МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«Юго-Западный государственный университет»
(ЮЗГУ)

Кафедра космического приборостроения и систем связи

УТВЕРЖДАЮ
Проректор по учебной работе
_____ О.Г. Локтионова
«11» 06 _____ 2024 г.

МИКРОПРОЦЕССОРНАЯ ТЕХНИКА

Методические указания к лабораторным занятиям

Курск 2024

УДК 621.316

Составитель О. Г. Бондарь

Рецензент

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

Микропроцессорная техника: Методические указания к лабораторным занятиям / Юго-Зап. гос. ун–т; сост.: О.Г. Бондарь. – Курск : ЮЗГУ, 2024. - 130 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.

Предназначены для обучающихся по направлению подготовки 11.03.03 «Конструирование и технология электронных средств» всех форм обучения.

Могут быть полезны обучающимся 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. 7,56. Уч.- изд. л. 6,84. Тираж 100 экз. Заказ 520.
Бесплатно.
Юго-Западный государственный университет.
305040, г. Курск, ул. 50 лет Октября, 94

Содержание

Планируемые результаты обучения

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

Обучающийся должен

знать:

- основы организации МП и МК, их подсистем и встроенной периферии;
- программные и аппаратные средства поддержки разработки программного обеспечения (ПО) систем на основе МК(МП);
- методы обработки данных в системах управления объектами и процессами и их влияние на требования к ресурсам МК (МП);
- особенности аппаратных и программных средств разработки, их состав и возможности.

уметь:

- оценивать параметры МП и МК и необходимые ресурсы для решения практических задач;
- использовать средства поддержки разработки ПО для одного из семейств МК или МП;
- выбирать эффективные методы обработки ориентированные на особенности архитектуры МК (МП);
- разрабатывать программное обеспечение в среде ASTUDIO.

владеть:

- методами оценивания параметров МП и МК и необходимых ресурсов для решения практических задач;

- навыками использования средств поддержки разработки ПО для одного из семейств МК или МП;
- навыками разработки алгоритмов решения задач профессиональной области;
- 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 ответственности 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 **A**lf-Egil Bogen и **V**egard Wollen (фамилий разработчиков) + **R**ISC. 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 объемом памяти программ от 1 до 128 кбайт.

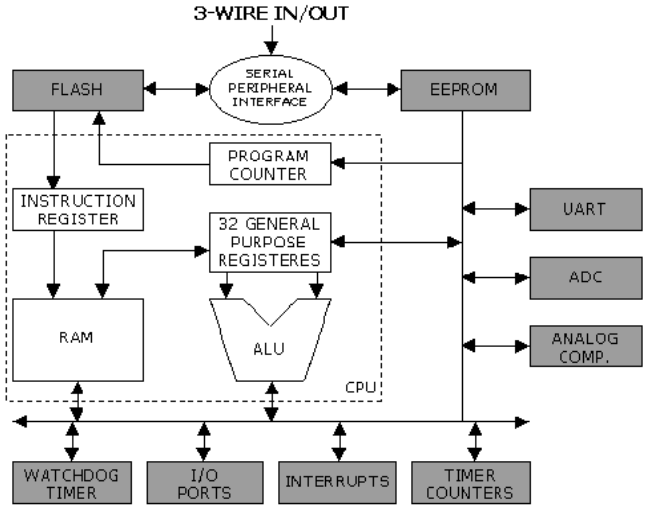


Рисунок 1- Структурная схема микроконтроллеров AVR.

- **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-

требление 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.

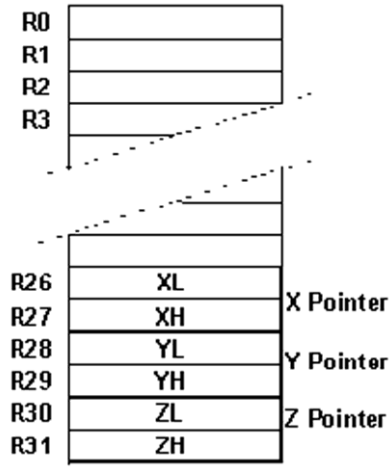


Рисунок 2- Регистровый файл AVR.

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 1K 16-разрядных слов. Флэш-память выдерживает не менее 1000 циклов перезаписи. Программный счетчик имеет ширину 10 бит и таким образом адресуется к 1024 словам программной флэш-памяти. Таблицы констант могут располагаться в диапазоне адресов 0-2K.

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

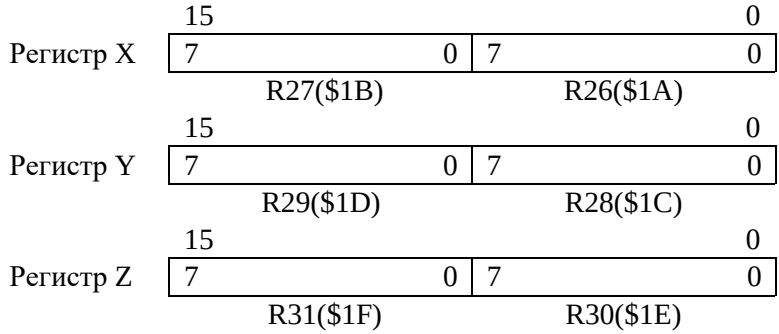
224 ячейки памяти включают в себя регистровый файл, память ввода/вывода и статическое ОЗУ данных. Первые 96 адресов используются для регистрового файла и памяти ввода/вывода, следующие 128 - для ОЗУ данных.

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:



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 постинкрементном. 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 1старший байт
$2A($4A)	ICR1L	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-

реннему 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.

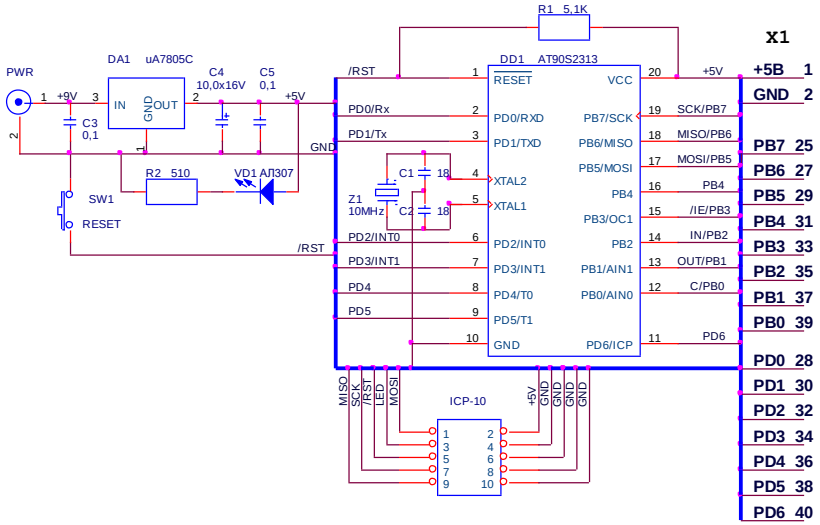


Рисунок 4 - Схема платы микроконтроллера

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

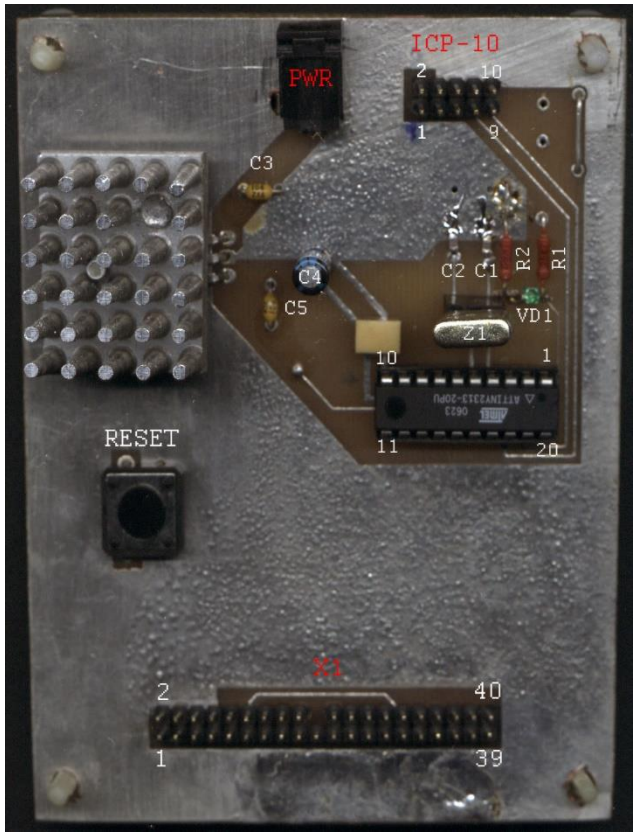


Рисунок 5 - Плата микроконтроллера.

Устройство статической индикации содержит 8 светодиодов и два ключа. Светодиоды подключены к порту **B** 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.

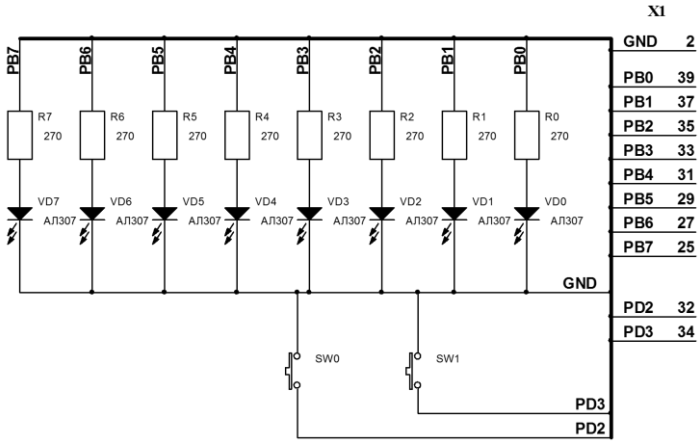


Рисунок 6 - Схема устройства статической индикации.

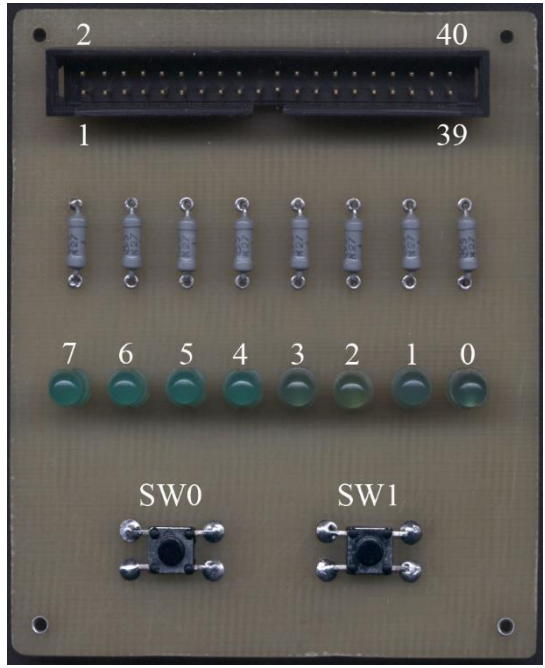


Рисунок 7 - Устройство статической индикации.

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.

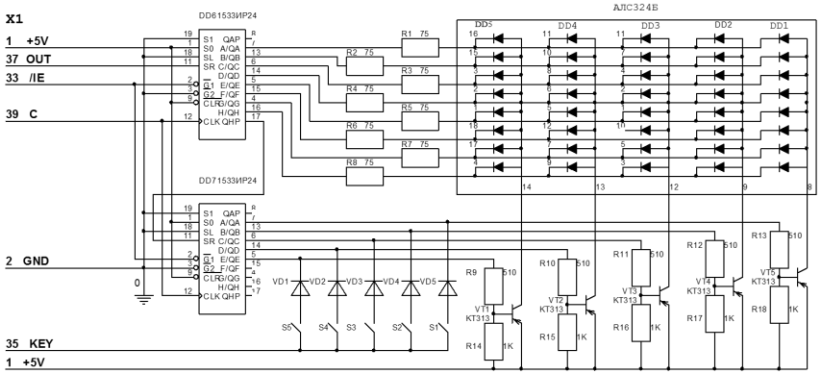


Рисунок 8-Схема устройства динамической индикации.

