

«Қостанай жоғары политехникалық колледжі» КМҚК
КГКП «Костанайский политехнический высший колледж»

Учебно-методический комплекс

**ПМ 06 Программирование цифровых устройств на базе
микроконтроллеров**

СОДЕРЖАНИЕ

Лекционные занятия	3
Лекция 1. Микроконтроллеры, их возникновение и применение.....	3
Лекция 2. Микроконтроллеры семейства Atmel AVR.....	6
Лекция 3. Общее устройство МК, организация памяти	9
Лекция 4. Работа с периферийными устройствами.....	12
Лекция 5. Прерывания и режимы энергосбережения	15
Лекция 6. Прерывания и режимы энергосбережения	18
Лекция 8. Арифметические и логические операции	21
Лекция 9. Программирование таймеров	24
Лекция 10. Аналоговый компаратор и АЦП	31
Лекция 11. Программирование SPI	33
Лекция 12. Программирование SPI	36
Лекция 13. Программирование UART/USART	39
Практические занятия	45
Практическая работа 1. Микроконтроллеры семейства Atmel AVR.....	45
Практическая работа 2. Особенности применения AVR в схемах	48
Практическая работа 3. Общее устройство МК, организация памяти.....	49
Практическая работа 4. Виды памяти МК	51
Практическая работа 5. Работа с периферийными устройствами.....	53
Практическая работа 6. Работа с периферийными устройствами.....	56
Тестовые задания	58
Лабораторные работы.....	85

Лекционные занятия

Лекция 1. Микроконтроллеры, их возникновение и применение

Микроконтроллер — это специальная микросхема, предназначенная для управления различными электронными устройствами. Микроконтроллеры впервые появились в том же году, что и микропроцессоры общего назначения (1971).

Разработчики микроконтроллеров придумали остроумную идею — объединить процессор, память, ПЗУ и периферию внутри одного корпуса, внешне похожего на обычную микросхему. С тех пор производство микроконтроллеров ежегодно во много раз превышает производство процессоров, а потребность в них не снижается.

Микроконтроллеры выпускают десятки компаний, причем производятся не только современные 32-битные микроконтроллеры, но и 16-, и даже 8-битные (как i8051 и аналоги). Внутри каждого семейства часто можно встретить почти одинаковые модели, различающиеся скоростью работы ЦПУ и объемом памяти.

Именно поэтому самые старые типы микроконтроллеров еще до сих пор в ходу — они многое могут: от автоматического открывания дверей и включения полива газонов до интеграции в систему «умный дом». При этом существуют и более мощные микроконтроллеры, способные выполнять сотни миллионов операций в секунду и обвязанные периферией «до зубов». У них и задачи соответствующие. Таким образом, разработчик сначала оценивает задачу, а уж потом выбирает под нее подходящее «железо».

На сегодняшний день существует более 200 модификаций микроконтроллеров, совместимых с i8051, выпускаемых двумя десятками компаний, и большое количество микроконтроллеров других типов. Популярностью у разработчиков пользуются 8-битные микроконтроллеры PIC фирмы Microchip Technology и AVR фирмы Atmel, 16-битные MSP430 фирмы TI, а также 32-битные микроконтроллеры, архитектуры ARM, которую разрабатывает фирма ARM Limited и продаёт лицензии другим фирмам для их производства.

С появлением однокристальных микроЭВМ связывают начало эры массового применения компьютерной автоматизации в области управления. По-видимому [источник не указан 826 дней], это обстоятельство и определило термин «контроллер» (англ. controller — регулятор, управляющее устройство).

В связи со спадом отечественного производства и возросшим импортом техники, в том числе вычислительной, термин «микроконтроллер» (МК) вытеснил из употребления ранее использовавшийся термин «однокристальная микроЭВМ».

Первый патент на однокристальную микроЭВМ был выдан в 1971 году инженерам М. Кочрену и Г. Буну, сотрудникам американской Texas

Instruments. Именно они предложили на одном кристалле разместить не только процессор, но и память с устройствами ввода-вывода.

В 1976 году американская фирма Intel выпускает микроконтроллер i8048. В 1978 году фирма Motorola выпустила свой первый микроконтроллер MC6801, совместимый по системе команд с выпущенным ранее микропроцессором MC6800. В 1980 году Intel выпускает следующий микроконтроллер: i8051. Удачный набор периферийных устройств, возможность гибкого выбора внешней или внутренней программной памяти и приемлемая цена обеспечили этому микроконтроллеру успех на рынке. С точки зрения технологии микроконтроллер i8051 являлся для своего времени очень сложным изделием — в кристалле было использовано 128 тыс. транзисторов, что в 4 раза превышало количество транзисторов в 16-разрядном микропроцессоре i8086.

В СССР велись разработки оригинальных микроконтроллеров, также осваивался выпуск клонов наиболее удачных зарубежных образцов. В 1979 году в СССР НИИ ТТ разработали однокристалльную 16-разрядную ЭВМ К1801BE1, микроархитектура которой получила название «Электроника НЦ».

На 2013 год существовало более 200 модификаций микроконтроллеров, совместимых с i8051, выпускавшихся двумя десятками компаний, и большое количество микроконтроллеров других типов. Популярностью у разработчиков пользуются 8-битные, 16-битные и 32-битные микроконтроллеры PIC фирмы Microchip Technology, микроконтроллеры AVR фирмы Atmel (с 2016 года производятся фирмой Microchip), 16-битные MSP430 фирмы TI, а также 32-битные микроконтроллеры архитектуры ARM, которую разрабатывает фирма ARM Limited и продаёт лицензии другим фирмам для их производства. Несмотря на популярность в России микроконтроллеров, упомянутых выше, на 2009 год мировой рейтинг по объёму продаж, по данным Gartner Group, выглядел иначе: первое место с большим отрывом занимала Renesas Electronics, на втором — Freescale, на третьем — Samsung, затем шли Microchip и TI, далее — все остальные.

Но давайте разберёмся, чем smd микроконтроллеры 14 pin отличаются от 12 пиновых и как применять микроконтроллеры для чайников.

Для начала стоит обозначить, что область применения МК – гигантская, каждый современный автомобиль, холодильник и любой электрический прибор, если не учитывать различные адаптеры и модули, содержат в себе тот самый однокристалльный (чаще поликристалльный) чип.

Ведь без него было бы невозможно, в принципе, контролировать приборы и каким-либо образом ими манипулировать.

А назначение устройства выливается напрямую из терминологии, описанной выше, ведь любой МК, по своей сути, – маленький процессор, обрабатывающий команды, способный принимать и передавать данные, а в исключительных случаях, даже сохранять их в постоянной памяти.

В предыдущих пунктах мы оперировали абстрактными понятиями, теперь пришло время перейти к реальным и практическим примерам. Принцип работы любого, даже самого сложного контроллера, сводится к следующему алгоритму:

1. Он принимает определённые переменные или другие данные, которые прежде должны быть преобразованы в двоичный сигнал. Это необходимо, поскольку на низшем уровне система способна воспринимать лишь 2 состояния – есть сигнал или нет сигнала. Такой принцип называют аналоговым. Существует аналогичный алгоритм, когда сигнал присутствует постоянно, но меняется по частоте – цифровой. У них множество различий, как в областях применения, так и в особенностях работы сигнала, но суть одна – процессор способен воспринимать лишь значения 0 и 1, или true и false, и не важно, какими путями микропроцессоры и микроконтроллеры будут их считывать.

2. Во внутренней памяти устройства хранится набор специальных инструкций, который позволяет, путем базовых математических преобразований, выполнять какие-то действия с полученными данными. Именно эти базовые операнды и берутся на вооружение компилируемых языков программирования, когда необходимо написать библиотеку готовых функций. Остальные нюансы языков программирования – это уже синтаксис и теория алгоритмов. Но в результате, всё сводится к базовым операндам, которые превращаются в двоичный код и обрабатываются внутренней системой процессора.

3. Всё, что было получено и сохранено после обработки, выдается на выход. На самом деле, данный пункт выполняется всегда, единственная разница, что выходом может быть и преобразование состояния объекта какой-то системы. Простейшим примером станет замыкание электрической цепи, в случае, если на специальный датчик подать ток, вследствие чего загорится лампочка. Здесь всё зависит от типа устройства, так, 8051 микроконтроллер может выполнять несколько видов выводов, имея 14 пинов, а какой-то другой – всего один, ведь у него 1 пин на выход. Количество выходов влияет на многопоточные свойства девайса, иными словами, возможность выводить информацию сразу на несколько устройств или совершать несколько действий одновременно.

В целом, любой моно или поликристалльный блок работает по этому алгоритму, разница лишь в том, что второй – способен параллельно выполнять несколько расчетов, а первый имеет конкретный список действий, который должен выполнить последовательно.

Лекция 2. Микроконтроллеры семейства Atmel AVR

AVR – семейство восьмибитных микроконтроллеров фирмы Atmel, впервые выпущенные в 1996 г. Они представляют собой мощный инструмент, универсальную основу для создания современных экономичных встраиваемых систем многоцелевого назначения. Идея разработки нового RISC-ядра принадлежит двум студентам Норвежского университета наук и технологий (г. Тронхейм) – Альфу Богену (Alf-Egil Bogen) и Вегарду Воллену (Vegard Wollen). В 1995 г. Боген и Воллен решили предложить американской корпорации Atmel выпускать новый 8-битный RISC-микроконтроллер и снабдить его Flash-памятью для программ на одном кристалле с вычислительным ядром. Идея была одобрена Atmel Corporation, и в конце 1996 г. был выпущен опытный микроконтроллер AT90S1200, а во второй половине 1997 г. корпорация Atmel приступила к серийному производству нового семейства микроконтроллеров. Новое ядро было запатентовано и получило название AVR. Существует несколько трактовок данной аббревиатуры. Кто-то утверждает, что это Advanced Virtual RISC, другие полагают, что не обошлось здесь без инициалов разработчиков Alf Egil Bogen Vegard Wollan RISC. Микроконтроллеры AVR имеют гарвардскую архитектуру (программа и данные находятся в разных адресных пространствах) и систему команд, близкую к идеологии RISC. Процессор AVR имеет 32 8-битных регистра общего назначения, объединенных в регистровый файл. В отличие от «идеального» RISC, регистры не абсолютно равноправны: – три «сдвоенных» 16-битных регистра-указателя X (r26:r27), Y (r28:r29) и Z (r30:r31); – некоторые команды работают только с регистрами r16...r31; – результат умножения (в тех моделях, в которых есть модуль умножения) всегда помещается в r0:r1. Система команд микроконтроллеров AVR Система команд микроконтроллеров AVR весьма развита и насчитывает в различных моделях от 90 до 133 различных инструкций. Большинство команд занимает только 1 ячейку памяти (16 бит).

Большинство команд выполняется за 1 такт. Все множество команд микроконтроллеров AVR можно разбить на несколько групп: – команды логических операций; – команды арифметических операций и команды сдвига; – команды операции с битами; – команды пересылки данных; – команды передачи управления; – команды управления системой. Управление периферийными устройствами осуществляется через адресное пространство данных. Для удобства существуют «сокращенные команды» IN/OUT. Семейства и версии микроконтроллеров Стандартные семейства: – tinyAVR (ATtinyxxx): Флеш-память до 16 Кб; SRAM до 512 б; EEPROM до 512 б; Число линий ввод-вывода 4-18 (общее количество выводов 6-32); Ограниченный набор периферийных устройств. – megaAVR (ATmegaxxx): Флеш-память до 256 Кб; SRAM до 8 Кб; EEPROM до 4 Кб; Число линий ввода-вывода 23-86 (общее количество выводов 28-100); Аппаратный умножитель; Расширенная система команд и периферийных устройств. – XMEGA AVR (ATxmegaxxx):

Флеш-память до 384 Кб; SRAM до 32 Кб; EEPROM до 4 Кб; Четырехканальный DMA-контроллер; Инновационная система обработки событий. На основе стандартных семейств выпускаются микроконтроллеры, адаптированные под конкретные задачи: – со встроенными интерфейсами USB, CAN, контроллером LCD; – со встроенным радиоприемопередатчиком – серии ATAxxxx, ATAMxxx; – для управления электродвигателями – серия AT90PWMxxx; – для автомобильной электроники; – для осветительной техники.

Кроме указанных выше семейств, ATMEL выпускает 32-разрядные микроконтроллеры семейства AVR32. Версии контроллеров:

– AT(mega/tiny)xxx – базовая версия.

– ATxxxL – версии контроллеров, работающих на пониженном (Low) напряжении питания (2,7 В).

– ATxxxV – версии контроллеров, работающих на низком напряжении питания (1,8 В).

– ATxxxP – малопотребляющие версии (до 100 нА в режиме Power-down), применена технология рiсoPower (анонсированы в июле 2007)[1], повыводно и функционально совместимы с предыдущими версиями. – ATxxxA – уменьшен ток потребления, перекрывается весь диапазон тактовых частот и напряжений питания двух предыдущих версий (также, в некоторых моделях, добавлены новые возможности и новые регистры, но сохранена полная совместимость с предыдущими версиями). Микроконтроллеры «А» и «не А» с точки зрения программатора ничем не отличаются.

– ATxxx-ууPI – корпус DIP

– ATxxx-ууPU – корпус DIP, бессвинцовый припой

– ATxxx-ууAI – корпус TQFP

– ATxxx-ууAU – корпус TQFP, бессвинцовый припой,

где уу (цифры 8/10/16/20) перед индексом означают максимальную частоту, на которой микроконтроллер может стабильно работать при нормальном для него напряжении питания.

Краткие характеристики встроенной периферии МК МК AVR имеют развитую периферию, многофункциональные, двунаправленные порты ввода-вывода со встроенными подтягивающими резисторами. Конфигурация портов ввода-вывода задается программно. В качестве источника тактовых импульсов может быть выбран: – кварцевый резонатор; – внешний тактовый сигнал; – внутренний RC-генератор (частота 1, 2, 4, 8 МГц). Внутренняя флеш-память команд до 256 Кб (не менее 10 000 циклов перезаписи).

Отладка программ осуществляется с помощью интерфейсов JTAG или debugWIRE. Внутреннее EEPROM данных до 4 Кб (100 000 циклов). Внутренняя SRAM до 8 Кб время доступа 1 такт. Внешняя память объемом до 64 Кб (Mega8515 и Mega162). Таймеры с разрядностью 8, 16 бит. ШИМ-модулятор (PWM) 8-, 9-, 10-, 16-битный. Аналоговые компараторы. АЦП (ADC) с дифференциальными входами, разрядность 10 бит (12 для XMEGA AVR): программируемый коэффициент усиления перед АЦП 1, 10 и 200;

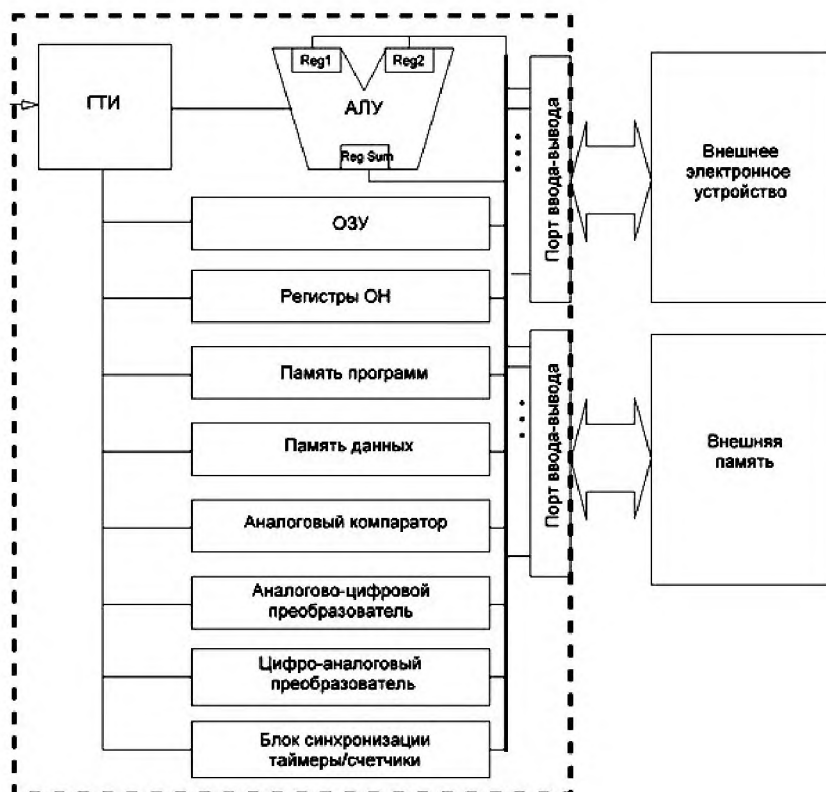
опорное напряжение 2,56 В. Различные последовательные интерфейсы, включая:

- двухпроводной интерфейс TWI, совместимый с I²C;
- универсальный синхронно/асинхронный приемопередатчик UART/USART;
- синхронный последовательный порт Serial Peripheral Interface (SPI).

Популярность микроконтроллеров AVR очень высока. С каждым годом они захватывают все новые и новые ниши на рынке. Не последнюю роль в этом играет соотношение показателей цена/быстродействие/энергопотребление, являющееся весьма привлекательным на рынке 8-битных микроконтроллеров. Кроме того, постоянно растет число выпускаемых сторонними производителями разнообразных программных и аппаратных средств поддержки разработок устройств на их основе.

Лекция 3. Общее устройство МК, организация памяти

В отличие от ПК он не является полностью самостоятельным (автономным) изделием. Микроконтроллеры, как правило, встраиваются в гаджеты, игрушки и в другие исполнительные устройства, функционированием которых они управляют.



Арифметико-логическое устройство

служит для производства логических и арифметических операций, выполняет работу процессора совместно с регистрами общего назначения.

Оперативно запоминающее устройство

служит для временного хранения информации во время функционирования микроконтроллера.

Память программ

является одним из основных структурных элементов. Она основана на постоянном запоминающем устройстве с возможностью перепрограммирования, и служит для сохранения микропрограммы управления работой микроконтроллером. Она называется прошивкой.

Память данных

используется в некоторых моделях микроконтроллеров для записи различных постоянных величин, табличных данных и т.д. Эта память имеется не во всех микроконтроллерах.

порты ввода-вывода.

Их также используют для подключения внешней памяти, различных датчиков, исполнительных устройств, светодиодов, индикаторов. Интерфейсы портов ввода-вывода разнообразны: параллельные, последовательные, оборудованные USB выходами, WI FI. Это расширяет возможности применения микроконтроллеров для различных сфер управления.

Центральный процессор

Этот элемент микроконтроллера содержит следующие узлы:

1. Арифметико-логическое устройство (АЛУ).
2. Специальный интерфейс, обеспечивающий распределение потока данных.
3. Генератор сигналов, вырабатываемых согласно инструкциям управляющей программы

Для работы всех модулей микроконтроллера потребуются языки высокого уровня, позволяющие адаптировать простейшие команды к виду, понятному вычислительному устройству.

Память

Современные микроконтроллеры имеют собственную энергонезависимую память, необходимую для хранения рабочих данных и заранее составленных и адаптированных программ. Последние представлены перечнем (списком) инструкций, написанных на машинном языке.

Они в точности указывают процессору, что ему предстоит делать с полученными данными. В ситуации с микроконтроллерами энергонезависимая память чаще всего представлена флеш накопителями (автономными хранилищами информационных массивов), (ПЗУ).

Оперативная память (ОЗУ или RAM) используется для промежуточного хранения машинных данных. Они безвозвратно теряются, когда от микроконтроллера отключается питающее напряжение. Еще одно средство временного хранения информации – внутренние регистры, представленные соответствующими структурами в составе МП.

Периферийные устройствам

Понятие «периферия» используется для описания аппаратных средств, позволяющих микроконтроллеру обмениваться информацией с внешними устройствами.

Основные виды периферийных средств это:

- АЦП.
- ЦАП.
- Генераторы опорного напряжения.

Помимо этого, микроконтроллеры включают в свой состав множество функциональных блоков, обеспечивающих выполнение предварительно введенной программы. Последние не относятся к периферийным средствам, как к таковым, поскольку отличаются от них по своему функционалу.

Оперативная память

Оперативная память, как правило, содержит 3 области:

- регистры общего назначения;
- служебные регистры;
- память для хранения данных.

Регистры общего назначения (РОН) находятся в непосредственной близости к АЛУ. Однако в микроконтроллерах некоторых фирм (в частности, PIC фирмы Microchip) имеется только один рабочий регистр, играющий роль одного из операндов в командах. Применение набора регистров общего назначения в сочетании с конвейерной обработкой позволяет АЛУ выполнять одну операцию (извлечение операндов из набора регистров, выполнение команды и запись результата обратно в регистр) за один такт.

Служебные регистры имеют свои имя, адрес и назначение. Они предназначены для конфигурации и обслуживания периферийных узлов микроконтроллера. Краткая характеристика служебных регистров должна быть приведена в руководстве по использованию микроконтроллера (Data Sheet).

Среди служебных регистров есть, как правило, один регистр, используемый наиболее часто в процессе выполнения программ. Это регистр состояния. Он содержит набор флагов, показывающих текущее состояние микроконтроллера. Большинство флагов автоматически устанавливаются в «1» или сбрасываются в «0» при наступлении определенных событий (в соответствии с результатом выполнения команд). Все биты этого регистра доступны как для чтения, так и для записи. Эта информация анализируется при выполнении условных переходов. При возникновении прерываний содержимое регистра состояния необходимо сохранять программно (чаще всего это является «заботой» компилятора).

Лекция 4. Работа с периферийными устройствами

Периферийные устройства — это обобщенное название устройств, подключаемых к ПК. Их разделяют на устройства ввода, вывода и ввода-вывода информации. Они могут быть как внешними, так и внутренними.

Внутренние – это те, которые устанавливаются на материнскую плату:

- Жесткий диск;
- Видеокарта;
- Сетевая карта;
- Wi-Fi адаптер;
- Звуковая карта;

И другое оборудование, которое подключается в слоты PCI, PCI Express и SATA.

Внешние – те, которые подключаются к системному блоку снаружи.

Основные:

- Монитор;
- Клавиатура;
- Мышь;
- Колонки;
- Наушники;
- Микрофон;
- Принтер;
- Сканер;
- МФУ;
- УПС.

Из дополнительных можно выделить USB устройства:

- Флешка;
- Bluetooth адаптер;
- Wi-Fi адаптер;
- Звуковая карта;
- Web камера;
- 3G и 4G модем;
- Удлинитель;
- Картридер;
- Джойстик.

А также некоторое профессиональное оборудование:

- Графический планшет;
- Проектор;
- Плоттер;
- Звуковой пульт;
- Сетевое оборудование.

Периферийные устройства работают с процессором и оперативной памятью с помощью контроллеров (от английского слова controller – устройство управления). У каждого периферийного устройства есть свой контроллер (или адаптер, который тоже является контроллером).

Контроллер периферийного устройства – это электронное устройство для управления работой периферийного устройства. Контроллер отвечает за:

- передачу данных от периферийного устройства к процессору;
- обратную передачу данных от процессора к периферийному устройству;
- согласование по времени работы медленного периферийного устройства и быстрого процессора.

Каждое внутреннее периферийное устройство подключено к компьютеру (ноутбуку) через контроллер. Каждое внешнее периферийное устройство подключается к порту ввода-вывода, у которого тоже есть свой контроллер. Допустим, периферийное устройство подключено к порту USB. Значит, функцию контроллера будет выполнять контроллер порта USB.

Контроллеры портов ввода-вывода являются универсальными контроллерами. Они умеют перестраиваться в зависимости от того, какое конкретное внешнее периферийное устройство к ним подключено в каждом конкретном случае.

Допустим, с мышкой контроллер порта USB станет работать совсем не так, как с внешним жестким диском. И совершенно иначе контроллер порта USB будет работать с принтером, если его подключить к этому же порту.

Все периферийные устройства подключены к общей шине компьютера через контроллеры. К этой же общей шине подключаются процессор и оперативная память компьютера. Таким образом все составные части компьютера (ноутбука) работают друг с другом через общую шину передачи данных.

Контроллеры быстродействующих периферийных устройства, например, контроллеры жестких дисков, могут работать с оперативной памятью в режиме прямого доступа. Это означает, что контроллеры этих устройств могут записывать/считывать данные из ячеек оперативной памяти, минуя обработку этих данных процессором. Подобный режим позволяет не перегружать процессор большими объемами обрабатываемой информации, позволяет разгружать процессор для повышения производительности компьютера.

Записывать данные от внешних устройств прямо в оперативную память обычные контроллеры периферийных устройств компьютера могут далеко не всегда. Сначала данные от контроллера попадают в процессор, и лишь потом, после обработки этих данных, они записываются в оперативную память. Но для контроллеров быстродействующих устройств может быть доступен режим прямого доступа к оперативной памяти компьютера (ноутбука). Отсюда получается значительное ускорение работы быстродействующего периферийного устройства.

Некоторые контроллеры периферийных устройств могут иметь также собственную оперативную память, и даже собственный специализированный процессор для автономной обработки данных. Это позволяет еще больше разгружать основной процессор и основную оперативную память. В итоге существенно ускоряется работа компьютера (ноутбука). Ведь основной процессор и основная оперативная память компьютера тогда может вообще не участвовать в обработке каких-то данных периферийного устройства. К таким контроллерам с собственной оперативной памятью относится, например, видеокарта, которая осуществляет вывод «картинки» на экран монитора.

Лекция 5. Прерывания и режимы энергосбережения

Начнем с базовых понятий мира электричества: Вольты и Амперы. Вольты – напряжение, оно же разность потенциалов. Напряжение задаёт источник питания, например батарейка или блок питания. Амперы – сила тока в цепи, показывает с какой силой “расходуется” электрическая энергия. Ток в цепи задаёт потребитель. (Примечание: описанное выше справедливо для источника напряжения, коим является любая батарейка/аккумулятор или обычный блок питания. Источником тока может быть специальное зарядное устройство или светодиодный драйвер, от них питать предназначенную для источника напряжения схему нельзя – сразу сгорит). Потребляемую и запасаемую энергию принято считать в Ампер*часах, работает это следующим образом: допустим, ёмкость аккумулятора составляет 1 А*ч (Ампер*час). Это означает, что такой аккумулятор сможет отдавать ток с силой 1 Ампер в течение одного часа, полностью при этом разрядившись. Если ток в цепи будет 0.5 А – аккумулятора хватит на $1 \text{ А*ч} / 0.5 \text{ А} = 2$ часа. Плата Ардуино потребляет в районе 24 мА, то есть тот же условный аккумулятор сможет питать её в течение $1000 \text{ мА*ч} / 24 \text{ мА} \sim 42$ часов. При параллельном подключении потребителей, как это обычно бывает в схеме, ток потребления суммируется. Если добавить в “схему” из предыдущего расчёта дисплей с подсветкой, который будет потреблять условно ещё 30 мА, то такая схема проработает от того же аккумулятора $1000 \text{ мА*ч} / (24+30 \text{ мА}) \sim 18.5$ часов.

Если в устройстве помимо МК есть какие-то другие модули/датчики/дисплеи/микросхемы, то больший вклад в потребление энергии будут вносить именно они, потому что МК можно погрузить в сон, а их – не всегда. Логично, что для максимальной экономии энергии нужно держать все компоненты в полностью отключенном состоянии и включать только на период активной работы: датчики – на время опроса, дисплеи и подсветки – на время взаимодействия с человеком, и тому подобное.

- Некоторые железки имеют очень удобный пин EN – enable, позволяющий логическим уровнем с МК полностью включать и выключать компонент, что позволяет очень просто управлять его состоянием.

- Некоторые микросхемы имеют встроенный режим энергосбережения, который можно активировать из программы (например, передав нужную команду по интерфейсу связи). Информацию нужно искать в даташите или библиотеке на конкретную железку.

- Если таких возможностей у железки нет – всегда можно просто разорвать ей питание при помощи транзистора или оптопары. Электромеханическое реле использовать не рекомендуется, т.к. оно само потребляет приличный ток.

- Маломощные (до 20 мА) компоненты можно питать напрямую от пинов МК, что ещё больше упрощает задачу по управлению питанием. Примечание: у “Ардуиновских” AVR весьма приличный запас по току – в районе 40 мА на пин, но на таком токе напряжение просаживается и работа

“железки” может стать нестабильной, поэтому не рекомендуется подключать на пин нагрузку выше 20 мА. К слову, у других процессоров (STM32, esp8266) максимальный ток с пинов на порядок ниже (2-5 мА) и что-то от них питать в принципе невозможно.

- Большинство “интерфейсных” микросхем при сбросе питания будут требовать повторной инициализации. У того же например lcd дисплея после отключения и включения питания нужно вызвать метод .init(), чтобы дисплей начал реагировать на остальные команды.

Честно говоря, сам микроконтроллер может работать абсолютно самостоятельно просто при наличии питания, а смена режима сна или частоты будет влиять на потребление ровно так, как написано в даташите. Если в основе проекта лежит плата ардуино – начинаем загибать пальцы: светодиоды индикации, стабилизатор питания и usb-ttl преобразователь – все они потребляют ток в холостом режиме, просто потому что они сидят на общем питании. Плата Nano в активном режиме потребляет около 24 мА, а если погрузить МК в максимальный сон – в районе 5 мА. В то же время по даташиту МК в таком режиме должен потреблять в районе 1 мкА, то есть в 5000 (пять тысяч) раз больше!!! Эти самые 5 мА потребляют перечисленные выше компоненты на плате ардуино, поэтому для создания действительно энергоэффективного проекта нужно делать свою плату и паять на неё МК, либо брать скальпель/паяльник и убирать лишнее с платы Ардуино.

У микроконтроллера есть несколько режимов энергосбережения, в каждом из которых остаются в активном режиме только некоторые из аппаратных блоков (таймеры, интерфейсы, АЦП, и т.д.). Также у мк есть блок BOD, отвечающий за постоянный мониторинг напряжения и перезагрузку в случае его падения ниже настроенного порога. Во всех режимах сна остаётся активен АЦП, его нужно отключать отдельно (всё реализовано в GyverPower). Режимы энергосбережения МК (AVR):

- IDLE
 - Легкий сон, отключается только блок CPU и Flash, пробуждается мгновенно от любых прерываний
- POWERDOWN
 - Наиболее глубокий сон, отключается всё кроме WDT и внешних прерываний, просыпается от аппаратных (обычных + PCINT) или WDT, пробуждение за 16+6 тактов (~1.375 мкс на 16 МГц). Прерывание должно быть длиннее этого времени для успешного пробуждения!
- STANDBY
 - Глубокий сон, идентичен POWERDOWN
 - + system clock активен, пробуждение за 6 тактов (0.4 мкс)
- POWERSAVE
 - Глубокий сон, идентичен POWERDOWN

+ timer 2 активен (+ можно проснуться от его прерываний), можно использовать для счета времени

- EXTSTANDBY

- Глубокий сон, идентичен

POWERSAVE

- + system clock активен, пробуждение за 6 тактов (0.4 мкс)

Самый часто используемый на практике режим – power down, самый глубокий сон. В нём отключается всё, кроме watchdog и аппаратных прерываний (обычные external и PCINT). В данном режиме МК потребляет минимальный ток (ATmega328 – чуть меньше 1 мкА), а проснуться можно только по прерыванию Watchdog таймера или по аппаратному прерыванию (по кнопке). Очевидно, что *в глубоком сне не работают таймеры и прерывания по ним*, поэтому счёт времени становится отдельной задачей (в GyverPower эта задача решена максимально удобно).

Лекция 6. Прерывания и режимы энергосбережения

Принципиальным достоинством МП является программируемость. Это означает, что, подавая на вход МП команды, можно обеспечить нужную последовательность операций, т.е. реализацию определенного алгоритма. Алгоритм решаемой задачи может быть сколь угодно сложным, необходимо лишь, чтобы этот алгоритм был разбит на шаги в соответствии с системой команд МП. Поэтому система команд важна не только с точки зрения, что МП может делать, но и как выполняется алгоритм. Наличие или отсутствие какой-либо команды или группы команд может существенно повлиять на выбор МП для конкретного применения.

Возвращаясь к языково-программным классификационным характеристикам, нельзя не упомянуть о языках программирования.

Все языки программирования условно можно разделить на три уровня:

- машинный код;
- автокод (язык ассемблера);
- языки высокого уровня (процедурные языки - BASIC, FORTRAN, PASCAL, C, MODULA-2, ADA; и языки искусственного интеллекта - LISP, PROLOG, SMALLTALK, OCCAM).

2. Машинно-ориентированные языки

Более понятные для ЭВМ - это так называемые машинно-ориентированные языки (машинный код и язык ассемблера). Более понятные для человека именуют языками высокого уровня.

Программное обеспечение на машинно-ориентированном языке экономично в эксплуатации, однако сравнительно высокая трудоемкость и длительность разработки программного обеспечения обуславливают преимущественное применение их для создания и развития программного обеспечения драйверов и операционных систем с целью наилучшего использования аппаратных особенностей каждой конкретной ЭВМ.

3. Языки высокого уровня

Алгоритмические языки (языки программирования высокого уровня общего назначения) являются машинно-независимыми, позволяют создавать компактные обозримые программы при относительно небольших затратах времени и труда программистов. Разработка программ значительно упрощается при использовании языков высокого уровня в качестве языков программирования. Однако при этом снижается эффективность программ по быстродействию и затратам памяти в сравнении с применением языка ассемблера. Но этот недостаток с лихвой перекрывается четкостью и легкостью написания программы.

Языки высокого уровня в свою очередь подразделяются на языки процедурного (или императивного) и эвристического (декларативного) стиля

программирования (языки искусственного интеллекта). Наиболее популярные

Язык	Год разработки	Разработчик	Основное применение
FORTRAN	1954	Дж. Бэкус (США)	Математические расчеты, научные исследования
BASIC	1965	Дж. Кенеми (США)	Обучение, тестовые программы
PASCAL	1971	Н.Вирт (Швейцария)	Обучение, широкое применение
C	1972	Д.М.Ричи (США)	Системное программирование
MODULA-2	1981	Н.Вирт (Швейцария)	Разработка больших программных комплексов
LISP	1960	Дж. Маккарти (США)	Системы искусственного интеллекта
PROLOG	1971	А.Колмедауэр (Франция)	Принятие решений, логический вывод
SMALLTALK	Середина 1970-х	А.Кей (Англия)	Системы диалога со средствами машинной графики
OCCAM	Начало 1980-х	Фирма INMOS (Англия)	Системы с параллельными процессами

языки программирования ПЭВМ высокого уровня приведены в таблице 1.

Кроме того, в настоящее время появились языки так называемого 4-го поколения - это языки СУБД, электронных таблиц, интегрированных систем и т.д., которые предназначены для решения узкого круга задач прикладного характера (например, обработка баз данных), но зато еще больше, по сравнению с языками общего назначения, снижают затраты времени и труда на создание выходного продукта.

Опыт применения ПЭВМ для построения прикладных систем обработки данных показывает, что самым эффективным инструментом создания контроллера являются не универсальные языки высокого уровня, а

узкоспециализированные языки - как правило языки высокого манипулирования с особенностями микропроцессора.

Система микропрограммирования является набором компактных программных продуктов для разработки программ для микропроцессоров. СМ реализована для работы на ряде компьютеров, от небольших 16-разрядных персональных машин до 32-разрядных суперминикомпьютеров.

В нем имеется ряд примеров использования как стандартных, так и имеющих особенности средств СМ. Отметим, что независимые средства ассемблера СМ очень просты и эффективны.

СМ ассемблеры - это мощные МАКРО-ассемблеры со средствами перемещения программ, с универсальными характеристиками и применением. Хотя ассемблеры созданы на базе одного и того же основного пакета, они обладают высокой степенью совместимости с ассемблерами разработчиков микропроцессоров. Основные предметы - это способы использования ассемблера, поддержка модульного программирования и связь с языками высокого уровня.

Все ассемблеры двухпроходные, выполняются как одна программа. Во время выполнения не создается временных файлов.

Все ассемблеры, так же как и XLINK, используют для внутренних вычислений 32-разрядные структуры, что позволяет виртуально генерировать код любого размера (т.е. не существует предела в 64 кБайт, что могло бы затруднить использование процессоров типа 68000).

Лекция 8. Арифметические и логические операции

AVR имеют всего 3 разновидности команды сложения. Инструкции `add Rd,Rr` (Сложение двух регистров), `adc Rd,Rr` (Сложение двух регистров с учётом переноса), позволяют складывать как однобайтовые числа, так числа произвольной разрядности. В последнем случае для связи байтов используется флаг переноса `C` в регистре `SREG`, который устанавливается всякий раз, когда разрядность суммы превысит 8 бит. Этот перенос должен быть добавлен к сумме старших байтов командой `adc Rd,Rr`:
`add R18,R16` ; сложение двухбайтовых чисел
`adc R19,R17` ; $R19:R18 = R19:R18 + R17:R16$

Сложение регистра с константой, к сожалению, отсутствует. И это приносит некоторые неудобства, так как приходится использовать две команды, одна из которых занесение числа во вспомогательный регистр, а уже вторая - сложение регистров:

```
ldi R17,0x30 ; заносим в R17 константу 0x30
add R16,R17 ; R16 = R17 + 0x30
```

Во многих случаях вышеуказанную проблему можно решить, используя команду `adiw Rdl,K` (Сложение константы со словом). Она позволяет прибавить 6-разрядную константу, лежащую в диапазоне $0 \dots 63$, к одной из 4х регистровых пар `Rdh:Rdl` (`R25:R24`, `R27:R26`, `R29:R28`, `R31:R30`). С её помощью также удобно реализовать 16-разрядный счётчик событий:

```
key_press:
    adiw R24,1 ; инкрементируем счётчик R25:R24
    sbis PINB,0 ; пока на линии 0 порта PORTB низкий
    rjmp key_press ; логический уровень
```

Из всех арифметических операций, вычитание в AVR - наиболее разнообразно по способам адресации.

Имеется две команды с прямой адресацией `POH`: `sub Rd,Rr` (Вычитание двух регистров), `sbc Rd,Rr` (Вычитание двух регистров с учётом заема). При совместном использовании они позволяют получить разность чисел любой разрядности. Для связи байт, при этом, и здесь служит флаг `C`. Но в отличие от сложения он устанавливается, когда разность оказалась отрицательной ($Rd < Rr$) и имеет в этом случае смысл заема, который должен быть вычтен из разности старших байт командой `sbc Rd,Rr`:

```
sub R18,R16 ; вычитание двухбайтовых чисел
sbc R19,R17 ;  $R19:R18 = R19:R18 - R17:R16$ 
```

Ещё две команды с непосредственной адресацией используются для тех же целей: `subi Rd,K` (Вычитание константы из регистра) и `sbcі Rd,K` (Вычитание константы из регистра с учётом заёма). Эти инструкции очень универсальны. Их с одинаковым успехом можно применять как в целях вычитания, так и сложения. В последнем случае необходимо использовать представление числа `K` в дополнительном коде (т.е. изменить знак числа $-K = 0xFF+1-K$):

```
subi R16,-0x30 ;  $R16 = R16 + 0x30 = R16 - (-0x30)$ 
```

Существует также инструкция, работающая в двухбайтовом формате, `sbiw Rd1,K` (Вычитание константы из слова). Подобно сложению `adiw Rd1,K`, с её помощью можно вычесть 6-разрядную K ($0 \dots 63$) из тех же самых регистровых пар (`R25:R24`, `R27:R26`, `R29:R28`, `R31:R30`). Главная область применения команды – реализация счётчика циклов:

`delay:`

`sbiw R24,1` ; декрементируем счётчик `R25:R24` пока
`brne delay` ; его содержимое не станет 0, формируя
. ; задержку времени $4 * R25:R24$ циклов

Группа логических инструкций представлена 6-ю командами.

Операция “НЕ” производится по команде `com Rd` (Дополнение до одного). При этом фактически производится действие $Rd \leftarrow 0xFF - Rd$, при котором инвертируются все биты регистра. Команды `and Rd,Rr` (“Логическое И” регистров), `andi Rd,K` (“Логическое И” регистра и константы) и `or Rd,Rr` (“Логическое ИЛИ” регистров), `ori Rd,K` (“Логическое ИЛИ” регистра и константы) производят соответствующие логические операции как с регистрами, так и регистра с константой. Операция же “Исключающее ИЛИ” возможна только между регистрами. Для этих целей служит команда `eor Rd,Rr` (“Исключающее ИЛИ” регистров). Если провести операцию “Исключающее ИЛИ” регистра с самим собой (`eor Rd,Rd`), то будет получен нулевой результат ($Rd \leftarrow Rd \text{ EOR } Rd = 0$). Это свойство часто применяется, когда необходимо отчистить регистр, а команда `eor Rd,Rd` может иметь при этом альтернативную форму записи `clr Rd` (Очистить регистр). Кроме того, с помощью последовательности всего трех команд `eor Rd,Rr` можно обменять содержимое двух регистров (команда обмена `POHов` у AVR отсутствует) местами не используя ни одной дополнительной ячейки памяти:

`eor R16,R17` ; $R16 = R16 \text{ XOR } R17$
`eor R17,R16` ; $R17 = R17 \text{ XOR } (R16 \text{ XOR } R17) = R16$
`eor R16,R17` ; $R16 = (R16 \text{ XOR } R17) \text{ XOR } R16 = R17$

Все описанные логические операции могут использоваться, например, в следующем контексте:

`in R16,PORTB` ; заносим для изменения `PBB PORTB` в `POH R16`
`ori R16,(1«PB0»)|(1«PB5»)` ; устанавливаем биты 0,5 в `R16`
`andi R16,~((1«PB1»)|(1«PB2»))`; сбрасываем биты 1,2 в `R16`
`out PORTB,R16` ; выводим изменённое значение `R16` в `PORTB`
`eor R16,R16` ; обнуляем `R16`
`out PORTC,R16` ; заносим 0 в `PORTC`

Как во многих других случаях, у инструкции `ori Rd,K` существует другая форма написания `sbr Rd,K` (Установка бита(ов) в регистре), числящаяся, однако, как самостоятельная команда. Применяя её необходимо помнить, что, несмотря на аббревиатуру, на самом деле производится именно операция “Логическое ИЛИ” регистра и битовой маски, а не установка битов в регистре. Так, например, для установки бита n в регистре `Rd` правильно писать `sbr Rd,1<`

`in R16,PORTB` ; заносим для изменения `PBB PORTB` в `POH R16`

```
sbr R16,(1«PB0)|(1«PB5) ; устанавливаем биты 0,5 в R16
cbr R16,(1«PB1)|(1«PB2) ; сбрасываем биты 1,2 в R16
out PORTB,R16 ; выводим изменённое значение R16 в PORTB
clr R16 ; обнуляем R16
out PORTC,R16 ; заносим 0 в PORTC
```

Когда надо установить все биты регистра, то можно воспользоваться инструкцией `ser Rd` (Установка регистра), которая заносит константу `0xFF` в регистр `Rd`. Она является частной формой написания команды `ldi Rd,K` (Загрузка константы в регистр), относящейся к группе команд пересылки. Подобное действие можно осуществить и другими способами. Например, в результате команды `ori R16,0xFF` все биты регистра `R16` также будут установленными.

Но в отличие от `ser R16` или, что тоже самое, `ldi R16,0xFF` - будут изменены значения флагов `N` и `Z`, что может повлиять на правильный ход выполнения программы. По той же причине в ряде случаев для обнуления регистра лучше использовать `ldi R16,0` вместо `clr R16` (`eor R16,R16`); все флаги в регистре `SREG` останутся неизменными. Единственное, что мешает повсеместному применению команды `ldi Rd,K`, это ограниченный набор `POH` с которыми она работает. Только регистры `R16...R31` могут использоваться с этой инструкцией. Это существенное ограничение наложено и на все арифметические и логические команды, которые используют непосредственную адресацию (действие над регистром и константой).

На практике часто необходимо осуществить проверку регистра на ноль. Проще всего это сделать если произвести операцию “Логическое И” регистра самим с собой (`and Rd,Rd`). При этом содержимое `Rd` останется неизменным, а в регистре `SREG` будет установлен флаг нулевого результата `Z` если `Rd = 0`. Параллельно с ним будет переопределён и флаг `N`, который в случае использования знаковых чисел будет свидетельствовать об отрицательном содержимом регистра. Для инструкции `and Rd,Rn` где `Rd = Rn`, может быть использована другая форма записи в виде псевдокоманды `tst Rd` (Проверка на ноль и минус).

В тех случаях, когда необходимо изменить знак однобайтового числа используют команду `neg Rd` (Дополнение до двух). Она осуществляет действие $Rd \leftarrow 0x00 - Rd$ и используется только для чисел представленных дополнительном коде.

У AVR имеются также команды прямого и обратного счёта. Первая из них `inc Rd` (Инкремент) увеличивает на единицу содержимое регистра, а вторая `dec Rd` (Декремент), соответственно, уменьшает. Конечно, действия инкрементирования и декрементирования могут быть заменены, например, на прибавление 1 и вычитание 1 из регистра `Rd` (`subi R16,-1` эквивалентно `inc R16`, а `subi R16,1` тоже, что и `dec Rd`). Но, несмотря на это обе команды имеют большое самостоятельное значение.

Лекция 9. Программирование таймеров

Таймеры-счётчики — это такие устройства или модули в микроконтроллере, которые, как видно из названия, постоянно что-то считают. Считают они либо до определённой величины, либо до такой величины, сколько они битности. Считают они постоянно с одной скоростью, со скоростью тактовой частоты микроконтроллера, поправленной на делители частоты, которые мы будем конфигурировать в определённых регистрах.

И вот эти таймеры-счётчики постоянно считают, если мы их инициализируем.

Таймеров в МК Atmega8 три.

Два из них — это восьмибитные таймеры, то есть такие, которые могут максимально досчитать только до 255. Данной величины нам будет маловато. Даже если мы применим максимальный делитель частоты, то мы не то что секунду не отсчитаем, мы даже полсекунды не сможем посчитать. А у нас задача именно такая, чтобы досчитывать до 1 секунды, чтобы управлять наращиванием счёта светодиодного индикатора. Можно конечно применить ещё наращивание переменной до определенной величины, но хотелось бы полностью аппаратного счёта.

Но есть ещё один таймер — это полноправный 16-битный таймер. Он не только 16-битный, но есть в нём ещё определённые прелести, которых нет у других таймеров. С данными опциями мы познакомимся позже.

Вот этот 16-битный таймер мы и будем сегодня изучать и использовать. Также, познакомившись с данным таймером, вам ничего не будет стоить самостоятельно изучить работу двух других, так как они значительно проще. Но тем не менее 8-битные таймеры в дальнейшем мы также будем рассматривать, так как для достижения более сложных задач нам одного таймера будет недостаточно.

Теперь коротко о прерываниях.

Прерывания (Interrupts) — это такие механизмы, которые прерывают код в зависимости от определённых условий или определённой обстановки, которые будут диктовать некоторые устройства, модули и шины, находящиеся в микроконтроллере.

В нашем контроллере Atmega8 существует 19 видов прерываний. Вот они все находятся в таблице в технической документации на контроллер

Table 18. Reset and Interrupt Vectors

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

Два из них — это восьмибитные таймеры, то есть такие, которые могут максимально досчитать только до 255. Данной величины нам будет маловато. Даже если мы применим максимальный делитель частоты, то мы не то что секунду не отсчитаем, мы даже полсекунды не сможем посчитать. А у нас задача именно такая, чтобы досчитывать до 1 секунды, чтобы управлять наращиванием счёта светодиодного индикатора. Можно конечно применить ещё наращивание переменной до определенной величины, но хотелось бы полностью аппаратного счёта.

Но есть ещё один таймер — это полноправный 16-битный таймер. Он не только 16-битный, но есть в нём ещё определённые прелести, которых нет у других таймеров. С данными опциями мы познакомимся позже.

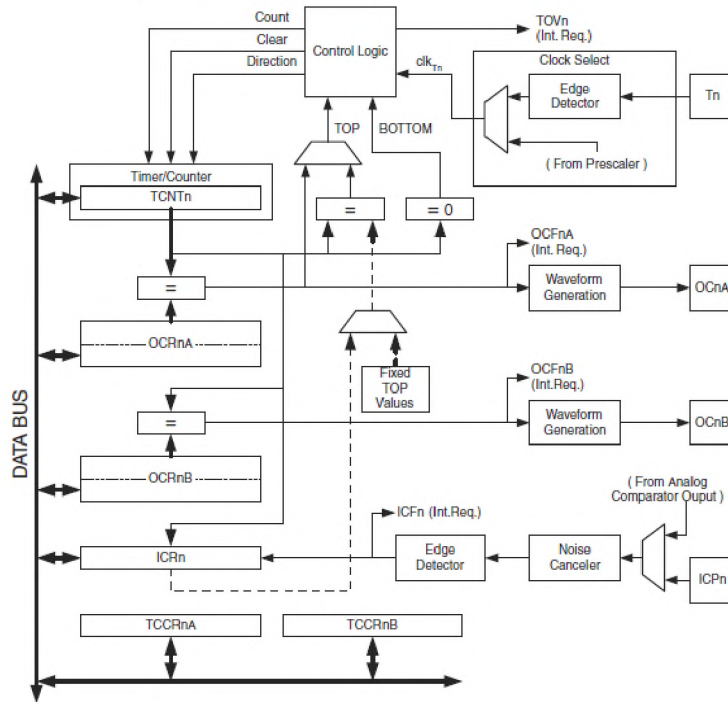
Вот этот 16-битный таймер мы и будем сегодня изучать и использовать. Также, познакомившись с данным таймером, вам ничего не будет стоить самостоятельно изучить работу двух других, так как они значительно проще. Но тем не менее 8-битные таймеры в дальнейшем мы также будем рассматривать, так как для достижения более сложных задач нам одного таймера будет недостаточно.

Теперь коротко о прерываниях.

Прерывания (Interrupts) — это такие механизмы, которые прерывают код в зависимости от определённых условий или определённой обстановки, которые будут диктовать некоторые устройства, модули и шины, находящиеся в микроконтроллере.

В нашем контроллере Atmega8 существует 19 видов прерываний. Вот они все находятся в таблице в технической документации на контроллер

Figure 32. 16-bit Timer/Counter Block Diagram⁽¹⁾



Мы видим там регистр TCNTn, в котором постоянно меняется число, то есть оно постоянно наращивается. Практически это и есть счётчик. То есть данный регистр и хранит число, до которого и досчитал таймер.

А в регистры OCRnA и OCRnB (буквы n — это номер таймера, в нашем случае будет 1) — это регистры, в которые мы заносим число, с которым будет сравниваться число в регистре TCNTn.

Например, занесли мы какое-нибудь число в регистр OCRnA и как только данное число совпало со значением в регистре счёта, то возникнет прерывание и мы его сможем обработать. Таймеры с прерываниями очень похожи на обычную задержку в коде, только когда мы находимся в задержке, то мы в это время не можем выполнять никакой код (ну опять же образно "мы", на самом деле АЛУ). А когда считает таймер, то весь код нашей программы в это время спокойно выполняется. Так что мы выигрываем колоссально, не давая простаивать огромным ресурсам контроллера по секунде или даже по полсекунды. В это время мы можем обрабатывать нажатия кнопок, которые мы также можем обрабатывать в таймере и многое другое.

Есть также регистр TCCR. Данный регистр — это регистр управления. Там настраиваются определенные биты, отвечающие за конфигурацию таймера.

Также у таймера существует несколько режимов, с которыми мы также познакомимся немного позднее.

Он состоит из двух половинок, так как у нас контроллер 8-битный и в нем не может быть 16-битных регистров. Поэтому в одной половине регистра (а физически в одном регистре) хранится старшая часть регистра, а в другом — младшая. Можно также назвать это регистровой парой, состоящей из двух

отдельных регистров TCCR1A и TCCR1B. Цифра 1 означает то, что регистр принадлежит именно таймеру 1.

Данный регистр TCCR отвечает за установку делителя, чтобы таймер не так быстро считал, также он отвечает (вернее его определённые биты) за установку определённого режима.

За установку режима отвечают биты WGM

Table 39. Waveform Generation Mode Bit Description

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation ⁽¹⁾	TOP	Update of OCR1X	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Мы видим здесь очень много разновидностей режимов.

Normal — это обычный режим, таймер считает до конца.

PWM — это ШИМ только разные разновидности, то есть таймер может играть роль широтно-импульсного модулятора. С данной технологией мы будем знакомиться в более поздних занятиях.

CTC — это сброс по совпадению, как раз то что нам будет нужно. Здесь то и сравнятся регистры TCNT и OCR. Таких режима два, нам нужен первый, второй работает с другим регистром.

Все разновидности режимов мы в данном занятии изучать не будем. Когда нам эти режимы потребуются, тогда и разберёмся.

Ну давайте не будем томить себя документацией и наконец-то попробуем что-то в какие-нибудь регистры занести.

Код, как всегда, был создан из прошлого проекта. Для протеуса также код был скопирован и переименован с прошлого занятия, также в свойствах контроллера был указан путь к новой прошивке. Проекты мы назовем Test07.

Попробуем как всегда скомпилировать код и запустить его в протеусе. Если всё нормально работает, то начинаем добавлять новый код.

Добавим ещё одну функцию, благо добавлять функции мы на прошлом занятии научились. Код функции разместим после функции `segchar` и до функции `main`. После из-за того, что мы будем внутри нашей новой функции вызывать функцию `segchar`.

Мало того, мы создадим не одну функцию, а целых две. В одну функцию мы разместим весь код инициализации нашего таймера, а другая функция будет являться обработчиком прерывания от таймера, а такие функции они специфичны и вызывать их не требуется. Когда возникнет необходимость, они вызовутся сами в зависимости от определённых условий, которые были оговорены выше.

Настройка таймера/счетчика 0

```
void timer_init(void)
{
    TCNT0=0b00000000;    //Очистка TCNT0
    OCR0=250;            //Содержимое компаратора TC0
    TCCR0=0b00001110;    //Сброс при совпадении
                        //Запуск с предделителем на 256
    TIMSK=0b00000010;    //Разрешение прерывания по событию
                        //«Совпадение» таймера/счетчика T0
    TIFR=0b00000011;    //Очистка флагов
}
```

Фрагмент основной программы

```
int main(void)
{
    ///
    asm ("cli");        //Команда ассемблера "запрет прерываний"
    init();             //Вызов функции инициализации портов
    asm ("sei");        //Команда ассемблера "разрешение прерываний"
    ///
    init_lcd();        //Вызов функции инициализации ЖКИ
    timer_init();
    ms_counter=0;
    ///
    write_str(" 00 HRS  00 MIN");    //Вызов функции вывода строки
    lcd_com(0xC3);
    write_str("00 SECONDS");
}
```

```

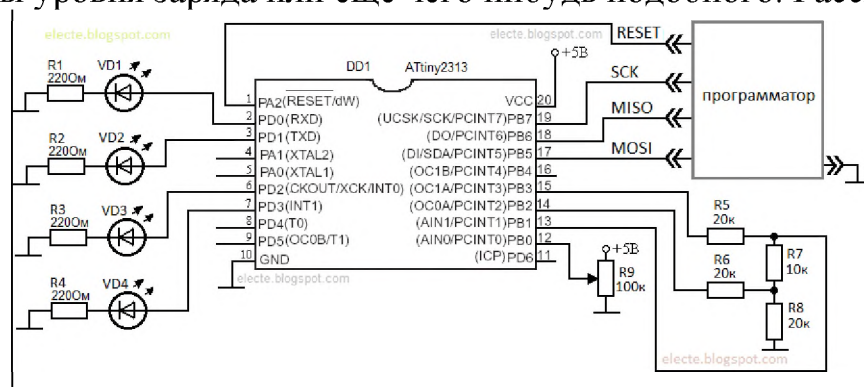
while(1)
{
    if (sec_flag==1)//Проверка установки флага события «секунда»
    {
        sec_flag=0;      //Обнуление флага события «секунда»
        tim[0]++;      //Инкремент счетчика единиц секунд
        if (tim[0]>9)    //Проверка переполнения разряда единиц секунд
        {
            tim[0]=0;
            tim[1]++; //Инкремент счетчика десятков секунд
            PORTA=~PORTA;
            if(tim[1]>5)
            {
                tim[1]=0;
                tim[2]++; //Инкремент счетчика единиц минут
                if(tim[2]>9)
                {
                    tim[2]=0;
                    tim[3]++; //Инкремент счетчика десятков минут
                    if(tim[3]>5)
                    {
                        tim[3]=0;
                        tim[4]++; //Инкремент счетчика единиц часов
                        if((tim[4]>9)&&(tim[5]<2))
                        {
                            tim[4]=0;
                            tim[5]++; //Инкремент счетчика десятков часов
                        }
                        if((tim[4]>3)&&(tim[5]==2))
                        {
                            tim[4]=0;
                            tim[5]=0; //Обнуление счетчика десятков часов
                        }
                    }
                }
            }
        }
    }
}

```

```
/* Вывод значения времени */  
    lcd_com(0x81);  
    lcd_data(TIM[5]+48);  
    lcd_data(TIM[4]+48);  
    lcd_com(0x89);  
    lcd_data(TIM[3]+48);  
    lcd_data(TIM[2]+48);  
    lcd_com(0xC3);  
    lcd_data(TIM[1]+48);  
    lcd_data(TIM[0]+48);  
    }  
}  
  
return 0;  
  
} //Окончание основной программы
```

Лекция 10. Аналоговый компаратор и АЦП

Многие современные микроконтроллеры, как правило, имеют в своем составе аналого цифровые преобразователи (АЦП) для измерения аналогового сигнала. Такая возможность часто бывает необходима. Однако не все микроконтроллеры имеют именно аппаратный встроенный АЦП. Например ATtiny2313 не имеет его. Зато он имеет аналоговый компаратор. Аналоговый компаратор, сам по себе, не может измерять аналоговое напряжение. Зато он может определить больше ли это напряжение некоторого опорного или нет. А также он работает гораздо быстрее чем полноценный АЦП. Бывают ситуации когда этого функционала достаточно, например когда надо определить превышение тока в нагрузке, включить защиту и сделать это быстро, в этом случае аналоговый компаратор будет гораздо лучше чем АЦП. Также этот компаратор может быть использован в каких нибудь преобразователях напряжения чтобы оперативно следить за напряжением. Или же небольшими внешними доработками можно сделать АЦП на основе имеющегося компаратора. Работа с аналоговым компаратором микроконтроллеров AVR достаточно проста. На основе компаратора микроконтроллера ATtiny2313 можно сделать например двухбитный АЦП. Такой АЦП не очень точный но зато очень быстрый и его можно использовать например для какой нибудь шкалы уровня заряда или ещё чего нибудь подобного. Рассмотрим схему:



Резистором R9 задается напряжение которое измеряет АЦП, вместо этого резистора можно поставить какой либо другой источник напряжения в пределах от 0 до напряжения питания микроконтроллера. На резисторах R5-R8 собрана R-2R цепь которая является двухбитным цифро аналоговым преобразователем (ЦАПом). На светодиодах VD1-VD2 сделана шкала уровня напряжения. Резисторы R1-R4 нужны для ограничения тока светодиодов VD1-VD2. Вместо R-2R цепи можно было бы использовать какие либо другие способы задания напряжения например ШИМ+RC цепь и т.о добиться гораздо большей точности но при этом скорость измерения напряжения значительно снизиться. Теперь давайте рассмотрим код программы на ассемблере:

```

.INCLUDE "tn23134def_for_avra.inc" ; загрузка предопределений

.CSEG
.ORG 0x0000

;-- инициализация стека --
LDI R16, Low(RAMPEND)
OUT SPL, R16

; -- настройка пинов --
LDI R16, 0b00001111 ; 0,1,2,3 - светодиоды
OUT DDRD, R16

LDI R16, 0b00001100 ; 2,3 - R-2R цап
OUT DDRD, R16

; -- основной цикл программы --
loop:
; - ЦАП 50% питания -
SBI PORTB, 3 ; лог. 1 на пин 3 порта B
SBI PORTB, 2 ; лог. 0 на пин 2 порта B

RCALL delay ; задержка для синхронизации компаратора и пренега

SBIS ACSR, ACO ; пропустить следующую команду если на выходе компаратор
RJMP less50

; - ЦАП 75% питания -
SBI PORTB, 3 ; лог. 1 на пин 3 порта B
SBI PORTB, 2 ; лог. 1 на пин 2 порта B

RCALL delay ; задержка для синхронизации компаратора и пренега

SBIS ACSR, ACO ; пропустить следующую команду если на выходе компаратор
RJMP less75

; - вывести на светодиодной шкале "напряжение более 75 %" -
LDI R16, 0b00001111 ; 0,1,2,3 - светодиоды
OUT PORTD, R16

RJMP loop

less75:
; - вывести на светодиодной шкале "напряжение 50...75 %" -
LDI R16, 0b00000111 ; 0,1,2,3 - светодиоды
OUT PORTD, R16

RJMP loop

less50:
; - ЦАП 25% питания -
SBI PORTB, 3 ; лог. 0 на пин 3 порта B
SBI PORTB, 2 ; лог. 1 на пин 2 порта B

RCALL delay ; задержка для синхронизации компаратора и пренега

SBIS ACSR, ACO ; пропустить следующую команду если на выходе компаратор
RJMP less25

; - вывести на светодиодной шкале "напряжение 25...50 %" -
LDI R16, 0b00000011 ; 0,1,2,3 - светодиоды
OUT PORTD, R16

RJMP loop

less25:
; - вывести на светодиодной шкале "напряжение 0...25 %" -
LDI R16, 0b00000001 ; 0,1,2,3 - светодиоды
OUT PORTD, R16

RJMP loop

; -- задержка --
delay:
LDI R20, 10
L1: DEC R20
BRNE L1

RET

```

Настройки компаратора в коде нет т.к. он уже правильно настроен по умолчанию. Есть настройка только пинов ввода/вывода и стека. В основном цикле реализован алгоритм поразрядного уравнивания и управление светодиодами. Под основным циклом есть небольшая подпрограмма задержки.

Лекция 11. Программирование SPI

SPI – последовательный синхронный интерфейс между одним мастером (главное устройство) и одним помощником (подчиненное устройство). Каждый передаваемый бит данных в любом из направлений имеет собственный тактовый импульс. Поэтому, SPI использует по крайней мере три различных сигнала:

1. SCK – тактирование последовательной связи (синхронизация)
2. SI – последовательный ввод (данные от «помощника» к «мастеру»)
3. SO – последовательный вывод (данные от «мастера» к «помощнику»)

В некоторых конфигурациях SPI использует четвертый сигнал “Выбор микросхемы” (CS), который используется для разрешения последовательной связи мастером. Данный сигнал может иметь активный уровень как низкий, так и высокий.

