

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное
учреждение высшего образования
«СЕВАСТОПОЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

**ИССЛЕДОВАНИЕ УСТРОЙСТВ И МЕТОДОВ
ПРОГРАММИРОВАНИЯ ВСТРАИВАЕМЫХ
МИКРОПРОЦЕССОРНЫХ СИСТЕМ**

Методические указания
к выполнению лабораторных работ
по дисциплине «Встроенные микропроцессорные системы»
для студентов, обучающихся по направлению 09.03.02
«информационные системы и технологии»
по учебному плану подготовки бакалавров
дневной и заочной форм обучения

Севастополь
СевГУ
2020

УДК 004.732
И88

Р е ц е н з е н т:
В.Г. Козырев - канд. техн. наук, доцент,
доцент кафедры ИиУТС

Составитель: В.С. Чернега

И88 Исследование устройств и методов программирования встраиваемых микропроцессорных систем: методические указания к лабораторным занятиям по дисциплине «Встроенные микропроцессорные системы» / Сост. В.С. Чернега ; Министерство науки и высшего образования РФ, Севастопольский государственный университет. – Севастополь: СевГУ, 2020. – 86 с. – Текст : электронный.

Методические указания предназначены для проведения лабораторных работ по дисциплине «Встроенные микропроцессорные системы». Целью методических указаний является помощь студентам в изучении архитектуры микроконтроллеров, используемых в информационных устройствах и системах, способов программирования таких устройств на языке ассемблера и языке Си. Излагаются теоретические и практические сведения необходимые для выполнения лабораторной работы, требования к содержанию отчета.

УДК 004.732

Методические указания рассмотрены и утверждены на методическом семинаре и заседании кафедры «Информационные системы», протокол № 1 от 30 августа 2019 г.

СОДЕРЖАНИЕ

1. Лабораторная работа 1. Изучение интегрированной среды разработки программного обеспечения и исследование функционирования микроконтроллеров AVR	4
2. Лабораторная работа 2. Исследование системы команд микроконтроллеров AVR	19
3. Лабораторная работа 3. Исследование функционирования микроконтроллеров AVR с использованием инструментальных средств моделирования	29
4. Лабораторная работа 4. Исследование процедур программирования восьмибитовых таймеров/счетчиков и обслуживания прерываний	44
5. Лабораторная работа 5. Исследование способов программирования АЦП и вывода информации в микроконтроллерах AVR	64
6. Лабораторная работа 6. Исследование методов программирования и настройки последовательного интерфейса USART микроконтроллеров AVR	75
Библиографический список	86

ЛАБОРАТОРНАЯ РАБОТА №1

ИЗУЧЕНИЕ ИНТЕГРИРОВАННОЙ СРЕДЫ РАЗРАБОТКИ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И ИССЛЕДОВАНИЕ ФУНКЦИОНИРОВАНИЯ МИКРОКОНТРОЛЛЕРОВ AVR

1 Цель работы

Ознакомиться с назначением и органами управления среды разработки, исследовать процессы содержимого регистров и портов микроконтроллера в процессе отладки программы. Приобрести практические навыки программирования и отладки программ на языке Ассемблера и Си.

2 Краткие теоретические сведения

2.1 Архитектура микроконтроллеров AVR семейства MEGA

Микроконтроллеры семейства Mega представляют собой однокристалльную ЭВМ, состоящую из процессорного ядра, памяти, периферийных устройств и портов ввода/вывода. Эти микроконтроллеры предназначены для использования в мобильных телефонах, в контроллерах различного периферийного оборудования (такого как принтеры, сканеры, современные дисковые накопители, приводы CD-ROM/DVD-ROM и т. п.), в сложной офисной технике и т. д.

Процессорное ядро AVR-микроконтроллеров (рисунок 2.1) выполнено по Гарвардской архитектуре на основе сокращенного набора команд (RISC команд). Эта архитектура характеризуется раздельной памятью программ и данных, каждая из которых имеет собственные шины доступа. Такая организация позволяет одновременно работать как с памятью программ, так и с памятью данных.

Арифметико-логическое устройство (АЛУ), выполняющее все вычисления, подключено непосредственно к 32 рабочим регистрам, объединенным в регистровый файл. Благодаря такому подключению АЛУ может выполнять одну операцию (чтение содержимого регистров, выполнение операции и запись результата обратно в регистровый файл) за такт. Кроме того, практически каждая из команд (за исключением команд, у которых одним из операндов является 16-битный адрес) занимает одну ячейку памяти программ.

Разделение информационных шин позволяет использовать для каждого типа памяти шины различной разрядности, причем способы адресации и доступа к каждому типу памяти также различаются. В сочетании с двухуровневым конвейером команд (одна команда выполняется, другая параллельно выбирается и декодируется) такая архитектура позволяет достичь производительности в 1 MIPS на 1 МГц тактовой частоты.

AVR-микроконтроллеры изготавливаются по малопотребляющей КМОП-технологии и имеют полностью статическую архитектуру, не позволяющую подключение дополнительных внешних устройств.

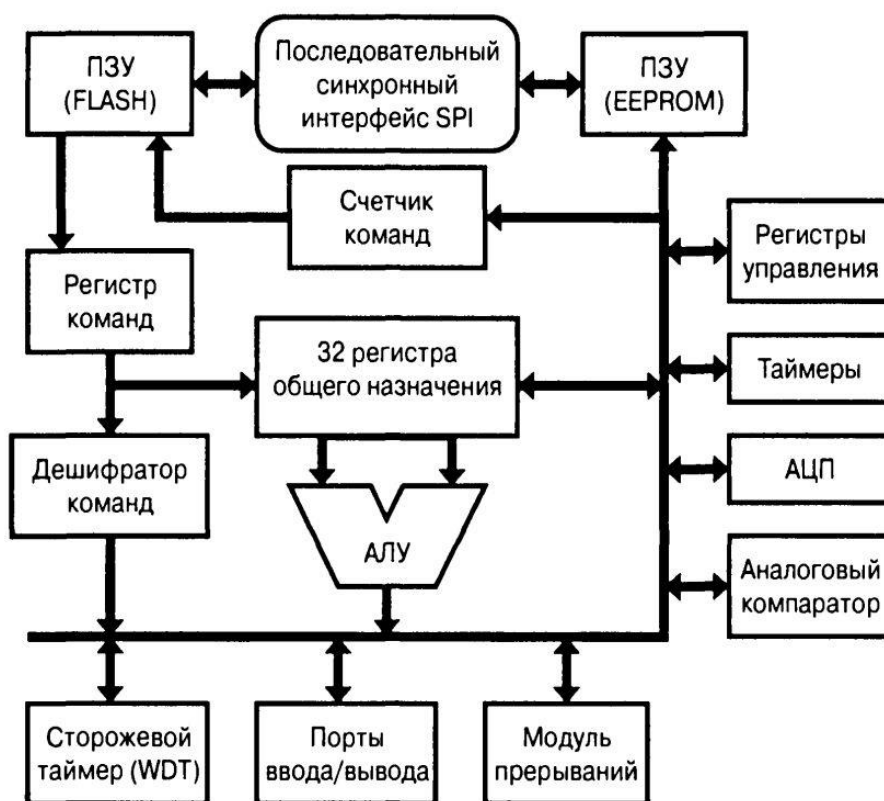


Рисунок 2.1– Архитектура процессорного ядра с периферийными устройствами AVR-микроконтроллера

Основные параметры и характеристики микроконтроллеров AVR семейства Mega:

- FLASH-память программ объемом от 8 до 256 Кбайт (число циклов стирания/записи не менее 10 000);
- оперативная память (статическое ОЗУ) объемом от 512 байт до 8 Кбайт;
- память данных на основе ЭСППЗУ (EEPROM) объемом от 256 байт до 4 Кбайт (число циклов стирания/записи не менее 100 000);
- возможность защиты от чтения и модификации памяти программ и данных;

- возможность программирования непосредственно в системе через последовательные интерфейсы SPI и JTAG;
- возможность внутрисхемной отладки в соответствии со стандартом IEEE 1149.1 (JTAG), а также наличие собственного однопроводного интерфейса внутрисхемной отладки (debugWire1);
- разнообразные способы синхронизации: встроенный RC-генератор с внутренней или внешней времязадающей RC-цепочкой, встроенный генератор с внешним кварцевым или пьезокерамическим резонатором, внешний сигнал синхронизации;
 - наличие нескольких режимов пониженного энергопотребления;
 - наличие сторожевого таймера;
 - наличие детектора пониженного напряжения питания;
 - возможность программного снижения частоты тактового генератора.
 - векторная система прерываний, поддержка очереди прерываний;
 - большое число источников прерываний (до 45 внутренних и до 32 внешних);
 - наличие аппаратного умножителя.

Структурная схема микроконтроллера ATmega16 приведена в приложении Б. Микроконтроллеры AVR семейства Mega имеют от 23 до 86 линий ввода/вывода, которые объединяются в 8-разрядные порты ввода/вывода, причем отдельные линии могут быть запрограммированы как входные или как выходные независимо друг от друга. Каждая линия имеет входной буфер с триггером Шмита, а индивидуально отключаемый внутренний (подтягивающий) резистор сопротивлением 20...50 кОм.

Кроме того, имеется большой набор периферийных устройств:

- один или два 8-битных таймера/счетчика, во всех моделях с двумя 8-битными таймерами/счетчиками один из них может работать в качестве часов реального времени (в асинхронном режиме);
- от одного до четырех 16-битных таймеров/счетчиков;
- одно- и двухканальные генераторы 8-битного ШИМ-сигнала (один из режимов работы 8-битных таймеров/счетчиков);
- двух- и трехканальные генераторы ШИМ-сигнала регулируемой разрядности (один из режимов работы 16-битных таймеров/счетчиков), разрешение формируемого сигнала может составлять от 1 до 16 бит;
- аналоговый компаратор;
- многоканальный 10-битный АЦП последовательного приближения, имеющий как несимметричные, так и дифференциальные входы;
- последовательный синхронный интерфейс SPI;
- последовательный двухпроводный интерфейс TWI (аналог интерфейса I²C);
- от одного до четырех полнодуплексных универсальных синхронных/асинхронных приемо-передатчиков (USART), которые в ряде моделей могут использоваться в качестве ведущего устройства шины SPI;

- универсальный последовательный интерфейс USI, который может использоваться в качестве интерфейса SPI или I²C, а также полудуплексного UART или 4/12-битного счетчика.

Обобщенная карта памяти микроконтроллеров AVR семейства Mega приведена на рисунке 2.2. Поскольку микроконтроллеры AVR имеют 16-битную систему команд, объем памяти программ на рисунке указан не в байтах, а в 16-битных словах. Символ «\$» перед числом означает, что это число записано в шестнадцатеричной системе счисления. Для микроконтроллера ATmega8535 объем памяти программ составляет 8 Кбайт (4К*16), ее верхняя граница F_END = \$FFF, объем ОЗУ данных – 512 байт, верхняя граница S_END = \$25F, объем ПЗУ данных – также 512 байт, верхняя граница E_END = \$1FF.

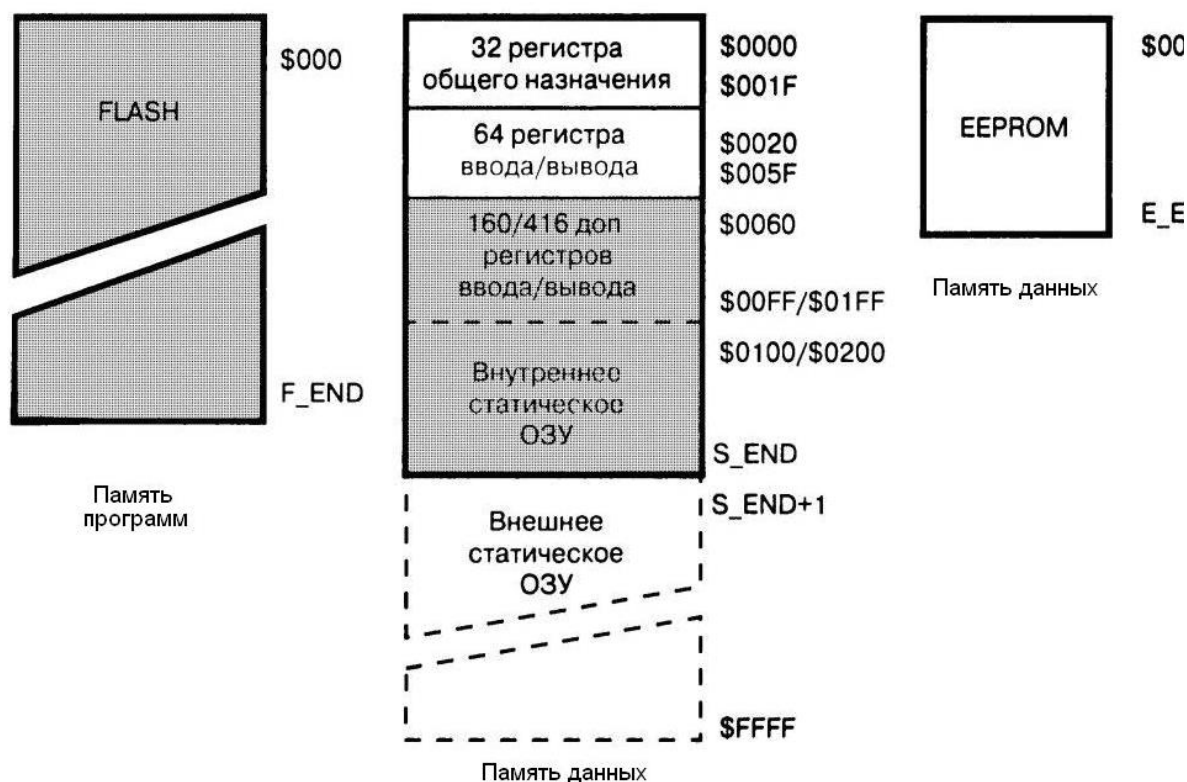


Рисунок 2.2 – Карта памяти AVR-микроконтроллеров семейства Mega

Память программ (объемом от 8 до 256 Кбайт) предназначена для хранения команд, управляющих работой микроконтроллера, а также часто используется для хранения таблиц констант, не меняющихся во время работы программы. Для пересылки байта из памяти программ в память данных существует специальная команда — LPM. При использовании команды LPM адрес, по которому производится чтение, определяется содержимым индексного регистра Z. При этом старшие 15 битов содержимого регистра будут определять адрес слова (0 ... 32 К), младший бит будет определять, какой из байтов будет прочитан: 0 — младший байт, 1 — старший байт.

В подавляющем большинстве моделей микроконтроллеров семейства Mega память программ логически разделена на две неравные части: область

прикладной программы и область загрузчика. В последней может располагаться специальная программа (загрузчик), позволяющая микроконтроллеру самостоятельно управлять загрузкой и выгрузкой прикладных программ. Если же возможность самопрограммирования микроконтроллера не используется, прикладная программа может располагаться и в области загрузчика.

По адресу \$0000 памяти программ находится вектор сброса (команда перехода на начало инициализации). После сброса микроконтроллера выполнение программы начинается с этого адреса (по этому адресу должна размещаться команда перехода к инициализационной части программы). Начиная с адреса \$001 (модели с памятью программ 8 Кбайт и меньше) или \$0002 (остальные модели) памяти программ располагается таблица векторов прерываний. Размер этой области зависит от модели микроконтроллера.

При возникновении прерывания, после сохранения в стеке текущего значения счетчика команд, происходит выполнение команды, расположенной по адресу соответствующего вектора. Поэтому по данным адресам располагаются команды перехода к подпрограммам обработки прерываний. В моделях с памятью программ небольшого объема (8 Кбайт и менее) в таблице векторов прерываний используются команды относительного перехода (RJMP), а в остальных моделях — команды абсолютного перехода (JMP). Если в программе прерывания не используются, то основная программа может начинаться непосредственно с адреса \$0000 (при отсутствии вектора сброса).

Память данных микроконтроллеров семейства Mega разделена на три части: регистровая память, оперативная память (статическое ОЗУ) и энергонезависимое ЭСППЗУ (EEPROM).

Регистровая память включает 32 регистра общего назначения (РОН), и 64 служебных регистра ввода/вывода (РВВ). В сложных моделях с развитой периферией имеется также область дополнительных (extended) регистров ввода/вывода (ДРВВ). Под РВВ в памяти микроконтроллера отводится 64 байта, а под ДРВВ — 160 или 416 байт (в зависимости от модели).

В отличие от процессоров с аккумулятором, в процессорном ядре AVR все 32 РОН непосредственно доступны АЛУ. Благодаря этому любой РОН может использоваться практически во всех командах и как операнд-источник, и как операнд-приемник. Последние 6 регистров общего назначения (R26...R31) могут также объединяться в три 16-битных регистра X, Y и Z, используемых в качестве указателей при косвенной адресации памяти данных.

В областях регистров ввода/вывода располагаются различные служебные регистры: регистр состояния (флагов), а также регистры управления периферийными устройствами, входящими в состав микроконтроллера. Общее их количество зависит от конкретной модели микроконтроллера.

К регистрам ввода/вывода, расположенным в основном пространстве ввода/вывода, можно напрямую обратиться с помощью команд IN и OUT, выполняющих пересылку данных между одним из 32-х РОН и пространством ввода/вывода. В системе команд имеется также четыре команды побитового доступа, использующие в качестве операндов регистры ввода/вывода: команды

установки/сброса отдельного бита (SBI и CBI) и команды проверки состояния отдельного бита (SBIS и SBIC). Но эти команды могут обращаться только к 1-й половине основного пространства ввода/вывода (адреса \$00...\$1F).

Среди служебных регистров есть один регистр, используемый наиболее часто в процессе выполнения программ. Это регистр состояния SREG (Status REGister). Он располагается по адресу \$3F (\$5F) и содержит набор флагов, показывающих текущее состояние микроконтроллера. Большинство флагов автоматически устанавливаются в 1 или сбрасываются в 0 при наступлении определенных событий (в соответствии с результатом выполнения команд). Все биты этого регистра доступны как для чтения, так и для записи; после сброса микроконтроллера все биты регистра сбрасываются в 0. Формат этого регистра следующий:

Регистр состояния – SREG

Бит	7	6	5	4	3	2	1	0
SREG	I	T	H	S	V	N	Z	C
\$3F(\$5F)								
Чтение/Запись	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Исходное значение	0	0	0	0	0	0	0	0

- **Бит 7 – I: Общее разрешение прерываний.** Для разрешения прерываний этот флаг должен быть установлен в «1». Разрешение/запрещение отдельных прерываний производится установкой или сбросом соответствующих разрядов регистров масок прерываний (регистра управления прерываниями). Если флаг сброшен, то прерывания запрещены независимо от состояния разрядов этих регистров. Флаг сбрасывается аппаратно после входа в прерывание и восстанавливается командой RETI для разрешения обработки следующих прерываний.

- **Бит 6 – T: Хранение копируемого бита.** Этот разряд регистра используется в качестве источника или приемника команд копирования битов BLD (Bit Load) и BST (Bit Store). Заданный разряд любого РОН может быть скопирован в этот разряд командой BST или установлен в соответствии с содержимым данного разряда командой BLD.

- **Бит 5 – H: Флаг половинного переноса.** Устанавливается в «1», если произошел перенос из младшей половины байта в старшую (из 3-го разряда в 4-й) или заем из старшей половины байта при выполнении некоторых арифметических операций.

- **Бит 4 – S: Флаг знака.** Этот флаг равен результату операции «Исключающее ИЛИ» между флагами N (отрицательный результат) и V (переполнение числа в дополнительном коде). Если после операций сложения или вычитания чисел со знаком флаг переполнения V будет установлен в 1, то результатом будет 9-разрядное число, причем старшим (т.е. знаковым) разрядом будет флаг S, а 8 остальных битов результата будут храниться в 8-разрядном регистре-приемнике выполнявшейся операции.

- Бит 3 – **V: Флаг переполнения дополнительного кода.** Устанавливается в «1», при переполнении разрядной сетки знакового результата. Используется при работе со знаковыми числами (представленными в дополнительном коде).

- Бит 2 – **N: Флаг отрицательного значения.** Устанавливается в «1», если старший (7-й) разряд результата операции равен «1». В противном случае флаг равен «0».

- Бит 1 – **Z: Флаг нулевого значения.** Этот флаг устанавливается в «1», если результат выполнения операции равен нулю.

- Бит 0 – **C: Флаг переноса/заема.** Устанавливается в «1», если в результате выполнения операции произошел выход за границы байта.

Адреса всех регистров ввода-вывода микроконтроллера ATmega8535 можно найти в справочном листке на микроконтроллер.

Для хранения переменных помимо регистров общего назначения также используется статическое ОЗУ объемом от 512 байт до 8 Кбайт. Ряд микроконтроллеров семейства, кроме того, имеют возможность подключения внешнего статического ОЗУ объемом до 64 Кбайт.

В адресном пространстве ОЗУ также расположены все регистры микроконтроллеров, под них отведены младшие 96 (256) адресов. Каждый регистр имеет свой собственный адрес в пространстве памяти данных. Поэтому к регистрам можно обращаться двумя способами — как к регистрам и как к памяти, несмотря на то что физически эти регистры не являются ячейками ОЗУ. Так, регистрам общего назначения R0 – R31 соответствуют адреса ОЗУ \$000 – \$01F, регистрам ввода/вывода \$00 – \$3F соответствуют адреса ОЗУ \$020 – \$05F (номер регистра плюс \$20).

2.2 Общие сведения о платформах разработки

Для разработки программного обеспечения для микроконтроллеров типа AVR используется несколько сред (платформ) разработки. Основными из них являются WinAVR, AVR Studio и некоторые другие.

WinAVR представляет собой среду разработки для операционных систем семейства Windows, созданную с целью написания и отладки программ для микроконтроллеров серии AVR и AVR32 от компании Atmel.

WinAVR состоит из GNU GCC компилятора, поддерживающего языки C, C+ и Objective-C, обеспечивая полный цикл разработки для AVR/AVR32 и дополнительных инструментов, каждый из которых выполняет определенную задачу, помогая написанию программного обеспечения для микроконтроллеров. Наиболее важные из них:

- Programmers Notepad – текстовый редактор для создания программ;
- AVR-LibC – библиотека микроконтроллеров AVR;
- AVRDUDE – утилита, предназначенная для программирования микросхем и использующая SPI-интерфейс;

- GNU Debugger (GDB) – отладчик с командной строкой;
- Insight – отладчик с графическим интерфейсом;
- Simulavr – симулятор микроконтроллеров AVR с поддержкой отладчика GDB;
- SRecord – набор утилит для работы с загрузочными файлам EPROM разных форматов.

Существует несколько версий интегрированной среды разработки IDE (Integrated Development Environment) микропроцессорных устройств AVR Studio, функциональные возможности которых постоянно расширяются. Одна из самых простых и удобных в работе является версия AVR Studio 4, предназначенная для написания и отладки прикладных программ для AVR микропроцессоров в среде Windows. AVR Studio 4 содержит ассемблер и симулятор.

В интегрированную среду разработки AVR Studio 5 внесены ряд добавок, расширяющих функциональные возможности среды и облегчающих процесс разработки программного обеспечения. В частности, она содержит встроенные примеры программ со всеми настройками для AVR32 UC3 и AVR Xmega, встроенный набор библиотек AVR Software Framework, написанных единообразно для разных семейств и архитектур, который обеспечивает возможность быстрого перевода проектов как между семействами, так и между архитектурами и др.

Интегрированная платформа разработки Atmel Studio 6 предназначена для проектирования и отладки приложений как на базе микроконтроллеров AVR Atmel, так и новых типов контроллеров ARM Cortex-M. Atmel Studio 6 представляет собой единую, простую в использовании среду разработки, компоновки и отладки приложений, написанных на языке C/C++ либо ассемблере. Эта платформа, как и ее предшественники являются бесплатными.

В нее интегрирована обширная библиотека бесплатного исходного кода Atmel Software Framework (ASF), содержащая свыше 1600 примеров ARM- и AVR-проектов. Эта библиотека дополняет преимущества платформы Atmel Studio 6, предоставляя доступ к готовому коду в той же среде, что сводит к минимуму необходимость низкоуровневого программирования в проектах.

В версии платформы, Atmel Studio 6.2, были добавлены современные функции отладки, в частности отслеживание данных и прерываний (Data and Interrupt Trace). С появлением решений Atmel Gallery и Atmel Spaces проектирование встраиваемых микроконтроллеров на платформе Atmel Studio 6 стало еще проще, а время разработки и затраты снизились. Atmel Gallery представляет собой сетевой магазин приложений, где можно найти инструменты разработки и встраиваемое программное обеспечение. Atmel Spaces — это облачное пространство для совместной разработки, позволяющее хранить проекты по созданию программного и аппаратного обеспечения для микроконтроллеров Atmel.

Основное достоинство новой версии бесплатной популярной среды разработки Atmel Studio 6 от Atmel - она теперь основывается на движке Visual Studio 2010, и имеет абсолютно такой же интерфейс, как и популярная среда разработки программ для PC - Microsoft Visual Studio 2010. Это упрощает освоение среды

программирования AVR для тех, кто уже знаком с традиционным интерфейсом Visual Studio.

Существенным недостатком 5-й и последующих версий Atmel Studio является их громоздкость (от 600 до 900 Мбайт), что затрудняет использовать их на лабораторных компьютерах с ограниченным объемом оперативной памяти.

3 Описание лабораторной установки

Лабораторная установка для создания и отладки приложений и исследования процессов функционирования микроконтроллеров типа AVR Atmel представляет собой персональный компьютер, на котором установлена интегрированная среда разработки Atmel Studio 6 (либо аналогичная). При запуске Atmel Studio 6 система предлагает создать новый проект (New Project) или открыть уже существующий (Open) проект приложения. Если приложение уже существует, то открывается файл этой программы, а если она создается вновь, то выбирается New Project. В результате открывается окно (рисунок 3.1), в котором нужно выбрать тип проекта.

