Airplane cabinet pressure detection

Prepared by: Ahmed Khaled Khalaf

- Contents: -
  - Case study
  - Methodology
  - (HW/SW Partitioning)
  - Requirement Diagram
  - System analysis
    - System analysis: Use case diagram
    - System analysis: Activity diagram
    - System analysis: Sequence diagram
  - System design
    - Pressure sensor state diagram
    - Main Block state diagram
    - Alarm manager state diagram
    - Alarm Actuator state diagram
  - A brief look at source files
  - Finding the entry point using readelf utility
  - A brief look at the map file and symbols
  - Debugging and testing using keil
  - Software usage and hardware simulation

- Case study:

Pressure Detection system to inform the crew of the airplane in the cabinet with an alarm when the pressure exceeds 20 bars in that cabinet.

The alarm duration equals 60 seconds. Keep tracking of the measured values.

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

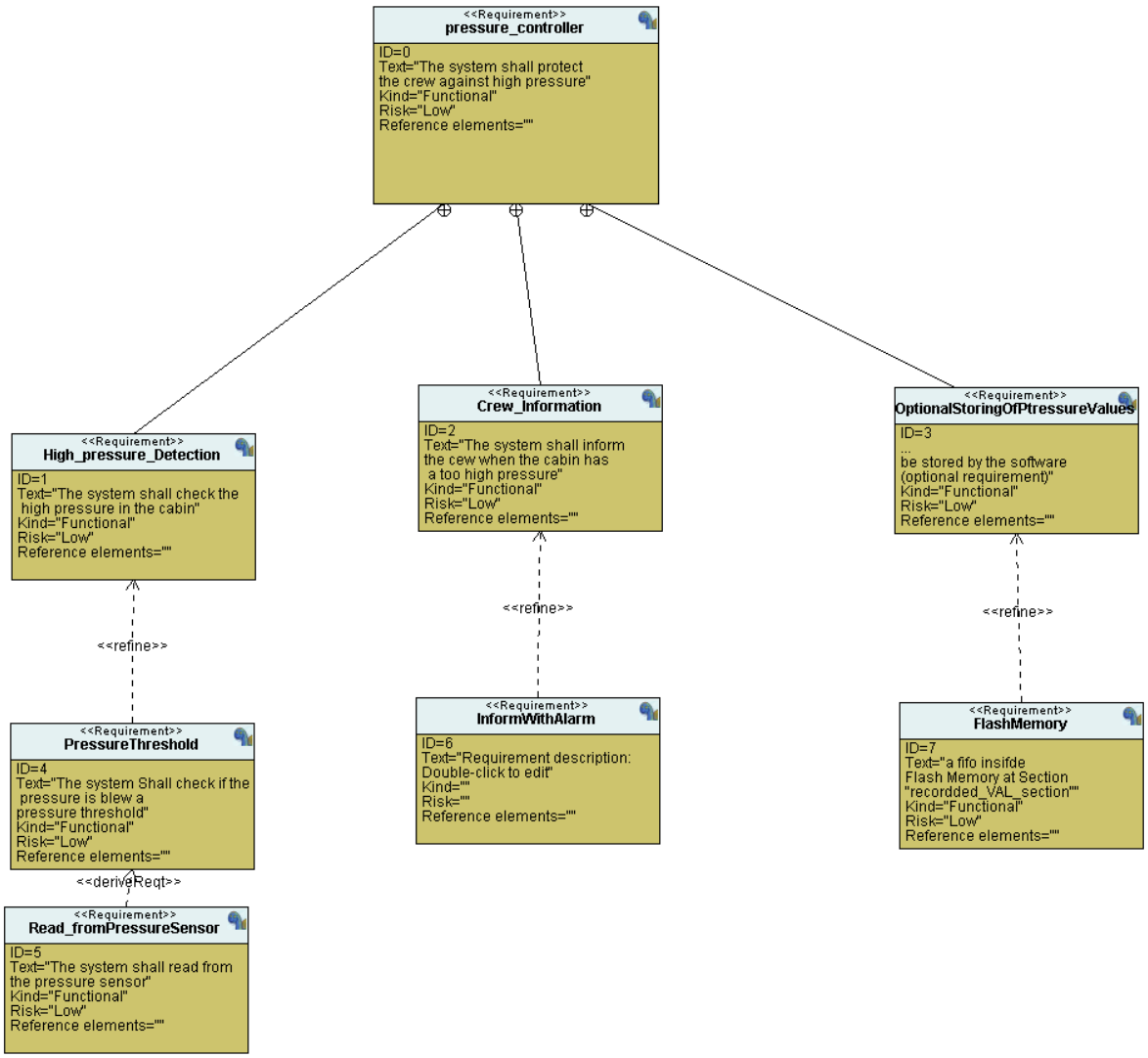
- Methodology:

Since the system has multiple modules and seems to be at the large scale, the system will use a testing-based model like v-model. Every phase in this project will be tested and especially the implementation phase. Each software module will be implemented and unit-tested separately then integrated and integration testing will be performed.

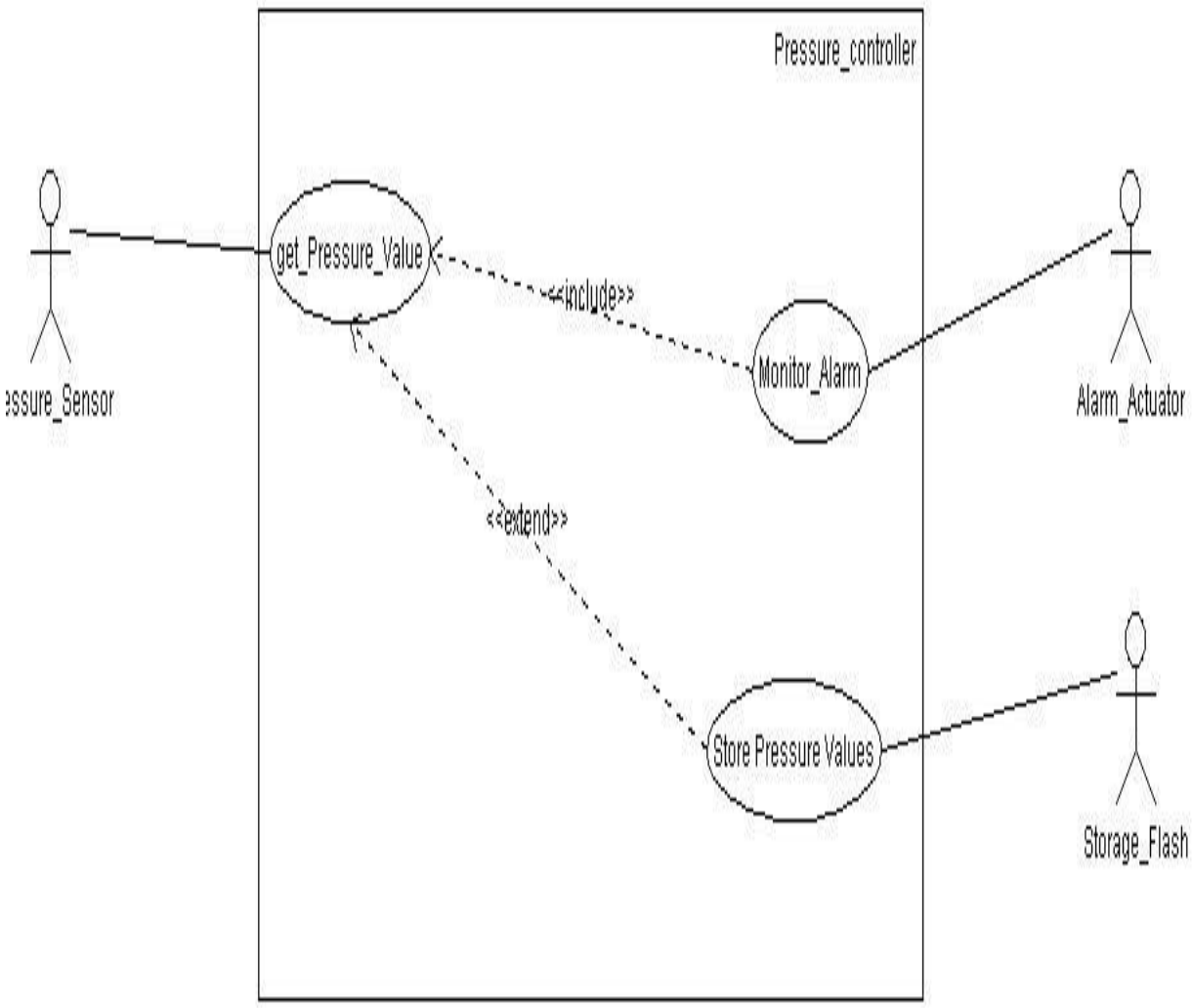
- HW/SW Partitioning:

  For the hardware, we have STM32 microcontroller with a cortex-m3 processor that will be enough for this application, pressure sensor (simulated as 8 bush buttons connected with Pull up resistances), and Alarm actuator (simulated as yellow led and pull up resistance).

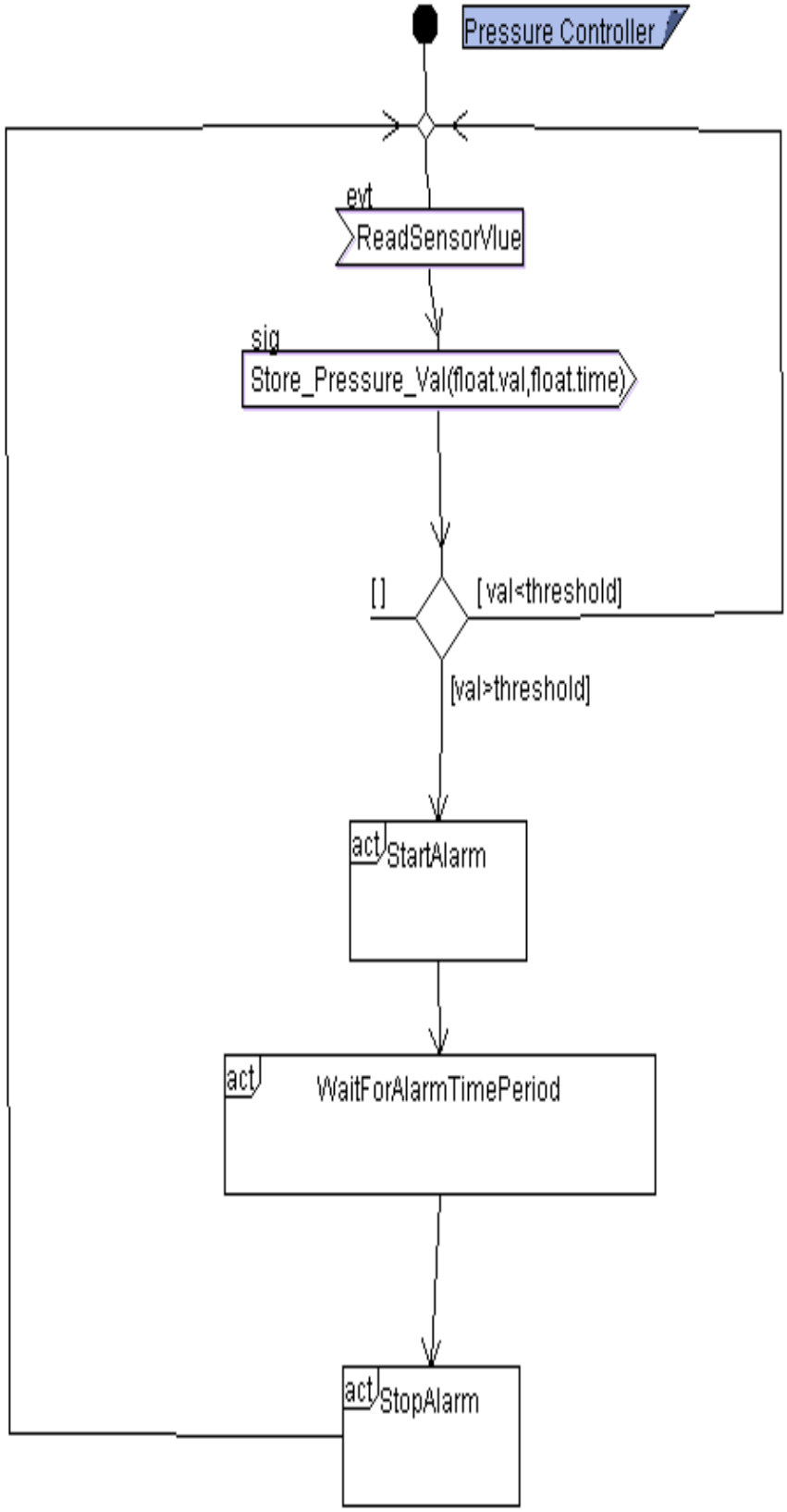
- Requirement Diagram:



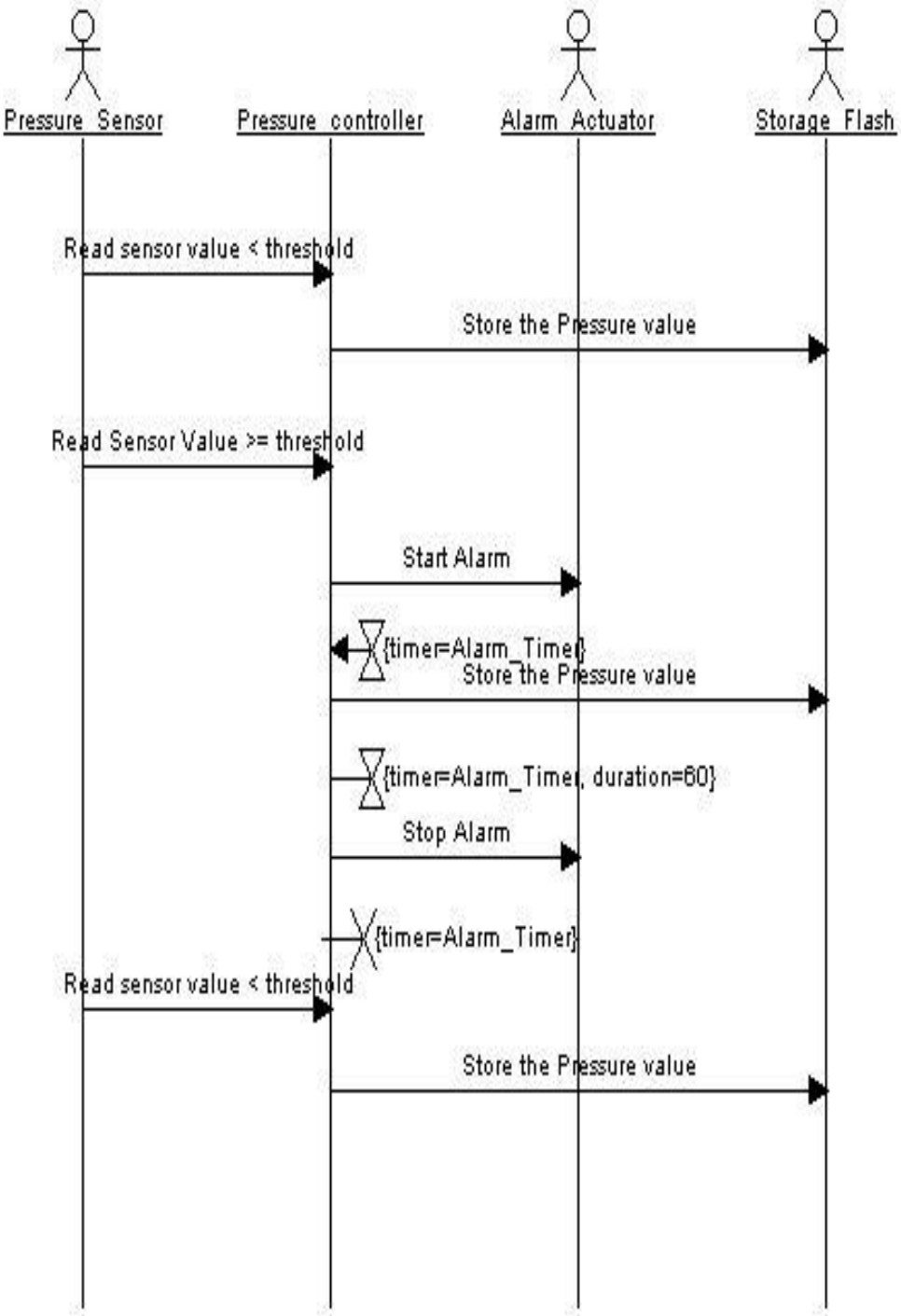
- System analysis:
  - Use case diagram:



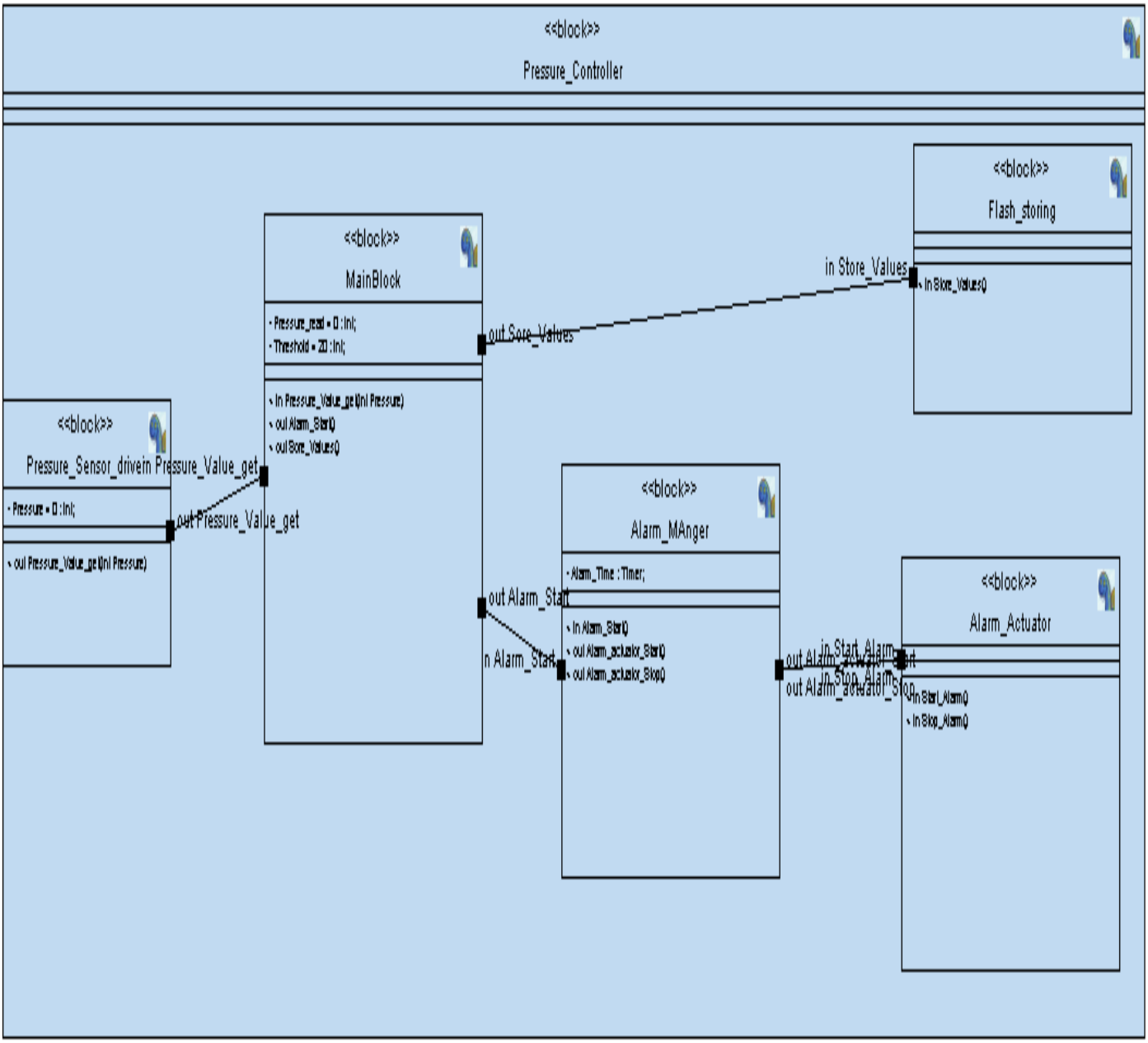
- Activity diagram:



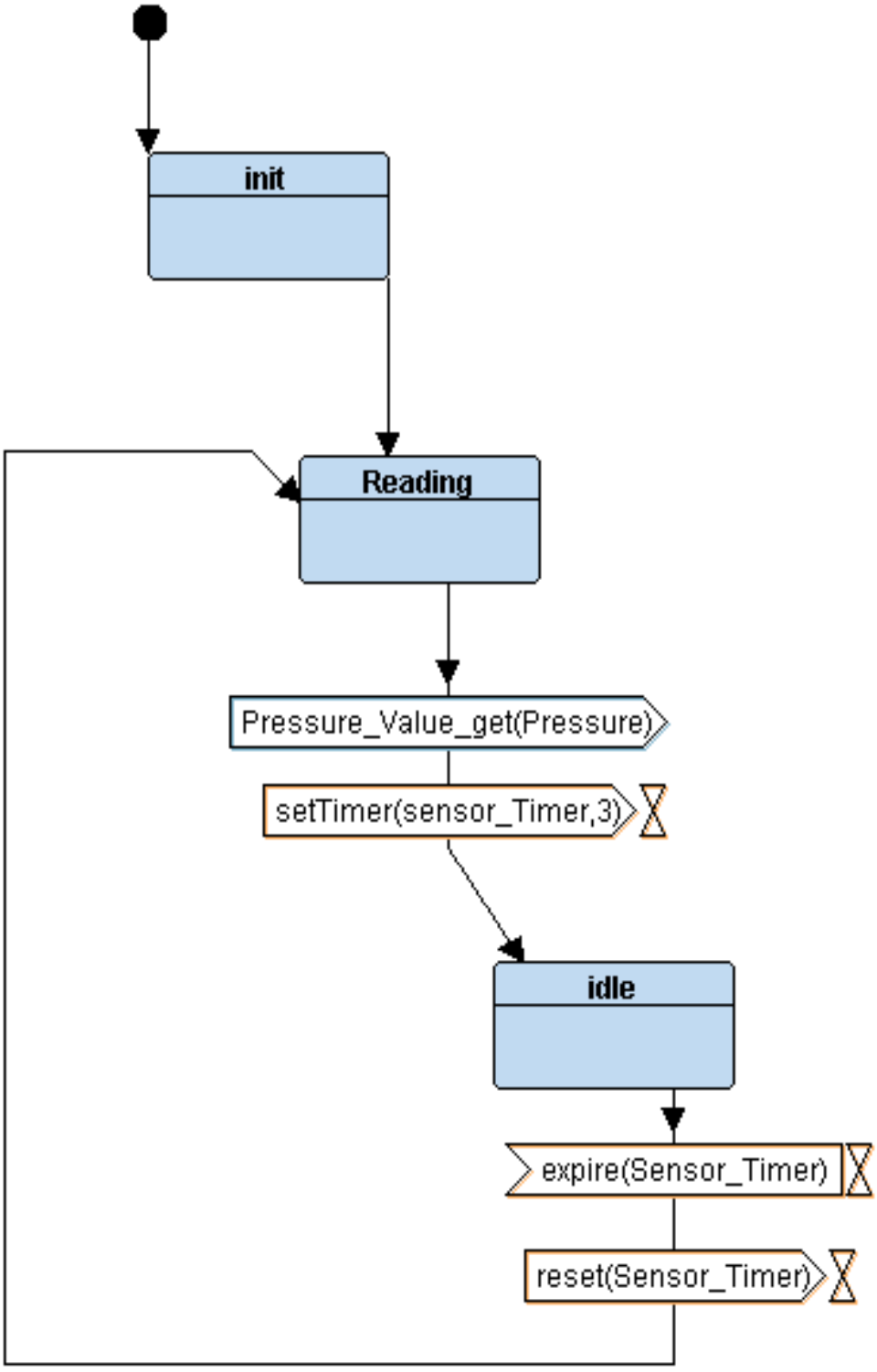
- Sequence diagram:



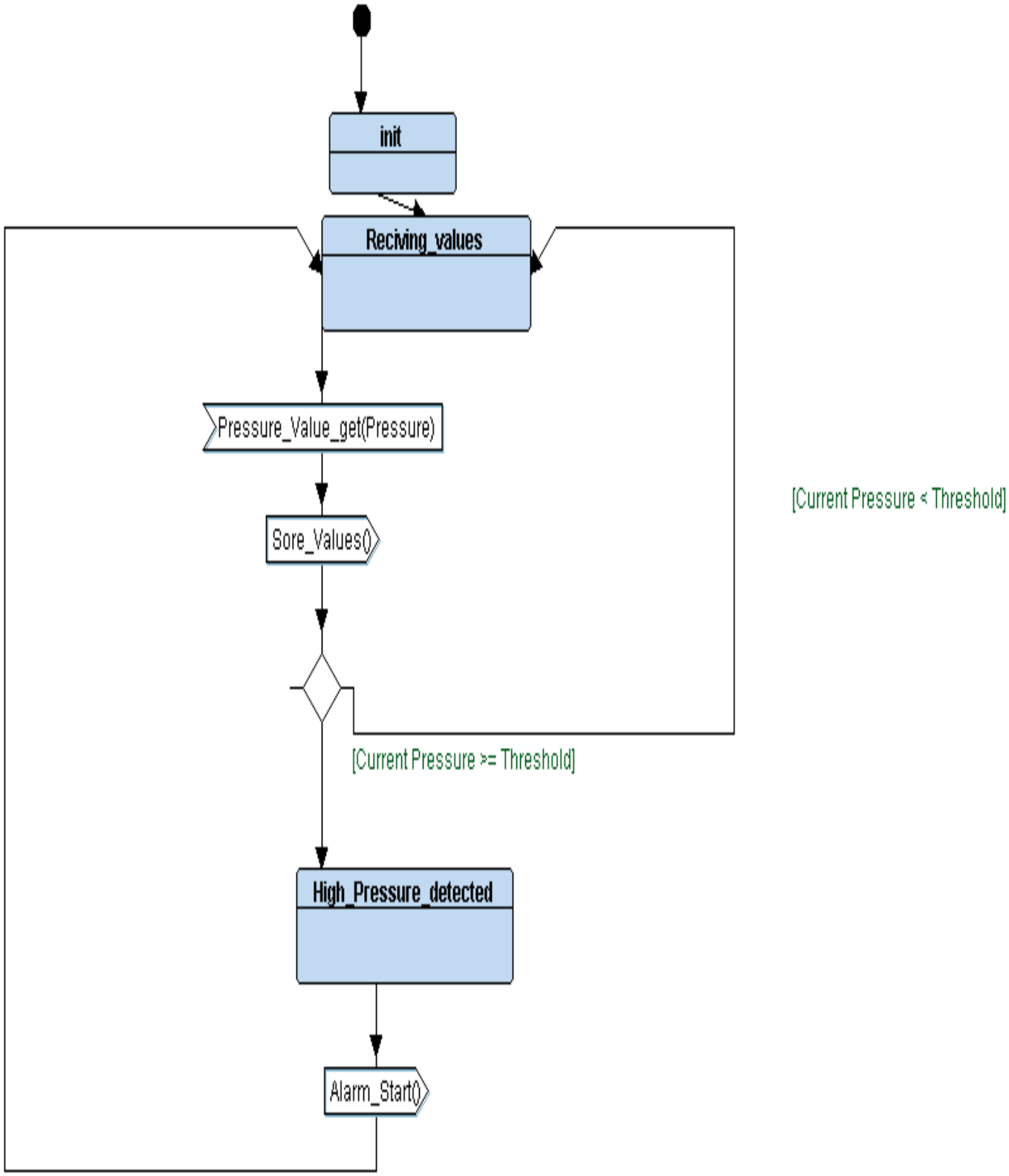
- System design:



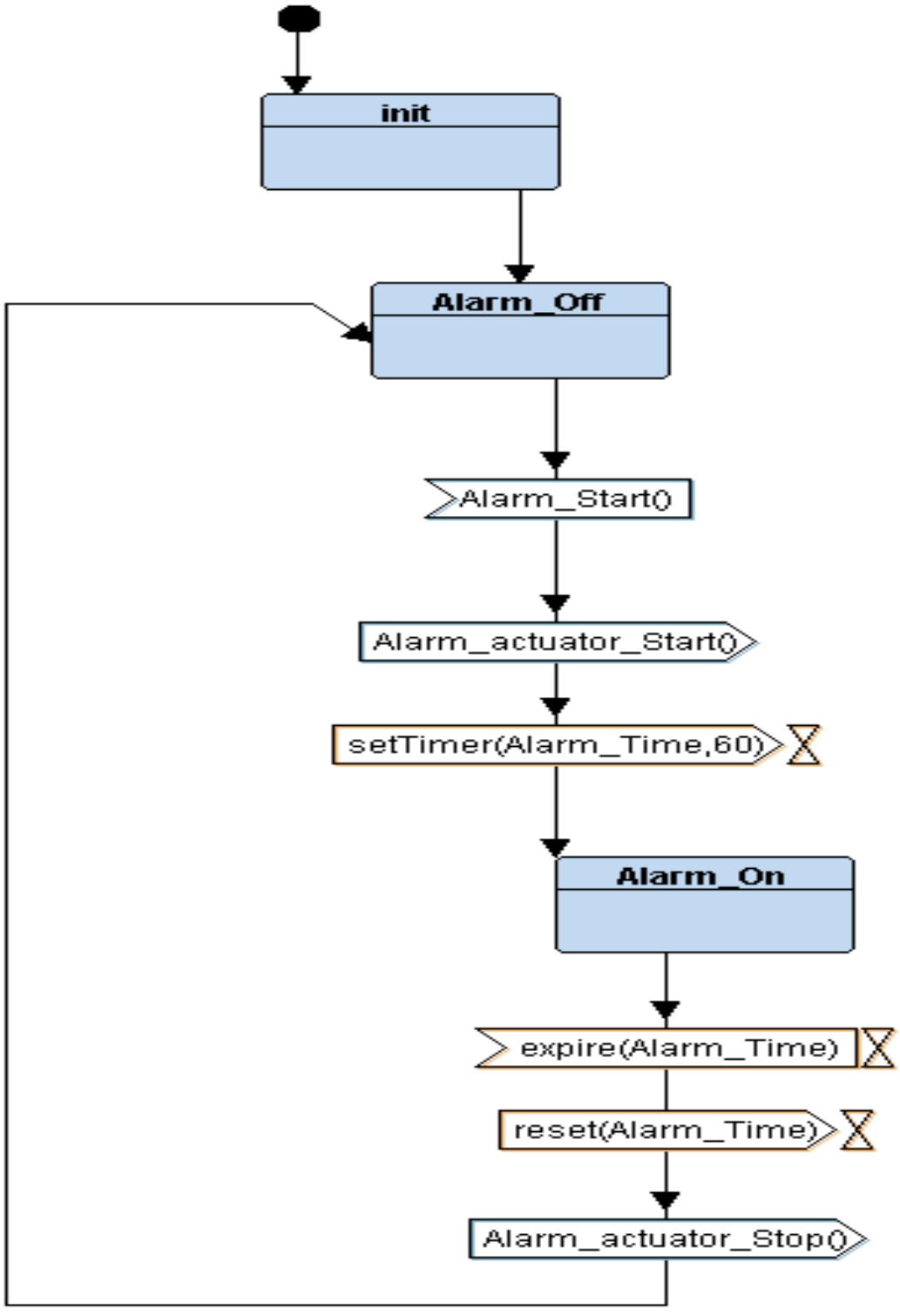
- Pressure state diagram:



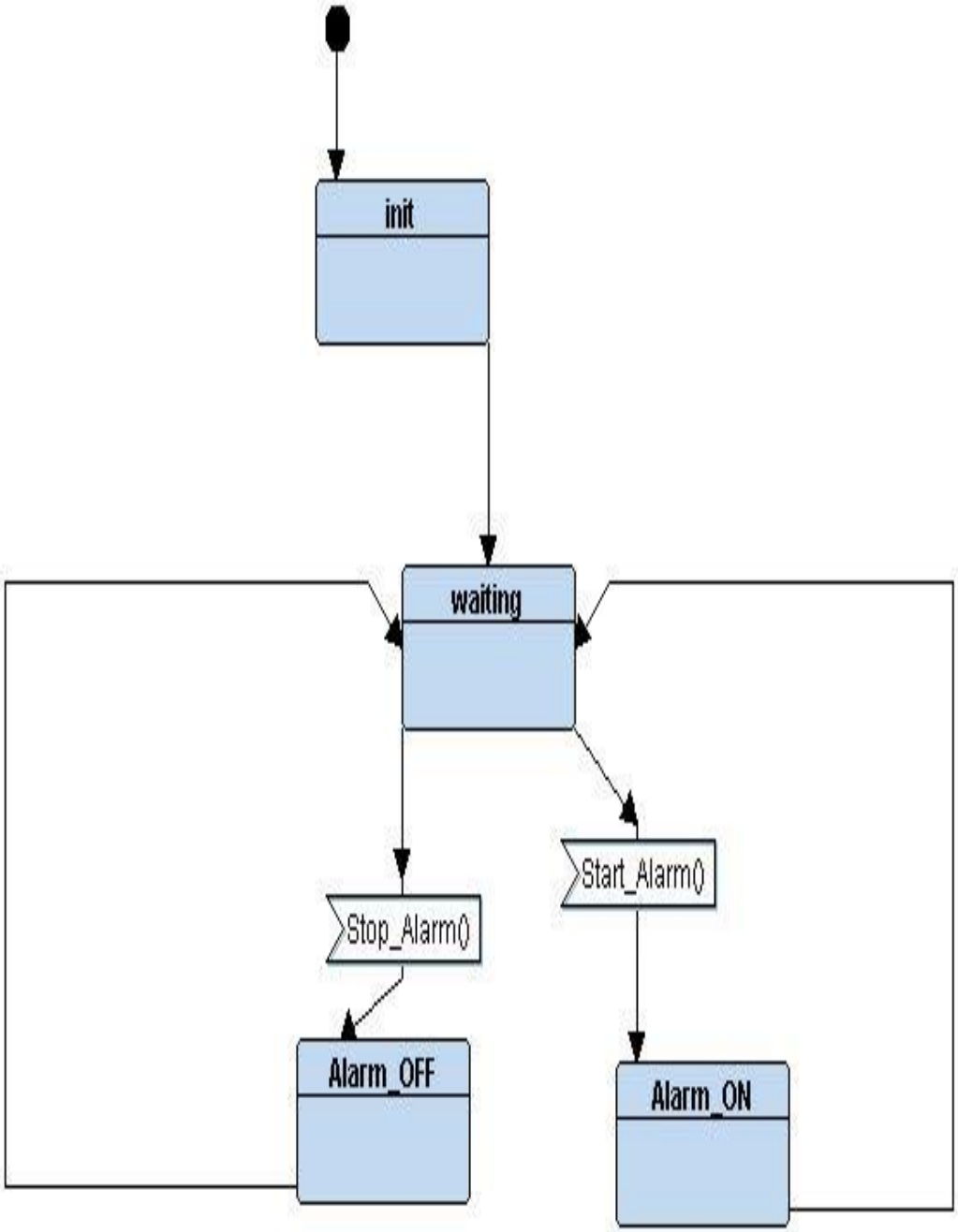
- Main Block state diagram:



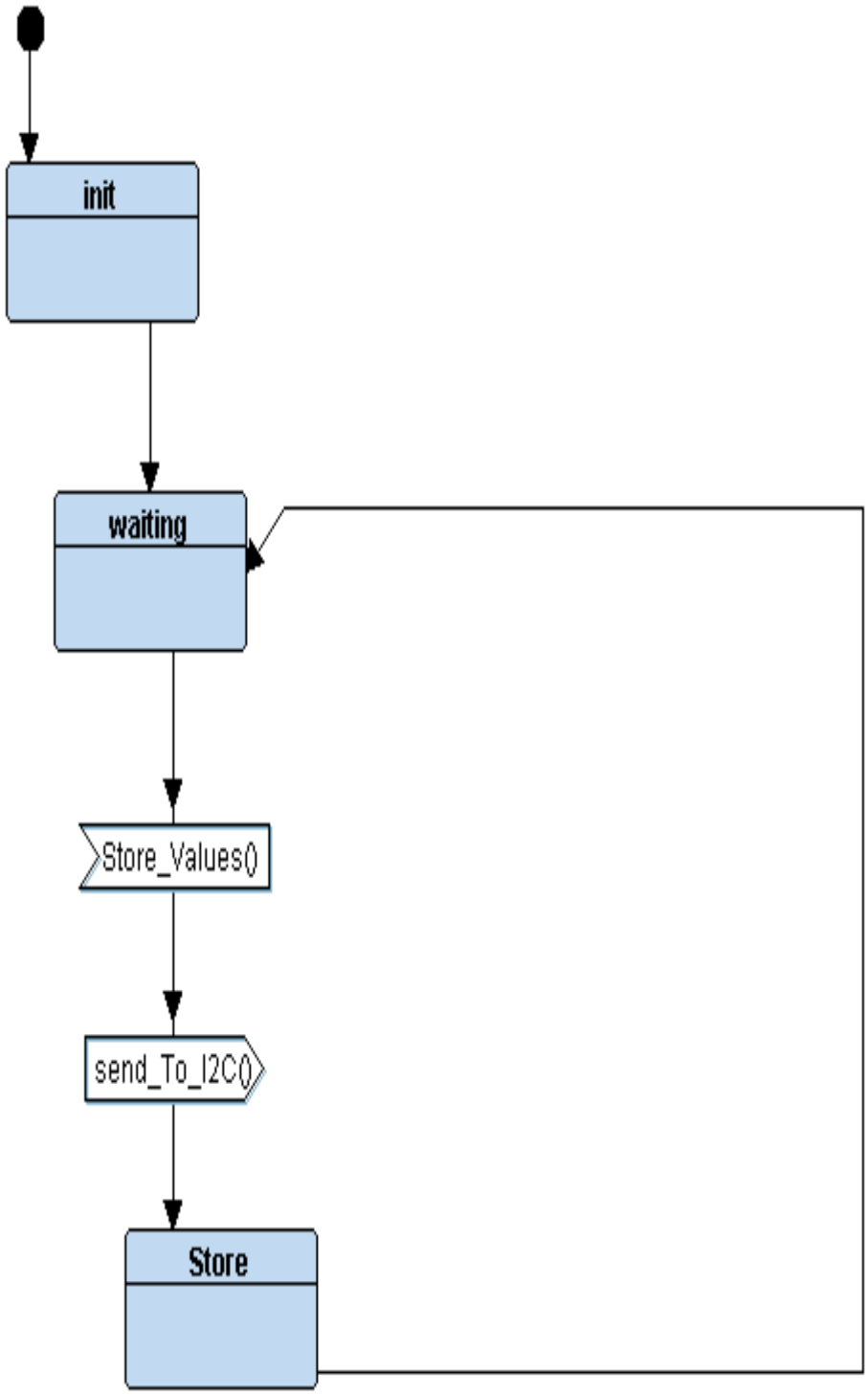
- Alarm Manager state diagram:



- Alarm Actuator state diagram:



- Flash memory store state diagram:



- Main.c

F: > Mstering Embedded systems > Embedded C > unit 5 > FIRST TERM Project 1 > C main.c

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

#include "driver.h"
#include "Main_Block.h"

void Setup(void)
{
    GPIO_INITIALIZATION();
    MB_State = STATE(MB_Init);
    AM_State = STATE(AM_Init);
    Pressure_Sensor_State = STATE(PressureSensor_Init);
    Alarm_Act_State = STATE(Alarm_Act_Init);

}
void main ()
{
    Setup();
    while (1)
    {
        Pressure_Sensor_State();
        MB_State();
        AM_State();
        Alarm_Act_State();
    }

}

```

- State.h

```c
/**
 * @file state.h
 * @author Ahmed khaled khalaf (khalafawy22899@gmail.com)
 * @brief
 * @version 0.1
 * @date 2024-06-22
 *
 * @copyright Copyright (c) 2024
 *
 */

#ifndef _STATE_H_
#define _STATE_H_

//Standered data types file include
#include "STD_DataTypes.h"
#include "driver.h"

//Automatic State Functions declaration
#define STATE_define(_state_Func_) void ST_##_state_Func_()
#define STATE(_state_Func_) ST_##_state_Func_

#endif
```

- driver .h and .c

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

```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;
}
```

- Pressure sensor driver

Pressure_sensor.h > ...

```c
/**
 * @file Pressure_sensor.h
 * @author Ahmed khaled khalaf (khalafawy22899@gmail.com)
 * @brief This file contains all the prototypes and macros needed to Pressure sensor driver implementatio
 * @version 0.1
 * @date 2024-06-21
 *
 * @copyright Copyright (c) 2024
 *
 */


#ifndef  _PRESSURE_SENSOR_H_
#define  _PRESSURE_SENSOR_H_
#include "state.h"

//the Pressure sensor state
enum{

    PressureSensor_Init,
    PressureSensor_Reading,
    PressureSensor_Idle
}Pressure_Sensor_Status_t;

//prototypes of the pressure sensor states function
STATE_define(PressureSensor_Init);
STATE_define(PressureSensor_Reading);
STATE_define(PressureSensor_Idle);
uint32_t Pressure_Sensor_Value_Send(void);

//Pressure Sensor Module Pointer to state functions
void (*Pressure_Sensor_State) ();

#endif
```

Pressure_sensor.c > Pressure_Sensor_Value_Send(void)

```c
/**
 * @file Pressure_sensor.c
 * @author Ahmed khaled khalaf (khalafawy22899@gmail.com)
 * @brief the source file of the pressure sensor driver contains the implementation and APIs
 * @version 0.1
 * @date 2024-06-22
 *
 * @copyright Copyright (c) 2024
 *
 */


#include "Pressure_sensor.h"
static uint32_t Pressure_Value;

/**
 * @brief  The startup function which will initialize the pressure sensor
 * @note   none
 * @retval void
 */


STATE_define(PressureSensor_Init)
{
    //initialize the pressure sensor
    //state define
    Pressure_Sensor_Status_t = PressureSensor_Init;

    //action
    //check event and update status pointer by updating the pointer to the next state
    Pressure_Sensor_State = STATE(PressureSensor_Reading);
}

```

Pressure_sensor.c > Pressure_Sensor_Value_Send(void)

```c
/**
 * @brief  At this function the Pressure sensor enter the state reading and shall store the reading at a variable
 * @param  Pressure_Value is a static float type which will contain the reading of the sensor
 * @note   the physical world has floating quantities more than integers
 * @retval none
 */

STATE_define(PressureSensor_Reading)
{
    //state define
    Pressure_Sensor_Status_t = PressureSensor_Reading;


    //action
    Pressure_Value = getPressureVal();

    //action
    //enter the next state
    Pressure_Sensor_State = STATE(PressureSensor_Idle);

```

Pressure_sensor.c > Pressure_Sensor_Value_Send(void)

```c
}

/**
 * @brief At this function the Pressure sensor shall wait and reurn to the reading state
 * @note   it's a delay function instead of the timer
 * @retval none
 */

STATE_define(PressureSensor_Idle)
{
    //state define
    Pressure_Sensor_Status_t = PressureSensor_Idle;

    //fire a delay
    Delay(1000);


    //check event and update status by updating the pointer to the next state
    Pressure_Sensor_State = STATE(PressureSensor_Reading);
}

/**
 * @brief this function usage is to interface to the upper layer and send the read value of the sensor
 * @note   none
 * @retval pressure sensor value
 */

uint32_t Pressure_Sensor_Value_Send(void)
{
    return Pressure_Value;
}

```

- Main block:

Main_Block.h > ...

