# 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).

A computer screen shot of a computer program

Description automatically generated

1

2

Figure : 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).

A computer screen shot of a computer

Description automatically generated

4

3

2

1

Figure : 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.

A screenshot of a test

Description automatically generated

Figure : 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 µsec, or 100 kHz, we employed a Python script. This script's workings and the results it rendered can be observed in Figure (4).

A screenshot of a computer program

Description automatically generated

Figure : 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 ±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](#_Main.c_from_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).

A screenshot of a computer

Description automatically generated

Figure : 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](#_Start-up_file_startup_stm32f401xe.s).

## 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.

A screenshot of a computer

Description automatically generated

Figure : Image Entry Point & Reset Vector Location

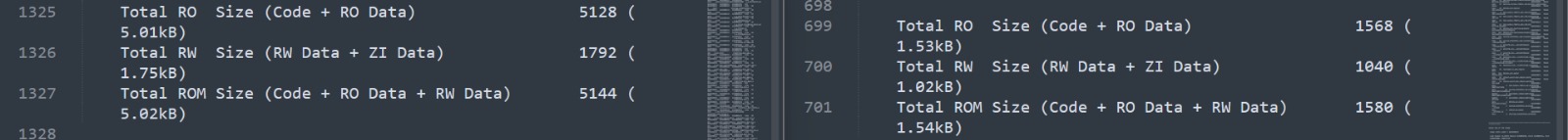


Figure : Code Size Comparison

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 :

## 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 \*/

## 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