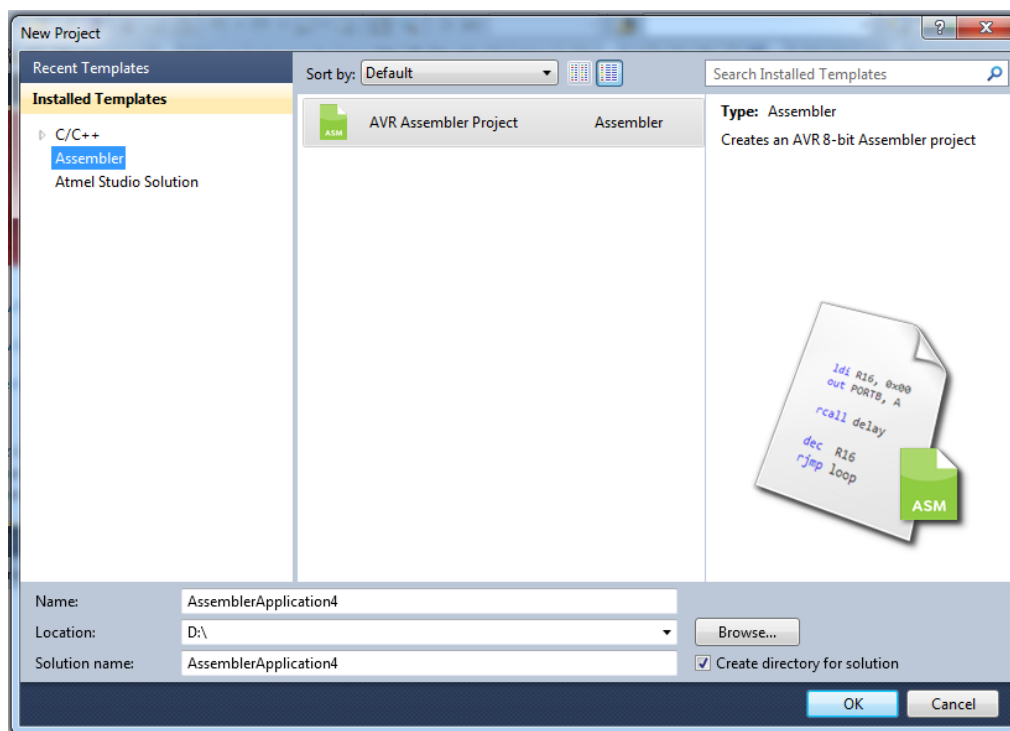


Рисунок 3.1 – Вид окна Atmel Studio 6 после выбора нового проекта

В данной работе программирование будет выполняться на языке Ассемблер. Для этого в окне тип проекта (Installed Templates) выбирается строка Assembler, задается произвольное имя проекта (Name) и место локализации файла (Location). Затем нажимается клавиша ОК. Открывается окно (рисунок 3.2), в котором выбирается платформа отладки (Device Selection) и тип

микроконтроллера в окне Device Family, в нашем примере ATmega16. При этом в правом окне Device Info можно ознакомиться со схемными особенностями и параметрами выбранного микроконтроллера (при наличии выхода в Интернет).

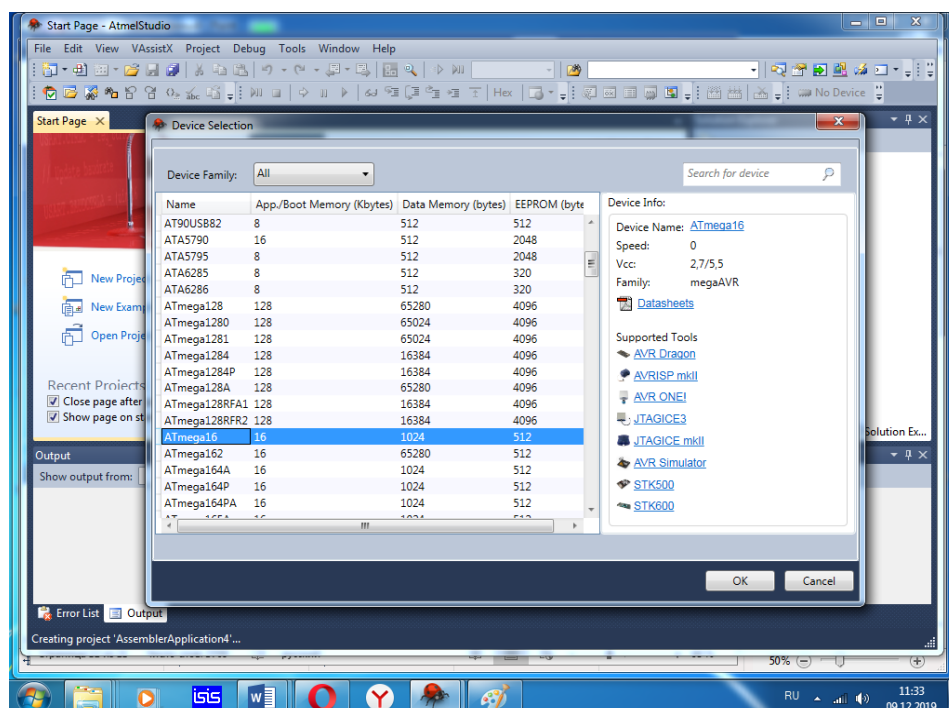


Рисунок 3.2 – Вид окна при выборе типа микроконтроллера

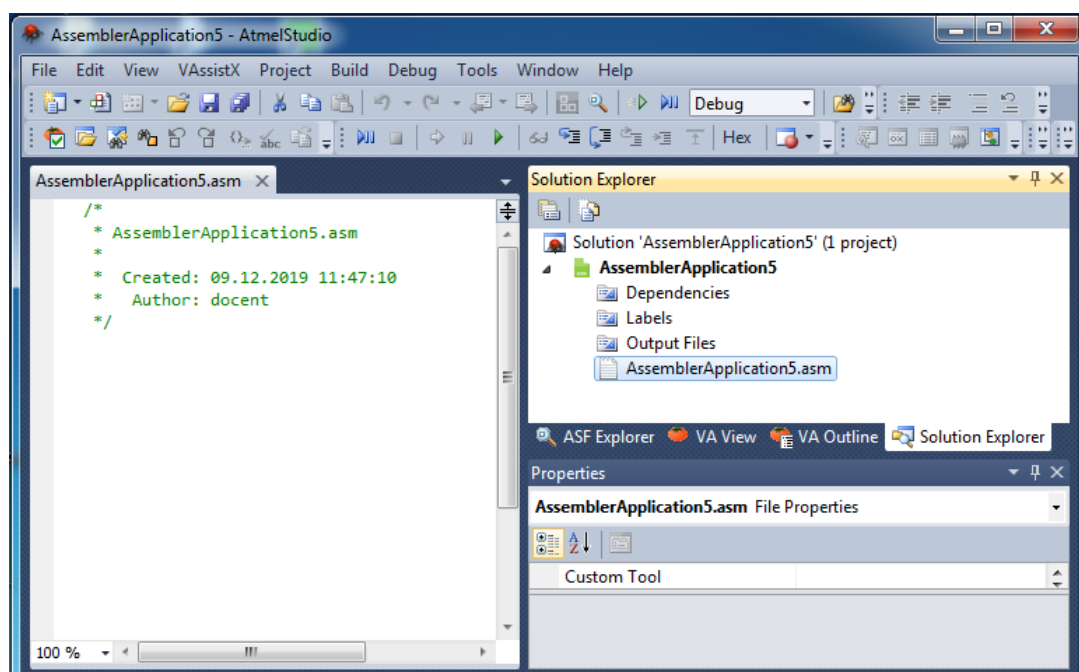


Рисунок 3.3 – Вид окна с редактором ассемблерной программы

После нажатия кнопки ОК появится листинг чистой программы с синтаксисом ассемблера по умолчанию (рисунок 3.3).

Кроме стандартных кнопок управления главного меню, имеются специальные кнопки, применяемые преимущественно при компиляции и отладки программы. Назначение этих кнопок выводится при наведении на них курсора мышки.

После ввода в рабочее окно текста ассемблерной программы и нажатия кнопки ассемблирования программы Build Solution при отсутствии ошибок в нижнем окне появится сообщение об успешном ассемблировании Build succeeded. При наличии ошибок будут указаны номера ошибочных строк и характер ошибок. После исправления ошибок нужно снова провести ассемблирование.

После ассемблирования программы можно перейти к пошаговой отладке программы для чего нужно нажимать клавишу Step Into. При нажатии может появиться сообщение: Выбрать симулятор. В этом случае нужно активировать AVR Simulator и нажать ОК. Программа готова к отладке.

Для просмотра состояния процессора, содержимого регистров, портов ввода-вывода и памяти следует открыть соответствующие окна путем нажатия клавиш: Processor View, I/O View, Registers или Memory.

Информация о регистрах ввода/вывода, процессоре и регистрах общего пользования расположена и распределена по группам на вкладке I/O View (рисунок 3.4).

Register	
R00= 0x00	R01= 0x00
R02= 0x00	R03= 0x00
R04= 0x00	R05= 0x00
R06= 0x00	R07= 0x00
R08= 0x00	R09= 0x00
R10= 0x00	R11= 0x00
R12= 0x00	R13= 0x00
R14= 0x00	R15= 0x00
R16= 0x00	R17= 0x00
R18= 0x00	R19= 0x00
R20= 0x00	R21= 0x00
R22= 0x00	R23= 0x00
R24= 0x00	R25= 0x00
R26= 0x00	R27= 0x00
R28= 0x00	R29= 0x00
R30= 0x00	R31= 0x00

Рисунок 3.8 - Окно регистров

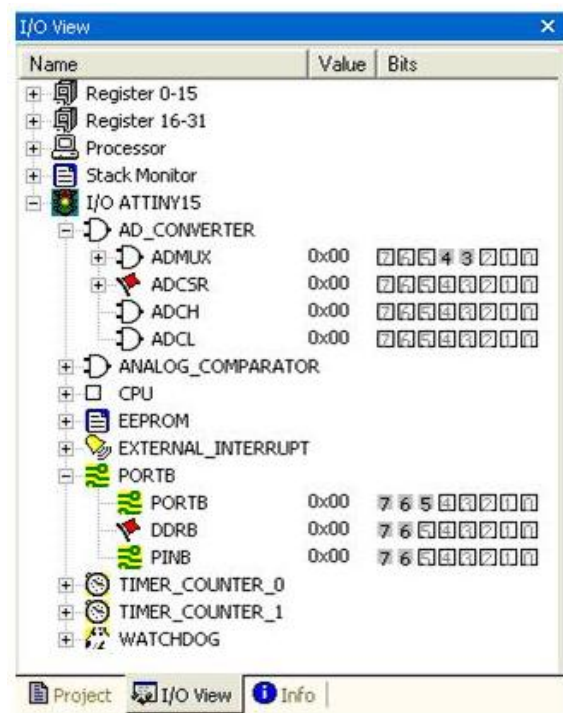


Рисунок 3.9 – Вид окна ввода/вывода

Инструкция, которая будет выполняться следующей, обозначается желтой стрелочкой. Для удобства отладки применяется контрольные точки.

Контрольная точка – инструкция в программе дойдя до которой выполнение программы приостановится. Установленная контрольная точка отмечена красным кружком.

После того как проект прошел отладку данную программу можно «зашивать» в микроконтроллер.

4 Программа и методика исследований

В процессе выполнения лабораторной работы при программировании в среде Atmel Studio необходимо выполнить следующие действия.

- 4.1 Изучить структуру и назначение функциональных блоков микроконтроллера.
- 4.2 Ознакомиться с особенностями системы команд микроконтроллеров типа AVR.
- 4.3 Ознакомиться со средой программирования и отладки программ типа Atmel Studio 6 (или 7).
- 4.4 Подготовить в редакторе Atmel Studio ознакомительную программу на ассемблере, приведенную в приложении А п.1.
- 4.5 Записать в комментариях значение каждой команды.
- 4.6 Выполнить ассемблирование программы.
- 4.7 Запустить отладчик программы и исправить, при их наличии, синтаксические ошибки.
- 4.8 Исследовать изменение содержимых рабочих регистров, указателя стека, флагов и ячеек памяти при пошаговом выполнении программы.
- 4.9 Повторить пп.4.4-4.8 для задания 2, приложения А согласно варианту.
- 4.10 Оформить отчет по лабораторной работе.

ПРИМЕЧАНИЕ: Первые три пункта программы выполняются в процессе домашней подготовки.

5 Содержание отчета

- 5.1 Цель и программа работы.
- 5.2 Структурная схема микроконтроллера.
- 5.3 Скриншоты и текст программы с комментариями на ассемблере.
- 5.4 Выводы по результатам исследований.

6 Контрольные вопросы

- 6.1 В чем состоит отличие архитектуры микроконтроллеров AVR от архитектуры универсальных микропроцессоров?
- 6.2 Поясните назначение функциональных узлов микроконтроллера ATmega16 и расскажите о работе устройства.

- 6.3 Расскажите об особенностях настройки портов ввода/вывода.
- 6.4 Какие функции выполняют регистры X, Y, Z?
- 6.5 С какой целью в схему микроконтроллера включен сторожевой таймер?
- 6.6 Для чего в микроконтроллер встроен аналоговый компаратор?
- 6.7 Расскажите о структуре flash –памяти программ и о адресном пространстве статической памяти микроконтроллера.
- 6.8 Раскройте особенности программирования RISC процессоров.
- 6.9 Какие ассемблерные директивы наиболее широко применяются в ассемблере МК AVR и каково их назначение?
- 6.10 Какие битовые операции и с какой целью они используются в программах для микроконтроллеров?
- 6.11 Как осуществляется процедура исследования функционирования микроконтроллера?

Приложение А. Варианты задания

1) Ознакомительная программа

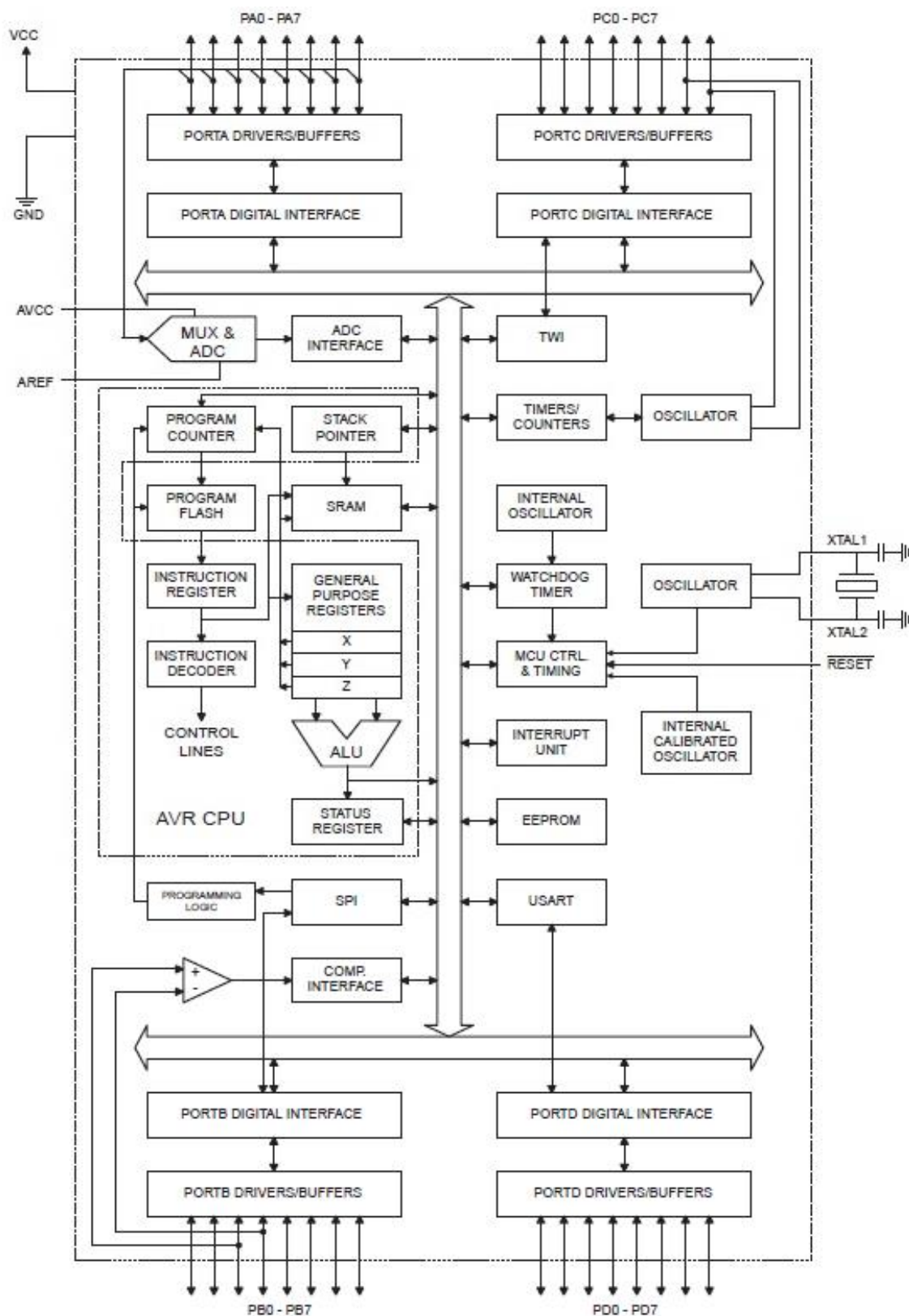
```
1.  * Lab_1mega16.asm
2.  *
3.  */
4.  .include "m16def.inc"
5.  .def temp=r16
6.  .def count=r17
7.  rjmp init
8.  ;
9.  init:
10. ldi temp,$80
11. out SPL,temp
12. ldi count,5
13. m1:
14. inc r18
15. inc r19
16. add r18,r19
17. mov r20,r18
18. push r20
19. pop r21
20. subi r20,2
21. dec count
22. brne m1
23. ret
```

2) Составить программу на ассемблере по следующему словесному описанию и исследовать функционирование (содержимое регистров) при пошаговом выполнении программы:

- Установить указатель стека на ячейку с адресом $N+80$.
- Организовать цикл. Установить значение счетчика циклов равное $N+5$.
- Инкрементировать регистр 18, затем 19, сложить их содержимое и перенести сумму в регистр 20.
- Сохранить значение регистра 20 в стеке, затем перенести это число из вершины стека в регистр 21.
- Вычесть из регистра 20 число 2, уменьшить счетчик циклов на единицу.
Продолжать выполнять действия в цикле, пока счетчик не станет равным нулю.

N - последняя цифра в номере зачетной книжки.

Приложение Б. Структурная схема микроконтроллера ATmega16



ЛАБОРАТОРНАЯ РАБОТА № 2

ИССЛЕДОВАНИЕ СИСТЕМЫ КОМАНД МИКРОКОНТРОЛЛЕРОВ AVR

1 Цель работы

Изучить основные команды микроконтроллеров семейства AVR и исследовать влияние команд различного типа на флаги регистра состояния и регистры управления портами ввода/вывода, а также на работу функциональных узлов устройства. Приобрести практические навыки разработки программ средней сложности на языке Ассемблера и отладки их в специализированной инструментальной среде.

2 Особенности архитектуры микроконтроллеров семейства ATmega

2.1 Краткая характеристика системы команд

Система команд микроконтроллеров ATMEL семейства AVR достаточно большая. Одной из отличительных особенностей микроконтроллеров AVR является то, что почти все команды выполняются за 1 тактовый цикл. Исключение составляют команды перехода. Это существенно увеличивает производительность микроконтроллера даже при относительно невысокой тактовой частоте.

Все команды можно классифицировать на 5 типов:

- 1) арифметические команды и команды сдвига;
- 2) логические команды;
- 3) команды перехода (передачи управления);
- 4) команды передачи (пересылки) данных;
- 5) побитовые команды и команды тестирования битов.

Операнды команд обозначаются следующим способом:

Rd	Регистр - приемник, место, куда заносится результат выполнения команды
Rs	Регистр - источник в двухоперандных командах. Его значение после выполнения команды не изменяется.
I/O	Регистр ввода-вывода. Это порты, таймеры и т.д.
K	8-ми разрядная константа в операциях со "старшими" регистрами общего назначения (R16-R31)
b	Номер бита в операциях с регистрами ввода-вывода
A	16-ти разрядный адрес при работе с памятью данных (прямая адресация)

q	6-ти разрядное смещение при работе с памятью данных
X	Регистровая пара X. Состоит их регистров XL (R26) и XH (R27)
Y	Регистровая пара Y. Состоит их регистров YL (R28) и YH (R29)
Z	Регистровая пара Z. Состоит их регистров ZL (R30) и ZH (R31)

В микроконтроллерах имеются следующие флаги, являющиеся битами регистра состояния SREG. Адрес регистра 0x3F (либо 0x5F).

I	SREG.7	Бит разрешения прерывания. Если он = 0, то все прерывания в МК запрещены. Если он =1, то разрешением прерываний будут управлять соответствующие биты периферии.
T	SREG.6	Битовый аккумулятор. С этим битом работают команды BST и BLD
H	SREG.5	Флаг переноса из младшей тетрады
S	SREG.4	Sign - исключающее ИЛИ битов N и V
V	SREG.3	oVerflow - переполнение
N	SREG.2	Negative - Результат операции < 0
Z	SREG.1	Zero - Результат операции равен нулю
C	SREG.0	Carry - Флаг переноса

Основные команды микроконтроллеров семейства AVR даны в приложении Б. Наиболее часто используются следующие команды.

Команды передачи данных	
MOV Rd, Rr	; Move between Registers $Rd \leftarrow Rr$
LDI Rd, K	; Load Immediate $Rd \leftarrow K$
LD Rd, X	; Load Indirect $Rd \leftarrow (X)$
LD Rd, X+	; Load Indirect and Post-inc. $Rd \leftarrow (X)$, $X \leftarrow X + 1$
LD Rd, -X	; Load Indirect and Pre-dec. $X \leftarrow X - 1$, $Rd \leftarrow (X)$
CLR Rd	; Clear Register $Rd \leftarrow Rd \oplus Rd$
SER Rd	; Set Register $Rd \leftarrow \$FF$
ST X, Rr	; Store Indirect $(X) \leftarrow Rr$
ST X+, Rr	; Store Indirect and Post-inc. $(X) \leftarrow Rr$, $X \leftarrow X + 1$
IN Rd, P	; Input Port $Rd \leftarrow P$
OUT P, Rr	; Output Port $P \leftarrow Rr$
PUSH Rr	; Push Register on Stack $STACK \leftarrow Rr$
POP Rd	; Pop Register from Stack $Rd \leftarrow STACK$
Команды с битами	
SBI P, b	; Set Bit in I/O Register $I/O(P,b) \leftarrow 1$
CBI P, b	; Clear Bit in I/O Register $I/O(P,b) \leftarrow 0$
Команды арифметические и сдвига	
ADD Rd, Rr	; Add Two Registers $Rd \leftarrow Rd + Rr$ Z,C,N,V,H

ADC Rd, Rr	; Add with Carry Two Registers $Rd \leftarrow Rd + Rr + C$
SUBI Rd, K	; Subtract Constant from Register $Rd \leftarrow Rd - K$
LSL Rd	; Logical Shift Left $Rd(n+1) \leftarrow Rd(n)$, $Rd(0) \leftarrow 0$
ROR Rd	; Rotate Right through Carry $Rd(7) \leftarrow C$, $Rd(n) \leftarrow Rd(n+1)$
ASR Rd	; Arithmetic Shift Right $Rd(n) \leftarrow Rd(n+1)$
Команды перехода	
SBRC Rr, b	; Skip if Bit in Register Cleared if ($Rr(b) = 0$) $PC \leftarrow PC + 2$ or 3
SBRS Rr, b	; Skip if Bit in Register is Set if ($Rr(b) = 1$) $PC \leftarrow PC + 2$ or 3
SBIC P, b	; Skip if Bit in I/O Register Cleared if ($P(b) = 0$) $PC \leftarrow PC + 2$ or 3
SBIS P, b	; Skip if Bit in I/O Register is Set if ($P(b) = 1$) $PC \leftarrow PC + 2$ or 3
RJMP k	; Relative Jump $PC \leftarrow PC + k + 1$
IJMP	; Indirect Jump to (Z) $PC \leftarrow Z$
RCALL k	; Relative Subroutine Call $PC \leftarrow PC + k + 1$
ICALL	; Indirect Call to (Z) $PC \leftarrow Z$
RET	; Subroutine Return $PC \leftarrow STACK$
RETI	; Interrupt Return $PC \leftarrow STACK I$
CPSE Rd, Rr	; Compare, Skip if Equal if ($Rd = Rr$) $PC \leftarrow PC + 2$ or 3
CP Rd, Rr	; Compare $Rd - Rr$
CPC Rd, Rr	; Compare with Carry $Rd - Rr - C$
BRBS s, k	; Branch if Status Flag Set if ($SREG(s) = 1$) then $PC \leftarrow PC + k + 1$
BRBC s, k	; Branch if Status Flag Cleared if ($SREG(s) = 0$) then $PC \leftarrow PC + k + 1$
BREQ k	; Branch if Equal if ($Z = 1$) then $PC \leftarrow PC + k + 1$
BRNE k	; Branch if Not Equal if ($Z = 0$) then $PC \leftarrow PC + k + 1$
BRCS k	; Branch if Carry Set if ($C = 1$) then $PC \leftarrow PC + k + 1$
BRCC k	; Branch if Carry Cleared if ($C = 0$) then $PC \leftarrow PC + k + 1$
Прочие, часто используемые команды	
SEI	; Global Interrupt Enable $I \leftarrow 1$
CLI	; Global Interrupt Disable $I \leftarrow 0$
NOP	; No Operation

Примеры применения некоторых команд:

ldi R16, 0b00001001 out PORTD, R16 in R25, PORTB cpi R25, 4 breq exit exit: nop	; загрузка константы в POH (R16 - R32) ; запись значения регистра в порт D ; считать значения Port B в регистр R25 ; сравнить считанное значение с константой =4 ; переход на метку если было равно ; ; метка ; пустой такт
sbi PORTD, PD4 rcall my_delay cbi PORTD, PD4 my_dlay: nop nop nop nop nop	; записать в 4-й бит порта D лог. "1" ; вызов подпрограммы задержки ; сброс 4-го бита порта D ; ; подпрограмма задержки (4 такта) ; холостой такт ; ; ; ; ;

ret	; возвращение из подпрограммы (3 такта)
	; вызов и выполнение подпрограммы my_delay ; займёт 4+ 5+3 = 12 тактов

2.2 Особенности программирования регистров и портов ввода/вывода AVR-микроконтроллеров

Одним из важных аспектов программирования микроконтроллеров является работа с регистрами и портами. У микроконтроллеров серии AVR имеется несколько регистров ввода/вывода и 32 рабочих регистра общего назначения. Регистры ввода/вывода имеют номера, например от \$0 до \$3F. На схемах порты ввода/вывода имеют буквенные обозначения PortB, PinB, PortD, PinD и др. Причем, в разных моделях микроконтроллеров их адреса могут отличаться. Для того чтобы связать адреса регистров ввода/вывода и других регистров микроконтроллера с их именами, следует в начале программы подключить заголовочный файл, соответствующий используемому микроконтроллеру. Так, например, если программа предназначена для микроконтроллера AT90s8515, то нужно с помощью специальной директивы подключить соответствующий файл:

```
.include "8515def.inc"
```

При программировании микроконтроллеров нельзя непосредственно записать число в регистр ввода/вывода. Вместо этого нужно записать число в один из рабочих регистров общего назначения, а затем скопировать значение этого регистра в регистр ввода/вывода. Рабочие регистры обозначаются как R0, R1, R2, ..., R31. Записать число – непосредственный операнд (константу K) можно только в регистры R16- R31 (командой ldi Rd,K), а в регистры R0- R15 нельзя.

