High Pressure Detection System Report

Embedded Systems Online Diploma

Eng. Mahmoud Essam Mahfouz

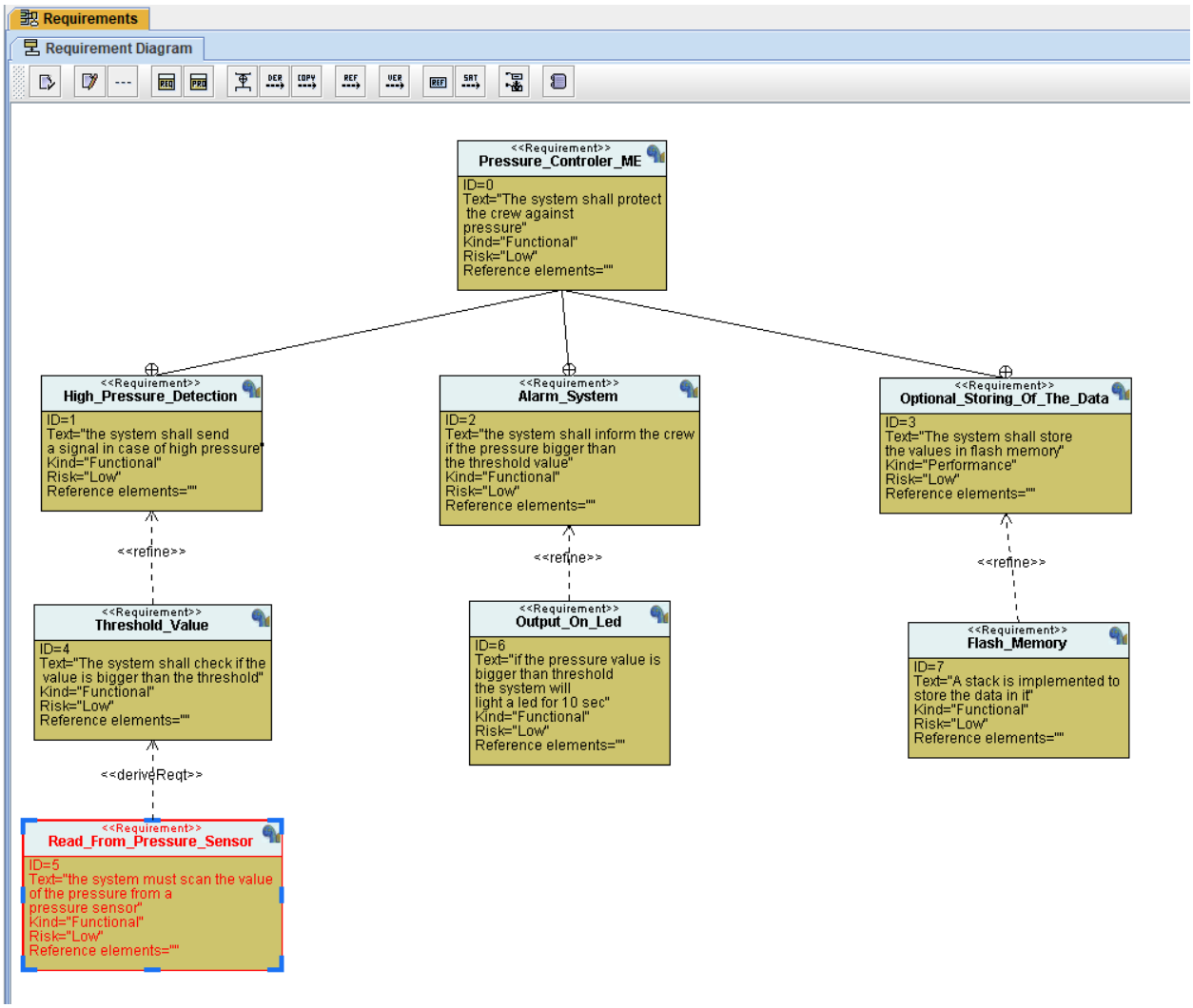
Progress Page

6/12/2022

System Requirements :

