

```

/* USER CODE BEGIN Header */
/**
 * *****
 * @file           : main.c
 * @brief          : Main program body
 * *****
 * @attention
 *
 * <h2><center>&copy; Copyright (c) 2024 STMicroelectronics.
 * All rights reserved.</center></h2>
 *
 * This software component is licensed by ST under BSD 3-Clause license,
 * the "License"; You may not use this file except in compliance with the
 * License. You may obtain a copy of the License at:
 *
 *             opensource.org/licenses/BSD-3-Clause
 * *****
 */
/* 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 ----- */
UART_HandleTypeDef huart3;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes ----- */
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_USART3_UART_Init(void);
/* USER CODE BEGIN PFP */

void USART_Send_Char(char x){
    USART3->DR = x;
    while(!(USART3->SR &(1<<6))){
    }
}

void USART_Send_Str(char *x){
    int i = 0;
    while(x[i]!='\0'){
        USART_Send_Char(x[i]);
    }
}

```

```

        i+=1;
    }
}

int _write(int file, char *ptr, int len)
{
    int DataIdx;

    for (DataIdx = 0; DataIdx < len; DataIdx++)
    {
        ITM_SendChar(*ptr++);
    }
    return len;
}
/* 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 */

    RCC->AHB1ENR = (1 << 2) ;
    RCC->APB1ENR = (1 << 18);
    GPIOC->MODER = (2 << 20) | (2 << 22) ;
    GPIOC->AFR[1] = (7 << 8) | (7 << 12) ;
    USART3->CR1 = (1 << 13) | (1 << 3) | (1 << 2);
    USART3->BRR = (7 << 0) | (24 << 4) ;

    /* 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_USART3_UART_Init();
    /* USER CODE BEGIN 2 */

    /* USER CODE END 2 */

    /* Infinite loop */
    /* USER CODE BEGIN WHILE */

```

```

while (1)
{
    printf("ADITYA");
    USART_Send_Str("ADITYA\n\r");
    HAL_Delay(100);
    /* 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_SCALE1);
    /** Initializes the RCC Oscillators according to the specified parameters
    * in the RCC_OscInitTypeDef structure.
    */
    RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
    RCC_OscInitStruct.HSISState = 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();
    }
}

/**
 * @brief USART3 Initialization Function
 * @param None
 * @retval None
 */
static void MX_USART3_UART_Init(void)
{
    /* USER CODE BEGIN USART3_Init 0 */

    /* USER CODE END USART3_Init 0 */

    /* USER CODE BEGIN USART3_Init 1 */

```

```

/* USER CODE END USART3_Init 1 */
huart3.Instance = USART3;
huart3.Init.BaudRate = 115200;
huart3.Init.WordLength = UART_WORDLENGTH_8B;
huart3.Init.StopBits = UART_STOPBITS_1;
huart3.Init.Parity = UART_PARITY_NONE;
huart3.Init.Mode = UART_MODE_TX_RX;
huart3.Init.HwFlowCtl = UART_HWCONTROL_NONE;
huart3.Init.OverSampling = UART_OVERSAMPLING_16;
if (HAL_UART_Init(&huart3) != HAL_OK)
{
    Error_Handler();
}
/* USER CODE BEGIN USART3_Init 2 */

/* USER CODE END USART3_Init 2 */

}

/**
 * @brief GPIO Initialization Function
 * @param None
 * @retval None
 */
static void MX_GPIO_Init(void)
{
    /* GPIO Ports Clock Enable */
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOC_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();
}

/* 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 */

/***** (C) COPYRIGHT STMicroelectronics *****/
```