2110363 HW Sys Lab Report

Topic:

LIS302DL (3-axis Accelerometer)

MP45DT02 (Digital Microphone)

CS43L22 (Audio DAC, Speaker)

by

5731079821 Patcharida Pudpadee

Semester 1, Academic Year 2015

Note : I used STM32F4-Discovery

Link Video :

Playlist -

https://www.youtube.com/playlist?list=PLkkObKfyogUUkLvGOF67KBuen0MKqJN4Z

Accelerometer-

https://www.youtube.com/watch?v=w8KWZ_Dwt2w&list=PLkkObKfyogUUkLvGOF67KBuen0MKqJN4Z&index=1

Microphone-

https://www.youtube.com/watch?v=__Qvo-yR-YU&list=PLkkObKfyogUUkLvGOF67KBuen0MKqJN4Z&index=2

Speaker-

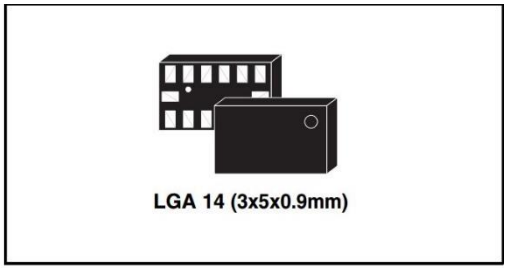
https://www.youtube.com/watch?v=f_d7KY3qr1w&list=PLkkObKfyogUUkLvGOF67KBuen0MKqJN4Z&index=3

Project-

https://www.youtube.com/watch?v=8vYTF169h1g&index=4&list=PLkkObKfyogUUkLvGOF67KBuen0MKqJN4Z

LIS302DL

3-axis Accelerometer



-The LIS302DL is a compact low-power 3-axis linear accelerometer.

-It includes a sensing element and an IC interface able to provide the measured acceleration to external world through I2C/SPI serial interface.

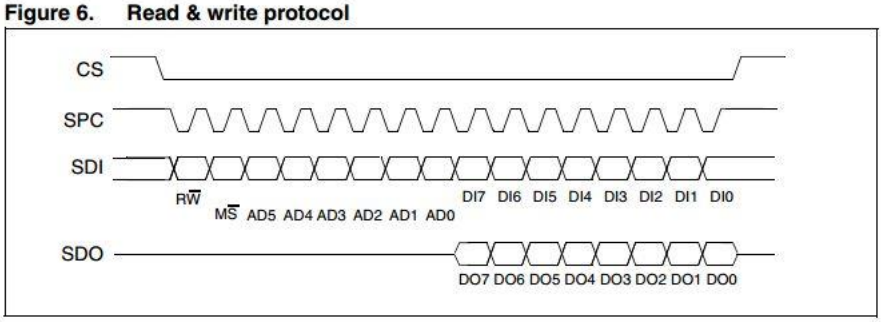
-It has dynamically of +/- 2 g or +/- 8 g and capable to measure acceleration with an output data rate of 100 Hz or 400 Hz.

-The registers embedded inside the LIS302DL may be accessed through both the I2C and SPI serial interfaces.

SPI bus interface (I use this protocol)

The LIS302DL SPI is a bus slave. The SPI allows to write and read the registers of the device. It interacts with 4 wires; CS(enable), SPC(clock), SDI(input) and SDO(output).

Figure 6. Read & write protocol



I2C Serial Interface

The LIS302DL I2C is a bus slave.

Register that use to config Accelerometer

Table 9. Serial interface pin description

Term	Description
Transmitter	The device which sends data to the bus
Receiver	The device which receives data from the bus
Master	The device which initiates a transfer, generates clock signals and terminates a transfer
Slave	The device addressed by the master

Accelerometer contains registers which are used to control its behavior and receive accelerometer data. Registers use 7 bit address.

OUT_X (29h)

Table 28. OUT_X (29h) register

XD7	XD6	XD5	XD4	XD3	XD2	XD1	XD0

X axis output data.

OUT_Y (2Bh)

Table 29. OUT_Y (2Bh) register description

YD7	YD6	YD5	YD4	YD3	YD2	YD1	YD0

Y axis output data.