```c
/**
 * @file Main_Block.h
 * @author Ahmed khaled khalaf (khalafawy22899@gmail.com)
 * @brief  This file contains all the prototypes and macros needed to Main Block driver implementation
 * @version 0.1
 * @date 2024-06-22
 *
 * @copyright Copyright (c) 2024
 *
 */

#ifndef _MAIN_BLOCK_H_
#define _MAIN_BLOCK_H_
#include "state.h"
#include "alarm_manager.h"
#include "Pressure_sensor.h"

//define states of the man block
enum
{
    MB_Init,
    MB_Reciving_Values,
    MB_HP_detected
}MB_Status_t;

// Declare State Function
STATE_define(MB_Init);
STATE_define(MB_Reciving_Values);
STATE_define(MB_HP_detected);
uint8_t HP_detected_signal(void);
// State Pointer to Function
void (* MB_State)() ;

#endif
```

Main_Block.c > ...

```c
/**
 * @file Main_Block.c
 * @author Ahmed khaled khalaf (khalafawy22899@gmail.com)
 * @brief the source file of the Main Block driver contains the implementation and APIs
 * @version 0.1
 * @date 2024-06-22
 *
 * @copyright Copyright (c) 2024
 *
 */

#include "Main_Block.h"

//define a variable to recive the current pressure value
static uint32_t CP_Value;
static uint8_t P_Threshold = 20;

/**
 * @brief  initialize the Main block
 * @note   none
 * @retval none
 */

STATE_define(MB_Init)
{
    //state define
    //initialize the Main Block
    MB_Status_t = MB_Init;

    //state action
    MB_State = STATE(MB_Reciving_Values);

}
```

Main_Block.c > ...

```c
/**
 * @brief  Reciving values form the Pressure sensor
 * @note   none
 * @retval none
 */

STATE_define(MB_Reciving_Values)
{
    //state define
    MB_Status_t = MB_Reciving_Values;

    //recive the current pressure value
    CP_Value = Pressure_Sensor_Value_Send();

    //check if the pressure value high or low
    if (CP_Value >= P_Threshold )
    {
        MB_State = STATE(MB_HP_detected);

    }
    MB_State = STATE(MB_Reciving_Values);

}

/**
 * @brief  the main block detected a high pressure signal so it will send a signal to the alarm manager
 *         to tigger the alarm actuator for 60 sec
 * @note   none
 * @retval none
 */
```

Main_Block.c > ...

```c
/**
 * @brief  the main block detected a high pressure signal so it will send a signal to the alarm manager
 *         to tigger the alarm actuator for 60 sec
 * @note   none
 * @retval none
 */

STATE_define(MB_HP_detected)
{
    //define state
    MB_Status_t = MB_HP_detected;

    //Actiorn

    HP_detected_signal();

    MB_State = STATE(MB_Reciving_Values);

}

/**
 * @brief  sending a signal to the alarm manger
 * @note   none
 * @retval True
 */

uint8_t HP_detected_signal(void)
{


    return (CP_Value > P_Threshold);


}
```

- Alarm Manager:

```c
/**
 * @brief  at this function the hogh pressure signal is detected and it will send a signal to trigger the alarm actuator
 * @note   none
 * @retval none
 */

STATE_define(AM_Alarm_ON)
{
    //state define
    AM_Status_t = AM_Alarm_ON;

    //action
    //send alarm actuator start signal
    AA_Start_Alarm();

    AM_State = STATE(AM_Alarm_OFF);


}
```

```c
/**
 * @file alarm_manager.c
 * @author Ahmed khaled khalaf (khalafawy22899@gmail.com)
 * @brief the source file of the alarm manager driver contains the implementation and APIs
 * @version 0.1
 * @date 2024-06-22
 *
 * @copyright Copyright (c) 2024
 *
 */


#include "alarm_manager.h"
#include "alarm_actuator.h"
#include "Main_Block.h"


/**
 * @brief  initialize the alarm actuator
 * @note   none
 * @retval none
 */
STATE_define(AM_Init)
{
    //initialize the Alarm manager
    //state define
    AM_Status_t = AM_Init;

    //action
    AM_State = STATE(AM_Alarm_OFF);
}
```

```c
C alarm_manager.c > STATE_define(AM_Alarm_OFF)
/**
 * @brief  At this function the alarm manger shall wait and the alarm actuatutor remain off
 * @note   it is the idle state of the alarm manager
 * @retval none
 */

STATE_define(AM_Alarm_OFF)
{
    //define the state
    AM_Status_t = AM_Alarm_OFF;
    Delay(50000);
    //send alarm actuator stop signal
    AA_Stop_Alarm();

    //checking whether the high pressure detected or not
    if (HP_detected_signal() == TRUE)
    {
        //the high pressure signal is detected and update status by updating the pointer to the next state alarm on
       AM_State = STATE(AM_Alarm_ON);
    }
    else
    {
        //the high pressure signal is not detected and update status by updating the pointer to the next state alarm off
        //remain at the state
        AM_State = STATE(AM_Alarm_OFF);
    }

    //sending alarm actuator off signal

}
```

```c
C alarm_manager.c > STATE_define(AM_Alarm_OFF)

/**
 * @brief  at this function the hogh pressure signal is detected and it will send a signal to trigger the alarm actuator
 * @note   none
 * @retval none
 */

STATE_define(AM_Alarm_ON)
{
    //state define
    AM_Status_t = AM_Alarm_ON;

    //action
    //send alarm actuator start signal
    AA_Start_Alarm();

    AM_State = STATE(AM_Alarm_OFF);


}
```

- Alarm actuator:

alarm_actuator.h

```c
/**
 * @file alarm_actuator.h
 * @author Ahmed khaled khalaf (khalafawy22899@gmail.com)
 * @brief  This file contains all the prototypes and macros needed to Alarm actuator driver implementation
 * @version 0.1
 * @date 2024-06-22
 *
 * @copyright Copyright (c) 2024
 *
 */
#ifndef _ALARM_ACTUATOR_H
#define _ALARM_ACTUATOR_H
#include "state.h"
// Define Status
enum {
        Alarm_Act_Init,
        Alarm_Act_Waiting,
        Alarm_Act_ON,
        Alarm_Act_OFF
  }Alarm_Act_Status_t ;

// Declare State Fuctions
STATE_define(Alarm_Act_Init);
STATE_define(Alarm_Act_ON);
STATE_define(Alarm_Act_OFF);
STATE_define(Alarm_Act_Waiting);
//functions to recive the signal of the alarm actuator
extern void AA_Start_Alarm(void);
extern void AA_Stop_Alarm(void);
// State Pointer to Function
void (*Alarm_Act_State)() ;

#endif
```

alarm_manager.c > STATE_define(AM_Alarm_OFF)

```c
/**
 * @file alarm_manager.c
 * @author Ahmed khaled khalaf (khalafawy22899@gmail.com)
 * @brief the source file of the alarm manager driver contains the implementation and APIs
 * @version 0.1
 * @date 2024-06-22
 *
 * @copyright Copyright (c) 2024
 *
 */


#include "alarm_manager.h"
#include "alarm_actuator.h"
#include "Main_Block.h"


/**
 * @brief  initialize the alarm actuator
 * @note   none
 * @retval none
 */
STATE_define(AM_Init)
{
    //initialize the Alarm manager
    //state define
    AM_Status_t = AM_Init;

    //action
    AM_State = STATE(AM_Alarm_OFF);
}
```

```c
C alarm_manager.c > STATE_define(AM_Alarm_OFF)
23  STATE_define(AM_Init)
31  }
32  /**
33   * @brief  At this function the alarm manger shall wait and the alarm actuatutor remain off
34   * @note   it is the idle state of the alarm manager
35   * @retval none
36   */
37
38  STATE_define(AM_Alarm_OFF)
39  {
40      //define the state
41      AM_Status_t = AM_Alarm_OFF;
42      Delay(50000);
43      //send alarm actuator stop signal
44      AA_Stop_Alarm();
45
46      //checking whether the high pressure detected or not
47      if (HP_detected_signal() == TRUE)
48      {
49          //the high pressure signal is detected and update status by updating the pointer to the next state alarm on
50         AM_State = STATE(AM_Alarm_ON);
51      }
52      else
53      {
54          //the high pressure signal is not detected and update status by updating the pointer to the next state alarm off
55          //remain at the state
56          AM_State = STATE(AM_Alarm_OFF);
57      }
58
59      //sending alarm actuator off signal
60
61  }
62
```

```c
C alarm_manager.c > STATE_define(AM_Alarm_OFF)
64
65  /**
66   * @brief  at this function the hogh pressure signal is detected and it will send a signal to trigger the alarm actuator
67   * @note   none
68   * @retval none
69   */
70
71  STATE_define(AM_Alarm_ON)
72  {
73      //state define
74      AM_Status_t = AM_Alarm_ON;
75
76      //action
77      //send alarm actuator start signal
78      AA_Start_Alarm();
79
80      AM_State = STATE(AM_Alarm_OFF);
81
82
83  }
```

- Startup.c:

```c
/**
 * @file startup.c
 * @author your name (you@domain.com)
 * @brief
 * @version 0.1
 * @date 2024-06-18
 *
 * @copyright Copyright (c) 2024
 *
 */

#include "STD_DataTypes.h"

void Reset_Handler();
extern int main(void);

void Defult_Handler(void)
{
    Reset_Handler();
}

void NMI_Handler () __attribute__((weak, alias("Defult_Handler")));;
void H_fault_Handler () __attribute__((weak, alias("Defult_Handler")));;

//reserve stack size

static uint_t32 Stack_top[256]; //size = 256*4=1024bytes
//=======================================================

void (* G_P_Fn_Vector[])() __attribute__((section (".vectors"))) =
{
    (void (*)())   ((uint_t32)Stack_top + sizeof(Stack_top)),
    &Reset_Handler,
    &NMI_Handler,
    &H_fault_Handler
};
```

```c
//============================

extern uint_t32 _E_text ;
extern uint_t32 _S_DATA ;
extern uint_t32 _E_DATA ;
extern uint_t32 _S_bss  ;
extern uint_t32 _E_bss  ;


void Reset_Handler()
{
    /*copying the data from flash to RAM*/
    uint_t32 DATA_Size = (uint_t8 *) &_E_DATA - (uint_t8 *) &_S_DATA;
    uint_t8* P_SRC = (uint_t8*) &_E_text;
    uint_t8* P_dst = (uint_t8*) &_S_DATA;

    for (uint_t32 i = 0; i < DATA_Size; i++)
    {
        *((uint_t8*)P_dst++) = *((uint_t8*)P_SRC++);

    }

    // init .bss section in SRAM = 0

    uint_t32 bss_Size = (uint_t8 *)&_E_bss - (uint_t8 *)&_S_bss;
    P_dst = (uint_t8 *) &_S_bss;

        for (int i = 0; i < bss_Size; i++)
        {
            *((uint_t8 *)P_dst++) = (uint_t8) 0;
        }
        main();


}
```

- Linker script:

linkerscript.ld

```
/* linker script CortexM3
Auther: Ahmed Khaled Khalaf
*/


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*)
        . = ALIGN(4);
        _E_DATA = .;

    }SRAM AT> FLASH

    .bss : {
        _S_bss = .;
        *(.bss*)
        _E_bss = .;
        . = ALIGN(4);
        . = . + 0x1000;
        Stack_top = .;

    }>SRAM

```

- Symbol table:

```
MINGW32:/f/Mstering Embedded systems/Embedded C/unit 5/FIRST TERM Project 1
Eng Ahmed khaled@DESKTOP-PQD68QP MINGW32 /f/Mstering Embedded systems/Embedded C
/unit 5/FIRST TERM Project 1
$ arm-none-eabi-nm.exe Pressure_Controller_project.elf
2000040c B _E_bss
20000004 D _E_DATA
08000478 T _E_text
20000004 B _S_bss
20000000 D _S_DATA
2000140c B _stack_top
080001e8 T AA_Start_Alarm
08000204 T AA_Stop_Alarm
20001418 B Alarm_Act_State
2000141c B Alarm_Act_Status_t
20001420 B AM_State
20001424 B AM_Status_t
20000004 b CP_Value
080003e8 T Defult_Handler
080002ac T Delay
08000000 T G_P_Fn_Vector
080002cc T getPressureVal
08000320 T GPIO_INITIALIZATION
080003e8 W H_fault_Handler
080000a4 T HP_detected_signal
080003b4 T main
2000140c B MB_State
20001425 B MB_Status_t
080003e8 W NMI_Handler
20000000 d P_Threshold
20001414 B Pressure_Sensor_State
20001410 B Pressure_Sensor_Status_t
08000148 T Pressure_Sensor_Value_Send
20000008 b Pressure_Value
080003f4 T Reset_Handler
080002e4 T Set_Alarm_actuator
08000370 T Setup
0800015c T ST_Alarm_Act_Init
080001c0 T ST_Alarm_Act_OFF
08000198 T ST_Alarm_Act_ON
08000180 T ST_Alarm_Act_Waiting
08000244 T ST_AM_Alarm_OFF
08000288 T ST_AM_Alarm_ON
08000220 T ST_AM_Init
08000080 T ST_MB_HP_detected
08000010 T ST_MB_Init
08000034 T ST_MB_Reciving_Values
08000120 T ST_PressureSensor_Idle
080000cc T ST_PressureSensor_Init
080000f0 T ST_PressureSensor_Reading
2000000c b Stack_top
```

- Map file:

Pressure_Controller_project_map.map - Notepad

File Edit Format View Help

