Mastering Embedded System Online Diploma
https://www.learn-in-depth.com/
https://www.facebook.com/groups/embedded.system.KS/

Pressure Control System Report

First term (Final Project 1)

Made by: Eng. Farha Emad Mohamed

My Profile:
https://www.learn-in-depth.com/online-diploma/farhaemad59%40gmail.com

- Case study:

A ”client” expects you to deliver the software of the following system:

- 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.
- keeps track of the measured values(optionally).

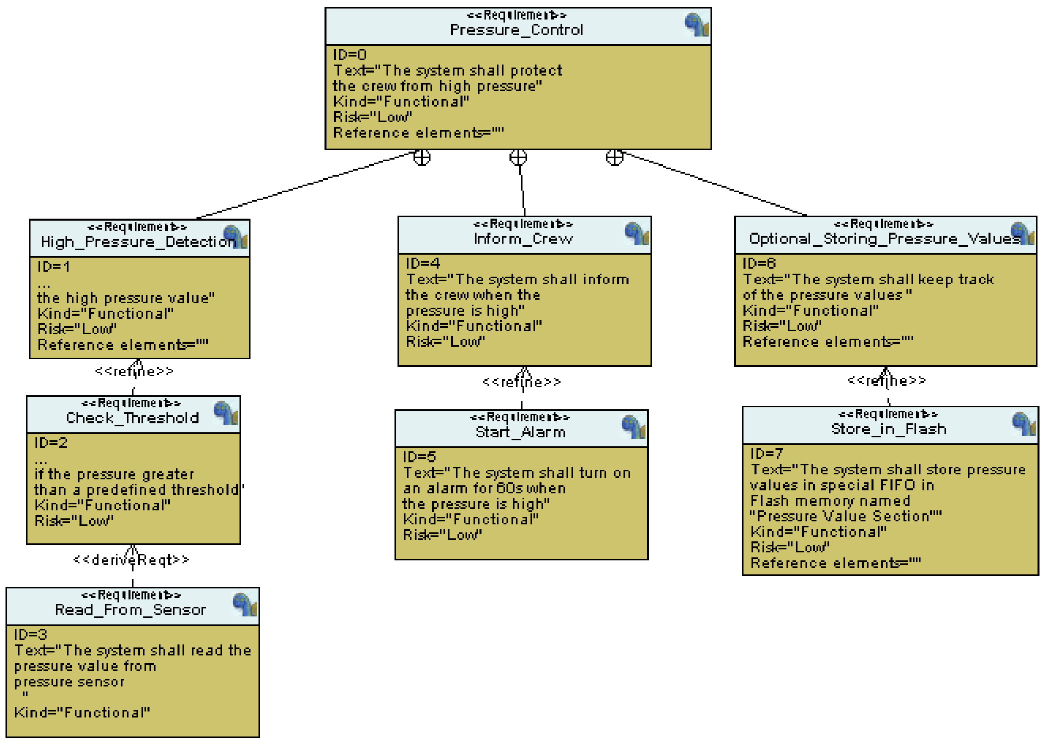
Pressure Controller Assumptions:

- The controller setup and shutdown procedures are not modeled
- The controller maintenance is not modeled
- The pressure sensor never fails
- The alarm never fails
- The controller never faces a power cut
- The ”keep track of measured value” option is not modeled in the first version of the design

- Method:

Choosing the V-model because the system is divided into several modules that need to be unit-tested separately first and then system-tested integratedly.

- System Requirement Diagram:



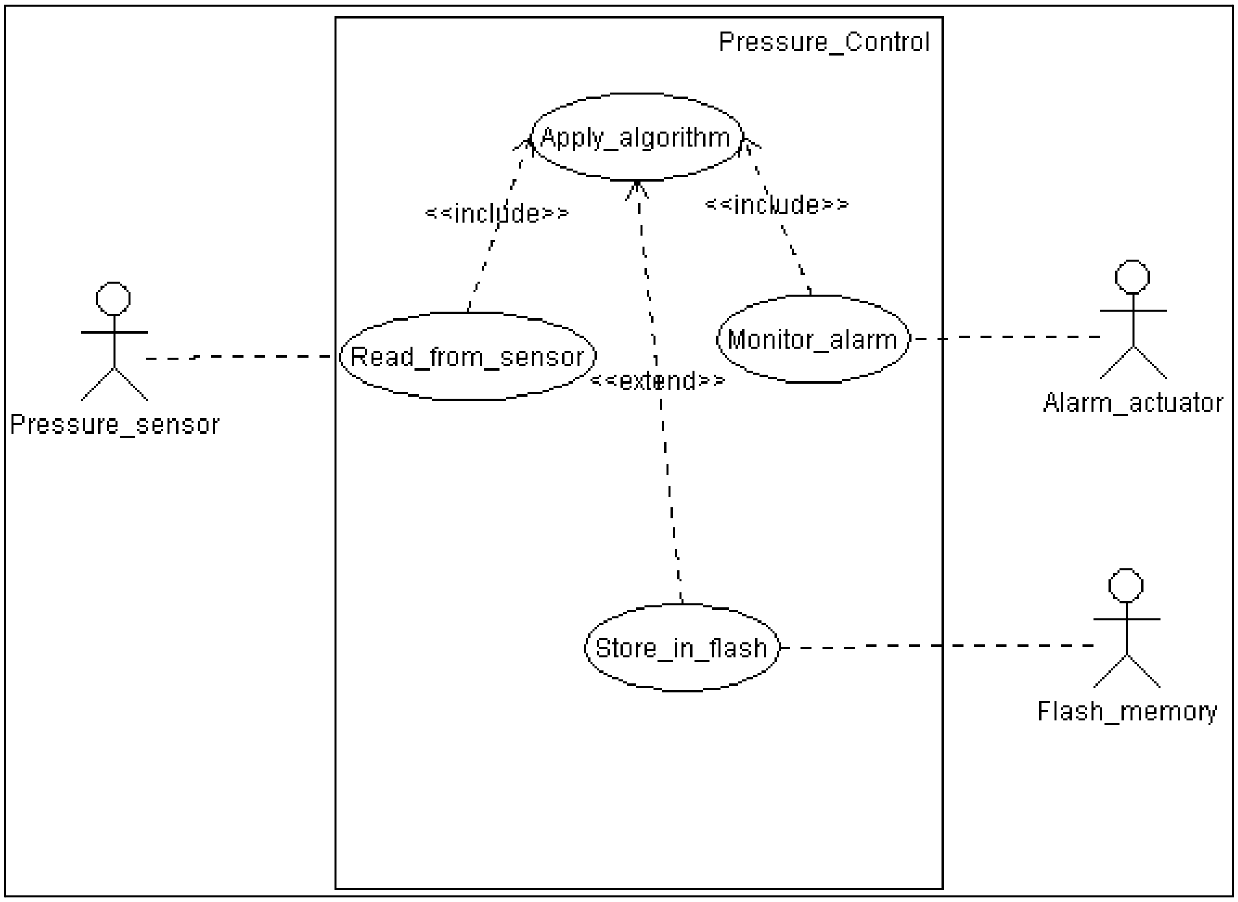
<<Requirement>> Pressure_Control
ID=0
Text="The system shall protect the crew from high pressure"
Kind="Functional"
Risk="Low"
Reference elements=""
<<Requirement>> High_Pressure_Detection
ID=1
...
the high pressure value"
Kind="Functional"
Risk="Low"
Reference elements=""
<<refine>>
<<Requirement>> Check_Threshold
ID=2
...
if the pressure greater than a predefined threshold"
Kind="Functional"
Risk="Low"
<<deriveReqt>>
<<Requirement>> Read_From_Sensor
ID=3
Text="The system shall read the pressure value from pressure sensor "
Kind="Functional"
<<Requirement>> Inform_Crew
ID=4
Text="The system shall inform the crew when the pressure is high"
Kind="Functional"
Risk="Low"
<<refine>>
<<Requirement>> Start_Alarm
ID=5
Text="The system shall turn on an alarm for 60s when the pressure is high"
Kind="Functional"
Risk="Low"
<<Requirement>> Optional_Storing_Pressure_Values
ID=6
Text="The system shall keep track of the pressure values "
Kind="Functional"
Risk="Low"
Reference elements=""
<<refine>>
<<Requirement>> Store_in_Flash
ID=7
Text="The system shall store pressure values in special FIFO in Flash memory named "Pressure Value Section""
Kind="Functional"
Risk="Low"
Reference elements=""

- Space Exploration:

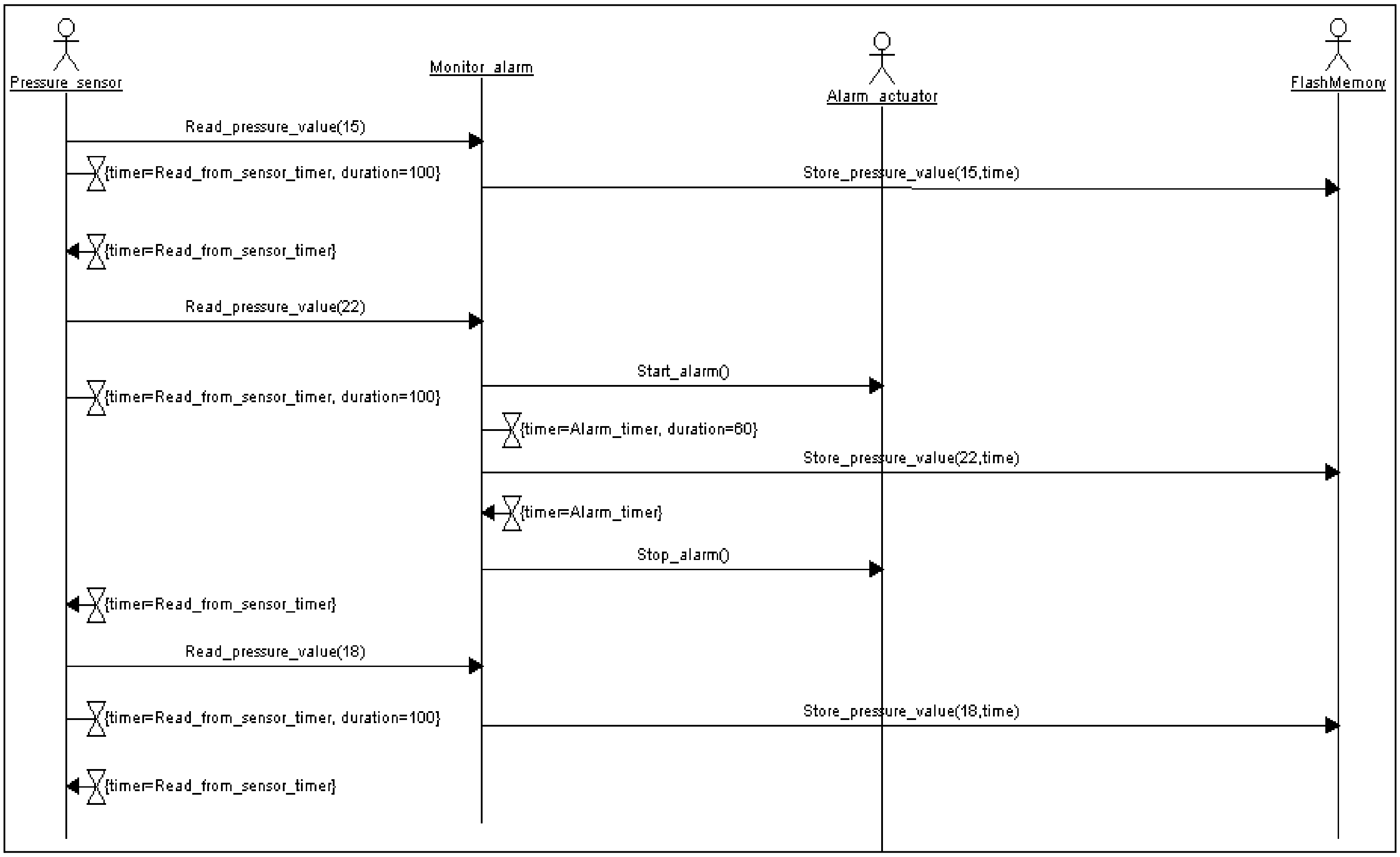
Choosing STM32F103C6 microcontroller with cortex-m3 CPU.

- System Analysis:

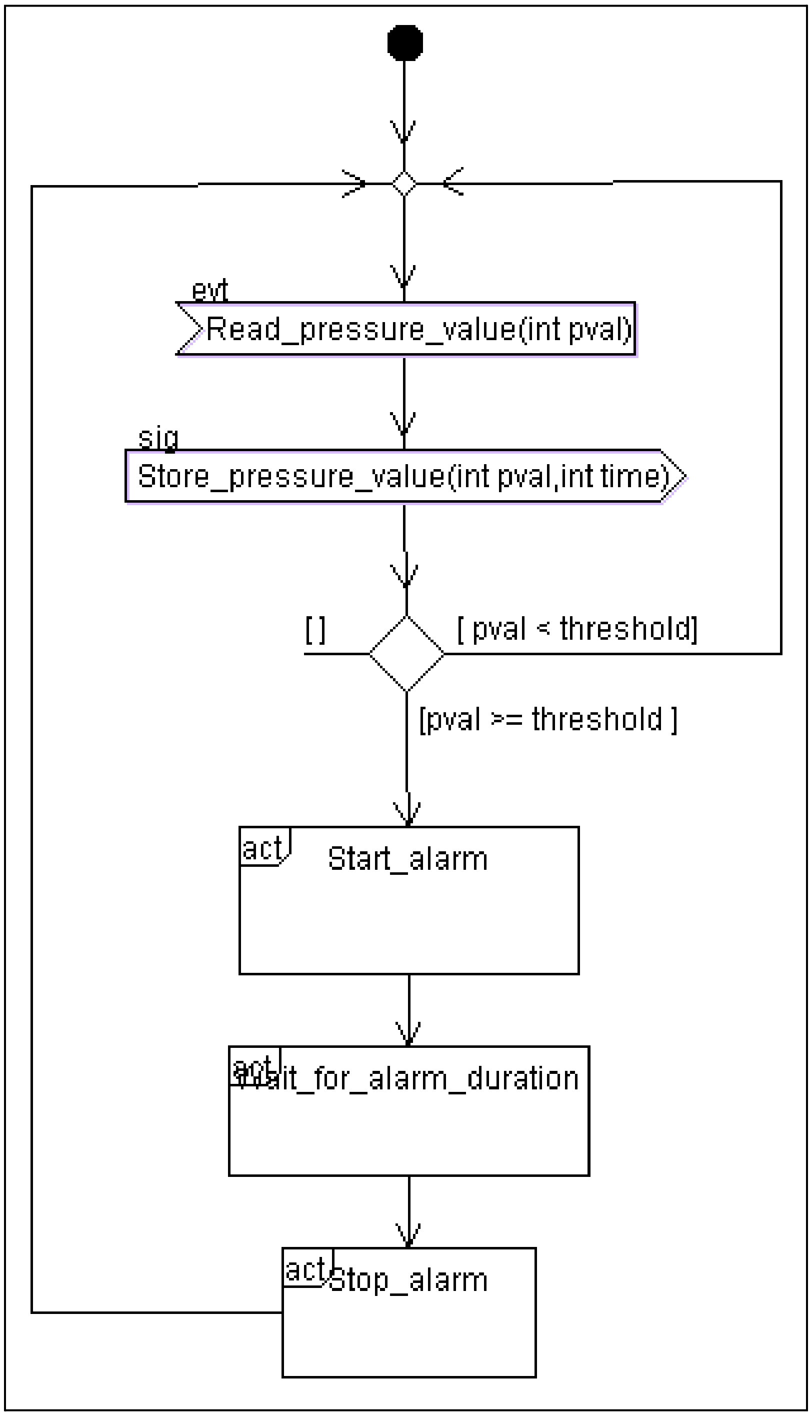
- Use case diagram:



