Main.c

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) .
  * 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.
  *
 *
******************************************************************************
      * PROJECT: SMART FAN CONTROL SYSTEM
      * AUTHOR:  ISHA SHARMA
      * COURSE:  EMBEDDED SYSTEMS DESIGN SP'23
      * TOOLS:   STMCUBEIDE
      * DATE:    06-05-2023
      * FILE:    main.c
      * BRIEF:   This is the main.c
      *
      *
******************************************************************************
      *
      * References used for the project:
      * [1] https://www.youtube.com/watch?v=M5ddTjrcvEs&t=54s
      * [2] https://microcontrollerslab.com/dc-motor-l293d-motor-driver-ic-
arduino-tutorial/
      * [3] https://microcontrollerslab.com/l293d-motor-driver-ic-introduction-
pinouts-example/
      * [4] https://www.youtube.com/watch?v=4jcxeJxvi3Y
      * [5] https://www.ti.com/lit/ds/symlink/ads1115.pdf
      * [6]
https://www.digikey.com/htmldatasheets/production/2047793/0/0/1/SSD1306.pdf
      * [7] https://thecodeprogram.com/ads1115-with-stm32-cubemx
      * [8] https://cdn-shop.adafruit.com/datasheets/l293d.pdf
      * [9] STM32F411VE Datasheet, Reference manual, and User Manuals
      * [10] https://www.thomasnet.com/articles/instruments-controls/types-of-
motor-controllers-and-drives/
      * [11] https://microcontrollerslab.com/analog-to-digital-adc-converter-
working/
      * [12] https://deepbluembedded.com/stm32-pwm-example-timer-pwm-mode-
tutorial/
      * [13] https://deepbluembedded.com/?s=dc+motor
      * [14] https://controllerstech.com/oled-display-using-i2c-stm32/
      * [15] Professor, TAs and CLassmates
      *
      *
      *
 */
/* USER CODE END Header */
/* Includes ------------------------------------------------------------------*/
#include "main.h"
```

```c
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include "fonts.h"
#include "ssd1306.h"
#include "dcmotor.h"
#include "led.h"
/* 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 ---------------------------------------------------------*/
I2C_HandleTypeDef hi2c1;
I2C_HandleTypeDef hi2c2;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);
static void MX_I2C2_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
#define SLAVE_ADDRESS 0x48 //SLAVE ADDRESS FOR I2C SLAVE 1001000

/* Variables*/
unsigned char ADCwrite[6]; //buffer to write to pointer register
int16_t reading_from_pot; //variable for raw reading of pot
float voltage_from_pot; //variable for converted voltage value from the raw
reading
const float voltage_conversion = 6.114 / 32768.0; //PGA FOR MORE THAN 4 WAS
6.114, FOR BITS X2
uint32_t duty_cycle; //variable to determine duty cycle for pwm

/* USER CODE END 0 */

/**
  * @brief  The application entry point.
  * @retval int
```

```c
  */
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_I2C1_Init();
  MX_I2C2_Init();
  /* USER CODE BEGIN 2 */

  /* initialse pwm, motor direction pins, led and ssd1306 */
  init_gpio_pwmpin();
  init_pwm_timer();
  init_motor_pins();
  init_led();
  SSD1306_Init(); //initialising display

  //display startup message and led startup sequence
  SSD1306_GotoXY(0,0);
  SSD1306_Puts("SMART FAN",&Font_11x18,1);
  SSD1306_GotoXY(0,30);
  SSD1306_Puts("CONTROL SYSTEM",&Font_11x18,1);
  SSD1306_UpdateScreen(); //update display on OLED

  GPIOA->BSRR |= LED_BSRR_ON;
  HAL_Delay(500);
  GPIOA->BSRR |= LED_BSRR_OFF;
  HAL_Delay(500);
  GPIOA->BSRR |= LED_BSRR_ON;
  HAL_Delay(500);
  GPIOA->BSRR |= LED_BSRR_OFF;

  SSD1306_ScrollRight(0x00,0x0f); //scroll right entire screen using pages
  HAL_Delay(2000); //2 sec

  SSD1306_ScrollLeft(0x00,0x0f); //scroll left entire screen using pages
  HAL_Delay(2000); //2 sec

  SSD1306_Stopscroll();
  SSD1306_Clear();
```

```
  /* USER CODE END 2 */

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

    /* USER CODE BEGIN 3 */

        //SET CONFIG REG AND CONVERSION REG OF ADC
        ADCwrite[0] = 0x01; // CONVERION REG=00, CONFIG REG =01
        ADCwrite[1]=0xC1; //REGISTER FIELDS: os-1,mux-100,pga,000,mode-1
(11000001) msbs
        ADCwrite[2]= 0x83; // REGISTER FIELDS: dr-default-100,comp-alldefault
00011 (10000011)
        HAL_I2C_Master_Transmit(&hi2c2,SLAVE_ADDRESS<<1,ADCwrite,3,100); //hal
I2C transmit with timeout=100ms

        //retrieve data from conversion reg
        ADCwrite[0] = 0x00; // CONVERION REG=00, CONFIG REG =01
        HAL_I2C_Master_Transmit(&hi2c2,SLAVE_ADDRESS<<1,ADCwrite,1,100); //hal
I2C transmit with timeout=100ms
        HAL_Delay(20);
        HAL_I2C_Master_Receive(&hi2c2,SLAVE_ADDRESS<<1,ADCwrite,2,100); //hal
I2C receive with timeout=100ms

        //conversion to voltage, TAKING 8 BITS SHIFTING, OTHER sequentially
        reading_from_pot = (ADCwrite[0]<<8 | ADCwrite[1]);
        if (reading_from_pot<0)
        {
              reading_from_pot=0;// truncating the negative values just incase
        }

        //converted voltage value
        voltage_from_pot = reading_from_pot * voltage_conversion; //MAX VALUE
4.75, MIN VALUE 0.0

        //pwm duty cycle updation
        duty_cycle = (uint32_t)(voltage_from_pot * 1000 / 4.75);
        TIM4->CCR4 = duty_cycle;

        if(voltage_from_pot>0.2)
        {
                GPIOA->BSRR |= LED_BSRR_ON; //fan on
        }
        else
        {
                GPIOA->BSRR |= LED_BSRR_OFF; //fan off
        }

        if((voltage_from_pot>2.5)&&(voltage_from_pot<4.5))
        {
                SSD1306_GotoXY(0,0);
                SSD1306_Puts("SPEED: 2",&Font_11x18,1);
                SSD1306_UpdateScreen(); //update display on OLED
                HAL_Delay(500);
                SSD1306_Clear();
```

```
        }
        else if(voltage_from_pot>4.5)
        {
                SSD1306_GotoXY(0,0);
                SSD1306_Puts("MAX SPEED",&Font_11x18,1);
                SSD1306_UpdateScreen(); //update display on OLED
                HAL_Delay(500);
                SSD1306_Clear();

        }
        else if((voltage_from_pot>0.2)&&(voltage_from_pot<2.5))
        {
                SSD1306_GotoXY(0,0);
                SSD1306_Puts("SPEED: 1",&Font_11x18,1);
                SSD1306_UpdateScreen(); //update display on OLED
                HAL_Delay(500);
                SSD1306_Clear();
        }
        else
        {
                SSD1306_GotoXY(0,0);
                SSD1306_Puts("FAN OFF",&Font_11x18,1);
                SSD1306_UpdateScreen(); //update display on OLED
                HAL_Delay(500);
                SSD1306_Clear();

        }

  }
  /* 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_ON;
  RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSI;
  RCC_OscInitStruct.PLL.PLLM = 8;
  RCC_OscInitStruct.PLL.PLLN = 96;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
```

```c
  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_DIV2;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV4;

  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_1) != HAL_OK)
  {
    Error_Handler();
  }
}

/**
  * @brief I2C1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_I2C1_Init(void)
{

  /* USER CODE BEGIN I2C1_Init 0 */

  /* USER CODE END I2C1_Init 0 */

  /* USER CODE BEGIN I2C1_Init 1 */

  /* USER CODE END I2C1_Init 1 */
  hi2c1.Instance = I2C1;
  hi2c1.Init.ClockSpeed = 400000;
  hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
  hi2c1.Init.OwnAddress1 = 0;
  hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  hi2c1.Init.OwnAddress2 = 0;
  hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN I2C1_Init 2 */

  /* USER CODE END I2C1_Init 2 */

}

/**
  * @brief I2C2 Initialization Function
  * @param None
  * @retval None
  */
```

```c
static void MX_I2C2_Init(void)
{

  /* USER CODE BEGIN I2C2_Init 0 */

  /* USER CODE END I2C2_Init 0 */

  /* USER CODE BEGIN I2C2_Init 1 */

  /* USER CODE END I2C2_Init 1 */
  hi2c2.Instance = I2C2;
  hi2c2.Init.ClockSpeed = 400000;
  hi2c2.Init.DutyCycle = I2C_DUTYCYCLE_2;
  hi2c2.Init.OwnAddress1 = 0;
  hi2c2.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
  hi2c2.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
  hi2c2.Init.OwnAddress2 = 0;
  hi2c2.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
  hi2c2.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
  if (HAL_I2C_Init(&hi2c2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN I2C2_Init 2 */

  /* USER CODE END I2C2_Init 2 */

}

/**
  * @brief GPIO Initialization Function
  * @param None
  * @retval None
  */
static void MX_GPIO_Init(void)
{
/* 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_GPIOB_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();

/* 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 */
```

Main.h

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.h
  * @brief          : Header for main.c file.
  *                   This file contains the common defines of the application.
  ******************************************************************************
  * @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.
  *
  ******************************************************************************

*******************************************************************************
*******************************************************************************
       * PROJECT: SMART FAN CONTROL SYSTEM
       * AUTHOR:  ISHA SHARMA
       * COURSE:  EMBEDDED SYSTEMS DESIGN SP'23
       * TOOLS:   STMCUBEIDE
       * DATE:    06-05-2023
       * FILE:    main.h
       * BRIEF:   This is header for the main.c
       *
```

```
  *
******************************************************************************
  */
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __MAIN_H
#define __MAIN_H

#ifdef __cplusplus
extern "C" {
#endif

/* Includes ------------------------------------------------------------------*/
#include "stm32f4xx_hal.h"

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

/* USER CODE END Includes */

/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */

/* USER CODE END ET */

/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */

/* USER CODE END EC */

/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */

/* USER CODE END EM */

/* Exported functions prototypes ---------------------------------------------*/
void Error_Handler(void);

/* USER CODE BEGIN EFP */

/* USER CODE END EFP */

/* Private defines -----------------------------------------------------------*/

/* USER CODE BEGIN Private defines */

/* USER CODE END Private defines */

#ifdef __cplusplus
}
#endif

#endif /* __MAIN_H */
```