```
Allocating common symbols
Common symbol       size              file

MB_State            0x4               Main_Block.o
Pressure_Sensor_Status_t
                    0x1               Main_Block.o
Pressure_Sensor_State
                    0x4               Main_Block.o
Alarm_Act_State     0x4               Main_Block.o
Alarm_Act_Status_t  0x1               Main_Block.o
AM_State            0x4               Main_Block.o
AM_Status_t         0x1               Main_Block.o
MB_Status_t         0x1               Main_Block.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      0x478
 *(.vectors*)
 .vectors       0x08000000       0x10 startup.o
                0x08000000                G_P_Fn_Vector
 *(.text*)
 .text          0x08000010       0xbc Main_Block.o
                0x08000010                ST_MB_Init
                0x08000034                ST_MB_Reciving_Values
                0x08000080                ST_MB_HP_detected
                0x080000a4                HP_detected_signal
 .text          0x080000cc       0x90 Pressure_sensor.o
                0x080000cc                ST_PressureSensor_Init
                0x080000f0                ST_PressureSensor_Reading
```

Pressure_Controller_project_map.map - Notepad

File Edit Format View Help

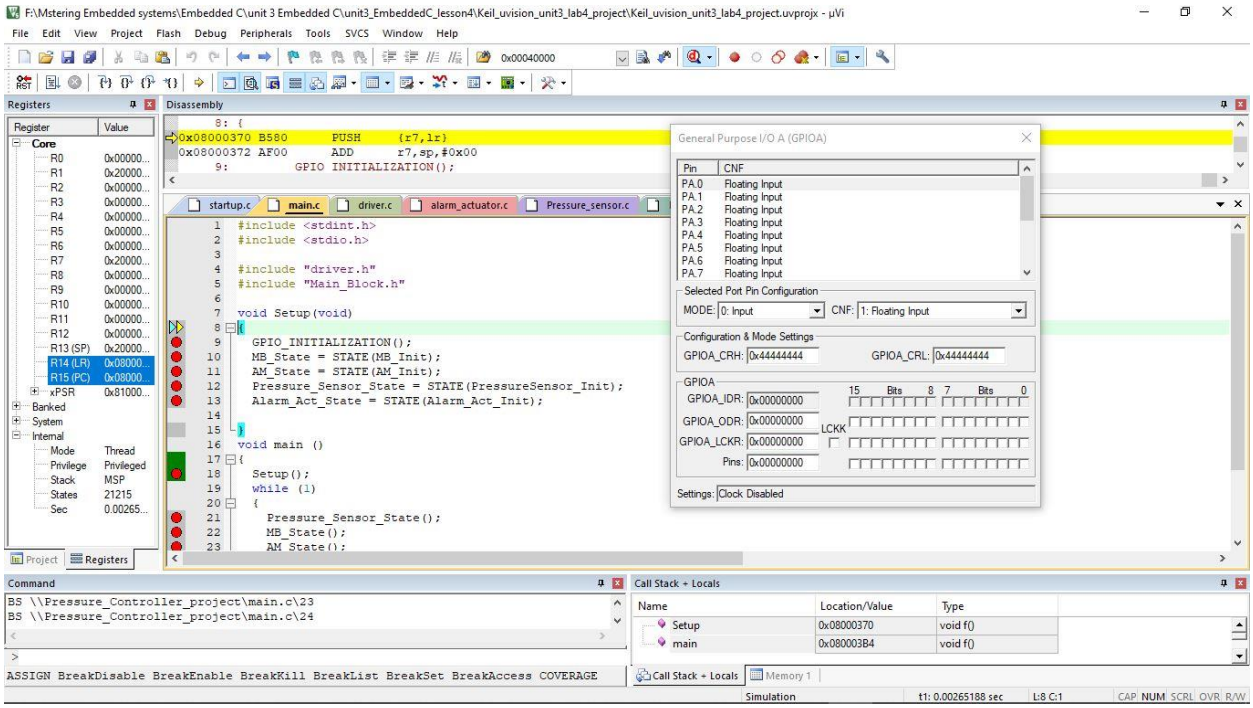
```
                0x08000180                ST_Alarm_Act_Waiting
                0x08000198                ST_Alarm_Act_ON
                0x080001c0                ST_Alarm_Act_OFF
                0x080001e8                AA_Start_Alarm
                0x08000204                AA_Stop_Alarm
 .text          0x08000220       0x8c alarm_manager.o
                0x08000220                ST_AM_Init
                0x08000244                ST_AM_Alarm_OFF
                0x08000288                ST_AM_Alarm_ON
 .text          0x080002ac       0xc4 driver.o
                0x080002ac                Delay
                0x080002cc                getPressureVal
                0x080002e4                Set_Alarm_actuator
                0x08000320                GPIO_INITIALIZATION
 .text          0x08000370       0x78 main.o
                0x08000370                Setup
                0x080003b4                main
 .text          0x080003e8       0x90 startup.o
                0x080003e8                Defult_Handler
                0x080003e8                H_fault_Handler
                0x080003e8                NMI_Handler
                0x080003f4                Reset_Handler
 *(.rodata*)
                0x08000478                _E_text = .

.glue_7         0x08000478        0x0
 .glue_7        0x08000478        0x0 linker stubs

.glue_7t        0x08000478        0x0
 .glue_7t       0x08000478        0x0 linker stubs

.vfp11_veneer   0x08000478        0x0
 .vfp11_veneer  0x08000478        0x0 linker stubs

.v4_bx          0x08000478        0x0
 .v4_bx         0x08000478        0x0 linker stubs

.iplt           0x08000478        0x0
```

Pressure_Controller_project_map.map - Notepad

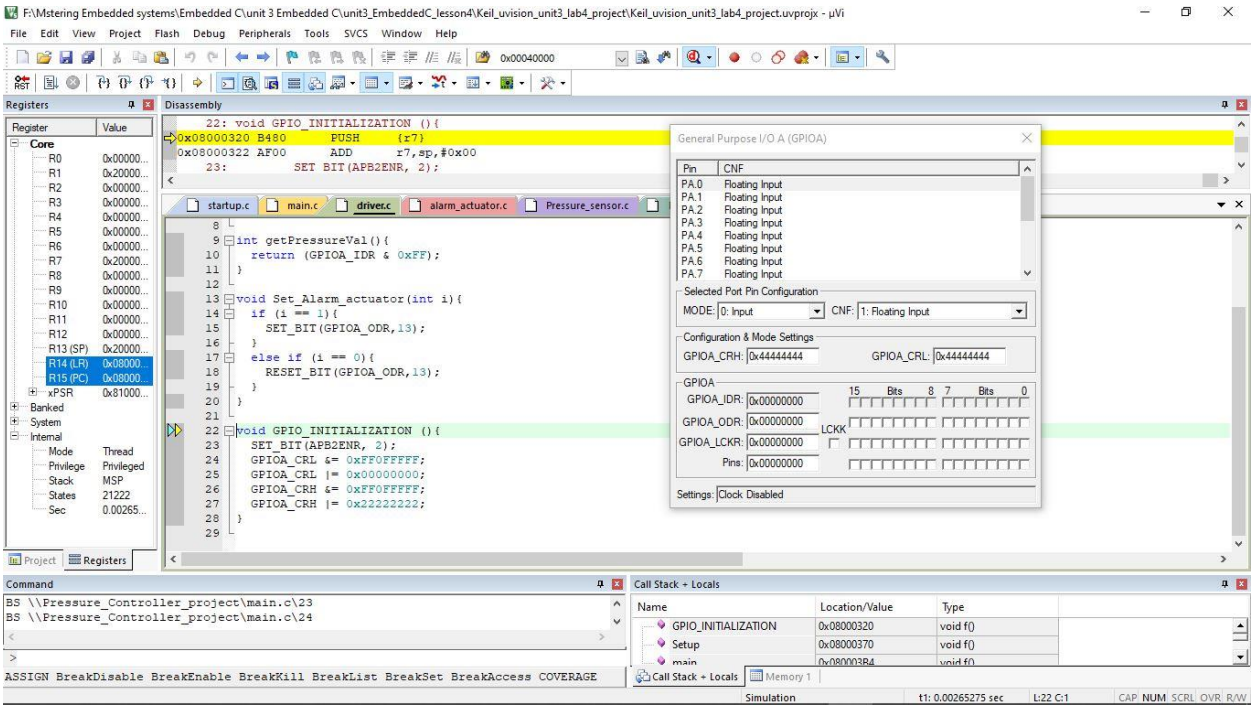
File Edit Format View Help

```
.rel.dyn        0x08000478        0x0
 .rel.iplt      0x08000478        0x0 Main_Block.o

.data           0x20000000        0x4 load address 0x08000478
                0x20000000                _S_DATA = .
 *(.data*)
 .data          0x20000000        0x1 Main_Block.o
 .data          0x20000001        0x0 Pressure_sensor.o
 .data          0x20000001        0x0 alarm_actuator.o
 .data          0x20000001        0x0 alarm_manager.o
 .data          0x20000001        0x0 driver.o
 .data          0x20000001        0x0 main.o
 .data          0x20000001        0x0 startup.o
                0x20000004                . = ALIGN (0x4)
 *fill*         0x20000001        0x3
                0x20000004                _E_DATA = .

.igot.plt       0x20000004        0x0 load address 0x0800047c
 .igot.plt      0x20000004        0x0 Main_Block.o

