Министерство образования Российской Федерации

Ульяновский государственный технический университет

ТЕХНИКА МИКРОПРОЦЕССОРНЫХ СИСТЕМ В КОММУТАЦИИ

Сборник лабораторных работ
Часть 1 «Проектирование микропроцессорных систем на базе Микроконтроллеров AVR фирмы Atmel»

Составители: В.Н. Горохин

Ульяновск 2015

УДК 621.391 (076)
ББК 32я7
Т 33

Рецензент канд. тех. наук, доцент кафедры САПР Кадеев Д.Н.

Одобрено секцией методических пособий научно–методического совета университета.

Т 33 Техника микропроцессорных систем в коммутации: Сборник лабораторных работ. Часть 1 «Проектирование микропроцессорных систем на базе микроконтроллеров AVR фирмы Atmel»/ сост. В.Н. Горохин. – Ульяновск : УлГТУ, 2015. – 100 с.

Сборник лабораторных работ разработан в соответствии с программой курса «Техника микропроцессорных систем в коммутации» и предназначен для студентов радиотехнического факультета специальности «Сети связи и системы коммутации», но может использоваться и студентами других специальностей. Лабораторные работы посвящены проектированию и отладке программного обеспечения для микроконтроллеров AVR фирмы Atmel в среде Visual Micro Lab (VMLab).

Сборник подготовлен на кафедре «Телекоммуникации».

УДК 621.391 (076)
ББК 32я7

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Микроконтроллеры семейства AVR отличаются высоким быстродействием и низким энергопотреблением. В лабораторных работах рассматриваются структура, система команд, периферийные устройства и работа микроконтроллеров, выпускаемых фирмой Atmel.

Корпорация Atmel (США) хорошо известна как на мировом, так и на российском рынке электронных компонентов и является одним из признанных мировых лидеров в разработке и производстве сложных изделий современной микроэлектроники — устройств энергонезависимой памяти высокого быстродействия и минимального удельного энергопотребления, микроконтроллеров общего назначения и микросхем программируемой логики.

Лабораторные работы познакомят Вас с одним из интересных и активно развиваемых Atmel Corp. направлений современной микроэлектроники – линией 8–разрядных высокопроизводительных RISC (Reduced Instruction Set Computers) микроконтроллеров общего назначения, объединенных общей маркой AVR.

Можно считать, что AVR постепенно становится еще одним индустриальным стандартом среди 8–ми разрядных микроконтроллеров общего назначения. В настоящее время в производстве у Atmel Corp. находятся три семейства AVR: «tiny», «classic» и «mega».

Visual Micro Lab (VMLab) представляет собой программный продукт, позволяющий производить моделирование, тестирование, разработку и отладку программных и технических средств микропроцессорных систем, содержащих микроконтроллеры AVR фирмы ATmel и различные компоненты, в том числе резисторы, конденсаторы, ключи, светодиоды, аналоговые, импульсные и другие генераторы, источники напряжения, операционные усилители и компараторы, логические элементы, 8–ми разрядные цифроаналоговые преобразователи, интерфейс RS232, LCD модули, I2C интерфейсы и другие устройства. С помощью пакета можно проектировать простейшую мультипроцессорную сеть, состоящую из двух микроконтроллеров (микропроцессорных систем) и проводить их совместное моделирование и тестирование.

Для работы программного комплекса необходим IBM – совместимый компьютер с процессором Pentium 4 и выше. Операционная система Windows XP и выше.

Эффективность использования пакета VMLab определяется:

- достаточно простым интерфейсом пользователя;
- большим количеством моделей микроконтроллеров и компонентов микропроцессорной системы;
- возможностью разрабатывать программы на языке высокого уровня СИ;
- свободным распространением пакета VMlab (версия 3.12) и транслятора с языка СИ WinAVR.

Лабораторная работа №1

ЗНАКОМСТВО С ПРОГРАММОЙ VISUAL MICRO LAB ПРОГРАММА ВВОДА ВЫВОДА ЧЕРЕЗ UART

Цель работы: Освоение основных правил разработки и отладки прикладного программного обеспечения на языке AVR Ассемблере для микропроцессорных систем на базе AVR микроконтроллеров фирмы Atmel.

Программно–аппаратные средства поддержки программирования AVR микроконтроллеров

Подготовка программ для микроконтроллера выполняется на персональном компьютере и состоит из следующих этапов:

– создание текста программы;

– трансляция текста в машинные коды и исправление синтаксических ошибок;

– отладка программы, то есть устранение логических ошибок;

– окончательное программирование микроконтроллера.

Каждый из этапов требует использования специальных программных и аппаратных средств. Программные и аппаратные средства разрабатываются как производителями микроконтроллеров, так и независимыми фирмами. Они всегда ориентированы на конкретную архитектуру микроконтроллера. Программные средства обычно оформляются в виде интегрированной отладочной среды и могут распространяться бесплатно, условно– бесплатно или на платной основе. Программные средства VMLab (версия 3.12) и транслятор с языка СИ WinAVR распространяются свободно и бесплатно.

Текст программы, создаваемый на первом этапе проектирования, оформляется в виде файла на языке ассемблера (с расширением .asm). Этот файл является входным для программ–трансляторов, которые, в свою очередь, создают новые файлы, ориентированные на использование с конкретными отладочными средствами. Обычно это:

– файл–листинг (с расширением .lst),

– объектный файл (с расширением .obj),

– файл–прошивка FLASH–памяти (с расширением .hex),

– файл–прошивка EEPROM–памяти (с расширением .eep).

Файл–листинг – это отчет транслятора о своей работе. В нем приводится транслируемая программа в виде исходного текста, каждой строке которого сопоставлены соответствующие двоичные коды. Кроме того, листинг содержит сообщения о выявленных ошибках.

Объектный файл, создаваемый программой ассемблером, используется в дальнейшем как входной для программы–отладчика и имеет специальный формат.

Файлы прошивки FLASH и EEPROM блоков памяти предназначены для работы с последовательными и параллельными программаторами и имеют стандартные форматы.

Кроме языка ассемблера, для программирования встраиваемых микропроцессоров широкое распространение получили языки программирования высокого уровня. Для микроконтроллеров AVR разработан транслятор с языка СИ WinAVR, работающий совместно с пакетом VMLab. Пакет WinAVR предоставляет программисту такой же легкий доступ ко всем ресурсам микроконтроллера, как и ассемблер, но вместе с тем дают возможность создавать хорошо структурированные программы, снимает с программиста заботу о распределении памяти данных и содержат большой набор библиотечных функций для выполнения стандартных операций.

Отладка программ микроконтроллеров может выполняться двумя основными способами: на персональном компьютере при помощи программы–симулятора или в реальной микропроцессорной системе. Два эти способа взаимно дополняют друг друга.

Программы–симуляторы, к которым относится среда VMLab, отображают на экране компьютера программу пользователя и состояние внутренних регистров микроконтроллера. В результате, появляется возможность для наблюдения за изменениями в регистрах, памяти и процессорном ядре микроконтроллера при выполнении тех или иных команд программы. В реальной системе состояние внутренних регистров микроконтроллера посмотреть при помощи измерительных приборов невозможно. Использование симуляторов эффективно при отладке подпрограмм, которые выполняют численную обработку внутренних данных.

Внутрисхемные эмуляторы являются специальными схемами, которые с одной стороны связываются с проектируемой микропроцессорной системой через панель, предназначенную для установки микроконтроллера, а с другой – с персональным компьютером и работают под управлением установленного на компьютере программного обеспечения. Внутрисхемные эмуляторы позволяют выполнять программу в системе в пошаговом режиме и неограниченное число раз вносить изменения в программу. При работе с внутрисхемным эмулятором на экране компьютера можно наблюдать состояние внутренних ресурсов процессора, а на микропроцессорной плате – реакцию системы на те или иные команды программы.

Окончательная отладка программного обеспечения производится в рабочей системе. Обычно производители микроконтроллеров предлагают пользователям различные аппаратные средства для создания такой системы. Например, для разработки микропроцессорных систем на основе AVR–микроконтроллеров фирма Atmel выпускает стартовый набор AVR STARTER KIT типа STK500.

Завершающим этапом программирования микроконтроллера является занесение в память уже отлаженной программы. Оно может быть выполнено так же, как и при отладке программы, то есть через SPI–интерфейс. Если в системе

SPI–интерфейс не предусмотрен, то необходимо использовать программаторы, которые выполняют параллельное программирование. Параллельные программаторы обычно являются универсальными устройствами и позволяют работать с различными микроконтроллерами.

Краткие теоретические сведения

Память

В соответствии с гарвардской архитектурой память AVR–микроконтроллера разделена на две области: память данных и память программ. Кроме того, ATmega16 содержит память EEPROM для энергонезависимого хранения данных.

ATmega16 содержит 16 кбайт внутренней системно перепрограммируемой флэш–памяти для хранения программы (рис.1).

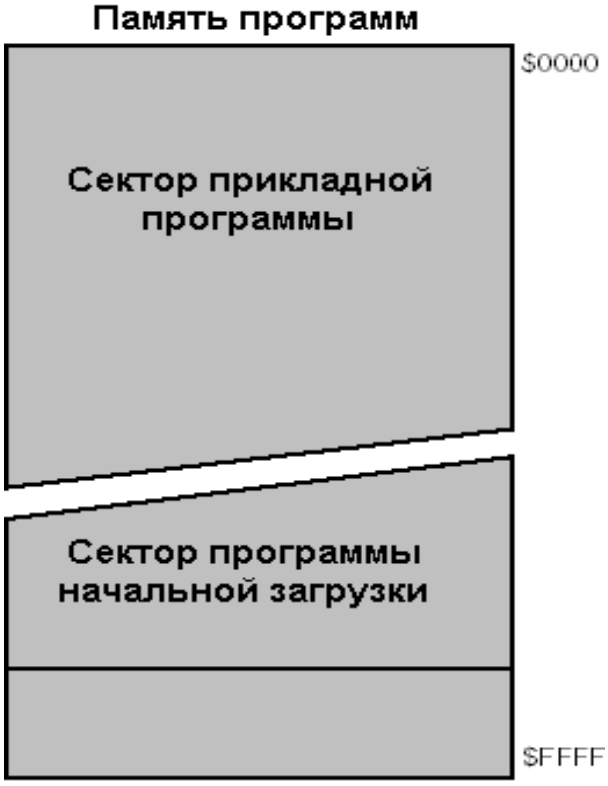


Рис. 1. Память программ

Первые 1120 ячеек памяти данных относятся к файлу регистров (32), памяти ввода/вывода (64) и встроенному статическому ОЗУ (1024).

Реализовано пять различных способов адресации для охвата всей памяти: прямая, косвенная со смещением, косвенная, косвенная с предварительным декрементом и косвенная с последующим инкрементом. Регистры R26 – R31 из файла регистров используются как регистры–указатели для косвенной адресации.

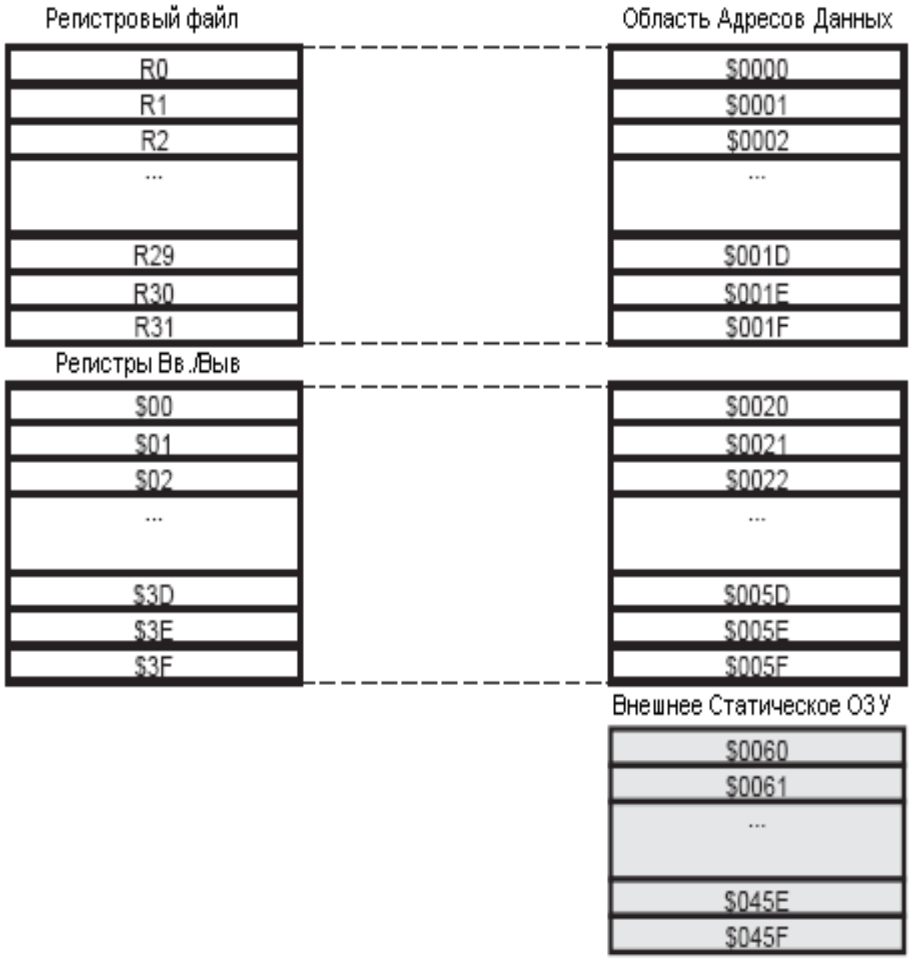


Рис. 2. Распределение памяти в ATmega 16

Порты ввода/вывода

Порт B – 8–разрядный двунаправленный порт. Для обслуживания порта отведено три регистра: регистр данных – PORTB ($18, $38), регистр направления данных – DDRB ($17, $37) и выводы порта B – PINB ($16, $36). Адрес выводов порта B предназначен только для чтения, в то время как регистр данных и регистр направления данных – для чтения/записи.

Все выводы порта имеют отдельно подключаемые подтягивающие резисторы. Дополнительные функции выводов порта B приведены в таблице 1.

Таблица 1. Альтернативные функции выводов порта B

Вывод	Альтернативная функция
PB0	XCS/T0 (внешний вход таймера счетчика 0)
PB1	T1 (внешний вход таймера счетчика 1)
PB2	INT2/AIN0 (Вход прерывания 2/ Положительный вход аналогового компаратора)
PB3	OC0/AIN1 (Выход совпадения TC0/Отр. вход аналогового компаратора)
PB4	SS
PB5	MOSI (Вход данных для загрузки памяти)
PB6	MISO (Выход данных для чтения памяти)
PB7	SCK (Вход тактовых импульсов последовательного обмена для зап./чт. памяти)

Для порта D зарезервированы 3 ячейки памяти – регистр PORTD ($12, $32), регистр направления данных – DDRD ($11, $31) и выводы порта D – PIND ($10, $30). Регистры данных и направления данных могут читаться/записываться, ячейка PIND – только для чтения.

Порт D – 7-разрядный двунаправленный порт со встроенными подтягивающими регистрами. Некоторые из выводов порта имеют альтернативные функции, как показано в таблице 2.

Таблица 2. Альтернативные функции порта D

Вывод порта	Альтернативная функция
PD0	RXD (вход данных UART)
PD1	TXD (выход данных UART)
PD2	INT0 (вход внешнего прерывания 0)
PD3	INT1 (вход внешнего прерывания 1)
PD4	OC1B
PD5	OC1A
PD6	ICP (вход захвата таймера счетчика 1)

Порт A (PA0 – PA7) имеет альтернативные функции входов аналого–цифрового преобразователя ADC – ADC7. Для порта зарезервированы 3 ячейки памяти – регистр PORTA ($1B, $3B), регистр направления данных – DDRA ($1A, $3A) и выводы порта A – PIND ($19, $39).

Порт C (PC0 – PC7) также имеет альтернативные функции, представленные в приложении 2. Для порта зарезервированы 3 ячейки памяти: регистр PORTC ($15, $35), регистр направления данных – DDRC ($14, $34) и выводы порта C – PINC ($13, $33).

Последовательный иньерфейс ввода/вывода UART

Порт UART содержит передатчик, приемник, тактовый генератор и аппаратуру управления передачей и приемом. В состав порта входят регистр данных передатчика UDR (T), регистр данных приемника UDR (R), регистр управления UCR или UCSRB (№ $0A), регистр состояния USR или UCSRA (№ $0B), регистр задания скорости передачи/приема UBRR (№ $09) и другие элементы.

Передатчик готов к работе при установке в единичное состояние разряда TXEN регистра управления UCR. Передача кадра начинается при загрузке байта в регистр UDR (T). Загрузку можно выполнять при единичном состоянии разряда UDRE регистра состояния USR. При сбросе микроконтроллера в исходное состояние устанавливается UDRE = 1.

Загруженный байт передается в сдвигающий регистр передатчика TSR и происходит выдача кадра на выход микроконтроллера TXD.

При нулевом состоянии разряда CHR9 регистра UCR формируется кадр из десяти битов. При CHR9 = 1 кадр содержит одиннадцать битов. Значение дополнительного бита в этом случае должно быть указано в разряде TXB8 регистра UCR.

Первый байт при загрузке немедленно передается в регистр TSR и разряд UDRE регистра USR сохраняет единичное состояние, что позволяет сразу после загрузки первого байта загружать в регистр UDR (T) второй байт. Второй и последующие байты сохраняются в регистре UDR(T) до завершения выдачи из регистра TSR предыдущего кадра. При этом разряд UDRE регистра USR находится в нулевом состоянии и загрузка очередного байта в регистр UDR (T) запрещена.

При завершении выдачи кадра из регистра TSR и отсутствии очередного байта в регистре UDR (T) устанавливается в единичное состояние разряд TXC регистра USR и при единичном состоянии разряда TXCIE регистра UCR в блок прерываний поступает запрос прерывания UART TXC.

Разряд TXC регистра USR сбрасывается в нулевое состояние аппаратно при переходе микроконтроллера к выполнению соответствующей прерывающей программы или программно при выполнении команды установка бита в единичное состояние.

При единичном состоянии разряда UDRE регистра USR и единичном состоянии разряда UDRIE регистра UCR в блок прерываний поступает запрос прерывания UART DRE. Разряд UDRE сбрасывается в нулевое состояние при записи байта в регистр UDR (T). Прерывающая программа, выполняемая по запросу прерывания UART DRE, должна содержать команду записи в регистр UDR (T) для прекращения действия этого запроса прерывания.

Приемник готов к работе при установке в единичное состояние разряда RXEN регистра UCR. При этом вход приемника RXD подключается к выводу определенного порта микроконтроллера. Получаемая последовательность значений вводится в сдвигающий регистр приемника RSR. Если принимается кадр из одиннадцати битов (CHR9 = 1), дополнительный бит принимается в разряд RXB8 регистра UCR. Если на месте ожидаемого стопового бита сигнал имеет нулевое значение, устанавливается в единичное состояние разряд FE регистра USR (ошибка формата). Разряд FE сбрасывается в нулевое состояние при появлении единичного значения стопового бита.

Принятый байт из регистра RSR переписывается в регистр UDR (R). При этом устанавливается в единичное состояние разряд RXC регистра USR и при единичном состоянии разряда RXCIE регистра UCR в блок прерываний поступает запрос прерывания UART RXC.

Разряд RXC регистра USR сбрасывается в нулевое состояние при чтении регистра UDR (R). Прерывающая программа, выполняемая по запросу прерывания UART RXC, должна содержать команду чтения из регистра UDR для прекращения действия этого запроса.

Если при завершении приема кадра принятый ранее байт не считан из регистра UDR (R), устанавливается в единичное состояние разряд OR регистра USR (состояние переполнения). Разряд OR сбрасывается в нулевое состояние при передаче байта из регистра RSR в регистр UDR (R).

Скорость передачи и приема BR, бит/с, зависит от частоты тактового сигнала микроконтроллера F_{CK} и числа (UBRR), записаного в регистре UBRR.

В порте UART микроконтроллеров типа m16 регистр управления вместо имени UCR имеет имя UCSRB, а регистр состояния вместо имени USR — имя UCSRA. Регистр UCSRA имеет дополнительный разряд MPCM. Наличие разряда MPCM позволяет организовать простейшую локальную сеть (мультипроцессорную систему).

В микроконтроллере типа m16, кроме того, регистр UCSRA имеет дополнительный разряд U2X, а для задания скорости передачи используются два регистра — регистр UBRR (№$09) для задания младших восьми разрядов кода числа и регистр UBRRHI (№$20) для задания старших четырех разрядов кода числа.

Выводы PD0 и PD0 микроконтроллера ATmega16 используются альтернативно как сигналы RXD и TXD соответственно порта UART (USART).

Управление USART осуществляется через регистры ввода/вывода. В контроллере ATmega16 для управления используется 5 регистров:

- Регистр UDR *(UART Data Register)* – регистр данных UART;
- Регистр UCSRA *(UART Control and Status Register A)* –регистр А управления и статуса UART;
- Регистр UCSRB *(UART Control and Status Register B)* – регистр В управления и статуса UART;
- Регистры UBRRH1 и UBRR *(UART Baud Rate registers)* – регистры скорости передачи.

Регистр данных UDR физически является двумя регистрами: регистром передачи данных и регистром приема данных по одному адресу $0C ($2C).

Скорость обмена данными в UART задается с помощью 12–битного регистра UBRR, который размещается в двух 8–битных регистрах UBRR, UBRRH1.

Установленный в состояние 1 бит TXEN регистра UCSRB разрешает передачу данных UART. Передача инициируется записью передаваемых данных в регистр данных UDR. Данные пересылаются из UDR в сдвиговый регистр передачи в следующих случаях:

- Новый символ записан в UDR после того как был выведен из регистра столовый бит предшествовавшего символа. Сдвиговый регистр загружается немедленно.
- Новый символ записан в UDR прежде, чем был выведен столовый бит предшествовавшего символа. Сдвиговый регистр загружается после выхода стопового бита передаваемого символа, находившегося в сдвиговом регистре.

Если из 10(11)–разрядного сдвигового регистра передачи выведена вся информация, устанавливается бит UDRE. При установленном в состояние 1 бите UDRE приемопередатчик готов принять следующий символ. Запись в UDR очищает бит UDRE. В то самое время, когда данные пересылаются из UDR в 10(11)–разрядный сдвиговый регистр, бит 0 сдвигового регистра сбрасывается в состояние 0 (состояние 0 – стартовый бит) а бит 9 или 10 устанавливается в состояние 1 (состояние 1 – стоповый бит). Если в регистре управления UCSRB установлен бит CHR9 (т.е. выбран режим 9–разрядного слова данных),

то бит TXB8 регистра UCSRB пересылается в бит 9 сдвигового регистра передачи. Сразу после пересылки данных в сдвиговый регистр тактом бод–генератора стартовый бит сдвигается на вывод TxD. За ним следует LSB данных. Когда будет выдан стоповый бит, сдвиговый регистр загружается новой порцией данных, если она была записана в UDR во время передачи. В процессе загрузки бит UDRE находится в установленном состоянии. Если же новые данные не будут загружены в UDR до выдачи стопового бита, флаг UDRE остается установленным. В этом случае, после того как столовый бит будет присутствовать на выводе TxD в течение одного такта, в регистре управления и статуса UCSRA устанавливается флаг завершения передачи TxC .

Логика восстановления данных производит выборку состояний вывода RxD с частотой в 16 раз большей, чем частота передачи. При нахождении линии в пассивном состоянии одиночная выборка нулевого логического уровня будет интерпретироваться как падающий фронт стартового бита и будет запущена последовательность детектирования стартового бита. Считается, что первая выборка обнаружила первый нулевой логический уровень вероятного стартового бита. На выборках 8, 9 и 10 приемник вновь тестирует вывод RxD на изменение логических состояний. Если две или более из этих трех выборок обнаружат логические 1, то данный вероятный стартовый бит отвергается как шумовой всплеск и приемник начнет выявлять и анализировать следующие переходы из 1 в 0.

Если же был обнаружен действительный стартовый бит, то начинает производиться выборка следующих за стартовым битом информационных битов. Эти биты также тестируются на выборках 8,9 и 10. Логическое состояние бита принимается по двум и более (из трех) одинаковым состояниям выборок. Все биты вводятся в сдвиговый регистр приемника с тем значением, которое было определено тестированием выборок.

При поступлении стопового бита необходимо, чтобы не менее двух выборок из трех подтвердили прием стопового бита (показали высокий уровень). Если же две или более выборок покажут состояния 0, то при пересылке принятого байта в UDR в регистре управления и статуса UCSRA устанавливается бит ошибки кадра FE *(Framing Error).* Для обнаружения ошибки кадра пользователь перед чтением регистра UDR должен проверять состояние бита FE. Флаг FE очищается при считывании содержимого регистра данных UART (UDR).

Вне зависимости от того принят правильный столовый бит или нет, данные пересылаются в регистр UDR и устанавливается флаг RXC в регистре управления UCSRA. Регистр UDR фактически является двумя физически отдельными регистрами, один из которых служит для передачи данных и другой для приема. При считывании UDR обращение ведется к регистру приема данных, при записи обращение ведется к регистру передачи. Если выбран режим обмена 9–разрядными словами данных (установлен бит CHR9 регистра UCR), при пересылке данных в UDR бит RXB8 регистра UCR загружается из девятого бита сдвигового регистра передачи. Если после получения символа к регистру UDR не было обращения, начиная с последнего приема, в регистре

UCSRA устанавливается флаг переполнения OR. Это означает, что новые данные, пересылаемые в сдвиговый регистр, не могут быть переданы в UDR и потеряны. Бит OR буферирован и доступен тогда, когда в UDR читается байт достоверных данных. Пользователю, для обнаружения переполнения, необходимо всегда проверять флаг OR после считывания содержимого регистра UDR.

При очищенном (сброшенном в логическое состояние 0) бите RXEN регистра UCR прием запрещен.

Регистр UCSRA имеет биты следующего назначения:

- Бит 7 - RXC - прием завершен.
- Бит 6 - TXC - передача завершена.
- Бит 5 - UDRE - регистр данных пуст.
- Бит 4 - FE - ошибка кадра.
- Бит 3 - OR - переполнение данных.
- Бит 1 - U2X - удвоение скорости передачи.
- Бит 0 - MPCM - режим мультипроцессорного обмена.

Регистр UCSRB имеет биты следующего назначения:

- Бит 7 - RXCIE - разрешение прерывания по завершению приема.
- Бит 6 - TXCIE - разрешение прерывания по завершению передачи.
- Бит 5 - UDRIE - Разрешение прерывания по пустому регистру данных.
- Бит 4 - RXEN - разрешение приемника.
- Бит 3 - TXEN - разрешение передатчика.
- Бит 2 -CHR9 - режим 9–разрядных символов.
- Бит 1 -RXB8 - прием 8–разрядных данных.
- Бит 0 -TXB8 - передача 8–разрядных данных.

Интегрированная отладочная среда VMLab

Интегрированная отладочная среда обычно включает в себя компилятор с языков ассемблера и СИ и позволяет пользователю полностью контролировать выполнение программ с использованием симулятора, который поддерживает различные типы микроконтроллеров. Например, отладочная среда VMLab поддерживает выполнение программ в виде ассемблерного текста формата AVR Assembler и в формате языка СИ компилятора GCC и WinAVR.

