

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.

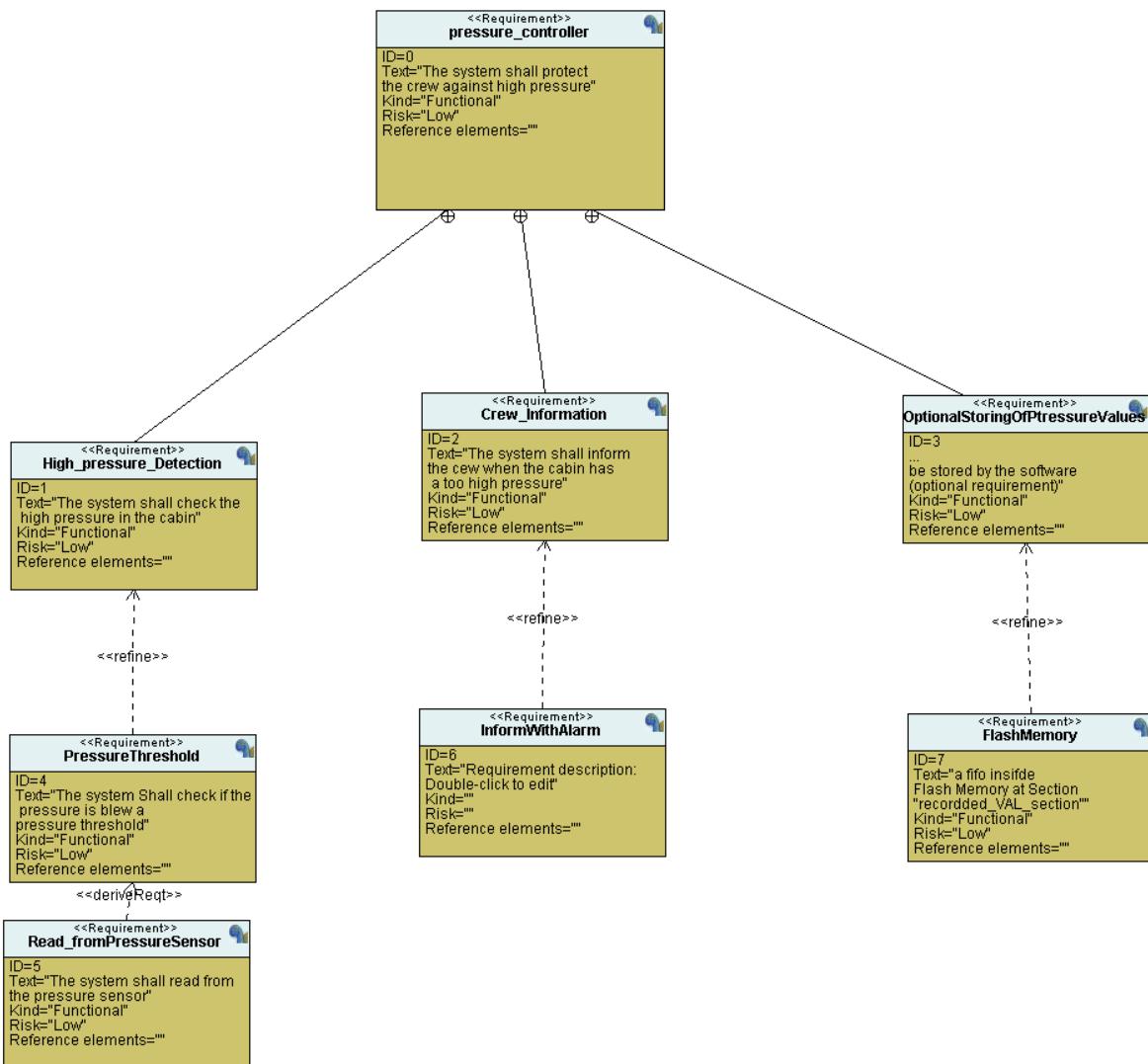
- Methodology:

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.

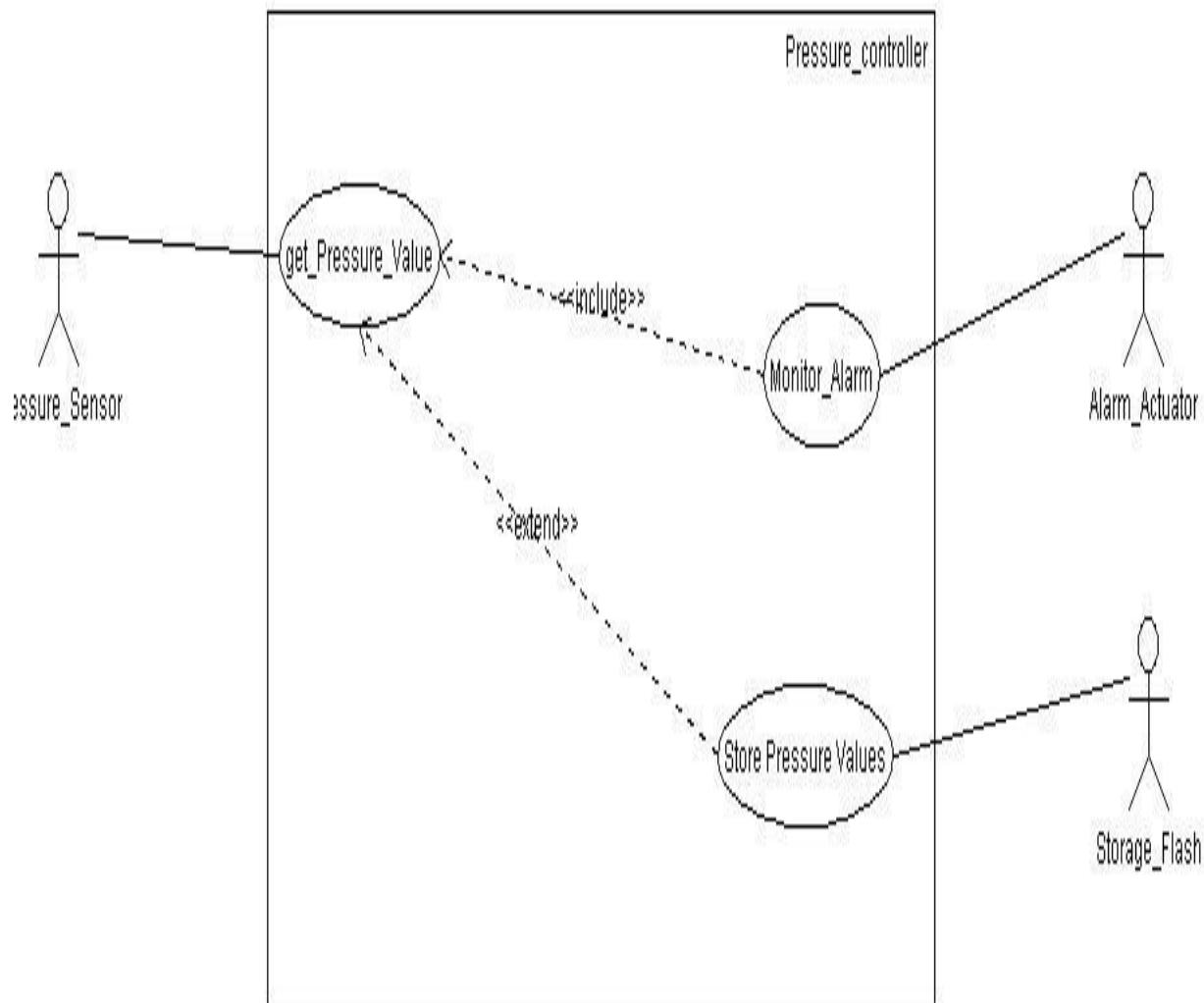
- **HW/SW Partitioning:**

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).

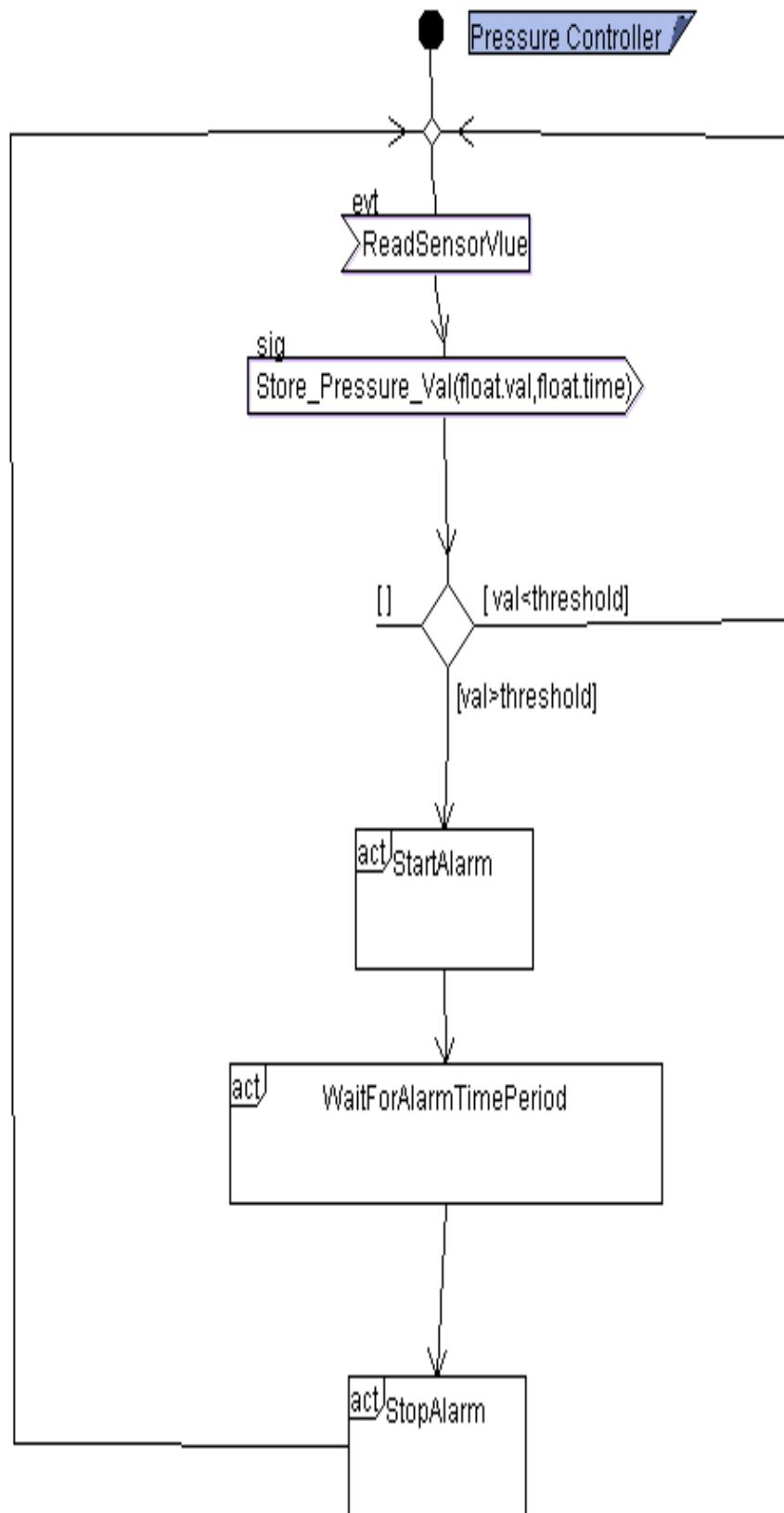
- **Requirement Diagram:**



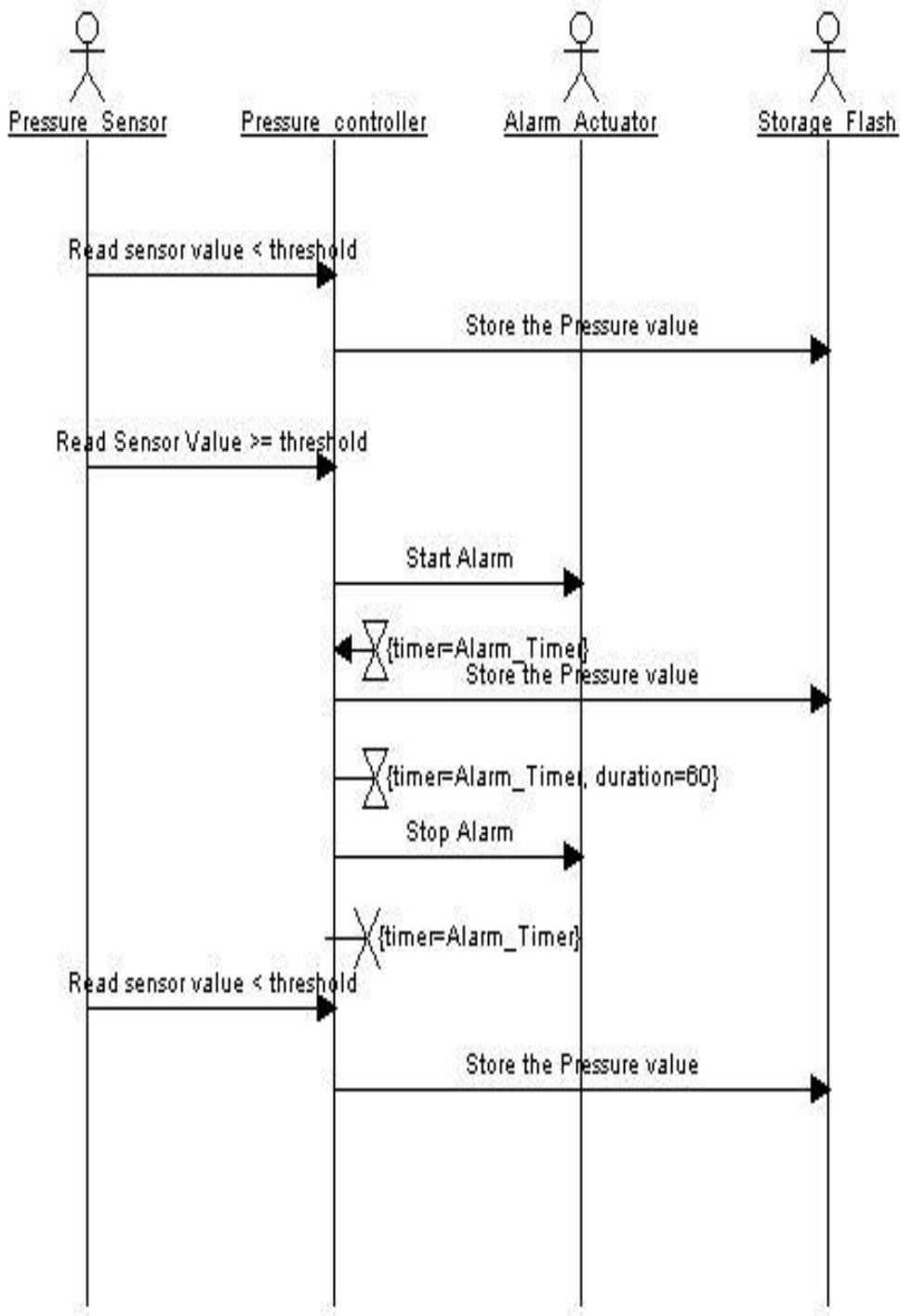
- System analysis:
 - Use case diagram:



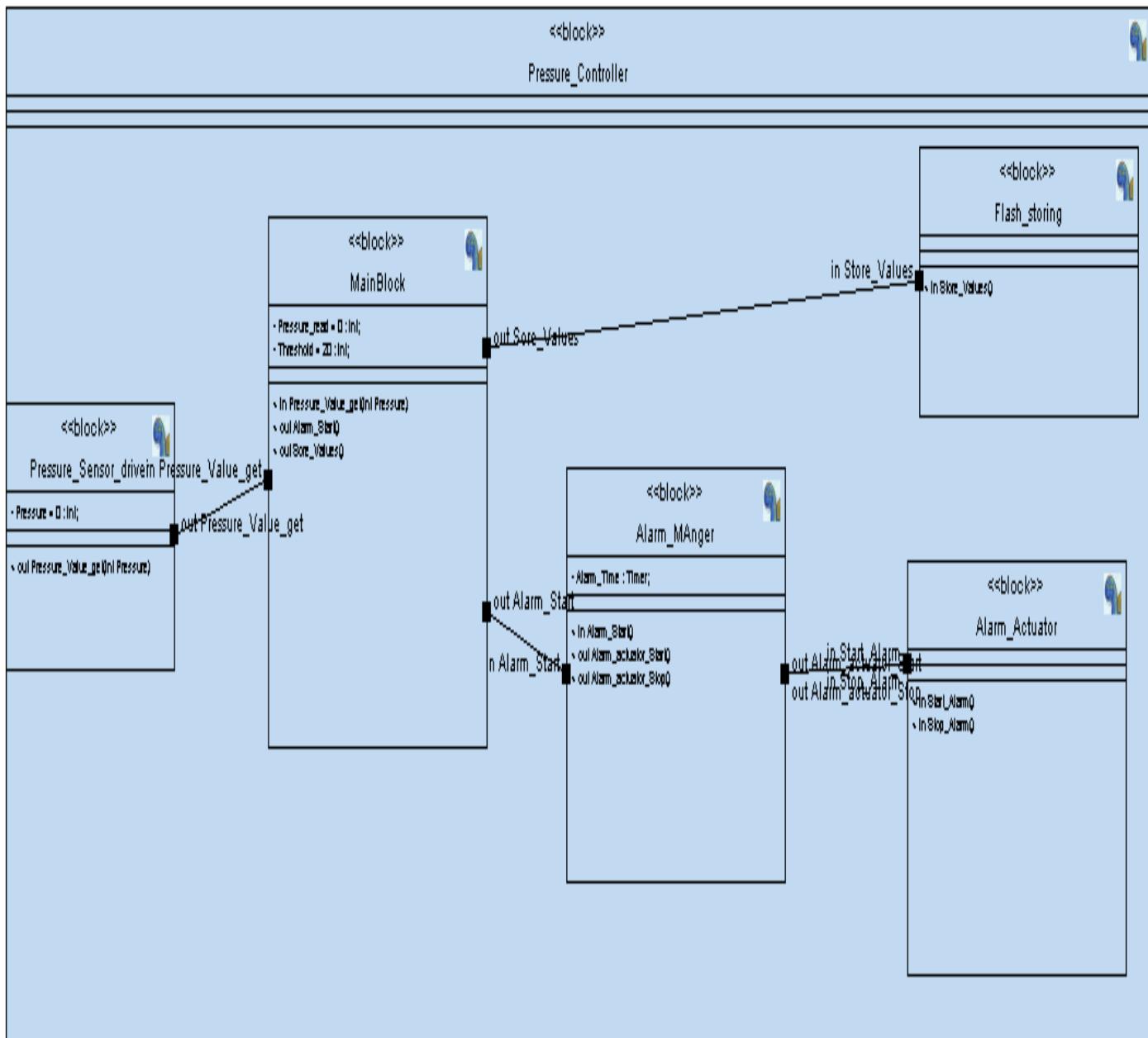
- Activity diagram:



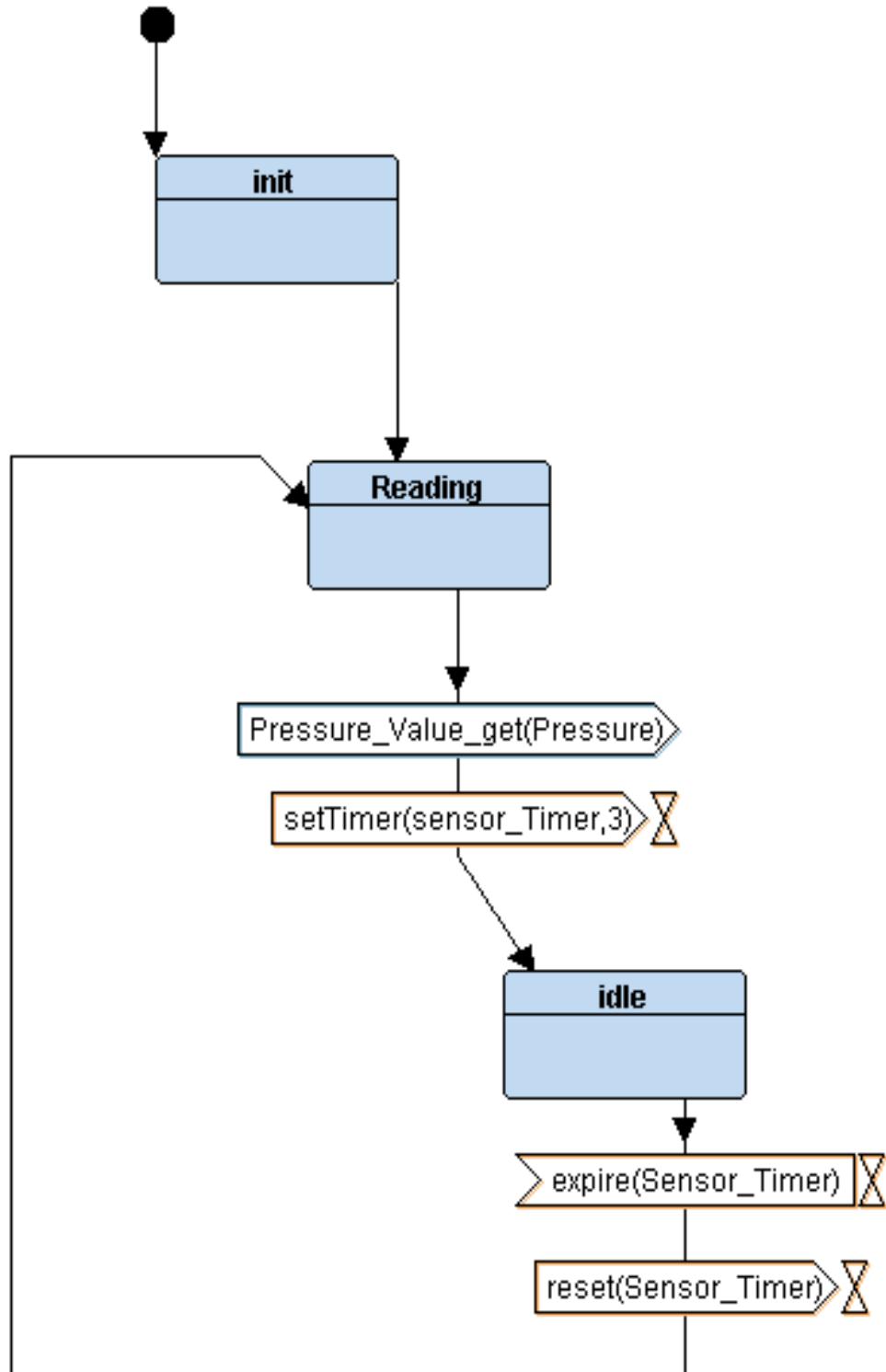
- Sequence diagram:



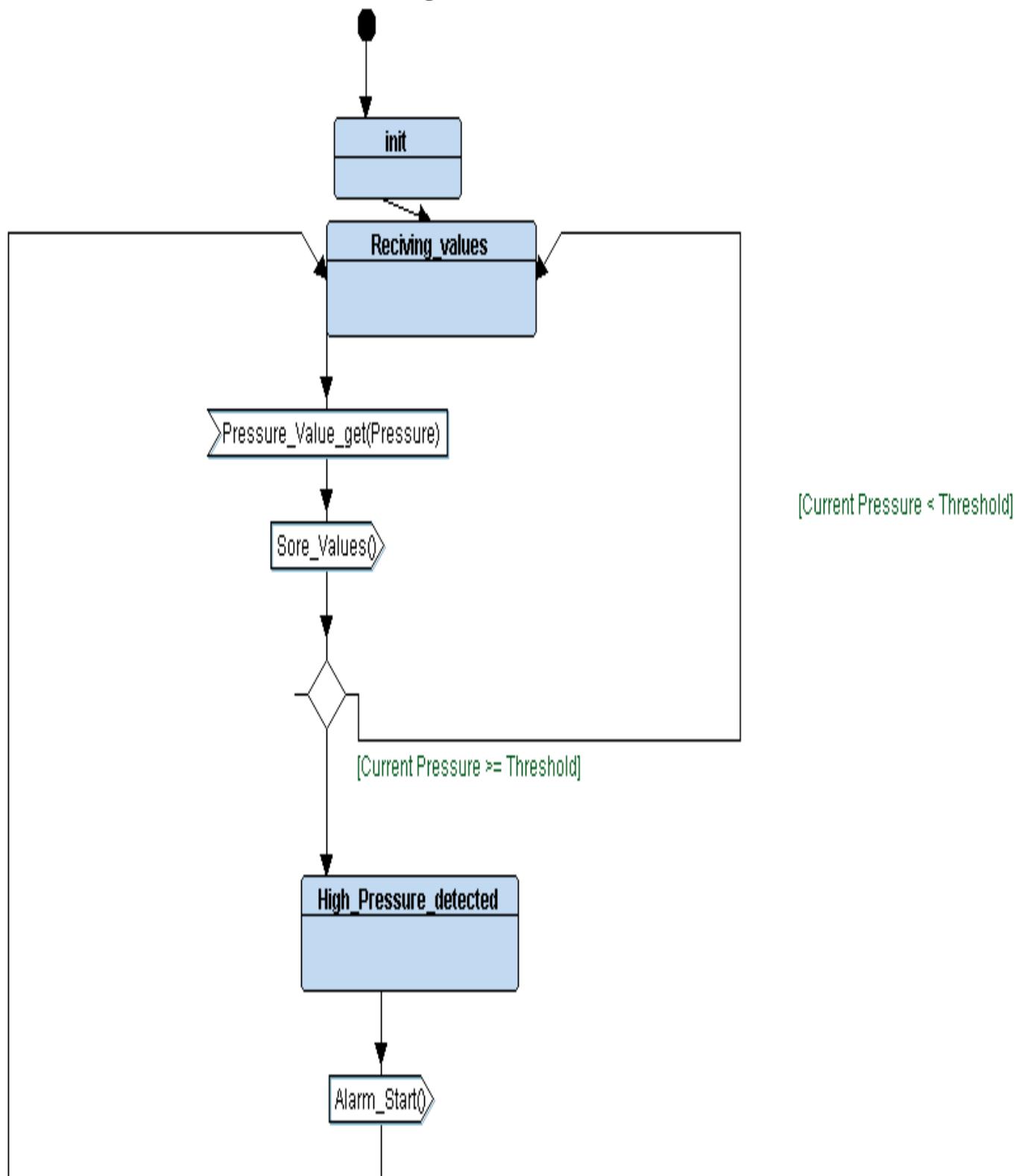
- System design:



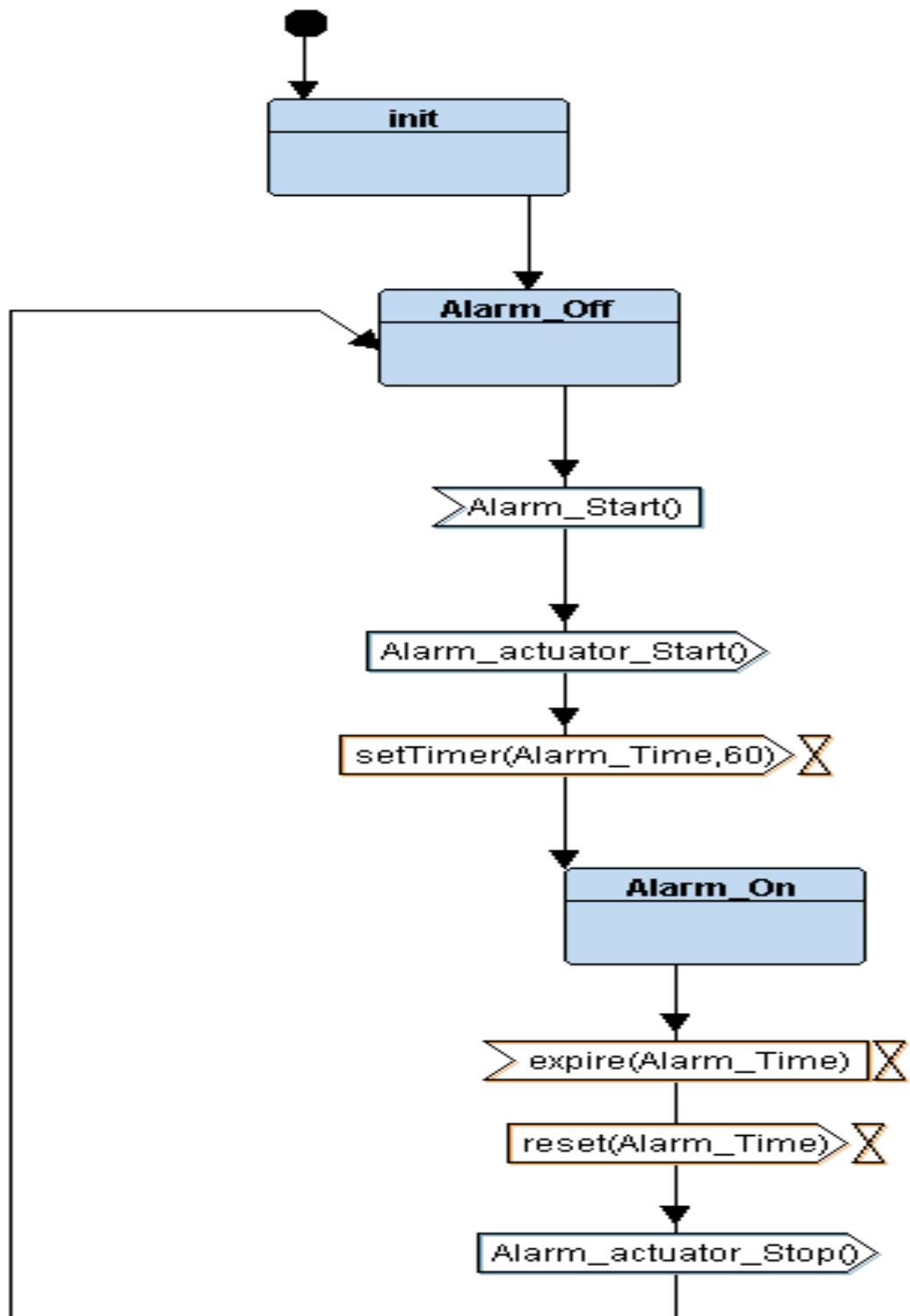
- Pressure state diagram:



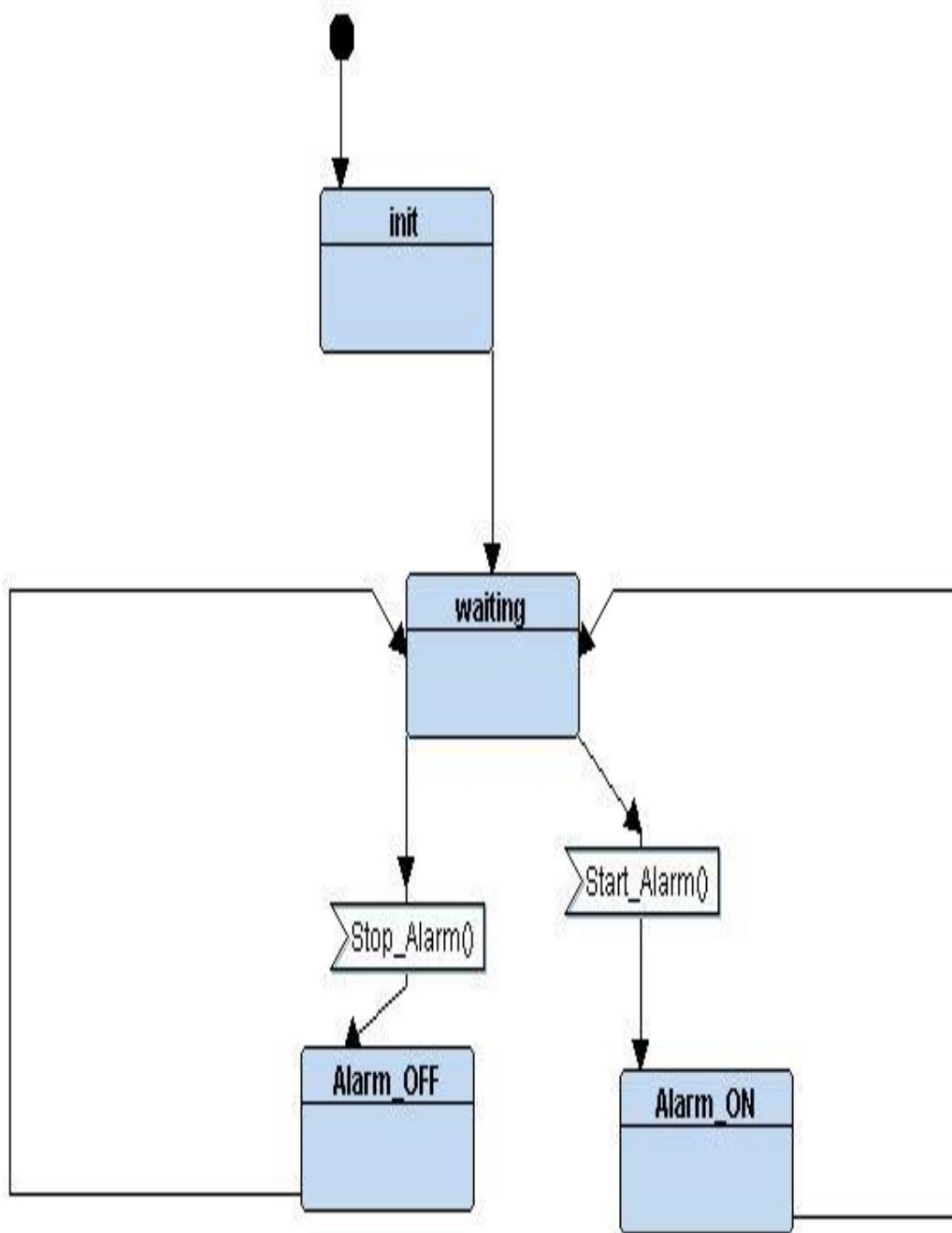
- Main Block state diagram:



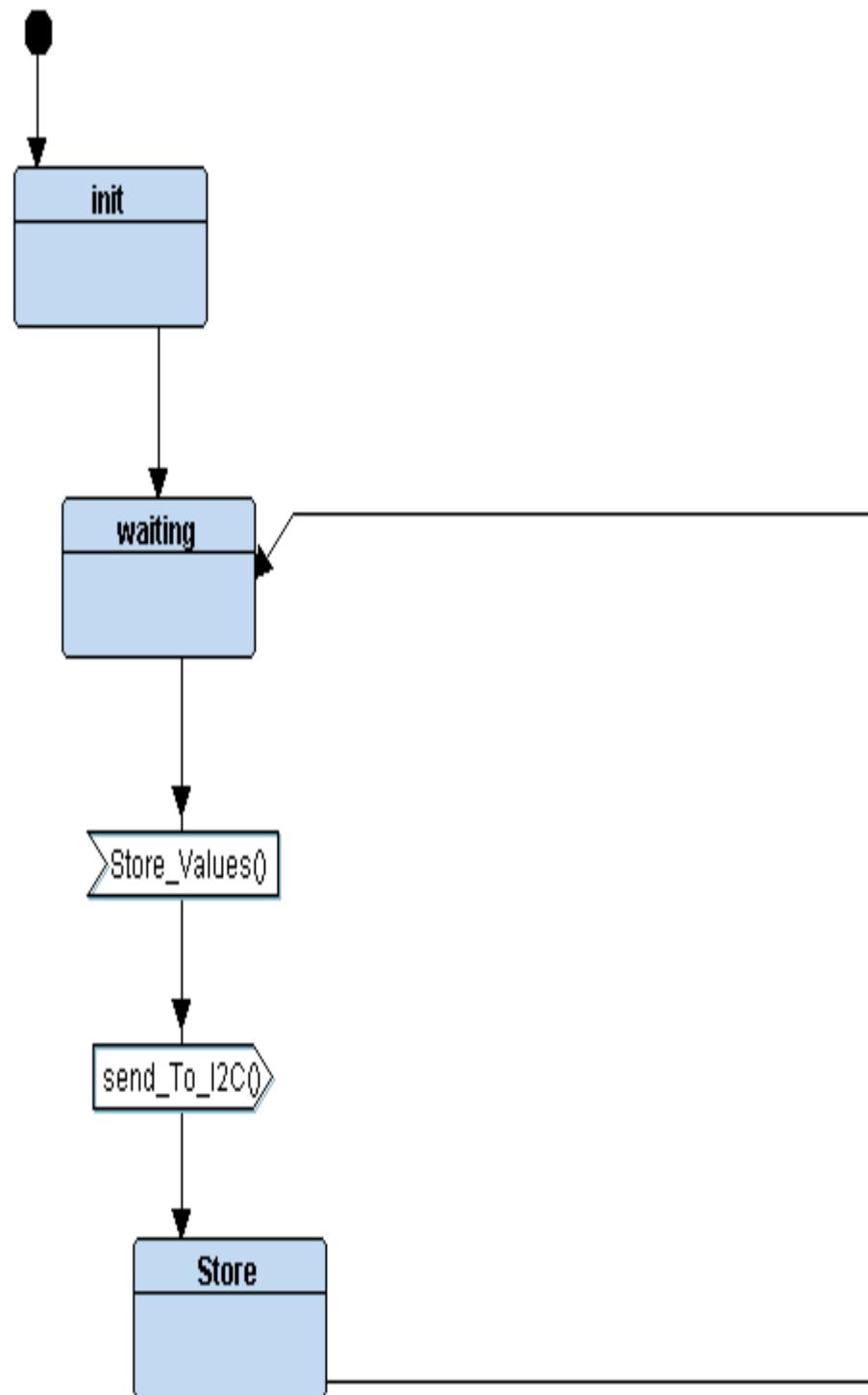
- Alarm Manager state diagram:



- 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

```
1 #include <stdint.h>
2 #include <stdio.h>
3
4 #include "driver.h"
5 #include "Main_Block.h"
6
7 void Setup(void)
8 {
9     GPIO_INITIALIZATION();
10    MB_State = STATE(MB_Init);
11    AM_State = STATE(AM_Init);
12    Pressure_Sensor_State = STATE(PressureSensor_Init);
13    Alarm_Act_State = STATE(Alarm_Act_Init);
14
15 }
16 void main ()
17 {
18     Setup();
19     while (1)
20     {
21         Pressure_Sensor_State();
22         MB_State();
23         AM_State();
24         Alarm_Act_State();
25     }
26
27 }
28
```

- State.h

```
C state.h > STATE_define(_state_Func_)
1  /**
2  * @file state.h
3  * @author Ahmed khaled khalaf (khalaafawy22899@gmail.com)
4  * @brief
5  * @version 0.1
6  * @date 2024-06-22
7  *
8  * @copyright Copyright (c) 2024
9  *
10 */
11
12 #ifndef _STATE_H_
13 #define _STATE_H_
14
15 //Standered data types file include
16 #include "STD_DataTypes.h"
17 #include "driver.h"
18
19 //Automatic State Functions declaration
20 #define STATE_define(_state_Func_) void ST_##_state_Func_()
21 #define STATE(_state_Func_) ST_##_state_Func_
22
23 #endif
24
```

