

[Click_Here](#)



[Learn_In_Depth](#)

First Term (Final Project 1)

Pressure Controller Project

Name : Michael Maged William

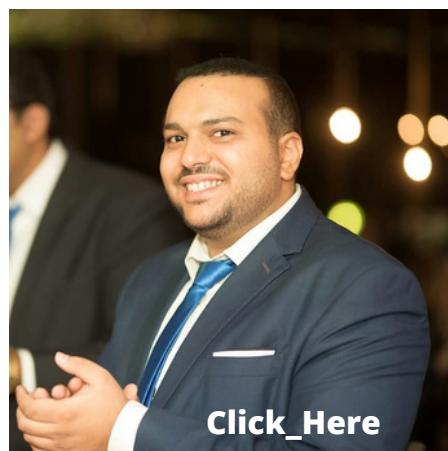
My_Profile →

**Under the supervision of
Engineer Keroloes Shenouda**



[Click_Here](#)

Master_EMBEDDED_Systems



[Click_Here](#)



Learn_In_Depth

Table Of Content :

- **Case study**
- **Requirement Diagram**
- **System Analysis :**
 - 1. Use Case Diagram**
 - 2. Activity Diagram**
 - 3. Sequence Diagram**
- **System Design**
- **Source and Header files with Block Diagram**
- **Sections and symbols for object files**
- **Symbols for final executable file**
- **Readelf utility for final executable file**
- **The map file and symbols**
- **Simulation using Proteus**



Master_EMBEDDED_Systems



Learn_In_Depth

Case study

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

Specification (from the client)

A pressure controller informs the crew of a cabin with an alarm when the pressure exceeds 20 bars in the cabin. □

The alarm duration equals 60 seconds.

Keep track of the measured values

□ Assumption

- **The system setup and shutdown procedures are not modeled**
- **The system maintenance is not modeled**
- **The pressure sensor never fails**
- **The alarm never fails**
- **The system never faces power cut.**



Master_EMBEDDED_Systems



Learn_In_Depth

Case study

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

Specification (from the client)

A pressure controller informs the crew of a cabin with an alarm when the pressure exceeds 20 bars in the cabin. □

The alarm duration equals 60 seconds.

Keep track of the measured values

□ Assumption

- **The system setup and shutdown procedures are not modeled**
- **The system maintenance is not modeled**
- **The pressure sensor never fails**
- **The alarm never fails**
- **The system never faces power cut.**



Master_EMBEDDED_Systems



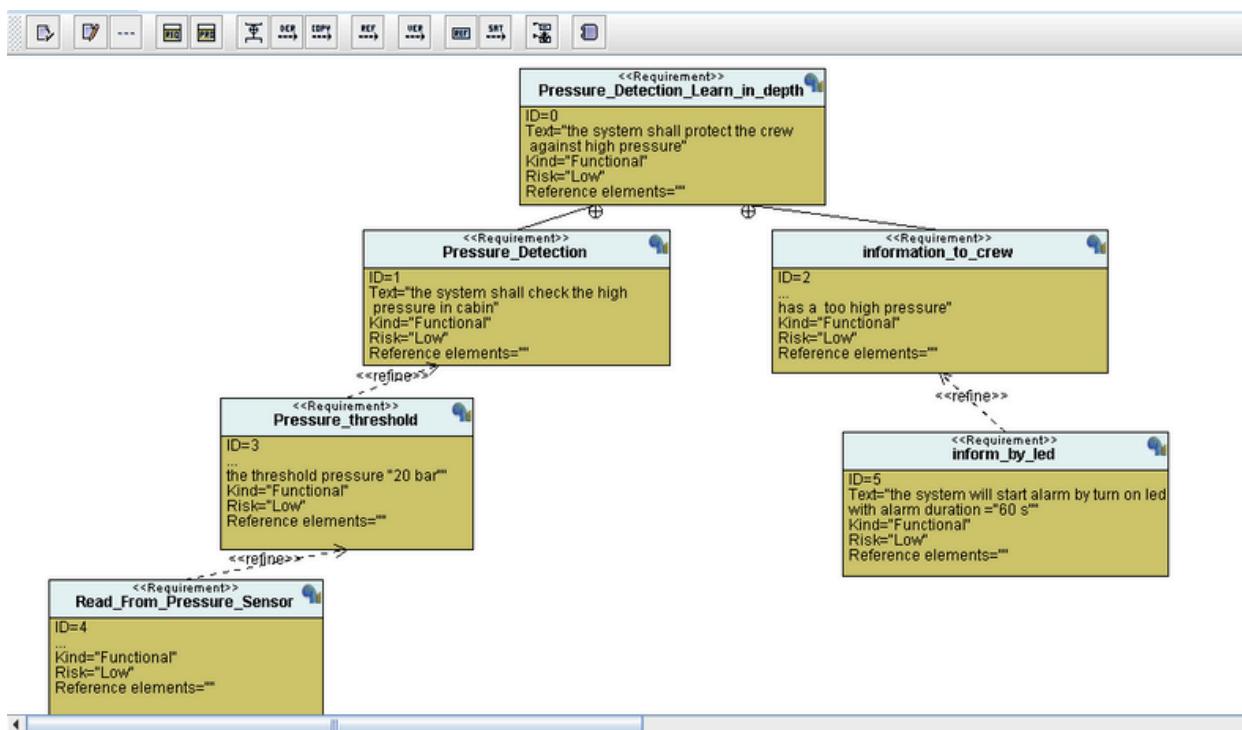
Learn_In_Depth

Requirement Diagram :

We will divide our case study to two main requirements

- **High Pressure Detection Process**
- **Inform The Crew at High Pressure Level**

Each main requirement has refinement requirements as mentioned below



Master_EMBEDDED_Systems

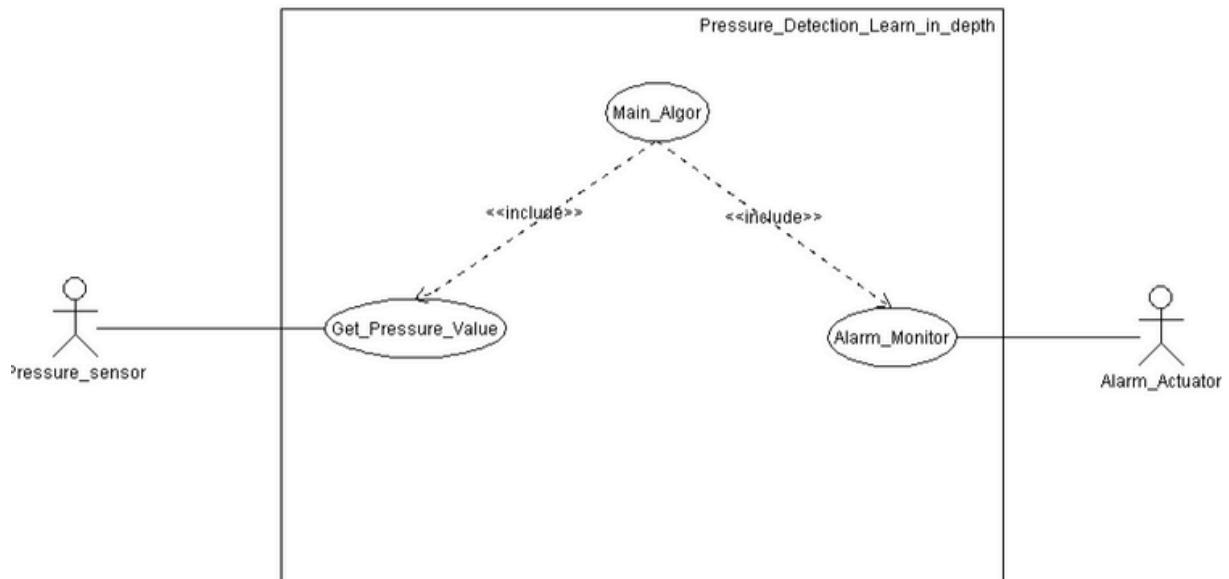


Learn_In_Depth

□ System Analysis Diagram :

- **Use Case Diagram :**

In use case diagram , we discuss system boundary and main functions , so in our use case diagram we will observe that our system boundary include the main function algorithm , Get pressure reading function and display alarm function and exclude the pressure sensor alarm actuator



