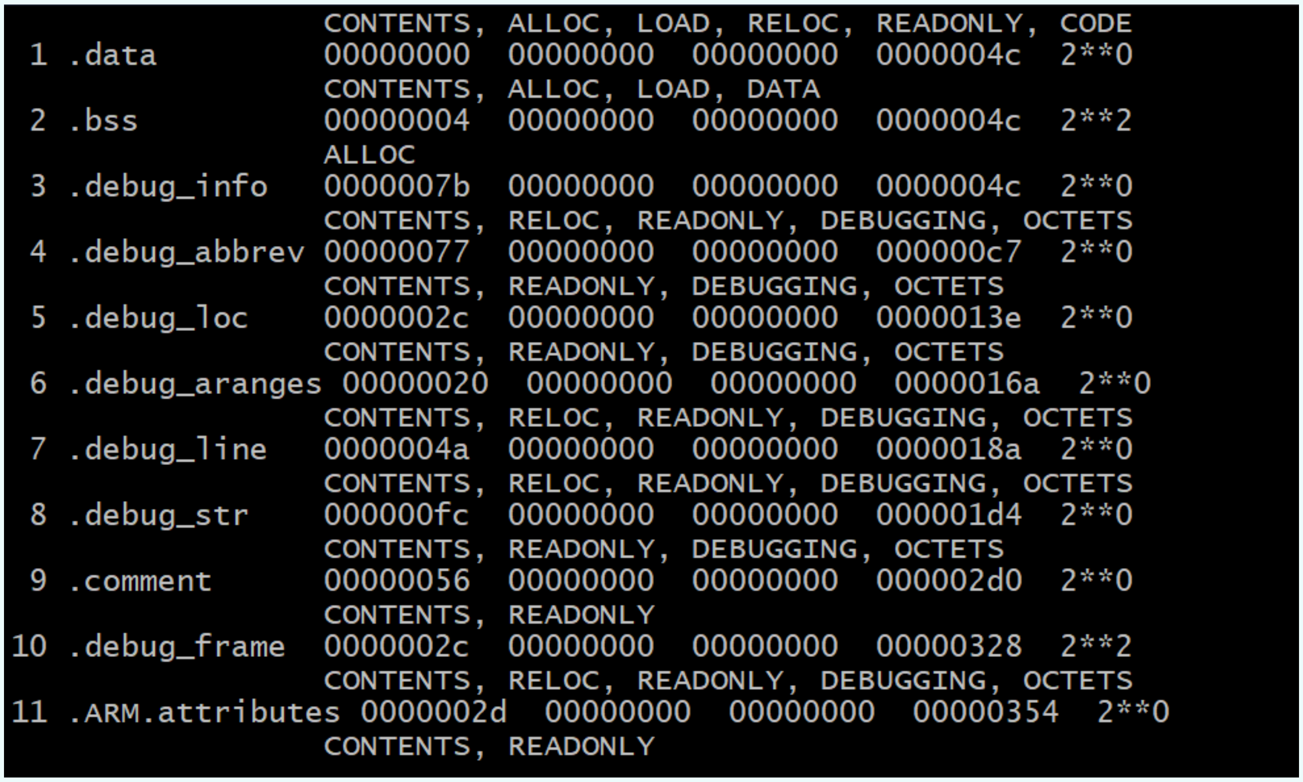
```c
#include "sensor_driver.h"

int pressure_val;

void pressure_sensor_read(){
    pressure_val=getPressureVal();
}
```

sensor_ driver.h file:



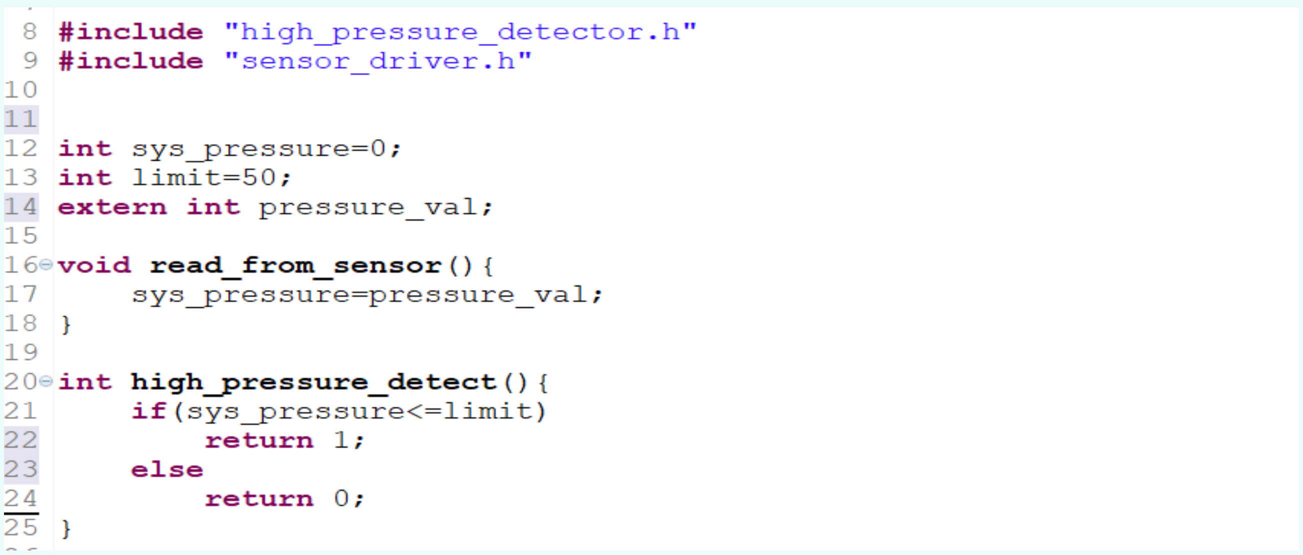
```c
#ifndef SENSOR_DRIVER_H_
#define SENSOR_DRIVER_H_

void pressure_sensor_read();

#endif /* SENSOR_DRIVER_H_ */
```

sensor_driver Section Table:



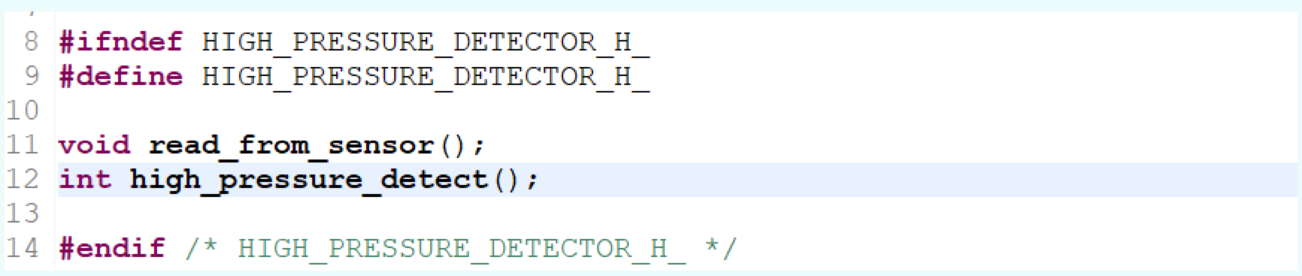
```
                     CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data             00000000  00000000  00000000  0000004c  2**0
                     CONTENTS, ALLOC, LOAD, DATA
 2 .bss              00000004  00000000  00000000  0000004c  2**2
                     ALLOC
 3 .debug_info       0000007b  00000000  00000000  0000004c  2**0
                     CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
 4 .debug_abbrev     00000077  00000000  00000000  000000c7  2**0
                     CONTENTS, READONLY, DEBUGGING, OCTETS
 5 .debug_loc        0000002c  00000000  00000000  0000013e  2**0
                     CONTENTS, READONLY, DEBUGGING, OCTETS
 6 .debug_aranges 00000020  00000000  00000000  0000016a  2**0
                     CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
 7 .debug_line       0000004a  00000000  00000000  0000018a  2**0
                     CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
 8 .debug_str        000000fc  00000000  00000000  000001d4  2**0
                     CONTENTS, READONLY, DEBUGGING, OCTETS
 9 .comment          00000056  00000000  00000000  000002d0  2**0
                     CONTENTS, READONLY
10 .debug_frame      0000002c  00000000  00000000  00000328  2**2
                     CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
11 .ARM.attributes 0000002d  00000000  00000000  00000354  2**0
                     CONTENTS, READONLY
```

high_pressure_detector.c file:



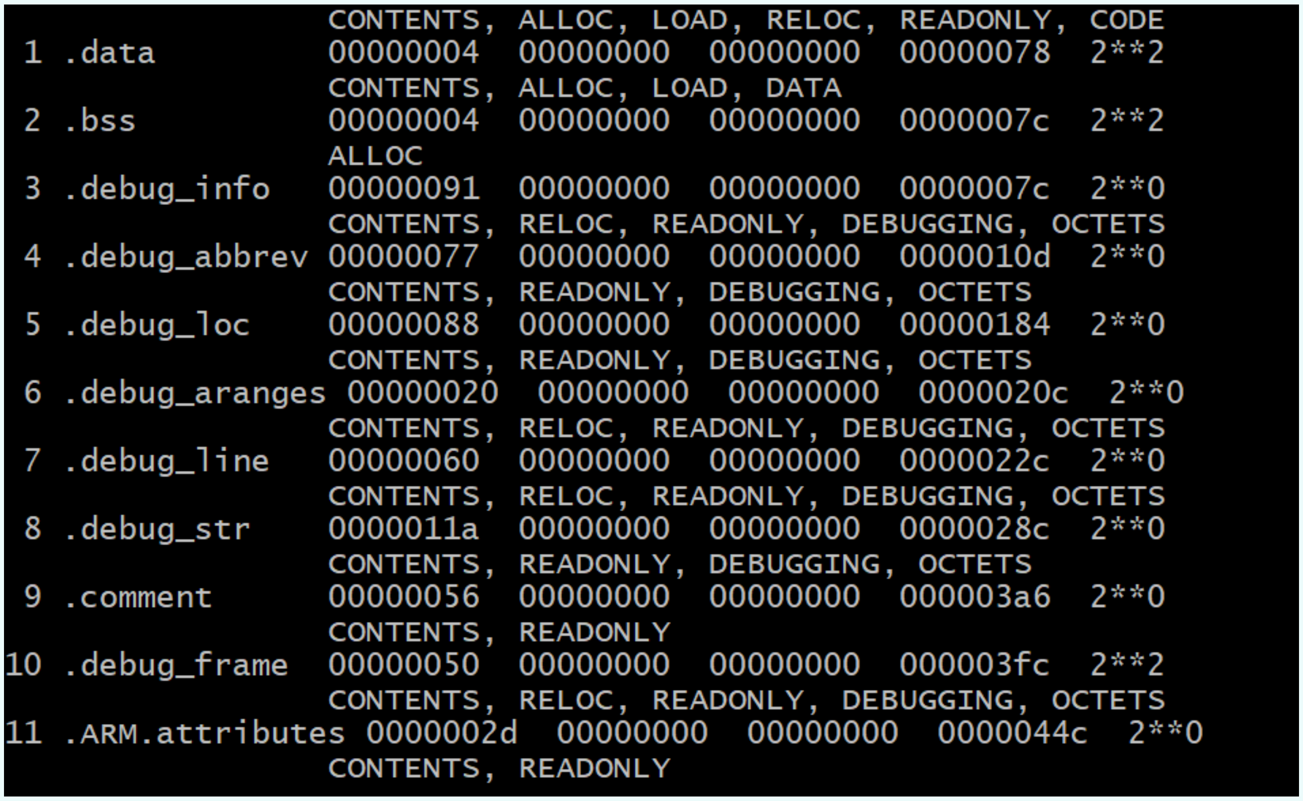
```c
#include "high_pressure_detector.h"
#include "sensor_driver.h"


int sys_pressure=0;
int limit=50;
extern int pressure_val;

void read_from_sensor(){
    sys_pressure=pressure_val;
}

int high_pressure_detect(){
    if(sys_pressure<=limit)
        return 1;
    else
        return 0;
}
```

high_pressure_detector.h file:



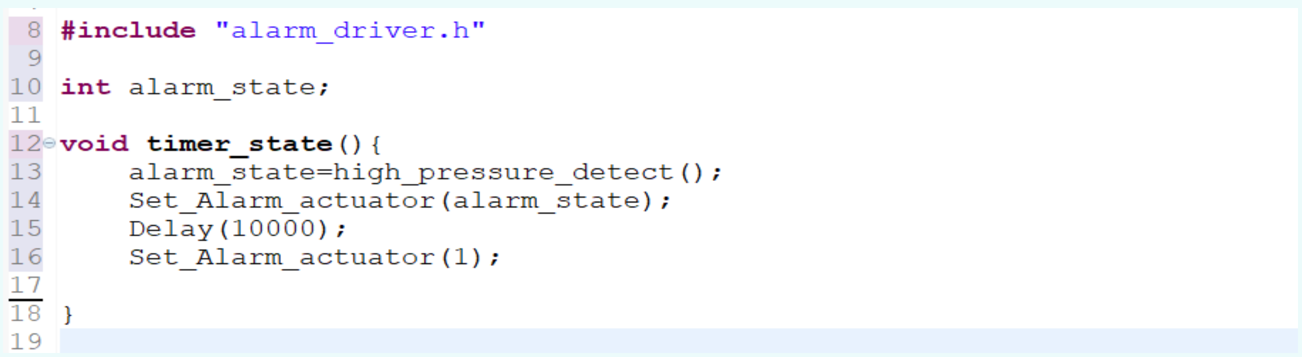
```c
#ifndef HIGH_PRESSURE_DETECTOR_H_
#define HIGH_PRESSURE_DETECTOR_H_

void read_from_sensor();
int high_pressure_detect();

#endif /* HIGH_PRESSURE_DETECTOR_H_ */
```

high_pressure_detector Swctions table:



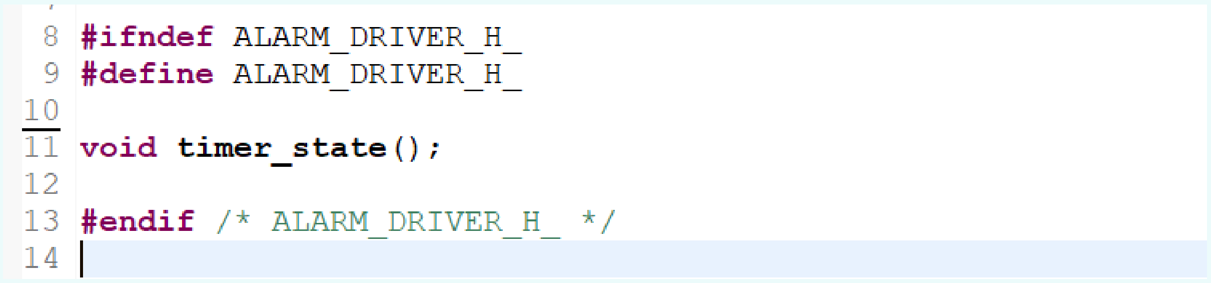
```
                   CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data           00000004  00000000  00000000  00000078  2**2
                   CONTENTS, ALLOC, LOAD, DATA
 2 .bss            00000004  00000000  00000000  0000007c  2**2
                   ALLOC
 3 .debug_info     00000091  00000000  00000000  0000007c  2**0
                   CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
 4 .debug_abbrev   00000077  00000000  00000000  0000010d  2**0
                   CONTENTS, READONLY, DEBUGGING, OCTETS
 5 .debug_loc      00000088  00000000  00000000  00000184  2**0
                   CONTENTS, READONLY, DEBUGGING, OCTETS
 6 .debug_aranges 00000020  00000000  00000000  0000020c  2**0
                   CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
 7 .debug_line     00000060  00000000  00000000  0000022c  2**0
                   CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
 8 .debug_str      0000011a  00000000  00000000  0000028c  2**0
                   CONTENTS, READONLY, DEBUGGING, OCTETS
 9 .comment        00000056  00000000  00000000  000003a6  2**0
                   CONTENTS, READONLY
10 .debug_frame    00000050  00000000  00000000  000003fc  2**2
                   CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
11 .ARM.attributes 0000002d  00000000  00000000  0000044c  2**0
                   CONTENTS, READONLY
```

alarm_driver.c file:



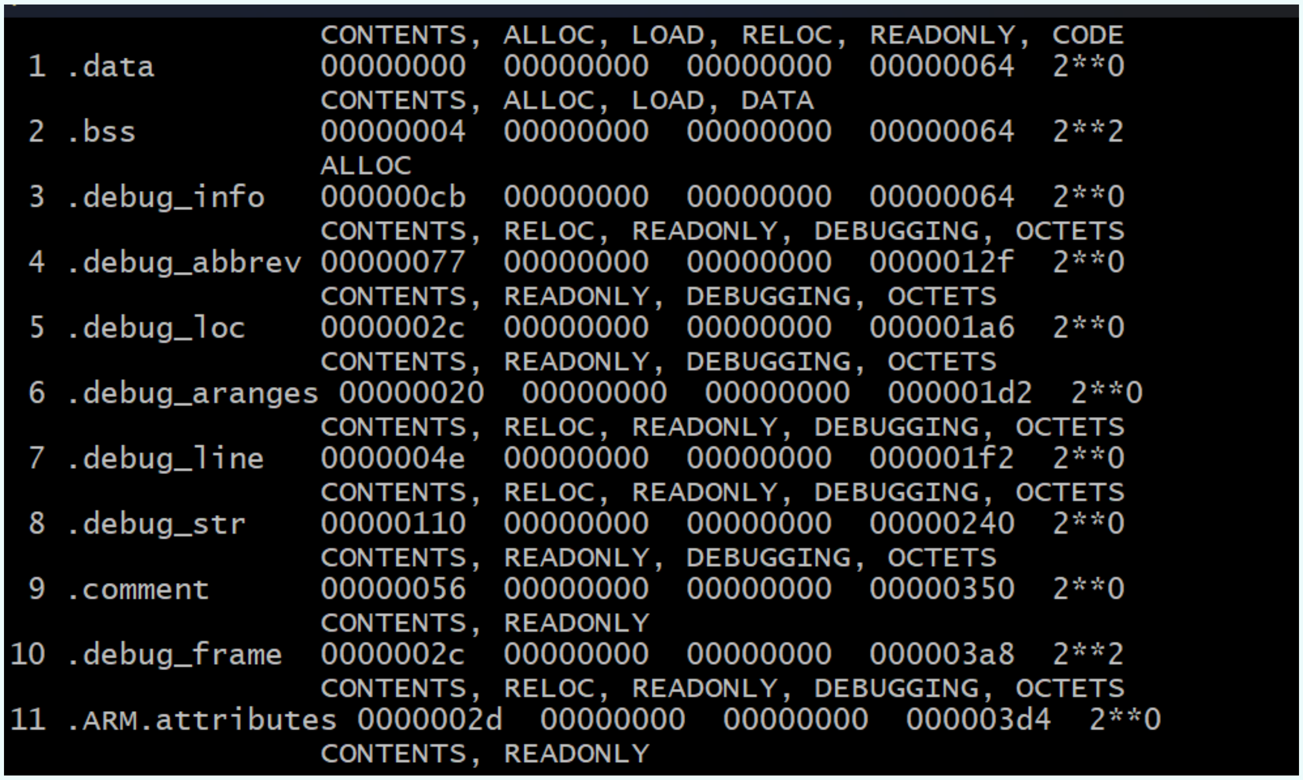
```c
#include "alarm_driver.h"

int alarm_state;

void timer_state(){
    alarm_state=high_pressure_detect();
    Set_Alarm_actuator(alarm_state);
    Delay(10000);
    Set_Alarm_actuator(1);

}

```

alarm_driver.h file:



```
#ifndef ALARM_DRIVER_H_
#define ALARM_DRIVER_H_

void timer_state();

#endif /* ALARM_DRIVER_H_ */

```

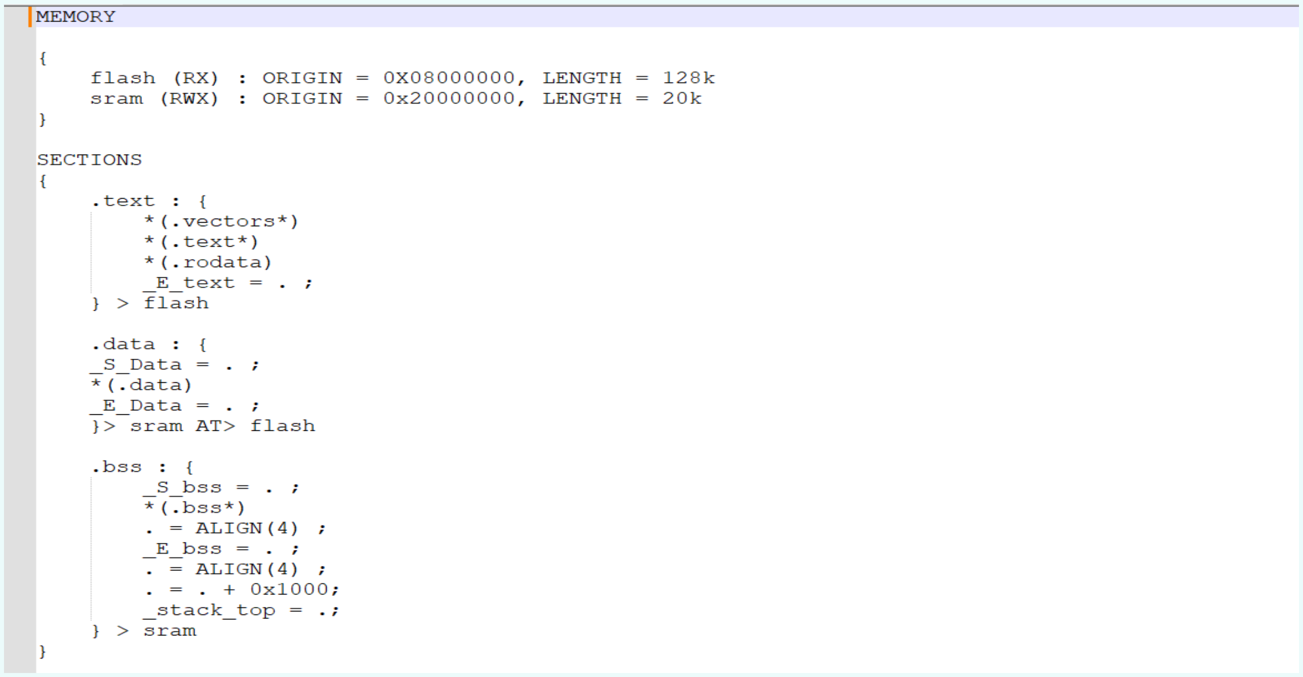
alarm_driver Sections table:



```
                      CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data              00000000  00000000  00000000  00000064  2**0
                      CONTENTS, ALLOC, LOAD, DATA
 2 .bss               00000004  00000000  00000000  00000064  2**2
                      ALLOC
 3 .debug_info        000000cb  00000000  00000000  00000064  2**0
                      CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
 4 .debug_abbrev 00000077  00000000  00000000  0000012f  2**0
                      CONTENTS, READONLY, DEBUGGING, OCTETS
 5 .debug_loc         0000002c  00000000  00000000  000001a6  2**0
                      CONTENTS, READONLY, DEBUGGING, OCTETS
 6 .debug_aranges 00000020  00000000  00000000  000001d2  2**0
                      CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
 7 .debug_line        0000004e  00000000  00000000  000001f2  2**0
                      CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
 8 .debug_str         00000110  00000000  00000000  00000240  2**0
                      CONTENTS, READONLY, DEBUGGING, OCTETS
 9 .comment           00000056  00000000  00000000  00000350  2**0
                      CONTENTS, READONLY
10 .debug_frame       0000002c  00000000  00000000  000003a8  2**2
                      CONTENTS, RELOC, READONLY, DEBUGGING, OCTETS
11 .ARM.attributes 0000002d  00000000  00000000  000003d4  2**0
                      CONTENTS, READONLY
```

