Mastering Embedded Systems Online Diploma

www.learn-in-depth.com

First-term (Final Project 1)

Eng. Mostafa Mahmoud Helmy Abd-elrahman

My Profile:

www.learn-in-depth.com/online-diploma/darshme7@gmail.com

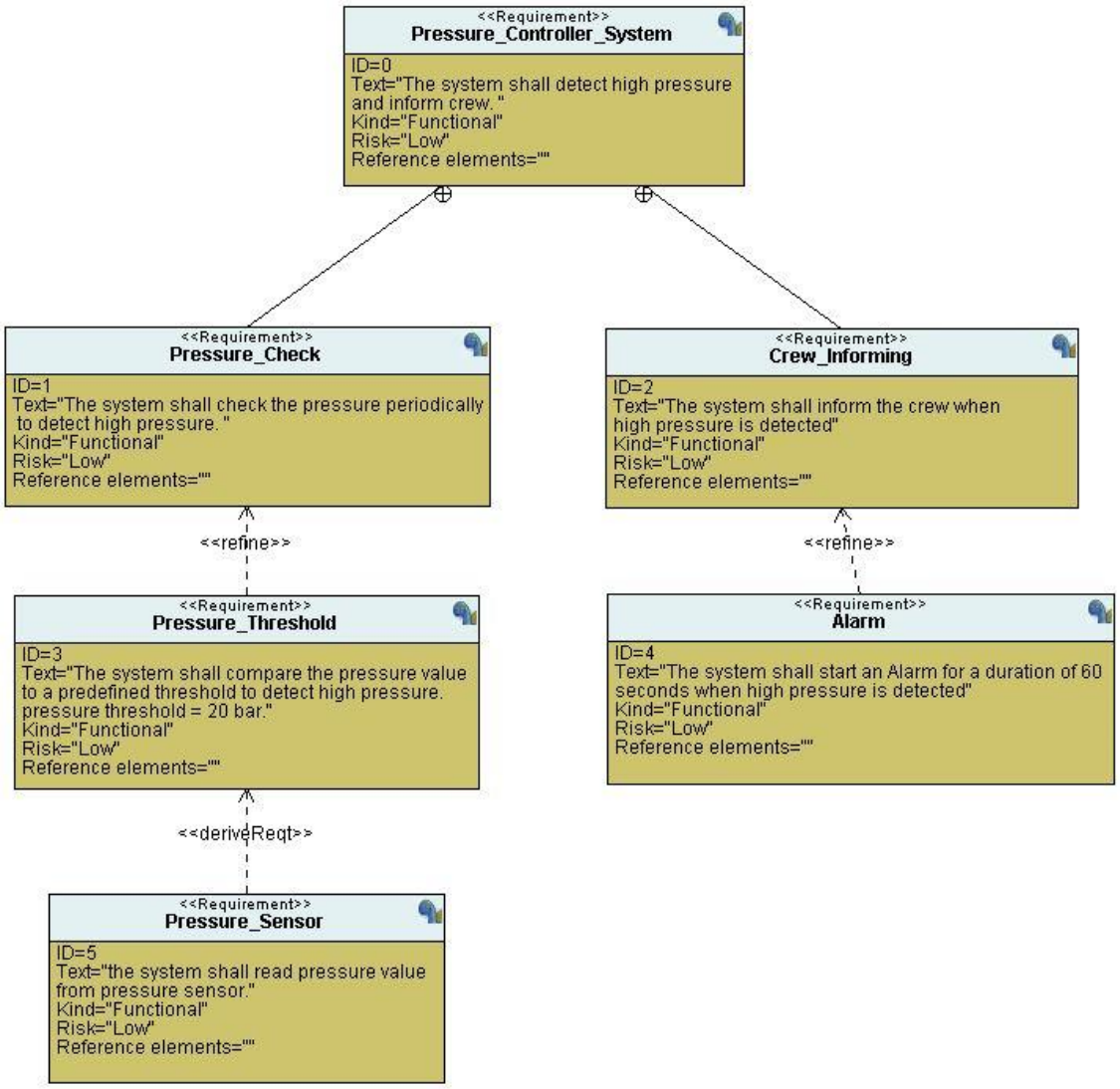
Pressure Controller Project

- **Case study: Pressure controller system**
  - **Pressure controller system should check if the pressure inside a cabin exceeds 20 bar.**
  - **If the pressure inside a cabin exceeds 20 bar the pressure controller system should inform the crew with an alarm for a duration of 60 seconds.**

- **Assumptions:**
  - **System setup and shutdown procedure are not modelled.**
  - **System maintenance is not modelled.**
  - **Pressure sensor never fails.**
  - **Alarm actuator never fails.**
  - **System never faces power cut.**

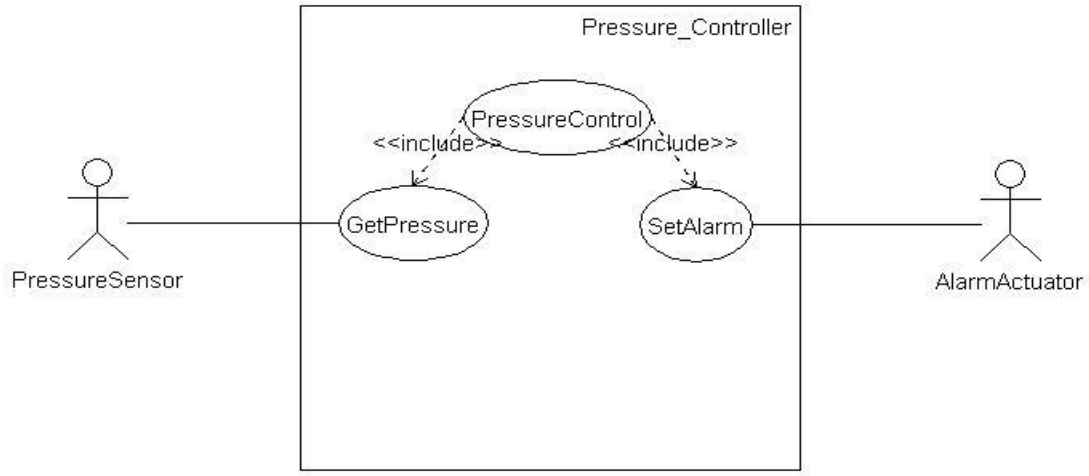
- **Method: for Pressure Controller software development cycle Waterfall Model was found to be the most suitable.**

