Learning Report _ Embedded C

L&T Technology Services

Document History

Ver. Rel. No.	Release Date	Prepared. By	Reviewed By	Approved By	Remarks/Revision Details
	08/03/2021	Akshansh Mishra (Ps No- 99003753)			

Contents

ACTIVITY 1 – COMPILATION APPROACH..... 4
1.1- MAKE FILE 4
1.2- STARTUP CODE..... 5
1.3- LINKER FILE..... 8
1.4- DEBUGGING TECHNIQUES..... 9

ACTIVITY 2 – IMPLEMENTATION OF PROTOCOLS USING STM IDE 10
2.1 GPIO..... 10
2.2 EXTI..... 12
2.3 ADC..... 14
2.4 SPI..... 16
2.5 UART..... 19

Activity 1 – COMPILATION APROACH

This is the complete compilation process of the sample program for ARM Cortex Mx processor based boards. Following are the compilation stages of a C program:

1. Preprocessor stage
2. Compilation stage
3. Assembly stage
4. Linking stage

1.1- MAKE FILE

Below is the make file for the sample program:

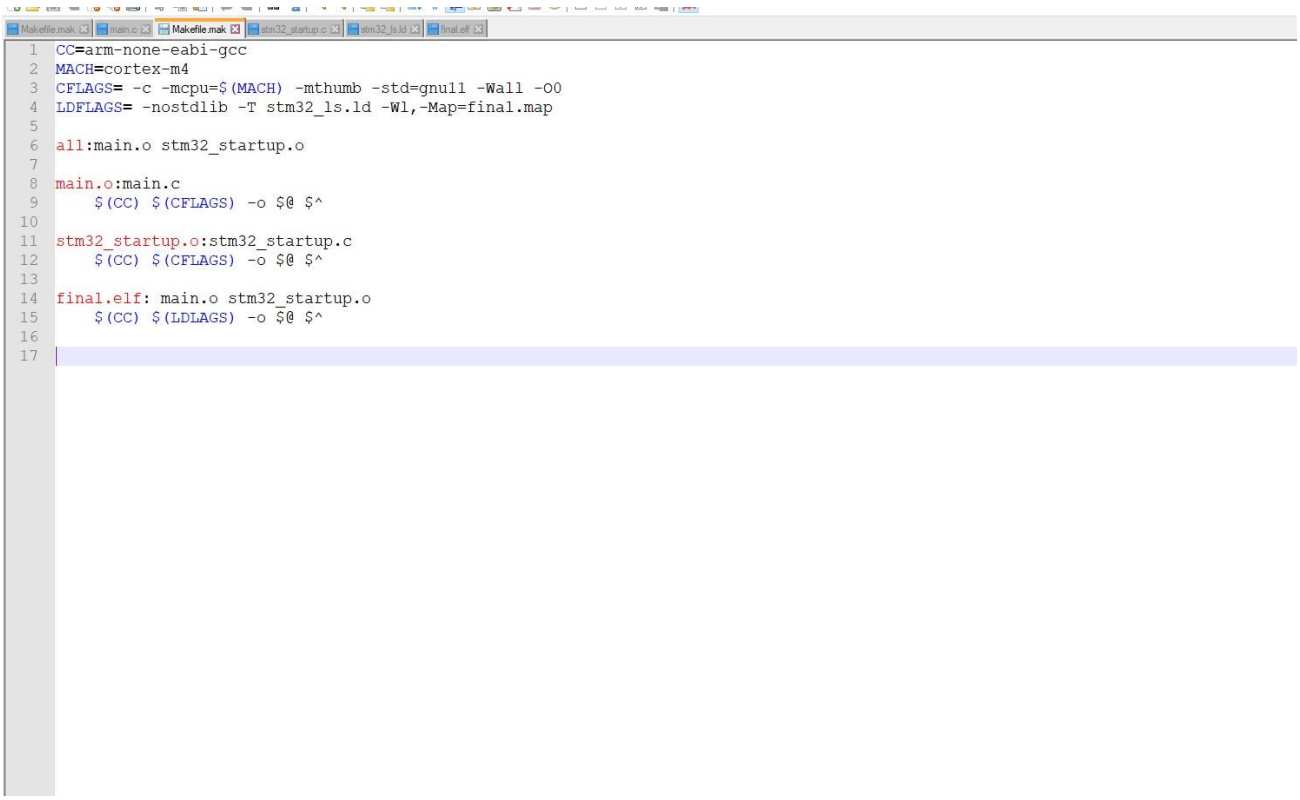


Fig 1.1.1 make file

The command to run this make file in the command prompt is:

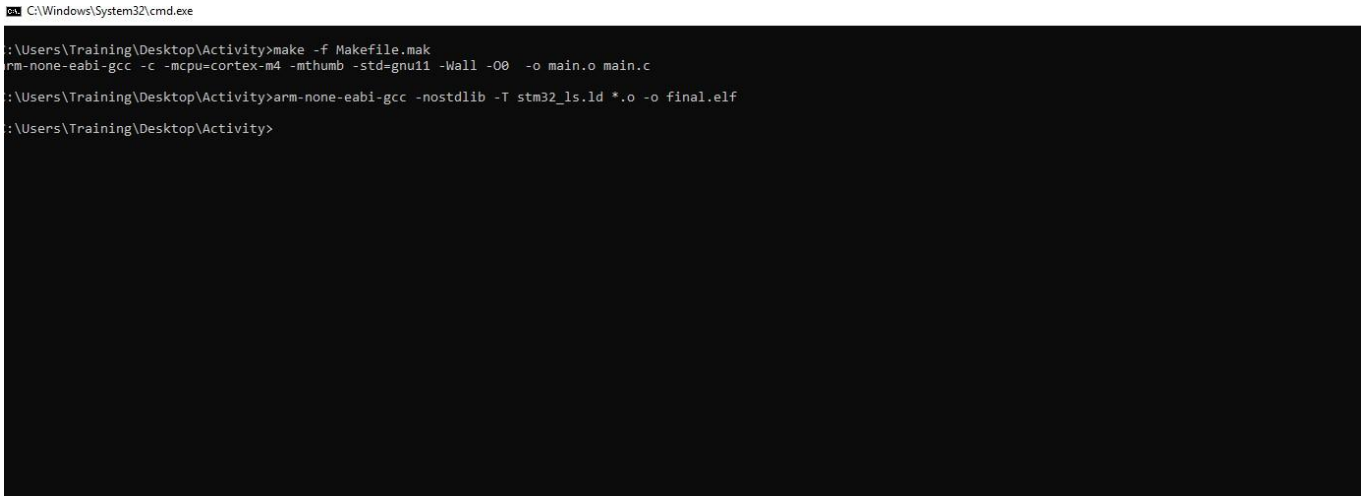


Fig 1.1.2 Make command

- **-mcpu=cortex-m4** is used to select our cortex-m4 processor which is used
- **-mthumb** is used to generate the code that executes in ARM state
- **main.o** is the target file
- **main.c** is the dependency

