|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство образования и науки Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

В.И. Солнцев

**Устройства управления на базе микроконтроллеров AVR**

Учебное пособие по курсу

**"** **Микроконтроллерные устройства управления"**

Москва

Издательство МГТУ им. Н.Э. Баумана

2022

Содержание

[Содержание 2](#_Toc213597908)

[Описание стенда 3](#_Toc213597909)

[Монтажная схема стенда 7](#_Toc213597910)

[Назначение разъемов 7](#_Toc213597911)

[Описание джамперов 8](#_Toc213597912)

[Описание работы с периферией стенда 11](#_Toc213597914)

[Светодиодные индикаторы 11](#_Toc213597915)

[Зуммер 11](#_Toc213597916)

[Семисегментный индикатор 11](#_Toc213597917)

[Клавиатура 12](#_Toc213597918)

[Жидкокристаллический индикатор HD44780 12](#_Toc213597919)

[Шина I2C 13](#_Toc213597920)

[Часы реального времени DS1307 15](#_Toc213597921)

[Энергонезависимая память данных EEPROM 24C01A 16](#_Toc213597922)

[Цифровой термодатчик DS1820 18](#_Toc213597923)

[Работа со стендом 20](#_Toc213597924)

[Настройка процессора 21](#_Toc213597925)

***Описание работы основных блоков контроллеров***

[Регистр состояния SREG 22](#_Toc213597967)

[Таймеры/счетчики 24](#_Toc213597968)

[АЦП 38](#_Toc213597970)

[Интерфейс SPI 44](#_Toc213597971)

[Интерфейс UART 51](#_Toc213597971)

[Программные средства разработки 60](#_Toc213597927)

[Запуск среды 60](#_Toc213597918)

[Настройка проекта 61](#_Toc213597919)

[Добавление файла 63](#_Toc213597920)

[Написание исходного кода 65](#_Toc213597921)

[Компиляция 65](#_Toc213597921)

[Программирование процессора 67](#_Toc213597922)

[Практическая часть 68](#_Toc213597952)

[Тестовая программа 69](#_Toc213597953)

[Список команд терминала для тестовой программы 70](#_Toc213597954)

[Приложения 71](#_Toc213597957)

[Приложение 1. Архитектура МК 71](#_Toc213597958)

[Приложение 2. Электрическая схема стенда](#_Toc213597959) 75

[Приложение 3. Таблица ASCII кодов 76](#_Toc213597960)

[Приложение 4. Тестовая программа на C 79](#_Toc213597961)

[Приложение 5. Тексты домашних заданий 86](#_Toc213597963)

[Приложение 6. Справочная информация 90](#_Toc213597964)

[Расположение пинов в DIP корпусе контроллеров ATmega 16 90](#_Toc213597965)

[Расположение пинов в DIP корпусе контроллеров ATmega 162 91](#_Toc213597966)

[Описание выводов контроллера ATmega 16 91](#_Toc213597966)

[Описание выводов контроллера ATmega 162 94](#_Toc213597966)

[Список используемой литературы 99](#_Toc213597964)

Раздел

1

Описание стенда

Микроконтроллер (MCU) — [микросхема](http://ru.wikipedia.org/wiki/%D0%9C%D0%B8%D0%BA%D1%80%D0%BE%D1%81%D1%85%D0%B5%D0%BC%D0%B0), предназначенная для управления [электронными](http://ru.wikipedia.org/wiki/%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%B8%D0%BA%D0%B0) устройствами. Типичный микроконтроллер сочетает в себе функции [процессора](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80) и [периферийных устройств](http://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B8%D1%84%D0%B5%D1%80%D0%B8%D0%B9%D0%BD%D1%8B%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%B0), может содержать [ОЗУ](http://ru.wikipedia.org/wiki/%D0%9E%D0%97%D0%A3) и [ПЗУ](http://ru.wikipedia.org/wiki/%D0%9F%D0%97%D0%A3). По сути, это однокристальный [компьютер](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80), способный выполнять простые задачи. Использование одной микросхемы, вместо целого набора, как в случае обычных процессоров, применяемых в персональных компьютерах, значительно снижает размеры, энергопотребление и стоимость устройств, построенных на базе микроконтроллеров.

Микроконтроллеры AVR по праву считаются одним из самых интересных направлений, активно развиваемых корпорацией Atmel. Они представляют собой мощный инструмент для создания современных высокопроизводительных и экономичных многоцелевых контроллеров. На настоящий момент соотношение "цена - производительность - энергопотребление" для микроконтроллеров AVR по-прежнему остается едва ли не лучшим на мировом рынке 8-разрядных микроконтроллеров. Объемы продаж AVR в мире имеют стабильный рост, постоянно растет число сторонних фирм, разрабатывающих и выпускающих программные и аппаратные средства поддержки разработок для них. Области применения AVR многогранны - от простейших игрушек и интеллектуальных датчиков до сложных промышленных систем управления и контроля и современного телекоммуникационного оборудования.

**Почему рекомендуется использовать AVR?**

Во-первых, архитектура Flash-микроконтроллеров AVR 8-bit RISC

является одной из самых удачных на мировом рынке микроконтроллеров. Наличие трех типов памяти на кристалле, высокая производительность, низкое энергопотребление, отличная адаптивность к языкам программирования высокого уровня (С/С++), разнообразные периферийные модули, широкий диапазон напряжений питания и совместимость кристаллов семейства "снизу-вверх" снискали заслуженное уважение и авторитет во всем мире.

Во-вторых, семейство микроконтроллеров AVR поддержано полноценным, профессиональным и доступным набором средств поддержки разработок - как аппаратных, так и программных.

В-третьих, объемы производства и продаж AVR постоянно увеличиваются, что является гарантией их развития и соответствия мировым требованиям, предъявляемым к современным микроконтроллерам. Легкий выбор нужного универсального микроконтроллера для конкретного приложения из относительно небольшого, но сбалансированного семейства AVR, хорошая совместимость микроконтроллеров по коду и расположению выводов, отличное соотношение "цена - производительность - энергопотребление" для 8-разрядных микроконтроллеров являются хорошими аргументами для выбора платформы.

Предлагаемый учебный стенд (рис. 1) предназначен для проведения лабораторных работ по изучению микроконтроллеров, а также решения задач управления, используя микроконтроллеры. Архитектура стенда полностью открыта, порты микроконтроллера выведены на разъёмы. Стенд имеет широкие возможности использования нескольких типов микроконтроллеров фирмы ATMEL, как с CISC, так и RISC архитектурой. Могут быть установлены микроконтроллеры семейства i51 с SPI программированием (AT89S8252 и др.) и семейства ATmega, совместимые по выводам с ATmega16,161,162,163.

ATmega16 является 8-разрядным микроконтроллером, построенным на расширенной AVR RISC архитектуре. За счет выполнения большинства инструкций за один машинный цикл ATmega16 достигает производительности 1 млн. операций в секунду/МГц, что позволяет разработчику эффективно оптимизировать потребление энергии за счёт выбора оптимальной производительности. Микро­контроллеры семейства Mega являются наиболее развитыми представителями микроконтроллеров AVR.

Микроконтроллер ATmega16 имеет следующие аппаратные особенности:

* 8-разрядный высокопроизводительный AVR микроконтроллер с малым потреблением
* Прогрессивная RISC архитектура  
  130 высокопроизводительных команд, большинство команд выполняется за один тактовый цикл  
  32 8-разрядных рабочих регистра общего назначения
* Полностью статическая работа  
  Производительность приближается к 16 MIPS (при тактовой частоте 16 МГц)   
  Встроенный 2-цикловый перемножитель
* Энергонезависимая память программ и данных  
  16 Кбайт внутрисистемное программируемой Flash памяти (In-System Self-Programmable Flash)  
  Обеспечивает 10 000 циклов стирания/записи  
  Дополнительный сектор загрузочных кодов с независимыми битами блокировки  
  Внутрисистемное программирование встроенной программой загрузки  
  Обеспечен режим одновременного чтения/записи (Read-While-Write)  
  512 байт EEPROM  
  Обеспечивает 100 000 циклов стирания/записи  
  1 Кбайт встроенной SRAM  
  Программируемая блокировка, обеспечивающая защиту программных средств пользователя
* Интерфейс JTAG (совместимый с IEEE 1149.1)  
    Возможность сканирования периферии, соответствующая стандарту JTAG  
    Расширенная поддержка встроенной отладки  
    Программирование через JTAG интерфейс: Flash, EEPROM памяти, перемычек и битов блокировки
* Встроенная периферия  
    Два 8-разрядных таймера/счетчика с отдельным предварительным делителем, один с режимом сравнения  
    Один 16-разрядный таймер/счетчик с отдельным предварительным делителем и режимами захвата и сравнения  
    Счетчик реального времени с отдельным генератором  
    Четыре канала PWM  
    8-канальный 10-разрядный аналого-цифровой преобразователь   
    8 несимметричных каналов   
    7 дифференциальных каналов (только в корпусе TQFP)  
    2 дифференциальных канала с программируемым усилением в 1, 10 или 200 крат (только в корпусе TQFP)  
    Байт-ориентированный 2-проводный последовательный интерфейс  
    Программируемый последовательный USART  
    Последовательный интерфейс SPI (ведущий/ведомый)  
    Программируемый сторожевой таймер с отдельным встроенным генератором  
    Встроенный аналоговый компаратор
* Специальные микроконтроллерные функции  
    Сброс по подаче питания и программируемый детектор кратковременного снижения напряжения питания  
    Встроенный калиброванный RC-генератор  
    Внутренние и внешние источники прерываний  
    Шесть режимов пониженного потребления: Idle, Power-save, Power-down, Standby, Extended Standby и снижения шумов ADC
* Выводы I/O и корпуса  
    32 программируемые линии ввода/вывода  
    40-выводной корпус PDIP и 44-выводной корпус TQFP
* Рабочие напряжения  
    2,7 - 5,5 В (ATmega16L)  
    4,5 - 5,5 В (ATmega16)
* Рабочая частота  
    0 - 8 МГц (ATmega16L)  
    0 - 16 МГц (ATmega16)

AVR ядро базируется на усовершенствованной RISC архитектуре, с регистровым файлом быстрого доступа, содержащим 32 регистра общего назначения, непосредственно связанных с арифметико-логическим устройством (ALU), и мощной системой команд. За один тактовый цикл из регистрового файла извлекаются два операнда, выполняется команда и результат записывается в регистр назначения. Такая высокоэффективная архитектура обеспечивает производительность почти в десять раз большую, чем стандартные CISC микроконтроллеры.

В микроконтроллерах AVR реализована Гарвардская архитектура, которая характеризуется раздельной памятью программ и данных, каждая из ко­торых имеет собственные шины доступа к ним. Такая организация позволя­ет одновременно работать как с памятью программ, так и с памятью данных. Разделение шин доступа позволяет использовать для каждого типа памяти шины различной разрядности, причем способы адресации и доступа к каж­дому типу памяти также различны.

Микроконтроллер производится по технологии высокоплотной энергонезависимой памяти компании Atmel. Встроенная внутрисистемно программируемая флэш-память позволяет перепрограммировать память программ непосредственно внутри системы через последовательный интерфейс SPI с помощью простого программатора или с помощью автономной программы в загрузочном секторе. Загрузочная программа может использовать любой интерфейс для загрузки прикладной программы во флэш-память. Программа в загрузочном секторе продолжает работу в процессе обновления прикладной секции флэш-памяти, тем самым поддерживая двухоперационность: чтение во время записи. За счет сочетания 8-разр. RISC ЦПУ с внутрисистемной само программируемой флэш-памятью в одной микросхеме ATmega16 является мощным микроконтроллером, позволяющим достичь высокой степени гибкости и эффективной стоимости при проектировании большинства приложений встроенного управления.

AVR ATmega16 (рис. 2) поддерживается полным набором программ и пакетов для разработки, включая: компиляторы С, макроассемблеры, отладчики/ симуляторы программ, внутрисхемные эмуляторы и наборы для макетирования.

**Микроконтроллер ATmega162.**

Отличительные особенности:

* Высокопроизводительный, 8-ми разрядный AVR микроконтроллер с низким уровнем энергопотребления
* Усовершенствованная RISC архитектура:   
      130 мощных инструкций - большинство выполняются за один такт   
      32 х 8 рабочих регистров общего назначения + регистры управления периферией   
      Полностью статическое функционирование   
      Производительность, вплоть до 8 MIPS при 8 МГц   
      Встроенный 2-х тактный умножитель
* Память данных и программного кода:   
      16 Кбайт Flash- энергонезависимая программная память с поддержкой внутрисистемного программирования, ресурс: 1000 циклов запись/ стирание   
      Опционная память загрузчика кода с независимыми разрядами блокировки, авто-программирование разделов памяти программного кода и данных
* 512 байт энергонезависимой памяти EEPROM с поддержкой внутрисистемного программирования, ресурс: 100 000 циклов записи/ стирания
* 1 Кбайт встроенной SRAM
* Программируемая блокировка для безопасности содержимого Flash и EEPROM
* Периферия:   
      Два 8-ми разрядных таймера/ счетчика с отдельным предварительным делителем частоты и ШИМ   
      Расширенная система с 16-ти разрядным таймером/ счетчиком с отдельным предварительным делителем частоты режимами сравнения и захвата, а также с 8-ми, 9-ти и 10-ти разрядным ШИМ   
      Двухканальный программируемый последовательный UART   
      Ведущий/ ведомый SPI- последовательный интерфейс   
      Контроллер реального времени (RTC) с выделенным тактовым генератором   
      Программируемый следящий таймер с встроенным генератором   
      Встроенный аналоговый компаратор   
      Специализированные функции микроконтроллера:   
      Функции инициализации при включении питания и определение аварийного отключения питания   
      Внешние и внутренние источники прерывания   
      Режимы пониженного энергопотребления:   
        Покоя (Idle), экономичный (Power Save) и отключения (Power Down)
* Функции I/O и корпус:   
      35 программируемых линий I/O, 8 выходных линий, 8 входных линий   
      корпус 40-pin PDIP, 44-pin PLCC и TQFP
* Напряжение питания:   
      От 2.7 В до 5.5 В (ATmega161L), от 4.0 В до 5.5 В (ATmega161)
* Диапазон тактовых частот:   
      От 0 до 4 МГц (ATmega161L)   
      От 0 до 8 МГц (ATmega161)
* Коммерческий и индустриальный диапазоны эксплуатационных температур [ 1, с.10-12 ]

# Рисmet_1

***Рис. 1. Лабораторный стенд***

***Рис. 2. Структурная схема подключения МК***

# Монтажная схема стенда

Монтажная схема стенда представлена на рис. 3.

## Назначение разъемов

**HL1-HL4** – светодиодные индикаторы (зелёные);

**HL5** – светодиодный индикатор питания (красный);

**JP1-JP8** – джамперы;

**U2** – зуммер;

**HG1** – семисегментный индикатор;

**XT1-XT15** – разъёмы стенда.

**Программатор** подключается к разъему **ХТ2**.

**RS-232** интерфейс выведен на разъем **ХТ3**. Дополнительно RS-232 выведен на разъем **ХТ1**.

**Термодатчик DS1820** подключается к **ХТ4**. К этому разъему можно подключить также любое устройство поддерживающее интерфейс ***1‑wire microlan***.

**RS-485** интерфейс выведен на разъем **ХТ5**.

**Клавиатура** подключается через разъем **ХТ13**.

**ЖКИ индикатор** подключается к разъему **ХТ15**.

На разъем **ХТ7** выведены все каналы **АЦП**.

На разъемы **ХТ6**, **ХТ10**, **ХТ12** выведены порты микросхемы **DD5**.

На разъемы **ХТ8**, **ХТ9**, **ХТ11**, **ХТ12**, **ХТ14** выведены порты микросхемы **DD7**.

🖞

**Многие сигналы выведены одновременно на несколько разъемов!**

Подстроечнный резистор **R20** используется для регулировки контрастности **ЖКИ**.

Кнопка **SB1** подает сигнал **RESET** установленной полярности на микроконтроллер.

## Описание джамперов

Джампер **JP1** служит для включения или отключения локального ЭХА для интерфейса **RS-485.**

Джампер **JP2** служит для отключения динамического генератора звука (пищалки).

Джамперы **JP3**, **JP4** служат для выбора типа интерфейсов между **RS‑232** и **RS-485**, в случае если микроконтроллер имеет всего один **UART**.

Джампер **JP5** служит для переключения сигнала -**EA** или подключения светодиода.

Джамперы **JP6**, **JP7** служат для выбора уровня сигнала **RESET**.

Джампер **JP8** служит для выбора одного из двух кварцевых резонаторов, в зависимости от типа устанавливаемого микроконтроллера в позицию **DD7**.

Питание стенда может осуществляться постоянным или переменным напряжением в диапазоне 8-20в. Питание подается на разъем ХТ17.

Батарейка U1 типа CR-2032 используется только для подержания службы времени организованной на базе DS1307. Отсутствие батарейки не влияет на работоспособность стенда.

