МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Кузбасский государственный технический университет имени Т.Ф. Горбачева»

Кафедра электропривода и автоматизации

Составители
А. В. Григорьев
В. Н. Немов

МИКРОПРОЦЕССОРНАЯ ТЕХНИКА

Методические указания к лабораторным работам

Рекомендованы учебно-методической комиссией направления
подготовки 13.03.02 Электроэнергетика и электротехника
в качестве электронного издания
для использования в образовательном процессе

Кемерово 2018

Рецензенты

Негадаев В. А. – доцент кафедры электропривода и автоматизации

Семыкина И. Ю. – председатель учебно-методической комиссии направления подготовки 13.03.02 Электроэнергетика и электротехника

Григорьев Александр Васильевич
Немов Владислав Николаевич
Микропроцессорная техника: методические указания к лабораторным работам [Электронный ресурс]: для обучающихся направления подготовки 13.03.02 Электроэнергетика и электротехника всех форм обучения / сост.: А. В. Григорьев, В. Н. Немов; КузГТУ. – Кемерово, 2018. – Систем. требования: Pentium IV ; ОЗУ 8 Мб ; Windows XP ; мышь. – Загл. с экрана.

Приведено содержание лабораторных работ.

Оглавление

ЛАБОРАТОРНАЯ РАБОТА №1.

Знакомство с аппаратной платформой Freeduino и средой разработки Atmel Studio. Написание простейших программ для микроконтроллера ATmega328.

Цель работы: получить первичные навыки работы в среде разработке программного обеспечения для микроконтроллеров, написать и запустить на отладочной плате простейшую программу.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

В настоящее время без микроконтроллеров не обходится практически ни одно современное электронное устройство. Микроконтроллеры широко применяются в устройствах автоматики, управления и защиты. Современной бытовой и промышленной электронной техники. В связи с этим, знание принципов работы и внутреннего устройства микроконтроллеров, умение конструировать различные изделия на их базе и писать для них управляющие программы становится важным навыком для современного инженера.

Для ознакомления с основами программирования микроконтроллеров, их структурой и особенностями применения, выбраны 8-ми битные микроконтроллер ATmega328, производства компании Atmel.

Основные характеристики ATmega328:

- Ядро: Atmel AVR;
- Разрядность ядра: 8 bit.
- Тактовая частота: 0–20 МГц;
- Объём Flash-памяти: 32 кб;
- Объём SRAM-памяти: 2 кб;
- Объём EEPROM-памяти: 1 кб;
- 32 8-разрядных рабочих регистра общего назначения;
- Напряжение питания: 1,8–5,5 В;
- Потребляемый ток в режиме работы: 0,2 мА (1 МГц, 1,8 В);
- Потребляемый ток в режиме сна: 0,75 мкА (1 МГц, 1,8 В);
- Количество таймеров/счётчиков: 2 восьмибитных, 1 шестнадцатибитный;
- Общее количество портов: 23;
- Количество ШИМ (PWM) выходов: 6;
- Количество каналов АЦП (аналоговые входы): 6;
- Количество аппаратных USART (Serial): 1;
- Количество аппаратных SPI: 1 Master/Slave;
- Количество аппаратных I²C/SPI: 1;
- Разрешение АЦП: 10 бит;
- Рабочий температурный диапазон, °C: –40...+85.

Для программирования данных микроконтроллеров применяют следующие среды разработки: Atmel Studio, IAR Systems, WinAVR, CodeVisionAVR и т.д. Они позволяют писать программы как на языке ассемблера, так и на Си.

Для выполнения лабораторных работ используется среда разработки «Atmel Studio», версии 6.2.

ОТЛАДОЧНАЯ ПЛАТА «Freeduino 2009»

Лабораторные работы выполняются на отладочной плате «Freeduino 2009», которая являются аналогом отладочный платы «Arduino Duemilanove». При необходимости она легко может быть дополнена платами расширения и иными внешними элементами.

Внешний вид платы приведен на рисунке 1.1.

Технические характеристики отладочной платы «Freeduino 2009»:

- Микроконтроллер: ATmega328;
- ППЗУ (Flash Memory): 32 Кбайт, из них 2 Кбайта используются загрузчиком (bootloader) «Arduino»;
- ОЗУ (SRAM): 2 Кбайт;
- ПЗУ (EEPROM): 1024 байт;
- Тактовая частота: 16 МГц;
- рабочие напряжение микроконтроллера: 5 вольт;
- Интерфейс с ПК: USB;
- Питание от USB либо от внешнего источника, выбор с помощью перемычки.

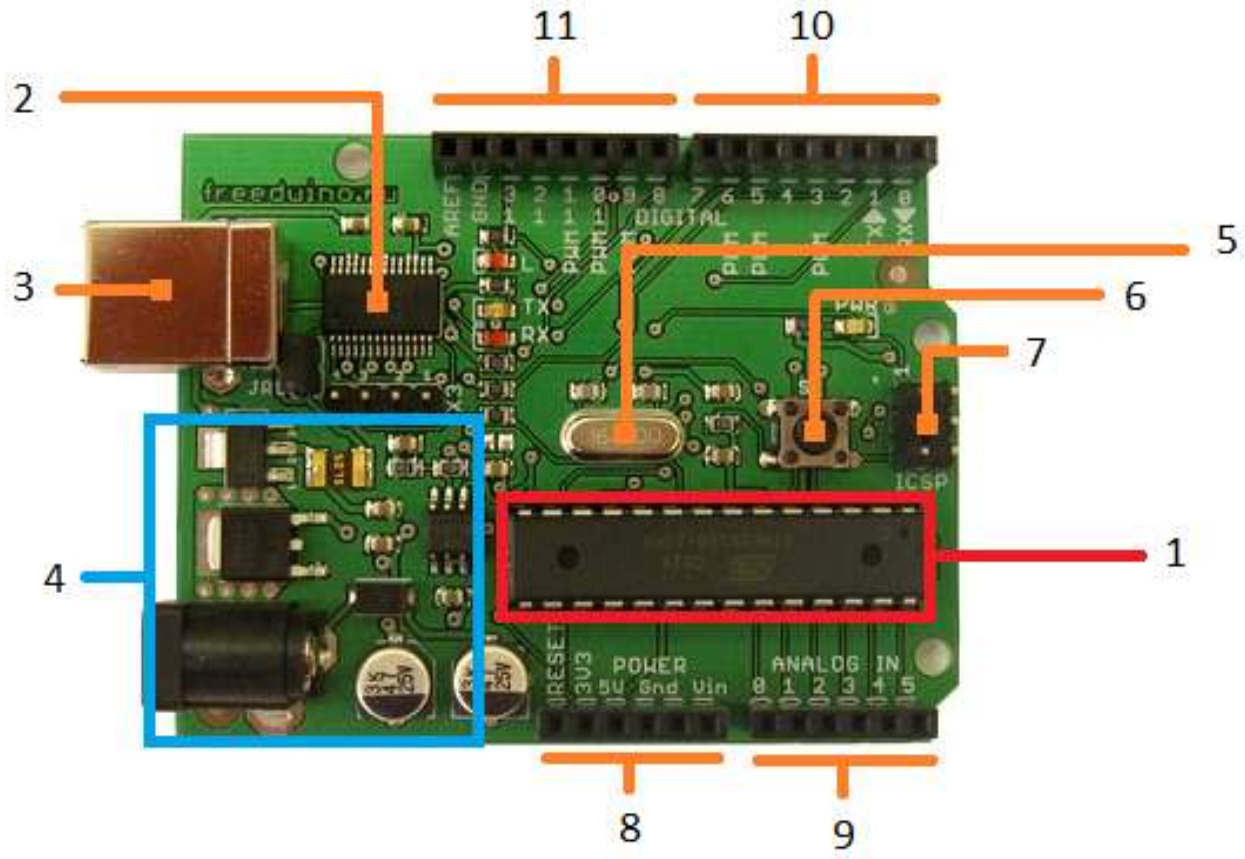

Рис. 1.1. Отладочная плата «Freeduino 2009»

На плате расположены (рис 1.1):

1) Микроконтроллер, ATmega328.

2) Микросхема FT232RL. Преобразовать Serial в USB.

3) Разъем USB-B, для подключения платы к компьютеру.

4) Разъем для подключения внешнего питания, микросхема линейного стабилизатора на 5 вольт и схема автоматического выбора источника питания – от USB или внешнего источника. В первых версиях платы выбор источника питания выполняется вручную, с помощью перестановкой джампера.

5) Кварцевый резонатор, 16 МГц.

6) Кнопка RESET.

7) Разъем подключения ISP программатора.

8) Выводы напряжения питания и общего провода.

9) Аналоговые порты ввода – PB0 – PB5.

10) Цифровые порты ввода/вывода – PD0 – PD7 (из них 3 с ШИМ-сигналом).

11) Цифровые порты ввода/вывода – PB0 – PB5 (из них 3 с ШИМ-сигналом).

Также на плате имеются четыре светодиода:

- TX и RX – отображают прием/передачу по последовательному (Serial) протоколу связи с ПК или другим устройством.

- PWR – индикатор наличия питания.

- L (или 13) – пользовательский светодиод. Соединен с 13 цифровым портом (порт на МК – PB5, 19 вывод). Также служит для индикации работы загрузчика. Мерцает первые 1-2 секунды после перезагрузки МК.

При использовании отладочной платы необходимо помнить, что номера выходов платы не соответствуют номерам ножек и портов микроконтроллера. Схема соответствия приведена на рисунке 1.2.

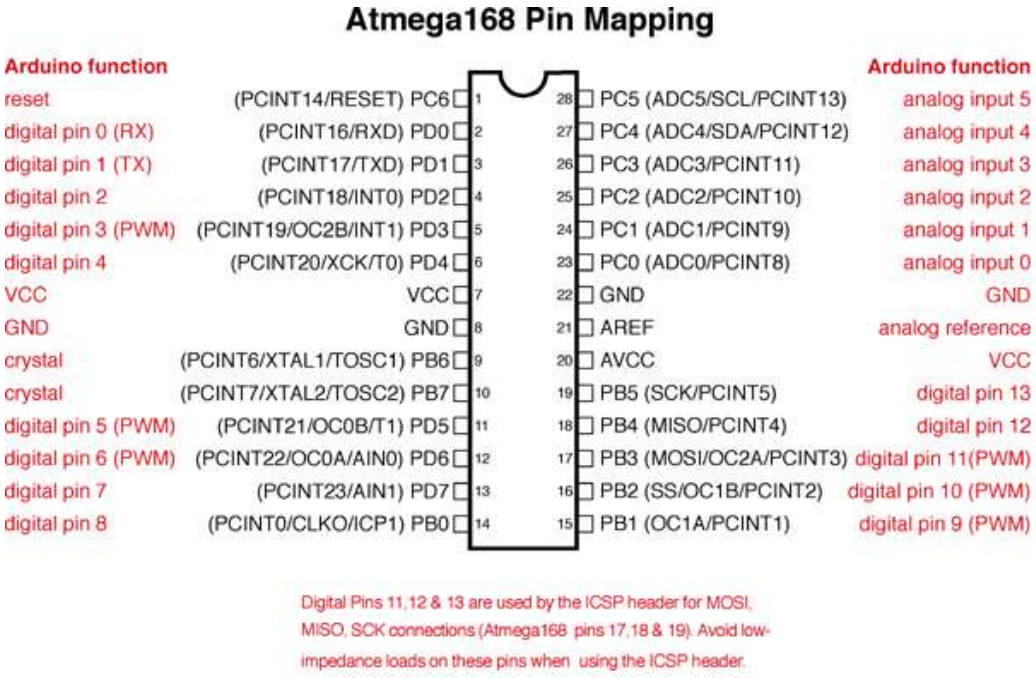

Рис. 1.2. Схема соответствия выводов микроконтроллера и платы

УПРАВЛЕНИЕ ЦИФРОВЫМИ ВЫВОДАМИ МК

Вывод микроконтроллера – это довольно сложный механизм (регистры + триггеры + мультиплексоры + фильтры + тактирование и т.д.). При этом за одним выводом может быть закреплено по несколько функций, число которых зависит от того, сколько периферийных устройств подключено к данному выводу. Но, несмотря на относительно сложную конструкцию, управление выводом в микроконтроллерах с ядром AVR возможно всего лишь через два регистра, плюс еще один регистр для контроля состояния вывода.

Поскольку ядро AVR являются восьмиразрядными, то разработчики для удобства пользователей объединили выводы в группы по восемь. Такая группа называется порт (Port). Удобство заключается в том, что для управления восьмиразрядными портами используются такие же восьмиразрядные регистры/переменные. Это позволяет сократить размер программы и достичь высокого быстродействия.

Название вывода формируется из инициала слова Port («P»), после чего идет имя порта (к примеру «B») и порядковый номер вывода в этом порте, от «0» до «7» (к примеру «5», тогда полное имя будет «PB5»). В документации на микроконтроллер (datasheet) можно встретить такое обозначение – «Pxn», где «x» это имя порта, а «n» это номер вывода.

Названия всех перечисленных ниже регистров представлены в виде ИМЯx, где «x» – буква порта, B, D или C.

DDRx – Data Direction Register of port x – регистр, управляющий режимом работы портов. Если в его ячейку (например, 5-ую), записать «1», то связанный с ней порт МК (Px5) станет работать как выход, «0» – как вход. Гарантированно, что после перезапуска микроконтроллера или подачи напряжения питания, значение любого DDRx регистра равно 0x00, это означает, что по умолчанию весь порт «x» настроен на вход.

PORTx – устанавливает уровень на портах МК. «1» – высокий, «0» – низкий. Если же вывод МК настроен на выход, то «1» и «0» в соответствующий ему ячейки этого регистра подключает, либо отключает, внутренний, «подтягивающий» вывод МК к «+» питания, резистор. Гарантированно, что после перезапуска микроконтроллера или подачи напряжения питания, значение любого PORTx регистра равно 0x00. При этом, DDRx тоже 0x00, а значит по умолчанию весь порт будет находится в состоянии входа, без подключения подтягивающего резистора.

PINx – принимает в себя логические уровни напряжения (высокое или низкое) на соответствующих выводах МК и хранит их. Содержимое этого регистра возможно только «прочитать».

На рис. 1.3 показана упрощенная блок-схема одного вывода.

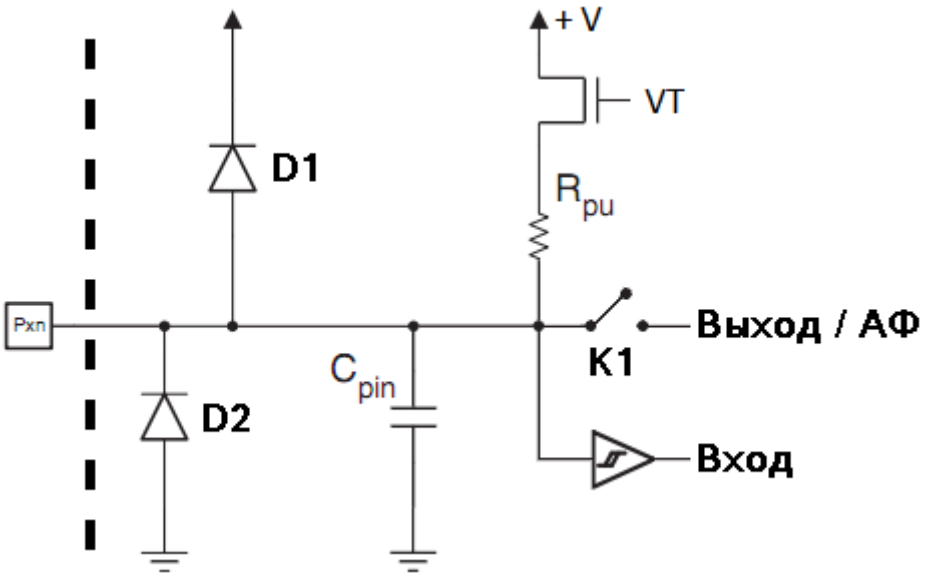

Рис.1.3 Блок-схема вывода (ножки) AVR микроконтроллера

Приведенные на рис.2 диоды D1 и D2, предназначены для защиты вывода от напряжений, превышающих напряжение питания (электростатическое (ESD) и т.п.), а емкость Cpin это паразитная емкость вывода. Rpu – внутренний «подтягивающий» резистор. Он подключается к «+» питания МК с помощью транзистора VT. Контакт К1 подключает/отключает режим работы ножки микроконтроллера на выход или альтернативную функцию. Состояние контакта К1 определяется регистром DDRx.