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).

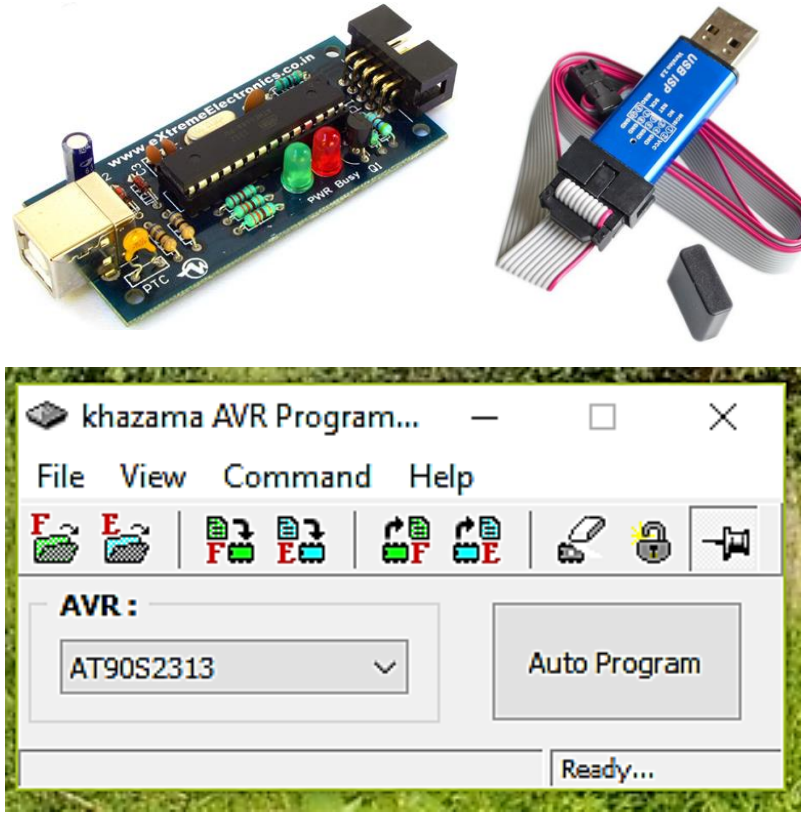


Рисунок 10 - Программатор.

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.

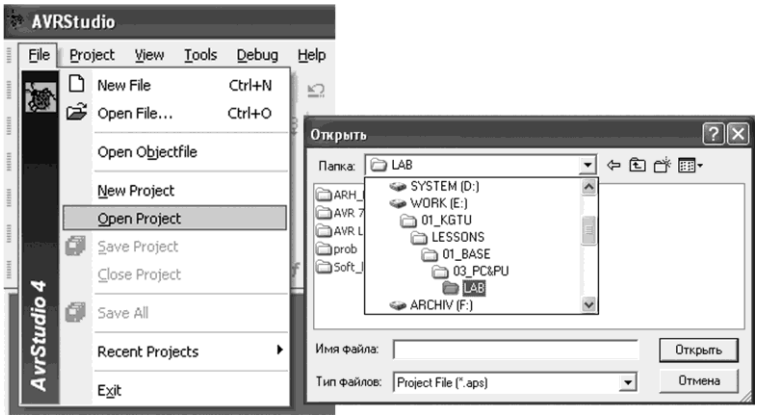


Рисунок 11- Последовательность открытия существующего проекта.

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**).

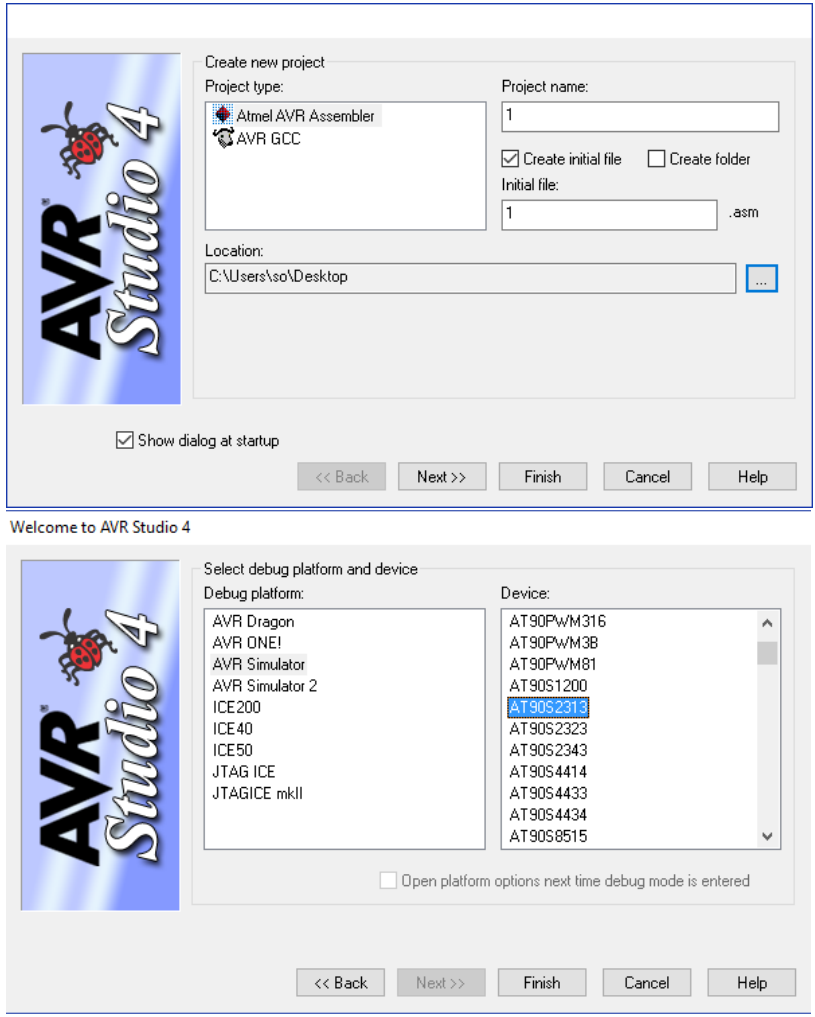


Рисунок 12 – Создание нового проекта

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

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 00000000₂=100₁₆=65536₁₀
; однократное повторение цикла занимает 5 тактов МК
; при этом общее время задержки примерно 327 680 тактов

DLY:
  subi  Delay_1,1     ;уменьшить на единицу
  sbci  Delay_2,0     ;24-х битовый счётчик
  sbci  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.

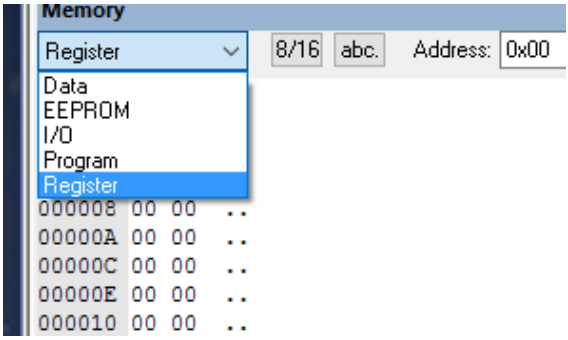


Рисунок 13 – Выбор отображаемых объектов в окне отладчика

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

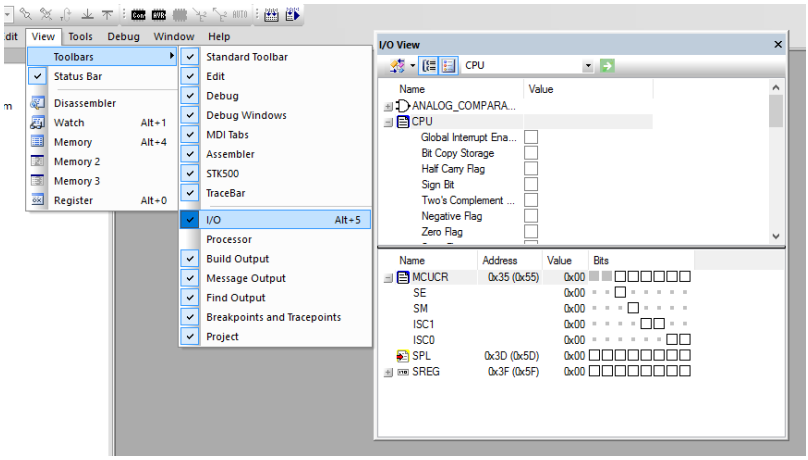


Рисунок14 – Включение отображения рабочей области.

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 стенда. 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 программировать 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

$$Tз=5 \times (30 \times N1 + 15 \times N2 + N3) \text{ мс},$$

где N1 – номер группы (1 или 2), N2-номер подгруппы (1 или 2), N3- 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

№ вар.	**Задание**
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
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 «PonyProg» и настроить программатор.
6. Загрузить HEX-файл откомпилированной программы, скопировать дамп 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 B в режим ввода, а другие – вывода?
2. Почему при инициализации управляющего регистра порта ввода 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. дамп памяти программ, копированный 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 **DDRx** - бит контроля направления передачи данных и привязки вывода к шине питания (V_{CC}), **PORTx** - бит привязки вывода к V_{CC} и бит выходных данных, **PINx** - 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 PINx. Если требуется обновить выходы - прочитайте PORTx защелку и потом модифицируйте данные.

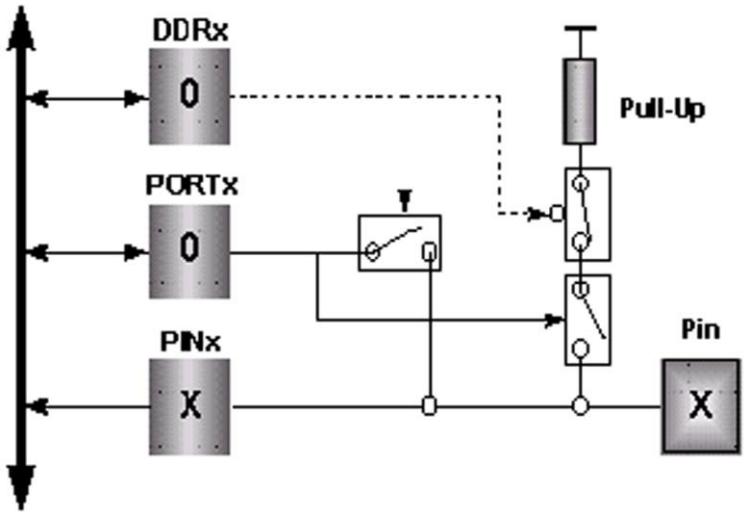


Рисунок 1 – Структура элемента ввода/вывода.

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 B при использовании для ввода/вывода равноправны. Бит DDBn регистра DDRB выбирает направление передачи данных. Если бит установлен (1), вывод сконфигурирован как выход. Если бит сброшен (0) - вывод сконфигурирован как вход. Если PORTBn установлен и вывод сконфигурирован как вход, включается КМОП подтягивающий резистор. Для отключения резистора, PORTBn должен быть сброшен (0) или вывод должен быть сконфигурирован как выход (таблица 4).

