CS43L22 : audio DAC, speaker driver

Extra ports that was opened and generated by STMCube.
- I2C : using I2C interface to control ports operations.
- I2S : using I2S standard format on digital interface.
- TIM1 : a timer with internal clock.
- UART2 : for sent data via USB by UART protocol.

```
41  /* Private variables ---------
42  I2C_HandleTypeDef hi2c1;
43
44  I2S_HandleTypeDef hi2s3;
45
46  TIM_HandleTypeDef htim1;
47
48  UART_HandleTypeDef huart2;
49
50  /* USER CODE BEGIN PV */
51  /* Private variables ---------
```

```
245  /* I2C1 init function */
246  static void MX_I2C1_Init(void)
247  {
248
249    hi2c1.Instance = I2C1;
250    hi2c1.Init.ClockSpeed = 50000;
251    hi2c1.Init.DutyCycle = I2C_DUTYCYCLE_2;
252    hi2c1.Init.OwnAddress1 = 0;
253    hi2c1.Init.AddressingMode = I2C_ADDRESSINGMODE_7BIT;
254    hi2c1.Init.DualAddressMode = I2C_DUALADDRESS_DISABLE;
255    hi2c1.Init.OwnAddress2 = 0;
256    hi2c1.Init.GeneralCallMode = I2C_GENERALCALL_DISABLE;
257    hi2c1.Init.NoStretchMode = I2C_NOSTRETCH_DISABLE;
258    if (HAL_I2C_Init(&hi2c1) != HAL_OK)
259    {
260      Error_Handler();
261    }
262
263  }
```

I2C1 Configuration

Clock speed : 50000
Addressing mode : 7 bits
Others : default

```
265  /* I2S3 init function */
266  static void MX_I2S3_Init(void)
267  {
268
269    hi2s3.Instance = SPI3;
270    hi2s3.Init.Mode = I2S_MODE_MASTER_TX;
271    hi2s3.Init.Standard = I2S_STANDARD_MSB;
272    hi2s3.Init.DataFormat = I2S_DATAFORMAT_16B;
273    hi2s3.Init.MCLKOutput = I2S_MCLKOUTPUT_ENABLE;
274    hi2s3.Init.AudioFreq = I2S_AUDIOFREQ_44K;
275    hi2s3.Init.CPOL = I2S_CPOL_LOW;
276    hi2s3.Init.ClockSource = I2S_CLOCK_PLL;
277    hi2s3.Init.FullDuplexMode = I2S_FULLDUPLEXMODE_DISABLE;
278    if (HAL_I2S_Init(&hi2s3) != HAL_OK)
279    {
280      Error_Handler();
281    }
282
283  }
```

I2S3 Configuration

Mode : Master Transmit
Standard : MSB
Data Format : 16 Bytes
Audio Frequency : 44K
Others : default

```c
/* TIM1 init function */
static void MX_TIM1_Init(void)
{

  TIM_ClockConfigTypeDef sClockSourceConfig;
  TIM_MasterConfigTypeDef sMasterConfig;

  htim1.Instance = TIM1;
  htim1.Init.Prescaler = 1680;
  htim1.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim1.Init.Period = 99;
  htim1.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim1.Init.RepetitionCounter = 0;
  if (HAL_TIM_Base_Init(&htim1) != HAL_OK)
  {
    Error_Handler();
  }

  sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
  if (HAL_TIM_ConfigClockSource(&htim1, &sClockSourceConfig) != HAL_OK)
  {
    Error_Handler();
  }

  sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
  sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
  if (HAL_TIMEx_MasterConfigSynchronization(&htim1, &sMasterConfig) != HAL_OK)
  {
    Error_Handler();
  }

}
```

TIM1 Configuration

Prescaler : 1680
Counter Mode : Up
Period : 99
Others : default

```c
/* 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();
  }

}
```

UART2 Configuration

use default configuration
(Baud rate = 115200)

All variable that used in this project.

```c
/* Private variables ---------------------------------------------------------*/

uint8_t initV[2];
uint16_t Istr[1];
uint8_t note[7] = {0x1F,0x2F,0x3F,0x4F,0x5F,0x6F,0x7F}; /*C D E F G A B*/
char msg;int k;
uint16_t size = 1;
uint32_t timeout = 1000;
```