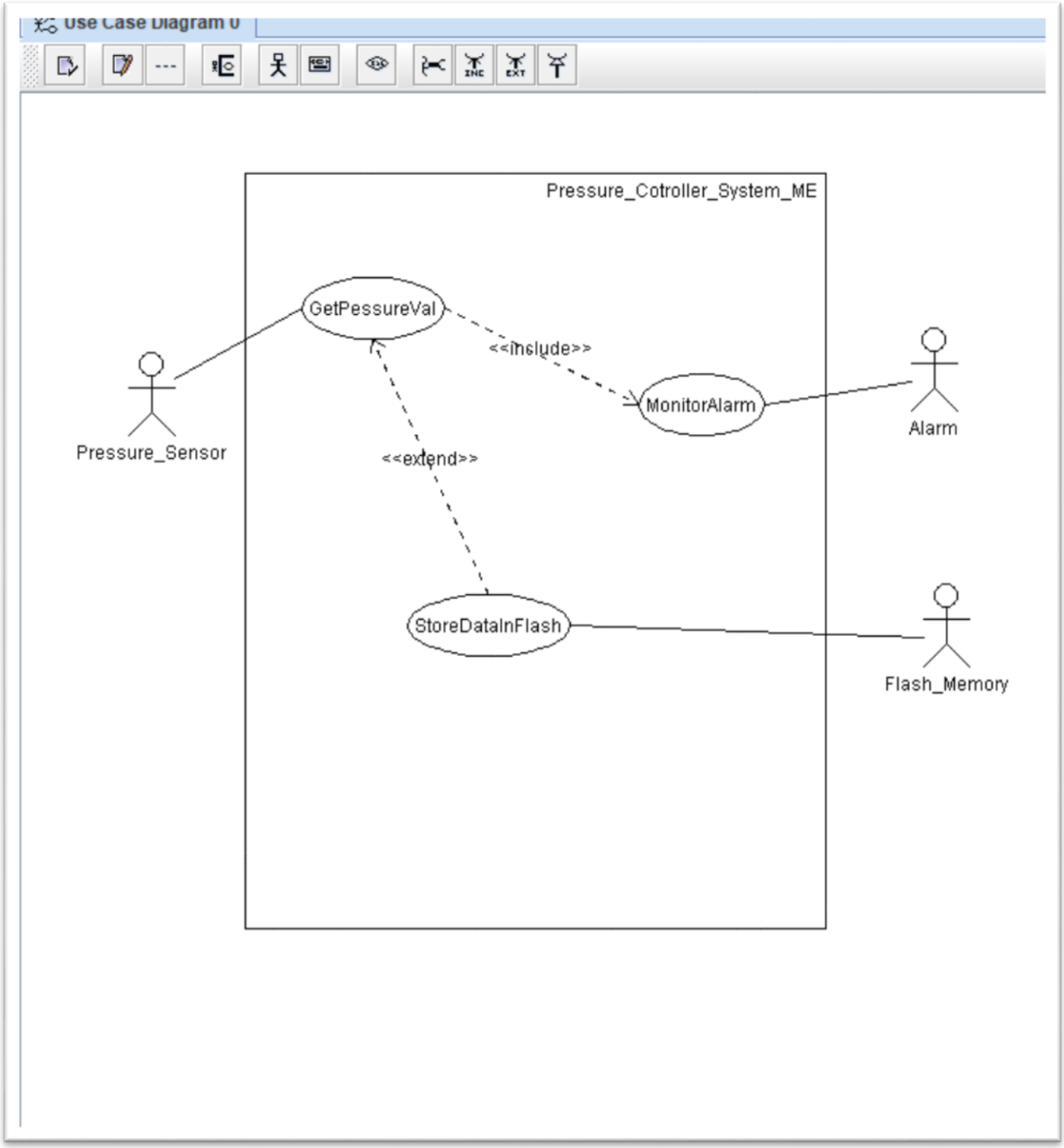
1. The System Shall Measure the pressure through a pressure sensor
2. If the pressure value is greater than threshold system will inform the crew
3. The system will inform the crew threw a led that will on for 60 seconds

Requirement Diagram

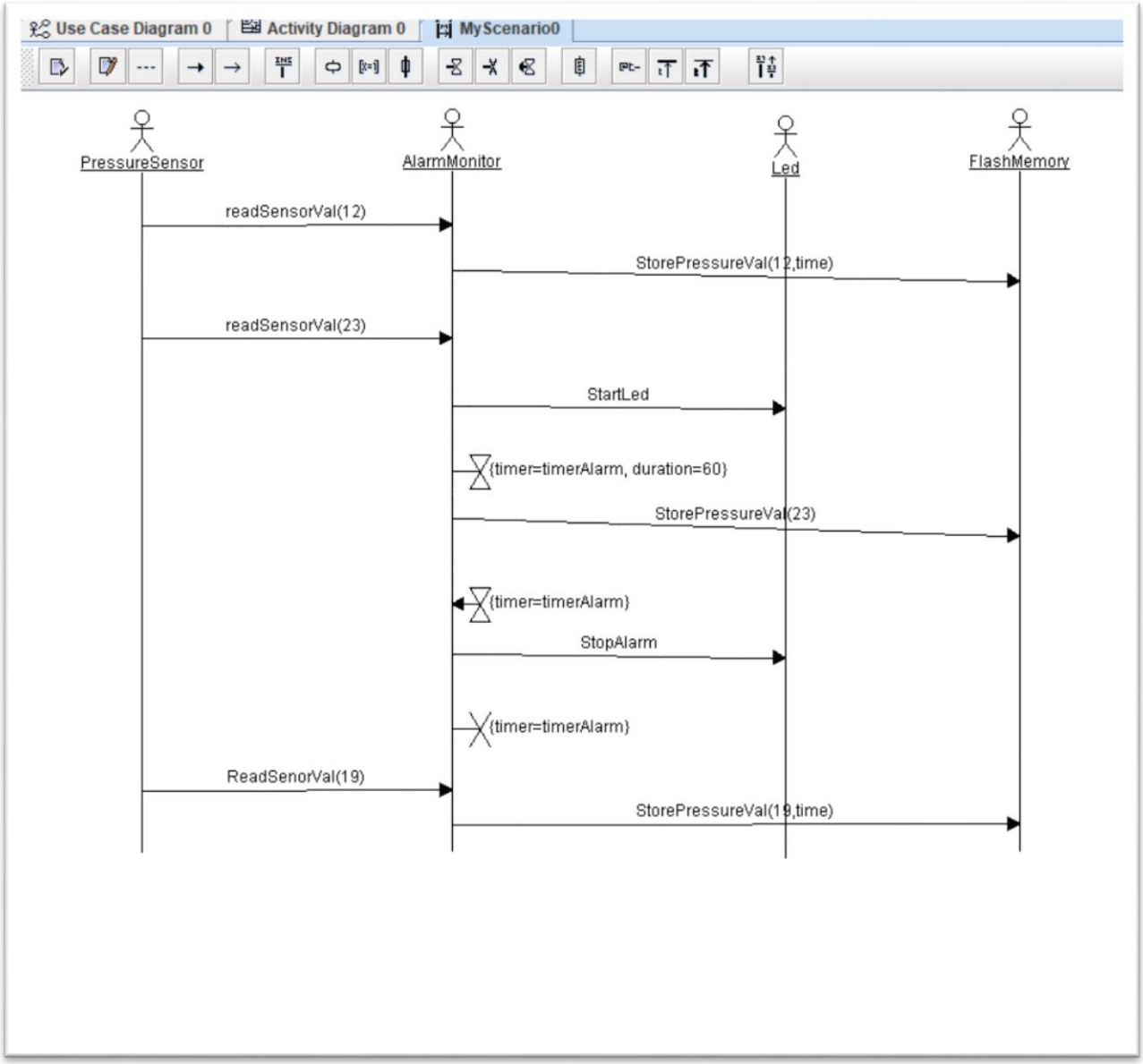


System Analysis :