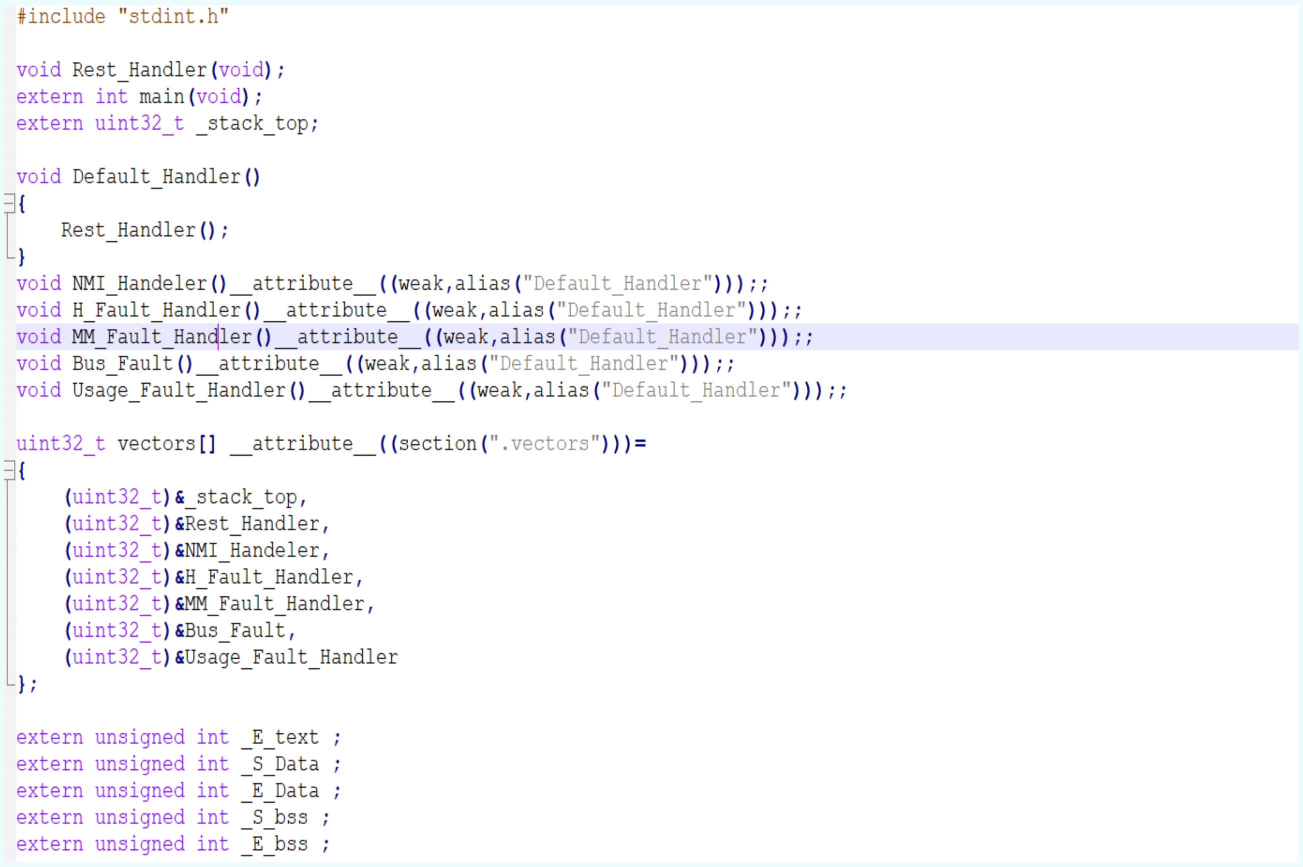
Building code:

Linker_script.ld file:

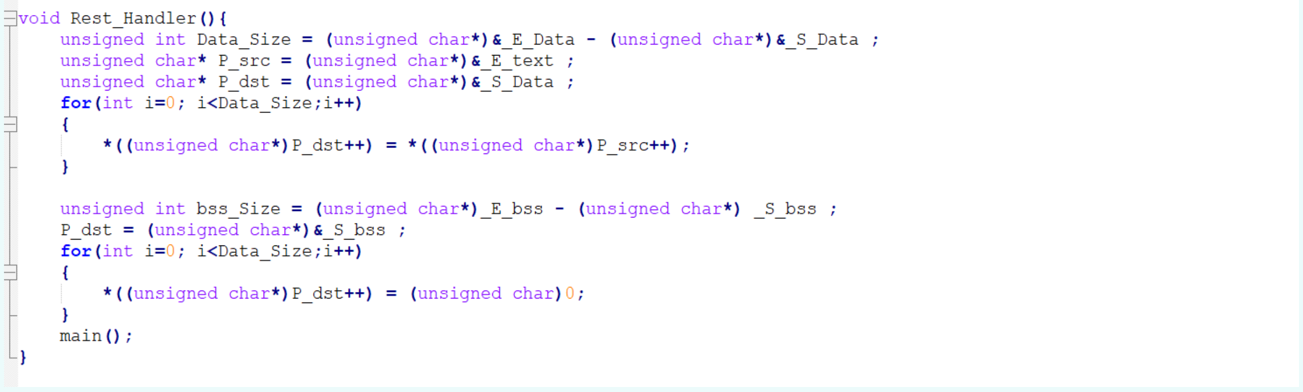


```
MEMORY

{
    flash  (RX)  : ORIGIN = 0X08000000, LENGTH = 128k
    sram  (RWX)  : ORIGIN = 0x20000000, LENGTH = 20k
}

SECTIONS
{
    .text : {
        *(.vectors*)
        *(.text*)
        *(.rodata)
        _E_text = . ;
    } > flash

    .data : {
    _S_Data = . ;
    *(.data)
    _E_Data = . ;
    }> sram AT> flash

    .bss : {
        _S_bss = . ;
        *(.bss*)
        . = ALIGN(4) ;
        _E_bss = . ;
        . = ALIGN(4) ;
        . = . + 0x1000;
        _stack_top = .;
    } > sram
}
```

startup.c file:

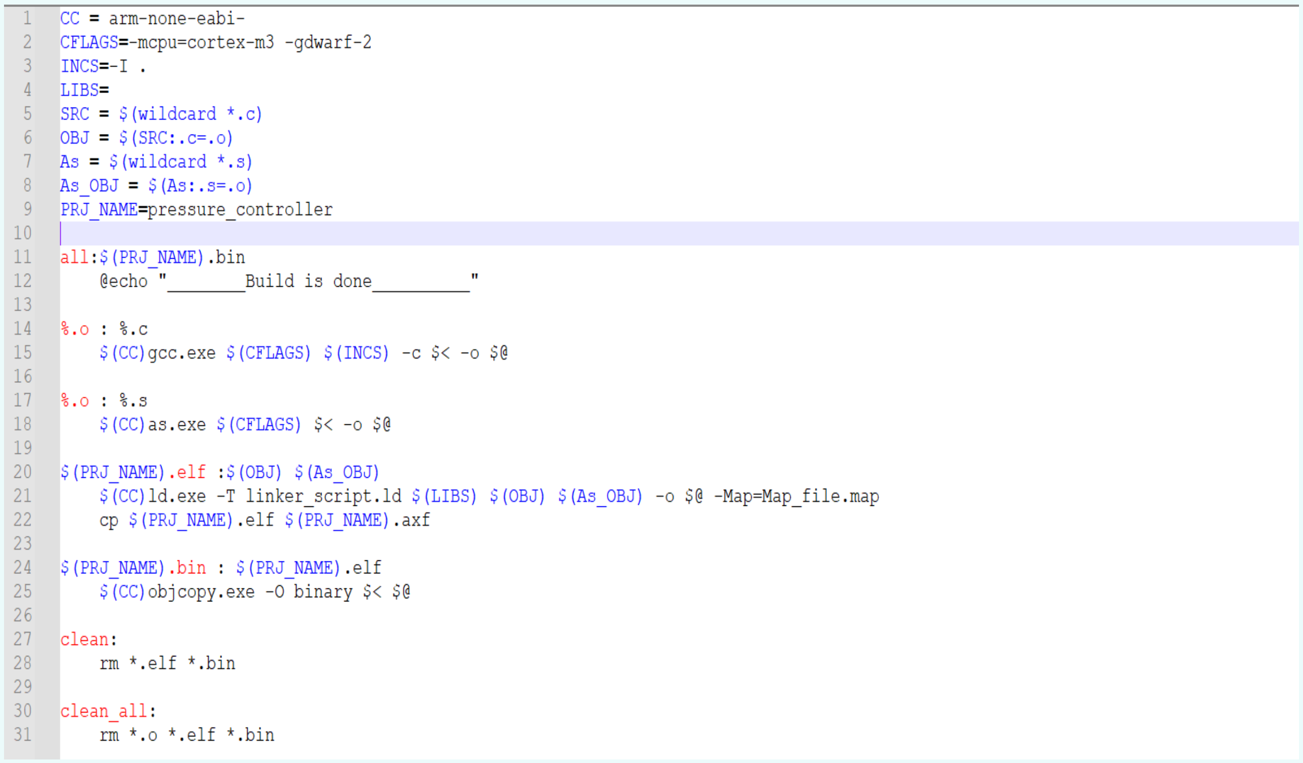


```
#include "stdint.h"

void Rest_Handler(void);
extern int main(void);
extern uint32_t _stack_top;

void Default_Handler()
{
    Rest_Handler();
}
void NMI_Handeler()__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")));;

uint32_t vectors[] __attribute__((section(".vectors")))=
{
    (uint32_t)&_stack_top,
    (uint32_t)&Rest_Handler,
    (uint32_t)&NMI_Handeler,
    (uint32_t)&H_Fault_Handler,
    (uint32_t)&MM_Fault_Handler,
    (uint32_t)&Bus_Fault,
    (uint32_t)&Usage_Fault_Handler
};

extern unsigned int _E_text ;
extern unsigned int _S_Data ;
extern unsigned int _E_Data ;
extern unsigned int _S_bss ;
extern unsigned int _E_bss ;
```



