Mastering Embedded System Online Diploma

www.learn-in-depth.com

First Term (Final Project 1)

ENG. HASSAN SAMY FAHMY

My Profile:
www.learn-in-depth.com/online-diploma/hsamy12@gmail.com

Contents

Introduction

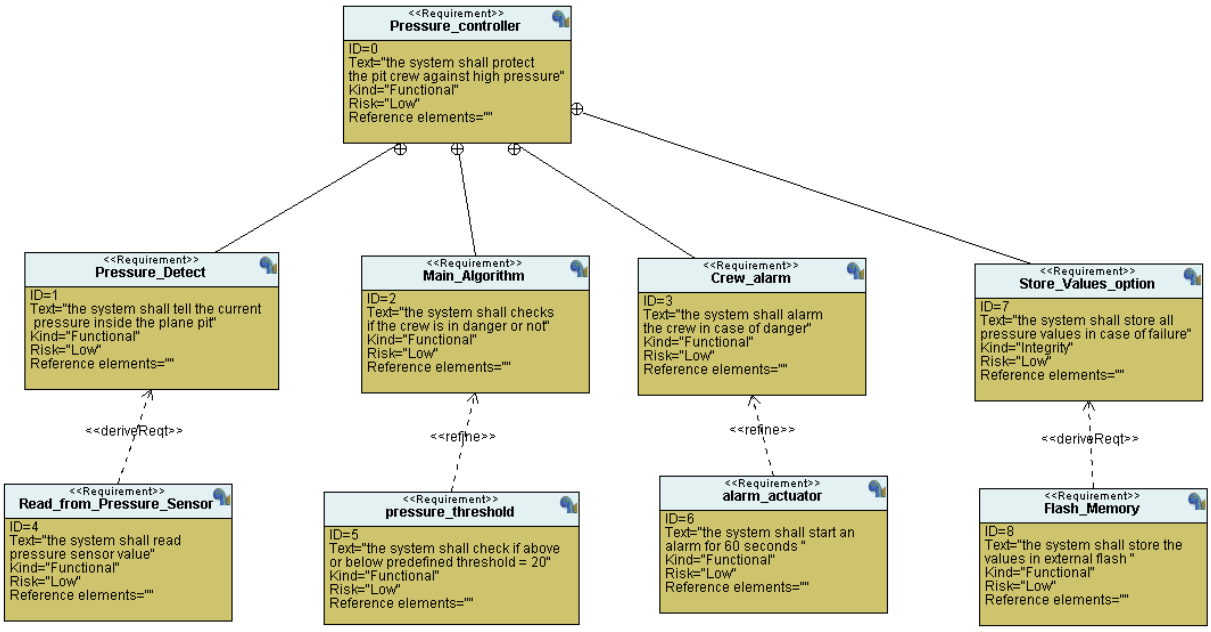
For finishing the first term the first project to be implemented is high pressure sensing algorithm to protect a cabin crew from pressure exceeding 20 bars, upon exceeding that pressure an alarm will be activated for 60 seconds to inform the crew of the situation.

I took the design sequence steps to reach a solution and implement my method for implementing the project which was implemented using waterfall method of designing and starts with making a requirements diagram of it then system analysis before implementing the design by state machine modules.

Microcontroller used in the project is STM32F103C6 from ARM with cortex-m3 with frequency 72 MHz

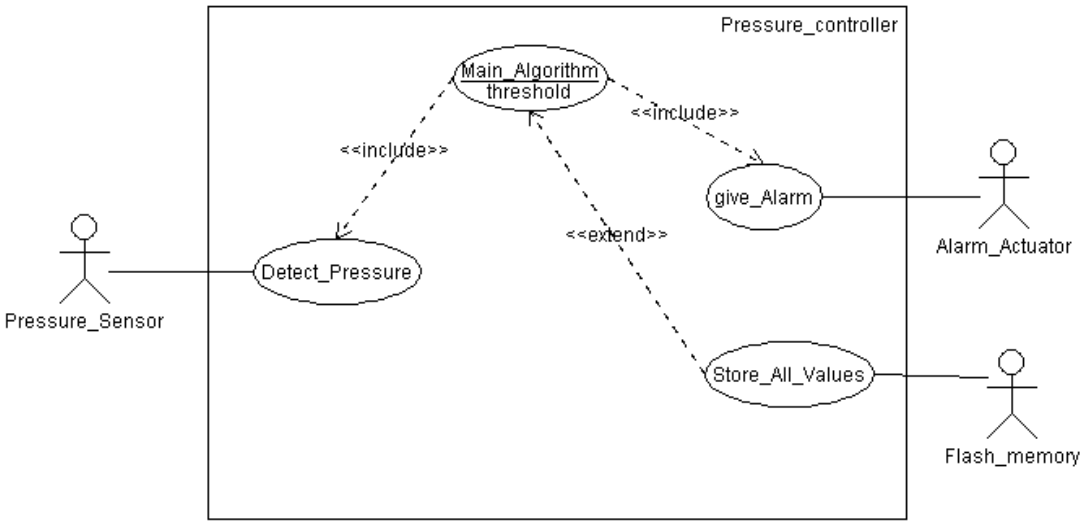
Although the requirements and analysis included an optional feature of storing values of pressure in a flash memory it wasn't implemented in the design.