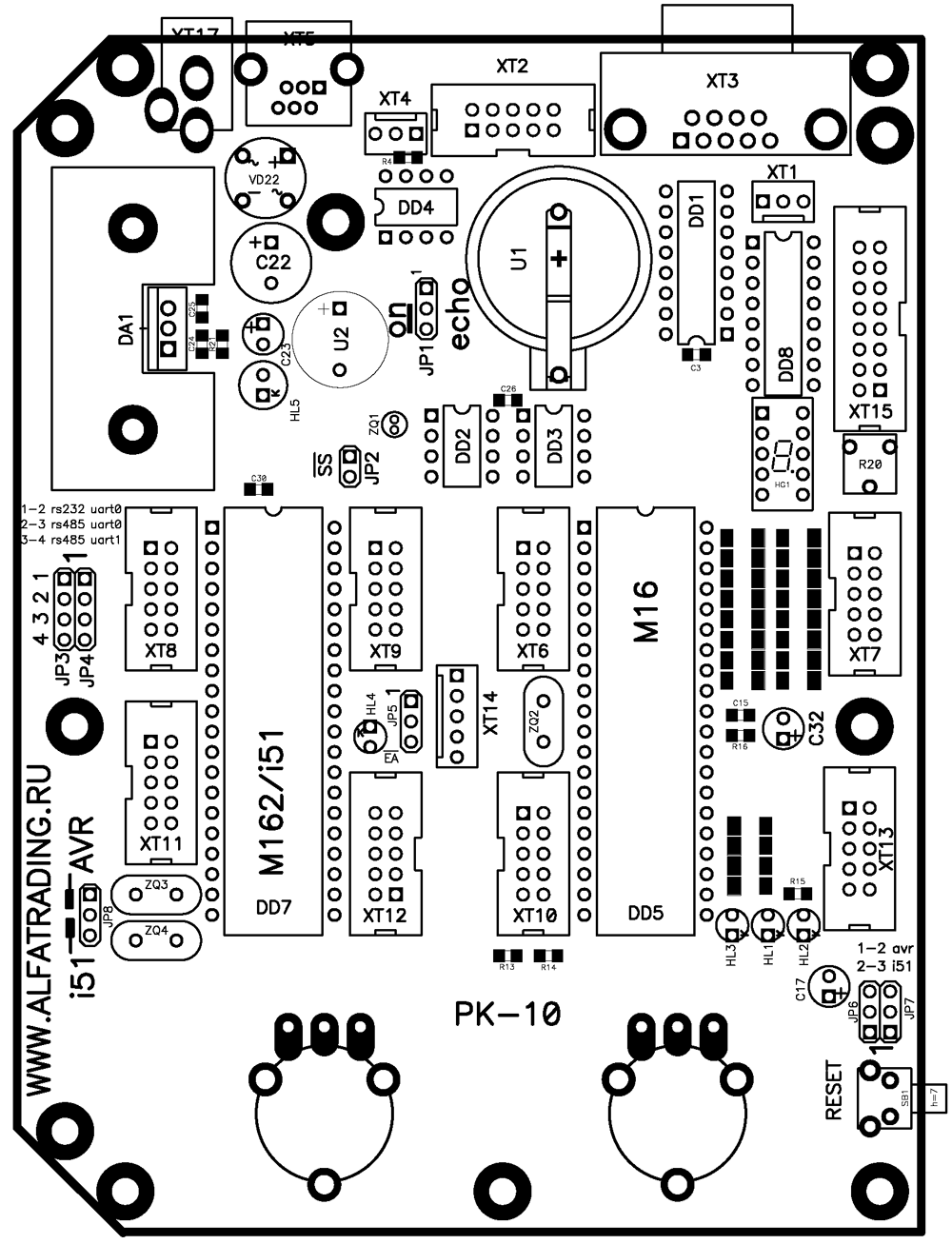


Рис. 3. Монтажная схема стенда

Раздел

2

# Описание работы с периферией стенда

## Светодиодные индикаторы

Светодиодные индикаторы подключены в следующем порядке (слева направо):

LED3 – к **P3.5**

LED1 – к **P1.0**

LED2 – к **P1.1**

Зажигание светодиода происходит, когда на соответствующий вывод порта подаётся 0. Выключение зажженного светодиода происходит, когда на соответствующий вывод порта подаётся 1.

## Зуммер

Зуммер подключен к выводу **P1.4**. Для получения звука надо подавать на зуммер меандр, то есть полпериода подавать на **P1.4** единицу, полпериода – ноль.

## Семисегментный индикатор

Семисегментный индикатор подключён к SPI (последовательный периферийный интерфейс) микроконтроллера через 8-битный сдвиговый регистр *74HC595*, обеспечивающий последовательный ввод и параллельный вывод данных.

Для передачи/считывания данных используется регистр данных **SPDR**.

Для высвечивания определённого символа на семисегментном индикаторе в регистр управления **SPCR** необходимо занести управляющий байт, а в регистр **SPDR** - байт данных соответствующий символу. И после того, как все данные занесены в сдвиговый регистр (проверка регистра статуса **SPSR**) тактирующим сигналом ST (**P3.4**) они подаются на индикатор.

***Табл. 1. Таблица кодов семисегментного индикатора.***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Цифра | A | B | C | D | E | F | G | H |
| «0» | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| «1» | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| «2» | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| «3» | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| «4» | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| «5» | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| «6» | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| «7» | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| «8» | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| «9» | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |

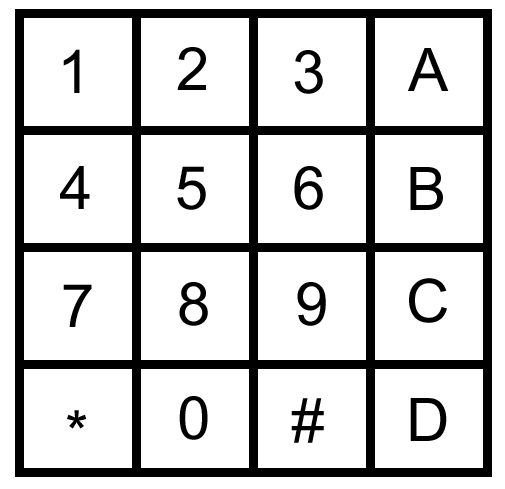


***Рис. 4. Назначение выводов семисегментного индикатора***

Подробное описание интерфейса SPI находится в приложении.

## Клавиатура

Клавиатура (рис.5) состоит из 16 кнопок и подключается к порту 2 (выводы **P2.0** – **P2.7**) (см. рис. клавиатуры на схеме). Клавиатура работает следующим образом. Программно устанавливаются биты **P2.0** – **P2.3** в начальное состояние 1110, затем проверяются биты **P2.4** – **P2.7** на наличие нулей. Если нуль есть (например, комбинация 1011), то это говорит о том, что нажата кнопка, соответствующая состоянию порта 2. (Когда на **P2.0** – **P2.3** подан код «1110», а с **P2.4** – **P2.7** считан код «1011», то нажата кнопка четвёртого столбца второй строки, т.е. кнопка «B».) Если в **P2.4** – **P2.7** одни единицы, то опрашивается следующий столбец (на **P2.0** – **P2.3** выставляется состояние 1101) и опять опрашиваются **P2.4** – **P2.7** на наличие нулей и так далее. На рисунке показано обозначение кнопок клавиатуры стенда.



***Рис. 5. Клавиатура стенда***

## Жидкокристаллический индикатор HD44780

Линии данных **DB0…DB7** жидкокристаллического индикатора подключены к SPI-интерфейсу микроконтроллера (как и семисегментный индикатор) через 8-битный сдвиговый регистр 74HC595, обеспечивающий последовательный ввод и параллельный вывод данных.

Остальные линии подключены к микроконтроллеру следующим образом:

**RS** (выбор регистра) – к **P3.3**

**E** (стробирование/синхронизация) – к **P3.2**

На линию **R\W** (выбор операции) выведена «земля», таким образом режим «чтение с ЖКИ» невозможен и индикатор всё время работает только в режиме «запись».

Для передачи данных через SPI используется регистр данных SPDR.

В регистр управления **SPCR** необходимо занести управляющий байт, а в регистр **SPDR** - байт данных соответствующий символу. И после того, как все данные занесены в сдвиговый регистр (проверка регистра статуса SPSR) тактирующим сигналом ST (**P3.4**) они могут быть поданы на линии данных ЖКИ.

***Табл. 2. Таблица кодов символов для ЖКИ HD44780:***



Пример процедуры вывода информации на ЖКИ:

* + - * 1. Подаём на **RS** уровень 0 (запись команд).
        2. Подаём на **E** уровень 1 (разрешение записи).
        3. На линии данных подаём последовательность команд: 038h, 08h, 01h, 06h, 0Ch.

🖞

* + - * 1. Подаём на **RS** уровень 1 (запись данных).
        2. Выдерживаем документированную паузу;
        3. На линии данных подаём последовательность кодов, выводимых на ЖКИ символов.
        4. Подаём на **E** уровень 0 (запрет записи);
        5. Выдерживаем паузу;

## Шина I2C

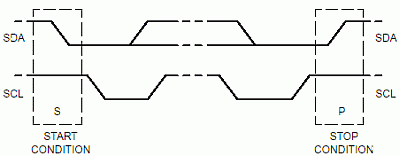
EEPROM 24C01A подключена к I2C-интерфейсу микроконтроллера. I2C шина является одной из модификаций последовательных протоколов обмена данных. Помимо EEPROM к I2C шине подключаются часы реального времени DS1307. Для осуществления процесса обмена информацией по I2C шине, используется всего два сигнала линия данных SDA линия синхронизации SCL.

SDA соответствует **PB3** порта B

SDL соответствует **PB2** порта B

Каждое устройство, подключённое к этой шине, распознается по уникальному адресу и может работать как передатчик или приёмник, в зависимости от назначения устройства. Кроме того, устройства могут быть классифицированы как ведущие и ведомые при передаче данных. В нашем случае ведущим является микроконтроллер, а ведомыми - EEPROM 24C01A и часы реального времени DS1307.

Процедура обмена (см. рис. 6, 7) начинается с того, что ведущий формирует состояние СТАРТ - ведущий генерирует переход сигнала линии **SDA** из ВЫСОКОГО состояния в НИЗКОЕ при ВЫСОКОМ уровне на линии **SCL**. Этот переход воспринимается всеми устройствами, подключенными к шине как признак начала процедуры обмена.

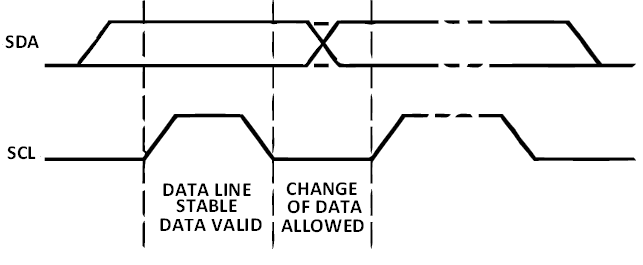


***Рис. 6. Процедура обмена от состояния СТАРТ до состояния СТОП***

Генерация синхросигнала - это всегда обязанность ведущего; каждый ведущий генерирует свой собственный сигнал синхронизации при пересылке данных по шине.

Процедура обмена завершается тем, что ведущий формирует состояние СТОП - переход состояния линии **SDA** из низкого состояния в ВЫСОКОЕ при ВЫСОКОМ состоянии линии **SCL**.

Состояния СТАРТ и СТОП всегда вырабатываются ведущим. Считается, что шина занята после фиксации состояния СТАРТ. Шина считается освободившейся через некоторое время после фиксации состояния СТОП.



***Рис. 7. Процедура обмена***

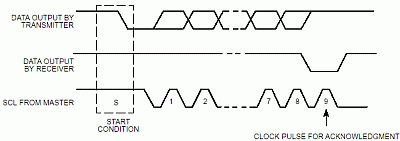
При передаче посылок по шине I2C каждый ведущий генерирует свой синхросигнал на линии SCL.

После формирования состояния СТАРТ, ведущий опускает состояние линии **SCL** в НИЗКОЕ состояние и выставляет на линию **SDA** старший бит первого байта сообщения. Количество байт в сообщении не ограничено.

Спецификация шины I2C разрешает изменения на линии **SDA** только при НИЗКОМ уровне сигнала на линии **SCL**.

Данные действительны и должны оставаться стабильными только во время ВЫСОКОГО состояния синхроимпульса.

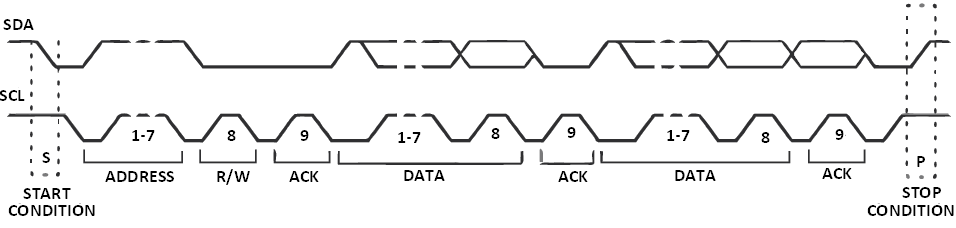
Для подтверждения приема байта от ведущего - передатчика ведомым - приемником в спецификации протокола обмена по шине I2C вводится специальный бит подтверждения, выставляемый на шину **SDA** после приема 8 бита данных (рис. 8).



***Рис. 8. Подтверждение процедуры обмена***

Таким образом передача 8 бит данных от передатчика к приемнику завершаются дополнительным циклом (формированием 9-го тактового импульса линии **SCL**), при котором приемник выставляет низкий уровень сигнала на линии **SDA**, как признак успешного приема байта.

В общем виде процесс обмена по шине от момента формирования состояния СТАРТ до состояния СТОП можно проиллюстрировать на рис. 9:



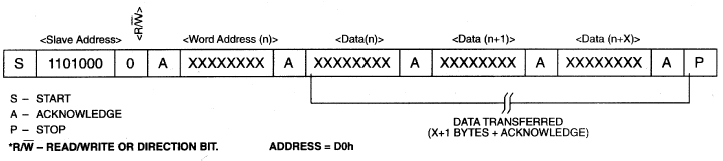
***Рис. 9. Процесс обмена по шине от момента формирования состояния СТАРТ до состояния СТОП***

Первые семь битов первого байта образуют адрес ведомого. Восьмой, младший бит, определяет направление пересылки данных. "Ноль" означает, что ведущий будет записывать информацию в выбранного ведомого. "Единица" означает, что ведущий будет считывать информацию из ведомого. После того, как адрес послан, каждое устройство в системе сравнивает первые семь бит после сигнала СТАРТ со своим адресом. При совпадении устройство полагает себя выбранным как ведомый-приёмник или как ведомый-передатчик, в зависимости от бита направления.

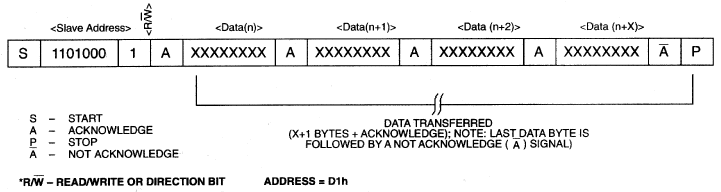
## Часы реального времени DS1307

Для часов реального времени DS1307 адресом устройства является последовательность 1101000.

Временные диаграммы обмена данными по шине представлены на рис. 10 и рис.11. Карта памяти микросхемы DS1307 приведена в табл. 3.



***Рис. 10. Процедура записи данных в DS1307***



***Рис 11. Процедура чтения данных из DS1307***

***Табл.3. Карта памяти микросхемы DS1307***

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | Bit7 | Bit6 | Bit5 | Bit4 | Bit3 | Bit2 | Bit1 | Bit0 |  |
| 00H | CH | 10 seconds | | | seconds | | | | 00-59 |
| 01H | \* | 10 minutes | | | minutes | | | | 00-59 |
| 02H | \* | 12 | 10 HR | 10 HR | hours | | | | 01-12 |
| 24 | A/P |
| 03H | \* | \* | \* | \* | \* | day | | | 00-23  00-23 |
| 04H | \* | \* | 10 date | | date | | | | 1-7 |
| 05H | \* | \* | \* | 10  month | month | | | | 01-28/29  01-30  01-31 |
| 06H | 10 year | | | | year | | | | 01-12 |
| 07H | OUT | \* | \* | SQWE | \* | \* | RS1 | RS0 |  |

|  |  |
| --- | --- |
| 00H | seconds |
| 01H | minutes |
| 02H | hours |
| 03H | day |
| 04H | date |
| 05H | month |
| 06H | year |
| 07H | control |
| 08H  3FH | RAM  56\*8 |

***Рис. 12. Карта памяти DS1307***

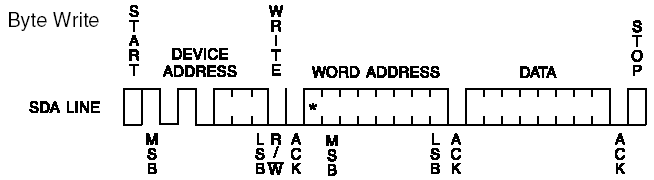
Энергонезависимая память данных **EEPROM** 24C01A

* 1. Устройство ЭСППЗУ EEPROM 24C01A

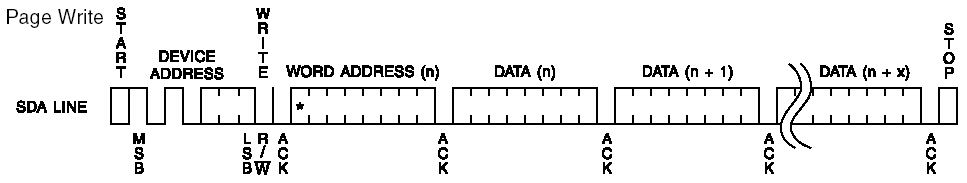
Для EEPROM 24C01A адресом устройства является последовательность **1010000**. Работа с устройством производится после инициализации модуля I2C микроконтроллера.

Временные диаграммы обмена данными по шине I2C приведены на рис. 13 – рис. 17. На этих рисунках звёздочкой отмечен неиспользуемый бит (для EEPROM 24C01A).

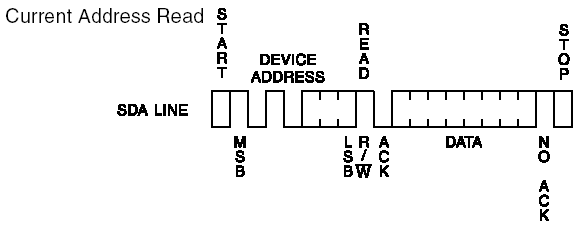
Приведенные временные диаграммы можно наблюдать при подключении осциллографа к соответствующим линиям шины.