OUT_Z (2Dh)

Table 30. OUT_Z (2Dh) register

ZD7	ZD6	ZD5	ZD4	ZD3	ZD2	ZD1	ZD0

Z axis output data.

CTRL_REG1 (20h) is used to initial set up sensor

Table 18. CTRL_REG1 (20h) register

DR	PD	FS	STP	STM	Zen	Yen	Xen

Table 19. CTRL_REG1 (20h) register description

DR	Data rate selection. Default value: 0 (0: 100 Hz output data rate; 1: 400 Hz output data rate)
PD	Power Down Control. Default value: 0 (0: power down mode; 1: active mode)
FS	Full Scale selection. Default value: 0 (refer to *Table 3* for typical full scale value)
STP, STM	Self Test Enable. Default value: 0 (0: normal mode; 1: self test P, M enabled)
Zen	Z axis enable. Default value: 1 (0: Z axis disabled; 1: Z axis enabled)
Yen	Y axis enable. Default value: 1 (0: Y axis disabled; 1: Y axis enabled)
Xen	X axis enable. Default value: 1 (0: X axis disabled; 1: X axis enabled)

Explain my code:

- Open SPI1 to full duplex maser, Set NVIC global ,Open USART
- Initialize all configurations register (REG1: DR=0, PD=1, FS=0,STP=0,STM=0,Zen=1,Yen=1,Xen=1)
- Reset GPIOE, GPIO_PIN_3, send and receive value of x, y and z then Set PIN_3
- Show the value

```
67 int main(void)
68 {
69
70   /* USER CODE BEGIN 1 */
71
72   /* USER CODE END 1 */
73
74   /* MCU Configuration--------------------------------------------------------*/
75
76   /* Reset of all peripherals, Initializes the Flash interface and the Systick. */
77   HAL_Init();
78
79   /* Configure the system clock */
80   SystemClock_Config();
81
82   /* Initialize all configured peripherals */
83   MX_GPIO_Init();
84   MX_SPI1_Init();
85   MX_USART2_UART_Init();
86
87   /* USER CODE BEGIN 2 */
88   uint8_t addr; //use to keep address of Register
89   uint8_t data; // use to keep data that want to send
90   uint8_t x,y,z; // represent value from each
91   uint8_t buf; // buffer use in sprintf
92
93   HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_RESET);//because PE3 is CS of SPI(RESET)/I2C(SET)
94   addr=0x20;  //address CTRL_REG1 @20h
95   HAL_SPI_Transmit(&hspi1,&addr,1,100);//send address
96 //  data=0x67;  //set value for CTRL_REG1 #01100111
97   data=0x67;  // 01000111 from DR=0, PD=1, FS=0,STP=0,STM=0,Zen=1,Yen=1,Xen=1
98   HAL_SPI_Transmit(&hspi1,&data,1,100);//send data
99   HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_SET);
100  /* USER CODE END 2 */
101  /* Infinite loop */
```

```
  /* USER CODE BEGIN WHILE */
  while (1)
  {
  /* USER CODE END WHILE */
  /* USER CODE BEGIN 3 */
      /*Get value from Accelerometer*/
      HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_RESET); //Bring CS Pin low to activate Slave device

      addr = 0x29+ 0x80;
      HAL_SPI_Transmit(&hspi1,&addr,1,50);//means want to read from 0x29 (Reg OutX)
      HAL_SPI_Receive(&hspi1,&x,1,50); // get x-axis value

      addr= 0x2B+ 0x80;
      HAL_SPI_Transmit(&hspi1,&addr,1,50);//means want to read from 0x2B (Reg OutY)
      HAL_SPI_Receive(&hspi1,&y,1,50);//get y-axis value

      addr = 0x2D+ 0x80;
      HAL_SPI_Transmit(&hspi1,&addr,1,50);//means want to read from 0x2D (Reg OutZ)
      HAL_SPI_Receive(&hspi1,&z,1,50);//get z-axis value

      HAL_GPIO_WritePin(GPIOE,GPIO_PIN_3,GPIO_PIN_SET);

      //convert to g scale
      float xx,yy,zz;
      if(x<=127) xx=x/64.0;
      else xx=(x-255)/64.0;
      if(y<=127) yy=y/64.0;
      else yy=(y-255)/64.0;
      if(z<=127) zz=z/64.0;
      else zz=(z-255)/64.0;

      char buff[10];



       int n = sprintf(&buf,"x= ");
       HAL_UART_Transmit(&huart2,&buf,n,100); //show value x
       gcvt(xx,10,buff);
       HAL_UART_Transmit(&huart2,buff,5,100);


       n = sprintf(&buf," y= ");
       HAL_UART_Transmit(&huart2,&buf,n,100); //show value y
       gcvt(yy,10,buff);
       HAL_UART_Transmit(&huart2,buff,5,100);


       n = sprintf(&buf," z= ");
       HAL_UART_Transmit(&huart2,&buf,n,100); // show value z
       gcvt(zz,10,buff);
       HAL_UART_Transmit(&huart2,buff,5,100);
       HAL_UART_Transmit(&huart2,"\n\r",2,100);

       HAL_Delay(100);
   }
   /* USER CODE END 3 */

}//end main
```

