Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра обчислювальної техніки

Архітектура комп'ютерів 3. Мікропроцесорні системи

Воркшоп 5

«Communication: I2C»

Виконав:
студент групи ІО-23
Корбут М. Я.
Залікова книжка №2313
Перевірив
Каплунов А.В.

Київ - 2025

Воркшоп 5

Тема: «Communication: I2C»

Мета: зрозуміти як працюють I2C інтерфейс та як його налаштовувати.

Хід роботи:

У процесі роботи було активовано та налаштовано послідовний інтерфейс I2C для керування зовнішнім ЦАП CS43L22 та його вбудованим генератором тонів. Також використовувався інтерфейси DMA та I2S3. З програмного боку використовувалися спеціальні функції для ініціалізації модуля та передачі йому команд для програвання мелодії. Також використовувалася спеціальна функція для створення світломузики за допомогою вбудованих світлодіодів. Відповідний фрагмент коду з файлу main.c:

```
typedef enum
{
  C4,
  C5,
  D5,
  E5,
  F5,
  G5,
  A5,
  B5,
  C6,
  D6,
  E6,
  F6,
  G6,
  A6,
  B6,
  C7,
  MAX_VALUE
} soundToneType;

static const uint8_t soundToneReg[16]={   //list of all possible register values
    0x09,
    0x19,
    0x29,
    0x39,
    0x49,
    0x59,
    0x69,
    0x79,
```

```
    0x89,
    0x99,
    0xA9,
    0xB9,
    0xC9,
    0xD9,
    0xE9,
    0xF9
};

/* USER CODE BEGIN PD */
#define CS43L22_I2C_ADDRESS   0x94
#define I2C_TIMEOUT       10

/* USER CODE END PD */

/* USER CODE BEGIN PV */
int16_t dataI2S[100] = {0};
/* USER CODE END PV */

void CS43L22_Init(void)
{
  // Enable chip
  HAL_GPIO_WritePin(GPIOD, GPIO_PIN_4, GPIO_PIN_SET);

  //
  // Initialization
  //
  uint8_t TxBuffer[2];

  TxBuffer[0] = 0x0D;
  TxBuffer[1] = 0x01;
  HAL_I2C_Master_Transmit(&hi2c1, CS43L22_I2C_ADDRESS, (uint8_t*) &TxBuffer, 2,
I2C_TIMEOUT);

  TxBuffer[0] = 0x00;
  TxBuffer[1] = 0x99;
  HAL_I2C_Master_Transmit(&hi2c1, CS43L22_I2C_ADDRESS, (uint8_t*) &TxBuffer, 2,
I2C_TIMEOUT);

  TxBuffer[0] = 0x47;
  TxBuffer[1] = 0x80;
  HAL_I2C_Master_Transmit(&hi2c1, CS43L22_I2C_ADDRESS, (uint8_t*) &TxBuffer, 2,
I2C_TIMEOUT);

  TxBuffer[0] = 0x32;
  TxBuffer[1] = 0xFF;
  HAL_I2C_Master_Transmit(&hi2c1, CS43L22_I2C_ADDRESS, (uint8_t*) &TxBuffer, 2,
I2C_TIMEOUT);

  TxBuffer[0] = 0x32;
```

```
  TxBuffer[1] = 0x7F;
  HAL_I2C_Master_Transmit(&hi2c1, CS43L22_I2C_ADDRESS, (uint8_t*) &TxBuffer, 2,
I2C_TIMEOUT);

  TxBuffer[0] = 0x00;
  TxBuffer[1] = 0x00;
  HAL_I2C_Master_Transmit(&hi2c1, CS43L22_I2C_ADDRESS, (uint8_t*) &TxBuffer, 2,
I2C_TIMEOUT);

  TxBuffer[0] = 0x04;
  TxBuffer[1] = 0xAF;
  HAL_I2C_Master_Transmit(&hi2c1, CS43L22_I2C_ADDRESS, (uint8_t*) &TxBuffer, 2,
I2C_TIMEOUT);

  TxBuffer[0] = 0x0D;
  TxBuffer[1] = 0x70;
  HAL_I2C_Master_Transmit(&hi2c1, CS43L22_I2C_ADDRESS, (uint8_t*) &TxBuffer, 2,
I2C_TIMEOUT);

  TxBuffer[0] = 0x05;
  TxBuffer[1] = 0x81;
  HAL_I2C_Master_Transmit(&hi2c1, CS43L22_I2C_ADDRESS, (uint8_t*) &TxBuffer, 2,
I2C_TIMEOUT);

  TxBuffer[0] = 0x06;
  TxBuffer[1] = 0x07;
  HAL_I2C_Master_Transmit(&hi2c1, CS43L22_I2C_ADDRESS, (uint8_t*) &TxBuffer, 2,
I2C_TIMEOUT);

  TxBuffer[0] = 0x0A;
  TxBuffer[1] = 0x00;
  HAL_I2C_Master_Transmit(&hi2c1, CS43L22_I2C_ADDRESS, (uint8_t*) &TxBuffer, 2,
I2C_TIMEOUT);

  TxBuffer[0] = 0x27;
  TxBuffer[1] = 0x00;
  HAL_I2C_Master_Transmit(&hi2c1, CS43L22_I2C_ADDRESS, (uint8_t*) &TxBuffer, 2,
I2C_TIMEOUT);

  TxBuffer[0] = 0x1A;
  TxBuffer[1] = 0x0A;
  HAL_I2C_Master_Transmit(&hi2c1, CS43L22_I2C_ADDRESS, (uint8_t*) &TxBuffer, 2,
I2C_TIMEOUT);

  TxBuffer[0] = 0x1B;
  TxBuffer[1] = 0x0A;
  HAL_I2C_Master_Transmit(&hi2c1, CS43L22_I2C_ADDRESS, (uint8_t*) &TxBuffer, 2,
I2C_TIMEOUT);

  TxBuffer[0] = 0x1F;
  TxBuffer[1] = 0x0F;
```

```
  HAL_I2C_Master_Transmit(&hi2c1, CS43L22_I2C_ADDRESS, (uint8_t*) &TxBuffer, 2,
I2C_TIMEOUT);

  TxBuffer[0] = 0x02;
  TxBuffer[1] = 0x9E;
  HAL_I2C_Master_Transmit(&hi2c1, CS43L22_I2C_ADDRESS, (uint8_t*) &TxBuffer, 2,
I2C_TIMEOUT);
}

void CS43L22_Beep(soundToneType pitch, uint32_t duration_ms)
{
