УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Проектирование вычислительных систем»

Лабораторная работа №3

Вариант 6

Студент

Кузнецов М. А.

Белогаев Д. В.

P34131

Преподаватель

Пинкевич В. Ю.

Санкт-Петербург, 2023 г.

Задание лабораторной работы

Разработать программу, которая использует таймеры для управления яркостью светодиодов и излучателем звука (по прерыванию или с использованием аппаратных каналов). Блокирующее ожидание (функция HAL_Delay()) в программе использоваться не должно.

Стенд должен поддерживать связь с компьютером по UART и выполнять указанные действия в качестве реакции на нажатие кнопок на клавиатуре компьютера. В данной лабораторной работе каждая нажатая кнопка (символ, отправленный с компьютера на стенд) обрабатываются отдельно, ожидание ввода полной строки не требуется.

Для работы с UART на стенде можно использован один из двух вариантов драйвера (по прерыванию и по опросу) на выбор исполнителя. Поддержка двух вариантов не требуется.

Вариант задания

Реализовать «музыкальную клавиатуру» с помощью излучателя звука. Существует девять стандартных октав от субконтроктавы (первая по порядку) до пятой октавы (девятая по порядку) (более подробно об октавах см. в специализированных источниках). Частоты нот в соседних октавах отличаются ровно в два раза и растут с номером октавы. Частоты для первой октавы (пятая по порядку):

Нота	Частота, Гц
До	261,63
Ре	293,67
Ми	329,63
Фа	349,23
Соль	392,00
Ля	440,00
Си	493,88

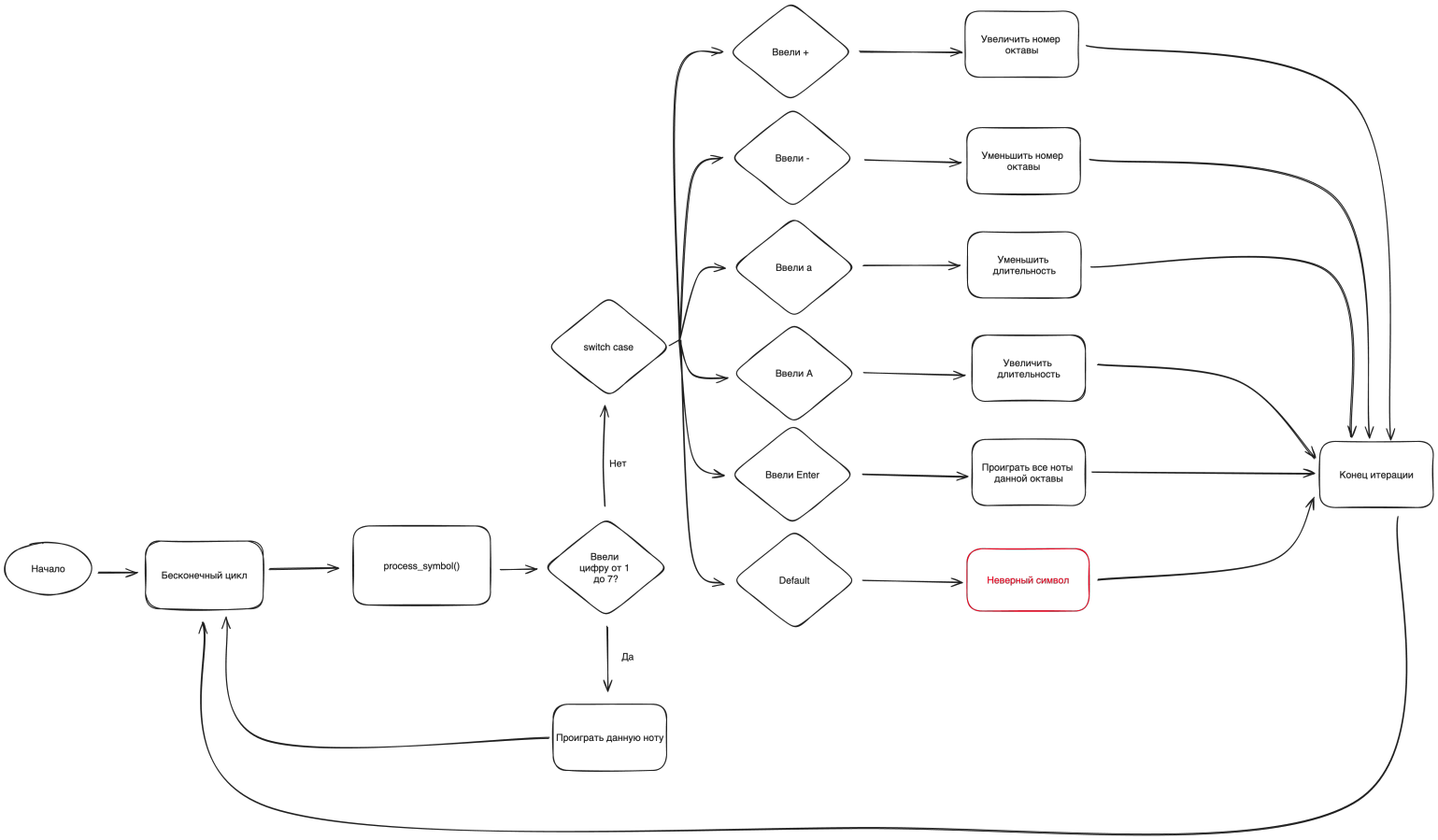
Символ	Действие
«1» – «7»	Воспроизведение одной ноты (от «до» до «си») текущей октавы с текущей длительностью звучания. Начальные значения: первая октава (пятая по порядку), длительность 1 с.
«+»	Увеличение номера текущей октавы (максимальная – пятая).
«-»	Уменьшение номера текущей октавы (минимальная – субконтроктава).
«A»	Увеличение длительности воспроизведения ноты на 0,1 с (максимум – 5 с).
«a»	Уменьшение длительности воспроизведения ноты на 0,1 с (минимум – 0,1 с).
«Enter»	Последовательное воспроизведение всех нот текущей октавы с текущей длительностью без пауз.

