



Learn-In-Depth

Mastering Embedded System
Online Diploma

Pressure Controller PROJECT

Presented by: Mohammed Hasan



For My Profile: [Click Here](#)



System Architecting & Design Sequence

1

Case Study

3

Requirement

5

System Analysis

2

Method

4

Space Exploration

6

System Design

System Architecting && Design Sequence

7

System Design Simulation

8

Source Code

9

Sections & Symbols for Object files

10

Symbol Table For Object file

11

Size Sections

12

Logic Verification

13

Simulation Output



Case Study

- A Pressure Controller informs the crew of cabin with an alarm when the pressure exceeds 20 Bars in the cabin .
- The Alarm duration equals 60 Seconds.
- Keep track of the measured values.





1 Assumptions

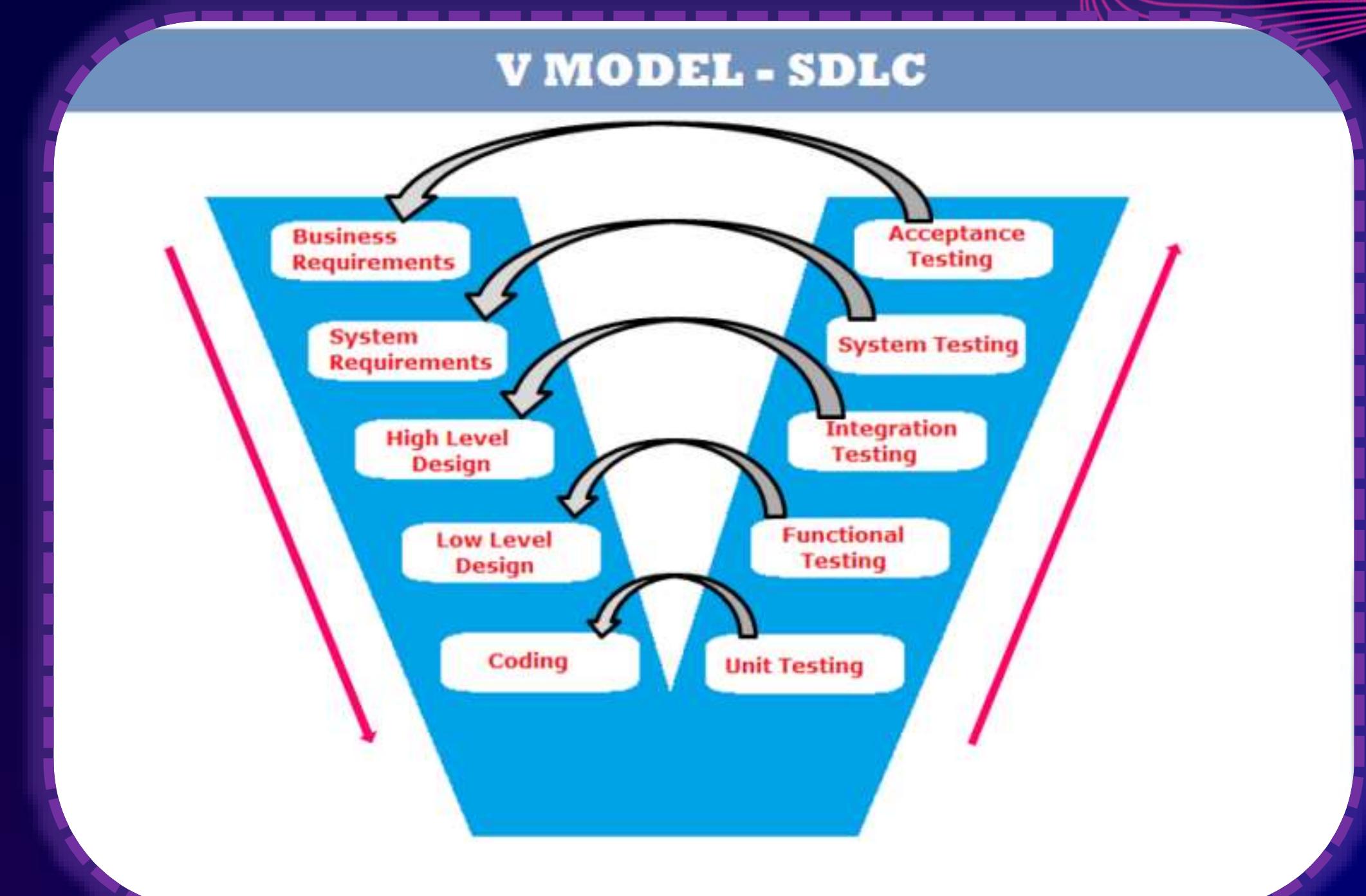
1	The System setup and shutdown procedures are not modeled .
2	The System maintenance is not modeled .
3	The Pressure Sensor never fails .
4	The Alarm never fails .
5	The System never faces power cut .