В связи с отсутствием общих технических требований к SPI-интерфейсу временная диаграмма тактового сигнала зависима от устройства (микросхемы), поскольку каждый производитель использует собственную временную диаграмму. Для большинства SPI протоколов можно выделить четыре типа настроек, которые обычно задаются внутренними установками “мастера” шины SPI: CPOL (полярность тактового сигнала) и CPHA (фаза тактового сигнала).

CPOL определяет активное состояние сигнала тактирования последовательной связи SPI и, следовательно, является отточечным уровнем.

CPHA определяет фазу тактового сигнала относительно бита данных на линии SO.

Не существует общей классификации SPI-протоколов и SPI-устройств, которая определяла бы установки CPOL и CPHA относительно собственно протокола. Поэтому в этом документе установки определяются следующим образом:

CPOL = 0 : тактовый сигнал имеет отточечный уровень “1”

CPOL = 1 : тактовый сигнал имеет отточечный уровень “0”

CPHA = 0 : тактовый сигнал синхронен битам данных на линии SO

CPHA = 1 : тактовый сигнал задержан на полпериода передачи бита данных относительно бита данных на линии SO

Последовательный ввод-вывод может поддерживать только SPI-форматы с CPHA = 0 в его нормальном режиме работы. При выборе режима внешнего тактирования сдвигового регистра последовательный ввод-вывод может тактироваться переключением порта относительно его вывода SCK.

Недокументированная особенность состоит в том, что последовательный ввод-вывод может быть также синхронизирован переключением бита NEG в регистре SES (выбор фронта). Поэтому, есть возможность установить первый бит данных на SO перед генерацией тактового сигнала на выводе порта.

}

Лекция 12. Программирование SPI

Иногда нужно сохранить данные так, чтобы они восстановились после перезагрузки контроллера. В этом тебе поможет EEPROM, почти все контроллеры серии AVR имеют на борту некоторое количество этой памяти. Физически и логически эта память находится в отдельном адресном пространстве, а чтение из EEPROM и запись туда осуществляется через специальные порты.

Чтобы что-то записать в EEPROM нужно в регистры адреса **EEARH** и **EEARL** (EEPROM Address Register) положить адрес ячейки в которую мы хотим записать байт. После чего нужно дождаться готовности памяти к записи – EEPROM довольно медленная штука. О готовности к записи нам доложит флаг **EWE** (EEPROM Write Enable) регистра управления состоянием **EESCR**, когда он будет равен 0, то память готова к следующей записи. Сам байт, который нужно записать, помещается в регистр **EEDR** (EEPROM Data Register). После чего взводится предохранительный бит **EEMWE** (EEPROM Master Write Enable), а затем, в течении четырех тактов, нужно установить бит **EWE** и байт будет записан. Если в течении четырех тактов не успеешь выставить бит **EWE** то предохранительный бит **EEMWE** сбросится и его придется выставлять снова. Это сделано для защиты от случайной записи в EEPROM память.

Чтение происходит примерно аналогичным образом, вначале ждем готовности памяти, потом заносим в регистры нужный адрес, а затем выставляем бит чтения **EERE** (EEPROM Read Enable) и следующей командой забираем из регистра данных **EEDR** наше число, сохраняя его в любом регистре общего назначения. Чтобы было понятно, я тебе набросал две процедуры – на чтение и на запись. Чтобы записать байт, нужно в регистры **R16** и **R17** занести младший и старший байт адреса нужной ячейки, а в регистр **R21** байт который мы хотим записать. После чего вызвать процедуру записи. Аналогично и с чтением – в регистра **R16** и **R17** адрес, а в регистре **R21** будет считанное значение.

Вот так выглядит запись в память:

```
...
LDI R16,0 ; Загружаем адрес нулевой
ячейки
1 LDI R17,0 ; EEPROM
2 LDI R21,45 ; и хотим записать в нее
3 число 45
4 RCALL EEWwrite ; вызываем
5 процедуру записи.
```

А так чтение:

```

        LDI    R16,0          ; Загружаем адрес нулевой
ячейки
        LDI    R17,0          ; EEPROM из которой хотим
1  прочитать байт
2  RCALL     EERead          ; вызываем
3  процедуру чтения. После которой
4  ; в R21 будет считанный байт.

```

Ну и, разумеется, сами процедуры чтения и записи

```

EEWrite:
        SBIC   EECR,EEWE          ; Ждем готовности памяти к за
Крутимся в цикле
        RJMP  EEWrite            ; до тех пор пока не очистится
1  EEWB
2
3  CLI                               ; Затем запрещаем прерывания.
4  OUT    EEARL,R16              ; Загружаем адрес нужной ячейки
5  OUT    EEARH,R17              ; старший и младший байт адреса
6  OUT    EEDR,R21              ; и сами данные, которые нам н
7  загрузить
8
9  SBI    EECR,EEMWE            ; взводим предохранитель
10 SBI    EECR,EEWE            ; записываем байт
11
12 SEI                               ; разрешаем прерывания
13 RET                               ; возврат из процедуры
14
15
16 EERead:
17 SBIC   EECR,EEWE          ; Ждем пока будет завершена про
18 запись.
19 RJMP  EERead                ; также крутимся в цикле.
20 OUT    EEARL, R16          ; загружаем адрес нужной ячейки
21 OUT    EEARH, R17          ; его старшие и младшие байты
22 SBI    EECR,EERE            ; Выставляем бит чтения
23 IN     R21, EEDR            ; Забираем из регистра данных резу.
24 RET

```

Да, при работе с EEPROM нужно в цикле ожидания готовности не забывать командой WDR сбрасывать Watch Dog Timer — специальный сторожевой таймер, отслеживающий зависание процессора. Если его не сбрасывать с нужной периодичностью, то он сбрасывает контроллер. Это, конечно, если Watch Dog используется. По дефолту он вырублен. Но помнить надо, иначе огребете трудно отслеживаемый глюк.

Впрочем, у EEPROM тоже есть свои прерывания. Это:

```
1 .ORG $01E
2 RETI ; (EE_RDY) EEPROM Ready
```

Лекция 13. Программирование UART/USART

Интерфейс обычно подключается к внешним системам тремя ножками (см. рис. 296). Любой двунаправленный обмен USART требует как минимум двух сигнальных выводов: входные принимаемые данные (Receive Data In, RX) и выходные передаваемые данные (Transmit Data Out, TX):

RX: вход последовательных принимаемых данных. Используются техники передискретизации для восстановления данных, чтобы отделить полезные входящие данные от шума.

TX: выход передаваемых данных. Когда передатчик запрещен, ножка выхода возвратит свою конфигурацию порта ввода/вывода (GPIO). Когда передатчик разрешен и ничего не передается, уровень выхода ножки TX находится в лог. 1. В режимах single-wire и smartcard, этот вывод I/O используется и для передачи, и для приема данных (на уровне USART данные затем принимаются на SW_RX).

Через эти выходы последовательные данные принимаются и передаются в нормальном режиме USART как фреймы. В этом процессе используется следующее:

- Состояние ожидания линии (Idle Line) до передачи или приема.
- Start-бит.
- Слово данных (8 или 9 бит), самый младший бит слова (LSB) идет первым.
- 0.5, 1, 1.5, 2 Stop-стоп-бит, показывающих завершение фрейма.
- Используется дробный генератор скорости - с 12-разрядной мантиссой и 4-битной дробной частью.
- Регистр статуса (USART_SR).
- Регистр данных (USART_DR).
- Регистр генератора скорости (USART_BRR) с 12-разрядной мантиссой и 4-битной дробной частью.
- Регистр защитного времени, Guardtime Register (USART_GTPR) в случае использования режима Smartcard.

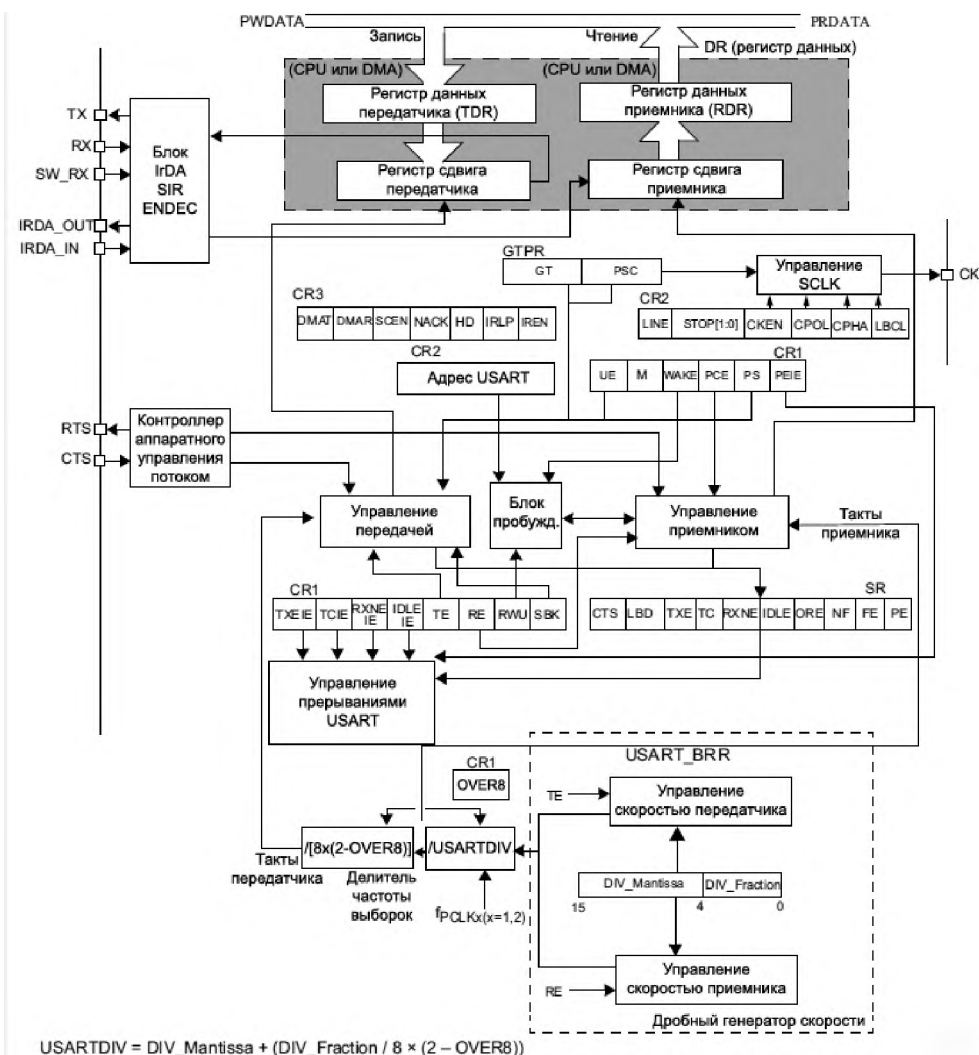
Описание регистров и их бит см. далее во врезках раздела "Регистры USART".

Для работы в синхронном режиме требуется дополнительный вывод тактов:

- СК: выход тактов передатчика. На этот вывод выдаются такты данных для синхронной передачи, соответствующей режиму SPI master (нет тактовых импульсов на битах start и stop, и программно выбираемая опция отправки тактового импульса на последнем бите данных). Параллельно и синхронно могут приниматься данные через ножку RX. Это можно использовать для управления внешними периферийными устройствами, у которых есть регистры сдвига (например драйверы LCD). Фаза и полярность тактов выбирается программно. В режиме smartcard СК может предоставлять такты для смарт-карты.

В режиме аппаратного управления потоком требуются еще 2 вывода:

- CTS: сигнал Clear To Send, блокирующий передачу данных по окончании текущей передачи (когда CTS = 1).
- RTS: сигнал Request To Send, показывающий, что USART готов принимать данные (когда RTS = 0).



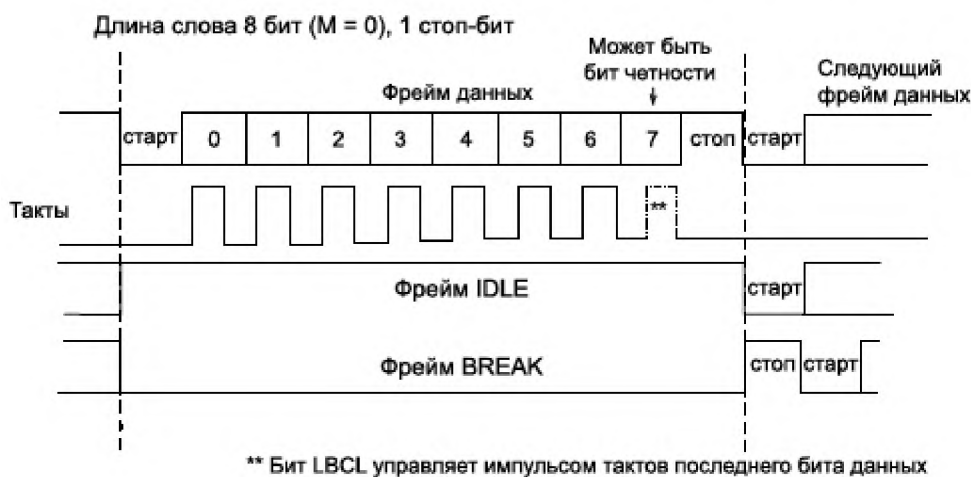
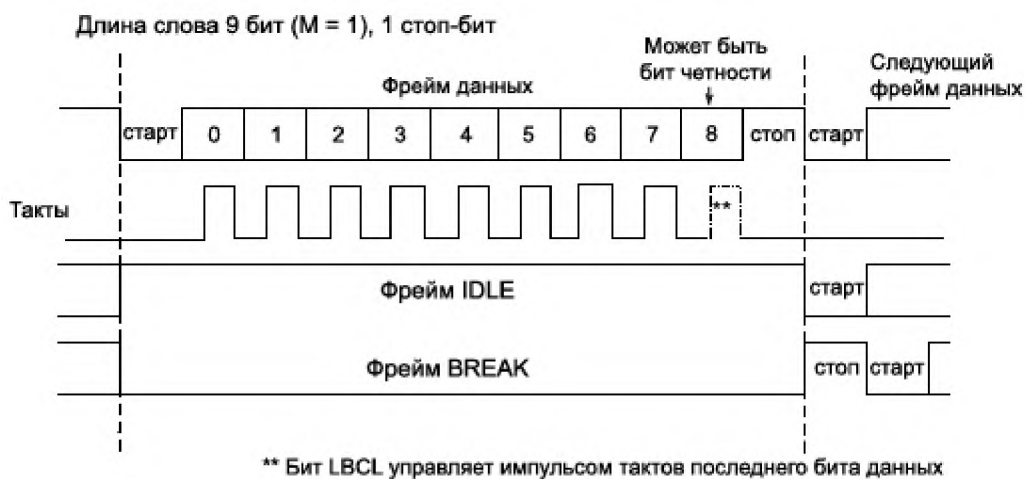
Символ USART. Для передачи можно выбрать длину слова 8 или 9 бит путем программирования бита M в регистре USART_CR1 (см. рис. 297).

Ножка TX находится в лог. 0, когда передается start-бит, и в лог. 1, когда передается stop-бит.

Idle. Символ Idle интерпретируется как фрейм (идуший после start-бита), где все биты равны 1 (все биты данных в лог. 1, это состояние сохраняется и во время передачи stop-битов).

Break. Символ Break интерпретируется, когда принимаются все 0 в течение всего периода фрейма. По окончании данных фрейма передатчик вставляет 1 или 2 stop-бита (лог. 1), чтобы подтвердить start-бит.

Передача и прием управляются общим генератором скорости (baud rate generator). Такты для передатчика и приемника генерируются, когда установлен соответствующий бит разрешения.



Лекция 12. Режимы энергосбережения и сторожевой таймер

Продолжаем тему минимизации энергопотребления Ардуино и теперь пора разобраться в режимах работы МК. Все AVR микроконтроллеры поддерживают различные режимы энергосбережения. Их описание можно найти в документации к МК в разделе Power Management and Sleep Modes. ATmega328P имеет 6 режимов:

Idle Mode (режим ожидания). В нем процессор приостанавливается, но остальная периферия - SPI, USART, аналоговый компаратор, шина TWI (I2C), таймеры/счетчики, сторожевой таймер и система прерываний - продолжают работать. Поэтому выход из режима Idle возможен как по внешнему, так и по внутреннему прерыванию. Основным преимуществом данного режима является быстрая реакция на события, приводящие к пробуждению МК. Другими словами, выполнение программы начинается сразу же после перехода из режима Idle в рабочий режим.

ADC Noise Reduction Mode (режим снижения шумов АЦП). Данный режим присутствует только в моделях МК, содержащих в своем составе АЦП и предназначен для уменьшения всевозможных наводок во время его работы. В этом режиме прекращает работу процессор и отключается синхронизация

ввода-вывода, остальные устройства продолжают работу. Вывести МК из этого режима, кроме прерывания по завершению преобразования АЦП, могут:

- прерывание от сторожевого таймера;
- прерывание по совпадению адреса от интерфейса TWI;
- прерывание от асинхронного таймера/счетчика 2;
- прерывание по готовности EEPROM;
- прерывание при изменения уровня (pin change interrupt)
- внешние прерывания INT0 и INT1;
- а так же сбросы (аппаратного, от сторожевого таймера, от схемы BOD).

Power-Down mode (режим микропотребления) - самый экономный режим, присутствует во всех AVR. В этом режиме отключаются все внутренние тактовые сигналы, соответственно, прекращается функционирование всех узлов МК, работающих в синхронном режиме. Единственными узлами, продолжающими работать в этом режиме, являются асинхронные модули МК: сторожевой таймер (если он включен), подсистема обработки внешних прерываний и блок сравнения адреса модуля TWI. Пробуждение из режима Power-Down возможно при возникновении сброса (аппаратного, от сторожевого таймера, от схемы BOD) или в результате генерации следующих прерываний:

- прерывание от сторожевого таймера;
- прерывание по совпадению адреса от интерфейса TWI;
- прерывание изменения уровня (pin change interrupt)
- внешние прерывания INT0 и INT1.

Power Save Mode (экономичный режим). Идентичен режиму Power Down с одним отличием: в режиме Power Save таймер/счетчик 2 может продолжать работу, причем как в синхронном, так и асинхронном режиме. Вывести МК из этого режима могут те же события, что и в случае использования режима Power Down, а так же прерывания от таймера/счетчика 2.

Standby Mode (режим ожидания). Этот режим рекомендуется задействовать только при использовании внешнего резонатора. Он идентичен режиму Power Down, за исключением того, что тактовый генератор продолжает функционировать. Благодаря этому пробуждение МК и переход в рабочий режим требует всего 6 тактов.

Extended Standby Mode (расширенный режим ожидания). Как и режим Standby этот режим рекомендуется использовать совместно с внешним резонатором. Он идентичен режиму Power Save, за исключением того, что тактовый генератор продолжает функционировать. Благодаря этому пробуждение МК и переход в рабочий режим требует всего 6 тактов.

Для того чтобы перевести Ардуино в один из перечисленных режимов достаточно выполнить 2 функции: `set_sleep_mode(<mode>)` для задания конкретного режима и `sleep_mode()`. Они объявлены в файле `sleep.h`, поэтому он должен быть добавлен в секцию `include` вашего скетча.

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

```
....
```

```
set_sleep_mode(SLEEP_MODE_PWR_DOWN);
```

```
sleep_mode();
```

Пробуждение по сторожевому таймеру

В основе сторожевого таймера (WatchDog Timer) лежит многоразрядный счетчик, снабженный собственным тактовым генератором. Если таймер включен, то значение счетчика будет постоянно увеличиваться и при его переполнении будет сгенерирован сигнал сброса МК. Чтобы избежать сброса по переполнению программа должна постоянно обнулять счетчик специальной командой. Если программа зависла и счетчик не был вовремя сброшен, то сигнал сброса выведет МК из зависшего состояния, таким образом повышается стабильность системы на основе МК.

Наличие собственного тактового генератора и способность работать когда другие узлы МК остановлены позволяют использовать сторожевой таймер для вывода МК из спящего режима. Для этого следует настроить его на генерацию прерывания, а не сигнала сброса. Также необходимо задать значение предделителя, чтобы получить интересующую задержку. Ниже приведен пример скетча, в котором Ардуино будет просыпаться каждые 2 секунды и мигать встроенным светодиодом для индикации пробуждения.

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

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

```
volatile bool f = 0;
```

```
void setup() {  
  pinMode(LED_BUILTIN, OUTPUT);  
}
```

```
void loop() {  
  wdt_enable(WDTO_2S); //Задаем интервал сторожевого таймера (2с)  
  WDTCSR |= (1 << WDIE); //Устанавливаем бит WDIE регистра WDTCSR для разрешения прерывания  
  set_sleep_mode(SLEEP_MODE_PWR_DOWN); //Устанавливаем интересующий нас режим  
  sleep_mode(); // Переводим МК в спящий режим  
  digitalWrite(LED_BUILTIN, f);  
}
```

```
ISR (WDT_vect) {  
  wdt_disable();  
  f = !f;  
}
```

Здесь задается интервал сторожевого таймера и режим его работы для генерации прерываний через 2 секунды. После выполнения функции `sleep_mode()` Ардуино переходит в спящий режим. При генерации прерывания от сторожевого таймера и пробуждении МК, управление передается соответствующему обработчику (ISR (WDT_vect)). После его выполнения

продолжится работа основной программы, т.е. выполнится следующая после `sleep_mode()` команда. Для задания значения предделителя сторожевого таймера можно использовать одну из следующих констант:

```
WDTO_15MS  
WDTO_30MS  
WDTO_60MS  
WDTO_120MS  
WDTO_250MS  
WDTO_500MS  
WDTO_1S  
WDTO_2S  
WDTO_4S  
WDTO_8S
```

Для получения более длительных интервалов можно использовать цикл и переводить МК в спящий режим внутри цикла. Но лучше использовать для этого часы реального времени с будильником.

Практическая работа 1. Микроконтроллеры семейства Atmel AVR

Atmel AVR представляет собой семейство универсальных 8-разрядных микроконтроллеров на основе общего ядра с различными встроенными периферийными устройствами. Возможности МК AVR позволяют решить множество типовых задач, возникающих перед разработчиками радиоэлектронной аппаратуры.

Особенности микроконтроллеров Atmel AVR.

- **Производительность порядка 1 MIPS/МГц.** MIPS (Millions of Instructions Per Second, миллион команд в секунду) — одна из самых старых и во многом формальная характеристика производительности процессоров, т. к. наборы команд для различных процессоров различаются, и, соответственно, одно и то же число инструкций на различных системах даст разную полезную работу. Тем не менее для простых 8-разрядных вычислительных систем, не содержащих команд, оперирующих с большими числами, числами с плавающей точкой и массивами данных, это неплохой показатель для сравнения их производительности. Вычислительное ядро AVR на ряде задач по производительности превосходит 16-разрядный процессор 80286.

- **Усовершенствованная RISC-архитектура.** Концепция RISC (Reduced Instruction Set Computing, вычисления с сокращенным набором команд) предполагает наличие набора команд, состоящего из минимума компактных и быстро выполняющихся инструкций; при этом такие более громоздкие операции, как вычисления с плавающей точкой или арифметические действия с многоразрядными числами, предполагается реализовать на уровне подпрограмм. Концепция RISC упрощает устройство ядра (в типовом ядре AVR содержится лишь 32 тыс. транзисторов, в отличие от десятков миллионов в процессорах для ПК) и ускоряет его работу: типовая инструкция выполняется за один такт (кроме команд ветвления программы, обращения к памяти и некоторых других, оперирующих с данными большой длины). В AVR имеется простейший двухступенчатый конвейер, когда команда выполняется в одном такте с выборкой следующей. В отличие от Intel-архитектур, в "классическом" AVR нет аппаратного умножения/деления, однако в подсемействе Mega присутствуют операции умножения.

- **Раздельные шины памяти команд и данных.** AVR (как и большинство других микроконтроллеров) имеет т. н. гарвардскую архитектуру, где области памяти программ и данных разделены (в отличие от классической архитектуры фон Неймана в обычных компьютерах, где память общая). Раздельные шины для этих областей памяти значительно ускоряют выполнение программы: данные и команды могут выбираться одновременно.

- **32 регистра общего назначения (РОН).** Atmel была первой компанией, далеко отошедшей от классической модели вычислительного ядра, в которой выполнение команд предусматривает обмен данными между

АЛУ и запоминающими ячейками в общей памяти. Введение РОН в таком количестве (напомним, что в архитектуре x86 всего четыре таких регистра, а в x51 понятие РОН, как таковое, отсутствует) в ряде случаев позволяет вообще отказаться от расположения глобальных и локальных переменных в ОЗУ и от использования стека, операции с которым усложняют и загромождают программу. В результате структура ассемблерной программы приближается к программам на языках высокого уровня. Правда, это привело к некоторому усложнению системы команд, номенклатура которых для AVR больше, чем в других RISC-семействах (хотя значительная часть инструкций — псевдонимы).

- **Flash-память программ** (10 000 циклов стирание/запись) с возможностью внутрисистемного перепрограммирования и загрузки через последовательный канал прямо в готовой схеме. О преимуществах такого подхода, ныне ставшего общепринятым, подробно рассказано во введении.

- **Отдельная область энергонезависимой памяти (EEPROM, 100 000 циклов стирание/запись)** для хранения данных, с возможностью записи программным путем, или внешней загрузки через SPI-интерфейс.

- **Встроенные устройства для обработки аналоговых сигналов:** аналоговый компаратор и многоканальный 10-разрядный АЦП.

- **Сторожевой таймер**, позволяющий осуществлять автоматическую перезагрузку контроллера через определенные промежутки времени (например, для выхода из "спящего" режима).

- **Последовательные интерфейсы SPI, TWI (I²C) и UART (USART)**, позволяющие осуществлять обмен данными с большинством стандартных датчиков и других внешних устройств (в том числе таких, как персональные компьютеры) аппаратными средствами.

- **Таймеры-счетчики** с предустановкой и возможностью выбора источника счетных импульсов: как правило, один-два 8-разрядных и как минимум один 16-разрядный, в том числе могущие работать в режиме многоканальной 8-, 9-, 10-, 16-битовой широтно-импульсной модуляции (PWM).

- **Возможность работы при тактовой частоте от 0 Гц до 16–20 МГц.**

- **Диапазон напряжений питания от 2,7 до 5,5 В** (в некоторых случаях от 1,8 или до 6,0 В).

- **Многочисленные режимы энергосбережения**, отличающиеся числом узлов, остающихся подключенными. Выход из "спящих" режимов по сторожевому таймеру или по внешним прерываниям. 46

- **Встроенный монитор питания** — детектор падения напряжения (Brown-out Detection).

Здесь перечислены далеко не все особенности, характерные для различных моделей AVR. С некоторыми другими мы познакомимся в дальнейшем, а также на практике рассмотрим перечисленные подробнее. Но

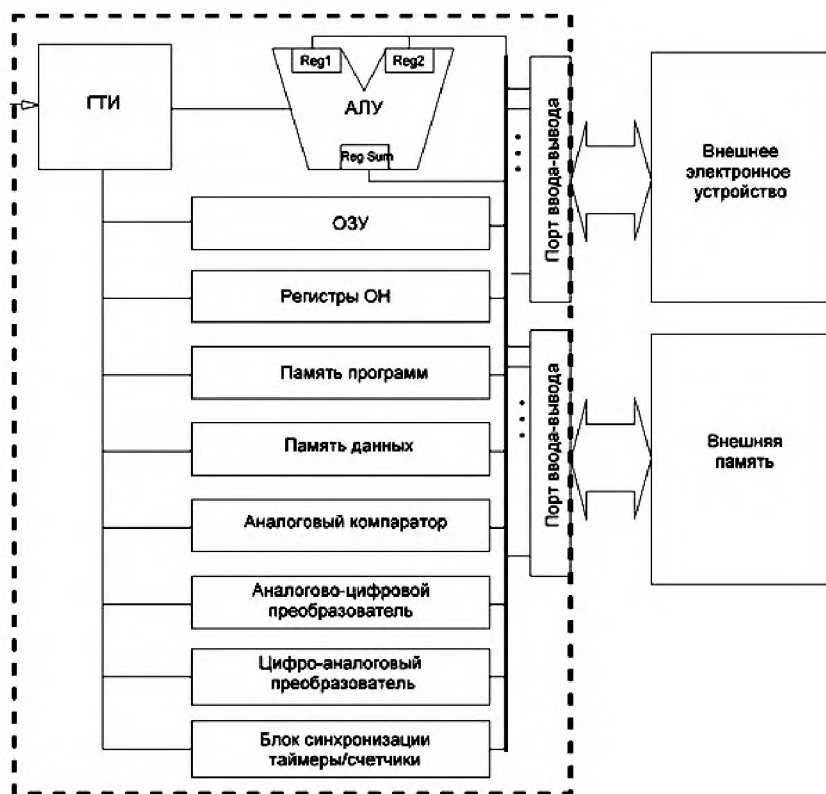
сначала дадим общую характеристику различных семейств AVR с точки зрения их преимущественного назначения.

Практическая работа 2. Особенности применения AVR в схемах

У большинства выводов МК имеется встроенный подключаемый "подтягивающий" (т. е. подсоединенный к шине питания) резистор, что, казалось бы, решает одну из обычных схемотехнических проблем, когда наличие такого резистора требуется для подключения двухвыводных кнопок или выходов с "открытым коллектором". Однако в критичных случаях необходим внешний резистор сопротивлением 2–5 кОм (в критичных для потребления случаях до 10–30 кОм). "Подтягивающий" резистор следует устанавливать не только на выводе /RESET (о чем пойдет речь в главе 2), но и в том случае, когда выводы SCK, MOSI и MISO соответствующих портов используются для программирования и подключены к программирующему разъему ISP (см. главу 5), а также по выводам внешних прерываний, если они задействованы. Если эти выводы не "подтягивать" к напряжению питания дополнительными резисторами (хотя это и не оговорено в технической документации), то не исключены ложные срабатывания внешних прерываний, перезапуск системы, а при очень мощных помехах — даже порча программы в памяти программ. С другой стороны, когда выводы программирования служат и в качестве обычных портов, сконфигурированных на выход, а в устройстве применяются режимы энергосбережения, наличие "подтягивающих" резисторов может привести к лишнему потреблению тока (при установке вывода в логический ноль через резистор потечет ток от источника питания на вход МК). Если реализован один из режимов энергосбережения, то нужно тщательно проанализировать схему, чтобы исключить ситуации, при которых через эти резисторы протекает ток. Также всегда следует устанавливать внешние резисторы при работе выводов МК на общую шину, как в интерфейсе I2C (или просто при подсоединении входа МК к выходу другого устройства с открытым коллектором, например, мониторов питания, описанных в главе 3), при подключении к двухвыводным кнопкам (особенно при наличии внешнего прерывания, см. главы 4 и 5). Сопротивление встроенного резистора (на самом деле представляющего собой, разумеется, полевой транзистор) в таких случаях слишком велико для того, чтобы электромагнитные помехи ("наводки") на нем эффективно "садились". Микросхемы AVR, как и всякая КМОП-логика, благодаря высокому порогу срабатывания эффективно защищены от помех по шине "земли". Однако они ведут себя гораздо хуже при помехах по шине питания. Поэтому не забывайте о развязывающих конденсаторах, которые нужно устанавливать непосредственно у выводов питания (керамические 0,1–0,5 мкФ), а также про качество сетевых выпрямителей и стабилизаторов

Практическая работа 3. Общее устройство МК, организация памяти

В отличие от ПК он не является полностью самостоятельным (автономным) изделием. Микроконтроллеры, как правило, встраиваются в гаджеты, игрушки и в другие исполнительные устройства, функционированием которых они управляют.



Арифметико-логическое устройство

служит для производства логических и арифметических операций, выполняет работу процессора совместно с регистрами общего назначения.

Оперативно запоминающее устройство

служит для временного хранения информации во время функционирования микроконтроллера.

Память программ

является одним из основных структурных элементов. Она основана на постоянном запоминающем устройстве с возможностью перепрограммирования, и служит для сохранения микропрограммы управления работой микроконтроллером. Она называется прошивкой.

Память данных

используется в некоторых моделях микроконтроллеров для записи различных постоянных величин, табличных данных и т.д. Эта память имеется не во всех микроконтроллерах.

порты ввода-вывода.

Их также используют для подключения внешней памяти, различных датчиков, исполнительных устройств, светодиодов, индикаторов. Интерфейсы портов ввода-вывода разнообразны: параллельные, последовательные, оборудованные USB выходами, WI FI. Это расширяет возможности применения микроконтроллеров для различных сфер управления.

Центральный процессор

Этот элемент микроконтроллера содержит следующие узлы:

4. Ар
ифметико-логическое устройство (АЛУ).
5. Сп
ециальный интерфейс, обеспечивающий распределение потока данных.
6. Ге
нератор сигналов, вырабатываемых согласно инструкциям управляющей программы

Для работы всех модулей микроконтроллера потребуются языки высокого уровня, позволяющие адаптировать простейшие команды к виду, понятному вычислительному устройству.

Практическая работа 4. Виды памяти МК

Память

Современные микроконтроллеры имеют собственную энергонезависимую память, необходимую для хранения рабочих данных и заранее составленных и адаптированных программ. Последние представлены перечнем (списком) инструкций, написанных на машинном языке.

Они в точности указывают процессору, что ему предстоит делать с полученными данными. В ситуации с микроконтроллерами энергонезависимая память чаще всего представлена флеш накопителями (автономными хранилищами информационных массивов), (ПЗУ).

Оперативная память (ОЗУ или RAM) используется для промежуточного хранения машинных данных. Они безвозвратно теряются, когда от микроконтроллера отключается питающее напряжение. Еще одно средство временного хранения информации – внутренние регистры, представленные соответствующими структурами в составе МП.

Периферийные устройства

Понятие «периферия» используется для описания аппаратных средств, позволяющих микроконтроллеру обмениваться информацией с внешними устройствами.

Основные виды периферийных средств это:

- АЦ
- П. ЦА
- ЦА
- П. Ге
- нераторы опорного напряжения.

Помимо этого, микроконтроллеры включают в свой состав множество функциональных блоков, обеспечивающих выполнение предварительно введенной программы. Последние не относятся к периферийным средствам, как к таковым, поскольку отличаются от них по своему функционалу.

Оперативная память

Оперативная память, как правило, содержит 3 области:

- регистры общего назначения;
- служебные регистры;
- память для хранения данных.

Регистры общего назначения (РОН) находятся в непосредственной близости к АЛУ. Однако в микроконтроллерах некоторых фирм (в частности, PIC фирмы Microchip) имеется только один рабочий регистр, играющий роль одного из операндов в командах. Применение набора регистров общего назначения в сочетании с конвейерной обработкой позволяет АЛУ выполнять одну операцию (извлечение операндов

из набора регистров, выполнение команды и запись результата обратно в регистр) за один такт.

Служебные регистры имеют свои имя, адрес и назначение. Они предназначены для конфигурации и обслуживания периферийных узлов микроконтроллера. Краткая характеристика служебных регистров должна быть приведена в руководстве по использованию микроконтроллера (Data Sheet).

Среди служебных регистров есть, как правило, один регистр, используемый наиболее часто в процессе выполнения программ. Это регистр состояния. Он содержит набор флагов, показывающих текущее состояние микроконтроллера. Большинство флагов автоматически устанавливаются в «1» или сбрасываются в «0» при наступлении определенных событий (в соответствии с результатом выполнения команд). Все биты этого регистра доступны как для чтения, так и для записи. Эта информация анализируется при выполнении условных переходов. При возникновении прерываний содержимое регистра состояния необходимо сохранять программно (чаще всего это является «заботой» компилятора).

Практическая работа 5. Работа с периферийными устройствами

Периферийные устройства — это обобщенное название устройств, подключаемых к ПК. Их разделяют на устройства ввода, вывода и ввода-вывода информации. Они могут быть как внешними, так и внутренними.

Внутренние – это те, которые устанавливаются на материнскую плату:

- Жесткий диск;
- Видеокарта;
- Сетевая карта;
- Wi-Fi адаптер;
- Звуковая карта;

И другое оборудование, которое подключается в слоты PCI, PCI Express и SATA.

Внешние – те, которые подключаются к системному блоку снаружи.

Основные:

- Монитор;
- Клавиатура;
- Мышь;
- Колонки;
- Наушники;
- Микрофон;
- Принтер;
- Сканер;
- МФУ;
- УПС.

Из дополнительных можно выделить USB устройства:

- Флешка;
- Bluetooth адаптер;
- Wi-Fi адаптер;
- Звуковая карта;
- Web камера;
- 3G и 4G модем;
- Удлинитель;
- Картридер;
- Джойстик.

А также некоторое профессиональное оборудование:

- Графический планшет;
- Проектор;
- Плоттер;
- Звуковой пульт;
- Сетевое оборудование.

Периферийные устройства работают с процессором и оперативной памятью с помощью контроллеров (от английского слова controller – устройство управления). У каждого периферийного устройства есть свой контроллер (или адаптер, который тоже является контроллером).

Контроллер периферийного устройства – это электронное устройство для управления работой периферийного устройства. Контроллер отвечает за:

- передачу данных от периферийного устройства к процессору;
- обратную передачу данных от процессора к периферийному устройству;
- согласование по времени работы медленного периферийного устройства и быстрого процессора.

Каждое внутреннее периферийное устройство подключено к компьютеру (ноутбуку) через контроллер. Каждое внешнее периферийное устройство подключается к порту ввода-вывода, у которого тоже есть свой контроллер. Допустим, периферийное устройство подключено к порту USB. Значит, функцию контроллера будет выполнять контроллер порта USB.

Контроллеры портов ввода-вывода являются универсальными контроллерами. Они умеют перестраиваться в зависимости от того, какое конкретное внешнее периферийное устройство к ним подключено в каждом конкретном случае.

Допустим, с мышкой контроллер порта USB станет работать совсем не так, как с внешним жестким диском. И совершенно иначе контроллер порта USB будет работать с принтером, если его подключить к этому же порту.

Все периферийные устройства подключены к общей шине компьютера через контроллеры. К этой же общей шине подключаются процессор и оперативная память компьютера. Таким образом все составные части компьютера (ноутбука) работают друг с другом через общую шину передачи данных.

Контроллеры быстродействующих периферийных устройств, например, контроллеры жестких дисков, могут работать с оперативной памятью в режиме прямого доступа. Это означает, что контроллеры этих устройств могут записывать/считывать данные из ячеек оперативной памяти, минуя обработку этих данных процессором. Подобный режим позволяет не перегружать процессор большими объемами обрабатываемой информации, позволяет разгружать процессор для повышения производительности компьютера.

Записывать данные от внешних устройств прямо в оперативную память обычные контроллеры периферийных устройств компьютера могут далеко не всегда. Сначала данные от контроллера попадают в процессор, и лишь потом, после обработки этих данных, они записываются в оперативную память. Но для контроллеров быстродействующих устройств может быть доступен режим прямого доступа к оперативной памяти компьютера (ноутбука). Отсюда

получается значительное ускорение работы быстродействующего периферийного устройства.

Некоторые контроллеры периферийных устройств могут иметь также собственную оперативную память, и даже собственный специализированный процессор для автономной обработки данных. Это позволяет еще больше разгружать основной процессор и основную оперативную память. В итоге существенно ускоряется работа компьютера (ноутбука). Ведь основной процессор и основная оперативная память компьютера тогда может вообще не участвовать в обработке каких-то данных периферийного устройства. К таким контроллерам с собственной оперативной памятью относится, например, видеокарта, которая осуществляет вывод «картинки» на экран монитора.

Практическая работа 6. Работа с периферийными устройствами

Порты ввода-вывода

Портов ввода-вывода в разных моделях может быть от 1 до 7. Номинально порты 8-разрядные, в некоторых случаях разрядность ограничена числом выводов корпуса и может быть меньше восьми. Порты обозначаются буквами A, B, C, D и далее, причем необязательно по порядку: в младших моделях могут наличествовать, например, только порты B и D (как в ATtiny2313) или вообще только один порт B (как в ATtiny1x). Для сокращения числа контактов корпуса в подавляющем большинстве случаев внешние выводы, соответствующие портам, кроме своей основной функции (двунаправленного ввода-вывода) несут также и дополнительную. Отметим, что никакого специального переключения выводов портов не требуется, просто, если вы, к примеру, в своей программе инициализируете последовательный порт UART, то соответствующие выводы порта (например, в ATmega8335 это выводы PD0 и PD1) будут работать именно в альтернативной функции, как ввод и вывод UART. При этом в промежутках между таким специальным использованием выводов они в принципе доступны, как обычные двунаправленные выводы. На практике приходится применять схемотехнические меры для изоляции функций друг от друга, поэтому злоупотреблять этой возможностью не рекомендуется: особенно это критично для выводов того порта (обычно это порт A), к которому подсоединены входы АЦП. Более того: при наличии АЦП не рекомендуется задействовать для выполнения логических функций другие выводы того же порта (которые в работе АЦП не участвуют), потому что это значительно увеличивает уровень помех (подробнее см. далее в этой главе, а также главу 10). Выводы портов в достаточной степени автономны, и их режим может устанавливаться независимо друг от друга. По умолчанию при включении питания все дополнительные устройства отключены, а порты работают на вход, причем находятся в состоянии с высоким импедансом (т. е. высоким входным сопротивлением). Работа на выход требует специального указания, для чего в программе нужно установить соответствующий нужному выводу бит в регистре направления данных (этот регистр обозначается DDRx, где x — буква, обозначающая конкретный порт, например для порта A это будет DDRA). Если бит сброшен (т. е. равен лог. 0), то вывод работает на вход (установка по умолчанию), если установлен (т. е. равен лог. 1) — то на выход. Для установки выхода в состояние 1 нужно отдельно установить, а для установки в 0 — сбросить соответствующий бит в регистре данных порта (обозначается PORTx). Направление работы вывода (вход-выход, регистр DDRx) и его состояние (0–1, PORTx) путать не следует. Регистр данных PORTx фактически есть просто выходной буфер, все, что в него записывается, тут же оказывается на выходе. Но если установить вывод порта на вход (т. е. записать в регистр направления лог. 0), как это сделано по умолчанию, то регистр данных PORTx будет играть несколько иную роль — установка его

разрядов в 0 (так по умолчанию) означает, что вход находится в третьем состоянии с высоким сопротивлением, а установка в 1 подключит к выводу "подтягивающий" (pull-up) резистор сопротивлением 35–120 кОм. Еще раз отметим, что встроенного pull-up-резистора в большинстве случаев (например, если вы подключаете ко входу выносную кнопку с двумя выводами, которая коммутируется на "землю", или при работе на "общую шину" с удаленными устройствами) оказывается недостаточно для надежной работы, и лучше устанавливать дополнительный внешний резистор параллельно этому внутреннему, с сопротивлением от 1 до 5 кОм (в критичных для потребления случаях его величину можно увеличить до 20–30 кОм). О "подтягивающих" резисторах см. также раздел "Особенности практического использования МК AVR" главы 1. Процедура чтения уровня на выводе порта, если он находится в состоянии работы на вход, не совсем тривиальна. Возникает искушение прочесть данные из регистра данных PORTx, но это ничего не даст вы получите только то, что там записано вами же ранее. А для чтения того, что действительно имеется на входе (непосредственно на выводе микросхемы), предусмотрена другая возможность: нужно обратиться к некоторому массиву, который обозначается PINx. Обращение осуществляется так же, как и к отдельным битам обычных регистров (см. главу 6), но PINx это не регистр, а просто некий диапазон адресов, чтение по которым предоставляет доступ к информации из буферных элементов на входе порта. Записывать что-то по адресам PINx, естественно, нельзя.

Тестовые задания

Тест на тему: "Микроконтроллеры Atmel, основы. Часть 1."

1. Укажите самые распространенные компании, которые занимаются производством микроконтроллеров:

- А) Microchip; +
- б) PIC;
- в) Atmel; +
- г) AVR;
- д) Intel; +
- е) Philips; +
- ж) Scinex; +
- з) Zilog; +

2. Микроконтроллеры делятся на:

- А) CISC – устройства; +
- б) RISC – устройства; +
- в) DSP – устройства;
- г) MIPS – устройства;

3. Производительность микроконтроллера измеряют:

- А) в MIPS; +
- Б) в DSP;
- В) разрядностью памяти данных;
- Г) разрядностью памяти программ;

4. Микроконтроллеры по способу программирования классифицируют на:

- А) масочно-программируемые; +
- Б) однократно программируемые; +
- В) перепрограммируемые; +
- Г) флеш-программируемые;
- Д) последовательно-программируемые;

5. Укажите какие существуют подсемейства для микроконтроллером AVR:

- а) tiny; +
- б) Classic; +
- в) mega; +
- г) normal;

д) standart;

6. В микроконтроллерах AVR обозначение EEPROM означает:

А) энергонезависимая память данных; +

Б) энергонезависимая память программ;

В) регистровая память;

Г) сторожевой таймер;

7. Память программ микроконтроллеров семейства AVR разделена на следующие области:

А) область прикладной программы; +

Б) область загрузчика; +

В) область счётчика команд;

Г) область энергонезависимой EEPROM;

Д) область регистров ввода-вывода;

8. Регистровая память микроконтроллеров семейства AVR включает:

А) 32 регистра общего назначения; +

Б) 64 регистра общего назначения;

В) область дополнительных регистров ввода-вывода; +

Г) регистры статического ОЗУ;

9. Выберите правильное утверждение:

А) последние 6 регистров общего назначения объединены в 3 шестнадцатитрибитных регистра; +

Б) последние 6 регистров общего назначения объединены в 3 тридцатидвухбитных регистра;

В) последние 8 регистров общего назначения объединены в 4 шестнадцатитрибитных регистра;

Г) последние 8 регистров общего назначения объединены в 4 тридцатидвухбитных регистра;

10. Пусть все выходы PB0...PB7 микроконтроллера ATmega16x/32x используются в качестве входов. К ним подключены кнопки, которые другими выводами подключены к шине питания +5В. Что будет находиться в регистре PinB, когда все кнопки нажаты? Что в этом случае должен содержать регистр DDRB? Что будет находиться в регистре PinB, когда нажаты все кнопки, кроме кнопки, подключённой к выводу PB7? Выберите правильные утверждения.

А) в регистре PinB будет находиться число 0b11111111; +

Б) в регистре PinB будет находиться число 0b00000000;

В) регистр DDRB будет содержать число 0b00000000; +

Г) регистр DDRB будет содержать число 0b11111111;

Д) если все кнопки нажаты кроме кнопки, подключённой к выводу PB7, то в регистре PinB в данном случае будет находиться число 0b01111111; +

Е) если все кнопки нажаты кроме кнопки, подключённой к выводу PB7, то в регистре PinB в данном случае будет находиться число 0b10000000;

11. Пусть все выходы PB0...PB7 микроконтроллера ATmega16x/32x используются в качестве выходов и подключены к светодиодам. Другие выходы светодиодов подключены через резисторы к общему проводу. Что должен содержать регистр PortB, чтобы все светодиоды были включены? Что в этом случае должен содержать регистр DDRB? Что должен содержать регистр PortB, чтобы были включены все светодиоды, кроме двух центральных? Выберите правильные утверждения:

А) чтобы все светодиоды были включены, регистр PortB должен содержать число 0b11111111; +

Б) чтобы все светодиоды были включены, регистр PortB должен содержать число 0b00000000;

В) регистр DDRB будет содержать число 0b11111111; +

Г) регистр DDRB будет содержать число 0b00000000;

Д) чтобы были включены все светодиоды, кроме двух центральных регистр PortB должен содержать число 0b11100111; +

Е) чтобы были включены все светодиоды, кроме двух центральных регистр PortB должен содержать число 0b00011000;

Ж) содержимое регистра PortB не влияет на включение и выключение светодиодов в данном случае;

12. Выберите правильные утверждения:

А) регистр SREG содержит набор флагов, показывающих текущее состояние микроконтроллера; +

Б) регистр SREG используется для подключения внешнего ОЗУ;

В) регистр SREG содержит адрес пересылаемого байта по интерфейсу SPI;

Г) регистр SREG хранит значение глобальных переменных;

13. Прямая адресация для доступа к данным в микроконтроллерах AVR семейства mega делится на:

А) прямая адресация одного РОН; +

Б) прямая адресация двух РОН; +

В) прямая адресация РВВ; +

Г) прямая адресация ОЗУ; +

Д) прямая адресация с индексным регистром;

Е) прямая косвенная адресация;

14. Укажите, какой способ адресации изображён на рисунке (см. рис. 1.1.):

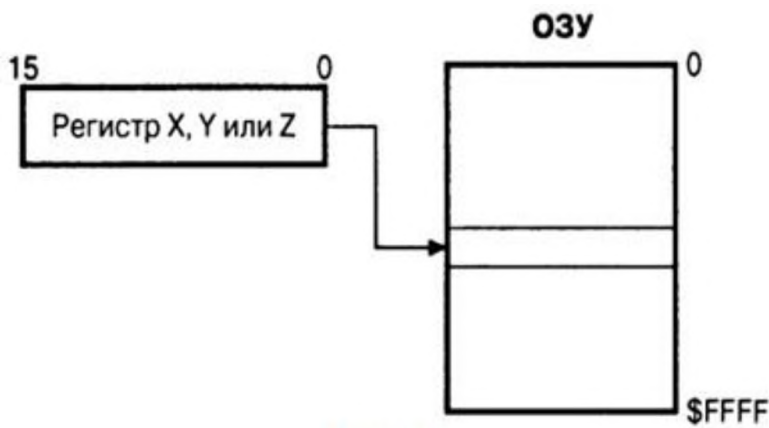


Рис. 1.1.

- А) простая косвенная адресация; +
- Б) прямая адресация одного регистра общего назначения;
- В) прямая адресация трёх регистров общего назначения;
- Г) прямая адресация ОЗУ;
- Д) относительная косвенная адресация;

15. Укажите, какой способ адресации изображён на рисунке (см. рис. 1.2.):

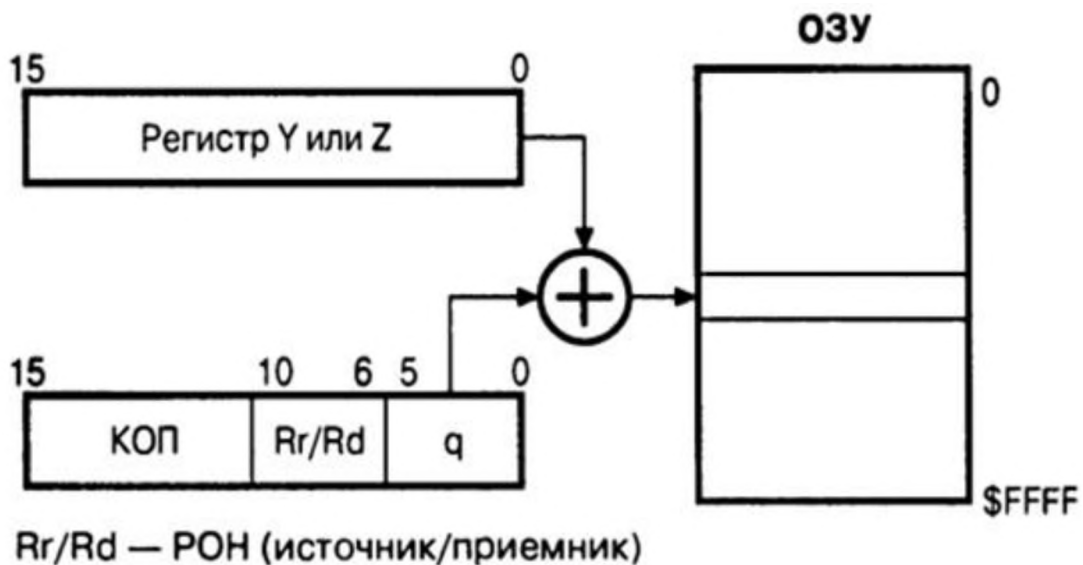


Рис. 1.2.

- А) относительная косвенная адресация; +
- Б) простая косвенная адресация;
- В) прямая адресация одного регистра общего назначения;
- Г) прямая адресация трёх регистров общего назначения;
- Д) прямая адресация ОЗУ;

16. Для работы с EEPROM-памятью используются регистры:

- А) EEAR; +
- б) EEDR; +

- в) EECR; +
- г) EEIR;
- д) EEPR;

17. Процедура записи одного байта в EEPROM-память состоит из следующих этапов (Укажите последовательность действий, см. рис. 1.3.:

- I) установить в 1 флаг EEMWE (EEMPE) регистра EECR;
- II) в течение 4 тактов после установки флага EEMWE (EEMPE) записать в бит EEWE (EEPE) регистра EECR лог. 1;
- III) дождаться завершения записи во FLASH-память программ;
- IV) необходимо дождаться пока не сбросится флаг EEWE (EEPE) регистра EECR;
- V) загрузить байт данных в регистр EEDR, а требуемый адрес — в регистр EEAR (при необходимости);

Рис. 1.3.

- А) II-III-I-V-IV;
- Б) I-V-IV-III-II;
- В) IV-III-V-I-II; +
- Г) III-I-V-IV-II;
- Д) V-I-III-IV-II;
- Е) I-II-III-IV-V;
- Ж) II-IV-V-III-I;
- З) IV-V-I-II-III;

18. Для предотвращения проблем, которые могут возникнуть при записи данных в EEPROM рекомендуется:

- А) запрещать все прерывания при выполнении записи в EEPROM; +
- Б) запрещать все прерывания при выполнении чтения из EEPROM;
- В) удерживать микроконтроллер в «спящем» режиме пока производится запись;

19. Счётчик команд – это:

- А) регистр, в котором содержится адрес следующей исполняемой команды; +
- Б) регистр, в котором содержится количество выполненных команд программы;
- В) регистр, в котором содержится общее количество команд программы;
- Г) регистр, в котором содержится общее количество команд условного перехода в программе;

20. Если в команде условного перехода под значение смещения отводится семь битов, то максимальная величина перехода составляет:

- А) -63... +64 слова; +
- Б) -126... + 127 байт;
- В) -254... +254 байт;
- Д) -7... + 7 байт;
- Е) -3... +3 слова;

21. Если под значение операнда в слове команды относительного перехода RJMP отводится 11 битов, то максимальная величина перехода составляет:

- А) -2047... +2048 слов; +
- Б) -254... +254 слов;
- В) только +2048 слов;
- Г) только +254 слова;
- Д) -2047... +2048 байт;

22. При косвенном переходе IJMP в качестве адреса перехода используется содержимое:

- А) индексного регистра Z; +
- б) индексного регистра X;
- в) индексного регистра Y;
- г) одного из регистров ввода-вывода;
- д) одного из дополнительных регистров ввода-вывода;

23. При косвенном вызове подпрограммы в счётчик команд загружается:

- А) содержимое индексного регистра Z; +
- б) содержимое индексного регистра X;
- в) содержимое индексного регистра Y;
- г) содержимое одного из регистров ввода-вывода;
- д) содержимое одного из дополнительных регистров ввода-вывода;

24. Стек в микроконтроллерах семейства mega размещается в:

- А) памяти данных; +
- Б) памяти программ;
- В) ОЗУ;

25. Внутренний нагрузочный резистор, подключённый к выводу порта микроконтроллера:

- А) создаёт вытекающий ток для внешних устройств, подключённых между выводом порта и общим проводом; +
- Б) создаёт вытекающий ток на выводе порта;

В) уменьшает напряжение на выводе порта;

Г) увеличивает напряжение на выводе порта;

26. Для того, чтобы подключить к выводу PA0 порта А внутренний нагрузочный резистор необходимо установить значения:

А) DDA0=0; PA0=1; +

б) DDA0=1; PA0=1;

в) DDA0=1; PA0=0;

г) DDA0=0; PA0=0;

27. Пусть имеется устройство для управления светодиодом при помощи кнопки. При нажатии кнопки – светодиод горит, иначе – нет. Схема устройства приведена на рис. 2.1. Выберите правильные утверждения:

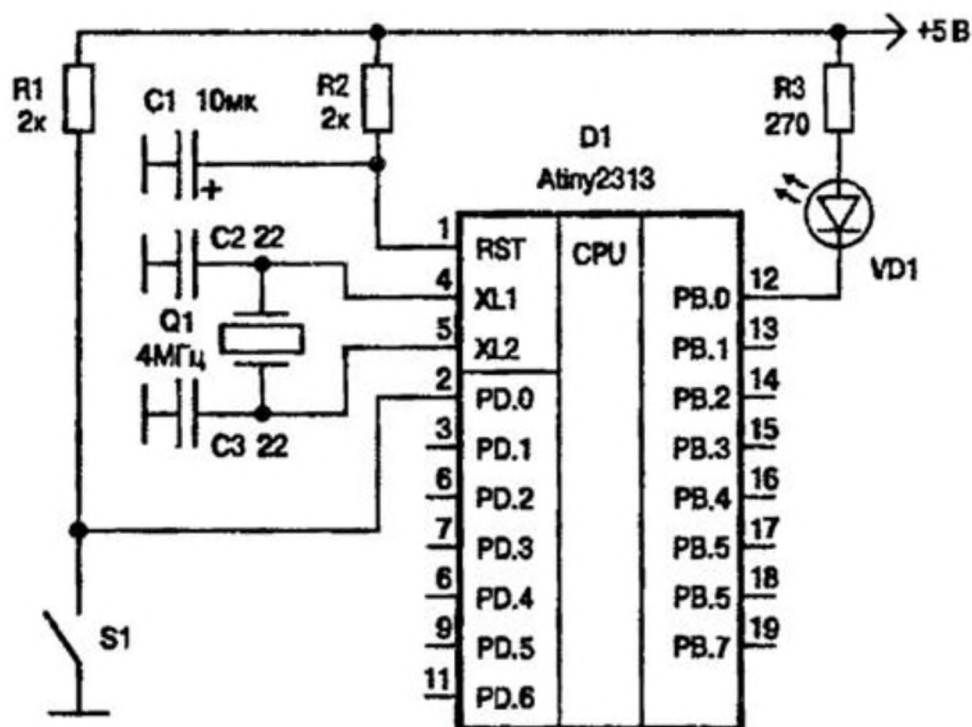


Рис. 2.1.

А) когда кнопка S1 разомкнута, на выводе PD0 будет сигнал, соответствующий логической единице; +

Б) когда кнопка S1 разомкнута, на выводе PD0 будет сигнал, соответствующий логическому нулю;

В) когда на выводе PD0 имеется сигнал логической единицы, то на выводе PB0 должен быть сигнал высокого логического уровня; +

Г) когда на выводе PD0 имеется сигнал логической единицы, то на выводе PB0 должен быть сигнал низкого логического уровня;

Д) когда на выводе PD0 имеется сигнал логического нуля, то на выводе PB0 должен быть сигнал низкого логического уровня; +

Е) когда на выводе PD0 имеется сигнал логического нуля, то на выводе PB0 должен быть сигнал высокого логического уровня;

28. Выберите правильные утверждения:

- А) чем меньше адрес прерывания в таблице прерываний, тем выше приоритет прерывания; +
- Б) чем больше адрес прерывания в таблице прерываний, тем выше приоритет прерывания;
- В) чем меньше адрес прерывания в таблице прерываний, тем меньше приоритет прерывания;
- Г) чем больше адрес прерывания в таблице прерываний, тем больше приоритет прерывания;

29. Для того, чтобы разрешить или запретить отдельные прерывания используются регистры:

- А) GIMSK; +
- б) TIMSK; +
- в) GIFR;
- г) TIFR;
- д) SREG;

30. Алгоритм работы системы прерываний сводится к следующему. (Установите правильно последовательность действий, см. рис. 2.2.:

- I) при поступлении запроса на прерывание устанавливается флаг соответствующего прерывания;
- II) для включения прерывания программа должна установить флаг I регистра SREG в единицу и записать в регистры маски такой код, который разрешит лишь нужные в данный момент прерывания;
- III) после окончания обработки очередного прерывания происходит проверка остальных флагов, и если имеется хоть одно не обработанное прерывание МК переходит к его обработке;
- IV) флаг I автоматически сбрасывается, запрещая обработку других прерываний. Флаг, соответствующий вызванному прерыванию, также сбрасывается, сигнализируя о том, что МК уже приступил к его обработке;

Рис. 2.2.

- А) I-III-IV-II;
- Б) III-II-IV-I;
- В) II-IV-III-I;
- Г) IV-III-II-I;
- Д) III-IV-I-II;
- Е) II-I-IV-III; +
- Ж) I-II-IV-III;

31. Регистры восьмибитных таймеров-счётчиков делятся на:

- А) регистры управления; +
- Б) счётные регистры; +
- В) регистры сравнения; +
- Г) регистры состояния асинхронного режима; +
- Д) регистры флагов;

Е) надо подумать....

32. При работе восьмибитных таймеров-счётчиков в нормальном режиме:

А) производится непрерывное сравнение содержимого регистра сравнения OCRn с содержимым счётного регистра TCNTn. В случае равенства содержимого этих регистров устанавливается флаг OCFn в регистре флагов и генерируется прерывание; +

Б) производится непрерывное сравнение содержимого регистра сравнения OCFn с содержимым регистра сравнения TCNTn. В случае равенства содержимого этих регистров устанавливается флаг OCRn в регистре флагов и генерируется прерывание;

В) производится непрерывное сравнение содержимого регистра сравнения OCFn с содержимым регистра сравнения TCNTn. В случае не равенства содержимого этих регистров устанавливается флаг OCRn в регистре флагов и генерируется прерывание;

33. Режимы работы восьмибитных таймеров-счётчиков T0, T2 определяются:

А) состоянием битов WGMn2:WGMn0 регистра управления TCCRn; +

Б) состоянием битов COMnA1:COMnA0 счётного регистра TCNTn;

в) состоянием битов COMnA1:COMnA0 регистра сравнения OCRn;

г)) состоянием битов FOCnA:FOCnB регистра управления TCCRn;

34. При работе восьмибитного таймера-счётчика в в режиме CTC:

А) максимально возможное значение счётного регистра определяется регистром сравнения OCRn; +

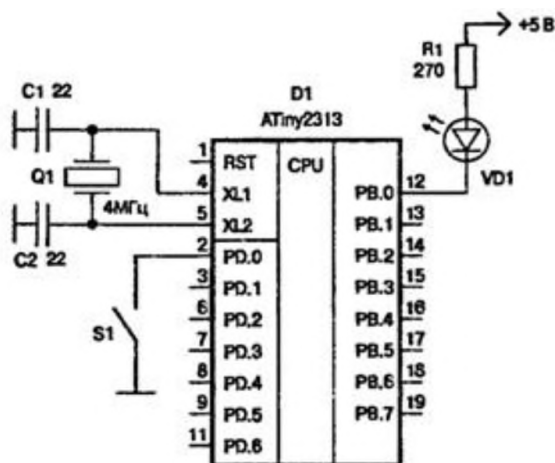
Б) при обнулении счётного регистра устанавливается флаг прерывания по переполнению TOVn; +

В) при достижении счётчиком максимального значения устанавливается флаг OCFn; +

Г) записываемое в регистр сравнения OCRn максимальное значение сохраняется дополнительно в буферном регистре;

Д) при обнулении счётного регистра изменяется состояние выводов OCn;

35. Дана схема устройства и программный код на рис. Укажите, что будет происходить при нажатии кнопки (см. рис. 2.3.):



```

void main(void)
{
    PORTB=0xFF; // Инициализация
    DDRB=0xFF;

    PORTD=0x7F; // Инициализация
    DDRD=0x00;

    ACSR=0x80; // Инициализация
    while (1)
    {
        if (PIND.0==1)
        { PORTB.0=1; } //
        else
        {
            PORTB.0=1;
            delay_ms(200);
            PORTB.0=0;
            delay_ms(200);
        }
    }
};

```

Рис. 2.3.

