Learning Report – Embedded C – Hardware + Programming + Testing

GLOBAL ENGINEERING ACADEMY

L&T Technology Services

Document History

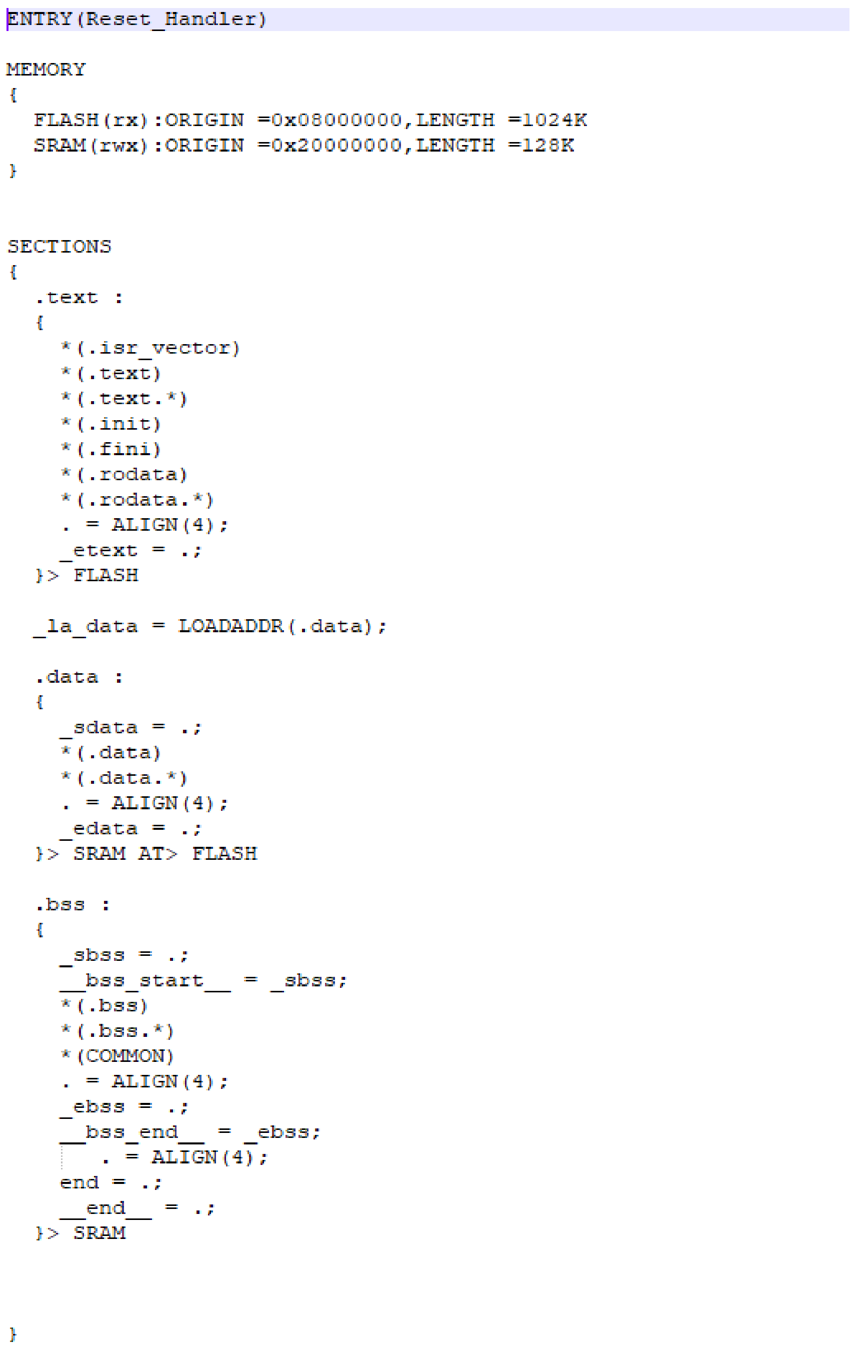
Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	Approved By	Remarks/Revision Details
1.0	24-12-20	Reethu M		Dr. Vivek K and Bhargav N	
1.1	28-12-20	Reethu M		Dr. Vivek K and Bhargav N	
1.2	29-12-20	Reethu M		Dr. Vivek K and Bhargav N	