Initial Codes

```
104     /*Initialize CS43L22*/
105     HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, 0);
106     HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, 1);
107
108     initV[0] = 0x47;initV[1] = 0x80;
109     HAL_I2C_Master_Transmit(&hi2c1, 0x94, initV, 2, 50);
110
111     initV[0] = 0x32;initV[1] = 0x80;
112     HAL_I2C_Master_Transmit(&hi2c1, 0x94, initV, 2, 50);
113
114     initV[0] = 0x32;initV[1] = 0x00;
115     HAL_I2C_Master_Transmit(&hi2c1, 0x94, initV, 2, 50);
116
117     initV[0] = 0x1C;initV[1] = 0xAF;
118     HAL_I2C_Master_Transmit(&hi2c1, 0x94, initV, 2, 50);
119
120     initV[0] = 0x1E;initV[1] = 0xE0;
121     HAL_I2C_Master_Transmit(&hi2c1, 0x94, initV, 2, 50);
122
123     initV[0] = 0x02;initV[1] = 0x9E;
124     HAL_I2C_Master_Transmit(&hi2c1, 0x94, initV, 2, 50);
125
126     /*Show LED for initializing complete*/
127     HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13,1);
128
```

Objective : to prepare speaker driver and setting an initial note to be played in the while-loop codes.

4.11 Required Initialization Settings

Various sections in the device must be adjusted by implementing the initialization settings shown below after power-up sequence step 3. All performance and power consumption measurements were taken with the following settings:

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.

*Using this guideline in datasheet.

While-Loop Codes

1st Part : Checking if user pressing some key (sending a char through USB by UART protocol), If it happen send that char back to display in serial port terminal (putty) .

```
137   /* USER CODE BEGIN 3 */
138
139     if (HAL_UART_Receive(&huart2,&msg,size,timeout) == HAL_OK) {
140           HAL_UART_Transmit(&huart2,&msg,size,timeout);
```

2nd Part : Checking if that char is the correct note (C D E F G A B), If corrected it will preparing to change note to the note relate to that char by 4 steps.

1. Send a signal for prepare changing note. (lines : 145,146)
2. Changing note to specific value. (lines : 148-156)
3. Send a signal to finish changing note. (lines : 158,159)
4. Playing a sound in 1000 times loop (lines : 162)

```
141         if (msg=='c' || msg=='d'|| msg=='e' || msg=='f' || msg=='g' || msg=='a' || msg=='b' || msg=='C'
142           || msg=='D' || msg=='E' || msg=='F' || msg=='G' || msg=='A' || msg=='B') {
143           HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15,0);
144
145           initV[0] = 0x1E;initV[1] = 0x20;
146           HAL_I2C_Master_Transmit(&hi2c1, 0x94, initV, 2, 50);
147
148           initV[0] = 0x1C; /*Select the right note*/
149           if (msg=='c' || msg=='C') { initV[1] = note[0]; }
150           if (msg=='d' || msg=='D') { initV[1] = note[1]; }
151           if (msg=='e' || msg=='E') { initV[1] = note[2]; }
152           if (msg=='f' || msg=='F') { initV[1] = note[3]; }
153           if (msg=='g' || msg=='G') { initV[1] = note[4]; }
154           if (msg=='a' || msg=='A') { initV[1] = note[5]; }
155           if (msg=='b' || msg=='B') { initV[1] = note[6]; }
156           HAL_I2C_Master_Transmit(&hi2c1, 0x94, initV, 2, 50);
157
158           initV[0] = 0x1E;initV[1] = 0xE0;
159           HAL_I2C_Master_Transmit(&hi2c1, 0x94, initV, 2, 50);
160
161           /*Play note loop*/
162           for (k=0;k<1000;k++) { HAL_I2S_Transmit (&hi2s3, Istr , 0x10, 10 );}
163
164           HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12,1); /*Show LED for pressing a right keys*/
165
166         } else { HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14,1);HAL_Delay(50); }  /*Show LED for pressing a wrong keys*/
167     }
```

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	

*Showing value of each note in 0-16. (EX. C5 = 0x01)

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

*Showing how long that sound will be played.

** If we want to play sound C5 as long time as it provide we need to send 0x1F via function *HAL_I2S_TRANSMIT*

3rd Part : Showing Blue LED to let me know that it's working in waiting state.

```
169       /*Show LED for waiting state*/
170       HAL_Delay(50);
171       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12,0);
172       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14,0);
173       HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15,1);
174
175     }
176     /* USER CODE END 3 */
```

For more information about how it works, you can read it in datasheet of CS43L22_F2