First Term (Final Project 1)

Eng. Hesham mohamed Mostafa

My profile

heshammuhammed14@gmail.com (learn-in-depth.com)

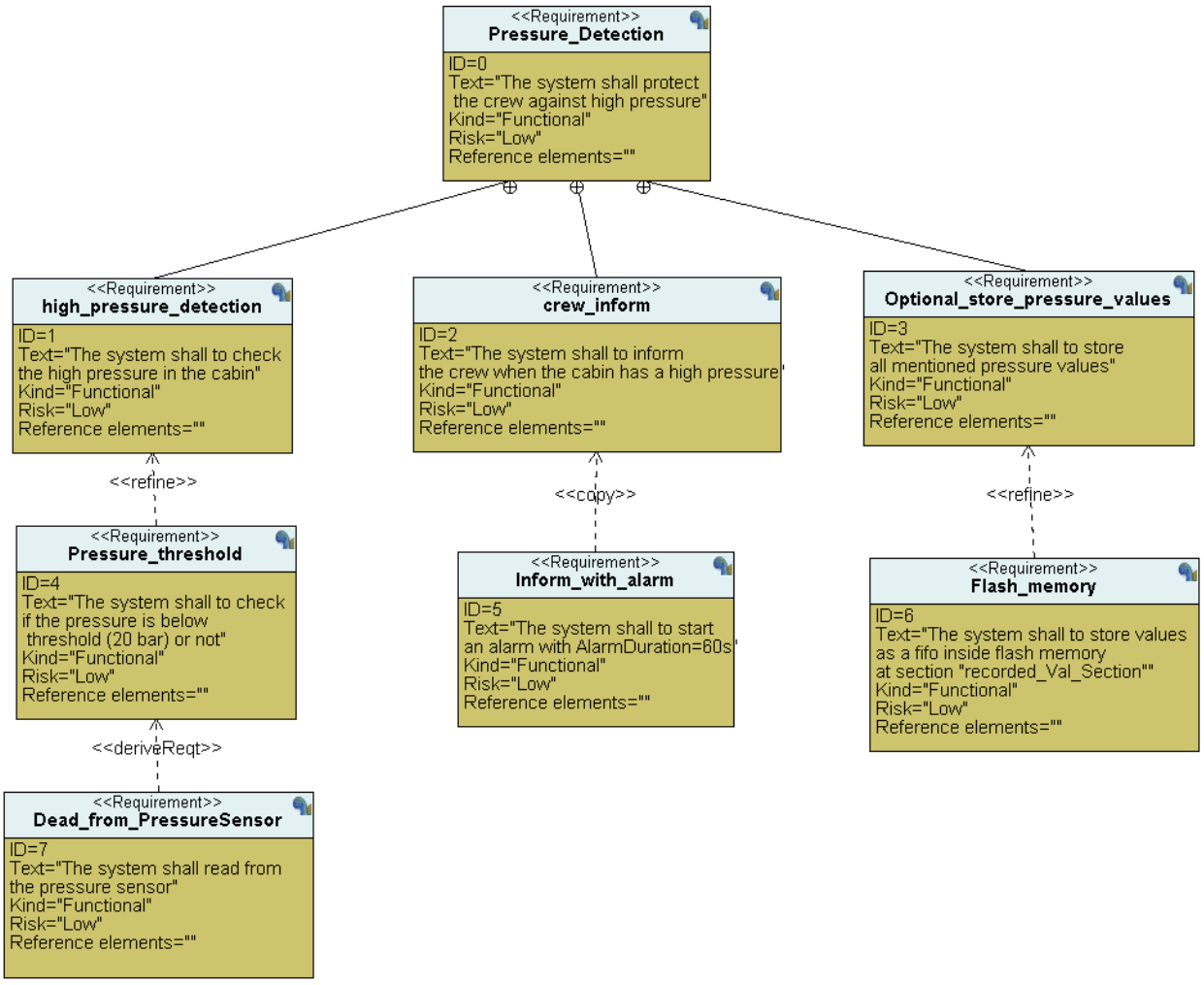
Specification

We need to deliver a Pressure Detection system.
after case studying, our system must do the following:

1- Informs the crew with an alarm when the pressure exceeds 20 bars.
2- The alarm duration equals 60 seconds.
3- Store pressure values into FLASH memory.

Requirements

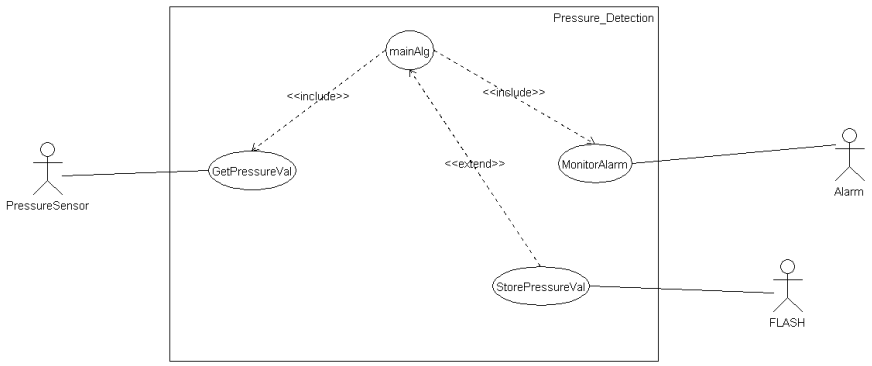
Our requirements diagram:



Analysis

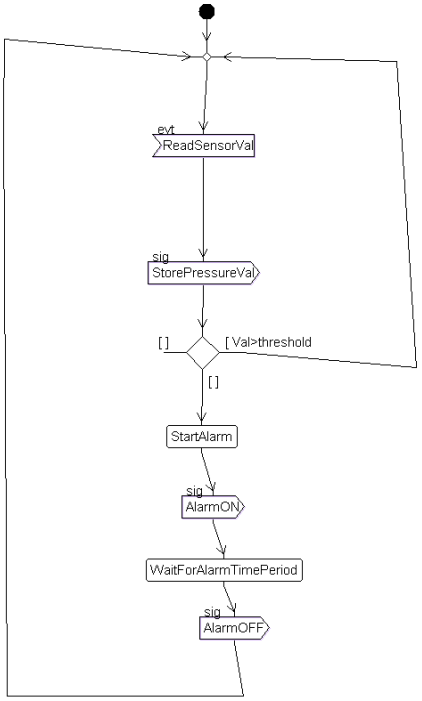
• Use Case Diagram

The following analysis helps us to know the main use cases and actors (sensors)

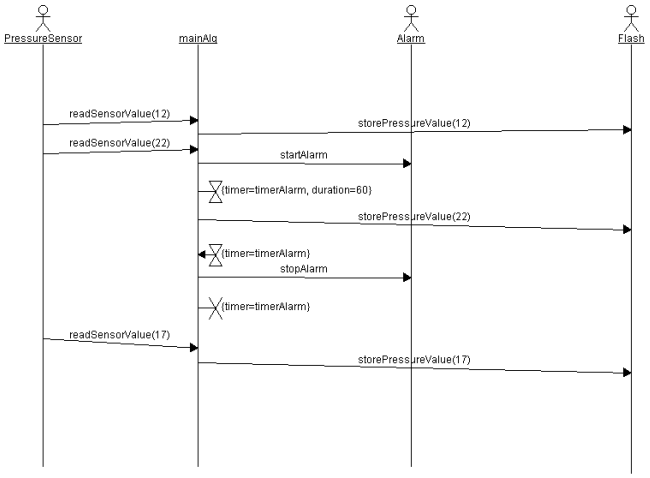


• Activity Diagram | Sequence Diagram

The following analysis helps us to know the flow of the system.

