УНИВЕРСИТЕТ ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки 09.03.04 Программная инженерия

Дисциплина «Проектирование вычислительных систем»

**Лабораторная работа №4**

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 интерфейс I2C для считывания нажатий

кнопок клавиатуры стенда SDK-1.1.

Подсистема опроса клавиатуры должна удовлетворять следующим требованиям:

* реализуется защита от дребезга
* нажатие кнопки фиксируется сразу после того, как было обнаружено, что кнопка нажата (с учетом защиты от дребезга), а не в момент отпускания кнопки; если необходимо, долгое нажатие может фиксироваться отдельно
* кнопка, которая удерживается дольше, чем один цикл опроса, не считается повторно нажатой до тех пор, пока не будет отпущена (нет переповторов)
* распознается и корректно обрабатывается множественное нажатие (при нажатии более чем одной кнопки считается, что ни одна кнопка не нажата, если это не противоречит требованиям к программе)
* всем кнопкам назначаются коды от 1 до 12 (порядок на усмотрение исполнителей).

Программа должна иметь два режима работы, переключение между которыми

производится по нажатию кнопки на боковой панели стенда:

* режим тестирования клавиатуры
* прикладной режим.

Уведомление о смене режима выводится в UART.

В режиме тестирования клавиатуры программа выводит в UART коды нажатых кнопок.

В прикладном режиме программа обрабатывает нажатия кнопок и выполняет действия в соответствии с вариантом задания.

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*

#define KB\_I2C\_ADDRESS (0xE2)  
#define KB\_I2C\_READ\_ADDRESS ((KB\_I2C\_ADDRESS) | 1)  
#define KB\_I2C\_WRITE\_ADDRESS ((KB\_I2C\_ADDRESS) & ~1)  
#define KB\_INPUT\_REG (0x0)  
#define KB\_OUTPUT\_REG (0x1)  
#define KB\_CONFIG\_REG (0x3)  
#define KB\_KEY\_DEBOUNCE\_TIME (200)  
  
#define ROW1 0xFE  
#define ROW2 0xFD  
#define ROW3 0xFB  
#define ROW4 0xF7  
  
const uint16\_t row\_size = 3;  
uint32\_t keys[] = {  
 '1', '2', '3',  
 '4', '5', '6',  
 '7', '+', '-',  
 'A', 'a', '\r'  
};  
const uint32\_t oct\_size = 7;  
uint64\_t freqs[] = {  
 16350, 18475, 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, 5479600, 6272000, 6880000, 7202000  
};  
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;  
  
uint32\_t key\_press\_delay = 200;  
uint32\_t side\_press\_delay = 250;  
uint32\_t last\_press\_time = 0;  
uint32\_t last\_side = 0;  
  
uint8\_t is\_test = 0;  
  
int32\_t last\_index = -1;

*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(uint64\_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);  
 set\_frequency(freq);  
 restart\_timer();  
 unmute();  
}  
  
void start\_playing() {  
 is\_all\_playing = 1;  
 note\_index = 0;  
 char answer[100];  
 sprintf(answer, "Note: %s, octave: %d", note\_name[note\_index], octave+1);  
 write(answer);  
 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 switch\_mode() {  
 if (is\_test) {  
 is\_test = 0;  
 } else is\_test = 1;  
}  
  
void check\_button() {  
 uint32\_t now = HAL\_GetTick();  
 if ((now - last\_side) > side\_press\_delay) {  
 if (HAL\_GPIO\_ReadPin(GPIOC,GPIO\_PIN\_15) == 0) {  
 last\_side = now;  
 switch\_mode();  
 if (is\_test) {  
 write("Режим тестирования включен");  
 } else write("Режим тестирования выключен");  
 }  
 }  
}  
  
void process\_symbol(char current) {  
 char answer[100];  
 if (current >= '1' && current <= '7') {  
 note\_index = current - '1';  
 sprintf(answer, "Note: %s, octave: %d", note\_name[note\_index], octave+1);  
 write(answer);  
 play(note\_index);  
 } else {  
 switch (current) {  
 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", current);  
 write(answer);  
 break;  
 }  
 }  
}  
  
