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

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

/* Private typedef -----------------------------------------------------------*/
/* USER CODE BEGIN PTD */
typedef struct {
    int16_t x, y, z;
}LIS3DSH_Data;
/* USER CODE END PTD */

/* Private define ------------------------------------------------------------*/
/* USER CODE BEGIN PD */
#define BV(n) (1<<(n))

#define LIS3DSH_CONTROL_REG4_ADDR   0x20
#define LIS3DSH_STATUS_REG_ADDR     0x27
#define LIS3DSH_OUT_XL_ADDR         0x28
#define LIS3DSH_OUT_YL_ADDR         0x2A
#define LIS3DSH_OUT_ZL_ADDR         0x2C

#define LIS3DSH_CR4_ODR_OFF         (0x00 << 4)
#define LIS3DSH_CR4_ODR_25HZ        (0x04 << 4)
#define LIS3DSH_CR4_XEN             (BV(0))
#define LIS3DSH_CR4_YEN             (BV(1))
#define LIS3DSH_CR4_ZEN             (BV(2))

#define LIS3DSH_SR_XDA              (BV(0))
#define LIS3DSH_SR_YDA              (BV(1))
#define LIS3DSH_SR_ZDA              (BV(2))
#define LIS3DSH_SR_ZYXDA            (BV(3))
#define LIS3DSH_SR_XYZ_MSK          (LIS3DSH_SR_XDA | LIS3DSH_SR_YDA | LIS3DSH_SR_ZDA)

/* USER CODE END PD */

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

/* USER CODE END PM */

/* Private variables ---------------------------------------------------------*/
SPI_HandleTypeDef hspi1;
```

```
UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */

/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_SPI1_Init(void);
static void MX_USART2_UART_Init(void);
/* USER CODE BEGIN PFP */

/* USER CODE END PFP */

/* Private user code ---------------------------------------------------------*/
/* USER CODE BEGIN 0 */
void SPI_Write(uint8_t reg, uint8_t *data, uint8_t size) {
    // Enable CS (PE3=0)
    HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_RESET);
    // Send register (internal) address
    HAL_SPI_Transmit(&hspi1, &reg, 1, HAL_MAX_DELAY);
    // Send data of given size
    HAL_SPI_Transmit(&hspi1, data, size, HAL_MAX_DELAY);
    // Disable CS (PE3=1)
    HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_SET);
}

void SPI_Read(uint8_t reg, uint8_t *data, int size) {
    // Enable CS (PE3=0)
    HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_RESET);
    // Send register (internal) address (msb=1 for read op with LIS3DSH)
    reg |= BV(7);
    HAL_SPI_Transmit(&hspi1, &reg, 1, HAL_MAX_DELAY);
    // Receive data of given size
    HAL_SPI_Receive(&hspi1, data, size, HAL_MAX_DELAY);
    // Disable CS (PE3=1)
    HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_SET);
}

void LIS3DSH_Init(void) {
    // cr4 = odr = 25 hz, xen, yen, zen
    uint8_t val = LIS3DSH_CR4_ODR_25HZ | LIS3DSH_CR4_XEN | LIS3DSH_CR4_YEN |
LIS3DSH_CR4_ZEN;
    SPI_Write(LIS3DSH_CONTROL_REG4_ADDR, &val, 1);
}

void LIS3DSH_WaitForDataAvail(void) {
    uint8_t val;
    // wait while status regr xda, yda and zda bits are zero
    do {
        SPI_Read(LIS3DSH_STATUS_REG_ADDR, &val, 1);
    } while( (val & LIS3DSH_SR_XYZ_MSK) == 0);
}

void LIS3DSH_ReadData(LIS3DSH_Data *data) {
    uint8_t buf[2];
    // read xl and xh readings and convert to x 16-bit reading
    SPI_Read(LIS3DSH_OUT_XL_ADDR, buf, 2);
    data->x = (((uint16_t)buf[1]) << 8) | buf[0];

    // read yl and yh readings and convert to y 16-bit reading
    SPI_Read(LIS3DSH_OUT_YL_ADDR, buf, 2);
    data->y = (((uint16_t)buf[1]) << 8) | buf[0];