Для упрощения написания программ очень удобно давать регистрам имена. Целесообразно давать имена, соответствующие хранимой информации. Например, если регистр R16 используется для хранения временной информации, то его можно назвать temp. Это выполняется с помощью директивы def:

```
.def temp = R16
```

Микроконтроллеры серии Atmel Mega имеют несколько портов Ввода/Вывода, обозначаемых буквами (PortA, PortB и т.д.). Эти порты являются двунаправленными 8-разрядными, т.е. каждый такой порт имеет 8 выводов (ножек или пинов, от англ. *pin*) позволяет читать или записывать до 8 двоичных сигналов. Каждая линия порта может быть запрограммирована на вход или на выход. Выходные усилители (драйверы) обеспечивают токовую нагрузочную способность 20 мА на линию порта (максимальное значение 40 мА), что позволяет, например, непосредственно подключать к микроконтроллеру светодиоды и биполярные транзисторы. Общая токовая нагрузка на все линии одного порта не должна превышать 80 мА (все значения приведены для напряжения питания 5 В). Для управления каждым из портов ВВ имеются три 8-разрядных регистра в/в: DDRx, PORTx и PINx.

Упрощенная схема ввода/вывода изображена на рисунке 2.1. Биты регистра DDR задают режим ввода (лог. 0) или вывода (лог.1). Регистр PINx позволяет считать логическое значение порта, независимо от того, какой режим ввода/вывода установлен. Регистр PORTx в режиме вывода задает выходные сигналы на выводах порта, а в режиме ввода задает включение (лог. 1) или отключение (лог. 0) «подтягивающих» резисторов для выводов порта.

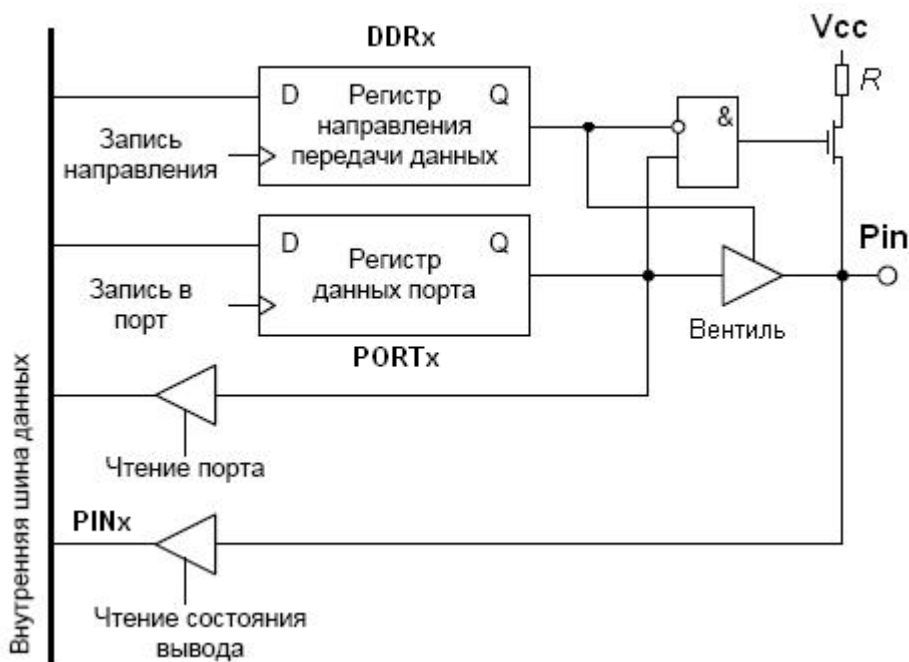


Рисунок 2.1 – Упрощенная схема порта ввода/вывода

Для того чтобы сделать пины выходами, необходимо записать в соответствующие биты регистра DDRx "1". Например, если требуется сделать пин PB7 порта В входом, а остальные ножки выходами, то для этого необходимо записать в регистр DDRB значение 0b01111111. Приставка 0b означает, что число записано в двоичном виде. При начальном запуске регистры DDRx обнулены, т.е. все ножки (пины) микроконтроллера являются входами. Рекомендуется неиспользуемые ножки в устройстве делать входами и «потянуть» их потенциал к источнику питания.

«Подтягивание» внешнего вывода к высокому потенциалу осуществляется путем подключения вывода через транзисторный ключ и подтягивающий резистор R. Как видно из схемы, «подтягивание» реализуется только тогда, когда внешний вывод (Pin) работает в режиме ввода данных, т.е., когда бит регистра направления данных сброшен в 0, а бит регистра данных порта установлен в единицу.

Для побитного обращения к линиям порта, можно использовать команды SBI, CBI, SBIS, SBIC... Команда SBI устанавливает соответствующий бит порта

в "1", а команда CBI делает обратное — очищает соответствующий бит т.е. записывает 0 в этот бит. Например:

1. ldi R16, 0xFF
2. out DDRB, R16 ; все ножки (пины) — выходы
3. sbi PortB, 5 ; установить на ножке лог. "1"
4. cbi PortB, 5 ; сбросить, на ножке появится лог. "0"

3 Программа и методика выполнения лабораторной работы

- 3.1. Ознакомиться с системой команд микроконтроллеров AVR и изучить наиболее часто используемые команды.
- 3.2. Повторить материал по архитектуре микроконтроллера и схеме организации памяти и портов ввода/вывода микроконтроллера.
- 3.3. Составить алгоритм программы согласно заданному варианту.
- 3.4. Составить текст программы на языке ассемблера и откомпилировать программу.
- 3.5. Исправить ошибки, при их обнаружении компилятором.
- 3.6. Провести исследование процесса выполнения программы в среде AVR Studio.

4 Описание лабораторной установки

Лабораторная установка для создания и отладки приложений и исследования процессов функционирования микроконтроллеров типа AVR Atmel представляет собой персональный компьютер, на котором установлена интегрированная среда разработки Atmel Studio 6 либо Atmel Studio старших версий. Описание установки приведено в лабораторной работе №1.

5 Содержание отчета

- 5.1. Цель работы.
- 5.2. Программа исследований.
- 5.3. Перечень программно доступных регистров и схема портов ввода/вывода.
- 5.4. Текст исследуемой программы на языке ассемблера.
- 5.5. Описание результатов исследований.
- 5.6. Выводы по работе.

6 Контрольные вопросы

- 6.1.Расскажите о структуре регистров микроконтроллера и их назначении.
- 6.2.В чем состоит особенность регистров X, Y, Z?
- 6.3.Какие флаги определяют состояние микроконтроллера?
- 6.4.Проведите анализ системы команд микроконтроллера и приведите примеры их применения в конкретных задачах.
- 6.5.Проведите сравнительный анализ команд перехода микроконтроллера.
- 6.6.Начертите упрощенную схему регистров ввода/вывода и поясните принцип задания режимов работы.
- 6.7.Что означает термин «подтягивающий» резистор и каково его назначение?
- 6.8.Для чего предназначены директивы ассемблера? Приведите примеры наиболее часто используемых директив.
- 6.9.Что такое «нагрузочная способность» линии порта и в каких единицах она измеряется?
- 6.10. Какие команды условного перехода используются при программировании задач в микроконтроллерах типа Atmel?
- 6.11. Приведите примеры использования команд работы с памятью.

Приложение А. Варианты задания

1. Составить программу сложения 5 однобайтных беззнаковых чисел, находящихся в ОЗУ. Если в результате сложения происходит переполнение разрядной сетки, вывести в порт В число \$FF. Если переполнения нет, то выводить непрерывно в порт D поочередно числа \$55 и \$AA.
2. Составить программу сложения 8 однобайтных беззнаковых чисел, находящихся в ОЗУ. Затем из полученной суммы вычесть число. Если в результате сложения происходит переполнение разрядной сетки, вывести в порт В число \$FF. Если переполнения нет, то выводить непрерывно в порт D поочередно числа \$0F и \$F0.
3. Составить программу сложения 2 однобайтных беззнаковых чисел, находящихся в ОЗУ. Затем полученную сумму умножить на 7. Если в результате сложения происходит переполнение разрядной сетки, вывести в порт В число \$FF. Если переполнения нет, то выводить непрерывно в порт D поочередно числа \$55 и \$AA.

Приложение Б. Команды микроконтроллеров семейства AVR

Таблица 1. Арифметические и сдвиговые команды

Мнемоника	Описание	Операция	Флаги
ADD Rd, Rr	Сложение двух POH	$Rd = Rd + Rr$	Z,C,N,V, H
ADC Rd, Rr	Сложение двух POH с переносом	$Rd = Rd + Rr + C$	Z,C,N,V, H
ADIW Rd, K **	Сложение регистровой пары с константой	$Rd+1:Rd = Rd+1:Rd + K$	Z,C,N,V, S
SUB Rd, Rr	Вычитание двух POH	$Rd = Rd - Rr$	Z,C,N,V, H
SUBI Rd, K *	Вычитание константы из POH	$Rd = Rd - K$	Z,C,N,V, H
SBC Rd, Rr	Вычитание двух POH с заемом	$Rd = Rd - Rr - C$	Z,C,N,V, H
SBCI Rd, K *	Вычитание константы из POH с заемом	$Rd = Rd - K - C$	Z,C,N,V, H
SBIW Rd, K **	Вычитание константы из регистровой пары	$Rd+1:Rd = Rd+1:Rd - K$	Z,C,N,V, S
DEC Rd	Декрементирование POH	$Rd = Rd - 1$	Z,N,V
INC Rd	Инкрементирование POH	$Rd = Rd + 1$	Z,N,V
ASR Rd	Арифметический сдвиг вправо	$Rd(n) = Rd(n+1),$ $n = 0..6$	Z,C,N,V
LSL Rd	Логический сдвиг влево	$Rd(n+1) = Rd(n),$ $Rd(0) = 0$	Z,C,N,V
LSR Rd	Логический сдвиг вправо	$Rd(n) = Rd(n+1),$ $Rd(7) = 0$	Z,C,N,V
ROL Rd	Сдвиг влево через перенос	$Rd(0) = C,$ $Rd(n+1) = Rd(n),$ $C = Rd(7)$	Z,C,N,V
ROR Rd	Сдвиг вправо через перенос	$Rd(7) = C,$ $Rd(n) = Rd(n+1),$ $C = Rd(0)$	Z,C,N,V
MUL Rd, Rr	Умножение беззнаковых чисел	$R1:R0 = Rd * Rr$	Z,C
MULS Rd, Rr *	Умножение чисел со знаком	$R1:R0 = Rd * Rr$	Z,C

MULSU Rd, Rr ***	Умножение беззнакового числа на число со знаком	$R1:R0 = Rd * Rr$	Z,C
FMUL Rd, Rr ***	Умножение дробных беззнаковых чисел	$R1:R0 = (Rd * Rr) << 1$	Z,C
FMULS Rd, Rr ***	Умножение дробных чисел со знаком	$R1:R0 = (Rd * Rr) << 1$	Z,C
FMULSU Rd, Rr ***	Умножение дробного беззнакового числа и дробного числа со знаком	$R1:R0 = (Rd * Rr) << 1$	Z,C
$0 \leq d \leq 31, 0 \leq r \leq 31, 0 \leq K \leq 255$ (* в командах SUBI, SBCI, MULS: $16 \leq d \leq 31, 16 \leq r \leq 31$; ** в командах ADIW, SBIW: $d = \{24, 26, 28, 30\}, 0 \leq K \leq 63$; *** в командах MULSU, FMUL, FMULS, FMULSU $16 \leq d \leq 23, 16 \leq r \leq 23$)			

ЛАБОРАТОРНАЯ РАБОТА № 3

ИССЛЕДОВАНИЕ ФУНКЦИОНИРОВАНИЯ МИКРОКОНТРОЛЛЕРОВ AVR С ИСПОЛЬЗОВАНИЕМ ИНСТРУМЕНТАЛЬНЫХ СРЕДСТВ МОДЕЛИРОВАНИЯ

1 Цель работы

Целью работы является изучение среды моделирования электронных схем в среде PROTEUS VSM и приобретение практических навыков составления, исследования и отладки микропроцессорных систем и их программного обеспечения.

2 Назначение и краткая характеристика симулятора

Инструментальное средство PROTEUS VSM представляет собой программный симулятор электронных и микропроцессорных систем, который позволяют, без сборки реального устройства, изучить функционирование и отладить работу схемы, найти ошибки, полученные на стадии проектирования, снять необходимые характеристики и др. PROTEUS VSM является так называемой средой сквозного проектирования. Это означает создание устройства, начиная с его графического изображения (принципиальной схемы) и заканчивая изготовлением печатной платы устройства, с возможностью контроля на каждом этапе производства.

Proteus VSM состоит из двух самостоятельных систем: ISIS и ARES. **ISIS** - средство разработки и отладки в режиме реального времени электронных схем, а ARES - это средство разработки печатных плат. В данной дисциплине будет использоваться только система **ISIS**.

3 Описание лабораторной установки и методика создания и исследования схем

Лабораторная установка состоит из персонального компьютера, на котором установлены система редактирования и отладки программ микроконтроллеров Atmel Studio версий 4 – 6 и система моделирования электронных и микропроцессорных систем Proteus VSM. Состав и работа с Atmel Studio описано в методических указаниях к лабораторной работе №1.

Proteus VSM по умолчанию устанавливается в папку C:\Program\Files\Labcenter Electronics\Proteus. При запуске программы **ISIS.exe** появляется основное окно (Рисунок 3.1). Оно в свою очередь состоит из нескольких окон.

Самое большое пространство отведено под Рабочее окно, в котором выполняются все основные процессы создания, редактирования и отладки схемы устройства. Слева вверху маленькое окно предварительного просмотра (Overview Window) с его помощью можно перемещаться по окну редактирования (щелкая левой кнопкой мыши по окну предварительного просмотра, мы перемещаем окно редактирования по схеме, если конечно схема не вмещается в окно).

Все возможные функции и инструменты Proteus VSM доступны через меню, расположенное в самом верху основного окна программы, через пиктограммы находящиеся под меню и в левом углу основного окна, а также через горячие клавиши, которые могут переназначаться пользователем.

В самом низу основного окна расположены: слева направо кнопки вращения и разворота объекта вокруг своей оси, панель управления интерактивной симуляцией (функции: ПУСК-ПОШАГОВЫЙ РЕЖИМ-ПАУЗА-СТОП). В строке журнала событий отображаются ошибки, подсказки, текущее состояние процесса симуляции и т.д., а также координаты курсора, которые отображаются в дюймах.

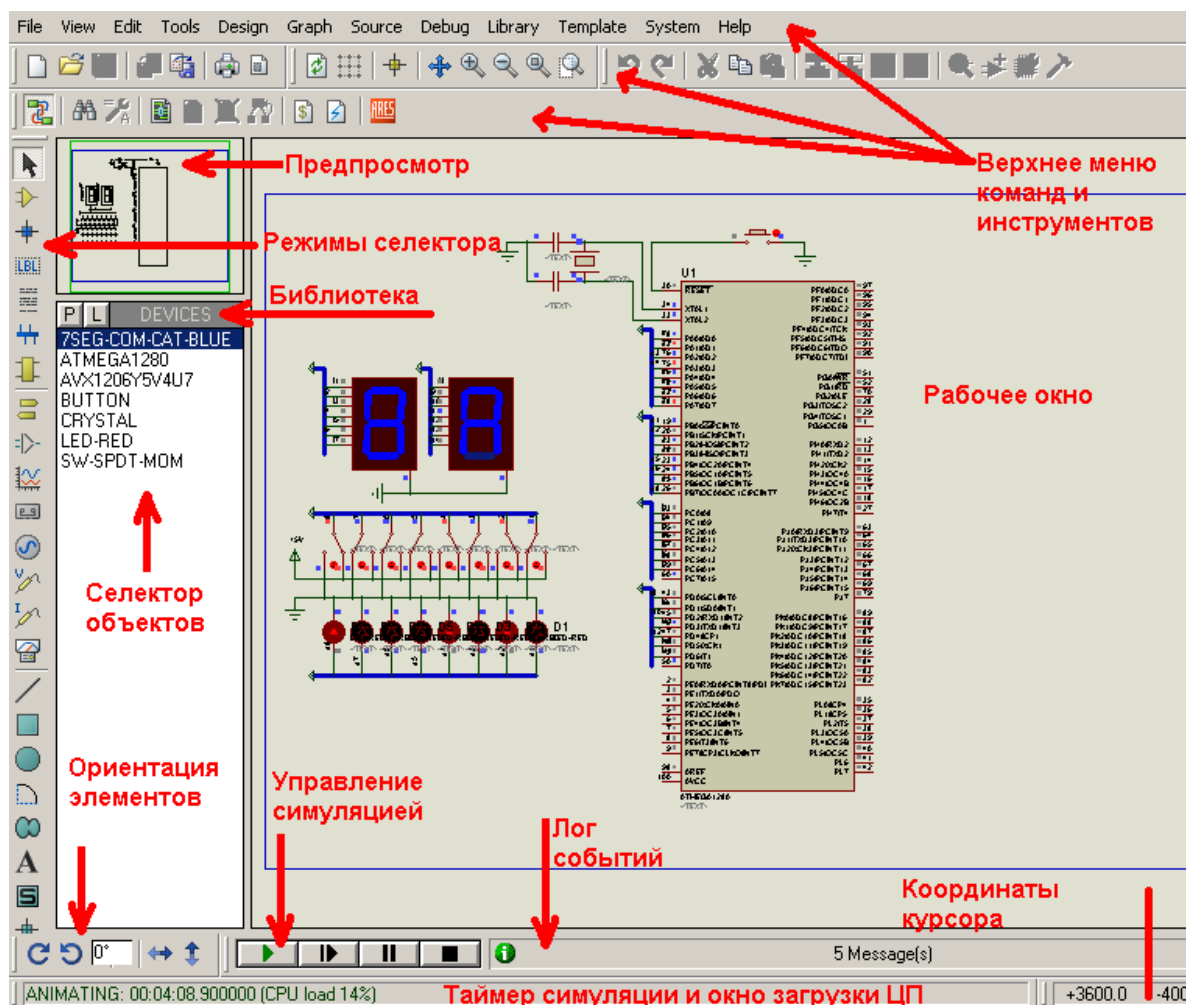


Рисунок 3.1 – Главное окно системы Proteus VSM

Перед созданием и моделированием микропроцессорного устройства необходимо сначала на бумаге начертить его схему, составить программу управления устройством и отладить ее в Atmel Studio.

3.1. Создать новый проект, для чего в меню Файл выбрать соответствующую опцию (при запуске ISIS.exe он создаётся по умолчанию, как и в любой другой программе).

3.2. Зайти в библиотеку компонентов через клавиши меню «Библиотека-Выбор устройства» или нажав кнопку «P» в окне DEVICES (Рисунок 3.2).

3.3. Выбрать нужные компоненты. При выборе элементов можно воспользоваться группами элементов (в библиотеке элементы сгруппированы по классам), а можно поиском, для чего в строку keywords (Маска) необходимо ввести имя требуемого элемента, после выбора каждого элемента нажать ОК, после чего снова можно зайти в библиотеку. Компоненты набираются по одному экземпляру, размножить их можно уже потом просто выбирая в списке Object Selector.

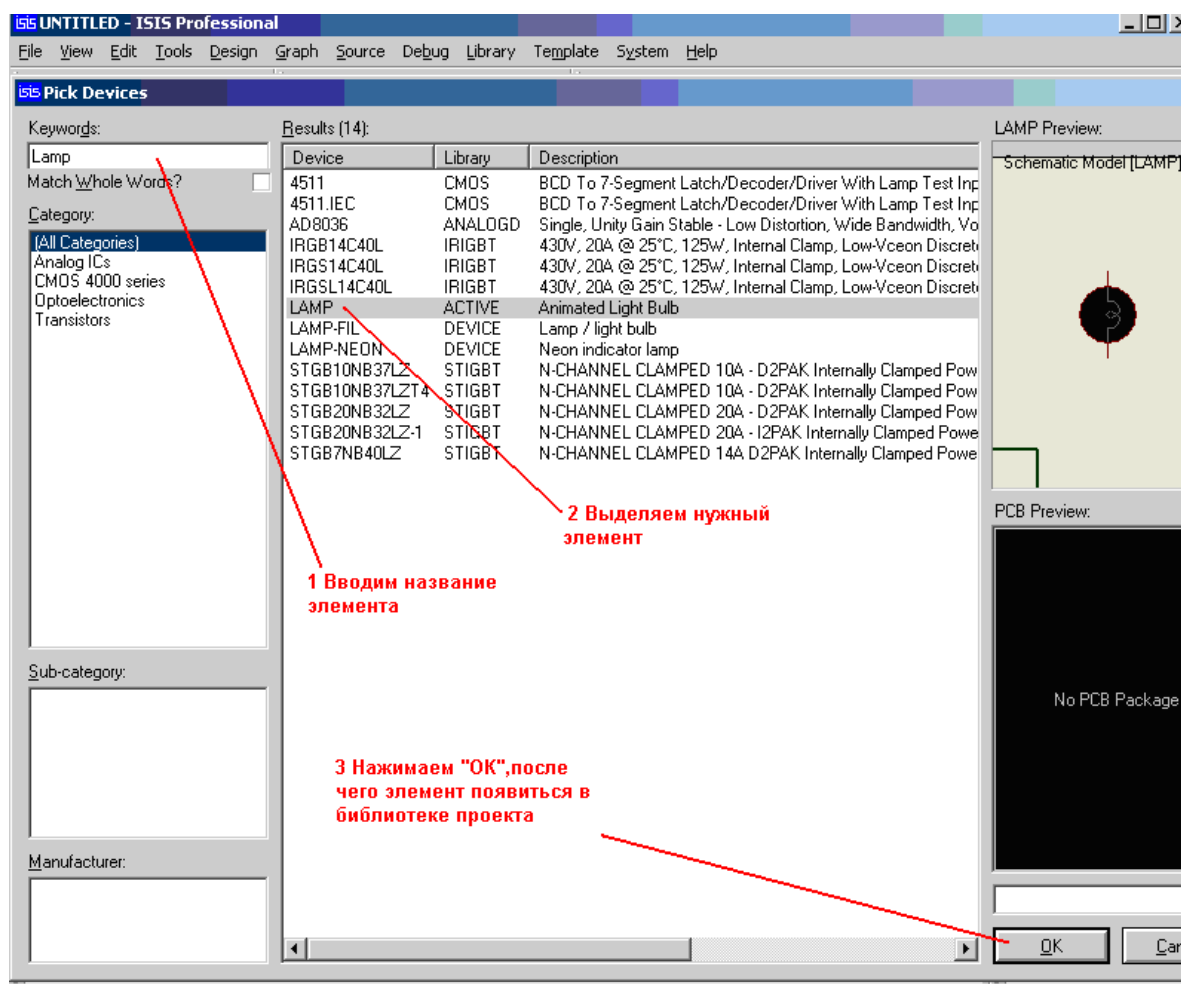


Рисунок 3.2 – Выбор элементов из библиотеки

Для того чтобы изменять параметры объектов, их нужно сначала выделить, что можно сделать только на остановленном проекте. Для выделения одного объекта надо

щелкнуть по нему правой кнопкой мыши. Для выделения группы выделить первый элемент, затем нажать CTRL и последовательно щелкать левой кнопкой мыши по всем необходимым объектам. Выделять объекты надо осторожно, повторный щелчок правой кнопкой мыши по выделенному объекту удалит его, (удалить выделенные объекты можно еще, нажав кнопку DELETE). После выбора всех элементов схемы осуществляется переход к следующему пункту.

