Pressure Detection System

BY

Mohamed Eid

Mastering Embedded System Online Diploma

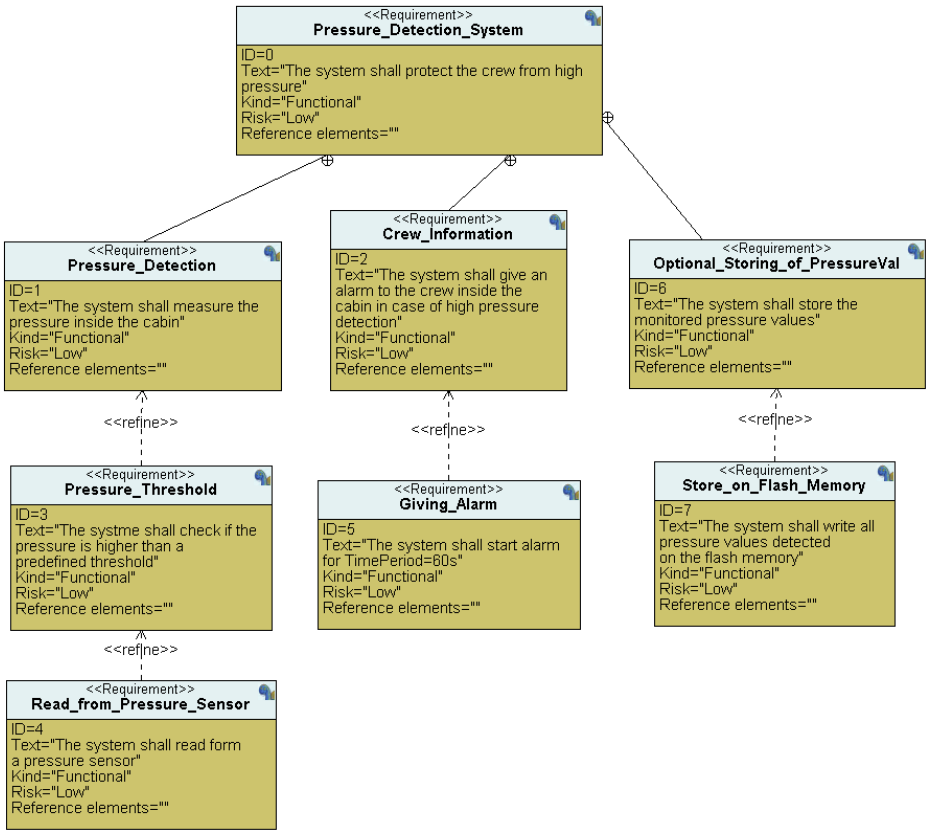
Progress Page

https://www.learn-in-depth.com/online-diploma/m0hamed.3ed98%40gmail.com

CASE STUDY

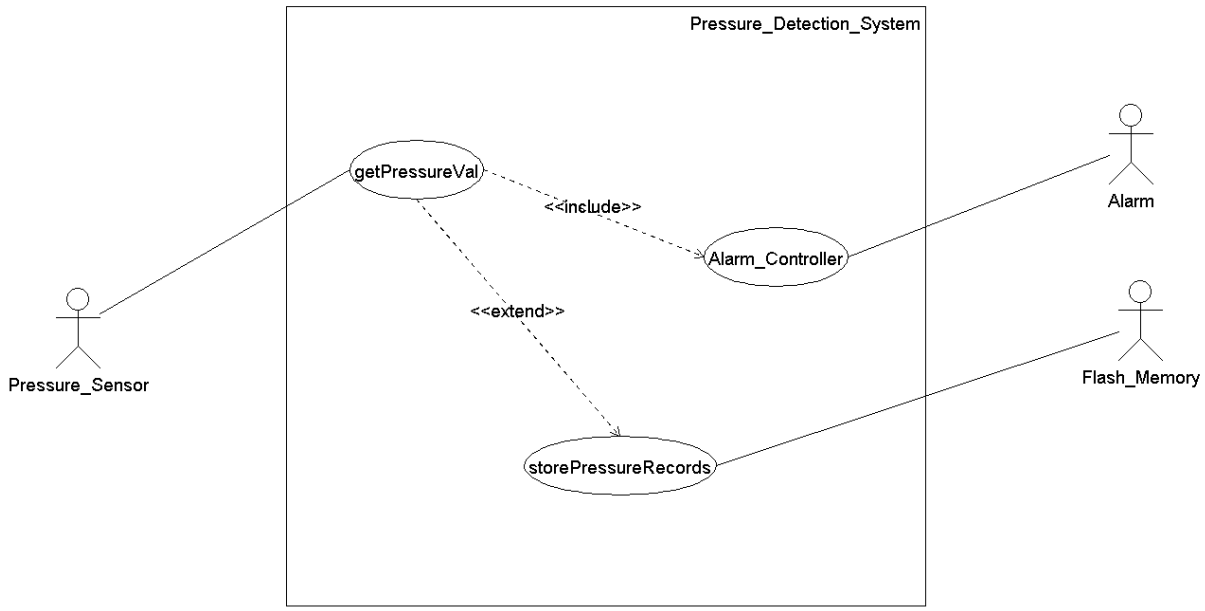
A pressure detection system aims to inform the crew of a cabin with an alarm, typically an LED, when the pressure exceeds 20 bars in the cabin.