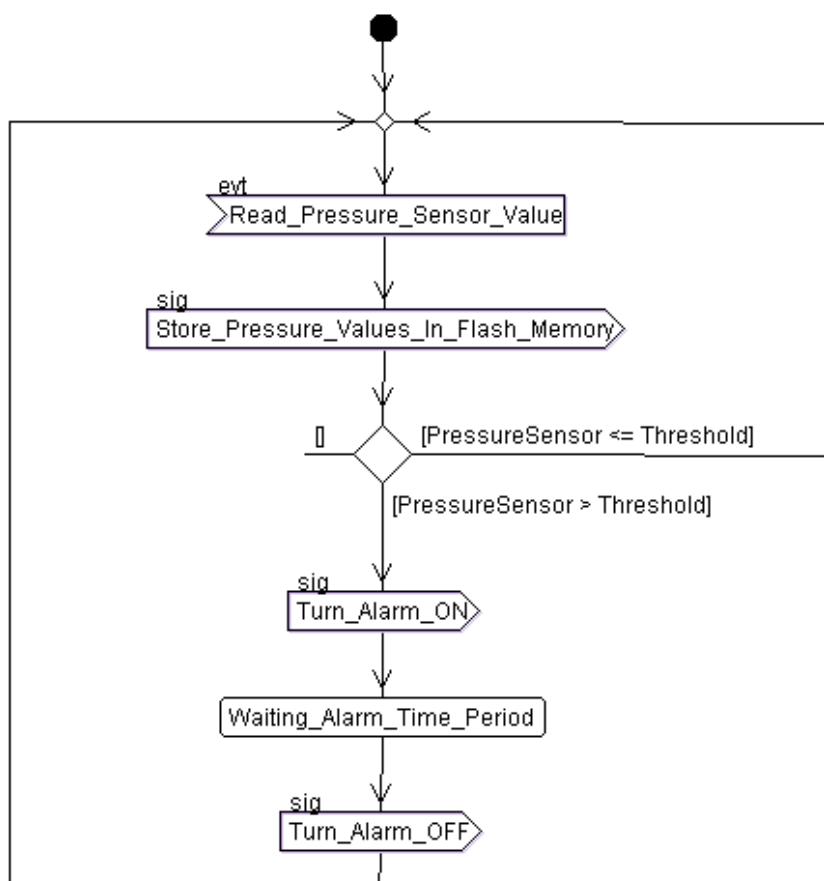
Master_EMBEDDED_Systems



Learn_In_Depth

- **Activity Diagram :**

In Activity Diagram , we discuss relation between main functions , as shown below , firstly we will read the pressure value that we compare the value of pressure with threshold in case study , if it exceeded this value we will turn on the alarm for 60 seconds to inform the crew that the pressure exceeded the threshold .



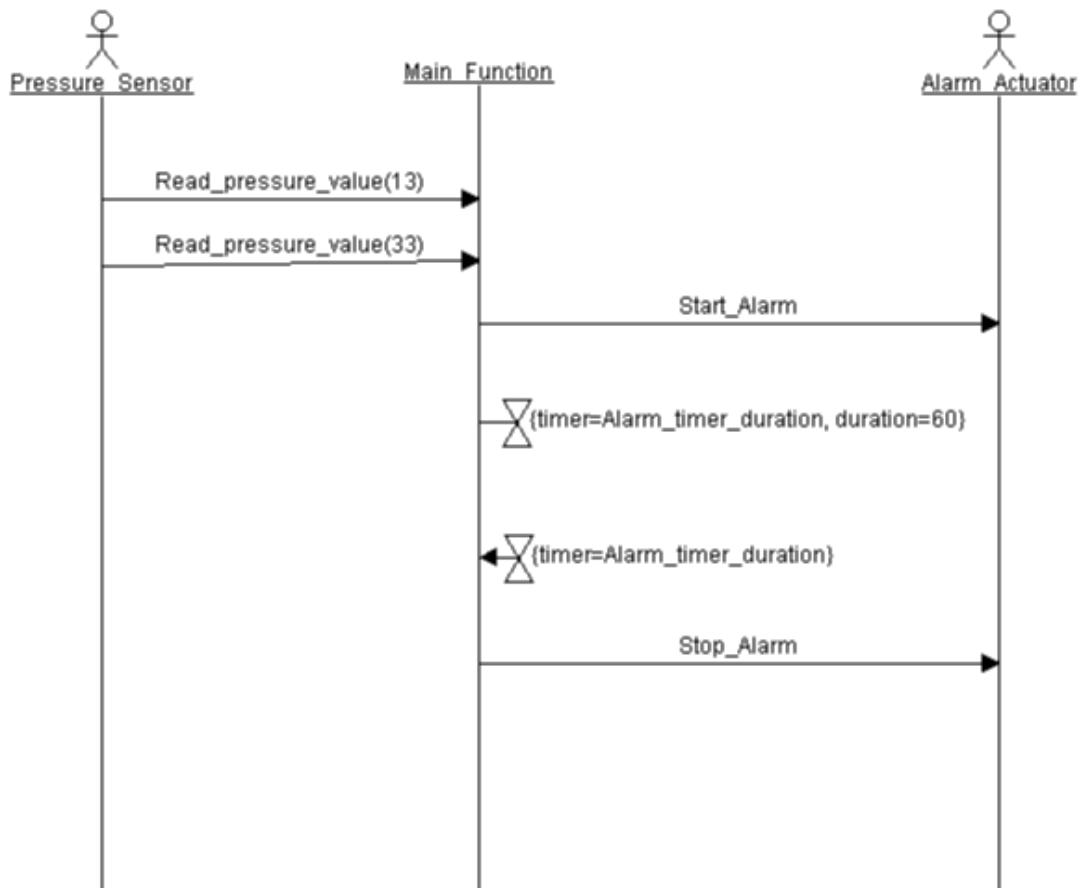
Master_EMBEDDED_Systems



Learn_In_Depth

• Sequence Diagram :

In use Sequence Diagram, we discuss Communication between main system entities and actors ,as we mentioned above in activity diagram , we will see interaction between Pressure sensor actor , main function algorithm , alarm actuator



Master_EMBEDDED_Systems



Learn_In_Depth

System Design:

We will divide our case study to **FOUR** main modules

- **Pressure_Sensor Module:**

It is has one function , senses the pressure of the cabin and send it to the cont

- **Main_Function Module :**

It has three function , the first function is receiving the reading from pressure sensor module , and the second function is checking the value of pressure and take the decision if alarm will turn on or not , the third function is to send the reading to the alarm to display it

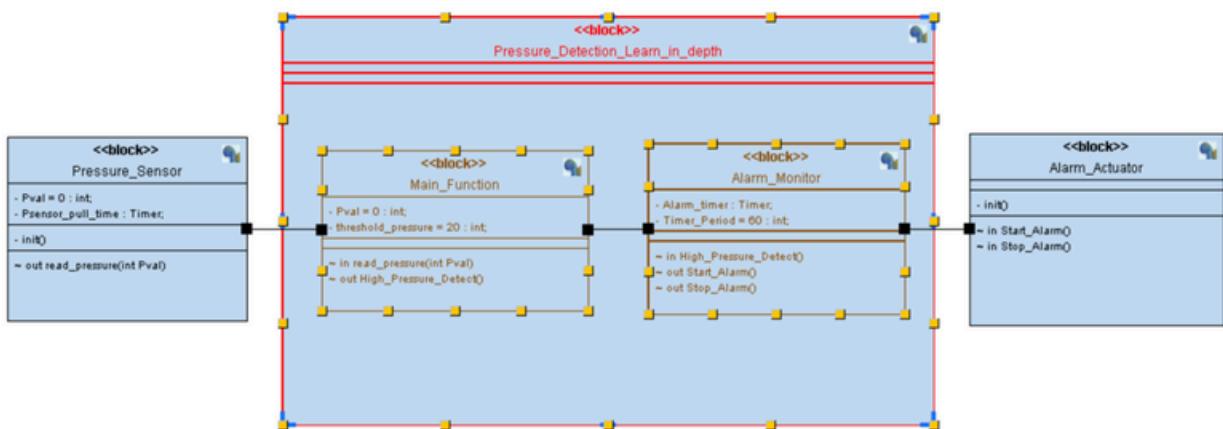
- **Alarm Monitor Module :**

It has two function , the first function is receiving the reading from controller , the second function to send the action will be token to the alarm monitor drive