Fonts.c

```
/**
 * Reference and token of thanks to code written by Alexander Lutsai, Tilen Majerle
 * https://controllerstech.com/oled-display-using-i2c-stm32/
 *
*******************************************************************************
      * PROJECT: SMART FAN CONTROL SYSTEM
      * AUTHOR:  ISHA SHARMA
      * COURSE:  EMBEDDED SYSTEMS DESIGN SP'23
      * TOOLS:   STMCUBEIDE
      * DATE:    06-05-2023
      * FILE:    fonts.c
      * BRIEF:   This file is used for fonts library to be used on the OLED
      *
      *
*******************************************************************************
 */

#include "fonts.h"

const uint16_t Font7x10 [] = {
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
// sp
0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x0000, 0x1000, 0x0000, 0x0000,
// !
0x2800, 0x2800, 0x2800, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
// "
0x2400, 0x2400, 0x7C00, 0x2400, 0x4800, 0x7C00, 0x4800, 0x4800, 0x0000, 0x0000,
// #
0x3800, 0x5400, 0x5000, 0x3800, 0x1400, 0x5400, 0x5400, 0x3800, 0x1000, 0x0000,
// $
0x2000, 0x5400, 0x5800, 0x3000, 0x2800, 0x5400, 0x1400, 0x0800, 0x0000, 0x0000,
// %
0x1000, 0x2800, 0x2800, 0x1000, 0x3400, 0x4800, 0x4800, 0x3400, 0x0000, 0x0000,
// &
0x1000, 0x1000, 0x1000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
// '
0x0800, 0x1000, 0x2000, 0x2000, 0x2000, 0x2000, 0x2000, 0x2000, 0x1000, 0x0800,
// (
0x2000, 0x1000, 0x0800, 0x0800, 0x0800, 0x0800, 0x0800, 0x0800, 0x1000, 0x2000,
// )
0x1000, 0x3800, 0x1000, 0x2800, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
// *
0x0000, 0x0000, 0x1000, 0x1000, 0x7C00, 0x1000, 0x1000, 0x0000, 0x0000, 0x0000,
// +
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x1000, 0x1000, 0x1000,
// ,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x3800, 0x0000, 0x0000, 0x0000, 0x0000,
// -
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x1000, 0x0000, 0x0000,
// .
0x0800, 0x0800, 0x1000, 0x1000, 0x1000, 0x1000, 0x2000, 0x2000, 0x0000, 0x0000,
// /
0x3800, 0x4400, 0x4400, 0x5400, 0x4400, 0x4400, 0x4400, 0x3800, 0x0000, 0x0000,
// 0
0x1000, 0x3000, 0x5000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x0000, 0x0000,
// 1
0x3800, 0x4400, 0x4400, 0x0400, 0x0800, 0x1000, 0x2000, 0x7C00, 0x0000, 0x0000,
// 2
```

```
0x3800, 0x4400, 0x0400, 0x1800, 0x0400, 0x0400, 0x4400, 0x3800, 0x0000, 0x0000,
// 3
0x0800, 0x1800, 0x2800, 0x2800, 0x4800, 0x7C00, 0x0800, 0x0800, 0x0000, 0x0000,
// 4
0x7C00, 0x4000, 0x4000, 0x7800, 0x0400, 0x0400, 0x4400, 0x3800, 0x0000, 0x0000,
// 5
0x3800, 0x4400, 0x4000, 0x7800, 0x4400, 0x4400, 0x4400, 0x3800, 0x0000, 0x0000,
// 6
0x7C00, 0x0400, 0x0800, 0x1000, 0x1000, 0x2000, 0x2000, 0x2000, 0x0000, 0x0000,
// 7
0x3800, 0x4400, 0x4400, 0x3800, 0x4400, 0x4400, 0x4400, 0x3800, 0x0000, 0x0000,
// 8
0x3800, 0x4400, 0x4400, 0x4400, 0x3C00, 0x0400, 0x4400, 0x3800, 0x0000, 0x0000,
// 9
0x0000, 0x0000, 0x1000, 0x0000, 0x0000, 0x0000, 0x0000, 0x1000, 0x0000, 0x0000,
// :
0x0000, 0x0000, 0x0000, 0x1000, 0x0000, 0x0000, 0x0000, 0x1000, 0x1000, 0x1000,
// ;
0x0000, 0x0000, 0x0C00, 0x3000, 0x4000, 0x3000, 0x0C00, 0x0000, 0x0000, 0x0000,
// <
0x0000, 0x0000, 0x0000, 0x7C00, 0x0000, 0x7C00, 0x0000, 0x0000, 0x0000, 0x0000,
// =
0x0000, 0x0000, 0x6000, 0x1800, 0x0400, 0x1800, 0x6000, 0x0000, 0x0000, 0x0000,
// >
0x3800, 0x4400, 0x0400, 0x0800, 0x1000, 0x1000, 0x0000, 0x1000, 0x0000, 0x0000,
// ?
0x3800, 0x4400, 0x4C00, 0x5400, 0x5C00, 0x4000, 0x4000, 0x3800, 0x0000, 0x0000,
// @
0x1000, 0x2800, 0x2800, 0x2800, 0x2800, 0x7C00, 0x4400, 0x4400, 0x0000, 0x0000,
// A
0x7800, 0x4400, 0x4400, 0x7800, 0x4400, 0x4400, 0x4400, 0x7800, 0x0000, 0x0000,
// B
0x3800, 0x4400, 0x4000, 0x4000, 0x4000, 0x4000, 0x4400, 0x3800, 0x0000, 0x0000,
// C
0x7000, 0x4800, 0x4400, 0x4400, 0x4400, 0x4400, 0x4800, 0x7000, 0x0000, 0x0000,
// D
0x7C00, 0x4000, 0x4000, 0x7C00, 0x4000, 0x4000, 0x4000, 0x7C00, 0x0000, 0x0000,
// E
0x7C00, 0x4000, 0x4000, 0x7800, 0x4000, 0x4000, 0x4000, 0x4000, 0x0000, 0x0000,
// F
0x3800, 0x4400, 0x4000, 0x4000, 0x5C00, 0x4400, 0x4400, 0x3800, 0x0000, 0x0000,
// G
0x4400, 0x4400, 0x4400, 0x7C00, 0x4400, 0x4400, 0x4400, 0x4400, 0x0000, 0x0000,
// H
0x3800, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x3800, 0x0000, 0x0000,
// I
0x0400, 0x0400, 0x0400, 0x0400, 0x0400, 0x0400, 0x4400, 0x3800, 0x0000, 0x0000,
// J
0x4400, 0x4800, 0x5000, 0x6000, 0x5000, 0x4800, 0x4800, 0x4400, 0x0000, 0x0000,
// K
0x4000, 0x4000, 0x4000, 0x4000, 0x4000, 0x4000, 0x4000, 0x7C00, 0x0000, 0x0000,
// L
0x4400, 0x6C00, 0x6C00, 0x5400, 0x4400, 0x4400, 0x4400, 0x4400, 0x0000, 0x0000,
// M
0x4400, 0x6400, 0x6400, 0x5400, 0x5400, 0x4C00, 0x4C00, 0x4400, 0x0000, 0x0000,
// N
0x3800, 0x4400, 0x4400, 0x4400, 0x4400, 0x4400, 0x4400, 0x3800, 0x0000, 0x0000,
// O
```

```
0x7800, 0x4400, 0x4400, 0x4400, 0x7800, 0x4000, 0x4000, 0x4000, 0x0000, 0x0000,
// P
0x3800, 0x4400, 0x4400, 0x4400, 0x4400, 0x4400, 0x5400, 0x3800, 0x0400, 0x0000,
// Q
0x7800, 0x4400, 0x4400, 0x4400, 0x7800, 0x4800, 0x4800, 0x4400, 0x0000, 0x0000,
// R
0x3800, 0x4400, 0x4000, 0x3000, 0x0800, 0x0400, 0x4400, 0x3800, 0x0000, 0x0000,
// S
0x7C00, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x0000, 0x0000,
// T
0x4400, 0x4400, 0x4400, 0x4400, 0x4400, 0x4400, 0x4400, 0x3800, 0x0000, 0x0000,
// U
0x4400, 0x4400, 0x4400, 0x2800, 0x2800, 0x2800, 0x1000, 0x1000, 0x0000, 0x0000,
// V
0x4400, 0x4400, 0x5400, 0x5400, 0x5400, 0x6C00, 0x2800, 0x2800, 0x0000, 0x0000,
// W
0x4400, 0x2800, 0x2800, 0x1000, 0x1000, 0x2800, 0x2800, 0x4400, 0x0000, 0x0000,
// X
0x4400, 0x4400, 0x2800, 0x2800, 0x1000, 0x1000, 0x1000, 0x1000, 0x0000, 0x0000,
// Y
0x7C00, 0x0400, 0x0800, 0x1000, 0x1000, 0x2000, 0x4000, 0x7C00, 0x0000, 0x0000,
// Z
0x1800, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1800,
// [
0x2000, 0x2000, 0x1000, 0x1000, 0x1000, 0x1000, 0x0800, 0x0800, 0x0000, 0x0000,
/* \ */
0x3000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x3000,
// ]
0x1000, 0x2800, 0x2800, 0x4400, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
// ^
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0xFE00,
// _
0x2000, 0x1000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
// `
0x0000, 0x0000, 0x3800, 0x4400, 0x3C00, 0x4400, 0x4C00, 0x3400, 0x0000, 0x0000,
// a
0x4000, 0x4000, 0x5800, 0x6400, 0x4400, 0x4400, 0x6400, 0x5800, 0x0000, 0x0000,
// b
0x0000, 0x0000, 0x3800, 0x4400, 0x4000, 0x4000, 0x4400, 0x3800, 0x0000, 0x0000,
// c
0x0400, 0x0400, 0x3400, 0x4C00, 0x4400, 0x4400, 0x4C00, 0x3400, 0x0000, 0x0000,
// d
0x0000, 0x0000, 0x3800, 0x4400, 0x7C00, 0x4000, 0x4400, 0x3800, 0x0000, 0x0000,
// e
0x0C00, 0x1000, 0x7C00, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x0000, 0x0000,
// f
0x0000, 0x0000, 0x3400, 0x4C00, 0x4400, 0x4400, 0x4C00, 0x3400, 0x0400, 0x7800,
// g
0x4000, 0x4000, 0x5800, 0x6400, 0x4400, 0x4400, 0x4400, 0x4400, 0x0000, 0x0000,
// h
0x1000, 0x0000, 0x7000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x0000, 0x0000,
// i
0x1000, 0x0000, 0x7000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0xE000,
// j
0x4000, 0x4000, 0x4800, 0x5000, 0x6000, 0x5000, 0x4800, 0x4400, 0x0000, 0x0000,
// k
0x7000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x0000, 0x0000,
// l
```

```
0x0000, 0x0000, 0x7800, 0x5400, 0x5400, 0x5400, 0x5400, 0x5400, 0x0000, 0x0000,
// m
0x0000, 0x0000, 0x5800, 0x6400, 0x4400, 0x4400, 0x4400, 0x4400, 0x0000, 0x0000,
// n
0x0000, 0x0000, 0x3800, 0x4400, 0x4400, 0x4400, 0x4400, 0x3800, 0x0000, 0x0000,
// o
0x0000, 0x0000, 0x5800, 0x6400, 0x4400, 0x4400, 0x6400, 0x5800, 0x4000, 0x4000,
// p
0x0000, 0x0000, 0x3400, 0x4C00, 0x4400, 0x4400, 0x4C00, 0x3400, 0x0400, 0x0400,
// q
0x0000, 0x0000, 0x5800, 0x6400, 0x4000, 0x4000, 0x4000, 0x4000, 0x0000, 0x0000,
// r
0x0000, 0x0000, 0x3800, 0x4400, 0x3000, 0x0800, 0x4400, 0x3800, 0x0000, 0x0000,
// s
0x2000, 0x2000, 0x7800, 0x2000, 0x2000, 0x2000, 0x2000, 0x1800, 0x0000, 0x0000,
// t
0x0000, 0x0000, 0x4400, 0x4400, 0x4400, 0x4400, 0x4C00, 0x3400, 0x0000, 0x0000,
// u
0x0000, 0x0000, 0x4400, 0x4400, 0x2800, 0x2800, 0x2800, 0x1000, 0x0000, 0x0000,
// v
0x0000, 0x0000, 0x5400, 0x5400, 0x5400, 0x6C00, 0x2800, 0x2800, 0x0000, 0x0000,
// w
0x0000, 0x0000, 0x4400, 0x2800, 0x1000, 0x1000, 0x2800, 0x4400, 0x0000, 0x0000,
// x
0x0000, 0x0000, 0x4400, 0x4400, 0x2800, 0x2800, 0x1000, 0x1000, 0x1000, 0x6000,
// y
0x0000, 0x0000, 0x7C00, 0x0800, 0x1000, 0x2000, 0x4000, 0x7C00, 0x0000, 0x0000,
// z
0x1800, 0x1000, 0x1000, 0x1000, 0x2000, 0x2000, 0x1000, 0x1000, 0x1000, 0x1800,
// {
0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000, 0x1000,
// |
0x3000, 0x1000, 0x1000, 0x1000, 0x0800, 0x0800, 0x1000, 0x1000, 0x1000, 0x3000,
// }
0x0000, 0x0000, 0x0000, 0x7400, 0x4C00, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
// ~
};

const uint16_t Font11x18 [] = {
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // sp
0x0000, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00,
0x0C00, 0x0C00, 0x0000, 0x0C00, 0x0C00, 0x0000, 0x0000, 0x0000,   // !
0x0000, 0x1B00, 0x1B00, 0x1B00, 0x1B00, 0x1B00, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // "
0x0000, 0x1980, 0x1980, 0x1980, 0x1980, 0x7FC0, 0x7FC0, 0x1980, 0x3300, 0x7FC0,
0x7FC0, 0x3300, 0x3300, 0x3300, 0x3300, 0x0000, 0x0000, 0x0000,   // #
0x0000, 0x1E00, 0x3F00, 0x7580, 0x6580, 0x7400, 0x3C00, 0x1E00, 0x0700, 0x0580,
0x6580, 0x6580, 0x7580, 0x3F00, 0x1E00, 0x0400, 0x0400, 0x0000,   // $
0x0000, 0x7000, 0xD800, 0xD840, 0xD8C0, 0xD980, 0x7300, 0x0600, 0x0C00, 0x1B80,
0x36C0, 0x66C0, 0x46C0, 0x06C0, 0x0380, 0x0000, 0x0000, 0x0000,   // %
0x0000, 0x1E00, 0x3F00, 0x3300, 0x3300, 0x3300, 0x1E00, 0x0C00, 0x3CC0, 0x66C0,
0x6380, 0x6180, 0x6380, 0x3EC0, 0x1C80, 0x0000, 0x0000, 0x0000,   // &
0x0000, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // '
0x0080, 0x0100, 0x0300, 0x0600, 0x0600, 0x0400, 0x0C00, 0x0C00, 0x0C00, 0x0C00,
0x0C00, 0x0C00, 0x0400, 0x0600, 0x0600, 0x0300, 0x0100, 0x0080,   // (
0x2000, 0x1000, 0x1800, 0x0C00, 0x0C00, 0x0400, 0x0600, 0x0600, 0x0600, 0x0600,
0x0600, 0x0600, 0x0400, 0x0C00, 0x0C00, 0x1800, 0x1000, 0x2000,   // )
```

```
0x0000, 0x0C00, 0x2D00, 0x3F00, 0x1E00, 0x3300, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // *
0x0000, 0x0000, 0x0000, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0xFFC0, 0xFFC0, 0x0C00,
0x0C00, 0x0C00, 0x0C00, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // +
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0C00, 0x0C00, 0x0400, 0x0400, 0x0800,   // ,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x1E00,
0x1E00, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // -
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0C00, 0x0C00, 0x0000, 0x0000, 0x0000,   // .
0x0000, 0x0300, 0x0300, 0x0300, 0x0600, 0x0600, 0x0600, 0x0600, 0x0C00, 0x0C00,
0x0C00, 0x0C00, 0x1800, 0x1800, 0x1800, 0x0000, 0x0000, 0x0000,   // /
0x0000, 0x1E00, 0x3F00, 0x3300, 0x6180, 0x6180, 0x6180, 0x6D80, 0x6D80, 0x6180,
0x6180, 0x6180, 0x3300, 0x3F00, 0x1E00, 0x0000, 0x0000, 0x0000,   // 0
0x0000, 0x0600, 0x0E00, 0x1E00, 0x3600, 0x2600, 0x0600, 0x0600, 0x0600, 0x0600,
0x0600, 0x0600, 0x0600, 0x0600, 0x0600, 0x0000, 0x0000, 0x0000,   // 1
0x0000, 0x1E00, 0x3F00, 0x7380, 0x6180, 0x6180, 0x0180, 0x0300, 0x0600, 0x0C00,
0x1800, 0x3000, 0x6000, 0x7F80, 0x7F80, 0x0000, 0x0000, 0x0000,   // 2
0x0000, 0x1C00, 0x3E00, 0x6300, 0x6300, 0x0300, 0x0E00, 0x0E00, 0x0300, 0x0180,
0x0180, 0x6180, 0x7380, 0x3F00, 0x1E00, 0x0000, 0x0000, 0x0000,   // 3
0x0000, 0x0600, 0x0E00, 0x0E00, 0x1E00, 0x1E00, 0x1600, 0x3600, 0x3600, 0x6600,
0x7F80, 0x7F80, 0x0600, 0x0600, 0x0600, 0x0000, 0x0000, 0x0000,   // 4
0x0000, 0x7F00, 0x7F00, 0x6000, 0x6000, 0x6000, 0x6E00, 0x7F00, 0x6380, 0x0180,
0x0180, 0x6180, 0x7380, 0x3F00, 0x1E00, 0x0000, 0x0000, 0x0000,   // 5
0x0000, 0x1E00, 0x3F00, 0x3380, 0x6180, 0x6000, 0x6E00, 0x7F00, 0x7380, 0x6180,
0x6180, 0x6180, 0x3380, 0x3F00, 0x1E00, 0x0000, 0x0000, 0x0000,   // 6
0x0000, 0x7F80, 0x7F80, 0x0180, 0x0300, 0x0300, 0x0600, 0x0600, 0x0C00, 0x0C00,
0x0C00, 0x0800, 0x1800, 0x1800, 0x1800, 0x0000, 0x0000, 0x0000,   // 7
0x0000, 0x1E00, 0x3F00, 0x6380, 0x6180, 0x6180, 0x2100, 0x1E00, 0x3F00, 0x6180,
0x6180, 0x6180, 0x6180, 0x3F00, 0x1E00, 0x0000, 0x0000, 0x0000,   // 8
0x0000, 0x1E00, 0x3F00, 0x7300, 0x6180, 0x6180, 0x6180, 0x7380, 0x3F80, 0x1D80,
0x0180, 0x6180, 0x7300, 0x3F00, 0x1E00, 0x0000, 0x0000, 0x0000,   // 9
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0C00, 0x0C00, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0C00, 0x0C00, 0x0000, 0x0000, 0x0000,   // :
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0C00, 0x0C00, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0C00, 0x0C00, 0x0400, 0x0400, 0x0800,   // ;
0x0000, 0x0000, 0x0000, 0x0000, 0x0080, 0x0380, 0x0E00, 0x3800, 0x6000, 0x3800,
0x0E00, 0x0380, 0x0080, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // <
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x7F80, 0x7F80, 0x0000, 0x0000, 0x7F80,
0x7F80, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // =
0x0000, 0x0000, 0x0000, 0x0000, 0x4000, 0x7000, 0x1C00, 0x0700, 0x0180, 0x0700,
0x1C00, 0x7000, 0x4000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // >
0x0000, 0x1F00, 0x3F80, 0x71C0, 0x60C0, 0x00C0, 0x01C0, 0x0380, 0x0700, 0x0E00,
0x0C00, 0x0C00, 0x0000, 0x0C00, 0x0C00, 0x0000, 0x0000, 0x0000,   // ?
0x0000, 0x1E00, 0x3F00, 0x3180, 0x7180, 0x6380, 0x6F80, 0x6D80, 0x6D80, 0x6F80,
0x6780, 0x6000, 0x3200, 0x3E00, 0x1C00, 0x0000, 0x0000, 0x0000,   // @
0x0000, 0x0E00, 0x0E00, 0x1B00, 0x1B00, 0x1B00, 0x1B00, 0x3180, 0x3180, 0x3F80,
0x3F80, 0x3180, 0x60C0, 0x60C0, 0x60C0, 0x0000, 0x0000, 0x0000,   // A
0x0000, 0x7C00, 0x7E00, 0x6300, 0x6300, 0x6300, 0x6300, 0x7E00, 0x7E00, 0x6300,
0x6180, 0x6180, 0x6380, 0x7F00, 0x7E00, 0x0000, 0x0000, 0x0000,   // B
0x0000, 0x1E00, 0x3F00, 0x3180, 0x6180, 0x6000, 0x6000, 0x6000, 0x6000, 0x6000,
0x6000, 0x6180, 0x3180, 0x3F00, 0x1E00, 0x0000, 0x0000, 0x0000,   // C
0x0000, 0x7C00, 0x7F00, 0x6300, 0x6380, 0x6180, 0x6180, 0x6180, 0x6180, 0x6180,
0x6180, 0x6300, 0x6300, 0x7E00, 0x7C00, 0x0000, 0x0000, 0x0000,   // D
0x0000, 0x7F80, 0x7F80, 0x6000, 0x6000, 0x6000, 0x6000, 0x7F00, 0x7F00, 0x6000,
0x6000, 0x6000, 0x6000, 0x7F80, 0x7F80, 0x0000, 0x0000, 0x0000,   // E
0x0000, 0x7F80, 0x7F80, 0x6000, 0x6000, 0x6000, 0x6000, 0x7F00, 0x7F00, 0x6000,
0x6000, 0x6000, 0x6000, 0x6000, 0x6000, 0x0000, 0x0000, 0x0000,   // F
```

```
0x0000, 0x1E00, 0x3F00, 0x3180, 0x6180, 0x6000, 0x6000, 0x6000, 0x6380, 0x6380,
0x6180, 0x6180, 0x3180, 0x3F80, 0x1E00, 0x0000, 0x0000, 0x0000,   // G
0x0000, 0x6180, 0x6180, 0x6180, 0x6180, 0x6180, 0x6180, 0x7F80, 0x7F80, 0x6180,
0x6180, 0x6180, 0x6180, 0x6180, 0x6180, 0x0000, 0x0000, 0x0000,   // H
0x0000, 0x3F00, 0x3F00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00,
0x0C00, 0x0C00, 0x0C00, 0x3F00, 0x3F00, 0x0000, 0x0000, 0x0000,   // I
0x0000, 0x0180, 0x0180, 0x0180, 0x0180, 0x0180, 0x0180, 0x0180, 0x0180, 0x0180,
0x6180, 0x6180, 0x7380, 0x3F00, 0x1E00, 0x0000, 0x0000, 0x0000,   // J
0x0000, 0x60C0, 0x6180, 0x6300, 0x6600, 0x6600, 0x6C00, 0x7800, 0x7C00, 0x6600,
0x6600, 0x6300, 0x6180, 0x6180, 0x60C0, 0x0000, 0x0000, 0x0000,   // K
0x0000, 0x6000, 0x6000, 0x6000, 0x6000, 0x6000, 0x6000, 0x6000, 0x6000, 0x6000,
0x6000, 0x6000, 0x6000, 0x7F80, 0x7F80, 0x0000, 0x0000, 0x0000,   // L
0x0000, 0x71C0, 0x71C0, 0x7BC0, 0x7AC0, 0x6AC0, 0x6AC0, 0x6EC0, 0x64C0, 0x60C0,
0x60C0, 0x60C0, 0x60C0, 0x60C0, 0x60C0, 0x0000, 0x0000, 0x0000,   // M
0x0000, 0x7180, 0x7180, 0x7980, 0x7980, 0x7980, 0x6D80, 0x6D80, 0x6D80, 0x6580,
0x6780, 0x6780, 0x6780, 0x6380, 0x6380, 0x0000, 0x0000, 0x0000,   // N
0x0000, 0x1E00, 0x3F00, 0x3300, 0x6180, 0x6180, 0x6180, 0x6180, 0x6180, 0x6180,
0x6180, 0x6180, 0x3300, 0x3F00, 0x1E00, 0x0000, 0x0000, 0x0000,   // O
0x0000, 0x7E00, 0x7F00, 0x6380, 0x6180, 0x6180, 0x6180, 0x6380, 0x7F00, 0x7E00,
0x6000, 0x6000, 0x6000, 0x6000, 0x6000, 0x0000, 0x0000, 0x0000,   // P
0x0000, 0x1E00, 0x3F00, 0x3300, 0x6180, 0x6180, 0x6180, 0x6180, 0x6180, 0x6180,
0x6580, 0x6780, 0x3300, 0x3F80, 0x1E40, 0x0000, 0x0000, 0x0000,   // Q
0x0000, 0x7E00, 0x7F00, 0x6380, 0x6180, 0x6180, 0x6380, 0x7F00, 0x7E00, 0x6600,
0x6300, 0x6300, 0x6180, 0x6180, 0x60C0, 0x0000, 0x0000, 0x0000,   // R
0x0000, 0x0E00, 0x1F00, 0x3180, 0x3180, 0x3000, 0x3800, 0x1E00, 0x0700, 0x0380,
0x6180, 0x6180, 0x3180, 0x3F00, 0x1E00, 0x0000, 0x0000, 0x0000,   // S
0x0000, 0xFFC0, 0xFFC0, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00,
0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0000, 0x0000, 0x0000,   // T
0x0000, 0x6180, 0x6180, 0x6180, 0x6180, 0x6180, 0x6180, 0x6180, 0x6180, 0x6180,
0x6180, 0x6180, 0x7380, 0x3F00, 0x1E00, 0x0000, 0x0000, 0x0000,   // U
0x0000, 0x60C0, 0x60C0, 0x60C0, 0x3180, 0x3180, 0x3180, 0x1B00, 0x1B00, 0x1B00,
0x1B00, 0x0E00, 0x0E00, 0x0E00, 0x0400, 0x0000, 0x0000, 0x0000,   // V
0x0000, 0xC0C0, 0xC0C0, 0xC0C0, 0xC0C0, 0xC0C0, 0xCCC0, 0x4C80, 0x4C80, 0x5E80,
0x5280, 0x5280, 0x7380, 0x6180, 0x6180, 0x0000, 0x0000, 0x0000,   // W
0x0000, 0xC0C0, 0x6080, 0x6180, 0x3300, 0x3B00, 0x1E00, 0x0C00, 0x0C00, 0x1E00,
0x1F00, 0x3B00, 0x7180, 0x6180, 0xC0C0, 0x0000, 0x0000, 0x0000,   // X
0x0000, 0xC0C0, 0x6180, 0x6180, 0x3300, 0x3300, 0x1E00, 0x1E00, 0x0C00, 0x0C00,
0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0000, 0x0000, 0x0000,   // Y
0x0000, 0x3F80, 0x3F80, 0x0180, 0x0300, 0x0300, 0x0600, 0x0C00, 0x0C00, 0x1800,
0x1800, 0x3000, 0x6000, 0x7F80, 0x7F80, 0x0000, 0x0000, 0x0000,   // Z
0x0F00, 0x0F00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00,
0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0F00, 0x0F00,   // [
0x0000, 0x1800, 0x1800, 0x1800, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0600, 0x0600,
0x0600, 0x0600, 0x0300, 0x0300, 0x0300, 0x0000, 0x0000, 0x0000,   /* \ */
0x1E00, 0x1E00, 0x0600, 0x0600, 0x0600, 0x0600, 0x0600, 0x0600, 0x0600, 0x0600,
0x0600, 0x0600, 0x0600, 0x0600, 0x0600, 0x0600, 0x1E00, 0x1E00,   // ]
0x0000, 0x0C00, 0x0C00, 0x1E00, 0x1200, 0x3300, 0x3300, 0x6180, 0x6180, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // ^
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0xFFE0, 0x0000,   // _
0x0000, 0x3800, 0x1800, 0x0C00, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // `
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x1F00, 0x3F80, 0x6180, 0x0180, 0x1F80,
0x3F80, 0x6180, 0x6380, 0x7F80, 0x38C0, 0x0000, 0x0000, 0x0000,   // a
0x0000, 0x6000, 0x6000, 0x6000, 0x6000, 0x6E00, 0x7F00, 0x7380, 0x6180, 0x6180,
0x6180, 0x6180, 0x7380, 0x7F00, 0x6E00, 0x0000, 0x0000, 0x0000,   // b
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x1E00, 0x3F00, 0x7380, 0x6180, 0x6000,
0x6000, 0x6180, 0x7380, 0x3F00, 0x1E00, 0x0000, 0x0000, 0x0000,   // c
```

```
0x0000, 0x0180, 0x0180, 0x0180, 0x0180, 0x1D80, 0x3F80, 0x7380, 0x6180, 0x6180,
0x6180, 0x6180, 0x7380, 0x3F80, 0x1D80, 0x0000, 0x0000, 0x0000,   // d
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x1E00, 0x3F00, 0x7300, 0x6180, 0x7F80,
0x7F80, 0x6000, 0x7180, 0x3F00, 0x1E00, 0x0000, 0x0000, 0x0000,   // e
0x0000, 0x07C0, 0x0FC0, 0x0C00, 0x0C00, 0x7F80, 0x7F80, 0x0C00, 0x0C00, 0x0C00,
0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0000, 0x0000, 0x0000,   // f
0x0000, 0x0000, 0x0000, 0x0000, 0x1D80, 0x3F80, 0x7380, 0x6180, 0x6180, 0x6180,
0x6180, 0x7380, 0x3F80, 0x1D80, 0x0180, 0x6380, 0x7F00, 0x3E00,   // g
0x0000, 0x6000, 0x6000, 0x6000, 0x6000, 0x6F00, 0x7F80, 0x7180, 0x6180, 0x6180,
0x6180, 0x6180, 0x6180, 0x6180, 0x6180, 0x0000, 0x0000, 0x0000,   // h
0x0000, 0x0600, 0x0600, 0x0000, 0x0000, 0x3E00, 0x3E00, 0x0600, 0x0600, 0x0600,
0x0600, 0x0600, 0x0600, 0x0600, 0x0600, 0x0000, 0x0000, 0x0000,   // i
0x0600, 0x0600, 0x0000, 0x0000, 0x3E00, 0x3E00, 0x0600, 0x0600, 0x0600, 0x0600,
0x0600, 0x0600, 0x0600, 0x0600, 0x0600, 0x4600, 0x7E00, 0x3C00,   // j
0x0000, 0x6000, 0x6000, 0x6000, 0x6000, 0x6180, 0x6300, 0x6600, 0x6C00, 0x7C00,
0x7600, 0x6300, 0x6300, 0x6180, 0x60C0, 0x0000, 0x0000, 0x0000,   // k
0x0000, 0x3E00, 0x3E00, 0x0600, 0x0600, 0x0600, 0x0600, 0x0600, 0x0600, 0x0600,
0x0600, 0x0600, 0x0600, 0x0600, 0x0600, 0x0000, 0x0000, 0x0000,   // l
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0xDD80, 0xFFC0, 0xCEC0, 0xCCC0, 0xCCC0,
0xCCC0, 0xCCC0, 0xCCC0, 0xCCC0, 0xCCC0, 0x0000, 0x0000, 0x0000,   // m
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x6F00, 0x7F80, 0x7180, 0x6180, 0x6180,
0x6180, 0x6180, 0x6180, 0x6180, 0x6180, 0x0000, 0x0000, 0x0000,   // n
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x1E00, 0x3F00, 0x7380, 0x6180, 0x6180,
0x6180, 0x6180, 0x7380, 0x3F00, 0x1E00, 0x0000, 0x0000, 0x0000,   // o
0x0000, 0x0000, 0x0000, 0x0000, 0x6E00, 0x7F00, 0x7380, 0x6180, 0x6180, 0x6180,
0x6180, 0x7380, 0x7F00, 0x6E00, 0x6000, 0x6000, 0x6000, 0x6000,   // p
0x0000, 0x0000, 0x0000, 0x0000, 0x1D80, 0x3F80, 0x7380, 0x6180, 0x6180, 0x6180,
0x6180, 0x7380, 0x3F80, 0x1D80, 0x0180, 0x0180, 0x0180, 0x0180,   // q
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x6700, 0x3F80, 0x3900, 0x3000, 0x3000,
0x3000, 0x3000, 0x3000, 0x3000, 0x3000, 0x0000, 0x0000, 0x0000,   // r
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x1E00, 0x3F80, 0x6180, 0x6000, 0x7F00,
0x3F80, 0x0180, 0x6180, 0x7F00, 0x1E00, 0x0000, 0x0000, 0x0000,   // s
0x0000, 0x0000, 0x0800, 0x1800, 0x1800, 0x7F00, 0x7F00, 0x1800, 0x1800, 0x1800,
0x1800, 0x1800, 0x1800, 0x1F80, 0x0F80, 0x0000, 0x0000, 0x0000,   // t
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x6180, 0x6180, 0x6180, 0x6180, 0x6180,
0x6180, 0x6180, 0x6380, 0x7F80, 0x3D80, 0x0000, 0x0000, 0x0000,   // u
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x60C0, 0x3180, 0x3180, 0x3180, 0x1B00,
0x1B00, 0x1B00, 0x0E00, 0x0E00, 0x0600, 0x0000, 0x0000, 0x0000,   // v
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0xDD80, 0xDD80, 0xDD80, 0x5500, 0x5500,
0x5500, 0x7700, 0x7700, 0x2200, 0x2200, 0x0000, 0x0000, 0x0000,   // w
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x6180, 0x3300, 0x3300, 0x1E00, 0x0C00,
0x0C00, 0x1E00, 0x3300, 0x3300, 0x6180, 0x0000, 0x0000, 0x0000,   // x
0x0000, 0x0000, 0x0000, 0x0000, 0x6180, 0x6180, 0x3180, 0x3300, 0x3300, 0x1B00,
0x1B00, 0x1B00, 0x0E00, 0x0E00, 0x0E00, 0x1C00, 0x7C00, 0x7000,   // y
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x7FC0, 0x7FC0, 0x0180, 0x0300, 0x0600,
0x0C00, 0x1800, 0x3000, 0x7FC0, 0x7FC0, 0x0000, 0x0000, 0x0000,   // z
0x0380, 0x0780, 0x0600, 0x0600, 0x0600, 0x0600, 0x0600, 0x0E00, 0x1C00, 0x1C00,
0x0E00, 0x0600, 0x0600, 0x0600, 0x0600, 0x0600, 0x0780, 0x0380,   // {
0x0600, 0x0600, 0x0600, 0x0600, 0x0600, 0x0600, 0x0600, 0x0600, 0x0600, 0x0600,
0x0600, 0x0600, 0x0600, 0x0600, 0x0600, 0x0600, 0x0600, 0x0600,   // |
0x3800, 0x3C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0E00, 0x0700, 0x0700,
0x0E00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x0C00, 0x3C00, 0x3800,   // }
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x3880, 0x7F80, 0x4700,
0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000, 0x0000,   // ~
};

const uint16_t Font16x26 [] = {
```

```
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0
000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [ ]
0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x03C0,0x03C0,0x01C0,0x0
1C0,0x01C0,0x01C0,0x0000,0x0000,0x0000,0x03E0,0x03E0,0x03E0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [!]
0x1E3C,0x1E3C,0x1E3C,0x1E3C,0x1E3C,0x1E3C,0x1E3C,0x0000,0x0000,0x0000,0x0000,0x0
000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = ["]
0x01CE,0x03CE,0x03DE,0x039E,0x039C,0x079C,0x3FFF,0x7FFF,0x0738,0x0F38,0x0F78,0x0
F78,0x0E78,0xFFFF,0xFFFF,0x1EF0,0x1CF0,0x1CE0,0x3CE0,0x3DE0,0x39E0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [#]
0x03FC,0x0FFE,0x1FEE,0x1EE0,0x1EE0,0x1EE0,0x1EE0,0x1FE0,0x0FE0,0x07E0,0x03F0,0x0
1FC,0x01FE,0x01FE,0x01FE,0x01FE,0x01FE,0x01FE,0x3DFE,0x3FFC,0x0FF0,0x01E0,0x01E0
,0x0000,0x0000,0x0000, // Ascii = [$]
0x3E03,0xF707,0xE78F,0xE78E,0xE39E,0xE3BC,0xE7B8,0xE7F8,0xF7F0,0x3FE0,0x01C0,0x0
3FF,0x07FF,0x07F3,0x0FF3,0x1EF3,0x3CF3,0x38F3,0x78F3,0xF07F,0xE03F,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [%]
0x07E0,0x0FF8,0x0F78,0x1F78,0x1F78,0x1F78,0x0F78,0x0FF0,0x0FE0,0x1F80,0x7FC3,0xF
BC3,0xF3E7,0xF1F7,0xF0F7,0xF0FF,0xF07F,0xF83E,0x7C7F,0x3FFF,0x1FEF,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [&]
0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x03C0,0x01C0,0x0000,0x0000,0x0000,0x0000,0x0
000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = ['']
0x003F,0x007C,0x01F0,0x01E0,0x03C0,0x07C0,0x0780,0x0780,0x0F80,0x0F00,0x0F00,0x0
F00,0x0F00,0x0F00,0x0F00,0x0F80,0x0780,0x0780,0x07C0,0x03C0,0x01E0,0x01F0,0x007C
,0x003F,0x000F,0x0000, // Ascii = [(]
0x7E00,0x1F00,0x07C0,0x03C0,0x01E0,0x01F0,0x00F0,0x00F0,0x00F8,0x0078,0x0078,0x0
078,0x0078,0x0078,0x0078,0x00F8,0x00F0,0x00F0,0x01F0,0x01E0,0x03C0,0x07C0,0x1F00
,0x7E00,0x7800,0x0000, // Ascii = [)]
0x03E0,0x03C0,0x01C0,0x39CE,0x3FFF,0x3F7F,0x0320,0x0370,0x07F8,0x0F78,0x1F3C,0x0
638,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [*]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x01C0,0x01C0,0x01C0,0x01C0,0x01C0,0x0
1C0,0x01C0,0xFFFF,0xFFFF,0x01C0,0x01C0,0x01C0,0x01C0,0x01C0,0x01C0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [+]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0
000,0x0000,0x0000,0x0000,0x0000,0x0000,0x03E0,0x03E0,0x03E0,0x03E0,0x01E0,0x01E0
,0x01E0,0x01C0,0x0380, // Ascii = [,]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x3
FFE,0x3FFE,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [-]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0
000,0x0000,0x0000,0x0000,0x0000,0x0000,0x03E0,0x03E0,0x03E0,0x03E0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [.]
0x000F,0x000F,0x001E,0x001E,0x003C,0x003C,0x0078,0x0078,0x00F0,0x00F0,0x01E0,0x0
1E0,0x03C0,0x03C0,0x0780,0x0780,0x0F00,0x0F00,0x1E00,0x1E00,0x3C00,0x3C00,0x7800
,0x7800,0xF000,0x0000, // Ascii = [/]
0x07F0,0x0FF8,0x1F7C,0x3E3E,0x3C1E,0x7C1F,0x7C1F,0x780F,0x780F,0x780F,0x780F,0x7
80F,0x780F,0x780F,0x7C1F,0x7C1F,0x3C1E,0x3E3E,0x1F7C,0x0FF8,0x07F0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [0]
0x00F0,0x07F0,0x3FF0,0x3FF0,0x01F0,0x01F0,0x01F0,0x01F0,0x01F0,0x01F0,0x01F0,0x0
1F0,0x01F0,0x01F0,0x01F0,0x01F0,0x01F0,0x01F0,0x01F0,0x3FFF,0x3FFF,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [1]
0x0FE0,0x3FF8,0x3C7C,0x003C,0x003E,0x003E,0x003E,0x003C,0x003C,0x007C,0x00F8,0x0
1F0,0x03E0,0x07C0,0x0780,0x0F00,0x1E00,0x3E00,0x3C00,0x3FFE,0x3FFE,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [2]
```

```
0x0FF0,0x1FF8,0x1C7C,0x003E,0x003E,0x003E,0x003C,0x003C,0x00F8,0x0FF0,0x0FF8,0x0
07C,0x003E,0x001E,0x001E,0x001E,0x001E,0x003E,0x1C7C,0x1FF8,0x1FE0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [3]
0x0078,0x00F8,0x00F8,0x01F8,0x03F8,0x07F8,0x07F8,0x0F78,0x1E78,0x1E78,0x3C78,0x7
878,0x7878,0xFFFF,0xFFFF,0x0078,0x0078,0x0078,0x0078,0x0078,0x0078,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [4]
0x1FFC,0x1FFC,0x1FFC,0x1E00,0x1E00,0x1E00,0x1E00,0x1E00,0x1FE0,0x1FF8,0x00FC,0x0
07C,0x003E,0x003E,0x001E,0x003E,0x003E,0x003C,0x1C7C,0x1FF8,0x1FE0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [5]
0x01FC,0x07FE,0x0F8E,0x1F00,0x1E00,0x3E00,0x3C00,0x3C00,0x3DF8,0x3FFC,0x7F3E,0x7
E1F,0x3C0F,0x3C0F,0x3C0F,0x3C0F,0x3E0F,0x1E1F,0x1F3E,0x0FFC,0x03F0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [6]
0x3FFF,0x3FFF,0x3FFF,0x000F,0x001E,0x001E,0x003C,0x0038,0x0078,0x00F0,0x00F0,0x0
1E0,0x01E0,0x03C0,0x03C0,0x0780,0x0F80,0x0F80,0x0F00,0x1F00,0x1F00,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [7]
0x07F8,0x0FFC,0x1F3E,0x1E1E,0x3E1E,0x3E1E,0x1E1E,0x1F3C,0x0FF8,0x07F0,0x0FF8,0x1
EFC,0x3E3E,0x3C1F,0x7C1F,0x7C0F,0x7C0F,0x3C1F,0x3F3E,0x1FFC,0x07F0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [8]
0x07F0,0x0FF8,0x1E7C,0x3C3E,0x3C1E,0x7C1F,0x7C1F,0x7C1F,0x7C1F,0x3C1F,0x3E3F,0x1
FFF,0x07EF,0x001F,0x001E,0x001E,0x003E,0x003C,0x38F8,0x3FF0,0x1FE0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [9]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x03E0,0x03E0,0x03E0,0x03E0,0x0000,0x0
000,0x0000,0x0000,0x0000,0x0000,0x0000,0x03E0,0x03E0,0x03E0,0x03E0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [:]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x03E0,0x03E0,0x03E0,0x03E0,0x0000,0x0
000,0x0000,0x0000,0x0000,0x0000,0x0000,0x03E0,0x03E0,0x03E0,0x03E0,0x01E0,0x01E0
,0x01E0,0x03C0,0x0380, // Ascii = [;]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0003,0x000F,0x003F,0x00FC,0x03F0,0x0
FC0,0x3F00,0xFE00,0x3F00,0x0FC0,0x03F0,0x00FC,0x003F,0x000F,0x0003,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [<]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0xFFFF,0xF
FFF,0x0000,0x0000,0x0000,0xFFFF,0xFFFF,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [=]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0xE000,0xF800,0x7E00,0x1F80,0x07E0,0x0
1F8,0x007E,0x001F,0x007E,0x01F8,0x07E0,0x1F80,0x7E00,0xF800,0xE000,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [>]
0x1FF0,0x3FFC,0x383E,0x381F,0x381F,0x001E,0x001E,0x003C,0x0078,0x00F0,0x01E0,0x0
3C0,0x03C0,0x07C0,0x07C0,0x0000,0x0000,0x0000,0x07C0,0x07C0,0x07C0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [?]
0x03F8,0x0FFE,0x1F1E,0x3E0F,0x3C7F,0x78FF,0x79EF,0x73C7,0xF3C7,0xF38F,0xF38F,0xF
38F,0xF39F,0xF39F,0x73FF,0x7BFF,0x79F7,0x3C00,0x1F1C,0x0FFC,0x03F8,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [@]
0x0000,0x0000,0x0000,0x03E0,0x03E0,0x07F0,0x07F0,0x07F0,0x0F78,0x0F78,0x0E7C,0x1
E3C,0x1E3C,0x3C3E,0x3FFE,0x3FFF,0x781F,0x780F,0xF00F,0xF007,0xF007,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [A]
0x0000,0x0000,0x0000,0x3FF8,0x3FFC,0x3C3E,0x3C1E,0x3C1E,0x3C1E,0x3C3E,0x3C7C,0x3
FF0,0x3FF8,0x3C7E,0x3C1F,0x3C1F,0x3C0F,0x3C0F,0x3C1F,0x3FFE,0x3FF8,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [B]
0x0000,0x0000,0x0000,0x01FF,0x07FF,0x1F87,0x3E00,0x3C00,0x7C00,0x7800,0x7800,0x7
800,0x7800,0x7800,0x7C00,0x7C00,0x3E00,0x3F00,0x1F83,0x07FF,0x01FF,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [C]
0x0000,0x0000,0x0000,0x7FF0,0x7FFC,0x787E,0x781F,0x781F,0x780F,0x780F,0x780F,0x7
80F,0x780F,0x780F,0x780F,0x780F,0x781F,0x781E,0x787E,0x7FF8,0x7FE0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [D]
0x0000,0x0000,0x0000,0x3FFF,0x3FFF,0x3E00,0x3E00,0x3E00,0x3E00,0x3E00,0x3E00,0x3
FFE,0x3FFE,0x3E00,0x3E00,0x3E00,0x3E00,0x3E00,0x3E00,0x3FFF,0x3FFF,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [E]
```

```
0x0000,0x0000,0x0000,0x1FFF,0x1FFF,0x1E00,0x1E00,0x1E00,0x1E00,0x1E00,0x1E00,0x1
FFF,0x1FFF,0x1E00,0x1E00,0x1E00,0x1E00,0x1E00,0x1E00,0x1E00,0x1E00,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [F]
0x0000,0x0000,0x0000,0x03FE,0x0FFF,0x1F87,0x3E00,0x7C00,0x7C00,0x7800,0xF800,0xF
800,0xF87F,0xF87F,0x780F,0x7C0F,0x7C0F,0x3E0F,0x1F8F,0x0FFF,0x03FE,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [G]
0x0000,0x0000,0x0000,0x7C1F,0x7C1F,0x7C1F,0x7C1F,0x7C1F,0x7C1F,0x7C1F,0x7C1F,0x7
FFF,0x7FFF,0x7C1F,0x7C1F,0x7C1F,0x7C1F,0x7C1F,0x7C1F,0x7C1F,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [H]
0x0000,0x0000,0x0000,0x3FFF,0x3FFF,0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x0
3E0,0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x3FFF,0x3FFF,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [I]
0x0000,0x0000,0x0000,0x1FFC,0x1FFC,0x007C,0x007C,0x007C,0x007C,0x007C,0x007C,0x0
07C,0x007C,0x007C,0x007C,0x007C,0x0078,0x0078,0x38F8,0x3FF0,0x3FC0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [J]
0x0000,0x0000,0x0000,0x3C1F,0x3C1E,0x3C3C,0x3C78,0x3CF0,0x3DE0,0x3FE0,0x3FC0,0x3
F80,0x3FC0,0x3FE0,0x3DF0,0x3CF0,0x3C78,0x3C7C,0x3C3E,0x3C1F,0x3C0F,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [K]
0x0000,0x0000,0x0000,0x3E00,0x3E00,0x3E00,0x3E00,0x3E00,0x3E00,0x3E00,0x3E00,0x3
E00,0x3E00,0x3E00,0x3E00,0x3E00,0x3E00,0x3E00,0x3FFF,0x3FFF,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [L]
0x0000,0x0000,0x0000,0xF81F,0xFC1F,0xFC1F,0xFE3F,0xFE3F,0xFE3F,0xFF7F,0xFF77,0xF
F77,0xF7F7,0xF7E7,0xF3E7,0xF3E7,0xF3C7,0xF007,0xF007,0xF007,0xF007,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [M]
0x0000,0x0000,0x0000,0x7C0F,0x7C0F,0x7E0F,0x7F0F,0x7F0F,0x7F8F,0x7F8F,0x7FCF,0x7
BEF,0x79EF,0x79FF,0x78FF,0x78FF,0x787F,0x783F,0x783F,0x781F,0x781F,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [N]
0x0000,0x0000,0x0000,0x07F0,0x1FFC,0x3E3E,0x7C1F,0x780F,0x780F,0xF80F,0xF80F,0xF
80F,0xF80F,0xF80F,0xF80F,0x780F,0x780F,0x7C1F,0x3E3E,0x1FFC,0x07F0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [O]
0x0000,0x0000,0x0000,0x3FFC,0x3FFF,0x3E1F,0x3E0F,0x3E0F,0x3E0F,0x3E0F,0x3E1F,0x3
E3F,0x3FFC,0x3FF0,0x3E00,0x3E00,0x3E00,0x3E00,0x3E00,0x3E00,0x3E00,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [P]
0x0000,0x0000,0x0000,0x07F0,0x1FFC,0x3E3E,0x7C1F,0x780F,0x780F,0xF80F,0xF80F,0xF
80F,0xF80F,0xF80F,0xF80F,0x780F,0x780F,0x7C1F,0x3E3E,0x1FFC,0x07F8,0x007C,0x003F
,0x000F,0x0003,0x0000, // Ascii = [Q]
0x0000,0x0000,0x0000,0x3FF0,0x3FFC,0x3C7E,0x3C3E,0x3C1E,0x3C1E,0x3C3E,0x3C3C,0x3
CFC,0x3FF0,0x3FE0,0x3DF0,0x3CF8,0x3C7C,0x3C3E,0x3C1E,0x3C1F,0x3C0F,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [R]
0x0000,0x0000,0x0000,0x07FC,0x1FFE,0x3E0E,0x3C00,0x3C00,0x3C00,0x3E00,0x1FC0,0x0
FF8,0x03FE,0x007F,0x001F,0x000F,0x000F,0x201F,0x3C3E,0x3FFC,0x1FF0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [S]
0x0000,0x0000,0x0000,0xFFFF,0xFFFF,0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x0
3E0,0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [T]
0x0000,0x0000,0x0000,0x7C0F,0x7C0F,0x7C0F,0x7C0F,0x7C0F,0x7C0F,0x7C0F,0x7C0F,0x7
C0F,0x7C0F,0x7C0F,0x7C0F,0x7C0F,0x3C1E,0x3C1E,0x3E3E,0x1FFC,0x07F0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [U]
0x0000,0x0000,0x0000,0xF007,0xF007,0xF807,0x780F,0x7C0F,0x3C1E,0x3C1E,0x3E1E,0x1
E3C,0x1F3C,0x1F78,0x0F78,0x0FF8,0x07F0,0x07F0,0x07F0,0x03E0,0x03E0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [V]
0x0000,0x0000,0x0000,0xE003,0xF003,0xF003,0xF007,0xF3E7,0xF3E7,0xF3E7,0x73E7,0x7
BF7,0x7FF7,0x7FFF,0x7F7F,0x7F7F,0x7F7E,0x3F7E,0x3E3E,0x3E3E,0x3E3E,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [W]
0x0000,0x0000,0x0000,0xF807,0x7C0F,0x3E1E,0x3E3E,0x1F3C,0x0FF8,0x07F0,0x07E0,0x0
3E0,0x03E0,0x07F0,0x0FF8,0x0F7C,0x1E7C,0x3C3E,0x781F,0x780F,0xF00F,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [X]
```

```
0x0000,0x0000,0x0000,0xF807,0x7807,0x7C0F,0x3C1E,0x3E1E,0x1F3C,0x0F78,0x0FF8,0x0
7F0,0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x03E0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [Y]
0x0000,0x0000,0x0000,0x7FFF,0x7FFF,0x000F,0x001F,0x003E,0x007C,0x00F8,0x00F0,0x0
1E0,0x03E0,0x07C0,0x0F80,0x0F00,0x1E00,0x3E00,0x7C00,0x7FFF,0x7FFF,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [Z]
0x07FF,0x0780,0x0780,0x0780,0x0780,0x0780,0x0780,0x0780,0x0780,0x0780,0x0780,0x0
780,0x0780,0x0780,0x0780,0x0780,0x0780,0x0780,0x0780,0x0780,0x0780,0x0780,0x0780
,0x07FF,0x07FF,0x0000, // Ascii = [[]
0x7800,0x7800,0x3C00,0x3C00,0x1E00,0x1E00,0x0F00,0x0F00,0x0780,0x0780,0x03C0,0x0
3C0,0x01E0,0x01E0,0x00F0,0x00F0,0x0078,0x0078,0x003C,0x003C,0x001E,0x001E,0x000F
,0x000F,0x0007,0x0000, // Ascii = [\]
0x7FF0,0x00F0,0x00F0,0x00F0,0x00F0,0x00F0,0x00F0,0x00F0,0x00F0,0x00F0,0x00F0,0x0
0F0,0x00F0,0x00F0,0x00F0,0x00F0,0x00F0,0x00F0,0x00F0,0x00F0,0x00F0,0x00F0,0x00F0
,0x7FF0,0x7FF0,0x0000, // Ascii = []]
0x00C0,0x01C0,0x01C0,0x03E0,0x03E0,0x07F0,0x07F0,0x0778,0x0F78,0x0F38,0x1E3C,0x1
E3C,0x3C1E,0x3C1E,0x380F,0x780F,0x7807,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [^]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0
000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0xFFFF,0xFFFF
,0x0000,0x0000,0x0000, // Ascii = [_]
0x00F0,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0
000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [`]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0FF8,0x3FFC,0x3C7C,0x003E,0x003E,0x0
03E,0x07FE,0x1FFE,0x3E3E,0x7C3E,0x783E,0x7C3E,0x7C7E,0x3FFF,0x1FCF,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [a]
0x3C00,0x3C00,0x3C00,0x3C00,0x3C00,0x3C00,0x3DF8,0x3FFE,0x3F3E,0x3E1F,0x3C0F,0x3
C0F,0x3C0F,0x3C0F,0x3C0F,0x3C0F,0x3C1F,0x3C1E,0x3F3E,0x3FFC,0x3BF0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [b]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x03FE,0x0FFF,0x1F87,0x3E00,0x3E00,0x3
C00,0x7C00,0x7C00,0x7C00,0x3C00,0x3E00,0x3E00,0x1F87,0x0FFF,0x03FE,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [c]
0x001F,0x001F,0x001F,0x001F,0x001F,0x001F,0x07FF,0x1FFF,0x3E3F,0x3C1F,0x7C1F,0x7
C1F,0x7C1F,0x781F,0x781F,0x7C1F,0x7C1F,0x3C3F,0x3E7F,0x1FFF,0x0FDF,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [d]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x03F8,0x0FFC,0x1F3E,0x3E1E,0x3C1F,0x7
C1F,0x7FFF,0x7FFF,0x7C00,0x7C00,0x3C00,0x3E00,0x1F07,0x0FFF,0x03FE,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [e]
0x01FF,0x03E1,0x03C0,0x07C0,0x07C0,0x07C0,0x7FFF,0x7FFF,0x07C0,0x07C0,0x07C0,0x0
7C0,0x07C0,0x07C0,0x07C0,0x07C0,0x07C0,0x07C0,0x07C0,0x07C0,0x07C0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [f]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x07EF,0x1FFF,0x3E7F,0x3C1F,0x7C1F,0x7
C1F,0x781F,0x781F,0x781F,0x7C1F,0x7C1F,0x3C3F,0x3E7F,0x1FFF,0x0FDF,0x001E,0x001E
,0x001E,0x387C,0x3FF8, // Ascii = [g]
0x3C00,0x3C00,0x3C00,0x3C00,0x3C00,0x3C00,0x3DFC,0x3FFE,0x3F9E,0x3F1F,0x3E1F,0x3
C1F,0x3C1F,0x3C1F,0x3C1F,0x3C1F,0x3C1F,0x3C1F,0x3C1F,0x3C1F,0x3C1F,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [h]
0x01F0,0x01F0,0x0000,0x0000,0x0000,0x0000,0x7FE0,0x7FE0,0x01E0,0x01E0,0x01E0,0x0
1E0,0x01E0,0x01E0,0x01E0,0x01E0,0x01E0,0x01E0,0x01E0,0x01E0,0x01E0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [i]
0x00F8,0x00F8,0x0000,0x0000,0x0000,0x0000,0x3FF8,0x3FF8,0x00F8,0x00F8,0x00F8,0x0
0F8,0x00F8,0x00F8,0x00F8,0x00F8,0x00F8,0x00F8,0x00F8,0x00F8,0x00F8,0x00F8,0x00F8
,0x00F0,0x71F0,0x7FE0, // Ascii = [j]
0x3C00,0x3C00,0x3C00,0x3C00,0x3C00,0x3C00,0x3C1F,0x3C3E,0x3C7C,0x3CF8,0x3DF0,0x3
DE0,0x3FC0,0x3FC0,0x3FE0,0x3DF0,0x3CF8,0x3C7C,0x3C3E,0x3C1F,0x3C1F,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [k]
```

```
0x7FF0,0x01F0,0x01F0,0x01F0,0x01F0,0x01F0,0x01F0,0x01F0,0x01F0,0x01F0,0x01F0,0x0
1F0,0x01F0,0x01F0,0x01F0,0x01F0,0x01F0,0x01F0,0x01F0,0x01F0,0x01F0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [l]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0xF79E,0xFFFF,0xFFFF,0xFFFF,0xFBE7,0xF
9E7,0xF1C7,0xF1C7,0xF1C7,0xF1C7,0xF1C7,0xF1C7,0xF1C7,0xF1C7,0xF1C7,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [m]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x3DFC,0x3FFE,0x3F9E,0x3F1F,0x3E1F,0x3
C1F,0x3C1F,0x3C1F,0x3C1F,0x3C1F,0x3C1F,0x3C1F,0x3C1F,0x3C1F,0x3C1F,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [n]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x07F0,0x1FFC,0x3E3E,0x3C1F,0x7C1F,0x7
80F,0x780F,0x780F,0x780F,0x780F,0x7C1F,0x3C1F,0x3E3E,0x1FFC,0x07F0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [o]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x3DF8,0x3FFE,0x3F3E,0x3E1F,0x3C0F,0x3
C0F,0x3C0F,0x3C0F,0x3C0F,0x3C0F,0x3C1F,0x3E1E,0x3F3E,0x3FFC,0x3FF8,0x3C00,0x3C00
,0x3C00,0x3C00,0x3C00, // Ascii = [p]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x07EE,0x1FFE,0x3E7E,0x3C1E,0x7C1E,0x7
81E,0x781E,0x781E,0x781E,0x781E,0x7C1E,0x7C3E,0x3E7E,0x1FFE,0x0FDE,0x001E,0x001E
,0x001E,0x001E,0x001E, // Ascii = [q]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x1F7F,0x1FFF,0x1FE7,0x1FC7,0x1F87,0x1
F00,0x1F00,0x1F00,0x1F00,0x1F00,0x1F00,0x1F00,0x1F00,0x1F00,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [r]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x07FC,0x1FFE,0x1E0E,0x3E00,0x3E00,0x3
F00,0x1FE0,0x07FC,0x00FE,0x003E,0x001E,0x001E,0x3C3E,0x3FFC,0x1FF0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [s]
0x0000,0x0000,0x0000,0x0780,0x0780,0x0780,0x7FFF,0x7FFF,0x0780,0x0780,0x0780,0x0
780,0x0780,0x0780,0x0780,0x0780,0x0780,0x07C0,0x03FF,0x01FF,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [t]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x3C1E,0x3C1E,0x3C1E,0x3C1E,0x3C1E,0x3
C1E,0x3C1E,0x3C1E,0x3C1E,0x3C1E,0x3C3E,0x3C7E,0x3EFE,0x1FFE,0x0FDE,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [u]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0xF007,0x780F,0x780F,0x3C1E,0x3C1E,0x3
E1E,0x1E3C,0x1E3C,0x0F78,0x0F78,0x0FF0,0x07F0,0x07F0,0x03E0,0x03E0,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [v]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0xF003,0xF1E3,0xF3E3,0xF3E7,0xF3F7,0xF
3F7,0x7FF7,0x7F77,0x7F7F,0x7F7F,0x7F7F,0x3E3E,0x3E3E,0x3E3E,0x3E3E,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [w]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x7C0F,0x3E1E,0x3E3C,0x1F3C,0x0FF8,0x0
7F0,0x07F0,0x03E0,0x07F0,0x07F8,0x0FF8,0x1E7C,0x3E3E,0x3C1F,0x781F,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [x]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0xF807,0x780F,0x7C0F,0x3C1E,0x3C1E,0x1
E3C,0x1E3C,0x1F3C,0x0F78,0x0FF8,0x07F0,0x07F0,0x03E0,0x03E0,0x03C0,0x03C0,0x03C0
,0x0780,0x0F80,0x7F00, // Ascii = [y]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x3FFF,0x3FFF,0x001F,0x003E,0x007C,0x0
0F8,0x01F0,0x03E0,0x07C0,0x0F80,0x1F00,0x1E00,0x3C00,0x7FFF,0x7FFF,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [z]
0x01FE,0x03E0,0x03C0,0x03C0,0x03C0,0x03C0,0x01E0,0x01E0,0x01E0,0x01C0,0x03C0,0x3
F80,0x3F80,0x03C0,0x01C0,0x01E0,0x01E0,0x01E0,0x03C0,0x03C0,0x03C0,0x03C0,0x03E0
,0x01FE,0x007E,0x0000, // Ascii = [{]
0x01C0,0x01C0,0x01C0,0x01C0,0x01C0,0x01C0,0x01C0,0x01C0,0x01C0,0x01C0,0x01C0,0x0
1C0,0x01C0,0x01C0,0x01C0,0x01C0,0x01C0,0x01C0,0x01C0,0x01C0,0x01C0,0x01C0,0x01C0
,0x01C0,0x01C0,0x0000, // Ascii = [|]
0x3FC0,0x03E0,0x01E0,0x01E0,0x01E0,0x01E0,0x01C0,0x03C0,0x03C0,0x01C0,0x01E0,0x0
0FE,0x00FE,0x01E0,0x01C0,0x03C0,0x03C0,0x01C0,0x01E0,0x01E0,0x01E0,0x01E0,0x03E0
,0x3FC0,0x3F00,0x0000, // Ascii = [}]
0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x3
F07,0x7FC7,0x73E7,0xF1FF,0xF07E,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000,0x0000
,0x0000,0x0000,0x0000, // Ascii = [~]
};
```

```
FontDef_t Font_7x10 = {
	7,
	10,
	Font7x10
};

FontDef_t Font_11x18 = {
	11,
	18,
	Font11x18
};

FontDef_t Font_16x26 = {
	16,
	26,
	Font16x26
};

char* FONTS_GetStringSize(char* str, FONTS_SIZE_t* SizeStruct, FontDef_t* Font)
{
	/* Fill settings */
	SizeStruct->Height = Font->FontHeight;
	SizeStruct->Length = Font->FontWidth * strlen(str);

	/* Return pointer */
	return str;
}
```