- driver .h and .c

```
C driver.h > Set_Alarm_actuator(int)
1  #include <stdint.h>
2  #include <stdio.h>
3
4  #define SET_BIT(ADDRESS,BIT) ADDRESS |= (1<<BIT)
5  #define RESET_BIT(ADDRESS,BIT) ADDRESS &= ~(1<<BIT)
6  #define TOGGLE_BIT(ADDRESS,BIT) ADDRESS ^= (1<<BIT)
7  #define READ_BIT(ADDRESS,BIT) ((ADDRESS) & (1<<(BIT)))
8
9
10 #define GPIOA_PORTA 0x40010800
11 #define BASE_RCC 0x40021000
12
13 #define APB2ENR *(volatile uint32_t *)(BASE_RCC + 0x18)
14
15 #define GPIOA_CRL *(volatile uint32_t *)(GPIO_PORTA + 0x00)
16 #define GPIOA_CRH *(volatile uint32_t *)(GPIO_PORTA + 0X04)
17 #define GPIOA_IDR *(volatile uint32_t *)(GPIO_PORTA + 0x08)
18 #define GPIOA_ODR *(volatile uint32_t *)(GPIO_PORTA + 0x0C)
19
20
21 void Delay(int nCount);
22 int getPressureVal();
23 void Set_Alarm_actuator(int i);
24 void GPIO_INITIALIZATION ();
```

```
C driver.c > getPressureVal()
1  #include "driver.h"
2  #include <stdint.h>
3  #include <stdio.h>
4  void Delay(int nCount)
5  {
6      for(; nCount != 0; nCount--);
7  }
8
9  int getPressureVal(){
10     return (GPIOA_IDR & 0xFF);
11 }
12
13 void Set_Alarm_actuator(int i){
14     if (i == 1){
15         SET_BIT(GPIOA_ODR,13);
16     }
17     else if (i == 0){
18         RESET_BIT(GPIOA_ODR,13);
19     }
20 }
21
22 void GPIO_INITIALIZATION (){
23     SET_BIT(APB2ENR, 2);
24     GPIOA_CRL &= 0xF0FFFFFF;
25     GPIOA_CRL |= 0x00000000;
26     GPIOA_CRH &= 0xF0FFFFFF;
27     GPIOA_CRH |= 0x22222222;
28 }
29
```

● Pressure sensor driver

```
C Pressure_sensor.h > ...
1  /**
2   * @file Pressure_sensor.h
3   * @author Ahmed khaled khalaf (khalaafawy22899@gmail.com)
4   * @brief This file contains all the prototypes and macros needed to Pressure sensor driver implementation
5   * @version 0.1
6   * @date 2024-06-21
7   *
8   * @copyright Copyright (c) 2024
9   *
10 */
11
12
13 #ifndef _PRESSURE_SENSOR_H_
14 #define _PRESSURE_SENSOR_H_
15 #include "state.h"
16
17 //the Pressure sensor state
18 enum{
19
20     PressureSensor_Init,
21     PressureSensor_Reading,
22     PressureSensor_Idle
23 }Pressure_Sensor_Status_t;
24
25 //prototypes of the pressure sensor states function
26 STATE_define(PressureSensor_Init);
27 STATE_define(PressureSensor_Reading);
28 STATE_define(PressureSensor_Idle);
29 uint32_t Pressure_Sensor_Value_Send(void);
30
31 //Pressure Sensor Module Pointer to state functions
32 void (*Pressure_Sensor_State) ();
33
34#endif

C Pressure_sensor.c > Pressure_Sensor_Value_Send(void)
1 /**
2  * @file Pressure_sensor.c
3  * @author Ahmed khaled khalaf (khalaafawy22899@gmail.com)
4  * @brief the source file of the pressure sensor driver contains the implementation and APIs
5  * @version 0.1
6  * @date 2024-06-22
7  *
8  * @copyright Copyright (c) 2024
9  *
10 */
11
12
13 #include "Pressure_sensor.h"
14 static uint32_t Pressure_Value;
15
16 /**
17  * @brief The startup function which will initialize the pressure sensor
18  * @note none
19  * @retval void
20  */
21
22
23 STATE_define(PressureSensor_Init)
24 {
25     //initialize the pressure sensor
26     //state define
27     Pressure_Sensor_Status_t = PressureSensor_Init;
28
29     //action
30     //check event and update status pointer by updating the pointer to the next state
31     Pressure_Sensor_State = STATE(PressureSensor_Reading);
32 }
33
```

```

C Pressure_sensor.c > ⚡ Pressure_Sensor_Value_Send(void)
34 /**
35 * @brief At this function the Pressure sensor enter the state reading and shall store the reading at a variable
36 * @param Pressure_Value is a static float type which will contain the reading of the sensor
37 * @note the physical world has floating quantities more than integers
38 * @retval none
39 */
40
41 STATE_define(PressureSensor_Reading)
42 {
43     //state define
44     Pressure_Sensor_Status_t = PressureSensor_Reading;
45
46     //action
47     Pressure_Value = getPressureVal();
48
49     //action
50     //enter the next state
51     Pressure_Sensor_State = STATE(PressureSensor_Idle);
52
53 }

C Pressure_sensor.c > ⚡ Pressure_Sensor_Value_Send(void)
54 }
55
56 /**
57 * @brief At this function the Pressure sensor shall wait and return to the reading state
58 * @note it's a delay function instead of the timer
59 * @retval none
60 */
61
62 STATE_define(PressureSensor_Idle)
63 {
64     //state define
65     Pressure_Sensor_Status_t = PressureSensor_Idle;
66
67     //fire a delay
68     Delay(1000);
69
70
71     //check event and update status by updating the pointer to the next state
72     Pressure_Sensor_State = STATE(PressureSensor_Reading);
73 }

74 /**
75 * @brief this function usage is to interface to the upper layer and send the read value of the sensor
76 * @note none
77 * @retval pressure sensor value
78 */
79
80
81 uint32_t Pressure_Sensor_Value_Send(void)
82 {
83     return Pressure_Value;
84 }
85

```

- Main block:

```
C Main_Block.h > ...
1  /**
2  * @file Main_Block.h
3  * @author Ahmed khaled khalaf (khalaftawy22899@gmail.com)
4  * @brief This file contains all the prototypes and macros needed to Main Block driver implementation
5  * @version 0.1
6  * @date 2024-06-22
7  *
8  * @copyright Copyright (c) 2024
9  *
10 */
11
12 #ifndef _MAIN_BLOCK_H_
13 #define _MAIN_BLOCK_H_
14 #include "state.h"
15 #include "alarm_manager.h"
16 #include "Pressure_sensor.h"
17
18 //define states of the man block
19 enum
20 {
21     MB_Init,
22     MB_Reciving_Values,
23     MB_HP_detected
24 }MB_Status_t;
25
26 // Declare State Function
27 STATE_define(MB_Init);
28 STATE_define(MB_Reciving_Values);
29 STATE_define(MB_HP_detected);
30 uint8_t HP_detected_signal(void);
31 // State Pointer to Function
32 void (* MB_State)();
33
34 #endif
```

```
C Main_Block.c > ...
1  /**
2  * @file Main_Block.c
3  * @author Ahmed khaled khalaf (khalaftawy22899@gmail.com)
4  * @brief the source file of the Main Block driver contains the implementation and APIs
5  * @version 0.1
6  * @date 2024-06-22
7  *
8  * @copyright Copyright (c) 2024
9  *
10 */
11
12 #include "Main_Block.h"
13
14 //define a variable to recive the current pressure value
15 static uint32_t CP_Value;
16 static uint8_t P_Threshold = 20;
17
18 /**
19 * @brief initialize the Main block
20 * @note none
21 * @retval none
22 */
23
24 STATE_define(MB_Init)
25 {
26     //state define
27     //initialize the Main Block
28     MB_Status_t = MB_Init;
29
30     //state action
31     MB_State = STATE(MB_Reciving_Values);
32
33 }
34
```

```

C Main_Block.c > ...
35  /**
36   * @brief Reciving values form the Pressure sensor
37   * @note none
38   * @retval none
39   */
40
41 STATE_define(MB_Reciving_Values)
42 {
43     //state define
44     MB_Status_t = MB_Reciving_Values;
45
46     //receive the current pressure value
47     CP_Value = Pressure_Sensor_Value_Send();
48
49     //check if the pressure value high or low
50     if (CP_Value >= P_Threshold )
51     {
52         MB_State = STATE(MB_HP_detected);
53     }
54     MB_State = STATE(MB_Reciving_Values);
55
56 }
57
58 /**
59  * @brief the main block detected a high pressure signal so it will send a signal to the alarm manager
60  *        to tigger the alarm actuator for 60 sec
61  * @note none
62  * @retval none
63  */
64
65
C Main_Block.c > ...
59 /**
60  * @brief the main block detected a high pressure signal so it will send a signal to the alarm manager
61  *        to tigger the alarm actuator for 60 sec
62  * @note none
63  * @retval none
64  */
65
66 STATE_define(MB_HP_detected)
67 {
68     //define state
69     MB_Status_t = MB_HP_detected;
70
71     //Action
72
73     HP_detected_signal();
74
75     MB_State = STATE(MB_Reciving_Values);
76
77 }
78
79 /**
80  * @brief sending a signal to the alarm manger
81  * @note none
82  * @retval True
83  */
84
85 uint8_t HP_detected_signal(void)
86 {
87
88
89     return (CP_Value > P_Threshold);
90
91
92 }

```

● Alarm Manager:

```
C alarm_manager.c > STATE_define(AM_Alarm_OFF)
64
65 /**
66 * @brief at this function the high 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 }
```

```
C alarm_manager.c > STATE_define(AM_Alarm_OFF)
1 /**
2  * @file alarm_manager.c
3  * @author Ahmed khaled khalaaf (khalaafawy22899@gmail.com)
4  * @brief the source file of the alarm manager driver contains the implementation and APIs
5  * @version 0.1
6  * @date 2024-06-22
7  *
8  * @copyright Copyright (c) 2024
9  *
10 */
11
12
13 #include "alarm_manager.h"
14 #include "alarm_actuator.h"
15 #include "Main_Block.h"
16
17
18 /**
19 * @brief initialize the alarm actuator
20 * @note none
21 * @retval none
22 */
23 STATE_define(AM_Init)
24 {
25     //initialize the Alarm manager
26     //state define
27     AM_Status_t = AM_Init;
28
29     //action
30     AM_State = STATE(AM_Alarm_OFF);
31 }
```

```
C alarm_manager.c > STATE_define(AM_Alarm_OFF)
32 /**
33 * @brief At this function the alarm manger shall wait and the alarm actuator 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 }
```

```
C alarm_manager.c > STATE_define(AM_Alarm_ON)
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 }
```

- Alarm actuator:

```
C alarm_actuator.h > ...
1  /**
2   * @file alarm_actuator.h
3   * @author Ahmed khaled khalaif (khalaafawy22899@gmail.com)
4   * @brief This file contains all the prototypes and macros needed to Alarm actuator driver implementation
5   * @version 0.1
6   * @date 2024-06-22
7   *
8   * @copyright Copyright (c) 2024
9   *
10  */
11 #ifndef _ALARM_ACTUATOR_H
12 #define _ALARM_ACTUATOR_H
13 #include "state.h"
14 // Define Status
15 enum {
16     Alarm_Act_Init,
17     Alarm_Act_Waiting,
18     Alarm_Act_ON,
19     Alarm_Act_OFF
20 }Alarm_Act_Status_t ;
21
22 // Declare State Fuctions
23 STATE_define(Alarm_Act_Init);
24 STATE_define(Alarm_Act_ON);
25 STATE_define(Alarm_Act_OFF);
26 STATE_define(Alarm_Act_Waiting);
27 //functions to recive the signal of the alarm actuator
28 extern void AA_Start_Alarm(void);
29 extern void AA_Stop_Alarm(void);
30 // State Pointer to Function
31 void (*Alarm_Act_State)() ;
32
33 #endif
```

```
C alarm_manager.c > STATE_define(AM_Alarm_OFF)
● 1  /**
2   * @file alarm_manager.c
3   * @author Ahmed khaled khalaif (khalaafawy22899@gmail.com)
4   * @brief the source file of the alarm manager driver contains the implementation and APIs
5   * @version 0.1
6   * @date 2024-06-22
7   *
8   * @copyright Copyright (c) 2024
9   *
10  */
11
12
13 #include "alarm_manager.h"
14 #include "alarm_actuator.h"
15 #include "Main_Block.h"
16
17
18 /**
19  * @brief initialize the alarm actuator
20  * @note none
21  * @retval none
22  */
23 STATE_define(AM_Init)
24 {
25     //initialize the Alarm manager
26     //state define
27     AM_Status_t = AM_Init;
28
29     //action
30     AM_State = STATE(AM_Alarm_OFF);
31 }
```

```

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 actuator 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[5000];
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 alarm_manager.c > STATE_define(AM_Alarm_ON)
64 /**
65 * @brief at this function the hogh pressure signal is detected and it will send a signal to trigger the alarm actuator
66 * @note none
67 * @retval none
68 */
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 startup.c > ...
1  /**
2   * @file startup.c
3   * @author your name (you@domain.com)
4   * @brief
5   * @version 0.1
6   * @date 2024-06-18
7   *
8   * @copyright Copyright (c) 2024
9   *
10 */
11
12 #include "STD_DataTypes.h"
13
14 void Reset_Handler();
15 extern int main(void);
16
17 void Default_Handler(void)
18 {
19     Reset_Handler();
20 }
21
22 void NMI_Handler () __attribute__((weak, alias("Default_Handler")));
23 void H_fault_Handler () __attribute__((weak, alias("Default_Handler")));
24
25 //reserve stack size
26
27 static uint_t32 Stack_top[256]; //size = 256*4=1024bytes
28 //=====
29
30 void (* G_P_Fn_Vector[])(() ) __attribute__((section (".vectors")))=
31 {
32     (void (*)()) _((uint_t32)Stack_top + sizeof(Stack_top)),
33     &Reset_Handler,
34     &NMI_Handler,
35     &H_fault_Handler
36 };
37
```

```
C startup.c > ...
37 //=====
38
39
40 extern uint_t32 _E_text ;
41 extern uint_t32 _S_DATA ;
42 extern uint_t32 _E_DATA ;
43 extern uint_t32 _S_bss ;
44 extern uint_t32 _E_bss ;
45
46
47 void Reset_Handler()
48 {
49     /*copying the data from flash to RAM*/
50     uint_t32 DATA_Size = (uint_t8 *) &_E_DATA - (uint_t8 *) &_S_DATA;
51     uint_t8* P_SRC = (uint_t8*) &_E_text;
52     uint_t8* P_dst = (uint_t8*) &_S_DATA;
53
54     for (uint_t32 i = 0; i < DATA_Size; i++)
55     {
56         *((uint_t8*)P_dst++) = *((uint_t8*)P_SRC++);
57     }
58
59     // init .bss section in SRAM = 0
60
61     uint_t32 bss_Size = (uint_t8 *) &_E_bss - (uint_t8 *) &_S_bss;
62     P_dst = (uint_t8 *) &_S_bss;
63
64         for (int i = 0; i < bss_Size; i++)
65         {
66             *((uint_t8 *)P_dst++) = (uint_t8) 0;
67         }
68         main();
69
70
71 }
72 }
```