А) если PD0=0, то светодиод будет мигать с задержкой в 200 мс. Мигание начнётся с того, что светодиод не будет гореть; +

Б) если PD0=0, то светодиод будет мигать с задержкой в 200 мс. Мигание начнётся с того, что светодиод будет гореть;

В) если PD0=1, то светодиод будет мигать с задержкой в 200 мс. Мигание начнётся с того, что светодиод не будет гореть;

Г) если PD0=1, то светодиод будет мигать с задержкой в 200 мс. Мигание начнётся с того, что светодиод будет гореть;

Д) если PD0=1, то светодиод будет потушен, то есть на выводе PB0 будет сигнал высокого логического уровня; +

Е) если PD0=1, то светодиод будет потушен, то есть на выводе PB0 будет сигнал низкого логического уровня;

36. Сторожевой таймер в микроконтроллерах семейства mega необходим для:

А) защиты микроконтроллера от сбоев в процессе работы; +

Б) для генерирования прерывания в случае совпадения содержимого счётного регистра с содержимым сторожевого таймера;

В) для генерирования прерывания в случае совпадения содержимого регистра сравнения с содержимым сторожевого таймера;

37. Аналоговый компаратор предназначен для:

- А) сравнения значений напряжения, которое присутствует на двух выводах микроконтроллера и генерировании в данном случае прерывания; +
- Б) защиты вывода микроконтроллера в случае, если на вывод будет подан сигнал больше 5 В;
- В) управления схемой захвата таймера-счётчика; +
- Г) преобразования аналогового сигнала на выводе микроконтроллера в цифровой сигнал;

38. Модуль SPI в микроконтроллерах AVR семейства mega использует следующие выводы:

- А) SCK; +
- Б) MISO; +
- В) MOSI; +
- Г) SS; +
- Д) SPCR;
- Е) SPE;

39. Последовательный интерфейс TWI:

- А) имеет линию синхронизации SCL; +
- б) имеет линию данных SDA; +
- в) имеет линию команд SDC;
- г) имеет линию адреса SDA;
- д) может связать до 128 различных устройств; +
- е) может связать до 256 различных устройств;

40. При передаче данных по шине TWI:

- А) каждый передаваемый бит сопровождается импульсом на линии тактового сигнала SCL; +
- Б) сигнал на линии SDA должен быть стабильным в течение всего времени, пока на шине SCL присутствует сигнал лог. 1; +
- В) сигнал на линии SDA должен быть стабильным в течение всего времени, пока на шине SCL присутствует сигнал лог. 0;
- Г) каждый передаваемый бит сопровождается сигналом высокого логического уровня на линии тактового сигнала SCL;

41. Чтобы сформировать состояние СТАРТ на шине TWI необходимо:

- А) изменить уровень сигнала на линии SDA с 1 на 0 при ВЫСОКОМ уровне сигнала на линии SCL; +
- Б) изменить уровень сигнала на линии SDA с 0 на 1 при ВЫСОКОМ уровне сигнала на линии SCL;
- В) изменить уровень сигнала на линии SDA с 0 на 1 при НИЗКОМ уровне сигнала на линии SCL;
- Г) изменить уровень сигнала на линии SDA с 1 на 0 при НИЗКОМ уровне сигнала на линии SCL;

42. Адресный пакет, передаваемый по шине TWI содержит:

- А) в общем 9 бит; +
- Б) в общем 10 бит;
- В) 7-ми битный адрес; +
- Г) 8-ми битный адрес;
- Д) один управляющий бит R/W; +
- Е) два управляющих бита R/W;
- Ж) бит квитирования ACK; +
- З) бит сравнения OCR;

43. Регистр TWAR модуля TWI:

- А) в старших семи битах содержит адрес ведомого устройства; +
- Б) маскирует значения отдельных битов данных, передаваемых по линии SDA;
- В) управляет всем модулем TWI;
- Г) предназначен для генерации прерываний при наступлении состояний СТАРТ(ПОВСТАРТ)/СТОП;

44. Укажите последовательность действий, которые выполняются при приеме одного байта данных ведомым устройством по шине TWI (см. рис. 3.1.:

- I) передается бит подтверждения путем установки $TWEN = 1$ и выполняется сброс флага $TWINT$;
- II) в регистр $TWCR$ выводится команда, сбрасывающая флаг $TWINT$;
- III) проверяется код статуса в регистре $TWSR$ (если он равен $S80$, значит в регистр $TWDR$ принят байт данных); передается бит подтверждения ($TWEN = 1$) и выполняется сброс флага $TWINT$;
- IV) после приема первого байта с адресом ведомого устройства формируется запрос на прерывание, проверяется код статуса в регистре $TWSR$; если он равен $S60$, прием собственного адреса выполнен успешно;
- V) при поступлении состояния Stop в регистре $TWSR$ формируется код статуса $SA0$ (конец пакета);

Рис. 3.1.

- а) I-III-V-IV-II;
- б) II-IV-I-III-V; +
- в) III-I-V-IV-II;
- г) IV-I-V-III-II;
- д) V-II-III-I-IV;
- е) I-V-IV-II-III;
- ж) III-IV-II-I-V;
- з) V-II-IV-III-I;

45. Для управления модулем USART в микроконтроллерах AVR семейства mega используются следующие регистры:

- А) UCSRA; +
- б) UCSRB; +
- в) UCSRC; +
- г) UCSOCRA;
- д) UCSOCRB;
- е) UCSOCRC;

46. При передаче данных с помощью модуля USART:

- А) данные записываются в регистр данных передатчика UDR; +
- Б) данные пересылаются из регистра UDR (UDRn) в сдвиговый регистр передатчика; +
- В) данные записываются в сдвиговый регистр UDR передатчика;
- Г) данные пересылаются из сдвигового регистра в регистр данных передатчика UDR (UDRn);

47. Допустим имеется проект z1, созданный в CVAVR. В папке проекта содержатся файлы z1.hex, z1__.c, z1.cof. Выберите правильные утверждения:

- А) z1.hex - файл-прошивка для "загрузки" в МК; +
- Б) z1__.c - копия файла z1.c для симуляторов; +
- В) z1.cof - информация связывающая содержимое файлов z1__.c и z1.hex; +
- Г) z1__.c - информация связывающая содержимое файлов z1.cof и z1.hex;
- Д) z1.cof - файл-прошивка для "загрузки" в МК;
- Е) z1.hex - копия файла z1.c для симуляторов;
- Ж) z1__.c - файл-прошивка для "загрузки" в МК;
- З) z1.cof - копия файла z1.c для симуляторов;
- И) z1.hex - информация связывающая содержимое файлов z1.cof и z1__.c;

48. Каждому порту в МК AVR соответствуют следующие регистры:

- А) DDRx - регистр направления работы - вход или выход; +
- Б) PINx - регистр содержит значения физических уровней сигнала на соответствующих ножках МК; +
- В) PORTx - регистр в который записываются желаемые значения "1" или "0" и которые необходимо получить на соответствующих ножках МК при назначении их выходом; +
- Г) PORTx - регистр содержит значения физических уровней сигнала на соответствующих ножках МК;
- Д) DDRx - регистр в который записываются желаемые значения "1" или "0" и которые необходимо получить на соответствующих ножках МК при назначении их выходом;
- Е) PINx - регистр направления работы - вход или выход;
- Ж) TIMERx – счётный регистр вывода;
- З) TIMERx – регистр управления таймера-счётчика;

49. Выберите правильные утверждения, применительно к программному коду (см. рис. 3.2.:

```
while (1) {  
    PORTA++;  
    while (!(TIFR&0x01));  
    TIFR = 0x01;  
};
```

Рис. 3.2.

- А) в цикле while (!(TIFR&0x01.) устанавливается событие «переполнение таймера»; +
- Б) цикл while (1. – бесконечный цикл; +
- В) цикл while (1. закончит свою работу, когда в регистре TIFR будет установлена 1;
- Г) цикл while (!(TIFR&0x01.) будет выполняться до тех пор, пока результат вычисления выражения TIFR&0x01 будет равен 0; +
- Д) цикл while (!(TIFR&0x01.) будет выполняться до тех пор, пока результат вычисления выражения TIFR&0x01 будет равен 1;

50. Запись (см. рис. 3.3. означает, что:

```
Interrupt [TIM1_COMP] void timer1_comp_isr(void)
```

Рис. 3.3.

- А) что функция timer1_comp_isr является процедурой обработки прерывания; +
- Б) что процедура Interrupt является функцией обработки прерывания;
- В) выражение Interrupt [TIM1_COMP] означает, что данная функция является процедурой обработки прерывания по совпадению таймера T1; +
- Г) выражение Interrupt [TIM1_COMP] означает, что данная функция является процедурой обработки прерывания по совпадению таймера COMP;

51. Запись gab>>1 означает:

- А) сдвиг разрядов переменной gab на 1 вправо; +
- Б) сдвиг разрядов переменной gab на 1 влево;
- В) сравнение последнего разряда переменной gab с 1;
- Г) сравнение первого разряда переменной gab с 1;

52. Интерфейс SPI расшифровывается как:

- А) Serial Peripheral Interface; +
- Б) Serial Programming Interface;
- в) Synchronous Programming Interface;
- г) Synchronous Peripheral Interface;
- д) System Peripheral Interface;
- е) System Programming Interface;

53. Недостатком параллельного подключения к шине SPI является:

- А) необходимость в дополнительных линиях для адресации подчиненных микросхем; +
- Б) то, что не всегда данное подключение возможно, так как не все микросхемы SPI-совместимы;
- В) то, что выход передачи данных одной микросхемы соединяется со входом приема данных другой, что в свою очередь ведёт к невозможности создания полнодуплексной передачи данных;
- Г) то, что для синхронизации двух микросхем при передаче данных используется 4 такта тактового генератора ведущего МК, что приводит к снижению скорости передачи данных;
54. При параллельном подключении к шине SPI 7 ведомых микросхем, общее число линий связи будет равно:
- А) 10; +
- Б) 7;
- В) 8;
- Г) 4;
- Д) 5;
55. Интерфейс I2C характеризуется тем, что:
- А) количество линий связи не зависит от количества подключённых к нему микросхем; +
- Б) имеет возможность мультимастерной работы; +
- В) имеет 2 программируемые скорости обмена данными;
- Г) использует 2 вывода микроконтроллера;
56. Если в названии микроконтроллера написано следующее MC 68HLC 7 05 C9A M FU, то это значит:
- А) что это МК с пониженным напряжением питания $V_{п}=2В$; +
- Б) что этот МК был произведён как опытный образец;
- В) что этот МК имеет электрически программируемое ПЗУ; +
- Г) что этот МК имеет диапазон рабочих температур $-40... +125\text{ C}$; +
- Д) что этот МК имеет масочное ПЗУ;
57. В МК фирмы Motorola модуль асинхронной приемо-передачи называется:
- А) SCI; +
- Б) SPI;
- в) UART;
- г) USART;
- д) RxD;
- е) TxD;
58. Модуль асинхронной приемо-передачи в МК Motorola обеспечивает приемо-передачу:
- А) 10-ти битовых кадров, если бит M=0 в регистре SCCR1; +
- Б) 11-ти битовых кадров, если бит M=1 в регистре SCCR1;

- В) с использованием бита четности, если бит M=1 в регистре SCCR1; +
- Г) с использованием бита четности, если бит M=0 в регистре SCCR1;
- Д) с использованием бита четности, если бит T8=0 в регистре SCCR1;
- Е) без использования бита четности, если бит T8=0 в регистре SCCR1;

59. Модуль АЦП может работать в следующих режимах:

- А) одиночного преобразования; +
- Б) непрерывного преобразования; +
- В) дискретного преобразования;
- Г) сброс при совпадении;
- Д) захват при совпадении;

60. Чтобы сконфигурировать вывод PD2 как выход применяют следующую конструкцию:

- А) $DDRD |= 1 \ll 2$; +
- Б) $DDRD = 1 \ll 2$;
- В) $DDRD |= 1 \gg 2$;
- Г) $DDRD |= 1 \ll 1$;
- Д) $DDRD = 1 \ll 1$;
- Е) $DDRD |= 1 \ll 2$;
- Ж) $DDRD \&= \sim(1 \ll 2)$;
- З) $DDRD \&= (1 \ll 2)$;
- И) $DDRD \&= \sim(1 \gg 2)$;

61. Чтобы сконфигурировать вывод PD2 как вход применяют следующую конструкцию:

- А) $DDRD |= 1 \ll 2$;
- Б) $DDRD = 1 \ll 2$;
- В) $DDRD |= 1 \gg 2$;
- Г) $DDRD |= 1 \ll 1$;
- Д) $DDRD = 1 \ll 1$;
- Е) $DDRD |= 1 \ll 2$;
- Ж) $DDRD \&= \sim(1 \ll 2)$; +
- З) $DDRD \&= (1 \ll 2)$;
- И) $DDRD \&= \sim(1 \gg 2)$;
- Ж) $DDRB \&= \sim(1 \ll 2)$;
- З) $DDRB \&= \sim(1 \gg 2)$;

62. Дана схема подключения светодиода (см. рис. 1.1.. Укажите сопротивление токоограничительного резистора, включаемого последовательно со светодиодом (ножка,

светодиод, резистор, +5 вольт питания МК), которое бы обеспечивало протекание тока через светодиод величиной в 20 мА.

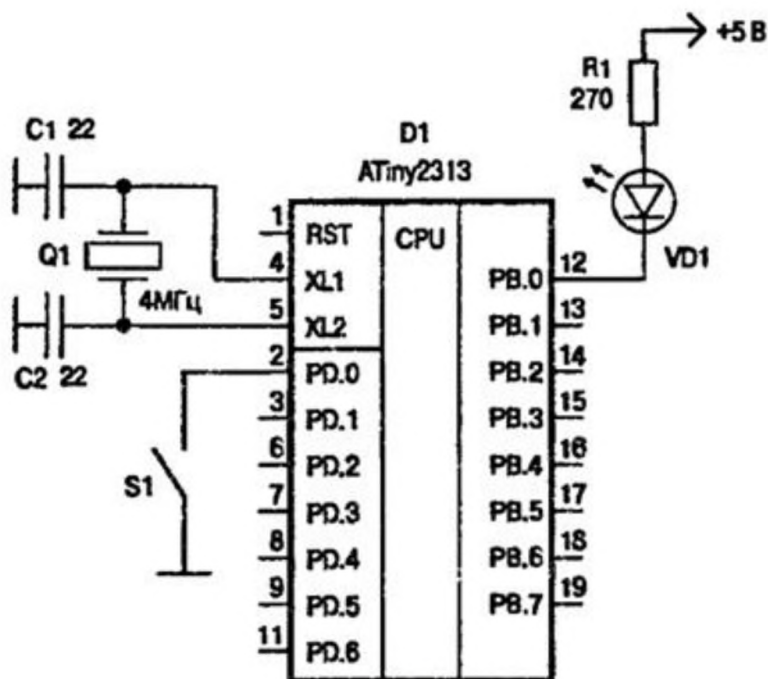


Рис. 1.1.

- А) приблизительно 170 Ом; +
 - Б) приблизительно 180 Ом;
 - В) приблизительно 150 Ом;
 - Г) приблизительно 160 Ом;
63. Регистр управления сторожевым таймером называется:
- А) WDTCR; +
 - б) WDCR;
 - в) WDT;
 - г) WDTOI;
 - д) WDTCOI;
64. Для процессоров с RISC-архитектурой характерно, что:
- А) все команды имеют формат фиксированной длины; +
 - Б) выборка команды из памяти и ее исполнение осуществляется за один цикл; +
 - В) система команд RISC-процессора имеет возможность равноправно использовать всех регистры процессора; +
 - Г) для хранения программ и данных используется общая память;
 - Д) для хранения программ и данных используются отдельные адресные пространства;
 - Е) режим ШИМ легче программируем за счёт плавающей длины команд;
65. Укажите какой тип памяти использует МК, изображённый на рис. 1.2:



Рис. 1.2.

- А) ПЗУ масочного типа; Б) Электрически программируемые ПЗУ с плавкими перемычками;
- В) Перепрограммируемые ПЗУ с ультрафиолетовым стиранием; +
- Г) Однократно программируемые ПЗУ;
- Д) Электрически стираемые перепрограммируемые ПЗУ;
- Е) ПЗУ с электрическим стиранием типа Flash;

66. Arduino – это:

- А) аппаратная вычислительная платформа для МК Motorola, основными компонентами которой являются простая плата ввода/вывода и среда разработки; +
- Б) семейство МК фирмы Atmel;
- в) семейство МК фирмы Motorola;
- г) семейство МК фирмы PIC;
- д) семейство МК фирмы MicroChip;
- е) аппаратная вычислительная платформа, основными компонентами которой являются простая плата ввода/вывода и среда разработки;

67. Выберите правильные утверждения:

- А) Arduino Diecimila использует USB-интерфейс; +
- Б) Arduino Diecimila использует ATmega168 в DIP28 корпусе; +
- В) Arduino Diecimila использует ATmega1280 в DIP28 корпусе;
- Г) Arduino Diecimila использует ATmega8 в DIP28 корпусе;
- Д) Arduino Diecimila программируется через разъём DB9;
- Е) Arduino Diecimila использует ATmega8;

68. Триггер Шмитта:

- А) используется во входных буферах на всех выводах МК AVR; +
- Б) используется для программирования выводов как на вход, так и на выход;
- В) преобразует входной сигнал произвольной формы в сигнал, принимающий два стандартных уровня "0" и "1"; +
- Г) статическая характеристика триггера Шмитта никогда не имеет петлю гистерезиса;
- Д) статическая характеристика триггера Шмитта имеет петлю гистерезиса; +
- Е) преобразует дискретный входной сигнал "0" и "1" в непрерывный;

69. На рис. 1.3 представлен МК в корпусе:

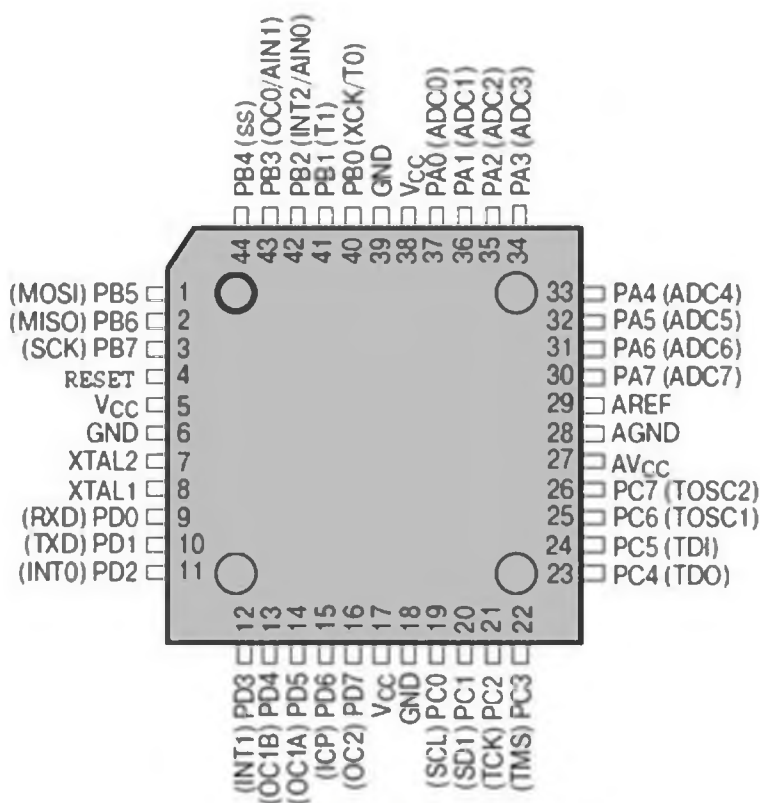


Рис. 1.3.

- А) TQFP; +
 - Б) PDIP;
 - В) SOIC;
 - Г) PLCC;
 - д) DIP;
70. Память данных включает в себя:
- А) 32 регистра общего назначения; +
 - Б) 64 регистра ввода-вывода; +
 - В) 160 доп. Регистров ввода-вывода; +
 - Г) внутренне статическое ОЗУ; +

Д) ROM;

е) FLASH – память;

71. Регистровая память МК AVR включает:

А) 32 регистра общего назначения; +

Б) служебные регистры ввода/вывода; +

В) дополнительные регистры ввода/вывода; +

Г) системные регистры ввода-вывода;

Д) регистры для подключения внешней памяти ОЗУ;

Е) 64 регистра общего назначения;

72. Флаг общего разрешения прерывания I находится в регистре:

А) SREG; +

б) TIFR;

в) GPIOR;

г) EEAR;

д) EEDR;

Е) EECR;

73. Команда sbis ioreg,bit:

А) проверяет бит в регистре ввода/вывода и пропускает следующую команду, если бит установлен; +

Б) проверяет бит в регистре ввода/вывода и пропускает следующую команду, если бит сброшен;

В) устанавливает в I заданный бит регистра ввода/вывода ioreg;

Г) пересылает содержимое регистра ввода/вывода ioreg в рабочий регистр reg;

Д) пересылает содержимое рабочего регистра ioreg в регистр ввода/вывода ioreg;

Е) записывает данные из регистра ioreg в регистр данных EEPROM;

74. Определите, сколько контактов ввода-вывода МК необходимо задействовать в схеме, которая использует стробирование, для обслуживания массива из 30 кнопок:

А) 11; +

Б) 12;

В) 20;

Г) 17;

Д) 30;

Е) 31;

Ж) 25;

75. Для управления размещением таблицы прерываний в МК AVR используются:

- А) бит IVSEL; +
- Б) бит IVCE; +
- В) регистр IVSEL;
- Г) регистр IVCE;
- Д) регистр SREG;

76. 16-битные таймеры-счётчики микроконтроллеров Atmel могут работать в следующих режимах:

- А) в нормальном режиме; +
- Б) в режиме ШИМ с точной фазой; +
- В) в режиме Сброс при совпадении; +
- Г) в режиме Быстродействующий ШИМ; +
- Д) в режиме ШИМ с точной фазой и частотой; +
- Е) в режиме ШИМ с точной фазой и амплитудой;
- Ж) в режиме Сброс при несовпадении;

77. При работе таймера-счётчика в режиме Сброс при совпадении:

- А) чем больше значение в регистре сравнения OCRnA, тем больше частота выходного сигнала на выводе OCnA; +
- Б) чем меньше значение в регистре сравнения OCRnA, тем больше частота выходного сигнала на выводе OCnA;
- В) чем больше значение в регистре сравнения ICRnA, тем больше частота выходного сигнала на выводе OCnA;
- Г) выводом OCn можно управлять с помощью бит COMn1 и COMn0 регистра управления; +
- Д) выводом OCn можно управлять с помощью бит WGMn1 и WGMn0 регистра управления;

78. Режим «Быстродействующий ШИМ» для 16-битных таймеров-счётчиков отличается от режима «Быстродействующий ШИМ» для 8-битных таймеров-счётчиков тем, что:

- А) при работе с 16-битным таймером-счётчиков в режиме «Быстродействующий ШИМ» можно изменять разрешающую способность модулятора; +
- Б) при работе с 16-битным таймером-счётчиков в режиме «Быстродействующий ШИМ» можно использовать регистр захвата; +
- В) при работе с 16-битным таймером-счётчиков в режиме «Быстродействующий ШИМ» можно использовать регистр сравнения;
- Г) при работе с 16-битным таймером-счётчиков в режиме «Быстродействующий ШИМ» можно настроить работу счётного регистра так, чтобы он работал как реверсивный счётчик;

79. Режим «ШИМ с точной фазой» для 16-битных таймеров-счётчиков отличается от режима «ШИМ с точной фазой» для 8-битных таймеров-счётчиков тем, что:

- А) при работе с 16-битным таймером-счётчиков в режиме «ШИМ с точной фазой» можно изменять разрешающую способность модулятора; +

Б) при работе с 16-битным таймером-счётчиков в режиме «ШИМ с точной фазой» можно использовать регистр захвата; +

В) при работе с 16-битным таймером-счётчиков в режиме «ШИМ с точной фазой» можно использовать регистр сравнения;

Г) при работе с 16-битным таймером-счётчиков в режиме «ШИМ с точной фазой» можно настроить работу счётного регистра так, чтобы он работал как реверсивный счётчик;

Д) при работе с 16-битным таймером-счётчиков в режиме «ШИМ с точной фазой» обновление содержимого регистра сравнения происходит в момент достижения счетчиком максимального значения;

Е) при работе с 16-битным таймером-счётчиков в режиме «ШИМ с точной фазой» обновление содержимого регистра сравнения происходит в момент достижения счетчиком минимального значения;

80. Работа 16-битного таймера-счётчика в режиме «ШИМ с точной фазой и частотой» характеризуется следующим:

А) обновление содержимого регистра сравнения происходит в момент достижения счетчиком минимального значения; +

Б) можно изменять разрешающую способность модулятора; +

В) можно использовать регистр захвата; +

Г) можно использовать регистр сравнения; +

Д) можно генерировать симметричные выходные импульсы в одном периоде; +

Е) можно генерировать ассиметричные выходные импульсы в одном периоде;

Ж) нельзя изменять разрешающую способность модулятора;

З) обновление содержимого регистра сравнения происходит в момент достижения счетчиком максимального значения;

81. Для МК ATmega16 функция вывода SS зависит от:

А) разряда DDRB.4; +

Б) разряда PIN.4;

В) разряда PORTB.4;

Г) разряда DDRB.6;

Д) разряда PIN.6;

Е) разряда PORTB.6;

82. Если модуль SPI работает в режиме ведомого, то:

А) вывод SS работает на вход; +

Б) вывод SS работает на выход;

В) если на вывод SS подан низкий логический уровень сигнала - SPI активируется; +

Г) если на вывод SS подан высокий логический уровень сигнала - SPI активируется;

Д) вывод MISO является выходом; +

Е) вывод MISO является входом;

83. Интерфейс I2C характеризуется следующим:

А) использует последовательную линию данных SDA; +

Б) использует последовательную линию тактирования SCL; +

В) использует 10-битную адресацию; +

Г) использует последовательную линию SS – выбор ведомого;

Д) при передаче данных использует бит подтверждения; +

Е) независимо от подключённых микросхем шина I2C остаётся трёхпроводной;

Ж) независимо от подключённых микросхем шина I2C остаётся двухпроводной; +

84. Адресные пакеты, передаваемые по шине TWI имеют:

А) 7-битный адрес; +

Б) управляющий бит R/W; +

В) бит подтверждения ACK; +

Г) 6-битный адрес;

Д) бит DORD, указывающий направление сдвига при передаче данных;

Е) 5-битный адрес;

85. Формирование запроса на прерывание при работе с модулем TWI осуществляется при возникновении следующих событий:

А) окончание формирования состояния СТАРТ/ПОВСТАРТ; +

Б) окончание передачи адресного пакета (SLA+R/W); +

в) окончание передачи байта адреса; +

г) потеря устройством приоритета; +

д) адресация устройства или наличие общего вызова; +

ж) окончание приема байта данных; +

з) возникновение ошибок на шине, обусловленных недопустимыми условиями формирования состояний СТАРТ/СТОП; +

и) переполнение регистра данных TWDR;

к) изменение содержимого регистра TWAR;

86. Пакеты данных, передаваемые по шине TWI имеют:

А) 8 бит данных; +

Б) управляющий бит R/W;

В) бит подтверждения ACK; +

Г) 7-бит данных;

Д) бит DORD, указывающий направление сдвига при передаче данных;

Е) 5-бит данных;

87. При широтно-импульсной модуляции:

А) изменяющийся аналоговый сигнал получают посредством цифровых устройств; +

Б) ширина импульса выходного цифрового сигнала пропорциональна амплитуде аналогового сигнала; +

В) изменяющийся цифровой сигнал получают посредством аналоговых устройств;

Г) амплитуда импульса выходного цифрового сигнала пропорциональна частоте аналогового сигнала;

Д) происходит приближение желаемого сигнала (многоуровневого или непрерывного) к действительным бинарным сигналам; +

Ж) площадь фигуры, образованной аналоговым сигналом равна площади фигуры, образованной цифровым сигналом; +

88. Система автоматизированного проектирования Proteus состоит из следующих частей:

А) ISIS; +

Б) ARES; +

В) Designer;

Г) Modeler;

Д) Sprint LayOut;

89. Система автоматизированного проектирования Altium Designer состоит из следующих частей:

А) Altium Designer Custom Board Front-End Design; +

Б) Altium Designer Custom Board Implementation; +

В) Altium Designer Custom Board Modeler;

Г) Altium Designer Custom Board Sprint-Layout Arduino;

Д) Altium Designer Custom Board Sprint-Layout;

90. Arduino представляет собой:

А) линейку электронных блоков-плат, которые можно подключать к компьютеру по USB, а в качестве периферии — любые устройства; +

Б) аппаратно-вычислительную платформу, основными компонентами которой являются простая плата ввода/вывода и среда разработки на языке Wiring; +

В) простой программатор для МК ATMEL, поддерживаемый программой avrdude;

Г) простой программатор для МК ATMEL, поддерживаемый программой AVRStudio;

Д) простую программу для разработки как односторонних, так и двухсторонних печатных плат;

Е) среда разработки принципиальных схем;

91. Зона неоднозначности, которую обеспечивает триггер Шмитта на всех выводах микроконтроллера:

А) описывается петлёй гистерезиса; +

- Б) необходима для ликвидации внешних помех на выводах МК; +
- В) определяет то, что если напряжение на ножке выше, чем 60% напряжения питания МК, то сигнал на ножке воспринимается как ВЛУ или "1"; +
- Г) определяет то, что если напряжение на ножке ниже, чем 20% напряжения питания МК, то сигнал на ножке воспринимается как НЛУ или "0";
- Д) определяет то, что если напряжение на ножке выше, чем 80% напряжения питания МК, то сигнал на ножке воспринимается как ВЛУ или "1"; +
- Е) указывает на то, что любое изменение напряжения на ножке МК лежащее в зоне неоднозначности не ведет к изменению того, каким логическим уровнем считает МК напряжение на этой ножке в данный момент; +
92. Тактовый генератор микроконтроллеров семейства Mega может работать:
- А) с внешним кварцевым/керамическим резонатором; +
- Б) с внешней или внутренней RC-цепочкой; +
- В) с внешним сигналом синхронизации; +
- Г) надо подумать;
- Д) с внешним сигналом для FLASH-памяти программ;
93. Выбор режима работы тактового генератора осуществляется:
- А) программированием конфигурационных ячеек; +
- Б) установкой FUSE Bits CKSEL3...0; +
- В) установкой FUSE Bits BOOTLOCK02, BOOTLOCK01, BOOTLOCK12, BOOTLOCK11;
- Г) установкой Lock Bits LB1 и LB2;
94. Конфигурационные ячейки в МК:
- А) объединяются в байты. Различные микросхемы AVR имеют от одного до трех байтов конфигурационных ячеек; +
- Б) считаются незапрограммированными, если они содержат «1»; +
- В) считаются незапрограммированными, если они содержат «0»;
- Г) необходимы для установки защиты от записи и чтения FLASH-памяти;
- Д) необходимы для установки защиты от записи и чтения EEPROM-памяти;
95. Для обеспечения высокой стабильности частоты в генераторе тактовых импульсов необходим элемент, обладающий следующими параметрами:
- А) высокой фиксирующей способностью; +
- Б) большой эталонностью; +
- В) малыми габаритами; +
- Г) большой ЭДС;
- Д) большой ёмкостью;
- Е) малой эталонностью;

96. Работа кварцевого резонатора основана на:

- А) пьезоэлектрическом эффекте; +
- Б) возможности кварца аккумулировать энергию;
- В) возможности кварца усиливать входной сигнал;
- Г) возможности кварца понижать уровень входного сигнала;

97. Кварцевый резонатор подключается к следующим выводам МК:

- А) XTAL1; +
- Б) XTAL2; +
- В) INT0;
- Г) INT1;
- Д) INT2;
- Е) MOSI;
- ж) MISO;

98. Внешняя RC-цепочка подключается к следующим выводам МК:

- А) XTAL1; +
- Б) GND; +
- в) XTAL2;
- г) RESET;
- д) INT1;
- е) INT2;

99. Внутренний RC-генератор микроконтроллеров AVR может работать на частоте:

- А) 1,0 МГц, если биты CKSEL3...0 равны соответственно 0001; +
- Б) 2,0 МГц, если биты CKSEL3...0 равны соответственно 0010; +
- В) 4,0 МГц, если биты CKSEL3...0 равны соответственно 0011; +
- Г) 8,0 МГц, если биты CKSEL3...0 равны соответственно 0100; +
- Д) 12,0 МГц, если биты CKSEL3...0 равны соответственно 0110;
- Е) 16,0 МГц, если биты CKSEL3...0 равны соответственно 1100;

100. Коэффициент деления частоты следования сигналов сброса от сторожевого таймера в микроконтроллерах AVR можно установить с помощью бит:

- А) WDP0; +
- б) WDP1; +
- в) WDP2; +
- г) PS0;
- д) PS1;

e) PS2;

ж) WDE;

з) WDTOE;

Лабораторная работа №1

Изучение конструкции, организации и состава системы микроконтроллера AVR

Цель работы

Изучение конструкции и архитектуры микроконтроллера AVR Atmega8535, Atmega16, Atmega32 изучить основные приёмы программирования и отладки микроконтроллера.

Компетенции:

Индекс	Формулировка:
ОПК-3	способностью решать задачи анализа и расчета характеристик электрических цепей
ОПК-7	способностью учитывать современные тенденции развития теории автоматического управления, электронного оборудования, измерительной и вычислительной техники, информационных технологий в профессиональной деятельности
ПК-1	способностью выполнять эксперименты на действующих объектах по заданным методикам, оценивать динамические характеристики рассматриваемых объектов и идентифицировать передаточные функции объектов с применением современных информационных технологий и технических средств
ПК-7	готовностью участвовать в разработке и изготовлении стендов для комплексной отладки и испытаний программно-аппаратных управляющих комплексов
ПК-8	готовностью к внедрению результатов разработок средств и систем автоматизации и управления в производстве с использованием современных программируемых логических контролеров (ПЛК)
ПК-11	готовностью производить инсталляцию и настройку системного, прикладного и инструментального программного обеспечения систем автоматизации и управления
ПК-14	способностью разрабатывать электромеханические системы и использовать современную элементную базу при проектировании средств и систем управления

Теоретическая часть

Микроконтроллеры.

Вопрос 1 Конструкция МК микроконтроллера Микроконтроллер (англ. Micro Controller Unit, MCU) — микросхема, предназначенная для управления электронными устройствами. Типичный микроконтроллер сочетает на одном кристалле функции процессора и периферийных устройств, содержит ОЗУ и (или) ПЗУ. По сути, это однокристалльный компьютер, способный выполнять простые задачи.

С появлением однокристалльных микро-ЭВМ связывают начало эры массового применения компьютерной автоматизации в области управления. Повидимому, это обстоятельство и определило термин «контроллер» (англ. controller— регулятор, управляющее устройство).

В связи со спадом отечественного производства и возросшим импортом техники, в том числе вычислительной, термин «микроконтроллер» (МК) вытеснил из употребления ранее использовавшийся термин «однокристалльная микро-ЭВМ».

Первый патент на однокристалльную микро-ЭВМ был выдан в 1971 году инженерам М. Кочрену и Г. Буну, сотрудникам американской Texas Instruments.

Именно они предложили на одном кристалле разместить не только процессор, но и память с устройствами ввода-вывода.

В 1976 году американская фирма Intel выпускает микроконтроллер i8048. В 1978 году фирма Motorola выпустила свой первый микроконтроллер MC6801, совместимый по системе команд с выпущенным ранее микропроцессором MC6800. Через 4 года, в 1980 году, Intel выпускает следующий микроконтроллер: i8051. Удачный набор периферийных устройств, возможность гибкого выбора внешней или внутренней программной памяти и приемлемая цена обеспечили этому микроконтроллеру успех на рынке. С точки зрения технологии микроконтроллер i8051 являлся для своего времени очень сложным изделием — в кристалле было использовано 128 тыс. транзисторов, что в 4 раза превышало количество транзисторов в 16-разрядном микропроцессоре i8086.

На сегодняшний день существует более 200 модификаций микроконтроллеров, совместимых с i8051, выпускаемых двумя десятками компаний, и большое количество микроконтроллеров других типов.

Популярностью у разработчиков пользуются 8-битные микроконтроллеры PIC фирмы Microchip Technology и AVR фирмы Atmel, 16-битные MSP430 фирмы TI, а также 32-битные микроконтроллеры, архитектуры ARM, которую разрабатывает фирма ARM Limited и продаёт лицензии другим фирмам для их производства. Несмотря на популярность в России микроконтроллеров упомянутых выше, по данным Gartner Group от 2009 года мировой рейтинг по объёму продаж выглядит иначе: первое место с большим отрывом занимает Renesas Electronics на втором Freescale, на третьем Samsung, затем идут Microchip и TI, далее все остальные.

В СССР велись разработки оригинальных микроконтроллеров, также осваивался выпуск клонов наиболее удачных зарубежных образцов. В 1979 году в СССР НИИ ТТ разработали однокристалльную 16-разрядную ЭВМ К1801BE1, микроархитектура которой называлась «Электроника НЦ».

При проектировании микроконтроллеров приходится соблюдать баланс между размерами и стоимостью с одной стороны и гибкостью и производительностью с другой. Для разных приложений оптимальное соотношение этих и других параметров может различаться очень сильно.

Поэтому существует огромное количество типов микроконтроллеров, отличающихся архитектурой процессорного модуля, размером и типом встроенной памяти, набором периферийных устройств, типом корпуса и т. д. В отличие от обычных компьютерных микропроцессоров, в микроконтроллерах часто используется гарвардская архитектура памяти, то есть раздельное хранение данных и команд в ОЗУ и ПЗУ соответственно.

Кроме ОЗУ, микроконтроллер может иметь встроенную энергонезависимую память для хранения программы и данных. Во многих контроллерах вообще нет шин для подключения внешней памяти. Наиболее дешёвые типы памяти допускают лишь однократную запись. Такие устройства подходят для массового производства в тех случаях, когда программа контроллера не будет обновляться. Другие модификации контроллеров обладают возможностью многократной перезаписи энергонезависимой памяти.

Неполный список периферии, которая может присутствовать в микроконтроллерах, включает в себя:

- универсальные цифровые порты, которые можно настраивать как на ввод, так и на вывод;
- различные интерфейсы ввода-вывода, такие как UART, I2C, SPI, CAN, USB, IEEE 1394, Ethernet;
- аналого-цифровые и цифро-аналоговые преобразователи;

- компараторы;
- широтно-импульсные модуляторы;
- таймеры;
- контроллеры бесколлекторных двигателей;
- контроллеры дисплеев и клавиатур;
- радиочастотные приемники и передатчики;
- массивы встроенной флеш-памяти;
- встроенный тактовый генератор и сторожевой таймер;

Ограничения по цене и энергопотреблению сдерживают также рост тактовой частоты контроллеров. Хотя производители стремятся обеспечить работу своих изделий на высоких частотах, они, в то же время, предоставляют заказчикам выбор, выпуская модификации, рассчитанные на разные частоты и напряжения питания. Во многих моделях микроконтроллеров используется статическая память для ОЗУ и внутренних регистров. Это даёт контроллеру возможность работать на меньших частотах и даже не терять данные при полной остановке тактового генератора. Часто предусмотрены различные режимы энергосбережения, в которых отключается часть периферийных устройств и вычислительный модуль.

Известные семейства

- MCS 51 (Intel)
- MSP430 (TI)
- ARM (ARM Limited)
- ST Microelectronics STM32 ARM-based MCUs
- Atmel Cortex, ARM7 и ARM9-based MCUs
- Texas Instruments Stellaris MCUs
- NXP ARM-based LPC MCUs
- Toshiba ARM-based MCUs
- Analog Devices ARM7-based MCUs
- Cirrus Logic ARM7-based MCUs
- Freescale Semiconductor ARM9-based MCUs
- AVR (Atmel)
- ATmega
- ATtiny
- XMega
- PIC (Microchip)
- STM8 (STMicroelectronics)

Применение

Использование в современном микроконтроллере достаточно мощного вычислительного устройства с широкими возможностями, построенного на одной микросхеме вместо целого набора, значительно снижает размеры, энергопотребление и стоимость построенных на его базе устройств. Используются в управлении различными устройствами и их отдельными блоками:

- в вычислительной технике: материнские платы, контроллеры дисководов жестких и гибких дисков, CD и DVD, калькуляторах;
- электронике и разнообразных устройствах бытовой техники, в которой используется электронные системы управления — стиральных машинах, микроволновых печах, посудомоечных машинах, телефонах и современных приборах;

В промышленности:

- устройств промышленной автоматики — от программируемого реле и встраиваемых систем до ПЛК,
- систем управления станками

В то время как 8-разрядные процессоры общего назначения полностью

вытеснены более производительными моделями, 8-разрядные микроконтроллеры продолжают широко использоваться. Это объясняется тем, что существует большое количество применений, в которых не требуется высокая производительность, но важна низкая стоимость. В то же время, есть микроконтроллеры, обладающие большими вычислительными возможностями, например цифровые сигнальные процессоры.

Микроконтроллеры семейства AVR.

AVR — семейство восьмибитных микроконтроллеров фирмы Atmel. Год разработки — 1996. В AVR микроконтроллерах есть область (4 байта) которую называют Fuse Bits (фьюз биты), в которой хранится конфигурация микроконтроллера. У каждого AVR микроконтроллера есть свой набор Fuse Bits.

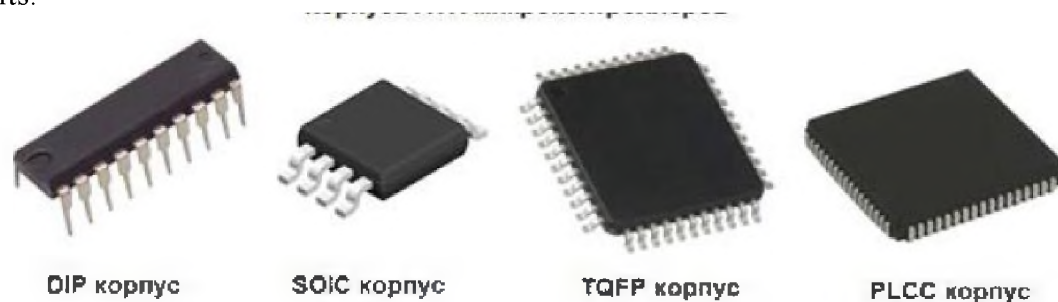


Рис. 1.1

Микроконтроллеры (МК) выпускаются в нескольких корпусах рис. 1.1. Большинство МК имеют SPI интерфейс связи, по которому можно программировать (прошивать) микроконтроллер.

Для AVR микроконтроллеров существует множество компиляторов, например: BASCOM-AVR (Basic компилятор), Code Vision AVR (C (си) компилятор), WinAVR (компилятор C (си) и ассемблера) и т.д. В данной задаче по микроконтроллерам AVR будет использоваться компилятор Algorithm Builder.

Для того чтобы запрограммировать (прошить) микроконтроллер, необходим программатор. Программатор это устройство, которое связывает микроконтроллер и компьютер вместе. Компьютер с помощью специальной программы прошивает микроконтроллер указанной программой (прошивкой).

Отличительные особенности. К особенностям микроконтроллеров AVR семейства Mega можно отнести:

- FLASH-память программ объемом от 8 до 256 Кбайт (число циклов стирания/записи не менее 10 000);
- оперативная память (статическое ОЗУ) объемом от 512 байт до 8 Кбайт;

- память данных на основе ЭСППЗУ (EEPROM) объемом от 256 байт до 4 Кбайт (число циклов стирания/записи не менее 100 000);

- возможность защиты от чтения и модификации памяти программ и данных;

- возможность программирования непосредственно в системе через последовательные интерфейсы SPI и JTAG;

- возможность самопрограммирования;

- возможность внутрисхемной отладки в соответствии со стандартом IEEE 1149.1 (JTAG), а также наличие собственного однопроводного интерфейса внутрисхемной отладки debugWire);

- разнообразные способы синхронизации: встроенный RC-генератор с внутренней или внешней времязадающей RC-цепочкой, встроенный генератор с внешним кварцевым или пьезокерамическим резонатором, внешний сигнал синхронизации;

- наличие нескольких режимов пониженного энергопотребления;

Характеристики процессора

Основными характеристиками процессора микроконтроллеров AVR семейства Mega являются:

- полностью статическая архитектура, минимальная тактовая частота равна нулю;
- арифметико-логическое устройство (АЛУ) подключено непосредственно к регистрам общего назначения С2 регистра);
- большинство команд выполняются за один период тактового сигнала;
- векторная система прерываний, поддержка очереди прерываний;
- большое число источников прерываний (до 45 внутренних и до 32 внешних);
- наличие аппаратного умножителя.

Периферийные устройства

Микроконтроллеры семейства Mega имеют богатый набор периферийных устройств (ПУ):

- один или два 8-битных таймера/счетчика. Во всех моделях с двумя 8-битными таймерами/счетчиками один из них может работать в качестве часов реального времени (в асинхронном режиме);
- от одного до четырех 16-битных таймеров/счетчиков;
- сторожевой таймер;
- одно- и двухканальные генераторы 8-битного ШИМ-сигнала (один из режимов работы 8-битных таймеров/счетчиков);
- двух- и трехканальные генераторы ШИМ-сигнала регулируемой разрядности (один из режимов работы 16-битных таймеров/счетчиков). Разрешение формируемого сигнала может составлять от 1 до 16 бит;
- аналоговый компаратор;
- многоканальный 10-битный АЦП последовательного приближения, имеющий как несимметричные, так и дифференциальные входы;
- последовательный синхронный интерфейс SPI;
- последовательный двухпроводный интерфейс TWI (полный аналог интерфейса I²C);
- от одного до четырех полнодуплексных универсальных синхронных/асинхронных приемо-передатчиков (USART). В ряде моделей эти приемо-передатчики могут использоваться в качестве ведущего устройства шины SPI;
- универсальный последовательный интерфейс USI, который может использоваться в качестве интерфейса SPI или I²C. Кроме того, USI может использоваться в качестве полудуплексного UART или 4/12-битного счетчика.

Вопрос 2 Архитектура ядра

Ядро микроконтроллеров AVR семейства Mega выполнено по усовершенствованной RISC-архитектуре (enhanced RISC) (Рис. 1.2), в которой используется ряд решений, направленных на повышение быстродействия микроконтроллеров.

Арифметико-логическое устройство (АЛУ), выполняющее все вычисления, подключено непосредственно к 32 рабочим регистрам, объединенным в регистровый файл. Благодаря этому, АЛУ может выполнять одну операцию (чтение содержимого регистров, выполнение операции и запись результата обратно в регистровый файл) за

такт. Кроме того, практически каждая из команд (за исключением команд, у которых одним из операндов является 16-битный адрес) занимает одну ячейку памяти программ. В микроконтроллерах AVR реализована Гарвардская архитектура, характеризующаяся отдельной памятью программ и данных, каждая из которых имеет собственные шины доступа. Такая организация позволяет одновременно работать как с памятью программ, так и с памятью данных. Разделение информационных шин позволяет использовать для каждого типа памяти шины различной разрядности, причем способы адресации и доступа к каждому типу памяти также различаются. В сочетании с двухуровневым конвейером команд такая архитектура позволяет достичь производительности в 1 MIPS на каждый МГц тактовой частоты.

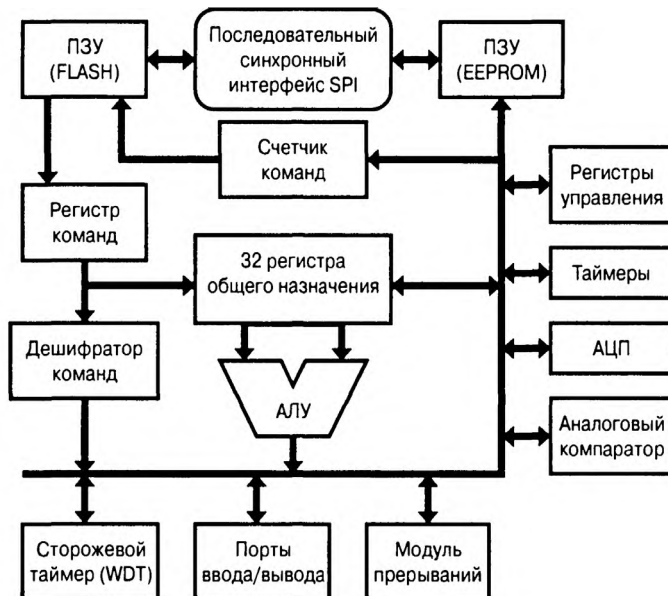


Рис. 1.2. Архитектура ядра микроконтроллеров

Микроконтроллеры ATmega8535, ATmega8535L (Рис. 1.3) — имеют FLASH-память программ объемом 8 Кбайт, ОЗУ объемом 512 байт и EEPROM-память данных объемом 512 байт. Выпускаются также в 44-выводном корпусе типа PLCC.

Описание выводов исследуемого микроконтроллера приведено в табл. 1.1.

Таблица 1.1. Описание выводов моделей ATmega8535x

Обозначение	Номер вывода			Тип вывода	Описание
	DIP	TQFP MLF	PLCC		
XTAL1	13	8	14	I	Вход тактового генератора
XTAL2	12	7	13	O	Выход тактового генератора
RESET	9	4	10	I	Вход сброса
Порт А. 8-битный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами					
PA0 (ADC0)	40	37	43	I/O	0-й бит порта А Вход АЦП
PA1 (ADC1)	39	36	42	I/O	1-й бит порта А Вход АЦП
PA2 (ADC2)	38	35	41	I/O	2-й бит порта А Вход АЦП
PA3 (ADC3)	37	34	40	I/O	3-й бит порта А Вход АЦП
PA4 (ADC4)	36	33	39	I/O	4-й бит порта А Вход АЦП
PA5 (ADC5)	35	32	38	I/O	5-й бит порта А Вход АЦП
PA6 (ADC6)	34	31	37	I/O	6-й бит порта А Вход АЦП
PA7 (ADC7)	33	30	36	I/O	7-й бит порта А Вход АЦП

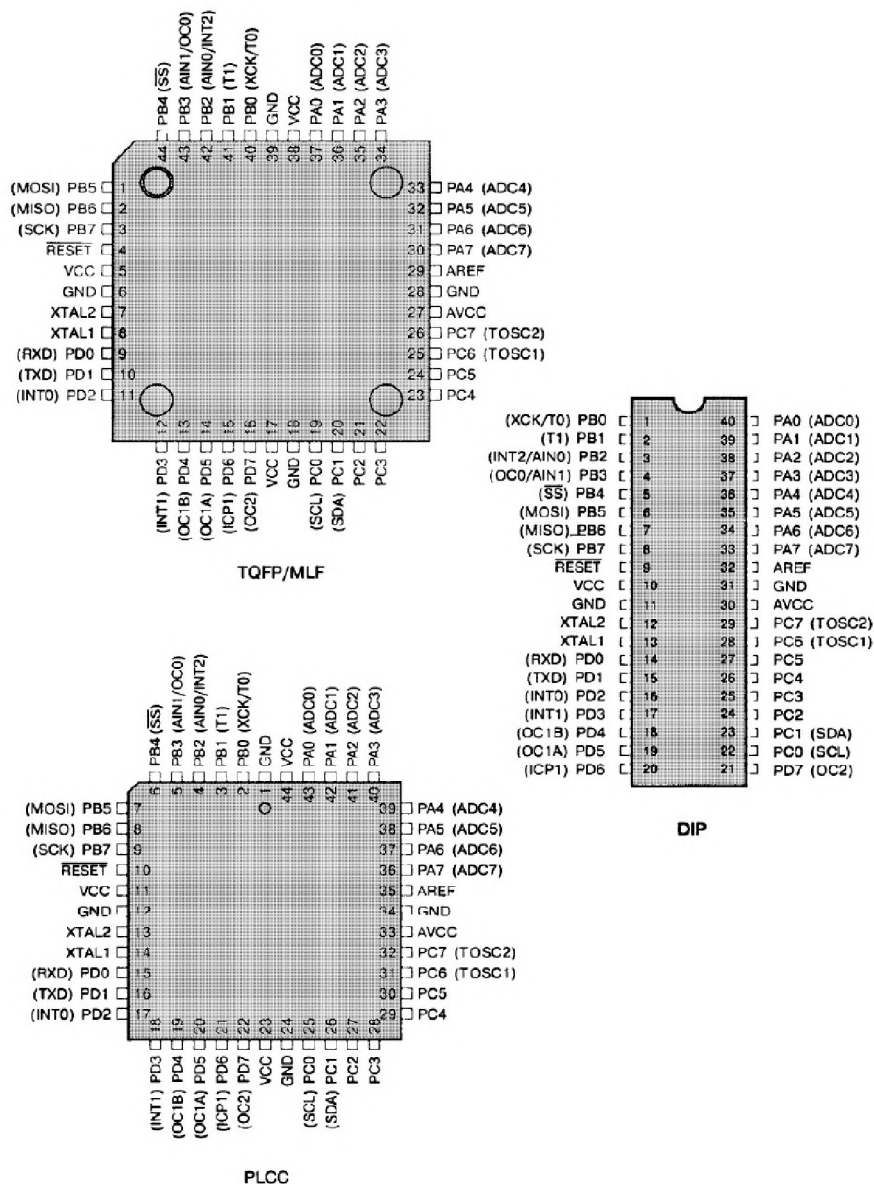


Рис. 1.3. Расположение выводов (вид сверху) моделей ATmega8535x

Таблица 1.1. Описание выводов моделей ATmega8535x
Продолжение в табл. 1.1.

Порт В. 8-битный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами					
PB0 (T0/XCK)	1	40	2	I/O	0-й бит порта В Вход внешнего тактового сигнала таймера/счетчика T0 Вход/выход внешнего тактового сигнала USART
PB1 (T1)	2	41	3	I/O	1-й бит порта В Вход внешнего тактового сигнала таймера/счетчика T1
PB2 (AIN0/INT2)	3	42	4	I/O	2-й бит порта В Неинвертирующий вход компаратора Вход внешнего прерывания
PB3 (AIN1/OC0)	4	43	5	I/O	3-й бит порта В Инвертирующий вход компаратора Выход таймера/счетчика T0

Продолжение в табл. 1.1.

Обозначение	Номер вывода			Тип вывода	Описание
	DIP	TQFP MLF	PLCC		
PB4 (\overline{SS})	5	44	6	I/O	4-й бит порта B Выбор Slave-устройства на шине SPI
PB5 (MOSI)	6	1	7	I/O	5-й бит порта B Выход (Master) или вход (Slave) данных модуля SPI
PB6 (MISO)	7	2	8	I/O	6-й бит порта B Вход (Master) или выход (Slave) данных модуля SPI
PB7 (SCK)	8	3	9	I/O	7-й бит порта B Выход (Master) или вход (Slave) тактового сигнала модуля SPI
Порт C. 8-битный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами					
PC0 (SCL)	22	19	25	I/O	0-й бит порта C Вход/выход тактового сигнала модуля TWI
PC1 (SDA)	23	20	26	I/O	1-й бит порта C Вход/выход данных модуля TWI
PC2	24	21	27	I/O	2-й бит порта C
PC3	25	22	28	I/O	3-й бит порта C
PC4	26	23	29	I/O	4-й бит порта C
PC5	27	24	30	I/O	5-й бит порта C
PC6 (TOSC1)	28	25	31	I/O	6-й бит порта C Вывод для подключения резонатора к таймеру/счетчику T2
PC7 (TOSC2)	29	26	32	I/O	7-й бит порта C Вывод для подключения резонатора к таймеру/счетчику T2
Порт D. 8-битный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами					
PD0 (RXD)	14	9	15	I/O	0-й бит порта D Вход USART
PD1 (TXD)	15	10	16	I/O	1-й бит порта D Выход USART
PD2 (INT0)	16	11	17	I/O	2-й бит порта D Вход внешнего прерывания

Продолжение в табл. 1.1.

Обозначение	Номер вывода			Тип вывода	Описание
	DIP	TQFP MLF	PLCC		
PD3 (INT1)	17	12	18	I/O	3-й бит порта D Вход внешнего прерывания
PD4 (OC1B)	18	13	19	I/O	4-й бит порта D Выход В таймера/счетчика T1
PD5 (OC1A)	19	14	20	I/O	5-й бит порта D Выход А таймера/счетчика T1
PD6 (ICP1)	20	15	21	I/O	6-й бит порта D Вход захвата таймера/счетчика T1
PD7 (OC2)	21	16	22	I/O	7-й бит порта D Выход таймера/счетчика T2
AREF	32	29	35	P	Вход опорного напряжения для АЦП
AVCC	30	27	33	P	Вывод источника питания АЦП
VCC	10	5, 17, 38	11, 23, 44	P	Вывод источника питания
GND	11, 31	6, 18, 28, 39	12, 24, 34, 1	P	Общий вывод

Оборудование и материалы.

Комплект лабораторных модулей микропроцессорная техника РТМТЛ. Принципиальная схема лабораторного стенда приведена ниже

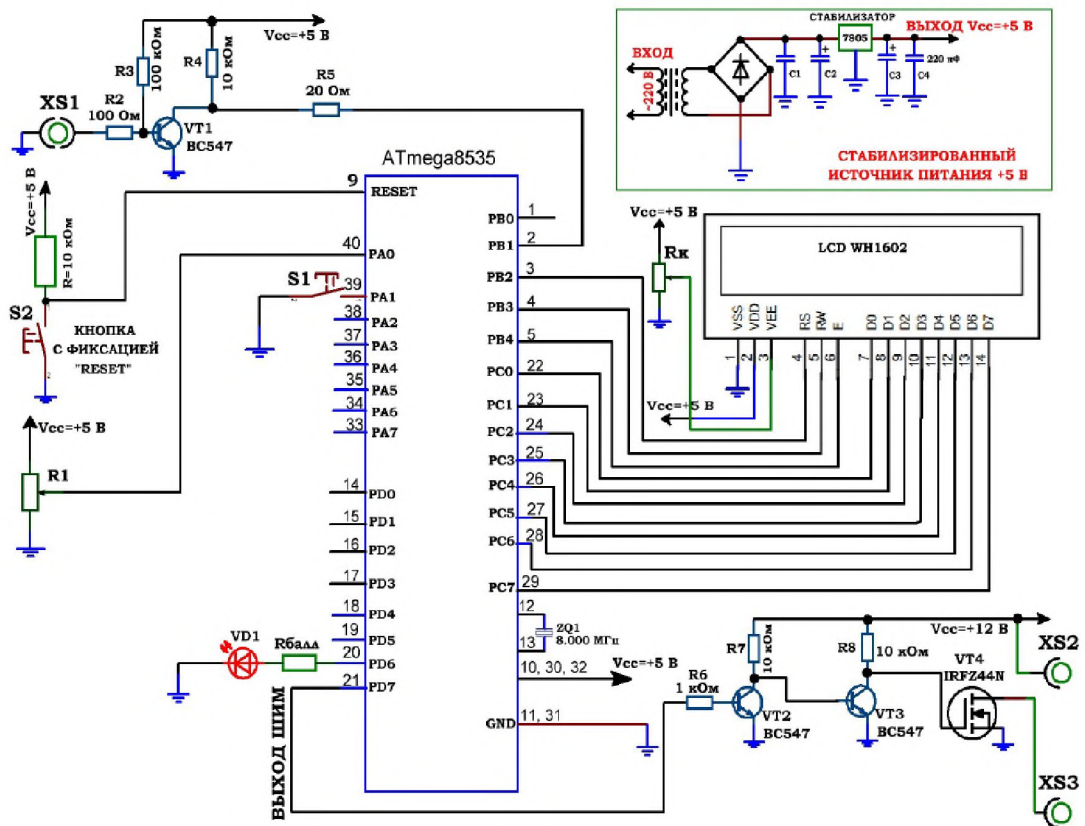


Рис. 3.1. Принципиальная электрическая схема учебного стенда для изучения микроконтроллеров серии Atmega8535.

Указания по технике безопасности

Соответствуют технике безопасности по работе с компьютерной техникой.