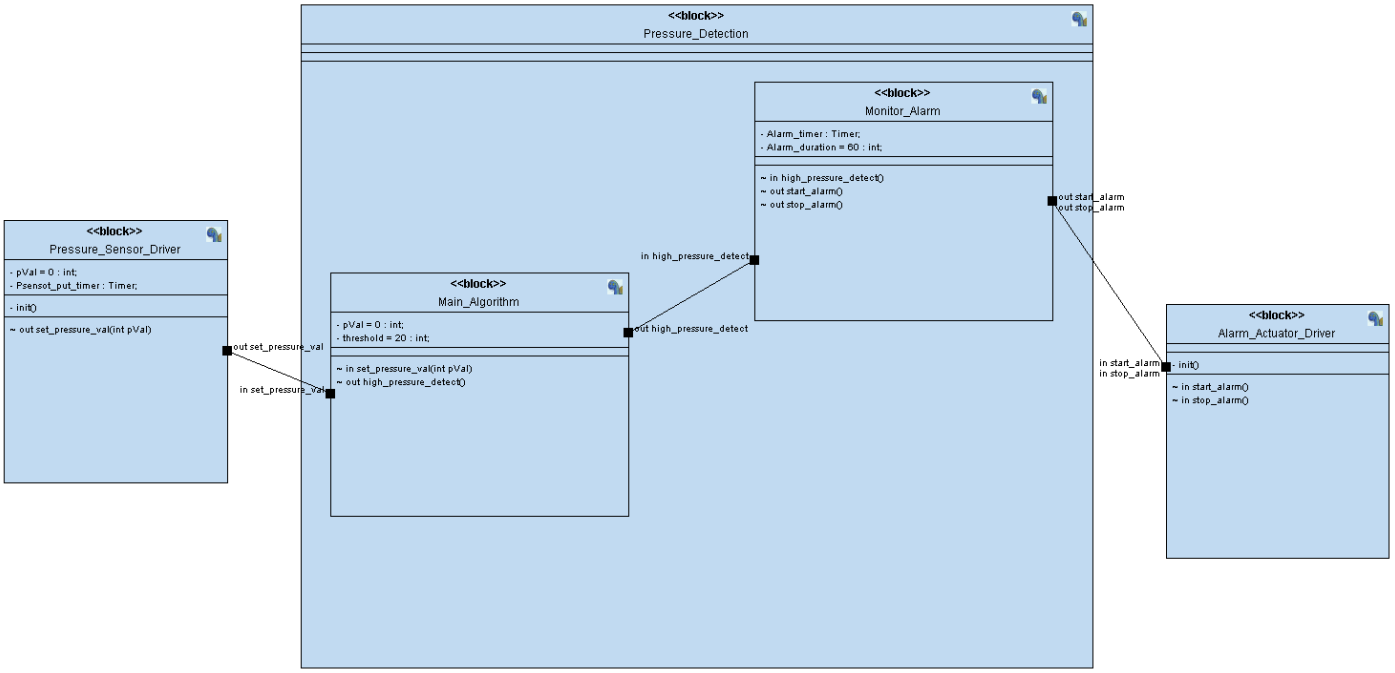


The following analysis helps us to know the algorithm of our system.

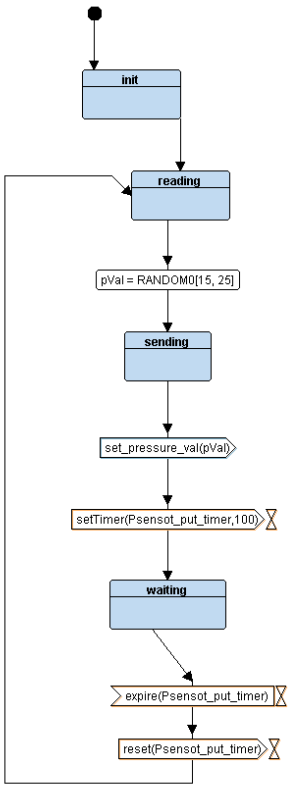


Design

1- Block Diagram



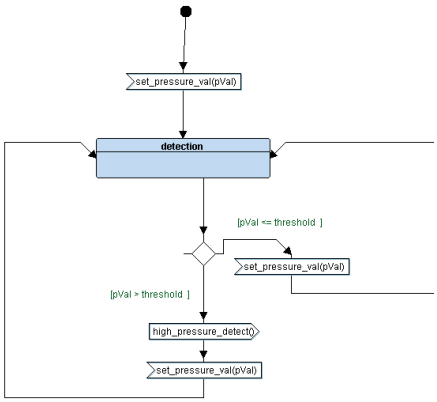
2- Pressure Sensor Driver – State diagram | PSD.h & PSD.c




```c
/*
 * PSD.c
 */

#include "PSD.h"
#include "driver.h"

//variables
static int pVal = 0;

//STATE Pointer to function
void (*PSD_state)();

//set random value for pressure
int PSD_set_pressure_random(int l, int r);

void PSD_init()
{
    //initialize Pressure Sensor Driver
    printf("PSD_init \n");
}

//first state
STATE_define(PSD_reading)
{
    //state_name
    PSD_state_id = PSD_reading;

    //state_action --> read pressure value
    pVal = getPressureVal();
    //pVal = PSD_set_pressure_random(15,25);

    //event check
    printf("PSD_reading state: pressure=%d\n", pVal);

    //moving to the next state
    STATE(PSD_sending)();

}

//second state
STATE_define(PSD_sending)
{
    //state_name
    PSD_state_id = PSD_sending;

    //event check
    printf("PSD_sending state: ...\n");

    //moving to the next state
    STATE(PSD_waiting)();

    //state_action --> send value to mainALG.c
    set_pressure_val(pVal);

}

//second state
STATE_define(PSD_waiting)
{
    //state_name
    PSD_state_id = PSD_waiting;

    //event check
    printf("PSD_waiting state: setting a delay of 90 seconds\n");

    //state action --> set a 90 second delay
    Delay(90000);

    //get back to reading state
    PSD_state = STATE(PSD_reading);

}

int PSD_set_pressure_random(int l, int r)
{
    //this function generates random number in range l and r

    int rand_num = (rand() % (r - l + 1)) + l;
    return rand_num;
}
```

```c
/*
 * PSD.h
 */

#ifndef PSD_H_
#define PSD_H_

#include "state.h"

//define states
enum{
    PSD_reading,
    PSD_sending,
    PSD_waiting
}PSD_state_id;

//declare states functions for PSD
STATE_define(PSD_reading);
STATE_define(PSD_sending);
STATE_define(PSD_waiting);

//initialize Pressure Sensor Driver
void PSD_init();

//STATE Pointer to function
void (*PSD_state)();

#endif /* PSD_H_ */
```

