Learn-In-Depth

Mastering Embedded System
Online Diploma

Pressure Controller
PROJECT

Presented by: **Mohammed Hasan**

For My Profile: Click Here

12 / 10 / 2023

Learn-In-Depth

System Architecting && Design Sequence

1. Case Study
2. Method
3. Requirement
4. Space Exploration
5. System Analysis
6. System Design

Learn-In-Depth

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

1 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.

Back to Agenda

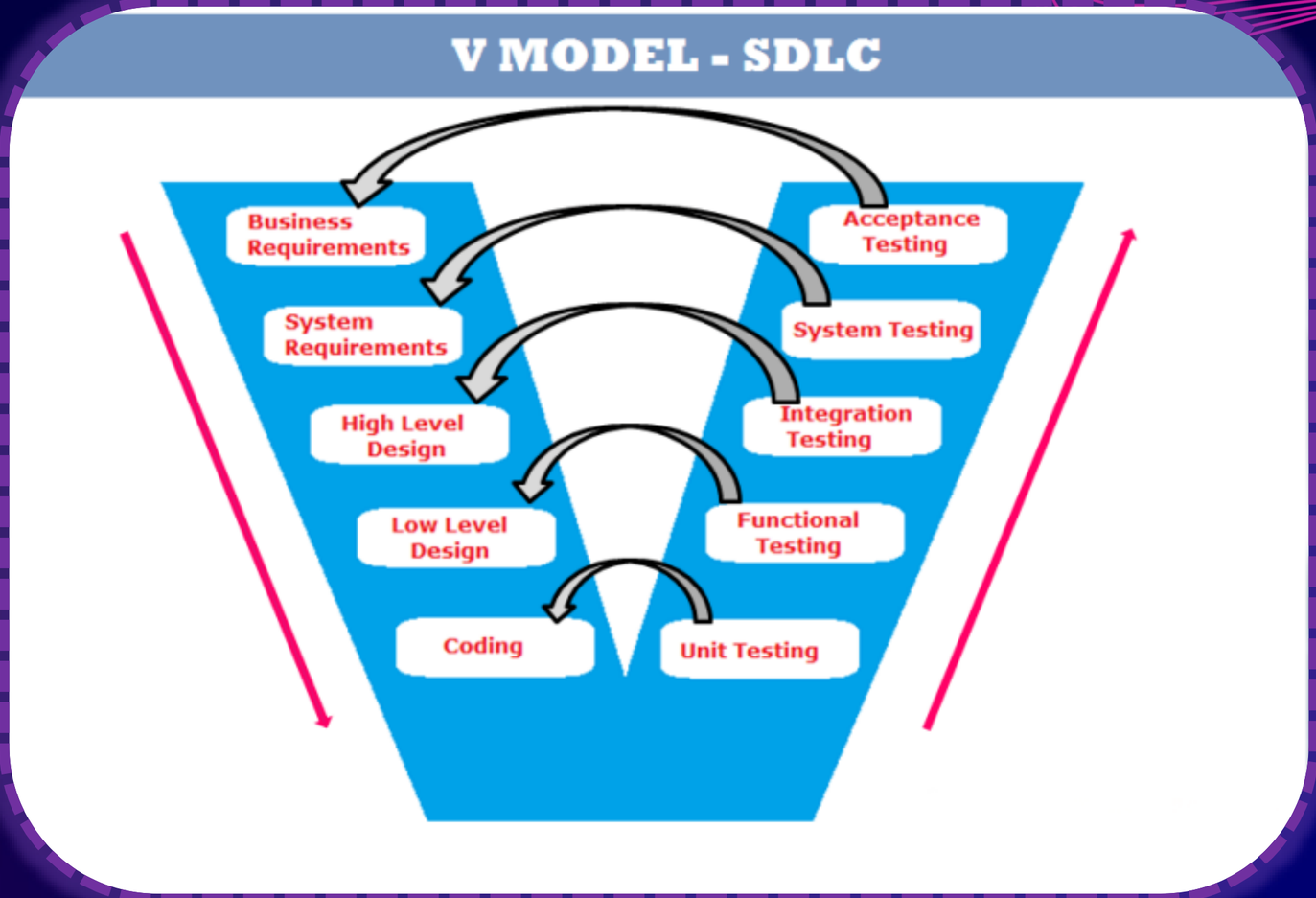
1 Assumptions

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 .