1.2- STARTUP CODE

- The startup file is responsible for setting up the right environments to run the code in main.c file.
- Some part of the startup code is target (processor) dependent.
- Role of startup file:
  1. Create a MCU specific vector table for microcontroller.
  2. To write a startup code which initializes .data and .bss section in SRAM.
  3. Call main()

```c
#include<stdint.h>

#define SRAM_START  0x20000000U
#define SRAM_SIZE   (128U * 1024U) //128KB
#define SRAM_END    ((SRAM_START) + (SRAM_SIZE))

#define STACK_START   SRAM_END

extern uint32_t _etext;
extern uint32_t _sdata;
extern uint32_t _edata;
extern uint32_t _la_data;

extern uint32_t _sbss;
extern uint32_t _ebss;

//prototype of main

int main(void);

void __libc_init_array(void);


/* function prototypes of STM32F407x system exception and IRQ handlers */

void Reset_Handler(void);

void NMI_Handler                    (void) __attribute__ ((weak, alias("Default_Handler")));
void HardFault_Handler              (void) __attribute__ ((weak, alias("Default_Handler")));
void MemManage_Handler              (void) __attribute__ ((weak, alias("Default_Handler")));
void BusFault_Handler               (void) __attribute__ ((weak, alias("Default_Handler")));
void UsageFault_Handler             (void) __attribute__ ((weak, alias("Default_Handler")));
void SVC_Handler                    (void) __attribute__ ((weak, alias("Default_Handler")));
void DebugMon_Handler               (void) __attribute__ ((weak, alias("Default_Handler")));
void PendSV_Handler                 (void) __attribute__ ((weak, alias("Default_Handler")));
void SysTick_Handler                (void) __attribute__ ((weak, alias("Default_Handler")));
void WWDG_IRQHandler                (void) __attribute__ ((weak, alias("Default_Handler")));
void PVD_IRQHandler                 (void) __attribute__ ((weak, alias("Default_Handler")));
void TAMP_STAMP_IRQHandler          (void) __attribute__ ((weak, alias("Default_Handler")));
void RTC_WKUP_IRQHandler            (void) __attribute__ ((weak, alias("Default_Handler")));
void RCC_IRQHandler                 (void) __attribute__ ((weak, alias("Default_Handler")));
void EXTI0_IRQHandler               (void) __attribute__ ((weak, alias("Default_Handler")));
void EXTI1_IRQHandler               (void) __attribute__ ((weak, alias("Default_Handler")));
void EXTI2_IRQHandler               (void) __attribute__ ((weak, alias("Default_Handler")));
void EXTI3_IRQHandler               (void) __attribute__ ((weak, alias("Default_Handler")));
void EXTI4_IRQHandler               (void) __attribute__ ((weak, alias("Default_Handler")));
void DMA1_Stream0_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
void DMA1_Stream1_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
void DMA1_Stream2_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
void DMA1_Stream3_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
void DMA1_Stream4_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
void DMA1_Stream5_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
void DMA1_Stream6_IRQHandler        (void) __attribute__ ((weak, alias("Default_Handler")));
void ADC_IRQHandler                 (void) __attribute__ ((weak, alias("Default_Handler")));
void CAN1_TX_IRQHandler             (void) __attribute__ ((weak, alias("Default_Handler")));
void CAN1_RX0_IRQHandler            (void) __attribute__ ((weak, alias("Default_Handler")));
void CAN1_RX1_IRQHandler            (void) __attribute__ ((weak, alias("Default_Handler")));
```

```c
    (uint32_t)DMA2_Stream3_IRQHandler,
    (uint32_t)DMA2_Stream4_IRQHandler,
    (uint32_t)ETH_IRQHandler,
    (uint32_t)ETH_WKUP_IRQHandler,
    (uint32_t)CAN2_TX_IRQHandler,
    (uint32_t)CAN2_RX0_IRQHandler,
    (uint32_t)CAN2_RX1_IRQHandler,
    (uint32_t)CAN2_SCE_IRQHandler,
    (uint32_t)OTG_FS_IRQHandler,
    (uint32_t)DMA2_Stream5_IRQHandler,
    (uint32_t)DMA2_Stream6_IRQHandler,
    (uint32_t)DMA2_Stream7_IRQHandler,
    (uint32_t)USART6_IRQHandler,
    (uint32_t)I2C3_EV_IRQHandler,
    (uint32_t)I2C3_ER_IRQHandler,
    (uint32_t)OTG_HS_EP1_OUT_IRQHandler,
    (uint32_t)OTG_HS_EP1_IN_IRQHandler,
    (uint32_t)OTG_HS_WKUP_IRQHandler,
    (uint32_t)OTG_HS_IRQHandler,
    (uint32_t)DCMI_IRQHandler,
    (uint32_t)CRYP_IRQHandler,
    (uint32_t)HASH_RNG_IRQHandler,
    (uint32_t)FPU_IRQHandler,
};


void Default_Handler(void)
{
    while(1);
}

void Reset_Handler(void)
{
    //copy .data section to SRAM
    uint32_t size = (uint32_t)&_edata - (uint32_t)&_sdata;

    uint8_t *pDst = (uint8_t*)&_sdata; //sram
    uint8_t *pSrc = (uint8_t*)&_la_data; //flash

    for(uint32_t i =0 ; i < size ; i++)
    {
        *pDst++ = *pSrc++;
    }

    //Init. the .bss section to zero in SRAM
    size = (uint32_t)&_ebss - (uint32_t)&_sbss;
    pDst = (uint8_t*)&_sbss;
    for(uint32_t i =0 ; i < size ; i++)
    {
        *pDst++ = 0;
    }

    //__libc_init_array();

    main();
}
```

Fig 1.2.1 Startup code

In startup code we use variable attributes to store some variables in the user defined function.
Function attributes:

- Weak: Lets programmer override already defined weak function (dummy function) with the same function name.
- Alias: Lets programmer give any alias name for same function.

The startup.o file generated is of elf executable format, various sections of which are shown below:

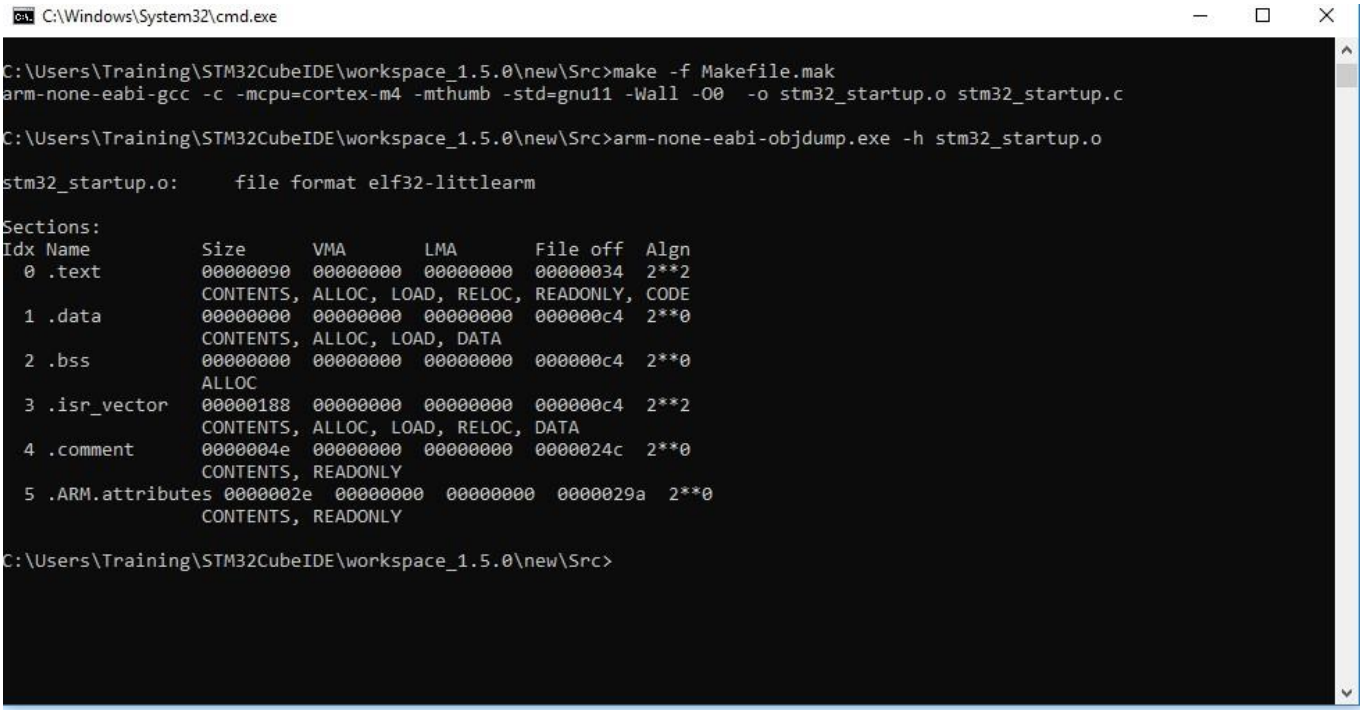


Fig 1.2.2: Startup command

1.3- LINKER SCRIPT

- Linkers take one or more object files or libraries as input and combines them to create a single executable file as output.
- Linker scripts decide how different sections of object file should be merged to create an output file.
- Reset handler is the entry point to the application
- Entry command is used to set the “Entry point address” information in the header of final elf file generated.
  Syntax: Entry(symbol_name)
  Entry(Reset_Handler)

```
{
  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


