Learn_In_Depth

First Term (Final Project 1)

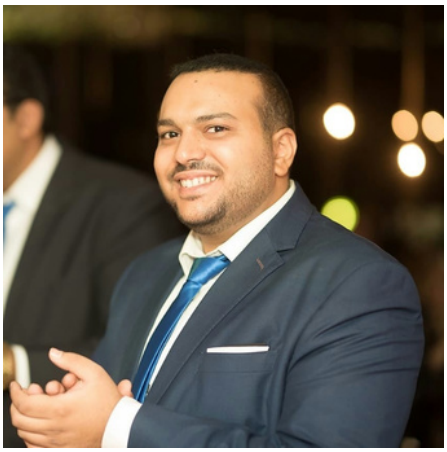
Pressure Controller Project

Name : Michael Maged William

My_Profile ⟶ 🔗

Under the supervision of
Engineer Keroloes Shenouda

Master_Embedded_Systems



Learn_In_Depth

Table Of Content :

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

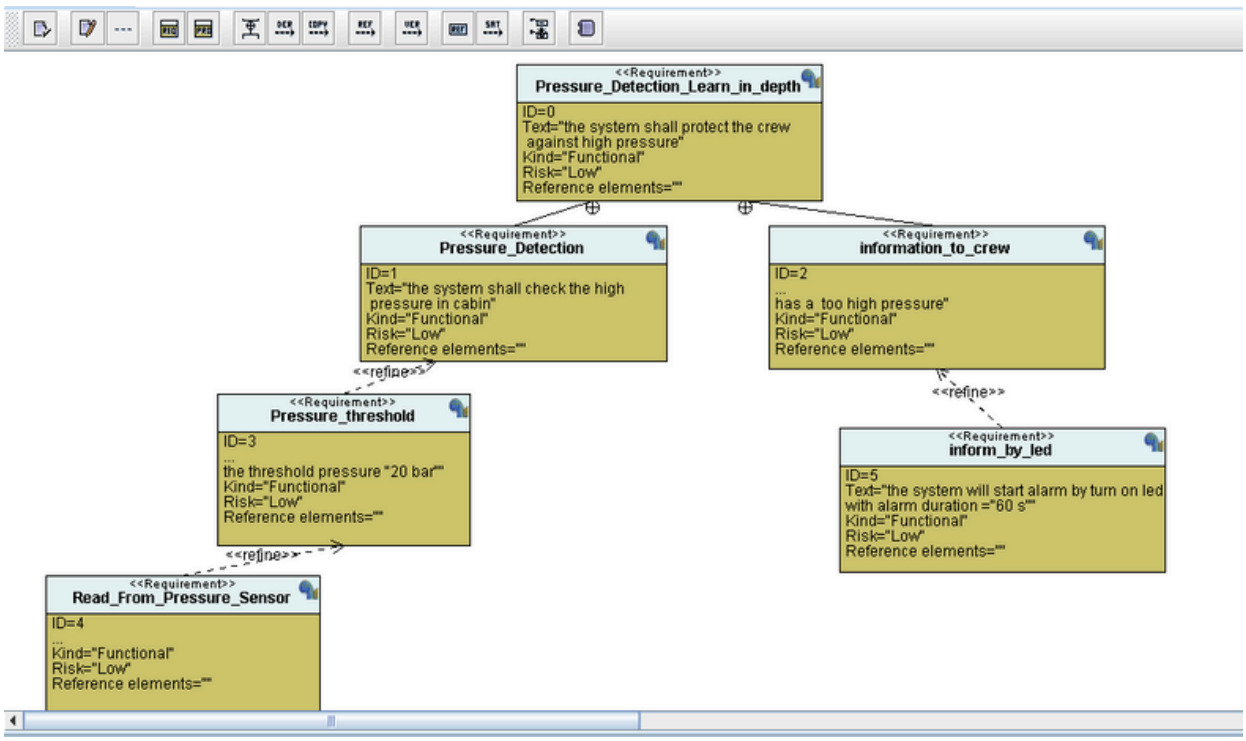
Learn_In_Depth

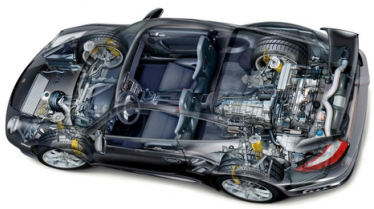
Requirement Diagram :