- **Requirement diagram:**

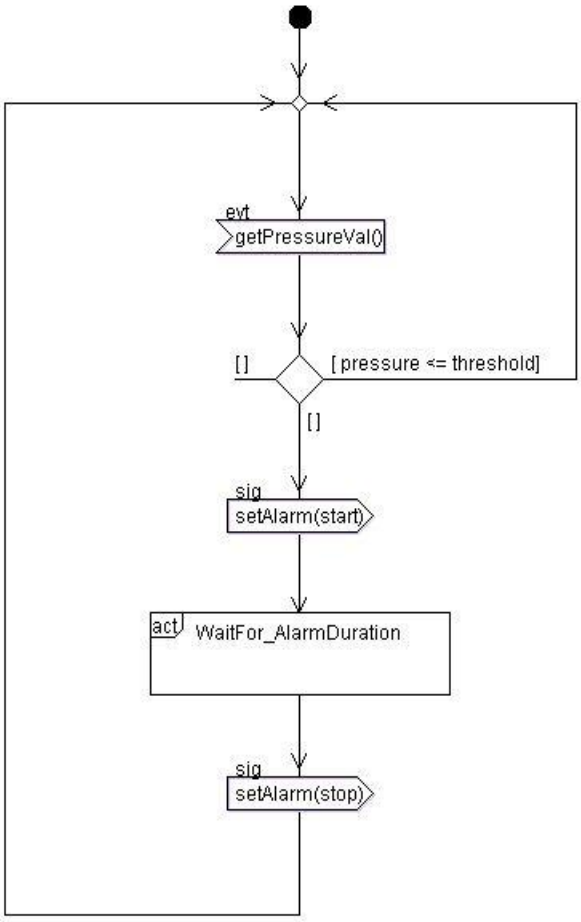


- **Space exploration/Partitioning: the project is quiet simple, it doesn't require more than one ECU and STM32 was found suitable for this project.**

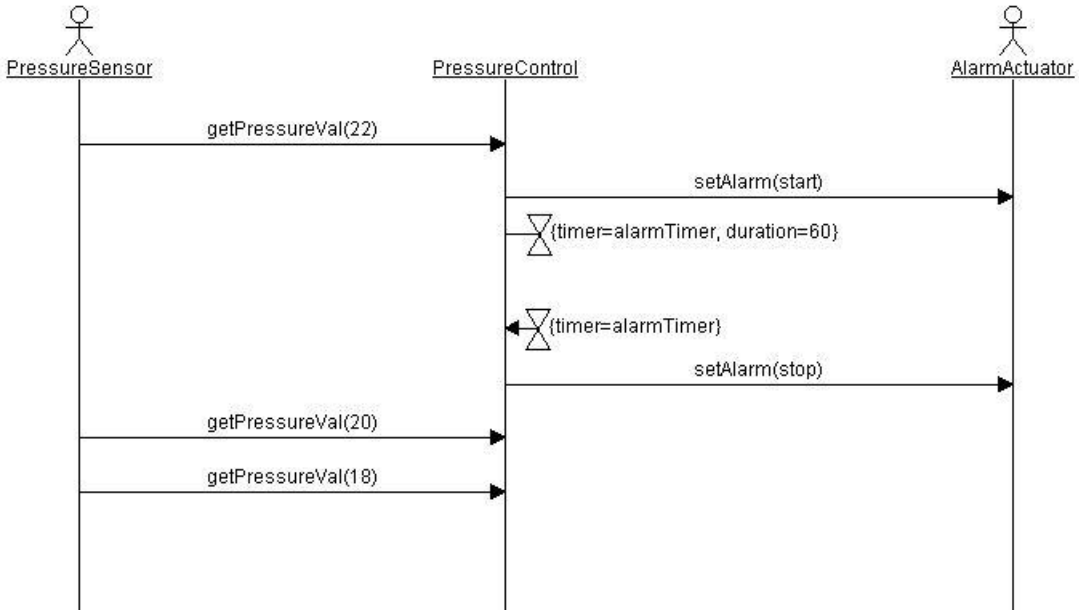
- **System analysis:**
  - **Use Case diagram:**



