Pressure Detection Project

Mastering Embedded Systems Diploma

www.learn-in-depth.com

First Term (Final Project 1)

Eng. Abdelrahman Mohamed Adel Tawfik

1. Case Study

➢ Requirements

The client requires a system with the following specifications:

1. Pressure controller informing the cabin crew when pressure exceeds 20 bars.
2. The informing method is an alarm operating for 60 seconds.
3. Keep track of the measured values (optional).

➢ Assumptions

The following assumptions are made:

1. No setup or shut down for the microcontroller.
2. No maintenance for the microcontroller.
3. Neither the pressure sensor nor the alarm fails
4. No power cuts for the microcontroller

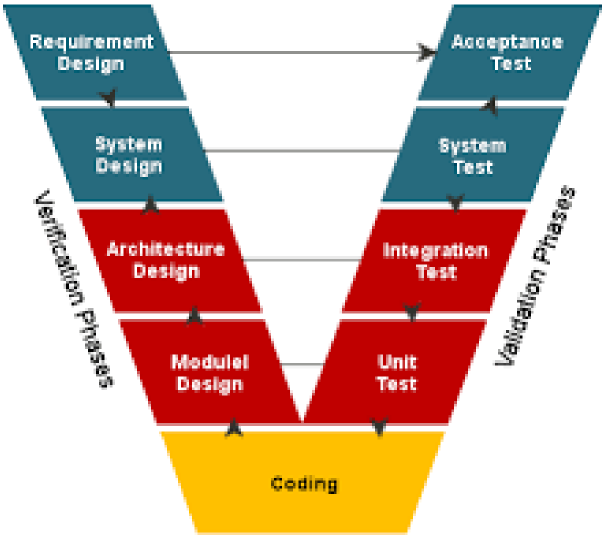
➢ Versioning

The possibility of storing pressure sensor readings might be included in a future version.

2. Method

Software Development Lifecycle and Software Testing Lifecycle

The (SDLC) and (STLC) will be approached based on the V-Model.



Requirements Gathering and Analysis: The first phase of the V-Model is the requirements gathering and analysis phase, where the customer's requirements for the software are gathered and analyzed to determine the scope of the project.

Design: In the design phase, the software architecture and design are developed, including the high-level design and detailed design.

Implementation: In the implementation phase, the software is actually built based on the design.

Testing: In the testing phase, the software is tested to ensure that it meets the customer's requirements and is of high quality.

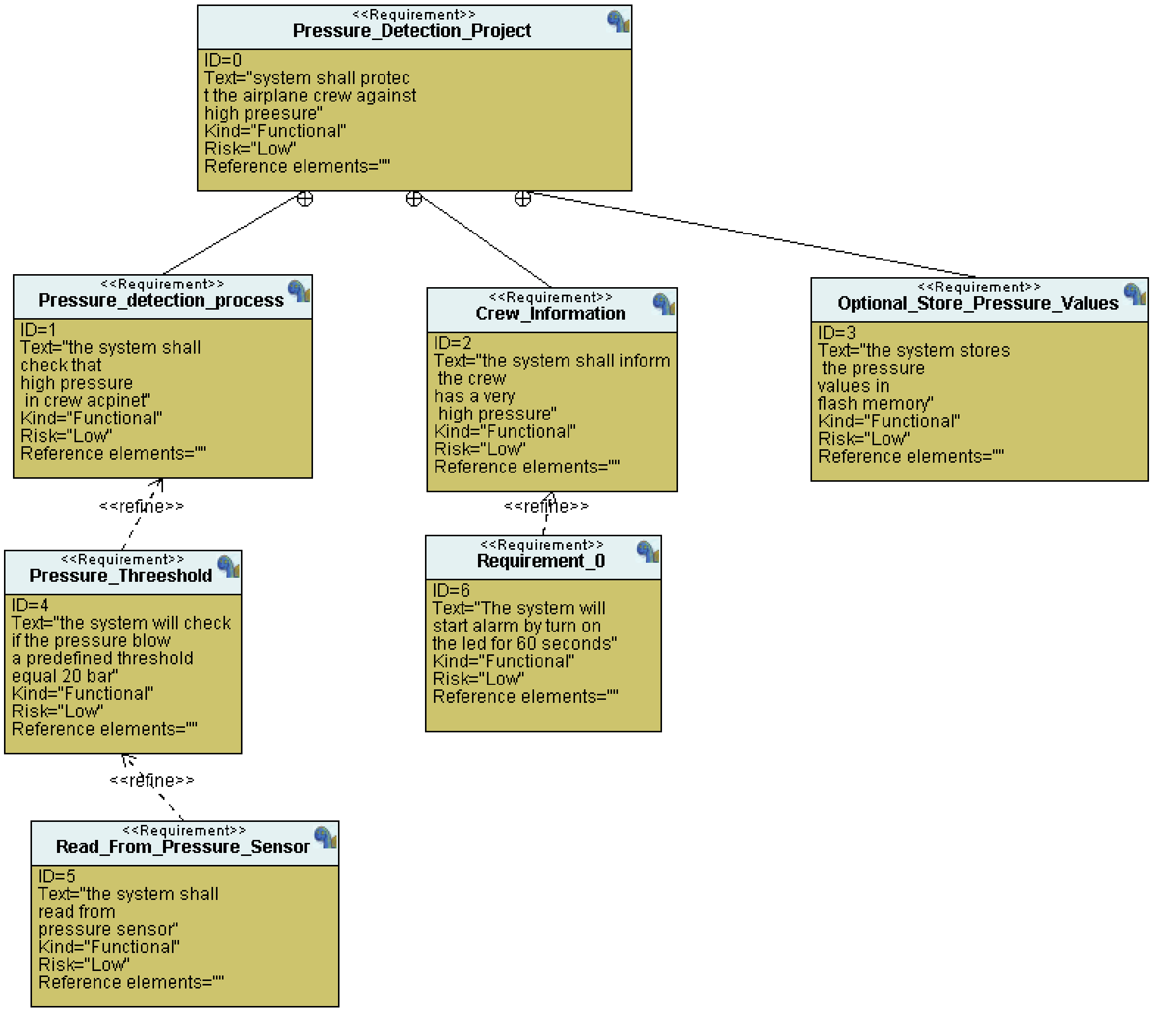
Deployment: In the deployment phase, the software is deployed and put into use.

Maintenance: In the maintenance phase, the software is maintained to ensure that it continues to meet the customer's needs and expectations.

The V-Model is often used in safety-critical systems, such as aerospace and defense systems, because of its emphasis on thorough testing and its ability to clearly define the steps involved in the software development process.

3. System Requirements

Requirement model


<<Requirement>>
Pressure_Detection_Project

ID=0
Text="system shall protec
t the airplane crew against
high preesure"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Pressure_detection_process

ID=1
Text="the system shall
check that
high pressure
in crew acpinet"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Crew_Information

ID=2
Text="the system shall inform
the crew
has a very
high pressure"
Kind="Functional"
Risk="Low"
Reference elements=""

<<Requirement>>
Optional_Store_Pressure_Values

ID=3
Text="the system stores
the pressure
values in
flash memory"
Kind="Functional"
Risk="Low"
Reference elements=""

ID=4
Text="the system will check
if the pressure blow
a predefined threshold
equal 20 bar"
Kind="Functional"
Risk="Low"
Reference elements=""

ID=6
Text="The system will
start alarm by turn on
the led for 60 seconds"
Kind="Functional"
Risk="Low"
Reference elements=""

ID=5
Text="the system shall
read from
pressure sensor"
Kind="Functional"
Risk="Low"
Reference elements=""

4. Design Space Exploration

Microcontroller: one stm32f103c8t6 SoC will be used as it meets all the needed technical requirements such as: suitable processor, acceptable flash memory size and small size as well as being cost efficient.

Overview: The STM32F103C8T6 is a medium density performance line, ARM Cortex-M3 32bit microcontroller in 48 pin LQFP package. It incorporates high performance RISC core with 72MHz operating frequency, high speed embedded memories, extensive range of enhanced I/Os and peripherals connected to two APB buses. The STM32F103C8T6 features 12bit ADC, timers, PWM timer, standard and advanced communication interfaces. A comprehensive set of power saving mode allows the design of low power applications.