ПРИМЕР ПРОСТЕЙШЕЙ ПРОГРАММЫ ДЛЯ МИКРОКОНТРОЛЛЕРА ATmega328

Ниже приведен пример простейшей программы для микроконтроллера.

Данная программа реализует включение/выключение светодиода, подключенного к порту PB5. Этот светодиод уже есть на отладочной плате, он обозначен буквой «L» или числом «13».

```
#include <avr/io.h>
#define F_CPU 16000000UL //16MHz
#include <util/delay.h>

int main(void)
{
	DDRB  |= 1<<5;
	PORTB &= ~1<<5;
	while(1)
{
		PORTB |= 1<<5;
		_delay_ms(1000);
		PORTB &= ~1<<5;
		_delay_ms(1000);
	}
}
```

Разберем её построчно:

- #include <avr/io.h> – подключение стандартной библиотеки для ядра AVR. В ней находятся определения констант, имен регистров и всего прочего, что может понадобиться для выполнения базового ввода-вывода.
- #define F_CPU 16000000UL – Замена в тексте слова F_CPU на 16000000UL. Это тактовая частота контроллера в герцах. Данная строчка нужна для корректной библиотеки delay.

- #include <util/delay.h> – подключение библиотеки для реализации задержки выполнения программы только одной «командой».
- Строки после «int main(void){» и до «while(1)» – это строки первичной инициализации периферии. В них выполняется настройка режимов работы портов и устанавливаются их начальные состояния.
- DDRB |= 1<<5 – установка «1», в 5-й бит регистра DDRB. Это установит порт PB5 в состояние «выход». Оператор «<<» это побитовый сдвиг влево, а запись «1<<5» означает, что в двоичном числе единицу необходимо сдвинуть пять раз, то есть начинается с 00000001 (позиция «0»), а в результате сдвига будет 0010000 (позиция «5»). Оператор «|=» производит логическое сложение содержимого регистра DDRB с числом «1<<5». При логическом побитовом сложении единицы с любым числом, результатом будет «1». При сложении любого числа с нулем, получится исходное число. Все это означает, что после выполнения команды «DDRB |= 1<<5;» в пятом бите регистра DDRB окажется «1», а остальные биты не изменят свое состояние.
- PORTB &= ~1<<5 – Задание начального состояние уровня на выводы PB5. Запись «0», в пятый бит регистра PORTB «0», чтобы на выводе был низкий уровень – напряжение 0 Вольт. Оператор «~» означает инверсию, а запись «~1<<5» означает двоичное число 0b11011111, то есть число, обратное числу 0b00100000. Это число также перемножается, знак «&», с содержимым регистра PORTB и результат записывается в регистр PORTB. Битовое умножение на «0» любого числа дает в результате «0». При умножении любого числа на «1» результатом будет исходное число. В итоге после выполнения команды «PORTB &= ~1<<5;», в пятом бите регистра PORTB будет ноль, а остальные не изменят свое состояние. Данная строчка не является обязательной, но желательна для надежного «старта», с жестко задачным начальным состоянием на используемом выводе МК.
- «while(1) { *ОСНОВНАЯ ПРОГРАММА* }». Основная программа выполняется в теле цикла while пока условие входа в цикл истинно, то есть равно «1». Так как это значение задано изначально и не является переменной, то условие будет выполняться всегда и микроконтроллер, пока подано питание, будет бесконечно, циклично, выполнять содержимое тела цикла.
- PORTB |= 1<<5 – установка «1» в 5-тый бит регистра состояния порта B, а следовательно высокого, 5 вольт, уровня на ножке PB5 (9 ножка микроконтроллера, 13 порт на отладочной плате). Включение пользовательского светодиода «L» («13»).
- _delay_ms(1000) – задержка в 1000 миллисекунд, что равно одно секунде.

- PORTB &= ~1<<5 – установка «0» в 5-тый бит регистра состояния порта B, а следовательно, низкого, 0 вольт, уровня на ножке PB5. Выключение пользовательского светодиода «L» («13»).
- _delay_ms(1000) – задержка в 1 секунду.

Стоить отметить, что запись битов в регистры может быть выполнена и в другом виде.

Например:

DDRB = 0x20; – в шестнадцатеричном.

DDRB = 0b00100000; – в двоичном.

При этом в обоих случаях будут изменены и все другие биты регистра. Это удобно, когда нужно настроит на выход/вход или установить высокий/низкий уровень сразу на нескольких портах, но излишне, если необходимо изменить состояние только одного порта.

ПЛАТА РАСШИРЕНИЯ «ТРЕХЦВЕТНЫЙ СВЕТОДИОД»

Для выполнения данной лабораторной работы дополнительно потребуется плата расширения «трехцветный светодиод».

На это плате установлен RGB (Red, Green. Blue) светодиод. Конструктивно он состоит из трех отдельных светодиодов, собранных в одном корпусе. Это позволяет получать или три цвета поочередно, или любой другой цвет, путём включения сразу двух светодиодов и регулировки их яркости. Включение всех трёх позволит получить псевдо-белый цвет.

Схема платы расширения приведена на рисунке 1.4.

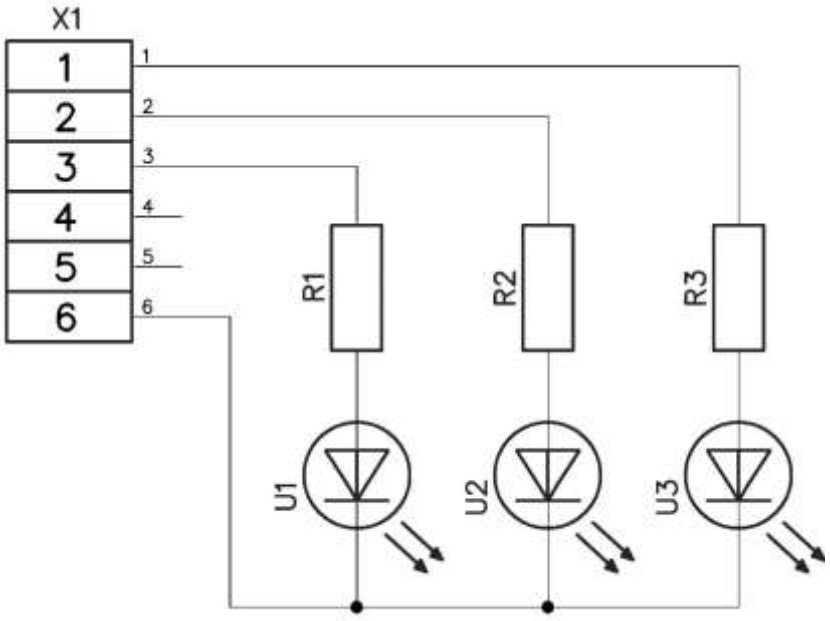

Рис. 1.4. Схема платы расширения «трехцветный светодиод»

На схеме обозначены:

- R1-R3 – резисторы. Они ограничивают ток через светодиод, а также порт микроконтроллера. Ток не превышает 15 mA.
- U1-U3 – светодиоды. Зелёный, красный и синий.

Плату расширения необходимо установить так, как показана на рисунке 1.5. Порты отладочной платы «8» и «AREF» не должны быть заняты.

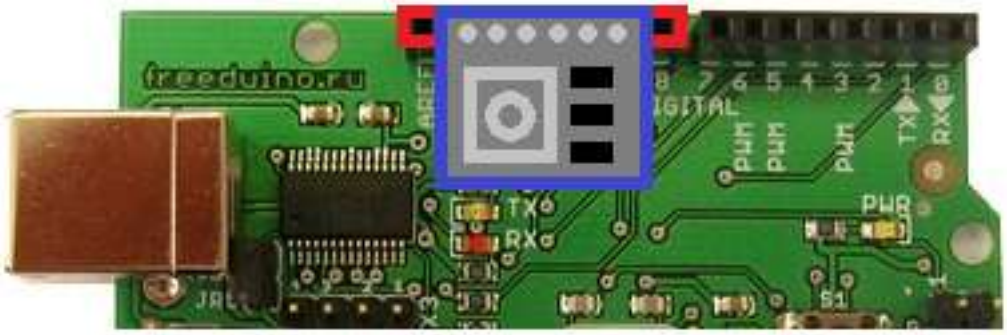

Рис. 1.5. Расположение платы расширения на отладочной плате

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Запустите Atmel Studio и выполните File -> New -> Project.

Появится окно создания нового проекта. В нижней его части Вы можете задать название проекта. Имя по умолчанию будет иметь вид «GccApplication1», при этом число в окончание может быть различным. Замените его на имя в формате «Группа, индивидуальный номер, случайное число до 100» (например, «EAb-083_06_53»). Свой индивидуальный номер можно узнать у преподавателя.

В появившемся окне выберите «GCC C Executable Project». После этого отобразится окно выбора используемого в проекте микроконтроллера. Найдите в списке «ATmega328P» Кликните мышкой по этой строке и нажмите ОК. Рекомендуется воспользоваться строкой поиска, в правом верхнем углу окна выбора микроконтроллера.

На странице с текстом программы сотрите все и наберите программу из разобранного выше примера.

Выполните Build -> Build Solutions, либо нажмите F7.

Дождитесь окончания компиляции. В нижнем окне «Output» должны появится строчки:

Build succeeded.

========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========

Для того чтобы непосредственно поместить содержимое этого файла в память микроконтроллера, необходимо воспользоваться дополнительной программой – XLoader.

Подключите отладочную плату к компьютеру, используя кабель типа USB-A – USB-B, и запустите программу XLoader.

Окно программы содержит следующие элементы:

- Hex file – здесь нужно указать путь до загружаемого HEX файла;
- Device – здесь необходимо выбрать тип платы и микроконтроллера. Duemilanove/Nano(тип МК – ATmeg328);
- COM port – здесь необходимо выбрать виртуальный комп порт, через который компьютер общается с МК;
- Baud rate – скорость передачи данных через COM-порт. Зависит от типа МК и выставляется автоматически. Для ATmega328 – 57600.

Полный путь до папки будет иметь вид:

Документы\Atmel Studio\6.2\Имя_проекта\ Имя_проекта\Debug

В папке Debug необходимо найти файл с расширением «.hex». Именно этот файл «заливается» в микроконтроллер. В нём содержится программа для микроконтроллера в виде машинных кодов.

Если все установки правильные, то после нажатия кнопки «Upload» программа свяжется с загрузчиком в МК и, получив от него ответ, загрузит содержимое HEX файла в память МК. Процесс обмена данными можно наблюдать визуально, по мерцанию расположенных на отладочной плате светодиодов RX и TX.

ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ

Подключите плату расширения «трехцветный светодиод».

Определите, какие порты микроконтроллера необходимо настроить как выход. Выполните настройку в соответствующем регистре DDRx.

Выполните индивидуальное задание. Номер задания уточняйте у преподавателя.

Напишите программу, реализующую следующий режим работы светодиодов:

1) Поочередное включение всех светодиодов с последующим поочередным отключением. Интервал произвольный, но не менее 500 миллисекунд. В конце по 2-3 «вспышки» разных пар светодиодов.

2) Кратковременное включение, «вспышка», каждого из трех светодиодов сначала по очереди, а потом попарно, в конце включение сразу всех светодиодов на 1 секунду.

3) Каждый светодиод включается/выключается по очереди 2-3 раза, потом по парно. В конце 2 «вспышки» всех светодиодов вместе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение регистров DDRx.
2. Назначение регистров PORTx.
3. Для чего необходимы резисторы на плате «трехцветный светодиод». Предельный ток через порт микроконтроллера.
4. Функция _delay_ms(). Принцип работы и недостатки.
5. Характеристики микроконтроллера ATmega328. Зависимость тактовой частоты от напряжения питания.

ЛАБОРАТОРНАЯ РАБОТА №2.

Написание программы для опроса состояний входов микроконтроллера ATmega328.

Цель работы: получить первичные навыки работы в среде разработке программного обеспечения для микроконтроллеров, написать и запустить на отладочной плате простейшую программу, реализующую различные алгоритмы реакции на изменение состояния кнопки.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

В настоящее время без микроконтроллеров не обходится практически ни одно современное электронное устройство. Микроконтроллеры широко применяются в устройствах автоматики, управления и защиты. Современной бытовой и промышленной электронной техники. В связи с этим, знание принципов работы и внутреннего устройства микроконтроллеров, умение конструировать различные изделия на их базе и писать для них управляющие программы становится важным навыком для современного инженера.

Для ознакомления с основами программирования микроконтроллеров, их структурой и особенностями применения, выбраны 8-битные микроконтроллер ATmega328, производства компании Atmel.

Основные характеристики ATmega328:

- Ядро: Atmel AVR;
- Разрядность ядра: 8 bit.
- Тактовая частота: 0–20 МГц;
- Объём Flash-памяти: 32 кб;
- Объём SRAM-памяти: 2 кб;
- Объём EEPROM-памяти: 1 кб;
- 32 8-разрядных рабочих регистра общего назначения;
- Напряжение питания: 1,8–5,5 В;
- Потребляемый ток в режиме работы: 0,2 мА (1 МГц, 1,8 В);
- Потребляемый ток в режиме сна: 0,75 мкА (1 МГц, 1,8 В);
- Количество таймеров/счётчиков:
  2 восьмибитных и 1 шестнадцатибитный;
- Общее количество портов: 23;
- Количество ШИМ (PWM) выходов: 6;
- Количество каналов АЦП (аналоговые входы): 6;
- Количество аппаратных USART (Serial): 1;
- Количество аппаратных SPI: 1 Master/Slave;
- Количество аппаратных I²C/SPI: 1;
- Разрешение АЦП: 10 бит;

- Рабочий температурный диапазон, °С: –40...+85.

Для программирования данных микроконтроллеров применяют следующие среды разработки: Atmel Studio, IAR Systems, WinAVR, CodeVisionAVR и т.д. Они позволяют писать программы как на языке ассемблера, так и на Си.

Для выполнения лабораторных работ используется среда разработки «Atmel Studio», версии 6.2.

ОТЛАДОЧНАЯ ПЛАТА «Freeduino 2009»

Лабораторные работы выполняются на отладочной плате «Freeduino 2009», которая является аналогом отладочный платы «Arduino Duemilanove». При необходимости она легко может быть дополнена платами расширения и иными внешними элементами.

Внешний вид платы приведен на рисунке 2.1.

Технические характеристики отладочной платы «Freeduino 2009»:

- Микроконтроллер: ATmega328;
- ППЗУ (Flash Memory): 32 Кбайт, из них 2 Кбайта используются загрузчиком (bootloader) «Arduino»;
- ОЗУ (SRAM): 2 Кбайт;
- ПЗУ (EEPROM): 1024 байт;
- Тактовая частота: 16 МГц;
- рабочие напряжение микроконтроллера: 5 вольт;
- Интерфейс с ПК: USB;
- Питание от USB либо от внешнего источника, выбор с помощью перемычки.