Fonts.h

```
/**
 * Reference and token of thanks to code written by Alexander Lutsai, Tilen Majerle
 * https://controllerstech.com/oled-display-using-i2c-stm32/
 *
 *
*******************************************************************************
	* PROJECT: SMART FAN CONTROL SYSTEM
	* AUTHOR:  ISHA SHARMA
	* COURSE:  EMBEDDED SYSTEMS DESIGN SP'23
	* TOOLS:   STMCUBEIDE
	* DATE:    06-05-2023
	* FILE:    fonts.h
	* BRIEF:   This is header file which is used for fonts library to be used on the OLED
	*
	*
*******************************************************************************
 */
#ifndef FONTS_H
#define FONTS_H 120

/* C++ detection */
#ifdef __cplusplus
extern C {
#endif
```

```c
/**
 *
 * Default fonts library. It is used in all LCD based libraries.
 *
 * \par Supported fonts
 *
 * Currently, these fonts are supported:
 *  - 7 x 10 pixels
 *  - 11 x 18 pixels
 *  - 16 x 26 pixels
 */
#include "stm32f4xx_hal.h"
#include "string.h"

/**
 * @defgroup LIB_Typedefs
 * @brief    Library Typedefs
 * @{
 */

/**
 * @brief  Font structure used on my LCD libraries
 */
typedef struct {
	uint8_t FontWidth;    /*!< Font width in pixels */
	uint8_t FontHeight;   /*!< Font height in pixels */
	const uint16_t *data; /*!< Pointer to data font data array */
} FontDef_t;

/**
 * @brief  String length and height
 */
typedef struct {
	uint16_t Length;      /*!< String length in units of pixels */
	uint16_t Height;      /*!< String height in units of pixels */
} FONTS_SIZE_t;

/**
 * @}
 */

/**
 * @defgroup FONTS_FontVariables
 * @brief    Library font variables
 * @{
 */

/**
 * @brief  7 x 10 pixels font size structure
 */
extern FontDef_t Font_7x10;

/**
 * @brief  11 x 18 pixels font size structure
 */
extern FontDef_t Font_11x18;

/**
```

```
 * @brief  16 x 26 pixels font size structure
 */
extern FontDef_t Font_16x26;

/**
 * @}
 */

/**
 * @defgroup FONTS_Functions
 * @brief    Library functions
 * @{
 */

/**
 * @brief  Calculates string length and height in units of pixels depending on string and font used
 * @param  *str: String to be checked for length and height
 * @param  *SizeStruct: Pointer to empty @ref FONTS_SIZE_t structure where informations will be saved
 * @param  *Font: Pointer to @ref FontDef_t font used for calculations
 * @retval Pointer to string used for length and height
 */
char* FONTS_GetStringSize(char* str, FONTS_SIZE_t* SizeStruct, FontDef_t* Font);

/**
 * @}
 */

/**
 * @}
 */

/**
 * @}
 */

/* C++ detection */
#ifdef __cplusplus
}
#endif


#endif
```