Features:

Operating voltage range from 2V to 3.6V

64Kbytes of flash memory

20Kbytes of SRAM

CRC calculation unit, 96bit unique ID

Two 12bit, 1µs A/D converter (up to 10 channels)

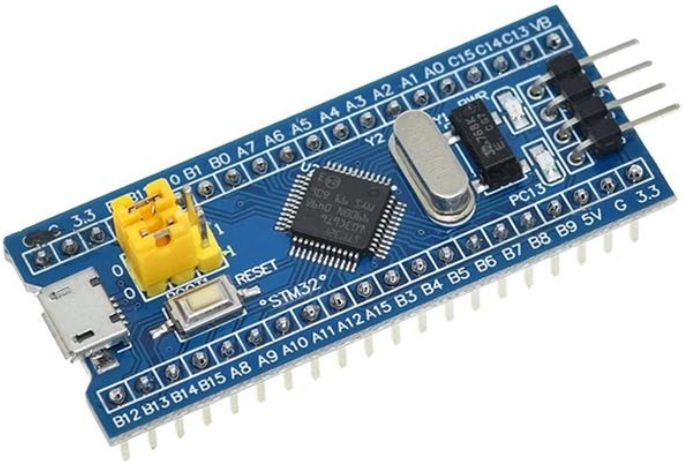
7 channel DMA controller, 3 general purpose timer and 1 advanced control timer

37 fast I/O ports

Serial wire debug (SWD) and JTAG interfaces

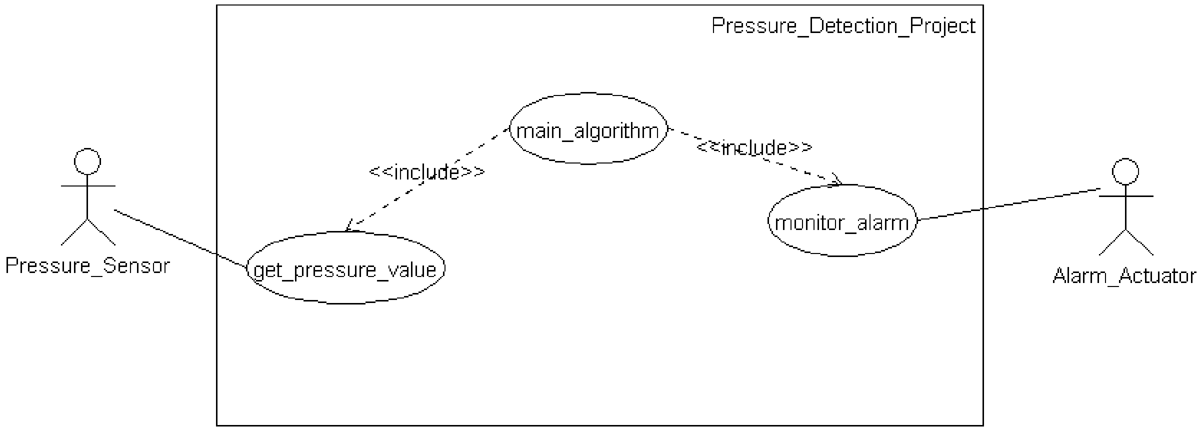
Two SPI, two I2C, three USART, one USB and one CAN interfaces