Пользователь может выполнять программу полностью в пошаговом режиме, трассируя блоки функций, или выполняя программу до места, где стоит курсор. В дополнение можно определять неограниченное число точек останова, каждая из которых может быть включена или выключена. Интегрированная отладочная среда VMLab используется с встроенным имитатором AVR.

Создание проекта в VMLab

Для работы над новым проектом создается папка проекта с произвольным именем, например на диске C:. В папку необходимо скопировать файл инициализации выбранного для реализации проекта контроллера, например,

«m16def.inc». Этот файл включен в прикладное программное обеспечение VMLab.

Программа VMLab функционирует в среде Windows 95/98/2000/XP/NT. Значок запуска программы изображен на рис.3.

В результате запуска AVR Studio на экране появляется окно программы, представленное на рис.4.

Для создания нового проекта необходимо в меню **Project** выбрать команду **New Proect**. В диалоговом окне (рис. 5) необходимо ввести название проекта *(Project name)* и его расположение *(Location).*

Рис. 3.Значок программы

Далее выполняются действия, указанные в Step1, Step2, Step3, Step4.

К проекту должен быть добавлен файл программы на языке ассемблера. Это можно сделать разными способами: или в проект добавляется уже существующий файл с расширением .asm, или создается новый.

После установки всех параметров проекта и нажатия на кнопку OK появляются два окна Code. Target File с программой на языке Ассемблера и окно файла проекта. Оба окна (рис.6.) допускают корректировку этих файлов встроенным текстовым редактором. Корректировка может быть выполнена также любым текстовым редактором вне среды VMLab.

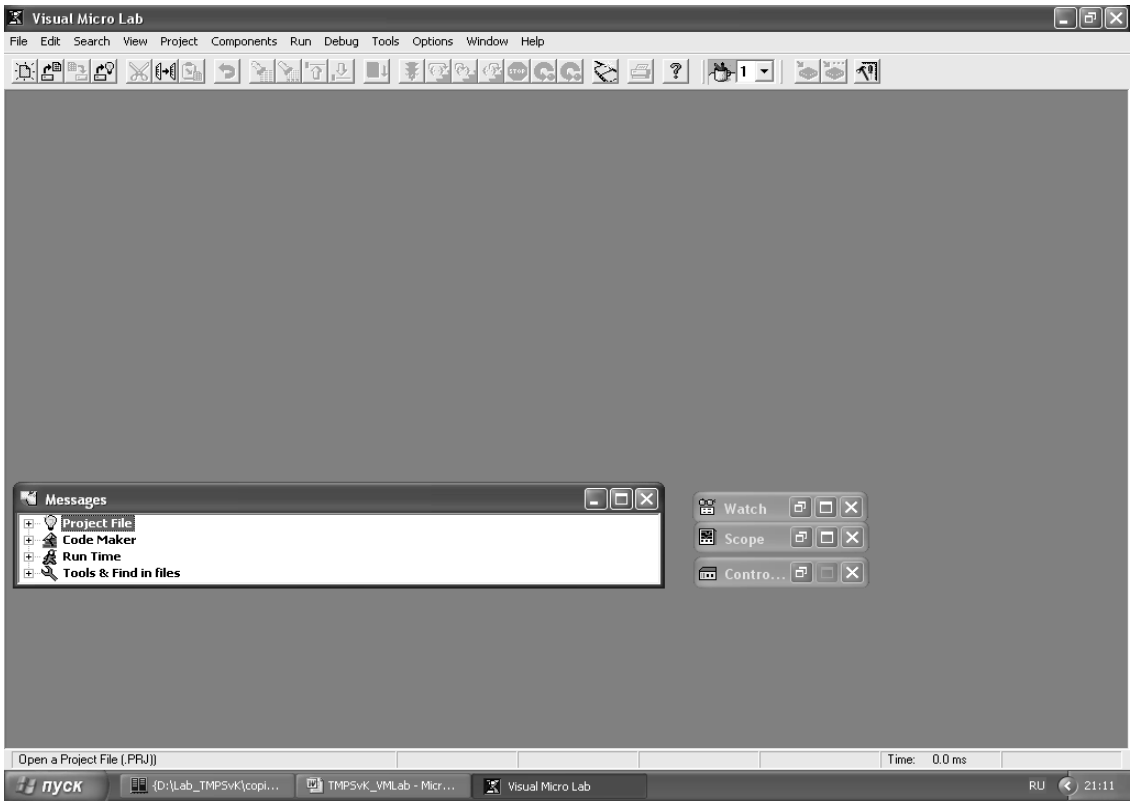


Рис. 4. Стартовое окно программы VNLab

Далее необходимо подключить к проекту необходимые компоненты (рис.7).

Выбрав компонент TTY (RS232), в окне проекта появляется соответствующая строка. Далее необходимо указать в окне проекта его имя, параметры и цепи подключения.

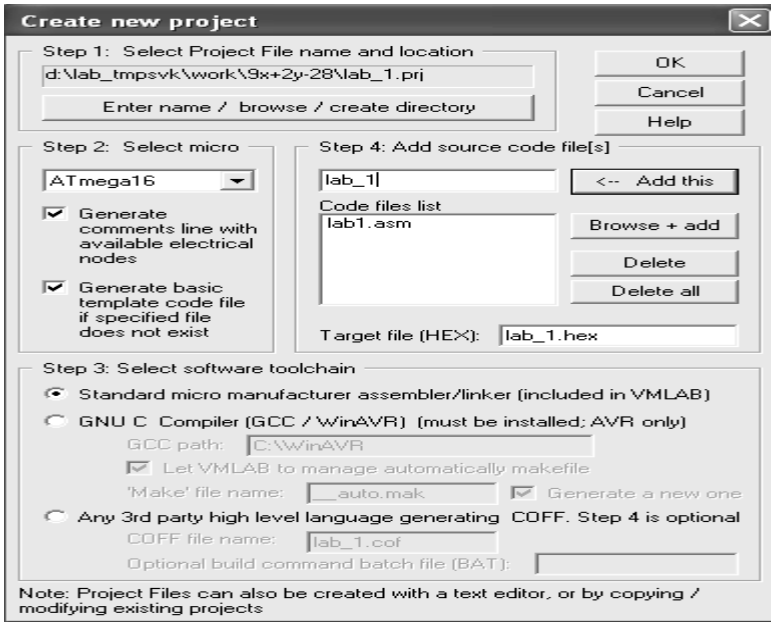


Рис. 5. Диалоговое окно нового проекта

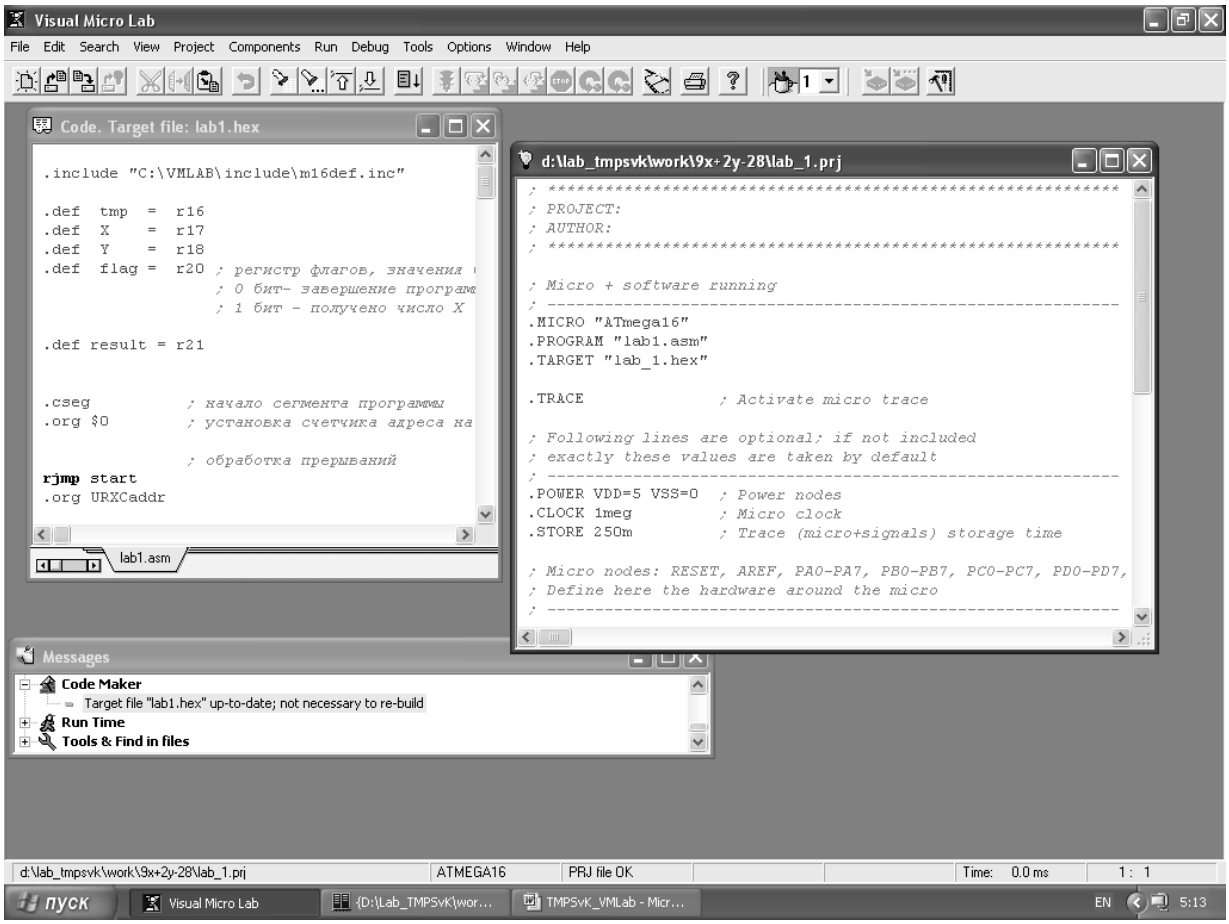


Рис. 6. Окна файла проекта и с программой

В проекте можно использовать различные компоненты, встроенные в VMLab. Для просмотра напряжений в различных узлах схемы проекта можно в файл проекта вставить директиву **.plot V(PD0) v(PD1)**, в которой указаны выводы 0 и 1 разрядов порта D микроконтроллера.

Перед тем, как начать трансляцию, а затем и запуск созданной и откомпилированной программы необходимо добавить в окно VMLab компоненты контроля и управления. Для этого в окне **View** необходимо выбрать соответствующие элементы.

Далее осуществляется трансляция программы и проверка правильности её написания. Выбирается пункт **Build** в меню **Project** рис.8**.** При этом создаются все файлы проекта и проверяется правильность написанной программы.

При трансляции без ошибок в окне **Messeges** появится об этом сообщение рис.9.

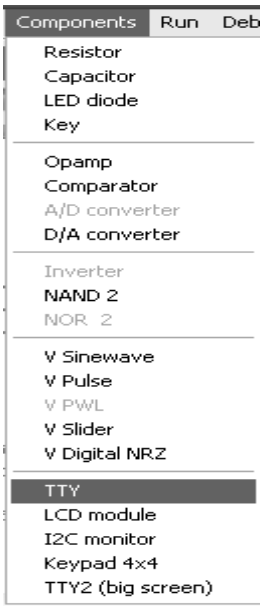


Рис. 7. Окно компонентов

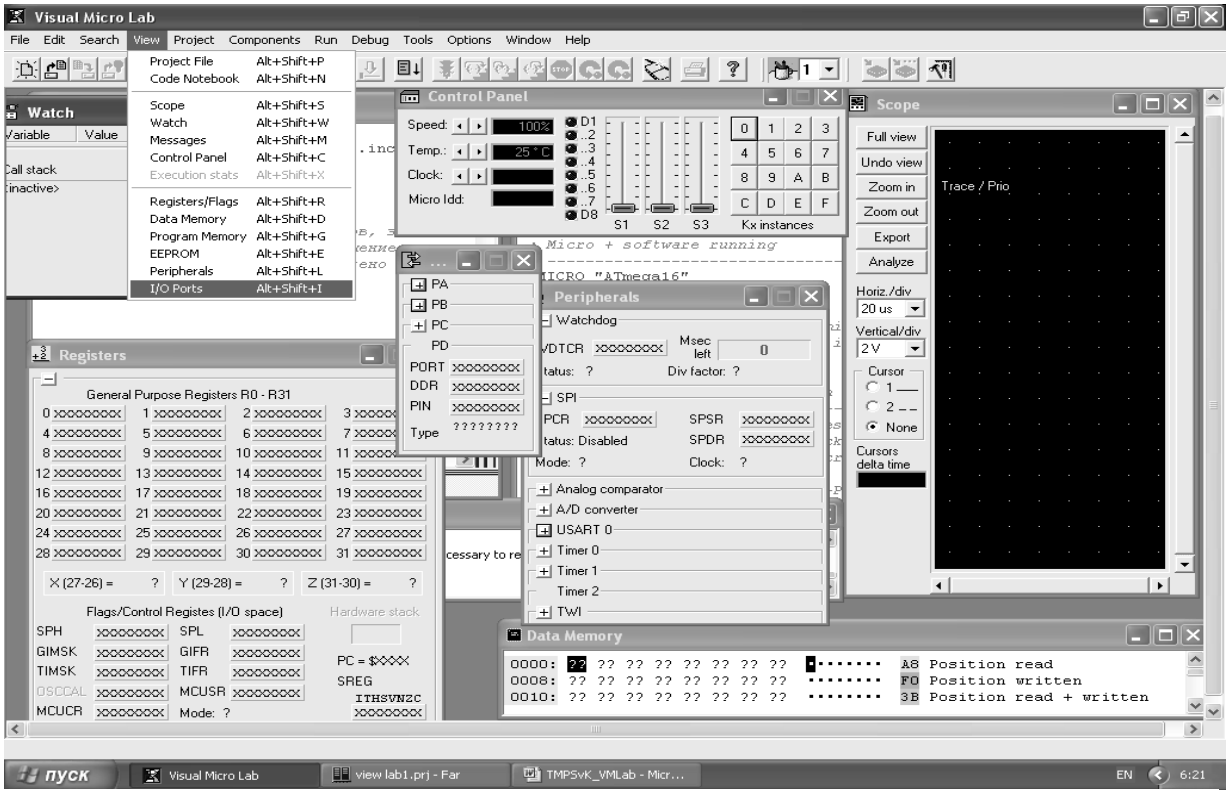


Рис. 8. Окна при трансляции и моделировании проекта

Рис. 9. Окно сообщений о результатах трансляции

Дальнейшая отладка программы осуществляется при ее запуске в автоматическом или пошаговом режиме рис.10.

Для запуска созданной и откомпилированной программы используется команда **GO/Continur** в меню **Run** или в пошаговом режиме команды **Step**.

Программа вводит через UART числа X и Y в коде ASCII. В показанном примере X=2 Y=3, после вычисления 9*X+2*Y–28, результат выводится в UART в коде ASCII в виде двухзначного десятичного числа, выводится минус, если результат – отрицательное число.

Для наблюдения за изменениями переменных можно использовать окно Watch.

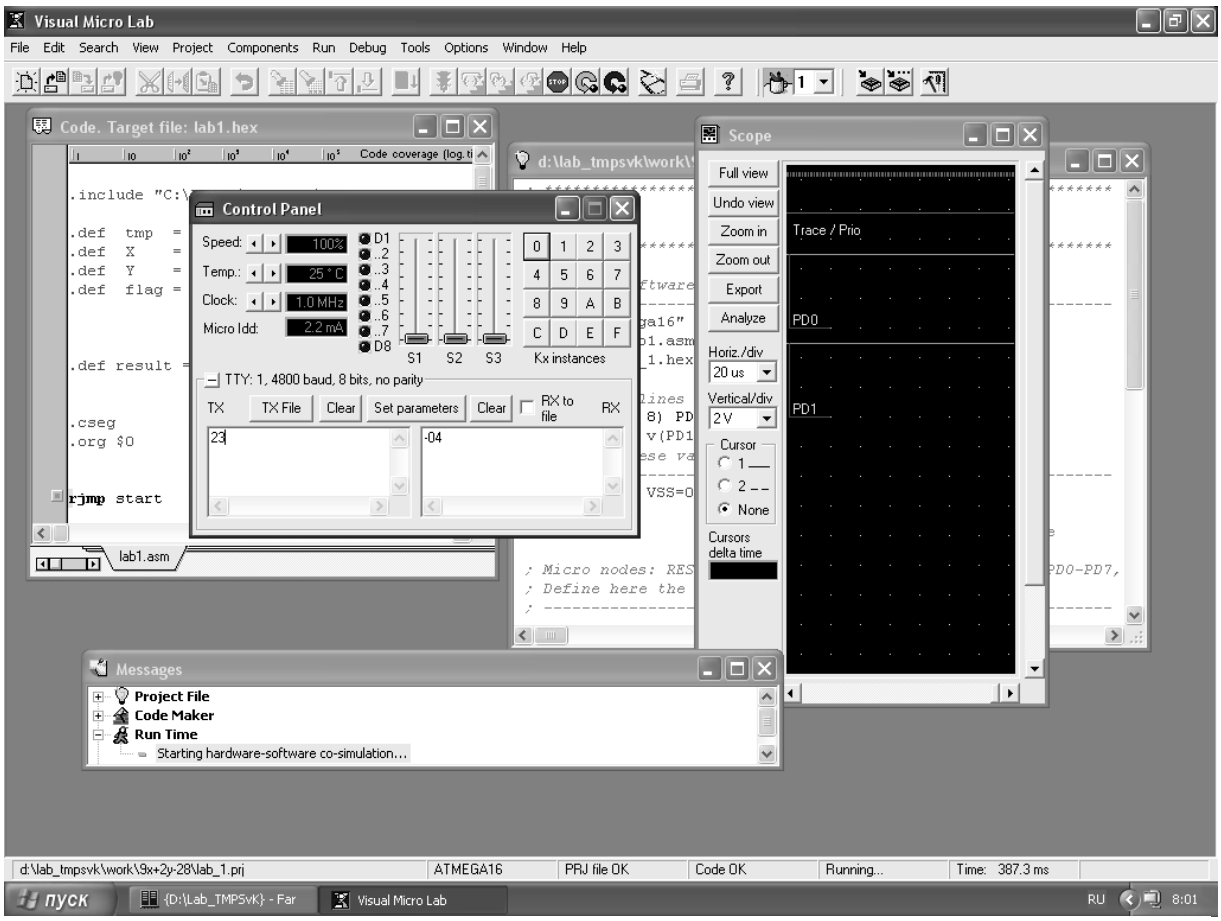


Рис. 10. Окна при запуске программы

Для просмотра программного счетчика, указателя стека, содержимого регистра статуса SREG и индексных регистров X,E и Z, памяти микроконтроллер и других регистров, а также состояния портов ввода/вывода и периферийных устройств через их регистры в процессе отладки программы можно воспользоваться соответствующими компонентами окна VMLab.

Задание к лабораторной работе

Студент самостоятельно осваивает все основные этапы разработки и отладки программ в среде VMLab, рассмотренные ранее.

В качестве примера используются файлы Lab1.asm и Lab_1.prj.

Файл Lab_1.prj

```
.MICRO "ATmega16"
.PROGRAM "lab1.asm"
.TARGET "lab_1.hex"
.TRACE            ; Activate micro trace
; Following lines are optional; if not included
X1  TTY(4800 8) PD0 PD1
.plot V(PD0) v(PD1)
.POWER VDD=5 VSS=0  ; Power nodes
.CLOCK 1meg         ; Micro clock
.STORE 250m         ; Trace (micro+signals) storage time
```

Файл Lab1.asm

```
.include "C:\VMLAB\include\m16def.inc"
.def  tmp  =  r16
.def  X   =  r17
.def  Y    =  r18
.def  flag =  r20 ; рег. флагов, значения битов будут определять ветвления в
                  :  программе  0 бит– завершение программы
                  ; 1 бит – получено число X
.def result = r21
.cseg               ; начало сегмента программы
.org $0             ; установка счетчика адреса на 0
                    ; обработка прерываний
rjmp start
.org URXCaddr
rjmp uart_rx
.org UDREaddr
reti
.org UTXCaddr
reti
                    ; начало программы
uart_rx:            ; ПП обработки прерывания –
                    ;получение символа в  UART
   in tmp, UDR      ; чтение значения из регистра данных
   subi tmp, $30    ; UART настроен на работу по умолчанию, поэтому
                    ; преобразовываем поступающий символ
   sbrc flag, 0     ; проп. возврат, если нет флага завершения (еще не все
                    ; символы приняты)
 reti
```

```
 sbrc flag, 1           ; решаем, какое число принимается X или Y
 rjmp setY
 rjmp setX
  reti
start:                  ; инициализация программы
 ldi flag, 0            ; обнуление флагов
 ldi tmp, high(RAMEND)  ; инициализация памяти стека
 out SPH, tmp
 ldi tmp, low(RAMEND)
 out SPL, tmp           ; завершение инициализации памяти стека
  ldi tmp, 12
  out UBRR, tmp         ; установка скорости UART 4800 бод
  ldi tmp, 0xF8
  out UCR, tmp          ; инициализация UART
  sei                   ; глобальное включение прерываний
forever :       ; бесконечный цикл, все действия по прерываниям
  nop
  nop
  rjmp forever;
setX:                   ; получаем число X и сообщаем об этом установкой флага
 mov X, tmp
 sbr flag, $02
 reti
setY:             ; получаем число Y и после этого переходим к вычислениям
 mov Y,tmp
 sbr flag, $01
 rjmp main
 main:                  ; вычисление значения
  ldi tmp, 9
  mul tmp, X
  mov result, R0
  ldi tmp, 2
  mul tmp, Y
  add result, R0
  subi result, 28
  brmi negativ  ; если получ. отр. число, осуществляем переход
  rjmp bcd_result

negativ:                ; выделяем модуль числа и выводим символ «–»
 ldi tmp, $2D           ; вывод ACII символа «–»
 out UDR, tmp           ; получаем модуль числа
 com result
 inc result
 rjmp bcd_result
```

```
bcd_result:    ; преобр. число: десятки будут храниться в X, а единицы в –Y
   ldi X, 0
   mov Y, result
   subi Y, 10
   brmi bcd_done
   inc X
   subi Y, 10
   brmi bcd_done
   inc X
   subi Y, 10
   brmi bcd_done
   inc X
   subi Y, 10
   brmi bcd_done
   inc X
   subi Y, 10
   brmi bcd_done
   inc X
   subi Y, 10
   brmi bcd_done
   inc X
   subi Y, 10
   brmi bcd_done
   inc X

   subi Y, 10
   brmi bcd_done
   inc X
   subi Y, 10
   brmi bcd_done
   inc X
  subi Y, 10
   brmi bcd_done
   inc X
bcd_done:
 com Y
 inc Y
 subi Y, 10
 com Y
 inc Y
 ldi tmp, $30
 add X, tmp
 add Y, tmp
 out UDR, X
```

```
wait:                ; ждем, пока завершится вывод числа X
in tmp, UCSRA
sbrs tmp, 5
rjmp wait
out UDR, Y
rjmp forever
```

Программа «Lab1.asm» осуществляет ввод символов (двух чисел от 0 до 9 каждый) через COM порт (UART, S232), результат вычисления также выводится через этот же интерфейс.

Порядок выполнения лабораторной работы

Необходимо выполнить все необходимые операции с проектом Lab_1.prj и программой Lab1.asm в среде VMLab.

Содержание отчета

Отчет по проделанной лабораторной работе должен содержать:

- структурную схему алгоритма программы на лабораторную работу;
- схему электрическую принципиальную разработанного устройства;
- текст программы на языке AVR Ассемблере (распечатка файла *asm);
- распечатка файла проекта (*.prj);
- результаты моделирования (окно VMLab с запущенной программой и результатами вычислений).

Контрольные вопросы

1. Перечислите основные этапы работы в программе VMLab?
2. Какие типы файлов создаются в программе VMLab?
3. Какие компоненты используются совместно с микроконтроллером?
4. Какие дополнительные окна используются для просмотра и управления системой на основе выбранного микроконтроллера?
5. Каким образом можно просмотреть напряжения в зависимости от времени (временные диаграммы)?

Лабораторная работа № 2

ПРОГРАММА ВВОДА С UART И ВЫВОДА В LCD МОДУЛЬ

Цель работы: Изучение и освоение управления периферийным устройством ввода/вывода – LCD модулем, подключенным к AVR микроконтроллеру фирмы Atmel, а также дальнейшее изучение системы команд и системы прерываний.

Краткие теоретические сведения

Система прерываний микроконтроллера ATmega16

В ATmega16 предусмотрен 21 источник прерываний. Полный список векторов прерываний приведен в таблице 3.

Таблица 3. Сброс и векторы прерываний

Номер вектора	Адрес	Источник	Описание прерывания
1	2	3	4
1	$000	RESET	Вывод сброса и сброс от ст. таймера
2	$002	INT0	Внешнее прерывание 0
3	$004	INT1	Внешнее прерывание 1
4	$006	TIMER2 COMP	Совпадение таймера/счетчика 2
5	$008	TIMER2 OVF	Переполнение таймера/счетчика 2
6	$00A	TIMER1 CAPT	Захват таймера/счетчика 1
7	$00C	TIMER1 COMPA	Совпадение таймера/счетчика 1
8	$00E	TIMER1 COMPB	Совпадение таймера/счетчика 1
9	$010	TIMER1 OVF	Переполнение таймера/счетчика 1
10	$012	TIMER0 OVF	Переполнения таймера/счетчика 0

Окончание табл.3

1	2	3	4
11	$014	SPI, STC	Полная последовательная передача
12	$016	USART, RXC	Usart, Rx complete
13	$018	USART, UDRE	Usart регистр данных пустой
14	$01A	USART, TXC	Usart, Tx complete
15	$01C	ADC	Аналого–цифровой преобразователь
16	$01E	EE_RDY	EEPROM Ready
17	$020	ANA_COMP	Аналоговый компаратор
18	$022	TWI	Двухпроводной последовательный интерфейс
19	$024	INT2	Внешнее прерывание 2

20	$026	TIMER0 COMP	Совпадение таймера/счетчика 0
21	$028	SPM_RDY	Загрузка программной памяти готова

Чаще всего используется следующая установка векторов прерываний в программе:

```
Адрес    Метка        Код                  Комментарий
$000                  jmp RESET            ; Reset Handler
$002                  jmp EXT_INT0         ; IRQ0 Handler
$004                  jmp EXT_INT1         ; IRQ1 Handler
$006                  jmp TIM2_COMP        ; Timer2 Compare Handler
$008                  jmp TIM2_OVF         ; Timer2 Overflow Handler
$00A                  jmp TIM1_CAPT        ; Timer1 Capture Handler
$00C                  jmp TIM1_COMPA       ; Timer1 CompareA Handler
$00E                  jmp TIM1_COMPB       ; Timer1 CompareB Handler
$010                  jmp TIM1_OVF         ; Timer1 Overflow Handler
$012                  jmp TIM0_OVF         ; Timer0 Overflow Handler
$014                  jmp SPI_STC          ; SPI Transfer Complete Handler
$016                  jmp USART_RXC        ; USART RX Complete Handler
$018                  jmp USART_UDRE       ; UDR Empty Handler
$01A                  jmp USART_TXC        ; USART TX Complete Handler
$01C                  jmp ADC              ; ADC Conversion Complete
Handler
$01E                  jmp EE_RDY           ; EEPROM Ready Handler
$020                  jmp ANA_COMP         ; Analog Comparator Handler
$022                  jmp TWSI             ; Two–wire Serial Interface Han-
dler
$024                  jmp EXT_INT2         ; IRQ2 Handler
$026                  jmp TIM0_COMP        ; Timer0 Compare Handler
$028                  jmp SPM_RDY    ; Store Program Memory Ready Handler
;Начало программы, инициализация стека, глоб. разрешение прерываний
$02A  RESET:     ldi r16,high(RAMEND ) ; Main program start
$02B             out SPH,r16           ; Set Stack Pointer to top of RAM
$02C             ldi r16,low(RAMEND)
$02D             out SPL,r16
$02E             sei                   ; Enable interrupts
;Инструкции программы
$02F             <instr> xxx
…                …                     …
```

Обработка прерываний

В микроконтроллере ATmega16 регистр управления прерыванием GICR – общий регистр управления прерывания табл.4, регулирует размещение вектора прерывания. Когда возникает прерывание, общий бит разрешения прерываний I очищается (ноль) и прерывания запрещаются. Программа пользователя может

установить этот бит для разрешения прерываний. Флаг разрешения прерываний I устанавливается в 1 при выполнении команды выхода из прерывания – RETI.

