Mastering Embedded System Online Diploma

www.learn-in-depth.com

First Term (Final Project 1)

Eng. Omar Khaled M. Anwer

My Profile:

Contents

Case Study

Specifications:

- A pressure controller should inform the crew of a cabin
  with an alarm when the pressure exceeds 20 bars in the cabin.
- The alarm duration equals 60 seconds.
- The system keeps tracking the measured values.

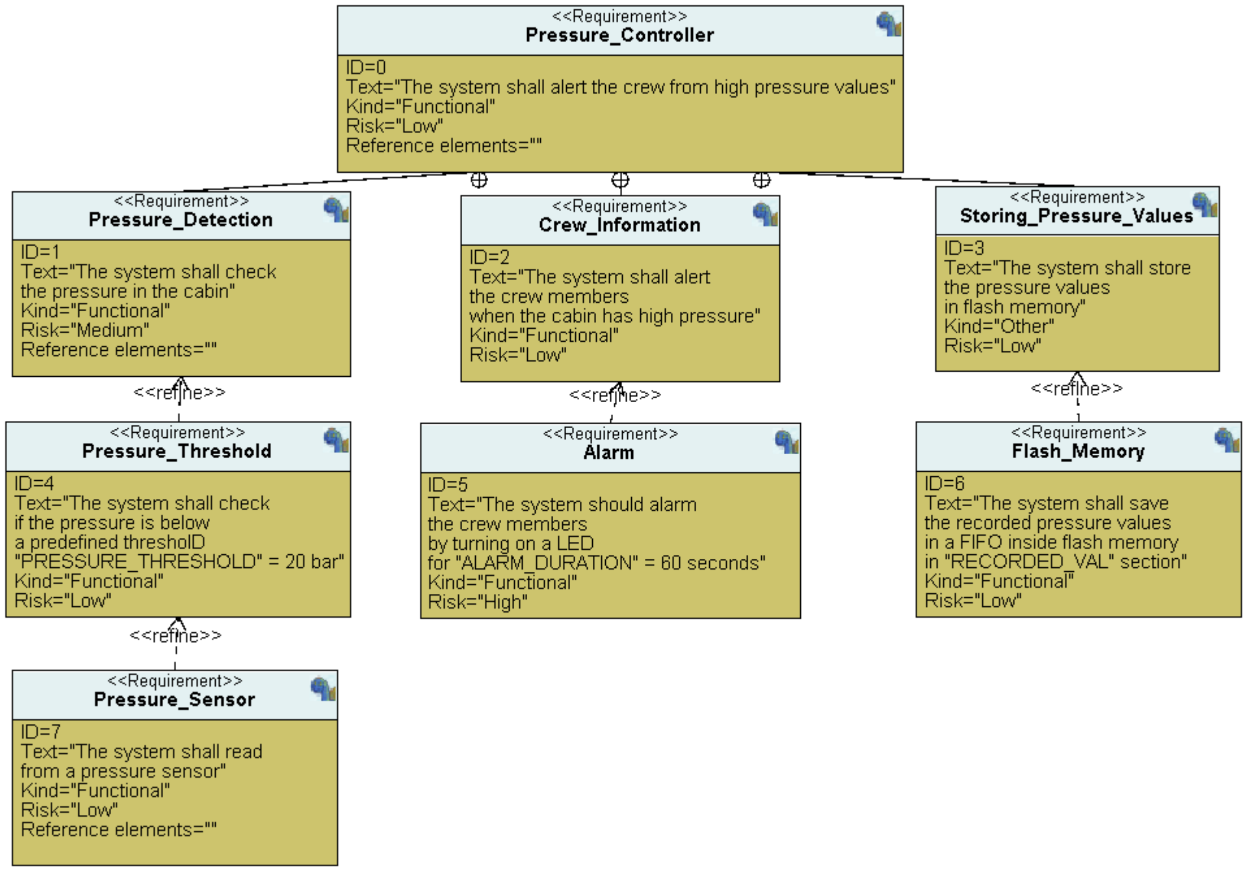
Pressure Controller Assumptions

- The controller set up 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.

Versioning:

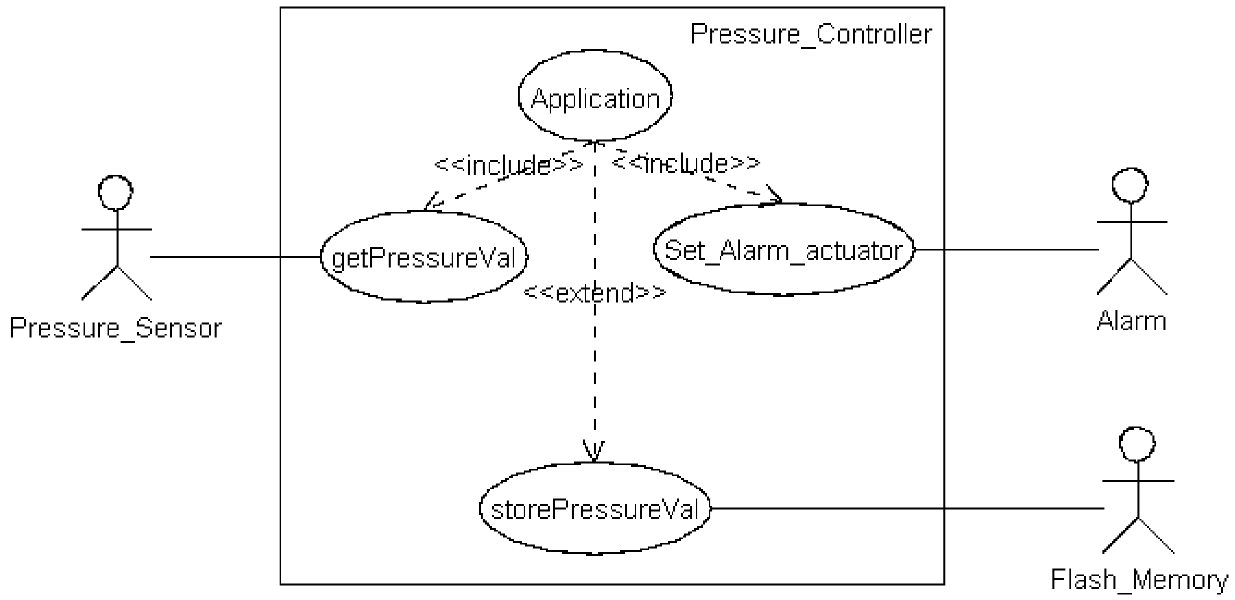
The “keeps tracking the measured values” option is not modeled in the first version of the design.

