

Документ подписан простой электронной подписью

Информация о владельце:

ФИО: Локтионова Оксана Григорьевна

**МИНОБРАЗОВАНИЯ И НАУКИ РОССИИ**

Должность: проректор по учебной работе

Федеральное государственное бюджетное

Дата подписания: 18.08.2024 03:35:03

образовательное учреждение высшего образования

Уникальный программный ключ:

«Юго-Западный государственный университет»

0b817ca911e6668abb15a5d426d59e5f1c11eabb173e945d14a4851fda56d089

(ЮЗГУ)

Кафедра космического приборостроения и систем связи

УТВЕРЖДАЮ  
Проректор по учебной работе  
«Юго-Западный  
государственный  
О.Г. Локтионова  
(ЮЗГУ)  
2024 г.  
«11» 06

## АНАЛОГО-ЦИФРОВАЯ ИНТЕГРАЛЬНАЯ ЭЛЕКТРОНИКА И МИКРОПРОЦЕССОРЫ

Методические указания к лабораторным занятиям. Ч. 2

Курск 2024

УДК 621.316

Составитель О. Г. Бондарь

Рецензент

Кандидат технических наук, доцент кафедры космического приборостроения и систем связи *И.Г. Бабанин*

**Аналого-цифровая электроника и микропроцессоры:** Методические указания к лабораторным занятиям. Ч. 2 / Юго-Зап. гос. ун-т; сост.: О.Г. Бондарь. – Курск : ЮЗГУ, 2024. - 77 с.

Содержатся базовые сведения по архитектуре микроконтроллеров семейства AVR, ассемблеру, техническим и программным средствам разработки программного обеспечения. Представлена информация о лабораторном стенде. Описаны последовательность и методика разработки и отладки программ на языке ассемблера. Приведены варианты заданий.

Методические указания соответствуют требованиям программы, утвержденной учебно-методическим объединением по специальностям автоматики и электроники (УМО АЭ).

Предназначены для обучающихся по направлению подготовки 11.03.02 «Инфокоммуникационные технологии и системы связи» всех форм обучения.

Текст печатается в авторской редакции

Подписано в печать 11.06.24 Формат 60×84 1/16.  
Усл. печ. л. 4,48. Уч.- изд. л. 4,05. Тираж 100 экз. Заказ 530.  
Бесплатно.

Юго-Западный государственный университет.  
305040, г. Курск, ул. 50 лет Октября, 94

## Содержание

Планируемые результаты обучения .....	5
Оборудование и программное обеспечение.....	6
Правила техники безопасности при проведении лабораторных работ .....	7
Общие требования безопасности.....	7
Требования безопасности перед началом работ.....	8
Требования безопасности во время работы .....	8
Требования безопасности по окончании работы .....	8
1. СРЕДСТВА РАЗРАБОТКИ МИКРОСИСТЕМ НА МИКРОКОНТРОЛЛЕРАХ AVR .....	10
1 Цель работы.....	10
2 Краткие сведения о микроконтроллерах AVR.....	10
2.1 Общие особенности микроконтроллеров AVR.....	10
2.2 Организация адресного пространства микроконтроллера AT90S2313 .....	20
2.3 Загружаемая память программ.....	20
2.4 EEPROM память данных .....	20
2.5 Статическое ОЗУ данных.....	21
2.6 Файл регистров общего назначения.....	22
2.7 Пространство ввода/вывода .....	23
3 Инструменты разработки и отладки программного обеспечения .....	25
3.1 Общая характеристика аппаратно-программных средств разработки программного обеспечения.....	25
3.2 Создание/загрузка проекта в ASTUDIO4.....	32
3.3 Ассемблер AVR.....	34
3.4 Отладка программы .....	37
3.5 Программирование микроконтроллера в составе прототипной системы .....	39
4 Подготовка к лабораторной работе .....	41
5 Программа исследований и порядок работы .....	44
6 Контрольные вопросы .....	45
7 Содержание отчёта .....	46

2. ОРГАНИЗАЦИЯ ЦИФРОВОГО ВВОДА/ВЫВОДА В СИСТЕМАХ НА МИКРОКОНТРОЛЛЕРАХ AVR .....	47
1 Цель работы.....	47
2 Порты ввода/вывода микроконтроллеров AVR.....	47
2.1 Организация портов ввода/вывода.....	47
2.2 Работа с портами ввода/вывода.....	49
3 Подготовка к лабораторной работе .....	50
4 Программа исследований и порядок работы .....	54
5 Методические указания .....	55
6 Контрольные вопросы .....	55
7 Содержание отчёта .....	56
3. ПОДПРОГРАММЫ И СТЕК.....	57
1 Цель работы.....	57
2 Организация стека в микроконтроллерах AVR.....	57
3 Подготовка к лабораторной работе .....	59
4 Программа исследований и порядок работы .....	63
5 Контрольные вопросы .....	64
6 Содержание отчёта .....	64
4. ОРГАНИЗАЦИЯ ПРЕРЫВАНИЙ В МИКРОКОНТРОЛЛЕРАХ AVR.	65
1 Цель работы.....	65
2 Внешние прерывания в микроконтроллерах AVR .....	65
2.1 Источники прерываний .....	65
2.2 Регистры прерываний.....	67
3 Подготовка к лабораторной работе .....	71
4 Программа исследований и порядок работы .....	74
5 Контрольные вопросы .....	75
6 Содержание отчёта .....	76
Литература.....	77

## Планируемые результаты обучения

При выполнении лабораторного цикла формируются компетенции ОПК-2, ОПК-3 и ОПК-5.

Обучающийся должен

### **знать:**

- физические основы, принцип действия элементной базы, принципы построения функциональных узлов телекоммуникационных устройств (ТКУ) и их сравнительные характеристики;
- последовательность проектирования функциональных узлов ТКУ и ожидаемые результаты на каждом этапе проектирования;
- основы архитектуры микропроцессоров (МП), микроконтроллеров (МК), их характеристики, принципы действия и организации систем на их основе, принципы и средства поддержки разработки программного обеспечения;
- принципы и средства решения задач обработки данных и управления микропроцессорными системами и микроконтроллерами;
- особенности программных и аппаратных средств разработки, их состав и возможности;
- методы обработки данных в системах управления объектами и процессами и их влияние на требования к ресурсам МК (МП).

### **уметь:**

- выбирать элементную базу функциональных узлов ТКУ;
- проводить расчёты основных функциональных узлов ТКУ и оценивать их погрешности;

- определять ресурсы необходимые для выполнения функций микропроцессорной системы (МПС) в составе ТКУ и выбирать её архитектуру и элементную базу, вычленять аппаратные и программные функции, разрабатывать алгоритмы функционирования;
- разрабатывать алгоритмы и программы обработки данных для МПС и МК;
- разрабатывать программное обеспечение в среде ASTUDIO;
- Выбирать эффективные методы обработки ориентированные на особенности архитектуры МК (МП).

**владеть:**

- техническими параметрами и характеристиками элементной базы в степени достаточной для оптимального выбора её для реализации функциональных узлов ТКУ;
- навыками расчёта функциональных узлов ТКУ и проверки их соответствия экспериментальным данным;
- навыками построения несложных средств обработки данных и управления аппаратными средствами ТКУ на основе МК и МП;
- навыками работы в среде разработки программного обеспечения;
- навыками разработки программного обеспечения для семейства МК на ядре AVR.

**Оборудование и программное обеспечение**

- ПК (Processor i5-2500, RAM DDR3 4 GB, HDD 320 GB, DVD RW, TFT-монитор 24” 1920x1080);
- стенд с МК AVR;
- программатор AVR USBASP V2;
- интегрированная среда разработки Avr Studio 4;

- программные средства поддержки загрузки ПО в МК Khazama AVR Programmer, ProgISP v.1.72;
- USBasp-win-driver-x86-x64-v3.0.7.

## **Правила техники безопасности при проведении лабораторных работ**

### **Общие требования безопасности**

1. К работе с электроизмерительными приборами, электроустановками, ЭВМ под руководством преподавателя или ответственного за лабораторию допускаются лица, прошедшие инструктаж по охране труда, медицинский осмотр и не имеющие противопоказаний по состоянию здоровья.

2. При работе в лаборатории студенты должны соблюдать правила поведения, расписание учебных занятий, установленные режимы труда и отдыха.

3. При работе с электроизмерительными приборами возможно воздействие на работающих следующих опасных факторов:

а) поражение электрическим током при прикосновении к оголенным проводам и при работе с приборами, находящимися под напряжением;

б) травмирование рук при использовании неисправного инструмента.

4. При несчастном случае пострадавший или очевидец несчастного случая обязан немедленно поставить в известность преподавателя или зав. лабораторией, который сообщает об этом администрации университета. При неисправности электроизмерительных приборов, инструмента следует прекратить работу и сообщить об этом преподавателю или зав. лабораторией.

5. Студенты, допустившие невыполнение или нарушение инструкции по охране труда, привлекаются к дисциплинарной от-

ветственности в соответствии с правилами внутреннего трудового распорядка университета и подвергаются внеочередной проверке знаний правил техники безопасности.

### **Требования безопасности перед началом работ**

1. Получив разрешение на проведение лабораторных работ, ПРОВЕРЬТЕ состояние и исправность приборов и кабелей, наличие и исправность защитного заземления.

2. Подготовьте необходимые для работы материалы, приспособления и разложите на свои места, уберите с рабочего стола все лишнее.

### **Требования безопасности во время работы**

1. ПОМНИТЕ! Электрический ток величиной 0,1 А и напряжением свыше 42 В опасен для жизни человека.

2. Лабораторные работы студенты проводят только в присутствии преподавателя или ответственного за лабораторию.

3. Включение ПК осуществлять кнопкой включения на системном блоке.

4. При сборке электрической схемы использовать провода без видимых повреждений изоляции, избегать пересечений и скручиваний проводов, питание схемы подключать в последнюю очередь.

5. Все изменения в схеме проводить при отключённом от компьютера USB-кабеле.

6. Не допускать попадания влаги на элементы схемы и компьютера.

7. Наличие напряжения в электрической цепи проверять только приборами.

8. Не допускать предельных нагрузок измерительных приборов.

9. Не оставлять без надзора невыключенные электрические устройства и приборы.

Требования безопасности по окончании работы

1. Выключить ПК программно завершением работы.



2. Отключить электроизмерительные приборы и лабораторные установки от электросети / компьютера.
3. Привести в порядок рабочее место.
4. Сообщить преподавателю или ответственному за лабораторию об окончании работы и получить разрешение на уход из лаборатории.

# 1. СРЕДСТВА РАЗРАБОТКИ МИКРОСИСТЕМ НА МИКРОКОНТРОЛЛЕРАХ AVR

## 1 Цель работы

Изучение основ архитектуры микроконтроллеров серии AVR и базовых средств разработки и отладки программного обеспечения.

## 2 Краткие сведения о микроконтроллерах AVR

### 2.1 Общие особенности микроконтроллеров AVR

Микроконтроллеры серии AVR выпускаются компанией Atmel с 1997 года. Название происходит от Alf-Egil Bogen и Vegard Wollen (фамилий разработчиков) + RISC. На архитектуре AVR изначально базировались три семейства контроллеров: "tiny", "classic" и "mega". Они отличаются объёмом памяти, составом периферийных устройств на кристалле, количеством портов ввода вывода, количеством выводов корпусов, предельными тактовыми частотами. Обобщённая структурная схема микроконтроллеров представлена на рисунке 1.

Отметим ключевые особенности платформы AVR 8-bit RISC.

- Скоростная **RISC**-архитектура Гарвардского типа с 32 регистрами общего назначения.
- **Flash**-память программ, которая может быть загружена как с помощью параллельного программатора, так и с помощью **SPI**-интерфейса (трёхпроводной последовательный интерфейс), в том числе непосредственно на целевой плате не менее 1000 раз. При этом обеспечена функциональная совместимость AVR с объёмом памяти программ от 1 до 128 кбайт.

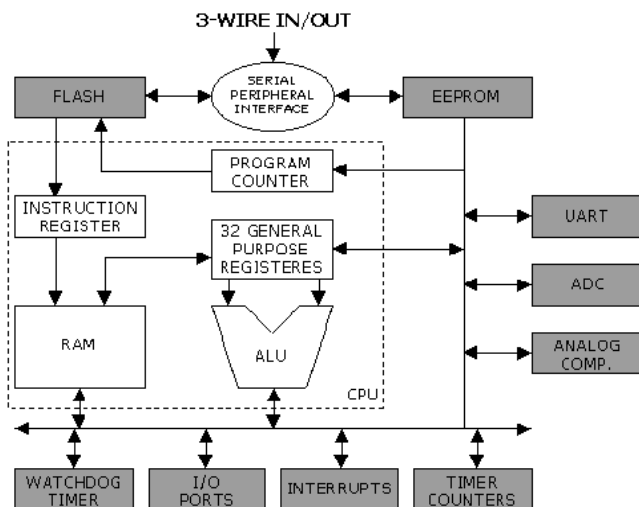


Рисунок 1- Структурная схема микроконтроллеров AVR.

- **EEPROM**-энергонезависимая электрически стираемая память для хранения промежуточных данных, различных констант, таблиц перекодировок, калибровочных коэффициентов с числом циклов перезаписи - не менее 100000.
- Внутренняя оперативная память **RAM** у всех AVR семейств "classic" и "mega" и у некоторых кристаллов семейства "tiny", с возможностью подключения для некоторых микроконтроллеров внешней памяти данных объемом до 64К.
- Важнейшая составляющая микроконтроллеров AVR - активно развивающаяся периферия, отслеживающая мировые требования к современным микроконтроллерам.
- Сторожевой (**WATCHDOG**) таймер предназначен для защиты микроконтроллера от сбоев в процессе работы. Он имеет свой собственный RC-генератор, работающий на частоте 1 МГц. Эта частота является приближенной и зависит

прежде всего от величины напряжения питания микроконтроллера и от температуры. WATCHDOG-таймер снабжен своим собственным предделителем входной частоты с программируемым коэффициентом деления, что позволяет подстраивать временной интервал переполнения таймера и сброса микроконтроллера. WATCHDOG-таймер может быть отключен программным образом во время работы микроконтроллера как в активном режиме, так и в любом из режимов пониженного энергопотребления. В последнем случае это приводит к значительному снижению потребляемого тока.