Для прерываний, включаемых статическими событиями, флаг прерывания взводится, когда происходит событие. Если флаг прерывания очищен и присутствует условие возникновения прерывания, флаг не будет установлен, пока не произойдет следующее событие.

Когда программный счетчик устанавливается на текущий вектор прерывания для обработки прерывания, соответствующий флаг, сгенерированный прерыванием, аппаратно сбрасывается. Некоторые флаги прерывания могут быть сброшены записью логической единицы в бит, соответствующий флагу.

Табл. 4. Регистр GICR

Бит	7	6	5	4	3	2	1	0
	INT1	INT0	INT2	–	–	–	IVSEL	IVCE
Чт./зап.	R/W	R/W	R	R	R	R	R/W	R/W
Нач. знач.	0	0	0	0	0	0	0	0

Бит 1 – IVSEL: Выбор вектора прерывания.
Бит 0 – IVCE: Векторное изменение прерывания.

В таблице 5 представлена структура регистра режимы управления и начальные значения разрядов регистра общего регистра флагов GIFR.

Табл. 5. Регистр GIFR

Бит	7	6	5	4	3	2	1	0
Чт./зап.	INTF1	INTF0	INTF2	–	–	–	–	–
(R/W)	R/W	R/W	R	R	R	R	R	R
Нач.знач.	0	0	0	0	0	0	0	0

Бит 7 – INTF1: Флаг внешнего прерывания 1.
Бит 6 – INTF0: Флаг внешнего прерывания 0.
Бит 5 – INTF2: Флаг внешнего прерывания 2.

В таблице 6 представлена структура регистра, режимы управления и начальные значения разрядов регистра TIMSK.

Табл. 6. Регистр TIMSK

Бит	7	6	5	4	3	2	1	0
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0
Чт./зап.	R/W	R/W	R	R	R/W	R	R/W	R
Нач. знач.	0	0	0	0	0	0	0	0

Бит 7 – OCIE2: Разрешение прерывания по совпадению таймера/счетчика 2.

Бит 6 – TOIE2: Разрешение прерывания по переполнению таймера/счетчика 2.

Бит 5 – TICIE1: Разрешение прерывания по входу захвата.

Бит 4 – OCIE1A: Разрешение прерывания по совпадению таймера/счетчика 1.

Бит 3 – OCIE1B: Разрешение прерывания по совпадению таймера/счетчика 1.

Бит 2 – TOIE1: Разрешение прерывания по переполнению таймера/счетчика 1.

Бит 1 – OCIE0: Разрешение прерывания по совпадению таймера/счетчика 1.

Бит 0 – TOIE0: Разрешение прерывания по переполнению таймера/счетчика 0.

В таблице 7 представлена структура регистра режимы управления и начальные значения разрядов регистра TIFR.

Табл. 7. Регистр TIFR

Бит	7	6	5	4	3	2	1	0
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0
Чт./зап.	R/W	R/W	R	R	R/W	R	R/W	R
Нач.зн.	0	0	0	0	0	0	0	0

Бит 7 – OCF2: Флаг выхода совпадения Т/С2.

Бит 6 – TOV2: Флаг переполнения таймера/счетчика 2.

Бит 5 – ICF1: Флаг входа захвата 1.

Бит 4 – OCF1A: Флаг выхода совпадения 1А.

Бит 3 – OCF1B: Флаг выхода совпадения 1В.

Бит 2 – TOV1: Флаг переполнения таймера/счетчика 1.

Бит 1 – OCF0: Флаг выхода совпадения Т/С0.

Бит 0 – TOV0: Флаг переполнения таймера счетчика 1.

Внешние прерывания

Внешние прерывания управляются выводами INT0, INT1 и INT2. Эти прерывания обрабатываются даже тогда, когда выводы сконфигурированы как выходы.

Регистр состояния SREG аппаратно не обрабатывается процессором, если программа требует сохранения SREG, то это должно производиться программой пользователя.

Архитектура и система команд контроллера ЖКИ

Рассмотрим вопросы аппаратного и программного сопряжения микроконтроллеров AVR и символьных ЖКИ, построенных на базе контроллера **HD44780 (**LCD – модуль**)**. Рассматриваемый ЖКИ при помощи 14–контактного разъема (табл. 8) обменивается информацией с микроконтроллером AVR. Микроконтроллер посылает в ЖКИ команды (табл. 9) и ASCII–коды выводимых символов. В свою очередь, ЖКИ может посылать AVR–микроконтроллеру по его запросу информацию о своем состоянии и данные из своих внутренних блоков памяти.

Таблица 8. Описание выводов ЖКИ на базе HD44780

№	Название вывода	Описание
1	V_{SS}	(–) Питание. 0 V.
2	V_{DD}	(+) Питание.+5V.
3	V_0	Напряжение смещения, управляющее контрастностью
4	RS	Вход. Высокий уровень – Данные; Низкий – Команды
5	R/W	Вход. Высокий–Чтение, Низкий–Запись
6	E	Вход. Строб, сопровождающий сигналы на шине «команды/данные»
7 – 14	DB_0 – DB_7	Шина «команды/данные»

Три вывода 14–контактного разъема предназначены для подачи питающего напряжения и напряжения смещения, которое управляет контрастностью дисплея. На рис. 11 показана рекомендуемая схема подключения этих выводов.

Из оставшихся 11 выводов 8 (DB0 – DB7) используются для организации мультиплексированной шины «команды / данные», и на 3 вывода (RS, R/W, E) AVR–микроконтроллер выставляет управляющие сигналы. На рис. 12 изображены временные диаграммы этих сигналов при записи команд / данных в контроллер ЖКИ.При помощи сигнала на линии RS микропроцессор сообщает контроллеру индикатора о том, что именно передается по шине: команда или данные. Сигнал на линии E является стробом, сопровождающим сигналы на шине «команды / данные». Запись информации в ЖКИ происходит по спаду этого сигнала. Потенциал на управляющем выводе R/W задает направление передачи данных: запись в RAM индикатора (R/W=0) или считывание оттуда (R/W=1).

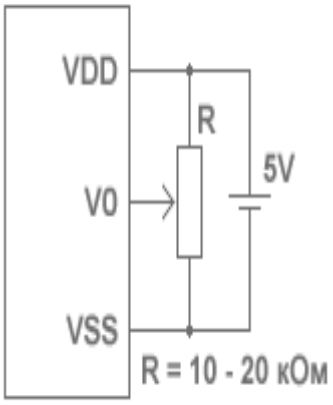


Рис. 11. Схема питания

Для случая, когда микроконтроллер имеет ограниченное количество линий ввода / вывода, предусмотрен второй вариант подключения ЖКИ с использованием 4–х разрядной шины «команды / данные». При этом каждый байт данных передается по линиям DB4 – DB7 последовательно двумя тетрадами, начиная со старшей. Контроллер ЖКИ после приема байта команды или байта данных требует некоторого времени табл. 9 для обработки полученной информации, в течение которого AVR–микроконтроллер не должен выполнять новых передач.

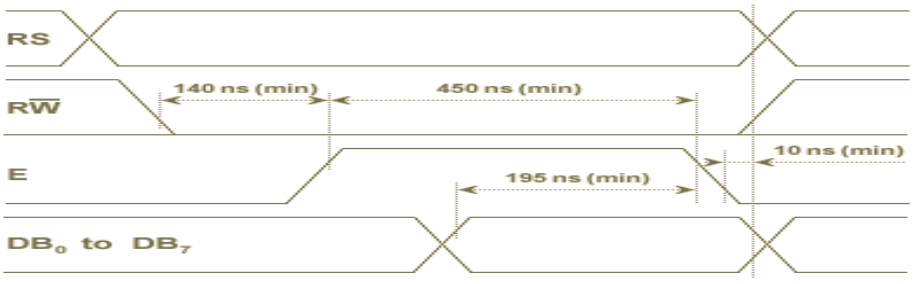


Рис. 12. Запись данных в ЖКИ

Для того, чтобы определить, когда контроллер ЖКИ закончит свои внутренние операции, AVR может опрашивать BUSY–флаг индикатора (команда «чтение busy–флага»). Второй, более простой способ заключается в том, что управляющий микроконтроллер просто выполняет временную задержку после каждой передачи информации.

Если во время цикла записи AVR–микроконтроллер передает в контроллер индикатора код команды, то этот код записывается в регистр команд контроллера ЖКИ, и команда сразу же начинает выполняться. Если AVR–микроконтроллер передает в контроллер ЖКИ данные, которые представляют собой ASCII–коды отображаемых символов, то они записываются в буфер данных (DDRAM), который обычно содержит 80 ячеек (рис. 13). При записи или считывании буфера данных обращение осуществляется к ячейке, на которую в данный момент указывает курсор.

Таблица 9. Система команд контроллера HD4478

Описание команды	Код RS,R/W, DB7–DB0	Время (f_{osc}=250кГц)
Очистить дисплей и установить курсор в нулевую позицию	0 0 0 0 0 0 0 0 0 1	82 мкс до 1.64 мс
Установить курсор в нулевую позицию (адрес 0). Установить дисплей относительно буфера DDRAM в начальную позицию. Содержимое DDRAM при этом не меняется.	0 0 0 0 0 0 0 0 1 *	40 мкс до 1.6 мс
Установить направление сдвига курсора вправо (I/D=1) или влево (I/D=0) при записи/чтении очередного кода в DDRAM. Разрешить (S=1) сдвиг дисплея вместе со сдвигом курсора.	0 0 0 0 0 0 0 0 I/D S	40 мкс
Включить(D=1)/выключить(D=0) дисплей. Зажечь(C=1)/погасить(C=0) курсор. Изображение курсора сделать мигающим (B=1).	0 0 0 0 0 0 1 D C B	40 мкс
Переместить курсор (S/C=0) или сдвинуть дисплей (S/C=1) вправо (R/L=1) или влево(R/L=0).	0 0 0 0 0 1 S/C R/L * *	40 мкс
Установить разрядность шины данных 4 бита (DL=0) или 8 бит (DL=1), количество строк дисплея – одна (N=0) или две (N=1), шрифт – 5x7 точек (F=0) или 5x10 точек (F=1).	0 0 0 0 1 DL N F * *	40 мкс
Установка адреса CGRAM. После этой команды данные будут записываться/считываться в/из CGRAM.	0 0 0 1 . . A_{CG} . .	40 мкс
Установка адреса DDRAM	0 0 1 . . A_{DD} . .	40 мкс
Чтение состояния busy–флага (BF) и счетчика адреса	0 1 BF . . AC . .	1 мкс
Запись данных в DDRAM или CGRAM.	1 0 Данные	40 мкс
Чтение данных из DDRAM или CGRAM.	1 1 Данные	40 мкс

Примечание:
DDRAM– Display data RAM – ОЗУ ASCII–кодов, отображаемых символов
CGRAM–Character generator RAM – ОЗУ знакогенератора

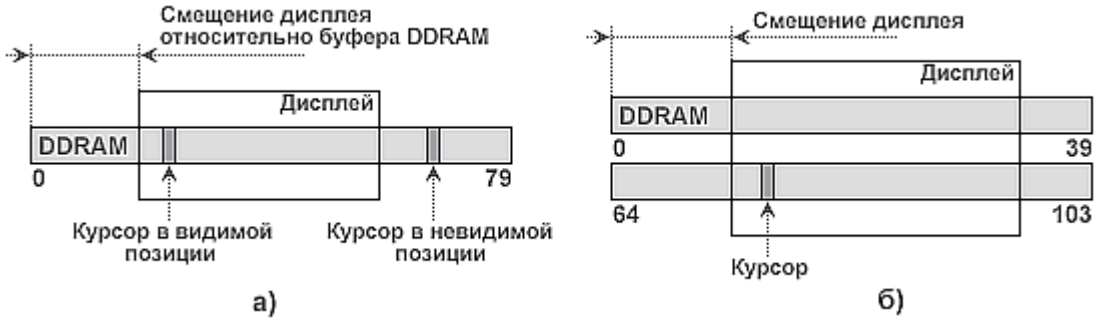


Рис. 13. Отображение на дисплее символов, ASCII–коды которых записаны в DDRAM (а – однострочный ЖКИ, б – двухстрочный ЖКИ)

Буфер данных имеет больше ячеек, чем число знакомест дисплея. Смещая окно индикатора относительно буфера данных (см. систему команд), можно отображать на дисплее различные области буфера. У индикаторов с двумя строками первые 40 ячеек буфера данных, обычно, отображаются на верхней строке дисплея, а вторые 40 ячеек – на нижней строке. Сдвиг окна дисплея относительно буфера данных для верхней и нижней строк происходит синхронно. Курсор будет виден на индикаторе только в том случае, если он попал в зону видимости дисплея (и если предварительно была подана команда – отображать курсор).

Кроме DDRAM, контроллер ЖКИ содержит еще один блок памяти – знакогенератор. Его «прошивка», то есть соответствие ASCII–кодов начертанию символов, обычно имеется в описании индикатора. Знакогенератор состоит из двух частей. Основная его часть представляет собой ПЗУ (CGROM) и ее, следовательно, нельзя изменить. Вторая часть, в которой задаются начертания символов для первых 16–ти кодов таблицы знакогенератора, представляет собой перепрограммируемое ОЗУ(CGRAM). Имеется возможность задать начертание 8 символов, соответствующих кодам 0(8), 1(9), 2(10) ... 7(15). Для каждого из восьми перепрограммируемых символов в CGRAM отводится по 8 ячеек памяти, каждая из которых соответствует одной строке точек в изображении символа. Таким образом, перепрограммируемая часть знакогенератора содержит 64 байта памяти (8x8). Пример кодирования CGRAM для одной буквы приведен на рис. 14.

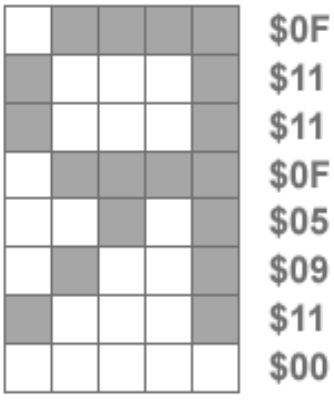


Рис. 14: Пример кодирования символа

На рис. 15 и рис. 17 приведены две возможные схемы сопряжения ЖКИ и AVR–микроконтроллера. В первой схеме (**рис. 5**) ЖКИ подключается к AVR как блок внешней памяти данных.

Узел формирования стробирующего сигнала E здесь выполнен так же, как и на фирменных платах STK200, STK300 и использует сигнал записи WR AVR–микроконтроллера и старший разряд адреса A15. Таким образом, информация

на индикатор поступает при выполнении AVR–микроконтроллером команды записи данных во внешнюю память с любым адресом, имеющим A15=1.

На рис. 16 представлена временная диаграмма записи данных во внешнюю память AVR–микроконтроллера.

Сравнивая ее с диаграммой сигналов записи данных в контроллер ЖКИ (рис. 12), не трудно заметить, что сигналы E и WR имеют разную полярность. Различаются также временные соотношения этих сигналов с сигналами на шине данных. Поэтому узел формирования сигнала E на рис. 15 имеет дополнительный инвертор и дифференцирующую цепочку, укорачивающую импульс записи.

В качестве сигнала выбора регистра RS в рассматриваемой схеме используется еще один разряд адреса – A14. Таким образом, операции записи в контроллер ЖКИ команд или данных отличаются между собой только адресом. Например, данные можно записывать по адресу $C000, а команды – по адресу $8000.

Вывод R/W ЖКИ в предлагаемой схеме непосредственно присоединен к линии «земля», тем самым задается единственно возможное направление передачи информации от AVR–микроконтроллера на ЖКИ. При этом потеря возможности опрашивать BUSY–флаг и считывать данные из блоков памяти контроллера ЖКИ окупается экономией одного управляющего вывода AVR.

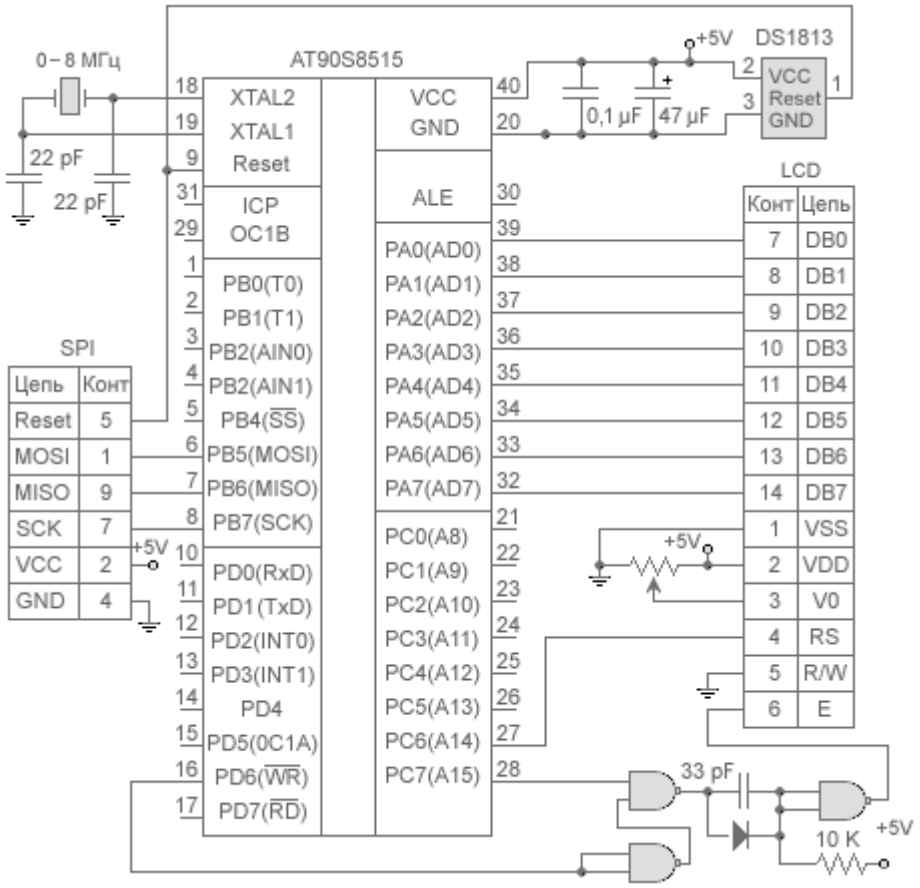


Рис. 15. Включение ЖКИ как блока внешней памяти

На рис. 17 изображена схема подключения ЖКИ, которую можно использовать для AVR–микроконтроллеров, не имеющих возможности подключения

внешней памяти данных. В этой схеме управляющие сигналы E и RS формируются программно на обычных линиях ввода/вывода AVR. В приведенном примере шина данных состоит из 4 разрядов. Каждый байт данных при этом, как упоминалось выше, передается за две последовательные посылки, начиная со старшей тетрады.

Рис. 16. Запись данных во внешнюю память AVR–микроконтроллера

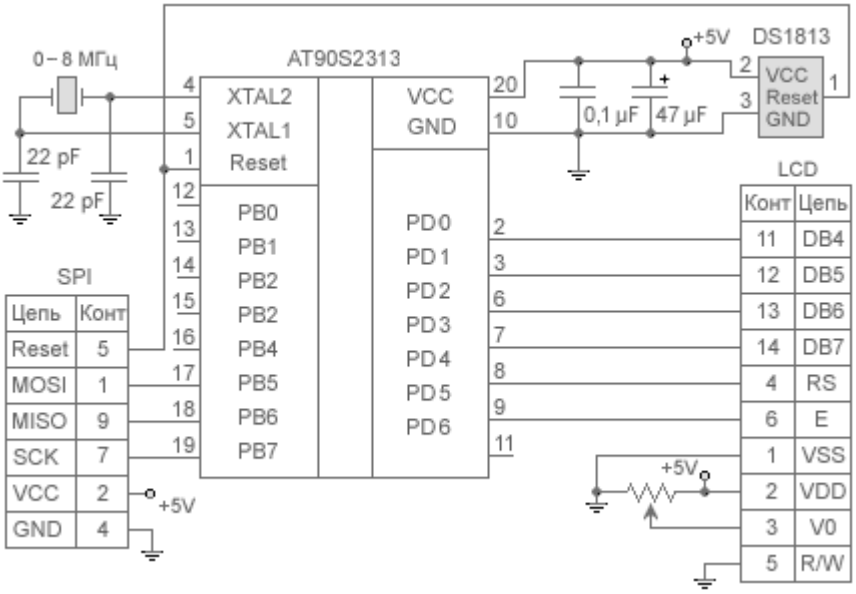


Рис. 17. Подключение ЖКИ при помощи 6 цифровых выводов

Простейшими составными «кирпичиками» драйвера ЖКИ могут быть подпрограммы, представленные в файле Lab2.asm:

LCD_SETUP – подпрограмма инициализации и установки LCD модуля;

LCD_HELLO – подпрограмма вывода на LCD модуль сообщения «HELLO»;

LCD_CMD – подпрограмма вывода в LCD модуль команды;

LCD_DATA – подпрограмма вывода в LCD модуль данных;

LCD_CURSOR – подпрограмма установки курсора в LCD.

Используется 4 битовая схема подключения LCD модуля к микроконтроллеру. Подпрограмма «WAIT_2M» осуществляет задержку 2.3 мсек с тактовой частотой микроконтроллера 1 МГц;

В файле Lab_2.prj используется строка с LCD модулем:

X[inst_name] LCD(chars lines oscil_freq) RS RW E D7 D6 D5 D4 D3 D2 D1 D0

В следуэщей редакции:

Xdisp LCD(24 2 250K) PD2 PB0 PD3 PD7 PD6 PD5 PD4 nc3 nc2 nc1 nc0, где Disp – имя LCD модуля;

nc3 nc2 nc1 nc0 – выводы D3 D2 D1 D0 LCD модуля не используются (применена схема с 4 битовым подключением к микроконтроллеру)

24 2 250K – параметры LCD модуля (число символов, число строк, частота)

PD2 – RS PB0 – RW PD3 – E/

Задание к лабораторной работе

Изучить программу Lab2.asm и файл проекта Lab_2.prj. Отладить программу в среде VMLab, подключив необходимую периферию к микроконтроллеру AVR – интерфейс UART и LCD модуль (дисплей). На рис.18 представлены результаты моделирования программы в среде VMLab.

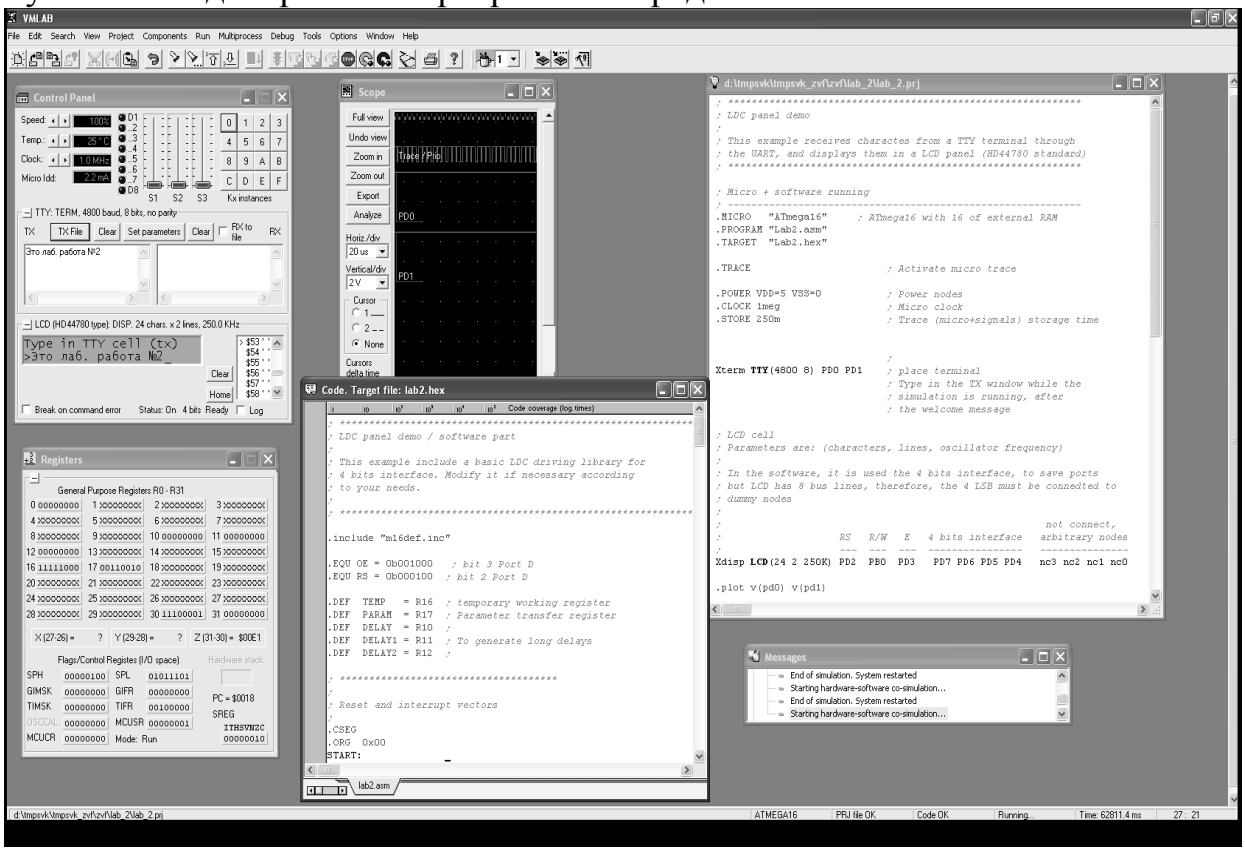


Рис. 18. Результаты моделирования программы Lab2.asm в среде VMLab.