2

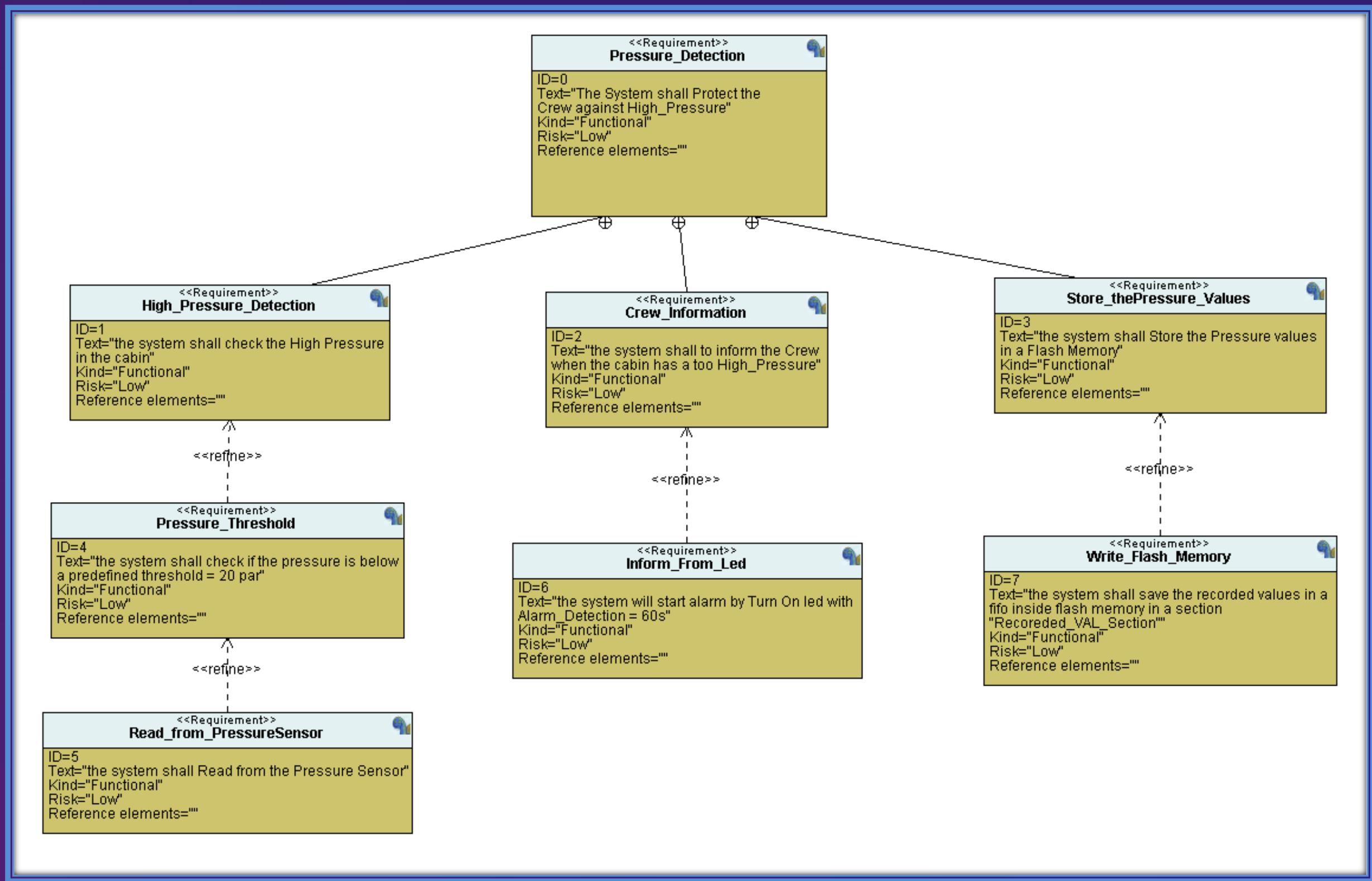
Method

- We will use a **testing-based model** Like V-model



3

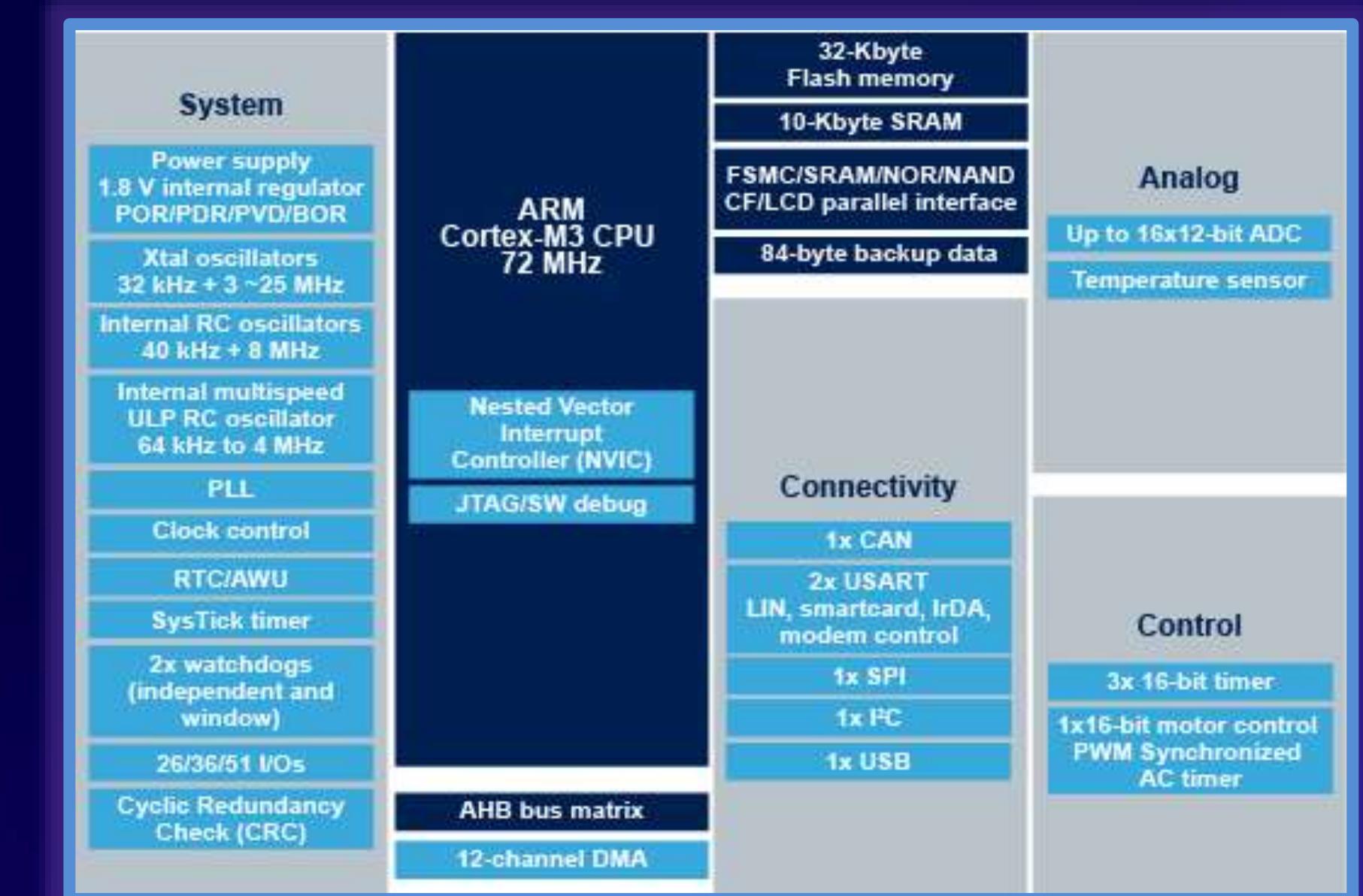
Requirement



4

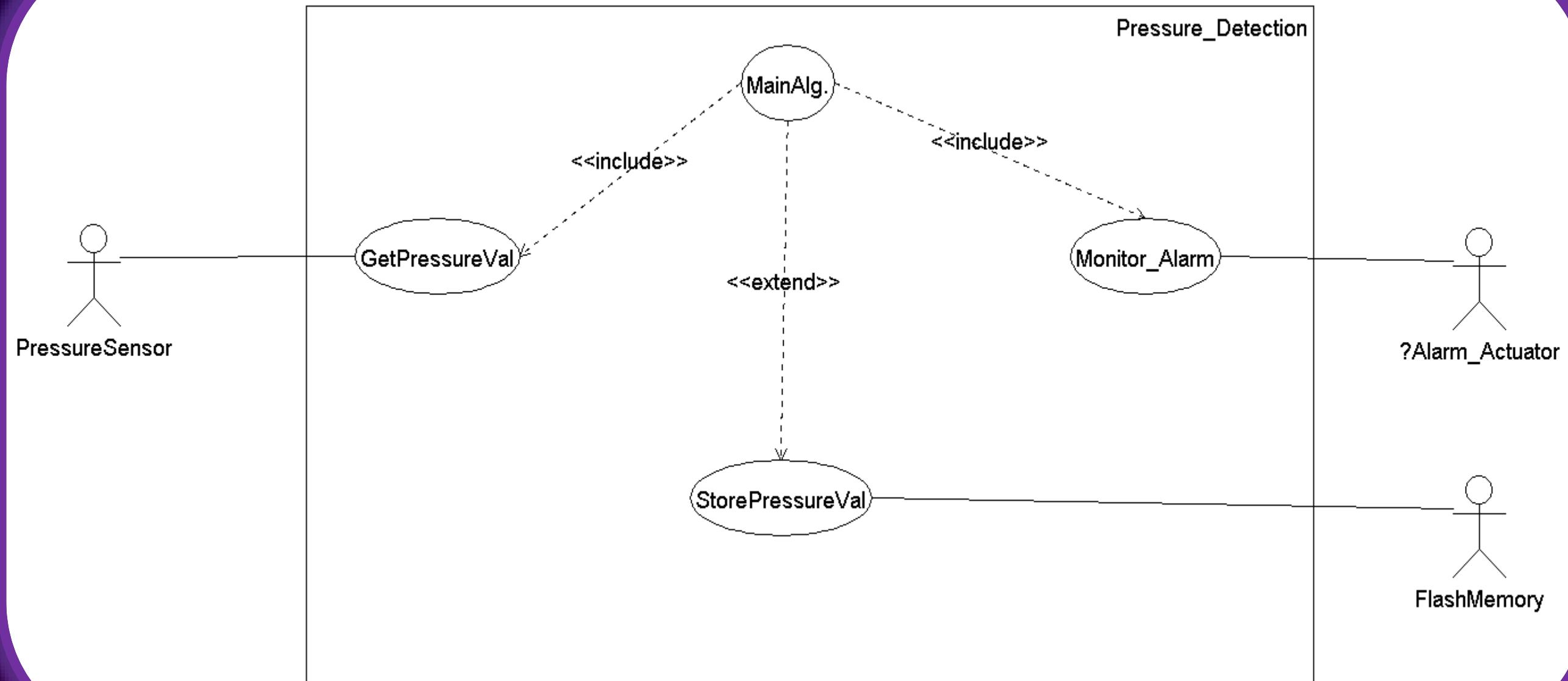
Space Exploration

- We have STM32 microcontroller with a Cortex-m3 Processor that will be more than enough for this application .