Requirements

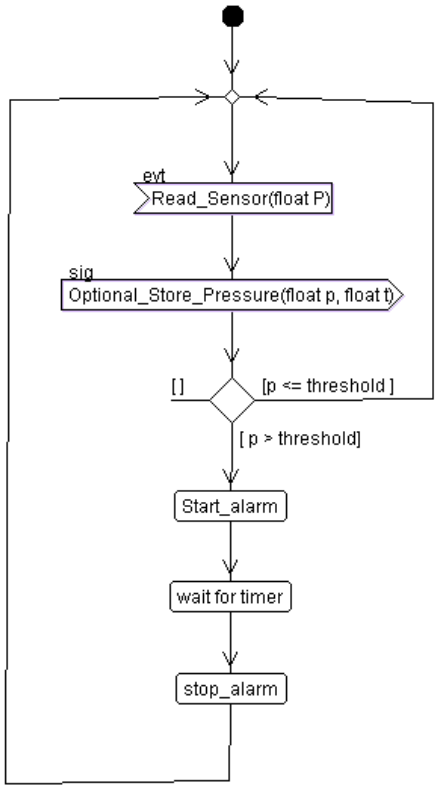


System analysis

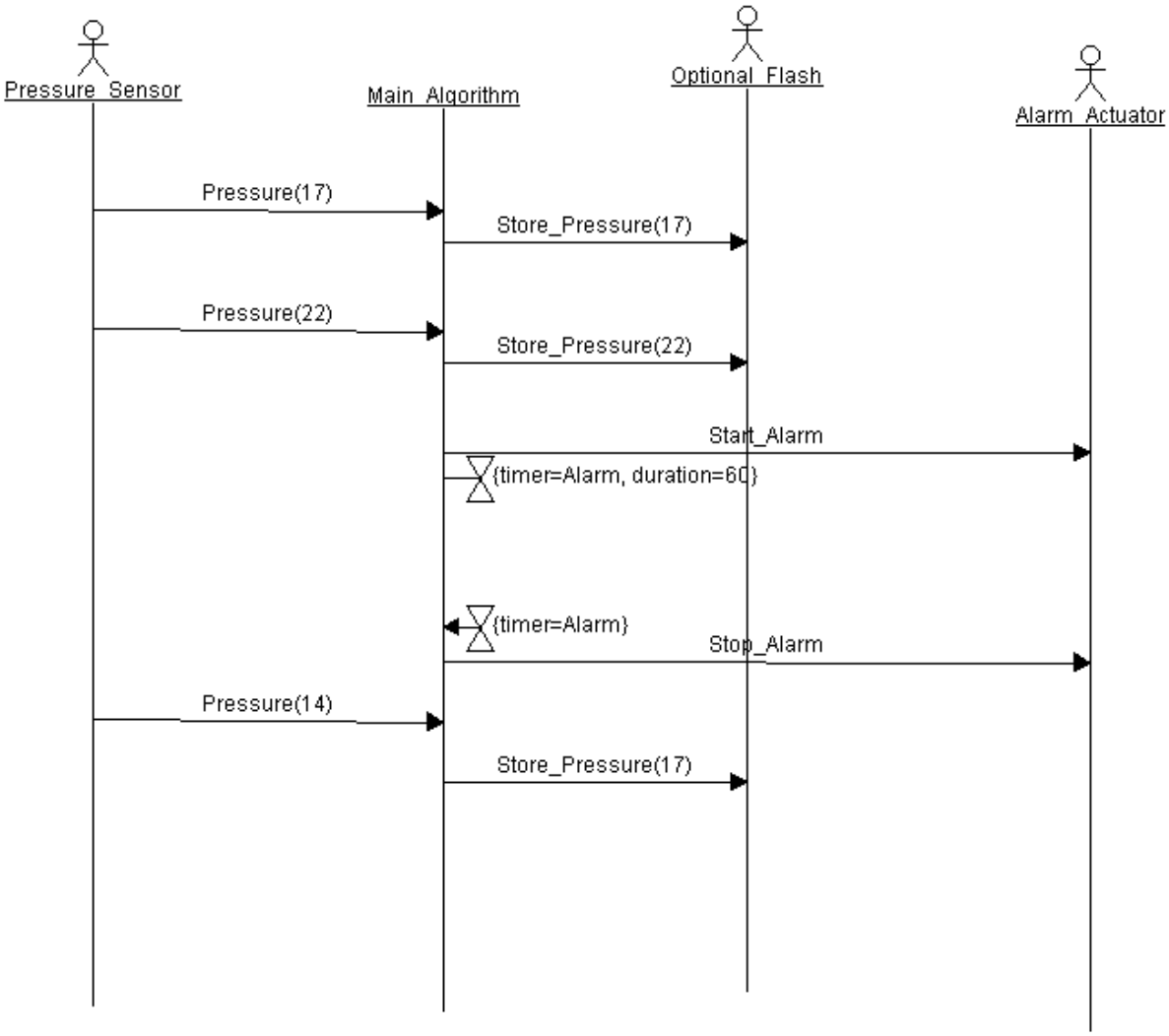
1. Case diagram



2. Activity diagram

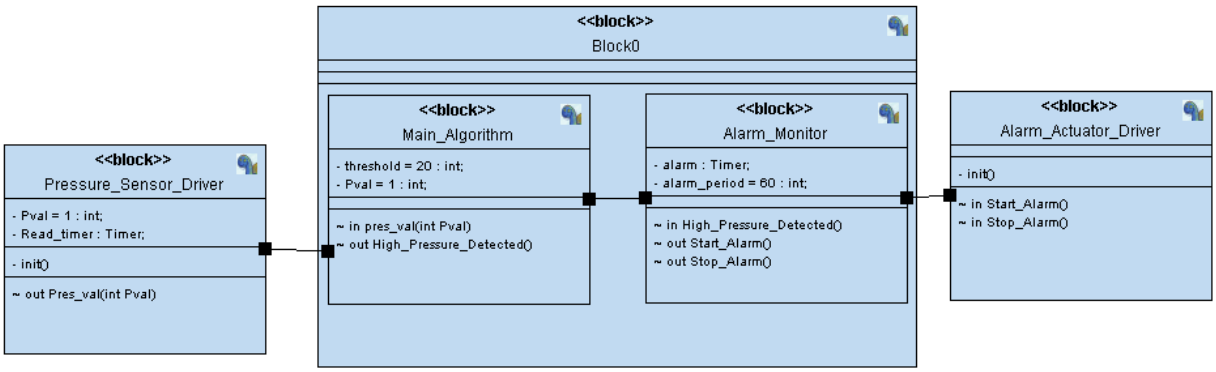


3. Scenario diagram



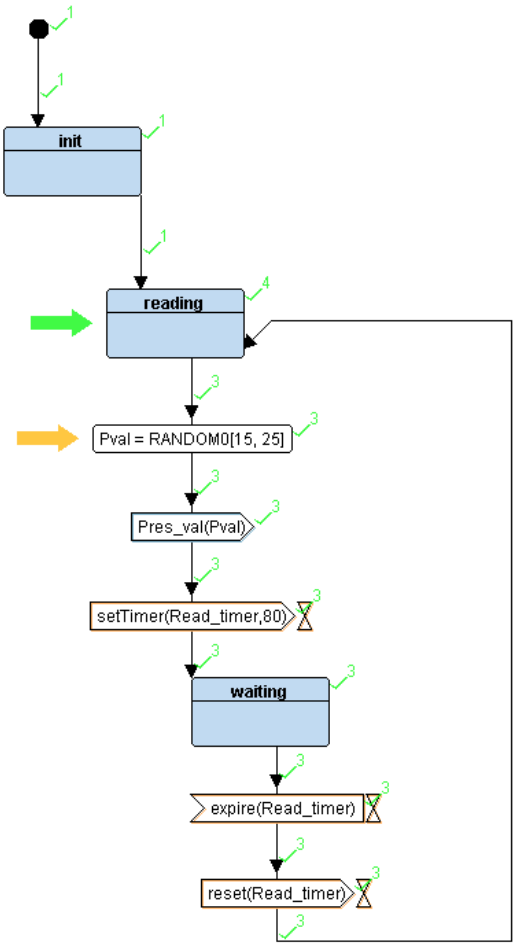
System design

1. Block diagram

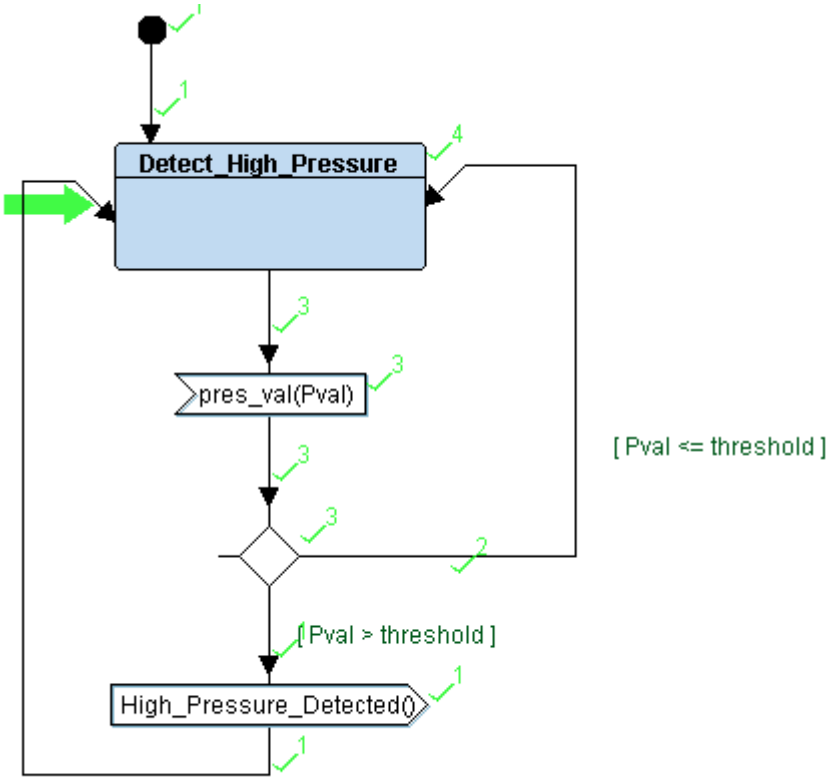