Файл Lab_2.prj

```
.MICRO   "ATmega16"     ; ATmega16 with 16 of external RAM
.PROGRAM "Lab2.asm"
.TARGET  "Lab2.hex"
.TRACE                  ; Activate micro trace
.POWER VDD=5 VSS=0      ; Power nodes
.CLOCK 1meg             ; Micro clock
.STORE 250m             ; Trace (micro+signals) storage time
Xterm TTY(4800 8) PD0 PD1    ; place terminal
Xdisp LCD(24 2 250K) PD2  PB0  PD3   PD7 PD6 PD5 PD4   nc3 nc2 nc1 nc0
```

```
        .plot v(pd0) v(pd1)
```

Файл Lab2.asm

```
.include "m16def.inc"
.EQU OE = 0b001000   ; bit 3 Port D
.EQU RS = 0b0001004  ; bit 2 Port D
.DEF  TEMP   = R16  ; temporary working register
.DEF  PARAM  = R17  ; Parameter transfer register
.DEF  DELAY  = R10  ;
.DEF  DELAY1 = R11  ; To generate long delays
.DEF  DELAY2 = R12  ;
; Reset and interrupt vectors
.CSEG
.ORG  0x00
START:
  rjmp  RESET  ; Reset vector
        .org URXCaddr
        rjmp uart_rx
        .org UDREaddr
        reti
        .org UTXCaddr
        reti
; Device initialization
RESET:
  ldi temp, high(RAMEND)  ; инициализация памяти стека
        out SPH, temp
        ldi temp, low(RAMEND)
        out SPL, temp          ; завершение инициализации памяти стека
  ldi TEMP, 0x17   ; 00010111, setting bits for OC1, PB2 and PB4, PB0
  out DDRB, TEMP   ; set Port B direction as above
  ldi TEMP, 0x28   ; 00101000, PB5 pullup, PB4 lo, PB3 pullup, PB2 lo, OC1 low,
  out PORTB, TEMP
  out PORTB, TEMP  ; set Port B
  ldi TEMP, 0xFC   ; All outputs except bits 0,1 (UART)
  out DDRD, TEMP   ;
  rcall LCD_SETUP; ; Initialize LCD and send
  rcall LCD_HELLO; ; a welcome message
  ldi TEMP, 12     ; Set UART 4800 bauds for 1 MHz
  out UBRR, TEMP   ;
  ldi TEMP, 0xF8   ; Activate all interrupts and TX/RCV
  out UCR, TEMP   ; and go !
  sei
FOREVER:          ; Endless loop. RX interrupt
```

```
   nop            ; will do the job
   nop            ;
   rjmp FOREVER;    ;
UART_RX:          ; Read UART chararter and
   in PARAM, UDR    ; send it to the LCD
   rcall LCD_DATA   ;
   reti
;  LCD Driving Library example
; Display setup for 4 bits interface
LCD_SETUP:
   ldi TEMP, 10          ; Wait about 20msec after powerup
SET1:                    ; LCD is a quite slow,
   rcall WAIT_2M         ; mind delays !
   dec TEMP              ;
   brne SET1             ;
   ldi TEMP, 0b00101000    ; Set 4 bit interface (but we are
   out PORTD, TEMP         ; still in 8 bits!)
   nop
   nop                   ; Data write cycle must be > 1 ms
   cbi PORTD, 3          ; OE low to clock in data
   rcall WAIT_2M         ;
;     *** !! From now on, interface is 4 bits !! ***
   ldi PARAM, 0b00101000   ; Send again to catch the bit N
   rcall LCD_CMD           ; Display is 2 lines, so N = 1
   ldi PARAM, 0b00001000   ; Display off, cursor off, and blink off
   rcall LCD_CMD
   ldi PARAM, 0b00000001   ; Display clear
   rcall LCD_CMD
   rcall WAIT_2M           ; Wait 2 sec after clear
   ldi PARAM, 0b00000110   ; Increment RAM, dont shift display
   rcall LCD_CMD
   ldi PARAM, 0b00001110   ; Display on, cursor on / blink off
   rcall LCD_CMD
   ret
; 2.3 ms delay with 1 MHz clock.
WAIT_2M:
   push TEMP
   ldi TEMP, 3
   mov DELAY1, TEMP
W2_1:
   clr DELAY2
   dec DELAY2            ; Start at 0xFF
W2_2:
   dec DELAY2
```

```
   brne W2_2
   dec DELAY1
   brne W2_1
   pop TEMP
   ret
;  Display a welcome message. Mesage comes at the end
LCD_HELLO:
   ldi ZH, HIGH(2*TXT)
   ldi ZL, LOW(2*TXT)
L3A:
   lpm               ; From prog mem into R0
   mov PARAM, R0
   tst PARAM
   brne L3B
   rjmp L4
L3B:
   cpi PARAM, 0x0D      ; Look for CR in message (new line)
   brne L3C
   ldi PARAM, 0xC0      ; Cursor to position 0x40, start of 2nd line
   rcall LCD_CMD        ; Sends command
   rjmp L3D
L3C:
   rcall LCD_DATA
L3D:
   adiw ZL, 1           ; Increment pointer. This neat instruction does
   rjmp L3A             ; w16 bit addition to pointer ZHI:ZLO.
L4:
   ret               ; Done
TXT:
   .DB   "Type in TTY cell (tx)  ",0x0D  ; Must contain an even
   .DB   ">",0x00                        ; number of bytes per line
;  Sends a control function to the display (comes in PARAM)
LCD_CMD:
   push PARAM
   mov TEMP, PARAM
   andi PARAM, 0xF0     ; Mask off lower 4 bits
   sbr PARAM, 8         ; OE bit high (bit 3)
   out PORTD, PARAM     ; Send upper 4 bits to display
   nop                  ; Brief delay to give reasonable OE
   nop                  ; pulse width
   cbi PORTD,3          ; OE goes low to clock in data
   mov PARAM, TEMP      ; Data back
   swap PARAM           ; Lower 4 bits
   andi PARAM, 0xF0
```

```
   sbr PARAM, 8        ; OE high (bit 3, port D)
   out PORTD, PARAM    ; Write lower 4 bits to LCD
   nop
   nop
   cbi PORTD,3         ; OE clock low
   ldi PARAM, 100      ; 50 usec approx
DEL0:
   dec PARAM
   brne DEL0
   pop PARAM
   ret
; Sends an ASCII character to the display (comes in PARAM)
LCD_DATA:
   push PARAM
   push TEMP
   mov TEMP, PARAM
   andi PARAM, 0xF0    ; Mask off lower 4 bits
   sbr PARAM, OE       ; OE bit high
   sbr PARAM, RS       ; Data/command bit high
   out PORTD, PARAM    ; Write upper 4 bits to display
   nop                 ; Brief delay
   nop
   cbi PORTD, 3        ; OE low to clock data
   swap TEMP           ; Lower 4 bits
   andi TEMP, 0xF0     ; Masked off
   sbr TEMP, OE        ; OE high
   sbr TEMP, RS
   out PORTD, TEMP     ; Lower 4 bits to LCD
   nop
   nop
   cbi PORTD, 3        ; OE low to clock data
   ldi TEMP, 100
   mov DELAY, TEMP     ; 50 usec
DEL1:
   dec DELAY
   brne DEL1
   pop TEMP
   pop PARAM
   ret
;  Moves the LCD display cursor to address specified in PARAM
LCD_CURSOR:
   ori   PARAM, 0x80   ; MSB specifies address command
   rcall LCD_CMD
   ret
```

Порядок выполнения лабораторной работы

Необходимо выполнить все необходимые операции с проектом Lab_2.prj и программой Lab2.asm в среде VMLab как и в лабораторной работе №1.

Содержание отчета

Отчет по проделанной лабораторной работе должен содержать:

- структурную схему алгоритма программы на лабораторную работу;
- схему электрическую принципиальную разработанного устройства;
- текст программы на языке AVR Ассемблере (распечатка файла *asm);
- распечатка файла проекта (*.prj);
- результаты моделирования (окно VMLab с запущенной программой и результатами вычислений).

Контрольные вопросы

1. Структура микроконтроллера ATmega16.
2. Состав файла m16def.inc.
3. Регистры управления UART микроконтроллера ATmega16.
4. Порты ввода/вывода микроконтроллера ATmega16.
5. Порты ввода/вывода микроконтроллера ATmega16.
6. Подключение LCD модуля к микроконтроллеру AVR ATmega16.
7. Управление процессом вывода информации из AVR микроконтроллера ATmega16 в LCD модуль (дисплей).
8. Назначение и состав файла m16def.inc.

Лабораторная работа № 3
ПРОГРАММА УПРАВЛЕНИЯ ТАЙМЕР–СЧЕТЧИКОМ В РЕЖИМЕ ШИРОТНО–ИМПУЛЬСНОЙ МОДУЛЯЦИИ

Цель работы: Изучение проектирования таймера/счетчика в контроллере AVR на примере программы генерации синусоидального сигнала в режиме широтно–импульсной модуляции.

Краткие теоретические сведения

Таймер/счетчик 0

Таймер-счетчик 0 – модуль многофункционального одноканального 8–разрядного таймера–счетчика. Функциональная схема таймера–счетчика представлена на рис. 19.

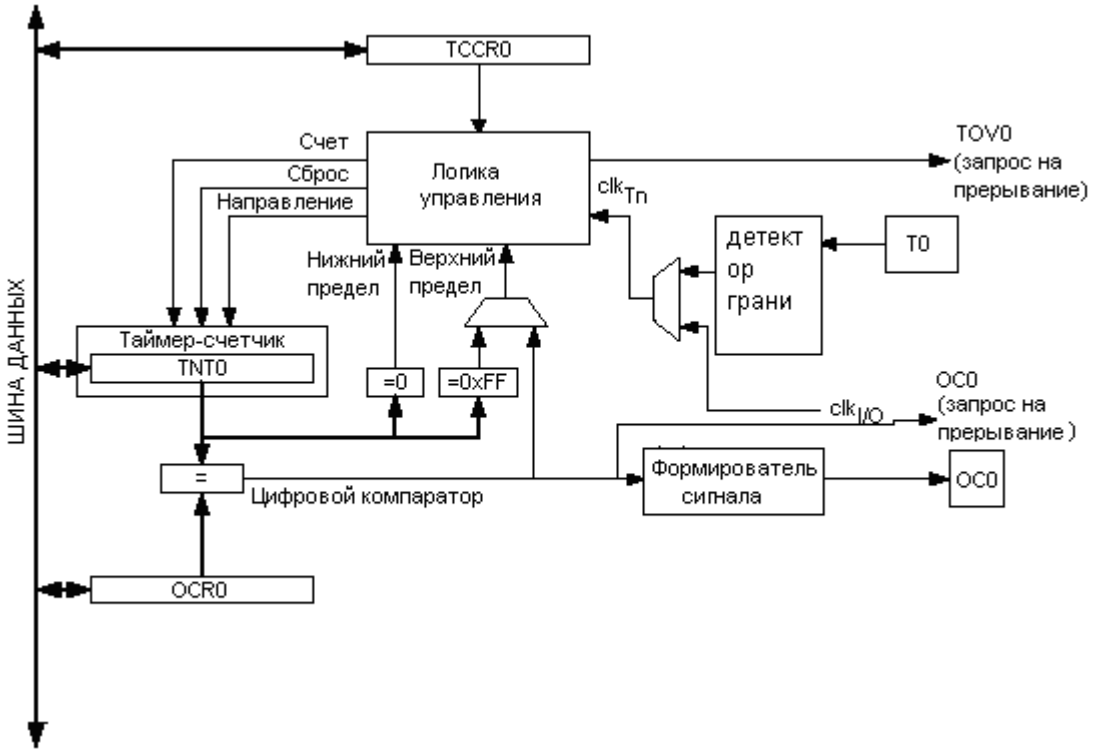


Рис. 19. Функциональная схема 8–разр. таймера–счетчика 0

Описание регистров 8–разрядного таймера–счетчика 0

Регистр управления таймером–счетчиком 0 – TCCR0

В таблице 10 представлена структура регистра, режимы управления и начальные значения разрядов регистра TCCR0.

Табл. 10. Регистр TCCR0

Разряд	7	6	5	4	3	2	1	0
	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
Чт./зап.	Чт.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.
Исх.зн.	0	0	0	0	0	0	0	0

Разряд 7 – FOC0: Принудительная установка результата сравнения

Строб FOC0 не генерирует каких–либо прерываний, а также не вызывает сброс таймера в режиме CTC, где регистр OCR0 задает верхний предел счета. Бит FOC0 всегда считывается как 0.

Разряд 6, 3 – WGM01:0: Режим работы таймера–счетчика 0

Данные биты представлены в таблице 11, определяют: алгоритм счета счетчика, источник, который задает верхний предел счета и тип генерируемых прямоугольных импульсов.

Таблица 11. Описание бит, задающих режим работы таймера–счетчика 0

Номер реж.	WGM01	WGM00	Режим работы тайм/сч. 0	Верхний предел счета	Условие обновления регистра OCR0	Условие уст. флага TOV0
0	0	0	Нормальный	0xFF	Сразу после записи в регистр	Дост–ние макс–го зн. (0xFF)
1	0	1	ШИМ с фаз. кор–й	0xFF	Достижение верхнего предела счета	Дост–ние мин. зн. (0x00)
2	1	0	Сброс при совп–нии	OCR0	Сразу после записи в регистр	Дост–ние макс. зн. (0xFF)
3	1	1	Быстрый ШИМ	0xFF	Достижение верхнего предела счета	Дост–ние макс. зн. (0xFF)

Разряд 5:4 – COM01, COM00: Режим формирования выходного сигнала

Данные биты представлены в таблице 12, определяют алгоритм изменения сигнала на выводе OC0.

Таблица 12. Режимы формирования выходного сигнала в режимах работы таймера 0 без ШИМ

COM01	COM00	Описание
0	0	Функция обычного порта ввода–вывода. OC0 отключен.
0	1	Переключение (инвертирование) OC0 при каждом совпадении
1	0	Сброс OC0 при каждом совпадении
1	1	Установка OC0 при каждом совпадении

В таблице 13 приведено назначение бит COM01, COM00 для режима работы таймера–счетчика 0 с быстрой ШИМ (WGM01:0).

Таблица 13. Режимы формирования выходного сигнала в режиме таймера 0 с быстрым ШИМ(1)

COM01	COM00	Описание
0	0	Функция обычного порта ввода–вывода. OC0 отключен.
0	1	Зарезервировано
1	0	Сброс OC0 при совпадении, установка по достижении верхнего предела (0xFF)
1	1	Установка OC0 при совпадении, сброс по достижении верхнего предела (0xFF)

В таблице 14 приведено действие бит COM01, COM00 для режима ШИМ с фазовой коррекцией, заданного с помощью бит WGM01, WGM00.

Таблица 14. Режимы выходного сигнала в режиме ШИМ с фазовой коррекцией(1)

COM01	COM00	Описание
0	0	Функция обычного порта ввода–вывода. OC0 отключен.
0	1	Зарезервировано
1	0	Сброс OC0 при совпадении во время прямого счета. Установка OC0 при совпадении во время обратного счета.
1	1	Установка OC0 при совпадении во время прямого счета. Сброс OC0 при совпадении во время обратного счета.

Разряд 2:0 – CS02:0: Настройка частоты синхронизации таймера

С помощью трех настроечных бит имеется возможность выбрать различные тактовые частоты, кратные исходной частоте синхронизации (см. табл. 15).

Таблица 15. Выбор частоты синхронизации таймера 0

CS02	CS01	CS00	Описание
0	0	0	Нет синхронизации. Таймер–счетчик 0 оставлен.
0	0	1	clkT0S/1 (без предделения)
0	1	0	clkT0S/8 (с предделением)
0	1	1	clkT0S/32 (с предделением)
1	0	0	clkT0S/64 (с предделением)
1	0	1	clkT0S/128 (с предделением)
1	1	0	clkT0S/256 (с предделением)
1	1	1	clkT0S/1024 (с предделением)

Регистр таймера–счетчика, разряды которого представлены в таблице 16, характеризуется двунаправленностью доступа к 8–разрядному счетчику тай-

мера 0. Запись в регистр TCNT0 блокирует отработку возникающего совпадения на следующем после записи такте синхронизации таймера. Изменение содержимого счетчика (TCNT0) во время счета связано с риском потери результата сравнения между TCNT0 и регистром OCR0.

Таблица 16. Регистр таймера–счетчика – TCNT0

Разряд	7	6	5	4	3	2	1	0
TCNT0	TCNT0[7:0]							
Чт./зап.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.
Исх. знач.	0	0	0	0	0	0	0	0

Регистр порога сравнения, разряды которого представлены в таблице 17, содержит 8–разр. значение, которое непрерывно сравнивается цифровым компаратором со значением 8–разр. счетчика (TCNT0). Факт совпадения значений может использоваться для генерации прерывания по выполнению условия сравнения или для генерации прямоугольных импульсов на выводе OC0.

Таблица 17. Регистр порога сравнения – OCR0

Разряд	7	6	5	4	3	2	1	0
OCR0	OCR0[7:0]							
Чт./зап.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.
Исх. зн.	0	0	0	0	0	0	0	0

Таймер/счетчик 1

16–ти разрядный таймер–счетчик 1 предназначен для точного задания временных интервалов, генерации прямоугольных импульсов и измерения временных характеристик импульсных сигналов.

На рисунках и в тексте описания индекс “n” заменяет номер таймера–счетчика , а “x” заменяет наименование канала сравнения (A, или B). Однако при программировании необходимо использовать фактические номера и наименования.

В таблице 18 представлены разряды регистра A управления таймером–счетчиком 1 – TCCR1A.

В таблице 19 приведены режимы управления сигналов OCnA/OCnB.

В таблице 20 представлены разряды регистра B управления таймером–счетчиком 1 – TCCR1B.

Функциональная схема 16–разр. таймера–счетчика показана на рисунке 20.

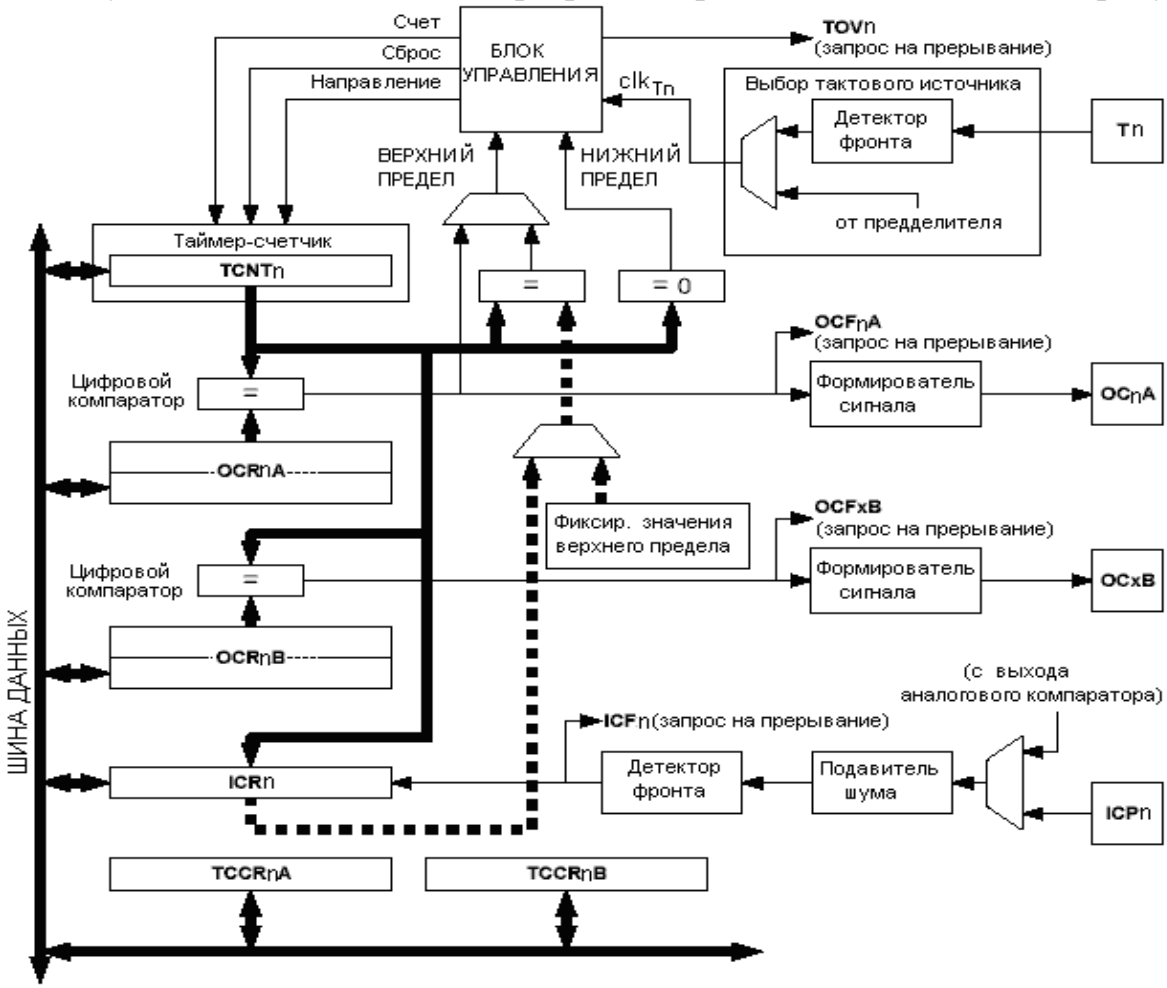


Рис. 20. Функциональная схема 16–разр. таймера–счетчика

Таблица 18. Регистр А управления таймером–счетчиком 1 – TCCR1A

Bit	7	6	5	4	3	2	1	0
	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10
Чт./зап.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.
Исх. зн.	0	0	0	0	0	0	0	0

Биты 7:6 – COM1A1:0 Режим формирования выходного сигнала канала А
Биты 5:4 – COM1B1:0 Режим формирования выходного сигнала канала В

Таблица 19. Управление режимами сигналов OCnA/OCnB

COMnA1/ COMnB1	COMnA0/ COMnB0	Описание
0	0	Нормальная работа порта, сигналы OCnA/OCnB отключены.
0	1	Переключение (инвертирование) OCnA/OCnB при совпадении.

1	0	Сброс OCnA/OCnB при совпадении (установка лог. 0).
1	1	Установка OCnA/OCnB при совпадении (установка лог. 1).

Биты 3:2 – FOC1A:FOC1B**:** Режим формирования силы выходного сигнала.

Разряд 1:0 – WGMn1:0: Режим работы таймера–счетчика

Таблица 20. Регистр B управления таймером–счетчиком 1 – TCCR1B

Разряд	7	6	5	4	3	2	1	0
	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10
Чт./зап.	Чт./Зп.	Чт./Зп.	Чт.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.
Исх. зн.	0	0	0	0	0	0	0	0

Разряд 7 **– ICNC1**: Подавитель шума на входе захвата (задержка сигнала с входа захвата на 4 такта)

Разряд 6 – ICES1**:** Выбор детектируемого фронта на входе захвата

Разряд 5 – Зарезервированный бит

Разряд 4:3 – WGM1 3:2: Режим работы таймера–счетчика

Разряд 2:0 **– CS12:0**: Выбор тактового источника

Данные три бита, представленные в таблице 21, позволяют выбрать тактовый источник для таймера–счетчика.

Таблица 21. Описание бит выбора тактового источника

S12	S11	S10	Описание
			Нет синхронизации. Таймер–счетчик остановлен.
			clkI/O/1 (без предделения)
			clkI/O /8 (с предделением)
			clkI/O/64 (с предделением)
			clkI/O/256 (с предделением)
			clkI/O/1024 (с предделением)
			Внешний тактовый источник с выв. T1. Синхронизация по падающему фронту.
			Внешний тактовый источник с выв. T1. Синхронизация по нарастающему фронту.

Если для тактирования таймера выбран внешний вывод T1, то данная функция за ним сохраняется, даже при его настройке на вывод. Данная функция позволяет программно управлять счетом.

В таблице 22 представлены разряды таймер–счетчиков 1 – TCNT1H и TCNT1L.

Таблица 22. Таймер–счетчик 1 – TCNT1H и TCNT1L

Разряд	7	6	5	4	3	2	1	0	
	TCNT1[15:8]								TCNT1H
	TCNT1[7:0]								TCNT1L
Чт./зап.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	
Исх. зн.	0	0	0	0	0	0	0	0	

Две ячейки в области ввода–вывода (TCNT1H и TCNT1L, вместе TCNT1) дают полный доступ, как на чтение, так и на запись к 16–разрядному счетчику. В целях гарантирования одновременности чтения и записи старшего и младшего байтов этих регистров, доступ организован с использованием 8–разрядного временного регистра старшего байта (TEMP). Временный регистр является общим для всех 16–разрядных регистров таймера.

Изменение содержимого счетчика TCNT1 во время его работы (счета) связано с риском возникновения совпадения между TCNT1 и одним из регистров OCR1x. Запись в регистр TCNT1 блокирует отработку совпадения, которое возникнет на следующем такте, для всех блоков сравнения.

В таблицах 23 и 24 представлены разряды регистров сравнения 1A – OCR1AH, OCR1AL и 1B – OCR1BH и OCR1BL

Таблица 23. Регистр сравнения 1 A – OCR1AH и OCR1AL

Разряд	7	6	5	4	3	2	1	0	
	OCR1A [15:8]								OCR1AH
	OCR1A [7:0]								OCR1AL
Чт./зап.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	
Исх. зн.	0	0	0	0	0	0	0	0	

Таблица 24. Регистр сравнения 1B – OCR1BH и OCR1BL

Разряд	7	6	5	4	3	2	1	0	
	OCR1B [15:8]								OCR1BH
	OCR1B [7:0]								OCR1BL
Чт./зап.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	
Исх. зн.	0	0	0	0	0	0	0	0	

В регистрах сравнения хранится 16–разр. значение, которое непрерывно сравнивается со значением счетчика (TCNTn). Возникающее совпадение может использоваться для генерации прерывания по результату сравнения и генерации прямоугольных импульсов на выводе OCnx.

Регистры сравнения являются 16–разрядными, поэтому, одновременность записи младшего и старшего байтов достигнута за счет использования 8–разр. временного регистра старшего байта (TEMP). Временный регистр является общим для всех 16–разрядных регистров таймера.

Регистры захвата, представленные в таблице 25, обновляются содержимым соответствующего счетчика (TCNTn) при каждом определении условия захвата на входе ICPn (или альтернативно на выходе аналогового компаратора для таймера–счетчика 1).

Таблица 25. Регистр захвата 1 – ICR1H и ICR1L

Разряд	7	6	5	4	3	2	1	0	
	ICR1H [15:8]								ICR1H
	ICR1L [7:0]								ICR1L
Чт./зап.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	Зп./Чт.	
Исх. зн.	0	0	0	0	0	0	0	0	

Регистры захвата альтернативно могут использоваться для задания верхнего предела счета.

Регистры захвата также являются 16–разрядными, поэтому, одновременность записи младшего и старшего байтов достигнута за счет использования 8–разр. временного регистра старшего байта (TEMP). Временный регистр является общим для всех 16–разрядных регистров таймера.

В таблице 26 представлены разряды регистра маски прерываний таймера–счетчика – TIMSK

Таблица 26. Регистр маски прерываний таймера–счетчика – TIMSK

Разряд	7	6	5	4	3	2	1	0	
	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	TIMSK
Чт//зап.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. зн.	0	0	0	0	0	0	0	0	

Прим.: *Данный регистр биты управления прерываниями для нескольких таймер–счетчиков, но в данном разделе детализированы только биты таймера 1. Описание остальных бит необходимо искать при описании соответствующих таймеров.*

Разряд 5 **– TICIE1**: Разрешение прерывания по захвату состояния таймера–счетчика 1

Если в данный бит записана лог. 1, а также установлен флаг I в регистре статуса (активно общее разрешение прерываний), то разрешается прерывание по захвату состояния таймера–счетчика 1. Если устанавливается флаг в регистре TIFR, программа переходит на соответствующий вектор прерывания .

