

Accelerometer – Source code detail.

1. Including files and Initialize all components.

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

/* USER CODE BEGIN Includes */

/* USER CODE END Includes */

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

UART_HandleTypeDef huart2;

/* USER CODE BEGIN PV */
/* Private variables -----*/
/* USER CODE END PV */
/* Private function prototypes -----*/
void SystemClock_Config(void);
void Error_Handler(void);
static void MX_GPIO_Init(void);
static void MX_SPI1_Init(void);
static void MX_USART2_UART_Init(void);

/* USER CODE BEGIN PFP */
/* Private function prototypes -----*/
/* USER CODE END PFP */
/* USER CODE BEGIN 0 */
/* USER CODE END 0 */

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();
    /* Configure the system clock */
    SystemClock_Config();
    /* Initialize all configured peripherals */
    MX_GPIO_Init();
    MX_SPI1_Init();
    MX_USART2_UART_Init();
    /* USER CODE BEGIN 2 */
```

2. SPI Transmission

2.1 Variable (address and control value of SPI transmission)

```
uint8_t addr;  
uint8_t data;  
int8_t x,y,z = 0;
```

2.2 Starting SPI transmission to send control register

```
//start with CS is low  
HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_RESET);  
  
addr = 0x20; // control register 1  
HAL_SPI_Transmit(&hspi1,&addr,1,50);  
  
data = 0x47; // 0100 0111 that means active X Y Z and full scale is +-2.0g  
HAL_SPI_Transmit(&hspi1,&data,1,50);  
  
HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_SET);  
/* USER CODE END 2 */  
/* Infinite loop */  
/* USER CODE BEGIN WHILE */  
while (1)  
{  
    /* USER CODE END WHILE */  
    /* USER CODE BEGIN 3 */
```

2.3 Starting SPI transmission to get x y z axis values

```
HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_RESET);  
// OUT_X  
addr = 0x29 | 0x80;  
HAL_SPI_Transmit(&hspi1,&addr,1,50);  
HAL_SPI_Receive(&hspi1,&x,1,50);  
// OUT_Y  
addr = 0x2B | 0x80;  
HAL_SPI_Transmit(&hspi1,&addr,1,50);  
HAL_SPI_Receive(&hspi1,&y,1,50);  
// OUT_Z  
addr = 0x2D | 0x80;  
HAL_SPI_Transmit(&hspi1,&addr,1,50);  
HAL_SPI_Receive(&hspi1,&z,1,50);
```

2.4 Show how the STM32F4 board moves by LED

```
//-127 to 127
```

2.4.1 If board is upside down, all LED will be toggled

```
if (z <= -30) {  
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);  
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET);  
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET);  
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_SET);  
    HAL_Delay(500);  
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12,GPIO_PIN_RESET);  
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13,GPIO_PIN_RESET);  
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14,GPIO_PIN_RESET);
```

```

        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15,GPIO_PIN_RESET);
        HAL_Delay(10);
    } else {
2.4.2 If board is rotated, LED will be lighted up
        //LEFT
        if (x <= -15)
            HAL_GPIO_WritePin(GPIOD,GPIO_PIN_12,GPIO_PIN_SET);
        else
            HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12,GPIO_PIN_RESET);
        //RIGHT
        if (x >= 15)
            HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET);
        else
            HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);
        //UP
        if (y >= 15)
            HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET);
        else
            HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_RESET);
        //DOWN
        if (y <= -15)
            HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_SET);
        else
            HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_RESET);
    }
    HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_SET);

```

3. UART Transmission for x y z value

```

    char uart_output[60];
    int uart_output_size;
    uart_output_size = sprintf(uart_output,"accelerometer result x = %d :
y = %d : z = %d \n\r",x,y,z);
    HAL_UART_Transmit(&huart2,uart_output, uart_output_size, 100000);

    HAL_Delay(500);
}
/* USER CODE END 3 */
}

```

4. Other configuration

```

/** System Clock Configuration
*/
void SystemClock_Config(void)
{
    RCC_OscInitTypeDef RCC_OscInitStruct;
    RCC_ClkInitTypeDef RCC_ClkInitStruct;

    /**Configure the main internal regulator output voltage */
    __HAL_RCC_PWR_CLK_ENABLE();
    __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE1);

    /**Initializes the CPU, AHB and APB busses clocks */

```

```

RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSI;
RCC_OscInitStruct.HSIState = RCC_HSI_ON;
RCC_OscInitStruct.HSICalibrationValue = 16;
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 busses 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();
}

/**Configure the SysTick interrupt time */
HAL_SYSTICK_Config(HAL_RCC_GetHCLKFreq()/1000);

/**Configure the SysTick */
HAL_SYSTICK_CLKSourceConfig(SYSTICK_CLKSOURCE_HCLK);

/* SysTick_IRQn interrupt configuration */
HAL_NVIC_SetPriority(SysTick_IRQn, 0, 0);
}

/* SPI1 init function */
static void MX_SPI1_Init(void)
{
    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_2;
    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();
    }
}

```

```

    }

}

/* USART2 init function */
static void MX_USART2_UART_Init(void)
{
    huart2.Instance = USART2;
    huart2.Init.BaudRate = 115200;
    huart2.Init.WordLength = UART_WORDLENGTH_8B;
    huart2.Init.StopBits = UART_STOPBITS_1;
    huart2.Init.Parity = UART_PARITY_NONE;
    huart2.Init.Mode = UART_MODE_TX_RX;
    huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
    huart2.Init.OverSampling = UART_OVERSAMPLING_16;
    if (HAL_UART_Init(&huart2) != HAL_OK)
    {
        Error_Handler();
    }
}