We will divide our case study to two main requirements

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

Each main requirement has refinement requirements as mentioned below





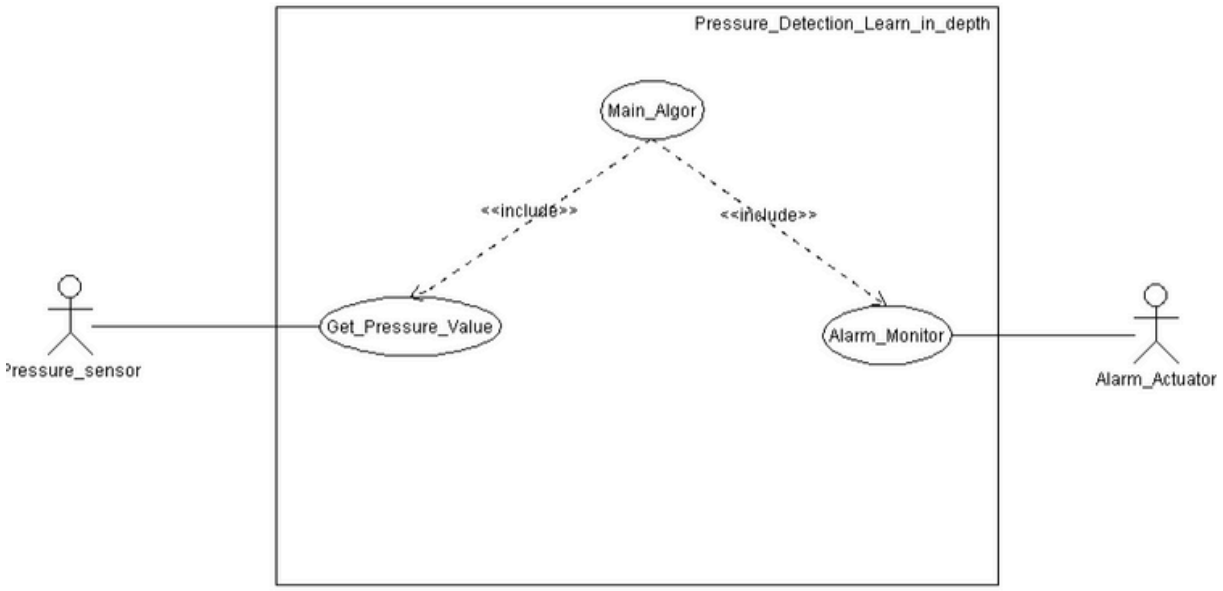
Master_Embedded_Systems

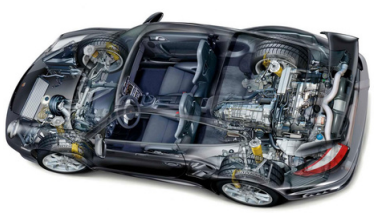
Learn_In_Depth

System Analysis Diagram :

- **Use Case Diagram :**

In use case diagram , we discuss system boundary and main functions , so in our use case digram 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

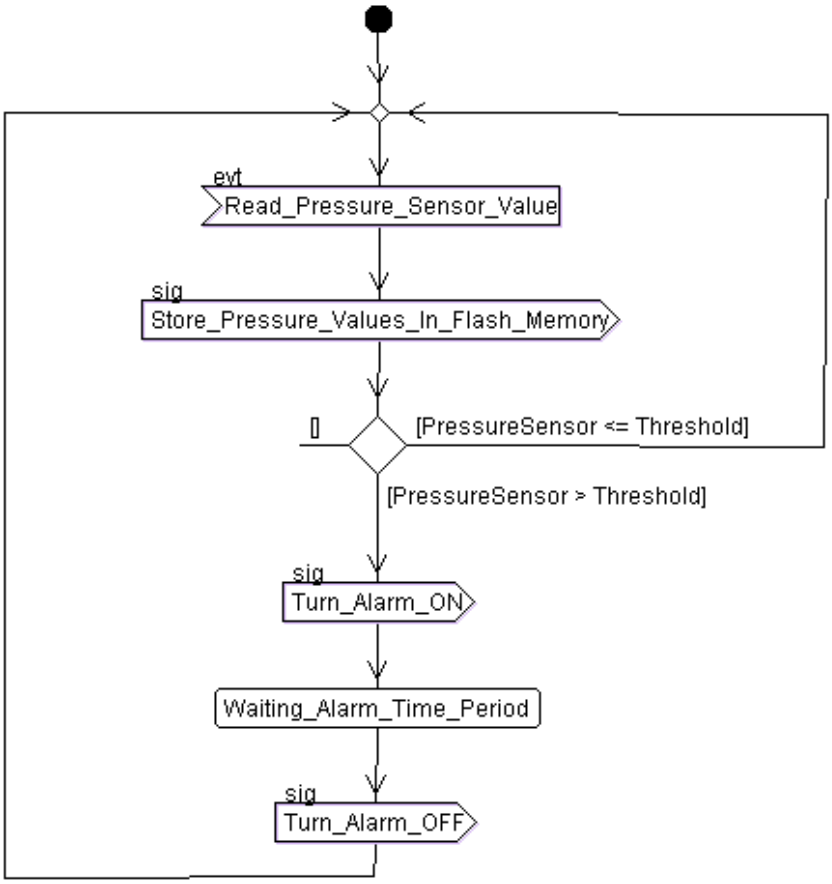




Master_Embedded_Systems

- **Activity Diagram :**

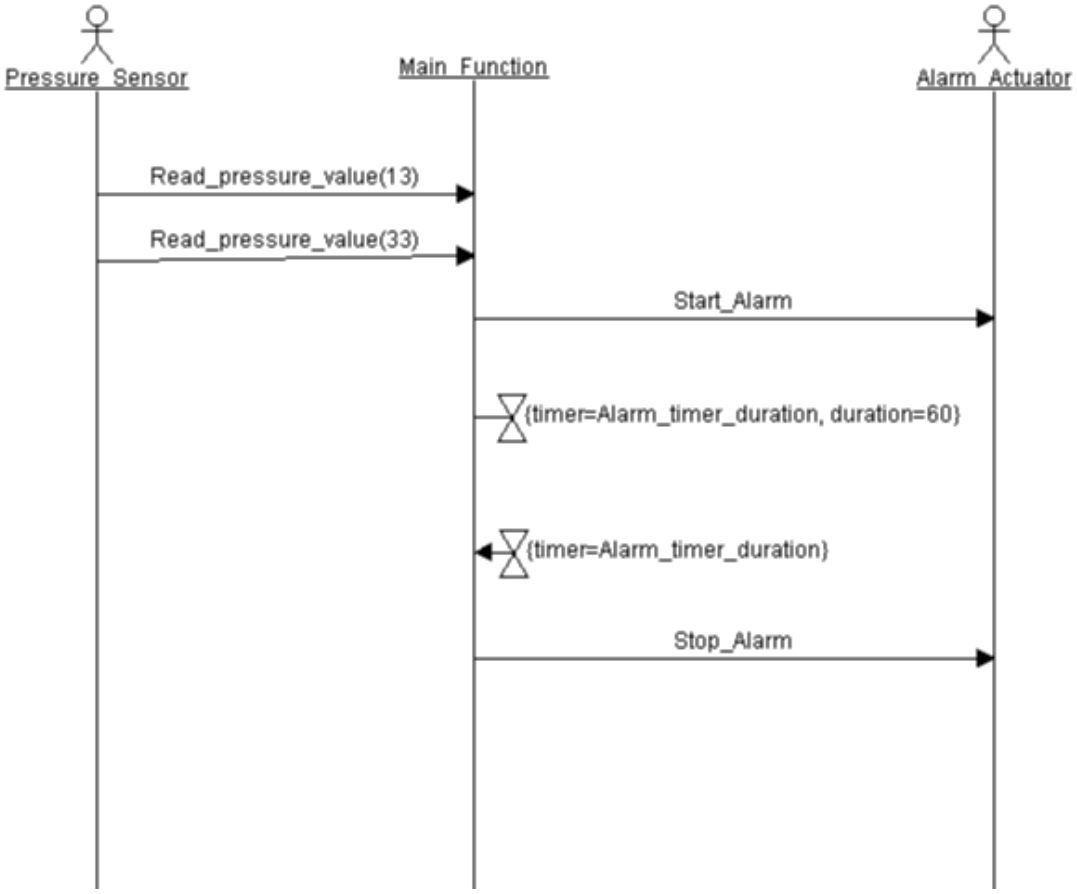
In Activity Diagram , we discuss relation between main functions , as shown below , firstly we will read the pressure value that we compare the value of pressure with threshold in case study , if it exceeded this value we will turn on the alarm for 60 seconds to inform the crew that the pressure exceeded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