Разряд 4 **– OCIE1A**: Разрешение прерывания по результату сравнения канала А таймера–счетчика 1

Если в данный бит записана лог. 1 и установлен флаг I в регистре статуса, то разрешается работа прерывания по результату сравнения канала А. Если устанавливается флаг OCF1A в регистре TIFR, то программа переходит на соответствующий вектор прерываний.

Разряд 3 **– OCIE1B**: Разрешение прерывания по результату сравнения канала В таймера–счетчика 1

Действие аналогично предыдущему, но в отношении канала сравнения В.

Разряд 2 – TOIE1: Разрешение прерывания при переполнении таймера–счетчика 1

Если в данный бит записана лог. 1 и установлен флаг I в регистре статуса, то разрешается прерывание по переполнению таймера–счетчика 1. После этого, установка флага TOV1 в регистре TIFR приведет к переходу на соответствующий вектор прерывания.

В таблице 27 представлен регистр флагов прерываний таймеров–счетчиков TIFR.

Таблица 27. Регистр флагов прерываний таймеров–счетчиков – TIFR

Разряд	7	6	5	4	3	2	1	0	
	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOF0	TIFR
Чт./зап.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	Чт./Зп.	
Исх. зн.	0	0	0	0	0	0	0	0	

Прим.: *Биты данного регистра относятся к нескольким таймерам, но в данном параграфе рассматриваются биты только одного таймера. Описание остальных бит необходимо смотреть в соответствующих разделах.*

Разряд 5 – ICF1: Флаг захвата состояния таймера–счетчика 1

Флаг устанавливается, если на входе ICP1 определяется условие захвата. Если регистр захвата ICR1 выбран с помощью бит WGMn3:0 в качестве источника верхнего предела счета, флаг ICF1 устанавливается по достижении верхнего предела счета.

ICF1 автоматически сбрасывается при переходе на вектор прерывания по захвату состояния таймера–счетчика. Альтернативно флаг ICF1 можно сбрасывать путем записи в него лог. 1.

Разряд 4 – OCF1A: Флаг результата сравнения канала A таймера–счетчика 1

Данный флаг устанавливается следующим тактом после совпадения значения TCNT1 с регистром A порога сравнения (OCR1A).

Обратите внимание, что строб принудительной установки результата сравнения (FOC1A) не устанавливает флаг OCF1A. Флаг OCF1A автоматически сбрасывается при переходе на соответствующий вектор прерывания.

Альтернативно, флаг OCF1A сбрасывается путем записи в него лог. 1.

Разряд 3 – OCF1B: Флаг результата сравнения канала B таймера–счетчика 1

Данный флаг действует аналогично предыдущему, но в отношении канала сравнения B.

Разряд 2 – TOV1: Флаг переполнения таймера–счетчика 1

Установка данного флага зависит от значений бит WGMn3:0. В нормальном режиме и режиме CTC флаг TOV1 устанавливается при переполнении таймера–счетчика. См. табл. 61 для изучения поведения флага TOV1 при задании других значений WGMn3:0. Флаг TOV1 автоматически сбрасывается при переходе на вектор прерывания по переполнению таймера–счетчика 1. Альтернативно флаг TOV1 сбрасывается путем записи в него лог. 1.

Задание к лабораторной работе

Изучить файл проекта Lab_3.prj и файл программы Lab3.asm на языке Ассемблер. Отмоделировать программу в среле VMLab. Результаты моделирования представлены на рис. 21.

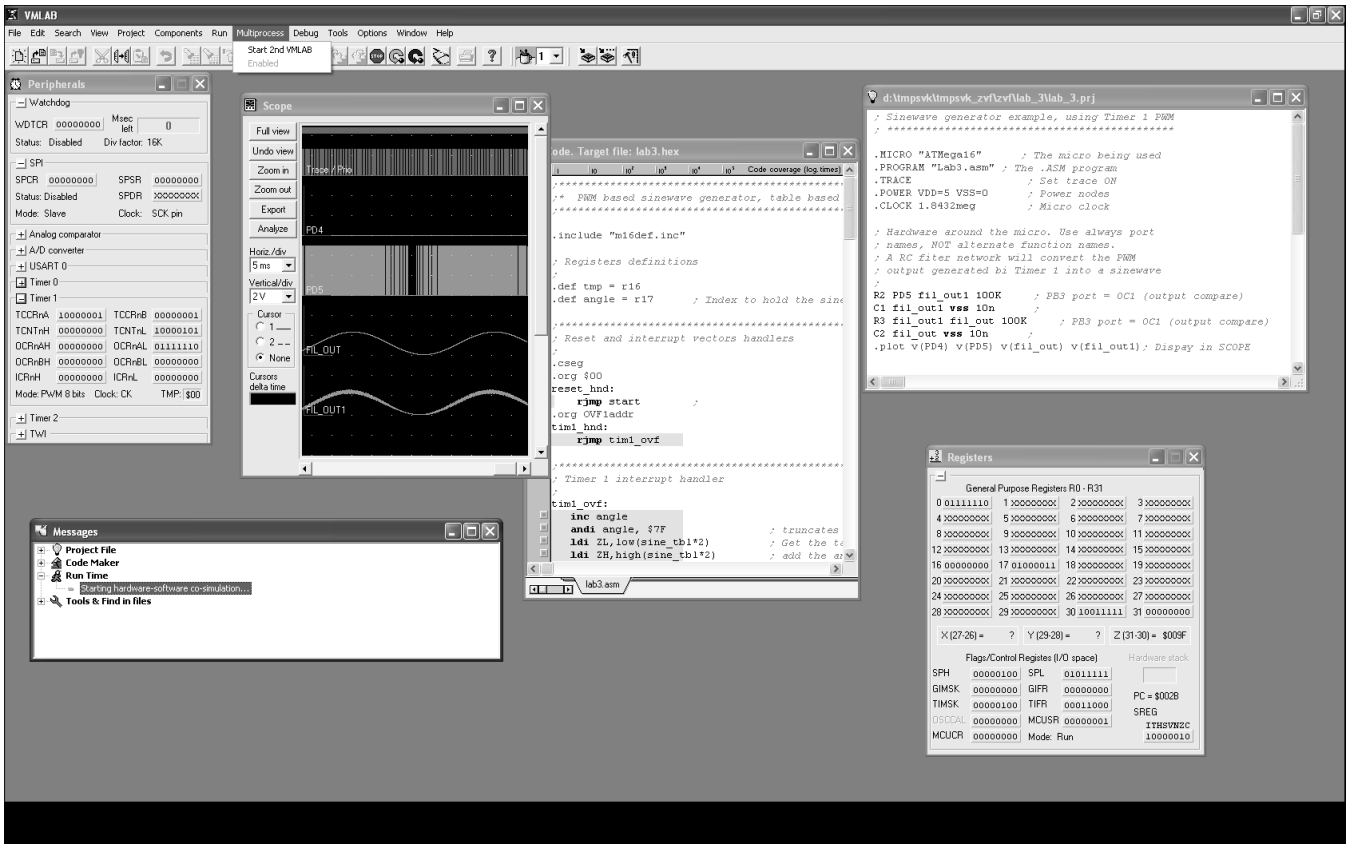


Рис.21. Результаты моделирования программы LAB3.asm

Файл Lab_3.prj

```
; Sinewave generator example, using Timer 1 PWM
.MICRO "ATmega16"      ; The micro being used
.PROGRAM "Lab3.asm" ; The .ASM program
.TRACE                 ; Set trace ON
.POWER VDD=5 VSS=0     ; Power nodes
```

```
.CLOCK 1.8432meg        ; Micro clock

R2 PD5 fil_out1 100K     ; PB3 port = OC1 (output compare)
C1 fil_out1 vss 10n      ;
R3 fil_out1 fil_out 100K     ; PB3 port = OC1 (output compare)
C2 fil_out vss 10n      ;
.plot v(PD4) v(PD5) v(fil_out) v(fil_out1); Dispay in SCOPE
```

Файл Lab3.asm

```
;*****************************************************************
;*  PWM based sinewave generator, table based
;*****************************************************************

.include "m16def.inc"

; Registers definitions
;
.def tmp = r16
.def angle = r17      ; Index to hold the sine phase angle (0 to 127)

;*****************************************************************
; Reset and interrupt vectors handlers
;
.cseg
.org $00
reset_hnd:
   rjmp start       ;
.org OVF1addr
tim1_hnd:
   rjmp tim1_ovf

;*****************************************************************
; Timer 1 interrupt handler
;
tim1_ovf:
  inc angle
  andi angle, $7F              ; truncates to 7 bits (0 – 127)
  ldi ZL,low(sine_tbl*2)        ; Get the table address and
  ldi ZH,high(sine_tbl*2)       ; add the angle phase. Result
  add ZL,angle                 ; will be in R0 after calling 'lpm'
  clr tmp                     ; lpm uses 16 bits index (Z)
  adc ZH, tmp                  ; Z index is the pointer to the
  lpm                         ; sine_tbl value for the angle phase
```

```
  clr tmp
  out OCR1AH, tmp          ; Reload Timer 1 compare value.
  out OCR1AL,R0            ; Necessary to reload both registers
  reti

;*************************************************************
; Reset handler. Initalizes port and Timer 1, and stay in a endless loop
;
start:
  sbi DDRD, PD4 ; Set pin PD5 as output (is OC1 pin)
  sbi DDRD, PD5
 ldi tmp, high(RAMEND)  ; инициализация памяти стека
 out SPH, tmp
 ldi tmp, low(RAMEND)
 out SPL, tmp          ; завершение инициализации памяти стека

  ldi tmp,(1<<TOIE1)
  out TIMSK,tmp            ; Enable Timer1_ovf interrupt

  ldi tmp,(1<<PWM10)+(1<<COM1A1) ; Set Timer 1 in PWM mode
  out TCCR1A,tmp           ; 8 bit PWM not reverse (Fck/510)
  ldi tmp,(1<<CS10)
  out TCCR1B,tmp           ; prescaler = 1

  clr angle                ; Start with phase = 0

  sei                      ; Enable interrupts

main:
nop
nop
rjmp  main  ; Infinite loop: Timer1 interrupt will handle all

;*************************** SINE   TABLE
**************************************
; Samples table : one period sampled on 128 samples and
; quantized on 7 bit
;
sine_tbl:
.db 0,0
.db 1,1
.db 2,3
.db 4,6
```

```
.db 7,9
.db 10,12
.db 14,16
.db 18,21
.db 23,25
.db 28,31
.db 33,36
.db 39,42
.db 45,48
.db 51,54
.db 57,60
.db 64,67
.db 70,73
.db 76,79
.db 82,85
.db 88,91
.db 94,96
.db 99,102
.db 104,106
.db 109,111
.db 113,115
.db 117,118
.db 120,121
.db 123,124
.db 125,126
.db 126,127
.db 127,127
.db 127,127
.db 127,127
.db 126,126
.db 125,124
.db 123,121
.db 120,118
.db 117,115
.db 113,111
.db 109,106
.db 104,102
.db 99,96
.db 94,91
.db 88,85
.db 82,79
.db 76,73
.db 70,67
.db 64,60
```

```
.db 57,54
.db 51,48
.db 45,42
.db 39,36
.db 33,31
.db 28,25
.db 23,21
.db 18,16
.db 14,12
.db 10,9
.db 7,6
.db 4,3
.db 2,1
.db 1,0
.db 0,0
.db 0,0
```

Порядок выполнения работы

Нужно выполнить все необходимые операции с проектом Lab_3.prj и программой Lab3.asm в среде VMLab как и в лабораторных работах №1,2.

Содержание отчета

Отчет по проделанной лабораторной работе должен содержать:

- структурную схему алгоритма программы на лабораторную работу;
- схему электрическую принципиальную разработанного устройства;
- текст программы на языке AVR Ассемблере (распечатка файла *asm);
- распечатка файла проекта (*.prj);
- результаты моделирования (окно VMLab с запущенной программой и результатами вычислений).

Контрольные вопросы

1. Регистр состояния микроконтроллера ATmega16.
2. Стек и его инициализация.
3. Память микроконтроллера ATmega16.
4. Процессорное ядро микроконтроллеров AVR.
5. Непосредственная адресация данных в микроконтроллерах AVR.
6. Прямая адресация данных в микроконтроллерах AVR.
7. Косвенная адресация данных в микроконтроллерах AVR.
8. Относительная адресация данных в микроконтроллерах AVR.

Лабораторная работа № 4

ПРОГРАММА ВВОДА С КЛАВИАТУРЫ И ВЫВОДА В LCD МОДУЛЬ

Цель работы: Изучение и освоение управления периферийным устройством ввода с клавиатуры, подключенным к AVR микроконтроллеру фирмы Atmel, а также дальнейшее изучение системы команд и системы прерываний.

Задание к лабораторной работе

Изучить программу Lab4.asm и файл проекта Lab_4.prj. Отладить программу в среде VMLab, подключив необходимую периферию к микроконтроллеру AVR – интерфейс клавиатуры и LCD модуль (дисплей).

Кнопки «0» и «1» клавиатуры в «Control Panel» VMLab подключены совместно с резисторами R1 и R2 к микроконтроллеру соответственно к входам PD2 и PD3 таким образом, что при нажатии на них возникают dytiybt прерывания «INT0» и «INT1» соответственно. В файле «Lab5.asm» предусмотрены соответствующие инициализации векторов прерываний и имеются подпрограммы обработки этих прерываний. Кнопка «0» запускает таймер счета, при этом происходит отображение результатов счета на модуле LCD. Кнопка «1» останавливает таймер счета. Результаты моделирования представлены на рис.22.

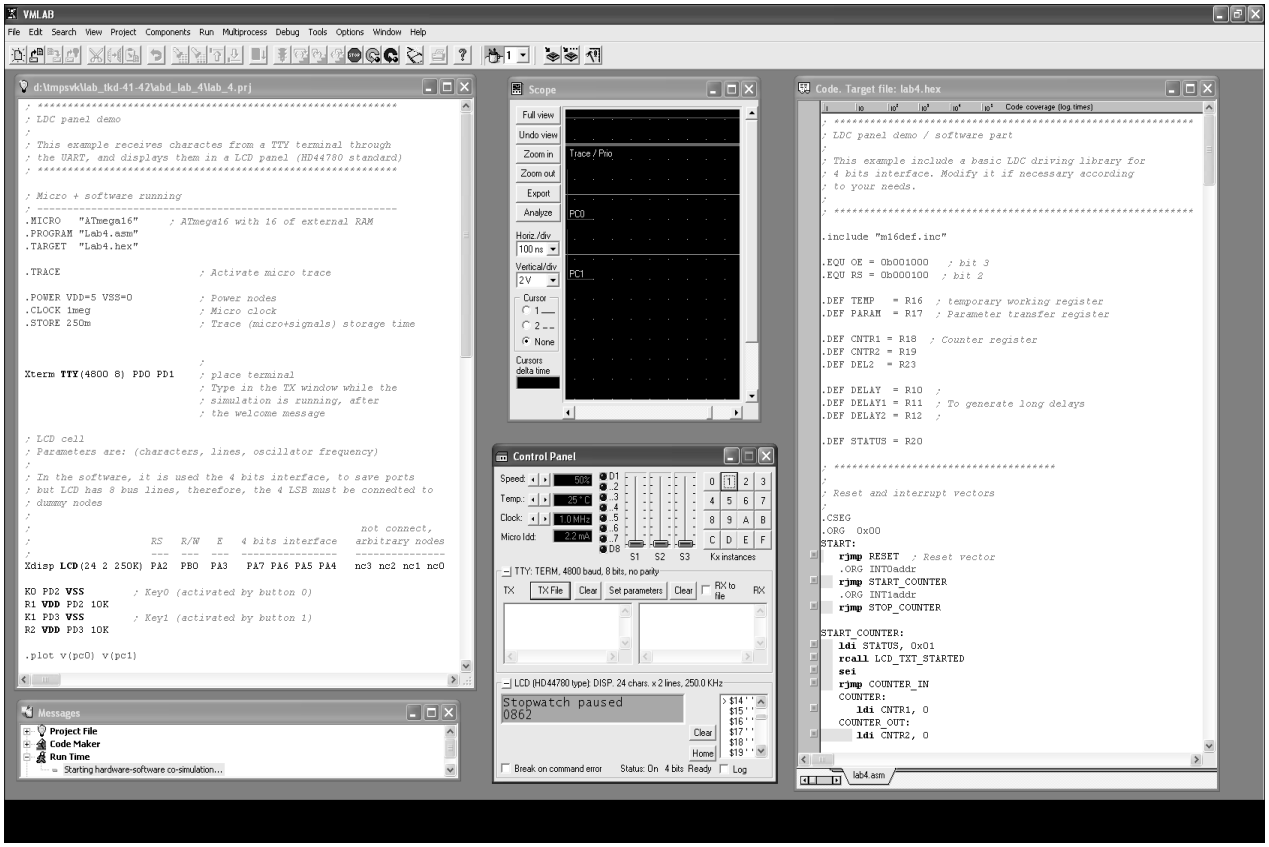


Рис.22. Результаты моделирования программы Lab5.asm.

Файл Lab_4.prj

```
.MICRO   "ATmega16"     ; ATmega16 with 16 of external RAM
.PROGRAM "Lab4.asm"
.TARGET  "Lab4.hex"

.TRACE                  ; Activate micro trace

.POWER VDD=5 VSS=0          ; Power nodes
.CLOCK 1meg                 ; Micro clock
.STORE 250m                 ; Trace (micro+signals) storage time
                        ;
Xterm TTY(4800 8) PD0 PD1   ; place terminal
                            ; Type in the TX window while the
                            ; simulation is running, after
                            ; the welcome message
;                RS   R/W   E   4 bits interface   arbitrary nodes
Xdisp LCD(24 2 250K) PA2  PB0  PA3   PA7 PA6 PA5 PA4   nc3 nc2 nc1 nc0

K0 PD2 VSS       ; Key0 (activated by button 0)
R1 VDD PD2 10K
K1 PD3 VSS       ; Key1 (activated by button 1)
R2 VDD PD3 10K

.plot v(pc0) v(pc1)
```

Файл Lab4.asm

```
.include "m16def.inc"

.EQU OE = 0b001000  ; bit 3
.EQU RS = 0b000100  ; bit 2

.DEF TEMP   = R16  ; temporary working register
.DEF PARAM  = R17  ; Parameter transfer register

.DEF CNTR1 = R18  ; Counter register
.DEF CNTR2 = R19
.DEF DEL2  = R23

.DEF DELAY  = R10   ;
.DEF DELAY1 = R11  ; To generate long delays
.DEF DELAY2 = R12  ;

.DEF STATUS = R20
```

```
; ************************************
.CSEG
.ORG  0x00
START:
  rjmp RESET  ; Reset vector
  .ORG INT0addr
        rjmp START_COUNTER
  .ORG INT1addr
        rjmp STOP_COUNTER

START_COUNTER:
        ldi STATUS, 0x01
        rcall LCD_TXT_STARTED
        sei
        rjmp COUNTER_IN
        COUNTER:
                ldi CNTR1, 0
        COUNTER_OUT:
                ldi CNTR2, 0
        COUNTER_IN:
                rcall LCD_PRINT_CNTRS
                inc CNTR2
                cpi CNTR2, 100
                brne COUNTER_IN
                inc CNTR1
                cpi CNTR1, 100
                brne COUNTER_OUT
                rjmp COUNTER
STOP_COUNTER:
        tst STATUS
        breq CLEAR_COUNTER
        ldi STATUS, 0x00
        rcall LCD_TXT_PAUSED
        sei
        rjmp FOREVER
CLEAR_COUNTER:
        rcall LCD_TXT_STOPPED
        ldi CNTR1, 0
        ldi CNTR2, 0
        rcall LCD_PRINT_CNTRS
        sei
        rjmp FOREVER
```

```
; ****************************************
; Device initialization
; .ORG  0x10
RESET:
  ldi TEMP, HIGH(RAMEND)  ; инициализация памяти стека
        out SPH, TEMP
        ldi TEMP, LOW(RAMEND)
        out SPL, TEMP          ; завершение инициализации памяти стека

  ldi TEMP, 0x0A    ; For INT0, INT1
  out MCUCR, TEMP   ; select falling edge
  ldi TEMP, 0xC0   ;
  out GICR, TEMP  ; Enable INT0, INT1

  ldi TEMP, 0x17   ; 00010111, setting bits for OC1, PB2 and PB4, PB0
  out DDRB, TEMP   ; set Port B direction as above
  ldi TEMP, 0x28   ; 00101000, PB5 pullup, PB4 lo, PB3 pullup, PB2 lo, OC1 low,
  out PORTB, TEMP  ; set Port B
  ldi TEMP, 0xFF   ; All outputs except bits 0,1 (UART)
  out DDRA, TEMP   ;

        ldi STATUS, 0x00
        ldi CNTR1, 0x00
        ldi CNTR2, 0x00

  rcall LCD_SETUP; ; Initialize LCD and send
  rcall LCD_TXT_READY; ; a welcome message

        sei

FOREVER:
        rjmp FOREVER

LCD_PRINT_CNTRS:
        rcall RETURN_CURSOR_4
        mov PARAM, CNTR1
        rcall BIN_TO_TEXT
        mov PARAM, CNTR2
        rcall BIN_TO_TEXT
        ret

BIN_TO_TEXT:
        push PARAM
        ldi XL, 0
```

```
        ldi XH, 0
DEC_1000:
        inc XH
        subi PARAM, 10
        brpl DEC_1000
        dec XH
        tst PARAM
        brmi BIN_TO_TEXT_NOT_10
        inc XH
        ldi XL, 0
        rjmp BIN_TO_TEXT_DONE
BIN_TO_TEXT_NOT_10:
        ldi TEMP, 10
        add PARAM, TEMP
        mov XL, PARAM
BIN_TO_TEXT_DONE:
        ldi TEMP, 0x30
        mov PARAM, XH
        add PARAM, TEMP
        rcall LCD_DATA
        mov PARAM, XL
        add PARAM, TEMP
        rcall LCD_DATA
        pop PARAM
        ret

;  *********************************************
;  LCD Driving Library example
; Display setup for 4 bits interface
;
LCD_SETUP:

  ldi TEMP, 10          ; Wait about 20msec after powerup
SET1:                 ; LCD is a quite slow,
  rcall WAIT_2M         ; mind delays !
  dec TEMP             ;
  brne SET1            ;

  ldi TEMP, 0b00101000   ; Set 4 bit interface (but we are
  out PORTA, TEMP        ; still in 8 bits!)
  nop
  nop                 ; Data write cycle must be > 1 ms
  cbi PORTA, 3          ; OE low to clock in data
  rcall WAIT_2M         ;
```

```
; ******* !! From now on, interface is 4 bits !! ***

  ldi PARAM, 0b00101000   ; Send again to catch the bit N
  rcall LCD_CMD           ; Display is 2 lines, so N = 1

  ldi PARAM, 0b00001000   ; Display off, cursor off, and blink off
  rcall LCD_CMD

  ldi PARAM, 0b00000001   ; Display clear
  rcall LCD_CMD
  rcall WAIT_2M           ; Wait 2 sec after clear

  ldi PARAM, 0b00000110   ; Increment RAM, dont shift display
  rcall LCD_CMD
  ldi PARAM, 0b00001100   ; Display on, cursor off / blink off
  rcall LCD_CMD
  ret

; ***************************
;
; 2.3 ms delay with 1 MHz clock.
;
WAIT_1SEC:
  clr DEL2
  dec DEL2              ; Start at 0xFF
W1_2:
  dec DELAY2
  brne W1_2
  pop TEMP
  ret

WAIT_2M:
  push TEMP
  ldi TEMP, 3
  mov DELAY1, TEMP
W2_1:
  clr DELAY2
  dec DELAY2            ; Start at 0xFF
W2_2:
  dec DELAY2
  brne W2_2
  dec DELAY1
  brne W2_1
```

```
  pop TEMP
  ret

RETURN_CURSOR_4:
        push PARAM
        ldi PARAM, 0x40
        rcall LCD_CURSOR
        pop PARAM
        ret

CURSOR_TO_FIRST_STRING:
        push PARAM
        ldi PARAM, 0x00
        rcall LCD_CURSOR
        pop PARAM
        ret

LCD_TXT_READY:
        rcall CURSOR_TO_FIRST_STRING
        ldi ZH, HIGH(2*TXT_READY)
  ldi ZL, LOW(2*TXT_READY)
  rcall L3A
  ret

LCD_TXT_STARTED:
        rcall CURSOR_TO_FIRST_STRING
        ldi ZH, HIGH(2*TXT_STARTED)
  ldi ZL, LOW(2*TXT_STARTED)
  rcall L3A
  ret

LCD_TXT_PAUSED:
        rcall CURSOR_TO_FIRST_STRING
        ldi ZH, HIGH(2*TXT_PAUSED)
  ldi ZL, LOW(2*TXT_PAUSED)
  rcall L3A
  ret

LCD_TXT_STOPPED:
        rcall CURSOR_TO_FIRST_STRING
        ldi ZH, HIGH(2*TXT_STOPPED)
  ldi ZL, LOW(2*TXT_STOPPED)
  rcall L3A
  ret
```

```
L3A:
        ldi PARAM, 0x00
        rcall LCD_CURSOR
L3B:
  lpm               ; From prog mem into R0
  mov PARAM, R0
  tst PARAM
  brne L3C
  rjmp L4
L3C:
  rcall LCD_DATA
L3D:
  adiw ZL, 1        ; Increment pointer. This neat instruction does
  rjmp L3B          ; w16 bit addition to pointer ZHI:ZLO.
L4:
  ret               ; Done

TXT_READY:
        .DB "Press a key to start",0x00
TXT_STARTED:
  .DB "Stopwatch started   ",0x00
TXT_PAUSED:
  .DB "Stopwatch paused    ",0x00
TXT_STOPPED:
  .DB "Stopwatch stopped   ",0x00

;*********************************************************
;  Sends a control function to the display (comes in PARAM)
LCD_CMD:
  push PARAM
  mov TEMP, PARAM
  andi PARAM, 0xF0     ; Mask off lower 4 bits
  sbr PARAM, 8         ; OE bit high (bit 3)
  out PORTA, PARAM     ; Send upper 4 bits to display
  nop                  ; Brief delay to give reasonable OE
  nop                  ; pulse width
  cbi PORTA,3          ; OE goes low to clock in data

  mov PARAM, TEMP      ; Data back
  swap PARAM           ; Lower 4 bits
  andi PARAM, 0xF0
  sbr PARAM, 8         ; OE high (bit 3, port D)
  out PORTA, PARAM     ; Write lower 4 bits to LCD
```

```
  nop
  nop
  cbi PORTA,3         ; OE clock low

  ldi PARAM, 100      ; 50 usec approx
DEL0:
  dec PARAM
  brne DEL0
  pop PARAM
  ret

;*****************************************************
; Sends an ASCII character to the display (comes in PARAM)
LCD_DATA:
  push PARAM
  push TEMP
  mov TEMP, PARAM
  andi PARAM, 0xF0    ; Mask off lower 4 bits
  sbr PARAM, OE       ; OE bit high
  sbr PARAM, RS       ; Data/command bit high
  out PORTA, PARAM    ; Write upper 4 bits to display
  nop                 ; Brief delay
  nop
  cbi PORTA, 3        ; OE low to clock data

  swap TEMP           ; Lower 4 bits
  andi TEMP, 0xF0     ; Masked off
  sbr TEMP, OE        ; OE high
  sbr TEMP, RS
  out PORTA, TEMP     ; Lower 4 bits to LCD
  nop
  nop
  cbi PORTA, 3        ; OE low to clock data

  ldi TEMP, 100
  mov DELAY, TEMP     ; 50 usec
DEL1:
  dec DELAY
  brne DEL1
  pop TEMP
  pop PARAM
  ret

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