3- Main Algorithm – State diagram | ALG.c & ALG.h

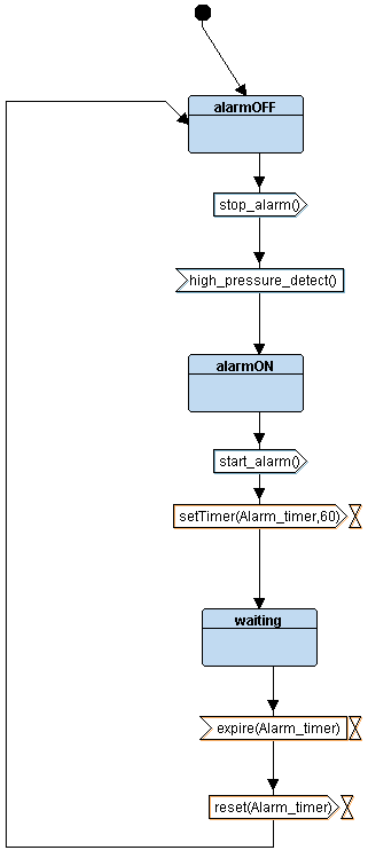



```c
/* AAD.c
 *
 */

#include "ALG.h"


//variables
static int ALG_pVal = 0;
static int ALG_threshold = 20;

//STATE Pointer to function
void (*ALG_state)();

//read from Pressure Sensor
void set_pressure_val(int pVal)
{
    ALG_pVal = pVal;
    printf("   PSD -----pVal=%d-----> ALG   \n", pVal);
    if(ALG_pVal<=ALG_threshold)
        ALG_state = STATE(detection);
    else
    {
        //first send signal to MonitorAlarm.c
        high_pressure_detect();

        //loop
        ALG_state = STATE(detection);
    }
}

STATE_define(detection)
{
    //state_name
    ALG_state_id = detection;

    //event check
    printf("ALG_detection state: ...\n");

    //loop
    ALG_state = STATE(detection);
}

```

ALG.h

```c
/* ALG.h
 *
 */

#ifndef ALG_H_
#define ALG_H_

#include "state.h"

//define states
enum{
    detection
}ALG_state_id;

//declare states functions for mainALG
STATE_define(detection);

//STATE Pointer to function
void (*ALG_state)();

#endif /* ALG_H_ */

```

4- Monitor Alarm – State diagram | MonitorAlarm.c & MonitorAlarm.h

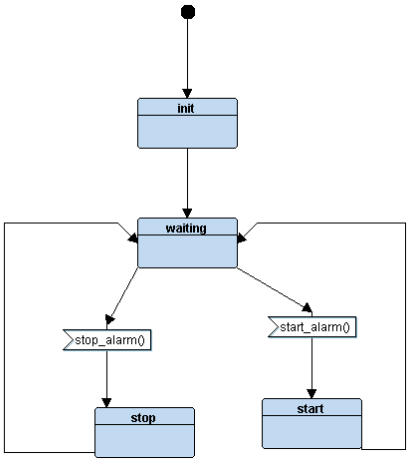



```c
/* MonitorAlarm.c
 *
 */

#include "MonitorAlarm.h"
#include "driver.h"

//variables
volatile static int alarmDuration = 60000;

//STATE Pointer to function
void (*M_state)();


void high_pressure_detect()
{
    printf("   ALG -----high pressure-----> MonitorAlarm   \n");
    //moving to M_alarmON state
    M_state = STATE(M_alarmON);
    //M_state();
}

//first state
STATE_define(M_alarmOFF)
{
    //state name
    M_state_id = M_alarmOFF;

    //event check
    printf("M_alarmOFF state: ...\n");

    //state action --> stop alarm
    stop_alarm();

}

//second state
STATE_define(M_alarmON)
{
    //state name
    M_state_id = M_alarmON;

    //event check
    printf("M_alarmOFF state: ...\n");

    //state action --> start alarm
    start_alarm();

    //moving to the next state
    M_state = STATE(M_waiting);
    //M_state();

}

//third state
STATE_define(M_waiting)
{
    //state name
    M_state_id = M_waiting;

    //event check
    printf("M_waiting state: starting a delay of 60 seconds\

    //state action --> set a 60 second delay
    Delay(alarmDuration);

    //moving to the next state
    M_state = STATE(M_alarmOFF);
    M_state();

}

```

MonitorAlarm.h

```c
/* MonitorAlarm.h
 *
 */

#ifndef MonitorAlarm_H_
#define MonitorAlarm_H_

#include "state.h"

//define states
enum{
    M_alarmOFF,
    M_alarmON,
    M_waiting
}M_state_id;

//declare states functions for MonitorAlarm
STATE_define(M_alarmOFF);
STATE_define(M_alarmON);
STATE_define(M_waiting);

//STATE Pointer to function
void (*M_state)();

#endif /* MonitorAlarm_H_ */

```

5- Alarm Actuator – State diagram | AAD.c & AAD.h




```c
/* AAD.c
 *
 */

#include "AAD.h"
#include "driver.h"

//STATE Pointer to function
void (*AAD_state_ptr)();
/*
STATE_define(AAD_waiting);
STATE_define(AAD_start);
STATE_define(AAD_stop);*/

void AAD_init()
{
    //initialize AAD
    //printf("Alarm_Actuator_driver_init \n");
}

STATE_define(AAD_waitingf)
{
    //state name
    AAD_state_id = AAD_waiting;

    //printf("AAD_waiting state: ...\n");
    AAD_state_ptr = STATE(AAD_waitingf);
}

STATE_define(AAD_startf)
{
    //printf("AAD_start state: Alarm ON\n");

    //state action --> set alarm actuator ON
    Set_Alarm_actuator(0);

    //moving to waiting state
    AAD_state_ptr = STATE(AAD_waitingf);
}


STATE_define(AAD_stopf)
{
    //printf("AAD_stop state: Alarm OFF\n");

    //state action --> set alarm actuator OFF
    Set_Alarm_actuator(1);

    //moving to waiting state
    AAD_state_ptr = STATE(AAD_waitingf);
}

void start_alarm()
{
    //printf("   MonitorAlarm -----start_alarm-----> A
    AAD_state_ptr = STATE(AAD_startf);
    AAD_state_ptr();
}

void stop_alarm()
{
    //printf("   MonitorAlarm -----stop_alarm-----> AA
    AAD_state_ptr = STATE(AAD_stopf);
    AAD_state_ptr();
}
```

AAD.h

```c
/* AAD.h
 *
 */

#ifndef AAD_H_
#define AAD_H_

#include "state.h"

//define states
enum{
    AAD_waiting,
    AAD_start,
    AAD_stop
}AAD_state_id;

//declare states functions for AAD
STATE_define(AAD_waitingf);
STATE_define(AAD_startf);
STATE_define(AAD_stopf);

//initialize Pressure Sensor Driver
//void AAD_init();

//STATE Pointer to function
void (*AAD_state_ptr)();

#endif /* AAD_H_ */
```