- **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

---



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

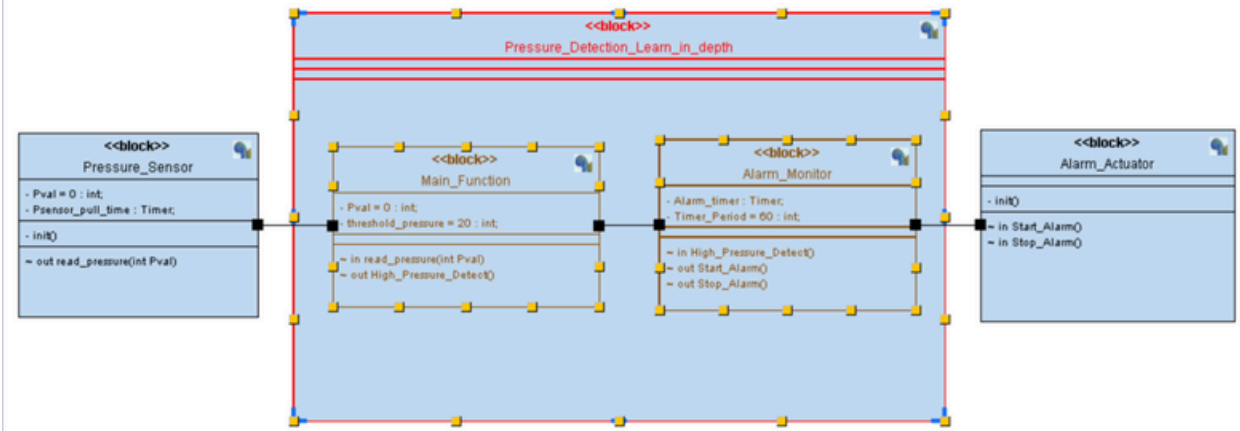
- **Alarm Monitor Module :**

  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

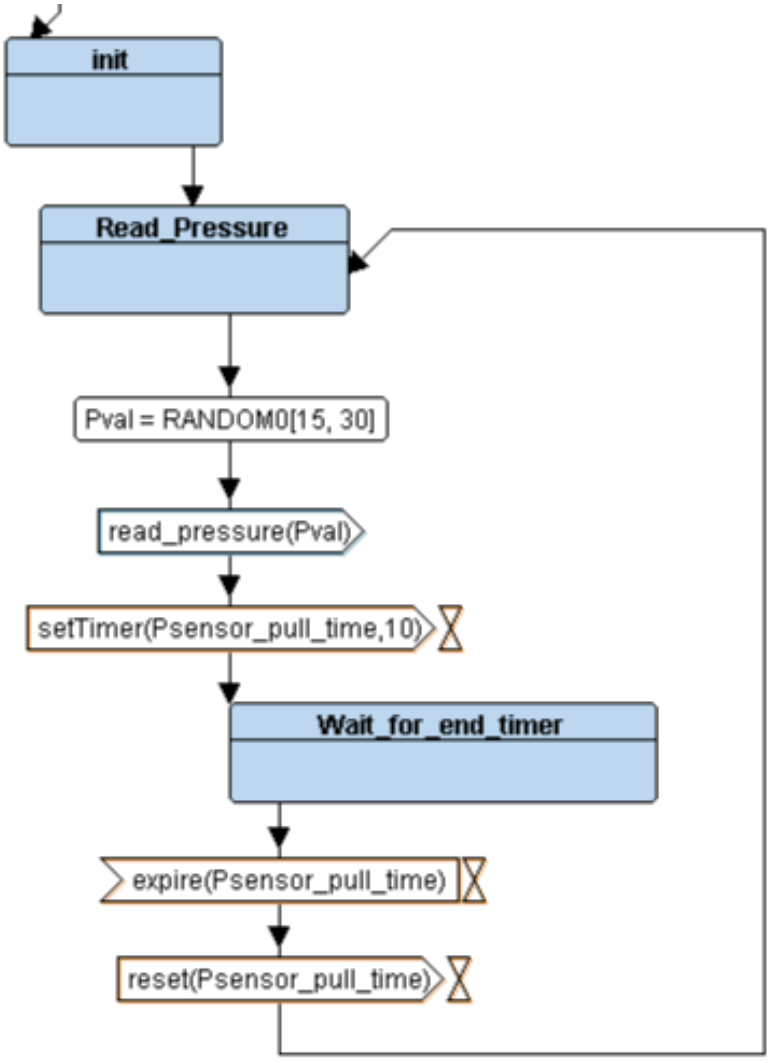
Block_Diagram



Learn_In_Depth

Source files with Block Diagrams

- **Pressure_Sensor _Diagram :**



Master_Embedded_Systems

Learn_In_Depth

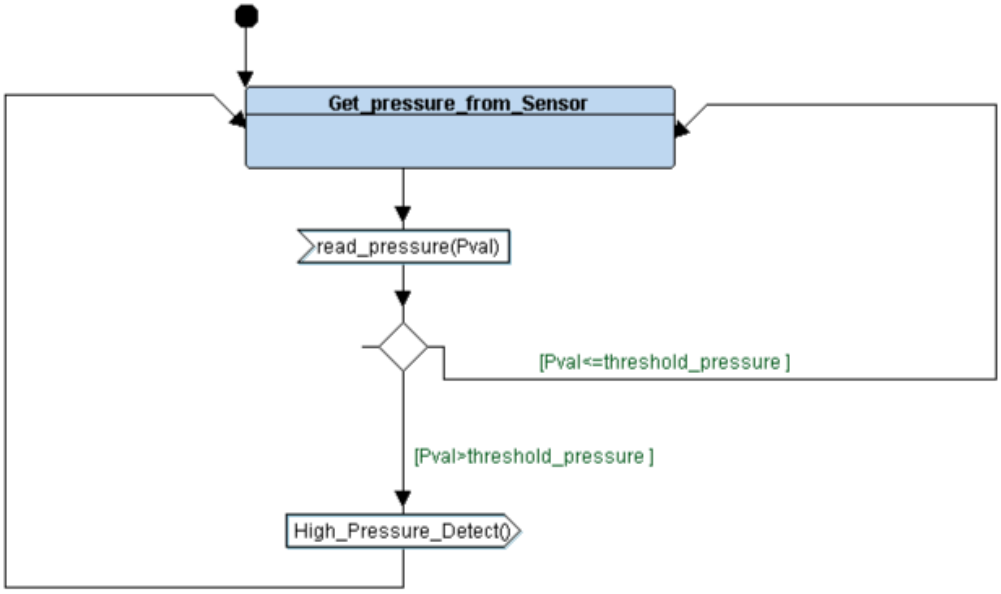
- **Pressure_Sensor _Header_File :**

```c
/*
 * Pressure_Sensor.h
 *
 *  Created on: Sep 15, 2024
 *      Author: Michael Maged
 */

#ifndef PRESSURE_SENSOR_H_
#define PRESSURE_SENSOR_H_
#include "driver.h"
//================================================
//Define States of Presure Sensor
typedef enum {
    waiting_state,
    reading_state
} state_sensor;
//================================================
// function to initialize the Pressure Sensor
void Pressure_Sensor_init();
//================================================
//declare the function of states
STATE_Define(waiting_state);
STATE_Define(reading_state);
//================================================
//declare the pointer to function to read from the main function
extern void (*Pointer_sensor)();
#endif /* PRESSURE_SENSOR_H_ */

```

- **Pressure_Sensor _Source_File :**

```c
/*
 * Pressure_Sensor.c
 *
 *  Created on: Sep 15, 2024
 *      Author: Michael Maged
 */
#include "driver.h"
#include "Pressure_Sensor.h"
//define the variables of block
int Pval=0,Psensor_pull_time=1000;