3.4. Добавить терминальные элементы питания и сигнальной земли. В окне выбора режима селектора нажать на кнопку «Терминалы» (terminals), выделить Ground (земля) и поместить её на рабочем поле так же, как и предыдущие элементы. Аналогично выделить терминал Power (источник питания) и поместить на поле схемы.

3.5. Задать параметры терминалу Power. Двойным щелчком левой кнопки мыши по элементу Power (источник питания) открыть окно его свойств (рисунок 3.3), в появившемся окне во вкладке Label в графе string задать «+5V» и нажать «ОК».

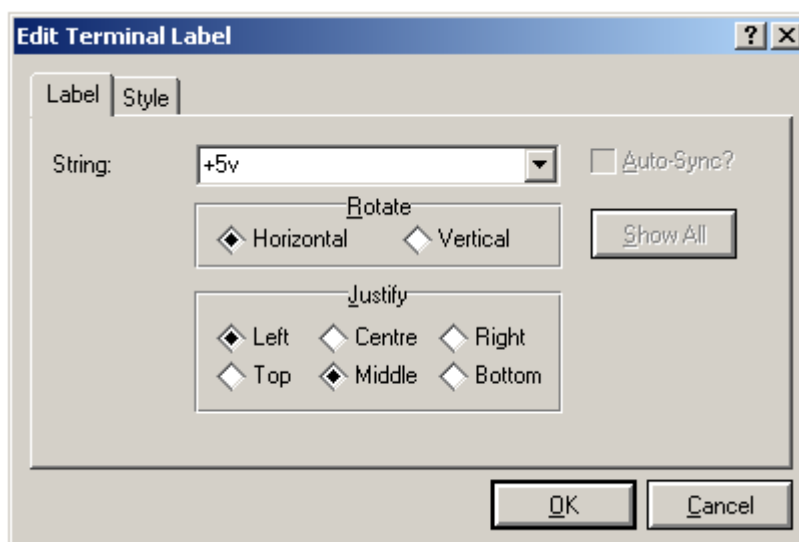


Рисунок 3.3 – Окно задания параметров терминальному элементу

3.6. Соединить элементы согласно разработанной схеме. Подвести курсор мыши к проводнику, выходящему из элемента (курсор меняет цвет на зелёный), нажать левой кнопкой мыши и подвести проводник к элементу, с которым нам необходимо его соединить, после чего нажать левую кнопку мыши ещё раз. Более подробно способы выполнения соединения компонентов приведены в приложении В. Пример подключения компонентов схемы к микроконтроллеру показан на рисунке 3.4.

ПРИМЕЧАНИЕ 1: Такие элементы, как питание микроконтроллера, кварцевый резонатор, конденсаторы по цепям питания нет необходимости добавлять в схему. Все это эмулируется программно по умолчанию. Однако если проект

будет разрабатываться до уровня изготовления печатной платы, необходимо добавлять все элементы.

ПРИМЕЧАНИЕ 2: Тип модели переключателей, резисторов и светодиодов целесообразно изменить на digital (цифровой) для того, чтобы симулятор не тратил время на обсчет аналоговых свойств резисторов и светодиодов. Нужно только задать, светится или нет светодиод, нажата кнопка или нет, то есть чисто логические уровни.

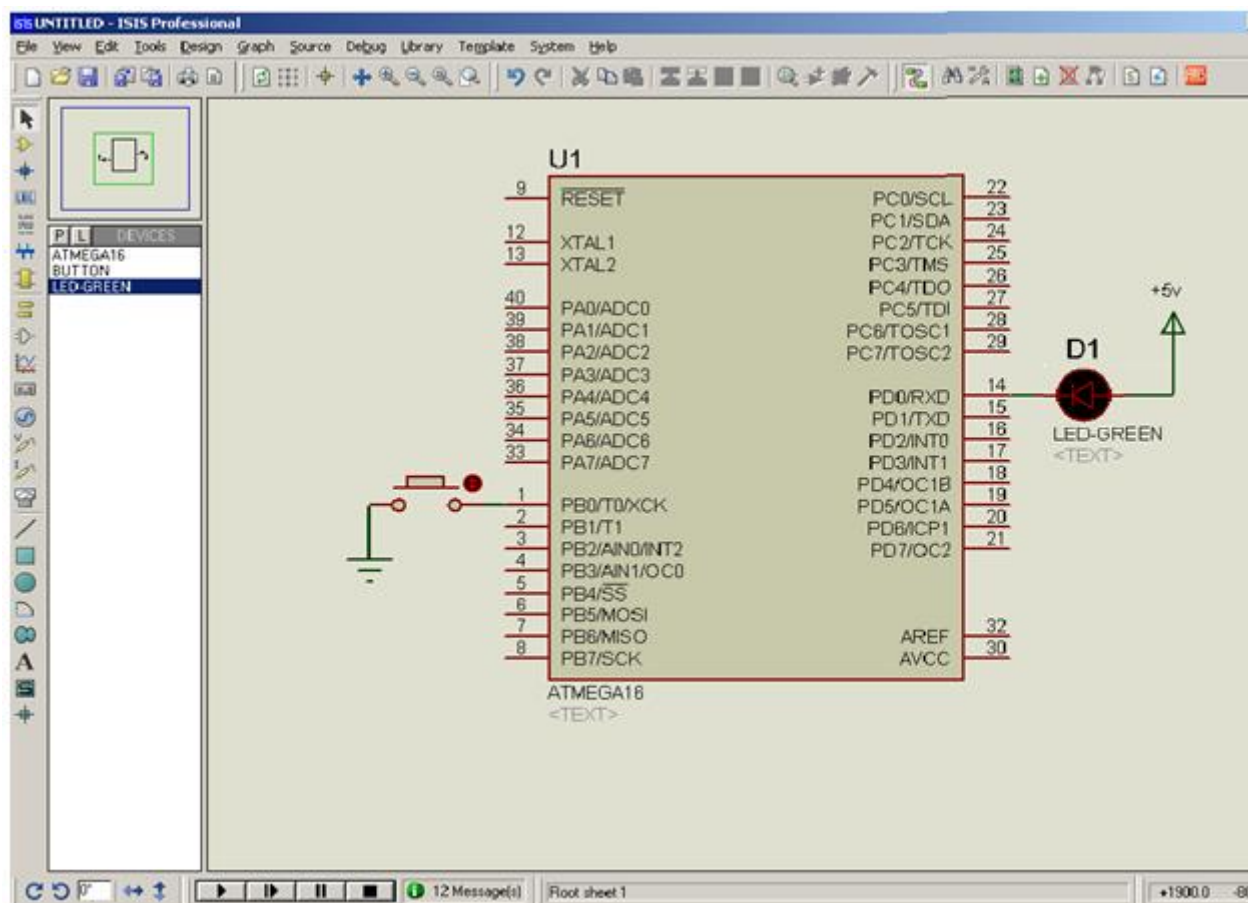


Рисунок 3.4 – Пример подключения компонентов к микроконтроллеру

3.7. Загрузить разработанный на этапе моделирования на AVR Studio **HEX**-файл в микроконтроллер. Щелкнуть ПКМ по микроконтроллеру и выбрать Edit properties (правка свойств), найти строку Program file и указать в ней путь к требуемому hex файлу.

3.8. Запустить процесс симуляции. В меню Отладка (DEBUG) выбрать опцию «Запуск» или нажать внизу на кнопку «Воспроизвести».

3.9. Оценить правильность функционирования спроектированного устройства в непрерывном и пошаговом режимах отладки.

В процессе отладки очень часто приходится контролировать состояния регистров и ячеек памяти. Для этого в симуляторе предусмотрено несколько способов просмотра. Самый удобный способ – использование опций WATCH WINDOW и AVR Internal RAM. Они доступны в меню «Отладка» и становятся

активными только в паузе запущенного процесса симуляции. Нужно щелкнуть ПКМ по появившемуся окну WATCH WINDOW и выбрать одну из предлагаемых опций.

Наиболее наглядное представление дают опции:

ADD ITEMS (BY NAME) — добавить элемент по имени;

ADD ITEMS (BY ADDRESSES) — добавить элемент по адресу.

Путем двойного щелчка ЛКМ в таблицу добавляется соответствующий регистр.

4 Программа выполнения лабораторной работы

- 4.1. Ознакомиться со схемами подключения переключателей (клавиш) и элементов индикации к микроконтроллеру.
- 4.2. Начертить схему микропроцессорного устройства согласно заданному варианту (Приложение Б).
- 4.3. Составить программу опроса клавиш и управления индикацией.
- 4.4. Отладить программу в среде Atmel Studio и убедиться в правильности функционирования устройства.
- 4.5. Начертить разработанную схему микропроцессорного устройства в рабочем окне симулятора Proteus VSM и загрузить исполняемый программный модуль.
- 4.6. Запустить пошаговый режим процесса симуляции и убедиться в корректной работе устройства.
- 4.7. Проверить функционирование разработанного устройства в динамическом режиме.
- 4.8. Подключить виртуальный осциллограф к элементам индикации и исследовать вид сигналов на соответствующих электродах.
- 4.9. Составить отчет о проведенных исследованиях и сформулировать выводы по работе.

ПРИМЕЧАНИЕ: Пп. программы 1-4 выполняются в процессе домашней подготовки.

5 Содержание отчета

- 5.1. Цель и программа работы.
- 5.2. Электрическая схема исследуемого устройства.
- 5.3. Текст программы управления устройством.
- 5.4. Скриншот рабочего окна симулятора с начерченной схемой и с изображением сигналов на экране виртуального осциллографа.
- 5.5. Выводы по работе.

6 Контрольные вопросы

- 6.1. Начертите упрощенную схему микроконтроллера и поясните назначение его функциональных узлов и укажите их параметры.
- 6.2. Поясните схемы подключения к микроконтроллеру кнопочных переключателей и индикаторов. С какой целью последовательно с переключателем включаются резистор и какова должна быть его величина сопротивлений?
- 6.3. Поясните, как микроконтроллер определяет состояние кнопочных переключателей и покажите на листинге программы, какие команды реализуют эту операцию.
- 6.4. Начертите упрощенную схему портов ввода/вывода микроконтроллера и поясните, с помощью каких команд осуществляется настройка портов.
- 6.5. Продемонстрируйте и поясните на примере разработанной схемы, каким образом в среде Atmel Studio осуществляется контроль правильности составленной программы.
- 6.6. Продемонстрируйте на примере разработанной схемы, как с помощью виртуального осциллографа можно проконтролировать формы сигналов в различных точках схемы.
- 6.7. Как с помощью виртуального осциллографа можно определить частоту следования импульсов?

Приложение А. Выполнение электрических соединений на схемах

Существует три способа выполнения электрических соединений: напрямую, посредством шин и через указатели.

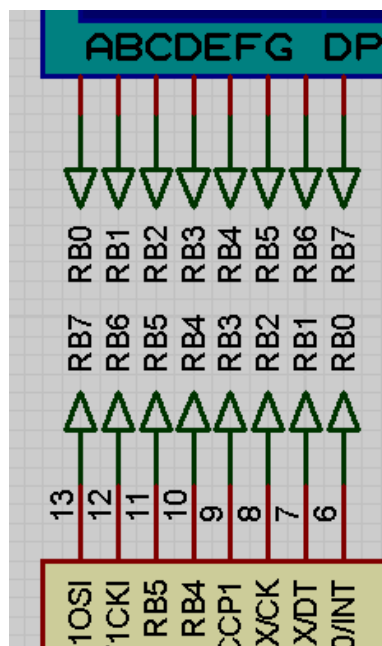
При **соединении напрямую** соединяемые выводы объединяются электрической линией. Для этого указатель мыши наводится на вывод до появления выделения. Выполняется щелчок левой кнопкой мыши (ЛКМ), отпускается кнопка мыши и проводится линия к другому выводу до появления выделения. Снова выполняется щелчок ЛКМ и соединение установлено. Если соединение не нужно, то следует щёлкнуть по соединению правой кнопкой мыши (ПКМ) и удалить связь. В случае необходимости объединения нескольких выводов проводится линия до предполагаемой точки объединения и двойным щелчком выполняется соединение, в результате чего образуется точка электрического соединения.

При **соединении посредством шины** нужно сначала на панели режима селекторов выбрать шину. Одинарным щелчком указывается начало шины. Затем выполняется протяжка шины. Одинарным щелчком задаются повороты, а двойным щелчком ЛКМ указывается конец шины. Шина может быть и разветвленной.

Затем выводы компонентов соединяются с шиной. Для этого указатель мыши нужно привести на вывод компонента до появления выделения, выполняется щелчок и протягивается линия до шины до появления выделения, снова выполняется щелчок для фиксации линии. Чтобы симулятор понимал, что с чем соединяется в шине, нужно именовать линии. Для этого следует щёлкнуть на линии ПКМ и выбрать в меню «Разместить метку связи» (Place Wire Label).

В открывшемся окне выбирается одно из имеющихся именований, либо вручную дописывается новое и нажимается ОК. Целесообразно именовать линии именами выводов микроконтроллера. В результате рядом с линией появляется именование, которое однозначно указывает её соединение в шине. Несколько выводов может быть объединено между собой; для этого линиям присваиваются одни и те же имена.

Третий способ соединения осуществляется с помощью указателей (терминалов). На левой вертикальной панели режима селекторов активировать Терминал. В селекторе объектов выбрать OUTPUT и щелчком ЛКМ на рабочем поле установить терминал. Терминал можно с помощью мыши перемещать в нужную точку и поворачивать в любом направлении. Затем терминал соединяется с нужной точкой схемы. Теперь нужно наименовать терминал. При двойном щелчке по терминалу открывается окно, в котором либо выбирается имеющиеся наименования, либо вручную дописывается новое и нажимается клавиша ОК. Точки схемы, которые должны быть соединены, должны иметь одноименные терминалы.



Приложение Б. Варианты заданий на лабораторную работу

1. Подключите к микроконтроллеру типа ATmega16 два кнопочных переключателя и два светодиода, один желтый, другой – зеленый. При нажатии первой кнопки мигает желтый светодиод, а при нажатии второй – мигает зеленый светодиод с частотой в два раза меньше частоты мигания желтого. Светодиоды при отпущенных кнопках не светятся. Предусмотреть схему сброса микроконтроллера от отдельной кнопки.

2. Подключите к микроконтроллеру типа ATmega16 один кнопочный переключатель и 8 светодиодов. При нажатии кнопки диоды должны поочередно загораться (бегущие огни). Светодиоды при отпущенной кнопке не должны светиться. Предусмотреть схему сброса микроконтроллера от отдельной кнопки.

3. Подключите к микроконтроллеру типа ATmega16 два кнопочных переключателя и 8 светодиодов, первые 4 – желтого цвета, другие 4 - красного. При нажатии первой кнопки светятся желтые светодиоды, а при нажатии второй – красные. При одновременном нажатии кнопок желтые и красные диоды должны поочередно загораться. Светодиоды при отпущенной кнопке не должны светиться. Предусмотреть схему сброса микроконтроллера от отдельной кнопки.

4. Подключите к микроконтроллеру типа ATmega16 один кнопочный переключатель и громкоговоритель (динамик). При нажатии кнопки должен мигать светодиод и гудеть динамик. Предусмотреть схему сброса микроконтроллера от отдельной кнопки.

Приложение В. Система команд микроконтроллеров AVR

Таблица В.1. Логические команды

Мнемоника	Описание	Операция	Флаги
AND Rd, Rr	«Логическое И» двух РОН	$Rd = Rd \cdot Rr$	Z,N,V
ANDI Rd, K *	«Логическое И» РОН и константы	$Rd = Rd \cdot K$	Z,N,V
EOR Rd, Rr	«Исключающее ИЛИ» двух РОН	$Rd = Rd \oplus Rr$	Z,N,V
OR Rd, Rr	«Логическое ИЛИ» двух РОН	$Rd = Rd \vee Rr$	Z,N,V
ORI Rd, K *	«Логическое ИЛИ» РОН и константы	$Rd = Rd \vee K$	Z,N,V
COM Rd	Перевод в обратный код	$Rd = \$FF - Rd$	Z,C,N,V
NEG Rd	Перевод в дополнительный код	$Rd = \$00 - Rd$	Z,C,N,V, H
CLR Rd	Сброс всех битов РОН	$Rd = Rd \oplus Rd$	Z,N,V
SER Rd *	Установка всех битов РОН	$Rd = \$FF$	-
TST Rd	Проверка РОН на отрицательное или нулевое значение	$Rd \cdot Rd$	Z,N,V
SWAP Rd	Обмен местами полубайтов в РОН	$Rd(3...0) = Rd(7...4),$ $Rd(7...4) = Rd(3...0)$	-
$0 \leq d \leq 31, 0 \leq r \leq 31, 0 \leq K \leq 255$ (* в командах ANDI, ORI и SER: $16 \leq d \leq 31$)			

Таблица В2. Команды передачи управления

Мнемоника	Описание	Операция	Флаги
RJMP k **	Относительный безусловный переход	$PC = PC + k + 1$	-
IJMP	Косвенный безусловный переход	$PC = Z$	-
JMP k ***	Абсолютный переход	$PC = k$	-
RCALL k **	Относительный вызов подпрограммы	$STACK = PC,$ $PC = PC + k + 1$	-
ICALL	Косвенный вызов подпрограммы	$STACK = PC, PC = Z$	-
CALL k ***	Абсолютный вызов подпрограммы	$STACK = PC, PC = k$	-
RET	Возврат из подпрограммы	$PC = STACK$	-

RETI	Возврат из п/п обработки прерывания	$PC = STACK$	I
CP Rd, Rr	Сравнение POH	$Rd - Rr$	Z,N,V,C, H
CPC Rd, Rr	Сравнение POH с учетом переноса	$Rd - Rr - C$	Z,N,V,C, H
CPI Rd, K *	Сравнение POH с константой	$Rd - K$	Z,N,V,C, H
CPSE Rd, Rr	Сравнение и пропуск следующей команды при равенстве	Если $Rd = Rr$, то $PC = PC + 2(3)$	-
SBRC Rr, b	Пропуск следующей команды, если бит POH сброшен	Если $Rr.b = 0$, то $PC = PC + 2(3)$	-
SBRS Rr, b	Пропуск следующей команды, если бит POH установлен	Если $Rr.b = 1$, то $PC = PC + 2(3)$	-
SBIC P, b	Пропуск следующей команды, если бит PVB сброшен	Если $P.b = 0$, то $PC = PC + 2(3)$	-
SBIS P, b	Пропуск следующей команды, если бит PVB установлен	Если $P.b = 1$, то $PC = PC + 2(3)$	-
BRBC s, k	Переход, если флаг s регистра SREG сброшен	Если $SREG.s = 0$, то $PC = PC + k + 1$	-

Таблица В3. Команды передачи управления (продолжение)

Мнемоника	Описание	Операция	Флаги
BRBS s, k	Переход, если флаг s регистра SREG установлен	Если $SREG.s = 1$, то $PC = PC + k + 1$	-
BRCS k	Переход по переносу	Если $C = 1$, то $PC = PC + k + 1$	-
BRCC k	Переход, если нет переноса	Если $C = 0$, то $PC = PC + k + 1$	-
BREQ k	Переход по «равно»	Если $Z = 1$, то $PC = PC + k + 1$	-
BRNE k	Переход по «не равно»	Если $Z = 0$, то $PC = PC + k + 1$	-
BRSH k	Переход по «больше или равно»	Если $C = 0$, то $PC = PC + k + 1$	-
BRLO k	Переход по «меньше»	Если $C = 1$, то $PC = PC + k + 1$	-
BRMI k	Переход по результату «отрицательное значение»	Если $N = 1$, то $PC = PC + k + 1$	-
BRPL k	Переход по результату «положительное значение»	Если $N = 0$, то $PC = PC + k + 1$	-

BRGE k	Переход по «больше или равно» (числа со знаком)	Если $(N \oplus V) = 0$, то $PC = PC + k + 1$	-
BRLT k	Переход по «меньше нуля» (числа со знаком)	Если $(N \oplus V) = 1$, то $PC = PC + k + 1$	-
BRHS k	Переход по половинному переносу	Если $H = 1$, то $PC = PC + k + 1$	-
BRHC k	Переход, если нет половинного переноса	Если $H = 0$, то $PC = PC + k + 1$	-
BRTS k	Переход, если флаг T установлен	Если $T = 1$, то $PC = PC + k + 1$	-
BRTC k	Переход, если флаг T сброшен	Если $T = 0$, то $PC = PC + k + 1$	-
BRVS k	Переход по переполнению дополнительного кода	Если $V = 1$, то $PC = PC + k + 1$	-
BRVC k	Переход, если нет переполнения дополнительного кода	Если $V = 0$, то $PC = PC + k + 1$	-
BRID k	Переход, если прерывания запрещены	Если $I = 0$, то $PC = PC + k + 1$	-
BRIE k	Переход, если прерывания разрешены	Если $I = 1$, то $PC = PC + k + 1$	-
$0 \leq d \leq 31, 0 \leq r \leq 31, 0 \leq b \leq 7, 0 \leq s \leq 7, 0 \leq K \leq 255, -64 \leq k \leq 63, 0 \leq P \leq 31$ (* в команде CPI: $16 \leq d \leq 31$; ** в командах RJMP, RCALL: $-2048 \leq k \leq +2047$; *** в командах JMP, CALL: $0 \leq k \leq 4M$)			

Таблица В4. Команды пересылки данных

Мнемоника	Описание	Операция	Флаги
MOV Rd, Rr	Пересылка между РОН	$Rd = Rr$	-
MOVW Rd, Rr	Пересылка 2-байтных значений	$Rd+l:Rd = Rr+l:Rr$	-
LDI Rd, K *	Загрузка константы в РОН	$Rd = K$	-
LD Rd, X	Косвенное чтение	$Rd = [X]$	-
LD Rd, X+	Косвенное чтение с постинкрементом	$Rd = [X], X = X + 1$	-
LD Rd, -X	Косвенное чтение с преддекрементом	$X = X - 1, Rd = [X]$	-
LD Rd, Y	Косвенное чтение	$Rd = [Y]$	-
LD Rd, Y+	Косвенное чтение с постинкрементом	$Rd = [Y], Y = Y + 1$	-
LD Rd, -Y	Косвенное чтение с преддекрементом	$Y = Y - 1, Rd = [Y]$	-
LDD Rd, Y+q	Косвенное относительное чтение	$Rd = [Y + q]$	-
LD Rd, Z	Косвенное чтение	$Rd = [Z]$	-
LD Rd, Z+	Косвенное чтение с постинкрементом	$Rd = [Z], Z = Z + 1$	-
LD Rd, -Z	Косвенное чтение с преддекрементом	$Z = Z - 1, Rd = [Z]$	-
LDD Rd, Z+q	Косвенное относительное чтение	$Rd = [Z + q]$	-
LDS Rd, k	Непосредственное чтение из ОЗУ	$Rd = [k]$	-
ST X, Rr	Косвенная запись	$[X] = Rr$	-
ST X+, Rr	Косвенная запись с постинкрементом	$[X] = Rr, X = X + 1$	-
ST -X, Rr	Косвенная запись с преддекрементом	$X = X - 1, [X] = Rr$	-
ST Y, Rr	Косвенная запись	$[Y] = Rr$	-
ST Y+, Rr	Косвенная запись с постинкрементом	$[Y] = Rr, Y = Y + 1$	-
ST -Y, Rr	Косвенная запись с преддекрементом	$Y = Y - 1, [X] = Rr$	-
STD Y+q, Rr	Косвенная относительная запись	$[Y + q] = Rr$	-
ST Z, Rr	Косвенная запись	$[Z] = Rr$	-
ST Z+, Rr	Косвенная запись с постинкрементом	$[Z] = Rr, Z = Z + 1$	-
ST -Z, Rr	Косвенная запись с преддекрементом	$Z = Z - 1, [Z] = Rr$	-

STD Z+q, Rr	Косвенная относительная запись	$[Z + q] = Rr$	-
STS k, Rr	Непосредственная запись в ОЗУ	$[k] = Rr$	-
LPM	Загрузка данных из памяти программ	$R0 = \{Z\}$	-
LPM Rd, Z	Загрузка данных из памяти программ	$Rd = \{Z\}$	-
LPM Rd, Z+	Загрузка данных из памяти программ	$Rd = \{Z\}, Z = Z + 1$	-
SPM	Запись в память программ	$\{Z\} = R1:R0$	-
IN Rd, P	Пересылка из PBB в POH	$Rd = P$	-
OUT P, Rr	Пересылка из POH в PBB	$P = Rr$	-
PUSH Rr	Сохранение байта в стеке	$STACK = Rr$	-
POP Rd	Извлечение байта из стека	$Rd = STACK$	-
$0 \leq d \leq 31, 0 \leq r \leq 31, 0 \leq K \leq 255, 0 \leq k \leq 65535, 0 \leq q \leq 63, 0 \leq P \leq 63$ (* в команде LDI: $16 \leq d \leq 31$); [] – содержимое памяти данных; { } – содержимое памяти команд			

Таблица В5. Команды битовых операций

Мнемоника	Описание	Операция	Флаги
CBR Rd, K *	Сброс бита(ов) POH	$Rd = Rd \cdot (\$FF - K)$	Z, N, V
SBR Rd, K *	Установка бита(ов) POH	$Rd = Rd \vee K$	Z, N, V
CBI P, b	Сброс бита PBB	$P.b = 0$	-
SBI P, b	Установка бита PBB	$P.b = 1$	-
BCLR s	Сброс флага	$SREG.s = 0$	SREG.s
BSET s	Установка флага	$SREG.s = 1$	SREG.s
BLD Rd, b	Загрузка бита POH из флага T (SREG)	$Rd.b = T$	-
BST Rr, b	Запись бита POH в флаг T (SREG)	$T = Rr.b$	T
CLC	Сброс флага переноса	$C = 0$	C
SEC	Установка флага переноса	$C = 1$	C
CLN	Сброс флага отрицательного числа	$N = 0$	N
SEN	Установка флага отрицательного числа	$N = 1$	N
CLZ	Сброс флага нуля	$Z = 0$	Z
SEZ	Установка флага нуля	$Z = 1$	Z
CLI	Общее запрещение прерываний	$I = 0$	I
SEI	Общее разрешение прерываний	$I = 1$	I
CLS	Сброс флага знака	$S = 0$	S
SES	Установка флага знака	$S = 1$	S
CLV	Сброс флага переполнения дополнительного кода	$V = 0$	V
SEV	Установка флага переполнения дополнительного кода	$V = 1$	V
CLT	Сброс флага T	$T = 0$	T
SET	Установка флага T	$T = 1$	T
CLH	Сброс флага половинного переноса	$H = 0$	H
SEN	Установка флага половинного переноса	$H = 1$	H
$0 \leq d \leq 31, 0 \leq r \leq 31, 0 \leq b \leq 7, 0 \leq s \leq 7, 0 \leq K \leq 255$ (* в командах CBR, SBR: $16 \leq d \leq 31$)			

ИССЛЕДОВАНИЕ ПРОЦЕДУР ПРОГРАММИРОВАНИЯ ВОСЬМИБИТОВЫХ ТАЙМЕРОВ/СЧЕТЧИКОВ И ОБСЛУЖИВАНИЯ ПРЕРЫВАНИЙ

1 Цель работы

Целью работы является изучение схем построения программируемых таймеров/счетчиков и системы прерывания однокристальных микро-ЭВМ, исследование процедур программирования таймеров/счетчиков, а также приобретение практических навыков программирования микроконтроллеров.