- Микроконтроллеры AVR имеют в своем составе от 1 до 4 таймеров/счетчиков общего назначения (**Timer Counters**) с разрядностью 8 или 16 бит, которые могут работать и как таймеры от внутреннего источника опорной частоты, и как счетчики внешних событий с внешним тактированием. Все таймеры/счётчики имеют программируемые предделители входной частоты с различными градациями коэффициента деления. Отличительной чертой является возможность работы таймеров/счетчиков на основной тактовой частоте микроконтроллера без предварительного ее понижения, что существенно повышает точность генерации временных интервалов системы. Они могут функционировать независимо от режима работы процессорного ядра микроконтроллера (т.е. они могут быть как считаны, так и загружены новым значением в любое время). Допускается работа от внутреннего источника опорной частоты, или в качестве счетчика событий. Верхний частотный порог определен в этом случае как половина основной тактовой частоты микроконтроллера. Выбор перепада внешнего источника (фронт или срез) программируется пользователем. Важным обстоятельством является наличие различных векторов прерываний для различных событий (переполнение, захват, сравнение).

- **I/O Ports** - порты ввода/вывода имеют от 3 до 53 независимых линий "Вход/Выход". Каждый разряд порта может быть запрограммирован на ввод или на вывод информации.
- **Analog comparator** - аналоговый компаратор входит в состав большинства микроконтроллеров AVR. Типовое напряжение смещения равно 10 мВ, время задержки распространения составляет 500 нс и зависит от напряжения питания микроконтроллера. Так, например, при напряжении питания 2,7 Вольт оно равно 750 нс. Аналоговый компаратор имеет свой собственный вектор прерывания в общей системе прерываний микроконтроллера. При этом тип перепада, вызывающий запрос на прерывание при срабатывании компаратора, может быть запрограммирован пользователем как фронт, срез или переключение. Логический выход компаратора может быть программным образом подключен ко входу одного из 16-разрядных таймеров/счетчиков, работающего в режиме захвата. Это дает возможность измерять длительность аналоговых сигналов, а также максимально просто реализовывать АЦП двухтактного интегрирования.
- **ADC** - аналого-цифровой преобразователь (АЦП) построен по классической схеме последовательных приближений с устройством выборки/хранения (УВХ). Каждый из аналоговых входов может быть соединен со входом УВХ через аналоговый мультиплексор. Устройство выборки/хранения имеет свой собственный усилитель и гарантирует, что измеряемый аналоговый сигнал будет стабильным в течение всего времени преобразования. Разрядность АЦП составляет 10 бит при нормируемой погрешности +/- 2 единицы младшего разряда. АЦП может работать в двух режимах - однократное преобразование по любому выбранному каналу и последовательный циклический опрос всех каналов. Время преобразования выбирается программно с помощью установки коэффициента

деления частоты специального предделителя, входящего в состав блока АЦП. Оно составляет 70...280 мкс для ATmega103 и 65...260 мкс для всех остальных микроконтроллеров, имеющих в своем составе АЦП. Важной особенностью аналого-цифрового преобразователя является функция подавления шума при преобразовании. Пользователь имеет возможность, выполнив короткий ряд программных операций, запустить АЦП в то время, когда центральный процессор находится в одном из режимов пониженного энергопотребления. При этом на точность преобразования не будут оказывать влияние помехи, возникающие при работе процессорного ядра.

- AVR - микроконтроллеры могут быть переведены программным путем в один из **шести режимов пониженного энергопотребления**. Для разных семейств AVR и разных микроконтроллеров в пределах каждого семейства изменяются количество и сочетание доступных режимов пониженного энергопотребления. Подробную информацию можно найти в оригинальной технической документации Atmel Corporation (Microship).

Режим холостого хода (**IDLE**), в котором прекращает работу только процессор и фиксируется содержимое памяти данных, а внутренний генератор синхросигналов, таймеры, система прерываний и WATCHDOG-таймер продолжают функционировать.

Режим микропотребления (**Power Down**), в котором сохраняется содержимое регистрового файла, но останавливается внутренний генератор синхросигналов. Выход из Power Down возможен либо по общему сбросу микроконтроллера, либо по сигналу (уровень) от внешнего источника прерывания. При включенном WATCHDOG-таймере ток потребления в этом режиме составляет около 60...80 мкА, а при выключенном - менее 1 мкА для всех

типов AVR. Вышеприведенные значения справедливы для величины питающего напряжения 5 В.

Режим сохранения энергии (**Power Save**), который реализован только у тех микроконтроллеров, которые имеют в своем составе систему реального времени. В основном, режим Power Save идентичен Power Down, но здесь допускается независимая работа дополнительного таймера/счетчика **режима реального времени (RTC)**. Выход из режима Power Save возможен по прерыванию, вызванному или переполнением таймера/счетчика RTC, или срабатыванием блока сравнения этого счетчика. Ток потребления в этом режиме составляет 6...10 мкА при напряжении питания 5 В на частоте 32,768 КГц.

Режим подавления шума при работе аналого-цифрового преобразователя (**ADC Noise Reduction**). Как уже отмечалось, в этом режиме останавливается процессорное ядро, но разрешена работа АЦП, двухпроводного интерфейса **I2C** и сторожевого таймера.

Основной режим ожидания (**Standby**). Идентичен режиму Power Down, но работа тактового генератора не прекращается. Это гарантирует быстрый выход микроконтроллера из режима ожидания всего за 6 тактов генератора.

Дополнительный режим ожидания (**Extended Standby**). Идентичен режиму Power Save, но работа тактового генератора тоже не прекращается. Это гарантирует быстрый выход микроконтроллера из режима ожидания всего за 6 тактов генератора.

- Микроконтроллеры AVR функционируют в широком диапазоне питающих напряжений от 1,8 до 6,0 Вольт. Энергопо-

ребление в активном режиме зависит от величины напряжения питания, от рабочей частоты и от конкретного типа микроконтроллера.

С точки зрения программиста AVR представляет собой 8-разрядный RISC микроконтроллер, имеющий быстрый Гарвардский процессор, память программ, память данных, порты ввода/вывода и различные интерфейсные схемы. Гарвардская архитектура AVR реализует полное логическое и физическое разделение не только адресных пространств, но и информационных шин для обращения к памяти программ и к памяти данных, причем способы адресации и доступа к этим массивам памяти также различны. Подобное построение уже ближе к структуре цифровых сигнальных процессоров и обеспечивает существенное повышение производительности. Центральный процессор работает одновременно как с памятью программ, так и с памятью данных; разрядность шины памяти программ расширена до 16 бит.

В микроконтроллерах AVR используется одноуровневый конвейер при обращении к памяти программ, и короткая команда в общем потоке выполняется за один машинный цикл. Цикл у AVR составляет всего один период тактовой частоты. При этом пиковая производительность соответствует одному MIPS/MHz (миллион инструкций в секунду на один мегагерц тактовой частоты).

Следующая отличительная черта архитектуры микроконтроллеров AVR - регистровый файл быстрого доступа, структурная схема которого показана на рисунке 2.



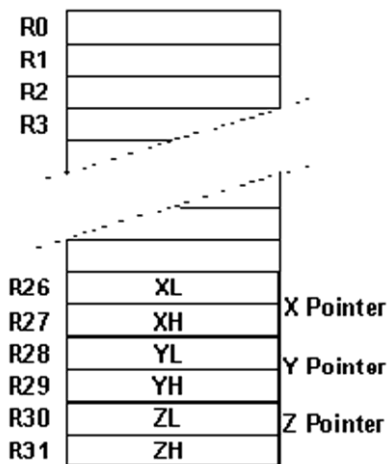


Рисунок 2- Регистровый файл AVR.

Каждый из 32-х регистров общего назначения длиной один байт непосредственно связан с арифметико-логическим устройством (ALU) процессора. Другими словами, в AVR существует 32 регистра - аккумулятора. Это обстоятельство позволяет в сочетании с конвейерной обработкой выполнять одну операцию в арифметико-логическом устройстве (ALU) за один машинный цикл. Так, два операнда извлекаются из регистрового файла, выполняется команда и результат записывается обратно в регистровый файл в течение только одного машинного цикла.

Шесть из 32-х регистров файла могут использоваться как три 16-разрядных указателя адреса при косвенной адресации данных. Один из этих указателей (**Z Pointer**) применяется также для доступа к данным, записанным в памяти программ микроконтроллера. Использование трех 16-битных указателей (X, Y и Z Pointers) существенно повышает скорость пересылки данных при работе прикладной программы.

Регистровый файл занимает младшие 32 байта в общем адресном пространстве SRAM AVR. Такое решение позволяет получать доступ к быстрой "регистровой" оперативной памяти микроконтроллера двумя путями - непосредственной адресацией в коде команды к любой ячейке и другими способами адресации ячеек SRAM. В технической документации фирмы Atmel это полезное свойство носит название "быстрое контекстное переключение". Оно является еще одной отличительной особенностью архитектуры AVR, повышающей эффективность работы микроконтроллера и его производительность (следует отметить, что принцип контекстного переключения трактуется несколько вольно, а пример его корректной реализации можно увидеть у микроконтроллеров фирмы Zilog семейств Z86 и Z8Encore). Особенно заметно данное преимущество при реализации процедур целочисленной 16-битной арифметики, когда исключаются многократные пересылки между различными ячейками памяти данных при обработке арифметических операндов в ALU.

Система команд AVR весьма развита и насчитывает до 133 различных инструкций. Почти все команды имеют фиксированную длину в одно слово (16 бит), что позволяет в большинстве случаев объединять в одной команде и код операции, и операнд(ы). Лишь немногие команды имеют размер в 2 слова (32 бит) и относятся к группе команд вызова процедуры CALL, длинных переходов в пределах всего адресного пространства JMP, возврата из подпрограмм RET и команд работы с памятью программ LPM.

Различают пять групп команд AVR: условного ветвления, безусловного ветвления, арифметические и логические операции, команды пересылки данных, команды работы с битами. В последних версиях кристаллов AVR семейства "mega" реализована функция аппаратного умножения.

По разнообразию и количеству реализованных инструкций AVR больше похожи на **CISC**, чем на **RISC** процессоры.

АЛУ поддерживает арифметические и логические операции с регистрами, с константами и регистрами. Операции над отдельными регистрами также выполняются в АЛУ.

Кроме регистровых операций, для работы с регистровым файлом могут использоваться доступные режимы адресации, поскольку регистровый файл занимает адреса \$00-\$1F в области данных, обращаться к ним можно как к ячейкам памяти.

Пространство ввода состоит из 64 адресов для периферийных функций процессора, таких как управляющие регистры, таймеры/счетчики и другие. Доступ к пространству ввода/вывода может осуществляться непосредственно, как к ячейкам памяти, расположенным после регистрового файла (\$20-\$5F).

При обработке прерываний и вызове подпрограмм адрес возврата запоминается в стеке. Стек у микроконтроллеров, имеющих ОЗУ размещается в памяти данных общего назначения, соответственно размер стека ограничен только размером доступной памяти данных и ее использованием в программе. Все программы пользователя должны инициализировать указатель стека (SP) в программе, выполняемой после сброса (до того, как вызываются подпрограммы и разрешаются прерывания). 8-разрядный указатель стека доступен для чтения/записи в области ввода/вывода.

Доступ к 128 байтам статического ОЗУ, регистровому файлу и регистрам ввода/вывода осуществляется при помощи пяти доступных режимов адресации, поддерживаемых архитектурой AVR.

Все пространство памяти AVR является линейным и непрерывным.

Гибкий модуль прерываний имеет собственный управляющий регистр в пространстве ввода/вывода, и флаг глобального разрешения прерываний в регистре состояния. Каждому преры-

ванию назначен свой вектор в начальной области памяти программ. Различные прерывания имеют приоритет в соответствии с расположением их векторов. По младшим адресам расположены векторы с большим приоритетом.

Детали архитектуры AVR зависят от типа микроконтроллера. Далее они рассматриваются на примере микроконтроллера AT90S2313.

## **2.2 Организация адресного пространства микроконтроллера AT90S2313**

Адресное пространство микроконтроллера включает в себя 3 независимых подпространства: память программ, перепрограммируемая память данных и статическое ОЗУ данных.

### **2.3 Загружаемая память программ**

AT90S2313 содержит 2кБ загружаемой флэш-памяти для хранения программ. Поскольку все команды занимают одно 16-разрядное слово, флэш-память организована как 1К 16-разрядных слов. Флэш-память выдерживает не менее 1000 циклов перезаписи. Программный счетчик имеет ширину 10 бит и таким образом адресует к 1024 словам программной флэш-памяти. Таблицы констант могут располагаться в диапазоне адресов 0-2К.

### **2.4 EEPROM память данных**

AT90S2313 содержит 128 байт электрически стираемой энергонезависимой памяти (EEPROM). EEPROM организована как отдельная область данных, каждый байт которой может быть прочитан и перезаписан. EEPROM выдерживает не менее 100000 циклов записи/стирания. Доступ к энергонезависимой памяти данных задается регистром адреса, регистром данных и управляющим регистром.

## 2.5 Статическое ОЗУ данных

На рисунке 3 показана организация памяти данных в AT90S2313.