2. State diagrams

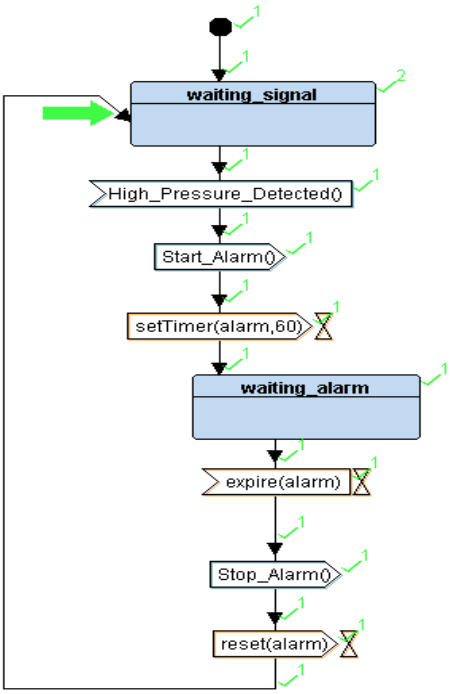
- Pressure sensor Driver



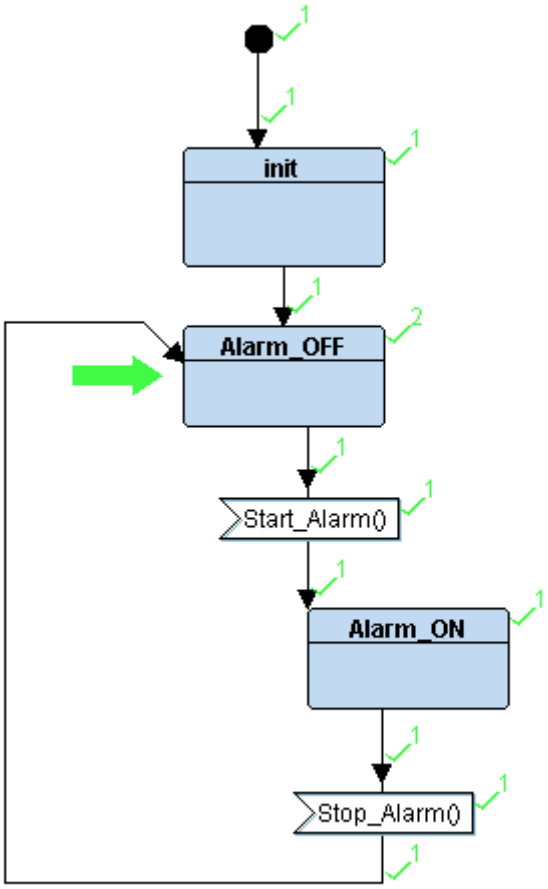
- Main Algorithm



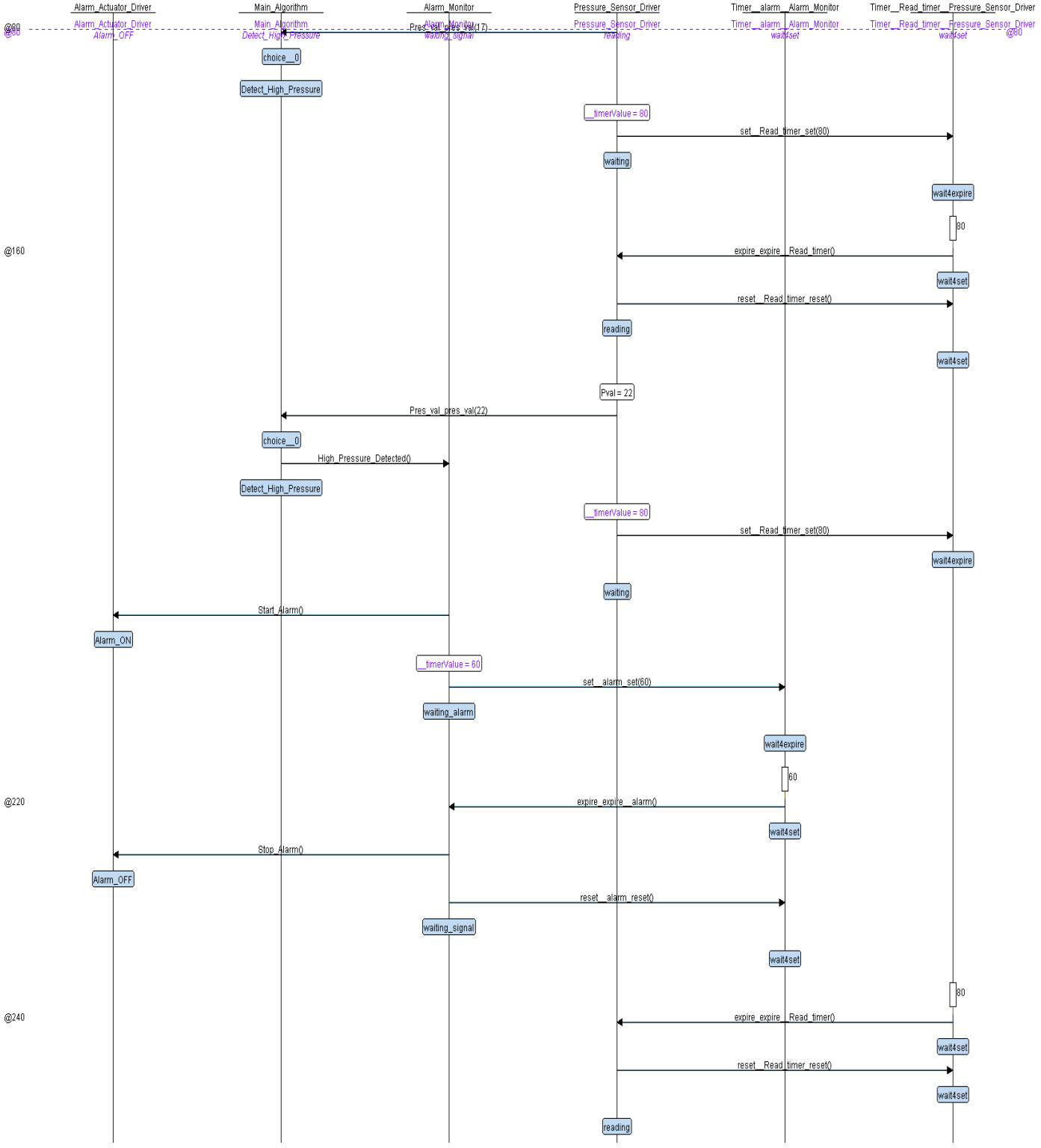
- Alarm monitor



- Alarm actuator driver



3. Simulation



Code

1. Pressure sensor state machine modules.

```c
/*
 * PS_Driver.h
 *
 *  Created on: Feb 28, 2022
 *      Author: Hassan
 */

#ifndef PS_DRIVE_H_
#define PS_DRIVE_H_

#include "state.h"

enum{
    PS_reading,
    PS_waiting
}PS_state_id;

/* prototypes */
STATE_define_(PS_reading);
STATE_define_(PS_waiting);

/* ptr to function for states in PS */
extern void (*PS_state)();

#endif /* PS_DRIVE_H_ */

