МИНОБРНАУКИ РОССИИ

Сарапульский политехнический институт (филиал)

федерального государственного бюджетного образовательного учреждения

высшего образования

"Ижевский государственный технический университет имени М.Т. Калашникова"

(СПИ (филиал) ФГБОУ ВО “ИжГТУ имени М.Т. Калашникова”)

УТВЕРЖДАЮ

Директор филиала

\_\_\_\_\_\_\_\_\_\_\_\_\_ Г.В. Миловзоров \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2019г.

**Микропроцессорные устройства и ПЛИС**

Методические указания

к выполнению лабораторных работ по дисциплине «Микропроцессорные устройства и ПЛИС»

для студентов направления 11.03.03 «Конструирование и технология электронных средств»

всех форм обучения

Сарапул 2022

**Лабораторная работа №1**

**Изучение структуры ядра микроконтроллера *AVR ATmega*16 и работы его портов ввода-вывода**

**Цель работы:** ознакомление со структурой ядра микроконтроллера *ATmega*16 и получение навыков управления его портами ввода-вывода.

**Краткие теоретические сведения.** Ядро микроконтроллеров *AVR* выполнено по усовершенствованной *RISC*-архитектуре (*enhanced RISC*). Арифметико-логическое устройство (АЛУ), выполняющее все вычисления, подключено непосредственно к 32 рабочим регистрам, объединенным в регистровый файл. Благодаря этому АЛУ выполняет одну операцию (чтение содержимого регистров, выполнение операции и запись результата обратно в регистровый файл) за один машинный цикл. Практически каждая из команд (за исключением команд, у которых одним из операндов является 16-разрядный адрес) занимает одну ячейку памяти программ.

В микроконтроллерах *AVR* реализована Гарвардская архитектура, которая характеризуется раздельной памятью программ и данных, каждая из которых имеет собственные шины доступа к ним. Такая организация позволяет одновременно работать как с памятью программ, так и с памятью данных. Разделение шин доступа позволяет использовать для каждого типа памяти шины различной разрядности, причем способы адресации и доступа к каждому типу памяти также различны.

Еще одним решением, направленным на повышение быстродействия, является использование технологии конвейеризации. Конвейеризация заключается в том, что во время исполнения текущей команды производится выборка из памяти и дешифрация кода следующей команды. Причем длительность машинного цикла микроконтроллеров *AVR* составляет всего один период тактового генератора.

Структурная схема микроконтроллера *ATmega*16 представлена на рис 1.

Файл регистров быстрого доступа содержит 32 8-разрядных рабочих регистра общего назначения связанных непосредственно с *ALU*. За один тактовый цикл из файла регистров выбираются два операнда, выполняется операция и результат вновь возвращается в файл регистров.

Шесть из 32 регистров могут быть использованы как три 16-разрядных регистра указателя косвенной адресации адресного пространства данных, обеспечивающие эффективное вычисление адресов. Один из этих указателей адреса используется, также, как указатель адреса для функции непрерывного просмотра таблиц. Эти 16-разрядные дополнительные регистры обозначаются как X-регистр, Y-регистр и Z-регистр.

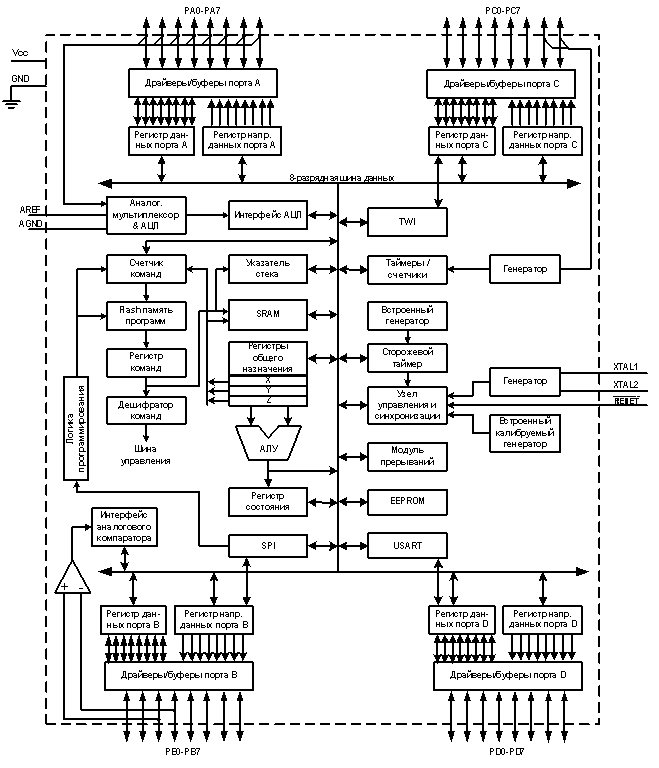


Рисунок 1 – Структурная схема микроконтроллера ATmega16

В дополнение к операциям с регистрами, регистровый файл может использоваться и для обычной адресации памяти. Это объясняется тем, что файл регистров располагается по 32 младшими адресами пространства данных, и к ним можно обращаться, как к обычным ячейкам памяти.

Пространство памяти I/O содержит 64 адреса периферийных функций *CPU* таких как: регистры управления, таймеры/счетчики, аналого-цифровые преобразователи и другие I/O функции. К памяти I/O можно обращаться непосредственно или как к ячейкам пространства памяти соответствующим адресам файла регистров $20 – $5F.

В процессе обработки прерываний и вызовов подпрограмм адрес возврата счетчика команд (*PC*) сохраняется в стеке. Стек размещается в *SRAM* данных. Все пользовательские программы в подпрограммах возврата (прежде, чем подпрограммы или прерывания будут выполняться) должны инициализировать указатель стека (*SP*).

*AVR* архитектура поддерживает пять различных режимов адресации байт *SRAM* данных.

Гибкий модуль обработки прерываний имеет в пространстве I/O свой управляющий регистр с дополнительным битом разрешения глобального прерывания в регистре статуса. Все прерывания имеют свои векторы прерывания в таблице векторов прерывания, располагаемой в начале памяти программ. Приоритеты прерываний соответствуют положению векторов прерываний - прерывание с наименьшим адресом вектора имеет наивысший приоритет [1].