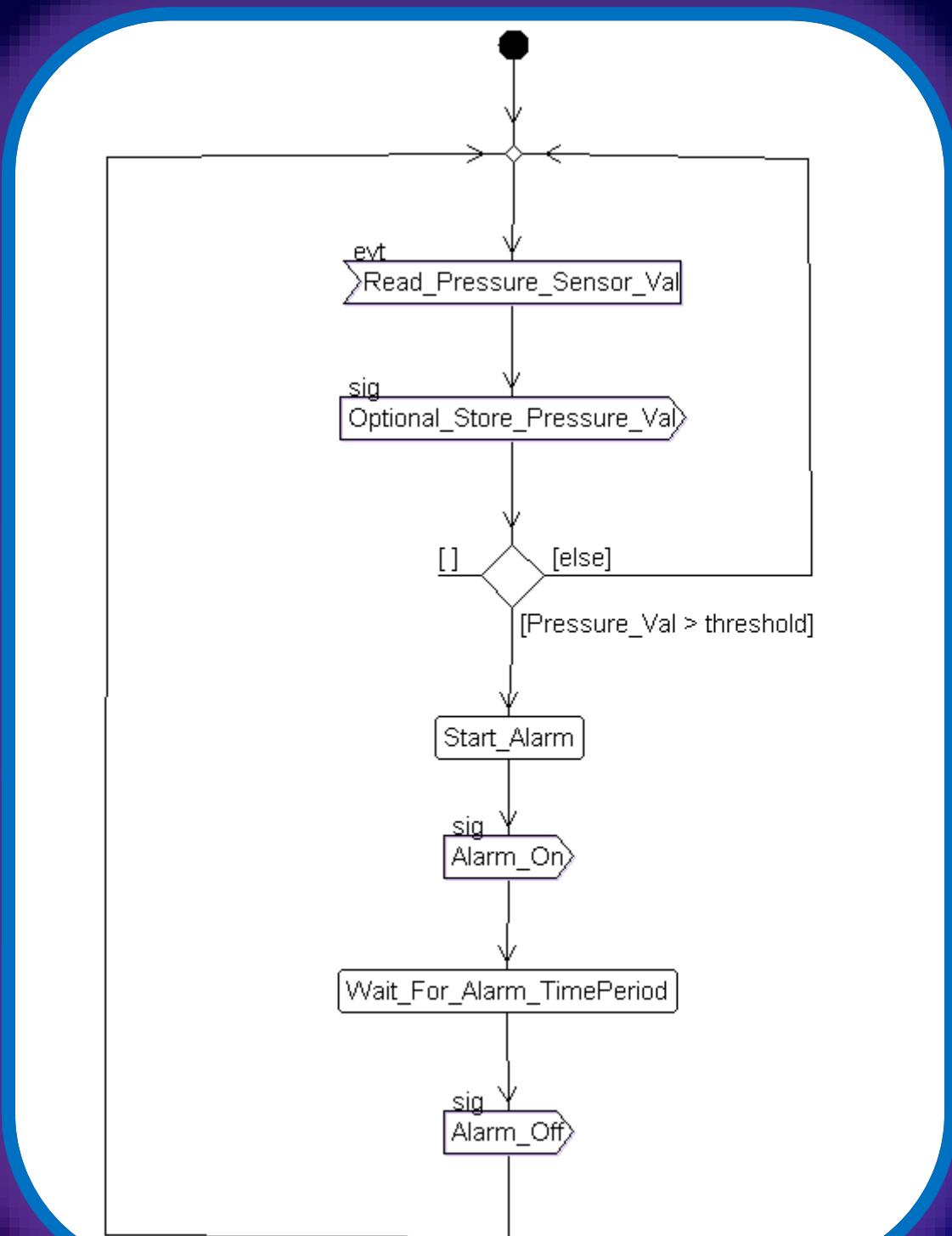
System Analysis : Use Case Diagram





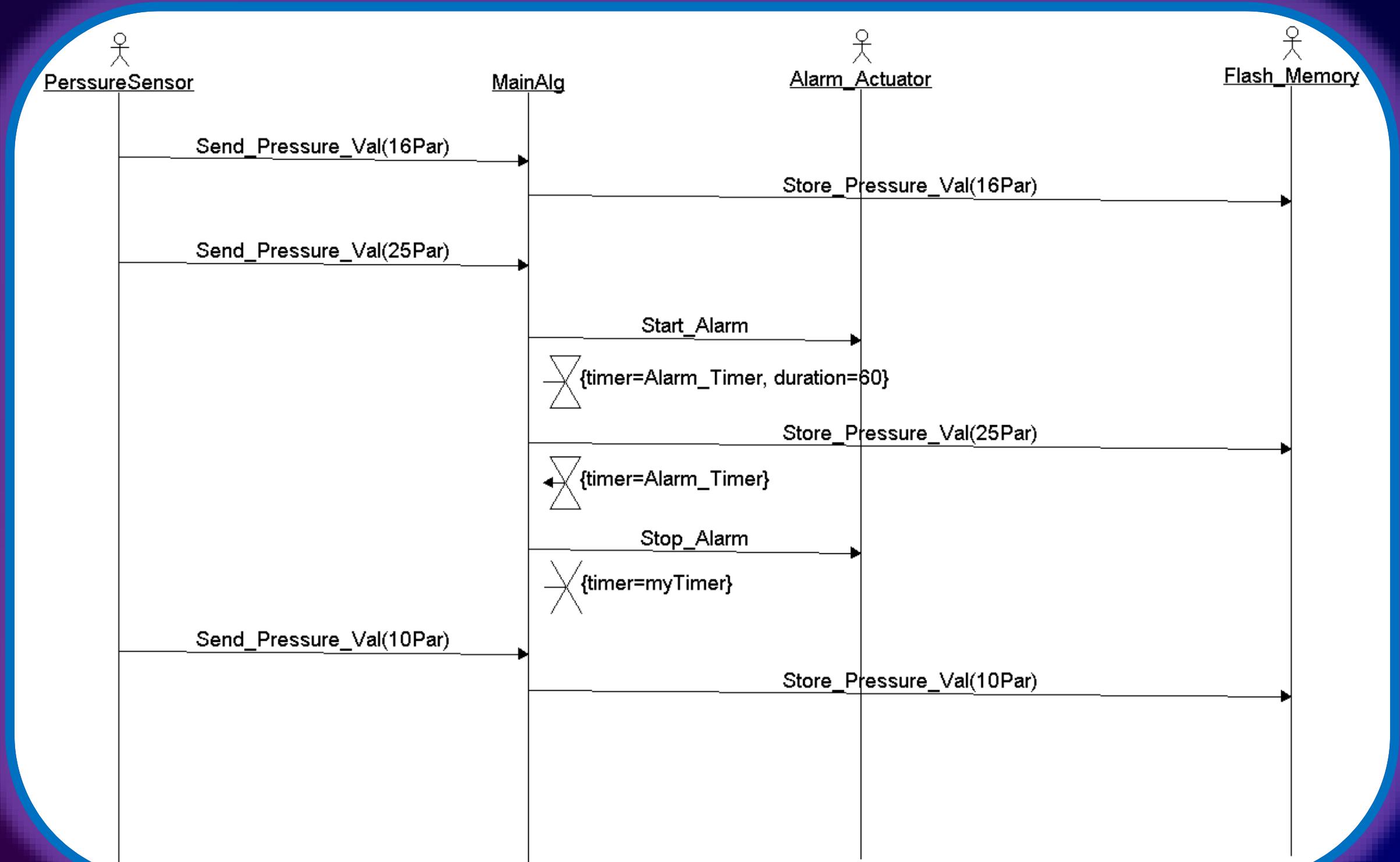
5

System Analysis : Activity Diagram



5

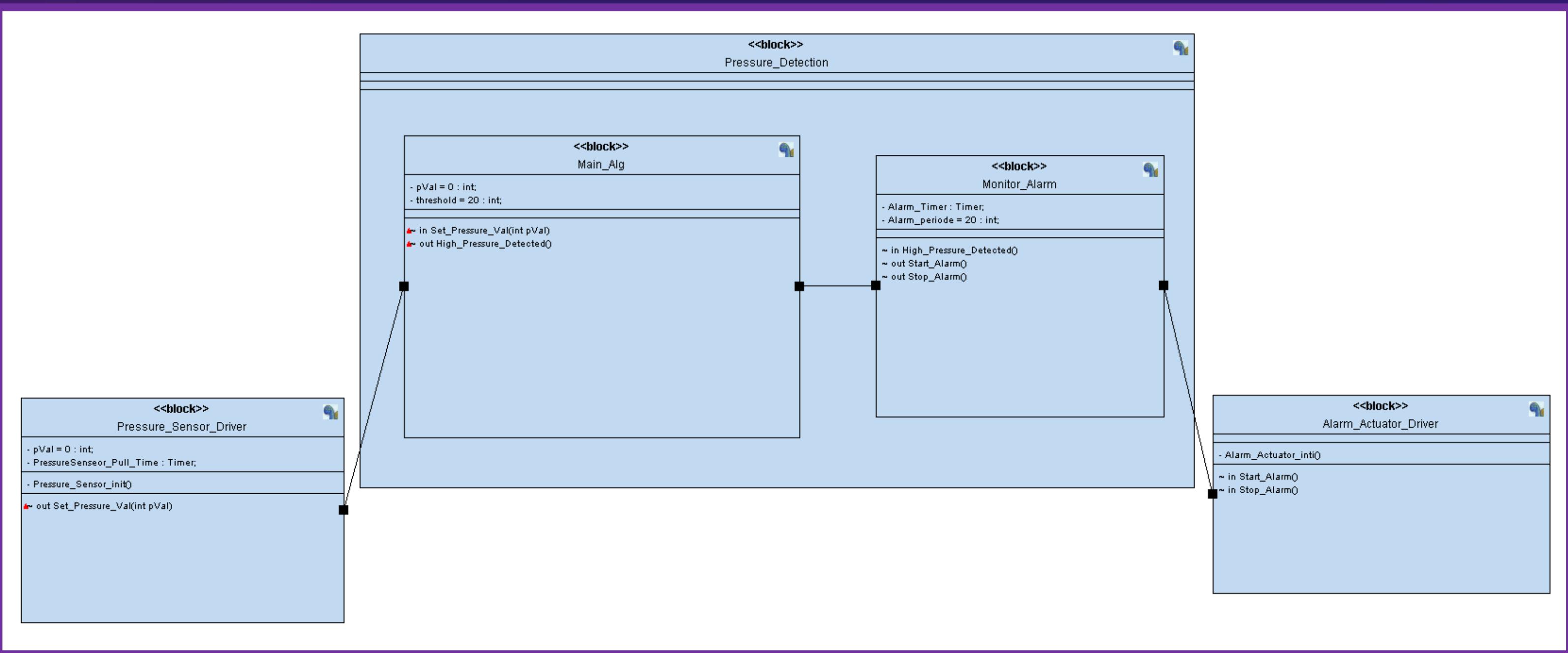
System Analysis : Sequence Diagram





6

System Design

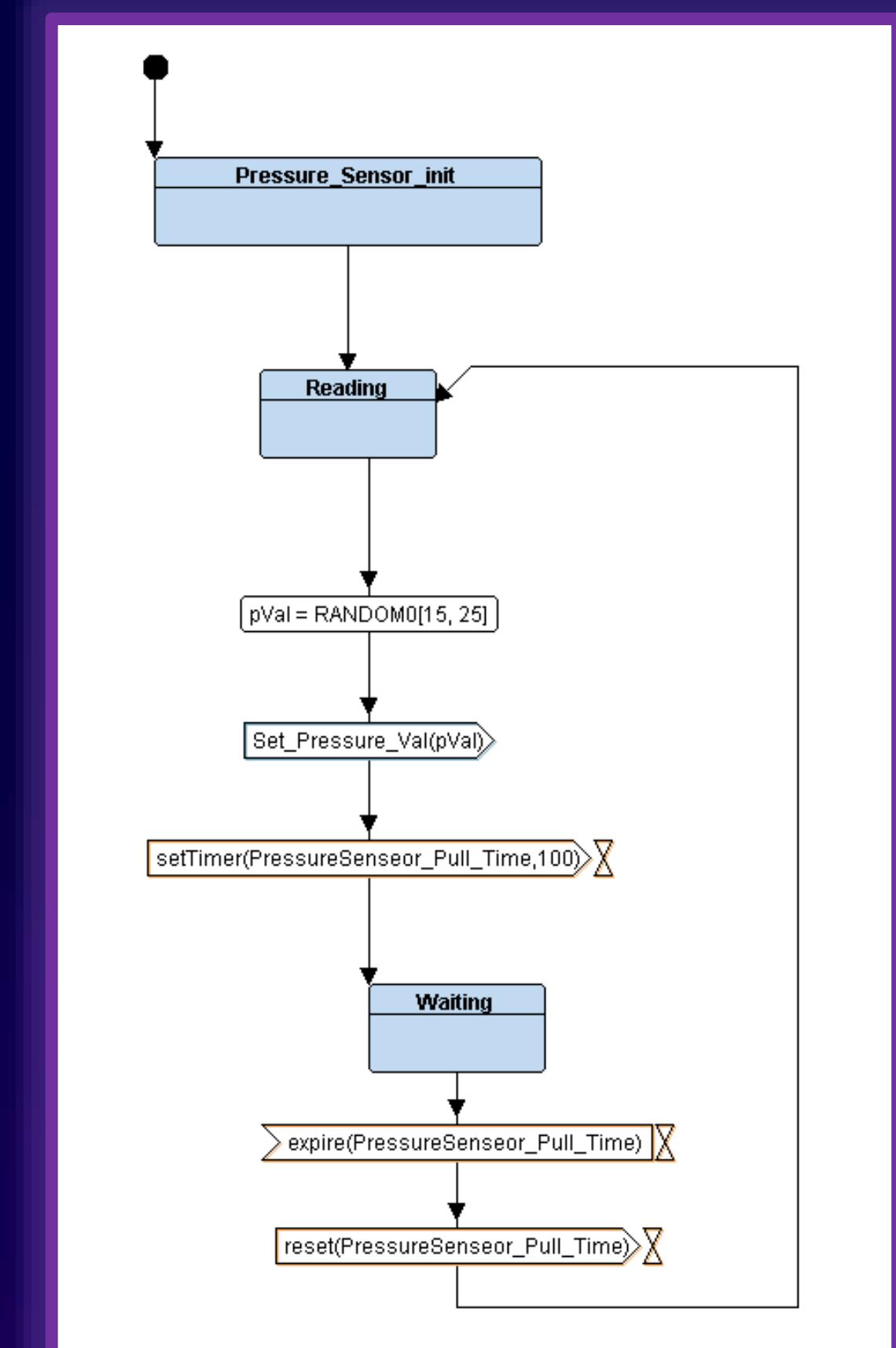




System Design



Pressure Sensor

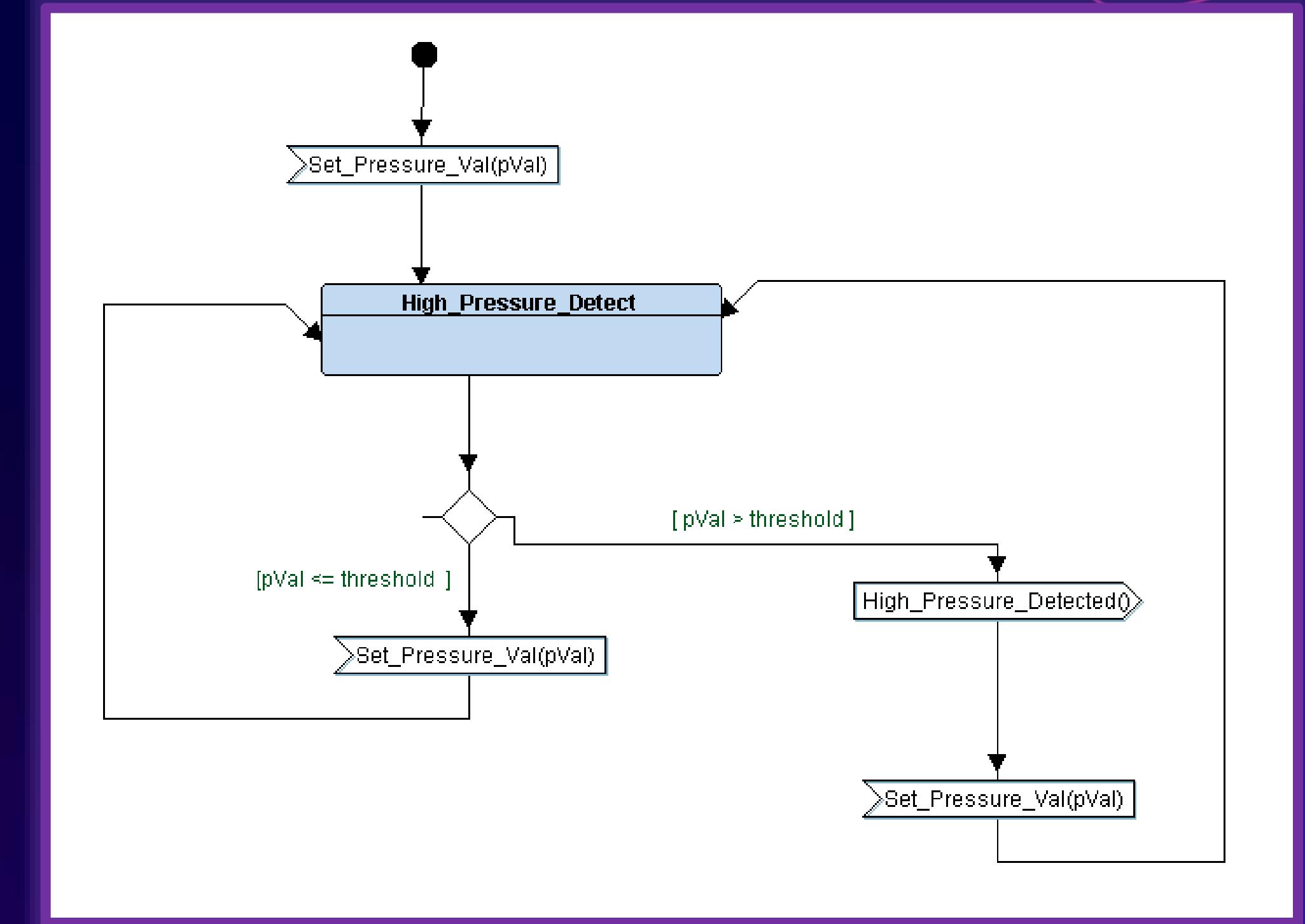




System Design



Main Algorithm

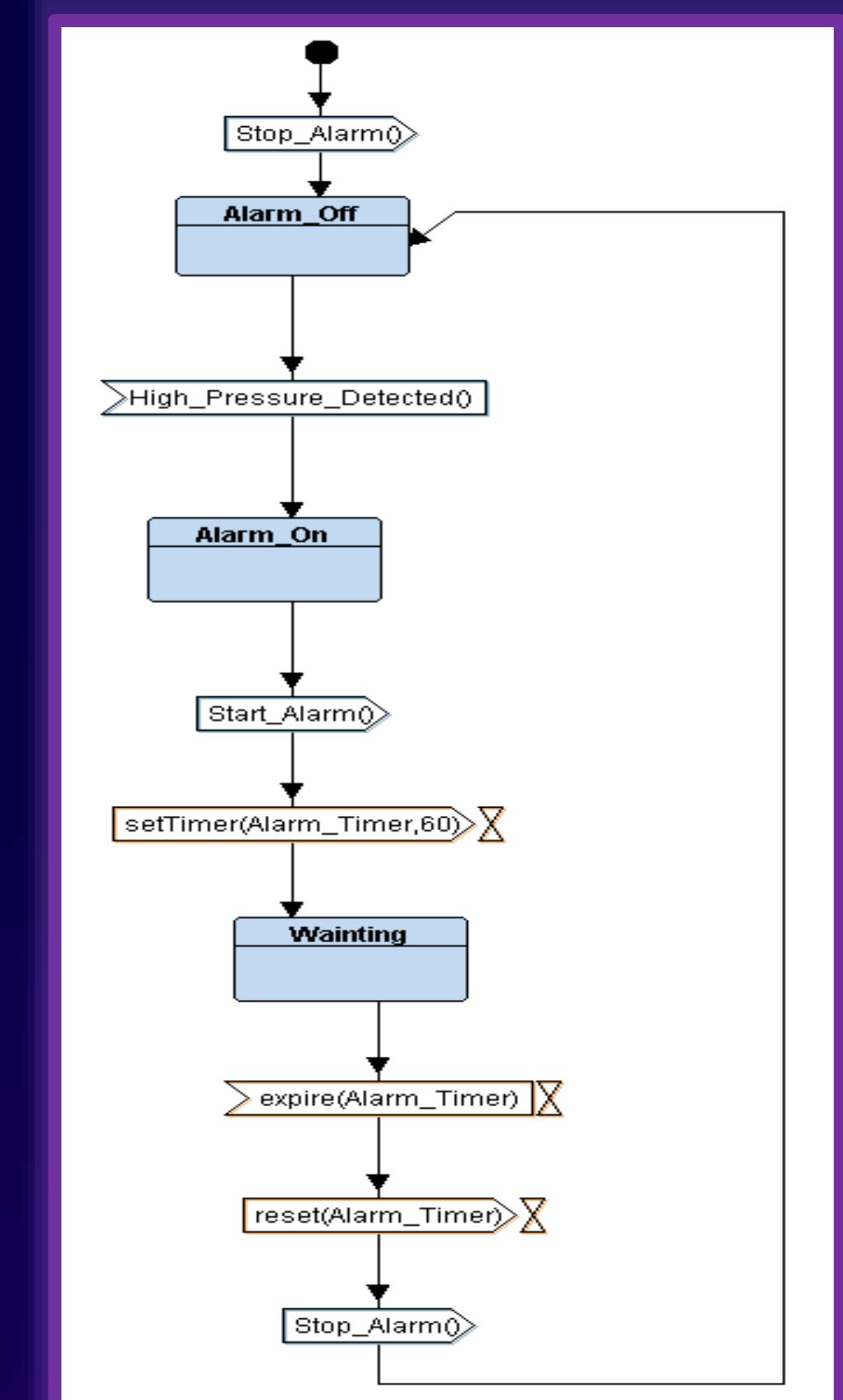




System Design



Monitor Alarm



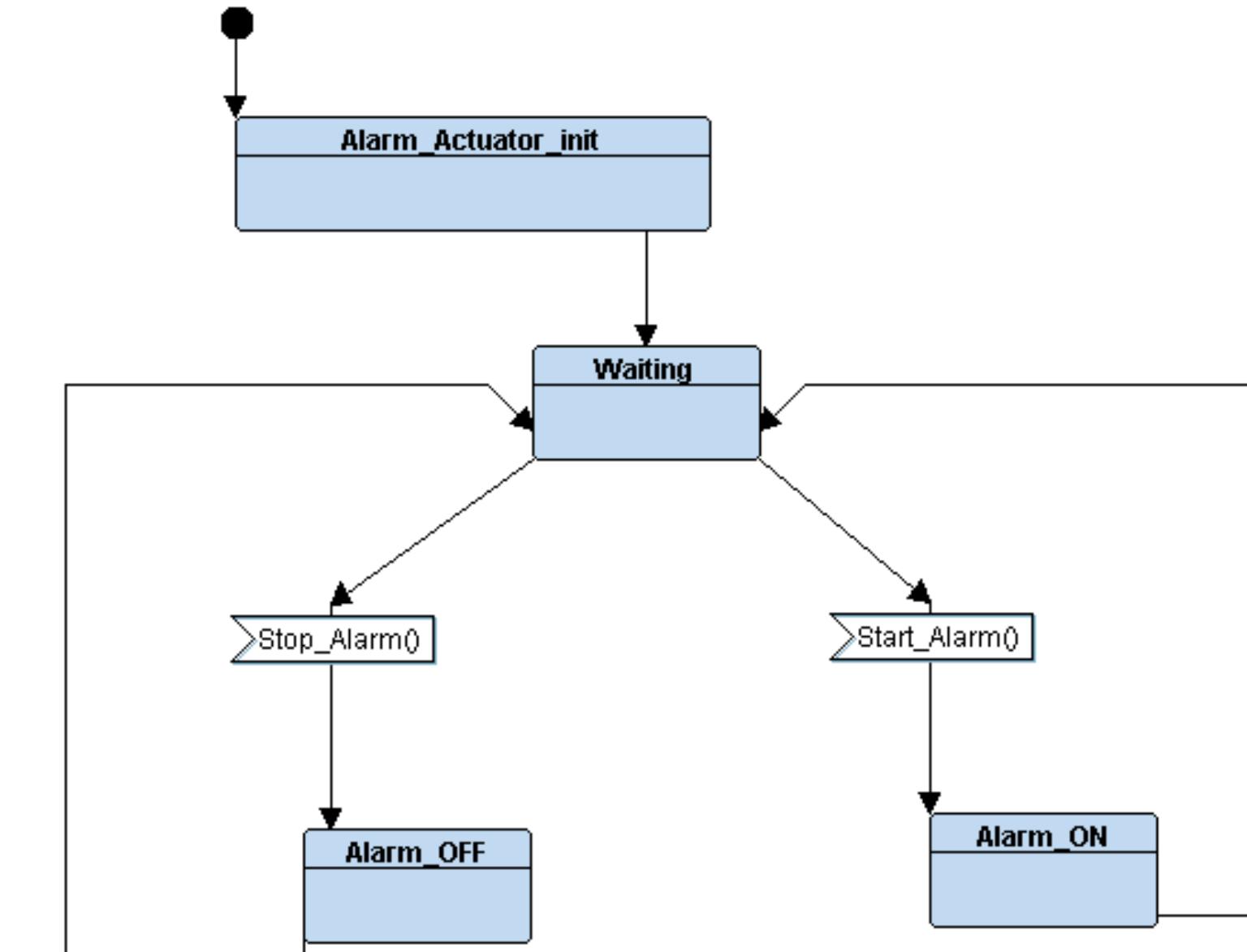


6

System Design

4

Alarm Actuator



System Design Simulation

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

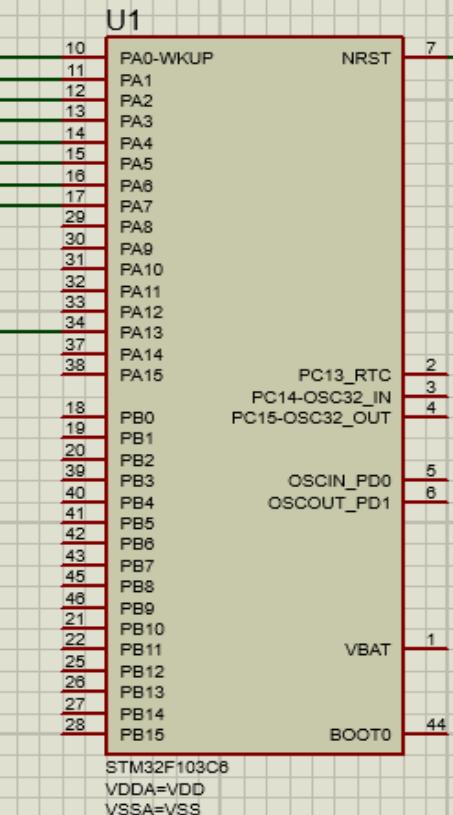
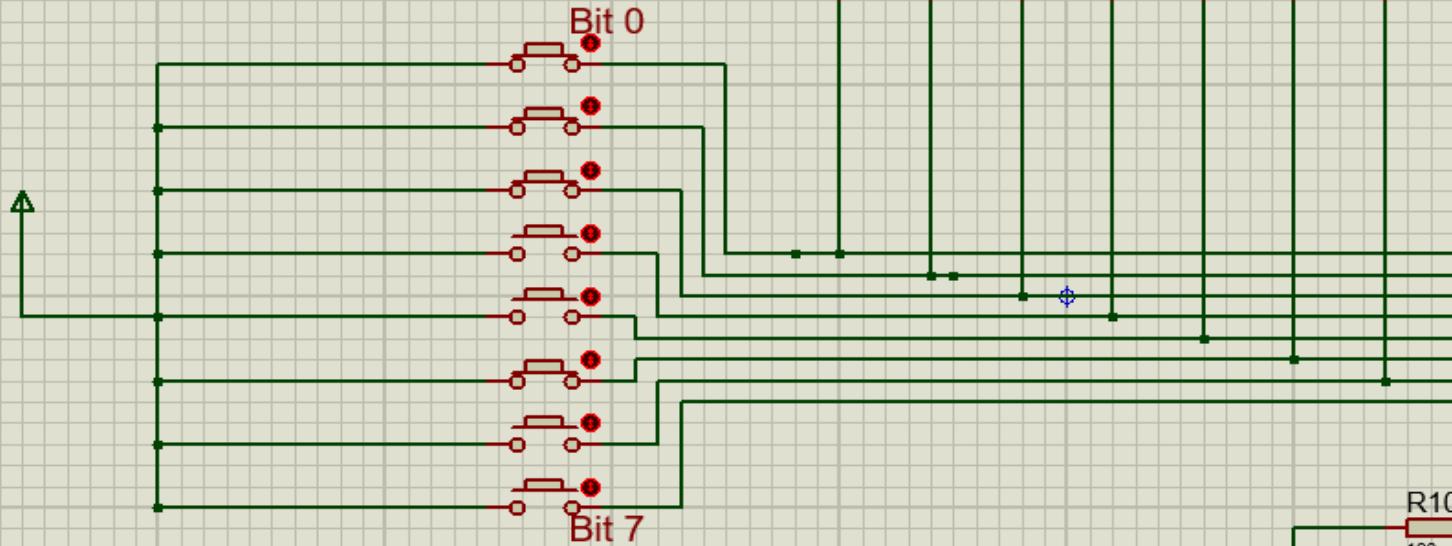
Mastering Embedded System Online Diploma (KS)

www.learn-in-depth.com

FIRst Term Project 1

Eng: Mohammed Hasan

Pressure Sensor





8

Source Code

1

State

```
1  /*
2  =====
3  Name      : State.h
4  Author    : Mohammed Hasan
5  Created on : Oct 10, 2023
6  Description : First Term Project 1 ==> Pressure Controller
7  =====
8  */
9 #ifndef STATE_H_
10#define STATE_H_
11
12#include <stdio.h>
13#include <stdlib.h>
14#include "driver.h"
15
16/*===== Automatic State Function Generated =====*/
17#define State_Define(_STFUNC_) void ST_##_STFUNC_()
18#define State(_STFUNC_) ST_##_STFUNC_
19
20// States Collection
21void Set_Pressure_Value(int Pressure_Value);
22void High_Pressure_Detected();
23void Start_Alarm();
24void Stop_Alarm();
25
26#endif /* STATE_H_ */
```