Ambient operating temperature range from -40°C to 85°C

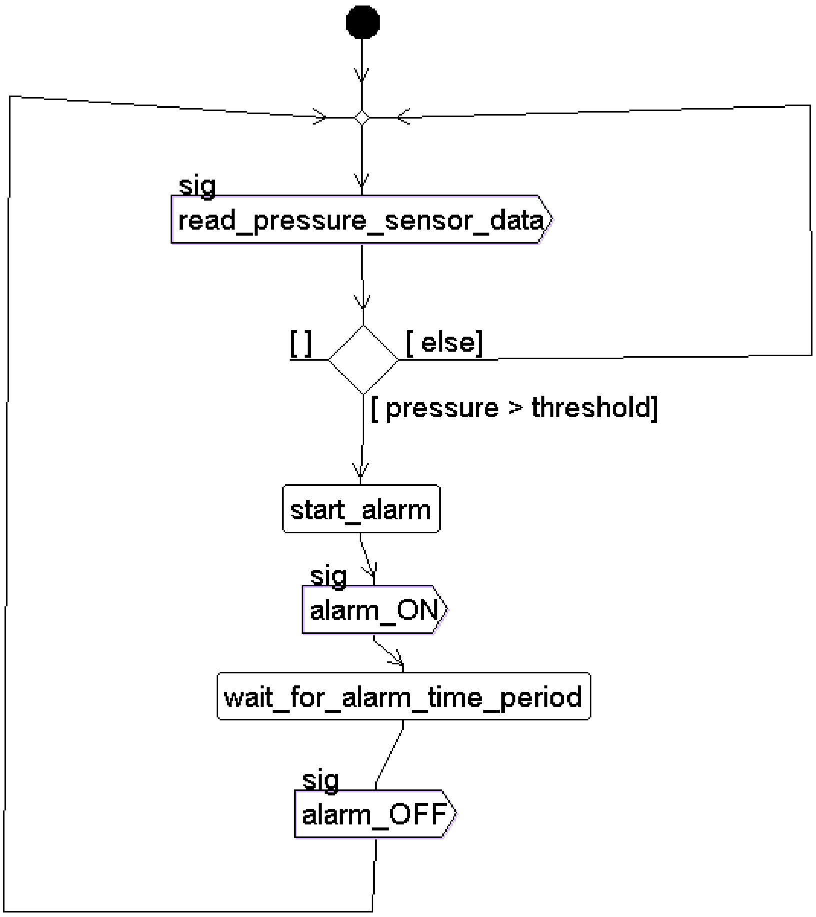


5. System Analysis

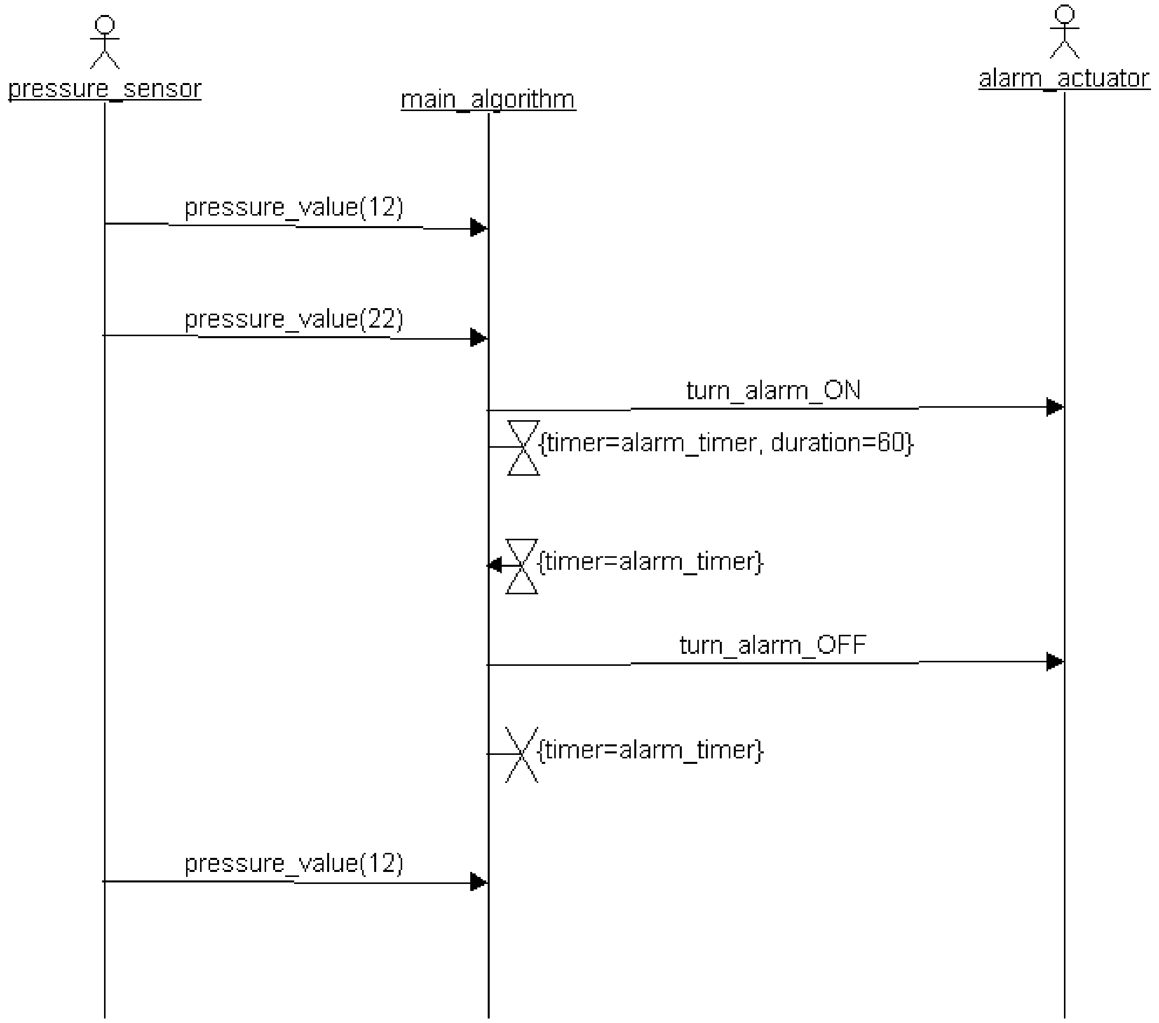
➢ Use Case Diagram



- Activity Diagram

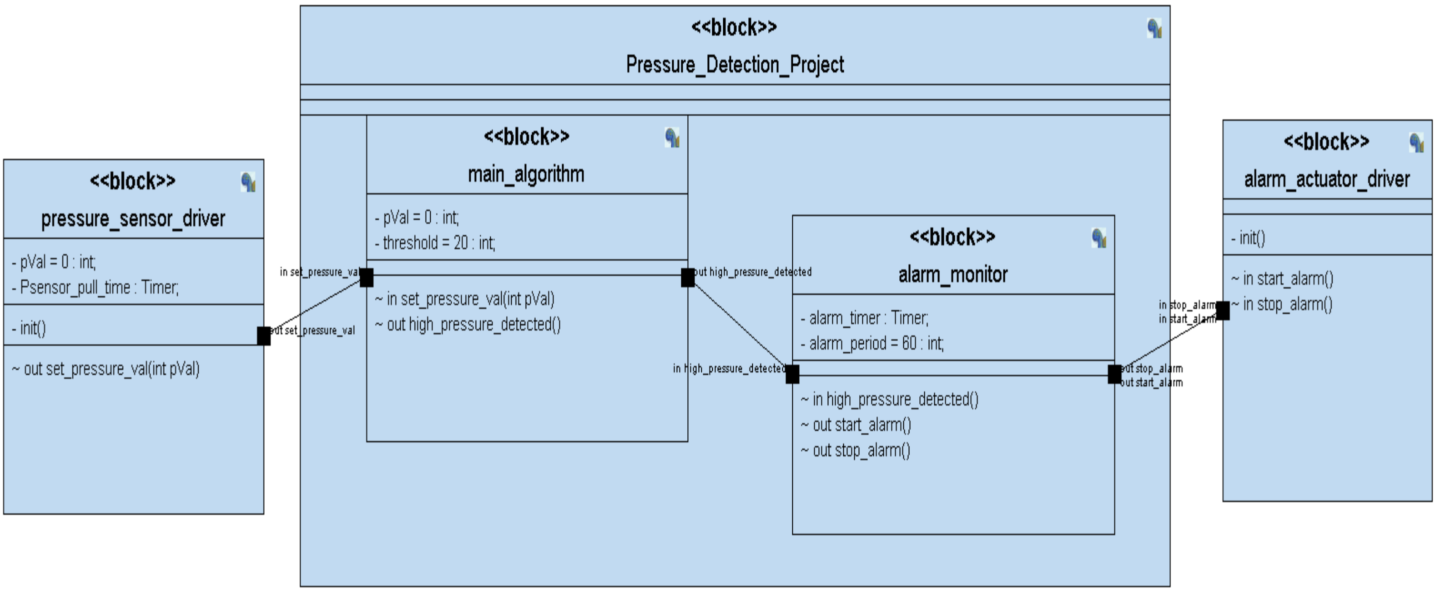


➢ Sequence Diagram



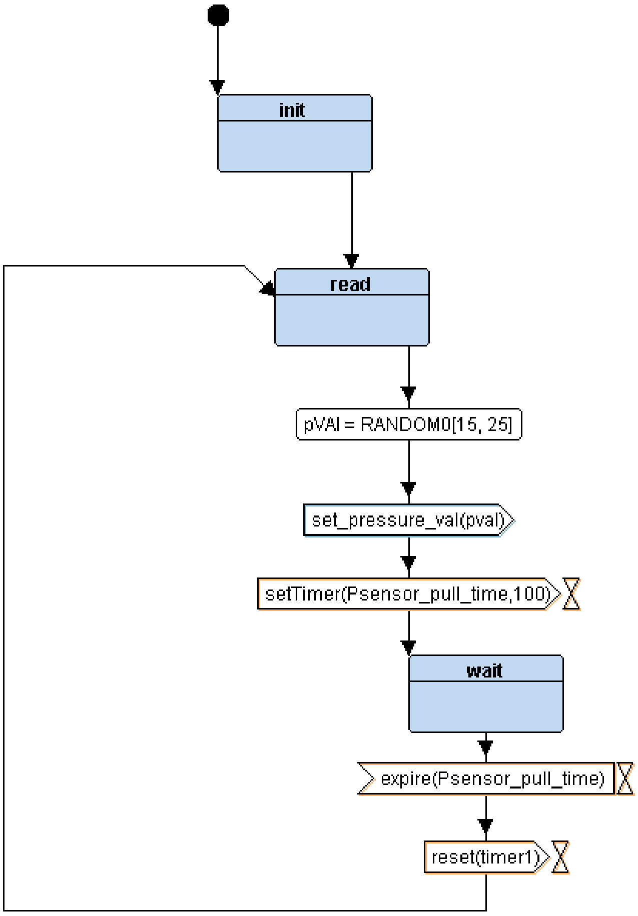
6. System Design

➢ Block Diagrams

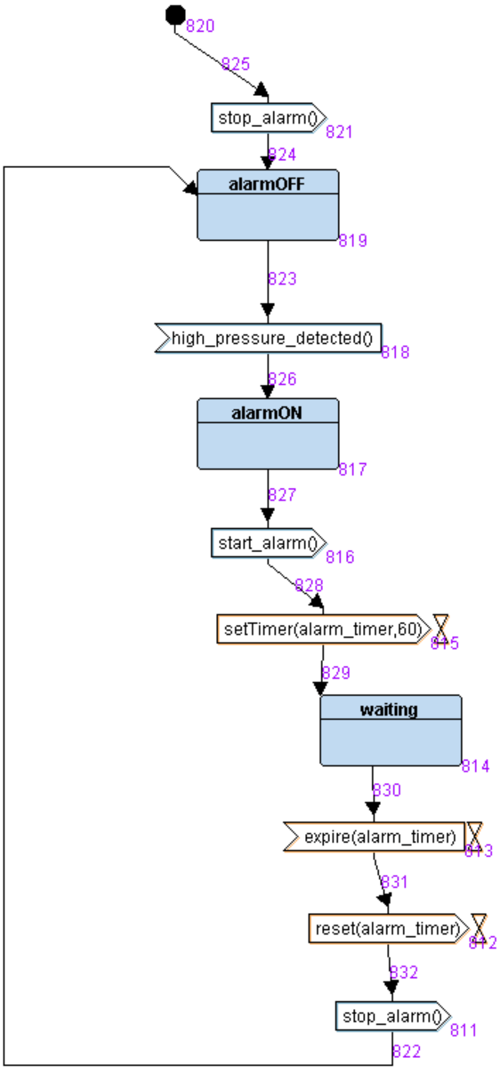


