LIS302DL : 3-axis accelerometer

Extra ports that was opened and generated by STMCube.
- SPI1 : using SPI digital output interface.
- UART2 : for sent data via USB by UART protocol.

```
40
41  /* Private variables ---------
42  SPI_HandleTypeDef hspi1;
43
44  UART_HandleTypeDef huart2;
45
46  /* USER CODE BEGIN PV */
47  /* Private variables ---------
48
```

SPI1 Configuration
Mode : Master
Others : default

UART2 Configuration
use default configuration
(Baud rate = 115200)

```
248  /* SPI1 init function */
249  static void MX_SPI1_Init(void)
250  {
251
252    hspi1.Instance = SPI1;
253    hspi1.Init.Mode = SPI_MODE_MASTER;
254    hspi1.Init.Direction = SPI_DIRECTION_2LINES;
255    hspi1.Init.DataSize = SPI_DATASIZE_8BIT;
256    hspi1.Init.CLKPolarity = SPI_POLARITY_LOW;
257    hspi1.Init.CLKPhase = SPI_PHASE_1EDGE;
258    hspi1.Init.NSS = SPI_NSS_SOFT;
259    hspi1.Init.BaudRatePrescaler = SPI_BAUDRATEPRESCALER_8;
260    hspi1.Init.FirstBit = SPI_FIRSTBIT_MSB;
261    hspi1.Init.TIMode = SPI_TIMODE_DISABLE;
262    hspi1.Init.CRCCalculation = SPI_CRCCALCULATION_DISABLE;
263    hspi1.Init.CRCPolynomial = 10;
264    if (HAL_SPI_Init(&hspi1) != HAL_OK)
265    {
266      Error_Handler();
267    }
268
269  }
270
271  /* USART2 init function */
272  static void MX_USART2_UART_Init(void)
273  {
274
275    huart2.Instance = USART2;
276    huart2.Init.BaudRate = 115200;
277    huart2.Init.WordLength = UART_WORDLENGTH_8B;
278    huart2.Init.StopBits = UART_STOPBITS_1;
279    huart2.Init.Parity = UART_PARITY_NONE;
280    huart2.Init.Mode = UART_MODE_TX_RX;
281    huart2.Init.HwFlowCtl = UART_HWCONTROL_NONE;
282    huart2.Init.OverSampling = UART_OVERSAMPLING_16;
283    if (HAL_UART_Init(&huart2) != HAL_OK)
284    {
285      Error_Handler();
286    }
```

All variable and modified method that used in this project.

```
63  /* USER CODE BEGIN 0 */
64  uint8_t address,data,x_accel,y_accel,z_accel;
65  int timeout = 1000,x_sign,y_sign,z_sign;
66  char x_str[10],y_str[10],z_str[10];
67  char x_str2[10],y_str2[10],z_str2[10];
68  /*Magic Happened, I need to declare this unused char array*/
69
70  void my_dtoc(double f,char * buffer){
71      gcvt(f,10,buffer);
72  }
73
74  /* USER CODE END 0 */
```

my_dtoc method : receive double and return it in string format.

Initial Codes

```
96   /* USER CODE BEGIN 2 */
97   /* ----------------------------------------
98    #1 : Bring the CS pin low to activate the active device.
99    #2 : Declare transmit address.
100   #3 : Write data to register.
101   #4 : Bring CS pin High again*/
102
103  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3,GPIO_PIN_RESET);
104  address = 0x20;data = 0x67;
105  HAL_SPI_Transmit(&hspi1,&address,1,50);
106  HAL_SPI_Transmit(&hspi1,&data,1,50);
107  HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3,GPIO_PIN_SET);
108
109  /*Show LED for initializing complete*/
110  HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12,1);
111  HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13,1);
112  HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14,1);
113  HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15,1);
114
```

Objective : to prepare accelerometer IC for receiving data , can be done by this 4 steps.

1. Bring the CS pin to low (reset) to activate IC.
2. Declare transmit address (0x20) and stored data (0x67).
3. Write declared address in step 3 to register.
4. Bring the CS pin back to high (set) to finish initializing.

7.2 CTRL_REG1 (20h)

Table 18. CTRL_REG1 (20h) register

DR	PD	FS	STP	STM	Zen	Yen	Xen

*0x20 is CTRL Register address.

