МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ
«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ имени
В.И.ВЕРНАДСКОГО»
ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ
Кафедра компьютерной инженерии и моделирования

Сосновский Ю.В.

Методические указания к выполнению лабораторных работ по курсу «Микропроцессорные системы»

для студентов 3 курса дневной формы обучения
направления подготовки 09.03.01 «Информатика и
вычислительная техника»
образовательно-квалификационного уровня «бакалавр»

Симферополь, 2020

Рекомендовано к печати заседанием кафедры
от 15.01.2020 г. № _5__

Рекомендовано к печати:

Учебно-методическим советом Физико-технического института (структурное подразделение) ФГАОУ ВО «КФУ им. В.И.Вернадского»

протокол № __5__ от 24.01.2020

Составитель (автор): Сосновский Юрий Вячеславович, к.т.н., доцент кафедры компьютерной инженерии и моделирования

Содержание

Введение

Курс «Микропроцессорные системы» является дисциплиной по выбору учебного заведения и преподается студентам 3 курса направления подготовки 09.03.01 «Информатика и вычислительная техника».

Целью курса является формирование у студентов знания общей методологии и конкретных методов проектирования основных разновидностей современных микропроцессорных средств, а также знаний и умений в области архитектуры, принципов функционирования и программирования микропроцессорных систем.

Предусмотренные учебной программой лабораторно – практические занятия служат для закрепления теоретических знаний и получения навыков работы с микропроцессорными системами. Для лучшего усвоения и закрепления знаний в ходе выполнения лабораторно-практических работ целесообразно разделение студентов на подгруппы.

Задачи преподавания дисциплины «Микропроцессорные системы» – дать студентам систематизированные сведения об архитектуре современных микропроцессоров – микроЭВМ на одном кристалле (микроконтроллеров), научить основам работы в специализированных средах программирования и симуляции микроконтроллеров, а также дать практические навыки в программировании микропроцессорных систем.

В результате изучения курса студент должен знать:

– основные принципы архитектурного построения микропроцессоров, микроЭВМ на одном кристалле – микроконтроллеров;

– основные архитектуры микропроцессоров – гарвардская, фон-Неймановская (принстонская), понимать их преимущества и недостатки, уметь определять для каждой целевую область использования;

– основные блоки периферии современных микроконтроллеров;

– принципы написания управляющей программы микроконтроллера, отладка и симуляция ее в специальных программных средах, программирование (прошивку) микроконтроллера с использованием программатора.

В результате изучения курса студент должен уметь:

– читать и понимать информацию, представленную в технической документации на микроконтроллеры семейства ATMega;

– создавать управляющую программу (прошивку) базовой сложности;
– программно использовать периферию микроконтроллера;
– выполнять отладку и симуляцию управляющей программы в виртуальном микроконтроллере в специализированном программном комплексе;
– осуществлять прошивку микроконтроллера.

Данные методические указания предназначаются для оказания учебно-методической поддержки курса «Микропроцессорные системы» и являются неотъемлемым дополнением к лекционному материалу курса, а также ссылаются на ряд дополнительных справочных и иных материалов, которые представлены в виде электронного учебно-методического комплекса. Данный электронный учебно-методический комплекс включает в себя такие учебные материалы, как примеры проектов программ, используемые студентами для «быстрого старта» в программировании микроконтроллеров, полный спектр требуемой справочной и технической документации на рассматриваемую аппаратуру (включая Datasheet на все устройства), некоторую основную литературу из библиографического списка данных методических указаний. Ссылка на раздел электронного методического комплекса выдается студенту на первом лекционном или лабораторном занятии в электронном виде. Таким образом, указание на какой-либо файл электронного учебно-методического комплекса по дисциплине означает необходимость перейти по полученной ссылке и найти требуемый файл.

Для успешного изучения материалов курса требуются знания учебного материала таких дисциплин, как: программирование, компьютерная электроника, архитектура компьютеров, компьютерная схемотехника, микроэлектронные компоненты компьютерных систем.

Знания, полученные в курсе, используются при изучении последующих учебных курсов: проектирование микропроцессорных систем, автоматизированные системы на встроенных контроллерах, проектирование БИС на базе ПЛИС.

В методических указаниях представлена информация как теоретического, так и практического характера. Электронная форма методических указаний содержит ряд цветных иллюстраций, что помогает восприятию материала. Используя изложенный материал, представленные алгоритмы и фрагменты программного кода, читатель может, используя соответствующую среду программирования, создавать полноценные управляющие программы для

микроконтроллеров, выполнять их тестирование и загрузку в реальное устройство.

В конце каждого раздела собраны контрольные вопросы, ответ на которые позволяет судить о степени усвоения теоретического материала.

Раздел 1. Базовые понятия. Системы сигналов

В рамках курса «Микропроцессорные системы» студенту предоставляется возможность работы с реальным оборудованием, в том числе с платами аппаратно-программного комплекса Arduino. Данный комплекс является мировым стандартом для макетирования микропроцессорных систем, изучения микроконтроллеров и широкого спектра дополнительного оборудования.

Важно помнить, что комплекс представляет собой реальную электрическую схему и соблюдать основные правила безопасности при работе с электроприборами, а также проявлять аккуратность при работе.

Стандартным уровнем напряжения в микроэлектронике считается 5 В. Для питания ядер высокоинтегрированных микроконтроллеров часто используется напряжение с уровнем 3,3 В. На ядро микропроцессора персонального компьютера подается, в зависимости от модели, от 1,7 до 2,8 В.

1.1. Понятие аналогового сигнала. Амплитуда, частота, период

Аналоговый сигнал – непрерывный сигнал, существующий в любой произвольный момент времени заданного временного интервала. Практически все сигналы, встречающиеся в живой природе, являются аналоговыми. Типичный аналоговый сигнал – осциллограмма электрических колебания, полученная с обычного микрофона.

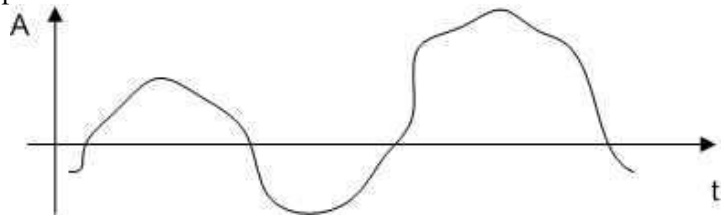


Рис. 1.1. Аналоговый сигнал

Аналоговый сигнал, подчиняющийся закону sin или cos называют гармоническим, общая запись его выглядит следующим образом: $y = A\cos(\omega t + \varphi_0)$.

Основные параметры аналогового сигнала следующие: амплитуда колебаний (A, расстояние между крайними отсчетами сигнала, отсчитывается по вертикали, для электрического сигнала измеряется в Вольтах (В); если сигнал гармонический, в таком случае он описывается также частотой колебаний ($f = 2\pi / \omega$, измеряется в Герцах (Гц) или периодом ($T = 1/f$, измеряется в секундах (с), который обратнопропорционален частоте.

1.2. Понятие цифрового сигнала. Амплитуда, частота, период, скважность, duty-цикл

Цифровой сигнал – дискретный сигнал, существующий только в определенные моменты времени. Практически все сигналы, встречающиеся в микропроцессорной технике, являются цифровыми. На рисунке 2 показан типичный цифровой сигнал.

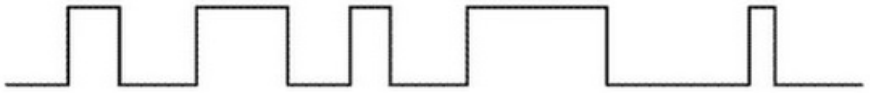


Рис. 1.2. Цифровой сигнал

Основные параметры цифрового сигнала аналогичны параметрам аналогового, в частности амплитуда колебаний (A), частота (Гц) или период.

Процесс перевода сигнала из аналоговой формы в цифровую называют оцифровкой, которая выполняется с помощью аналого-цифрового преобразователя. Суть его работы состоит в периодической регистрации величины аналогового сигнала и записи ее в цифровой форме.

Таким образом, цифровой сигнал приобретает такие параметры, как **частота дискретизации** – частота отсчетов исходного аналогового сигнала, ед./с., выражаемая в Гц, и **разрядность** – число двоичных разрядов, используемых для записи каждого отсчета. Несложно проверить, что максимальное число различимых уровней такого сигнала описывается как $2^{разрядность}$. Например, для сигнала с разрядностью 8 бит минимально различимый уровень сигнала будет составлять $1/256$ максимального уровня.

Сигнал, показанный на рисунке 1.2, также называют прямоугольным. Для такого сигнала цифрового сигнала, минимум

которого часто является выключенным, а максимум – включенным состоянием чего-либо, вводится дополнительный параметр – скважность (S). **Скважностью называют отношение времени включенного состояния к общему времени периода**. Величина, обратная скважности и часто используемая в англоязычной литературе, называется **коэффициентом заполнения** (англ. Duty cycle, D): $S = T/\tau = 1/D$. Частое применение в практике находит сигнал со скважностью, равной двум. Такой сигнал называется **меандр**.

1.3. Лабораторная работа: Техника безопасности. Анализ параметров аналогового и цифрового сигнала.

Цель: Получить практические навыки работы с цифровым осциллографом на базе ПК, самостоятельно изучить применение встроенных измерительных возможностей аппаратно-программного комплекса

Оборудование: Цифровой осциллограф (или осциллограф-приставка), генератор сигналов, дополнительное оборудование.

Теоретическая часть.

Осциллограф – прибор для визуального наблюдения электрических сигналов на экране.

Основные органы управления следующие. Масштаб изображения по вертикали определяется коэффициентом усиления по оси Y (входной аттенюатор) и подписывается как «Вольт на деление» или $V/дел$. При увеличении значений $V/дел$ размер картинки по оси Y будет уменьшаться, и появляется возможность наблюдать больший по амплитуде сигнал. Масштаб изображения по горизонтали определяется временем, за которое развертка двигает луч в горизонтальной плоскости слева направо, и это определяется временем, за которое луч проходит одно крупное деление шкалы ($время/дел$).

Также важно учитывать расположение вытяжного переключателя усиления по каналу Y ($x1$ или $x10$), и положение вытяжного переключателя развертки ($x1$ или $x0{,}2$).

Алгоритм работы с любым осциллографом в данной лабораторной работе следующий:

1. Подключить щуп к требуемому каналу и убедиться, что переключатель режимов входа рядом с ручкой $V/дел$ находится в

положении «~», что означает т.н. «закрытый» вход, т.е. вход через разделительный конденсатор.

2. Переключателем $V/дел$ установить максимальное значение возможного сигнала (минимальное усиление, ручка против часовой стрелки).

3. Переключатель $время/дел$ установить в середине шкалы.

4. Подключить щупы к исследуемому сигналу, и пошагово увеличивая усиление канала Y, вращая регулятор синхронизации, добиться, чтобы сигнал занимал более половины экрана по вертикали.

5. С помощью переключателя $время/дел$ установить такой режим, когда на экране будут отображаться от 2 до 5 периодов сигнала.

6. Выполнить измерение амплитуды сигнала, отсчитав целое и дробное число больших делений по оси Y, умножив на текущее значение переключателя $V/дел$.

7. Выполнить измерение длительности периода, отсчитав целое и дробное число больших делений по оси X и умножив их на текущее значение переключателя $время/дел$.

8. Перед следующим измерением вернуть переключатель $V/дел$ на максимальное значение возможного сигнала (минимальное усиление, ручка против часовой стрелки).

Важно: при переключениях не допускать остановки луча, т.к. это приводит к повреждению люминофора ЭЛТ осциллографа (в случае аналогового прибора), а также не выставлять чрезмерно большие усиления входного сигнала (правая треть делений переключателя $V/дел$ не должна использоваться).

Цифровой осциллограф-приставка выполняет функции, аналогичные аналоговому устройству. Состоит он из программно-регулируемого входного аттенюатора, усилителя и аналого-цифрового преобразователя, данные с которого передаются на компьютер, где специальная программа их обрабатывает и выводит на экран. Благодаря оцифровке и дальнейшей обработке сигнала в цифровом виде, кроме, собственно, наблюдения, цифровой осциллограф обладает рядом дополнительных возможностей, таких как: запоминание картинки, анализ сигнала путем вычисления амплитуды, периода, частоты, Duty-цикла и других параметров.

Для подключение цифрового осциллографа-приставки к компьютеру требуется специальный драйвер и программное обеспечение, которое можно скачать по адресу www.oscill.ru.

Практическая часть.

1. С помощью цифрового осциллографа-приставки и генератора сигналов измерить амплитуду, период и частоту аналоговых сигналов при трех различных положениях ручки частоты генератора. Измерения проводить с помощью встроенных средств программы. Для их корректной работы требуется выбрать сигнал с помощью «мыши».
2. Сделать снимки экрана с сигналом и открытыми окошками встроенных средств измерения. Распечатать и вклеить в тетрадь сигнал, положения отсчетных делений и результаты измерений.
3. С помощью цифрового осциллографа-приставки измерить амплитуду, период и частоту аналоговых сигналов на выходе 3 и 13 заранее подготовленной платы Arduino Uno.
4. Сделать снимки экрана с сигналом и открытыми окошками встроенных средств измерения. Распечатать и вклеить в тетрадь сигнал, положения отсчетных делений и результаты измерений.
5. Оформить работу и написать вывод.

Раздел 2. Микропроцессорная аппаратно-программная платформа Arduino

2.1. Техническое описание контроллерной платы Arduino Uno

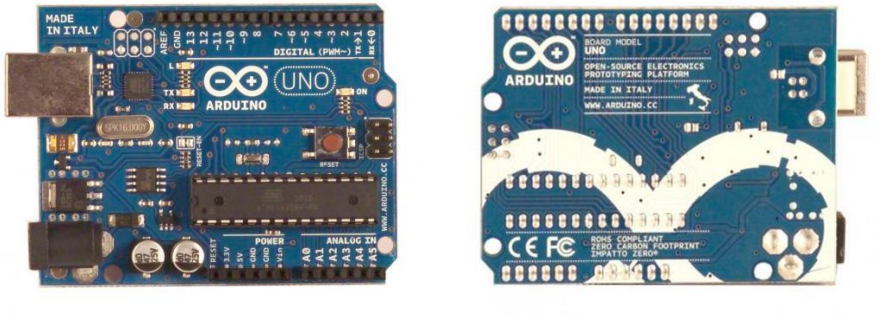


Рис. 2.1. Общий вид микроконтроллерной платы Arduino Uno Rev.3

Общие сведения

Контроллерная плата Arduino Uno Rev.3 построена на *ATmega328* (см. файл методического комплекса *ATmega328.pdf*). Платформа имеет 14 цифровых входов/выходов (6 из которых могут использоваться как выходы ШИМ), 6 входов АЦП, кварцевый генератор 16 МГц, разъем USB, силовой разъем, разъем ICSP и кнопку перезагрузки. Для работы необходимо подключить платформу к компьютеру посредством кабеля USB, либо подать питание на силовой разъем платы.

В отличие от всех предыдущих плат, использовавших FTDI USB микроконтроллер для связи по USB, новый Arduino Uno использует микроконтроллер *ATmega8U2* (см. файл методического комплекса *ATMega8.pdf*).

Характеристики

Микроконтроллер	ATmega328
Рабочее напряжение	5В
Входное напряжение (рекомендуемое)	7-12В
Входное напряжение (предельное)	6-20В
Постоянный ток через вход/выход	40 мА
Постоянный ток для вывода 3.3 В	50 мА
Флеш-память	32 Кб
ОЗУ (из которых 0.5 Кб используются для загрузчика)	2 Кб (ATmega328)
EEPROM	1 Кб (ATmega328)
Тактовая частота	16 МГц

Файлы электрической схемы, разводки и топологии платы в электронном методическом комплексе следующие:

Файлы EAGLE: *arduino-duemilanove-reference-design.zip*

Принципиальная схема*: arduino-uno-schematic.pdf*

Питание

Arduino Uno может получать питание через подключение USB или от внешнего источника питания. Источник питания выбирается автоматически.

Внешнее питание (не USB) может подаваться от лабораторного блока питания или от аккумуляторной батареи и подключается посредством разъема с диаметром 2.1 мм с центральным положительным полюсом.

Платформа может работать при внешнем питании от 6В до 20В. При напряжении питания ниже 7В вывод «5V» может выдавать менее

5 В, при этом платформа может работать нестабильно. При использовании напряжения выше 12В регулятор напряжения может перегреться и повредить плату. Рекомендуемый диапазон от 7В до 12В.

Выводы питания

VIN. Вход используется для подачи питания от внешнего источника в отсутствие напряжения питания на разъеме USB или на силовом разъеме.

5V. Регулируемый источник напряжения, используемый для питания микроконтроллера и компонентов на плате. Питание может подаваться от вывода VIN через встроенный регулятор напряжения, или от разъема USB, или от силового разъема.

3V3. Напряжение на выводе 3.3В, генерируется встроенным регулятором на плате. Максимальное потребление тока 50 мА.

GND. Выводы заземления.

Память

Микроконтроллер *ATmega328* располагает 32 кБ флэш памяти, из которых 0.5 кБ используется для хранения загрузчика, а также 2 кБ ОЗУ (SRAM) и 1 Кб EEPROM.

Входы и выходы

Последовательная шина: 0 (RX) и 1 (TX). Выводы используются для получения (RX) и передачи (TX) данных TTL. Данные выводы подключены к соответствующим выводам микросхемы последовательной шины *ATmega8U2* USB-to-TTL.

Внешнее прерывание: 2 и 3. Данные выводы могут быть сконфигурированы на вызов прерывания при изменении уровня сигнала по переднему или заднему фронту или при изменении уровня.

ШИМ: 3, 5, 6, 9, 10, и 11. Любой из выводов обеспечивает ШИМ с разрешением 8 бит.

SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK). Посредством данных выводов осуществляется связь по протоколу SPI и осуществляется внутрисхемное программирование.

LED: 13. Встроенный светодиод, подключенный к цифровому выводу 13. Если значение на выводе имеет высокий потенциал, то светодиод горит.

На платформе Arduino Uno установлены 6 аналоговых входов (обозначенных как A0 .. A5), каждый разрешением 10 бит (т.е. может принимать 1024 различных значений). Стандартно выводы имеют

диапазон измерения до 5В относительно общего провода, тем не менее, имеется возможность изменить верхний предел посредством вывода AREF. Некоторые выводы имеют дополнительные функции:

I2C: 4 (SDA) и 5 (SCL). Посредством выводов осуществляется связь по протоколу I2C (TWI), для создания которой часто используется библиотека Wire.

Дополнительная пара выводов платформы

AREF. Опорное напряжение для аналоговых входов. Используется с функцией analogReference().

Reset. Низкий уровень сигнала на выводе перезагружает микроконтроллер. Обычно применяется для подключения кнопки перезагрузки на плате расширения, закрывающей доступ к кнопке на самой плате Arduino.

Обратите внимание на соединение между выводами Arduino Uno и портами *ATmega328* (см. рис. 2.2)

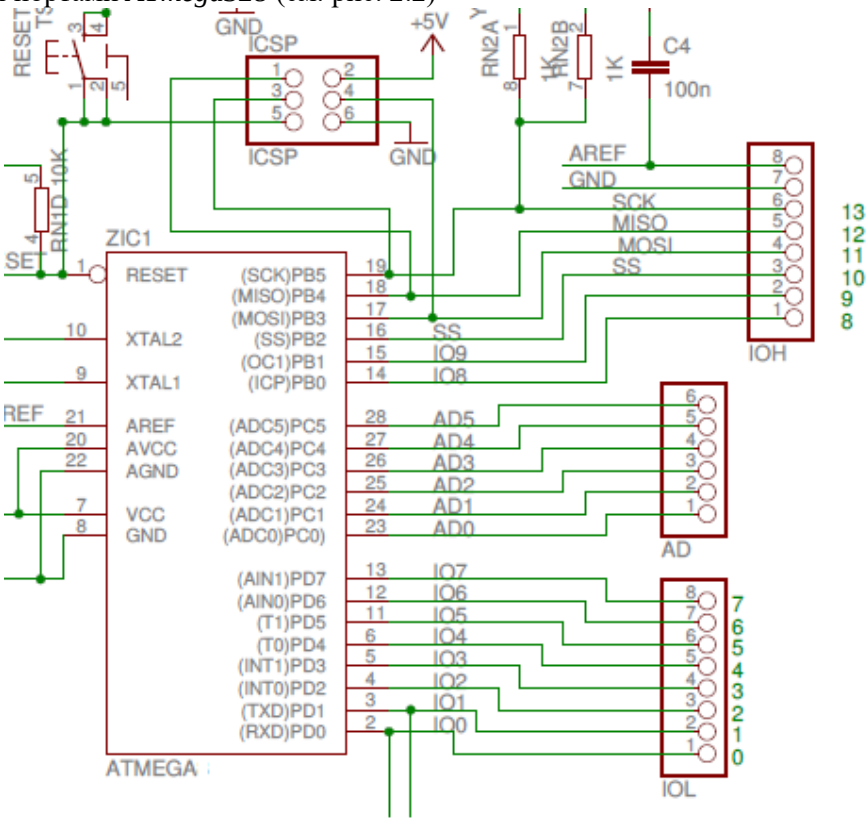


Рис. 2.2. Фрагмент схемы Arduino Uno

Связь

На платформе Arduino Uno установлено несколько устройств для осуществления связи с компьютером, другими устройствами Arduino или микроконтроллерами. *ATmega328* поддерживает последовательный интерфейс UART TTL (5В), реализуемый выводами 0 (RX) и 1 (TX). Установленный на плате второй микроконтроллер *ATmega8U2* преобразует данные этого интерфейса в формат USB, при этом программы на стороне компьютера взаимодействуют с платой через виртуальный COM порт. Прошивка в *ATmega8U2* использует стандартные драйвера USB–COM (на Windows для подключения потребуется файл *ArduinoUNO.inf*).

Программирование

Микроконтроллер *ATmega328* поставляется с записанным загрузчиком, облегчающим запись новых программ без использования внешних программаторов. Связь осуществляется оригинальным протоколом *STK500*.

Имеется возможность не использовать загрузчик и программировать микроконтроллер через выводы *ICSP* (внутрисхемное программирование).

Аббревиатуры **внутрисхемного программирования *ISP* и *ICSP*** означают *In System Programming* и *In Circuit Serial Programming* соответственно. Это метод программирования чипа, уже подключенного в некоторую схему (программирование в готовом устройстве) по последовательному протоколу.

Принципиально важным является то, что программируемый микроконтроллер должен успешно стартовать, и только после этого он будет в состоянии принимать данные от программатора. Это означает, что он должен быть подключен к питанию и иметь соответствующий источник тактовых сигналов.