- **Alarm_Actuator :**

It has one function , take the action if turning on the alarm or no

Block_Diagram

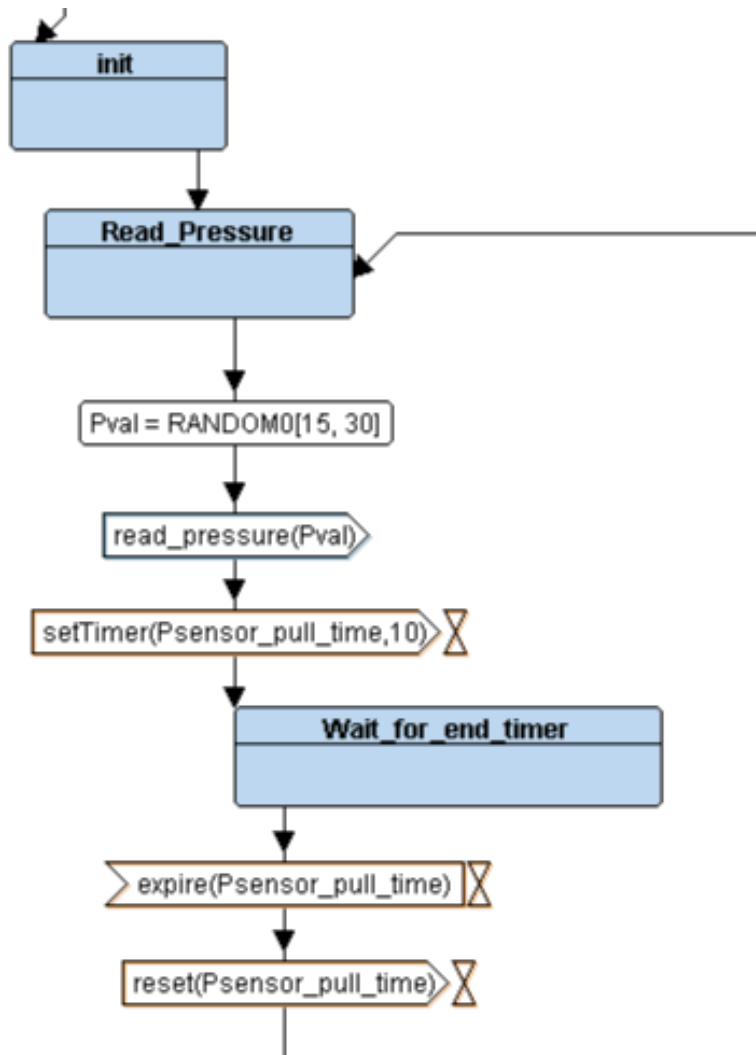




Learn_In_Depth

Source files with Block Diagrams

- Pressure_Sensor _Diagram :



Master_EMBEDDED_Systems



Learn_In_Depth

• Pressure_Sensor _Header_File :

```
1  /*
2  * Pressure_Sensor.h
3  *
4  * Created on: Sep 15, 2024
5  * Author: Michael Maged
6  */
7
8 #ifndef PRESSURE_SENSOR_H_
9 #define PRESSURE_SENSOR_H_
10 #include "driver.h"
11 //=====
12 //Define States of Presure Sensor
13 typedef enum {
14     waiting_state,
15     reading_state
16 } state_sensor;
17 //=====
18 // function to initialize the Pressure Sensor
19 void Pressure_Sensor_init();
20 //=====
21 //declare the function of states
22 STATE_Define(waiting_state);
23 STATE_Define(reading_state);
24 //=====
25 //declare the pointer to function to read from the main function
26 extern void (*Pointer_sensor)();
27 #endif /* PRESSURE_SENSOR_H_ */
```

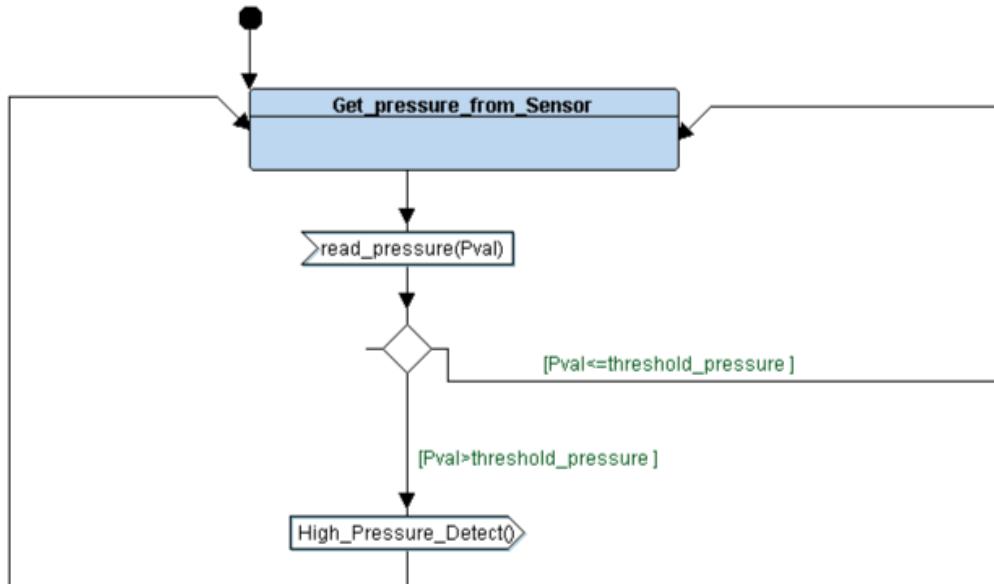
• Pressure_Sensor _Source_File :

```
1  /*
2  * Pressure_Sensor.c
3  *
4  * Created on: Sep 15, 2024
5  * Author: Michael Maged
6  */
7 #include "driver.h"
8 #include "Pressure_Sensor.h"
9 //define the variables of block
10 int Pval=0;Psensor_pull_time=1000;
11 //declare the function is to be send to main function
12 void Read_Pressure(int Pval);
13 //define the pointer to function to store the current state of pressure sensor
14 void (*Pointer_sensor)();
15 //init function of pressure sensor
16 void Pressure_Sensor_init()
17 {
18     Pointer_sensor=STATE_(reading_state); //by default the start of pressure sensor read of pressure
19 }
20 //=====
21 //defination the states of Pressure Sensor
22 STATE_Define(waiting_state)
23 {
24     Delay(Psensor_pull_time); //delay to waiting period and reading again
25     Pointer_sensor=STATE_(reading_state);
26 }
27 //=====
28 STATE_Define(reading_state)
29 {
30     Pval=getPressureVal();
31     Read_Pressure(Pval);
32     Pointer_sensor=STATE_(waiting_state);
33 }
```



Learn_In_Depth

- **Main Function Diagram :**



- **Main Function Header File :**

```
1  /*
2   * Main_Function.h
3   *
4   * Created on: Sep 15, 2024
5   *      Author: Michael Maged
6   */
7
8 #ifndef MAIN_FUNCTION_H_
9 #define MAIN_FUNCTION_H_
10 #include "driver.h"
11 //define the states of the main function
12 typedef enum {
13     main_waiting,
14     main_detect_high_pressure
15 } State_Main;
16 //=====
17 //declare the function for each states
18 STATE_Define(main_waiting);
19 STATE_Define(main_detect_high_pressure);
20 //=====
21 //declare the pointer to function to read form all files
22 extern void (*Pointer_main_state)();
23
24 #endif /* MAIN_FUNCTION_H_ */
25
```

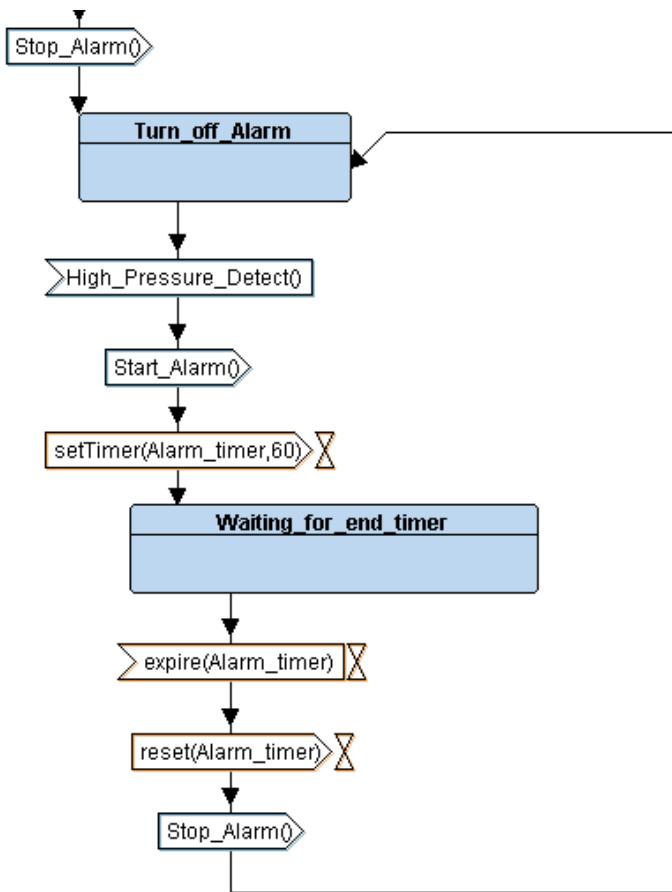


Learn_In_Depth

• Main Function Source File :

```
1  /*
2   * Main_Function.c
3   *
4   * Created on: Sep 15, 2024
5   * Author: Michael maged
6   */
7  #include "stdio.h"
8  #include "Main_Function.h"
9  //define the variables for this block
10 int P_value=0,P_threshold=20;
11 //declare the function to send signal to alarm in state detect the high pressure
12 void High_Pressure_Detect();
13 //define the pointer to function to store the state of main function
14 void (*Pointer_main_state)();
15 //-----
16 //defination the function it read from pressure sensor
17 void Read_Pressure(int Pval)
18 {
19     P_value=Pval;
20     (P_value <= P_threshold)? (Pointer_main_state=STATE_(main_waiting)) : (Pointer_main_state=STATE_(main_detect_high_pressure));
21 }
22 //-----
23 //defination the function for each states
24 STATE_Define(main_waiting)
25 {
26     Pointer_main_state=STATE_(main_waiting);
27 }
28 STATE_Define(main_detect_high_pressure)
29 {
30     High_Pressure_Detect();
31     Pointer_main_state=STATE_(main_waiting);
32 }
33 //-----
```