➢ State Machine Diagrams

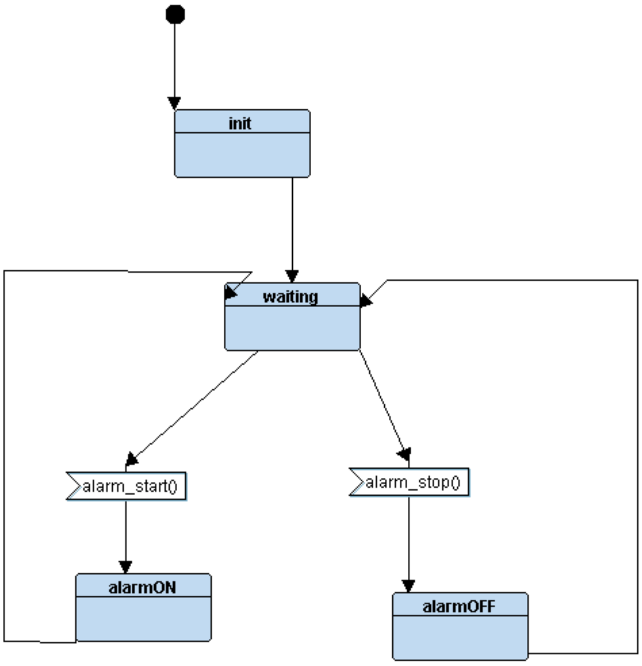
Pressure Sensor



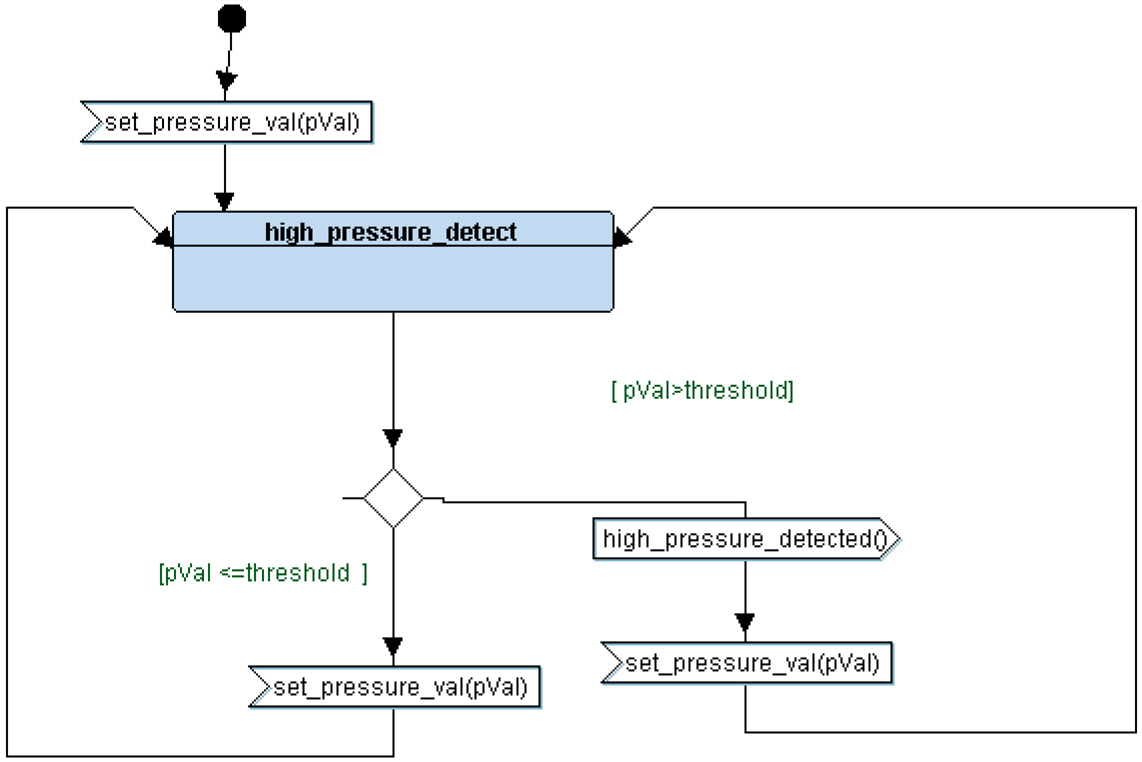
Alarm Monitor



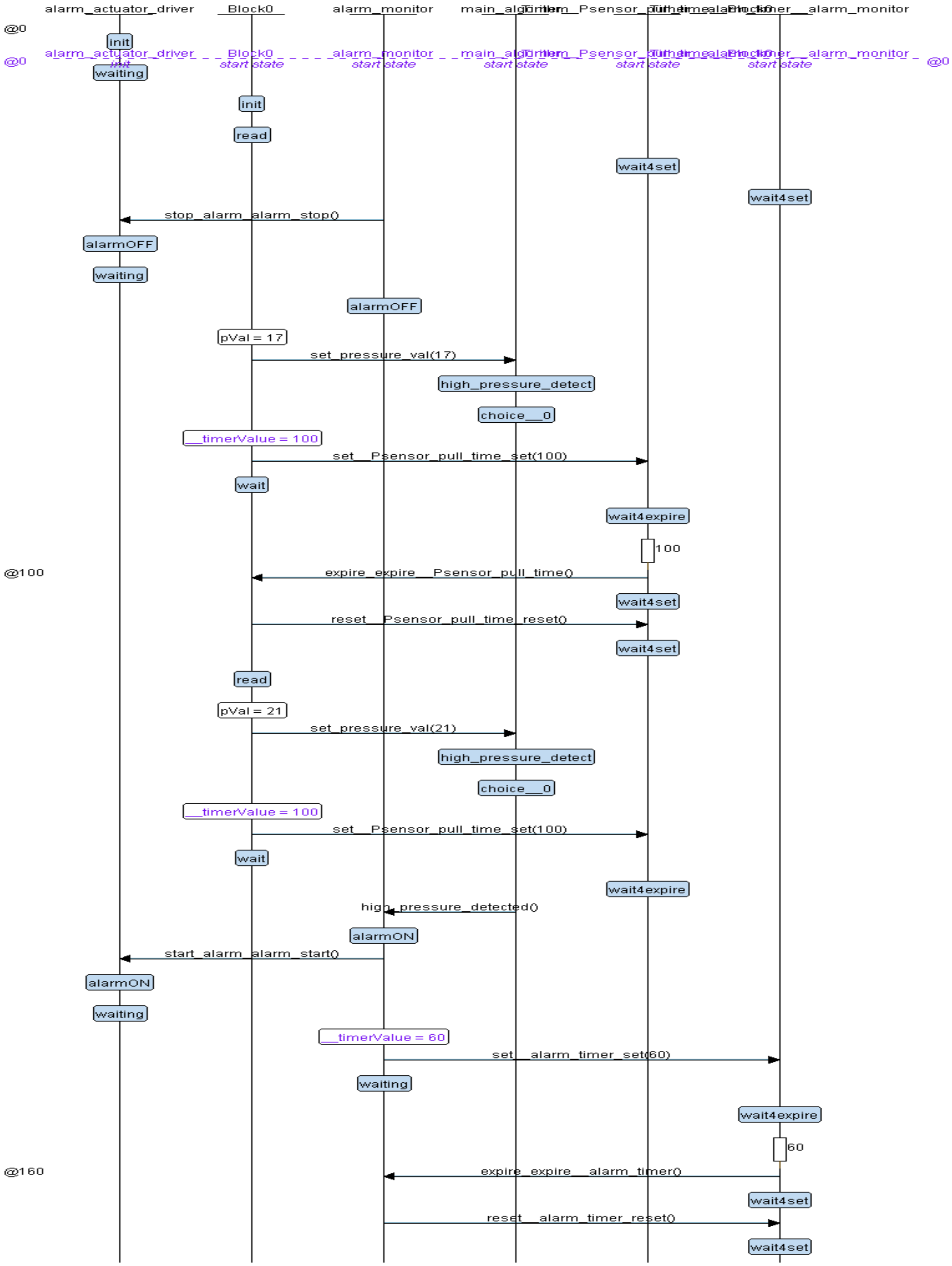
Alarm Actuator



Algorithm


set_pressure_val(pVal)
high_pressure_detect
[pVal>threshold]
[pVal <=threshold]
high_pressure_detected()
set_pressure_val(pVal)
set_pressure_val(pVal)