Источник тактовых сигналов выбирается в микроконтроллерах серии ATMega с помощью так называемых fuse-битов, которые, как и память программ и EEPROM, доступны для изменения с помощью программатора.

Производитель перед продажей выставляет fuse-биты так, что в качестве источника тактовых сигналов выбран внутренний генератор. Такой микроконтроллер можно просто подключить к ISP программатору с учетом расположения его выводов и начать работу.

Однако, если с помощью ISP программатора изменить значения fuse-битов так, что изменится источник тактовых сигналов, тогда, чтобы ISP программатор начал работать с микроконтроллером, придется подключить соответствующий источник тактовых сигналов к микроконтроллеру. Таким образом, нужно быть внимательным при изменении значений fuse-битов.

Программирование (прошивку) микроконтроллера ATMega328P контроллерного модуля Arduino Uno rev.3 возможно осуществлять посредством внутрисхемного последовательного интерфейса ISP (6-ти контактный разъем на плате модуля), а также с помощью программы-загрузчика, находящегося во Flash-памяти микроконтроллера. Данный загрузчик занимает примерно 0,5 кБ, однако существенно упрощает процесс прошивки микроконтроллера, снижает до минимума вероятность неправильной установки аппаратных конфигурационных битов (Fuse bits), т.к. не дает возможности их изменять.

BootLoader – это небольшая программа, которая находится в специальной области памяти микроконтроллера и «слушает» какой-либо интерфейс. Обычно это **UART**, но быть **SPI**, **USB** и даже **SoftUSB**.

При загрузке контроллера управление передается загрузчику, он проверяет, есть ли условие для запуска. Условие может быть любым, но обычно это наличие специального байта по интерфейсу, либо наличие нужного логического уровня на выбранной ножке контроллера, сигнализирующее о том, что необходимо обратиться к загрузчику. Если условие есть, то загрузчик может, например, принять прошивку по **UART** и записать ее во Flash-память МК. Обычно с помощью программы-загрузчика осуществляют прошивку микроконтроллера без применения специального программатора.

Если разрешающего условия при старте нет, то загрузчик завершает свою работу и передает управление основной программе.

Bootloader в AVR

Что происходит при старте контроллера? В нормальном состоянии процессор начинает по одной выполнять инструкции из памяти программ, начиная с нулевого адреса. На рис. 2.3 показано, как это примерно выглядит в ATmega16, в других микроконтроллерах процесс аналогичен, адреса могут быть другими.

Но если активировать fuse-бит **BOOTRST,** то процессор будет стартовать не с нулевого адреса, а с адреса начала Boot сектора. Этот

сектор расположен в самом конце памяти программ и его размер задается fuse-битами **BOOTSZx.**

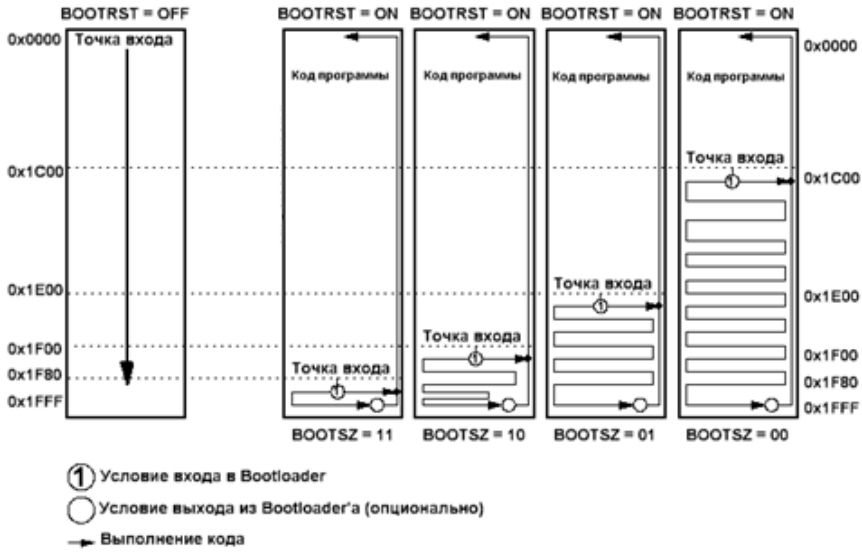


Рис. 2.3. Обращение к памяти при загрузке МК ATMega16

Упрощенно идея использования программы-загрузчика (bootloader) описывается алгоритмом, приведенным на рис. 2.4.

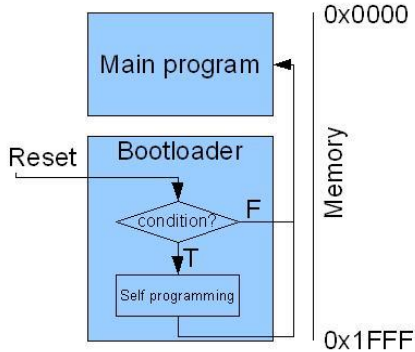


Рис. 2.4. Алгоритм загрузки МК с программой-загрузчиком

Для установки на компьютер драйвера модуля микроконтроллера Arduino UNO Rev.3 выполните следующую последовательность действий:

1. Скачайте последнюю версию оболочки Arduino (для Windows 7 32bit – http://arduino.googlecode.com/files/arduino-1.0-windows.zip или отдельно файл *«Arduino UNO Rev3.inf»*). Для другой операционной системы – по ссылке http://arduino.cc/en/Main/Software скачайте соответствующий программный пакет и найти в папке «Drivers» файл .inf для Arduino Uno Rev.3.

2. Подключите микроконтроллерный блок с помощью кабеля и в «Диспечере устройств» Windows найдите это устройство. Щелкните на нем правой кнопкой мыши, выберите «Обновить драйвер», затем «Указать расположение драйверов вручную» и укажите папку, в которой находится файл Arduino UNO Rev3.inf.

3. Нажмите «Продолжить», должны установиться драйвера.

4. Раскройте в «Диспетчере устройств» укладку Ports(COM&LPT) и запишите, на каком порту установлен Arduino Uno Rev.3 (**ИД устройства:** USB\VID_2341&PID_0043&REV_0001 и USB\VID_2341&PID_0043) .

Для программирования (прошивки) микроконтроллерного модуля Arduino Uno Rev.3 можно использовать любую программу, имеющую возможность взаимодействия с загрузчиком (bootloader) Arduino Uno Rev.3, например Arduino HEX Uploader или XLoader.

Скачать Arduino HEX Uploader можно по ссылке *http://www.ngcoders.com/downloads/arduino-hex-uploader-and-programmer/*

Скачать Xloader можно по ссылке *http://russemotto.com/xloader/*

Настоятельно рекомендуется использовать XLoader!

После подключения микроконтроллерного модуля Aruino Uno Rev.3 запустите xloader.exe, выберите тип прошиваемого микроконтроллера (ATMega328), порт на который установлен Arduino и файл с прошивкой (.hex).

После программирования (прошивки) микроконтроллера в строке состояния программы XLoader появится надпись «X bytes uploaded», где X – объем прошивки в байтах.

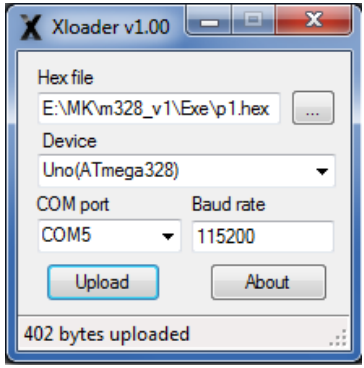


Рис. 2.5. XLoader v1.00

Автоматическая (программная) перезагрузка

Платформа Arduino Uno разработана таким образом, чтобы перед записью нового кода перезагрузка осуществлялась самой программной средой Arduino на компьютере, а не нажатием кнопки на платформе. Одна из линий DTR микросхемы *ATmega8U2*, управляющих потоком данных (DTR), подключена к выводу перезагрузки микроконтроллера *ATmega328* через конденсатор номиналом 100 нФ. Активация данной линии, т.е. подача сигнала низкого уровня, перезагружает микроконтроллер. Подача сигнала низкого уровня по линии DTR скоординирована с началом записи кода, что сокращает таймаут загрузчика.

Данная функция имеет еще одно применение. Перезагрузка платформы Arduino Uno происходит каждый раз при подключении к программе Arduino на компьютере с ОС Mac X или Linux (через USB). Следующие полсекунды после перезагрузки работает загрузчик. Во время программирования происходит задержка нескольких первых байт кода во избежание получения платформой некорректных данных (всех, кроме кода новой программы). Если производится разовая отладка скетча, записанного в платформу, или ввод каких-либо других данных при первом запуске, необходимо убедиться, что программа на компьютере ожидает в течение секунды перед передачей данных.

На Arduino Uno имеется возможность отключить автоматическую перезагрузкы разрывом соответствующей линии. Линия маркирована «**RESET-EN**». Отключить автоматическую перезагрузку также возможно, подключив резистор номиналом 110 Ом между источником питания 5В и данной линией.

Токовая защита разъема USB

В Arduino Uno встроен самовосстанавливающийся предохранитель (автомат), защищающий порт USB компьютера от токов короткого замыкания. Хотя, практически все компьютеры имеют подобную защиту, тем не менее, данный предохранитель обеспечивает дополнительный барьер. Предохранитель срабатывает при прохождении тока величиной более 500 мА через порт USB и размыкает цепь до тех пор, пока нормальные значения токов не будут восстановлены.

Физические характеристики

Длина и ширина печатной платы Arduino Uno составляют 6.9 см и 5.3 см соответственно. Разъем USB и силовой разъем выходят за границы данных размеров. Четыре отверстия в плате позволяют закрепить ее на поверхности. Расстояние между цифровыми выводами 7 и 8 равняется 0,4 см, между другими выводами оно составляет 0,25 см.

2.2. Модули расширения Arduino

Концепция аппаратной части Arduino подразумевает быстрое создание различных по возможностям и функционалу аппаратно-программных устройств, для чего требуются различные аппаратные модули, расширяющие возможности процессорного модуля. Для этого на всех основных платах Arduino предусмотрены выводы питающих напряжений, интерфейсов передачи данных, аналоговых и цифровых входов/выходов. По устоявшейся терминологии, платы расширения для Arduino часто называют «shield». Типичный вид процессорного блока с установленными платами расширения показан на рисунке 2.6.

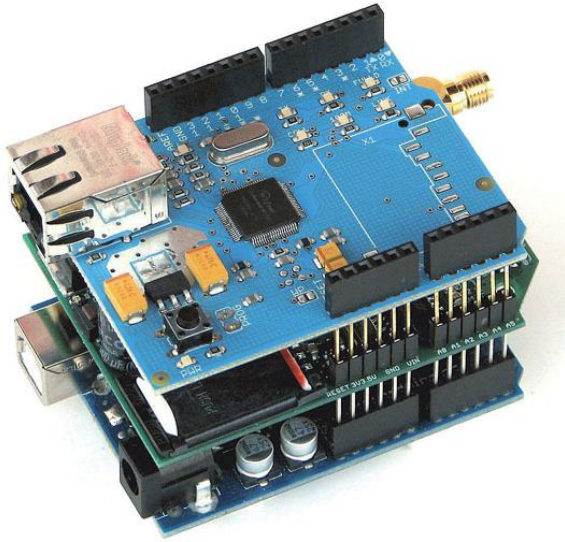


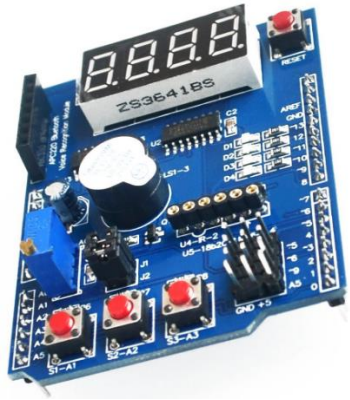
Рис. 2.6. Общий вид Arduino с платами расширения

Для лабораторных работ доступны специальные блоки расширения, позволяющие работать с 8 светодиодами, двумя

цифровыми семисегментными индикаторами с динамической индикацией, а также с кнопками для организации простого меню.

Модуль расширения №1 для лабораторных работ, использование модуля

Один из модулей расширения для лабораторных работ показан на рисунке 2.7.



На плате имеется 4 семисегментных индикатора, включенных через сдвиговые регистры 74HC595, рядом с которыми располагается кнопка перезагрузки и разъем APC220 для подключения модулей Bluetooth или голосового модуля. Кроме этого на плате имеется четыре красных светодиода, подключенные к портам D10, D11, D12, D13 платы Arduino.

Рис. 2.7. Модуль расширения

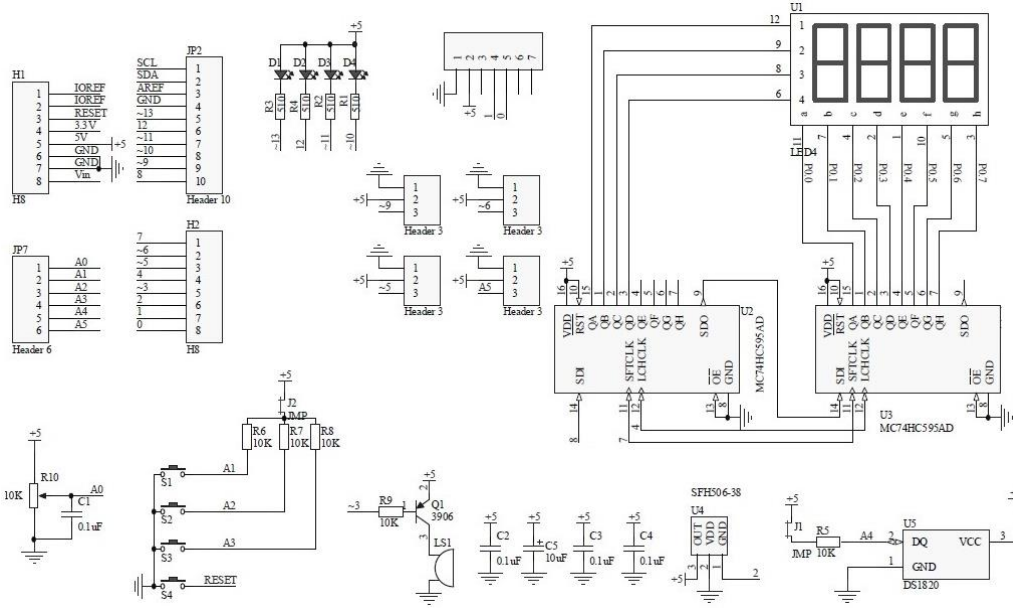


Рис. 2.8. Схема модуля расширения

Зуммер подключается к порту D3, следует заметить, что звукоизлучатель оснащен встроенным генератором, так что воспроизвести простую мелодию с его помощью не получится. В нижней части платы располагается подстроечный резистор, подключенный к порту A0.

Три кнопки, подсоединены к портам A1, A2, A3 (цифровые порты D15, D16, D17, соответственно). Четыре трехконтактных разъема подключены портам D5, D6, D9, A5 и предназначены для подключения внешних устройств. Завершает список устройств разъем для подключения аналогового LM35 или цифрового DS18B20 датчиков температуры. Датчики подключаются к порту A4. Перемычка J1 подключает или отключает резистор 10 кОм для корректной работы датчиков. Таблица сводных параметров шилда приведена ниже.

Таблица 2.1. Параметры и номера выводов Arduino UNO

Шилд	**PIN №**
4 светодиода LEDs	D10, D11, D12, D13
3 тактовые кнопки reset button	A1, A2, A3
Потенциометр (10 kОм)	A0
4-разрядный 7-семисегментный LED индикатор на 2-х сдвиговых регистрах 74HC595	Latch 4, Clock 7, Data 8
Звуковой излучатель (Зуммер)	D3 (digital On/Off)
Разъем для подключения инфракрасного датчика (remote control)	D2
Разъем для подключения аналогово LM35 или цифрового DS18B20 датчиков температуры (Джампер J1 подключает или отключает резистор 10 кОм для правильной работы этих датчиков)	A4
Разъем APC220 для подключения модулей (Bluetooth, голосового модуля и др.)	GND, +5v, D0, D1 (rx/tx)
Свободные пины (с ШИМ (pwm)), например для сервоприводов и др.	D5, D6, D9, A5

2.3. Программная среда Arduino. Алгоритм выполнения, загрузка программы в микроконтроллер

Кроме аппаратной части, интегрированная среда для написания кода и отладки также носит название «Arduino». Общий вид окна IDE приведен на рисунке 2.10. Программа в среде Arduino именуется «Скетч».

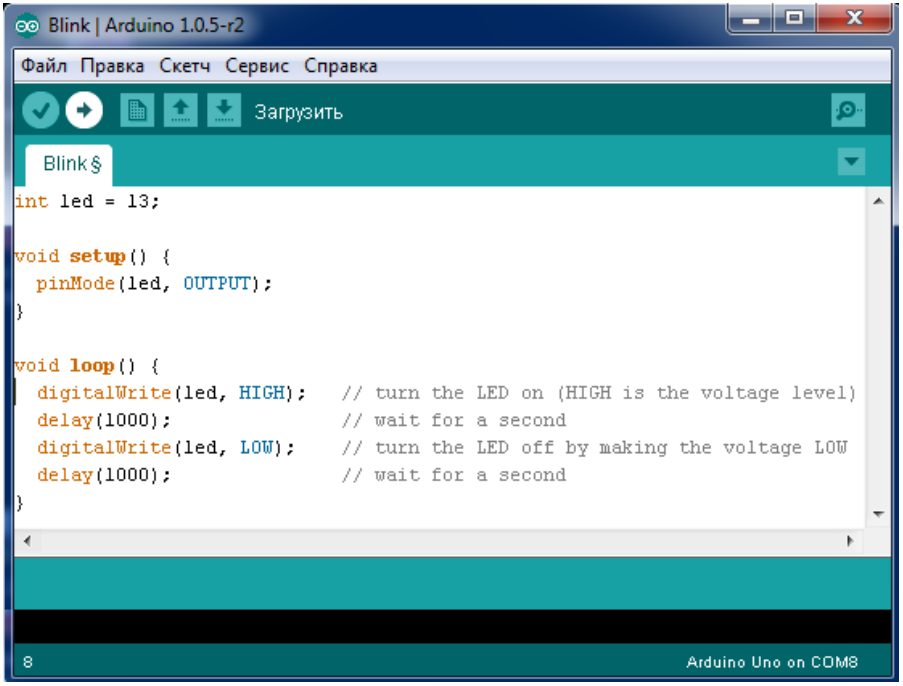


Рис. 2.10. IDE Arduino

На примере простейшей программы «Blink», взятой из папки примеров, можно пояснить принцип работы управляющей программы микроконтроллера, который в виде блок-схемы показан на рисунке 2.11.

Начинается программа с объявления переменных (в данном случае переменная с именем «led» и типом «int», затем следует функция «setup», которая будет выполняться однократно при загрузке микроконтроллера. В эту функцию вписываются команды, отвечающие за начальную настройку аппаратной части МК – в данном случае вывод платы Arduino под номером 13 настраивается как выход.

Также, в функцию «setup» можно вписывать любые команды, для которых требуется однократное выполнение.

Следующая функция «loop» представляет собой вечный цикл без условия выхода. Т.о. она будет выполняться микроконтроллером до тех пор, пока он не будет выключен или перезагружен. Важно отметить, что команды выполняются друг за другом без задержек и, при тактовой частоте МК равной 16 МГц, разница времени между простыми командами может составлять $6{,}25*10^{-8}$ с., т.е. 62,5 наносекунды. Поэтому, если требуется значительная задержка, к примеру, для визуализации включения и выключения светодиода, необходимо использовать команду, формирующую управляющую задержку, например «delay(1000)» – в данном случае формируется задержка в 1000 мс., т.е. в 1 секунду.

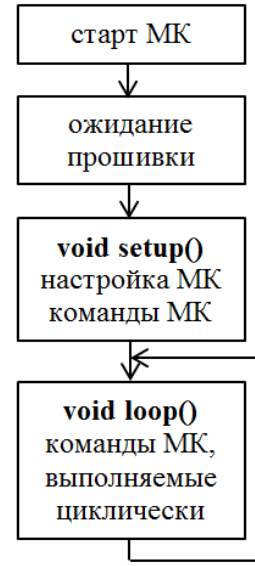


Рис. 2.11. Алгоритм работы программных модулей

После написания программы, требуется указать целевую платформу – тип используемой платы Arduino, выбрав соответствующий пункт из меню «Сервис», а также указать номер последовательного порта, на котором указывает драйвер виртуального Com-порта. Этот номер можно узнать, открыв «Диспетчер устройств» и вкладку «Порты (COM и LPT)».

Загрузка программы в микроконтроллер осуществляется при нажатии соответствующей кнопки программной среды. При этом, сначала осуществляется ее компиляция и в случае ее успешного завершения выполняется перезагрузка платы Arduino и, после этого, встроенному загрузчику передается скомпилированный код в формате «intel hex», который загрузчик помещает с начала адресного пространства флеш-памяти программы микроконтроллера. По окончании процесса передачи МК снова перезагружается.

2.4. Программы-симуляторы Arduino и МК AVR ATMega

Существует два способа проверки и отладки программного обеспечения разрабатываемых радиотехнических устройств на микроконтроллерах:
– отладка программы непосредственно на микроконтроллере;
– отладка программы виртуально с помощью специальных программных пакетов (симуляторов). При этом составление программы, проверка ее на синтаксические ошибки, компиляция, моделирование работы устройства, поиск ошибок в алгоритме программы выполняется виртуально на компьютере. Такой способ отладки называется программым. Он имеет много преимуществ: позволяет выполнять эмуляцию работы устройства в пошаговом режиме; просматривать содержание регистров общего назначения, регистров ввода/вывода, состояния портов, периферийных устройств микроконтроллера и т.д; видеть работу устройства в динамике, просматривать осциллограммы сигналов в любой точке схемы и измерять временные параметры сигналов.

2.4.1. Online-симулятор Arduino

Один из лучших Онлайн-симуляторов простых схем – Arduino Tinkercad (https://www.tinkercad.com/dashboard), представляющий собой web-приложение с хорошей имитацией платформы Arduino, которое позволяет в визуальном режиме прямо из браузера редактировать код и строить схемы без паяльника и проводов.

Работать над схемами можно совместно с другими людьми, используя библиотеку компонентов. В визуальном режиме можно накидывать провода и различные компоненты на макетную плату и подключать к виртуальной микропроцессорной плате Arduino. Также есть редактор кода, который аналогично IDE Arduino, позволяет создавать программный код, компилировать его и загружать в виртуальную плату Arduino.