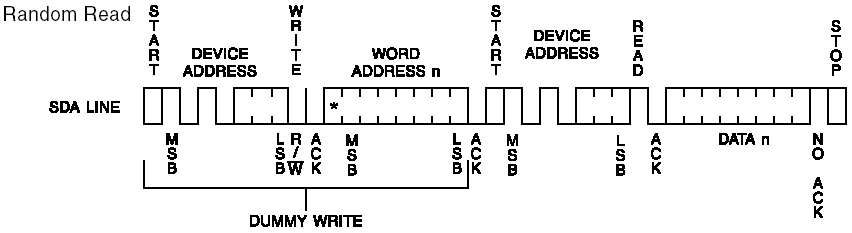
***Рис. 13. Процедура записи байта в EEPROM 24C01A***



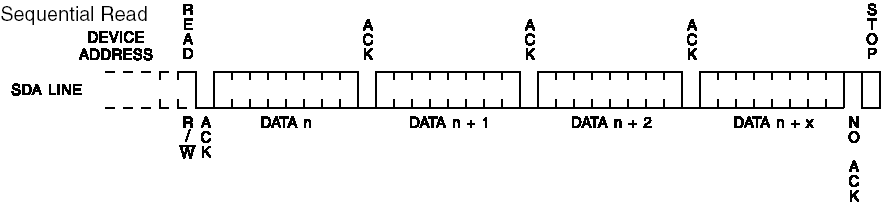
***Рис. 14. Процедура записи страницы в EEPROM 24C01A***



***Рис. 15. Процедура чтения данных по текущему адресу***



***Рис. 16. Процедура чтения данных по произвольному адресу.***



***Рис. 17. Процедура чтения последовательности данных***

*\* звёздочкой отмечен неиспользуемый бит (для EEPROM 24C01A)*

Приведенные временные диаграммы можно наблюдать при подключении осциллографа к соответствующим линиям шины.

## Цифровой термодатчик DS1820

Данные с датчика считываются через 1-проводную последовательную шину (1-WIRE). Эта шина подключена к **P0.2**.

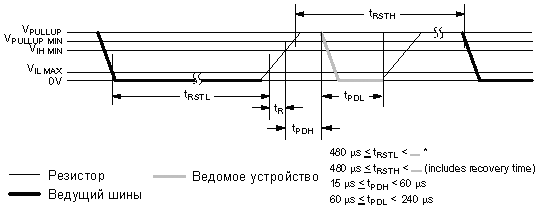
Пример чтения температуры с датчика DS1820:

* + - * 1. Сбрасываем все устройства сети.
        2. Пропускаем импульс присутствия.
        3. Выводим команду "Пропуск ПЗУ" (команда 0CCh).
        4. Выводим команду "Преобразование температуры" (команда 044h).
        5. Ждем, пока температура преобразовывается (в худшем случае 0.5 секунды).
        6. Сбрасываем все устройства сети.
        7. Выводим команду "Пропуск ПЗУ" (команда 0CCh).
        8. Выводим команду "Чтение блокнотной памяти" (команда 0BEh).
        9. Считываем последовательно 9 байт блокнотной памяти (с 0ого по 8ой байт).
        10. Проверяем контрольную сумму у считанных данных (восьмой байт - CRC).

Карта блокнотной памяти DS1820 представлена в табл. 4.

***Табл. 4. Карта блокнотной памяти DS1820*:**

|  |  |
| --- | --- |
| Блокнотная память | Номер байта |
| ТЕМПЕРАТУРА младший байт | 0 |
| ТЕМПЕРАТУРА старший байт | 1 |
| TH/пользовательский байт 1 | 2 |
| TL/пользовательский байт 2 | 3 |
| ЗАРЕЗЕРВИРОВАН | 4 |
| ЗАРЕЗЕРВИРОВАН | 5 |
| COUNT REMAIN | 6 |
| COUNT PER °C | 7 |
| CRC | 8 |

*Импульс сброса* определен как одиночный импульс НИЗКОГО уровня минимальной продолжительностью в восемь временных интервалов (480 мкс) после которого следует ВЫСОКИЙ уровень импульса сброса tRSTH длительностью также 480 мкс. Это состояние ВЫСОКОГО уровня необходимо для того, чтобы приборы на шине могли генерировать *Импульс присутствия*. Временная диаграмма импульса сброса представленна на рис. 18.

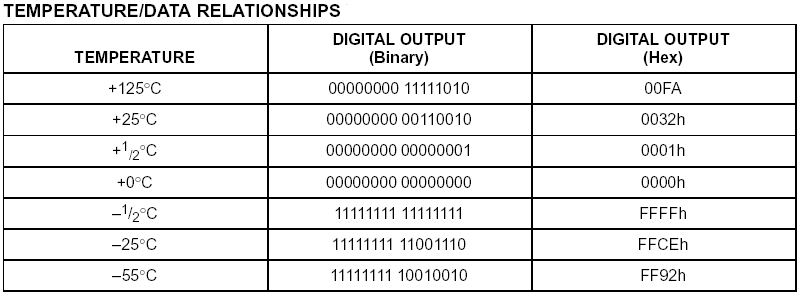
***Рис. 18. Временная диаграмма импульса сброса***

В течение tRSTH никакая другая связь на 1-проводной шине не допускается. *Импульс сброса* предназначен, чтобы обеспечить стартовое условие, которое отменяет любой обмен на шине и возвращает все приборы на шине в исходное состояние. Если ведущий шины посылает *Импульс сброса*, то термодатчик ожидает в течение времени tPDH, и затем генерирует *Импульс присутствия* продолжительностью tPDL. Это позволяет ведущему легко определить, находится ли на шине прибор. Кроме того, если несколько приборов включены параллельно, ведущий может измерять оба интервала времени и таким образом получить информацию о разбросе временных параметров всех приборов на шине.

Обмен на шине начинается с генерации ведущим шины импульса сброса. *Импульс присутствия* указывает ведущему, что на шине присутствует прибор. После этого ведущий шины передает ведомому команду. Далее, в зависимости от команды, ведущий либо читает данные, либо записывает.

После получения датчиком команды "Преобразование температуры" происходит измерение температуры. Результат сохраняется в блокнотной памяти, которую мы считываем после измерения.

Единица в 16-битном числе "ТЕМПЕРАТУРА" соответствует 0.5 градуса. То есть значение 25 будет соответствовать 12.5 °C. Чтобы получить реальную температуру, необходимо это число умножить на 5. Тогда младшая цифра в десятичной системе будет отображать десятые доли температуры. В этом случае 25\*5=125, младшая цифра - десятые, следовательно, температура будет 12.5 °C. Старший байт отвечает только за знак. Соответствие реальных температур и передаваемых по шине данных представлено в табл. 5.

***Табл.5. Таблица соответствия реальных температур с передаваемыми по шине данными:***

Раздел

3

Работа со стендом

В позицию DD5 возможно установить следующие микроконтроллеры: Atmega16, Atmega163, Atmega8535, AT90S8535.

В позицию DD7 возможно установить следующие микроконтроллеры: Atmega162, Atmega161, Atmega8515, AT90S8515, AT89S8252.

⦸

**ЗАПРЕЩАЕТСЯ одновременная установка 2 микроконтроллеров**.

Для удобства ремонта и возможности замены микросхем на аналоги, либо для удаления их с целью отсутствия влияния на подключаемые внешние устройства все микросхемы размещены в панельках.

После установки в соответствующие гнезда процессоров необходимо произвести правильную установку джамперов.

Положения джамперов должны быть следующими:

(*для микроконтроллера Atmega16*):

JP1 – может быть установлен в любое положение

JP2 – установлен если в программе задействован зуммер

JP3, JP4 – установлены в положение 1-2 (для RS-232) или 2-3 (для RS-485)

JP5 – не установлен

JP6, JP7 – установлены в положение 1-2

JP8 – не установлен

*для микроконтроллера Atmega162:*

JP1 – может быть установлен в любое положение

JP2 – установлен если в программе задействован зуммер

JP3, JP4 – установлены в положение 1-2 (для RS-232) или 2-3 (для RS-485)

JP5, JP6, JP7 – установлены в положение 1-2

JP8 – установлен в положение AVR

Питание стенда может осуществляться постоянным или переменным напряжением в диапазоне 8-20в. Питание подается на разъем **ХТ17**.

Батарейка U1 типа CR-2032 используется только для подержания службы времени организованной на базе DS1307. Отсутствие батарейки не влияет на работоспособность стенда.

Порядок подключения:

Подключить программатор к COM-порту персонального компьютера и к разъёму стенда XT2 через шлейф.

Подключить кабель связи к COM-порту компьютера и к разъёму стенда XT3.

Подключить блок питания к сети 220В и разъём питания к разъёму стенда XT17. При этом загорится светодиодный индикатор питания HL5 и микроконтроллер начнёт выполнять программу, прошитую в памяти.

Стенд готов к работе.

## Настройка процессора

Для удобства ремонта и возможности замены микросхем на аналоги, либо для удаления их с целью отсутствия влияния на подключаемые внешние устройства все микросхемы размещены в панельках.

После установки в соответствующие гнезда процессоров необходимо произвести правильную установку джамперов согласно табл. 6.

***Табл. 6. Сводная таблица установки джамперов:***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Микроконтроллер** | **Семейство** | **JP6, JP7** | **JP5** | **JP8** | **JP3, JP4** | |
| ***Atmega16*** | AVR | 1-2 | - | - | 1-2 RS-232 | 2-3 RS-485 |
| ***Atmega162*** | AVR | 1-2 | 1-2 | 1-2 | 1-2 3-4 | |

Знак «**-**» в графе означает, что в данном случае этот джампер ни на что не влияет.

Раздел

4

Описание работы основных блоков контроллеров

# *Регистр состояния.*

# Регистр состояния SREG располагается по адресу $3F (S5F) и содержит набор флагов, показывающих текущее со­стояние микроконтроллера. Большинство флагов автоматически устанав­ливаются в 1 или сбрасываются в 0 при наступлении определенных собы­тий (в соответствии с результатом выполнения команд). Все биты этого ре­гистра доступны как для чтения, так и для записи; после сброса микроконтроллера все биты регистра сбрасываются в 0. Формат этого ре­гистра показан на рис. 19 а его описание приведено в Табл.7

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |
| $3F($5F) | I | T | H | S | V | N | Z | C |  |
| Чтение(R)/Запись(W) | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |  |
| Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |

***Рис.19. Формат регистра состояния SREG***

***Табл. 7. Биты регистра состояния SREG***

|  |  |
| --- | --- |
| Название бита | Описание |
| I | **Общее разрешение прерываний**. Для разрешения прерываний этот флаг должен быть установлен в 1. Разрешение/запрещение от­дельных прерываний производится установкой или сбросом соответствующих битов регистров масок прерываний (регистров управления прерываниями). Если флаг сброшен, то прерывания за­прещены независимо от состояния битов этих регистров. Флаг сбрасывается аппаратно после входа в прерывание и восста­навливается командой RETI для разрешения обработки следующих прерываний |
| T | **Хранение копируемого бита**. Этот бит регистра используется в качестве источника или приемника командами копирования битов BLD (Bit LoaD) и BST (Bit Store). Заданный бит любого РОН может быть скопирован в этот бит командой BST или установлен в соответствии с содержимым данного бита командой BLD. |
| H | **Флаг половинного переноса**. Этот флаг устанавливается *в* 1, если произошел перенос из младшей половины байта (из 3-го бита в 4-й) или заем из старшей половины байта при выполнении неко­торых арифметических операций. |
| S | **Флаг знака**. Этот флаг равен результату операции “Исключающее ИЛИ” (XOR) нежау флагами N (отрицательный результат) и V (переполнение числа в дополнительном коде). Соответственно, этот флаг устанавливается в 1, если результат выполнения арифметической операции меньше нуля. |
| V | **Флаг переполнения дополнительного кода.** Этот флаг устанавливается в 1 при переполнении разрядной сетки знакового результата. Используется при работе со знаковыми числами (представленными в дополнительном коде). |
| N | **Флаг отрицательного значения**. Этот флаг устанавливается в 1*,* если старший бит (7-й) результата операции равен 1. В противном случае флаг равен 0. |
| Z | **Флаг нуля**. Этот флаг устанавливается в 1, если результат выполне­ния операции равен нулю. |
| C | **Флаг переноса.** Этот флаг устанавливается в 1, если в результате выполнения операции произошел выход за границы байта. |

# *Таймеры*

Микроконтроллеры семейства Mega, в зависимости от модели, имеют в своем составе от двух до шести таймеров/счетчиков общего назначения (см. Табл. 8).

***Табл. 8. Таймеры/счетчики общего назначения***

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Таймер/счетчик | ATmega8515x | ATmega8535x | ATmega8x **ATmega16x/**32x | ATmega64x/128x | ATmega48x/88x/168x | **ATmega162x** | ATmega640x/1280x/  1281x/2560x/2561x |
| Таймер/счетчик Т0 |  |  | **** |  |  | **** |  |
| Таймер/счетчик Т1 |  |  | **** |  |  | **** |  |
| Таймер/счетчик Т2 |  | 1) | **1)** |  | 1) | **1)** | 1) |
| Таймер/счетчик Т3 |  |  |  |  |  | **** |  |
| Таймер/счетчик Т4 |  |  |  |  |  |  |  |
| Таймер/счетчик Т5 |  |  |  |  |  |  |  |
| 1) Асинхронный таймер/счётчик | | | | | | | |

Как видно из таблицы, во всех моделях микроконтроллеров семейства присутствуют как минимум два таймера/счётчика- T0 и Т1.Таймер/счётчик Т0 имеет минимальный набор функций, зависящий, тем не менее, от модели микроконтроллера. В одних моделях он может использоваться только для отсчета и измерении временных интервалов или как счетчик внешних событий, В других моделях к этим функциям добавляется воз­можность генерации сигналов с широтно-импульсной модуляцией (ШИМ) фиксированной разрядности (один или два канала), а также воз­можность работать в асинхронном режиме в качестве часов реального вре­мени (вмоделяхАТтедаб4х/128х).

Таймер/счетчик T1 тоже может использоваться для отсчета временных интервалов и как счетчик внешних событий. Кроме того, он может осу­ществлять запоминание своего состояния по внешнему сигналу. Как и таймер/счетчик ТО, аи может работать в качестве 2- или 3-канального широтно-импульсного модулятора, но уже переменной разрядности. Коли­чество каналов ШИ М зависит от модели.

Таймер/счетчик Т2 практически полностью аналогичен таймеру/счет­чику ТО. Во всех моделях, кроме ATmega64x/128x, таймер/счетчик Т2 мо­жет работать в асинхронном режиме.

Таймеры/счетчики ТЗ...Т5 по функциональным возможностям иден­тичны таймеру/счетчику Т1.

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

## Назначение выводов таймеров/счетчиков

Каждый таймер/счетчик использует один или более выводов микро­контроллера. Как правило, эти выводы — линии портов ввода/вывода об­щего назначения, а функции, реализуемые этими выводами при работе совместно с таймерами/счетчиками, являются их альтернативными функ­циями.

Все выводы микроконтроллеров, используемые таймерами/счетчика-мн общего назначения, приведены в табл. 9. Там же указаны функции этих выводов.

Не забывайте о том, что при использовании альтернативных функций линий портов ввода/вывода необходимо, как правило, самостоятельно сконфигурировать эти выводы в соответствии с их функциональным на­значением.

***Табл. 9. Выводы, используемые таймерами/счетчиками общего назначения***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Название | ATmega8515x | ATmega8535x | ATmega8x | **ATmega16x/32x** | ATmega64x/128x | ATmega48x/88x/168x | **ATmega162x** | Описание |
| Т0 | PB0 | PB0 | PD4 | PB0 | - | РD4 | РВ0 | Вход внешнего сигнала таймера Т0 |
| OC0 | PB0 | PB3 | - | PB3 | PB4 | - | РВ0 | Выход схемы сравнения таймера T0 |
| OC0A | - | - | - | - | - | PD6 | - |
| OC0B | - | - | - | - | - | PD5 | - |
| T1 | PB1 | PB1 | PD5 | PB1 | PD6 | PD5 | PB1 | Вход внешнего сигнала таймера Т1 |
| ICP | PE0 | - | - | - | - | - | - | Вход захвата таймера0 T1 |
| ICP1 | - | PD6 | PB0 | PD6 | PD4 | PB0 | PE0 |
| OC1A | PD5 | PD5 | PB1 | PD5 | PB5 | РB1 | PD5 | Выход схемы сравнения таймера T1 |
| OC1B | PE2 | PD4 | PB2 | PD4 | PB6 | РB2 | PE2 |
| OC1C | - |  | - | - | PB7 | - | - |
| T2 | - |  | - | - | PD7 | - | - | Вход внешнего сигнала таймера Т2 |
| OC2 | - | - | PB3 | PD7 | PB7 | - | PB1 | Выход схемы сравнения таймера T2 |
| OC2A | - | - | - | - | - | PB3 | - |
| OC2B | - | - | - | - | - | PD3 | - |
| T3 | - | - | - | - | PE6 | - | - | Вход внешнего сигнала таймера Т3 |
| ICP3 | - | - |  |  | PE7 | - | PD3 | Вход захвата таймера T3 |
| OC3A | - | - | - | - | PE3 | - | PD4 | Выход схемы сравнения таймера T3 |
| OC3B | - | - | - | - | PE4 | - | PB4 |
| OC3C | - | - | - | - | PE5 | - | - |
| T4 | - | - | - | - | - | - | - | Вход внешнего сигнала таймера Т4 |
| ICP4 | - | - | - | - | - | - | - | Вход захвата таймера T4 |
| OC4A | - | - | - | - | - | - | - | Выход схемы сравнения таймера T4 |
| OC4B | - | - | - | - | - | - | - |
| OC4C | - | - | - | - | - | - | - |
| T5 | - | - | - | - | - | - | - | Вход внешнего сигнала таймера Т5 |
| ICP5 | - | - | - | - | - | - | - | Вход захвата таймера T5 |
| OC5A | - | - | - | - | - | - | - | Выход схемы сравнения таймера T4 |
| OC5B | - | - | - | - | - | - | - |
| OC5C | - | - | - | - | - | - | - |
| TOSC1 | - | PC6 | PB6 | PC6 | PG4 | PB6 | PD4 | Вход для подключения резонатора |
| TOSC2 | - | PC7 | PB7 | PC7 | PG3 | PB7 | PD3 | Вход для подключения резонатора |

***Прерывания от таймеров счётчиков***

Для разрешения/запрещения прерывания от таймеров/счётчиков используется от 3-6 регистров ввода/вывода, которые равны числу счётчиков в конкретной модели. Точно так же дело обстоит и с регистрами, содержащими флаги прерываний. Названия и адреса всех эти регистров приведены в табл. 10.