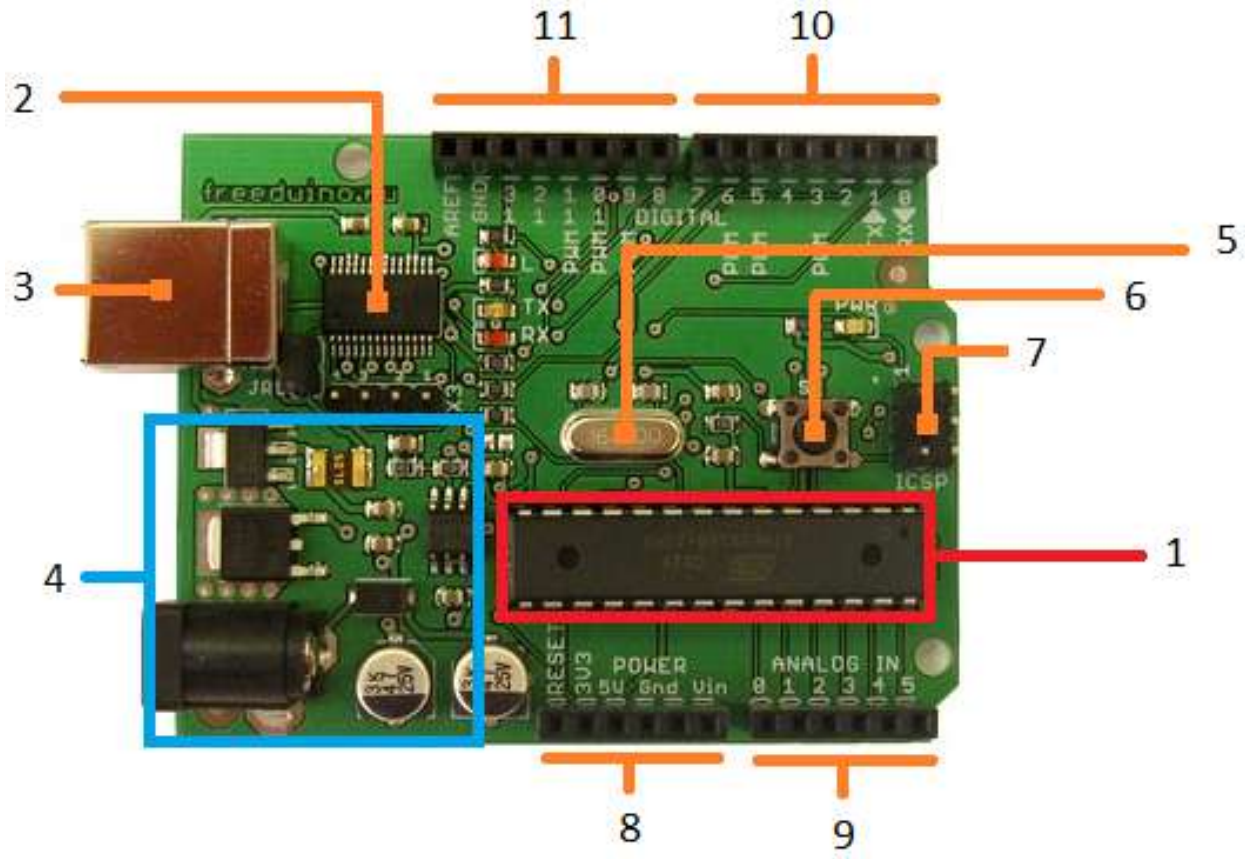

Рис. 2.1. Отладочная плата «Freeduino 2009»

На плате расположены (рис. 2.1):

1) Микроконтроллер, ATmega328.

2) Микросхема FT232RL. Преобразовать Serial в USB.

3) Разъем USB-B, для подключения платы к компьютеру.

4) Разъем для подключения внешнего питания, микросхема линейного стабилизатора на 5 вольт и схема автоматического выбора источника питания – от USB или внешнего источника. В первых версиях платы выбор источника питания выполняется вручную, с помощью перестановкой джампера.

5) Кварцевый резонатор, 16 МГц.

6) Кнопка RESET.

7) Разъем подключения ISP программатора.

8) Выводы напряжения питания и общего провода.

9) Аналоговые порты ввода – PB0 – PB5.

10) Цифровые порты ввода/вывода – PD0 – PD7 (из них 3 с ШИМ-сигналом).

11) Цифровые порты ввода/вывода – PB0 – PB5 (из них 3 с ШИМ-сигналом).

Также на плате имеются четыре светодиода:

- TX и RX – отображают прием/передачу по последовательному (Serial) протоколу связи с ПК или другим устройством.

- PWR – индикатор наличия питания.

- L (или 13) – пользовательский светодиод. Соединен с 13 цифровым портом (порт на МК – PB5, 19 вывод). Также служит для индикации работы загрузчика. Мерцает первые 1-2 секунды после перезагрузки МК.

При использовании отладочной платы необходимо помнить, что номера выходов платы не соответствуют номерам ножек и портов микроконтроллера. Схема соответствия приведена на рисунке 2.2.

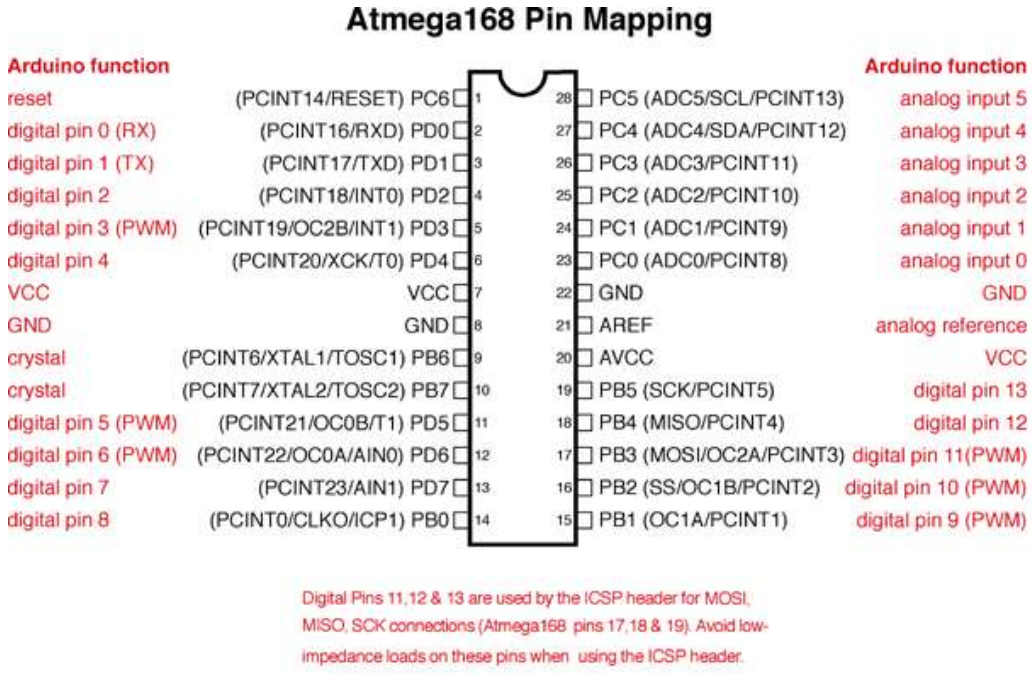

Рис. 2.2. Схема соответствия выводов микроконтроллера и платы

УПРАВЛЕНИЕ ЦИФРОВЫМИ ВВОДАМИ МК

Вывод микроконтроллера – это довольно сложный механизм (регистры + триггеры + мультиплексоры + фильтры + тактирование и т.д.). При этом за одним выводом может быть закреплено по несколько функций, число которых зависит от того, сколько периферийных устройств подключено к данному выводу. Но, несмотря на относительно сложную конструкцию, управление выводом в микроконтроллерах с ядром AVR возможно всего лишь через два регистра, плюс еще один регистр для контроля состояния вывода.

Поскольку ядро AVR являются восьмиразрядными, то разработчики для удобства пользователей объединили выводы в группы по восемь. Такая группа называется порт (Port). Удобство заключается в том, что для управления восьмиразрядными портами используются такие же восьмиразрядные регистры/переменные. Это позволяет сократить размер программы и достичь высокого быстродействия.

Название вывода формируется из инициала слова Port («P»), после чего идет имя порта (к примеру «B») и порядковый номер вывода в этом порте, от «0» до «7» (к примеру «5», тогда полное имя будет «PB5»). В документации на микроконтроллер (datasheet) можно встретить такое обозначение – «Pxn», где «x» – это имя порта, а «n» – это номер вывода.

Названия всех перечисленных ниже регистров представлены в виде ИМЯx, где «x» – буква порта, B, D или C.

DDRx – Data Direction Register of port x – регистр, управляющий режимом работы портов. Если в его ячейку (например, 5-ую), записать «1», то связанный с ней порт МК (Px5) станет работать как выход, «0» – как вход. Гарантированно, что после перезапуска микроконтроллера или подачи напряжения питания, значение любого DDRx регистра равно 0x00, это означает, что по умолчанию весь порт «x» настроен на вход.

PORTx – устанавливает уровень на портах МК. «1» – высокий, «0» – низкий. Если же вывод МК настроен на выход, то «1» и «0» в соответствующий ему ячейки этого регистра подключает, либо отключает, внутренний, «подтягивающий» вывод МК к «+» питания, резистор. Гарантированно, что после перезапуска микроконтроллера или подачи напряжения питания, значение любого PORTx регистра равно 0x00. При этом, DDRx тоже 0x00, а значит по умолчанию весь порт будет находиться в состоянии входа, без подключения подтягивающего резистора.

PINx – принимает в себя логические уровни напряжения (высокое или низкое) на соответствующих выводах МК и хранит их. Содержимое этого регистра возможно только «прочитать».

На рис. 2.3 показана упрощенная блок-схема одного вывода.

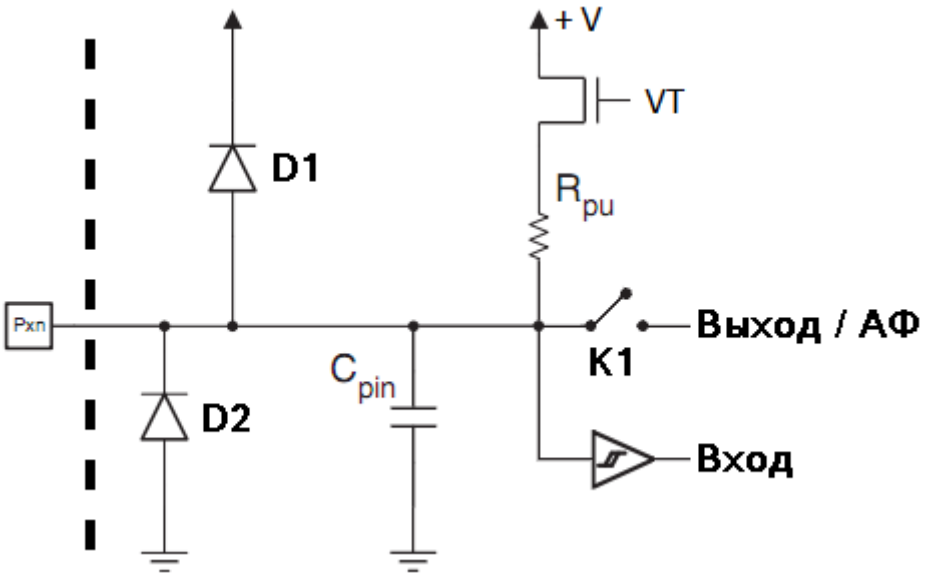

Рис. 2.3. Блок-схема вывода (ножки) AVR микроконтроллера

Приведенные на рис.2 диоды D1 и D2, предназначены для защиты вывода от напряжений, превышающих напряжение питания (электростатическое (ESD) и т.п.), а емкость Cpin – это паразитная емкость вывода. Rpu – внутренний «подтягивающий» резистор. Он подключается к «+» питания МК с помощью транзистора VT. Контакт К1 подключает/отключает режим работы ножки микроконтроллера на выход или альтернативную функцию. Состояние контакта К1 определяется регистром DDRx.

ПРИМЕР ПРОГРАММЫ ДЛЯ МИКРОКОНТРОЛЛЕРА

Ниже приведен пример программы для микроконтроллера, которая реализует опрос одной кнопки и визуальное отображение её состояния с помощью имеющегося на отладочной плате светодиода, подключенного к порту PB5. Этот светодиод обозначен на плате буквой «L» или числом «13».

```
#include <avr/io.h> //подключение стандартной библиотеки ввода/вывода

int main(void)
{
	DDRB  |= 1<<5; ; //пин 5 порта B работает как выход – светодиод
	DDRD = 0b00110000; //пины 4 и 5 порта D работают как выход
	PORTD |= 1<<5; //пин 5 порта D настроен на вывод +5 вольт
	PORTD &= ~1<<4; // пин 4 порта D настроен на вывод GND (-5 вольт)

	while(1) //бесконечный цикл (основная программа)
	{
	// нажата ли кнопка на пин 2?
	// при нажатой кнопке – низкий уровень –  GND (-5 вольт)
		if(~PIND & (1<<2))
		{
			PORTB |=1<<5; //да – светодиод на PB5 ВКЛ
		}
		else
		{
			PORTB &= ~1<<5; //нет – светодиод на PB5 ВЫКЛ
		}
	}
}
```

Разберем её построчно:

- #include <avr/io.h> – подключение стандартной библиотеки для ядра AVR. В ней находятся определения констант, имен регистров и всего прочего, что может понадобиться для выполнения базового ввода-вывода.
- Строки после «int main(void){» и до «while(1)» – это строки первичной инициализации периферии. В них выполняется настройка режимов работы портов и устанавливаются начальные состояния.
- DDRB |= 1<<5 – установка «1», в 5-ый бит регистра DDRB. Это установит порт PB5 в состояние «выход». Оператор «<<» это побитовый сдвиг влево, а запись «1<<5» означает, что в двоичном числе единицу необходимо сдвинуть пять раз, то есть начинается с 00000001 (позиция «0»), а в результате сдвига будет 0010000 (позиция «5»). Оператор «|=» производит логическое сложение содержимого регистра DDRB с числом «1<<5». При логическом побитовом сложении единицы с любым числом, результатом будет «1». При сложении любого числа с нулем, получится исходное число. Все это означает, что после выполнения команды «DDRB |= 1<<5;» в пятом бите регистра DDRB окажется «1», а остальные биты не изменят свое состояние.
- DDRD = 0b00110000 – установка выводов PD4 и PD5 на выход. Необходимо для использования платы расширения с кнопками.
- PORTD |= 1<<5 – установка высокого уровня на выводе PD5. Подача положительного потенциала от источника питания на плату с кнопками.
- PORTD &= ~1<<4 – установка низкого уровня на выводе PD4. Подача отрицательного потенциала от источника питания на плату с кнопками.
- PORTB &= ~1<<5 – Задание начального состояние уровня на выводы PB5. Запись «0», в пятый бит регистра PORTB «0», чтобы на выводе был низкий уровень – напряжение 0 Вольт. Оператор «~» означает инверсию, а запись «~1<<5» означает двоичное число 0b11011111, то есть число, обратное число 0b00100000. Это число также перемножается, знак «&», с содержимым регистра PORTB и результат записывается в регистр PORTB. Битовое умножение на «0» любого числа дает в результате «0». При умножении любого числа на «1» результатом будет исходное число. В итоге, после выполнения команды «PORTB &= ~1<<5;», в пятом бите регистра PORTB будет ноль, а остальные не изменят свое состояние. Данная строчка не является обязательной, но желательна для надежного «старта», с жестко задачным начальным состоянием на используемом выводе МК.
- «while(1) { ОСНОВНАЯ ПРОГРАММА }». Основная программа выполняется в теле цикла while пока условие входа в цикл истинно, то есть равно «1». Так как это значение задано изначально и не является переменной, то условие будет выполняться всегда и микроконтроллер,

пока подано питание, будет бесконечно, циклично, выполнять содержимое тела цикла.

- if(~PIND & (1<<2)) – оператор ветвления. Условие, указанное в скобках после if, читается так – на входе PD2 низкий уровень? Если «да», то делается код в первой паре фигурных кавычек. Если «нет» (на входе PD2 высокий уровень), то делается код внутри пары фигурных кавычек после «else». Вариант для «нет» может отсутствовать, тогда «else» не пишут.
- PORTB |= 1<<5 – установка «1» в 5-тый бит регистра состояния порта B, а следовательно высокого, 5 вольт, уровня на ножке PB5 (9 ножка микроконтроллера, 13 порт на отладочной плате). Включение пользовательского светодиода «L» («13»).
- PORTB &= ~1<<5 – установка «0» в 5-тый бит регистра состояния порта B, а следовательно, низкого, 0 вольт, уровня на ножке PB5. Выключение пользовательского светодиода «L» («13»).

Стоить отметить, что запись битов в регистры может быть выполнена и в другом виде.

Например:

DDRB = 0x20; – в шестнадцатеричном.

DDRB = 0b00100000; – в двоичном.