//declare the function is to be send to main function
void Read_Pressure(int Pval);
//define the pointer to function to store the current state of pressure sensor
void (*Pointer_sensor)();
//init function of pressure sensor
void Pressure_Sensor_init()
{
    Pointer_sensor=STATE_(reading_state);//by default the start of pressure sensor read of pressure
}
//================================================
//defination the states of Pressure Sensor
STATE_Define(waiting_state)
{
    Delay(Psensor_pull_time);//delay to waiting period and reading again
    Pointer_sensor=STATE_(reading_state);
}
//================================================
STATE_Define(reading_state)
{
    Pval=getPressureVal();
    Read_Pressure(Pval);
    Pointer_sensor=STATE_(waiting_state);
}
```

- **Main Function Diagram :**



- **Main Function Header File :**

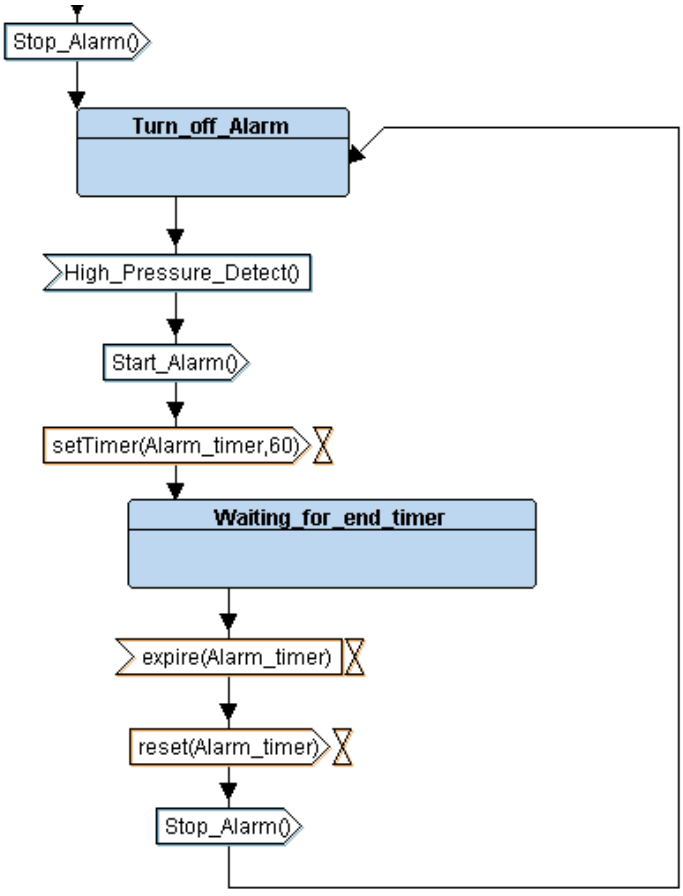
```c
/*
 * Main_Function.h
 *
 *  Created on: Sep 15, 2024
 *      Author: Michael Maged
 */

#ifndef MAIN_FUNCTION_H_
#define MAIN_FUNCTION_H_
#include "driver.h"
//define the states of the main function
typedef enum {
    main_waiting,
    main_detect_high_pressure
} State_Main;
//=====================================================
//declare the function for each states
STATE_Define(main_waiting);
STATE_Define(main_detect_high_pressure);
//=====================================================
//declare the pointer to function to read form all files
extern void (*Pointer_main_state)();

#endif /* MAIN_FUNCTION_H_ */