MP45DT02

digital microphone

- It is digital MEMS microphone. It is built with a sensing element and an IC interface with stereo capability.
- Its applications are mobile terminals, portable media player, speech recognition, etc.

PDM Protocol

-PDM= Pulse Density Modulation use to represent digital and analog signal.

-PDM signal can be converted by take range of data (16-bit) and shift 1 bit to next data and find accumulated sum of sample data. To find the amplitude of the sine wave, we have to take absolute of such sum. The amplitude can be considered volume of the sound wave.

- analog when using the low pass filter
- USB connector: from external mass storage, such as a USB key, USB HDD, and so on
- Internal memory of the STM32F4

Explain my code:

-Open I2S2 to full duplex master , change mode to Receive at 32 kHz

- Set NVIC to global, open USART

-sampling a range of data

-convert to base2 for better understanding

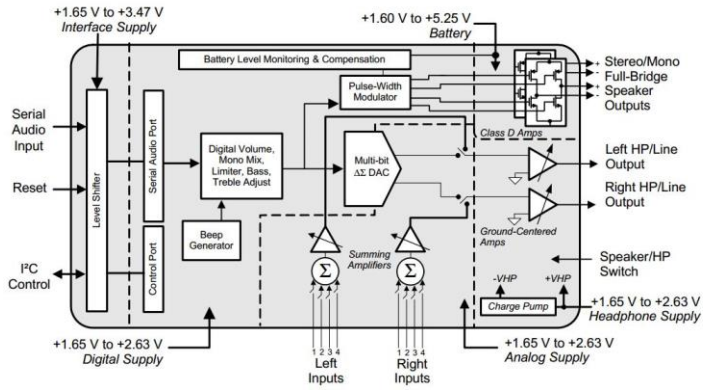
-find amplitude from contiguous sum

-set condition to show the sound level

```
/* USER CODE BEGIN 0 */
uint16_t pData[3500];
int base2inv[3500];
int base2[3500];
int tmp[3500];
/* 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_I2S2_Init();
  MX_USART2_UART_Init();

  /* USER CODE BEGIN 2 */

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
      int vol=0;
      HAL_I2S_Receive(&hi2s2,pData, 200, 1000);//receive PDM signal

      for (int i = 0; i < 200; i++) { //convert to base2
        for (int j = 0; j < 16; j++) {
            base2inv[j] = pData[i] % 2;
            pData[i] /=2;
        }

        //reverse (above cause 1 = 1000 I want =0001
        for (int j = 0; j < 16; j++) {
            base2[i * 16 + j] = base2inv[15 - j];
        }
      }
      int i =7;
      while(i + 8 < 200*16){ //do contiguous sum
         int keep = 0;
         for (int j = -7; j <= 8; j++)
          keep += base2[i + j];
         if(keep - 8 < 0)  -(keep-8);
         tmp[i] = keep-8;
         i++;
      }


    for (int i = 14; i + 16 < 50*16; i++) {
        for (int j = -7; j <= 8; j++)
            vol += tmp[i + j];
        if(vol<0) vol*-1;
    }


    while (vol > 1000){//show sound level
         HAL_UART_Transmit(&huart2,"*",1,100);
        vol -= 1000;
    }
     HAL_UART_Transmit(&huart2,"*\r\n",3,100);//print one * as default
  }
  /* USER CODE END 3 */

}//end main
```