Таблица 4. Влияние DDBn на выводы порта B

DDRBn	PORTBn	Вх/Вых	Подт. резист.	Комментарий
0	0	Вход	Нет	Третье состояние (Hi-Z)
0	1	Вход	Да	PBn источник тока I_{IL}, 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 B

;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 B. 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    ;инициировать 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 измениться 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 на вопросы для самопроверки;
9. текст программы в соответствии с вариантом задания;
10. протокол отладки с анализом ошибок (при наличии);
11. листинг программы;
12. дамп 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-регистровой 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 стековых 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
        sbci    Delay2,0        ;24-х битовый счётчик
        sbci    Delay3,0        ;задержки и если он не обнулился
        brcc    DLY             ;повторять до обнуления, а затем
        ret                     ;возврат из подпрограммы
```

5. 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×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 протоколировать 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 протоколировать содержимое модифицируемых регистров, сопоставляя с предсказанными значениями.
6. После сброса в пошаговом режиме наблюдать и протоколировать 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. Протоколировать 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 ключи, запротоколировать 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. дамп 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 OVF1	Переполнение таймера/счетчика 1
7	$006	TIMER0, OVF0	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_OVF1       ; Обработка переполнен. таймера 1
$006    rjmp TIM_OVF0       ; 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)	**INT**	**INT0**	GIM
Чт./зап. (R/W)	**1**		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 отрабатывается 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 отрабатывается 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 (DDRD), 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)	**INTF**	**INTF**	GIFR
Чт./зап. (R/W)	**1**	**0**	
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)	**ISC1**	**ISC1**	**ISC0**	**ISC0**	
Чт./зап. (R/W)	**1**	**0**	**1**	**0**	MCUC
Начальн.знач.					R
	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 **B** 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 обнулился
        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 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.

5. ПРИМЕНЕНИЕ ТАЙМЕРА ДЛЯ ВРЕМЕННОЙ СИНХРОНИЗАЦИИ ПРОГРАММНЫХ ПРОЦЕССОВ

1 Цель работы

Изучение устройства аппаратных таймеров микроконтроллеров AVR и особенностей их применения.

2. Аппаратные таймеры микроконтроллеров AVR

2.1 Организация таймеров

Во многих областях применения средств вычислительной техники требуется привязка тех или иных действий к определённым моментам времени. При невысоких требованиях к точности это можно осуществить программно, что потребует дополнительной памяти и приведёт к снижению производительности системы. Аппаратные средства поддержки функций связанных со счётом событий, измерением временных интервалов между ними, выдержкой точных временных интервалов существуют уже давно в виде специализированных БИС (например, широко применяемая ранее в персональных компьютерах БИС таймера i8254). В настоящее время практически все микроконтроллеры имеют для этих целей встроенные таймеры – счётчики или системы на их основе.

В простейшем случае такое устройство представляет собой суммирующий или вычитающий счётчик с предварительной установкой. Тактирующие импульсы могут быть поданы на него с внешнего источника или от тактового генератора процессора. Импульс переполнения (заёма в случае вычитающего счётчика) устанавливает специальный триггер (флаг переполнения). Это событие может обнаруживаться программно, но существенно эффективней использование механизма прерываний. Частота тактирующих импульсов, подаваемых на вход таймера счётчика, может быть снижена установкой предварительного программно управляемого делителя частоты. Это существенно расширяет

диапазон временных интервалов, с которыми может работать система.

В микроконтроллере серии AVR AT90S2313 предусмотрены два таймера/счетчика общего назначения, 8-разрядный и 16-разрядный [1]. Каждый из таймеров индивидуально подключается к одному из выходов 10-разрядного предварительного делителя частоты [рисунок 1.]. Оба таймера могут использоваться как таймеры с внутренним источником импульсов или счетчики импульсов, поступающих извне. Источниками импульсов для таймеров можно выбрать сигнал с тактовой частотой процессора (CK), импульсы предварительного делителя (CK/8, CK/64, CK/256 или CK/1024) или импульсы с соответствующего внешнего вывода. Кроме того, таймеры могут быть остановлены, запретом прохождения импульсов на них.

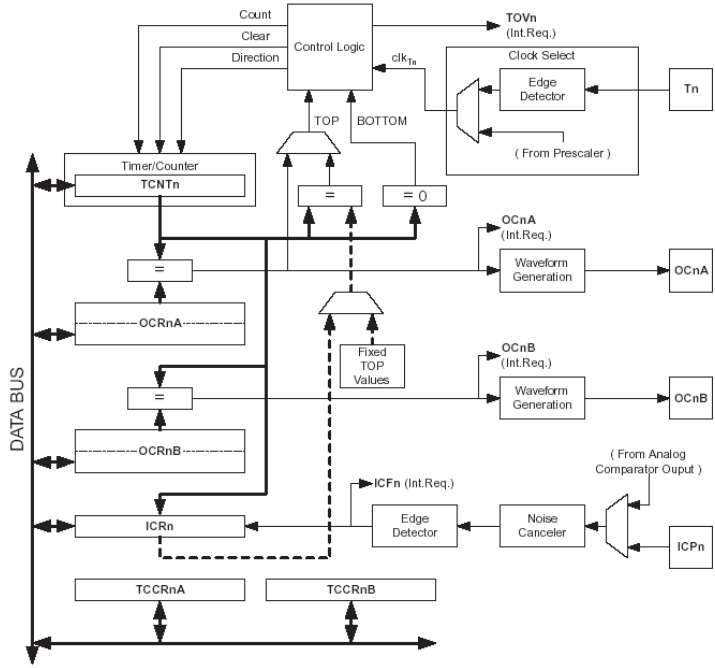


Рисунок 1 – Структурная схема таймеров микроконтроллера AT90S2313.

2.2 Управление таймерами

16-разрядный таймер/счетчик 1 (**TC1**) можно использовать как счетчик с высоким разрешением. TC1 поддерживает функцию совпадения, используя регистр совпадения OCR1A в качестве источника для сравнения с содержимым счетчика. Функция совпадения поддерживает очистку счетчика и переключение выхода по совпадению.

TC1 можно использовать как 8-, 9- или 10-разрядный широтно-импульсный модулятор (ШИМ).

Функция захвата по входу предусматривает захват содержимого таймера/счетчика 1 в регистр захвата ICR1 и управляется внешним сигналом на входе захвата - ICP.

Работа TC1 определяется управляющим регистром **TCCR1,** состоящим из 8-и битных регистров A и B (табл. 7).

Таблица 7. Регистр управления TC1 – TCCR1A

Бит	7	6	1	0
$2F ($4F)	**COM1A1**	**COM1A0**	**PWM11**	**PWM10**
Чт./зап. (R/W)	R/W	R/W	R/W	R/W
Нач.знач.	0	0	0	0

Биты 7,6 - COM1A1, COM1A0: Режим выхода совпадения, биты 1 и 0: Эти управляющие биты задают отклик вывода OC1 процессора на совпадение регистра сравнения и таймера/счетчика 1. Поскольку это альтернативная функция порта, соответствующий бит направления должен устанавливать вывод на выход.

Биты 5..2 - в AT90S2313 эти биты зарезервированы и всегда читаются как 0.

Биты 1,0 - PWM11, PWM10: Биты установки ШИМ: Эти биты устанавливают режим работы таймера/счетчика 1 в качестве ШИМ.

Управляющие комбинации бит показаны в таблице 8:

Таблица 8. Установка режима совпадения.

COM1A1	COM1A0	Описание
0	0	Таймер/счетчик 1 отключен от вывода
0	1	Переключение выхода OC1
1	0	Сброс (0) вывода OC1
1	1	Установка (1) вывода OC1

В режиме ШИМ эти биты имеют другие функции. При изменении битов COM1A1 и COM1A0 прерывание по совпадению должно быть запрещено, очисткой соответствующего бита в регистре TIMSK. Иначе, прерывание может произойти во время изменения битов.

Из всех комбинаций битов, управляющих установкой ШИМ, приведенных в таблице 3, в данной работе нас будет интересовать лишь комбинации запрета работы ШИМ – 00.

Таблица 3. Установка режима работы ШИМ.

PWM11	PWM10	*Описание*
0	0	Работа ШИМ запрещена
0	1	8-разрядный ШИМ
1	0	9-разрядный ШИМ
1	1	10-разрядный ШИМ

В таблице 4 приведено назначение управляющих битов регистра TCCR1B.

Таблица 4. Регистр управления TC1 – TCCR1B

Бит	7	6	3	2	1	0
$2E ($4E)	**ICNC1**	**ICES1**	**CTC1**	**CS12**	**CS11**	**CS10**
Чт./зап.	R/W	R/W	R/W	R/W	R/W	R/W
(R/W)	0	0	0	0	0	0
Нач.знач.						

Бит 7 - ICNC1: Подавитель входного шума входа захвата: Если бит сброшен (0), подавление входного шума входа запрещено. При этом захват срабатывает по первому заданному (нарастающему или спадающему) фронту сигнала на выводе ICP. При установке бита обрабатываются четыре последовательные выборки сигнала на выводе ICP. Для срабатывания захвата все выборки должны соответствовать уровню, заданному битом ICES1. Частота выборок равна тактовой частоте процессора.

Бит 6 - ICES1: выбор фронта сигнала захвата: Если бит ICES1 сброшен (0) содержимое таймера/счетчика 1 переписывается в регистр захвата по спадающему фронту сигнала на выводе ICP. Если бит установлен - по нарастающему фронту сигнала.

Биты 5,4 - в AT90S2313 эти биты зарезервированы и читаются как 0.

Бит 3 - CTC1: Очистка таймера счетчика 1 по совпадению: Если бит установлен (1), таймер/счетчик 1 устанавливается в $0000 в такте, следующем за событием совпадения. Если бит сброшен, таймер/счетчик 1 продолжает считать пока не будет остановлен, сброшен, произойдет его переполнение или изменение направления счета. В режиме ШИМ этот бит не работает.

Биты 2,1,0 - CS12, CS11, CS10: выбор тактирования: Эти биты определяют источник счетных импульсов для таймера/счетчика 1 (Таблица 5).

Таблица 5. Выбор источника счетных импульсов.

CS12	**CS11**	**CS10**	**Описание**
0	0	0	Таймер/счетчик остановлен
0	0	1	CK
0	1	0	CK/8
0	1	1	CK/64
1	0	0	CK/256

1	0	1	CK/1024
1	1	0	Внешний вывод T1, нарастающий фронт
1	1	1	Внешний вывод T1, спадающий фронт

Собственно, таймер представлен двумя регистрами TCNT1H и TCNT1L. Это 16-разрядный регистр, содержащий текущее значение таймера/счетчика 1. Чтобы чтение и запись двух байт счетчика происходило синхронно, для работы с ним используется временный регистр (TEMP).

– **Запись** в таймер счетчик 1: При записи старшего байта в TCNT1H, записываемые данные помещаются в регистр TEMP. Затем, при записи младшего байта, он вместе с данными из TEMP переписывается в таймер/счетчик 1. Таким образом, при записи 16-разрядного значения первым должен записываться байт в **TCNT1H**.
– **Чтение** таймера/счетчика 1: При чтении младшего байта из TCNT1L, он посылается в процессор, а данные из TCNT1H переписываются в регистр TEMP, то есть одновременно читаются все 16-разрядов. При последующем чтении регистра TCNT1H, данные берутся из регистра TEMP.

Таймер/счетчик 1 организован как суммирующий счетчик (в режиме ШИМ - суммирующий/вычитающий) с возможностью чтения и записи. Если выбран источник тактовых импульсов для таймера/счетчика 1, после записи в него нового значения, он продолжает счет в следующем после записи периоде тактовой частоты.

Регистр совпадения **OCR1AH** и **OCR1AL** - 16-разрядный регистр, доступный для чтения и записи.

В этом регистре хранятся данные, которые непрерывно сравниваются с текущим значением таймера/счетчика 1. Действие по совпадению задается регистрами управления таймером/счетчиком 1 и регистром состояния.

Поскольку регистр OCR1A является 16-разрядным, при записи нового значения в регистр, для того чтобы оба байта регистра записывались одновременно, используется временный регистр. При записи старшего байта, данные помещаются во временный регистр, который переписывается в OCR1AH при записи младшего байта в OCR1AL. Таким образом, для записи в регистр первым должен записываться **старший байт**.

Регистр захвата ICR1H и ICR1L 16-разрядный регистр, доступный только для чтения.

По нарастающему или спадающему фронту сигнала (в соответствии с выбором фронта импульса захвата ICES1) на выводе ICP текущее значение таймера/счетчика 1 переписывается в регистр захвата ICR1. В это же время устанавливается флаг захвата ICF1.

Поскольку регистр захвата является 16-разрядным, для одновременного считывания байтов, используется временный регистр. При чтении младшего байта ICR1L, он посылается в ЦПУ, а старший байт регистра ICR1H переписывается во временный регистр. При чтении старшего байта, он принимается из временного регистра. Таким образом, для чтения 16-разрядного регистра первым должен читаться младший байт.

2.3 Применение таймеров-счётчиков

Таймеры-счётчики могут быть использованы для задания задержек при формировании сигналов, периода повторения процедур, интервалов накопления данных, темпа ввода или вывода данных, измерения интервалов между двумя событиями, для цифро-аналогового преобразования (ШИМ). При этом микроконтроллер может выполнять вычислительную работу не расходуя ресурсы на формирование временных интервалов. Формирование временных интервалов в микроконтроллерах AVR может осуществляться методом обнаружения факта переполнения счётчика таймера или методом сравнения содержимого счётчика-таймера с регистром совпадения.

В первом случае код таймера возрастает по каждому тактовому импульсу от некоторого предварительно записанного значения до переполнения. Переполнение чаще всего используют для генерирования прерывания. Меняя начальное значение можно варьировать временной интервал. Для формирования периодических событий в этом случае требуется программная перезагрузка счётчика. Т.к. прерывания обрабатываются лишь по завершению текущий команды, длительность которой может быть один или 2 такта, то возникает эффект накопления ошибки.

Метод сравнения свободен от этого недостатка благодаря возможности применения функции автозагрузки счётчика-таймера. Достоинство метода состоит и в том, что временной интервал пропорционален загруженному коду.

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 **B** 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, INT1, а также внутреннего прерывания по совпадению таймера 1.
3. Изучить порядок программирования таймера-счётчика 1 в режиме совпадения с автозагрузкой.
4. Изучить шаблон программы, выполняющей следующие функции:
   – при однократном нажатии на ключ “SW0” световое пятно на светодиодном индикаторе (циклически) перемещается вправо каждые 25+15×g+7×p+0,5×n [мс],

g–номер группы, где p-номер подгруппы, n- номер рабочего места;

- при однократном нажатии на ключ “SW1” световое пятно на светодиодном индикаторе (циклически) перемещается влево каждые 25+15×g+7×p+0,5×n [мс];
- если не нажат ни один из ключей, не изменять изображение.