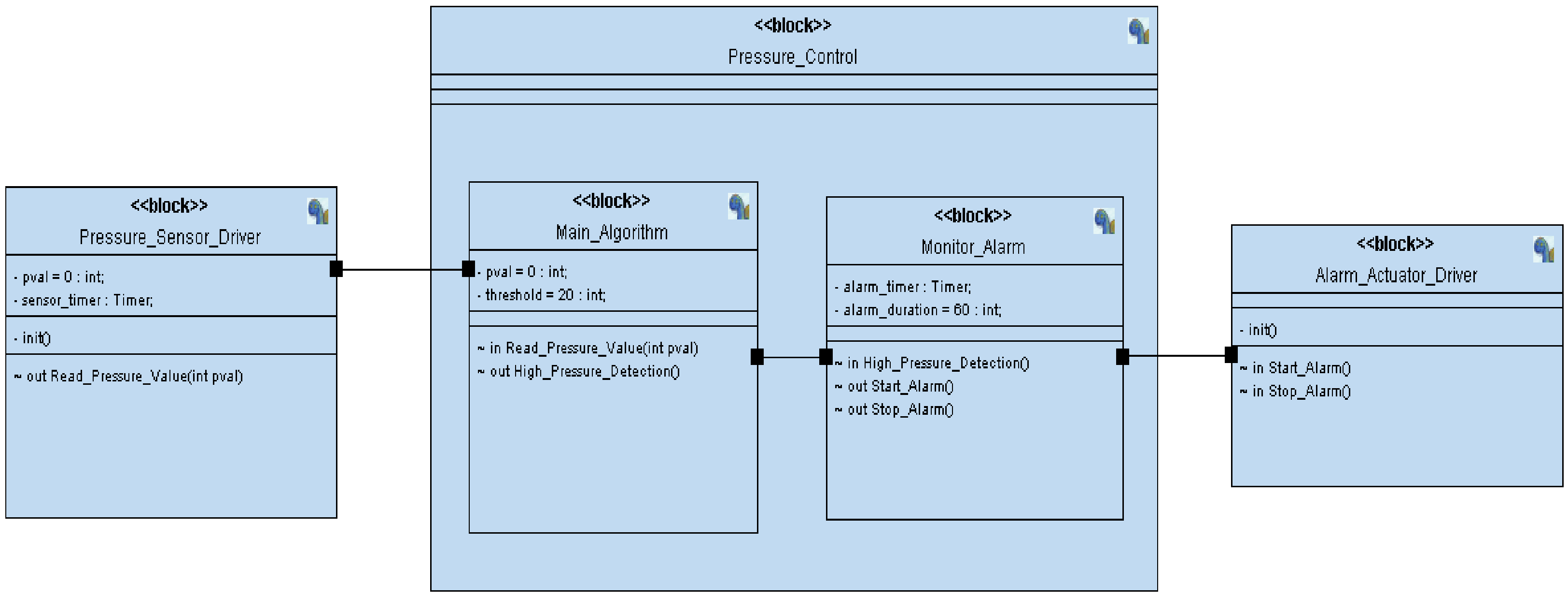
- Sequence diagram:


Pressure_sensor
Monitor_alarm
Alarm_actuator
FlashMemory
Read_pressure_value(15)
{timer=Read_from_sensor_timer, duration=100}
Store_pressure_value(15,time)
{timer=Read_from_sensor_timer}
Read_pressure_value(22)
Start_alarm()
{timer=Read_from_sensor_timer, duration=100}
{timer=Alarm_timer, duration=60}
Store_pressure_value(22,time)
{timer=Alarm_timer}
Stop_alarm()
{timer=Read_from_sensor_timer}
Read_pressure_value(18)
{timer=Read_from_sensor_timer, duration=100}
Store_pressure_value(18,time)
{timer=Read_from_sensor_timer}

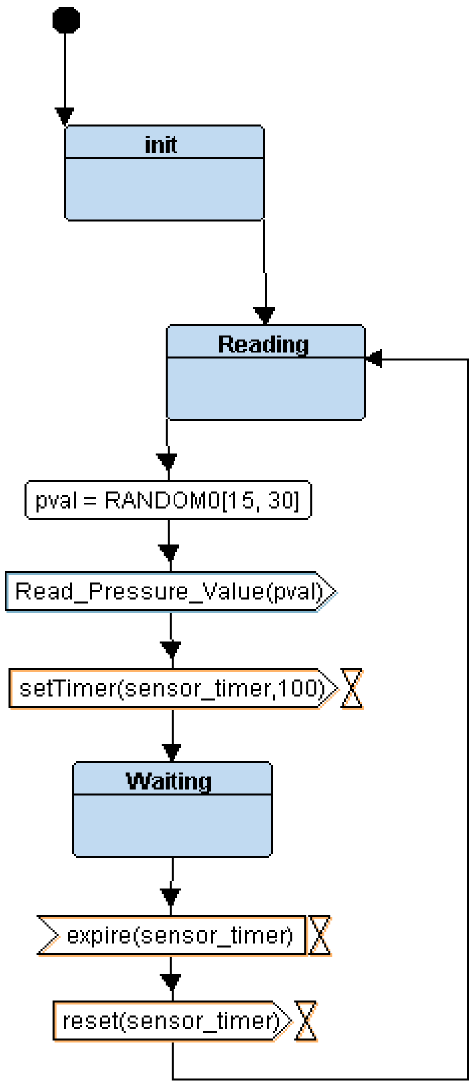
- Activity diagram:



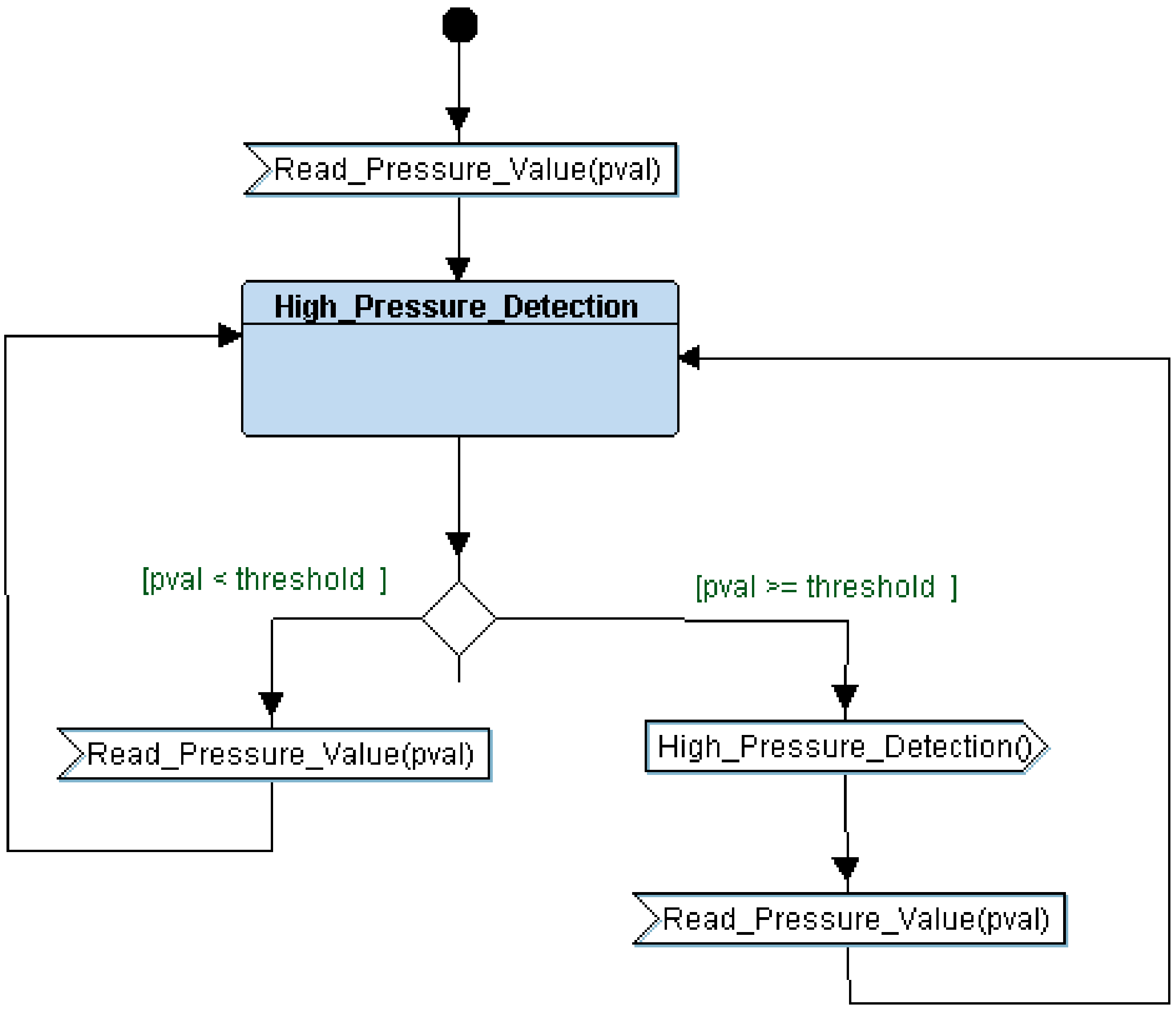
- System design(State machine):



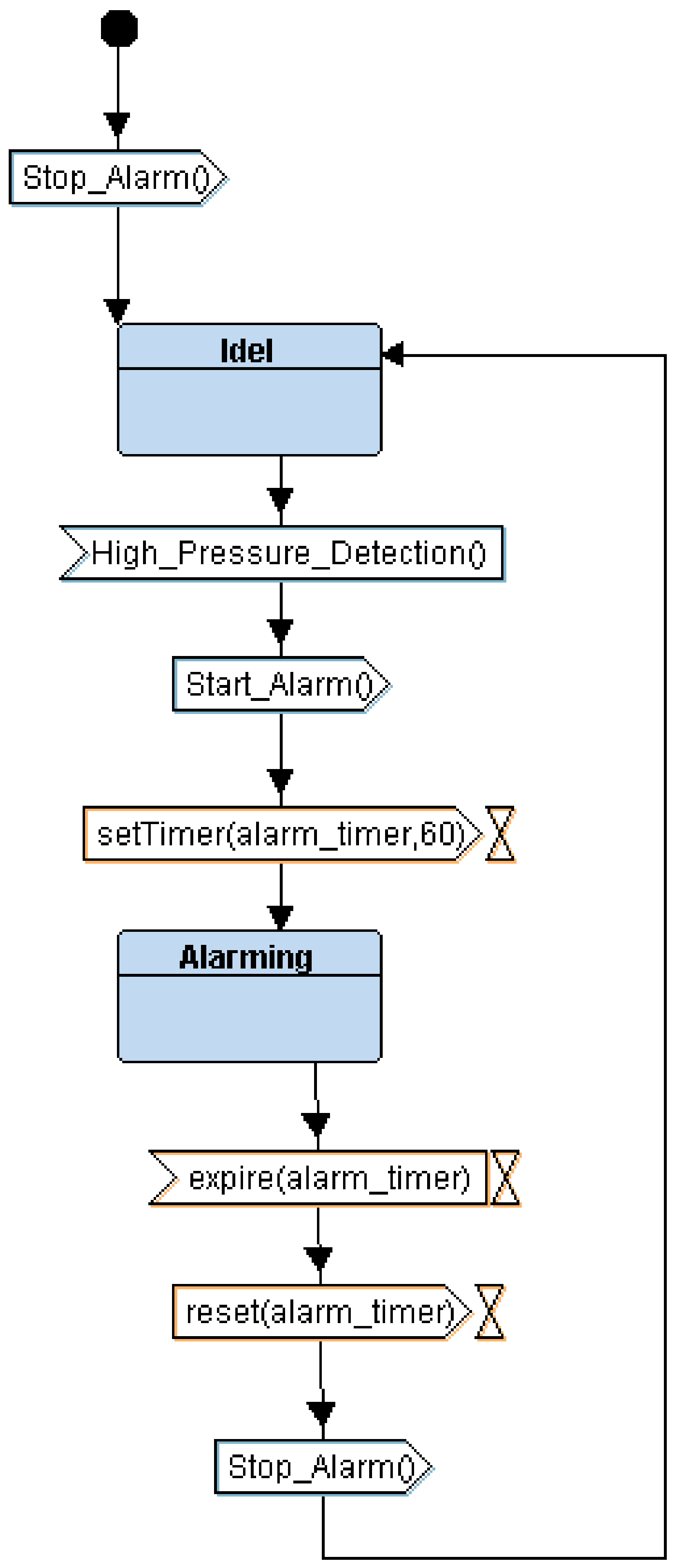
1) Pressure sensor state diagram:



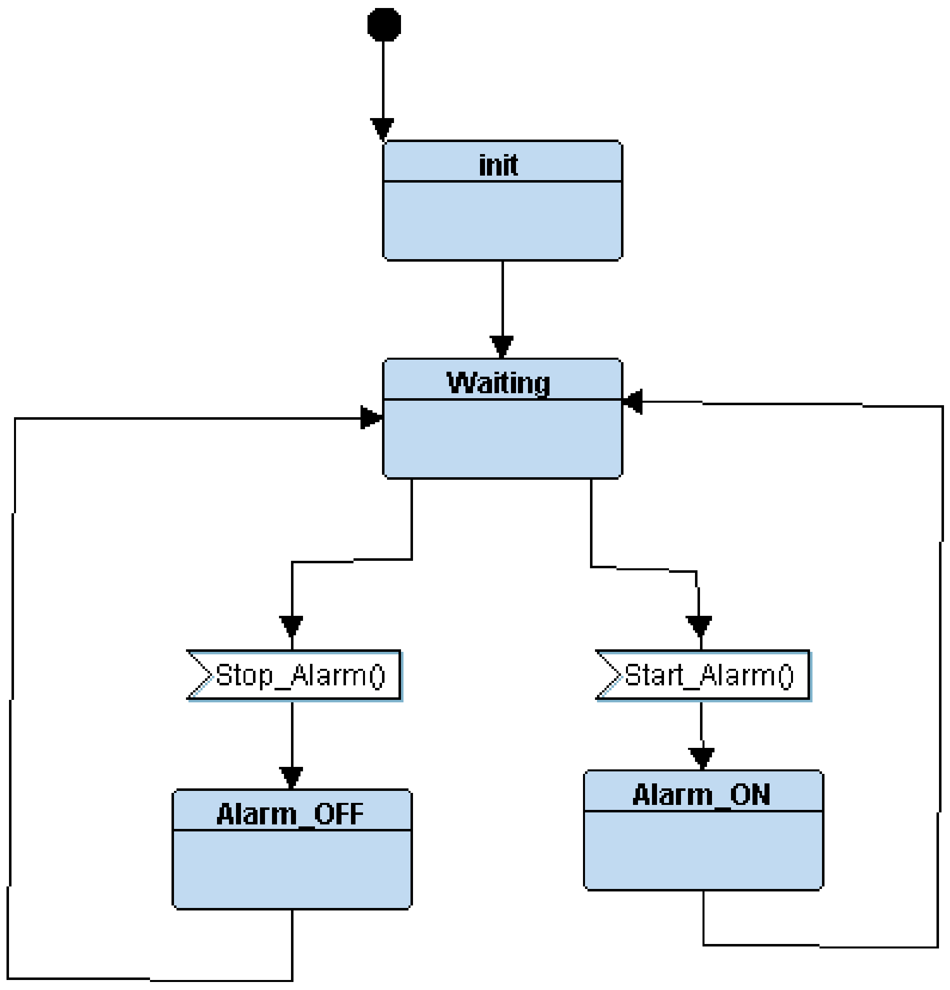
2) Main Algorithm state diagram:


Read_Pressure_Value(pval)
High_Pressure_Detection
[pval < threshold]
[pval >= threshold]
Read_Pressure_Value(pval)
High_Pressure_Detection()
Read_Pressure_Value(pval)