- Linker script:

```
█ linkerscript.ld
 1  /* linker script CortexM3
 2  Author: Ahmed Khaled Khalaf
 3  */
 4
 5
 6 MEMORY
 7 {
 8     FLASH(RX) : ORIGIN = 0x08000000, LENGTH = 128K
 9     SRAM(RWX) : ORIGIN = 0x20000000, LENGTH = 20K
10 }
11
12
13 SECTIONS
14 {
15     .text : {
16         *(.vectors*)
17         *(.text*)
18         *(.rodata)
19         _E_text = . ;
20     }>FLASH
21
22     .data : {
23         _S_DATA = .;
24         *(.data*)
25         . = ALIGN(4);
26         _E_DATA = .;
27
28     }>SRAM AT> FLASH
29
30     .bss : {
31         _S_bss = .;
32         *(.bss*)
33         _E_bss = .;
34         . = ALIGN(4);
35         . = . + 0x1000;
36         Stack_top = .;
37
38     }>SRAM
39
```

- 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_Receiving_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_Status_t
                  0x4      Main_Block.o
Alarm_Act_Status_t
                  0x1      Main_Block.o
AM_Status
                  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_veenear | 0x08000478 | 0x0 |
| .vfp11_veenear | 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 = .

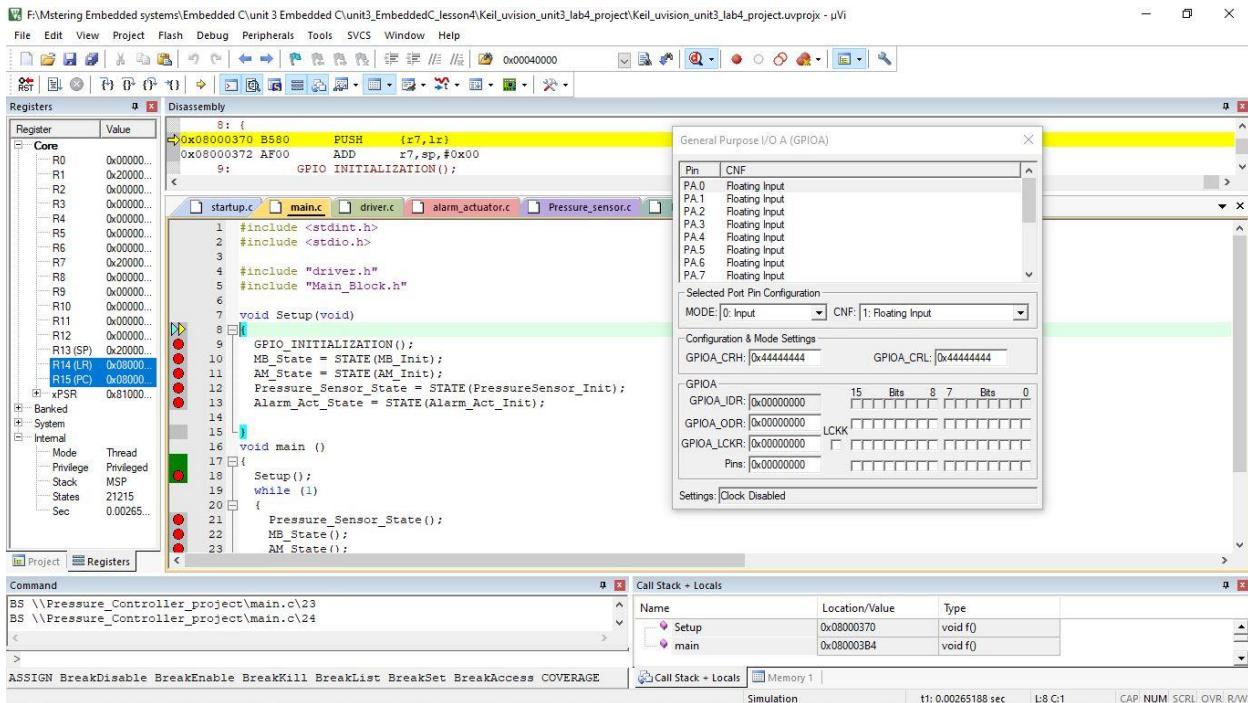
.igot.plt    0x20000004      0x0 load address 0x0800047c
.igot.plt    0x20000004      0x0 Main_Block.o

.bss          0x20000004      0x1422 load address 0x0800047c
              0x20000004      _S_bss = .

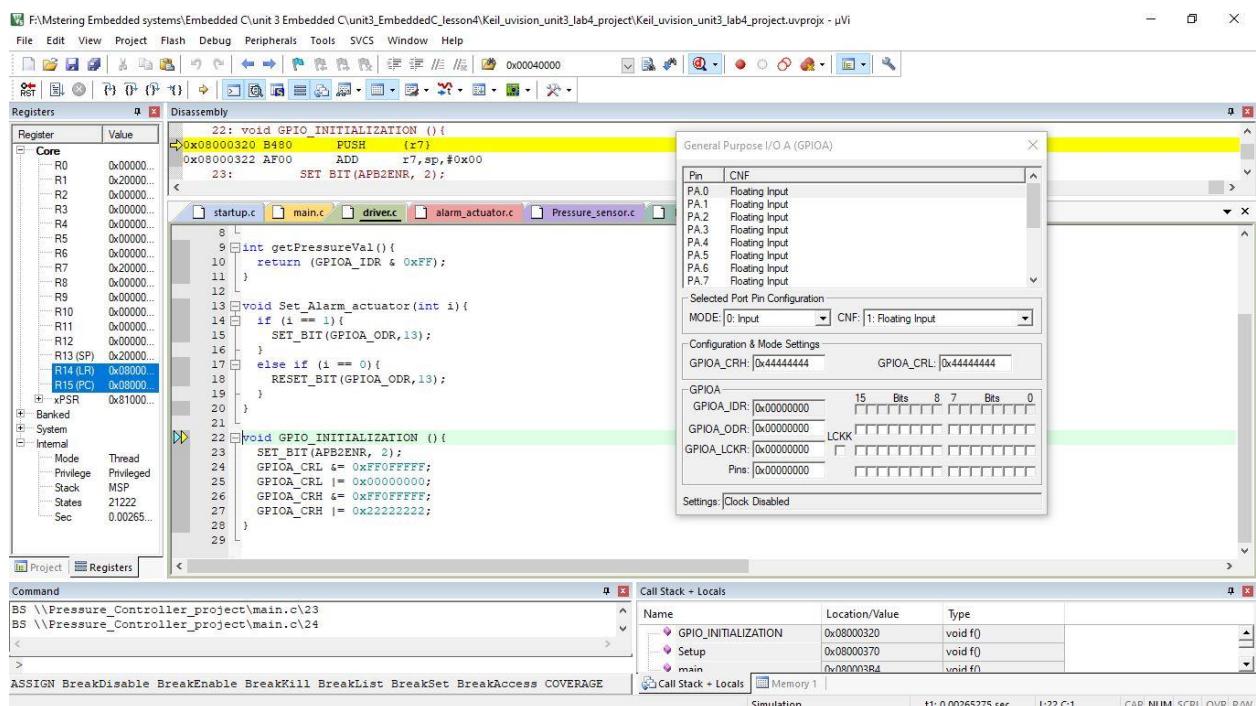
*(.bss*)
.bss          0x20000004      0x4 Main_Block.o
.bss          0x20000008      0x4 Pressure_sensor.o
.bss          0x2000000c      0x0 alarm_actuator.o
.bss          0x2000000c      0x0 alarm_manager.o
.bss          0x2000000c      0x0 driver.o
.bss          0x2000000c      0x0 main.o
.bss          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:



o Main Block initialization:

The screenshot shows the Keil uVision IDE interface with the following details:

- File Path:** F:\Mastering Embedded systems\Embedded C\unit 3 Embedded C\unit3_EMBEDDED_C_Lesson4\Keil_uvision_unit3_lab4_project\Keil_uvision_unit3_lab4_project.uvproj - µVi
- Registers Window:** Shows memory locations R0 to R15 with their corresponding values.
- Disassembly Window:** Displays assembly code for the startup function. The highlighted instruction is `B480 PUSH (r7)`. The assembly code includes comments like `/* @brief initialize the Main block` and `/* @note none`.
- Call Stack + Locals Window:** Shows the call stack with `ST_MB_Init` and `main` as the current frame.
- General Purpose I/O A (GPIOA) Configuration Window:** Provides configuration for GPIOA pins PA0 to PA7. It shows the mode as Input, CNF as Floating Input, and various control register settings (CRH, CRL, IDR, ODR, LCKR).
- Command Window:** Contains the command `ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE`.

o Pressure sensor initialization:

The screenshot shows the Keil uVision IDE interface with the following details:

- File Path:** F:\Mastering Embedded systems\Embedded C\unit 3 Embedded C\unit3_EMBEDDED_C_Lesson4\Keil_uvision_unit3_lab4_project\Keil_uvision_unit3_lab4_project.uvproj - µVi
- Registers Window:** Shows memory locations R0 to R15 with their corresponding values.
- Disassembly Window:** Displays assembly code for the startup function. The highlighted instruction is `B480 PUSH (r7)`. The assembly code includes comments like `/* @brief The startup function which will initialize the pressure sensor` and `/* @note none`.
- Call Stack + Locals Window:** Shows the call stack with `ST_PressureSensor_Init` and `main` as the current frame.
- General Purpose I/O A (GPIOA) Configuration Window:** Provides configuration for GPIOA pins PA0 to PA7. It shows the mode as Input, CNF as Floating Input, and various control register settings (CRH, CRL, IDR, ODR, LCKR).
- Command Window:** Contains the command `ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE`.

○ Pressure sensor initialization:

The screenshot shows the Keil uVision IDE interface. The assembly code in the Disassembly window is as follows:

```

24: {
25:     //initialize the Alarm manager
26:     //state define
27:     PUSH    (r7)
28: 
29:     /* @brief initialize the alarm actuator
30:      * @note none
31:      * @retval none
32:      */
33:     STATE_define(AM_Init)
34: 
35:     //initialize the Alarm manager
36:     //state define
37:     AM_Status_t = AM_Init;
38: 
39:     //action
40:     AM_State = STATE(AM_Alarm_OFF);
41: 
42:     /* @brief At this function the alarm manger shall wait and
43:      * @note it is the idle state of the alarm manager
44:      * @retval none
45:      */
46:     STATE_define(AM_Alarm_WAITING);
47: }

```

The right side of the interface shows the "General Purpose I/O A (GPIOA)" configuration window. It displays pin settings for PA0 to PA7, with PA0 selected as a floating input. The configuration section shows the following settings:

- Selected Port Pin Configuration:** MODE: 0: Input, CNF: 1: Floating Input.
- Configuration & Mode Settings:** GPIOA_CRH: 0x66266666, GPIOA_CRL: 0x44044444.
- GPIOA:** GPIOA_IDR: 0x00000000, GPIOA_ODR: 0x00000000, GPIOA_LCKR: 0x00000000, Pins: 0x00000000.
- Settings:** Clock Enabled.

○ Alarm actuator initialization:

The screenshot shows the Keil uVision IDE interface. The assembly code in the Disassembly window is as follows:

```

21: {
22:     // Initialize The Alarm Actuator
23:     // State define
24:     PUSH    (r7)
25: 
26:     /* @brief initialize the alarm actuator
27:      * @note none
28:      * @retval none
29:      */
30:     STATE_define(Alarm_Act_Init)
31: 
32:     // Initialize The Alarm Actuator
33:     // State define
34:     Alarm_Act_Status_t = Alarm_Act_Init;
35: 
36:     //State Action
37:     // Call The Alarm Actuator Driver Function to update the
38:     // Alarm Act_Status = STATE(Alarm_Act_WAITING);
39: 
40:     STATE_define(Alarm_Act_WAITING)
41: 
42:     //Alarm Act Status t = Alarm Act Waiting;
43: }

```

The right side of the interface shows the "General Purpose I/O A (GPIOA)" configuration window. It displays pin settings for PA0 to PA7, with PA0 selected as a floating input. The configuration section shows the following settings:

- Selected Port Pin Configuration:** MODE: 0: Input, CNF: 1: Floating Input.
- Configuration & Mode Settings:** GPIOA_CRH: 0x66266666, GPIOA_CRL: 0x44044444.
- GPIOA:** GPIOA_IDR: 0x00000000, GPIOA_ODR: 0x00000000, GPIOA_LCKR: 0x00000000, PIns: 0x00000000.
- Settings:** Clock Enabled.