***Табл. 10. Регистры для управления прерывания от таймеров/счётчиков***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Модель | Таймер/счётчик | Разрешение прерываний | | Флаги прерываний | |
| Регистр | Адрес | Регистр | Адрес |
| Atmega 16x | T0, T1, T2 | TIMSK | $39($59) | TIFR | $38($58) |
| Atmega 162x | T0, T1, T2 | TIMSK | $39($59) | TIFR | $38($58) |
| T1, T3 | ETIMSK | ($7D) | ETIFR | ($7C) |

Форматы регистров, используемых для разрешения/запрещения прерывания от таймеров/счётчиков показаны на рис. 20, а описание их битов приведено в табл. 11.

Для разрешения какого-либо прерывания от таймера/счётчика необходимо установить в 1 соответствующий бит регистра TIMSK (TIMSKn, где n=0,1,2,3)/ETIMSK и, разумеется, флаг 1 регистра SREG.

***Табл. 11. Биты регистров TIMSK, ETIMSK и TIMSK0…..TIMSK3***

|  |  |
| --- | --- |
| Название бита | Описание |
| TOIEn | Флаг разрешения прерывания по переполнению таймера/счётчика Tn(n=0-3) |
| OCIEn | Флаг разрешения прерывания по событию ”Совпадение” таймера/счётчика Tn (n=0,2) |
| OCIEnA | Флаг разрешения прерывания по событию ”Совпадение A” таймера/счётчика Tn (n=0-3) |
| OCIEnB | Флаг разрешения прерывания по событию ”Совпадение B” таймера/счётчика Tn (n=0-3) |
| OCIEnC | Флаг разрешения прерывания по событию ”Совпадение C” таймера/счётчика Tn (n=0-3) |
| TICIEIn | Флаг разрешения прерывания по событию ”Захват” таймера/счётчика» Tn (n=0,2) |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  | |  | OCIE2 | TOIE2 | TICIE1 | OCIE1A | OCIE1B | TOIE1 | OCIE0 | TOIE0 | Atmega16x | | Чтение(R)/Запись(W) | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |  | | Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  | |  |  |  |  |  |  |  |  |  |  | |  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  | |  | TOIE1 | OCIE1A | OCIE1B | OCIE2 | TICIE1 | TOIE2 | TOIE0 | OCIE0 | Atmega162x | | Чтение(R)/Запись(W) | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |  | | Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  | |
|  |

***Рис. 20. Формат регистров TIMSK***

Форматы регистров, используемых для индикации наступления прерываний от таймеров/счётчиков, показаны на рис. 21, а описание их битов приведено в табл. 12.

***Табл. 12. Биты регистров TIFR***

|  |  |
| --- | --- |
| Название бита | Описание |
| TOVn | Флаг прерывания по переполнению таймера/счётчика Tn(n=0-3) |
| OCFn | Флаг прерывания по событию ”Совпадение”таймера/счётчика Tn(n=0,2) |
| OCFnA | Флаг прерывания по событию ”Совпадение A”таймера/счётчика Tn(n=0-3) |
| OCFnB | Флаг прерывания по событию ”Совпадение B”таймера/счётчика Tn(n=0-3) |
| OCFnC | Флаг прерывания по событию ”Совпадение C”таймера/счётчика Tn(n=0-3) |
| ICFn | Флаг разрешения прерывания по событию ”Захват”таймера/счётчика Tn(n=0-3) |

|  |
| --- |
|  |
| |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  | |  | OCF2 | TOV2 | ICF1 | OCF1A | OCF1B | TOV1 | OCF0 | TOV0 | Atmega16x | | Чтение(R)/Запись(W) | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |  | | Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  | |  |  |  |  |  |  |  |  |  |  | |  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  | |  | TOV1 | OCF1A | OCF1B | OCF2 | ICF1 | TOV2 | TOV0 | OCF0 | Atmega162x | | Чтение(R)/Запись(W) | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |  | | Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  | |
| ***Рис. 21. Формат регистров TIFR*** |  |  |  |  |  |  |  |  |  |

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

***Управление предделителями***

Помимо управления тактовым сигналом таймера/счетчика, все мик­роконтроллеры семейства позволяют осуществлять сброс предделителей, а отдельные модели позволяют также осуществлять их остановку. Для этого используется либо регистр специальных функций SFIOR, расположенный по адресу $23 ($43). Формат этого регистра для различ­ных моделей микроконтроллеров приведен на рис. .22 (биты, не исполь­зуемые для управления предделителями таймеров/счетчиков, указаны на рисунке как X).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |
|  | X | X | X | X | X | X | PSR2 | PSR 10 | Atmega16x |
| Чтение(R)/Запись(W) |  |  |  |  |  |  | R/W | R/W |  |
| Начальное значение |  |  |  |  |  |  | 0 | 0 |  |
|  |  |  |  |  |  |  |  |  |  |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |
|  | TSM | X | X | X | X | X | PSR 2 | PSR310 | Atmega162x |
| Чтение(R)/Запись(W) | R/W |  |  |  |  |  | R/W | R/W |  |
| Начальное значение | 0 |  |  |  |  |  | 0 | 0 |  |

***Рис. 22. Формат регистров SFIOR***

Для сброса предделителей таймеров/счетчиков используются биты PSRx регистра. При записи в эти биты лог. 1 предделители соответствующих таймеров/счетчиков переводятся в ис­ходное состояние. Биты сбрасываются в 0 аппаратно после выполне­ния операции сброса. Напоминаю, что один предделитель, как прави­ло, используется несколькими таймерами/счетчиками, и соответствен­но сброс предделителя повлияет на все таймеры/счетчики, которые его используют.

Остановка всех предделителей микроконтроллера осуществляется за­писью лог. 1 вбит TSM регистра SFSOR или GTCCR. Последующий запуск предделителей осуществляется записью в бит TSM лог. 0. Указанная функция может использоваться, в частности, для синхронизации таймеров/счетчиков. После установки бита TSM и битов PSRx соответствующие таймеры/счетчики останавлива­ются и могут быть проинициализированы требуемыми значениями. После сброса бита TSM биты PSRx аппаратно сбрасывают­ся и все таймеры/счетчики начинают работать одновременно.

***Восьмибитные таймеры счётчики***

Таймер/счётчик T0, T2 используются для отсчёта временных интервалов или как счётчики внешних событий, а также в качестве одноканальных генератора 8-битного ШИМ-сигнала. Таймер/счётчик T2 отличается от Т0 тем, что может работать в асинхронном режиме (обычно этот режим используется для реализации часов реального времени).

***Табл. 13. Функции таймера/счётчика T0***

|  |  |  |  |
| --- | --- | --- | --- |
| Функции | 8-битный счётчик | Счётчик внешних событий | ШИМ, число каналов |
| Atmega 16x |  |  | 1 |
| Atmega 162x |  |  | 1 |

***Табл. 14. Функции таймера/счётчика T2***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Функции | 8-битный  счётчик | Счётчик внешних  событий | ШИМ, число  каналов | Часы реального времени |
| Atmega 16x |  | **-** | 1 | 1 |
| Atmega 162x |  | **-** | 1 | 1 |

***Табл. 15. Регистры 8-битных таймеров/счётчиков***

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Регистр | TCCR0 | TCNT0 | OCR0 | TCCR2 | TCNT2 | OCR2 | ASSR |
| Atmega 16x |  |  |  |  |  |  |  |
| Atmega 162x |  |  |  |  |  |  |  |
| Назначение | Регистр управления | Счётный регистр | Регистр сравнения | Регистр управления | Счётный регистр | Регистр сравнения | Регистр  состояния  асинхронного  режима |

Счетный регистр таймера/счетчика TCNTn входит в состав основно­го блока модуля — блока реверсивного счетчика. В зависимости от режи­ма работы модуля содержимое счетного регистра сбрасывается, инкрементируется или декрементируется по каждому импульсу тактового сиг­нала таймера/счетчика clkТ0). Независимо от того, присутствует тактовый сигнал или нет, регистр доступен в любой момент времени как для чтения, так и для записи. Однако следует помнить, что любая опера­ция записи в счетный регистр блокирует работу блока сравнения на вре­мя одного периода тактового сигнала таймера/счетчика. После подачи напряжения питания в регистре TCNTn находится нулевое значение. При достижении таймером/счетчиком максимального или минимально­го значения (конкретный вариант зависит от его режима работы) устанавливается флаг TOVn в регистре флагов TIFR (TIFRn). Разрешение прерывания осуществляется установкой в 1 бита ТОIЕn регистра маски TIMSK (TIMSKn). Разумеется, флаг I регистра SREG также должен быть установлена 1.

Регистры сравнения ОСRn (OCRnA/OCRnB) входят в состав бло­ков сравнения модуля. Во время работы таймера/счетчика производит­ся непрерывное (в каждом такте) сравнение этих регистров с регистром TCNTn.В случае равенства содержимого этих регистров в следующем такте устанавливается флаг OCFn (OCFnA/OCFnB) в соответствующем регистре флагов и генерируется прерывание (если оно разрешено). Кроме того, при наступлении этого события может изменяться состоя­ние вывода OCn (ОСnА/ОСnВ) микроконтроллера. Чтобы таймер/счетчик мог управлять состоянием этих выводов, они должны быть сконфигурированы как выходы (соответствующий бит регистра DDRx должен быть установлен в 1).

Напоминаю, что любая операция записи в счетный регистр блокиру­ет формирование сигнала о совпадении, если оно произойдет в следую­щем такте.

Регистры TCCRn (ТССRnА/ТССRnВ) предназначены для управ­ления модулем таймера/счетчика. Формат этих регистров приведен на рис. .23**,** а описание их битов — соответственно в табл. 16.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |
|  | FOC0 | WGM00 | COM01 | COM00 | WGM01 | CS02 | CS01 | CS00 | Atmega16x |
| Чтение(R)/Запись(W) | W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |  |
| Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |
|  |  |  |  |  |  |  |  |  |  |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |
|  | FOC2 | WGM20 | COM21 | COM20 | WGM21 | CS22 | CS21 | CS20 | Atmega162x |
| Чтение(R)/Запись(W) | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |  |
| Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |

***Рис. 23. Формат регистров TCCR0 и TCCR2***

***Табл. 16. Биты регистров TCCR0 и TCCR2***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Название бита | Описание | | | |
| FOCn | **Принудительное изменение состояния вывода OCn (режимы Normal и СТС).**  При записи лог.1 в этот бит состояние вывода OCn изменяется в соответствии с установками битов COMn1: COMn0. Прерывание при этом не генерируется и сброс таймера (в режиме СТС) не производится. В режимах FAST PWM и Phase Correct PWM этот бит должен быть сброшен в 0.При чтении бита всегда возвращается 0. | | | |
| WGMn1:  WGMn0 | **Режим работы таймера/счётчика.** Эти биты определяют режим работы таймера/счётчика следующим образом: | | | |
| **Номер режима** | **WGMn1** | **WGMn0** | **Режим работы таймера/счётчика Tn** |
| 0 | 0 | 0 | Normal |  |  |  |
| 1 | 0 | 1 | Phase correct PWM |
| 2 | 1 | 0 | СТС(сброс при совпадении) |
| 3 | 1 | 1 | FAST PWM |
| COMn1:  COMn0 | **Режим работы блока сравнения.** Эти биты определяют поведение вывода OCn при наступлении события” Совпадение”. Влияние содержимого этих битов на состояние вывода зависит от режима работы/таймера счётчика. | | | |
| CSn2…CSn0 | **Управление тактовым сигналом.** Эти биты определяют источник тактового сигнала таймера/счётчика. Действие этих битов зависит от исполнения таймера/счётчика и будет описано ниже. | | | |

***Управление тактовым сигналом***

Формирование тактового сигнала таймера/счетчика clkT0 (clkT2) осу­ществляется блоком предделителя, который был рассмотрен выше.

В качестве тактового сигнала clkTO (clkT2) таймеров/счетчиков, не име­ющих асинхронного режима, может использоваться:

* системный тактовый сигнал (clkTO(T2) =clkI/O );
* масштабированный системный тактовый сигнал (clkTO(T2) =clkI/O /n);
* внешний сигнал, поступающий на вход ТО (Т2) микроконтроллера (clkTO(T2) =clkI/O /ext);

Тактовый сигнал таймеров/счетчиков с асинхронным режимом может формироваться либо из системного тактового сигнала clkI/O (clkTO(T2) =clkI/O /n), либо — в асинхронном режиме — из сигнала от допол­нительного кварцевого резонатора ((clkTO(T2) =clkTOSC! /n)). Переключение между синхронным и асинхронным режимами работы осуществляется с помощью бита ASO (AS2) регистра ASSR.

Выбор источника тактового сигнала, а также запуск и остановка тай­меров/счетчиков осуществляются с помощью битов CS02...CSOO (CS21..CS20) регистров управления таймером TCCRn (TCCRnA/TCCRnB) согласно табл. 17.

**Табл. 17***.* **Выбор источника тактового сигнала таймеров/счетчиков ТО и Т2.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **CSn2** | **CSn1** | **CSn0** | **Источник тактового сигнала(n=0,2)** | | |
| **Обычный**  **таймер/счётчик** | **Асинхронный таймер/счётчик** | |
| **ASn=0** | **ASn=1** |
| 0 | 0 | 0 | Таймер/счётчик остановлен | Таймер/счётчик остановлен | |
| 0 | 0 | 1 | clkI/O | clkI/O | ClkTOSC1 |
| 0 | 1 | 0 | clkI/O**/**8 | clkI/O**/**8 | clk TOSC1**/**8 |
| 0 | 1 | 1 | clkI/O**/**64 | clkI/O**/**32 | clk TOSC1**/**32 |
| 1 | 0 | 0 | clkI/O**/**256 | clkI/O**/**64 | clk TOSC1**/**64 |
| 1 | 0 | 1 | clkI/O**/**1024 | clkI/O**/**128 | clk TOSC1**/**128 |
| 1 | 1 | 0 | Вывод Tn, счёт осуществляется по спадающему фронту импульсов | clkI/O**/**256 | clk TOSC1**/**256 |
| 1 | 1 | 1 | Вывод Tn, счёт осуществляется по нарастающему фронту импульсов | clkI/O**/**1024 | clk TOSC1**/**1024 |

***Режимы работы***

Режим работы таймера/счетчика ТО (Т2) Определяется состоянием битов WGMn2:WGMn0 регистра ТССRn (TCCRnA/TCCRnB). Зависимость режима работы таймеров/счетчиков от состояния этих битов показана в табл. 18.

***Табл. 18. Режимs работы таймеров/счётчиков( n=0,2)***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Номер режима | WGMn1 | WGMn0 | Режим работы таймера/счетчика Tn | Модуль счёта(TOP) | Обновление регистров OCRnx | Момент установки флага TOVn |
| 0 | 0 | 0 | Normal | $FF | Немедленно | $FF |
| 1 | 0 | 1 | Phase correct PWM | $FF | При TOP | $00 |
| 2 | 1 | 0 | СТС(сброс или совпадении) | OCRn(OCRnA) | Немедленно | $FF |
| 3 | 1 | 1 | Fast PWM | $FF | При TOP | $FF |

***16-битные таймеры счётчики***

Количество 16-битных таймеров (см. табл. 19) зависит от модели. Контроллер ATmega 16 имеет один таймер/счётчик T1,в модели ATmega 162 уже два 16-битных таймера/счётчика-T1,T3. Они используются для формирования временных интервалов, для подсчёта числа внешних событий, формирования сигналов и генерации сигналов с ШИМ.В дополнение к этому 16-битные таймеры/счётчики могут по внешнему сигналу сохранять своё текущее состояние в отдельном регистре ввода/вывода.

Каждый 16-битный регистр физически размещается в двух регистрах ввода/вывода, названия которых получаются путём добавления к названию регистра буквы” H” (старший байт) и” L” (младший байт).

***Табл. 19. Регистры 16-битных таймеров/счётчиков***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Регистр | TCCR1A | TCCR1B | TCNT1 | OCR1A | OCR1B | ICR1 |
| Atmega 16x |  |  |  |  |  |  |
| Atmega 162x |  |  |  |  |  |  |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Регистр | TCCR3A | TCCR3B | OCR3A | OCR3AB | ICR3 |
| Atmega 16x |  |  |  |  |  |
| Atmega 162x |  |  |  |  |  |

Таймеры T1 и T3 могут генерировать прерывание при наступлении следующих событий:

* переполнение счётного регистра
* равенство счётного регистра и регистра сравнения (по одному

прерыванию на каждый блок сравнения);

* сохранение счётного регистра в регистре захвата

Флаги всех прерываний 16-битных таймеров/счетчиков находятся в регистрах флагов TIFR/ETlFR/TIFRn, а разрешение/запрещение этих прерываний осуществляется установкой/сбросом соответствующих фла­гов регистров TIMSK/ETIMSK/TIMSKit (см. раздел 7.3).

Счетный регистр таймера/счетчика TCNTrt входит в состав основного блока модуля — блока реверсивного счетчика. В зависимости от режима работы модуля содержимое счетного регистра сбрасывается, инкрементируется или декрементируется по каждому импульсу тактового сигнала тай­мера/счетчика clkTn. Независимо от того, присутствует тактовый сигнал или нет, регистр доступен в любой момент времени как для чтения, так и для записи. При этом любая операция записи в счетный регистр блокирует работу всех блоков сравнения на время одного периода тактового сигнала таймера/счетчика. После подачи напряжения питания в регистре TCNT/i находится нулевое значение. При некоторых изменениях состояния тай­мера/счетчика, определяемых режимом его работы, устанавливается бит TOVn в соответствующем регистре флагов. Разрешение прерывания осу­ществляется установкой в I бита ТО1Е соответствующего регистра маски.

Регистры сравнения OCRnA/OCInB входят в состав блоков сравнения. Во время работы таймера/счетчика производится непрерывное (в каждом такте) сравнение этих регистров с регистром TCNTn. В случае равенства содержимого регистра сравнения и счетного регистра в следую­щем такте устанавливается флаг OCFnA/OCFnB в соответствую­щем регистре флагов и генерируется прерывание (если оно разрешено). Также при наступлении этого события может изменяться состояние выво­да ОСnА/ОСnВ/ОСnС микроконтроллера. Чтобы таймер/счетчик мог уп­равлять состоянием какого-либо из этих выводов, он должен быть скон­фигурирован как выходной (соответствующий бит регистра DDRx должен быть установлен в 1).