```

Learn_In_Depth

- ## Main Function Source File :

```c
/*
 * Main_Function.c
 *
 *  Created on: Sep 15, 2024
 *      Author: Michael maged
 */
#include"stdio.h"
#include "Main_Function.h"
//define the variables for this block
int P_value=0,P_threshold=20;
//declare the function to send signal to alarm in state detect the high pressure
void High_Pressure_Detect();
//define the pointer to function to store the state of main function
void (*Pointer_main_state)();
//================================================
//defination the function it read from pressure sensor
void Read_Pressure(int Pval)
{
    P_value=Pval;
    (P_value <= P_threshold)? (Pointer_main_state=STATE_(main_waiting)) : (Pointer_main_state=STATE_(main_detect_high_pressure));

}
//================================================
//defination the function for each states
STATE_Define(main_waiting)
{
    Pointer_main_state=STATE_(main_waiting);
}
STATE_Define(main_detect_high_pressure)
{
    High_Pressure_Detect();
    Pointer_main_state=STATE_(main_waiting);
}
//================================================
```

- ## Alarm Monitor Diagram :



Learn_In_Depth

- **Alarm Monitor Header File :**

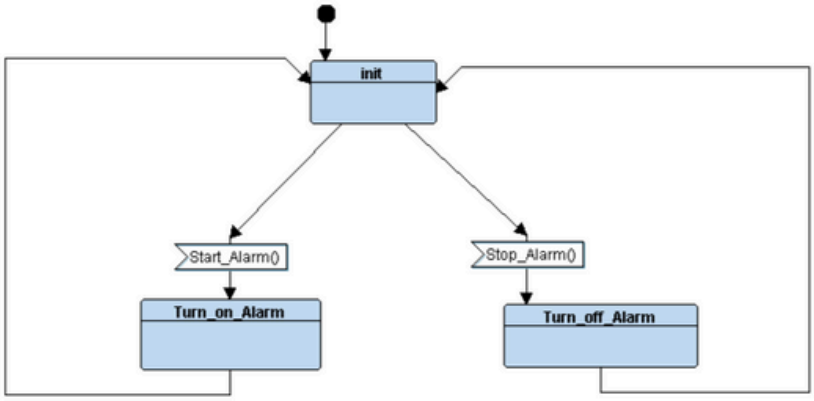
```c
/*
 * Alarm_Monitor.h
 *
 *  Created on: Sep 15, 2024
 *      Author: Michael Maged
 */

#ifndef ALARM_MONITOR_H_
#define ALARM_MONITOR_H_
#include "driver.h"
//define states of this block
typedef enum
{
    stop_alarm,
    start_alarm,
}State_Alarm;
//define the function for each state
STATE_Define(stop_alarm);
STATE_Define(start_alarm);
//declare the pointer to function to read form all files
extern void (*Pointer_Alarm_state)();

#endif /* ALARM_MONITOR_H_ */

```

- **Alarm Monitor Source File :**

```c
#include "Alarm_Monitor.h"
#include "Alarm_Actuator.h"

//=====================================================
//define the pointer to store the state of alarm
void (*Pointer_Alarm_state)();
//defination the function that get the signal from the main
void High_Pressure_Detect()
{
    Pointer_Alarm_state=STATE_(start_alarm) ;
}
//=====================================================

//the defination for each state in alarm
STATE_Define(stop_alarm)
{
    State_Actuator(stop);
    Pointer_Alarm_state=STATE_(stop_alarm) ;
}
STATE_Define(start_alarm)
{
    State_Actuator(start);
    Pointer_Alarm_state=STATE_(stop_alarm) ;
}

//=====================================================
```

Learn_In_Depth

- **Alarm_Actuator_Diagram :**



- **Alarm_Actuator_Header_File :**

```c
/*
 * Alarm_Actuator.h
 *
 *  Created on: Sep 15, 2024
 *      Author: Michael Maged
 */

#ifndef ALARM_ACTUATOR_H_
#define ALARM_ACTUATOR_H_
#include "driver.h"
//define the state of alarm actuator
typedef enum {
    start,
    stop,
    wait
} state_alarm;

//=====================================================
//define function to send state of alarm to alarm actuator
void State_Actuator(state_alarm state);
//=====================================================
//define the function of each state
STATE_Define(stop);
STATE_Define(start);
STATE_Define(wait);
extern void (*state_alarm_actuator)();
#endif /* ALARM_ACTUATOR_H_ */
```

Learn_In_Depth

- **Alarm_Actuator_Source_File :**

```c
/*
 * Alarm_Actuator.c
 *
 *  Created on: Sep 15, 2024
 *      Author: Michael maged
 */
#include "Alarm_Actuator.h"

//define pointer to store the state of alarm actuator
void (*state_alarm_actuator)();
//=====================================================

void Alarm_Actuator_init()
{
    state_alarm_actuator=STATE_(stop);
}
//=====================================================

void State_Actuator(state_alarm state)
{
    if( state == start)
    {
        state_alarm_actuator=STATE_(start);
    }
    else if(state == stop)
    {
        state_alarm_actuator=STATE_(stop);
    }
}
//=====================================================

STATE_Define(stop)
{
    Set_Alarm_actuator(1);
    state_alarm_actuator=STATE_(stop);
}
STATE_Define(start)
{
    Set_Alarm_actuator(0);
    state_alarm_actuator =STATE_(wait);
}
STATE_Define(wait)
{
    Delay(60000);
    state_alarm_actuator=STATE_(stop);
}
```

Learn_In_Depth

- **Driver_Header_file :**

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

// Automatic state function generated by macros
#define STATE_Define(_STATEFUNC_) void ST_##_STATEFUNC_()
#define STATE_(_stateFUN_)          ST_##_stateFUN_

#define SET_BIT(ADDRESS,BIT)   ADDRESS |=  (1<<BIT)
#define RESET_BIT(ADDRESS,BIT) ADDRESS &= ~(1<<BIT)
#define TOGGLE_BIT(ADDRESS,BIT)  ADDRESS ^=  (1<<BIT)
#define READ_BIT(ADDRESS,BIT) ((ADDRESS) &   (1<<(BIT)))


#define GPIO_PORTA 0x40010800
#define BASE_RCC   0x40021000

#define APB2ENR   *(volatile uint32_t *)(BASE_RCC + 0x18)