```

```c
/*
 * PS_Driver.c
 *
 *  Created on: Feb 28, 2022
 *      Author: Hassan
 */

#include "PS_Driver.h"

void (*PS_state)();
unsigned int PS_Pval;

/*init function is inside GPIO initialization */


STATE_define_(PS_reading){

    PS_state_id = PS_reading;   //state name

    PS_Pval = getPressureVal(); // state action
    Pres_val(PS_Pval);
    PS_state = STATE(PS_waiting);
}

STATE_define_(PS_waiting){

    PS_state_id = PS_waiting;   //state name

    Delay( 80000 ); // state action

    PS_state = STATE(PS_reading);
}
```

2. Alarm state machine modules

```c
/*
 * Alarm.h
 *
 *  Created on: Feb 28, 2022
 *      Author: Hassan
 */

#ifndef ALARM_H_
#define ALARM_H_

#include "state.h"

enum{
    High_Pressure,
    Norm_Pressure
}Alarm_state_id;

/* prototypes */
STATE_define_(High_Pressure);
STATE_define_(Norm_Pressure);

/* ptr to function for states in Alarm */
extern void (*Alarm_state)();

#endif /* ALARM_H_ */
```

```c
/*
 * Alarm.c
 *
 *  Created on: Feb 28, 2022
 *      Author: Hassan
 */

#include "Alarm.h"

#define THRESHOLD 20

void (*Alarm_state)();
unsigned int Al_Pval = 1;
unsigned int Al_Signal = 0;

void Pres_val(int p){ //checks if pressure is high or normal
    Al_Pval = p;
    Alarm_state= (Al_Pval > THRESHOLD)? (STATE(High_Pressure)):(STATE(Norm_Pressure));
}

STATE_define_(High_Pressure){
    Alarm_state_id = High_Pressure; //state name

    Al_Signal =0;   // state action
    Set_Alarm_actuator(Al_Signal);
    Delay(60000);
    Alarm_state=STATE(Norm_Pressure);
}

STATE_define_(Norm_Pressure){
    Alarm_state_id = Norm_Pressure; //state name

    Al_Signal =1;   // state action
    Set_Alarm_actuator(Al_Signal);
    Delay(60000);
    Alarm_state=STATE(Norm_Pressure);
}

```

3. Driver

```
#ifndef DRIVER_H_
#define DRIVER_H_

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

#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 ();

#endif /* DRIVER_H_ */

```

```c

#include "driver.h"

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

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

void Set_Alarm_actuator(int i){
    if (i == 1){
        SET_BIT(GPIOA_ODR,13);  // lamb closes due to pull up resistor
    }
    else if (i == 0){
        RESET_BIT(GPIOA_ODR,13);    // lamb opens due to pull up resistor
    }
}

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

4. Main file

```c
/*
 * main.c
 *
 *  Created on: Feb 28, 2022
 *      Author: Hassan
 */

#include "PS_Driver.h"
#include "Alarm.h"


void setup(){
    GPIO_INITIALIZATION();

    Alarm_state = STATE(Norm_Pressure);
    PS_state = STATE(PS_reading);
}

int main (){

    setup();

    while (1){
        PS_state();
        Alarm_state();
    }
}
```

5. Make file

```
#@copyright: Hassan Samy
CC=arm-none-eabi-
CFLAGS=-mcpu=cortex-m3 -mthumb -gdwarf-2
INCS= -I ./driver
LIBS=
SRC=$(wildcard *.c)
OBJ=$(SRC:.c=.o)
As=$(wildcard *.s)
AsOBJ=$(As:.s=.o)
Project_name=Final_Project1

all:$(Project_name).bin
    @echo "*"
    @echo "*******Build is done.*******"
    @echo "*"


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

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

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

clean_all:
    rm *.o *.map *.elf *.bin
    @echo "*******all clean.*******"