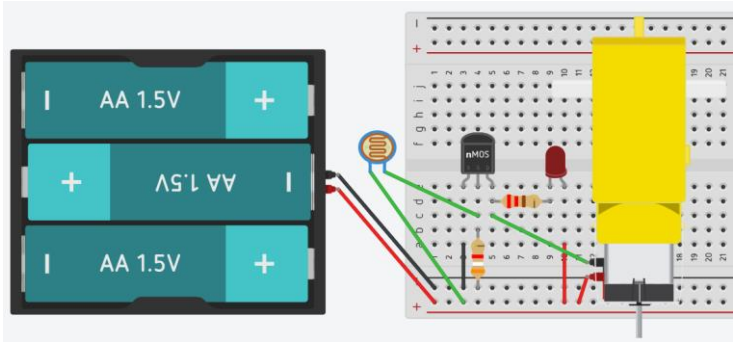


Рис. 2.12. Online-симулятор, основное окно и макетная плата

На рисунке 2.12 показана пустая макетная плата, на которой можно собирать требуемую схему и к ней добавлять детали. Для добавления компонент, для отображения окна ввода кода программы и для ее компиляции и загрузки в МК используются соответствующие клавиши, обведенные рамкой и обозначенные цифрой «1».

Важно отметить, что макетная плата обладает встроенными электрическими соединениями контактных отверстий. Так, два верхних и два нижних горизонтальных ряда контактов соединены между собой, на рисунке 2.12 показано соединение контактов верхнего ряда. Обычно, эти контакты используют для подачи положительного и отрицательного полюса питания. Остальные контакты объединены в группы по 5 контактов, соединенные вертикально. Пример одной такой группы также показан на рисунке 2.12, остальные контакты соединены аналогично. Такой тип соединения диктует особенности реализации схемы с помощью макетной платы.

2.4.2. Симулятор VMLAB

Существует целый ряд полноценных программных пакетов (эмуляторов и симуляторов), обеспечивающих отладку программного обеспечения микроконтроллеров, например: AVR Studio, CodeVisionAVR, Proteus, Binder, Visual Micro Lab и т.д.

Программный пакет Visual Micro Lab (в дальнейшем VMLAB) – это программный пакет, предназначенный для отладки программного обеспечения и моделирования работы радиоэлектронных устройств, в состав которых входят микроконтроллеры AVR. Его часто называют

виртуальной лабораторией или симулятором. VMLAB легко связывается с ассемблерными и Си–компиляторами.

Исходным файлом для VMLAB является непосредственно .hex файл прошивки (можно подключать и файл ассемблера .asm). Симулятор позволяет работать со светодиодами, использовать осциллограф, изменять программно температуру, частоту кристалла, и параллельно наблюдать, как изменяются иные параметры (например, ток потребления), наблюдать содержимое регистрового файла, памяти EEPROM, значения регистров периферийных устройств микроконтроллера.

Типовой файл проекта выглядит следующим образом:

```
; Micro + software running
; ------------------------------------------------------------
.MICRO "ATmega8"
.TOOLCHAIN "GENERIC"
.TARGET "kpprj_v1.hex"
.SOURCE "kp_v1.c"
.COFF "kpprj_v1.cof"

; Following lines are optional; if not included ; exactly these values
; ------------------------------------------------------------
.POWER VDD=5 VSS=0  ; Supply,  VDD+5v, VSS - ground
.CLOCK 1meg                 ; Clock
.STORE 2000m                ; Trace (micro+signals) storage time

; Micro nodes: RESET, AREF, PB0-PB7, PC0-PC6,
; Define here the hardware around the micro
; ------------------------------------------------------------

; LED diodes D2 and current limiting resistor R1
D1 VDD res16 ; LED 1 on the control panel
R16 res16 PC0 430;

; add key (VSS, PB0)
K0 PB0 VSS ; if you need latching, add ;LATCHED

;add pull-up to the VDD «+» with resistors 5КОм
R0 VDD PB0 5000
R1 VDD PB1 5000

;add Scope
.plot V(PB0) V(PC0)
```

2.4.3. Симулятор электронных схем ISIS Proteus

Симулятор ISIS Proteus (рассматриваемая версия – 7) является интегрированной системой, позволяющей разрабатывать аналоговые и цифровые электронные схемы, выполнять рисование схемы, ее симуляцию и поиск ошибок. Симулятор содержит в себе значительный перечень виртуальных инструментов с функционалом равным или даже большим, чем аналогичные физические устройства. Пример внешнего вида программы показан на рисунке 2.13.

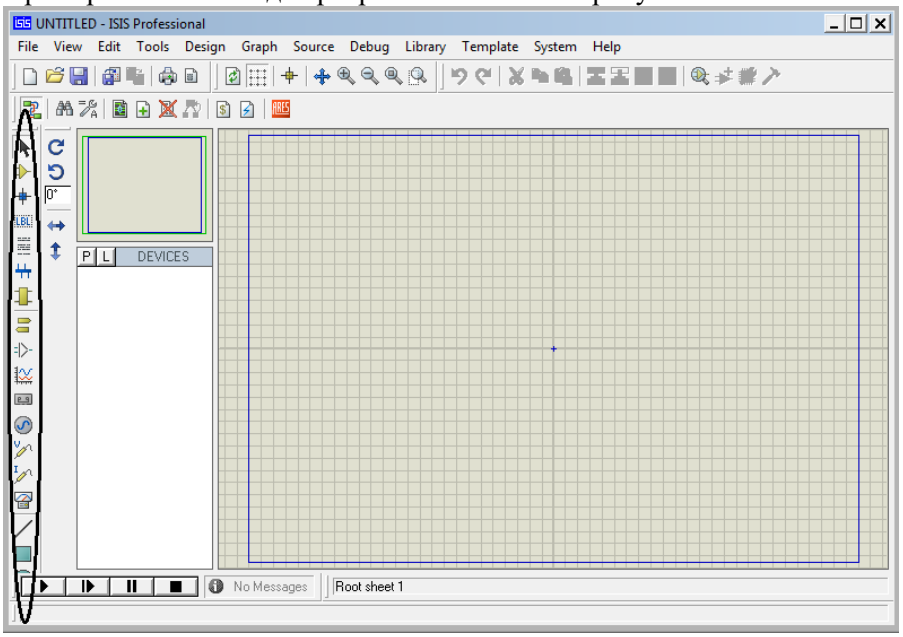


Рисунок 2.13. Симулятор ISIS Proteus

Важной особенностью программы является возможность моделирования реальных цифровых компонент, например микроконтроллеров. Так, к примеру, на схему можно добавить микроконтроллер, в его модель загрузить файл прошивки и наблюдать работу такой электрической схемы под управлением микроконтроллера.

Важно: детализация представления процессов, происходящих в микроконтроллере существенно ниже, чем в специализированном симуляторе VMLAB, однако для большинства учебных задач этого достаточно.

Также следует отметить тот факт, что виртуальное время моделирования может существенно отличаться от реального, что следует наблюдать по индикатору системного времени внизу окна программы. Кроме того, использование любых аналоговых элементов в схеме существенно (в несколько раз) замедляет процесс симуляции.

Овальным выделением на рисунке 2.13 показаны переключатели режимов работы программы и панели компонент и инструментов. При наведении указателя мыши на каждый элемент появляется подсказка.

Рассмотрим краткий алгоритм работы с программой для решения следующей задачи: создать минимальную схему подключения микроконтроллера ATMega8, подать на него питание, подключить кнопку и светодиоды, загрузить прошивку и пронаблюдать выполнение программы.

Первое. Для выбора устройства (детали) из библиотеки нажать кнопку (P) – «Pick from library». Откроется окно поиска компонент по библиотекам программы. Если известно наименование, можно ввести несколько букв названия компонента, это существенно ускорит процесс поиска (рисунок 2.14).

Найденный элемент отображается в списке элементов («Devices») и может быть добавлен в рабочую область. Аналогично добавляем резисторы («resistors»), конденсаторы «capacitors» - керамические, электролитические, светодиоды («led» - оптоэлектроника – «led-red»), кнопки «button»). Важные компоненты любой схемы – общий (т.н. «земля» - «ground») и питание («power» или «vcc») находятся в панели компонент (выделение на рисунке 2.13 «terminals»).

Второе. Соединения элементов выполняются мышью в режиме указателя (стрелочка). Факт соединения проводников в Proteus отображается круглой точкой на пересечении. Если точки нет – значит это просто пересечение без электрического контакта проводников.

Схема соединения элементов показана на рисунке 2.15. Стоит отметить, что в данной схеме светодиоды D1-D3 являются активными моделями, т.е. при симуляции они будут «загораться» и «гаснуть». Аналогично, используемая кнопка также является активной, ее можно нажимать как мышью, так и назначить в свойствах клавишу на клавиатуре для активации кнопки.

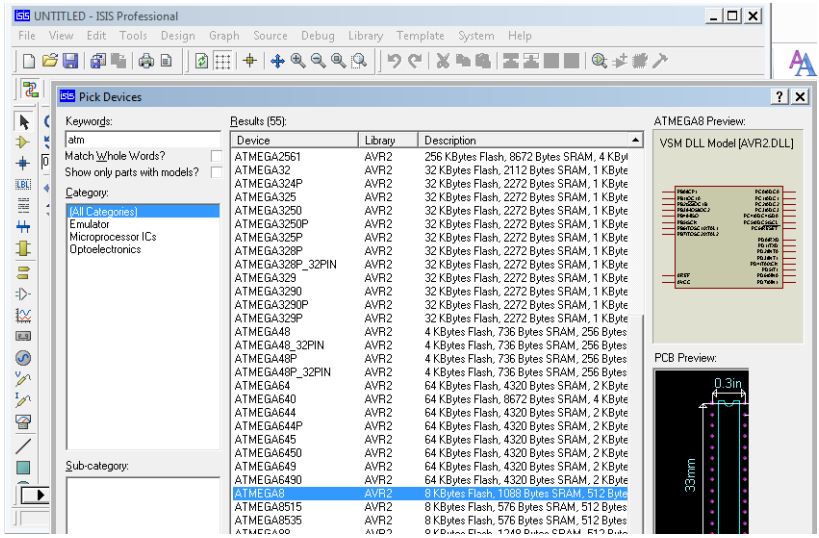


Рисунок 2.14. Поиск компонента в библиотеке

От вывода с номером 1 на общий включен конденсатор, а на источник питания – резистор с номиналом 10 КОм. Эта рекомендованная схема подключения вывода сброса микроконтроллера (Reset) для обеспечения стабильной работы МК.

Также на схеме показаны блокировочные конденсаторы по питанию, С2 – желательно использовать полярный с номиналом 10мкФ и С3 – керамический с номиналом 0,1мкФ.

Третье. На рисунке 2.16 показан выбор МК и тактовой частоты, на рисунке 2.17 – конфигурация портов ввода-вывода. В данной схеме порты PC0-PC2 устанавливаются как выходы, а порт PC3 – как вход с подтяжкой. Все остальные порты ввода-вывода, согласно рекомендациям производителя, указанным в Datasheet, установлены в режим входа с подтяжкой.

После того, как код написан, выполняется компиляция («Build all project files»). Если ошибки не выявлены, будет показано соответствующее окно – рисунок 2.18. Важно обращать внимание на соответствие типа МК, тактовой частоты, а также внимательно читать сообщения о наличии или отсутствии ошибок и предупреждений. В данном случае их нет (показано выделением).

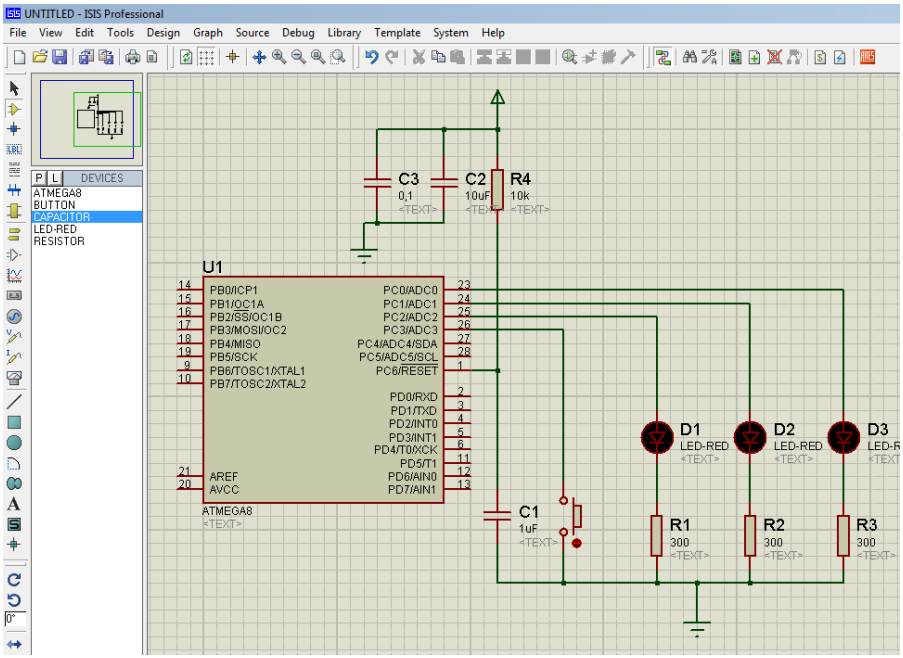


Рисунок 2.15. Схема соединений

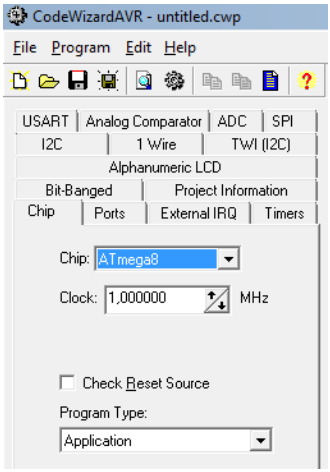


Рисунок 2.16. Выбор тактовой частоты МК

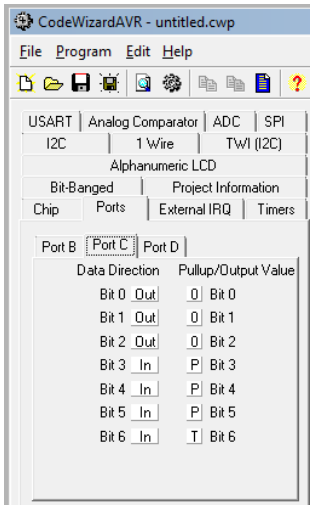


Рисунок 2.17. Конфигурация портов ввода-вывода

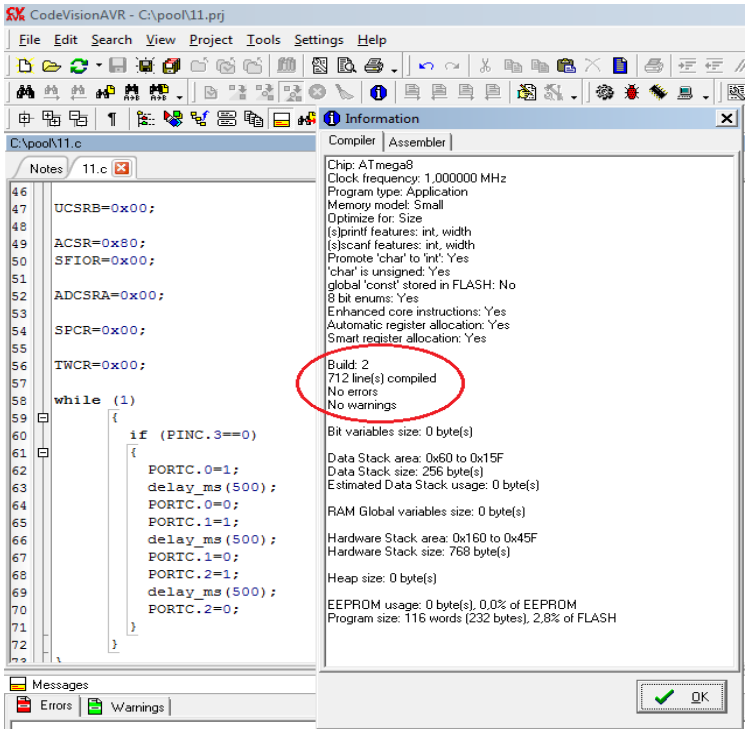


Рисунок 2.18. Результат компиляции файла прошивки

Четвертое. Результат успешной компиляции проекта – прошивку, файл формата Intel HEX с расширением .hex нужно загрузить в Proteus. Выполнить это можно щелкнув дважды ЛКМ на МК в Proteus, в строке «Program files» указать файл прошивки (рисунок 2.19).

Пятое. Для запуска симуляции нужно нажать кнопку «Play», похожую на кнопку воспроизведения звукозаписи. Если ошибок в схеме и прошивке не найдено, запустится процесс симуляции электронной схемы. Рядом с блоком кнопок будет надпись «Animating» и бегущее системное время, а также коэффициент загрузки центрального процессора ПК.

Процесс симуляции показан на рисунке 2.20. Обратите внимание на следующее:

Рядом со всеми выводами компонент появились синие или красные прямоугольники - метки. Красные означают напряжение,

соответствующее логической единице, в данном случае +5В. Синие – 0В;

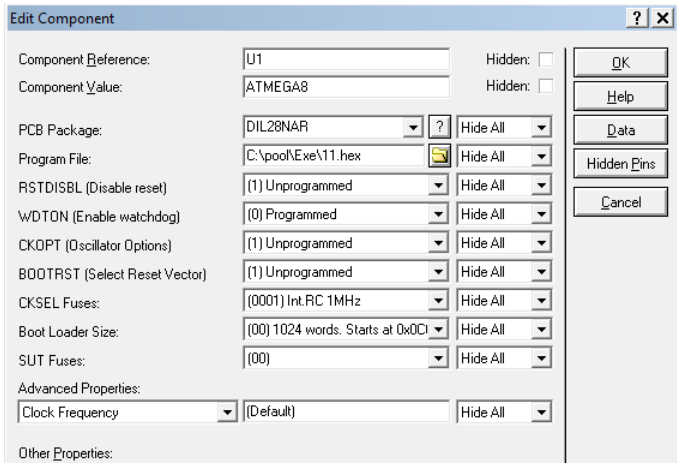


Рисунок 2.19. Выбор прошивки МК

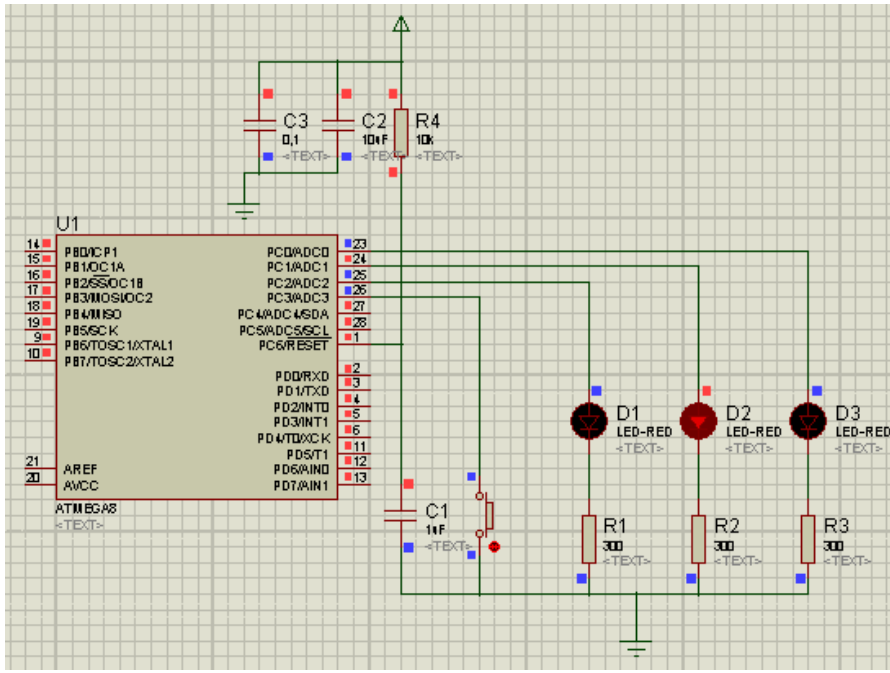


Рисунок 2.20. Выполнение программы МК (симуляция)

Также видно, что светодиод D2 (активная модель) в данный момент «горит» красным цветом, т.е. является активным в текущий момент времени. Это подтверждается красным цветом меток рядом с его выводами;

Все неиспользуемые выводы МК помечены красными метками. Это правильно, т.к. в настройках портов ввода-вывода было указано, что они конфигурируются как вход с подтяжкой. Ввиду отсутствия на этих выводах нагрузки, на них присутствует высокий потенциал (в данном случае +5В).

Внимательный читатель может усмотреть в схеме некоторую странность, а именно – отсутствие выводов питания у самого микроконтроллера. Ответ прост – в симуляторе Proteus выводы питания активных элементов (в частности, логических микросхем, микроконтроллеров и т.п.) являются скрытыми (без специального указания на схеме не отображаются, чтобы ее не загромождать), однако по умолчанию сама программа эти выводы подключает, соответственно, к общему проводнику и к питанию.

2.5. Лабораторная работа: Реализация программы управления «бегущими огнями» в графическом симуляторе. Простейший ввод данных.

Цель: Получить практические навыки работы с аппаратно-программным комплексом Arduino с помощью online-симулятора. Реализовать ввод данных в МК.

Оборудование: Персональный компьютер с выходом в Интернет.

Теоретическая часть.

Основные специальные знания, необходимые для решения поставленных задач, рассмотрены в предыдущих пунктах данного пособия.

Практическая часть.

1. На сайте симулятора (https://www.tinkercad.com/dashboard) зарегистрироваться;

2. Собрать схему, состоящую из Arduino Uno, 5 светодиодов и токоограничивающих резисторов, номинал которых выбрать исходя из питающего напряжения 5 вольт, напряжения падения на светодиоде 2,1 В и тока через светодиод 20 мА, двух кнопок и вольтметра. Схема показана на рисунке 2.21.

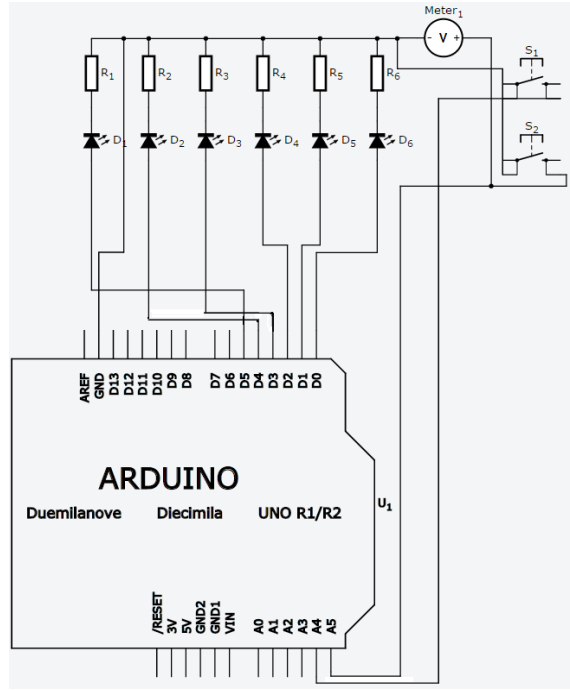


Рис. 2.21. Схема модели

Общий лабораторный вид собранной схемы показан на рисунке 2.22. Переключение между представлениями схемы осуществляется кнопками, расположенными над группой «1», показанной на рисунке.