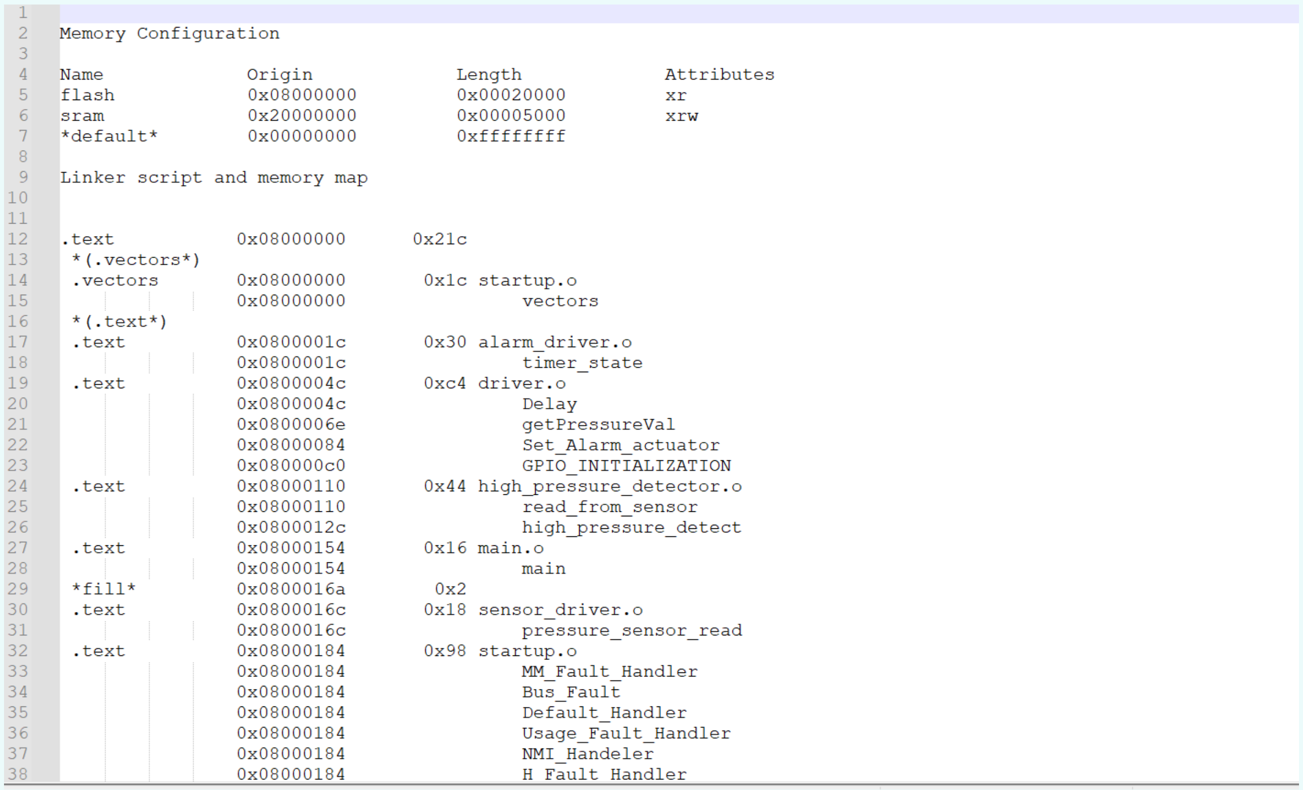
```c
void Rest_Handler(){
    unsigned int Data_Size = (unsigned char*)&_E_Data - (unsigned char*)&_S_Data ;
    unsigned char* P_src = (unsigned char*)&_E_text ;
    unsigned char* P_dst = (unsigned char*)&_S_Data ;
    for(int i=0; i<Data_Size;i++)
    {
        *((unsigned char*)P_dst++) = *((unsigned char*)P_src++);
    }

    unsigned int bss_Size = (unsigned char*)_E_bss - (unsigned char*) _S_bss ;
    P_dst = (unsigned char*)&_S_bss ;
    for(int i=0; i<Data_Size;i++)
    {
        *((unsigned char*)P_dst++) = (unsigned char)0;
    }
    main();
}
```

Makefile file:

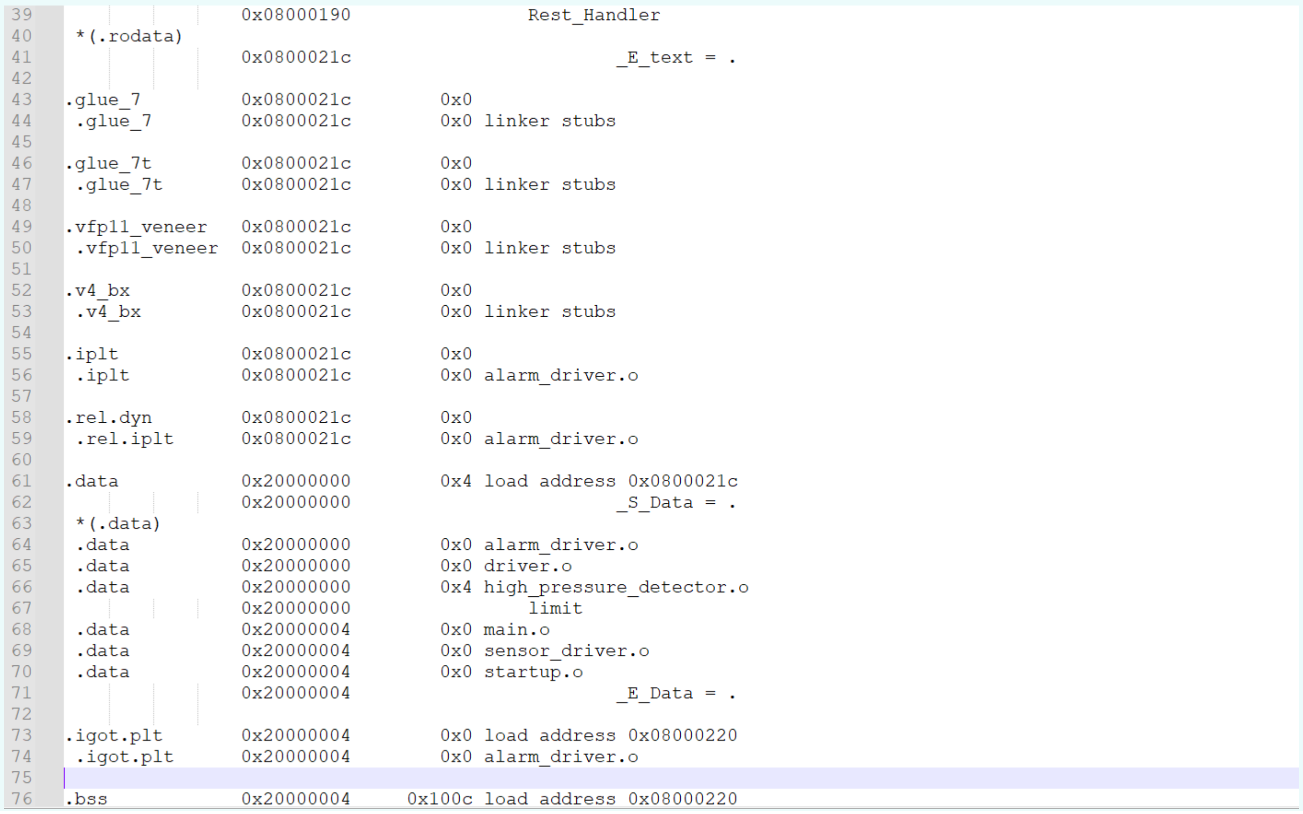


```makefile
CC = arm-none-eabi-
CFLAGS=-mcpu=cortex-m3 -gdwarf-2
INCS=-I .
LIBS=
SRC = $(wildcard *.c)
OBJ = $(SRC:.c=.o)
As = $(wildcard *.s)
As_OBJ = $(As:.s=.o)
PRJ_NAME=pressure_controller

all:$(PRJ_NAME).bin
    @echo "________Build is done__________"

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

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

$(PRJ_NAME).elf :$(OBJ) $(As_OBJ)
    $(CC)ld.exe -T linker_script.ld $(LIBS) $(OBJ) $(As_OBJ) -o $@ -Map=Map_file.map
    cp $(PRJ_NAME).elf $(PRJ_NAME).axf

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

clean:
    rm *.elf *.bin

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

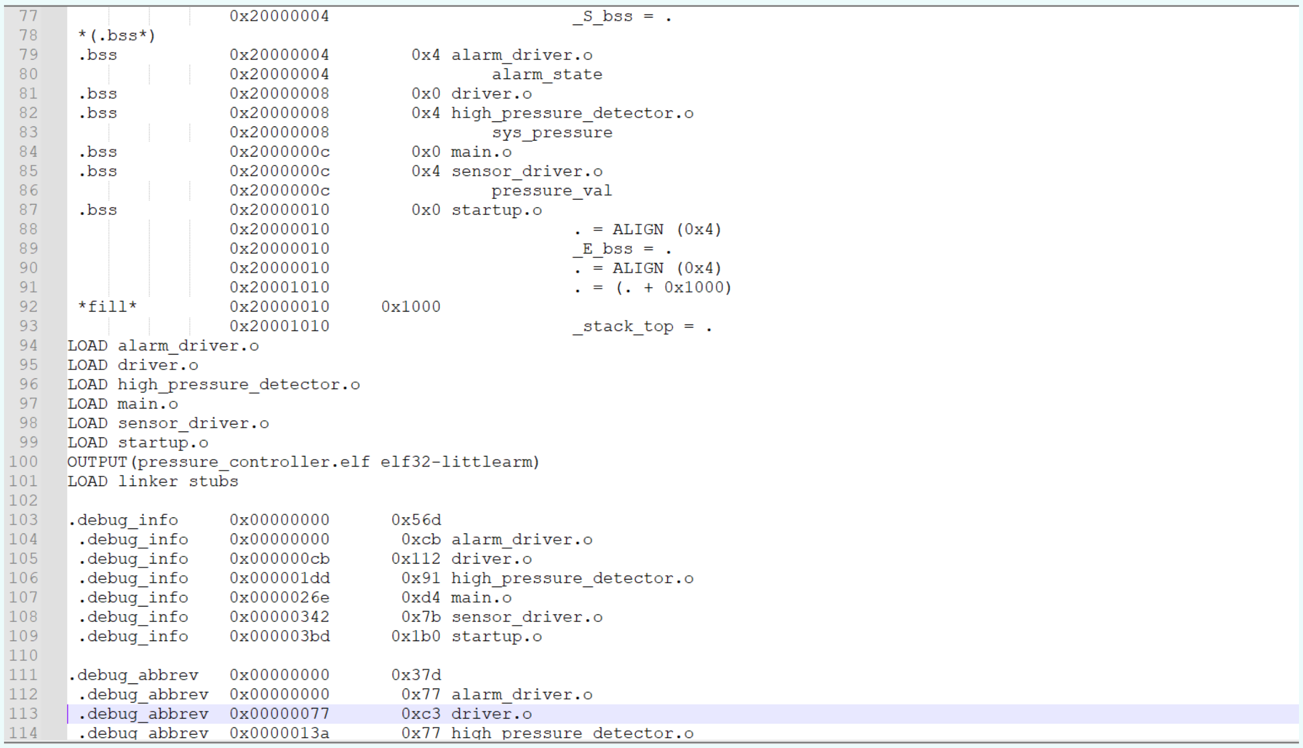
Mapfile file:



```

Memory Configuration

Name             Origin             Length             Attributes
flash            0x08000000         0x00020000         xr
sram             0x20000000         0x00005000         xrw
*default*        0x00000000         0xffffffff

Linker script and memory map


.text           0x08000000      0x21c
 *(.vectors*)
 .vectors       0x08000000       0x1c startup.o
                0x08000000                vectors
 *(.text*)
 .text          0x0800001c       0x30 alarm_driver.o
                0x0800001c                timer_state
 .text          0x0800004c       0xc4 driver.o
                0x0800004c                Delay
                0x0800006e                getPressureVal
                0x08000084                Set_Alarm_actuator
                0x080000c0                GPIO_INITIALIZATION
 .text          0x08000110       0x44 high_pressure_detector.o
                0x08000110                read_from_sensor
                0x0800012c                high_pressure_detect
 .text          0x08000154       0x16 main.o
                0x08000154                main
 *fill*         0x0800016a        0x2
 .text          0x0800016c       0x18 sensor_driver.o
                0x0800016c                pressure_sensor_read
 .text          0x08000184       0x98 startup.o
                0x08000184                MM_Fault_Handler
                0x08000184                Bus_Fault
                0x08000184                Default_Handler
                0x08000184                Usage_Fault_Handler
                0x08000184                NMI_Handeler
                0x08000184                H_Fault_Handler
```



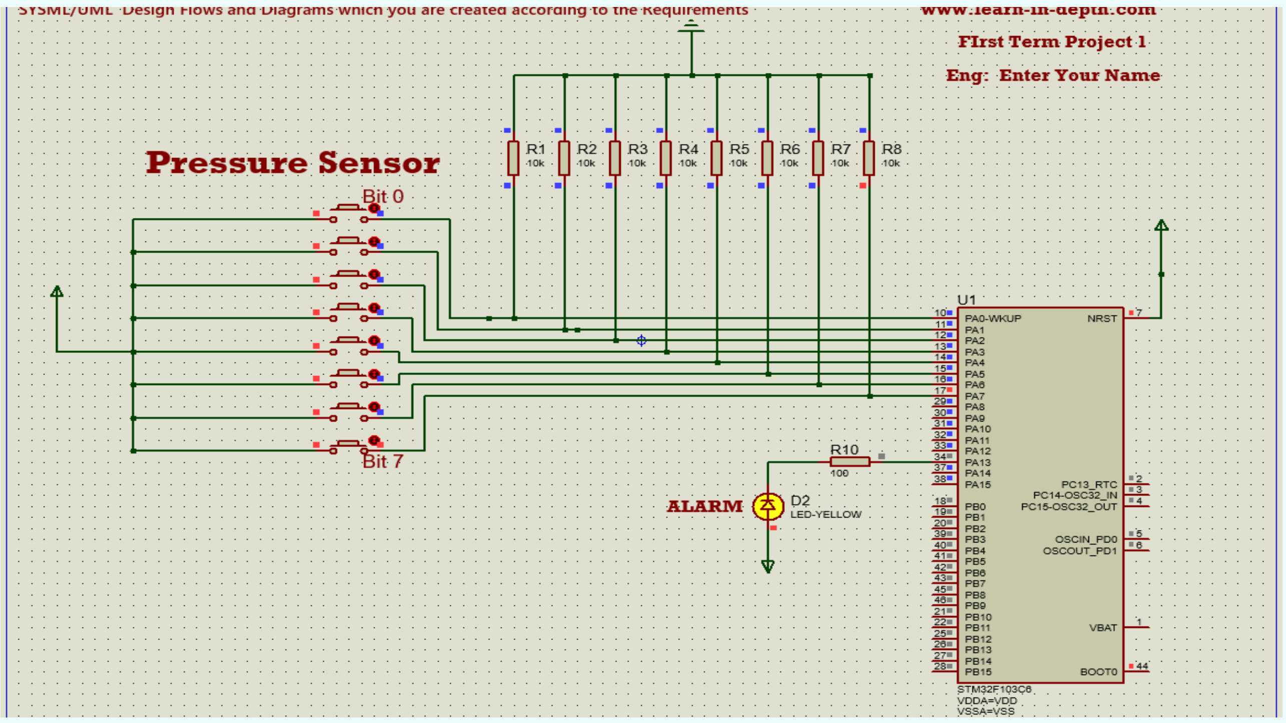
```
                0x08000190                Rest_Handler
 *(.rodata)
                0x0800021c                        _E_text = .

