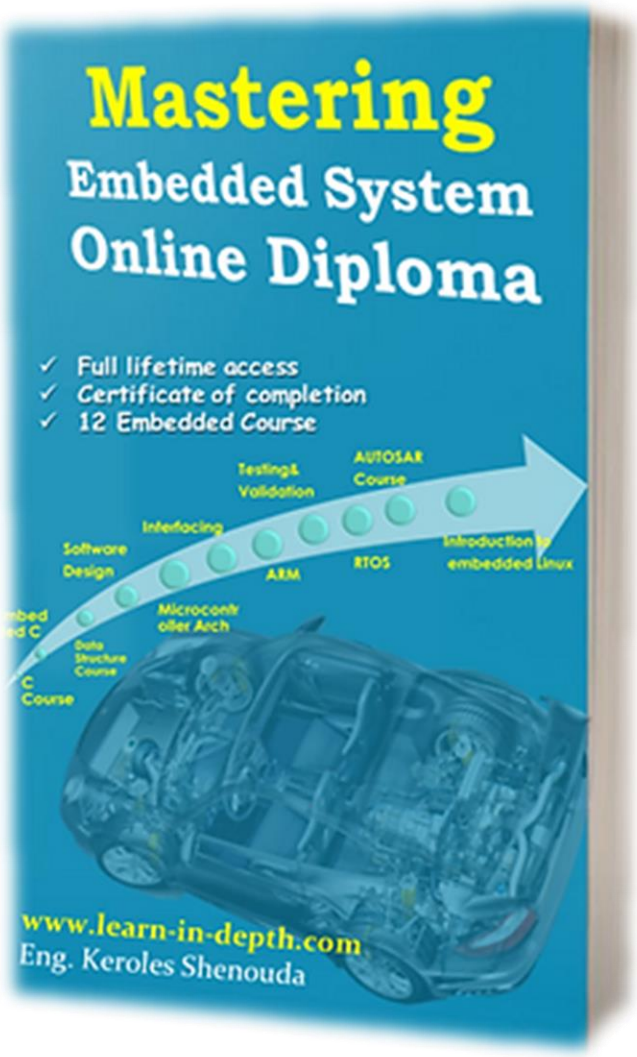

Mastering Embedded System Online Diploma

www.learn-in-depth.com

Pressure Detection System

First Term (Final Project 1)

Eng. Omar Mohamed Yamany

Table of Contents

1. Case Study

- A client expects a software for a system with the following specifications:
    - A pressure controller that informs the cabin's crew with an alarm when the pressure exceeds a pre-defined value of 20 bars
    - The alarm duration is 60 seconds
    - Optional: Keep track of the measured values
- Assumptions:
    - The controller startup and shutdown procedures are not modeled
    - The controller maintenance is not modeled
    - The pressure sensor never fails
    - The alarm never fails
    - The controller never faces power cut

2. Methodology

- Waterfall Method has been chosen for its simplicity.