Back to Agenda

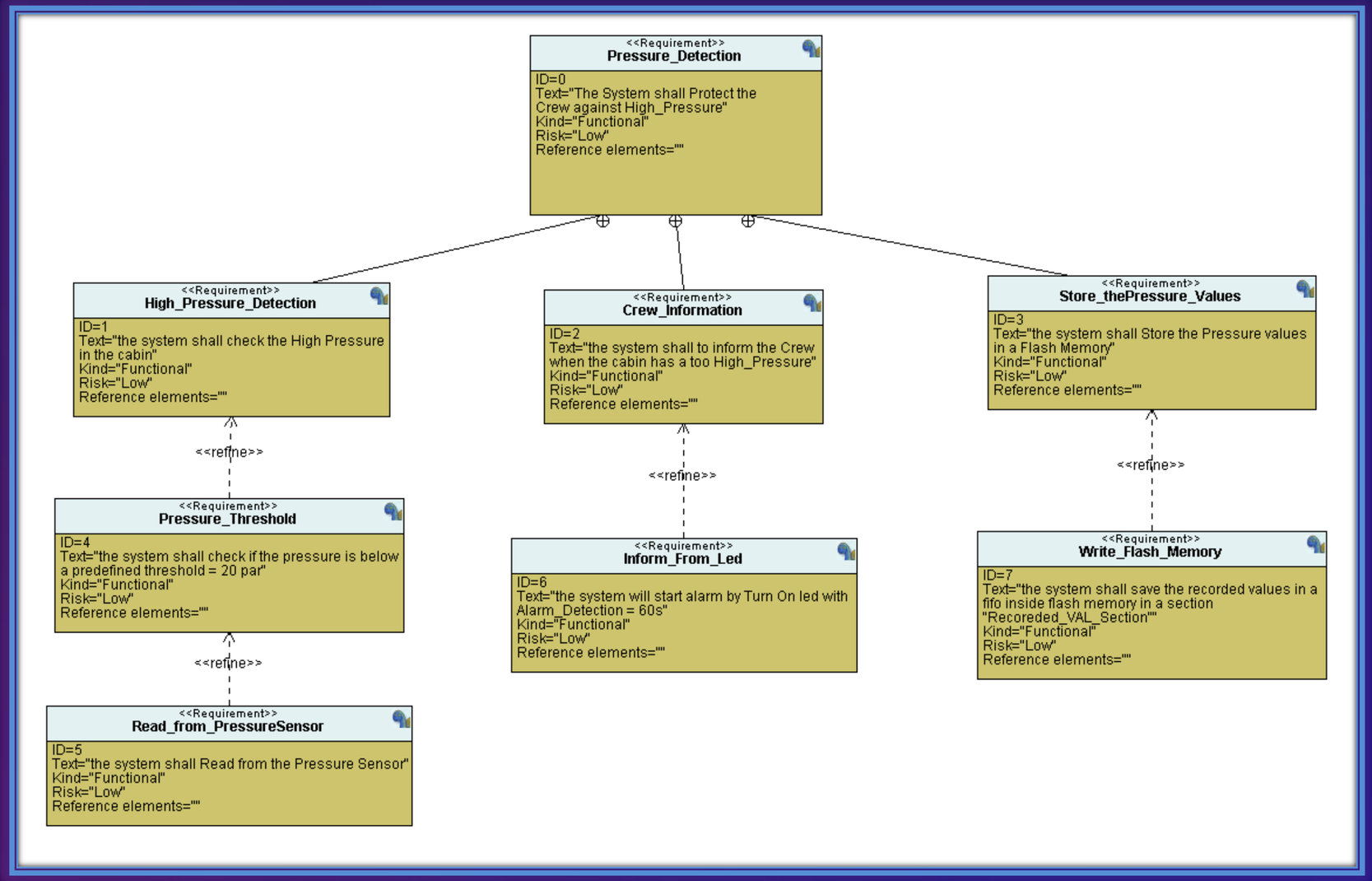
2 Method

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



Back to Agenda

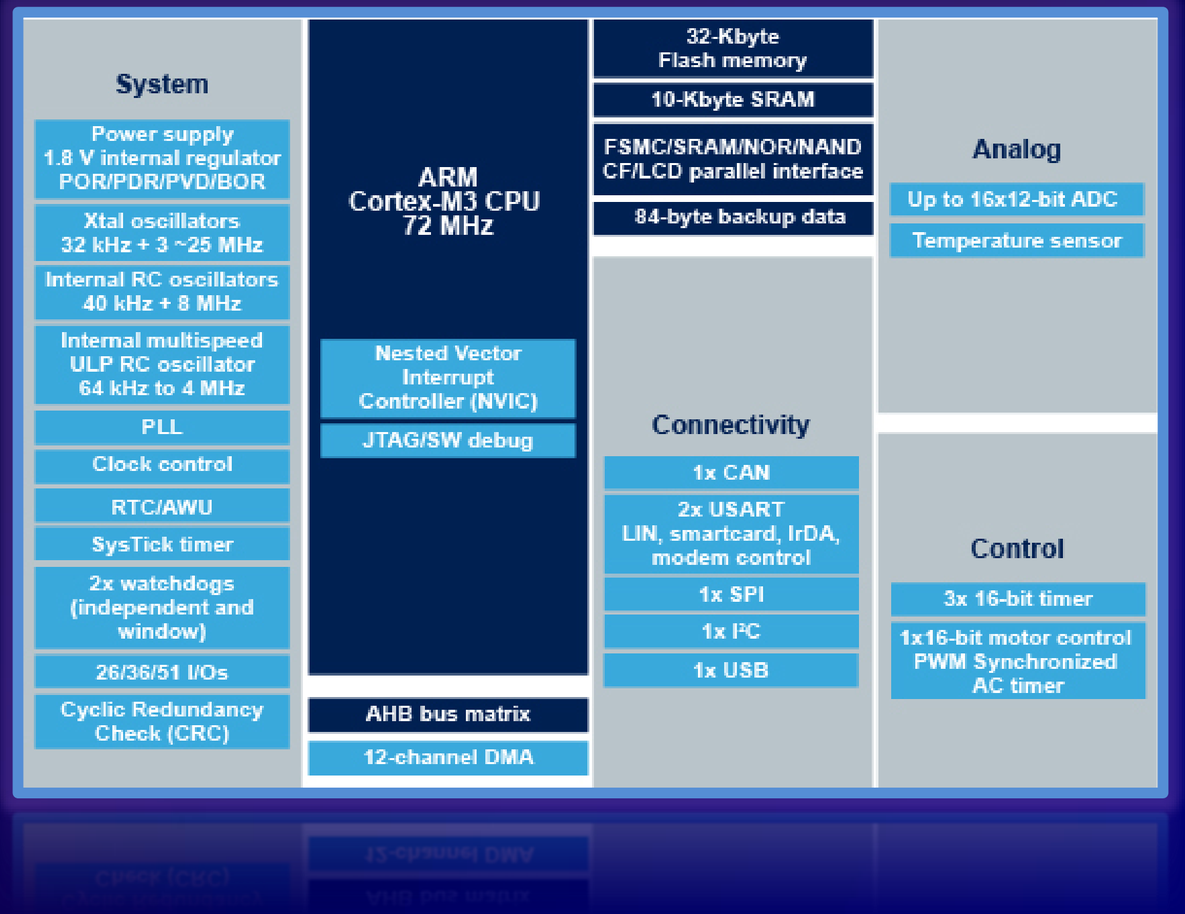
3 Requirement



Back to Agenda

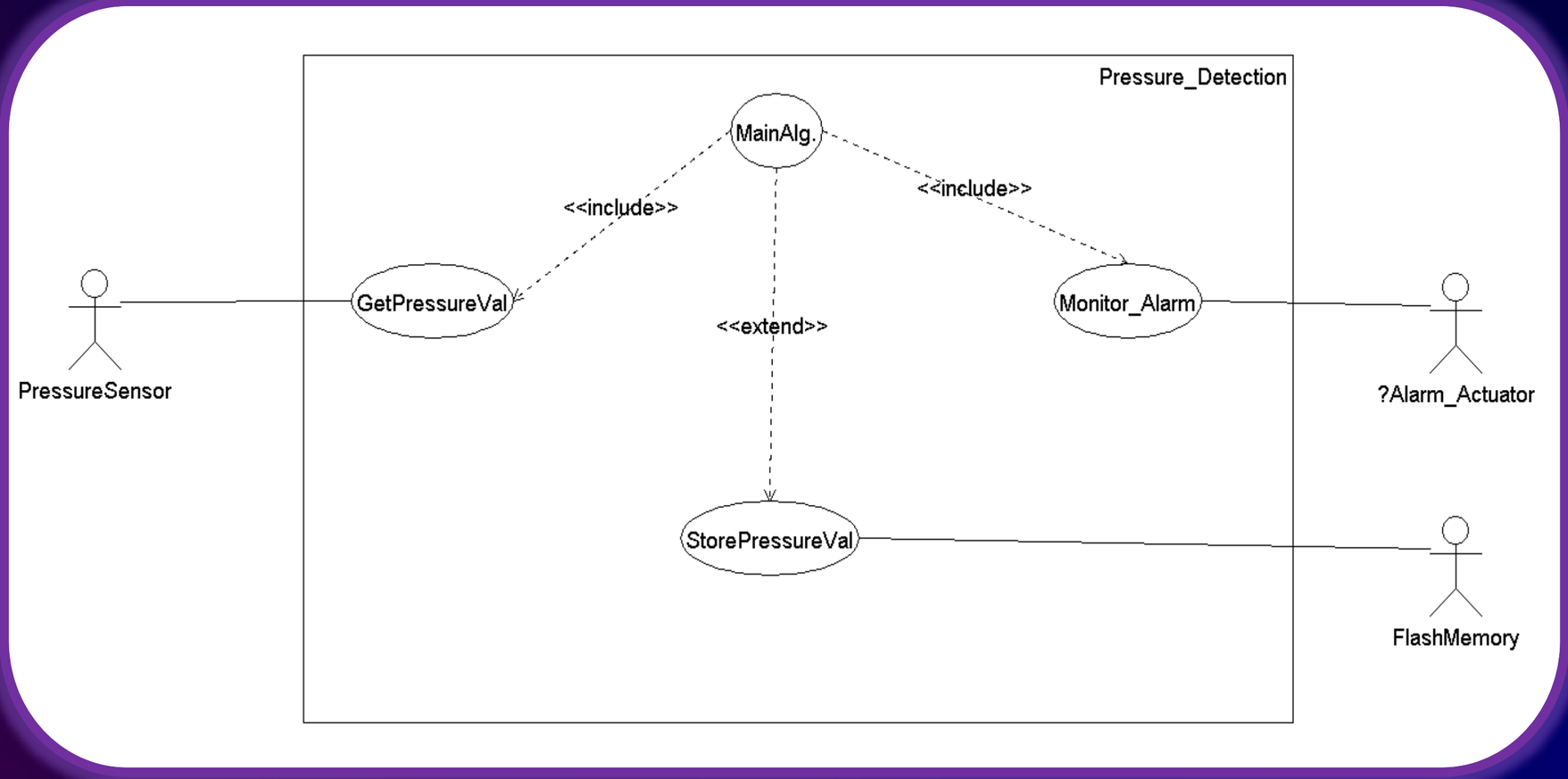
4 Space Exploration

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



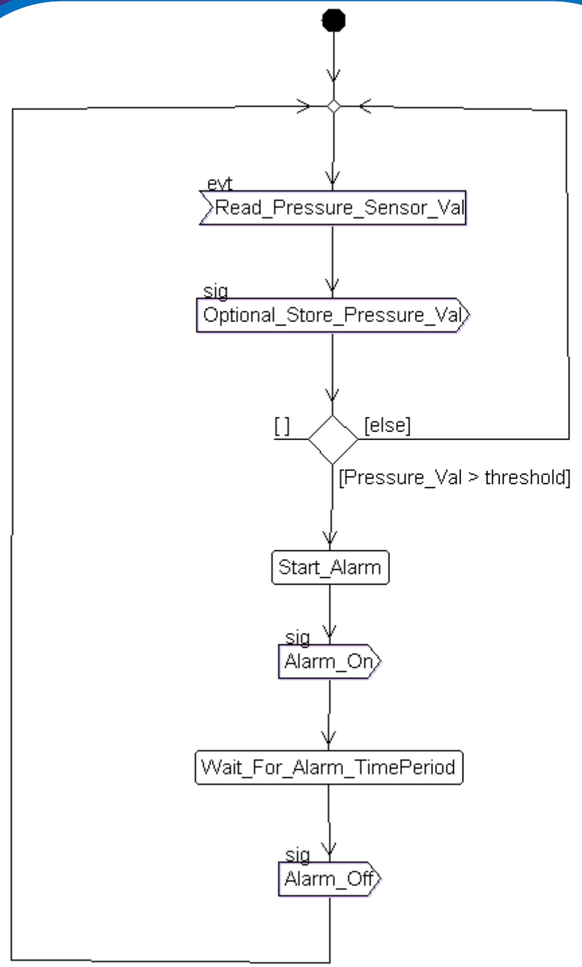
Back to Agenda

5 System Analysis : Use Case Diagram



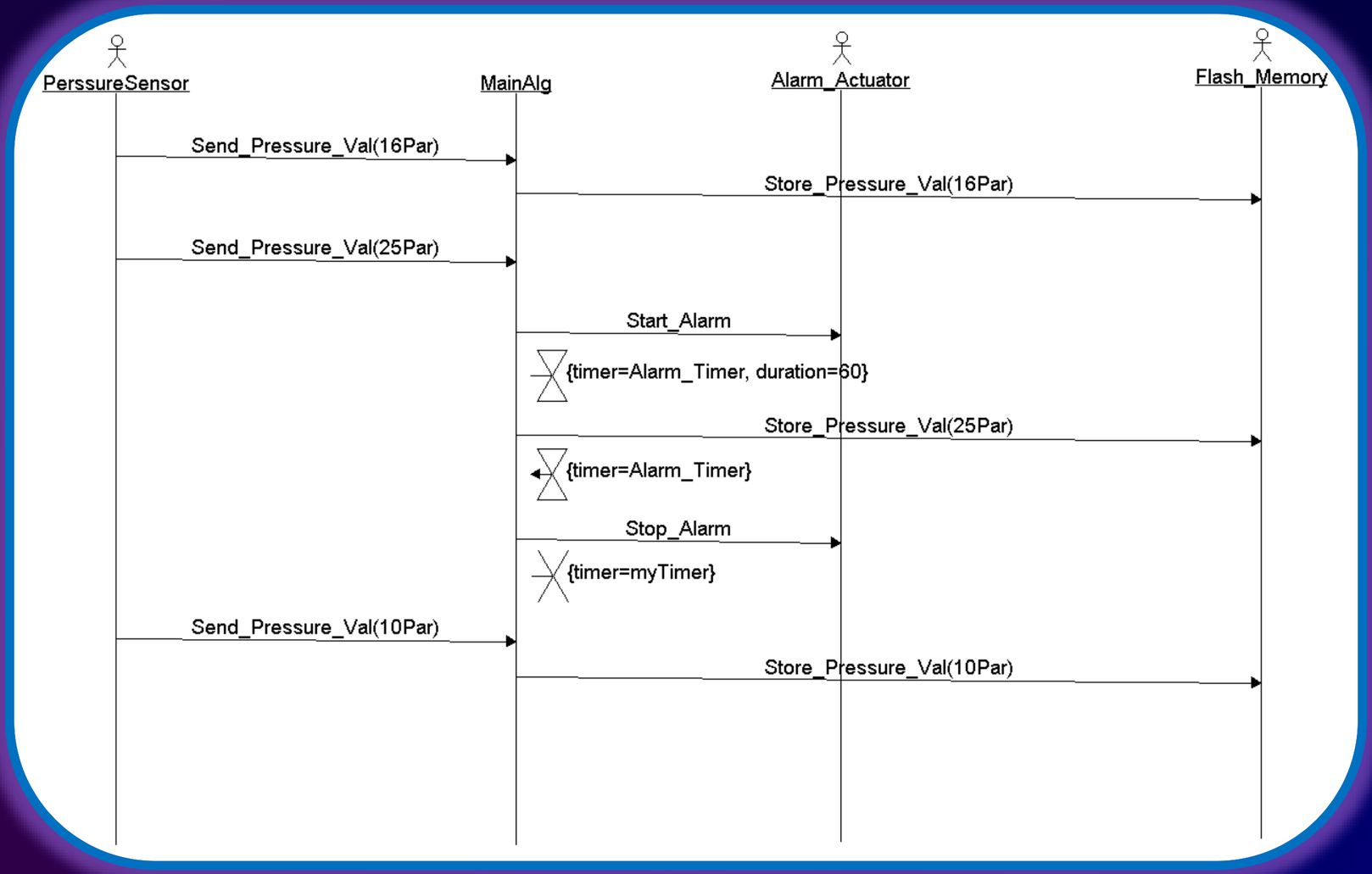
Back to Agenda

5 System Analysis : Activity Diagram



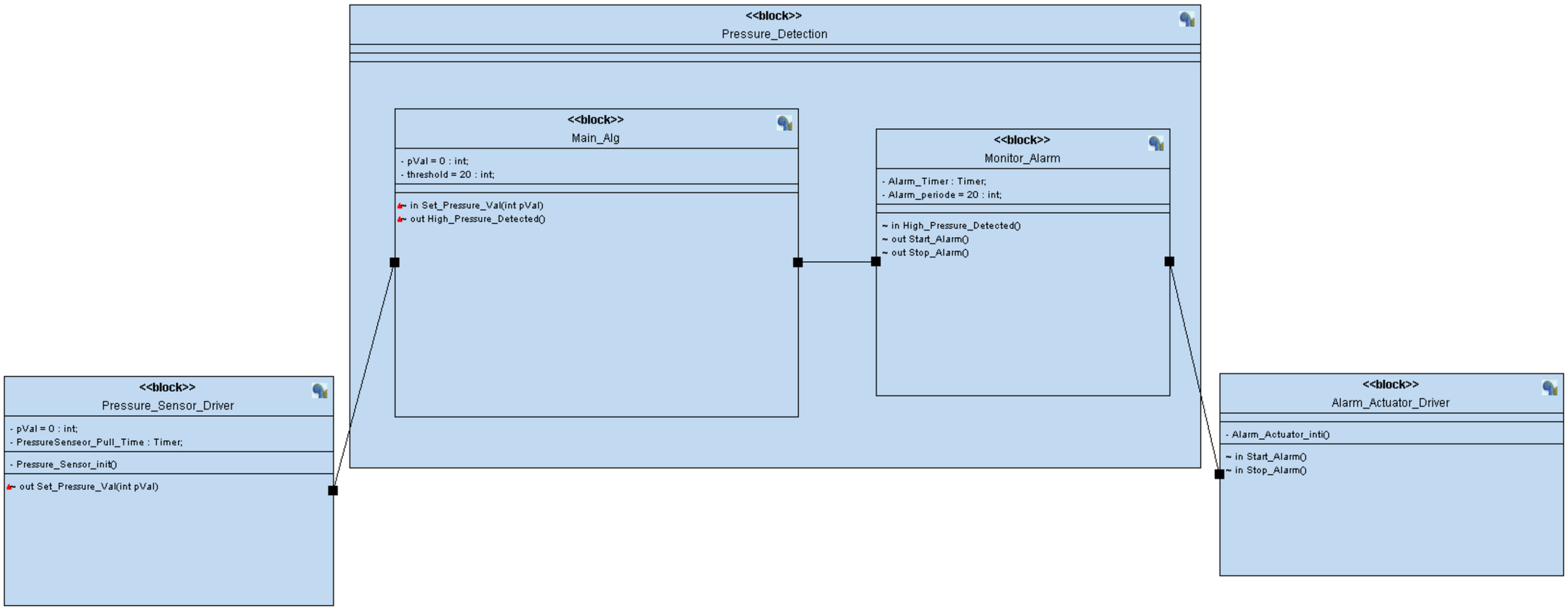
Back to Agenda

5 System Analysis : Sequence Diagram



Back to Agenda

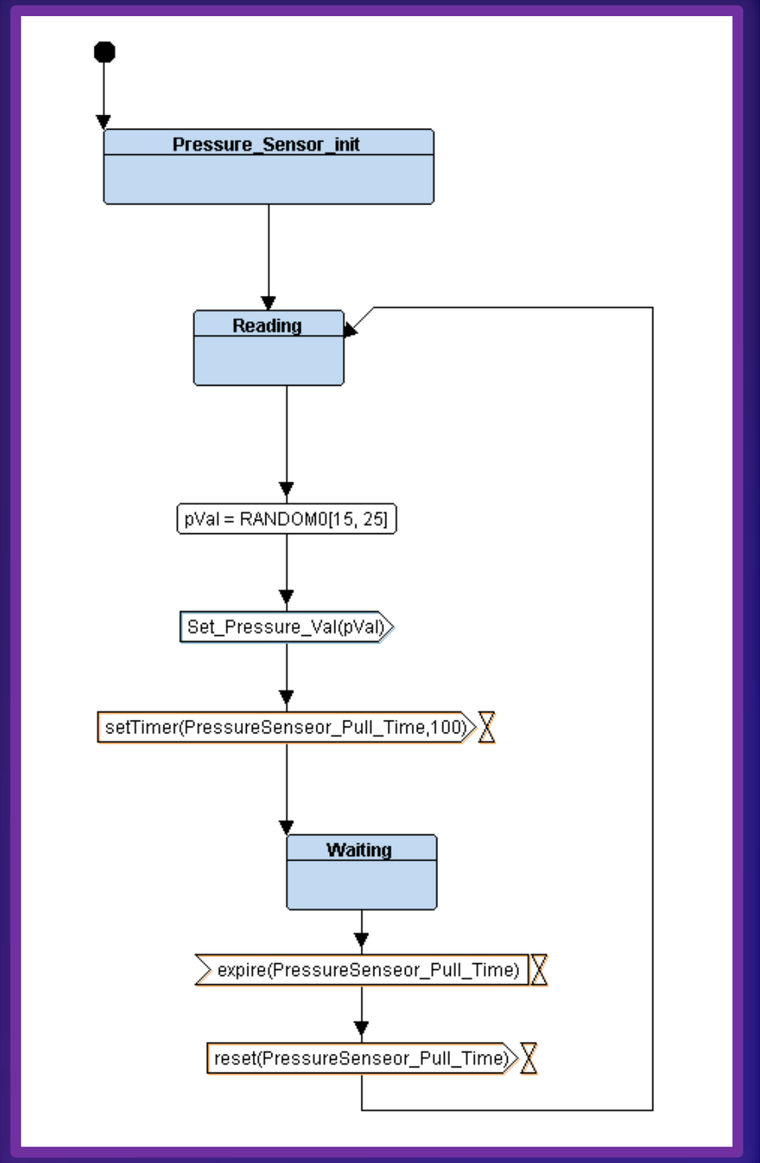
6 System Design



Back to Agenda

6 System Design

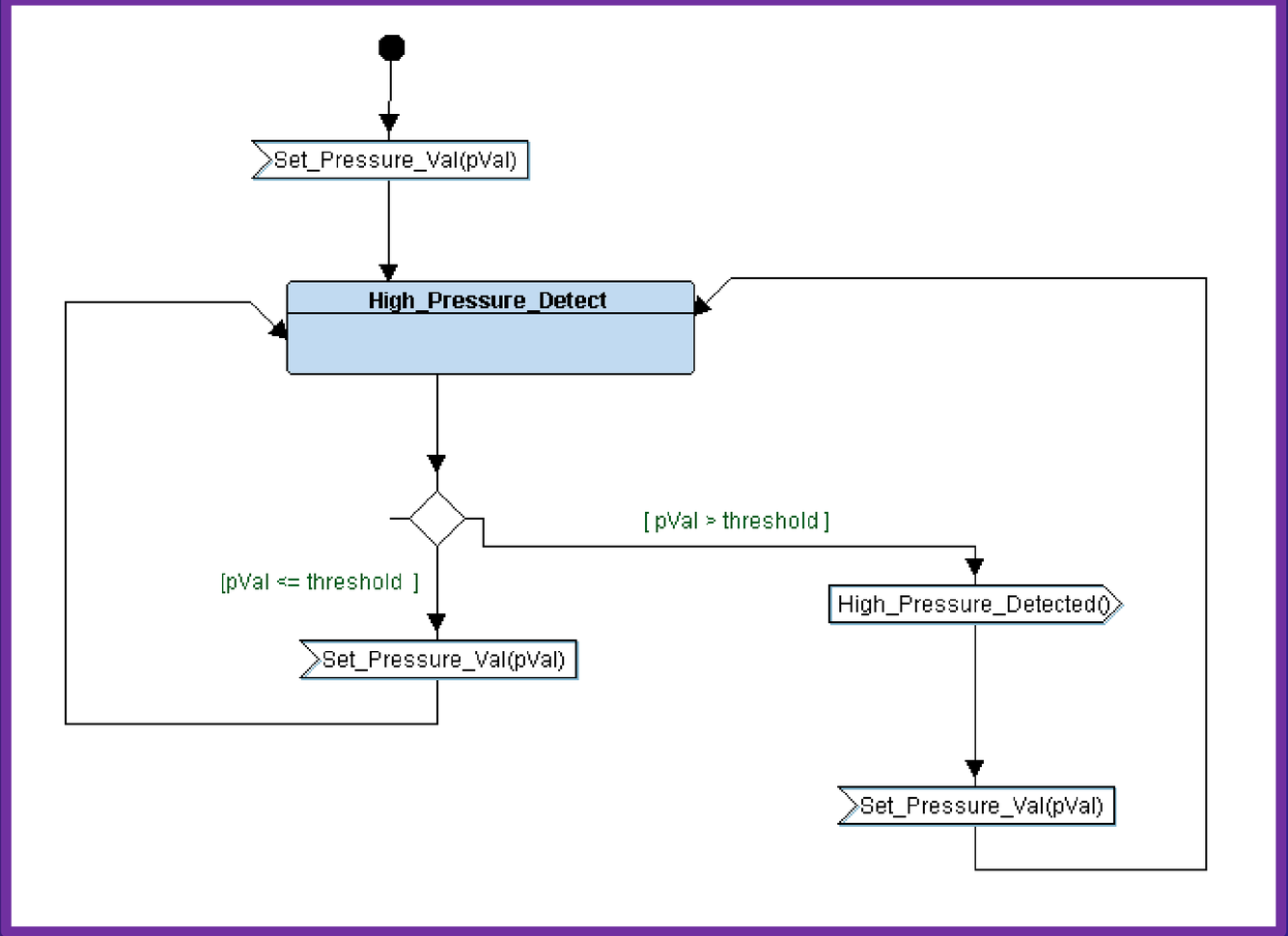
1 # Pressure Sensor



Back to Agenda

6 System Design

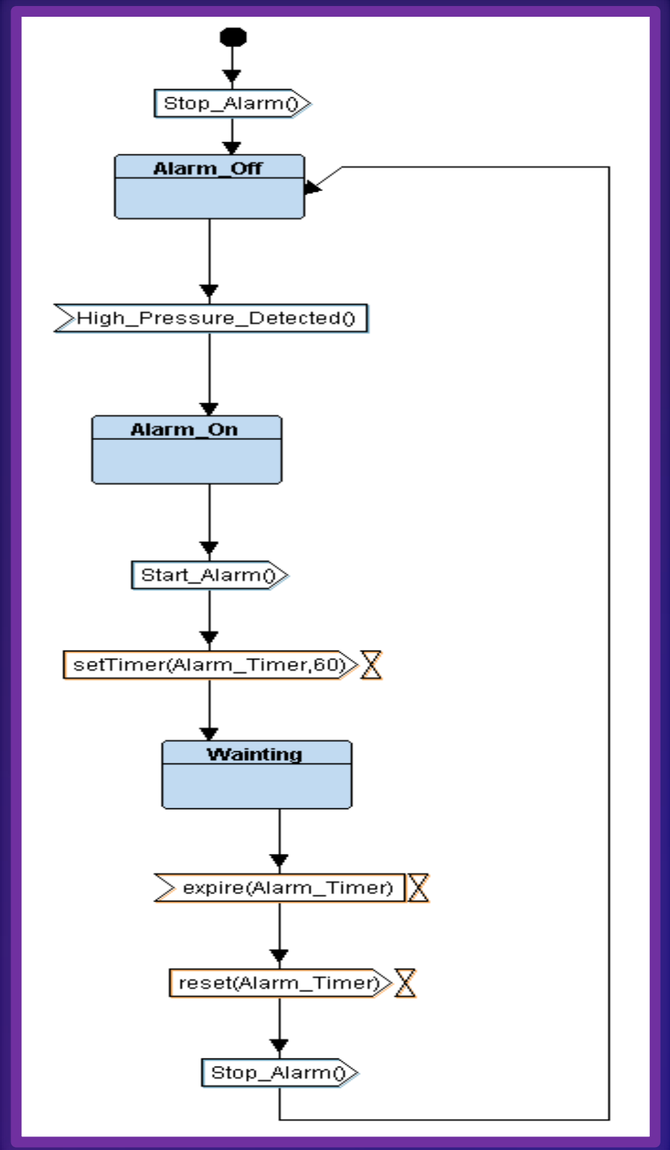
2 Main Algorithm