```
;  Moves the LCD display cursor to address specified in PARAM
LCD_CURSOR:
  ori   PARAM, 0x80      ; MSB specifies address command
  rcall LCD_CMD
  ret
```

Порядок выполнения лабораторной работы

Необходимо выполнить все необходимые операции с проектом Lab_4.prj и программой Lab4.asm в среде VMLab как и в лабораторной работе №1, 2, 3.

Содержание отчета

Отчет по проделанной лабораторной работе должен содержать:

- структурную схему алгоритма программы на лабораторную работу;
- схему электрическую принципиальную разработанного устройства;
- текст программы на языке AVR Ассемблере (распечатка файла *asm);
- распечатка файла проекта (*.prj);
- результаты моделирования (окно VMLab с запущенной программой и результатами вычислений).

Контрольные вопросы

1. Директивы языка ассемблера AVR.
2. Схемы тактовых генераторов микроконтроллера ATmega16 и их программирование.
3. Источники сброса микроконтроллера ATmega16.
4. Сторожевой таймер микроконтроллера ATmega16.
5. Алгоритм обработки прерываний микроконтроллера ATmega16.
6. Вектора прерываний микроконтроллера ATmega16.
7. Энергонезависимая память данных микроконтроллера ATmega16.
8. Асинхронный обмен данными с микроконтроллером ATmega16.

Лабораторная работа № 5

ПРОГРАММИРОВАНИЕ В СРЕДЕ VISUAL MICRO LAB В МУЛЬТИПРОЦЕССОРНОМ РЕЖИМ

Цель работы: Изучение и освоение программирования в мультипроцессорном режиме, а также дальнейшее изучение системы команд и системы прерываний.

Задание к лабораторной работе

Изучить программы mp.asm, mp1.asm и файлы проекта mp.prj. mp1.prj. Отладить программы в среде VMLab, подключив необходимую периферию к микроконтроллерам AVR – к первому интерфейс ввода, ко второму интерфейс вывода UART. Обмен между микроконтроллерами также через интерфейс UART. Результаты моделирования в мультипроцессорном режиме представлены на рис.23.

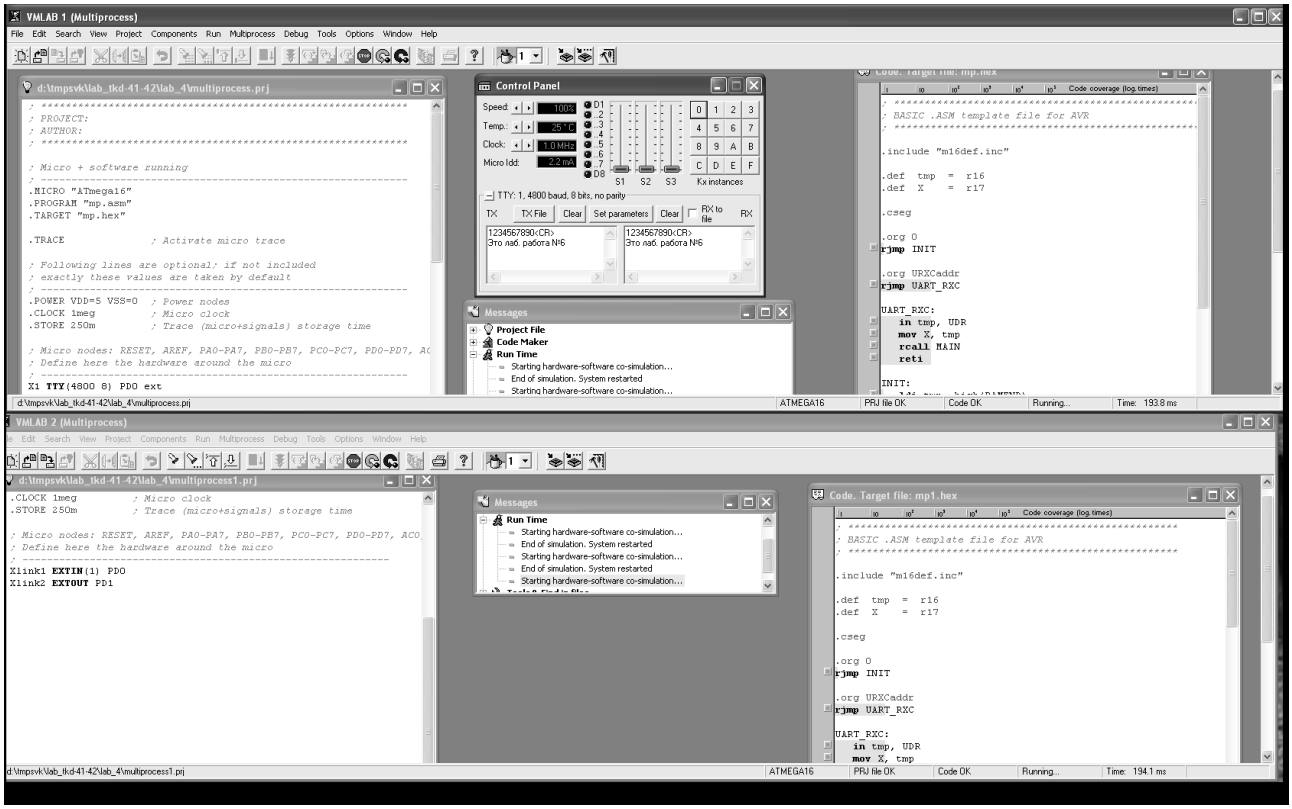


Рис.23. Результаты моделирования микроконтроллеров в мультипроцессорном режиме

Файл mp.prj

```
; ************************************************************
; Micro + software running
```

```
.MICRO "ATmega16"
.PROGRAM "mp.asm"
.TARGET "mp.hex"
.TRACE          ; Activate micro trace
; Following lines are optional; if not included
; exactly these values are taken by default
.POWER VDD=5 VSS=0  ; Power nodes
.CLOCK 1meg       ; Micro clock
.STORE 250m       ; Trace (micro+signals) storage time

;nodes: RESET,AREF,PA0–PA7, PB0–PB7, PC0–PC7, PD0–PD7, ACO,
TIM1OVF
; ————————————————————————————————————————
X1 TTY(4800 8) PD0 ext
Xlink1 EXTOUT PD1
Xlink2 EXTIN(1) ext
```

Файл mp1.prj

```
; ***********************************************************
.MICRO "ATmega16"
.PROGRAM "mp1.asm"
.TARGET "mp1.hex"
.TRACE          ; Activate micro trace
; Following lines are optional; if not included
; exactly these values are taken by default
.POWER VDD=5 VSS=0  ; Power nodes
.CLOCK 1meg       ; Micro clock
.STORE 250m       ; Trace (micro+signals) storage time
;nodes:RESET,AREF, PA0–PA7, PB0–PB7, PC0–PC7, PD0–PD7, ACO,
TIM1OVF
; ————————————————————————————————————————
Xlink1 EXTIN(1) PD0
Xlink2 EXTOUT PD1
```

Файл mp.asm

```
; ***********************************************************
.include "m16def.inc"
.def  tmp  =  r16
.def  X   =  r17
.cseg
.org 0
rjmp INIT
.org URXCaddr
rjmp UART_RXC
UART_RXC:
```

```
        in tmp, UDR
        mov X, tmp
        rcall MAIN
        reti
INIT:
       ldi tmp, high(RAMEND)  ; инициализация памяти стека
       out SPH, tmp
       ldi tmp, low(RAMEND)
       out SPL, tmp          ; завершение инициализации памяти стека
        ldi tmp, 0x0C
        out UBRR, tmp         ; установка скорости UART 4800 бод
        ldi tmp, 0x98
        out UCR, tmp          ; инициализация UART (запрет прерываний по окон-
чанию передачи и по пустому значению регистра UDR)
        sei                   ; глобальное включение прерываний
LOOP:
        nop
        nop
        rjmp LOOP
MAIN:
        ldi tmp, 1
        add X, tmp
        out UDR, X
        ret
```

Файл mp1.asm

```
; ******************************************************
      .include "m16def.inc"
      .def  tmp  =  r16
      .def  X    =  r17
      .cseg
      .org 0
      rjmp INIT
      .org URXCaddr
      rjmp UART_RXC
      UART_RXC:
        in tmp, UDR
        mov X, tmp
        rcall MAIN
        reti
INIT:
       ldi tmp, high(RAMEND)  ; инициализация памяти стека
       out SPH, tmp
       ldi tmp, low(RAMEND)
```

```
        out SPL, tmp          ; завершение инициализации памяти стека
        ldi tmp, 0x0C
        out UBRR, tmp         ; установка скорости UART 4800 бод
        ldi tmp, 0x98
        out UCR, tmp          ; инициализация UART
        sei                   ; глобальное включение прерываний
LOOP:
        nop
        nop
        rjmp LOOP
MAIN:
        subi X, 1
        ;subi X, 0x30
        out UDR, X
        ret
```

Порядок выполнения лабораторной работы

Необходимо выполнить все необходимые операции с проектами mp.prj и mp1.prj и программами mp.asm и mp1.asm в среде VMLab как и в лабораторной работах №1, 2, 3, 4.

Содержание отчета

Отчет по проделанной лабораторной работе должен содержать:

- структурную схему алгоритма программы на лабораторную работу;
- схему электрическую принципиальную разработанного устройства;
- текст программы на языке AVR Ассемблере (распечатка файла *asm);
- распечатка файла проекта (*.prj);
- результаты моделирования (окно VMLab с запущенной программой и результатами вычислений).

Контрольные вопросы

1. Симплексный обмен данными с микроконтроллером ATmega16.
2. Полудуплексный обмен данными с микроконтроллером ATmega16.
3. Дуплексный обмен данными с микроконтроллером ATmega16.
4. Аналого–цифровой преобразователь последовательного приближения микроконтроллера ATmega16.
5. Управление устройством ADC микроконтроллера ATmega16.
6. Аналоговый компаратор микроконтроллера AVR ATmega16.

7. Управление АС микроконтроллера ATmega16.
8. Таймеры–счетчики микроконтроллера ATmega16.

Лабораторная работа № 6

ПРОГРАММИРОВАНИЕ АНАЛОГОЦИФРОВОГО ПРЕОБРАЗОВАТЕЛЯ МИКРОКОНТРОЛЛЕРОВ

Цель работы: Изучение и освоение методики программирования аналогоцифрового преобразователя микроконтроллеров.

Краткие теоретические сведения.
Управление аналого–цифровым преобразователем

Микроконтроллер ATmega16 оснащен 10–разрядным ADC последовательных приближений (рис. 24). ADC подсоединен к 10– канальному аналоговому мультиплексору (MUX), позволяющему подать на вход преобразователя любой из восьми входных сигналов со входов ADC0…. ADC7, либо эталонное напряжение 1,22 В, либо сигнал со входа AGND. Вывод AGND рекомендуется подсоединить к точке с нулевым потенциалом GND. ADC содержит схему выборки/хранения SH.

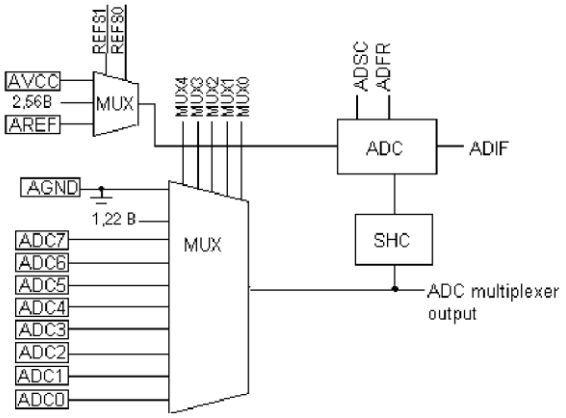


Рис. 24. Структура аналого–цифрового преобразователя

Результат AD преобразования в виде 10–битного двоичного числа D равен:

$$D = \frac{U}{U_0} \cdot 2^{10},$$

где *U*– входное напряжение, а U_o– опорное напряжение преобразователя.

В качестве источника опорного напряжения преобразователя можно использовать внешний сигнал с вывода AREF, внутренний источник 2,56 В, либо напряжение питания аналоговой части микроконтроллера с вывода AVCC.

Для управления преобразователем в микроконтроллере используются регистры (рис. 25):

[0] Регистр управления мультиплексором ADMUX
[0] Регистр управления аналого–цифровым преобразователем ADCSR
[0] Регистры данных ADCL и ADCH
[0] Регистр состояния микроконтроллера SREG

Биты	7	6	5	4	3	2	1	0
ADMUX $07 ($27)	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0
ADCSR $06 ($26)	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0
ADCH $05 ($25)	SIGN	-	-	-	-	-	ADC9	ADC8
ADCL $04 ($24)	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
SREG $3F ($5F)	I							

Рис. 25. Регистры, используемые аналого–цифровым преобразователем

Аналого–цифровой преобразователь может работать в двух режимах: режиме однократного преобразования и в циклическом режиме. Выбор режима производится битом ADFR регистра ADCSR.

Работа аналого–цифрового преобразователя разрешается установкой в состояние 1 бита ADEN в регистре ADCSR. Преобразование начинается с установки в состояние 1 бита начала преобразования ADSC

Поскольку аналого–цифровой преобразователь формирует 10–разрядный результат, то по завершении преобразования результирующие данные размещаются в двух регистрах данных ADCH и ADCL.

Аналого–цифровой преобразователь имеет свое собственное прерывание ADC (вектор $1C), которое может быть активизировано по завершению преобразования. Когда обращение к регистрам запрещено, в процессе считывания регистров ADCL и ADCH, прерывание будет активизироваться, даже при потере результата.

Регистр ADMUX предназначен для управления входным аналоговым мультиплексором.

[0] Биты 7 и 6 - REFS1..0 – обеспечивают выбор эталонного напряжения на входе AREF аналого–цифрового преобразователя. Выбор производится в соответствии с таблицей 28.

Таблица 28. Выбор источника опорного напряжения ADC

REFS1	REFS0	Выбор источника напряжения
0	0	AREF, внутреннее напряжение Vref отключено
0	1	AVCC с внешним конденсатором на контакте AREF
1	0	Резерв
1	1	Внутренний источник 2.56 В с внешним конденсатором на AREF

[0] Бит 5 - ADLAR – воздействует на запись результата в регистры данных ADCL и ADCH. При ADLAR=0 можно использовать упрощенное 8–битное преобразование.

⁰ Биты 4..0 - MUX4..MUX0 – предназначены для выбора входа, коммутируемого на вход преобразователя. Выбор осуществляется в соответствии с таблицей 29. Изменение этих битов в процессе преобразования, когда флаг ADIF в регистре ADCSR установлен, не приводит к изменению результата.

Таблица 29. Выбор входного сигнала ADC

MUX4..0	Подключаемый контакт
00000	ADC0
00001	ADC1
00010	ADC2
00011	ADC3
00100	ADC4
00101	ADC5
00110	ADC6
00111	ADC7
01000. .11101	Резерв
11110	1.22V
11111	0V (AGND)

Регистр – ADCSR предназначен для управления работой аналого–цифрового преобразователя.

⁰ Бит 7 – ADEN – разрешение работы ADC.

⁰ Бит 6 – ADSC – запуск преобразования ADC.

⁰ Бит 5 – ADFR – установка циклического режима работы ADC.

⁰ Бит 4 – ADIF – флаг прерывания ADC.

⁰ Бит 3 – ADIE – разрешение прерывания ADC.

⁰ Биты 2..0 – ADPS2..ADPS0 – выбор коэффициента предварительного деления (табл.30).

Таблица 30. Выбор коэффициента предварительного деления

ADPS2	ADPS1	ADPS0	Коэффициент деления
0	0	0	Без деления
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

Регистры ADCL и ADCH являются регистрами данных.

Если ADLAR сброшен, то результат представлен в виде, изображенном на рис. 26.

	Биты	7	6	5	4	3	2	1	0
ADCH	$05 ($25)	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2
ADCL	$04 ($24)	ADC1	ADC0						

Рис. 26. Представление результата в регистре данных при ADLAR=0

Если достаточным является 8-битное преобразование, то при ADLAR=0 можно считывать только старший байт результата (регистр ADCH).

Задание к лабораторной работе

Изучить программу Lb6.asm и файл проекта Lb_6.prj. Отладить программу в среде VMLab, подключив необходимую периферию к микроконтроллеру AVR – источник синусоидального напряжения. Программа преобразует напряжение на выводе PA5 микроконтроллера в двоичный 10-разрядный код. Используется 8 старших разрядов. Преобразование осуществляется по прерываниям от ADC. На рис. 27 представлены результаты моделирования программы управления аналогоцифровым преобразователем по прерываниям.

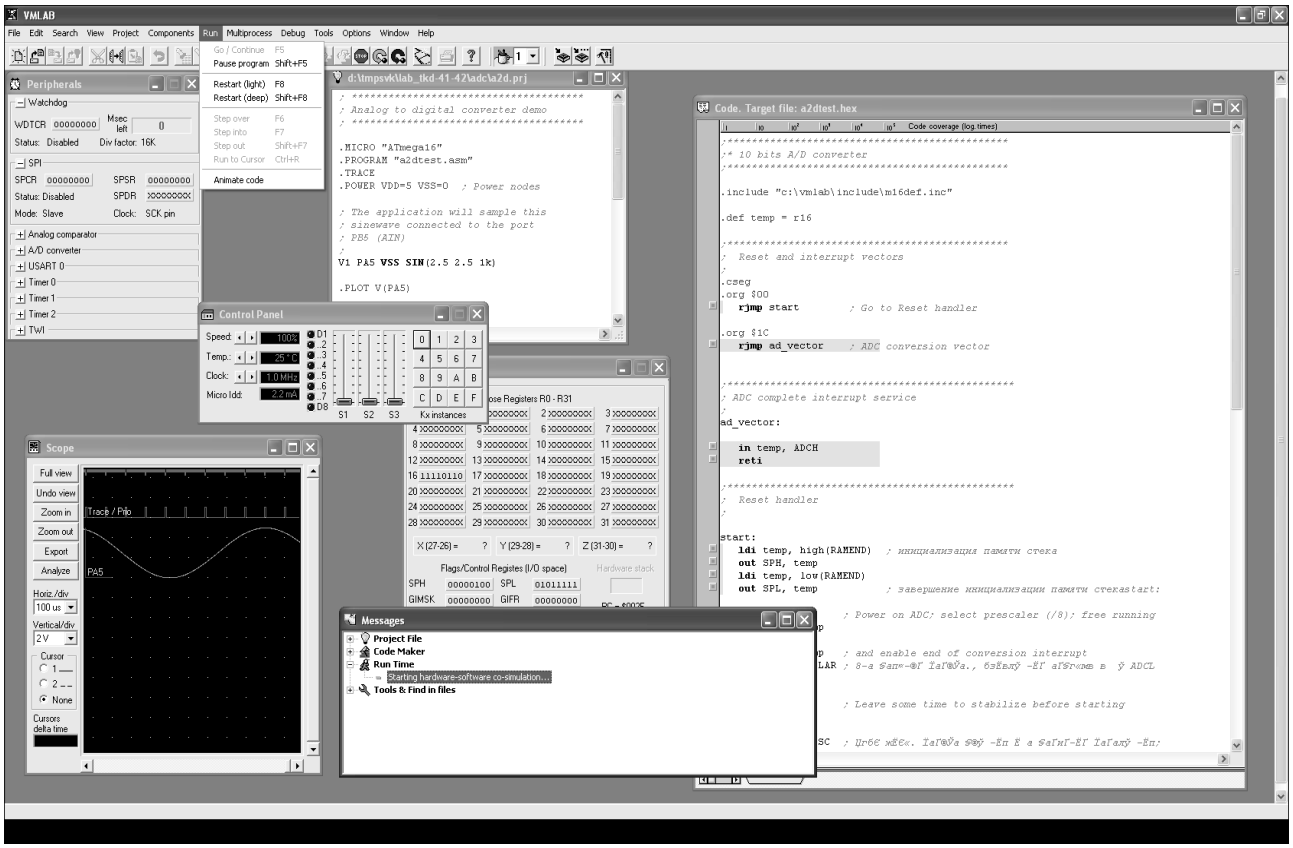


Рис.27. Результаты моделирования программы
управления аналого–цифровым преобразователем

Файл Lab_6.prj

```
; ************************************
; Analog to digital converter demo
; ************************************

.MICRO "ATmega16"
.PROGRAM "Lab6.asm"
.TRACE
.POWER VDD=5 VSS=0  ; Power nodes

; Используется вход PA5 микроконтроллера
V1 PA5 VSS SIN(2.5 2.5 1)
.PLOT V(PA5)
```

Файл Lab6.asm

```
;************************************************
;* 10 bits A/D converter
;************************************************
.include "c:\vmlab\include\m16def.inc"
.def temp = r16

;************************************************
;  Reset and interrupt vectors
;
.cseg
.org $00
  rjmp start        ; Go to Reset handler

.org $1C
  rjmp ad_vector    ; ADC conversion vector

;************************************************
; ADC complete interrupt service
;
ad_vector:

  in temp, ADCH
  reti

;************************************************
;  Reset handler
```

```
;

start:
  ldi temp, high(RAMEND)  ; инициализация памяти стека
        out SPH, temp
        ldi temp, low(RAMEND)
        out SPL, temp          ; завершение инициализации памяти стекаstart:

  ldi temp, $45    ; Power on ADC; select prescaler (/8); free running
  out ADMUX,temp
  ldi temp, $BB
  out ADCSR,temp   ; and enable end of conversion interrupt
  sbi ADMUX, ADLAR
  nop
  nop
  nop            ; Leave some time to stabilize before starting
  nop