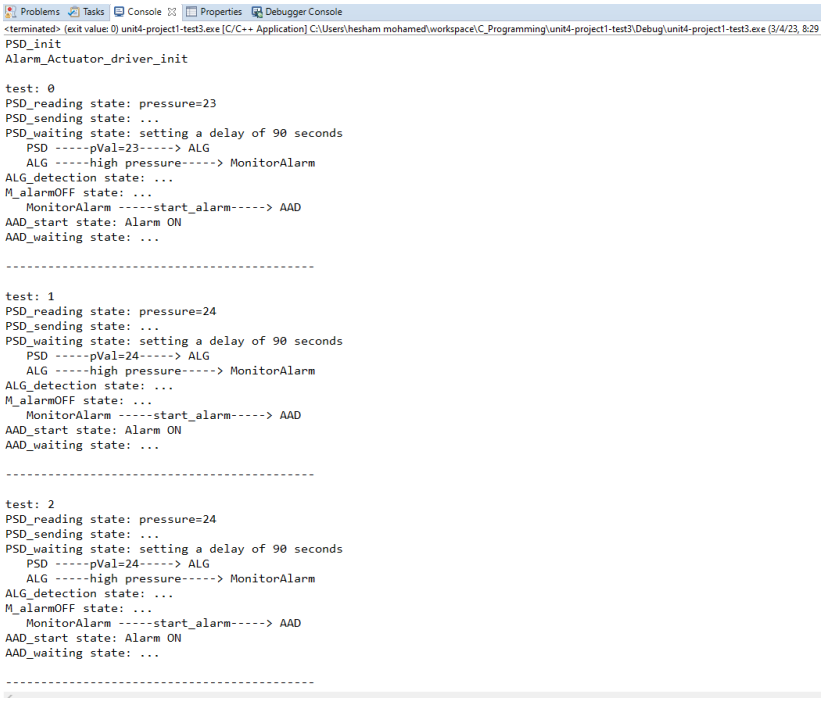
1- Main.c | state.h

```c
/* main.c
#include <stdio.h>
#include "ALG.h"
#include "PSD.h"
#include "AAD.h"
#include "MonitorAlarm.h"
#include "driver.h"

void setup()
{
    //init all the drivers
    GPIO_INITIALIZATION();
    //init IRQ ...
    //init HAL
    //init block
    PSD_init();
    AAD_init();
    //set states pointers for each block
    PSD_state = STATE (PSD_reading);
    ALG_state = STATE (detection);
    M_state = STATE (M_alarmOFF);
    AAD_state_ptr = STATE(AAD_waitingf);
}

int main()
{
    volatile int d, a =0;
    setup();

    while(a<30)
    {
        //call state for each block
        printf("\ntest: %d\n",a);
        PSD_state();
        ALG_state();
        M_state();
        AAD_state_ptr();
        for(d=0; d<1000; d++);
        a++;
        printf("\n-----------------------------------------\n");
    }

    return 0;
}
```

```c
/* state.h
 *
 */

#ifndef STATE_H_
#define STATE_H_

#include <stdio.h>
#include <stdlib.h>


//Automatic state function generated
#define STATE_define(_statFUN_) void ST_##_statFUN_()
#define STATE(_statFUN_) ST_##_statFUN_

//states connection
void set_pressure_val(int pVal);    //define it in the receiving block.c
void high_pressure_detect();        //define it in the receiving block.c
void start_alarm();                 //define it in the receiving block.c
void stop_alarm();                  //define it in the receiving block.c


#endif /* STATE_H_ */

```

testing Code

1- Successful output



```
Problems  Tasks  Console  Properties  Debugger Console
<terminated> (exit value: 0) unit4-project1-test3.exe [C/C++ Application] C:\Users\hesham mohamed\workspace\C_Programming\unit4-project1-test3\Debug\unit4-project1-test3.exe (3/4/23, 8:29
PSD_init
Alarm_Actuator_driver_init

test: 0
PSD_reading state: pressure=23
PSD_sending state: ...
PSD_waiting state: setting a delay of 90 seconds
   PSD -----pVal=23-----> ALG
   ALG -----high pressure-----> MonitorAlarm
ALG_detection state: ...
M_alarmOFF state: ...
   MonitorAlarm -----start_alarm-----> AAD
AAD_start state: Alarm ON
AAD_waiting state: ...

-----------------------------------------

test: 1
PSD_reading state: pressure=24
PSD_sending state: ...
PSD_waiting state: setting a delay of 90 seconds
   PSD -----pVal=24-----> ALG
   ALG -----high pressure-----> MonitorAlarm
ALG_detection state: ...
M_alarmOFF state: ...
   MonitorAlarm -----start_alarm-----> AAD
AAD_start state: Alarm ON
AAD_waiting state: ...

-----------------------------------------

test: 2
PSD_reading state: pressure=24
PSD_sending state: ...
PSD_waiting state: setting a delay of 90 seconds
   PSD -----pVal=24-----> ALG
   ALG -----high pressure-----> MonitorAlarm
ALG_detection state: ...
M_alarmOFF state: ...
   MonitorAlarm -----start_alarm-----> AAD
AAD_start state: Alarm ON
AAD_waiting state: ...

-----------------------------------------
```

Makefile & Startup & linker files

1- makefile

```
#@copyright : Hesham mohamed
#$< replaced by the dependencies
#$@ replaced by the target

CC=arm-none-eabi-
CFLAGS= -mcpu=cortex-m3 -gdwarf-2
INCS=-I .
LIBS=
#all .c files
SRC=$(wildcard *.c)
#anything .c replace it with .o
OBJ=$(SRC:.c=.o)
#all .s files
AS=$(wildcard *.s)
#anything .s replace it with .o
ASOBJ=$(AS:.s=.o)
Project_name=pressure_detection_project

all: $(Project_name).bin
    @echo "=======Build is Done, Hesham!======="

%.o: %.s
    $(CC)as.exe $(CFLAGS) $< -o $@

#Here, it will use this target for every dependence with target .o and dependence .c
%.o: %.c
    $(CC)gcc.exe -c $(INCS) $(CFLAGS) -mthumb $< -o $@

$(Project_name).elf: $(OBJ) $(ASOBJ)
    $(CC)ld.exe -T linker-script.ld $(LIBS) $(OBJ) $(ASOBJ) -Map=map_file.map -o $@


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

clean_all:
    rm *.o *.elf *.bin *.map

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

2- startup

```c
//startup.c
//Eng. Hesham Mohamed

#include <stdint.h>

//init the main() from main.c
extern int main(void);

//init the symbols from the linker
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;

//to specify the stack_top location there are two ways
/*1st: to make sure that .BSS doesn't interface with the stack_top section in sram,
 we will get its postion size from the linker*/
extern uint32_t _STACK_TOP;
//2nd: booking 1024B located by .bss through un intialized array of int 256 element
static unsigned long stack_top[256];

//intializing handler functions
void _DEFAULT_handler(void);
void _reset(void);
void _NMI_handler(void) __attribute__((weak,alias ("_DEFAULT_handler")));
void _HARD_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")));


