IC DESCRIPTION

Accelerometer (LIS302DL)

Initialize

```
88   // ----- initial accelerometer ---- transmit
89   HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_RESET);
90   uint8_t address = 0x20;
91   HAL_SPI_Transmit(&hspi1,&address,1,50);
92
93   uint8_t data = 0x67;
94   HAL_SPI_Transmit(&hspi1,&data,1,50);
95   HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_SET);
```

First we have to initialize the accelerometer, sending a signal through SPI.

PE3 must be ‘0’ when you want to communicate and ‘1’ when the communication is done

We must first make sure that PE3 is ‘0’ in order to communicate and ‘1’ when the communication is done

Getting values

```
108     //Receive
109     HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_RESET);
110
111     address = 0x29 + 0x80;
112     HAL_SPI_Transmit(&hspi1, &address, 1, 50);
113     HAL_SPI_Receive(&hspi1, &x, 1, 50);
114
115     address = 0x2B + 0x80;
116     HAL_SPI_Transmit(&hspi1, &address, 1, 50);
117     HAL_SPI_Receive(&hspi1, &y, 1, 50);
118
119     address = 0x2D + 0x80;
120     HAL_SPI_Transmit(&hspi1, &address, 1, 50);
121     HAL_SPI_Receive(&hspi1, &z, 1, 50);
122
123     HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3, GPIO_PIN_SET);
```

The address 0x29 is where the x value of accelerometer is collected.

The address 0x2B is where the y value of accelerometer is collected.

The address 0x2D is where the z value of accelerometer is collected.

After we send the address, we receive the value and put it in the corresponding variable.

```
124     itoa(x, x_out, 10);
125     itoa(y, y_out, 10);
126     itoa(z, z_out, 10);
```

Convert the value into string for print to UART.

VDO demo : https://youtu.be/guwO1R7IZcs

Microphone (MP45DT02)

Set I2S2 to Full-Duplex Master and UART2 as asynchronous.

```
/* Microphone stuffs */

uint16_t pdm_buffer[PDM_BUFFER_SIZE];
uint16_t pdm_value=0;
uint8_t  pcm_value=0;


uint16_t pcm_count = 0;

char uart_temp_display_buffer[100];

float leaky_pcm_buffer = 0.0;
float leaky_amp_buffer = 0.0;

double pcm_square = 0;
float max_amp = 0;
int currentMicrophoneAmp = 0;
int oldMicrophoneAmp = 0;

```

This is variable for using microphone.

```
  uint8_t i = 0;
  HAL_I2S_Receive(&hi2s2, pdm_buffer, PDM_BUFFER_SIZE, 1000); // Receive PDM from Mic

  for (i = 0; i < PDM_BUFFER_SIZE; i++) {
    pcm_value = -PDM_BLOCK_SIZE_BITS / 2;
    pdm_value = pdm_buffer[i];
    // calculate PCM value
    while (pdm_value != 0)        // while pdm_value still have 1s in binary
    {
      pcm_value++;
      pdm_value ^= pdm_value & -pdm_value; // remove left most 1 in binary
    }
    leaky_pcm_buffer += pcm_value;
    leaky_pcm_buffer *= LEAKY_KEEP_RATE;
    leaky_amp_buffer += float_abs(leaky_pcm_buffer);
    leaky_amp_buffer *= LEAKY_KEEP_RATE;
  }
  pcm_count++;
  if (max_amp < leaky_amp_buffer)
    max_amp = leaky_amp_buffer;
  pcm_square += (leaky_amp_buffer / 2500) * leaky_amp_buffer;
  if (pcm_count >= 2500) {
    currentMicrophoneAmp = max_amp;
    pcm_count = 0;
    pcm_square = 0;
    max_amp = 0;
  }

```

I2S2 is used to receive the value from our microphone in PDM format. Then transform PDM into PCM format. The ‘leaky_amp_buffer’ will determine the volume of the sound that received from microphone. But as this value is very ‘swing’, we need to have enough samples to actually do something. The method here to determine the volume is to use the highest value of amplitude of one sample interval (2500 samples here).

VDO demo : **https://youtu.be/Z40Q7KGZHIU**

Speaker (CS43L22)

Set I2S3 to Full-Duplex Master, I2C1 to I2C and UART2 as asynchronous.

```
53  uint16_t Istr[1];
54  uint8_t note[7] = {0x0F,0x1F,0x2F,0x3F,0x4F,0x5F,0x6F,0x7F,0x8F,0x9F,
55      0xAF,0xBF,0xCF,0xDF,0xEF,0xFF };
56  uint8_t comSound[2];
57  char mes;
58  int k;
```

This is variable for using speaker.

Initialize

```
81      Istr[0] = 0;
82
83      HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, 0);  //Reset is set down
84
85      HAL_Delay(500);
86      //Initialization sequence for CS43L22:
87      HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, 1);//Reset is set Up (Power CS43L22)
88      HAL_Delay(500);
89
90      comSound[0] = 0x00;
91      comSound[1] = 0x99;
92      HAL_I2C_Master_Transmit(&hi2c1, 0x94, comSound, 2, 50);
93      comSound[0] = 0x47;
94      comSound[1] = 0x80;
95      HAL_I2C_Master_Transmit(&hi2c1, 0x94, comSound, 2, 50);
96
97      comSound[0] = 0x32;
98      comSound[1] = 0x80; // 0xBB or 0x80
99      HAL_I2C_Master_Transmit(&hi2c1, 0x94, comSound, 2, 50);
100
101     comSound[0] = 0x32;
102     comSound[1] = 0x00; // 0x3B or 0x00
103     HAL_I2C_Master_Transmit(&hi2c1, 0x94, comSound, 2, 50);
104
105     comSound[0] = 0x00;
106     comSound[1] = 0x00;
107     HAL_I2C_Master_Transmit(&hi2c1, 0x94, comSound, 2, 50);
108
109     comSound[0] = 0x1E;
110     comSound[1] = 0xC0;
111     HAL_I2C_Master_Transmit(&hi2c1, 0x94, comSound, 2, 50);
112
113     comSound[0] = 0x02;
114     comSound[1] = 0x9E;
115     HAL_I2C_Master_Transmit(&hi2c1, 0x94, comSound, 2, 50);
```

Initialize the speaker by set the values of the speaker through I2C.

comSound[0] is the address of the register that we want to communicate and comSound[1] is the value for communicate.

Play

```
122     // Off
123     comSound[0] = 0x1E;
124     comSound[1] = 0x20;
125     HAL_I2C_Master_Transmit(&hi2c1, 0x94, comSound, 2, 50);
126     // Change note
127     comSound[0] = 0x1C;
128     comSound[1] = note[r];
129     HAL_I2C_Master_Transmit (&hi2c1, 0x94, comSound, 2, 50 );
130     // On
131     comSound[0] = 0x1E;
132     comSound[1] = 0xE0;
133     HAL_I2C_Master_Transmit(&hi2c1, 0x94, comSound, 2, 50);
134
135     int i;
136     for(i=0;i<100;i++)HAL_I2S_Transmit (&hi2s3, Istr , 200, 10 );
137
```

When we want to change the sound, we have set the speaker to ‘off’ first by sending the off signal. Then we change the value to the register 1C.

The first 4 bits of value will determine the pitch of the beep signal, with 0000 as the lowest and 1111 as the highest. The other 4 bits will determine the ‘length’.

To play the actual sound, we transmit a signal through I2S3. The sound will be played very shortly when I2S has transmitted the signal since the sound will be played longer when the loop is longer (i < 1000, i < 10000).

VDO demo : https://youtu.be/w3Ft05zKM9A