МК имеет 4 порта ввода-вывода: порт A, порт В, порт С и порт D. Для обслуживания каждого порта отведено три регистра: регистр данных – PORTx, регистр направления данных – DDRx и регистр вывода порта – PINx. Регистры PINx предназначены только для чтения, а регистры PORTx и DDRx для чтения и записи. Каждый вывод (pin) порта работает автономно от других выводов. Внутренние электронные схемы каждого вывода могут управлять подключенными к ним светодиодами.

Перед использованием портов в программе их необходимо настроить, то есть определить направление (ввод или вывод) отдельного бита порта и задать начальное значение. Для того чтобы использовать вывод для ввода данных, необходимо соответствующий бит регистра DDR сбросить и тот же бит регистра PORT установить. Например, нужно настроить шестой вывод порта D на ввод данных. На языке Си фрагмент кода для этого будет выглядеть так:

DDRD&=0b10111111;

PORTD|=0b01000000;

Если нужно настроить на ввод все выводы порта D:

DDRD = 0;

PORTD = 0хFF;

Для настройки порта на вывод данных необходимо соответствующие биты регистра DDR этого порта установить и эти же биты регистра PORT сбросить. Например, настроить все выводы порта B на вывод данных:

DDRB = 0xFF;

PORTB = 0;

После настройки порта на вывод данных, в дальнейшем сам процесс вывода данных осуществляется следующим образом.

PORTD = 0; // сбросить все выводы порта D

PORTD = 0xFF; // установить все выводы порта D (установить на них высокий уровень напряжения 5 V).

*Побитовые операции и маскирование данных.* Маскирование данных позволяет выполнять операции над отдельными битами регистра при наложении маски на слово данных. При этом используются операции побитового И (&), ИЛИ (|), XOR (^).

Пример 1. Прочитать данные с порта С и, сохранив значения битов 0…2, 4, 5, установить в логический «0» биты 3, 6, 7 и записать результат в регистр PORTD:

PORTD = PINC & 0b00110111; // маска 0b00110111

Пример 2. Прочитать данные с порта С и, не изменяя содержимого других разрядов, установить в разрядах 1, 3 и 5 лог. «1» и записать результат в регистр PORTD:

PORTD = PINC | 0b00101010; // маска 0b00101010

Операторы сдвига >> и << сдвигают биты в переменной вправо и влево соответственно, на указанное число. Общий вид оператора сдвига вправо:

переменная >> число сдвигов

а общий вид оператора сдвига влево:

переменная << число сдвигов

При сдвиге биты, сдвигающиеся на один конец, не появляются с другого. Сдвинутые биты теряются, а с другого конца появляются нули.

**Задание для лабораторной работы.** Закрепление навыков работы с портами ввода-вывода удобно производить посредством схемы, представленной на рисунке 2.

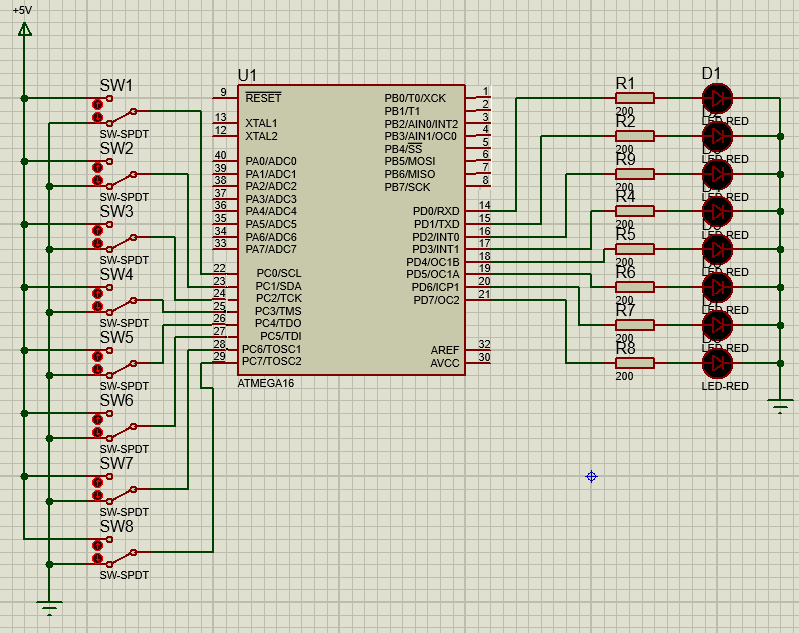


Рисунок 2 – Электрическая схема соединения светодиодов и тумблеров с портами ввода-вывода МК

К выводам порта C PC0…PC7 подключены тумблеры SW1…SW8 соответственно. К выводам порта D - светодиоды через токоограничивающие резисторы. При включении тумблера на соответствующем выводе порта C устанавливается высокий уровень напряжения 5 V, что соответствует логической «1», в выключенном состоянии тумблера на соответствующем выводе порта С логический «0».

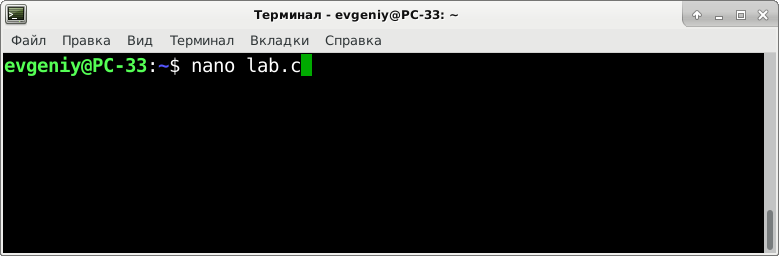
**Задание для лабораторной работы.** Написать программу работы МК в соответствии с вариантом (таблица 1).