Back to Agenda

6 System Design

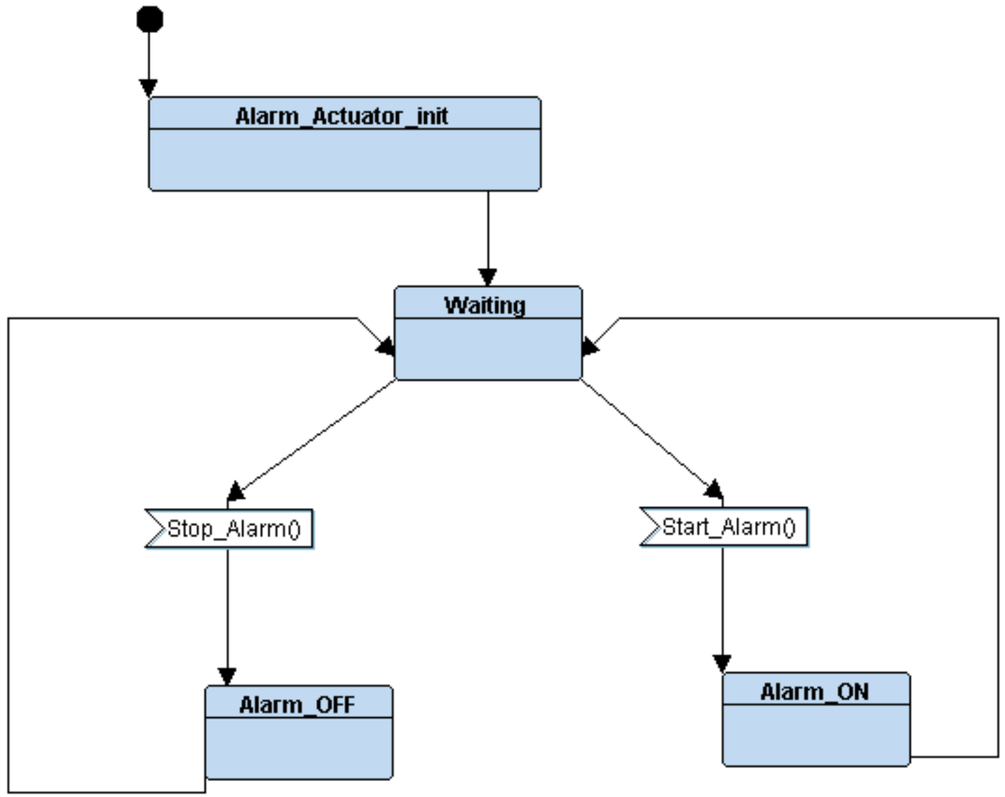
3 Monitor Alarm



Back to Agenda

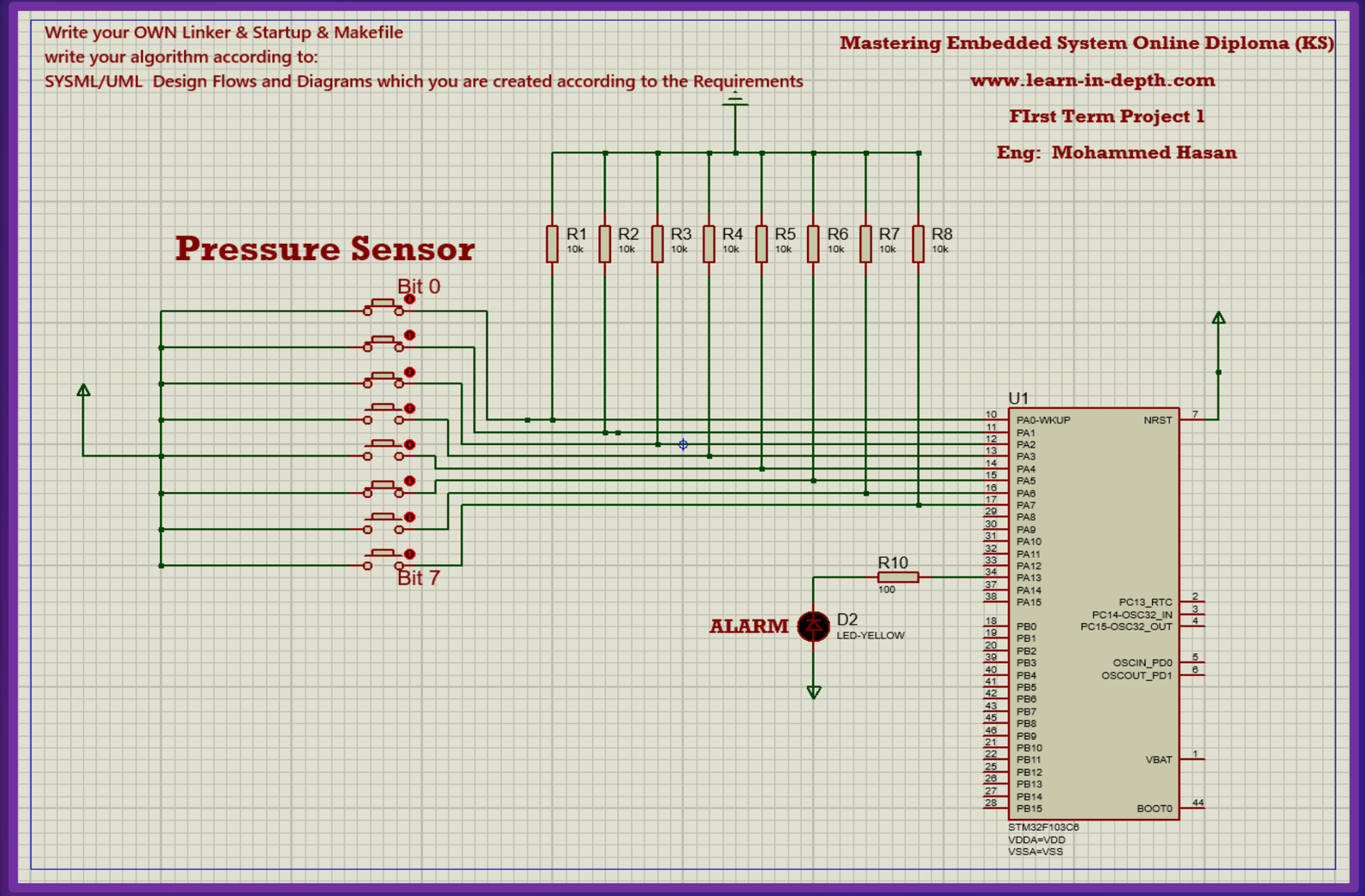
6 System Design

4 Alarm Actuator



Back to Agenda

7 System Design Simulation



Back to Agenda

8 Source Code

1 State

```h
/*
 ================================================================================
 Name        : State.h
 Author      : Mohammed Hasan
 Created on  : Oct 10, 2023
 Description : First Term Project 1 ==>> Pressure Controller
 ================================================================================
 */
#ifndef STATE_H_
#define STATE_H_

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

/*=========== Automatic State Function Generated ===========*/
#define State_Define(_STFUNC_) void ST_##_STFUNC_()
#define State(_STFUNC_) ST_##_STFUNC_

// States Collection
void Set_Pressure_Value(int Pressure_Value);