Ход работы

1. Перед включением установки в сеть проверить целостность всех соединительных сигнальных и сетевых проводов. Все работы по подключению комплекса к компьютеру следует выполнять только при отключенных от сети приборах. Разобраться с принципиальными блок-схемами опытов, в назначении кнопок, переключателей и ручек прибора.
2. Перед выполнением работы следует изучить инструкцию по работе с учебной средой программирования Algorithm Builder, а также ознакомиться с полным методическим руководством по микроконтроллерам семейства AVR, изучить все доступные паспорта на исследуемые микроконтроллеры, а также на подключенный LCD индикатор.
3. Соединить монитор с системным блоком ПЭВМ, подключить клавиатуру и мышь к системному блоку используя стандартные провода для подключения. Подключить системный блок ПЭВМ и монитор к сети ~220 В.
4. Загрузить операционную систему согласно стандартным процедурам загрузки.
5. При необходимости, настроить компьютер для работы с учебной установкой и программной средой Algorithm Builder.
6. Запустить программу Algorithm Builder для работы с учебной установкой для данного эксперимента пользуясь ярлыком на рабочем столе либо другим способом, указанным лаборантом. **Для работы можно воспользоваться комплексной программной оболочкой LabVisual для РТМТЛ-1-5.**

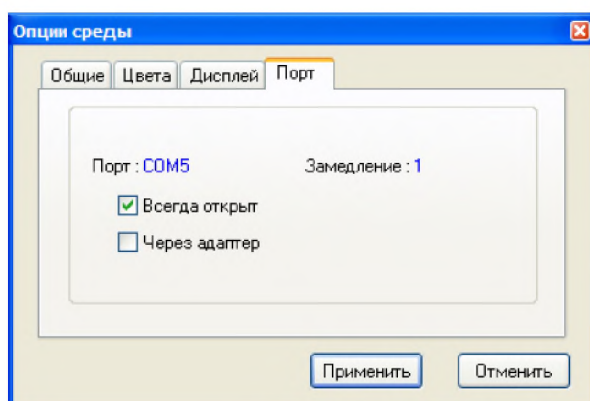


Рис. 4.1

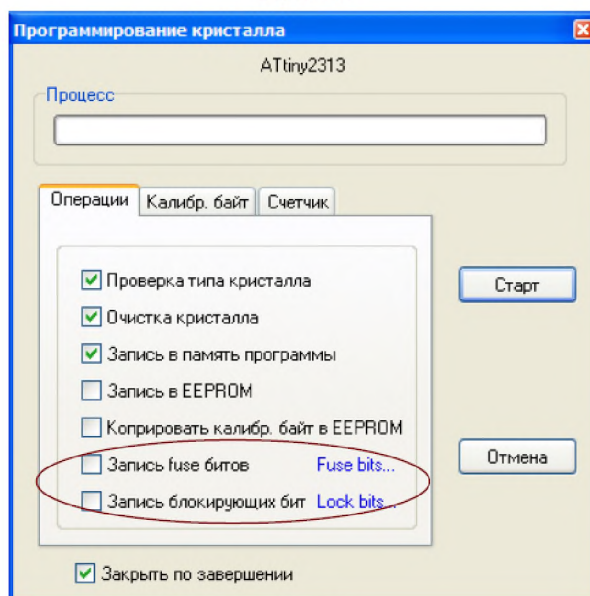


Рис. 4.2

г

6. Подключить разъем «ПРОГРАММАТОР» учебного прибора к СОМ – порту ПЭВМ проводом типа СОМ 9m/9f («мама» - «папа»)
7. Настроить программу Algorithm Builder для работы с данным СОМ портом ОПЦИИ-ОПЦИИ СРЕДЫ-ПОРТ рис. 4.1.
8. Для подробных инструкций следует обращаться к методическому руководству по среде Algorithm Builder.
9. Загрузить Пример 1 (папка 1) в среду Algorithm Builder (ФАЙЛ- ОТКРЫТЬ-ВЫБРАТЬ ФАЙЛ *.alp).
10. «Прошить» тестовую программу в кристалл. Для этого нажать кнопку «ЗАПУСК С КРИСТАЛЛОМ», при этом откроется диалог программирования кристалла рис. 4.2. **Перед прошивкой следует нажать кнопку с фиксацией #RESET# на учебном приборе.**
11. Перед нажатием кнопки «СТАРТ» следует внимательно проверить положения флажков установки программатора рис. 4.2. **Особенно внимательно следует отнестись к опциям #ЗАПИСЬ Fuse bit# и #ЗАПИСЬ блокирующих бит#. ФЛАЖКИ ЭТИХ ОПЦИЙ ОБЯЗАТЕЛЬНО ДОЛЖНЫ БЫТЬ СНЯТЫ, Т. К. НЕ ПРАВИЛЬНО ПРОШИТЫЕ FUSE БИТЫ МОГУТ ПРИВЕСТИ К ДАЛЬНЕЙШЕЙ НЕВОЗМОЖНОСТИ РАБОТЫ С ДАННЫМИ КРОКОНТРОЛЛЕРОМ!**
12. **Отжать кнопку RESET и проверить работу тестовой программы.**
13. Соединить выход LINE IN звуковой карты со входом XS1 стенда проводом «тюльпан — jack» из комплекта.
14. Проверить работу программа-частотомера, подавая на вход XS1 прямоугольный сигнал звуковой частоты 20 — 9000 Гц. При подачи сигнала с линейного выхода LINE OUT ПК посредством программы «ГЕНЕРАТОР», следует установить амплитуду выходного сигнала генератора в программе не менее 80 % (рис. 3.2) и амплитуду выходного сигнала не менее 70 — 80 % непосредственно в звуковом драйвере (стандартная настройка громкости выходного сигнала). При низкой амплитуде входного сигнала, так же как и при слишком высокой амплитуде, частотомер может работать некорректно из за особенностей работы схемы однокаскадного транзисторного усилителя.
15. Загрузить пример 2 из соответствующей папки и прошить в кристалл тестовую программу.
16. Подключить к клеммам XS2 – XS3 нагрузку (вентилятор 12 В). **Обратить внимание, что вентилятор плюсовым красным выводом должен подключаться строго к клемме XS2 (соблюдать полярность).** Наблюдать работу тестовой программы.
17. Исходя из готовых примеров составить собственные программы в среде Algorithm Builder.
18. На основании Примера 1 можно составить программы, изменив, например, максимальное значение измеряемой частоты (в примере 9999 Гц), настройки TIMER1, действия, осуществляемые кнопкой S1 (включать светодиод), либо изменить вывод информации на ЖК индикатор. Также можно проводить различные операции с микроконтроллером.
19. На основании Примера 2 следует изучить работу ШИМ генератора микроконтроллера; изменить действия, осуществляемые кнопкой S1; изменить вывод информации на ЖК индикатор; поработать с потенциометром R1.
20. Для составления программ и подробного изучения работы микроконтроллеров AVR следует обращаться к учебной литературе, полному руководству к микроконтроллерам семейства AVR, а также к документации по среде Algorithm Builder.
21. По окончании работы следует закрыть программу и все открытые подпрограммы, закрыть виртуальную среду VirtualBox (при работе в среде Linux).

22. Выключить компьютер, нажав на кнопку, находящуюся в крайнем нижнем левом углу экрана. Из доступных действий выбрать «ВЫХОД»--> «ВЫКЛЮЧИТЬ КОМПЬЮТЕР».

23. Отключить установку от сети, поставив переключатели «СЕТЬ» на панели установки в положение «ВЫКЛ» и вынуть сетевые вилки из розеток. __

Оформление отчета

В отчете должны содержаться описание команд использованных в выполнении заданий и программы, написанные по заданным в задании алгоритмам

Контрольные вопросы

1. Чем отличается микропроцессор (МП) от микроконтроллера (МК)?
2. Какие типы корпусов используются для микропроцессоров и микроконтроллеров?
3. Какие программные средства используются для программирования МП и МК?
4. Как организована архитектура ядра микроконтроллеров AVR семейства Mega?
5. Какие функциональные блоки имеются в МК и отсутствуют в МП?
6. Каким образом используются выходы МК AVR семейства Mega?
7. Каким напряжением питается AVR семейства Mega?

Список литературы, рекомендуемый к использованию по данной теме:

1. Клингман Э. Проектирование микропроцессорных систем. М.: Мир. 1988. - 575с.
2. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристалльных микроконтроллерах.
3. Трамперт В. AVR-RISK микроконтроллеры.: Пер. с нем. – К.: «МК-ПРЕСС», 2006.-464с., ил.
4. Ю.А.Шпак. Программирование на языке С для AVR и PIC микроконтроллеров. – К.: «МК-ПРЕСС», 2006.-400с., ил.
5. Якубовский С.В. Цифровые и аналоговые интегральные микросхемы. М.: Радио и связь. 1990.
6. Токхейм Р. Основы цифровой электроники. Пер. с англ. - М.: Мир, 1988. – 390 с.
7. Шило В.Л. Популярныe цифровые микросхемы. – М.: Радио и связь, 1990.– 350 с.
8. Бирюков С.А. Цифровые устройства на МОП-интегральных микросхемах - М.: Радио и связь. 1992 (1996).
9. Опадчий Ю.Ф., Глудкин О.П., Гуров А.И. Аналоговая и цифровая электроника. Полный курс: учебник для вузов. - М.: Горячая линия, 1999 (2000, 2005). – 768 с.
10. Алексенко А.В., Шагуров И.И. Микросхемотехника.– М.: Радио и связь, 1990 (1982).
11. Большие интегральные схемы ЗУ./ Под ред. А.Ю. Гордонова, Ю.Н. Дьякова. Справочное пособие. – М: Радио и связь, 1990. – 286 с.
12. Федорков Б.Г., Телец В.А. Микросхемы ЦАП и АЦП: Функционирование, параметры, применение. – М.: Энергоатомиздат, 1990. – 320 с.
13. Гилмор Ч. Введение в микропроцессорную технику. Пер. с англ. – М.:

Лабораторная работа 2

Изучение системы команд микроконтроллера AVR

Цель работы

Изучение системы команд микроконтроллера AVR Atmega8535, Atmega16, Atmega32 и совместимого изучить основные приёмы программирования и отладки микроконтроллера..

Компетенции:

Индекс	Формулировка:
ОПК-3	способностью решать задачи анализа и расчета характеристик электрических цепей
ОПК-7	способностью учитывать современные тенденции развития теории автоматического управления, электронного оборудования, измерительной и вычислительной техники, информационных технологий в профессиональной деятельности
ПК-1	способностью выполнять эксперименты на действующих объектах по заданным методикам, оценивать динамические характеристики рассматриваемых объектов и идентифицировать передаточные функции объектов с применением современных информационных технологий и технических средств
ПК-7	готовностью участвовать в разработке и изготовлении стендов для комплексной отладки и испытаний программно-аппаратных управляющих комплексов
ПК-8	готовностью к внедрению результатов разработок средств и систем автоматизации и управления в производстве с использованием современных программируемых логических контролеров (ПЛК)
ПК-11	готовностью производить инсталляцию и настройку системного, прикладного и инструментального программного обеспечения систем автоматизации и управления
ПК-14	способностью разрабатывать электромеханические системы и использовать современную элементную базу при проектировании средств и систем управления

Теоретическая часть

Система команд микроконтроллеров AVR семейства Mega весьма развита и насчитывает в различных моделях от 130 до 135 различных инструкций. Несмотря на то что микроконтроллеры AVR являются микроконтроллерами с RISC-архитектурой (процессор с сокращенным набором команд), по количеству реализованных инструкций и их разнообразию они больше похожи на микроконтроллеры с CISC-архитектурой (процессор с полным набором команд). Практически каждая из команд (за исключением команд, у которых одним из операндов является 16-битный адрес) занимает одну ячейку памяти программ. Причем это достигнуто не за счет сокращения числа команд процессора, а за счет увеличения разрядности памяти программ.

3.2. Операнды

Программа для любого микроконтроллера представляет собой последовательность команд, записанных в памяти программ. Большинство команд при выполнении изменяют содержимое одного или нескольких регистров общего назначения, регистров ввода/вывода или ячеек ОЗУ.

Для обращения к различным областям адресного пространства памяти данных используются различные команды, реализующие, в свою очередь, различные способы адресации. Подробно способы адресации памяти данных были рассмотрены в главе 2.

Доступ к регистрам ввода/вывода осуществляется по их адресам, являющихся операндами команды. Однако при написании программ гораздо удобнее обращаться к регистрам, используя вместо числовых значений адресов их стандартные, принятые в документации символические имена. Чтобы задать соответствие этих имен реальным адресам, необходимо подключить в начале программы (при помощи директивы ассемблера `.INCLUDE`) файл определения адресов регистров ввода/вывода. Помимо всего прочего, такое решение облегчит перенос программного обеспечения с одного типа кристалла на другой.

Эти файлы (для каждой модели микроконтроллеров семейства) свободно распространяются фирмой Atmel вместе с документацией на микроконтроллеры (в частности, включаемые файлы для всех выпускаемых микроконтроллеров AVR входят в комплект бесплатно распространяемой интегрированной среды AVRStudio). Для РОН, используемых в индексных регистрах, в этих файлах определяются также дополнительные символические имена (Табл. 3.1).

Таблица 3.1. Дополнительные символические имена индексных регистров

Регистр	Символическое имя
R26	XL
R27	XH
R28	YL
R29	YH
R30	ZL
R31	ZH

Названия этих включаемых файлов унифицированы и определяются следующим образом:

```
<номер_модели>def.inc
```

Например, программа для микроконтроллера ATmega128x должна содержать следующую директиву ассемблера:

```
.include "m128def.inc",
```

Необходимо только помнить, что если для обращения к регистру ввода/вывода используются команды обмена с ОЗУ, то к символическому имени требуется прибавить число \$20.

Как говорилось выше, в микроконтроллерах семейства память программ является 16-битной. Поэтому большинство команд описываются 16-битным словом, которое называется также кодом операции (КОП). Код операции — это число, расположенное в памяти программ и определяющее действие, которое необходимо произвести между источником и приемником. Некоторые команды, у которых один из операндов является 16-битным адресом, занимают две ячейки памяти программ. Соответственно, код операции таких команд является 4-байтным числом.

В ряде случаев значение операнда-источника может содержаться непосредственно в коде операции, а не в регистре. Это происходит в том случае, когда операндом-источником является константа.

Некоторые константы, которые могут быть полезны при написании программ, определены в упомянутых включаемых файлах:

IOEND	— значение верхнего адреса области ПВВ;
SRAM_START	— значение младшего адреса ОЗУ, доступного для хранения данных;
SRAM_SIZE	— объем ОЗУ в байтах;
RAMEND	— значение верхнего адреса внутреннего ОЗУ;
XRAMEND	— значение верхнего адреса внешнего ОЗУ (для моделей, не поддерживающих подключение внешнего ОЗУ, эта константа равна 0);
E2END	— значение верхнего адреса EEPROM;
EEPROMEND	— то же, что и E2END;
EEADRBITS	— число задействованных битов регистра адреса EEPROM;

FLASHEND	— значение верхнего адреса памяти программ (в 2-байтных словах);
PAGESIZE	— размер страницы памяти программ (в 2-байтных словах);
NRWW_START_ADDR	— младший адрес области NRWW;
NRWW_STOP_ADDR	— старший адрес области NRWW;
RWW_START_ADDR	— младший адрес области RWW;
RWW_STOP_ADDR	— старший адрес области RWW;
FIRSTBOOTSTART	— наименьший размер области загрузчика;
SMALLBOOTSTART	— то же, что и FIRSTBOOTSTART;
SECONDBOOTSTART	— вторая возможная величина области загрузчика;
THIRDBOOTSTART	— третья возможная величина области загрузчика;
FOURTHBOOTSTART	— наибольший размер области загрузчика;
LARGEBOOTSTART	— то же, что и FOURTHBOOTSTART.

3.3. Типы команд

Все множество команд микроконтроллеров AVR семейства Mega можно разбить на несколько групп:

- команды логических операций;
- команды арифметических операций и команды сдвига;
- команды операций с битами;
- команды пересылки данных;
- команды передачи управления;
- команды управления системой.

Далее подробно описана каждая группа команд.

3.3.1. Команды логических операций

Эти команды позволяют выполнять стандартные логические операции над байтами, такие как логическое умножение (И), логическое сложение (ИЛИ), операцию «Исключающее ИЛИ», а также вычисление обратного (дополнение до единицы) и дополнительного (дополнение до двух) кодов числа. К этой группе можно отнести также команды очистки/установки регистров и команду перестановки полубайтов. Операции производятся между регистрами общего назначения, либо между регистром и константой; результат сохраняется в РОН. Все команды из этой группы выполняются за один такт.

3.3.2. Команды арифметических операций и команды сдвига

К данной группе относятся команды, позволяющие выполнять такие базовые операции, как сложение, вычитание, сдвиг (вправо и влево), инкрементирование, декрементирование, а также умножение. Все операции производятся только над регистрами общего назначения. При этом мик-

роконтроллеры AVR позволяют легко оперировать как знаковыми, так и беззнаковыми числами, а также работать с числами, представленными в дополнительном коде.

Почти все команды рассматриваемой группы выполняются за один такт. Команды умножения и команды, оперирующие 2-байтными значениями, выполняются за два такта.

3.3.3. Команды битовых операций

К данной группе относятся команды, выполняющие установку или сброс заданного бита PОН или PВВ. Причем для изменения битов регистра состояния SREG имеются отдельные команды (точнее говоря, эквивалентные мнемонические обозначения общих команд), так как проверка состояния битов именно этого регистра производится чаще всего. Условно к этой группе можно отнести также две команды передачи управления типа «проверка/пропуск», которые пропускают следующую команду в зависимости от состояния бита PОН или PВВ.

Все задействованные биты PВВ имеют свои символические имена. Определения этих имен описаны в том же включаемом файле, что и определения символических имен адресов регистров (см. раздел 3.2). Соответственно, после включения в программу указанного файла в командах вместо числовых значений номеров битов можно будет указывать их символические имена.

Следует помнить, что в командах CBR и SBR операндом является битовая маска, а не номер бита. Для получения битовой маски из номера бита следует воспользоваться ассемблерным оператором «сдвиг влево» («<<»), как показано в следующем примере:

```
sbr    r16, (1<<SE)+(1<<SM)
out    MCUCR, r16          ; Установить флаги SE и SM регистра MCUCR
```

Всем командам данной группы требуется один такт для выполнения, за исключением случаев, когда в результате проверки происходит пропуск команды. В этом случае команда выполняется за 2 или 3 такта, в зависимости от пропускаемой команды.

3.3.4. Команды пересылки данных

Команды этой группы предназначены для пересылки содержимого ячеек, находящихся в адресном пространстве памяти данных. Разделение адресного пространства на три части (PОН, PВВ, ОЗУ) предопределило разнообразие команд данной группы. Пересылка данных, выполняемая командами группы, может производиться в следующих направлениях:

- PОН ⇔ PОН;
- PОН ⇔ PВВ;
- PОН ⇔ память данных.

Также к данной группе можно отнести стековые команды PUSH и POP, позволяющие сохранять в стеке и восстанавливать из стека содержимое PОН.

На выполнение команд данной группы требуется от одного до трех тактов, в зависимости от команды.

3.3.5. Команды передачи управления

В эту группу входят команды перехода, вызова подпрограмм и возврата из них и команды типа «проверка/пропуск», пропускающие следующую за ними команду при выполнении некоторого условия. Также к этой группе относятся команды сравнения, формирующие флаги регистра SREG и предназначенные, как правило, для работы совместно с командами условного перехода.

В системе команд микроконтроллеров семейства имеются команды как безусловного, так и условного перехода. Команды относительного (RJMP), косвенного (IJMP, EIJMP) и абсолютного (JMP) безусловного перехода являются самыми простыми в этой группе. Их функция заключается только в записи нового адреса в счетчик команд. Команды условного перехода также изменяют содержимое счетчика команд, однако это изменение происходит только при выполнении некоторого условия или, точнее, при определенном состоянии различных флагов регистра SREG.

Таблица 3.2. Сводная таблица команд условного перехода

Проверка	Логическое условие	Команда	Обратная проверка	Логическое условие	Команда	Тип данных
$Rd > Rr$	$Z \cdot (N \oplus V) = 0$	BRLT ¹⁾	$Rd \leq Rr$	$Z + (N \oplus V) = 1$	BRGE ¹⁾	Со знаком
$Rd \geq Rr$	$(N \oplus V) = 0$	BRGE	$Rd < Rr$	$(N \oplus V) = 1$	BRLT	Со знаком
$Rd = Rr$	$Z = 1$	BREQ	$Rd \neq Rr$	$Z = 0$	BRNE	Со знаком
$Rd \leq Rr$	$Z + (N \oplus V) = 1$	BRGE ¹⁾	$Rd > Rr$	$Z \cdot (N \oplus V) = 0$	BRLT ¹⁾	Со знаком
$Rd < Rr$	$(N \oplus V) = 1$	BRLT	$Rd \geq Rr$	$(N \oplus V) = 0$	BRGE	Со знаком
$Rd > Rr$	$C + Z = 0$	BRLO ¹⁾	$Rd \leq Rr$	$C + Z = 1$	BRSH ¹⁾	Без знака
$Rd \geq Rr$	$C = 0$	BRHS/BRCC	$Rd < Rr$	$C = 1$	BRLO/BRCS	Без знака
$Rd = Rr$	$Z = 1$	BREQ	$Rd \neq Rr$	$Z = 0$	BRNE	Без знака
$Rd \leq Rr$	$C = Z = 1$	BRSH ¹⁾	$Rd > Rr$	$C = Z = 0$	BRLO ¹⁾	Без знака
$Rd < Rr$	$C = 1$	BRLO/BRCS	$Rd \geq Rr$	$C = 0$	BRSH/BRCC	Без знака
«Перенос»	$C = 1$	BRCS	«Нет переноса»	$C = 0$	BRCC	—
«Меньше нуля»	$N = 1$	BRMI	«Больше нуля»	$N = 0$	BRPL	—
«Переполнение»	$V = 1$	BRVS	«Нет переполнения»	$V = 0$	BRVC	—
«Ноль»	$Z = 1$	BREQ	«Не ноль»	$Z = 0$	BRNE	—

¹⁾ Для перехода по этому условию операнды предшествующей команды сравнения должны быть записаны в обратном порядке, т. е. вместо CP Rd, Rr → CP Rr, Rd.

Вообще говоря, команды, указанные в Табл. 3.2, являются всего лишь эквивалентными мнемоническими обозначениями команд BRBS *s, k* и BRBC *s, k* с определенными значениями операнда *s*. Команда BREQ *k* имеет, например, такой же код операции, что и команда BRBS *1, k*, а команда BRGE *k* — BRBC *4, k*.

Команды вызова подпрограммы (RCALL, ICALL, EICALL и CALL) работают практически так же, как и команды безусловного перехода. Отличие заключается в том, что, перед тем как выполнить переход, значение счетчика команд сохраняется в стеке. Кроме того, подпрограмма должна заканчиваться командой возврата RET, как показано в следующем примере:

```

...
rcall sp_test      ; Вызов подпрограммы «sp_test»
...               ; Текст основной программы
...
sp_test           ; Метка подпрограммы
push  r2          ; Сохранить r2 в стеке
...              ; Выполнение подпрограммы
...
pop  r2           ; Восстановить r2 из стека
ret              ; Возврат из подпрограммы

```

В приведенном выше примере команда RET заменяет адрес, находящийся в счетчике команд, адресом команды, следующей за командой CALL.

Очевидно, что команды передачи управления нарушают нормальное (линейное) выполнение основной программы. Поэтому при каждом выполнении команды этой группы (кроме команд сравнения) нормальное функционирование конвейера нарушается. Перед загрузкой в конвейер нового адреса производится остановка и очистка выполняемой последовательности команд. В результате на выполнение команд затрачивается больше одного такта. Чтобы получить более точную информацию, обратитесь к таблицам, приведенным в разделе 3.4.

3.3.6. Команды управления системой

В эту группу входят всего 4 команды:

- NOP — пустая команда;
- SLEEP — перевод микроконтроллера в режим пониженного энергопотребления;
- WDR — сброс сторожевого таймера;
- BREAK — команда, используемая внутрисхемным отладчиком.

Все команды этой группы, кроме последней, выполняются за один такт.

3.4. Сводные таблицы команд

В Табл. 3.3...3.8 указаны все команды, которыми располагают микроконтроллеры AVR семейства Mega. В каждой таблице команды сгруппированы по функциональному признаку. В таблицах приведены основные сведения о командах, такие как мнемоническое обозначение команды, ее описание, число тактов, необходимых для ее выполнения, а также флаги регистра SREG, на которые воздействует эта команда. Информация в таблицах изложена в сжатом виде, а детальное описание всех команд приведено в разделе 3.5.

Таблица 3.4. Группа команд арифметических операций

Мнемоника	Описание	Операция	Число тактов	Флаги
SUB Rd, Rr	Вычитание двух РОН	$Rd = Rd - Rr$	1	Z, C, N, V, H
SUBI Rd, K	Вычитание константы из РОН	$Rd = Rd - K$	1	Z, C, N, V, H
SBC Rd, Rr	Вычитание двух РОН с заемом	$Rd = Rd - Rr - C$	1	Z, C, N, V, H
SBCI Rd, K	Вычитание константы из РОН с заемом	$Rd = Rd - K - C$	1	Z, C, N, V, H
SBIW Rdh:Rdl, K	Вычитание константы из регистровой пары	$Rdh:Rdl = Rdh:Rdl - K$	2	Z, C, N, V, S
DEC Rd	Декрементирование РОН	$Rd = Rd - 1$	1	Z, N, V
INC Rd	Инкрементирование РОН	$Rd = Rd + 1$	1	Z, N, V

Таблица 3.3. Группа команд логических операций

Мнемоника	Описание	Операция	Число тактов	Флаги
AND Rd, Rr	«Логическое И» двух РОН	$Rd = Rd \cdot Rr$	1	Z, N, V
ANDI Rd, K	«Логическое И» РОН и константы	$Rd = Rd \cdot K$	1	Z, N, V
EOR Rd, Rr	«Исключающее ИЛИ» двух РОН	$Rd = Rd \oplus Rr$	1	Z, N, V
OR Rd, Rr	«Логическое ИЛИ» двух РОН	$Rd = Rd \vee Rr$	1	Z, N, V
ORI Rd, K	«Логическое ИЛИ» РОН и константы	$Rd = Rd \vee K$	1	Z, N, V
COM Rd	Перевод в обратный код	$Rd = \$FF - Rd$	1	Z, C, N, V
NEG Rd	Перевод в дополнительный код	$Rd = \$00 - Rd$	1	Z, C, N, V, H
CLR Rd	Сброс всех битов РОН	$Rd = Rd \oplus Rd$	1	Z, N, V
SER Rd	Установка всех битов РОН	$Rd = \$FF$	1	—
TST Rd	Проверка РОН на отрицательное или нулевое значение	$Rd \cdot Rd$	1	Z, N, V
SWAP Rd	Обмен местами полубайтов в РОН	$Rd(3...0) = Rd(7...4),$ $Rd(7...4) = Rd(3...0)$	1	—

Таблица 3.4. Группа команд арифметических операций

Мнемоника	Описание	Операция	Число тактов	Флаги
ADD Rd, Rr	Сложение двух РОН	$Rd = Rd + Rr$	1	Z, C, N, V, H
ADC Rd, Rr	Сложение двух РОН с переносом	$Rd = Rd + Rr + C$	1	Z, C, N, V, H
ADIW Rdh:Rdl, K	Сложение регистровой пары с константой	$Rdh:Rdl = Rdh:Rdl + K$	2	Z, C, N, V, S

ASR Rd	Арифметический сдвиг вправо	$Rd(n) = Rd(n+1), n = 0..6$	1	Z, C, N, V
LSL Rd	Логический сдвиг влево	$Rd(n+1) = Rd(n), Rd(0) = 0$	1	Z, C, N, V
LSR Rd	Логический сдвиг вправо	$Rd(n) = Rd(n+1), Rd(7) = 0$	1	Z, C, N, V
ROL Rd	Сдвиг влево через перенос	$Rd(0) = C, Rd(n+1) = Rd(n), C = Rd(7)$	1	Z, C, N, V
ROR Rd	Сдвиг вправо через перенос	$Rd(7) = C, Rd(n) = Rd(n+1), C = Rd(0)$	1	Z, C, N, V
MUL Rd, Rr	Умножение беззнаковых чисел	$R1:R0 = Rd \times Rr$	2	Z, C
MULS Rd, Rr	Умножение чисел со знаком	$R1:R0 = Rd \times Rr$	2	Z, C
MULSU Rd, Rr	Умножение беззнакового числа на число со знаком	$R1:R0 = Rd \times Rr$	2	Z, C
FMUL Rd, Rr	Умножение дробных беззнаковых чисел	$R1:R0 = (Rd \times Rr) \ll 1$	2	Z, C
FMULS Rd, Rr	Умножение дробных чисел со знаком	$R1:R0 = (Rd \times Rr) \ll 1$	2	Z, C
FMULSU Rd, Rr	Умножение дробного беззнакового числа и дробного числа со знаком	$R1:R0 = (Rd \times Rr) \ll 1$	2	Z, C

Мнемоника	Описание	Операция	Число тактов	Флаги
CBR Rd, K	Сброс бита(ов) PОН	$Rd = Rd \bullet (\$FF - K)$	1	Z, N, V
SBR Rd, K	Установка бита(ов) PОН	$Rd = Rd \vee K$	1	Z, N, V
СВI A, b	Сброс бита PВВ	$A.b = 0$	2	—
SBI A, b	Установка бита PВВ	$A.b = 1$	2	—
BCLR s	Сброс флага	$SREG.s = 0$	1	SREG.s
BSET s	Установка флага	$SREG.s = 1$	1	SREG.s
BLD Rd, b	Загрузка бита PОН из флага T (SREG)	$Rd.b = T$	1	—
BST Rr, b	Запись бита PОН в флаг T (SREG)	$T = Rr.b$	1	T
CLC	Сброс флага переноса	$C = 0$	1	C
SEC	Установка флага переноса	$C = 1$	1	C
CLN	Сброс флага отрицательного числа	$N = 0$	1	N
SEN	Установка флага отрицательного числа	$N = 1$	1	N

CLZ	Сброс флага нуля	$Z = 0$	1	Z
SEZ	Установка флага нуля	$Z = 1$	1	Z
CLI	Общее запрещение прерываний	$I = 0$	1	I
SEI	Общее разрешение прерываний	$I = 1$	1	I
CLS	Сброс флага знака	$S = 0$	1	S
SES	Установка флага знака	$S = 1$	1	S
CLV	Сброс флага переполнения дополнительного кода	$V = 0$	1	V
SEV	Установка флага переполнения дополнительного кода	$V = 1$	1	V
CLT	Сброс флага T	$T = 0$	1	T
SET	Установка флага T	$T = 1$	1	T
CLH	Сброс флага половинного переноса	$H = 0$	1	H
SEH	Установка флага половинного переноса	$H = 1$	1	H

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

ST Z, Rr	Косвенная запись	$[Z] = Rr$	2	--
ST Z+, Rr	Косвенная запись с постинкрементом	$[Z] = Rr, Z = Z + 1$	2	--
ST -Z, Rr	Косвенная запись с преддекрементом	$Z = Z - 1, [Z] = Rr$	2	--
STD Z+q, Rr	Косвенная относительная запись	$[Z+q] = Rr$	2	--
STS k, Rr	Непосредственная запись в ОЗУ	$[k] = Rr$	2	-
LPM	Загрузка данных из памяти программ	$R0 = \{Z\}$	3	--
LPM Rd, Z	Загрузка данных из памяти программ	$Rv = \{Z\}$	3	--
LPM Rd, Z+	Загрузка данных из памяти программ с постинкрементом	$Rv = \{Z\}, Z = Z + 1$	3	--
ELPM	Расширенная загрузка данных из памяти программ	$R0 = \{RAMPZ Z\}$	3	--
ELPM Rd, Z	Расширенная загрузка данных из памяти программ	$Rv = \{RAMPZ Z\}$	3	--

Мнемоника	Описание	Операция	Число тактов	Флаги
ELPM Rd, Z+	Расширенная загрузка данных из памяти программ с постинкрементом	$Rv = \{RAMPZ Z\}, RAMPZ.Z = RAMPZ.Z + 1$	3	--
SPM	Запись в память программ	$\{Z\} = R1:R0$	--	--
IN Rd, A	Пересылка из RBV в PОН	$Rd = A$	1	--
OUT A, Rr	Пересылка из PОН в RBV	$A = Rr$	1	--
PUSH Rr	Сохранение байта в стеке	$STACK = Rr$	2	--
POP Rd	Извлечение байта из стека	$Rd = STACK$	2	--

Таблица 3.7. Группа команд передачи управления

Мнемоника	Описание	Операция	Число тактов	Флаги
RJMP k	Относительный безусловный переход	$PC = PC + k + 1$	2	--
IJMP	Косвенный безусловный переход	$PC = Z$	2	--
EIJMP	Расширенный косвенный безусловный переход	$PC = EIND:Z$	2	--
JMP k	Абсолютный переход	$PC = k$	3	--
RCALL k	Относительный вызов подпрограммы	$PC = PC + k + 1$	3 (4)	--
ICALL	Косвенный вызов подпрограммы	$PC = Z$	3 (4)	--
EICALL	Расширенный косвенный вызов подпрограммы	$PC = EIND.Z$	4	--
CALL k	Абсолютный вызов подпрограммы	$PC = k$	4 (5)	--
RET	Возврат из подпрограммы	$PC = STACK$	4 (5)	--
RETI	Возврат из подпрограммы обработки прерывания	$PC = STACK$	4 (5)	I
CP Rd, Rr	Сравнение PОН	$Rd - Rr$	1	Z, N, V, C, H
CPC Rd, Rr	Сравнение PОН с учетом переноса	$Rd - Rr - C$	1	Z, N, V, C, H
CPI Rd, K	Сравнение PОН с константой	$Rd - K$	1	Z, N, V, C, H

CPSE R_d, R_r	Сравнение и пропуск следующей команды при равенстве	Если $R_d = R_r$, то $PC = PC + 2$ (3)	1/2/3	—
SBRC R_r, b	Пропуск следующей команды, если бит РОН сброшен	Если $R_r.b = 0$, то $PC = PC + 2$ (3)	1/2/3	—
SBRS R_r, b	Пропуск следующей команды, если бит РОН установлен	Если $R_r.b = 1$, то $PC = PC + 2$ (3)	1/2/3	—
SBIC A, b	Пропуск следующей команды, если бит РВВ сброшен	Если $A.b = 0$, то $PC = PC + 2$ (3)	1/2/3	—

Мнемоника	Описание	Операция	Число тактов	Флаги
SBIS A, b	Пропуск следующей команды, если бит РВВ установлен	Если $A.b = 1$, то $PC = PC + 2$ (3)	1/2/3	—
BRBC s, k	Переход, если флаг s регистра SREG сброшен	Если $SREG.s = 0$, то $PC = PC + k + 1$	1/2	—
BRBS s, k	Переход, если флаг s регистра SREG установлен	Если $SREG.s = 1$, то $PC = PC + k + 1$	1/2	—
BRCS k	Переход по переносу	Если $C = 1$, то $PC = PC + k + 1$	1/2	—
BRCC k	Переход, если нет переноса	Если $C = 0$, то $PC = PC + k + 1$	1/2	—
BREQ k	Переход по «равно»	Если $Z = 1$, то $PC = PC + k + 1$	1/2	—
BRNE k	Переход по «не равно»	Если $Z = 0$, то $PC = PC + k + 1$	1/2	—
BRSR k	Переход по «больше или равно»	Если $C = 0$, то $PC = PC + k + 1$	1/2	—
BRLO k	Переход по «меньше»	Если $C = 1$, то $PC = PC + k + 1$	1/2	—
BRMI	Переход по «отрицательное значение»	Если $N = 1$, то $PC = PC + k + 1$	1/2	—
BRPL	Переход по «положительное значение»	Если $N = 0$, то $PC = PC + k + 1$	1/2	—
BRGE	Переход по «больше или равно» (числа со знаком)	Если $(N \oplus V) = 0$, то $PC = PC + k + 1$	1/2	—
BRLT	Переход по «меньше нуля» (числа со знаком)	Если $(N \oplus V) = 1$, то $PC = PC + k + 1$	1/2	—
BRHS	Переход по половинному переносу	Если $H = 1$, то $PC = PC + k + 1$	1/2	—
BRHC	Переход, если нет половинного переноса	Если $H = 0$, то $PC = PC + k + 1$	1/2	—
BRTS	Переход, если флаг T установлен	Если $T = 1$, то $PC = PC + k + 1$	1/2	—
BRVS	Переход по переполнению дополнительного кода	Если $V = 1$, то $PC = PC + k + 1$	1/2	—
BRVC	Переход, если нет переполнения дополнительного кода	Если $V = 0$, то $PC = PC + k + 1$	1/2	—

Таблица 3.8. Группа команд управления системой

Мнемоника	Описание	Операция	Число тактов	Флаги
NOP	Нет операции		1	—
SLEEP	Переход в «спящий» режим	См. соответствующие разделы	3	—
WDR	Сброс сторожевого таймера	См. соответствующие разделы	1	—
BREAK	Останов	Используется только внутри-схемным отладчиком	—	—

3.5. Описание команд

В этом разделе в алфавитном порядке перечислены все команды, поддерживаемые микроконтроллерами семейства Mega. Для каждой команды приводится ее детальное описание. При описании команд используются обозначения, приведенные в Табл. 3.9.

Таблица 3.9. Обозначения, используемые при описании команд

Обозначение, символ	Описание
Регистр состояния	
SREG	Регистр состояния микроконтроллера
C	Флаг переноса (0-й бит регистра SREG)
Z	Флаг нуля (1-й бит регистра SREG)
N	Флаг отрицательного значения (2-й бит регистра SREG)
V	Флаг переполнения дополнительного кода (3-й бит регистра SREG)
S	Флаг знака (4-й бит регистра SREG); $S = N \oplus V$
H	Флаг половинного переноса (5-й бит регистра SREG)
T	Пользовательский флаг (6-й бит регистра SREG)
I	Флаг общего разрешения прерываний (7-й бит регистра SREG)
Регистры и операнды	
Rd	Регистр-приемник (иногда также регистр-источник) в регистровом файле
Rr	Регистр-источник в регистровом файле
K	Константа (данные)
k	Адрес — константа
b	Номер бита PОН или PВВ (0...7)
s	Номер бита регистра состояния SREG (0...7)
X, Y, Z	Индексные регистры ($X = R27:R26$, $Y = R29:R28$, $Z = R31:R30$)

Обозначение, символ	Описание
I/O	Регистр ввода/вывода
A	Адрес в пространстве ввода/вывода
q	Смещение при относительной косвенной адресации (6-битное значение)
.	Разделитель между названием (адресом) регистра и номером бита
[XX]	Содержимое ячейки памяти данных по адресу XX
{XX}	Содержимое ячейки памяти программ по адресу XX

Операции	
—	Инверсия
•	Логическое И
v	Логическое ИЛИ
⊕	Исключающее ИЛИ
Система	
PC	Счетчик команд
STACK	Текущий уровень стека
SP	Указатель стека
Флаги	
↔	Команда воздействует на флаг
0	Флаг сбрасывается командой в 0
1	Флаг устанавливается командой в 1
—	Команда не влияет на состояние флага

Оборудование и материалы.

Комплект лабораторных модулей микропроцессорная техника РТМТЛ. Принципиальная схема лабораторного стенда приведена ниже

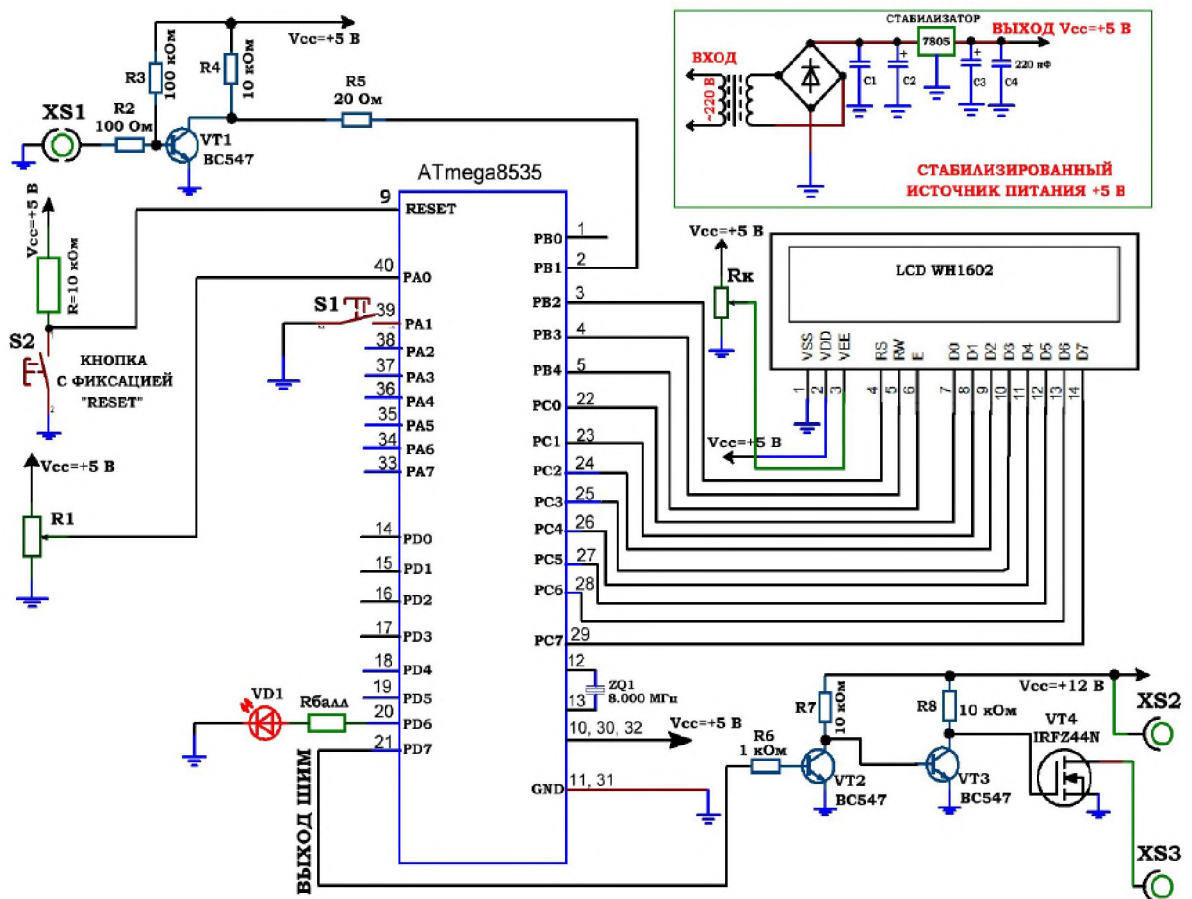


Рис. 3.1. Принципиальная электрическая схема учебного стенда для изучения микроконтроллеров серии Atmega8535.

Указания по технике безопасности

Соответствуют технике безопасности по работе с компьютерной техникой.

Ход работы

1. Перед включением установки в сеть проверить целостность всех соединительных сигнальных и сетевых проводов. Все работы по подключению комплекса к компьютеру следует выполнять только при отключенных от сети приборах. Разобраться с принципиальными блок-схемами опытов, в назначении кнопок, переключателей и ручек прибора.

2. Перед выполнением работы следует изучить инструкцию по работе с учебной средой программирования Algorithm Builder, а также ознакомиться с полным методическим руководством по микроконтроллерам семейства AVR, изучить все доступные паспорта на исследуемые микроконтроллеры, а также на подключенный LCD индикатор.

3. Соединить монитор с системным блоком ПЭВМ, подключить клавиатуру и мышь к системному блоку используя стандартные провода для подключения. Подключить системный блок ПЭВМ и монитор к сети ~220 В.

Задание

4. Загрузить операционную систему согласно стандартным процедурам загрузки.

5. При необходимости, настроить компьютер для работы с учебной установкой и программной средой Algorithm Builder. папке Examples_RTMTL-2.

6. Пример 1. По нажатию кнопки S1, подключенной к 22 выводу МК начинает светиться светодиод VD1, подключенный к выводу 1 микроконтроллера. По отжатию кнопки, светодиод перестает светиться (индикация работы кнопки);

7. Пример 2. При нажатии кнопки S1, подключенной к 22 выводу МК начинает мигать светодиод VD1, подключенный к выводу 1 микроконтроллера;

8. Пример 3. Мигание светодиода VD1, подключенного к выводу 1. 9. Пример 4. Светодиоды VD1 – VD8 мигают по очереди (по порядку подключения к выводам 1, 2, 3, 4, 18, 19, 20, 21 МК);

5) Пример 5. Светодиоды VD1 – VD8 мигают по очереди в той же последовательности с одновременной передачей по USART через микросхему MAX232 на разъём «ДАННЫЕ» (выход на COM порт) номера включенного светодиода в ASCII коде (натуральное число);

Оформление отчета

В отчете должны содержаться описание команд использованных в выполнении заданий и программы написанные по заданным в задании алгоритмам

Контрольные вопросы

1. Какие типы команд имеет микроконтроллер AVR семейства Mega?
2. Какие типы команд изменяют флаги микроконтроллера?
3. Какие виды адресации использует микроконтроллер AVR?
4. Зачем нужны команды передачи управления?
5. Как организована архитектура ядра микроконтроллеров AVR семейства Mega?

Список литературы, рекомендуемый к использованию по данной теме:

1. Клингман Э. Проектирование микропроцессорных систем. М.: Мир. 1988. - 575с.
2. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах.
3. Трамперт В. AVR-RISK микроконтроллеры.: Пер. с нем. – К.: «МК-ПРЕСС», 2006.-464с., ил.
4. Ю.А.Шпак. Программирование на языке С для AVR и PIC микроконтроллеров. – К.: «МК-ПРЕСС», 2006.-400с., ил.

5. Якубовский С.В. Цифровые и аналоговые интегральные микросхемы. М.: Радио и связь. 1990.
6. Токхейм Р. Основы цифровой электроники. Пер. с англ. - М.: Мир, 1988. – 390 с.
7. Шило В.Л. Популярныe цифровые микросхемы. – М.: Радио и связь, 1990.– 350 с.
8. Бирюков С.А. Цифровые устройства на МОП-интегральных микросхемах - М.: Радио и связь. 1992 (1996).
9. Опадчий Ю.Ф., Глудкин О.П., Гуров А.И. Аналоговая и цифровая электроника. Полный курс: учебник для вузов. - М.: Горячая линия, 1999 (2000, 2005). – 768 с.
10. Алексенко А.В., Шагуров И.И. Микросхемотехника.– М.: Радио и связь, 1990 (1982).

Лабораторная работа №3

Изучение работы и программирования портов микроконтроллера AVR

Цель работы

Изучение системы команд микроконтроллера AVR Atmega8535, Atmega16, Atmega32 и совместимого изучить основные приёмы программирования и отладки микроконтроллера..

Компетенции:

Индекс	Формулировка:
ОПК-3	способностью решать задачи анализа и расчета характеристик электрических цепей
ОПК-7	способностью учитывать современные тенденции развития теории автоматического управления, электронного оборудования, измерительной и вычислительной техники, информационных технологий в профессиональной деятельности
ПК-1	способностью выполнять эксперименты на действующих объектах по заданным методикам, оценивать динамические характеристики рассматриваемых объектов и идентифицировать передаточные функции объектов с применением современных информационных технологий и технических средств
ПК-7	готовностью участвовать в разработке и изготовлении стендов для комплексной отладки и испытаний программно-аппаратных управляющих комплексов
ПК-8	готовностью к внедрению результатов разработок средств и систем автоматизации и управления в производстве с использованием современных программируемых логических контролеров (ПЛК)
ПК-11	готовностью производить инсталляцию и настройку системного, прикладного и инструментального программного обеспечения систем автоматизации и управления
ПК-14	способностью разрабатывать электромеханические системы и использовать современную элементную базу при проектировании средств и систем управления

Теоретическая часть

6.2. Регистры портов ввода/вывода

Обращение к портам производится через регистры ввода/вывода. Под каждый порт в адресном пространстве ввода/вывода зарезервировано по 3 адреса, по которым размещены следующие регистры: регистр данных порта PORTx, регистр направления данных DDRx и регистр выводов порта PINx. Действительные названия регистров получаются подстановкой названия порта вместо символа x. Соответственно, регистры порта A называются PORTA, DDRA, PINA, порта B — PORTB, DDRB, PINB и т. д. Поскольку с помощью регистров PINx осуществляется доступ к физическим значениям сигналов на выводах порта, они доступны только для чтения,

тогда как остальные два регистра доступны и для чтения, и для записи. Тем не менее, в новых моделях микроконтроллеров (все модели, кроме ATmega8515x/8535x, ATmega8x/16x/32x/64x/128x и ATmega162x) запись 1 в бит регистра PINx приводит к переключению состояния соответствующего бита регистра данных PORTx.

Адреса регистров всех портов ввода/вывода приведены в Табл. 6.2.

Таблица 6.2. Регистры портов ввода/вывода

Порт	Регистр	ATmega8515x	ATmega8535x	ATmega8x	ATmega16x	ATmega162x	ATmega64x, ATmega128x	ATmega48x/88x/168x	ATmega164x/324x/644x	ATmega165x, ATmega325x/645x	ATmega3250x/6450x	ATmega1281x/2561x	ATmega640x, ATmega1280x/2560x
A	PORTA	\$1B (\$3B)	—	—	\$1B (\$3B)	—	—	—	—	—	\$02 (\$22)	—	—
	DDRA	\$1A (\$3A)	—	—	\$1A (\$3A)	—	—	—	—	—	\$01 (\$21)	—	—
	PINA	\$19 (\$39)	—	—	\$19 (\$39)	—	—	—	—	—	\$00 (\$20)	—	—
B	PORTB	—	—	—	\$18 (\$38)	—	—	—	—	—	\$05 (\$25)	—	—
	DDRB	—	—	—	\$17 (\$37)	—	—	—	—	—	\$04 (\$24)	—	—
	PINB	—	—	—	\$16 (\$36)	—	—	—	—	—	\$03 (\$23)	—	—
C	PORTC	—	—	—	\$15 (\$35)	—	—	—	—	—	\$08 (\$28)	—	—
	DDRC	—	—	—	\$14 (\$34)	—	—	—	—	—	\$07 (\$27)	—	—
	PINC	—	—	—	\$13 (\$33)	—	—	—	—	—	\$06 (\$26)	—	—

D	PORTD	\$12 (\$32)						\$0B (\$2B)				
	DDRD	\$11 (\$31)						\$0A (\$2A)				
	PIND	\$10 (\$30)						\$09 (\$29)				
E	PORTE	\$07 (\$27)	—	—	\$07 (\$27)	\$03 (\$23)	—	—	\$0E (\$2E)			
	DDRE	\$06 (\$26)	—	—	\$06 (\$26)	\$02 (\$22)	—	—	\$0D (\$2D)			
	PINE	\$05 (\$25)	—	—	\$05 (\$25)	\$01 (\$21)	—	—	\$0C (\$2C)			

F	PORTF	\$03 (\$23)	—	—	—	—	(\$62)	—	—	\$11 (\$31)					
	DDRF	\$02 (\$22)	—	—	—	—	(\$61)	—	—	\$10 (\$30)					
	PINF	\$01 (\$21)	—	—	—	—	\$00 (\$20)	—	—	\$0F (\$2F)					
G	PORTG	—	—	—	—	—	(\$65)	—	—	\$14 (\$34)					
	DDRG	—	—	—	—	—	(\$64)	—	—	\$13 (\$33)					
	PING	—	—	—	—	—	(\$63)	—	—	\$12 (\$32)					
H	PORTH	—	—	—	—	—	—	—	—	—	—	—	(\$DA)	—	(\$102)
	DDRH	—	—	—	—	—	—	—	—	—	—	—	(\$D9)	—	(\$101)
	PINH	—	—	—	—	—	—	—	—	—	—	—	(\$D8)	—	(\$100)
J	PORTJ	—	—	—	—	—	—	—	—	—	—	—	(\$DD)	—	(\$105)
	DDRJ	—	—	—	—	—	—	—	—	—	—	—	(\$DC)	—	(\$104)
	PINJ	—	—	—	—	—	—	—	—	—	—	—	(\$DB)	—	(\$103)

K	PORTK	—	—	—	—	—	—	—	—	—	—	—	—	—	(\$108)
	DDRK	—	—	—	—	—	—	—	—	—	—	—	—	—	(\$107)
	PINK	—	—	—	—	—	—	—	—	—	—	—	—	—	(\$106)
L	PORTL	—	—	—	—	—	—	—	—	—	—	—	—	—	(\$10B)
	DDRL	—	—	—	—	—	—	—	—	—	—	—	—	—	(\$10A)
	PINL	—	—	—	—	—	—	—	—	—	—	—	—	—	(\$109)

6.3. Конфигурирование портов ввода/вывода

Упрощенная структурная схема одного из каналов порта ввода/вывода Pxi при работе его в качестве цифрового входа/выхода общего назначения приведена на Рис. 6.1.

Если же вывод функционирует как вход ($DDxn = 0$), то бит $PORTxn$ определяет состояние внутреннего подтягивающего резистора для данного вывода. При установке бита $PORTxn$ в 1 подтягивающий резистор подключается между выводом микроконтроллера и линией питания.

Вообще говоря, управление подтягивающими резисторами во всех микроконтроллерах семейства осуществляется на двух уровнях. Общее управление (для всех выводов портов) осуществляется битом PUD регистра специальных функций SFIOR или регистра управления микроконтроллера MCUCR (в зависимости от модели). В моделях ATmega64x и ATmega128x регистр SFIOR располагается по адресу \$20 (\$40), а в остальных моделях — по адресу \$30 (\$50). Регистр MCUCR располагается по адресу \$35 (\$55). Форматы этих регистров приведены на Рис. 6.2.

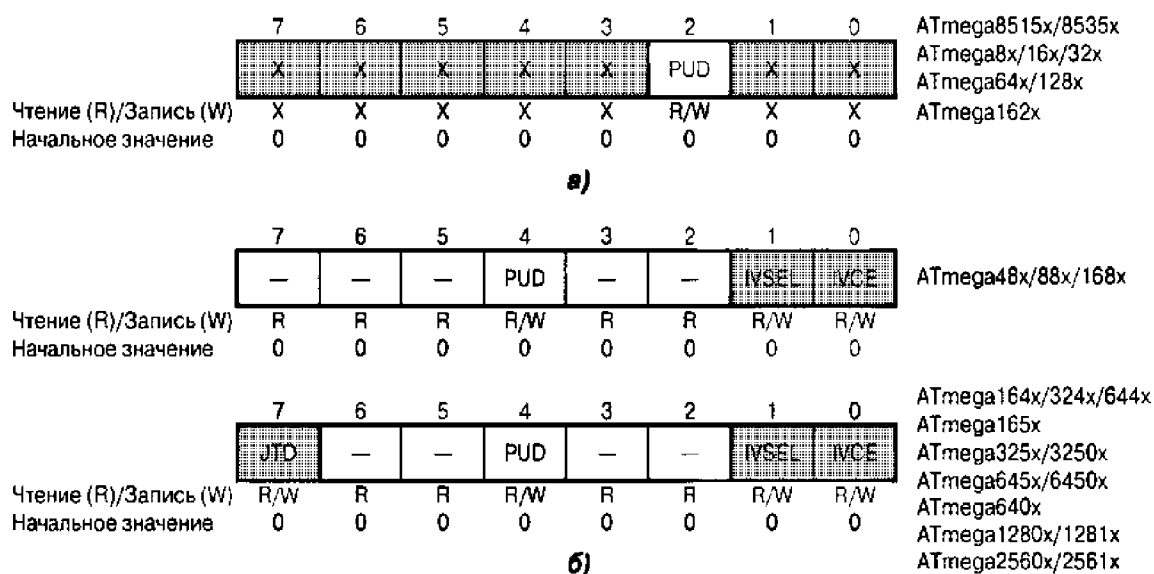


Рис. 6.2 Регистры управления подтяжкой SFIOR (а) и MCUCR (б)

Если бит PUD сброшен в 0 (начальное состояние), то состояние подтягивающих резисторов будет определяться состоянием битов $PORTxn$ для каждого входа порта. Если же бит PUD установлен в 1, подтягивающие резисторы отключаются от всех выводов микроконтроллера.

Обратите внимание, что при переключении вывода между третьим состоянием ($DDxn = 0$, $PORTxn = 0$) и состоянием ВЫСОКОГО уровня ($DDxn = 1$, $PORTxn = 1$) происходит переход через одно из промежуточных состояний: либо включается подтягивающий резистор ($DDxn = 0$, $PORTxn = 1$), либо выход переключается в состояние НИЗКОГО уровня ($DDxn = 1$, $PORTxn = 0$). Наиболее применимым является, как правило, первый вариант, поскольку для высокоимпедансных систем безразлично, каким образом формируется ВЫСОКИЙ уровень. Если в каком-либо слу-

Если же вывод функционирует как вход ($DDxn = 0$), то бит $PORTxn$ определяет состояние внутреннего подтягивающего резистора для данного вывода. При установке бита $PORTxn$ в 1 подтягивающий резистор подключается между выводом микроконтроллера и линией питания.

Вообще говоря, управление подтягивающими резисторами во всех микроконтроллерах семейства осуществляется на двух уровнях. Общее управление (для всех выводов портов) осуществляется битом PUD регистра специальных функций SFIOR или регистра управления микроконтроллера MCUCR (в зависимости от модели). В моделях ATmega64x и ATmega128x регистр SFIOR располагается по адресу \$20 (\$40), а в остальных моделях — по адресу \$30 (\$50). Регистр MCUCR располагается по адресу \$35 (\$55). Форматы этих регистров приведены на Рис. 6.2.

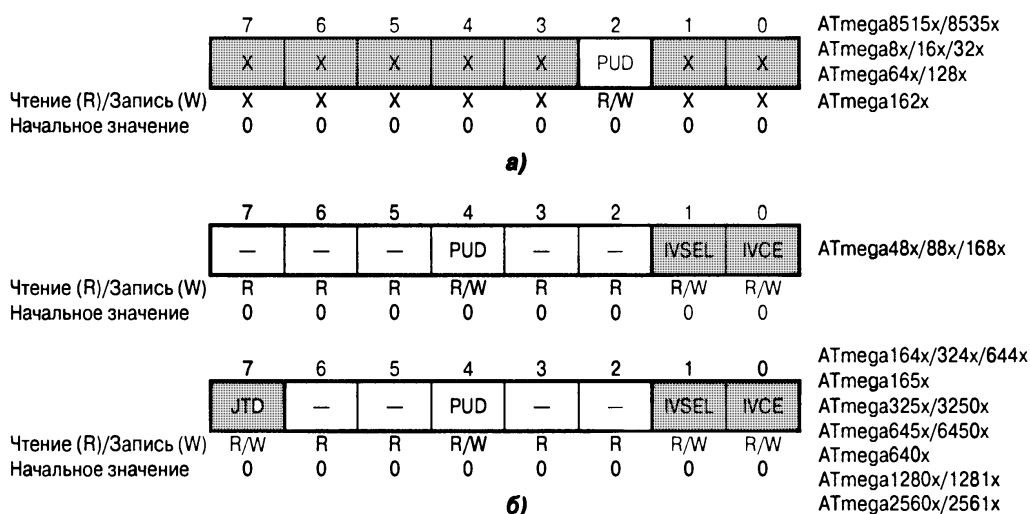


Рис. 6.2 Регистры управления подтяжкой SFIOR (а) и MCUCR (б)

Если бит PUD сброшен в 0 (начальное состояние), то состояние подтягивающих резисторов будет определяться состоянием битов $PORTxn$ для каждого входа порта. Если же бит PUD установлен в 1, подтягивающие резисторы отключаются от всех выводов микроконтроллера.

Обратите внимание, что при переключении вывода между третьим состоянием ($DDxn = 0$, $PORTxn = 0$) и состоянием ВЫСОКОГО уровня ($DDxn = 1$, $PORTxn = 1$) происходит переход через одно из промежуточных состояний: либо включается подтягивающий резистор ($DDxn = 0$, $PORTxn = 1$), либо выход переключается в состояние НИЗКОГО уровня ($DDxn = 1$, $PORTxn = 0$). Наиболее применимым является, как правило, первый вариант, поскольку для высокоимпедансных систем безразлично, каким образом формируется ВЫСОКИЙ уровень. Если в каком-либо случае это не подходит, пользователь может отключить подтягивающие резисторы от всех портов установкой бита PUD в 1.

Аналогичная ситуация возникает и при переключении между состоянием с включенным подтягивающим резистором ($DDx_n = 0, PORTx_n = 1$) и состоянием НИЗКОГО уровня ($DDx_n = 1, PORTx_n = 0$). В этом случае промежуточным состоянием является либо высокоимпедансное состояние ($DDx_n = 0, PORTx_n = 0$), либо состояние ВЫСОКОГО уровня ($DDx_n = 1, PORTx_n = 1$).

Все возможные сочетания состояний управляющих битов и соответственно конфигурации выводов портов приведены в Табл. 6.3.

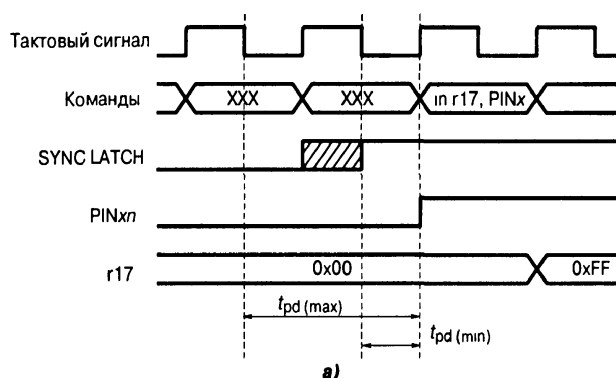
Таблица 6.3. Конфигурации выводов портов

DDx _n	PORTx _n	PUD	Функция вывода	Резистор	Примечание
0	0	X	Вход	Отключен	Третье состояние (Hi-Z) ¹⁾
0	1	0	Вход	Подключен	При подключении нагрузки между выводом и общим проводом вывод является источником тока
0	1	1	Вход	Отключен	Третье состояние (Hi-Z)
1	0	X	Выход	Отключен	Выход установлен в 0
1	1	X	Выход	Отключен	Выход установлен в 1

¹⁾ Состояние выводов портов при сбросе.

Состояние вывода микроконтроллера (независимо от установок бита DDx_n) может быть получено путем чтения бита $PINx_n$ регистра $PINx$. При этом следует помнить, что между действительным изменением сигнала на выводе и изменением бита $PINx_n$ существует задержка. Эта задержка вносится узлом синхронизации, состоящим, как показано на Рис. 6.1, из бита $PINx_n$ и дополнительного триггера-зашелки. Значение сигнала на выводе микроконтроллера фиксируется триггером-зашелкой при НИЗКОМ уровне тактового сигнала и переписывается затем в бит $PINx_n$ по нарастающему фронту тактового сигнала. Соответственно, величина задержки может составлять от 0.5 до 1.5 периодов системного тактового сигнала, как показано на Рис. 6.3, а.

По этой же причине между операциями изменения и повторного считывания состояний вывода необходимо вставлять команду NOP. Поскольку команда OUT устанавливает сигнал «SYNC LATCH» в 1 по положительному фронту тактового сигнала, задержка в этом случае равна одному периоду тактового сигнала (Рис. 6.3, б).