Contents

Table of Figures

1. Activity 1

Linker Script

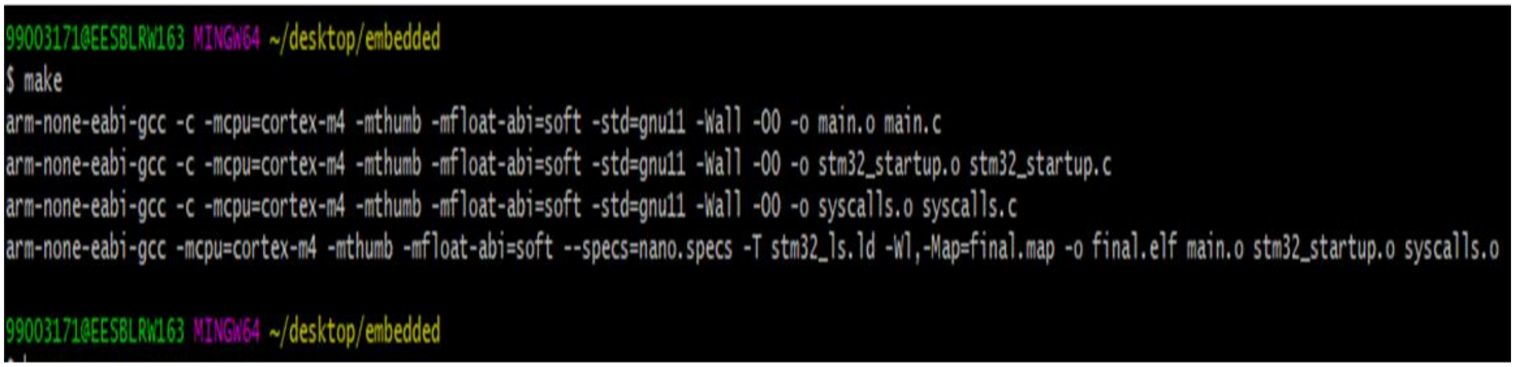


```
ENTRY(Reset_Handler)

MEMORY
{
  FLASH(rx):ORIGIN =0x08000000,LENGTH =1024K
  SRAM(rwx):ORIGIN =0x20000000,LENGTH =128K
}


SECTIONS
{
  .text :
  {
    *(.isr_vector)
    *(.text)
    *(.text.*)
    *(.init)
    *(.fini)
    *(.rodata)
    *(.rodata.*)
    . = ALIGN(4);
    _etext = .;
  }> FLASH

  _la_data = LOADADDR(.data);

  .data :
  {
    _sdata = .;
    *(.data)
    *(.data.*)
    . = ALIGN(4);
    _edata = .;
  }> SRAM AT> FLASH

  .bss :
  {
    _sbss = .;
    __bss_start__ = _sbss;
    *(.bss)
    *(.bss.*)
    *(COMMON)
    . = ALIGN(4);
    _ebss = .;
    __bss_end__ = _ebss;
       . = ALIGN(4);
    end = .;
    __end__ = .;
  }> SRAM


}
```

Make file

Main.c code

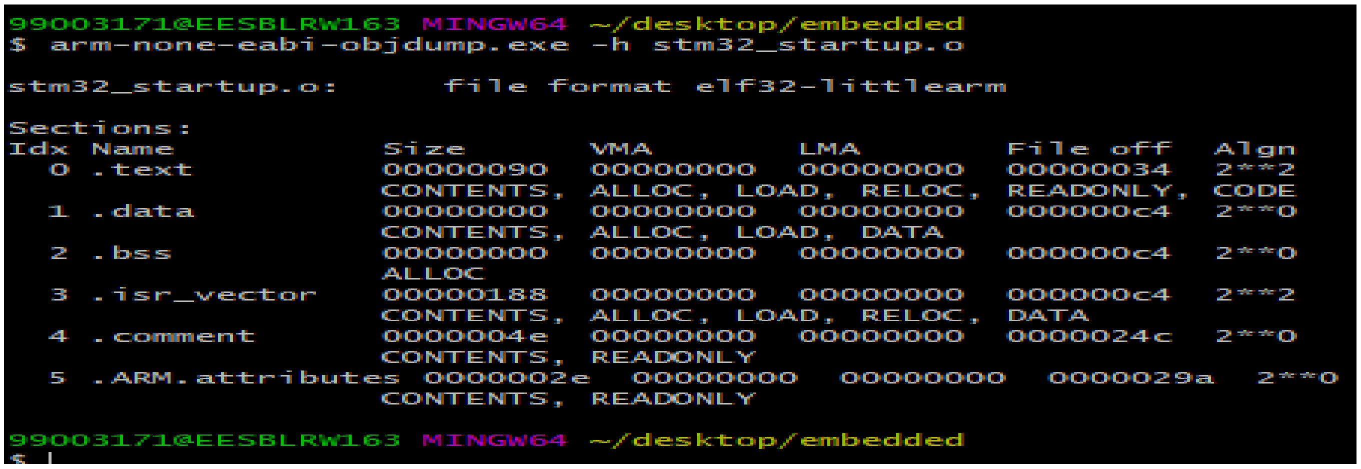
```
#include<stdio.h>
int main(){
//printf("hello world\n");
return 0;
}
```



```
99003171@EESBLRW163 MINGW64 ~/desktop/embedded
$ make
arm-none-eabi-gcc -c -mcpu=cortex-m4 -mthumb -mfloat-abi=soft -std=gnu11 -Wall -O0 -o main.o main.c
arm-none-eabi-gcc -c -mcpu=cortex-m4 -mthumb -mfloat-abi=soft -std=gnu11 -Wall -O0 -o stm32_startup.o stm32_startup.c
arm-none-eabi-gcc -c -mcpu=cortex-m4 -mthumb -mfloat-abi=soft -std=gnu11 -Wall -O0 -o syscalls.o syscalls.c
arm-none-eabi-gcc -mcpu=cortex-m4 -mthumb -mfloat-abi=soft --specs=nano.specs -T stm32_ls.ld -Wl,-Map=final.map -o final.elf main.o stm32_startup.o syscalls.o

99003171@EESBLRW163 MINGW64 ~/desktop/embedded
```