8

Source Code

2

Pressure Sensor

```
1 /*
2 =====
3 Name      : Pressure_Sensor.h
4 Author    : Mohammed Hasan
5 Created on : Oct 10, 2023
6 Description : First Term Project 1 ==> Pressure Controller
7 =====
8 */
9 #ifndef PRESSURE_SENSOR_DRIVER_H_
10 #define PRESSURE_SENSOR_DRIVER_H_
11 #include "State.h"
12
13 /*===== Define States =====*/
14 enum
15 {
16     PS_Reading ,
17     PS_Waiting
18 }PS_State_id;
19
20 void PS_init();
21 // Declare State Function
22 State_Define(PS_Reading);
23 State_Define(PS_Waiting);
24
25 // extern State pointer
26 extern void (*PS_State)() ;
27
28 #endif /*PRESSURE_SENSOR_DRIVER_H_*/
```

[Back to Agenda](#)

```
1 /*
2 =====
3 Name      : Pressure_Sensor.c
4 Author    : Mohammed Hasan
5 Created on : Oct 10, 2023
6 Description : First Term Project 1 ==> Pressure Controller
7 =====
8 */
9 #include "Pressure_Sensor.h"
10 /*===== Variables =====*/
11 unsigned int PS_Value = 0;
12 unsigned int PS_Pull_Time = 100;
13
14 // State Pointer to Function
15 void (*PS_State)();
16
17 void PS_init()
18 {
19     // Intialize the Pressure Sensor
20 }
21
22 State_Define(PS_Reading)
23 {
24     // State Name
25     PS_State_id = PS_Reading;
26     // Read from Pressure Sensor
27     PS_Value = getPressureVal();
28     Set_Pressure_Value(PS_Value);
29     PS_State = State(PS_Waiting);
30 }
31
32 State_Define(PS_Waiting)
33 {
34     // State Name
35     PS_State_id = PS_Waiting;
36     Delay(50000);
37     PS_State = State(PS_Reading);
38 }
```



8

Source Code

3

Main Algorithm

```
1 /*
2 =====
3 Name      : Main_Algorithm.h
4 Author    : Mohammed Hasan
5 Created on : Oct 10, 2023
6 Description : First Term Project 1 ==> Pressure Controller
7 =====
8 */
9
10 #ifndef MAIN_ALGORITHM_H_
11 #define MAIN_ALGORITHM_H_
12 #include "State.h"
13
14 /*===== Define States =====*/
15 enum
16 {
17     MA_High_Pressure_Detect
18 }MA_State_id ;
19
20 // Declare State Function
21 State_Define(MA_High_Pressure_Detect);
22
23 // extern pointer to function MA_State
24 extern void (*MA_State)();
25
26 #endif /*MAIN_ALGORITHM_H_*/
```

```
1 /*
2 =====
3 Name      : Main_Algorithm.c
4 Author    : Mohammed Hasan
5 Created on : Oct 10, 2023
6 Description : First Term Project 1 ==> Pressure Controller
7 =====
8 */
9 #include "Main_Algorithm.h"
10 /*===== Variables =====*/
11 unsigned int MA_Pressure_Value = 0 ;
12 unsigned int MA_Pressure_Threshold = 20 ;
13
14 // State Pointer to Function
15 void (*MA_State)();
16
17 void Set_Pressure_Value(int Pressure_Value)
18 {
19     MA_Pressure_Value = Pressure_Value ;
20     MA_State = State(MA_High_Pressure_Detect);
21 }
22 // Declare State Function
23 State_Define(MA_High_Pressure_Detect)
24 {
25     // State Name
26     MA_State_id = MA_High_Pressure_Detect ;
27
28     //State Action
29     if(MA_Pressure_Value <= MA_Pressure_Threshold)
30     {
31         MA_State = State(MA_High_Pressure_Detect);
32     }
33     else
34     {
35         High_Pressure_Detected();
36         MA_State = State(MA_High_Pressure_Detect);
37     }
38 }
```



8

Source Code

4

Monitor Alarm

```
1 /*
2 =====
3 Name      : Monitor_Alarm.h
4 Author    : Mohammed Hasan
5 Created on : Oct 10, 2023
6 Description : First Term Project 1 ==> Pressure Controller
7 =====
8 */
9 #ifndef MONITOR_ALARM_H_
10 #define MONITOR_ALARM_H_
11 #include "State.h"
12
13 /*===== Define States =====*/
14 enum
15 {
16     MoA_Alarm_OFF ,
17     MoA_Alarm_ON ,
18     MoA_Waiting
19 }MoA_State_id;
20
21 // Declare State Function
22 State_Define(MoA_Alarm_OFF);
23 State_Define(MoA_Alarm_ON);
24 State_Define(MoA_Waiting);
25
26 // extern pointer
27 extern void(*MoA_State)();
28
29 #endif /*MONITOR_ALARM_H_*/
```

```
1 /*
2 ====
3 Name      : Monitor_Alarm.c
4 Author    : Mohammed Hasan
5 Created on : Oct 10, 2023
6 Description : First Term Project 1 ==> Pressure Controller
7 =====
8 */
9 #include "Monitor_Alarm.h"
10
11 /*===== Variables =====*/
12 unsigned int MoA_Periode = 50000;
13
14 // State Pointer to Function
15 void(*MoA_State)();
16
17 void High_Pressure_Detected()
18 {
19     MoA_State = State(MoA_Alarm_ON);
20 }
21
22 State_Define(MoA_Alarm_OFF)
23 {
24     // State Name
25     MoA_State_id = MoA_Alarm_OFF ;
26     Stop_Alarm();
27     MoA_State = State(MoA_Alarm_OFF);
28 }
29 State_Define(MoA_Alarm_ON)
30 {
31     // State Name
32     MoA_State_id = MoA_Alarm_ON ;
33     Start_Alarm();
34     MoA_State = State(MoA_Waiting);
35 }
36 State_Define(MoA_Waiting)
37 {
38     // State Name
39     MoA_State_id = MoA_Waiting ;
40     Stop_Alarm();
41     Delay(MoA_Periode);
42     MoA_State = State(MoA_Alarm_OFF);
43 }
```



8

Source Code

5

Alarm Actuator

```
1 /*
2 =====
3 Name      : Alarm_Actuator.h
4 Author    : Mohammed Hasan
5 Created on : Oct 10, 2023
6 Description : First Term Project 1 ==> Pressure Controller
7 =====
8 */
9 #ifndef ALARM_ACTUATOR_DRIVER_H_
10 #define ALARM_ACTUATOR_DRIVER_H_
11 #include "State.h"
12 /*===== Define States =====*/
13 enum
14 {
15     AC_Waiting ,
16     AC_Alarm_OFF ,
17     AC_Alarm_ON
18 }AC_State_id;
19
20 void AC_init();
21 // Declare State Function
22 State_Define(AC_Waiting);
23 State_Define(AC_Alarm_OFF);
24 State_Define(AC_Alarm_ON);
25
26 // extern State pointer
27 extern void(*AC_State)();
28#endif /*ALARM_ACTUATOR_DRIVER_H_*/
```

```
1 /*
2 =====
3 Name      : Alarm_Actuator.c
4 Author    : Mohammed Hasan
5 Created on : Oct 10, 2023
6 Description : First Term Project 1 ==> Pressure Controller
7 =====
8 */
9 #include "Alarm_Actuator.h"
10 void(*AC_State)();
11
12 void AC_init()
13 {
14     // intialize Alarm_Actuator Driver
15 }
16
17 void Stop_Alarm()
18 {
19     AC_State = State(AC_Alarm_OFF);
20 }
21
22 void Start_Alarm()
23 {
24     AC_State = State(AC_Alarm_ON);
25 }
26
27 State_Define(AC_Waiting)
28 {
29     //State Name
30     AC_State_id = AC_Waiting;
31     AC_State = State(AC_Waiting);
32 }
33
34 State_Define(AC_Alarm_ON)
35 {
36     //State Name
37     AC_State_id = AC_Alarm_ON;
38     Set_Alarm_actuator(0);
39     AC_State = State(AC_Waiting);
40 }
41
42 State_Define(AC_Alarm_OFF)
43 {
44     //State Name
45     AC_State_id = AC_Alarm_OFF;
46     Set_Alarm_actuator(1);
47     AC_State = State(AC_Waiting);
48 }
```





8

Source Code

6

Driver

```
● ● ●  
1 #include <stdint.h>  
2 #include <stdio.h>  
3  
4 #define SET_BIT(ADDRESS,BIT) ADDRESS |= (1<<BIT)  
5 #define RESET_BIT(ADDRESS,BIT) ADDRESS &= ~(1<<BIT)  
6 #define TOGGLE_BIT(ADDRESS,BIT) ADDRESS ^= (1<<BIT)  
7 #define READ_BIT(ADDRESS,BIT) ((ADDRESS) & (1<<(BIT)))  
8  
9  
10#define GPIO_PORTA 0x40010800  
11#define BASE_RCC 0x40021000  
12  
13#define APB2ENR *(volatile uint32_t *)(BASE_RCC + 0x18)  
14  
15#define GPIOA_CRL *(volatile uint32_t *)(GPIO_PORTA + 0x00)  
16#define GPIOA_CRH *(volatile uint32_t *)(GPIO_PORTA + 0X04)  
17#define GPIOA_IDR *(volatile uint32_t *)(GPIO_PORTA + 0x08)  
18#define GPIOA_ODR *(volatile uint32_t *)(GPIO_PORTA + 0x0C)  
19  
20  
21 void Delay(int nCount);  
22 int getPressureVal();  
23 void Set_Alarm_actuator(int i);  
24 void GPIO_INITIALIZATION();
```

● ● ●

```
1 #include "driver.h"  
2 #include <stdint.h>  
3 #include <stdio.h>  
4 void Delay(int nCount)  
5 {  
6     for(; nCount != 0; nCount--);  
7 }  
8  
9 int getPressureVal(){  
10    return (GPIOA_IDR & 0xFF);  
11 }  
12  
13 void Set_Alarm_actuator(int i){  
14     if (i == 1){  
15         SET_BIT(GPIOA_ODR,13);  
16     }  
17     else if (i == 0){  
18         RESET_BIT(GPIOA_ODR,13);  
19     }  
20 }  
21  
22 void GPIO_INITIALIZATION (){  
23     SET_BIT(APB2ENR, 2);  
24     GPIOA_CRL &= 0xFF0FFFFF;  
25     GPIOA_CRL |= 0x00000000;  
26     GPIOA_CRH &= 0xFF0FFFFF;  
27     GPIOA_CRH |= 0x22222222;  
28 }
```