```

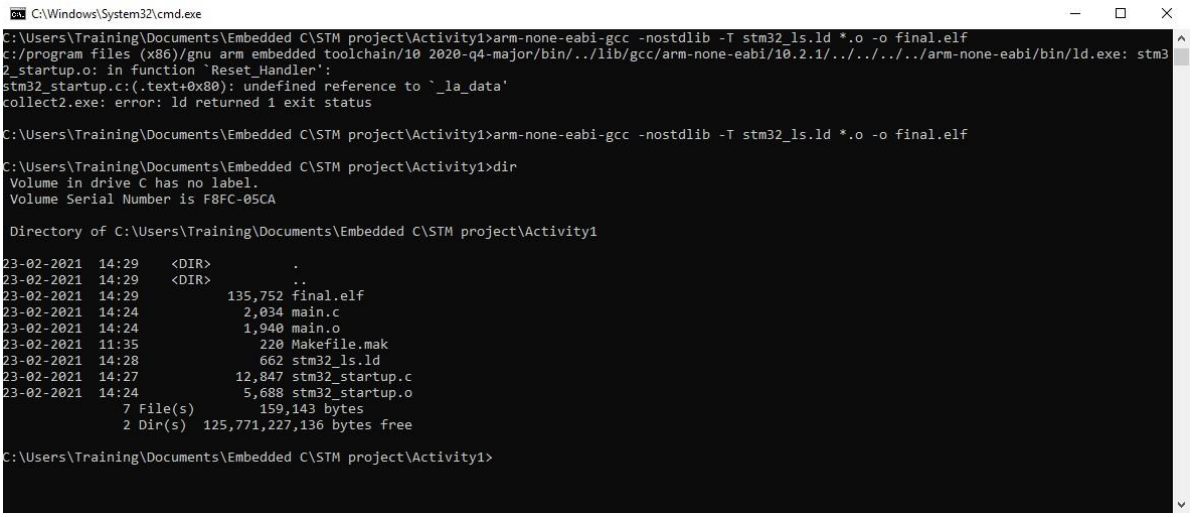


Fig 1.3.1 command to generate final.elf file

1.4- DEBUGGING TECHNIQUES

- The STM32F407VG is embedded with on chip debugger for debugging the code.
- The OCD ON-Chip Debugger aims to provide debugging, in system programming and boundary scan testing for embedded target devices.
- OCD is a free and opensource host application allows you to program, debug, and analyze your applications using GDB.
- It supports various target boards based on different processor architecture.

Activity 2 – IMPLEMENTATION OF PROTOCOLS USING STM IDE

Implementation of protocols for STM32F407VG microcontroller featuring ARM32 bit ARM-cortex M4 with FPU core using HAL library.

2.1 GPIO:

Toggling LED at pin PD12 at GREEN_LED_GPIO_PORT. Serial wire is enabled at pin PA13.

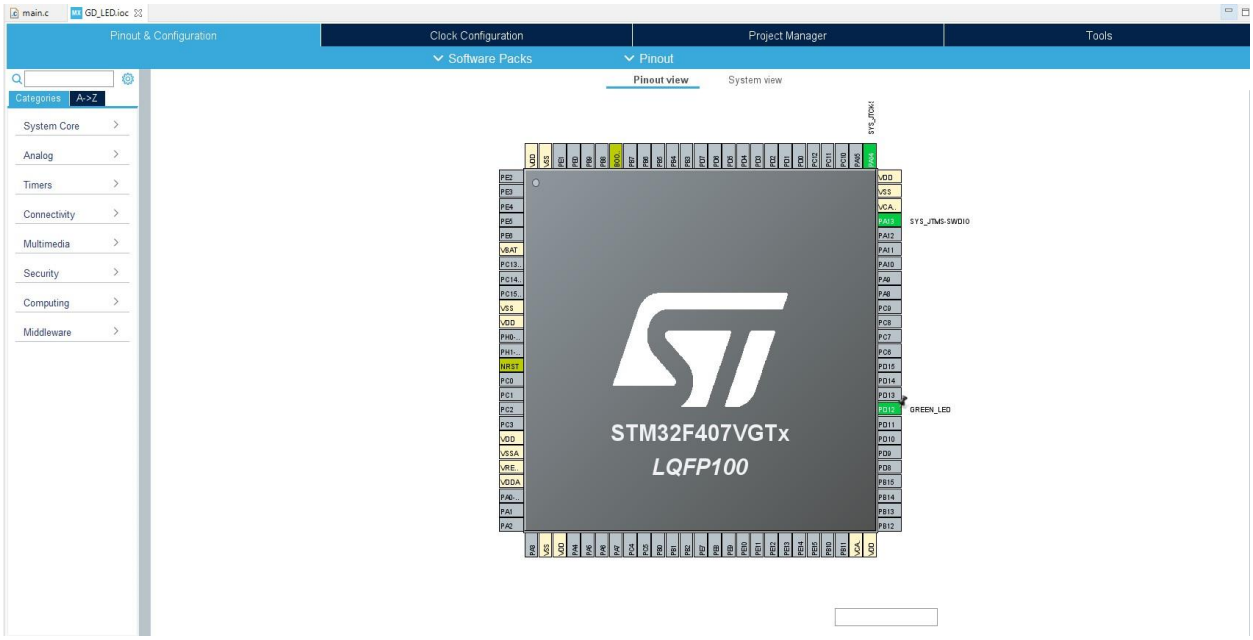


Fig: 2.1.1 GPIO pin configuration

```
/* Initialize all configured peripherals */
MX_GPIO_Init();
/* USER CODE BEGIN 2 */

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    HAL_GPIO_TogglePin(GREEN_LED_GPIO_Port, GREEN_LED_Pin);
    HAL_Delay(500);
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
}
  /* USER CODE END 3 */
}
```