clean:
    rm *.elf *.bin
```

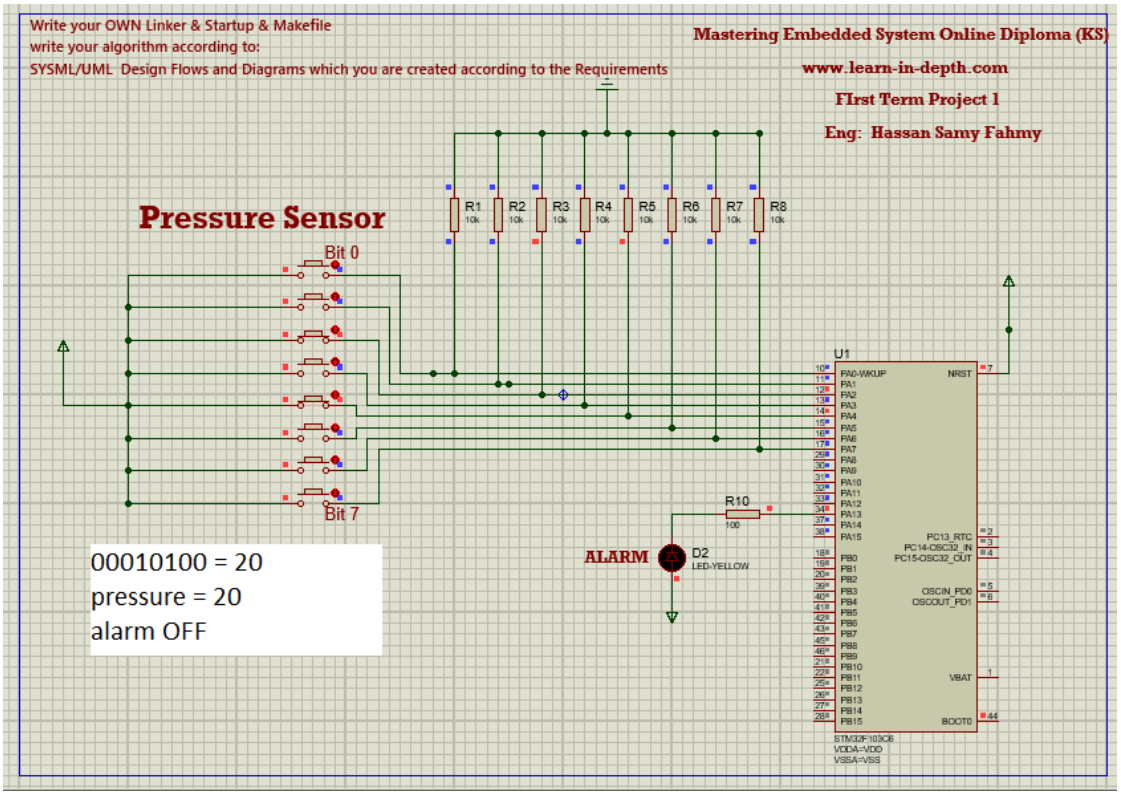
6. Startup file

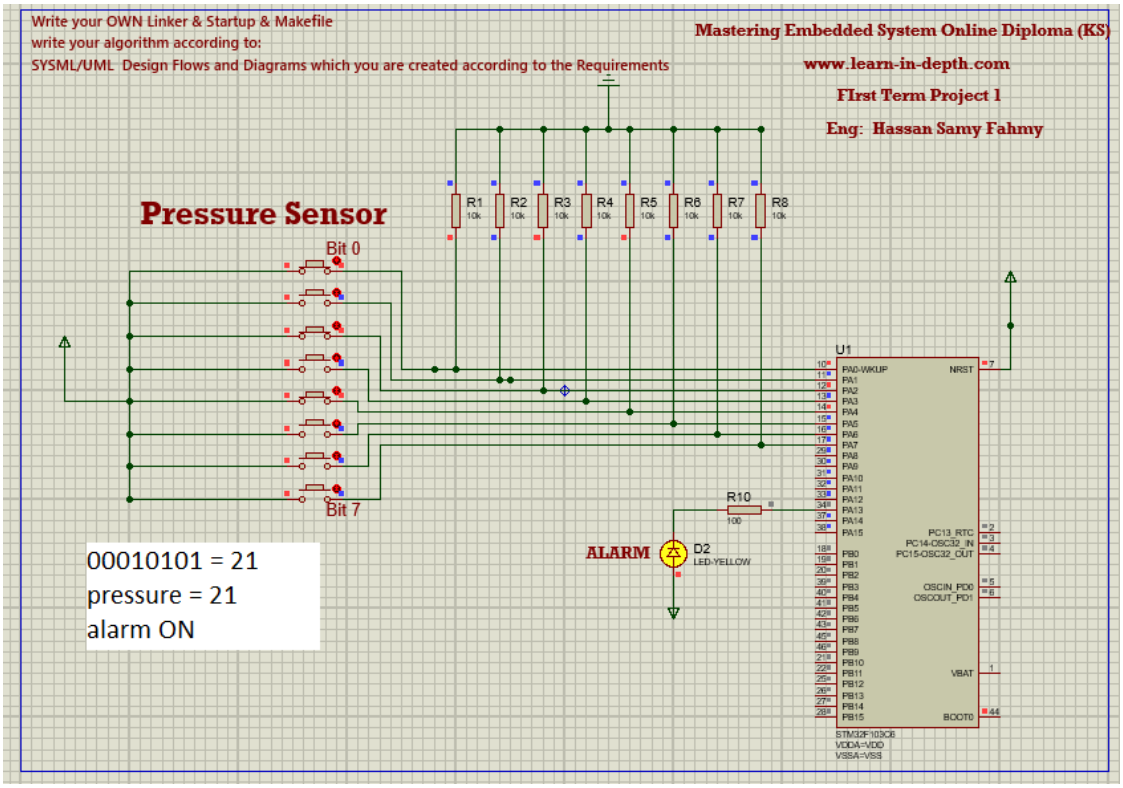
```c
/*****************************************
*   file        : startup.c
*   author      : Hassan Samy
*   description : performs the start
*       mechanism for the board until
*       reaching main function.
*****************************************/

#include "Plat_type.h"