.bss            0x20000004     0x1422 load address 0x0800047c
                0x20000004                _S_bss = .
 *(.bss*)
 .bss           0x20000004        0x4 Main_Block.o
 .bss           0x20000008        0x4 Pressure_sensor.o
 .bss           0x2000000c        0x0 alarm_actuator.o
 .bss           0x2000000c        0x0 alarm_manager.o
 .bss           0x2000000c        0x0 driver.o
 .bss           0x2000000c        0x0 main.o
 .bss           0x2000000c      0x400 startup.o
                0x2000040c                _E_bss = .
                0x2000040c                . = ALIGN (0x4)
                0x2000140c                . = (. + 0x1000)
 *fill*         0x2000040c     0x1000
                0x2000140c                _stack_top = .
 COMMON         0x2000140c       0x1a Main_Block.o
                0x2000140c                MB_State
```

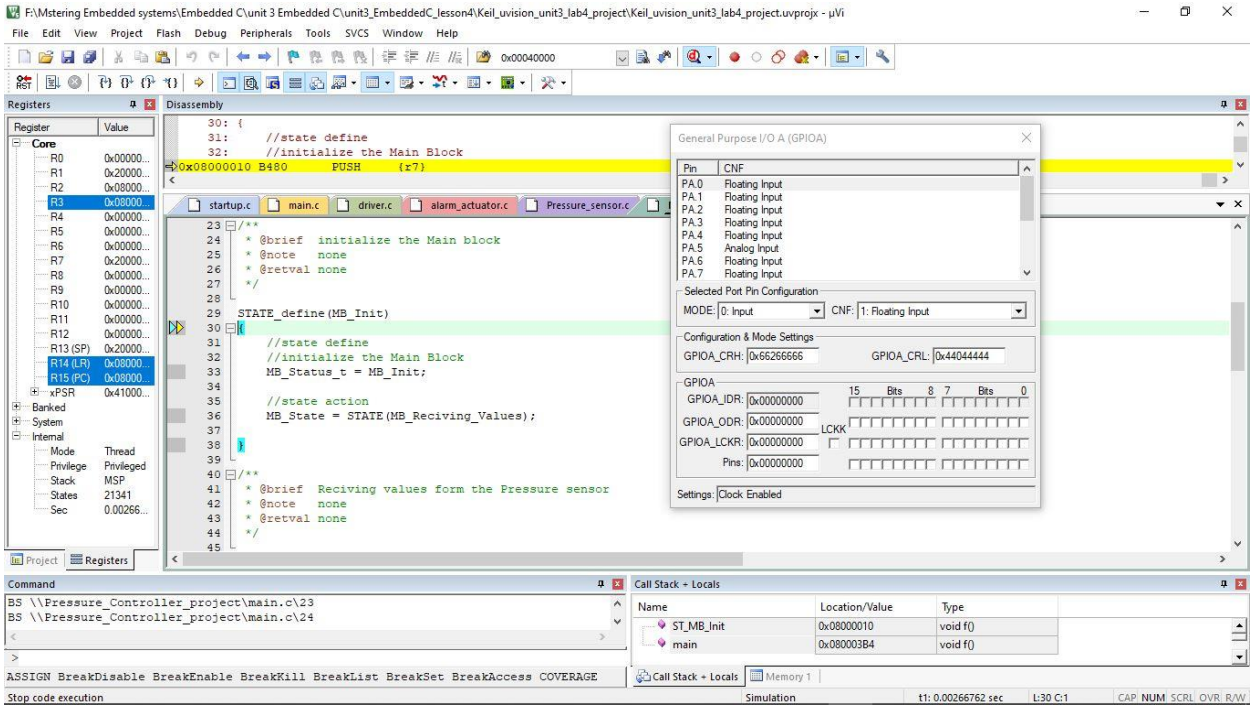
- Debugging and testing using keil:
  - Setup function:



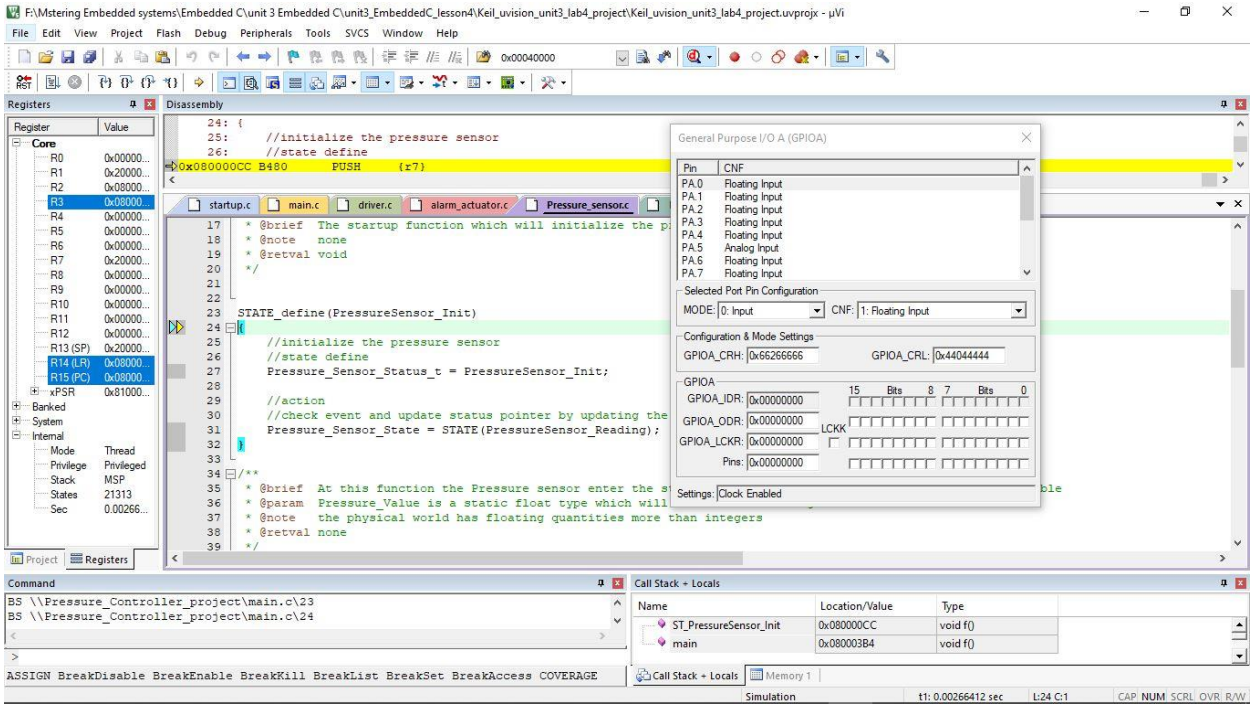
  - GPIO PortA initialization:



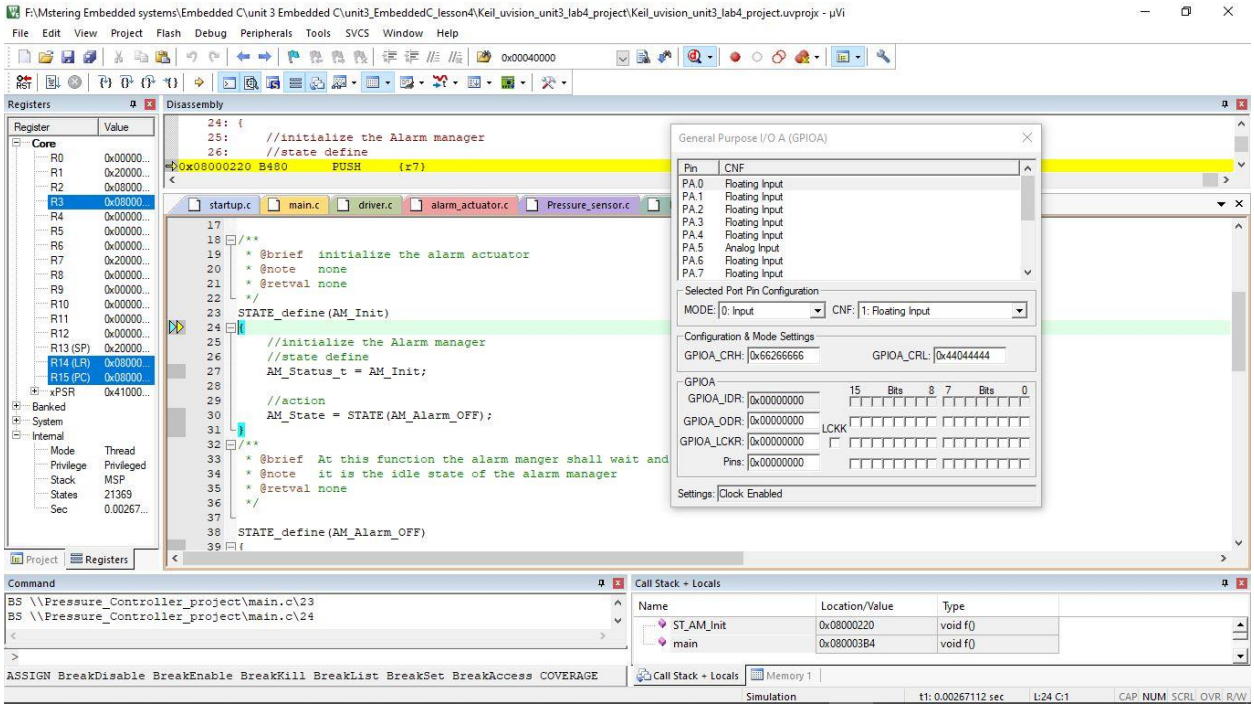