|  |  |
| --- | --- |
| **Вариант** | **Задание** |
| 1 | При включении любого четного тумблера зажигаются только все нечетные светодиоды в линейке светодиодов, включение нечетных тумблеров игнорируется. |
| 2 | При включении любого нечетного тумблера зажигаются только все четные светодиоды в линейке светодиодов, включение четных тумблеров игнорируется. |
| 3 | При включении тумблера 0, 1, 2 и т.д. зажигается светодиод позиции, соответственно 7, 6, 5-й и т.д., то есть в обратном порядке, возможно включение нескольких тумблеров. |
| 4 | Все светодиоды в линейке зажигаются только при одновременном включении 3, 4 и 7-го тумблеров, включение других гасит светодиоды. |
| 5 | Все светодиоды в линейке зажигаются последовательно с интервалом 0,5 секунд, начиная с 0-го, при одновременном включении тумблера 0 и 7. |
| 6 | Все светодиоды в линейке зажигаются последовательно с интервалом 0,5 секунд, начиная с 7-го, при одновременном включении тумблера 0 и 3. |
| 7 | При включении тумблера 3 в линейке светодиодов зажигаются светодиоды 7, 5, 4 и сдвигаются вправо с интервалом времени одного сдвига 0,5 секунд. |
| 8 | При включении тумблера 0 в линейке светодиодов зажигаются светодиоды 0 и 1 и сдвигаются влево с интервалом времени одного сдвига 0,5 секунд. |
| 9 | При включении четных тумблеров зажигаются все нечетные и мигают с периодом 0,5 секунд, включение нечетных тумблеров гасит светодиоды. |
| 10 | При включении нечетных тумблеров зажигаются все четные и мигают с периодом 1 секунду. |

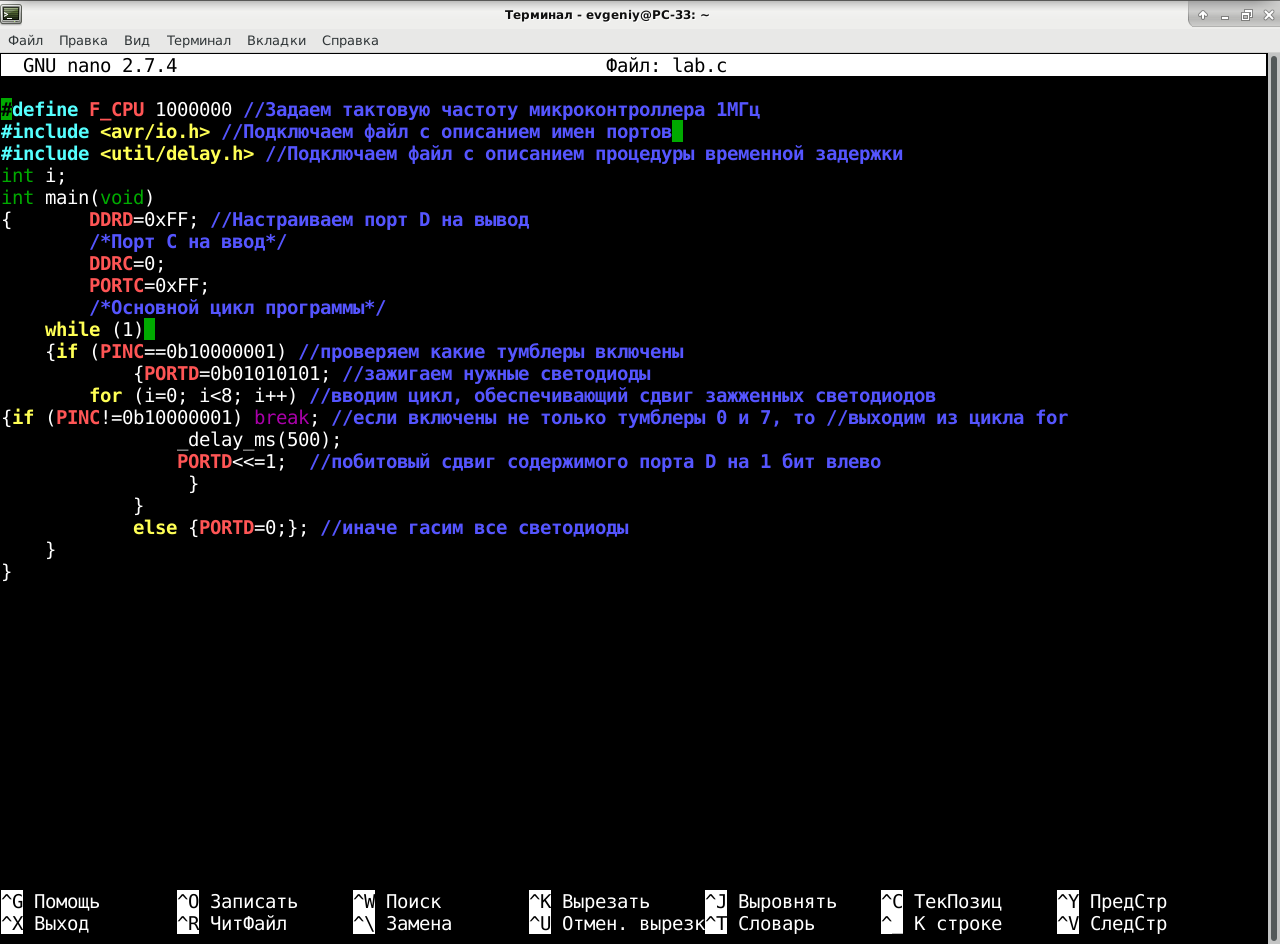
**Пример выполнения.** Задание. При включении тумблера 0 и 7 в линейке зажигаются все четные светодиоды и сдвигаются влево с временным интервалом одного сдвига 1 секунда.

Решение. В меню кнопки «Приложения» выбираем пункт «Эмулятор терминала», в появившемся окне терминала вводим команду

nano lab.c



Тем самым в текущем каталоге (в данном случае это рабочий каталог пользователя, перейти в который можно командой cd ~, а посмотреть содержимое — командой ls) создаем файл с расширением .c, который будет содержать исходный код программы. При этом имя файла может быть любым, например my\_prog.c. В открывшемся окне текстового редактора вводим текст программы:

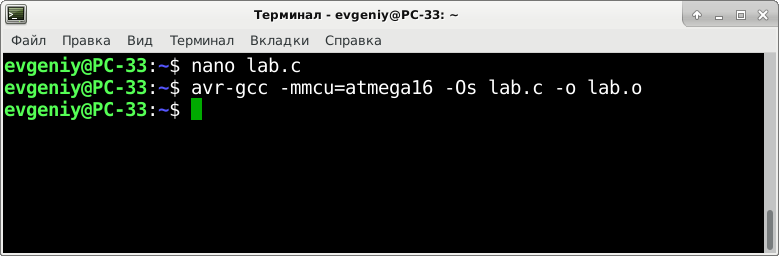


Сохраняем изменения в файле и выходим из текстового редактора, нажав комбинацию клавиш «Ctrl+x», затем «y» и далее «Enter». После этого в терминале появится новое приглашение к вводу.