Fig: 2.1.2 GPIO configuration code

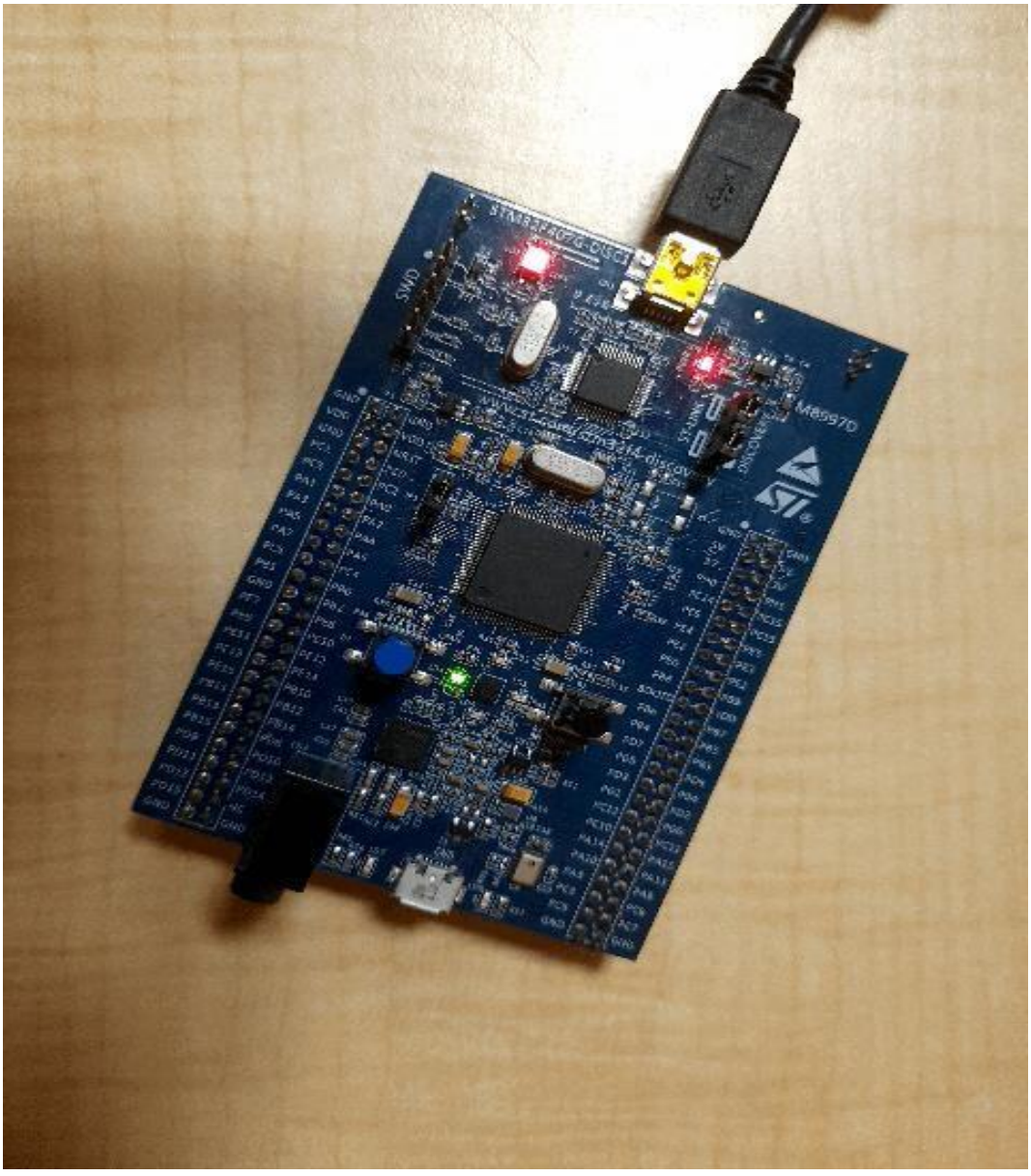


Fig: 2.1.2 LED toggling

2.2 EXTI :-

Blue button at PA0 works as an external interrupt.
When the blue button is pressed the Green LED at pin PD12 toggles.

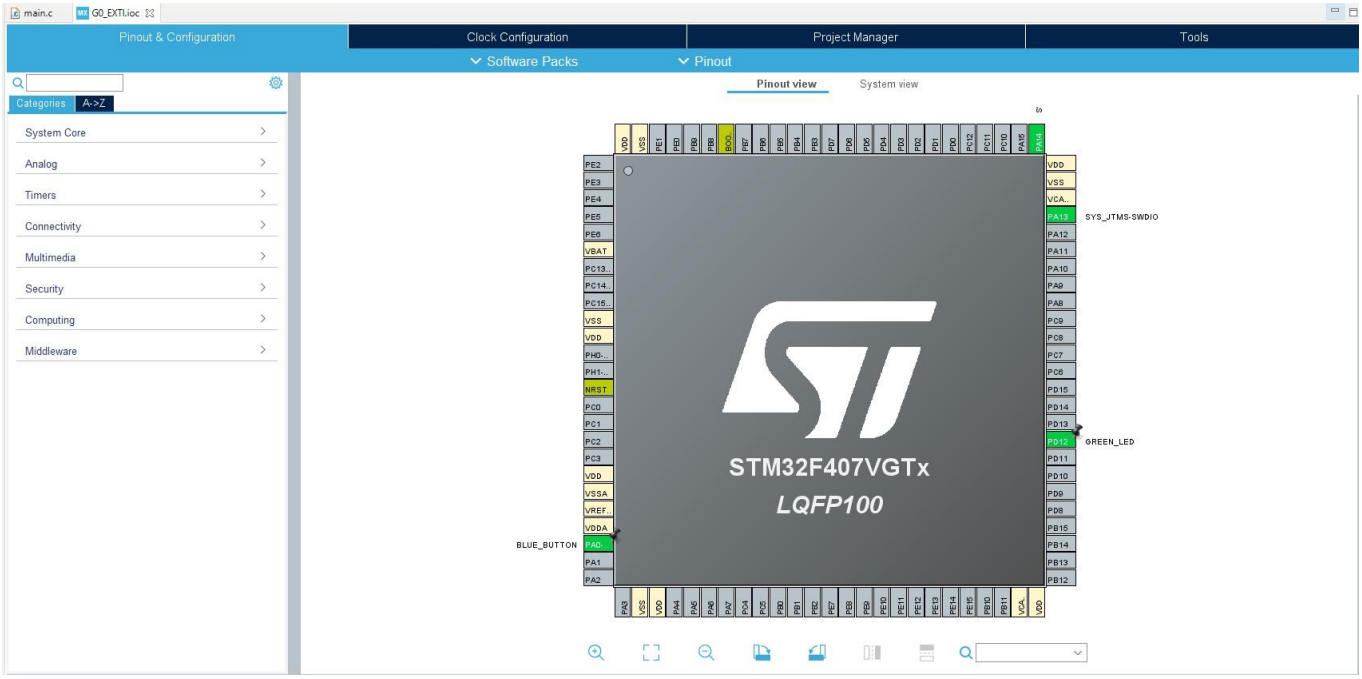


Fig: 2.2.1 EXTI pin configuration

In the main.c file a flag is initialized and if the flag == 1, the condition under the if loop executed to toggle the LED at PD12.

main.c G0_EXTI.ioc

```
43
44  /* USER CODE BEGIN PV */
45  uint8_t flag = 0;
46  /* USER CODE END PV */
47
48  /* Private function prototypes -----------------------------------------------*/
49  void SystemClock_Config(void);
50  static void MX_GPIO_Init(void);
51  /* USER CODE BEGIN PFP */
52
53  /* USER CODE END PFP */
54
55  /* Private user code ---------------------------------------------------------*/
56  /* USER CODE BEGIN 0 */
57
58  /* USER CODE END 0 */
59
60  /**
61    * @brief  The application entry point.
62    * @retval int
63    */
64  int main(void)
65  {
66    /* USER CODE BEGIN 1 */
67
68    /* USER CODE END 1 */
69
70    /* MCU Configuration--------------------------------------------------------*/
71
72    /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
73    HAL_Init();
74
75    /* USER CODE BEGIN Init */
76
77    /* USER CODE END Init */
78
79    /* Configure the system clock */
80    SystemClock_Config();
81
82    /* USER CODE BEGIN SysInit */
83
84    /* USER CODE END SysInit */
85
86    /* Initialize all configured peripherals */
87    MX_GPIO_Init();
88    /* USER CODE BEGIN 2 */
89
90    /* USER CODE END 2 */
91
92    /* Infinite loop */
93    /* USER CODE BEGIN WHILE */
94    while (1)
95    {
96      /* USER CODE END WHILE */
97       if(1==flag){
98              HAL_GPIO_TogglePin(GREEN_LED_GPIO_Port, GREEN_LED_Pin);
99              HAL_Delay(500);
100             flag = 0;
101           }
102     /* USER CODE BEGIN 3 */
103   }
```