При этом, в обоих случаях, будут изменены и все другие биты регистра. Это удобно, когда нужно настроит на выход/вход или установить высокий/низкий уровень сразу на нескольких портах, но излишне, если необходимо изменить состояние только одного порта.

ПЛАТА РАСШИРЕНИЯ «КНОПКИ»

Для выполнения данной лабораторной работы дополнительно потребуется плата расширения «кнопки».

На этой плате установлены две кнопки без фиксации.

Конструкция платы разработана для установки как непосредственно в отладочную плату типа «Arduino», так и на макетную плату для беспаечного монтажа. Для работы платы требуется дополнительное питание.

Схема платы расширения приведена на рисунке 2.4.

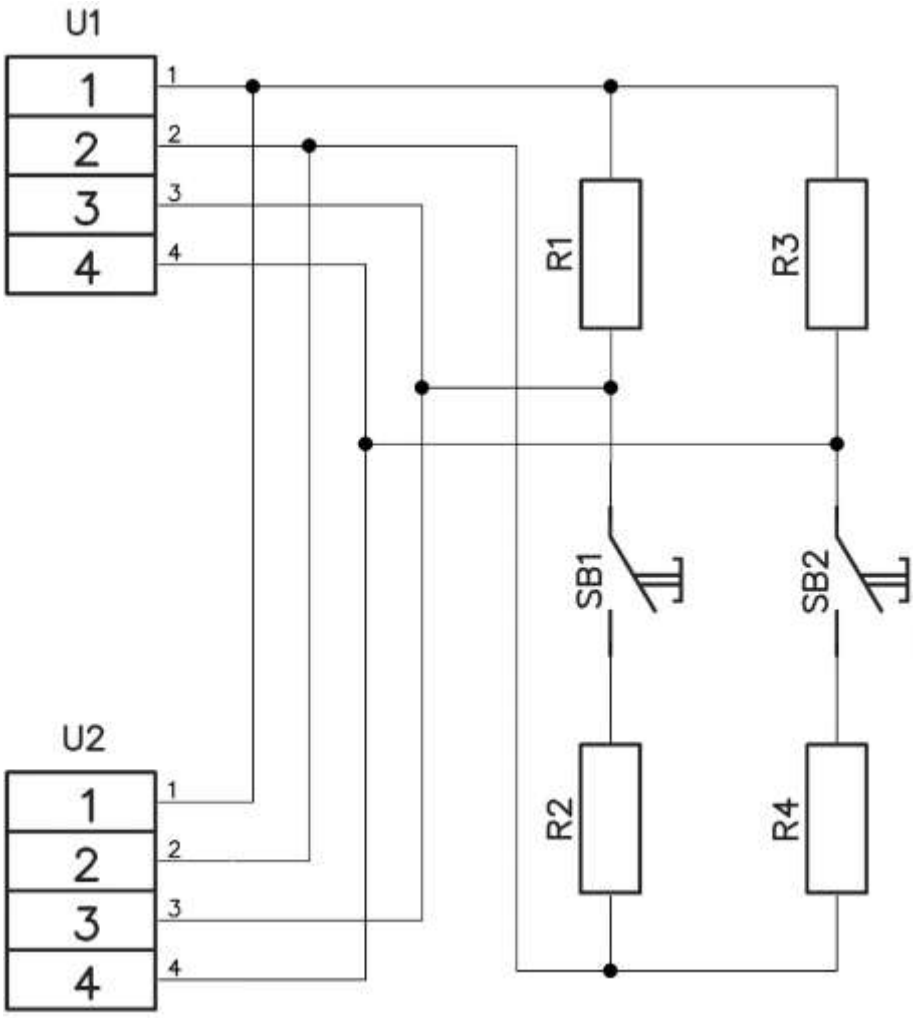

Рис. 2.4. Схема платы расширения «кнопки»

На схеме обозначены:

- R1, R3 – резисторы подтяжки. Резистор подтяжки обеспечивает высокий логический уровень на входе МК до тех пор, пока не нажата кнопка.
- R2, R4 – защитные резисторы. Предохраняют цепи питания и входа МК от больших токов.
- SB1, SB2 – кнопки без фиксации.

Плату расширения необходимо установить так, как показано на рисунке 2.5. Порты отладочной платы «7», «6» и «1», «0» не должны быть заняты.

При такой установке ближняя к микроконтроллеру кнопка будет подключена к выводу PD3, дальняя – PD2.

Выводы PD4 и PD5 необходимо настроить на выход. После чего установить высокий уровень на PD5 и низкий на PD4.

При нажатой кнопке в соответствующий бит регистра PIND будет записываться «0», при не нажатой – «1».

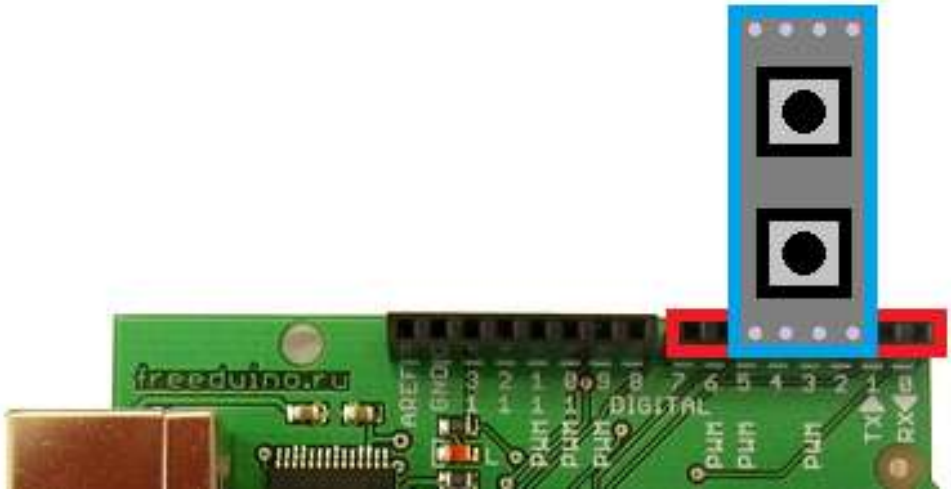

Рис. 2.5. Расположение платы расширения на отладочной плате

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Запустите Atmel Studio и выполните File -> New -> Project.

Появится окно создания нового проекта. В нижней его части Вы можете задать название проекта. Имя по умолчанию будет иметь вид «GccApplication1», при этом число в окончание может быть различным. Замените его на имя в формате «Группа, индивидуальный номер, случайное число до 100» (например, «EAb-083_06_53»). Свой индивидуальный номер можно узнать у преподавателя.

В появившемся окне выберите «GCC C Executable Project». После этого отобразится окно выбора используемого в проекте микроконтроллера. Найдите в списке «ATmega328P». Кликните мышкой по этой строке и нажмите ОК. Рекомендуется воспользоваться строкой поиска, в правом верхнем углу окна выбора микроконтроллера.

На странице с текстом программы сотрите все и наберите программу из разобранного выше примера.

Выполните Build -> Build Solutions, либо нажмите F7.

Дождитесь окончания компиляции. В нижнем окне «Output» должны появиться строчки:

Build succeeded.

========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========

Для того чтобы непосредственно поместить содержимое этого файла в память микроконтроллера, необходимо воспользоваться дополнительной программой – XLoader.

Подключите отладочную плату к компьютеру, используя кабель типа USB-A – USB-B, и запустите программу XLoader.

Окно программы содержит следующие элементы:

- Hex file – здесь нужно указать путь до загружаемого HEX файла;

- Device – здесь необходимо выбрать тип платы и микроконтроллера. Duemilanove/Nano(тип МК – ATmeg328);
- COM port – здесь необходимо выбрать виртуальный комп порт, через который компьютер общается с МК;
- Baud rate – скорость передачи данных через COM-порт. Зависит от типа МК и выставляется автоматически. Для ATmega328 – 57600.

Полный путь до папки будет иметь вид:
Документы\Atmel Studio\6.2\Имя_проекта\ Имя_проекта\Debug

В папке Debug необходимо найти файл с расширением «.hex». Именно этот файл «заливается» в микроконтроллер. В нём содержится программа для микроконтроллера в виде машинных кодов.

Если все установки правильные, то после нажатия кнопки «Upload» программа свяжется с загрузчиком в МК и, получив от него ответ, загрузит содержимое HEX файла в память МК. Процесс обмена данными можно наблюдать визуально, по мерцанию расположенных на отладочной плате светодиодов RX и TX.

ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ

Подключите плату расширения «трехцветный светодиод».

Определите, какие порты микроконтроллера необходимо настроить как выход. Выполните настройку в соответствующем регистре DDRx.

Выполните индивидуальное задание. Номер задания уточняйте у преподавателя.

Напишите программу, реализующую следующий режим работы светодиодов:

1) Количество нажатий на одну из кнопок подсчитывается. При нажатии на другую кнопку любой светодиод вспыхнет столько раз, сколько раз была нажата первая кнопка. После этого счетчик нажатий первой кнопки обнуляется.

2) Нажатие на одну кнопку запускает следующий алгоритм – один из светодиодов вспыхивает (кратковременное включение) два раза, после чего другой светодиод включается на 3 секунды. Придумайте собственную последовательность.

3) При нажатии на одну кнопку загорается красный светодиод, на другую – синий, на обе – зеленый.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение регистра PINx.
2. Способ подключение внутреннего подтягивающего резистора.
3. Назначение подтягивающего резистора. Способы подтяжки вывода микроконтроллера.

4. Возможно ли опросить состояние пина микроконтроллера, настроенного на выход, а не на вход?

ЛАБОРАТОРНАЯ РАБОТА №3.

Написание программы для работы с таймерами микроконтроллера ATmega328. Работа таймеров в режиме ШИМ.

Цель работы: изучить работу таймеров/счетчиков; научиться настраивать таймеры для генерации сигнала с широтной-импульсной модуляцией заданной частоты и скважности.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Таймер-счетчик является одним из самых используемых периферийных устройств микроконтроллера. Его основное назначение – отсчитывать заданные временные интервалы. Кроме того, таймеры-счетчики могут выполнять ряд дополнительных функций – формировать сигналы с широтной-импульсной модуляцией (ШИМ), вести подсчет длительности и количества входящих импульсов.

В микроконтроллере ATmega328 есть три таймера – 2 по 8 бит (T0, T2) и один на 16 бит (T1). Для них всех есть одинаковые режимы работы:

- Обычный режим работы.
  Самый распространенный режим, когда таймер просто считает приходящие импульсы и при переполнении счетного регистра устанавливает флаг прерывания по переполнению. При этом счетный регистр сбрасывается в 0 и подсчет импульсов начинается сначала.

- Режим подсчета импульсов (Сброс при совпадении).
  Также называется CTC. В этом режиме при совпадении счетного регистра с одним из регистров сравнения выставляется флаг прерывания по совпадению, счетный регистр обнуляется и подсчет начинается сначала. При этом по совпадению может меняться сигнал на выходе таймера, соответствующего используемому регистру совпадения.

- Режим ШИМ.
  В данном режиме изменяется ширина импульса в зависимости от значения, записанного в регистр совпадения.

- Режим коррекции фазы ШИМ.
  В обычном режиме ШИМ мы получаем плавающую фазу выходного сигнала. Для того чтобы этого избежать, в режиме коррекции фазы

ШИМ при достижении максимального значения, счетный регистр таймера/счетчика начинает уменьшаться. Это происходит циклически.

Это основные режимы работы таймеров/счетчиков. У таймера/счетчика T0 присутствуют только они, а T1 и T2 имеют дополнительные режимы.

Дополнительные режимы работы таймера/счетчика T2:

- Асинхронный режим работы.
  В асинхронном режиме работы к микроконтроллеру подключается внешний кварцевый резонатор 32 кГц. Чаще всего этот режим используется для работы таймера/счетчика T2 в качестве часов реального времени.

Дополнительные режимы работы таймера/счетчика T1:

- Режим коррекции фазы и частоты ШИМ.
  Отличается от режима коррекции фазы ШИМ лишь моментом обновления регистра сравнения. Регистр сравнения обновляется, когда значение счетного регистра достигает минимума.

- Режим захвата.
  При поступлении сигнала от аналогового компаратора, а также на вывод ICP1 (14 ножка МК) значение счетного регистра сохраняется в регистре захвата, устанавливая при этом флаг прерывания по захвату.

Если таймер работает в режиме счётчика, то он считает количество импульсов, поступивших на выбранный вход микроконтроллера. В этом случае порт, связанный со счётчиком таймера, должен быть настроен на вход.

Если таймер работает в качестве таймера, то частота его тактирования зависит от частоты генератора такта микроконтроллера.

Каждый такт счетчик в таймере увеличивается на единицу (итерация). Переполнение 8 битного счётчика наступает при достижении 255 итераций, 16 битного – при достижении 65535 итераций.

Полное описание возможностей и настройки регистров для всех таймеров приведены в приложение 1.

ШИРОТНО-ИМПУЛЬСНАЯ МОДУЛЯЦИЯ

Широтно-импульсная модуляция (ШИМ, англ. pulse-width modulation (PWM)) – управление средним значением напряжения на нагрузке путём изменения скважности импульсов, управляющих силовым ключом. Различают аналоговую ШИМ и цифровую ШИМ, двоичную (двухуровневую) ШИМ и троичную (трёхуровневую) ШИМ.

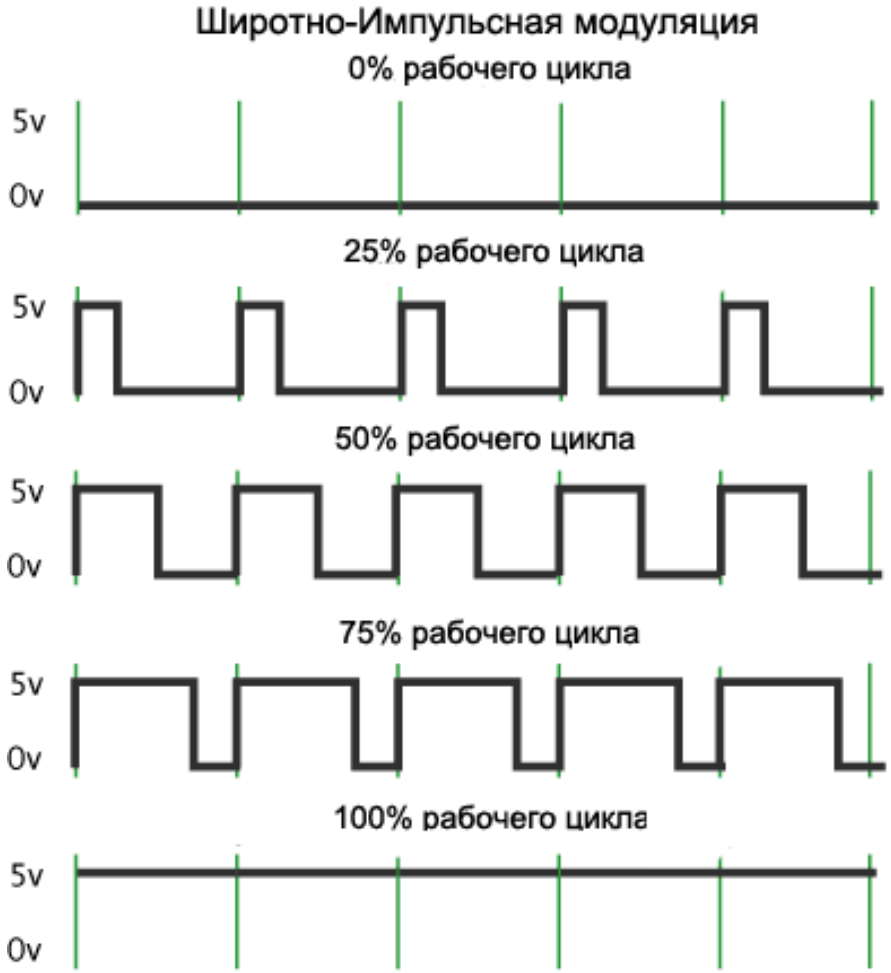

Рис. 3.1. Форма выходного сигнала с широтно-импульсной модуляцией

На рисунке 3.1 приведен вид формы выходного сигнала с широтно-импульсной модуляцией.