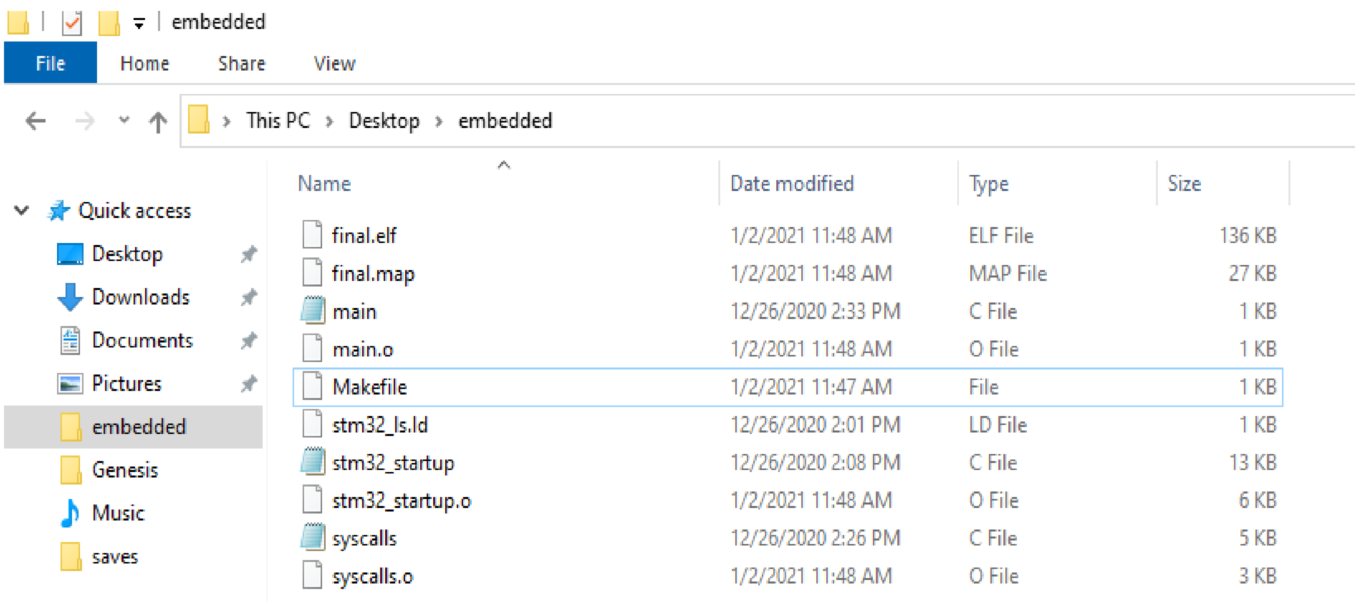
Figure 1 Makefile

Startup



```
99003171@EESBLRW163 MINGW64 ~/desktop/embedded
$ arm-none-eabi-objdump.exe -h stm32_startup.o

stm32_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 .isr_vector   00000188  00000000  00000000  000000c4  2**2
                  CONTENTS, ALLOC, LOAD, RELOC, DATA
  4 .comment      0000004e  00000000  00000000  0000024c  2**0
                  CONTENTS, READONLY
  5 .ARM.attributes 0000002e  00000000  00000000  0000029a  2**0
                  CONTENTS, READONLY

99003171@EESBLRW163 MINGW64 ~/desktop/embedded
$ |
```

Figure 2 Startup



Name	Date modified	Type	Size
final.elf	1/2/2021 11:48 AM	ELF File	136 KB
final.map	1/2/2021 11:48 AM	MAP File	27 KB
main	12/26/2020 2:33 PM	C File	1 KB
main.o	1/2/2021 11:48 AM	O File	1 KB
Makefile	1/2/2021 11:47 AM	File	1 KB
stm32_ls.ld	12/26/2020 2:01 PM	LD File	1 KB
stm32_startup	12/26/2020 2:08 PM	C File	13 KB
stm32_startup.o	1/2/2021 11:48 AM	O File	6 KB
syscalls	12/26/2020 2:26 PM	C File	5 KB
syscalls.o	1/2/2021 11:48 AM	O File	3 KB