CS43L22:

audio DAC, Speaker driver



- STM32F4 uses an audio DAC (CS43L22) to output sounds through audio jack
- STM32F4 use I2C protocol to communicate with device (from picture below speaker address is at 0x94

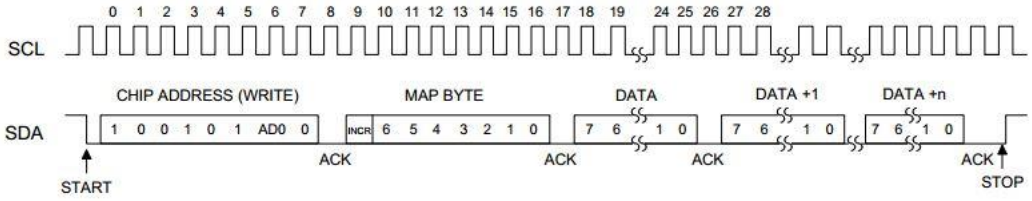


Figure 16. Control Port Timing, I²C Write

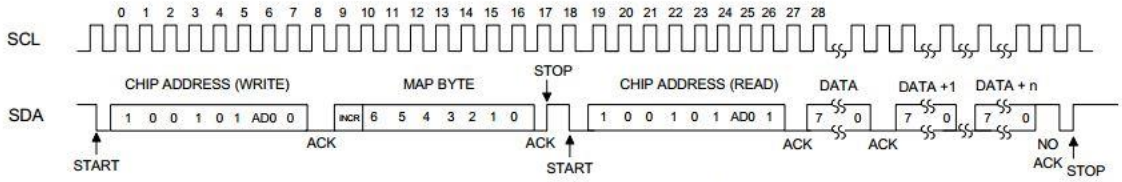


Figure 17. Control Port Timing, I²C Read

- processes digital signals with an I2S connection or an analog input signal

<u>Setting Require</u>

1. Write 0x99 to register 0x00.
2. Write 0x80 to register 0x47.
3. Write '1'b to bit 7 in register 0x32.
4. Write '0'b to bit 7 in register 0x32.
5. Write 0x00 to register 0x00.

Configuration Registers

7.15 Beep Frequency & On Time (Address 1Ch)

7	6	5	4	3	2	1	0
FREQ3	FREQ2	FREQ1	FREQ0	ONTIME3	ONTIME2	ONTIME1	ONTIME0

7.15.1 Beep Frequency

Sets the frequency of the beep signal.

FREQ[3:0]	Frequency *(Fs = 12, 24, 48 or 96 kHz)*	Pitch
0000	260.87 Hz	C4
0001	521.74 Hz	C5
0010	585.37 Hz	D5
0011	666.67 Hz	E5
0100	705.88 Hz	F5
0101	774.19 Hz	G5
0110	888.89 Hz	A5
0111	1000.00 Hz	B5
1000	1043.48 Hz	C6
1001	1200.00 Hz	D6
1010	1333.33 Hz	E6
1011	1411.76 Hz	F6
1100	1600.00 Hz	G6
1101	1714.29 Hz	A6
1110	2000.00 Hz	B6
1111	2181.82 Hz	C7
Application:	"Beep Generator" on page 22	

Notes:

1. This setting must not change when BEEP is enabled.
2. Beep frequency will scale directly with sample rate, Fs, but is fixed at the nominal Fs within each speed mode.

7.15.2 Beep On Time

Sets the on duration of the beep signal.