По вводу каждого символа в UART должно выводиться сообщение:

- для символов «1» – «7», «Enter»: какая нота какой октавы и с какой длительностью проигрывается
- для символов настройки: новые значения номера октавы и длительности звучания ноты;

- для символов, не перечисленных в таблице выше: сообщение «неверный символ» и его код.

Блок схема



Исходный код

Объявление переменных

```
const uint32_t oct_size = 7;
uint32_t freqs[] = { 16350, 18350, 20610, 21820, 24500, 27500, 30870, 32700, 36950,
                     41210, 43650, 49000, 55000, 61740, 65410, 73910, 82410, 87310, 98000,
                     110000, 123480, 130820, 147830, 164810, 174620, 196000, 220000, 110000,
                     261630, 293670, 329630, 349230, 392000, 440000, 493880, 523260, 587340,
                     659260, 698460, 784000, 880000, 987760, 1046520, 1174680, 1318500,
                     1396900, 1568000, 1720000, 1975500, 2093000, 2349200, 2637000, 2739800,
                     3136000, 3440000, 3951000, 4186000, 4698400, 5274000, 5587000, 6271000,
7040000, 7902000 };
uint32_t note_index = 0;
uint32_t octave = 4;
uint32_t duration = 1000;
uint8_t is_all_playing = 0;
char* note_name[] = {"До", "Ре", "Ми", "Фа", "Соль", "Ля", "Си"};

uint8_t is_writing_now = 0;
char read_buffer[100];
char write_buffer[100];
char* cur_process_char = read_buffer;
char* cur_read_char = read_buffer;
char* transmit_from_pointer = write_buffer;
char* write_to_pointer = write_buffer;

char* concat(char *s1, char *s2) {
    char *result = malloc(strlen(s1) + strlen(s2) + 1);
```

```
    strcpy(result, s1);
    strcat(result, s2);
    return result;
}

void next(char **pointer, char *buffer) {
    if(*pointer >= buffer + 100){
        *pointer = buffer;
    }
    else {
        (*pointer)++;
    }
}

void write_char_to_buff(char c) {
    *write_to_pointer = c;
    next(&write_to_pointer, write_buffer);
}

void write(char* str) {
    char* str_with_newlines = concat("\r\n", str);
    int size = sizeof(char)*strlen(str_with_newlines);
    for(size_t i = 0; str_with_newlines[i] != '\0'; i++) {
        write_char_to_buff(str_with_newlines[i]);
    }
}

int is_number(char* str) {
    for (size_t i = 0; str[i] != '\0'; i++) {
      if (!isdigit(str[i])) return 0;
    }
    return 1;
}

void restart_timer() {
    TIM6->CNT = 0;
}

void mute() {
    TIM1->CCR1 = 0;
}

void unmute() {
    TIM1->CCR1 = TIM1->ARR / 2;
}

void set_frequency(uint32_t freq_millis) {
        TIM1->PSC = ((2 * HAL_RCC_GetPCLK2Freq()) / (2 * (TIM1->ARR) *
(freq_millis / 1000))) - 1;
;
}

int get_frequency(uint32_t index) {
    return freqs[index + (octave * oct_size)];
}

void play(uint32_t index) {
    int freq = get_frequency(index);
    if (freq > 0) {
        set_frequency(freq);
        restart_timer();
        unmute();
    } else {
        if (is_all_playing) {
            mute();
            is_all_playing = 0;
```

```
        } else {
            char answer[100];
            sprintf(answer, "Нет ноты %s в октаве %d!",
note_name[note_index], octave);
            write(answer);
        }
    }

}

void start_playing() {
    is_all_playing = 1;
    if (octave > 0) {
        note_index = 0;
    } else {
        note_index = 2;
    }
    play(note_index);
}

void decreas_duration(){
    if (duration > 100) {
        duration -= 100;
        TIM6->ARR = duration;
        restart_timer();
    }
}

void increase_duration() {
    if (duration < 5000) {
        duration += 100;
        TIM6->ARR = duration;
        restart_timer();