Requirement Diagram

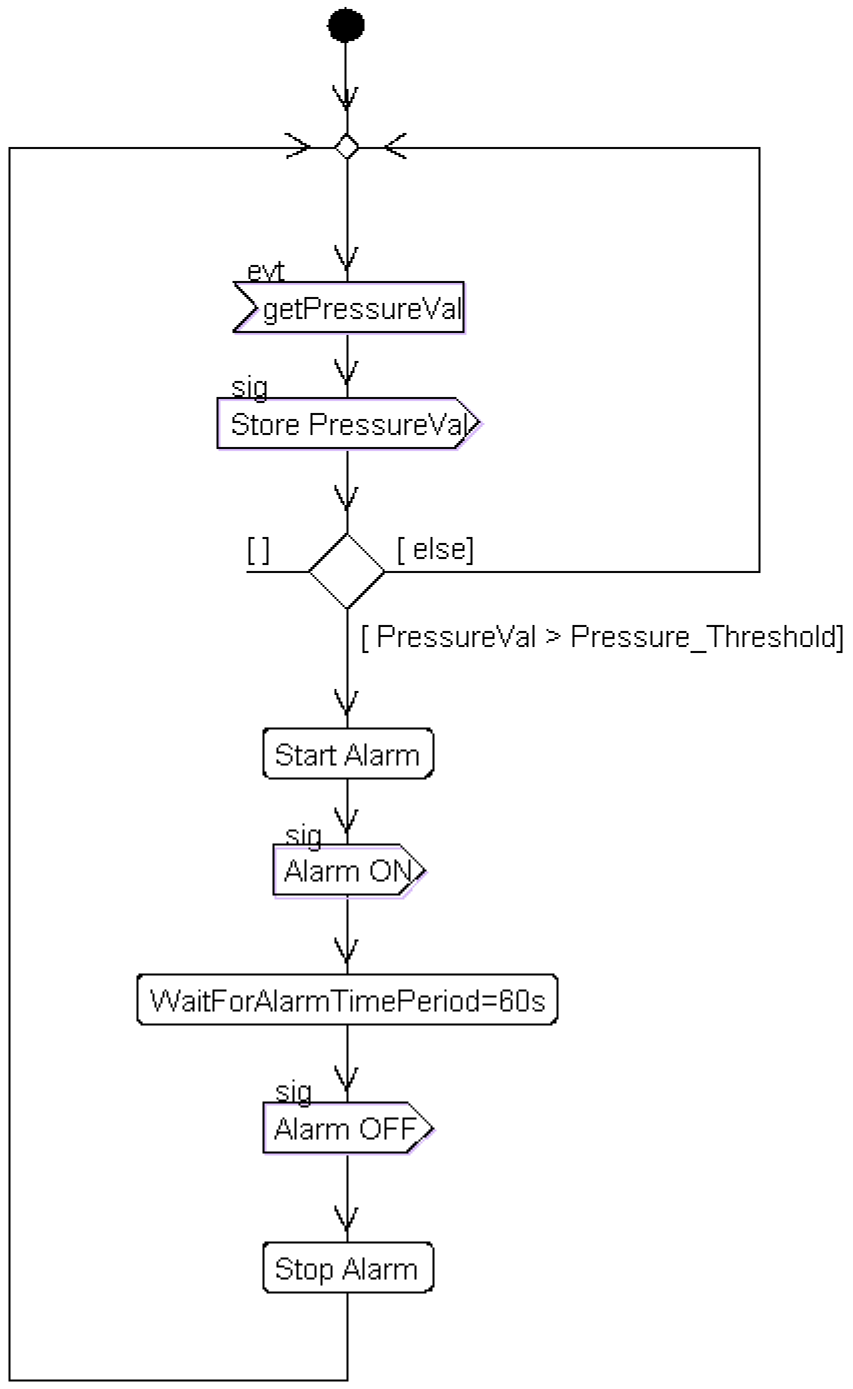

<<Requirement>>
Pressure_Controller
ID=0
Text="The system shall alert the crew from high pressure values"
Kind="Functional"
Risk="Low"
Reference elements=""
<<Requirement>>
Pressure_Detection
ID=1
Text="The system shall check the pressure in the cabin"
Kind="Functional"
Risk="Medium"
Reference elements=""
<<Requirement>>
Crew_Information
ID=2
Text="The system shall alert the crew members when the cabin has high pressure"
Kind="Functional"
Risk="Low"
<<Requirement>>
Storing_Pressure_Values
ID=3
Text="The system shall store the pressure values in flash memory"
Kind="Other"
Risk="Low"
<<refine>>
<<refine>>
<<refine>>
<<Requirement>>
Pressure_Threshold
ID=4
Text="The system shall check if the pressure is below a predefined thresholD "PRESSURE_THRESHOLD" = 20 bar"
Kind="Functional"
Risk="Low"
<<Requirement>>
Alarm
ID=5
Text="The system should alarm the crew members by turning on a LED for "ALARM_DURATION" = 60 seconds"
Kind="Functional"
Risk="High"
<<Requirement>>
Flash_Memory
ID=6
Text="The system shall save the recorded pressure values in a FIFO inside flash memory in "RECORDED_VAL" section"
Kind="Functional"
Risk="Low"
<<refine>>
<<Requirement>>
Pressure_Sensor
ID=7
Text="The system shall read from a pressure sensor"
Kind="Functional"
Risk="Low"
Reference elements=""