Figure 3 Output files

2. Activity 2

Header File

2.1 MCU Specific Header File

```
/*
 * STM32F4xx.h => MCU specific header file =>driver level
 *
 * Created on: Dec 28, 2020
 *     Author: 99003171
 */

#ifndef INC_STM32F4XX_H_
#define INC_STM32F4XX_H_
#include<stdint.h>
#define __vo volatile

//defining macros for the various memories

#define FLASHADDR 0x08000000U
#define SRAM1ADDR 0x20000000U
#define SRAM2ADDR 0x2001C000U
#define SRAMADDR SRAM1ADDR //same as SRAM1
#define ROM_BASEADDR 0x1FFF0000U

//defining macros for the various buses

#define APB1_BASEADDR   0x40000000U
#define APB2_BASEADDR   0x40010000U
#define AHB1_BASEADDR   0x40020000U
#define AHB2_BASEADDR   0x50000000U
#define PERI_BASEADDR APB1_BASEADDR  //peripheral base address
//#define AHB3ADDR 0x60000000U

//defining macros for peripherals hanging on to AHB1 bus

#define GPIOA_BASEADDR (AHB1_BASEADDR + (0x0000U)) //GPIO Peripherals
#define GPIOB_BASEADDR (AHB1_BASEADDR + (0x0400U))
#define GPIOC_BASEADDR (AHB1_BASEADDR + (0x0800U))
#define GPIOD_BASEADDR (AHB1_BASEADDR + (0x0C00U))
#define GPIOE_BASEADDR (AHB1_BASEADDR + (0x1000U))
#define GPIOF_BASEADDR (AHB1_BASEADDR + (0x1400U))
#define GPIOG_BASEADDR (AHB1_BASEADDR + (0x1800U))
#define GPIOH_BASEADDR (AHB1_BASEADDR + (0x1C00U))
#define GPIOI_BASEADDR (AHB1_BASEADDR + (0x2000U))
#define RCC_BASEADDR (AHB1_BASEADDR + (0x3800U))

//defining macros for peripherals hanging on to AHB2 bus

//NONE

//defining macros for peripherals hanging on to APB1 bus
```

```
#define SPI2_BASEADDR (APB1_BASEADDR + (0x3800U)) //SPI peripheral
#define SPI3_BASEADDR (APB1_BASEADDR + (0x3C00U))

#define USART2_BASEADDR (APB1_BASEADDR + (0x4400U)) //USART peripheral
#define USART3_BASEADDR (APB1_BASEADDR + (0x4800U))

#define UART4_BASEADDR (APB1_BASEADDR + (0x4C00U)) //UART peripheral
#define UART5_BASEADDR (APB1_BASEADDR + (0x5000U))

#define I2C1_BASEADDR    (APB1_BASEADDR + (0x5400U)) //I2C peripheral
#define I2C2_BASEADDR    (APB1_BASEADDR + (0x5800U))
#define I2C3_BASEADDR    (APB1_BASEADDR + (0x5C00U))

#define TIM2_BASEADDR (APB1_BASEADDR + (0x0000U)) //Timer peripheral
#define TIM3_BASEADDR (APB1_BASEADDR + (0x0400U))
#define TIM4_BASEADDR (APB1_BASEADDR + (0x0800U))
#define TIM5_BASEADDR (APB1_BASEADDR + (0x0C00U))
#define TIM6_BASEADDR (APB1_BASEADDR + (0x1000U))
#define TIM7_BASEADDR (APB1_BASEADDR + (0x1400U))
#define TIM12_BASEADDR (APB1_BASEADDR + (0x1800U))
#define TIM13_BASEADDR (APB1_BASEADDR + (0x1C00U))
#define TIM14_BASEADDR (APB1_BASEADDR + (0x2000U))

//defining macros for peripherals hanging on to APB2 bus

#define SPI1_BASEADDR (APB2_BASEADDR + (0x3000U)) //SPI peripheral
#define USART1_BASEADDR (APB2_BASEADDR + (0x1000U)) //USART peripheral
#define USART6_BASEADDR (APB2_BASEADDR + (0x0000U))

//#define ADC1-ADC2-ADC3_BASEADDR (APB2_BASEADDR + (0x2000U)) //ADC peripheral

#define TIM1_BASEADDR (APB2_BASEADDR + (0x2000U))// Timer peripheral
#define TIM8_BASEADDR (APB2_BASEADDR + (0x0400U))
#define TIM9_BASEADDR (APB2_BASEADDR + (0x4000U))
#define TIM10_BASEADDR (APB2_BASEADDR + (0x4400U))
#define TIM11_BASEADDR (APB2_BASEADDR + (0x4800U))


//defining macros for GPIO Peripheral registers

//#define GPIOA_ODR_BASEADDR      (AHB1_BASEADDR + (0x0000U) + (0x0014U))

typedef struct
{    // at port level definitions
__vo uint32_t  MODER; // Address offset: 0x00
__vo uint32_t  OTYPER; // Address offset: 0x04
__vo uint32_t  OSPEEDR; // Address offset: 0x08
__vo uint32_t  PUPDR;    // Address offset: 0x0C
__vo uint32_t  IDR; // Address offset: 0x10
__vo uint32_t  ODR; // Address offset: 0x14
__vo uint32_t  BSRRL; // Address offset: 0x18
__vo uint32_t  BSRRH; // Address offset: 0x1A
__vo uint32_t  LCKR; // Address offset: 0x1C
__vo uint32_t  AFR [2]; // AFR[0] - AFR Low registers, AFR[1] - AFR high registers   // Address offset: 0x20-0x24
} GPIO_Reg_def_t;

//Reg_def_t *pGPIOA = (Reg_def_t*)GPIOA_BASEADDR; //== #define GPIOA
((Reg_def_t*)GPIOA_BASEADDR)

#define GPIOA ((GPIO_Reg_def_t*)GPIOA_BASEADDR)
#define GPIOB ((GPIO_Reg_def_t*)GPIOB_BASEADDR)
#define GPIOC ((GPIO_Reg_def_t*)GPIOC_BASEADDR)
#define GPIOD ((GPIO_Reg_def_t*)GPIOD_BASEADDR)
#define GPIOE ((GPIO_Reg_def_t*)GPIOE_BASEADDR)
#define GPIOF ((GPIO_Reg_def_t*)GPIOF_BASEADDR)
#define GPIOG ((GPIO_Reg_def_t*)GPIOG_BASEADDR)
#define GPIOH ((GPIO_Reg_def_t*)GPIOH_BASEADDR)
#define GPIOI ((GPIO_Reg_def_t*)GPIOI_BASEADDR)

// defining macros for RCC peripheral registers

typedef struct
{
__vo uint32_t CR;
__vo uint32_t PLLCFGR;
__vo uint32_t CFGR;
__vo uint32_t CIR;
__vo uint32_t AHB1RSTR;
__vo uint32_t AHB2RSTR;
__vo uint32_t AHB3RSTR;
uint32_t RESERVED0;
__vo uint32_t APB1RSTR;
__vo uint32_t APB2RSTR;
uint32_t Reserved1[2];
__vo uint32_t AHB1ENR;
__vo uint32_t AHB2ENR;
__vo uint32_t AHB3ENR;
uint32_t RESERVED2;
__vo uint32_t APB1ENR;
__vo uint32_t APB2ENR;
uint32_t RESERVED3[2];
__vo uint32_t AHB1LPENR;
__vo uint32_t AHB2LPENR;
__vo uint32_t AHB3LPENR;
uint32_t RESERVED4;
__vo uint32_t APB1LPENR;
__vo uint32_t APB2LPENR;
uint32_t RESERVED5[2];
__vo uint32_t BDCR;
__vo uint32_t CSR;
uint32_t RESERVED6[2];
__vo uint32_t SSCGR;
__vo uint32_t PLLI2SCFGR;
__vo uint32_t PLLSAICFGR;
__vo uint32_t DCKCFGR;
```

```
} RCC_Reg_def_t;
/*
#define RCC1  ((RCC_Reg_def_t*)0x4002 3800)
#define RCC2  ((RCC_Reg_def_t*)0x4002 3800)
#define RCC3  ((RCC_Reg_def_t*)0x4002 3800)
#define RCC4  ((RCC_Reg_def_t*)0x4002 3800)
#define RCC5  ((RCC_Reg_def_t*)0x4002 3800)
#define RCC6  ((RCC_Reg_def_t*)0x4002 3800)
#define RCC7  ((RCC_Reg_def_t*)0x4002 3800)
#define RCC8  ((RCC_Reg_def_t*)0x4002 3800)
#define RCC9  ((RCC_Reg_def_t*)0x4002 3800)
#define RCC10 ((RCC_Reg_def_t*)0x4002 3800)
#define RCC11 ((RCC_Reg_def_t*)0x4002 3800)
#define RCC12 ((RCC_Reg_def_t*)0x4002 3800)
#define RCC13 ((RCC_Reg_def_t*)0x4002 3800)
#define RCC14 ((RCC_Reg_def_t*)0x4002 3800)
#define RCC15 ((RCC_Reg_def_t*)0x4002 3800)
#define RCC16 ((RCC_Reg_def_t*)0x4002 3800)
#define RCC17 ((RCC_Reg_def_t*)0x4002 3800)
#define RCC18 ((RCC_Reg_def_t*)0x4002 3800)
#define RCC19 ((RCC_Reg_def_t*)0x4002 3800)
#define RCC20 ((RCC_Reg_def_t*)0x4002 3800)
#define RCC21 ((RCC_Reg_def_t*)0x4002 3800)
#define RCC22 ((RCC_Reg_def_t*)0x4002 3800)
#define RCC23 ((RCC_Reg_def_t*)0x4002 3800)
#define RCC24 ((RCC_Reg_def_t*)0x4002 3800)
#define RCC25 ((RCC_Reg_def_t*)0x4002 3800)
*/


#define RCC ((RCC_Reg_def_t*)RCC_BASEADDR)

//GPIO clock enable
#define GPIOA_pclock_enable() (RCC->AHB1ENR |=(1<<0))
#define GPIOB_pclock_enable() (RCC->AHB1ENR |=(1<<1))
#define GPIOC_pclock_enable() (RCC->AHB1ENR |=(1<<2))
#define GPIOD_pclock_enable() (RCC->AHB1ENR |=(1<<3))
#define GPIOE_pclock_enable()   (RCC->AHB1ENR |=(1<<4))
#define GPIOF_pclock_enable() (RCC->AHB1ENR |=(1<<5))
#define GPIOG_pclock_enable() (RCC->AHB1ENR |=(1<<6))
#define GPIOH_pclock_enable() (RCC->AHB1ENR |=(1<<7))
#define GPIOI_pclock_enable() (RCC->AHB1ENR |=(1<<8))

//GPIO peripheral clock disable macros
#define GPIOA_pclock_disable()  do{ (RCC->AHB1RSTR |= (1 << 0)); (RCC->AHB1RSTR &= ~(1 << 0)); }while(0)
#define GPIOB_pclock_disable()  do{ (RCC->AHB1RSTR |= (1 << 0)); (RCC->AHB1RSTR &= ~(1 << 0)); }while(0)
#define GPIOC_pclock_disable()  do{ (RCC->AHB1RSTR |= (1 << 0)); (RCC->AHB1RSTR &= ~(1 << 0)); }while(0)
#define GPIOD_pclock_disable()  do{ (RCC->AHB1RSTR |= (1 << 0)); (RCC->AHB1RSTR &= ~(1 << 0)); }while(0)
#define GPIOE_pclock_disable()  do{ (RCC->AHB1RSTR |= (1 << 0)); (RCC->AHB1RSTR &= ~(1 << 0)); }while(0)
#define GPIOF_pclock_disable()  do{ (RCC->AHB1RSTR |= (1 << 0)); (RCC->AHB1RSTR &= ~(1 << 0)); }while(0)
#define GPIOG_pclock_disable()  do{ (RCC->AHB1RSTR |= (1 << 0)); (RCC->AHB1RSTR &= ~(1 << 0)); }while(0)
#define GPIOH_pclock_disable()  do{ (RCC->AHB1RSTR |= (1 << 0)); (RCC->AHB1RSTR &= ~(1 << 0)); }while(0)
#define GPIOI_pclock_disable()  do{ (RCC->AHB1RSTR |= (1 << 0)); (RCC->AHB1RSTR &= ~(1 << 0)); }while(0)
```

```
        //important macro definitions
        #define ENABLE    1
        #define DISABLE  0
        #define GPIO_Pin_Set  ENABLE
        #define GPIO_Pin_Reset DIABLE

        #include "STM32FXX_GPIO_DRIVER.h"

    #endif /* INC_STM32F4XX_H_ */
```