SSD1306.c

```
/**
 * Reference and token of thanks to code written by Alexander Lutsai, Tilen Majerle
 * https://controllerstech.com/oled-display-using-i2c-stm32/
 *
*******************************************************************************
      * PROJECT: SMART FAN CONTROL SYSTEM
      * AUTHOR:  ISHA SHARMA
      * COURSE:  EMBEDDED SYSTEMS DESIGN SP'23
      * TOOLS:   STMCUBEIDE
```

```
       * DATE:    06-05-2023
       * FILE:    ssd1306.c
       * BRIEF:   This file is used for functions related to the SSD1306 OLED
       *
       *
*******************************************************************************
 */

#include "ssd1306.h"

extern I2C_HandleTypeDef hi2c1; //i2c1 for ssd1306

/* Write command */
#define SSD1306_WRITECOMMAND(command)      ssd1306_I2C_Write(SSD1306_I2C_ADDR,
0x00, (command))

/* Absolute value */
#define ABS(x)   ((x) > 0 ? (x) : -(x))

/* SSD1306 data buffer */
static uint8_t SSD1306_Buffer[SSD1306_WIDTH * SSD1306_HEIGHT / 8];

/* Private SSD1306 structure */
typedef struct {
       uint16_t CurrentX;
       uint16_t CurrentY;
       uint8_t Inverted;
       uint8_t Initialized;
} SSD1306_t;

/* Private variable */
static SSD1306_t SSD1306;
#define SSD1306_RIGHT_HORIZONTAL_SCROLL              0x26 // scrool right
#define SSD1306_LEFT_HORIZONTAL_SCROLL               0x27 // scroll left
#define SSD1306_DEACTIVATE_SCROLL                    0x2E // Stop scroll
#define SSD1306_ACTIVATE_SCROLL                      0x2F // Start scroll

/*
name: SSD1306_ScrollRight
description: function to scroll right for fixed rows
parameters: start row, end row
returns: none
 */
void SSD1306_ScrollRight(uint8_t start_row, uint8_t end_row)
{
  SSD1306_WRITECOMMAND (SSD1306_RIGHT_HORIZONTAL_SCROLL);  // send 0x26
  SSD1306_WRITECOMMAND (0x00);  // send dummy
  SSD1306_WRITECOMMAND(start_row);  // start page address
  SSD1306_WRITECOMMAND(0X00);  // time interval 5 frames
  SSD1306_WRITECOMMAND(end_row);  // end page address
  SSD1306_WRITECOMMAND(0X00);
  SSD1306_WRITECOMMAND(0XFF);
  SSD1306_WRITECOMMAND (SSD1306_ACTIVATE_SCROLL); // start scroll
}

/*
name: SSD1306_ScrollLeft
description: function to scroll left for fixed rows
parameters: start row, end row
```

```
returns: none
 */
void SSD1306_ScrollLeft(uint8_t start_row, uint8_t end_row)
{
  SSD1306_WRITECOMMAND (SSD1306_LEFT_HORIZONTAL_SCROLL);  // send 0x26
  SSD1306_WRITECOMMAND (0x00);  // send dummy
  SSD1306_WRITECOMMAND(start_row);  // start page address
  SSD1306_WRITECOMMAND(0X00);  // time interval 5 frames
  SSD1306_WRITECOMMAND(end_row);  // end page address
  SSD1306_WRITECOMMAND(0X00);
  SSD1306_WRITECOMMAND(0XFF);
  SSD1306_WRITECOMMAND (SSD1306_ACTIVATE_SCROLL); // start scroll
}

/*
name: SSD1306_Stopscroll
description: function to stop scrolling on the screen
parameters: none
returns: none
 */
void SSD1306_Stopscroll(void)
{
	SSD1306_WRITECOMMAND(SSD1306_DEACTIVATE_SCROLL);
}

/*
name: SSD1306_Init
description: function to initialize SSD1306 LCD
parameters: none
returns: 0 if I2c port not detected for oled, 1 if init done
 */
uint8_t SSD1306_Init(void) {

	/* delay */
	uint32_t p = 250000;
	while(p>0)
		p--;

	/* Check if LCD connected to I2C */
	if (HAL_I2C_IsDeviceReady(&hi2c1, SSD1306_I2C_ADDR, 1, 20000) != HAL_OK)
{
		/* Return false if lcd not found at the port */
		return 0;
	}

	/*delay */
	uint32_t g = 2500;
	while(g>0)
		g--;

	/* Init LCD based on commands from the datasheet */
	SSD1306_WRITECOMMAND(0xAE); //display off
	SSD1306_WRITECOMMAND(0x20); //Set Memory Addressing Mode
	SSD1306_WRITECOMMAND(0x10); //00,Horizontal Addressing Mode;01,Vertical
Addressing Mode;10,Page Addressing Mode (RESET);11,Invalid
	SSD1306_WRITECOMMAND(0xB0); //Set Page Start Address for Page Addressing
Mode,0-7
	SSD1306_WRITECOMMAND(0xC8); //Set COM Output Scan Direction
	SSD1306_WRITECOMMAND(0x00); //---set low column address
```

```
	SSD1306_WRITECOMMAND(0x10); //---set high column address
	SSD1306_WRITECOMMAND(0x40); //--set start line address
	SSD1306_WRITECOMMAND(0x81); //--set contrast control register
	SSD1306_WRITECOMMAND(0xFF);
	SSD1306_WRITECOMMAND(0xA1); //--set segment re-map 0 to 127
	SSD1306_WRITECOMMAND(0xA6); //--set normal display
	SSD1306_WRITECOMMAND(0xA8); //--set multiplex ratio(1 to 64)
	SSD1306_WRITECOMMAND(0x3F); //
	SSD1306_WRITECOMMAND(0xA4); //0xa4,Output follows RAM content;0xa5,Output
ignores RAM content
	SSD1306_WRITECOMMAND(0xD3); //-set display offset
	SSD1306_WRITECOMMAND(0x00); //-not offset
	SSD1306_WRITECOMMAND(0xD5); //--set display clock divide ratio/oscillator
frequency
	SSD1306_WRITECOMMAND(0xF0); //--set divide ratio
	SSD1306_WRITECOMMAND(0xD9); //--set pre-charge period
	SSD1306_WRITECOMMAND(0x22); //
	SSD1306_WRITECOMMAND(0xDA); //--set com pins hardware configuration
	SSD1306_WRITECOMMAND(0x12);
	SSD1306_WRITECOMMAND(0xDB); //--set vcomh
	SSD1306_WRITECOMMAND(0x20); //0x20,0.77xVcc
	SSD1306_WRITECOMMAND(0x8D); //--set DC-DC enable
	SSD1306_WRITECOMMAND(0x14); //
	SSD1306_WRITECOMMAND(0xAF); //--turn on SSD1306 panel


	SSD1306_WRITECOMMAND(SSD1306_DEACTIVATE_SCROLL); //stop scroll

	/* Clear screen */
	SSD1306_Fill(SSD1306_COLOR_BLACK);

	/* Update screen */
	SSD1306_UpdateScreen();

	/* Set default values */
	SSD1306.CurrentX = 0;
	SSD1306.CurrentY = 0;

	/* Initialized OK */
	SSD1306.Initialized = 1;

	/* Return OK */
	return 1;
}

/*
name: SSD1306_UpdateScreen
description: function to update the SS1306 buffer from internal RAM to OLED
parameters: none
returns: none
 */
void SSD1306_UpdateScreen(void) {
	uint8_t m;

	for (m = 0; m < 8; m++) {
		SSD1306_WRITECOMMAND(0xB0 + m);
		SSD1306_WRITECOMMAND(0x00);
		SSD1306_WRITECOMMAND(0x10);
```

```
			/* Write multi data */
			ssd1306_I2C_WriteMulti(SSD1306_I2C_ADDR, 0x40,
&SSD1306_Buffer[SSD1306_WIDTH * m], SSD1306_WIDTH);
		}
}

/*
name: SSD1306_Fill
description: function to fill the screen with a colour
parameters: color from the enum to be filled (black/white)
returns: none
 */
void SSD1306_Fill(SSD1306_COLOR_t color) {
		/* Set memory */
		memset(SSD1306_Buffer, (color == SSD1306_COLOR_BLACK) ? 0x00 : 0xFF,
sizeof(SSD1306_Buffer));
}

/*
name: SSD1306_DrawPixel
description: function to draw a pixel at a location on the OLED
parameters: x location, y location and the color to fill the pixel with
returns: none
 */
void SSD1306_DrawPixel(uint16_t x, uint16_t y, SSD1306_COLOR_t color) {
		if (
			x >= SSD1306_WIDTH ||
			y >= SSD1306_HEIGHT
		) {
			/* Error */
			return;
		}

		/* Check if pixels are inverted */
		if (SSD1306.Inverted) {
			color = (SSD1306_COLOR_t)!color;
		}

		/* Set color */
		if (color == SSD1306_COLOR_WHITE) {
			SSD1306_Buffer[x + (y / 8) * SSD1306_WIDTH] |= 1 << (y % 8);
		} else {
			SSD1306_Buffer[x + (y / 8) * SSD1306_WIDTH] &= ~(1 << (y % 8));
		}
}

/*
name: SSD1306_GotoXY
description: function to go to a particular cordinate on the DDRAM
parameters: x location, y location
returns: none
 */
void SSD1306_GotoXY(uint16_t x, uint16_t y) {
		/* Set write pointers */
		SSD1306.CurrentX = x;
		SSD1306.CurrentY = y;
}

/*
```

```
name: SSD1306_Putc
description: function to put a character on the internal RAM
parameters: character to be printed, font to be used, color of the character
returns: character that was written
 */
char SSD1306_Putc(char ch, FontDef_t* Font, SSD1306_COLOR_t color) {
	uint32_t i, b, j;

	/* Check available space in LCD */
	if (
		SSD1306_WIDTH <= (SSD1306.CurrentX + Font->FontWidth) ||
		SSD1306_HEIGHT <= (SSD1306.CurrentY + Font->FontHeight)
	) {
		/* Error */
		return 0;
	}

	/* Go through font */
	for (i = 0; i < Font->FontHeight; i++) {
		b = Font->data[(ch - 32) * Font->FontHeight + i];
		for (j = 0; j < Font->FontWidth; j++) {
			if ((b << j) & 0x8000) {
				SSD1306_DrawPixel(SSD1306.CurrentX + j,
(SSD1306.CurrentY + i), (SSD1306_COLOR_t) color);
			} else {
				SSD1306_DrawPixel(SSD1306.CurrentX + j,
(SSD1306.CurrentY + i), (SSD1306_COLOR_t)!color);
			}
		}
	}

	/* Increase pointer */
	SSD1306.CurrentX += Font->FontWidth;

	/* Return character written */
	return ch;
}

/*
name: SSD1306_Puts
description: function to put a string on the internal RAM
parameters: string to be printed, font to be used, color of the string
characters
returns: 0 for success, character for which function failed
 */
char SSD1306_Puts(char* str, FontDef_t* Font, SSD1306_COLOR_t color) {
	/* Write characters */
	while (*str) {
		/* Write character by character */
		if (SSD1306_Putc(*str, Font, color) != *str) {
			/* Return error */
			return *str;
		}

		/* Increase string pointer */
		str++;
	}

	/* Everything OK, zero should be returned */
```

```
	return *str;
}


/*
name: SSD1306_Clear
description: function to clear the LCD
parameters: none
returns: none
 */
void SSD1306_Clear (void)
{
	SSD1306_Fill (0);
    SSD1306_UpdateScreen();
}

/*
name: SSD1306_ON
description: function to turn the OLED on
parameters: none
returns: none
 */
void SSD1306_ON(void) {
	SSD1306_WRITECOMMAND(0x8D);
	SSD1306_WRITECOMMAND(0x14);
	SSD1306_WRITECOMMAND(0xAF);
}

/*
name: SSD1306_OFF
description: function to turn the OLED off
parameters: none
returns: none
 */
void SSD1306_OFF(void) {
	SSD1306_WRITECOMMAND(0x8D);
	SSD1306_WRITECOMMAND(0x10);
	SSD1306_WRITECOMMAND(0xAE);
}

/*
name: ssd1306_I2C_WriteMulti
description: function to write multiple bytes to the slave address
parameters: slave address, register to write to, data to write, count of how
many bytes to be written
returns: none
 */
void ssd1306_I2C_WriteMulti(uint8_t address, uint8_t reg, uint8_t* data,
uint16_t count) {
uint8_t dt[256];
dt[0] = reg;
uint8_t i;
for(i = 0; i < count; i++)
dt[i+1] = data[i];
HAL_I2C_Master_Transmit(&hi2c1, address, dt, count+1, 10);
}

/*
name: ssd1306_I2C_Write
```

```
description: function to write single byte to the slave address
parameters: slave address, register to write to, data to write
returns: none
 */
void ssd1306_I2C_Write(uint8_t address, uint8_t reg, uint8_t data) {
	uint8_t dt[2];
	dt[0] = reg;
	dt[1] = data;
	HAL_I2C_Master_Transmit(&hi2c1, address, dt, 2, 10);
}
```