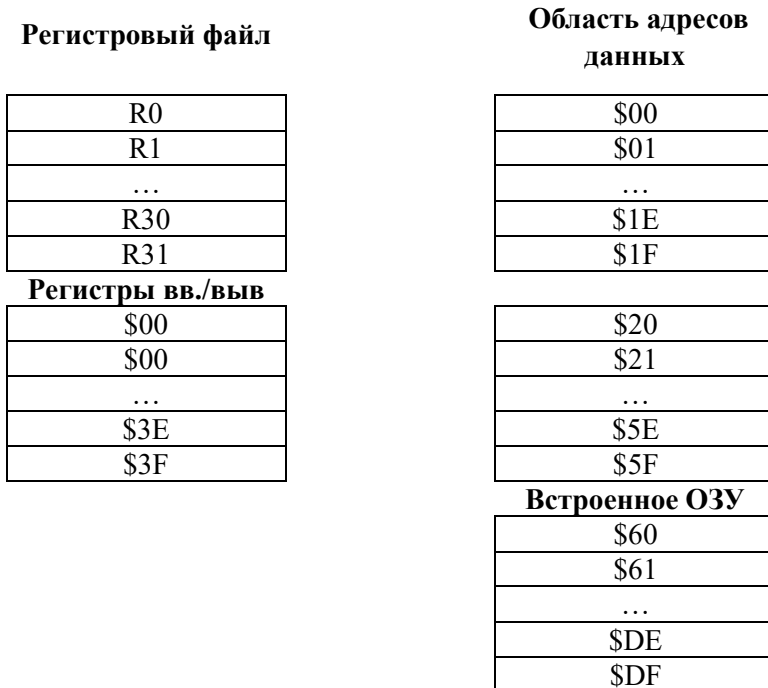


Рисунок 3- Организация памяти данных

224 ячейки памяти включают в себя регистровый файл, память ввода/вывода и статическое ОЗУ данных. Первые 96 адресов используются для регистрового файла и памяти ввода/вывода, следующие 128 - для ОЗУ данных.

## 2.6 Файл регистров общего назначения

Все команды, оперирующие регистрами прямо адресуются к любому из регистров за один машинный цикл. Исключением являются пять команд, оперирующих с константами SBCI, SUBI, CPI, ANDI, ORI и команда LDI, загружающая регистр константой. Эти команды работают только со второй половиной регистрового файла - R16-R31. Команды SBC, SUB, CP, AND и OR, также, как и все остальные, применимы ко всему регистровому файлу.

Каждому регистру присвоен адрес в пространстве данных, они отображаются на первые 32 ячейки ОЗУ. Хотя регистровый файл физически размещен вне ОЗУ, подобная организация памяти дает гибкий доступ к регистрам. Регистры X, Y и Z могут использоваться для индексации любого регистра.

Кроме обычных функций, регистры R26-R31 имеют дополнительные функции, эти регистры можно использовать как адресные указатели в области памяти данных. Эти регистры обозначаются как X, Y, Z и определены следующим образом:

Регистр X	15	0		0
	7	0	7	0
	R27(\$1B)		R26(\$1A)	
Регистр Y	15	0		0
	7	0	7	0
	R29(\$1D)		R28(\$1C)	
Регистр Z	15	0		0
	7	0	7	0
	R31(\$1F)		R30(\$1E)	

При различных режимах адресации эти регистры могут использоваться как фиксированный адрес, для адресации с автоинкрементом или с автодекрементом.

При обращении к памяти используются пять различных режимов адресации: прямой, косвенный со смещением, косвенный, косвенный с предварительным декрементом и косвенный с постинкрементом. Регистры R26-R31 регистрового файла используются как указатели для косвенной адресации.

Косвенная адресация со смещением используется для доступа к 63 ячейкам, базовый адрес которых задается содержимым регистров Y или Z. Для косвенной адресации с инкрементом и декрементом адреса используются адресные регистры X, Y и Z.

При помощи любого из этих режимов производится доступ ко всем 32 регистрам общего назначения, 64 регистрам ввода/вывода и 128 ячейкам ОЗУ.

## 2.7 Пространство ввода/вывода

Ниже приведено описание пространства ввода/вывода для процессоров AT90S2313.

Все устройства ввода/вывода и периферийные устройства AT90S2313 располагаются в пространстве ввода/вывода. Различные ячейки этого пространства доступны через команды IN и OUT, пересылающие данные между одним из 32-х регистров общего назначения и пространством ввода/вывода. К регистрам \$00..\$1F можно осуществлять побитовый доступ командами SBI и CBI. Значение отдельного бита этих регистров можно проверить командами SBIC и SBIS. Дополнительную информацию по этому вопросу можно найти в описании системы команд.

При использовании специальных команд **IN**, **OUT**, **SBIS** и **SBIC**, должны использоваться адреса **\$00-\$3F**. При доступе к регистру ввода/вывода как к ячейке ОЗУ, к его адресу необходимо добавить \$20. В приведенной ниже таблице 1 адреса регистров в памяти данных приведены в скобках.

**Таблица 1. Пространство ввода/вывода AT90S2313.**

\$3F(\$5F)	SREG	Регистр Состояния
\$3D(\$5D)	SPL	Указатель стека, мл. байт
\$3B(\$5B)	GIMSK	Общий регистр маски прерываний
\$3A(\$5A)	GIFR	Общий регистр флагов прерываний
\$39(\$59)	TIMSK	Регистр маски прерываний от таймера/счетчика
\$38(\$58)	TIFR	Регистр флага прерывания таймера
\$35(\$55)	MCUCR	Общий регистр управл. микроконтроллером
\$33(\$53)	TCCR0	Регистр управления таймером счетчиком 0
\$32(\$52)	TCNT0	Таймер/счетчик 0 (8 бит)
\$2F(\$4F)	TCCR1A	Регистр А управления таймером 1
\$2E(\$4E)	TCCR1B	Регистр В управления таймером счетчиком 1
\$2D(\$4D)	TCNT1H	Таймер 1 старший байт
\$2C(\$4C)	TCNT1L	Таймер 1 младший байт
\$2B(\$4B)	OCR1H	Выход регистра совпадения 1 старший байт
\$2A(\$4A)	OCR1L	Выход регистра совпадения 1 младший байт
\$25(\$45)	ICR1H	Регистр захвата T/C 1 старший байт
\$24(\$44)	ICR1L	Регистр захвата T/C 1 младший байт
\$21(\$41)	WDTCR	Регистр управления сторожевым таймером
\$1E(\$3E)	EEAR	Регистр адреса энергонезависимой памяти
\$1D(\$3D)	EEDR	Регистр данных энергонезависимой памяти
\$1C(\$3C)	EECR	Регистр управления энергонезавис. памятью
\$18(\$38)	PORTB	Регистр данных порта В
\$17(\$37)	DDRB	Регистр направления данных порта В
\$16(\$36)	PINB	Выводы порта В
\$12(\$32)	PORTD	Регистр данных порта D
\$11(\$31)	DDRD	Регистр направления данных порта D
\$10(\$30)	PIND	Выводы порта D
\$0C(\$2C)	UDR	Регистр данных последовательн. порта
\$0B(\$2B)	USR	Регистр состояния последоват. порта
\$0A(\$2A)	UCR	Регистр управления последовательного порта
\$09(\$29)	UBRR	Регистр скорости последовательного порта
\$08(\$28)	ACSR	Регистр управления и состояния аналогового компаратора

*Примечание: зарезервированные и неиспользуемые ячейки не показаны.*



Форматы и назначение полей регистров пространства ввода/вывода детально описаны в [2].

### **3 Инструменты разработки и отладки программного обеспечения**

Разработка программного обеспечения (ПО) нередко происходит параллельно с созданием аппаратных средств системы, что создаёт определённые трудности. Для микросистем такая ситуация скорее правило, чем исключение. На практике это означает, что ответы на некоторые вопросы, появляющиеся в процессе разработки программного обеспечения, приходится получать, прибегая к моделированию функций в программной среде чужой системы, или на прототипной системе, не полностью совпадающей с целевой системой.

Разработка ПО многоэтапная процедура. Она начинается с выбора методов решения основных задач и составления алгоритма функционирования системы. Далее осуществляется разбиение программы на отдельные модули и их написание. Важнейшим этапом, занимающим до 60-70% времени, является отладка ПО. В микросистемах на заключительном этапе **исполняемый или двоичный** код программы должен быть загружен в память программ микроконтроллера с помощью специализированных аппаратно-программных средств – программаторов.

Отладка может осуществляться в среде программного симулятора, с помощью аппаратных эмуляторов, но на последнем этапе - на целевой системе.

#### **3.1 Общая характеристика аппаратно-программных средств разработки программного обеспечения**

**Стенд**, используемый в лабораторном цикле, состоит из: платы микроконтроллера с разъёмом программирования по внут-

ренному стандарту фирмы ATMEL ICP10; подключаемых периферийных устройств; программатора; адаптера питания. Периферийные устройства подключаются к стенду стандартным 40-жильным интерфейсным кабелем IDE.

**Плата микроконтроллера** (рисунки 4, 5) содержит ИС микроконтроллера AT90S2313 с 2Кбайт перепрограммируемой памяти программ; кварцевый резонатор, определяющий тактовую частоту процессора (4 или 10 МГц в зависимости от экземпляра платы); цепь принудительного рестарта микроконтроллера с кнопкой сброса; стабилизатор напряжения питания с выходным напряжением +5 В.

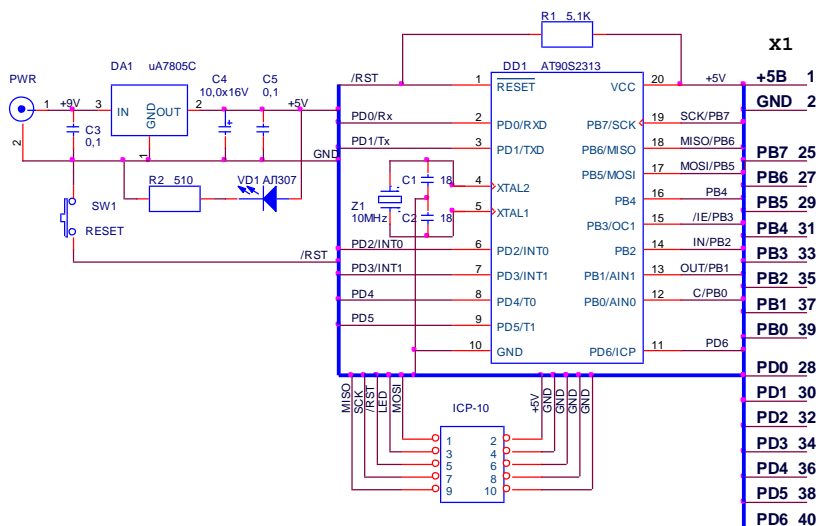


Рисунок 4 - Схема платы микроконтроллера

В состав периферийных устройств стенда входят: устройство статической индикации, устройство динамической индикации и адаптер канала RS232.

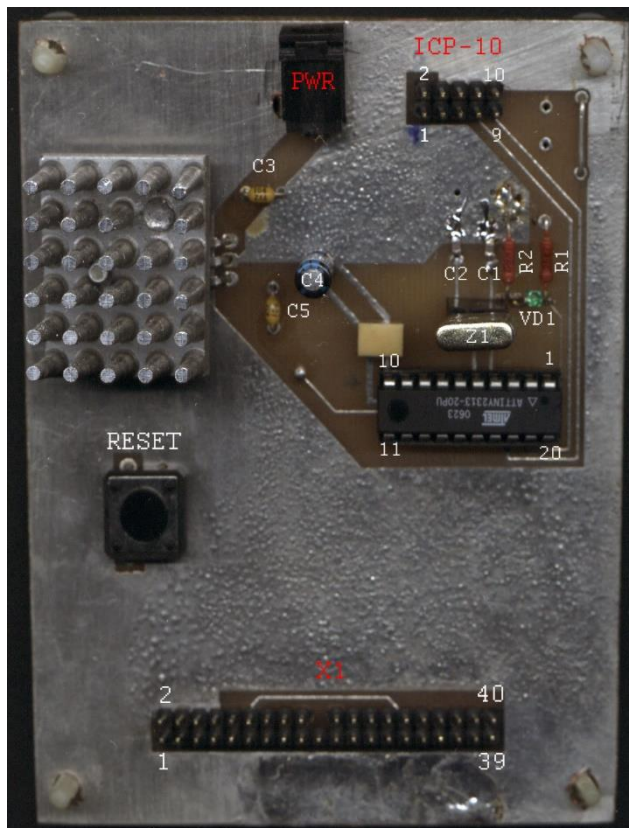


Рисунок 5 - Плата микроконтроллера.

**Устройство статической индикации** содержит 8 светодиодов и два ключа. Светодиоды подключены к порту **В** микроконтроллера. Ключи подключены к входам внешних прерываний микроконтроллера Int0, Int1 (PortD, разряды 2 и 3). Схема и внешний вид устройства представлены на рисунках 6 и 7.

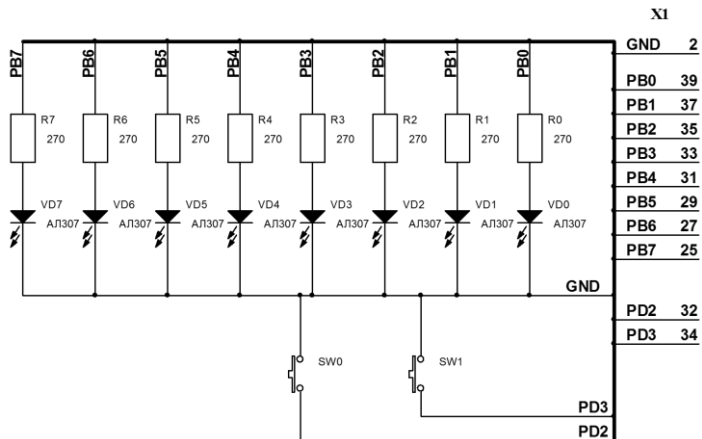


Рисунок 6 - Схема устройства статической индикации.