.glue_7         0x0800021c        0x0
 .glue_7        0x0800021c        0x0 linker stubs

.glue_7t        0x0800021c        0x0
 .glue_7t       0x0800021c        0x0 linker stubs

.vfp11_veneer   0x0800021c        0x0
 .vfp11_veneer  0x0800021c        0x0 linker stubs

.v4_bx          0x0800021c        0x0
 .v4_bx         0x0800021c        0x0 linker stubs

.iplt           0x0800021c        0x0
 .iplt          0x0800021c        0x0 alarm_driver.o

.rel.dyn        0x0800021c        0x0
 .rel.iplt      0x0800021c        0x0 alarm_driver.o

.data           0x20000000        0x4 load address 0x0800021c
                0x20000000                        _S_Data = .
 *(.data)
 .data          0x20000000        0x0 alarm_driver.o
 .data          0x20000000        0x0 driver.o
 .data          0x20000000        0x4 high_pressure_detector.o
                0x20000000                limit
 .data          0x20000004        0x0 main.o
 .data          0x20000004        0x0 sensor_driver.o
 .data          0x20000004        0x0 startup.o
                0x20000004                        _E_Data = .

.igot.plt       0x20000004        0x0 load address 0x08000220
 .igot.plt      0x20000004        0x0 alarm_driver.o

.bss            0x20000004      0x100c load address 0x08000220
```



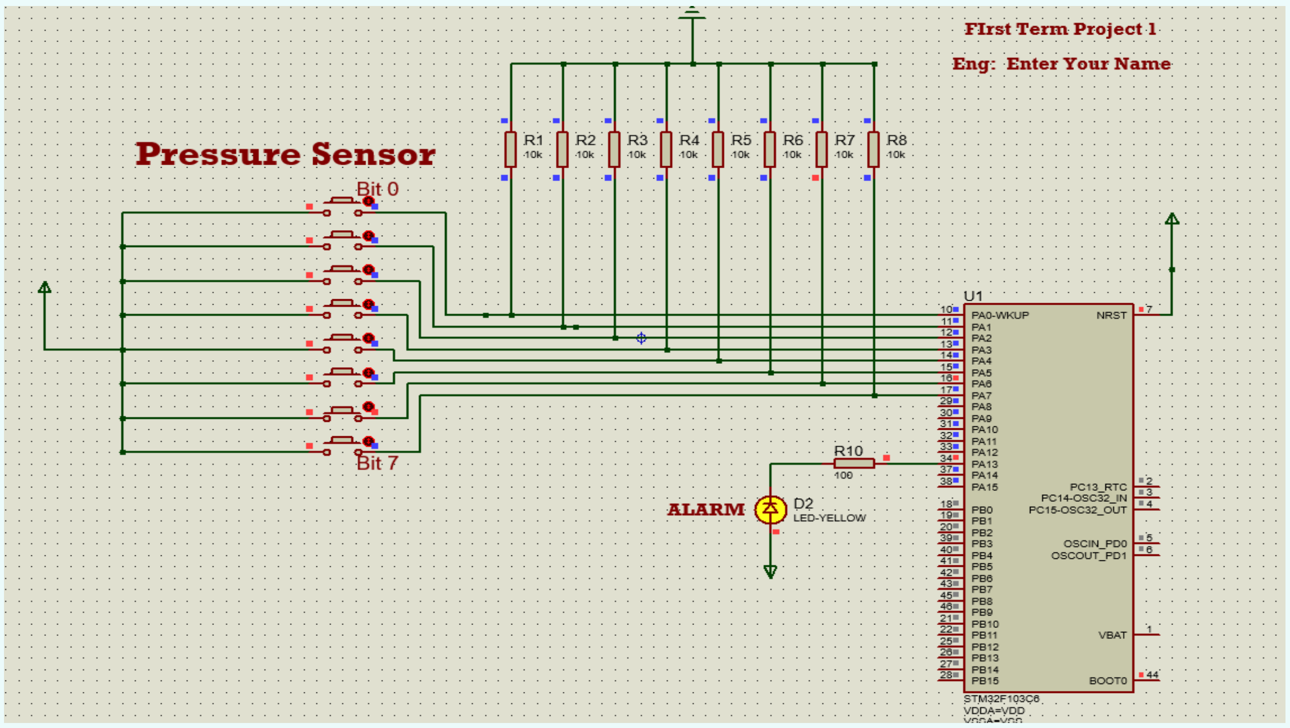
```
                0x20000004                      _S_bss = .
 *(.bss*)
 .bss           0x20000004        0x4 alarm_driver.o
                0x20000004                alarm_state
 .bss           0x20000008        0x0 driver.o
 .bss           0x20000008        0x4 high_pressure_detector.o
                0x20000008                sys_pressure
 .bss           0x2000000c        0x0 main.o
 .bss           0x2000000c        0x4 sensor_driver.o
                0x2000000c                pressure_val
 .bss           0x20000010        0x0 startup.o
                0x20000010                      . = ALIGN (0x4)
                0x20000010                      _E_bss = .
                0x20000010                      . = ALIGN (0x4)
                0x20001010                      . = (. + 0x1000)
 *fill*         0x20000010     0x1000
                0x20001010                      _stack_top = .
LOAD alarm_driver.o
LOAD driver.o
LOAD high_pressure_detector.o
LOAD main.o
LOAD sensor_driver.o
LOAD startup.o
OUTPUT(pressure_controller.elf elf32-littlearm)
LOAD linker stubs

.debug_info     0x00000000      0x56d
 .debug_info    0x00000000       0xcb alarm_driver.o
 .debug_info    0x000000cb      0x112 driver.o
 .debug_info    0x000001dd       0x91 high_pressure_detector.o
 .debug_info    0x0000026e       0xd4 main.o
 .debug_info    0x00000342       0x7b sensor_driver.o
 .debug_info    0x000003bd      0x1b0 startup.o