REQUIREMENT DIAGRAM

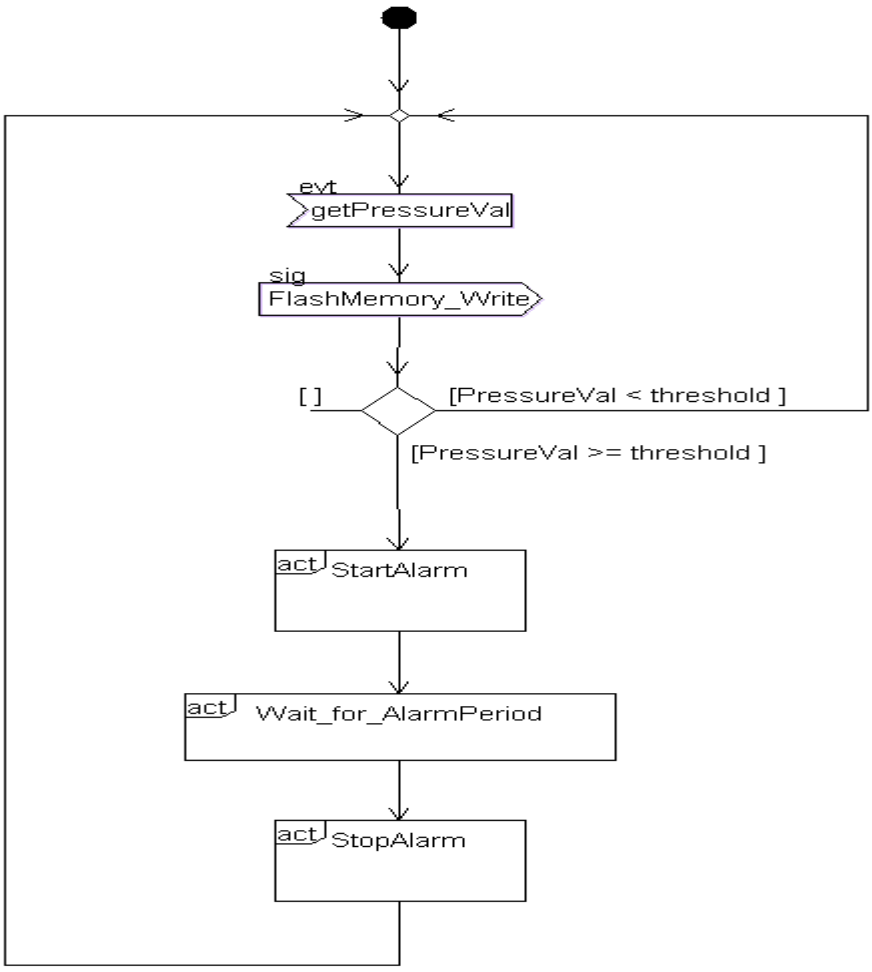


SYSTEM ANALYSIS

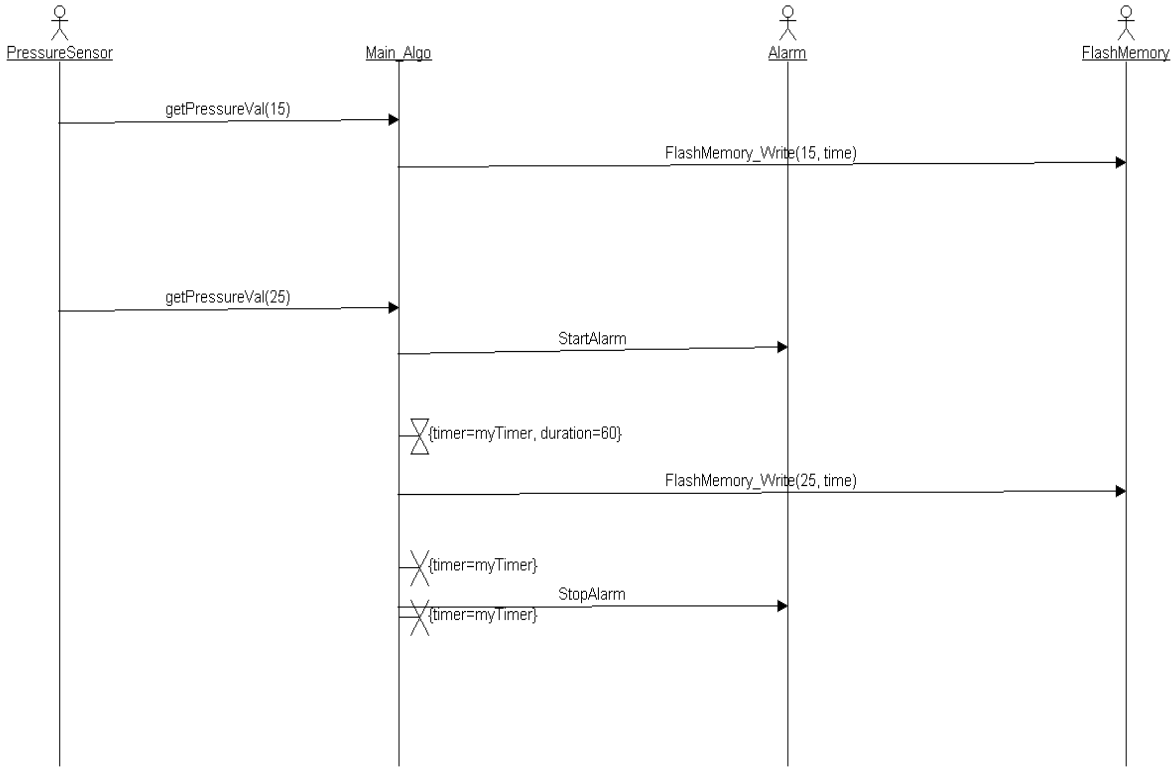
Use Case Diagram



Activity Diagram

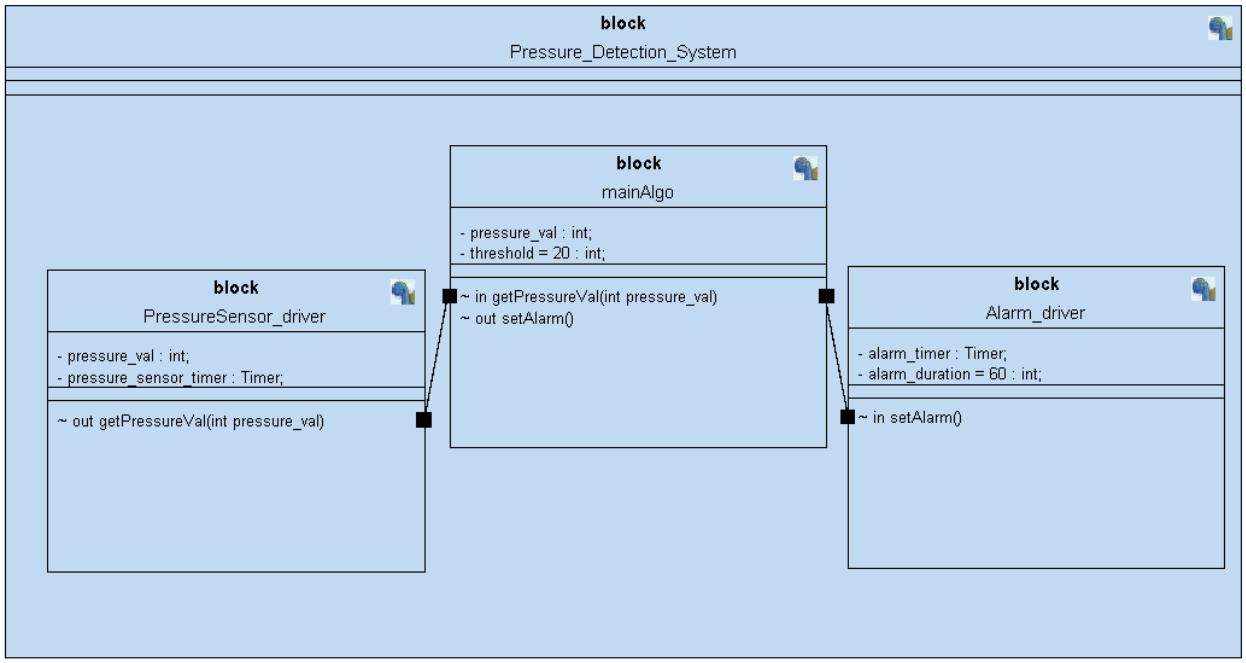