SSD1306.h

```
/**
 * Reference and token of thanks to code written by Alexander Lutsai, Tilen Majerle
 * https://controllerstech.com/oled-display-using-i2c-stm32/
 *
******************************************************************************
	* PROJECT: SMART FAN CONTROL SYSTEM
	* AUTHOR:  ISHA SHARMA
	* COURSE:  EMBEDDED SYSTEMS DESIGN SP'23
	* TOOLS:   STMCUBEIDE
	* DATE:    06-05-2023
	* FILE:    ssd1306.h
	* BRIEF:   This is header file which is used for functions related to the SSD1306 OLED
	*
	*
******************************************************************************
 */
#ifndef SSD1306_H
#define SSD1306_H 100

/* C++ detection */
#ifdef __cplusplus
extern C {
#endif

#include "stm32f4xx_hal.h"
#include "fonts.h"
#include "stdlib.h"
#include "string.h"

/* I2C address */
#ifndef SSD1306_I2C_ADDR
#define SSD1306_I2C_ADDR         0x78 //slave address
#endif

/* SSD1306 settings */
/* SSD1306 width in pixels */
#ifndef SSD1306_WIDTH
#define SSD1306_WIDTH            128
#endif

/* SSD1306 LCD height in pixels */
#ifndef SSD1306_HEIGHT
```

```
#define SSD1306_HEIGHT          64
#endif

/**
 * @brief  SSD1306 color enum
 */
typedef enum {
	SSD1306_COLOR_BLACK = 0x00, /* no pixel, blank */
	SSD1306_COLOR_WHITE = 0x01  /* pixel is set to color of oled */
} SSD1306_COLOR_t;

uint8_t SSD1306_Init(void);
void SSD1306_UpdateScreen(void);
void SSD1306_Fill(SSD1306_COLOR_t Color);
void SSD1306_DrawPixel(uint16_t x, uint16_t y, SSD1306_COLOR_t color);
void SSD1306_GotoXY(uint16_t x, uint16_t y);
char SSD1306_Putc(char ch, FontDef_t* Font, SSD1306_COLOR_t color);
char SSD1306_Puts(char* str, FontDef_t* Font, SSD1306_COLOR_t color);

#ifndef ssd1306_I2C_TIMEOUT
#define ssd1306_I2C_TIMEOUT                                    20000 //i2c timeout
of 2000
#endif

void ssd1306_I2C_Write(uint8_t address, uint8_t reg, uint8_t data);
void ssd1306_I2C_WriteMulti(uint8_t address, uint8_t reg, uint8_t *data,
uint16_t count);
void SSD1306_ScrollRight(uint8_t start_row, uint8_t end_row);
void SSD1306_ScrollLeft(uint8_t start_row, uint8_t end_row);
void SSD1306_Stopscroll(void);
void SSD1306_Clear (void);


/* C++ detection */
#ifdef __cplusplus
}
#endif

#endif
```

Dcmotor.c

```
/* *
*******************************************************************************
	* PROJECT: SMART FAN CONTROL SYSTEM
	* AUTHOR:  ISHA SHARMA
	* COURSE:  EMBEDDED SYSTEMS DESIGN SP'23
	* TOOLS:   STMCUBEIDE
	* DATE:    06-05-2023
	* FILE:    dcmotor.c
	* BRIEF:   This file is used for the functions related to dcm motor and
pwm
	*
	*
*******************************************************************************
 */
```

```c
#include "stm32f4xx.h"
#include <stdio.h>
#include <stdint.h>
#include "dcmotor.h"

/*
name: init_gpio_pwmpin
description: function to initialise PWM output pin
parameters: none
returns: none
 */
void init_gpio_pwmpin(void)
{
    RCC->AHB1ENR |= RCC_AHB1ENR_GPIOBEN; // Enable clock for GPIO Port B

    // Configure PB9 as an alternate function output
    GPIOB->MODER |= GPIO_MODER_MODE9_1; // Set to alternate function mode
    GPIOB->AFR[1] |= (2 << GPIO_AFRH_AFSEL9_Pos); // Set alternate function to
AF2 (TIM4_CH4)

    // Set output type to push-pull
    GPIOB->OTYPER &= ~(GPIO_OTYPER_OT9);

    // Set output speed to high
    GPIOB->OSPEEDR |= GPIO_OSPEEDR_OSPEED9;

    // Set initial output to low
    GPIOB->BSRR |= GPIO_BSRR_BR9;
}

/*
name: init_motor_pins
description: function to initialise gpio pins for motor direction control
parameters: none
returns: none
 */
void init_motor_pins(void)
{
    // Enable GPIOB clock
    RCC->AHB1ENR |= RCC_AHB1ENR_GPIOBEN;

    // Set PB12 and PB13 as outputs
    GPIOB->MODER |= GPIO_MODER_MODE12_0 | GPIO_MODER_MODE13_0;
    GPIOB->MODER &= ~(GPIO_MODER_MODE12_1 | GPIO_MODER_MODE13_1);

    // Set output type to push-pull
    GPIOB->OTYPER &= ~(GPIO_OTYPER_OT12 | GPIO_OTYPER_OT13);

    // Set output speed to high
    GPIOB->OSPEEDR |= GPIO_OSPEEDR_OSPEED12 | GPIO_OSPEEDR_OSPEED13;

    // Set PB12 and PB13 initial output to low
    GPIOB->BSRR |= GPIO_BSRR_BR12 | GPIO_BSRR_BR13;

    // Set in1 to high and in2 to low
    GPIOB->BSRR |= GPIO_BSRR_BS12;
    GPIOB->BSRR &= ~(GPIO_BSRR_BS13);
}
```

```c
/*
name: init_pwm_timer
description: function to intiliase pwm
parameters: none
returns: none
 */void init_pwm_timer(void)
{
	// Enable clock for TIM4
	RCC->APB1ENR  |= RCC_APB1ENR_TIM4EN;

	// Set the prescaler value to achieve10 KHz freq with system clock of 84
MHz
	//PSC = (48 MHz / 10 kHz) - 1 = 4799
	TIM4->PSC = 4799;

	// Set the auto-reload value to achieve a frequency of 10 kHz
	//ARR = (48 MHz / (10 kHz * (4799 + 1))) - 1 = 999
	//PWM frequency = (system clock frequency) / (prescaler x (arr + 1))
	//frequency is determined by the arr register
	TIM4->ARR = 999;

	// Set the duty cycle using CCR4 register
	TIM4->CCR4 = duty_cycle;

	// Set output mode to PWM mode 1
	TIM4->CCMR2 |= TIM_CCMR2_OC4M_2 | TIM_CCMR2_OC4M_1;
	TIM4->CCMR2 &= ~(TIM_CCMR2_OC4M_0);

	// Enable output for channel 4
	TIM4->CCER |= TIM_CCER_CC4E;

	// Enable counter for TIM4
	TIM4->CR1 |= TIM_CR1_CEN;
}
```

Dcmotor.h

```c
/* *
******************************************************************************
	* PROJECT: SMART FAN CONTROL SYSTEM
	* AUTHOR:  ISHA SHARMA
	* COURSE:  EMBEDDED SYSTEMS DESIGN SP'23
	* TOOLS:   STMCUBEIDE
	* DATE:    06-05-2023
	* FILE:    dcmotor.h
	* BRIEF:   This is the header file used for the functions related to dcm
motor and pwm
	*
	*
******************************************************************************
 */

#ifndef INC_DCMOTOR_H_
#define INC_DCMOTOR_H_
```

```
extern uint32_t duty_cycle;
void init_gpio_pwmpin(void);
void init_motor_pins(void);
void init_pwm_timer(void);
#endif /* INC_DCMOTOR_H_ */
```

Led.c

```
/* *
*******************************************************************************
       * PROJECT: SMART FAN CONTROL SYSTEM
       * AUTHOR:  ISHA SHARMA
       * COURSE:  EMBEDDED SYSTEMS DESIGN SP'23
       * TOOLS:   STMCUBEIDE
       * DATE:    06-05-2023
       * FILE:    led.c
       * BRIEF:   This file is used for the functions related to indicator led
       *
       *
*******************************************************************************
 */

#include "stm32f4xx.h"
#include <stdio.h>
#include <stdint.h>
#include "led.h"

/*
name: init_led
description: function to initialise pin for led
parameters: none
returns: none
 */
void init_led(void)
{
    // Enable the GPIOA clock
       RCC->AHB1ENR |= RCC_GPIOA_EN;

    // Set the PA5 mode of the GPIO pin as output
    GPIOA->MODER |= GPIOA_P0_OUTPUT;

    // Turn off the LED initially
    GPIOA->BSRR |= LED_BSRR_OFF;
}
```

Led.h

```
/* *
*******************************************************************************
       * PROJECT: SMART FAN CONTROL SYSTEM
       * AUTHOR:  ISHA SHARMA
       * COURSE:  EMBEDDED SYSTEMS DESIGN SP'23
       * TOOLS:   STMCUBEIDE
       * DATE:    06-05-2023
       * FILE:    led.h
```

```
      * BRIEF:   This header file is used for the functions related to
indicator led
      *
      *
*******************************************************************************
 */

#ifndef INC_LED_H_
#define INC_LED_H_

void init_led(void);

#define LED_BSRR_ON (0b01) //bsrr register value for led to be lit
#define LED_BSRR_OFF (0b01<<16) //bsrr register value for the led to be turned
off
#define RCC_GPIOA_EN (0b01) //enable clock
#define GPIOA_P0_OUTPUT (0b01) //set as output

#endif /* INC_LED_H_ */
```

Stm32f4xx_hal_conf.h

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file    stm32f4xx_hal_conf_template.h
  * @author  MCD Application Team
  * @brief   HAL configuration template file.
  *          This file should be copied to the application folder and renamed
  *          to stm32f4xx_hal_conf.h.
  ******************************************************************************
  * @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.
  *
  ******************************************************************************
  */
/* USER CODE END Header */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_HAL_CONF_H
#define __STM32F4xx_HAL_CONF_H

#ifdef __cplusplus
 extern "C" {
#endif

/* Exported types ------------------------------------------------------------*/
/* Exported constants --------------------------------------------------------*/

/* ########################## Module Selection ############################## */
/**
```

```
  * @brief This is the list of modules to be used in the HAL driver
  */
#define HAL_MODULE_ENABLED

  /* #define HAL_CRYP_MODULE_ENABLED */
/* #define HAL_ADC_MODULE_ENABLED */
/* #define HAL_CAN_MODULE_ENABLED */
/* #define HAL_CRC_MODULE_ENABLED */
/* #define HAL_CAN_LEGACY_MODULE_ENABLED */
/* #define HAL_DAC_MODULE_ENABLED */
/* #define HAL_DCMI_MODULE_ENABLED */
/* #define HAL_DMA2D_MODULE_ENABLED */
/* #define HAL_ETH_MODULE_ENABLED */
/* #define HAL_ETH_LEGACY_MODULE_ENABLED */
/* #define HAL_NAND_MODULE_ENABLED */
/* #define HAL_NOR_MODULE_ENABLED */
/* #define HAL_PCCARD_MODULE_ENABLED */
/* #define HAL_SRAM_MODULE_ENABLED */
/* #define HAL_SDRAM_MODULE_ENABLED */
/* #define HAL_HASH_MODULE_ENABLED */
#define HAL_I2C_MODULE_ENABLED
/* #define HAL_I2S_MODULE_ENABLED */
/* #define HAL_IWDG_MODULE_ENABLED */
/* #define HAL_LTDC_MODULE_ENABLED */
/* #define HAL_RNG_MODULE_ENABLED */
/* #define HAL_RTC_MODULE_ENABLED */
/* #define HAL_SAI_MODULE_ENABLED */
/* #define HAL_SD_MODULE_ENABLED */
/* #define HAL_MMC_MODULE_ENABLED */
/* #define HAL_SPI_MODULE_ENABLED */
/* #define HAL_TIM_MODULE_ENABLED */
/* #define HAL_UART_MODULE_ENABLED */
/* #define HAL_USART_MODULE_ENABLED */
/* #define HAL_IRDA_MODULE_ENABLED */
/* #define HAL_SMARTCARD_MODULE_ENABLED */
/* #define HAL_SMBUS_MODULE_ENABLED */
/* #define HAL_WWDG_MODULE_ENABLED */
/* #define HAL_PCD_MODULE_ENABLED */
/* #define HAL_HCD_MODULE_ENABLED */
/* #define HAL_DSI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_QSPI_MODULE_ENABLED */
/* #define HAL_CEC_MODULE_ENABLED */
/* #define HAL_FMPI2C_MODULE_ENABLED */
/* #define HAL_FMPSMBUS_MODULE_ENABLED */
/* #define HAL_SPDIFRX_MODULE_ENABLED */
/* #define HAL_DFSDM_MODULE_ENABLED */
/* #define HAL_LPTIM_MODULE_ENABLED */
#define HAL_GPIO_MODULE_ENABLED
#define HAL_EXTI_MODULE_ENABLED
#define HAL_DMA_MODULE_ENABLED
#define HAL_RCC_MODULE_ENABLED
#define HAL_FLASH_MODULE_ENABLED
#define HAL_PWR_MODULE_ENABLED
#define HAL_CORTEX_MODULE_ENABLED