System Analysis

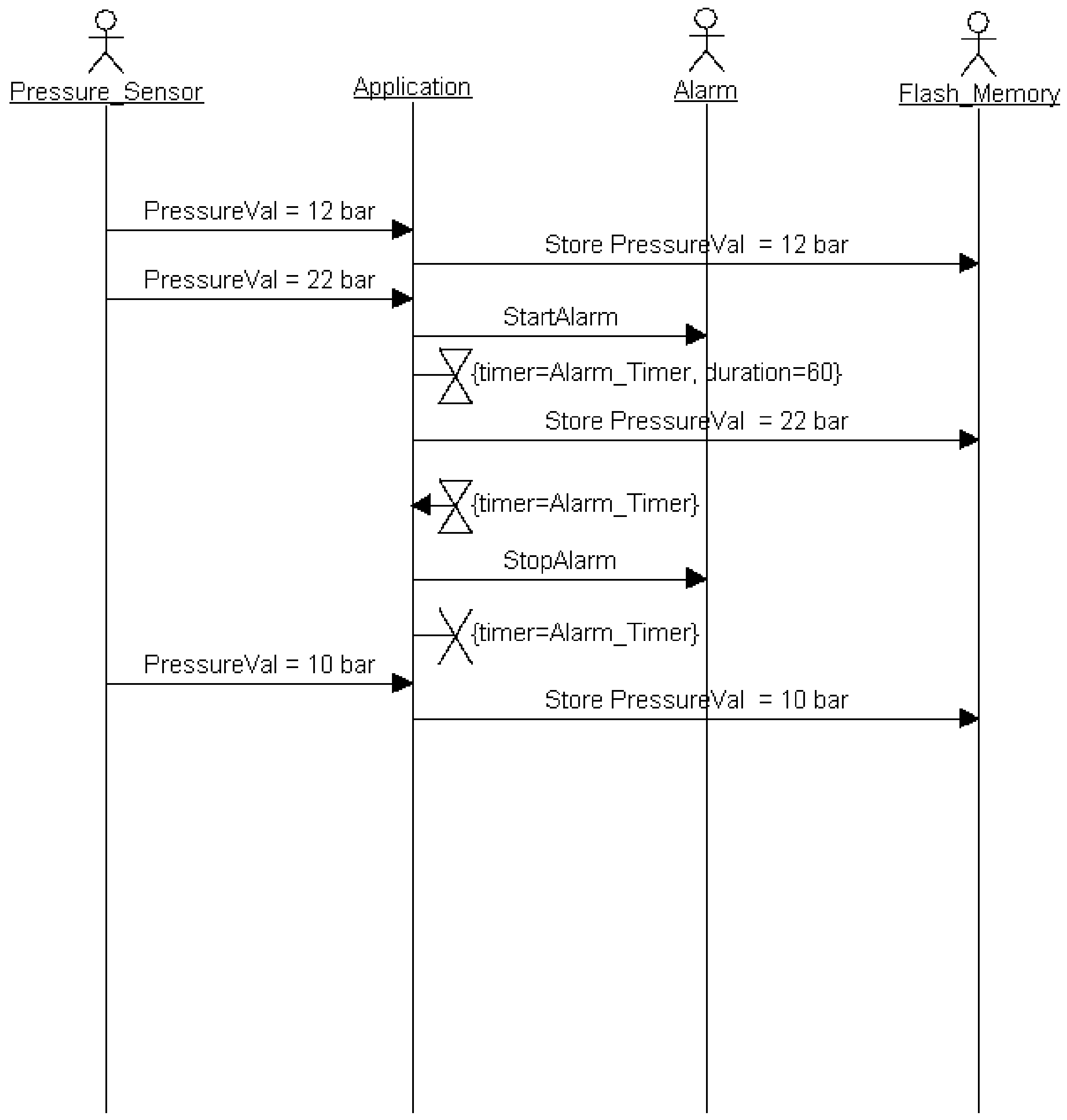
Use Case Diagram



Activity Diagram

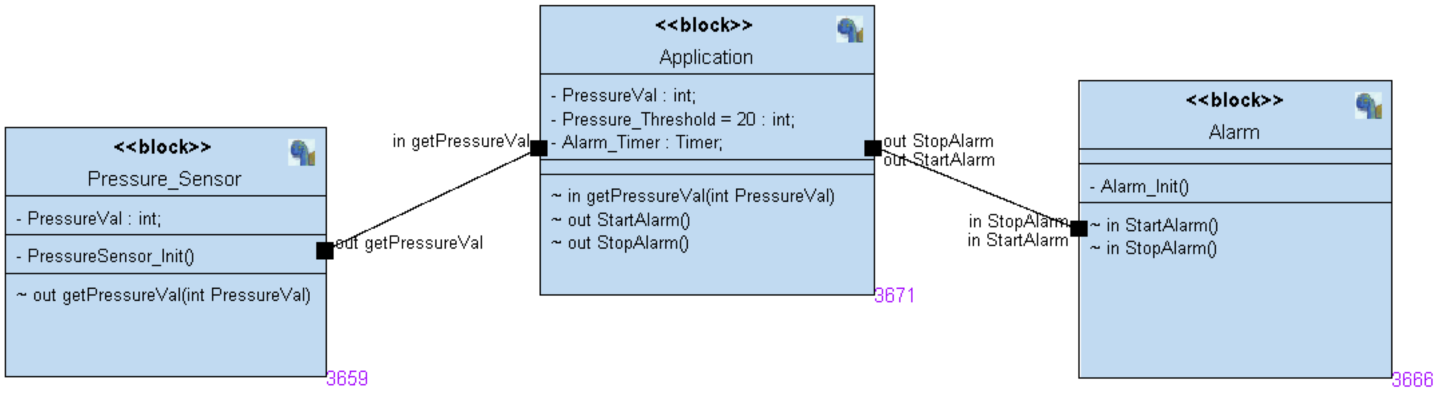


Sequence Diagram



This sequence diagram describes a scenario of the system.

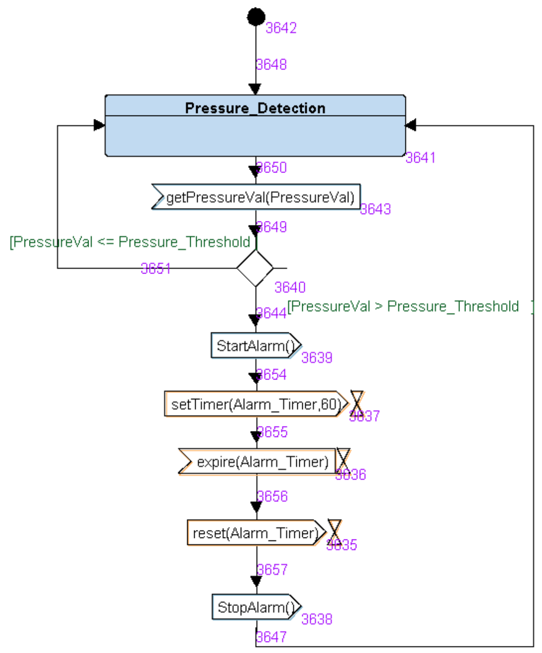
System Design

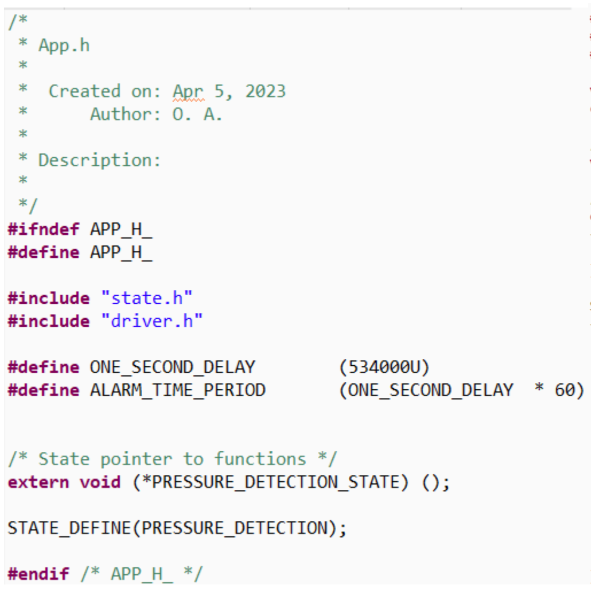


System Components

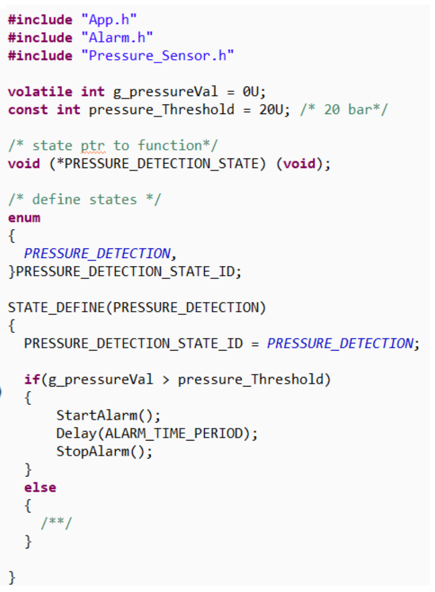
- Pressure sensor
- Alarm actuator
- Main application