3. Написать программу «Бегущий огонь», которая будет попеременно зажигать каждый из светодиодов на время 0,5 с., затем текущий светодиод должен гаснуть и, одновременно, зажигаться следующий. По достижении, к примеру, самого левого светодиода «огонь» переходит на самый правый светодиод и все повторяется в бесконечном цикле.

4. Модифицировать программу таким образом, чтобы после старта программа ожидала нажатия кнопки. При нажатии левой кнопки «огонь» должен двигаться влево, правой – вправо. Переключение направления движения может осуществляться в момент, когда «огонь» дошел до крайнего (левого или правого) светодиода.

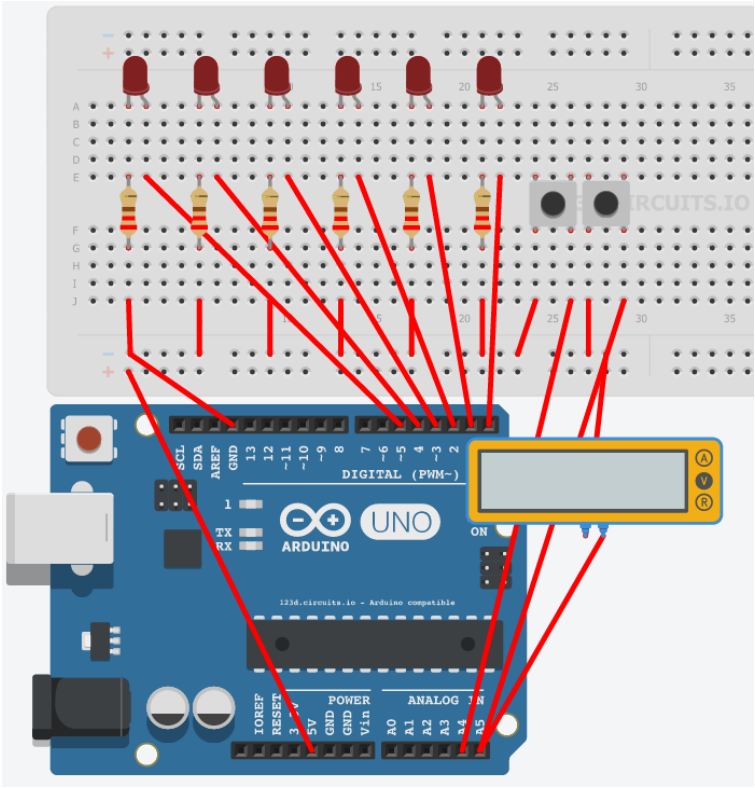


Рис.2.22. Лабораторное представление схемы

5. В тетрадь написать отчет, распечатать и вклеить схему и лабораторный вид собранной модели.

6. Нарисовать в тетради блок-схему алгоритма работы собственной программы, включая процедуру старта микроконтроллера и процесс ожидания прошивки встроенным загрузчиком.

7. Оформить отчет по работе, написать вывод.

2.6. Лабораторная работа Управление устройствами вывода информации. Управление семисегментным индикатором.

Цель: Получить практические навыки работы с аппаратно-программным комплексом Arduino, использовать его для управления семисегментым индикатором

Оборудование: Персональный компьютер с выходом в Интернет.

Теоретическая часть.

Основные специальные знания, необходимые в ходе выполнения работы, описаны в предыдущих разделах.

Практическая часть.

1. В online-симуляторе собрать схему, состоящую из Arduino Uno, семисегментного индикатора и токоограничивающих резисторов. Их номинал рассчитать известным способом. Общий лабораторный вид модели показан на рисунке 2.23.

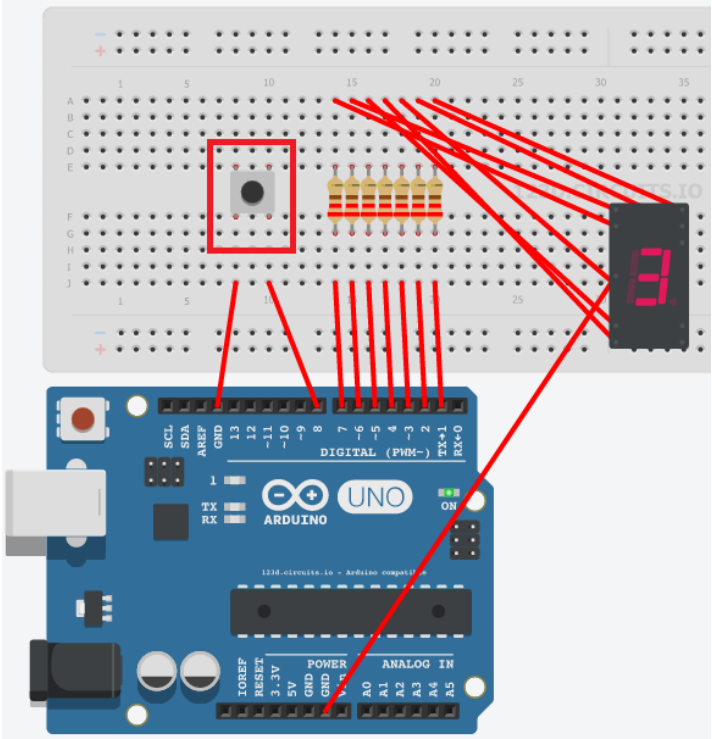


Рис. 2.23. Лабораторный вид схемы

Важно: токоограничивающие резисторы подключаются к выводам A-G индикатора и цифровым выходам Arduino Uno, выводы Com1 и Com2 индикатора требуется соединить вместе и подключить к Gnd Arduino Uno.

2. Создать программу, которая выведет на индикатор цифру «3».

3. Модифицировать программу таким образом, чтобы вывод требуемой цифры (от 0 до 9) управлялся отдельной процедурой,

которая, получая требуемое значение, будет активировать соответствующие разряды индикатора.

4. Модифицировать схему, добавив кнопку (обведено на рисунке 2.11). Модифицировать программу таким образом, чтобы цифровой вывод 8 Arduino был в режиме входа с подтяжкой. При однократном нажатии кнопки показания индикатора должны увеличиваться на 1.

5. Нарисовать в тетради блок-схему алгоритма работы собственной программы, включая процедуру старта микроконтроллера и процесс ожидания прошивки встроенным загрузчиком.

6. Оформить отчет по работе, написать вывод.

2.7. Лабораторная работа: Структурная организация Arduino. Изучение аппаратной и программной части

Цель работы: изучение технических характеристик микроконтроллерной платы Arduino Uno. Написание простой программы и программирование платы.

Необходимый уровень знаний и умений: знание основ микропроцессорной техники, умение ориентироваться в Datasheet на микроконтроллер и понимать представленную там информацию.

Дальнейшее развитие знаний и умений, полученных в ходе выполнения лабораторной работы: использование микроконтроллерной платы Arduino Uno для отладки программ в реальном микроконтроллере, использование микроконтроллерного блока в составе лабораторного стенда для считывания информации с клавиатуры 4x4, отдельных кнопок, вывода информации на ЖК экран 16x2, а также для организации связи между микроконтроллером и компьютером посредством протокола RS-232.

Оборудование и материалы: данное руководство, техническая документация на МК *ATMega328* (Datasheet), схема микроконтроллерного блока Arduino Uno, лабораторный стенд в составе: два микроконтроллерных блока Arduino Uno, блок ЖК 16 символов 2 строки и 6 кнопок управления, цифровая клавиатура 4x4, переходники для организации связи по протоколу RS-232 между микроконтроллерными блоками, кабель USB–USB тип B, лабораторный блок питания.

Теоретическая часть

Теоретическая часть подробно описана в соответствующем разделе пособия.

Практическая часть

1. В тетрадь выписать тему, цель, оборудование и материалы лабораторной работы.

2. В теоретической части – распечатать и прикрепить схему микроконтроллерной платы Arduino UNO Rev.3;

– описать особенности подачи питания на плату (через разъем USB и разъем внешнего питания), в т.ч. согласно контрольным вопросам;

– описать дополнительные (альтернативные) функции выводов Arduino Uno Rev.3 0-13;

– описать технические характеристики выводов Analog IN.

3. В практической части – создать программу по изменению состояния светодиода (LED, подключен к выводу 13) при нажатии внешней кнопки;

– произвести отладку программы и выполнить симуляцию проекта в VMLAB;

– осуществить программирование платы Arduino UNO Rev.3 (с использованием инструкции по программированию и программы-загрузчика).

4. Выполнить тестирование созданной программы.

5. Описать работу программы и ее особенности.

6. Написать вывод.

Дополнительное (бонусное) задание:

7. Скачать и установить программное приложении Eagle Cad, открыть в нем файлы схемы и топологии Arduino UNO Rev.3.

8. Добавить в схему элемент (к примеру, фильтрующий конденсатор между входом VIN и GND) и разместить его на печатной плате. Результат (схема, топология печатной платы) распечатать и прикрепить к работе.

Контрольные вопросы

1. Пояснить назначение элементов **T1**, **F1** на принципиальной схеме микроконтроллерного модуля Arduino UNO.

2. Пояснить назначение элемента **U2** на принципиальной схеме микроконтроллерного модуля Arduino UNO, привести характеристики выводов **VIN, 5V, 3V3, GND** (для ответа возможно использование принципиальной схемы).

3. Описать используемые типы памяти микроконтроллера (3), описать целевое использование каждого типа памяти и особенности

использования (скорость доступа, возможность перезаписи в разных режимах работы МК и т.п.).

4. Объяснить принципиальные различия между программированием с помощью загрузчика и ICSP. Почему при программировании с помощью загрузчика недоступны fuse-биты?

5. Укажите расположение основного (*ATMega328P*) и вспомогательного (*ATMega8U*) микроконтроллера на плате Arduino UNO, разъемов ICSP, разъемов для подключения периферии.

6. Можно ли, не отключая аппаратно (не выпаивая) системный кварцевый резонатор на 16 МГц, обеспечить другую тактовую частоту работы микроконтроллера? Что для этого нужно и какой тип программирования необходимо использовать?

7. Какие функции выполняет второй (не основной) микроконтроллер *ATMega8U2* на микроконтроллерной плате Arduino UNO?

8. Какой тип интерфейса между основным (*ATMega328P*) и вспомогательным (*ATMega8U*) микроконтроллером на плате Arduino UNO?

9. Можно ли на вход «*Analog*» микроконтроллерной платы подавать цифровой сигнал стандартных TTL уровней 5В и 3.3В? Что для этого необходимо сделать?

10.Какие функции выполняет программное приложение Eagle Cad?

2.8. Лабораторная работа: Динамическая индикация в управлении

Цель: Получить практические навыки работы с аппаратно-программным комплексом Arduino, использовать его для управления несколькими семисегментными индикаторами в режиме динамической индикации.

Оборудование: Персональный компьютер, плата Arduino Uno, модуль расширения №1.

Теоретическая часть.

Основные специальные знания, необходимые в ходе выполнения работы, описаны в предыдущих разделах, в частности в разделе, посвященном модулям расширения.

Практическая часть.

Работа выполняется с использованием микропроцессорного модуля Arduino Uno и модуля расширения №1, который подробно описан в соответствующем разделе данного пособия.

1. Для модуля расширения №1 адаптировать программу из лабораторной работы 2.7 с учетом изменений схемы и метода управления индикатором. Задача №1 – выводить на один семисегментный индикатор цифры от 0 до 9, увеличивающиеся при однократном нажатии на кнопку модуля.

Важно: для работы с семисегментным индикатором DIS0 достаточно замкнуть перемычку JP1POW7SEG, в таком случае подача низкого уровня на входы 0-7 будет приводить к активации сегментов A-G индикатора, стандартное расположение которых показано на рисунке 2.9.

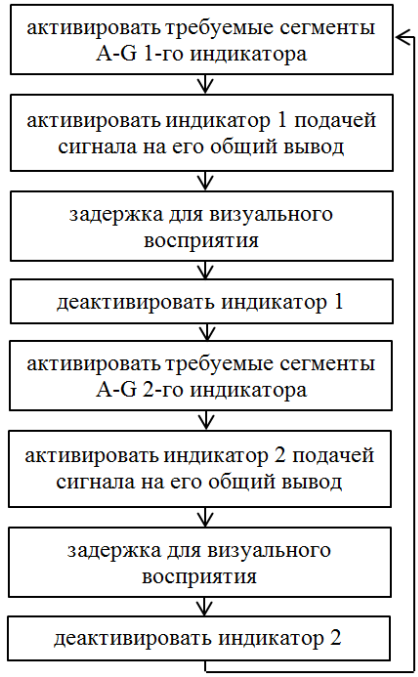


Рис. 2.24. Алгоритм динамической индикации с двумя индикаторами

2. Задача №2 – реализация принципа динамической индикации, при котором одна и та же группа выводов будет управлять двумя индикаторами, попеременно активируя один и другой элемент. Алгоритм динамической индикации показан на рисунке 2.24.

Важно: для полноценной работы с двумя индикаторами необходимо разомкнуть перемычку JP1POW7SEG, управление сегментами не изменяется (подача низкого уровня на входы 0-7 будет приводить к активации сегментов A-G индикатора), а активация того или иного индикатора осуществляется подачей низкого уровня на базы транзисторов T1 и T2 (входы 9,10).

3. Задача №3 – реализация цифрового счетчика. При нажатии и удерживании одной кнопки показания двух индикаторов должны раз в

0,5 с. инкрементироваться в диапазоне от 0 до 99, при нажатии другой кнопки – декрементироваться в диапазоне от 99 до 0.

4. Нарисовать в тетради блок-схему алгоритма работы собственной программы, включая процедуру старта микроконтроллера и процесс ожидания прошивки встроенным загрузчиком.

5. Оформить отчет по работе, написать вывод.

Раздел 3. Микропроцессорные системы на основе AVR AT Mega

Микропроцессор — устройство, выполняющее алгоритмическую обработку информации, и управление другими узлами электронной системы. История микропроцессоров в современном понимании началась в 1971 г., когда фирма Intel выпустила свой 4-разрядный микропроцессор, содержащий 2300 транзисторов, выполненный по Гарвардской архитектуре – Intel 4004. Тактовая частота составляла 0,75 МГц, тех.процесс 10 мкм.

Микропроцессорная система может рассматриваться как частный случай электронной системы, предназначенной для обработки входных сигналов и выдачи выходных сигналов.

Принципы фон-Неймана, которые легли в основу т.н. «фон-Неймановской» архитектуры микропроцессоров:

- принцип двоичного кодирования;
- принцип однородности памяти (программы и данные хранятся в одной и той же памяти, над командами можно выполнять такие же действия, как и над данными);
- принцип адресуемости памяти;
- принцип последовательного программного управления;
- принцип жесткости архитектуры.

Основное отличие т.н. «Гарвардской архитектуры» состоит в том, что в ней реализовано раздельное хранение и обработка команд и данных.

Основные функциональные направления развития микропроцессорной техники следующие:

- процессоры общего применения (для персональных компьютеров, мобильных устройств и т.д.);
- процессоры цифровой обработки сигналов (DSP-процессоры);
- микроконтроллеры.

Микроконтроллер – микросхема, сочетающая в себе функции процессора и периферийных устройств, как правило, содержит ОЗУ, ПЗУ, источники тактирования и набор периферийных устройств.

Область применения микроконтроллеров чрезвычайно широка, подавляющее число всевозможных автоматических устройств выполнено именно на микроконтроллерах.

При построении микроконтроллеров чаще всего используется Гарвардская архитектура, которая обеспечивает более рациональное использование аппаратных возможностей устройства при неизменности решаемых микроконтроллером задач.

Популярные семейства микроконтроллеров следующие:
- 8-битные МК: PIC (Microchip Technology), AVR (Atmel), MCS 51 (Intel);
- 16-битные МК: MSP430 (Texas Instrument);
- 32-битные МК: архитектура ARM (ARM Limited).

3.1. Техническое описание 8-битного микроконтроллера AVR ATMega

Микроконтроллер AVR ATMega содержит в себе центральный процессорный блок в составе АЛУ (арифметико-логического устройства), регистрового файла в составе 32 8-битных регистров общего назначения, счетчика команд и оперативной памяти со случайным доступом – RAM (random access memory). Оперативная память данного семейства МК является статической, что обеспечивает максимальное быстродействие подсистемы памяти. Оно необходимо ввиду того, процессор микроконтроллера построен по принципу RISC-архитектуры и выполняет по одной команде за такт.

Счетчик команд при запуске микроконтроллера обнуляется и выбирает первую команду из flash-памяти, в которой хранится программный код. Как видно из структурной схемы микроконтроллера (рисунок 3.1), доступ извне к flash-памяти программ может осуществляться по трехпроводному интерфейсу SPI (Serial Peripheral Interface), с его использованием доступна также электрически стираемая перепрограммируемая постоянная память EEPROM (Electrically Erasable Programmable Read-Only Memory), в которой программа может хранить сохраняемые значения, константы.

Центральный процессорный блок соединен с flash, eeprom и периферийными устройствами посредством внутренней шины.

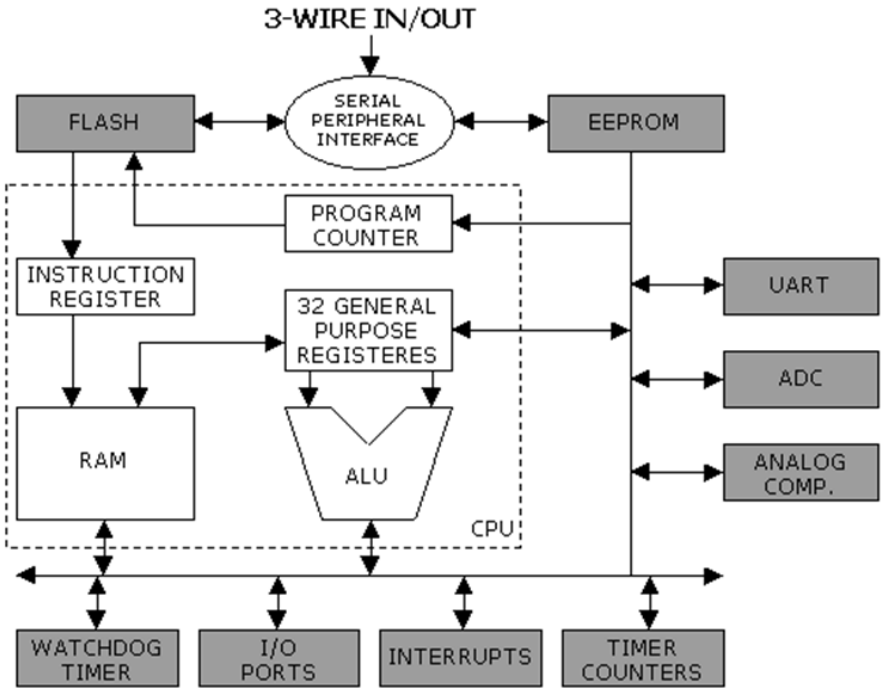


Рис. 3.1. Структурная схема AVR ATMega

Watchdog timer – изолированный аппаратный таймер со своим источником тактирования. Используется для контроля работоспособности (отсутствия зависания) основного процессора. Для этого в программный код внедряются специальный инструкции (например, wdt_reset();), которые сбрасывают счетчик. В случае зависания программы, счетчик сброшен не будет и, доведя счет до своего максимального значения перезагрузит микроконтроллер.

I/O Ports – порты ввода-вывода. Каждый 8-битный порт состоит из регистров PORTx, DDRx и PINx. Регистр PINx предназначен для считывания информации с порта, регистр PORTx и DDRx – для настройки и управления портом. Принцип работы с одним выводом порта с помощью указанных регистров показан на рисунке 3.2. [27]

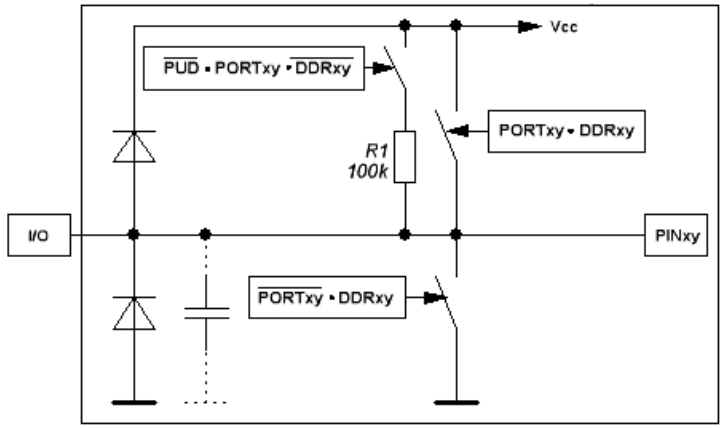


Рис. 3.2. Работа с регистрами порта ввода-вывода

Анализируя логическую схему на рисунке 3.2., можно составить таблицу режима работы порта ввода-вывода. Обозначения режимов в таблице следующие:
- IN, T – высокоимпедансный вход (т.е. с высоким входным полным сопротивлением). Считывание из регистра PINx(y);
- IN, P – вход с «подтяжкой», т.е. к высокоимпедансному входу добавлен резистор номиналом порядка 100 кОм, вторым выводом подключенный к положительному полюсу источника питания. Считывание из регистра PINx(y);
- OUT,0 – выход, по умолчанию с низким уровнем. Запись в регистр POTRx(y);
- OUT,1 – выход, по умолчанию с высоким уровнем. Запись в регистр POTRx(y).

Таблица 3.1. Режимы портов ввода-вывода

состояние	**PORTxy**	**DDRxy**
IN, T	0	0
IN, P	1	0
OUT, 0	0	1
OUT, 1	1	1

Interrupts – блок аппаратных прерываний, имеет значительные отличия у различных микроконтроллеров одного семейства.

Timer, Counters – блок аппаратных таймеров и счетчиков.

ADC, Analog Comp. – блок 12-битного аналого-цифрового преобразователя последовательного приближения (т.н. аналоговые входы в терминологии Arduino) и аналоговый компаратор.

UART – универсальный асинхронный приемник-передатчик (Universal asynchronous receiver/transmitter), часто используемый интерфейс для передачи данных внутри системы и на небольшие расстояния вне системы.