○ pressure sensor reading state

```

42: {
43:     //state define
44:     PUSH    (r7,lr)
45:     ADD    r7,sp,#0x00
46: 
47:     /* @brief At this function the Pressure sensor enter the state
48:      * @param Pressure_Value is a static float type which will
49:      * @note the physical world has floating quantities more
50:      * @retval none
51: 
52: STATE_define(PressureSensor_Reading)
53: 
54: //action
55: Pressure_Value = getPressureVal();
56: 
57: //action
58: //enter the next state
59: Pressure_Sensor_Status_t = STATE(PressureSensor_Idle);
60: 
61: /**
62:  * @brief At this function the Pressure sensor shall wait and return to the reading state
63: */

```

General Purpose I/O A (GPIOA)

| Pin | CNF |
|------|----------------|
| PA.0 | Floating Input |
| PA.1 | Floating Input |
| PA.2 | Floating Input |
| PA.3 | Floating Input |
| PA.4 | Floating Input |
| PA.5 | Analog Input |
| PA.6 | Floating Input |
| PA.7 | Floating Input |

Selected Port Pin Configuration

Mode: 0: Input CNF: 1: Floating Input

Configuration & Mode Settings

GPIOA_CRH: 0x66266666 GPIOA_CRL: 0x44044444

GPIOA

| GPIOA_IDR | 15 Bits | 8 Bits | 7 Bits | 0 |
|------------|---------|--------|--------|-----|
| 0x00000000 | [] | [] | [] | [] |
| GPIOA_ODR | [] | [] | [] | [] |
| GPIOA_LCKR | [] | [] | [] | [] |
| Pins | [] | [] | [] | [] |

Settings: Clock Enabled

○ main block receiving pressure values state:

```

47: {
48:     //state define
49:     PUSH    (r7,lr)
50:     ADD    r7,sp,#0x00
51: 
52:     /* @brief Receiving values from the Pressure sensor
53:      * @note none
54:      * @retval none
55: 
56: STATE_define(MB_Reciving_Values)
57: 
58: //state define
59: MB_Status_t = MB_Reciving_Values;
60: 
61: //receive the current pressure value
62: CP_Value = Pressure_Sensor_Value_Send();
63: 
64: //check if the pressure value high or low
65: if (CP_Value >= P_Threshold )
66: {
67:     MB_State = STATE(MB_HD_detected);
68: }
69: 
70: MB_Status_t = STATE(MB_Reciving_Values);
71: 
72: /**
73:  * @brief At this function the Pressure sensor shall wait and return to the reading state
74: */

```

General Purpose I/O A (GPIOA)

| Pin | CNF |
|------|----------------|
| PA.0 | Floating Input |
| PA.1 | Floating Input |
| PA.2 | Floating Input |
| PA.3 | Floating Input |
| PA.4 | Floating Input |
| PA.5 | Analog Input |
| PA.6 | Floating Input |
| PA.7 | Floating Input |

Selected Port Pin Configuration

Mode: 0: Input CNF: 1: Floating Input

Configuration & Mode Settings

GPIOA_CRH: 0x66266666 GPIOA_CRL: 0x44044444

GPIOA

| GPIOA_IDR | 15 Bits | 8 Bits | 7 Bits | 0 |
|------------|---------|--------|--------|-----|
| 0x00000000 | [] | [] | [] | [] |
| GPIOA_ODR | [] | [] | [] | [] |
| GPIOA_LCKR | [] | [] | [] | [] |
| Pins | [] | [] | [] | [] |

Settings: Clock Enabled

o Alarm manager alarm off state:

```

39: {
40:     //define the state
41:     PUSH    (r7,lr)
42:     ADD     r7,sp,#0x00
43: 
44:     STATE_define(AM_Alarm_OFF)
45: 
46:     //define the state
47:     AM_Status_t = AM_Alarm_OFF;
48:     Delay(50000);
49:     //send alarm actuator stop signal
50:     AA_Stop_Alarm();
51: 
52:     //checking whether the high pressure detected or not
53:     if (HF_detected_signal) == TRUE)
54:     {
55:         //the high pressure signal is detected and update state
56:         AM_Status_t = AM_Alarm_ON;
57:     }
58:     else
59:     {
60:         //the high pressure signal is not detected and update state
61:         //remain at the state
62:         AM_Status_t = AM_Alarm_OFF;
63:     }
64: 
65:     //sending alarm actuator off signal

```

General Purpose I/O A (GPIOA)

| Pin | CNF |
|------|----------------|
| PA.0 | Floating Input |
| PA.1 | Floating Input |
| PA.2 | Floating Input |
| PA.3 | Floating Input |
| PA.4 | Floating Input |
| PA.5 | Analog Input |
| PA.6 | Floating Input |
| PA.7 | Floating Input |

Selected Port Pin Configuration

MODE: 0: Input CNF: 1: Floating Input

Configuration & Mode Settings

GPIOA_CRH: 0x66266666 GPIOA_CRL: 0x44044444

GPIOA

| | | | | |
|------------------------|-----------|------------------|--------|---|
| GPIOA_IDR: 0x00000000 | 15 Bits | 8 Bits | 7 Bits | 0 |
| GPIOA_ODR: 0x00000000 | LCKK | 1111111111111111 | | |
| GPIOA_LCKR: 0x00000000 | Pins: | 1111111111111111 | | |
| GPIOA_PCR: 0x00000000 | Settings: | Clock Enabled | | |

alarm off

Call Stack + Locals

| Name | Location/Value | Type |
|-----------------|----------------|----------|
| ST_AM_Alarm_OFF | 0x08000244 | void f() |
| main | 0x080003B4 | void f() |

o alarm actuator off state:

```

67: {
68:     // State define
69:     PUSH    (r7,lr)
70:     ADD     r7,sp,#0x00
71: 
72:     /* @brief the idle state of the alarm actuator is to be off
73:      * @note none
74:      * @return none
75:      */
76: 
77:     STATE_define(Alarm_Act_OFF)
78: 
79:     // State define
80:     Alarm_Act_Status_t = Alarm_Act_OFF;
81: 
82:     //state action
83:     //write GPIO turn off the alarm
84:     Set_Alarm_actuator(TRUE);
85:     Alarm_Act_State = STATE(Alarm_Act_Waiting);
86: 
87:     extern void AA_Start_Alarm(void)
88: 
89:     // Update State
90:     Alarm_Act_Status_t = STATE(Alarm_Act_ON) ;

```

General Purpose I/O A (GPIOA)

| Pin | CNF |
|------|----------------|
| PA.0 | Floating Input |
| PA.1 | Floating Input |
| PA.2 | Floating Input |
| PA.3 | Floating Input |
| PA.4 | Floating Input |
| PA.5 | Analog Input |
| PA.6 | Floating Input |
| PA.7 | Floating Input |

Selected Port Pin Configuration

MODE: 0: Input CNF: 1: Floating Input

Configuration & Mode Settings

GPIOA_CRH: 0x66266666 GPIOA_CRL: 0x44044444

GPIOA

| | | | | |
|------------------------|-----------|------------------|--------|---|
| GPIOA_IDR: 0x00000000 | 15 Bits | 8 Bits | 7 Bits | 0 |
| GPIOA_ODR: 0x00000000 | LCKK | 1111111111111111 | | |
| GPIOA_LCKR: 0x00000000 | Pins: | 1111111111111111 | | |
| GPIOA_PCR: 0x00000000 | Settings: | Clock Enabled | | |

Call Stack + Locals

| Name | Location/Value | Type |
|------------------|----------------|----------|
| ST_Alarm_Act_OFF | 0x080001C0 | void f() |
| main | 0x080003B4 | void f() |

- pressure sensor idle state:

```

F:\Mstering Embedded systems\Embedded C\unit 3 Embedded C\unit3_EMBEDDED_C_Lesson4\Keil_uvision_unit3_lab4_project\Keil_uvision_unit3_lab4_project.uvproj - µVI
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers Disassembly
Register Value
Core
R0 0x0000...
R1 0x2000...
R2 0x0000...
R3 0x0000...
R4 0x0000...
R5 0x0000...
R6 0x0000...
R7 0x0000...
R8 0x0000...
R9 0x0000...
R10 0x0000...
R11 0x0000...
R12 0x0000...
R13 (SP) 0x2000...
R14 (LR) 0x08000...
R15 (PC) 0x08000...
+ xPSR 0x1000...
Banked
System
Internal
Mode Thread
Privilege Privileged
Stack MSP
States 582745
Sec 0.07284...
Project Registers
Command
BS \\Pressure_Controller_project\\main.c\23
BS \\Pressure_Controller_project\\main.c\24
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE
Call Stack + Locals
Name Location/Value Type
ST_PressureSensor_Idle 0x08000120 void f()
main 0x080003B4 void f()
Call Stack + Locals Memory 1
Simulation t1: 0.07284325 sec L:72 C:1 CAP NUM SCRL OVR R/W

```

General Purpose I/O A (GPIOA)

| Pin | CNF |
|------|----------------|
| PA.0 | Floating Input |
| PA.1 | Floating Input |
| PA.2 | Floating Input |
| PA.3 | Floating Input |
| PA.4 | Floating Input |
| PA.5 | Analog Input |
| PA.6 | Floating Input |
| PA.7 | Floating Input |

Selected Port Pin Configuration

MODE: [0: Input] CNF: [1: Floating Input]

Configuration & Mode Settings

GPIOA_CRH: 0x66266666 GPIOA_CRL: 0x44044444

GPIOA

| | | | | |
|------------------------|------------------|--------|--------|--------|
| GPIOA_IDR: 0x00002000 | 15 Bits | 8 Bits | 7 Bits | 0 Bits |
| GPIOA_ODR: 0x00002000 | LCKK | [] | [] | [] |
| GPIOA_LCKR: 0x00000000 | Pins: 0x00002000 | [] | [] | [] |

Settings: Clock Enabled

- main block detects high pressure:

```

F:\Mstering Embedded systems\Embedded C\unit 3 Embedded C\unit3_EMBEDDED_C_Lesson4\Keil_uvision_unit3_lab4_project\Keil_uvision_unit3_lab4_project.uvproj - µVI
File Edit View Project Flash Debug Peripherals Tools SVCS Window Help
Registers Disassembly
Register Value
Core
R0 0x0000...
R1 0x2000...
R2 0x0000...
R3 0x0000...
R4 0x0000...
R5 0x0000...
R6 0x0000...
R7 0x0000...
R8 0x0000...
R9 0x0000...
R10 0x0000...
R11 0x0000...
R12 0x0000...
R13 (SP) 0x2000...
R14 (LR) 0x08000...
R15 (PC) 0x08000...
+ xPSR 0x1000...
Banked
System
Internal
Mode Thread
Privilege Privileged
Stack MSP
States 1133081
Sec 0.14163...
Project Registers
Command
IS \\Pressure_Controller_project\\main.c\23
IS \\Pressure_Controller_project\\main.c\24
>
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess COVERAGE
Call Stack + Locals
Name Location/Value Type
ST_MB_Reciving_Values 0x08000034 void f()
main 0x080003B4 void f()
Call Stack + Locals Memory 1
Simulation t1: 0.14163638 sec L:57 C:1 CAP NUM SCRL OVR R/W

```

General Purpose I/O A (GPIOA)

| Pin | CNF |
|------|----------------|
| PA.0 | Floating Input |
| PA.1 | Floating Input |
| PA.2 | Floating Input |
| PA.3 | Floating Input |
| PA.4 | Floating Input |
| PA.5 | Analog Input |
| PA.6 | Floating Input |
| PA.7 | Floating Input |

Selected Port Pin Configuration

MODE: [0: Input] CNF: [1: Floating Input]

Configuration & Mode Settings

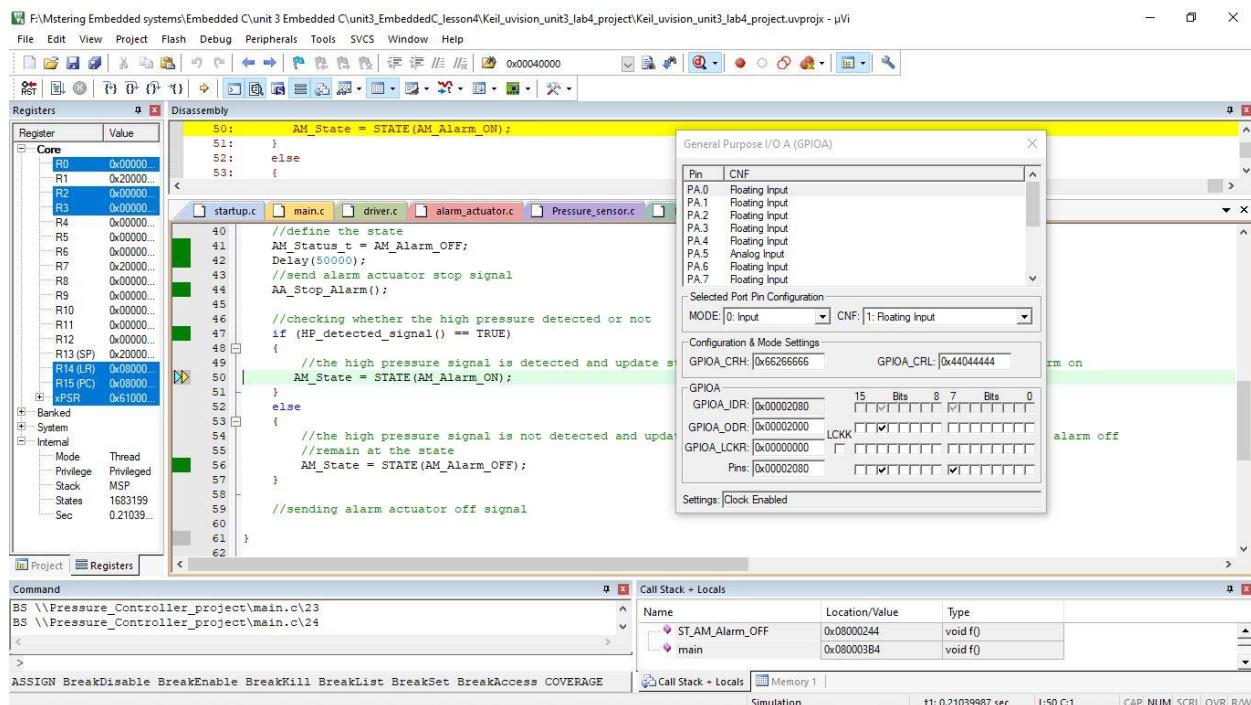
GPIOA_CRH: 0x66266666 GPIOA_CRL: 0x44044444