2.2 GPIO Driver File

```
/*
 * STM32Fxx_GPIO_DRIVER.h
 *
 *  Created on: Dec 28, 2020
 *      Author: 99003171
 */

#ifndef INC_STM32FXX_GPIO_DRIVER_H_
#define INC_STM32FXX_GPIO_DRIVER_H_

#include "STM32F4XX.h"

//GPIO Pin configuration

typedef struct
{
uint8_t GPIO_PinNumber;
uint8_t GPIO_PinMode;
uint8_t GPIO_PinSpeed;
uint8_t GPIO_PinPuPdControl;
uint8_t GPIO_PinOType;
uint8_t GPIO_PinAltFunMode;
}GPIO_Pin_Config_t;

// GPIO Handle Structure

typedef struct
{     // pin definitions
GPIO_Reg_def_t *pGPIOx; //this holds base address of GPIO port to which port belongs
GPIO_Pin_Config_t pin_config;//GPIO pin config setting
}GPIO_Handle_t;

// macros for pin numbers
#define GPIO_Pin_Number_0 0
#define GPIO_Pin_Number_1 1
#define GPIO_Pin_Number_2 2
#define GPIO_Pin_Number_3 3
#define GPIO_Pin_Number_4 4
#define GPIO_Pin_Number_5 5
#define GPIO_Pin_Number_6 6
```

```
#define GPIO_Pin_Number_7 7
#define GPIO_Pin_Number_8 8
#define GPIO_Pin_Number_9 9
#define GPIO_Pin_Number_10 10
#define GPIO_Pin_Number_11 11
#define GPIO_Pin_Number_12 12
#define GPIO_Pin_Number_13 13
#define GPIO_Pin_Number_14 14
#define GPIO_Pin_Number_15 15

// macros for pin modes
#define GPIO_PinMode_IN 0 // non interrupt modes
#define GPIO_PinMode_OUT 1
#define GPIO_PinMode_ALTFN  2
#define GPIO_PinMode_ANALOG 3
#define GPIO_PinMode_IT_FT  4 // falling edge triggered
#define GPIO_PinMode_IT_RT  5 // raising edge triggered
#define GPIO_PinMode_IT_RFT 6 // falling & raising edge triggered

// macros for pin speed
#define GPIO_Speed_LOW 0  //low speed
#define GPIO_Speed_MEDIUM 1 //Medium speed
#define GPIO_Speed_FAST 2 //High Speed
#define GPIO_Speed_HIGH 3 //Very High speed

// macros for pin PUPD control
#define GPIO_PinPuPdControl_PUPD 0 // no pull up pull down
#define GPIO_PinPuPdControl_PU 1 // pull up
#define GPIO_PinPuPdControl_PD 2 // pull down
#define GPIO_PinPuPdControl_Reserved 3 // reserved

// macros for pin OType
#define GPIO_PinOType_PP 0 //push pull
#define GPIO_PinOType_OD 1 //open drain

// GPIO driver API'S

//Peripheral clock setup
void GPIO_PeriClockControl(GPIO_Reg_def_t *pGPIOx, uint8_t EnorDi);

//Init and Deinit
void GPIO_Init(GPIO_Handle_t *pGPIOHandle);
void GPIO_DeInit(GPIO_Reg_def_t *pGPIOx);

//Data Read and Write
uint8_t GPIO_ReadFromInputPin(GPIO_Reg_def_t *pGPIOx, uint8_t PinNumber);
uint16_t GPIO_ReadFromInputPort(GPIO_Reg_def_t *pGPIOx);
void GPIO_WriteToOutputPin(GPIO_Reg_def_t *pGPIOx, uint8_t PinNumber, uint8_t Value);
void GPIO_WriteToOutputPort(GPIO_Reg_def_t *pGPIOx, uint16_t Value);
void GPIOToggleOutputPin(GPIO_Reg_def_t *pGPIOx, uint8_t PinNumber);


#endif /* INC_STM32FXX_GPIO_DRIVER_H_ */
```