/** Configure pins as
    * Analog
    * Input
    * Output
    * EVENT_OUT
    * EXTI
    PC3  -----> I2S2_SD
    PA4  -----> I2S3_WS
    PB10 -----> I2S2_CK
    PC7  -----> I2S3_MCK
    PA9  -----> USB_OTG_FS_VBUS
    PA10 -----> USB_OTG_FS_ID
    PA11 -----> USB_OTG_FS_DM
    PA12 -----> USB_OTG_FS_DP
    PC10 -----> I2S3_CK
    PC12 -----> I2S3_SD
    PB6  -----> I2C1_SCL
    PB9  -----> I2C1_SDA
*/
static void MX_GPIO_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct;

    /* GPIO Ports Clock Enable */
    __HAL_RCC_GPIOE_CLK_ENABLE();
    __HAL_RCC_GPIOC_CLK_ENABLE();
    __HAL_RCC_GPIOH_CLK_ENABLE();
    __HAL_RCC_GPIOA_CLK_ENABLE();
    __HAL_RCC_GPIOB_CLK_ENABLE();
    __HAL_RCC_GPIOD_CLK_ENABLE();

    /*Configure GPIO pin Output Level */
    HAL_GPIO_WritePin(CS_I2C_SPI_GPIO_Port, CS_I2C_SPI_Pin, GPIO_PIN_RESET);

```

```

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(OTG_FS_PowerSwitchOn_GPIO_Port, OTG_FS_PowerSwitchOn_Pin,
GPIO_PIN_SET);

/*Configure GPIO pin Output Level */
HAL_GPIO_WritePin(GPIOD, LD4_Pin|LD3_Pin|LD5_Pin|LD6_Pin
                  |Audio_RST_Pin, GPIO_PIN_RESET);

/*Configure GPIO pin : CS_I2C_SPI_Pin */
GPIO_InitStruct.Pin = CS_I2C_SPI_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(CS_I2C_SPI_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pin : OTG_FS_PowerSwitchOn_Pin */
GPIO_InitStruct.Pin = OTG_FS_PowerSwitchOn_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(OTG_FS_PowerSwitchOn_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pin : PDM_OUT_Pin */
GPIO_InitStruct.Pin = PDM_OUT_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.Alternate = GPIO_AF5_SPI2;
HAL_GPIO_Init(PDM_OUT_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pin : PA4 */
GPIO_InitStruct.Pin = GPIO_PIN_4;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.Alternate = GPIO_AF6_SPI3;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/*Configure GPIO pin : BOOT1_Pin */
GPIO_InitStruct.Pin = BOOT1_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(BOOT1_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pin : CLK_IN_Pin */
GPIO_InitStruct.Pin = CLK_IN_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.Alternate = GPIO_AF5_SPI2;
HAL_GPIO_Init(CLK_IN_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pins : LD4_Pin LD3_Pin LD5_Pin LD6_Pin Audio_RST_Pin */
GPIO_InitStruct.Pin = LD4_Pin|LD3_Pin|LD5_Pin|LD6_Pin |Audio_RST_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;

```

```

GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
HAL_GPIO_Init(GPIOD, &GPIO_InitStruct);

/*Configure GPIO pins : PC7 I2S3_SCK_Pin PC12 */
GPIO_InitStruct.Pin = GPIO_PIN_7|I2S3_SCK_Pin|GPIO_PIN_12;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.Alternate = GPIO_AF6_SPI3;
HAL_GPIO_Init(GPIOC, &GPIO_InitStruct);

/*Configure GPIO pin : VBUS_FS_Pin */
GPIO_InitStruct.Pin = VBUS_FS_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(VBUS_FS_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pins : OTG_FS_ID_Pin OTG_FS_DM_Pin OTG_FS_DP_Pin */
GPIO_InitStruct.Pin = OTG_FS_ID_Pin|OTG_FS_DM_Pin|OTG_FS_DP_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
GPIO_InitStruct.Pull = GPIO_NOPULL;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.Alternate = GPIO_AF10_OTG_FS;
HAL_GPIO_Init(GPIOA, &GPIO_InitStruct);

/*Configure GPIO pin : OTG_FS_OverCurrent_Pin */
GPIO_InitStruct.Pin = OTG_FS_OverCurrent_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(OTG_FS_OverCurrent_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pins : Audio_SCL_Pin Audio_SDA_Pin */
GPIO_InitStruct.Pin = Audio_SCL_Pin|Audio_SDA_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_AF_OD;
GPIO_InitStruct.Pull = GPIO_PULLUP;
GPIO_InitStruct.Speed = GPIO_SPEED_FREQ_LOW;
GPIO_InitStruct.Alternate = GPIO_AF4_I2C1;
HAL_GPIO_Init(GPIOB, &GPIO_InitStruct);

/*Configure GPIO pin : MEMS_INT1_Pin */
GPIO_InitStruct.Pin = MEMS_INT1_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_IT_RISING;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(MEMS_INT1_GPIO_Port, &GPIO_InitStruct);

/*Configure GPIO pin : MEMS_INT2_Pin */
GPIO_InitStruct.Pin = MEMS_INT2_Pin;
GPIO_InitStruct.Mode = GPIO_MODE_EVT_RISING;
GPIO_InitStruct.Pull = GPIO_NOPULL;
HAL_GPIO_Init(MEMS_INT2_GPIO_Port, &GPIO_InitStruct);

}

/* USER CODE BEGIN 4 */
/* USER CODE END 4 */

/**

```

```

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

#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

/***** (C) COPYRIGHT STMicroelectronics *****/
FILE*****/

```