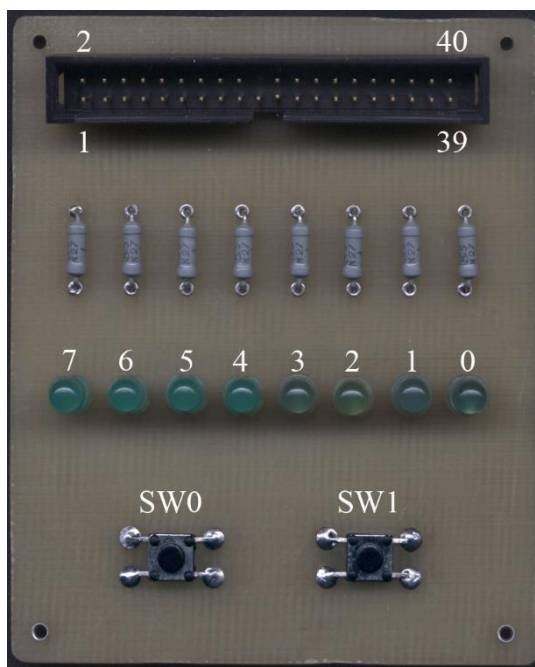


Рисунок 7 - Устройство статической индикации.

**Устройство динамической индикации** состоит: из 16-битового регистра с последовательной загрузкой на ИС DD6-DD7 (1533ИР24), пяти силовых ключей для управления разрядами индикатора на транзисторах VT1-VT5, пятиразрядного индикаторного устройства на семисегментных светодиодных индикаторах, пяти ключей S1-S5. Светодиодные индикаторы прикрыты красным светофильтром. Схема и внешний вид устройства представлены на рисунках 8 и 9, соответственно.

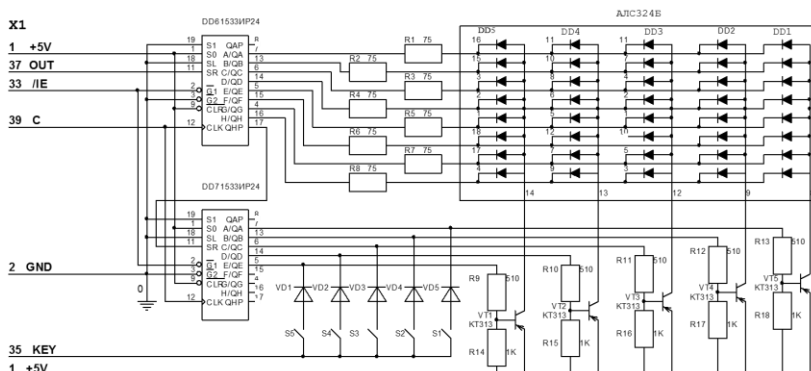


Рисунок 8-Схема устройства динамической индикации.



Рисунок 9 - Устройство динамической индикации.

**Программатор USBasp V2.0** (рисунок 10) имеет несколько исполнений. Подключение программатора к компьютеру осуществляется кабелем или непосредственно, в зависимости от конструкции. К плате микроконтроллера программатор подключается десятипроводным шлейфом. Программатор управляется программой “Khazama”. Программа имеет простой интуитивно понятный интерфейс (рисунок 10).

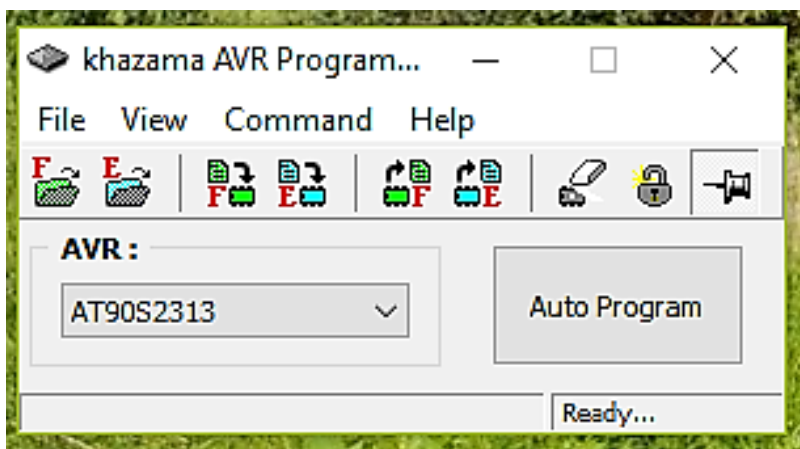
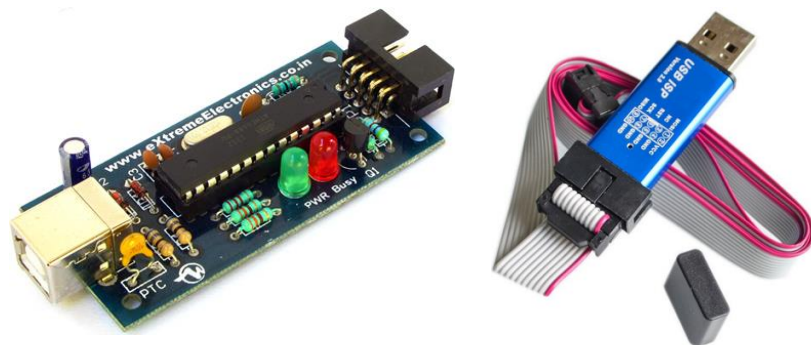


Рисунок 10 - Программатор.

Программное обеспечение состоит из интегрированной среды разработки ASTUDIO4 и программы Khazama, реализующей процедуру программирования микроконтроллера. Интегрированная среда поддерживает программирование на языке **C** и языке **ассемблера**. В лабораторном цикле используется только язык ассемблера.

Интегрированная среда поддерживает все стадии разработки: написание программы, её компиляцию, отладку в режиме симуляции и программирование конечной системы при примене-

нии фирменных программаторов. В лабораторном цикле для последней стадии используется специализированный программатор со своим программным обеспечением.

Детальная информация по ассемблеру AVR и полный перечень команд микроконтроллера приведен в фирменных документах [3,4].

## 3.2 Создание/загрузка проекта в ASTUDIO4

Запуск интегрированной среды разработки может быть осуществлён традиционными для Windows способами:

- через ярлык на рабочем столе,
- из меню программ,
- через файл описания существующего проекта с расширением .aps.

В первых двух случаях после запуска программы потребуется создать или загрузить готовый проект. Для загрузки готового проекта следует выбрать функцию открытия проекта и выбрать путь к нему, как показано на рисунке 11.

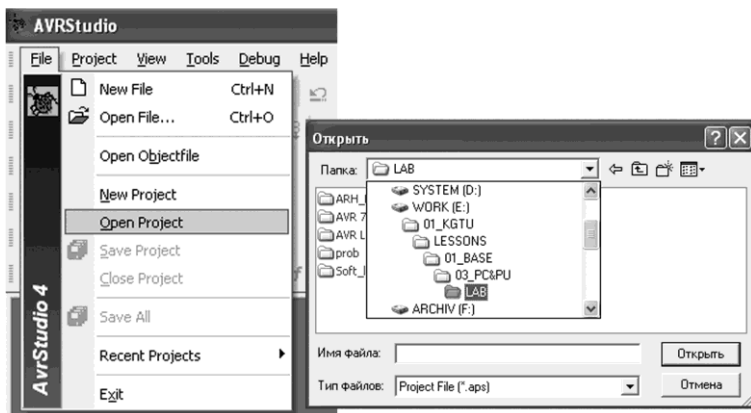
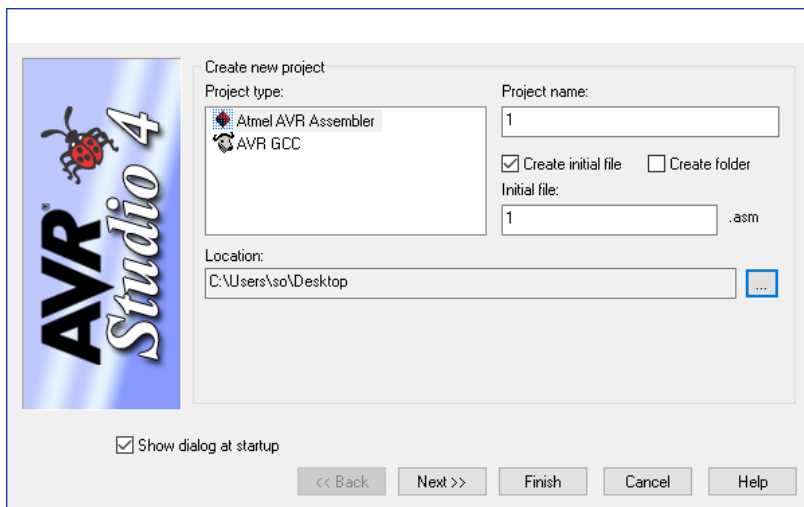


Рисунок 11- Последовательность открытия существующего проекта.



Создание нового проекта осуществляется через пункт меню **New Project** (рисунок11). При этом в всплывающем окне (рисунок12) надо выбрать имя для нового проекта и, указав путь для его размещения, перейти к следующему окну (**Next**).



Welcome to AVR Studio 4

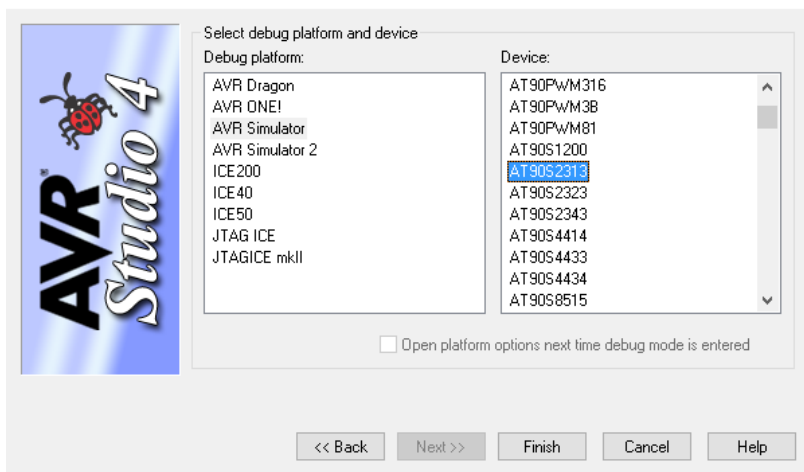


Рисунок 12 – Создание нового проекта

В следующем окне следует выбрать аппаратные средства для отладки или программный симулятор и завершить создание заготовки нового проекта (**Finish**).

После этого в основном поле программы появится текстовое окно будущей программы.

### 3.3 Ассемблер AVR

Средствами встроенного текстового редактора в нём набирается исходный текст программы в соответствии с соглашениями ассемблера. Рекомендуется типовые части текстов заимствовать из готовых проектов и далее редактировать в соответствии с особенностями проекта. Это существенно экономит время при разработке программного обеспечения. Лабораторный цикл целесообразно выполнять на основе шаблонов программ, размещаемых в методических указаниях. Текст программы может быть непосредственно скопирован в редактор среды разработки.

Ниже в качестве примера приведена небольшая программа на языке ассемблера, иллюстрирующая основные соглашения и синтаксис. Она является шаблоном для первой лабораторной работы. Программа осуществляет визуализацию содержимого временного регистра, которое циклически наращивается на единицу на каждом шаге (инкремент).

```
; ***** 01_AVR_L.asm
; визуализация содержимого циклически
; инкрементируемого регистра

.include "2313def.inc"

.def    Temp    =r16      ; рабочий регистр
.def    Delay_1  =r17      ; регистры хранения кода
.def    Delay_2  =r18      ; трёхбайтовой
.def    Delay_3  =r19      ; задержки

; Инициализация (настройка оборудования)
```

INIT:

```
ser Temp ; установка всех бит регистра в единицу
out DDRB,Temp ; PORTB ориентирован на вывод
```

```
=====
; вывод на индикацию буфера отображения Temp
```

loop:

```
out PORTB,Temp ; вывод данных в PORTB
inc Temp ; увеличение на 1 отображаемого кода
```

```
=====
; Задержка на время визуализации.
```

```
ldi Delay_1,0 ;
ldi Delay_2,0 ; задать величину задержки
ldi Delay_3,1 ; трёхбайтовым числом
```

```
=====
; задержки для визуализации результата
; реализуется декрементом переменной до заёма
```

```
; время задержки до следующего инкрементирования
; определяется количеством повторов 4-х команд цикла
; Количество повторов представлено двоичным
; трёхбайтовым числом «Delay_3 Delay_2 Delay_1»
; в данном примере  $1\ 00000000\ 00000000_2 = 100_{16} = 65536_{10}$ 
; однократное повторение цикла занимает 5 тактов МК
; при этом общее время задержки примерно 327 680 тактов
```

DLY:

```
subi Delay_1,1 ;уменьшить на единицу
sbc_i Delay_2,0 ;24-х битовый счётчик
sbc_i Delay_3,0 ;пока не произойдёт
brcc DLY ;заём, а затем
```

```
=====
rjmp loop ;визуализация нового кода
```

**Примечание:**

1. *Задержка может реализовываться вложенными циклами*
2. *В некоторые программы лабораторного цикла включены преднамеренные ошибки.*

Строка текста может иметь одну из четырёх форм:

[метка:] директива [операнды] [Комментарий]  
[метка:] инструкция [операнды] [Комментарий]  
Комментарий  
Пустая строка

Комментарий имеет следующую форму:

; [Текст]

Позиции в квадратных скобках необязательны. Текст после точки с запятой (;) и до конца строки игнорируется компилятором. Первый комментарий – имя программы.

Директивы ассемблера начинаются с точки. Первая из них подключает специальный файл, в котором описываются особенности конкретного микроконтроллера. Там же определяются типовые имена для регистров управления, имена отдельных битов и т.п. Вторая директива напрямую определяет тип устройства. Следует обратить внимание, что ASTUDIO выдаст сообщение об ошибке при компиляции проекта, если в подключаемом файле уже имеется директива определения типа устройства. В этом случае вторую директиву следует исключить из текста программы. Полезно ознакомиться с содержанием подключаемых файлов, для чего следует открыть их с помощью любого текстового редактора.

Последующие три директивы определяют имена и адреса регистров, используемых в программе (необязательно, но полезно). Использование символических имён упрощает восприятие программы и облегчает её написание и сопровождение.