//to define vectors section there are two ways
/*
//1st: using array
uint32_t vectors[] __attribute__((section(".vectors"))) = {
//  (uint32_t) &_STACK_TOP,                             //first way
    (uint32_t) ((unsigned long)stack_top + sizeof(stack_top)),      //second way
    (uint32_t) &_reset,
    (uint32_t) &_NMI_handler,
    (uint32_t) &_HARD_FAULT_handler,
    (uint32_t) &_MM_FAULT_handler,
    (uint32_t) &_BUS_FAULT_handler,
    (uint32_t) &_USAGE_FAULT_handler
};*/

/*2nd: using array of pointers to the handler functions*/
void (* const g_ptr_func_vectors[])() __attribute__((section(".vectors"))) = {
    (void (*)()) ((unsigned long)stack_top + sizeof(stack_top)),    //stack top pointer
    &_reset,
    &_NMI_handler,
    &_HARD_FAULT_handler,
    &_MM_FAULT_handler,
    &_BUS_FAULT_handler,
    &_USAGE_FAULT_handler
};

void _reset(void)
{
    int i=0;
    //we need to copy data section from flash to sram
    /*we cast it to uchar to move byte by byte, but as we have make an allignment on the .data section
      we can use uint casting*/
    uint32_t DATA_size = (unsigned char*)&_E_DATA - (unsigned char*)&_S_DATA;
    unsigned char* P_src = (unsigned char*)&_E_TEXT;
    unsigned char* P_dst = (unsigned char*)&_S_DATA;
    for(i = 0; i<DATA_size; i++)
    {
        *((unsigned char*) P_dst++) = *((unsigned char*) P_src++);
    }


    //init .bss section in sram with value = 0
    uint32_t BSS_size = (unsigned char*)&_E_BSS - (unsigned char*)&_S_BSS;
    P_dst = (unsigned char*)&_S_BSS;        //can i skip this step?? as _E_DATA == _S_BSS ??
    for(i = 0; i<BSS_size; i++)
    {
        *((unsigned char*) P_dst++) = (unsigned char)0;
    }

    //jumping to the main()
    main();
}

void _DEFAULT_handler(void)
{
    _reset();
}
```

3- linker script

```
/* linker script: CortexM3
    Eng. Hesham Mohamed
    */

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)
            . = ALIGN(4) ;
            _E_DATA = . ;
    }> sram AT> flash

    .bss : {
            _S_BSS = . ;
            *(.bss*)
            _E_BSS = . ;
            . = ALIGN(4) ;
            . = . + 0x1000;
            _STACK_TOP = . ;
    }> sram

}
```

4- Run makefile

```
hesham mohamed@DESKTOP-GPKPFCS MINGW32 /e/Study/Embedded System K.S/Unit 4/Project 1/Source
$ mingw32-make.exe
arm-none-eabi-gcc.exe -c -I . -mcpu=cortex-m3 -gdwarf-2 -mthumb PSD.c -o PSD.o
arm-none-eabi-gcc.exe -c -I . -mcpu=cortex-m3 -gdwarf-2 -mthumb startup.c -o startup.o
arm-none-eabi-gcc.exe -c -I . -mcpu=cortex-m3 -gdwarf-2 -mthumb main.c -o main.o
arm-none-eabi-gcc.exe -c -I . -mcpu=cortex-m3 -gdwarf-2 -mthumb ALG.c -o ALG.o
arm-none-eabi-gcc.exe -c -I . -mcpu=cortex-m3 -gdwarf-2 -mthumb MonitorAlarm.c -o MonitorAlarm.o
arm-none-eabi-gcc.exe -c -I . -mcpu=cortex-m3 -gdwarf-2 -mthumb AAD.c -o AAD.o
arm-none-eabi-gcc.exe -c -I . -mcpu=cortex-m3 -gdwarf-2 -mthumb driver.c -o driver.o
arm-none-eabi-ld.exe -T linker-script.ld  PSD.o startup.o main.o ALG.o MonitorAlarm.o AAD.o driver.o  -Map=map_file.map -o pressure_detection_project.elf
arm-none-eabi-objcopy.exe -O binary pressure_detection_project.elf pressure_detection_project.bin
=======Build is Done, Hesham!=======

hesham mohamed@DESKTOP-GPKPFCS MINGW32 /e/Study/Embedded System K.S/Unit 4/Project 1/Source
$
```

Check sections!

- Using objdump.exe

```
hesham mohamed@DESKTOP-GPKPFCS MINGW32 /e/Study/Embedded System K.S/Unit 4/Project 1/Source
$ arm-none-eabi-objdump.exe -h pressure_detection_project.elf

pressure_detection_project.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000540  08000000  08000000  00008000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         00000008  20000000  08000540  00010000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00001424  20000008  08000548  00010008  2**2
                  ALLOC
  3 .debug_info   00000926  00000000  00000000  00010008  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev 000004c2  00000000  00000000  0001092e  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    0000045c  00000000  00000000  00010df0  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 000000e0  00000000  00000000  0001124c  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   0000031e  00000000  00000000  0001132c  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_str    00000322  00000000  00000000  0001164a  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      00000011  00000000  00000000  0001196c  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  0001197d  2**0
                  CONTENTS, READONLY
 11 .debug_frame  00000300  00000000  00000000  000119b0  2**2
                  CONTENTS, READONLY, DEBUGGING
```

- Using readelf.exe

```
hesham mohamed@DESKTOP-GPKPFC5 MINGW32 /e/Study/Embedded System K.S/Unit 4/Project 1/Source
$ arm-none-eabi-readelf.exe -a pressure_detection_project.elf
ELF Header:
  Magic:   7f 45 4c 46 01 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF32
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           ARM
  Version:                           0x1
  Entry point address:               0x8000000
  Start of program headers:          52 (bytes into file)
  Start of section headers:          73040 (bytes into file)
  Flags:                             0x5000002, has entry point, Version5 EABI
  Size of this header:               52 (bytes)
  Size of program headers:           32 (bytes)
  Number of program headers:         2
  Size of section headers:           40 (bytes)
  Number of section headers:         16
  Section header string table index: 13

Section Headers:
  [Nr] Name              Type            Addr     Off    Size   ES Flg Lk Inf Al
  [ 0]                   NULL            00000000 000000 000000 00      0   0  0
  [ 1] .text             PROGBITS        08000000 008000 000540 00  AX  0   0  4
  [ 2] .data             PROGBITS        20000000 010000 000008 00  WA  0   0  4
  [ 3] .bss              NOBITS          20000008 010008 001424 00  WA  0   0  4
  [ 4] .debug_info       PROGBITS        00000000 010008 000926 00      0   0  1
  [ 5] .debug_abbrev     PROGBITS        00000000 01092e 0004c2 00      0   0  1
  [ 6] .debug_loc        PROGBITS        00000000 010df0 00045c 00      0   0  1
  [ 7] .debug_aranges    PROGBITS        00000000 01124c 0000e0 00      0   0  1
  [ 8] .debug_line       PROGBITS        00000000 01132c 00031e 00      0   0  1
  [ 9] .debug_str        PROGBITS        00000000 01164a 000322 01  MS  0   0  1
  [10] .comment          PROGBITS        00000000 01196c 000011 01  MS  0   0  1
  [11] .ARM.attributes   ARM_ATTRIBUTES  00000000 01197d 000033 00      0   0  1
  [12] .debug_frame      PROGBITS        00000000 0119b0 000300 00      0   0  4
  [13] .shstrtab         STRTAB          00000000 011cb0 00009d 00      0   0  1
  [14] .symtab           SYMTAB          00000000 011fd0 0005a0 10     15  46  4
  [15] .strtab           STRTAB          00000000 012570 0002ac 00      0   0  1
Key to Flags:
  W (write), A (alloc), X (execute), M (merge), S (strings)
  I (info), L (link order), G (group), T (TLS), E (exclude), x (unknown)
  O (extra OS processing required) o (OS specific), p (processor specific)