/* ########################## HSE/HSI Values adaptation ##################### */
/**
```

```
  * @brief Adjust the value of External High Speed oscillator (HSE) used in your
application.
  *        This value is used by the RCC HAL module to compute the system
frequency
  *        (when HSE is used as system clock source, directly or through the
PLL).
  */
#if !defined  (HSE_VALUE)
  #define HSE_VALUE    25000000U /*!< Value of the External oscillator in Hz */
#endif /* HSE_VALUE */

#if !defined  (HSE_STARTUP_TIMEOUT)
  #define HSE_STARTUP_TIMEOUT    100U   /*!< Time out for HSE start up, in ms */
#endif /* HSE_STARTUP_TIMEOUT */

/**
  * @brief Internal High Speed oscillator (HSI) value.
  *        This value is used by the RCC HAL module to compute the system
frequency
  *        (when HSI is used as system clock source, directly or through the
PLL).
  */
#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000U) /*!< Value of the Internal
oscillator in Hz*/
#endif /* HSI_VALUE */

/**
  * @brief Internal Low Speed oscillator (LSI) value.
  */
#if !defined  (LSI_VALUE)
 #define LSI_VALUE  32000U       /*!< LSI Typical Value in Hz*/
#endif /* LSI_VALUE */                      /*!< Value of the Internal Low Speed
oscillator in Hz
                                             The real value may vary depending
on the variations
                                             in voltage and temperature.*/
/**
  * @brief External Low Speed oscillator (LSE) value.
  */
#if !defined  (LSE_VALUE)
 #define LSE_VALUE  32768U    /*!< Value of the External Low Speed oscillator in
Hz */
#endif /* LSE_VALUE */

#if !defined  (LSE_STARTUP_TIMEOUT)
  #define LSE_STARTUP_TIMEOUT    5000U   /*!< Time out for LSE start up, in ms
*/
#endif /* LSE_STARTUP_TIMEOUT */

/**
  * @brief External clock source for I2S peripheral
  *        This value is used by the I2S HAL module to compute the I2S clock
source
  *        frequency, this source is inserted directly through I2S_CKIN pad.
  */
#if !defined  (EXTERNAL_CLOCK_VALUE)
  #define EXTERNAL_CLOCK_VALUE    12288000U /*!< Value of the External audio
frequency in Hz*/
```

```
#endif /* EXTERNAL_CLOCK_VALUE */

/* Tip: To avoid modifying this file each time you need to use different HSE,
   ===  you can define the HSE value in your toolchain compiler preprocessor. */

/* ########################### System Configuration ######################### */
/**
  * @brief This is the HAL system configuration section
  */
#define  VDD_VALUE               3300U /*!< Value of VDD in mv */
#define  TICK_INT_PRIORITY            15U   /*!< tick interrupt priority */
#define  USE_RTOS                     0U
#define  PREFETCH_ENABLE              1U
#define  INSTRUCTION_CACHE_ENABLE     1U
#define  DATA_CACHE_ENABLE            1U

#define  USE_HAL_ADC_REGISTER_CALLBACKS         0U /* ADC register callback
disabled       */
#define  USE_HAL_CAN_REGISTER_CALLBACKS         0U /* CAN register callback
disabled       */
#define  USE_HAL_CEC_REGISTER_CALLBACKS         0U /* CEC register callback
disabled       */
#define  USE_HAL_CRYP_REGISTER_CALLBACKS        0U /* CRYP register callback
disabled      */
#define  USE_HAL_DAC_REGISTER_CALLBACKS         0U /* DAC register callback
disabled       */
#define  USE_HAL_DCMI_REGISTER_CALLBACKS        0U /* DCMI register callback
disabled      */
#define  USE_HAL_DFSDM_REGISTER_CALLBACKS       0U /* DFSDM register callback
disabled     */
#define  USE_HAL_DMA2D_REGISTER_CALLBACKS       0U /* DMA2D register callback
disabled     */
#define  USE_HAL_DSI_REGISTER_CALLBACKS         0U /* DSI register callback
disabled       */
#define  USE_HAL_ETH_REGISTER_CALLBACKS         0U /* ETH register callback
disabled       */
#define  USE_HAL_HASH_REGISTER_CALLBACKS        0U /* HASH register callback
disabled      */
#define  USE_HAL_HCD_REGISTER_CALLBACKS         0U /* HCD register callback
disabled       */
#define  USE_HAL_I2C_REGISTER_CALLBACKS         0U /* I2C register callback
disabled       */
#define  USE_HAL_FMPI2C_REGISTER_CALLBACKS      0U /* FMPI2C register callback
disabled    */
#define  USE_HAL_FMPSMBUS_REGISTER_CALLBACKS    0U /* FMPSMBUS register callback
disabled  */
#define  USE_HAL_I2S_REGISTER_CALLBACKS         0U /* I2S register callback
disabled       */
#define  USE_HAL_IRDA_REGISTER_CALLBACKS        0U /* IRDA register callback
disabled      */
#define  USE_HAL_LPTIM_REGISTER_CALLBACKS       0U /* LPTIM register callback
disabled     */
#define  USE_HAL_LTDC_REGISTER_CALLBACKS        0U /* LTDC register callback
disabled      */
#define  USE_HAL_MMC_REGISTER_CALLBACKS         0U /* MMC register callback
disabled       */
#define  USE_HAL_NAND_REGISTER_CALLBACKS        0U /* NAND register callback
disabled      */
```

```
#define  USE_HAL_NOR_REGISTER_CALLBACKS         0U /* NOR register callback
disabled       */
#define  USE_HAL_PCCARD_REGISTER_CALLBACKS      0U /* PCCARD register callback
disabled    */
#define  USE_HAL_PCD_REGISTER_CALLBACKS         0U /* PCD register callback
disabled       */
#define  USE_HAL_QSPI_REGISTER_CALLBACKS        0U /* QSPI register callback
disabled      */
#define  USE_HAL_RNG_REGISTER_CALLBACKS         0U /* RNG register callback
disabled       */
#define  USE_HAL_RTC_REGISTER_CALLBACKS         0U /* RTC register callback
disabled       */
#define  USE_HAL_SAI_REGISTER_CALLBACKS         0U /* SAI register callback
disabled       */
#define  USE_HAL_SD_REGISTER_CALLBACKS          0U /* SD register callback
disabled        */
#define  USE_HAL_SMARTCARD_REGISTER_CALLBACKS   0U /* SMARTCARD register
callback disabled */
#define  USE_HAL_SDRAM_REGISTER_CALLBACKS       0U /* SDRAM register callback
disabled     */
#define  USE_HAL_SRAM_REGISTER_CALLBACKS        0U /* SRAM register callback
disabled      */
#define  USE_HAL_SPDIFRX_REGISTER_CALLBACKS     0U /* SPDIFRX register callback
disabled   */
#define  USE_HAL_SMBUS_REGISTER_CALLBACKS       0U /* SMBUS register callback
disabled     */
#define  USE_HAL_SPI_REGISTER_CALLBACKS         0U /* SPI register callback
disabled       */
#define  USE_HAL_TIM_REGISTER_CALLBACKS         0U /* TIM register callback
disabled       */
#define  USE_HAL_UART_REGISTER_CALLBACKS        0U /* UART register callback
disabled      */
#define  USE_HAL_USART_REGISTER_CALLBACKS       0U /* USART register callback
disabled     */
#define  USE_HAL_WWDG_REGISTER_CALLBACKS        0U /* WWDG register callback
disabled      */

/* ########################## Assert Selection ############################## */
/**
  * @brief Uncomment the line below to expanse the "assert_param" macro in the
  *        HAL drivers code
  */
/* #define USE_FULL_ASSERT    1U */

/* ################### Ethernet peripheral configuration ##################### */

/* Section 1 : Ethernet peripheral configuration */

/* MAC ADDRESS: MAC_ADDR0:MAC_ADDR1:MAC_ADDR2:MAC_ADDR3:MAC_ADDR4:MAC_ADDR5 */
#define MAC_ADDR0   2U
#define MAC_ADDR1   0U
#define MAC_ADDR2   0U
#define MAC_ADDR3   0U
#define MAC_ADDR4   0U
#define MAC_ADDR5   0U

/* Definition of the Ethernet driver buffers size and count */
#define ETH_RX_BUF_SIZE                /* buffer size for receive
*/
```

```
#define ETH_TX_BUF_SIZE                ETH_MAX_PACKET_SIZE /* buffer size for
transmit             */
#define ETH_RXBUFNB                    4U       /* 4 Rx buffers of size
ETH_RX_BUF_SIZE  */
#define ETH_TXBUFNB                    4U       /* 4 Tx buffers of size
ETH_TX_BUF_SIZE  */

/* Section 2: PHY configuration section */

/* DP83848_PHY_ADDRESS Address*/
#define DP83848_PHY_ADDRESS           0x01U
/* PHY Reset delay these values are based on a 1 ms Systick interrupt*/
#define PHY_RESET_DELAY                0x000000FFU
/* PHY Configuration delay */
#define PHY_CONFIG_DELAY               0x00000FFFU

#define PHY_READ_TO                    0x0000FFFFU
#define PHY_WRITE_TO                   0x0000FFFFU

/* Section 3: Common PHY Registers */

#define PHY_BCR                        ((uint16_t)0x0000U)    /*!< Transceiver
Basic Control Register   */
#define PHY_BSR                        ((uint16_t)0x0001U)    /*!< Transceiver
Basic Status Register    */

#define PHY_RESET                      ((uint16_t)0x8000U)  /*!< PHY Reset */
#define PHY_LOOPBACK                   ((uint16_t)0x4000U)  /*!< Select loop-
back mode */
#define PHY_FULLDUPLEX_100M            ((uint16_t)0x2100U)  /*!< Set the full-
duplex mode at 100 Mb/s */
#define PHY_HALFDUPLEX_100M            ((uint16_t)0x2000U)  /*!< Set the half-
duplex mode at 100 Mb/s */
#define PHY_FULLDUPLEX_10M             ((uint16_t)0x0100U)  /*!< Set the full-
duplex mode at 10 Mb/s  */
#define PHY_HALFDUPLEX_10M             ((uint16_t)0x0000U)  /*!< Set the half-
duplex mode at 10 Mb/s  */
#define PHY_AUTONEGOTIATION            ((uint16_t)0x1000U)  /*!< Enable auto-
negotiation function     */
#define PHY_RESTART_AUTONEGOTIATION    ((uint16_t)0x0200U)  /*!< Restart auto-
negotiation function    */
#define PHY_POWERDOWN                  ((uint16_t)0x0800U)  /*!< Select the
power down mode           */
#define PHY_ISOLATE                    ((uint16_t)0x0400U)  /*!< Isolate PHY
from MII                 */

#define PHY_AUTONEGO_COMPLETE          ((uint16_t)0x0020U)  /*!< Auto-
Negotiation process completed   */
#define PHY_LINKED_STATUS              ((uint16_t)0x0004U)  /*!< Valid link
established               */
#define PHY_JABBER_DETECTION           ((uint16_t)0x0002U)  /*!< Jabber
condition detected            */

/* Section 4: Extended PHY Registers */
#define PHY_SR                         ((uint16_t)0x10U)    /*!< PHY status
register Offset                     */

#define PHY_SPEED_STATUS               ((uint16_t)0x0002U)  /*!< PHY Speed mask
*/
```

```
#define PHY_DUPLEX_STATUS                ((uint16_t)0x0004U)  /*!< PHY Duplex
mask                                  */

/* ################## SPI peripheral configuration ############################ */

/* CRC FEATURE: Use to activate CRC feature inside HAL SPI Driver
* Activated: CRC code is present inside driver
* Deactivated: CRC code cleaned from driver
*/

#define USE_SPI_CRC                      0U

/* Includes ------------------------------------------------------------------*/
/**
  * @brief Include module's header file
  */

#ifdef HAL_RCC_MODULE_ENABLED
  #include "stm32f4xx_hal_rcc.h"
#endif /* HAL_RCC_MODULE_ENABLED */

#ifdef HAL_GPIO_MODULE_ENABLED
  #include "stm32f4xx_hal_gpio.h"
#endif /* HAL_GPIO_MODULE_ENABLED */

#ifdef HAL_EXTI_MODULE_ENABLED
  #include "stm32f4xx_hal_exti.h"
#endif /* HAL_EXTI_MODULE_ENABLED */

#ifdef HAL_DMA_MODULE_ENABLED
  #include "stm32f4xx_hal_dma.h"
#endif /* HAL_DMA_MODULE_ENABLED */

#ifdef HAL_CORTEX_MODULE_ENABLED
  #include "stm32f4xx_hal_cortex.h"
#endif /* HAL_CORTEX_MODULE_ENABLED */

#ifdef HAL_ADC_MODULE_ENABLED
  #include "stm32f4xx_hal_adc.h"
#endif /* HAL_ADC_MODULE_ENABLED */

#ifdef HAL_CAN_MODULE_ENABLED
  #include "stm32f4xx_hal_can.h"
#endif /* HAL_CAN_MODULE_ENABLED */

#ifdef HAL_CAN_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_can_legacy.h"
#endif /* HAL_CAN_LEGACY_MODULE_ENABLED */

#ifdef HAL_CRC_MODULE_ENABLED
  #include "stm32f4xx_hal_crc.h"
#endif /* HAL_CRC_MODULE_ENABLED */

#ifdef HAL_CRYP_MODULE_ENABLED
  #include "stm32f4xx_hal_cryp.h"
#endif /* HAL_CRYP_MODULE_ENABLED */

#ifdef HAL_DMA2D_MODULE_ENABLED
  #include "stm32f4xx_hal_dma2d.h"
```

```
#endif /* HAL_DMA2D_MODULE_ENABLED */

#ifdef HAL_DAC_MODULE_ENABLED
  #include "stm32f4xx_hal_dac.h"
#endif /* HAL_DAC_MODULE_ENABLED */

#ifdef HAL_DCMI_MODULE_ENABLED
  #include "stm32f4xx_hal_dcmi.h"
#endif /* HAL_DCMI_MODULE_ENABLED */

#ifdef HAL_ETH_MODULE_ENABLED
  #include "stm32f4xx_hal_eth.h"
#endif /* HAL_ETH_MODULE_ENABLED */

#ifdef HAL_ETH_LEGACY_MODULE_ENABLED
  #include "stm32f4xx_hal_eth_legacy.h"
#endif /* HAL_ETH_LEGACY_MODULE_ENABLED */

#ifdef HAL_FLASH_MODULE_ENABLED
  #include "stm32f4xx_hal_flash.h"
#endif /* HAL_FLASH_MODULE_ENABLED */

#ifdef HAL_SRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sram.h"
#endif /* HAL_SRAM_MODULE_ENABLED */

#ifdef HAL_NOR_MODULE_ENABLED
  #include "stm32f4xx_hal_nor.h"
#endif /* HAL_NOR_MODULE_ENABLED */

#ifdef HAL_NAND_MODULE_ENABLED
  #include "stm32f4xx_hal_nand.h"
#endif /* HAL_NAND_MODULE_ENABLED */

#ifdef HAL_PCCARD_MODULE_ENABLED
  #include "stm32f4xx_hal_pccard.h"
#endif /* HAL_PCCARD_MODULE_ENABLED */

#ifdef HAL_SDRAM_MODULE_ENABLED
  #include "stm32f4xx_hal_sdram.h"
#endif /* HAL_SDRAM_MODULE_ENABLED */

#ifdef HAL_HASH_MODULE_ENABLED
 #include "stm32f4xx_hal_hash.h"
#endif /* HAL_HASH_MODULE_ENABLED */

#ifdef HAL_I2C_MODULE_ENABLED
 #include "stm32f4xx_hal_i2c.h"
#endif /* HAL_I2C_MODULE_ENABLED */

#ifdef HAL_SMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_smbus.h"
#endif /* HAL_SMBUS_MODULE_ENABLED */

#ifdef HAL_I2S_MODULE_ENABLED
 #include "stm32f4xx_hal_i2s.h"
#endif /* HAL_I2S_MODULE_ENABLED */

#ifdef HAL_IWDG_MODULE_ENABLED
```

```
 #include "stm32f4xx_hal_iwdg.h"
#endif /* HAL_IWDG_MODULE_ENABLED */

#ifdef HAL_LTDC_MODULE_ENABLED
 #include "stm32f4xx_hal_ltdc.h"
#endif /* HAL_LTDC_MODULE_ENABLED */

#ifdef HAL_PWR_MODULE_ENABLED
 #include "stm32f4xx_hal_pwr.h"
#endif /* HAL_PWR_MODULE_ENABLED */

#ifdef HAL_RNG_MODULE_ENABLED
 #include "stm32f4xx_hal_rng.h"
#endif /* HAL_RNG_MODULE_ENABLED */

#ifdef HAL_RTC_MODULE_ENABLED
 #include "stm32f4xx_hal_rtc.h"
#endif /* HAL_RTC_MODULE_ENABLED */

#ifdef HAL_SAI_MODULE_ENABLED
 #include "stm32f4xx_hal_sai.h"
#endif /* HAL_SAI_MODULE_ENABLED */

#ifdef HAL_SD_MODULE_ENABLED
 #include "stm32f4xx_hal_sd.h"
#endif /* HAL_SD_MODULE_ENABLED */

#ifdef HAL_SPI_MODULE_ENABLED
 #include "stm32f4xx_hal_spi.h"
#endif /* HAL_SPI_MODULE_ENABLED */

#ifdef HAL_TIM_MODULE_ENABLED
 #include "stm32f4xx_hal_tim.h"
#endif /* HAL_TIM_MODULE_ENABLED */

#ifdef HAL_UART_MODULE_ENABLED
 #include "stm32f4xx_hal_uart.h"
#endif /* HAL_UART_MODULE_ENABLED */

#ifdef HAL_USART_MODULE_ENABLED
 #include "stm32f4xx_hal_usart.h"
#endif /* HAL_USART_MODULE_ENABLED */

#ifdef HAL_IRDA_MODULE_ENABLED
 #include "stm32f4xx_hal_irda.h"
#endif /* HAL_IRDA_MODULE_ENABLED */

#ifdef HAL_SMARTCARD_MODULE_ENABLED
 #include "stm32f4xx_hal_smartcard.h"
#endif /* HAL_SMARTCARD_MODULE_ENABLED */

#ifdef HAL_WWDG_MODULE_ENABLED
 #include "stm32f4xx_hal_wwdg.h"
#endif /* HAL_WWDG_MODULE_ENABLED */

#ifdef HAL_PCD_MODULE_ENABLED
 #include "stm32f4xx_hal_pcd.h"
#endif /* HAL_PCD_MODULE_ENABLED */
```

```
#ifdef HAL_HCD_MODULE_ENABLED
 #include "stm32f4xx_hal_hcd.h"
#endif /* HAL_HCD_MODULE_ENABLED */

#ifdef HAL_DSI_MODULE_ENABLED
 #include "stm32f4xx_hal_dsi.h"
#endif /* HAL_DSI_MODULE_ENABLED */

#ifdef HAL_QSPI_MODULE_ENABLED
 #include "stm32f4xx_hal_qspi.h"
#endif /* HAL_QSPI_MODULE_ENABLED */

#ifdef HAL_CEC_MODULE_ENABLED
 #include "stm32f4xx_hal_cec.h"
#endif /* HAL_CEC_MODULE_ENABLED */

#ifdef HAL_FMPI2C_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpi2c.h"
#endif /* HAL_FMPI2C_MODULE_ENABLED */

#ifdef HAL_FMPSMBUS_MODULE_ENABLED
 #include "stm32f4xx_hal_fmpsmbus.h"
#endif /* HAL_FMPSMBUS_MODULE_ENABLED */

#ifdef HAL_SPDIFRX_MODULE_ENABLED
 #include "stm32f4xx_hal_spdifrx.h"
#endif /* HAL_SPDIFRX_MODULE_ENABLED */

#ifdef HAL_DFSDM_MODULE_ENABLED
 #include "stm32f4xx_hal_dfsdm.h"
#endif /* HAL_DFSDM_MODULE_ENABLED */

#ifdef HAL_LPTIM_MODULE_ENABLED
 #include "stm32f4xx_hal_lptim.h"
#endif /* HAL_LPTIM_MODULE_ENABLED */