• Alarm Monitor Diagram :





Learn_In_Depth

- **Alarm Monitor Header File :**

```
1  /*
2   * Alarm_Monitor.h
3   *
4   * Created on: Sep 15, 2024
5   * Author: Michael Maged
6   */
7
8  #ifndef ALARM_MONITOR_H_
9  #define ALARM_MONITOR_H_
10 #include "driver.h"
11 //define states of this block
12 typedef enum
13 {
14     stop_alarm,
15     start_alarm,
16 }State_Alarm;
17 //define the function for each state
18 STATE_Define(stop_alarm);
19 STATE_Define(start_alarm);
20 //declare the pointer to function to read form all files
21 extern void (*Pointer_Alarm_state)();
22
23 #endif /* ALARM_MONITOR_H_ */
```

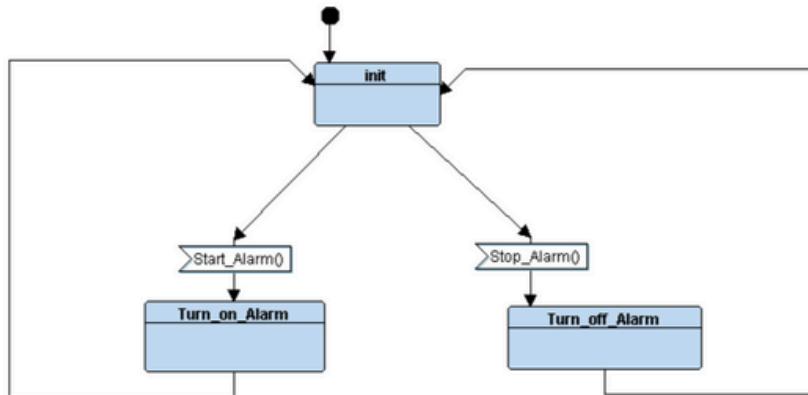
- **Alarm Monitor Source File :**

```
6
7  #include "Alarm_Monitor.h"
8  #include "Alarm_Actuator.h"
9
10 //=====
11 //define the pointer to store the state of alarm
12 void (*Pointer_Alarm_state)();
13 //defination the function that get the signal from the main
14 void High_Pressure_Detect()
15 {
16     Pointer_Alarm_state=STATE_(start_alarm) ;
17 }
18 //=====
19
20 //the defination for each state in alarm
21 STATE_Define(stop_alarm)
22 {
23     State_Actuator(stop);
24     Pointer_Alarm_state=STATE_(stop_alarm) ;
25 }
26 STATE_Define(start_alarm)
27 {
28     State_Actuator(start);
29     Pointer_Alarm_state=STATE_(stop_alarm) ;
30 }
31 //=====
```



Learn_In_Depth

- **Alarm_Actuator_Diagram :**



- **Alarm_Actuator_Header_File :**

```
1  /*
2   * Alarm_Actuator.h
3   *
4   * Created on: Sep 15, 2024
5   *      Author: Michael Maged
6   */
7
8  #ifndef ALARM_ACTUATOR_H_
9  #define ALARM_ACTUATOR_H_
10 #include "driver.h"
11 //define the state of alarm actuator
12 typedef enum {
13     start,
14     stop,
15     wait
16 } state_alarm;
17
18 //=====
19 //define function to send state of alarm to alarm actuator
20 void State_Actuator(state_alarm state);
21 //=====
22 //define the function of each state
23 STATE_Define(stop);
24 STATE_Define(start);
25 STATE_Define(wait);
26 extern void (*state_alarm_actuator)();
27 #endif /* ALARM_ACTUATOR_H_ */
```



Learn_In_Depth

- **Alarm_Actuator_Source_File :**

```
1  /*
2  * Alarm_Actuator.c
3  *
4  * Created on: Sep 15, 2024
5  * Author: Michael maged
6  */
7  #include "Alarm_Actuator.h"
8
9  //define pointer to store the state of alarm actuator
10 void (*state_alarm_actuator)();
11 //=====
12
13 void Alarm_Actuator_init()
14 {
15     state_alarm_actuator=STATE_(stop);
16 }
17 //=====
18
19 void State_Actuator(state_alarm state)
20 {
21     if( state == start)
22     {
23         state_alarm_actuator=STATE_(start);
24     }
25     else if(state == stop)
26     {
27         state_alarm_actuator=STATE_(stop);
28     }
29 }
30 //=====
31
32 STATE_Define(stop)
33 {
34     Set_Alarm_actuator(1);
35     state_alarm_actuator=STATE_(stop);
36 }
37 STATE_Define(start)
38 {
39     Set_Alarm_actuator(0);
40     state_alarm_actuator =STATE_(wait);
41 }
42 STATE_Define(wait)
43 {
44     Delay(60000);
45     state_alarm_actuator=STATE_(stop);
46 }
```



Learn_In_Depth

- **Driver_Header_file :**

```
1 #include <stdint.h>
2 #include <stdio.h>
3
4 // Automatic state function generated by macros
5 #define STATE_Define(_STATEFUNC_) void ST##_STATEFUNC_()
6 #define STATE_( _stateFUN_)           ST##_stateFUN_
7
8 #define SET_BIT(ADDRESS,BIT)   ADDRESS |= (1<<BIT)
9 #define RESET_BIT(ADDRESS,BIT) ADDRESS &= ~(1<<BIT)
10 #define TOGGLE_BIT(ADDRESS,BIT) ADDRESS ^= (1<<BIT)
11 #define READ_BIT(ADDRESS,BIT) ((ADDRESS) & (1<<(BIT)))
12
13
14 #define GPIO_PORTA 0x40010800
15 #define BASE_RCC 0x40021000
16
17 #define APB2ENR *(volatile uint32_t *)(BASE_RCC + 0x18)
18
19 #define GPIOA_CRL *(volatile uint32_t *)(GPIO_PORTA + 0x00)
20 #define GPIOA_CRH *(volatile uint32_t *)(GPIO_PORTA + 0X04)
21 #define GPIOA_IDR *(volatile uint32_t *)(GPIO_PORTA + 0x08)
22 #define GPIOA_ODR *(volatile uint32_t *)(GPIO_PORTA + 0x0C)
23
24
25 void Delay(int nCount);
26 int getPressureVal();
27 void Set_Alarm_actuator(int i);
28 void GPIO_INITIALIZATION ();
29
```

- **Driver_Source_file :**

```
1 #include "driver.h"
2 #include <stdint.h>
3 #include <stdio.h>
4
5 void Delay(int nCount)
6 {
7     for(; nCount != 0; nCount--);
8 }
9
10 int getPressureVal(){
```



Learn_In_Depth

```
10     int getPressureVal(){
11         return (GPIOA_IDR & 0xFF);
12     }
13
14     void Set_Alarm_actuator(int i){
15         if (i == 1){
16             SET_BIT(GPIOA_ODR,13);
17         }
18         else if (i == 0){
19             RESET_BIT(GPIOA_ODR,13);
20         }
21     }
22
23     void GPIO_INITIALIZATION (){
24         SET_BIT(APB2ENR, 2);
25         GPIOA_CRL &= 0xFF0FFFFF;
26         GPIOA_CRL |= 0x00000000;
27         GPIOA_CRH &= 0xFF0FFFFF;
28         GPIOA_CRH |= 0x22222222;
29     }
```

- **Main_Program_File :**

```
1 #include <stdint.h>
2 #include <stdio.h>
3
4 #include "Alarm_Monitor.h"
5 #include "Alarm_Actuator.h"
6 #include "Pressure_Sensor.h"
7 #include "Main_Function.h"
8 void setup()
9 {
10     GPIO_INITIALIZATION();
11     Pressure_Sensor_init();
12     Pointer_main_state = STATE_(main_waiting);
13     Pointer_Alarm_state = STATE_(stop_alarm);
14     Alarm_Actuator_init();
15 }
```