8

Source Code

7

Main

```
1  /*
2  ****
3  Name      : main.c
4  Author    : Mohammed Hasan
5  Created on : Oct 10, 2023
6  Description : First Term Project 1 ===> Pressure_Controller
7  ****
8  */
9  /*
10  * PS ===> Pressure_Sensor
11  * AC ===> Alarm_Actuator
12  * MA ===> Main_Algorithm
13  * MoA ===> Monitor_Alarm
14  */
15 #include <stdint.h>
16 #include <stdio.h>
17
18 #include "driver.h"
19 #include "Pressure_Sensor.h"
20 #include "Main_Algorithm.h"
21 #include "Monitor_Alarm.h"
22 #include "Alarm_Actuator.h"
23
24 void Setup()
25 {
26     /*
27     * init all the drivers
28     * init IRQ...
29     * init HAL US_Driver DC_Driver
30     * init Block
31     * set States Pointers for each Block
32     */
33     PS_init();
34     AC_init();
35     PS_State = State(PS_Reading);
36     MA_State = Set_Pressure_Value;
37     MoA_State =State(MoA_Alarm_OFF);
38     AC_State = State(AC_Waiting);
39 }
40
41 int main (){
42     GPIO_INITIALIZATION();
43     Setup();
44     while (1)
45     {
46         PS_State();
47         MA_State();
48         MoA_State();
49         AC_State();
50     }
51 }
```





8

Source Code

8

Startup

```
1  /*=====
2  Name      : startup.c
3  Author    : Mohammed Hasan
4  Created on : Oct 10, 2023
5  Description : First Term Project 1 ==> Pressure Controller
6  =====*/
7 #include "Platform_Types.h"
8
9 extern int main(void);
10 extern uint32_t _Stack_Top ;
11 extern uint32_t _E_Text ;
12 extern uint32_t _S_Data ;
13 extern uint32_t _E_Data ;
14 extern uint32_t _S_Bss ;
15 extern uint32_t _E_Bss ;
16
17 void Reset_Handler(void);
18
19 void Default_Handler()
20 {
21     Reset_Handler();
22 }
23
24 void NMI_Handler(void)      __attribute__ ((weak, alias("Default_Handler")));
25 void H_Fault_Handler(void)   __attribute__ ((weak, alias("Default_Handler")));
26 void MM_Fault_Handler(void) __attribute__ ((weak, alias("Default_Handler")));
27 void Bus_Fault(void)        __attribute__ ((weak, alias("Default_Handler")));
28 void Usage_Fault_Handler(void) __attribute__ ((weak, alias("Default_Handler")));
29
30
31 uint32_t vectors[] __attribute__ ((section(".vectors"))) =
32 {
33     (uint32_t) &_Stack_Top ,
34     (uint32_t) &Reset_Handler ,
35     (uint32_t) &NMI_Handler ,
36     (uint32_t) &H_Fault_Handler ,
37     (uint32_t) &MM_Fault_Handler ,
38     (uint32_t) &Bus_Fault ,
39     (uint32_t) &Usage_Fault_Handler
40 };
41
42 void Reset_Handler(void)
43 {
44     //Copy Data From Rom to Ram
45     uint32_t Data_Size = (uint8_t*)&_E_Data - (uint8_t*)&_S_Data ;
46     uint8_t* P_src = (uint8_t*)&_E_Text ;
47     uint8_t* P_dst = (uint8_t*)&_S_Data ;
48
49     for(int i =0 ; i < Data_Size ; i++)
50     {
51         *((uint8_t*)P_dst++) = *((uint8_t*)P_src++);
52     }
53
54     // init the .bss with 0
55
56     uint32_t Bss_Size = (uint8_t*)&_E_Bss - (uint8_t*)&_S_Bss ;
57     P_dst = (uint8_t*)&_S_Bss ;
58
59     for(int i =0 ; i < Bss_Size ; i++)
60     {
61         *((uint8_t*)P_dst++) = (uint8_t)0;
62     }
63     // jump to main()
64     main();
65 }
```





Source Code



LinkerScript

```
1  /*=====
2  Name      : linker_script.ld
3  Author    : Mohammed Hasan
4  Created on : Oct 10, 2023
5  Description : First Term Project 1 ==> Pressure Controller
6  =====*/
7
8 MEMORY
9 {
10     flash(RX) : ORIGIN = 0x08000000, LENGTH = 128K
11     sram(RWX) : ORIGIN = 0x20000000, LENGTH = 20K
12 }
13
14 SECTIONS
15 {
16     .text : {
17         *(.vectors*)
18         *(.text*)
19         *(.rodata*)
20         _E_Text = . ;
21     }>flash
22
23     .data : {
24         _S_Data = . ;
25         *(.data*)
26         . = ALIGN(4);
27         _E_Data = . ;
28     }>sram AT> flash
29
30     .bss : {
31         _S_Bss = . ;
32         *(.bss*)
33         . = ALIGN(4);
34         _E_Bss = . ;
35
36         . = ALIGN(4);
37         . = . + 0x1000 ;
38         _Stack_Top = . ;
39     }>sram
40 }
```





8

Source Code

10

Makefile

```
1 #Learn-In-Depth
2 #Eng.Mohammed_Hasan
3 #Unit 5 ==> First Term Project 1 ==> Pressure Controller
4 #makefile
5 CC=arm-none-eabi-
6 CFLAGS=-mcpu=cortex-m3 -gdwarf-2
7 INCs=-I .
8 LIBS=
9 SRC=$(wildcard *.c)
10 OBJ=$(SRC:.c=.o)
11 As=$(wildcard *.s)
12 AsOBJ=$(As:.s=.o)
13 Project_name=Pressure_Controller
14
15 all: $(Project_name).bin
16     @echo "===== Build is Done ====="
17
18 %.o: %.s
19     $(CC)as.exe $(CFLAGS) $< -o $@
20
21 %.o: %.c
22     $(CC)gcc.exe -c $(CFLAGS) $(INCs) $< -o $@
23
24 $(Project_name).elf: $(OBJ) $(AsOBJ)
25     $(CC)ld.exe -T linker_script.ld $(LIBS) $(OBJ) -Map=Map_File.map -o $@
26
27 $(Project_name).bin: $(Project_name).elf
28     $(CC)objcopy.exe -O binary $< $@
29
30 clean_all:
31     rm *.bin *.o *.elf
32
33 clean:
34     rm *.bin *.elf
```



9

Sections & Symbols of object files

1

Pressure Sensor

```
$ arm-none-eabi-objdump.exe -h Pressure_Sensor.o

Pressure_Sensor.o:      file format elf32-littlearm

Sections:
Idx Name      Size    VMA      LMA      File off  Align
 0 .text     00000070 00000000 00000000 00000034 2**2
              CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data     00000004 00000000 00000000 000000a4 2**2
              CONTENTS, ALLOC, LOAD, DATA
 2 .bss     00000004 00000000 00000000 000000a8 2**2
              ALLOC
 3 .debug_info 00000a38 00000000 00000000 000000a8 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 4 .debug_abbrev 000001e1 00000000 00000000 00000ae0 2**0
              CONTENTS, READONLY, DEBUGGING
 5 .debug_loc   0000009c 00000000 00000000 00000cc1 2**0
              CONTENTS, READONLY, DEBUGGING
 6 .debug_aranges 00000020 00000000 00000000 00000d5d 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 7 .debug_line   000001a9 00000000 00000000 00000d7d 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 8 .debug_str    00000580 00000000 00000000 00000f26 2**0
              CONTENTS, READONLY, DEBUGGING
 9 .comment     0000007f 00000000 00000000 000014a6 2**0
              CONTENTS, READONLY
10 .debug_frame 00000068 00000000 00000000 00001528 2**2
              CONTENTS, RELOC, READONLY, DEBUGGING
11 .ARM.attributes 00000033 00000000 00000000 00001590 2**0
              CONTENTS, READONLY
```

2

Main Algorithm

```
$ arm-none-eabi-objdump.exe -h Main_Algorithm.o

Main_Algorithm.o:      file format elf32-littlearm

Sections:
Idx Name      Size    VMA      LMA      File off  Align
 0 .text     0000006c 00000000 00000000 00000034 2**2
              CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data     00000004 00000000 00000000 000000a0 2**2
              CONTENTS, ALLOC, LOAD, DATA
 2 .bss     00000004 00000000 00000000 000000a4 2**2
              ALLOC
 3 .debug_info 00000a2d 00000000 00000000 000000a4 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 4 .debug_abbrev 000001f2 00000000 00000000 00000ad1 2**0
              CONTENTS, READONLY, DEBUGGING
 5 .debug_loc   00000088 00000000 00000000 00000cc3 2**0
              CONTENTS, READONLY, DEBUGGING
 6 .debug_aranges 00000020 00000000 00000000 00000d4b 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 7 .debug_line   000001a6 00000000 00000000 00000d6b 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 8 .debug_str    000005ac 00000000 00000000 00000f11 2**0
              CONTENTS, READONLY, DEBUGGING
 9 .comment     0000007f 00000000 00000000 000014bd 2**0
              CONTENTS, READONLY
10 .debug_frame 00000054 00000000 00000000 0000153c 2**2
              CONTENTS, RELOC, READONLY, DEBUGGING
11 .ARM.attributes 00000033 00000000 00000000 00001590 2**0
              CONTENTS, READONLY
```



9

Sections & Symbols of object files

3

Monitor Alarm

```
$ arm-none-eabi-objdump.exe -h Monitor_Alarm.o

Monitor_Alarm.o:      file format elf32-littlearm

Sections:
Idx Name      Size    VMA      LMA      File off  Align
 0 .text     00000098 00000000 00000000 00000034 2**2
              CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data     00000004 00000000 00000000 000000cc 2**2
              CONTENTS, ALLOC, LOAD, DATA
 2 .bss      00000000 00000000 00000000 000000d0 2**0
              ALLOC
 3 .debug_info 00000a41 00000000 000000d0 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 4 .debug_abbrev 000001e1 00000000 00000000 00000b11 2**0
              CONTENTS, READONLY, DEBUGGING
 5 .debug_loc   000000c8 00000000 00000000 00000cf2 2**0
              CONTENTS, READONLY, DEBUGGING
 6 .debug_aranges 00000020 00000000 00000000 00000dba 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 7 .debug_line   000001ab 00000000 00000000 00000dda 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 8 .debug_str    000005aa 00000000 00000000 00000f85 2**0
              CONTENTS, READONLY, DEBUGGING
 9 .comment     0000007f 00000000 00000000 0000152f 2**0
              CONTENTS, READONLY
10 .debug_frame 00000084 00000000 00000000 000015b0 2**2
              CONTENTS, RELOC, READONLY, DEBUGGING
11 .ARM.attributes 00000033 00000000 00000000 00001634 2**0
              CONTENTS, READONLY
```

4

Alarm Actuator

```
$ arm-none-eabi-objdump.exe -h Alarm_Actuator.o

Alarm_Actuator.o:      file format elf32-littlearm

Sections:
Idx Name      Size    VMA      LMA      File off  Align
 0 .text     000000b8 00000000 00000000 00000034 2**2
              CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data     00000000 00000000 00000000 000000ec 2**0
              CONTENTS, ALLOC, LOAD, DATA
 2 .bss      00000000 00000000 00000000 000000ec 2**0
              ALLOC
 3 .debug_info 00000a59 00000000 00000000 000000ec 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 4 .debug_abbrev 000001e1 00000000 00000000 00000b45 2**0
              CONTENTS, READONLY, DEBUGGING
 5 .debug_loc   00000068 00000000 00000000 00000d26 2**0
              CONTENTS, READONLY, DEBUGGING
 6 .debug_aranges 00000020 00000000 00000000 00000e8e 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 7 .debug_line   000001b0 00000000 00000000 00000eae 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 8 .debug_str    0000059f 00000000 00000000 0000105e 2**0
              CONTENTS, READONLY, DEBUGGING
 9 .comment     0000007f 00000000 00000000 000015fd 2**0
              CONTENTS, READONLY
10 .debug_frame 000000c8 00000000 00000000 0000167c 2**2
              CONTENTS, RELOC, READONLY, DEBUGGING
11 .ARM.attributes 00000033 00000000 00000000 00001744 2**0
              CONTENTS, READONLY
```