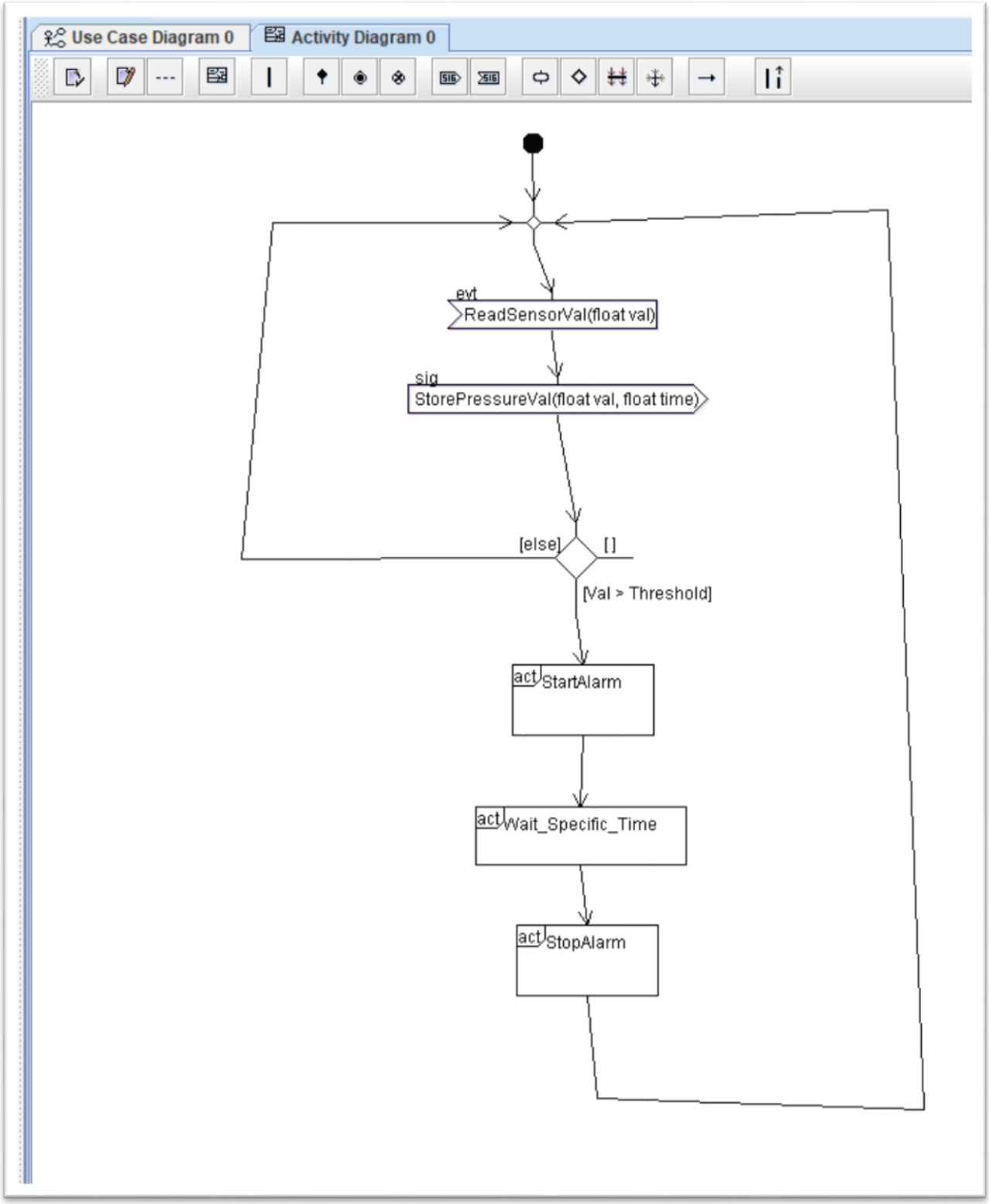
Use Case Diagram



Sequence Diagram

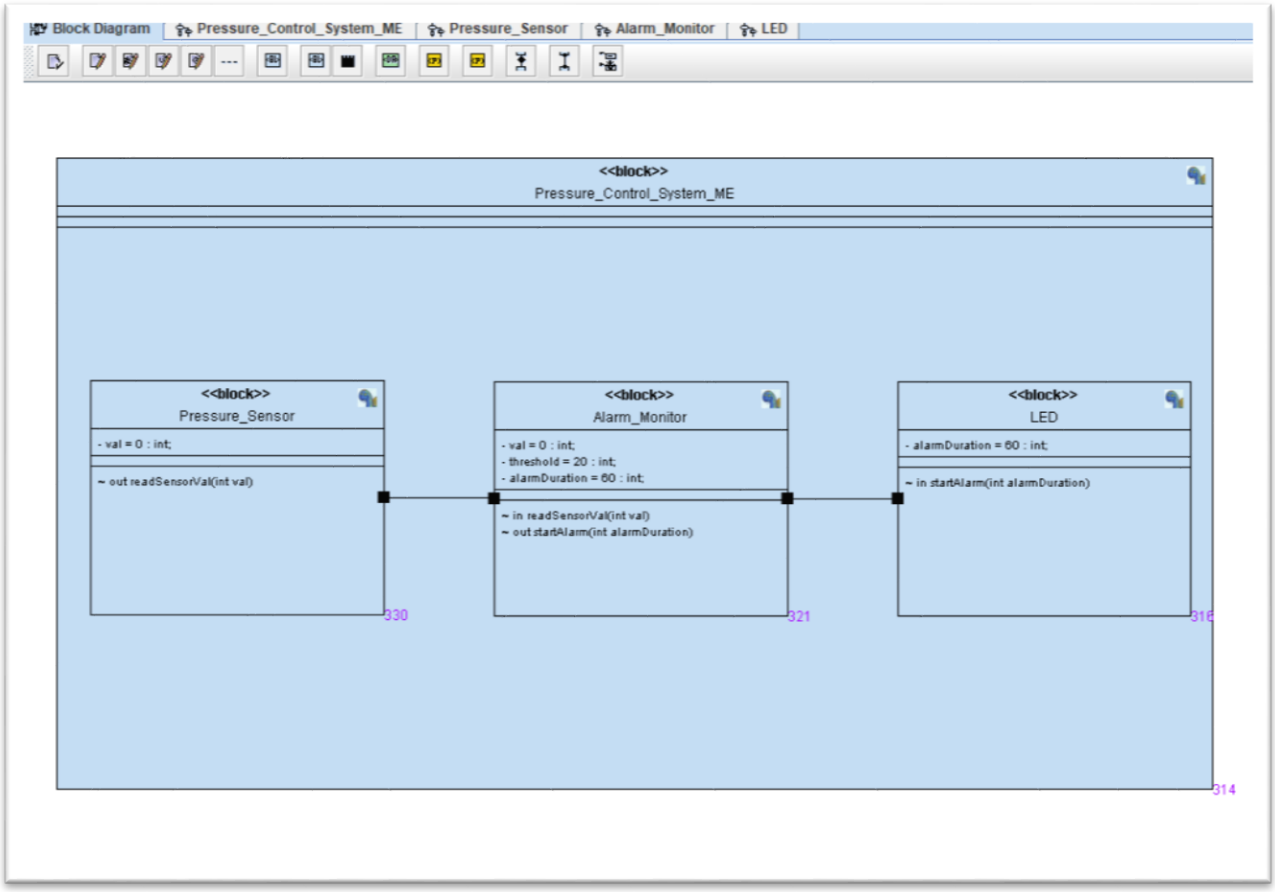


Activity Diagram

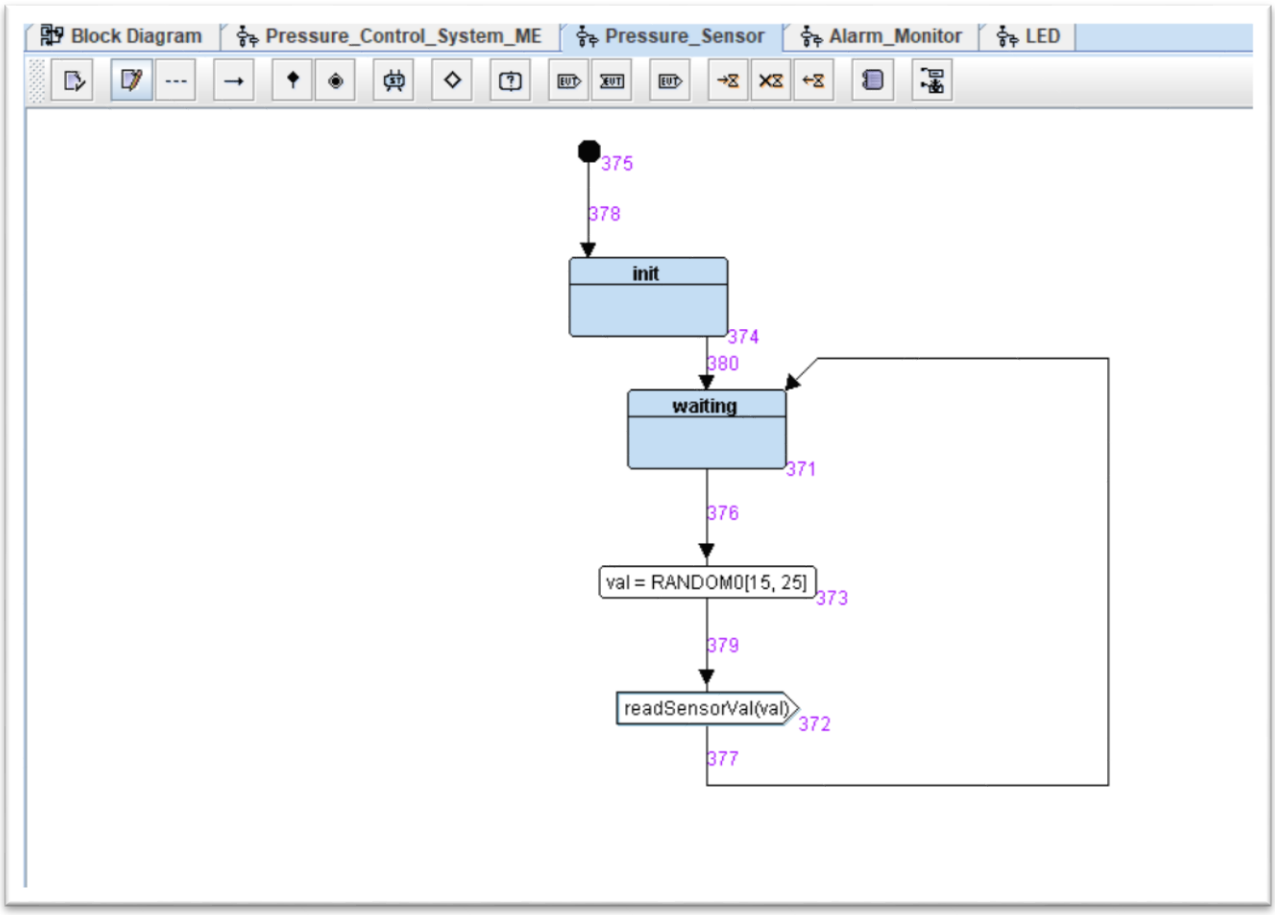


System Design

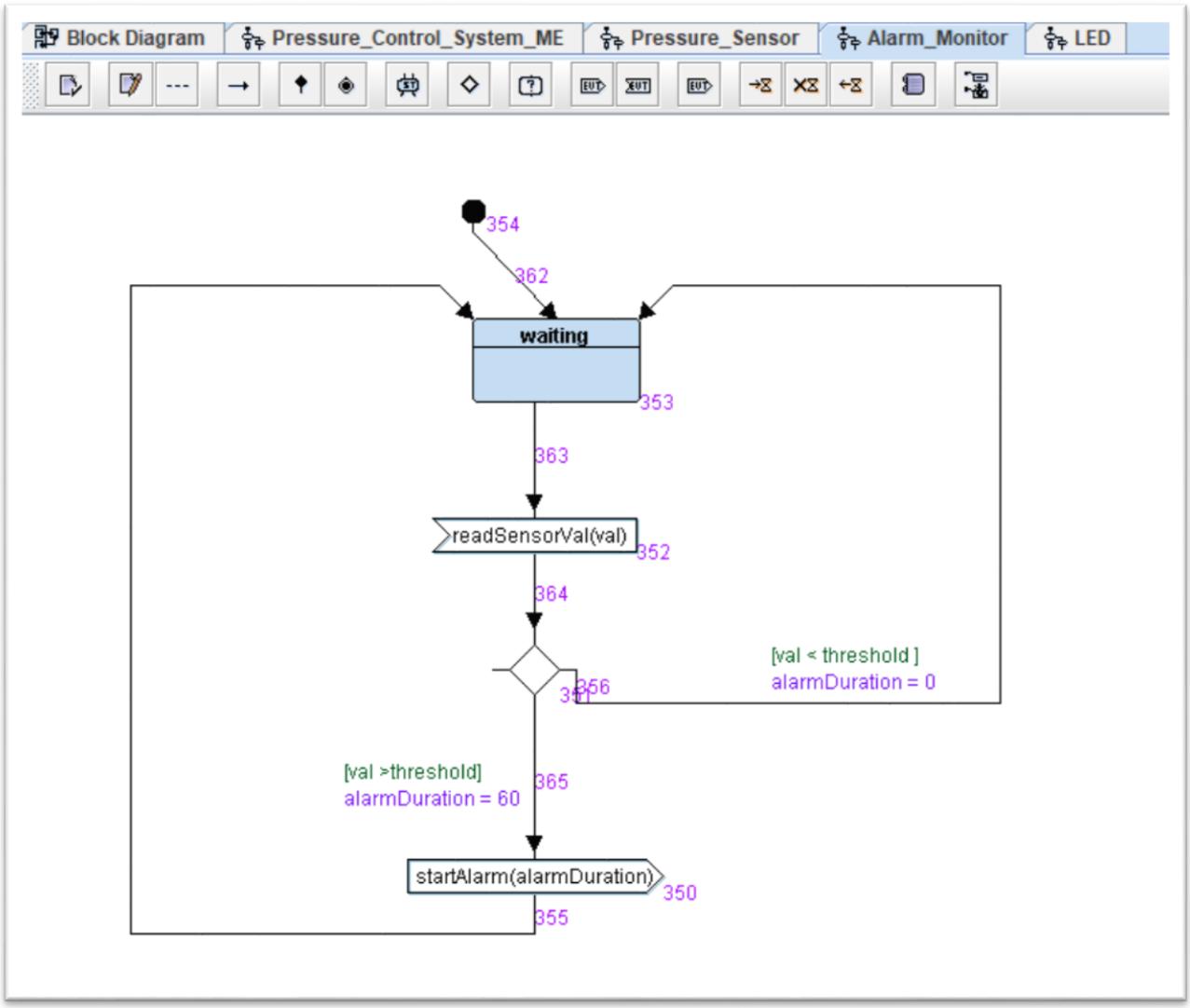
System Modules



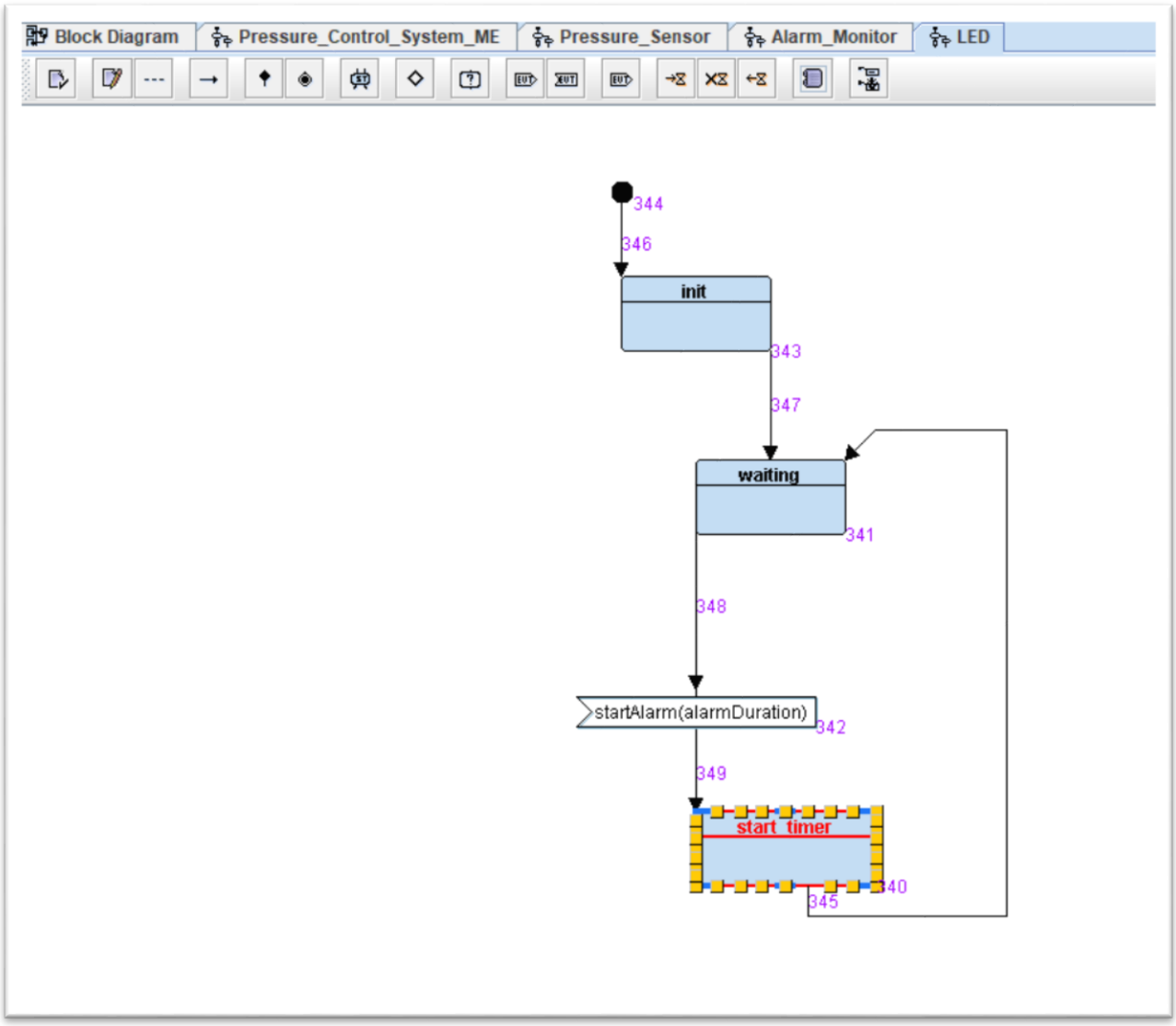
Pressure Sensor Diagram



Alarm Monitor Diagram

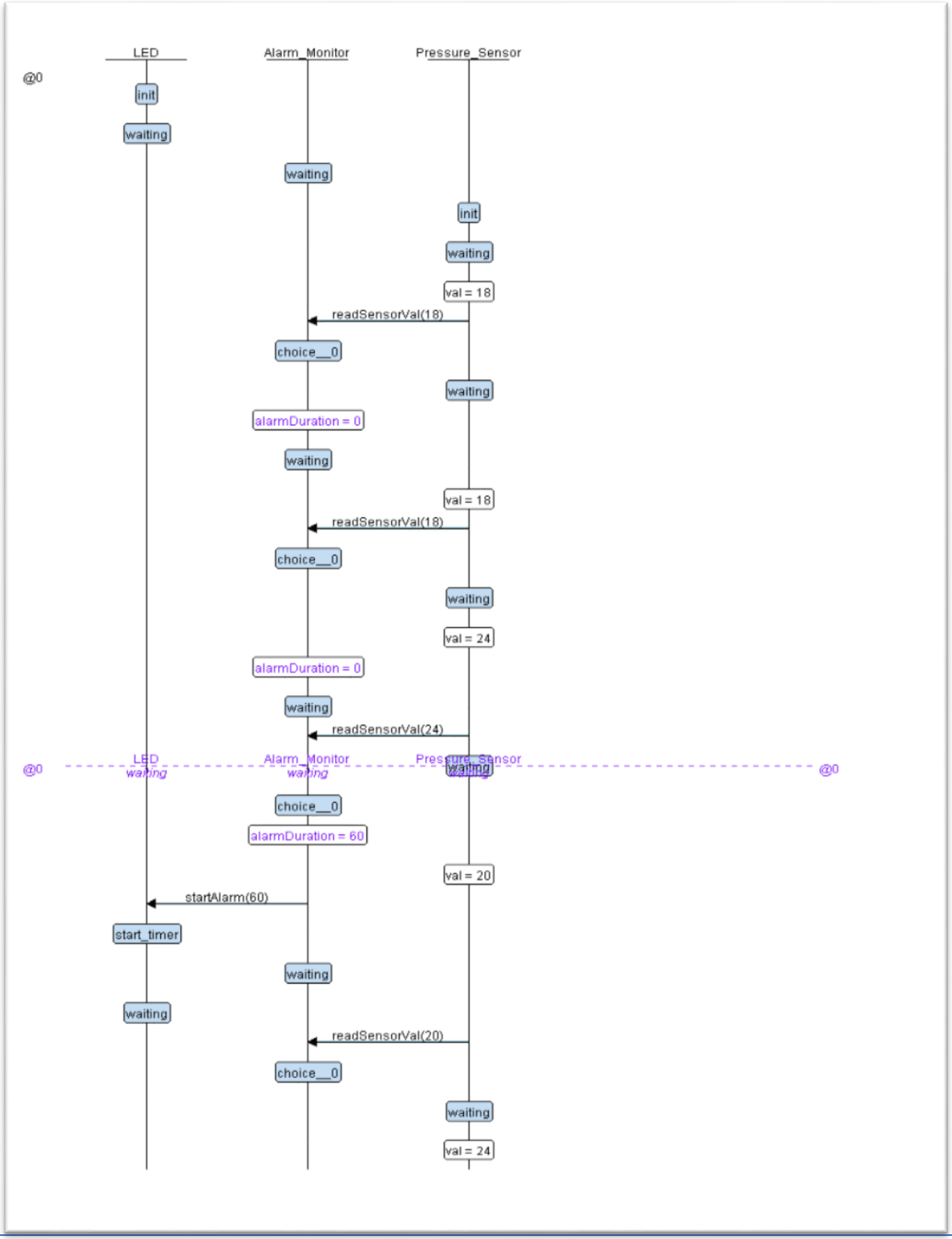


Led Diagram

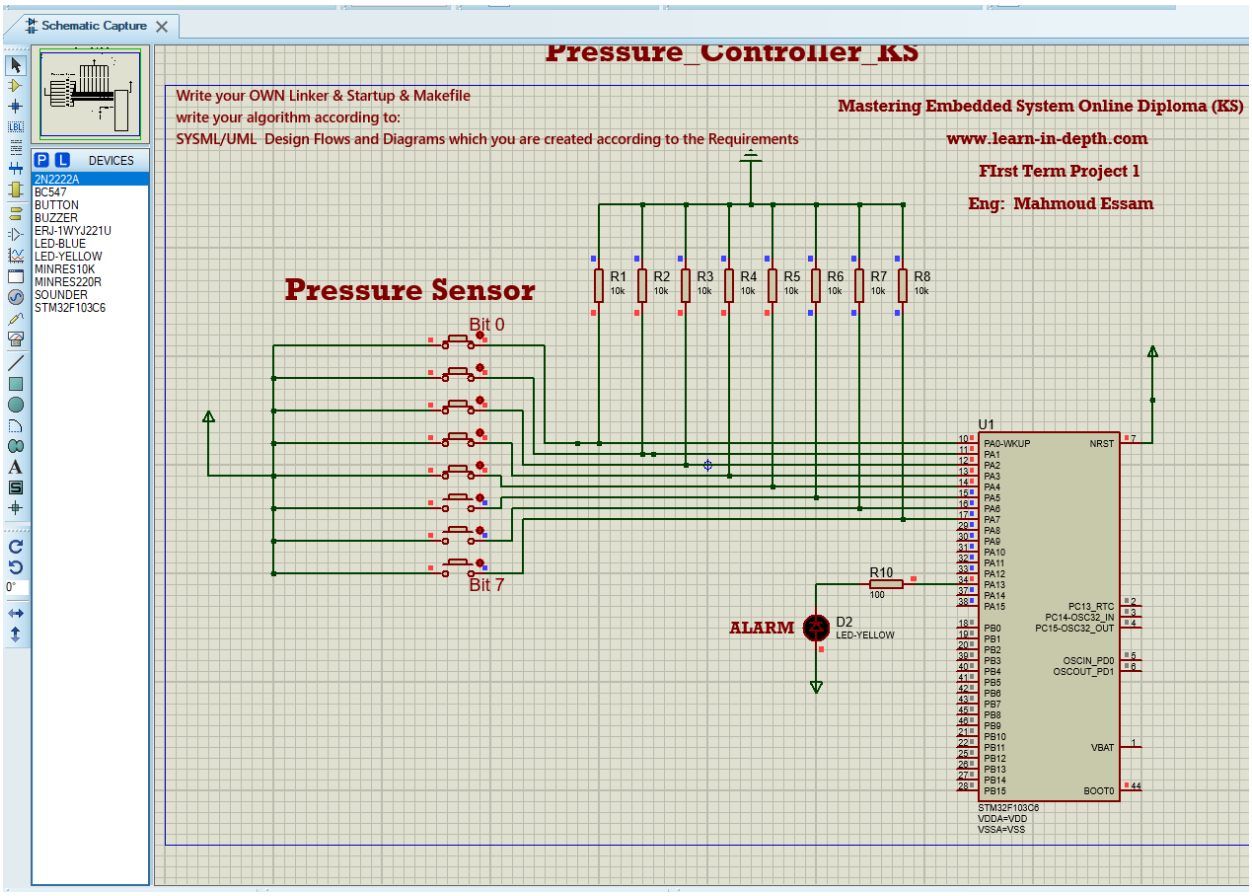

Block Diagram