There are no section groups in this file.
```

```
Program Headers:
  Type           Offset   VirtAddr   PhysAddr   FileSiz MemSiz  Flg Align
  LOAD           0x008000 0x08000000 0x08000000 0x00540 0x00540 R E 0x8000
  LOAD           0x010000 0x20000000 0x08000540 0x00008 0x0142c RW  0x8000

 Section to Segment mapping:
  Segment Sections...
   00     .text
   01     .data .bss

There is no dynamic section in this file.

There are no relocations in this file.

There are no unwind sections in this file.

Symbol table '.symtab' contains 90 entries:
   Num:    Value  Size Type    Bind   Vis      Ndx Name
     0: 00000000     0 NOTYPE  LOCAL  DEFAULT  UND
     1: 08000000     0 SECTION LOCAL  DEFAULT    1
     2: 20000000     0 SECTION LOCAL  DEFAULT    2
     3: 20000008     0 SECTION LOCAL  DEFAULT    3
     4: 00000000     0 SECTION LOCAL  DEFAULT    4
     5: 00000000     0 SECTION LOCAL  DEFAULT    5
     6: 00000000     0 SECTION LOCAL  DEFAULT    6
     7: 00000000     0 SECTION LOCAL  DEFAULT    7
     8: 00000000     0 SECTION LOCAL  DEFAULT    8
     9: 00000000     0 SECTION LOCAL  DEFAULT    9
    10: 00000000     0 SECTION LOCAL  DEFAULT   10
    11: 00000000     0 SECTION LOCAL  DEFAULT   11
    12: 00000000     0 SECTION LOCAL  DEFAULT   12
    13: 00000000     0 FILE    LOCAL  DEFAULT  ABS startup.c
    14: 2000000c     0 NOTYPE  LOCAL  DEFAULT    3 $d
    15: 2000000c     0 NOTYPE  LOCAL  DEFAULT    3 stack_top
    16: 08000000     0 NOTYPE  LOCAL  DEFAULT    1 $d
    17: 080000ac     0 NOTYPE  LOCAL  DEFAULT    1 $t
    18: 0000008c     0 NOTYPE  LOCAL  DEFAULT   12 $d
    19: 00000000     0 FILE    LOCAL  DEFAULT  ABS PSD.c
    20: 20000008     0 NOTYPE  LOCAL  DEFAULT    3 $d
    21: 20000008     0 NOTYPE  LOCAL  DEFAULT    3 pVal
    22: 0800001c     0 NOTYPE  LOCAL  DEFAULT    1 $t
    23: 00000010     0 NOTYPE  LOCAL  DEFAULT   12 $d
    24: 00000000     0 FILE    LOCAL  DEFAULT  ABS main.c
    25: 08000170     0 NOTYPE  LOCAL  DEFAULT    1 $t
    26: 000000d8     0 NOTYPE  LOCAL  DEFAULT   12 $d
    27: 00000000     0 FILE    LOCAL  DEFAULT  ABS ALG.c
    28: 2000040c     0 NOTYPE  LOCAL  DEFAULT    3 $d
    29: 2000040c     0 NOTYPE  LOCAL  DEFAULT    3 ALG_pVal
    30: 20000000     0 NOTYPE  LOCAL  DEFAULT    2 $d
    31: 20000000     4 OBJECT  LOCAL  DEFAULT    2 ALG_threshold
    32: 08000244     0 NOTYPE  LOCAL  DEFAULT    1 $t
    33: 00000124     0 NOTYPE  LOCAL  DEFAULT   12 $d
    34: 00000000     0 FILE    LOCAL  DEFAULT  ABS MonitorAlarm.c
    35: 20000004     0 NOTYPE  LOCAL  DEFAULT    2 $d
    36: 20000004     4 OBJECT  LOCAL  DEFAULT    2 alarmDuration
    37: 080002d0     0 NOTYPE  LOCAL  DEFAULT    1 $t
```