  - **Activity diagram:**

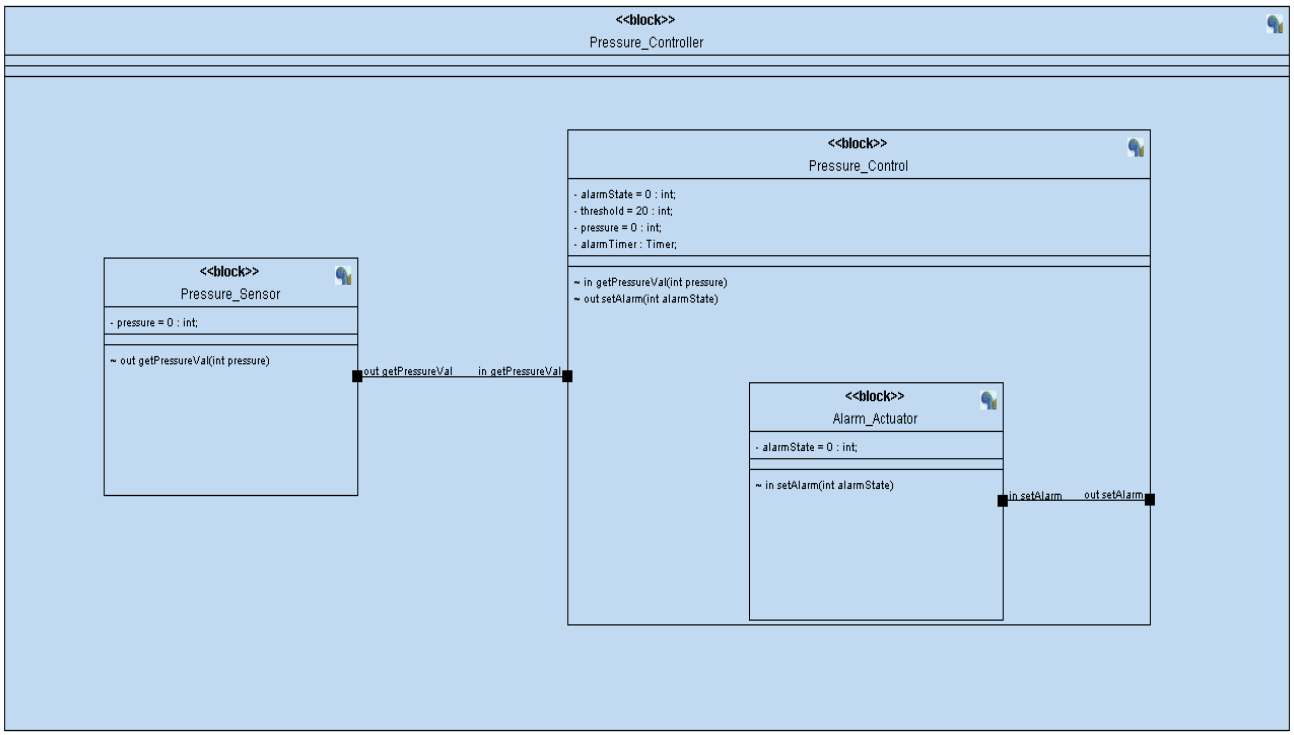


- **Sequence diagram:**

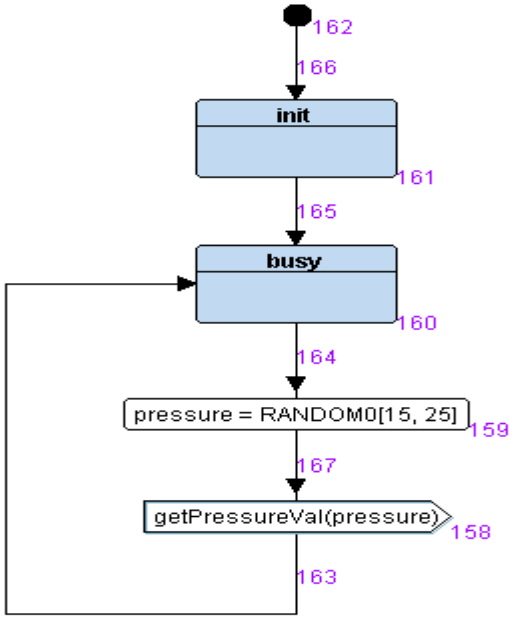


- **System Design:**

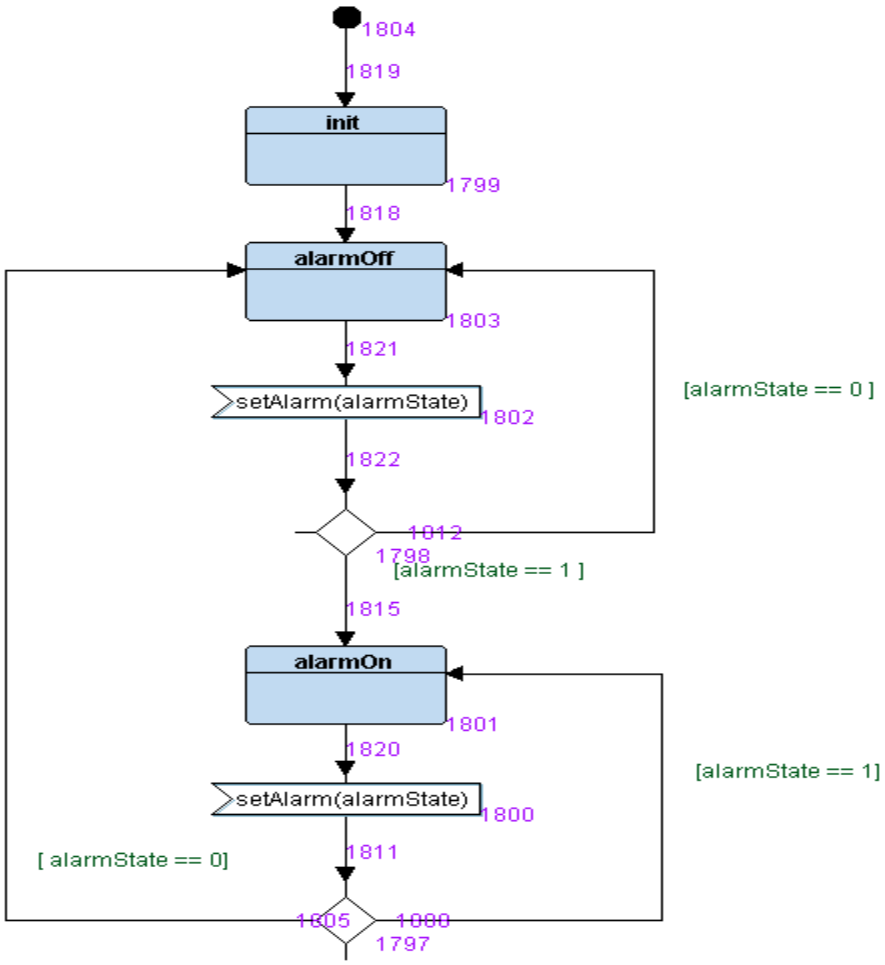
  - **Block diagram:**



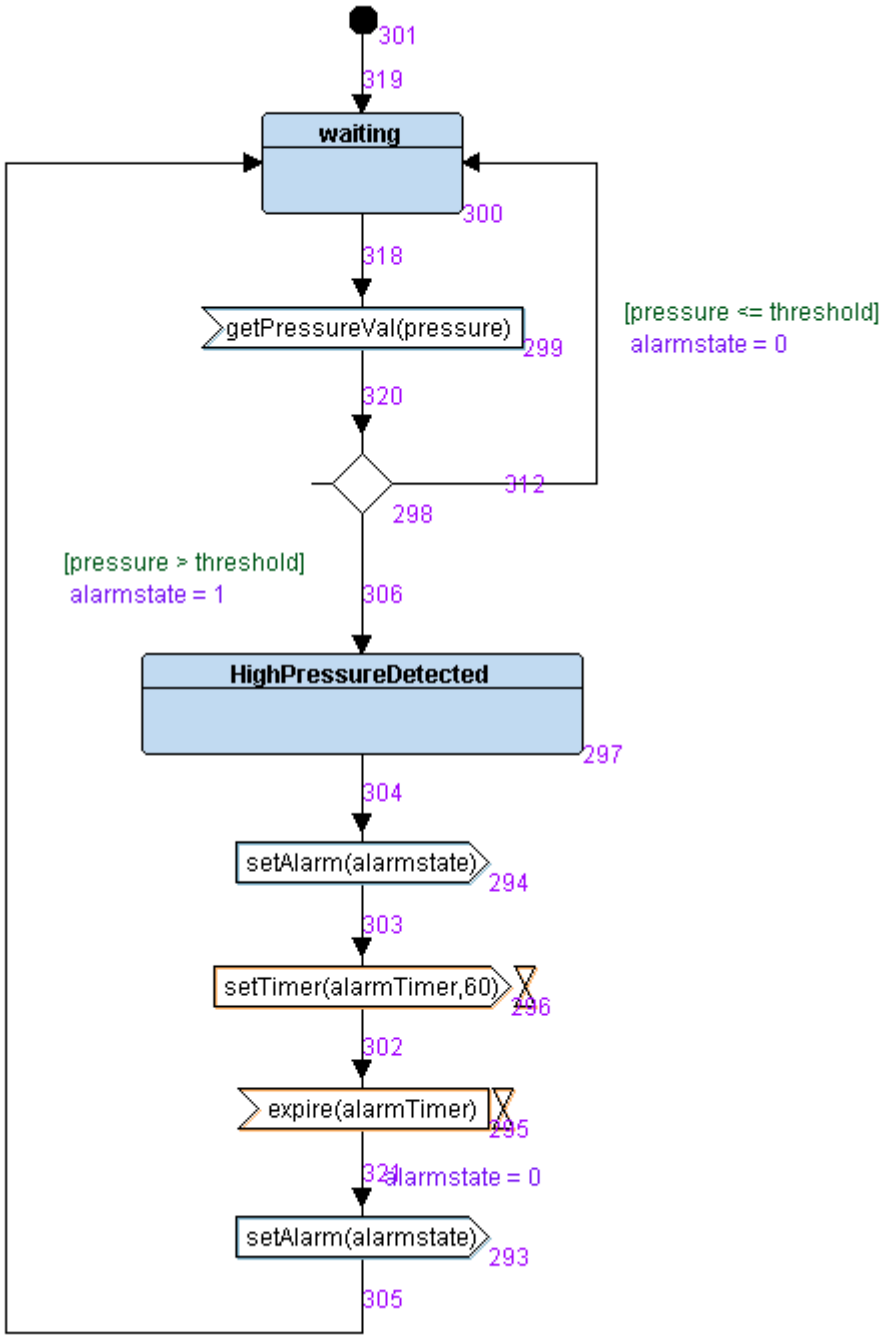
- **State Machine Pressure_Sensor:**



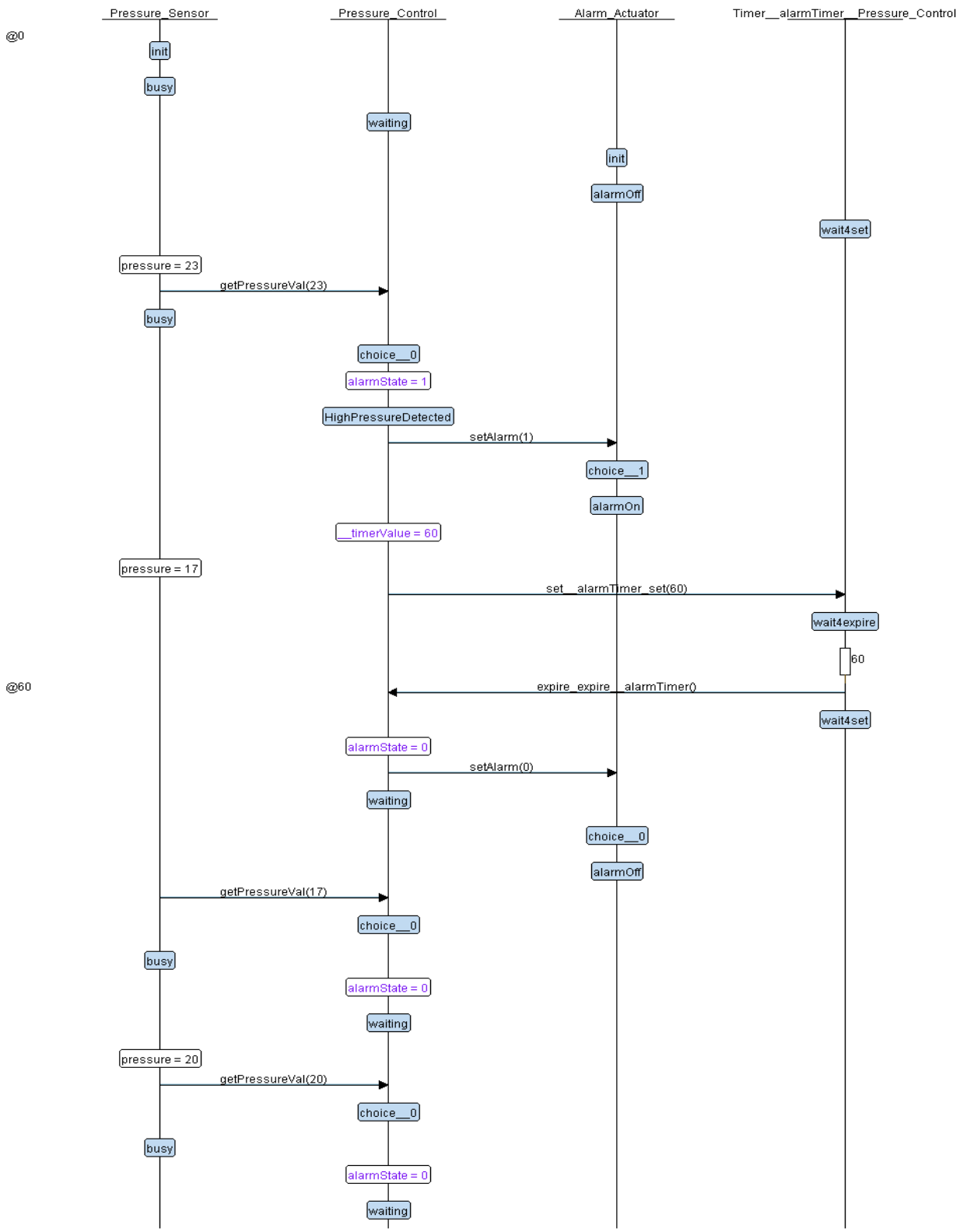
- **State Machine Alarm_Actuator:**



- **State Machine Pressure_Control:**



- **Simulation:**


Pressure_Sensor
Pressure_Control
Alarm_Actuator
Timer__alarmTimer__Pressure_Control
@0
init
busy
waiting
init
alarmOff
wait4set
pressure = 23
getPressureVal(23)
busy
choice__0
alarmState = 1
HighPressureDetected
setAlarm(1)
choice__1
alarmOn
__timerValue = 60
pressure = 17