#define GPIOA_CRL *(volatile uint32_t *)(GPIO_PORTA + 0x00)
#define GPIOA_CRH *(volatile uint32_t *)(GPIO_PORTA + 0X04)
#define GPIOA_IDR *(volatile uint32_t *)(GPIO_PORTA + 0x08)
#define GPIOA_ODR *(volatile uint32_t *)(GPIO_PORTA + 0x0C)


void Delay(int nCount);
int getPressureVal();
void Set_Alarm_actuator(int i);
void GPIO_INITIALIZATION ();

```

- **Driver_Source_file :**

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

void Delay(int nCount)
{
    for(; nCount != 0; nCount--);
}

int getPressureVal(){
```


```c
int getPressureVal(){
    return (GPIOA_IDR & 0xFF);
}

void Set_Alarm_actuator(int i){
    if (i == 1){
        SET_BIT(GPIOA_ODR,13);
    }
    else if (i == 0){
        RESET_BIT(GPIOA_ODR,13);
    }
}

void GPIO_INITIALIZATION (){
    SET_BIT(APB2ENR, 2);
    GPIOA_CRL &= 0xFF0FFFFF;
    GPIOA_CRL |= 0x00000000;
    GPIOA_CRH &= 0xFF0FFFFF;
    GPIOA_CRH |= 0x22222222;
}
```

- **Main_Program_File :**

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

#include "Alarm_Monitor.h"
#include "Alarm_Actuator.h"
#include "Pressure_Sensor.h"
#include "Main_Function.h"
void setup()
{
    GPIO_INITIALIZATION();
    Pressure_Sensor_init();
    Pointer_main_state = STATE_(main_waiting);
    Pointer_Alarm_state = STATE_(stop_alarm);
    Alarm_Actuator_init();
}
```

Learn_In_Depth

```c
int main (){
    setup();

    while (1)
    {
        Pointer_sensor();
        Pointer_main_state();
        Pointer_Alarm_state();
        state_alarm_actuator();
    }

}
```

- **Make_File :**

```
#eng: Michael Maged
CC=arm-none-eabi-
CFLAGS= -mcpu=cortex-m4 -gdwarf-2 -g
INCS=-I .
LIBS=
SRC =$(wildcard *.c)
OBJ= $(SRC:.c=.o)
As =$(wildcard *.s)
AsOBJ = $(As:.s=.o)
Project_name=Pressure_Detection_Learn_in_depth
all:$(Project_name).bin
    @echo ======Build is Done======

%.o:%.c
        $(CC)gcc.exe -c $(INCS)   $(CFLAGS) $< -o $@

$(Project_name).elf: $(OBJ) $(AsOBJ)
    $(CC)ld.exe -T linker_script.ld $(LIBS) $(OBJ) $(AsOBJ) -o $(Project_name).elf -Map=Map_file.map
    cp $(Project_name).elf  $(Project_name).axf

$(Project_name).bin: $(Project_name).elf
    $(CC)objcopy.exe -O binary $< $@

```

Master_Embedded_Systems

- **Startup Program File :**

```c
// Startup.c
// Eng.Michael
//First_Project__High_Pressure_Detection
#include <stdint.h>
extern int main(void);
/*****************************************************************************/
// External symbols from startup file (locator)

extern uint32_t S_TEXT      ;
extern uint32_t E_TEXT      ;
extern uint32_t S_DATA      ;
extern uint32_t E_DATA      ;
extern uint32_t S_BSS       ;
extern uint32_t E_BSS       ;
extern uint32_t _STACK_TOP  ;


/*****************************************************************************/

// Defualt Handler to handel any interupt and to laylout the memory boundries

void Default_Handler(void)
{
    // Copy the data section from Flash to Sram

    uint32_t DATA_SIZE = (uint8_t *) &E_DATA - (uint8_t *) &S_DATA ;
    uint8_t* ptr_SOURCE = (uint8_t *) &E_TEXT ;
    uint8_t* ptr_DISTINATION = (uint8_t *) &S_DATA ;
    for(uint32_t i = 0 ; i < DATA_SIZE ; i++ )
    {
        *((uint8_t *)(ptr_DISTINATION++)) = *((uint8_t *)(ptr_SOURCE++)) ;
    }
   // Inilialize .bss section by zeros
    uint32_t BSS_SIZE = (uint8_t *) &E_BSS - (uint8_t *) &S_BSS ;
      for(uint32_t i = 0 ; i < BSS_SIZE ; i++ )
    {
        *((uint8_t *)(ptr_DISTINATION++)) = ((uint8_t)(0)) ;
    }


    main();
}

/*****************************************************************************/

// some Interupts prototypes may be occuer at runtime execuation

void Reset_Handler(void)         __attribute__((weak,alias("Default_Handler")));

void NMI_Handler(void)           __attribute__((weak,alias("Default_Handler")));

void H_Fault_Handler(void)       __attribute__((weak,alias("Default_Handler")));

void MM_Fault_Handler(void)     __attribute__((weak,alias("Default_Handler")));

void Bus_Fault(void)            __attribute__((weak,alias("Default_Handler")));

void Usage_Fault_Handler(void)  __attribute__((weak,alias("Default_Handler")));

/*****************************************************************************/

// Initialize the stack pointer
// define the entry point to progam to execute the main function using Default_Handler interupt
// set the other used interupts

uint32_t vectors[] __attribute__((section(".vectors")))=
{
    (uint32_t ) &_STACK_TOP          ,
    (uint32_t ) &Reset_Handler       ,
    (uint32_t ) &NMI_Handler         ,
    (uint32_t ) &H_Fault_Handler     ,
    (uint32_t ) &MM_Fault_Handler    ,
    (uint32_t ) &Bus_Fault           ,
    (uint32_t ) &Usage_Fault_Handler


};
/*****************************************************************************/
```

- **Linker script.ld :**

```
/*****************************************************************************
 * @file           : linker_script.ld
 * engineer        : Michael maged
 *****************************************************************************/
/* Memory Types and lengthes */
MEMORY
{