Следующий этап — компиляция написанного кода программы. Для этого необходимо ввести команду:

avr-gcc -mmcu=atmega16 -Os lab.c -o lab.o

В результате работы, если нет ошибок, в текущем каталоге создается объектный файл lab.o и появляется приглашение к вводу:



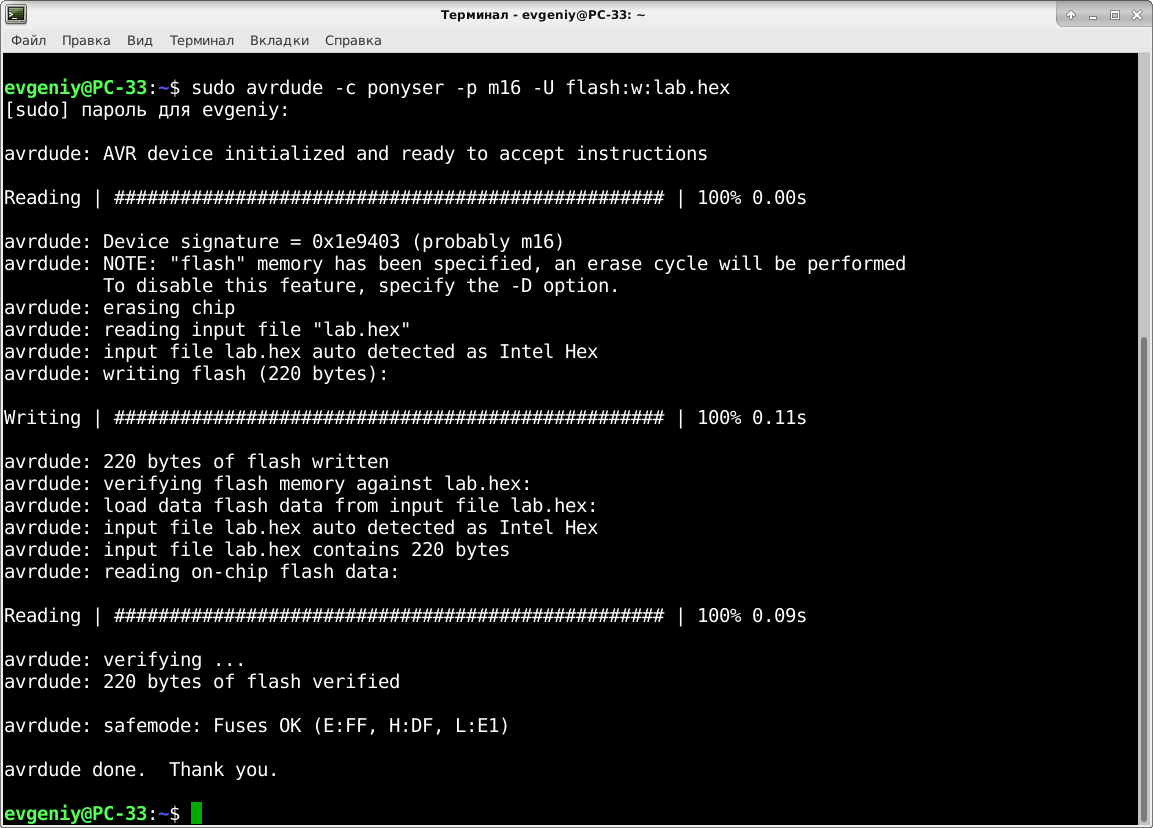
Для формирования файла прошивки в формате HEX необходимо выполнить команду:

avr-objcopy -j .text -j .data -O ihex lab.o lab.hex

Загрузка содержимого HEX-файла в память FLASH микроконтроллера осуществляется программой avrdude командой (программатор ponyprog, загрузка посредством интерфейса RS-232):

avrdude -c ponyser -p m16 -U flash:w:lab.hex

В результате успешной «прошивки» на экран будут выведены следующие сообщения:



**Литература**

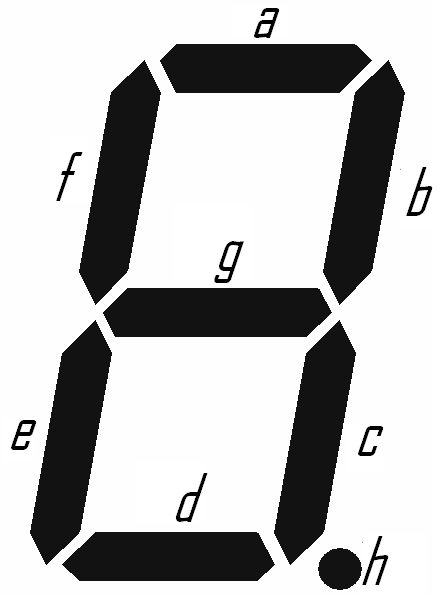
1. <http://microchipinf.com/articles/45/142>
2. <https://ph0en1x.net/80-simple-program-for-avr-microcontroller-c-language.html>

**Лабораторная работа №2**

**Организация динамической индикации**

**Цель работы:** изучение способа сопряжения МК *AVR AtMega16* со светодиодным многоразрядным индикатором

**Краткие теоретические сведения.** К устройствам сопряжения МК относятся семисегментные индикаторы с дополнительным сегментом – точкой (рис. 1), позволяющие выводить цифры, литеры и различные специальные символы.

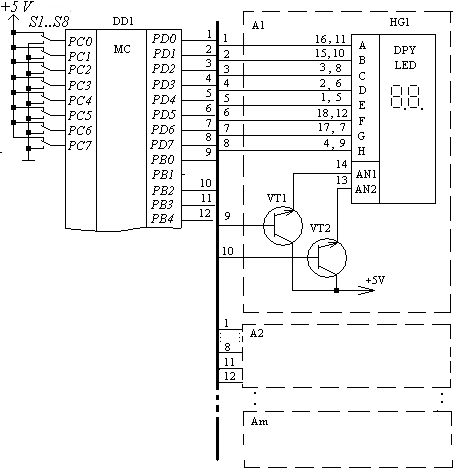


*Рис. 1.* Обозначение сегментов индикатора

Для одновременного вывода символов на несколько таких индикаторов используется динамическая индикация. При этом цифры отображаются последовательно на каждом разряде, но со скоростью, при которой человеческий глаз не замечает мигания. В этом случае расчет времени свечения одного разряда считается по формуле 1/(*N*∙50), где *N* – количество разрядов индикатора, а число 50 определяет число засвечиваний в секунду. Для отображения цифры одного разряда необходимо погасить ненужные сегменты и зажечь требуемые. Например, цифра **7** изображается так: сегменты *a,b,c* и *f* включены, при выключенных *d,e,g* и *h*. В результате получается восьмиразрядный код, который ставится в соответствие с *двоично-десятичным* кодом отображаемой цифры.