9

Sections & Symbols of object files

5

Startup

```
$ arm-none-eabi-objdump.exe -h startup.o

startup.o:      file format elf32-littlearm

Sections:
Idx Name      Size    VMA     LMA     File off  Align
 0 .text      00000090 00000000 00000000 00000034 2**2
              CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data      00000000 00000000 00000000 000000c4 2**0
              CONTENTS, ALLOC, LOAD, DATA
 2 .bss       00000000 00000000 00000000 000000c4 2**0
              ALLOC
 3 .vectors   0000001c 00000000 00000000 000000c4 2**2
              CONTENTS, ALLOC, LOAD, RELOC, DATA
 4 .debug_info 0000019e 00000000 00000000 000000e0 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 5 .debug_abbrev 000000d6 00000000 00000000 0000027e 2**0
              CONTENTS, READONLY, DEBUGGING
 6 .debug_loc   0000007c 00000000 00000000 00000354 2**0
              CONTENTS, READONLY, DEBUGGING
 7 .debug_aranges 00000020 00000000 00000000 000003d0 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 8 .debug_line   0000007d 00000000 00000000 000003f0 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 9 .debug_str    000001a0 00000000 00000000 0000046d 2**0
              CONTENTS, READONLY, DEBUGGING
10 .comment     0000007f 00000000 00000000 0000060d 2**0
              CONTENTS, READONLY
11 .debug_frame 00000050 00000000 00000000 0000068c 2**2
              CONTENTS, RELOC, READONLY, DEBUGGING
12 .ARM.attributes 00000033 00000000 00000000 000006dc 2**0
              CONTENTS, READONLY
```

6

Driver

```
$ arm-none-eabi-objdump.exe -h driver.o

driver.o:      file format elf32-littlearm

Sections:
Idx Name      Size    VMA     LMA     File off  Align
 0 .text      000000c4 00000000 00000000 00000034 2**2
              CONTENTS, ALLOC, LOAD, READONLY, CODE
 1 .data      00000000 00000000 00000000 000000f8 2**0
              CONTENTS, ALLOC, LOAD, DATA
 2 .bss       00000000 00000000 00000000 000000f8 2**0
              ALLOC
 3 .debug_info 00000a05 00000000 00000000 000000f8 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 4 .debug_abbrev 000001de 00000000 00000000 00000afd 2**0
              CONTENTS, READONLY, DEBUGGING
 5 .debug_loc   00000140 00000000 00000000 00000cdb 2**0
              CONTENTS, READONLY, DEBUGGING
 6 .debug_aranges 00000020 00000000 00000000 00000e1b 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 7 .debug_line   000001b9 00000000 00000000 00000e3b 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 8 .debug_str    0000055f 00000000 00000000 00000ff4 2**0
              CONTENTS, READONLY, DEBUGGING
 9 .comment     0000007f 00000000 00000000 00001553 2**0
              CONTENTS, READONLY
10 .debug_frame 000000a0 00000000 00000000 000015d4 2**2
              CONTENTS, RELOC, READONLY, DEBUGGING
11 .ARM.attributes 00000033 00000000 00000000 00001674 2**0
              CONTENTS, READONLY
```



9

Sections & Symbols of object files

7

Main

```
$ arm-none-eabi-objdump.exe -h main.o

main.o:      file format elf32-littlearm

Sections:
Idx Name      Size    VMA     LMA     File off  Align
 0 .text      00000080 00000000 00000000 00000034 2**2
              CONTENTS, ALLOC, LOAD, RELOC, READONLY, CODE
 1 .data      00000000 00000000 00000000 000000b4 2**0
              CONTENTS, ALLOC, LOAD, DATA
 2 .bss       00000000 00000000 00000000 000000b4 2**0
              ALLOC
 3 .debug_info 00000aa7 00000000 00000000 000000b4 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 4 .debug_abbrev 000001d6 00000000 00000000 00000b5b 2**0
              CONTENTS, READONLY, DEBUGGING
 5 .debug_loc   00000058 00000000 00000000 00000d31 2**0
              CONTENTS, READONLY, DEBUGGING
 6 .debug_aranges 00000020 00000000 00000000 00000d89 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 7 .debug_line   000001f1 00000000 00000000 00000da9 2**0
              CONTENTS, RELOC, READONLY, DEBUGGING
 8 .debug_str    000005ea 00000000 00000000 00000f9a 2**0
              CONTENTS, READONLY, DEBUGGING
 9 .comment     0000007f 00000000 00000000 00001584 2**0
              CONTENTS, READONLY
10 .debug_frame 00000048 00000000 00000000 00001604 2**2
              CONTENTS, RELOC, READONLY, DEBUGGING
11 .ARM.attributes 00000033 00000000 00000000 0000164c 2**0
              CONTENTS, READONLY
```

8

Pressure Controller

```
$ arm-none-eabi-objdump.exe -h Pressure_Controller.elf

Pressure_Controller.elf:      file format elf32-littlearm

Sections:
Idx Name      Size    VMA     LMA     File off  Align
 0 .text      0000041c 08000000 08000000 00010000 2**2
              CONTENTS, ALLOC, LOAD, READONLY, CODE
 1 .data      0000000c 20000000 0800041c 00020000 2**2
              CONTENTS, ALLOC, LOAD, DATA
 2 .bss       00001020 2000000c 08000428 0002000c 2**2
              ALLOC
 3 .debug_info 00003f49 00000000 00000000 0002000c 2**0
              CONTENTS, READONLY, DEBUGGING
 4 .debug_abbrev 00000c1f 00000000 00000000 00023f55 2**0
              CONTENTS, READONLY, DEBUGGING
 5 .debug_loc   00000568 00000000 00000000 00024b74 2**0
              CONTENTS, READONLY, DEBUGGING
 6 .debug_aranges 000000e0 00000000 00000000 000250dc 2**0
              CONTENTS, READONLY, DEBUGGING
 7 .debug_line   00000ad1 00000000 00000000 000251bc 2**0
              CONTENTS, READONLY, DEBUGGING
 8 .debug_str    00000777 00000000 00000000 00025c8d 2**0
              CONTENTS, READONLY, DEBUGGING
 9 .comment     0000007e 00000000 00000000 00026404 2**0
              CONTENTS, READONLY
10 .ARM.attributes 00000033 00000000 00000000 00026482 2**0
              CONTENTS, READONLY
11 .debug_frame 00000340 00000000 00000000 000264b8 2**2
              CONTENTS, READONLY, DEBUGGING
```



10

Symbol Table for object files

1

Pressure Sensor

```
$ arm-none-eabi-nm.exe Pressure_Sensor.o
    U Delay
    U getPressureVal
00000000 T PS_init
00000000 D PS_Pull_Time
00000004 C PS_State
00000001 C PS_State_id
00000000 B PS_Value
    U Set_Pressure_Value
0000000c T ST_PS_Reading
00000048 T ST_PS_Waiting
```

2

Main Algorithm

```
$ arm-none-eabi-nm.exe Main_Algorithm.o
    U High_Pressure_Detected
00000000 D MA_Pressure_Threshold
00000000 B MA_Pressure_Value
00000004 C MA_State
00000001 C MA_State_id
00000000 T Set_Pressure_Value
0000002c T ST_MA_High_Pressure_Detect
```

3

Monitor Alarm

```
$ arm-none-eabi-nm.exe Monitor_Alarm.o
    U Delay
00000000 T High_Pressure_Detected
00000000 D MoA_Periode
00000004 C MoA_State
00000001 C MoA_State_id
00000001c T ST_MoA_Alarm_OFF
00000040 T ST_MoA_Alarm_ON
00000064 T ST_MoA_Waiting
    U Start_Alarm
    U Stop_Alarm
```

4

Alarm Actuator

```
$ arm-none-eabi-nm.exe Alarm_Actuator.o
00000000 T AC_init
00000004 C AC_State
00000001 C AC_State_id
    U Set_Alarm_actuator
00000090 T ST_AC_Alarm_OFF
00000068 T ST_AC_Alarm_ON
00000044 T ST_AC_Waiting
00000028 T Start_Alarm
0000000c T Stop_Alarm
```



10

Symbol Table for object files

5

Startup

```
$ arm-none-eabi-nm.exe startup.o
U __Bss
U __Data
U __Text
U __S_Bss
U __S_Data
U __Stack_Top
00000000 W Bus_Fault
00000000 T Default_Handler
00000000 W H_Fault_Handler
U main
00000000 W MM_Fault_Handler
00000000 W NMI_Handler
0000000c T Reset_Handler
00000000 W Usage_Fault_Handler
00000000 D vectors
```

6

Driver

```
$ arm-none-eabi-nm.exe driver.o
00000000 T Delay
00000020 T getPressureVal
00000074 T GPIO_INITIALIZATION
00000038 T Set_Alarm_actuator
```

7

Main

```
$ arm-none-eabi-nm.exe main.o
U AC_init
U AC_State
00000001 C AC_State_id
U GPIO_INITIALIZATION
U MA_State
00000001 C MA_State_id
00000048 T main
U MoA_State
00000001 C MoA_State_id
U PS_init
U PS_State
00000001 C PS_State_id
U Set_Pressure_Value
00000000 T Setup
U ST_AC_waiting
U ST_MoA_Alarm_OFF
U ST_PS_Reading
```



10

Symbol Table for object files

8

Pressure Controller

```
$ arm-none-eabi-nm.exe Pressure_Controller.elf
20000014 B _E_Bss
2000000c D _E_Data
0800041c T _E_Text
2000000c B _S_Bss
20000000 D _S_Data
20001014 B _Stack_Top
0800001c T AC_init
20001018 B AC_State
20001014 B AC_State_id
0800038c W Bus_Fault
0800038c T Default_Handler
080000d4 T Delay
080000f4 T getPressureVal
08000148 T GPIO_INITIALIZATION
0800038c W H_Fault_Handler
08000284 T High_Pressure_Detected
20000000 D MA_Pressure_Threshold
2000000c B MA_Pressure_Value
20001020 B MA_State
2000101c B MA_State_id
080001e0 T main
0800038c W MM_Fault_Handler
20000004 D MoA_Periode
20001024 B MoA_State
2000101e B MoA_State_id
0800038c W NMI_Handler
0800031c T PS_init
20000008 D PS_Pull_Time
20001028 B PS_State
2000101d B PS_State_id
20000010 B PS_Value
08000398 T Reset_Handler
0800010c T Set_Alarm_actuator
08000218 T Set_Pressure_Value
08000198 T Setup
080000ac T ST_AC_Alarm_OFF
08000084 T ST_AC_Alarm_ON
08000060 T ST_AC_Waiting
08000244 T ST_MA_High_Pressure_Detect
080002a0 T ST_MoA_Alarm_OFF
080002c4 T ST_MoA_Alarm_ON
080002e8 T ST_MoA_Waiting
08000328 T ST_PS_Reading
08000364 T ST_PS_Waiting
08000044 T Start_Alarm
08000028 T Stop_Alarm
0800038c W Usage_Fault_Handler
08000000 T vectors
```

[Back to Agenda](#)



11

Size Sections

text	data	bss	dec	hex	filename
184	0	0	184	b8	Alarm_Actuator.o
196	0	0	196	c4	driver.o
128	0	0	128	80	main.o
108	4	4	116	74	Main_Algorithm.o
152	4	0	156	9c	Monitor_Alarm.o
112	4	4	120	78	Pressure_Sensor.o
144	28	0	172	ac	startup.o
1052	12	4128	5192	1448	Pressure_Controller.elf