ONTIME[3:0]	On Time *(Fs = 12, 24, 48 or 96 kHz)*
0000	~86 ms
0001	~430 ms
0010	~780 ms
0011	~1.20 s
0100	~1.50 s
0101	~1.80 s
0110	~2.20 s
0111	~2.50 s
1000	~2.80 s
1001	~3.20 s
1010	~3.50 s
1011	~3.80 s
1100	~4.20 s
1101	~4.50 s
1110	~4.80 s
1111	~5.20 s
Application:	"Beep Generator" on page 22

Notes:

1. This setting must not change when BEEP is enabled.
2. Beep on time will scale inversely with sample rate, Fs, but is fixed at the nominal Fs within each speed mode.

7.17 Beep & Tone Configuration (Address 1Eh)

7	6	5	4	3	2	1	0
BEEP1	BEEP0	BEEPMIXDIS	TREBCF1	TREBCF0	BASSCF1	BASSCF0	TCEN

Explain my code:

-open I2S3 to full duplex master with 192 kHz

-open I2C1 to I2C and Open USART

-Initialize all required

-Wait for input from USART to play chosen note

```
/* Private variables ---------------------------------------------------------*/
uint8_t init[2];
uint8_t addrDat[2];
uint8_t addrKey[2];
uint16_t Istr[1];
uint8_t note[7] = {0x0F,0x2F,0x3F,0x4F,0x5F,0x6F,0x7F}; /*C D E F G A B*/
char mes;
int k;


/* USER CODE END PV */

/* Private function prototypes -----------------------------------------------*/
void SystemClock_Config(void);
void Error_Handler(void);
static void MX_GPIO_Init(void);
static void MX_I2C1_Init(void);
static void MX_I2S3_Init(void);
static void MX_TIM1_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_I2C1_Init();
  MX_I2S3_Init();
  MX_TIM1_Init();
  MX_USART2_UART_Init();

  /* USER CODE BEGIN 2 */

    /*Initialization*/
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, 0);
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, 1);

    init[0]=0x00; init[1]=0x99;
    HAL_I2C_Master_Transmit(&hi2c1,0x94,init,2,50);

    init[0] = 0x47;init[1] = 0x80;
    HAL_I2C_Master_Transmit(&hi2c1, 0x94, init, 2, 50);

    init[0] = 0x32;init[1] = 0x80; // write 1
    HAL_I2C_Master_Transmit(&hi2c1, 0x94, init, 2, 50);

    init[0] = 0x32;init[1] = 0x00; // write 0
```

```
    HAL_I2C_Master_Transmit(&hi2c1, 0x94, init, 2, 50);

    init[0]=0x00; init[1]=0x00;
    HAL_I2C_Master_Transmit(&hi2c1,0x94,init,2,50);

    init[0] = 0x1C;init[1] = 0xFF; //Beep Frequency config
    HAL_I2C_Master_Transmit(&hi2c1, 0x94, init, 2, 50);

    init[0] = 0x1E;init[1] = 0xE0; //Beep & Tone config
    HAL_I2C_Master_Transmit(&hi2c1, 0x94, init, 2, 50);

    init[0] = 0x02;init[1] = 0x9E; // Set Power Ctl1 register
    HAL_I2C_Master_Transmit(&hi2c1, 0x94, init, 2, 50);

  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */

      if (HAL_UART_Receive(&huart2,&mes,1,1000) == HAL_OK) {
              HAL_UART_Transmit(&huart2,&mes,1,1000); //wait for input from uart

              addrDat[0] = 0x1E;addrDat[1] = 0x20;
              HAL_I2C_Master_Transmit(&hi2c1, 0x94, addrDat, 2, 50);

              addrKey[0] = 0x1C; //select note key 1-7
              if(mes-'1'>=0&&mes-'7'<=0){
                    addrKey[1]=note[mes-'0'];
              }
              //send the note that selected
               HAL_I2C_Master_Transmit(&hi2c1, 0x94, addrKey, 2, 50);

               init[0] = 0x1E;init[1] = 0xE0; //Beep & Tone config
               HAL_I2C_Master_Transmit(&hi2c1, 0x94, init, 2, 50);

               //play note loop
               for (k=0;k<1000;k++) { HAL_I2S_Transmit (&hi2s3, Istr , 0x10, 10 );}

          }

      HAL_Delay(50);


   }
   /* USER CODE END 3 */

}//end main
```