Схема подключения светодиодных индикаторов может быть как на рис. 2.

В схеме используется двухразрядный индикатор с общим анодом, например, типа КИПЦ22Б, АЛС324А и др. Для выбора засвечиваемого разряда на базы транзисторов *VT1…VT4* подаются управляющие сигналы с выводов портов *PB0…PB3.* При этом согласно схеме (рис. 2), лог. «1» включает разряд, а лог. «0» выключает. Для зажигания сегмента на соответствующий вывод портов *PD0…PD7* необходимо подать лог. «0».

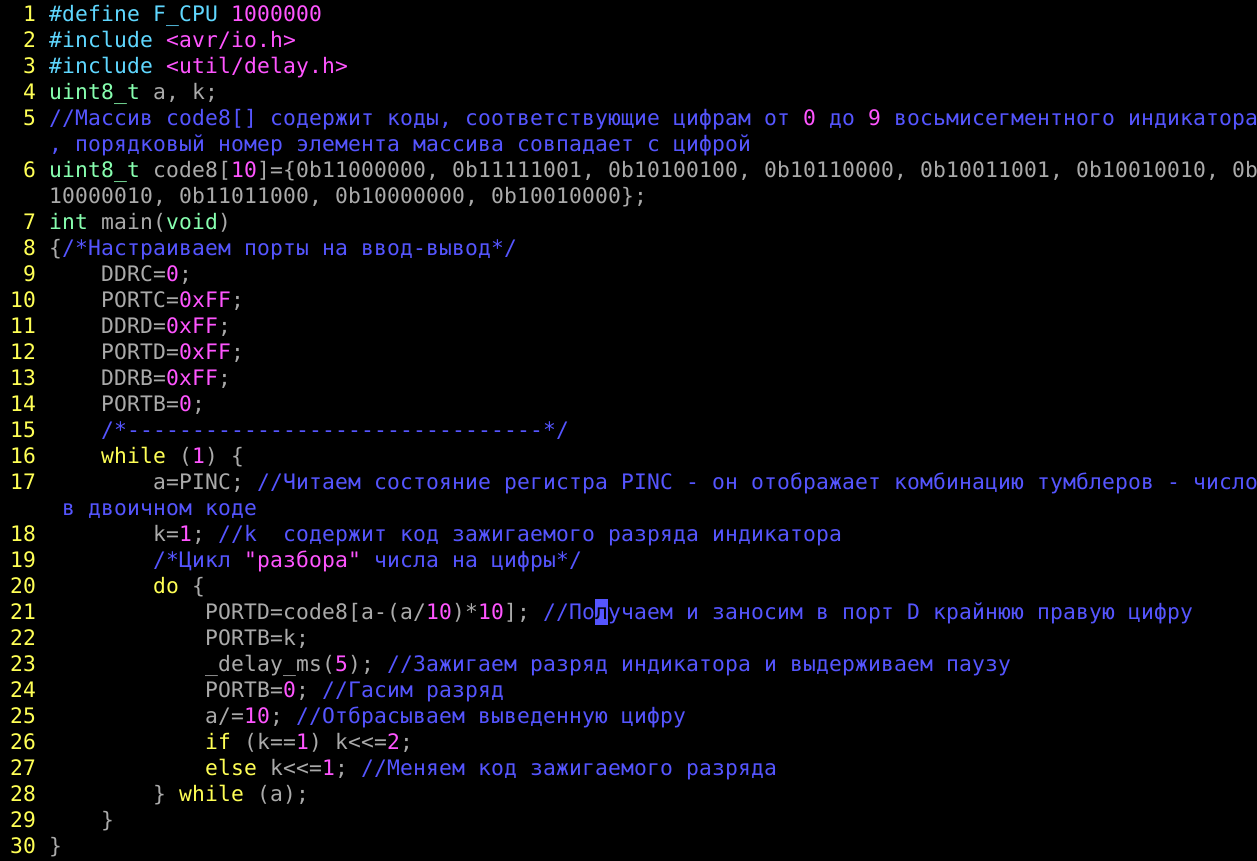


*Рис. 2.* Схема подключения тумблеров и индикаторов к МК

В соответствии со схемой (рис. 2) для каждой цифры, отображаемой на индикаторе, представим код в виде таблицы перекодировки (табл. 1).

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Таблица 1 | | | | | | | | | | |
| Представление десятичных цифр для восьмисегментного индикатора | | | | | | | | | | |
| Вывод | Цифра | | | | | | | | | |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| *PD0 (a)* | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| *PD1 (b)* | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| *PD2 (c)* | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| *PD3 (d)* | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| *PD4 (e)* | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |
| *PD5 (f)* | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| *PD6 (g)* | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| *PD6 (h)* | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Пример** программы на языке Си, осуществляющей считывание комбинации тумблеров S1..S8, представляющей собой двоичный код числа в интервале 0..255, и вывод на первые три восьмисегментных разряда, представлен ниже.