3.2. Побитовые операции, побитовые маски

В отличие от программирования под платформу x86, IA64, при написании программ для микроконтроллеров часто требуется управлять отдельными битами в байте или регистре. Некоторые компиляторы для этого предоставляют специальные, свои команды. Так, например, в CodeVisionAVR опрос состояния вывода 0 порта PORTB будет выглядеть следующим образом: `if PINB.0==0…` , а выставление высокого логического уровня на выводе 1 PORTC: `PORTC.1=1`.

В тоже время, не прибегая к использованию специальных конструкций, управление отдельными битами в байте возможно с использованием т.н. побитовых операций. Данная тема подробно и освещена на ресурсе [27], ниже приводятся основные положения и команды для управления отдельными битами.

Для выставления (т.е. установки в 1) отдельного бита в байте (регистре) можно использовать следующую конструкцию: `PORTB=1<<2`; `PORTB=1<<3`. В операции побитового сдвига «<<» слева записывается значение, а справа – номер бита, которые изменяется.

Важно: Операция идет с целым байтом, и в PORTB поочередно записывается число 00100000 (1<<2) и 00010000 (1<<3), перезаписывая друг друга. Поэтому, когда происходит запись одного значения, теряется предыдущее.

Важно: `PORTB=0<<1` это по факту просто 0 и такая конструкция не приносит никаких изменений. Особенно часто эта ошибка возникает у начинающих, когда они пытаются инициализировать периферию не всю сразу, а по мере надобности.

Учитывая вышеизложенное, для изменения отдельного бита в байте, не затрагивая соседние, требуется использование масок. Кратко маски представлены в таблице 3.2.

Таблица 3.2. Применение масок

	Установка бита	Сброс бита	Инверсия байта	Инверсия отдельных бит
Исходное значение	10001000	10001000	10001000	10001000
Операция	**OR (|)**	**AND (&)**	**NOT (~)**	**XOR (^)**
Битовая маска	00010000	01111111	n/a	11000000
Результат	10011000	00001000	01110111	01001000

Применяя на исходный байт битовую маску (байт с установленными в нужном порядке битами) можно сбросить, установить, или инвертировать любой бит, а также сразу инвертировать весь байт.

Важно: не следует путать поразрядные операции (например &) с логическими (&&). Первые действуют бит против бита, вторые – целиком весь байт, анализируя его, равен он нулю или нет. Если в байте будет хоть одна единица, то он уже не равен нулю.

Операция установки бита 2 (третьего по счету, счет начинается с 0) будет выглядеть так:

```
PORTB = PORTB  | 1 << 2;
```

Т.е. берется предыдущее значение порта PORTB и накладывается на него маску 1<<1, которая установит биты. Результат помещается обратно в PORTB.

Для сброса бита 2 маску нужно инвертировать, поэтому команда будет выглядеть так:

```
PORTB = PORTB & ~(1<<2);
```

Согласно приоритету выполнения операций вначале выполняется выражение в скобках (битмаска), потом оно инвертируется, а затем накладывается на значение порта. И результат записывается в прежнее значение.

Для инверсии бита 2 (0 меняется на 1, а 1 на 0) используется конструкция:

```
PORTB = PORTB ^ (1<<2);
```

Также Си допускает сокращенную конструкцию подобной записи логических операций (на примере бита 2):

```
PORTB &= ~(1<<2);
PORTB |= 1<<2;
PORTB ^=1<<2;
```

В качестве примера можно рассмотреть настройку АЦП, выставление логического «0» (Выбор канала ADC0):

ADMUX &= ~((1<<MUX4) | (1<<MUX3) | (1<<MUX2) | (1<<MUX1) | (1<<MUX0));

Выставление логической «1»:

ADMUX |= ((1<<REFS1) | (1<<REFS0));

Рекомендуемая последовательность записи при инициализации регистров оборудования – перечисление всех значащих бит вне зависимости от того нужно ли их выставлять в логический «0» или «1». На рисунке 3.3 приведены названия бит регистра TCCR0 аппаратного Timer0.

Bit	7	6	5	4	3	2	1	0	
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	TCCR0
Read/Write	W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Рис. 3.3. Названия бит регистра TCCR0 аппаратного Timer0

Рекомендуемая форма инициализации (для примера – Timer0, нормальный режим вывода, OC0 отключен, делитель = 256):

TCCR0 |= ((0<<FOC0) | (0<<WGM00) | (0<<COM01) | (0<<COM00) | (0<<WGM01) | (1<<CS02) | (0<<CS01) | (0<<CS00));

Альтернативный вариант:

TCCR0 |= ((0<<7) | (0<<6) | (0<<5) | (0<<4) | (0<<3) | (1<<2) | (0<<1) | (0<<0));

Важно: во втором (альтернативном) варианте вместо названий бит фигурирует их номер. С точки зрения компилятора эти записи абсолютно равнозначны, отличия лишь в том, что первый вариант лучше воспринимается человеком.

3.3 Лабораторная работа по изучению среды программирования CodeVisionAVR (AVRStudio) и симулятора VMLAB

Цель работы: изучение принципов построения программного проекта в среде программирования CodeVisionAVR (или AVR Studio),

изучение симулятора прошивки микроконтроллеров Atmel VMLAB и связки симулятор–среда программирования.

Необходимый уровень знаний и умений: знание основ программирования на Си, знание материалов курса «Компьютерная электроника».

Дальнейшее развитие знаний и умений, полученных в ходе выполнения лабораторной работы: используя полученные знания, научиться читать техническую документацию (Datasheet), создавать простые программы для МК *ATMega*, выполнять их проверку (симуляцию).

Оборудование и материалы: данное руководство, техническая документация на МК *ATMega16* (Datasheet), установленная на компьютер среда программирования CodeVisionAVR (AVR Studio), симулятор VMLAB.

Теоретическая часть.

CodeVisionAVR – кросс-компилятор Си, интегрированная среда разработки (IDE) и автоматический генератор программ на основе графического конфигуратора (CodeWizardAVR), разработанные для семейства AVR-микроконтроллеров фирмы Atmel.

CodeVisionAVR включает в себя следующие компоненты:

- компилятор Си-подобного языка для AVR;
- компилятор языка ассемблер для AVR;
- генератор начального кода программы, позволяющего произвести инициализацию периферийных устройств;
- модуль взаимодействия с отладочной платой STK-500;
- модуль взаимодействия с программатором;
- редактор исходного кода с подсветкой синтаксиса;
- терминал.

Выходными файлами CodeVisionAVR являются:

- HEX, BIN или ROM-файл для загрузки в микроконтроллер посредством программатора;
- COFF — файл, содержащий информацию для отладчика;
- OBJ — объектный файл.

Полное описание директив VMLAB находится в файле электронного методического комплекса дисциплины «Микропроцессорные системы»: «*Проектный файл Директивы VMLAB.doc*»

Практическая часть.

1. В тетрадь выписать тему, цель, оборудование и материалы лабораторной работы.
2. В тетрадь записать техническое задание на создание программы следующего вида:
   – с помощью языка программирования Си для микроконтроллера ATMega16 написать программу, реагирующую на нажатие 3 клавиш, подключенных к выводам одного порта, результат работы программы вывести на три светодиода, подключенные к выводам другого порта;
   – в техническом задании описать тип подключения кнопки (фрагмент схемы), тип подключения светодиода (фрагмент схемы), логику работы программы (какая индикация соответствует комбинации нажатых кнопок).
3. Создать (написать) данную программу, скомпилировать.
4. Создать (отредактировать) файл проекта VMLAB для возможности загрузки программного проекта (прошивки) в симулятор, подключить периферию в VMLAB согласно технического задания (кнопки, светодиоды).
5. Запустив VMLAB, собрать проект и выполнить симуляцию. Проверить соответствие действий программы техническому заданию.
6. В тетради зарисовать (записать) алгоритм работы программы.
7. Написать вывод.

Контрольные вопросы

1. Какие Вам известны типы алгоритмов?
2. Какими способами можно представить алгоритм?
3. Что представляет собой формат файла .hex?
4. Можно ли запрограммировать отдельно энергонезависимую память EEPROM?
5. Какой файл в качестве входного (программного) использует VMLAB, по которому осуществляется симуляция?
6. Каким образом (с помощью чего) VMLAB дает возможность наблюдать выполнение программы по тексту программного файла, написанного на языке Си?
7. В чем отличия реализации синтаксиса языка Си среды программирования CodeVisionAVR и классического Си (к примеру, в AVR Studio)?

8. Какой набор виртуальной периферии предоставляет пользователю стимулятор VMLAB?
9. Допустимо ли графическое конфигурирование периферии в VMLAB? Приведите пример подключения виртуального осциллографа для наблюдения сигналов на выводах PORTB0, PORTB1 и PORTC4.
10. Объясните функцию регуляторов (в виде списка) приведенных на рис. 3.4. Рассчитайте временные и амплитудные характеристики сигнала

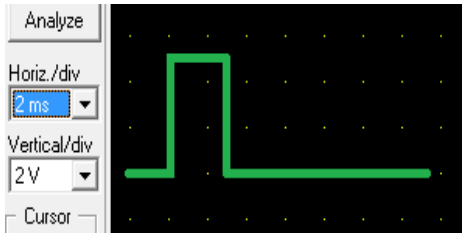


Рис. 3.4. Пример сигнала

11. Какую функцию позволяют реализовать ползунковые регуляторы S1, S2, S3?
12. Чем отличаются действия «Light restart» и «Deep restart» в VMLAB?

3.4. Лабораторная работа: Управление динамическим объектом с обратной связью. Реализация программы управления роботом, движущимся по линии.

Цель работы: изучение технических характеристик мобильной платформы робота для гонок по линии, методов управления мобильной платформой. Написание простой программы и программирование робота.

Необходимый уровень знаний и умений: знание основ микропроцессорной техники, умение ориентироваться в Datasheet на микроконтроллер и понимать представленную там информацию.

Дальнейшее развитие знаний и умений, полученных в ходе выполнения лабораторной работы: полученные знания и умения найдут свое приложение в конструировании более сложных управляющих систем на микроконтроллерах.

Оборудование и материалы: данное руководство, техническая документация на МК *ATMega8* (Datasheet), программатор USBASP, программа AVR DUDE, кабель USB–USB тип B, лабораторный блок питания.

Теоретическая часть
Шасси следящего робота с микроконтроллерным управлением

Соревнования среди роботов в дисциплине «гонки по линии» набирают все большую популярность, в том числе и в студенческой среде. Причин несколько: кроме того, что гонки роботов – это интересно, в данном виде соревнований присутствует значительная доля технической мысли как при создании платформы-робота, так и при написании для него управляющей программы.

Основная идея соревнования заключается в том, что абсолютно автономный (без какого либо внешнего управления) мобильный (т.е. самодвижущийся) робот должен проехать круг за минимальное время вдоль нарисованной на белом фоне черной линии, повторяя все ее изгибы. Покидать всеми колесами линию (или покидать ее более чем на 3-5 секунд) правила запрещают. Существует целый ряд различных регламентов, более подробно с одним из них можно ознакомиться, прочитав файл электронного методического комплекса *«Регламент соревнований СЛЕДОВАНИЕ ПО ЛИНИИ Кубок политехнического музея 2011.pdf»*

Для соревнований используют различные аппаратные платформы. Как правило, шасси робота имеет два ведущих колеса, каждое приводится в движение отдельным двигателем (через редуктор или без него). Для регистрации факта нахождения над черной линией используются от 2 до 5-6 оптических сенсоров, иногда применяют инфракрасные сенсоры (ИК). Но применение ИК сенсоров осложняется тем, что некоторые виды черной краски (чернил струйных принтеров) не достаточно хорошо отражают ИК излучение, тем самым нарушая работу сенсорного блока.

Сенсорный блок позволяет отслеживать положение робота относительно черной линии. К примеру, если робот сместился влево (левый сенсор вышел за границу черной линии), то в левый сенсор будет попадать отраженный белой поверхностью сигнал, который передается на микроконтроллер управления. В тот сенсор, который находится над черной линией, отраженный сигнал не попадает (либо значительно ей ослабляется). На этом принципе основана регистрация положения робота в пространстве.

В некоторых случаях применяют более совершенные методы контроля положения робота. Это может быть как увеличение числа сенсоров (до 5-6), так и применение видеокамер, которые, находясь на борту робота, в реальном режиме времени сканируют положение

линии, длину прямой, на которой робот может разгоняться, наличие поворота, его направление и крутизну и т.п. Для достижения лучших показателей используют ПИД-регуляторы.

Практическая часть

Рассматриваемое шасси робота представляет собой двухколесную платформу, привод выполнен на каждое колесо в отдельности (левое и правое). Поворот осуществляется включением одного из двигателей (по принципу танка: поворот направо – включен левый двигатель и т.п.), движение прямо – включением обоих двигателей.

Характеристики платформы:

Длина	14 см
Ширина	12 см
Скорость движения	10 см/с
Напряжение питания, номинал	4.5В
Напряжение питания, минимум	3.3В
Потребляемый ток, максимум	0.5 А
Потребляемый ток, среднее значение	0.15 А

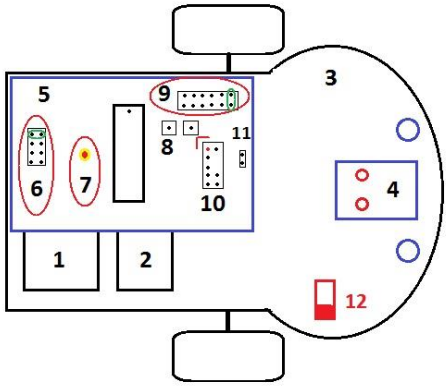


Рис. 3.5. Схема шасси

Элементы робота следующие (рис. 3.5.)**:**

1 – кассета на 3 стандартных пальчиковых батарейки АА;
2 – один из двух двигателей (правый);
3 – основная плата (платформа) робота;
4 – блок оптических датчиков линии (направлены вниз);
5 – управляющая плата микроконтроллера и драйверов двигателей;
6 – разъем для подключения двигателей, зеленым показана пара контактов для одного двигателя;
7 – светодиод-индикатор включения управляющей платы;
8 – программируемые кнопки (2 шт) для пользовательского управления программой;
9 – разъем подключения датчиков, зеленым показана пара для подключения одного датчика, нижний на рисунке вывод – общий;
10 – разъем ISP. **Красной точкой и уголком показан пин №1, который должен совпадать с таким же на программаторе;**
11 – джампер (перемычка) питания управляющей платы от программатора. **При замкнутой перемычке перед подключением**

программатора НЕОБХОДИМО отключить двигатели от разъема 6;
12 – выключатель питания робота.

Постановка задачи: управляющая программа микроконтроллера должна обеспечить следование вдоль черной линии шириной 1,8см, выполненной на белом фоне (бумаге). План трассы изображен на рисунке 3.5.

Схемотехника

Входными сигналами являются сигналы от датчиков линии (4). Схема подключения разъема датчиков (9) к микроконтроллеру приведена на рис. 3.6 (к выводу PC5 – правый датчик, PC4 – левый датчик, PC3, PC2, PC1, PC0 – запасные).

Управляющие кнопки подключены аналогично к выводам контроллера PB1 (левая) и PB2 (правая).

Выходы контроллера, которые управляют включением двигателя (подачей на них логической 1) – PD0 (правый двигатель), PD1 (левый двигатель).

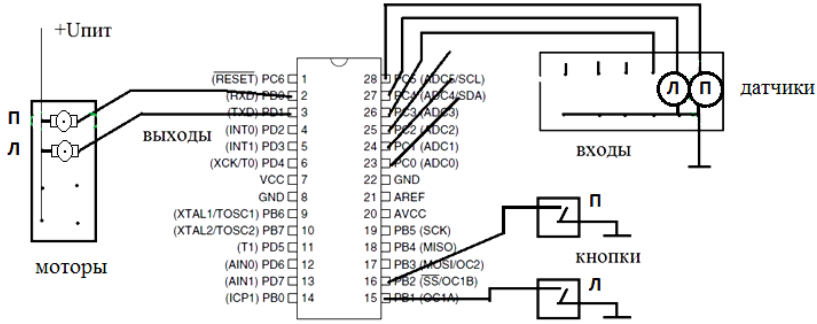


Рис. 3.6. Схема подключений управляющей платы робота

Задание

1. Для данного робота написать управляющую программу, которая обеспечит автономное прохождение трассы, начиная с линии старта по направлению стрелки и, как минимум, до черты (вверху).

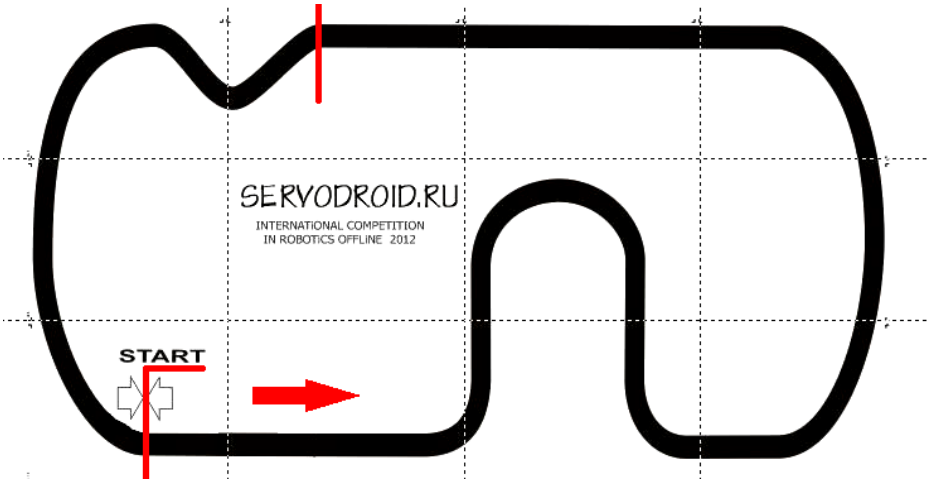


Рисунок 3.7. Траектория трассы

2. Данный путь робот должен пройти автономно за время **t** не более 3 минут. В ходе попытки допускается кратковременная помощь в виде непродолжительных корректировок траектории в числе не более трех раз до красной черты, или не более шести корректировок для полной дистанции от старта до старта.
3. Создать алгоритм управляющей программы. Учесть, что использование полной мощности двигателей будет приводить к очень быстрому движению и, как следствие, к «проскакиванию» точек поворотов. Таким образом, необходимо предусмотреть программный механизм запоминания направления поворота или ограничение мощности двигателей.

Контрольные вопросы

1. Каковы особенности регламента соревновании «Следование по линии»?
2. Опишите проблемы использования ИК-сенсоров.
3. ПИД-регулятор. Основные принципы работы. Простейший ПИД-регулятор.
4. Пропорциональное управление. Перерегулирование.
5. Интегральное управление в ПИД-регуляторе.
6. Дифференциальное управление в ПИД-регуляторе.

7. Особенности рассматриваемого шасси робота. Подключение периферии.
8. Опишите настройки портов микроконтроллера для датчиков, кнопок и управлением двигателями.
9. Нарисуйте блок-схему возможного алгоритма программы с запоминанием направления поворота.
10. Нарисуйте блок-схему возможного алгоритма программы с ограничением мощности двигателей.

3.5 Лабораторная работа: Программная реализация конечного цифрового автомата в системах управления малой автоматизации

Цель работы: изучение принципов использования цифрового автомата в микропроцессорных устройствах управления.

Необходимый уровень знаний и умений: знание основ программирования на Си, знание материалов курса «Компьютерная электроника».

Дальнейшее развитие знаний и умений, полученных в ходе выполнения лабораторной работы: используя полученные знания, научиться создавать управляющие алгоритмы на основе автоматного подхода.

Оборудование и материалы: данное руководство, техническая документация на МК *ATMega16* (Datasheet), установленная на компьютер среда программирования CodeVisionAVR (AVR Studio), симулятор VMLAB.

Теоретическая часть.

Создание программ для микроконтроллеров при решении некоторых задач может в корне отличаться от программирования под операционные системы, например, семейства Windows. В последнем случае, чаще всего, программа строится вокруг, или, по крайней мере, с учетом функционирования графического интерфейса. При этом вводятся такие понятия, как сообщение, событие и т.п., которые играют роль управляющих элементов в программе.

Программирование систем управления на основе микропроцессорной техники часто требует создания программы, которая сама будет заниматься управленческими функциями и работать непосредственно на аппаратном обеспечении.

Рассмотрим типичный пример: требуется создать контроллер простого робота, который будет иметь несколько режимов работы. В основном режиме робот должен обрабатывать ряд датчиков,

принимать данные и трансформировать их в выполнение команд. В режиме зарядки батареи робот должен находиться в покое, и ждать только окончания зарядки или ее принудительного прерывания человеком. В режиме сна робот должен ожидать поступления некоторых, выделенных команд и реагировать только на определенные сигналы с датчиков.

Попытка создания программы управления для решения указанной задачи типовыми методами приведет к необходимости постоянного перебора всех вариантов или она окажется громоздкой и сложной в понимании и, что важно, отладке.

Выходом из такой ситуации является представление процессов в виде диаграммы состояний, пользуясь, например, языком UML. Пример такой диаграммы для описанной задачи приведен ниже.

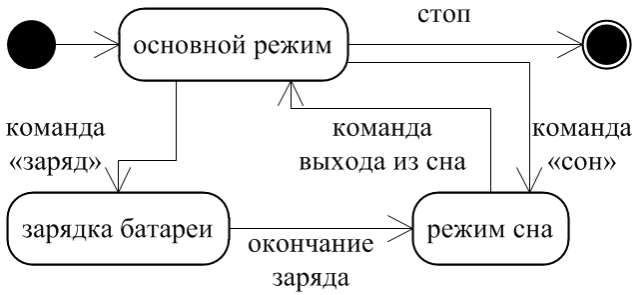


Рис. 3.8 Диаграмма состояний

Пользуясь разделением технологического процесса на явно выделенные состояния и условия перехода между ними можно записать управляющую программу в этих же формах. При этом следует учитывать тот факт, что в большинстве микропроцессорных систем управления как малой автоматизации, так и в программируемых логических контроллерах (ПЛК) основной блок программы выполняется циклически. Этот факт требуется учесть при написании программы.

Для создания программы с использованием выделенных состояний применяется оператор switch-case.

Switch-технология – технология для поддержки автоматного программирования (**технология автоматного программирования**), была предложена А.А. Шалыто в 1991 году [активная дискуссия по данному вопросу в 28]. Она охватывает спецификацию, проектирование, реализацию, отладку, документирование и сопровождение программ. При этом под термином «автоматное

программирование» понимается не только построение и реализация конечных автоматов, но и проектирование и реализация программ в целом, поведение которых описывается автоматами.