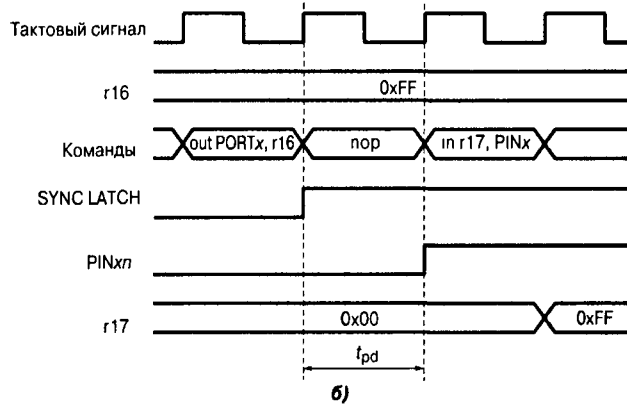


Рис. 6.3. Синхронизация при чтении состояния вывода:
а — при считывании бита $PINxI$; *б* — при считывании состояния вывода, заданного программно

Далее приведен пример конфигурирования одного из портов микроконтроллера. В примере выходы 0 и 1 порта В устанавливаются в 1, выходы 2 и 3 — в 0. Выводы 4...7 порта конфигурируются как входы, при этом к выводам 6 и 7 подключаются подтягивающие резисторы.

Пример на ассемблере

```

...
ldi r16, (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0)
ldi r17, (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0)
out PORTB, r16 ; Задать состояние выходов и подтягивающих
                ; резисторов
out DDRB, r17 ; Задать режимы работы выводов
nop           ; для синхронизации
in r16, PINB ; Считать состояние выводов порта
...

```

Пример на Си

```

unsigned char i;
...
/* Задать состояние выходов и подтягивающих резисторов */
/* Задать режимы работы выводов */
PORTB = (1<<PB7)|(1<<PB6)|(1<<PB1)|(1<<PB0);
DDRB = (1<<DDB3)|(1<<DDB2)|(1<<DDB1)|(1<<DDB0);
_NOP(); /* Синхронизация */
i = PINB; /* Считать состояние выводов порта */
...

```

В заключение отметим, что подавляющее большинство контактов ввода/вывода всех микроконтроллеров семейства имеют дополнительные функции и могут использоваться различными периферийными устройствами микроконтроллеров. При этом возможны две ситуации. В одних случаях пользователь должен самостоятельно задавать конфигурацию вывода, а в других вывод конфигурируется автоматически при включении соответствующего периферийного устройства. Об этом будет сказано при рассмотрении соответствующих периферийных устройств.

Оборудование и материалы.

Комплект лабораторных модулей микропроцессорная техника РТМТЛ. Принципиальная схема лабораторного стенда приведена ниже

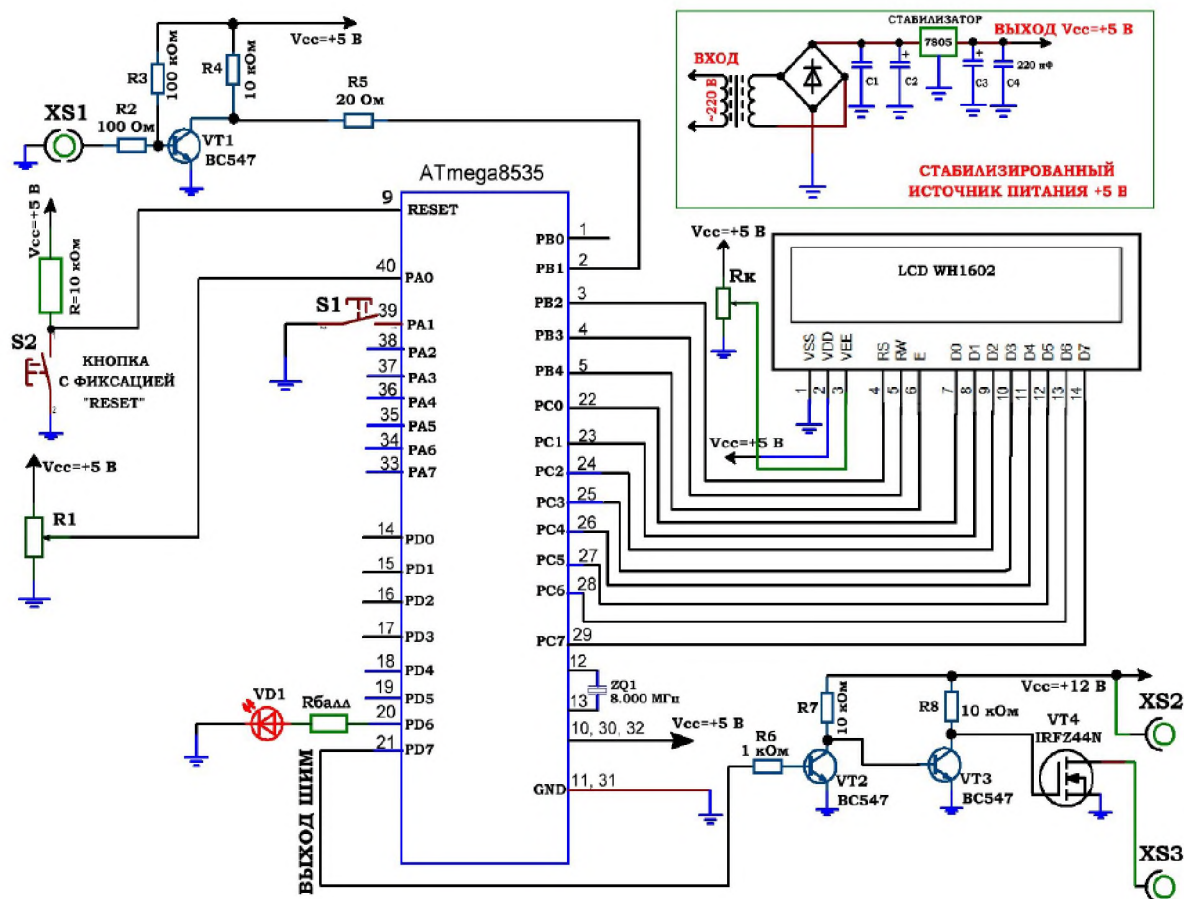


Рис. 3.1. Принципиальная электрическая схема учебного стенда для изучения микроконтроллеров серии Atmega8535.

Указания по технике безопасности

Соответствуют технике безопасности по работе с компьютерной техникой.

Ход работы

Задание 1. Изучение работы АЦП микроконтроллера. Напряжение считывается с переменного резистора R1, подключённого средней точкой к 40 выводу МК (вход АЦП) в код АЦП от 0 до 63 единиц и с интервалом ~ 1 сек. передаётся по USART через микросхему MAX232 на разъём «ДАННЫЕ» (выход на COM порт);

Задание 2. Приёмник RS232 с ПК. Передавая АСП код 1 ... 8 (натуральные числа) с ПК посредством программы «ТЕРМИНАЛ» можно управлять включением светодиодов;

Задание 3. Работа с семисегментными индикаторами, RS232 и реле. Реализация таймера обратного отсчета. Текущие показания таймера выводятся на семисегментные индикаторы с общими катодами (a, b, c, d, e, f, g) и анодами A1 и A2, и одновременно передаётся по USART через микросхему MAX232 на разъём «ДАННЫЕ» (выход на COM порт). При достижении конца отсчета Принципиальная электрическая схема учебного стенда для изучения микроконтроллеров серии Atmega8535. срабатывает реле K1, подключённое через транзисторный усилитель на KT3102 к выводу 5 микроконтроллера и вторичная обмотка трансформатора TV1 (~12 В) соединяется с клеммами XS1 – XS2 (нагрузка). К выходу нагрузки можно подключать лампу накаливания 12 В, 21 Вт либо меньшей мощности. Запрещается замыкать выходы контрольных точек XS1 – XS2 (нагрузка)!;

Задание 4. Реализация вольтметра. Работа с семисегментными индикаторами и реле. Простейшая реализация вольтметра с выводом показаний на семисегментный индикатор. Напряжение считывается с переменного резистора R1, подключённого средней точкой к 40 выводу МК (вход АЦП) и выводится на семисегментный индикатор в показаниях 0-50 (0-5 Вольт, т. к. вывод h(точка dp) не подключен). При определенном пороге (определенном значении напряжения) срабатывает реле K1 и вторичная обмотка трансформатора TV1 (~12 В) соединяется с клеммами XS1 – XS2 (нагрузка). К выходу нагрузки можно подключать лампу накаливания 12 В, 21 Вт либо меньшей мощности. Запрещается замыкать выходы контрольных точек XS1 –__

Оформление отчета

В отчете должны содержаться описание команд использованных в выполнении заданий и программы написанные по заданным в задании алгоритмам

Контрольные вопросы

1. Какой уровень сигнала у цифровых выходов МК?
2. Какие типы команд имеет микроконтроллер AVR семейства Mega?
3. Какие типы команд изменяют флаги микроконтроллера?
4. Какие виды адресации использует микроконтроллер AVR?
5. Зачем нужны команды передачи управления?
6. Как организована архитектура ядра микроконтроллеров AVR семейства Mega?

Список литературы, рекомендуемый к использованию по данной теме:

11. Клингман Э. Проектирование микропроцессорных систем. М.: Мир. 1988. - 575с.
12. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристалльных микроконтроллерах.
13. Трамперт В. AVR-RISK микроконтроллеры.: Пер. с нем. – К.: «МК-ПРЕСС», 2006.-464с., ил.
14. 4. Ю.А.Шпак. Программирование на языке С для AVR и PIC микроконтроллеров. – К.: «МК-ПРЕСС», 2006.-400с., ил.
15. Якубовский С.В. Цифровые и аналоговые интегральные микросхемы. М.:Радио и связь. 1990.
16. Токхейм Р. Основы цифровой электроники. Пер. с англ. - М.: Мир, 1988. – 390 с.
17. Шило В.Л. Популярныe цифровые микросхемы. – М.: Радио и связь, 1990.– 350 с.
18. Бирюков С.А. Цифровые устройства на МОП-интегральных микросхемах - М.: Радио и связь. 1992 (1996).
19. Опачий Ю.Ф., Глудкин О.П., Гуров А.И. Аналоговая и цифровая электроника. Полный курс: учебник для вузов. - М.: Горячая линия, 1999 (2000, 2005). – 768 с.
20. Алексенко А.В., Шагуров И.И. Микросхемотехника.– М.: Радио и связь,1990 (1982).

Лабораторная работа 4

Изучение работы и программирования АЦП и ЦАП блоков ШИМ микроконтроллера AVR

Цель работы

Изучить работу АЦП и ЦАП блоков ШИМ микроконтроллера AVR Atmega8535, Atmega16, изучить основные приёмы программирования и отладки микроконтроллера..

Компетенции:

Индекс	Формулировка:
ОПК-3	способностью решать задачи анализа и расчета характеристик электрических цепей
ОПК-7	способностью учитывать современные тенденции развития теории автоматического управления, электронного оборудования, измерительной и вычислительной техники, информационных технологий в профессиональной деятельности
ПК-1	способностью выполнять эксперименты на действующих объектах по заданным методикам, оценивать динамические характеристики рассматриваемых объектов и идентифицировать передаточные функции объектов с применением современных информационных технологий и технических средств
ПК-7	готовностью участвовать в разработке и изготовлении стендов для комплексной отладки и испытаний программно-аппаратных управляющих комплексов
ПК-8	готовностью к внедрению результатов разработок средств и систем автоматизации и управления в производстве с использованием современных программируемых логических контроллеров (ПЛК)
ПК-11	готовностью производить инсталляцию и настройку системного, прикладного и инструментального программного обеспечения систем автоматизации и управления
ПК-14	способностью разрабатывать электромеханические системы и использовать современную элементную базу при проектировании средств и систем управления

Теоретическая часть

Общие сведения

Модуль 10-битного АЦП последовательного приближения входит в состав практически всех моделей семейства, за исключением ATmega8515x и ATmega162x. Основные параметры этого АЦП следующие:

- абсолютная погрешность: $\pm 2 \text{ LSB}^1$;
- интегральная нелинейность: $\pm 0.5 \text{ LSB}$;
- быстродействие: до 15 тыс. выборок/с.

На входе модуля АЦП имеется 8-канальный (в моделях ATmega640x/1280x/2560x — 16-канальный) аналоговый мультиплексор, предоставляющий в распоряжение пользователя 8 (16) каналов с несимметричными входами. Кроме того, в моделях ATmega8x, выпускаемых в корпусе DIP, доступно только 6 каналов из восьми.

В большинстве моделей входы АЦП могут объединяться попарно для формирования различного числа каналов с дифференциальным входом. При этом в некоторых каналах имеется возможность 10- и 200-кратного предварительного усиления входного сигнала. При коэффициентах усиления 1x и 10x действительная разрешающая способность АЦП по этим каналам составляет 8 бит, а при коэффициенте усиления 200x — 7 бит.

В качестве источника опорного напряжения для АЦП может использоваться как напряжение питания микроконтроллера, так и внутренний либо внешний источник опорного напряжения.

Модуль АЦП может работать в двух режимах:

- режим одиночного преобразования, когда запуск каждого преобразования инициируется пользователем;
- режим непрерывного преобразования, когда запуск преобразований выполняется непрерывно через определенные интервалы времени.

Функционирование модуля АЦП

Обобщенная структурная схема модуля АЦП приведена на Рис. 9.1. В моделях ATmega8x и ATmega48x/88x/168x элементы, выделенные на рисунке серым цветом, и связанные с ними сигналы отсутствуют, а неинвертирующий вход компаратора выборки-хранения подключен непосредственно к выходу мультиплексора (показано пунктиром).

Регистры, используемые для управления модулем АЦП в различных моделях, приведены в Табл. 9.1.

Формат регистров ADCSRA (ADCSR) и ADMUX приведен на Рис. 9.2 и Рис. 9.3, а краткое описание функций их битов приведено в Табл. 9.2 и Табл. 9.3 соответственно.

Таблица 9.1. Регистры управления модулем АЦП

Регистр	Адрес	Модели процессора											Описание	
		ATmega8535x	ATmega8x	ATmega16x/32x	ATmega64x	ATmega128x	ATmega48x/88x/168x	ATmega164x/324x/644x	ATmega165x	ATmega325x/3250x, ATmega645x/6450x	ATmega640x, ATmega1280x/1281x, ATmega2560x/2561x			
ADCSR	\$06 (\$26)		•											Регистр управления и состояния
ADCSRA	\$06 (\$26)	•		•	•	•								Регистр А управления и состояния
	(\$7A)						•	•	•	•	•			
ADCSRB	(\$8E)				•									Регистр В управления и состояния
	(\$7B)						•	•	•	•	•			
ADMUX	\$07 (\$27)	•	•	•	•	•								Регистр управления мультиплексором
	(\$7C)						•	•	•	•	•			
SFIOR	\$30 (\$50)	•	•	•										Регистр специальных функций
	\$20 (\$40)					•								

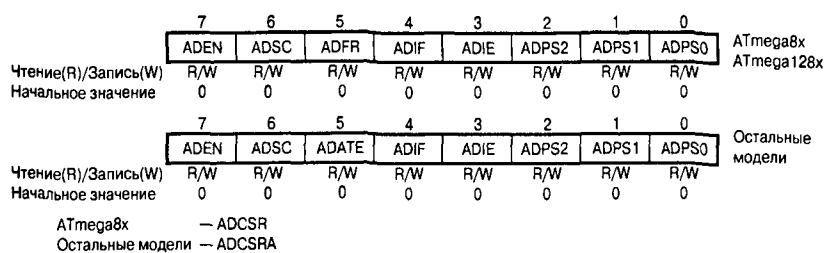


Рис. 9.2. Формат регистра ADCSRA (ADCSR)

Таблица 9.2. Биты регистра ADCSRA (ADCSR¹⁾)

Бит	Название	Описание
7	ADEN	Разрешение АЦП (1 — включено, 0 — выключено)
6	ADSC	Запуск преобразования (1 — начать преобразование)
5	ADATE (ADFR ²⁾)	Выбор режима работы АЦП
4	ADIF	Флаг прерывания от компаратора
3	ADIE	Разрешение прерывания от компаратора
2...0	ADPS2:ADPS0	Выбор частоты преобразования

¹⁾ В модели ATmega8x.
²⁾ В моделях ATmega8x и ATmega128x.

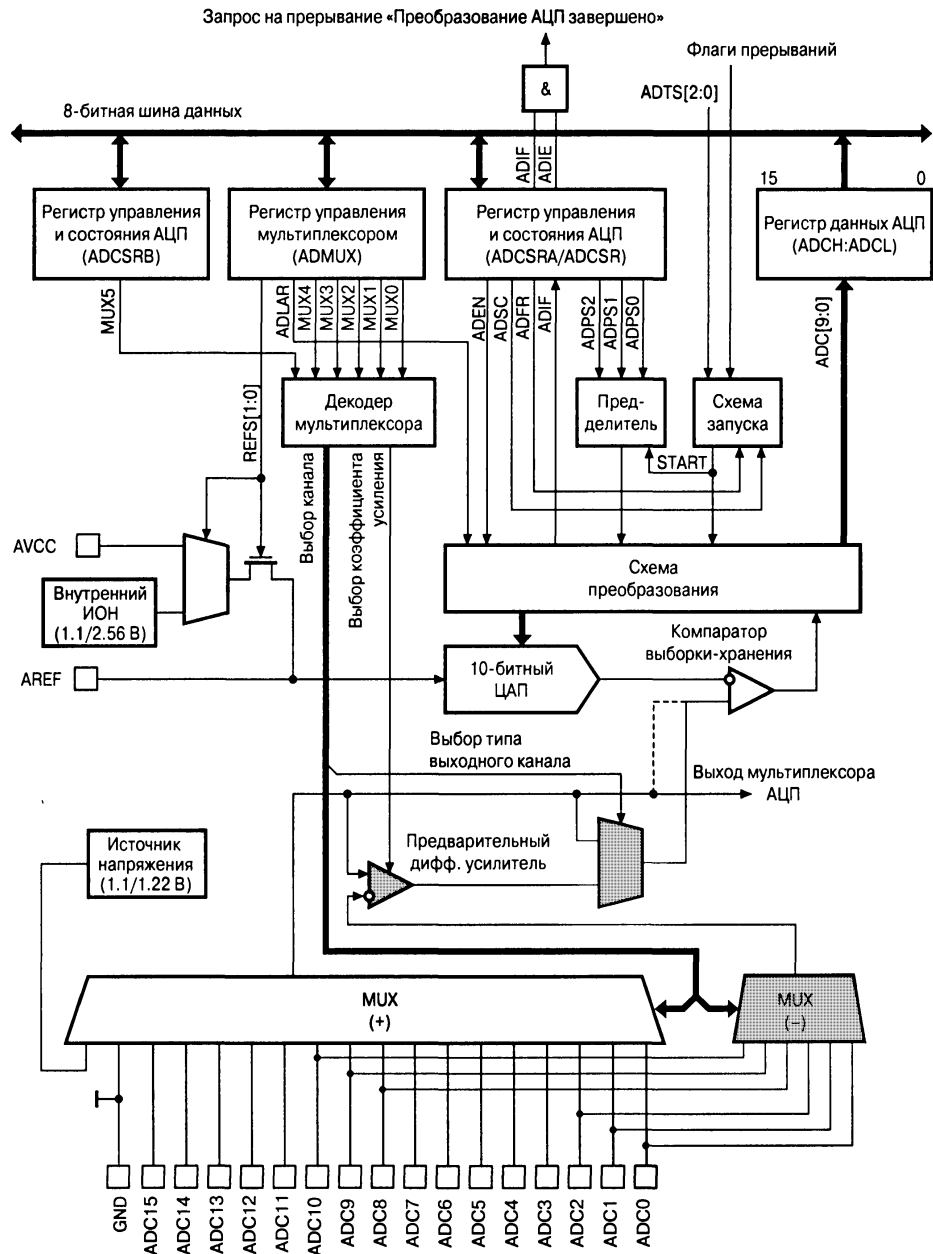


Рис. 9.1. Структурная схема модуля АЦП

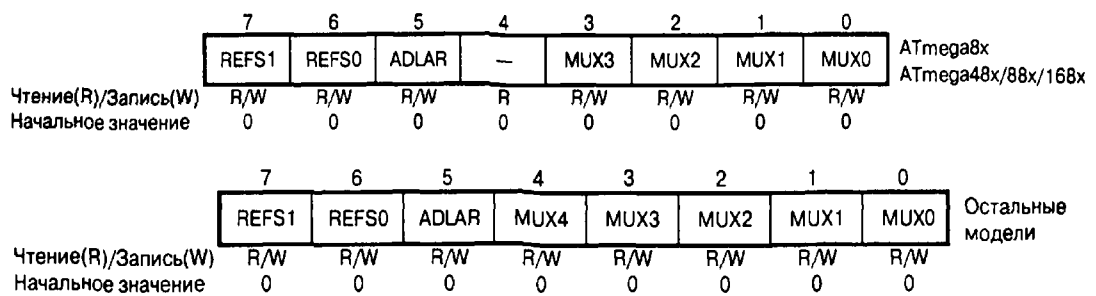


Рис. 9.3. Формат регистра ADMUX

Таблица 9.3. Биты регистра ADMUX

Бит	Название	Описание	Модель
7, 6	REFS1:REFS0	Выбор источника опорного напряжения	Все модели
5	ADLAR	Выравнивание результата преобразования	Все модели
4	—	Зарезервировано	ATmega8x, ATmega48x/88x/168x
	MUX4	Выбор входного канала	Остальные
3...0	MUX3...MUX0	Выбор входного канала	Все модели

Формат регистров ADCSRB и SFIOR приведен на **Рис. 9.4** и **Рис. 9.5** соответственно (неиспользуемые в данном случае биты регистра SFIOR указаны на рисунке как «X»).

Для разрешения работы АЦП необходимо записать лог. 1 в бит ADEN регистра ADCSR, а для выключения — соответственно лог. 0. Если АЦП будет выключен во время цикла преобразования, то преобразование завершено не будет (в регистре данных АЦП останется результат предыдущего преобразования).

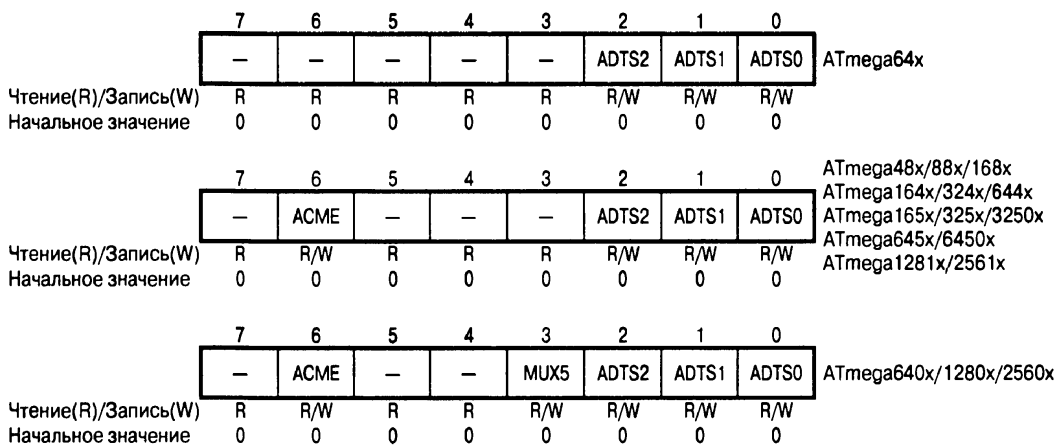


Рис. 9.4. Формат регистра ADCSRB

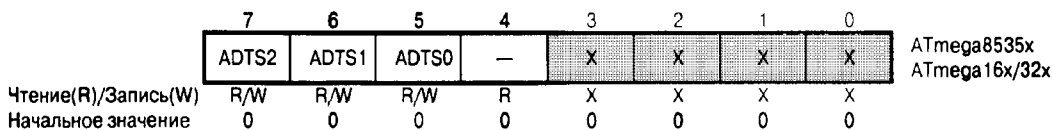


Рис. 9.5. Регистр SFIOR

В моделях ATmega8x и ATmega128x режим работы АЦП определяется состоянием бита ADFR. Если он установлен в 1, АЦП работает в режиме непрерывного преобразования. В этом режиме запуск каждого следующего преобразования осуществляется автоматически после окончания текущего. Если же бит ADFR сброшен в 0, АЦП работает в режиме одиночного преобразования и запуск каждого преобразования осуществляется по команде пользователя.

В моделях ATmega8x и ATmega128x режим работы АЦП определяется состоянием бита ADFR. Если он установлен в 1, АЦП работает в режиме непрерывного преобразования. В этом режиме запуск каждого следующего преобразования осуществляется автоматически после окончания текущего. Если же бит ADFR сброшен в 0, АЦП работает в режиме одиночного преобразования и запуск каждого преобразования осуществляется по команде пользователя.

Если бит ADATE сброшен в 0, АЦП работает в режиме одиночного преобразования. Если же бит ADTAE установлен в 1, функционирование АЦП определяется содержимым битов ADTS2:0 согласно Табл. 9.4.

Таблица 9.4. Источник сигнала для запуска преобразования

ADTS2	ADTS1	ADTS0	Источник стартового сигнала
0	0	0	Режим непрерывного преобразования
0	0	1	Прерывание от аналогового компаратора
0	1	0	Внешнее прерывание INT0
0	1	1	Прерывание по событию «Совпадение» («Совпадение А») таймера/счетчика T0
1	0	0	Прерывание по переполнению таймера/счетчика T0
1	0	1	Прерывание по событию «Совпадение В» таймера/счетчика T1
1	1	0	Прерывание по переполнению таймера/счетчика T1
1	1	1	Прерывание по событию «Захват» таймера/счетчика T1

Запуск каждого преобразования в режиме одиночного преобразования, а также запуск первого преобразования в режиме непрерывного преобразования осуществляется установкой в 1 бита ADSC регистра ADCSRA (ADCSR). Запуск преобразования по прерыванию осуществляется при установке в 1 флага выбранного прерывания. Бит ADSC регистра ADCSRA при этом аппаратно устанавливается в 1. Запуск преобразования в этих режимах также может быть осуществлен установкой бита ADSC регистра ADCSRA в 1.

В режимах одиночного и непрерывного преобразований цикл преобразования начинается по первому нарастающему фронту тактового сигнала после установки бита ADSC. Если используется запуск по прерыванию, то цикл преобразования начинается по первому нарастающему фронту тактового сигнала после установки флага выбранного прерывания. Причем в момент установки этого флага осуществляется сброс предделителя модуля АЦП. Тем самым обеспечивается фиксированная задержка между генерацией запроса на прерывание и началом цикла преобразования. Обратите внимание, что преобразование запускается при установке соответствующего флага, т. е. даже если само прерывание запрещено.

Длительность цикла составляет 13 тактов при использовании несимметричного входа и 13 либо 14 тактов при использовании дифференциального входа (разные значения связаны с работой схемы синхронизации); выборка и запоминание входного сигнала осуществляется в течение первых 1.5 и 2.5 тактов соответственно. Через 13 (14) тактов преобразование

завершается, бит ADSC аппаратно сбрасывается в 0 (в режиме одиночного преобразования), и результат преобразования сохраняется в регистре данных АЦП. Одновременно устанавливается флаг прерывания ADIF регистра ADCSR и генерируется запрос на прерывание. Как и флаги остальных прерываний, флаг ADIF сбрасывается аппаратно при запуске подпрограммы обработки прерывания от АЦП или программно, записью в него лог. 1. Разрешение прерывания осуществляется установкой в 1 бита ADIE регистра ADCSR при установленном флаге I регистра SREG.

Если АЦП работает в режиме непрерывного преобразования, то новый цикл начнется сразу же после записи результата. В режиме одиночного преобразования новое преобразование может быть запущено сразу же после сброса бита ADSC (до сохранения результата текущего преобразования). Однако реально цикл преобразования начнется не ранее чем через один такт после окончания текущего преобразования. Временные диаграммы, иллюстрирующие сказанное, приведены на Рис. 9.6.

При первом запуске после включения АЦП для выполнения преобразования потребуется 25 тактов, т. е. на 12 тактов больше, чем обычно. В течение этих 12 тактов выполняется «холостое» преобразование, во время которого производится инициализация АЦП (Рис. 9.7).

Отдельно следует сказать об использовании режима запуска по прерыванию совместно с дифференциальными каналами. В этом случае АЦП необходимо выключать между преобразованиями, чтобы избежать некорректных измерений, связанных с неопределенностью момента сброса делителя АЦП. В результате выключения и включения АЦП между преобразованиями будут выполняться только «длинные» преобразования, результаты которых всегда будут корректными.

Для формирования тактовой частоты модуля АЦП в нем имеется отдельный делитель. Коэффициент деления делителя и соответственно длительность преобразования определяются состоянием битов ADPS2...ADPS0 регистра ADCSRA/ADCSR (см. Табл. 9.5).

Наибольшая точность преобразования достигается, если тактовая частота модуля АЦП находится в диапазоне 50...200 кГц. Соответственно, коэффициент деления делителя рекомендуется выбирать таким, чтобы тактовая частота модуля АЦП находилась в указанном диапазоне. Если же точности преобразования меньше 10 битов достаточно, можно использовать более высокую частоту, увеличивая тем самым частоту выборки.

В зависимости от модели выводы микроконтроллера, подключенные к входу АЦП, определяются состоянием битов MUX3:0, MUX4:0 либо MUX5:0 (MUX5 — в ADCSRB, MUX4...MUX0 — в ADMUX). Для каналов с дифференциальным входом указанные биты определяют также коэффициент предварительного усиления входного сигнала. Соответствие установок этих битов используемым входам АЦП приведено в Табл. 9.6...9.8.

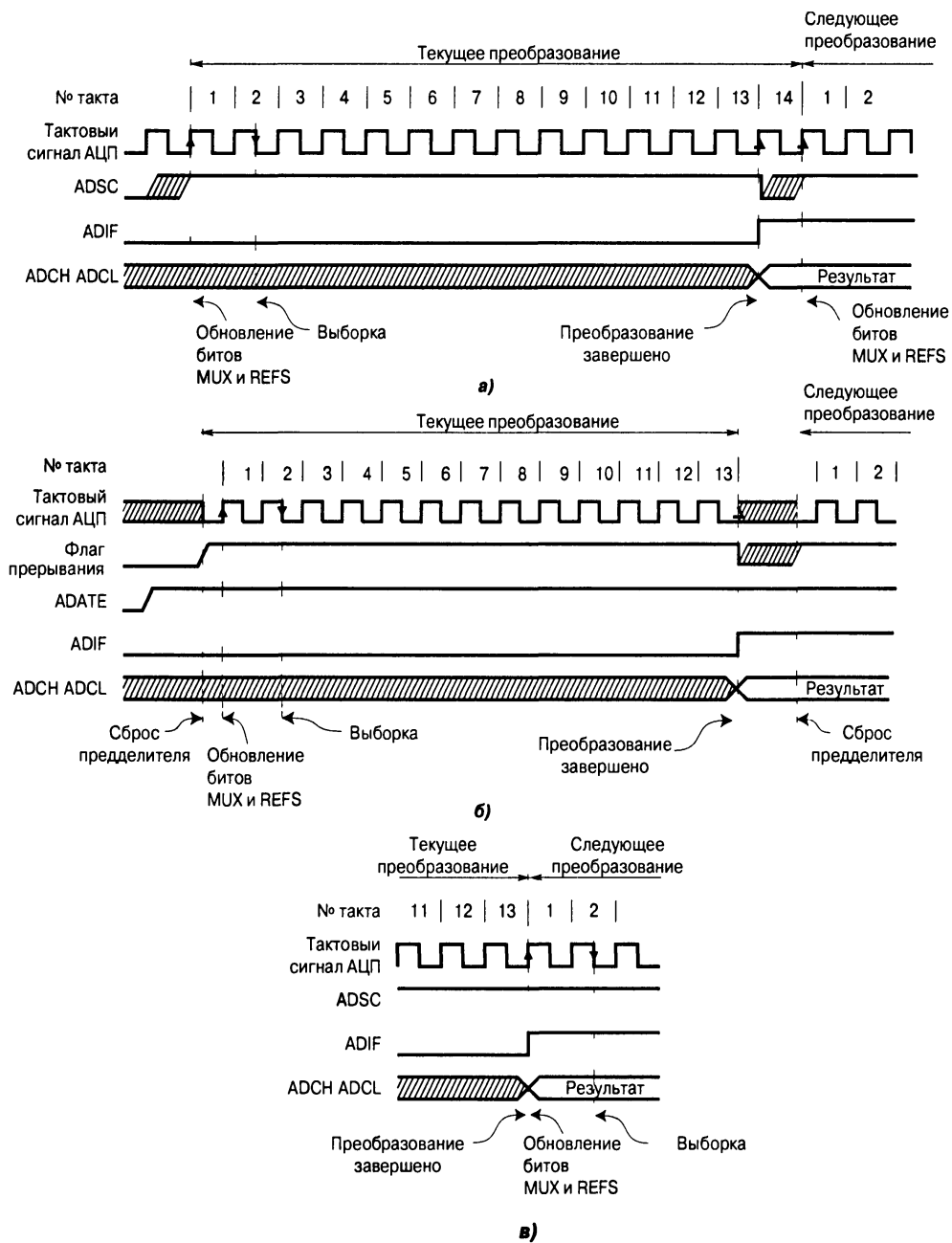


Рис. 9.6. Временные диаграммы работы АЦП в режиме одиночного преобразования (а), в режиме запуска по прерыванию (б) и в режиме непрерывного преобразования (в)

Таблица 9.5. Задание коэффициента деления предделителя АЦП

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

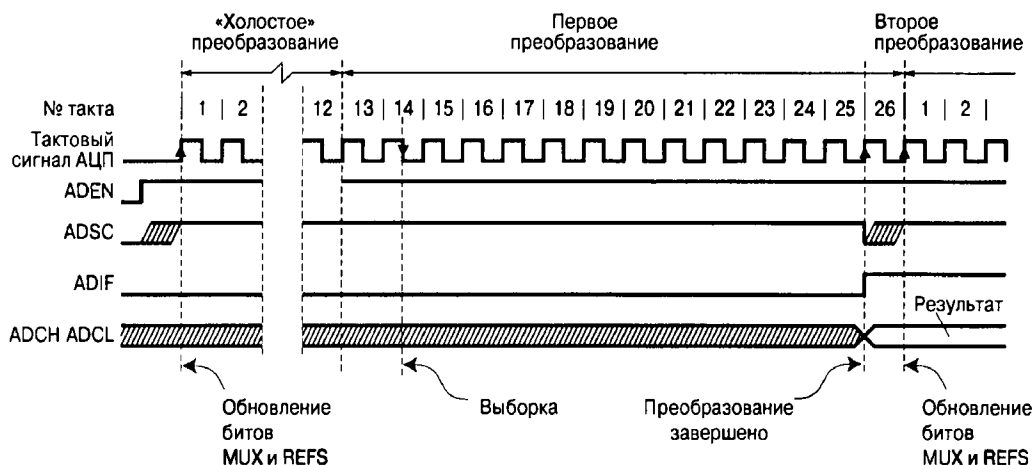


Рис. 9.7. Временные диаграммы работы АЦП при первом преобразовании (режим одиночного преобразования)

Таблица 9.6. Управление входным мультиплексором в моделях АТmega8х и АТmega48х/88х/168х

MUX3...MUX0	Несимметричный вход	MUX3...MUX0	Несимметричный вход
0000	ADC0	0110	ADC6 ¹⁾
0001	ADC1	0111	ADC7 ¹⁾
0010	ADC2	1000...1101	Зарезервировано
0011	ADC3	1110	1.22 В (1.1 В ²⁾)
0100	ADC4	1111	0 В (GND)
0101	ADC5		

¹⁾ Только в корпусах TQFP-32 и MLF-32.
²⁾ В моделях АТmega48х/88х/168х.

Следует отметить, что предварительный усилитель, используемый каналами с дифференциальным входом, имеет встроенную схему коррекции смещения. Оставшаяся после коррекции величина смещения может быть учтена программным способом. Для этого следует оба входа дифференциального усилителя подключить к одному и тому же выводу микроконтроллера (Табл. 9.7 и Табл. 9.8), а затем вычитать полученное значение из результата последующих преобразований. Таким образом, ошибка смещения может быть снижена до величины, меньшей 1 LSB.

Состояние битов MUX5...MUX0 можно изменить в любой момент, однако если это будет сделано во время цикла преобразования, то смена канала произойдет только после завершения преобразования. Благодаря этому в режиме непрерывного преобразования можно легко осуществлять последовательное преобразование сигналов нескольких каналов.

Отдельно следует сказать о каналах с дифференциальным входом. После смены таких каналов первое измерение следует производить не ранее чем через 125 мкс после выбора канала. Указанное время требуется для установления значения коэффициента усиления предусилителя. Соответственно, значения, измеренные до истечения этого срока, не могут считаться достоверными. Кроме того, при переключении на канал с изменяемым коэффициентом усиления результат первого преобразования может

иметь пониженную точность из-за времени установления автоматической схемы коррекции смещения. Поэтому будет лучше считать результат первого преобразования некорректным.

В новых микроконтроллерах, а именно в ATmega48x/88x/168x, ATmega164x/324x/644x, ATmega165x, ATmega325x/3250x/645x/6450x и ATmega640x/1280x/1281x/2560x/2561x, имеется возможность отключения входных цифровых буферов на выводах ADC0...ADC15 в случае, если соответствующие выводы используются только для считывания аналоговых сигналов. При отключенных цифровых буферах уменьшается общий ток потребления микроконтроллера, а соответствующие биты регистров PINx всегда читаются как 0.

Отключение цифровых буферов на входах ADC0...ADC7 осуществляется записью лог. 1 соответственно в биты ADC0D...ADC7D регистра DIDR0, расположенного по адресу (\$7E). А отключение буферов на входах ADC8...ADC15 (модели ATmega640x/1280x/2560x) осуществляется записью лог. 1 в биты ADC8D...ADC15D регистра DIDR2, расположенного по адресу (\$7D). Формат этих регистров приведен на Рис. 9.8.

Как уже было отмечено, в модуле АЦП могут использоваться различные источники опорного напряжения (ИОН). Выбор конкретного источника опорного напряжения осуществляется с помощью битов REFS1:REFS0 регистра ADMUX (см. Табл. 9.9).

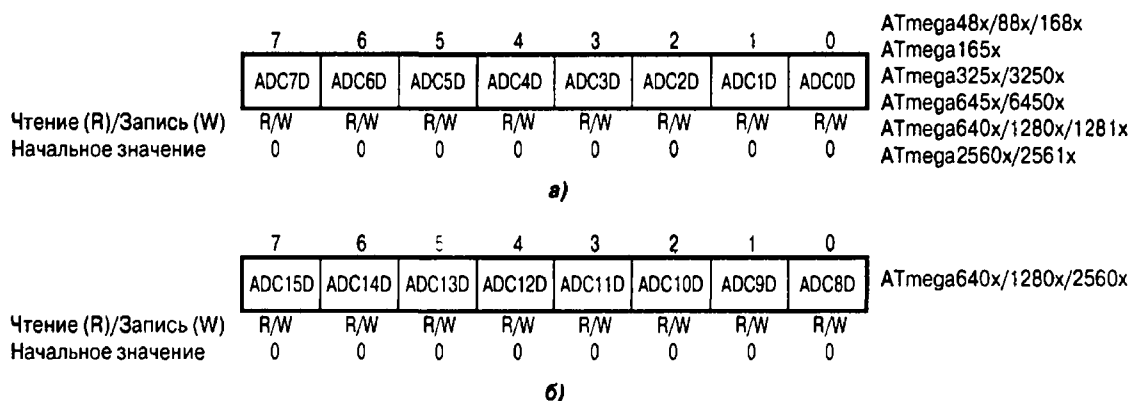


Рис. 9.8. Формат регистров DIDR0 (а) и DIDR2 (б)

Поскольку внутренний ИОН соединяется с выводом AREF микроконтроллера, при его использовании к выводу AREF желательно подключить внешний фильтрующий конденсатор для повышения помехозащищенности. Кроме того, при использовании канала с дифференциальным входом первое измерение после смены источника опорного напряжения следует производить не ранее чем через 125 мкс. Указанное время требуется для установления значения коэффициента усиления предусилителя.

Таблица 9.9. Выбор источника опорного напряжения

REFS1	REFS0	Источник опорного напряжения ¹⁾	Модель
0	0	Внешний ИОН, подключенный к выводу AREF; внутренний ИОН отключен	Все модели
0	1	Напряжение питания V_{CC}	Все модели
1	0	Внутренний ИОН напряжением 1.1 В ²⁾	ATmega164x/324x/644x, ATmega640x/1280x/1281x, ATmega2560x/2561x
		Зарезервировано	Остальные модели
1	1	Внутренний ИОН напряжением 1.1 В ²⁾	ATmega48x/88x/168x, ATmega165x/325x/3250x, ATmega645x/6450x
		Внутренний ИОН напряжением 2.56 В ²⁾	Остальные модели
¹⁾ При работе с усилением 10x или 200x в качестве внутреннего ИОН можно использовать только ИОН, подключаемый при REFS1:0 = 11. ²⁾ Если к выводу AREF подключен источник напряжения, то эти варианты использоваться не могут.			

9.3. Результат преобразования

После завершения преобразования (при установке в 1 флага ADIF регистра ADCSR) его результат сохраняется в регистре данных АЦП. Поскольку АЦП — 10-битный, этот регистр физически размещен в двух регистрах ввода/вывода ADCH:ADCL, доступных только для чтения. Эти регистры расположены по адресам \$05:\$04 (\$25:\$24) в моделях ATmega8535x, ATmega8x/16x/32x/64x/128x и (\$79:\$78) в остальных моделях. При включении микроконтроллера в регистре данных АЦП содержится значение \$0000.

По умолчанию результат преобразования выравнивается вправо (старшие 6 битов регистра ADCH — незначащие). Однако он может выравниваться и влево (младшие 6 битов регистра ADCL — незначащие). Для управления выравниванием результата преобразования служит бит ADLAR регистра ADMUX. Если этот бит установлен в 1, результат преобразования выравнивается по левой границе 16-битного слова, если сброшен в 0 — по правой границе.

Обращение к регистрам ADCH и ADCL для получения результата преобразования должно выполняться в определенной последовательности: сначала необходимо прочитать регистр ADCL, а затем ADCH. Это требование связано с тем, что после обращения к регистру ADCL процессор блокирует доступ к регистрам данных со стороны АЦП до тех пор, пока не будет прочитан регистр ADCH. Благодаря этому можно быть уверенным, что при чтении регистров в них будут находиться составляющие одного и того же результата. Соответственно, если очередное преобразование завершится до обращения к регистру ADCH, результат преобразования будет потерян. С другой стороны, если результат преобразования выравнивается влево и достаточно 8-битной точности, то для получения результата можно прочитать только содержимое регистра ADCH.

Для каналов с несимметричным входом результат преобразования определяется выражением

$$ADC = \frac{V_{IN} \cdot 1024}{V_{REF}},$$

где V_{IN} — значение входного напряжения, а V_{REF} — величина опорного напряжения.

Для каналов с дифференциальным входом результат преобразования определяется выражением

$$ADC = \frac{(V_{POS} - V_{NEG}) \cdot K \cdot 512}{V_{REF}},$$

где V_{POS} — величина напряжения на положительном входе, V_{NEG} — величина напряжения на отрицательном входе, а K — коэффициент усиления. Результат преобразования представляется в этом случае в дополнительном коде, а его значение лежит в диапазоне от $\$200$ (-512) до $\$1FF$ ($+512$).

9.4. Повышение точности преобразования

В этом разделе приведены некоторые рекомендации, позволяющие в наибольшей степени использовать возможности АЦП. Прежде всего, для минимизации погрешности самого АЦП необходимо правильно выбрать тактовую частоту преобразования. С этой же целью на входе АЦП рекомендуется устанавливать фильтр нижних частот. Кроме того, при разработке конструкции и разводке печатной платы следует придерживаться общих правил проектирования цифро-аналоговых устройств:

1. На печатной плате необходимо предусмотреть область сплошной металлизации под аналоговую «землю». Аналоговая часть микроконтроллера и аналоговая часть всего устройства должна располагаться над этой областью. Аналоговая и цифровая «земли» должны соединяться друг с другом только в одной точке печатной платы.
2. Проводники, по которым распространяются аналоговые сигналы, должны быть как можно короче и располагаться над аналоговой «землей». Кроме того, они должны быть размещены как можно дальше от быстродействующих цифровых цепей.
3. Вывод AVCC микроконтроллера должен подключаться к источнику питания V_{CC} через LC-фильтр, как показано на Рис. 9.9 (расположение выводов показано условно).
4. Если какие-либо выводы АЦП используются как цифровые выходы, они не должны переключаться во время преобразования.

Для сведения к минимуму электромагнитных помех, наводимых ядром процессора, во всех рассматриваемых микроконтроллерах имеется дополнительный «спящий» режим — ADC Noise Reduction (режим снижения шумов АЦП). В этом режиме из всех периферийных устройств функционируют только АЦП и сторожевой таймер. Для той же цели (но с меньшим эффектом) может быть использован режим Idle. Для использования АЦП в любом из указанных режимов необходимо выполнить следующее:

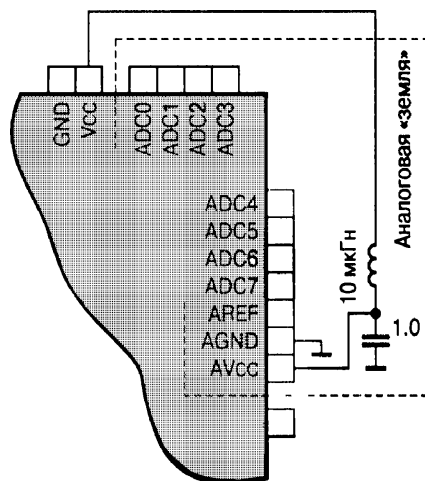


Рис. 9.9. Подключение цепей питания АЦП

1. Убедиться, что АЦП включен и не занят преобразованием. Затем переключить АЦП в режим одиночного преобразования и разрешить прерывание от АЦП.
2. Перевести микроконтроллер в режим ADC Noise Reduction (или Idle). Сразу же после остановки процессора начнется цикл преобразования.
3. По завершении преобразования будет сгенерировано прерывание от АЦП, которое переведет микроконтроллер в рабочий режим и начнется выполнение подпрограммы обработки этого прерывания.

Оборудование и материалы.

Комплект лабораторных модулей микропроцессорная техника РТМТЛ. Принципиальная схема лабораторного стенда приведена ниже лабораторных работ по исследованию программирования микроконтроллеров на различных языках программирования выполняется на учебном макетном лабораторном комплексе РТМТЛ.

На передней крышке прибора закреплена отладочная плата, содержащая исследуемый микроконтроллер (Atmega8535, Atmega16, Atmega32 или совместимый); резистивно-диодный последовательный (работающий по интерфейсу RS232 через COM-порт ПК) программатор; обвязку микроконтроллера согласно паспортным данным; набор из 8 светодиодов, подключенных к выводам 33 – 40, см. рис. 2.1; набор из 8 кнопок с фиксацией нормально разомкнутых (кнопка нажата — замкнута; отжата — разомкнута), подключенных к выводам 22 — 29; а также кнопку с фиксацией «RESET», при нажатии которой вывод RESET кристалла соединяется с корпусом схемы. Для программирования прибора используется разъем «ПРОГРАММАТОР». При этом «ПРОГРАММАТОР» следует подключать к COM – порту ПЭВМ ТОЛЬКО проводом типа COM 9m/9f («мама» - «папа»). В данной работе на примере микроконтроллера AVR (Atmega8535, Atmega16, Atmega32 или совместимого) предлагается освоить простейшие приемы программирования на следующих языках:

- 1) Программирование микроконтроллера на языке ассемблер в интегральной среде разработки AVR Studio
- 2) Программирование микроконтроллера на языке Basic в интегральной среде разработки Vascom-AVR
- 3) Программирование микроконтроллера на языке Си в интегральной среде разработки AVR Studio с подключенным Си компилятором GCC из WinAVR
- 4) Программирование микроконтроллера в среде графического ассемблера Algorithm Builder.

Естественно, что в данном кратком методическом руководстве невозможно описать все тонкости работы с той или иной средой программирования или с тем или иным языком программирования. Поэтому, перед началом работы необходимо ознакомиться с документацией на исследуемый микроконтроллер, документацией на исследуемую среду программирования, а также со статьями по данным вопросам, размещенными в соответствующих папках на прилагаемом к учебной установке диске или Flash карте.

Принципиальная электрическая схема прибора приведена на рис. 2.1. Демонстрационный пример готовой программы, на основе которого могут быть построены задания по программированию микроконтроллера, приведены в папке Examples_RTMTL-4. Пример выполнен на четырех распространенных языках программирования для микроконтроллера (ASM, Си, Basic, Графический ASM в среде Algorithm Builder).

Задание 1. В данном примере кнопки с фиксацией подключены к порту А (выводы PA0-PA7), логическое состояние которого представляет собой 1 байт данных. При этом состояние нажатой кнопки это логический 0, состояние отжатой кнопки — логическая 1 соответствующего вывода. К данным порта А прибавляется константа (число 2 в среде графического ассемблера Algorithm Builder; число 3 в среде разработки AVR Studio ассемблер; число 4 в среде разработки Bascom-AVR и число 5 в среде разработки AVR Studio Си с подключенным компилятором GCC из WinAVR). Полученный результат копируется в порт С (PC0-PC7), к которому подключены светодиоды, визуализирующие логическое состояние соответствующих выводов (PC0-PC7).-

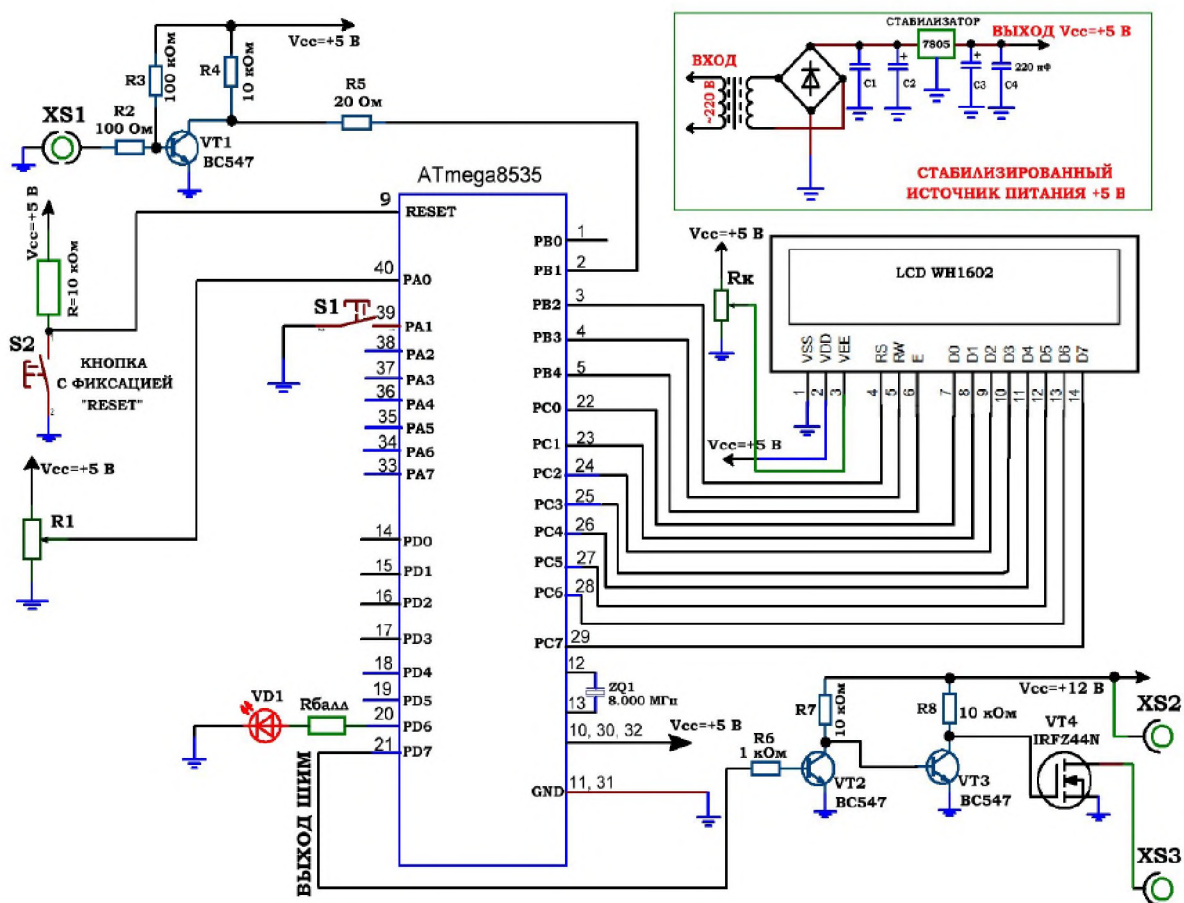


Рис. 3.1. Принципиальная электрическая схема учебного стенда для изучения микроконтроллеров серии Atmega8535.

Указания по технике безопасности

Соответствуют технике безопасности по работе с компьютерной техникой.

Ход работы

1. Перед включением установки в сеть проверить целостность всех соединительных сигнальных и сетевых проводов. Все работы по подключению комплекса к компьютеру следует выполнять только при отключенных от сети приборах. Разобраться с принципиальными блок-схемами опытов, в назначении кнопок, переключателей и ручек прибора.

2. Перед выполнением работы следует изучить инструкцию по работе с учебной средой программирования Algorithm Builder, а также ознакомиться с полным методическим руководством по микроконтроллерам семейства AVR.

3. Соединить монитор с системным блоком ПЭВМ, подключить клавиатуру и мышь к системному блоку используя стандартные провода для подключения. Подключить системный блок ПЭВМ и монитор к сети ~220 В.

4. Загрузить операционную систему согласно стандартным процедурам загрузки.

5. При необходимости, настроить компьютер для работы с учебной установкой и программной средой Algorithm Builder.

6. Запустить программу Algorithm Builder для работы с учебной установкой для данного эксперимента пользуясь ярлыком на рабочем столе либо другим способом, указанным лаборантом. Для работы можно воспользоваться комплексной программной оболочкой LabVisual для РТМТЛ-1-5.

6. Подключить разъем «ПРОГРАММАТОР» учебного прибора к СОМ – порту ПЭВМ проводом типа СОМ 9m/9f («мама» - «папа»)

7. Настроить программу Algorithm Builder для работы с данным СОМ портом
ОПЦИИ-ОПЦИИ СРЕДЫ-ПОРТ

Задание . В данном примере кнопки с фиксацией подключены к порту А (выводы РА0-РА7), логическое состояние которого представляет собой 1 байт данных. При этом состояние нажатой кнопки это логический 0, состояние отжатой кнопки — логическая 1 соответствующего вывода. К данным порта А прибавляется константа (число 2 в среде графического ассемблера Algorithm Builder; число 3 в среде разработки AVR Studio ассемблер; число 4 в среде разработки Bascom-AVR и число 5 в среде разработки AVR Studio Си с подключенным компилятором GCC из WinAVR). Полученный результат копируется в порт С (РС0-РС7), к которому подключены светодиоды, визуализирующие логическое состояние соответствующих выводов (РС0-РС7).

Оформление отчета

В отчете должны содержаться описание команд использованных в выполнении заданий и программы написанные по заданным в задании алгоритмам

Контрольные вопросы

1. Какой уровень сигнала у цифровых выходов МК?
2. Какие типы команд имеет микроконтроллер AVR семейства Mega?
3. Какие типы команд изменяют флаги микроконтроллера?
4. Какие виды адресации использует микроконтроллер AVR?
5. Зачем нужны команды передачи управления?
6. Как организована архитектура ядра микроконтроллеров AVR семейства Mega?

Список литературы, рекомендуемый к использованию по данной теме:

1. Клингман Э. Проектирование микропроцессорных систем. М.: Мир. 1988. - 575с.

2. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах.
3. Трамперт В. AVR-RISK микроконтроллеры.: Пер. с нем. – К.: «МК-ПРЕСС», 2006.-464с., ил.
4. Ю.А.Шпак. Программирование на языке С для AVR и PIC микроконтроллеров. – К.: «МК-ПРЕСС», 2006.-400с., ил.
5. Якубовский С.В. Цифровые и аналоговые интегральные микросхемы. М.:Радио и связь. 1990.
6. Токхейм Р. Основы цифровой электроники. Пер. с англ. - М.: Мир, 1988. – 390 с.
7. Шило В.Л. Популярныe цифровые микросхемы. – М.: Радио и связь, 1990.– 350 с.
8. Бирюков С.А. Цифровые устройства на МОП-интегральных микросхемах - М.: Радио и связь. 1992 (1996).
9. Опадчий Ю.Ф., Глудкин О.П., Гуров А.И. Аналоговая и цифровая электроника. Полный курс: учебник для вузов. - М.: Горячая линия, 1999 (2000, 2005). – 768 с.
10. Алексенко А.В., Шагуров И.И. Микросхемотехника.– М.: Радио и связь,1990 (1982).

Лабораторная работа 5

Изучение работы и программирования блоков последовательной передачи данных микроконтроллера AVR с помощью инструментальных средств

Цель работы

Изучение работы и программирования блоков последовательной передачи данных микроконтроллера AVR с помощью инструментальных средств AVR Atmega8535, Atmega16, Atmega32 и совместимого изучить основные приёмы программирования и отладки микроконтроллера.

Компетенции:

Индекс	Формулировка:
ОПК-3	способностью решать задачи анализа и расчета характеристик электрических цепей
ОПК-7	способностью учитывать современные тенденции развития теории автоматического управления, электронного оборудования, измерительной и вычислительной техники, информационных технологий в профессиональной деятельности
ПК-1	способностью выполнять эксперименты на действующих объектах по заданным методикам, оценивать динамические характеристики рассматриваемых объектов и идентифицировать передаточные функции объектов с применением современных информационных технологий и технических средств
ПК-7	готовностью участвовать в разработке и изготовлении стендов для комплексной отладки и испытаний программно-аппаратных управляющих комплексов
ПК-8	готовностью к внедрению результатов разработок средств и систем автоматизации и управления в производстве с использованием современных программируемых логических контроллеров (ПЛК)
ПК-11	готовностью производить инсталляцию и настройку системного, прикладного и инструментального программного обеспечения систем автоматизации и управления
ПК-14	способностью разрабатывать электромеханические системы и использовать современную элементную базу при проектировании средств и систем управления

Теоретическая часть

Общие сведения

Модуль универсального последовательного интерфейса (Universal Serial Interface — USI) реализован только в моделях ATmega165x и ATmega325x/3250x/645x/6450x. Этот модуль является своего рода «полу-фабрикатом», предоставляющим базовые аппаратные ресурсы, необходимые для осуществления обмена по последовательному каналу. Используя данный модуль, можно достичь гораздо большей скорости передачи и получить более компактный код, нежели при чисто программной реализации различных протоколов обмена.

Обращаю ваше внимание на то, что в данной главе мы будем рассматривать собственно модуль USI. Принципы обмена по трехпроводной (SPI) и двухпроводной (TWI) шинам были подробно рассмотрены в двух предыдущих главах.

Упрощенная структурная схема модуля USI приведена на **Рис. 12.1**.

В общей сложности модуль USI задействует три линии ввода/вывода микроконтроллера:

- PE6 — выход данных (DO). Используется в трехпроводном режиме (SPI);
- PE5 — вход данных (DI)/линия данных (SDA);
- PE4 — вход/выход тактового сигнала (USCK)/линия тактового сигнала (SCL).

В 8-битном сдвиговом регистре содержатся входящие и исходящие данные. Старший бит сдвигового регистра подключается, в зависимости от режима работы, к одной из двух линий данных модуля. Входящие биты всегда считываются с вывода DI, независимо от конфигурации модуля. Четырехбитный счетчик, показанный на **Рис. 12.1**, доступен как для чтения, так и для записи. Кроме того, при его переполнении может генерироваться прерывание. Поскольку сдвиговый регистр и счетчик используют один и тот же тактовый сигнал, счетчик может считать число переданных или принятых битов и сгенерировать прерывание по окончании процесса обмена. Обратите внимание, что при использовании внешнего тактового сигнала изменение состояния счетчика происходит по каждому фронту сигнала, т. е. счетчик считает количество фронтов, а не битов. Тактовый сигнал может сниматься с входа USCK, с выхода блока сравнения таймера/счетчика T0 или формироваться программно.

Блок управления тактовым сигналом используется в двухпроводном режиме и может генерировать прерывание при обнаружении на шине состояния СТАРТ. Кроме того, он может формировать на шине циклы ожидания, удерживая на линии SCL НИЗКИЙ уровень после обнаружения состояния СТАРТ или после переполнения счетчика.

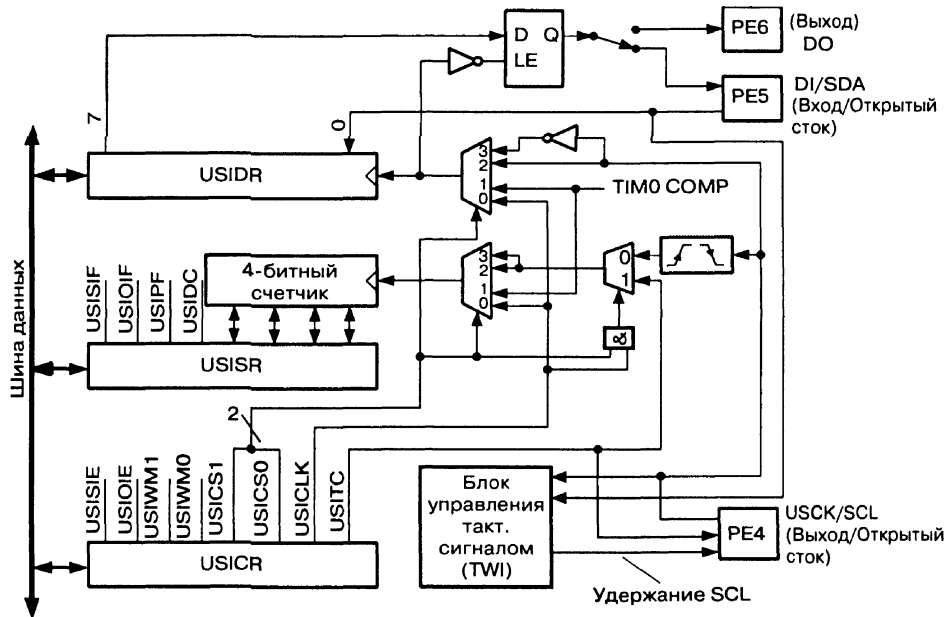


Рис. 12.1. Структурная схема модуля USI

13.2. Использование модулей USART

Упрощенная структурная схема одного модуля USART приведена на Рис. 13.1.