Learn_In_Depth

```
16
17
18 int main (){
19     setup();
20
21     while (1)
22     {
23         Pointer_sensor();
24         Pointer_main_state();
25         Pointer_Alarm_state();
26         state_alarm_actuator();
27     }
28
29 }
```

- **Make_File :**

```
1 #eng: Michael Maged
2 CC=arm-none-eabi-
3 CFLAGS= -mcpu=cortex-m4 -gdwarf-2 -g
4 INCs=-I .
5 LIBS=
6 SRC =$(wildcard *.c)
7 OBJ= $(SRC:.c=.o)
8 As =$(wildcard *.s)
9 AsOBJ = $(As:.s=.o)
10 Project_name=Pressure_Detection_Learn_in_depth
11 all:$(Project_name).bin
12     @echo -----Build is Done-----
13
14 %.o:%.c
15     | $(CC)gcc.exe -c $(INCs)  $(CFLAGS) $< -o $@
16
17 $(Project_name).elf: $(OBJ) $(AsOBJ)
18     | $(CC)ld.exe -T linker_script.ld $(LIBS) $(OBJ) $(AsOBJ) -o $(Project_name).elf -Map=Map_file.map
19     cp $(Project_name).elf $(Project_name).axf
20
21 $(Project_name).bin: $(Project_name).elf
22     | $(CC)objcopy.exe -O binary $< $@
```



Master_EMBEDDED_Systems



Learn_In_Depth

- **Startup Program File :**

```
1 // Startup.c
2 // Eng.Michael
3 //First_Project_High_Pressure_Detection
4 #include <stdint.h>
5 extern int main(void);
6 /*****
7 // External symbols from startup file (locator)
8
9 extern uint32_t S_TEXT      ;
10 extern uint32_t E_TEXT      ;
11 extern uint32_t S_DATA      ;
12 extern uint32_t E_DATA      ;
13 extern uint32_t S_BSS       ;
14 extern uint32_t E_BSS       ;
15 extern uint32_t _STACK_TOP  ;
16
17 *****/
18 // Default Handler to handle any interrupt and to layout the memory boundaries
19
20 void Default_Handler(void)
21 {
22     // Copy the data section from Flash to Sram
23
24     uint32_t DATA_SIZE = (uint8_t *) &E_DATA - (uint8_t *) &S_DATA ;
25     uint8_t* ptr_SOURCE = (uint8_t *) &E_TEXT ;
26     uint8_t* ptr_DESTINATION = (uint8_t *) &S_DATA ;
27     for(uint32_t i = 0 ; i < DATA_SIZE ; i++)
28     {
29         *((uint8_t *) (ptr_DESTINATION++)) = *((uint8_t *) (ptr_SOURCE++)) ;
30     }
31
32     // Initialize .bss section by zeros
33     uint32_t BSS_SIZE = (uint8_t *) &E_BSS - (uint8_t *) &S_BSS ;
34     for(uint32_t i = 0 ; i < BSS_SIZE ; i++)
35     {
36         *((uint8_t *) (ptr_DESTINATION++)) = ((uint8_t)(0)) ;
37     }
38
39
40     main();
41 }
42 *****/
43
44 // some Interrupts prototypes may be occur at runtime execution
45
46 void Reset_Handler(void)      __attribute__((weak,alias("Default_Handler")));
47
48 void NMI_Handler(void)        __attribute__((weak,alias("Default_Handler")));
49
50 void H_Fault_Handler(void)    __attribute__((weak,alias("Default_Handler")));
51
52 void MM_Fault_Handler(void)   __attribute__((weak,alias("Default_Handler")));
53
54 void Bus_Fault(void)          __attribute__((weak,alias("Default_Handler")));
55
56 void Usage_Fault_Handler(void) __attribute__((weak,alias("Default_Handler")));
57
58 *****/
59
60 // Initialize the stack pointer
61 // define the entry point to program to execute the main function using Default_Handler interrupt
62 // set the other used interrupts
63
64 uint32_t vectors[] __attribute__((section(".vectors")))
65 {
66     (uint32_t ) &_STACK_TOP      ,
67     (uint32_t ) &Reset_Handler   ,
68     (uint32_t ) &NMI_Handler     ,
69     (uint32_t ) &H_Fault_Handler ,
70     (uint32_t ) &MM_Fault_Handler,
71     (uint32_t ) &Bus_Fault       ,
72     (uint32_t ) &Usage_Fault_Handler
73
74
75
76
77 };
78 *****/
79
```



Learn_In_Depth

• Linker script.ld :

```
1  ****
2  * @file      : linker_script.ld
3  * engineer   : Michael maged
4  ****
5  /* Memory Types and lengths */
6  MEMORY
7  {
8
9    FLASH(RX): ORIGIN = 0X08000000 , LENGTH = 128K
10   SRAM(RWX): ORIGIN = 0X20000000 , LENGTH = 20K
11
12 }
13
14
15 SECTIONS
16 {
17   .text :
18   {
19     S_TEXT = . ;
20     *(.vectors*)
21     *(.text*)
22     *(.rodata)
23     E_TEXT = . ;
24
25   }> FLASH
26   .data :
27   {
28     S_DATA = . ;
29     *(.data)
30     . = ALIGN(4) ;
31     E_DATA = . ;
32
33   }> SRAM AT> FLASH
34
35   .bss :
36   {
37     S_BSS = . ;
38     *(.bss)
39     E_BSS = . ;
40     . = ALIGN(4) ;
41     . = . + 0x1000 ;
42     _STACK_TOP = . ;
43
44   }> SRAM
45   .comment :
46   {
47     *(.comment) *(COMMON)
48
49   }> FLASH
50
51
52 }
```



Learn_In_Depth

Sections and symbols for object files

- Pressure Sensor object file sections :

```
Graphic@DESKTOP-QIET4IR MINGW32 /e/Mastering Embedded System (online)/First Term/first project/The_First_Project_First_Term
$ arm-none-eabi-objdump.exe -h Pressure_Sensor.o

Pressure_Sensor.o:      file format elf32-littlearm

Sections:
Idx Name      Size    VMA     LMA     File off  Align
 0 .text      00000070 00000000 00000000 00000034 2**2
                CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data      00000004 00000000 00000000 000000a4 2**2
                CONTENTS, ALLOC, LOAD, DATA
 2 .bss       00000004 00000000 00000000 000000a8 2**2
                ALLOC
 3 .debug_info 000009ff 00000000 00000000 000000a8 2**0
                CONTENTS, RELOC, READONLY, DEBUGGING
 4 .debug_abbrev 000001c7 00000000 00000000 000000a7 2**0
                CONTENTS, READONLY, DEBUGGING
 5 .debug_loc   0000009c 00000000 00000000 00000c6e 2**0
                CONTENTS, READONLY, DEBUGGING
 6 .debug_aranges 00000020 00000000 00000000 00000d0a 2**0
                CONTENTS, RELOC, READONLY, DEBUGGING
 7 .debug_line   00000152 00000000 00000000 00000d2a 2**0
                CONTENTS, RELOC, READONLY, DEBUGGING
 8 .debug_str    000005aa 00000000 00000000 00000e7c 2**0
                CONTENTS, READONLY, DEBUGGING
 9 .comment     0000007f 00000000 00000000 00001426 2**0
                CONTENTS, READONLY
10 .debug_frame 00000068 00000000 00000000 000014a8 2**2
                CONTENTS, RELOC, READONLY, DEBUGGING
11 .ARM.attributes 00000033 00000000 00000000 00001510 2**0
                CONTENTS, READONLY
```

- Pressure Sensor object file symbols :

```
Graphic@DESKTOP-QIET4IR MINGW32 /e/Mastering Embedded System (online)/First Term/first project/The_First_Project_First_Term
$ arm-none-eabi-nm.exe Pressure_Sensor.o
U Delay
U getPressureVal
00000004 C Pointer_sensor
00000000 T Pressure_Sensor_init
00000000 D Psensor_pull_time
00000000 B Pval
        U Read_Pressure
00000040 T ST_reading_state
0000001c T ST_waiting_state
```