Поведение автоматов задается графами переходов (диаграммами состояний), на которых для их компактности входные и выходные воздействия обозначаются символами, а слова используются только для названий пронумерованных состояний. Расшифровка символов выполняется на схеме связей. Применение символов позволяет изображать сложные графы переходов весьма компактно — так, что человек может в большинстве случаев охватить каждый из них одним взглядом. Это обеспечивает когнитивное восприятие указанных графов.

Пример основного блока программы с использованием автоматного подхода показан ниже.

```
while(1)
{
   XXX                        // Считывание входных переменных управляющего автомата
   switch (State)             // Основной автомат программы
   {
      case (0):               // Явное выделение нач. состояния
      {
         State = 1;            // Безусловный переход в состояние 1
         if ( State == 1 ) XXX // Выходное действие состояния 0
      };
      break;                  // Конец состояния 0 - Начальное
      case (1):               // Состояние 1 - Основной режим
      {
         if ( firstEnterto1 )    // Входное действие состояния 1
         {
            firstEnterto1=0;
            XXX                  // Входное действие состояния 1
         };
         XXX                     // Действия состояния 1 - считывание датчиков, выработка
                                 // команд, контроль выхода из состояния
         if ( Charge ) { State=2; firstEnterto1=1; }  // Если "Зарядка" - переход в (2)
         if ( Sleep )   { State=3; firstEnterto1=1; }   // Если "Сон" - переход в (3)
         if ( Stop )    { State=4;  firstEnterto1=1; }   // Если "Стоп" - переход в (4)
      };
      break;                  // Конец состояния 1 - Основной режим
      case (2):               // Состояние State = 2 - Зарядка
      {
         if ( firstEnterto2 )    // Входное действие состояния 2
         {
            firstEnterto2=0;
            XXX                  // Входное действие состояния 2
         };
         XXX       // Действия состояния 2 - контроль зарядки и выхода из состояния
         if ( ChargeEnd ) { State=3; firstEnterto2=1; }// Если зарядка окончена - в  (3)
      };
      break;                  // Конец состояния 2 - Зарядка
```

```
      case (3):                  // Состояние State = 3 - Сон
      {
        if ( firstEnterto3 )     // Входное действие состояния 3
        {
          firstEnterto3=0;
          XXX                    // Входное действие состояния 3
        };
        XXX        // Действия состояния 3 - контроль команд выхода из сна
        if ( SleepEnd ) State=1;          // Если выход из сна - переход в состояние (1)
        if ( State == 1 ) {               // Выходное действие состояния 3
          firstEnterto3=1;
          XXX                             // Выходное действие состояния 3
        };
      };
      break;                     // Конец состояния 3 - Сон
      case (4):                  // Состояние State = 4 - Стоп
      {
        if ( firstEnterto4 )     // Входное действие состояния 4
        {
          firstEnterto4=0;
          STOP_ALL;          // Входное действие состояния 4 - Остановка всего
        };
      };
      break;                     // Конец состояния 4 - Стоп
      default: State = 0;        // В случае нештатной ситуации начать с состояния 0
    };                           // конец Switch
};                               // конец while(1)
```

Практическая часть.

1. При неизменной аппаратной части робота для движения по линии, создать диаграмму состояний, алгоритм управляющей программы.

2. Для данного робота написать управляющую программу с использованием switch-технологии (автоматного подхода), которая обеспечит автономное прохождение трассы, начиная с линии старта по направлению стрелки и, как минимум, до черты (вверху)

3. С применением автоматного подхода добавить функцию запоминания поворотов в диаграмму состояний и реализовать практически в программе управления роботом функцию, когда при потере линии в повороте робот будет продолжать поворачивать до тех пор, пока линия не будет найдена.

4. Оформить диаграмму состояний, блок-схему алгоритма в тетради, записать вывод.

3.6. Лабораторная работа: Тестирование, отладка и настройка программы движения по линии. Мини-соревнование

Цель работы: практическое сравнение разных подходов к программированию встраиваемых систем, отладка и настройка программ управления для робота.

Необходимый уровень знаний и умений: знание основ программирования с использованием процедурного и автоматного подходов, базовые знания по схемотехнике.

Дальнейшее развитие знаний и умений, полученных в ходе выполнения лабораторной работы: оптимизация программы с учетом физических свойств объекта управления.

Оборудование и материалы: данное руководство, робот для езды по линии.

Практическая часть

1. Выполнить тестирование программ на основе процедурного и автоматного подхода;
2. Добиться стабильного прохождения всей трассы без внешней помощи;
3. За отведенное команде время добиться максимально быстрого прохождения трассы.

3.7. Лабораторная работа: Генерация ШИМ встроенным аппаратными средствами МК

Цель работы: изучение режимов аппаратной генерации широтно-импульсной модуляции, создание программы для генерации 1,2 и 3-х фазной ШИМ. Создание программного модуля для отсчета произвольных промежутков времени.

Необходимый уровень знаний и умений: знание основ микропроцессорной техники, умение ориентироваться в Datasheet на микроконтроллер и понимать представленную там информацию, базовые знания по схемотехнике.

Дальнейшее развитие знаний и умений, полученных в ходе выполнения лабораторной работы: использование сигналов ШИМ для управления исполнительными устройствами. Программный модуль генерации произвольных промежутков времени целесообразно использовать для запуска задач и т.п.

Оборудование и материалы: данное руководство, техническая документация на МК *ATMega328* (Datasheet), схема микроконтроллерного блока Arduino Uno, лабораторный стенд в составе: два микроконтроллерных блока Arduino Uno, блок ЖК 16

символов 2 строки и 6 кнопок управления, цифровая клавиатура 4x4, переходники для организации связи по протоколу RS-232 между микроконтроллерными блоками, кабель USB–USB ТИП В, лабораторный блок питания.

Теоретическая часть

Широтно–импульсная модуляция (**PWM** – **Pulse Width Modulation**) – это способ задания аналогового сигнала **цифровым методом**. Таким образом, **ШИМ** – методика формирования сигнала, основанная на изменении скважности.

Если проинтегрировать полученный сигнал, будет получен аналоговый сигнал с некоторым числом характеристических точек.

Частота следования импульсов – это количество полных импульсов в единицу времени.

Период импульсов – это промежуток времени, между двумя характерными точками двух соседних импульсов **T=1/F.**

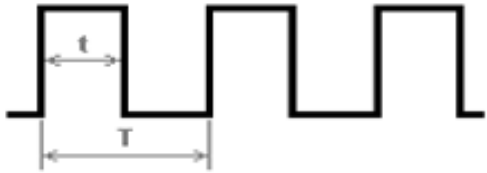


Рис. 3.9. Пример сигнала

Если T = 2t, сигнал называется **меандр.**

Скважность импульсов – отношение периода следования импульсов **T** к их длительности **t** (**S=T/t**).

Коэффициент заполнения (Duty cycle) обратная величина к скважности **(D=1/S)**

Цепочка-интегратор изображена на рис. 3.10.

Эквивалентные значения постоянного напряжения приведены на рис. 3.11.

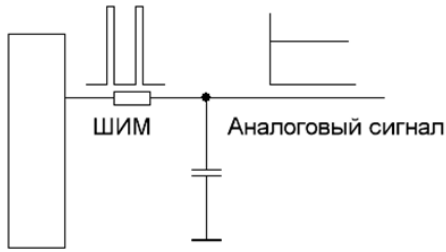


Рис. 3.10. Схема интегратора

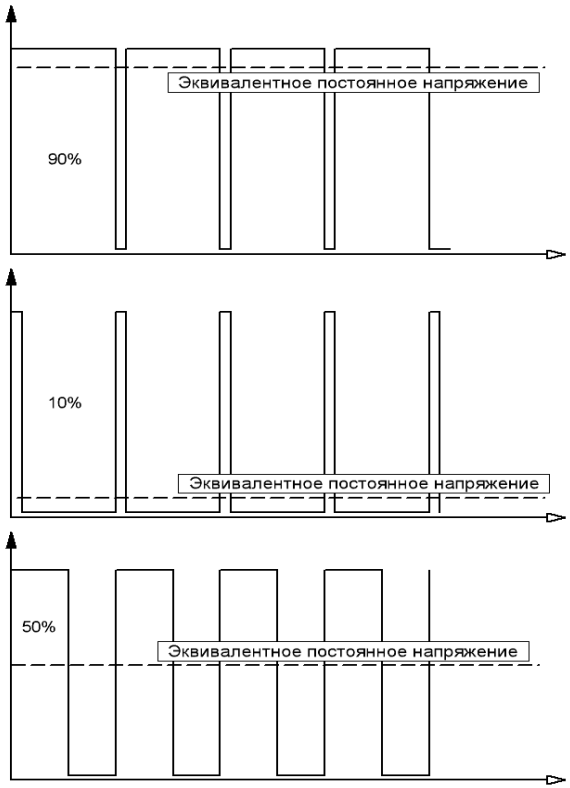


Рис. 3.11. Эквивалентное постоянное напряжение ШИМ сигнала

Механическая аналогия следующая: тяжелый маховик, который вращается посредством двигателя, причем двигатель можно либо включить, либо выключить. Если включить его постоянно, то маховик раскрутится до максимально возможной угловой скорости и будет продолжать вращаться. Если двигатель выключить, то маховик со временем остановится за счет сил трения.

А если двигатель включать на десять секунд каждую минуту, то маховик раскрутится, но не на полную скорость — большая инерция сгладит рывки от включающегося двигателя, а сопротивление от трения не даст ему раскрутиться до бесконечно большой скорости. Иными словами, инерция сглаживает рывки (включение двигателя), а сила трения ограничивает максимальную скорость вращения.

Чем дольше **продолжительность включения** двигателя в минуту, тем быстрей будет крутиться маховик.

При использовании **ШИМ** на выходе образуется сигнал, состоящий из высоких и низких уровней (применимо к используемой аналогии — включение и выключение двигателя), то есть нулей и единиц. Затем сигнал пропускается через интегрирующую цепочку (в аналогии — маховик). В результате интегрирования на выходе будет получена величина напряжения, равная площади под импульсами.

Меняя **скважность** (отношение длительности периода к длительности импульса), можно плавно менять эту площадь, а значит и напряжение на выходе.

Аппаратная генерация ШИМ в МК ATMega

Микроконтроллеры семейства ATMega имеют, как правило, несколько аппаратных таймеров/счетчиков с возможностью аппаратной генерации ШИМ сигнала. На рисунке отмечены желтым такие выводы.

Принцип работы таймера/счетчика в режиме генерации ШИМ описан ниже.

У таймера есть особый регистр сравнения **OCR**** и регистр счетчика **TCNT***, значение в котором инкрементируется каждый такт счетчика.

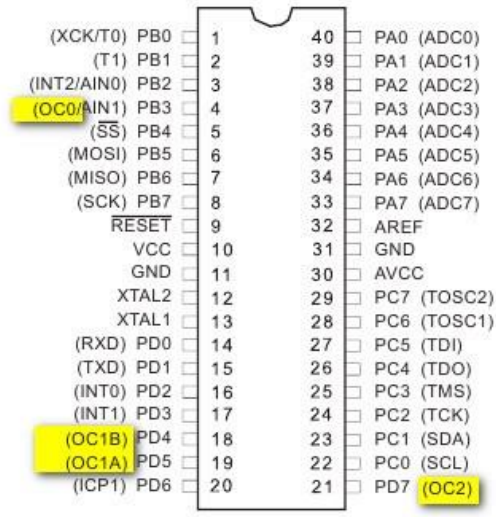


Рис. 3.12. Выводы ATMega

Когда значение в счётном регистре таймера **TCNT*** достигает значения в регистре сравнения **OCR****, могут возникнуть следующие аппаратные события:

а) Прерывание по совпадению;

б) Изменение состояния внешнего выхода сравнения **OC**** (отмечено на рисунке).

Упрощенно изменение значения в счетном регистре **TCNT***, значение регистра сравнения **OCR**** и состояния выхода **OC**** изображено на рис. 4.5.

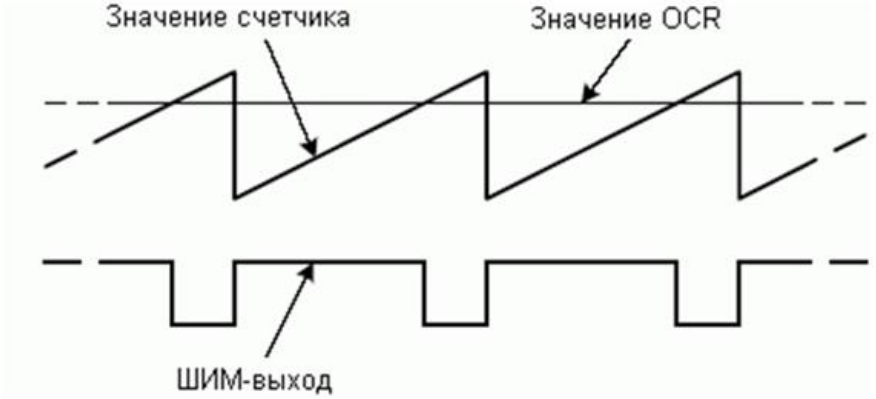


Рис. 3.13. Значения регистров и состояние выхода **OC****

Можно настроить ШИМ генератор так, чтобы в случае, когда значение в счетном регистре больше чем в регистре сравнения, на выходе будет логическая 1, а когда меньше – 0. Рис. 3.14 поясняет работу ШИМ генератора при текущей настройке (первый вариант).

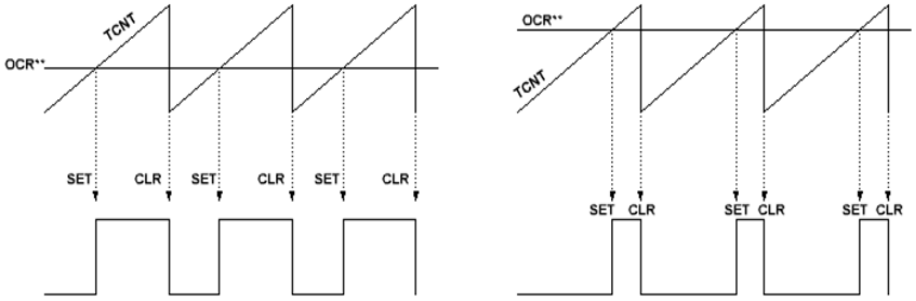


Рис. 3.14. Настройка ШИМ, первый вариант

Очевидно, что при выбранном первом варианте настройки ШИМ генератора при записи в регистр **OCR**** максимально возможного значения (0xFF или 255 для 8-битного регистра) на выходе ШИМ генератора будет постоянно логический 0.

В тоже время, можно выбрать обратную настройку (второй вариант), когда в моменты совпадения значений регистров **TCNT*** и

OCR** состояние выхода ШИМ будет сбрасываться в 0 (clear), а в моменты начала счета заново устанавливаться в 1.

При выбранном втором варианте настройки ШИМ генератора при записи в регистр **OCR**** максимально возможного значения (0xFF или 255 для 8-битного регистра) на выходе ШИМ генератора будет постоянно логическая 1 или максимальный уровень аналогового сигнала после цепочки интегратора.

Режим генерации «быстрого ШИМ» (Fast PWM Mode) может быть описан следующим алгоритмом:

- Счетчик считает от **0** до **255**, после переполнения сбрасывается в **0** и счет начинается снова;
- Когда значение в счетчике (**TCNT***) достигает значения регистра сравнения (**OCR***), соответствующий вывод **OCR**** сбрасывается в ноль;
- При обнулении счетчика вывод **OCR**** устанавливается в **1.**

Графическая иллюстрация режима «Быстрый ШИМ» приведена на рис. 3.15.

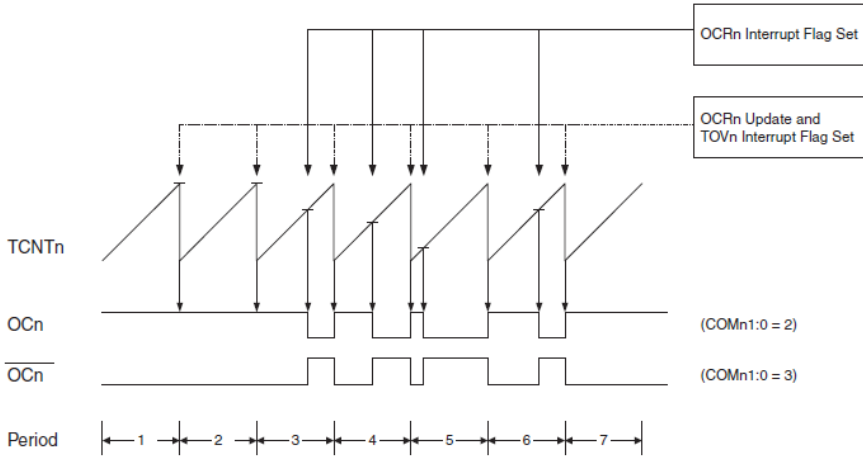


Рис. 3.15 Генерация в режиме «Быстрый ШИМ»

Режим генерации «ШИМ с постоянной фазой» (Phase correct PWM Mode) описывается следующим алгоритмом:

- Счетчик считает **от 0 до 255, потом от 255 до 0;**
- Вывод **OC**** при первом совпадении сбрасывается, при втором устанавливается (~).

Графическая иллюстрация режима генерации «ШИМ с постоянной фазой» приведена на рис. 3.16.

Несмотря на преимущество данного метода, проявляющееся в том, что фаза прямоугольного сигнала не меняется (т.е. не смещается во времени центр импульса), максимально возможная частота ШИМ при данном режиме генерации падает вдвое.

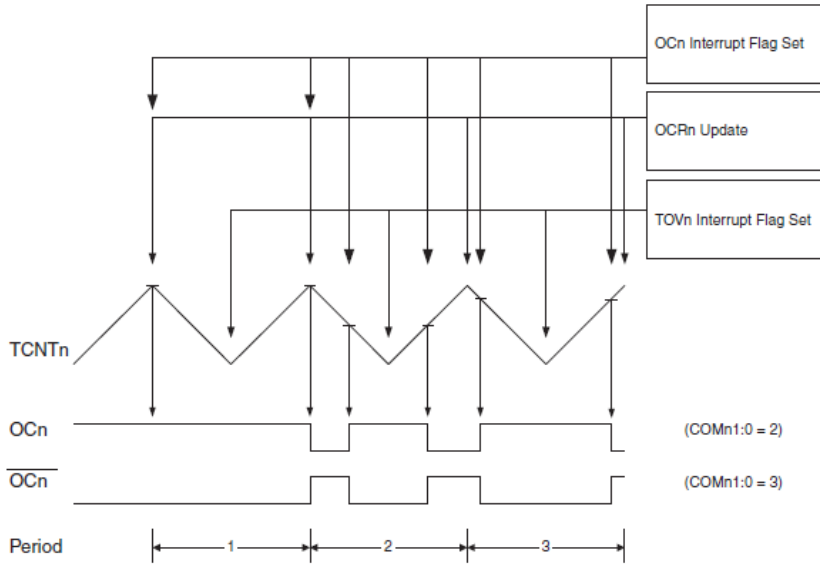


Рис. 3.16. Генерация ШИМ в режиме с постоянной фазой

Основное предназначение данного режима — генерация многофазных ШИМ сигналов, например, трехфазной синусоиды для управления синхронными двигателями.

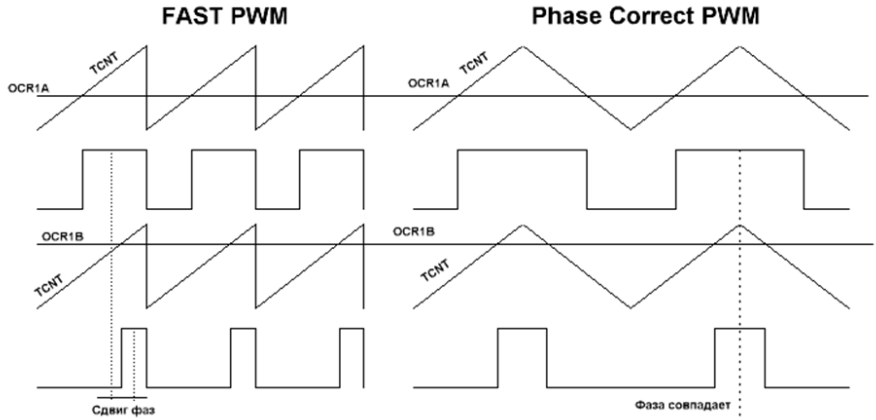


Рис. 3.17. Сравнение режимов генерации Fast PWM и Phase Correct PWM

Режим генерации ШИМ «Сброс при сравнении» (Clear Timer On Compare (CTC). В результате использования данного режима, вообще говоря, на выходе образуется не ШИМ–сигнал, а ЧИМ – частотно-импульсно моделированный сигнал.

Работа в данном режиме описывается следующим алгоритмом:

- Счетный таймер считает от 0 до регистра сравнения (**OCR****);
- После достижения значения OCR** счетный таймер сбрасывается.

На выходе получаются импульсы всегда одинаковой скважности, но разной частоты.

Режим применяется для отсчета таймером периодов (и генерации прерываний) с заданной точностью, т.е. через произвольные промежутки времени.

Практическая часть

1. В тетрадь выписать тему, цель, оборудование и материалы лабораторной работы.
2. В теоретической части – распечатать и прикрепить рисунок 14-1 из технического описания (Datasheet) на МК ATMega 328p (блок-диаграмма 8-ми битного таймера/счетчика);

   – описать, каким образом выбирается тактовая частота таймера/счетчика, какие биты какого регистра за это отвечают. Привести пример установки значений для коэффициента

делителя clkI/O/64 и для остановки таймера/счетчика0 (т.е. No clock source);

– перечислить регистры, участвующие в установке параметров таймера/счетчика0. Привести пример установки конфигурационных бит для режима Fast PWM, **TOP=**0xFF;

– описать конфигурационные биты, определяющие сигнал на выходе **OC**** при занесении в регистр сравнения **OCR****=0xFF в режиме FastPWM (с.107, т.14-6). Привести пример записи конфигурационных бит для этого режима, чтобы при записи **OCR****=0xFF, после интегратора, подключенного к выводу **OC*,** было максимальное значение напряжения (примерно равное напряжению питания).

3. В практической части – создать программу по плавному изменению яркости светодиода (LED, подключенному к выводу OC0 микроконтроллера и, соответственно, к выводу 5 микроконтроллерного блока Arduino UNO). При нажатии 1-й внешней кнопки яркость должна плавно нарастать, при нажатии 2-й кнопки плавно снижаться. При нажатии 3-й кнопки яркость должна постоянно меняться от минимума до максимума, и наоборот;

   – произвести отладку программы (на примере ATMega32) и выполнить симуляцию проекта в VMLAB;

   – осуществить программирование платы Arduino Uno (с использованием инструкции по программированию и загрузчика);

   – выполнить тестирование созданной программы.
4. Описать работу программы и ее особенности.
5. Написать вывод.