set__alarmTimer_set(60)
wait4expire
60
@60
expire_expire__alarmTimer()
wait4set
alarmState = 0
setAlarm(0)
waiting
choice__0
alarmOff
getPressureVal(17)
choice__0
busy
alarmState = 0
waiting
pressure = 20
getPressureVal(20)
choice__0
busy
alarmState = 0
waiting

- **Software Implementation of project for each module:**
  - **Pressure_Controller main.c:**

```c
/*
 * main.c
 *
 *  Created on: May 9, 2021
 *      Author: mostafa
 */

#include "PS.h"
#include "PC.h"


void setup(void){

    GPIO_INITIALIZATION();
    PS_state=STATE(PS_busy);
    PC_state=STATE(PC_waiting);

}

int main(void){

    setup();
    while(1){

        PS_state();
        PC_state();
    }

    return 0;
}
```

  - **Pressure_Controller states.h:**

```c
/*
 * states.h
 *
 *  Created on: May 9, 2021
 *      Author: mostafa
 */

#ifndef STATES_H_
#define STATES_H_

#include"Platform_Types.h"
#include"driver.h"


#define STATE_define(_statFUN_) void ST_##_statFUN_()
#define STATE(_statFUN_) ST_##_statFUN_

void getPressureValue(uint32 p);
void setAlarm(uint32 st);

#endif /* STATES_H_ */

```

- **Presure_Sensor PS.c:**

```c
/*
 * PS.c
 *
 *  Created on: May 9, 2021
 *      Author: mostafa
 */

#include "PS.h"


void (*PS_state)();

uint32 PS_pressure = 0;

STATE_define(PS_busy){

    PS_state_id = PS_busy;

    PS_pressure = getPressureVal();

    getPressureValue(PS_pressure);

    PS_state=STATE(PS_busy);
}