2 Краткие теоретические сведения

2.1 Прерывания в микроконтроллерах семейства ATmega

Прерывание прекращает нормальный ход программы для выполнения более приоритетной задачи, определяемой внутренним или внешним событием микроконтроллера. При возникновении прерывания микроконтроллер (МК) сохраняет в стеке содержимое счетчика команд PC и загружает в него адрес соответствующего вектора прерывания. По этому адресу, как правило, находится команда безусловного перехода к подпрограмме обработки прерывания. Последней командой подпрограммы обработки прерывания должна быть команда RETI, которая осуществляет возврат в основную программу и восстановление предварительно сохраненного счетчика команд.

Микроконтроллеры AVR семейства Mega имеют многоуровневую систему приоритетных прерываний. Младшие адреса памяти программ, начиная с адреса \$0002, отведены под таблицу векторов прерывания. Каждому прерыванию соответствует адрес в этой таблице, который загружается в счетчик команд при возникновении прерывания. Положение вектора в таблице также определяет и приоритет соответствующего прерывания: чем меньше адрес, тем выше приоритет прерывания. Размер вектора прерывания составляет 2 байта, а для перехода процессора к подпрограммам обработки прерываний используются команды JMP.

Положение таблицы векторов прерываний может быть изменено. Таблица может располагаться не только в начале памяти программ, но также и в начале области загрузчика, причем перемещение таблицы может быть осуществлено непосредственно в ходе выполнения программы. Для управления размещением таблицы прерываний используется регистр управления микроконтроллера MCUCR, расположенный по адресу \$35 (\$55). Здесь адрес в скобках относится к вектору прерывания при обращении к нему, как к ячейке ОЗУ (так как регистровый файл, блок регистров ввода/вывода и оперативная память образуют единое адресное пространство).

При размещении векторов прерываний в области загрузчика к значениям адресов, указанным в таблице, следует прибавить величину начального адреса области загрузчика. Таблица векторов прерываний ATmega128 приведена в приложении А. Если прерывания в работе микроконтроллера не предусматриваются, то на месте таблицы векторов прерываний может быть размещена часть основной программы.

Для глобального разрешения или запрещения прерываний предназначен флаг I статусного регистра SREG. Что разрешить прерывания флаг должен быть установлен в 1, а для запрещения — сброшен в 0. Индивидуальное разрешение или запрещение (маскирование) прерываний производится установкой или сбросом соответствующих битов регистров масок прерываний, рассматриваемых ниже.

При возникновении прерывания флаг I регистра состояния микроконтроллера SREG аппаратно сбрасывается, запрещая тем самым обработку следующих прерываний. Однако в подпрограмме обработки прерывания этот флаг можно снова установить в 1 для разрешения вложенных прерываний. При возврате из подпрограммы обработки прерывания (при выполнении команды RETI) флаг I автоматически устанавливается аппаратным способом.

Все имеющиеся прерывания можно разделить на два типа. Прерывания первого типа генерируются при наступлении некоторого события, в результате которого устанавливается флаг прерывания. Затем, если прерывание разрешено, в счетчик команд загружается адрес вектора соответствующего прерывания. При этом флаг прерывания аппаратно сбрасывается. Он также может быть сброшен программно, записью логической 1 в бит регистра, соответствующий этому флагу.

Прерывания второго типа не имеют флагов прерываний и генерируются в течение всего времени, пока присутствуют условия, необходимые для генерации прерывания. Соответственно, если условия, вызывающие прерывание, исчезнут до разрешения прерывания, генерации прерывания не произойдет.

Следует помнить, что при вызове подпрограмм обработки прерываний регистр состояния SREG не сохраняется автоматически. Поэтому пользователь должен самостоятельно запоминать содержимое этого регистра при входе в подпрограмму обработки прерывания (если это необходимо) и восстанавливать его значение перед вызовом команды RETI.

Микроконтроллеры семейства Mega поддерживают очередь прерываний, которая работает следующим образом: если условия генерации одного или более прерываний возникают в то время, когда флаг общего разрешения прерываний сброшен (все прерывания запрещены), соответствующие флаги устанавливаются в 1 и остаются в этом состоянии до установки флага общего разрешения прерываний. После разрешения прерываний выполняется их обработка в порядке приоритета.

Наименьшее время отклика для любого прерывания составляет 4 такта. В течение этого времени происходит сохранение счетчика команд в стеке. В течение последующих трех тактов выполняется команда перехода к подпрограмме

обработки прерывания. Если прерывание произойдет во время выполнения команды, длящейся несколько циклов, то генерация прерывания произойдет только после выполнения этой команды. Если же прерывание произойдет во время нахождения микроконтроллера в «спящем» режиме, то время отклика увеличивается еще на 4 или 5 тактов.

Возврат в основную программу занимает 4 такта, в течение которых происходит восстановление счетчика команд из стека. После выхода из прерывания процессор всегда выполняет одну команду основной программы, прежде чем обслужить любое отложенное прерывание.

2.2 Таймеры/счетчики микроконтроллеров AVR

Микроконтроллеры AVR, в зависимости от класса (Tiny, Classic, Mega) и типа модели, имеют в своем составе от одного до четырех таймеров/счетчиков общего назначения, обозначаемыми символами T0, T1, T2 и T3. Таймер T3 присутствует только в моделях ATmega 162x и ATmega 64x/128x.

Первый таймер T0 является 8-разрядным. Он имеется во всех моделях МК, может использоваться для отсчета и измерения временных интервалов или как счетчик внешних событий, а в некоторых моделях (напр. ATmega8515) еще и для сравнения с заданным значением. При переполнении счетного регистра таймера генерируется запрос на прерывание. Два других таймера (16-разрядный T1 и 8-разрядный T2) кроме уже названных имеют дополнительные функции. Оба таймера могут генерировать запрос на прерывание не только при переполнении счетного регистра, но и при наступлении ряда других событий. Они могут также использоваться в качестве широтно-импульсных модуляторов. Кроме того, таймер T2 может работать в асинхронном (относительно тактового сигнала микроконтроллера) режиме. 16-разрядный таймер/счетчик T3 по функциональным возможностям идентичен таймеру/счетчику T1.

Каждый таймер/счетчик использует один или более выводов (пинов) микроконтроллера. Эти выводы могут быть либо линиями портов ввода/вывода с альтернативной функцией, либо выделенными выводами микроконтроллера. Основные выводы некоторых микроконтроллеров, относящиеся к таймерам/счетчикам, приведены в таблице 2.1. Для выяснения назначения выводов других типов микроконтроллеров следует воспользоваться справочным листком на соответствующий контроллер.

Упрощенная структурная схема таймера/счетчика изображена на рисунке 2.1. В его состав входит основной двоичный счетчик TCNT0, регистр управления TCCR0 (*Timer/Counter Control Register*) и схема управления. Кроме этого, в таймере/счетчике используется один разряд регистра флагов запросов прерываний TIFR (*Timer/Counter0 Interrupt Flag Register*) и один разряд регистра маски прерываний TIMSK (*Timer/Counter Interrupt Mask Register*).

Таблица 2.1. Выводы МК, относящиеся к таймерам

Название вывода	Тип микроконтроллера			Описание выводов
	ATx8515	AT-mega16	AT-mega128	
T0	PB0	PB0	PD7 (T2)	Вход внешнего сигнала T0
T1	PB1	PB1	PD6	Вход внешнего сигнала T1
ICP	ICP	PD6	PD4	Вход захвата таймера T1
OC0	—	PB3	PB4	Выход схемы сравнения T0
OC1A	PD5	PD5	PB5	Выход схемы сравнения T1
OC1B	—	PD4	PB6	Выход схемы сравнения T1
OC1C	—	—	PB7	

Счетчик TCNT0 доступен в любой момент времени как для чтения, так и для записи. При переходе TCNT0 из состояния \$FF в состояние \$00 устанавливается в 1 флаг переполнения TOV0 (*Timer Overflow*) в регистре TIFR и генерируется запрос на прерывание. Разрешение прерывания выполняется установкой в 1 разряда TOIE0 (*Timer Overflow Interrupt Enable*) регистра маски TIMSK. Естественно, что флаг общего разрешения прерывания I регистра состояния SREG микроконтроллера также должен быть установлен в 1.

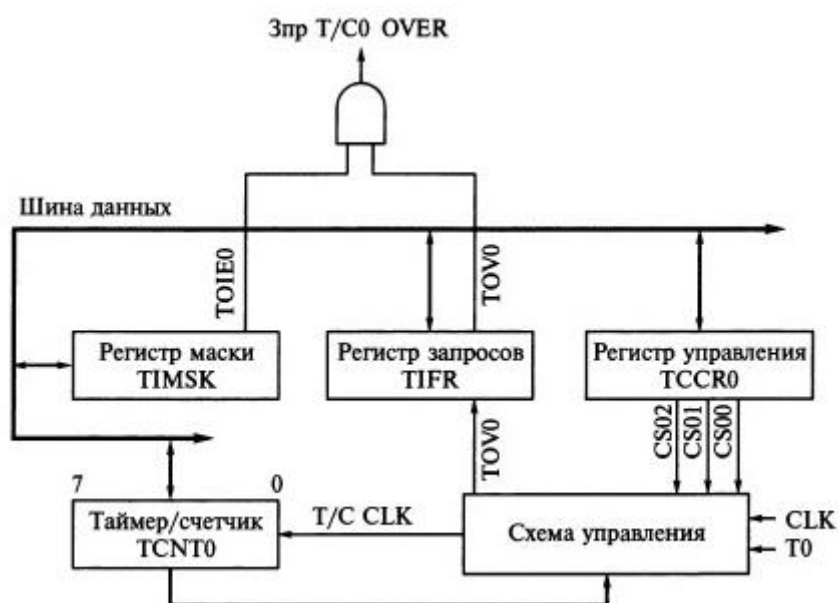


Рисунок 2.1 – Схема таймера/счетчика T0

Таймер/счетчик T0 может функционировать в двух режимах:

1) таймера — в этом режиме на вход поступают импульсы тактового генератора микроконтроллера CLK (непосредственно или через предделитель схемы управления);

2) счетчика внешних событий — в этом режиме инкремент содержимого счетчика производится по активному фронту сигнала на входе T0 микроконтроллера.

Выбор источника и частоты счетных сигналов, а также запуск и останов таймера/счетчика осуществляется с помощью битов CS00 – CS02 (*Clock Select*) регистра управления таймером TCCR0 (таблица 2.2).

Таблица 2.2. Задание коэффициента деления тактовой частоты

CS12	CS11	CS10	Описание
0	0	0	Нет источника (таймер выключен)
0	0	1	CLK
0	1	0	CLK/8
0	1	1	CLK/64
1	0	0	CLK/256
1	0	1	CLK/1024
1	1	0	Внешний источник тактирования на T1. Падающий фронт
1	1	1	Внешний источник тактирования на T1. Растущий фронт

Таймер-счетчик T0 соединен с двумя внешними выводами микроконтроллера: T0 и OC0 (*Output Compare*). Пин T0 является для МК входом внешнего тактового сигнала. В МК типа ATmega16 этот пин совмещен с выводом PB0 порта В. Это позволяет использовать т/с T0 для подсчета внешних импульсов. Вывод (пин) OC0 (PB3) — это выход схемы сравнения таймера-счетчика. На этом выводе с помощью таймера может формироваться прямоугольный импульс типа «меандр» (длительность 1 равна длительности 0) или сигнал с широтно-импульсной модуляцией ШИМ. Этот вывод также может менять свое состояние при срабатывании схемы сравнения.

Шестнадцатиразрядный таймер/счетчик (т/с) T1 выполняет значительно больше функций, чем T0. Таймер/счетчик T1, также как и т/с T0, может использоваться для формирования временных интервалов или для подсчета внешних событий по входу T1. Кроме этого, т/с T1 может по внешнему сигналу «Захват» сохранять свое текущее состояние в отдельном регистре ввода/вывода, а также выполнять определенные действия при равенстве содержимого счетчика и заданного значения. И, наконец, он может работать как широтно-импульсный модулятор (ШИМ).

Упрощенная структурная схема таймера/счетчика T1 микроконтроллера показана на рисунке 2.2. В состав таймера/счетчика входят базовый 16-разрядный счетчик TCNT1, три 16-разрядных регистра: регистр захвата ICR1 и два регистра сравнения OCR1A и OCR1B; два 16-разрядных компаратора (схемы сравнения) и два 8-разрядных управляющих регистра TCCR1A и TCCR1B, а также блок управления таймером.

Все флаги состояния таймера/счетчика (переполнения, совпадения и захвата) располагаются в регистре флагов прерываний от таймера TIFR, а

разрешение (или запрещение) прерываний от таймера выполняется установкой соответствующих разрядов регистра маски TIMSK.

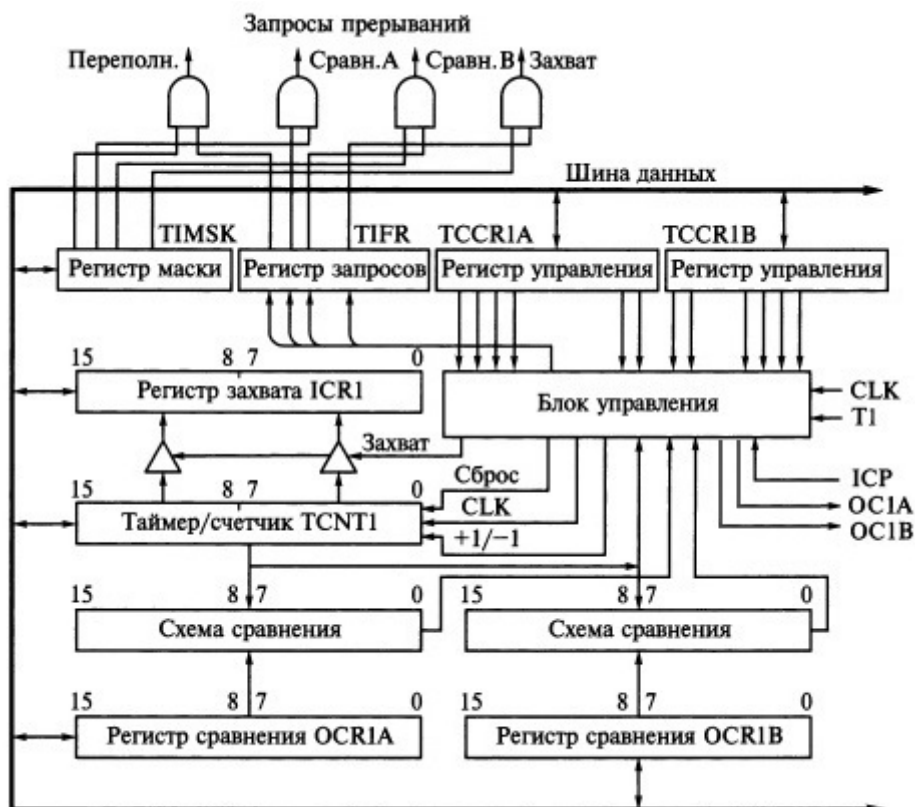


Рисунок 2.2 – Схема таймера/счетчика T1

Физически таймер TCNT1 состоит из двух частей TCN1H:TCN1L. В связи с тем, что установка обоих счетчиков в начальное состояние или чтение их текущего состояния должны осуществляться одновременно, а запись или чтение их происходит побайтно, то для согласования этих процедур введен регистр временного хранения, который программно недоступен. При инициализации счетчика первым должен записываться старший байт в регистр TCN1H. А при чтении его текущего состояния первым считывается младший байт (регистр TCN1L), а затем старший — регистр TCN1H.

Регистры сравнения OCRn (*Output Compare Register*) входят в состав блоков сравнения модуля. Во время работы таймера/счетчика производится непрерывное (в каждом такте) сравнение этих регистров с регистром TCNTn. В случае равенства содержимого этих регистров в следующем такте устанавливается флаг наличия совпадения OSn в соответствующем регистре флагов и генерируется прерывание (если оно разрешено). Кроме того, при наступлении этого события может изменяться состояние вывода OSn микроконтроллера. Чтобы таймер/счетчик мог управлять состоянием этих выводов, они должны быть сконфигурированы как выходы (соответствующий бит регистра DDRx должен быть установлен в 1). Во время процедуры записи в счетный регистр блокируется

(запрещается) формирование сигнала о совпадении, если оно произойдет в следующем такте.

Шестнадцатиразрядные таймеры в большинстве типов микроконтроллеров имеют возможность самостоятельно сбрасывать счётный регистр сразу после его совпадения с регистром сравнения, после чего счёт автоматически продолжается с нулевого значения. Такой режим называется режимом сброса при совпадении (СТС) и может быть использован, в частности, для генерации сигналов фиксированной частоты. Для перевода таймера в данный режим необходимо установить биты WGMn2 и WGMn3 (*Wave Generator Mode*) регистра TCCRnB.

Шестнадцатиразрядные таймеры/счетчики также соединены с внешними выводами микроконтроллера T1 и T3, а их схемы совпадения соответственно с выводами OC2 и OC3. Некоторые типы МК, например ATmega168, имеют несколько схем сравнения, обозначаемыми символами A, B и C, и, соответственно, такое же количество выводов схем сравнения (OCxA, OCxB, OCxC). Здесь символ x принимает значение 1 или 3, соответствующее таймеру T1 или T3.

Таймер/счетчик T1 тоже может использоваться для отсчета временных интервалов и как счетчик внешних событий. Кроме того, он может осуществлять запоминание своего состояния по внешнему сигналу захвата ICP (*Input Capture Pin*). Как и таймер/счетчик T0, он может работать в качестве 2-х или 3-х канального широтно-импульсного модулятора, но уже переменной разрядности.

Таймер/счетчик T2 практически полностью аналогичен таймеру/счетчику T0, за исключением того, что он может использоваться также как счетчик внешних событий, и в нем отсутствует возможность работы в асинхронном режиме. Таймер/счетчик T3 по функциональным возможностям идентичен таймеру/счетчику T1.

В составе микроконтроллера имеется также сторожевой таймер, являющийся неизменяемым атрибутом всех современных микроконтроллеров. Этот таймер позволяет избежать несанкционированного закливания программы, возникающего по тем или иным причинам.

2.3 Прерывания от таймеров/счетчиков

В микроконтроллерах семейства ATmega имеется регистр TIFR, содержащий флаги прерываний. Микроконтроллер ATmega128 содержит дополнительный расширенный регистр флагов прерываний ETIFR. Для разрешения или запрещения прерываний от таймеров/счетчиков используются регистры ввода/вывода (регистры маскирования): TIMSK (и ETIMSK). Форматы регистров, используемых для разрешения/запрещения прерываний от таймеров/счетчиков, показаны на рисунке 2.3, а назначение их битов приведено в таблице 2.3.

Для разрешения какого-либо прерывания от таймера/счетчика необходимо установить в 1 соответствующий бит регистра TIMSK (ETIMSK) и флаг разрешения I прерывания регистра SREG.

Форматы регистров, используемых для сигнализации поступления прерываний от таймеров/счетчиков, показаны на рисунке 2.4, а назначение их битов приведено в таблице 2.4.

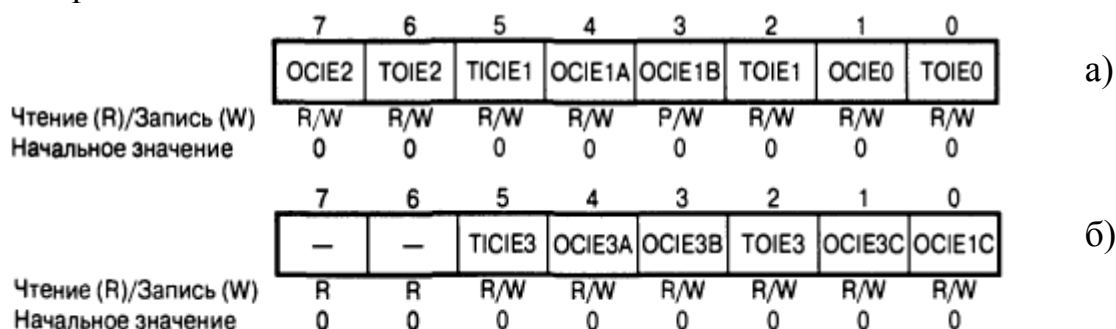


Рисунок 2.3 - Формат регистров TIMSK (а) и ETIMSK (б)

Таблица 2.3. Биты регистров TIMSK, ETIMSK

Название бита	Описание
TOIE _n	Флаг разрешения прерывания по переполнению таймера/счетчика T _n
OCIE _n	Флаг разрешения прерывания по событию «Совпадение» таймера/счетчика T _n
OCIE _{nA}	Флаг разрешения прерывания по событию «Совпадение А» таймера/счетчика T _n
OCIE _{nB}	Флаг разрешения прерывания по событию «Совпадение В» таймера/счетчика T _n
OCIE _{nC}	Флаг разрешения прерывания по событию «Совпадение С» таймера/счетчика T _n
TICIE _n	Флаг разрешения прерывания по событию «Захват» таймера/счетчика T _n

При наступлении какого-либо события соответствующий флаг регистра TIFR (ETIFR) устанавливается в 1. При запуске подпрограммы обработки прерывания он аппаратно сбрасывается в 0. Любой флаг может быть также сброшен программно, записью в него лог. 0.

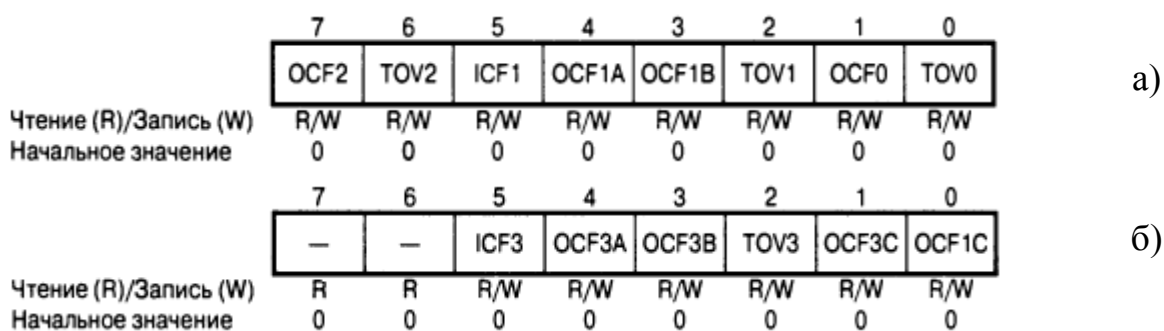


Рисунок 2.4 - Формат регистров TIFR (а) и ETIFR (б)

Таблица 2.4. Биты регистров TIFR и ETIFR

Название бита	Описание
TOVn	Флаг прерывания по переполнению таймера/счетчика Tn
OCFn	Флаг прерывания по событию «Совпадение» таймера/счетчика Tn
OCFnA	Флаг прерывания по событию «Совпадение А» таймера/счетчика Tn
OCFnB	Флаг прерывания по событию «Совпадение В» таймера/счетчика Tn
OCFnC	Флаг прерывания по событию «Совпадение С» таймера/счетчика Tn
ICFn	Флаг прерывания по событию «Захват» таймера/счетчика Tn

Регистры TCCRn предназначены для управления модулем таймера/счетчика. Форматы этих регистров приведены на Рис. 2.5, а описание их битов — соответственно в табл. 2.5.