После завершения написания теста программы следует выполнить компиляцию проекта через подпункт **Build** пункта основного меню **Project**. При сообщениях об ошибках изучить соответствующие строки исходной программы и исправить ошибки.

### 3.4 Отладка программы

При корректной компиляции переходят к отладке программы. В лабораторном цикле отладка выполняется в режиме симуляции (режим выбирается при создании проекта). Для этого выбирают подпункт меню “**Start Debugging**” пункта “**Debug**” основного меню. Далее в пункте основного меню программы “**View**” выбирают необходимое количество окон для отображения информации о состоянии микроконтроллера. Любое окно может быть настроено на отображение памяти данных, констант, памяти, регистров или регистров ввода вывода, как показано на рисунке 13.

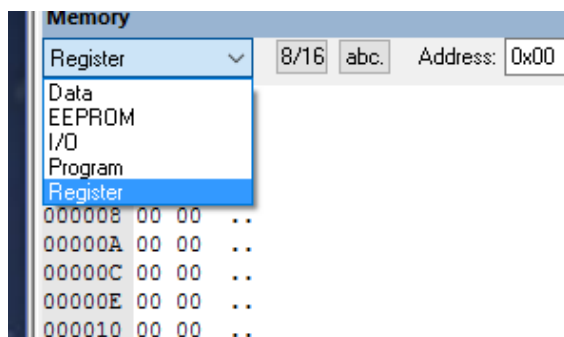


Рисунок 13 – Выбор отображаемых объектов в окне отладчика

Помимо этого, можно дополнительно включить отображение панели инструментов/вывод вывода (**Toolbars I/O**), информация в которой представляется побитово. Этот режим нагляден при работе с внешними устройствами (рисунок 14.).

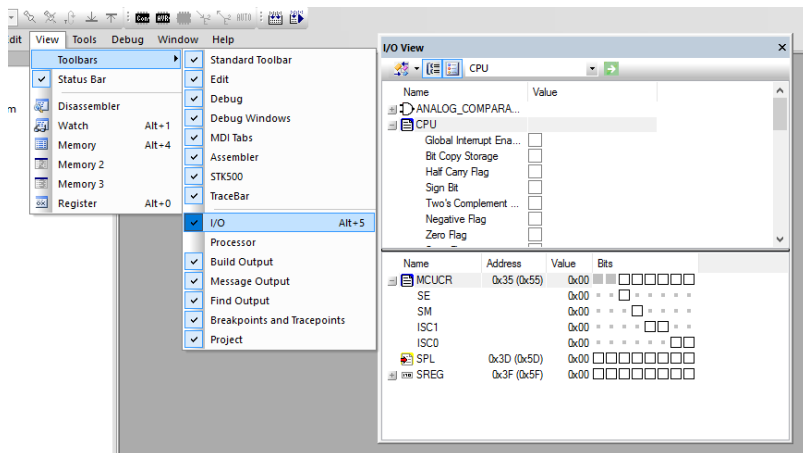


Рисунок14 – Включение отображения рабочей области.

С помощью инструментов отладки на панели инструментов или через подпункты меню “**Debug**” можно провести пошаговую отладку программы, выполнить фрагмент программы, включить автоматический пошаговый режим, установить или снять точки останова.

Выбор адекватных методов отладки и разработка тестовых примеров являются самостоятельной задачей. Кроме того, отладка на симуляторе позволяет проконтролировать и отладить логику программы, но **не исключает отладку в режиме реального времени на прототипе системы.**

В лабораторном цикле рекомендуется проводить отладку по приведенной методике.

1. Определить оценку сверху максимального количества тестов необходимых для полной проверки программы. В качестве

этой оценки (избыточной) можно взять количество различных путей от точки входа до выхода из программы.

2. Формально описать все тесты и осуществить поочерёдный прогон тестов с контролем результатов до обнаружения первой ошибки.
3. Локализовать ошибку, сочетая метод точек останова и пошаговый режим.
4. Определить тип ошибки и исправить её.
5. Продолжить тестирование, начиная с первого теста.

Программа считается отлаженной в режиме симуляции, если все тесты завершены безошибочно.

Отладчик среды разработки включает все варианты пошаговых процедур.

**Step Into** – остановка осуществляется на вызываемой процедуре, команде без её выполнения.

**Step Over** – остановка осуществляется после выполнения вызываемой процедуры, функции или команды.

**Step Out** – предназначена для выхода из функции в вызывающую функцию. Выполняется функция и осуществляется остановка на первой строке после выхода из неё.

**Auto Step** – автоматический пошаговый режим с визуализацией изменений. При его использовании рекомендуется изменить в требуемую сторону начальные значения переменных или констант, определяющих скорость выполнения программы с целью облегчения наблюдения за состоянием программы.

### 3.5 Программирование микроконтроллера в составе прототипной системы

В лабораторной работе программирование осуществляется с помощью программаторов, подключаемых к USB порту компьютера. В зависимости от версии программатор управляется программой Khazama 1.6.2, или Progisp1.72. Первая версия требует установки драйвера USBasp-win-driver-x86-x64-v3.0.7. Вто-

рой – является HID устройством и не требует установки дополнительного драйвера. Перед программированием следует подключить кабель программирования к USB порту компьютера и разъёму программирования станда. После этого можно загрузить программное обеспечение.

Для этого необходимо выбрать тип устройства AT90S2313 и загрузить файл программы в hex формате в буфер, нажав на панели инструментов иконку с изображением папки и символом **F**, и/или файл устройства с расширением e2p, если надо запрограммировать не только память программ, но и память констант, воспользовавшись иконкой с изображением папки и символом **E** (рисунок 10). Выбрать действие можно и в меню команд программатора.

С помощью панели инструментов программирования или пункта меню “**Command**” выполнить запись программы, и/или памяти констант. Прогрессор хода программирования покажет ход выполнения процедуры. В конце её будет выдано сообщение об ошибке или нормальном завершении процесса программирования. Возможные проблемы могут быть связаны с неправильным подключением кабеля программатора, переходного кабеля к разъёму программирования на плате микроконтроллера (ключ должен быть у первого контакта разъёма), шлейфа соединяющего платы микроконтроллера и устройства ввода вывода. Если все соединения выполнены правильно, следует в диспетчере устройств убедиться в наличии драйвера устройства программирования.

В Progisp1.72 hex код программы загружается в буфер через пункт меню File/Load Flash. В основном меню программы выбирается закладка PROGRAM, а далее программируемый контроллер. В опциях следует отметить Chip Erase, Program Flash и Verify Flash. Затем нажать на кнопку Auto.



## 4 Подготовка к лабораторной работе

При внеаудиторной подготовке к лабораторной работе следует придерживаться изложенного ниже плана.

1. Ознакомиться со структурой микроконтроллера AT90S2313, организацией портов ввода-вывода.
2. Подготовить отчёт к лабораторной работе в электронной форме, включив в него:
  - сведения об организации адресного пространства МК;
  - схему платы микроконтроллера и модуля индикации на 8-и светодиодах; рассчитать величину задержки и выписать задание на лабораторную работу в соответствии с вариантом;
  - текст программы из методических указаний к работе с комментариями и пояснениями;
  - указать в пояснениях к программе адреса используемых регистров настройки и собственно портов ввода/вывода микроконтроллера в десятичной и шестнадцатеричной системах счисления;
3. Разработать на основе представленной программы исходный текст программы в соответствии с вариантом задания и детально комментировать внесенные изменения и дополнения.
  - Величина задержки определяется в соответствии с выражением
 
$$T_z = 5 \times (30 \times N_1 + 15 \times N_2 + N_3) \text{ мс,}$$
 где  $N_1$  – номер группы (1 или 2),  $N_2$  – номер подгруппы (1 или 2),  $N_3$  – порядковый номер в алфавитном списке группы.
  - Отображаемые на дисплее комбинации выбираются в соответствии с таблицей 2. Неуказанные разряды должны быть всегда погашены.

4. Для каждой команды привести краткое описание выполняемых действий, формат и состояние флагов. Указать пределы изменения операндов и их конкретные значения.
5. Опираясь на комментарии в тексте программы и описание действия команд, представить программу в виде функциональных модулей и составить граф-схему алгоритма, выполненную по ГОСТ 19.701-90.
6. Изучить порядок создания проекта средствами ASTUDIO4.
9. Продумать последовательность и способ отладки каждого модуля программы. Привести план отладки.

Таблица 2. Варианты заданий

<b>№ вар.</b>	<b>Задание</b>
1.	Циклическое перемещение одиночного светящегося пятна справа налево шаг – одна позиция (бегущее световое пятно)
2.	Циклическое перемещение одиночного светящегося пятна справа налево с шагом в две позиции
3.	Циклическое перемещение одиночного светящегося пятна справа налево с шагом в три позиции
4.	Циклическое перемещение светящейся пары справа налево (шаг – одна позиция)
5.	Циклическое перемещение светящейся пары справа налево (шаг – две позиции)
6.	Циклическое перемещение светящейся пары справа налево (шаг – три позиции)
7.	Циклическое перемещение светящейся триады справа налево (шаг – одна позиция)
8.	Циклическое перемещение светящейся триады справа налево (шаг – две позиции)
9.	Циклическое перемещение светящейся триады справа налево (шаг – три позиции)
10.	Циклическое перемещение светящейся тетрады справа налево (шаг – одна позиция)

№ вар.	Задание
11.	Циклическое перемещение светящейся тетрады справа налево (шаг – две позиции)
12.	Циклическое поочерёдное включение светодиодов справа налево начиная с 0-го до 8 светящихся одновременно
13.	Циклическое поочерёдное включение светодиодов справа налево начиная с 0-го до 7 светящихся одновременно
14.	Циклическое поочерёдное включение светодиодов справа налево начиная с 0-го до 6 светящихся одновременно
15.	Циклическое поочерёдное включение светодиодов справа налево начиная с 0-го до 5 светящихся одновременно
16.	Циклическое поочерёдное включение светодиодов справа налево начиная с 1-го до 7 светящихся одновременно
17.	Циклическое поочерёдное включение светодиодов справа налево начиная с 1-го до 6 светящихся одновременно
18.	Циклическое поочерёдное включение светодиодов справа налево начиная с 1-го до 5 светящихся одновременно
19.	Циклический инкремент разрядов 1-7
20.	Циклический инкремент разрядов 2-7
21.	Циклический инкремент разрядов 3-7
22.	Циклический инкремент разрядов 0-6
23.	Циклический инкремент разрядов 0-5
24.	Циклический инкремент разрядов 0-4
25.	Циклический инкремент разрядов 1-6
26.	Циклический инкремент разрядов 2-6
27.	Циклический инкремент разрядов 3-6
28.	Циклический инкремент разрядов 1-5
29.	Циклический инкремент разрядов 2-5

№ вар.	Задание
30.	Циклический инкремент разрядов 3-5

Для самоконтроля письменно ответить на приведенные далее вопросы.

1. Как организовать длинную программную задержку?
2. Как задать режим работы конкретной линии порта ввода/вывода (ввод или вывод)?
3. Какой командой можно последовательно смещать единицу в регистре?
4. Какая команда позволяет принудительно установить некоторые биты регистра в 0?
5. Какие логические уровни соответствуют светящемуся/погашенному состояниям диода?

## 5 Программа исследований и порядок работы

Запустить программу AVR Studio, создать проект и ввести текст программы управления светодиодным индикатором, модифицированный при самостоятельной подготовке к лабораторной работе.

1. Компилировать проект и протоколировать сообщения. Если имеются ошибки, привести ошибочные строки в отчёте, сделать анализ ошибок и ввести исправления.
2. Выполнить пошаговую отладку модулей программы в соответствии с методикой, разработанной при самостоятельной подготовке. Протоколировать содержимое модифицируемых регистров при пошаговой процедуре, сопоставляя их с предсказанными значениями.
3. Опираясь на показания счётчика циклов “**Cycle Counter**” в разделе “**PROCESSOR**” окна “**View/Toolbars**” определить время

исполнения в тактах одного внутреннего и внешнего цикла задержки. Сопоставить полученные экспериментальные данные с результатами домашних расчетов.

4. По завершении отладки внести в отчёт и изучить листинг программы, находящийся в папке проекта. Сравнить транслированные адреса управления портом и регистра вывода порта с адресами, установленными при самостоятельной подготовке и результат сравнения фиксировать в отчёте.
5. Подключить стенд с помощью программирующего кабеля к параллельному порту компьютера (см. раздел *3.5 Программирование микроконтроллера в составе прототипной системы*), затем включить питание стенда, запустить программу «PopuProg» и настроить программатор.
6. Загрузить HEX-файл откомпилированной программы, скопировать дампы памяти программ из окна программатора и внести его в отчёт. Провести сопоставление с листингом программы. Фиксировать и объяснить расхождения.
7. Выполнить программирование микроконтроллера. По завершении программирования загруженная программа начнёт исполняться автоматически.
8. Описать функционирование стенда, выделив характерные особенности поведения индикатора. Сравнить соответствие исполняемых функций варианту задания и результат внести в отчёт.
9. С помощью секундомера измерить и внести в отчёт длительность цикла индикации. Сделать выводы о соответствии времени исполнения расчётному значению или о причинах расхождения. При необходимости внести коррективы в программу.

## **6 Контрольные вопросы**

1. Напишите команды, позволяющие выборочно установить некоторые линии порта В в режим ввода, а другие – вывода?

2. Почему при инициализации управляющего регистра порта вывода используется временный регистр?
3. Какие команды позволяют изменить состояние линий порта вывода микроконтроллера?
4. В чём смысл представления области памяти данных в двойной интерпретации: в виде отдельных сегментов и общего массива?

## 7 Содержание отчёта

Отчёт должен содержать:

1. титульный лист;
2. схему электрическую функциональную устройства и её краткое описание;
3. текст программы по варианту задания;
4. краткое описание всех команд разработанной программы в соответствии с п.4.3 методических указаний;
5. описание модулей программы и их функций;
6. граф-схему алгоритма;
7. план отладки программы;
8. ответы на вопросы для самопроверки;
9. протокол отладки с анализом ошибок (при наличии);
10. листинг программы с выделением адреса регистра управления вводом/выводом и адреса регистра вывода;
11. дампы памяти программ, скопированный из окна программатора и результаты сравнения его с листингом программы;
12. результаты оценки длительности задержки в среде симулятора (в тактах микроконтроллера);
13. результаты экспериментальной проверки программы на прототипной системе;
14. ответы на контрольные вопросы.