- Main Block initialization:



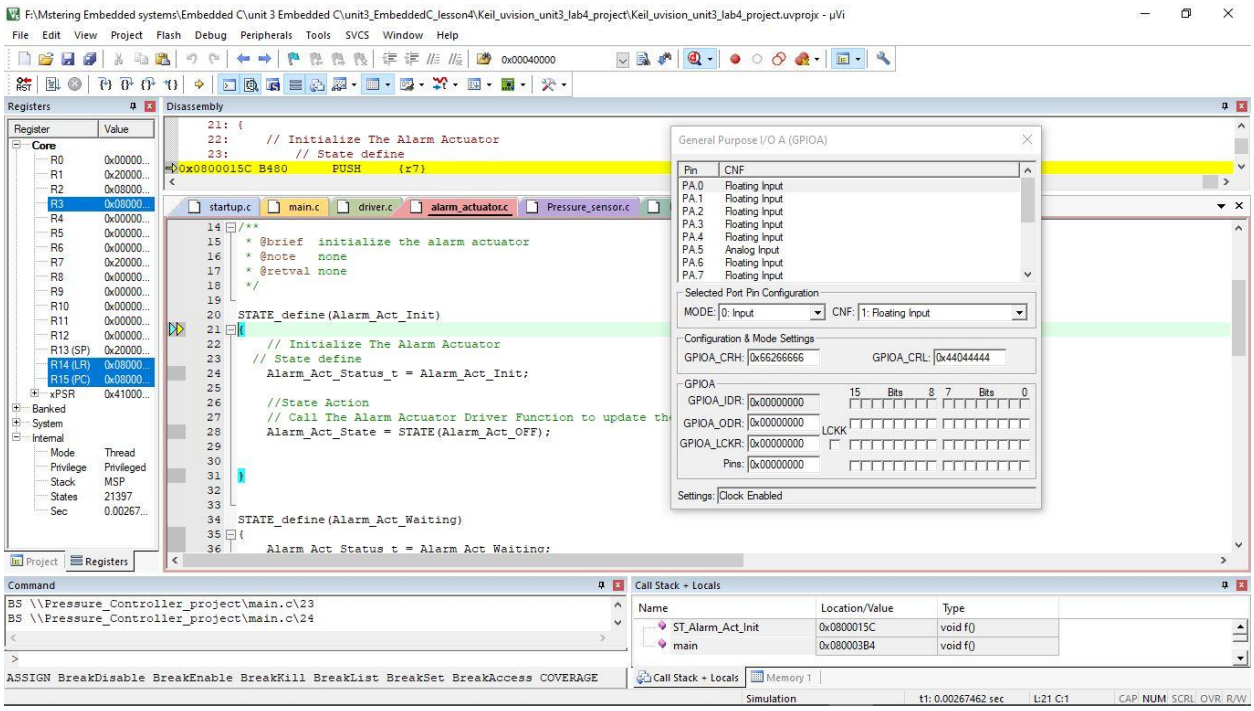
- Pressure sensor initialization:



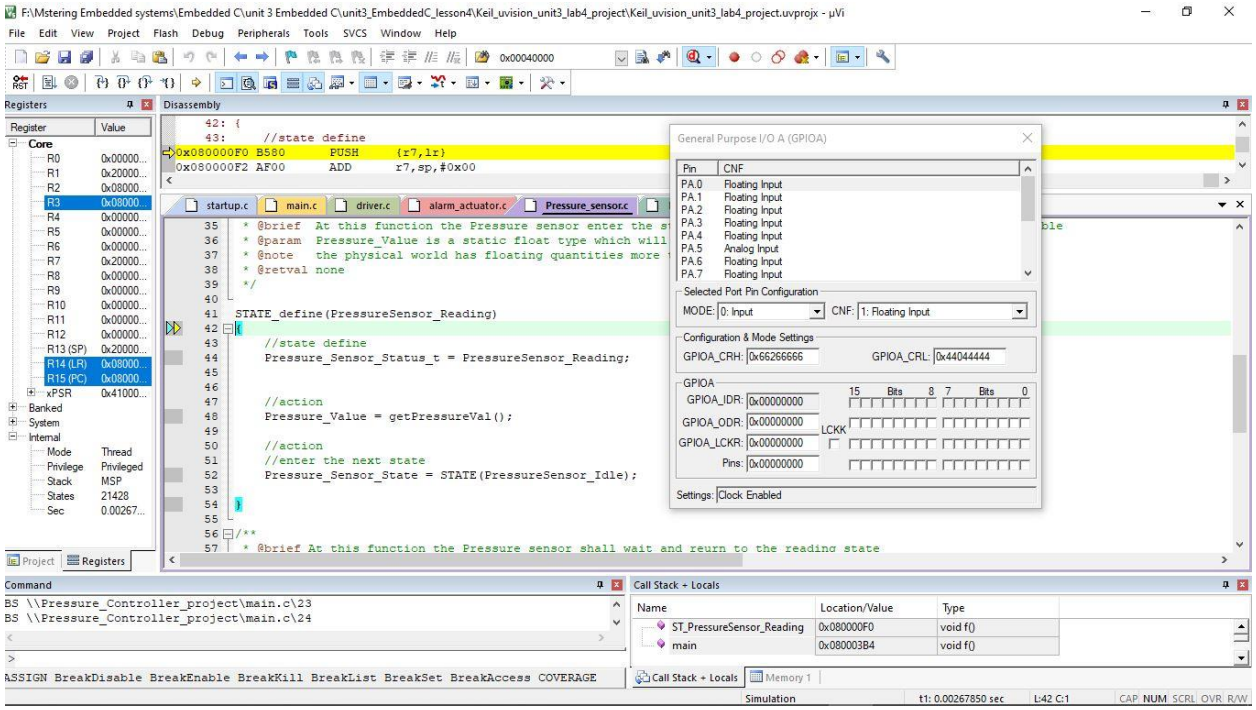
- Pressure sensor initialization:



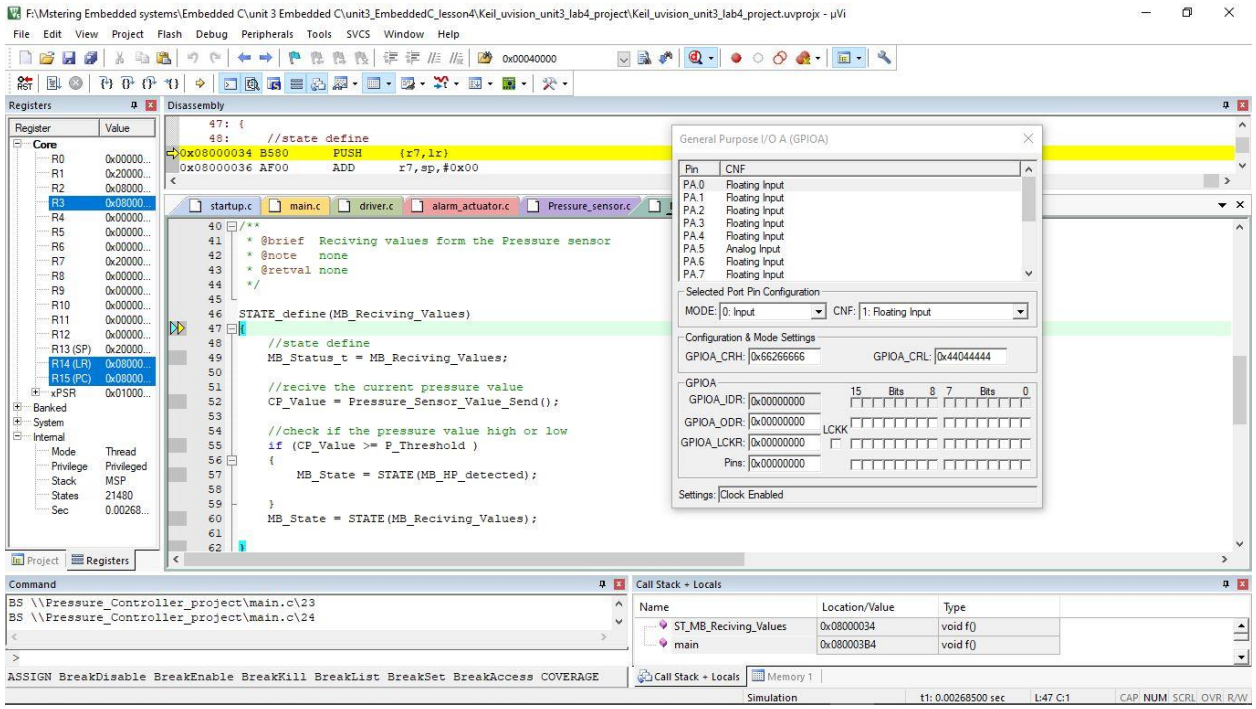
- Alarm actuator initialization:



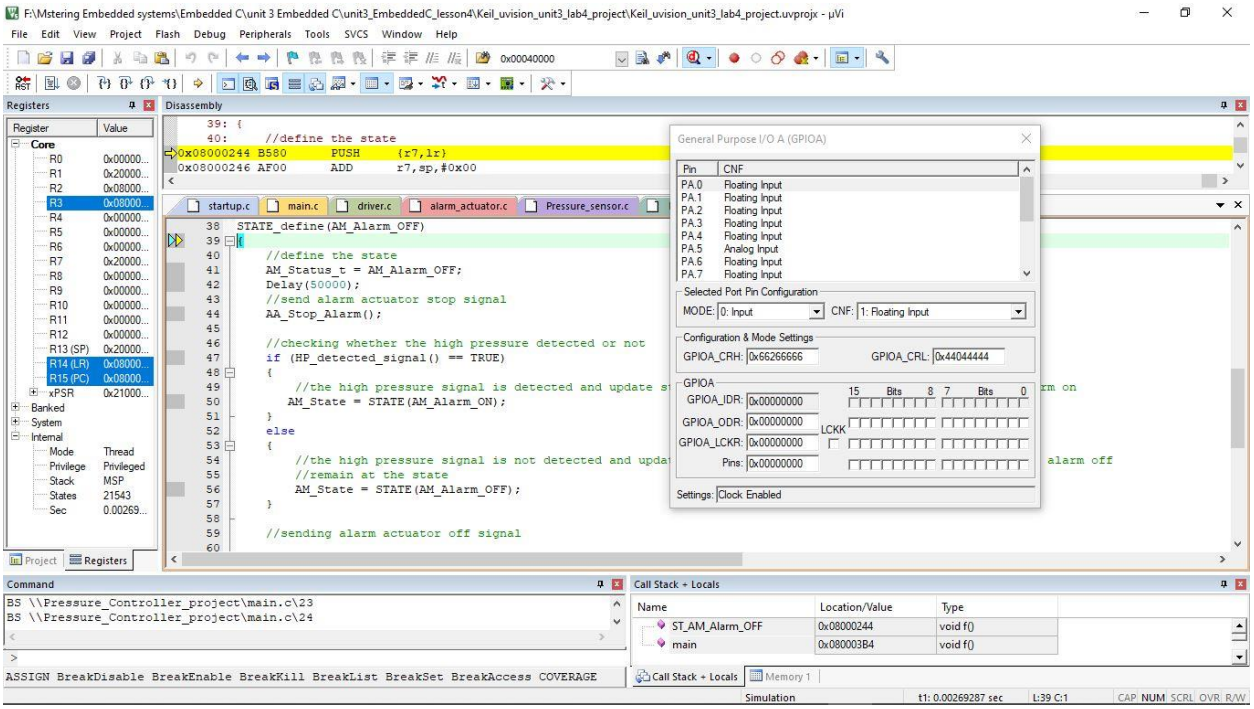
- pressure sensor reading state



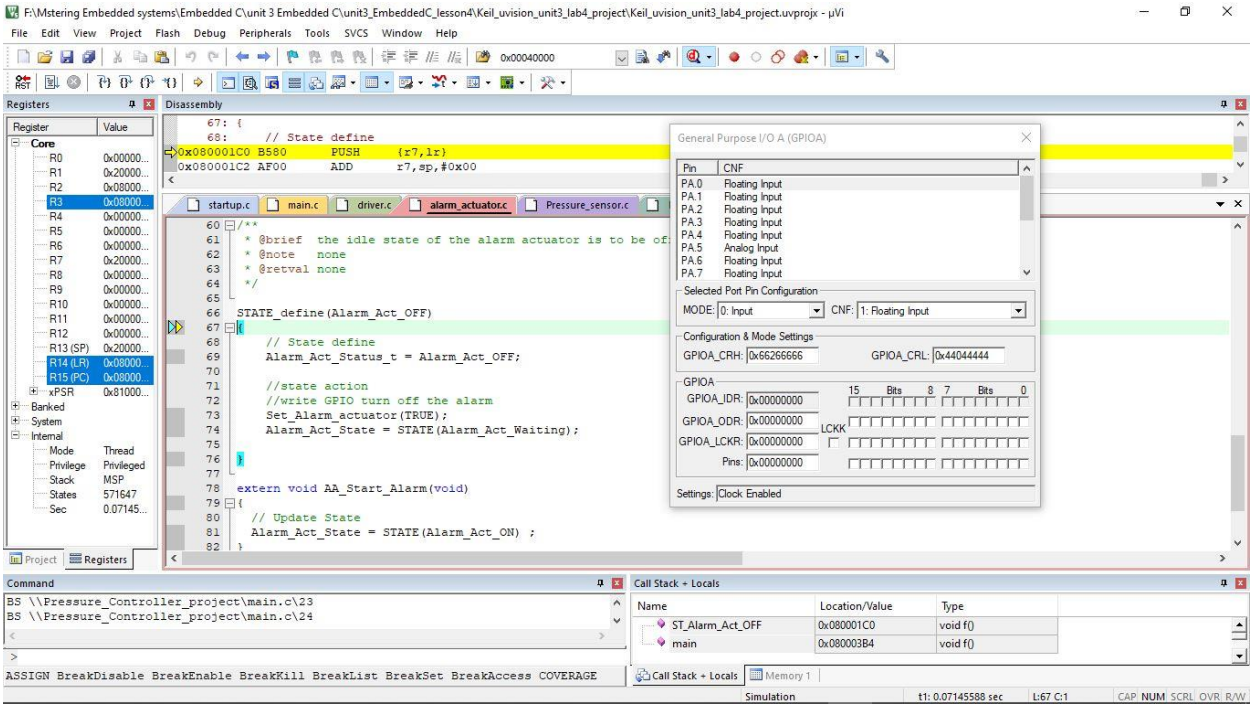
- main block receiving pressure values state:



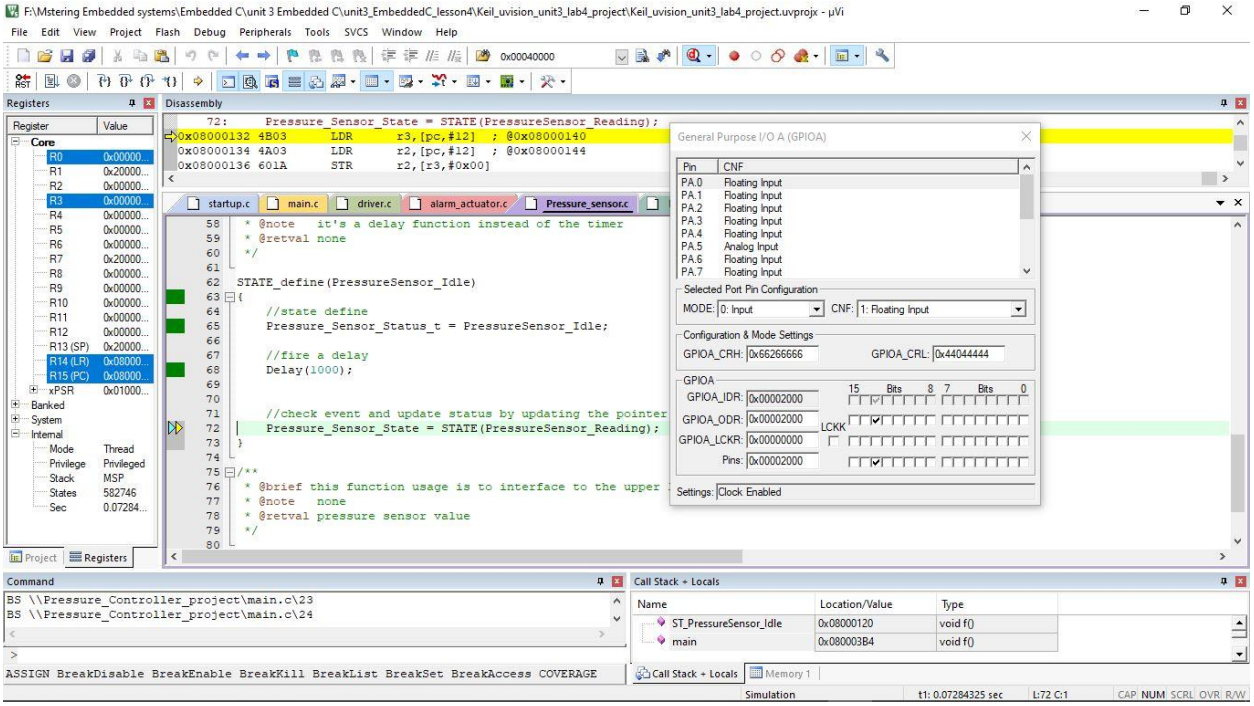
- Alarm manager alarm off state:



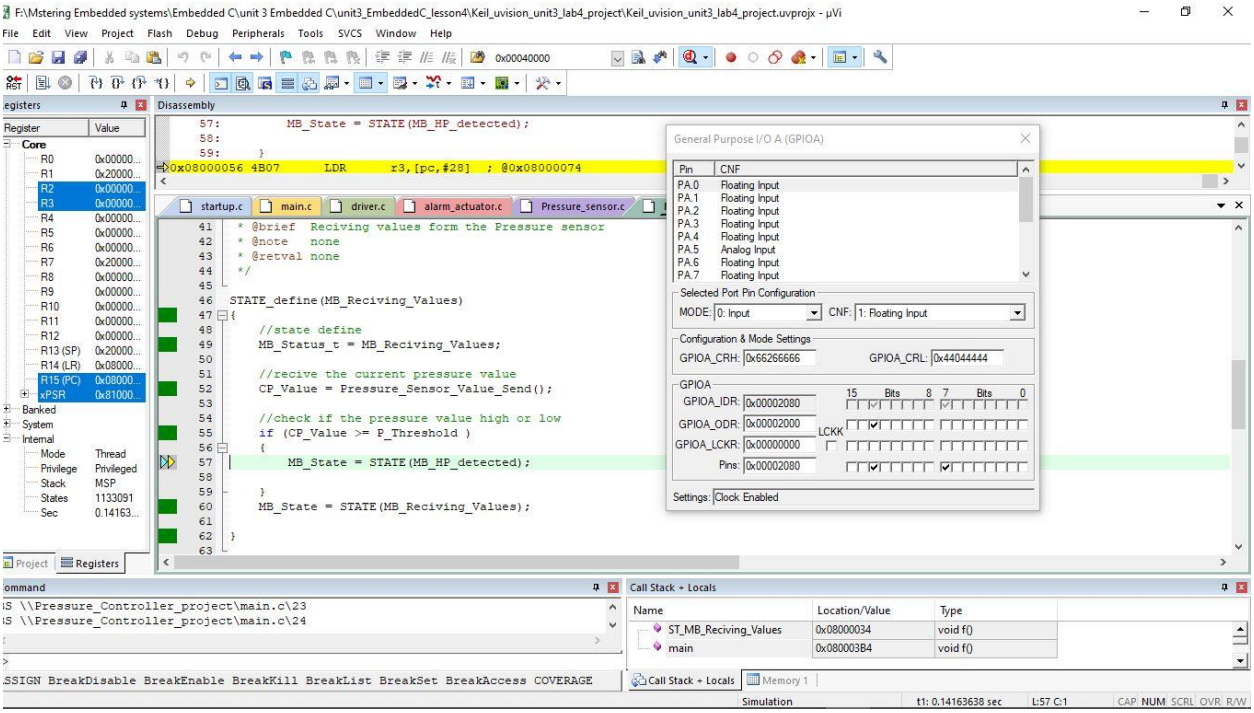
- alarm actuator off state:



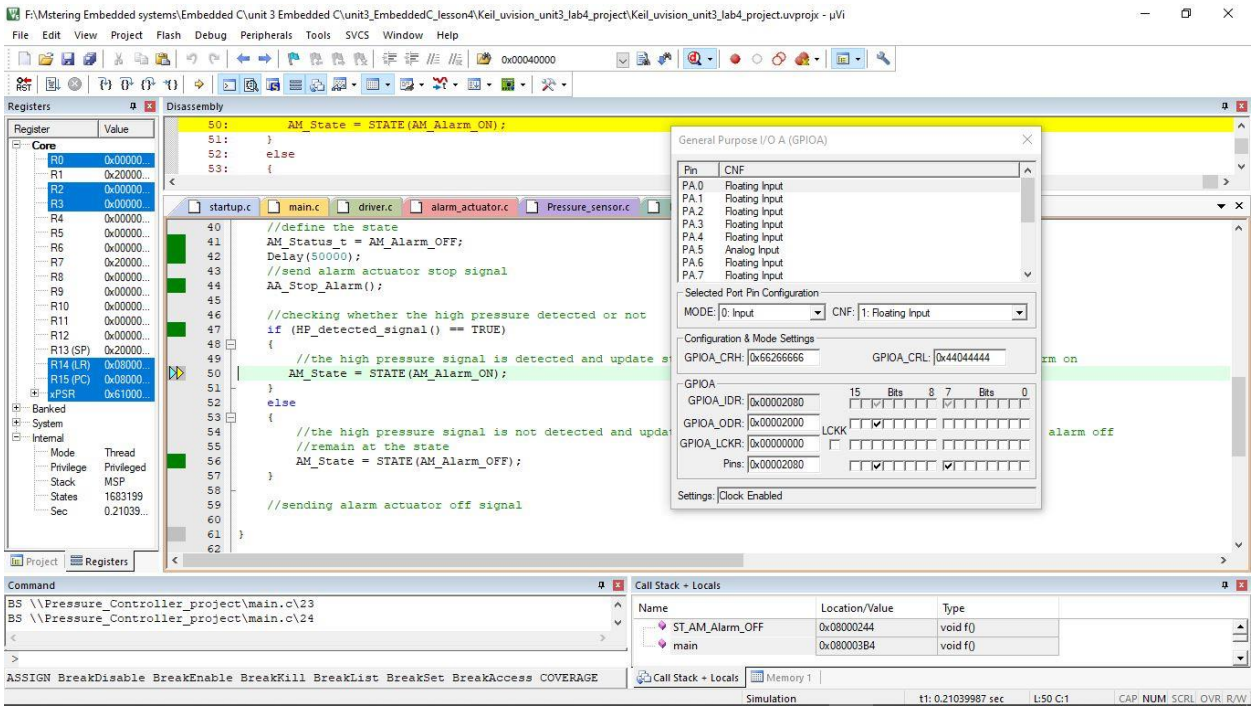
- pressure sensor idle state:



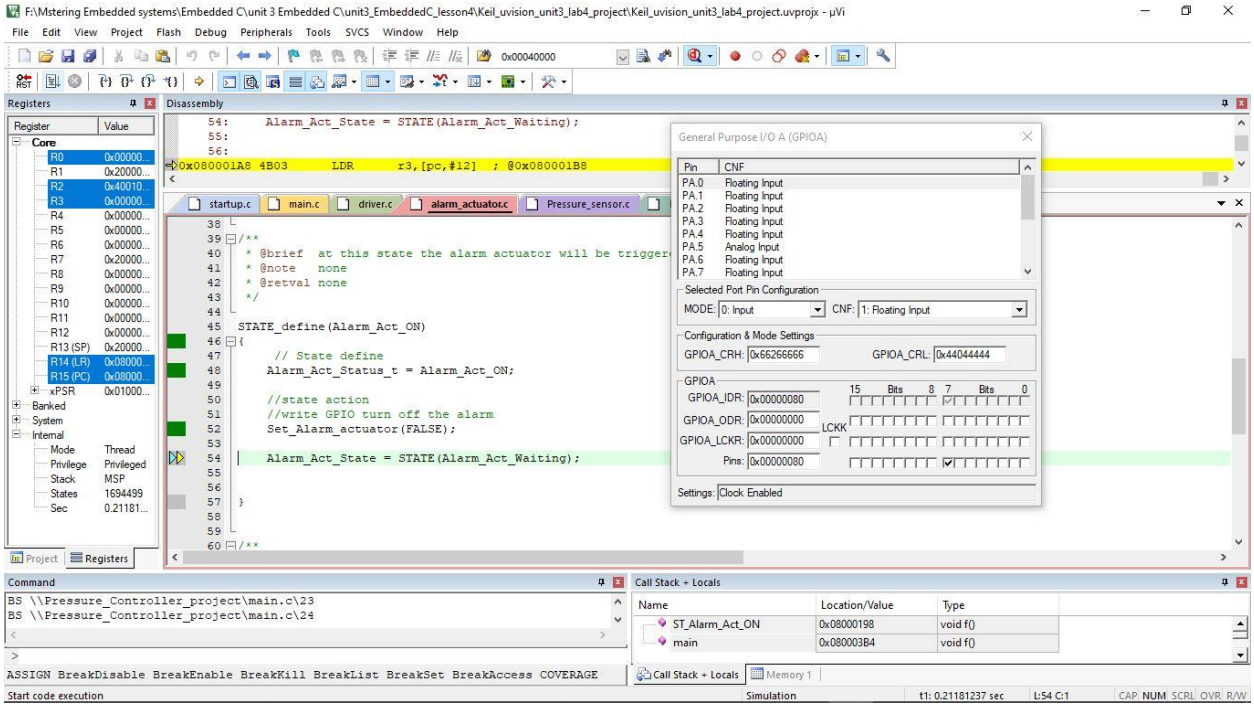
- main block detects high pressure:



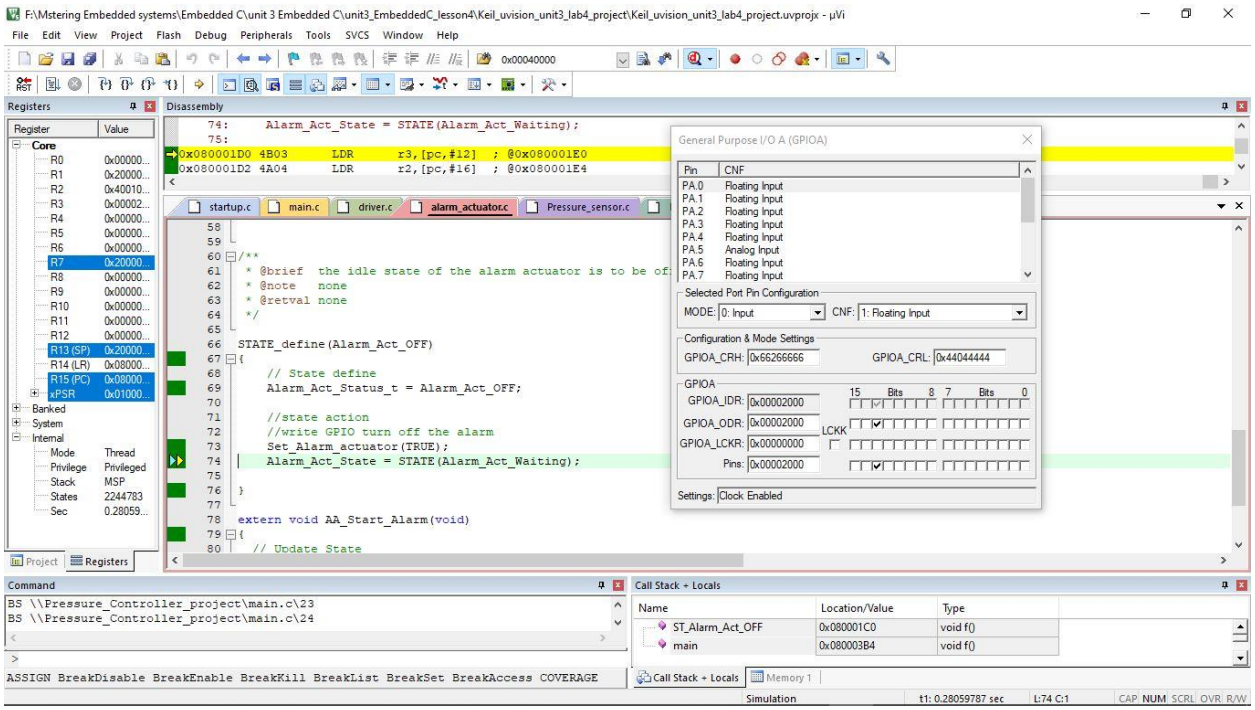
- alarm manager received high pressure signal from the main block and get into alarm on state:



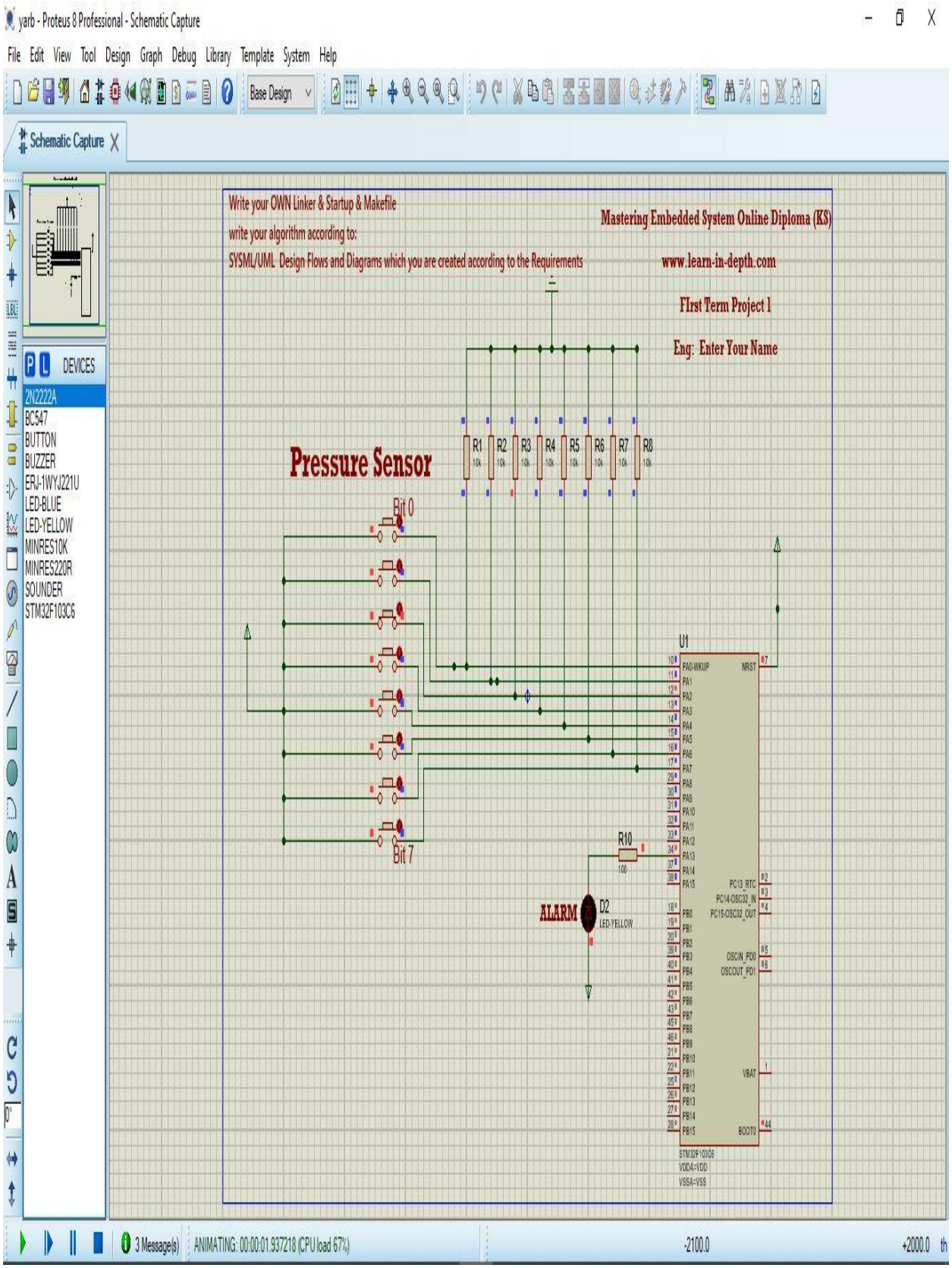
- alarm actuator turn on the alarm led:

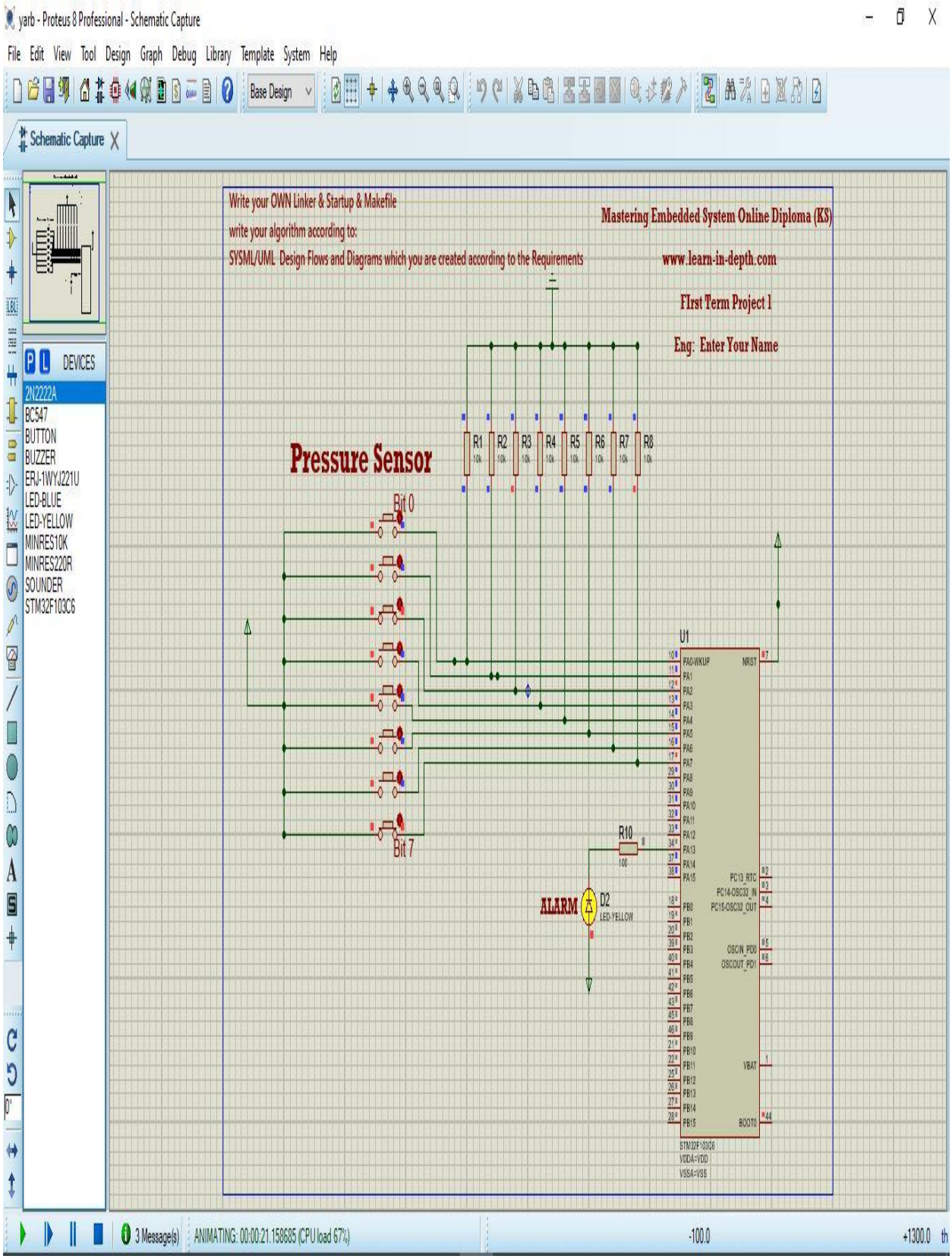


- alarm actuator turn off the led after the end of the delay:



- Simulation:


yarb - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Base Design
Schematic Capture
Write your OWN Linker & Startup & Makefile
write your algorithm according to:
SYSML/UML Design Flows 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: Enter Your Name
Pressure Sensor
DEVICES
2N2222A
BC547
BUTTON
BUZZER
ERJ-1WYJ221U
LED-BLUE
LED-YELLOW
MINRES10K
MINRES220R
SOUNDER
STM32F103C6
R1 R2 R3 R4 R5 R6 R7 R8
Bit 0
Bit 7
U1
R10
ALARM
D2
LED-YELLOW
3 Message(s)
ANIMATING: 00:00:01.937218 (CPU load 67%)
-2100.0
+2000.0


yarb - Proteus 8 Professional - Schematic Capture
File Edit View Tool Design Graph Debug Library Template System Help
Base Design
Schematic Capture
Write your OWN Linker & Startup & Makefile
write your algorithm according to:
SYSML/UML Design Flows 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: Enter Your Name
Pressure Sensor
DEVICES
2N2222A
BC547
BUTTON
BUZZER
ERJ-1WYJ221U
LED-BLUE
LED-YELLOW
MINRES10K
MINRES220R
SOUNDER
STM32F103C6
R1 R2 R3 R4 R5 R6 R7 R8
Bit 0
Bit 7
U1
R10
ALARM
D2
LED-YELLOW
3 Message(s)
ANIMATING: 00:00:21.158685 (CPU load 67%)
-100.0
+1300.0