  
HAL\_StatusTypeDef init\_kb( void ) {  
 HAL\_StatusTypeDef ret = HAL\_OK;  
 uint8\_t buf;  
  
 buf = 0;  
 ret = HAL\_I2C\_Mem\_Write(&hi2c1, KB\_I2C\_WRITE\_ADDRESS, 0x02, 1, &buf, 1, 100);  
 if( ret != HAL\_OK ) {  
 return ret;  
 }  
  
 buf = 0;  
 ret = HAL\_I2C\_Mem\_Write(&hi2c1, KB\_I2C\_WRITE\_ADDRESS, 0x01, 1, &buf, 1, 100);  
  
 return ret;  
}  
  
int get\_index() {  
 int index = -1;  
 uint8\_t reg\_buffer = ~0;  
 for (int row = 0; row < 4; row++) {  
 uint8\_t buf = ~((uint8\_t) (1 << row));  
  
 int16\_t n\_key = 0x00;  
 HAL\_StatusTypeDef ret = HAL\_OK;  
  
 ret = init\_kb();  
  
 ret = HAL\_I2C\_Mem\_Write(&hi2c1, KB\_I2C\_WRITE\_ADDRESS, 0x03, 1, &buf, 1, 100);  
 HAL\_Delay(10);  
 ret = HAL\_I2C\_Mem\_Read(&hi2c1, KB\_I2C\_READ\_ADDRESS, 0x00, 1, &buf, 1, 100);  
  
 uint8\_t mask = 0x7;  
 n\_key = (~(buf>>4)) & mask;  
  
 switch (n\_key) {  
 case 0x1:  
 if (index != -1) return -1;  
 index = row\*row\_size;  
 break;  
 case 0x2:  
 if (index != -1) return -1;  
 index = (row\*row\_size) + 1;  
 break;  
 case 0x4:  
 if (index != -1) return -1;  
 index = (row\*row\_size) + 2;  
 break;  
 }  
 }  
 return index;  
}  
  
int32\_t get\_index\_checked() {  
 const uint32\_t t = HAL\_GetTick();  
 if (t - last\_press\_time < key\_press\_delay) return -1;  
 int16\_t ind = get\_index();  
 if (ind != last\_index) {  
 last\_index = ind;  
 if (ind != -1) {  
 last\_press\_time = t;\  
 }  
 return ind;  
 }  
 return -1;  
}

*Main функция*

HAL\_TIM\_PWM\_Start(&htim1, TIM\_CHANNEL\_1);  
HAL\_TIM\_Base\_Start\_IT(&htim6);  
TIM6->ARR = duration\*10;

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 \*/  
 check\_button();  
 int ind = get\_index\_checked();  
 if (ind != -1) {  
 if (is\_test) {  
 char answer[100];   
 sprintf(answer, "%d", ind+1);  
 write(answer);  
 }  
 last\_index = ind;  
 char cur = keys[ind];  
 process\_symbol(cur);  
 }  
 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 ));  
 }  
 }  
}

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) {  
 if (TIM1->CCR1 > 0) {  
 note\_index++;  
 if (note\_index < 7 && is\_all\_playing) {  
 play(note\_index);  
 char answer[100];  
 sprintf(answer, "Note: %s, octave: %d", note\_name[note\_index], octave+1);  
 write(answer);  
 } else {  
 note\_index = 0;  
 is\_all\_playing = 0;  
 mute();  
 }  
 }  
 }  
  
}

Вывод

Во время выполнения лабораторной работы мы изучили работу интерфейса I2C и особенности передачи данных через него. Также мы получили навыки программирования для контроллера интерфейса и разработали программу, использующую его для считывания нажатий клавиатуры.