```
; Применение таймера при формировании временных интервалов
.include "2313def.inc"
;f=8 MHz

.def    Temp    =r16            ;временный регистр
.def    Test    =r17            ;регистр состояния клавиатуры

; Const definitions
.equ    Kdt1    = 50000         ;модуль счёта таймера-счётчика 1

.cseg
.org $00
        rjmp start      ;на начало программы
        rjmp L_SW0      ;обработчик прерываний от ключа SW0   rjmp
        L_SW1           ;обработчик прерываний от ключа SW1
.org $04
        rjmp L_CMP1     ;обраб. Прерыв. по совпадению таймера 1

; 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

; Предделитель Кдп=8, таймер Кдт=1250. Общий Kd=8x20000.
; Для F=8МГц интервал прерываний по совпадению - 20 мс.

        ldi Temp,(1<<OCIE1A)    ;разрешение прерываний
        out TIMSK,tmp           ;совпадения таймера 1

        clr Temp                ; запрет ШИМ и модификации
        out TCCR1A,tmp          ; выходного сигнала по совпадению

        ldi Temp,(1<<CTC1)+(1<<CS11)    ; очистка таймера по
        out TCCR1B, Temp                ; совпадению и преддел. = 8

        ldi     Temp,high(Kdt1) ; загрузить
        out     OCR1AH, Temp    ; старший байт
        ldi     Temp,low(Kdt1)  ; и младший байт коэффициента
        out     OCR1AL, Temp    ; пересчёта таймера 1

        ldi     Temp,1          ;активизировать один бит
                                ;в регистре бегущих огней
        clr     Test            ;сбросить состояние клавиатуры
        sei     ;разрешить прерывания
loop:
        rjmp    loop            ;

;**** обработка прерываний при замыкании ключа SW0
L_SW0:
        ldi     Test,1          ;нажат ключ SW0
        reti

;**** обработка прерываний при замыкании ключа SW1
L_SW1:
        ldi     Test,2          ; нажат ключ SW1
        reti

; бегущий огонь
```

```
   L_CMP1:
        sbrc    Test,1          ;не SW0, проверять далее
        rjmp    L0_CMP1         ;сдвигать вправо по ключу SW0
        sbrc    Test,2          ;не SW1, ждать нажатия
        rjmp    L1_CMP1         ;сдвигать влево по ключу SW1
        reti                    ;ожидать нажатия любого ключа

; было нажатие ключа SW0
L0_CMP1:
        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
        reti                    ;ждать очередные 50 мс

