```
/* USER CODE BEGIN Header */
/**
  ******************************************************************************
  * @file           : main.c
  * @brief          : Main program body
  ******************************************************************************
  * @attention
  *
  * Copyright (c) 2024 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 "i2c.h"
#include "usart.h"
#include "gpio.h"

/* Private includes ----------------------------------------------------------*/
/* USER CODE BEGIN Includes */
#include <stdio.h>
```

```
#include <string.h>
/* USER CODE END Includes */

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

/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define LPS25HB_ADDRESS 0x5D
#define LPS25HB_WHO_AM_I 0x0F

#define LPS25HB_CTRL_REG1 0x20 //Control register 1
#define LPS25HB_CTRL_REG2 0x21 //Control register 2
#define LPS25HB_CTRL_REG3 0x22 //Control register 3 (interrupt control)
#define LPS25HB_CTRL_REG4 0x23 //Control register 4 (interrupt configuration)
#define LPS25HB_STATUS_REG 0x27 //Status register
#define LPS25HB_FIFO_CTRL 0x2E //FIFO control F_MODE[2:0] WTM_POINT[4:0]
#define LPS25HB_PRESS_OUT_XL 0x28 //Pressure output value (LSB)
#define LPS25HB_PRESS_OUT_L 0x29 //Pressure output value (mid part)
#define LPS25HB_PRESS_OUT_H 0x2A //Pressure output value (MSB)
#define LPS25HB_TEMP_OUT_L 0x2B //Temperature output value (LSB)
#define LPS25HB_TEMP_OUT_H 0x2C //Temperature output value (MSB)
/* USER CODE END PD */

/* Private macro -------------------------------------------------------------*/
```

```
/* USER CODE BEGIN PM */

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/

/* USER CODE BEGIN PV */
char usartWiadWych[] = "\0";
uint8_t i2cMemRead[4] = {0, 0, 0, 0};
uint8_t i2cMemWrite[4] = {0, 0, 0, 0};
uint8_t lps25hbPressOut[3] = {0, 0, 0};
uint8_t lps25hbTempOut[2] = {0, 0};
uint8_t userButtonFlag = 0;
uint8_t LPS25HB_bootFlag = 0;
uint8_t LPS25HB_powerDownFlag = 0;
/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
/* USER CODE BEGIN PFP */
int __io_putchar(int ch){
	HAL_UART_Transmit(&huart2, (uint8_t*)&ch, 1, HAL_MAX_DELAY);
	return ch;
}

void HAL_GPIO_EXTI_Callback(uint16_t GPIO_Pin)
{
```

```
  if(GPIO_Pin == GPIO_PIN_13) userButtonFlag = 1;
}
/* USER CODE END PFP */

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

/* USER CODE END 0 */

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

  /* USER CODE BEGIN 1 */

  /* USER CODE END 1 */

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

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

  /* USER CODE BEGIN Init */
```

```
  /* USER CODE END Init */

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

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */

  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_I2C1_Init();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */
  HAL_I2C_Mem_Read(&hi2c1, LPS25HB_ADDRESS<<1, LPS25HB_WHO_AM_I, 1,
(uint8_t*)i2cMemRead, 1, HAL_MAX_DELAY);
  printf("WHO AM I: %hx\r\n", (unsigned short int)i2cMemRead[0]);
  HAL_I2C_Mem_Read(&hi2c1, LPS25HB_ADDRESS<<1, LPS25HB_FIFO_CTRL, 1,
(uint8_t*)i2cMemRead, 1, HAL_MAX_DELAY);
  printf("FIFO CONTROL: %hx\r\n", (unsigned short int)i2cMemRead[0]);
  HAL_I2C_Mem_Read(&hi2c1, LPS25HB_ADDRESS<<1, LPS25HB_CTRL_REG1, 1,
(uint8_t*)i2cMemRead, 1, HAL_MAX_DELAY);
  printf("CONTROL REGISTER 1: %hx\r\n", (unsigned short int)i2cMemRead[0]);
  HAL_I2C_Mem_Read(&hi2c1, LPS25HB_ADDRESS<<1, LPS25HB_CTRL_REG2, 1,
(uint8_t*)i2cMemRead, 1, HAL_MAX_DELAY);
  printf("CONTROL REGISTER 2: %hx\r\n", (unsigned short int)i2cMemRead[0]);
  HAL_I2C_Mem_Read(&hi2c1, LPS25HB_ADDRESS<<1, LPS25HB_CTRL_REG3, 1,
(uint8_t*)i2cMemRead, 1, HAL_MAX_DELAY);
  printf("CONTROL REGISTER 3: %hx\r\n", (unsigned short int)i2cMemRead[0]);
```

```
 HAL_I2C_Mem_Read(&hi2c1, LPS25HB_ADDRESS<<1, LPS25HB_STATUS_REG, 1,