#ifdef HAL_MMC_MODULE_ENABLED
 #include "stm32f4xx_hal_mmc.h"
#endif /* HAL_MMC_MODULE_ENABLED */

/* Exported macro ------------------------------------------------------------*/
#ifdef  USE_FULL_ASSERT
/**
  * @brief  The assert_param macro is used for function's parameters check.
  * @param  expr If expr is false, it calls assert_failed function
  *         which reports the name of the source file and the source
  *         line number of the call that failed.
  *         If expr is true, it returns no value.
  * @retval None
  */
  #define assert_param(expr) ((expr) ? (void)0U : assert_failed((uint8_t
*)__FILE__, __LINE__))
/* Exported functions ------------------------------------------------------- */
  void assert_failed(uint8_t* file, uint32_t line);
#else
  #define assert_param(expr) ((void)0U)
#endif /* USE_FULL_ASSERT */

#ifdef __cplusplus
```

```
}
#endif

#endif /* __STM32F4xx_HAL_CONF_H */
```

Stm32f4xx_it.h

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file    stm32f4xx_it.h
  * @brief   This file contains the headers of the interrupt handlers.
  ******************************************************************************
  * @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 */

/* Define to prevent recursive inclusion -------------------------------------*/
#ifndef __STM32F4xx_IT_H
#define __STM32F4xx_IT_H

#ifdef __cplusplus
 extern "C" {
#endif

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

/* USER CODE END Includes */

/* Exported types ------------------------------------------------------------*/
/* USER CODE BEGIN ET */

/* USER CODE END ET */

/* Exported constants --------------------------------------------------------*/
/* USER CODE BEGIN EC */

/* USER CODE END EC */

/* Exported macro ------------------------------------------------------------*/
/* USER CODE BEGIN EM */

/* USER CODE END EM */

/* Exported functions prototypes ---------------------------------------------*/
void NMI_Handler(void);
void HardFault_Handler(void);
```

```
void MemManage_Handler(void);
void BusFault_Handler(void);
void UsageFault_Handler(void);
void SVC_Handler(void);
void DebugMon_Handler(void);
void PendSV_Handler(void);
void SysTick_Handler(void);
/* USER CODE BEGIN EFP */

/* USER CODE END EFP */

#ifdef __cplusplus
}
#endif

#endif /* __STM32F4xx_IT_H */
```

stm32f4xx_hal_msp.c

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file         stm32f4xx_hal_msp.c
  * @brief        This file provides code for the MSP Initialization
  *               and de-Initialization codes.
  ******************************************************************************
  * @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"
/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

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

/* USER CODE END TD */

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

/* USER CODE END Define */

/* Private macro -------------------------------------------------------------*/
/* USER CODE BEGIN Macro */
```

```c
/* USER CODE END Macro */

/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* External functions --------------------------------------------------------*/
/* USER CODE BEGIN ExternalFunctions */

/* USER CODE END ExternalFunctions */

/* USER CODE BEGIN 0 */

/* USER CODE END 0 */
/**
  * Initializes the Global MSP.
  */
void HAL_MspInit(void)
{
  /* USER CODE BEGIN MspInit 0 */

  /* USER CODE END MspInit 0 */

  __HAL_RCC_SYSCFG_CLK_ENABLE();
  __HAL_RCC_PWR_CLK_ENABLE();

  /* System interrupt init*/

  /* USER CODE BEGIN MspInit 1 */

  /* USER CODE END MspInit 1 */
}

/**
* @brief I2C MSP Initialization
* This function configures the hardware resources used in this example
* @param hi2c: I2C handle pointer
* @retval None
*/
void HAL_I2C_MspInit(I2C_HandleTypeDef* hi2c)
{
  GPIO_InitTypeDef GPIO_InitStruct = {0};
  if(hi2c->Instance==I2C1)
  {
  /* USER CODE BEGIN I2C1_MspInit 0 */

  /* USER CODE END I2C1_MspInit 0 */

    __HAL_RCC_GPIOB_CLK_ENABLE();
    /**I2C1 GPIO Configuration
    PB6     ------> I2C1_SCL
    PB7     ------> I2C1_SDA
```

```
    */
    GPIO_InitStruct.Pin = GPIO_PIN_6|GPIO_PIN_7;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF4_I2C1;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

    /* Peripheral clock enable */
    __HAL_RCC_I2C1_CLK_ENABLE();
  /* USER CODE BEGIN I2C1_MspInit 1 */

  /* USER CODE END I2C1_MspInit 1 */
  }
  else if(hi2c->Instance==I2C2)
  {
  /* USER CODE BEGIN I2C2_MspInit 0 */

  /* USER CODE END I2C2_MspInit 0 */

    __HAL_RCC_GPIOB_CLK_ENABLE();
    /**I2C2 GPIO Configuration
    PB10     ------> I2C2_SCL
    PB3     ------> I2C2_SDA
    */
    GPIO_InitStruct.Pin = GPIO_PIN_10;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF4_I2C2;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

    GPIO_InitStruct.Pin = GPIO_PIN_3;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_VERY_HIGH;
    GPIO_InitStruct.Alternate = GPIO_AF9_I2C2;
    HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

    /* Peripheral clock enable */
    __HAL_RCC_I2C2_CLK_ENABLE();
  /* USER CODE BEGIN I2C2_MspInit 1 */

  /* USER CODE END I2C2_MspInit 1 */
  }

}

/**
* @brief I2C MSP De-Initialization
* This function freeze the hardware resources used in this example
* @param hi2c: I2C handle pointer
* @retval None
*/
void HAL_I2C_MspDeInit(I2C_HandleTypeDef* hi2c)
{
  if(hi2c->Instance==I2C1)
  {
  /* USER CODE BEGIN I2C1_MspDeInit 0 */
```

```
  /* USER CODE END I2C1_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_I2C1_CLK_DISABLE();

    /**I2C1 GPIO Configuration
    PB6     ------> I2C1_SCL
    PB7     ------> I2C1_SDA
    */
    HAL_GPIO_DeInit(GPIOB, GPIO_PIN_6);

    HAL_GPIO_DeInit(GPIOB, GPIO_PIN_7);

  /* USER CODE BEGIN I2C1_MspDeInit 1 */

  /* USER CODE END I2C1_MspDeInit 1 */
  }
  else if(hi2c->Instance==I2C2)
  {
  /* USER CODE BEGIN I2C2_MspDeInit 0 */

  /* USER CODE END I2C2_MspDeInit 0 */
    /* Peripheral clock disable */
    __HAL_RCC_I2C2_CLK_DISABLE();

    /**I2C2 GPIO Configuration
    PB10     ------> I2C2_SCL
    PB3     ------> I2C2_SDA
    */
    HAL_GPIO_DeInit(GPIOB, GPIO_PIN_10);

    HAL_GPIO_DeInit(GPIOB, GPIO_PIN_3);

  /* USER CODE BEGIN I2C2_MspDeInit 1 */

  /* USER CODE END I2C2_MspDeInit 1 */
  }

}

/* USER CODE BEGIN 1 */

/* USER CODE END 1 */
```

Stm32f4xx_it.h

```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file    stm32f4xx_it.c
  * @brief   Interrupt Service Routines.
  ******************************************************************************
  * @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"
#include "stm32f4xx_it.h"
/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
/* USER CODE END Includes */

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

/* USER CODE END TD */

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

/* USER CODE END PD */

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

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

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

/* USER CODE END 0 */

/* External variables --------------------------------------------------------*/

/* USER CODE BEGIN EV */

/* USER CODE END EV */

/******************************************************************************/
/*           Cortex-M4 Processor Interruption and Exception Handlers          */
/******************************************************************************/
/**
  * @brief This function handles Non maskable interrupt.
  */
void NMI_Handler(void)
{
```

```
  /* USER CODE BEGIN NonMaskableInt_IRQn 0 */

  /* USER CODE END NonMaskableInt_IRQn 0 */
  /* USER CODE BEGIN NonMaskableInt_IRQn 1 */
  while (1)
  {
  }
  /* USER CODE END NonMaskableInt_IRQn 1 */
}

/**
  * @brief This function handles Hard fault interrupt.
  */
void HardFault_Handler(void)
{
  /* USER CODE BEGIN HardFault_IRQn 0 */

  /* USER CODE END HardFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_HardFault_IRQn 0 */
    /* USER CODE END W1_HardFault_IRQn 0 */
  }
}

/**
  * @brief This function handles Memory management fault.
  */
void MemManage_Handler(void)
{
  /* USER CODE BEGIN MemoryManagement_IRQn 0 */

  /* USER CODE END MemoryManagement_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_MemoryManagement_IRQn 0 */
    /* USER CODE END W1_MemoryManagement_IRQn 0 */
  }
}

/**
  * @brief This function handles Pre-fetch fault, memory access fault.
  */
void BusFault_Handler(void)
{
  /* USER CODE BEGIN BusFault_IRQn 0 */

  /* USER CODE END BusFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_BusFault_IRQn 0 */
    /* USER CODE END W1_BusFault_IRQn 0 */
  }
}

/**
  * @brief This function handles Undefined instruction or illegal state.
  */
void UsageFault_Handler(void)
```

```
{
  /* USER CODE BEGIN UsageFault_IRQn 0 */

  /* USER CODE END UsageFault_IRQn 0 */
  while (1)
  {
    /* USER CODE BEGIN W1_UsageFault_IRQn 0 */
    /* USER CODE END W1_UsageFault_IRQn 0 */
  }
}

/**
  * @brief This function handles System service call via SWI instruction.
  */
void SVC_Handler(void)
{
  /* USER CODE BEGIN SVCall_IRQn 0 */

  /* USER CODE END SVCall_IRQn 0 */
  /* USER CODE BEGIN SVCall_IRQn 1 */

  /* USER CODE END SVCall_IRQn 1 */
}

/**
  * @brief This function handles Debug monitor.
  */
void DebugMon_Handler(void)
{
  /* USER CODE BEGIN DebugMonitor_IRQn 0 */

  /* USER CODE END DebugMonitor_IRQn 0 */
  /* USER CODE BEGIN DebugMonitor_IRQn 1 */

  /* USER CODE END DebugMonitor_IRQn 1 */
}

/**
  * @brief This function handles Pendable request for system service.
  */
void PendSV_Handler(void)
{
  /* USER CODE BEGIN PendSV_IRQn 0 */

  /* USER CODE END PendSV_IRQn 0 */
  /* USER CODE BEGIN PendSV_IRQn 1 */

  /* USER CODE END PendSV_IRQn 1 */
}

/**
  * @brief This function handles System tick timer.
  */
void SysTick_Handler(void)
{
  /* USER CODE BEGIN SysTick_IRQn 0 */

  /* USER CODE END SysTick_IRQn 0 */
  HAL_IncTick();
```

```
  /* USER CODE BEGIN SysTick_IRQn 1 */

  /* USER CODE END SysTick_IRQn 1 */
}

/******************************************************************************/
/* STM32F4xx Peripheral Interrupt Handlers                                    */
/* Add here the Interrupt Handlers for the used peripherals.                  */
/* For the available peripheral interrupt handler names,                      */
/* please refer to the startup file (startup_stm32f4xx.s).                    */
/******************************************************************************/

/* USER CODE BEGIN 1 */

/* USER CODE END 1 */
```

Syscalls.c

```
/**
 ******************************************************************************
 * @file      syscalls.c
 * @author    Auto-generated by STM32CubeIDE
 * @brief     STM32CubeIDE Minimal System calls file
 *
 *            For more information about which c-functions
 *            need which of these lowlevel functions
 *            please consult the Newlib libc-manual
 ******************************************************************************
 * @attention
 *
 * Copyright (c) 2020-2023 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.
 *
 ******************************************************************************
 */

/* Includes */
#include <sys/stat.h>
#include <stdlib.h>
#include <errno.h>
#include <stdio.h>
#include <signal.h>
#include <time.h>
#include <sys/time.h>
#include <sys/times.h>


/* Variables */
extern int __io_putchar(int ch) __attribute__((weak));
extern int __io_getchar(void) __attribute__((weak));


char *__env[1] = { 0 };
```

```
char **environ = __env;


/* Functions */
void initialise_monitor_handles()
{
}

int _getpid(void)
{
  return 1;
}

int _kill(int pid, int sig)
{
  (void)pid;
  (void)sig;
  errno = EINVAL;
  return -1;
}

void _exit (int status)
{
  _kill(status, -1);
  while (1) {}    /* Make sure we hang here */
}

__attribute__((weak)) int _read(int file, char *ptr, int len)
{
  (void)file;
  int DataIdx;

  for (DataIdx = 0; DataIdx < len; DataIdx++)
  {
    *ptr++ = __io_getchar();
  }

  return len;
}

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

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

int _close(int file)
{
  (void)file;
  return -1;
}
```

```c
int _fstat(int file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}

int _isatty(int file)
{
  (void)file;
  return 1;
}

int _lseek(int file, int ptr, int dir)
{
  (void)file;
  (void)ptr;
  (void)dir;
  return 0;
}

int _open(char *path, int flags, ...)
{
  (void)path;
  (void)flags;
  /* Pretend like we always fail */
  return -1;
}

int _wait(int *status)
{
  (void)status;
  errno = ECHILD;
  return -1;
}

int _unlink(char *name)
{
  (void)name;
  errno = ENOENT;
  return -1;
}

int _times(struct tms *buf)
{
  (void)buf;
  return -1;
}

int _stat(char *file, struct stat *st)
{
  (void)file;
  st->st_mode = S_IFCHR;
  return 0;
}

int _link(char *old, char *new)
{
  (void)old;
```

```
  (void)new;
  errno = EMLINK;
  return -1;
}

int _fork(void)
{
  errno = EAGAIN;
  return -1;
}

int _execve(char *name, char **argv, char **env)
{
  (void)name;
  (void)argv;
  (void)env;
  errno = ENOMEM;
  return -1;
}
```

Sysmem.c

```
/**
 ******************************************************************************
 * @file      sysmem.c
 * @author    Generated by STM32CubeIDE
 * @brief     STM32CubeIDE System Memory calls file
 *
 *            For more information about which C functions
 *            need which of these lowlevel functions
 *            please consult the newlib libc manual
 ******************************************************************************
 * @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.
 *
 ******************************************************************************
 */

/* Includes */
#include <errno.h>
#include <stdint.h>

/**
 * Pointer to the current high watermark of the heap usage
 */
static uint8_t *__sbrk_heap_end = NULL;

/**
 * @brief _sbrk() allocates memory to the newlib heap and is used by malloc
 *        and others from the C library
 *
```

```
 * @verbatim
 * ##################################################################################
 * #  .data  #  .bss  #       newlib heap       #          MSP stack          #
 * #         #        #                         # Reserved by _Min_Stack_Size #
 * ##################################################################################
 * ^-- RAM start      ^-- _end                             _estack, RAM end --^
 * @endverbatim
 *
 * This implementation starts allocating at the '_end' linker symbol
 * The '_Min_Stack_Size' linker symbol reserves a memory for the MSP stack
 * The implementation considers '_estack' linker symbol to be RAM end
 * NOTE: If the MSP stack, at any point during execution, grows larger than the
 * reserved size, please increase the '_Min_Stack_Size'.
 *
 * @param incr Memory size
 * @return Pointer to allocated memory
 */
void *_sbrk(ptrdiff_t incr)
{
  extern uint8_t _end; /* Symbol defined in the linker script */
  extern uint8_t _estack; /* Symbol defined in the linker script */
  extern uint32_t _Min_Stack_Size; /* Symbol defined in the linker script */
  const uint32_t stack_limit = (uint32_t)&_estack - (uint32_t)&_Min_Stack_Size;
  const uint8_t *max_heap = (uint8_t *)stack_limit;
  uint8_t *prev_heap_end;

  /* Initialize heap end at first call */
  if (NULL == __sbrk_heap_end)
  {
    __sbrk_heap_end = &_end;
  }

  /* Protect heap from growing into the reserved MSP stack */
  if (__sbrk_heap_end + incr > max_heap)
  {
    errno = ENOMEM;
    return (void *)-1;
  }

  prev_heap_end = __sbrk_heap_end;
  __sbrk_heap_end += incr;

  return (void *)prev_heap_end;
}
```

System_stm32f4xx.c

```
/**
  ******************************************************************************
  * @file    system_stm32f4xx.c
  * @author  MCD Application Team
  * @brief   CMSIS Cortex-M4 Device Peripheral Access Layer System Source File.
  *
  *   This file provides two functions and one global variable to be called from
  *   user application:
  *      - SystemInit(): This function is called at startup just after reset and
  *                      before branch to main program. This call is made inside
```

```
  *                      the "startup_stm32f4xx.s" file.
  *
  *      - SystemCoreClock variable: Contains the core clock (HCLK), it can be
used
  *                                  by the user application to setup the
SysTick
  *                                  timer or configure other parameters.
  *
  *      - SystemCoreClockUpdate(): Updates the variable SystemCoreClock and
must
  *                                 be called whenever the core clock is changed
  *                                 during program execution.
  *
  *
  ******************************************************************************
  * @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.
  *
  ******************************************************************************
  */

/** @addtogroup CMSIS
  * @{
  */

/** @addtogroup stm32f4xx_system
  * @{
  */

/** @addtogroup STM32F4xx_System_Private_Includes
  * @{
  */


#include "stm32f4xx.h"

#if !defined  (HSE_VALUE)
  #define HSE_VALUE    ((uint32_t)25000000) /*!< Default value of the External
oscillator in Hz */
#endif /* HSE_VALUE */

#if !defined  (HSI_VALUE)
  #define HSI_VALUE    ((uint32_t)16000000) /*!< Value of the Internal
oscillator in Hz*/
#endif /* HSI_VALUE */

/**
  * @}
  */

/** @addtogroup STM32F4xx_System_Private_TypesDefinitions
  * @{
  */
```

```
/**
  * @}
  */

/** @addtogroup STM32F4xx_System_Private_Defines
  * @{
  */

/************************ Miscellaneous Configuration ************************/
/*!< Uncomment the following line if you need to use external SRAM or SDRAM as
data memory  */
#if defined(STM32F405xx) || defined(STM32F415xx) || defined(STM32F407xx) ||
defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) ||
defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx) || defined(STM32F412Zx) ||
defined(STM32F412Vx)
/* #define DATA_IN_ExtSRAM */
#endif /* STM32F40xxx || STM32F41xxx || STM32F42xxx || STM32F43xxx ||
STM32F469xx || STM32F479xx ||\
          STM32F412Zx || STM32F412Vx */

#if defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) ||
defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) || defined(STM32F479xx)
/* #define DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx || STM32F439xx ||
STM32F446xx || STM32F469xx ||\
          STM32F479xx */

/* Note: Following vector table addresses must be defined in line with linker
         configuration. */
/*!< Uncomment the following line if you need to relocate the vector table
     anywhere in Flash or Sram, else the vector table is kept at the automatic
     remap of boot address selected */
/* #define USER_VECT_TAB_ADDRESS */

#if defined(USER_VECT_TAB_ADDRESS)
/*!< Uncomment the following line if you need to relocate your vector Table
     in Sram else user remap will be done in Flash. */
/* #define VECT_TAB_SRAM */
#if defined(VECT_TAB_SRAM)
#define VECT_TAB_BASE_ADDRESS   SRAM_BASE       /*!< Vector Table base address
field.
                                                     This value must be a
multiple of 0x200. */
#define VECT_TAB_OFFSET         0x00000000U     /*!< Vector Table base offset
field.
                                                     This value must be a
multiple of 0x200. */
#else
#define VECT_TAB_BASE_ADDRESS   FLASH_BASE      /*!< Vector Table base address
field.
                                                     This value must be a
multiple of 0x200. */
#define VECT_TAB_OFFSET         0x00000000U     /*!< Vector Table base offset
field.
```

```
                                                    This value must be a
multiple of 0x200. */
#endif /* VECT_TAB_SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
/******************************************************************************/

/**
  * @}
  */

/** @addtogroup STM32F4xx_System_Private_Macros
  * @{
  */

/**
  * @}
  */

/** @addtogroup STM32F4xx_System_Private_Variables
  * @{
  */
  /* This variable is updated in three ways:
      1) by calling CMSIS function SystemCoreClockUpdate()
      2) by calling HAL API function HAL_RCC_GetHCLKFreq()
      3) each time HAL_RCC_ClockConfig() is called to configure the system clock
frequency
         Note: If you use this function to configure the system clock; then
there
               is no need to call the 2 first functions listed above, since
SystemCoreClock
               variable is updated automatically.
  */
uint32_t SystemCoreClock = 16000000;
const uint8_t AHBPrescTable[16] = {0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 6, 7, 8,
9};
const uint8_t APBPrescTable[8]  = {0, 0, 0, 0, 1, 2, 3, 4};
/**
  * @}
  */