Зеленые линии отмечают постоянные временные периоды. Длительность периода обратно пропорциональна частоте ШИМ. Т.е. если частота ШИМ составляет 500 Гц, то зеленые линии будут отмечать интервалы длительностью в 2 миллисекунды каждый.

Заполнение периода высоким уровнем определяется параметром «скважность». От этого заполнения зависит «выходное напряжение». В простом приближение можно считать, что доля заполнения – это процент от максимального выходного напряжения, равного напряжению питания МК.

Задание скважности выполняется регистром сравнения OCRxx. Когда значение в счётном регистре таймера достигает значения, находящегося в регистре сравнения, то могут возникнуть следующие аппаратные события:

- прерывание по совпадению;
- изменение состояния внешнего выхода сравнения OCxx. Эти выходы выведены на выводы микроконтроллера.

Предположим, что мы настроили наш ШИМ генератор так, что когда значение в счетном регистре больше, чем в регистре сравнения, то на выходе у нас 0, а когда меньше, то 1.

Что при этом произойдет? Таймер будет считать, как ему и положено, от нуля до 255, с частотой, которую мы настроим битами пред-делителя таймера. А когда число в счетчике таймера совпадет с числом в регистре OCxx, тогда состояние выхода OCxx изменится с высокого на низкое. После переполнения счетчик таймера сбрасывается в 0 и продолжает считать по новой. В этом случае на выводе OCxx снова 1, до тех пор, пока число в счетчике вновь на совпадет с числом в регистре OCxx. Такой режим работы ШИМ называется неинверсным. Его график показан на рисунке 3.2.

Возможно настроить и наоборот – переход состояние выхода из 0 в 1, при совпадение числа в OCxx с числом в счетчике таймера. Это инверсный ШИМ.

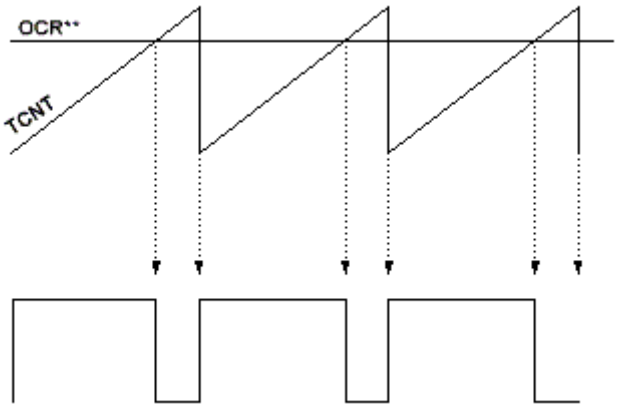

Рис. 3.2. График работы неинверсного ШИМ

Таким образом, получается, что, изменяя значение в регистре сравнения можно менять скважность ШИМ сигнала. А если пропустить этот ШИМ сигнал через сглаживающую RC цепочку (интегратор), то получим непрерывный, аналоговый, сигнал.

ПРИМЕР ПРОГРАММЫ ГЕНЕРИРАЦИИ ШИМ СИГНАЛА ДЛЯ УПРОВЛЕНИЯ ЯРКОСТЬЮ СВЕТОДИОДОВ

```
#include <avr/io.h> //подключение стандартной библиотеки ввода/вывода
#define F_CPU 16000000UL //определение тактовой частоты микроконтроллера для корректной работы функций задержки
#include <util/delay.h> //подключение библиотеки для генерации задержек
```

```
int main(void)
{
	//SETUP PORTS
	DDRB=0b00001110; //порт D работает как выход, так как к нему относятся выводы OC2A, OC1A и OC1B
	//SETUP TIMER 2
	TCCR2A=0b10100011; //выбираем неинверсный режим шим для обоих выходов таймера 2
	TCCR2B=0b00000001;  //выбираем работу таймера без делителя
	//SETUP TIMER 1
	TCCR1A=0b10100001; //выбираем неинверсный режим шим для обоих выходов таймера 1
	TCCR1B=0b00001001; //выбираем работу таймера без делителя

	while(1) //бесконечный цикл (основная программа)
	{

		for(int i=0;i<256;i++) //увеличиваем яркость диода каждые 10 мс
		{
			OCR2A=i;
			_delay_ms(10);
		}

	}
}
```

TCCR2A=0b10100011 – настройка регистра TCCR2A, для Таймера 2:

- Биты WGM21, WGM20 (позиции 1 и 0) регистра TCCR0A устанавливают режим работы таймера/счетчика T0: 11 – режим ШИМ.

- Биты COM2A1 и COM2A0 (позиции 7 и 6) влияют на то, какой сигнал появится на выводе OC2A при совпадении значения счетного регистра TCNT2 со значением регистра сравнения OCR2A. Сейчас установлено 10 – сброс вывода OC2A в 0 при совпадении с A, установка вывода OC2A в 1, если регистр TCNT0 принимает значение 0x00 (неинверсный режим).

- Биты COM2B1 и COM2B0 (позиции 5 и 4) влияют на то, какой сигнал появится на выводе OC2B (12 ножка) при совпадении с B (совпадение значения счетного регистра TCNT2 со значением регистра сравнения OCR2B). Сейчас установлено 10 – сброс вывода OC2B в 0 при совпадении с A, установка вывода OC2B в 1, если регистр TCNT0 принимает значение 0x00 (неинверсный режим).

TCCR2B=0b00000001 – настройка регистра TCCR2B, для Таймера 2:

- Биты CS22, CS21, CS20 (позиции 2, 1 и 0) регистра TCCR2B устанавливают режим тактирования и предделителя тактовой частоты

таймера/счетчика T2. Сейчас выставлены биты 001 – тактовый генератор CLK, без делителя.

Установкой режима работы таймера/счетчика T1 выполнятся немного сложнее – двумя парами бит в разных регистрах. Битами WGM13, WGM12 (позиция 4 и 3, в регистре TCCR1B) и битами WGM11, WGM10 (позиция 1 и 0, в регистре TCCR1A). В обоих парах записано 01 – PWM, 8-бит.

TCCR1A=0b10100001 – настройка регистра TCCR1A, для Таймера 1:

- Биты COM1A1 и COM1A0 (позиции 7 и 6) влияют на то, какой сигнал появится на выводе OC1A при совпадении значения счетного регистра TCNT2 со значением регистра сравнения OCR01A.

- Биты COM1B1 и COM1B0 (позиции 5 и 4) влияют на то, какой сигнал появится на выводе OC1B (12 ножка) при совпадении с B (совпадение значения счетного регистра TCNT2 со значением регистра сравнения OCR1B).

TCCR1B=0b00001001 – настройка регистра TCCR1B, для Таймера 1:

- Биты CS12, CS11, CS10 (позиции 2, 1 и 0) регистра TCCR2B устанавливают режим тактирования и предделителя тактовой частоты таймера/счетчика T2. Сейчас выставлены биты 001 – тактовый генератор CLK, без делителя.

ПЛАТА РАСШИРЕНИЯ «ТРЕХЦВЕТНЫЙ СВЕТОДИОД»

Для выполнения данной лабораторной работы дополнительно потребуется плата расширения «трехцветный светодиод».

На это плате установлен RGB (Red, Green. Blue) светодиод. Конструктивно он состоит из трех отдельных светодиодов, собранных в одном корпусе. Это позволяет получать или три цвета поочередно, или любой другой цвет, путём включения сразу двух светодиодов и регулировки их яркости. Включение всех трёх позволит получить белый цвет.

Схема платы расширения приведена на рисунке 3.3.

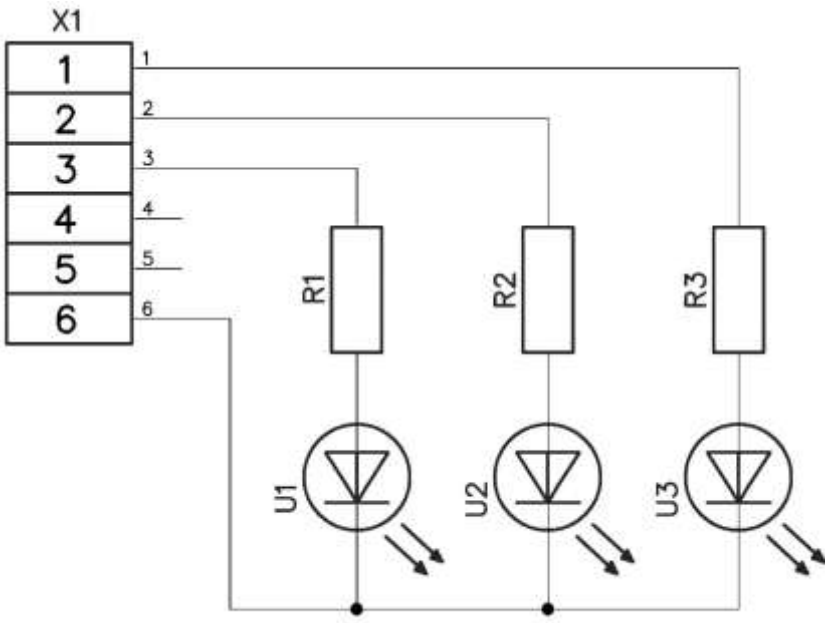

Рис. 3.3. Схема платы расширения «трехцветный светодиод»

На схеме обозначены:

- R1-R3 – резисторы. Они ограничивают ток через светодиод, а также порт микроконтроллера. Ток не превышает 15 mA.
- U1-U3 – светодиоды. Зелёный, красный и синий.

Плату расширения необходимо установить так, как показано на рисунке 3.4. Порты отладочной платы «8» и «AREF» не должны быть заняты.

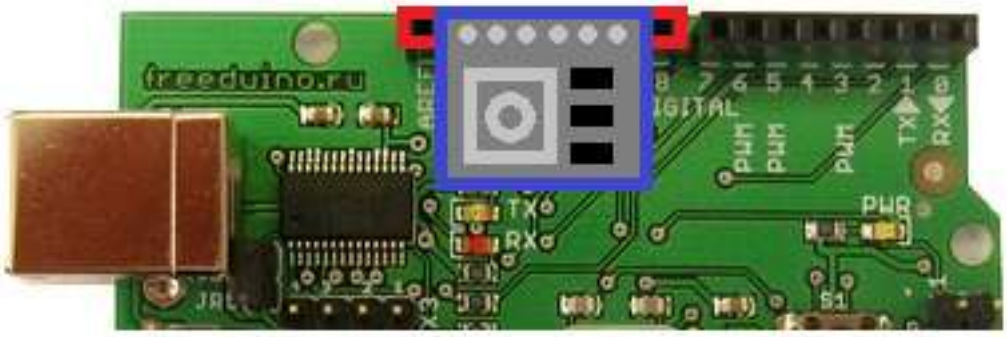

Рис. 3.4. Расположение платы расширения на отладочной плате

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Запустите Atmel Studio и выполните File -> New -> Project.

Появится окно создания нового проекта. В нижней его части Вы можете задать название проекта. Имя по умолчанию будет иметь вид «GccApplication1», при этом число в окончании может быть различным. Замените его на имя в формате «Группа_индивидуальный номер случайное число до 100» (например, «EAb-083_06_53»). Свой индивидуальный номер можно узнать у преподавателя.

В появившемся окне выберите «GCC C Executable Project». После этого отобразится окно выбора используемого в проекте микроконтроллера. Найдите в списке «ATmega328» (или «ATmega168»). Кликните мышкой по этой строке и нажмите ОК. Рекомендуется воспользоваться строкой поиска, в правом верхнем углу окна выбора микроконтроллера.

На странице с текстом программы сотрите все и наберите программу из разобранного выше примера.

Выполните Build -> Build Solutions, либо нажмите F7.

Дождитесь окончания компиляции. В нижнем окне «Output» должны появится строчки:

Build succeeded.

========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========

Для того чтобы непосредственно поместить содержимое этого файла в память микроконтроллера, необходимо воспользоваться дополнительной программой – XLoader.

Подключите отладочную плату к компьютеру, используя кабель типа USB-A – USB-B, и запустите программу XLoader.

Окно программы содержит следующие элементы:

- Hex file – здесь нужно указать путь до загружаемого HEX файла;
- Device – здесь необходимо выбрать тип платы и микроконтроллера. Duemilanove/Nano(тип МК – ATmega168 или ATmeg328);
- COM port – здесь необходимо выбрать виртуальный комп порт, через который компьютер общается с МК;
- Baud rate – скорость передачи данных через COM-порт. Зависит от типа МК и выставляется автоматически. Для ATmega168 – 19200, для ATmega328 – 57600.

Полный путь до папки будет иметь вид:

Документы\Atmel Studio\6.2\Имя_проекта\ Имя_проекта\Debug

В папке Debug необходимо найти файл с расширением «.hex». Именно этот файл «заливается» в микроконтроллер. В нём содержится программа для микроконтроллера в виде машинных кодов.

Если все установке правильные, то после нажатия кнопки «Upload» программа свяжется с загрузчиком в МК и, получив от него ответ, загрузит содержимое HEX файла в память МК. Процесс обмена данными можно наблюдать визуально, по мерцанию расположенных на отладочной плате светодиодов RX и TX.

ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ

Подключите плату расширения «трехцветный светодиод».

Определите, какие порты микроконтроллера необходимо настроить как выход. Выполните настройку в соответствующем регистре DDRx.

Напишите программу, реализующую один из следующих режимов работы светодиодов:

1) Настроить оба выхода Таймера 2 на работу в режиме ШИМ. Написать программу для поочередного нарастания и затухания яркости каждого из трех светодиодов.
2) Настроить оба выхода Таймера 2 на работу в режиме ШИМ. Написать программу для одновременной работы любой пары светодиодов. Когда яркость одного возрастает, то у другого она убывает.
3) Настроить оба выхода Таймера 2 на работу в режиме ШИМ. Написать программу для работы всех светодиодов в режиме «радуги».

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение таймеров.
2. Режимы работы таймеров.
3. Типы таймеров в микроконтроллере ATmega328.
4. Настройка Таймера 1 в режим генерации ШИМ.
5. Настройка Таймера 2 в режим генерации ШИМ.

ЛАБОРАТОРНАЯ РАБОТА №4.

Разработка программы для использования прерывания в микроконтроллере ATmega328. Прерывание по таймеру. Внешние прерывания.

Цель работы: изучить принципы работы различных прерываний, получить первичные навыки написания программ с использованием прерываний.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Прерывание – это остановка выполнения текущей программы извне на время выполнения некоторой подпрограммы (которая называется "обработчик прерывания").

Любое прерывание имеет «вектор» – строчку кода, указывающие на место в памяти, где начинается код программы, который необходимо выполнить при срабатывании этого прерывания.

По окончания выполнения вызванной прерыванием программы микроконтроллер возвращается к тому месту основной программы, где его «прервали».

Например, если микроконтроллер находился внутри функции задержки, «delay», то он продолжит выполнять её с той же позиции счетчика, на которой находился в момент вызова прерывания.

Прерывания имеют приоритет. Чем выше адрес прерывания в блоке векторов, тем ниже приоритет прерывания, то есть в очереди прерываний, прерывания с большим приоритетом будут выполняться раньше. Если началась обработка какого-нибудь прерывания, никакое другое не может быть вызвано, даже с большим приоритетом.

Источниками прерывания могут быть таймеры, регистры последовательного порта, внешние выводы, SPI, АЦП и т.д.

Внешнее прерывание может срабатывать при изменении уровня на выводе INTx с низкого на высокий или наоборот, все время при наличии низкого уровня, или же при любом изменение уровня.

Прерывания по таймеру могу срабатывать при совпадении счетчика таймера с заданным числом или же при его переполнении.

ПРЕРЫВАНИЕ ПО ПЕРЕПОЛНЕНИЮ ТАЙМЕРА

Ниже приведен пример программы для работы с прерыванием по переполнению таймера.

Программа реализует следующий алгоритм:

1) Включение светодиода на выводе PB5;

2) Ожидание 1 секунду;
3) Выключение светодиода на выводе PB5;
4) Ожидание 1 секунду;
5) Бесконечное повторение предыдущих пунктов.