Sequence Diagram

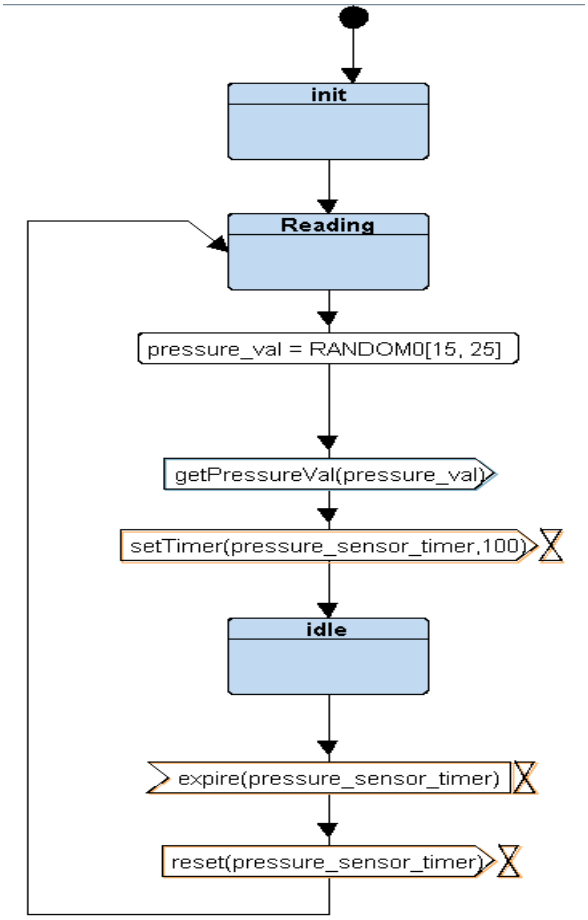


SYSTEM DESIGN

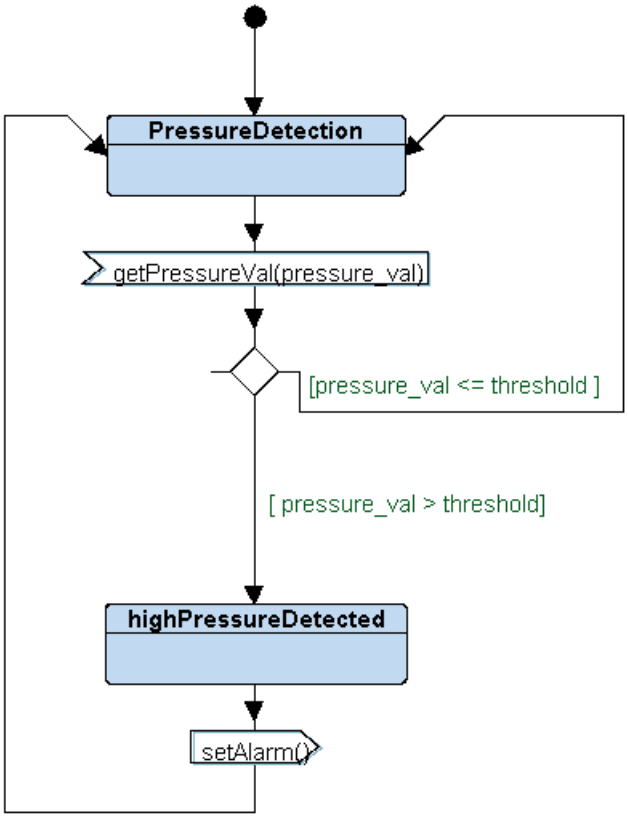
The system consists of three modules as follows:



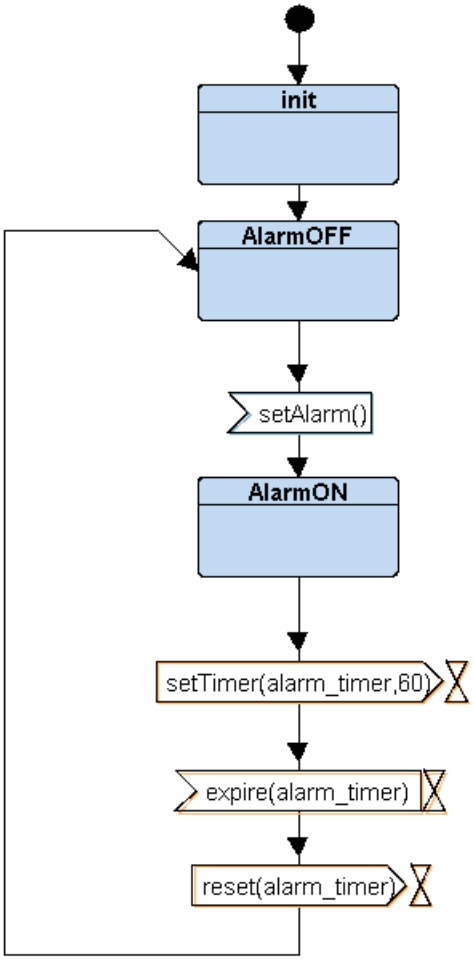
PressureSensor_driver



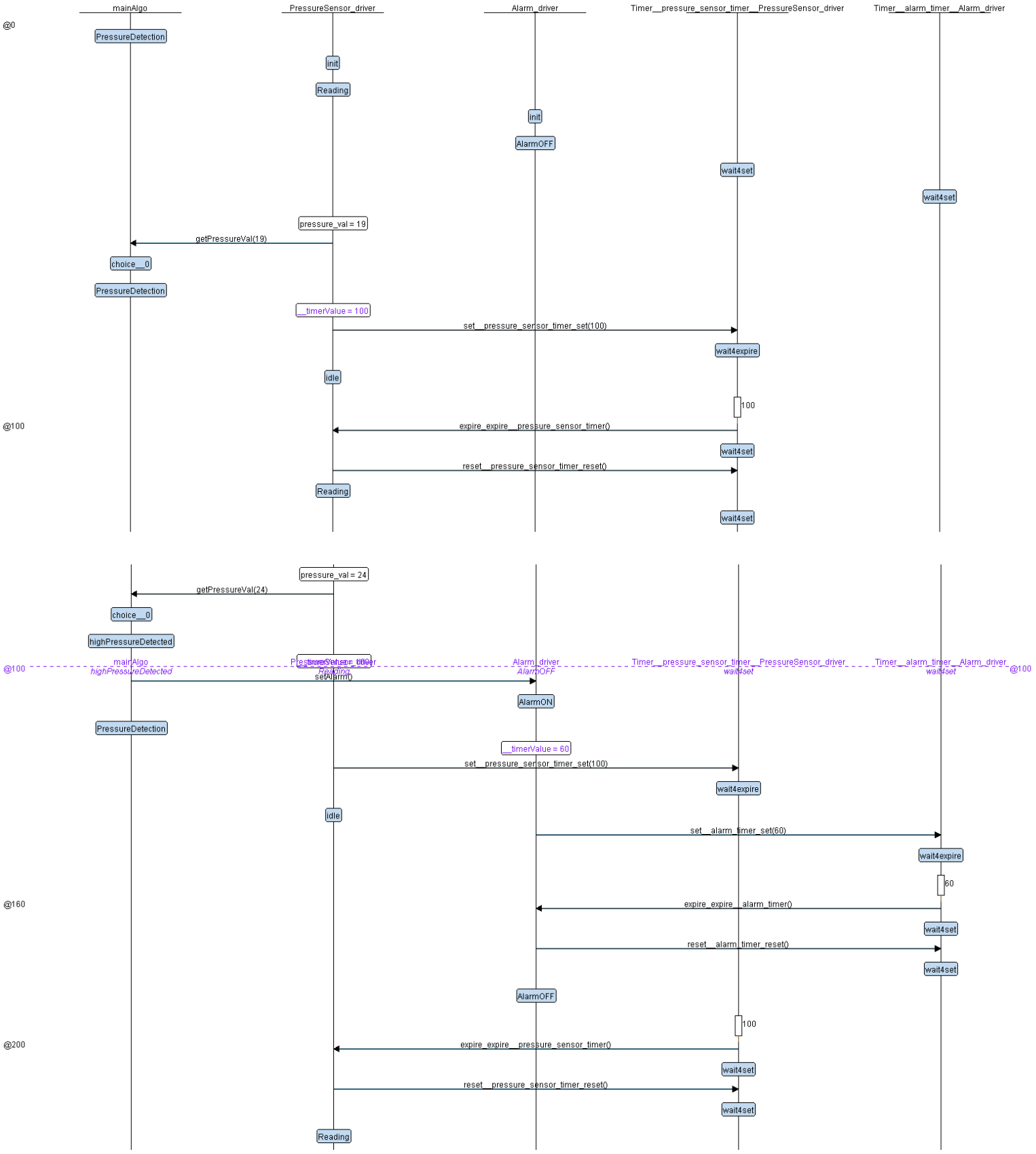
mainAlog



Alarm_driver

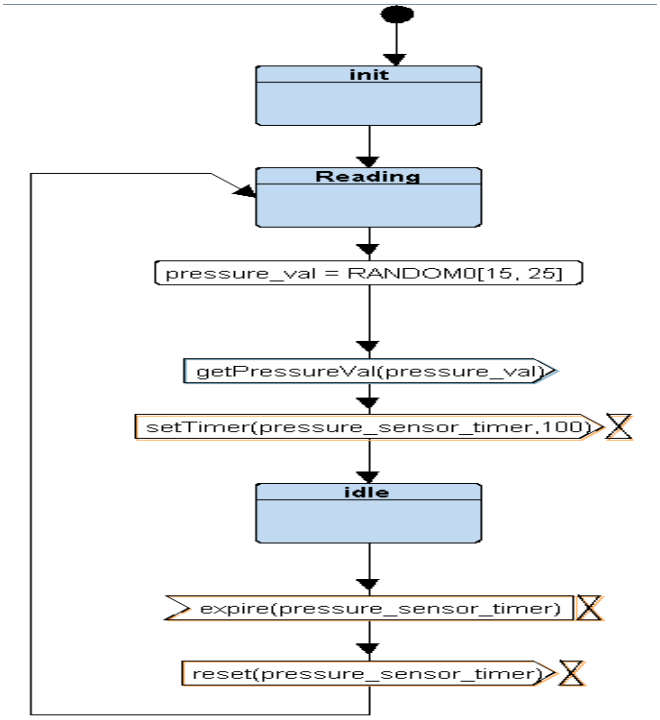


TEST DESIGN BEFORE IMPLEMENTATION


mainAlgo
PressureSensor_driver
Alarm_driver
Timer__pressure_sensor_timer__PressureSensor_driver
Timer__alarm_timer__Alarm_driver
@0
PressureDetection
init
Reading
init
AlarmOFF
wait4set
wait4set
pressure_val = 19
getPressureVal(19)
choice__0
PressureDetection
__timerValue = 100
set__pressure_sensor_timer_set(100)
wait4expire
idle
100
@100
expire_expire__pressure_sensor_timer()
wait4set
reset__pressure_sensor_timer_reset()
Reading
wait4set
pressure_val = 24
getPressureVal(24)
choice__0
highPressureDetected
mainAlgo
highPressureDetected
Alarm_driver
AlarmOFF
Timer__pressure_sensor_timer__PressureSensor_driver
wait4set
Timer__alarm_timer__Alarm_driver
wait4set
@100
@100
setAlarm()
AlarmON
PressureDetection
__timerValue = 60
set__pressure_sensor_timer_set(100)
wait4expire
idle
set__alarm_timer_set(60)
wait4expire
60
@160
expire_expire__alarm_timer()
wait4set
reset__alarm_timer_reset()
wait4set
AlarmOFF
100
@200
expire_expire__pressure_sensor_timer()
wait4set
reset__pressure_sensor_timer_reset()
wait4set
Reading