```
    38: 00000160     0 NOTYPE  LOCAL  DEFAULT   12 $d
    39: 00000000     0 FILE    LOCAL  DEFAULT  ABS AAD.c
    40: 08000374     0 NOTYPE  LOCAL  DEFAULT    1 $t
    41: 000001e8     0 NOTYPE  LOCAL  DEFAULT   12 $d
    42: 00000000     0 FILE    LOCAL  DEFAULT  ABS driver.c
    43: 08000434     0 NOTYPE  LOCAL  DEFAULT    1 $t
    44: 00000298     0 NOTYPE  LOCAL  DEFAULT   12 $d
    45: 00000000     0 FILE    LOCAL  DEFAULT  ABS
    46: 08000540     0 NOTYPE  GLOBAL DEFAULT    1 _E_TEXT
    47: 080004c1   126 FUNC    GLOBAL DEFAULT    1 GPIO_INITIALIZATION
    48: 08000165    10 FUNC    WEAK   DEFAULT    1 _HARD_FAULT_handler
    49: 20001410     4 OBJECT  GLOBAL DEFAULT    3 PSD_state
    50: 20001418     1 OBJECT  GLOBAL DEFAULT    3 AAD_state_id
    51: 08000165    10 FUNC    GLOBAL DEFAULT    1 _DEFAULT_handler
    52: 20001414     1 OBJECT  GLOBAL DEFAULT    3 PSD_state_id
    53: 08000375    10 FUNC    GLOBAL DEFAULT    1 AAD_init
    54: 08000029    40 FUNC    GLOBAL DEFAULT    1 ST_PSD_reading
    55: 080002ed    24 FUNC    GLOBAL DEFAULT    1 ST_M_alarmOFF
    56: 2000141c     4 OBJECT  GLOBAL DEFAULT    3 ALG_state
    57: 080003ad    32 FUNC    GLOBAL DEFAULT    1 ST_AAD_startf
    58: 080003ed    36 FUNC    GLOBAL DEFAULT    1 start_alarm
    59: 20000008     0 NOTYPE  GLOBAL DEFAULT    2 _E_DATA
    60: 08000459    24 FUNC    GLOBAL DEFAULT    1 getPressureVal
    61: 08000381    42 FUNC    GLOBAL DEFAULT    1 ST_AAD_waitingf
    62: 080003cd    32 FUNC    GLOBAL DEFAULT    1 ST_AAD_stopf
    63: 20000410     0 NOTYPE  GLOBAL DEFAULT    3 _E_BSS
    64: 08000471    80 FUNC    GLOBAL DEFAULT    1 Set_Alarm_actuator
    65: 080001cd   120 FUNC    GLOBAL DEFAULT    1 main
    66: 080002a5    42 FUNC    GLOBAL DEFAULT    1 ST_detection
    67: 08000331    66 FUNC    GLOBAL DEFAULT    1 ST_M_waiting
    68: 20001420     4 OBJECT  GLOBAL DEFAULT    3 M_state
    69: 08000165    10 FUNC    WEAK   DEFAULT    1 _BUS_FAULT_handler
    70: 08000305    42 FUNC    GLOBAL DEFAULT    1 ST_M_alarmON
    71: 08000000    28 OBJECT  GLOBAL DEFAULT    1 g_ptr_func_vectors
    72: 20000000     0 NOTYPE  GLOBAL DEFAULT    2 _S_DATA
    73: 20001410     0 NOTYPE  GLOBAL DEFAULT    3 _STACK_TOP
    74: 08000051    40 FUNC    GLOBAL DEFAULT    1 ST_PSD_sending
    75: 08000079    50 FUNC    GLOBAL DEFAULT    1 ST_PSD_waiting
    76: 0800001d    10 FUNC    GLOBAL DEFAULT    1 PSD_init
    77: 20001424     1 OBJECT  GLOBAL DEFAULT    3 ALG_state_id
    78: 08000171    90 FUNC    GLOBAL DEFAULT    1 setup
    79: 08000435    34 FUNC    GLOBAL DEFAULT    1 Delay
    80: 08000165    10 FUNC    WEAK   DEFAULT    1 _NMI_handler
    81: 20000008     0 NOTYPE  GLOBAL DEFAULT    3 _S_BSS
    82: 080000ad   182 FUNC    GLOBAL DEFAULT    1 _reset
    83: 08000411    36 FUNC    GLOBAL DEFAULT    1 stop_alarm
    84: 20001425     1 OBJECT  GLOBAL DEFAULT    3 M_state_id
    85: 20001428     4 OBJECT  GLOBAL DEFAULT    3 AAD_state_ptr
    86: 080002d1    28 FUNC    GLOBAL DEFAULT    1 high_pressure_detect
    87: 08000165    10 FUNC    WEAK   DEFAULT    1 _MM_FAULT_handler
    88: 08000165    10 FUNC    WEAK   DEFAULT    1 _USAGE_FAULT_handler
```

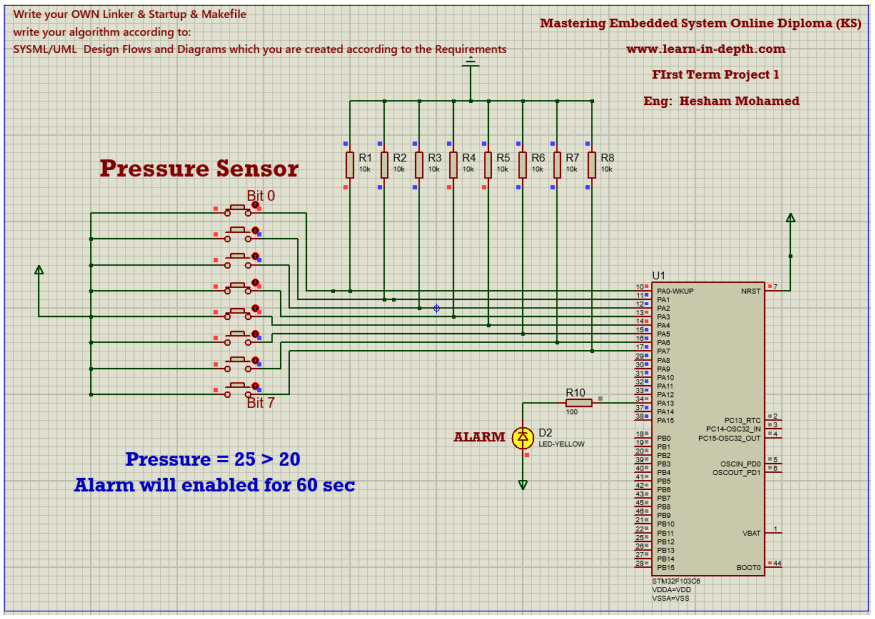
```
No version information found in this file.
Attribute Section: aeabi
File Attributes
  Tag_CPU_name: "Cortex-M3"
  Tag_CPU_arch: v7
  Tag_CPU_arch_profile: Microcontroller
  Tag_THUMB_ISA_use: Thumb-2
  Tag_ABI_PCS_wchar_t: 4
  Tag_ABI_FP_denormal: Needed
  Tag_ABI_FP_exceptions: Needed
  Tag_ABI_FP_number_model: IEEE 754
  Tag_ABI_align_needed: 8-byte
  Tag_ABI_align_preserved: 8-byte, except leaf SP
  Tag_ABI_enum_size: small
  Tag_ABI_optimization_goals: Aggressive Debug
  Tag_CPU_unaligned_access: v6
```

Map file

```
Allocating common symbols
Common symbol       size              file

PSD_state           0x4               PSD.o
AAD_state_id        0x1               main.o
PSD_state_id        0x1               PSD.o
ALG_state           0x4               main.o
M_state             0x4               main.o
ALG_state_id        0x1               main.o
M_state_id          0x1               main.o
AAD_state_ptr       0x4               main.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     0x540
 *(.vectors*)
 .vectors       0x08000000      0x1c startup.o
                0x08000000                g_ptr_func_vectors
 *(.text*)
 .text          0x0800001c      0x90 PSD.o
                0x0800001c                PSD_init
                0x08000028                ST_PSD_reading
                0x08000050                ST_PSD_sending
                0x08000078                ST_PSD_waiting
 .text          0x080000ac      0xc4 startup.o
                0x080000ac                _reset
                0x08000164                _BUS_FAULT_handler
                0x08000164                _NMI_handler
                0x08000164                _DEFAULT_handler
                0x08000164                _MM_FAULT_handler
                0x08000164                _USAGE_FAULT_handler
                0x08000164                _HARD_FAULT_handler
 .text          0x08000170      0xd4 main.o
                0x08000170                setup
                0x080001cc                main
 .text          0x08000244      0x8c ALG.o
                0x08000244                set_pressure_val
                0x080002a4                ST_detection
 .text          0x080002d0      0xa4 MonitorAlarm.o
                0x080002d0                high_pressure_detect
                0x080002ec                ST_M_alarmOFF
                0x08000304                ST_M_alarmON
                0x08000330                ST_M_waiting
 .text          0x08000374      0xc0 AAD.o
                0x08000374                AAD_init
                0x08000380                ST_AAD_waitingf
                0x080003ac                ST_AAD_startf
                0x080003cc                ST_AAD_stopf
                0x080003ec                start_alarm
                0x08000410                stop_alarm
 .text          0x08000434      0x10c driver.o
                0x08000434                Delay
                0x08000458                getPressureVal
                0x08000470                Set_Alarm_actuator
                0x080004c0                GPIO_INITIALIZATION
 *(.rodata)
                0x08000540                _E_TEXT = .