  sbi ADCSR, ADSC
  sei

forever:
  rjmp  forever  ; Infinite loop, interrupted by ADC conversions
```

Порядок выполнения лабораторной работы

Необходимо выполнить все необходимые операции с проектом Lab_6.prj и программой Lab6.asm в среде VMLab.

Содержание отчета

Отчет по проделанной лабораторной работе должен содержать:

- структурную схему алгоритма программы на лабораторную работу;
- схему электрическую принципиальную разработанного устройства;
- текст программы на языке ассемблера или СИ (распечатка файлов *c, *asm);
- распечатка файла проекта (*.prj);

– результаты моделирования (окно VMLab с запущенной программой и результатами вычислений).

Контрольные вопросы

1. Входные сигналы аналого–цифрового преобразователя микроконтроллера ATmega16.
2. Управление мультиплексором входных сигналов для ADC микроконтроллера ATmega16.
3. Задание режима 8 и 10 разрядного преобразования ADC (бит ADLAR) микроконтроллера ATmega16.
4. Источники опорного напряжения для ADC и их задание в микроконтроллере ATmega16.
5. Выбор коэффициента предварительного деления тактовой частоты для ADC в микроконтроллере ATmega16.
6. Бит разрешения прерывания в ADC микроконтроллера AVR ATmega16.
7. Флаг преобразования ADC микроконтроллера ATmega16.
8. Бит задания режима однократного или циклического преобразования ADC микроконтроллера ATmega16.
9. Бит пуска преобразования ADC в однократном режиме микроконтроллера ATmega16.
10. Бит разрешения преобразования ADC микроконтроллера ATmega16.

Лабораторная работа № 7

ПРОГРАММИРОВАНИЕ АНАЛОГОВОГО КОМПАРАТОРА МИКРОКОНТРОЛЛЕРОВ

Цель работы: Изучение и освоение методики программирования аналогового компаратора микроконтроллеров.

Краткие теоретические сведения
АНАЛОГОВЫЙ КОМПАРАТОР

Аналоговый компаратор микроконтроллера ATmega16 имеет два входа — AIN0 и AIN1. В состав аналогового компаратора кроме базового компаратора входит регистр управления–состояния ACSR (№ $08) и элементы, управляющие работой схемы. Результатом работы компаратора является запрос прерывания ANA COMP, который формируется, когда разность значений напряжения на входах компаратора меняет знак.

Схема управления СУ при определенном изменении сигнала ACO устанавливает в единичное состояние разряд ACI регистра ACSR и при единичном состоянии разряда ACIE регистра ACSR в блок прерываний поступает запрос прерывания ANA COMP. Разряд ACI сбрасывается в нулевое состояние аппаратно при переходе к выполнению прерывающей программы или программно путем записи единицы в разряд ACI.

Выбор вида изменения сигнала ACO на входе схемы управления СУ, при котором формируется запрос прерывания, определяется комбинацией состояний разрядов ACISO и ACIS1 регистра ACSR в соответствии с табл. 31.

Таблица 31. Выбор вида сигнала компаратора

ACIS1	ACIS0	Изменение сигнала ACO
0	0	любое
0	1	—
1	0	1→0
1	1	0→1

В микроконтроллере ATmega16 сигнал ACO с выхода базового компаратора при единичном состоянии разряда ACIC принимается в таймер–счетчик в качестве сигнала, управляющего захватом.

При установке в единичное состояние разряда ACD регистра ACSR отключается питание базового компаратора и уменьшается ток потребления микроконтроллера.

В микроконтроллере ATmega16 имеется возможность подключать к входу «+» базового компаратора вместо входа AIN0 выход внутреннего источника эталонного напряжения VR (1,22 ± 0,05 В). Подключение источника VR выпол-

няется при единичном состоянии разряда AINBG регистра ACSR, кроме того, имеется возможность подключать к входу «–» базового компаратора входы аналого–цифрового преобразователя ADC0...ADC7. Подключение выполняется при нулевом состоянии разряда ADEN регистра ADCSR (№ $06) и единичном состоянии разряда ACME регистра SFIOR (№ $30). Выбор подключаемого входа определяется комбинацией состояний разрядов MUX2, MUX1 и MUX0 регистра ADMUX (№ $07).

В табл. 32 указаны выводы микроконтроллера, используемые в качестве входов AIN0 и AIN1, у микроконтроллеров разных типов.

Таблица 32. Входы AIN0 и AIN1

Вход	t11 t12	t15 t28	1200 2313	4433	8515 8535	m163	m103*
AIN0	PB0	PB0	PB0	PD6	PB2	PB2	PE2
AIN1	PB1	PB1	PB1	PD7	PB3	PB3	PE3

* — AC+, AC–

Схема компаратора микроконтроллера *ATmega16* приведена на рис. 28.

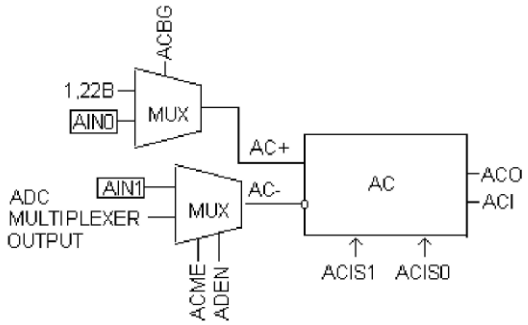


Рис. 28. Функциональная схема аналогового компаратора

В работе компаратора используются регистры (рис. 29):

- Регистр управления аналоговым компаратором ACSR
- Регистр специальных функций ввода/вывода
- Регистр состояния аналого–цифрового преобразователя
- Регистр мультиплексора аналого–цифрового преобразователя ADCMUX
- Регистр состояния микроконтроллера SREG.

Биты		7	6	5	4	3	2	1	0
ACSR	$08 ($28)	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0
SFIOR	$30 ($50)					ACME			
ADCSR	$06 ($26)	ADEN							
ADMUX	$07 ($27)						MUX2	MUX1	MUX0
SREG	$3F ($5F)	I							

Рис. 29. Регистры, задействованные в работе аналогового компаратора

Регистр ACSR предназначен для управления аналоговым компаратором.

[0] Бит 7 – ACD – отключает аналоговый компаратор.

[0] Бит 6 – ACBG – выбор эталона аналогового компаратора.

[0] Бит 5 – ACO – выход аналогового компаратора.

[0] Бит 4 – ACI – флаг прерывания по аналоговому компаратору.

[0] Бит 3 – ACIE – разрешение прерывания по аналоговому компаратору.

[0] Бит 2 – ACIC – разрешение входа захвата аналогового компаратора.

[0] Биты 1,0 – ACIS1, ACIS0 – выбор режима прерывания по аналоговому компаратору. Варианты установок показаны в таблице 33.

Таблица 33. Установки битов ACIS1/ACIS0

A	A	Режим прерывания
0	0	Прерывание по переключению выхода компаратора
1	0	Зарезервировано
1	0	Прерывание по падающему фронту на выходе компаратора
1	1	Прерывание по нарастающему фронту на выходе компаратора

При изменении состояния битов ACIS1/ACIS0 прерывание по аналоговому компаратору должно быть запрещено очисткой бита разрешения прерывания в регистре ACSR. В противном случае, при изменении состояния битов может произойти прерывание.

На отрицательный вход аналогового компаратора можно скоммутировать любой из входов порта PORTA (PA7 .. PA0). Для выбора входа используется мультиплексор аналогоцифрового преобразователя.

[0] Бит ACME в регистре специальных функций ввода вывода SFIOR предназначен для подключения мультиплексора к аналоговому компаратору.

[0] Битами MUX2..0 в регистре мультиплексора ADMUX выбирается контакт на входе (табл. 34).

Таблица 34. Логика подключения отрицательного входа компаратора.

ACME	ADEN	MUX2..0	Отрицательный вход аналогового компаратора
0	**X**	**XXX**	AIN1
1	1	**XXX**	AIN1
1	0	000	PAO
1	0	001	PA1
1	0	010	PA2
1	0	011	PA3
1	0	100	PA4
1	0	101	PA5
1	0	110	PA6
1	0	111	PA7

Задание к лабораторной работе

Изучить программу Lb7.asm и файл проекта Lb_7.prj. Отладить программу в среде VMLab, подключив необходимую периферию к микроконтроллеру AVR – источники синусоидального напряжения. Программа сравнивает аналоговые напряжения на контактах PB2 и PB3, на которые подаются сигналы от генераторов синусоидальных сигналов с разной частотой. Визуально просматривается сигнал с выхода аналогового компаратора. На рис. 30 представлены результаты моделирования программы управления аналоговым преобразователем по прерываниям.

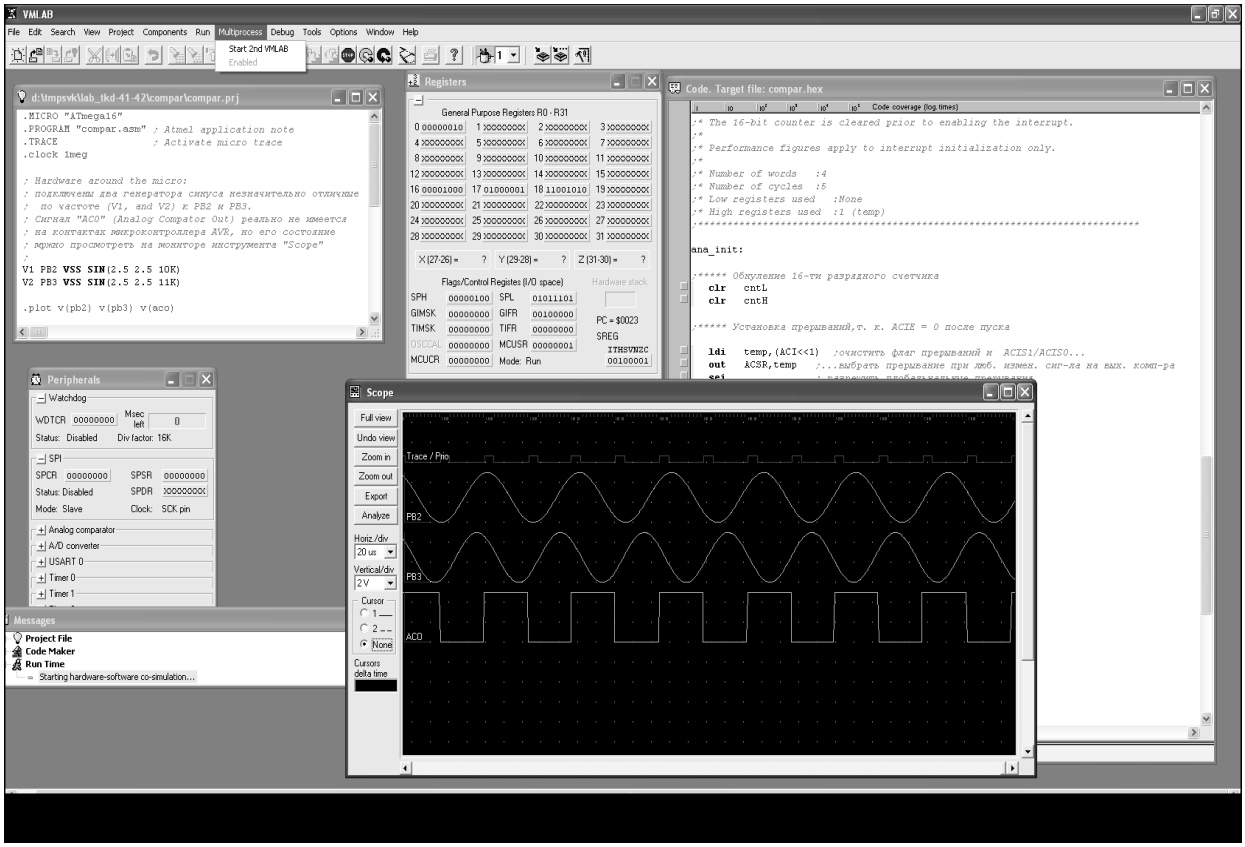


Рис.30. Результаты моделирования программы управления аналоговым компаратором

Файл Lab_7.prj

```
.MICRO "ATmega16"
.PROGRAM "Lab7.asm" ; Atmel application note
.TRACE              ; Activate micro trace
.clock 1meg
; подключены два генератора синуса незначительно отличные
;  по частоте (V1, and V2) к PB2 и PB3.
; Сигнал «AC0» (Analog Compator Out) реально не имеется
```

```
; на контактах микроконтроллера AVR, но его состояние
; можно просмотреть на мониторе инструмента «Scope»
V1 PB2 VSS SIN(2.5 2.5 10K)
V2 PB3 VSS SIN(2.5 2.5 11K)
.plot v(pb2) v(pb3) v(aco)
```

Файл Lab7.asm

```
;* Программа использует аналоговый компаратор микроконтроллера
;* – ожидание по положительному перепаду выхода компаратора
;* – ожидание по положительному перепаду флага прерывния
;* – Enable interrupt on comparator output toggle. The interrupt routine
;* – инкремент 16–ти разрядного счетчика ter counter each time it is executed
.include "C:\vmlab\include\m16def.inc"
.def  temp  =r16     ;temporary storage register
.def  cntL  =r17     ;регистр счета младшего байта
.def  cntH  =r18     ;регистр счета старшего байта
;* Инициализация векторов прерываний
  rjmp  RESET    ;В начало программы
.org  ACIaddr
  rjmp  ANA_COMP ;В подпрограмму обработки прерывания от AC
;******** Подпрограмма обработки прерывания от AC**********************
;* Эта подпрограмма увеличивает содержимое 16–ти разрядного счетчика
;* всякий раз когда возникает прерывание по аналоговому компаратору
;*****************************************************************
.def  ac_tmp   =r0      ;временный регистр хранения для SREG
ANA_COMP:
  in ac_tmp,SREG ;запоминание SREG
  subi  cntL,low(–1)
  sbci  cntH,high(–1)  ;инкремент 16–ти разрядного счетчика cnt(H,L)=cnt(H,L)–
(–1)
  out   SREG,ac_tmp ;восстановление SREG
  reti ; возврат из прерываний
;******************** Основная программа ****************************
RESET:
;***** Инициализация стека
 ldi   temp,low(RAMEND)
 out   SPL,temp
 ldi   temp,high(RAMEND)
 out   SPH,temp
;***************** «ожидание_edge1»
;* Эта часть осуществляет ожидание (задержку) пока выход компаратора
;* (ACO–bit в ACSR) не станет равным 1.
;* extremely short pulses can be missed, since the program runs three clock
```

```
;* cycles between each time the comparator is checked. Another disadvantage
;* is that the program has to wait for the output to be come negative first,
;* in case the output is positive when polling starts.
;* Number of words   :4
;* Number of cycles  :4 per loop. Response time: 3 – 5 clock cycles
;* Low registers used   :None
;* High registers used  :None
wait_edge1:
  sbic  ACSR,ACO ;if output is high
  rjmp  wait_edge1 ;    wait
we1_1:
  sbis  ACSR,ACO ;if output is low
  rjmp  we1_1    ;    wait
;********************** «ожидание_edge2» ***************************
;* This piece of code waits until the output of the comparator (the ACO–bit
;* in ACSR) goes high. This is a more secure solution, since the interrupt
;* flag is polled. This allows the user to insert code within the wait loop
;* because hardware «remembers» pulses of shorter duration than the polling
;* interval. Another positive feature is that there is no need to wait for
;* a preceeding negative edge.
;* Number of words   :5
;* Number of cycles  :Inital setup :2
;*       Flag clearing:1
;*       Loop       :4
;*       Response time:3 – 5
;* Low registers used   :None
;* High registers used  :None
wait_edge2:
;***** Инициализация режима прерывания, при пуске (reset) ACIE = 0
  sbi   ACSR,ACIS0
  sbi   ACSR,ACIS1  ;установка режима прерывания с 0 в 1
;***** Ожидание
  sbi   ACSR,ACI ;запись «1» в ACI
we2_1:
  sbis  ACSR,ACI ;if ACI is low
  rjmp  we2_1
;******************* «ana_init» ************************************
;* This code segment enables Analog Comparator Interrupt on output toggle.
;* The program then enters an infinite loop.
;* The 16–bit counter is cleared prior to enabling the interrupt.
;* Performance figures apply to interrupt initialization only.
;* Number of words   :4
;* Number of cycles  :5
;* Low registers used   :None
```

```
;* High registers used  :1 (temp)
;*****************************************************************
*
ana_init:
;***** Обнуление 16–ти разрядного счетчика
  clr   cntL
  clr   cntH
;***** Установка прерываний, т. к. ACIE = 0 после пуска
  ldi   temp,(ACI<<1)  ;очистить флаг прерываний и  ACIS1/ACIS0...
  out   ACSR,temp   ;...выбрать прерывание при люб. измен. сиг–ла на вых.
комп–ра
  sei             ; разрешить глобальные прерывания
  sbi   ACSR,ACIE   ;разрешение прерывания от AC
forever:rjmp   forever
```

Порядок выполнения лабораторной работы

Нужно выполнить все необходимые операции с проектом Lab_7.prj и программой Lab7.asm в среде VMLab.

Содержание отчета

Отчет по проделанной лабораторной работе должен содержать:

- структурную схему алгоритма программы на лабораторную работу;
- схему электрическую принципиальную разработанного устройства;
- текст программы на языке ассемблера (распечатка файла *asm);
- распечатка файла проекта (*.prj);
- результаты моделирования (окно VMLab с запущенной программой и результатами вычислений).

Контрольные вопросы

1. Входные сигналы аналогового компаратора микроконтроллера ATmega16.
2. Управление компаратором микроконтроллера ATmega16.
3. Задание режима прерывания микроконтроллера ATmega16.
4. Источники входных сигналов аналогового компаратора в микроконтроллере ATmega16.
5. Разрешение входа захвата таймером AC в микроконтроллере ATmega16.
6. Бит разрешения прерывания в AC микроконтроллера AVR ATmega16.
7. Флаг прерывания в AC микроконтроллера ATmega16.
8. Бит выхода аналогового компаратора микроконтроллера ATmega16.

9. Бит выбора эталона аналогового компаратора микроконтроллера ATmega16.

10. Бит отключения аналогового компаратора микроконтроллера ATmega16.

Лабораторная работа № 8

ПРОГРАММИРОВАНИЕ МИКРОКОНТРОЛЛЕРОВ НА ЯЗЫКЕ СИ

Цель работы: Изучение и освоение методики программирования микроконтроллеров на языке СИ.

Задание к лабораторной работе

Изучить программу Lab8.c и файл проекта Lab_8.prj. Отладить программу в среде VMLab, подключив необходимую периферию к микроконтроллеру AVR – интерфейс UART. Программа вводит через устройство UART целое число, извлекает корень квадратный , вычитает 23 и выводит результат через устройство UART. На рис 31 представлены результаты моделирования программы на языке СИ.

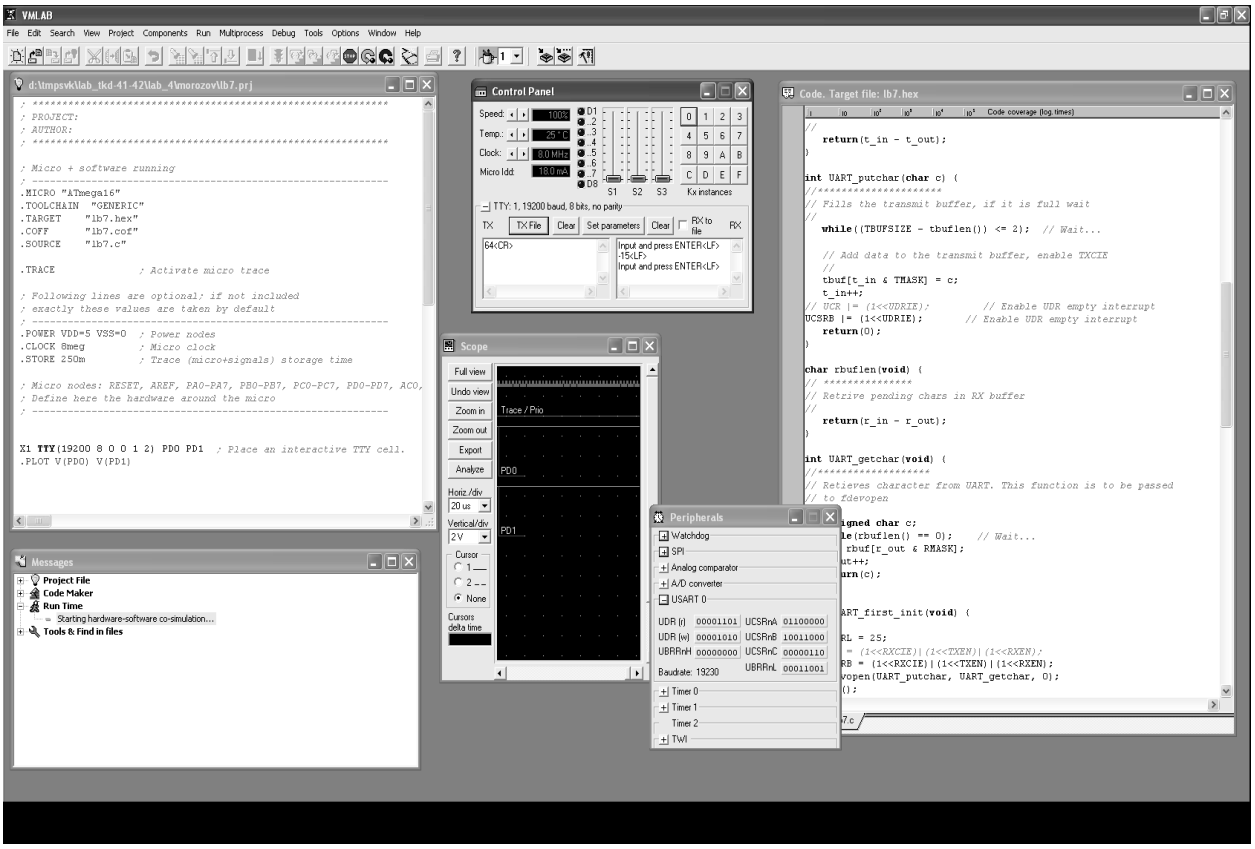


Рис. 31. Результаты моделирования программы на языке СИ

Файл Lab_8.prj

```
.MICRO "ATmega16"
.TOOLCHAIN  "GENERIC"
```

```
.TARGET    "lab8.hex"
.COFF      "lab8.cof"
.SOURCE    "lab8.c"
.TRACE             ; Activate micro trace
.POWER VDD=5 VSS=0  ; Power nodes
.CLOCK 8meg         ; Micro clock
.STORE 250m         ; Trace (micro+signals) storage time
X1 TTY(19200 8 0 0 1 2) PD0 PD1  ; Place an interactive TTY cell.
.PLOT V(PD0) V(PD1)
```

Файл Lab8.c

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/signal.h>
#include <avr/pgmspace.h>
#include <stdio.h>
#include <math.h>
#define TBUFSIZE        32
#define RBUFSIZE        32

#define TMASK           (TBUFSIZE–1)
#define RMASK           (RBUFSIZE–1)

// Static variables
unsigned char tbuf[TBUFSIZE];  // TX buffer
unsigned char rbuf[RBUFSIZE];  // RX buffer
unsigned char t_in;           // TX buffer in index
unsigned char t_out;           // TX buffer out index
unsigned char r_in;           // RX buffer in index
unsigned char r_out;           // RX buffer out index

SIGNAL(SIG_UART_RECV) {
//завершение приема
        char c;
        c = UDR;
        rbuf[r_in] = c;
        r_in++;
}
SIGNAL(SIG_UART_DATA) {
//завершение передачи
        if(t_in != t_out) {
                UDR = tbuf[t_out & TMASK];
                t_out++;
```

```
        }
        else {
UCSRB &= ~(1<<UDRIE);
        }
}
char tbuflen(void) {
// Retrieve pending chars in TX buffer
        return(t_in – t_out);
}
int UART_putchar(char c) {
// Fills the transmit buffer, if it is full wait
        while((TBUFSIZE – tbuflen()) <= 2);  // Wait...
        // Add data to the transmit buffer, enable TXCIE
        tbuf[t_in & TMASK] = c;
        t_in++;
UCSRB |= (1<<UDRIE);                    // Enable UDR empty interrupt
        return(0);
}
char rbuflen(void) {
// Retrive pending chars in RX buffer
return(r_in – r_out);
}
int UART_getchar(void) {
// Retieves character from UART. This function is to be passed
// to fdevopen
        unsigned char c;
        while(rbuflen() == 0);        // Wait...
        c = rbuf[r_out & RMASK];
        r_out++;
        return(c);
}
void UART_first_init(void) {
        UBRRL = 25;
        UCSRB = (1<<RXCIE)|(1<<TXEN)|(1<<RXEN);
        fdevopen(UART_putchar, UART_getchar, 0);
        sei();
}
int main(void) {
   char s[10];
   int y,x;
   char r[10];
   UART_first_init();                              // First init UART
        for(;;) {
                puts("Input and press ENTER");
```

```
        while(scanf("%s",s) == 0);
    x=atoi(s);
    y=sqrt(x)–23;
    itoa(y,r,10);
        puts(r);
    }
}
```

Порядок выполнения лабораторной работы

Необходимо выполнить все необходимые операции с проектом Lab_8.prj и программой Lab8.c в среде VMLab.

Содержание отчета

Отчет по проделанной лабораторной работе должен содержать:

- структурную схему алгоритма программы на лабораторную работу;
- схему электрическую принципиальную разработанного устройства;
- текст программы на языке СИ (распечатка файла *c);
- распечатка файла проекта (*.prj);
- результаты моделирования (окно VMLab с запущенной программой и результатами вычислений).

Контрольные вопросы

1. 8–ми разрядный таймер/счетчик 0 микроконтроллера ATmega16.
2. Управление T/C0 микроконтроллера ATmega16.
3. 16–ти разрядный таймер/счетчик 1 микроконтроллера ATmega16.
4. Управление T/C1 микроконтроллера ATmega16.
5. Захват, сравнение и широтно–импульсная модуляция в T/C микроконтроллера ATmega16.
6. Часы реального времени RTC (8–ми разрядный T/C2) микроконтроллера AVR ATmega16.
7. Асинхронный последовательный интерфейс UART микроконтроллера ATmega16.
8. Последовательный периферийный интерфейс SPI микроконтроллера ATmega16.
9. 2–проводной последовательный интерфейс *(2–Wire Serial Interface)* или I2C (Inter– *Integrated Circuit)* микроконтроллера ATmega16.
10. Режимы энергосбережения микроконтроллера ATmega16.
11. Технология программирования микроконтроллера ATmega16.

Лабораторная работа № 9

РАЗРАБОТКА ПРОГРАММЫ ВВОДА, ВЫЧИСЛЕНИЯ И ВЫВОДА РЕЗУЛЬТАТА

Цель работы: Освоить основные этапы разработки и отладки программного обеспечения на языке AVR Ассемблера и СИ для микроконтроллера ATmega16 фирмы Atmel.

Задание к лабораторной работе

Задания на разработку программ на языках AVR Ассемблере и СИ выдаются каждому студенту преподавателем.

Необходимо разработать схему микропроцессорного устройства, алгоритм и программу на языках AVR Ассемблере и языке СИ.

Порядок выполнения работы

В соответствии с вариантом задания на лабораторную работу необходимо разработать файлы проектов и программы на языках Ассемблере и СИ.

Содержание отчёта

Отчет по проделанной лабораторной работе должен содержать:

- структурную схему алгоритма программы согласно заданию;
- схему электрическую принципиальную микропроцессорного устройства;
- тексты программ на языках AVR Ассемблере и СИ (распечатка файла *asm и *.c);
- распечатка соответствующего файла проекта (*.prj);
- результаты моделирования (окно VMLab с запущенной программой и результатами вычислений).

Контрольные вопросы

1. Классификация команд микроконтроллера ATmega16.
2. Команды пересылки микроконтроллера ATmega16.
3. Команды пересылки микроконтроллера ATmega16.
4. Арифметические команды микроконтроллера ATmega16.
5. Логические команды микроконтроллера ATmega16.
6. Битовые команды микроконтроллера ATmega16.
7. Команды переходов (инструкции ветвления) микроконтроллера ATmega16.

8. Команды (инструкции) управления микроконтроллера ATmega16.
9. Операнды языка ассемблера AVR.
10. Функции языка ассемблера AVR.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Баранов В.Н. Применение микроконтроллеров AVR: схемы, алгоритм*ы,* программы./ Баранов В.Н. Москва, Додэка–XXI, 2004. – 288 с.1. Гребнев В.В. Микроконтроллеры семейства AVR фирмы Atmel./В.В. Гребнев — М.: ИП РадиоСофт, 2002.–176 с.: ил.

2. Бродин В.Б. Микроконтроллеры. Архитектура, программирование, интерфейс. / Бродин В.Б.– М.: Издательство ЭКОМ, 1999.– 400 с.

3. Голубцов М.С. Микроконтроллеры AVR от простого к сложному./ Голубцов М.С. – М.: СОЛОН–Пресс, 2003. –288 с.

4. Евстифеев А.В. Микроконтроллеры AVR семейств Tiny и Mega фирмы «Atmel» / Евстифеев А.В. – М.: Издательский дом «Додэка–XXI», 2004. – 560 с.

5. Евстифеев А.В.Микроконтроллеры AVR семейства Classic фирмы Atmel./Евстифеев А.В. 3–е издание, стереотипное. – Москва: Додэка–XXI, 2006. – 228 с.

6. Микроконтроллеры семейства Z86 фирмы ZILIG. Руководство программиста. – М., 1999. – 96 с.

7. Однокристальные микроконтроллеры P1C12C5x, P1C12C6x, P1C16x8x, PIC14000, M16C/61/62. / Под ред. Б. Я. Прокопенко.– М.: ДОДЕКА, 2000.– 336 с.9. Трамперт В. Измерение, управление и регулирование с помощью AVR микроконтроллеров./ Трамперт В., пер. с нем. – Киев: МК–Пресс, 2006. – 208 с.

8. Ремизевич Т.В. Микроконтроллеры для встраиваемых приложений: от общих подходов – к семействам HC05 и HC08 фирмы Motorola / под ред. И. С. Кирюхина – М.: ДОДЕКА, 2000.– 272 с.

9. Современные микроконтроллеры: Архитектура, средства проектирования, примеры применения, ресурсы сети Интернет / Под ред. И. В. Коршуна – М.: Издательство «Аким», 1998 – 272 с.