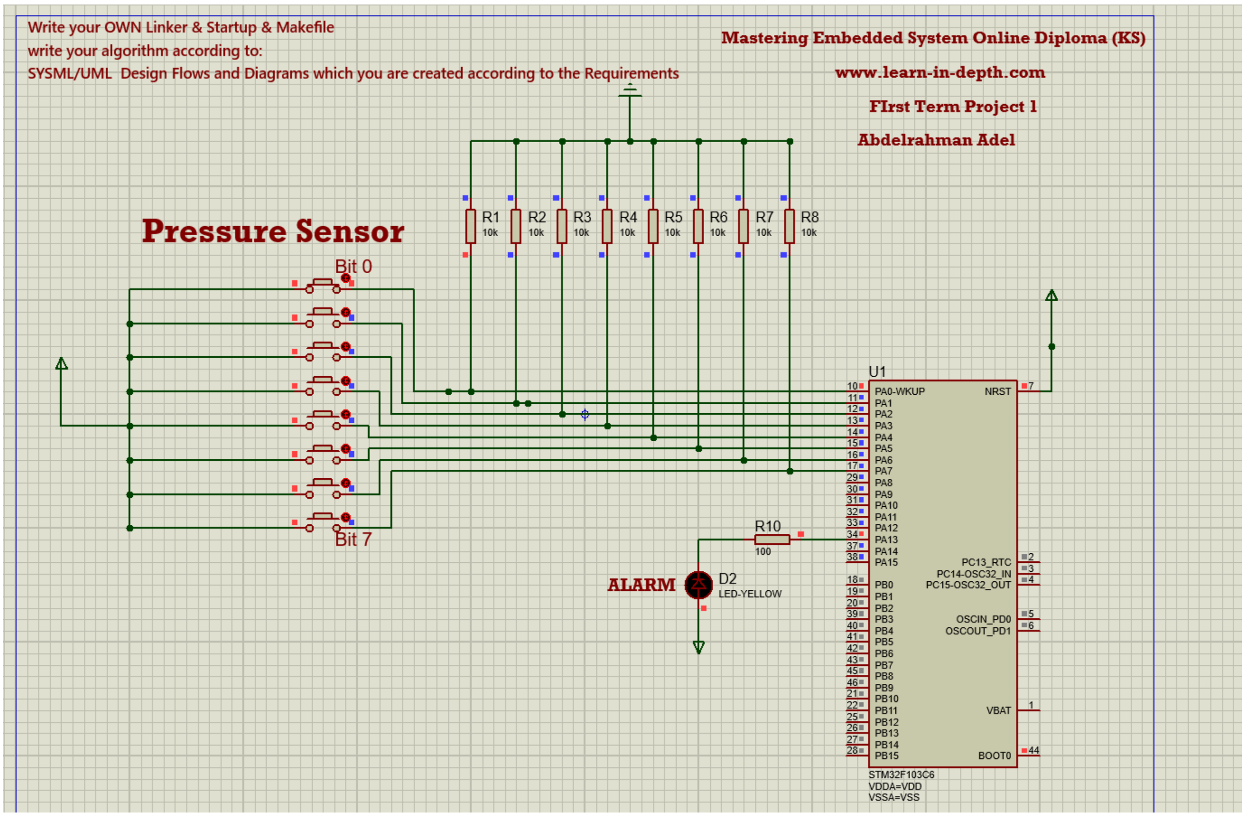
Traces



alarm_actuator_driver
Block0
alarm_monitor
main_algorithm
Timer_Psensor_pull_time
Timer_alarm_timer__alarm_monitor
@0
init
alarm_actuator_driver
init
Block0
start state
alarm_monitor
start state
main_algorithm
start state
Timer_Psensor_pull_time
start state
Timer_alarm_timer__alarm_monitor
start state
@0
@0
waiting
init
read
wait4set
wait4set
stop_alarm_alarm_stop()
alarmOFF
waiting
alarmOFF
pVal = 17
set_pressure_val(17)
high_pressure_detect
choice__0
__timerValue = 100
set__Psensor_pull_time_set(100)
wait
wait4expire
100
@100
expire_expire__Psensor_pull_time()
wait4set
reset__Psensor_pull_time_reset()
wait4set
read
pVal = 21
set_pressure_val(21)
high_pressure_detect
choice__0
__timerValue = 100
set__Psensor_pull_time_set(100)
wait
wait4expire
high_pressure_detected()
alarmON
start_alarm_alarm_start()
alarmON
waiting
__timerValue = 60
set__alarm_timer_set(60)
waiting
wait4expire
60
@160
expire_expire__alarm_timer()
wait4set
reset__alarm_timer_reset()
wait4set

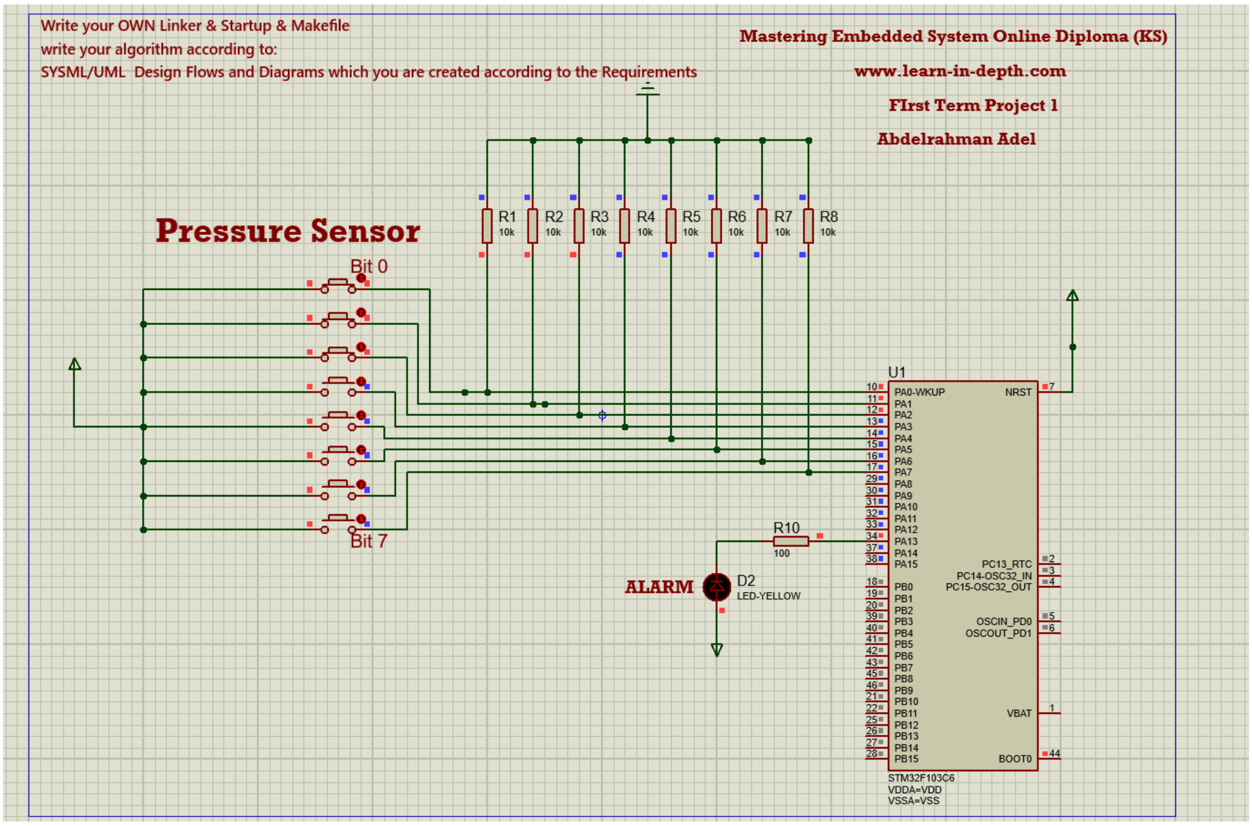
➢ Simulation

Pressure less than threshold



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

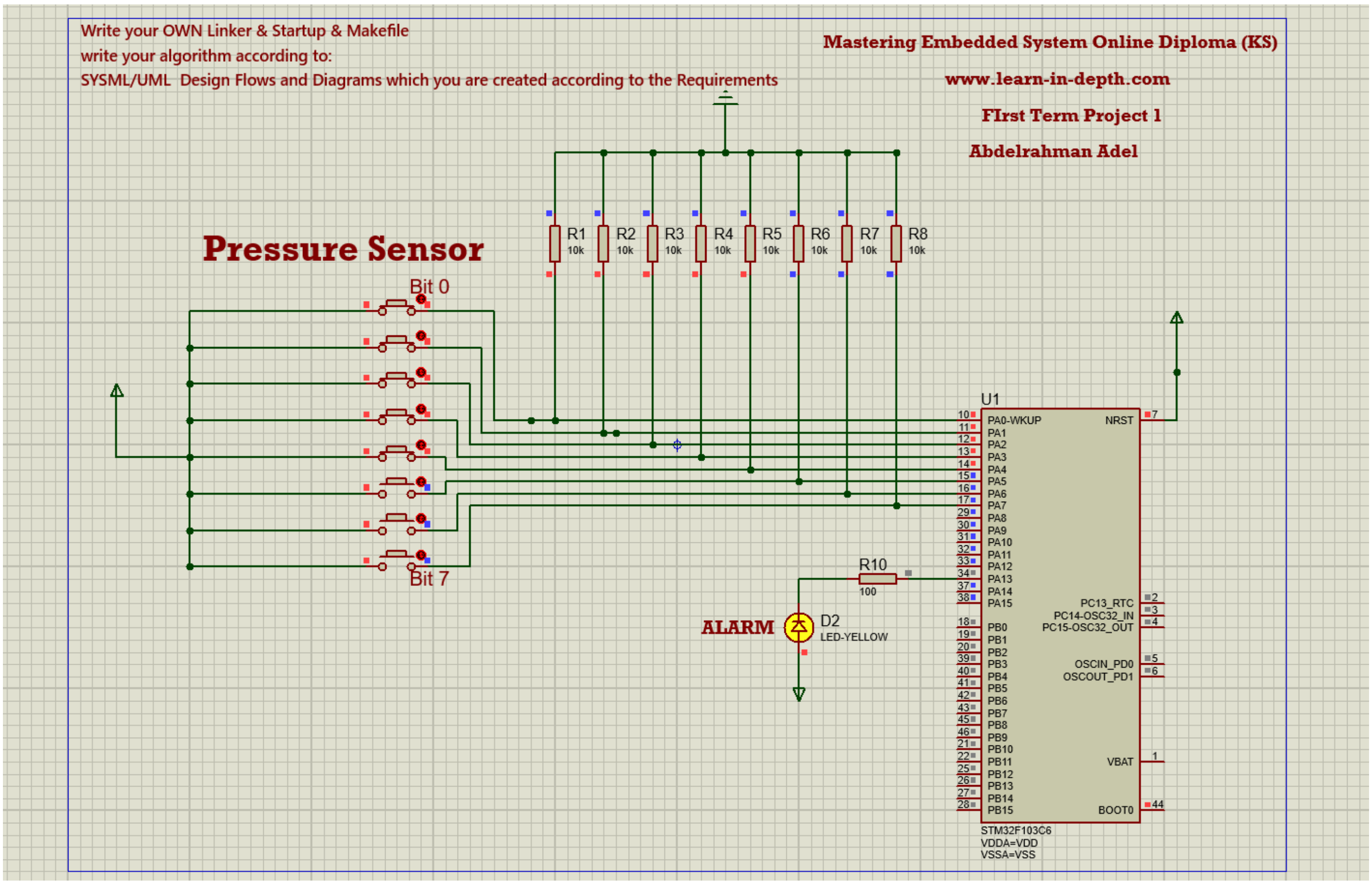
Mastering Embedded System Online Diploma (KS)
www.learn-in-depth.com
FIrst Term Project 1
Abdelrahman Adel