CODES

Pressure Sensor Module



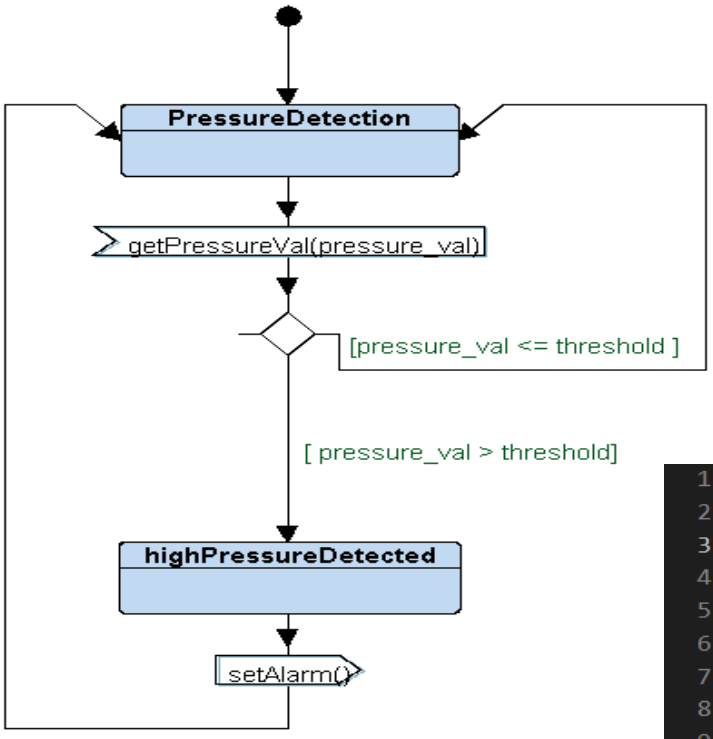

```h
#ifndef PRESSURE_SENSOR_H_
#define PRESSURE_SENSOR_H_

#include "state.h"

enum
{
    PS_reading,
    PS_idle
}PS_state_id;

extern void (*PS_state)();

STATE_define(PS_reading);
STATE_define(PS_idle);
void PS_init();


#endif
```

Pressure_sensor.h

```c
#include "Pressure_Sensor.h"


int PS_pressure_val = 0;

void (*PS_state)();

void PS_init()
{
    //printf("Pressure Sensor init!");
}

STATE_define(PS_reading)
{
    PS_state_id = PS_reading;
    PS_pressure_val = getPressureVal();
    PressureVal_signal(PS_pressure_val);
    PS_state = STATE(PS_idle);
}

STATE_define(PS_idle)
{
    PS_state_id = PS_idle;
    Delay(100000);
    PS_state = STATE(PS_reading);
}
```

pressure_sensor.c

Pressure Detection Module



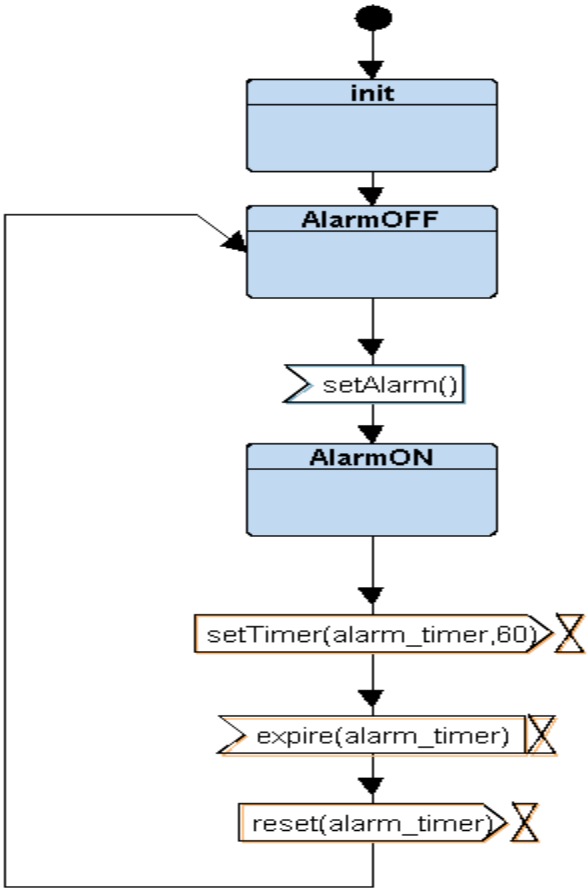

```c
#ifndef PRESSURE_DETECTION_H_
#define PRESSURE_DETECTION_H_

#include "state.h"

enum
{
    PD_pressure_detection,
    PD_high_pressure_detected

}PD_state_id;

STATE_define(PD_pressure_detection);
STATE_define(PD_high_pressure_detected);

extern void (*PD_state)();

#endif
```

Pressure_Detection.h

```c
#include "Pressure_Dtection.h"

int PD_threshold = 20;

void (*PD_state)();

void PressureVal_signal(int PD_pressure_val)
{
    if (PD_pressure_val > PD_threshold)
    {
        PD_state = STATE(PD_high_pressure_detected);
    }
    else
    {
        PD_state = STATE(PD_pressure_detection);
    }
}

STATE_define(PD_pressure_detection)
{
    PD_state_id = PD_pressure_detection;
    PD_state = STATE(PD_pressure_detection);
}

STATE_define(PD_high_pressure_detected)
{
    PD_state_id = PD_high_pressure_detected;
    setAlarm_signal();
    PD_state = STATE(PD_pressure_detection);
}
```