void High_Pressure_Detected();
void Start_Alarm();
void Stop_Alarm();

#endif /* STATE_H_ */
```

Back to Agenda

8 Source Code

2 Pressure Sensor

```h
/*
 ================================================================================
 Name        : Pressure_Sensor.h
 Author      : Mohammed Hasan
 Created on  : Oct 10, 2023
 Description : First Term Project 1 ==>> Pressure Controller
 ================================================================================
 */
#ifndef PRESSURE_SENSOR_DRIVER_H_
#define PRESSURE_SENSOR_DRIVER_H_
#include "State.h"

/*============= Define States =============*/
enum
{
    PS_Reading ,
    PS_Waiting
}PS_State_id;

void PS_init();
// Declare State Function
State_Define(PS_Reading);
State_Define(PS_Waiting);

// extern State pointer
extern void (*PS_State)() ;

#endif /*PRESSURE_SENSOR_DRIVER_H_*/
```

```c
/*
 ================================================================================
 Name        : Pressure_Sensor.c
 Author      : Mohammed Hasan
 Created on  : Oct 10, 2023
 Description : First Term Project 1 ==>> Pressure Controller
 ================================================================================
 */
#include "Pressure_Sensor.h"
/*========= Variables =========*/
unsigned int PS_Value = 0;
unsigned int PS_Pull_Time = 100;

// State Pointer to Function
void (*PS_State)();

void PS_init()
{
    // Intialize the Pressure Sensor
}

State_Define(PS_Reading)
{
    // State Name
    PS_State_id = PS_Reading;
    // Read from Pressure Sensor
    PS_Value = getPressureVal();
    Set_Pressure_Value(PS_Value);
    PS_State = State(PS_Waiting);
}

State_Define(PS_Waiting)
{
    // State Name
    PS_State_id = PS_Waiting;
    Delay(50000);
    PS_State = State(PS_Reading);
}
```