Контрольные вопросы

1. Пояснить, почему режим Phase Correct PWM обладает вдвое меньшей максимальной частотой ШИМ, чем режим Fast PWM при той же частоте микроконтроллера?
2. Запишите следующие значения в битовом (двоичном) представлении: 0xFF, 0x7F, 0x01.
3. Почему микроконтроллер не может напрямую генерировать синусоидальный сигнал?
4. Поясните, каким образом можно отмерять произвольные интервалы времени с помощью режима CTC? Чем данные интервалы времени ограничены?

5. Рассчитайте частоту ШИМ для МК ATMega16, используя Timer0 в режиме Fast PWM, частота тактирования МК 8 МГц.
6. Рассчитать значение аналогового напряжения после интегратора при напряжении питания U=5В для каждого из четырех случаев, изображенных на рис. 3.18.

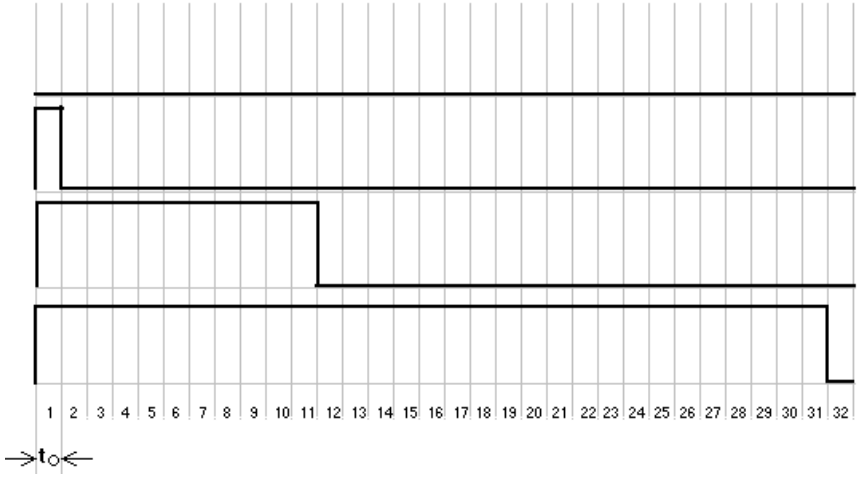


Рис. 3.18. Пример сигнала

7. Используя Datasheet ATMega 16, определите число выходов, которые могут использоваться для аппаратной генерации ШИМ. Перечислите их.
8. Используя Datasheet ATMega 328p, определите число выходов, которые могут использоваться для аппаратной генерации ШИМ. Перечислите их.
9. Опишите, в чем заключается разница между Timer/Counter0 МК ATMega16 и ATMega328p, которая позволяет иметь большее число выходов для аппаратной генерации ШИМ у последнего.
10. Можно ли использовать сигнал внешнего тактового генератора для инкрементирования счетчиков микроконтроллера?

3.8. Лабораторная работа по реализации аналого-цифрового преобразователя на МК серии ATMega и вывода значений на ЖК дисплей (сдвоенная лабораторная работа)

Цель работы: изучение режимов аппаратного аналого-цифрового преобразователя (АЦП) микроконтроллера ATMega. Создание алгоритма инициализации, коммутации каналов, считывания результатов АЦП. Создание соответствующего программного модуля для оцифровки аналогового напряжения.

Необходимый уровень знаний и умений: знание основ микропроцессорной техники, умение ориентироваться в Datasheet на микроконтроллер и понимать представленную там информацию, базовые знания по схемотехнике.

Дальнейшее развитие знаний и умений, полученных в ходе выполнения лабораторной работы: использование аналого-цифрового преобразователя и аппаратного генератора ШИМ позволяет полноценно интегрировать цифровой микроконтроллер в состав реального физического устройства, нередко имеющего на своих входах аналоговый сигнал.

Оборудование и материалы: данное руководство, техническая документация на МК *ATMega328* (Datasheet), схема микроконтроллерного блока Arduino Uno, лабораторный стенд в составе: два микроконтроллерных блока Arduino Uno, блок ЖК 16 символов 2 строки и 6 кнопок управления, цифровая клавиатура 4x4, источник аналогового сигнала плавно меняющегося уровня, подключенный к входу АЦП микроконтроллера, переходники для организации связи по протоколу RS-232 между микроконтроллерными блоками, кабель USB–USB тип B, лабораторный блок питания.

Теоретическая часть

Аналого-цифровой преобразователь (АЦП) – аппаратный модуль, осуществляющий перевод величин (значений) аналогового (т.е. непрерывно изменяющегося сигнала, $y = f(x)$) в цифровую кодировку с известной разрядностью.

Среди множества характеристик АЦП можно выделить следующие:

а) Разрешающая способность (разрешение) – способность АЦП различать два значения входного сигнала, минимально отличающиеся по величине. **Определяется** как величина, обратная максимальному числу кодовых комбинаций на выходе АЦП.

У микроконтроллеров AVR серии ATMega: $2^{10} = 1024$ (Разрешающая способность 1/1024)

б) **Абсолютная точность** – отклонение результатов реального преобразования от идеального. Это составной результат нескольких погрешностей АЦП. Выражается в количестве младших значащих разрядов (**LSB – least significant bit**) АЦП. **Для AVR серии ATMega** абсолютная погрешность АЦП = **±2LSB**.

в) **Предельная частота дискретизации** определяет быстродействие АЦП (измеряется в Гц или SPS – samples per second). Для МК AVR серии ATMega эта величина равна **15 kSPS** (килосемплов в секунду).

Теорема Котельникова (теорема Найквиста-Шеннона) устанавливает связь между максимальной частотой спектра аналогового сигнала, который может быть восстановлен по его цифровым отсчетам и частотой дискретизации.

Теорема: аналоговый сигнал, имеющий ограниченный спектр, может быть восстановлен однозначно и без потерь по своим дискретным отсчётам, **если частота выборки (дискретизации) превышает максимальную частоту спектра сигнала более, чем в 2 раза**.

Пример: если нужно оцифровать аналоговый сигнал с полосой спектра 0 – 7 КГц, то необходимая частота дискретизации должна быть более 14 КГц.

Наиболее часто применяются несколько основных типов аналого-цифрового преобразователя.

Параллельный АЦП. В параллельном АЦП используется массив компараторов, каждый из которых сравнивает входное напряжение с индивидуальным опорным напряжением. Такое опорное напряжение формируется на встроенном прецизионном резистивном делителе. Блок-схема реализации параллельного АЦП приведена на рисунке справа.

Обычно параллельные АЦП имеют ограниченное разрешение, 8 – 10 разрядов.

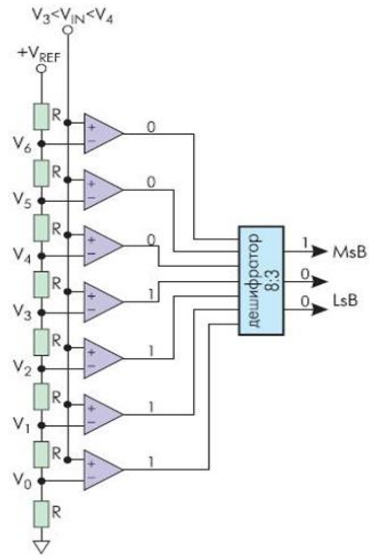


Рис. 3.19. Параллельный АЦП

Большинство высокоскоростных осциллографов и некоторые высокочастотные измерительные приборы используют параллельные АЦП из-за их высокой скорости преобразования, которая может достигать $5*10^9$ отсчетов/сек для стандартных устройств и $20*10^9$ отсчетов/сек для устройств специального применения.

АЦП последовательного приближения. Когда необходимо разрешение 12, 14 или 16 разрядов и не требуется высокая скорость преобразования, а определяющими факторами являются невысокая цена и низкое энергопотребление, обычно применяют АЦП последовательного приближения.

Этот тип АЦП чаще всего используется в разнообразных измерительных приборах и в системах сбора данных. В настоящий момент АЦП последовательного приближения позволяют измерять уровень сигнала с точностью до 16 разрядов и с частотой дискретизации от 100К ($1x10^3$) до 1М ($1x10^6$) отсчетов/сек.

На рис. 3.20 показана упрощенная блок-схема АЦП последовательного приближения. В основе АЦП данного типа лежит специальный регистр последовательного приближения. В начале цикла преобразования все разряды этого регистра устанавливаются равными логическому 0, за исключением первого (старшего) разряда. Это формирует на выходе внутреннего цифро-аналогового преобразователя (ЦАП) сигнал, значение которого равно половине входного диапазона АЦП. А выход компаратора переключается в состояние, определяющее разницу между сигналом на выходе ЦАП и измеряемым входным напряжением.

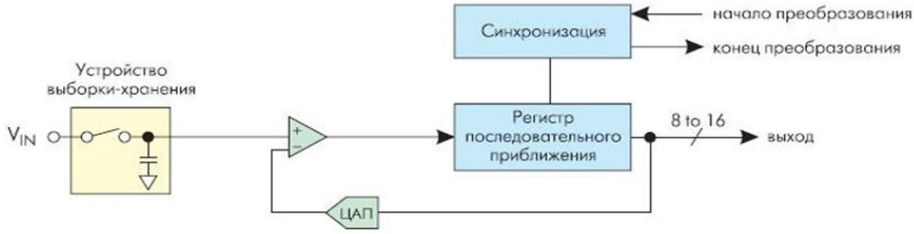


Рис. 3.20. Упрощенная блок-схема АЦП с последовательным приближением

Например, для 8-разрядного АЦП последовательного приближения (рис. 3.20) разряды регистра при этом устанавливаются в 10000000. Если входное напряжение меньше половины входного диапазона АЦП, тогда выход компаратора примет значение

логического 0. Это дает регистру последовательного приближения команду переключить свои разряды в состояние 01000000, что, соответственно, приведет к изменению выходного напряжения, подаваемого с ЦАП на компаратор.

Если при этом выход компаратора по-прежнему оставался бы в 0, то выходы регистра переключились бы в состояние 00100000. Но на этом такте преобразования выходное напряжение ЦАП меньше, чем входное напряжение (рис. 3.21), и компаратор переключается в состояние логической 1. Это предписывает регистру последовательного приближения сохранить 1 во втором разряде и подать 1 на третий разряд. Описанный алгоритм работы затем вновь повторяется до последнего разряда.

Таким образом, АЦП последовательного приближения требуется один внутренний такт преобразования для каждого разряда, или N тактов для N-разрядного преобразования.

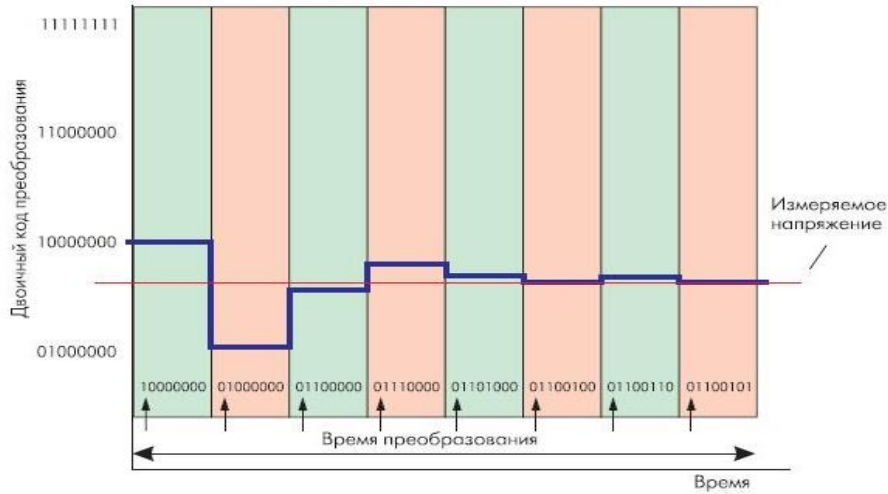


Рис. 3.21. Последовательность работы 8-разрядного АЦП последовательного приближения

Блок-схема аппаратной части АЦП и используемые регистры МК ATMega изображены на рис. 3.22.

Характеристики АЦП МК серии ATMega следующие:

Разрешение – 10 бит. Абсолютная точность **±2LSB.** Время расчета значения 13-260µS. Частота работы АЦП до 15kSPS при максимальном разрешении. При увеличении частоты АЦП разрешение будет уменьшаться.

8 переключаемых каналов (входов). Следует отметить, что расчетный блок один, поэтому одновременно может обрабатываться только один канал.

Опциональное выравнивание по левому краю результата АЦП. Применяется в том случае, если считываются только 8 старших бит результата АЦП. Данная опция позволяет скомпоновать старшие значащие биты (8) в один регистр.

Диапазон входного напряжения АЦП от 0 до «+» напряжения питания.

Возможность инициации прерывания по завершению АЦП преобразования.

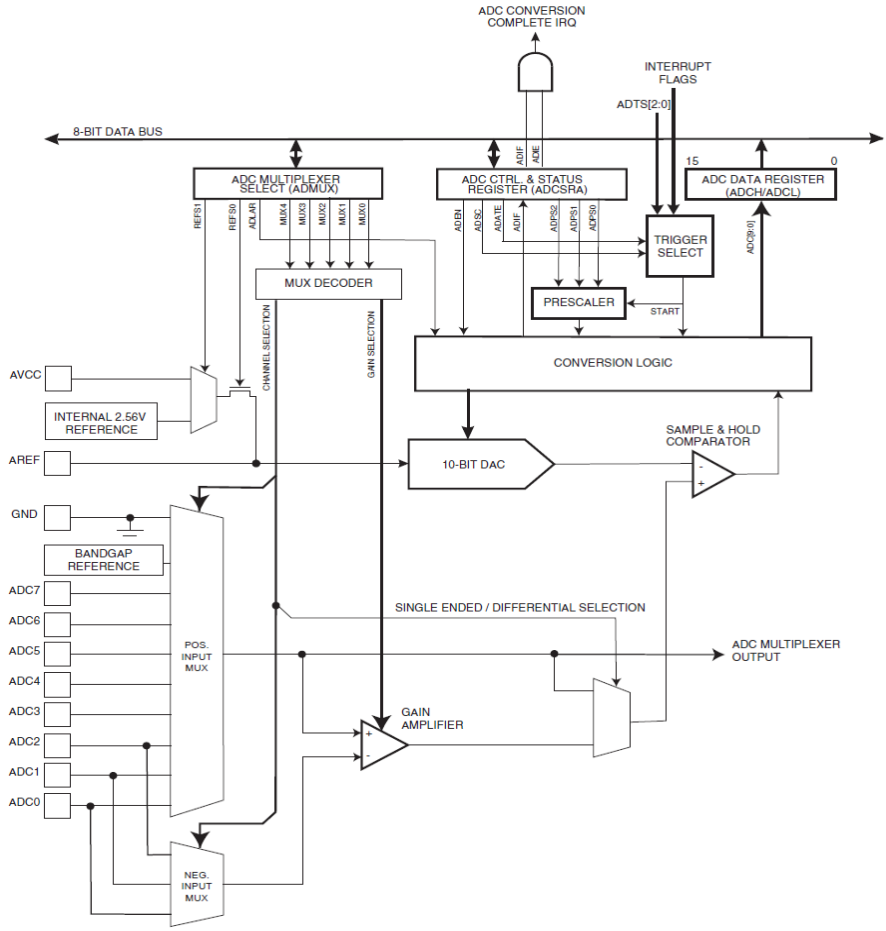


Рис. 3.22. Блок-схема аппаратной части АЦП МК ATMega 16

Как видно из рис. 3.23, в АЦП задействованы следующие регистры:

ADMUX (ADC Multiplexer Selection) – регистр, отвечающий за переключение каналов АЦП. Выбирает канал, который будет оцифровываться в следующий запуск АЦП.

ADCSRA (ADC Control and Status register) – регистр, задающий режимы работы АЦП. Кроме этого, содержит некоторые флаги (например, флаг окончания АЦП–преобразования).

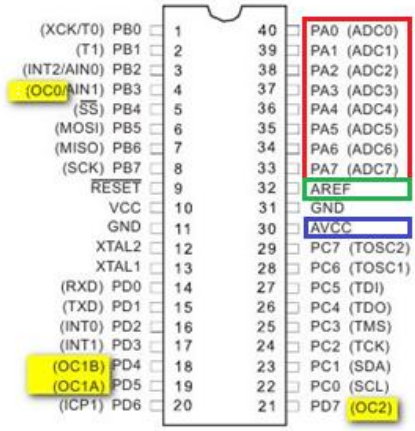


Рис. 3.23. Входы АЦП, ИОН, +Uпит

ADCH/ADCL (ADC Data register) – регистры, в которые записывается результат АЦП–преобразования.

Кроме вышеописанных регистров, блок АЦП насчитывает следующие входы:

- Входы каналов АЦП – ADC0-ADC7;
- Вход аналогового общего провода (GND);
- Вход положительного напряжения питания (+Uпит) аналоговой части микроконтроллера (AVCC);
- Вход опорного напряжения для АЦП (AREF, источник опорного напряжения – ИОН). AVR в качестве ИОНа может использовать напряжение питания, внутренний опорный источник на 2,56 В и напряжение на выводе AREF (внешний ИОН).

Соответствующие входы микроконтроллера показаны на рис. 3.23 справа.

На рис. 3.24 приведена схема съемного модуля LCD Keypad Shield с жидкокристаллическим (ЖК) дисплеем и кнопками, подключаемыми к одному каналу АЦП.

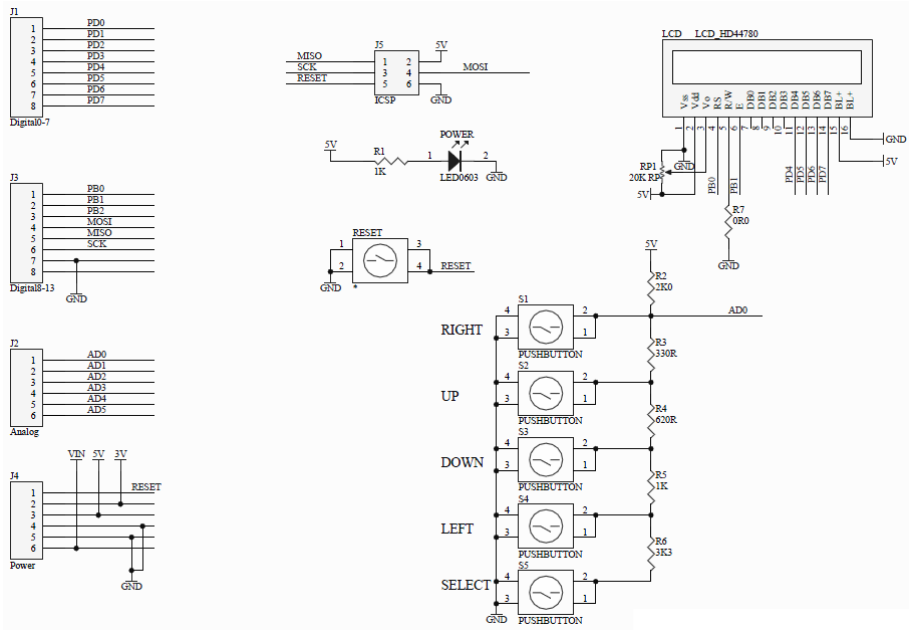


Рис. 3.24. Схема съемного модуля с ЖК дисплеем и кнопками

На рисунке 3.25 изображен общий вид модуля LCD Keypad Shield.

Как видно из рисунка 3.24, кнопки «Right», «Up», «Down», «Left» и «Select» (рисунок 5.6 и рисунок 5.7) подключены к одному входу модуля AD0. Таким образом, достигается экономия используемых ножек модуля и микроконтроллера.

Рис. 3.25. LCD Keypad Shield

Ряд резисторов R2 совместно с R3-R6 образуют делитель напряжения, параметры которого определяются нажатой кнопкой «Right», «Up», «Down», «Left» и «Select».

Так, к примеру, если кнопки не нажаты, на выходе модуля U(AD0)=5.00В. Если нажата кнопка «Select», U(AD0)=3.62В.

Практическая часть

1. В тетрадь выписать тему, цель, оборудование и материалы лабораторной работы. При выполнении работы будет использоваться, кроме микроконтроллерного модуля Arduino Uno Rev.3, съемный модуль с ЖК алфавитно-цифровым дисплеем и 5-ю кнопками, которые можно применить для управления интерфейсом.
   Соответственно, распечатайте и прикрепите в тетрадь рис. 3.24.
2. В практической части:
   – **создать алгоритм первой модификации** (в виде блок-схемы) программы, которая сконфигурирует АЦП на работу с одним каналом (ADC0), формат данных – 8 бит (выравнивание данных по левому краю). Программа должна постоянно считывать значения аналогового напряжения на входе ADC0 контроллера (допускается интервал времени между измерениями 100mS). После получения данных будет реализована проверка, какому интервалу принадлежит значение аналогового напряжения и, соответственно, какая кнопка на LCD Keypad Shield нажата;
   – создать алгоритм второй модификации (в виде блок-схемы), где, кроме функций первой модификации, будет осуществляться переключение каналов, т.е. кнопки остаются подключенными к каналу ADC0, а к каналу АЦП ADC1 подключен источник аналогового сигнала, значения которого оцифровываются с интервалом 100mS.
3. При создании алгоритмов и программ использовать режим «источник опорного напряжения» («AVCC with external capacitor at AREF pin»). Режим работы АЦП – «free running mode». Делитель частоты работы АЦП Division Factor=8. Режим формата данных – 8 бит (выравнивание по левому краю, бит ADLAR=1).
4. **Создать программу**, соответствующую первому алгоритму, используя МК ATMega16. Запустить симуляцию в VMLAB и изучить состояние регистра ADCH с помощью встроенных средств VMLAB при изменении уровня аналогового сигнала на входе ADC0 (используя движковые регуляторы VMLAB).
5. **Описать работу программы и ее особенности**.
6. **Написать вывод**.

Контрольные вопросы

1. Дайте определение разрешающей способности АЦП. Чему она численно равна для АЦП 8 бит, 10 бит, 12 бит, 16 бит?
2. Дайте определение понятию LSB. Чему равен этот параметр для микроконтроллеров AVR?
3. Какие величины связывает теорема Котельникова? Какой должна быть частота дискретизации для сигнала с максимальной частотой в спектре 20 кГц?
4. Опишите преимущества, недостатки и область применения параллельных АЦП.
5. Опишите преимущества, недостатки и область применения АЦП последовательного приближения.
6. Для чего используется выравнивание результата АЦП преобразования в регистрах ADCH:ADCL в МК серии ATMega?
7. Какую функцию выполняет мультиплексор каналов АЦП?
8. Какими системными событиями может сопровождаться окончание АЦП преобразования?
9. Для чего применяется разделение питания цифровой и аналоговой части микроконтроллера?
10. Какую возможность предоставляет использование входа опорного напряжения для АЦП – AREF?