Pressure_Detection.c

Alarm Driver Module




```c
#include "Alarm_Driver.h"


void (*AL_state)();

void AL_init()
{
    //printf("Alarm init");
}

void setAlarm_signal()
{
    AL_state = STATE(AL_ON);
}


STATE_define(AL_ON)
{
    AL_state_id = AL_ON;
    Set_Alarm_actuator(0);
    Delay(100000);
    AL_state = STATE(AL_OFF);
}
STATE_define(AL_OFF)
{
    AL_state_id = AL_OFF;
    Set_Alarm_actuator(1);
    Delay(100000);

}
```

Alarm_Driver.c

```c
#ifndef ALARM_DRIVER_H_
#define ALARM_DRIVER_H_

#include "state.h"

extern void (*AL_state)();

enum
{
    AL_ON,
    AL_OFF
}AL_state_id;

void AL_init();
STATE_define(AL_ON);
STATE_define(AL_OFF);

#endif
```

Alarm_Driver.h

Makefile

Makefile

```
# @Author Eng. Mohamed Eid

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

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

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

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

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

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


clean:
    rm *.elf *.bin *.map

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

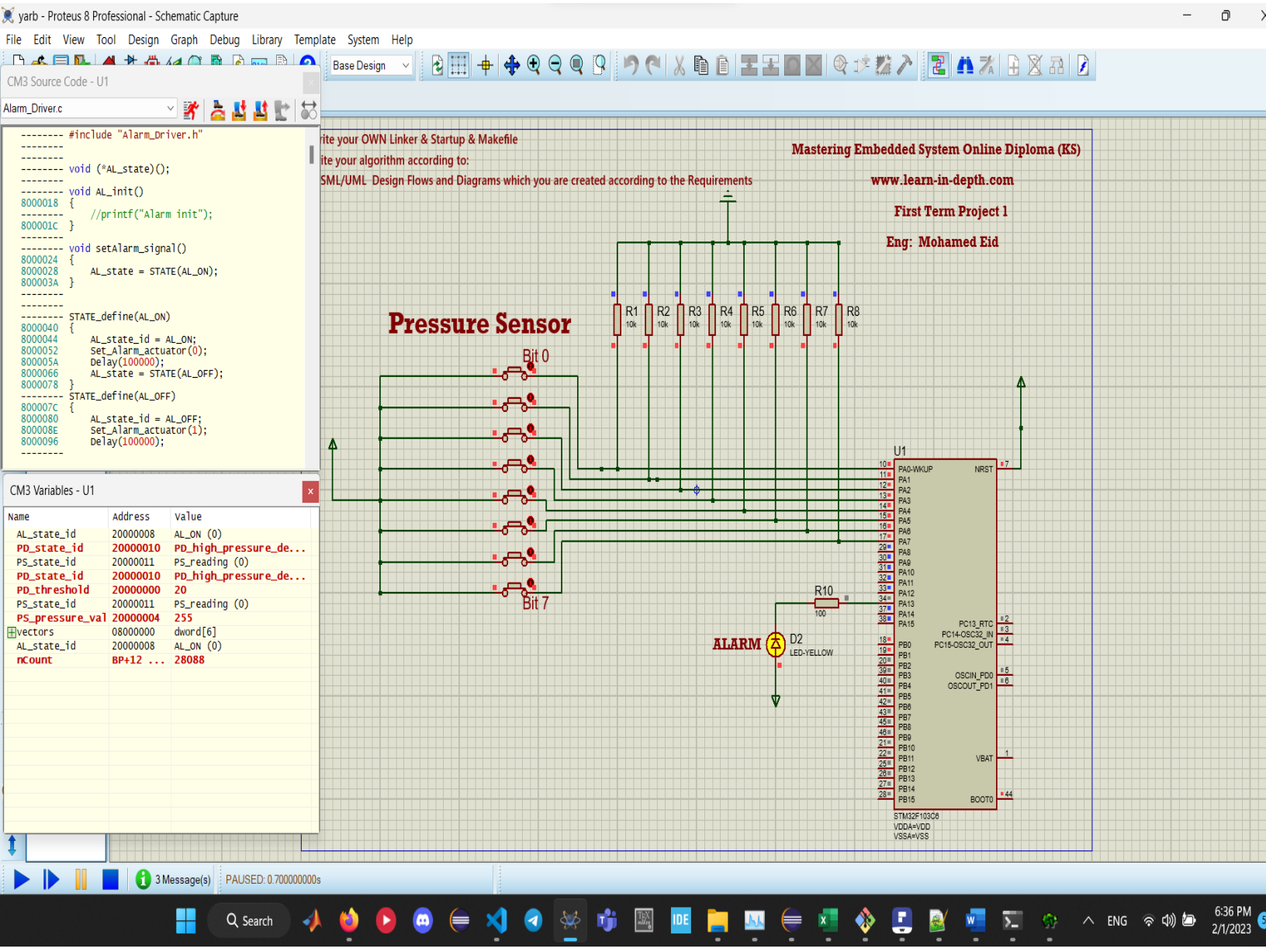
Linker Script