(uint8_t*)i2cMemRead, 1, HAL_MAX_DELAY);

 printf("STATUS REGISTER: %hx\r\n", (unsigned short int)i2cMemRead[0]);


 i2cMemWrite[0] = 0x80;

 printf("WRITE TO CONTROL REGISTER 2: %hx\r\n", i2cMemWrite[0]);

 HAL_I2C_Mem_Write(&hi2c1, LPS25HB_ADDRESS<<1, LPS25HB_CTRL_REG2, 1,
(uint8_t*)i2cMemWrite, 1, HAL_MAX_DELAY);

 do{

        HAL_Delay(200);

        HAL_I2C_Mem_Read(&hi2c1, LPS25HB_ADDRESS<<1, LPS25HB_CTRL_REG2, 1,
(uint8_t*)i2cMemRead, 1, HAL_MAX_DELAY);

        printf("CONTROL REGISTER 2: %hx\r\n", (unsigned short int)i2cMemRead[0]);

        if(i2cMemRead[0] == 0x80) LPS25HB_bootFlag = 1;

        else LPS25HB_bootFlag = 0;

 }while(LPS25HB_bootFlag);


 i2cMemWrite[0] = 0x80;

 printf("WRITE TO CONTROL REGISTER 1: %hx\r\n", i2cMemWrite[0]);

 HAL_I2C_Mem_Write(&hi2c1, LPS25HB_ADDRESS<<1, LPS25HB_CTRL_REG1, 1,
(uint8_t*)i2cMemWrite, 1, HAL_MAX_DELAY);

 do{

        HAL_Delay(200);

        HAL_I2C_Mem_Read(&hi2c1, LPS25HB_ADDRESS<<1, LPS25HB_CTRL_REG1, 1,
(uint8_t*)i2cMemRead, 1, HAL_MAX_DELAY);

        printf("CONTROL REGISTER 1: %hx\r\n", (unsigned short int)i2cMemRead[0]);

        if(i2cMemRead[0] == 0x80) LPS25HB_powerDownFlag = 0;

        else LPS25HB_powerDownFlag = 1;

 }while(LPS25HB_powerDownFlag);
```

```
 /* USER CODE END 2 */


 /* Infinite loop */
 /* USER CODE BEGIN WHILE */
 while (1)
 {
        if(userButtonFlag == 1){
                userButtonFlag = 0;
                HAL_GPIO_TogglePin(USER_LED_GPIO_Port, USER_LED_Pin);
                i2cMemWrite[0] = 0x01;
                printf("WRITE TO CONTROL REGISTER 2: %hx\r\n", i2cMemWrite[0]);
                HAL_I2C_Mem_Write(&hi2c1, LPS25HB_ADDRESS<<1, LPS25HB_CTRL_REG2, 1,
(uint8_t*)i2cMemWrite, 1, HAL_MAX_DELAY);


                do{
                HAL_I2C_Mem_Read(&hi2c1, LPS25HB_ADDRESS<<1, LPS25HB_CTRL_REG2, 1,
(uint8_t*)i2cMemRead, 1, HAL_MAX_DELAY);
                printf("CONTROL REGISTER 2: %hx\r\n", (unsigned short int)i2cMemRead[0]);
                HAL_Delay(100);
                }while(i2cMemRead[0] & 0x01);


                HAL_I2C_Mem_Read(&hi2c1, LPS25HB_ADDRESS<<1, LPS25HB_STATUS_REG, 1,
(uint8_t*)i2cMemRead, 1, HAL_MAX_DELAY);
                printf("STATUS REGISTER: %hx\r\n", (unsigned short int)i2cMemRead[0]);


                HAL_I2C_Mem_Read(&hi2c1, LPS25HB_ADDRESS<<1,
LPS25HB_TEMP_OUT_L|0x80, I2C_MEMADD_SIZE_8BIT, (uint8_t*)lps25hbTempOut, 2,
HAL_MAX_DELAY);
```

```
                printf("TEMP OUT LOW: %hx\r\nTEMP OUT HIGH: %hx\r\n", (unsigned short int)lps25hbTempOut[0], (unsigned short int)lps25hbTempOut[1]);

                HAL_I2C_Mem_Read(&hi2c1, LPS25HB_ADDRESS<<1, LPS25HB_PRESS_OUT_XL|0x80, I2C_MEMADD_SIZE_8BIT, (uint8_t*)lps25hbPressOut, 3, HAL_MAX_DELAY);

                printf("PRESSURE LSB: %hx\r\nPRESSURE MID: %hx\r\nPRESSURE MSB: %hx\r\n", (unsigned short int)lps25hbPressOut[0], (unsigned short int)lps25hbPressOut[1], (unsigned short int)lps25hbPressOut[2]);

            }

   /* USER CODE END WHILE */


   /* USER CODE BEGIN 3 */

  }

  /* USER CODE END 3 */

}


/**

  * @brief System Clock Configuration

  * @retval None

  */

void SystemClock_Config(void)

{

  RCC_OscInitTypeDef RCC_OscInitStruct = {0};

  RCC_ClkInitTypeDef RCC_ClkInitStruct = {0};


  /** Initializes the RCC Oscillators according to the specified parameters

  * in the RCC_OscInitTypeDef structure.

  */

  RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
```

```
  RCC_OscInitStruct.HSIState = RCC_HSI_ON;
  RCC_OscInitStruct.HSICalibrationValue = RCC_HSICALIBRATION_DEFAULT;
  RCC_OscInitStruct.PLL.PLLState = RCC_PLL_NONE;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_HSI;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV1;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV1;

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

/* 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 */
```