; было нажатие ключа SW1
L1_CMP1:
        <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
        reti                    ;ждать очередные 50 мс
```

5. Изучить и описать последовательность действий совершаемых микроконтроллером при возникновении прерывания по совпадению таймера 1.
6. Рассчитать и установить величины коэффициента деления предварительного делителя и модуля счёта таймера-счётчика Kdt1 в соответствии индивидуальным заданием временного интервала.

Для самоконтроля письменно ответить на приведенные далее вопросы.

1. Объяснить назначение и работу директивы ассемблера *.org $04*.
2. Изучить конструкцию ldi *Temp,(1<<CTC1)+(1<<CS11)*. Объяснить её смысл. Записать константу эквивалентную данному выражению.
3. Изучить конструкцию *ldi Temp,high(Kdt1)*. Записать константу эквивалентную данному выражению.
4. Чем обусловлен порядок загрузки регистров OCR1AH и OCR1AL?
5. Изучить и описать работу инструкции *lsr Temp* [2]. В чём причина усложнения модуля формирования бегущего огня? Почему нельзя ограничиться лишь одной вышеприведенной инструкцией?
6. Как включить автозагрузку таймера-счётчика 1 по совпадению?

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 соответствии с методикой разработанной при самостоятельной подготовке. В процессе отладки протоколировать содержимое модифицируемых регистров, сопоставляя с предсказанными значениями.
4. Исследовать и объяснить поведение микроконтроллера при удержании ключа клавиатуры в разомкнутом состоянии, замкнутом состоянии, после изменения состояния (0-1, 1-0).
5. Исследовать и объяснить поведение микроконтроллера при одновременных запросах на прерывание.

6. Подключить стенд с помощью программирующего кабеля к параллельному порту компьютера и выполнить программирование микроконтроллера.
7. По завершении программирования загруженная программа начнёт исполняться автоматически. Нажимая поочерёдно ключи, запротоколировать реакцию прототипного устройства.

5 Методические указания

При отладке модулей обработчиков прерываний от клавиатуры в пошаговом режиме необходимо принудительно устанавливать входы PIND D2 или D3 в единичный, а после исполнения очередной раз команды *rjmp loop* - в нуль.

В пунктах 4, 5 программы исследований повторное изменение состояния ключей производить обязательно после исполнения хотя бы одной команды.

Для удобства отладки прерываний по совпадению установить минимальный интервал прерываний от таймера немногим больший времени исполнения его обработчика прерываний.

6 Контрольные вопросы

1. Перечислить сигналы, которыми может тактироваться таймер-счётчик.
2. Какой максимальный временной интервал может быть сформирован таймером-счётчиком 1?
3. Чем определяется минимальный квант времени, отсчитываемый таймером-счётчиком 1?
4. Перечислить факторы, определяющие временной интервал от совпадения до момента выполнения первой команды обработчиком прерывания совпадений.
5. Обработчик прерываний по совпадению формирует коды, периодически выдаваемые в порт микроконтроллера. Что необходимо предпринять для минимизации вариаций периода.

6. Предложите вариант аппаратно-программного решения, допускающий прямое измерение периодичности прерываний по совпадению с помощью частотомера.

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. текст программы в соответствии с индивидуальным заданием;
5. описание функций программы и модулей;
6. граф-схему алгоритма;
7. план отладки программы;
8. ответы на вопросы для самопроверки;
9. протокол отладки с анализом ошибок (при наличии) и результатами оценки периодичности прерываний по совпадению таймера-счётчика 1;
10. результаты экспериментальной проверки программы на прототипной системе;
11. ответы на контрольные вопросы.

6. ОРГАНИЗАЦИЯ ПРОГРАММНОГО ПОСЛЕДОВАТЕЛЬНОГО ВВОДА/ВЫВОДА

1 Цель работы

Изучение интерфейса SPI и особенностей его программной реализации.

2 Особенности интерфейса SPI

2.1 Последовательные интерфейсы. Достоинства и недостатки

В современной вычислительной технике используется большое количество последовательных интерфейсов. Вот некоторые из них: RS232, RS422, RS485, One-Wire, I2C, USB, SPI, SATA, CAN. Применение последовательных интерфейсов при передаче данных на большие расстояния обусловлено существенным удешевлением линий связи. При малых расстояниях преимущества не столь очевидны и сводятся к следующему:

- упрощение топологии межсоединений интегральных компонент;
- сокращение количества выводов интегральных схем (ИС) необходимого для организации интерфейсов и, следовательно, стоимости ИС;
- упрощение и удешевление разъёмных соединений;
- упрощение защиты от электромагнитных помех;
- возможность существенного повышения скорости обмена, благодаря устранению такого серьёзного фактора, как взаимные помехи интерфейсных линий.

Главный недостаток последовательных интерфейсов – необходимость преобразования форматов, данных и сложные протоколы обмена, требующие применения дополнительных ап-

паратных и программных средств. Встроенные средства поддержки таких интерфейсов как CAN, USB существенно повышают стоимость микроконтроллеров. Периферийные устройства, подключаемые по последовательным интерфейсам, усложняются, и растёт их стоимость. Поэтому выбор оптимального последовательного интерфейса относится к нетривиальным задачам.

2.1 Интерфейс SPI

На фоне других последовательных интерфейсов SPI имеет неоспоримое преимущество, связанное с простотой его реализации. SPI (Serial Peripheral Interface) - это полнодуплексный скоростной синхронный трёхпроводный интерфейс. В настоящее время его поддерживают не только AVR-контроллеры, но и некоторые модели **Microchip**, **Motorola**, **Zilog**. Интерфейс стал фактически промышленным стандартом.

Особенности интерфейса:

- в слабой степени регламентируется лишь физический уровень, канальный уровень не регламентирован вообще;
- отсутствие ограничений на скорость обмена (лимитируется аппаратными средствами или программной поддержкой);
- отсутствие ограничений на формат передаваемых данных;
- свобода в выборе способа синхронизации (полярность и фронт);
- для канального уровня оговорена дисциплина доступа по схеме ведущий-ведомый (в некоторых микроконтроллерах допустим режим нескольких ведущих).

Низкий уровень стандартизации придаёт интерфейсу гибкость, но одновременно порождает проблемы совместимости оборудования разных производителей. Это и определяет область применения интерфейса преимущественно как внутрисистемного интерфейса точка-точка и реже шинного.

SPI в реализации **Atmel** имеет четыре задаваемые программно скорости передачи, может передавать байты от старшего к младшему биту и наоборот, обнаруживает ошибки пакета. К основным линиям интерфейса относятся:

– **MISO** - Master In, Slave Out - выход ведомого, вход ведущего;
– **MOSI** - Master Out, Slave In - вход ведомого, выход ведущего;
– **SCK** - SPI ClocK - тактовая линия.

Микроконтроллер допускает работу как в режиме ведущего, так и ведомого. В режиме ведущего он формирует тактовые импульсы. Схема соединений двух устройств по интерфейсу SPI представлена на рисунке 1.

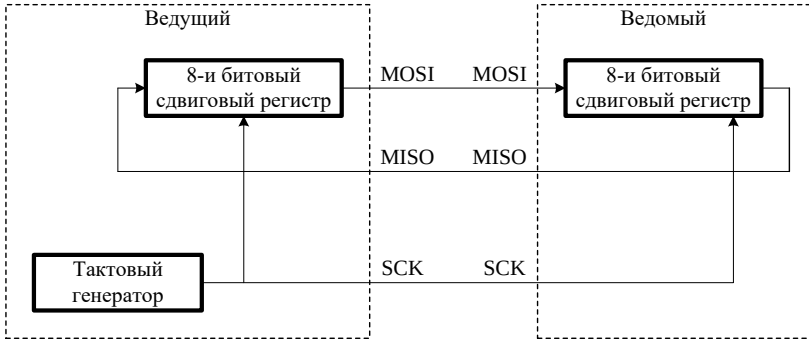


Рисунок 1 – Соединение двух устройств по интерфейсу SPI.

Таким образом, для организации интерфейса необходим сдвиговый регистр и тактовый генератор. Сдвиговый регистр предназначен для преобразования данных из параллельной формы представления в последовательную и обратно, а тактовый генератор определяет скорость передачи через темп сдвига данных. Регистры могут иметь разрядность и отличную от 8-битной. На рисунке 2. приведен **вариант** временных диаграмм

интерфейсных сигналов с фиксацией данных по фронту синхроимпульса.

Функции преобразования формы представления данных и генерирования тактовых импульсов легко реализуемы программно. Аппаратная поддержка интерфейсов повышает скорость обмена данными и разгружает процессор от дополнительной работы.

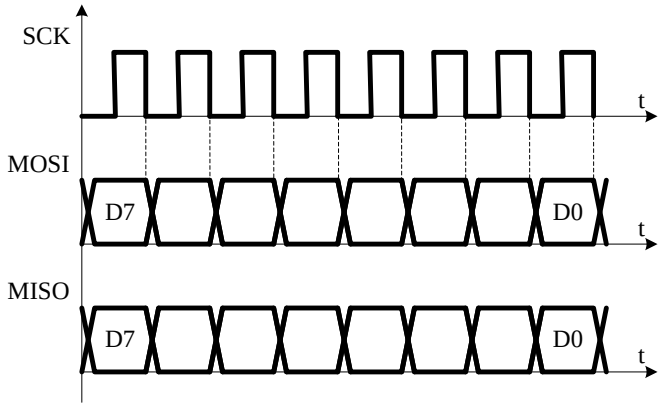


Рисунок 2 – Временные диаграммы интерфейсных сигналов.

2.2 Контроллер клавиатуры и дисплея с последовательным интерфейсом

Применение интерфейса SPI позволяет существенно сократить количество линий для обслуживания символьных дисплеев и клавиатур при несущественных дополнительных аппаратных затратах. Иллюстрацией этого утверждения может служить схема используемого в лабораторном макете модуля клавиатуры и дисплея (рисунок 3).

ИС DD2-DD3 1533ИР24 – образуют 16-битовый сдвиговый регистр. При этом DD2 (разряды D0-D7) управляет сегментами

индикатора, а DD3 – коммутирует разряды индикатора через ключи на биполярных транзисторах VT1-VT5 (разряды D8-D12). Дисплей реализует принцип динамической индикации. В каждый фиксированный момент времени активен лишь один разряд индикатора. Разряды D8-D12 активизируются поочерёдно. D12 – управляет старшим, D8 – младшим разрядом индикатора. Выходные линии регистра DD3 управляют и линейками клавиатуры. В данном случае каждая линейка содержит только один ключ (S1-S5). Таким образом, в устройстве реализуется часто используемый принцип совмещения функций выводов. Сигнальная линия OUT эквивалентна линии MOSI интерфейса SPI. Линия C - SCK.

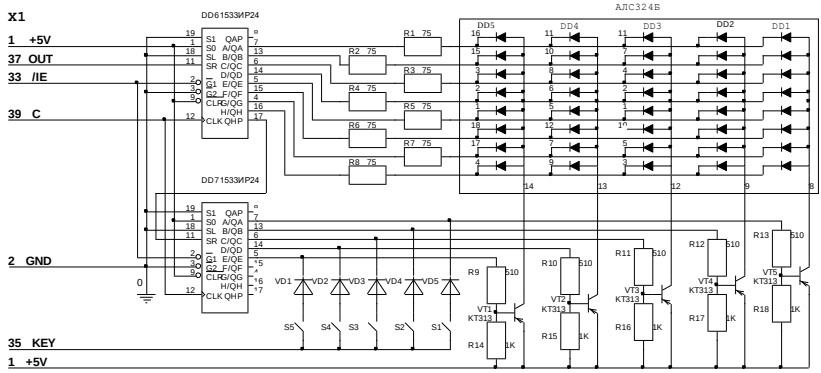


Рисунок 3 – Клавиатура и устройство индикации

Следовательно, устройство вывода – символьный семисегментный дисплей может интерпретироваться как 13-битный сдвиговый регистр (старшие 3 разряда не используются). Устройство ввода – клавиатура представляет собой матричное поле 5×1 активизируемое старшими 5-ю битами 13 – битного слова. Выходные данные клавиатуры представляют собой однобитное слово – состояние клавиатуры (ключ замкнут/разомкнут). Признаком активности линейки и, соответственно, замыкания ключа является низкий уровень сигнала на линии IN. При разомкнутом

ключе или пассивности линейки на линии IN должна генерироваться логическая единица. Для этого на линию необходимо подать высокий уровень напряжения с помощью резистора. В качестве этого резистора целесообразно использовать встроенный подтягивающий резистор порта. Идентификатором замкнутого ключа является 5-и битный унитарный код в разрядах D8-D12 сдвигового регистра. Необязательный сигнал /IE высоким уровнем гасит индикаторы и исключает эффект паразитной засветки сегментов при загрузке нового кода. Интерфейсные сигналы формируются программно на выходах порта B микроконтроллера:

PB0 – C, сигнал синхронизации;
PB1 – OUT, последовательные выходные данные;
PB2 – IN, вход – состояние ключей клавиатуры.

3 Подготовка к лабораторной работе

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

1. Изучить приведенную схему устройства индикации и клавиатуры.
2. Изучить временные диаграммы сигналов интерфейса SPI.
3. Изучить шаблон программы, выполняющей следующие функции:
   – Если ключ “S” нажат, отображать в разряде]N1/2[светодиодного индикатора цифру N1;
   – Если ключ “S” отпущен, отображать цифру N0.

Отображаемая цифра N1 – порядковый номер студента в списке группы, N0 = N1-1.
]N1/2[- целая часть результата деления на 2 порядкового номера студента в списке группы
При N1>9 уменьшить его на 9.
При выполнении работы учитывать, что номер активного ключа S совпадает с номером активного разряда.

```
   ;***** Последовательный вывод данных
.include "2313def.inc"

.def tmp = r16
.def Symbol = r17          ; Отображаемый символ
.def Send_Byte = r18       ; Передаваемый байт
.def Count = r19           ; Счётчик сдвигов
.def dl = r20              ; младший
.def dh = r21              ; старший байт задержки
;====== Bit definitions ========================================
.equ    I_S = PB0          ;вывод синхронизации
.equ    I_D = PB1          ;вывод последовательных данных
.equ    K_I = PB2          ;вход - статус клавиатуры
.equ    I_E = PB3          ;вывод разрешения дисплея

; ===== Const definitions ====================================

.equ    N0 = 0b10011001          ;семисегм. отображаемый символ N0
.equ    N1 = 0b10010010          ;семисегм. отображаемый символ N1
.equ    Position = 0b11111101    ;код активируемого разряда индик.

;====== Инициализация ====================================

RESET:
;Настроить порт B в соответствии с функциями в Bit definitions

        ldi tmp,b11111011
        out DDRB, tmp    ;PORTB,PB2-ввод, остальные-вывод


        sbi PORTB, K_I   ;Подтягивающий резистор на входе PB2
        cbi PORTB, I_S   ;сбросить вывод синхронизации I_S

        ldi Symbol,N0    ;предварительная загрузка символа N0
        cbi PORTB, I_E   ;включить индикатор (LOW)

        ldi tmp,low(RAMEND)      ;установить вершину
        out SPL,tmp              ;стека
```

```
;==подготовка передачи по интерфейсу SPI позиции индикации======
loop:
        cbi PORTB, I_E              ;включить индикатор (LOW)

;===== передача позиции символа ==============================
Send_Position:
        ldi Send_Byte,Position      ;номер индицируемого разряда
        rcall Send

;===== Передача отображаемого символа ========================
Send_Symbol:
        mov Send_Byte,Symbol    ;индицируемый символ
        rcall Send

;== Анализ состояния ключа и загрузка отображаемого символа ====
Key:
        ldi Symbol,N0  ;предварительная загрузка символа=N0

        in tmp,pinB         ; копировать состояние ключа S и при 1
        sbrs tmp,PB2        ; пропустить загрузку символа N1
        ldi Symbol,N1       ;загрузка передаваемого символа=N1

;=============================================================
; задержка для уверенного наблюдения  отображаемого символа
; задержка основана на декременте 16-битной переменной до
; возникновения заёма. Здесь - 65536 раз.
; младший байт переменной размещается в регистре ind_c
; старший байт переменной размещается в регистре ind_s

        ser dl     ;загрузить в пару регистров
        ser dh     ;максимальное число - задержку
dly:
        subi dl,1 ;декрементировать два
        sbci dh,0           ;байта задержки (вычесть заём)
        brcc dly  ;продолжать до возникновения заёма

        rjmp loop
```

```
    ;===== передача ========================================
Send:
        clr Count                   ;очистить счётчик сдвигов
SendB:
        sbrc Count,7        ;если очередной сдвинутый бит 1
        sbi PORTB,I_D       ;передаваемый бит = 1

        sbrs Send_Byte,7;если очередной бит 0
        cbi PORTB,I_D       ;сбросить передаваемый бит

        sbi PORTB,I_S       ;Синхроимпульс HIGH
        cbi PORTB,I_S       ;Синхроимпульс LOW

        cpi Count,7                 ;8 разрядов вытолкнуты - управление
        breq le_Send        ;разрядами завершено