Application



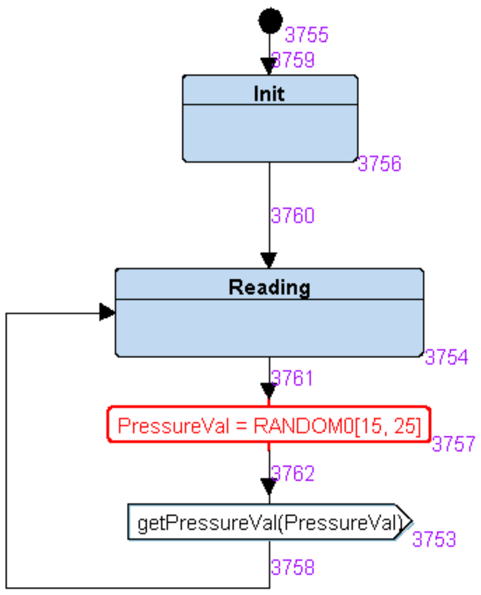


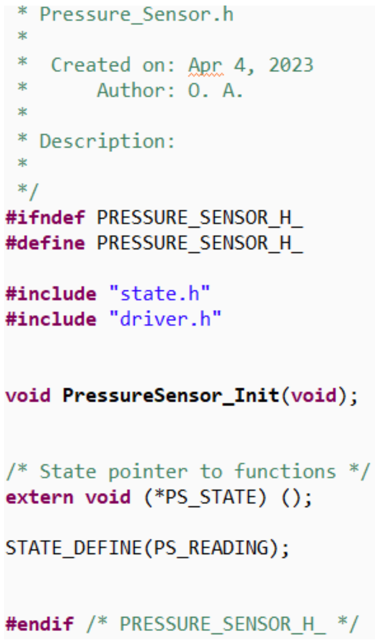
```
/*
 * App.h
 *
 *  Created on: Apr 5, 2023
 *      Author: O. A.
 *
 * Description:
 *
 */
#ifndef APP_H_
#define APP_H_

#include "state.h"
#include "driver.h"

#define ONE_SECOND_DELAY        (534000U)
#define ALARM_TIME_PERIOD       (ONE_SECOND_DELAY  * 60)


/* State pointer to functions */
extern void (*PRESSURE_DETECTION_STATE) ();

STATE_DEFINE(PRESSURE_DETECTION);

#endif /* APP_H_ */
```



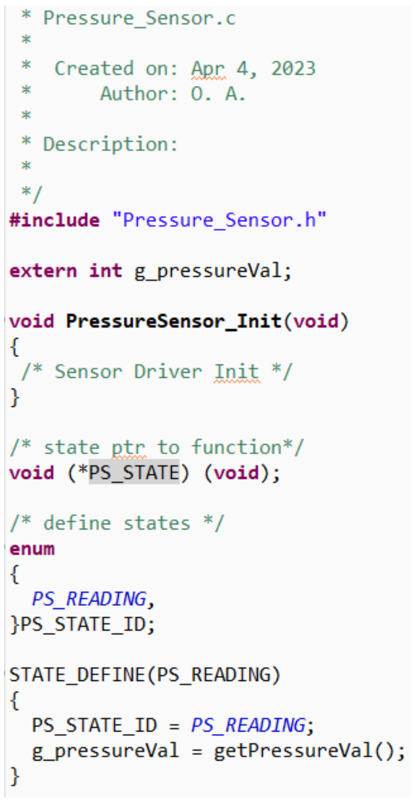
```
#include "App.h"
#include "Alarm.h"
#include "Pressure_Sensor.h"

volatile int g_pressureVal = 0U;
const int pressure_Threshold = 20U; /* 20 bar*/

/* state ptr to function*/
void (*PRESSURE_DETECTION_STATE) (void);

/* define states */
enum
{
  PRESSURE_DETECTION,
}PRESSURE_DETECTION_STATE_ID;

STATE_DEFINE(PRESSURE_DETECTION)
{
  PRESSURE_DETECTION_STATE_ID = PRESSURE_DETECTION;

  if(g_pressureVal > pressure_Threshold)
  {
      StartAlarm();
      Delay(ALARM_TIME_PERIOD);
      StopAlarm();
  }
  else
  {
    /**/
  }

}
```

Pressure Sensor Module



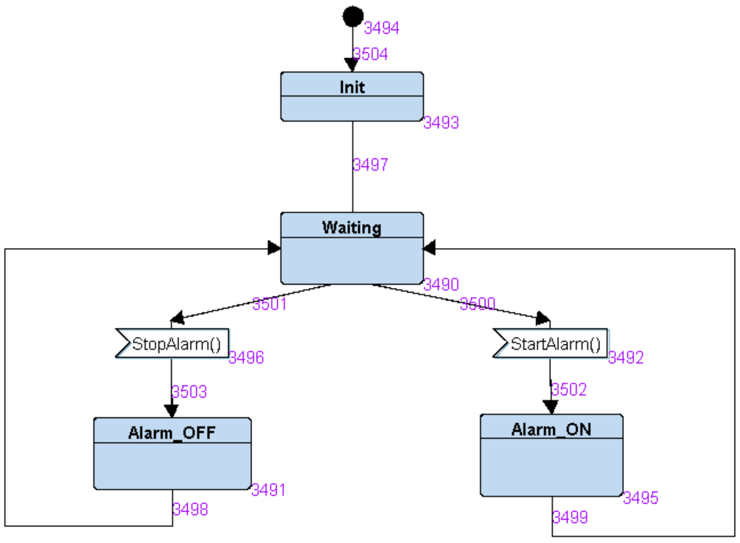


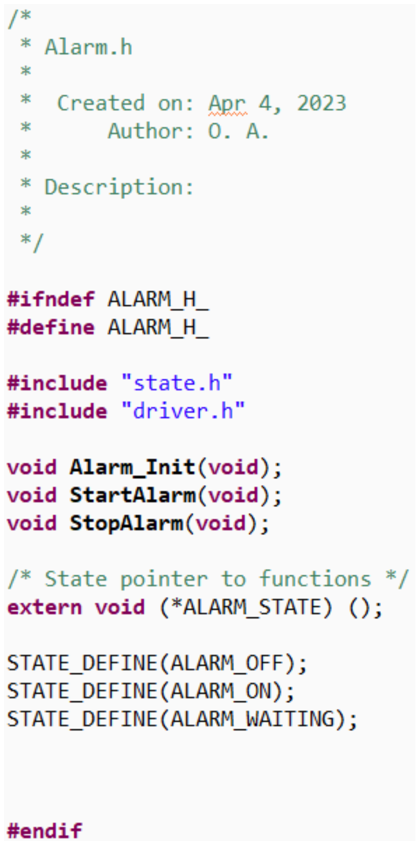
```
 * Pressure_Sensor.h
 *
 *  Created on: Apr 4, 2023
 *      Author: O. A.
 *
 * Description:
 *
 */
#ifndef PRESSURE_SENSOR_H_
#define PRESSURE_SENSOR_H_

#include "state.h"
#include "driver.h"


void PressureSensor_Init(void);


/* State pointer to functions */
extern void (*PS_STATE) ();

STATE_DEFINE(PS_READING);


#endif /* PRESSURE_SENSOR_H_ */
```



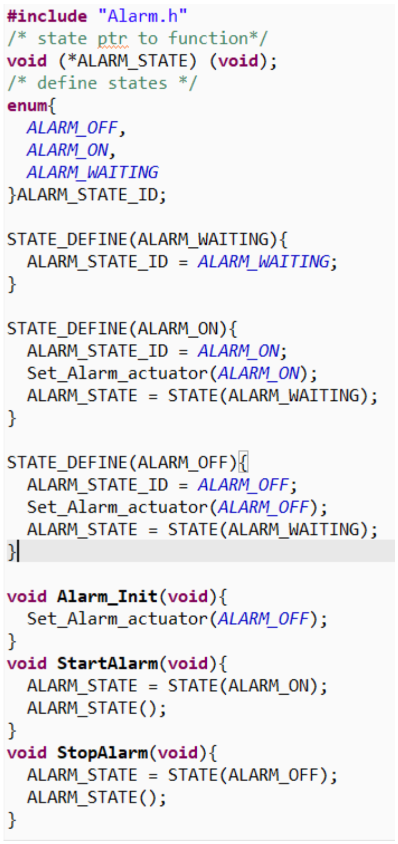
```
 * Pressure_Sensor.c
 *
 *  Created on: Apr 4, 2023
 *      Author: O. A.
 *
 * Description:
 *
 */
#include "Pressure_Sensor.h"

extern int g_pressureVal;

void PressureSensor_Init(void)
{
 /* Sensor Driver Init */
}

/* state ptr to function*/
void (*PS_STATE) (void);

/* define states */
enum
{
  PS_READING,
}PS_STATE_ID;

STATE_DEFINE(PS_READING)
{
  PS_STATE_ID = PS_READING;
  g_pressureVal = getPressureVal();
}
```