**Задание для лабораторной работы.**

|  |  |
| --- | --- |
| Вариант | Задание |
| 1 | Вывести на индикатор в двух младших разрядах число в шестнадцатиричной форме, набранное тумблерами как двоичный код. В старшие разряды индикатора выводить ноль. Символы *B* и *D* выводить как *b* и *d* |
| 2 | Вывести на индикатор в старшем (№ 3) и младшем (№ 0) разрядах число в шестнадцатиричной форме, набранное тумблерами как двоичный код. Разряды индикатора № 1 и № 2 не зажигать. Символы *B* и *D* выводить как *b* и *d* |
| 3 | Вывести на индикатор в три младших разряда результат в десятичной системе счисления разности чисел, набранных тумблерами старшей и младшей тетрад как двоичный код |
| 4 | Вывести на индикатор в младший разряд цифру в десятичной системе счисления, обозначающую порядковый номер включенного тумблера (от 0 до 7). В старшие разряды индикатора выводить нули. При одновременном включении нескольких тумблеров выводить все нули |
| 5 | Вывести на индикатор во все разряды цифру 0, мигающую со скоростью пропорционально числу включенных тумблеров |
| 6 | Выводить в соответствующем тумблеру старшей тетрады разряде индикатора нуль, тумблеры младшей тетрады игнорировать. |
| 7 | Выводить в соответствующем тумблеру младшей тетрады разряде индикатора нуль, тумблеры старшей тетрады игнорировать |
| 8 | Вывести на индикатор в два младших разряда время от 0 до 99 секунд, отсчитываемое от момента включения первого тумблера. При выключении тумблера отсчет должен заканчиваться |

**Литература**

1. Чикуров Т.Г. Применение микроконтроллеров *AVR ATMEL* в схемотехнике радиоэлектронных средств [Текст]. – Ижевск: Изд-во ИжГТУ, 2014 . – 80 с.

**Лабораторная работа №3**

**Сопряжение микроконтроллера и жидкокристаллического дисплея (ЖКД)**

**Цель работы:** изучение структуры контроллера ЖКД и его сопряжения с МК *AVR ATmega*16.

**Краткие теоретические сведения приведены в:** Чикуров Т.Г. Применение микроконтроллеров *AVR ATMEL* в схемотехнике радиоэлектронных средств [Текст]. – Ижевск: Изд-во ИжГТУ, 2014 . – 80 с. Лабораторная работа № 4.

**Задание для лабораторной работы.**

|  |  |
| --- | --- |
| Вариант | Задание |
| 1 | Вывести на ЖКД строку приветствия, содержащую не менее 10 символов, слева направо, с интервалом вывода символов 0.5 секунды |
| 2 | Вывести на ЖКД строку приветствия, содержащую не менее 10 символов, справа налево, с интервалом вывода символов 0.5 секунды |
| 3 | Заполнить ЖКД символами строки «0123456789ABCDEF». Заполнение символами должно происходить поочередно слева и справа с интервалом 1 секунда. |
| 4 | Вывести на ЖКД мигающую строку приветствия, содержащую не менее 10 символов, с началом строки слева. |
| 5 | Вывести на ЖКД мигающую строку приветствия, содержащую не менее 10 символов, с началом строки справа. |
| 6 | Вывести на ЖКД мигающую строку приветствия, содержащую не менее 10 символов, с началом строки слева. |
| 7 | Заполнить ЖКД символами строки «0123456789ABCDEF». Заполнение символами должно происходить поочередно слева и справа от центра с интервалом 1 секунда. |
| 8 | Вывести на ЖКД бегущую слева направо строку приветствия. |
| 9 | Вывести на ЖКД бегущую справа налево строку приветствия. |

**Пример выполнения задания.** Вывести последовательно цифры от 0 до 9 и буквы от A до F на ЖКД с интервалом вывода каждого символа 1 с. Листинг программы приведен ниже.

#define F\_CPU 1000000

#include <avr/io.h>

#include <util/delay.h>

int main(void) {

DDRA=0xFF;

PORTA=0;

/\*4-хбитный интерфейс, 1строка\*/

for (uint8\_t i = 0; i < 2; i++) {

PORTA=0b00101000;

PORTA&=0b11110111;

PORTA=0b11001000;

PORTA&=0b11110111;

\_delay\_us(42);

}

/\*Включить дисплей, выключить курсор\*/

PORTA=0b00001000;

PORTA&=0b11110111;

PORTA=0b11001000;

PORTA&=0b11110111;

\_delay\_us(42);

while (1) {

/\*Очистить дисплей и установить курсор в начало\*/

PORTA=0b00001000;

PORTA&=0b11110111;

PORTA=0b00011000;

PORTA&=0b11110111;

\_delay\_ms(2);

uint8\_t i = 48;//Код символа "0" в таблице CGROM

do{

if (i == 0x3A) i = 0x41;//Код символа "A"

PORTA=i & 0xF0 | 0b1100;

PORTA&=0b11110111;

PORTA=i << 4 | 0b1100;

PORTA&=0b11110111;

\_delay\_ms(1000);

/\*Если выведены 8 символов, устанавливаем адрес DDRAM 63\*/

if (i == 0x37) {

PORTA=0b10111000;

PORTA&=0b11110111;

PORTA=0b11111000;

PORTA&=0b11110111;

\_delay\_us(42);

}

i++;

} while (i < 0x47);

}

}

**Лабораторная работа №4**

**Организация работы встроенного АЦП**

**Цель работы:** ознакомиться с работой аналого-цифрового преобразователя микроконтроллера *ATmega*16.

**Краткие теоретические сведения приведены в:** Чикуров Т.Г. Применение микроконтроллеров *AVR ATMEL* в схемотехнике радиоэлектронных средств [Текст]. – Ижевск: Изд-во ИжГТУ, 2014 . – 80 с. Лабораторная работа № 5.

**Задание для лабораторной работы.**

|  |  |
| --- | --- |
| Вариант | Задание |
| 1 | На вход *PA0* подать напряжение +3 В от источника постоянного напряжения, проконтролировать его вольтметром. При опорном напряжении *U=Uvcc* измерить входное напряжение с помощью АЦП, а результат вывести на ЖКД |
| 2 | На вход *PA0* подать напряжение +2 В от источника постоянного напряжения, проконтролировать его вольтметром. При опорном напряжении *U*=2,56 В измерить входное напряжение с помощью АЦП, а результат вывести на ЖКД |
| 3 | На вход *PA0* подать напряжение +1 В от источника постоянного напряжения, проконтролировать его вольтметром. При опорном напряжении *U=Uvcc* измерить входное напряжение с помощью АЦП, а результат вывести на ЖКД |
| 4 | На вход *PA0* подать напряжение +3 В от источника постоянного напряжения, проконтролировать его вольтметром. При опорном напряжении *U=Uvcc* измерить входное напряжение с помощью АЦП, а результат вывести на семисегментный индикатор |
| 5 | На вход *PA0* подать напряжение +0,5 В от источника постоянного напряжения, проконтролировать его вольтметром. При опорном напряжении *U=Uvcc* измерить входное напряжение с помощью АЦП, а результат вывести на семисегментный индикатор |

**Пример выполнения задания.** Вывести на семисегментный индикатор значение напряжения встроенного источника питания, измеренное АЦП при опорном напряжениии *Uvcc,* с точностью 1 знак после запятой. Листинг программы приведен ниже.