Особенностью работы блока сравнения в режимах, предназначенных для формирования ШИМ-сигналов, является двойная буферизация запи­си в регистры сравнения. Она заключается в том, что записываемое число на самом деле сохраняется в специальном буферном регистре. А измене­ние содержимого регистра сравнения происходит только при достижении счетчиком максимального значения.

Для управления таймером/счётчиком используются два регистра управления: TCCRnA, TCCRnB (рис.24, 25).

TCCR1A

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |
|  | COM1A1 | COM1A0 | COM1B1 | COM1B0 | FOC1A | FOC1B | WGM11 | WGM10 | Atmega  16x, 162x |
| Чтение(R)/Запись(W) | R/W | R/W | R/W | W | W | R/W | R/W | R/W |  |
| Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| TCCR3A |  |  |  |  |  |  |  |  |  |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |
|  | COM3A1 | COM3A0 | COM3B1 | COM1B0 | FOC3A | FOC3B | WGM31 | WGM30 | Atmega162x |
| Чтение(R)/Запись(W) | R/W | R/W | R/W | R/W | W | W | R/W | R/W |  |
| Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |

***Рис. 24. Формат регистров TCCR1A и TCCR3A***

***Табл. 20. N=1,3 Биты регистров TCCRnA***

|  |  |
| --- | --- |
| Название бита | Описание |
| FOCnA | **Принудительное изменение состояния вывода OCnx.**  При записи в бит FOCnx лог.1 состояние вывода OCnx изменяется в соответствии с установками битов COMn1x:COMn0x регистра TCCRnA.Прерывание при этом не генерируется и сброс таймера (в режиме СТС) не производится.” Эта функция доступна только в тех режимах, которые не используются для генерации сигнала с ШИМ. При чтении бита всегда возвращается 0. |
| FOCnB |
| WGMn1:  WGMn0 | **Режим работы таймера/счётчика.**Эти биты определяют режим работы таймера/счётчика Tn совместно с WGMn3:WGMn2 .. |
| COMnA1:  COMnA0 | **Режим работы блока сравнения.** Эти биты определяют поведение вывода OCnx при наступлении события” Совпадение”. Влияние содержимого этих битов на состояние вывода зависит от режима работы/таймера счётчика. |
| COMnB1:  COMnB0 |

TCCR1B

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |
|  | ICNC1 | ICES1 | - | WGM13 | WGM12 | CS12 | CS11 | CS10 | Atmega  16x, 162x |
| Чтение(R)/Запись(W) | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |  |
| Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |
| TCCR3B |  |  |  |  |  |  |  |  |  |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |  |
|  | ICNC3 | ICES3 | - | WGM33 | WGM32 | CS32 | CS31 | CS30 | Atmega162x |
| Чтение(R)/Запись(W) | R/W | R/W | R | R/W | R/W | R/W | R/W | R/W |  |
| Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |  |

***Рис. 25. Формат регистров TCCR1B и TCCR3B***

***Табл. 21. N=1,3 Биты регистров TCCRnB***

|  |  |
| --- | --- |
| Название бита | Описание |
| ICNCn | **Управление схемой подавления помех блока захвата.**  Если бит сброшен в 0, схема подавления помех выключена (захват производится по первому активному фронту). Если бит установлен в 1, схема подавления помех включена и захват осуществляется только в случае четырёх одинаковых выборок, соответствующих активному фронту сигнала. |
| ICESn | **Выбор активного фронта сигнала захвата.** Если бит ICESn сброшен в ноль, сохранение счётного регистра в регистре захвата осуществляется по нарастающему фронту сигнала. Одновременно с сохранением счётного регистра устанавливается также флаг прерывания ICFn регистра флагов. |
| WGMn3:  WGMn2 | **Режим работы таймера/счётчика.** Эти биты определяют режим работы таймера/счётчика Tn совместно с WGMn1:WGMn0 . |
| CSn2…CSn0 | **Управление тактовым сигналом.** Эти биты определяют источник тактового сигнала таймера/счётчика. |

***Обращение к 16-битным регистрам.***

Каждый 16-битный регистр таймеров/счетчиков физически размеща­ется в двух 8-битных регистрах, соответственно, для обращения к ним требуется выполнить по две операции чтения или записи. Для того чтобы запись или чтение обоих байтов содержимого 16-битного регистра проис­ходило одновременно, в составе каждого таймера/счетчика имеется спе­циальный 8-битный регистр TEMP, предназначенный для хранения стар­шего байта значения (этот регистр используется только процессором и программно недоступен).

Для выполнения цикла записи 16-битного регистра первым должен быть загружен старший байт значения, который помещается в регистр TEMP. При последующей записи младшего байта он объединяется с со­держимым регистра TEMP, и оба байта одновременно (в одном и том же тахте) записываются в 16-битныЙ регистр. Если требуется изменить не­сколько 16-битных регистров таймера/счетчика, а старшие байты всех за­писываемых значений одинаковы, то загрузку старшего байта достаточно выполнить только один раз.

Для выполнения цикла чтения 16-битного регистра первым должен быть прочитан младший байт. При его чтении содержимое старшего байта помещается в регистр TEMP. При последующем чтении старшего байта возвращается значение, сохраненное в регистре TEMP. Исключение со­ставляют только регистры сравнения ОСnА/В, при чтении которых регистр TEMP не задействуется.

При выполнении цикла обращения к 16-битному регистру тайме­ра/счетчика прерывания должны быть запрещены. В противном случив, если прерывание произойдет между двумя командами обращения к 16-битному регистру, а в подпрограмме обработки этого прерывания тоже будет произведено обращение к какому-либо из 16-битных регистров того же таймера/счетчика, содержимое регистра TEMP будет изменено. Как следствие, результат обращения к 16-битному регистру в основной про­грамме будет неверным.

***Управление тактовым сигналом***

Формирование тактового сигнала 16-битных таймеров/счетчиков clkT1 (clkT3) осу­ществляется блоком предделителя, который был рассмотрен выше.

В качестве тактового сигнала clkT1 (clkT3) таймеров/счетчиков может использоваться:

* системный тактовый сигнал (clkT1(T3) =clkI/O );
* масштабированный системный тактовый сигнал (clkT1(T3) =clkI/O /n);
* внешний сигнал, поступающий на вход Т1 (Т3) микроконтроллера

(clkTO(T2) =clkI/O /ext);

Исключение составляет лишь таймер/счётчик T3 модели Atmega 162, который не может работать от внешнего тактового сигнала

Выбор источника тактового сигнала, а также запуск и остановка тай­меров/счетчиков осуществляются с помощью битов CS02...CSOO (CS21..CS20) регистров управления таймером TCCRnB (табл. 22).

***Табл. 22****.* ***Выбор источника тактового сигнала таймеров/счетчиков ТО и Т2.***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **CSn2** | **CSn1** | **CSn0** | **Источник тактового сигнала(n=1,3)** | |
| **T3в модели ATmega 162x** | **ATmega 16** |
| 0 | 0 | 0 | Таймер/счётчик остановлен | Таймер/счётчик остановлен |
| 0 | 0 | 1 | clkI/O | clkI/O |
| 0 | 1 | 0 | clkI/O**/**8 | clkI/O**/**8 |
| 0 | 1 | 1 | clkI/O**/**64 | clkI/O**/**64 |
| 1 | 0 | 0 | clkI/O**/**256 | clkI/O**/**256 |
| 1 | 0 | 1 | clkI/O**/**1024 | clkI/O**/**1024 |
| 1 | 1 | 0 | clkI/O**/**16 | Вывод Tn, счёт осуществляется по спадающему фронту импульсов |
| 1 | 1 | 1 | clkI/O**/**32 | Вывод Tn, счёт осуществляется по нарастающему фронту импульсов |

***Режимы работы***

Режим работы таймера/счетчика Т1 (Т3) Определяется состоянием битов WGMn3:WGMn2 регистра ТССRnB совместно с битами WGMn1:WGMn0 регистра TCCRnA. Зависимость режима работы таймеров/счетчиков от состояния этих битов показана в табл. 23.

***Табл. 23. Режимs работы таймеров/счётчиков ( n=1,3)***

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Номер режима | WGM  n3 | WGM  n2 | WGM  n1 | WGM  n0 | Режим работы таймера/счетчика Tn | Модуль счёта  (TOP) | Обновление регистров OCRnx | Момент установки флага TOVn |
| 0 | 0 | 0 | 0 | 0 | Normal | $FFFF | Немедленно | $FFFF |
| 1 | 0 | 0 | 0 | 1 | Phase correct PWM,  8-битный | $ 00FF | При TOP | $0000 |
| 2 | 0 | 0 | 1 | 0 | Phase correct PWM,  9-битный | $ 01FF | При TOP | $0000 |
| 3 | 0 | 0 | 1 | 1 | Phase correct PWM,  10-битный | $ 03FF | При TOP | $0000 |
| 4 | 0 | 1 | 0 | 0 | СТС (сброс при совпадении) | OCRnA | Немедленно | $FFFF |
| 5 | 0 | 1 | 0 | 1 | FAST PWM  8-битный | $ 00FF | При TOP | При TOP |
| 6 | 0 | 1 | 1 | 0 | FAST PWM  9-битный | $ 01FF | При TOP | При TOP |
| 7 | 0 | 1 | 1 | 1 | FAST PWM  10-битный | $ 03FF | При TOP | При TOP |
| 8 | 1 | 0 | 0 | 0 | Phase and Frequency correct PWM | ICRn | $ 0000 | $0000 |
| 9 | 1 | 0 | 0 | 1 | Phase and Frequency correct PWM | OCRnA | $ 0000 | $0000 |
| 10 | 1 | 0 | 1 | 0 | Phase correct PWM | ICRn | При TOP | $0000 |
| 11 | 1 | 0 | 1 | 1 | Phase correct PWM | OCRnA | При TOP | $0000 |
| 12 | 1 | 1 | 0 | 0 | СТС(сброс при совпадении) | ICRn | Немедленно | $FFFF |
| 13 | 1 | 1 | 0 | 1 | Зарезервировано | - | - | - |
| 14 | 1 | 1 | 1 | 0 | FAST PWM | ICRn | При TOP | При TOP |
| 15 | 1 | 1 | 1 | 1 | FAST PWM | OCRnA | При TOP | При TOP |

# *Аналого-цифровой преобразователь*

Модуль 10-битного АЦП последовательного приближения входит в со­став ATmega16x, но не входит в состав микроконтроллера ATmega l62x.

Основные параметры АЦП следующие:

* абсолютная погрешность: *±2* LSB1J;
* интегральная нелинейность: ±0.5 LSВ;
* быстродействие: до 15 тыс. выборок/с.

На входе модуля АЦП имеется 8-канальный аналоговый мультиплексор, предоставляющий в распоряжение пользователя 8 каналов с несим­метричными входами.

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

Модуль АЦП может работать в двух режимах(табл.24):

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

Табл. 24. Регистры управления модулем АЦП

|  |  |  |
| --- | --- | --- |
| Название | **ATmega16x** | Описание |
| ADCSRA | \* | Регистр A управления и состояния |
| ADMUX | \* | Регистр управления мультиплексером |
| SFIOR | \* | Регистр специальных функций |
|  |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  | ADEN | ADSC | ADATE | ADIF | ADIE | ADPS2 | ADPS1 | ADPS0 |
| Чтение(R)/Запись(W) | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

***Рис. 26. Формат регистров ADCSRA***

***Табл. 25. Биты регистра ADCRA***

|  |  |
| --- | --- |
| **Название** | **Описание** |
| ADEN | Разрешение АЦП(1-включено,0-выключено) |
| ADSC | Запуск преобразования (1-начать преобразования) |
| ADATE | Выбор режима работы АЦП |
| ADIF | Флаг прерывания от компаратора |
| ADIE | Разрешение прерывания от компаратора |
| ADPS2:ADPS0 | Выбор частоты преобразования |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  | REFS1 | REFS0 | ADLAR | MUX4 | MUX3 | MUX2 | MUX1 | MUX0 |
| Чтение(R)/Запись(W) | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

***Рис.27. Формат регистра ADMUX***

***Табл. 26. Биты регистра ADCRA***

|  |  |
| --- | --- |
| **Название** | **Описание** |
| REFS1:REFS0 | Выбор источника опорного напряжения |
| ADLAR | Выравнивание результата преобразования |
| MUX4…0 | Выбор входного канала |

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

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  | ADTS2 | ADTS1 | ADTS0 | - | X | X | X | X |
| Чтение(R)/Запись(W) | R/W | R/W | R | X | X | X | X | X |
| Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

***Рис.28. Формат регистра SFIOR***

Запуск АЦП возможен не только по команде пользователя, но и по прерыванию от некоторых периферийных устройств, имеющихся в составе микроконтроллера. Для выбора режима работы используется бит ADATE регистра ADCSRA и биты ADTDS2:0 регистра SFIOR.

Если бит ADATE сброшен в 0, АЦП работает в режиме одиночного преобразования. Если же бит ADTAE установлен в 1, функционирование АЦП определяется содержимым битов ADTS2:0 (см. табл. 27).

**Табл. 27. Источник сигнала для запуска преобразования**

|  |  |  |  |
| --- | --- | --- | --- |
| ADTS2 | ADTS1 | ADTS0 | **Источник стартового сигнала** |
| 0 | 0 | 0 | Режим непрерывного преобразования |
| 0 | 0 | 1 | Прерывание от аналогового компаратора |
| 0 | 1 | 0 | Внешнее прерывание INTO |
| 0 | 1 | 1 | Прерывание па событию “Совпадение” («Совпадение А») таймера/счётчика ТО |
| 1 | 0 | 0 | Прерывание по переполнению таймера/счётчика ТО |
| 1 | 0 | 1 | Прерывание по событию «Совпадение В» таймера/счетчика Т1 |
| 1 | 1 | 0 | Прерывание па переполнению таймера/счётчика Т1 |
| 1 | 1 | 1 | Прерывание по событию «Захват» таймера/счётчика T1 |

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

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

Длительность цикла составляет 13 тактов при использовании несим­метричного входа и 13 либо 14 тактов при использовании дифференциаль­ного входа (разные значения связаны с работой схемы синхронизации); выборка и запоминание входного сигнала осуществляется в течение первых 1.5 и 2.5 тактов соответственно. Через 13 (14) тактов преобразование, бит ADSC аппаратно сбрасывается в 0 (в режиме одиночного преобразования), и результат преобразования сохраняется в регистре данных АЦП. Одновременно устанавливается флаг прерывания ADIF регистра ADCSRA и генерируется запрос на прерывание. Как и флаги остальных прерываний, флаг ADIF сбрасывается аппаратно при запуске подпрограммы обработки прерывания от АЦП или программно, записью в него лог. 1. Разрешение прерывания осуществляется установкой в 1 бита ADIE регистра ADCSRA при установленном флаге 1 регистра SREG.

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

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

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

Для формирования тактовой частоты модуля АЦП в нем имеется от­дельный предделитель. Коэффициент деления предделителя и соответ­ственно длительность преобразования определяются состоянием битов ADPS2...ADPS0 регистра ADCSRA (см. табл. 28).

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

***Табл. 28. Задание коэффициента деления предделителя частоты***

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

***Табл. 29. Управление входным мультиплексером***

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **MUX4…MUX0** | **Несимметричный вход** | **Дифференциальный вход** | | | **Прелварительное усиление** |
| **положительный** | | **отрицательный** |
| 00000 | ADC0 | **-** | | | |
| 00001 | ADC1 |
| 00010 | ADC2 |
| 00011 | ADC3 |
| 00100 | ADC4 |
| 00101 | ADC5 |
| 00110 | ADC6 |
| 00111 | ADC7 |
| 01000 |  | ADC0 | ADC0 | | 10x |
| 01001 |  | ADC1 | ADC0 | | 10x |
| 01010 |  | ADC0 | ADC0 | | 200x |
| 01011 |  | ADC1 | ADC0 | | 200x |
| 01100 |  | ADC2 | ADC2 | | 10x |
| 01101 |  | ADC3 | ADC2 | | 10x |
| 01110 |  | ADC2 | ADC2 | | 200x |
| 01111 |  | ADC3 | ADC2 | | 200x |
| 10000 |  | ADC0 | ADC1 | | 1x |
| 10001 |  | ADC1 | ADC1 | | 1x |
| 10010 |  | ADC2 | ADC1 | | 1x |
| 10011 |  | ADC3 | ADC1 | | 1x |
| 10100 |  | ADC4 | ADC1 | | 1x |
| 10101 |  | ADC5 | ADC1 | | 1x |
| 10110 |  | ADC6 | ADC1 | | 1x |
| 10111 |  | ADC7 | ADC1 | | 1x |
| 11000 |  | ADC0 | ADC2 | | 1x |
| 11001 |  | ADC1 | ADC2 | | 1x |
| 11010 |  | ADC2 | ADC2 | | 1x |
| 11011 |  | ADC3 | ADC2 | | 1x |
| 11100 |  | ADC4 | ADC2 | | 1x |
| 11101 |  | ADC5 | ADC2 | | 1x |
| 11110 | 1.22В | **-** | | | |
| 11111 | 0 В(GND) |

***Таблица 30. Выбор источника опорного напряжения***

|  |  |  |
| --- | --- | --- |
| **REFS1** | **REFS0** | **Источник опорного напряжения** |
| 0 | 0 | Внешний ИОН, подключенный к выводу AREF; внутренний ИОН отключен |
| 0 | 1 | Напряжение питания AVCC |
| 1 | 0 | Зарезервировано |
| 1 | 1 | Внутренний ИОН |

***Результат преобразования***

После завершения преобразования (при установке в 1 флага ADIF регистра ADCSRA) его результат сохраняется в регистре данных АЦП. Поскольку АЦП — 10-битный, этот регистр физически размещен в двух регистрах ввода/вывода ADCH: ADCL, доступных только для чтения.