## 2. ОРГАНИЗАЦИЯ ЦИФРОВОГО ВВОДА/ВЫВОДА В СИСТЕМАХ НА МИКРОКОНТРОЛЛЕРАХ AVR

### 1 Цель работы

Изучение организации портов ввода/вывода микроконтроллеров серии AVR и приёмов работы с ними.

## 2 Порты ввода/вывода микроконтроллеров AVR

### 2.1 Организация портов ввода/вывода

Порты ввода/вывода имеют от 3 до 53 независимых линий "Вход/Выход". Каждый разряд порта может быть запрограммирован на ввод или на вывод информации. Мощные выходные драйверы обеспечивают токовую нагрузочную способность 20 мА на линию порта (втекающий ток) при максимальном значении 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 каждого физического вывода существует 3 бита контроля/управления. Упрощенная структурная схема элемента ввода/вывода AVR - микроконтроллера приведена на рисунке 1. Здесь **DDR<sub>x</sub>** - бит контроля направления передачи данных и привязки вывода к шине питания ( $V_{CC}$ ), **PORT<sub>x</sub>** - бит привязки вывода к  $V_{CC}$  и бит выходных данных, **PIN<sub>x</sub>** - бит для отображения логического уровня сигнала на физическом выводе микросхемы.

Архитектура построения портов ввода/вывода AVR с тремя битами контроля/управления позволяет разработчику полностью контролировать процесс ввода/вывода. Если необходимо получить реальное значение сигнала на физическом выводе микроконтроллера - читайте содержимое бита по адресу PIN<sub>x</sub>. Если требуется обновить выходы - прочитайте PORT<sub>x</sub> защелку и потом модифицируйте данные.

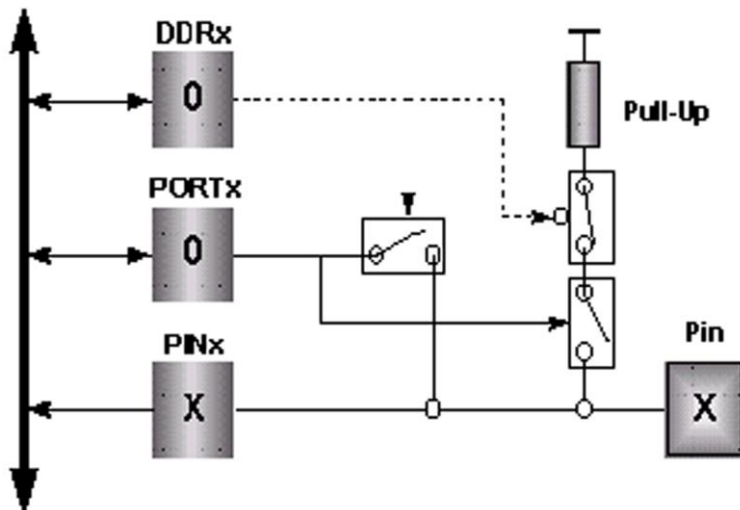


Рисунок 1 – Структура элемента ввода/вывода.

Это позволяет избежать необходимости иметь копию содержимого порта в памяти для безопасности и повышает скорость работы микроконтроллера при работе с внешними устройствами. Особую значимость приобретает данная возможность AVR для реализации систем, работающих в условиях внешних электрических помех.

Порты ввода/вывода и регистры управления ими располагаются в пространстве ввода/вывода. Они доступны через команды IN и OUT, пересылающие данные между одним из 32-х регистров общего назначения и пространством ввода/вывода.

При использовании команд IN, OUT, SBIS и SBIC, должны использоваться адреса из диапазона \$00-\$3F. При доступе к регистру ввода/вывода как к ячейке ОЗУ, к его адресу необходимо добавить \$20. В таблице 3 приведены адреса портов и регистров управления ими [1].



**Таблица 3. Порты ввода/вывода и регистры управления ими.**

\$18(\$38)	PORTB	Data Register, Port B	Регистр данных порта B
\$17(\$37)	DDRB	Data Direction Register Port B	Регистр направления данных порта B
\$16(\$36)	PINB	Input pins, Port B	Выводы порта B
\$12(\$32)	PORTD	Data Register, Port D	Регистр данных порта D
\$11(\$31)	DDRD	Data Direction Register Port D	Регистр направления данных порта D
\$10(\$30)	PIND	Input pins, Port D	Выводы порта D

## 2.2 Работа с портами ввода/вывода

**Порт B** 8-разрядный двунаправленный порт. Для обслуживания порта отведено три регистра: регистр данных PORTB, регистр направления данных - DDRB и выводы порта PINB. Адрес выводов порта B предназначен только для чтения, в то время как регистр данных и регистр направления данных - для чтения/записи.

Все выводы порта имеют отдельно подключаемые подтягивающие резисторы. Помимо этого, порт может выполнить дополнительные функции.

**PINB** не является регистром, по этому адресу осуществляется доступ к физическим значениям каждого из выводов порта B. При чтении PORTB, читаются данные из регистра-защелки, при чтении PINB читаются логические значения, присутствующие на выводах порта. После сброса начальные значения соответствуют следующим:

$PORTB = 0x00,$

$DDRB = 0x00,$

PINB = высокоимпедансное состояние.

Все 8 бит порта В при использовании для ввода/вывода равноправны. Бит DDB<sub>n</sub> регистра DDRB выбирает направление передачи данных. Если бит установлен (1), вывод сконфигурирован как выход. Если бит сброшен (0) - вывод сконфигурирован как вход. Если PORTB<sub>n</sub> установлен и вывод сконфигурирован как вход, включается КМОП подтягивающий резистор. Для отключения резистора, PORTB<sub>n</sub> должен быть сброшен (0) или вывод должен быть сконфигурирован как выход (таблица 4).

**Таблица 4. Влияние DDB<sub>n</sub> на выводы порта В**

DDRВ <sub>n</sub>	PORTВ <sub>n</sub>	Вх/Вых	Подт. резист.	Комментарий
0	0	Вход	Нет	Третье состояние (Hi-Z)
0	1	Вход	Да	РВ <sub>n</sub> источник тока I <sub>IL</sub> , если извне соединен с землей
1	0	Выход	Нет	Выход установлен в 0
1	1	Выход	Нет	Выход установлен в 1

n = 7,6...0 - номер вывода

### 3 Подготовка к лабораторной работе

При внеаудиторной подготовке к лабораторной работе придерживаться изложенного ниже плана.

1. Изучить организацию портов ввода/вывода и порядок их настройки.
2. Изучить шаблон программы, выполняющей следующие функции:
  - Если ключ “SW0” нажат, наращивать с заданным темпом счётчик и отображать на светодиодном индикаторе;

- Если ключ “SW1” нажат, уменьшать с заданным темпом счётчик и отображать на светодиодном индикаторе;
- Если не нажат ни один из ключей выключить светодиодный индикатор.

;02\_AVR\_L.asm (Цифровой ввод/вывод)

;MCU AT90S2313

;Fтакт=4 (10) МГц

;Версия 1.0

=====

;Функциональное описание

;Программа реализует бесконечный цикл сканирования  
 ;двух цифровых датчиков (ключей) и формирует  
 ;управляющие сигналы отображаемые на 8 светодиодах  
 ;подключенных к порту В

;Цикл

;Опрашиваются ключи SW0 -PinD2, SW1-PinD3.

;При замкнутом SW0 декрементируется регистр,

;содержимое которого выводится на индикацию

;При замкнутом SW1 инкрементируется регистр и

;выводится на индикацию

;Если ни один из ключей не замкнут отображаемый

;регистр сбрасывается (индикаторы гасятся)

;На основе вложенных циклов формируется

;программная задержка

=====

;Распределение управляющих линий

;Порт D. Входы D2, D3 конфигурируются на ввод.

;Остальные - безразлично.

```

;D2 - SW0 при замыкании ключа на входе 0
;D3 - SW1 "-"
;
;Порт В. Все линии конфигурируются на вывод
;D0-D7 - линии управления светодиодами (свечение - 1)
;=====
.include "2313def.inc"
;****Назначение регистров
.def Temp =r16 ;рабочий регистр
.def Test =r19 ;регистр состояния ключей
.def Delay =r17 ;счётчик задержки
.def Delay2 =r18 ;второй счётчик задержки

;=====
;***** Установка начального значения рабочих регистров
; и конфигурирование портов ввода/вывода
INIT:
    ser Temp ;Temp = $FF
    out DDRB, Temp ;PORTB = все на вывод
    out PORTB, Temp ;инициализировать PORTB
    out PORTD, Temp ;подтягив. резисторы PORTD

;=====
loop:
;Формирование отображаемого кода во
;вспомогательном регистре Temp
;Инкрементировать или декрементировать
;отображаемый регистр
;в зависимости от состояния ключей SW0 и SW1
    sbis PIND,2;SW0 нажат?
    dec Temp ;декрементировать Temp
    sbis PIND,3;SW1 нажат?
    inc Temp ;инкрементировать Temp

;Если не нажат ни один из ключей сбросить
;отображаемый регистр (погасить индикаторы)

```

```

in      Test,PIND          ;читать PORTD
andi   Test,0b00001100    ;маскировать биты
cpi    Test,0b00001100    ;оба ключа разомкнуты?
brne   outled             ;при нажатом SW0/SW1-отобразить
clr    Temp               ;при разомкнутых SW0 и SW1
                          ;сбросить отображаемый регистр.
;=====
; Отображение регистра на светодиодном индикаторе
outled:
      out   PORTB,Temp
;=====
; задержка на основе вложенных циклов
DLY:
      dec   Delay          ;вложенный цикл задержки
      brne  DLY

      dec   Delay2         ;внешний цикл задержки
      brne  DLY

      rjmp  loop           ;следующий цикл

```

3. Используя приведенный выше шаблон программы разработать программу на основе варианта задания, приведенного в работе №1. При нажатом ключе SW0 выполняется действие, соответствующее заданию, а при нажатом SW1 выполняется противоположное действие.

4. Представить программу в виде функциональных модулей с кратким описанием функции каждого модуля в текстовой форме.

5. Опираясь на представление программы в виде функциональных модулей и текст программы составить граф-схему алгоритма.

6. Продумать последовательность и способ отладки каждого модуля программы. Привести план отладки.

7. Изобразить схему электрическую функциональную аппаратных средств необходимых для выполнения данной лабораторной работы.

Для самоконтроля письменно ответить на приведенные далее вопросы.

1. В каком режиме должны работать разряды D2 и D3 порта D?
2. С какой целью для этих разрядов подключаются внутренние подтягивающие резисторы?
3. Как программно отличить состояние ключей?
4. Какие уровни соответствуют каждому из состояний?
5. Чем определяются уровни напряжений, попадающие на входы порта D2, D3?
6. С какой целью последовательно со светодиодами включают резисторы?
7. Какой логический уровень в соответствии со схемой подключения светодиодов обеспечивает их свечение.

#### **4 Программа исследований и порядок работы**

1. Создать проект и ввести текст программы, созданной при самостоятельной подготовке к лабораторной работе.
2. Компилировать проект и протоколировать сообщения. Если имеются ошибки, привести ошибочные строки в отчёте, сделать анализ ошибок и ввести исправления.
3. Выполнить отладку программы в **среде симулятора** в соответствии с методикой разработанной при самостоятельной подготовке. В процессе отладки протоколировать содержимое модифицируемых регистров, сопоставляя с предсказанными значениями.
4. Подключить стенд с помощью программирующего кабеля к параллельному порту компьютера и выполнить программирование микроконтроллера.
5. По завершении программирования загруженная программа начнёт исполняться автоматически. Нажимая поочерёдно ключи,

запротоколировать реакцию прототипного устройства в протоколе испытаний.

## 5 Методические указания

При симуляции взаимодействия ключей клавиатуры (любых устройств, подключаемых к портам ввода) и микроконтроллера программист должен самостоятельно устанавливать значение логической переменной на соответствующем контакте порта D (PIND). Для этого необходимо выполнить последовательность действий, представленную ниже.

- Запустить отладчик.
- Открыть окно рабочей области отладчика, выделив пункт View/Workspace главного меню.
- Развернуть в окне **Workspace** пункт **I/O AT90S2313**.
- Перед выполнением в пошаговом режиме команды  
in Test, PIND

следует установить значения актуальных битов щелчком на их изображении в окне. Выделенная чёрным цветом прямоугольная область соответствует логической единице, невыделенная – нулю. Симуляция внешних устройств не поддерживается. Это означает, что подключение подтягивающих резисторов не приводит к автоматической установке всех входов в состояние логической единицы.

## 6 Контрольные вопросы

1. К каким последствиям в работе программы приведёт отсутствие модуля инициализации?
2. Что изменится в работе программы, если при выборе режима ввода для порта D программно не подключить подтягивающие резисторы?
3. Почему в программе не инициирован режим ввода порта D?

4. Как обнаружить ошибку, связанную с неподключением подтягивающих резисторов порта D?

## 7 Содержание отчёта

Отчёт должен содержать:

1. титульный лист;
2. наименование работы и цель исследований;
3. схему электрическую функциональную устройства и её краткое описание;
4. исходный текст программы;
5. описание модулей программы и их функций;
6. граф-схему алгоритма;
7. план отладки программы;
8. ответы на вопросы для самопроверки;
9. текст программы в соответствии с вариантом задания;
10. протокол отладки с анализом ошибок (при наличии);
11. листинг программы;
12. дампы программной памяти микроконтроллера;
13. результаты экспериментальной проверки программы на прототипной системе в виде протокола испытаний;
14. ответы на контрольные вопросы.