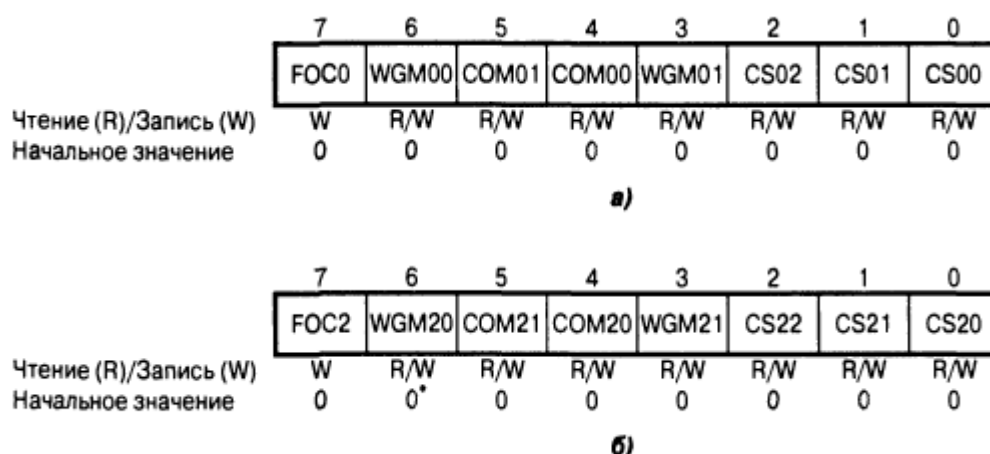


Рисунок 2.5. Формат регистров TCCR0 (а) и TCCR2 (б)

Таблица 2.5. Биты регистра TCCR0 (TCCR2)

Бит	Название	Описание
7	FOCn	Принудительное изменение состояния вывода OCn (режимы Normal и CTC). При записи лог. 1 в этот бит состояние вывода OCn изменяется в соответствии с установками битов COMn1:COMn0. Прерывание при этом не генерируется и сброс таймера (в режиме CTC) не производится. В режимах Fast PWM и Phase Correct PWM этот бит должен быть сброшен в 0. При чтении бита всегда возвращается 0.
6, 3	WGMn1 WGMn0	Режим работы таймера/счетчика. Эти биты определяют режим работы таймера/счетчика в соответствии с Табл. 3.6.
5, 4	COMn1 COMn0	Режим работы блока сравнения. Эти биты определяют поведение вывода OCn при наступлении события «Совпадение».

		Влияние содержимого этих битов на состояние вывода зависит от режима работы таймера/счетчика
2...0	CSn2... CSn0	Управление тактовым сигналом. Эти биты определяют источник тактового сигнала таймера/счетчика Действие этих битов зависит от исполнения таймера/счетчика и будет описано ниже

Режим работы таймера/счетчика T0 (T2) определяется состоянием битов WGMn2:WGMn0 регистра TCCRn. Зависимость режима работы таймеров/счетчиков от состояния этих битов показана в табл. 2.6.

Режим Normal — это наиболее простой режим работы таймеров/счетчиков. В режиме Normal счетный регистр функционирует как обычный суммирующий счетчик. По каждому импульсу тактового сигнала CLK_{Tn} осуществляется инкрементирование счетного регистра. При переходе через значение \$FF возникает переполнение, и счет продолжается со значения \$00. В том же такте сигнала CLK_{Tn}, в котором обнуляется регистр TCNTn, флаг прерывания по переполнению TOVn устанавливается в 1.

При равенстве счетного регистра и регистра сравнения устанавливается соответствующий флаг прерывания OCFn и, если бит OCIE_n регистра маски установлен в 1, генерируется прерывание. Наряду с установкой флага при равенстве счетного регистра и регистра сравнения может изменяться состояние вывода OCn микроконтроллера. Каким образом оно будет изменяться, определяется битами COMn1:COMn0 регистра управления TCCRn в соответствии с табл. 2.7.

Таблица 2.6. Режимы работы таймера/счетчика T0

Номер режима	WGMn1	WGMn2	Название режима
0	0	0	Normal
1	0	1	Phase Correct PWM
2	1	0	CTC (сброс при совпадении)
3	1	1	Fast PWM

При необходимости состояние вывода OCn может быть изменено принудительно записью лог. 1 в бит FOCn соответствующего регистра управления. Прерывание при этом не генерируется.

Таблица 2.7. Управление выводами OCn в режиме Normal

COMn1	COMn0	Описание
0	0	Таймер/счетчик Tn отключен от вывода OCn
0	1	Состояние вывода меняется на противоположное
1	0	Вывод сбрасывается в 0
1	1	Вывод устанавливается в 1

В режиме **СТС** (*Clear Timer on Compare Match*) Сброс при совпадении счетный регистр тоже функционирует как обычный суммирующий счетчик, инкрементирование которого осуществляется по каждому импульсу тактового сигнала CLK_{Tn} . Однако максимально возможное значение счетного регистра и, следовательно, разрешающая способность счетчика определяются регистром сравнения OCR_n . После достижения значения, записанного в регистре сравнения, счет продолжается со значения \$00. Если в регистре сравнения записано \$FF, то в том же такте сигнала CLK_{Tn} , в котором обнуляется счетный регистр, устанавливается флаг прерывания по переполнению TOV_n в соответствующем регистре флагов.

При достижении счетчиком максимального значения устанавливается флаг OCF_n , и, если бит $OCIE_n$ соответствующего регистра маски установлен в 1, генерируется прерывание. Одновременно с установкой флага может изменяться состояние выводов OC_n микроконтроллера. Состояние выводов определяется битами $COM_n1:COM_n0$ регистра управления $TCCR$, как указано в табл. 2.7.

Для генерации сигнала заданной частоты необходимо записать в биты $COM_n1:COM_n0$ значение 01 (переключение вывода с поступлением каждого сигнала). Частота генерируемого сигнала будет определяться выражением

$$f_{OCn} = \frac{f_{clk_IO}}{2 \cdot N \cdot (1 + OCR_{nx})},$$

где N — коэффициент деления предделителя.

При необходимости состояние вывода OC_n можно изменить принудительно записью логической 1 в бит FOC_n соответствующего регистра управления. Прерывание при этом не генерируется и сброса счетного регистра не производится.

16-разрядный таймер/счетчик $T1$ управляется двумя регистрами $TCCR1A$ и $TCCR1B$. Формат управляющего слова регистра $TCCR1A$ показан на рисунке 2.6.

Бит	7	6	5	4	3	2	1	0
\$2F (\$4F)	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10
Чтение/ Запись	R/W	R/W	R/W	R/W	W	W	R/W	R/W
Исх. значение	0	0	0	0	0	0	0	0

Рисунок 2.6 - Формат управляющего слова регистра **TCCR1A**

Биты 7, 6 – **COM1A1, COM1A0** (*Compare Output Mode A*) – Режим работы блока сравнения А. Эти биты определяют поведение вывода $OC1A$ при совпадении таймера/счетчика 1 с регистром $OCR1A$. В зависимости от режима работы

таймера/счетчика 1 состояние вывода изменяется аналогично таймеру/счетчику 0 в соответствии с таблицей 2.7.

Биты 5, 4 – **COM1B1, COM1B0** (*Compare Output Mode B* – Режим работы блока сравнения В). Эти биты определяют поведение вывода OC1B при совпадении таймера/счетчика 1 с регистром OCR1B. В зависимости от режима работы таймера/счетчика 1 состояние вывода изменяется аналогично таймеру/счетчику 0 в соответствии с таблицей 2.7.

Биты 3, 2 – **FOC1A, FOC1B** (*Force Output Compare*)– Принудительное изменение состояния выводов OC1A и OC1B. При записи в эти разряды логической единицы состояние выводов OC1A и OC1B изменяется в соответствии с установками разрядов COM1A1:COM1A0 и COM1B1:COM1B0. Прерывание при этом не генерируется и сброс таймера (в режиме CTC) не производится. Эта функция доступна только в режимах, которые не используются для генерации сигнала с ШИМ. При чтении разрядов всегда возвращается «0».

Биты 1, 0 – **WGM11, WGM10** (*Waveform Generation Mode*) – Режим работы таймера/счетчика. Данные биты совместно с битами WGM13 и WGM12 регистра TCCR1B определяют режим работы таймера/счетчика. При нулевых значениях этих бит устанавливается нормальный режим работы таймера/счетчика. Если WGM12=1, а остальные биты равны нулю, то устанавливается режим CTC (Сброс при совпадении). При необходимости задания других режимов следует обратиться к справочному листку на соответствующий микроконтроллер.

Формат управляющего слова таймера/счетчика T1 – TCCR1B показан на рисунке 2.7.

Бит	7	6	5	4	3	2	1	0
\$2E(\$4E)	ICNC1	ICES1	—	WGM13	WGM12	CS12	CS11	CS10
Чтение/Запись	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Исходное значение	0	0	0	0	0	0	0	0

Рисунок 2.7 - Формат управляющего слова регистра TCCR1B

Бит 7 – **ICNC1** (*Input Capture Noise Canceler*) – Установка режима подавления шума на входе захвата. При сброшенном в состояние 0 бите ICNC1 функция подавления шума входного триггера захвата запрещена. Вход захвата переключается по первому нарастающему/падающему фронту, поступившему на вывод входа захвата ICP. При установленном в состояние 1 бите ICNC1 выполняются четыре последовательных опроса состояния вывода ICP и все четыре выборки должны иметь одинаковый (высокий/низкий), определяемый битом ICES1, уровень. Частота опроса соответствует частоте тактового сигнала.

Бит 6 – **ICES1** (*Input Capture Edge Select*) – Выбор фронта срабатывания на входе захвата. При сброшенном в состояние 0 бите ICES1 содержимое таймера/счетчика 1 по падающему фронту на выводе входа захвата ICP пересылается в регистр входного захвата ICR1. При установленном в 1 бите ICES1

содержимое таймера/счетчика 1 пересылается в регистр входного захвата ICR1 по нарастающему фронту на выводе входа захвата ICP.

Бит 5 – **Res** (*Reserved Bit*) – Зарезервированный бит. Данный бит в микроконтроллерах ATmega зарезервирован и при считывании всегда покажет состояние 0.

Биты 4, 3 – **WGM13, WGM12** (*Waveform Generation Mode*) – Режим работы таймера/счетчика. Совместно с разрядами WGM11, WGM10 регистра TCCR1A определяют режим работы таймера/счетчика 1.

Биты 2, 1, 0 – **CS12, CS11, CS10** (*Clock Select*) – Выбор тактовой частоты. Эти биты позволяют выбрать источник тактового сигнала для таймера/счетчика 1 (см. таблицу 2.2). Если таймер/счетчик 1 используется как счетчик, то вывод T1 конфигурируется как вход.

В режиме Normal таймер/счетчик 1 функционирует как обычный суммирующий счетчик. При переходе через значение \$FFFF возникает переполнение, устанавливается флаг прерывания по переполнению TOV1, и счет продолжается со значения \$0000. Оба блока сравнения таймера в этом режиме могут использоваться как для генерации прерываний, так и для формирования сигналов. Состояние выходов OC1A и OC1B каждого из блоков сравнения определяется содержанием битов COM1A1, COM1A0 и COM1B1, COM1B0 регистра TCCR1A аналогично таймеру/счетчику 0 в соответствии с таблицей 2.7.

В режиме CTC (сброс при совпадении) счетный регистр тоже функционирует как обычный суммирующий счетчик. Однако максимально возможное значение счетного регистра и, следовательно, разрешающая способность счетчика определяются либо регистром сравнения OCR1A, либо регистром захвата ICR1. После достижения максимального значения счет продолжается со значения \$0000. Флаг прерывания TOV1 устанавливается при изменении значения счетного регистра с \$FFFF на \$0000. При достижении счетчиком максимального значения устанавливается флаг OCF1A, если модуль счета определяется регистром сравнения OCR1A, или ICF1, если модуль счета определяется регистром захвата ICR1. Одновременно с установкой соответствующих флагов может изменяться состояние выводов OC1A и OC1B микроконтроллера (таблицей 2.7).

3 Описание лабораторного стенда

В состав лабораторного стенда входит персональный компьютер с установленными отладочными системами Atmel Studio версии 6 и выше и Proteus 8 (демоверсия). Описание этих систем и особенности работы с ними приведены в методических указаниях к предыдущим лабораторным работам.

4 Программа работы

4.1. Изучить устройство 8-ми и 16-ти разрядных таймеров/счетчиков микроконтроллеров семейства AVR и особенности их программирования.

- 4.2. Составить схему устройства на основе микроконтроллера типа ATmega, состоящую из нескольких дискретных датчиков (ключей) и светодиодов. Тип микроконтроллера, наличие и количество датчиков и светодиодов определяется вариантом задания.
- 4.3. Разработать схему управления устройством зажигания светодиодов с использованием прерываний от таймера. Алгоритм работы устройства определяется вариантом задания.
- 4.4. Выполнить отладку программы управления устройством в среде Atmel Studio.
- 4.5. Составить схему устройства, произвести запись программы управления в память микроконтроллера и исследовать функционирование разработанной схемы устройства в среде Proteus.
- 4.6. Измерить частоту мигания при изменении коэффициента деления тактовой частоты (биты CS10 – CS12) от 1 до 1024.
- 4.7. Составить отчет и сформулировать выводы по результатам лабораторных исследований.

Примечание: Пп. 4.1-4.3 программы работы выполняются в процессе домашней подготовки.

5 Методические рекомендации по выполнению работы

- 5.1. Все функции опроса датчиков и управления светодиодами (формирование бегущих огней) должна выполнять процедура обработки прерываний от таймера. При каждом вызове прерывания осуществляется сдвиг «огней» свечения светодиодов.
- 5.2. Для имитации бегущих огней целесообразно использовать операции сдвига.
- 5.3. Программа управления устройством создается на основе двух самостоятельных алгоритмов: алгоритма основной программы и алгоритма процедуры обработки прерывания.
- 5.4. Алгоритм основной программы должен содержать следующие действия:
 - Настроить стек и порты ввода/вывода микроконтроллера.
 - Настроить таймер и систему прерываний. Рекомендуется использовать режим таймера «Сброс при совпадении», при котором таймер периодически вырабатывает запросы на прерывание с заданным периодом.
 - Задать коэффициент деления тактовой частоты.
 - Записать в рабочий регистр слово управления светодиодами в начальном состоянии.
 - Разрешить работу таймера.
 - Разрешить прерывания.
 - Перейти к выполнению основного цикла программы. В связи с тем, что все операции опроса датчиков и управления свечением осуществляются процедурой обработки прерываний, то в основном цикле не нужно

выполнять никаких действий, т.е. его можно оставить пустым. Процессор при этом находится в состоянии ожидания прерывания от таймера.

5.5 Алгоритм процедуры обработки прерываний.

- Проверить состояние датчиков.
- Если контакты датчика разомкнуты (или, если контакты одного датчика разомкнуты, а второго замкнуты), произвести сдвиг всех разрядов рабочего регистра на один (или два) разряд(а) вправо. Если в результате этого сдвига единичный бит выйдет за пределы байта, сформировать новый единичный бит в крайней левой позиции.
- Если контакты датчика разомкнуты (или датчики поменяли свое состояние на обратное), выполнить сдвиг всех разрядов рабочего регистра на один (два) разряд(а) влево. При выходе единичного бита за пределы байта сформировать новый единичный бит в крайней правой позиции.
- Вывести содержимое рабочего регистра в порт PB.
- Закончить процедуру обработки прерывания.

6 Содержание отчета

- 6.1 Цель и программа работы.
- 6.2 Структурные схемы таймеров.
- 6.3 Тексты программ с комментариями на ассемблере.
- 6.4 Выводы по результатам исследований.

7 Контрольные вопросы

- 7.1. С какой целью в состав микроконтроллеров введены таймеры/счетчики?
- 7.2. Приведите общую характеристику таймеров/счетчиков, входящих в состав программируемых микроконтроллеров семейства AVR.
- 7.3. Расскажите о составе и функционировании 8-разрядного таймера/счетчика T0.
- 7.4. Расскажите о составе и функционировании 16-разрядного таймера/счетчика T1.
- 7.5. Какие сигналы прерывания используются при работе с таймерами/счетчиками?
- 7.6. Как можно, используя прерывания от таймера T0, сформировать прямоугольную последовательность заданной частоты?
- 7.7. Какие действия совершаются в микроконтроллере при поступлении сигнала прерывания?
- 7.8. Как можно изменять задержку, формируемую таймером / счетчиком?
- 7.9. Что такое «захват таймера», как он происходит и с какой целью применяется?
- 7.10. Что такое «Широтно-импульсная модуляция» и в каких случаях она применяется?

- 7.11. Для чего в схеме тактового генератора используется кварцевый резонатор?
- 7.12. Каким логическим уровнем (нулем или единицей) целесообразно зажигать светодиод и почему?
- 7.13. Как влияет коэффициент деления тактовой частоты на частоту мигания индикаторов?
- 7.14. Как на практике измерить частоту колебаний, генерируемых микроконтроллером?

Приложение А. Варианты заданий

№ Вар.	Тип микро-контроллера	Кол-во датчиков	Кол-во свето-диодов	Алгоритм работы
1	ATmega8535	-	8	При включении устройства светящиеся диоды перемещаются от старшего разряда к младшему
2	ATmega8	1	8	При сигнале датчика 0 светящиеся диоды перемещаются от старшего разряда к младшему, при сигнале 1 – от младшего к старшему
3	ATmega8535	2	8	При включении первого датчика светящиеся диоды перемещаются от старшего разряда к младшему, при включении второго – от младшего к старшему. При выключенном состоянии датчиков свечение диодов отсутствует.
4	ATmega8535	1	4	При включении датчика мигают одновременно все светодиоды, при отключенном состоянии диоды не светятся.
5	ATmega16	2	8	При включении первого датчика светящиеся диоды перемещаются от младшего разряда к старшему, при включении второго – от старшего к младшему. При одновременном включении датчиков происходит мигание всех диодов одновременно, а при выключенном состоянии датчиков свечение диодов отсутствует.

Приложение Б. Таблица размещения векторов прерываний

Источник	Описание	№	Адрес
INT0	Внешнее прерывание 0	1	\$0002
INT1	Внешнее прерывание 1	2	\$0004
INT2	Внешнее прерывание 2	3	\$0006
INT3	Внешнее прерывание 3	4	\$0008
INT4	Внешнее прерывание 4	5	\$000A
INT5	Внешнее прерывание 5	6	\$000C
INT6	Внешнее прерывание 6	7	\$000E
INT7	Внешнее прерывание 7	8	\$0010
TIMER2COMP	Совпадение таймера/счетчика T2	9	\$0012
TIMER2 OVF	Переполнение т/с T2	10	\$0014
TIMER1 CAPT	Захват таймера/счетчика T1	11	\$0016
TIMER1 COMPA	Совпадение А т/с T1	12	\$0018
TIMER1 COMPB	Совпадение В т/с T1	13	\$001A
TIMER1 OVF	Переполнение т/с T1	14	\$001C
TIMER0COMP	Совпадение таймера/счетчика T0	15	\$001E
TIMER0 OVF	Переполнение т/с T0	16	\$0020
SPI, STC	Передача по SPI завершена	17	\$0022
USART0, RX	USART0, прием завершен	18	\$0024
USART0, UDRE	Регистр данных USART0 пуст	19	\$0026
USART0, TX	USART0, передача завершена	20	\$0028
ADC	Преобразование АЦП завершено	21	\$002A
EE_RDY	EEPROM готово	22	\$002C
ANA_COMP	Аналоговый компаратор	23	\$002E
TIMER1 COMPC	Совпадение С т/с T1	24	\$0030
TIMER3CAPT	Захват таймера/счетчика T3	25	\$0032
TIMER3 COMPA	Совпадение А т/с T3	26	\$0034
TIMER3 COMPB	Совпадение В т/с T3	27	\$0036
TIMER3 COMPC	Совпадение С т/с T3	28	\$0038
TIMER3 OVF	Переполнение таймера/счетчика T3	29	\$003A
USART1,RX	USART1, прием завершен	30	\$003C
USART1,UDRE	Регистр данных USART1 пуст	31	\$003E
USART1,TX	USART1, передача завершена	32	\$0040
TWI	Прерывание от модуля TWI	33	\$0042
SPM_RDY	Готовность SPM	34	\$0044

Приложение В.

Пример программы управления бегущими огнями с использованием прерывания от таймера

```
.include "m8def.inc" ; Присоединение файла описания
.def temp=r16        ; Определение главного регистра
.def rab=r17         ; Определение раб. регистра для команд сдвига
.equ kdel=780        ; Задание длительности задержки
; ----- Резервирование ячеек памяти -----
.dseg                ; выбор сегмента ОЗУ
.org 0x60             ; установка текущего адреса сегмента
Buf:
.byte 1               ; выделение одного байта для хранения рабочего регистра
;
; ----- начало программного кода -----
.cseg                ; Выбор сегмента программного кода
.org 0                ; Установка текущ адреса на ноль

; ----- Переопределение векторов прерывания -----
Start:
Rjmp init             ; 0x00 переход на начало программы (перепрыгивание
                      ; векторов прерывания)
Reti                  ; 0x01 внешнее прерывание 0
Reti                  ; 0x02 внешнее прерывание 1
Reti                  ; 0x03 прерывание таймер 2, Compare
Reti                  ; 0x04 прерывание таймер 2 OVF
Reti                  ; 0x05 таймер/счетчик 1, TI CAPT
Rjmp prtim1           ; 0x06 т аймер/счетчик 1, совпадение. Канал А
Reti                  ; 0x07 таймер/счетчик 1, совпадение. Канал В
Reti                  ; 0x08 таймер/счетчик 1, OVF прерывание по переполнению
Reti                  ;
Reti                  ;
Reti                  ;
Reti                  ;
Reti                  ;
Reti                  ;
Reti                  ;
Reti                  ; Reti -команда-заглушка, возврат из прерывания

; ----- Модуль инициализации -----
Init:
; ----- Инициализация стека -----
Ldi temp,Low(RAMEND)
Out SPL,temp
Ldi temp,High(RAMEND)
Out SPH,temp
; ----- Инициализация портов ВВ -----
Ldi temp, 0
Out DDRD,temp ; порт D на ввод
Ldi temp,0xFF
Out DDRB,temp ; порт на вывод
```

```

Out PORTB,temp
Out PortD,temp ; подключение внутренних резисторов
; ----- Инициализация таймера T1 -----
Ldi temp,0x0D ;выбор режима: сброс таймера канала A по совпадению и
Out TCCR1B,temp ;делитель тактовой частоты CLC/1024
Ldi temp,high(kdel) ; старший байт кода совпадения
Out OCR1AH, temp ; запись в регистр совпадения ст. байта
Ldi temp,low(kdel) ;мл. Байт кода совпадения
Out OCR1AL,temp
; ----- Определение маски прерываний -----
Ldi temp,0x14 ; байт маски. Разрешено только одно прерывание
Out TIMSK,temp ; по совпадению в канале A
; ----- Инициализация компаратора -----
Ldi temp,0x80 ; Выключение компаратора, чтобы случайно не возникло
Out ACSR,temp ; прерывание от компаратора, т.к. он после сброса включен

; ----- Начало основной программы -----
Ldi rab,0b00010000 ; т.к. сдвиг м.б. влево-вправо, то 1 в середине
Sts buf,rab ; Запись слова управления индикаторами в ОЗУ
Sei ; Установка Флага разрешение всех прерываний
m1:
Rjmp m1 ; Пустой бесконечный цикл. Ожидание прерывания

; ----- Подпрограмма обработки прерываний -----
;Prtim1:
Push temp ; Сохранение регистра
Push rab
Lds rab,buf ; Чтение содержимого из ОЗУ
In temp,PIND ; Чтение содержимого порта, куда подключена кнопка
Sbrs temp,0 ; Проверка младшего разряда (пропустить, если 0-й бит в 1)
Rjmp p2 ; Если не ноль, переход к сдвигу влево
; ----- Сдвиг вправо -----
p1:
Lsr rab ; Сдвиг вправо содержимого регистра
Brcc p3 ; Если не дошло до конца регистра, пропустить
Ldi rab,0b10000000 ; Запись начального значения
Rjmp p3 ; Перейти в конец
; ----- Сдвиг влево -----
p2:
Lsl rab ; Сдвиг влево содержимого регистра
Brcc p3 ; Если не дошло до конца регистра, пропустить
Ldi rab,0b00000001 ; Запись начального значения
;
; ----- Конец процедуры обработки прерываний -----
p3:
Ldi temp,0xFF
Eor temp,rab ; Инверсия содержимого, если диод зажигается нулем.
Out PORTB,temp
Sts buf,rab ; Сохранение регистра в ОЗУ
Pop rab ; Восстановление регистров
Pop temp
Reti

```

ИССЛЕДОВАНИЕ СПОСОБОВ ПРОГРАММИРОВАНИЯ АЦП И ВЫВОДА ИНФОРМАЦИИ В МИКРОКОНТРОЛЛЕРАХ AVR

1 Цель работы

Целью работы является изучение и исследование процессов цифро-аналогового преобразования сигналов и вывода информации на цифровые индикаторы в однокристальных микро-ЭВМ, приобретение практических навыков составления, исследования и отладки микропроцессорных систем и их программного обеспечения.

2 Краткие теоретические сведения

В своем составе ряд микроконтроллеров: ATtiny15, AT90S/4433, AT90S8535, ATmega8x, ATmega16x, ATmega32x и др. имеют аналого-цифровой преобразователь. Структурная схема аналого-цифрового преобразователя микроконтроллера AT90S8535 приведена на рисунке 2.1.