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

Обращение к регистрам ADCH и ADCL для получения результата пре­образования должно выполняться в определенной последовательности: сначала необходимо прочитать регистр ADCL, а затем ADCH. Это требо­вание связано с тем, что после обращения к регистру ADCL процессор блокирует доступ к регистрам данных со стороны АЦП до тех пор, пока не будет прочитан регистр ADCH. Благодаря этому можно быть уверенным, что при чтении регистров в них будут находиться составляющие одного и тога же результата. Соответственно, если очередное преобразование завер­шится до обращения к регистру ADCH, результат преобразования будет потерян. С другой стороны, если результат преобразования выравнивается влево и достаточно 3-битной точности, то для получения результата можно прочитать только содержимое регистра ADCH.

Для каналов с несимметричным входом результат преобразования определяется выражением



где VIN — значение входного напряжения, а VREF *-*величина опорного на­пряжения.

Для каналов с дифференциальным входом результат преобразования определяется выражением



где VPOS — величина напряжения на положительном входе, VNEG— величина напряжения на отрительном входе, а K- коэффициент усиления.

***Параметры АЦП***

***Таблица 31.Основные параметры АЦП***

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Обозначение | Параметр | Условия | min | typ | max | Ед. изм. |
|  | Разрешение | Несимметричный вход |  | 10 |  | бит |
| Дифференциальный вход,  KU=1х и 20х | 8 |
| Дифференциальный вход,  KU=200х | 7 |
|  | Абсолютная  погрешность | Несимметричный вход,  VREF = 4В  fADC =200 КГц |  | 2 | 2.5 | LSB |
| Несимметричный вход,  VREF = 4В  fADC =1 МГц |  | 4 |  | LSB |
|  |  |  |  |
| INL | Интегральная  нелинейность | VREF = 4B |  | 0,5 |  | LSB |
| DNL | Дифференциальная  нелинейность | VREF = 4B |  | 0.25 |  | LSB |
|  | Ошибка смещения | VREF = 4B |  | 0.75 |  | LSB |
|  | Время преобразования | Режим непрерывного преобразования | 13 |  | 260 | мкс |
| fADC | Тактовая частота |  | 50 |  | 1000 | кГц |
| AVcc | Напряжение питания |  | Vсс-0,3 |  | Vcc+0,3 | В |
| VREF | Опорное напряжение | Несимметричный вход | 2.0 |  | Vсс | В |
| Дифференциальный вход | 2.0 |  | Vсс-0,2 | В |
| Напряжение внутреннего ИОН | VINT | 2.56 В | 2.4 | 2.56 | 2.7 | В |
| 1.1 В | 1.0 | 1.1 | 1.2 | В |
| RREF | Входное сопротивление канала опорного напряжения | - | - | 32 | - | кОм |
| RAIN | Входное сопротивление аналогового входа | - | - | 100 | - | МОм |

**Интерфейс SPI**

Последовательный периферийный интерфейс SPI (Serial Peripheral Interface), реализованный в микроконтроллерах семейства, имеет два назначения. Прежде всего, через него может быть осуществлено программирование микроконтроллера (так называемый режим последовательного программирования).

Вторым назначением интерфейса является организация высокоскоростного обмена данными между микроконтроллером и различными периферийными устройствами, такими как цифровые потенциометры ЦАП/АЦП, Flash-ПЗУ и др. Посредством этого интерфейса также может производиться обмен данными между несколькими микроконтроллерами AVR.

При обмене данными по интерфейсу SPI микроконтроллер AVR может работать как в режиме Master, так и в режиме Slave. При этом пользователь может задать следующие параметры:

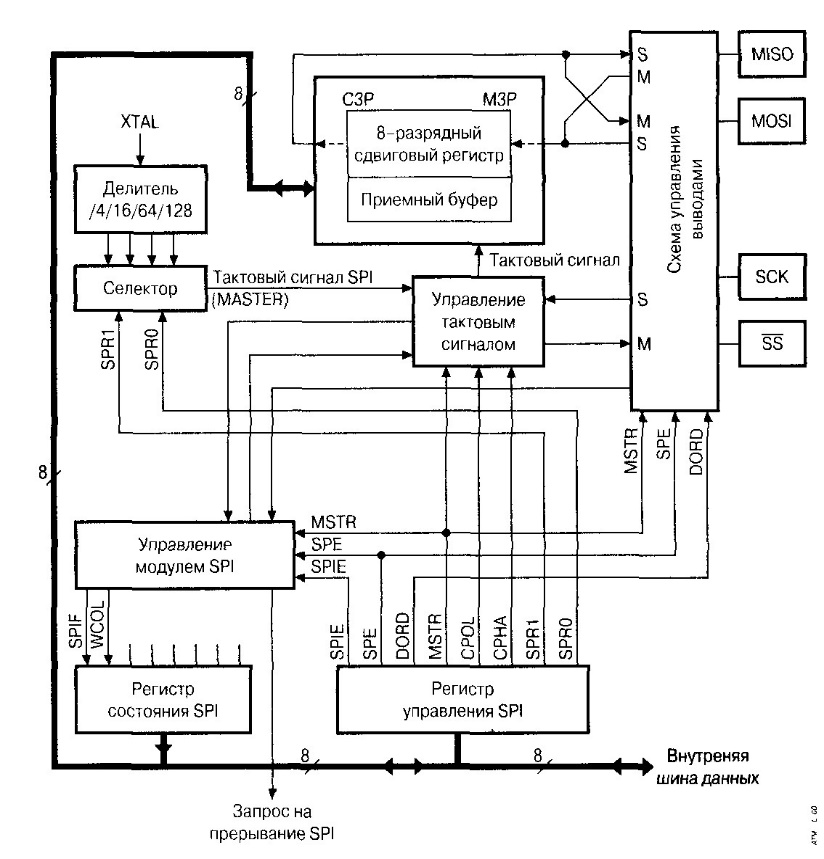
* скорость передачи (четыре программируемых значения);
* формат передачи (от младшего разряда к старшему или наоборот).

Дополнительной возможностью подсистемы SPI является «пробуждение» микроконтроллера из режима Idle при поступлении данных.

## Функционирование модуля SPI

Структурная схема модуля SPI приведена на Рис.11.

Модуль SPI использует четыре вывода микроконтроллера. Как и для большинства прочих периферийных устройств, эти выводы являются линиями порта ввода/вывода общего назначения



***Рис. 28. Структурная схема модуля SPI***

***Табл. 32. Выводы, используемые модулем SPI***

|  |  |  |  |
| --- | --- | --- | --- |
| Название | ATmega16x | ATmega162x | Описание |
| SCK | PB7 | PB7 | Выход (master)/вход(slave) тактового сигнала |
| MISO | PB6 | PB6 | Вход (master)/выход (slave) данных |
| MOSI | PB5 | PB5 | Выход (master)/вход (slave) данных |
| SS | PB4 | PB4 | Выбор ведомого устройства |

При включенном модуле SPI режим работы указанных выводов (направление передачи данных) переопределяется согласно табл. 33.

***Табл. 33. Переназначение режима работы выводов модуля SPI***

|  |  |  |
| --- | --- | --- |
| Вывод | Режим Master | Режим Slave |
| MOSI | Определяется пользователем\* | Вход |
| MISO | Вход | Определяется пользователем\* |
| SCK. | Определяется пользователем\* | Вход |
| SS | Определяется пользователем\* | Вход |

Как видно из таблицы, в некоторых случаях пользователь должен самостоятельно задать режим работы вывода, используемого модулем SPI, в соответствии с его назначением. Причем возможность управления внутренними подтягивающими резисторами выводов, работающих как входы, сохраняется независимо от способа управления их режимом работы.

Для управления модулем SPI предназначен регистр управления SPCR, расположенный по адресу $0D ($2D) . Формат этого регистра приведен на рис. 29, а краткое описание функций разрядов регистра приведено в табл. 34.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  | SPIE | SPE | DORD | MSTR | CPOL | CPHA | SPR1 | SPR0 |
| Чтение(R)/Запись(W) | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

***Рис. 29. Формат регистра SPCR***

***Табл. 34. Регистр SPCR***

|  |  |  |
| --- | --- | --- |
| Разряд | Название | Описание |
| 7 | SPIE | Разрешение прерывания от SPI |
| 6 | SPE | Включение/выключение SPI |
| 5 | DORD | Порядок передачи данных |
| 4 | MSTR | Выбор режима работы (Master/Slave) |
| 3 | CPOL | Полярность тактового сигнала |
| 2 | CPHA | Фаза тактового сигнала |
| 1,0 | SPR1:SPR0 | Скорость передачи |

Контроль состояния модуля осуществляется с помощью регистра состояния SPSR (доступен только для чтения), расположенного по адресу $0Е ($2Е). Формат этого регистра приведен на рис. 30, а назначение его разрядов описано в табл. 35.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  | SPIF | WCOL | - | - | - | - | - | SPI2X |
| Чтение(R)/Запись(W) | R | R | R | R | R | R | R | R/W |
| Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

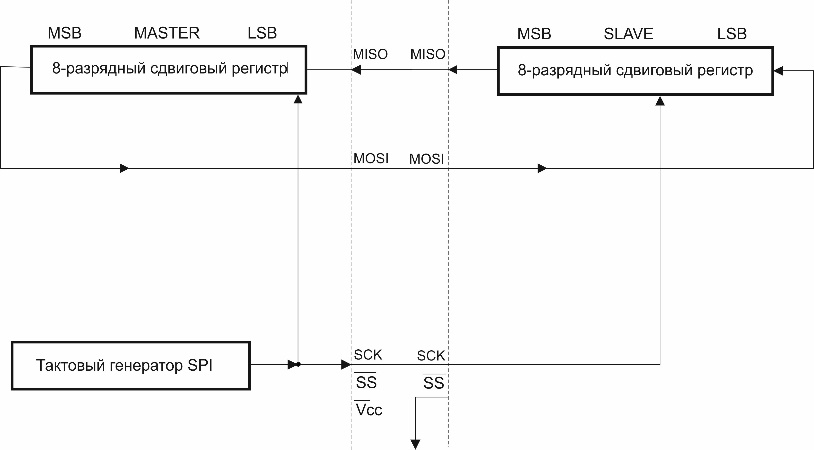
***Рис. 30. Формат регистра SPSR***

***Табл. 35. Описание регистра SPSR***

|  |  |  |
| --- | --- | --- |
| Разряд | Название | Описание |
| 7 | SPIF | **Флаг прерывания от SPI**  Данный флаг устанавливается в «1» по окончании передачи очередного байта.  Если флаг SPIE регистра SPCR установлен в «1» и прерывания разрешены, одновременно с установкой флага генерируется прерывание от SPI. Также флаг SPIF устанавливается в «1» при переводе микроконтроллера из режима Master в режим Slave посредством вывода SS (см. раздел 10.3).  Флаг сбрасывается аппаратно либо при старте подпрограммы обработки прерывания, либо после чтения регистра состояния SPI с последующим обращением к регистру данных SPI (SPDR). |
| 6 | WCOL | **Флаг конфликта записи**  Данный флаг устанавливается в «1» при попытке записи в регистр данных (SPDR) во время передачи очередного байта.  Флаг сбрасывается аппаратно после чтения регистра состояния SPI с последующим обращением к регистру данных SPI. |
| 5...0 | - | Зарезервированы, читаются как «0» |
| 0 | SPI2X | **Удвоение скорости обмена.**  При установке этого бита в 1 и работе микроконтроллера в режиме Master частота сигнала SCKудваивается. |

Передаваемые данные записываются, а принимаемые - считываются из регистра данных SPDR, расположенного по адресу $0F ($2F). Запись в этот регистр инициирует начало передачи, а при его чтении считывается содержимое приемного буфера сдвигового регистра. Поэтому этот регистр можно назвать буфером между регистровым файлом микроконтроллера и сдвиговым регистром модуля SPI.

Соединение двух микроконтроллеров (ведущий - ведомый) по интерфейсу SPI показано на рис. 31. Вывод SCK ведущего микроконтроллера является выходом тактового сигнала, а ведомого микроконтроллера - входом.



***Рис. 31. Соединение микроконтроллеров по интерфейсу SPI***

Перед выполнением обмена необходимо, прежде всего, разрешить работу модуля SPI. Для этого следует установить в «1» разряд SPE регистра SPCR. Режим работы определяется состоянием разряда MSTR этого регистра: если разряд установлен в «1», микроконтроллер работает в режиме Master (ведущий), если сброшен в «0» — в режиме Slave (ведомый).

Передача данных осуществляется следующим образом. При записи в регистр данных SP1 ведущего микроконтроллера запускается генератор тактового сигнала модуля SPI, и данные начинают поразрядно выдаваться на вывод MOSI и, соответственно, поступать на вывод MOSI ведомого микроконтроллера. Порядок передачи разрядов данных определяется состоянием разряда DORD регистра SPCR. Если разряд установлен в «1», первым передается младший разряд байта, если же сброшен в «0» — старший разряд. После выдачи последнего разряда текущего байта генератор тактового сигнала останавливается с одновременной установкой в «1» флага «Конец передачи» (SPIF). Если прерывания от модуля SPI разрешены (флаг SPIE регистра SPCR установлен в «1»), генерируется запрос на прерывание. При подключении к ведущему устройству нескольких ведомых, что разрешено спецификацией SPI, выбор конкретного ведомого устройства осуществляется подачей на его вход SS сигнала ВЫСОКОГО уровня.

Образно говоря, два сдвиговых регистра ведомого и ведущего устройств можно считать одним распределенным 16-разрядным циклическим сдвиговым регистром, как показано на Рис. 10.4. Одновременно с передачей данных от ведущего к ведомому происходит передача и в обратном направлении. Таким образом, в каждом цикле сдвига происходит обмен данными между устройствами.

В модуле используется одинарная буферизация при передаче и двойная — при приеме. Это означает, что готовый для передачи байт данных не может быть записан в регистр данных SPI до окончания предыдущего цикла обмена. При попытке изменить содержимое регистра данных во время передачи устанавливается в «1» флаг WCOL регистра SPSR. Сбрасывается этот флаг после чтения регистра SPSR с последующим обращением к регистру данных SPI.

Соответственно во время приема принятый байт должен быть прочитан из регистра данных SPI до того, как в сдвиговый регистр поступит последний разряд следующего байта. В противном случае первый байт будет потерян.

## Режимы передачи данных

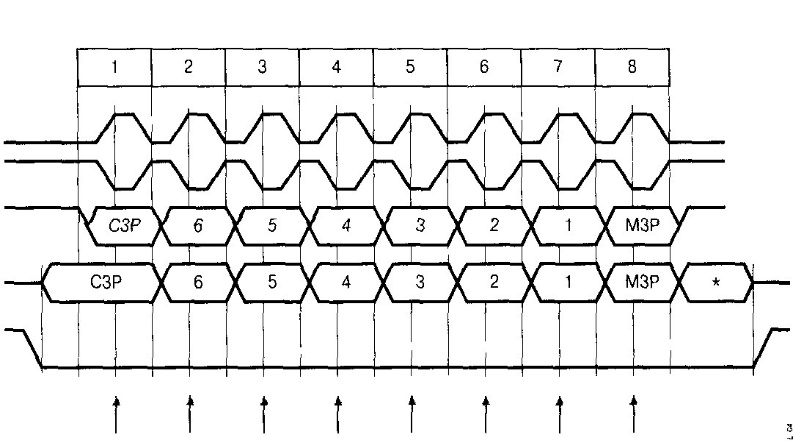
Спецификация интерфейса SPI предусматривает 4 режима передачи данных. Эти режимы различаются соответствием между фазой (момент считывания сигнала) тактового сигнала SCK, его полярностью и передаваемыми данными. Всего существует 4 такие комбинации, определяемые состоянием разрядов СРНА и CPOL регистра SPCR (см. табл. 36).

***Табл. 36. Задание режима передачи данных***

|  |  |
| --- | --- |
| Разряд | Описание |
| CPOL | **Полярность тактового сигнала**  0 - генерируются импульсы положительной полярности, при отсутствии импульсов на выводе присутствует НИЗКИЙ уровень;  1 - генерируются импульсы отрицательной полярности, при отсутствии импульсов на выводе присутствует ВЫСОКИЙ уровень. |
| СРНА | **Фаза тактового сигнала**  0 - обработка данных производится по переднему фронту импульсов сигнала SCK (для CPOL = 0 - по нарастающему, а для CPOL = 1 - по спадающему фронту).  1 - обработка производится по заднему фронту импульсов сигнала SCK. |

Соответствующие этим режимам форматы обмена данными через SPI приведены на рис. 32 и 33 (передача ведется от старшего разряда к младшему).

Частота тактового сигнала SCK и, соответственно, скорость передачи данных по интерфейсу определяются состоянием разрядов SPR1:SPR0 регистра SPCR (см. табл. 37). Разумеется, речь идет о микроконтроллере, работающем в режиме Master, т.к. именно он является источником тактового сигнала. Для устройства, находящегося в режиме Slave, состояние этих разрядов безразлично.

Такт SCK (для справки) 

SCK (CPOL = 0)

SCK (CPOL = 1)

MOSI (от ведущего)

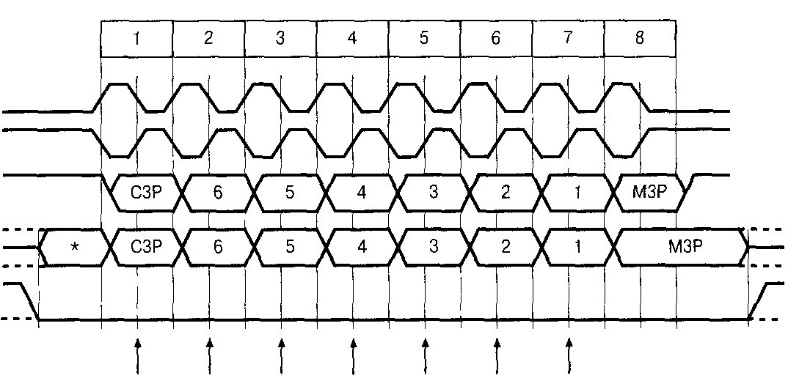
MISO (от ведомого)

SS (к ведомому)

Моменты выборок

\*Не определено (как правило, С3Р только что принятого байта)

***Рис. 32. Передача данных при СРНА = 0 и DORD = 0***

Такт SCK (для справки) 

SCK (CPOL = 0)

SCK (CP0L = 1)

MOSI (от ведущего)

MISO (от ведомого)

SS (к ведомому)

Моменты выборок

\* Не определено (как правило, М3Р только что переданного байта)

***Рис. 33. Передача данных при СРНА = 1 и DORD = 0***

***Табл. 37. Задание частоты тактового сигнала SCK***

|  |  |  |  |
| --- | --- | --- | --- |
| SPI2X | SPR1 | SPR0 | Частота сигнала SCK |
| 0 | 0 | 0 | FCLK / 4 |
| 0 | 0 | 1 | FCLK / 16 |
| 0 | 1 | 0 | FCLK / 64 |
| 0 | 1 | 1 | FCLK / 128 |
| 1 | 0 | 0 | FCLK / 2 |
| 1 | 0 | 1 | FCLK / 8 |
| 1 | 1 | 0 | FCLK / 32 |
| 1 | 1 | 1 | FCLK / 64 |

FCLK - тактовая частота микроконтроллера.