2.3 Source File

```
/*
* STM32Fxx_GPIO_DRIVER.c
*
*  Created on: Dec 28, 2020
*      Author: 99003171
*/
#include "STM32FXX_GPIO_DRIVER.h"

// GPIO driver API'S

//Peripheral clock setup
void GPIO_PeriClockControl(GPIO_Reg_def_t *pGPIOx, uint8_t EnorDi)
{
		if( EnorDi == ENABLE )
					{
					if(pGPIOx == GPIOA)
					{
					GPIOA_pclock_enable();
					}
					else if(pGPIOx == GPIOB)
					{
					GPIOB_pclock_enable();
					}
					else if(pGPIOx == GPIOC)
					{
					GPIOC_pclock_enable();
					}
					else if(pGPIOx == GPIOD)
					{
					GPIOD_pclock_enable();
					}
					else if(pGPIOx == GPIOE)
					{
					GPIOE_pclock_enable();
					}
					else if(pGPIOx == GPIOF)
					{
					GPIOF_pclock_enable();
					}
					else if(pGPIOx == GPIOG)
					{
					GPIOG_pclock_enable();
					}
					else if(pGPIOx == GPIOH)
					{
					GPIOH_pclock_enable();
					}
					else if(pGPIOx == GPIOI)
					{
					GPIOI_pclock_enable();
```