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

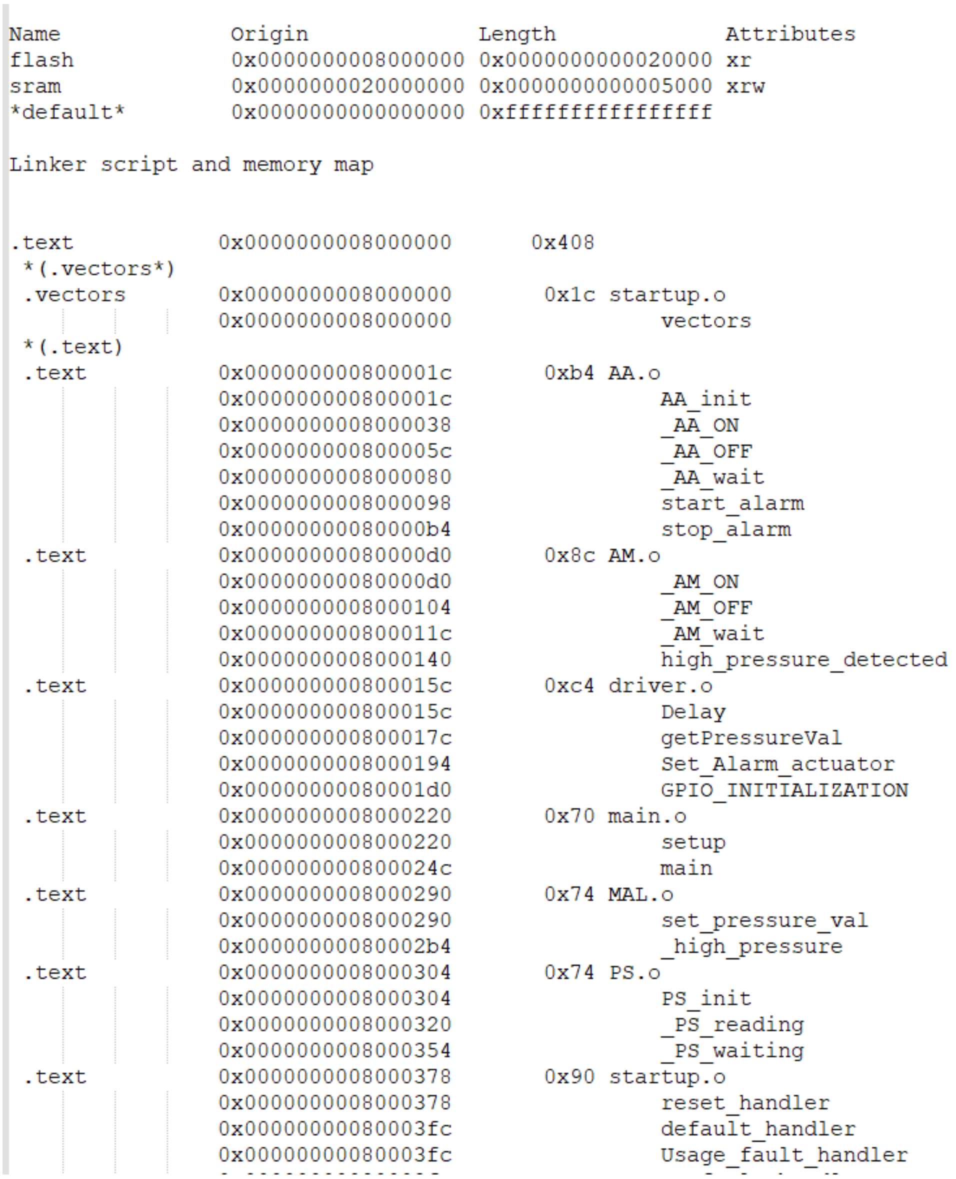
Mastering Embedded System Online Diploma (KS)
www.learn-in-depth.com
FIrst Term Project 1
Abdelrahman Adel