Такой же алгоритм выполнялся программой из первой лабораторной работы, но теперь есть существенное отличие – теперь в течение 1 секунды между сменами состояний светодиода микроконтроллер может выполнять любые действия, тогда как прежде он бездействовал.

```
#include <avr/io.h> // Подключаем библиотеку ввода-вывода
#include <avr/interrupt.h> // Подключаем библиотеку прерываний
#define CPU _SPEED 16000000 // Записываем тактовую частоту

// Функция для инициализации прерывания
void t1_init()
{
	// Настраиваем делитель на 1024
	TCCR1B = (1<<CS12)|(0<<CS11)|(1<<CS10);
	// Выставляем значение TCNT1
	// для коррекции счетчика, чтобы время было ровно 1 секунде
	TCNT1 = 65535 – CPU _SPEED/1024;
	// Разрешаем прерывание по переполнению таймера
	TIMSK1 |= (1<<TOIE1);
}

// Функция для действий при срабатывании прерывания
ISR(TIMER1_OVF_vect)
{
	// Выставляем значение TCNT1
	// для коррекции счетчика, чтобы время было ровно 1 секунде
	TCNT1 = 65535 – CPU _SPEED/1024;
	// Делаем каждую секунду
// инверсию 5-го бита в регистре PORTB
// в результате мигаем светодиодом на выводе PB5
	PORTB ^=1<<5;
}

int main(void)
{
	// Все вывода PBx настроены на выход
DDRB = 0xFF;
 // Начальное состояние на всех выводах PBx – низкий уровень
	PORTB = 0x00;
```

```
// вызываем функцию инициализации прерывания
// по переполнению таймера
        t1_init();

        // Разрешаем все прерывания – функция из библиотеки interrupt.h
        sei();

    while(1)
    {
      // В главном цикле ничего не происходит.
    }
}
```

Строка «TCNT1 = 65535 – CPU _SPEED/1024;» непосредственно определяет продолжительность периода времени между вызовом прерывания по переполнению таймера.

Переполнение таймера – условное название. На самом деле переполняется счетный регистр TCNTx. Так как Таймер 1 имеет счетный регистры TCNT1 с длинной в 16 бит, то максимальное число, которое он может хранить в себе составляет, 65535. Как только это число будет достигнуто, тогда и вызовется вектор прерывания по переполнению таймера – TIMER1_OVF_vect. Этот вектор укажет на место в памяти программ, где лежит фрагмент программы, которые необходимо выполнять при срабатывании данного прерывания.

Для того чтобы переполнение наступило через желаемый промежуток времени, необходимо рассчитать за какое время заполнится счетный регистр при заданной тактовой частоте микроконтроллера. Простейшей способ – разделить максимальное значение счетчика на таковую частоту микроконтроллера. Полученное время будет измеряться в секундах. Для частоты 16 МГц время переполнения составит примерно 4 миллисекунды.

Для уменьшения этого числа необходимо при инициализации и каждом срабатывании прерывания записывать в счетный регистр необходимое число. Тем самым регистр будет считать не с нуля и переполняться быстрее.

Для увеличения промежутка времени необходимо уменьшить скорость заполнения регистра. Это достигается включением пред-делителя. В приведенном примере предделитель настроен на максимальное значение – 1024. Это значит, что счетчик будет увеличиваться на единицу каждые 1024 такта.

Однако, 16МГц / 1024 = 15625. Отсюда видно, что полученное число меньше чем максимальное значение счетного регистра. Следовательно, его переполнение произойдёт не через 1 секунду, а, примерно, через 4,1, что больше, чем нам необходимо. Поэтому мы все время записываем в счетный регистр число, получаемое по формуле: 65535 – 16000000/1024 = 49910.

Интересным решением является отдать расчёт начального числа непосредственно микроконтроллеру. И хотя это немного замедляет его работу, но упрощает перенос программы на такой же микроконтроллер, но работающий с иной тактовой частотой.

ВНЕШНЕЕ ПРЕРЫВАНИЕ

Ниже приведён пример программы для демонстрации работы внешнего прерывания по входу INT1.

Программа включает/выключает светодиод на выводе PB5 независимо от работы основного цикла, в котором происходит включение/выключение светодиода на выводе PB2, с использование функции delay.

```
#include <avr/io.h>
#include <avr/interrupt.h>
#define F_CPU 16000000UL
#include <util/delay.h>

//настройка внешнего прерывания INT1
void int1_init( void )
{
//настраиваем на срабатывание INT1 по переднему фронту
EICRA |= (0<<ISC11)|(0<<ISC10);
//разрешаем внешнее прерывание INT1
EIMSK |= (1<<INT1);
}


//функция обработчик внешнего прерывания INT1
ISR( INT1_vect )
{
        // Делаем инверсию 5-го бита в регистре PORTB
// в результате управляем светодиодом на выводе PB5
PORTB ^= 1<<5;
}


int main( )
{
DDRD = 0b00110000;
PORTD |= 1<<5;
PORTD &= ~1<<4;
DDRB = 0xFF;
PORTB = 0x00;
```

```
int1_init();
sei();

while(1)
{
        PORTB |= 1<<2;
        _delay_ms( 1000 );
        PORTB &= ~1<<2;
        _delay_ms( 1000 );
}
return 0;
}
```

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Для выполнения первой обоих работ присоедините к отладочной плате платы расширения «трёхцветный светодиод» и «кнопки».

Запустите Atmel Studio и выполните File -> New -> Project.

Появится окно создания нового проекта. В нижней его части Вы можете задать название проекта. Имя по умолчанию будет иметь вид «GccApplication1», при этом число в окончание может быть различным. Замените его на имя в формате «Группа_индивидуальный номер случайное число до 100» (например, «EAb-083_06_53»). Свой индивидуальный номер можно узнать у преподавателя.

В появившемся окне выберите «GCC C Executable Project». После этого отобразится окно выбора используемого в проекте микроконтроллера. Найдите в списке «ATmega328» (или «ATmega168»). Кликните мышкой по этой строке и нажмите ОК. Рекомендуется воспользоваться строкой поиска, в правом верхнем углу окна выбора микроконтроллера.

На странице с текстом программы сотрите все и наберите программу из разобранного выше примера.

Выполните Build -> Build Solutions, либо нажмите F7.

Дождитесь окончания компиляции. В нижнем окне «Output» должны появится строчки:

Build succeeded.

========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========

Для того чтобы непосредственно поместить содержимое этого файла в память микроконтроллера, необходимо воспользоваться дополнительной программой – XLoader.

Подключите отладочную плату к компьютеру, используя кабель типа USB-A – USB-B, и запустите программу XLoader.

Окно программы содержит следующие элементы:

- Hex file – здесь нужно указать путь до загружаемого HEX файла;
- Device – здесь необходимо выбрать тип платы и микроконтроллера. Duemilanove/Nano(тип МК – ATmega168 или ATmeg328);
- COM port – здесь необходимо выбрать виртуальный комп порт, через который компьютер общается с МК;
- Baud rate – скорость передачи данных через COM-порт. Зависит от типа МК и выставляется автоматически. Для ATmega168 – 19200, для ATmega328 – 57600.

Полный путь до папки будет иметь вид:

Документы\Atmel Studio\6.2\Имя_проекта\ Имя_проекта\Debug

В папке Debug необходимо найти файл с расширением «.hex». Именно этот файл «заливается» в микроконтроллер. В нём содержится программа для микроконтроллера в виде машинных кодов.

Если все установке правильные, то после нажатия кнопки «Upload» программа свяжется с загрузчиком в МК и, получив от него ответ, загрузит содержимое HEX файла в память МК. Процесс обмена данными можно наблюдать визуально, по мерцанию расположенных на отладочной плате светодиодов RX и TX.

ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ

Напишите программу, реализующую следующий режим работы светодиодов:

1) Переключение светодиодов по внутреннему прерыванию.

2) Переключение светодиодов по внешнему прерыванию.

3) Выполнение опроса кнопки по внутреннему прерыванию. Если в момент опроса кнопка нажата, то состояние светодиода должно изменится на противоположное.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение прерываний.
2. Источники прерываний.
3. Настройка Таймера 1 на работу в режиме прерывания.
4. Настройки внешнего прерывания.

ЛАБОРАТОРНАЯ РАБОТА №5.

Разработка программы для модуля UART микроконтроллера ATmega328. Управление выходами микроконтроллера по командам с ПК.

Цель работы: изучить принципы работы модуля приема-передатчика UART, получить первичные навыки написания программ для реализации последовательного протокола связи через UART.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

UART – Universal Asynchronous Receiver-Transmitter (Универсальный асинхронный приёмопередатчик) – периферийный блок микроконтроллера, предназначенный для организации связи с другими цифровыми устройствами по последовательному интерфейсу. Передача данных через UART выполняется в виде последовательной цепочки бит. Для этого UART преобразует передаваемые данные в последовательный вид. Метод преобразования стандартизован и широко применялся в компьютерной технике.

UART представляет собой логическую схему, с одной стороны, подключённую к шине вычислительного устройства, а с другой имеющую два (прием/передача) или более выводов для внешнего соединения.

Он позволяет передать данные, используя различные стандартные протоколы, такие как RS-232, RS-485, MODBUS и т.д.

Настройка UART в микроконтроллере ATmega328 выполняется с помощью соответствующих регистров. Полное описание этих регистров и их настроек приведено в документации на микроконтроллер.

ПРИЕМ И ПЕРЕДАЧА ДАННЫХ С ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА

Ниже приведен пример программы для работы с UART в режиме повторения отправленного сообщения.

Микроконтроллер принимает посланные ему байт и сразу отправляет его обратно. Это позволяет убедиться в исправности UART и отсутствие потерь на линии связи между персональным компьютером и микроконтроллером.

```
#include <avr/io.h>

void uart_init( void ) // функция инициализации UART
{
```

```
        //настройка скорости обмена
        UBRR0H = 0;
        UBRR0L = 103; // ( 16000000/( 9600*16 ) ) – 1 = 103     число для 16 МГц и скорости 9600 бод
        UCSR0A=0x00;
        UCSR0C = ( 1 << UCSZ01 ) | ( 1 << UCSZ00 ); // настройка на прием 8-ми битных данных, без контроля четности
        //разрешить прием и передачу данных
        UCSR0B = ( 0 << UCSZ02 );
        UCSR0B = ( 1 << TXEN0 ) | ( 1 <<RXEN0 );
    }

    unsigned char uart_getc( void ) // функция забирает полученный байт из буфера и записывает его в переменную C
    {
        //ждем приема байта
        while( ( UCSR0A & ( 1 << RXC0 ) ) == 0  );
        //считываем принятый байт
        return UDR0;
    }

    void uart_putc( char c ) // функция для получения одного символа. При передаче строки получим набор символов
    {
        //ждем окончания передачи предыдущего байта
        while( ( UCSR0A & ( 1 << UDRE0 ) ) == 0 );
        UDR0 = c;
    }

    void uart_puts( char *str )   // функция для упрощения отправки текстовых сообщений
    {
        unsigned char c;
        while( ( c = *str++ ) != 0 ) // поочередно отправляем один байт из массива – передаем строку по буквам
        {
            uart_putc( c );
        }
    }

    int main( void )
    {
        uart_init(); //запускаем UART
```

```
	while(1)
{
		char c = uart_getc(); // принимаем символ
		uart_putc( c ); // передаем символ
	}
	return 0;
}
```

УПРАВЛЕНИЕ СВЕТОДИОДОМ С ПЕРСОНАЛЬНОГО КОМПЬЮТЕРА

Ниже приведён пример программы для демонстрации возможности управления удаленным устройством (светодиодом) с персонального компьютера через UART.

Отправляя в порт «1» или «0» через программу-терминал, можно включать и выключать светодиод на PB5.

```
#include <avr/io.h>

void uart_init( void )
{
	//настройка скорости обмена
	UBRR0H = 0;
	UBRR0L = 103; // ( 16000000/( 9600*16 ) ) – 1 = 103     число для 16 МГц и скорости 9600 бод
	//8 бит данных, 1 стоп бит, без контроля четности
	//UCSR0C = ( 1 << URSEL) | ( 1 << UCSZ01 ) | ( 1 << UCSZ00 ); // настройка на прием 8-ми битных данных
	UCSR0A=0x00;
	UCSR0C = ( 1 << UCSZ01 ) | ( 1 << UCSZ00 ); // настройка на прием 8-ми битных данных
	//разрешить прием и передачу данных
	UCSR0B = ( 0 << UCSZ02 );
	UCSR0B = ( 1 << TXEN0 ) | ( 1 <<RXEN0 );
}

unsigned char uart_getc( void )
{
	//ждем приема байта
	while( ( UCSR0A & ( 1 << RXC0 ) ) == 0  );
	//считываем принятый байт
	return UDR0;
}
```

```
void uart_putc( char c ) //самодельная функция для получения одного символа. При передаче строки получим набор символов
{
	//ждем окончания передачи предыдущего байта
	while( ( UCSR0A & ( 1 << UDRE0 ) ) == 0 );
	UDR0 = c;
}

int main( void )
{
	DDRB  |= 1<<5; // подключаем светодиод на плате

	uart_init(); //запускаем UART

	while( 1 )
	{
		char c = uart_getc();
		if(c == '1')
		{
			PORTB |= 1<<5;//Led_ON
		}
		if(c == '0')
		{
			PORTB &= ~1<<5;//Led_OFF
		}

	}
	return 0;
}
```

ПЕРЕДАЧА ИНФОРМАЦИИ О СОСТОЯНИИ КНОПОК НА ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР

Ниже приведён пример программы для демонстрации возможности получения информации о состоянии датчиков (кнопок) или оборудования на персональный компьютер через UART.

Контроллер постоянно, с интервалом в 300 мс, отправляет в порт составную строку с текстовой информацией о состоянии кнопок.

```
#include <avr/io.h>
#define F_CPU 16000000UL //16MHz
#include <util/delay.h>

void uart_init( void )
```

```
{
	//настройка скорости обмена
	UBRR0H = 0;
	UBRR0L = 103; // ( 16000000/( 9600*16 ) ) – 1 = 103     число для 16 МГц и скорости 9600 бод
	//8 бит данных, 1 стоп бит, без контроля четности
	//UCSR0C = ( 1 << URSEL) | ( 1 << UCSZ01 ) | ( 1 << UCSZ00 ); // настройка на прием 8-ми битных данных
	UCSR0A=0x00;
	UCSR0C = ( 1 << UCSZ01 ) | ( 1 << UCSZ00 ); // настройка на прием 8-ми битных данных
	//разрешить прием и передачу данных
	UCSR0B = ( 0 << UCSZ02 );
	UCSR0B = ( 1 << TXEN0 ) | ( 1 <<RXEN0 );
}

unsigned char uart_getc( void )
{
	//ждем приема байта
	while( ( UCSR0A & ( 1 << RXC0 ) ) == 0  );
	//считываем принятый байт
	return UDR0;
}

void uart_putc( char c ) //самодельная функция для получения одного символа. При передаче строки получим набор символов
{
	//ждем окончания передачи предыдущего байта
	while( ( UCSR0A & ( 1 << UDRE0 ) ) == 0 );
	UDR0 = c;
}

void uart_puts( char *str ) //самодельная функция для упрощения отправки текстовых сообщений
{
	unsigned char c;
	while( ( c = *str++ ) != 0 ) {
		uart_putc( c );
	}
}

int main( void )
{
	DDRD = 0b00110000;
```

```
	PORTD |= 1<<5;
	PORTD &= ~1<<4;



	uart_init(); //запускаем UART

	while( 1 )
{

		if(~PIND & (1<<3))
		{
			uart_puts( "VKL ");
		}
		else
		{
			uart_puts( "OFF ");
		}

		if(~PIND & (1<<2))
		{
			uart_puts( "VKL\r\n");
		}
		else
		{
			uart_puts( "OFF\r\n");
		}
		_delay_ms(300);

		}
	return 0;
}
```

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Для выполнения работ присоедините к отладочной плате плату «кнопки».

Запустите Atmel Studio и выполните File -> New -> Project.

Появится окно создания нового проекта. В нижней его части Вы можете задать название проекта. Имя по умолчанию будет иметь вид «GccApplication1», при этом число в окончание может быть различным. Замените его на имя в формате «Группа_индивидуальный номер случайное число до 100» (например, «EAb-083_06_53»). Свой индивидуальный номер можно узнать у преподавателя.