Alarm Module





```
/*
 * Alarm.h
 *
 *  Created on: Apr 4, 2023
 *      Author: O. A.
 *
 * Description:
 *
 */

#ifndef ALARM_H_
#define ALARM_H_

#include "state.h"
#include "driver.h"

void Alarm_Init(void);
void StartAlarm(void);
void StopAlarm(void);

/* State pointer to functions */
extern void (*ALARM_STATE) ();

STATE_DEFINE(ALARM_OFF);
STATE_DEFINE(ALARM_ON);
STATE_DEFINE(ALARM_WAITING);



#endif
```



```
#include "Alarm.h"
/* state ptr to function*/
void (*ALARM_STATE) (void);
/* define states */
enum{
   ALARM_OFF,
   ALARM_ON,
   ALARM_WAITING
}ALARM_STATE_ID;

STATE_DEFINE(ALARM_WAITING){
   ALARM_STATE_ID = ALARM_WAITING;
}

STATE_DEFINE(ALARM_ON){
   ALARM_STATE_ID = ALARM_ON;
   Set_Alarm_actuator(ALARM_ON);
   ALARM_STATE = STATE(ALARM_WAITING);
}

STATE_DEFINE(ALARM_OFF){
   ALARM_STATE_ID = ALARM_OFF;
   Set_Alarm_actuator(ALARM_OFF);
   ALARM_STATE = STATE(ALARM_WAITING);
}

void Alarm_Init(void){
   Set_Alarm_actuator(ALARM_OFF);
}
void StartAlarm(void){
   ALARM_STATE = STATE(ALARM_ON);
   ALARM_STATE();
}
void StopAlarm(void){
   ALARM_STATE = STATE(ALARM_OFF);
   ALARM_STATE();
}
```

Compilation

Makefile

```makefile
#@copyright : Omar Anwer

#Project name
PROJECT_NAME = Pressure_Controller

#Architectures Specific Flags
CPU                  = cortex-m3
CFLAGS_ARCH  = -mcpu=$(CPU)
#CFLAGS_ARCH    = -mcpu=$(CPU) -m$(ARCH) --specs=$(SPECS)

#Compiler Flags and Defines
CC                          = arm-none-eabi-
DBGCFLAGS            = -g -gdwarf-2
CFLAGS               = -ansi -std=c89 -O0 -Wall $(CFLAGS_ARCH) $(DBGCFLAGS)

#Linker Flags
LINKER_FILE          = linker_script.ld
LDFLAGS_ARCH = -T $(LINKER_FILE)
STARTUP_FILE = startup.s

#includes
INCS                 = -I .
LIBS                 =

#.c and .s files
SRC                         = $(wildcard *.c)
AS                          = $(wildcard *.s)

PRE                         = $(SRC:.c=.i)
SRCOBJ               = $(SRC:.c=.o)
ASOBJ                = $(AS:.s=.o)


all: $(PROJECT_NAME).bin
	@echo ""
	@$(CC)size.exe $(PROJECT_NAME).elf
	@echo ""
	@echo "Building done..."

%.o: %.s
	$(CC)as.exe $(INCS) $< -o $@
	@echo ""

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

```
%.i: %.c
	$(CC)gcc.exe -E $(INCS) $< -o $@
	@echo ""

$(PROJECT_NAME).elf: $(ASOBJ) $(SRCOBJ) $(PRE)
	$(CC)ld.exe $(LDFLAGS_ARCH) $(LIBS) $(ASOBJ) $(SRCOBJ) -o $@ -
Map=$(PROJECT_NAME).map
	@echo ""
	$(CC)objdump.exe -h $@
	@echo ""
	$(CC)readelf.exe -S $@
	@echo ""
	@cp $(PROJECT_NAME).elf $(PROJECT_NAME).axf

$(PROJECT_NAME).bin: $(PROJECT_NAME).elf
	$(CC)objcopy.exe -O binary $< $@
	@echo ""
	$(CC)objcopy.exe -O ihex $< $(PROJECT_NAME).hex
	@echo ""

clean:
	@rm *.bin *.hex *.elf *.axf *.map

clean-all:
	#@rm  -rf $(filter-out $(STARTUP_FILE), $(AS))
	@rm *.s
	@rm  *.i *.o *.bin *.hex *.elf *.axf *.map
	@echo "All cleaned..."
```

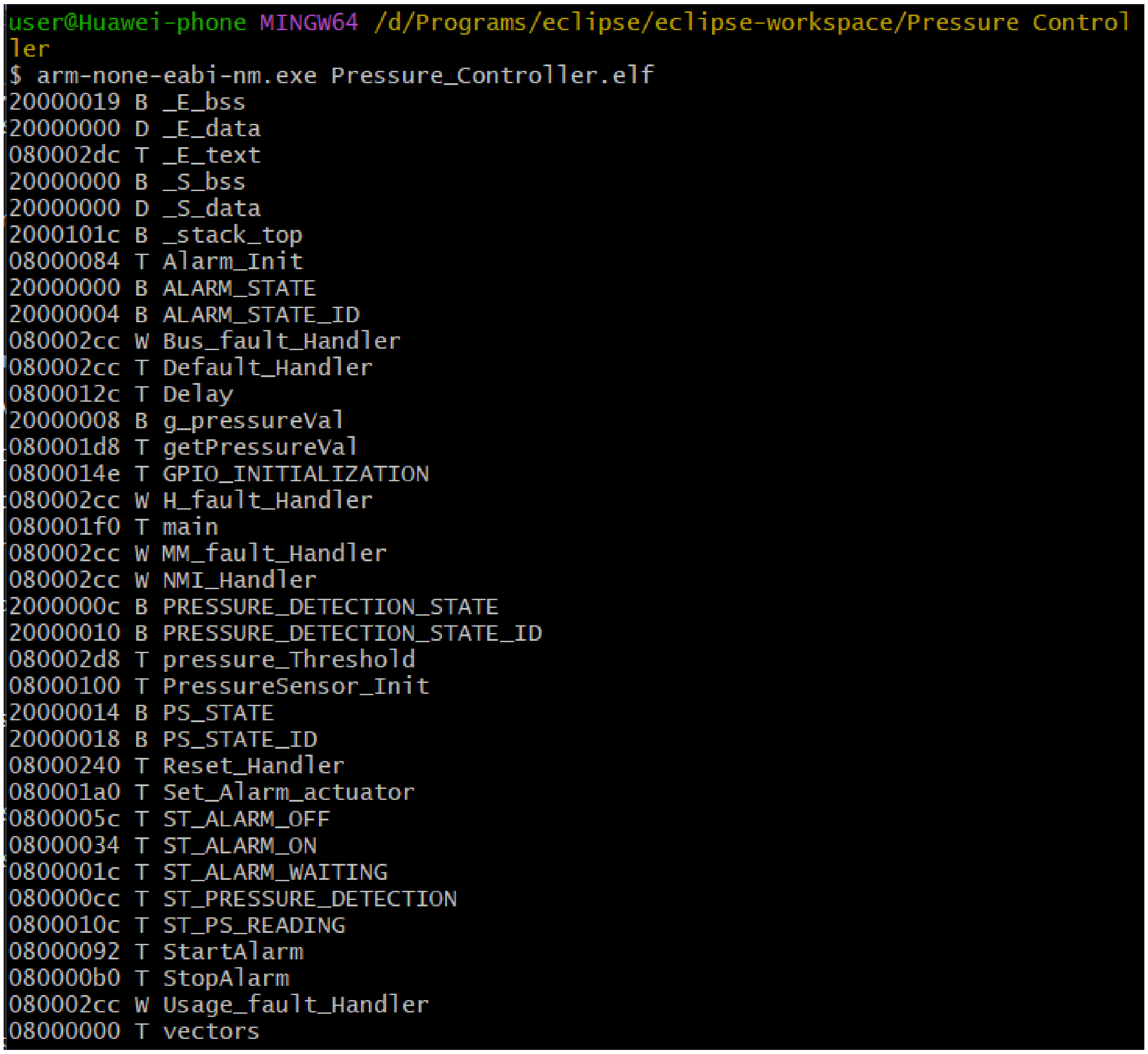
Startup code