        inc Count
        rol Send_Byte       ;следующий бит
        rjmp SendB
le_Send:
        ret
```

4. Используя шаблон программы вставить необходимые инструкции микроконтроллера AVR (описание инструкций приведено в [2]).
5. Представить программу в виде функциональных модулей с кратким описанием функции каждого модуля в текстовой форме.
6. Опираясь на представление программы в виде функциональных модулей и текст программы составить граф-схему алгоритма.
7. Продумать последовательность и способ отладки каждого модуля программы. Привести план отладки.
8. Изучить 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 порта B?
2. С какой целью для некоторых разрядов подключаются внутренние подтягивающие резисторы?
3. Чем определяется номер ключа, контролируемого программой?
4. Какие уровни соответствуют каждому из состояний?
5. Каким логическим уровнем сдвигового регистра определяется индицируемый разряд?
6. Каким логическим уровнем сдвигового регистра засвечивается сегмент активного разряда?
7. Как обозначаются сегменты семисегментных индикаторов? Схематично изобразить одноразрядный индикатор и обозначить на нём сегменты.
8. Какие сегменты должны светится при отображении цифры соответствующей вашему номеру в списке группы?
9. Напишите код загружаемый в старший байт сдвигового регистра для индикации символа в младшем разряде пятиразрядного индикатора, в старшем разряде.

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. Отключить внешние блоки от микроконтроллера.

5. Выполнить программирование микроконтроллера через USB - программатор.
6. Отключить программатор от микроконтроллера.
7. Присоединить модуль семисегментного индикатора к микроконтроллеру.
8. Подключить выносной блок питания микроконтроллера и включить его в сеть.
9. Нажимая и отпуская ключ, соответствующий варианту задания запротоколировать реакцию устройства.
10. При обнаружении ошибок функционирования произвести повторную отладку программы.

5 Контрольные вопросы

1. Какие уровни напряжений, в соответствии со схемой подключения семисегментных матриц, должны быть на их электродах для светящихся и погашенных сегментов активного разряда?
2. Какие уровни напряжений, в соответствии со схемой подключения семисегментных матриц, должны быть на электродах пассивных разрядов?
3. Какие уровни напряжений, в соответствии со схемой подключения, должны быть на выходах сдвигового регистра, управляющего сегментами пассивных разрядов?
4. Какой величины должен быть ток светящегося сегмента активного разряда?

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. результаты экспериментальной проверки программы на прототипной системе (протокол лабораторных испытаний);

12. ответы на контрольные вопросы.

13. при защите лабораторной работы иметь на персональном компьютере/съёмном носителе работоспособный проект.

7. ИССЛЕДОВАНИЕ УСТРОЙСТВА И ФУНКЦИОНИРОВАНИЯ ДИНАМИЧЕСКОЙ ИНДИКАЦИИ

1 Цель работы

Целями работы является:

1. изучение принципов построения устройств динамической индикации и клавиатуры;
2. изучение принципов организации обслуживания внешних устройств по прерываниям системного таймера;
3. изучение организации декодирования с применением таблиц, размещаемых в программной памяти.

2 Модуль клавиатуры и дисплея с последовательным интерфейсом

Клавиатура и дисплей относятся к основным видам функциональных узлов присутствующих в любом измерительном приборе. Наиболее просто они реализуются при наличии в составе измерительного прибора микропроцессора/микроконтроллера. Применение интерфейса SPI позволяет существенно сократить количество линий управления микропроцессора/микроконтроллера при минимальных дополнительных аппаратных затратах. Это объясняется необходимостью использования токовых буферов светодиодных семисегментных индикаторов, снижающих электрическую нагрузку на выводы микроконтроллера. Использование сдвиговых регистров позволяет одновременно поддержать интерфейс SPI и функцию буферирования. Схема используемого в лабораторном макете модуля клавиатуры и дисплея представлена на рисунке 1.

ИС DD2-DD3 1533ИР24 – образуют 16-битовый сдвиговый регистр. При этом DD2 (разряды D0-D7) управляет сегментами индикатора, а DD3 – коммутирует разряды индикатора через

ключи на биполярных транзисторах VT1-VT5 (разряды D8-D12). Дисплей реализует принцип динамической индикации. В каждый фиксированный момент времени активен лишь один разряд индикатора. Разряды D8-D12 активизируются поочерёдно. D12 – управляет старшим, D8 – младшим разрядом индикатора. Выходные линии регистра DD3 управляют и линейками клавиатуры. В данном случае каждая линейка содержит только один ключ (S1-S5). Таким образом, в устройстве реализуется часто используемый принцип совмещения функций выводов. Сигнальная линия OUT эквивалентна линии MOSI интерфейса SPI. Линия C - SCK. Количество входных линий можно увеличивать, подключая к ним дополнительные ключи.

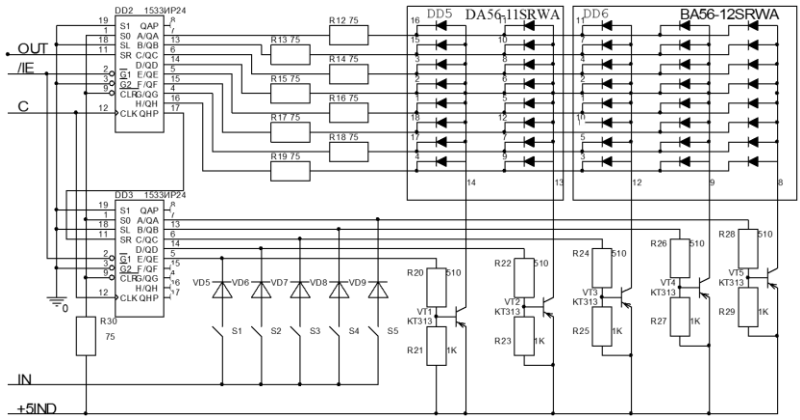


Рисунок 1 – Клавиатура и устройство индикации

Резисторы R13-R19 определяют ток сегментов светодиодных матриц. Так как разряды устройства индикации включаются поочерёдно, на один и тот же интервал времени, то среднее значение тока необходимое для поддержания яркости свечения следует поддерживать равным номинальному значению тока сег-

мента за счёт увеличения максимального импульсного тока пропорционально числу разрядов N. При этом величина сопротивления резисторов R13-R19:

$$R = \frac{E - U_0 - U_T}{NI_{SNOM}}, \quad (1)$$

где E – э.д.с. источника питания, U_0 – напряжение логического нуля буфера, UT- напряжение насыщения транзисторного ключа, I_{SNOM} – номинальный ток сегмента светодиодной матрицы.

Максимальный ток коллектора транзисторных ключей VT1-VT5 – равен сумме токов всех сегментов матрицы (с учётом десятичной точки -8):

$$I_K = 8 \times NI_{SNOM}. \quad (2)$$

Номинальное значение сопротивлений резисторов R20, R22, R24, R26, R28, определяющих токи насыщения баз транзисторных ключей:

$$R_{B1} \leq \frac{E - U_{BE} - U_0}{I_K + U_{BE}/R_{B2}}, \quad (3)$$

где U_{BE} – падение напряжения на базо-эмиттерном переходе открытого транзистора, RB_2 – сопротивление резисторов, шунтирующих базо-эмиттерные переходы транзисторов.

Следует отметить, что для применения в устройствах динамической индикации матрицы должны гарантировать отношение максимального импульсного тока к номинальному току сегмента не менее N.

Для микроконтроллера устройство вывода – символьный семисегментный дисплей может интерпретироваться как 13-битный сдвиговый регистр (старшие 3 разряда не используются). Устройство ввода – клавиатура представляет собой матричное поле 5×1, активизируемое старшими 5-ю битами 13 – битного слова. Выходные данные клавиатуры представляют собой однобитное слово – состояние клавиатуры (ключ замкнут/разомкнут). Признаком активности линейки и, соответственно, замыкания

ключа является низкий уровень сигнала на линии IN. При разомкнутом ключе или пассивности линейки на линии IN должна генерироваться логическая единица. Для этого на линию необходимо подать высокий уровень напряжения с помощью резистора. В качестве этого резистора целесообразно использовать встроенный подтягивающий резистор порта.

Идентификатором замкнутого ключа является 5-и битный унитарный код в разрядах D12-D8 сдвигового регистра. Самый старший разряд отвечает за активность старшего разряда устройства индикации. Необязательный сигнал /IE высоким уровнем гасит индикаторы и исключает эффект паразитной засветки сегментов при загрузке нового кода. Интерфейсные сигналы формируются программно на выходах порта B микроконтроллера (рисунок 2):

PB0 – C, сигнал синхронизации;

PB1 – OUT, последовательные выходные данные;

PB2 – IN, вход – состояние ключей клавиатуры;

PB3 – /IE, выход – разрешение индикации после загрузки очередного кода символа.

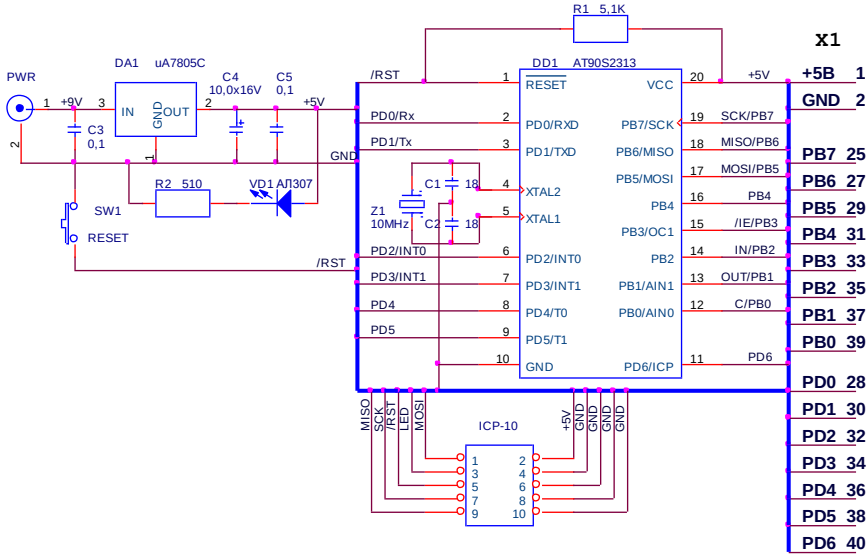


Рисунок 2 - Схема платы микроконтроллера

3 Программная поддержка устройства динамической индикации

Так как помимо поддержки устройства индикации и клавиатуры микроконтроллер может взаимодействовать и с другими устройствами, а также выполнять обработку получаемых данных и реализовывать функции управления, то необходимо правильно выбрать методы взаимодействия с внешними устройствами. Функции микроконтроллера при обслуживании устройства динамической индикации сводятся формированию номера очередного обслуживаемого разряда, к преобразованию соответствующего ему отображаемого символа в семисегментный код, передаче составного кода (унитарный номер активного разряда и код отображаемого разряда) по последовательному интерфейсу в сдвиговый регистр. Эта процедура выполняется циклически для всех пяти разрядов устройства индикации. Частота обновления выбирается так, чтобы исключить видимое мерцание индикаторов. В данном

случае частота обновления всех пяти символов принята равной 100 Гц. Таким образом, новый символ должен передаваться каждые 2 мс (5 символов за 10 мс). Очередной 13 битный код должен храниться в буфере. Объём буфера выбирается исходя из ресурсоёмкости задачи в целом и наличия свободной памяти или свободных регистров. Минимальный его объём два байта, а максимальный 6 байт (1 байт номер текущего отображаемого символа и 5 байт - коды всех отображаемых на дисплее символов). В первом случае увеличивается нагрузка на микроконтроллер за счёт необходимости декодирования очередного выводимого символа. Во-втором случае декодирование должно осуществляться для всех 5-и отображаемых символов лишь при изменении отображаемой информации.

Темп обновления отображаемых данных может задаваться цикличностью выполняемых микроконтроллером задач, что далеко не всегда возможно. Чаще этот темп задаётся системным таймером, периодически прерывающим выполнение программы и вызывающим подпрограмму вывода данных. Данные для очередной передачи могут подготавливаться как в обработчике прерываний, так и в фоновом процессе.

Для передачи данных используется программная реализация интерфейса SPI, разработанная в предыдущей лабораторной работе.

Для преобразования номера отображаемого разряда в унитарный код и отображаемого символа - в семисегментный код используются таблицы, размещаемые в программной памяти. Такой метод широко применяется, в том числе, при синтезе сигналов произвольной формы, благодаря простоте реализации и высокой скорости.

Суть метода состоит в следующем. Упорядоченные коды (по возрастанию/убыванию) размещаются в соседних ячейках программной памяти. Адрес размещения первого кода считается базовым. Для преобразования необходимо считать код из памяти

по адресу равному базовому + смещение (при упорядочении в порядке возрастания). Поскольку преобразуемые данные являются цифрами, то смещение представляет собой отображаемую цифру.

3 Подготовка к лабораторной работе

1. Изучить команды чтения данных из памяти программ.
2. Изучить команды работы с оперативной памятью.
3. Проанализировать структуру программы, приведенной в приложении. Найти и выделить в ней фрагменты, отвечающие за работу с таблицами преобразования в унитарный и семисегментный коды. Изучить работу указанных фрагментов программы.
4. По тексту программы определить и выписать функции основной программы и обработчика прерываний.
5. Оценить абсолютное время исполнения обработчика прерываний для частоты тактового генератора 10 МГц и в процентах от интервала прерываний.
6. Определить из текста программы номер таймера, используемого в качестве системного и режим его работы. По технической документации на микроконтроллер выписать имена используемых регистров конфигурации и назначение их битов для данного таймера.
7. Пересчитать настройки таймера для частоты кварцевого генератора 4 МГц.
8. Назначьте ключам клавиатуры следующие функции*:

S0 - приостановка счёта;
S1 – продолжение счёта;
S4 – сброс и повторное начало счёта.

Напишите подпрограммы для каждой из заданных функций.

10. Найти строки программы, определяющие частоту регенерации дисплея. Рассчитать значения установок таймера для частот регенерации 80, 60, 40, 20 Гц.

**Каждый из студентов вводит только одну функцию в соответствии с указаниями преподавателя.*
Письменно ответить на вопросы для самоконтроля.

1. В какой регистровой паре находится текущий адрес таблицы ПЗУ?
2. В каком регистре находятся данные считываемые из программной памяти?
3. В какой регистровой паре хранятся адреса отображаемых данных?
4. Напишите адреса ячеек ОЗУ, соответствующие данным отображаемым каждым разрядом индикатора.

4 Программа исследований и порядок работы

1. Создать проект и ввести текст программы из приложения.
2. Компилировать проект и протоколировать сообщения.
3. При отсутствии ошибок, отключить модуль динамической индикации от микроконтроллера и запрограммировать микроконтроллер.
4. Отключить программатор и блок питания от микроконтроллера.
5. Подсоединить модуль динамической индикации, а затем блок питания.
6. Составить протокол испытания устройства, записав в него алгоритм функционирования модуля и измеренное значение темпа обновления значений на дисплее.
7. Если темп существенно отклоняется от 1 сек, ввести соответствующую коррекцию в настройки таймера.

8. Ввести в исходную программу, разработанный при домашней подготовке модуль и повторно компилировать программу.
9. Загрузив программу в соответствии с п.п.3-5 провести испытания и сопроводить их протоколом испытаний.
10. Устанавливая поочерёдно частоты регенерации 80, 60, 40, 20 Гц (коды загрузки таймера определены при домашней подготовке) определить порог заметности мерцаний дисплея.

5 Контрольные вопросы

1. Какой следует выбирать частоту регенерации устройства индикации для комфортного восприятия?
2. Какие изменения придётся сделать в программе, если в качестве системного таймера использовать Таймер 0? К каким последствиям это может привести?
3. Какие функции обработчика прерываний можно перенести в основную программу?
4. В чём с вашей точки зрения может состоять смысл такого переноса?
5. Определите максимальное и среднее значение тока сегмента индикатора. Сравните с паспортными значениями. Сделайте выводы о корректности расчётных значений.

6 Содержание отчёта

Отчёт должен содержать:

1. титульный лист;
2. наименование работы и цель исследований;
3. схему электрическую функциональную устройства индикации и краткое описание её работы;

4. оценку значений резисторов в соответствии с предложенной методикой их оценки и результаты сравнения с приведенными на схеме номиналами. Светодиодные матрицы выбрать самостоятельно.
5. привести оценку загрузки микроконтроллера обслуживанием устройства индикации, сделанную при домашней подготовке.
6. исходный текст программы;
7. описание команд работы с памятью программ и оперативной памятью (форматы, флаги, адреса регистров для размещения адресов и данных);
8. описание модулей программы и их функций;
9. граф-схему алгоритма;
10. ответы на вопросы для самопроверки;
11. текст и протокол отладки самостоятельно разработанного модуля с анализом ошибок (при наличии);
12. протоколы экспериментальной проверки программ и анализ результатов;
13. ответы на контрольные вопросы.

Приложение 1

Исходный текст программы обслуживания модуля динамической индикации