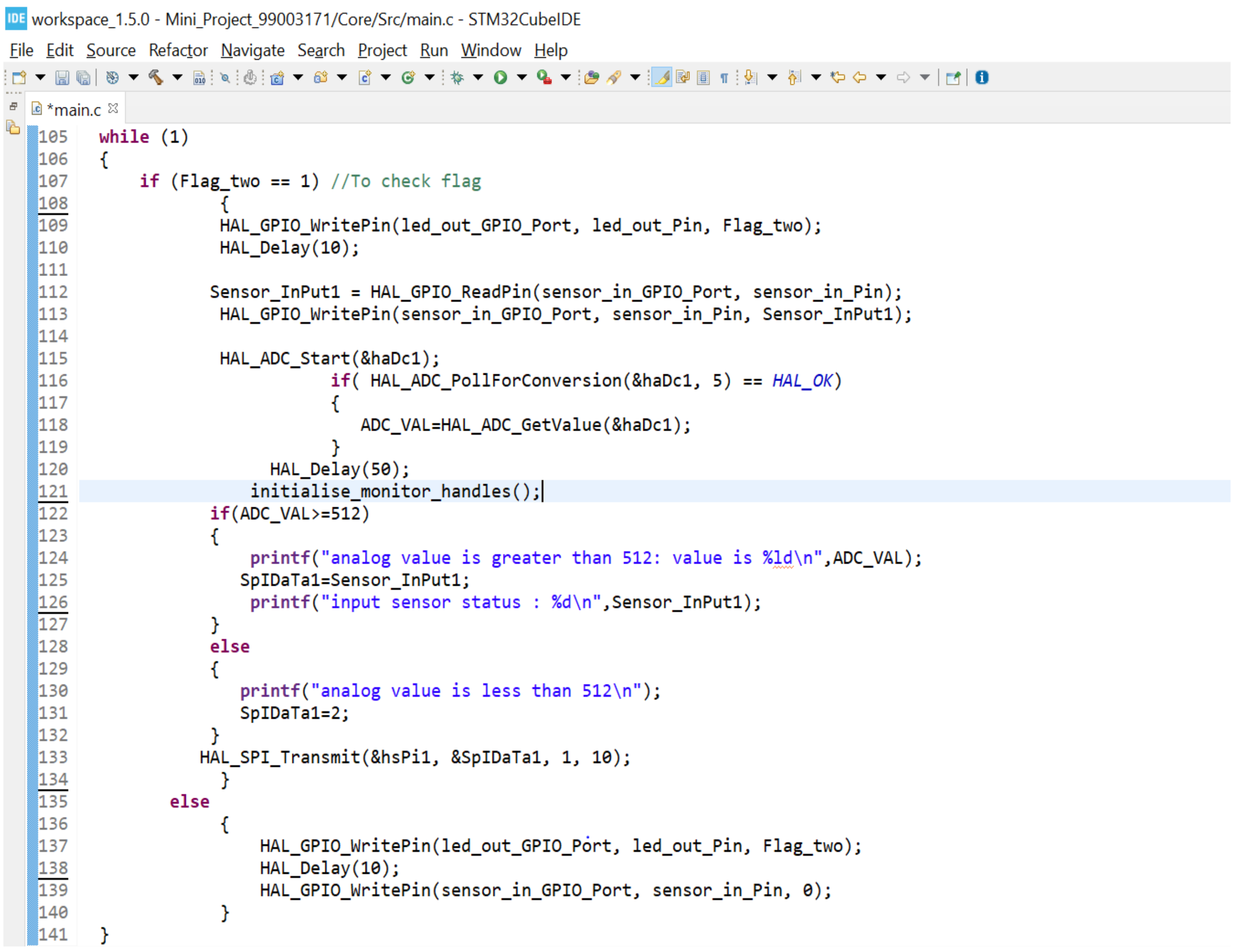
```
                              }

                              }
        else
                              {
                              if(pGPIOx == GPIOA)
                              {
                              GPIOA_pclock_disable();
                              }
                              else if(pGPIOx == GPIOB)
                              {
                              GPIOB_pclock_disable();
                              }
                              else if(pGPIOx == GPIOC)
                              {
                              GPIOC_pclock_disable();
                              }
                              else if(pGPIOx == GPIOD)
                              {
                              GPIOD_pclock_disable();
                              }
                              else if(pGPIOx == GPIOE)
                              {
                              GPIOE_pclock_disable();
                              }
                              else if(pGPIOx == GPIOF)
                              {
                              GPIOF_pclock_disable();
                              }
                              else if(pGPIOx == GPIOG)
                              {
                              GPIOG_pclock_disable();
                              }
                              else if(pGPIOx == GPIOH)
                              {
                              GPIOH_pclock_disable();
                              }
                              else if(pGPIOx == GPIOI)
                              {
                              GPIOI_pclock_disable();
                              }
                              }
}

//Initialization and Deinitialization
void GPIO_Init(GPIO_Handle_t *pGPIOHandle)
{
//1. configuring the mode
uint32_t temp=0;
if(pGPIOHandle->pin_config.GPIO_PinMode <= GPIO_PinMode_ANALOG )//non interrupt modes
{
temp= pGPIOHandle->pin_config.GPIO_PinMode<<(2*pGPIOHandle->pin_config.GPIO_PinNumber);
pGPIOHandle->pGPIOx->MODER |= temp;
```

```
}
//2. configuring the speed
uint32_t temp1=0;
temp1= pGPIOHandle->pin_config.GPIO_PinSpeed<<(2*pGPIOHandle->pin_config.GPIO_PinNumber);
pGPIOHandle->pGPIOx->OSPEEDR |= temp1;

//3. configuring the pu pd control
uint32_t temp2=0;
temp2= pGPIOHandle->pin_config.GPIO_PinPuPdControl<<(2*pGPIOHandle->pin_config.GPIO_PinNumber);
pGPIOHandle->pGPIOx->PUPDR |= temp2;

//4. configuring the output type
uint32_t temp3=0;
temp3= pGPIOHandle->pin_config.GPIO_PinOType<<(pGPIOHandle->pin_config.GPIO_PinNumber);
pGPIOHandle->pGPIOx->OTYPER |= temp3;

//Alternate function
if(pGPIOHandle->pin_config.GPIO_PinMode==GPIO_PinMode_ALTFN)
{
uint32_t temp4,temp5;
temp4=pGPIOHandle->pin_config.GPIO_PinNumber/8;
temp5=pGPIOHandle->pin_config.GPIO_PinNumber%8;
pGPIOHandle->pGPIOx->AFR[temp4] |= pGPIOHandle->pin_config.GPIO_PinAltFunMode<<(4*temp5);
}

}

void GPIO_DeInit(GPIO_Reg_def_t *pGPIOx)
{
            if(pGPIOx == GPIOA)
            {
            GPIOA_pclock_disable();
            }
            else if(pGPIOx == GPIOB)
            {
            GPIOB_pclock_disable();
            }
            else if(pGPIOx == GPIOC)
            {
            GPIOC_pclock_disable();
            }
            else if(pGPIOx == GPIOD)
            {
            GPIOD_pclock_disable();
            }
            else if(pGPIOx == GPIOE)
            {
            GPIOE_pclock_disable();
            }
            else if(pGPIOx == GPIOF)
            {
            GPIOF_pclock_disable();
            }
```

```
                    else if(pGPIOx == GPIOG)
                    {
                    GPIOG_pclock_disable();
                    }
                    else if(pGPIOx == GPIOH)
                    {
                    GPIOH_pclock_disable();
                    }
                    else if(pGPIOx == GPIOI)
                    {
                    GPIOI_pclock_disable();
                    }

}

//Data Read and Write
uint8_t GPIO_ReadFromInputPin(GPIO_Reg_def_t *pGPIOx, uint8_t PinNumber)
{
uint8_t value;
value=(uint8_t)((pGPIOx->IDR>>PinNumber)*(0x00000001));
return value;
}

uint16_t GPIO_ReadFromInputPort(GPIO_Reg_def_t *pGPIOx)
{
uint16_t value1;
value1=(uint16_t)(pGPIOx->IDR);
return value1;
}

void GPIO_WriteToOutputPin(GPIO_Reg_def_t *pGPIOx, uint8_t PinNumber, uint8_t Value)
{
if(Value==GPIO_Pin_Set)
{
pGPIOx->ODR |= (1<<PinNumber);
}
else
{
pGPIOx->ODR &= ~(1<<PinNumber);
}
}

void GPIO_WriteToOutputPort(GPIO_Reg_def_t *pGPIOx, uint16_t Value)
{
pGPIOx->ODR = Value;
}

void GPIOToggleOutputPin(GPIO_Reg_def_t *pGPIOx, uint8_t PinNumber)
{
pGPIOx->ODR = pGPIOx->ODR ^ (1<<PinNumber);
}
```