```
#include <stdint.h>

/*#define stack_top 0x20001000*/

extern int main(void);

extern uint32_t _stack_top;
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;

void Reset_Handler(void)
{
	/* copy .data section byte by byte from FLASH to SRAM */
	uint8_t* pSrc = (uint8_t*)(&_E_text);
	uint8_t* pDst = (uint8_t*)(&_S_data);
	uint32_t DATA_SIZE = (uint8_t*)(&_E_data) - (uint8_t*)(&_S_data);

	uint32_t i;
	for(i = 0; i < DATA_SIZE; ++i)
	{
		*pDst = *pSrc;
		pSrc++;
		pDst++;
	}

	/* initialize .data section in SRAM */
	pDst = (uint8_t*)(&_S_bss);
	uint32_t BSS_SIZE = (uint8_t*)(&_E_bss) - (uint8_t*)(&_S_bss);

	for(i = 0; i < BSS_SIZE; ++i)
	{
		*pDst = 0;
		pDst++;
	}

	/* jump to main() */
	main();
}

/* initialize vectors */
void Default_Handler(void)
{
	Reset_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_Handler(void)     __attribute__(( weak, alias("Default_Handler") ));
void Usage_fault_Handler(void)   __attribute__(( weak, alias("Default_Handler") ));

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_Handler,
       (uint32_t) &Usage_fault_Handler,
};
```

Linker Script

```
/*
	Linker script Cortex-M3
	By Eng.Omar
*/

ENTRY(Reset_Handler)

MEMORY
{
	FLASH (rx) : ORIGIN = 0x08000000, LENGTH = 64K
	SRAM (rwx) : ORIGIN = 0x20000000, LENGTH = 20K

}

SECTIONS
{
	.text :
	{
		*(.vectors*)
		*(.text*)
		*(.rodata)
		. = ALIGN(4);
		_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 = .;			/*stack top after 4 KB*/
	}>SRAM
}
```

Symbol table



```
user@Huawei-phone MINGW64 /d/Programs/eclipse/eclipse-workspace/Pressure Control
ler
$ arm-none-eabi-nm.exe Pressure_Controller.elf
20000019 B _E_bss
20000000 D _E_data
080002dc T _E_text
20000000 B _S_bss
20000000 D _S_data
2000101c B _stack_top
08000084 T Alarm_Init
20000000 B ALARM_STATE
20000004 B ALARM_STATE_ID
080002cc W Bus_fault_Handler
080002cc T Default_Handler
0800012c T Delay
20000008 B g_pressureVal
080001d8 T getPressureVal
0800014e T GPIO_INITIALIZATION
080002cc W H_fault_Handler
080001f0 T main
080002cc W MM_fault_Handler
080002cc W NMI_Handler
2000000c B PRESSURE_DETECTION_STATE
20000010 B PRESSURE_DETECTION_STATE_ID
080002d8 T pressure_Threshold
08000100 T PressureSensor_Init
20000014 B PS_STATE
20000018 B PS_STATE_ID
08000240 T Reset_Handler
080001a0 T Set_Alarm_actuator
0800005c T ST_ALARM_OFF
08000034 T ST_ALARM_ON
0800001c T ST_ALARM_WAITING
080000cc T ST_PRESSURE_DETECTION
0800010c T ST_PS_READING
08000092 T StartAlarm
080000b0 T StopAlarm
080002cc W Usage_fault_Handler
08000000 T vectors
```

Map file

```
Memory Configuration

Name             Origin             Length             Attributes

FLASH            0x08000000         0x00010000         xr

SRAM             0x20000000         0x00005000         xrw

*default*        0x00000000         0xffffffff


Linker script and memory map




.text           0x08000000      0x2dc

 *(.vectors*)

 .vectors       0x08000000       0x1c startup.o

                0x08000000                vectors

 *(.text*)

 .text          0x0800001c       0xb0 Alarm.o

                0x0800001c                ST_ALARM_WAITING

                0x08000034                ST_ALARM_ON

                0x0800005c                ST_ALARM_OFF

                0x08000084                Alarm_Init

                0x08000092                StartAlarm

                0x080000b0                StopAlarm

 .text          0x080000cc       0x34 App.o

                0x080000cc                ST_PRESSURE_DETECTION

 .text          0x08000100       0x2c Pressure_Sensor.o
```

```
               0x08000100                PressureSensor_Init
               0x0800010c                ST_PS_READING
 .text         0x0800012c       0xc4 driver.o
               0x0800012c                Delay
               0x0800014e                GPIO_INITIALIZATION
               0x080001a0                Set_Alarm_actuator
               0x080001d8                getPressureVal
 .text         0x080001f0       0x50 main.o
               0x080001f0                main
 .text         0x08000240       0x98 startup.o
               0x08000240                Reset_Handler
               0x080002cc                MM_fault_Handler
               0x080002cc                Usage_fault_Handler
               0x080002cc                Bus_fault_Handler
               0x080002cc                Default_Handler
               0x080002cc                H_fault_Handler
               0x080002cc                NMI_Handler
 *(.rodata)
 .rodata       0x080002d8        0x4 App.o
               0x080002d8                pressure_Threshold
               0x080002dc                . = ALIGN (0x4)
               0x080002dc                _E_text = .