### 3. ПОДПРОГРАММЫ И СТЕК

#### 1 Цель работы

Изучение особенностей организации и использования стека и механизма подпрограмм в семействе микроконтроллеров AVR.

#### 2 Организация стека в микроконтроллерах AVR

Как известно, стековый механизм организации работы с памятью существенно повышает скорость обмена данными между памятью и регистрами за счёт применения метода косвенно-регистрационной адресации и автоинкрементного, автодекрементного способа модификации адреса. Память стека может быть изолированной (предназначенной исключительно для сохранения данных и адресов возврата в основную программу) или представлять собой часть общего поля памяти данных. Кроме того, стек может размещаться во внешней или внутренней памяти микроконтроллеров.

Важнейшей составной частью стекового механизма памяти является регистр косвенной адресации, именуемый указателем стека. В общем случае для микроконтроллеров AVR указатель представляет собой 16-битовый регистр. Он состоит из двух 8-битовых регистров – младшего SPL и старшего SPH. При этом объём адресуемой памяти может составлять до 64 Кбайт. В младших моделях микроконтроллеров с объёмом памяти данных, не превышающим 256 байт, присутствует лишь однобайтовый регистр SPL.

Указатель стека процессора AT90S2313 8-разрядный регистр с адресом \$3D (\$5D). 8-ми разрядов достаточно для адресации ОЗУ в пределах \$60-\$DF. Его содержимое доступно для чтения и записи. Указатель стека указывает на область памяти (на конкретную ячейку, именуемую вершиной стека), в которой расположен стек вызова подпрограмм и прерываний. Область стека

в ОЗУ должна быть задана до того, как произойдет любой вызов подпрограммы или будут разрешены прерывания.

Указатель стека уменьшается на 1 при записи данных в стек командой PUSH и уменьшается на 2 при вызове подпрограммы командой CALL или обработке прерывания. Указатель стека увеличивается на 1 при выборе данных из стека командой POP и увеличивается на 2 при выполнении команд возврата из подпрограммы или обработчика прерывания (RET или RETI). Такой подход к управлению содержимым указателя стека принят в большинстве микроконтроллеров.

Основной проблемой при использовании стека является перекрытие области данных (программ) и стека. При этом могут искажаться и данные, и содержимое стека. Следствием являются ошибки при обработке данных или полная неработоспособность системы. Последняя возникает при затирании адресов возврата из подпрограмм или обработчиков прерываний. Для корректной работы стека нужно выполнить несколько условий.

1. При инициализации указателя стека вершину стека задают так, чтобы заполняющийся и освобождающийся стек ни при каких обстоятельствах не перекрывался с областью памяти, содержащей данные (или программу в микроконтроллерах принстонского класса). Одним из разумных подходов к этой проблеме состоит в заполнении памяти данными с младших адресов и организации стека в старших адресах. В этом случае при инициализации в указатель стека заносится адрес последней ячейки оперативной памяти.
2. Размер стека не должен превышать объёма свободной оперативной памяти. Размер определяется количеством сохраняемых в стеке данных при работе программ и так называемой **вложенностью** подпрограмм и прерываний. Вложенность – это использование подпрограмм внутри подпрограмм или обработка прерывания во время работы другого обработчика прерывания. По возможности следует запрещать вложенные

прерывания или ограничиваться двумя уровнями вложенности, разрешая прерывания внутри прерываний лишь для одного – критического события. Проблема вложенности может стоять очень остро для систем с событийным механизмом обслуживания (по прерываниям) на основе микроконтроллеров с малым объёмом оперативной памяти.

3. Причиной безудержного роста стека могут быть ошибка программиста, состоящая в нарушении правила **парности стекowych операций**. В соответствии с ним каждому вызову подпрограммы должен соответствовать возврат из неё, каждому прерыванию – возврат из прерывания, каждому оператору записи в стек – оператор чтения стека.

### 3 Подготовка к лабораторной работе

При внеаудиторной подготовке к лабораторной работе придерживаться изложенного ниже плана.

1. Изучить и описать формат и действия, выполняемые командами CALL, RET, PUSH, POP.
2. Выяснить и записать разрядность и адреса указателя стека микроконтроллера AT90S2313 в шестнадцатеричной системе счисления.
3. Уточнить и записать объём, адрес первой и последней ячейки оперативной памяти микроконтроллера в шестнадцатеричной системе счисления.
4. Изучить шаблон программы для устройства, изученного в лабораторной работе №2, выполняющей следующие функции:
  - Если ключ “SW0” нажат, отображать на светодиодном индикаторе пятно, бегущее слева – направо;
  - Если ключ “SW1” нажат, отображать на светодиодном индикаторе пятно бегущее справа – налево;

- Если не нажат ни один из ключей, не изменять изображение.

;Применение подпрограмм

;по нажатии SW1 (PortD3)- бегущий огонь влево, SW0(D2)-вправо

.include "2313def.inc"

; Fтакт = 10МГц

```
.def    Temp    =r16           ;временный регистр
.def    Test    =r20           ;временный регистр контроля
.def    Delay   =r17           ;младший регистр задержки
.def    Delay2  =r18           ;второй регистр задержки
.def    Delay3  =r19           ;старший регистр задержки
```

;Инициализация

RESET:

```
<AVR Instruction> ;установить вершину
<AVR Instruction> ;стека

ser    Temp      ;Temp = $FF
out    DDRB, Temp ;PORTB = все на вывод
<AVR Instruction>
out    PORTB, Temp ;погасить светодиоды
<AVR Instruction> ;подтягив. резисторы PORTD
<AVR Instruction> ;в регистре бегущих огней

ldi    Temp,1     ;подготовить LED0 к зажиганию
```

;\*\*\*\* чтение порта клавиатуры и формирование кода ключа

in\_kbd:

```
in     Test, PIND ;читать состояние клавиатуры
andi   Test, 12  ;наложение маски <0000 1100>
```

;\*\*\*\* анализ кодов ключей и подготовка к индикации

L\_Test:

```
cpi    Test, 4    ;ключ SW1 нажат?
brne   L2_Test    ;нет - проверять дальше

lsl    Temp       ;бегущий огонь влево
brne   LE_Test    ;если обнулится,
```

```

ldi    Temp, 1      ;переместить 1 в младший разряд
rjmp   LE_Test     ;и завершить подготовку

L2_Test:
cpi    Test, 8     ;ключ SW0 нажат?)
brne   LE_Test     ;нет - завершить подготовку

<AVR Instruction> ;бегущий огонь вправо
<AVR Instruction> ;не нуль завершить, иначе
<AVR Instruction> ;переместить 1 в старший разряд

LE_Test:           ;подготовка к индикации завершена

```

; отобразить бегущий огонь (с учётом 1-горит,0-погашен)

outled:

```

<AVR Instruction> ;отобразить на индик. регистр Temp

```

```

ldi    Delay,255   ; задать
ldi    Delay2,255  ; трёхбайтовую
ldi    Delay3,2    ; задержку

```

```

rcall   DLY        ;вызвать подпрограмму
rjmp   in_kbd     ;повторять бесконечно

```

;\*\*\*\* подпрограммы

;\*\*\*\* задержка для визуализации результата

DLY:

```

subi   Delay,1     ;уменьшить на единицу
sbc    Delay2,0    ;24-х битовый счётчик
sbc    Delay3,0    ;задержки и если он не обнулится
brcc   DLY         ;повторять до обнуления, а затем
ret    ;возврат из подпрограммы

```

- Используя шаблон программы вставить необходимые инструкции микроконтроллера AVR (описание инструкций приведено в [2]).

6. Представить программу в виде функциональных модулей с кратким описанием функции каждого модуля в текстовой форме.
7. Изучить используемый в программе модуль формирования задержки и детально (покомандно) описать его.
8. Опираясь на представление программы в виде функциональных модулей и текст программы составить граф-схему алгоритма.
9. Продумать последовательность и способ отладки каждого модуля программы. Привести план отладки.

Для самоконтроля письменно ответить на приведенные далее вопросы.

1. В каком режиме работает порт D – ввода или вывода?
2. Как можно загрузить указатель стека? Написать команды загрузки.
3. Каков адрес последней ячейки оперативной памяти контроллера AT90S2313?
4. В чём смысл наложения маски на код, считанный с порта в модуле `scan_cod: (andi Test,12)`?
5. Насколько, и в какую сторону изменится указатель стека при выполнении команды `rcall`?
6. Как рассчитать величину задержки в подпрограмме задержки?

## 4 Программа исследований и порядок работы

1. Рассчитать шестнадцатеричный код задержки для величины задержки  $T = 300 + 50 \times n$  мс, где  $n$  – номер компьютера.
2. Создать проект и ввести текст программы, созданной при самостоятельной подготовке к лабораторной работе.
3. Содержимое регистров задержки установить в соответствии с расчётами.
4. Компилировать проект и протолировать сообщения. Если имеются ошибки, привести ошибочные строки в отчёте, сделать анализ ошибок и ввести исправления.
5. Выполнить отладку модулей программы в соответствии с методикой разработанной при самостоятельной подготовке. В процессе отладки протолировать содержимое модифицируемых регистров, сопоставляя с предсказанными значениями.
6. После сброса в пошаговом режиме наблюдать и протолировать изменение содержимого стековой области и указателя стека. Для сокращения количества шагов задержку установить равной 1.
7. Дополнить подпрограмму задержки двумя командами сохранения содержимого произвольных регистров в стеке. Предпринять попытку выполнения нескольких циклов вывода на индикацию. Протолировать содержимое указателя стека, стековой области и последовательности выполняемых операторов. Объяснить результаты наблюдения (задержка должна быть равна 1).
8. Восстановить программу и установить величину задержки в соответствии с вариантом. Провести оценку реальной задержки.
9. Подключить стенд с помощью программирующего кабеля к параллельному порту компьютера и выполнить программирование микроконтроллера.
10. По завершении программирования загруженная программа начнёт исполняться автоматически. Нажимая поочерёдно клавиши, запротолировать реакцию прототипного устройства.

## 5 Контрольные вопросы

1. К каким последствиям в работе программы приведёт отсутствие команд инициализации стека?
2. Если вершина стека указывает на младшую ячейку ОЗУ, что произойдёт при двух последующих загрузках стека?

## 6 Содержание отчёта

Отчёт должен содержать:

1. титульный лист;
2. наименование работы и цель исследований;
3. исходный текст программы;
4. описание модулей программы и их функций;
5. граф-схему алгоритма;
6. план отладки программы;
7. ответы на вопросы для самопроверки;
8. текст программы в соответствии с вариантом задания;
9. протокол отладки с анализом ошибок (при наличии);
10. результаты исследований и анализа работы стека;
11. дампы памяти программ;
12. результаты экспериментальной проверки программы на прототипной системе;
13. ответы на контрольные вопросы.



## 4. ОРГАНИЗАЦИЯ ПРЕРЫВАНИЙ В МИКРОКОНТРОЛЛЕРАХ AVR

### 1 Цель работы

Изучение подсистемы прерываний и организации внешних прерываний.

## 2 Внешние прерывания в микроконтроллерах AVR

### 2.1 Источники прерываний

В AT90S2313 предусмотрены десять источников прерываний. Эти прерывания и сброс имеют различные векторы в области памяти программ. Каждому из прерываний присвоен отдельный бит разрешающий данное прерывание при установке бита в 1, если бит *I* регистра состояния **SREG** разрешает общее обслуживание прерываний.

Самые младшие адреса памяти программ определены как векторы сброса и прерываний. Полный список векторов прерываний приведен в таблице 5. Этот список определяет и приоритет различных прерываний. Меньшие адреса соответствуют более высокому уровню приоритета. Самый высокий уровень у сброса, следующий приоритет у INT0 - внешнего запроса прерывания 0 и т.д.

Таблица 5. Сброс и векторы прерываний.

Номер вектора	Адрес	Источник	Описание прерывания
1	\$000	RESET	Вывод сброса и сброс от сторожевого таймера
2	\$001	INT0	Внешнее прерывание 0

3	\$002	INT1	Внешнее прерывание 1
4	\$003	TIMER1 CAPT1	Захват таймера/счетчика 1
5	\$004	TIMER1 COMP1	Совпадение таймера/счетчика 1
6	\$005	TIMER1 OVFI	Переполнение таймера/счетчика 1
7	\$006	TIMER0, OVFO	Переполнение таймера/счетчика 0
8	\$007	UART RX	Последовательный порт: прием закончен
9	\$008	UART UDRE	Последовательный порт: регистр данных пуст
10	\$009	UART TX	Последовательный порт: передача закончена
11	\$00A	ANA_COMP	Аналоговый компаратор

Чаще всего используется следующая установка векторов прерываний в программе:

Адрес	Код	Комментарий
\$000	rjmp RESET	; Обработка сброса
\$001	rjmp EXT_INT0	; Обработка IRQ0
\$002	rjmp EXT_INT1	; Обработка IRQ1
\$003	rjmp TIM_CAPT1	; Обработка захвата таймера 1
\$004	rjmp TIM_COMP1	; Обработка совпадения таймера 1
\$005	rjmp TIM_OVFI	; Обработка переполнен. таймера 1
\$006	rjmp TIM_OVFO	; Обработка переполнен. таймера 0
\$007	rjmp UART_RXC	; Обработка приема байта
\$008	rjmp UART_DRE	; Обработка освобождения UDR
\$009	rjmp UART_TXC	; Обработка передачи байта
\$00A	rjmp ANA_COMP	; Обработка реакции компаратора
		; Начало основной программы

## 2.2 Регистры прерываний

AT90S2313 имеет два регистра маски прерываний GIMSK - общий регистр маски прерываний, расположенный по адресу \$3B(\$5B) и TIMSK - регистр маски прерываний от таймера/счетчика - по адресу \$39(\$59).

Когда возникает прерывание, общий бит разрешения прерываний **I** регистра **SREG** очищается (ноль) и прерывания запрещаются. Программа пользователя может установить этот бит для разрешения прерываний. Флаг разрешения прерываний **I** устанавливается в 1 при выполнении команды выхода из прерывания - RETI.