```

- **Presure_Sensor PS.h:**

```c
/*
 * PS.h
 *
 *  Created on: May 9, 2021
 *      Author: mostafa
 */

#ifndef PS_H_
#define PS_H_

#include"states.h"

enum{
    PS_busy
}PS_state_id;

STATE_define(PS_busy);


extern void (*PS_state)();

#endif /* PS_H_ */

```

- **Pressure_Control PC.c:**

```c
#include "PC.h"
#include "AA.h"

void (*PC_state)();

uint32 PC_pressure = 0;
uint32 PC_threshold = 20;
enum {
    ON,
    OFF
}PC_alarmstate ;

void getPressureValue(uint32 p){

    PC_pressure = p;

    (PC_pressure > PC_threshold)?(PC_state=STATE(PC_HPdetected)):(PC_state=STATE(PC_waiting));
}

STATE_define(PC_waiting){

    PC_state_id = PC_waiting;
    PC_alarmstate = OFF;
    setAlarm(PC_alarmstate);
    AA_state();
}

STATE_define(PC_HPdetected){

    PC_state_id = PC_HPdetected;
    PC_alarmstate = ON;
    setAlarm(PC_alarmstate);
    AA_state();

    Delay(6000000);

    PC_alarmstate = OFF;
    setAlarm(PC_alarmstate);
    AA_state();

    PC_state = STATE(PC_waiting);
}

```

- **Presure_Control PC.h:**

```c
/*
 * PC.h
 *
 *  Created on: May 9, 2021
 *      Author: mostafa
 */

#ifndef PC_H_
#define PC_H_

#include "states.h"

enum {
    PC_waiting,
    PC_HPdetected
}PC_state_id;

STATE_define(PC_waiting);
STATE_define(PC_HPdetected);

extern void (*PC_state)();

#endif /* PC_H_ */

```

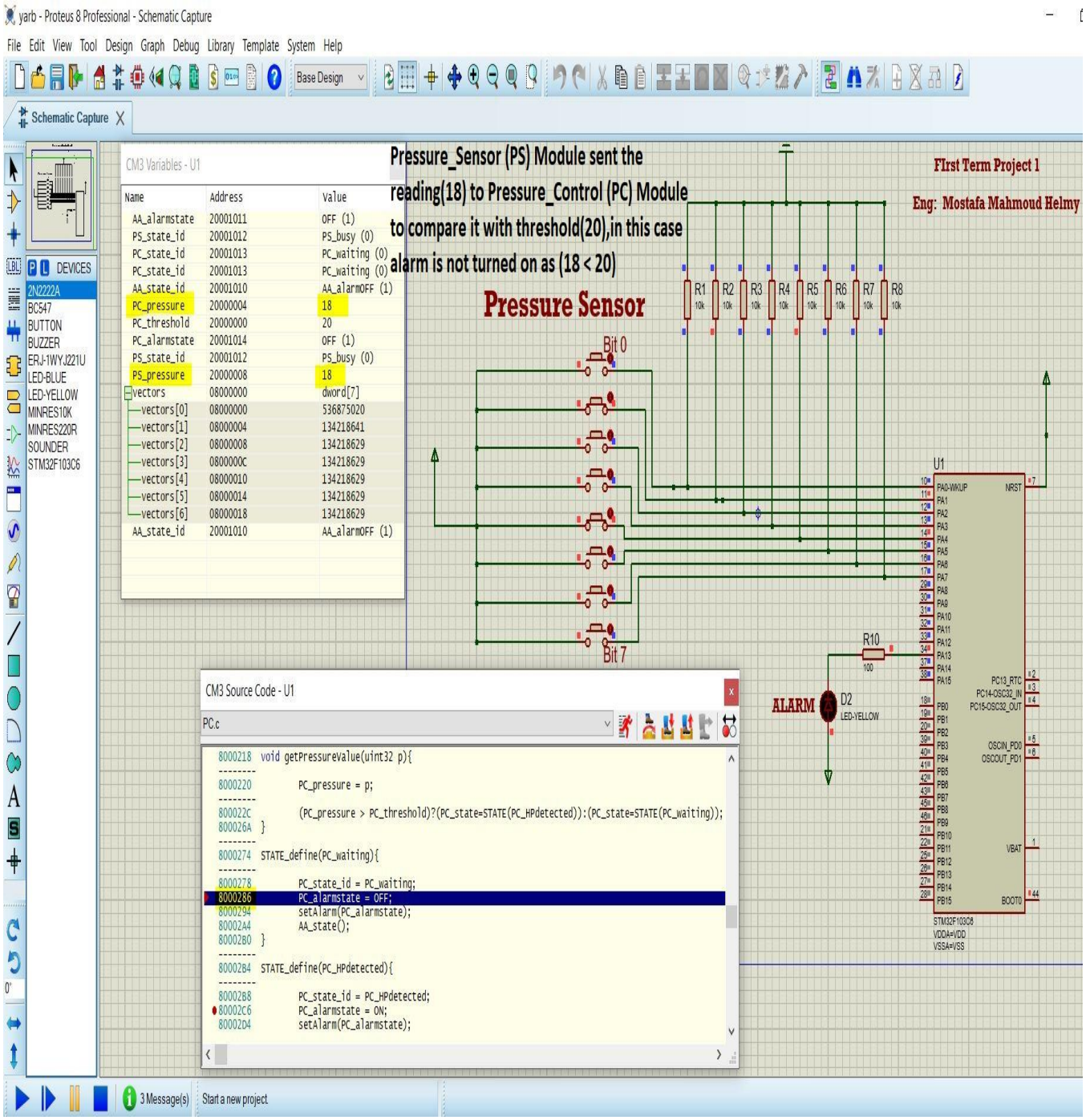
- **Alarm_Actuator AA.c:**

```c
/*
 * AA.c
 *
 *  Created on: May 9, 2021
 *      Author: mostafa
 */
#include"AA.h"

void (*AA_state)();

enum {
    ON,
    OFF
}AA_alarmstate;

void setAlarm(uint32 st){

    AA_alarmstate = st;

    (AA_alarmstate == ON)?(AA_state = STATE(AA_alarmOn)):(AA_state = STATE(AA_alarmOff));

}


STATE_define(AA_alarmOn){

    AA_state_id = AA_alarmON;

    Set_Alarm_actuator(AA_alarmstate);

}

STATE_define(AA_alarmOff){

    AA_state_id = AA_alarmOFF;

    Set_Alarm_actuator(AA_alarmstate);

}