  uint8_t TxBuffer[2];

  // Set volume and off time
  TxBuffer[0] = 0x1D;   // Register address
  TxBuffer[1] = 0x00;   // Value (volume and off time)
  HAL_I2C_Master_Transmit(&hi2c1, CS43L22_I2C_ADDRESS, (uint8_t*) &TxBuffer, 2,
I2C_TIMEOUT);

  // Set sound frequency
  TxBuffer[0] = 0x1C;   // Register address
  if (pitch < MAX_VALUE) {
    TxBuffer[1] = soundToneReg[pitch]; //Set pitch from the list
  }
  else{
    TxBuffer[1] = soundToneReg[15]; //Set the highest available
  }
  HAL_I2C_Master_Transmit(&hi2c1, CS43L22_I2C_ADDRESS, (uint8_t*) &TxBuffer, 2,
I2C_TIMEOUT);

  // Enable continuous mode (SOUND STARTED)
  TxBuffer[0] = 0x1E;   // Register address
  TxBuffer[1] = 0xC0;   // Value (beep and tone configuration)
  HAL_I2C_Master_Transmit(&hi2c1, CS43L22_I2C_ADDRESS, (uint8_t*) &TxBuffer, 2,
I2C_TIMEOUT);

  // Playing...
  HAL_Delay(duration_ms);

  // Disable continuous mode (SOUND STOPED)
  TxBuffer[0] = 0x1E;   // Register address
  TxBuffer[1] = 0x00;   // Value (beep and tone configuration)
  HAL_I2C_Master_Transmit(&hi2c1, CS43L22_I2C_ADDRESS, (uint8_t*) &TxBuffer, 2,
I2C_TIMEOUT);
}

void lightMusic(soundToneType note){
  HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12 | GPIO_PIN_13 | GPIO_PIN_14 | GPIO_PIN_15,
GPIO_PIN_RESET);  // disables all LEDs

  switch(note){ // picks LEDs to light based on the note selected
```

```
  case C4:
  case C5:
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12, GPIO_PIN_SET);
    break;
  case D5:
  case D6:
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13, GPIO_PIN_SET);
    break;
  case E5:
  case E6:
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14, GPIO_PIN_SET);
    break;
  case F5:
  case F6:
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15, GPIO_PIN_SET);
    break;
  case G5:
  case G6:
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12 | GPIO_PIN_13, GPIO_PIN_SET);
    break;
  case A5:
  case A6:
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_13 | GPIO_PIN_14, GPIO_PIN_SET);
    break;
  case B5:
  case B6:
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_14 | GPIO_PIN_15, GPIO_PIN_SET);
    break;
  case C6:
  case C7:
  default:
    HAL_GPIO_WritePin(GPIOD, GPIO_PIN_15 | GPIO_PIN_12, GPIO_PIN_SET);
    break;
  }
}

/* USER CODE END 0 */

/* USER CODE BEGIN WHILE */
  HAL_Delay(1000);  //delay before starting

  int i = 1;

  soundToneType melody[] = {
    A5, A5, A5, F5, C6, A5, F5, C6, A5,
      E6, E6, E6, F6, C6, G5, F5, C6,
      A5, F5, C6, A5, F5, C6, A5,
      E6, E6, E6, F6, C6, G5, F5, C6,
      // remaining unused notes for full coverage
      B5, D5, B6, D6, G6, A6, B6, C7, C4};
```

```
  uint32_t noteDurations[] = {
      500, 500, 500, 350, 150, 500, 350, 150, 1000,
      500, 350, 150, 500, 350, 150, 500, 1000,
      500, 500, 350, 150, 500, 350, 150,
      500, 350, 150, 500, 350, 150, 500, 1000,
      // Fast run through all remaining notes
      200, 200, 200, 200, 200, 200, 200, 500,
      400};

  int melodyNoteCount = sizeof(melody) / sizeof(melody[0]);

  while (i<4) { //repeat the melody 3 times

      for(int j=0; j<melodyNoteCount; ++j){ //for each note
          CS43L22_Beep(melody[j], noteDurations[j]);  //Play the note
          lightMusic(melody[j]);  //activate LEDs
          HAL_Delay(50);  //wait before the next note
      }
      HAL_Delay(1000);  //delay between melodies
      i++;
  }

  HAL_GPIO_WritePin(GPIOD, GPIO_PIN_12 | GPIO_PIN_13 | GPIO_PIN_14 | GPIO_PIN_15,
GPIO_PIN_RESET);  // disables all LEDs
  while (1) {}

  /* USER CODE END 3 */
```

Зауважте, що коментарі написано англійською, адже вони писалися на віддаленій машині, де я не знайшов можливості змінити розкладку.

Результати роботи програми:

Після запуску, програма тричі відтворює мелодію з “зоряних війн”, супроводжуючи її світломузикою.

Репозиторій

Код було завантажено до репозиторію GitHub. Переглянути його можна за посиланням.