Master_EMBEDDED_Systems



Learn_In_Depth

• Main Function object file section :

```
Graphic@DESKTOP-QIET4IR MINGW32 /e/Mastering Embedded System (online)/First Term/first project/The_First_Project_First_Term
$ arm-none-eabi-objdump.exe -h Main_Function.o

Main_Function.o:      file format elf32-littlearm

Sections:
Idx Name      Size    VMA     LMA     File off  Align
 0 .text      00000080 00000000 00000000 00000034 2**2
               CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data      00000004 00000000 00000000 000000b4 2**2
               CONTENTS, ALLOC, LOAD, DATA
 2 .bss      00000004 00000000 00000000 000000b8 2**2
               ALLOC
 3 .debug_info 00000a0f 00000000 00000000 000000b8 2**0
               CONTENTS, RELOC, READONLY, DEBUGGING
 4 .debug_abbrev 000001ee 00000000 00000000 00000ac7 2**0
               CONTENTS, READONLY, DEBUGGING
 5 .debug_loc   000000cc 00000000 00000000 00000cbs 2**0
               CONTENTS, READONLY, DEBUGGING
 6 .debug_aranges 00000020 00000000 00000000 00000d81 2**0
               CONTENTS, RELOC, READONLY, DEBUGGING
 7 .debug_line   0000015e 00000000 00000000 00000dal 2**0
               CONTENTS, RELOC, READONLY, DEBUGGING
 8 .debug_str    000005b2 00000000 00000000 00000eff 2**0
               CONTENTS, READONLY, DEBUGGING
 9 .comment     0000007f 00000000 00000000 000014b1 2**0
               CONTENTS, READONLY
10 .debug_frame 00000074 00000000 00000000 00001530 2**2
               CONTENTS, RELOC, READONLY, DEBUGGING
11 .ARM.attributes 00000033 00000000 00000000 000015a4 2**0
               CONTENTS, READONLY
```

• Main Function object file symbols :

```
Graphic@DESKTOP-QIET4IR MINGW32 /e/Mastering Embedded System (online)/First Term/first project/The_First_Project_First_Term
$ arm-none-eabi-nm.exe Main_Function.o
U High_Pressure_Detect
00000000 D P_threshold
00000000 B P_value
00000004 C Pointer_main_state
00000000 T Read_Pressure
00000064 T ST_main_detect_high_pressure
00000048 T ST_main_waiting
```



Master_EMBEDDED_Systems



Learn_In_Depth

• Alarm Monitor object file sections :

```
Graphic@DESKTOP-QIET4IR MINGW32 /e/Mastering Embedded System (online)/First Term
/First project/The_First_Project_First_Term
$ arm-none-eabi-objdump.exe -h Alarm_Monitor.o

Alarm_Monitor.o:      file format elf32-littlearm

Sections:
Idx Name      Size    VMA     LMA     File off  Align
 0 .text      00000054 00000000 00000000 00000034 2**2
                CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data      00000000 00000000 00000000 00000088 2**0
                CONTENTS, ALLOC, LOAD, DATA
 2 .bss       00000000 00000000 00000000 00000088 2**0
                ALLOC
 3 .debug_info 00000a08 00000000 00000000 00000088 2**0
                CONTENTS, RELOC, READONLY, DEBUGGING
 4 .debug_abbrev 000001d0 00000000 00000000 00000a90 2**0
                CONTENTS, READONLY, DEBUGGING
 5 .debug_loc   0000009c 00000000 00000000 00000c60 2**0
                CONTENTS, READONLY, DEBUGGING
 6 .debug_aranges 00000020 00000000 00000000 00000cf0 2**0
                CONTENTS, RELOC, READONLY, DEBUGGING
 7 .debug_line   00000161 00000000 00000000 00000d1c 2**0
                CONTENTS, RELOC, READONLY, DEBUGGING
 8 .debug_str    000005a4 00000000 00000000 00000e7d 2**0
                CONTENTS, READONLY, DEBUGGING
 9 .comment     0000007f 00000000 00000000 00001421 2**0
                CONTENTS, READONLY
10 .debug_frame 00000068 00000000 00000000 000014a0 2**2
                CONTENTS, RELOC, READONLY, DEBUGGING
11 .ARM.attributes 00000033 00000000 00000000 00001508 2**0
                CONTENTS, READONLY
```

• Alarm Monitor object file symbols :

```
Graphic@DESKTOP-QIET4IR MINGW32 /e/Mastering
$ arm-none-eabi-nm.exe Alarm_Monitor.o
00000000 T High_Pressure_Detect
00000004 C Pointer_Alarm_state
00000038 T ST_start_alarm
0000001c T ST_stop_alarm
                           U State_Actuator
```





Learn_In_Depth

- **Alarm Actuator object file sections :**

```
Graphic@DESKTOP-QIET4IR MINGW32 /e/Mastering Embedded System (online)/First Term/First project/The_First_Project_First_Term
$ arm-none-eabi-objdump.exe -h Alarm_Actuator.o

Alarm_Actuator.o:      file format elf32-littlearm

Sections:
Idx Name      Size    VMA     LMA   File off  Align
 0 .text     000000b0 00000000 00000000 00000034 2**2
              CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data     00000000 00000000 00000000 000000e4 2**0
              CONTENTS, ALLOC, LOAD, DATA
 2 .bss     00000000 00000000 00000000 000000e4 2**0
              ALLOC
 3 .debug_info 00000a44 00000000 00000000 000000e4 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 4 .debug_abbrev 000001f9 00000000 00000000 00000b28 2**0
              CONTENTS, READONLY, DEBUGGING
 5 .debug_loc   00000124 00000000 00000000 00000d21 2**0
              CONTENTS, READONLY, DEBUGGING
 6 .debug_aranges 00000020 00000000 00000000 00000e45 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 7 .debug_line   0000015a 00000000 00000000 00000e65 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 8 .debug_str    000005bf 00000000 00000000 0000fbff 2**0
              CONTENTS, READONLY, DEBUGGING
 9 .comment     0000007f 00000000 00000000 0000157e 2**0
              CONTENTS, READONLY
10 .debug_frame 000000ac 00000000 00000000 00001600 2**2
              CONTENTS, RELOC, READONLY, DEBUGGING
11 .ARM.attributes 00000033 00000000 00000000 000016ac 2**0
              CONTENTS, READONLY
```

- **Alarm Actuator object file symbols :**

```
Graphic@DESKTOP-QIET4IR MINGW32 /e/Mastering
$ arm-none-eabi-nm.exe Alarm_Actuator.o
00000000 T Alarm_Actuator_init
          U Delay
          U Set_Alarm_actuator
00000074 T ST_start
00000058 T ST_stop
00000090 T ST_wait
0000001c T State_Actuator
00000004 C state_alarm_actuator
```



Master_EMBEDDED_Systems



Learn_In_Depth

• Sections for final executable file :

```
Graphic@DESKTOP-QIET4IR MINGW32 /e/Mastering Embedded System (online)/First Term
/first project/The_First_Project_First_Term
$ arm-none-eabi-objdump.exe -h Pressure_Detection_Learn_in_depth.elf

Pressure_Detection_Learn_in_depth.elf:      file format elf32-littlearm

Sections:
Idx Name      Size    VMA     LMA     File off  Align
 0 .text      000003b8 08000000 08000000 00010000 2**2
              CONTENTS, ALLOC, LOAD, READONLY, CODE
 1 .data      00000008 20000000 080003b8 00020000 2**2
              CONTENTS, ALLOC, LOAD, DATA
 2 .bss       00001008 20000008 080003c0 00030008 2**2
              ALLOC
 3 .comment   00000090 080003c0 080003c0 000203c0 2**2
              CONTENTS, ALLOC, LOAD, DATA
 4 .ARM.attributes 00000033 00000000 00000000 00020450 2**0
              CONTENTS, READONLY
 5 .debug_info 00003df1 00000000 00000000 00020483 2**0
              CONTENTS, READONLY, DEBUGGING
 6 .debug_abbrev 00000bd8 00000000 00000000 00024274 2**0
              CONTENTS, READONLY, DEBUGGING
 7 .debug_loc   00000510 00000000 00000000 00024e4c 2**0
              CONTENTS, READONLY, DEBUGGING
 8 .debug_aranges 000000e0 00000000 00000000 0002535c 2**0
              CONTENTS, READONLY, DEBUGGING
 9 .debug_line   00000a0a 00000000 00000000 0002543c 2**0
              CONTENTS, READONLY, DEBUGGING
10 .debug_str   000007c8 00000000 00000000 00025e46 2**0
              CONTENTS, READONLY, DEBUGGING
11 .debug_frame 0000030c 00000000 00000000 00026610 2**2
              CONTENTS, READONLY, DEBUGGING
```