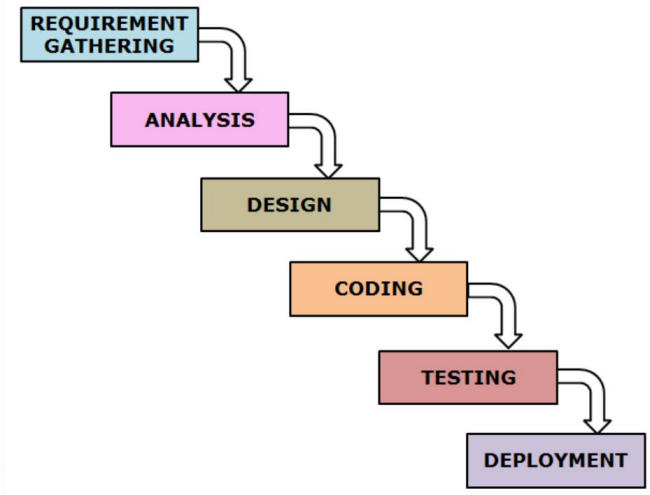

Figure (1) Waterfall Model

3. System Requirements

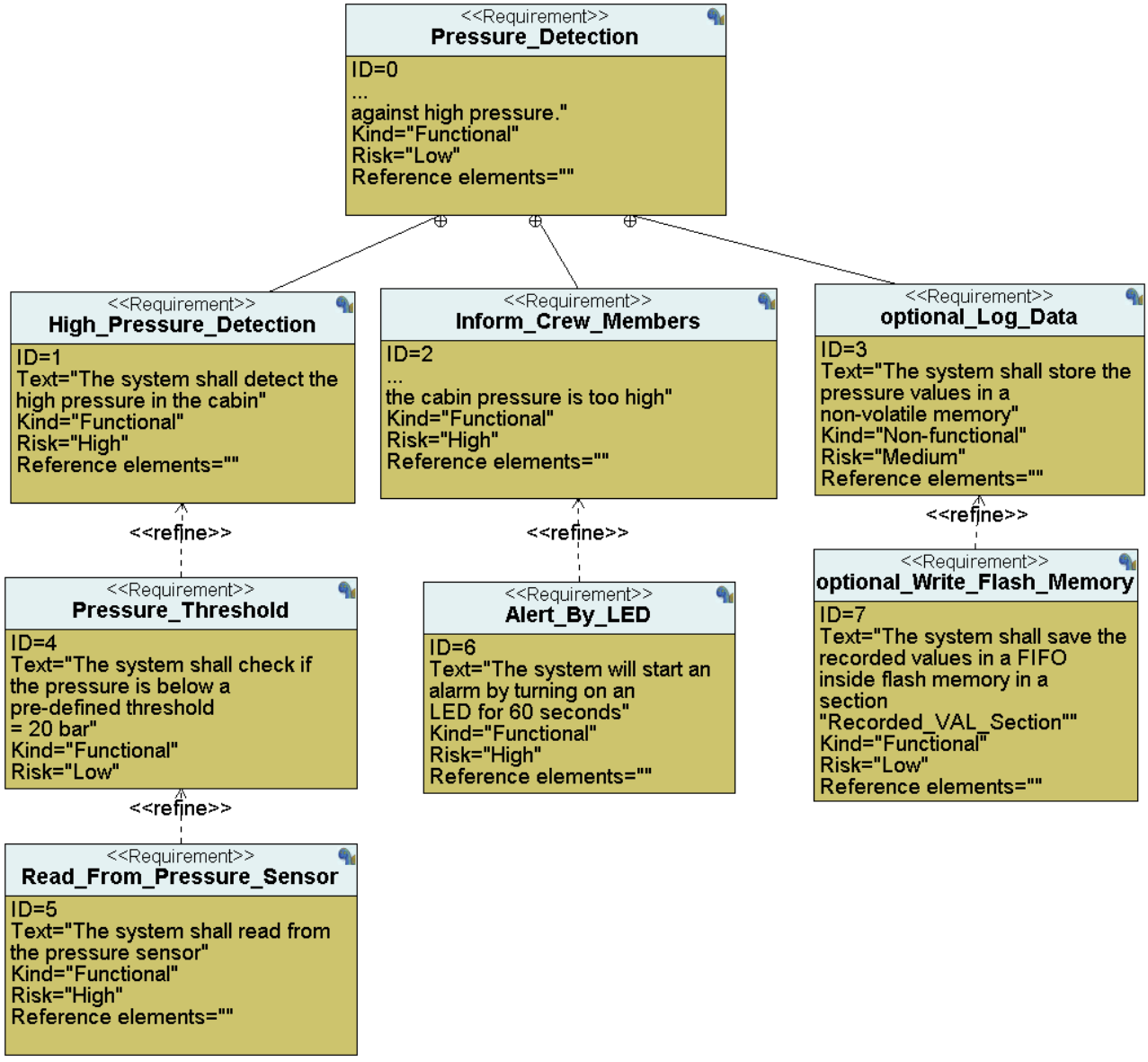

Figure (2) Requirements Diagram

4. System Analysis

- UML Use Case Diagram

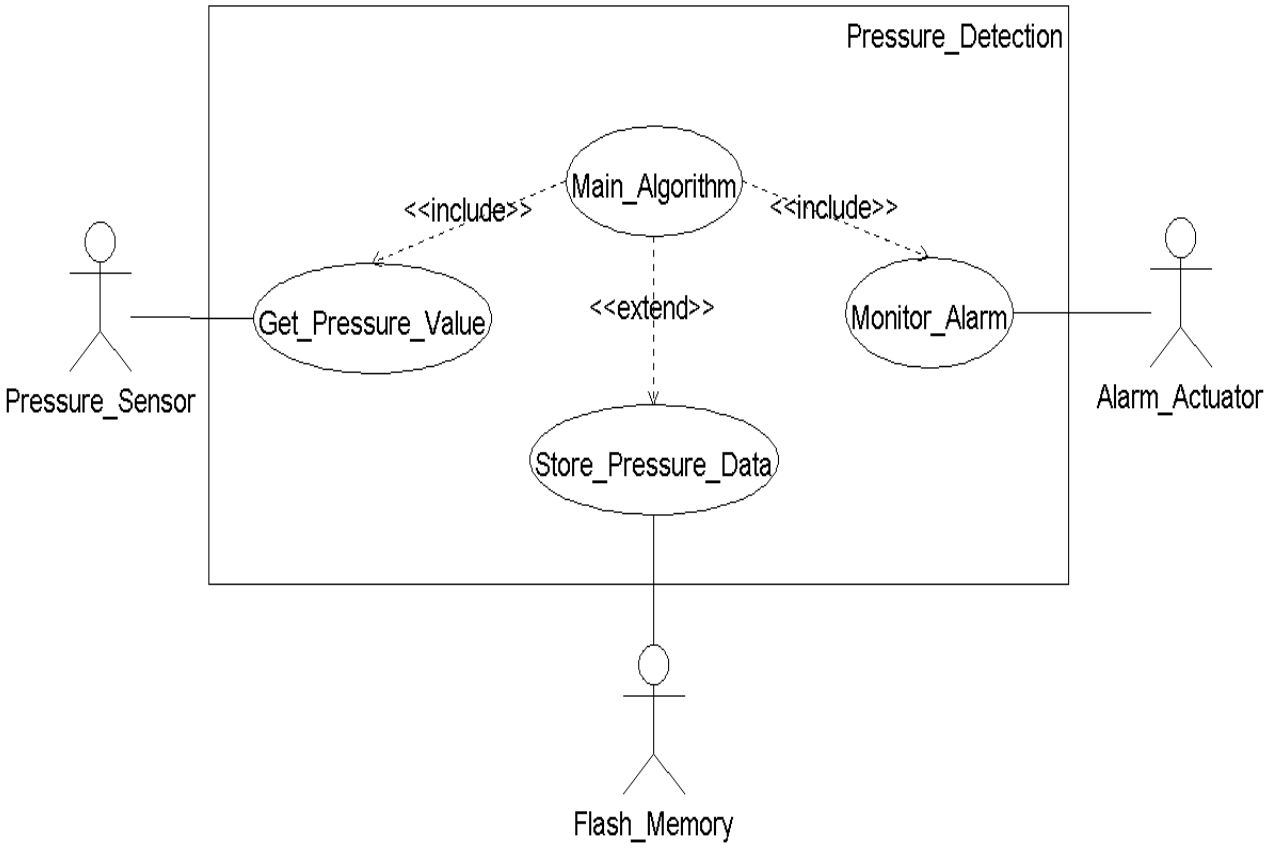

Figure (3) UML Use Case Diagram

- UML Activity Diagram

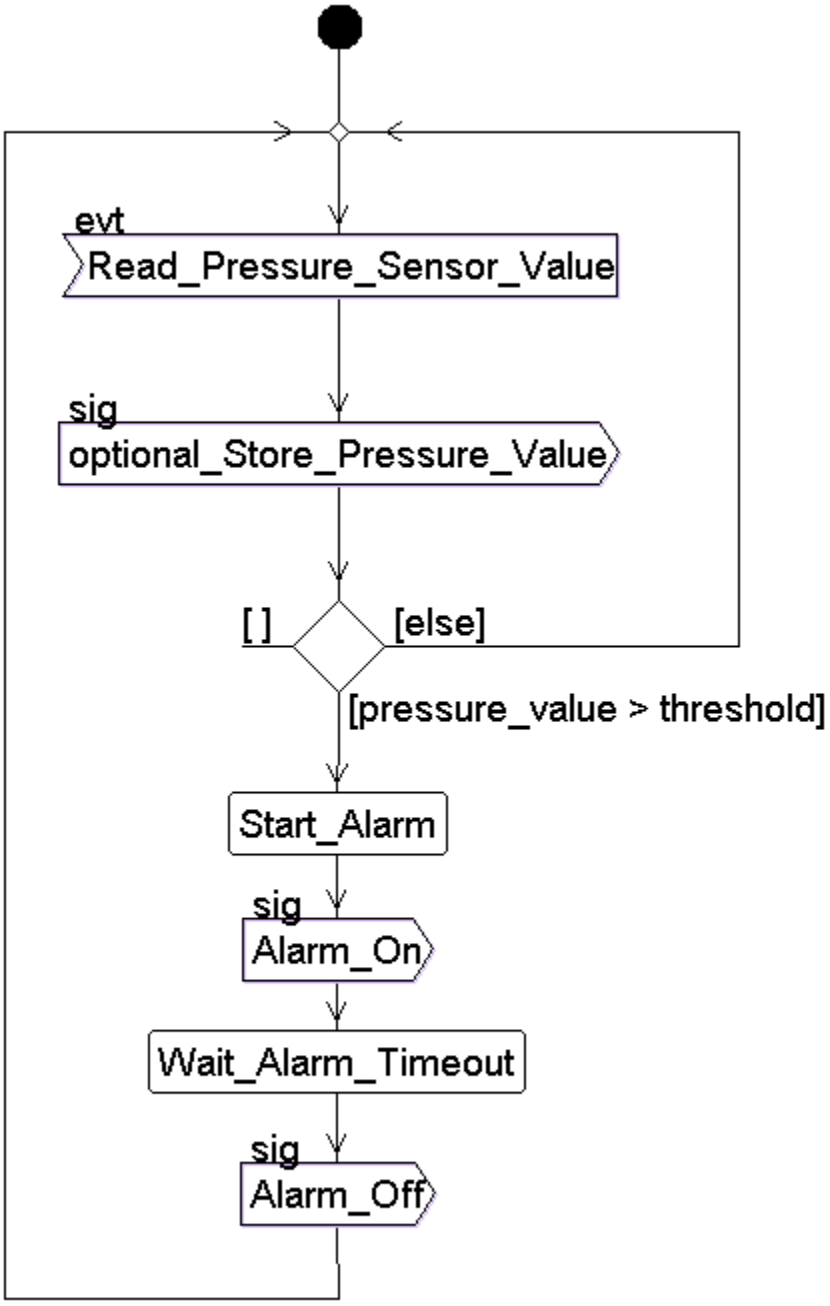

Figure (4) UML Activity Diagram

- UML Sequence Diagram

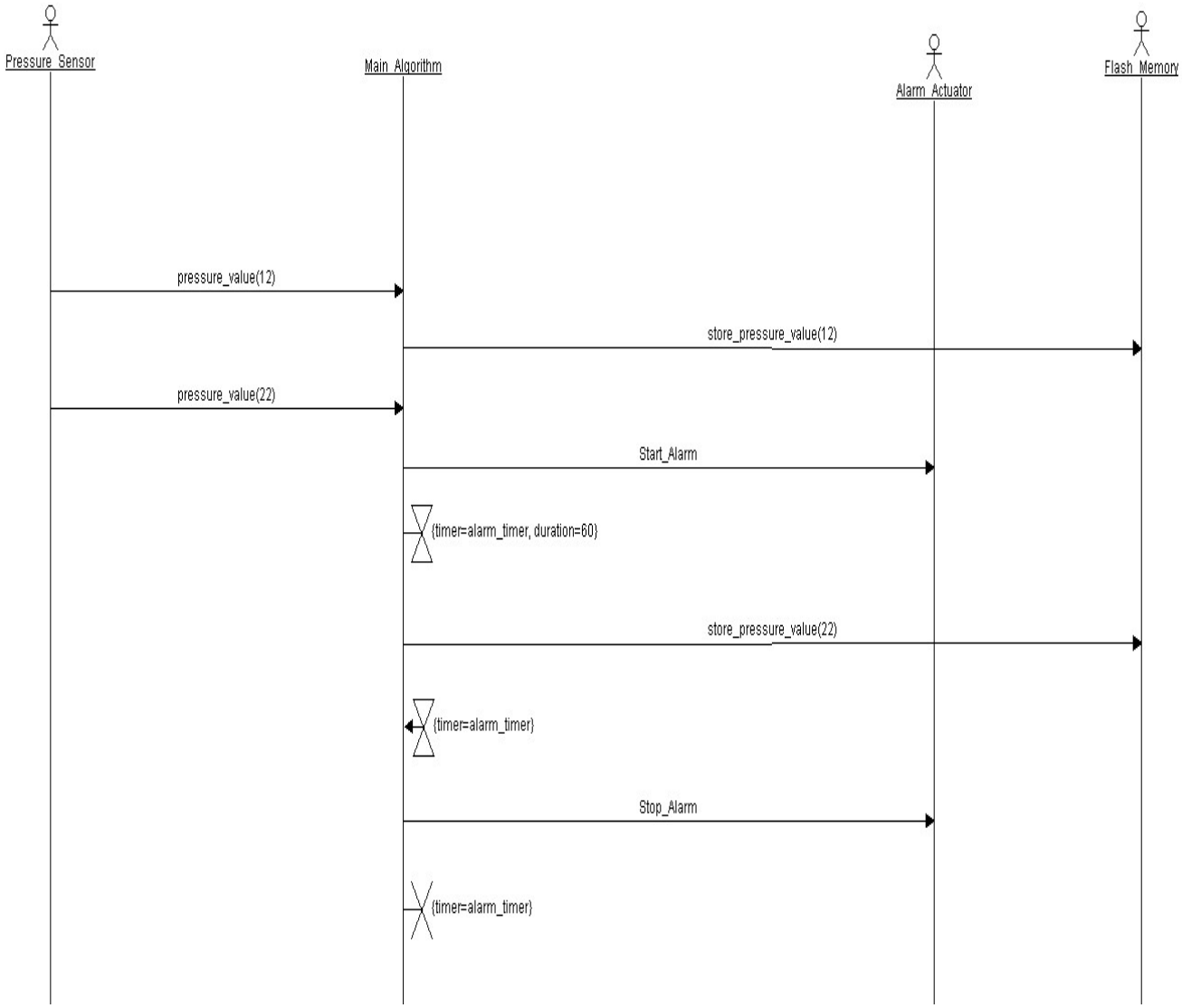

Figure (5) UML Sequence Diagram

5. System Design

- UML Class Diagram

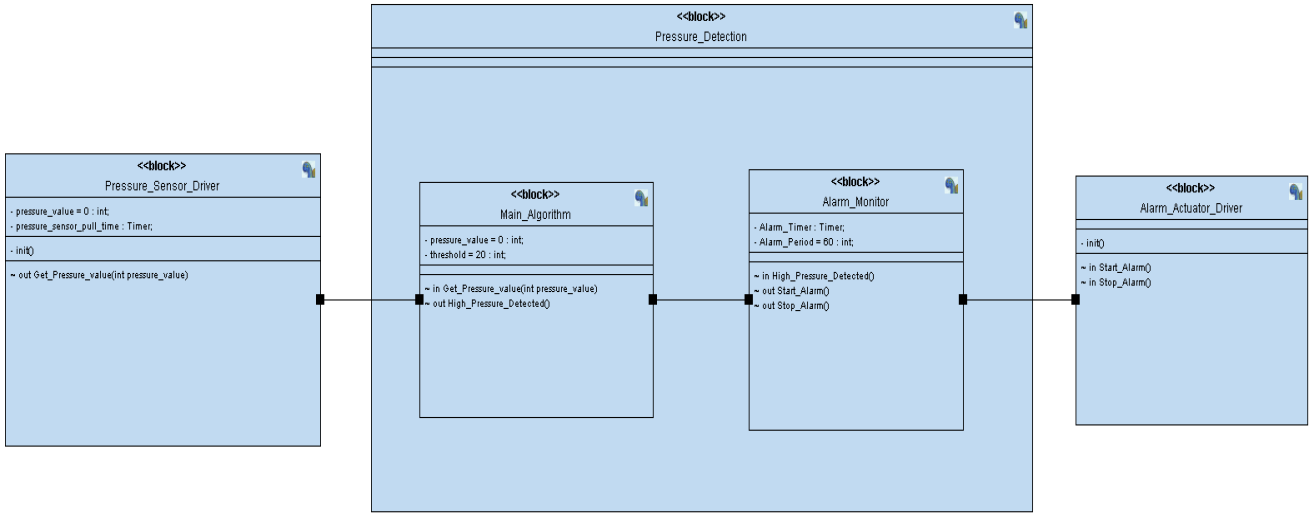