Fig: 2.2.1 EXTI configuration code

2.3 ADC

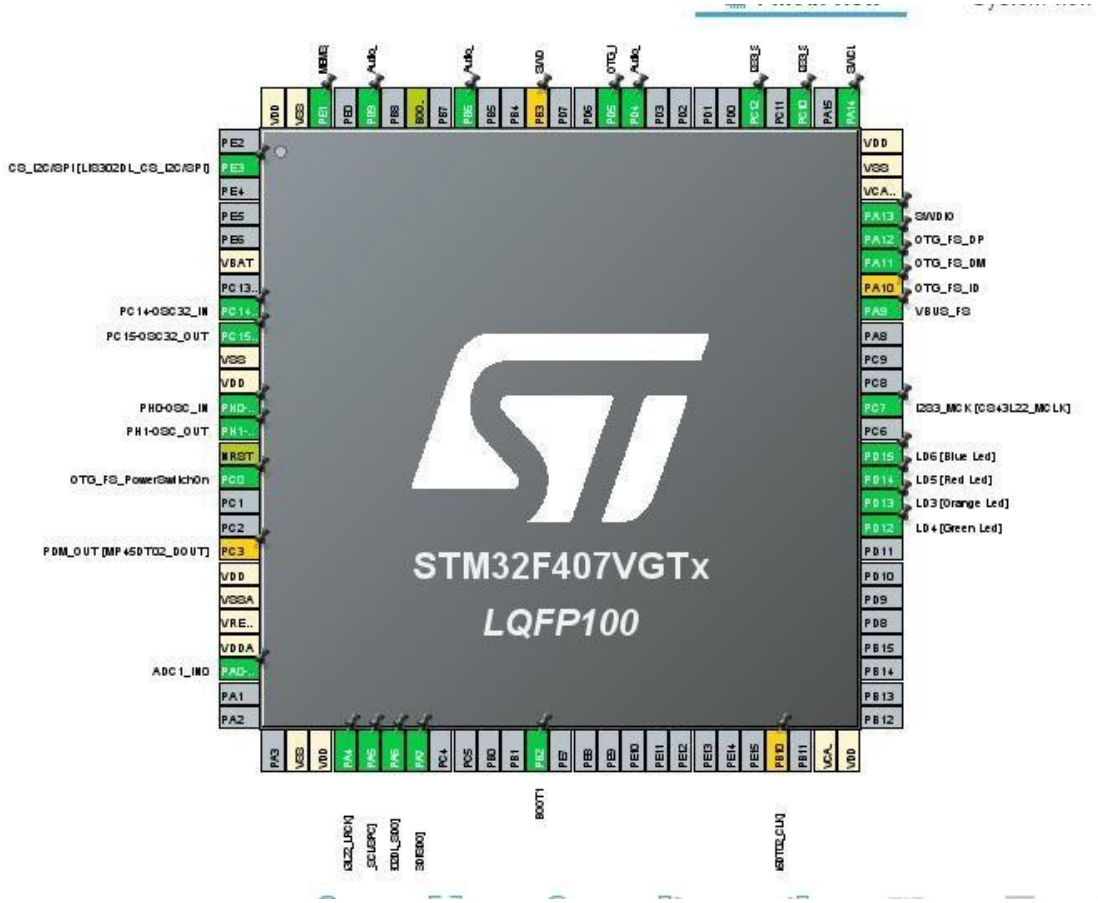


Fig: 2.3.1 ADC pin configuration

2.4 SPI

*main.c

```
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
#include "usb_host.h"


ADC_HandleTypeDef hadc1;

I2C_HandleTypeDef hi2c1;

I2S_HandleTypeDef hi2s3;

SPI_HandleTypeDef hspi1;


/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);
static void MX_I2S3_Init(void);
static void MX_SPI1_Init(void);
static void MX_ADC1_Init(void);
void MX_USB_HOST_Process(void);


uint32_t adc_value;
int main(void)
{

  HAL_Init();

  SystemClock_Config();

  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_I2S3_Init();
```

*main.c

```
  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_I2S3_Init();
  MX_SPI1_Init();
  MX_USB_HOST_Init();
  MX_ADC1_Init();

  while (1)
  {

    MX_USB_HOST_Process();


    HAL_ADC_Start(&hadc1);


    if(HAL_ADC_PollForConversion(&hadc1, 5)== HAL_OK){

        adc_value= HAL_ADC_GetValue(&hadc1);

    }


    HAL_ADC_Stop(&hadc1);
    HAL_Delay(100);

  }

}

void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};
  RCC_PeriphCLKInitTypeDef PeriphClkInitStruct = {0};
```