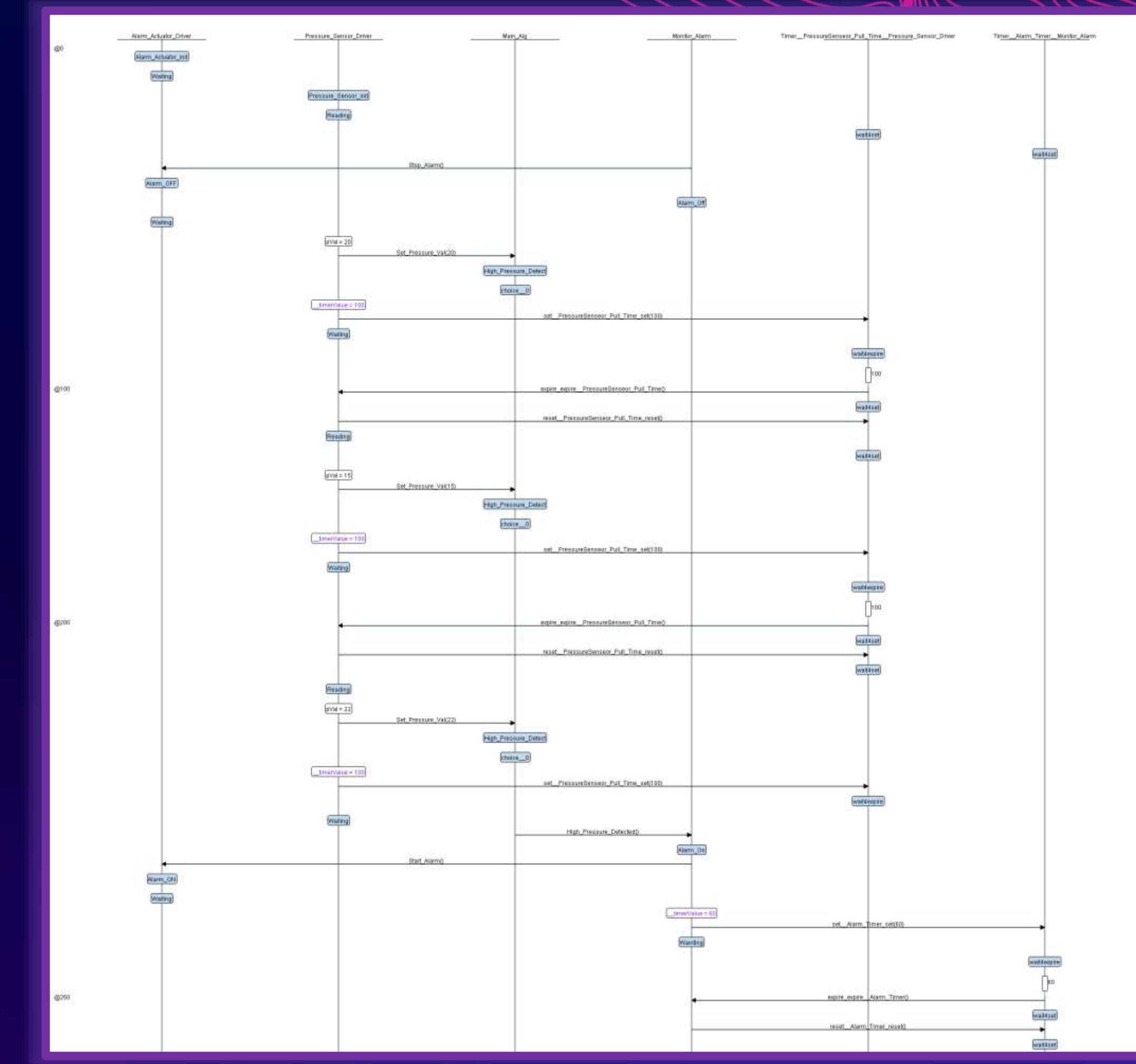




12

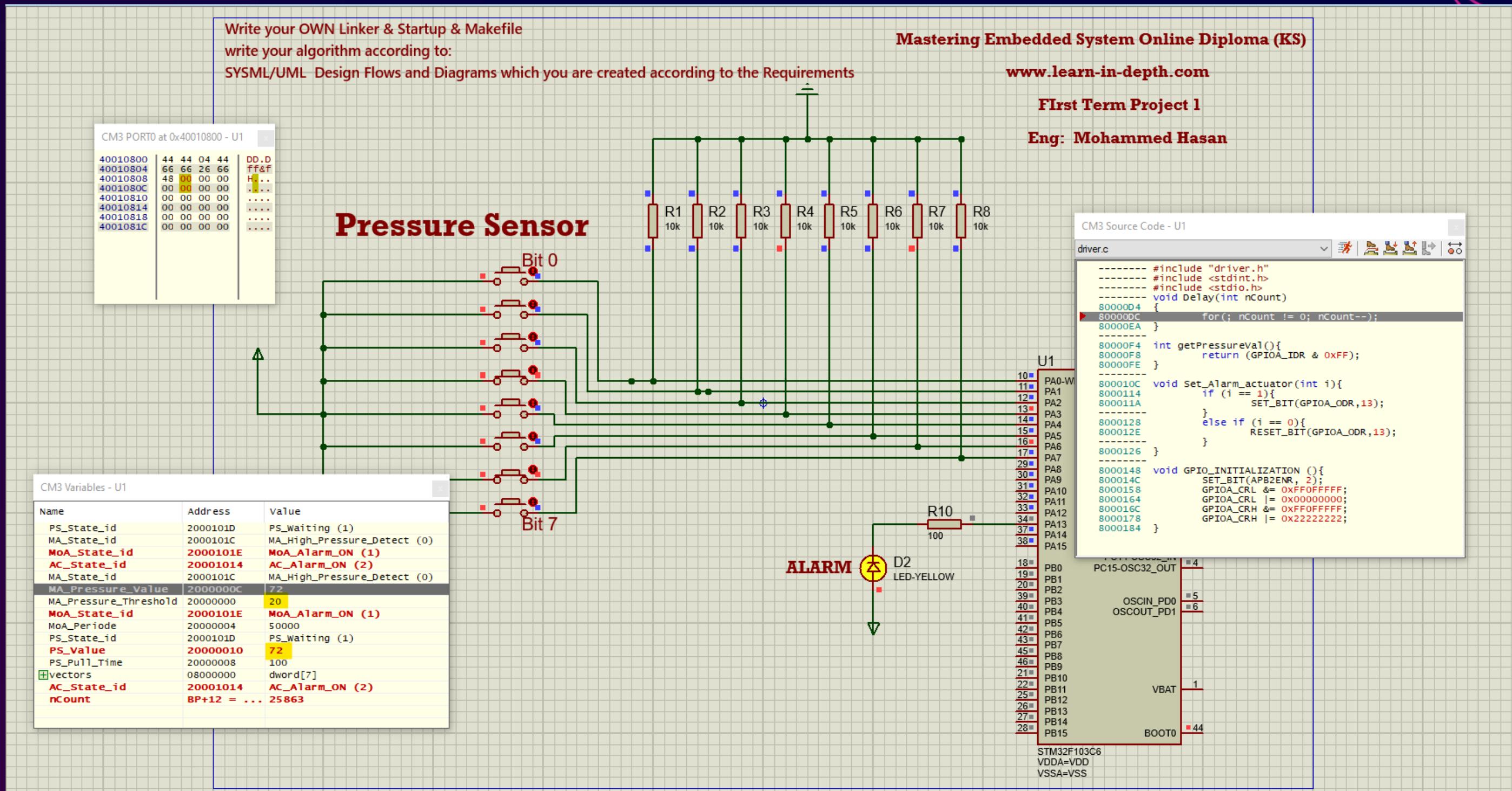
Logic Verification

[Back to Agenda](#)



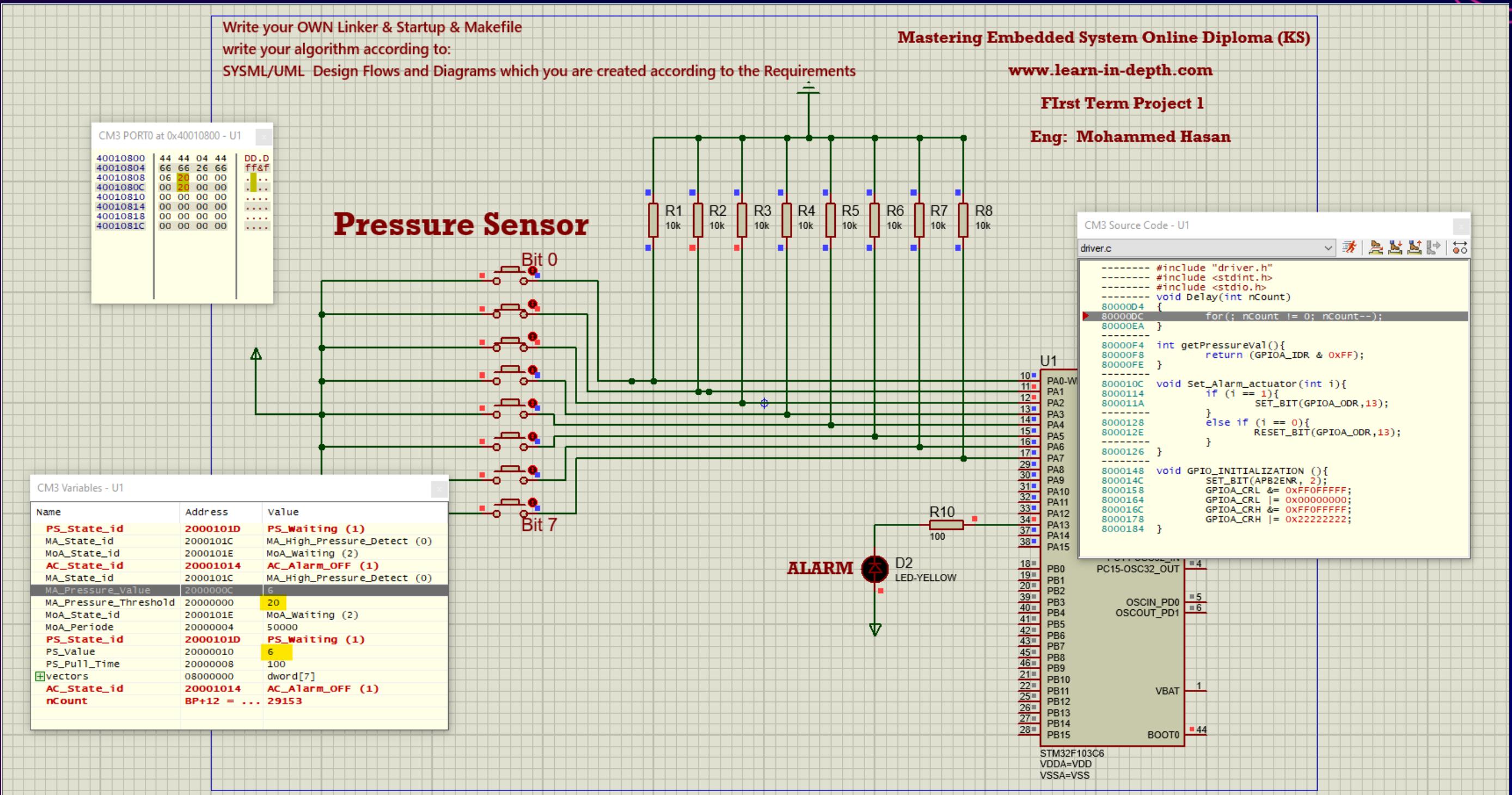
13

Simulation Output


[Back to Agenda](#)


13

Simulation Output


[Back to Agenda](#)




13

Simulation Output

You can see it in my Drive: [click here](#)





Learn-In-Depth

THANK YOU



mohasanbder@gmail.com



My Progress profile: [Click Here](#)



My Drive: [Click Here](#)



GitHub: [Click Here](#)