GPIOA

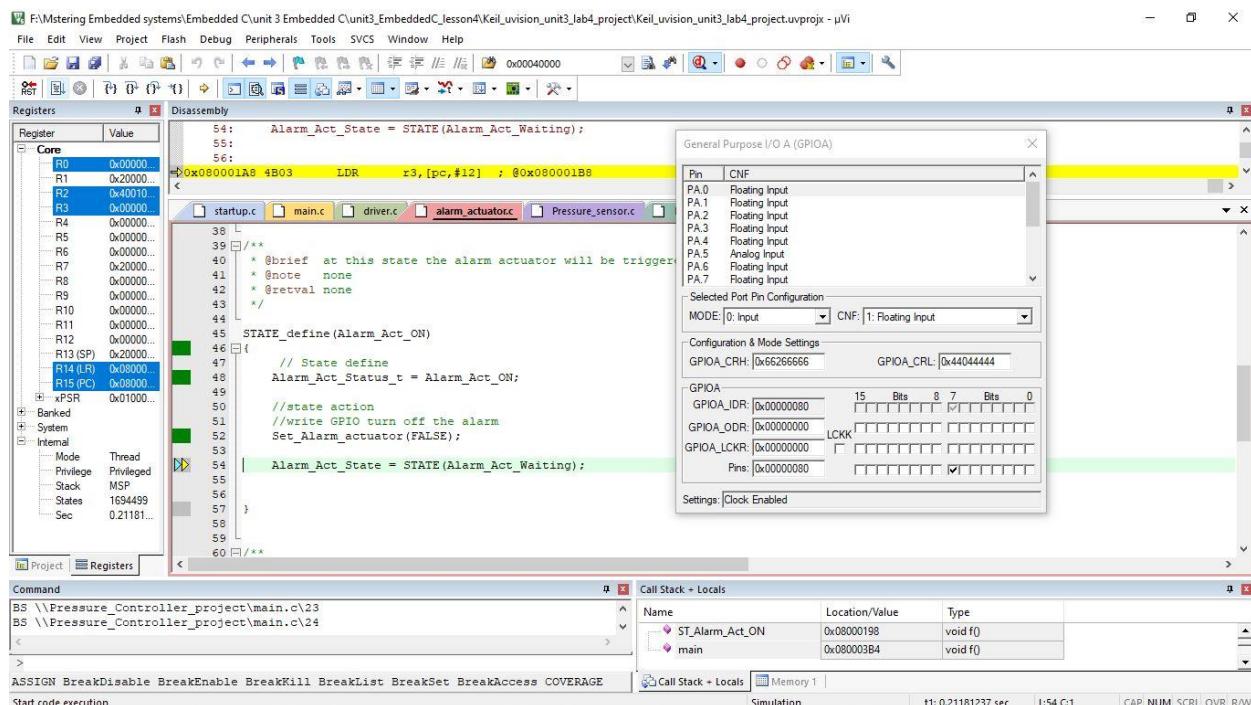
| | | | | |
|------------------------|------------------|--------|--------|--------|
| GPIOA_IDR: 0x00002080 | 15 Bits | 8 Bits | 7 Bits | 0 Bits |
| GPIOA_ODR: 0x00002000 | LCKK | [] | [] | [] |
| GPIOA_LCKR: 0x00000000 | Pins: 0x00002080 | [] | [] | [] |

Settings: Clock Enabled

- 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:

S:\F\Mastering Embedded systems\Embedded C\unit 3 Embedded C\unit3_EMBEDDED_C\lesson4\Keil_uition_unit3\lab4_project\Keil_uition_unit3_lab4_project.uvprojx - μVI