  FLASH(RX): ORIGIN = 0X08000000 , LENGTH = 128K
  SRAM(RWX): ORIGIN = 0X20000000 , LENGTH = 20K

}


SECTIONS
{
    .text :
    {
      S_TEXT = . ;
      *(.vectors*)
      *(.text*)
      *(.rodata)
      E_TEXT = . ;

    }> FLASH
    .data :
    {
      S_DATA = . ;
      *(.data)
      . = ALIGN(4) ;
      E_DATA = . ;

    }> SRAM AT> FLASH
```

```
      .bss :
      {
        S_BSS = . ;
        *(.bss)
        E_BSS = . ;
        . = ALIGN(4) ;
        . = . + 0x1000 ;
        _STACK_TOP = . ;
      }> SRAM
      .comment :
      {

        *(.comment) *(COMMON)

      }> FLASH


}
```

Learn_In_Depth

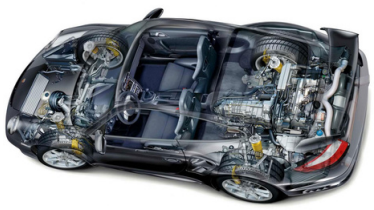
Sections and symbols for object files

- **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  Algn
  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  00000aa7  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 Embe
$ 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
```



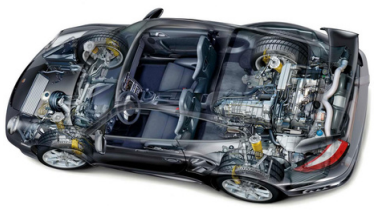
Learn_In_Depth

- **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  Algn
  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  00000cb5  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 00000020  00000000  00000000  00000d81  2**0
                  CONTENTS, RELOC, READONLY, DEBUGGING
  7 .debug_line   0000015e  00000000  00000000  00000da1  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 Embed
$ 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

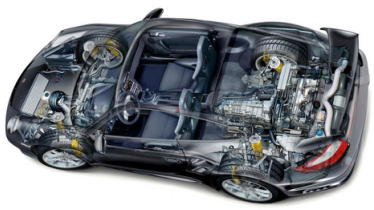
Learn_In_Depth

- **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  Algn
  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  00000cfc  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
```



Master_Embedded_Systems

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  Algn
  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  00000fbf  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  Algn
  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 (or
/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
```


```
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            0x08000000         0x00020000         xr
16  SRAM             0x20000000         0x00005000         xrw
17  *default*        0x00000000         0xffffffff
18
19  Linker script and memory map
20
21
22  .text           0x08000000      0x3b8
23                  0x08000000                S_TEXT = .
24   *(.vectors*)
25   .vectors       0x08000000       0x1c startup.o
26                  0x08000000                vectors
27   *(.text*)
28   .text          0x0800001c       0xb0 Alarm_Actuator.o
29                  0x0800001c                Alarm_Actuator_init
30                  0x08000038                State_Actuator
31                  0x08000074                ST_stop
32                  0x08000090                ST_start
33                  0x080000ac                ST_wait
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)
63                 0x080003b8            E_TEXT = .
```

```
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
```

Master_Embedded_Systems

Learn_In_Depth

```
                0x20000000                P_threshold
 .data          0x20000004        0x4 Pressure_Sensor.o
                0x20000004                Psensor_pull_time
 .data          0x20000008        0x0 startup.o
                0x20000008                . = ALIGN (0x4)
                0x20000008                E_DATA = .

.igot.plt       0x20000008        0x0 load address 0x080003c0
 .igot.plt      0x20000008        0x0 Alarm_Actuator.o

.bss            0x20000008     0x1008 load address 0x080003c0
                0x20000008                S_BSS = .
 *(.bss)
 .bss           0x20000008        0x0 Alarm_Actuator.o
 .bss           0x20000008        0x0 Alarm_Monitor.o
 .bss           0x20000008        0x0 driver.o
 .bss           0x20000008        0x0 main.o
 .bss           0x20000008        0x4 Main_Function.o
                0x20000008                P_value
 .bss           0x2000000c        0x4 Pressure_Sensor.o
                0x2000000c                Pval
 .bss           0x20000010        0x0 startup.o
                0x20000010                E_BSS = .
                0x20000010                . = ALIGN (0x4)
                0x20001010                . = (. + 0x1000)
 *fill*         0x20000010     0x1000
                0x20001010                _STACK_TOP = .

.comment        0x080003c0       0x90
 *(.comment)
 .comment       0x080003c0       0x7e Alarm_Actuator.o
                                 0x7f (size before relaxing)
 .comment       0x0800043e       0x7f Alarm_Monitor.o
 .comment       0x0800043e       0x7f driver.o
 .comment       0x0800043e       0x7f main.o
 .comment       0x0800043e       0x7f Main_Function.o
 .comment       0x0800043e       0x7f Pressure_Sensor.o
 .comment       0x0800043e       0x7f startup.o
 *(COMMON)
 *fill*         0x0800043e        0x2
 COMMON         0x08000440        0x4 Alarm_Actuator.o
                0x08000440                state_alarm_actuator
 COMMON         0x08000444        0x4 Alarm_Monitor.o
                0x08000444                Pointer_Alarm_state
 COMMON         0x08000448        0x4 Main_Function.o
                0x08000448                Pointer_main_state
 COMMON         0x0800044c        0x4 Pressure_Sensor.o
                0x0800044c                Pointer_sensor
LOAD Alarm_Actuator.o
LOAD Alarm_Monitor.o
LOAD driver.o
LOAD main.o
LOAD Main_Function.o
LOAD Pressure_Sensor.o
LOAD startup.o
OUTPUT(Pressure_Detection_Learn_in_depth.elf elf32-littlearm)

.ARM.attributes
                0x00000000       0x33
 .ARM.attributes
                0x00000000       0x33 Alarm_Actuator.o
 .ARM.attributes
                0x00000033       0x33 Alarm_Monitor.o
 .ARM.attributes
                0x00000066       0x33 driver.o
 .ARM.attributes
                0x00000099       0x33 main.o
```

