Московский государственный технический университет им. Н. Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»

Отчет по лабораторной работе №4

по курсу «ЭУ АСОИУ»

«Формирование мелодии с помощью микроконтроллера ATmega16»

Москва, 2020 г.

Цель работы.

Изучить основные приёмы формирования упорядоченных звуковых сигналов в виде простейшей мелодии с помощью микроконтроллера ATmega16.

Освоить методику построения на основе микроконтроллера ATmega16 систем формирования упорядоченных звуковых сигналов , работающих как в как автоколебательном, так и в ждущем режимах.

Научиться создавать программы на языке ассемблера и/или С# для ATmega16 с использованием подпрограмм.

1. Задание лабораторной работы – нотная запись мелодии.



2. Последовательность частот и длительностей нот в мелодии.

7 нот.

Первой октавы: соль, ми, фа, ля, си.

Второй октавы: ре, до.

Мелодия не содержит более 8 нот → дешифратор не используется.

Мелодия в ритме 4 четверти.

Такт первый: соль 1/4, ми 1/8, фа 1/8, соль 1/4, ля 1/8, фа 1/8.

Такт второй: соль 3/4, пауза 1/4.

Такт третий: си 1/2, си 1/4, ре 2 октавы 1/8, си 1/8.

Такт четвертый: до 2 октавы 3/4, пауза 1/4.

Такт пятый: до 2 октавы 1/4, си 1/8, до 2 октавы 1/8, ре 2 октавы 1/8, до 2 октавы 1/8, си 1/8, ля 1/8.

Такт шестой: си 1/4, ля 1/4, соль 1/2.

Такт седьмой: ля 1/4, соль 1/4, фа 1/4, соль 1/8, фа 1/8.

Такт восьмой: ми 3/4, пауза 1/4.

Далее - повторение тактов с пятого по восьмой.

3. Программа для микроконтроллера ATMEGA16, реализующая заданную мелодию.

#include <avr/io.h>

#include <util/delay.h>

#define h0 0b00000000

#define n1 0b00000001

#define n2 0b00000010

#define n3 0b00000100

#define n4 0b00001000

#define n5 0b00010000

#define n6 0b00100000

#define n7 0b01000000

void delay(int duration)

{

for (int k=0; k<duration; k++)

{

delay\_ms(1);

}

}

void play\_note(const int note\_port, double duration, \_Bool flag)

{

PORTB = note\_port;

delay(duration);

if(flag)

{

PORTB = h0;

}

}

void repeat\_sequence()

{

play\_note(n7, 800, 1);

play\_note(n5, 400, 1);

play\_note(n7, 400, 1);

play\_note(n6, 400, 1);

play\_note(n7, 400, 1);

play\_note(n5, 400, 1);

play\_note(n4, 400, 1);

play\_note(n5, 800, 0);

play\_note(n4, 800, 1);

play\_note(n1, 1600, 1);

play\_note(n4, 800, 0);

play\_note(n1, 800, 1);

play\_note(n3, 800, 1);

play\_note(n1, 400, 1);

play\_note(n3, 400, 1);

play\_note(n2, 2400, 1);

delay(800);

}

int main(void)

{

DDRB = 0xFF;

PORTD = h0;

PORTB = h0;

while (1)

{

play\_note(n1, 800, 0);

play\_note(n2, 400, 1);

play\_note(n3, 400, 1);

play\_note(n1, 800, 1);

play\_note(n4, 400, 1);

play\_note(n3, 400, 1);

play\_note(n1, 2400, 1);

delay(800);

play\_note(n5, 1600, 0);

play\_note(n5, 800, 1);

play\_note(n6, 400, 1);

play\_note(n5, 400, 1);

play\_note(n7, 2400, 1);

delay(800);

repeat\_sequence();

repeat\_sequence();

}

}

4. Функционирование разработанного генератора продемонстрировано в программе PROTEUS.