.glue_7        0x080002dc        0x0
 .glue_7       0x080002dc        0x0 linker stubs
```

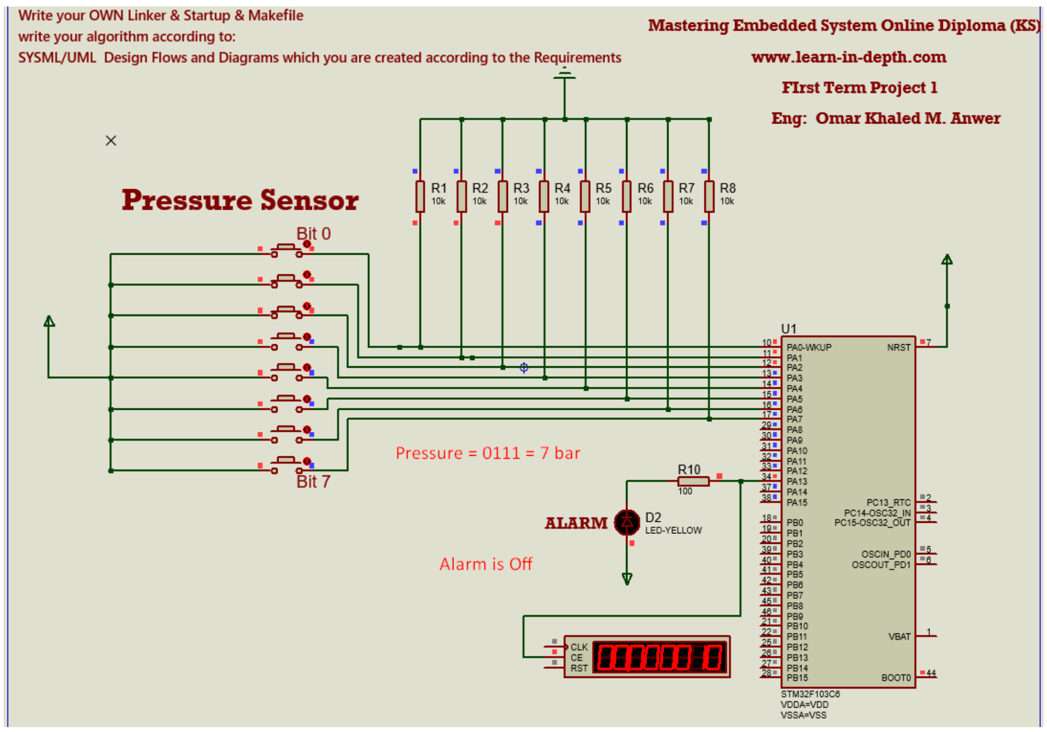
```
.glue_7t        0x080002dc        0x0

 .glue_7t       0x080002dc        0x0 linker stubs

.vfp11_veneer   0x080002dc        0x0

 .vfp11_veneer  0x080002dc        0x0 linker stubs

.v4_bx          0x080002dc        0x0

 .v4_bx         0x080002dc        0x0 linker stubs

.iplt           0x080002dc        0x0

 .iplt          0x080002dc        0x0 Alarm.o

.rel.dyn        0x080002dc        0x0

 .rel.iplt      0x080002dc        0x0 Alarm.o

.data           0x20000000        0x0 load address 0x080002dc

                0x20000000                _S_data = .

 *(.data)

 .data          0x20000000        0x0 Alarm.o

 .data          0x20000000        0x0 App.o

 .data          0x20000000        0x0 Pressure_Sensor.o

 .data          0x20000000        0x0 driver.o

 .data          0x20000000        0x0 main.o

 .data          0x20000000        0x0 startup.o

                0x20000000                . = ALIGN (0x4)

                0x20000000                _E_data = .

.igot.plt       0x20000000        0x0 load address 0x080002dc

 .igot.plt      0x20000000        0x0 Alarm.o


.bss            0x20000000     0x101c load address 0x080002dc
```

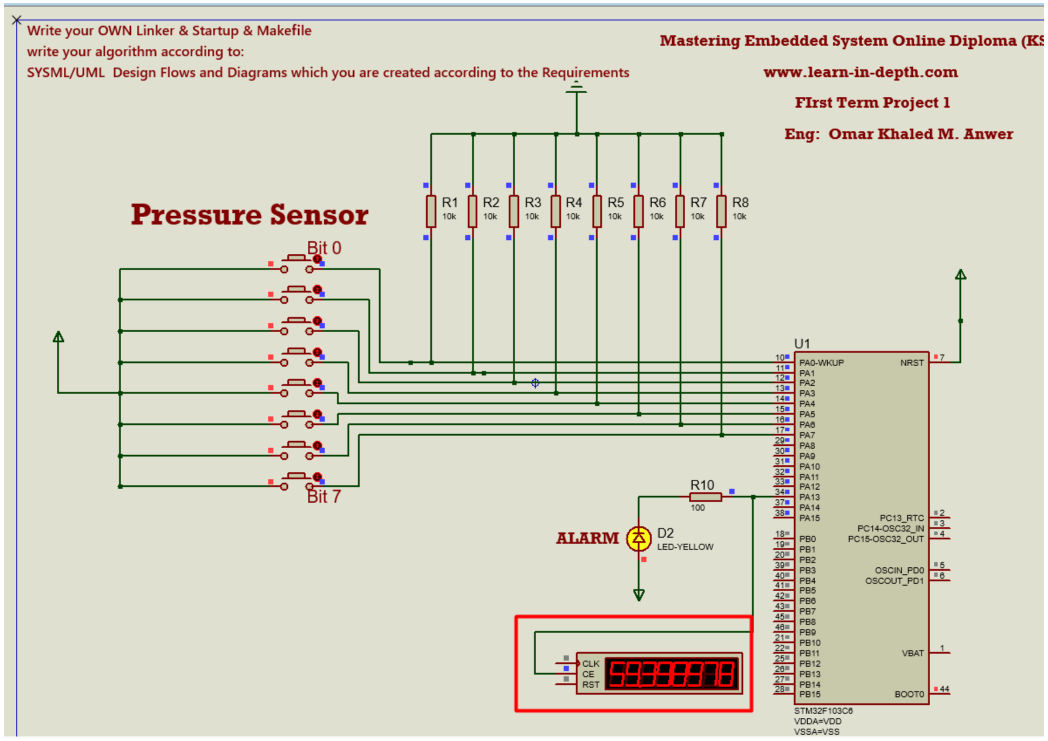
```
                0x20000000                _S_bss = .
 *(.bss*)
 .bss           0x20000000        0x5 Alarm.o
                0x20000000                ALARM_STATE
                0x20000004                ALARM_STATE_ID
 *fill*         0x20000005        0x3
 .bss           0x20000008        0x9 App.o
                0x20000008                g_pressureVal
                0x2000000c                PRESSURE_DETECTION_STATE
                0x20000010                PRESSURE_DETECTION_STATE_ID
 *fill*         0x20000011        0x3
 .bss           0x20000014        0x5 Pressure_Sensor.o
                0x20000014                PS_STATE
                0x20000018                PS_STATE_ID
 .bss           0x20000019        0x0 driver.o
 .bss           0x20000019        0x0 main.o
 .bss           0x20000019        0x0 startup.o
                0x20000019                _E_bss = .
                0x2000001c                . = ALIGN (0x4)
 *fill*         0x20000019        0x3
                0x2000101c                . = (. + 0x1000)
 *fill*         0x2000001c     0x1000
                0x2000101c                _stack_top = .
LOAD Alarm.o
LOAD App.o
LOAD Pressure_Sensor.o

LOAD driver.o

LOAD main.o

LOAD startup.o

OUTPUT(Pressure_Controller.elf elf32-littlearm)