Master_Embedded_Systems

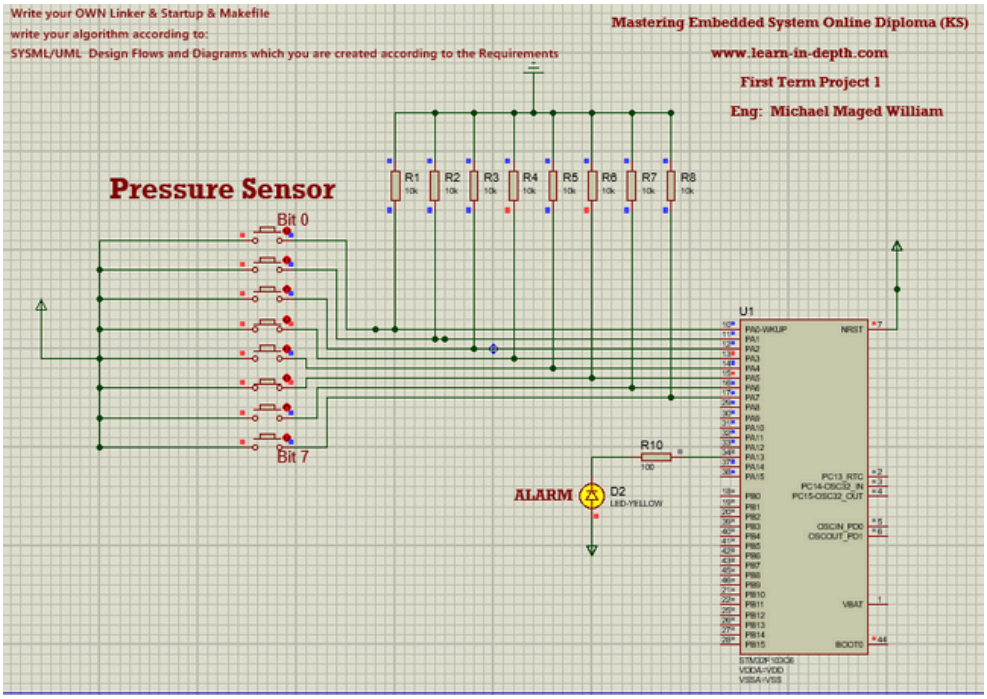

```
                0x00000099       0x33 main.o
 .ARM.attributes
                0x000000cc       0x33 Main_Function.o
 .ARM.attributes
                0x000000ff       0x33 Pressure_Sensor.o
 .ARM.attributes
                0x00000132       0x33 startup.o

.debug_info     0x00000000     0x3df1
 .debug_info    0x00000000      0xa44 Alarm_Actuator.o
 .debug_info    0x00000a44      0xa08 Alarm_Monitor.o
 .debug_info    0x0000144c      0xa05 driver.o
 .debug_info    0x00001e51      0x9f4 main.o
 .debug_info    0x00002845      0xa0f Main_Function.o
 .debug_info    0x00003254      0x9ff Pressure_Sensor.o
 .debug_info    0x00003c53      0x19e startup.o

.debug_abbrev   0x00000000      0xbd8
 .debug_abbrev  0x00000000      0x1f9 Alarm_Actuator.o
 .debug_abbrev  0x000001f9      0x1d0 Alarm_Monitor.o
 .debug_abbrev  0x000003c9      0x1de driver.o
 .debug_abbrev  0x000005a7      0x1bc main.o
 .debug_abbrev  0x00000763      0x1ee Main_Function.o
 .debug_abbrev  0x00000951      0x1c7 Pressure_Sensor.o
 .debug_abbrev  0x00000b18       0xc0 startup.o

.debug_loc      0x00000000      0x510
 .debug_loc     0x00000000      0x124 Alarm_Actuator.o
 .debug_loc     0x00000124       0x9c Alarm_Monitor.o
 .debug_loc     0x000001c0      0x140 driver.o
 .debug_loc     0x00000300       0x58 main.o
 .debug_loc     0x00000358       0xcc Main_Function.o
 .debug_loc     0x00000424       0x9c Pressure_Sensor.o
 .debug_loc     0x000004c0       0x50 startup.o
```

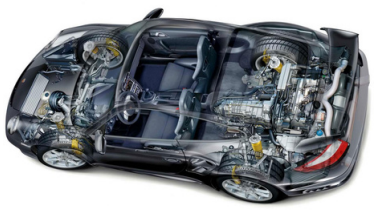
```
 .debug_loc     0x000004c0       0x50 startup.o

.debug_aranges  0x00000000       0xe0
 .debug_aranges
                0x00000000       0x20 Alarm_Actuator.o
 .debug_aranges
                0x00000020       0x20 Alarm_Monitor.o
 .debug_aranges
                0x00000040       0x20 driver.o
 .debug_aranges
                0x00000060       0x20 main.o
 .debug_aranges
                0x00000080       0x20 Main_Function.o
 .debug_aranges
                0x000000a0       0x20 Pressure_Sensor.o
 .debug_aranges
                0x000000c0       0x20 startup.o

.debug_line     0x00000000      0xa0a
 .debug_line    0x00000000      0x15a Alarm_Actuator.o
 .debug_line    0x0000015a      0x161 Alarm_Monitor.o
 .debug_line    0x000002bb      0x1c8 driver.o
 .debug_line    0x00000483      0x195 main.o
 .debug_line    0x00000618      0x15e Main_Function.o
 .debug_line    0x00000776      0x152 Pressure_Sensor.o
 .debug_line    0x000008c8      0x142 startup.o

.debug_str      0x00000000      0x7c8
 .debug_str     0x00000000      0x551 Alarm_Actuator.o
                                0x5bf (size before relaxing)
 .debug_str     0x00000551       0xa0 Alarm_Monitor.o
                                0x5a4 (size before relaxing)
 .debug_str     0x000005f1       0x57 driver.o
                                0x59b (size before relaxing)
```

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

ıe_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  }
--------
```

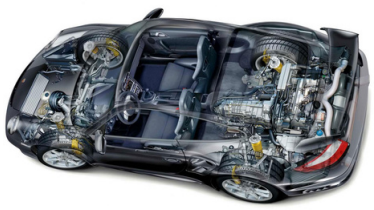
CM3 Source Code - U1

ıe_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 frc
-------- 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