Figure (6) UML Class Diagram

- UML State Diagrams
  - Main Block State Diagram

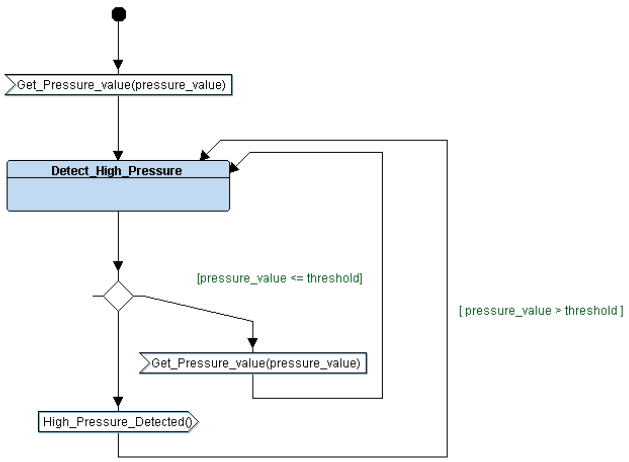

Figure (7) Main Block State Diagram

- Pressure Driver State Diagram

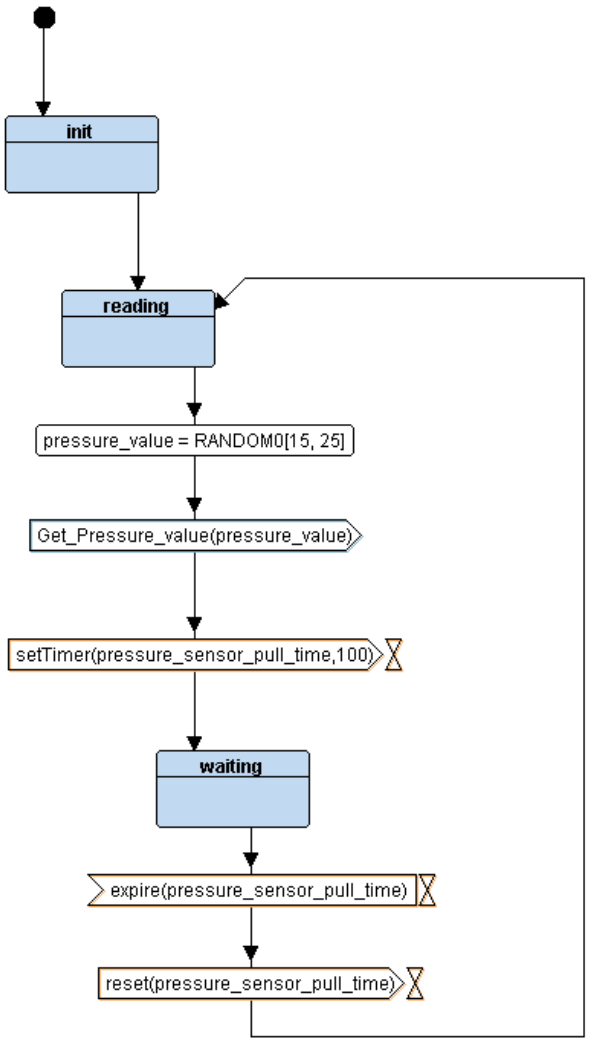

Figure (8) Pressure Driver State Diagram

- Alarm Driver State Diagram

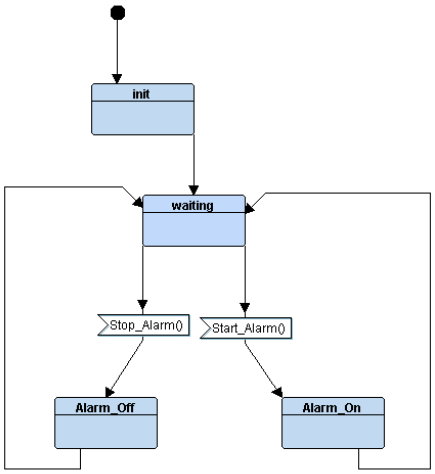

Figure (9) Alarm Driver State Diagram

- Alarm Monitor State Diagram

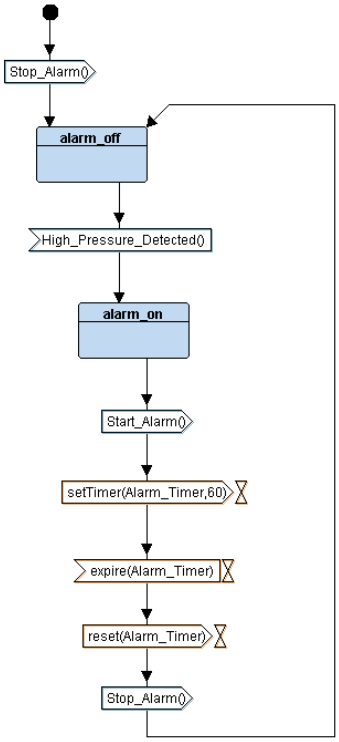

Figure (10) Alarm Monitor State Diagram

- Simulated UML Sequence Diagram

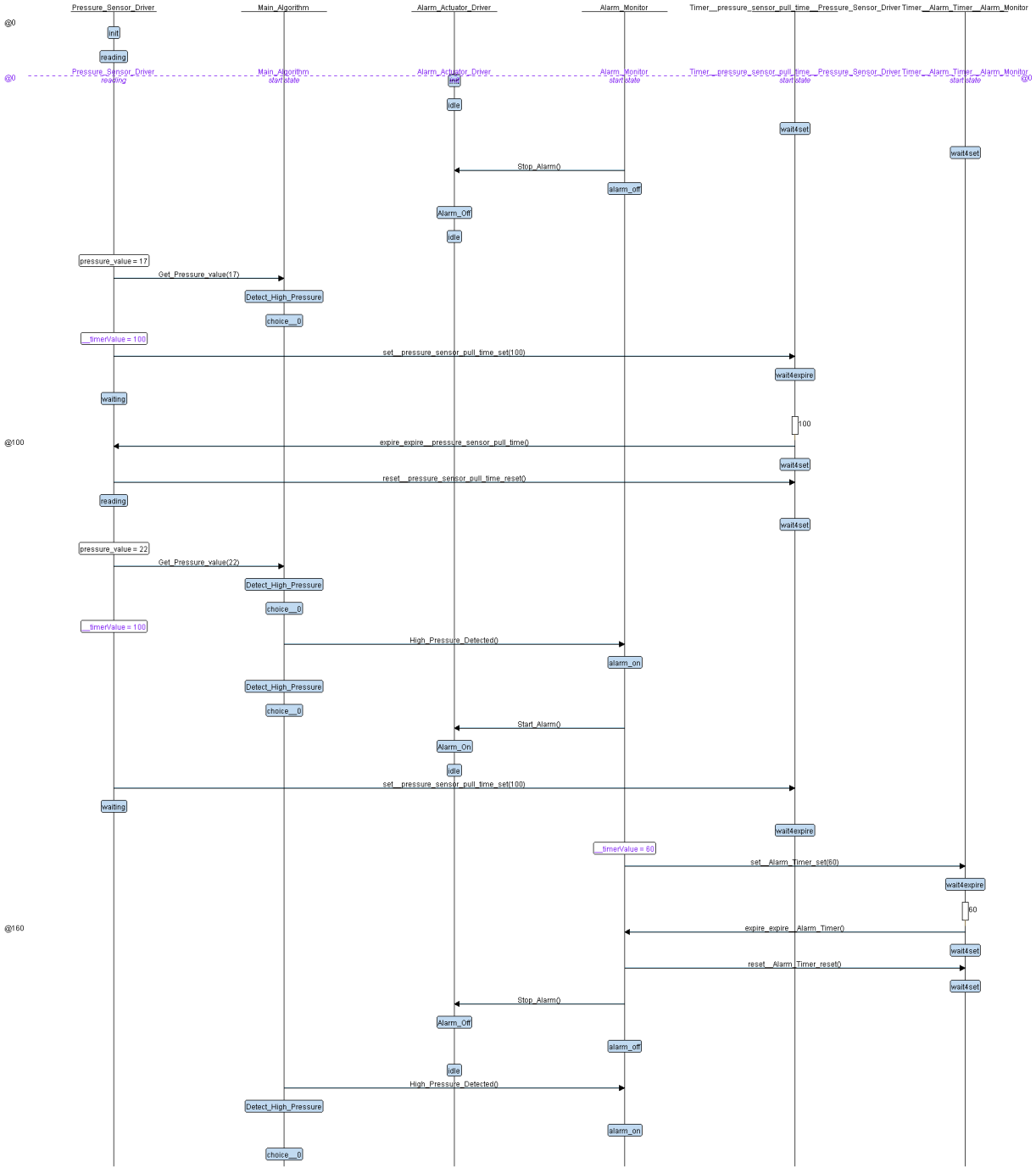