## Использование вывода SS

Вообще этот вывод предназначен для выбора активного ведомого устройства и в режиме Slave всегда является входом. При подаче на него напряжения НИЗКОГО уровня модуль SPI активируется и вывод MOSI переключается в режим вывода данных (если это задано пользователем). Остальные выводы модуля SPI являются в этом режиме входами. А при подаче на вывод SS напряжения ВЫСОКОГО уровня все выводы модуля SPI переключаются в режим ввода данных. При этом модуль переходит в неактивное состояние и прием данных не производится.

Следует помнить, что каждый раз, когда на вывод SS подается напряжение ВЫСОКОГО уровня, происходит сброс модуля SPI. Соответственно, если изменение состояния этого вывода произойдет во время передачи данных, и прием и передача немедленно прекратятся, а передаваемый и принимаемый байты будут потеряны.

Если же микроконтроллер находится в режиме Master (разряд MSTR регистра SPCR установлен в «1»), направление передачи данных через вывод SS определяется пользователем. Если вывод сконфигурирован как выход, он работает как линия вывода общего назначения и не влияет на работу модуля SPI. Если же он сконфигурирован как вход, то для обеспечения нормальной работы модуля SPI на него должен быть подан сигнал ВЫСОКОГО уровня. Дело в том, что подача на этот вход сигнала НИЗКОГО уровня от какой-либо внешней схемы будет воспринята модулем SP1 как выбор данного микроконтроллера в качестве ведомого и, соответственно, начало передачи ему данных. Во избежание конфликта на шине система SPI в таких случаях выполняет следующие действия:

1. Флаг MSTR регистра SPCR сбрасывается, и микроконтроллер переключается в режим Slave. Как следствие, выводы MOSI и SCK начинают функционировать как входы.
2. Устанавливается флаг SPIF регистра SPSR, генерируя запрос на прерывание от SPI. Если прерывания от SPI разрешены и флаг I регистра SREG установлен в «1», происходит запуск подпрограммы обработки прерывания.

Таким образом, если ведущий микроконтроллер использует передачу данных, управляемую прерыванием, и существует вероятность подачи на вход SS сигнала НИЗКОГО уровня, в подпрограмме обработки прерывания от SPI обязательно должна осуществляться проверка состояния флага MSTR. При обнаружении сброса этого флага он должен быть программно установлен обратно в «1» для обратного перевода микроконтроллера в режим Master. [ 1, с.419-426 ]

## Интерфейс UART

Микроконтроллеры Atmega 16x и Atmega 162x имеют в своем составе модуль полнодуплексного универсального асинхронного приемопередатчика (UART). Через него осуществляется прием и передача информации, представленной последовательным кодом, поэтому модуль UART часто называют также последовательным портом. С помощью этого модуля микроконтроллер может обмениваться данными с различными внешними устройствами.

Скорость передачи данных может варьироваться в широких пределах, причем высокие скорости передачи могут быть достигнуты даже при относительно низкой тактовой частоте микроконтроллера.

Известно, что при передаче данных могут происходить различные сбои. Модуль UART может обнаруживать и сигнализировать о следующих внештатных ситуациях:

* переполнение;
* ошибка кадрирования;
* неверный старт-бит.

Для уменьшения вероятности сбоев в модуле реализована такая полезная функция, как фильтрация помех.

Для взаимодействия с программой в модуле предусмотрены 3 раздельных прерывания, запрос на которые генерируется при наступлении следующих событий: «передача завершена», «регистр данных передатчика пуст» и «прием завершен».

Выводы микроконтроллера, используемые модулем UART, являются линиями портов ввода/вывода общего назначения.

***Табл. 38. Выводы, используемые модулями UART***

|  |  |  |  |
| --- | --- | --- | --- |
| Название | **ATmega16x** | **ATmega162x** | Описание |
| RXD | PD0 |  | Вход UART0 |
| RXD0 |  | PD0 |
| TXD | PD1 |  | Выход UART0 |
| TXD0 |  | PD1 |
| XCK | PB0 | PD4 | Вход/выход внешнего тактового сигнала UART0 |
| RXD1 |  | PB2 | Вход UART1 |
| TXD1 |  | PB3 | Выход UART1 |
| XCK1 |  | PD2 | Вход/выход внешнего тактового сигнала UART1 |

***Табл. 39.Размещение регистров данных модулей UART***

|  |  |  |  |
| --- | --- | --- | --- |
| Регистр | **ATmega16x** | **ATmega162x** | Описание |
| UDR | \* |  | Регистр данных UART |
| UDR0 |  | \* | Регистр данных UART0 |
| UDR1 |  | \* | Регистр данных UART1 |
|  | | | |

***Табл. 40. Регистры управления и состояния модулей UART***

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Регистр | | **ATmega16x** | **ATmega162x** | | Описание | | | | | | | | |
| UСSRA | | \* |  | | Регистр управления A | | | | | | | | |
| UСSRB | | \* | \* | | Регистр управления B | | | | | | | | |
| UСSRC | | \* |  | | Регистр управления С | | | | | | | | |
| UСSR0A | |  | \* | | Регистр управления A UART0 | | | | | | | | |
| UСSR0B | |  | \* | | Регистр управления B UART0 | | | | | | | | |
| UСSR0C | |  | \* | | Регистр управления С UART0 | | | | | | | | |
| UСSR1A | |  | \* | | Регистр управления A UART1 | | | | | | | | |
| UСSR1B | |  | \* | | Регистр управления B UART1 | | | | | | | | |
| UСSR1C | |  | \* | | Регистр управления С UART1 | | | | | | | | |
|  | | | | | | | | | | | | | |
|  | | | 7 | | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  | | | RXCn | | TXCn | UDREn | FEn | DORn | UPEn | U2Xn | MPCMn |
| Чтение(R)/Запись(W) | | | R | | R/W | R | R | R | R | R/W | R/W |
| Начальное значение | | | 0 | | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

***Рис. 34. Формат регистра UCSRnA***

***Табл. 41. Описание регистра UCSRA (UCSRnA)***

|  |  |  |
| --- | --- | --- |
| 7 | RXCn | **Флаг завершения приема**  Флаг устанавливается в 1 при наличии непрочи­танных данных в буфере приемника (регистр данных UDR). Сбрасывается флаг апаратно после опустошения буфера. Если бит RXCIE (RXCIEn) ре­гистра UCSRB (UCSRnB) установлен, то при установке флата генерируется запрос на прерывание –“прием завершен”. |
| 6 | TXCn | **Флаг завершения передачи**  Данный флаг устанавливается в «1» после передачи всех разрядов посылки (включая стоп-бит) из сдвигового регистра передатчика, при условии, что в регистр данных UDR не было загружено новое значение. Этот флаг наиболее полезен при полудуплексной связи, при которой передающее устройство должно освободить линию и перейти в режим приема сразу же после окончания передачи.  Если разряд TXCIEn регистра UCSRB (UCSRnB) установлен, то при установке флага ТХСn генерируется запрос на прерывание «передача завершена». Флаг сбрасывается аппаратно при выполнении подпрограммы обработки прерывания или программно записью в него лог. «1». |
| 5 | UDREn | **Регистр данных пуст**  Данный флаг устанавливается в «1» после пересылки байта из регистра данных UDR в сдвиговый регистр передатчика. Установка этого флага означает, что передатчик готов к получению нового значения для передачи.  Если разряд UDRIEn регистра UCSRB (UCSRBn) установлен, генерируется запрос на прерывание «регистр данных пуст». Сбрасывается флаг аппаратно при записи в регистр данных. Соответственно, если прерывание используется, в обработчике прерывания следует обязательно произвести запись в этот регистр. В противном случае после окончания обработки прерывания оно будет вызвано снова. |
| 4 | FEn | **Флаг ошибки кадрирования**  Данный флаг устанавливается в «1», при обнаружении ошибки кадрирования, т.е. если стоп-бит принятого слова равен «0».  Флаг сбрасывается при приеме стоп-бита, равного «1». |
| 3 | DORn | **Флаг переполнения**  Данный флаг устанавливается в «1», если в сдвиговом регистре приемника находится новое принятое слово, а старое содержимое регистра данных не прочитано. Флаг остается установленным до тех пор, пока не будет прочитано содержимое регистра данных.  Флаг сбрасывается при пересылке принятых данных из сдвигового регистра приемника в регистр данных. |
| 2 | UPEn | **Флаг ошибки контроля чётности.**  Флаг устанавливается в 1, если в данных, находящихся в буфере приемника, выявлена ошибка контроля чётности. При отключенном контроле чётности этот бит постоянно сброшен в 0. |
| 1 | U2Xn | **Удвоение скорости обмена.**  Если этот бит установлен в 1, то коэффициент деления предделителя контроллера скорости передачи уменьшается с 16 до 8, удваивая тем самым скорость асинхронного обмена по последовательному каналу. Этот бит используется только при асинхронном режиме работы и в синхронном режиме должен быть сброшен. |
| 0 | MPCMn | **Режим мультипроцессорного обмена.**  Если этот бит установлен в 1, ведомый микроконтроллер ожидает приема кадра, содержащего адрес. Кадры, не содержащие адреса устройства, игнорируются. |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  | RXCIEn | TXCIEn | UDRIEEn | RXENn | TXENn | UCZn2 | RXB8n | ТХВ8n |
| Чтение(R)/Запись(W) | R/W | R/W | R/W | R/W | R/W | R/W | R | R/W |
| Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

***Рис. 35. Формат регистра UCSRnB***

***Табл. 42. Описание регистра UCSRnB***

|  |  |  |
| --- | --- | --- |
| Разряд | Название | Описание |
| 7 | RXCIEn | **Разрешение прерывания по завершении приема**  Если данный разряд установлен в «1», то при установке флага RXC в регистре UCSRA (UCSRAn) происходит запуск подпрограммы обработки прерывания «прием завершен» (если прерывания разрешены) |
| 6 | ТХСIn | **Разрешение прерывания по завершении передачи**  Если данный разряд установлен в «1», то при установке флага ТХСn в регистре USR (UCSRA) происходит запуск подпрограммы обработки прерывания «передача завершена» (если прерывания разрешены) |
| 5 | UDRIEn | **Разрешение прерывания при очистке регистра данных UART**  Если данный разряд установлен в «1», то при установке флага UDREn в регистре UCSRA (UCSRAn) происходит запуск подпрограммы обработки прерывания «регистр данных пуст» (если прерывания разрешены) |
| 4 | RXENn | **Разрешение приема**  При установке данного разряда в «1» разрешается работа приемника UART. Если приемник выключен, флаги ТХСn, DORn и FnE не могут быть установлены. При сбросе разряда RXENn эти флаги не сбрасываются |
| 3 | TXENn | **Разрешение передачи**  При установке данного разряда в «1» разрешается работа передатчика UART. Если разряд сбрасывается в «0» во время передачи, выключение передатчика произойдет только после завершения передачи текущего слова и слова, находящегося на момент выключения в регистре данных UDR. |
| 2 | UCSZn2 | **Формат посылок**  Если разряд установлен в «1», производится передача и прием 9-разрядных данных. При передаче значение старшего (8-го) разряда берется из разряда ТХВ8 регистра, а при приеме записывается в разряд RXB8. |
| 1 | RXB8n | **8-й разряд принимаемых данных**  Если флаг CHR9 установлен в «1», этот разряд содержит значение старшего разряда принятого слова. |
| 0 | ТХВ8n | **8-й разряд передаваемых данных**  Если флаг CHR9 установлен в «1», содержимое этого разряда передается как старший разряд слова. |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | URSELn | UMSELn | UPMn1 | UPMn0 | USBSn | UCSZn1 | UCSZn0 | UCPOLn |
| Чтение(R)/Запись(W) | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Начальное значение | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |

***Рис. 36. Формат регистра UCSRC (UCSRnC)***

***Таблица 43. Описание регистра UCSRnB***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Разряд | Название | Описание | | |
| 7 | URSELn | **Выбор регистра**  Этот бит определяет, в какой из регистров модуля производится запись. Если бит установлен в 1, обращение производится к регистру UCSRnC. Если же бит сброшен в 0 обращение производится к регистру UBRRnH. | | |
| 6 | UMSELn | **Режим работы UART**  Если данный разряд установлен в «0», то модуль работает в асинхронном режиме. Если бит установлен в 1, то модуль работает в синхронном режиме. | | |
| 5  4 | UPMn1  UPMn0 | **Режим работы схемы контроля и формирования бита четности.**  Эти биты определяют функционирование схем контроля и формирования бита чётности. | | |
| 3 | USBSn | **Количество стоп-битов.**  Этот бит определяет количество стоп-битов, посылаемых передатчиком. Если бит сброшен в 0, передатчик посылает 1 стоп-бит, если установлен в 1, то 2 стоп-бита. Для приёмника содержимое этого бита безразлично. | | |
| 2 | UCSZn1 | **Количество стоп-битов.**  Если разряд установлен в «1», производится передача и прием 9-разрядных данных. При передаче значение старшего (8-го) разряда берется из разряда ТХВ8 регистра, а при приеме записывается в разряд RXB8. | | |
| 1 | UCSZn0 | **Формат посылок.**  Совместно с битом UCSZn2 регистра UCSRnB эти биты определяют количество битов данных в посылках. | | |
| 0 | UСPOLn | **Полярность тактового сигнала.**  Значение этого бита определяет момент выдачи и считывания данных на выводах модуля. Бит используется только при работе в синхронном режиме.При работе в асинхронном режиме он должен быть сброшен в 0. | | |
| **UCPOL** | **Выдача данных на вывод TXDn** | **Считывание данных с вывода RXDn** |
| 0 | Спадающий фронт XCKn | Нарастающий фронт XCKn |
| 1 | Нарастающий фронт XCKn | Спадающий фронт XCKn |

***Передача данных***

Работа передатчика разрешается установкой в 1 бита TXEN (TXENn) регистра UCSRB (UCSRnB). При установке бита вывод TXD (ТХDn) под­ключается к передатчику UART и начинает функционировать как выход независимо от установок регистров управления портом. Если используется синхронный режим работы, то переопределяется также функционирова­ние вывода ХСК (ХСКn).

Передача инициируется записью передаваемых данных в буферный ре­гистр передатчика — регистр данных UDR (UDRjn). После этого данные пересылаются из регистра UDR (UDRn) в сдвиговый регистр передатчика. Одновременно, если используются 9-битные данные, значение бита ТХВ8(ТХВ8n) регистра UCSRB (UCSRnB) копируется в 9-й бит сдвигового ре­гистра. При этой возможны два варианта:

• запись в регистр UDR осуществляется в тот момент, когда передат­чик находится в состоянии ожидания (предыдущие данные уже пе­реданы). В этом случае данные пересылаются в сдвиговый регистр сразу же после записи в регистр UDR;

• запись в регистр UDR осуществляется во время передачи, в этом случае данные пересылаются в сдвиговый регистр после передачи последнего стоп-бита текущего кадра.

Очевидно, что 9-й бит данных должен быть загружен в бит ТХВ8 (ТХВ8n) до записи младшего байта слова в регистр данных.

П*ос*ле пересылки слова данных в сдвиговый регистр флаг UDRE (UDREn) регистра UCSRA (UCSRnA) устанавливается в 1, что означает готовность передатчика к получению нового слова данных. В этом состоя­нии флаг остается до следующей записи в буфер. Одновременно с пере­сылкой в регистре формируется служебная информация — старт-бит, воз­можный бит четности, а также один или два стоп-бита.

После загрузки сдвигового регистра его содержимое начинает сдвигать­ся вправо и поступать на вывод TXD (TXDn). Скорость сдвига определяется настройками контроллера тактовых сигналов. В синхронном режиме изменение состояния вывода TXD (TXDn) происходит по одному из фронтов сигнала ХСК (ХСКn). Если бит UCPOL (UCPOLn) регистра UCSRC (UCSRnC) сброшен в 0, измене­ние состояния вывода происходит по нарастающему фронту сигнала, если же установлен в 1 — по спадающему фронту.

Если во время передачи в регистр UDR было записано новое слово данных, то после передачи последнего стоп-бита оно автоматически пере­сылается в сдвиговый регистр. Если же к моменту окончания передачи кадра буфер передатчика будет пуст, устанавливается флаг прерывания «передача завершена» ТХС (ТХСn) регистра UCSRA (UCSRnA). Сброс флага осуществляется аппаратно при входе в подпрограмму обработки со­ответствующего прерывания или программно, записью в этот бит лог. 1

Выключение передатчика осуществляется сбросом бита TXEN (TXENn) регистра UCSRB (UCSRnB). Если в момент выполнения этой команды осу­ществлялась передача, сброс бита произойдет только после завершения те­кущей и отложенной передач, т. е. после очистки сдвигового и буферного регистров передатчика. При выключенном передатчике вывод TXD (TXDn) может использоваться как контакт ввода/вывода общего назначения.

Ниже приведен простейший пример подпрограммы передачи по ин­терфейсу USART Эта подпрограмма ждет очистки буфера передатчика, а затем загружает в него новое значение.