Pressure_Control_System_ME
Pressure_Sensor
Alarm_Monitor
LED
344
346
init
343
347
waiting
341
348
startAlarm(alarmDuration)
342
349
start_timer
340
345

Simulation

Diagrams Simulation



Protues Simulation



Code Screenshots

Main.c

linker_script.ld | main.c

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

#include "lib.h"

int main (){
    GPIO_INITIALIZATION();
    while (1)
    {
        AlarmMonitor();
    }

}

```

Lib.h

makefile | Pressure

```c
#ifndef LIB_H_
#define LIB_H_
#include <stdint.h>
#include <stdio.h>
#include "driver.h"
#include "PressureSensor.h"
#include "Alarm.h"
#include "AlarmMonitor.h"
// APIs

int getPressureVal();


#endif /* LIB_H_ */
```

Alarm.c

makefile | PressureSensor.c | Alarm.c

```c
#include "Alarm.h"

void Set_Alarm_actuator(int i)
{
    if (i == 1){
        SET_BIT(GPIOA_ODR,13);
    }
    else if (i == 0){
        RESET_BIT(GPIOA_ODR,13);
    }
}
```

Alarm.h

makefile | PressureSensor.c | Alarm.h

```c
#ifndef ALARM_H_
#define ALARM_H_
#include "lib.h"

// APIs

void Set_Alarm_actuator(int i);



#endif /* ALARM_H_ */
```

AlarmMonitor.c

```c
#include "lib.h"
#define threshold 20
int x = 0;
void AlarmMonitor()
{
    x = getPressureVal();
    // if pressure value > threshold start alarm
    if( x > threshold)
    {
        Set_Alarm_actuator(1);
        Delay(100000);
        Set_Alarm_actuator(0);
        Delay(100000);
    }

}
```

AlramMonitor.h

```c
#ifndef ALARMMONITOR_H_
#define ALARMMONITOR_H_
#include "lib.h"

// APIs

void AlarmMonitor();


#endif /* ALARMMONITOR_H_ */
```

Driver.c

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

void Delay(int nCount)
{
    for(; nCount != 0; nCount--);
}


void GPIO_INITIALIZATION (){
    SET_BIT(APB2ENR, 2);
    GPIOA_CRL &= 0xFF0FFFFF;
    GPIOA_CRL |= 0x00000000;
    GPIOA_CRH &= 0xFF0FFFFF;
    GPIOA_CRH |= 0x22222222;
}