Figure (11) Simulated UML Sequence Diagram

6. Proteus Simulation

- Simulation for case: Pressure is equal to threshold

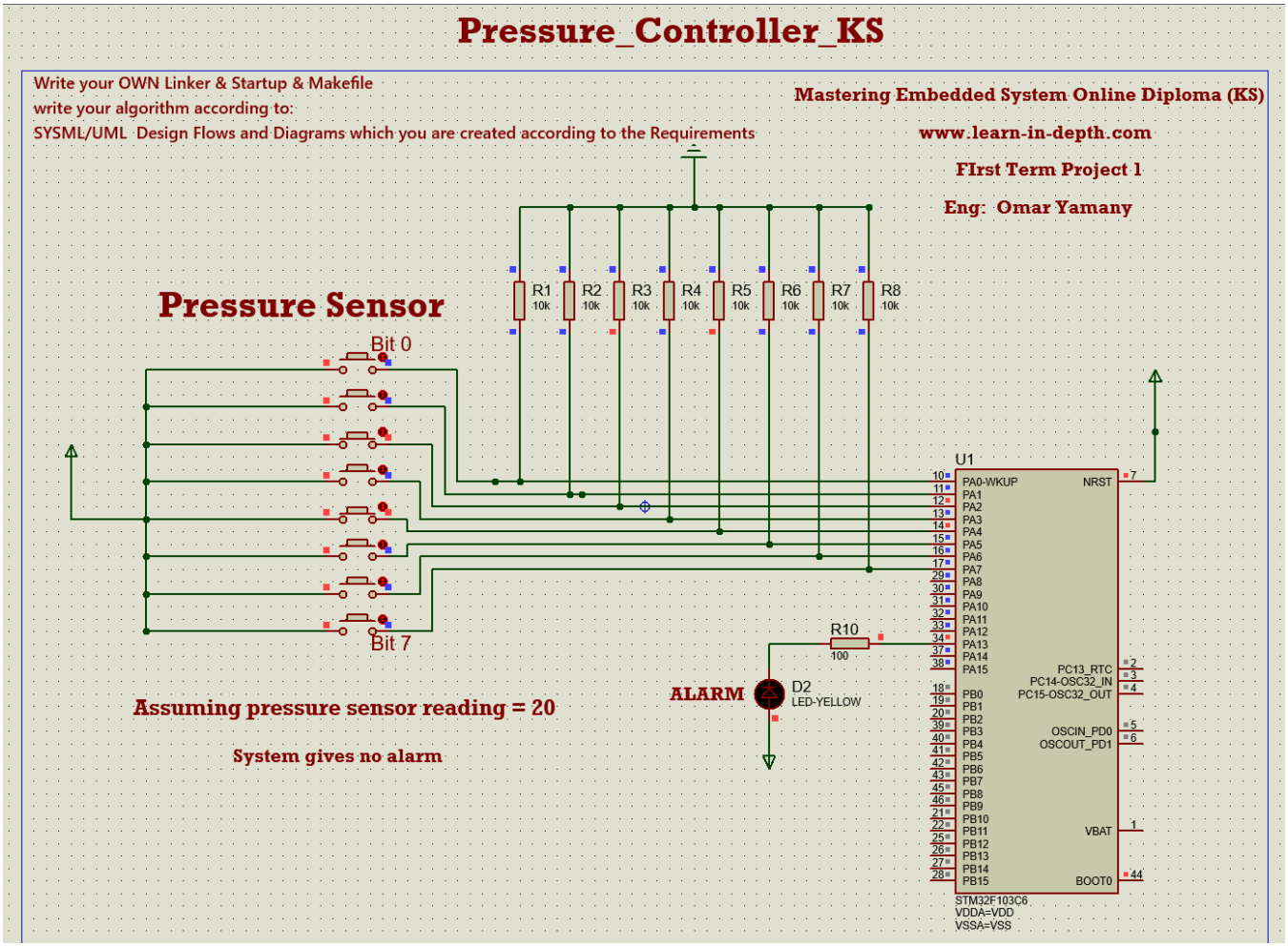

Figure (12) Proteus Simulation No Alarm

- Simulation for case: Pressure is more than the threshold

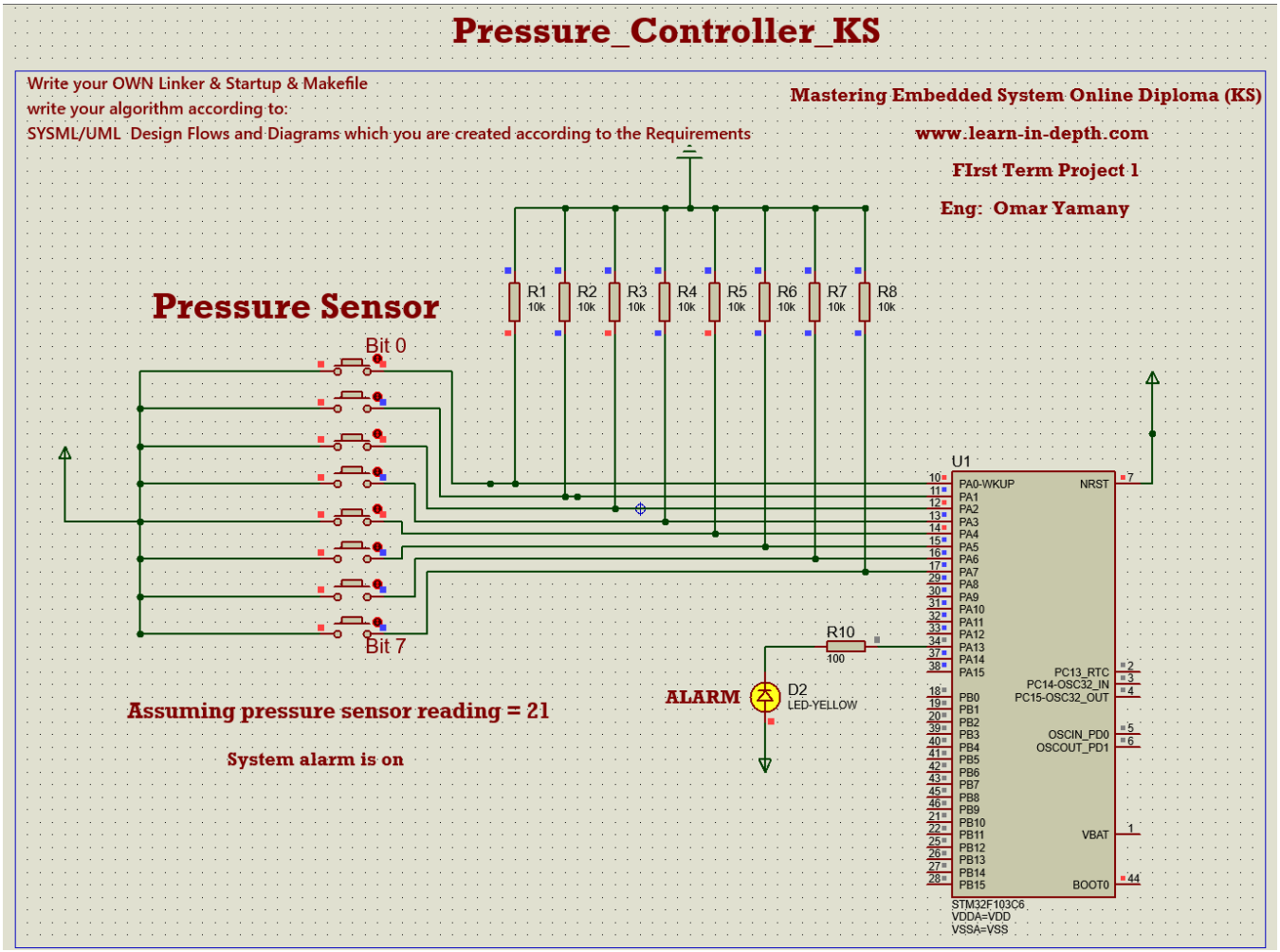

Figure (13) Proteus Simulation with Alarm

7. State machines and their codes

- Main Module

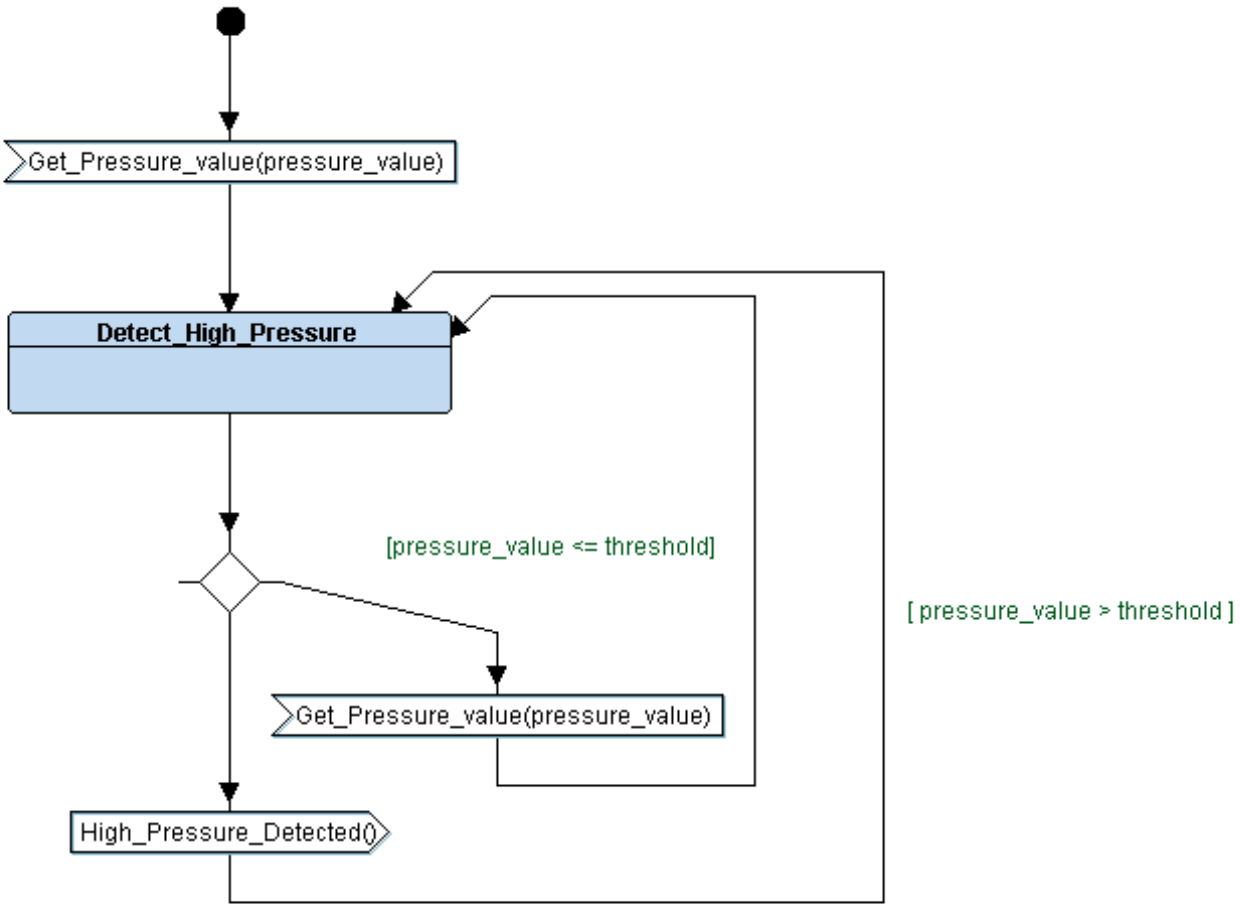

Figure (14) Main Module State Diagram

```c
#include "alarm_driver.h"
#include "alarm_monitor.h"
#include "pressure_driver.h"

#define threshold 20

void system_init();

static int local_pressure_value;

void main(void){
    /* Initialization */
    system_init();

    /* Super loop */
    while(1){
        Pressure_State_Handler();
        if(threshold < local_pressure_value)
            High_Pressure_Detected();
        mAlarm_State_Handler();
        dAlarm_State_Handler();
    }
}

void system_init(){
    Pressure_Init();
    mAlarm_init();
    dAlarm_init();
}

void Update_Pressure_value(int pressure_value){
    local_pressure_value = pressure_value;
}
```