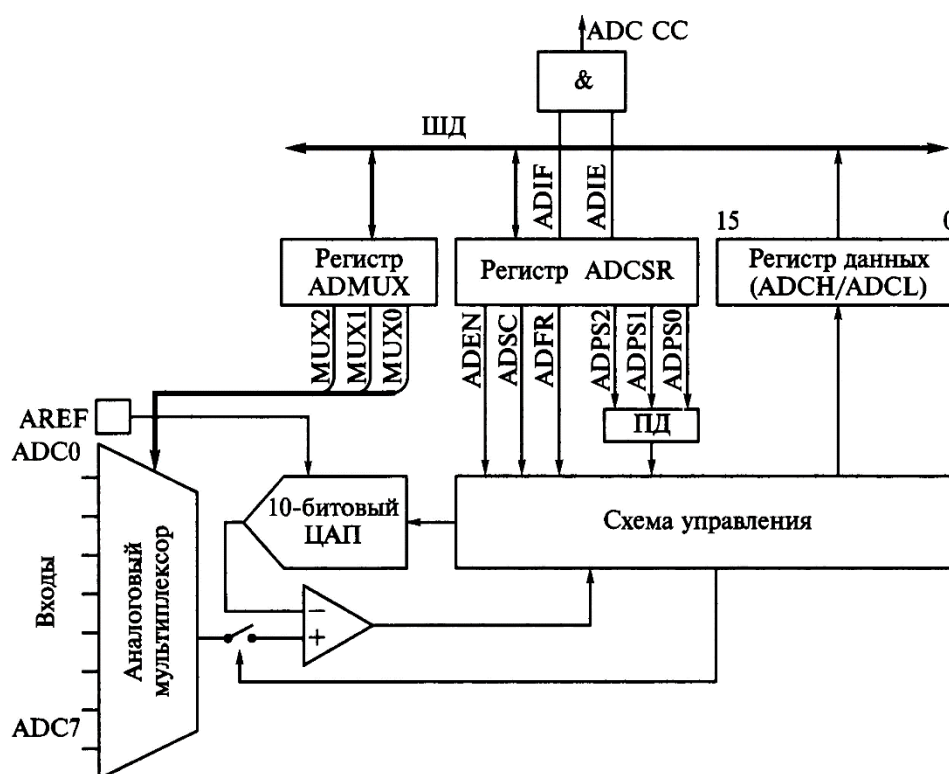


Рисунок 2.1 – Структурная схема АЦП

АЦП подсоединен к 8-канальному аналоговому мультиплексору, позволяющему использовать любой вывод порта А в качестве входа АЦП. АЦП содержит усилитель выборки/хранения, усиливающий и удерживающий напряжение входа

АЦП во время преобразования на неизменном уровне. Усиления сигнала может устанавливаться программно от 10 до 100.

С целью уменьшения наводок помех, имеющих в цепях питания цифровых устройств (VCC и GND), для питания АЦП используются отдельный источник питания аналоговых цепей (AVCC и AGND). Вывод AGND должен быть подсоединен к GND, а напряжение AVCC не должно отличаться от напряжения VCC более чем на $\pm 0,3$ В. Кроме того, на АЦП для подачи напряжения на резистивную сетку цифроаналогового преобразователя (ЦАП) подается внешнее опорное напряжение (AREF), которое должно быть в диапазоне от AGND до AVCC. Большинство МК имеют внутренний калиброванный источник опорного напряжения. Для микроконтроллера это напряжение равно 2,56 В. Следует заметить, что входное напряжение не должно превышать опорное.

Для управления работой АЦП используется два регистра: регистр состояния и управления ADC – ADCSR и регистр управления мультиплексором ADC – ADMUX. Результат аналого-цифрового преобразования заносится в регистр данных ADCH и ADCL, из 16 разрядов которого задействованы всего лишь 10. Для изменения тактовой частоты работы преобразователя в схему АЦП введен предварительный делитель импульсной последовательности задающего генератора (ПД). Обычно АЦП работает с тактовой частотой в диапазоне от 50 до 250 кГц. Делитель частоты работает, когда установлен бит ADEN.

Аналого-цифровой преобразователь может работать в двух режимах: однократного преобразования и циклического преобразования. В режиме однократного преобразования каждое преобразование инициируется пользователем. В режиме циклического преобразования АЦП осуществляет выборку и обновление содержимого регистра данных преобразователя непрерывно. Выбор режима производится битом ADATE регистра состояния и управления ADCSR.

Работа АЦП разрешается установкой в состояние 1 бита ADEN в регистре ADCSR. Первому преобразованию, начинающемуся после разрешения АЦП, предшествует пустое инициализирующее преобразование. На пользователе это отражается лишь тем, что первое преобразование будет занимать 26 тактовых интервалов вместо обычных 14. Преобразование начинается с установки в состояние 1 бита начала преобразования ADSC. Этот бит находится в состоянии 1 в течение всего цикла преобразования и сбрасывается по завершению преобразования аппаратно. Если в процессе преобразования выполняется смена канала данных, то АЦП вначале закончит текущее преобразование и лишь потом выполнит переход к другому каналу.

АЦП формирует 10-разрядный результат в двух регистрах – ADCH и ADCL. Для чтения правильного результата из этих регистров существует следующий механизм. При чтении результата первым должен читаться регистр ADCL. После этого доступ АЦП к регистрам данных блокируется. Это обозначает, что если следующее преобразование завершено между чтением ADCL и ADCH, его результат будет потерян. После чтения регистра ADCH доступ АЦП к регистрам данных восстанавливается. АЦП имеет свой флаг и вектор прерывания. Флаг запроса ADIF устанавливается при завершении преобразования.

Формат регистра состояния и управления **ADCSR** показан на рисунке 2.2.

Бит	7	6	5	4	3	2	1	0
\$06 (\$26)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
Чтение/Запись	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Рисунок 2.2 — Формат слова регистра состояния и управления ADCSR

- Бит 7 – **ADEN** (*ADC Enable*) – Разрешение АЦП. Установка данного бита в состояние 1 разрешает работу АЦП. Очистка бита запрещает работу АЦП. Запрещение АЦП в процессе преобразования прекращает преобразование.

- Бит 6 – **ADSC** (*ADC Start Conversion*) – Запуск преобразования. В режиме однократного преобразования для запуска каждого цикла преобразования необходимо каждый раз программно устанавливать бит ADSC в состояние 1. В циклическом режиме бит ADSC устанавливается в состояние 1 только при запуске первого цикла преобразования. Каждый раз после первой установки бита ADSC, выполненной после разрешения ADC или одновременно с разрешением ADC, будет выполняться пустое преобразование, предшествующее активируемому преобразованию. Это пустое преобразование активирует АЦП. ADSC будет сохранять состояние 1 в течение всего цикла преобразования и сбрасывается по завершении преобразования.

- Бит 5 – **ADATE** (*ADC Auto Trigger Enable*) – Выбор режима работы АЦП. В микроконтроллере Atmega8535 запуск АЦП возможен не только по команде пользователя, но и по прерыванию от некоторых периферийных устройств, имеющих в составе микроконтроллера. Для выбора режима работы используется разряд ADATE регистра ADCSRA и разряды ADTS2..0 регистра специальных функций SFIOR.

Если разряд ADATE сброшен в «0», АЦП работает в режиме одиночного преобразования. Если же разряд ADATE установлен в «1», функционирование АЦП определяется содержимым разрядов ADTS2...0 согласно табл. 2.1.

Таблица 2.1. Источник сигнала для запуска преобразования Atmega8535

ADTS2	ADTS1	ADTS0	Источник стартового сигнала
0	0	0	Режим непрерывного преобразования
0	0	1	Прерывание от аналогового компаратора
0	1	0	Внешнее прерывание INT0
0	1	1	Прерывание по событию «Совпадение» таймера/счетчика 0
1	0	0	Прерывание по переполнению таймера/счетчика 0
1	0	1	Прерывание по событию «Совпадение В» таймера/счетчика 1
1	1	0	Прерывание по переполнению таймера/счетчика 1
1	1	1	Прерывание по событию «Захват» таймера/счетчика 1

- Бит 4 - **ADIF** (*ADC Interrupt Flag*) - Флаг прерывания АЦП. Данный бит устанавливается в состояние 1 по завершению преобразования и обновления регистров данных. Прерывание по завершению преобразования АЦП выполняется, если в состояние 1 установлены бит ADIE и I-бит регистра SREG. Бит ADIF сбрасывается аппаратно при выполнении подпрограммы обработки соответствующего вектора прерывания.

- Бит 3 – **ADIE** (*ADC Interrupt Enable*) - Разрешение прерывания от АЦП. При установленных в состояние 1 бите ADIE и I-бите регистра SREG активируется прерывание по завершению преобразования АЦП.

- Биты 2...0 – **ADPS2...ADPS0** (*ADC Prescaler Select Bits*) – Выбор коэффициента предварительного деления. Данные биты определяют коэффициент деления частоты XTAL для получения необходимой тактовой частоты АЦП (табл. 2.2).

Таблица 2.2. Задание коэффициента предварительного деления

ADPS2	ADPS1	ADPS0	Коэффициент деления
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

Формат регистра выбора канала и источника опорного напряжения аналого-цифрового преобразователя – ADMUX показан на рисунке 2.3.

Бит	7	6	5	4	3	2	1	0
\$07 (\$27)	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0
Чтение/ Запись	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

Рисунок 2.3 - Формат регистра выбора мультиплексора ADC – ADMUX

- Биты 7, 6 – **REFS1:REFS0** (*Reference Selection Bits*) – Выбор источника опорного напряжения. Модуль АЦП может использовать различные источники опорного напряжения. Выбор конкретного источника опорного напряжения осуществляется с помощью разрядов REFS1:REFS0 регистра ADMUX (табл. 2.3). Как указано в таблице, внутренний источник опорного напряжения подключается ко входу AREF микроконтроллера. Поэтому при его использовании к пину AREF можно подключить внешний фильтрующий конденсатор для повышения помехозащищенности.

Таблица 2.3. Выбор источника опорного напряжения

REFS1	REFS0	Источник опорного напряжения (ИОН)
0	0	Внешний ИОН, подключенный к выводу AREF, внутренний ИОН отключен
0	1	Напряжение питания AVCC
1	0	Зарезервировано
1	1	Внутренний ИОН напряжением 2,56 В, подключенный к выходу AREF

- Бит 5 – **ADLAR** (*ADC Left Adjust Result*) – Выравнивание результата преобразования влево. По умолчанию результат преобразования выравнивается вправо (старшие 6 разрядов регистра ADCH незначащие). Однако он может выравниваться и влево (младшие 6 разрядов регистра ADCL незначащие). Для управления выравниванием результата преобразования служит разряд ADLAR регистра ADMUX. Если этот разряд установлен в «1», результат преобразования выравнивается по левой границе 16-разрядного слова, если обращен в «0» – по правой границе.

- Биты 4...0 – **MUX4.. MUX0** (*Analog Channel Select Bits 2-0*) – Биты выбора аналогового канала. Состояние данных битов определяет, какой из восьми аналоговых каналов (0 – 7) будет подключен к АЦП, а также различные варианты подключения каналов с дифференциальным входом, в том числе с предварительным усилением входного сигнала с коэффициентом 10 или 200.

3 Описание лабораторного стенда

В состав лабораторного стенда входит персональный компьютер с установленными отладочными системами Atmel Studio версии 6 и выше и Proteus 8 (демоверсия). Описание этих систем и особенности работы с ними приведены в методических указаниях к предыдущим лабораторным работам.

4 Программа работы

- 4.1 Изучить устройство аналого-цифрового преобразователя микроконтроллеров семейства AVR и особенности его программирования.
- 4.2 Составить схему устройства на основе микроконтроллера типа ATmega, состоящую из задатчика непрерывного напряжения, преобразователя двоичного кода в 7-сегментный и дисплея на 7-сегментных индикаторах.
- 4.3 Разработать программу на языке ассемблера измерения напряжения и вывода результатов измерения на индикацию. Способ измерения и отображения результатов определяется вариантом задания.

- 4.4 Выполнить отладку программы управления устройством в среде AVRStudio.
- 4.5 Составить схему устройства, произвести запись программы управления в память микроконтроллера и исследовать функционирование разработанной схемы устройства в среде Proteus.
- 4.6 Составить отчет по результатам лабораторных исследований.
- 4.7 Примечание: Пп. 4.1-4.3 программы работы выполняются в процессе домашней подготовки.

5 Методические рекомендации по выполнению работы

- 5.1 В качестве датчика входного непрерывного напряжения целесообразно воспользоваться линейным потенциометром (в системе Proteus POT-LIN), подключенным к одно- или двуполярному источнику напряжения.
- 5.2 Для преобразования двоично-десятичного кода числа в 7-сегментный код рекомендуется воспользоваться микросхемой NSC 7477), в качестве регистра — микросхему 74LS174
- 5.3 В качестве 7-сегментного дисплея можно взять устройство LED MPX, модель которого имеется в системе Proteus (рисунок 5.1).

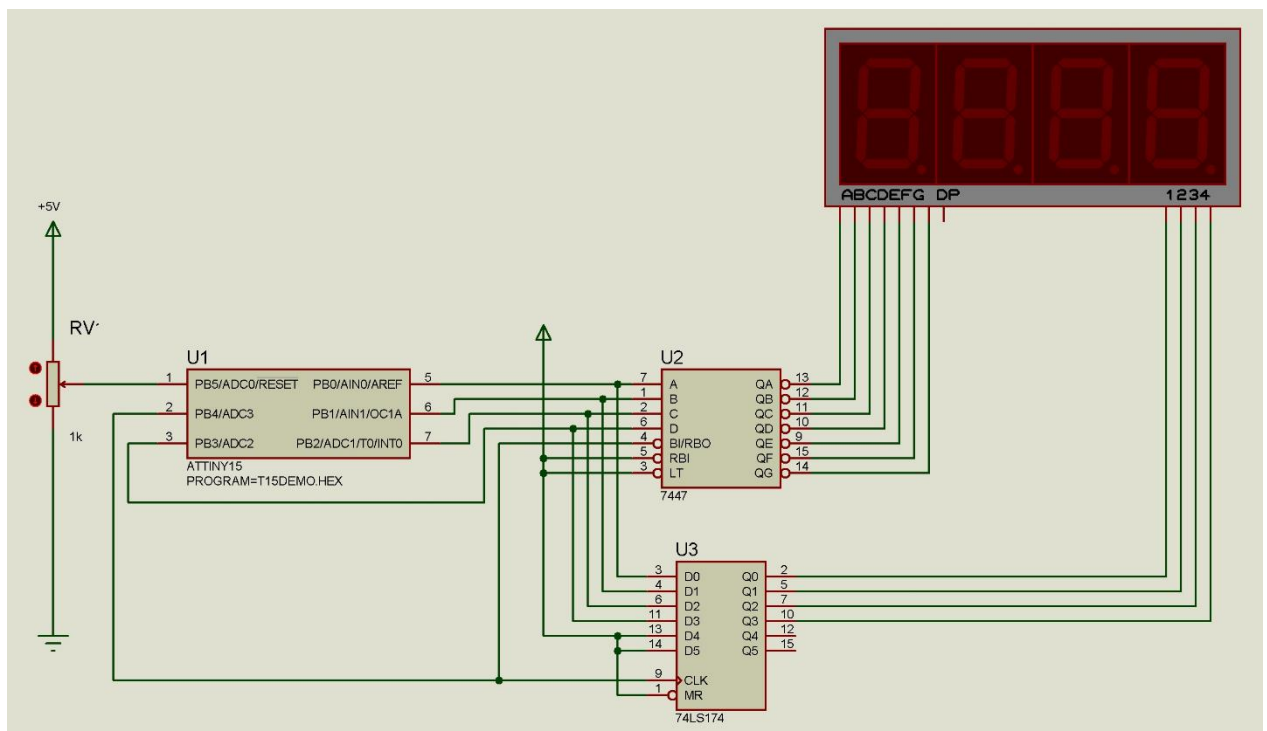


Рисунок 5.1 – Вариант схемы исследуемого устройства

6 Контрольные вопросы

- 6.1 Расскажите о составе и функционировании АЦП микроконтроллеров семейства AVR.
- 6.2 С какой целью на АЦП подаются напряжения от трех независимых источников питания?
- 6.3 От чего зависит погрешность преобразования и как она рассчитывается?
- 6.4 Расскажите о назначении регистра состояния и управления АЦП.
- 6.5 Поясните какие режимы работы имеются в АЦП и в каких случаях какие из них применяются?
- 6.6 Как осуществляется чтение 10-битового результата АЦП, если регистры микроконтроллера 8-разрядные?
- 6.7 Каким образом осуществляется разовый запуск аналого-цифрового преобразования?
- 6.8 Как можно осуществлять запуск АЦП через строго определенные промежутки времени?
- 6.9 С какой целью в схеме установки (рис.5.1) установлены микросхемы V2 и V3?
- 6.10 Каким образом можно управлять цифровым дисплеем без использования микросхем V2 и V3?

Приложение А. Варианты индивидуальных заданий

1. В одном из регистров общего назначения записано число, определяющее заданный уровень порога преобразуемого напряжения. Если напряжение на входе АЦП меньше (ниже) этого порога, то на семисегментном индикаторе HG2 высвечивается буква Н, при равенстве напряжений – буква Р, при превышении преобразуемым напряжением уровня порога (больше) – буква Б. При выполнении этого задания два младших разряда преобразованного числа АЦП опустить.

2. Полученный результат преобразования АЦП вывести в шестнадцатеричном формате на семисегментные индикаторы HG1 и HG2 (динамическая индикация). При изменении положения движка потенциометрического резистора наблюдать изменение показаний на индикаторах HG1 и HG2. При выполнении задания два младших разряда преобразованного числа АЦП опустить.

3. В двух регистрах общего назначения записаны числа, определяющие заданные уровни порогов преобразуемого напряжения. В первом регистре число меньше, чем во втором регистре. Если напряжение на входе АЦП меньше (ниже) порога первого регистра, то на семисегментном индикаторе HG1 высвечивается буква Н. Если результат преобразования АЦП находится между двумя порогами, то он отображается на индикаторах HG1 и HG2 (динамическая индикация). Если же результат преобразования превышает порог второго регистра, то на индикатор HG1 выводится буква Б. При выполнении задания два младших разряда преобразованного числа АЦП опустить.

4. Организовать мигание светодиода VD6. В момент загорания светодиода должен в течение 0,5 с прозвучать звукоизлучатель HA1. При увеличении уровня напряжения на входе АЦП, скорость мигания увеличивается, при уменьшении напряжения – скорость уменьшается.

5. В двух регистрах общего назначения записаны числа, определяющие заданные уровни порогов преобразуемого напряжения. В первом регистре число меньше, чем во втором регистре. Если напряжение на входе АЦП меньше порога первого регистра, то на семисегментных индикаторах HG1 и HG2 в шестнадцатеричном формате выводится число первого регистра. Если напряжение на входе АЦП больше порога второго регистра, то – число второго регистра. Если же напряжение на входе АЦП располагается между указанными порогами, то выводится результат преобразования АЦП.

6. Организовать бегущий огонь на семисегментном индикаторе HG2. При увеличении уровня напряжения на входе АЦП скорость перемещения свечения на семисегментном индикаторе увеличивается, при уменьшении – скорость перемещения уменьшается.

7. В двух регистрах общего назначения записаны числа, определяющие заданные уровни порогов преобразуемого напряжения. В первом регистре число меньше, чем во втором регистре. Если напряжение на входе АЦП меньше (ниже) первого уровня, то горит 1 светодиод. Если оно находится между двумя уровнями, то горят 2 светодиода. Если же напряжение на входе АЦП выше второго уровня, то горят 3 светодиода.

8. Организовать бегущий огонь на светодиодах VD1 – VD8. При увеличении уровня напряжения на входе АЦП, скорость бега на светодиодах увеличивается. При уменьшении – скорость бега уменьшается.

9. Полученный результат преобразования АЦП вывести в виде двоичного кода на светодиоды VD1 – VD8. При изменении положения движка потенциометрического резистора наблюдать изменение кода на светодиодах VD1 – VD8. При выполнении задания два младших разряда преобразованного числа АЦП опустить.

10. Разбить весь диапазон напряжений на входе АЦП на 8 равных поддиапазонов. При попадании преобразуемого напряжения в какой-либо поддиапазон зажечь соответствующий светодиод (в порядке VD1, VD2, ..., VD8).

11. Организовать формирование звука изменяющейся частоты с помощью звукоизлучателя HA1. При увеличении уровня напряжения на входе АЦП частота звука увеличивается, при уменьшении напряжения – уменьшается.

12. Восемь младших разрядов полученного результата преобразования АЦП вывести в шестнадцатеричном формате на семисегментные индикаторы HG1 и HG2 (динамическая индикация). При изменении положения движка потенциометрического резистора наблюдать изменение информации на индикаторах HG1 и HG2. При равенстве или превышении преобразованным числом АЦП значения 256 зажечь на индикаторе HG1 букву Б.

13. Разбить весь диапазон напряжений на входе АЦП на 16 равных поддиапазонов. При попадании преобразуемого напряжения в какой-либо поддиапазон высветить на семисегментном индикаторе HG4 соответствующую шестнадцатеричную цифру от 0 до F.

14. Разбить весь диапазон напряжений на входе АЦП на 4 равных поддиапазонов. При попадании преобразуемого напряжения в какой-либо поддиапазон высвечивать на поочередно на семисегментных индикаторах HG1 – HG4 соответствующую цифру от 0 до 3. В каждом более высоком диапазоне скорость поочередного зажигания индикаторов увеличивается.

15. Разбить весь диапазон напряжений на входе АЦП на 8 равных поддиапазонов. При попадании преобразуемого напряжения в какой-либо поддиапазон зажечь соответствующее количество светодиодов – 1, 2, ..., 8 (в порядке VD1, VD2, ..., VD8).

16. Полученный результат преобразования АЦП вывести в десятичном формате на семисегментные индикаторы HG1 и HG2 (динамическая индикация). При изменении положения движка потенциометрического резистора наблюдать изменение информации на индикаторах HG1 и HG2. При выполнении задания четыре младших разряда преобразованного числа АЦП опустить.

Приложение Б. Пример составления программы управления работой АЦП

Задача. Составить для микроконтроллера ATmega8535 программу аналого-цифрового преобразования и вывода результата преобразования на 7-сегментный индикатор в следующем виде:

На индикатор выводится цифра 0, если напряжение на входе АЦП находится в пределах 0...0,5 В;

1 – в пределах 0,5...1,5 В;

2 – в пределах 1,5...2,5 В;

3 – в пределах 2,5...3,5 В;

4 – в пределах 3,5...4,5 В;

5 – в пределах 4,5...5,0 В.

```
.include "m8535def.inc" ; включить файл описания ATmega8535
.def code = r20          ; регистр куда помещается преобразованное число
.ORG$0                   ; Обработка прерывания сброса
rjmp Reset
.ORG$0e                   ; Вектор АЦП
rjmp inADC
;
Reset:
ldi r16,$02               ; Инициализация вершины стека по адресу $025F
out SPH,r16
ldi r16,$5f
out SPL,r16
cbi DDRA,0                ; PA0 на ввод
sbi PORTA,0               ; Отключить подтягивающий резистор от пина PA0
ldi r16,$0ff              ; Определить все выводы портов D и C
out DDRD,r16               ; на выход
out DDRC,r16
sbi PORTD,4                ; Активизировать семисегментный индикатор HG1
;
ldi r16,$00                ; выбор нулевого канала АЦП (PA0)
out ADMUX,r16
ldi r16,$0EA               ; инициализация АЦП: разрешение и запуск циклического
out ADCSRA,r16             ; преобразования без деления тактовой частоты
sei                         ; установка общего флага прерываний
m1:
rjmp m1                    ; рабочий цикл программы, процессор ожидает прерывания
;
In ADC:                    ; Подпрограмма обслуживающая прерывания
; по запросу АЦП
in r16,ADCL                ; Считать младший байт регистра данных АЦП
in code,ADCH                ; Считать старший байт регистра данных АЦП
lsr r16                    ; Исключить два младших разряда
lsr r16                    ; преобразованного сигнала
swap code                  ; Поменять местами биты 3...0 с
lsl code                    ; битами 7...4 и сдвинуть влево на
lsl code                    ; два разряда
andi code,$0c0              ; Выделить биты 7 и 6 code
```

or code,r16	; Получить в code восьмиразрядное число
rcall kod	; Вызвать подпрограмму зажигания цифры
reti	
;	
kod:	; Подпрограмма зажигания цифры
ldi r22,51	; В r22 отсчет напряжения 1 В на РА0 для цифр
	; на семисегментном индикаторе
ldi r21,26	; В r21 максимальное значение кода АЦП при
	; котором на индикаторе горит 0 - порог нуля
cp code,r21	; Если код АЦП не превышает порог нуля, то
brlo k0	; зажечь 0
add r21,r22	; Установить порог единицы
cp code,r21	; Если код АЦП не превышает порог единицы, то
brlo k1	; зажечь 1
add r21,r22	; Установить порог двойки
cp code,r21	; Если код АЦП не превышает порог двойки, то
brlo k2	; зажечь 2
add r21,r22	; Установить порог тройки
cp code,r21	; Если код АЦП не превышает порог тройки, то
brlo k3	; зажечь 3
add r21,r22	; Установить порог четверки
cp code,r21	; Если код АЦП не превышает порог четверки, то
brlo k4	; зажечь 4
ldi r23,\$6d	; Если код АЦП выше порога четверки, то зажечь 5
out PORTC,r23	
ret	; Выход из подпрограммы
;	
k0:	
ldi r23,\$3f	; Зажечь 0
out PORTC,r23	
ret	; Возврат
k1:	
ldi r23,\$06	; Зажечь 1
out PORTC,r23	
ret	; Возврат
k2:	
ldi r23,\$5b	; Зажечь 2
out PORTC,r23	
ret	; Возврат
k3:	
ldi r23,\$4f	; Зажечь 3
out PORTC,r23	
ret	; Возврат
k4:	
ldi r23,\$66	; Зажечь 4
out PORTC,r23	
ret	; Возврат

ИССЛЕДОВАНИЕ МЕТОДОВ ПРОГРАММИРОВАНИЯ И НАСТРОЙКИ ПОСЛЕДОВАТЕЛЬНОГО ИНТЕРФЕЙСА USART МИКРОКОНТРОЛЛЕРОВ AVR

7 Цели работы

Целями работы является углубление теоретических знаний об устройстве и особенностях программирования типового последовательного интерфейса микроконтроллеров USART, экспериментальное исследование структуры и параметров сигналов передаваемых кадров, приобретение практических навыков составления, исследования и отладки микропроцессорных систем и их программного обеспечения.