```
.def SREG_COPY = r15 ; копия флагового регистра
.def tmp = r16
.def ind_n = r17          ; Номер индицируемого разряда (0 - 4)
.def ind_p = r18          ; Битовая позиция активного разряда
.def ind_b0 = r19         ; Буфер индикации (5 разрядов)
.def ind_b1 = r20
.def ind_b2 = r21
.def ind_b3 = r22
.def ind_b4 = r23

.def ind_c = r24          ; Счётчик сдвигов
.def symbol = r25         ; регистр для обработки символа
.def tmp_c = r26          ; Накопитель секундных интервалов

;***** Bit definitions

.equ     ind_max = 5      ; Количество разрядов инидикатора


.equ     I_S      = PB0   ; Ind_Sync синхровывод
.equ     I_D      = PB1   ; Ind_Date вывод данных
.equ     K_I      = PB2   ; Key_In вход состояния ключей (5 кл)
.equ     I_E      = PB3   ; Разрешение индикации (0)

;***** Const definitions

.equ     Kdt1     = 2499  ; Коэффициент деления таймера 2499
                          ; предделитель -8. Итого 8*2500=20000
.equ     ind_b =19        ; базовый адрес буфера индикации
.equ     RAMB = 0x60      ; начало ОЗУ

;*****************************************************
```

```
; Точка старта и таблица прерывания
;
.cseg
.org $00
reset_hnd:
        rjmp start          ; начало программы (обход табл.прер.)

        .org OC1addr    ; размещ. вектора прерываний
tim1_hnd:
        rjmp tim1_cmp ; на обработчик прерываний таймера

;*****************************************************
; вначале памяти размещаем  подпрограммы и обработчик пре-
рываний
; Табличное преобразование цифр в семисегментный код

dec_7seg:
        ldi     ZL,low(seg_tbl*2)
        ldi     ZH,high(seg_tbl*2)      ; нач. адрес табл. декодир.

        clr     tmp             ; текущий 16-битовый адрес
        add ZL,symbol           ; 7-сегментного кода  в ПЗУ
        adc     ZH,tmp          ;

        lpm                     ; 7-сегм. код в регистре R0
        mov     symbol,R0
        ret

;*****************************************************
; Обработчик прерываний таймера 1

; Управление динамической ИНДИКАЦИЕЙ и КЛАВИАТУ-
РОЙ.
; Индикаторы  с общим анодом. Обслуживаются 16 битовым
сдвиговым регистром
```

```
    ; Регистр управляется 3 выводами процессора
; Последовательный вход предназначен для последов. опроса
ключей
; I_S    = PB0  ; Вывод порта Ind_Sync
; I_D    = PB1  ; Вывод порта - последовательные данные
; K_I    = PB2  ; Вход клавиатурной линейки (5 ключей)
; I_E    = PB4  ; Разрешение индикации (низким уровнем)

; Управление индикацией осуществляется последовательным
; выводом во внешний 16 битовый регистр двух байт:

; старший байт - его 5 младших разрядов управляют активно-
стью
; разряда индикации (LOW - активен) через отрытие ключей
; на p-n-p транзисторах  и  одновременно сканируют клавиатуру,
; три старших разряда не используются

; младший байт - сегментный код, активный сегмент - LOW

; За одно прерывание индицируется один разряд и опрашивается
; один ключ (частота прерываний 500 Гц). Все 5 разрядов
; обрабатываются за 10 мс
; т.е. частота регенерации всех  разрядов дисплея -100 Гц

tim1_cmp:

        sbi PORTB, I_E      ; погасить индикатор (HIGH)
        push tmp            ; сохранить рабочий регистр

        lds     tmp,SREG    ; сохранить флаги (криво в AVR),
        push tmp            ; т.к. PUSH-POP работают с r0-r31

        cbi PORTB,I_S       ; Синхроимпульс LOW

;---------------------------------------------------------------
; Табличное декодирование в унитарный код
; Цифра находится в ind_n - унитарный код в ind_p
```

```
ucod:
        ldi     ZL,low(ucod_tbl*2)
        ldi     ZH,high(ucod_tbl*2)    ; нач/ адрес табл.декодир.

        clr     tmp             ; 16-битный. адрес декодируемого
        add     ZL,ind_n        ; унитарного кода  в ПЗУ
        adc     ZH,tmp          ;

        lpm                     ; унитарный код  в  R0
        mov     ind_p,R0        ; сохранить в ind_p
;---------------------------------------------------------------
; Последовательный вывод 8 разрядов управления
; подключением индикатора (старшим вперёд)
; три старших разряда не используются

l_pos:
        sbrc ind_p,7            ; если очередной сдвинутый бит 1
        sbi PORTB,I_D           ; выходной бит в 1

        sbrs ind_p,7            ; если очередной бит 0
        cbi     PORTB,I_D       ; сбросить выходной бит

        sbi PORTB,I_S           ; Синхроимпульс HIGH
        cbi PORTB,I_S           ; Синхроимпульс LOW

        cpi tmp,7       ; 8 разрядов вытолкнуты - управление
        breq le_pos     ; разрядами завершено

        inc tmp         ;
        rol ind_p       ; следующий бит
        rjmp l_pos
le_pos:
; ---------------------------------------------------------------
; Последовательный вывод 8 битов сегментного кода.
; Номер символа в ind_n. Символ в tmp
```

```
        andi ind_n,7            ; ограничение разрядности
                                ; номера отображаемого разряда
        clr ind_c               ; сбросить счётчик сдвигов

        clr     zh              ; косвенная загрузка
                                ; со  смещением
        mov     zl,ind_n          ; отображаемого символа
        ldd     symbol,Z+ind_b  ; через Z=ind_n+ind_b0

        rcall dec_7seg          ; преобразов. цифру в 7-сег.код
        mov tmp,symbol          ; и поместить её во времен. рег.
l_seg:
        sbrc tmp,7              ; если очередной бит очищен
        sbi PORTB,I_D           ; пропустить передачу  1

        sbrs tmp,7              ; если очередной бит установлен
        cbi PORTB,I_D           ; пропустить передачу  0

        sbi PORTB,I_S           ; Синхроимпульс HIGH
        cbi PORTB,I_S           ; Синхроимпульс LOW

        cpi ind_c,7             ; если  8 бит вытолкнуты -
        breq le_seg             ; завершить

        inc ind_c               ; если не завершено -
        rol tmp                 ; следующий бит
        rjmp l_seg

le_seg:
        cbi PORTB, I_E          ; включить индикатор (LOW)
        inc     ind_n           ; следующ. отображаемый разряд
        cpi     ind_n,ind_max   ; не последний, продолжить
        brlo    le_ind  ;
        clr ind_n       ; подготовить к новому циклу индикации
        ldi     ind_p,0xFE      ; счётчик и указатель позиции
```

```
;================================================

; ЧАСЫ. Цикл индикации 5-разрядов -10 мс
;
; 0-4 -> накопитель минутного интервала (60 сек),
; мин, дес.мин, часы,
; дес.час, сутки (до 30) упаков. двоично-десятичный формат