```

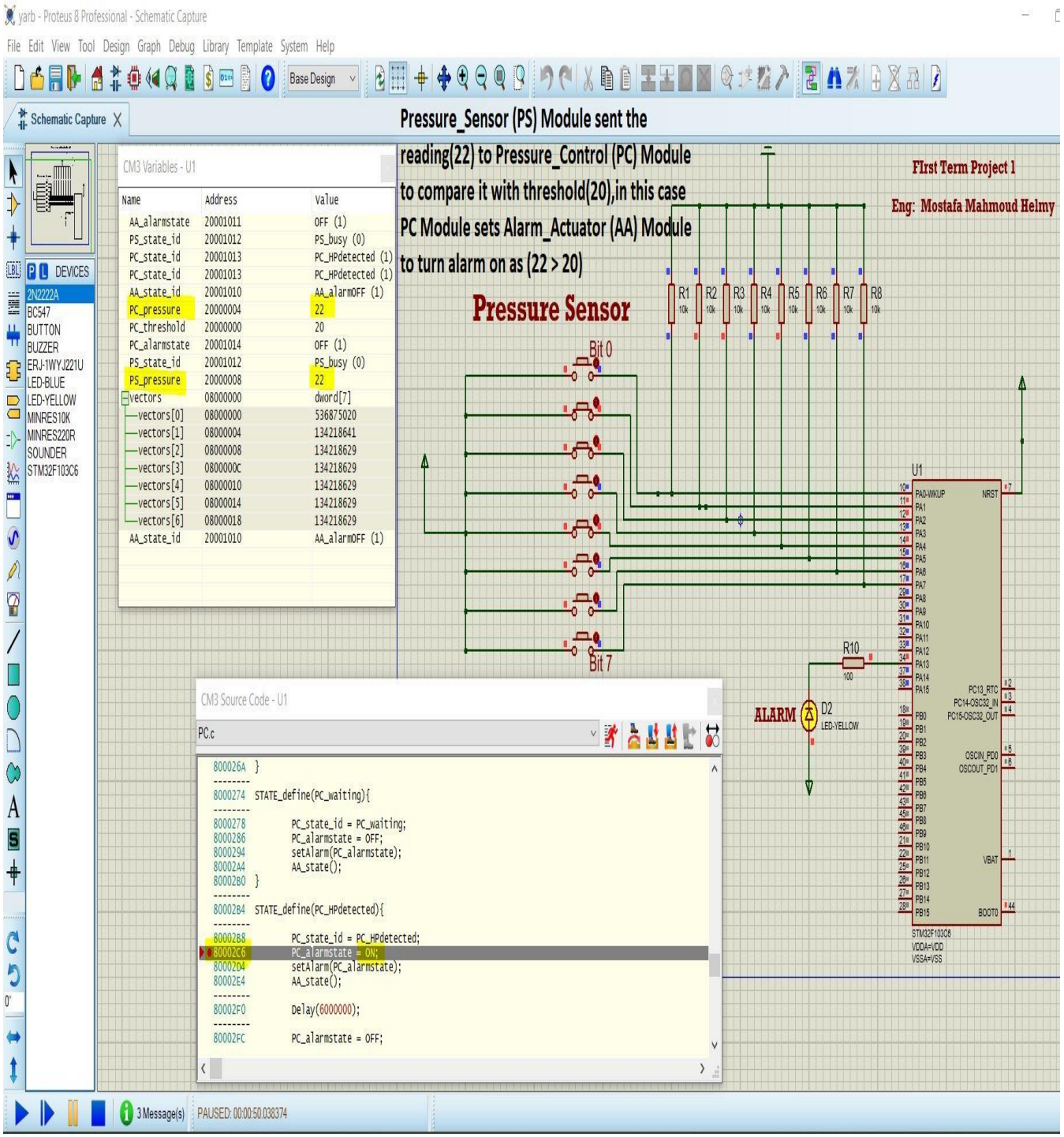
- **Alarm_Actuator AA.h:**

```c
/*
 * AA.h
 *
 *  Created on: May 9, 2021
 *      Author: mostafa
 */


#ifndef AA_H_
#define AA_H_

#include"states.h"

enum{
    AA_alarmON,
    AA_alarmOFF
}AA_state_id;

STATE_define(AA_alarmOn);
STATE_define(AA_alarmOff);

extern void (*AA_state)();
#endif /* AA_H_ */

```

- **Simulation on proteus:**
  - **Case 1:**
    - **Pressure_Sensor (PS) Module sent the reading(18) to Pressure_Control (PC) Module to compare it with threshold(20), in this case alarm is kept off as (18 < 20)**



- **Case 2:**
  - **Pressure_Sensor (PS) Module sent the reading(22) to Pressure_Control (PC) Module to compare it with threshold(20), In this case PC Module sets Alarm_Actuator (AA) Module to turn alarm on for 60 seconds as (22 > 20)**