Back to Agenda

8 Source Code

3

Main Algorithm

```h
/*
 ================================================================================
 Name        : Main_Algorithm.h
 Author      : Mohammed Hasan
 Created on  : Oct 10, 2023
 Description : First Term Project 1 ==>> Pressure Controller
 ================================================================================
 */

#ifndef MAIN_ALGORITHM_H_
#define MAIN_ALGORITHM_H_
#include "State.h"

/*============= Define States =============*/
enum
{
    MA_High_Pressure_Detect
}MA_State_id ;

// Declare State Function
State_Define(MA_High_Pressure_Detect);

// extern pointer to function MA_State
extern void (*MA_State)() ;

#endif /*MAIN_ALGORITHM_H_*/
```

```c
/*
 ================================================================================
 Name        : Main_Algorithm.c
 Author      : Mohammed Hasan
 Created on  : Oct 10, 2023
 Description : First Term Project 1 ==>> Pressure Controller
 ================================================================================
 */
#include "Main_Algorithm.h"
/*========= Variables =========*/
unsigned int MA_Pressure_Value = 0 ;
unsigned int MA_Pressure_Threshold = 20 ;

// State Pointer to Function
void (*MA_State)() ;

void Set_Pressure_Value(int Pressure_Value)
{
    MA_Pressure_Value = Pressure_Value ;
    MA_State = State(MA_High_Pressure_Detect);
}
// Declare State Function
State_Define(MA_High_Pressure_Detect)
{
    // State Name
    MA_State_id = MA_High_Pressure_Detect ;

    //State Action
    if(MA_Pressure_Value <= MA_Pressure_Threshold)
    {
        MA_State = State(MA_High_Pressure_Detect);
    }
    else
    {
        High_Pressure_Detected();
        MA_State = State(MA_High_Pressure_Detect);
    }
}
```

Back to Agenda

8 Source Code

4

Monitor Alarm

```h
/*
 ================================================================================
 Name        : Monitor_Alarm.h
 Author      : Mohammed Hasan
 Created on  : Oct 10, 2023
 Description : First Term Project 1 ==>> Pressure Controller
 ================================================================================
 */
#ifndef MONITOR_ALARM_H_
#define MONITOR_ALARM_H_
#include "State.h"

/*============= Define States =============*/
enum
{
    MoA_Alarm_OFF ,
    MoA_Alarm_ON  ,
    MoA_Waiting
}MoA_State_id;

// Declare State Function
State_Define(MoA_Alarm_OFF);
State_Define(MoA_Alarm_ON);
State_Define(MoA_Waiting);

// extern pointer
extern void(*MoA_State)();

#endif /*MONITOR_ALARM_H_*/
```

```c
/*
 ================================================================================
 Name        : Monitor_Alarm.c
 Author      : Mohammed Hasan
 Created on  : Oct 10, 2023
 Description : First Term Project 1 ==>> Pressure Controller
 ================================================================================
 */
#include "Monitor_Alarm.h"

/*========== Variables ==========*/
unsigned int MoA_Periode = 50000;

// State Pointer to Function
void(*MoA_State)();

void High_Pressure_Detected()
{
    MoA_State = State(MoA_Alarm_ON);
}

State_Define(MoA_Alarm_OFF)
{
    // State Name
    MoA_State_id = MoA_Alarm_OFF ;
    Stop_Alarm();
    MoA_State = State(MoA_Alarm_OFF);
}
State_Define(MoA_Alarm_ON)
{
    // State Name
    MoA_State_id = MoA_Alarm_ON ;
    Start_Alarm();
    MoA_State = State(MoA_Waiting);
}
State_Define(MoA_Waiting)
{
    // State Name
    MoA_State_id = MoA_Waiting ;
    Stop_Alarm();
    Delay(MoA_Periode);
    MoA_State = State(MoA_Alarm_OFF);
}
```

Back to Agenda

8 Source Code

5

Alarm Actuator

```c
/*
 ================================================================================
 Name        : Alarm_Actuator.h
 Author      : Mohammed Hasan
 Created on  : Oct 10, 2023
 Description : First Term Project 1 ==>> Pressure Controller
 ================================================================================
 */
#ifndef ALARM_ACTUATOR_DRIVER_H_
#define ALARM_ACTUATOR_DRIVER_H_
#include "State.h"
/*============= Define States =============*/
enum
{
    AC_Waiting   ,
    AC_Alarm_OFF ,
    AC_Alarm_ON
}AC_State_id;

void AC_init();
// Declare State Function
State_Define(AC_Waiting);
State_Define(AC_Alarm_OFF);
State_Define(AC_Alarm_ON);

// extern State pointer
extern void(*AC_State)();
#endif /*ALARM_ACTUATOR_DRIVER_H_*/
```

```c
/*
 ================================================================================
 Name        : Alarm_Actuator.c
 Author      : Mohammed Hasan
 Created on  : Oct 10, 2023
 Description : First Term Project 1 ==>> Pressure Controller
 ================================================================================
 */
#include "Alarm_Actuator.h"
void(*AC_State)();

void AC_init()
{
    // intialize Alarm_Actuator Driver
}

void Stop_Alarm()
{
    AC_State = State(AC_Alarm_OFF);
}

void Start_Alarm()
{
    AC_State = State(AC_Alarm_ON);
}

State_Define(AC_Waiting)
{
    //State Name
    AC_State_id = AC_Waiting;
    AC_State = State(AC_Waiting);
}

State_Define(AC_Alarm_ON)
{
    //State Name
    AC_State_id = AC_Alarm_ON;
    Set_Alarm_actuator(0);
    AC_State = State(AC_Waiting);
}

State_Define(AC_Alarm_OFF)
{
    //State Name
    AC_State_id = AC_Alarm_OFF;
    Set_Alarm_actuator(1);
    AC_State = State(AC_Waiting);
}
```

Back to Agenda

8 Source Code

6

Driver

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

#define SET_BIT(ADDRESS,BIT)   ADDRESS |=  (1<<BIT)
#define RESET_BIT(ADDRESS,BIT) ADDRESS &= ~(1<<BIT)
#define TOGGLE_BIT(ADDRESS,BIT)  ADDRESS ^=  (1<<BIT)
#define READ_BIT(ADDRESS,BIT) ((ADDRESS) &   (1<<(BIT)))


#define GPIO_PORTA 0x40010800
#define BASE_RCC   0x40021000

#define APB2ENR   *(volatile uint32_t *)(BASE_RCC + 0x18)

#define GPIOA_CRL *(volatile uint32_t *)(GPIO_PORTA + 0x00)
#define GPIOA_CRH *(volatile uint32_t *)(GPIO_PORTA + 0X04)
#define GPIOA_IDR *(volatile uint32_t *)(GPIO_PORTA + 0x08)
#define GPIOA_ODR *(volatile uint32_t *)(GPIO_PORTA + 0x0C)


void Delay(int nCount);
int getPressureVal();
void Set_Alarm_actuator(int i);
void GPIO_INITIALIZATION ();
```

```c
#include "driver.h"
#include <stdint.h>
#include <stdio.h>
void Delay(int nCount)
{
    for(; nCount != 0; nCount--);
}

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

void Set_Alarm_actuator(int i){
    if (i == 1){
        SET_BIT(GPIOA_ODR,13);
    }
    else if (i == 0){
        RESET_BIT(GPIOA_ODR,13);
    }
}