Figure (15) Main Module Source code

- Pressure Sensor Module

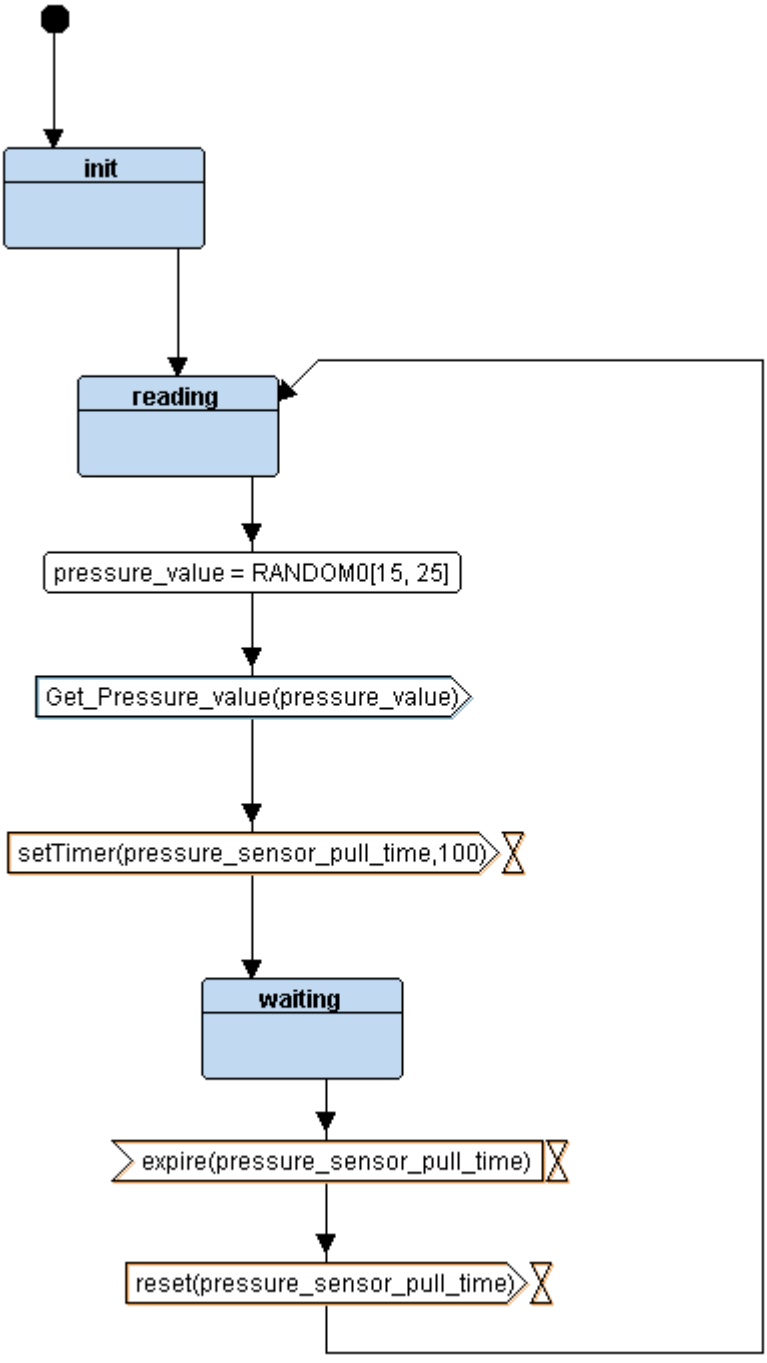

Figure (16) Pressure Sensor Module State Diagram

```c
#include "pressure_driver.h"

void (*Pressure_State_Handler)();
static int pressure_value;
Pressure_States_enu Pressure_Current_State;

void Pressure_Init(){
    GPIO_INITIALIZATION();
    Pressure_State_Handler = STATELBL(Pressure_reading);
}

STATEFUN(Pressure_reading){
    /* State Name */
    Pressure_Current_State = Pressure_reading;

    /* State Action */
    /* Get pressure sensor reading */
    pressure_value = getPressureVal();

    /* State Event */
    /* Set new state */
    Pressure_State_Handler = STATELBL(Pressure_updating);
}

STATEFUN(Pressure_updating){
    /* State Name */
    Pressure_Current_State = Pressure_updating;

    /* State Action */
    /* Update pressure value in main */
    Update_Pressure_value(pressure_value);

    /* State Event */
    /* Start timer for 10 seconds and wait for expiration */
    Delay(100000);

    /* Set new state */
    Pressure_State_Handler = STATELBL(Pressure_reading);

}
```

Figure (17) Pressure Sensor Module Source Code

```c
#ifndef PRESSURE_DRIVER_H_
#define PRESSURE_DRIVER_H_

/* Section: Includes */
#include "states.h"
#include "driver.h"

/* Section: Data Types Declaration */
typedef enum{
    Pressure_reading,
    Pressure_updating
}Pressure_States_enu;

/* Section: Variables declarations */
extern void (*Pressure_State_Handler)();
extern Pressure_States_enu Pressure_Current_State;

/* APIs and states */
void Pressure_Init();

STATEFUN(Pressure_reading);
STATEFUN(Pressure_updating);

#endif /* PRESSURE_DRIVER_H_ */
```

Figure (18) Pressure Sensor Module Header File

- Alarm Actuator Module

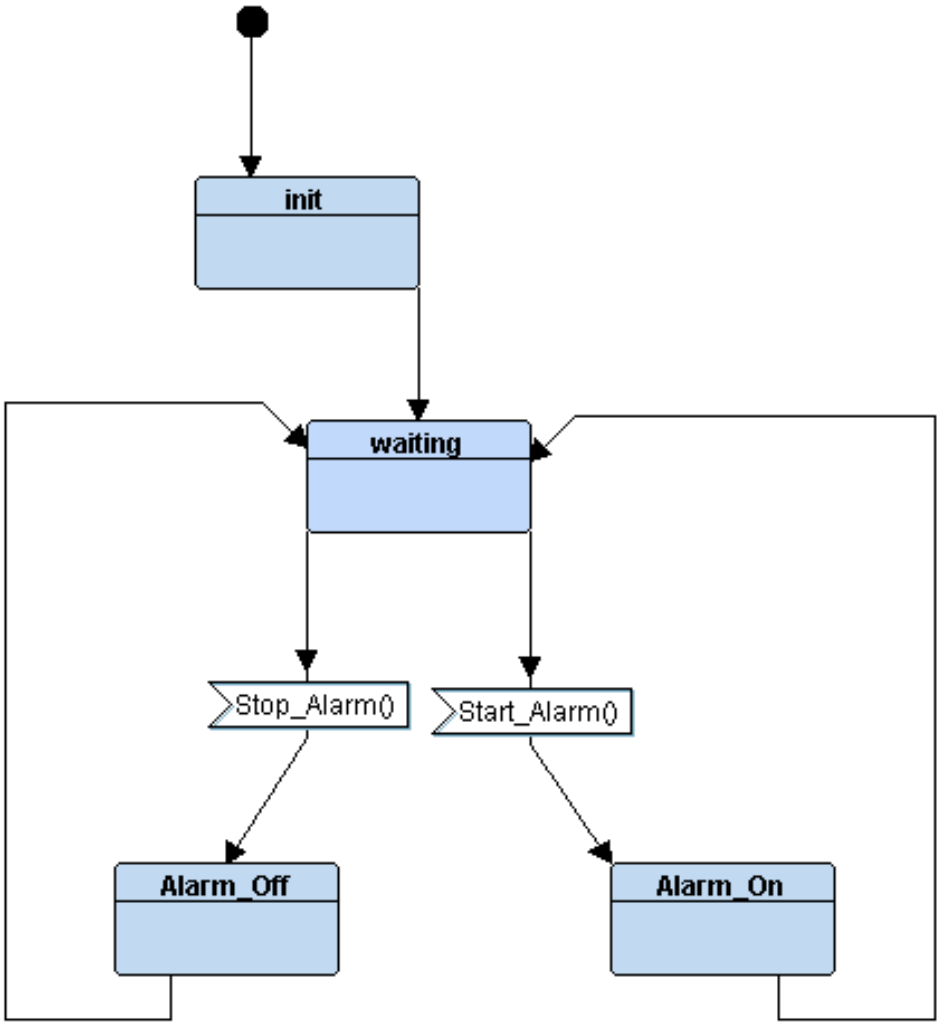

Figure (19) Alarm Actuator Module State Diagram

```c
#include "alarm_driver.h"

void (*dAlarm_State_Handler)();
dAlarm_States_enu dAlarm_Current_State;

void dAlarm_init(){
    GPIO_INITIALIZATION();
    dAlarm_State_Handler = STATELBL(dAlarm_waiting);
}

void Start_Alarm(){
    dAlarm_State_Handler = STATELBL(dAlarm_On);
    dAlarm_State_Handler();
}

void Stop_Alarm(){
    dAlarm_State_Handler = STATELBL(dAlarm_Off);
    dAlarm_State_Handler();
}

STATEFUN(dAlarm_waiting){
    /* State Name */
    dAlarm_Current_State = dAlarm_waiting;

    /* State Action */
}

STATEFUN(dAlarm_On){
    /* State Name */
    dAlarm_Current_State = dAlarm_On;

    /* State Action */
    /* Start Alarm Actuator */
    Set_Alarm_actuator(ALARM_ON);

    /* State Event */
    /* Set new state */
    dAlarm_State_Handler = STATELBL(dAlarm_waiting);
}

STATEFUN(dAlarm_Off){
    /* State Name */
    dAlarm_Current_State = dAlarm_On;

    /* State Action */
    /* Stop Alarm Actuator */
    Set_Alarm_actuator(ALARM_OFF);

    /* State Event */
    /* Set new state */
    dAlarm_State_Handler = STATELBL(dAlarm_waiting);
}
```

Figure (20) Alarm Actuator Module Source Code

```c
#ifndef ALARM_DRIVER_H_
#define ALARM_DRIVER_H_

/* Section: Includes */
#include "states.h"
#include "driver.h"

/* Section: Macro Definitions */
#define ALARM_ON     0
#define ALARM_OFF    1

/* Section: Data Types Declaration */
typedef enum{
    dAlarm_waiting,
    dAlarm_On,
    dAlarm_Off
}dAlarm_States_enu;

/* Section: Variables declarations */
extern void (*dAlarm_State_Handler)();
extern dAlarm_States_enu dAlarm_Current_State;

/* APIs and states */
void dAlarm_init();

STATEFUN(dAlarm_waiting);
STATEFUN(dAlarm_On);
STATEFUN(dAlarm_Off);

#endif /* ALARM_DRIVER_H_ */
```