    }
}

void process_symbol() {
    char answer[100];
    if (*cur_process_char >= '1' && *cur_process_char <= '7') {
        note_index = *cur_process_char - '1';
        sprintf(answer, "Нота: %s, октава: %d, частота: %d",
note_name[note_index], octave+1, TIM1->PSC);
        write(answer);
        play(note_index);
    } else {
        switch (*cur_process_char) {
        case '+':
            octave++;
            if (octave > 8) octave = 8;
            sprintf(answer, "Октава: %d", octave+1);
            write(answer);
            break;
        case '-':
            if (octave != 0) {
                octave--;
            }
            sprintf(answer, "Октава: %d", octave+1);
            write(answer);
            break;
        case 'a':
            duration_decrease();
            sprintf(answer, "Длительность: %d мс", duration);
            write(answer);
            break;
        case 'A':
            duration_increase();
```

```
            sprintf(answer, "Длительность: %d мс", duration);
            write(answer);
            break;
        case '\r':
            start_playing();
            sprintf(answer, "Все!", duration);
            write(answer);
            break;
        default:
            sprintf(answer, "Неверный символ %u", *cur_process_char);
            write(answer);
            break;
        }
    }
    next(&cur_process_char, read_buffer);
}

void HAL_UART_RxCpltCallback(UART_HandleTypeDef *huart) {
        if (huart->Instance == huart6.Instance) {
                next(&cur_read_char, read_buffer);
                HAL_UART_Receive_IT(&huart6, (uint8_t*) cur_read_char,
sizeof(char));
        }
}

void HAL_UART_TxCpltCallback(UART_HandleTypeDef *huart) {
        if (huart->Instance == huart6.Instance) {
                is_writing_now = 0;
                next(&transmit_from_pointer, write_buffer);
        }
}

void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
        if (htim->Instance == TIM6) {
                mute();
                if (is_all_playing){
                        note_index++;
                        if (note_index < oct_size)
                                play(note_index);
                        else
                                is_all_playing = 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();

  /* USER CODE BEGIN Init */

  /* USER CODE END Init */

  /* Configure the system clock */
  SystemClock_Config();

  /* USER CODE BEGIN SysInit */

  /* USER CODE END SysInit */
```

```
  /* Initialize all configured peripherals */
  MX_GPIO_Init();
  MX_TIM1_Init();
  MX_TIM6_Init();
  MX_USART6_UART_Init();
  /* USER CODE BEGIN 2 */

  HAL_TIM_PWM_Start(&htim1, TIM_CHANNEL_1);
  HAL_TIM_Base_Start_IT(&htim6);
  TIM6->ARR = duration;
  HAL_UART_Receive_IT(&huart6, (uint8_t *) cur_read_char, sizeof( char ));
  /* USER CODE END 2 */

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

    /* USER CODE BEGIN 3 */
            if (cur_process_char != cur_read_char) {
                process_symbol();
            }
            if(is_writing_now == 0){
                if(transmit_from_pointer != write_to_pointer) {
                    is_writing_now = 1;
                    HAL_UART_Transmit_IT( &huart6, (uint8_t *)
transmit_from_pointer, sizeof( char ));
                }
            }
        }
  /* USER CODE END 3 */
}
```

Вывод

Во время выполнения лабораторной работы мы:

- изучили работу таймеров в STM32
- применили знания на практике, разработав программу, использующую таймеры