Pressure larger than threshold

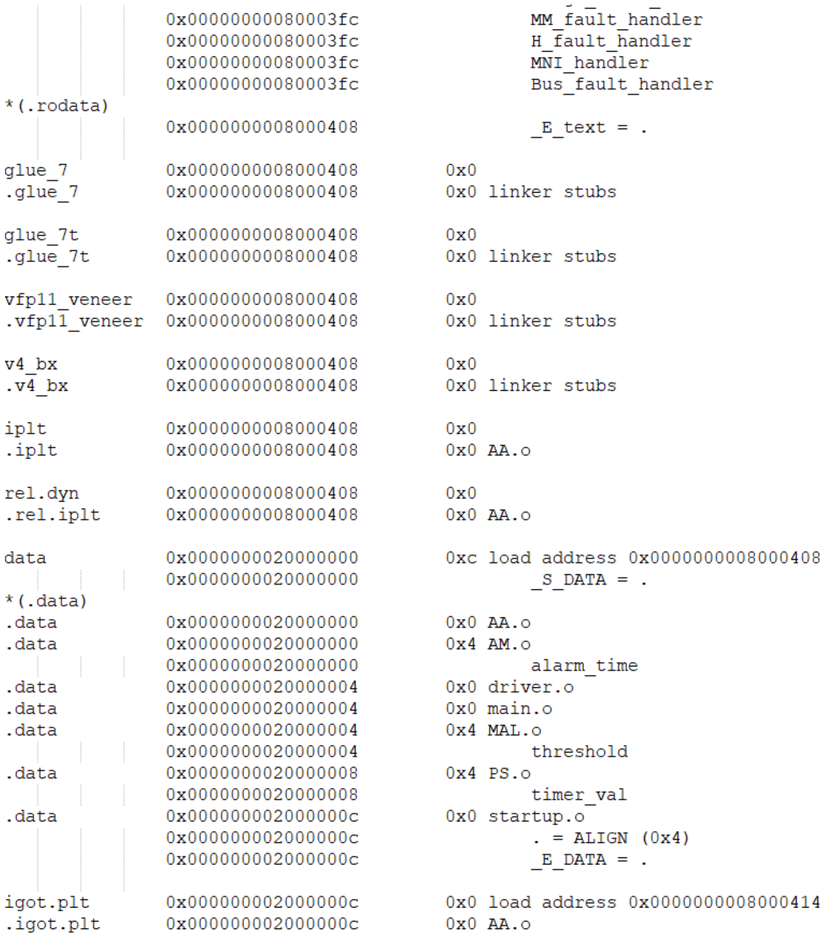


Software Analysis

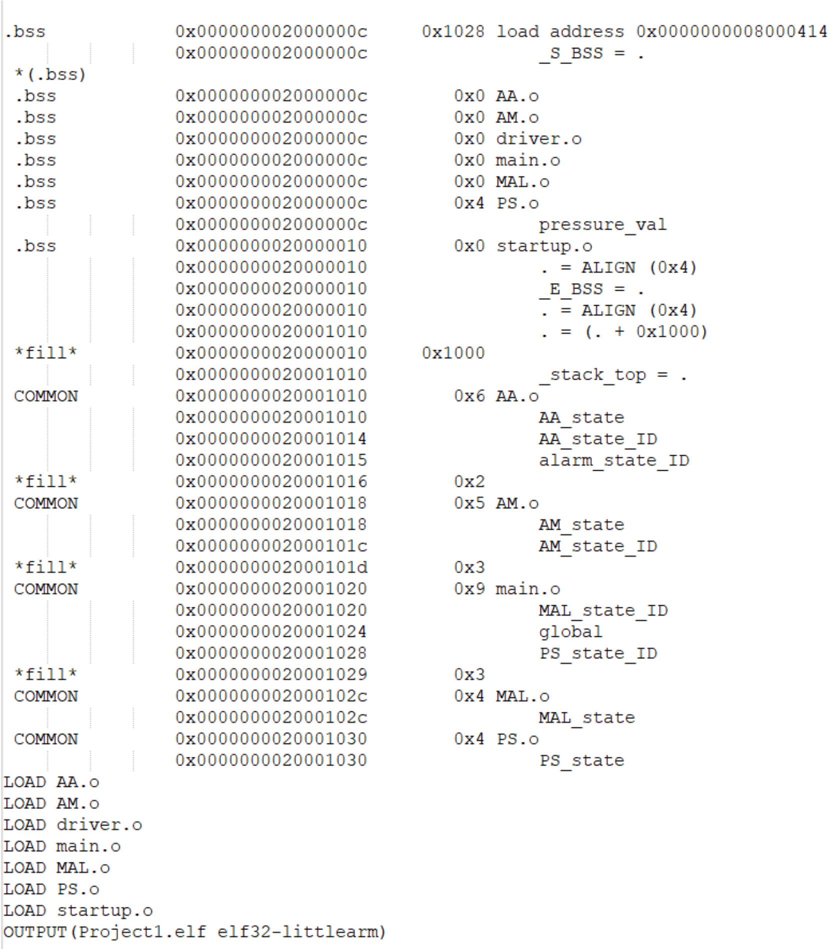
.map file



```
Name             Origin             Length             Attributes
flash            0x0000000008000000 0x0000000000020000 xr
sram             0x0000000020000000 0x0000000000005000 xrw
*default*        0x0000000000000000 0xffffffffffffffff

Linker script and memory map


.text           0x0000000008000000      0x408
 *(.vectors*)
 .vectors       0x0000000008000000       0x1c startup.o
                0x0000000008000000                vectors
 *(.text)
 .text          0x000000000800001c       0xb4 AA.o
                0x000000000800001c                AA_init
                0x0000000008000038                _AA_ON
                0x000000000800005c                _AA_OFF
                0x0000000008000080                _AA_wait
                0x0000000008000098                start_alarm
                0x00000000080000b4                stop_alarm
 .text          0x00000000080000d0       0x8c AM.o
                0x00000000080000d0                _AM_ON
                0x0000000008000104                _AM_OFF
                0x000000000800011c                _AM_wait
                0x0000000008000140                high_pressure_detected
 .text          0x000000000800015c       0xc4 driver.o
                0x000000000800015c                Delay
                0x000000000800017c                getPressureVal
                0x0000000008000194                Set_Alarm_actuator
                0x00000000080001d0                GPIO_INITIALIZATION
 .text          0x0000000008000220       0x70 main.o
                0x0000000008000220                setup
                0x000000000800024c                main
 .text          0x0000000008000290       0x74 MAL.o
                0x0000000008000290                set_pressure_val
                0x00000000080002b4                _high_pressure
 .text          0x0000000008000304       0x74 PS.o
                0x0000000008000304                PS_init
                0x0000000008000320                _PS_reading
                0x0000000008000354                _PS_waiting
 .text          0x0000000008000378       0x90 startup.o
                0x0000000008000378                reset_handler
                0x00000000080003fc                default_handler
                0x00000000080003fc                Usage_fault_handler
```

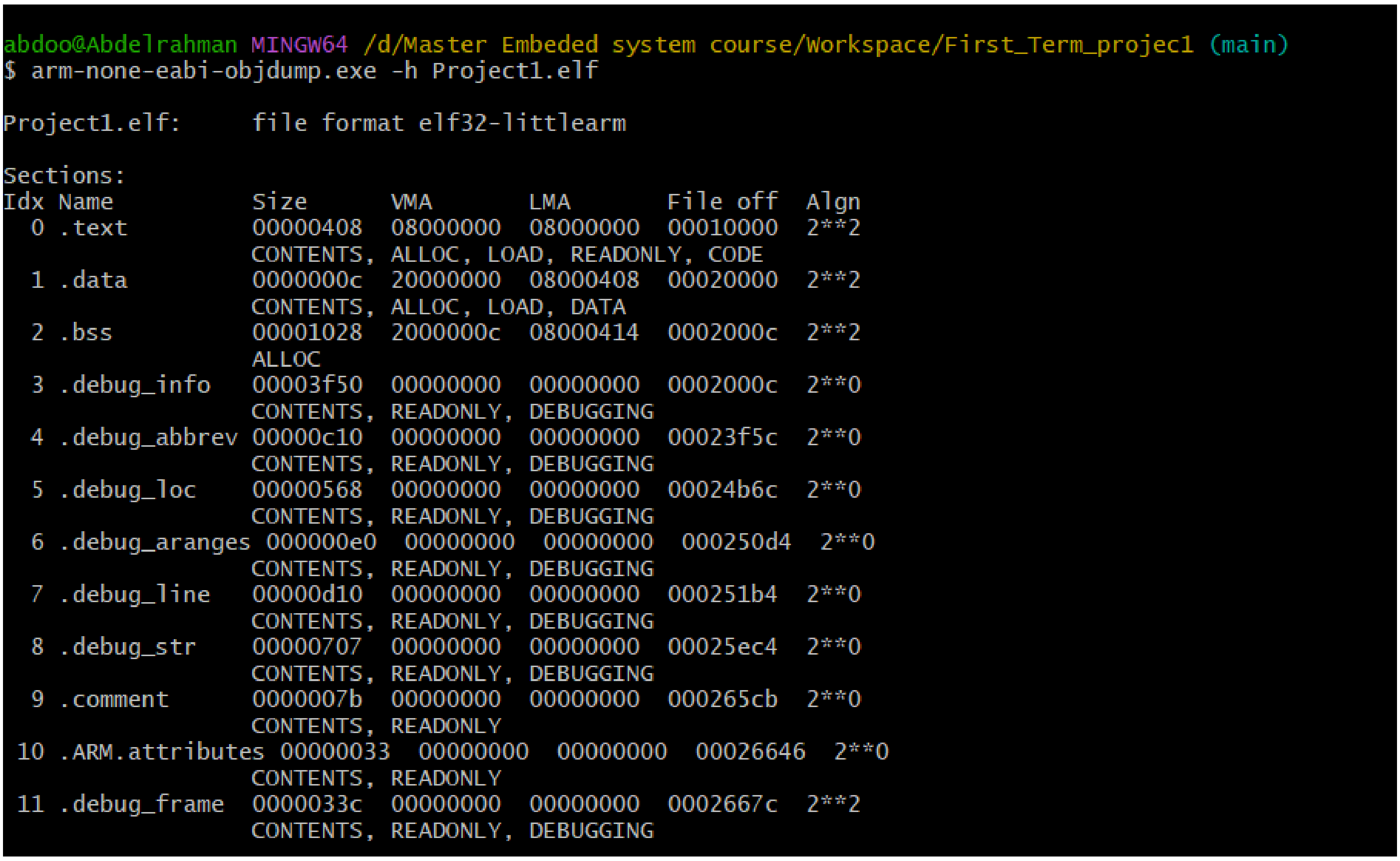


```
                0x00000000080003fc                MM_fault_handler
                0x00000000080003fc                H_fault_handler
                0x00000000080003fc                MNI_handler
                0x00000000080003fc                Bus_fault_handler
 *(.rodata)
                0x0000000008000408                _E_text = .

glue_7          0x0000000008000408        0x0
.glue_7         0x0000000008000408        0x0 linker stubs

glue_7t         0x0000000008000408        0x0
.glue_7t        0x0000000008000408        0x0 linker stubs

vfp11_veneer    0x0000000008000408        0x0
.vfp11_veneer   0x0000000008000408        0x0 linker stubs

v4_bx           0x0000000008000408        0x0
.v4_bx          0x0000000008000408        0x0 linker stubs

iplt            0x0000000008000408        0x0
.iplt           0x0000000008000408        0x0 AA.o

rel.dyn         0x0000000008000408        0x0
.rel.iplt       0x0000000008000408        0x0 AA.o

data            0x0000000020000000        0xc load address 0x0000000008000408
                0x0000000020000000                _S_DATA = .
 *(.data)
.data           0x0000000020000000        0x0 AA.o
.data           0x0000000020000000        0x4 AM.o
                0x0000000020000000                alarm_time
.data           0x0000000020000004        0x0 driver.o
.data           0x0000000020000004        0x0 main.o
.data           0x0000000020000004        0x4 MAL.o
                0x0000000020000004                threshold
.data           0x0000000020000008        0x4 PS.o
                0x0000000020000008                timer_val
.data           0x000000002000000c        0x0 startup.o
                0x000000002000000c                . = ALIGN (0x4)
                0x000000002000000c                _E_DATA = .

igot.plt        0x000000002000000c        0x0 load address 0x0000000008000414
.igot.plt       0x000000002000000c        0x0 AA.o
```



