Hardware Synthesis Laboratory

Final Project Report

Kosate Limpongsa — 5731012721

---

2. Source Code & Video

2.1 LIS302DL — Accelerometers

Source Code : https://github.com/neungkl/STM32F4-hardware-project/tree/master/task1
Video : https://www.youtube.com/watch?v=X1yB6p0slpo&index=1&list=PLFt9kotlOsQLRj9_FBZgsDHt8URf2Fik9
Or file names “task1” under the “source_code” directory.

```
uint8_t address = 0x20;
uint8_t data = 0x67;

uint8_t x, y, z;

char debugTxt[50];

HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_RESET);
HAL_SPI_Transmit(&hspi1, &address, 1, 50);
HAL_SPI_Transmit(&hspi1, &data, 1, 50);
HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_SET);
```

The Initialize part is transmits controls setting by passing address 0x20 and value with 0x67 for enable power mode and X, Y, Z preparation.

You must toggle PE3 between reset and switch back to set while transmits the data because switching the interface mode to I2S.

```
address = 0x29 | 0x80;
HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_RESET);
HAL_SPI_Transmit(&hspi1, &address, 1, 50);
HAL_SPI_Receive(&hspi1, &x, 1, 50);
HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_SET);

address = 0x2B | 0x80;
HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_RESET);
HAL_SPI_Transmit(&hspi1, &address, 1, 50);
HAL_SPI_Receive(&hspi1, &y, 1, 50);
HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_SET);

address = 0x2C | 0x80;
HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_RESET);
HAL_SPI_Transmit(&hspi1, &address, 1, 50);
HAL_SPI_Receive(&hspi1, &z, 1, 50);
HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_SET);
```

```
if (x < 255 - rotateThreshold && x > 128) {
        // Left On
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);
} else if (x <= 128 && x > rotateThreshold) {
        // Right On
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET);
} else {
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_RESET);
}

if (y > rotateThreshold && y < 128) {
        // Up On
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_RESET);
} else if (y >= 128 && y < 255 - rotateThreshold) {
        // Down On
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_SET);
} else {
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_RESET);
}
```

Read the three axis data by transmit the address 0x29, 0x2B, 0x2C

From the X,Y,Z value. You get the value with 8 bits (0 - 255) and give some logic to determine the oblique of board. Then, show the LED light depends on board rotation.

And transmit debug text via UART API.

2.2 MP45DT02 — Microphone

Source Code : https://github.com/neungkl/STM32F4-hardware-project/tree/master/task2

Video : https://www.youtube.com/watch?v=GVRDzz0FhP0&list=PLFt9kotlOsQLRj9_FBZgsDHt8URf2Fik9&index=2

Or file names “task2” under the “source_code” directory.

First, receive the 16 bytes PDM value from HAL_I2S_Recieve

```
pdm_count = 0;

for (i = 0; i < PDM_BUFFER_SIZE; i++) {
        bit_position = (1 << 15);
        for (j = 0; j < 16; j++) {
                if (bit_position & buffer[i]) {
                        pdm_count++;
                }
                bit_position >>= 1;
        }
}

if (pdm_count > 100) {
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);
        pdm_count = 0;
        HAL_Delay(500);
} else {
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_RESET);
}
```

The algorithm were use in this project is counting the 1. Start from 16 bytes value that equal to 256 bits. Then, counting the number 1 that appear on 256 bits.

```
cur_time = HAL_GetTick();
if (cur_time - start_time > 100) {
        start_time = cur_time;
        sprintf(display_buffer, "Loud : %d\n", pdm_count);
        HAL_UART_Transmit(
                &huart2,
                (uint8_t*) display_buffer,
                strlen(display_buffer),
                100
        );
}
```

Summarize the value and set the threshold. If summarize value exceed 100, means the sound is too loud. But if value lower than 100, that means the sound is silent.

Ticking the time every 100 ms for sending the logs message via UART.

2.3. CS43L22 — Speaker

Source Code : https://github.com/neungkl/STM32F4-hardware-project/tree/master/task3

Video : https://www.youtube.com/watch?v=Aepxf0yS_kM&index=3&list=PLFt9kotlOsQLRj9_FBZgsDHt8URf2Fik9

Or file names “task3” under the “source_code” directory.

```
int Transmit_Audio_Data(uint8_t address,uint8_t data){
        sound_send_data[0] = address;
        sound_send_data[1] = data;
        return HAL_I2C_Master_Transmit(&hi2c1,0x94,sound_send_data,2,50);
}
```

Transmit_Audio_Data is function for transmit the pairs of sound data using I2C interface.

```
uint8_t Mapping_Choord(uint8_t character) {
        if(character >= 'a' && character <= 'z')
                character = character - 'a' + 'A';
        switch(character) {
        case 'A': return 0x0A;
        case 'S': return 0x1A;
        case 'D': return 0x2D;
        case 'F': return 0x3D;
        case 'G': return 0x4D;
        case 'H': return 0x5D;
        case 'J': return 0x6D;
        case 'K': return 0x7D;
        case 'L': return 0x8D;
        }

        return 0x00;
}
```

Convert the alphabet that received from keyboard to the number value that represent the frequency from the sound.

```
void Play_Sound(uint8_t character){

        chord = Mapping_Choord(character);
        if(chord == 0x00) return;

        uint8_t uart_message[20];
        sprintf(uart_message, "\r\nChord: %c\0", character);

        HAL_UART_Transmit(&huart2,(uint8_t*)uart_message, strlen(uart_message),100);

        Transmit_Audio_Data(0x1E,0x20);
        Transmit_Audio_Data(0x1C,chord);
        Transmit_Audio_Data(0x1E,0xE0);

        int i;
        for(i = 0; i < 100; i++) HAL_I2S_Transmit(&hi2s3, temp, 100, 10);
}
```

Play the sound with the given character and transmit the message to UART.

```
void Audio_Init(){

        int i;
        for(i = 0; i < 100; i++) temp[i] = 0;

        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, 0);
        HAL_Delay(500);
        HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, 1); //
        HAL_Delay(500);

        Transmit_Audio_Data(0x00,0x99);
        Transmit_Audio_Data(0x47,0x80);
        Transmit_Audio_Data(0x32,0x80);
        Transmit_Audio_Data(0x32,0x00);
        Transmit_Audio_Data(0x00,0x00);
        Transmit_Audio_Data(0x1E,0xC0);
        Transmit_Audio_Data(0x02,0x9E);
```

Initialize section following these condition from reference sheet.

4.9 Recommended Power-Up Sequence

1. Hold $\overline{\text{RESET}}$ low until the power supplies are stable.
2. Bring $\overline{\text{RESET}}$ high.
3. The default state of the “Power Ctl. 1” register (0x02) is 0x01. Load the desired register settings while keeping the “Power Ctl 1” register set to 0x01.
4. Load the required initialization settings listed in Section 4.11.
5. Apply MCLK at the appropriate frequency, as discussed in Section 4.6. SCLK may be applied or set to master at any time; LRCK may only be applied or set to master while the PDN bit is set to 1.
6. Set the “Power Ctl 1” register (0x02) to 0x9E.
7. Bring $\overline{\text{RESET}}$ low if the analog or digital supplies drop below the recommended operating condition to prevent power glitch related issues.