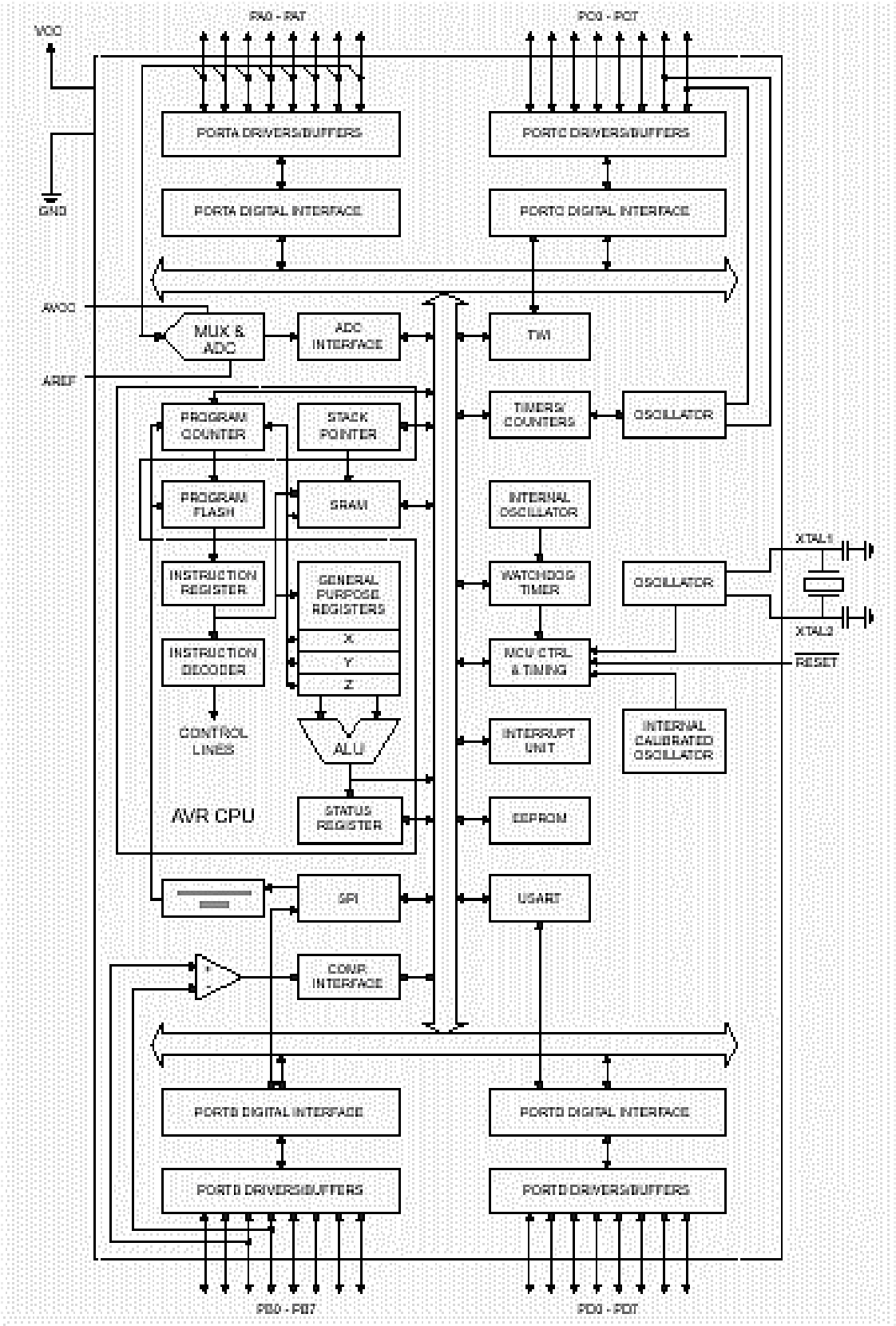
10. Фрунзе А.В. Микроконтроллеры? Это же просто! Т. 1, 2, 3./ Фрунзе А.В – М.: ООО «ИД СКИМЕН», 2003.

11. Шагурин И.И. Микропроцессоры и микроконтроллеры фирмы Motorola: Справ, пособие./ Шагурин И.И.– М.: Радио и связь, 1998.– 560 с.

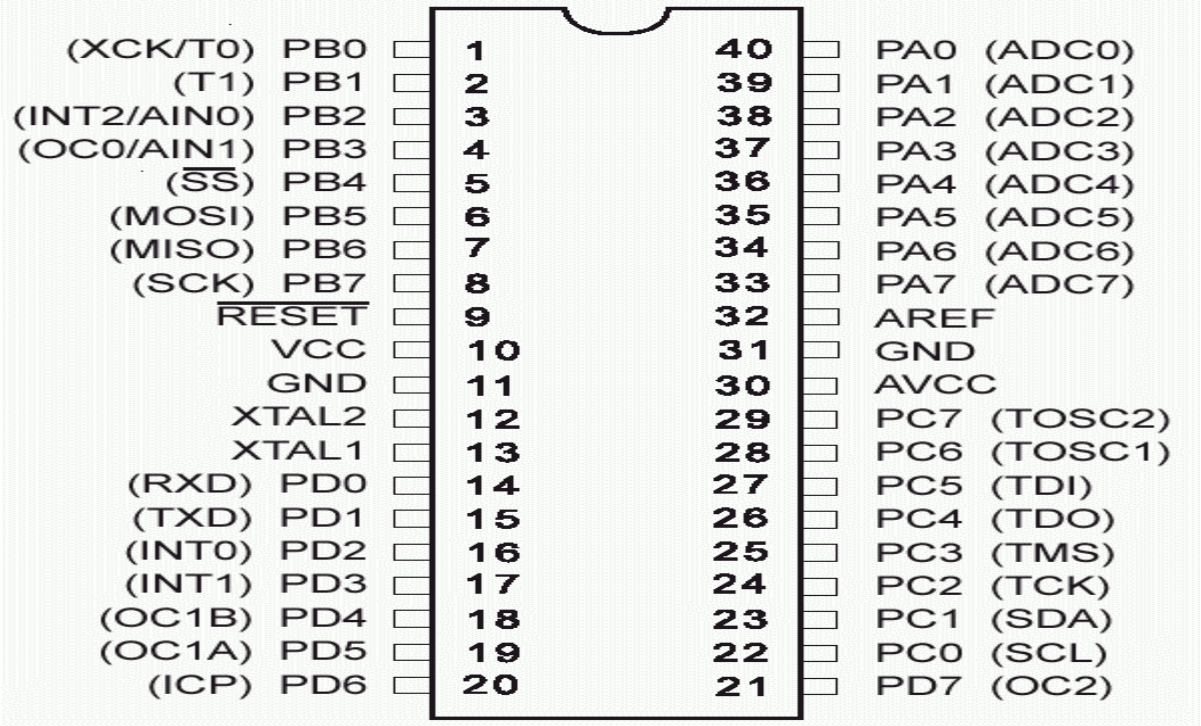
12. Шпак Ю.А. Программирование на языке С для AVR и PIC микроконтроллеров./ Шпак Ю.А. МК–Пресс, Киев: 2006. – 400 с.

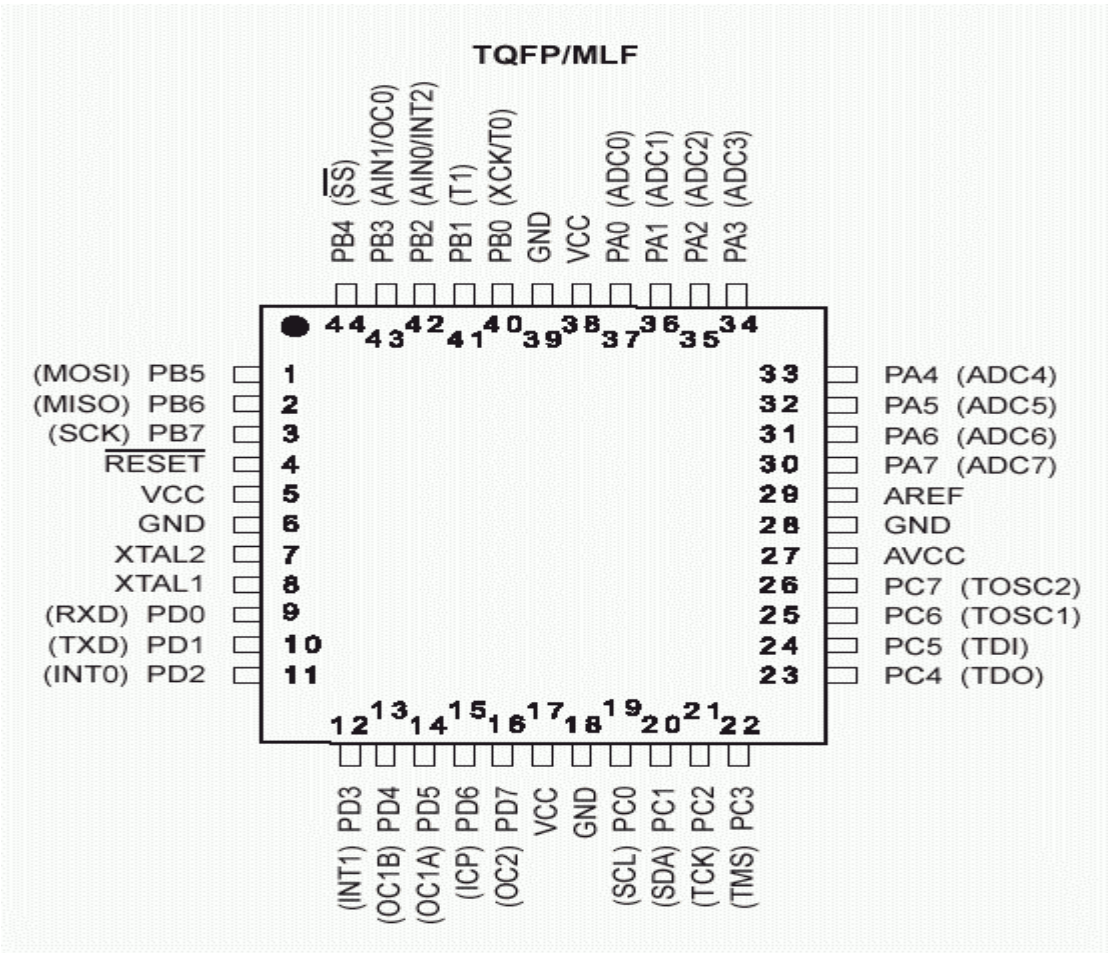
13. Joe Pardue. C Programming for Microcontrollers.. Featuring Atmel's AVR Butterfly and the Free WinAVR Compiler Published by Smiley Micros./ Joe Pardue., 2005. – 300 с.

Приложение 1. Структурная схема микроконтроллера ATmega16



Приложение 2. Типы корпусов и наименование выводов микроконтроллера ATmega16





Приложение 3. Регистры ввода–вывода микроконтроллера ATmega16

I/O адрес (SRAM адрес)	*Имя*	Функция регистра
$3F ($5F)	SREG	*Status Register*
$3E ($5E)	SPH	*Stack Pointer High*
$3D ($5D)	SPL	*Stack Pointer Low*
$3B ($5B)	GIMSK	*General Interrupt Mask Register*
$3A ($5A)	GIFR	*General Interrupt Flag Register*
$39 ($59)	TIMSK	*Timer/Counter Interrupt Mask Register*
$38 ($58)	TIFR	*Timer/Counter Interrupt Flag Register*
$37 ($57)	SPMCR	*SPM Control Register*
$36 ($56)	TWCR	*2–wire Serial Interface Control Register*
$35 ($55)	MCUCR	*MCU general Control Register*
$34 ($54)	MCUSR	*MCU general Status Register*
$33 ($53)	TCCR0	*Timer/Counter0 Control Register*
$32 ($52)	TCNT0	*Timer/Counter0 (8–bit)*
$31 ($51)	OSCCAL	*Oscillator Calibration Register*
$30 ($50)	SFIOR	*Special Function I/O Register*
$2F ($4F)	TCCR1A	*Timer/Counter1 Control Register A*
$2E ($4E)	TCCR1B	*Timer/Counteri1Control Register B*
$2D ($4D)	TCNT1H	*Timer/Counter1 High–byte*
$2C ($4C)	TCNT1L	*Timer/Counter1 Low–byte*
$2B ($4B)	OCR1AH	*Timer/Counter1Output Compare Register A*
$2A ($4A)	OCR1AL	*Timer/Counter1 Output Compare Register A*
$29 ($49)	OCR1BH	*Timer/Counter1 Output Compare Register B*
$28 ($48)	OCR1BL	*Timer/Counter1 Output Compare Register B*
$27 ($47)	ICR1HT/C1	*Input Capture Register High–byte*
$26 ($46)	ICR1LT/C1	*Input Capture Register Low–byte*
$25 ($45)	TCCR2	*Timer/Counter2 Control Register*
$24 ($44)	TCNT2	*Timer/Counter2 (8–bit)*
$23 ($43)	OCR2	*Timer/Counter2 Output Compare Register*
$22 ($42)	ASSR	*Asynchronous Mode Status Register*
$21 ($41)	WDTCR	*Watchdog Timer Control Register*
$20 ($40)	UBRRHI	*UART Baud Rate Register High–byte*
$1F($3F)	EEARH	*EEPROM Address Register High–byte*
$1E($3E)	EEARL	*EEPROM Address Register Low–byte*
$1D($3D)	EEDR	*EEPROM Data Register*
$1C($3C)	EECR	*EEPROM Control Register*
$1B($3B)	PORTA	*Data Register, Port A*
$1A ($3A)	DDRA	*Data Direction Register, Port A*
$19 ($39)	PINA	*Input Pins, Port A*
$18 ($38)	PORTB	*Data Register, Port B*
$17 ($37)	DDRB	*Data Direction Register, Port B*

$16 ($36)	PINB	*Input Pins, Port B*
$15 ($35)	PORTC	*Data Register, Port C*
$14 ($34)	DDRC	*Data Direction Register, Port C*
$13 ($33)	PINC	*Input Pins, Port C*
$12 ($32)	PORTD	*Data Register, Port D*
$11 ($31)	DDRD	*Data Direction Register, Port D*
$10 ($30)	PIND	*Input Pins, Port D*
$0F ($2F)	SPDR	*SPI I/O Data Register*
$0E ($2E)	SPSR	*SPI Status Register*
$0D ($2D)	SPCR	*SPI Control Register*
$0C ($2C)	UDR	*UART I/O Data Register*
$0B ($2B)	UCSRA	*UART Control and Status Register A*
$0A ($2A)	UCSRB	*UART Control and Status Register B*
$09 ($29)	UBRR	*UART Baud Rate Register*
$08 ($28)	ACSR	*Analog Comparator Control and Status Reg-*
$07 ($27)	ADMUX	*ADC Multiplexer Select Register*
$06 ($26)	ADCSR	*ADC Control and Status Register*
$05 ($25)	ADCH	*ADC Data Register High*

Приложение 4. Система команд микроконтроллера ATmega16 и директивы языка Assembler

Команды пересылки AVR–микроконтроллеров

Мнемо-ника	Опранды	Описание инструкции	Выполняемая операция	Фла-ги	Такты
mov	Rd, Rr	Move Between Registers	Rd<– Rr	None	1
mov	Rd, Rr	Copy Register Word	Rd+1:Rd<– Rr+1:Rr	None	1
ldi	Rd*, K	Load Immediate	Rd<– K	None	1
ld	Rd, X	Load Indirect	Rd<– (X)	None	2
ld	Rd, X+	Load Indirect and Post–Inc.	Rd <– (X), X <– X + 1	None	2
ld	Rd,–X	Load Indirect and Pre–Dec.	X<–X–1,Rd<– (X)	None	2
ld	Rd, Y	Load Indirect	Rd <– (Y)	None	2
ld	Rd, Y+	Load Indirect and Post–Inc.	Rd <– (Y), Y <– Y + 1	None	2
ld	Rd,–Y	Load Indirect and Pre–Dec.	Y<–Y–1,Rd<– (Y)	None	2
ldd	Rd, Y+q	Load Indirect with Displace-ment	Rd<– (Y + q)	None	2
ld	Rd, Z	Load Indirect	Rd<– (Z)	None	2
ld	Rd, Z+	Load Indirect and Post–Inc.	Rd <– (Z), Z <– Z+1	None	2
ld	Rd,–Z	Load Indirect and Pre–Dec	Z<– Z–1,Rd<– (Z)	None	2
ldd	Rd, Z+q	Load Indirect with Displacement	Rd<– (Z + q)	None	2
lds	Rd, k	Load Direct from SRAM	Rd<– (k)	None	2
st	X, Rr	Store Indirect	(X)<– Rr	None	2
st	X+, Rr	Store Indirect and Post–Inc.	(X) <– Rr, X <– X + 1	None	2
st	–X, Rr	Store Indirect and Pre–Dec.	X<– X–1,(X)<– Rr	None	2
st	Y, Rr	Store Indirect	(Y) <– Rr	None	2
st	Y+, Rr	Store Indirect and Post–Inc.	(Y) <– Rr, Y <– Y + 1	None	2
st	–Y, Rr	Store Indirect and Pre–Dec.	Y<– Y–1,(Y)<– Rr	None	2
std	Y+q, Rr	Store Indirect withDisplacement	(Y + q)<– Rr	None	2
st	Z, Rr	Store Indirect	(Z) <– Rr	None	2
st	Z+, Rr	Store Indirect and Post–Inc.	(Z) <– Rr, Z <– Z + 1	None	2
st	–Z, Rr	Store Indirect and Pre–Dec	Z<– Z–1,(Z)<– Rr	None	2
std	Z+q,Rr	Store Indirect with Displacement	(Z + q)<– Rr	None	2
sts	k, Rr	Store Direct to SRAM	(k)<– Rr	None	2
lpm		Load Program Memory	R0<– (Z)	None	3
lpm	Rd, Z	Load Program Memory	Rd<–Z)	None	3
lpm	Rd, Z+	Load Program Memory and Post–Inc.	Rd <– (Z), Z=Z+1	None	3
spm		Store Program Memory	(Z)<– R1:R0	None	–
in	Rd, P	In Port	Rd<– P	None	1
out	P, Rr	Out Port	P<– Rr	None	1
pus	Rr	Push Register on Stack	STACK <– Rr; SP<–	None	2
pop	Rd	Pop Register from Stack	SP<– SP+1, Rd<–	None	2

Арифметические и логические команды микроконтроллера *ATmega163*

Мнем.	Опер–ды	Описание	Операция	Флаги	Так
add	Rd, Rr	Add without Carry two Registers	Rd <– Rd + Rr	Z,C,N,V,H	1
adc	Rd, Rr	Add with Carry two Registers	Rd <– Rd + Rr + C	Z,C,N,V	1
adiw	Rdl, K	ADD Immediate from Word	Rdh:Rdl <– Rdh:Rdl+K	Z,C,N,V,S	2
sub	Rd, Rr	Subtract without Carry two Registers	Rd<– Rd–Rr	Z,C,N,V,H	1
subi	Rd*, K	Subtract Constant from Register	Rd<– Rd–K	Z,C,N,V,H	1
sbc	Rd, Rr	Subtract with Carry two Registers	Rd<– Rd–Rr–C	Z,C,N,V,H	1
sbci	Rd*, K	Subtract with Carry Constant from Register	Rd<– Rd–K–C	Z,C,N,V,H	1
sbiw	Rdl, K	Subtract Immediate from Word	Rdh:Rd <– Rdh:Rdl– K	Z,C,N,V,S	2
and	Rd, Rr	Logical AND Registers	Rd <– Rd Rr	Z,N,V	1
andi	Rd*, K	Logical AND Register and Constant	Rd <– Rd K	Z,N,V	1
or	Rd, Rr	Logical OR Registers	Rd <– Rd v Rr	Z,N,V	1
ori	Rd*, K	Logical OR Register and Constant	Rd<– RdvK	Z,N,V	1
eor	Rd, Rr	Exclusive OR Registers	Rd <– Rd Rr	Z,N,V	1
com	Rd	One's Complement	Rd <– $FF – Rd	Z,C,N,V	1
neg	Rd	Two's Complement	Rd<– $00–Rd	Z,C,N,V	1
sbr	Rd*, K	Set Bit(s) in Register	Rd<– RdvK	Z,N,V	1
ser	Rd	Set Register	Rd <– $FF	None	1
cbr	Rd*, K	Clear Bit(s) in Register	Rd <– Rd ($FF – K)	Z,N,V	1
clr	Rd	Clear Register	Rd <– $00	Z,N,V	1
inc	Rd	Increment	Rd <– Rd + 1	Z,N,V	1
dec	Rd	Decrement	Rd<– Rd–1	Z,N,V	1
tst	Rd	Test for Zero or Minus	Rd <– Rd Rd	Z,N,V	1
cp	Rd, Rr	Compare	Rd–Rr	Z, N,V,C,H	1
cpc	Rd, Rr	Compare with Carry	Rd–Rr–C	Z, N,V,C,H	1
cpi	Rd*, K	Compare Register with Immediate	Rd–K	Z, N,V,C,H	1
mul	Rd, Rr	Multiply Unsigned	R1:R0<–RdxRr	Z,C	2
muls	Rd, Rr	Multiply Signed	R1:R0<–RdxRr	Z,C	2
mulsu	Rd, Rr	Multiply Signed with Unsigned	R1:R0<–RdxRr	Z,C	2
fmul	Rd, Rr	Fractional Multiply Unsigned	R1:R0<– (RdxRr)»1	Z,C	2
fmuls	Rd, Rr	Fractional Multiply Signed	R1:R0<– (RdxRr)»1	Z,C	2
fmulsu	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:RCX– (RdxRr)»1	Z,C	2

Битовые команды микроконтроллера *ATmeda163*

Мнемо–ника	Операнды	Описание	Операция	Флаги	Такт
sbi	P*,b	Set Bit in I/O Register	I/O(P,b) <–1	None	2
cbi	P*,b	Clear Bit in I/O Register	I/O(P,b) <– 0	None	2
lsl	Rd	Logical Shift Left	Rd(n+1) <– Rd(n), Rd(0) <– 0	Z,C,N,V	1
lsr	Rd	Logical Shift Right	Rd(n)<–Rd(n+1), Rd(7)<–0	Z,C,N,V	1
rol	Rd	Rotate Left Through Carry	Rd(0)<–C, Rd(n+1)<– Rd(n), C <– Rd(7)	Z,C,N,V	1
ror	Rd	Rotate Right Through Carry	Rd(7) <– C,Rd(n)<– Rd(n+1),C<–Rd(0)	Z,C,N,V	1
rsr	Rd	Arithmetic Shift Right	Rd(n)<–Rd(n+1), n=0..6	Z,C,N,V	1
swap	Rd	Swap Nibbles	Rd(3..0)<–Rd(7..4),	None	1
bset	s	Flag Set	SREG(s) <–1 SREG(s)		1
bclr	s	Flag Clear	SREG(s) <– 0 SREG(s)		1
bld	Rd, b	Bit load from T to Register	Rd(b) <– T	None	1
bst	Rr, b	Bit Store from Register to T	T <– Rr(b)	T	1
sec		Set Carry	C<–1	C	1
clc		Clear Carry	C<–0	c	1
sen		Set Negative Flag	N<–1	N	1
cln		Clear Negative Flag	N<–0	N	1
sez		zSet Zero Flag	Z <–1	z	1
clz		Clear Zero Flag	Z<–0	z	1
sei		Global Interrupt Enable	K–1	I	1
cli		Global Interrupt Disable	<–0	I	1
ses		Set Signed Test Flag	S<–1	s	1
cls		Clear Signed Test Flag	S<–0	s	1
sev		Set Twos Complement Overflow	V<–1	V	1
clv		Clear Twos Complement Overflow	V<–0	V	1
set		SetTinSREG	T<–1	T	1
clt		Clear T in SREG	T<–0	T	1
she		Set Half Carry Flag in SREG	H<–1	H	1
clh		Clear Half Carry Flag in SREG	H<–0	H	1

Команды переходов микроконтроллера *ATmega163*

Мнемо-ника	Операнды	Описание	Операция	Флаги	Такт
rjmp	k	Relative Jump	PC <– PC + k + 1	None	2
ijmp		Indirect Jump to (Z)	PC<–Z	None	2
jmp	k	Jump	PC<–k	None	3
rcall	k	Relative Subroutine Call	PC <– PC + k + 1	None	3
call	k	Call Subroutine	PC<–k	None	4
icall		Indirect Call to (Z)	PC<–Z	None	3
ret		Subroutine Return	PC <– STACK	None	4
reti		Interrupt Return	PC <– STACK	I	4
cpse	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC <– PC + 2 or3	None	1/2/ 3
sbrc	Rr, b	Skip if Bit in Register Cleared	if(Rr(b)=0) PC<–PC + 2 or3	None	1/2
sbrs	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1)PC<–PC + 2 or3	None	1/2
sbic	P*, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC <– PC + 2 or 3	None	1/2
sbis	P*, b	Skip if Bit in I/O Register is Set	if(P(b)=1)PC<–PC + 2or 3	None	1/2
brbs	s, k	Branch if Status Flag Set	if(SREG(s)=1)thenPC <– PC+k + 1	None	1/2
brbc	s, k	Branch f Status Flag Cleared	if(SREG(s) = 0) then PC <– PC+k + 1	None	1/2
breq	k	Branch if Equal	if(Z=1)thenPC<–PC + k+1	None	1/2
brcs	k	Branch if Carry Set	if(C=1)thenPC<–PC + k+ 1	None	1/2
brne	k	Branch if Not Equal	if (Z = 0) then PC <– PC + k+1	None	1/2
brcc	k	Branch if Carry Cleared	if (C = 0) then PC <– PC + k+ 1	None	1/2
brsh	k	Branch if Same or Higher	if (C = 0) then PC <– PC + k+ 1	None	1/2
brlo	k	Branch if Lower	if(C=1)thenPC<–PC + k+1	None	1/2
brmi	k	Branch if Minus	if (N = 1)thenPC<–PC + k+ 1	None	1/2

Команды переходов микроконтроллера *ATmeda163* продолжение

Мнемо-ника	Операн-ды	Описание	Операция	Флаги	Такт
brpl	k	Branch if Plus	if (N = 0) then PC <– PC + k+1	None	1/2
brge	k	Branch if Greater or Equal, Signed	if (N © V= 0) then PC <– PC + k + 1	None	1/2
brlt	k	Branch if Less Than Zero, Signed	if (N®V= 1)thenPC<– PC + k + 1	None	1/2
brhs	k	Branch if Half Carry Flag Set	if(H = 1)thenPC<–PC + k+1	None	1/2
brhc	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC <– PC + k+ 1	None	
brts	k	Branch if T Flag Set	if (T = 1)thenPC<–PC	None	1/2
brtc	k	Branch if T Flag Cleared	if (T = 0) then PC <– PC + k+ 1	None	1/2
brvs	k	Branch if Overflow Flag is Set	if (V = 1)thenPC<–PC + k+1	None	1/2
brvc	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC <– PC + k+1	None	1/2
brie	k	Branch if Interrupt Enabled	if (I = 1)thenPC<–PC + k+ 1	None	1/2
brid	k	Branch if Interrupt Disabled	if (I = 0) then PC <– PC + k+1	None	1/2

Инструкции управления микроконтроллера *ATmeda163*

Мнемоника	Описание	Операция	Такт
пор	No Operation	None	1
sleep	Sleep (see specific de-	None	3
wdr	Watchdog Reset	None	1

Директивы языка AVR Assembler

Директивы	Мнемоника	Наименование
byte	Reserve byte to a variable	Резервирование байта
cseg	Code Segment	Сегмент памяти программ
db	Define constant byte(s)	Опр. однобайтной константы
def	Define a symb. name on a register	Опр. символ. названия регистра
device	Define which device to assemble for	Определение устройства
dseg	Data Segment	Сегмент памяти данных
dw	Define Constant word(s)	Определение слова
endmacro	End macro	Конец макроса
equ	Set a symbol equal to an expression	Задание имени константы
eseg	EEPROM Segment	Сегмент EEPROM
exit	Exit from file	Конец файла
include	Read source from another file	Подключение файла
list	Turn listfile generation on	Включение генерации листинга
listmac	Turn Macro expansion in list file on	Включение макроса в листинг
nolist	Turn listfile generation off	Отключение генерации листинга
org	Set program origin	Задание нач. адреса памяти
set	Set a symbol to an expression	Задание имени выражения

Учебное издание

ТЕХНИКА МИКРОПРОЦЕССОРНЫХ СИСТЕМ
В КОММУТАЦИИ

Сборник лабораторных работ
Часть 1 «Проектирование микропроцессорных систем на базе микроконтроллеров AVR фирмы Atmel»

Составитель: **Горохин** Валерий Николаевич

Редактор М. В.Теленкова

Подписано в печать 30.03.2009. Формат 60×84/16.
Усл. печ. л. 5.81. Уч.–изд. л. 1,20. Тираж 50 экз. Заказ .
Ульяновский государственный технический университет,
432027, Ульяновск, Сев. Венец, 32.

Типография УлГТУ, 432027, Ульяновск, Сев. Венец, 32.