/** @addtogroup STM32F4xx_System_Private_FunctionPrototypes
  * @{
  */

#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  static void SystemInit_ExtMemCtl(void);
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */

/**
  * @}
  */

/** @addtogroup STM32F4xx_System_Private_Functions
  * @{
  */

/**
  * @brief  Setup the microcontroller system
```

```
  *         Initialize the FPU setting, vector table location and External
memory
  *         configuration.
  * @param  None
  * @retval None
  */
void SystemInit(void)
{
  /* FPU settings ------------------------------------------------------------*/
  #if (__FPU_PRESENT == 1) && (__FPU_USED == 1)
    SCB->CPACR |= ((3UL << 10*2)|(3UL << 11*2));  /* set CP10 and CP11 Full
Access */
  #endif

#if defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
  SystemInit_ExtMemCtl();
#endif /* DATA_IN_ExtSRAM || DATA_IN_ExtSDRAM */

  /* Configure the Vector Table location -------------------------------------*/
#if defined(USER_VECT_TAB_ADDRESS)
  SCB->VTOR = VECT_TAB_BASE_ADDRESS | VECT_TAB_OFFSET; /* Vector Table
Relocation in Internal SRAM */
#endif /* USER_VECT_TAB_ADDRESS */
}

/**
   * @brief  Update SystemCoreClock variable according to Clock Register Values.
  *         The SystemCoreClock variable contains the core clock (HCLK), it can
  *         be used by the user application to setup the SysTick timer or
configure
  *         other parameters.
  *
  * @note   Each time the core clock (HCLK) changes, this function must be
called
  *         to update SystemCoreClock variable value. Otherwise, any
configuration
  *         based on this variable will be incorrect.
  *
  * @note   - The system frequency computed by this function is not the real
  *           frequency in the chip. It is calculated based on the predefined
  *           constant and the selected clock source:
  *
  *           - If SYSCLK source is HSI, SystemCoreClock will contain the
HSI_VALUE(*)
  *
  *           - If SYSCLK source is HSE, SystemCoreClock will contain the
HSE_VALUE(**)
  *
  *           - If SYSCLK source is PLL, SystemCoreClock will contain the
HSE_VALUE(**)
  *             or HSI_VALUE(*) multiplied/divided by the PLL factors.
  *
  *         (*) HSI_VALUE is a constant defined in stm32f4xx_hal_conf.h file
(default value
  *             16 MHz) but the real value may vary depending on the variations
  *             in voltage and temperature.
  *
  *         (**) HSE_VALUE is a constant defined in stm32f4xx_hal_conf.h file
(its value
```

```
  *              depends on the application requirements), user has to ensure
that HSE_VALUE
  *              is same as the real frequency of the crystal used. Otherwise,
this function
  *              may have wrong result.
  *
  *         - The result of this function could be not correct when using
fractional
  *           value for HSE crystal.
  *
  * @param  None
  * @retval None
  */
void SystemCoreClockUpdate(void)
{
  uint32_t tmp = 0, pllvco = 0, pllp = 2, pllsource = 0, pllm = 2;

  /* Get SYSCLK source -------------------------------------------------------*/
  tmp = RCC->CFGR & RCC_CFGR_SWS;

  switch (tmp)
  {
    case 0x00:  /* HSI used as system clock source */
      SystemCoreClock = HSI_VALUE;
      break;
    case 0x04:  /* HSE used as system clock source */
      SystemCoreClock = HSE_VALUE;
      break;
    case 0x08:  /* PLL used as system clock source */

      /* PLL_VCO = (HSE_VALUE or HSI_VALUE / PLL_M) * PLL_N
         SYSCLK = PLL_VCO / PLL_P
         */
      pllsource = (RCC->PLLCFGR & RCC_PLLCFGR_PLLSRC) >> 22;
      pllm = RCC->PLLCFGR & RCC_PLLCFGR_PLLM;

      if (pllsource != 0)
      {
        /* HSE used as PLL clock source */
        pllvco = (HSE_VALUE / pllm) * ((RCC->PLLCFGR & RCC_PLLCFGR_PLLN) >> 6);
      }
      else
      {
        /* HSI used as PLL clock source */
        pllvco = (HSI_VALUE / pllm) * ((RCC->PLLCFGR & RCC_PLLCFGR_PLLN) >> 6);
      }

      pllp = (((RCC->PLLCFGR & RCC_PLLCFGR_PLLP) >>16) + 1 ) *2;
      SystemCoreClock = pllvco/pllp;
      break;
    default:
      SystemCoreClock = HSI_VALUE;
      break;
  }
  /* Compute HCLK frequency --------------------------------------------------*/
  /* Get HCLK prescaler */
  tmp = AHBPrescTable[((RCC->CFGR & RCC_CFGR_HPRE) >> 4)];
  /* HCLK frequency */
  SystemCoreClock >>= tmp;
```

```
}

#if defined (DATA_IN_ExtSRAM) && defined (DATA_IN_ExtSDRAM)
#if defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) ||
defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
/**
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories (SRAM/SDRAM)
  *         This SRAM/SDRAM will be used as program data memory (including heap
and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;

  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;

  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and GPIOI interface clock
*/
  RCC->AHB1ENR |= 0x000001F8;

  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);

  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xAAAA0A8A;
  /* Configure PDx pins speed to 100 MHz */
  GPIOD->OSPEEDR = 0xFFFF0FCF;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOD->PUPDR   = 0x00000000;

  /* Connect PEx pins to FMC Alternate function */
  GPIOE->AFR[0]  = 0xC00CC0CC;
  GPIOE->AFR[1]  = 0xCCCCCCCC;
  /* Configure PEx pins in Alternate function mode */
  GPIOE->MODER   = 0xAAAA828A;
  /* Configure PEx pins speed to 100 MHz */
  GPIOE->OSPEEDR = 0xFFFFC3CF;
  /* Configure PEx pins Output type to push-pull */
  GPIOE->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PEx pins */
  GPIOE->PUPDR   = 0x00000000;

  /* Connect PFx pins to FMC Alternate function */
  GPIOF->AFR[0]  = 0xCCCCCCCC;
  GPIOF->AFR[1]  = 0xCCCCCCCC;
  /* Configure PFx pins in Alternate function mode */
  GPIOF->MODER   = 0xAA800AAA;
  /* Configure PFx pins speed to 50 MHz */
```

```
  GPIOF->OSPEEDR = 0xAA800AAA;
  /* Configure PFx pins Output type to push-pull */
  GPIOF->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PFx pins */
  GPIOF->PUPDR   = 0x00000000;

  /* Connect PGx pins to FMC Alternate function */
  GPIOG->AFR[0]  = 0xCCCCCCCC;
  GPIOG->AFR[1]  = 0xCCCCCCCC;
  /* Configure PGx pins in Alternate function mode */
  GPIOG->MODER   = 0xAAAAAAAA;
  /* Configure PGx pins speed to 50 MHz */
  GPIOG->OSPEEDR = 0xAAAAAAAA;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;

  /* Connect PHx pins to FMC Alternate function */
  GPIOH->AFR[0]  = 0x00C0CC00;
  GPIOH->AFR[1]  = 0xCCCCCCCC;
  /* Configure PHx pins in Alternate function mode */
  GPIOH->MODER   = 0xAAAA08A0;
  /* Configure PHx pins speed to 50 MHz */
  GPIOH->OSPEEDR = 0xAAAA08A0;
  /* Configure PHx pins Output type to push-pull */
  GPIOH->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PHx pins */
  GPIOH->PUPDR   = 0x00000000;

  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;

/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);

  FMC_Bank5_6->SDCR[0] = 0x000019E4;
  FMC_Bank5_6->SDTR[0] = 0x01115351;

  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }
```

```
  /* Delay */
  for (index = 0; index<1000; index++);

  /* PALL command */
  FMC_Bank5_6->SDCMR = 0x00000012;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }

  /* Auto refresh command */
  FMC_Bank5_6->SDCMR = 0x00000073;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }

  /* MRD register program */
  FMC_Bank5_6->SDCMR = 0x00046014;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }

  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));

  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);

#if defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) ||
defined(STM32F439xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx || STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */

  (void)(tmp);
}
#endif /* STM32F427xx || STM32F437xx || STM32F429xx || STM32F439xx ||
STM32F469xx || STM32F479xx */
#elif defined (DATA_IN_ExtSRAM) || defined (DATA_IN_ExtSDRAM)
/**
```

```
  * @brief  Setup the external memory controller.
  *         Called in startup_stm32f4xx.s before jump to main.
  *         This function configures the external memories (SRAM/SDRAM)
  *         This SRAM/SDRAM will be used as program data memory (including heap
and stack).
  * @param  None
  * @retval None
  */
void SystemInit_ExtMemCtl(void)
{
  __IO uint32_t tmp = 0x00;
#if defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) ||
defined(STM32F439xx)\
 || defined(STM32F446xx) || defined(STM32F469xx) || defined(STM32F479xx)
#if defined (DATA_IN_ExtSDRAM)
  register uint32_t tmpreg = 0, timeout = 0xFFFF;
  register __IO uint32_t index;

#if defined(STM32F446xx)
  /* Enable GPIOA, GPIOC, GPIOD, GPIOE, GPIOF, GPIOG interface
      clock */
  RCC->AHB1ENR |= 0x0000007D;
#else
  /* Enable GPIOC, GPIOD, GPIOE, GPIOF, GPIOG, GPIOH and GPIOI interface
      clock */
  RCC->AHB1ENR |= 0x000001F8;
#endif /* STM32F446xx */
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIOCEN);

#if defined(STM32F446xx)
  /* Connect PAx pins to FMC Alternate function */
  GPIOA->AFR[0]  |= 0xC0000000;
  GPIOA->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOA->MODER   |= 0x00008000;
  /* Configure PDx pins speed to 50 MHz */
  GPIOA->OSPEEDR |= 0x00008000;
  /* Configure PDx pins Output type to push-pull */
  GPIOA->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOA->PUPDR   |= 0x00000000;

  /* Connect PCx pins to FMC Alternate function */
  GPIOC->AFR[0]  |= 0x00CC0000;
  GPIOC->AFR[1]  |= 0x00000000;
  /* Configure PDx pins in Alternate function mode */
  GPIOC->MODER   |= 0x00000A00;
  /* Configure PDx pins speed to 50 MHz */
  GPIOC->OSPEEDR |= 0x00000A00;
  /* Configure PDx pins Output type to push-pull */
  GPIOC->OTYPER  |= 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOC->PUPDR   |= 0x00000000;
#endif /* STM32F446xx */

  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x000000CC;
  GPIOD->AFR[1]  = 0xCC000CCC;
```

```
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xA02A000A;
  /* Configure PDx pins speed to 50 MHz */
  GPIOD->OSPEEDR = 0xA02A000A;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOD->PUPDR   = 0x00000000;

  /* Connect PEx pins to FMC Alternate function */
  GPIOE->AFR[0]  = 0xC00000CC;
  GPIOE->AFR[1]  = 0xCCCCCCCC;
  /* Configure PEx pins in Alternate function mode */
  GPIOE->MODER   = 0xAAAA800A;
  /* Configure PEx pins speed to 50 MHz */
  GPIOE->OSPEEDR = 0xAAAA800A;
  /* Configure PEx pins Output type to push-pull */
  GPIOE->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PEx pins */
  GPIOE->PUPDR   = 0x00000000;

  /* Connect PFx pins to FMC Alternate function */
  GPIOF->AFR[0]  = 0xCCCCCCCC;
  GPIOF->AFR[1]  = 0xCCCCCCCC;
  /* Configure PFx pins in Alternate function mode */
  GPIOF->MODER   = 0xAA800AAA;
  /* Configure PFx pins speed to 50 MHz */
  GPIOF->OSPEEDR = 0xAA800AAA;
  /* Configure PFx pins Output type to push-pull */
  GPIOF->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PFx pins */
  GPIOF->PUPDR   = 0x00000000;

  /* Connect PGx pins to FMC Alternate function */
  GPIOG->AFR[0]  = 0xCCCCCCCC;
  GPIOG->AFR[1]  = 0xCCCCCCCC;
  /* Configure PGx pins in Alternate function mode */
  GPIOG->MODER   = 0xAAAAAAAA;
  /* Configure PGx pins speed to 50 MHz */
  GPIOG->OSPEEDR = 0xAAAAAAAA;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;

#if defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) ||
defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx)
  /* Connect PHx pins to FMC Alternate function */
  GPIOH->AFR[0]  = 0x00C0CC00;
  GPIOH->AFR[1]  = 0xCCCCCCCC;
  /* Configure PHx pins in Alternate function mode */
  GPIOH->MODER   = 0xAAAA08A0;
  /* Configure PHx pins speed to 50 MHz */
  GPIOH->OSPEEDR = 0xAAAA08A0;
  /* Configure PHx pins Output type to push-pull */
  GPIOH->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PHx pins */
  GPIOH->PUPDR   = 0x00000000;
```

```
  /* Connect PIx pins to FMC Alternate function */
  GPIOI->AFR[0]  = 0xCCCCCCCC;
  GPIOI->AFR[1]  = 0x00000CC0;
  /* Configure PIx pins in Alternate function mode */
  GPIOI->MODER   = 0x0028AAAA;
  /* Configure PIx pins speed to 50 MHz */
  GPIOI->OSPEEDR = 0x0028AAAA;
  /* Configure PIx pins Output type to push-pull */
  GPIOI->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PIx pins */
  GPIOI->PUPDR   = 0x00000000;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx || STM32F439xx ||
STM32F469xx || STM32F479xx */

/*-- FMC Configuration ------------------------------------------------------*/
  /* Enable the FMC interface clock */
  RCC->AHB3ENR |= 0x00000001;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);

  /* Configure and enable SDRAM bank1 */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCR[0] = 0x00001954;
#else
  FMC_Bank5_6->SDCR[0] = 0x000019E4;
#endif /* STM32F446xx */
  FMC_Bank5_6->SDTR[0] = 0x01115351;

  /* SDRAM initialization sequence */
  /* Clock enable command */
  FMC_Bank5_6->SDCMR = 0x00000011;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }

  /* Delay */
  for (index = 0; index<1000; index++);

  /* PALL command */
  FMC_Bank5_6->SDCMR = 0x00000012;
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }

  /* Auto refresh command */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x000000F3;
#else
  FMC_Bank5_6->SDCMR = 0x00000073;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
```

```
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }

  /* MRD register program */
#if defined(STM32F446xx)
  FMC_Bank5_6->SDCMR = 0x00044014;
#else
  FMC_Bank5_6->SDCMR = 0x00046014;
#endif /* STM32F446xx */
  tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  timeout = 0xFFFF;
  while((tmpreg != 0) && (timeout-- > 0))
  {
    tmpreg = FMC_Bank5_6->SDSR & 0x00000020;
  }

  /* Set refresh count */
  tmpreg = FMC_Bank5_6->SDRTR;
#if defined(STM32F446xx)
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000050C<<1));
#else
  FMC_Bank5_6->SDRTR = (tmpreg | (0x0000027C<<1));
#endif /* STM32F446xx */

  /* Disable write protection */
  tmpreg = FMC_Bank5_6->SDCR[0];
  FMC_Bank5_6->SDCR[0] = (tmpreg & 0xFFFFFDFF);
#endif /* DATA_IN_ExtSDRAM */
#endif /* STM32F427xx || STM32F437xx || STM32F429xx || STM32F439xx ||
STM32F446xx || STM32F469xx || STM32F479xx */

#if defined(STM32F405xx) || defined(STM32F415xx) || defined(STM32F407xx) ||
defined(STM32F417xx)\
 || defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) ||
defined(STM32F439xx)\
 || defined(STM32F469xx) || defined(STM32F479xx) || defined(STM32F412Zx) ||
defined(STM32F412Vx)

#if defined(DATA_IN_ExtSRAM)
/*-- GPIOs Configuration -----------------------------------------------------*/
   /* Enable GPIOD, GPIOE, GPIOF and GPIOG interface clock */
  RCC->AHB1ENR   |= 0x00000078;
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB1ENR, RCC_AHB1ENR_GPIODEN);

  /* Connect PDx pins to FMC Alternate function */
  GPIOD->AFR[0]  = 0x00CCC0CC;
  GPIOD->AFR[1]  = 0xCCCCCCCC;
  /* Configure PDx pins in Alternate function mode */
  GPIOD->MODER   = 0xAAAA0A8A;
  /* Configure PDx pins speed to 100 MHz */
  GPIOD->OSPEEDR = 0xFFFF0FCF;
  /* Configure PDx pins Output type to push-pull */
  GPIOD->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PDx pins */
  GPIOD->PUPDR   = 0x00000000;

  /* Connect PEx pins to FMC Alternate function */
```

```
  GPIOE->AFR[0]  = 0xC00CC0CC;
  GPIOE->AFR[1]  = 0xCCCCCCCC;
  /* Configure PEx pins in Alternate function mode */
  GPIOE->MODER   = 0xAAAA828A;
  /* Configure PEx pins speed to 100 MHz */
  GPIOE->OSPEEDR = 0xFFFFC3CF;
  /* Configure PEx pins Output type to push-pull */
  GPIOE->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PEx pins */
  GPIOE->PUPDR   = 0x00000000;

  /* Connect PFx pins to FMC Alternate function */
  GPIOF->AFR[0]  = 0x00CCCCCC;
  GPIOF->AFR[1]  = 0xCCCC0000;
  /* Configure PFx pins in Alternate function mode */
  GPIOF->MODER   = 0xAA000AAA;
  /* Configure PFx pins speed to 100 MHz */
  GPIOF->OSPEEDR = 0xFF000FFF;
  /* Configure PFx pins Output type to push-pull */
  GPIOF->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PFx pins */
  GPIOF->PUPDR   = 0x00000000;

  /* Connect PGx pins to FMC Alternate function */
  GPIOG->AFR[0]  = 0x00CCCCCC;
  GPIOG->AFR[1]  = 0x000000C0;
  /* Configure PGx pins in Alternate function mode */
  GPIOG->MODER   = 0x00085AAA;
  /* Configure PGx pins speed to 100 MHz */
  GPIOG->OSPEEDR = 0x000CAFFF;
  /* Configure PGx pins Output type to push-pull */
  GPIOG->OTYPER  = 0x00000000;
  /* No pull-up, pull-down for PGx pins */
  GPIOG->PUPDR   = 0x00000000;

/*-- FMC/FSMC Configuration ------------------------------------------------*/
  /* Enable the FMC/FSMC interface clock */
  RCC->AHB3ENR         |= 0x00000001;

#if defined(STM32F427xx) || defined(STM32F437xx) || defined(STM32F429xx) ||
defined(STM32F439xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001011;
  FMC_Bank1->BTCR[3]  = 0x00000201;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F427xx || STM32F437xx || STM32F429xx || STM32F439xx */
#if defined(STM32F469xx) || defined(STM32F479xx)
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FMC_Bank1->BTCR[2]  = 0x00001091;
  FMC_Bank1->BTCR[3]  = 0x00110212;
  FMC_Bank1E->BWTR[2] = 0x0fffffff;
#endif /* STM32F469xx || STM32F479xx */
#if defined(STM32F405xx) || defined(STM32F415xx) || defined(STM32F407xx)||
defined(STM32F417xx)\
   || defined(STM32F412Zx) || defined(STM32F412Vx)
```

```
  /* Delay after an RCC peripheral clock enabling */
  tmp = READ_BIT(RCC->AHB3ENR, RCC_AHB3ENR_FSMCEN);
  /* Configure and enable Bank1_SRAM2 */
  FSMC_Bank1->BTCR[2]  = 0x00001011;
  FSMC_Bank1->BTCR[3]  = 0x00000201;
  FSMC_Bank1E->BWTR[2] = 0x0FFFFFFF;
#endif /* STM32F405xx || STM32F415xx || STM32F407xx || STM32F417xx ||
STM32F412Zx || STM32F412Vx */

#endif /* DATA_IN_ExtSRAM */
#endif /* STM32F405xx || STM32F415xx || STM32F407xx || STM32F417xx ||
STM32F427xx || STM32F437xx ||\
          STM32F429xx || STM32F439xx || STM32F469xx || STM32F479xx ||
STM32F412Zx || STM32F412Vx  */
  (void)(tmp);
}
#endif /* DATA_IN_ExtSRAM && DATA_IN_ExtSDRAM */
/**
  * @}
  */

/**
  * @}
  */

/**
  * @}
  */
```