Fig: Configuration code

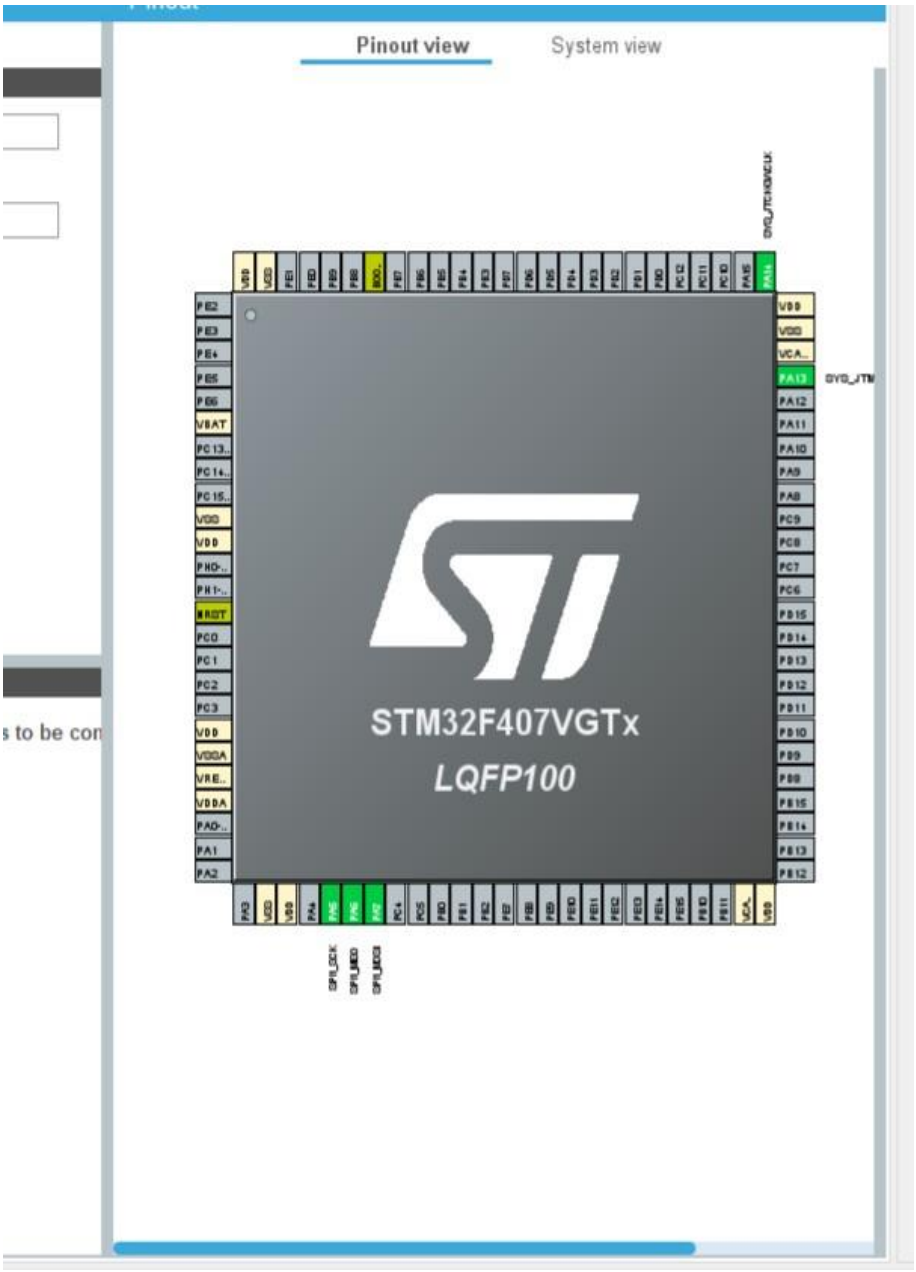
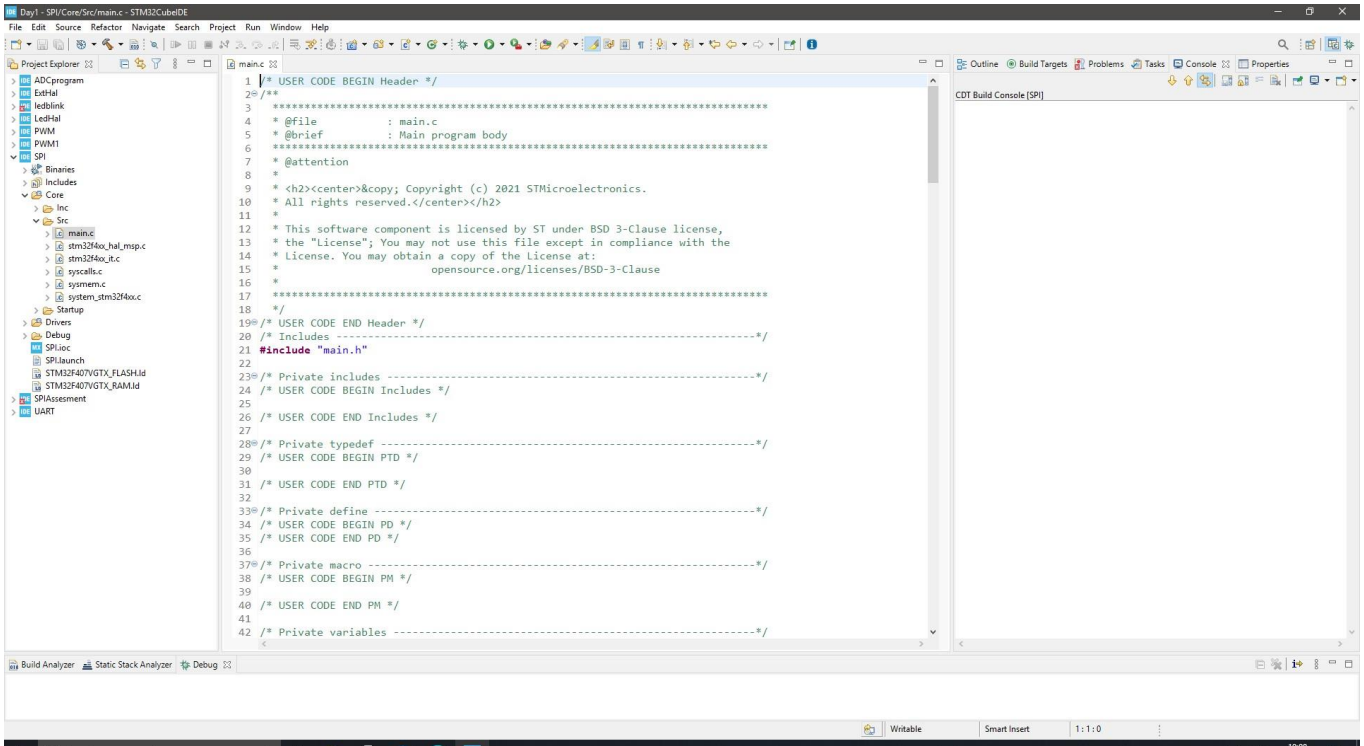
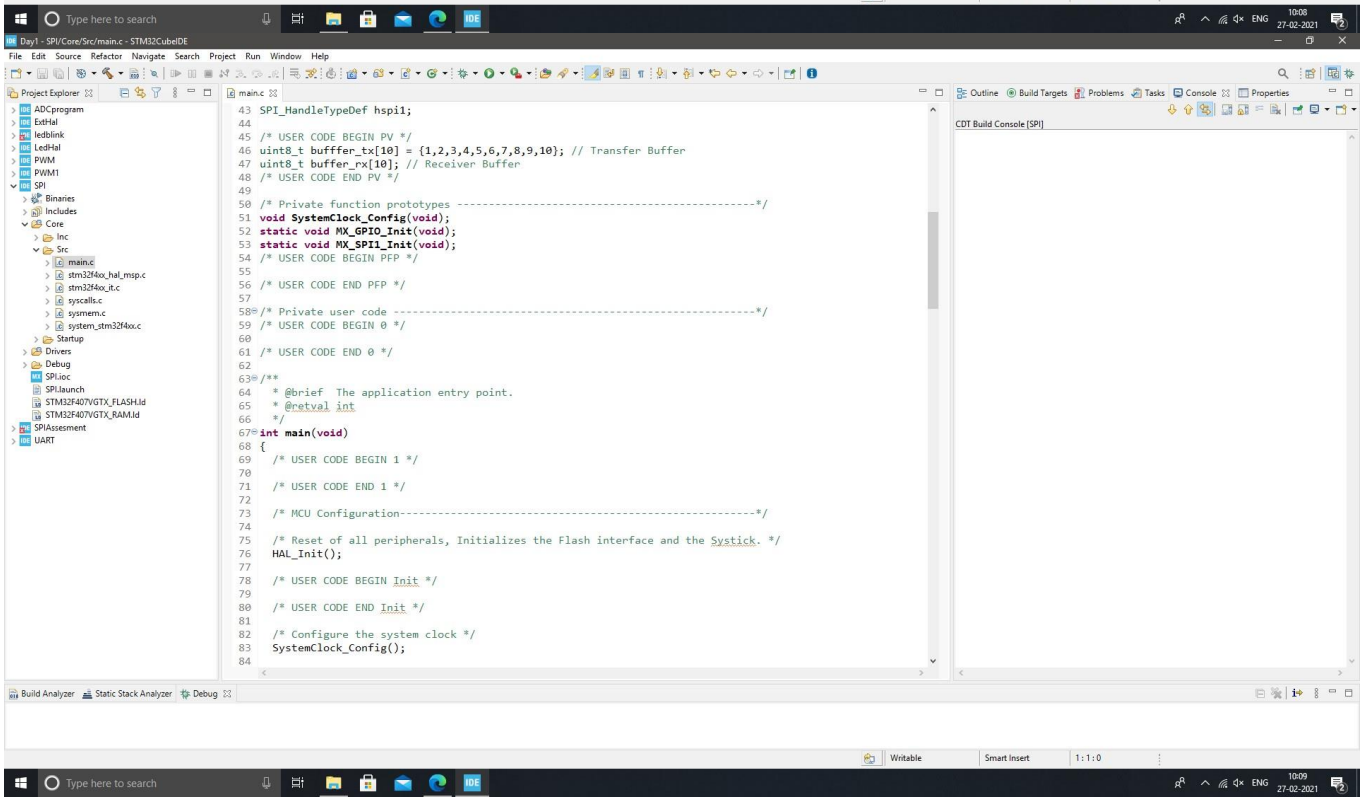