File Edit View Project Flash Debug Peripherals Tools SVCS Window Help

Registers Disassembly

```

74:     Alarm_Act_State = STATE(Alarm_Act_Waiting);
75:
76: 0x080001D0 4B03 LDR r3,[pc,#12] ; 0x080001E0
77: 0x080001D2 4A04 LDR r2,[pc,#16] ; 0x080001E4
78:
79: /*
80:  * @brief the idle state of the alarm actuator is to be off
81:  * @note none
82:  * @retval none
83:  */
84:
85: STATE_define(Alarm_Act_OFF)
86:
87: // State define
88: Alarm_Act_Status_t = Alarm_Act_OFF;
89:
90: //state action
91: //write GPIO turn off the alarm
92: Set_Alarm_actuator(TRUE);
93: Alarm_Act_State = STATE(Alarm_Act_Waiting);
94:
95: }
96:
97: extern void AA_Start_Alarm(void)
98:
99: // Update State
100:
101:
102:
103:
104:
105:
106:
107:
108:
109:
110:
111:
112:
113:
114:
115:
116:
117:
118:
119:
120:
121:
122:
123:
124:
125:
126:
127:
128:
129:
130:
131:
132:
133:
134:
135:
136:
137:
138:
139:
140:
141:
142:
143:
144:
145:
146:
147:
148:
149:
150:
151:
152:
153:
154:
155:
156:
157:
158:
159:
160:
161:
162:
163:
164:
165:
166:
167:
168:
169:
170:
171:
172:
173:
174:
175:
176:
177:
178:
179:
180:
181:
182:
183:
184:
185:
186:
187:
188:
189:
190:
191:
192:
193:
194:
195:
196:
197:
198:
199:
200:
201:
202:
203:
204:
205:
206:
207:
208:
209:
210:
211:
212:
213:
214:
215:
216:
217:
218:
219:
220:
221:
222:
223:
224:
225:
226:
227:
228:
229:
230:
231:
232:
233:
234:
235:
236:
237:
238:
239:
240:
241:
242:
243:
244:
245:
246:
247:
248:
249:
250:
251:
252:
253:
254:
255:
256:
257:
258:
259:
260:
261:
262:
263:
264:
265:
266:
267:
268:
269:
270:
271:
272:
273:
274:
275:
276:
277:
278:
279:
280:
281:
282:
283:
284:
285:
286:
287:
288:
289:
290:
291:
292:
293:
294:
295:
296:
297:
298:
299:
300:
301:
302:
303:
304:
305:
306:
307:
308:
309:
310:
311:
312:
313:
314:
315:
316:
317:
318:
319:
320:
321:
322:
323:
324:
325:
326:
327:
328:
329:
330:
331:
332:
333:
334:
335:
336:
337:
338:
339:
339:
340:
341:
342:
343:
344:
345:
346:
347:
348:
349:
349:
350:
351:
352:
353:
354:
355:
356:
357:
358:
359:
359:
360:
361:
362:
363:
364:
365:
366:
367:
368:
369:
369:
370:
371:
372:
373:
374:
375:
376:
377:
378:
379:
379:
380:
381:
382:
383:
384:
385:
386:
387:
388:
389:
389:
390:
391:
392:
393:
394:
395:
396:
397:
398:
399:
399:
400:
401:
402:
403:
404:
405:
406:
407:
408:
409:
409:
410:
411:
412:
413:
414:
415:
416:
417:
418:
419:
419:
420:
421:
422:
423:
424:
425:
426:
427:
428:
429:
429:
430:
431:
432:
433:
434:
435:
436:
437:
438:
439:
439:
440:
441:
442:
443:
444:
445:
446:
447:
448:
449:
449:
450:
451:
452:
453:
454:
455:
456:
457:
458:
459:
459:
460:
461:
462:
463:
464:
465:
466:
467:
468:
469:
469:
470:
471:
472:
473:
474:
475:
476:
477:
478:
479:
479:
480:
481:
482:
483:
484:
485:
486:
487:
488:
489:
489:
490:
491:
492:
493:
494:
495:
496:
497:
498:
499:
499:
500:
501:
502:
503:
504:
505:
506:
507:
508:
509:
509:
510:
511:
512:
513:
514:
515:
516:
517:
518:
519:
519:
520:
521:
522:
523:
524:
525:
526:
527:
528:
529:
529:
530:
531:
532:
533:
534:
535:
536:
537:
538:
539:
539:
540:
541:
542:
543:
544:
545:
546:
547:
548:
549:
549:
550:
551:
552:
553:
554:
555:
556:
557:
558:
559:
559:
560:
561:
562:
563:
564:
565:
566:
567:
568:
569:
569:
570:
571:
572:
573:
574:
575:
576:
577:
578:
579:
579:
580:
581:
582:
583:
584:
585:
586:
587:
588:
589:
589:
590:
591:
592:
593:
594:
595:
596:
597:
598:
599:
599:
600:
601:
602:
603:
604:
605:
606:
607:
608:
609:
609:
610:
611:
612:
613:
614:
615:
616:
617:
618:
619:
619:
620:
621:
622:
623:
624:
625:
626:
627:
628:
629:
629:
630:
631:
632:
633:
634:
635:
636:
637:
638:
639:
639:
640:
641:
642:
643:
644:
645:
646:
647:
648:
649:
649:
650:
651:
652:
653:
654:
655:
656:
657:
658:
659:
659:
660:
661:
662:
663:
664:
665:
666:
667:
668:
669:
669:
670:
671:
672:
673:
674:
675:
676:
677:
678:
678:
679:
680:
681:
682:
683:
684:
685:
686:
687:
688:
688:
689:
689:
690:
691:
692:
693:
694:
695:
696:
697:
698:
699:
699:
700:
701:
702:
703:
704:
705:
706:
707:
708:
709:
709:
710:
711:
712:
713:
714:
715:
716:
717:
718:
719:
719:
720:
721:
722:
723:
724:
725:
726:
727:
728:
729:
729:
730:
731:
732:
733:
734:
735:
736:
737:
738:
739:
739:
740:
741:
742:
743:
744:
745:
746:
747:
748:
749:
749:
750:
751:
752:
753:
754:
755:
756:
757:
758:
759:
759:
760:
761:
762:
763:
764:
765:
766:
767:
768:
769:
769:
770:
771:
772:
773:
774:
775:
776:
777:
778:
778:
779:
779:
780:
781:
782:
783:
784:
785:
786:
787:
787:
788:
788:
789:
789:
790:
791:
792:
793:
794:
795:
796:
797:
798:
799:
799:
800:
801:
802:
803:
804:
805:
806:
807:
808:
809:
809:
810:
811:
812:
813:
814:
815:
816:
817:
818:
819:
819:
820:
821:
822:
823:
824:
825:
826:
827:
828:
829:
829:
830:
831:
832:
833:
834:
835:
836:
837:
838:
839:
839:
840:
841:
842:
843:
844:
845:
846:
847:
848:
849:
849:
850:
851:
852:
853:
854:
855:
856:
857:
858:
859:
859:
860:
861:
862:
863:
864:
865:
866:
867:
868:
869:
869:
870:
871:
872:
873:
874:
875:
876:
877:
878:
878:
879:
879:
880:
881:
882:
883:
884:
885:
886:
887:
888:
889:
889:
890:
891:
892:
893:
894:
895:
896:
897:
898:
899:
899:
900:
901:
902:
903:
904:
905:
906:
907:
908:
909:
909:
910:
911:
912:
913:
914:
915:
916:
917:
918:
919:
919:
920:
921:
922:
923:
924:
925:
926:
927:
928:
929:
929:
930:
931:
932:
933:
934:
935:
936:
937:
938:
939:
939:
940:
941:
942:
943:
944:
945:
946:
947:
948:
949:
949:
950:
951:
952:
953:
954:
955:
956:
957:
958:
959:
959:
960:
961:
962:
963:
964:
965:
966:
967:
968:
969:
969:
970:
971:
972:
973:
974:
975:
976:
977:
978:
978:
979:
979:
980:
981:
982:
983:
984:
985:
986:
987:
987:
988:
988:
989:
989:
990:
991:
992:
993:
994:
995:
996:
997:
998:
999:
999:
1000:
1000:
1001:
1002:
1003:
1004:
1005:
1006:
1007:
1008:
1009:
1009:
1010:
1011:
1012:
1013:
1014:
1015:
1016:
1017:
1018:
1019:
1019:
1020:
1021:
1022:
1023:
1024:
1025:
1026:
1027:
1028:
1029:
1029:
1030:
1031:
1032:
1033:
1034:
1035:
1036:
1037:
1038:
1039:
1039:
1040:
1041:
1042:
1043:
1044:
1045:
1046:
1047:
1048:
1049:
1049:
1050:
1051:
1052:
1053:
1054:
1055:
1056:
1057:
1058:
1059:
1059:
1060:
1061:
1062:
1063:
1064:
1065:
1066:
1067:
1068:
1069:
1069:
1070:
1071:
1072:
1073:
1074:
1075:
1076:
1077:
1078:
1078:
1079:
1079:
1080:
1081:
1082:
1083:
1084:
1085:
1086:
1087:
1087:
1088:
1088:
1089:
1089:
1090:
1091:
1092:
1093:
1094:
1095:
1095:
1096:
1096:
1097:
1097:
1098:
1099:
1099:
1100:
1100:
1101:
1101:
1102:
1103:
1104:
1105:
1106:
1107:
1108:
1109:
1109:
1110:
1111:
1112:
1113:
1114:
1115:
1116:
1117:
1118:
1119:
1119:
1120:
1121:
1122:
1123:
1124:
1125:
1126:
1127:
1128:
1129:
1129:
1130:
1131:
1132:
1133:
1134:
1135:
1136:
1137:
1138:
1139:
1139:
1140:
1141:
1142:
1143:
1144:
1145:
1146:
1147:
1148:
1148:
1149:
1149:
1150:
1151:
1152:
1153:
1154:
1155:
1156:
1157:
1158:
1159:
1159:
1160:
1161:
1162:
1163:
1164:
1165:
1166:
1167:
1168:
1169:
1169:
1170:
1171:
1172:
1173:
1174:
1175:
1176:
1177:
1178:
1178:
1179:
1179:
1180:
1181:
1182:
1183:
1184:
1185:
1186:
1187:
1187:
1188:
1188:
1189:
1189:
1190:
1191:
1192:
1193:
1194:
1195:
1195:
1196:
1196:
1197:
1197:
1198:
1199:
1199:
1200:
1200:
1201:
1201:
1202:
1203:
1204:
1205:
1206:
1207:
1208:
1209:
1209:
1210:
1211:
1212:
1213:
1214:
1215:
1216:
1217:
1218:
1219:
1219:
1220:
1221:
1222:
1223:
1224:
1225:
1226:
1227:
1228:
1229:
1229:
1230:
1231:
1232:
1233:
1234:
1235:
1236:
1237:
1238:
1239:
1239:
1240:
1241:
1242:
1243:
1244:
1245:
1246:
1247:
1248:
1249:
1249:
1250:
1251:
1252:
1253:
1254:
1255:
1256:
1257:
1258:
1259:
1259:
1260:
1261:
1262:
1263:
1264:
1265:
1266:
1267:
1268:
1269:
1269:
1270:
1271:
1272:
1273:
1274:
1275:
1276:
1277:
1278:
1278:
1279:
1279:
1280:
1281:
1282:
1283:
1284:
1285:
1286:
1287:
1287:
1288:
1288:
1289:
1289:
1290:
1291:
1292:
1293:
1294:
1295:
1295:
1296:
1296:
1297:
1297:
1298:
1299:
1299:
1300:
1300:
1301:
1301:
1302:
1303:
1304:
1305:
1306:
1307:
1308:
1308:
1309:
1309:
1310:
1311:
1312:
1313:
1314:
1315:
1316:
1317:
1318:
1319:
1319:
1320:
1321:
1322:
1323:
1324:
1325:
1326:
1327:
1328:
1329:
1329:
1330:
1331:
1332:
1333:
1334:
1335:
1336:
1337:
1338:
1339:
1339:
1340:
1341:
1342:
1343:
1344:
1345:
1346:
1347:
1348:
1348:
1349:
1349:
1350:
1351:
1352:
1353:
1354:
1355:
1356:
1357:
1358:
1359:
1359:
1360:
1361:
1362:
1363:
1364:
1365:
1366:
1367:
1368:
1369:
1369:
1370:
1371:
1372:
1373:
1374:
1375:
1376:
1377:
1378:
1378:
1379:
1379:
1380:
1381:
1382:
1383:
1384:
1385:
1386:
1387:
1387:
1388:
1388:
1389:
1389:
1390:
1391:
1392:
1393:
1394:
1395:
1396:
1397:
1398:
1398:
1399:
1399:
1400:
1401:
1402:
1403:
1404:
1405:
1406:
1407:
1408:
1408:
1409:
1409:
1410:
1411:
1412:
1413:
1414:
1415:
1416:
1417:
1418:
1418:
1419:
1419:
1420:
1421:
1422:
1423:
1424:
1425:
1426:
1427:
1428:
1429:
1429:
1430:
1431:
1432:
1433:
1434:
1435:
1436:
1437:
1438:
1439:
1439:
1440:
1441:
1442:
1443:
1444:
1445:
1446:
1447:
1448:
1448:
1449:
1449:
1450:
1451:
1452:
1453:
1454:
1455:
1456:
1457:
1458:
1459:
1459:
1460:
1461:
1462:
1463:
1464:
1465:
1466:
1467:
1468:
1469:
1469:
1470:
1471:
1472:
1473:
1474:
1475:
1476:
1477:
1478:
1478:
1479:
1479:
1480:
1481:
1482:
1483:
1484:
1485:
1486:
1487:
1487:
1488:
1488:
1489:
1489:
1490:
1491:
1492:
1493:
1494:
1495:
1496:
1497:
1497:
1498:
1498:
1499:
1499:
1500:
1501:
1502:
1503:
1504:
1505:
1506:
1507:
1508:
1508:
1509:
1509:
1510:
1511:
1512:
1513:
1514:
1515:
1516:
1517:
1518:
1519:
1519:
1520:
1521:
1522:
1523:
1524:
1525:
1526:
1527:
1528:
1529:
1529:
1530:
1531:
1532:
1533:
1534:
1535:
1536:
1537:
1538:
1539:
1539:
1540:
1541:
1542:
1543:
1544:
1545:
1546:
1547:
1548:
1548:
1549:
1549:
1550:
1551:
1552:
1553:
1554:
1555:
1556:
1557:
1558:
1559:
1559:
1560:
1561:
1562:
1563:
1564:
1565:
1566:
1567:
1568:
1569:
1569:
1570:
1571:
1572:
1573:
1574:
1575:
1576:
1577:
1578:
1578:
1579:
1579:
1580:
1581:
1582:
1583:
1584:
1585:
1586:
1587:
1587:
1588:
1588:
1589:
1589:
1590:
1591:
1592:
1593:
1594:
1595:
1596:
1597:
1598:
1598:
1599:
1599:
1600:
1601:
1602:
1603:
1604:
1605:
1606:
1607:
1608:
1608:
1609:
1609:
1610:
1611:
1612:
1613:
1614:
1615:
1616:
1617:
1618:
1619:
1619:
1620:
1621:
1622:
1623:
1624:
1625:
1626:
1627:
1628:
1629:
1629:
1630:
1631:
1632:
1633:
1634:
1635:
1636:
1637:
1638:
1639:
1639:
1640:
1641:
1642:
1643:
1644:
1645:
1646:
1647:
1648:
1648:
1649:
1649:
1650:
1651:
1652:
1653:
1654:
1655:
1656:
1657:
1658:
1659:
1659:
1660:
1661:
1662:
1663:
1664:
1665:
1666:
1667:
1668:
1669:
1669:
1670:
1671:
1672:
1673:
1674:
1675:
1676:
1677:
1678:
1678:
1679:
1679:
1680:
1681:
1682:
1683:
1684:
1685:
1686:
1687:
1687:
1688:
1688:
1689:
1689:
1690:
1691:
1692:
1693:
1694:
1695:
1696:
1697:
1698:
1698:
1699:
1699:
1700:
1701:
1702:
1703:
1704:
1705:
1706:
1707:
1708:
1708:
1709:
1709:
1710:
1711:
1712:
1713:
1714:
1715:
1716:
1717:
1718:
1719:
1719:
1720:
1721:
1722:
1723:
1724:
1725:
1726:
1727:
1728:
1729:
1729:
1730:
1731:
1732:
1733:
1734:
1735:
1736:
1737:
1738:
1739:
1739:
1740:
1741:
1742:
1743:
1744:
1745:
1746:
1747:
1748:
1748:
1749:
1749:
1750:
1751:
1752:
1753:
1754:
1755:
1756:
1757:
1758:
1759:
1759:
1760:
1761:
1762:
1763:
1764:
1765:
1766:
1767:
1768:
1769:
1769:
1770:
1771:
1772:
1773:
1774:
1775:
1776:
1777:
1778:
1778:
1779:
1779:
1780:
1781:
1782:
1783:
1784:
1785:
1786:
1787:
1787:
1788:
1788:
1789:
1789:
1790:
1791:
1792:
1793:
1794:
1795:
1796:
1797:
1797:
1798:
1798:
1799:
1799:
1800:
1801:
1802:
1803:
1804:
1805:
1806:
1807:
1808:
1808:
1809:
1809:
1810:
1811:
1812:
1813:
1814:
1815:
1816:
1817:
1818:
1819:
1819:
1820:
1821:
1822:
1823:
1824:
1825:
1826:
1827:
1828:
1829:
1829:
1830:
1831:
1832:
1833:
1834:
1835:
1836:
1837:
1838:
1839:
1839:
1840:
1841:
1842:
1843:
1844:
1845:
1846:
1847:
1848:
1849:
1849:
1850:
1851:
1852:
1853:
1854:
1855:
1856:
1857:
1858:
1859:
1859:
1860:
1861:
1862:
1863:
1864:
1865:
1866:
1867:
1868:
1869:
1869:
1870:
1871:
1872:
1873:
1874:
1875:
1876:
1877:
1878:
1878:
1879:
1879:
1880:
1881:
1882:
1883:
1884:
1885:
1886:
1887:
1887:
1888:
1888:
1889:
1889:
1890:
1891:
1892:
1893:
1894:
1895:
1896:
1897:
1898:
1898:
1899:
1899:
1900:
1901:
1902:
1903:
1904:
1905:
1906:
1907:
1908:
1908:
1909:
1909:
1910:
1911:
1912:
1913:
1914:
1915:
1916:
1917:
1918:
1919:
1919:
1920:
1921:
1922:
1923:
1924:
1925:
1926:
1927:
1928:
1929:
1929:
1930:
1931:
1932:
1933:
1934:
1935:
1936:
1937:
1938:
1939:
1939:
1940:
1941:
1942:
1943:
1944:
1945:
1946:
1947:
1948:
1949:
1949:
1950:
1951:
1952:
1953:
1954:
1955:
1956:
1957:
1958:
1959:
1959:
1960:
1961:
1962:
1963:
1964:
1965:
1966:
1967:
1968:
1969:
1969:
1970:
1971:
1972:
1973:
1974:
1975:
1976:
1977:
1978:
1978:
1979:
1979:
1980:
1981:
1982:
1983:
1984:
1985:
1986:
1987:
1987:
1988:
1988:
1989:
1989:
1990:
1991:
1992:
1993:
1994:
1995:
1996:
1997:
1998:
1999:
1999:
2000:
2000:
2001:
2002:
2003:
2004:
2005:
2006:
2007:
2008:
2009:
2009:
2010:
2011:
2012:
2013:
2014:
2015:
2016:
2017:
2018:
2019:
2019:
2020:
2021:
2022:
2023:
2024:
2025:
2026:
2027:
2028:
2029:
2029:
2030:
2031:
2032:
2033:
2034:
2035:
2036:
2037:
2038:
2039:
2039:
2040:
2041:
2042:
2043:
2044:
2045:
2046:
2047:
2048:
2049:
2049:
2050:
2051:
2052:
2053:
2054:
2055:
2056:
2057:
2058:
2059:
2059:
2060:
2061:
2062:
2063:
2064:
2065:
2066:
2067:
2068:
2069:
2069:
2070:
2071:
2072:
2073:
2074:
2075:
2076:
2077:
2078:
2078:
2079:
2079:
2080:
2081:
2082:
2083:
2084:
2085:
2086:
2087:
2087:
2088:
2088:
2089:
2089:
2090:
2091:
2092:
2093:
2094:
2095:
2096:
2097:
2098:
2098:
2099:
2099:
2100:
2100:
2101:
2102:
2103:
2104:
2105:
2106:
2107:
2108:
2109:
2109:
2110:
2111:
2112:
2113:
2114:
2115:
2116:
2117:
2118:
2119:
2119:
2120:
2121:
2122:
2123:
2124:
2125:
2126:
2127:
2128:
2129:
2129:
2130:
2131:
2132:
2133:
2134:
2135:
2136:
2137:
2138:
2139:
2139:
2140:
2141:
2142:
2143:
2144:
2145:
2146:
2147:
2148:
2149:
2149:
2150:
2151:
2152:
2153:
2154:
2155:
2156:
2157:
2158:
2159:
2159:
2160:
2161:
2162:
2163:
2164:
2165:
2166:
2167:
2168:
2169:
2169:
2170:
2171:
2172:
2173:
2174:
2175:
2176:
2177:
2178:
2178:
2179:
2179:
2180:
2181:
2182:
2183:
2184:
2185:
2186:
2187:
2187:
2188:
2188:
2189:
2189:
2190:
2191:
2192:
2193:
2194:
2195:
2196:
2197:
2197:
2198:
2198:
2199:
2199:
2200:
2200:
2201:
2202:
2203:
2204:
2205:
2206:
2207:
2208:
2208:
2209:
2209
```

• Simulation:

