

Project 1 Module 4 Overview :

In this specific module, our task revolved around utilizing the STM32 Cube MX software for the STM32 microcontroller. Our primary objective was to establish a system clock (SYSCLK) with a frequency of 84 MHz and set the ADC1 sampling rate to a frequency of 100 kHz.

System Clock 84 MHz :

Using the Clock Configuration interface within the STM32 Cube MX software, we successfully configured the system clock to operate at 84 MHz. The detailed process can be visualized in the screenshot provided in Figure 1 (Point 1 Highlighted in Yellow).

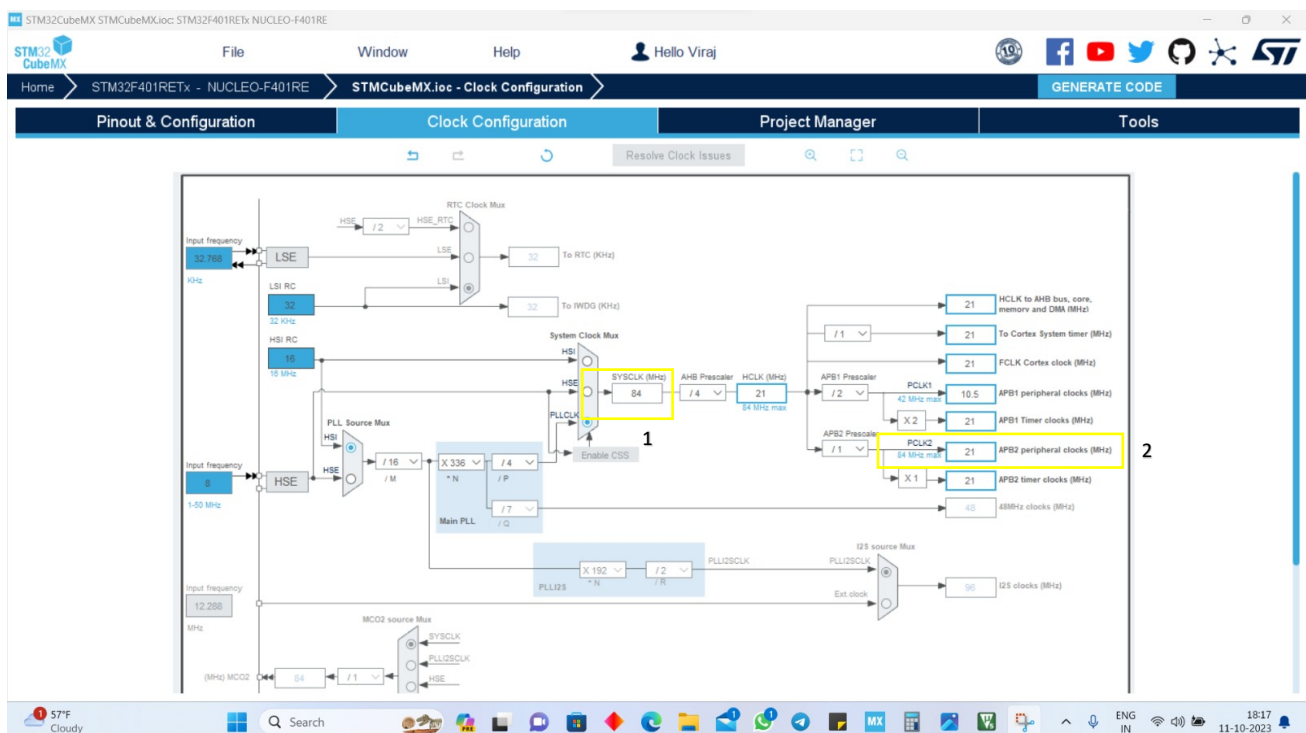


Figure 1 : SYS Clock configuration for 84 MHz

ADC1 Sampling at 100 KHz:

To ensure accurate ADC1 sampling, we took the following steps:

1. It's also worth noting that, for our sampling, we utilized the ADC1's IN0 channel, which is evident from the configurations displayed in Figure 2 (Point 1 Highlighted in Yellow).
2. We first confirmed that the APB2 (PCLK2) peripheral clock was directing the ADC1, as can be discerned from Figure 1 (Point 2 Highlighted in Yellow).
3. A prescaler was set for the APB2 frequency. This ensured that the PCLK (Peripheral Clock) feeding into the ADC1 was suitable to achieve our desired 100 kHz sampling rate. The exact configuration can be observed in Figure 2 (Point 2 Highlighted in Yellow).

4. Along with prescaler value, the resolution of ADC is set to 12 bits (15 ADC Clock Cycles) as seen in in Figure 2(Point 3 Highlighted in Yellow).
5. Sampling time was set to 15 Cycles as seen in in Figure 2(Point 4 Highlighted in Yellow).

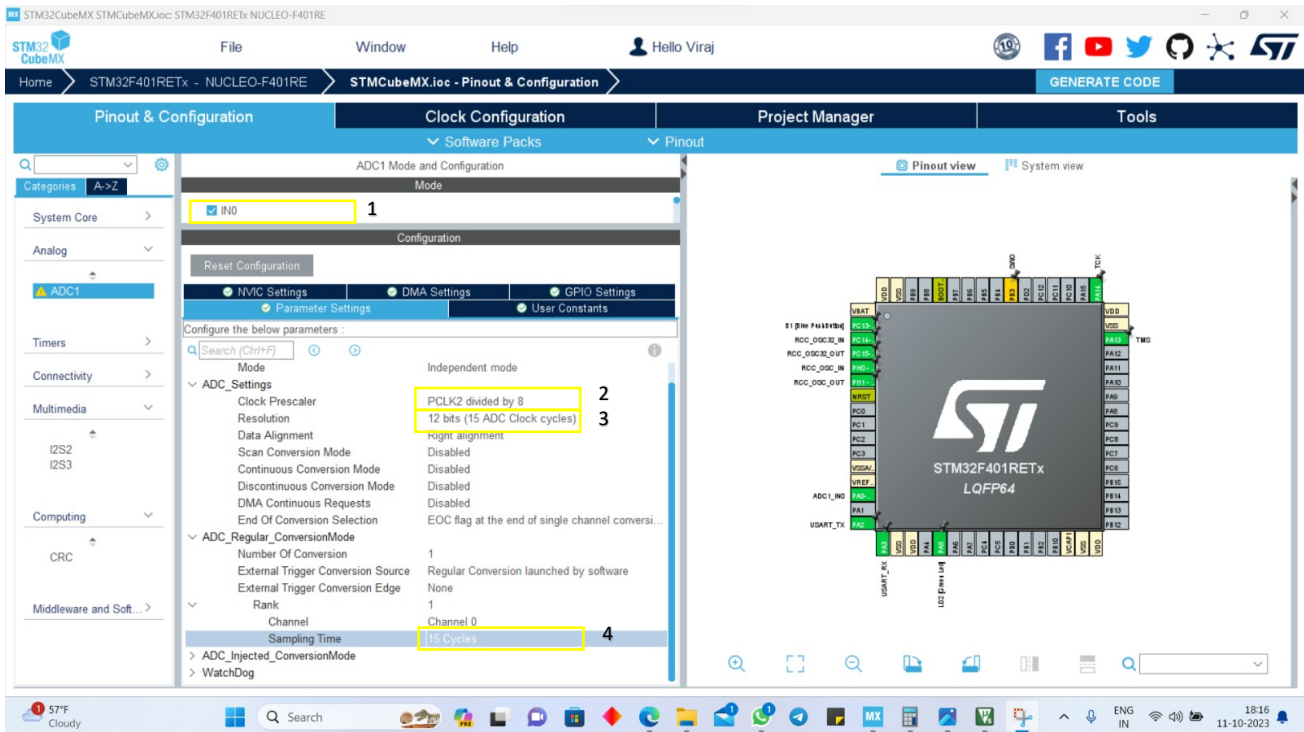


Figure 2 : ADC1 Configuration for its sampling frequency at 100KHz

ADC Sampling Rate Calculation :

When delving into the ADC section of the reference manual for the Nucleo 401-RE microcontroller, we uncovered vital guidelines on computing the sampling rate for an ADC channel. As per the documentation (highlighted in Figure 3 from the STM32F401RE Reference Manual), the ADC samples the input voltage based on the number of ADCCLOCK cycles. These cycles can be adjusted using the SMPR2 bits in the ADC_SMPR1 and ADC_SMPR2 registers. Every ADC channel can be set with a unique sampling time, and the manual provides a formula to calculate the total conversion time based on the chosen ADCCLOCK and sampling time.

RM0368

Analog-to-digital converter (ADC)

11.5 Channel-wise programmable sampling time

The ADC samples the input voltage for a number of ADCCLOCK cycles that can be modified using the SMP[2:0] bits in the ADC_SMPR1 and ADC_SMPR2 registers. Each channel can be sampled with a different sampling time.

The total conversion time is calculated as follows:

$$T_{\text{conv}} = \text{Sampling time} + 12 \text{ cycles}$$

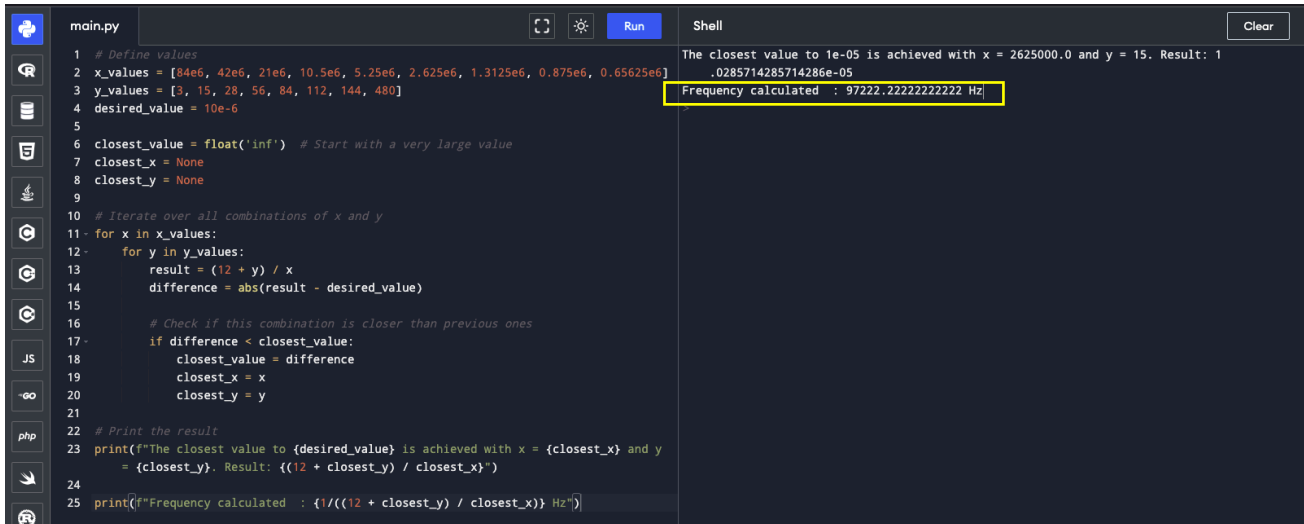
Example:

With ADCCLOCK = 30 MHz and sampling time = 3 cycles:

$$T_{\text{conv}} = 3 + 12 = 15 \text{ cycles} = 0.5 \mu\text{s with APB2 at 60 MHz}$$

Figure 3 : ADC Sampling Rate Calculation(STM32F401RE Reference Manual)

Venturing further, we tapped into the STM32 Cube MX IDE's Clock Configuration tab to extract all feasible clock configurations. To pinpoint the configuration that aligns most closely with our target of 10 μ sec, or 100 kHz, we employed a Python script. This script's workings and the results it rendered can be observed in Figure (4).



```

1 # Define values
2 x_values = [84e6, 42e6, 21e6, 10.5e6, 5.25e6, 2.625e6, 1.3125e6, 0.875e6, 0.65625e6]
3 y_values = [3, 15, 28, 56, 84, 112, 144, 480]
4 desired_value = 10e-6
5
6 closest_value = float('inf') # Start with a very large value
7 closest_x = None
8 closest_y = None
9
10 # Iterate over all combinations of x and y
11 for x in x_values:
12     for y in y_values:
13         result = (12 + y) / x
14         difference = abs(result - desired_value)
15
16         # Check if this combination is closer than previous ones
17         if difference < closest_value:
18             closest_value = difference
19             closest_x = x
20             closest_y = y
21
22 # Print the result
23 print(f"The closest value to {desired_value} is achieved with x = {closest_x} and y = {closest_y}. Result: {(12 + closest_y) / closest_x}")
24
25 print(f"Frequency calculated : {1/((12 + closest_y) / closest_x)} Hz")
  
```

The closest value to 1e-05 is achieved with x = 2625000.0 and y = 15. Result: 1.0285714285714286e-05
 Frequency calculated : 97222.22222222222 Hz

Figure 4 : Python script

Drawing insights from the Python script (shown in Figure 4) and integrating them with the formula we derived from Figure 3, we discerned that the optimal sampling time stands at 15. Moreover, with an ADC_CLK of 2.625 MHz (derived from a PCLK2 of 21 MHz and an ADC Clock Prescaler of 8), the resultant frequency was approximately 97.22 kHz. Impressively, this value is well within the permissible $\pm 5\%$ range of our target frequency of 100 kHz as said by Amey.

The same ADC and Clock configuration is also available in auto generated code obtained from STM Cube MX as seen in section [Code](#) in the this document.

Start-up Code Comparison :

To compare the start-up code generated from project 1 module 5 and one from previous assignments, we had to autogenerate this code for the Keil uVision. This was achieved using project manager window of STM Cube MX as seen in figure(5).

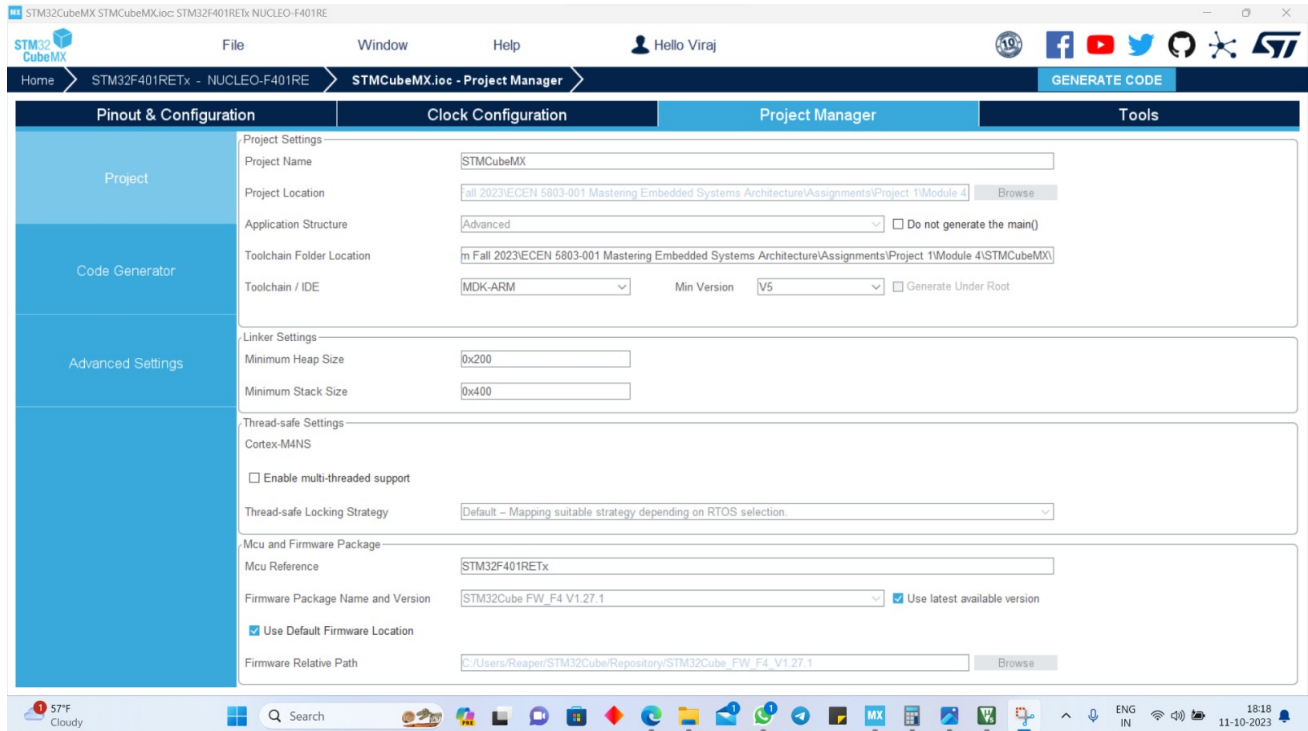


Figure 5 : Project Configuration

After looking into the file startup_stm32f401xe.s generated from STM Cube MX and from previous assignment , we found out that there was no change in the both the files. This implies that both the files remained invariant in terms of content, and even the comments embedded within them mirrored each other.

We have added the start-up file generated from this assignment and from the previous assignment in this document [here](#).

Memory Map:

At first, we looked at comparing the code sizes from both versions. But with each having different code logic, making a direct comparison wasn't meaningful.

It's crucial to point out that even with different code sizes, it doesn't automatically imply differences in the memory maps.

Furthermore, regarding hardware-specific details, aspects like image entry points and reset vector locations remain consistent. However, the difference in code size arises because, in this version, we are initializing the clock and ADC directly within the main function.

| Idx | Exec Addr | Load Addr | Size | Type | Attr |
|------|------------|------------|-----------------------|------|------|
| 1123 | 0x08000000 | 0x08000000 | 0x00000194 | Data | RO |
| 1124 | 3 | RESET | startup_stm32f401xe.o | | |
| 1125 | 0x08000194 | 0x08000194 | 0x00000008 | Code | RO |

Figure 6 : Image Entry Point & Reset Vector Location

| Statistic | Left Version (STM Cube MX) | Right Version (Blinky) |
|---|----------------------------|------------------------|
| Total RO Size (Code + RO Data) | 5128 (5.01kB) | 1568 (1.53kB) |
| Total RW Size (RW Data + ZI Data) | 1792 (1.75kB) | 1040 (1.02kB) |
| Total ROM Size (Code + RO Data + RW Data) | 5144 (5.02kB) | 1580 (1.54kB) |

Figure 7 : Code Size Comparison

The .map file for the code generated from STM Cube MX as well as from the previous assignments has been added in a zip file submitted over canvas.

Code :

1. Main.c from code generated from STMCube MX :

```
/* USER CODE BEGIN Header */
/**
 * @file      : main.c
 * @brief     : Main program body
 * @attention
 *
 * Copyright (c) 2023 STMicroelectronics.
 * All rights reserved.
 *
 * This software is licensed under terms that can be found in the LICENSE file
 * in the root directory of this software component.
 * If no LICENSE file comes with this software, it is provided AS-IS.
 */
/* USER CODE END Header */
/* Includes -----*/
#include "main.h"

/* Private includes -----*/
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

/* Private typedef -----*/
/* USER CODE BEGIN PTD */

/* USER CODE END PTD */

/* Private define -----*/
/* USER CODE BEGIN PD */

/* USER CODE END PD */

/* Private macro -----*/
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables -----*/
ADC_HandleTypeDef hadc1;

UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART2_UART_Init(void);
static void MX_ADC1_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code -----*/
/* USER CODE BEGIN 0 */

/* 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();
    MX_USART2_UART_Init();
    MX_ADC1_Init();
    /* USER CODE BEGIN 2 */

    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */
    while (1)
    {
        /* USER CODE END WHILE */

        /* USER CODE BEGIN 3 */
    }
    /* 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_SCALE2);

    /** 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_ON;
    RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
    RCC_OscInitStruct.PLL.PLLM = 16;
    RCC_OscInitStruct.PLL.PLLN = 336;
    RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;

```



```

RCC_OscInitStruct.PLL.PLLQ = 7;
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_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV4;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV2;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

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

/**
 * @brief ADC1 Initialization Function
 * @param None
 * @retval None
 */
static void MX_ADC1_Init(void)
{
    /* USER CODE BEGIN ADC1_Init 0 */

    /* USER CODE END ADC1_Init 0 */

    ADC_ChannelConfTypeDef sConfig = {0};

    /* USER CODE BEGIN ADC1_Init 1 */

    /* USER CODE END ADC1_Init 1 */

    /** Configure the global features of the ADC (Clock, Resolution, Data Alignment and number of conversion)
     */
    hadc1.Instance = ADC1;
    hadc1.Init.ClockPrescaler = ADC_CLOCK_SYNC_PCLK_DIV8;
    hadc1.Init.Resolution = ADC_RESOLUTION_12B;
    hadc1.Init.ScanConvMode = DISABLE;
    hadc1.Init.ContinuousConvMode = DISABLE;
    hadc1.Init.DiscontinuousConvMode = DISABLE;
    hadc1.Init.ExternalTrigConvEdge = ADC_EXTERNALTRIGCONVEDGE_NONE;
    hadc1.Init.ExternalTrigConv = ADC_SOFTWARE_START;
    hadc1.Init.DataAlign = ADC_DATAALIGN_RIGHT;
    hadc1.Init.NbrOfConversion = 1;
    hadc1.Init.DMAContinuousRequests = DISABLE;
    hadc1.Init.EOCSelection = ADC_EOC_SINGLE_CONV;
    if (HAL_ADC_Init(&hadc1) != HAL_OK)
    {
        Error_Handler();
    }

    /** Configure for the selected ADC regular channel its corresponding rank in the sequencer and its sample time.
     */
    sConfig.Channel = ADC_CHANNEL_0;
    sConfig.Rank = 1;
    sConfig.SamplingTime = ADC_SAMPLETIME_15CYCLES;
    if (HAL_ADC_ConfigChannel(&hadc1, &sConfig) != HAL_OK)
    {
        Error_Handler();
    }
    /* USER CODE BEGIN ADC1_Init 2 */

    /* USER CODE END ADC1_Init 2 */

```



```

}

/**
 * @brief USART2 Initialization Function
 * @param None
 * @retval None
 */
static void MX_USART2_UART_Init(void)
{

    /* USER CODE BEGIN USART2_Init 0 */

    /* USER CODE END USART2_Init 0 */

    /* USER CODE BEGIN USART2_Init 1 */

    /* USER CODE END USART2_Init 1 */
    huart2.Instance = USART2;
    huart2.Init.BaudRate = 115200;
    huart2.Init.WordLength = UART_WORDLENGTH_8B;
    huart2.Init.StopBits = UART_STOPBITS_1;
    huart2.Init.Parity = UART_PARITY_NONE;
    huart2.Init.Mode = UART_MODE_TX_RX;
    huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart2.Init.OverSampling = UART_OVERSAMPLING_16;
    if (HAL_UART_Init(&huart2) != HAL_OK)
    {
        Error_Handler();
    }
    /* USER CODE BEGIN USART2_Init 2 */

    /* USER CODE END USART2_Init 2 */

}

/**
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct = {0};
    /* USER CODE BEGIN MX_GPIO_Init_1 */
    /* USER CODE END MX_GPIO_Init_1 */

    /* GPIO Ports Clock Enable */
    __HAL_RCC_GPIOC_CLK_ENABLE();
    __HAL_RCC_GPIOH_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(LD2_GPIO_Port, LD2_Pin, GPIO_PIN_RESET);

    /*Configure GPIO pin : B1_Pin */
    GPIO_InitStruct.Pin = B1_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_IT_FALLING;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(B1_GPIO_Port, &GPIO_InitStruct);

    /*Configure GPIO pin : LD2_Pin */
    GPIO_InitStruct.Pin = LD2_Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
    HAL_GPIO_Init(LD2_GPIO_Port, &GPIO_InitStruct);

    /* USER CODE BEGIN MX_GPIO_Init_2 */

```

```

/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

/**
 * @brief This function is executed in case of error occurrence.
 * @retval None
 */
void Error_Handler(void)
{
    /* USER CODE BEGIN Error_Handler_Debug */
    /* User can add his own implementation to report the HAL error return state */
    __disable_irq();
    while (1)
    {
    }
    /* USER CODE END Error_Handler_Debug */
}

#ifdef USE_FULL_ASSERT
/**
 * @brief Reports the name of the source file and the source line number
 * where the assert_param error has occurred.
 * @param file: pointer to the source file name
 * @param line: assert_param error line source number
 * @retval None
 */
void assert_failed(uint8_t *file, uint32_t line)
{
    /* USER CODE BEGIN 6 */
    /* User can add his own implementation to report the file name and line number,
    ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */
    /* USER CODE END 6 */
}
#endif /* USE_FULL_ASSERT */

```

2. Start-up file startup_stm32f401xe.s

```

*****
; File Name      : startup_stm32f401xe.s
; Author        : MCD Application Team
; Description    : STM32F401xe devices vector table for MDK-ARM toolchain.
;
; This module performs:
; - Set the initial SP
; - Set the initial PC == Reset_Handler
; - Set the vector table entries with the exceptions ISR address
; - Branches to __main in the C library (which eventually
;   calls main()).
; After Reset the CortexM4 processor is in Thread mode,
; priority is Privileged, and the Stack is set to Main.
*****
; @attention
;
; Copyright (c) 2017 STMicroelectronics.
; All rights reserved.
;
; This software is licensed under terms that can be found in the LICENSE file
; in the root directory of this software component.
; If no LICENSE file comes with this software, it is provided AS-IS.
;
*****
; <<< Use Configuration Wizard in Context Menu >>>
;
; Amount of memory (in bytes) allocated for Stack
; Tailor this value to your application needs

```

```
; <h> Stack Configuration
; <o> Stack Size (in Bytes) <0x0-0xFFFFFFFF:8>
; </h>

Stack_Size      EQU    0x400

                AREA    STACK, NOINIT, READWRITE, ALIGN=3
Stack_Mem       SPACE   Stack_Size
__initial_sp


; <h> Heap Configuration
; <o> Heap Size (in Bytes) <0x0-0xFFFFFFFF:8>
; </h>

Heap_Size       EQU    0x200

                AREA    HEAP, NOINIT, READWRITE, ALIGN=3
__heap_base
Heap_Mem        SPACE   Heap_Size
__heap_limit


                PRESERVE8
                THUMB


; Vector Table Mapped to Address 0 at Reset
                AREA    RESET, DATA, READONLY
                EXPORT  __Vectors
                EXPORT  __Vectors_End
                EXPORT  __Vectors_Size

__Vectors       DCD    __initial_sp          ; Top of Stack
                DCD    Reset_Handler        ; Reset Handler
                DCD    NMI_Handler          ; NMI Handler
                DCD    HardFault_Handler    ; Hard Fault Handler
                DCD    MemManage_Handler    ; MPU Fault Handler
                DCD    BusFault_Handler     ; Bus Fault Handler
                DCD    UsageFault_Handler   ; Usage Fault Handler
                DCD    0                    ; Reserved
                DCD    0                    ; Reserved
                DCD    0                    ; Reserved
                DCD    0                    ; Reserved
                DCD    SVC_Handler          ; SVCall Handler
                DCD    DebugMon_Handler     ; Debug Monitor Handler
                DCD    0                    ; Reserved
                DCD    PendSV_Handler       ; PendSV Handler
                DCD    SysTick_Handler      ; SysTick Handler


; External Interrupts
                DCD    WWDG_IRQHandler      ; Window WatchDog
                DCD    PVD_IRQHandler       ; PVD through EXTI Line detection
                DCD    TAMP_STAMP_IRQHandler ; Tamper and TimeStamps through the EXTI line
                DCD    RTC_WKUP_IRQHandler  ; RTC Wakeup through the EXTI line
                DCD    FLASH_IRQHandler     ; FLASH
                DCD    RCC_IRQHandler       ; RCC
                DCD    EXTI0_IRQHandler     ; EXTI Line0
                DCD    EXTI1_IRQHandler     ; EXTI Line1
                DCD    EXTI2_IRQHandler     ; EXTI Line2
                DCD    EXTI3_IRQHandler     ; EXTI Line3
                DCD    EXTI4_IRQHandler     ; EXTI Line4
                DCD    DMA1_Stream0_IRQHandler ; DMA1 Stream 0
                DCD    DMA1_Stream1_IRQHandler ; DMA1 Stream 1
                DCD    DMA1_Stream2_IRQHandler ; DMA1 Stream 2
                DCD    DMA1_Stream3_IRQHandler ; DMA1 Stream 3
                DCD    DMA1_Stream4_IRQHandler ; DMA1 Stream 4
                DCD    DMA1_Stream5_IRQHandler ; DMA1 Stream 5
                DCD    DMA1_Stream6_IRQHandler ; DMA1 Stream 6
                DCD    ADC_IRQHandler       ; ADC1, ADC2 and ADC3s
                DCD    0                    ; Reserved
```

```

DCD 0 ; Reserved
DCD 0 ; Reserved
DCD 0 ; Reserved
DCD EXTI9_5_IRQHandler ; External Line[9:5]s
DCD TIM1_BRK_TIM9_IRQHandler ; TIM1 Break and TIM9
DCD TIM1_UP_TIM10_IRQHandler ; TIM1 Update and TIM10
DCD TIM1_TRG_COM_TIM11_IRQHandler ; TIM1 Trigger and Commutation and TIM11
DCD TIM1_CC_IRQHandler ; TIM1 Capture Compare
DCD TIM2_IRQHandler ; TIM2
DCD TIM3_IRQHandler ; TIM3
DCD TIM4_IRQHandler ; TIM4
DCD I2C1_EV_IRQHandler ; I2C1 Event
DCD I2C1_ER_IRQHandler ; I2C1 Error
DCD I2C2_EV_IRQHandler ; I2C2 Event
DCD I2C2_ER_IRQHandler ; I2C2 Error
DCD SPI1_IRQHandler ; SPI1
DCD SPI2_IRQHandler ; SPI2
DCD USART1_IRQHandler ; USART1
DCD USART2_IRQHandler ; USART2
DCD 0 ; Reserved
DCD EXTI15_10_IRQHandler ; External Line[15:10]s
DCD RTC_Alarm_IRQHandler ; RTC Alarm (A and B) through EXTI Line
DCD OTG_FS_WKUP_IRQHandler ; USB OTG FS Wakeup through EXTI line
DCD 0 ; Reserved
DCD 0 ; Reserved
DCD 0 ; Reserved
DCD 0 ; Reserved
DCD DMA1_Stream7_IRQHandler ; DMA1 Stream7
DCD 0 ; Reserved
DCD SDIO_IRQHandler ; SDIO
DCD TIM5_IRQHandler ; TIM5
DCD SPI3_IRQHandler ; SPI3
DCD 0 ; Reserved
DCD 0 ; Reserved
DCD 0 ; Reserved
DCD 0 ; Reserved
DCD DMA2_Stream0_IRQHandler ; DMA2 Stream 0
DCD DMA2_Stream1_IRQHandler ; DMA2 Stream 1
DCD DMA2_Stream2_IRQHandler ; DMA2 Stream 2
DCD DMA2_Stream3_IRQHandler ; DMA2 Stream 3
DCD DMA2_Stream4_IRQHandler ; DMA2 Stream 4
DCD 0 ; Reserved
DCD 0 ; Reserved
DCD 0 ; Reserved
DCD 0 ; Reserved
DCD 0 ; Reserved
DCD 0 ; Reserved
DCD OTG_FS_IRQHandler ; USB OTG FS
DCD DMA2_Stream5_IRQHandler ; DMA2 Stream 5
DCD DMA2_Stream6_IRQHandler ; DMA2 Stream 6
DCD DMA2_Stream7_IRQHandler ; DMA2 Stream 7
DCD USART6_IRQHandler ; USART6
DCD I2C3_EV_IRQHandler ; I2C3 event
DCD I2C3_ER_IRQHandler ; I2C3 error
DCD 0 ; Reserved
DCD 0 ; Reserved
DCD 0 ; Reserved
DCD 0 ; Reserved
DCD 0 ; Reserved
DCD 0 ; Reserved
DCD 0 ; Reserved
DCD FPU_IRQHandler ; FPU
DCD 0 ; Reserved
DCD 0 ; Reserved
DCD SPI4_IRQHandler ; SPI4

```

__Vectors_End

__Vectors_Size EQU __Vectors_End - __Vectors

AREA |.text|, CODE, READONLY

; Reset handler

```
Reset_Handler PROC
    EXPORT Reset_Handler    [WEAK]
    IMPORT SystemInit
    IMPORT __main

    LDR R0, =SystemInit
    BLX R0
    LDR R0, =__main
    BX  R0
ENDP
```

; Dummy Exception Handlers (infinite loops which can be modified)

```
NMI_Handler PROC
    EXPORT NMI_Handler      [WEAK]
    B     .
ENDP

HardFault_Handler\
PROC
    EXPORT HardFault_Handler    [WEAK]
    B     .
ENDP

MemManage_Handler\
PROC
    EXPORT MemManage_Handler    [WEAK]
    B     .
ENDP

BusFault_Handler\
PROC
    EXPORT BusFault_Handler     [WEAK]
    B     .
ENDP

UsageFault_Handler\
PROC
    EXPORT UsageFault_Handler   [WEAK]
    B     .
ENDP

SVC_Handler PROC
    EXPORT SVC_Handler          [WEAK]
    B     .
ENDP

DebugMon_Handler\
PROC
    EXPORT DebugMon_Handler     [WEAK]
    B     .
ENDP

PendSV_Handler PROC
    EXPORT PendSV_Handler       [WEAK]
    B     .
ENDP

SysTick_Handler PROC
    EXPORT SysTick_Handler      [WEAK]
    B     .
ENDP
```

Default_Handler PROC

```
    EXPORT WWDG_IRQHandler      [WEAK]
    EXPORT PVD_IRQHandler       [WEAK]
    EXPORT TAMP_STAMP_IRQHandler [WEAK]
    EXPORT RTC_WKUP_IRQHandler   [WEAK]
    EXPORT FLASH_IRQHandler      [WEAK]
    EXPORT RCC_IRQHandler        [WEAK]
    EXPORT EXTI0_IRQHandler      [WEAK]
    EXPORT EXTI1_IRQHandler      [WEAK]
    EXPORT EXTI2_IRQHandler      [WEAK]
    EXPORT EXTI3_IRQHandler      [WEAK]
```

```

EXPORT EXTI4_IRQHandler      [WEAK]
EXPORT DMA1_Stream0_IRQHandler [WEAK]
EXPORT DMA1_Stream1_IRQHandler [WEAK]
EXPORT DMA1_Stream2_IRQHandler [WEAK]
EXPORT DMA1_Stream3_IRQHandler [WEAK]
EXPORT DMA1_Stream4_IRQHandler [WEAK]
EXPORT DMA1_Stream5_IRQHandler [WEAK]
EXPORT DMA1_Stream6_IRQHandler [WEAK]
EXPORT ADC_IRQHandler        [WEAK]
EXPORT EXTI9_5_IRQHandler     [WEAK]
EXPORT TIM1_BRK_TIM9_IRQHandler [WEAK]
EXPORT TIM1_UP_TIM10_IRQHandler [WEAK]
EXPORT TIM1_TRG_COM_TIM11_IRQHandler [WEAK]
EXPORT TIM1_CC_IRQHandler     [WEAK]
EXPORT TIM2_IRQHandler        [WEAK]
EXPORT TIM3_IRQHandler        [WEAK]
EXPORT TIM4_IRQHandler        [WEAK]
EXPORT I2C1_EV_IRQHandler     [WEAK]
EXPORT I2C1_ER_IRQHandler     [WEAK]
EXPORT I2C2_EV_IRQHandler     [WEAK]
EXPORT I2C2_ER_IRQHandler     [WEAK]
EXPORT SPI1_IRQHandler        [WEAK]
EXPORT SPI2_IRQHandler        [WEAK]
EXPORT USART1_IRQHandler      [WEAK]
EXPORT USART2_IRQHandler      [WEAK]
EXPORT EXTI15_10_IRQHandler   [WEAK]
EXPORT RTC_Alarm_IRQHandler    [WEAK]
EXPORT OTG_FS_WKUP_IRQHandler [WEAK]
EXPORT DMA1_Stream7_IRQHandler [WEAK]
EXPORT SDIO_IRQHandler        [WEAK]
EXPORT TIM5_IRQHandler        [WEAK]
EXPORT SPI3_IRQHandler        [WEAK]
EXPORT DMA2_Stream0_IRQHandler [WEAK]
EXPORT DMA2_Stream1_IRQHandler [WEAK]
EXPORT DMA2_Stream2_IRQHandler [WEAK]
EXPORT DMA2_Stream3_IRQHandler [WEAK]
EXPORT DMA2_Stream4_IRQHandler [WEAK]
EXPORT OTG_FS_IRQHandler      [WEAK]
EXPORT DMA2_Stream5_IRQHandler [WEAK]
EXPORT DMA2_Stream6_IRQHandler [WEAK]
EXPORT DMA2_Stream7_IRQHandler [WEAK]
EXPORT USART6_IRQHandler      [WEAK]
EXPORT I2C3_EV_IRQHandler     [WEAK]
EXPORT I2C3_ER_IRQHandler     [WEAK]
EXPORT FPU_IRQHandler         [WEAK]
EXPORT SPI4_IRQHandler        [WEAK]

```

```

WWDG_IRQHandler
PVD_IRQHandler
TAMP_STAMP_IRQHandler
RTC_WKUP_IRQHandler
FLASH_IRQHandler
RCC_IRQHandler
EXTI0_IRQHandler
EXTI1_IRQHandler
EXTI2_IRQHandler
EXTI3_IRQHandler
EXTI4_IRQHandler
DMA1_Stream0_IRQHandler
DMA1_Stream1_IRQHandler
DMA1_Stream2_IRQHandler
DMA1_Stream3_IRQHandler
DMA1_Stream4_IRQHandler
DMA1_Stream5_IRQHandler
DMA1_Stream6_IRQHandler
ADC_IRQHandler
EXTI9_5_IRQHandler
TIM1_BRK_TIM9_IRQHandler
TIM1_UP_TIM10_IRQHandler
TIM1_TRG_COM_TIM11_IRQHandler

```

```
TIM1_CC_IRQHandler
TIM2_IRQHandler
TIM3_IRQHandler
TIM4_IRQHandler
I2C1_EV_IRQHandler
I2C1_ER_IRQHandler
I2C2_EV_IRQHandler
I2C2_ER_IRQHandler
SPI1_IRQHandler
SPI2_IRQHandler
USART1_IRQHandler
USART2_IRQHandler
EXTI15_10_IRQHandler
RTC_Alarm_IRQHandler
OTG_FS_WKUP_IRQHandler
DMA1_Stream7_IRQHandler
SDIO_IRQHandler
TIM5_IRQHandler
SPI3_IRQHandler
DMA2_Stream0_IRQHandler
DMA2_Stream1_IRQHandler
DMA2_Stream2_IRQHandler
DMA2_Stream3_IRQHandler
DMA2_Stream4_IRQHandler
OTG_FS_IRQHandler
DMA2_Stream5_IRQHandler
DMA2_Stream6_IRQHandler
DMA2_Stream7_IRQHandler
USART6_IRQHandler
I2C3_EV_IRQHandler
I2C3_ER_IRQHandler
FPU_IRQHandler
SPI4_IRQHandler
```

B .

ENDP

ALIGN

```
*****
; User Stack and Heap initialization
*****
IF :DEF:__MICROLIB

EXPORT __initial_sp
EXPORT __heap_base
EXPORT __heap_limit

ELSE

IMPORT __use_two_region_memory
EXPORT __user_initial_stackheap

__user_initial_stackheap

LDR R0, = Heap_Mem
LDR R1, =(Stack_Mem + Stack_Size)
LDR R2, =(Heap_Mem + Heap_Size)
LDR R3, = Stack_Mem
BX LR

ALIGN

ENDIF

END
```