Figure: 2.4.1 SPI Pin configuration


Day1 - SPI/Core/Src/main.c - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
main.c
Outline Build Targets Problems Tasks Console Properties
CDT Build Console [SPI]
ADCprogram
ExtHal
ledblink
LedHal
PWM
PWM1
SPI
Binaries
Includes
Core
Inc
Src
main.c
stm32f4xx_hal_msp.c
stm32f4xx_it.c
syscalls.c
sysmem.c
system_stm32f4xx.c
Startup
Drivers
Debug
SPI.ioc
SPI.launch
STM32F407VGTX_FLASH.ld
STM32F407VGTX_RAM.ld
SPIAssesment
UART
1 /* USER CODE BEGIN Header */
2 /**
3 **
4 * @file : main.c
5 * @brief : Main program body
6 **
7 * @attention
8 *
9 * <h2> © Copyright (c) 2021 STMicroelectronics.
10 * All rights reserved. </h2>
11 *
12 * This software component is licensed by ST under BSD 3-Clause license,
13 * the "License"; You may not use this file except in compliance with the
14 * License. You may obtain a copy of the License at:
15 * opensource.org/licenses/BSD-3-Clause
16 *
17 **
18 */
19 /* USER CODE END Header */
20 /* Includes --*/
21 #include "main.h"
22
23 /* Private includes --*/
24 /* USER CODE BEGIN Includes */
25
26 /* USER CODE END Includes */
27
28 /* Private typedef ---*/
29 /* USER CODE BEGIN PTD */
30
31 /* USER CODE END PTD */
32
33 /* Private define --*/
34 /* USER CODE BEGIN PD */
35 /* USER CODE END PD */
36
37 /* Private macro ---*/
38 /* USER CODE BEGIN PM */
39
40 /* USER CODE END PM */
41
42 /* Private variables ---*/
Build Analyzer Static Stack Analyzer Debug
Writable Smart Insert 1:1:0
Type here to search
ENG 10:08 27-02-2021


Day1 - SPI/Core/Src/main.c - STM32CubeIDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
main.c
Outline Build Targets Problems Tasks Console Properties
CDT Build Console [SPI]
ADCprogram
ExtHal
ledblink
LedHal
PWM
PWM1
SPI
Binaries
Includes
Core
Inc
Src
main.c
stm32f4xx_hal_msp.c
stm32f4xx_it.c
syscalls.c
sysmem.c
system_stm32f4xx.c
Startup
Drivers
Debug
SPI.ioc
SPI.launch
STM32F407VGTX_FLASH.ld
STM32F407VGTX_RAM.ld
SPIAssesment
UART
43 SPI_HandleTypeDef hspi1;
44
45 /* USER CODE BEGIN PV */
46 uint8_t bufffer_tx[10] = {1,2,3,4,5,6,7,8,9,10}; // Transfer Buffer
47 uint8_t buffer_rx[10]; // Receiver Buffer
48 /* USER CODE END PV */
49
50 /* Private function prototypes ---*/
51 void SystemClock_Config(void);
52 static void MX_GPIO_Init(void);
53 static void MX_SPI1_Init(void);
54 /* USER CODE BEGIN PFP */
55
56 /* USER CODE END PFP */
57
58 /* Private user code ---*/
59 /* USER CODE BEGIN 0 */
60
61 /* USER CODE END 0 */
62
63 /**
64 * @brief The application entry point.
65 * @retval int
66 */
67 int main(void)
68 {
69 /* USER CODE BEGIN 1 */
70
71 /* USER CODE END 1 */
72
73 /* MCU Configuration--*/
74
75 /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
76 HAL_Init();
77
78 /* USER CODE BEGIN Init */
79
80 /* USER CODE END Init */
81
82 /* Configure the system clock */
83 SystemClock_Config();
84
Build Analyzer Static Stack Analyzer Debug
Writable Smart Insert 1:1:0
Type here to search
ENG 10:09 27-02-2021