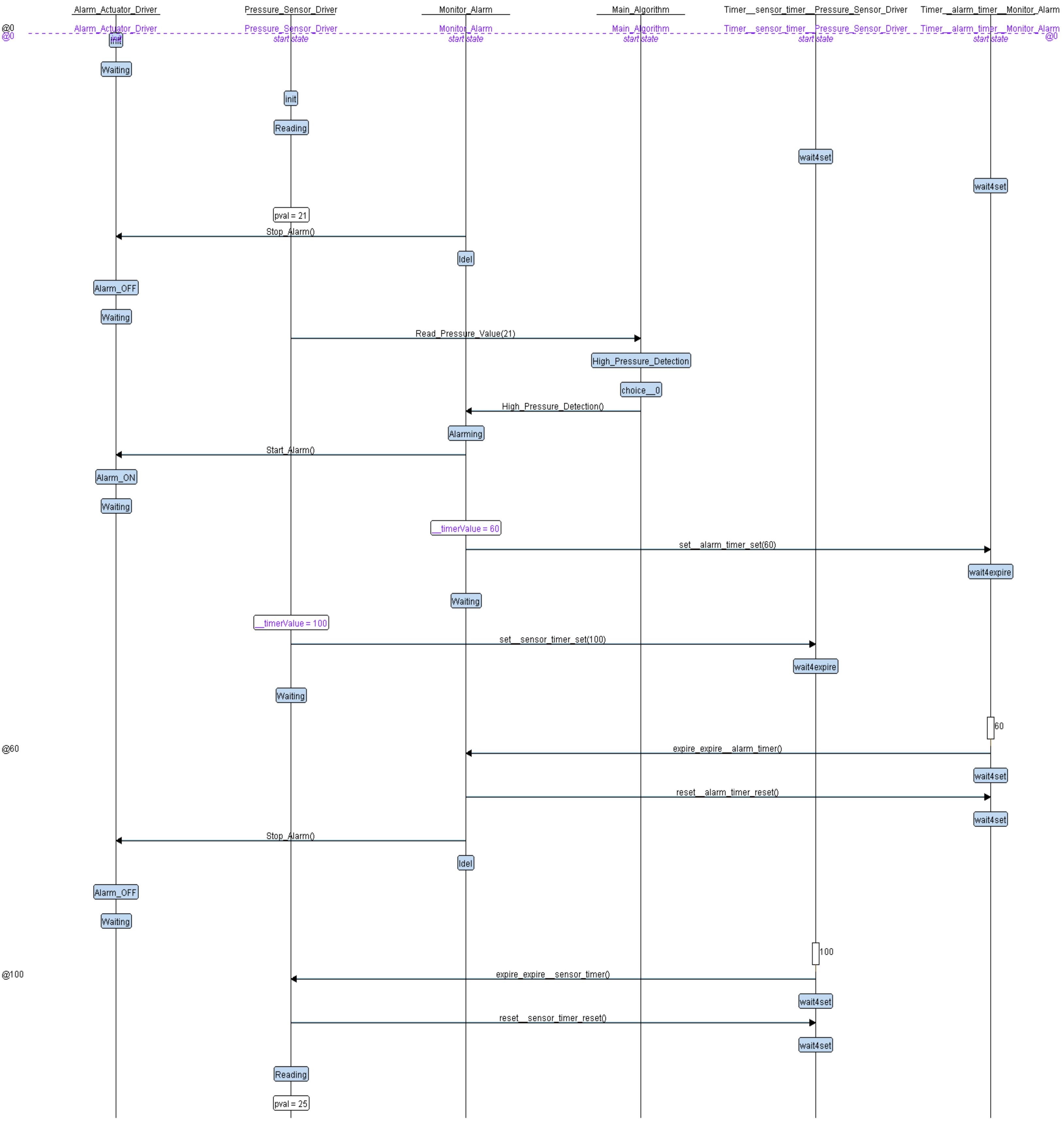
3) Alarm Monitor state diagram:



4) Alarm actuator state diagram:

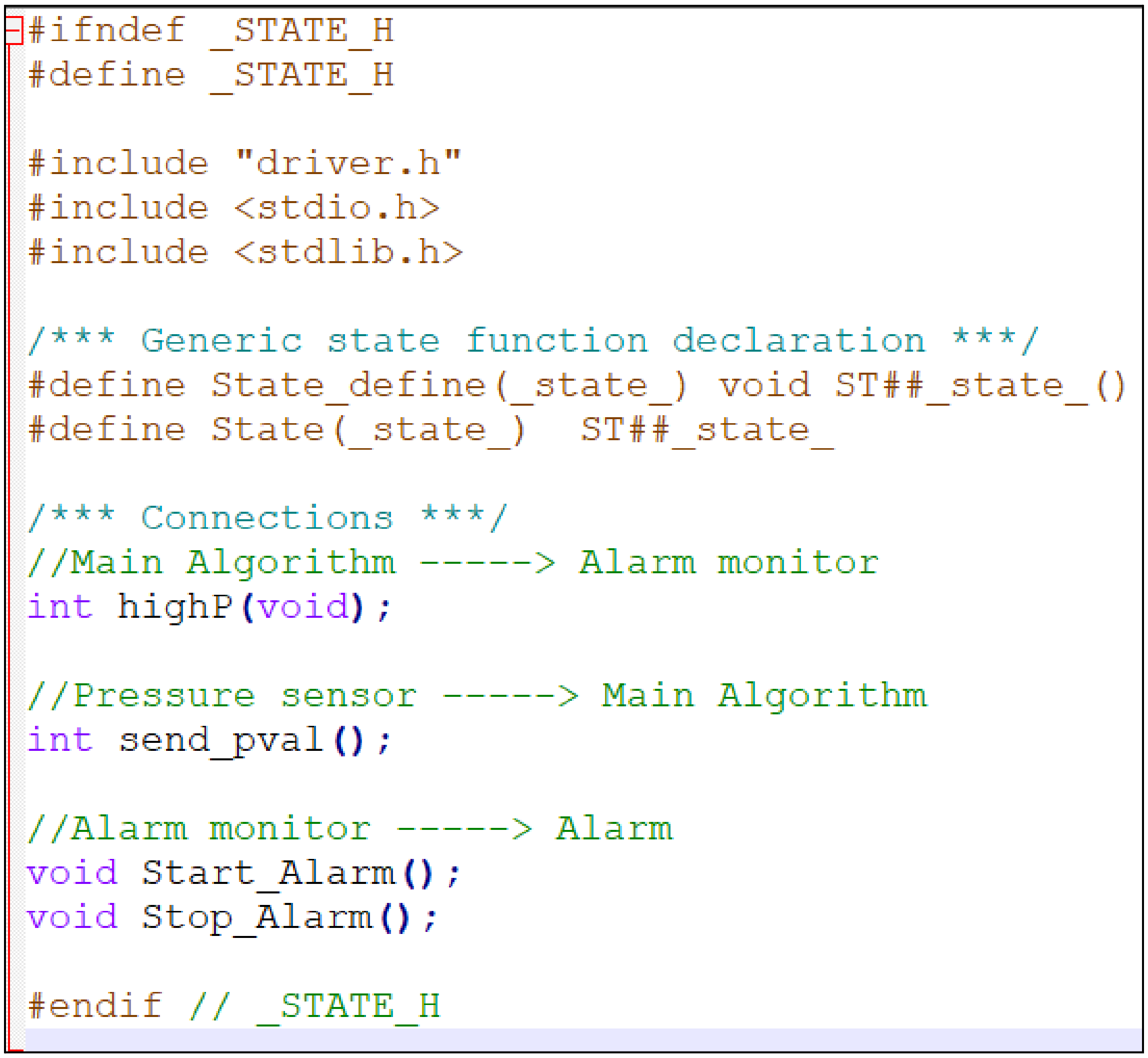


5) Simulation:



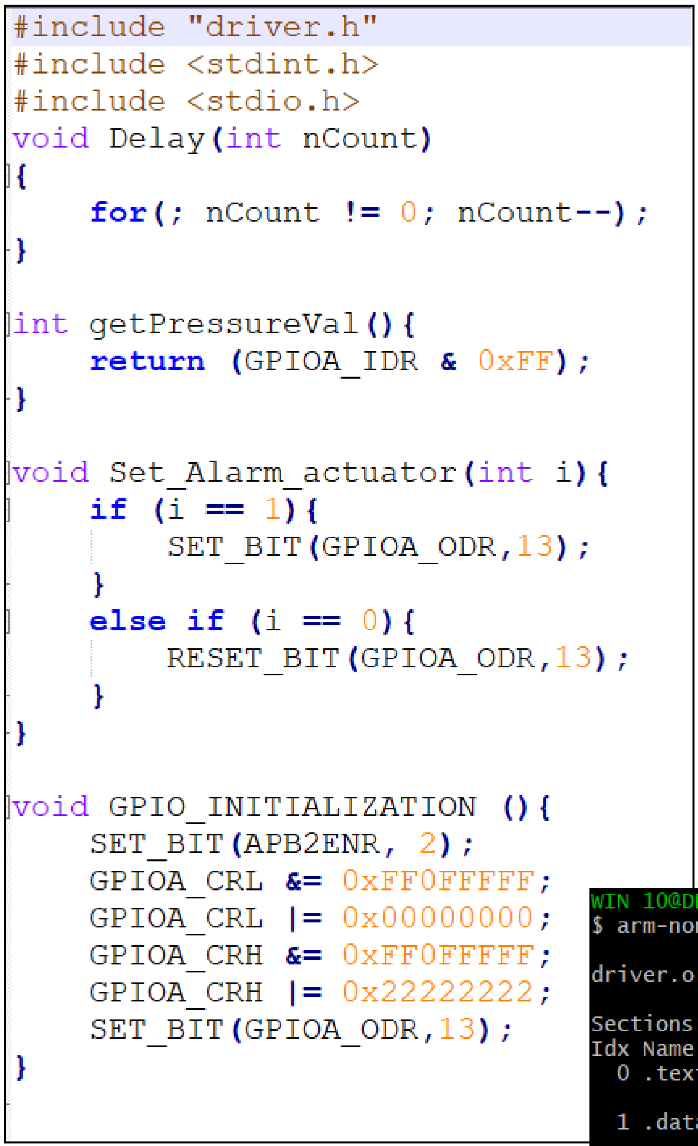
- Implementation:

- State. h

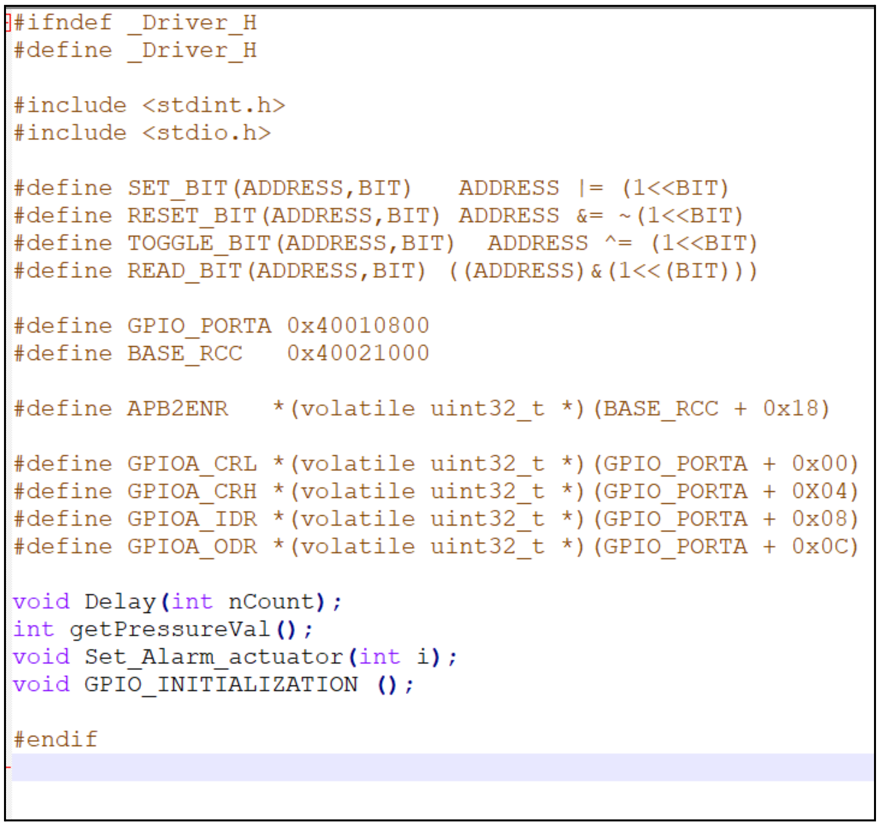


```c
#ifndef _STATE_H
#define _STATE_H

#include "driver.h"
#include <stdio.h>
#include <stdlib.h>

/*** Generic state function declaration ***/
#define State_define(_state_) void ST##_state_()
#define State(_state_)  ST##_state_

/*** Connections ***/
//Main Algorithm -----> Alarm monitor
int highP(void);

//Pressure sensor -----> Main Algorithm
int send_pval();

//Alarm monitor -----> Alarm
void Start_Alarm();
void Stop_Alarm();

#endif // _STATE_H
```

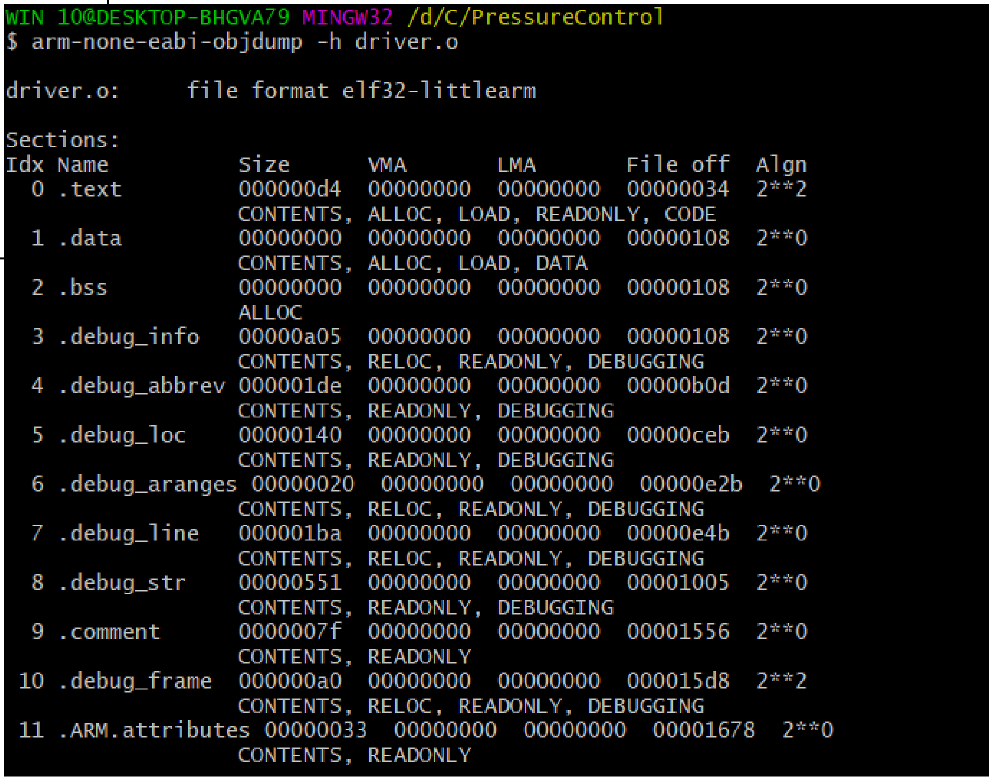
GPIO:



```c
#include "driver.h"
#include <stdint.h>
#include <stdio.h>
void Delay(int nCount)
{
    for(; nCount != 0; nCount--);
}

int getPressureVal(){
    return (GPIOA_IDR & 0xFF);
}

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

void GPIO_INITIALIZATION (){
    SET_BIT(APB2ENR, 2);
    GPIOA_CRL &= 0xFF0FFFFF;
    GPIOA_CRL |= 0x00000000;
    GPIOA_CRH &= 0xFF0FFFFF;
    GPIOA_CRH |= 0x22222222;
    SET_BIT(GPIOA_ODR,13);
}
```

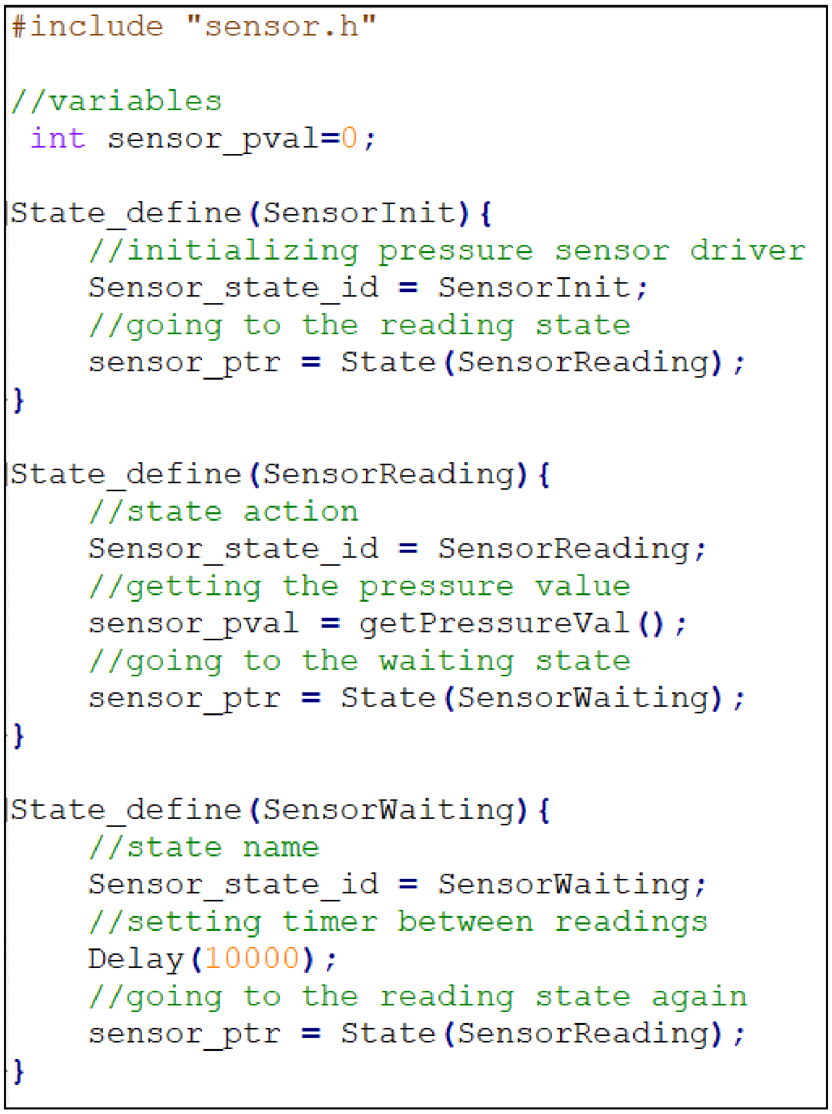


```c
#ifndef _Driver_H
#define _Driver_H

#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);
int getPressureVal();
void Set_Alarm_actuator(int i);
void GPIO_INITIALIZATION ();

#endif
```

