УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Проектирование вычислительных систем»

**Лабораторная работа №3**

Вариант 11

Студент

*Митрофанов Е. Ю.*

*Любкин А. С.*

*P34101*

Преподаватель

*Пинкевич В. Ю.*

Санкт-Петербург, 2022 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):

Изображение выглядит как стол

Автоматически созданное описание

Изображение выглядит как стол

Автоматически созданное описание

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.

Исходный код

*Объявление переменных*

const uint32\_t oct\_size = 7;  
uint32\_t freqs[] = {  
 0, 0, 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, 0, 0, 0, 0  
};  
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;

*Вспомогательные функции*

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);  
 //str\_with\_newlines = concat(str\_with\_newlines, "\r\n");  
 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\_by\_index(uint32\_t index) {  
 return freqs[index + (octave \* oct\_size)];  
}  
  
void play(uint32\_t index) {  
 int freq = get\_frequency\_by\_index(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, "There is no note: %s in octave: %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 duration\_decrease(){  
 if (duration > 100) {  
 duration -= 100;  
 TIM6->ARR = duration\*10;  
 restart\_timer();  
 }  
}  
  
void duration\_increase() {  
 if (duration < 5000) {  
 duration += 100;  
 TIM6->ARR = duration\*10;  
 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, "Note: %s, octave: %d", note\_name[note\_index], octave+1);  
 write(answer);  
 play(note\_index);  
 } else {  
 switch (\*cur\_process\_char) {  
 case '+':  
 octave++;  
 if (octave > 8) octave = 8;  
 sprintf(answer, "New octave is: %d", octave+1);  
 write(answer);  
 break;  
 case '-':  
 if (octave != 0) {  
 octave--;  
 }  
 sprintf(answer, "New octave is: %d", octave+1);  
 write(answer);  
 break;  
 case 'a':  
 duration\_decrease();  
 sprintf(answer, "New duration is: %d millis", duration);  
 write(answer);  
 break;  
 case 'A':  
 duration\_increase();  
 sprintf(answer, "New duration is: %d millis", duration);  
 write(answer);  
 break;  
 case '\r':  
 start\_playing();  
 break;  
 default:  
 sprintf(answer, "Incorrect symbol %u", \*cur\_process\_char);  
 write(answer);  
 break;  
 }  
 }  
 next(&cur\_process\_char, read\_buffer);  
 }

*Main функция*

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\_USART6\_UART\_Init();  
 MX\_TIM1\_Init();  
 MX\_TIM6\_Init();  
 /\* USER CODE BEGIN 2 \*/  
 HAL\_TIM\_PWM\_Start(&htim1, TIM\_CHANNEL\_1);  
 HAL\_TIM\_Base\_Start\_IT(&htim6);  
 TIM6->ARR = duration\*10;  
 HAL\_UART\_Receive\_IT(&huart6, (uint8\_t \*) cur\_read\_char, sizeof( char ));  
 /\* USER CODE END 2 \*/  
  
 /\* Infinite loop \*/  
 /\* USER CODE BEGIN WHILE \*/  
 while (1)  
 {  
 //HAL\_UART\_Receive\_IT(&huart6, (uint8\_t \*) s, sizeof( char ));  
 //HAL\_UART\_Transmit\_IT( &huart6, (uint8\_t \*) s, sizeof( s ));  
 /\* 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 и применили полученные знания на практике, разработав программу, использующую таймеры и аппаратные каналы ввода – вывода таймеров.