LOAD linker stubs


.debug_info     0x00000000      0x6ea

 .debug_info    0x00000000      0x163 Alarm.o

 .debug_info    0x00000163      0x116 App.o

 .debug_info    0x00000279      0x10b Pressure_Sensor.o

 .debug_info    0x00000384      0x10c driver.o

 .debug_info    0x00000490       0xc5 main.o

 .debug_info    0x00000555      0x195 startup.o


.debug_abbrev   0x00000000      0x472

 .debug_abbrev  0x00000000       0xe3 Alarm.o

 .debug_abbrev  0x000000e3       0xbf App.o

 .debug_abbrev  0x000001a2       0xcb Pressure_Sensor.o

 .debug_abbrev  0x0000026d       0xc5 driver.o

 .debug_abbrev  0x00000332       0x6e main.o

 .debug_abbrev  0x000003a0       0xd2 startup.o


.debug_loc      0x00000000      0x3a4

 .debug_loc     0x00000000      0x120 Alarm.o
 .debug_loc     0x00000120       0x2c App.o
 .debug_loc     0x0000014c       0x70 Pressure_Sensor.o
 .debug_loc     0x000001bc      0x140 driver.o
 .debug_loc     0x000002fc       0x2c main.o
 .debug_loc     0x00000328       0x7c startup.o

.debug_aranges  0x00000000       0xc0
 .debug_aranges
                0x00000000       0x20 Alarm.o
 .debug_aranges
                0x00000020       0x20 App.o
 .debug_aranges
                0x00000040       0x20 Pressure_Sensor.o
 .debug_aranges
                0x00000060       0x20 driver.o
 .debug_aranges
                0x00000080       0x20 main.o
 .debug_aranges
                0x000000a0       0x20 startup.o

.debug_line     0x00000000      0x431
 .debug_line    0x00000000       0x7f Alarm.o
 .debug_line    0x0000007f       0x51 App.o
 .debug_line    0x000000d0       0x68 Pressure_Sensor.o
 .debug_line    0x00000138      0x120 driver.o
 .debug_line    0x00000258       0x85 main.o
 .debug_line    0x000002dd      0x154 startup.o

.debug_str      0x00000000      0x317
 .debug_str     0x00000000      0x171 Alarm.o
                                0x1cb (size before relaxing)
 .debug_str     0x00000171       0x72 App.o
                                0x1c2 (size before relaxing)
 .debug_str     0x000001e3       0x49 Pressure_Sensor.o
                                0x19f (size before relaxing)
 .debug_str     0x0000022c       0x57 driver.o
                                0x19d (size before relaxing)
 .debug_str     0x00000283        0xc main.o
                                0x177 (size before relaxing)
 .debug_str     0x0000028f       0x88 startup.o
                                0x1dc (size before relaxing)

.comment        0x00000000       0x49
 .comment       0x00000000       0x49 Alarm.o
                                 0x4a (size before relaxing)
 .comment       0x00000049       0x4a App.o
 .comment       0x00000049       0x4a Pressure_Sensor.o
 .comment       0x00000049       0x4a driver.o
 .comment       0x00000049       0x4a main.o
 .comment       0x00000049       0x4a startup.o
```

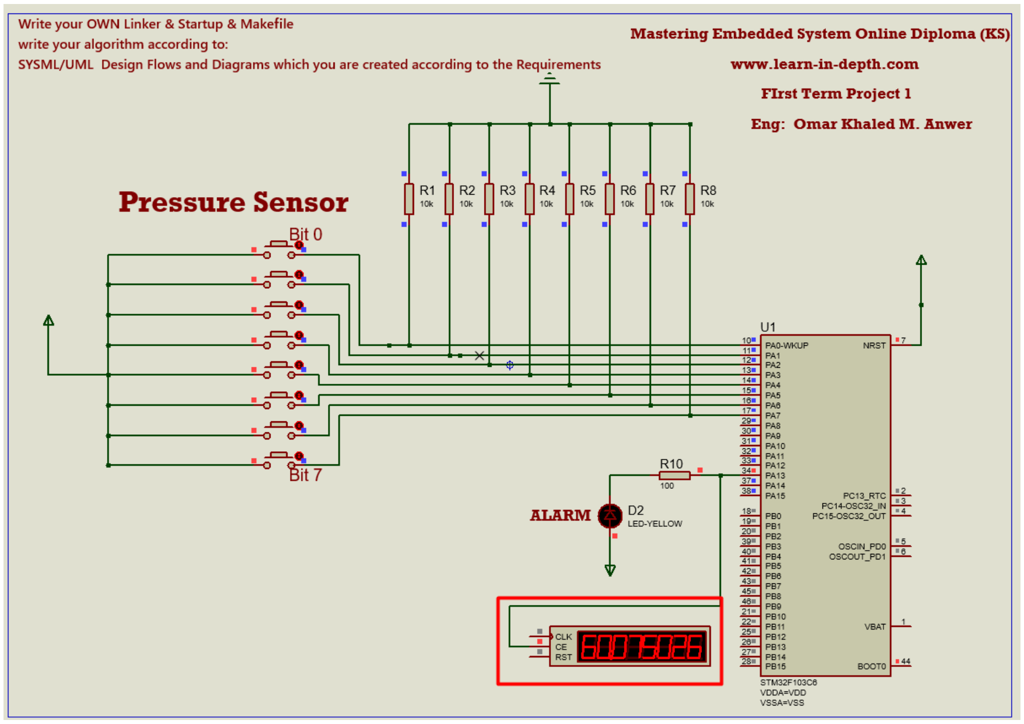
```
.ARM.attributes
                0x00000000       0x2d
 .ARM.attributes
                0x00000000       0x2d Alarm.o
 .ARM.attributes
                0x0000002d       0x2d App.o
 .ARM.attributes
                0x0000005a       0x2d Pressure_Sensor.o
 .ARM.attributes
                0x00000087       0x2d driver.o
 .ARM.attributes
                0x000000b4       0x2d main.o
 .ARM.attributes
                0x000000e1       0x2d startup.o

.debug_frame    0x00000000      0x250
 .debug_frame   0x00000000       0xbc Alarm.o
 .debug_frame   0x000000bc       0x2c App.o
 .debug_frame   0x000000e8       0x4c Pressure_Sensor.o
 .debug_frame   0x00000134       0xa0 driver.o
 .debug_frame   0x000001d4       0x2c main.o
 .debug_frame   0x00000200       0x50 startup.o
```

Simulation

Simulaton video


Write your OWN Linker & Startup & Makefile
write your algorithm according to:
SYSML/UML Design Flows and Diagrams which you are created according to the Requirements
Mastering Embedded System Online Diploma (KS)
www.learn-in-depth.com
FIrst Term Project 1
Eng: Omar Khaled M. Anwer
Pressure Sensor
Bit 0
Bit 7
R1 R2 R3 R4 R5 R6 R7 R8
10k
Pressure = 0111 = 7 bar
R10
100
ALARM
D2
LED-YELLOW
Alarm is Off
CLK
CE
RST
U1
NRST
STM32F103C6
VDDA=VDD
VSSA=VSS


Write your OWN Linker & Startup & Makefile
write your algorithm according to:
SYSML/UML Design Flows and Diagrams which you are created according to the Requirements
Mastering Embedded System Online Diploma (KS
www.learn-in-depth.com
FIrst Term Project 1
Eng: Omar Khaled M. Anwer
Pressure Sensor
Bit 0
Bit 7
R1 R2 R3 R4 R5 R6 R7 R8
10k
R10
100
ALARM
D2
LED-YELLOW
CLK
CE
RST
U1
NRST
STM32F103C6
VDDA=VDD
VSSA=VSS


Write your OWN Linker & Startup & Makefile
write your algorithm according to:
SYSML/UML Design Flows and Diagrams which you are created according to the Requirements

Mastering Embedded System Online Diploma (KS)
www.learn-in-depth.com
FIrst Term Project 1
Eng: Omar Khaled M. Anwer