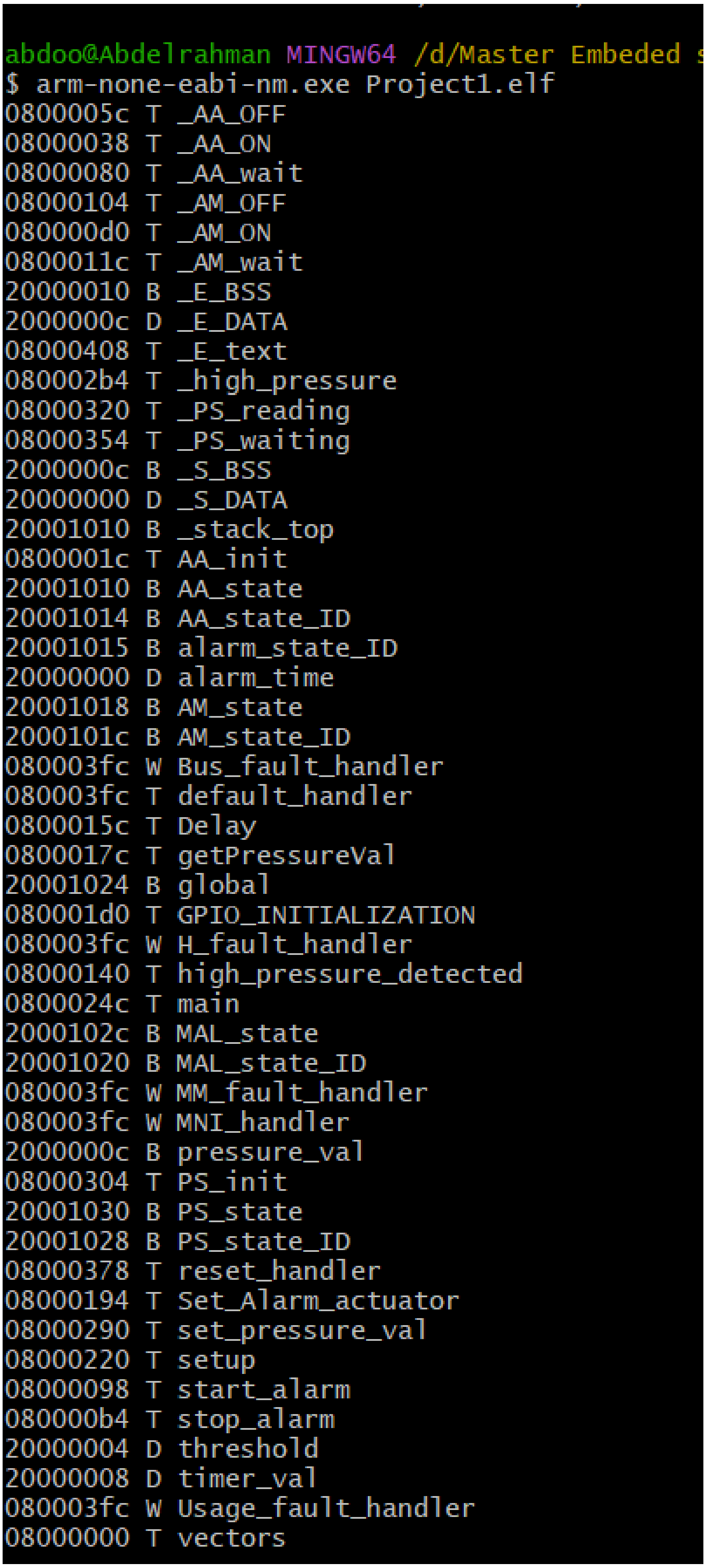
```
.bss            0x000000002000000c     0x1028 load address 0x0000000008000414
                0x000000002000000c                _S_BSS = .
 *(.bss)
 .bss           0x000000002000000c        0x0 AA.o
 .bss           0x000000002000000c        0x0 AM.o
 .bss           0x000000002000000c        0x0 driver.o
 .bss           0x000000002000000c        0x0 main.o
 .bss           0x000000002000000c        0x0 MAL.o
 .bss           0x000000002000000c        0x4 PS.o
                0x000000002000000c                pressure_val
 .bss           0x0000000020000010        0x0 startup.o
                0x0000000020000010                . = ALIGN (0x4)
                0x0000000020000010                _E_BSS = .
                0x0000000020000010                . = ALIGN (0x4)
                0x0000000020001010                . = (. + 0x1000)
 *fill*         0x0000000020000010     0x1000
                0x0000000020001010                _stack_top = .
 COMMON         0x0000000020001010        0x6 AA.o
                0x0000000020001010                AA_state
                0x0000000020001014                AA_state_ID
                0x0000000020001015                alarm_state_ID
 *fill*         0x0000000020001016        0x2
 COMMON         0x0000000020001018        0x5 AM.o
                0x0000000020001018                AM_state
                0x000000002000101c                AM_state_ID
 *fill*         0x000000002000101d        0x3
 COMMON         0x0000000020001020        0x9 main.o
                0x0000000020001020                MAL_state_ID
                0x0000000020001024                global
                0x0000000020001028                PS_state_ID
 *fill*         0x0000000020001029        0x3
 COMMON         0x000000002000102c        0x4 MAL.o
                0x000000002000102c                MAL_state
 COMMON         0x0000000020001030        0x4 PS.o
                0x0000000020001030                PS_state
LOAD AA.o
LOAD AM.o
LOAD driver.o
LOAD main.o
LOAD MAL.o
LOAD PS.o
LOAD startup.o
OUTPUT(Project1.elf elf32-littlearm)
```

Section Table



```
abdoo@Abdelrahman MINGW64 /d/Master Embeded system course/Workspace/First_Term_projec1 (main)
$ arm-none-eabi-objdump.exe -h Project1.elf

Project1.elf:     file format elf32-littlearm

Sections:
Idx Name          Size      VMA       LMA       File off  Algn
  0 .text         00000408  08000000  08000000  00010000  2**2
                  CONTENTS, ALLOC, LOAD, READONLY, CODE
  1 .data         0000000c  20000000  08000408  00020000  2**2
                  CONTENTS, ALLOC, LOAD, DATA
  2 .bss          00001028  2000000c  08000414  0002000c  2**2
                  ALLOC
  3 .debug_info   00003f50  00000000  00000000  0002000c  2**0
                  CONTENTS, READONLY, DEBUGGING
  4 .debug_abbrev 00000c10  00000000  00000000  00023f5c  2**0
                  CONTENTS, READONLY, DEBUGGING
  5 .debug_loc    00000568  00000000  00000000  00024b6c  2**0
                  CONTENTS, READONLY, DEBUGGING
  6 .debug_aranges 000000e0  00000000  00000000  000250d4  2**0
                  CONTENTS, READONLY, DEBUGGING
  7 .debug_line   00000d10  00000000  00000000  000251b4  2**0
                  CONTENTS, READONLY, DEBUGGING
  8 .debug_str    00000707  00000000  00000000  00025ec4  2**0
                  CONTENTS, READONLY, DEBUGGING
  9 .comment      0000007b  00000000  00000000  000265cb  2**0
                  CONTENTS, READONLY
 10 .ARM.attributes 00000033  00000000  00000000  00026646  2**0
                  CONTENTS, READONLY
 11 .debug_frame  0000033c  00000000  00000000  0002667c  2**2
                  CONTENTS, READONLY, DEBUGGING
```

Symbol Table



```
abdoo@Abdelrahman MINGW64 /d/Master Embeded s
$ arm-none-eabi-nm.exe Project1.elf
0800005c T _AA_OFF
08000038 T _AA_ON
08000080 T _AA_wait
08000104 T _AM_OFF
080000d0 T _AM_ON
0800011c T _AM_wait
20000010 B _E_BSS
2000000c D _E_DATA
08000408 T _E_text
080002b4 T _high_pressure
08000320 T _PS_reading
08000354 T _PS_waiting
2000000c B _S_BSS
20000000 D _S_DATA
20001010 B _stack_top
0800001c T AA_init
20001010 B AA_state
20001014 B AA_state_ID
20001015 B alarm_state_ID
20000000 D alarm_time
20001018 B AM_state
2000101c B AM_state_ID
080003fc W Bus_fault_handler
080003fc T default_handler
0800015c T Delay
0800017c T getPressureVal
20001024 B global
080001d0 T GPIO_INITIALIZATION
080003fc W H_fault_handler
08000140 T high_pressure_detected
0800024c T main
2000102c B MAL_state
20001020 B MAL_state_ID
080003fc W MM_fault_handler
080003fc W MNI_handler
2000000c B pressure_val
08000304 T PS_init
20001030 B PS_state
20001028 B PS_state_ID
08000378 T reset_handler
08000194 T Set_Alarm_actuator
08000290 T set_pressure_val
08000220 T setup
08000098 T start_alarm
080000b4 T stop_alarm
20000004 D threshold
20000008 D timer_val
080003fc W Usage_fault_handler
08000000 T vectors
```