• Symbols for final executable file :

```
Graphic@DESKTOP-QIET4IR MINGW32 /e/Mastering Embedded System (on
/first project/The_First_Project_First_Term
$ arm-none-eabi-nm.exe Pressure_Detection_Learn_in_depth.elf
20001010 B _STACK_TOP
0800001c T Alarm_Actuator_init
08000338 W Bus_Fault
08000338 T Default_Handler
08000120 T Delay
20000010 B E_BSS
20000008 D E_DATA
080003b8 T E_TEXT
08000140 T getPressureVal
08000194 T GPIO_INITIALIZATION
08000338 W H_Fault_Handler
080000cc T High_Pressure_Detect
08000214 T main
08000338 W MM_Fault_Handler
08000338 W NMI_Handler
20000000 D P_threshold
20000008 B P_value
08000444 D Pointer_Alarm_state
08000448 D Pointer_main_state
0800044c D Pointer_sensor
```



Learn_In_Depth

```
0800044c D Pointer_sensor
080002c8 T Pressure_Sensor_init
20000004 D Psensor_pull_time
2000000c B Pval
08000248 T Read_Pressure
08000338 W Reset_Handler
20000008 B S_BSS
20000000 D S_DATA
08000000 T S_TEXT
08000158 T Set_Alarm_actuator
080001e4 T setup
080002ac T ST_main_detect_high_pressure
08000290 T ST_main_waiting
08000308 T ST_reading_state
08000090 T ST_start
08000104 T ST_start_alarm
08000074 T ST_stop
080000e8 T ST_stop_alarm
080000ac T ST_wait
080002e4 T ST_waiting_state
08000038 T State_Actuator
08000440 D state_alarm_actuator
08000338 W Usage_Fault_Handler
```

- **The map file and symbols :**

```
1
2 Allocating common symbols
3 Common symbol      size      file
4
5 Pointer_Alarm_state
6           0x4          Alarm_Monitor.o
7 Pointer_main_state  0x4          Main_Function.o
8 state_alarm_actuator
9           0x4          Alarm_Actuator.o
10 Pointer_sensor     0x4          Pressure_Sensor.o
11
12 Memory Configuration
13
14 Name      Origin      Length      Attributes
15 FLASH    0x80000000 0x00020000  xr
16 SRAM    0x20000000 0x00005000  xrw
17 *default* 0x00000000 0xffffffff
18
19 Linker script and memory map
20
21
22 .text      0x08000000 0x3b8      S_TEXT = .
23           0x08000000
24 *(.vectors*)
25 .vectors   0x08000000 0x1c        startup.o
26           0x08000000
27 *(.text*)
28 .text      0x0800001c 0xb0        Alarm_Actuator.o
29           0x0800001c
30           0x08000038
31           0x08000074
32           0x08000090
33           0x080000ac
34 .text      0x080000cc 0x54        Alarm_Monitor.o
```



Learn_In_Depth

```
34 .text      0x080000cc    0x54 Alarm_Monitor.o
35           0x080000cc    High_Pressure_Detect
36           0x080000e8    ST_stop_alarm
37           0x08000104    ST_start_alarm
38 .text      0x08000120    0xc4 driver.o
39           0x08000120    Delay
40           0x08000140    getPressureVal
41           0x08000158    Set_Alarm_actuator
42           0x08000194    GPIO_INITIALIZATION
43 .text      0x080001e4    0x64 main.o
44           0x080001e4    setup
45           0x08000214    main
46 .text      0x08000248    0x80 Main_Function.o
47           0x08000248    Read_Pressure
48           0x08000290    ST_main_waiting
49           0x080002ac    ST_main_detect_high_pressure
50 .text      0x080002c8    0x70 Pressure_Sensor.o
51           0x080002c8    Pressure_Sensor_init
52           0x080002e4    ST_waiting_state
53           0x08000308    ST_reading_state
54 .text      0x08000338    0x80 startup.o
55           0x08000338    H_Fault_Handler
56           0x08000338    MM_Fault_Handler
57           0x08000338    Reset_Handler
58           0x08000338    Bus_Fault
59           0x08000338    Default_Handler
60           0x08000338    Usage_Fault_Handler
61           0x08000338    NMI_Handler
62 *(.rodata) 0x080003b8    E_TEXT = .
63
64
65 .glue_7     0x080003b8    0x0
66 .glue_7     0x080003b8    0x0 linker stubs
67
68 .glue_7t    0x080003b8    0x0
69 .glue_7t    0x080003b8    0x0 linker stubs
70
71 .vfp11_veneer 0x080003b8    0x0
72 .vfp11_veneer 0x080003b8    0x0 linker stubs
73
74 .v4_bx     0x080003b8    0x0
75 .v4_bx     0x080003b8    0x0 linker stubs
76
77 .iplt      0x080003b8    0x0
78 .iplt      0x080003b8    0x0 Alarm_Actuator.o
79
80 .rel.dyn   0x080003b8    0x0
81 .rel.iplt  0x080003b8    0x0 Alarm_Actuator.o
82
83 .data      0x20000000    0x8 load address 0x080003b8
84           0x20000000    S_DATA = .
85 *(.data)
86 .data      0x20000000    0x0 Alarm_Actuator.o
87 .data      0x20000000    0x0 Alarm_Monitor.o
88 .data      0x20000000    0x0 driver.o
89 .data      0x20000000    0x0 main.o
90 .data      0x20000000    0x4 Main_Function.o
91           0x20000000    P_threshold
92 .data      0x20000004    0x4 Pressure_Sensor.o
93           0x20000004    Psensor_pull_time
94 .data      0x20000008    0x0 startup.o
```



Learn_In_Depth

```
91      .data          0x20000000          P_threshold
92      .data          0x20000004          0x4 Pressure_Sensor.o
93      .data          0x20000004          Psensor_pull_time
94      .data          0x20000008          0x0 startup.o
95      .data          0x20000008          . = ALIGN (0x4)
96      .data          0x20000008          E_DATA = .
97
98      .igot.plt     0x20000008          0x0 load address 0x080003c0
99      .igot.plt     0x20000008          0x0 Alarm_Actuator.o
100
101     .bss           0x20000008          0x1008 load address 0x080003c0
102     .bss           0x20000008          S_BSS = .
103     *(.bss)
104     .bss           0x20000008          0x0 Alarm_Actuator.o
105     .bss           0x20000008          0x0 Alarm_Monitor.o
106     .bss           0x20000008          0x0 driver.o
107     .bss           0x20000008          0x0 main.o
108     .bss           0x20000008          0x4 Main_Function.o
109     .bss           0x20000008          P_value
110     .bss           0x2000000c          0x4 Pressure_Sensor.o
111     .bss           0x2000000c          Pval
112     .bss           0x20000010          0x0 startup.o
113     .bss           0x20000010          E_BSS = .
114     .bss           0x20000010          . = ALIGN (0x4)
115     .bss           0x20001010          . = (. + 0x1000)
116     *fill*
117     0x20000010          0x1000
118     0x20001010          _STACK_TOP = .
119
120     .comment        0x080003c0          0x90
121     *(.comment)
122     .comment        0x080003c0          0x7e Alarm_Actuator.o
123     .comment        0x0800043e          0x7f (size before relaxing)
124     .comment        0x0800043e          0x7f Alarm_Monitor.o
125     .comment        0x0800043e          0x7f driver.o
126
127     .comment        0x0800043e          0x7f main.o
128     .comment        0x0800043e          0x7f Main_Function.o
129     .comment        0x0800043e          0x7f Pressure_Sensor.o
130     .comment        0x0800043e          0x7f startup.o
131     *(COMMON)
132     *fill*
133     COMMON          0x08000440          0x2
134     COMMON          0x08000440          0x4 Alarm_Actuator.o
135     COMMON          0x08000440          state_alarm_actuator
136     COMMON          0x08000444          0x4 Alarm_Monitor.o
137     COMMON          0x08000444          Pointer_Alarm_state
138     COMMON          0x08000448          0x4 Main_Function.o
139     COMMON          0x08000448          Pointer_main_state
140     COMMON          0x0800044c          0x4 Pressure_Sensor.o
141     COMMON          0x0800044c          Pointer_sensor
142
143     LOAD Alarm_Actuator.o
144     LOAD Alarm_Monitor.o
145     LOAD driver.o
146     LOAD main.o
147     LOAD Main_Function.o
148     LOAD Pressure_Sensor.o
149     LOAD startup.o
150     OUTPUT(Pressure_Detection_Learn_in_depth.elf elf32-littlearm)
151
152     .ARM.attributes 0x00000000          0x33
153     .ARM.attributes 0x00000000          0x33 Alarm_Actuator.o
154     .ARM.attributes 0x00000033          0x33 Alarm_Monitor.o
155     .ARM.attributes 0x00000066          0x33 driver.o
156     .ARM.attributes 0x00000099          0x33 main.o
```