Activity 3

MiniProject


workspace_1.5.0 - Mini_Project_99003171/Core/Src/main.c - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help
*main.c

```
while (1)
{
    if (Flag_two == 1) //To check flag
            {
            HAL_GPIO_WritePin(led_out_GPIO_Port, led_out_Pin, Flag_two);
            HAL_Delay(10);

           Sensor_InPut1 = HAL_GPIO_ReadPin(sensor_in_GPIO_Port, sensor_in_Pin);
            HAL_GPIO_WritePin(sensor_in_GPIO_Port, sensor_in_Pin, Sensor_InPut1);

            HAL_ADC_Start(&haDc1);
                       if( HAL_ADC_PollForConversion(&haDc1, 5) == HAL_OK)
                       {
                          ADC_VAL=HAL_ADC_GetValue(&haDc1);
                       }
                 HAL_Delay(50);
               initialise_monitor_handles();
           if(ADC_VAL>=512)
           {
               printf("analog value is greater than 512: value is %ld\n",ADC_VAL);
              SpIDaTa1=Sensor_InPut1;
               printf("input sensor status : %d\n",Sensor_InPut1);
           }
           else
           {
              printf("analog value is less than 512\n");
              SpIDaTa1=2;
           }
          HAL_SPI_Transmit(&hsPi1, &SpIDaTa1, 1, 10);
            }
        else
            {
                HAL_GPIO_WritePin(led_out_GPIO_Port, led_out_Pin, Flag_two);
                HAL_Delay(10);
                HAL_GPIO_WritePin(sensor_in_GPIO_Port, sensor_in_Pin, 0);
            }
}
```

Github Link

Code files

https://github.com/99003171/Emb-C-code-files/tree/main/embedded

Miniproject

https://github.com/99003171/Embedded-miniproject

References

[1] http://web.cs.iastate.edu/~smkautz/cs227s13/labs/lab6/page04.html
[2]https://youtu.be/2Hm8eEHsgls
[3] https://youtu.be/Bsq6P1B8JqI
[4]https://www.youtube.com/watch?v=5aafG5mjZ_Y&list=PLERTijJOmYrDiiWd10iRHY0VRHdJwUH4g&index=5