2 Устройство и особенности программирования универсального последовательного синхронно/асинхронного интерфейса

Универсальный синхронный и асинхронный последовательный приемопередатчик **УСАПП** (англ. **USART** — *Universal Synchronous Asynchronous Receiver Transmitter*) предназначен для организации гибкой последовательной связи с периферийными устройствами, имеющими аналогичный интерфейс. Интерфейс обеспечивает функционирование в следующих режимах:

- полнодуплексная работа (раздельные регистры последовательного приема и передачи);
- асинхронная или синхронная работа;
- ведущее или подчиненное тактирование связи в синхронном режиме работы;
- высокая разрешающая способность генератора скорости связи;
- поддержка формата передаваемых данных с 5, 6, 7, 8 или 9 битами данных и 1 или 2 стоп-битами;
- аппаратная генерация и проверка бита паритета (четность/нечетность)
- определение переполнения данных;
- определение ошибки в структуре кадра (посылки);
- фильтрация шума с определением ложного старт-бита и цифровым фильтром нижних частот (ФНЧ);
- три раздельных прерывания по завершении передачи, освобождении регистра передаваемых данных и завершении приема;
- режим многопроцессорной связи;
- режим удвоения скорости связи в асинхронном режиме.

Передаваемая последовательность (кадр, посылка) состоит из битов данных, битов синхронизации (старт и стоп-биты), а также опционального бита паритета

для обнаружения ошибок. УСАПП поддерживает все комбинации следующих форматов посылок:

- один стартовый бит;
- 5, 6, 7, 8 или 9 битов данных;
- без паритета, с битом четности, с битом нечетности;
- 1 или 2 стоповых бита.

Посылка начинается со старт-бита, а за ним следует передача бит данных, начиная с самого младшего разряда. Затем следует передача остальных битов данных (макс. число бит данных 9), которая заканчивается передачей старшего разряда данных. Если разрешена функция контроля паритета, то сразу после битов данных передается бит паритета, а затем стоп-биты. После завершения передачи посылки имеется возможность либо передавать следующую посылку, либо перевести линию связи в состояние ожидания (высокий уровень). Рисунок 2.1 иллюстрирует возможность сочетания форматов посылки. Наличие прямоугольной скобки указывает на опциональность данного формата посылки.

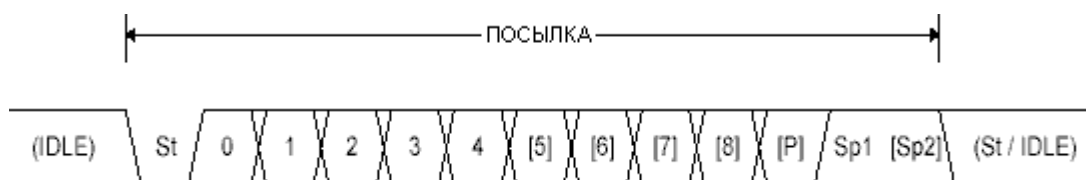


Рисунок 2.1 - Форматы посылки (кадра)

Обозначения, используемые на рисунке, имеют следующие значения:

- St - Старт-бит (всегда низкий уровень);
- 0...8 - Номер бита данных;
- P - бит паритета: четность или нечетность;
- Sp1, Sp2 - Стоп-бит (всегда высокий уровень);
- IDLE - состояние ожидания, в котором приостановлена передача на RxD или TxD. В состоянии ожидания на линии должен быть высокий уровень.

Формат посылки, который используется УСАПП, задается битами UCSZ2:0, UPM1:0 и USBS в регистрах состояния и управления UCSRB и UCSRC. Приемник и передатчик используют одни и те же установки форматов. Биты выбора длины передаваемых данных (UCSZ2:0) определяют, из скольких битов данных состоит посылка. Биты режима паритета УСАПП (UPM1:0) разрешают передачу/контроля бита паритета и устанавливают тип паритета: четность или нечетность. Выбрать один или два стоп-бита позволяет бит выбора стоп-бита УСАПП (USBS). Приемник игнорирует второй стоп-бит. Флаг ошибки посылки FE устанавливается в 1, если первый стоп-бит равен 0. Бит паритета вычисляется путем выполнения логической операции исключающего ИЛИ над всеми битами данных.

В микроконтроллерах типа ATMEGA для работы с модулем USART используются 6 регистров:

- регистры управления и состояния UCSRA, UCSRB и UCSRC (**USART Control and Status Register**);

- регистры задания скорости передачи UBRR1 и UBRRH (**USART Baud Rate Registers**);

- регистр данных UDR (**USART Data Register**).

Все настройки приемопередатчика хранятся в регистрах конфигурации. UCSRA, UCSRB и UCSRC, а скорость задается в паре UBRRH:UBRRL.

Форматы битовых полей регистров приведены в сводной таблице регистров (Таблица 2.1).

Таблица 2.1 - Сводная таблица регистров

Наименование регистра	Разряды							
	7	6	5	4	3	2	1	0
UCSR1A	RXC1	TXC1	UDRE1	FE1	DOR1	UPE1	U2X1	MPCM1
UCSR1B	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81
UDR1	Регистр данных УСАПП1							
UCSR0C	-	UMSEL0	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0
UCSR1C	-	UMSEL1	UPM11	UPM10	USBS1	UCSZ11	UCSZ10	UCPOL1
UBRR1L	Младший байт регистра скорости связи УСАПП1							
UBRR1H	-	-	-	-	Старший байт регистра скорости связи УСАПП1			
UBRR0H	-	-	-	-	Старший байт регистра скорости связи УСАПП0			

Биты управляющих регистров имеют следующие значения.

Регистр управления и состояния **UCSRA**:

RXC (*Receive Complete*) – Флаг завершения приема. Флаг устанавливается в «1» при наличии непрочитанных данных в буфере приемника (регистр данных UDR). Сбрасывается флаг аппаратно после опустошения буфера.

TXC (*Transmit Complete*) – Флаг завершения передачи. Флаг устанавливается в «1» после передачи всех разрядов посылки из сдвигового регистра передатчика, при условии, что в регистр данных UDR не было загружено нового значения. Флаг сбрасывается аппаратно при выполнении подпрограммы обработки прерывания или программно, записью в него лог. 1.

UDRE (*UDR Empty*) – Флаг опустошения регистра данных. Данный флаг устанавливается в «1» при пустом буфере передатчика (после пересылки байта из регистра данных UDR в сдвиговый регистр передатчика). Установленный флаг означает, что в регистр данных можно загружать новое значение. Если разряд UDRIE регистра UCR (UCSRB) установлен, генерируется запрос на прерывание «регистр данных пуст». Флаг сбрасывается аппаратно, при записи в регистр данных.

FE (*Frame Error*) – Флаг ошибки кадрирования. Флаг устанавливается в «1» при обнаружении ошибки кадрирования, т. е. если первый стоп-бит принятой посылки равен «0». Флаг сбрасывается при приеме стоп бита, равного «1».

DOR (*Data OverRun*) – Флаг переполнения. В USART флаг устанавливается в «1», если в момент обнаружения нового стартового бита в сдвиговом регистре приемника находится последнее принятое слово, а буфер приемника полон (два значения). В UART флаг устанавливается в «1», если новый кадр будет помещен в сдвиговый регистр приемника до того, как из регистра данных будет считано предыдущее слово. Флаг сбрасывается при пересылке принятых данных из сдвигового регистра приемника в буфер.

UPE (*USART Parity Error*) – Флаг ошибки контроля четности. Флаг устанавливается в «1», если в данных, находящихся в буфере приемника, выявлена ошибка контроля четности. При отключенном контроле четности этот разряд постоянно читается как «0».

U2X (*Double speed*) – Удвоение скорости обмена. Если этот разряд установлен в «1», коэффициент деления предделителя контроллера скорости передачи уменьшается с 16 до 8, удваивая тем самым скорость асинхронного обмена по последовательному каналу. В USART разряд U2X используется только при асинхронном режиме работы. В синхронном режиме он должен быть сброшен.

MPCM (*Multi-Processor Communication Mode*) – Режим мультипроцессорного обмена. Разряд MPCM используется в режиме мультипроцессорного обмена. Если он установлен в «1», ведомый микроконтроллер ожидает приема кадра, содержащего адрес. Кадры, не содержащие адреса устройства, игнорируются.

Управляющий регистр **UCSRB**:

RXCIE (*Receive Complete Interrupt Enable*) – Разрешение прерывания по завершению приема. Если данный разряд установлен в «1», то при установке флага RXC регистра UCSRA генерируется прерывание.

TXCIE (*Transmit Complete Interrupt Enable*) – Разрешение прерывания по завершению передачи. Если данный разряд установлен в «1», то при установке флага TXC регистра UCSRA генерируется прерывание.

UDRIE (*UDR Empty Interrupt Enable*) – Разрешение прерывания при очистке регистра данных UART. Если данный разряд установлен в «1», то при установке флага UDRE в регистре UCSRA генерируется прерывание.

RXEN (*Receiver Enable*) - Разрешение приема. При установке этого разряда в «1» разрешается работа приемника USART/UART и переопределяется функционирование вывода RXD.

TXEN (*Transmitter Enable*) – Разрешение передачи. При установке этого разряда в «1» разрешается работа передатчика UART и переопределяется функционирование вывода TXD.

UCSZ2 (*USART Character SiZe 2*) – Формат посылок. Этот разряд используется для задания размера слов данных, передаваемых по последовательному

каналу. В модулях USART он используется совместно с разрядами UCSZ1 и UCSZ0 регистра UCSRC.

RXB8 (*Receive Data Bit 8*) – 8 й разряд принимаемых данных. При использовании 9 разрядных слов данных этот разряд содержит значение старшего разряда принятого слова. В случае USART содержимое этого разряда должно быть считано до прочтения регистра данных UDR.

TXB8 (*Transmit Data Bit 8*) – 8 й разряд передаваемых данных. При использовании 9 разрядных слов данных, содержимое этого разряда является старшим разрядом передаваемого слова. Требуемое значение должно быть занесено в этот разряд до загрузки байта данных в регистр UDR.

Управляющий регистр **UCSRC**:

URSEL (*USART Register SElect*) – Выбор регистра. Этот разряд определяет, в какой из регистров UCSPP производится запись. Если разряд установлен в «1», обращение производится к регистру UCSRC. Если же разряд сброшен в «0», обращение производится к регистру UBRRH.

UMSEL (*USART Mode SElect*) – Режим работы USART. Если разряд сброшен в «0», модуль USART работает в асинхронном режиме. Если разряд установлен в «1», то модуль USART работает в синхронном режиме.

UPM1 и **UPM0** (*USART Parity Mode*) – Режим работы схемы контроля и формирования четности. Эти разряды определяют функционирование схем контроля и формирования четности (1:1 – нечётный; 1:0 – чётный; 0:0 – нет контроля паритета).

USBS (*USART Stop Bit Select*) – Количество стоп-битов. Этот разряд определяет количество стоп битов, посылаемых передатчиком. Если разряд сброшен в «0», передатчик посылает 1 стоп бит, если установлен в «1», то 2 стоп бита. Для приемника содержимое этого разряда безразлично.

UCSZ1 и **UCSZ0** (*USART Character SiZe*) – Формат посылок. Совместно с разрядом UCSZ2 эти разряды определяют количество разрядов данных в посылках (размер слова).

UCPOL (*USART Clock POLarity*) – Полярность тактового сигнала. Значение этого разряда определяет момент выдачи и считывания данных на выводах модуля. Разряд используется только при работе в синхронном режиме. При работе в асинхронном режиме он должен быть сброшен в «0».

Регистры скорости передачи **UBRRL** и **UBRRH**:

USEL (*USART Register Select*) – Выбор регистра. Этот разряд определяет, в какой из регистров модуля производится запись. Если разряд установлен в «1», обращение производится к регистру UCSRC. Если же разряд сброшен в «0», обращение производится к регистру UBRRH.

UBRR11 – UBRR0 (*USART Baud Rate Register*) – Значение скорости передачи в Бодах.

Установка скорости передачи в асинхронном режиме. При работе в асинхронном режиме скорость обмена определяется не только содержимым регистра **UBRR**, но и состоянием разряда U2X регистра **UCSRA**.

Если этот разряд установлен в «1», коэффициент деления предделителя уменьшается в два раза, а скорость обмена соответственно удваивается. При работе в синхронном режиме этот разряд должен быть сброшен.

Итак, скорость обмена определяется следующими формулами:

- асинхронный режим (обычный, U2X = «0»): $BAUD = f_{CK}/(16(UBRR+1))$;
- асинхронный режим (ускоренный, U2X = «1»): $BAUD = f_{CK}/(8(UBRR+1))$;
- синхронный режим ведущего: $BAUD = f_{CK}/(2(UBRR+1))$;

где BAUD — скорость передачи в бодах; f_{CK} — тактовая частота микроконтроллера; UBRR — содержимое регистра контроллера скорости передачи (0...4095).

Пример начальной инициализации УСАПП.

```
.equ FSK = 8000000 ; Частота задающего генератора МК
.equ baudrate = 9600
.equ bauddivider = FSK/(16*baudrate)-1 ;вычисление коэффициента деления
uart_init:
    LDI R16, low(bauddivider)
    OUT UBRRL, R16
    LDI R16, high(bauddivider)
    OUT UBRRH, R16
    LDI R16, 0
    OUT UCSRA, R16
    LDI R16, (1 << RXEN)|(1 << TXEN)|(0 << RXCIE)|(0 << TXCIE)
; Прерывания запрещены, прием-передача разрешены
    OUT UCSRB, R16
    LDI R16, (1 << URSEL)|(1 << UCSZ0)|(1 << UCSZ1)
; Формат кадра - 8 бит, запись в регистр UCSRC, это определяет бит
; селектор
    OUT UCSRC, R16
    RET
```

Пример процедур приема и передачи байта.

```
; Процедура ожидания готовности и прием байта
uart_rcv: SBIS    UCSRA, RXC    ; Ожидание в цикле флага прихода байта
           RJMP    uart_rcv
           IN      R16, UDR      ; Байт принят полностью - считывание
           RET                ; Выход из процедуры. Результат в R16

; Процедура ожидания готовности и передачи байта
uart_snt: SBIS    UCSRA, UDRE   ; Пропуск, если нет флага готовности
           RJMP    uart_snt      ; ожидание готовности - флага UDRE
           OUT     UDR, R16      ; Вывод байта
```

RET ; Возврат из процедуры

Пример главной процедуры приема, модификации и передачи байта

```
; Main =====
Main:          RCALL  uart_rcv          ; Ждем байта
              INC     R16              ; Выполняется некоторая операция
              RCALL  uart_snt          ; Отправляем обратно.

JMP  Main
```

3 Описание лабораторной установки и методика создания и исследования последовательного интерфейса

Лабораторная установка состоит из персонального компьютера, на котором установлены система редактирования и отладки программ микроконтроллеров Atmel Studio версий 4 – 6 и система симулирования электронных и микропроцессорных систем Proteus VSM. Состав и работа с Atmel Studio описаны в методических указаниях к лабораторным работам 1 и 3. Исследуемая в лабораторной работе схема вывода информации из микроконтроллера U1 по последовательному интерфейсу UART на виртуальный терминал VT изображена на рисунке 3.1. Для исследования структуры передаваемых по интерфейсу кадров и измерения параметров сигналов используется виртуальный осциллограф OSC.

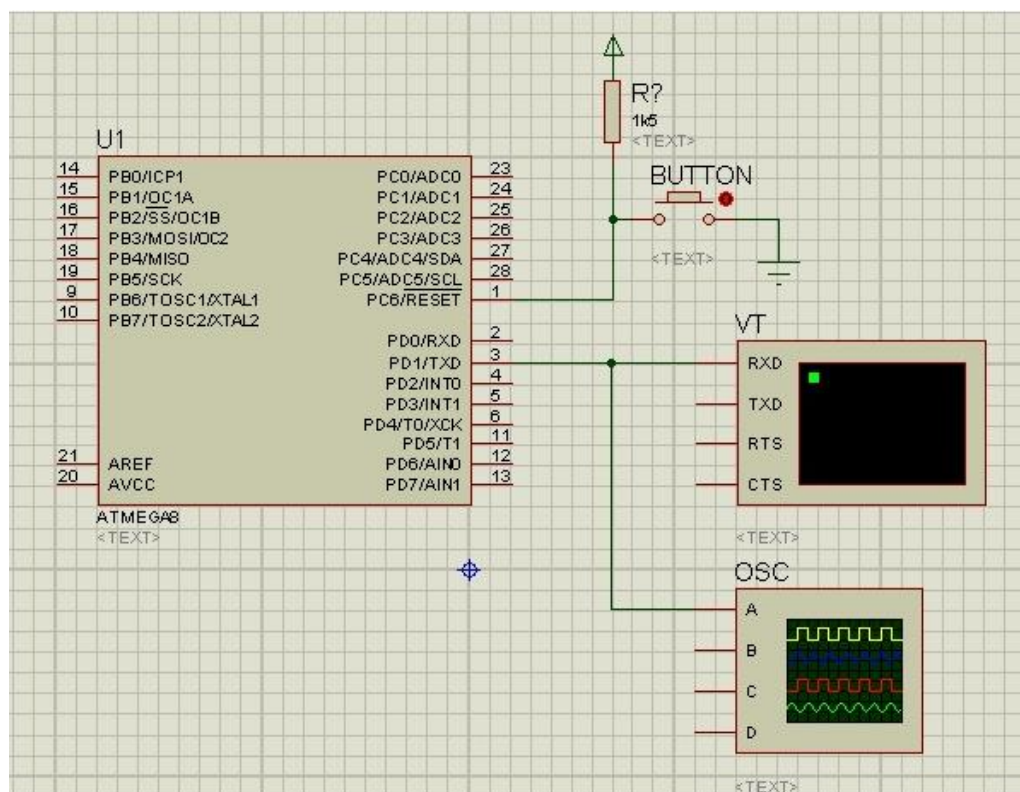


Рисунок 3.1 – Схема исследуемого устройства

4 Программа и методика выполнения лабораторной работы

4.1 Программа работы

- 4.1.1. Изучить принцип работы последовательного интерфейса USART/ UART, ознакомиться с составом программно управляемых регистров и назначением их битовых полей.
- 4.1.2. Начертить схему исследуемого устройства с указанием измерительных приборов.
- 4.1.3. Произвести расчет коэффициента деления для получения необходимой тактовой частоты и составить программу формирования и вывода заданного сообщения на виртуальный терминал согласно заданному варианту (Приложение А).
- 4.1.4. Отладить программу в среде Atmel Studio в пошаговом режиме и убедиться в правильности функционирования устройства.
- 4.1.5. Начертить исследуемую схему микропроцессорного устройства в рабочем окне симулятора Proteus VSM и загрузить исполняемый программный модуль.
- 4.1.6. Запустить процесс симуляции, убедиться, что сформированное микроконтроллером сообщение правильно отображается на экране виртуального терминала.
- 4.1.7. Получить осциллограмму передаваемого сообщения на экране виртуального осциллографа и измерить электрические и информационные параметры передаваемых сигналов.
- 4.1.8. Составить отчет о проведенных исследованиях и сформулировать выводы по работе.

ПРИМЕЧАНИЕ: Пп. программы 1-4 выполняются в процессе домашней подготовки.

4.2 Методические особенности проведения исследований.

- 4.2.1. Коды символов, входящих в передаваемое сообщение, приведены в таблице ASCII (Приложение Б).
- 4.2.2. При задании свойств виртуального терминала следует установить тактовую частоту согласно варианту.
- 4.2.3. Для исследования временной диаграммы передаваемого сообщения необходимо в виртуальном осциллографе задать режим одиночной развертки (Trigger/One-Shot).

5 Содержание отчета

- 5.6. Цель и программа работы.
- 5.7. Электрическая схема исследуемого устройства.

- 5.8. Форматы программно управляемых регистров USART с описанием назначения управляющих битов.
- 5.9. Текст программы управления процессом вывода сообщения.
- 5.10. Скриншот рабочего окна симулятора с начерченной схемой и с изображением формы сигналов на экране виртуального осциллографа.
- 5.11. Электрические и информационные параметры сигналов последовательного интерфейса.
- 5.12. Выводы по работе.

6 Контрольные вопросы

- 6.1 Какие типы последовательных интерфейсов применяются в микроконтроллерах типа AVR?
- 6.2 В чем состоит отличие синхронного и асинхронного режимов работы УСАПП?
- 6.3 Почему стоповый сигнал имеет уровень логической единицы, а не нуля?
- 6.4 По какому признаку можно определить, что передача очередного кадра завершена?
- 6.5 По какому признаку можно определить, что последовательный прием очередного кадра завершен?
- 6.6 Каким образом можно изменить скорость передачи битов?
- 6.7 Продемонстрируйте и поясните на примере разработанной схемы, каким образом в среде Atmel Studio осуществляется контроль правильности составленной программы.
- 6.8 Продемонстрируйте на примере разработанной схемы, как с помощью виртуального осциллографа можно определить параметры сигналов и вид кодовой комбинации очередного символа.

Приложение А. Варианты заданий.

Выводимое сообщение – имя студента. Если имя содержит более 5 букв, следует ограничиться первыми пятью символами.

№ Вар.	Тактовая частота МК, МГц	Скорость передачи, Бод	Символ начала сообщения	Символ окончания сообщения
0	1.00	2400	SOH	ACK
1	2.00	4800	STX	ETB
2	4.00	9600	SYN	ETX
3	8.00	19200	BEL	EOT
4	3.46	4800	BS	ENQ
5	1.84	4800	TAB	ACK
6	4.68	19200	DC1	CR
7	2.84	9600	DC2	ETB
8	4.00	38400	DC3	ESC
9	16.00	57600	DC4	EM

Приложение Б.

ASCII (American Standard Code for Information Interchange)

Dec												
0	0	NUL	32	20	(sp)	64	40	@	96	60	`	
1	1	SOH	33	21	!	65	41	A	97	61	a	
2	2	STX	34	22	"	66	42	B	98	62	b	
3	3	ETX	35	23	#	67	43	C	99	63	c	
4	4	EOT	36	24	\$	68	44	D	100	64	d	
5	5	ENQ	37	25	%	69	45	E	101	65	e	
6	6	ACK	38	26	&	70	46	F	102	66	f	
7	7	BEL	39	27	'	71	47	G	103	67	g	
8	8	BS	40	28	(	72	48	H	104	68	h	
9	9	TAB	41	29	)	73	49	I	105	69	i	
10	A	LF	42	2A	*	74	4A	J	106	6A	j	
11	B	VT	43	2B	+	75	4B	K	107	6B	k	
12	C	FF	44	2C	,	76	4C	L	108	6C	l	
13	D	CR	45	2D	-	77	4D	M	109	6D	m	
14	E	SO	46	2E	.	78	4E	N	110	6E	n	
15	F	SI	47	2F	/	79	4F	O	111	6F	o	
16	10	DLE	48	30	0	80	50	P	112	70	p	
17	11	DC1	49	31	1	81	51	Q	113	71	q	
18	12	DC2	50	32	2	82	52	R	114	72	r	
19	13	DC3	51	33	3	83	53	S	115	73	s	
20	14	DC4	52	34	4	84	54	T	116	74	t	
21	15	NAK	53	35	5	85	55	U	117	75	u	
22	16	SYN	54	36	6	86	56	V	118	76	v	
23	17	ETB	55	37	7	87	57	W	119	77	w	
24	18	CAN	56	38	8	88	58	X	120	78	x	
25	19	EM	57	39	9	89	59	Y	121	79	y	
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z	
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{	
28	1C	FS	60	3C	<	92	5C	\	124	7C		
29	1D	GS	61	3D	=	93	5D	]	125	7D	}	
30	1E	RS	62	3E	>	94	5E	^	126	7E	~	
31	1F	US	63	3F	?	95	5F	_	127	7F	DEL	

Библиографический список

1. Бальзамов А. Ю. Программирование на ассемблере для AVR-микроконтроллеров: Лаб. практикум по основам микропроцессорной техники / А. Ю. Бальзамов. – Саранск: Изд-во Мордовского ун-та, 2012. – 108 с.
2. Баранов В.Н. Применение микроконтроллеров AVR: схемы, алгоритмы, программы / В.Н. Баранов. – М.: Издательский дом «Додека-XXI», 2004. – 288 с.
3. Белов А.В. Разработка устройств на микроконтроллерах AVR: шагаем от чайников до профи / В.Н. Белов. – СПб.: Наука и техника, 2013. – 528 с.
4. Мортон Дж. Микроконтроллеры AVR. Вводный курс / Дж. Мортон / Пер. с англ.- М.: Издательский дом «Додека-XXI», 2006. – 272 с.
5. Ревич Ю.В. Практическое программирование микроконтроллеров Atmel AVR на языке ассемблера / Ю.В. Ревич. – СПб., БХВ-Петербург, 2008. – 384 с.
6. Хартов В. Я. Микроконтроллеры AVR. Практикум для начинающих: учеб. пособие / В. Я. Хартов. – 2-е изд., испр. и доп. – М. : Изд-во МГТУ им. Н. Э. Баумана, 2012. – 280 с.
7. Программирование с нуля в AVRStudio 5. <http://chipmk.ru/index.php/component/k2/item/118-programmirovaniye-s-nulya-v-avrstudio-5-ch-1>

**ИССЛЕДОВАНИЕ УСТРОЙСТВ
И МЕТОДОВ ПРОГРАММИРОВАНИЯ
ВСТРАИВАЕМЫХ МИКРОПРОЦЕССОРНЫХ СИСТЕМ**

*Методические указания
к выполнению лабораторных работ*

Составитель: Чернега Виктор Степанович

В авторской редакции

Изд. № 13/2020. Объем 5,5 п.л.
РИИЦМ ФГАОУ ВО «Севастопольский государственный университет»