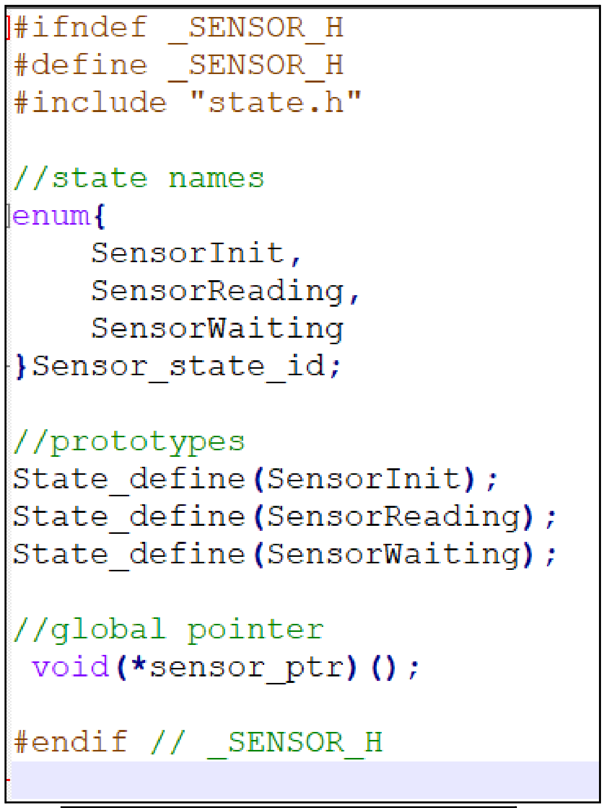


```
WIN 10@DESKTOP-BHGVA79 MINGW32 /d/C/PressureControl
$ arm-none-eabi-objdump -h driver.o

driver.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000000d4  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000108  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000108  2**0
                  ALLOC
  3 .debug_info   00000a05  00000000  00000000  00000108  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000001de  00000000  00000000  00000b0d  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000140  00000000  00000000  00000ceb  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000e2b  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   000001ba  00000000  00000000  00000e4b  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    00000551  00000000  00000000  00001005  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007f  00000000  00000000  00001556  2**0
                  CONTENTS, READONLY
 10 .debug_frame  000000a0  00000000  00000000  000015d8  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 11 .ARM.attributes 00000033  00000000  00000000  00001678  2**0
                  CONTENTS, READONLY
```

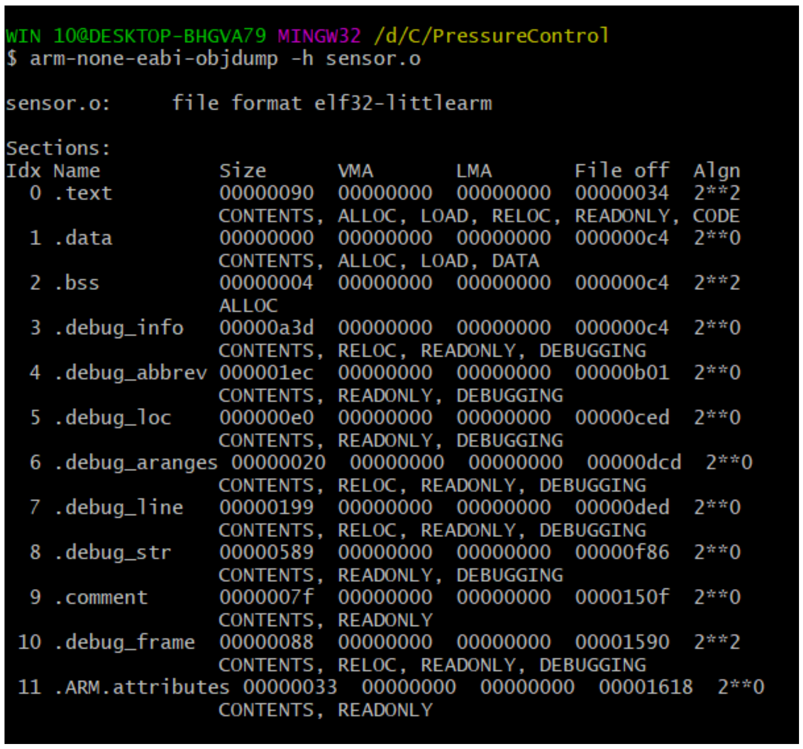
- Pressure Sensor :



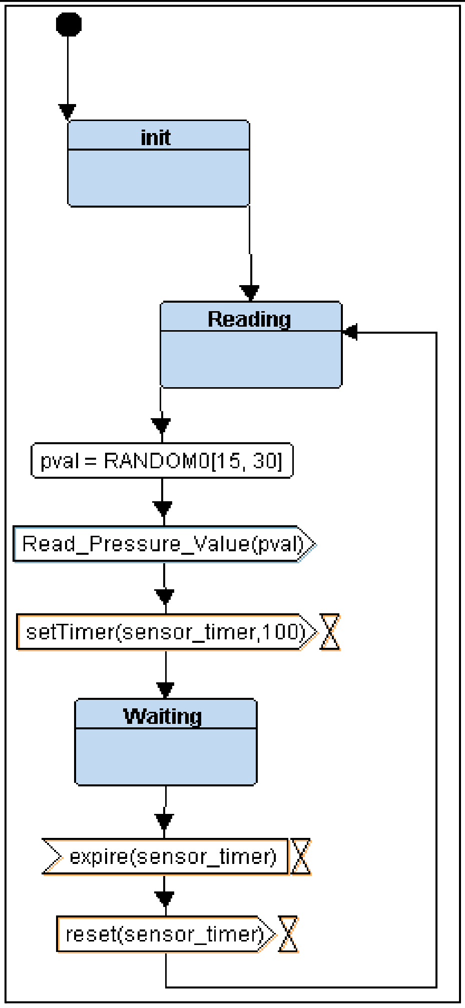
```c
#include "sensor.h"

//variables
 int sensor_pval=0;

State_define(SensorInit){
    //initializing pressure sensor driver
    Sensor_state_id = SensorInit;
    //going to the reading state
    sensor_ptr = State(SensorReading);
}

State_define(SensorReading){
    //state action
    Sensor_state_id = SensorReading;
    //getting the pressure value
    sensor_pval = getPressureVal();
    //going to the waiting state
    sensor_ptr = State(SensorWaiting);
}

State_define(SensorWaiting){
    //state name
    Sensor_state_id = SensorWaiting;
    //setting timer between readings
    Delay(10000);
    //going to the reading state again
    sensor_ptr = State(SensorReading);
}
```



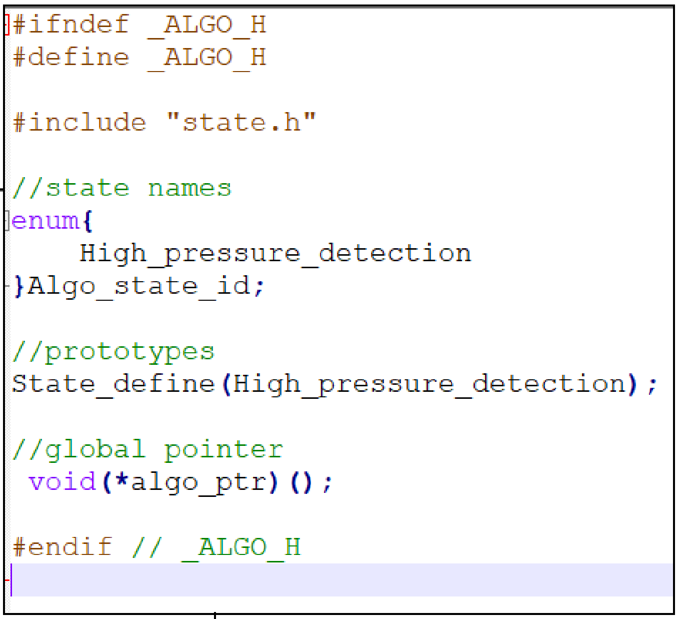
```c
#ifndef _SENSOR_H
#define _SENSOR_H
#include "state.h"

//state names
enum{
    SensorInit,
    SensorReading,
    SensorWaiting
}Sensor_state_id;

//prototypes
State_define(SensorInit);
State_define(SensorReading);
State_define(SensorWaiting);

//global pointer
 void(*sensor_ptr)();

#endif // _SENSOR_H
```



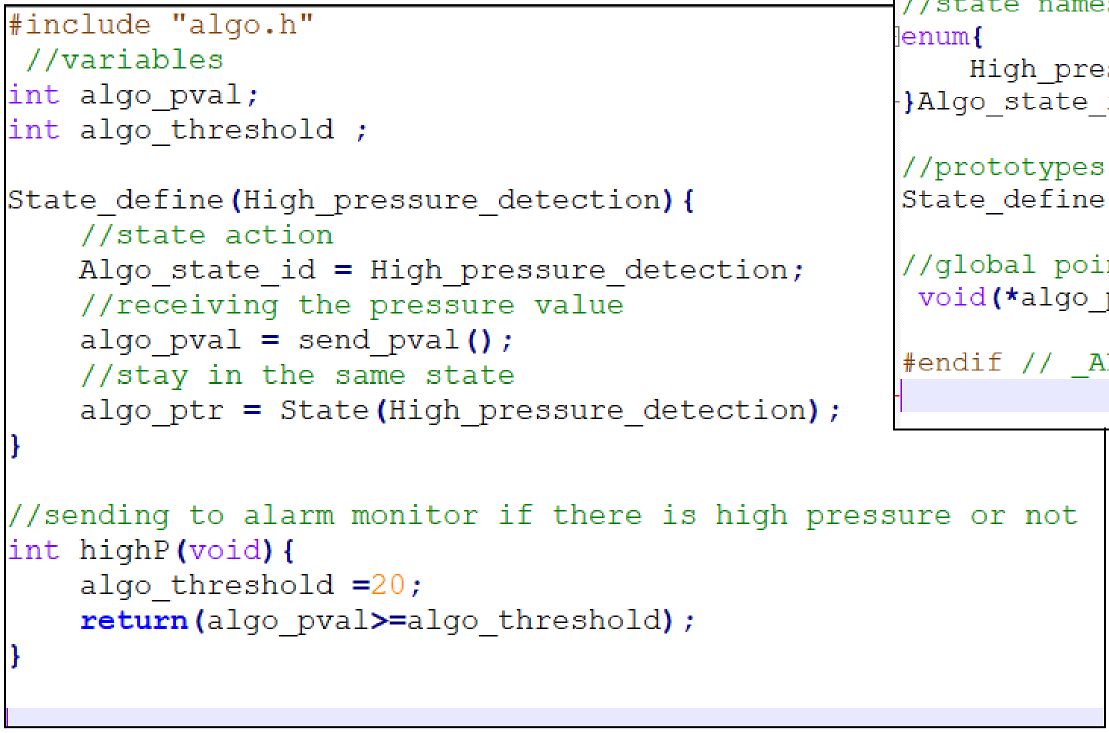
```
WIN 10@DESKTOP-BHGVA79 MINGW32 /d/C/PressureControl
$ arm-none-eabi-objdump -h sensor.o

sensor.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000090  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000c4  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000004  00000000  00000000  000000c4  2**2
                  ALLOC
  3 .debug_info   00000a3d  00000000  00000000  000000c4  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000001ec  00000000  00000000  00000b01  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    000000e0  00000000  00000000  00000ced  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000dcd  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   00000199  00000000  00000000  00000ded  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    00000589  00000000  00000000  00000f86  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007f  00000000  00000000  0000150f  2**0
                  CONTENTS, READONLY
 10 .debug_frame  00000088  00000000  00000000  00001590  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 11 .ARM.attributes 00000033  00000000  00000000  00001618  2**0
                  CONTENTS, READONLY
```



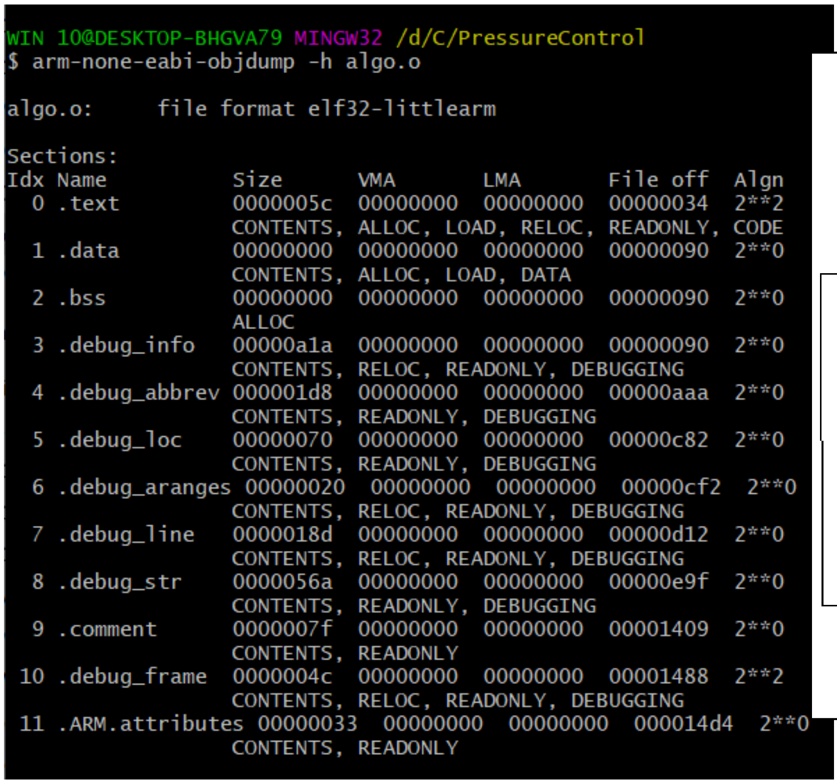
- Main Algorithm:



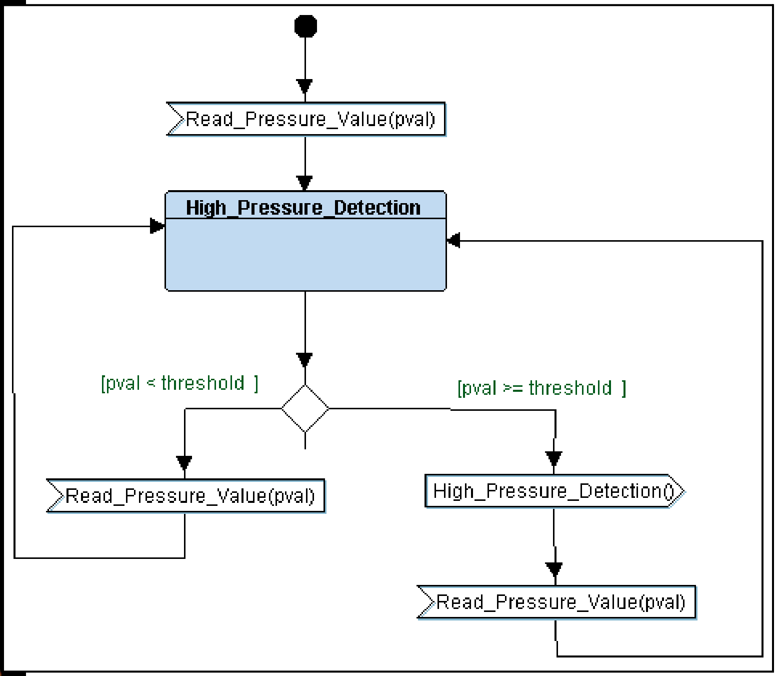
```
#ifndef _ALGO_H
#define _ALGO_H

#include "state.h"

//state names
enum{
    High_pressure_detection
}Algo_state_id;

//prototypes
State_define(High_pressure_detection);

//global pointer
 void(*algo_ptr)();

#endif // _ALGO_H
```



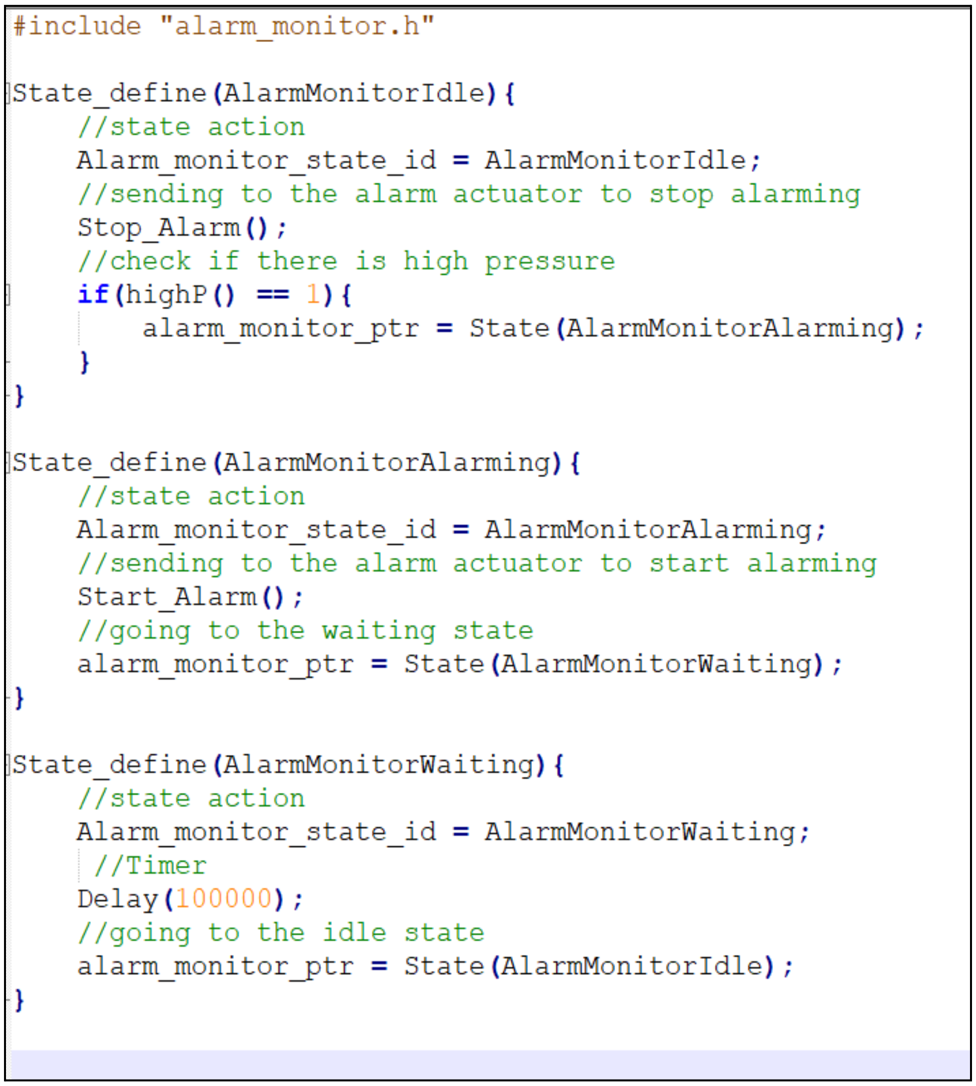
```
#include "algo.h"
 //variables
int algo_pval;
int algo_threshold ;

State_define(High_pressure_detection){
    //state action
    Algo_state_id = High_pressure_detection;
    //receiving the pressure value
    algo_pval = send_pval();
    //stay in the same state
    algo_ptr = State(High_pressure_detection);
}

//sending to alarm monitor if there is high pressure or not
int highP(void){
    algo_threshold =20;
    return(algo_pval>=algo_threshold);
}
```



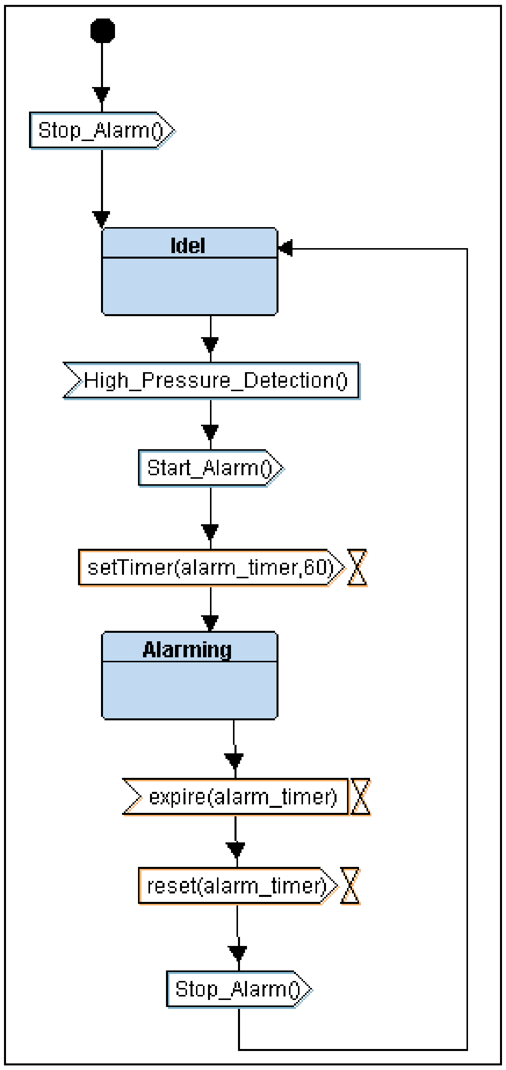
```
WIN 10@DESKTOP-BHGVA79 MINGW32 /d/C/PressureControl
$ arm-none-eabi-objdump -h algo.o

algo.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         0000005c  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  00000090  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  00000090  2**0
                  ALLOC
  3 .debug_info   00000a1a  00000000  00000000  00000090  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000001d8  00000000  00000000  00000aaa  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000070  00000000  00000000  00000c82  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000cf2  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   0000018d  00000000  00000000  00000d12  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    0000056a  00000000  00000000  00000e9f  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007f  00000000  00000000  00001409  2**0
                  CONTENTS, READONLY
 10 .debug_frame  0000004c  00000000  00000000  00001488  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 11 .ARM.attributes 00000033  00000000  00000000  000014d4  2**0
                  CONTENTS, READONLY
```

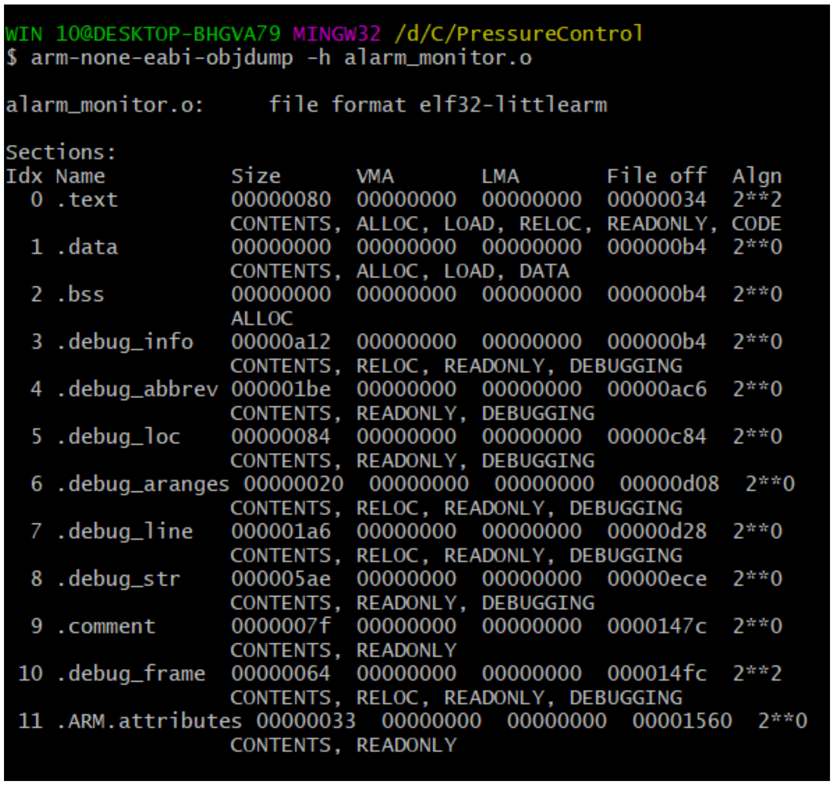

Read_Pressure_Value(pval)
High_Pressure_Detection
[pval < threshold]
[pval >= threshold]
Read_Pressure_Value(pval)
High_Pressure_Detection()
Read_Pressure_Value(pval)

- Alarm Monitor:

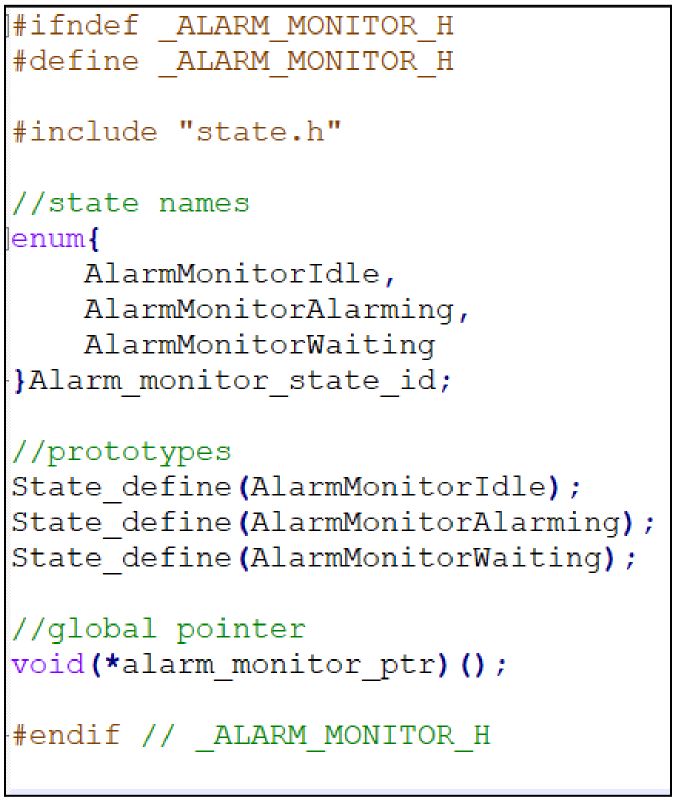


```c
#include "alarm_monitor.h"

State_define(AlarmMonitorIdle){
    //state action
    Alarm_monitor_state_id = AlarmMonitorIdle;
    //sending to the alarm actuator to stop alarming
    Stop_Alarm();
    //check if there is high pressure
    if(highP() == 1){
        alarm_monitor_ptr = State(AlarmMonitorAlarming);
    }
}

State_define(AlarmMonitorAlarming){
    //state action
    Alarm_monitor_state_id = AlarmMonitorAlarming;
    //sending to the alarm actuator to start alarming
    Start_Alarm();
    //going to the waiting state
    alarm_monitor_ptr = State(AlarmMonitorWaiting);
}

State_define(AlarmMonitorWaiting){
    //state action
    Alarm_monitor_state_id = AlarmMonitorWaiting;
     //Timer
    Delay(100000);
    //going to the idle state
    alarm_monitor_ptr = State(AlarmMonitorIdle);
}
```



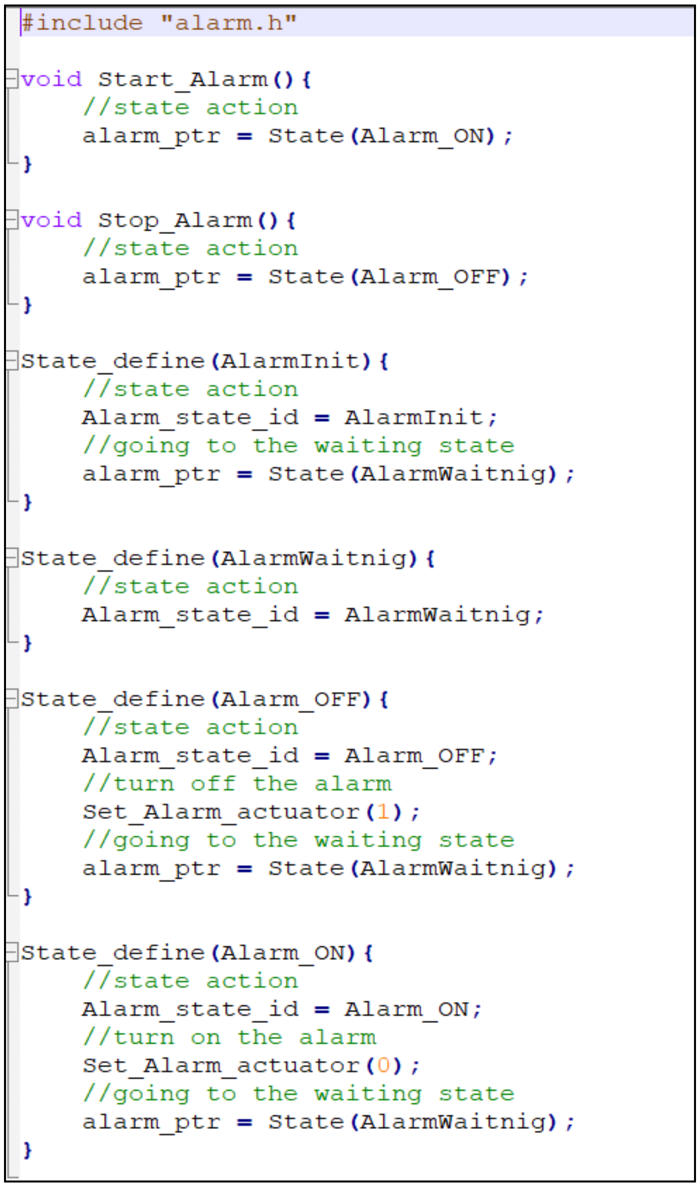


```
WIN 10@DESKTOP-BHGVA79 MINGW32 /d/C/PressureControl
$ arm-none-eabi-objdump -h alarm_monitor.o

alarm_monitor.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000080  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000b4  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000b4  2**0
                  ALLOC
  3 .debug_info   00000a12  00000000  00000000  000000b4  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000001be  00000000  00000000  00000ac6  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000084  00000000  00000000  00000c84  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000d08  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   000001a6  00000000  00000000  00000d28  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000005ae  00000000  00000000  00000ece  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007f  00000000  00000000  0000147c  2**0
                  CONTENTS, READONLY
 10 .debug_frame  00000064  00000000  00000000  000014fc  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 11 .ARM.attributes 00000033  00000000  00000000  00001560  2**0
                  CONTENTS, READONLY
```

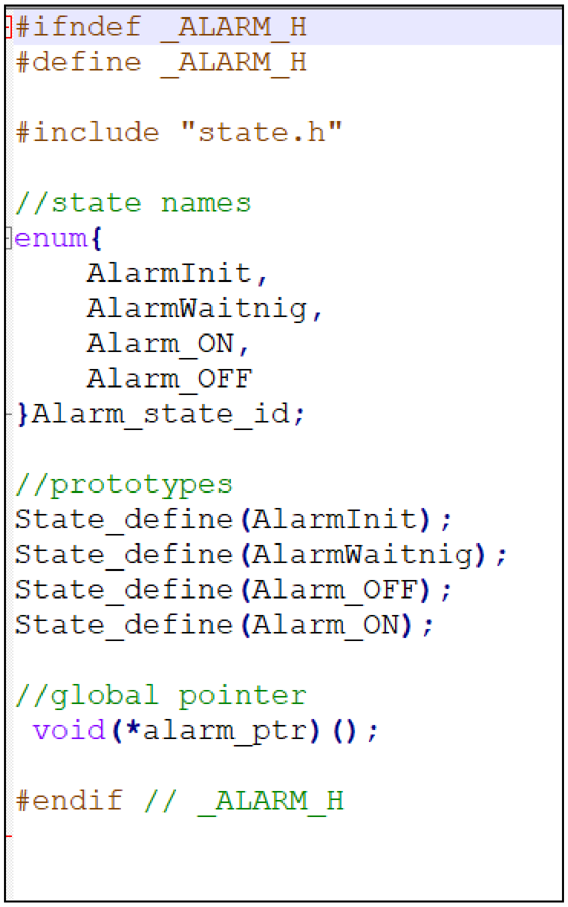


```c
#ifndef _ALARM_MONITOR_H
#define _ALARM_MONITOR_H

#include "state.h"

//state names
enum{
    AlarmMonitorIdle,
    AlarmMonitorAlarming,
    AlarmMonitorWaiting
}Alarm_monitor_state_id;

//prototypes
State_define(AlarmMonitorIdle);
State_define(AlarmMonitorAlarming);
State_define(AlarmMonitorWaiting);

//global pointer
void(*alarm_monitor_ptr)();

#endif // _ALARM_MONITOR_H
```