```
    @Author Eng. Mohamed Eid */
MEMORY
{
    flash(RX) : ORIGIN = 0x08000000, LENGTH = 128K
    sram(RWX) : ORIGIN = 0x20000000, LENGTH = 20K
}
SECTIONS
{
    .text : {
        *(.vectors)
        *(.text)
        _E_TEXT = .;
    } > flash

    .rodata : {
        *(.rodata)
    } > flash

    .data : {
        _S_DATA = .;
        *(.data)
        _E_DATA = .;
    } > sram AT> flash

    .bss : {
        _S_BSS = .;
        *(.bss)
        . = ALIGN(4);
        _E_BSS = .;
    } > sram

    .comment : {
        *(COMMON)
        *(*fill*)
        *(.comment)

    } > sram

    . = . + 0x1000;
    _STACK_TOP = .;
}
```

Startup.c

```c
#include <stdint.h>

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;

extern int main();


void reset_handler()
{
    int i;
    uint32_t data_size = (unsigned char*)&_E_DATA - (unsigned char*)&_S_DATA;
    unsigned char* p_scr = (unsigned char*)&_E_TEXT;
    unsigned char* p_dst = (unsigned char*)&_S_DATA;
    for(i = 0 ; i < data_size ; i++)
    {
        *((unsigned char*) p_dst) = *((unsigned char*) p_scr);
    }

    uint32_t bss_size = (unsigned char*)&_E_BSS - (unsigned char*)&_S_BSS;

    for(i = 0 ; i < bss_size ; i++)
    {
        p_dst = (unsigned char*)_S_BSS;
        *((unsigned char*) p_dst++) = (unsigned char)0;
    }


    main();
}

void default_handler(void)
{
    reset_handler();
}

void NMI_handler(void)__attribute__((weak, alias("default_handler")));
void MM_Fault_handler(void)__attribute__((weak, alias("default_handler")));
void BusFault(void)__attribute__((weak, alias("default_handler")));
void UsageFault(void)__attribute__((weak, alias("default_handler")));

uint32_t vectors[]__attribute__((section(".vectors"))) = {
    (uint32_t) &_STACK_TOP,
    (uint32_t) &reset_handler,
    (uint32_t) &NMI_handler,
    (uint32_t) &MM_Fault_handler,
    (uint32_t) &BusFault,
    (uint32_t) &UsageFault
};

```

SIMULATION AFTER IMPLEMENTATION



SW ANALYSIS

Symbol Table

```
20000008 B _E_BSS
20000004 D _E_DATA
08000400 T _E_TEXT
20000004 B _S_BSS
20000000 D _S_DATA
2000102d D _STACK_TOP
08000018 T AL_init
2000000c D AL_state
20000008 D AL_state_id
080003f4 W BusFault
080003f4 T default_handler
080000a4 T Delay
080000c8 T getPressureVal
08000130 T GPIO_INITIALIZATION
080001f4 T main
080003f4 W MM_Fault_handler
080003f4 W NMI_handler
20000014 D PD_state
20000010 D PD_state_id
20000000 D PD_threshold
08000228 T PressureVal_signal
080002c8 T PS_init
20000004 B PS_pressure_val
20000018 D PS_state
20000011 D PS_state_id
08000350 T reset_handler
080000e0 T Set_Alarm_actuator
08000024 T setAlarm_signal
080001b0 T setup
0800007c T ST_AL_OFF
08000040 T ST_AL_ON
0800029c T ST_PD_high_pressure_detected
08000270 T ST_PD_pressure_detection
0800031c T ST_PS_idle
080002d4 T ST_PS_reading
080003f4 W UsageFault
08000000 T vectors
```

Sections Table

```
Pressure_Detection_System.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000400  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000004  20000000  08000400  00010000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000004  20000004  08000404  00010004  2**2
                  ALLOC
  3 .comment      00000025  20000008  08000404  00010008  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  4 .ARM.attributes 00000033  00000000  00000000  0001002d  2**0
                  CONTENTS, READONLY
  5 .debug_info   00000706  00000000  00000000  00010060  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_abbrev 000003e3  00000000  00000000  00010766  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_loc    00000348  00000000  00000000  00010b49  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_aranges 000000c0  00000000  00000000  00010e91  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .debug_line   000002fb  00000000  00000000  00010f51  2**0
                  CONTENTS, READONLY, DEBUGGING
 10 .debug_str    000002d8  00000000  00000000  0001124c  2**0
                  CONTENTS, READONLY, DEBUGGING
 11 .debug_frame  00000244  00000000  00000000  00011524  2**2
                  CONTENTS, READONLY, DEBUGGING

```

Comment

An interesting notice here is that I have only 4 bytes in both .data and .bss sections and this is because I only have an initialized global uint32_t variable (PD_thershold) in Pressure_Detection.c file which will come out in .data section.

And also I have an uninitialized global uint32_t variable in (PS_pressure_val) in Pressure_Sensor.c which will come out at .bss section.