.debug_abbrev   0x00000000      0x37d
 .debug_abbrev  0x00000000       0x77 alarm_driver.o
 .debug_abbrev  0x00000077       0xc3 driver.o
 .debug_abbrev  0x0000013a       0x77 high_pressure_detector.o
```

Simulation Results:

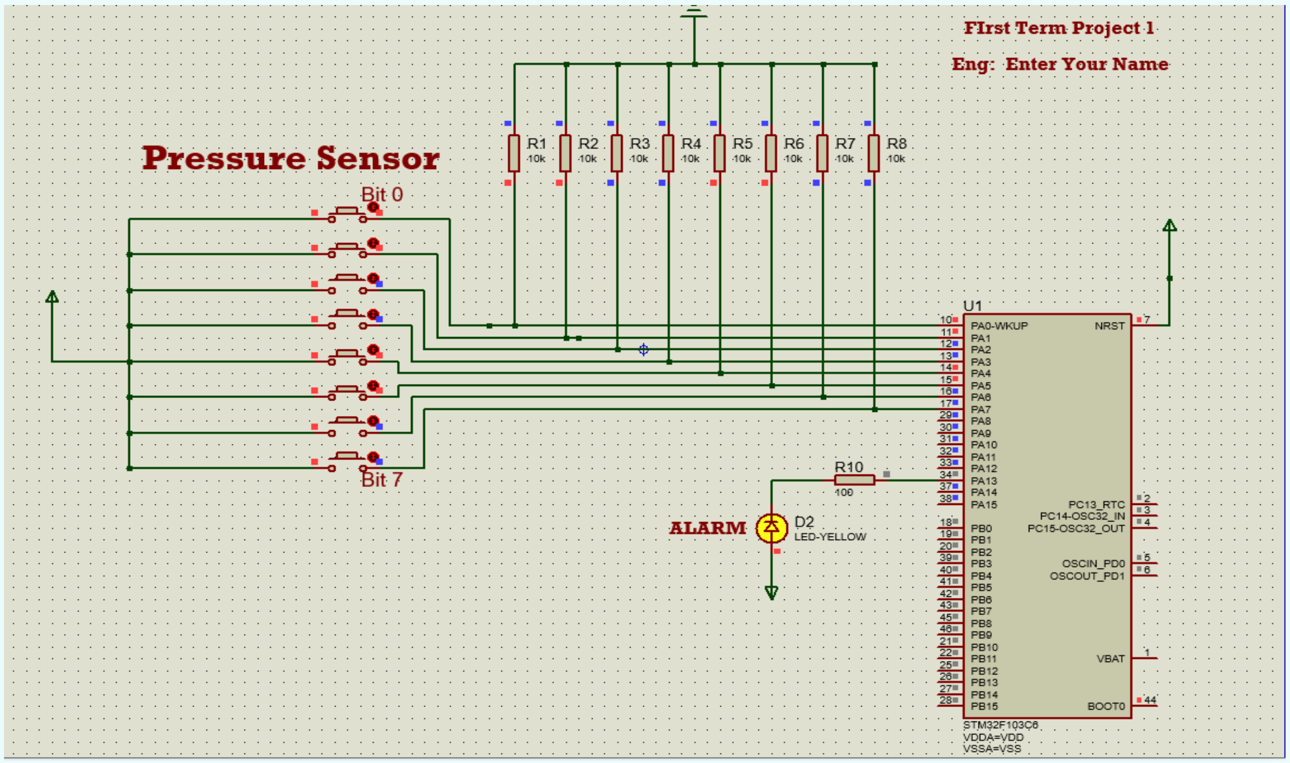
When pressure =128



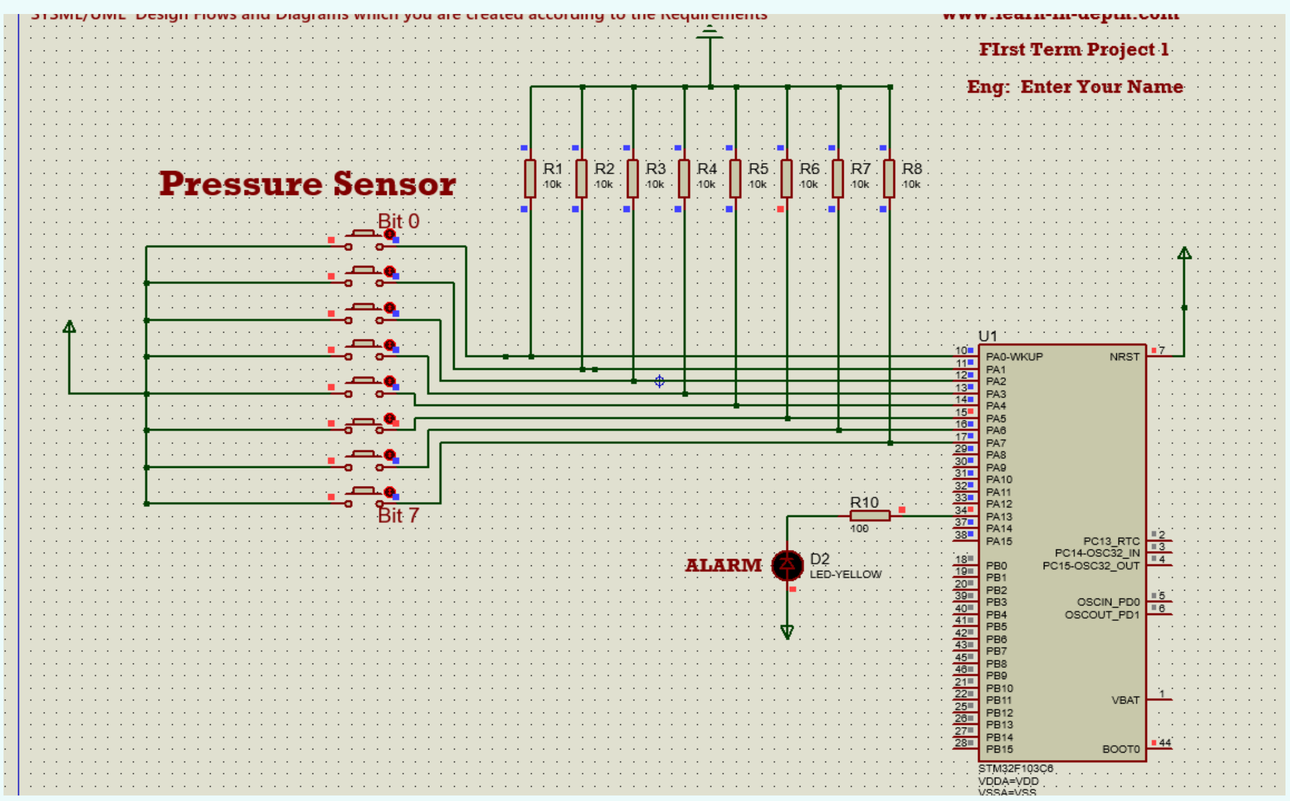
When pressure =64



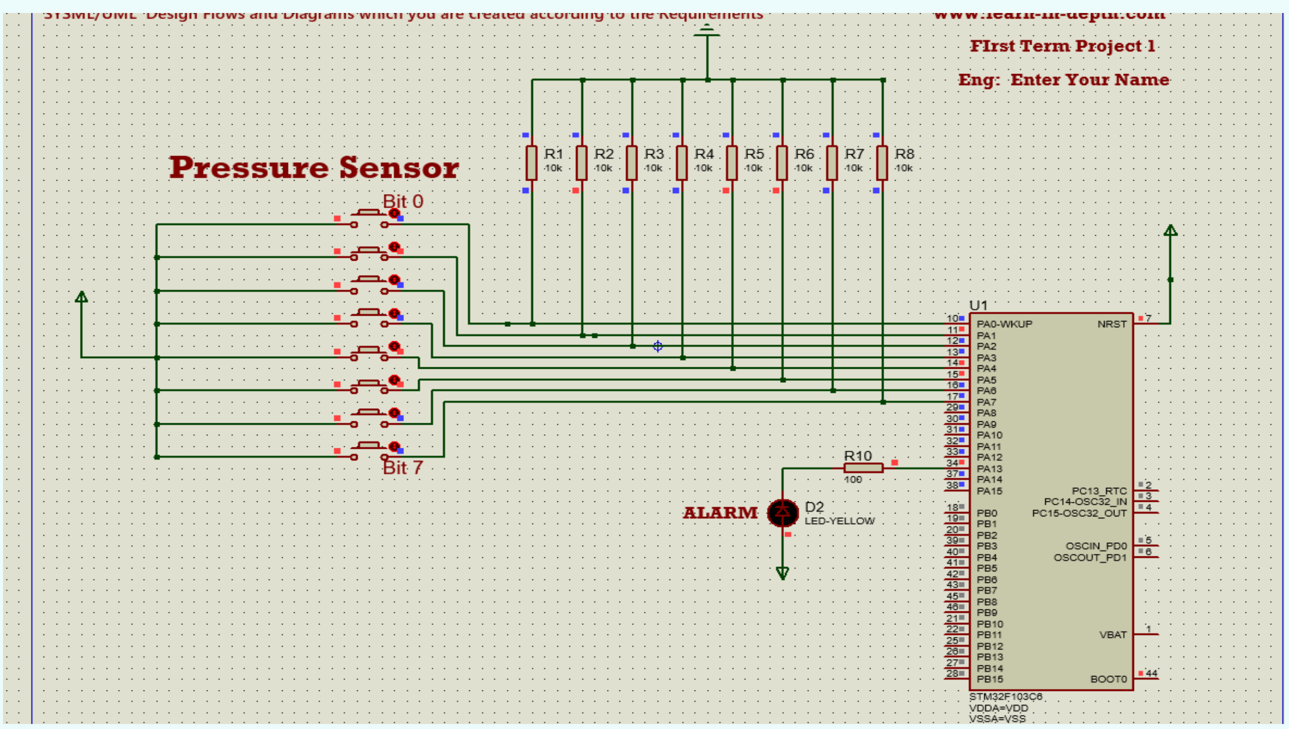
When pressure =51



When pressure =50



When pressure =32


SYSML/UML Design Flows and Diagrams which you are created according to the Requirements
www.learn-in-depth.com
FIrst Term Project 1
Eng: Enter Your Name
Pressure Sensor
Bit 0
Bit 7
R1 10k
R2 10k
R3 10k
R4 10k
R5 10k
R6 10k
R7 10k
R8 10k
R10 100
ALARM
D2 LED-YELLOW
U1
STM32F103C6
VDDA=VDD
VSSA=VSS