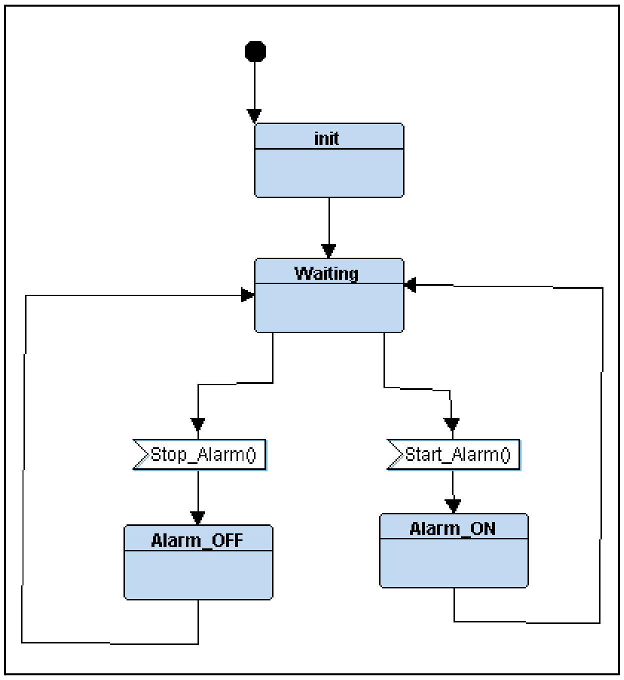
- Alarm Actuator:

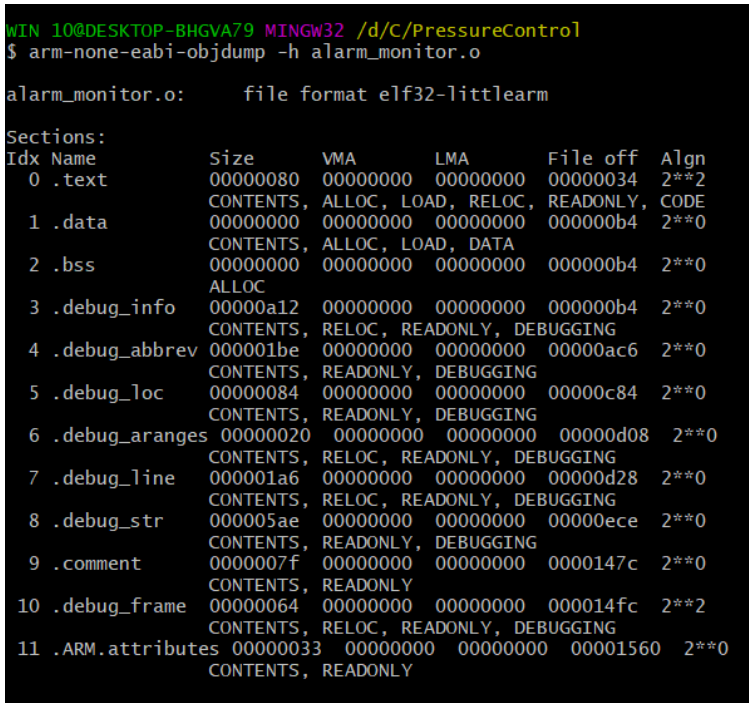


```c
#include "alarm.h"

void Start_Alarm(){
	//state action
	alarm_ptr = State(Alarm_ON);
}

void Stop_Alarm(){
	//state action
	alarm_ptr = State(Alarm_OFF);
}

State_define(AlarmInit){
	//state action
	Alarm_state_id = AlarmInit;
	//going to the waiting state
	alarm_ptr = State(AlarmWaitnig);
}

State_define(AlarmWaitnig){
	//state action
	Alarm_state_id = AlarmWaitnig;
}

State_define(Alarm_OFF){
	//state action
	Alarm_state_id = Alarm_OFF;
	//turn off the alarm
	Set_Alarm_actuator(1);
	//going to the waiting state
	alarm_ptr = State(AlarmWaitnig);
}

State_define(Alarm_ON){
	//state action
	Alarm_state_id = Alarm_ON;
	//turn on the alarm
	Set_Alarm_actuator(0);
	//going to the waiting state
	alarm_ptr = State(AlarmWaitnig);
}
```



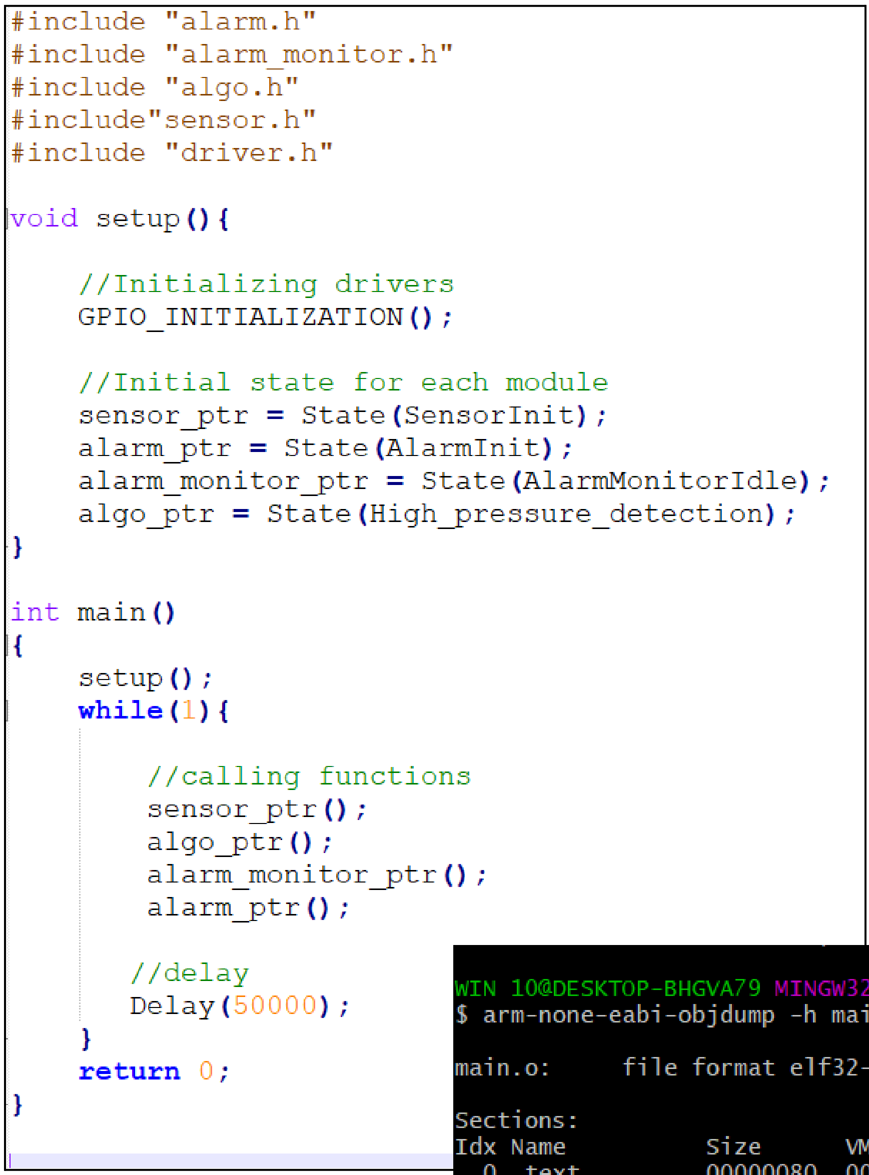
```c
#ifndef _ALARM_H
#define _ALARM_H

#include "state.h"

//state names
enum{
	AlarmInit,
	AlarmWaitnig,
	Alarm_ON,
	Alarm_OFF
}Alarm_state_id;

//prototypes
State_define(AlarmInit);
State_define(AlarmWaitnig);
State_define(Alarm_OFF);
State_define(Alarm_ON);

//global pointer
 void(*alarm_ptr)();

#endif // _ALARM_H
```



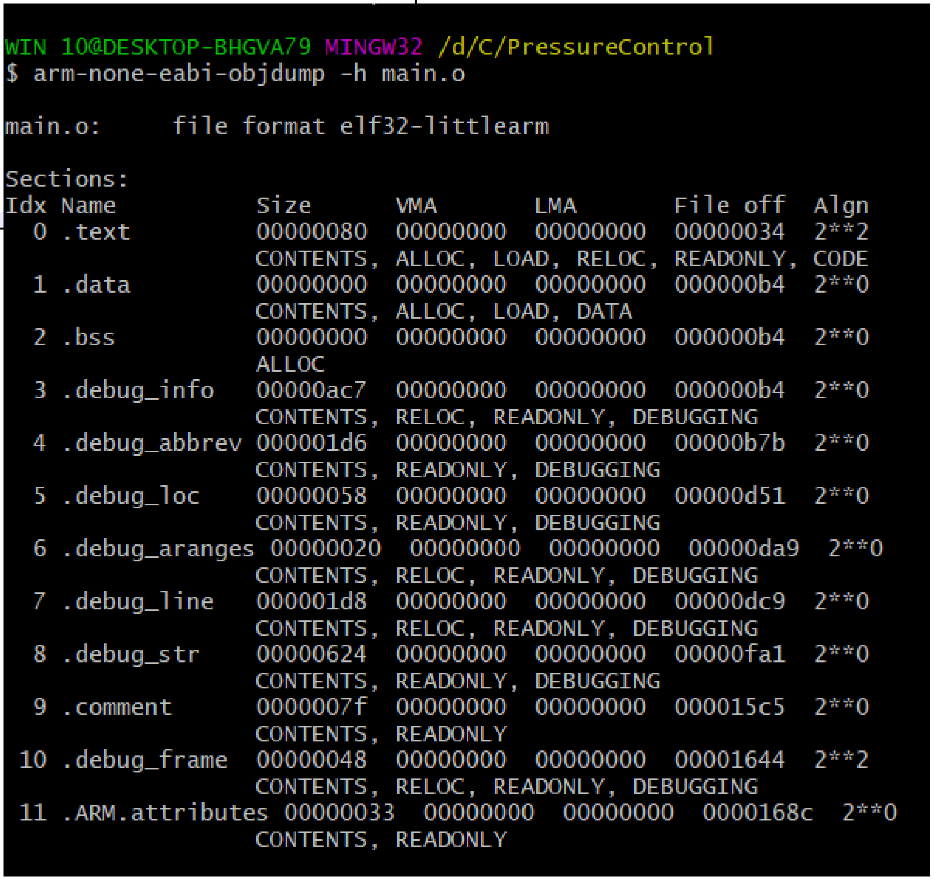


```
WIN 10@DESKTOP-BHGVA79 MINGW32 /d/C/PressureControl
$ arm-none-eabi-objdump -h alarm_monitor.o

alarm_monitor.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000080  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000b4  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000b4  2**0
                  ALLOC
  3 .debug_info   00000a12  00000000  00000000  000000b4  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000001be  00000000  00000000  00000ac6  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000084  00000000  00000000  00000c84  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000d08  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   000001a6  00000000  00000000  00000d28  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    000005ae  00000000  00000000  00000ece  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007f  00000000  00000000  0000147c  2**0
                  CONTENTS, READONLY
 10 .debug_frame  00000064  00000000  00000000  000014fc  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 11 .ARM.attributes 00000033  00000000  00000000  00001560  2**0
                  CONTENTS, READONLY
```

- Main

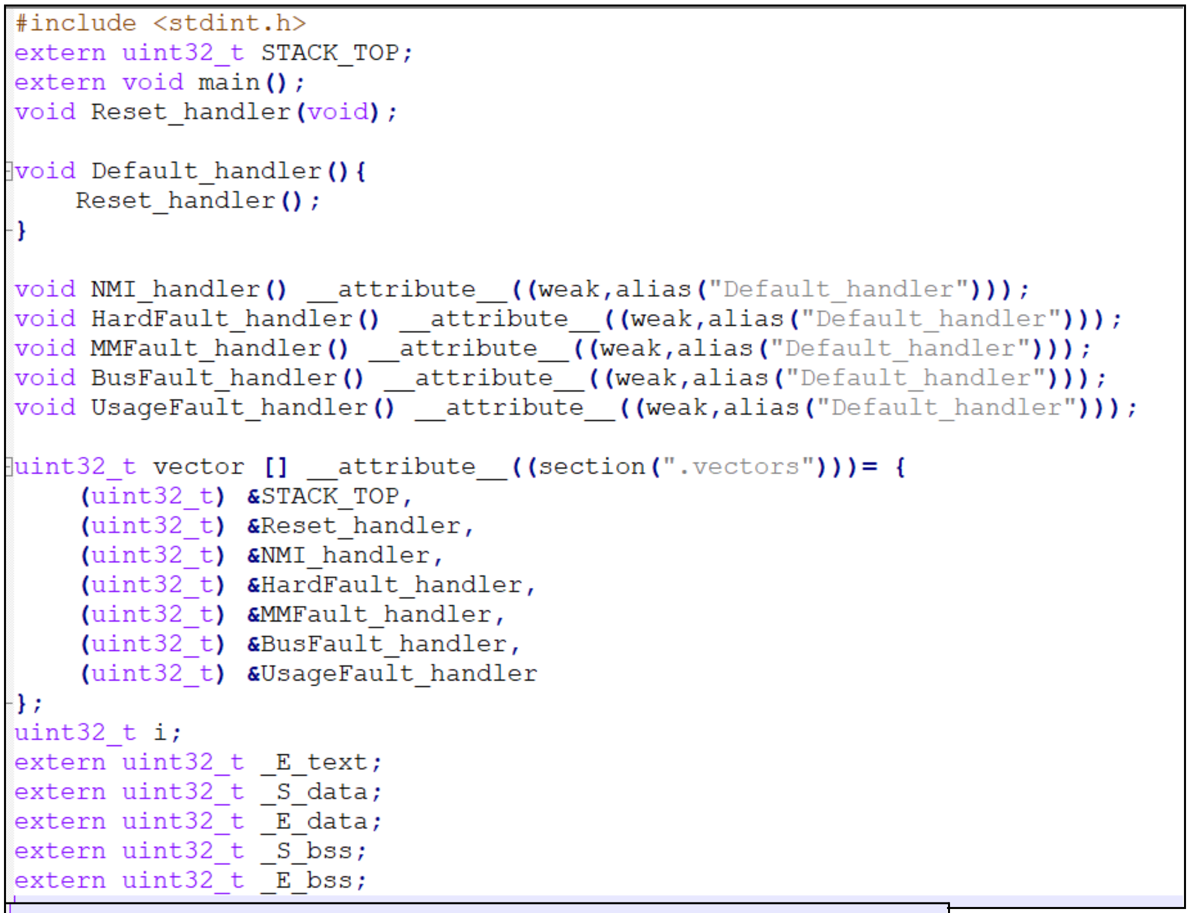


```c
#include "alarm.h"
#include "alarm_monitor.h"
#include "algo.h"
#include"sensor.h"
#include "driver.h"

void setup(){

    //Initializing drivers
    GPIO_INITIALIZATION();

    //Initial state for each module
    sensor_ptr = State(SensorInit);
    alarm_ptr = State(AlarmInit);
    alarm_monitor_ptr = State(AlarmMonitorIdle);
    algo_ptr = State(High_pressure_detection);
}

int main()
{
    setup();
    while(1){

        //calling functions
        sensor_ptr();
        algo_ptr();
        alarm_monitor_ptr();
        alarm_ptr();

       //delay
       Delay(50000);
    }
    return 0;
}
```