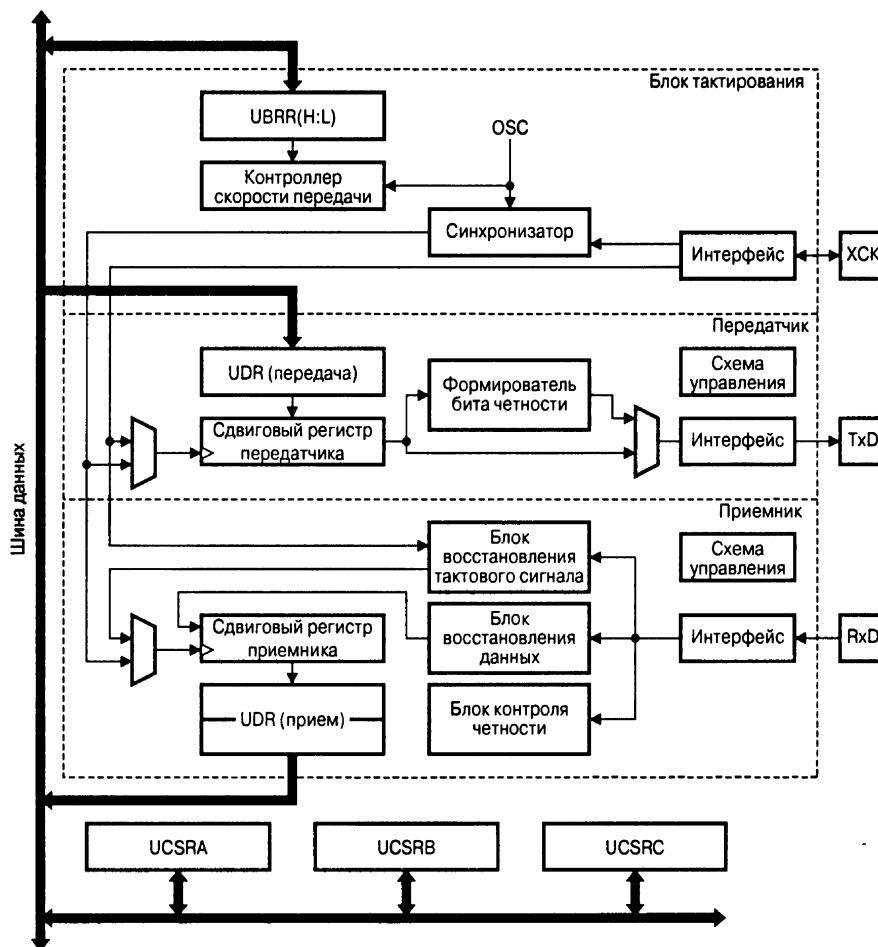


Рис. 13.1. Структурная схема модуля USART

Как показано на рисунке, модуль состоит из трех основных частей: блока тактирования, блока передатчика и блока приемника. Блок тактирования модулей USART содержит схему синхронизации, которая используется при работе в синхронном режиме, и контроллер скорости передачи.

Блок передатчика включает одноуровневый буфер, сдвиговый регистр, схему формирования бита четности и схему управления. Блок приемника, в свою очередь, содержит схемы восстановления тактового сигнала и данных, схему контроля четности, двухуровневый буфер, сдвиговый регистр, а также схему управления.

Буферные регистры приемника и передатчика располагаются по одному адресу пространства ввода/вывода и обозначаются как регистр данных UDR (UDR_n). В этом регистре хранятся младшие 8 битов принимаемых и передаваемых данных. При чтении регистра UDR выполняется обращение к буферному регистру приемника, при записи — к буферному регистру передатчика. Размещение регистров данных UDR для различных моделей микроконтроллеров приведено в Табл. 13.3.

В модулях USART буфер приемника является двухуровневым (FIFO-буфер), изменение состояния которого происходит при любом обращении к регистру UDR. В связи с этим не следует использовать регистр UDR в качестве операндов команд типа «чтение/модификация/запись» (SBI и CBI). Кроме того, следует быть очень аккуратными при использо-

Таблица 13.3. Размещение регистров данных модулей USART

Регистр	Адрес	ATmega8515x/8535x	ATmega8x/16x/32x	ATmega64x/128x	ATmega48x/88x/168x	ATmega162x	ATmega164x/324x/644x	ATmega165x, ATmega325x/3250x, ATmega645x/6450x	ATmega640x/1280x/2560x	ATmega1281x/2561x	Описание
UDR	\$0C (\$2C)	•	•								Регистр данных USART
	(\$C6)							•			
UDR0	\$0C (\$2C)			•		•					Регистр данных USART0
	(\$C6)				•		•		•	•	
UDR1	\$03 (\$23)					•					Регистр данных USART1
	(\$9C)			•							
	(\$CE)						•		•	•	
UDR2	(\$D6)								•		Регистр данных USART2
UDR3	(\$136)								•		Регистр данных USART3

Формат регистров UCSRA (UCSR_{nA}), UCSRB (UCSR_{nB}) и UCSRC (UCSR_{nC}) приведен на Рис. 13.2...13.4, а значение битов этих регистров описано в Табл. 13.5...13.7 соответственно.

Регистр	Адрес	АТмега8515х/8535х	АТмега8х/16х/32х	АТмега64х/128х	АТмега48х/88х/168х	АТмега162х	АТмега164х/324х/644х	АТмега165х, АТмега325х/3250х, АТмега645х/6450х	АТмега640х/1280х/2560х	АТмега1281х/2561х	Описание
UCSR1C	\$3C (\$5C)					•					Регистр С управления USART1
	(\$9D)			•							
	(\$CA)						•		•	•	
UCSR2A	(\$D0)								•		Регистр А управления USART2
UCSR2B	(\$D1)								•		Регистр В управления USART2
UCSR2C	(\$D2)								•		Регистр С управления USART2
UCSR3A	(\$130)								•		Регистр А управления USART3
UCSR3B	(\$131)								•		Регистр В управления USART3
UCSR3C	(\$132)								•		Регистр С управления USART3

	7	6	5	4	3	2	1	0
	RXC _n	TXC _n	UDRE _n	FE _n	DOR _n	UPE _n	U2X _n	MPCM _n
Чтение(R)/Запись(W)	R	R/W	R	R	R	R	R/W	R/W
Начальное значение	0	0	1	0	0	0	0	0

Рис. 13.2. Формат регистров UCSRA (UCSR_{nA})

Таблица 13.5. Биты регистров UCSRA (UCSRnA)

Бит	Название	Описание
7	RXC (RXCn)	Флаг завершения приема. Флаг устанавливается в 1 при наличии непрочитанных данных в буфере приемника (регистр данных UDR). Сбрасывается флаг аппаратно после опустошения буфера. Если бит RXCIE (RXCIE _n) регистра UCSRB (UCSRnB) установлен, то при установке флага генерируется запрос на прерывание «прием завершен»
6	TXC (TXCn)	Флаг завершения передачи. Флаг устанавливается в 1 после передачи всех битов посылки из сдвигового регистра передатчика при условии, что в регистр данных UDR не было загружено новое значение. Если бит TXCIE (TXCIE _n) регистра UCSRB (UCSRnB) установлен, то при установке флага генерируется прерывание «передача завершена». Флаг сбрасывается аппаратно при выполнении подпрограммы обработки прерывания или программно, запись в него лог. 1
5	UDRE (UDREn)	Флаг опустошения регистра данных. Данный флаг устанавливается в 1 при пустом буфере передатчика (после пересылки байта из регистра данных UDR в сдвиговый регистр передатчика). Установленный флаг означает, что в регистр данных можно загружать новое значение. Если бит UDRIE (UDRIE _n) регистра UCSRB (UCSRnB) установлен, генерируется запрос на прерывание «регистр данных пуст». Флаг сбрасывается аппаратно, при записи в регистр данных
4	FE (FE _n)	Флаг ошибки кадрирования. Флаг устанавливается в 1 при обнаружении ошибки кадрирования, т. е. если первый стоп-бит принятой посылки равен 0. Флаг сбрасывается при приеме стоп-бита, равного 1
3	DOR (DORn)	Флаг переполнения. Флаг устанавливается в 1, если в момент обнаружения нового старт-бита в сдвиговом регистре приемника находится последнее принятое слово, а буфер приемника полон (содержит два байта). Флаг сбрасывается при пересылке принятых данных из сдвигового регистра приемника в буфер
2	UPE (UPE _n)	Флаг ошибки контроля четности. Флаг устанавливается в 1, если в данных, находящихся в буфере приемника, выявлена ошибка контроля четности. При отключенном контроле четности этот бит постоянно сброшен в 0
1	U2X (U2Xn)	Удвоение скорости обмена. Если этот бит установлен в 1, то коэффициент деления предделителя контроллера скорости передачи уменьшается с 16 до 8, удваивая тем самым скорость асинхронного обмена по последовательному каналу. Этот бит используется только при асинхронном режиме работы и в синхронном режиме должен быть сброшен
0	MPCM (MPCMn)	Режим мультипроцессорного обмена. Если этот бит установлен в 1, ведомый микроконтроллер ожидает приема кадра, содержащего адрес. Кадры, не содержащие адреса устройства, игнорируются

Примечание. $n = 0, 1, 2$ или 3 .

	7	6	5	4	3	2	1	0
	RXCIE _n	TXCIE _n	UDRIE _n	RXEN _n	TXEN _n	UCSZ _{n2}	RXB8 _n	TXB8 _n
Чтение(R)/Запись(W)	R	R/W	R/W	R/W	R/W	R/W	R	R/W
Начальное значение	0	0	0	0	0	0	1	0

Рис. 13.3. Формат регистров UCSRB (UCSRnB)

Таблица 13.6. Биты регистров UCSRB (UCSRnB)

Бит	Название	Описание
7	RXCIE (RXCIE _n)	Разрешение прерывания по завершении приема. Если данный бит установлен в 1, то при установке флага RXC (RXC _n) регистра UCSRA (UCSRnA) генерируется прерывание «прием завершен» (если флаг I регистра SREG установлен в 1)
6	TXCIE (TXCIE _n)	Разрешение прерывания по завершении передачи. Если данный бит установлен в 1, то при установке флага TXC (TXC _n) регистра UCSRA (UCSRnA) генерируется прерывание «передача завершена» (если флаг I регистра SREG установлен в 1)
5	UDRIE (UDRIE _n)	Разрешение прерывания при очистке регистра данных UART. Если данный бит установлен в 1, то при установке флага UDRE (UDRE _n) регистра UCSRA (UCSRnA) генерируется прерывание «регистр данных пуст» (если флаг I регистра SREG установлен в 1)
4	RXEN (RXEN _n)	Разрешение приема. При установке этого бита в 1 разрешается работа приемника USART и переопределяется функционирование вывода RXD (RXD _n). При сбросе бита RXEN (RXEN _n) работа приемника запрещается, а его буфер сбрасывается. Значения флагов TXC(TXC _n), DOR (DOR _n) и FE (FE _n) при этом становятся недействительными
3	TXEN (TXEN _n)	Разрешение передачи. При установке этого бита в 1 разрешается работа передатчика UART и переопределяется функционирование вывода TXD (TXD _n). Если бит сбрасывается в 0 во время передачи, то выключение передатчика произойдет только после завершения передачи данных, находящихся в сдвиговом регистре и буфере передатчика
2	UCSZ2 (UCSZn2)	Формат посылок. Этот бит совместно с битами UCSZ1:0 (UCSZn1:0) регистра UCSRC (UCSRnC) используется для задания размера слов данных, передаваемых по последовательному каналу
1	RXB8 (RXB8 _n)	8-й бит принимаемых данных. При использовании 9-битных слов данных этот бит содержит значение старшего бита принятого слова. Содержимое этого бита должно быть считано до прочтения регистра данных UDR
0	TXB8 (TXB8 _n)	8-й бит передаваемых данных. При использовании 9-битных слов данных содержимое этого бита является старшим битом передаваемого слова. Требуемое значение должно быть занесено в этот бит до загрузки байта данных в регистр UDR

Примечание. $n = 0, 1, 2$ или 3 .

Таблица 13.7. Биты регистров UCSRC (UCSRnC)

Бит	Название	Описание
7	UMSELn1	Режим работы USART (модели 48х/88х/168х, 164х/324х/644х и 640х/1280х/1281х/2560х/2561х). Совместно с битом UMSELn0 определяет режим работы модуля USART
	URSEL ¹⁾ (URSELn)	Выбор регистра. Этот бит определяет, в какой из регистров модуля производится запись. Если бит установлен в 1, обращение производится к регистру UCSRC (UCSRnC). Если же бит сброшен в 0, обращение производится к регистру UBRRH (UBRRnH). Подробнее — см. следующий подраздел
6	UMSELn	Режим работы USART. Если бит сброшен в 0, то модуль работает в асинхронном режиме. Если бит установлен в 1, то модуль работает в синхронном режиме
	UMSELn0	Режим работы USART (модели 48х/88х/168х, 164х/324х/644х и 640х/1280х/1281х/2560х/2561х). Совместно с битом UMSELn1 определяет режим работы модуля USART
5	UPM1 (UPMn1)	Режим работы схемы контроля и формирования бита четности. Эти биты определяют функционирование схем контроля и формирования бита четности (см. подраздел 13.2.2)
4	UPM0 (UPMn0)	
3	USBS (USBSn)	Количество стоп-битов. Этот бит определяет количество стоп-битов, посылаемых передатчиком. Если бит сброшен в 0, передатчик посылает 1 стоп-бит, если установлен в 1, то 2 стоп-бита. Для приемника содержимое этого бита безразлично
2	UCSZ1 (UCSZn1)	Формат посылок. Совместно с битом UCSZ2 (UCSZn2) регистра UCSRB (UCSRnB) эти биты определяют количество битов данных в посылках (размер слова)
1	UCSZ0 (UCSZn0)	

	7	6	5	4	3	2	1	0	
	—	UMSELn	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	ATmega64x ATmega128x
Чтение(R)/Запись(W)	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	1	1	0	
	7	6	5	4	3	2	1	0	ATmega8515х/8535х ATmega8х/16х/32х ATmega162х
	URSELn	UMSELn	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	
Чтение(R)/Запись(W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	ATmega165х/325х/3250х
Начальное значение	0	0	0	0	0	1	1	0	ATmega645х/6450х
	7	6	5	4	3	2	1	0	ATmega48х/88х/168х ATmega164х/324х/644х ATmega640х/1280х/1281х ATmega2560х/2561х
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn	
Чтение(R)/Запись(W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	1	1	0	

Рис. 13.4. Формат регистров UCSRC (UCSRnC)

Обратите внимание на то, что в моделях ATmega48х/88х/168х, ATmega164х/324х/644х и ATmega640х/1280х/1281х/2560х/2561х для задания режима работы используется не один бит регистра UCSRC, а два — UMSELn1:0. Зависимость режима работы модуля USART от установок этих битов приведена в Табл. 13.8.

Бит	Название	Описание		
0	UCPOL (UCPOL n)	Полярность тактового сигнала. Значение этого бита определяет момент выдачи и считывания данных на выводах модуля. Бит используется только при работе в синхронном режиме. При работе в асинхронном режиме он должен быть сброшен в 0.		
		UCPOL (UCPOL n)	Выдача данных на вывод TXD (TXD n)	Считывание данных с вывода RXD (RXD n)
		0	Спадающий фронт ХСК (ХСК n)	Нарастающий фронт ХСК (ХСК n)
		1	Нарастающий фронт ХСК (ХСК n)	Спадающий фронт ХСК (ХСК n)
¹⁾ Зарезервирован в моделях АТmega64х/128х.				

Примечание. $n = 0, 1, 2$ или 3 .

Таблица 13.8. Управление режимом работы модулей USART моделей АТmega48х/88х/168х, АТmega164х/324х/644х и АТmega640х/1280х/1281х/2560х/2561х

UMSEL n 1	UMSEL n 0	Режим работы
0	0	Асинхронный USART
0	1	Синхронный USART
1	0	Зарезервировано
1	1	Ведущий шины SPI

13.2.1. Скорость приема/передачи

В асинхронном режиме, а также в синхронном режиме при работе в качестве ведущего скорость приема и передачи данных задается контроллером скорости передачи, работающим как делитель системного тактового сигнала с программируемым коэффициентом деления. Коэффициент определяется содержимым регистра контроллера UBRR (UBRR n). В блок приемника сформированный сигнал поступает напрямую, а в блок передатчика — через дополнительный делитель, коэффициент деления которого (2, 8 или 16) зависит от режима работы модуля USART.

Регистр UBRR является 12-битным и физически размещается в двух регистрах ввода/вывода. Адреса и названия этих регистров для различных моделей микроконтроллеров приведены в Табл. 13.9.

Таблица 13.9. Размещение регистров контроллера скорости передачи

Модель	Регистры	Адрес
АТmega8515х/8535х ¹⁾	UBRRH:UBRRL	\$20 (\$40):\$09 (\$29)
АТmega8х/16х/32х ¹⁾	UBRRH:UBRRL	\$20 (\$40):\$09 (\$29)
АТmega64х/128х	UBRR0H:UBRR0L	(\$90):\$09 (\$29)
	UBRR1H:UBRR1L	(\$98):(\$99)
АТmega48х/88х/168х	UBRR0H:UBRR0L	(\$C5):(\$C4)

ATmega162x ¹⁾	UBRR0H:UBRR0L	\$20 (\$40):\$09 (\$29)
	UBRR1H:UBRR1L	\$3C (\$5C):\$00 (\$20)
ATmega164x/324x/644x	UBRR0H:UBRR0L	(\$C5):(\$C4)
	UBRR1H:UBRR1L	(\$CD):(\$CC)
ATmega165x/325x/3250x/645x/6450x	UBRR0H:UBRR0L	(\$C5):(\$C4)
ATmega640x/1280x/2560x	UBRR0H:UBRR0L	(\$C5):(\$C4)
	UBRR1H:UBRR1L	(\$CD):(\$CC)
	UBRR2H:UBRR2L	(\$D5):(\$D4)
	UBRR3H:UBRR3L	(\$135):(\$134)
ATmega1281x/2561x	UBRR0H:UBRR0L	(\$C5):(\$C4)
	UBRR1H:UBRR1L	(\$CD):(\$CC)
¹⁾ В этих моделях регистр UBRRH (UBRR _n H) размещается по тому же адресу, что и регистр UCSRC (UCSR _n C).		

Обратите внимание на то, что в моделях ATmega8515x/8535x, ATmega8x/16x/32x и ATmega162x регистр UBRRH размещается по тому же адресу, что и регистр управления UCSRC. Поэтому при обращении по этим адресам необходимо выполнить ряд дополнительных действий для выбора конкретного регистра.

При записи регистр определяется состоянием старшего бита записываемого значения URSEL. Если этот бит сброшен в 0, изменяется содержимое регистра UBRRH. Если же старший бит значения установлен в 1, изменяется содержимое регистра управления UCSRC. Приведенные ниже фрагменты программ иллюстрируют сказанное:

Оборудование и материалы.

Комплект лабораторных модулей микропроцессорная техника РТМТЛ. Принципиальная схема лабораторного стенда приведена ниже

Комплект лабораторных модулей микропроцессорная техника РТМТЛ. Принципиальная схема лабораторного стенда приведена ниже лабораторных работ по исследованию программирования микроконтроллеров на различных языках программирования выполняется на учебном макетном лабораторном комплексе РТМТЛ.

На передней крышке прибора закреплена отладочная плата, содержащая исследуемый микроконтроллер (Atmega8535, Atmega16, Atmega32 или совместимый); резистивно-диодный последовательный (работающий по интерфейсу RS232 через COM-порт ПК) программатор; обвязку микроконтроллера согласно паспортным данным; набор из 8 светодиодов, подключенных к выводам 33 – 40, см. рис. 2.1; набор из 8 кнопок с фиксацией нормально разомкнутых (кнопка нажата — замкнута; отжата — разомкнута), подключенных к выводам 22 — 29; а также кнопку с фиксацией «RESET», при нажатии которой вывод RESET кристалла соединяется с корпусом схемы. Для программирования прибора используется разъем «ПРОГРАММАТОР». При этом «ПРОГРАММАТОР» следует подключать к COM – порту ПЭВМ ТОЛЬКО проводом типа COM 9m/9f («мама» - «папа»). В данной работе на примере микроконтроллера AVR (Atmega8535, Atmega16, Atmega32 или совместимого) предлагается освоить простейшие приемы программирования на следующих языках:

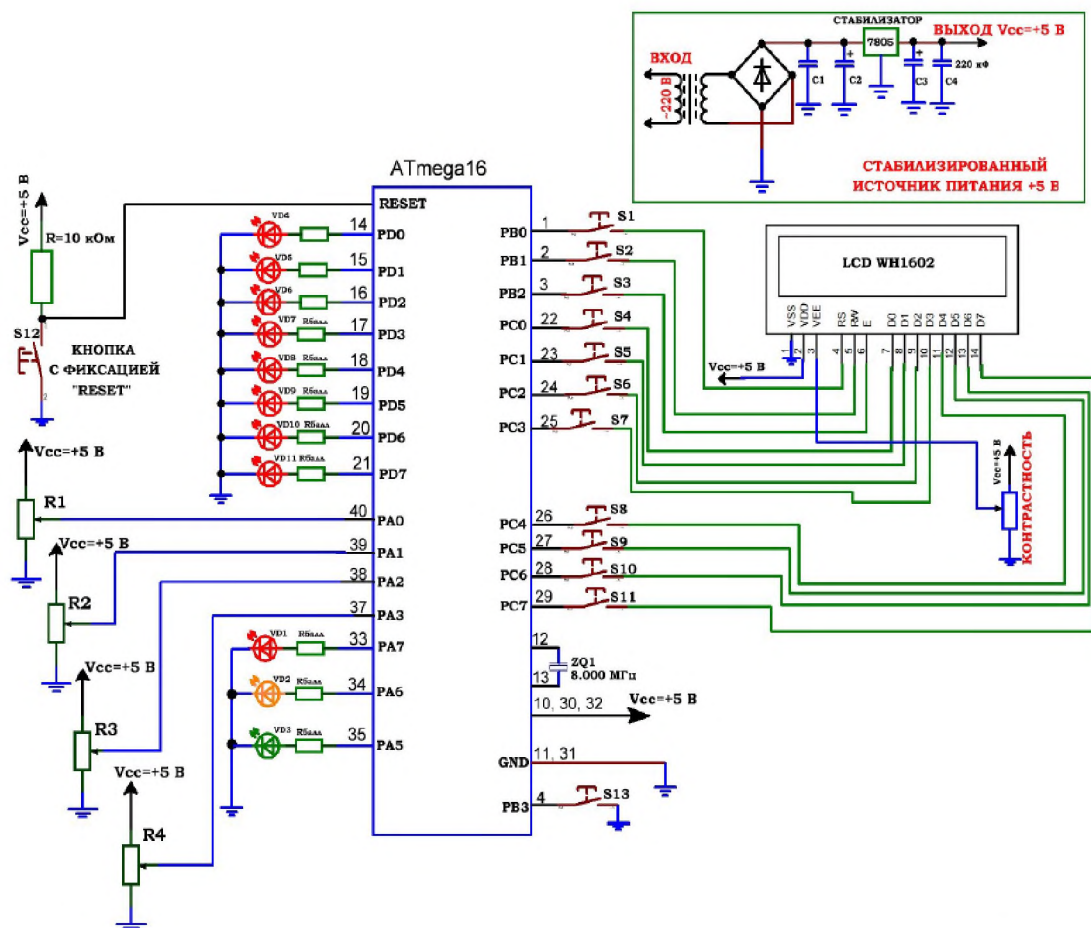


Рис. 4.1. Принципиальная электрическая схема учебного стенда для изучения работы LCD ЖК символического индикатора с микроконтроллером AVR.

Указания по технике безопасности

Соответствуют технике безопасности по работе с компьютерной техникой.

Ход работы

1. Перед включением установки в сеть проверить целостность всех соединительных сигнальных и сетевых проводов. Все работы по подключению комплекса к компьютеру следует выполнять только при отключенных от сети приборах. Разобраться с принципиальными блок-схемами опытов, в назначении кнопок, переключателей и ручек прибора.
2. Перед началом работы необходимо ознакомиться с документацией на исследуемый микроконтроллер, документацией на исследуемую среду программирования, а также со статьями по данным вопросам, размещенными в соответствующих папках на прилагаемом к учебной установке диске или Flash карте.
3. Соединить монитор с системным блоком ПЭВМ, подключить клавиатуру и мышь к системному блоку используя стандартные провода для подключения. Подключить системный блок ПЭВМ и монитор к сети ~220 В.
4. Загрузить операционную систему согласно стандартным процедурам загрузки.
5. Открыть демонстрационный пример (проект) в исследуемой среде программирования (AVR Studio, Vascom-AVR, WinAVR, Algorithm Builder). Откомпилировать проект в исследуемой среде и получить готовый бинарный HEX-файл.
6. Подключить разъём «ПРОГРАММАТОР» учебного прибора к COM – порту ПЭВМ проводом типа COM 9m/9f («мама» - «папа»).
7. Прошить тестовую программу (HEX-файл) в кристалл. Т. к. для программирования микроконтроллера (прошивки) используется резистивно-диодный

последовательный (работающий по интерфейсу RS232 через COM-порт ПК) программатор («ПРОГРАММАТОР ГРОМОВА»), то для прошивки hex-файлов, полученных в средах программирования AVR Studio, Bascom-AVR, WinAVR следует использовать утилиту SinaPROG, поддерживающую данный резистивно-диодный последовательный программатор. При работе в среде Algorithm Builder полученный HEX-файл может быть зашит в микроконтроллер непосредственно из среды.

8. При работе с программой SinaPROG следует избегать пробелов и русских символов в полном имени HEX-файла (имя с учетом полного пути к файлу) и общей длины пути к HEX-файлу более 255 символов. Идеальным вариантом будет копирование полученного HEX-файла в корень диска перед прошивкой. 9. Перед прошивкой следует нажать кнопку с фиксацией «RESET» на учебном приборе.

10. Особенно внимательно следует отнестись к опциям «ЗАПИСЬ Fuse bit» и «ЗАПИСЬ блокирующих бит» в среде Algorithm Builder и в утилите SinaPROG. ФЛАЖКИ ЭТИХ ОПЦИЙ ОБЯЗАТЕЛЬНО ДОЛЖНЫ БЫТЬ СНЯТЫ, И ЭТУ ОПЦИЮ В УЧЕБНЫХ ЦЕЛЯХ ИСПОЛЬЗОВАТЬ НЕ РЕКОМЕНДУЕТСЯ, Т. К. НЕ ПРАВИЛЬНО ПРОШИТЫЕ FUSE БИТЫ МОГУТ ПРИВЕСТИ К ДАЛЬНЕЙШЕЙ НЕВОЗМОЖНОСТИ РАБОТЫ С ДАННЫМ МИКРОКОНТРОЛЛЕРОМ!

11. Отжать кнопку RESET и проверить работу тестовой программы.

12. Исходя из данного учебного примера, составить свои программы в различных средах программирования. Например, программа может прибавлять к данным порта А (РА0-РА7) любое другое произвольное число. Так-же рекомендуется поработать с действиями «умножение» и «вычитание».

13. Для составления программ и подробного изучения работы микроконтроллеров AVR следует обращаться к учебной литературе, полному руководству к микроконтроллерам семейства AVR, а также к документации по средам программирования, используемым в работе.

14. По окончании работы следует закрыть программу и все открытые подпрограммы.

15. Выключить компьютер, нажав на кнопку, находящуюся в крайнем нижнем левом углу экрана. Из доступных действий выбрать «ВЫХОД»--> «ВЫКЛЮЧИТЬ КОМПЬЮТЕР».

16. Отключить установку от сети, поставив переключатели «СЕТЬ» на панели установки в положение «ВЫКЛ» и вынуть сетевые вилки из розеток. __

Оформление отчета

В отчете должны содержаться описание команд использованных в выполнении заданий и программы написанные по заданным в задании алгоритмам

Контрольные вопросы

7. Какой уровень сигнала у цифровых выходов МК?
8. Какие типы команд имеет микроконтроллер AVR семейства Mega?
9. Какие типы команд изменяют флаги микроконтроллера?
10. Какие виды адресации использует микроконтроллер AVR?
11. Зачем нужны команды передачи управления?
12. Как организована архитектура ядра микроконтроллеров AVR семейства Mega?

Список литературы, рекомендуемый к использованию по данной теме:

11. Клингман Э. Проектирование микропроцессорных систем. М.: Мир. 1988. - 575с.
12. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах.

13. Трамперт В. AVR-RISK микроконтроллеры.: Пер. с нем. – К.: «МК-ПРЕСС», 2006.-464с., ил.
14. Ю.А.Шпак. Программирование на языке С для AVR и PIC микроконтроллеров. – К.: «МК-ПРЕСС», 2006.-400с., ил.
15. Якубовский С.В. Цифровые и аналоговые интегральные микросхемы. М.:Радио и связь. 1990.
16. Токхейм Р. Основы цифровой электроники. Пер. с англ. - М.: Мир, 1988. – 390 с.
17. Шило В.Л. Популярные цифровые микросхемы. – М.: Радио и связь, 1990.– 350 с.
18. Бирюков С.А. Цифровые устройства на МОП-интегральных микросхемах - М.: Радио и связь. 1992 (1996).
19. Опачий Ю.Ф., Глудкин О.П., Гуров А.И. Аналоговая и цифровая электроника. Полный курс: учебник для вузов. - М.: Горячая линия, 1999 (2000, 2005). – 768 с.
20. Алексенко А.В., Шагуров И.И. Микросхемотехника.– М.: Радио и связь,1990 (1982).

Лабораторная работа 6

Инструментальные средства разработки программного обеспечения ПЛК. Разработка программ работы с дискретными сигналами микроконтроллера AVR

Цель работы

Изучение инструментальных средств разработки программного обеспечения МК.
Разработка программ работы с дискретными сигналами микроконтроллера AVR..

Компетенции:

Индекс	Формулировка:
ОПК-3	способностью решать задачи анализа и расчета характеристик электрических цепей
ОПК-7	способностью учитывать современные тенденции развития теории автоматического управления, электронного оборудования, измерительной и вычислительной техники, информационных технологий в профессиональной деятельности
ПК-1	способностью выполнять эксперименты на действующих объектах по заданным методикам, оценивать динамические характеристики рассматриваемых объектов и идентифицировать передаточные функции объектов с применением современных информационных технологий и технических средств
ПК-7	готовностью участвовать в разработке и изготовлении стендов для комплексной отладки и испытаний программно-аппаратных управляющих комплексов
ПК-8	готовностью к внедрению результатов разработок средств и систем автоматизации и управления в производстве с использованием современных программируемых логических контролеров (ПЛК)
ПК-11	готовностью производить инсталляцию и настройку системного, прикладного и инструментального программного обеспечения систем автоматизации и управления
ПК-14	способностью разрабатывать электромеханические системы и использовать современную элементную базу при проектировании средств и систем управления

Теоретическая часть

Учебная графическая среда для разработки программного обеспечения для микроконтроллеров с архитектурой AVR фирмы ATMEL.

Среда Algorithm Builder

Среда предназначена для производства полного цикла разработки начиная от ввода алгоритма, включая процесс отладки и заканчивая программированием кристалла рис. 2.1. Разработка программы может быть как на уровне ассемблера, так и на макроуровне с манипуляцией многобайтными величинами со знаком.

В отличие от классического ассемблера программа вводится в виде алгоритма с древовидными ветвлениями и отображается на плоскости, в двух измерениях. Сеть

условных и безусловных переходов отображается графически, в удобной векторной форме. Это к тому же освобождает программу от бесчисленных имен меток, которые в классическом ассемблере являются неизбежным балластом. Вся логическая структура программы становится наглядной.

Графические технологии раскрывают новые возможности для программистов. Визуальность логической структуры уменьшает вероятность ошибок и сокращает сроки разработки.

Для настройки периферийных устройств (таймеры, UART, ADC, SPI и т.д.) предусмотрен специальный элемент алгоритма - "настройщик" с раскрывающимся оконным интерфейсом. В нем достаточно выбрать необходимые параметры работы устройства, а набор инструкций обеспечивающих эти параметры, сформирует компилятор (в правой части окна) рис. 2.2

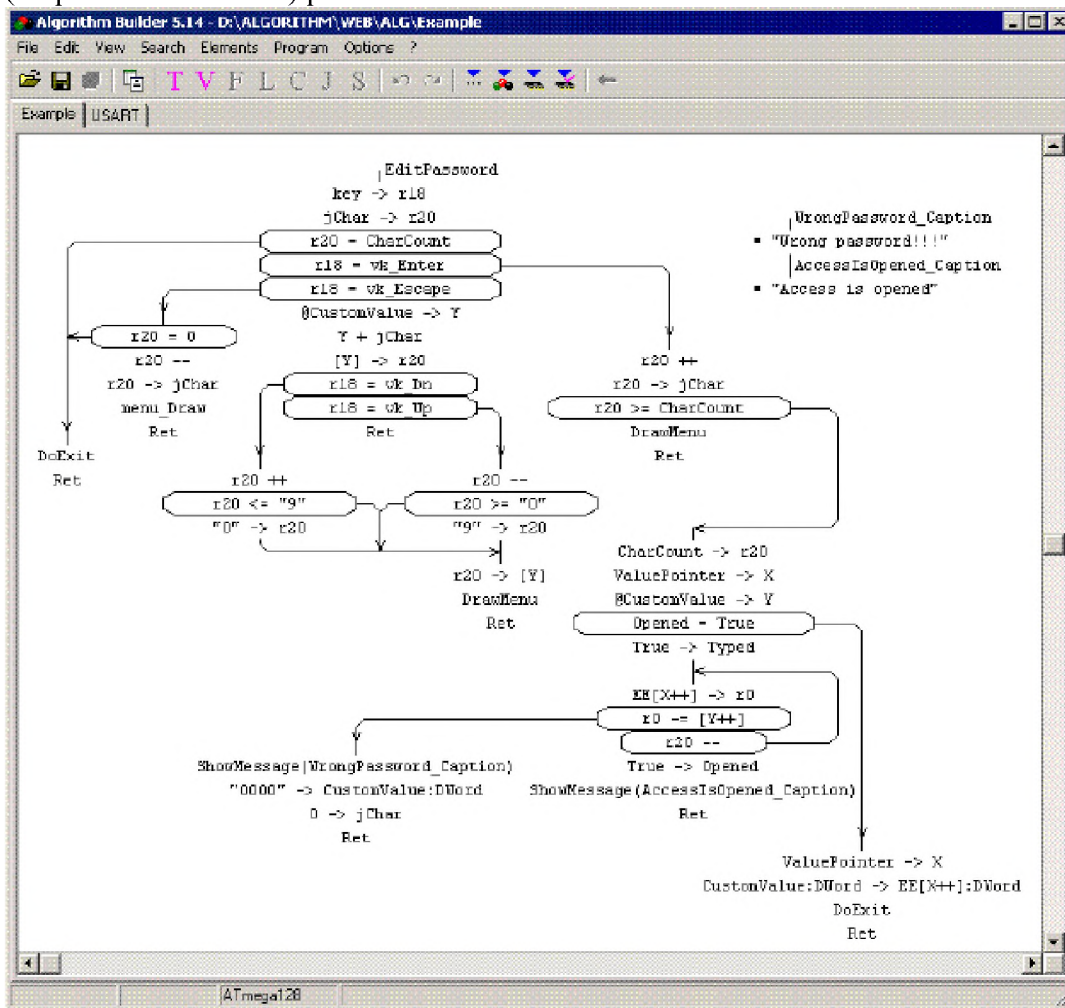


Рис. 2.1. Учебная среда программирования микроконтроллеров.

Поддерживается автоматическая перекодировка строк ANSI-кодов Windows в коды русифицированного буквенно-цифрового ЖКИ.

Среда объединяет в себе графический редактор, компилятор алгоритма, симулятор микроконтроллера, внутрисхемный программатор.

При использовании внутрисхемного программатора микроконтроллер подключается к СОМ порту компьютера через несложный адаптер (три диода и несколько резисторов). Программатор ведет подсчет числа перепрограммирований кристалла, сохраняя счетчик непосредственно в кристалле.

Algorithm Builder обеспечивает мониторинговую отладку на кристалле (On Chip debug) которая позволяет наблюдать содержимое реального кристалла в заданной точке останова. При этом, для связи микроконтроллера с компьютером используется только один вывод,

причем по выбору пользователя. Мониторная отладка может быть применена к любому типу кристалла, имеющего SRAM.

Среда предназначена для работы в ОС Windows 95-Windows 8.1 (32/64 bit).

Оборудование и материалы.

Комплект лабораторных модулей микропроцессорная техника РТМТЛ. Принципиальная схема лабораторного стенда приведена ниже

Комплект лабораторных модулей микропроцессорная техника РТМТЛ. Принципиальная схема лабораторного стенда приведена ниже лабораторных работ по исследованию программирования микроконтроллеров на различных языках программирования выполняется на учебном макетном лабораторном комплексе РТМТЛ.

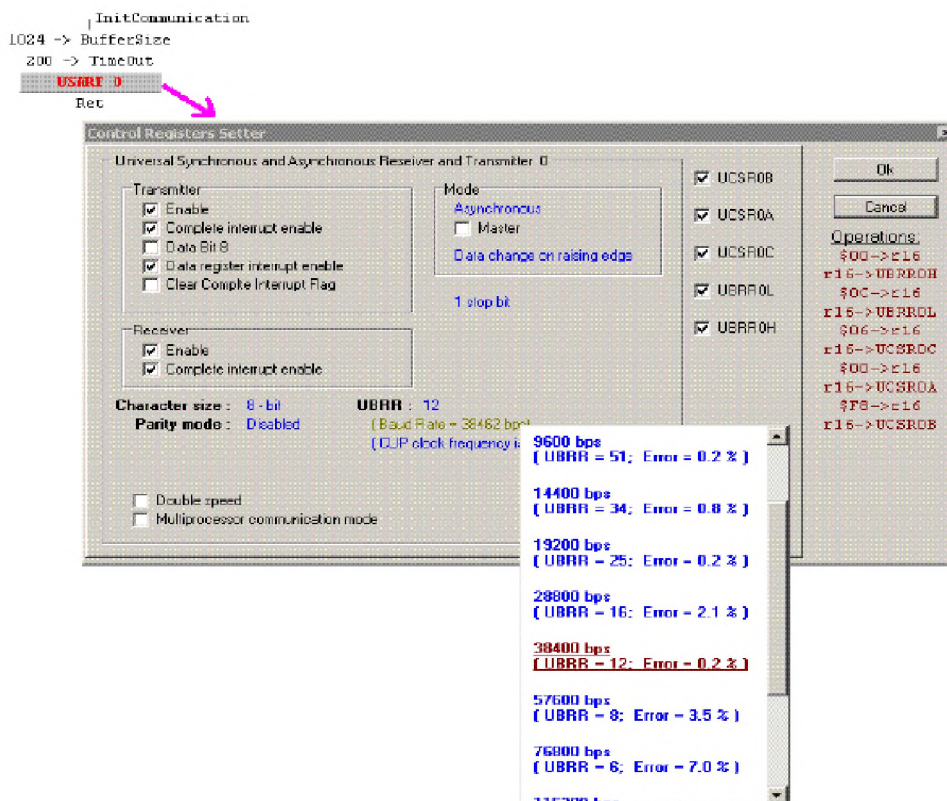


Рис. 2.2. Настройка периферийных устройств.

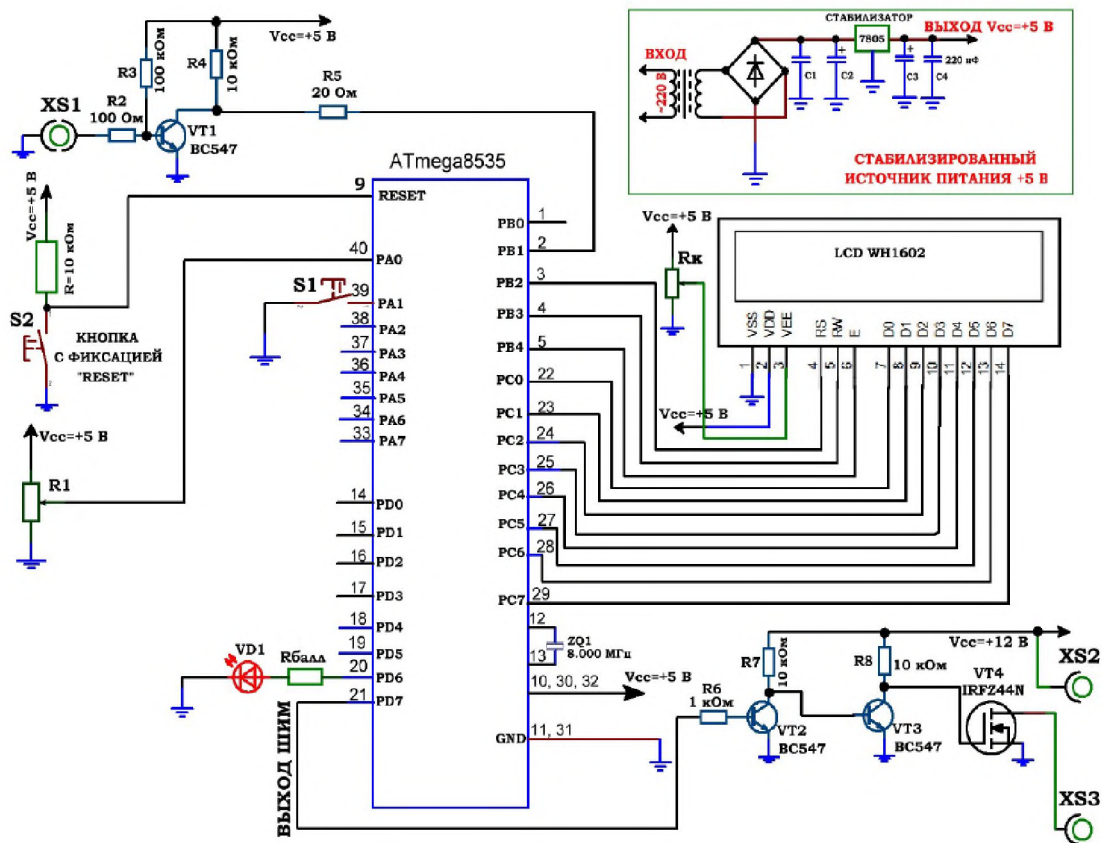


Рис. 3.1. Принципиальная электрическая схема учебного стенда для изучения микроконтроллеров серии Atmega8535.

Указания по технике безопасности

Соответствуют технике безопасности по работе с компьютерной техникой.

Ход работы

Комплекс лабораторных работ по исследованию микроконтроллеров AVR выполняется на учебном макетном лабораторном комплексе РТМТЛ-1.

На передней крышке прибора закреплена отладочная плата, содержащая исследуемый микропроцессор (Atmega8535); резистивно-диодный последовательный (работающий по интерфейсу RS232 через COM-порт ПК) программатор; обвязку микроконтроллера согласно паспортным данным; 1 светодиод, подключенный к выводу 20 рис. 3.1; 1 кнопка с фиксацией нормально разомкнутая (кнопка нажата — замкнуто; отжата — разомкнуто), подключенная к выводу 39; 1 потенциометр (переменный резистор), подключенный средней точкой к выводу 40 микроконтроллера; выходы «НАГРУЗКА» XS2 — XS3; вход XS1 (для примера работы частотомера); подстроечный резистор Rк — контрастность, подключенный средней точкой к выводу VEE LCD индикатора; а также кнопку с фиксацией «RESET», при нажатии которой вывод RESET кристалла соединяется с корпусом схемы. Для программирования прибора используется разъем «ПРОГРАММАТОР». При этом «ПРОГРАММАТОР» следует подключать к COM – порту ПЭВМ **ТОЛЬКО** проводом типа COM 9m/9f («мама» - «папа»).

Принципиальная электрическая схема прибора приведена на рис. 3.1.

Демонстрационные примеры готовых программ, на основе которых могут быть построены задания по программированию микроконтроллера, приведены в папке Examples_RTMTL-1.

Задание 1. Работа со светодиодом, таймером, ЖК индикатором и входом PB1 микроконтроллера. Построение программы-частотомера. В данном примере на вывод 2 PB1 микроконтроллера (вход XS1) подаётся сигнал с внешнего генератора (в качестве генератора можно использовать программу «LabVisual ЗВУКОВОЙ ГЕНЕРАТОР ПК»). Сигнал предварительно усилен однотранзисторным резистивным усилителем, собранном на транзисторе VT1 BC547 и резисторах R2-R5. Микроконтроллер программируется таким образом, чтобы его TIMER1 был настроен на режим синхронизации внешним сигналом, поступающим на порт PB1. Программа 1 раз в секунду выполняет подсчет входных импульсов синхронизации, поступающих на вход PB1 (макс. Значение 65535). Предусмотрен вывод показаний на LCD индикатор (максимальное значение 9999 Гц). Количество подсчитанных импульсов за 1 сек и есть частота входного сигнала в Гц.

Программа также одновременно работает с подключенным к микроконтроллеру ЖК LCD индикатором (инициализирует его, осуществляет вывод информации).

Одновременно с этим каждую секунду производится инвертирование состояние вывода PD6, к которому подключен светодиод VD1 (1 сек. Светодиод включен, 1 сек. - выключен). При отключенном от клеммы XS1 внешнем генераторе в нижней строке дисплея отображается INPUT DISCONNECT, при подключении внешнего генератора и частоте отличной от 0, в строке выводится сообщение INPUT CONNECT.

Задание 2. Работа переменного резистора, кнопки, ЖК индикатора и выхода ШИМ генератора. ШИМ генерируется на выводе PD7 микроконтроллера и подается на трёхкаскадный усилитель мощности, два каскада которого собраны на биполярных транзисторах с общим эмиттером VT2=VT3=BC547, а третий каскад — на полевом транзисторе IRFZ44N с общим истоком. В стоковую цепь полевого транзистора подключается нагрузка (клеммы XS2 – XS3) относительно +12 В питания (колебания напряжения питания может составлять +10 ... +12 В, т. к. данный источник не стабилизированный). В качестве нагрузки может быть использован электродвигатель вентилятора либо лампа накаливания 12 В, 5 Вт. **Обратить внимание, что вентилятор плюсовым красным выводом должен подключаться строго к клемме XS2 (соблюдать полярность).**

При запуске программы длительность импульсов плавно нарастает с определенной скоростью, в нижней строке ЖК индикатора при этом выводится SPEED=X1. Кнопка с фиксацией, подключенная к выводу PA1 в этом режиме должна быть отжата, т. к. при нажатии кнопки скорость нарастания длительности импульсов увеличивается в 4 раза и в нижней строке ЖК индикатора отображается SPEED=X4. В верхней строке ЖК индикатора выводится величина Ust=..., которая ограничивает нижний порог ШИМ и задается потенциометром R1, средней точкой подключенному к выводу 40 (PA0) микроконтроллера от 0 до 127 уровней. Отметим, что генерируемый ШИМ имеет всего 255 градаций (уровней), и, например, при Ust=127 будет меняться от 127 до 255 единиц. При достижении максимального значения, длительность импульсов ШИМ плавно убавляется.

Оформление отчета

В отчете должны содержаться описание команд использованных в выполнении заданий и программы написанные по заданным в задании алгоритмам

Контрольные вопросы

1. Какой уровень сигнала у цифровых выходов МК?
2. Какие типы команд имеет микроконтроллер AVR семейства Mega?
3. Какие типы команд изменяют флаги микроконтроллера?
4. Какие виды адресации использует микроконтроллер AVR?
5. Зачем нужны команды передачи управления?
6. Как организована архитектура ядра микроконтроллеров AVR семейства Mega?

Список литературы, рекомендуемый к использованию по данной теме:

1. Клингман Э. Проектирование микропроцессорных систем. М.: Мир, 1988. - 575с.
2. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах.
3. Трамперт В. AVR-RISK микроконтроллеры.: Пер. с нем. – К.: «МК-ПРЕСС», 2006.-464с., ил.
4. Ю.А.Шпак. Программирование на языке С для AVR и PIC микроконтроллеров. – К.: «МК-ПРЕСС», 2006.-400с., ил.
5. Якубовский С.В. Цифровые и аналоговые интегральные микросхемы. М.:Радио и связь. 1990.
6. Токхейм Р. Основы цифровой электроники. Пер. с англ. - М.: Мир, 1988. – 390 с.
7. Шило В.Л. Популярныe цифровые микросхемы. – М.: Радио и связь, 1990.– 350 с.
8. Бирюков С.А. Цифровые устройства на МОП-интегральных микросхемах - М.: Радио и связь. 1992 (1996).
9. Опадчий Ю.Ф., Глудкин О.П., Гуров А.И. Аналоговая и цифровая электроника. Полный курс: учебник для вузов. - М.: Горячая линия, 1999 (2000, 2005). – 768 с.
10. Алексенко А.В., Шагуров И.И. Микросхемотехника.– М.: Радио и связь,1990 (1982).

Лабораторная работа 7

Разработка программ работы с аналоговыми сигналами микроконтроллера AVR с устройствами импульсной модуляции микроконтроллера AVR

Цель работы

Изучение и разработка программ работы с аналоговыми сигналами микроконтроллера AVR с устройствами импульсной модуляции микроконтроллера AVR.

Компетенции:

Индекс	Формулировка:
ОПК-3	способностью решать задачи анализа и расчета характеристик электрических цепей
ОПК-7	способностью учитывать современные тенденции развития теории автоматического управления, электронного оборудования, измерительной и вычислительной техники, информационных технологий в профессиональной деятельности
ПК-1	способностью выполнять эксперименты на действующих объектах по заданным методикам, оценивать динамические характеристики рассматриваемых объектов и идентифицировать передаточные функции объектов с применением современных информационных технологий и технических средств
ПК-7	готовностью участвовать в разработке и изготовлении стендов для комплексной отладки и испытаний программно-аппаратных управляющих комплексов
ПК-8	готовностью к внедрению результатов разработок средств и систем автоматизации и управления в производстве с использованием современных программируемых логических контролеров (ПЛК)
ПК-11	готовностью производить инсталляцию и настройку системного, прикладного и инструментального программного обеспечения систем автоматизации и управления
ПК-14	способностью разрабатывать электромеханические системы и использовать современную элементную базу при проектировании средств и систем управления

Теоретическая часть

Модуль 10-битного АЦП последовательного приближения входит в состав практически всех моделей семейства, за исключением ATmega8515x и ATmega162x. Основные параметры этого АЦП следующие:

- абсолютная погрешность: $\pm 2 \text{ LSB}$];
- интегральная нелинейность: $\pm 0.5 \text{ LSB}$;
- быстродействие: до 15 тыс. выборков/с.

На входе модуля АЦП имеется 8-канальный (в моделях ATmega640x/1280x/2560x — 16-канальный) аналоговый мультиплексор, предоставляющий в распоряжение пользователя 8 АБ) каналов с несимметричными входами. Кроме того, в моделях ATmega8x, выпускаемых в корпусе DIP, доступно только 6 каналов из восьми.

В большинстве моделей входы АЦП могут объединяться попарно для формирования различного числа каналов с дифференциальным входом. При этом в

некоторых каналах имеется возможность 10- и 200-кратного предварительного усиления входного сигнала. При коэффициентах усиления 1x и 10x действительная разрешающая способность АЦП по этим каналам составляет 8 бит, а при коэффициенте усиления 200x — 7 бит.

В качестве источника опорного напряжения для АЦП может использоваться как напряжение питания микроконтроллера, так и внутренний либо внешний источник опорного напряжения. Модуль АЦП может работать в двух режимах:

- режим одиночного преобразования, когда запуск каждого преобразования инициируется пользователем;
- режим непрерывного преобразования, когда запуск преобразований выполняется непрерывно через определенные интервалы времени.

9.2. Функционирование модуля АЦП

Обобщенная структурная схема модуля АЦП приведена на Рис. 9.1. В моделях ATmega8x и ATmega48x/88x/168x элементы, выделенные на рисунке серым цветом, и связанные с ними сигналы отсутствуют, а инвертирующий вход компаратора выборки-хранения подключен непосредственно к выходу мультиплексора (показано пунктиром).

Регистры, используемые для управления модулем АЦП в различных моделях, приведены в Табл. 9.1.

Таблица 9.1. Регистры управления модулем АЦП

Регистр	Адрес	ATmega8/32/64/128/168/256	ATmega8x	ATmega16x/32x	ATmega64x	ATmega128x	ATmega48x/88x/168x	ATmega164x/324x/644x	ATmega165x	ATmega325x/3250x, ATmega645x/6450x	ATmega640x, ATmega1280x/1281x, ATmega2560x/2561x	Описание
ADCSR	\$06 (\$26)		•									Регистр управления и состояния
ADCSRA	\$06 (\$26)	•		•	•	•						Регистр А управления и состояния
	(\$7A)						•	•	•	•	•	
ADCSRB	(\$8E)				•							Регистр В управления и состояния
	(\$7B)						•	•	•	•	•	
ADMUX	\$07 (\$27)	•	•	•	•	•						Регистр управления мультиплексором
	(\$7C)						•	•	•	•	•	
SFIOR	\$30 (\$50)	•	•	•								Регистр специальных функций
	\$20 (\$40)					•						

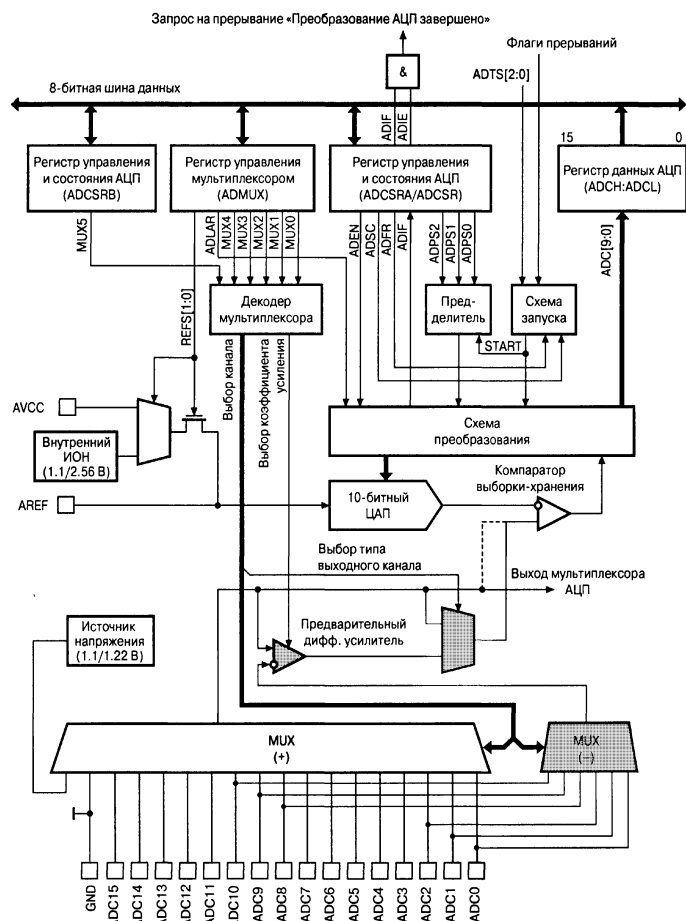


Рис. 9.1. Структурная схема модуля АЦП

Формат регистров ADCSRA (ADCSR) и ADMUX приведен на Рис. 9.2 и Рис. 9.3, а краткое описание функций их битов приведено в Табл. 9.2 и Табл. 9.3 соответственно.

	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ATmega8x ATmega128x
Чтение(R)/Запись(W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

	7	6	5	4	3	2	1	0	
	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	Остальные модели
Чтение(R)/Запись(W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Начальное значение	0	0	0	0	0	0	0	0	

ATmega8x — ADCSR
Остальные модели — ADCSRA

Рис. 9.2. Формат регистра ADCSRA (ADCSR)

Таблица 9.2. Биты регистра ADCSRA (ADCSR¹⁾)

Бит	Название	Описание
7	ADEN	Разрешение АЦП (1 – включено, 0 – выключено)
6	ADSC	Запуск преобразования (1 – начать преобразование)
5	ADATE (ADFR ²⁾)	Выбор режима работы АЦП
4	ADIF	Флаг прерывания от компаратора
3	ADIE	Разрешение прерывания от компаратора
2...0	ADPS2:ADPS0	Выбор частоты преобразования

¹⁾ В модели ATmega8x.
²⁾ В моделях ATmega8x и ATmega128x.

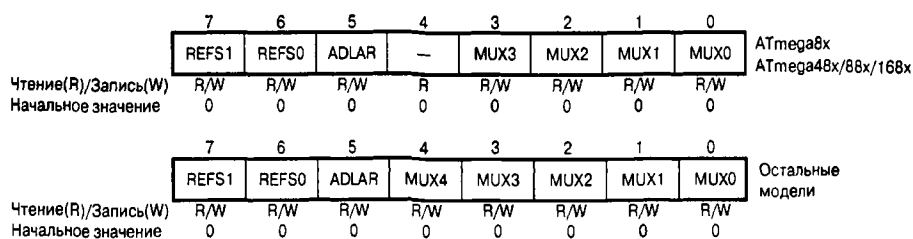


Рис. 9.3. Формат регистра ADMUX

Таблица 9.3. Биты регистра ADMUX

Бит	Название	Описание	Модель
7, 6	REFS1:REFS0	Выбор источника опорного напряжения	Все модели
5	ADLAR	Выравнивание результата преобразования	Все модели
4	—	Зарезервировано	ATmega8x, ATmega48x/88x/168x
	MUX4	Выбор входного канала	Остальные
3...0	MUX3...MUX0	Выбор входного канала	Все модели

Формат регистров ADCSRB и SFIOR приведен на Рис. 9.4 и Рис. 9.5 соответственно (неиспользуемые в данном случае биты регистра SFIOR указаны на рисунке как «X»).

Для разрешения работы АЦП необходимо записать лог. 1 в бит ADEN регистра ADCSR, а для выключения — соответственно лог. 0. Если АЦП будет выключен во время цикла преобразования, то преобразование завершено не будет (в регистре данных АЦП останется результат предыдущего преобразования).

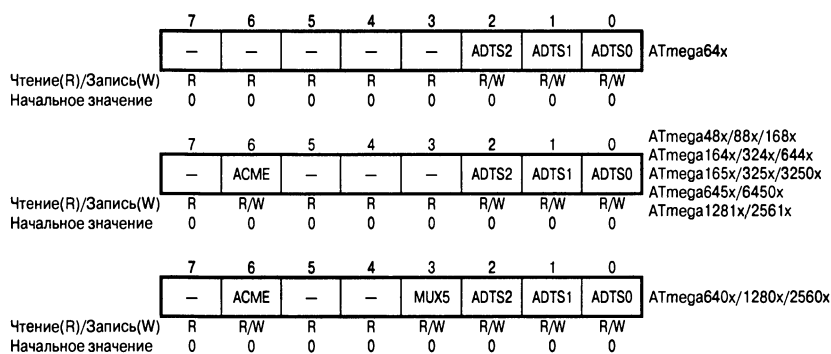


Рис. 9.4. Формат регистра ADCSRB

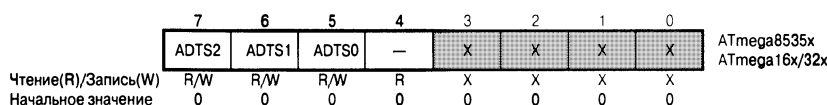


Рис. 9.5. Регистр SFIOR

В моделях ATmega8x и ATmega128x режим работы АЦП определяется состоянием бита ADFR. Если он установлен в 1, АЦП работает в режиме непрерывного преобразования. В этом режиме запуск каждого следующего преобразования осуществляется автоматически после окончания текущего. Если же бит ADFR сброшен в 0, АЦП работает в режиме одиночного преобразования и запуск каждого преобразования осуществляется по команде пользователя.

Во всех моделях, кроме ATmega8x и ATmega128x, запуск АЦП возможен не только по команде пользователя, но и по прерыванию от некоторых периферийных устройств, имеющих в составе микроконтроллера. Для выбора режима работы в этих моделях используется бит ADATE регистра ADCSRA и биты ADTS2:0 регистра SFIOR или ADCSRB (см. Табл. 9.1 и Рис. 9.4, 9.5).

В моделях ATmega8x и ATmega128x режим работы АЦП определяется состоянием бита ADFR. Если он установлен в 1, АЦП работает в режиме непрерывного преобразования. В этом режиме запуск каждого следующего преобразования осуществляется автоматически после окончания текущего.

Если же бит ADFR сброшен в 0, АЦП работает в режиме одиночного преобразования и запуск каждого преобразования осуществляется по команде пользователя.

Таблица 9.4. Источник сигнала для запуска преобразования

ADTS2	ADTS1	ADTS0	Источник стартового сигнала
0	0	0	Режим непрерывного преобразования
0	0	1	Прерывание от аналогового компаратора
0	1	0	Внешнее прерывание INT0
0	1	1	Прерывание по событию «Совпадение» («Совпадение А») таймера/счетчика T0
1	0	0	Прерывание по переполнению таймера/счетчика T0
1	0	1	Прерывание по событию «Совпадение В» таймера/счетчика T1
1	1	0	Прерывание по переполнению таймера/счетчика T1
1	1	1	Прерывание по событию «Захват» таймера/счетчика T1

Запуск каждого преобразования в режиме одиночного преобразования, а также запуск первого преобразования в режиме непрерывного преобразования осуществляется установкой в 1 бита ADSC регистра ADCSRA (ADCSR). Запуск преобразования по прерыванию осуществляется при установке в 1 флага выбранного прерывания. Бит ADSC регистра ADCSRA при этом аппаратно устанавливается в 1. Запуск преобразования в этих режимах также может быть осуществлен установкой бита ADSC регистра ADCSRA в 1.

В режимах одиночного и непрерывного преобразований цикл преобразования начинается по первому нарастающему фронту тактового сигнала после установки бита ADSC. Если используется запуск по прерыванию, то цикл преобразования начинается по первому нарастающему фронту тактового сигнала после установки флага выбранного прерывания. Причем в момент установки этого флага осуществляется сброс предделителя модуля АЦП. Тем самым обеспечивается фиксированная задержка между генерацией запроса на прерывание и началом цикла преобразования. Обратите внимание, что преобразование запускается при установке соответствующего флага, т. е. даже если само прерывание запрещено.