#define F\_CPU 8000000

#include <avr/io.h>

#include <util/delay.h>

uint8\_t code8[10]={0b11000000,0b11111001,0b10100100,0b10110000,0b10011001,0b10010010,0b10000010,0b11011000,0b10000000,0b10010000};

int main(void)

{/\*Настраиваем порты на ввод-вывод\*/

uint16\_t U, ADres;

uint8\_t k;

DDRD=0xFF;

PORTD=0xFF;

DDRB=0xFF;

PORTB=0;

//--0b01-->Опорное AVcc

//--0-->ADC выравниваем вправо,

//--00000--канал PA0

ADMUX = 0b01000000;

ADCSRA = 0b10000110;//ADC\_ON; F\_ADC=F\_CPU/64;

/\*--------------------------------\*/

while (1) {

ADCSRA |= 0x40;

while (ADCSRA & 0x40);

ADres = ADCL;

U = ADCH;

ADres |= (U << 8);

U = (uint32\_t)ADres\*5\*100/1024;//Опорное 5V, 100-2 знака после".", 2^10=1024-разрядность ADC

if (U - (U/10)\*10 >= 5) U = U/10 + 1;//Округление

else U = U/10;

/\*Цикл "разбора" числа на цифры\*/

k = 1;

for (uint8\_t i = 0; i<2; i++) {

if (i == 1) PORTD = code8[U-(U/10)\*10] & 0b01111111;

else PORTD=code8[U-(U/10)\*10];

PORTB=k;

\_delay\_ms(5);

PORTB=0;

U/=10;

if (k==1) k<<=2;

else k<<=1;

}

}

}

**Лабораторная работа №5**

**Встроенные таймеры-счетчики микроконтроллера *AtMega*16**

**Цель работы:** ознакомиться с работой таймеров-счетчиков МК *AtMega*16 на примере часов на базе МК.

**Краткие теоретические сведения приведены в:** Чикуров Т.Г. Применение микроконтроллеров *AVR ATMEL* в схемотехнике радиоэлектронных средств [Текст]. – Ижевск: Изд-во ИжГТУ, 2014 . – 80 с. Лабораторная работа № 4.

**Задание для лабораторной работы.**

|  |  |
| --- | --- |
| Вариант | Задание |
| 1 | Разработать на базе ТС1 часы с выводом времени на ЖКД в формате ЧЧ:ММ:СС. Настройку времени производить тумблерами 0 и 1. |
| 2 | Разработать на базе ТС1 секундомер с выводом времени на семисегментный индикатор в формате ММ:СС. Запуск и останов осуществлять тумблером 0, сброс — тумблером 1. |
| 3 | Разработать на базе ТС1 секундомер с выводом времени на ЖКД в формате ММ:СС:СОТЫЕ\_ДОЛИ\_СЕКУНДЫ. Запуск и останов осуществлять тумблером 0, сброс — тумблером 7. |
| 4 | Разработать на базе ТС1 и ТС0 частотомер с выводом частоты на ЖКД. |
| 5 | Разработать на базе ТС1 и ТС0 частотомер с выводом частоты на семисегментный индикатор. |

**Пример выполнения задания.** Разработать на базе ТС1 часы с выводом времени на семисегментный индикатор в формате ЧЧ:ММ. Настройку времени производить тумблерами 0 и 1. Листинг программы приведен ниже.

#define F\_CPU 8000000

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

unsigned char code8[10]={0b11000000,0b11111001,0b10100100,0b10110000,0b10011001,0b10010010,0b10000010,0b11011000,0b10000000,0b10010000};

unsigned char hour = 0, min = 0, sec = 0, k;

int main(void)

{/\*Настраиваем порты на ввод-вывод\*/

DDRC=0;

PORTC=0xFF;

DDRD=0xFF;

PORTD=0xFF;

DDRB=0xFF;

PORTB=0;

unsigned short clk;

TCNT1 = 0;//регистр значения

TIMSK = 0b10000;//Разрешение прерывания по совпадению содержимого OCR1A и TCNT1

OCR1A = 31250;// \* F\_CPU/256 = 1 sec

TCCR1B = 0b100;//F\_CPU/256

sei();

/\*--------------------------------\*/

while (1) {

if (sec == 60) {

sec = 0;

min++;

}

if (min == 60) {

min = 0;

hour++;

}

if (hour == 24) hour = 0;

/\*---Настройка времени---\*/

if (PINC & 0x01) {

min++;

sec = 0;

\_delay\_ms(300);

}

if (PINC & 0x02) {

hour++;

\_delay\_ms(300);

}

/\*-----------------------\*/

clk = hour \* 100 + min;//"Склеиваем" часы и минуты в одно число

/\*Цикл "разбора" числа на цифры\*/

k = 1;

for (char i = 0; i<4; i++) {

PORTD=code8[clk-(clk/10)\*10]; //Получаем и заносим в порт D единицы числа

if (i == 2 && sec%2 == 0) PORTD &=0b01111111;// Миающая точка

clk/=10; /

PORTB=k;

\_delay\_ms(5);

PORTB=0;

if (k==1) k<<=2;

else k<<=1;

}

}

}

ISR (TIMER1\_COMPA\_vect) {

TCNT1 = 0;

sec++;