L_CLOCK:
    dec tmp_c ; декремент счётчика секундного  интервала
    brbc SP1,le_clock              ; ещё не секунда выйти

        ldi     tmp_c,100          ; при секунде перезагрузить
        ld      tmp,Y              ; нарастить счётчик секунд
        inc     tmp                ; в RAM
        cpi     tmp,60             ; не минута ли?
        brne    l_no_min           ; не она уйти на сохранение
        clr     tmp                ; минута! Чистим счётчик секунд
        st      Y,tmp              ; и сохраняем

l_no_min:
        st      Y,tmp                                          ;
;-----------------------
        inc ind_b0
        cpi     ind_b0,10
        brne    le_clock
        clr ind_b0
l_1:
        inc ind_b1
        cpi     ind_b1,10
        brne    le_clock

        clr ind_b1
l_2:
```

```
        inc ind_b2
        cpi     ind_b2,10
        brne    le_clock

        clr ind_b2
l_3:
        inc ind_b3
        cpi     ind_b3,10
        brne    le_clock

        clr ind_b3
l_4:
        inc ind_b4
        cpi     ind_b4,10
        brne    le_clock

        clr ind_b4
;-----------------------
le_ind:                 ; конец обработчика прерываний
le_clock:
        pop     tmp
        sts SREG,tmp    ; восстановить флаги

        pop     tmp     ; восстановить рабочий регистр
        reti

;================================================
; Основная программа
;================================================

start:
  cli               ; запретить прерывания

; инициализация выводов последовательного интерфейса

  sbi DDRB, I_S  ; PB0 на вывод I_Sync
  sbi DDRB, I_   ; PB1 на вывод I_Date
```

```
  sbi DDRB, I_E      ; PB3 на вывод Ind_ENable
  cbi DDRB, K_I      ; PB2 на ввод  Key_In

  ldi tmp,low(RAMEND)
  out SPL,tmp        ; инициализировать вершину стека

; Настройка таймера 1 в качестве системного
; Предделитель Кдп=8, таймер Кдт=2500.
; Общий Kd=8x2500=20 000.
; Для F=10МГц интервал прерываний - 2 мс.
; 5-прерываний для регенерации 5 - разрядов дисплея
; Программное деление до 1 с.

        ldi tmp,(1<<OCIE1A)
        out TIMSK,tmp          ; разрешить режим сравнения

        clr tmp                ; запретить ШИМ и вывод
        out TCCR1A,tmp         ;

        ldi tmp,(1<<CTC1)+(1<<CS11) ; Timer1  compare interrupt
        out TCCR1B,tmp              ; prescaler = 8

        ldi     tmp,high(Kdt1) ; коэффициент пересчёта таймера1

        out     OCR1AH,tmp  ; загрузка регистра сравнения A
        ldi     tmp,low(Kdt1)
        out     OCR1Al,tmp
        clr     ind_n          ; номер активного разряда
        clr ind_c              ; сбросить счётчик сдвигов
        clr     YH             ; загрузить указатель
        ldi     YL,RAMB        ; начала RAM
;--------------------
        clr ind_b0             ; очистить буфер отображения
        clr ind_b1
        clr ind_b2
        clr ind_b3
```

```
        clr ind_b4
;--------------------
        sei                     ; Разрешить прерывания
main:
        rjmp  main              ; Пустой цикл, т.к. нет действий
;*******************************************************
; Таблица 7-сегментных кодов
;
seg_tbl:
                        ; HGFE DCBA  HGFE DCBA
.db 0xc0,0xf9           ; 1100 0000 -0  1111 1001 -1
.db 0xa4,0xb0           ; 1010 0100 -2  1011 0000 -3
.db 0x99,0x92           ; 1001 1001 -4  1001 0010 -5
.db 0x82,0xf8           ; 1000 0010 -6  1111 1000 -7
.db 0x80,0x90           ; 1000 0000 -8  1001 0000 -9

; Таблица преобразования цифры в унитарный код
;
ucod_tbl:

.db     0b11111110, 0b11111101
.db     0b11111011, 0b11110111
.db     0b11101111, 0b11011111
```

8. ПЕРЕПРОГРАММИРУЕМАЯ ПАМЯТЬ МИКРОКОНТРОЛЛЕРОВ AVR

1 Цель работы

Изучение особенностей организации и использования перепрограммируемой памяти (EEPROM) семейства микроконтроллеров AVR.

2 Энергонезависимая перепрограммируемая память микроконтроллеров AVR

Энергонезависимая электрически стираемая память (EEPROM) используется для хранения различных констант, таблиц перекодировок, калибровочных коэффициентов и других данных, которые необходимо хранить при отключенном питании микроконтроллера.

AT90S2313 [1] содержит 128 байт электрически стираемой энергонезависимой памяти. 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 обеспечивается регистром адреса, регистром данных и управляющим регистром, адреса которых указаны в таблице 9.

Таблица 9. Регистры управления EEPROM AT90S2313.

$1E($3E)	EEAR	EEPROM AddressRegister	Регистр адреса энергонезависимой памяти
$1D($3D)	EEDR	EEPROM DataRegister	Регистр данных энергонезависимой памяти

$1C($3C)	EECR	EEPROM ControlRegister	Регистр управления энергонезависимой памятью

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.

Для записи в ячейку EEPROM необходимо в регистры адреса EEARH и EEARL (EEPROM Address Register) записать адрес ячейки, в которую нужно записать байт. После чего необходимо выждать 2.5-4 мс до готовности памяти к записи. О готовности к записи сигнализирует флаг EEWE (EEPROM Write Enable) регистра управления состоянием EECR, когда он будет равен 0, то память готова к следующей записи. Информация, которую нужно записать, помещается в регистр EEDR (EEPROM Data Register). После чего активируется бит EEMWE (EEPROM Master Write Enable), а затем, в течении четырех тактов, нужно установить бит EEWE и байт будет записан в выбранную ячейку. Если в течении четырех тактов не установить бит EEWE, то бит EEMWE сбросится. Это сделано для защиты от случайной записи в EEPROM память.

Чтение происходит по схожему алгоритму. После сигнала о готовности памяти нужный адрес заносится в регистры EEARH и EEARL. Затем выставляется бит чтения EERE (EEPROM Read Enable) и следующей командой копируем из регистра данных EEDR нужное значение в регистр общего назначения.

3 Подготовка к лабораторной работе

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

1. Построить схему микроконтроллерного устройства с индикатором в виде 8 светодиодов, подключенных к порту B микроконтроллера и клавиатуры из двух кнопок, подключенных к разрядам D2-D3 порта D.
2. Изучить особенности работы EEPROM-памяти микроконтроллера AVR, описанные в технической документации.
3. Изучить шаблон программы, выполняющей следующие функции:

- При каждом нажатии на ключ “SW0” инкрементируется значение, хранящееся в ячейке EEPROM с заданным адресом (циклически) и выводится на светодиодный индикатор;
- При каждом нажатии на ключ “SW1” декрементируется значение (циклически), хранящееся в ячейке EEPROM с заданным адресом и выводится на светодиодный индикатор;
- Если не нажат ни один из ключей, не изменяется изображение.
- После нажатия кнопки “RESET” на светодиодный индикатор выводится значение, сохраненное в заданной ячейке EEPROM.

```
;***** EEProm_asm
.list
.include "2313def.inc"
.def r_Adr =r16              ; переменная, хранящая адрес
.def r_Data =r21             ; переменная данных,
                             ; выводимых в память и на табло
.def r_Inp =r17              ; переменная входных данных с кнопок
;***** Const definitions
.equ ADR = XXXXXXX           ; семисегм. код адреса
```

```
;***** Инициализация
RESET:
        ldi r_Data,Low(RAMEND) ;установка стека
        out XXXXXX,r_Data

        ser r_Data              ; r_Data = $FF
        out DDRB,r_Data         ; PORTB = все на вывод
        out PORTD,r_Data        ; включить подтягивающие
                                ; резисторы порта D
        ldi     r_Adr,ADR       ; Загружаем адрес ячейки EEPROM и
        rcall   eeread          ; вызываем процедуру чтения.
        out PORTB,r_Data        ;отображаем на индикаторе

;**** Контроль ключей SW0 и SW1
loop:
        sbis PIND,2             ; SW0 нажат?
        inc r_Data              ; инкрементировать Temp
        sbis PIND,3             ; SW1 нажат?
        dec r_Data              ; декрементировать Temp
        in r_Inp,XXXX           ; читать входы порта D
        cpi r_Inp,0x7F
        breq outled
        ldi     r_Adr,XXX       ; загружаем адрес ячейки
        rcall   eewrite         ; вызываем процедуру записи.
        out PORTB,r_Data        ; отобразить на индикаторе
outled:
        rjmp loop               ; продолжить после задержки
 eewrite:
        sbic    EECR,EEWE       ; ждем готовн.и памяти к записи.
        rjmp    eewrite  ; пока не очистится флаг EEWE
        out     XXXX,r_Adr      ; загружаем адрес нужной ячейки
        out     XXXX,r_Data     ; и сами данные
        sbi     EECR,EEMWE; взводим предохранитель
        sbi     EECR,EEWE       ; записываем байт
        ret                     ; возврат из процедуры
 eeread:
        sbic    EECR,EEWE       ; ждем заверш. прошлой записи.
        rjmp    eeread          ; цикл
```

```
out     XXXX, r_Adr     ; загружаем адрес нужной ячейки
sbi     EECR,EERE       ; выставляем бит чтения
in      r_Data, XXXX    ; забираем результат
ret
```

4. Используя шаблон программы вставить необходимые инструкции микроконтроллера AVR (описание инструкций приведено в [2]).
5. Представить программу в виде функциональных модулей с кратким описанием функции каждого модуля в текстовой форме.
6. Опираясь на представление программы в виде функциональных модулей и текст программы составить граф-схему алгоритма.
7. Продумать последовательность и способ отладки каждого модуля программы. Привести план отладки.

Для самоконтроля письменно ответить на приведенные далее вопросы.
1. Как задать режим работы конкретной линии порта ввода/вывода (ввод или вывод)?
2. Как выполнить запись в ячейку памяти EEPROM?
3. Как выполнить чтение из ячейки памяти EEPROM?

4 Программа исследований и порядок работы

1. Рассчитать шестнадцатеричный код адреса ячейки A= 10*n+5, где n – номер компьютера.
2. Создать проект и ввести текст программы, созданной при самостоятельной подготовке к лабораторной работе.
3. Содержимое регистра адреса установить в соответствии с расчётами.
4. Компилировать проект и протоколировать 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 протоколировать содержимое модифицируемых регистров и EEPROM, сопоставляя с предсказанными значениями.

6. После сброса в пошаговом режиме наблюдать и протоколировать изменение содержимого заданной ячейки памяти.

7. Подключить стенд к программатору, а последний – к USB порту компьютера и выполнить программирование микроконтроллера.

8. По завершении программирования загруженная программа начнёт исполняться автоматически. Нажимая поочерёдно ключи, запротоколировать реакцию прототипного устройства.

5 Контрольные вопросы

1. Для чего необходимо ожидать установления бита EEWE в регистре EECR при записи в память произвольного значения.

2. Какие команды можно использовать для чтения значения ячейки памяти EEPROM.

3. В каких ситуациях нельзя гарантировать корректную запись в EEPROM.

4. Каким образом можно увеличить количество циклов записи/стирания значений, хранящихся в EEPROM.

6 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, адреса регистра вывода и адреса ячейки;
11. дамп памяти программ, копированный из окна программатора и результаты сравнения его с листингом программы;
12. результаты экспериментальной проверки программы на прототипной системе;
13. дамп памяти EEPROM, после экспериментальной проверки на прототипной системе (нажимания клавиш).
14. 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 “ATMEL” [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.