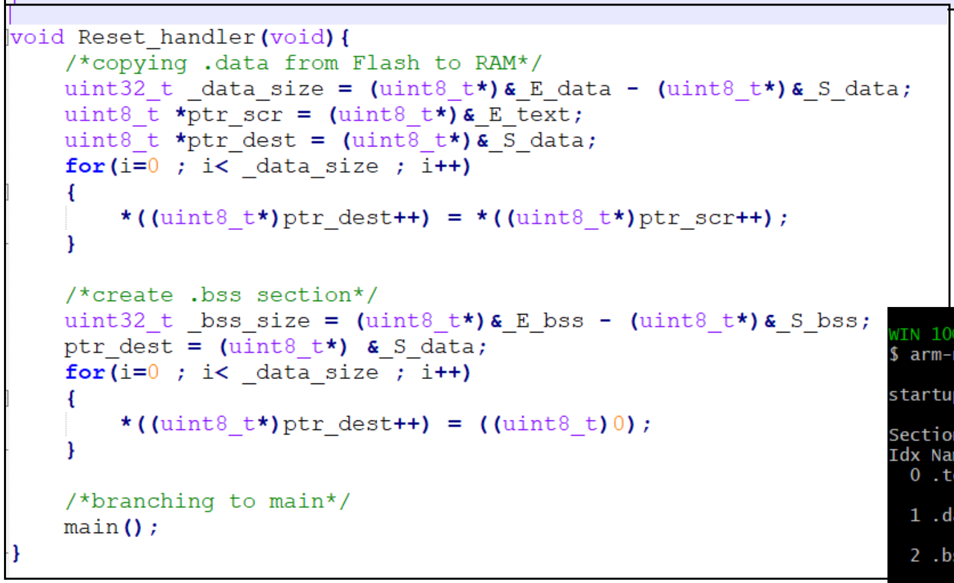


```
WIN 10@DESKTOP-BHGVA79 MINGW32 /d/C/PressureControl
$ arm-none-eabi-objdump -h main.o

main.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000080  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000b4  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000b4  2**0
                  ALLOC
  3 .debug_info   00000ac7  00000000  00000000  000000b4  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  4 .debug_abbrev 000001d6  00000000  00000000  00000b7b  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000058  00000000  00000000  00000d51  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000da9  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   000001d8  00000000  00000000  00000dc9  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_str    00000624  00000000  00000000  00000fa1  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007f  00000000  00000000  000015c5  2**0
                  CONTENTS, READONLY
 10 .debug_frame  00000048  00000000  00000000  00001644  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 11 .ARM.attributes 00000033  00000000  00000000  0000168c  2**0
                  CONTENTS, READONLY
```

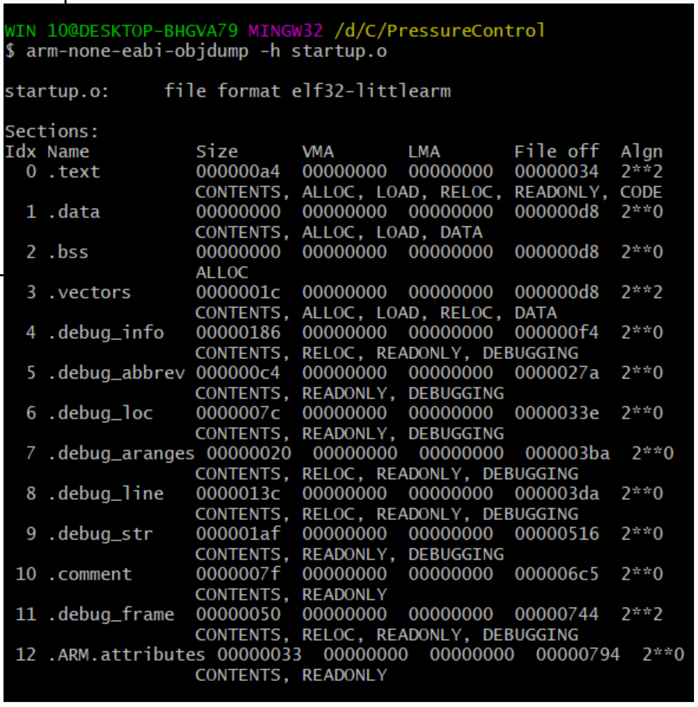
- Startup:



```c
#include <stdint.h>
extern uint32_t STACK_TOP;
extern void main();
void Reset_handler(void);

void Default_handler(){
    Reset_handler();
}

void NMI_handler() __attribute__((weak,alias("Default_handler")));
void HardFault_handler() __attribute__((weak,alias("Default_handler")));
void MMFault_handler() __attribute__((weak,alias("Default_handler")));
void BusFault_handler() __attribute__((weak,alias("Default_handler")));
void UsageFault_handler() __attribute__((weak,alias("Default_handler")));

uint32_t vector [] __attribute__((section(".vectors")))= {
    (uint32_t) &STACK_TOP,
    (uint32_t) &Reset_handler,
    (uint32_t) &NMI_handler,
    (uint32_t) &HardFault_handler,
    (uint32_t) &MMFault_handler,
    (uint32_t) &BusFault_handler,
    (uint32_t) &UsageFault_handler
};
uint32_t i;
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;
```

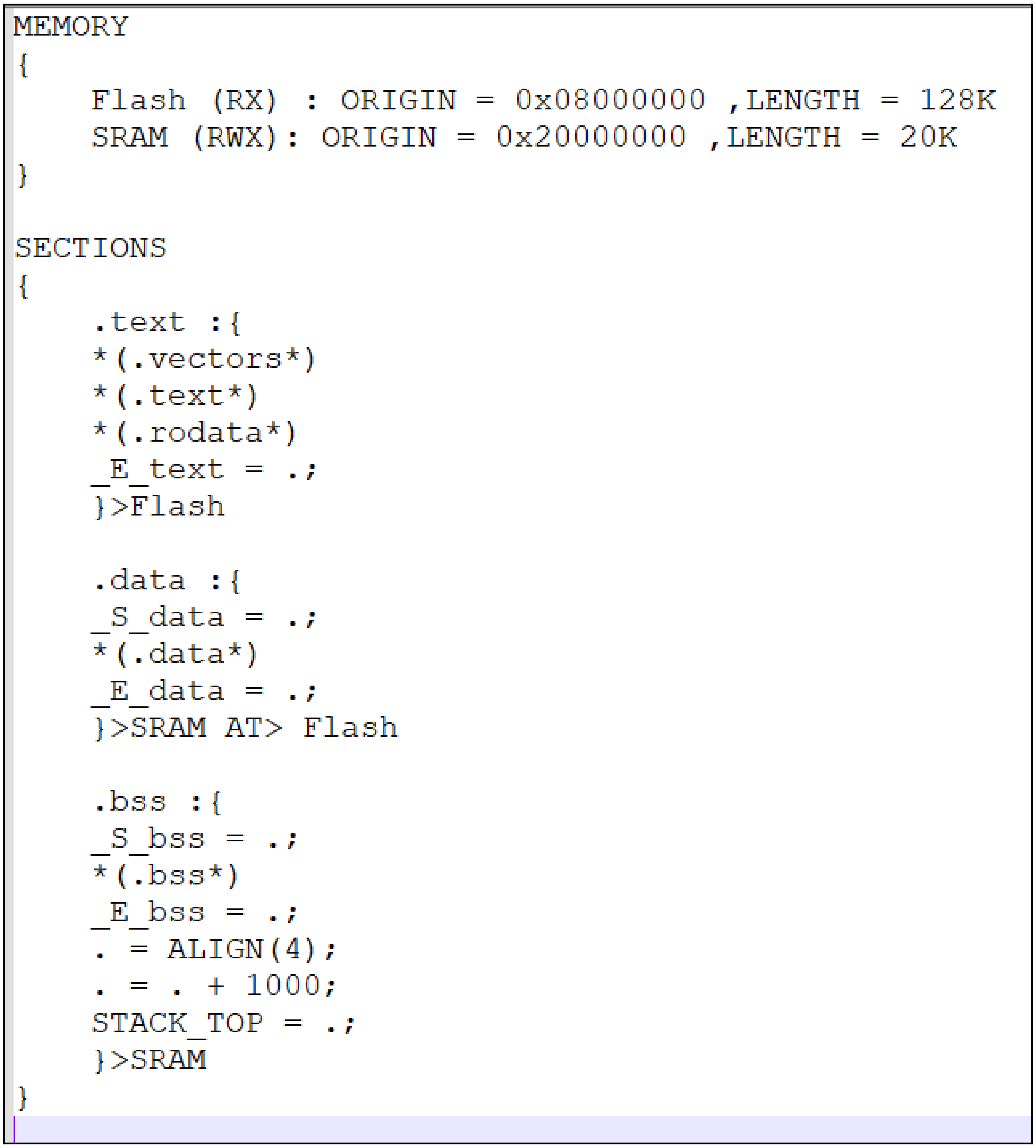


```c
void Reset_handler(void){
    /*copying .data from Flash to RAM*/
    uint32_t _data_size = (uint8_t*)&_E_data - (uint8_t*)&_S_data;
    uint8_t *ptr_scr = (uint8_t*)&_E_text;
    uint8_t *ptr_dest = (uint8_t*)&_S_data;
    for(i=0 ; i< _data_size ; i++)
    {
        *((uint8_t*)ptr_dest++) = *((uint8_t*)ptr_scr++);
    }

    /*create .bss section*/
    uint32_t _bss_size = (uint8_t*)&_E_bss - (uint8_t*)&_S_bss;
    ptr_dest = (uint8_t*) &_S_data;
    for(i=0 ; i< _data_size ; i++)
    {
        *((uint8_t*)ptr_dest++) = ((uint8_t)0);
    }

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



```
WIN 10@DESKTOP-BHGVA79 MINGW32 /d/C/PressureControl
$ arm-none-eabi-objdump -h startup.o

startup.o:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         000000a4  00000000  00000000  00000034  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
  1 .data         00000000  00000000  00000000  000000d8  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00000000  00000000  00000000  000000d8  2**0
                  ALLOC
  3 .vectors      0000001c  00000000  00000000  000000d8  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, DATA
  4 .debug_info   00000186  00000000  00000000  000000f4  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  5 .debug_abbrev 000000c4  00000000  00000000  0000027a  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_loc    0000007c  00000000  00000000  0000033e  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_aranges 00000020  00000000  00000000  000003ba  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  8 .debug_line   0000013c  00000000  00000000  000003da  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  9 .debug_str    000001af  00000000  00000000  00000516  2**0
                  CONTENTS, READONLY, DEBUGGING
 10 .comment      0000007f  00000000  00000000  000006c5  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000050  00000000  00000000  00000744  2**2
                  CONTENTS, RELOC, READONLY, DEBUGGING
 12 .ARM.attributes 00000033  00000000  00000000  00000794  2**0
                  CONTENTS, READONLY
```

- Linker Script:

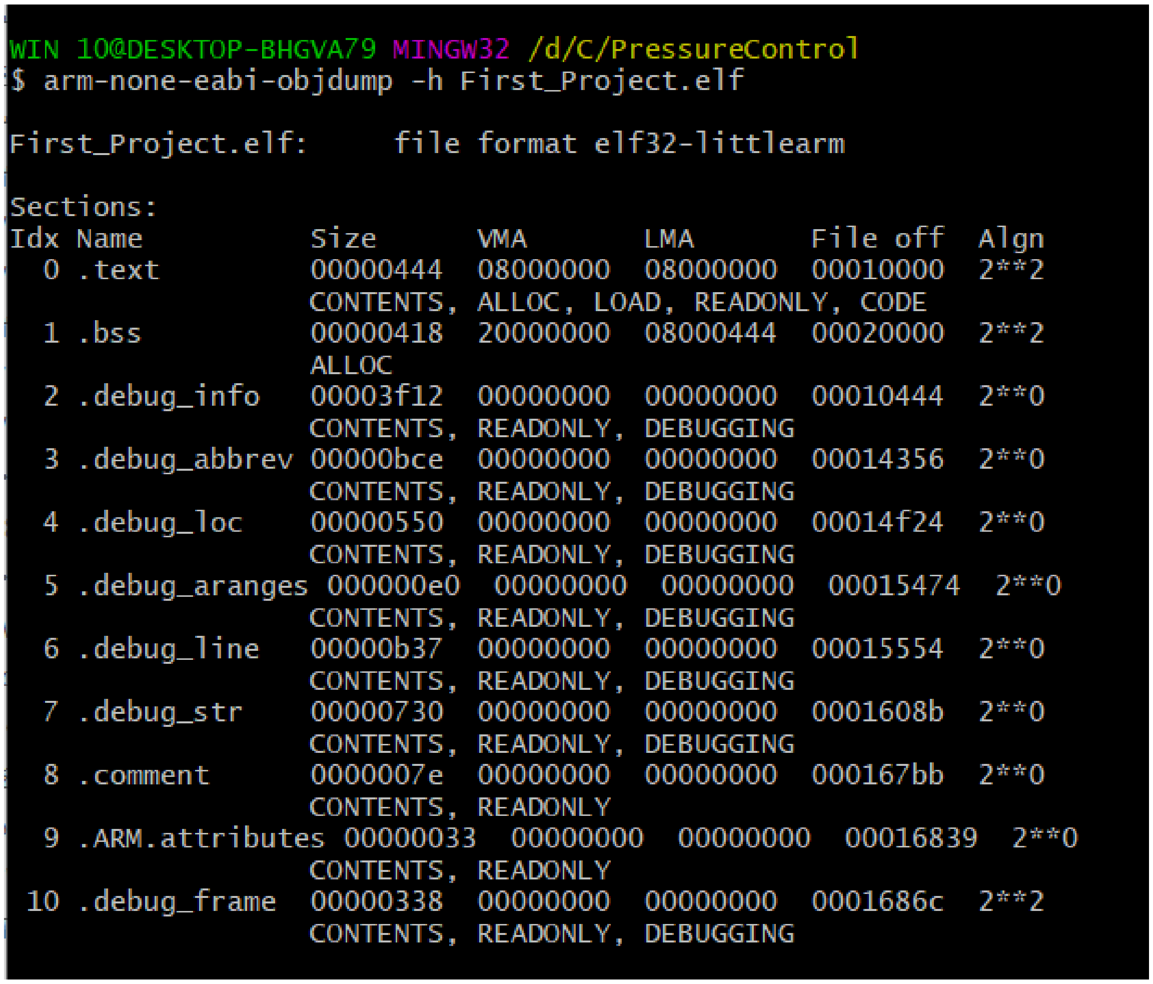


```
MEMORY
{
    Flash (RX) : ORIGIN = 0x08000000 ,LENGTH = 128K
    SRAM (RWX): ORIGIN = 0x20000000 ,LENGTH = 20K
}

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

    .data :{
    _S_data = .;
    *(.data*)
    _E_data = .;
    }>SRAM AT> Flash

    .bss :{
    _S_bss = .;
    *(.bss*)
    _E_bss = .;
    . = ALIGN(4);
    . = . + 1000;
    STACK_TOP = .;
    }>SRAM
}
```

- Make file:

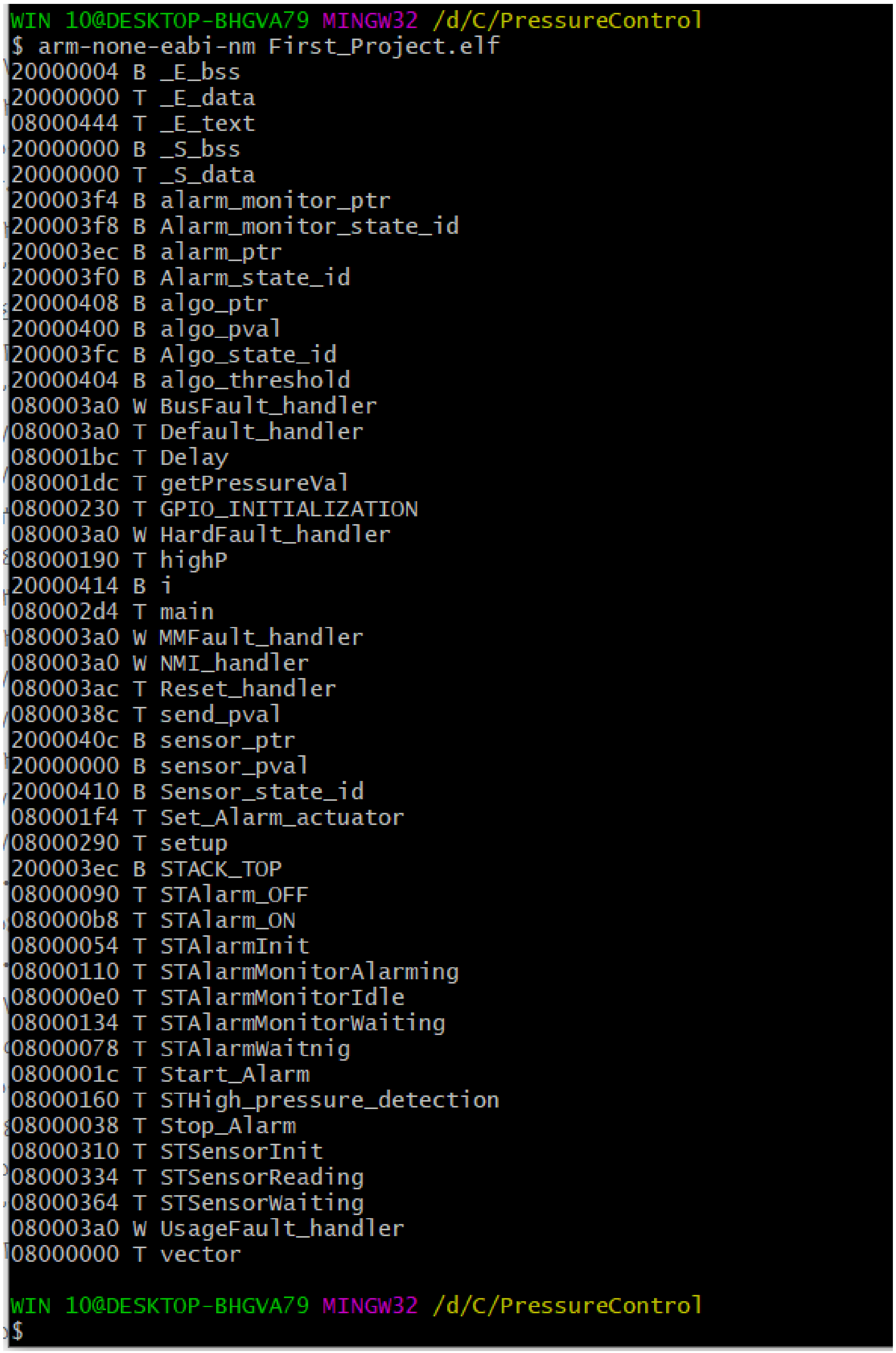
```
CC=arm-none-eabi-
CFLAGS= -mcpu=cortex-m3 -gdwarf-2
INCS=-I.
SRC= $(wildcard *.c)
OBJ= $(SRC:.c=.o)
AS= $(wildcard *.s)
OBJAS= $(AS:.s=.o)
NAME=First_Project

all: $(NAME).bin

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

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


$(NAME).elf:$(OBJ) $(OBJAS)
    $(CC)ld.exe -T linker_script.ld $(OBJ) $(OBJAS) -o $@ -Map=map_flie.map