While-Loop Codes

1st Part : Reading acceleration value (0-255) from each axis.

```
120     /*Send Signal for Start Reading Value*/
121     HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3,GPIO_PIN_RESET);
122
123     /*SPI Receive, read X-axis acceleration from address 0x29*/
124     address = 0x29+0x80;
125     HAL_SPI_Transmit(&hspi1,&address,1,50);
126     HAL_SPI_Receive(&hspi1,&x_accel,1,50);
127
128     /*SPI Receive, read Y-axis acceleration from address 0x2B*/
129     address = 0x2B+0x80;
130     HAL_SPI_Transmit(&hspi1,&address,1,50);
131     HAL_SPI_Receive(&hspi1,&y_accel,1,50);
132
133     /*SPI Receive, read Z-axis acceleration from address 0x2D*/
134     address = 0x2D+0x80;
135     HAL_SPI_Transmit(&hspi1,&address,1,50);
136     HAL_SPI_Receive(&hspi1,&z_accel,1,50);
137
138     /*Send Signal for End Reading Value*/
139     HAL_GPIO_WritePin(GPIOE, GPIO_PIN_3,GPIO_PIN_SET);
140
```

7.7 OUT_X (29h)

Table 28. OUT_X (29h) register

XD7	XD6	XD5	XD4	XD3	XD2	XD1	XD0

X axis output data.

*X-axis data address is 0x29+0x80

7.8 OUT_Y (2Bh)

Table 29. OUT_Y (2Bh) register description

YD7	YD6	YD5	YD4	YD3	YD2	YD1	YD0

Y axis output data.

*Y-axis data address is 0x2B+0x80

7.9 OUT_Z (2Dh)

Table 30. OUT_Z (2Dh) register

ZD7	ZD6	ZD5	ZD4	ZD3	ZD2	ZD1	ZD0

Z axis output data.

*Z-axis data address is 0x2D+0x80

2nd Part : Convert acceleration value from previous part to scaled in +/- 2g and prepare array of char for next part.

```
141     /*Convert -> +/-2g */
142     if (x_accel <= 67 && x_accel >= 65) { x_str[0]='1';x_str[1]='.';x_str[2]='0'; }
143     else if (x_accel >= 255 && x_accel <= 1) { x_str[0]='0';x_str[1]='.';x_str[2]='0'; }
144     else if (x_accel <= 124) { my_dtoc((x_accel/66.0),x_str); }
145     else { my_dtoc(((255-x_accel)/66.0),x_str); }
146
147     if (y_accel <= 67 && y_accel >= 65) { y_str[0]='1';y_str[1]='.';y_str[2]='0'; }
148     else if (y_accel >= 255 && y_accel <= 1) { y_str[0]='0';y_str[1]='.';y_str[2]='0'; }
149     else if (y_accel <= 124) { my_dtoc((y_accel/66.0),y_str); }
150     else { my_dtoc(((255-y_accel)/66.0),y_str); }
151
152     if (z_accel <= 67 && z_accel >= 65) { z_str[0]='1';z_str[1]='.';z_str[2]='0'; }
153     else if (z_accel >= 255 && z_accel <= 1) { z_str[0]='0';z_str[1]='.';z_str[2]='0'; }
154     else if (z_accel <= 124) { my_dtoc((z_accel/66.0),z_str); }
155     else { my_dtoc(((255-z_accel)/66.0),z_str); }
156
```

3rd Part : Send an array of char of each axis through USB by UART protocols to display a value (in +/-2g format) in serial port terminal (putty) .

```
157     /*UART X-Axis Jobs*/
158     HAL_UART_Transmit(&huart2," X-Axis = ",10,timeout);
159     if (x_accel <= 256 && x_accel >=126) { HAL_UART_Transmit(&huart2,"-",1,timeout); }
160     else { HAL_UART_Transmit(&huart2,"+",1,timeout); }
161     if (x_str[1] == '.') { HAL_UART_Transmit(&huart2,&x_str,3,timeout); }
162     else { HAL_UART_Transmit(&huart2,"0.0",3,timeout);}
163     HAL_UART_Transmit(&huart2,"g",1,timeout);
164
165     /*UART Y-Axis Jobs*/
166     HAL_UART_Transmit(&huart2," | Y-Axis = ",12,timeout);
167     if (y_accel <= 256 && y_accel >=126) { HAL_UART_Transmit(&huart2,"-",1,timeout); }
168     else { HAL_UART_Transmit(&huart2,"+",1,timeout); }
169     if (y_str[1] == '.') { HAL_UART_Transmit(&huart2,&y_str,3,timeout); }
170     else { HAL_UART_Transmit(&huart2,"0.0",3,timeout);}
171     HAL_UART_Transmit(&huart2,"g",1,timeout);
172
173     /*UART Z-Axis Jobs*/
174     HAL_UART_Transmit(&huart2," | Z-Axis = ",12,timeout);
175     if (z_accel <= 256 && z_accel >=126) { HAL_UART_Transmit(&huart2,"-",1,timeout); }
176     else { HAL_UART_Transmit(&huart2,"+",1,timeout); }
177     if (z_str[1] == '.') { HAL_UART_Transmit(&huart2,&z_str,3,timeout); }
178     else { HAL_UART_Transmit(&huart2,"0.0",3,timeout);}
179     HAL_UART_Transmit(&huart2,"g",1,timeout);
180
181     HAL_UART_Transmit(&huart2,"\r\n",2,timeout);
182
183     /* USER CODE END WHILE */
```

***Finally, this video helps me a lot**

www.youtube.com/watch?v=OBuuWzyPMPg