- **Software analysis:**
  - **Symbols table:**

```
Pressure_Controller.elf: Symbols

2000000c B _E_bss
20000004 D _E_data
08000440 T _E_text
20000004 B _S_bss
20000000 D _S_data
2000100c B _stack_top
20001011 B AA_alarmstate
2000100c B AA_state
20001010 B AA_state_id
08000384 W Bus_Fault_Handler
08000384 T Default_Handler
080000b8 T Delay
080000dc T getPressureVal
08000218 T getPressureValue
08000144 T GPIO_INITIALIZATION
08000384 W Hard_Fault_Handler
080001f4 T main
08000384 W MM_Handler
08000384 W NMI_Handler
20001014 B PC_alarmstate
20000004 B PC_pressure
20001018 B PC_state
20001013 B PC_state_id
20000000 D PC_threshold
20000008 B PS_pressure
2000101c B PS_state
20001012 B PS_state_id
08000390 T Reset_Handler
080000f4 T Set_Alarm_actuator
0800001c T setAlarm
080001c4 T setup
08000094 T ST_AA_alarmOff
08000070 T ST_AA_alarmOn
080002b4 T ST_PC_HPdetected
08000274 T ST_PC_waiting
0800033c T ST_PS_busy
08000384 W Usage_Fault_Handler
08000000 T vectors

```

- **Sections table:**

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

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000440  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000004  20000000  08000440  00010000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          0000101c  20000004  08000444  00010004  2**2
                  ALLOC
  3 .comment      00000011  00000000  00000000  00010004  2**0
                  CONTENTS, READONLY
  4 .ARM.attributes 00000033  00000000  00000000  00010015  2**0
                  CONTENTS, READONLY
```

- **Map file:**

```
Allocating common symbols
Common symbol       size              file

PC_alarmstate       0x1               PC.o
AA_state            0x4               AA.o
PC_state            0x4               PC.o
AA_state_id         0x1               AA.o
PS_state            0x4               PS.o
PS_state_id         0x1               main.o
PC_state_id         0x1               main.o
AA_alarmstate       0x1               AA.o

Memory Configuration

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

```
Linker script and memory map


.text           0x08000000      0x440
 *(.vectors)
 .vectors       0x08000000       0x1c startup.o
                0x08000000                vectors
 *(.text*)
 .text          0x0800001c       0x9c AA.o
                0x0800001c                setAlarm
                0x08000070                ST_AA_alarmOn
                0x08000094                ST_AA_alarmOff
 .text          0x080000b8      0x10c driver.o
                0x080000b8                Delay
                0x080000dc                getPressureVal
                0x080000f4                Set_Alarm_actuator
                0x08000144                GPIO_INITIALIZATION
 .text          0x080001c4       0x54 main.o
                0x080001c4                setup
                0x080001f4                main
 .text          0x08000218      0x124 PC.o
                0x08000218                getPressureValue
                0x08000274                ST_PC_waiting
                0x080002b4                ST_PC_HPdetected
 .text          0x0800033c       0x48 PS.o
                0x0800033c                ST_PS_busy
 .text          0x08000384       0xbc startup.o
                0x08000384                MM_Handler
                0x08000384                Bus_Fault_Handler
                0x08000384                Hard_Fault_Handler
                0x08000384                Default_Handler
                0x08000384                Usage_Fault_Handler
                0x08000384                NMI_Handler
                0x08000390                Reset_Handler
 *(.rodata)
                0x08000440                _E_text = .

```

```
75  .data           0x20000000        0x4 load address 0x08000440
76                  0x20000000                _S_data = .
77   *(.data*)
78   .data          0x20000000        0x0 AA.o
79   .data          0x20000000        0x0 driver.o
80   .data          0x20000000        0x0 main.o
81   .data          0x20000000        0x4 PC.o
82                  0x20000000                PC_threshold
83   .data          0x20000004        0x0 PS.o
84   .data          0x20000004        0x0 startup.o
85                  0x20000004                . = ALIGN (0x4)
86                  0x20000004                _E_data = .
87
88  .igot.plt       0x20000004        0x0 load address 0x08000444
89   .igot.plt      0x00000000        0x0 AA.o
90
91  .bss            0x20000004     0x101c load address 0x08000444
92                  0x20000004                _S_bss = .
93   *(.bss*)
94   .bss           0x20000004        0x0 AA.o
95   .bss           0x20000004        0x0 driver.o
96   .bss           0x20000004        0x0 main.o
97   .bss           0x20000004        0x4 PC.o
98                  0x20000004                PC_pressure
99   .bss           0x20000008        0x4 PS.o
100                 0x20000008                PS_pressure
101  .bss           0x2000000c        0x0 startup.o
102                 0x2000000c                . = ALIGN (0x4)
103                 0x2000000c                _E_bss = .
104                 0x2000100c                . = (. + 0x1000)
105  *fill*         0x2000000c     0x1000
106                 0x2000100c                _stack_top = .
107  COMMON         0x2000100c        0x6 AA.o
108                 0x2000100c                AA_state
109                 0x20001010                AA_state_id
110                 0x20001011                AA_alarmstate
111  COMMON         0x20001012        0x2 main.o
112                 0x20001012                PS_state_id
113                 0x20001013                PC_state_id
114  COMMON         0x20001014        0x8 PC.o
115                 0x20001014                PC_alarmstate
116                 0x20001018                PC_state
117  COMMON         0x2000101c        0x4 PS.o
118                 0x2000101c                PS_state
```