    // read zl and zh readings and convert to z 16-bit reading
```

```
    SPI_Read(LIS3DSH_OUT_ZL_ADDR, buf, 2);
    data->z = (((uint16_t)buf[1]) << 8) | buf[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_SPI1_Init();
  MX_USART2_UART_Init();
  /* USER CODE BEGIN 2 */
  LIS3DSH_Init(); // initialize LIS3DSH
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
      char str[32];
      LIS3DSH_Data data;
      LIS3DSH_WaitForDataAvail(); // wait for x, y or z reading to change
      LIS3DSH_ReadData(&data); // get accel reading
      sprintf(str, "x=%d, y=%d, z=%d\r\n", data.x, data.y, data.z); // convert reading
into string
      HAL_UART_Transmit(&huart2, (uint8_t*)str, strlen(str), HAL_MAX_DELAY); // send it
to uart
      HAL_Delay(1000);
    /* 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};
```

```c
  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 = 50;
  RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV4;
  RCC_OscInitStruct.PLL.PLLQ = 7;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK)
  {
    Error_Handler();
  }

  /** Initializes the CPU, AHB and APB buses clocks
  */
  RCC_ClkInitStruct.ClockType = RCC_CLOCKTYPE_HCLK|RCC_CLOCKTYPE_SYSCLK
                              |RCC_CLOCKTYPE_PCLK1|RCC_CLOCKTYPE_PCLK2;
  RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
  RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
  RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
  RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;

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

/**
  * @brief SPI1 Initialization Function
  * @param None
  * @retval None
  */
static void MX_SPI1_Init(void)
{

  /* USER CODE BEGIN SPI1_Init 0 */

  /* USER CODE END SPI1_Init 0 */

  /* USER CODE BEGIN SPI1_Init 1 */

  /* USER CODE END SPI1_Init 1 */
  /* SPI1 parameter configuration*/
  hspi1.Instance = SPI1;
  hspi1.Init.Mode = SPI_MODE_MASTER;
  hspi1.Init.Direction = SPI_DIRECTION_2LINES;
  hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
  hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
  hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
  hspi1.Init.NSS = SPI_NSS_SOFT;
  hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_8;
  hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
  hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
  hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
```

```
  hspi1.Init.CRCPolynomial = 10;
  if (HAL_SPI_Init(&hspi1) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN SPI1_Init 2 */

  /* USER CODE END SPI1_Init 2 */

}

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

  /* USER CODE BEGIN USART2_Init 0 */

  /* USER CODE END USART2_Init 0 */

  /* USER CODE BEGIN USART2_Init 1 */

  /* USER CODE END USART2_Init 1 */
  huart2.Instance = USART2;
  huart2.Init.BaudRate = 9600;
  huart2.Init.WordLength = UART_WORDLENGTH_8B;
  huart2.Init.StopBits = UART_STOPBITS_1;
  huart2.Init.Parity = UART_PARITY_NONE;
  huart2.Init.Mode = UART_MODE_TX_RX;
  huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
  huart2.Init.OverSampling = UART_OVERSAMPLING_16;
  if (HAL_UART_Init(&huart2) != HAL_OK)
  {
    Error_Handler();
  }
  /* USER CODE BEGIN USART2_Init 2 */

  /* USER CODE END USART2_Init 2 */

}

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

  /* GPIO Ports Clock Enable */
  __HAL_RCC_GPIOE_CLK_ENABLE();
  __HAL_RCC_GPIOA_CLK_ENABLE();

  /*Configure GPIO pin Output Level */
  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_RESET);

  /*Configure GPIO pin : PE3 */
  GPIO_InitStruct.Pin = GPIO_PIN_3;
  GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
  GPIO_InitStruct.Pull = GPIO_NOPULL;
```

```
  GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
  HAL_GPIO_Init(GPIOE, &GPIO_InitStruct);

/* USER CODE BEGIN MX_GPIO_Init_2 */
/* USER CODE END MX_GPIO_Init_2 */
}

/* USER CODE BEGIN 4 */

/* USER CODE END 4 */

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

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