.glue_7         0x08000540        0x0
 .glue_7        0x00000000        0x0 linker stubs

.glue_7t        0x08000540        0x0
 .glue_7t       0x00000000        0x0 linker stubs

.vfp11_veneer   0x08000540        0x0
 .vfp11_veneer  0x00000000        0x0 linker stubs

.v4_bx          0x08000540        0x0
 .v4_bx         0x00000000        0x0 linker stubs

.iplt           0x08000540        0x0
 .iplt          0x00000000        0x0 PSD.o

.rel.dyn        0x08000540        0x0
 .rel.iplt      0x00000000        0x0 PSD.o

.data           0x20000000        0x8 load address 0x08000540
                0x20000000                _S_DATA = .
 *(.data)
 .data          0x20000000        0x0 PSD.o
 .data          0x20000000        0x0 startup.o
 .data          0x20000000        0x0 main.o
 .data          0x20000000        0x4 ALG.o
 .data          0x20000004        0x4 MonitorAlarm.o
 .data          0x20000008        0x0 AAD.o
 .data          0x20000008        0x0 driver.o
                0x20000008                . = ALIGN (0x4)
                0x20000008                _E_DATA = .

.igot.plt       0x20000008        0x0 load address 0x08000548
 .igot.plt      0x00000000        0x0 PSD.o

.bss            0x20000008     0x1424 load address 0x08000548
                0x20000008                _S_BSS = .
 *(.bss*)
 .bss           0x20000008        0x4 PSD.o
 .bss           0x2000000c      0x400 startup.o
 .bss           0x2000040c        0x0 main.o
 .bss           0x2000040c        0x4 ALG.o
 .bss           0x20000410        0x0 MonitorAlarm.o
 .bss           0x20000410        0x0 AAD.o
 .bss           0x20000410        0x0 driver.o
                0x20000410                _E_BSS = .
                0x20000410                . = ALIGN (0x4)
                0x20001410                . = (. + 0x1000)
 *fill*         0x20000410     0x1000
                0x20001410                _STACK_TOP = .
 COMMON         0x20001410        0x5 PSD.o
                0x20001410                PSD_state
                0x20001414                PSD_state_id
 *fill*         0x20001415        0x3
 COMMON         0x20001418       0x14 main.o
                0x20001418                AAD_state_id
                0x2000141c                ALG_state
                0x20001420                M_state
                0x20001424                ALG_state_id
                0x20001425                M_state_id
                0x20001428                AAD_state_ptr
LOAD PSD.o
LOAD startup.o
LOAD main.o
LOAD ALG.o
LOAD MonitorAlarm.o
LOAD AAD.o
LOAD driver.o
OUTPUT(pressure_detection_project.elf elf32-littlearm)

.debug_info     0x00000000      0x926
 .debug_info    0x00000000      0x12b PSD.o
 .debug_info    0x0000012b      0x18f startup.o
 .debug_info    0x000002ba      0x1db main.o
 .debug_info    0x00000495      0x11a ALG.o
 .debug_info    0x000005af      0x130 MonitorAlarm.o
 .debug_info    0x000006df      0x144 AAD.o
 .debug_info    0x00000823      0x103 driver.o

.debug_abbrev   0x00000000      0x4c2
 .debug_abbrev  0x00000000       0xa1 PSD.o
 .debug_abbrev  0x000000a1       0xd7 startup.o
 .debug_abbrev  0x00000178       0xbf main.o
 .debug_abbrev  0x00000237       0xb4 ALG.o
 .debug_abbrev  0x000002eb       0xa8 MonitorAlarm.o
 .debug_abbrev  0x00000393       0x92 AAD.o
 .debug_abbrev  0x00000425       0x9d driver.o

.debug_loc      0x00000000      0x45c
 .debug_loc     0x00000000       0xb0 PSD.o
 .debug_loc     0x000000b0       0x64 startup.o
 .debug_loc     0x00000114       0x64 main.o
 .debug_loc     0x00000178       0x64 ALG.o
 .debug_loc     0x000001dc       0xb0 MonitorAlarm.o
 .debug_loc     0x0000028c      0x108 AAD.o
 .debug_loc     0x00000394       0xc8 driver.o

.debug_aranges  0x00000000       0xe0
 .debug_aranges
                0x00000000       0x20 PSD.o
 .debug_aranges
                0x00000020       0x20 startup.o
 .debug_aranges
                0x00000040       0x20 main.o
 .debug_aranges
                0x00000060       0x20 ALG.o
 .debug_aranges
                0x00000080       0x20 MonitorAlarm.o
 .debug_aranges
                0x000000a0       0x20 AAD.o
 .debug_aranges
                0x000000c0       0x20 driver.o

.debug_line     0x00000000      0x31e
 .debug_line    0x00000000       0x4e PSD.o
 .debug_line    0x0000004e       0xad startup.o
 .debug_line    0x000000fb       0x8c main.o
 .debug_line    0x00000187       0x48 ALG.o
 .debug_line    0x000001cf       0x61 MonitorAlarm.o
 .debug_line    0x00000230       0x53 AAD.o
 .debug_line    0x00000283       0x9b driver.o

.debug_str      0x00000000      0x322
 .debug_str     0x00000000       0xed PSD.o
                                 0x14f (size before relaxing)
 .debug_str     0x000000ed       0x8d startup.o
                                 0x15b (size before relaxing)
 .debug_str     0x0000017a       0x7f main.o
                                 0x1b6 (size before relaxing)
 .debug_str     0x000001f9       0x3b ALG.o
                                 0x134 (size before relaxing)
 .debug_str     0x00000234       0x5a MonitorAlarm.o
                                 0x15f (size before relaxing)
 .debug_str     0x0000028e       0x48 AAD.o
                                 0x15e (size before relaxing)
 .debug_str     0x000002d6       0x4c driver.o
                                 0x128 (size before relaxing)

.comment        0x00000000       0x11
 .comment       0x00000000       0x11 PSD.o
                                 0x12 (size before relaxing)
 .comment       0x00000000       0x12 startup.o
 .comment       0x00000000       0x12 main.o
 .comment       0x00000000       0x12 ALG.o
 .comment       0x00000000       0x12 MonitorAlarm.o
 .comment       0x00000000       0x12 AAD.o
 .comment       0x00000000       0x12 driver.o

.ARM.attributes
                0x00000000       0x33
.ARM.attributes
                0x00000000       0x33 PSD.o
.ARM.attributes
                0x00000033       0x33 startup.o
.ARM.attributes
                0x00000066       0x33 main.o
.ARM.attributes
                0x00000099       0x33 ALG.o
.ARM.attributes
                0x000000cc       0x33 MonitorAlarm.o
.ARM.attributes
                0x000000ff       0x33 AAD.o
.ARM.attributes
                0x00000132       0x33 driver.o

.debug_frame    0x00000000      0x300
 .debug_frame   0x00000000       0x7c PSD.o
 .debug_frame   0x0000007c       0x4c startup.o
 .debug_frame   0x000000c8       0x4c main.o
 .debug_frame   0x00000114       0x48 ALG.o
 .debug_frame   0x0000015c       0x7c MonitorAlarm.o
 .debug_frame   0x000001d8       0xb0 AAD.o
 .debug_frame   0x00000288       0x78 driver.o
```

Simulation results

1- Alarm ON



1- Alarm OFF