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

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

- Executable file sections:



```
WIN 10@DESKTOP-BHGVA79 MINGW32 /d/C/PressureControl
$ arm-none-eabi-objdump -h First_Project.elf

First_Project.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000444  08000000  08000000  00010000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .bss          00000418  20000000  08000444  00020000  2**2
                  ALLOC
  2 .debug_info   00003f12  00000000  00000000  00010444  2**0
                  CONTENTS, READONLY, DEBUGGING
  3 .debug_abbrev 00000bce  00000000  00000000  00014356  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_loc    00000550  00000000  00000000  00014f24  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_aranges 000000e0  00000000  00000000  00015474  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_line   00000b37  00000000  00000000  00015554  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_str    00000730  00000000  00000000  0001608b  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .comment      0000007e  00000000  00000000  000167bb  2**0
                  CONTENTS, READONLY
  9 .ARM.attributes 00000033  00000000  00000000  00016839  2**0
                  CONTENTS, READONLY
 10 .debug_frame  00000338  00000000  00000000  0001686c  2**2
                  CONTENTS, READONLY, DEBUGGING
```

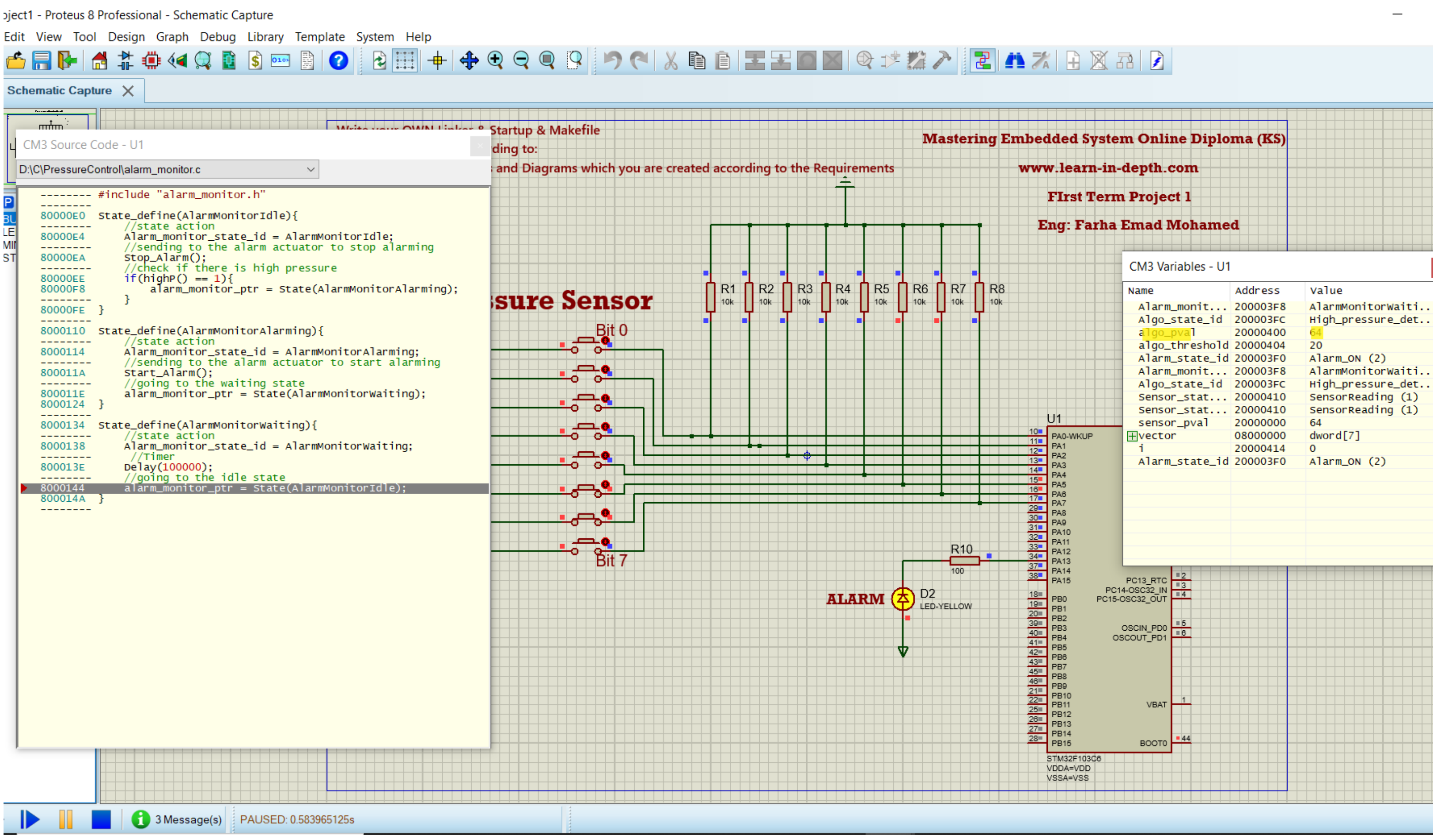
- Executable file symbol table:



```
WIN 10@DESKTOP-BHGVA79 MINGW32 /d/C/PressureControl
$ arm-none-eabi-nm First_Project.elf
20000004 B _E_bss
20000000 T _E_data
08000444 T _E_text
20000000 B _S_bss
20000000 T _S_data
200003f4 B alarm_monitor_ptr
200003f8 B Alarm_monitor_state_id
200003ec B alarm_ptr
200003f0 B Alarm_state_id
20000408 B algo_ptr
20000400 B algo_pval
200003fc B Algo_state_id
20000404 B algo_threshold
080003a0 W BusFault_handler
080003a0 T Default_handler
080001bc T Delay
080001dc T getPressureVal
08000230 T GPIO_INITIALIZATION
080003a0 W HardFault_handler
08000190 T highP
20000414 B i
080002d4 T main
080003a0 W MMFault_handler
080003a0 W NMI_handler
080003ac T Reset_handler
0800038c T send_pval
2000040c B sensor_ptr
20000000 B sensor_pval
20000410 B Sensor_state_id
080001f4 T Set_Alarm_actuator
08000290 T setup
200003ec B STACK_TOP
08000090 T STAlarm_OFF
080000b8 T STAlarm_ON
08000054 T STAlarmInit
08000110 T STAlarmMonitorAlarming
080000e0 T STAlarmMonitorIdle
08000134 T STAlarmMonitorWaiting
08000078 T STAlarmWaitnig
0800001c T Start_Alarm
08000160 T STHigh_pressure_detection
08000038 T Stop_Alarm
08000310 T STSensorInit
08000334 T STSensorReading
08000364 T STSensorWaiting
080003a0 W UsageFault_handler
08000000 T vector

WIN 10@DESKTOP-BHGVA79 MINGW32 /d/C/PressureControl
$
```

- Output:

1) High pressure is detected and the alarm is on.



oject1 - Proteus 8 Professional - Schematic Capture
Edit View Tool Design Graph Debug Library Template System Help
Schematic Capture
Write your OWN Linker & Startup & Makefile
ding to:
and Diagrams which you are created according to the Requirements
Mastering Embedded System Online Diploma (KS)
www.learn-in-depth.com
FIrst Term Project 1
Eng: Farha Emad Mohamed
ssure Sensor
Bit 0
Bit 7
R1 R2 R3 R4 R5 R6 R7 R8
10k
R10
100
ALARM
D2
LED-YELLOW
U1
STM32F103C6
VDDA=VDD
VSSA=VSS
3 Message(s)
PAUSED: 0.583965125s

CM3 Source Code - U1

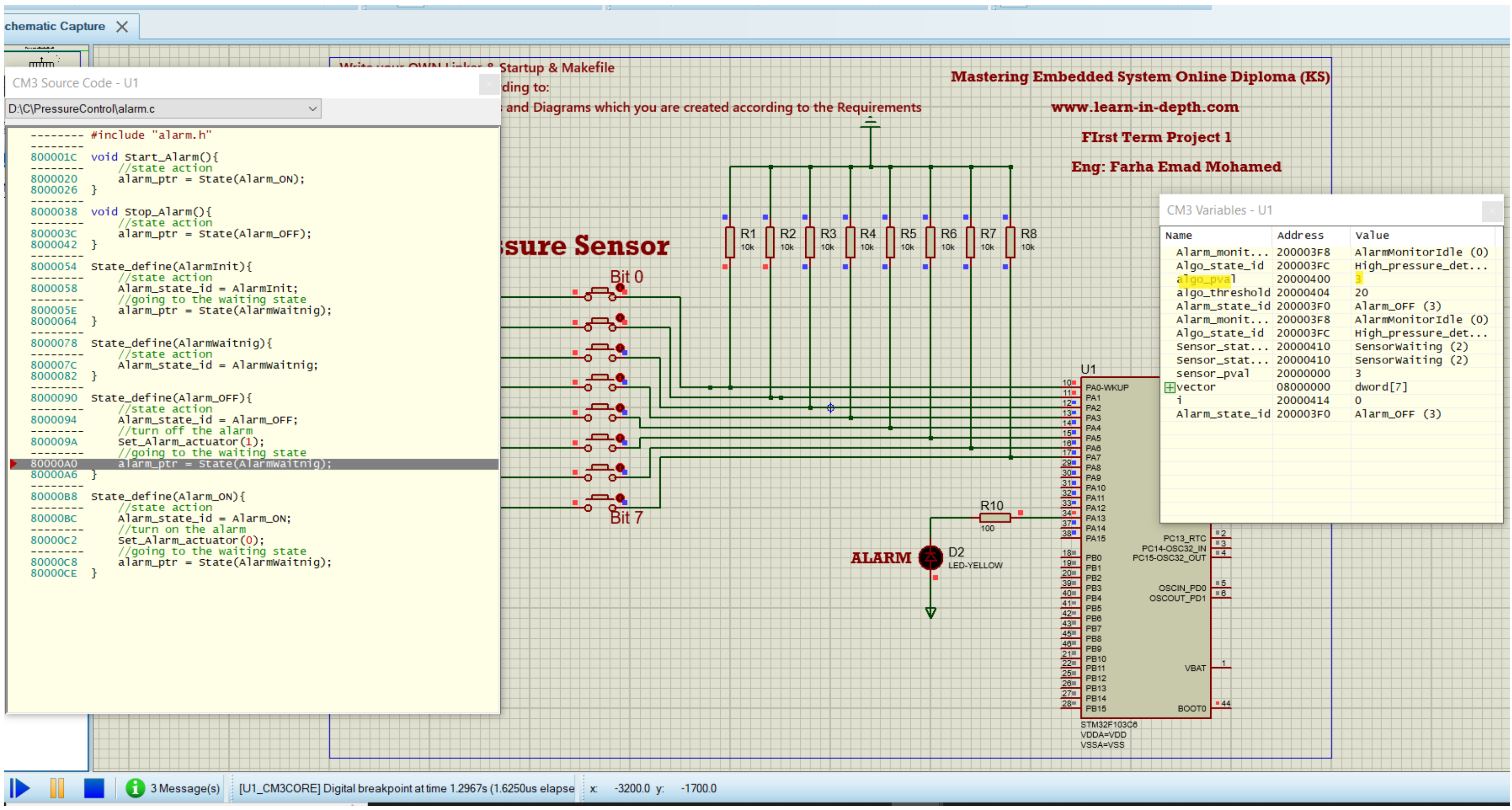
D:\C\PressureControl\alarm_monitor.c

```
-------- #include "alarm_monitor.h"
--------
80000E0  State_define(AlarmMonitorIdle){
--------     //state action
80000E4      Alarm_monitor_state_id = AlarmMonitorIdle;
--------     //sending to the alarm actuator to stop alarming
80000EA      Stop_Alarm();
--------     //check if there is high pressure
80000EE      if(highP() == 1){
80000F8          alarm_monitor_ptr = State(AlarmMonitorAlarming);
--------     }
80000FE  }
--------
8000110  State_define(AlarmMonitorAlarming){
--------     //state action
8000114      Alarm_monitor_state_id = AlarmMonitorAlarming;
--------     //sending to the alarm actuator to start alarming
800011A      Start_Alarm();
--------     //going to the waiting state
800011E      alarm_monitor_ptr = State(AlarmMonitorWaiting);
8000124  }
--------
8000134  State_define(AlarmMonitorWaiting){
--------     //state action
8000138      Alarm_monitor_state_id = AlarmMonitorWaiting;
--------      //Timer
800013E      Delay(100000);
--------     //going to the idle state
8000144      alarm_monitor_ptr = State(AlarmMonitorIdle);
800014A  }
--------
```

CM3 Variables - U1

Name	Address	Value
Alarm_monit...	200003F8	AlarmMonitorWaiti...
Algo_state_id	200003FC	High_pressure_det...
algo_pval	20000400	64
algo_threshold	20000404	20
Alarm_state_id	200003F0	Alarm_ON (2)
Alarm_monit...	200003F8	AlarmMonitorWaiti...
Algo_state_id	200003FC	High_pressure_det...
Sensor_stat...	20000410	SensorReading (1)
Sensor_stat...	20000410	SensorReading (1)
sensor_pval	20000000	64
vector	08000000	dword[7]
i	20000414	0
Alarm_state_id	200003F0	Alarm_ON (2)

2) Low pressure and the alarm is off.



```
-------- #include "alarm.h"
--------
800001C  void Start_Alarm(){
--------     //state action
8000020      alarm_ptr = State(Alarm_ON);
8000026  }
--------
8000038  void Stop_Alarm(){
--------     //state action
800003C      alarm_ptr = State(Alarm_OFF);
8000042  }
--------
8000054  State_define(AlarmInit){
--------     //state action
8000058      Alarm_state_id = AlarmInit;
--------     //going to the waiting state
800005E      alarm_ptr = State(AlarmWaitnig);
8000064  }
--------
8000078  State_define(AlarmWaitnig){
--------     //state action
800007C      Alarm_state_id = AlarmWaitnig;
8000082  }
--------
8000090  State_define(Alarm_OFF){
--------     //state action
8000094      Alarm_state_id = Alarm_OFF;
--------     //turn off the alarm
800009A      Set_Alarm_actuator(1);
--------     //going to the waiting state
80000A0      alarm_ptr = State(AlarmWaitnig);
80000A6  }
--------
80000B8  State_define(Alarm_ON){
--------     //state action
80000BC      Alarm_state_id = Alarm_ON;
--------     //turn on the alarm
80000C2      Set_Alarm_actuator(0);
--------     //going to the waiting state
80000C8      alarm_ptr = State(AlarmWaitnig);
80000CE  }
```

Name	Address	Value
Alarm_monit...	200003F8	AlarmMonitorIdle (0)
Algo_state_id	200003FC	High_pressure_det...
algo_pval	20000400	3
algo_threshold	20000404	20
Alarm_state_id	200003F0	Alarm_OFF (3)
Alarm_monit...	200003F8	AlarmMonitorIdle (0)
Algo_state_id	200003FC	High_pressure_det...
Sensor_stat...	20000410	SensorWaiting (2)
Sensor_stat...	20000410	SensorWaiting (2)
sensor_pval	20000000	3
vector	08000000	dword[7]
i	20000414	0
Alarm_state_id	200003F0	Alarm_OFF (3)

```
[U1_CM3CORE] Digital breakpoint at time 1.2967s (1.6250us elapse
```