}

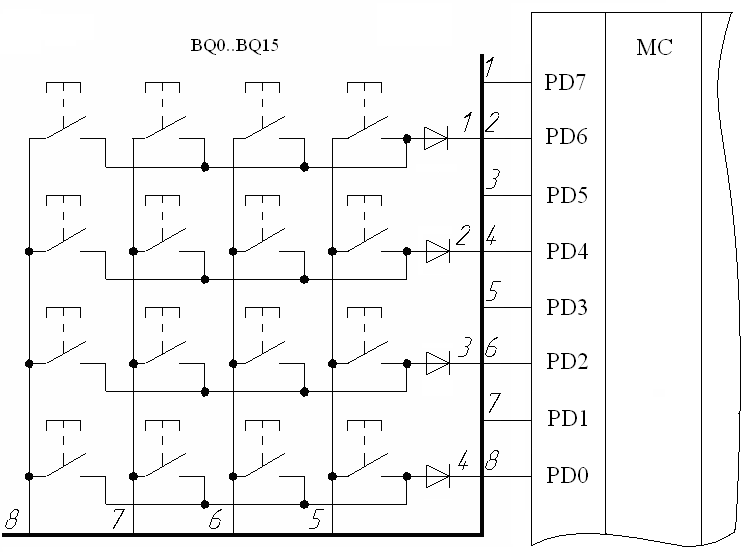
**Лабораторная работа №6**

**Сопряжение микроконтроллера и клавиатуры**

**Цель работы:** разработка интерфейса 16-клавишной клавиатуры.

**Краткие теоретические сведения.**

Электрическая схема соединения клавиатуры и микроконтроллера приведена ниже.



Клавиатуру можно представить матрицей, нажатую клавишу — ее индексом с нумерацией, соответствующей номерам проводников, объединенных в шину. Процедура идентификации нажатой клавиши заключается в организации сканирования матрицы «бегущим» нулем по линиям сканирования 1...4 — выводы порта PD4...PD7, и считывания состояния линий опроса — выводов PD0...PD3. После установки кода на линиях сканирования и перед считыванием состояния линий опроса необходимо выдержать паузу, необходимую для завершения переходного процесса, который обусловлен собственными паразитными параметрами схемы. После считывания состояния линий опроса так же необходима временная задержка для устранения исключения влияния на алгоритм работы явления, под названием «дребезг контактов». Диоды в схеме необходимы для защиты порта при нажатии более одной клавиши.

**Задание для лабораторной работы.**

|  |  |
| --- | --- |
| Вариант | Задание |
| 1 | Написать программу, реализующую функции калькулятора, выполняющего сложение, вычитание, деление и умножение целых чисел, вводимых с клавиатуры, с индикацией последних и результата действий над ними на ЖКД. Точность 1 знак после запятой, запрещается использовать тип float и double. |
| 2 | Написать программу, реализующую игру «Угадай число» (из интервала 0...10) с выводом строки приглашения к вводу и результата на ЖКД. |
| 3 | Написать программу вывода на ЖКД фиксированной строки при правильно введенном пароле, состоящем из 5 разных символов (цифр и букв из таблицы кодировки ЖКД), введенных с клавиатуры. |
| 4 | Написать программу вывода на ЖКД бегущей строки слева направо, состоящей из фамилий выполняющих вариант задания при правильно введенном коде, состоящем из 5 разных цифр (из таблицы кодировки ЖКД), введенных с клавиатуры. |
| 5 | Написать программу вывода на ЖКД бегущей строки справа налево, состоящей из фамилий выполняющих вариант задания при правильно введенном слове, состоящем из 5 букв (из таблицы кодировки ЖКД), введенных с клавиатуры. |

**Пример выполнения задания.** Написать программу вывода на ЖКД шестнадцати символов из таблицы кодировки ЖКД (начиная с «0»), вводимых с клавиатуры. По заполнению всех позиций и нажатия следующей клавиши дисплей должен очищаться, символ нажатой клавиши выводиться в начальной позиции. Листинг программы приведен ниже.

#define F\_CPU 8000000

#include <avr/io.h>

#include <util/delay.h>

int main (void){

DDRA = 0xFF;

PORTA = 0;

DDRD = 0xF0;

PORTD = 0x0F;

uint8\_t scan = 0x10, count\_num = 0, code;

uint8\_t num[16] = {

0b00011110, 0b00011101, 0b00011011, 0b00010111,

0b00101110, 0b00101101, 0b00101011, 0b00100111,

0b01001110, 0b01001101, 0b01001011, 0b01000111,

0b10001110, 0b10001101, 0b10001011, 0b10000111

};

/\*4-хбитный интерфейс, 1строка\*/

for (uint8\_t i = 0; i < 2; i++) {

PORTA=0b00101000;

PORTA&=0b11110111;

PORTA=0b11001000;

PORTA&=0b11110111;

\_delay\_us(42);

}

/\*Включить дисплей, выключить курсор\*/

PORTA=0b00001000;

PORTA&=0b11110111;

PORTA=0b11001000;

PORTA&=0b11110111;

\_delay\_us(42);

/\*Очистить дисплей и установить курсор в начало\*/

PORTA=0b00001000;

PORTA&=0b11110111;

PORTA=0b00011000;

PORTA&=0b11110111;

\_delay\_ms(2);

while (1) {

while (1) {

PORTD = ~scan;

\_delay\_ms(1);

code = PIND & 0x0F;

if (code != 0x0F) {

\_delay\_ms(300);

break;

}

scan <<= 1;

if (!scan) scan = 0x10;

}

code = scan | code;

for (uint8\_t i = 0; i < 16; i++) {

if (num[i] != code) continue;

if (count\_num == 16) {

/\*Очистить дисплей и установить курсор в начало\*/

PORTA=0b00001000;

PORTA&=0b11110111;

PORTA=0b00011000;

PORTA&=0b11110111;

\_delay\_ms(2);

count\_num = 0;

}

/\*Выводим символ клавиши\*/

i += 48;

PORTA = i & 0xF0 | 0b1100;

PORTA &= 0b11110111;

PORTA = i << 4 | 0b1100;

PORTA &= 0b11110111;

\_delay\_us(41);

break;

}

count\_num++;

/\*Если выведены 8 символов, устанавливаем адрес DDRAM 63\*/

if (count\_num == 8) {

PORTA=0b10111000;

PORTA&=0b11110111;

PORTA=0b11111000;

PORTA&=0b11110111;

\_delay\_us(42);

}

}

}

**Лабораторная работа №7**

**Аппаратная организация широтно-импульсной модуляции**

**Цель работы:** изучение работы таймеров/счетчиков МК *AtMega16* в режиме ШИМ.

**Краткие теоретические сведения и задания для лабораторной работы приведены в:** Чикуров Т.Г. Применение микроконтроллеров *AVR ATMEL* в схемотехнике радиоэлектронных средств [Текст]. – Ижевск: Изд-во ИжГТУ, 2014 . – 80 с. Лабораторная работа № 7.