В появившемся окне выберите «GCC C Executable Project». После этого отобразится окно выбора используемого в проекте микроконтроллера. Найдите в списке «ATmega328» (или «ATmega168»). Кликните мышкой по этой строке и нажмите ОК. Рекомендуется воспользоваться строкой поиска, в правом верхнем углу окна выбора микроконтроллера.

На странице с текстом программы сотрите все и наберите программу из разобранного выше примера.

Выполните Build -> Build Solutions, либо нажмите F7.

Дождитесь окончания компиляции. В нижнем окне «Output» должны появится строчки:

Build succeeded.

========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========

Для того чтобы непосредственно поместить содержимое этого файла в память микроконтроллера, необходимо воспользоваться дополнительной программой – XLoader.

Подключите отладочную плату к компьютеру, используя кабель типа USB-A – USB-B, и запустите программу XLoader.

Окно программы содержит следующие элементы:

- Hex file – здесь нужно указать путь до загружаемого HEX файла;
- Device – здесь необходимо выбрать тип платы и микроконтроллера. Duemilanove/Nano(тип МК – ATmega168 или ATmeg328);
- COM port – здесь необходимо выбрать виртуальный комп порт, через который компьютер общается с МК;
- Baud rate – скорость передачи данных через COM-порт. Зависит от типа МК и выставляется автоматически. Для ATmega168 – 19200, для ATmega328 – 57600.

Полный путь до папки будет иметь вид:

Документы\Atmel Studio\6.2\Имя_проекта\ Имя_проекта\Debug

В папке Debug необходимо найти файл с расширением «.hex». Именно этот файл «заливается» в микроконтроллер. В нём содержится программа для микроконтроллера в виде машинных кодов.

Если все установки правильные, то после нажатия кнопки «Upload» программа свяжется с загрузчиком в МК и, получив от него ответ, загрузит содержимое HEX файла в память МК. Процесс обмена данными можно наблюдать визуально, по мерцанию расположенных на отладочной плате светодиодов RX и TX.

ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ

Напишите программу, реализующую следующий режим работы светодиодов:

1) Переключение светодиодов по команде с ПК. Потребуется плата расширения «трехцветный светодиод».

2) Отправка информации о состояние кнопок только при получении с команды с ПК

3) Команда с ПК включает возможности переключать светодиоды нажатием на кнопки. Другая команда с ПК отключает эту возможность. Потребуется плата расширения «трехцветный светодиод».

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение UART. Интерфейсы на его основе.
2. Прерывания от UART. Их количество и описание.
3. Режимы работы UART. Их количество и описание.
4. Почему невозможна передача строки целиком? Как можно это реализовать?
5. В чем отличие UART от USART?

ЛАБОРАТОРНАЯ РАБОТА №6.

Разработка программы для модуля АЦП микроконтроллера ATmega328.

Цель работы: изучить принципы работы аналогово-цифрового преобразователя, получить первичные навыки написания программ с использованием аналогово-цифрового преобразователя.

КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Аналого-цифровой преобразователь (АЦП) – это устройство для преобразования аналогового сигнала в двоичный код.

АЦП микроконтроллера умеет измерять только напряжение. Чтобы произвести измерение других физических величин, их нужно вначале преобразовать в напряжение. Сигнал всегда измеряется относительно точки называемой опорное напряжение, эта же точка является максимальным значением измеряемого напряжения. В качестве источника опорного напряжения (ИОН) рекомендуется выбирать высокостабильный источник напряжения, иначе все измерения будут постоянно изменяться вслед за изменениями опорного напряжения.

Одной из важнейших характеристик является разрешающая способность, которая влияет на точность измерения. Весь диапазон измерения разбивается на части. Минимум ноль, максимум напряжение ИОН. Для 8 битного АЦП это 2^8=256 значений, для 10 битного 2^10=1024 значения. Таким образом, чем выше разрядность, тем точнее можно измерять сигнал.

Допустим, вы измеряете сигнал от 0 до 5 В. Микроконтроллер используем Atmega328, с 10-битным АЦП. Это значит, что диапазон 5 В будет разделен на 1024 значений. 5В/1024=0,00488В – с таким шагом мы сможем измерять напряжение. Но учтите, что микроконтроллер будет считать величины 0.0055, 0.0075 и даже 0.0095 одинаковыми, округляя их в произвольную сторону.

В качестве источника опорного напряжения можно использовать внутренний источник и внешний. Напряжение внутреннего источника (2,3-2,7 В) не рекомендуется использовать, по причине низкой стабильности. Внешний источник подключается к ножке Avcc или Aref, в зависимости от настроек в управляющей программе.

АНАЛОГОВО-ЦИФРОВОЕ ПРЕОБРАЗОВАНИЕ СИГНАЛА С ПЕРЕДАЧЕЙ ДАННЫХ НА ПЕРСОНАЛЬНЫЙ КОМПЬЮТЕР

Ниже приведен пример программы для работы с АЦП. Преобразованное значение аналогового сигнала передаётся на ПК с помощью UART.

```
#include <avr/io.h>
#define F_CPU 16000000UL //16MHz
#include <util/delay.h>

void ADC_Init()// функция инициализации АЦП
{
	ADCSRA |= (1 << ADEN) // Включаем АЦП
	|(1 << ADPS1)|(1 << ADPS0);	// устанавливаем предделитель преобразователя на 8
	ADMUX |= (0 << REFS1)|(1 << REFS0) //выбираем напряжение питания  как источник опорного напряжения – 5 вольт
	|(0 << MUX0)|(0 << MUX1)|(0 << MUX2)|(0 << MUX3); // снимать сигнал будем с входа PC0 – A0 на Arduino
}

void uart_init( void ) // функция инициализации UART
{
	//настройка скорости обмена
	UBRR0H = 0;
	UBRR0L = 103; // ( 16000000/( 9600*16 ) ) – 1 = 103	число для 16 МГц и скорости 9600 бод
	UCSR0A=0x00;
	UCSR0C = ( 1 << UCSZ01 ) | ( 1 << UCSZ00 ); // настройка на прием 8-битных данных, без контроля четности
	//разрешить прием и передачу данных
	UCSR0B = ( 0 << UCSZ02 );
	UCSR0B = ( 1 << TXEN0 ) | ( 1 <<RXEN0 );
}

unsigned char uart_getc( void ) // функция забирает полученный байт из буфера и записывает его в переменную C
{
	//ждем приема байта
	while( ( UCSR0A & ( 1 << RXC0 ) ) == 0  );
	//считываем принятый байт
	return UDR0;
}
```

```
void uart_putc( char c ) // функция для получения одного символа. При передаче строки получим набор символов
{
	//ждем окончания передачи предыдущего байта
	while( ( UCSR0A & ( 1 << UDRE0 ) ) == 0 );
	UDR0 = c;
}

void uart_puts( char *str )  // функция для упрощения отправки текстовых сообщений
{
	unsigned char c;
	while( ( c = *str++ ) != 0 ) // поочередно отправляем один байт из массива – передаем строку по буквам
	{
		uart_putc( c );
	}
}

int main( void )
{
	unsigned int voltage_Z[3], voltage[4];
	unsigned int u;


	ADC_Init();//запускаем АЦП
	uart_init(); //запускаем UART

	while( 1 )
	{
		// стартуем АЦП
		do
		{
			ADCSRA |= (1 << ADSC);		// Начинаем преобразование
		}
		while ((ADCSRA & (1 << ADIF)) == 0); // делаем это, пока не будет выставлен флаг об окончании преобразования

		u = (ADCL|ADCH << 8); // Считываем полученное значение. В переменной u лежит число от 0 до 1023
```

```
//разбиваем число из переменной u на отдельные числа в соответствие с местом в числе
voltage[0] = u/1000;
voltage[1] = u%1000/100;
voltage[2] = u%100/10;
voltage[3] = u%10;

//отправляем число по цифрам через UART
for(int i=0;i<4;i++)
{
	switch(voltage[i])
	{
		case 0:
		uart_puts("0");
		break;
		case 1:
		uart_puts("1");
		break;
		case 2:
		uart_puts("2");
		break;
		case 3:
		uart_puts("3");
		break;
		case 4:
		uart_puts("4");
		break;
		case 5:
		uart_puts("5");
		break;
		case 6:
		uart_puts("6");
		break;
		case 7:
		uart_puts("7");
		break;
		case 8:
		uart_puts("8");
		break;
		case 9:
		uart_puts("9");
		break;

	}
```

```
}

//восстанавливаем значение напряжения на входе АЦП и отправляем через UART
u = u * 48875 / 10000;
voltage_Z[0] = u/1000;
voltage_Z[1] = u%1000/100;
voltage_Z[2] = u%100/10;
uart_puts("  ");
for(int i=0;i<3;i++)
{
	if(i==1)
	{
		uart_puts(",");
	}
	switch(voltage_Z[i])
	{
		case 0:
		uart_puts("0");
		break;
		case 1:
		uart_puts("1");
		break;
		case 2:
		uart_puts("2");
		break;
		case 3:
		uart_puts("3");
		break;
		case 4:
		uart_puts("4");
		break;
		case 5:
		uart_puts("5");
		break;
		case 6:
		uart_puts("6");
		break;
		case 7:
		uart_puts("7");
		break;
		case 8:
		uart_puts("8");
		break;
```

```
                        case 9:
                        uart_puts("9");
                        break;

                    }
                }
                //задержка обновления АЦП и перенос курсора на новую
строку
                _delay_ms(200);
                uart_puts( "\r\n");
            }
            return 0;
        }
```

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

Для выполнения работ присоедините к отладочной плате блок с переменным резистором.

Запустите Atmel Studio и выполните File -> New -> Project.

Появится окно создания нового проекта. В нижней его части Вы можете задать название проекта. Имя по умолчанию будет иметь вид «GccApplication1», при этом число в окончание может быть различным. Замените его на имя в формате «Группа_индивидуальный номер случайное число до 100» (например, «EAb-083_06_53»). Свой индивидуальный номер можно узнать у преподавателя.

В появившемся окне выберите «GCC C Executable Project». После этого отобразится окно выбора используемого в проекте микроконтроллера. Найдите в списке «ATmega328». Кликните мышкой по этой строке и нажмите ОК. Рекомендуется воспользоваться строкой поиска, в правом верхнем углу окна выбора микроконтроллера.

На странице с текстом программы сотрите все и наберите программу из разобранного выше примера.

Выполните Build -> Build Solutions, либо нажмите F7.

Дождитесь окончания компиляции. В нижнем окне «Output» должны появиться строчки:

Build succeeded.

========== Build: 1 succeeded or up-to-date, 0 failed, 0 skipped ==========

Для того чтобы непосредственно поместить содержимое этого файла в память микроконтроллера, необходимо воспользоваться дополнительной программой – XLoader.

Подключите отладочную плату к компьютеру, используя кабель типа USB-A – USB-B, и запустите программу XLoader.

Окно программы содержит следующие элементы:

- Hex file – здесь нужно указать путь до загружаемого HEX файла;
- Device – здесь необходимо выбрать тип платы и микроконтроллера. Duemilanove/Nano (тип МК – ATmeg328);
- COM port – здесь необходимо выбрать виртуальный комп порт, через который компьютер общается с МК;
- Baud rate – скорость передачи данных через COM-порт. Зависит от типа МК и выставляется автоматически. Для ATmega328 – 57600.

Полный путь до папки будет иметь вид:

Документы\Atmel Studio\6.2\Имя_проекта\ Имя_проекта\Debug

В папке Debug необходимо найти файл с расширением «.hex». Именно этот файл «заливается» в микроконтроллер. В нём содержится программа для микроконтроллера в виде машинных кодов.

Если все установке правильные, то после нажатия кнопки «Upload» программа свяжется с загрузчиком в МК и, получив от него ответ, загрузит содержимое HEX файла в память МК. Процесс обмена данными можно наблюдать визуально, по мерцанию расположенных на отладочной плате светодиодов RX и TX.

ИНДИВИДУАЛЬНЫЕ ЗАДАНИЯ

Напишите программу, реализующую следующий режим работы светодиодов:

1) Включение различных светодиодов при различном уровне напряжения на входе АЦП. Потребуется плата расширения «трехцветный светодиод».

2) Включение светодиода на 13 выводе отладочной платы при уменьшении входного напряжения ниже задаваемого порога.

3) Отправка значения на входе АЦП через UART на ПК только при получении команды с ПК.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Назначение АЦП.
2. Возможно ли прерывание от АЦП?
3. Как можно увеличить точность АЦП для измерения сигнала с амплитудой менее 5 вольт?
4. Как можно увеличить диапазон измеряемых напряжений?

СПИСОК ЛИТЕРАТУРЫ

Основная литература

1. Водовозов, А. М. Микроконтроллеры для систем автоматики: учебное пособие [Электронный ресурс]. – Москва : Вологда: Инфра-Инженерия, 2016. – 164 с. – Режим доступа: http://biblioclub.ru/index.php?page=book_red&id=444183. – Загл. с экрана. (28.01.2017)

2. Нестеровский, А. В. Микроконтроллеры и интерфейсы связи: учебное пособие для студентов вузов, обучающихся по специальности 140604 "Электропривод и автоматика пром. установок и технолог. комплексов" / А. В. Нестеровский, А. П. Носков; ГОУ ВПО "Кузбас. гос. техн. ун-т". – Кемерово, 2010. – 115 с.

Дополнительная литература

1. Шегал, А. А. Применение программного комплекса Multisim для проектирования устройств на микроконтроллерах: лабораторный практикум [Электронный ресурс]. – Екатеринбург: Изд-во Уральского университета, 2014. – 116 с. – Режим доступа: http://biblioclub.ru/index.php?page=book_red&id=276471. – Загл. с экрана. (20.12.2016)

2. Васильев, А. Е. Микроконтроллеры. Разработка встраиваемых приложений [Текст]: учеб. Пособие для студентов вузов, обучающихся по специальности 220201 "Управление и информатика в техн. системах" / А. Е. Васильев. – Санкт-Петербург: БХВ-Петербург, 2008. – 304 с. CD-ROM

3. Схемотехника электронных систем: микропроцессоры и микроконтроллеры [Текст] / В. И. Бойко [и др.]. – Санкт-Петербург: БХВ-Петербург, 2004. – 464 с.

ПРИЛОЖЕНИЕ 1

Таймер/счетчик T0 (8 бит)

Характеристики таймера/счетчика T0 (8 бит):
- Два независимых выхода по совпадению;
- Таймер сброса при совпадении;
- Изменяемый период ШИМ сигнала;
- Фазовый корректор ШИМ сигнала;
- Тактовый генератор;
- Три независимых источника прерывания.

Регистры таймера/счетчика T0:
- TCNT0 – счетный регистр таймера/счетчика T0;
- OCR0A – регистр сравнения A;
- OCR0B – регистр сравнения B;
- TIMSK0 – регистр маски прерываний для таймера/счетчика T0;
- TIFR0 – регистр флагов прерываний для таймера/счетчика T0;
- TCCR0A – регистр управления A;
- TCCR0B – регистр управления B.

Источником тактового сигнала для таймера/счетчика T0 может быть как тактовый сигнал, используемый для всего микроконтроллера с использованием предделителя, так и сигнал, поступающий на вход T0 (6 ножка). Если не выбрано ни одного источника тактового сигнала, то таймер/счетчик останавливается.

Режим работы таймера/счетчика T0 устанавливается регистрами TCCR0A и TCCR0B.

Регистр TCCR0A:

7	6	5	4	3	2	1	0
COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00

Биты COM0A1 (7) и COM0A0 (6) влияют на то, какой сигнал появится на выводе OC0A (12 ножка) при совпадении с A (совпадение значения счетного регистра TCNT0 со значением регистра сравнения OCR0A):

1. Обычный режим

- 00 – вывод OC0A не функционирует
- 01 – изменение состояния вывода OC0A на противоположное при совпадении с A
- 10 – сброс вывода OC0A в 0 при совпадении с A
- 11 – установка вывода OC0A в 1 при совпадении с A

2. Режим ШИМ

- 00 – вывод OC0A не функционирует

- 01 – если бит WGM02 регистра TCCR0B установлен в 0, вывод OC0A не функционирует

- 01 – если бит WGM02 регистра TCCR0B установлен в 0, изменение состояния вывода OC0A на противоположное при совпадении с A

- 10 – сброс вывода OC0A в 0 при совпадении с A, установка вывода OC0A в 1 если регистр TCNT0 принимает значение 0x00 (неинверсный режим)

- 11 – установка вывода OC0A в 1 при совпадении с A, установка вывода OC0A в 0 если регистр TCNT0 принимает значение 0x00 (инверсный режим)

3. Режим коррекции фазы ШИМ

- 00 – вывод OC0A не функционирует

- 01 – если бит WGM02 регистра TCCR0B установлен в 0, вывод OC0A не функционирует

- 01 – если бит WGM02 регистра TCCR0B установлен в 0, изменение состояния вывода OC0A на противоположное

- 10 – сброс вывода OC0A в 0 при совпадении с A во время увеличения значения счетчика, установка вывода OC0A в 1 при совпадении с A во время уменьшения значения счетчика

- 11 – установка вывода OC0A в 1 при совпадении с A во время увеличения значения счетчика, сброс вывода OC0A в 0 при совпадении с A во время уменьшения значения счетчика

Биты COM0B1 (5) и COM0B0 (4) влияют на то, какой сигнал появится на выводе OC0B (12 ножка) при совпадении с B (совпадение значения счетного регистра TCNT0 со значением регистра сравнения OCR0B):

1. Обычный режим

- 00 – вывод OC0B не функционирует

- 01 – изменение состояния вывода OC0B на противоположное при совпадении с B

- 10 – сброс вывода OC0B в 0 при совпадении с B

- 11 – установка вывода OC0B в 1 при совпадении с B

2. Режим ШИМ

- 00 – вывод OC0B не функционирует

- 01 – резерв

- 10 – сброс вывода OC0B в 0 при совпадении с B, установка вывода OC0B в 1 если регистр TCNT0 принимает значение 0x00 (неинверсный режим)

- 11 – установка вывода OC0B в 1 при совпадении с B, установка вывода OC0B в 0 если регистр TCNT0 принимает значение 0x00 (инверсный режим)

3. Режим коррекции фазы ШИМ

- 00 – вывод OC0B не функционирует

- 01 – резерв

- 10 – сброс вывода OC0B в 0 при совпадении с B во время увеличения значения счетчика, установка вывода OC0B в 1 при совпадении с B во время уменьшения значения счетчика

- 11 – установка вывода OC0B в 1 при совпадении с B во время увеличения значения счетчика, сброс вывода OC0B в 0 при совпадении с B во время уменьшения значения счетчика

Биты WGM01 (1), WGM00 (0) регистра TCCR0A устанавливают режим работы таймера/счетчика T0:

- 00 – обычный режим
- 01 – режим коррекции фазы ШИМ
- 10 – режим подсчета импульсов (сброс при совпадении)
- 11 – режим ШИМ

Регистр TCCR0B:

7	6	5	4	3	2	1	0
FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00

Биты FOC0A (7) и FOC0B (6) регистра TCCR0B принудительно устанавливают значение на выводах OC0A и OC0B.

Биты CS02 (2), CS01 (1), CS00 (0) регистра TCCR0B устанавливают режим тактирования и предделителя тактовой частоты таймера/счетчика T0:

- 000 – таймер/счетчик T0 остановлен
- 001 – тактовый генератор CLK
- 010 – CLK/8
- 011 – CLK/64
- 100 – CLK/256
- 101 – CLK/1024
- 110 – внешний источник на выводе T0 (6 ножка) по спаду сигнала
- 111 – внешний источник на выводе T0 (6 ножка) по возрастанию сигнала

Управление прерываниями от таймера осуществляется в регистре TIMSK0.

Регистр TIMSK0:

7	6	5	4	3	2	1	0
-	-	-	-	-	OCIE0B	OCIE0A	TOIE0

Биты OCIE0B (2) и OCIE0A (1) разрешают прерывания при совпадении с A и B, а бит TOIE0 (0) разрешает прерывание по переполнению при установке 1. Если в эти биты записать 0, прерывания от таймера/счетчика будут запрещены.

Также есть регистр флагов прерываний TIFR0, который показывает, какое прерывание поступило от таймера/счетчика T0.

Регистр TIFR0:

7	6	5	4	3	2	1	0
-	-	-	-	-	OCF0B	OCF0A	TOV0

Биты OCF0B (2), OCF0A (1) и TOV0 (0) устанавливаются в 1 в зависимости от того, какое прерывание поступило – совпадение с A, B или переполнение.

Таймер/счетчик T1 (16 бит)

Характеристики таймера/счетчика T1 (16 бит):
- Два независимых выхода по совпадению;
- Таймер сброса при совпадении;
- Один вход захвата;
- Блок шумоподавления входа захвата;
- Изменяемый период ШИМ сигнала;
- Фазовый корректор ШИМ сигнала;
- Изменяемый период ШИМ сигнала;
- Тактовый генератор;
- Три независимых источника прерывания.

Регистры таймера/счетчика T1:
- TCNT1 – счетный регистр таймера/счетчика T1 (16 бит)
- OCR1A – регистр сравнения A (16 бит)
- OCR1B – регистр сравнения B (16 бит)
- TIMSK1 – регистр маски прерываний для таймера/счетчика T1
- TIFR1 – регистр флагов прерываний для таймера/счетчика T1
- TCCR1A – регистр управления A
- TCCR1B – регистр управления B
- TCCR1C – регистр управления C
- ICR1 – регистр захвата (16 бит)

Источником тактового сигнала для таймера/счетчика T1 может быть как тактовый сигнал, используемый для всего микроконтроллера с использованием предделителя, так и сигнал, поступающий на вход T1 (11 ножка). Если не выбрано ни одного источника тактового сигнала, то таймер/счетчик останавливается.

Регистр TCCR1A:

7	6	5	4	3	2	1	0
COM1A1	COM1A0	COM1B1	COM1B0	-	-	WGM11	WGM10

Режим работы таймера/счетчика T1 устанавливается регистрами TCCR1A и TCCR1B

Биты COM1A1 (7) и COM1A0 (6) влияют на то, какой сигнал появится на выводе OC1A (15 ножка) при совпадении с A (совпадение значения счетного регистра TCNT1 со значением регистра сравнения OCR1A):

1. Обычный режим

- 00 – вывод OC1A не функционирует
- 01 – изменение состояния вывода OC1A на противоположное при совпадении с A
- 10 – сброс вывода OC1A в 0 при совпадении с A
- 11 – установка вывода OC1A в 1 при совпадении с A

2. Режим ШИМ

- 00 – вывод OC1A не функционирует
- 01 – если биты WGM13 – WGM10 установлены в (0000 – 1101), вывод OC1A не функционирует
- 01 – если битыWGM13 – WGM10 установлены в 1110 или 1111, изменение состояния вывода OC0A на противоположное при совпадении с A
- 10 – сброс вывода OC1A в 0 при совпадении с A, установка вывода OC1A в 1, если регистр TCNT1 принимает значение 0x00 (неинверсный режим)
- 11 – установка вывода OC1A в 1 при совпадении с A, установка вывода OC1A в 0, если регистр TCNT1 принимает значение 0x00 (инверсный режим)

3. Режим коррекции фазы ШИМ

- 00 – вывод OC1A не функционирует
- 01 – если биты WGM13 – WGM10 установлены в (0000 – 1100, 1010, 1100 – 1111), вывод OC1A не функционирует
- 01 – если битыWGM13 – WGM10 установлены в 1101 или 1011, изменение состояния вывода OC1A на противоположное при совпадении с A
- 10 – сброс вывода OC1A в 0 при совпадении с A во время увеличения значения счетчика, установка вывода OC1A в 1 при совпадении с A во время уменьшения значения счетчика
- 11 – установка вывода OC1A в 1 при совпадении с A во время увеличения значения счетчика, сброс вывода OC1A в 0 при совпадении с A во время уменьшения значения счетчика

Биты COM1B1 (5) и COM1B0 (4) влияют на то, какой сигнал появится на выводе OC0B (12 ножка) при совпадении с B (совпадение значения счетного регистра TCNT1 со значением регистра сравнения OCR1B):

1. Обычный режим

- 00 – вывод OC1B не функционирует

- 01 – изменение состояния вывода OC1B на противоположное при совпадении с B
- 10 – сброс вывода OC1B в 0 при совпадении с B
- 11 – установка вывода OC1B в 1 при совпадении с B

2. Режим ШИМ
- 00 – вывод OC1B не функционирует
- 01 – вывод OC1B не функционирует
- 10 – сброс вывода OC1B в 0 при совпадении с B, установка вывода OC1B в 1, если регистр TCNT1 принимает значение 0x00 (неинверсный режим)
- 11 – установка вывода OC1B в 1 при совпадении с B, установка вывода OC1B в 0, если регистр TCNT1 принимает значение 0x00 (инверсный режим)

3. Режим коррекции фазы ШИМ
- 00 – вывод OC1B не функционирует
- 01 – вывод OC1B не функционирует
- 10 – сброс вывода OC1B в 0 при совпадении с B во время увеличения значения счетчика, установка вывода OC1B в 1 при совпадении с B во время уменьшения значения счетчика
- 11 – установка вывода OC1B в 1 при совпадении с B во время увеличения значения счетчика, сброс вывода OC1B в 0 при совпадении с B во время уменьшения значения счетчика

Регистр TCCR1B:

7	6	5	4	3	2	1	0
ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10

Биты WGM13 (4), WGM12 (3) регистра TCCR1B и биты WGM11 (1), WGM10 (0) регистра TCCR1A устанавливают режим работы таймера/счетчика T1:
- 0000 – обычный режим
- 0001 – коррекция фазы PWM, 8-бит
- 0010 – коррекция фазы PWM, 9-бит
- 0011 – коррекция фазы PWM, 10-бит
- 0100 – режим счета импульсов (OCR1A) (сброс при совпадении)
- 0101 – PWM, 8-бит
- 0110 – PWM, 9-бит
- 0111 – PWM, 10-бит
- 1000 – коррекция фазы и частоты PWM (ICR1)
- 1001 – коррекция фазы и частоты PWM (OCR1A)
- 1010 – коррекция фазы PWM (ICR1)

- 1011 – коррекция фазы и частоты PWM (OCR1A)
- 1100 – режим счета импульсов (ICR1) (сброс при совпадении)
- 1101 – резерв
- 1111 – PWM (OCR1A)

Бит ICNC1 (7) регистра TCCR1B управляет схемой подавления помех блока захвата (0 – выключена / 1 – включена).

Бит ICES1 (6) регистра TCCR1B выбирает активный фронт регистра захвата (0 – по спадающему фронту сигнала / 1 – по нарастающему фронту сигнала).

Биты CS12 (2), CS11 (1), CS10 (0) регистра TCCR1B устанавливают режим тактирования и предделителя тактовой частоты таймера/счетчика T1:

- 000 – таймер/счетчик T1 остановлен
- 001 – тактовый генератор CLK
- 010 – CLK/8
- 011 – CLK/64
- 100 – CLK/256
- 101 – CLK/1024
- 110 – внешний источник на выводе T1 (11 ножка) по спаду сигнала
- 111 – внешний источник на выводе T1 (11 ножка) по возрастанию сигнала

Регистр TCCR1C:

7	6	5	4	3	2	1	0
FOC1A	FOC1B	-	-	-	-	-	-

Биты FOC1A (7) и FOC1B (6) регистра TCCR1C принудительно устанавливают значение на выводах OC1A и OC1B.

Регистр TIMSK1:

7	6	5	4	3	2	1	0
-	-	ICIE1	-	-	OCIE1B	OCIE1A	TOIE1

Управление прерываниями от таймера осуществляется в регистре TIMSK1.

Бит ICIE1 (5) разрешает прерывание по захвату, биты OCIE1B (2) и OCIE1A (1) разрешают прерывания при совпадении с A и B, бит TOIE1 (0) разрешает прерывание по переполнению при установке 1. Если в эти биты записать 0, прерывания от таймера/счетчика будут запрещены.

Также есть регистр флагов прерываний TIFR1, который показывает, какое прерывание поступило от таймера/счетчика T0.

Регистр TIFR1:

7	6	5	4	3	2	1	0
-	-	ICF1	-	-	OCF1B	OCF1A	TOV1

Биты ICF1 (5), OCF1B (2), OCF1A (1) и TOV1 (0) устанавливаются в 1 в зависимости от того, какое прерывание поступило – захват, совпадение с A, B или переполнение.

Таймер/счетчик T2 (8 бит)

Характеристики таймера/счетчика T2 (8 бит):

- Два независимых выхода по совпадению;
- Таймер сброса при совпадении;
- Изменяемый период ШИМ сигнала;
- Фазовый корректор ШИМ сигнала;
- Тактовый генератор;
- Возможность работы от независимого внешнего часового тактового генератора 32 кГц;
- Три независимых источника прерывания;
- Делитель частоты 10-бит;
- Асинхронный режим.

Регистры таймера/счетчика T2:

- TCNT2 – счетный регистр таймера/счетчика T2
- OCR2A – регистр сравнения A
- OCR2B – регистр сравнения B
- TIMSK2 – регистр маски прерываний для таймера/счетчика T2
- TIFR2 – регистр флагов прерываний для таймера/счетчика T2
- TCCR2A – регистр управления A
- TCCR2B – регистр управления B
- ASSR – регистр асинхронного режима
- GTCCR – главный регистр всех таймеров/счетчиков

Источником тактового сигнала для таймера/счетчика T2 может быть тактовый сигнал, используемый для всего микроконтроллера с использованием предделителя. Если не выбран коэффициент деления, то таймер/счетчик останавливается.

Режим работы таймера/счетчика T2 устанавливается регистрами TCCR2A и TCCR2B аналогично таймеру/счетчику T0.

Регистр TCCR2B:

7	6	5	4	3	2	1	0
FOC2A	FOC2B	-	-	WGM22	CS22	CS21	CS20

Разница лишь в битах CS22 (2), CS21 (1), CS20 (0) регистра TCCR2B, которые устанавливают режим тактирования.

- 000 – таймер остановлен
- 001 – CLK
- 010 – CLK/8
- 011 – CLK/32
- 100 – CLK/64
- 101 – CLK/128
- 110 – CLK/256
- 111 – CLK/1024

Также у таймера/счетчика есть асинхронный режим работы.

Регистр ASSR:

7	6	5	4	3	2	1	0
-	EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB

Бит EXCLK (6) регистра ASSR разрешает использование внешнего тактового сигнала от кварцевого резонатора 32 кГц при записи в него 1.

Бит AS2 (5) регистра ASSR управляет способом тактирования (1 – от внешнего резонатора 32 кГц, подключенного к TOSC1 (9 ножка) / 0 – от внутреннего генератора CLK)/

Бит TCN2UB (4) регистра ASSR показывает, доступен ли для записи регистр TCNT2 (1 – недоступен / 0 – доступен).

Бит OCR2AUB (3) регистра ASSR показывает, доступен ли для записи регистр OCR2A (1 – недоступен / 0 – доступен).

Бит OCR2BUB (2) регистра ASSR показывает, доступен ли для записи регистр OCR2B (1 – недоступен / 0 – доступен).

Бит TCR2AUB (1) регистра ASSR показывает, доступен ли для записи регистр TCCR2A (1 – недоступен / 0 – доступен).

Бит TCR2BUB (0) регистра ASSR показывает, доступен ли для записи регистр TCCR2B (1 – недоступен / 0 – доступен).

Регистр GTCCR:

7	6	5	4	3	2	1	0
TSM	-	-	-	-	-	PSRASY	PSRSYNC

Бит PSRASY (1) регистра GTCCR сбрасывает предделитель таймера/счетчика T2, если установить в 1, после этого бит сбрасывается в 0 автоматически.

Бит PSRASYNC (0) регистра GTCCR сбрасывает предделитель таймера/счетчика T0 и T1, если установить в 1, после этого бит сбрасывается в 0 автоматически.

Бит TSM (0) регистра GTCCR запрещает автоматический сброс битов PSRASY и PSRASYNC регистра GTCCR.