Длительность цикла составляет 13 тактов при использовании несимметричного входа и 13 либо 14 тактов при использовании дифференциального входа (разные значения связаны с работой схемы синхронизации); выборка и запоминание входного сигнала осуществляется в течение первых 1.5 и 2.5 тактов соответственно. Через 13 A4) тактов преобразование завершается, бит ADSC аппаратно сбрасывается в 0 (в режиме одиночного преобразования), и результат преобразования сохраняется в регистре данных АЦП. Одновременно устанавливается флаг прерывания ADIF регистра ADCSR и генерируется запрос на прерывание. Как и флаги остальных прерываний, флаг ADIF сбрасывается аппаратно при запуске подпрограммы обработки прерывания от АЦП или программно, записью в него лог. 1. Разрешение прерывания осуществляется установкой в 1 бита ADIF регистра ADCSR при установленном флаге I регистра SREG.

Если АЦП работает в режиме непрерывного преобразования, то новый цикл начнется сразу же после записи результата. В режиме одиночного преобразования новое преобразование может быть запущено сразу же после сброса бита ADSC (до сохранения результата текущего преобразования). Однако реально цикл преобразования начнется не

ранее чем через один такт после окончания текущего преобразования. Временные диаграммы, иллюстрирующие сказанное, приведены на Рис. 9.6.

При первом запуске после включения АЦП для выполнения преобразования потребуется 25 тактов, т. е. на 12 тактов больше, чем обычно. В течение этих 12 тактов выполняется «холостое» преобразование, во время которого производится инициализация АЦП (Рис. 9.7).

Отдельно следует сказать об использовании режима запуска по прерыванию совместно с дифференциальными каналами. В этом случае АЦП необходимо выключать между преобразованиями, чтобы избежать некорректных измерений, связанных с неопределенностью момента сброса предделителя АЦП. В результате выключения и включения АЦП между преобразованиями будут выполняться только «длинные» преобразования, результаты которых всегда будут корректными.

Для формирования тактовой частоты модуля АЦП в нем имеется отдельный предделитель. Коэффициент деления предделителя и соответственно длительность преобразования определяются состоянием битов ADPS2...ADPS0 регистра ADCSRA/ADCSR (см. Табл. 9.5).

Наибольшая точность преобразования достигается, если тактовая частота модуля АЦП находится в диапазоне 50...200 кГц. Соответственно, коэффициент деления предделителя рекомендуется выбирать таким, чтобы тактовая частота модуля АЦП находилась в указанном диапазоне. Если же точности преобразования меньше 10 битов достаточно, можно использовать более высокую частоту, увеличивая тем самым частоту выборки.

В зависимости от модели выводы микроконтроллера, подключенные к входу АЦП, определяются состоянием битов MUX3:0, MUX4:0 либо MUX5:0 (MUX5 - в ADCSRB, MUX4...MUX0 - в ADMUX). Для каналов с дифференциальным входом указанные биты определяют также коэффициент предварительного усиления входного сигнала. Соответствие установок этих битов используемым входам АЦП приведено в Табл. 9.6...9.8.

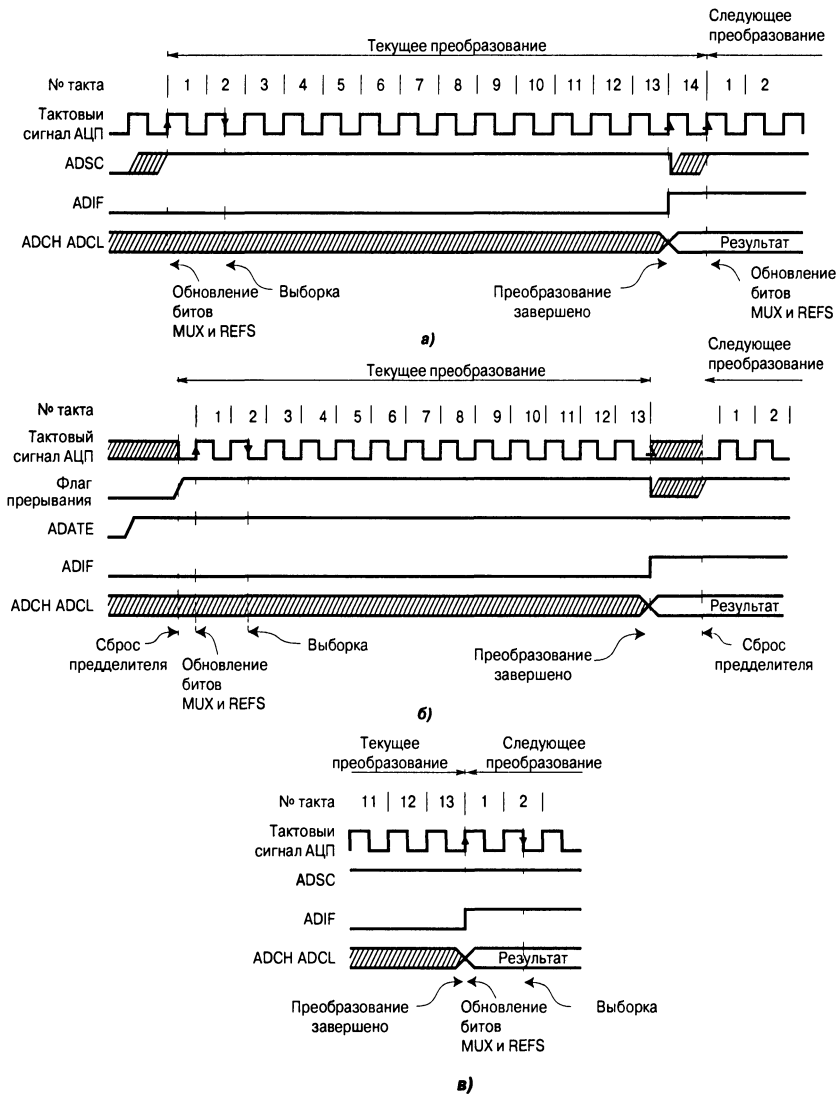


Рис. 9.6. Временные диаграммы работы АЦП в режиме одиночного преобразования (а), в режиме запуска по прерыванию (б) и в режиме непрерывного преобразования (в)

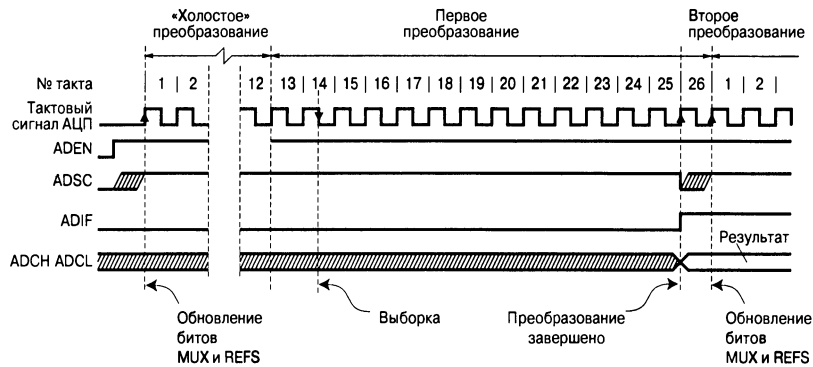


Рис. 9.7. Временные диаграммы работы АЦП при первом преобразовании (режим одиночного преобразования)

Таблица 9.5. Задание коэффициента деления предделителя АЦП

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

Таблица 9.6. Управление входным мультиплексором в моделях ATmega8x и ATmega48x/88x/168x

MUX3...MUX0	Несимметричный вход	MUX3...MUX0	Несимметричный вход
0000	ADC0	0110	ADC6 ¹⁾
0001	ADC1	0111	ADC7 ¹⁾
0010	ADC2	1000...1101	Зарезервировано
0011	ADC3	1110	1.22 В (1.1 В ²⁾)
0100	ADC4	1111	0 В (GND)
0101	ADC5		

¹⁾ Только в корпусах TQFP-32 и MLF-32.
²⁾ В моделях ATmega48x/88x/168x.

Таблица 9.7. Управление входным мультиплексором в моделях ATmega8535x, ATmega16x/32x/64x/128x, ATmega164x/324x/644x, ATmega165x/325x/3250x/645x/6450x и ATmega1281x/2561x

MUX4...MUX0	Несимметричный вход	Дифференциальный вход		Предварительное усиление
		положительный	отрицательный	
00000	ADC0			
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			
01000		ADC0	ADC0	10x
01001		ADC1	ADC0	10x
01010		ADC0	ADC0	200x
01011		ADC1	ADC0	200x
01100		ADC2	ADC2	10x
01101		ADC3	ADC2	10x
01110		ADC2	ADC2	200x
01111		ADC3	ADC2	200x
10000		ADC0	ADC1	1x
10001		ADC1	ADC1	1x
10010		ADC2	ADC1	1x
10011		ADC3	ADC1	1x
10100		ADC4	ADC1	1x
10101		ADC5	ADC1	1x
10110		ADC6	ADC1	1x
10111		ADC7	ADC1	1x
11000		ADC0	ADC2	1x
11001		ADC1	ADC2	1x
11010		ADC2	ADC2	1x
11011		ADC3	ADC2	1x
11100		ADC4	ADC2	1x
11101		ADC5	ADC2	1x
11110	1.22 В (1.1 В ¹⁾)			
11111	0 В (GND)			

¹⁾ В моделях ATmega165x/325x/3250x/645x/6450x и ATmega1281x/2561x

Таблица 9.8. Управление входным мультиплексором в моделях ATmega640х/1280х/2560х

MUX5...MUX0	Несимметричный вход	Дифференциальный вход		Предварительное усиление
		положительный	отрицательный	
000000	ADC0			
000001	ADC1			
000010	ADC2			
000011	ADC3			
000100	ADC4			
000101	ADC5			
000110	ADC6			
000111	ADC7			
001000		ADC0	ADC0	10x
001001		ADC1	ADC0	10x
001010		ADC0	ADC0	200x
001011		ADC1	ADC0	200x
001100		ADC2	ADC2	10x
001101		ADC3	ADC2	10x
001110		ADC2	ADC2	200x
001111		ADC3	ADC2	200x
010000		ADC0	ADC1	1x
010001		ADC1	ADC1	1x
010010		ADC2	ADC1	1x
010011		ADC3	ADC1	1x
010100		ADC4	ADC1	1x
010101		ADC5	ADC1	1x
010110		ADC6	ADC1	1x
010111		ADC7	ADC1	1x
011000		ADC0	ADC2	1x
011001		ADC1	ADC2	1x
011010		ADC2	ADC2	1x
011011		ADC3	ADC2	1x
011100		ADC4	ADC2	1x
011101		ADC5	ADC2	1x
011110	1.1 В			
011111	0 В (GND)			

Следует отметить, что предварительный усилитель, используемый каналами с дифференциальным входом, имеет встроенную схему коррекции смещения. Оставшаяся после коррекции величина смещения может быть учтена программным способом. Для этого следует оба входа дифференциального усилителя подключить к одному и тому же выводу микроконтроллера (Табл. 9.7 и Табл. 9.8), а затем вычитать полученное значение из результата последующих преобразований. Таким образом, ошибка смещения может быть снижена а до величины, меньшей 1 LSB.

Состояние битов MUX5...MUX0 можно изменить в любой момент, однако если это будет сделано во время цикла преобразования, то смена канала произойдет только после завершения преобразования. Благодаря этому в режиме непрерывного преобразования можно легко осуществлять последовательное преобразование сигналов нескольких каналов.

Отдельно следует сказать о каналах с дифференциальным входом. После смены таких каналов первое измерение следует производить не ранее чем через 125 мкс после выбора канала. Указанное время требуется для установления значения коэффициента усиления предусилителя. Соответственно, значения, измеренные до истечения этого

срока, не могут считаться достоверными. Кроме того, при переключении на канал с изменяемым коэффициентом усиления результат первого преобразования может иметь пониженную точность из-за времени установления автоматической схемы коррекции смещения. Поэтому будет лучше считать результат первого преобразования некорректным.

В новых микроконтроллерах, а именно в ATmega48x/88x/168x, ATmega164x/324x/644x, ATmega165x, ATmega325x/3250x/645x/6450x и ATmega640x/1280x/1281x/2560x/2561x, имеется возможность отключения входных цифровых буферов на выводах ADC0...ADC15 в случае, если соответствующие выводы используются только для считывания аналоговых сигналов. При отключенных цифровых буферах уменьшается общий ток потребления микроконтроллера, а соответствующие биты регистров PINx всегда читаются как 0.

Отключение цифровых буферов на входах ADC0...ADC7 осуществляется записью лог. 1 соответственно в биты ADC0D...ADC7D регистра DIDR0, расположенного по адресу (\$7E). А отключение буферов на входах ADC8...ADC15 (модели ATmega640x/1280x/2560x) осуществляется записью лог. 1 в биты ADC8D...ADC15D регистра DIDR2, расположенного по адресу (\$7D). Формат этих регистров приведен на Рис. 9.8.

Как уже было отмечено, в модуле АЦП могут использоваться различные источники опорного напряжения (ИОН). Выбор конкретного источника опорного напряжения осуществляется с помощью битов REFS1:REFS0 регистра ADMUX (см. Табл. 9.9).

Оборудование и материалы.

Комплекс лабораторных работ по исследованию микроконтроллеров AVR выполняется на учебном макетном лабораторном комплексе РТМТЛ-2.

На передней крышке прибора закреплена отладочная плата, содержащая исследуемый микропроцессор (Atmega8535); резистивно-диодный последовательный (работающий по интерфейсу RS232 через COM-порт ПК) программатор; обвязку микроконтроллера согласно паспортным данным; набор из 8 светодиодов, подключенных к выводам 1, 2, 3, 4, 18, 19, 20, 21 рис. 3.3; набор из 8 кнопок без фиксации нормально разомкнутых (кнопка нажата — замкнуто; отжата — разомкнуто); 1 потенциометр (переменный резистор), подключенный средней точкой к выводу 40 АЦП микроконтроллера; два семисегментных индикатора; выходы «НАГРУЗКА» XS — XS2; а также кнопку с фиксацией «RESET», при нажатии которой вывод RESET кристалла соединяется с корпусом схемы. Для связи учебного стенда с ПК (передача данных) используется микросхема типа MAX232 (рис. 3.1) и разъем «ДАННЫЕ», расположенный на задней крышке прибора. Для программирования прибора используется разъем «ПРОГРАММАТОР». При этом «ПРОГРАММАТОР» следует подключать к COM – порту ПЭВМ

ТОЛЬКО проводом типа COM 9m/9f («мама» - «папа»), разъем «ДАННЫЕ» может быть подключен непосредственно как к COM порту ПК проводом COM 9m/9f («мама» - «папа»), так и к USB порту ПК через переходник USB-COM (RS232), собранный на микросхеме Prologic PL2303 (либо аналог.) рис. 3.2.

Схема связи микроконтроллера AVR с платой mini-ITX через COM порт (RS-232)

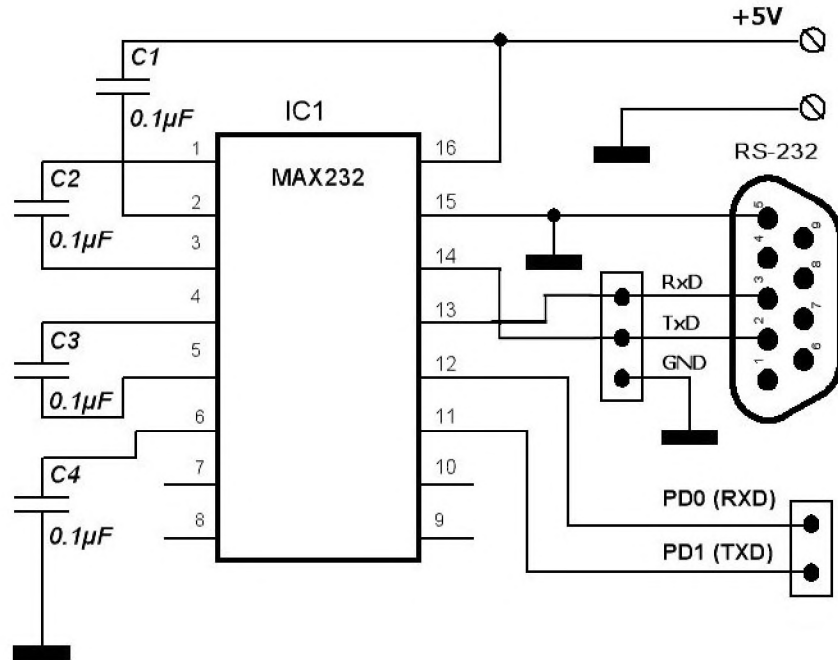


Рис. 3.1. Связь микроконтроллера с COM портом ПК через микросхему MAX232.
МК AVR Atmega8535 РАЗЪЁМ «ДАННЫЕ»

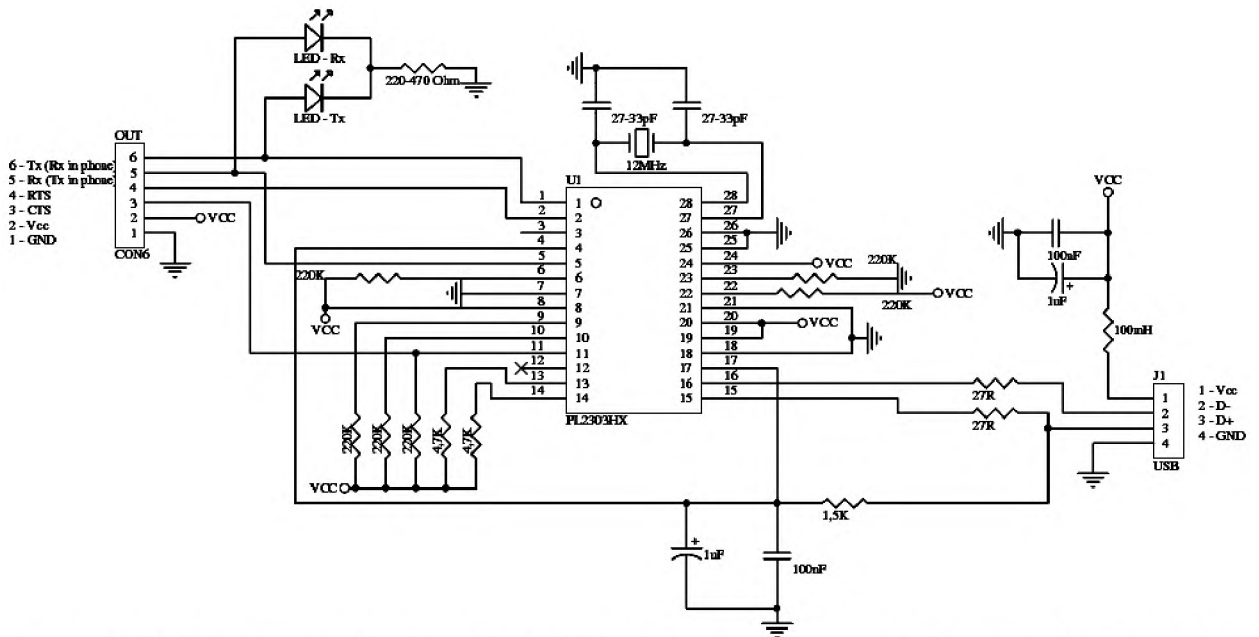


Рис. 3.2. Переходник USB-COM (RS232), собранный на микросхеме Prologic PL2303 (микросхема внутри кабеля USB – COM).

Принципиальная электрическая схема прибора приведена на рис. 3.3.

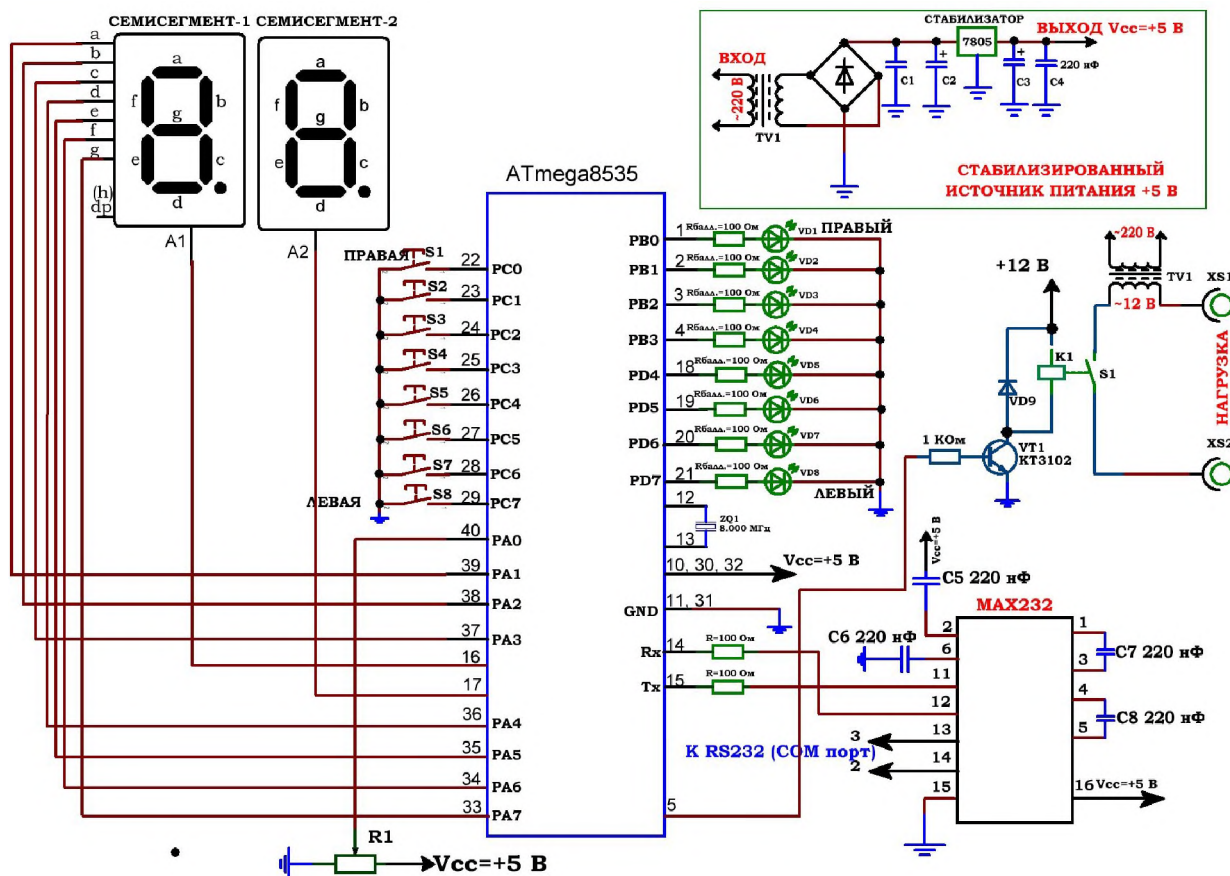


Рис. 3.3 Принципиальная электрическая схема прибора

Демонстрационные примеры готовых программ, на основе которых могут быть построены задания по программированию микроконтроллера, приведены в папке Examples_RTMTL-2.

1) Пример 1. По нажатию кнопки S1, подключенной к 22 выводу МК начинает светиться светодиод VD1, подключенный к выводу 1 микроконтроллера. По отжатию кнопки, светодиод перестаёт светиться (индикация работы кнопки);

2) Пример 2. При нажатии кнопки S1, подключенной к 22 выводу МК начинает мигать светодиод VD1, подключенный к выводу 1 микроконтроллера;

3) Пример 3. Мигание светодиода VD1, подключенного к выводу 1.

4) Пример 4. Светодиоды VD1 – VD8 мигают по очереди (по порядку подключения к выводам 1, 2, 3, 4, 18, 19, 20, 21 МК);

5) Пример 5. Светодиоды VD1 – VD8 мигают по очереди в той же последовательности с одновременной передачей по USART через микросхему MAX232 на разъём «ДАННЫЕ» (выход на COM порт) номера включенного светодиода в ASCII коде (натуральное число);

6) Пример 6. Изучение работы АЦП микроконтроллера. Напряжение считывается с переменного резистора R1, подключённого средней точкой к 40 выводу МК (вход АЦП) в код АЦП от 0 до 63 единиц и с интервалом ~ 1 сек. передаётся по USART через микросхему MAX232 на разъём «ДАННЫЕ» (выход на COM порт);

7) Пример 7. Приёмник RS232 с ПК. Передавая ASCII код 1 ... 8 (натуральные числа) с ПК посредством программы «ТЕРМИНАЛ» можно управлять включением светодиодов;

8) Пример 8. Работа с семисегментными индикаторами, RS232 и реле. Реализация таймера обратного отсчета. Текущие показания таймера выводятся на семисегментные индикаторы с общими катодами (a, b, c, d, e, f, g) и анодами A1 и A2, и одновременно передаётся по USART через микросхему MAX232 на разъём «ДАННЫЕ» (выход на

COM порт). При достижении конца отсчета срабатывает реле К1, подключённое через транзисторный усилитель на КТ3102 к выводу 5 микроконтроллера и вторичная обмотка трансформатора TV1 (~12 В) соединяется с клеммами XS1 – XS2 (нагрузка). К выходу нагрузки можно подключать лампу накаливания 12 В, 21 Вт либо меньшей мощности.

Запрещается замыкать выходы контрольных точек XS1 – XS2 (нагрузка)!;

9) Пример 9. Реализация вольтметра. Работа с семисегментными индикаторами и реле. Простейшая реализация вольтметра с выводом показаний на семисегментный индикатор. Напряжение считывается с переменного резистора R1, подключённого средней точкой к 40 выводу МК (вход АЦП) и выводится на семисегментный индикатор в показаниях 0-50 (0-5 Вольт, т. к. вывод h(точка dp) не подключен). При определенном пороге (определенном значении напряжения) срабатывает реле К1 и вторичная обмотка трансформатора TV1 (~12 В) соединяется с клеммами XS1 – XS2 (нагрузка). К выходу нагрузки можно подключать лампу накаливания 12 В, 21 Вт либо меньшей мощности. Запрещается замыкать выходы контрольных точек XS1 – XS2 (нагрузка)!

Указания по технике безопасности

Соответствуют технике безопасности по работе с компьютерной техникой.

Ход работы

1. Перед включением установки в сеть проверить целостность всех соединительных сигнальных и сетевых проводов. Все работы по подключению комплекса к компьютеру следует выполнять только при отключенных от сети приборах. Разобраться с принципиальными блок-схемами опытов, в назначении кнопок, переключателей и ручек прибора.

2. **Перед выполнением работы следует изучить инструкцию по работе с учебной средой программирования Algorithm Builder, а также ознакомиться с полным методическим руководством по микроконтроллерам семейства AVR.**

3. Соединить монитор с системным блоком ПЭВМ, подключить клавиатуру и мышь к системному блоку используя стандартные провода для подключения. Подключить системный блок ПЭВМ и монитор к сети ~220 В.

4. Загрузить операционную систему согласно стандартным процедурам загрузки.

5. При необходимости, настроить компьютер для работы с учебной установкой и программной средой Algorithm Builder.

6. Запустить программу Algorithm Builder для работы с учебной установкой для данного эксперимента пользуясь ярлыком на рабочем столе либо другим способом, указанным лаборантом. **Для работы можно воспользоваться комплексной программной оболочкой LabVisual для РТМТЛ-1-5.**

6. Подключить разъём «ПРОГРАММАТОР» учебного прибора к COM – порту ПЭВМ проводом типа COM 9m/9f («мама» - «папа»)

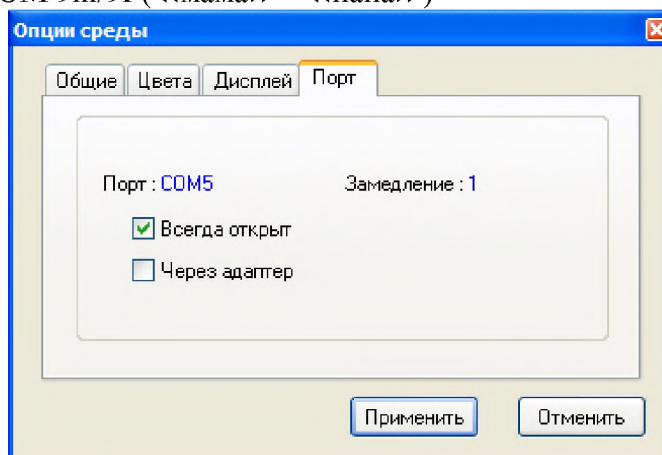


Рис. 4.1

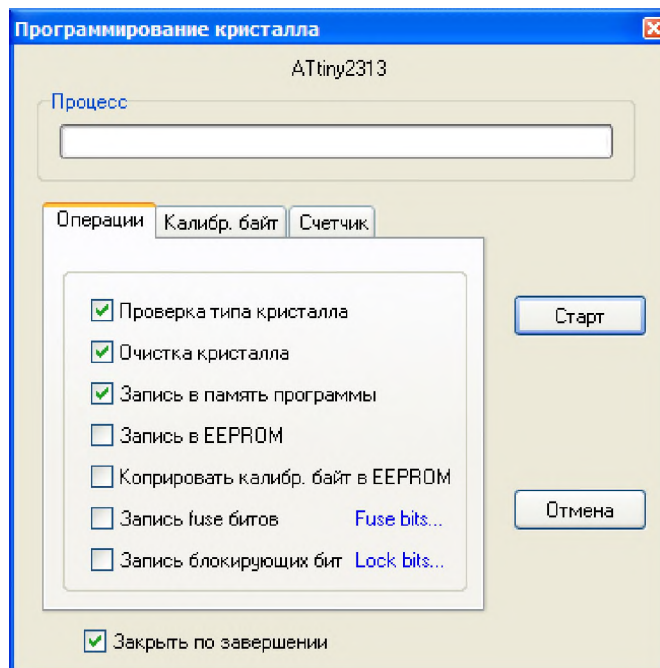


Рис. 4.2

7. Настроить программу Algorithm Builder для работы с данным COM портом ОПЦИИ-ОПЦИИ СРЕДЫ-ПОРТ рис. 4.1.

8. Для подробных инструкций следует обращаться к методическому руководству по среде Algorithm Builder.

9. Загрузить Пример 1 (папка 1) в среду Algorithm Builder (ФАЙЛ-ОТКРЫТЬ-ВЫБРАТЬ ФАЙЛ *.alp).

10. «Прошить» тестовую программу в кристалл. Для этого нажать кнопку «ЗАПУСК С КРИСТАЛЛОМ», при этом откроется диалог программирования кристалла рис. 4.2.

Перед прошивкой следует нажать кнопку с фиксацией $\#RESET\#$ на учебном приборе.

11. Перед нажатием кнопки «СТАРТ» следует внимательно проверить положения флажков установки программатора рис. 4.2. **Особенно внимательно следует отнестись к опциям $\#ЗАПИСЬ Fuse bit\#$ и $\#ЗАПИСЬ блокирующих бит\#$. ФЛАЖКИ ЭТИХ ОПЦИЙ ОБЯЗАТЕЛЬНО ДОЛЖНЫ БЫТЬ СНЯТЫ, Т. К. НЕ ПРАВИЛЬНО ПРОШИТЫЕ FUSE БИТЫ МОГУТ ПРИВЕСТИ К ДАЛЬНЕЙШЕЙ НЕВОЗМОЖНОСТИ РАБОТЫ С ДАННЫМ МИКРОКОНТРОЛЛЕРОМ!**

12. **Отжать кнопку RESET и проверить работу тестовой программы.**

13. Загрузить примеры 2 — 9 из соответствующих папок и прошить в кристалл каждую из тестовых программ.

14. При работе примеров 5 — 7 следует соединить разъём «ДАННЫЕ» с ПК.

Подключение может быть произведено непосредственно как к COM порту ПК проводом COM 9m/9f («мама» - «папа»), тем же, что вы использовали для программирования), так и к USB порту ПК через переходник USBCOM (RS232).

15. При подключении разъёма «ДАННЫЕ» через переходник USB-COM (RS232) в системе в диспетчере устройств появляется виртуальный COM- порт рис. 4.3.

16. Перед дальнейшей работы следует выяснить и запомнить номер виртуального COM-порта, присвоенного системой (в примере рис. 4.3 COM33)

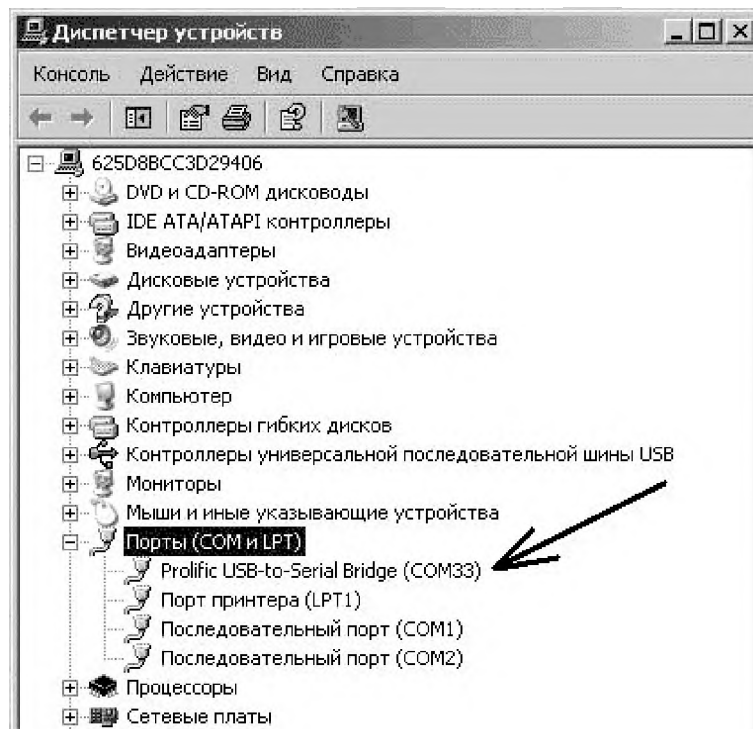


Рис. 4.3

17. Для анализа данных приходящих на СОМ порт ПК, а также для отправки байт управления на учебный стенд следует использовать программу «ТЕРМИНАЛ» для работы с СОМ – портом или аналогичную рис. 4.4.

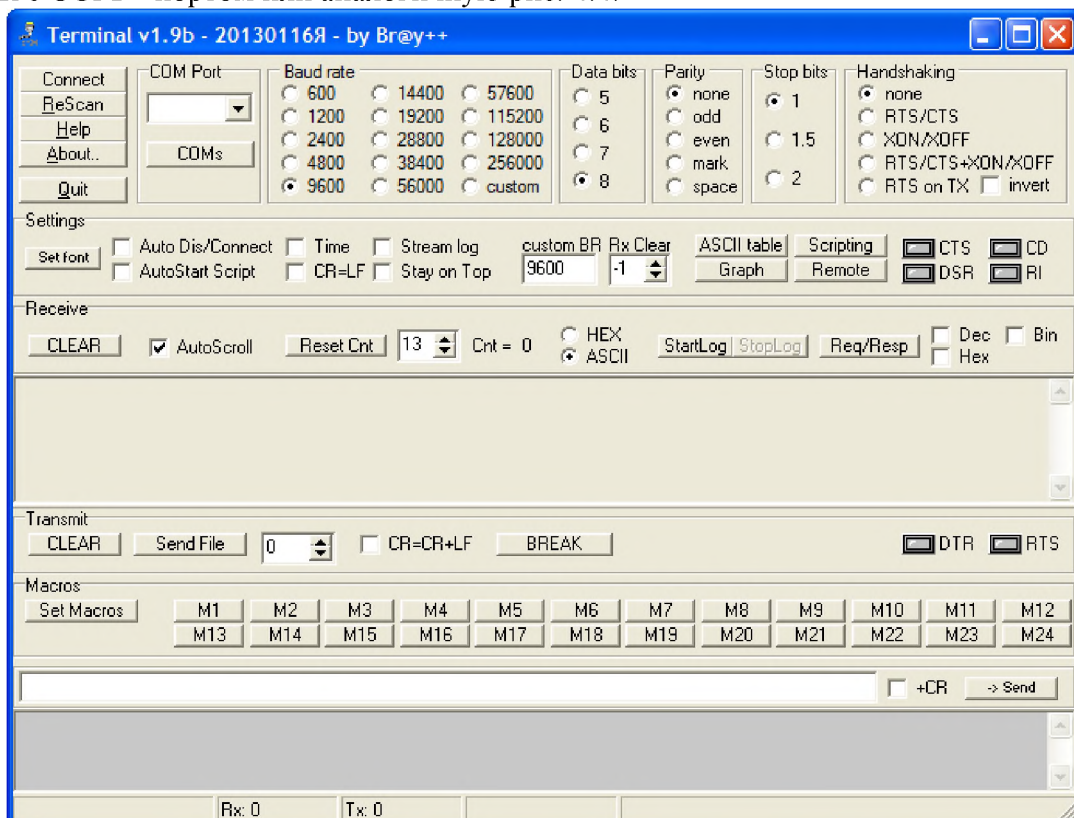


Рис. 4.4

18. В программе следует выбрать СОМ порт (либо виртуальный СОМ порт), к которому подключен разъём «ДАННЫЕ» и нажать кнопку «Connect».

19. Принятые байты данных можно наблюдать в окне терминала, для отправки байтов следует записать отправляемые данные в строку для отправки и нажать кнопку ->Send.

20. Исходя из готовых примеров составить собственные программы в среде Algorithm Builder.

21. На основании Примеров 1 — 4 можно составить программы, изменив, например, действия, осуществляемые кнопкой S1 (включать другой светодиод), либо изменить назначение других кнопок S2 – S8. Также можно изменить скорость мигания, порядок мигания светодиодов и проводить различные операции с микроконтроллером.

22. На основании Примеров 5 — 8 следует изучить работу с COM портом ПК, составляя и отлаживая различные программы для приёма — отправки данных по интерфейсу RS232.

Строка отправляемых данных

23. На основании примера 6,9 следует изучить работу АЦП микроконтроллера и реализацию динамической индикации на семисегментном индикаторе. На основании примера 8 изучить работу реле (порог срабатывания), изменить этот порог. Вывести результаты измерений по протоколу RS232 на COM порт (разъем данные).

24. На основании примера 8 изучить работу таймера, реализовать обратный отсчет таким образом, чтобы время отсчета (интервал) соответствовало строго 1 секунде (в примере время счета выбрано произвольно). Изучить работу с реле.

25. Для составления программ и подробного изучения работы микроконтроллеров AVR следует обращаться к учебной литературе, полному руководству к микроконтроллерам семейства AVR, а также к документации по среде Algorithm Builder.

26. По окончании работы следует закрыть программу и все открытые подпрограммы, закрыть виртуальную среду VirtualBox (при работе в среде Linux).

27. Выключить компьютер, нажав на кнопку, находящуюся в крайнем нижнем левом углу экрана. Из доступных действий выбрать «ВЫХОД»--> «ВЫКЛЮЧИТЬ КОМПЬЮТЕР».

28. Отключить установку от сети, поставив переключатели «СЕТЬ» на панели установки в положение «ВЫКЛ» и вынуть сетевые вилки из розеток__

Оформление отчета

В отчете должны содержаться описание команд использованных в выполнении заданий и программы написанные по заданным в задании алгоритмам

Контрольные вопросы

1. Какой предельный уровень сигнала у аналоговых входов МК?
2. Какие регистры и каким образом необходимо инициализировать для работы АЦП и ЦАП микроконтроллера AVR семейства Mega?
3. Каким образом происходит запуск преобразования в АЦП и ЦАП микроконтроллера AVR семейства Mega?
4. Какие выводы для АЦП использует микроконтроллер AVR семейства Mega?
5. Какие выводы для ЦАП использует микроконтроллер AVR семейства Mega?
6. Какие сигналы говорят об окончании процесса преобразования аналогового сигнала в цифровой?

Список литературы, рекомендуемый к использованию по данной теме:

1. Клингман Э. Проектирование микропроцессорных систем. М.: Мир. 1988. - 575с.
2. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах.
3. Трамперт В. AVR-RISK микроконтроллеры.: Пер. с нем. – К.: «МК-ПРЕСС», 2006.-464с., ил.
4. Ю.А.Шпак. Программирование на языке С для AVR и PIC микроконтроллеров. – К.: «МК-ПРЕСС», 2006.-400с., ил.

5. Якубовский С.В. Цифровые и аналоговые интегральные микросхемы. М.: Радио и связь. 1990.
6. Токхейм Р. Основы цифровой электроники. Пер. с англ. - М.: Мир, 1988. – 390 с.
7. Шило В.Л. Популярныe цифровые микросхемы. – М.: Радио и связь, 1990.– 350 с.
8. Бирюков С.А. Цифровые устройства на МОП-интегральных микросхемах - М.: Радио и связь. 1992 (1996).
9. Опадчий Ю.Ф., Глудкин О.П., Гуров А.И. Аналоговая и цифровая электроника. Полный курс: учебник для вузов. - М.: Горячая линия, 1999 (2000, 2005). – 768 с.
10. Алексенко А.В., Шагуров И.И. Микросхемотехника.– М.: Радио и связь, 1990 (1982).

Лабораторная работа 8

Разработка программ вывода информации на ЖКИ микроконтроллера AVR и программ для связи микроконтроллера AVR с ПК-ЭВМ

Цель работы

Изучение инструментальных средств разработки программного обеспечения МК.
Разработка программ вывода информации на ЖКИ микроконтроллера AVR и программ для связи микроконтроллера AVR с ПК-ЭВМ.

Компетенции:

Индекс	Формулировка:
ОПК-3	способностью решать задачи анализа и расчета характеристик электрических цепей
ОПК-7	способностью учитывать современные тенденции развития теории автоматического управления, электронного оборудования, измерительной и вычислительной техники, информационных технологий в профессиональной деятельности
ПК-1	способностью выполнять эксперименты на действующих объектах по заданным методикам, оценивать динамические характеристики рассматриваемых объектов и идентифицировать передаточные функции объектов с применением современных информационных технологий и технических средств
ПК-7	готовностью участвовать в разработке и изготовлении стендов для комплексной отладки и испытаний программно-аппаратных управляющих комплексов
ПК-8	готовностью к внедрению результатов разработок средств и систем автоматизации и управления в производстве с использованием современных программируемых логических контролеров (ПЛК)
ПК-11	готовностью производить инсталляцию и настройку системного, прикладного и инструментального программного обеспечения систем автоматизации и управления
ПК-14	способностью разрабатывать электромеханические системы и использовать современную элементную базу при проектировании средств и систем управления

Теоретическая часть

По отношению к обыкновенным 7-сегментным индикаторам (рис. 1.1), ЖКИ модули на базе контроллера HD44780 (рис. 1.2) обладают на порядок большими возможностями. Количество строк на экране у разных моделей - 1,2 или 4; число символов в строке:

8,10,16,20,24,30,32 или 40.

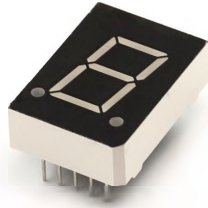


Рис 1.1

Контроллер HD44780 (а также совместимый с ним KS0066)— стандарт де-факто на контроллеры монохромных жидкокристаллических знаковсинтезирующих дисплеев с параллельным 4- или 8-битным интерфейсом.

Индикатор может иметь светодиодную или люминесцентную подсветку практически любого цвета свечения. На рис. 1.2 представлен ЖКИ индикатор производства фирмы WinSTAR, на базе контроллера HD44780. Аналогичный вид имеют и другие ЖК символьные индикаторы, серийно выпускаемые промышленностью.



Рис.1.2

Напряжение питания контроллера HD44780 5В (реже 3В). Ток потребления контроллера очень мал(100...200 мкА), чего не скажешь о светодиодной подсветке. В зависимости от производителя, его величина составляет 80...120 мА.

Для работы некоторых типов ЖКИ может потребоваться дополнительный источник напряжения отрицательной полярности. Технология производства модулей подобного рода непрерывно совершенствуется, что, в целом, положительно сказывается на их размерах и электрических характеристиках.

Символьный ЖКИ с контроллером HD44780 (KS0066). Интерфейс.

Символьный ЖКИ представляется матрицей из точек, разделенной на строки и поля символов (рис. 2.1):



Рис. 2.1

Для управления этой матрицей и вывода собственно символов используется специальный контроллер HD44780.

Контролер ЖКИ оперирует 3-мя блоками памяти:

Для вывода символа контроллер использует память **DDRAM** (Display Data RAM), где хранятся ASCII-коды символов, которые мы хотим видеть на ЖКИ.

Под нее отведено 80 ячеек памяти. Понятно, что на ЖКИ мы увидим лишь часть символов, которые находятся в DDRAM - если наш ЖКИ 1 или 2- строчный и отображает 8 символов в строке, то так:

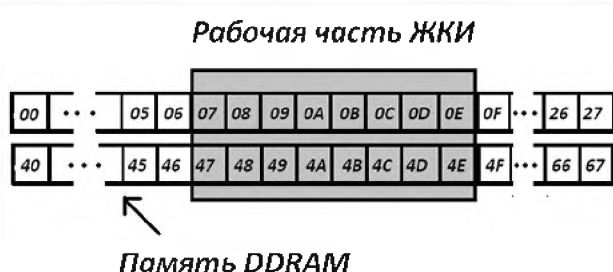


Рис. 2.2

Рабочую область дисплея, как видно, можно смещать по ячейкам DDRAM (получается эффект бегущей строки).

2. Шаблоны самих символов контроллер берет из CGROM (Character Generator ROM) – памяти знакогенератора. Таблицу символов можно посмотреть в спецификации на HD44780.

3. Для хранения пользовательских символов (их шаблонов) предусмотрена память CGRAM (Character Generator RAM).

Также, контроллер в зависимости от некоторых условий распределяет пришедшие в него данные в регистр инструкций или регистр данных.

Контрастность изображения на ЖКИ можно изменять, подключив дополнительно построочный резистор на 10 кОм по схеме рис. 2.3.

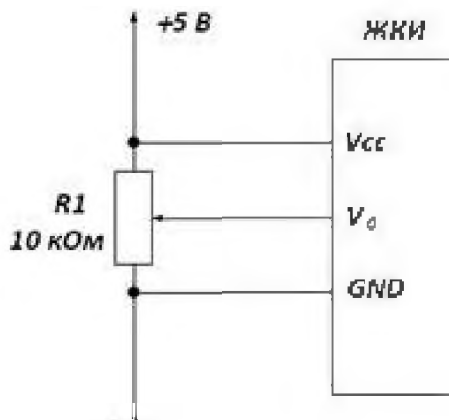


Рис. 2.3

Но следует смотреть в спецификацию на свой контроллер (например у ЖКИ K1sn10294v-0 на чипе KS0066 1-Vcc, а 2-GND). Подача питания подсветки может различаться от модели к модели в зависимости от её типа. Подсветка питается от 5 вольт, токоограничительный резистор (50-100 Ом) обычно не обязателен.

Назначение выводов R/S, R/W, E.

При переходе E с высокого логического уровня на низкий данные, которые уже «висят» на выводах DB0..DB7, записываются в память контроллера ЖКИ для последующей обработки.

При высоком лог. уровне на R/S(Register Select) контроллер ЖКИ воспринимает этот набор битов как данные(код символа), а при низком – как инструкцию и направляет их в соответствующий регистр.

R/W определяет направление работы выводов DB0..DB7 – если на R/W «0», то мы можем только писать в порт DB, а если R/W = «1», то можем прочитать с него (например

узнать занят контроллер или свободен для приема новых данных). Если мы не будем читать данные из ЖКИ, то можно «посадить» R/W на землю.

Набор команд HD44780

Для того чтобы начать выводить информацию на ЖКИ, его контроллер надо проинициализировать (сообщить ему об интерфейсе, шрифте, смещениях и т.д.). Контроллер может воспринимать всего 11 команд, приведенных в таблице 1.

Таблица

Название инструкции	Состояние выводов										Прим.	Время исполнения $f_{\text{раб}} = 270$ кГц
	RS	RW	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear display	0	0	0	0	0	0	0	0	0	1	Очистка Всего ЖКИ установка адреса DDRAM в 0	1.52 мс
Return home	0	0	0	0	0	0	0	0	1	X	Установка текущего адреса DDRAM в 0 (курсор – домой) Данные DDRAM не меняются	1.52 мс
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Установка направления движения курсора (I/D) и смещения дисплея (S) при выводе данных	37 мкс
Display on/off control	0	0	0	0	0	0	1	D	C	B	Вкл/выкл. дисплей(D), курсор(C) и его мерцание(B)	37 мкс
Cursor or display shift	0	0	0	0	0	1	S/C	R/L	X	X	Двигает курсор и смещает дисплей по DDRAM	37 мкс
Function set	0	0	0	0	1	DL	N	F	X	X	Установка интерфейса(DL), число строк(N) и шрифт символов(F)	37 мкс
Set CGRAM address	0	0	0	1	ACG5	ACG4	ACG3	ACG2	ACG1	ACG0	Установка счетчика адреса CGRAM. После этого можно записывать данные в CGRAM	37 мкс
Set DDRAM address	0	0	1	ADD6	ADD5	ADD4	ADD3	ADD2	ADD1	ADD0	Установка счетчика адреса DDRAM	37 мкс
Read busy flag & address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Если BF = 1 то контроллер ЖКИ выполняет внутреннюю операцию (занят). AC6-AC0 – текущее значение адреса DDRAM	0 мкс
Write data to RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Запись данных в RAM	37 мкс
Read data from RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Чтение данных из RAM	37 мкс

1

I/D = 1: адрес DDRAM увеличивается;

I/D = 0: уменьшается;

S = 1: сдвиг рабочей области дисплея по DDRAM разрешен;
D = 1: дисплей (изображение) включен;
C = 1: курсор включен;
B = 1: мерцание курсора включено;
S/C = 1: сдвинуть дисплей;
S/C = 0: переместить курсор
R/L = 1: вправо R/L = 0: влево;
DL = 1: 8 bit; DL = 0: 4 bits;
N = 1: 2 lines N = 0: 1 line;
F = 1: 5x10; F = 0: 5x8;
ACG: CGRAM address;
ADD: DDRAM address (адрес курсора);
AC: Address counter DD и CGRAM адресов

Инициализация ЖКИ.

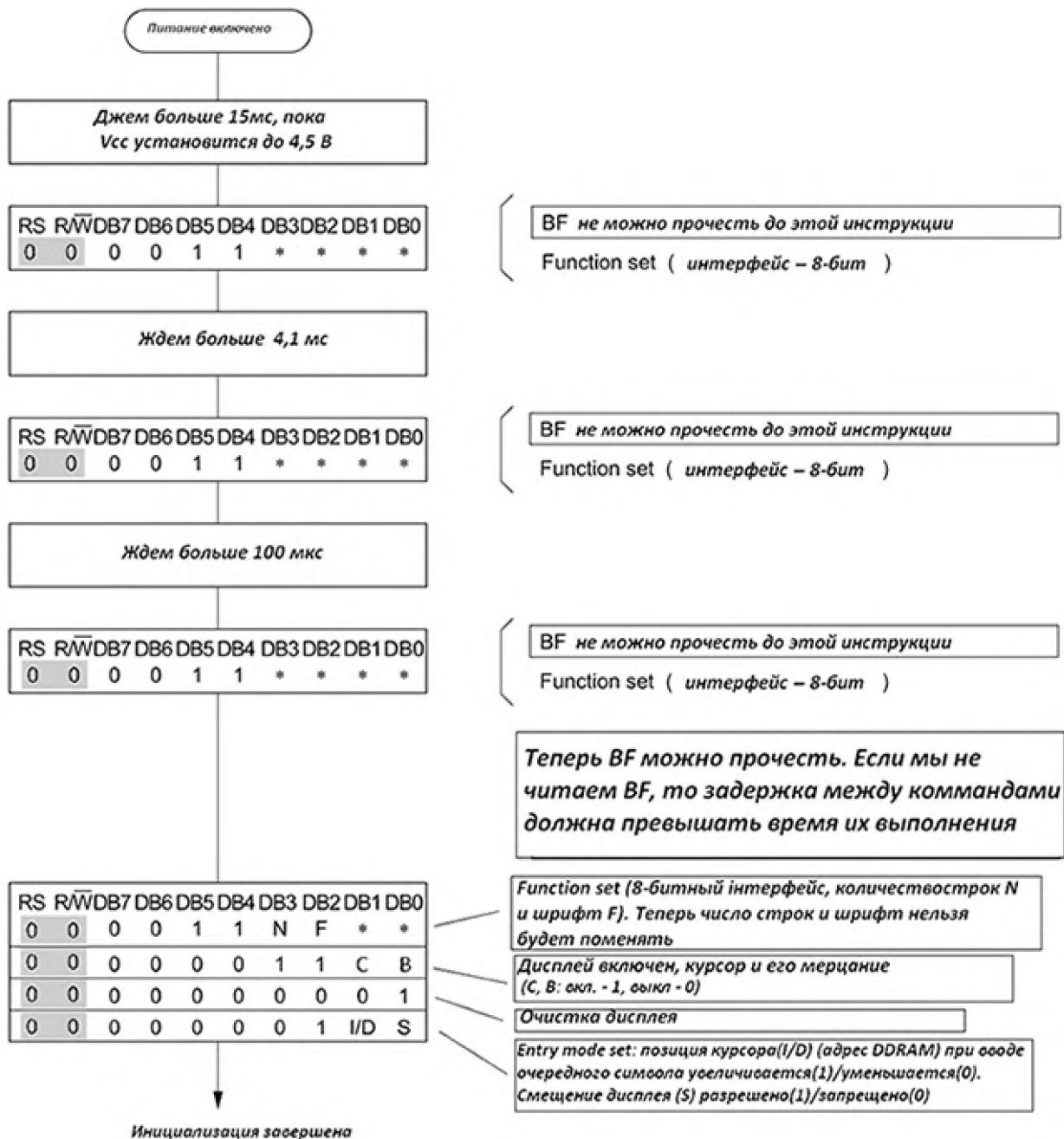
Есть 2 способа инициализации контроллера ЖКИ:

1. Через внутреннюю схему сброса.
2. В ручном режиме (через посылку в него ряда команд, которыми мы задаем режим работы ЖКИ) Внутренняя схема сброса контроллера начинает работать сразу после включения питания. В этом есть один минус – если питание у нас «ползет» до рабочего уровня медленно (медленнее, чем за 10 мс), то самоинициализация контроллера может пройти некорректно. При этом способе инициализации контроллер сам исполняет следующие команды:

1. Display clear
2. Function set:
DL = 1; 8-bit interface data
N = 0; 1-line display
F = 0; 5x8 dot character font
3. Display on/off control:
D = 0; Display off
C = 0; Cursor off
B = 0; Blinking off
4. Entry mode set:
I/D = 1; Increment by 1
S = 0; No shift

Второй способ исключает зависимость схемы от источника питания. Для инициализации контроллера ЖКИ в ручном режиме необходимо исполнить следующий алгоритм:

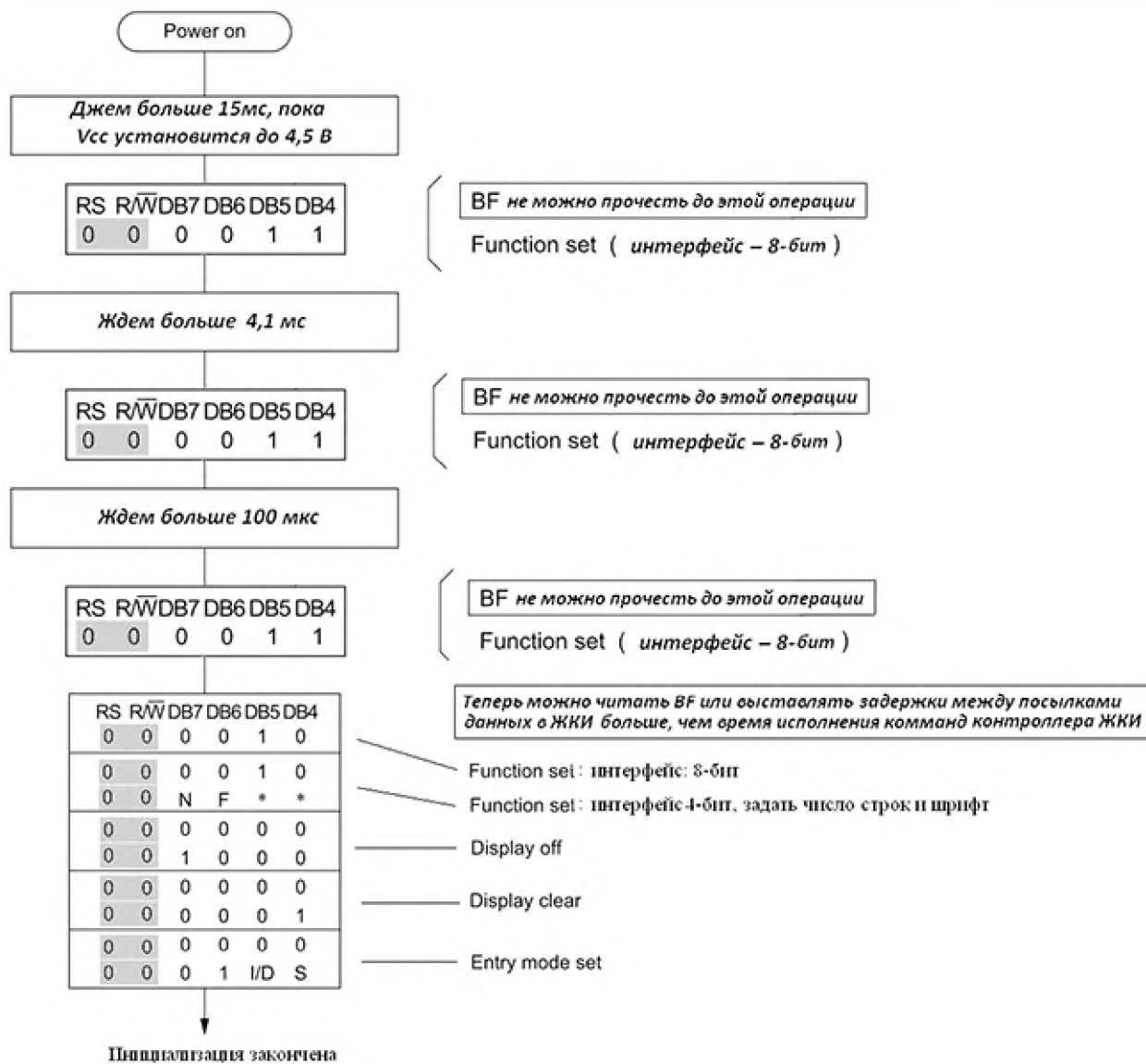
8-битный параллельный интерфейс:



4-битный параллельный интерфейс:

Как видно, здесь нет ничего сложного: посылаем в ЖКИ команду за командой, учитывая время их исполнения (около 40 мкс) или проверяя флаг занятости контроллера ЖКИ (тогда надо посадить пин RW на лапку микроконтроллера и выставлять его в «1», когда хотим узнать, занят ЖКИ или нет).

4-битный параллельный интерфейс:



Принцип работы ЖК модуля.

Типичный внешний вид ЖК символьного модуля приведен на рис. 3.1.

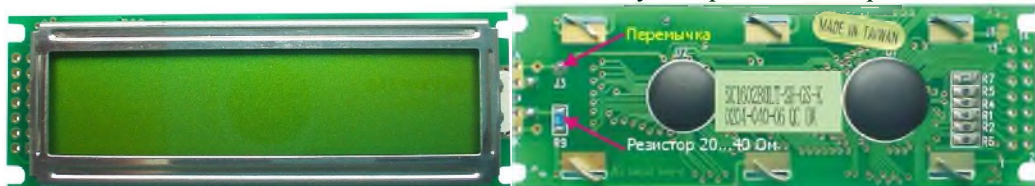


Рис. 3.1. Вид спереди и вид сзади ЖК модуля.

Как уже говорилось выше, изначально HD44780 имеет predetermined таблицу символов, размещенную в ОЗУ знакогенератора CGRAM (Character Generator RAM). Для отображения любого из них программа микроконтроллера должна передать координаты позиции и, непосредственно за ними, сам адрес символа из CGRAM. Пример таблицы CGRAM приведен на рис. 3.2.

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F		
0			0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1		!	"	#	\$	%	&	'	()	*	:	;	<	=	-	.	/
2		"	2	B	R	b	r				Ё	ё	Ъ	ъ	Ш	ш		
3		#	3	C	S	c	s				Ж	ж	Ы	ы	Щ	щ		
4		\$	4	D	T	d	t				Э	э	Ь	ь	Ъ	ъ		
5		%	5	E	U	e	u				И	и	Э	э	Ц	ц		
6		&	6	F	V	f	v				Й	й	Ю	ю	Щ	щ		
7		'	7	G	W	g	w				Л	л	Я	я	І	і		
8		(8	H	X	h	x				П	п	«	»	ІІ	іі		
9)	9	I	Y	i	y				У	у	»	«	~			
A		*	:	J	Z	j	z				Ф	ф	«	»	↓	↑		
B		+	:	K	[k]				Ч	ч	”	”	Н	н		
C		,	<	L	φ	l	φ				Ш	ш	Ъ	ъ	И	и		
D		-	=	M]	m]				Ь	ь	Ъ	ъ	Н	н		
E		.	>	N	^	n	^				Ы	ы	ф	ф	Ъ	ъ		
F		/	?	O	_	o	_				Э	э	£	£	•	•		

Рис3.2

Заглавные и прописные буквы латинского алфавита, числовые знаки, а также большинство знаков препинания совпадают в ней с кодами ASCII. Набор символов, размещенных по адресам 0xA0...0xFF, содержит национальный алфавит (в данном случае кириллицу) того региона, где предполагается его использование. Первые 16 ячеек CGRAM имеют особое значение. При желании, в них могут быть записаны любые пользовательские символы, которых нет таблице (сразу после включения модуля в них находится случайная информация).

Типичная нумерация и функциональное назначение выводов ЖКИ приведены в табл.2. Кроме напряжения питания контроллера VCC, модуль имеет вход регулировки контрастности изображения V0. Питание подсветки (если таковая имеется) подается на выводы А и К. HD44780 взаимодействует с AVR через 8-битную двунаправленную шину команд/данных DB7:DB0. Временная диаграмма работы шины показана на рис. 3.3а. В момент записи информации в ЖКИ ведущий микроконтроллер выставляет на линиях DB7...DB0 8-разрядный код, после чего формирует на выводе Е стробирующий импульс (активный фронт – задний). По окончанию импульса должна быть выдержана пауза до начала новой транзакции. Признаком записи команды/ данных является состояние линии RS. При RS=0 происходит запись команды, при RS=1 – данных. Когда необходимо считать данные из индикатора, то выводы порта DB7:DB0 микроконтроллера настраиваются на ввод. Затем следует импульс подтверждения на линии Е и байт данных переписывается во внутренний регистр для дальнейшей обработки. Направление передачи данных определяет уровень на линии R/W (R/W =1 – чтение из индикатора, R/W =0 – запись в индикатор). В реальных приложениях, как правило, нет необходимости в чтение данных. Поэтому вывод R/W всегда соединяют с общим проводом.