Figure (21) Alarm Actuator Module Header File

- Alarm Monitor Module

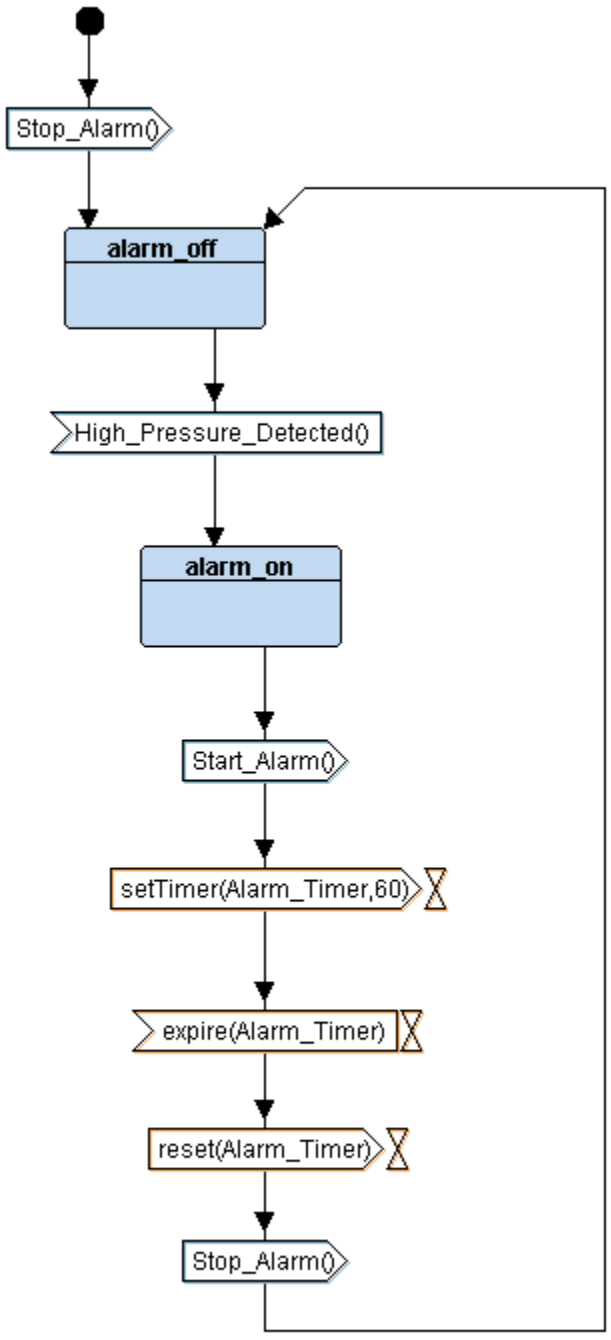

Figure (22) Alarm Monitor Module State Diagram

```c
#include "alarm_monitor.h"

void (*mAlarm_State_Handler)();
mAlarm_States_enu mAlarm_Current_State;

void mAlarm_init(){
    GPIO_INITIALIZATION();
    Stop_Alarm();
    mAlarm_State_Handler = STATELBL(mAlarm_Off);
}

void High_Pressure_Detected(){
    mAlarm_State_Handler = STATELBL(mAlarm_On);
}

STATEFUN(mAlarm_On){
    /* State Name */
    mAlarm_Current_State = mAlarm_On;

    /* State Action */
    /* Start Alarm */
    Start_Alarm();

    /* Start a 60 seconds timer */
    Delay(60000);

    /* State Event */
    /* When timer expires, turn off alarm */
    Stop_Alarm();

    /* Set new state */
    mAlarm_State_Handler = STATELBL(mAlarm_Off);
}

STATEFUN(mAlarm_Off){
    /* State Name */
    mAlarm_Current_State = mAlarm_Off;

    /* State Action */
    /* Do Nothing */
}
```

Figure (23) Alarm Monitor Module Source Code

```c
#ifndef ALARM_MONITOR_H_
#define ALARM_MONITOR_H_

/* Section: Includes */
#include "states.h"
#include "driver.h"

/* Section: Data Types Declaration */
typedef enum{
    mAlarm_On,
    mAlarm_Off
}mAlarm_States_enu;

/* Section: Variables declarations */
extern void (*mAlarm_State_Handler)();
extern mAlarm_States_enu mAlarm_Current_State;

/* APIs and states */
void mAlarm_init();

STATEFUN(mAlarm_On);
STATEFUN(mAlarm_Off);

#endif /* ALARM_MONITOR_H_ */
```

Figure (24) Alarm Monitor Module Header File

8. Software Analysis

- Map File

```
Memory Configuration

Name             Origin             Length             Attributes
FLASH            0x08000000         0x00010000
SRAM             0x20000000         0x00005000
*default*        0x00000000         0xffffffff

Linker script and memory map


.vector         0x08000000      0x150
 *(.vector*)
 .vector_table  0x08000000      0x150 startup.o
                0x08000000                vector_table

.text           0x08000150      0x370
 *(.text*)
 .text          0x08000150       0xbc alarm_driver.o
                0x08000150                dAlarm_init
                0x0800016c                Start_Alarm
                0x08000188                Stop_Alarm
                0x080001a4                ST_dAlarm_waiting
                0x080001bc                ST_dAlarm_On
                0x080001e4                ST_dAlarm_Off
 .text          0x0800020c       0x84 alarm_monitor.o
                0x0800020c                mAlarm_init
                0x0800022c                High_Pressure_Detected
                0x08000248                ST_mAlarm_On
                0x08000278                ST_mAlarm_Off
 .text          0x08000290       0xc4 driver.o
                0x08000290                Delay
                0x080002b2                getPressureVal
                0x080002c8                Set_Alarm_actuator
                0x08000304                GPIO_INITIALIZATION
 .text          0x08000354       0x68 main.o
                0x08000354                main
                0x0800038c                system_init
                0x080003a0                Update_Pressure_value
 .text          0x080003bc       0x84 pressure_driver.o
                0x080003bc                Pressure_Init
                0x080003d8                ST_Pressure_reading
                0x08000408                ST_Pressure_updating
 .text          0x08000440       0x80 startup.o
 *(.rodata*)
                0x080004c0                _E_text = .

.glue_7         0x080004c0        0x0
 .glue_7        0x080004c0        0x0 linker stubs

.glue_7t        0x080004c0        0x0
 .glue_7t       0x080004c0        0x0 linker stubs

.vfp11_veneer   0x080004c0        0x0
 .vfp11_veneer  0x080004c0        0x0 linker stubs

.v4_bx          0x080004c0        0x0
 .v4_bx         0x080004c0        0x0 linker stubs
```

Figure (25) Map File

```
.iplt           0x080004c0        0x0
 .iplt          0x080004c0        0x0 alarm_driver.o

.rel.dyn        0x080004c0        0x0
 .rel.iplt      0x080004c0        0x0 alarm_driver.o

.data           0x20000000        0x0 load address 0x080004c0
                0x20000000                _S_data = .
 *(.data*)
 .data          0x20000000        0x0 alarm_driver.o
 .data          0x20000000        0x0 alarm_monitor.o
 .data          0x20000000        0x0 driver.o
 .data          0x20000000        0x0 main.o
 .data          0x20000000        0x0 pressure_driver.o
 .data          0x20000000        0x0 startup.o
                0x20000000                . = ALIGN (0x4)
                0x20000000                _E_data = .

.igot.plt       0x20000000        0x0 load address 0x080004c0
 .igot.plt      0x20000000        0x0 alarm_driver.o

.bss            0x20000000      0x520 load address 0x080004c0
                0x20000000                _S_bss = .
 *(.bss*)
 .bss           0x20000000        0x5 alarm_driver.o
                0x20000000                dAlarm_State_Handler
                0x20000004                dAlarm_Current_State
 *fill*         0x20000005        0x3
 .bss           0x20000008        0x5 alarm_monitor.o
                0x20000008                mAlarm_State_Handler
                0x2000000c                mAlarm_Current_State
 .bss           0x2000000d        0x0 driver.o
 *fill*         0x2000000d        0x3
 .bss           0x20000010        0x4 main.o
 .bss           0x20000014        0x9 pressure_driver.o
                0x20000014                Pressure_State_Handler
                0x2000001c                Pressure_Current_State
 .bss           0x2000001d        0x0 startup.o
                0x20000020                . = ALIGN (0x4)
 *fill*         0x2000001d        0x3
                0x20000020                _E_bss = .
                0x20000520                . = (. + 0x500)
 *fill*         0x20000020      0x500
                0x20000520                _S_stack = .
LOAD alarm_driver.o
LOAD alarm_monitor.o
LOAD driver.o
LOAD main.o
LOAD pressure_driver.o
LOAD startup.o
OUTPUT(Pressure_Detection.elf elf32-littlearm)
LOAD linker stubs

.debug_info     0x00000000      0x733
 .debug_info    0x00000000      0x170 alarm_driver.o
 .debug_info    0x00000170      0x13e alarm_monitor.o
 .debug_info    0x000002ae      0x112 driver.o
 .debug_info    0x000003c0      0x11b main.o
```