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

Back to Agenda

8 Source Code

7

Main

```c
/*
 ================================================================================
 Name        : main.c
 Author      : Mohammed Hasan
 Created on  : Oct 10, 2023
 Description : First Term Project 1 ==>> Pressure Controller
 ================================================================================
 */
 /*
  * PS  ===>> Pressure_Sensor
  * AC  ===>> Alarm_Actuator
  * MA  ===>> Main_Algorithm
  * MoA ===>> Monitor_Alarm
  */
#include <stdint.h>
#include <stdio.h>

#include "driver.h"
#include "Pressure_Sensor.h"
#include "Main_Algorithm.h"
#include "Monitor_Alarm.h"
#include "Alarm_Actuator.h"

void Setup()
{
    /*
     * init all the drivers
     * init IRQ...
     * init HAL US_Driver DC_Driver
     * init Block
     * set States Pointers for each Block
     */
    PS_init();
    AC_init();
    PS_State = State(PS_Reading);
    MA_State = Set_Pressure_Value;
    MoA_State =State(MoA_Alarm_OFF);
    AC_State = State(AC_Waiting);
}

int main (){
    GPIO_INITIALIZATION();
    Setup();
    while (1)
    {
        PS_State();
        MA_State();
        MoA_State();
        AC_State();
    }
}
```

Back to Agenda

8 Source Code

8

Startup

```c
/*================================================================================
 Name        : startup.c
 Author      : Mohammed Hasan
 Created on  : Oct 10, 2023
 Description : First Term Project 1 ==>> Pressure Controller
 ================================================================================*/
#include "Platform_Types.h"

extern int main(void);
extern uint32_t _Stack_Top ;
extern uint32_t _E_Text ;
extern uint32_t _S_Data ;
extern uint32_t _E_Data ;
extern uint32_t _S_Bss ;
extern uint32_t _E_Bss ;

void Reset_Handler(void);

void Default_Handler()
{
    Reset_Handler();
}

void NMI_Handler(void)          __attribute__ ((weak, alias("Default_Handler")));
void H_Fault_Handler(void)      __attribute__ ((weak, alias("Default_Handler")));
void MM_Fault_Handler(void)     __attribute__ ((weak, alias("Default_Handler")));
void Bus_Fault(void)            __attribute__ ((weak, alias("Default_Handler")));
void Usage_Fault_Handler(void)  __attribute__ ((weak, alias("Default_Handler")));


uint32_t vectors[] __attribute__ ((section(".vectors"))) =
{
    (uint32_t) &_Stack_Top ,
    (uint32_t) &Reset_Handler ,
    (uint32_t) &NMI_Handler ,
    (uint32_t) &H_Fault_Handler ,
    (uint32_t) &MM_Fault_Handler ,
    (uint32_t) &Bus_Fault ,
    (uint32_t) &Usage_Fault_Handler
};

void Reset_Handler(void)
{
    //Copy Data From Rom to Ram
    uint32_t Data_Size = (uint8_t*)&_E_Data - (uint8_t*)&_S_Data ;
    uint8_t* P_src = (uint8_t*)&_E_Text ;
    uint8_t* P_dst = (uint8_t*)&_S_Data ;

    for(int i =0 ; i < Data_Size ; i++)
    {
        *((uint8_t*)P_dst++) = *((uint8_t*)P_src++) ;
    }

    // init the .bss with 0

    uint32_t Bss_Size = (uint8_t*)&_E_Bss - (uint8_t*)&_S_Bss ;
    P_dst = (uint8_t*)&_S_Bss ;

    for(int i =0 ; i < Bss_Size ; i++)
    {
        *((uint8_t*)P_dst++) = (uint8_t)0 ;
    }
    // jump to main()
    main();
}
```

Back to Agenda

8 Source Code

9

LinkerScript

```
/*=============================================================================
 Name        : linker_script.ld
 Author      : Mohammed Hasan
 Created on  : Oct 10, 2023
 Description : First Term Project 1 ==>> Pressure Controller
 ============================================================================*/
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*)
        . = ALIGN(4);
        _E_Bss = . ;

        . = ALIGN(4);
        . = . + 0x1000 ;
        _Stack_Top = . ;
    }>sram
}
```

Back to Agenda

8 Source Code

10

Makefile

```
#Learn-In-Depth
#Eng.Mohammed_Hasan
#Unit 5 ==>> First Term Project 1 ==>> Pressure Controller
#makefile
CC=arm-none-eabi-
CFLAGS=-mcpu=cortex-m3 -gdwarf-2
INCS=-I .
LIBS=
SRC=$(wildcard *.c)
OBJ=$(SRC:.c=.o)
As=$(wildcard *.s)
AsOBJ=$(As:.s=.o)
Project_name=Pressure_Controller

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

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

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

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

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

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

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

Back to Agenda

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  Algn
  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  Algn
  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
```

Back to Agenda

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  Algn
  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  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  Algn
  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    00000168  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
```

Back to Agenda

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  Algn
  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  Algn
  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
```

Back to Agenda

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  Algn
  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  Algn
  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
```

Back to Agenda

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
0000001c 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
```

Back to Agenda

10 Symbol Table for object files

5 Startup

```
$ arm-none-eabi-nm.exe startup.o
         U _E_Bss
         U _E_Data
         U _E_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
```