Таблица 2. Функциональное назначение выводов символьного ЖКИ на базе HD44780:

Номер вывода	Название выводов	Функциональное назначение
1	GND	Общий вывод
2	VCC	Напряжение питания
3	V0	Напряжение управления контрастностью
4	RS	Выбор записи команды/данные
5	R/W	Выбор направления передачи данных запись/чтение
6	E	Вход тактовых импульсов
7-14	BD7-DB0	Шина данных
15	A	Анод светодиодной подсветки
16	K	Катод светодиодной подсветки

Одна из типовых схем подключения микроконтроллера типа AVR к ЖК индикатору типа A162-D приведена на рис.3.4а.

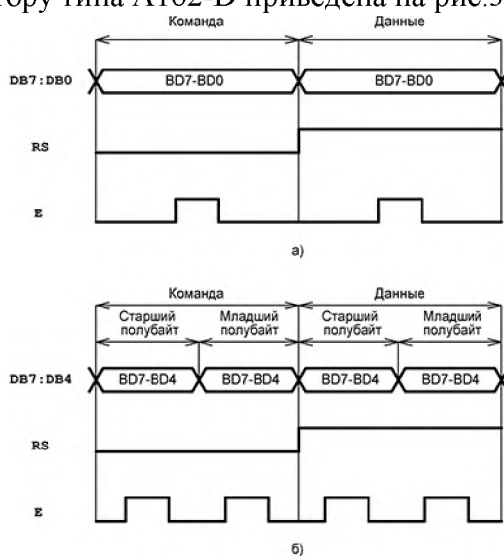


Рис. 3.3. Последовательность передачи данных в HD44780: а - по 8-разрядной шине команд/данных; б - по 4- разрядной шине команд/данных.

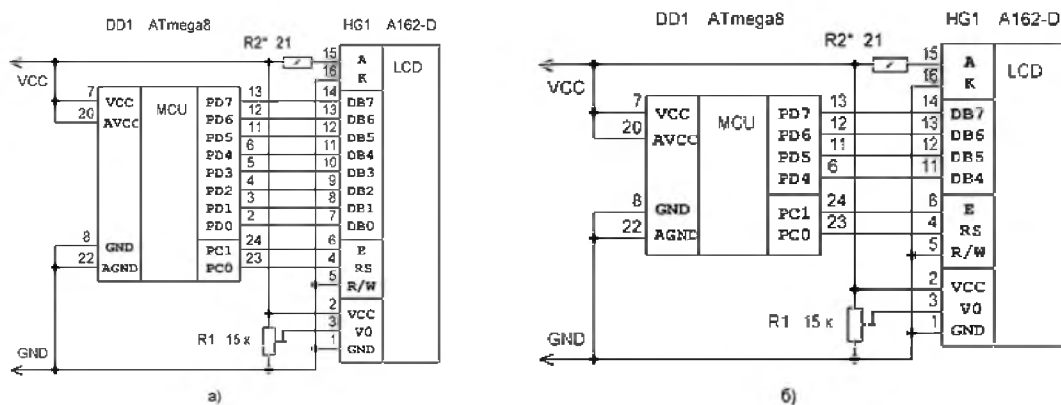


Рис. 3.4. Схема подключения символического ЖКИ к микроконтроллеру: а – при использовании 8-разрядной шины команд/данных; б - при использовании 4- разрядной шины команд/данных.

Для управления ЖКИ может быть использован также 4-проводный интерфейс (см. схему подключения на рис. 3.4б), что позволяет сэкономить 4 линии ввода-вывода, при незначительном усложнении программы.

В этом случае 4-разрядную шину команд/данных формируют линии DB7... DB4 (линии DB3...DB0 остаются незадействованными). Скорость записи снижается в 2 раза, но это, обычно, не вызывает ни каких проблем во время работы.

Последовательность передачи данных показана на рис. 3.3б. Команды/ данные передается за два такта. Первым следует старший полубайт, вторым – младший.

Каждая тетрада, естественно, должна быть зафиксирована импульсом на линии E.

Контроллер HD44780 имеет буфер видеопамати DDRAM (Display Data RAM), из которой символы переносятся на дисплей. Объем DDRAM зависит от числа строк и позиций на экране. Адреса ячеек видеопамати первой строки 0x80...0xA8, второй 0xC0...0xE8. В текущий момент времени в окно дисплея попадают только 16 символов из DDRAM рис. 2.2 (положение окна можно изменять программно).__

Оборудование и материалы.

Комплекс лабораторных работ по исследованию работы ЖК символьных индикаторов с микроконтроллером типа AVR выполняется на учебном макетном лабораторном комплексе РТМТЛ-3.

На передней крышке прибора закреплена отладочная плата, содержащая исследуемый микропроцессор (Atmega8535, Atmega16 или совместимый); резистивно-диодный последовательный (работающий по интерфейсу RS232 через СОМ-порт ПК) программатор; обвязку микроконтроллера согласно паспортным данным; набор из 11 светодиодов, подключенных к выводам 14 – 21, 33 – 35 микроконтроллера см. рис. 4.1; набор из 11 кнопок с фиксацией нормально разомкнутых (кнопка нажата — замкнуто; отжата — разомкнуто); 1 кнопку без фиксации нормально разомкнутую; 1 потенциометр (переменный резистор КОНТРАСТНОСТЬ), подключенный подключенный средней точкой к выводу VEE исследуемого LCD индикатора, для измерения напряжения контрастности выведены клеммы типа «РСА тюльпан» для подключения мультиметра; 4 потенциометра, подключенные к выводам 40, 39, 38, 37; а также кнопку с фиксацией «RESET», при нажатии которой вывод RESET кристалла соединяется с корпусом схемы. Для программирования прибора используется разъем «ПРОГРАММАТОР». При этом «ПРОГРАММАТОР» следует подключать к СОМ –порту ПЭВМ **ТОЛЬКО** проводом типа СОМ 9m/9f («мама» - «папа»).

Принципиальная электрическая схема прибора приведена на рис. 4.1. Демонстрационные примеры готовых программ, на основе которых могут быть построены задания по программированию микроконтроллера, приведены в папке Examples_RTMTL-3.

Задание 1. Реализация 8 битного режима работы LCD ЖК символьного индикатора. При этом с помощью кнопок S1 – S11 можно моделировать неисправности ЖК индикатора при обрыве того или иного провода, соединяющего LCD дисплей с микроконтроллером.

Задание 2. Реализация 4 битного режима работы LCD ЖК символьного индикатора.

Задание 3. Реализация 8 битного режима работы LCD индикатора и работа со светодиодами. При этом 8 светодиодов повторяют логическое состояние порта данных дисплея, а три светодиода — состояние выводов RS, RW и E. Можно наглядно наблюдать команды управления и данные, приходящие на ЖК индикатор.

Задание 4. Работа с потенциометрами, подключенными к портам PA0-PA3 микроконтроллера. Каждый из потенциометров подключен средней точкой к соответствующему выводу микроконтроллера, который является входом АЦП. Измерения выводятся на ЖК индикатор.

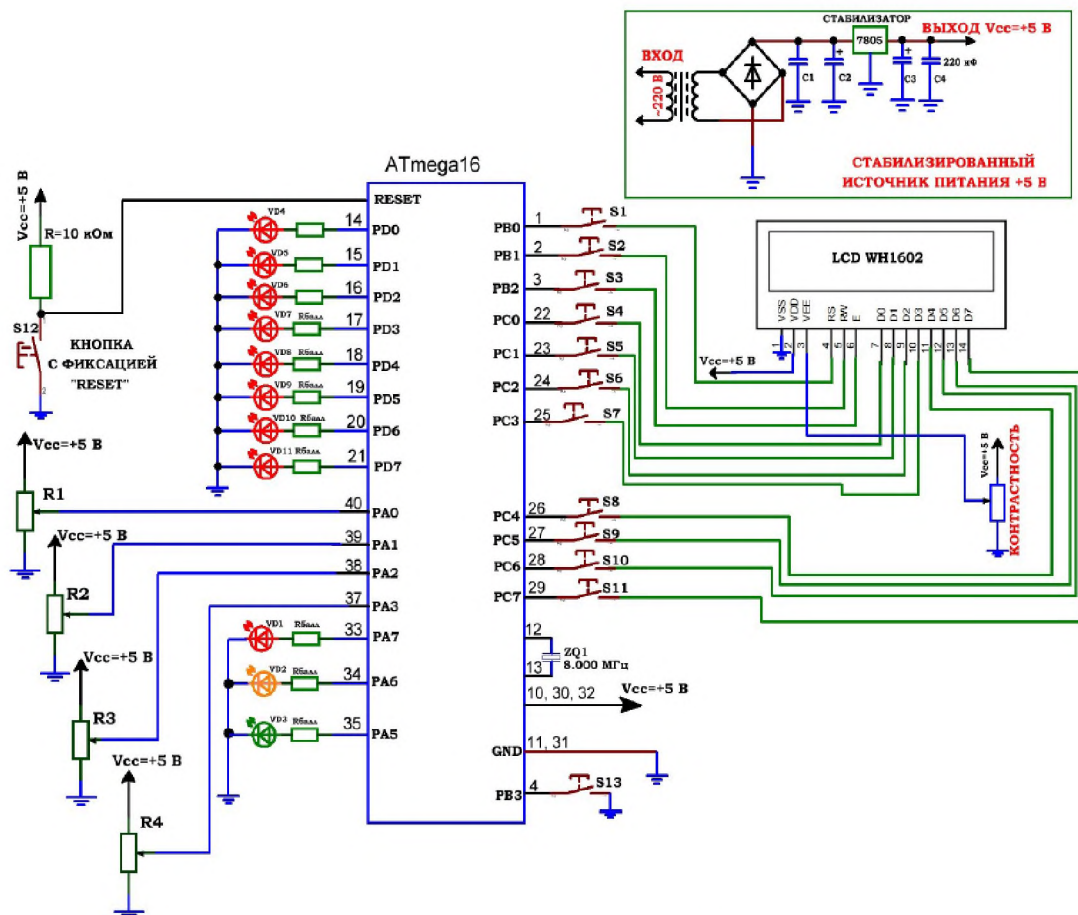


Рис. 4.1. Принципиальная электрическая схема учебного стенда для изучения работы LCD ЖК символьного индикатора с микроконтроллером AVR.

Задание 5. Работа с потенциометрами и кнопкой S13, подключенной к порту PB3. Кнопка настроена на переключение каналов АЦП. Измерения выводятся на ЖК индикатор.

Задание 6. Переменный резистор R1, подключен средней точкой ко входу 40 АЦП микроконтроллера. Измеренное значение напряжения в кодах АЦП выводится на ЖК индикатор с одновременной визуализацией условного уровня с помощью 8 светодиодов VD4 – VD11. Количество зажженных светодиодов пропорционально коду АЦП. Светодиоды VD1 – VD3 при этом мигают_

Указания по технике безопасности

Соответствуют технике безопасности по работе с компьютерной техникой.

Ход работы

1. Перед включением установки в сеть проверить целостность всех соединительных сигнальных и сетевых проводов. Все работы по подключению комплекса к компьютеру следует выполнять только при отключенных от сети приборах. Разобраться с принципиальными блок-схемами опытов, в назначении кнопок, переключателей и ручек прибора.

2. Перед выполнением работы следует изучить инструкцию по работе с учебной средой программирования Algorithm Builder, а также ознакомиться с полным методическим руководством по микроконтроллерам семейства AVR. Ознакомиться с кратким методическим руководством по программированию микроконтроллеров, приведенных в руководстве для учебного стенда РТМТЛ-2. Ознакомиться со статьей «Практическое объяснение работы ЖКИ модулей с HD44780» (файл Kea_45_1.html). Ознакомиться с паспортными данными на исследуемый ЖК индикатор (файл lcd_hd44780.pdf).

3. Соединить монитор с системным блоком ПЭВМ, подключить клавиатуру и мышь к системному блоку используя стандартные провода для подключения. Подключить системный блок ПЭВМ и монитор к сети ~220 В.

4. Загрузить операционную систему согласно стандартным процедурам загрузки.

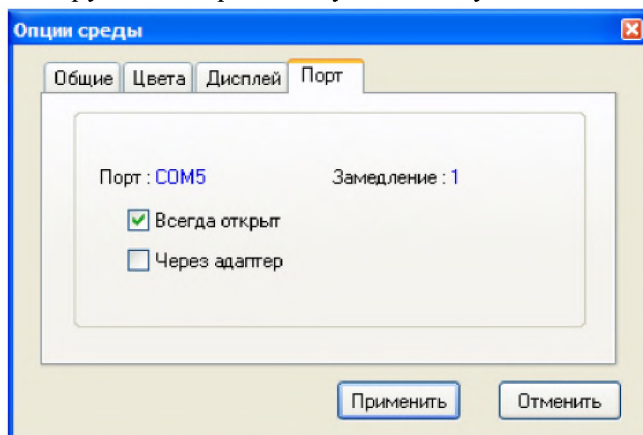


Рис. 5.1

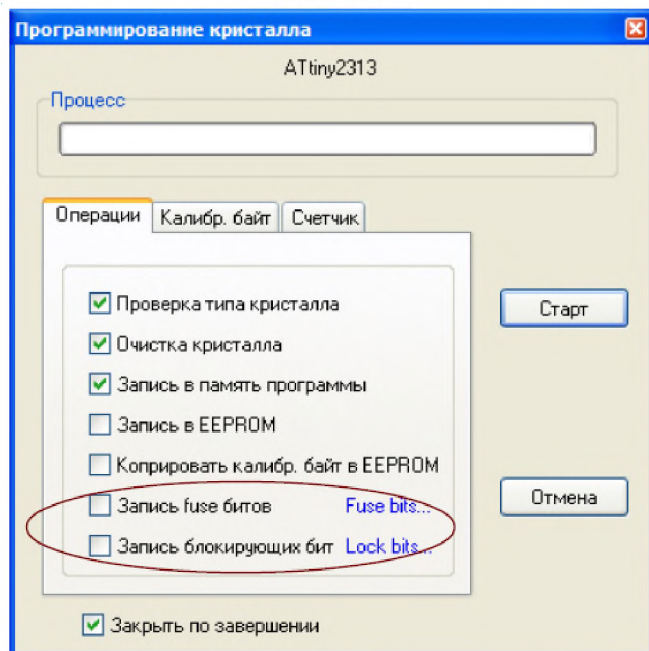


Рис. 5.2

5. При необходимости, настроить компьютер для работы с учебной установкой и программной средой Algorithm Builder.

6. Запустить программу Algorithm Builder для работы с учебной установкой для данного эксперимента пользуясь ярлыком на рабочем столе либо другим способом, указанным лаборантом. Для запуска можно воспользоваться комплексной программной оболочкой LabVisual для РТМТЛ-1-5.

7. Подключить разъем «ПРОГРАММАТОР» учебного прибора к COM – порту ПЭВМ проводом типа COM 9m/9f («мама» - «папа»)

8. Настроить программу Algorithm Builder для работы с данным COM портом ОПЦИИ-ОПЦИИ СРЕДЫ-ПОРТ рис. 5.1.

9. Для подробных инструкций следует обращаться к методическому руководству по среде Algorithm Builder.

10. Загрузить Пример 1 (папка 1) в среду Algorithm Builder (ФАЙЛ-ОТКРЫТЬ-ВЫБРАТЬ ФАЙЛ *.alp).

11. «Прошить» тестовую программу в кристалл. Для этого нажать кнопку «ЗАПУСК С КРИСТАЛЛОМ», при этом откроется диалог программирования кристалла

рис. 5.2. **Перед прошивкой следует нажать кнопку с фиксацией «RESET» на учебном приборе.**

12. Перед нажатием кнопки «СТАРТ» следует внимательно проверить положения флажков установки программатора рис. 5.2. Особенно внимательно следует отнестись к опциям «ЗАПИСЬ Fuse bit» и «ЗАПИСЬ блокирующих бит». **ФЛАЖКИ ЭТИХ ОПЦИЙ ОБЯЗАТЕЛЬНО ДОЛЖНЫ БЫТЬ СНЯТЫ, Т. К. НЕ ПРАВИЛЬНО ПРОШИТЫЕ FUSE БИТЫ МОГУТ ПРИВЕСТИ К ДАЛЬНЕЙШЕЙ НЕВОЗМОЖНОСТИ РАБОТЫ С ДАННЫМ МИКРОКОНТРОЛЛЕРОМ!**

13. **Отжать кнопку RESET и проверить работу тестовой программы.**

14. Загрузить примеры 2 — 6 из соответствующих папок и прошить в кристалл каждую из тестовых программ.

15. На основании данных примеров составить собственные программы, реализующие различные задачи (изменить вывод на ЖК индикатор, мигание светодиодов, опрос кнопок и потенциометров). Составить программы для реализации задач как для 8-битного, так и для 4-битного режима работы ЖК индикатора.

16. Отключая соответствующие выходы ЖК индикатора от микроконтроллера с помощью кнопок управления, моделировать неисправности, которые могут возникнуть при плохом соединении или обрыве связи LCD – микроконтроллер.

17. Измерить с помощью мультиметра напряжение, подаваемое на вход VEE (контрастность) дисплея при наилучшем визуальном отображении символов.

18. Для составления программ и подробного изучения работы микроконтроллеров AVR следует обращаться к учебной литературе, полному руководству к микроконтроллерам семейства AVR, а также к документации по среде Algorithm Builder и документацией на ЖК индикатор.

19. По окончании работы следует закрыть программу и все открытые подпрограммы, закрыть виртуальную среду VirtualBox (при работе в среде Linux).

20. Выключить компьютер, нажав на кнопку, находящуюся в крайнем нижнем левом углу экрана. Из доступных действий выбрать «ВЫХОД»--> «ВЫКЛЮЧИТЬ КОМПЬЮТЕР».

21. Отключить установку от сети, поставив переключатели «СЕТЬ» на панели установки в положение «ВЫКЛ» и вынуть сетевые вилки из розеток

Оформление отчета

В отчете должны содержаться описание команд использованных в выполнении заданий и программы написанные по заданным в задании алгоритмам

Контрольные вопросы

1. Какие способы взаимодействия имеет микроконтроллера AVR с ЖКИ модулем на базе контроллера HD44780
2. Сколько строк имеет ЖКИ модуль на базе контроллера HD44780?
3. Сколько символов в строке воспроизводит ЖКИ модуль на базе контроллера HD44780?
4. Какие регистры имеет ЖКИ модуль на базе контроллера HD44780 для управления ?
5. Какое количество символов и каким образом воспроизводит ЖКИ модуль на базе контроллера HD44780?
6. Каким сигналом синхронизируется запись кода символа в ЖКИ модуль на базе контроллера HD44780?
7. Каким сигналом синхронизируется запись кода команды в ЖКИ модуль на базе контроллера HD44780?

Список литературы, рекомендуемый к использованию по данной теме:

1. Клингман Э. Проектирование микропроцессорных систем. М.: Мир. 1988. - 575с.
2. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристалльных микроконтроллерах.
3. Трамперт В. AVR-RISK микроконтроллеры.: Пер. с нем. – К.: «МК-ПРЕСС», 2006.-464с., ил.
4. Ю.А.Шпак. Программирование на языке С для AVR и PIC микроконтроллеров. – К.: «МК-ПРЕСС», 2006.-400с., ил.
5. Якубовский С.В. Цифровые и аналоговые интегральные микросхемы. М.:Радио и связь. 1990.
6. Токхейм Р. Основы цифровой электроники. Пер. с англ. - М.: Мир, 1988. – 390 с.
7. Шило В.Л. Популярныe цифровые микросхемы. – М.: Радио и связь, 1990.– 350 с.
8. Бирюков С.А. Цифровые устройства на МОП-интегральных микросхемах - М.: Радио и связь. 1992 (1996).
9. Опадчий Ю.Ф., Глудкин О.П., Гуров А.И. Аналоговая и цифровая электроника. Полный курс: учебник для вузов. - М.: Горячая линия, 1999 (2000, 2005). – 768 с.
10. Алексенко А.В., Шагуров И.И. Микросхемотехника.– М.: Радио и связь,1990 (1982).

Лабораторная работа 9

Разработка программ для связи микроконтроллера AVR как регулятора

Цель работы

Изучение инструментальных средств разработки программного обеспечения МК.
Разработка программ вывода информации на ЖКИ микроконтроллера AVR и программ для связи микроконтроллера AVR с ПК-ЭВМ.

Компетенции:

Индекс	Формулировка:
ОПК-3	способностью решать задачи анализа и расчета характеристик электрических цепей
ОПК-7	способностью учитывать современные тенденции развития теории автоматического управления, электронного оборудования, измерительной и вычислительной техники, информационных технологий в профессиональной деятельности
ПК-1	способностью выполнять эксперименты на действующих объектах по заданным методикам, оценивать динамические характеристики рассматриваемых объектов и идентифицировать передаточные функции объектов с применением современных информационных технологий и технических средств
ПК-7	готовностью участвовать в разработке и изготовлении стендов для комплексной отладки и испытаний программно-аппаратных управляющих комплексов
ПК-8	готовностью к внедрению результатов разработок средств и систем автоматизации и управления в производстве с использованием современных программируемых логических контролеров (ПЛК)
ПК-11	готовностью производить инсталляцию и настройку системного, прикладного и инструментального программного обеспечения систем автоматизации и управления
ПК-14	способностью разрабатывать электромеханические системы и использовать современную элементную базу при проектировании средств и систем управления

Теоретическая часть

Микроконтроллеры семейства в зависимости от модели имеют в своем составе от двух до шести таймеров/счетчиков общего назначения (Табл. 7.1).

Таблица 7.1. Таймеры/счетчики общего назначения

Таймер/счетчик	ATmega8515x	ATmega8535x	ATmega8x, ATmega16x/32x	ATmega64x/128x	ATmega48x/88x/168x	ATmega162x	ATmega164x/324x/644x	ATmega165x	ATmega325x/3250x, ATmega645x/6450x	ATmega640x, ATmega1280x/1281x, ATmega2560x/2561x
Таймер/счетчик T0 (8-битный)	•	•	•	• ¹⁾	•	•	•	•	•	•
Таймер/счетчик T1 (16-битный)	•	•	•	•	•	•	•	•	•	•
Таймер/счетчик T2 (8-битный)		• ¹⁾	• ¹⁾	•	• ¹⁾	• ¹⁾	• ¹⁾	• ¹⁾	• ¹⁾	• ¹⁾
Таймер/счетчик T3 (16-битный)				•		•				•
Таймер/счетчик T4 (16-битный)										•
Таймер/счетчик T5 (16-битный)										•
¹⁾ Асинхронный таймер/счетчик.										

Как видно из таблицы, во всех моделях микроконтроллеров семейства присутствуют как минимум два таймера/счетчика — T0 и T1. Таймер/счетчик T0 имеет минимальный набор функций, зависящий, тем не менее, от модели микроконтроллера. В одних моделях он может использоваться только для отсчета и измерения временных интервалов или как счетчик внешних событий. В других моделях к этим функциям добавляется возможность генерации сигналов с широтно-импульсной модуляцией (ШИМ) фиксированной разрядности (один или два канала), а также возможность работать в асинхронном режиме в качестве часов реального времени (в моделях ATmega64x/128x).

Таймер/счетчик T1 тоже может использоваться для отсчета временных интервалов и как счетчик внешних событий. Кроме того, он может осуществлять запоминание своего состояния по внешнему сигналу. Как и таймер/счетчик T0, он может работать в качестве 2- или 3-канального широтно-импульсного модулятора, но уже переменной разрядности. Количество каналов ШИМ зависит от модели.

Таймер/счетчик T2 практически полностью аналогичен таймеру/счетчику T0. Во всех моделях, кроме ATmega64x/128x, таймер/счетчик T2 может работать в асинхронном режиме.

Таймеры/счетчики T3...T5 по функциональным возможностям идентичны таймеру/счетчику T1.

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

7.2. Назначение выводов таймеров/счетчиков

Каждый таймер/счетчик использует один или более выводов микроконтроллера. Как правило, эти выводы — линии портов ввода/вывода общего назначения, а функции, реализуемые этими выводами при работе совместно с таймерами/счетчиками, являются их альтернативными функциями.

Все выводы микроконтроллеров, используемые таймерами/счетчиками общего назначения, приведены в Табл. 7.2. Там же указаны функции этих выводов.

Таблица 7.2. Выводы, используемые таймерами/счетчиками общего назначения

Название	ATmega8515x	ATmega8535x	ATmega8x	ATmega16x/32x	ATmega64x/128x	ATmega48x/88x/168x	ATmega162x	ATmega164x/324x/644x	ATmega165x/325x/645x	ATmega3250x/6450x	ATmega1281x/2561x	ATmega640x/1280x/2560x	Описание
T0	PB0	PB0	PD4	PB0	—	PD4	PB0	PB0	PG4	PG4	PD7	PD7	Вход внешнего сигнала таймера T0
OC0	PB0	PB3	—	PB3	PB4	—	PB0	—	—	—	—	—	Выход схемы сравнения таймера T0
OC0A	—	—	—	—	—	PD6	—	PB3	PB4	PB4	PB7	PB7	
OC0B	—	—	—	—	—	PD5	—	PB4	—	—	PG5	PG5	
T1	PB1	PB1	PD5	PB1	PD6	PD5	PB1	PB1	PG3	PG3	PD6	PD6	Вход внешнего сигнала таймера T1
ICP	PE0	—	—	—	—	—	—	—	—	—	—	—	Вход захвата таймера T1
ICP1	—	PD6	PB0	PD6	PD4	PB0	PE0	PD6	PD0	PD0	PD4	PD4	
OC1A	PD5	PD5	PB1	PD5	PB5	PB1	PD5	PD5	PB5	PB5	PB5	PB5	Выход схемы сравнения таймера T1
OC1B	PE2	PD4	PB2	PD4	PB6	PB2	PE2	PD4	PB6	PB6	PB6	PB6	
OC1C	—	—	—	—	PB7	—	—	—	—	—	PB7	PB7	
T2	—	—	—	—	PD7	—	—	—	—	—	—	—	Вход внешнего сигнала таймера T2
OC2	—	—	PB3	PD7	PB7	—	PB1	—	—	—	—	—	Выход схемы сравнения таймера T2
OC2A	—	—	—	—	—	PB3	—	PD7	PB7	PB7	PB4	PB4	
OC2B	—	—	—	—	—	PD3	—	PD6	—	—	—	PH6	
T3	—	—	—	—	PE6	—	—	—	—	—	PE6	PE6	Вход внешнего сигнала таймера T3
ICP3	—	—	—	—	PE7	—	PD3	—	—	—	PE7	PE7	Вход захвата таймера T3

(продолжение)

Название													Описание
	АТmega8515х	АТmega8535х	АТmega8х	АТmega16х/32х	АТmega64х/128х	АТmega48х/88х/168х	АТmega162х	АТmega164х/324х/644х	АТmega165х/325х/645х	АТmega3250х/6450х	АТmega1281х/2561х	АТmega640х/1280х/2560х	
OC3A	—	—	—	—	PE3	—	PD4	—	—	—	PE3	PE3	Выход схемы сравнения таймера T3
OC3B	—	—	—	—	PE4	—	PB4	—	—	—	PE4	PE4	
OC3C	—	—	—	—	PE5	—	—	—	—	—	PE5	PE5	
T4	—	—	—	—	—	—	—	—	—	—	—	RH7	Вход внешнего сигнала таймера T4
ICP4	—	—	—	—	—	—	—	—	—	—	—	PL0	Вход захвата таймера T4
OC4A	—	—	—	—	—	—	—	—	—	—	—	RH3	Выход схемы сравнения таймера T4
OC4B	—	—	—	—	—	—	—	—	—	—	—	RH4	
OC4C	—	—	—	—	—	—	—	—	—	—	—	RH5	
T5	—	—	—	—	—	—	—	—	—	—	—	PL2	Вход внешнего сигнала таймера T5
ICP5	—	—	—	—	—	—	—	—	—	—	—	PL1	Вход захвата таймера T5
OC5A	—	—	—	—	—	—	—	—	—	—	—	PL3	Выход схемы сравнения таймера T5
OC5B	—	—	—	—	—	—	—	—	—	—	—	PL4	
OC5C	—	—	—	—	—	—	—	—	—	—	—	PL5	
TOSC1	—	PC6	PB6	PC6	PG4	PB6	PD4	PC6	— ¹⁾	— ¹⁾	PG4	PG4	Вход для подключения резонатора
TOSC2	—	PC7	PB7	PC7	PG3	PB7	PD5	PC7	— ¹⁾	— ¹⁾	PG3	PG3	Выход для подключения резонатора

¹⁾ Отдельные контакты ввода/вывода, совмещенные с выводами XTAL1/XTAL2.

Не забывайте о том, что при использовании альтернативных функций линий портов ввода/вывода необходимо, как правило, самостоятельно сконфигурировать эти выходы в соответствии с их функциональным назначением.

7.3. Прерывания от таймеров/счетчиков

В старых моделях для разрешения/запрещения прерываний от таймеров/счетчиков использовалось от одного до двух регистров ввода/вывода. В новых моделях число таких регистров С...6) равно числу счетчиков в конкретной модели. Точно так же дело обстоит и с регистрами, содержащими флаги прерываний. Названия и адреса всех этих регистров приведены в Табл. 7.3.

(продолжение)

Название	ATmega8515x												ATmega8535x	ATmega8x	ATmega16x/32x	ATmega64x/128x	ATmega48x/88x/168x	ATmega162x	ATmega164x/324x/644x	ATmega165x/325x/645x	ATmega3250x/6450x	ATmega1281x/2561x	ATmega640x/1280x/2560x	Описание
	OC3A	OC3B	OC3C	T4	ICP4	OC4A	OC4B	OC4C	T5	ICP5	OC5A	OC5B												
OC3A	—	—	—	—	PE3	—	PD4	—	—	—	PE3	PE3	Выход схемы сравнения таймера T3											
OC3B	—	—	—	—	PE4	—	PB4	—	—	—	PE4	PE4												
OC3C	—	—	—	—	PE5	—	—	—	—	—	PE5	PE5												
T4	—	—	—	—	—	—	—	—	—	—	—	—	PH7	Вход внешнего сигнала таймера T4										
ICP4	—	—	—	—	—	—	—	—	—	—	—	—	PL0	Вход захвата таймера T4										
OC4A	—	—	—	—	—	—	—	—	—	—	—	—	PH3	Выход схемы сравнения таймера T4										
OC4B	—	—	—	—	—	—	—	—	—	—	—	—	PH4											
OC4C	—	—	—	—	—	—	—	—	—	—	—	—	PH5											
T5	—	—	—	—	—	—	—	—	—	—	—	—	PL2	Вход внешнего сигнала таймера T5										
ICP5	—	—	—	—	—	—	—	—	—	—	—	—	PL1	Вход захвата таймера T5										
OC5A	—	—	—	—	—	—	—	—	—	—	—	—	PL3	Выход схемы сравнения таймера T5										
OC5B	—	—	—	—	—	—	—	—	—	—	—	—	PL4											
OC5C	—	—	—	—	—	—	—	—	—	—	—	—	PL5											
TOSC1	—	PC6	PB6	PC6	PG4	PB6	PD4	PC6	— ¹⁾	— ¹⁾	PG4	PG4	Вход для подключения резонатора											
TOSC2	—	PC7	PB7	PC7	PG3	PB7	PD5	PC7	— ¹⁾	— ¹⁾	PG3	PG3	Выход для подключения резонатора											

¹⁾ Отдельные контакты ввода/вывода, совмещенные с выводами XTAL1/XTAL2.

Форматы регистров, используемых для разрешения/запрещения прерываний от таймеров/счетчиков, показаны на Рис. 7.1...7.2, а описание их битов приведено в Табл. 7.4.

Для разрешения какого-либо прерывания от таймера/счетчика необходимо установить в 1 соответствующий бит регистра TIMSK (TIMSK1)/ETIMSK и, разумеется, флаг I регистра SREG.

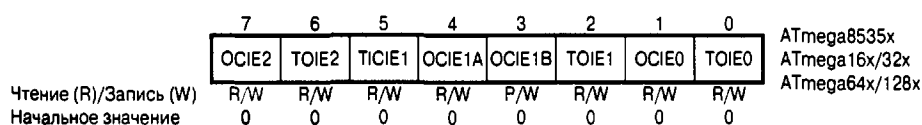


Рис. 7.1. Формат регистров TIMSK (а) и ETIMSK (б)

Таблица 7.4. Биты регистров TIMSK, ETIMSK и TIMSK0...TIMSK5

Название бита	Описание
TOIE _n	Флаг разрешения прерывания по переполнению таймера/счетчика T _n (n = 0...5)
OCIE _n	Флаг разрешения прерывания по событию «Совпадение» таймера/счетчика T _n (n = 0, 2)
OCIE _{nA}	Флаг разрешения прерывания по событию «Совпадение А» таймера/счетчика T _n (n = 0...5)
OCIE _{nB}	Флаг разрешения прерывания по событию «Совпадение В» таймера/счетчика T _n (n = 0...5)
OCIE _{nC}	Флаг разрешения прерывания по событию «Совпадение С» таймера/счетчика T _n (n = 1, 3...5)
TCIE _n	Флаг разрешения прерывания по событию «Захват» таймера/счетчика T _n (n = 1, 3)
ICIE _n	Флаг разрешения прерывания по событию «Захват» таймера/счетчика T _n (n = 1, 3...5)

Форматы регистров, используемых для индикации наступления прерываний от таймеров/счетчиков, показаны на Рис. 7.3 и Рис. 7.4, а описание их битов приведено в Табл. 7.5.

Таблица 7.5. Биты регистра TIFR

Название бита	Описание
TOV _n	Флаг прерывания по переполнению таймера /счетчика T _n (n = 0...5)
OCF _n	Флаг прерывания по событию «Совпадение» таймера/счетчика T _n (n = 0, 2)
OCF _{nA}	Флаг прерывания по событию «Совпадение А» таймера/счетчика T _n (n = 0...5)
OCF _{nB}	Флаг прерывания по событию «Совпадение В» таймера/счетчика T _n (n = 0...5)
OCF _{nC}	Флаг прерывания по событию «Совпадение С» таймера/счетчика T _n (n = 1, 3...5)
ICF _n	Флаг прерывания по событию «Захват» таймера/счетчика T _n (n = 1, 3...5)

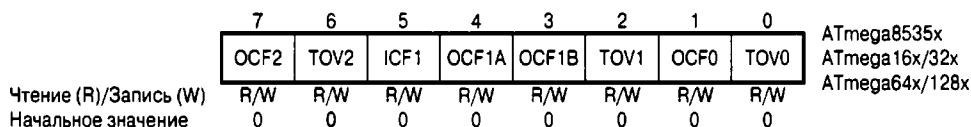


Рис. 7.3. Формат регистров TIFR (а) и ETIFR (б)

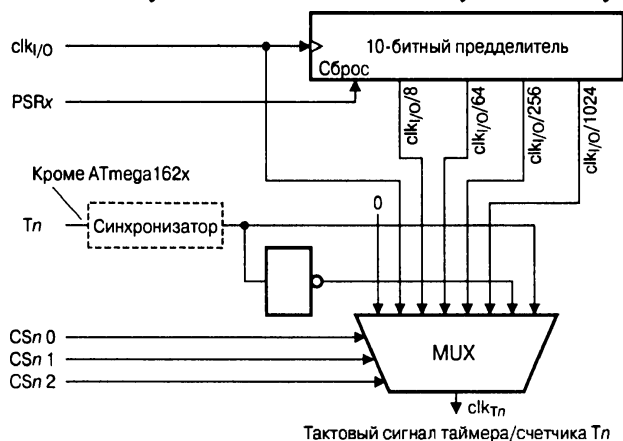
При наступлении какого-либо события соответствующий флаг регистра TIFR (TIFR_{rt})/ETIFR устанавливается в 1. При запуске подпрограммы обработки прерывания он аппаратно сбрасывается в 0. Любой флаг может быть также сброшен программно, записью в него лог. 1.

7.4. Пределители таймеров/счетчиков

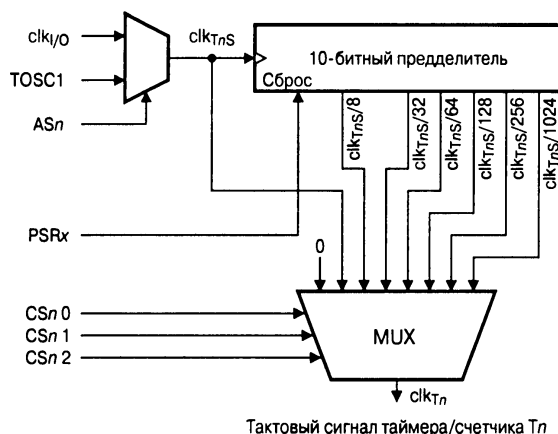
Блоки пределителей предназначены для формирования тактовых сигналов таймеров/счетчиков clkT₀, clkT₁, clkT₂, clkT₃. Упрощенная структурная схема блока пределителя таймеров/счетчиков, не имеющих асинхронного режима работы, приведена на Рис. 7.5, а. Структурная схема блока пределителя таймеров/счетчиков, имеющих возможность работы в асинхронном режиме, приведена на Рис. 7.5, б.

Как показано на рисунке, в состав каждого блока входят собственно 10-битный пределитель, выходной мультиплексор (селектор тактового сигнала), а для таймеров, имеющих возможность работы в асинхронном режиме, — еще и входной мультиплексор исходного тактового сигнала. По последней схеме выполнен пределитель таймера/счетчика T₀ моделей ATmega64x/128x и таймера/счетчика T₂ остальных моделей. Следует иметь в виду, что все таймеры/счетчики каждой модели семейства, не имеющие асинхронного режима работы, используют один и тот же 10-

битный предделитель. При этом управление тактовым сигналом каждого таймера/счетчика осуществляется индивидуально и будет описано при их рассмотрении.



- $n = 0, 1, 3$ — для ATmega162x (PSRx = PSR310),
 $1, 2, 3$ — для ATmega64x и ATmega128x (PSRx = PSR321),
 $0, 1$ — для ATmega8515x/8535x, ATmega8x/16x/32x
 и ATmega165x/325x/3250x/645x/6450x (PSRx = PSR10),
 $0, 1$ — для ATmega48x/88x/168x и ATmega164x/324x/644x (PSRx = PSRSYNC),
 $0, 1, 3, 4, 5$ — для ATmega640x/1280x/1281x/2560x/2561x (PSRx = PSRSYNC)



- $n = 1$ — для ATmega64x и ATmega128x (PSRx = PSR0);
 2 — для ATmega8535x, ATmega8x/16x/32x, ATmega162x
 и ATmega165x/325x/3250x/645x/6450x (PSRx = PSR2);
 2 — для ATmega48x/88x/168x, ATmega164x/324x/644x
 и ATmega640x/1280x/1281x/2560x/2561x (PSRx = PSRASY).

б)

Рис. 7.5. Блок предделителя таймеров/счетчиков:

a — без асинхронного режима; *б* — с асинхронным режимом

Следует понимать, что предделители работают независимо от таймеров/счетчиков. Следствием этого является, в частности, неопределенный промежуток времени $A \dots 7N+1$ тактов исходного сигнала, где N — коэффициент деления предделителя) между разрешением таймера/счетчика и первым его отсчетом при работе совместно с предделителем. Чтобы уйти от этой неопределенности, можно воспользоваться средствами, описанными в следующем подразделе.

7.4.1. Управление предделителями

Помимо управления тактовым сигналом таймера/счетчика, все микроконтроллеры семейства позволяют осуществлять сброс предделителей, а отдельные модели позволяют также осуществлять их остановку. Для этого используется либо регистр специальных функций SFIOR, либо (в новых моделях) регистр управления таймеров/счетчиков GTCCR, расположенный по адресу \$23 (\$43). Формат этого регистра для различных моделей микроконтроллеров приведен на Рис. 7.6 (биты, не используемые для управления предделителями таймеров/счетчиков, указаны на рисунке как X).

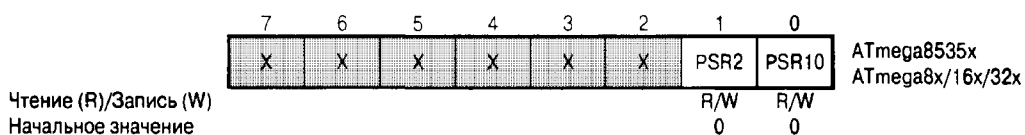


Рис. 7.6. Управление пределителями таймеров/счетчиков — регистры SFIOR (а) и GTCCR (б)

Для сброса пределителей таймеров/счетчиков используются биты PSRjс (PSRSYNC/PSRASY) регистра. При записи в эти биты лог. 1 пределители соответствующих таймеров/счетчиков переводятся в исходное состояние. Биты сбрасываются в 0 аппаратно после выполнения операции сброса. Напоминаю, что один пределитель, как правило, используется несколькими таймерами/счетчиками, и соответственно сброс пределителя повлияет на все таймеры/счетчики, которые его используют.

Остановка всех пределителей микроконтроллера осуществляется записью лог. 1 в бит TSM регистра SFIOR или GTCCR. Последующий запуск пределителей осуществляется записью в бит TSM лог. 0. Указанная функция может использоваться, в частности, для синхронизации таймеров/счетчиков. После установки бита TSM и битов PSRjс (PSRSYNC/PSRASY) соответствующие таймеры/счетчики останавливаются и могут быть проинициализированы требуемыми значениями. После сброса бита TSM биты PSRх (PSRSYNC/PSRASY) аппаратно сбрасываются и все таймеры/счетчики начинают работать одновременно.

Оборудование и материалы.

Комплекс лабораторных работ по исследованию микроконтроллеров AVR выполняется на учебном макетном лабораторном комплексе РТМТЛ-1.

На передней крышке прибора закреплена отладочная плата, содержащая исследуемый микропроцессор (Atmega8535); резистивно-диодный последовательный (работающий по интерфейсу RS232 через COM-порт ПК) программатор; обвязку микроконтроллера согласно паспортным данным; 1 светодиод, подключенный к выводу 20 рис. 3.1; 1 кнопка с фиксацией нормально разомкнутая (кнопка нажата — замкнута; отжата — разомкнута), подключенная к выводу 39; 1 потенциометр (переменный резистор), подключенный средней точкой к выводу 40 микроконтроллера; выходы «НАГРУЗКА» XS2 — XS3; вход XS1 (для примера работы частотомера); подстроечный резистор Rк — контрастность, подключенный средней точкой к выводу VEE LCD индикатора; а также кнопку с фиксацией «RESET», при нажатии которой вывод RESET кристалла соединяется с корпусом схемы. Для программирования прибора используется разъем «ПРОГРАММАТОР». При этом «ПРОГРАММАТОР» следует подключать к COM – порту ПЭВМ **ТОЛЬКО** проводом типа COM 9m/9f («мама» - «папа»).

Принципиальная электрическая схема прибора приведена на рис. 3.1.

Демонстрационные примеры готовых программ, на основе которых могут быть построены задания по программированию микроконтроллера, приведены в папке Examples_RTMTL-1.

Пример 1. Работа со светодиодом, таймером, ЖК индикатором и входом PB1 микроконтроллера. Построение программы-частотомера. В данном примере на вывод 2 PB1 микроконтроллера (вход XS1) подаётся сигнал с внешнего генератора (в качестве генератора можно использовать программу «LabVisual ЗВУКОВОЙ ГЕНЕРАТОР ПК»). Сигнал предварительно усилен однотранзисторным резистивным усилителем, собранном на транзисторе VT1 BC547 и резисторах R2-R5. Микроконтроллер программируется таким образом, чтобы его TIMER1 был настроен на режим синхронизации внешним сигналом, поступающим на порт PB1. Программа 1 раз в секунду выполняет подсчет входных импульсов синхронизации, поступающих на вход PB1 (макс. Значение 65535).

Предусмотрен вывод показаний на LCD индикатор (максимальное значение 9999 Гц). Количество подсчитанных импульсов за 1 сек и есть частота входного сигнала в Гц.

Программа также одновременно работает с подключенным к микроконтроллеру ЖК LCD индикатором (инициализирует его, осуществляет вывод информации).

Одновременно с этим каждую секунду производится инвертирование состояние вывода PD6, к которому подключен светодиод VD1 (1 сек. Светодиод включен, 1 сек. - выключен). При отключенном от клеммы XS1 внешнем генераторе в нижней строке дисплея отображается INPUT DISCONNECT, при подключении внешнего генератора и частоте отличной от 0, в строке выводится сообщение INPUT CONNECT.

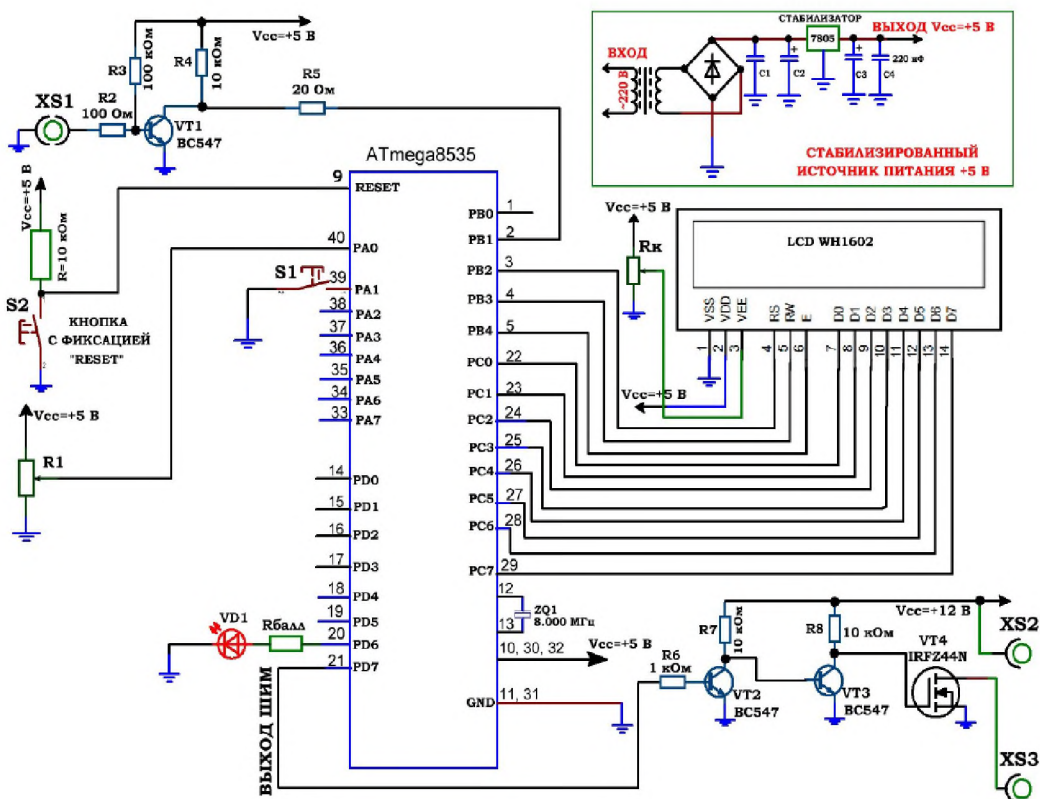


Рис. 3.1. Принципиальная электрическая схема учебного стенда для изучения микроконтроллеров серии Atmega8535.

Рис. 3.1. Принципиальная электрическая схема учебного стенда для изучения микроконтроллеров серии Atmega8535.

2) **Пример 2.** Работа переменного резистора, кнопки, ЖК индикатора и выхода ШИМ генератора. ШИМ генерируется на выводе PD7 микроконтроллера и подается на трёхкаскадный усилитель мощности, два каскада которого собраны на биполярных транзисторах с общим эмиттером $VT2=VT3=BC547$, а третий каскад — на полевом транзисторе IRFZ44N с общим истоком. В стоковую цепь полевого транзистора подключается нагрузка (клеммы XS2 – XS3) относительно +12 В питания (колебания напряжения питания может составлять +10 ... +12 В, т. к. данный источник не стабилизированный). В качестве нагрузки может быть использован электродвигатель вентилятора либо лампа накаливания 12 В, 5 Вт. **Обратить внимание, что вентилятор плюсовым красным выводом должен подключаться строго к клемме XS2 (соблюдать полярность).**

При запуске программы длительность импульсов плавно нарастает с определенной скоростью, в нижней строке ЖК индикатора при этом выводится $SPEED=X1$. Кнопка с фиксацией, подключенная к выводу PA1 в этом режиме должна быть отжата, т. к. при нажатии кнопки скорость нарастания длительности импульсов

увеличивается в 4 раза и в нижней строке ЖК индикатора отображается $SPEED=X4$. В верхней строке ЖК индикатора выводится величина $Ust=...$, которая ограничивает нижний порог ШИМ и задается потенциометром R1, средней точкой подключенному к выводу 40 (PA0) микроконтроллера от 0 до 127 уровней. Отметим, что генерируемый ШИМ имеет всего 255 градаций (уровней), и, например, при $Ust=127$ будет меняться от 127 до 255 единиц. При достижении максимального значения, длительность импульсов ШИМ плавно убавляется.

Для получения гармонического сигнала используется свободно распространяемый компонент LabVisual Генератор для ПК рис. 3.2, представляющий из себя низкочастотный комбинированный генератор функций. Выходные сигналы генератора могут поступать на два канала стерео выхода звуковой карты компьютера.

Генератор вырабатывает сигналы следующего вида:

- синусоидальный,
- прямоугольный (меандр),
- треугольный симметричный,
- пилообразный нарастающий,
- пилообразный спадающий.

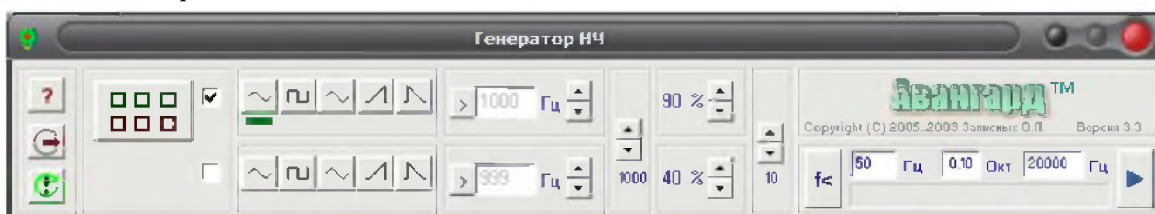



Рис. 3.2. Компонент «ГЕНЕРАТОР НЧ» для получения гармонических сигналов. Оптимальные настройки.

Компонент может работать в различных режимах. В постоянном независимом режиме форма, частота и амплитуда сигналов по двум каналам устанавливаются отдельно. В постоянном зависимом режиме форма, частота и амплитуда сигналов по обоим каналам одинаковые, сигнал второго (правого) канала может быть произвольно сдвинут по фазе относительно сигнала первого (левого) канала. В режиме нарастания частоты сигнал по первому (левому) каналу меняется по частоте в заданных пределах с установленной скоростью. Сигнал второго канала выключен. В режиме нарастания амплитуды сигнал по первому (левому) каналу меняется по амплитуде в заданных пределах с установленной скоростью. Сигнал второго канала выключен. [Экранные органы управления снабжены всплывающими комментариями.](#)

Частота сигналов может устанавливаться двумя способами. Установка **прямым**


вводом производится в окне вида . Для редактирования следует нажать кнопку слева от окна, ввести новое значение, затем нажать клавишу **Enter**. Второй способ - **ступенчатое изменение** частоты с заданным шагом. Для этого имеются переключатели "выше-ниже". Двумя изменяется частота, третьим переключается шаг этого изменения. Дискретность установки частоты 1 Гц.

Амплитуда задается ступенчатым изменением с заданным шагом. Для этого имеются переключатели "выше-ниже". Двумя изменяется частота, третьим переключается шаг этого изменения. Дискретность установки амплитуды 1 % от максимальной.

Сдвиг фаз между сигналами в **зависимом режиме** устанавливается прямым вводом значений в градусах. Для редактирования нажмите кнопку слева от окна, введите новое значение, затем нажмите клавишу **Enter**. Дискретность установки сдвига 1 градус.

Ввиду большого размера буферов обмена звуковой карты управление прибором несколько замедленно. Реакция на переключения составляет примерно 1 секунду.

В случае возникновения пауз выходных сигналов воспользуйтесь экранной

кнопкой перезапуска ().

Программа запоминает установки и настройки, и восстанавливает их при следующем включении. __

Указания по технике безопасности

Соответствуют технике безопасности по работе с компьютерной техникой.

Ход работы

1. Перед включением установки в сеть проверить целостность всех соединительных сигнальных и сетевых проводов. Все работы по подключению комплекса к компьютеру следует выполнять только при отключенных от сети приборах. Разобраться с принципиальными блок-схемами опытов, в назначении кнопок, переключателей и ручек прибора.

2. Перед выполнением работы следует изучить инструкцию по работе с учебной средой программирования Algorithm Builder, а также ознакомиться с полным методическим руководством по микроконтроллерам семейства AVR, изучить все доступные паспорта на исследуемые микроконтроллеры, а также на подключенный LCD индикатор.

3. Соединить монитор с системным блоком ПЭВМ, подключить клавиатуру и мышь к системному блоку используя стандартные провода для подключения. Подключить системный блок ПЭВМ и монитор к сети ~220 В.

4. Загрузить операционную систему согласно стандартным процедурам загрузки.

5. При необходимости, настроить компьютер для работы с учебной установкой и программной средой Algorithm Builder.

6. Запустить программу Algorithm Builder для работы с учебной установкой для данного эксперимента пользуясь ярлыком на рабочем столе либо другим способом, указанным лаборантом. Для работы можно воспользоваться комплексной программной оболочкой LabVisual для РТМТЛ-1-5-

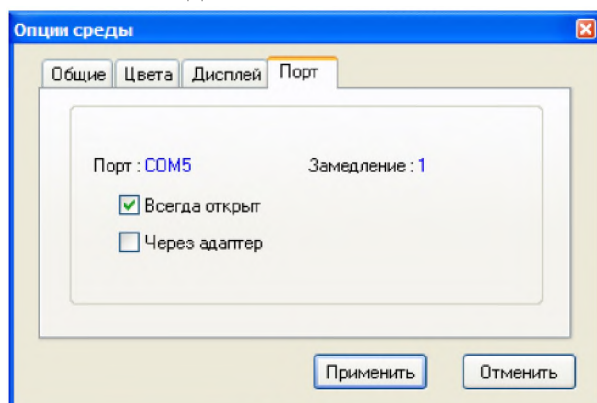


Рис. 4.1

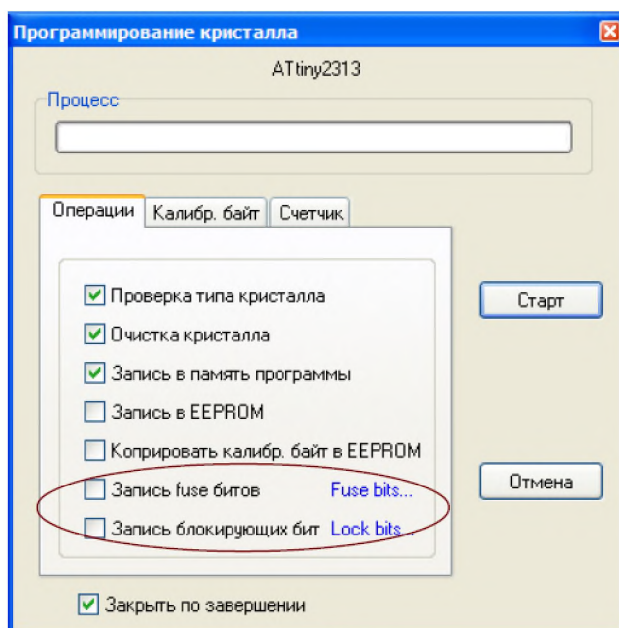


Рис. 4.2

6. Подключить разъём «ПРОГРАММАТОР» учебного прибора к COM – порту ПЭВМ проводом типа COM 9m/9f («мама» - «папа»)

7. Настроить программу Algorithm Builder для работы с данным COM портом ОПЦИИ-ОПЦИИ СРЕДЫ-ПОРТ рис. 4.1.

8. Для подробных инструкций следует обращаться к методическому руководству по среде Algorithm Builder.

9. Загрузить Пример 1 (папка 1) в среду Algorithm Builder (ФАЙЛ-ОТКРЫТЬ-ВЫБРАТЬ ФАЙЛ *.alp).

10. «Прошить» тестовую программу в кристалл. Для этого нажать кнопку «ЗАПУСК С КРИСТАЛЛОМ», при этом откроется диалог программирования кристалла рис. 4.2. **Перед прошивкой следует нажать кнопку с фиксацией $\#$ RESET $\#$ на учебном приборе.**

11. Перед нажатием кнопки «СТАРТ» следует внимательно проверить положения флажков установки программатора рис. 4.2. **Особенно внимательно следует отнестись к опциям $\#$ ЗАПИСЬ Fuse bit $\#$ и $\#$ ЗАПИСЬ блокирующих бит $\#$. ФЛАЖКИ ЭТИХ ОПЦИЙ ОБЯЗАТЕЛЬНО ДОЛЖНЫ БЫТЬ СНЯТЫ, Т. К. НЕ ПРАВИЛЬНО ПРОШИТЫЕ FUSE БИТЫ МОГУТ ПРИВЕСТИ К ДАЛЬНЕЙШЕЙ НЕВОЗМОЖНОСТИ РАБОТЫ С ДАННЫМ МИКРОКОНТРОЛЛЕРОМ!**

12. **Отжать кнопку RESET и проверить работу тестовой программы.**

13. Соединить выход LINE IN звуковой карты со входом XS1 стенда проводом «тюльпан — jack» из комплекта.

14. Проверить работу программа-частотомера, подавая на вход XS1 прямоугольный сигнал звуковой частоты 20 — 9000 Гц. При подачи сигнала с линейного выхода LINE OUT ПК посредством программы

«ГЕНЕРАТОР», следует установить амплитуду выходного сигнала генератора в программе не менее 80 % (рис. 3.2) и амплитуду выходного сигнала не менее 70 — 80 % непосредственно в звуковом драйвере (стандартная настройка громкости выходного сигнала). При низкой амплитуде входного сигнала, так же как и при слишком высокой амплитуде, частотомер может работать некорректно из за особенностей работы схемы однокаскадного транзисторного усилителя.

15. Загрузить пример 2 из соответствующей папки и прошить в кристалл тестовую программу.

16. Подключить к клеммам XS2 – XS3 нагрузку (вентилятор 12 В). **Обратить внимание, что вентилятор плюсовым красным выводом должен подключаться строго к клемме XS2 (соблюдать полярность).** Наблюдать работу тестовой программы.

17. Исходя из готовых примеров составить собственные программы в среде Algorithm Builder.

18. На основании Примера 1 можно составить программы, изменив, например, максимальное значение измеряемой частоты (в примере 9999 Гц), настройки TIMER1, действия, осуществляемые кнопкой S1 (включать светодиод), либо изменить вывод информации на ЖК индикатор. Также можно проводить различные операции с микроконтроллером.

19. На основании Примера 2 следует изучить работу ШИМ генератора микроконтроллера; изменить действия, осуществляемые кнопкой S1; изменить вывод информации на ЖК индикатор; поработать с потенциометром R1.

20. Для составления программ и подробного изучения работы микроконтроллеров AVR следует обращаться к учебной литературе, полному руководству к микроконтроллерам семейства AVR, а также к документации по среде Algorithm Builder.

21. По окончании работы следует закрыть программу и все открытые подпрограммы, закрыть виртуальную среду VirtualBox (при работе в среде Linux).

22. Выключить компьютер, нажав на кнопку, находящуюся в крайнем нижнем левом углу экрана. Из доступных действий выбрать «ВЫХОД»--> «ВЫКЛЮЧИТЬ КОМПЬЮТЕР».

23. Отключить установку от сети, поставив переключатели «СЕТЬ» на панели установки в положение «ВЫКЛ» и вынуть сетевые вилки из розеток. __

Оформление отчета

В отчете должны содержаться описание команд использованных в выполнении заданий и программы написанные по заданным в задании алгоритмам

Контрольные вопросы

1. Какие способы взаимодействия имеет микроконтроллера AVR с ЖКИ модулем на базе контроллера HD44780
2. Сколько строк имеет ЖКИ модуль на базе контроллера HD44780?
3. Сколько символов в строке воспроизводит ЖКИ модуль на базе контроллера HD44780?
4. Какие регистры имеет ЖКИ модуль на базе контроллера HD44780 для управления ?
5. Какое количество символов и каким образом воспроизводит ЖКИ модуль на базе контроллера HD44780?
6. Каким сигналом синхронизируется запись кода символа в ЖКИ модуль на базе контроллера HD44780?
7. Каким сигналом синхронизируется запись кода команды в ЖКИ модуль на базе контроллера HD44780?

Список литературы, рекомендуемый к использованию по данной теме:

1. Клингман Э. Проектирование микропроцессорных систем. М.: Мир. 1988. - 575с.
2. Сташин В.В., Урусов А.В., Мологонцева О.Ф. Проектирование цифровых устройств на однокристальных микроконтроллерах.
3. Трамперт В. AVR-RISK микроконтроллеры.: Пер. с нем. – К.: «МК-ПРЕСС», 2006.-464с., ил.
4. Ю.А.Шпак. Программирование на языке С для AVR и PIC микроконтроллеров. – К.: «МК-ПРЕСС», 2006.-400с., ил.

5. Якубовский С.В. Цифровые и аналоговые интегральные микросхемы. М.: Радио и связь. 1990.
6. Токхейм Р. Основы цифровой электроники. Пер. с англ. - М.: Мир, 1988. – 390 с.
7. Шило В.Л. Популярныe цифровые микросхемы. – М.: Радио и связь, 1990.– 350 с.
8. Бирюков С.А. Цифровые устройства на МОП-интегральных микросхемах - М.: Радио и связь. 1992 (1996).
9. Опадчий Ю.Ф., Глудкин О.П., Гуров А.И. Аналоговая и цифровая электроника. Полный курс: учебник для вузов. - М.: Горячая линия, 1999 (2000, 2005). – 768 с.
10. Алексенко А.В., Шагуров И.И. Микросхемотехника.– М.: Радио и связь, 1990 (1982).