#define SP_top 0x020001000

extern uint32 _S_DATA;
extern uint32 _E_DATA;
extern uint32 _S_bss;
extern uint32 _E_bss;
extern uint32 _E_text;

extern int main(void);
void Reset_Handler(void);

void Default_Handler()
{
    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 vectors[] __attribute__((section(".vectors"))) ={
    SP_top,
    (uint32)&Reset_Handler,
    (uint32)&NMI_Handler,
    (uint32)&H_fault_Handler,
    (uint32)&MM_fault_Handler,
    (uint32)&Bus_fault_Handler,
    (uint32)&Usage_fault_Handler
};

void Reset_Handler(void){   /* copy the .data from ROM to RAM*/
    uint32 DATA_Size =(uint8*)&_E_DATA - (uint8*)&_S_DATA ;
    uint8 *P_src =(uint8*)&_E_text;
    uint8 *P_dst =(uint8*)&_S_DATA;

    uint32 i;
    for(i=0; i<DATA_Size;i++){
        *((uint8*)P_dst++) = *((uint8*)P_src++);
    }
    /* init the .bss with 0 in RAM */
    uint32 bss_size = (uint8*)&_E_bss - (uint8*)&_S_bss ;
    P_dst =(uint8*)&_E_bss;
    for(i=0; i<bss_size;i++){
        *((uint8*)P_dst++) = (uint8)0;
    }

    main();
}
```

7. Linker script

```
/*****************************************
*   file        : linker_script
*   author      : Hassan Samy
*   description : performs the linking
*       between all .o files and libraries
*       as well as making the .elf file.
*****************************************/
MEMORY
{
    flash(RX): ORIGIN = 0x08000000 , LENGTH = 128k
    sram(RWX): ORIGIN = 0x20000000 , LENGTH = 20k
}
SECTIONS
{
    .text : {
        *(.vectors*)
        *(.text*)
        *(.rodata)
        _E_text = .;

    }>flash

    .data :{
        _S_DATA = .;
        *(.data)
        _E_DATA = .;
    }>sram AT>flash

    .bss :{
        _S_bss = .;
        *(.bss*)
        . = ALIGN(4);
        _E_bss = .;
        . = ALIGN(4);
        . = . + 0x1000;
        SP_top = .;
    }>sram
}
```

Simulation by proteus





Software analysis

1. Map file

```
Allocating common symbols
Common symbol       size              file

PS_Pval             0x4               PS_Driver.o
PS_state            0x4               PS_Driver.o
PS_state_id         0x1               main.o
Alarm_state         0x4               Alarm.o
Alarm_state_id      0x1               Alarm.o

Memory Configuration

Name             Origin             Length             Attributes
flash            0x08000000         0x00020000         xr
sram             0x20000000         0x00005000         xrw
*default*        0x00000000         0xffffffff

Linker script and memory map


.text           0x08000000      0x394
 *(.vectors*)
 .vectors       0x08000000       0x1c startup.o
                0x08000000                vectors
 *(.text*)
 .text          0x0800001c       0xe0 Alarm.o
                0x0800001c                Pres_val
                0x08000064                ST_High_Pressure
                0x080000b0                ST_Norm_Pressure
 .text          0x080000fc      0x10c driver.o
                0x080000fc                Delay
                0x08000120                getPressureVal
                0x08000138                Set_Alarm_actuator
                0x08000188                GPIO_INITIALIZATION
 .text          0x08000208       0x54 main.o
                0x08000208                setup
                0x08000238                main
 .text          0x0800025c       0x7c PS_Driver.o
                0x0800025c                ST_PS_reading
                0x080002a4                ST_PS_waiting
 .text          0x080002d8       0xbc startup.o
                0x080002d8                Bus_fault_Handler
                0x080002d8                MM_fault_Handler
                0x080002d8                Usage_fault_Handler
                0x080002d8                H_fault_Handler
                0x080002d8                Default_Handler
                0x080002d8                NMI_Handler
                0x080002e4                Reset_Handler
```

```
 *(.rodata)
                0x08000394                _E_text = .

.glue_7         0x08000394        0x0
 .glue_7        0x00000000        0x0 linker stubs

.glue_7t        0x08000394        0x0
 .glue_7t       0x00000000        0x0 linker stubs

.vfp11_veneer   0x08000394        0x0
 .vfp11_veneer  0x00000000        0x0 linker stubs

.v4_bx          0x08000394        0x0
 .v4_bx         0x00000000        0x0 linker stubs

.iplt           0x08000394        0x0
 .iplt          0x00000000        0x0 Alarm.o

.rel.dyn        0x08000394        0x0
 .rel.iplt      0x00000000        0x0 Alarm.o

.data           0x20000000        0x4 load address 0x08000394
                0x20000000                _S_DATA = .
 *(.data)
 .data          0x20000000        0x4 Alarm.o
                0x20000000                Al_Pval
 .data          0x20000004        0x0 driver.o
 .data          0x20000004        0x0 main.o
 .data          0x20000004        0x0 PS_Driver.o
 .data          0x20000004        0x0 startup.o
                0x20000004                _E_DATA = .

.igot.plt       0x20000004        0x0 load address 0x08000398
 .igot.plt      0x00000000        0x0 Alarm.o

.bss            0x20000004     0x1014 load address 0x08000398
                0x20000004                _S_bss = .
 *(.bss*)
 .bss           0x20000004        0x4 Alarm.o
                0x20000004                Al_Signal
 .bss           0x20000008        0x0 driver.o
 .bss           0x20000008        0x0 main.o
 .bss           0x20000008        0x0 PS_Driver.o
 .bss           0x20000008        0x0 startup.o
                0x20000008                . = ALIGN (0x4)
                0x20000008                _E_bss = .
                0x20000008                . = ALIGN (0x4)
                0x20001008                . = (. + 0x1000)
 *fill*         0x20000008     0x1000
```