```

Dirver.h

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

#define SET_BIT(ADDRESS,BIT)   ADDRESS |=  (1<<BIT)
#define RESET_BIT(ADDRESS,BIT) ADDRESS &= ~(1<<BIT)
#define TOGGLE_BIT(ADDRESS,BIT)  ADDRESS ^=  (1<<BIT)
#define READ_BIT(ADDRESS,BIT) ((ADDRESS) &   (1<<(BIT)))


#define GPIO_PORTA 0x40010800
#define BASE_RCC   0x40021000

#define APB2ENR   *(volatile uint32_t *)(BASE_RCC + 0x18)

#define GPIOA_CRL *(volatile uint32_t *)(GPIO_PORTA + 0x00)
#define GPIOA_CRH *(volatile uint32_t *)(GPIO_PORTA + 0X04)
#define GPIOA_IDR *(volatile uint32_t *)(GPIO_PORTA + 0x08)
#define GPIOA_ODR *(volatile uint32_t *)(GPIO_PORTA + 0x0C)


void Delay(int nCount);
void GPIO_INITIALIZATION ();
```

PressureSensor.c

```c

#include "PressureSensor.h"

// Function to read the value from the sensor
int getPressureVal()
{
    return (GPIOA_IDR & 0xFF);
}
```

PressureSensor.h

```c
#ifndef PRESSURE_SENSOR_H_
#define PRESSURE_SENSOR_H_
#include "lib.h"

// APIs

int getPressureVal();


#endif /* PRESSURE_SENSOR_H_ */
```

LinkerScript.ld

linker_script.ld | main.c

```
/*
        Linker_script.ld File
        Eng.Mahmoud Essam
*/

MEMORY
{
   flash(RX) : ORIGIN = 0X08000000, LENGTH = 128K
   sram(RWX) : ORIGIN = 0X20000000, LENGTH = 20K
}

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


   .data :
   {
       _S_DATA_ = .;
       *(.data)
       _E_DATA_ = .;
       . = ALIGN(4);
   } > sram AT> flash


   .bss :
   {
       _S_BSS_ = .;
       *(.bss)
       _E_BSS_ = .;
       . = ALIGN(4);
   } > sram

   . = . + 1000;
   _stack_top = .;

}
```

MakeFile

makefile | PressureSensor.c | PressureSensor.h | Map-file.map | startup.c

```makefile
# Makefile for ARM-CortexM3
# Eng.Mahmoud Essam
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=High_Pressure_Control_System
all: $(Project_name).bin
	@echo "=============== Build is Done ==============="

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

sym:
	$(CC)nm.exe $(Project_name).elf
dumb:
	$(CC)objdump.exe -h $(Project_name).elf

clean:
	rm *.elf *.bin

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

Startup.c

makefile | PressureSensor.c | PressureSensor.h | startup.c

```c
// Startup.c
// Eng.Mahmoud Essam

#include "stdint.h"
void Reset_Handler();
void Default_Handler()
{
    Reset_Handler();
}
extern int main(void);


void NMI_Handler(void) __attribute__((weak, alias("Default_Handler")));
void H_fault_Handler(void) __attribute__((weak, alias("Default_Handler")));
void MM_Fault_Handler(void) __attribute__((weak, alias("Default_Handler")));
void Bus_Fault(void) __attribute__((weak, alias("Default_Handler")));
void Usage_Fault_Handler(void) __attribute__((weak, alias("Default_Handler")));
extern unsigned int _stack_top;

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 unsigned int _E_TEXT_;
extern unsigned int _S_DATA_;
extern unsigned int _E_DATA_;
extern unsigned int _S_BSS_;
extern unsigned int _E_BSS_;

void Reset_Handler()
{
    // Copy .data section from flash to sram
    unsigned int DATA_size = (unsigned char*)&_E_DATA_ - (unsigned char*)&_S_DATA_;
    unsigned char* p_src = (unsigned char*)&_E_TEXT_; // end of flash
    unsigned char* p_dst = (unsigned char*)&_S_DATA_; // start of sram
    for(int i = 0; i < DATA_size ; i++)
    {
        *p_dst++ = *p_src++;
    }

    // Initialize .bss section = 0

    unsigned int BSS_size = (unsigned char*)&_E_BSS_ - (unsigned char*)&_S_BSS_;
    p_src = (unsigned char*)&_S_BSS_;
    for(int i = 0 ;i < BSS_size; i++)
    {
        *p_src++ = (unsigned char)0;
    }

    // Jumb main
    main();
}
```

Sympols

```
arm-none-eabi-nm.exe High_Pressure_Control_System.elf
20000004 B _E_BSS_
20000000 D _E_DATA_
080001bc T _E_TEXT_
20000000 B _S_BSS_
20000000 D _S_DATA_
200003ec B _stack_top
08000058 T AlarmMonitor
0800012c W Bus_Fault
0800012c T Default_Handler
08000094 T Delay
08000114 T getPressureVal
080000b4 T GPIO_INITIALIZATION
0800012c W H_fault_Handler
08000104 T main
0800012c W MM_Fault_Handler
0800012c W NMI_Handler
08000138 T Reset_Handler
0800001c T Set_Alarm_actuator
0800012c W Usage_Fault_Handler
08000000 T vectors
20000000 B x
```

Code Sections

```
arm-none-eabi-objdump.exe -h High_Pressure_Control_System.elf

High_Pressure_Control_System.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000001bc  08000000  08000000  00010000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000000  20000000  080001bc  00020000  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000004  20000000  080001bc  00020000  2**2
                  ALLOC
  3 .debug_info   000031ce  00000000  00000000  00020000  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev 0000086f  00000000  00000000  000231ce  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000214  00000000  00000000  00023a3d  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 000000c0  00000000  00000000  00023c51  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   00000dce  00000000  00000000  00023d11  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_str    000005e3  00000000  00000000  00024adf  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007b  00000000  00000000  000250c2  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  0002513d  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000168  00000000  00000000  00025170  2**2
                  CONTENTS, READONLY, DEBUGGING
```