Figure (26) Map File Cont.

```
.debug_info     0x000004db      0x13a pressure_driver.o
.debug_info     0x00000615      0x11e startup.o

.debug_abbrev   0x00000000      0x475
.debug_abbrev   0x00000000       0xbc alarm_driver.o
.debug_abbrev   0x000000bc       0xbc alarm_monitor.o
.debug_abbrev   0x00000178       0xc3 driver.o
.debug_abbrev   0x0000023b       0xc2 main.o
.debug_abbrev   0x000002fd       0xb5 pressure_driver.o
.debug_abbrev   0x000003b2       0xc3 startup.o

.debug_loc      0x00000000      0x4dc
.debug_loc      0x00000000      0x120 alarm_driver.o
.debug_loc      0x00000120       0xe0 alarm_monitor.o
.debug_loc      0x00000200      0x140 driver.o
.debug_loc      0x00000340       0xb4 main.o
.debug_loc      0x000003f4       0x84 pressure_driver.o
.debug_loc      0x00000478       0x64 startup.o

.debug_aranges  0x00000000       0xc0
.debug_aranges
                0x00000000       0x20 alarm_driver.o
.debug_aranges
                0x00000020       0x20 alarm_monitor.o
.debug_aranges
                0x00000040       0x20 driver.o
.debug_aranges
                0x00000060       0x20 main.o
.debug_aranges
                0x00000080       0x20 pressure_driver.o
.debug_aranges
                0x000000a0       0x20 startup.o

.debug_line     0x00000000      0x43e
.debug_line     0x00000000       0x9a alarm_driver.o
.debug_line     0x0000009a       0x81 alarm_monitor.o
.debug_line     0x0000011b      0x14f driver.o
.debug_line     0x0000026a       0x9a main.o
.debug_line     0x00000304       0x83 pressure_driver.o
.debug_line     0x00000387       0xb7 startup.o

.debug_str      0x00000000      0x3b5
.debug_str      0x00000000      0x18c alarm_driver.o
                                0x1e9 (size before relaxing)
.debug_str      0x0000018c       0x8a alarm_monitor.o
                                0x1c9 (size before relaxing)
.debug_str      0x00000216       0x4e driver.o
                                0x18a (size before relaxing)
.debug_str      0x00000264       0x5a main.o
                                0x1bd (size before relaxing)
.debug_str      0x000002be       0x74 pressure_driver.o
                                0x1e7 (size before relaxing)
.debug_str      0x00000332       0x83 startup.o
                                0x145 (size before relaxing)

.comment        0x00000000       0x49
.comment        0x00000000       0x49 alarm_driver.o
                                 0x4a (size before relaxing)
```

Figure (27) Map File Cont.

```
 .comment       0x00000049       0x4a alarm_monitor.o
 .comment       0x00000049       0x4a driver.o
 .comment       0x00000049       0x4a main.o
 .comment       0x00000049       0x4a pressure_driver.o
 .comment       0x00000049       0x4a startup.o

.ARM.attributes
                0x00000000       0x2d
 .ARM.attributes
                0x00000000       0x2d alarm_driver.o
 .ARM.attributes
                0x0000002d       0x2d alarm_monitor.o
 .ARM.attributes
                0x0000005a       0x2d driver.o
 .ARM.attributes
                0x00000087       0x2d main.o
 .ARM.attributes
                0x000000b4       0x2d pressure_driver.o
 .ARM.attributes
                0x000000e1       0x2d startup.o

.debug_frame    0x00000000      0x304
 .debug_frame   0x00000000       0xbc alarm_driver.o
 .debug_frame   0x000000bc       0x88 alarm_monitor.o
 .debug_frame   0x00000144       0xa0 driver.o
 .debug_frame   0x000001e4       0x70 main.o
 .debug_frame   0x00000254       0x64 pressure_driver.o
 .debug_frame   0x000002b8       0x4c startup.o
```

Figure (28) Map File Cont.

- Memory Sections

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

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .vector       00000150  08000000  08000000  00010000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  1 .text         00000370  08000150  08000150  00010150  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  2 .data         00000000  20000000  080004c0  00020000  2**0
                  CONTENTS, ALLOC, LOAD, DATA
  3 .bss          00000520  20000000  080004c0  00020000  2**2
                  ALLOC
  4 .comment      00000049  00000000  00000000  00020000  2**0
                  CONTENTS, READONLY
  5 .ARM.attributes 0000002d  00000000  00000000  00020049  2**0
                  CONTENTS, READONLY
```

Figure (29) Memory Sections

- Linker Symbols

```
$ arm-none-eabi-nm Pressure_Detection.elf
20000020 B _E_bss
20000000 D _E_data
080004c0 T _E_text
20000000 B _S_bss
20000000 D _S_data
20000520 B _S_stack
080004b4 W ADC1_2_Handler
080004b4 W Bus_Fault_Handler
080004b4 W CAN1_RX0_Handler
080004b4 W CAN1_RX1_Handler
080004b4 W CAN1_SCE_Handler
080004b4 W CAN1_TX_Handler
080004b4 W CAN2_RX0_Handler
080004b4 W CAN2_RX1_Handler
080004b4 W CAN2_SCE_Handler
080004b4 W CAN2_TX_Handler
20000004 B dAlarm_Current_State
08000150 T dAlarm_init
20000000 B dAlarm_State_Handler
080004b4 W Debug_Mon_Handler
080004b4 T DefaultInterruptHandler
08000290 T Delay
080004b4 W DMA_CH1_Handler
080004b4 W DMA_CH2_Handler
080004b4 W DMA_CH3_Handler
080004b4 W DMA_CH4_Handler
080004b4 W DMA_CH5_Handler
080004b4 W DMA_CH6_Handler
080004b4 W DMA_CH7_Handler
080004b4 W DMA2_Channel1_Handler
080004b4 W DMA2_Channel2_Handler
080004b4 W DMA2_Channel3_Handler
080004b4 W DMA2_Channel4_Handler
080004b4 W DMA2_Channel5_Handler
080004b4 W ETH_Handler
080004b4 W ETH_WKUP
080004b4 W EXTI0_Handler
080004b4 W EXTI1_Handler
080004b4 W EXTI15_10_Handler
080004b4 W EXTI2_Handler
080004b4 W EXTI3_Handler
080004b4 W EXTI4_Handler
080004b4 W EXTI9_5_Handler
080004b4 W Flash_Handler
080002b2 T getPressureVal
08000304 T GPIO_INITIALIZATION
080004b4 W H_Fault_Handler
0800022c T High_Pressure_Detected
080004b4 W I2C1_ER_Handler
080004b4 W I2C1_EV_Handler
080004b4 W I2C2_ER_Handler
080004b4 W I2C2_EV_Handler
20000010 b local_pressure_value
08000354 T main
2000000c B mAlarm_Current_State
0800020c T mAlarm_init
```

Figure (30) Linker Symbols

```
20000008 B mAlarm_State_Handler
080004b4 W MM_Fault_Handler
080004b4 W NMI_Handler
080004b4 W OTG_FS_Handler
080004b4 W OTG_FS_WKUP_Handler
080004b4 W PendSV_Handler
2000001c B Pressure_Current_State
080003bc T Pressure_Init
20000014 B Pressure_State_Handler
20000018 b pressure_value
080004b4 W PVD_Handler
080004b4 W RCC_Handler
08000440 T Reset_Handler
080004b4 W RTC_Handler
080004b4 W RTCAlarm_Handler
080002c8 T Set_Alarm_actuator
080004b4 W SPI1_Handler
080004b4 W SPI2_Handler
080004b4 W SPI3_Handler
080001e4 T ST_dAlarm_Off
080001bc T ST_dAlarm_On
080001a4 T ST_dAlarm_waiting
08000278 T ST_mAlarm_Off
08000248 T ST_mAlarm_On
080003d8 T ST_Pressure_reading
08000408 T ST_Pressure_updating
0800016c T Start_Alarm
08000188 T Stop_Alarm
080004b4 W SVCall_Handler
0800038c T system_init
080004b4 W SysTick_Handler
080004b4 W Tamper_Handler
080004b4 W TIM1_BRK_Handler
080004b4 W TIM1_CC_Handler
080004b4 W TIM1_TRG_COM_Handler
080004b4 W TIM1_UP_Handler
080004b4 W TIM2_Handler
080004b4 W TIM3_Handler
080004b4 W TIM4_Handler
080004b4 W TIM5_Handler
080004b4 W TIM6_Handler
080004b4 W TIM7_Handler
080004b4 W UART4_Handler
080004b4 W UART5_Handler
080003a0 T Update_Pressure_value
080004b4 W Usage_Fault_Handler
080004b4 W USART1_Handler
080004b4 W USART2_Handler
080004b4 W USART3_Handler
08000000 D vector_table
080004b4 W Window_WD_Handler
```

Figure (31) Linker Symbols Cont.