Для прерываний, включаемых статическими событиями (т.е. переключаемыми уровнем) (например, совпадение значения счетчика/таймера 1 с регистром совпадения) флаг прерывания взводится, когда происходит событие. Если флаг прерывания очищен и присутствует условие возникновения прерывания, флаг не будет установлен, пока не произойдет следующее событие.

Когда программный счетчик устанавливается на текущий вектор прерывания для обработки прерывания, соответствующий флаг, сгенерированный прерыванием, аппаратно сбрасывается. Некоторые флаги прерывания могут быть сброшены записью логической единицы в бит соответствующий флагу. Назначение битов общего регистра маски прерываний GIMSK представлено на рисунке 1.

	Бит	7	6	
\$3B(\$5B)		<b>INT</b>	<b>INT0</b>	GIM
Чт./зап. (R/W)		<b>1</b>		SK
Начальн. знач.		R/W	R/W	
		0	0	

Рисунок 1 – Назначение битов регистра маски прерываний.

Бит 7 – INT1: Запрос внешнего прерывания 1 разрешен. Когда этот бит установлен, а также установлен бит **I** регистра состояния, разрешается прерывание от внешнего вывода. Биты управления запуском прерывания (ISC11 и ISC10) в регистре управления микроконтроллером (MCUCR) определяют - по какому событию обрабатывается прерывание. Это может происходить по спадающему или нарастающему фронту или же по уровню. При возникновении прерывания выполняется программа, начинающаяся с адреса \$002 в памяти программ.

Бит 6 - INT0: Запрос внешнего прерывания 0 разрешен. Когда этот бит установлен, а также установлен бит **I** регистра состояния, разрешается прерывание от внешнего вывода. Биты управления запуском прерывания (ISC01 и ISC00) в регистре управления микроконтроллером (MCUCR) определяют - по какому событию обрабатывается прерывание. Это может происходить по спадающему или нарастающему фронту или же по уровню. Если вывод INT0 используется для работы с внешним источником прерывания, бит D2 в регистре направления данных порта D (DDRД), должен быть сброшен в 0, чтобы вывод INT0 работал как вход. При возникновении прерывания выполняется программа, начинающаяся с адреса \$001 в памяти программ.

Биты 5..0 - в AT90S2313 зарезервированы и читаются как 0.

При возникновении внешних прерываний устанавливаются биты в регистре флагов прерываний – GIFR (рисунок 2).

Бит	7	6	
\$3A(\$5A)	<b>INTF</b>	<b>INTF</b>	GIFR
Чт./зап. (R/W)	<b>1</b>	<b>0</b>	
Начальн. знач.	R/W	R/W	
	0	0	

Рисунок 2 – Назначение битов регистра флагов прерываний.

Бит 7 - INTF1: Флаг внешнего прерывания 1: При возникновении на выводе INT1 события, вызывающего прерывание, INTF1 устанавливается в "1". Если установлены бит I регистра SREG и бит INT1 в GIMSK, происходит переход на вектор прерывания по адресу \$002. Флаг очищается после выполнения обработчика прерывания. Кроме того, флаг можно очистить, записав в него логическую единицу.

Бит 6 - INTF0: Флаг внешнего прерывания 0: При возникновении на выводе INT0 события вызывающего прерывание, INTF0 устанавливается в "1". Если установлены бит I регистра SREG и бит INT0 в GIMSK, происходит переход на вектор прерывания по адресу \$001. Флаг очищается после выполнения обработчика прерывания. Кроме того, флаг можно очистить, записав в него логическую единицу.

Биты 5..0 - в AT90S2313 зарезервированы и читаются как 0.

Регистр MCUCR содержит биты общего управления микроконтроллером. Показаны только биты, отвечающие за способ включения внешних прерываний (рисунок 3).

Бит	3	2	1	0	
\$35(\$55)	<b>ISC1</b>	<b>ISC1</b>	<b>ISC0</b>	<b>ISC0</b>	MCUCR
Чт./зап. (R/W)	1	0	1	0	
Начальн.знач.	R/W	R/W	R/W	R/W	
	0	0	0	0	

Рисунок 3 – Назначение битов регистра управления.

Биты 7,6 - зарезервированы. В AT90S2313 эти биты зарезервированы и всегда читаются как 0.

Бит 5 - Sleep Enable -Разрешение режима Sleep. Этот бит должен быть установлен в 1, чтобы при выполнении команды

SLEEP процессор переходил в режим пониженного энергопотребления (Sleep). Для использования режима пониженного энергопотребления этот бит рекомендуется устанавливать в 1 до исполнения команды SLEEP.

Бит 4 - Sleep Mode - Режим Sleep. Этот бит выбирает один из доступных режимов пониженного энергопотребления. Если бит сброшен (0), то в качестве режима Sleep выбирается холостой режим (Idle mode). Если бит установлен, - выбирается экономичный режим (Power down). При старте биты 7-4 сброшены.

Биты 3,2 - ISC11, ISC10: биты управления срабатыванием прерывания 1: Внешнее прерывание активируется выводом INT1, если установлен флаг I регистра состояния SREG и установлена соответствующая маска в регистре GIMSK.

Биты 1,0 - ISC01, ISC00: биты управления срабатыванием прерывания 0: Внешнее прерывание активируется выводом INT0, если установлен флаг I регистра состояния SREG и установлена соответствующая маска в регистре GIMSK. Установка битов для задания срабатывания по уровню и фронтам осуществляется в соответствии с таблицей 6.

**Таблица 6. Выбор способа включения прерываний**

ISCn1	ISCn0	Описание
0	0	Запрос прерывания генерируется по низкому уровню на входе INTn
0	1	Зарезервировано
1	0	Запрос на прерывание по спадающему фронту на входе INTn
1	1	Запрос на прерывание по нарастающему фронту на входе INTn

**ПРИМЕЧАНИЕ:** При изменении битов ISCn1/ISCn0 прерывание INTn должно быть запрещено очисткой соответствующего

*бита в регистре GIMSK. Иначе прерывание может возникнуть во время изменения битов.*

### 3 Подготовка к лабораторной работе

При внеаудиторной подготовке к лабораторной работе придерживаться изложенного ниже плана.

1. Построить схему микроконтроллерного устройства с индикатором в виде 8 светодиодов, подключенных к порту **В** микроконтроллера и клавиатуры из двух кнопок, подключенных к разрядам **D2-D3** порта **D**.
2. Изучить организацию начальной области памяти программ и выписать адреса векторов внешних прерываний INT0 и INT1.
3. Изучить шаблон программы, выполняющей следующие функции:
  - При каждом нажатии на ключ “SW0” перемещать на одну позицию вправо световое пятно на светодиодном индикаторе (циклически);
  - При каждом нажатии на ключ “SW1” перемещать на одну позицию влево световое пятно на светодиодном индикаторе (циклически);
  - Если не нажат ни один из ключей, не изменять изображение.

\*\*\*\*\* Ввод данных по прерываниям

```
.include "2313def.inc"
```

```
.def    Temp    =r16    ;временный регистр
.def    Test    =r20    ;временный регистр контроля

.def    Delay   =r17    ;младший регистр задержки
.def    Delay2  =r18    ;второй регистр задержки
.def    Delay3  =r19    ;старший регистр задержки
```

```

.cseg
.org $00
    rjmp start      ;на начало программы
    rjmp L_SW0     ;обработчик прерываний от ключа SW0
    rjmp L_SW1     ;обработчик прерываний от ключа SW1

; Инициализация
start:
    ser    Temp      ;Temp = $FF
    out    DDRB,Temp ;PORTB = все на вывод
    out    PORTB,Temp ;погасить светодиоды
    out    PORTD,Temp ;подтягив. резисторы PORTD
    ldi    Temp,0b00001010 ;включать внешние прерывания
    out    MCUCR,Temp ;по спаду на D3,D2

    <AVR Instruction> ;установить вершину
    <AVR Instruction> ;стека

    ldi    Temp,0b11000000 ;разрешение внешних
    out    GIMSK,Temp      ;прерываний
    ldi    Temp,1          ;активизировать один бит
                                ;в регистре бегущих огней
    sei                                ; разрешить прерывания

loop:
    rjmp   loop

;обработка прерываний при замыкании ключа SW0
L_SW0:
    lsr    Temp      ;бегущий огонь вправо
    brne  L1_SW0    ;если обнулится,
    ldi    Temp,128 ;переместить 1 в старший разряд
L1_SW0:
;отобразить бегущий огонь (с учётом 0-горит,1-погашен)
    <AVR Instruction> ;инвертировать регистр Temp
    out    PORTB,Temp ;вывести данные в PORTB
    <AVR Instruction> ;восстановить регистр Temp
    rcall  DLY       ;вызвать подпрограмму задержки
    reti

```



; обработка прерываний при замыкании ключа SW1

L\_SW1:

```

    <AVR Instruction>      ;бегущий огонь влево
    brne   L1_SW1          ;если обнулится,
    <AVR Instruction>      ;переместить 1 в младший разряд

```

L1\_SW1:

;\*\*\*\* отобразить бегущий огонь (с учётом 0-горит,1-погашен)

```

    <AVR Instruction>      ;инвертировать регистр Temp
    out    PORTB,Temp      ;вывести данные в PORTB
    <AVR Instruction>      ;восстановить регистр Temp
    rcall  DLY             ;вызвать подпрограмму задержки
    reti

```

; подпрограммы

; задержка для защиты от дребезга контактов

DLY:

```

    ldi    Delay,255       ; задать задержку
    ldi    Delay2,255     ;

```

DLY2:

```

    subi   Delay,1        ; уменьшить на единицу
    sbci   Delay2,0;16-и битовый счётчик
                                ;задержки и если он не обнулится
    brcc   DLY2           ;повторять до заёма, а затем
    ret    ;завершить подпрограмму задержки

```

4. Изучить и описать последовательность действий, совершаемых микроконтроллером при возникновении внешних прерываний.
5. Описать формат, и действия, совершаемые инструкцией `iret`. Каково время исполнения команды?
6. Используя шаблон программы вставить необходимые инструкции микроконтроллера AVR [2].

7. Представить программу в виде функциональных модулей с кратким описанием функции каждого модуля в текстовой форме.
8. Опираясь на представление программы в виде функциональных модулей и текст программы составить граф-схему алгоритма.
9. Продумать последовательность и способ отладки каждого модуля программы. Привести план отладки.

Для самоконтроля письменно ответить на приведенные далее вопросы.

1. В каком режиме должны работать разряды D2 и D3 порта D?
2. С какой целью для этих разрядов подключаются внутренние подтягивающие резисторы?
3. Объяснить какие действия выполняют программные строки:  
ldi Temp, 0b00001010  
out MCUCR, Temp
4. Какие события включают внешние прерывания?
5. Перечислить биты и регистры, отвечающие за активность (чувствительность) подсистемы внешних прерываний к событиям на входе D2 порта D?
6. Какой глубины стек требуется для работы данной программы?

#### **4 Программа исследований и порядок работы**

1. Создать проект и ввести текст программы, созданной при самостоятельной подготовке к лабораторной работе.
2. Компилировать проект и протоколировать сообщения. Если имеются ошибки, привести ошибочные строки в отчёте, сделать анализ ошибок и ввести исправления.

3. Выполнить отладку модулей программы в соответствии с методикой разработанной при самостоятельной подготовке. В процессе отладки протоколировать содержимое модифицируемых регистров, сопоставляя с предсказанными значениями.
  4. В пошаговом режиме, имитируя нажатие и отпускание ключей протоколировать содержимое стека и указателя стека в моменты модификации. Объяснить значения слов, сохранённых в стеке.
  5. В пошаговом режиме имитировать одновременное нажатие двух ключей протоколировать поведение устройства и объяснить его.
- 
6. Подключить стенд с помощью программирующего кабеля к USB-порту компьютера и выполнить программирование микроконтроллера.
  7. По завершении программирования загруженная программа начнёт исполняться автоматически. Нажимая поочерёдно ключи, запротоколировать реакцию прототипного устройства.

Примечание: *Пункты 5, 6 исполняются факультативно.*

## **5 Контрольные вопросы**

1. Как обеспечивалось корректное функционирование программ в лабораторных работах 1-3, если они наложены на область векторов прерываний?
2. Как будет вести себя спроектированное устройство, если ключ удерживать в нажатом состоянии неограниченно долго?
3. Как решается вопрос обслуживания прерываний при одновременном возникновении двух и более прерываний в микроконтроллерах AVR?

4. Как изменится поведение устройства, если убрать программную задержку?

## **6 Содержание отчёта**

Отчёт должен содержать:

1. титульный лист;
2. наименование работы и цель исследований;
3. схему электрическую функциональную устройства и её краткое описание;
4. исходный текст программы;
5. описание модулей программы и их функций;
6. граф-схему алгоритма;
7. план отладки программы;
8. ответы на вопросы для самопроверки;
9. текст программы в соответствии с вариантом задания;
10. протокол отладки с анализом ошибок (при наличии);
11. протоколы исследования программы в соответствии с заданием и результаты анализа.
12. результаты экспериментальной проверки программы на прототипной системе;
13. ответы на контрольные вопросы.

## Литература

- 1 Голубцов, М.С. Микроконтроллеры AVR: от простого к сложному [Текст] /М. С. Голубцов. – М.: СОЛОН-Пресс, 2003. - 288 с.
- 2 Евстифеев, А.В. Микроконтроллеры AVR-семейств Tiny и Mega фирмы “АТМЕЛ” [Текст] / А.В. Евстифеев. -М.: Издательский дом “Додэка-XXI”, 2004. - 560 с.
- 3 0839I–AVR–06/02. [Электронный ресурс] : AT90S2313.pdf, Atmel Corporation, 2002.
- 4 AVR Instruction Set. [Электронный ресурс] : Doc0856.pdf, Atmel Corporation, 1999.

*Примечание: адаптированные переводы документов в электронной форме размещены на сервере кафедры К и Т ЭВС. Для поиска воспользуйтесь системой навигации компьютерной обучающей системы.*