Back to Agenda

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

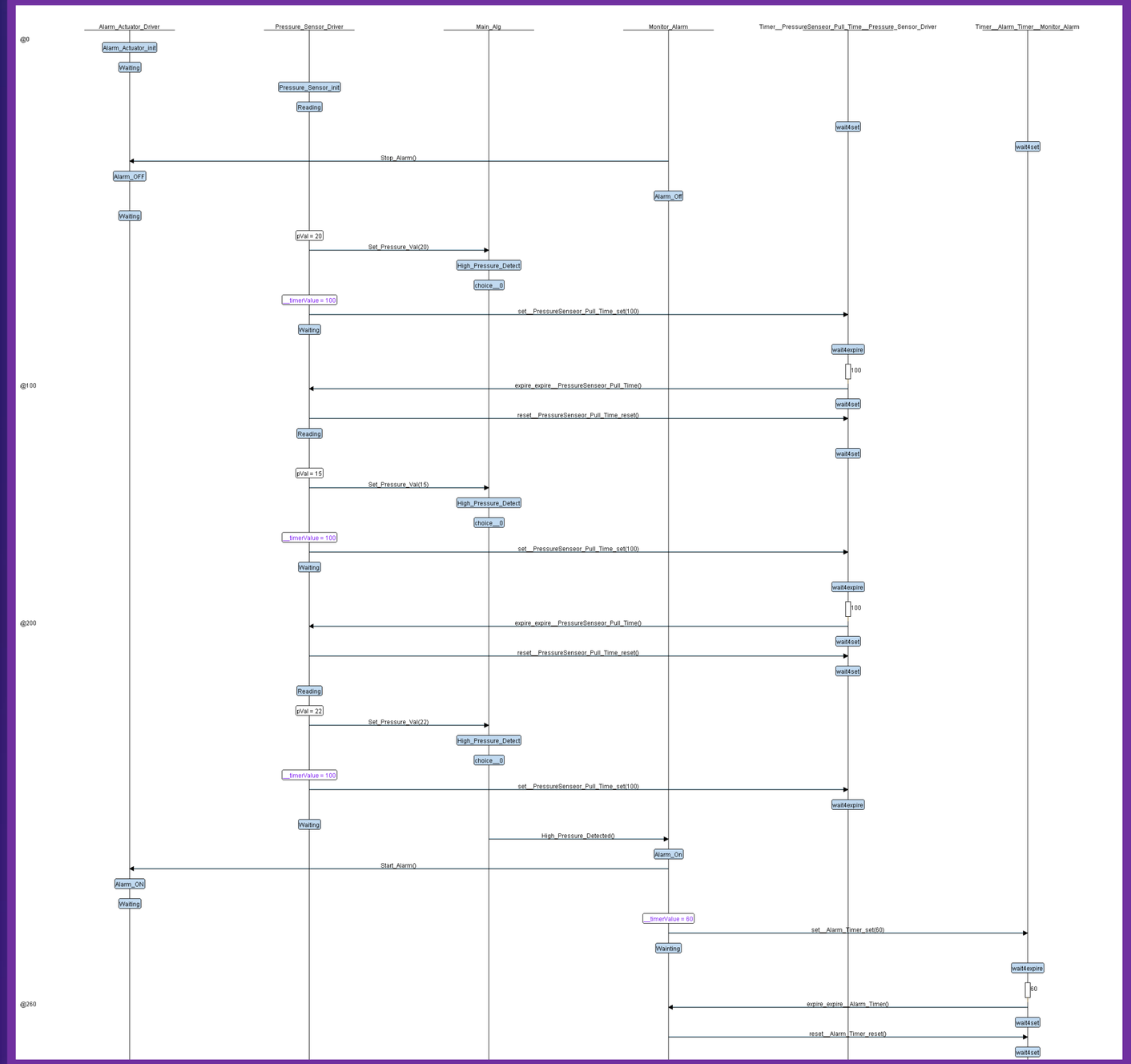
11 Size Sections

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

Back to Agenda

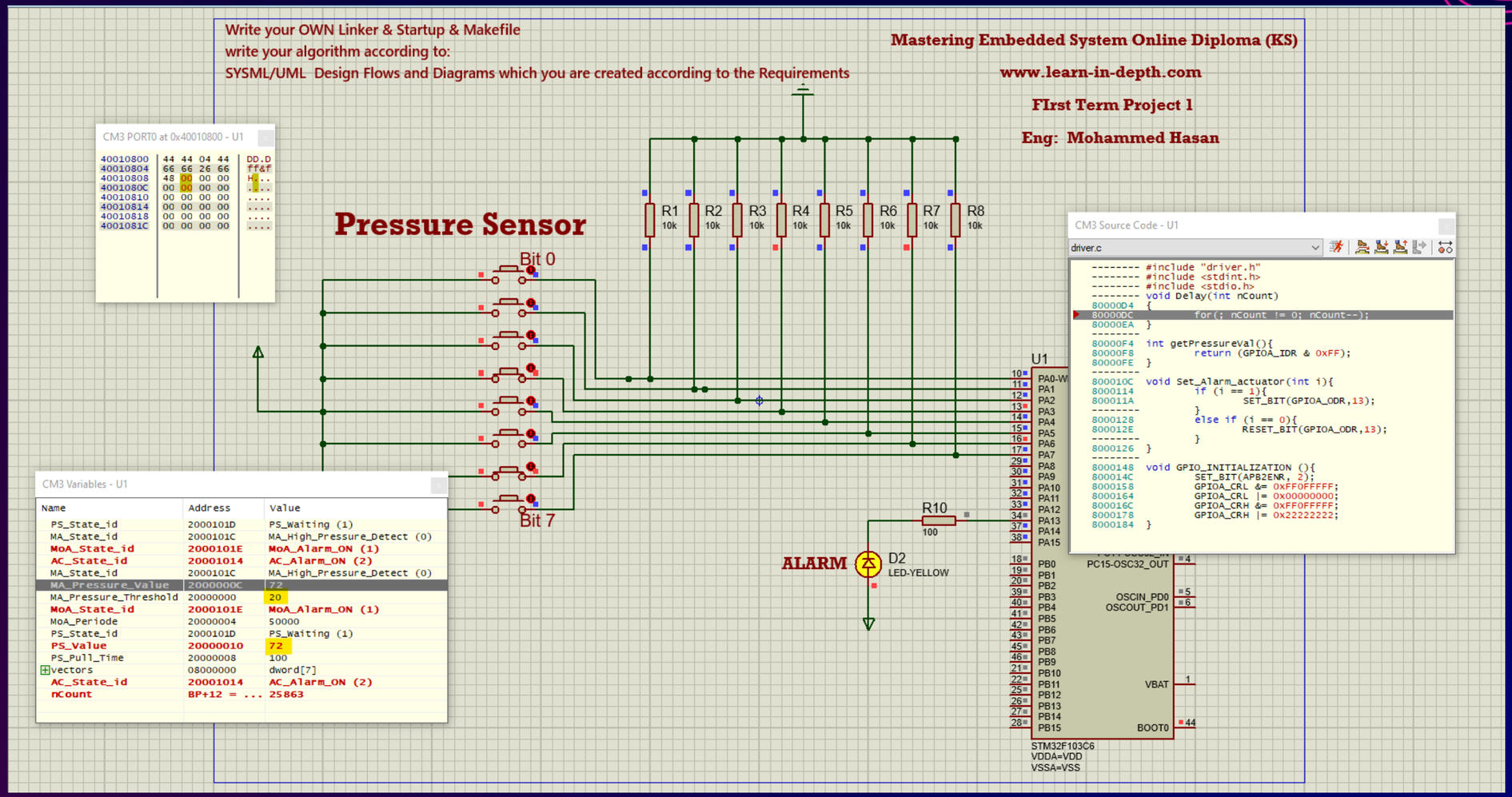
Learn-In-Depth

12 Logic Verification



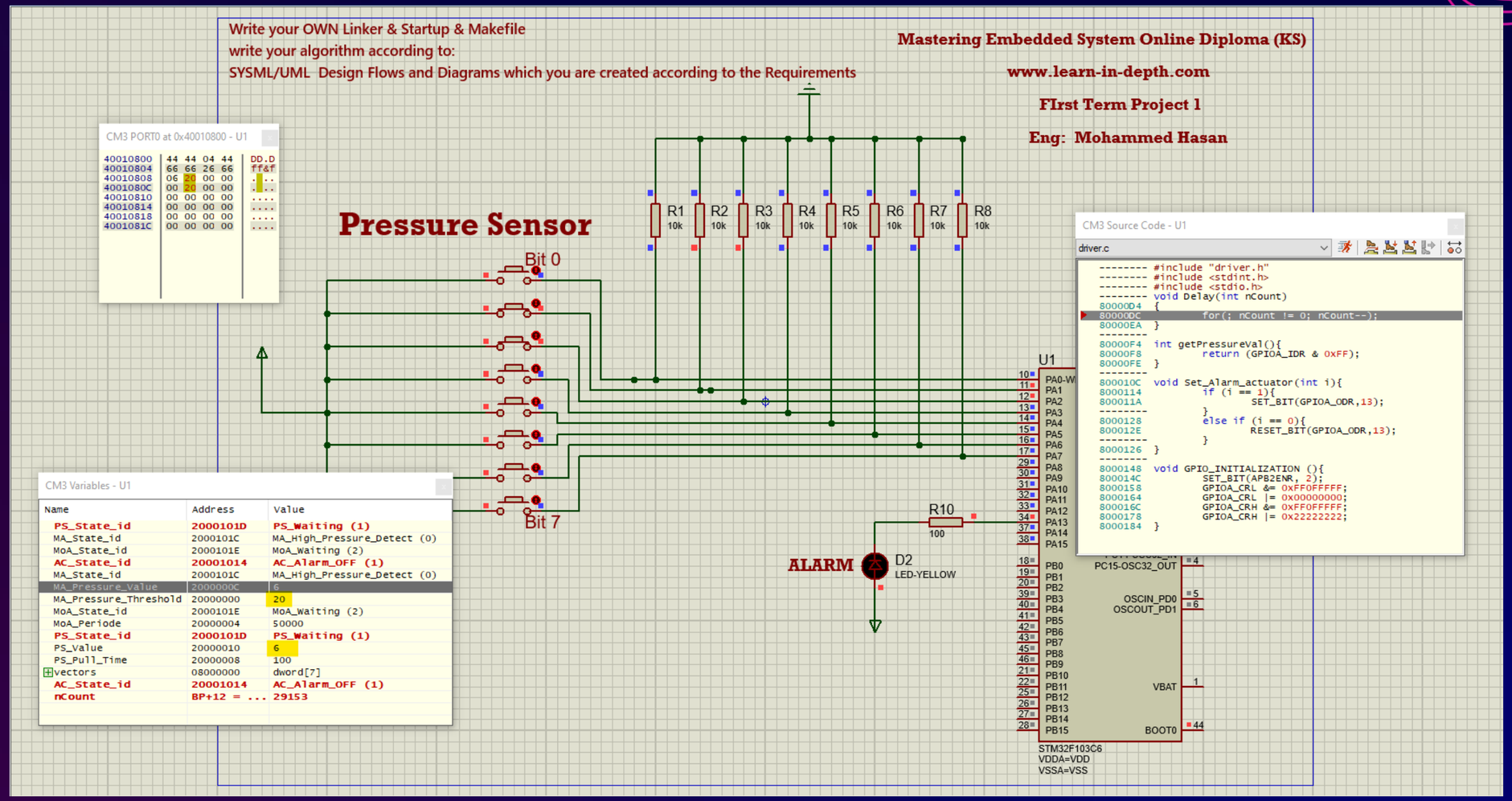
Back to Agenda

13 Simulation Output



Back to Agenda

13 Simulation Output



Back to Agenda

13 Simulation Video

You can see it in my Drive: click here

Back to Agenda

Learn-In-Depth

THANK YOU

mohasanbder@gmail.com

My Progress profile: Click Here

My Drive: Click Here

GitHub: Click Here