Master_EMBEDDED_Systems



Learn_In_Depth

```
157          0x00000099    0x33 main.o
158 .ARM.attributes      0x000000cc    0x33 Main_Function.o
159 .ARM.attributes      0x000000ff    0x33 Pressure_Sensor.o
160 .ARM.attributes      0x00000132    0x33 startup.o
161
162
163
164
165 .debug_info     0x00000000    0x3df1
166 .debug_info     0x00000000    0xa44 Alarm_Actuator.o
167 .debug_info     0x00000a44    0xa08 Alarm_Monitor.o
168 .debug_info     0x0000144c    0xa05 driver.o
169 .debug_info     0x00001e51    0x9f4 main.o
170 .debug_info     0x00002845    0xa0f Main_Function.o
171 .debug_info     0x00003254    0x9ff Pressure_Sensor.o
172 .debug_info     0x00003c53    0x19e startup.o
173
174 .debug_abbrev   0x00000000    0xbd8
175 .debug_abbrev   0x00000000    0x1f9 Alarm_Actuator.o
176 .debug_abbrev   0x000001f9    0x1d0 Alarm_Monitor.o
177 .debug_abbrev   0x000003c9    0x1de driver.o
178 .debug_abbrev   0x000005a7    0x1bc main.o
179 .debug_abbrev   0x00000763    0x1ee Main_Function.o
180 .debug_abbrev   0x00000951    0x1c7 Pressure_Sensor.o
181 .debug_abbrev   0x00000b18    0xc0 startup.o
182
183 .debug_loc      0x00000000    0x510
184 .debug_loc      0x00000000    0x124 Alarm_Actuator.o
185 .debug_loc      0x00000124    0x9c Alarm_Monitor.o
186 .debug_loc      0x000001c0    0x140 driver.o
187 .debug_loc      0x00000300    0x58 main.o
188 .debug_loc      0x00000358    0xcc Main_Function.o
189 .debug_loc      0x00000424    0x9c Pressure_Sensor.o
190 .debug_loc      0x000004c0    0x50 startup.o
191
192 .debug_aranges  0x00000000    0xe0
193 .debug_aranges  0x00000000    0x20 Alarm_Actuator.o
194 .debug_aranges  0x00000000    0x20 Alarm_Monitor.o
195 .debug_aranges  0x00000020    0x20 driver.o
196 .debug_aranges  0x00000040    0x20 main.o
197 .debug_aranges  0x00000060    0x20 Main_Function.o
198 .debug_aranges  0x00000080    0x20 Pressure_Sensor.o
199 .debug_aranges  0x000000a0    0x20 startup.o
200
201
202
203
204
205
206
207
208 .debug_line     0x00000000    0xa0
209 .debug_line     0x00000000    0x15a Alarm_Actuator.o
210 .debug_line     0x0000015a    0x161 Alarm_Monitor.o
211 .debug_line     0x000002bb    0x1c8 driver.o
212 .debug_line     0x00000483    0x195 main.o
213 .debug_line     0x00000618    0x15e Main_Function.o
214 .debug_line     0x00000776    0x152 Pressure_Sensor.o
215 .debug_line     0x000008c8    0x142 startup.o
216
217 .debug_str      0x00000000    0x7c8
218 .debug_str      0x00000000    0x551 Alarm_Actuator.o
219 .debug_str      0x00000000    0x5bf (size before relaxing)
220 .debug_str      0x00000551    0xa0 Alarm_Monitor.o
221 .debug_str      0x000005f1    0x5a4 (size before relaxing)
222 .debug_str      0x000005f1    0x57 driver.o
223 .debug_str      0x000005f1    0x59b (size before relaxing)
```

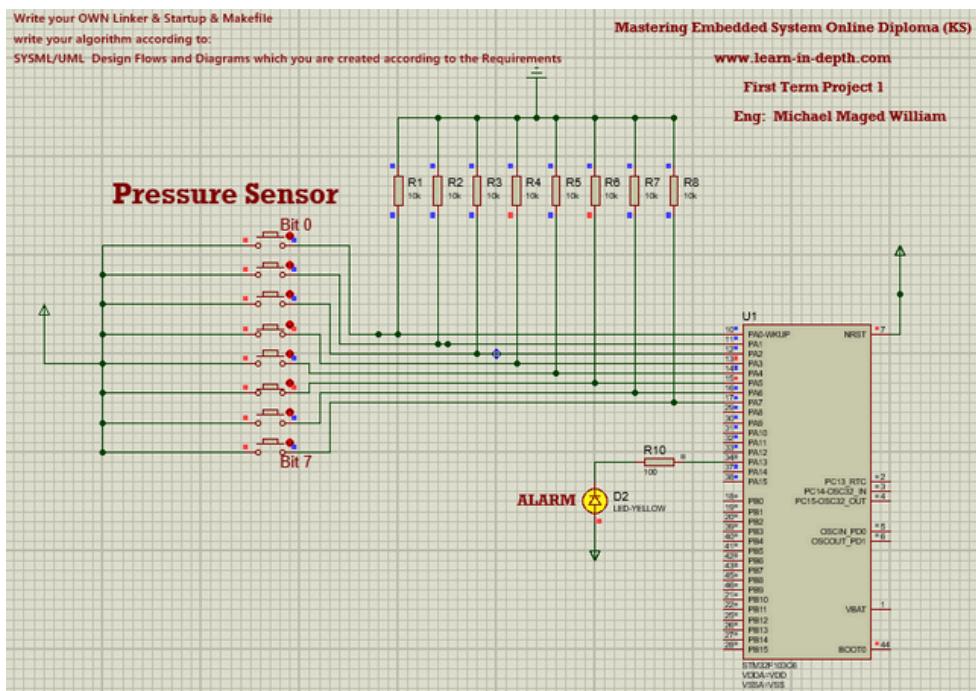
Master_EMBEDDED_Systems



Learn_In_Depth

```
217 .debug_str    0x00000000 0x7c8
218 .debug_str    0x00000000 0x551 Alarm_Actuator.o
219                                         0x5bf (size before relaxing)
220 .debug_str    0x00000551 0xa0 Alarm_Monitor.o
221                                         0x5a4 (size before relaxing)
222 .debug_str    0x000005f1 0x57 driver.o
223                                         0x59b (size before relaxing)
224 .debug_str    0x00000648 0x34 main.o
225                                         0x5ac (size before relaxing)
226 .debug_str    0x0000067c 0x64 Main_Function.o
227                                         0x5b2 (size before relaxing)
228 .debug_str    0x000006e0 0x5b Pressure_Sensor.o
229                                         0x5aa (size before relaxing)
230 .debug_str    0x0000073b 0x8d startup.o
231                                         0x1f7 (size before relaxing)
232
233 .debug_frame  0x00000000 0x30c
234 .debug_frame  0x00000000 0xac Alarm_Actuator.o
235 .debug_frame  0x000000ac 0x68 Alarm_Monitor.o
236 .debug_frame  0x00000114 0xa0 driver.o
237 .debug_frame  0x000001b4 0x48 main.o
238 .debug_frame  0x000001fc 0x74 Main_Function.o
239 .debug_frame  0x00000270 0x68 Pressure_Sensor.o
240 .debug_frame  0x000002d8 0x34 startup.o
241
```

Simulation using Proteus



Master_EMBEDDED_Systems



Learn_In_Depth

CM3 Source Code - U1

```
File: E:\First_Project_First_Term\driver.c
-----
----- #include "driver.h"
----- #include <stdint.h>
----- #include <stdio.h>
-----
----- void Delay(int nCount)
8000120 {
8000128     for(; nCount != 0; nCount--);
8000136 }
-----
8000140 int getPressureVal(){
8000144     return (GPIOA_IDR & 0xFF);
800014A }
-----
8000158 void Set_Alarm_actuator(int i){
8000160     if (i == 1){
8000166         SET_BIT(GPIOA_ODR,13);
8000174     } else if (i == 0){
800017A         RESET_BIT(GPIOA_ODR,13);
8000172 }
8000178 ... }
```

CM3 Source Code - U1

```
File: E:\First_Project_First_Term\Alarm_Monitor.c
-----
----- //=====
----- //define the pointer to store the state of alarm
----- void (*Pointer_Alarm_state)();
----- //defination the function that get the signal from
----- void High_Pressure_Detect()
80000CC {
80000D0     Pointer_Alarm_state=STATE_(start_alarm) ;
80000D6 }
----- //=====
-----
----- //the defination for each state in alarm
----- STATE_Define(stop_alarm)
80000E8 {
80000EC     State_Actuator(stop);
80000F2     Pointer_Alarm_state=STATE_(stop_alarm) ;
80000F8 }
----- STATE_Define(start_alarm)
8000104 {
8000108     State_Actuator(start);
800010E     Pointer_Alarm_state=STATE_(stop_alarm) ;
8000114 }
----- ...
```

**Thanks you Learn in Depth
Thanks you Eng-Keroloes Shenouda**



Master_EMBEDDED_Systems