```
 *fill*         0x20000008     0x1000
                0x20001008                SP_top = .
 COMMON         0x20001008        0x5 Alarm.o
                0x20001008                Alarm_state
                0x2000100c                Alarm_state_id
 COMMON         0x2000100d        0x1 main.o
                0x2000100d                PS_state_id
 *fill*         0x2000100e        0x2
 COMMON         0x20001010        0x8 PS_Driver.o
                0x20001010                PS_Pval
                0x20001014                PS_state
LOAD Alarm.o
LOAD driver.o
LOAD main.o
LOAD PS_Driver.o
LOAD startup.o
OUTPUT(Final_Project1.elf elf32-littlearm)

.debug_info     0x00000000      0x5a9
 .debug_info    0x00000000      0x135 Alarm.o
 .debug_info    0x00000135      0x103 driver.o
 .debug_info    0x00000238      0x11d main.o
 .debug_info    0x00000355       0xfc PS_Driver.o
 .debug_info    0x00000451      0x158 startup.o

.debug_abbrev   0x00000000      0x325
 .debug_abbrev  0x00000000       0xa5 Alarm.o
 .debug_abbrev  0x000000a5       0x9d driver.o
 .debug_abbrev  0x00000142       0xa5 main.o
 .debug_abbrev  0x000001e7       0x7c PS_Driver.o
 .debug_abbrev  0x00000263       0xc2 startup.o

.debug_loc      0x00000000      0x26c
 .debug_loc     0x00000000       0x90 Alarm.o
 .debug_loc     0x00000090       0xc8 driver.o
 .debug_loc     0x00000158       0x58 main.o
 .debug_loc     0x000001b0       0x58 PS_Driver.o
 .debug_loc     0x00000208       0x64 startup.o

.debug_aranges  0x00000000       0xa0
 .debug_aranges
                0x00000000       0x20 Alarm.o
 .debug_aranges
                0x00000020       0x20 driver.o
 .debug_aranges
                0x00000040       0x20 main.o
 .debug_aranges
                0x00000060       0x20 PS_Driver.o
```

```
 .debug_aranges
                0x00000080       0x20 startup.o

.debug_line     0x00000000      0x254
 .debug_line    0x00000000       0x71 Alarm.o
 .debug_line    0x00000071       0x99 driver.o
 .debug_line    0x0000010a       0x6c main.o
 .debug_line    0x00000176       0x5d PS_Driver.o
 .debug_line    0x000001d3       0x81 startup.o

.debug_str      0x00000000      0x231
 .debug_str     0x00000000      0x103 Alarm.o
                                0x158 (size before relaxing)
 .debug_str     0x00000103       0x55 driver.o
                                0x131 (size before relaxing)
 .debug_str     0x00000158       0x27 main.o
                                0x150 (size before relaxing)
 .debug_str     0x0000017f       0x30 PS_Driver.o
                                0x137 (size before relaxing)
 .debug_str     0x000001af       0x82 startup.o
                                0x13e (size before relaxing)

.comment        0x00000000       0x11
 .comment       0x00000000       0x11 Alarm.o
                                 0x12 (size before relaxing)
 .comment       0x00000000       0x12 driver.o
 .comment       0x00000000       0x12 main.o
 .comment       0x00000000       0x12 PS_Driver.o
 .comment       0x00000000       0x12 startup.o

.ARM.attributes
                0x00000000       0x33
 .ARM.attributes
                0x00000000       0x33 Alarm.o
 .ARM.attributes
                0x00000033       0x33 driver.o
 .ARM.attributes
                0x00000066       0x33 main.o
 .ARM.attributes
                0x00000099       0x33 PS_Driver.o
 .ARM.attributes
                0x000000cc       0x33 startup.o

.debug_frame    0x00000000      0x1b8
 .debug_frame   0x00000000       0x64 Alarm.o
 .debug_frame   0x00000064       0x78 driver.o
 .debug_frame   0x000000dc       0x48 main.o
 .debug_frame   0x00000124       0x48 PS_Driver.o
 .debug_frame   0x0000016c       0x4c startup.o
```

2. Symbols table

```
Final_Project1.elf:
20000008 B _E_bss
20000004 D _E_DATA
08000394 T _E_text
20000004 B _S_bss
20000000 D _S_DATA
20000000 D Al_Pval
20000004 B Al_Signal
20001008 B Alarm_state
2000100c B Alarm_state_id
080002d8 W Bus_fault_Handler
080002d8 T Default_Handler
080000fc T Delay
08000120 T getPressureVal
08000188 T GPIO_INITIALIZATION
080002d8 W H_fault_Handler
08000238 T main
080002d8 W MM_fault_Handler
080002d8 W NMI_Handler
0800001c T Pres_val
20001010 B PS_Pval
20001014 B PS_state
2000100d B PS_state_id
080002e4 T Reset_Handler
08000138 T Set_Alarm_actuator
08000208 T setup
20001008 B SP_top
08000064 T ST_High_Pressure
080000b0 T ST_Norm_Pressure
0800025c T ST_PS_reading
080002a4 T ST_PS_waiting
080002d8 W Usage_fault_Handler
08000000 T vectors
```

3. Sections table

```
Final_Project1.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000394  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000004  20000000  08000394  00010000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00001014  20000004  08000398  00010004  2**2
                  ALLOC
  3 .debug_info   000005a9  00000000  00000000  00010004  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev 00000325  00000000  00000000  000105ad  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    0000026c  00000000  00000000  000108d2  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 000000a0  00000000  00000000  00010b3e  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   00000256  00000000  00000000  00010bde  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_str    00000231  00000000  00000000  00010e34  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000011  00000000  00000000  00011065  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  00011076  2**0
                  CONTENTS, READONLY
 11 .debug_frame  000001b8  00000000  00000000  000110ac  2**2
                  CONTENTS, READONLY, DEBUGGING
```