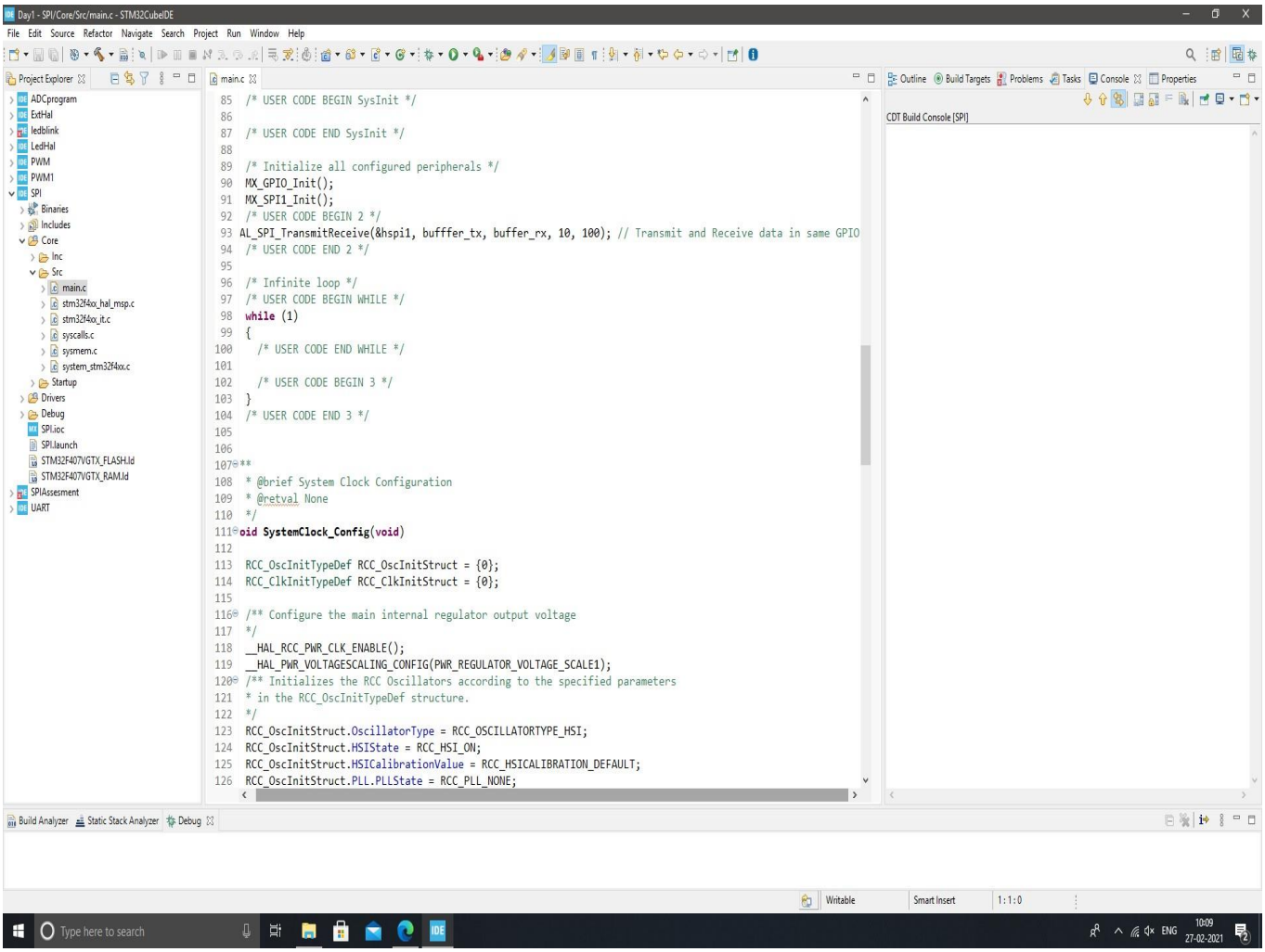


Fig: 2.4.2 SPI configuration code

2.5 UART

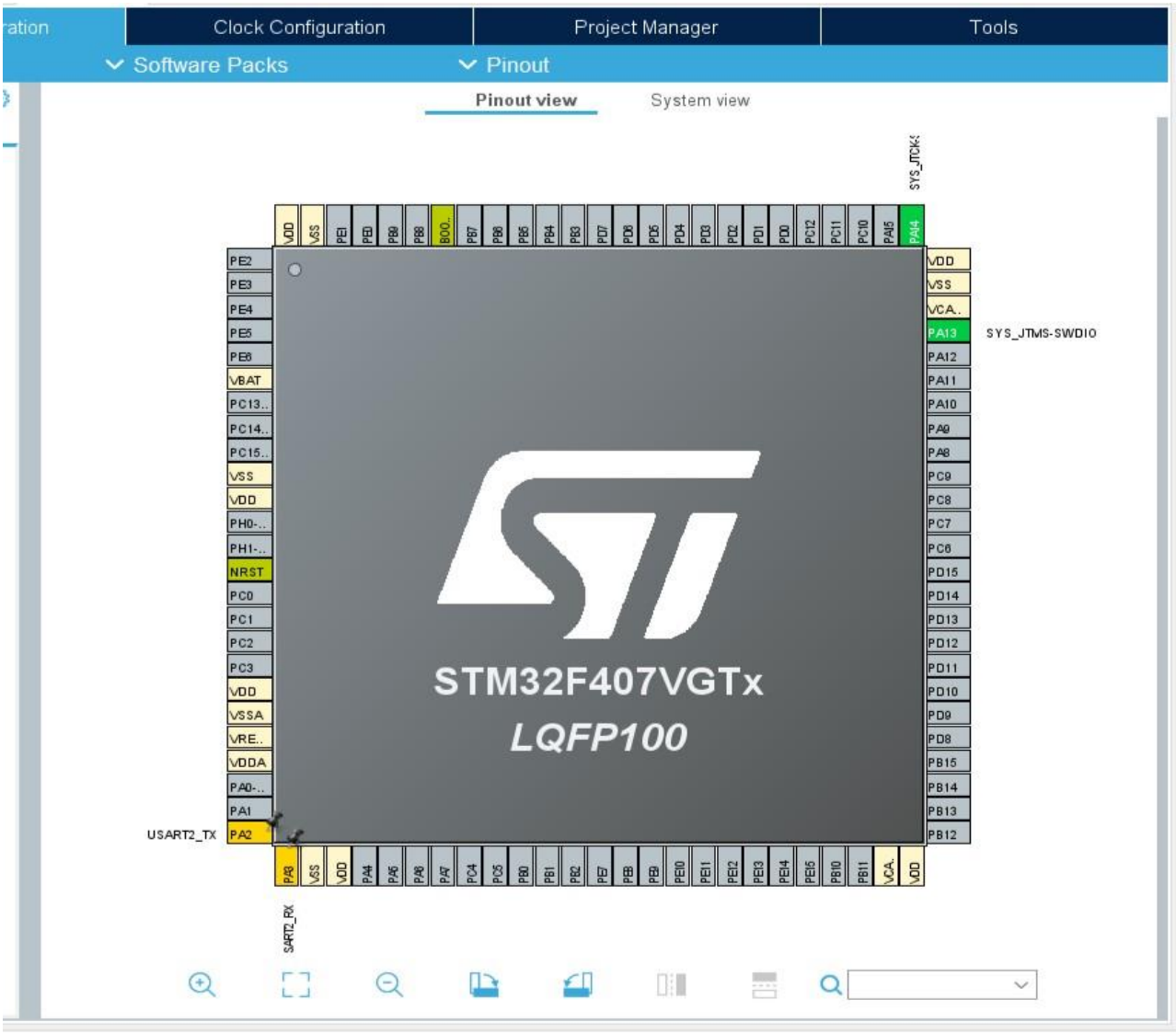
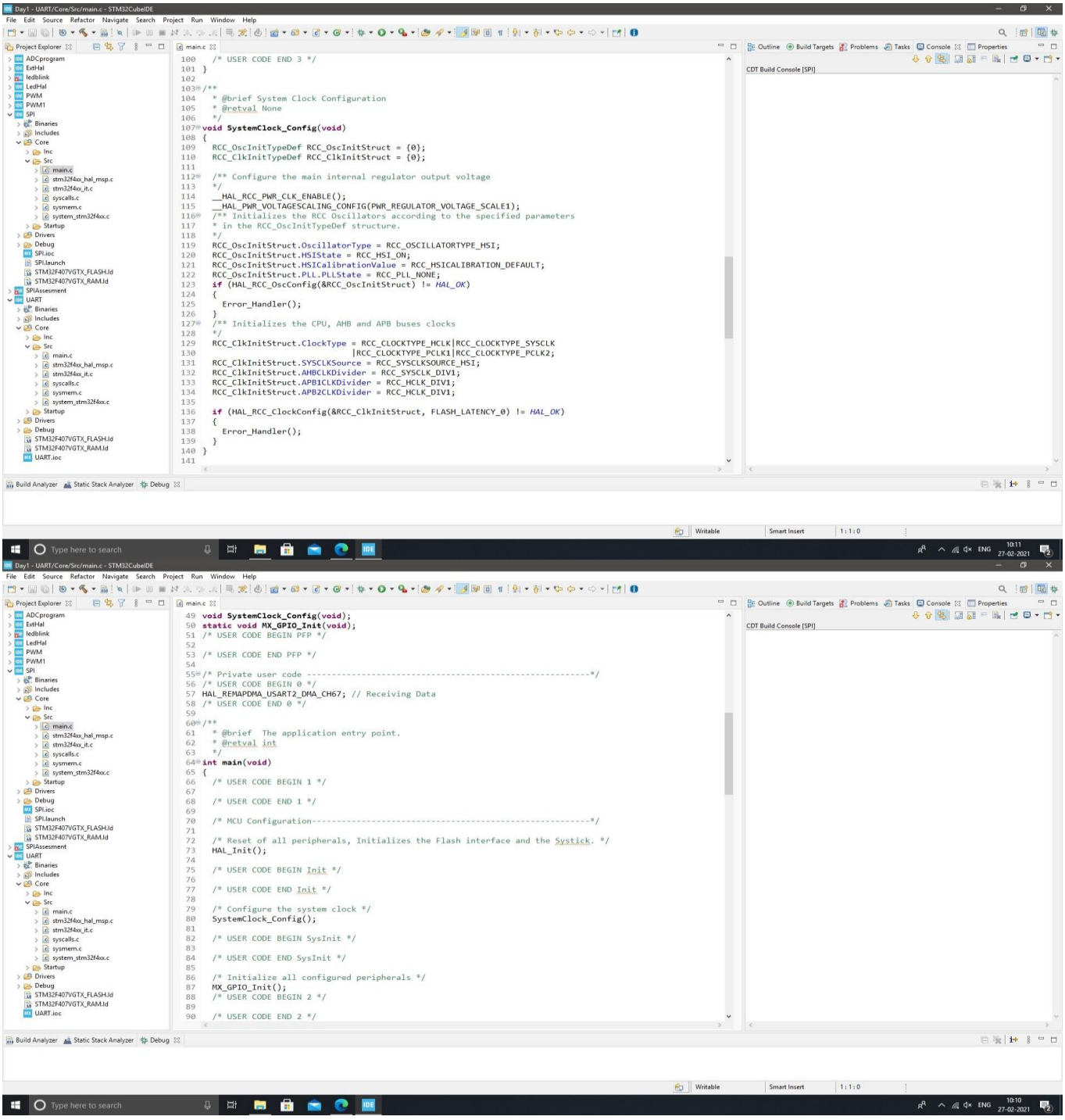


Fig: 2.5.1 UART Pin configuration




```c
  /* USER CODE END 3 */
}

/**
  * @brief System Clock Configuration
  * @retval None
  */
void SystemClock_Config(void)
{
  RCC_OscInitTypeDef RCC_OscInitStruct = {0};
  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};

  /** Configure the main internal regulator output voltage
  */
  __HAL_RCC_PWR_CLK_ENABLE();
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);
  /** Initializes the RCC Oscillators according to the specified parameters
  * in the RCC_OscInitTypeDef structure.
  */
  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }
  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_0) != HAL_OK)
  {
    Error_Handler();
  }
}
```

```c
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
HAL_REMAPDMA_USART2_DMA_CH67; // Receiving Data
/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
  */
int main(void)
{
  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

  /* MCU Configuration--------------------------------------------------------*/

  /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
  HAL_Init();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */
```

Fig: 2.5.2 UART configuration code