3.8. Лабораторная работа: Прерывания как событийная модель программирования. Разработка программы, реализующей данный принцип

Цель работы: изучение работы прерываний микроконтроллера..

Необходимый уровень знаний и умений: знание основ микропроцессорной техники, умение ориентироваться в Datasheet на микроконтроллер и понимать представленную там информацию, базовые знания по схемотехнике и программированию МК.

Дальнейшее развитие знаний и умений, полученных в ходе выполнения лабораторной работы: использование прерываний позволяет существенно повысить скорость реакции микроконтроллера на внешние события и является повсеместной практикой при создании реальных устройств.

Оборудование и материалы: данное руководство, техническая документация на МК *ATMega328* (Datasheet), схема

микроконтроллерного блока Arduino Uno, лабораторный стенд, лабораторный блок питания.

Теоретическая часть

Прерывание – сигнал, сообщающий о наступлении какого-либо события (Переполнение счетчика таймера, завершение преобразования АЦП и т.д.). При этом выполнение текущей последовательности команд прерывается, и управление передается обработчику прерывания, который в свою очередь корректно реагирует на событие и обрабатывает его, после чего управление передается в прерванный код.

Прерывания позволяют своевременно обрабатывать события периферийных устройств, таких как таймеры, АЦП, **приемопередатчики и так далее.**

В коде примера ниже, основной цикл программы изменяет значения переменной i. Однако, при возникновении сигнала от TIMER 1 о переполнении регистра TCNT1, основной цикл программы прерывается, и управление передается обработчику прерываний. Затем, по окончанию обработки прерывания от таймера, управление возвращается основному циклу программы. То есть контроллер продолжает арифметические действия с переменной i, до возникновения следующего сигнала таймера.

```
#include <avr/io.h>
#include <avr/interrupt.h>
uint8_t num=0;
int i = 0;

ISR(TIMER1_OVF_vect)
{
  PORTD=(1<<num);
  num++;
  If(num>2) {
    num=0;
  }
  TCNT1=61630;                    //Начальное значение таймера
}

int main(void)
{
  DDRD|=(1<<PD0)|(1<<PD1)|(1<<PD2);
```

```
    TCCR1B|=(1<<CS12)|(1<<CS10);      //Предделитель = 1024
    TIMSK|=(1<<TOIE1);//Разрешить прер. по переполнению T1
    TCNT1=61630;                      //Начальное значение таймера
    sei();                            //Разрешить глобальные прерывания

    while(1)
    {
      i++ ;
    }
  }
```

Таблица векторов прерываний имеет вид:

Vector No.	Program Address(2)	Source	Interrupt Definition
1	0x000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset, and Watchdog Reset
2	0x001	INT0	External Interrupt Request 0
3	0x002	INT1	External Interrupt Request 1
4	0x003	TIMER2 COMP	Timer/Counter2 Compare Match
5	0x004	TIMER2 OVF	Timer/Counter2 Overflow
6	0x005	TIMER1 CAPT	Timer/Counter1 Capture Event
7	0x006	TIMER1 COMPA	Timer/Counter1 Compare Match A
8	0x007	TIMER1 COMPB	Timer/Counter1 Compare Match B
9	0x008	TIMER1 OVF	Timer/Counter1 Overflow
10	0x009	TIMER0 OVF	Timer/Counter0 Overflow
11	0x00A	SPI, STC	Serial Transfer Complete
12	0x00B	USART, RXC	USART, Rx Complete
13	0x00C	USART, UDRE	USART Data Register Empty
14	0x00D	USART, TXC	USART, Tx Complete
15	0x00E	ADC	ADC Conversion Complete
16	0x00F	EE_RDY	EEPROM Ready
17	0x010	ANA_COMP	Analog Comparator
18	0x011	TWI	Two-wire Serial Interface
19	0x012	SPM_RDY	Store Program Memory Ready

Прерывания можно разделить на внешние и внутренние.

Внутренние прерывания – обработчики сигналов поступающих от внутренней периферии контроллера (Таймеры, АЦП, UART и т.д.).

Внешние прерывания – обработчики сигналов поступающих от внешних устройств. Именно внешние прерывания позволяют реализовать счетчики импульсов, измерять частоту чего либо, а так же быстро реагировать на события внешних периферийных устройств.

Таким образом, внешние прерывания происходят при изменении логического уровня на определенных ножках контроллера.

Практическая часть

1. Создать программу, отслеживающую нажатие внешней кнопки с помощью системы прерываний;
2. Выполнить отладку, тестирование программы в симуляторе.
3. Записать отчет и вывод по работе.

3.9. Лабораторная работа: Реализация связи с использованием последовательного интерфейса UART

Цель работы: изучение режимов аппаратного интерфейса USART/UART. Создание алгоритма инициализации, заполнения регистра передачи, использование прерываний передатчика USART. В качестве дополнительного задания предусматривается программная реализация алгоритма передачи данных через рассматриваемый интерфейс, а также программная реализация кольцевого буфера.

Необходимый уровень знаний и умений: знание основ микропроцессорной техники, умение ориентироваться в Datasheet на микроконтроллер и понимать представленную там информацию, базовые знания по схемотехнике.

Дальнейшее развитие знаний и умений, полученных в ходе выполнения лабораторной работы: использование интерфейсов связи в цифровых устройствах является одним из основных требований при разработке современных высокоинтегрированных микропроцессорных систем.

Оборудование и материалы: данное руководство, техническая документация на МК *ATMega328* (Datasheet), схема микроконтроллерного блока Arduino Uno, лабораторный стенд в составе: два микроконтроллерных блока Arduino Uno, переходники для организации связи по протоколу RS-232 между микроконтроллерными блоками, кабель USB–USB тип B, лабораторный блок питания.

Теоретическая часть

Практически каждый микроконтроллер имеет универсальный последовательный интерфейс – UART. По структуре это обычный асинхронный последовательный протокол, то есть передающая сторона по очереди выдает в линию 0 и 1, а принимающая отслеживает их и запоминает. Синхронизация идет по времени – приемник и передатчик заранее договариваются о том на какой частоте будет идти обмен.

Важно! Если скорость передатчика и приемника не будут совпадать, то передачи может не быть вообще, либо будут считаны не те данные.

Простой базовый алгоритм работы следующий: Передатчик записывает в линию низкий уровень сигнала – это старт бит. Определив, что линия имеет низкий уровень сигнала, приемник ждет интервал T1 и считывает первый бит, потом через интервалы T2 считываются остальные биты. Последний бит – стоп бит. Он является признаком того, что передача текущего байта завершена.

В конце байта, перед стоп битом, может передаваться бит четности. Его значение (0/1) определяется как результат операции XOR между всеми битами текущего байта и служит для контроля качества передачи. О всех этих параметрах протокола (скорость, наличие и число бит четности, число бит в байте) договариваются до начала передачи. Самым же популярным является 8 бит, один старт один стоп, без контроля четности. График сигналов во времени при передаче байта представлен на рисунке 3.26.

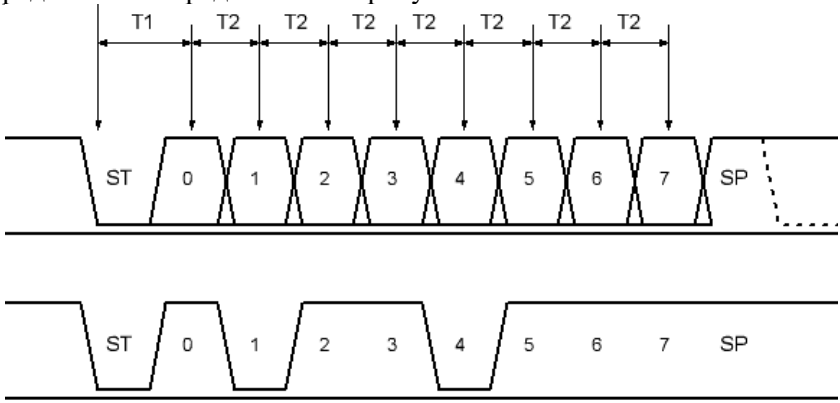


Рис 3.26. Передача байта 11101101

Вывод передатчика обозначается как TX, приемника – RX. Соединение выводов устройств выполняется крест-накрест (см. рисунок 3.27).

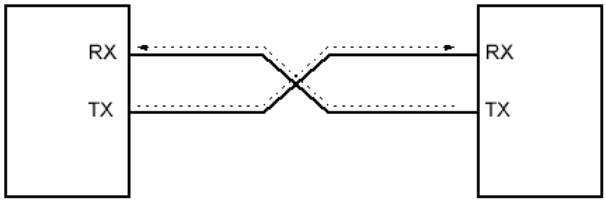


Рис. 3.27. Соединение устройств по UART

Настройка UART. Все настройки приемопередатчика хранятся в регистрах конфигурации. Это **UCSRA**, **UCSRB** и **UCSRC**. А скорость задается в паре **UBBRH:UBBRL**.

Регистр **UCSRA**

Основные биты **RXC** и **TXC**, это флаги завершения приема и передачи, соответственно. Также, одновременно с этими флагами вызывается прерывание (если оно было разрешено). Сбрасываются они аппаратно – принимающий после чтения из регистра **UDR**, передающий при переходе на прерывание, либо программно (чтобы сбросить флаг программно, в него надо записать 1).

Биты **UDRE** сигнализирует о том, что регистр **UDR** приемника пуст и в него можно записывать новый байт. Сбрасывается он аппаратно после записи в UDR какого-либо байта. Также генерируется прерывание «Регистр пуст».

Бит **U2X** – это бит удвоения скорости при работе в ассинхронном режиме. Его надо учитывать при расчете значения в **UBBRH:UBBRL**.

Регистр **UCSRB**

Основные биты **RXEN** и **TXEN** – разрешение приема и передачи. Если их сбросить, выводы UART микроконтроллера станут обычными ножками I/O.

Биты **RXCIE**, **TXCIE**, **UDRIE** разрешают прерывания по завершению приема, передачи и опустошении буфера передачи **UDR**.

Практическая часть

1. Создать программу, фиксирующую нажатие на кнопку, инкрементирующую при этом внутреннюю переменную и предающую эту переменную по интерфейсу UART.

2. Создать программу, принимающую по интерфейсу UART значение переменной и выводящее его на устройство вывода.

3. Создать модели в симуляторе, подключив два микроконтроллера друг к другу посредством интерфейса UART.

4. Выполнить отладку и тестирование программы по передачи данных посредством интерфейса UART.

5. Записать в тетрадь отчет, вывод.

3.10. Лабораторная работа: Управление сдвиговым регистром. Основы интерфейса SPI

Цель работы: изучение основ последовательного интерфейса периферии SPI на примере управления сдвиговым регистром.

Необходимый уровень знаний и умений: знание основ микропроцессорной техники, умение ориентироваться в Datasheet на микроконтроллер и понимать представленную там информацию, базовые знания по схемотехнике.

Дальнейшее развитие знаний и умений, полученных в ходе выполнения лабораторной работы: использование интерфейсов связи в цифровых устройствах является одним из основных требований при разработке современных высокоинтегрированных микропроцессорных систем.

Оборудование и материалы: данное руководство, техническая документация на МК *ATMega328* (Datasheet), схема микроконтроллерного блока Arduino Uno, лабораторный стенд в составе: микроконтроллерный блока Arduino Uno, модуль расширения со сдвиговым регистром, кабель USB–USB тип B, лабораторный блок питания.

Теоретическая часть

Сдвиговый регистр 74HC595 используется в основном для расширения количества выводов микроконтроллера. На рисунке 3.28 приведена схема сдвигового регистра.

Обозначение выводов следующие: **DS** - вход данных, **Q7S** - выход для каскадного подключения регистров, **Q0-Q7** - рабочие выходы, **SHCP** - вход тактовых импульсов, **MR** - вход для сброса регистра, **STCP** - вход помещения данных в регистр хранения, **OE** - вход, переводящий рабочие выходы из высокоомного в рабочее состояние.

Рис. 3.28. Сдвиговый регистр, структурная схема 74HC595

Алгоритм работы со сдвиговым регистром следующий. При поступлении тактового импульса на вход **SHCP** со входа **DS** считывается первый бит и записывается в младший разряд. Со следующим тактовым импульсом бит из младшего разряда сдвигается на один разряд, а на его места записывается бит, поступивший на вход **DS**. Так повторяется все время, а при переполнении сдвигового регистра, ранее поступившие биты последовательно появляются на выходе **Q7S**. Очистка регистра производится подачей низкого уровня на вход **MR**.

Чтобы принятые данные появились на рабочих выходах, их сначала необходимо записать в регистр хранения. Делается это подачей импульса высокого уровня на вход **STCP**. Данные в регистре хранения изменяются лишь при подаче следующего импульса записи.

Для перевода рабочих выходов в высокоомное состояние, на вход **OE** необходимо подать высокий уровень.

Практическая часть

1. В симуляторе собрать схему с использованием микроконтроллера, сдвигового регистра 74HC595, к выводам которого подключить светодиоды и токоограничивающие резисторы. Добавить кнопки для управления микроконтроллером.
2. Создать программу, реализующую «бегущий огонь» посредством управления сдвиговым регистром.
3. Выполнить тестирование. Записать отчет.

Библиографический список

1. Сонькин, М.А. Микропроцессорные системы. Средства разработки программного обеспечения для микроконтроллеров семейства AVR [Электронный ресурс] : учебное пособие / М.А. Сонькин, А.А. Шамин. — Электрон. дан. — Томск : ТПУ, 2016. — 90 с. — Режим доступа: https://e.lanbook.com/book/107725.
2. Новиков, Ю.В. Основы микропроцессорной техники [Электронный ресурс] : учебное пособие / Ю.В. Новиков, П.К. Скоробогатов. — Электрон. дан. — Москва : , 2016. — 406 с. — Режим доступа: https://e.lanbook.com/book/100250
3. Малахов В.П. Микроконтроллеры: уч. пособие для студентов бакалавров по направлению подготовки 6.050102 – Комп. инженерия / В.П. Малахов, Д.П. Яковлев. – О.: Наука и техника, 2008. – 224с.
4. Малахов В.П. Периферийные устройства: уч. пособие для студентов бакалавров по направлению подготовки 6.050102 – Комп. инженерия / В.П. Малахов, Д.П. Яковлев. – О.: Наука и техника, 2006. – 220с.
5. Балашов Е.П., Пузанков Д.В. Микропроцессоры и микропроцессорные системы: Учеб. Пособие для вузов / Под ред. В.Б. Смолова. – М.: Радио и связь, 2001.–328с.
6. Пухальский Г.И. Проектирование микропроцессорных устройств: Учебное пособие для вузов.– СПб.: Политехника, 2001.– 544 с.
7. Лебедев М. Б. CodeVisionAVR. Пособие для начинающих. М.: Додэка, 2008. – 592с.
8. Трамперт В. Измерение, управление и регулирование с помощью AVR-микроконтроллеров. М.: Додэка, 2006.
9. Шпак Ю.А. Программирование на языке Си для AVR и PIC микроконтроллеров. М.: Додэка, 2006.

10. Трамперт В. AVR-RISC микроконтроллеры. М.: Додэка, 2005. – 224 с.
11. Калабеков Б.А. Цифровые устройства и микропроцессорные системы / Б. А. Калабеков. Горячая Линия – Телеком, 2007. – 336 с.
12. Евстифеев А.В. Микроконтроллеры AVR семейства Tiny. Руководство пользователя / А.В. Евстифеев А.В. М.: Додэка-XXI, 2007. – 432 с.
13. Евстифеев А.В. Микроконтроллеры AVR семейства Mega. Руководство пользователя / А.В. Евстифеев А.В. М.: Додэка-XXI, 2007. – 592 с.
14. Пузанков Д.В. Микропроцессорные системы / Д.В. Пузанков. М.: ВУЗ, 2002. – 932 с.
15. Белов А.В. Самоучитель разработчика устройств на микроконтроллерах AVR / А.В.Белов. Наука и техника, 2005. – 530 с.
16. Иванов Ю.И., Югай В.Л. Микропроцессорные устройства систем управления. // Таганрог: Изд-во ТРТУ, 2005. – 342 с.
17. Агуров, П.В. Практика программирования USB / П.В. Агуров. – СПб. : БХВ-Петербург, 2006. – 624 с.
18. Коломбет, Е.А. Микроэлектронные средства обработки аналоговых сигналов / Е.А. Коломбет. – М. : Радио и связь, 1991. – 376 с.
19. Кулаков, В. Программирование на аппаратном уровне : специальный справочник / В. Кулаков. – Изд. 2-е. – СПб. : Питер, 2003. – 847 с.
20. Лысенко, А.А. Преобразователи интерфейса USB на микросхемах FT8U232AM, FT8U245AM / А.А. Лысенко, Р.Н. Назмутдинов // Радио. – 2002. – № 6. – С. 36–39.
21. Меркулов, А.В. Микропроцессорная система управления на базе интерфейсов персонального компьютера : учеб. пособие / А.В. Меркулов. – Хабаровск : Изд-во ДВГУПС, 2004. – 70 с.
22. Мортон, Дж. Микроконтроллеры AVR. Вводный курс / Дж. Мортон ; пер. с англ. – М. : Издат. дом «Додэка-XXI», 2006. – 272 с

23. Смит, Дж. Сопряжение компьютеров с внешними устройствами. Уроки реализации / Дж. Смит ; пер. с англ. – М.: Мир, 2000. – 266 с.
24. Хоровиц, П. Искусство схемотехники. В 2 т. Т. 1 / П. Хоровиц, У. Хилл. – Изд. 3-е, стереотип. – М. : Мир, 1986. – 598 с.
25. Коффрон, Дж. Технические средства микропроцессорных систем: Практический курс / Дж. Коффрон; Пер. с англ. – М.: Мир, 1983. – 344 с.
26. Микропроцессоры. В 3 кн. Кн.1. Архитектура и проектирование микроЭВМ. Организация вычислительных процессов: Учеб. для вузов / П.В. Нестеров, В.Ф. Шаньгин, В.Л. Горбунов и др.; Под ред. Л.Н. Преснухина. – М.: Высш. шк., 1986. – 495 с.
27. Кеннет, Дж. Данхоф. Основы микропроцессорных вычислительных систем / Дж. Данхоф Кеннет, Л. Смит Кэрол; Пер. с англ. А.А. Савельева, Ю.В. Сальникова. – М.: Высш. шк., 1986. – 288 с.
28. Щелкунов Н. Н. Микропроцессорные средства и системы / Н. Н. Щелкунов, А. П. Дианов М.: Радио и связь, 1989. – 288 с.
29. www.easyelectronics.ru
30. https://ru.wikipedia.org/wiki/Обсуждение:Автоматное_программирование

Приложение А.

Лабораторная работа: **Измерение электрических величин с помощью стандартного цифрового мультиметра.**

Цель: освоить использование измерителей электрических величин (вольтметр, амперметр, мультиметр, осциллограф)

Оборудование и материалы: мультиметр, блок питания, набор материалов.

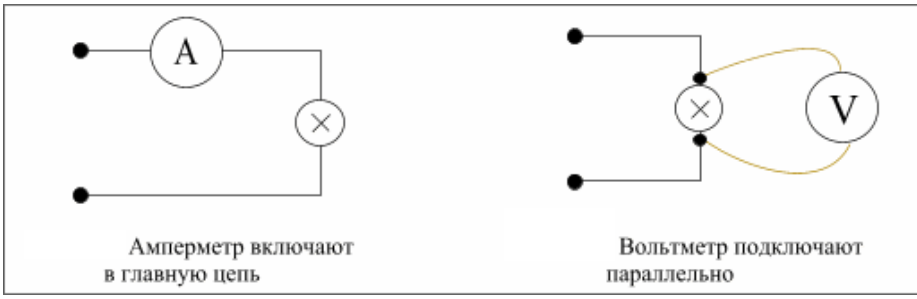


Рисунок 1 – Подключение мультиметра в режимах измерения напряжения и тока

Цифровой мультиметр

Наружная область переключателя на лицевой панели разбита на сектора с указанием выполняемых функций:

DCV и ACV – контроль напряжений соответственно постоянного и переменного тока;

DCA – измерение величины постоянного тока до 200мА;

10DCA – измерение величины постоянного тока до 10 ампер;

hFE – проверка транзисторов;

Ω – контроль сопротивления;

1,5v – 9v – измерение напряжения элементов питания;

COM – минусовое гнездо, общее для всех замеров;

VΩmA – плюсовое гнездо, общее для всех замеров за исключением 10DCA;

10A – гнездо для замера больших токов;

В каждом секторе можно выбирать требуемый предел измерения, устанавливая переключатель в нужное положение.

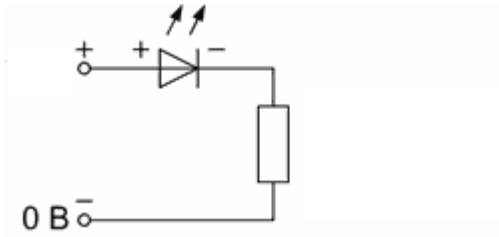
Общая последовательность действий для проведения измерений выглядит следующим образом:

- Вставить соединительные провода щупов в соответствующие гнезда;
- Переключатель выбора режима работы установить в положение, соответствующее виду измерения и роду тока, т.е. в нужный сектор;
- В секторе выбрать предел измерения, близкий к ожидаемому значению измеряемой величины. Если оно неизвестно – устанавливаем переключатель в положение максимального предела измерения;
- Прибор подключается к измеряемому объекту с помощью соединительных проводов щупов;
- Считываются показания прибора, учитывается установленный предел измерения.

Практические задания:

1. Рассчитать номинал добавочного резистора для напряжения питания 5,0 и 12,0 В. Параметры светодиода №1: ток 20 mA,

падение напряжения 2,1 В Параметры светодиода №2: ток 50 mA, напряжение падения 2,12 В



2. Измерить напряжения гальванических элементов, предоставленных преподавателем, записать результаты в таблицу. Отметить те элементы, которые могут считаться исправными.
3. Измерить реальные сопротивления резисторов с номиналом 68 Ом и 47 Ом. Результат записать.
4. Рассчитать и практически измерить электрический ток, протекающий через резистор сопротивлением 68 Ом, 47 Ом при напряжениях 4,8 В и 11,0 В. Записать результаты расчетов и измерений.
5. Оформить работу, записать вывод.

Для заметок

Для заметок

Для заметок

Методические указания к лабораторным работам по дисциплине «Микропроцессорные системы»

для студентов 3 курса дневной формы обучения направления подготовки 09.03.01 «Информатика и вычислительная техника» образовательно-квалификационного уровня «бакалавр»

Автор-составитель Сосновский Ю.В.

_____

_____