**Пример на Си**

void USART\_Transmit( unsigned int data )

{

*/\** Ждать очистки буфера передатчика \*/

while *(!(* UCSRA *&* ( 1<<UDRE ));

*/\** Скопировать 9-й бит данных из г17 в ТХВВ \*/

UCSRB &= ~ ( 1 << TXB8 );

if (data *&* 0x10O)

UCSRB |= ( 1 << TXB8 );

/\* Загрузить младший байт данных в буфер, начать передачу \*/

UDR = data;

}

***Приём данных***

Работа приемника разрешается установкой бита RXENn регистра UCSRnB.При установке бита RXDn подключается к приёмнику USART и начинает функционировать как вход независимо от установок регистров управления портом. Если используется синхронный режим работы, переопределяется также функционирование вывода ХСК (ХСКn).

Прием данных начинается сразу же после обнаружения приемником корректного старт-бита. Каждый бит содержимого кадра затем считывается с частотой, определяемой установками контроллера скорости передачи или тактовым сигналом ХСК (ХСКn). Считанные биты данных последова­тельно помещаются в сдвиговый регистр приемника до обнаружения пер­вого стоп-бита кадра. После этого содержимое сдвигового регистра пере­сылается в буфер приемника, из которого принятое значение может быть получено путем чтения регистра данных модуля. При использовании 9-битных слов данных значение старшего бита может быть определено по состоянию флага RXS (RXSn) регистра UCSRB (UCSRnB). Причем содер­жимое старшего бита данных должно быть считано до обращения к регист­ру данных. Это связано с тем, что флаг RX8 (RXS8n) отображает значение старшего бита слова данных кадра, находящегося на верхнем уровне буфера приемника, состояние которого при чтении регистра данных изменится.

Если во время приема кадра была включена схема контроля четности, она вычисляет бит четности для всех битов принятого слова данных и сравнивает его с принятым битом четности. Результат проверки запомина­ется в буфере приемника вместе с принятым словом данных и стоп-бита-ми. Наличие или отсутствие ошибки контроля четности может быть затем определено по состоянию флага UPE (UPEn). Этот флаг устанавливается в 1» если следующее слово, которое может быть прочитано из буфера, име­ет ошибку контроля четности. При выключенном контроле четности флаг UPE (UPEn) всегда читается как 0.

Блок приемника модулей USART имеет еще два флага, показывающих состояние обмена, — флаг ошибки кадрирования FE (FЕn) и флаг перепол­нения DOR (DORn). Флаг FE (FЕn) устанавливается в 1, если значение пер­вого стоп-бита принятого кадра не соответствует требуемому, т. е. равно 0.

Флаг DOR (DORn) индицирует потерю данных из-за переполнения бу­фера приемника. Этот флаг устанавливается в 1 в случае приема старт-бита нового кадра при заполненных буфере и сдвиговом регистре приемника. Установленный флаг DOR (DORn) означает, что между прошлым байтом, считанным из регистра UDR, и байтом, считанным в данный момент, про­изошла потеря одного или нескольких кадров.

Обратите внимание на то, что все флаги ошибок буферизуются вместе со словом данных, т. е. соответствующие биты регистра UCSRA (UCSRhA) относятся к кадру, слово данных которого будет прочитано при следующем обращении к регистру данных UDR (UDRn). Поэтому состояние этих флагов должно быть считано веред обращением к регистру данных. Кроне того, для совместимости с будущими устройствами рекомендуется при записи в регистр UCSRA (UCSRnА) сбрасывать соответствующие этим фла­гам биты записываемого значения в 0.

Для индикации состояния приемника в модулях USART используется флаг прерывания «прием завершен» RXC (RXCn) регистра UCSRA (UCSRnA). Этот флаг устанавливается в 1 при наличии в буфере приемни­ка непрочитанных данных и сбрасывается в 0 при опустошении буфера (после считывания всех находящихся в нем данных).

Выключение приемника осуществляется сбросом бита RXEN (RXENn) регистра UCSRB (UCSRnB). В отличие от передатчика, приемник выклю­чается сразу же после сброса бита, а значит, кадр, принимаемый в этот мо­мент, теряется. Кроме того, при выключении приемника очищается его буфер, т. е. теряются также все непрочитанные данные. При выключенном приемнике вывод RXD (RXDn) может использоваться как контакт вво­да/вывода общего назначения.

Пример подпрограммы приема по интерфейсу USART приведен ниже. Как и в предыдущем примере, здесь используется опрос флага прерывания.

***Пример на Си***

unsigned int USART\_Receive (void)

{

unsigned char status, resh, resl

/\* ждать заполнения буфера приемника \*/

while(!( UCSRA *&* ( 1 << RXC)));

*/\** Прочитать 9-й бит данных и флаги состояния \*/

status = UCSRA;

resh = UCSRB;

/\* Прочитать младший байт данных \*/

resl = UDR;

/\* В случае ошибки вернуть \*/

if ( status & ( 1 << FE) | (1<<DOR) | (1<<UPE)

return –l;

/\* Выделить 9-й бит данных \*/

resh = (resh >> l) & 0x01;

return ((resh << 6) | resl);

}

Собственно, прием всех битов кадра осуществляется по-разному, в за­висимости от режима работы модуля. При работе модуля USART в синх­ронном режиме состояние вывода RXD (RXDn) считывается по одному из фронтов сигнала ХСК (ХСn). Если бит UCPOL (UCPOn) регистра UCSRC (UCSRnC) сброшен в 0, считывание состояния вывода происхо­дит по спадающему фронту сигнала ХСК (ХСn), если же установлен в 1 — по нарастающему фронту сигнала. Другими словами, считывание данных с вывода RXD (RXDn) и их выдача на вывод TXD (ТХDn) происходят по противоположным фронтам.

Для обеспечения приема в асинхронном режиме работы используются схемы восстановления тактового сигнала и данных. Схема восстановления тактового сигнала предназначена для синхронизации внутреннего такто­вого сигнала, формируемого контроллером скорости передачи, и кадров, поступающих на вывод RXD (RXDn) микроконтроллера. Схема восста­новлении данных осуществляет считывание и фильтрацию каждого бита принимаемого кадра.

Схема восстановления тактового сигнала осуществляет опрос входа приемника с целью определения старт-бита кадра. Частота опроса зависит от состояния бита U2X(U2Xn) регистра UCSRA (UCSRnA). В обычном ре­жиме (при *U2Xn =* 0) частота опроса в 16 раз превышает скорость передачи данных, а в ускоренном режиме (при U2X = 1) — в 8 раз.

Обнаружение изменения сигнала на выводе RXD (RXDn) с лог. 1 (ре­жим ожидания) на лог 0 интерпретируется как возможное появление пе­реднего фронта старт-бита. После этого в нормальном режиме проверяет­ся значение 8- й, 9-й и 10- й выборок входного сигнала, а в ускоренном ре­жиме — 4- й, 5- й и 6- й выборок. Если значение хотя бы двух выборок из указанных равно лог. 1 старт-бит считается ложным (помеха), а приемник переходит к ожиданию следующего изменения входного сиг­нала с лог. 1 на лог. 0. В противном случае считается, что обнаружен Старт-бит новой последовательности, с которым синхронизируется внут­ренний тактовый сигнал приемника. После этого начинает работать схема восстановления данных. [ 1, с.477-501 ]

Раздел

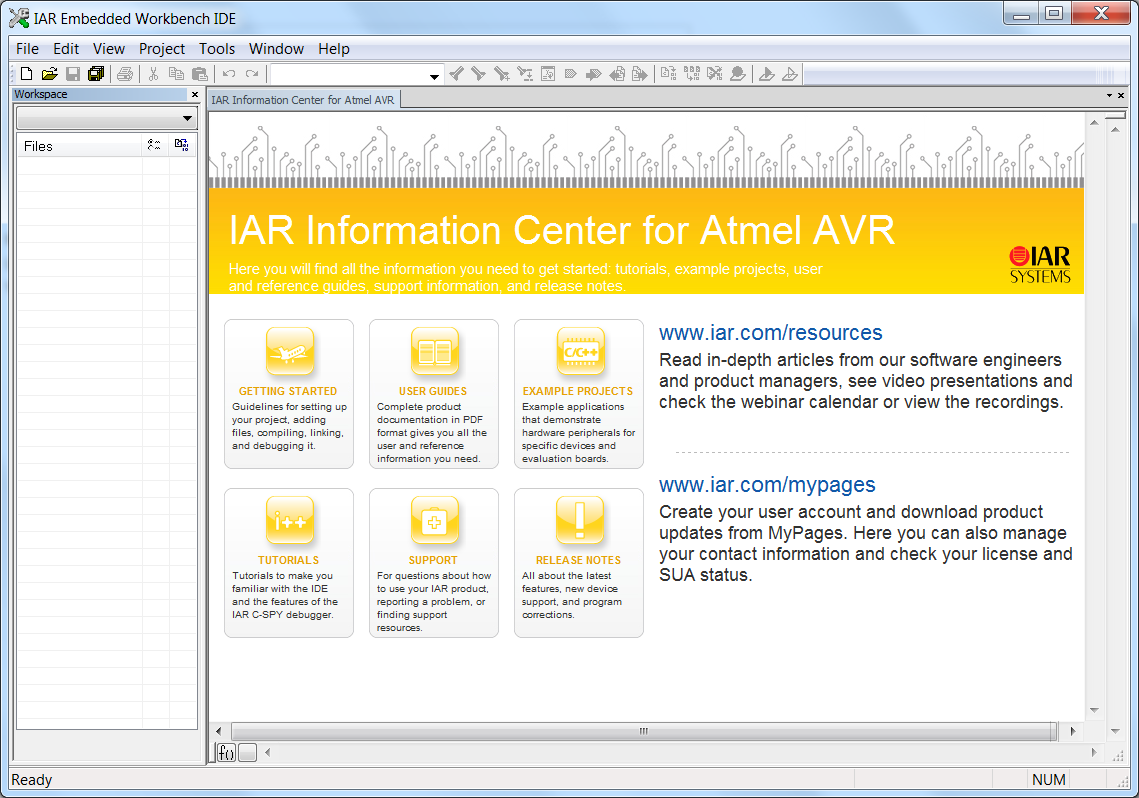
4

Программные средства разработки

## Запуск среды

Среда разработки IAR Embedded Workbench.

На рис. 37 представлено главное окно программы IAR Embedded Workbench.



***Рис. 37. Главное окно IAR Embedded Workbench***

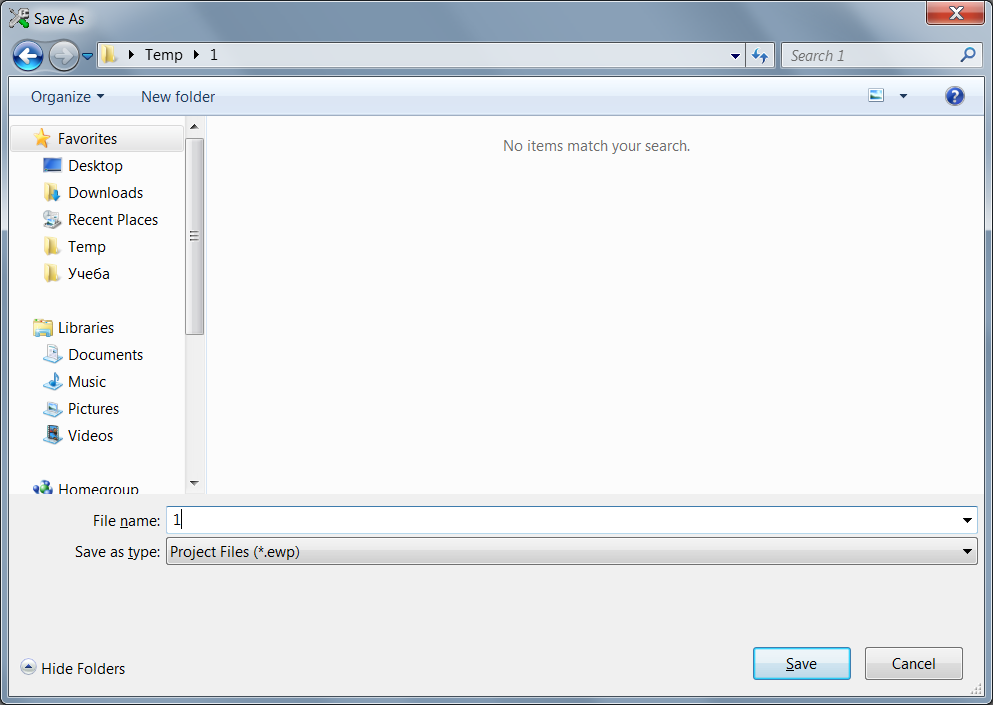
## Настройка проекта

При создании нового проекта необходимо зайти в меню Project -> Create New Project. Возникнет диалоговое окно (рис. 38).

Необходимо выбрать тип создаваемого проекта – Empty Project.

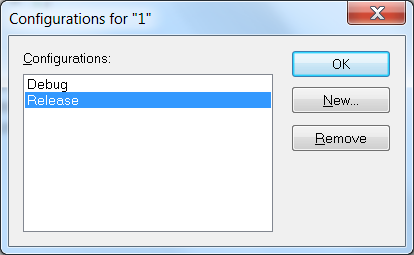
|  |
| --- |
|  |
| ***Рис. 38. Выбор типа создаваемого файла*** |

Сохраните проект в известном вам месте (рис. 39).



***Рис. 39. Сохранение проекта***

После сохранения зайдите в меню Project -> Edit Configurations. Появится диалоговое окно (рис. 40). Выберите Release, нажмите OK.

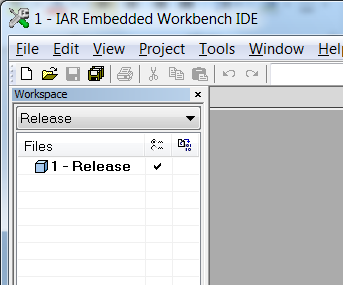


***Рис. 40. Выбор конфигурации проекта***

Далее откройте меню настроек проекта Project -> Options. Выберите категорию General Options, вкладку Target. Задайте тип процессора ATmega16, выбрав его в поле Processor configuration (рис. 41).

|  |
| --- |
| ***Рис. 41. Выбор процессора***  Выберите категорию Linker (рис. 42), вкладку Output, в разделе Output file отметьте Override default и измените расширение файла на hex. В разделе Format отметьте Other, для поля Output format укажите intel-standard. |
| ***Рис. 42. Настройка выходных файлов*** |

Нажмите ОК. Вы перейдёте обратно в главное окно программы. В разделе Workspace отобразится только что созданный проект (рис. 43).



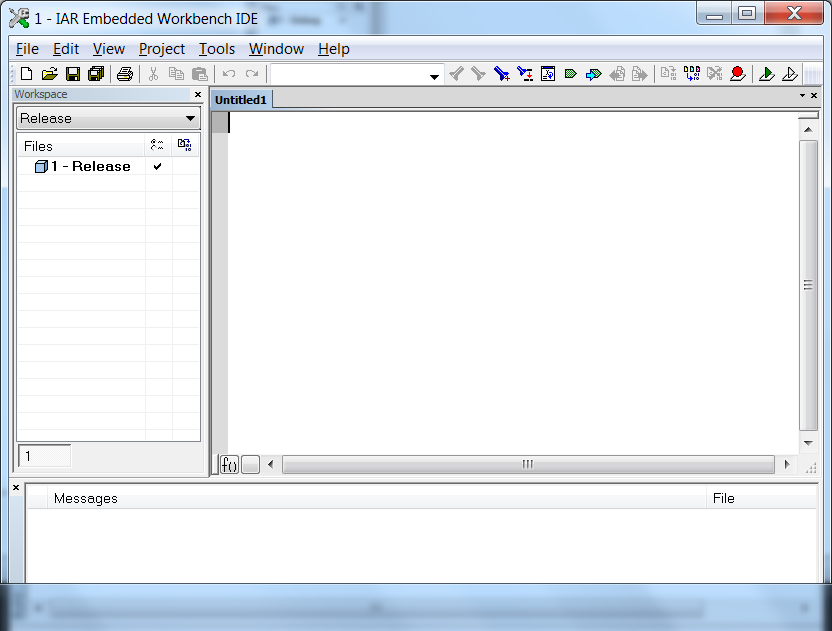
***Рис. 43. Созданный проект.***

На данном этапе у нас есть пустой настроенный проект.

## Добавление файлов

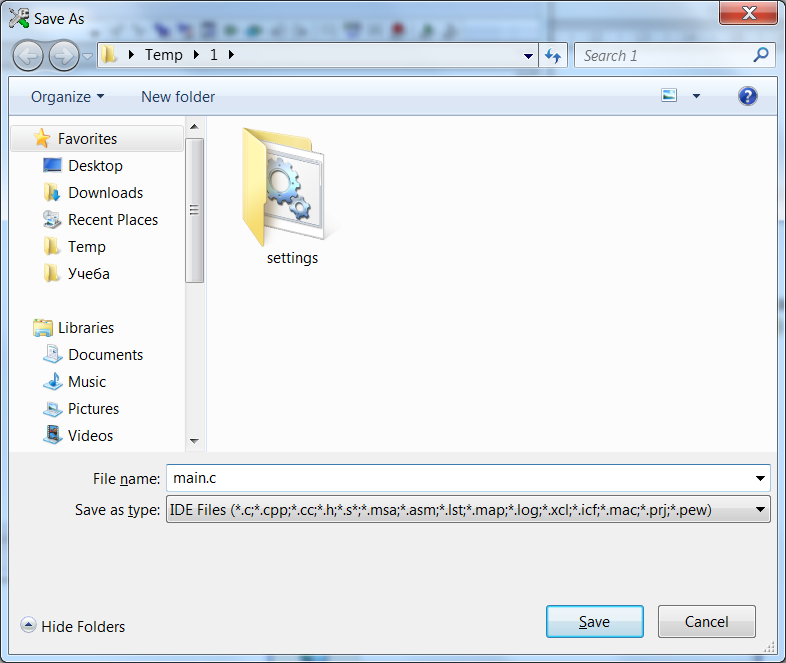
Теперь необходимо добавить в проект файл, написать исходный код и прошить процессор.

Чтобы добавить файл в проект зайдите в меню File -> New -> File. Создается пустой файл (рис. 44).



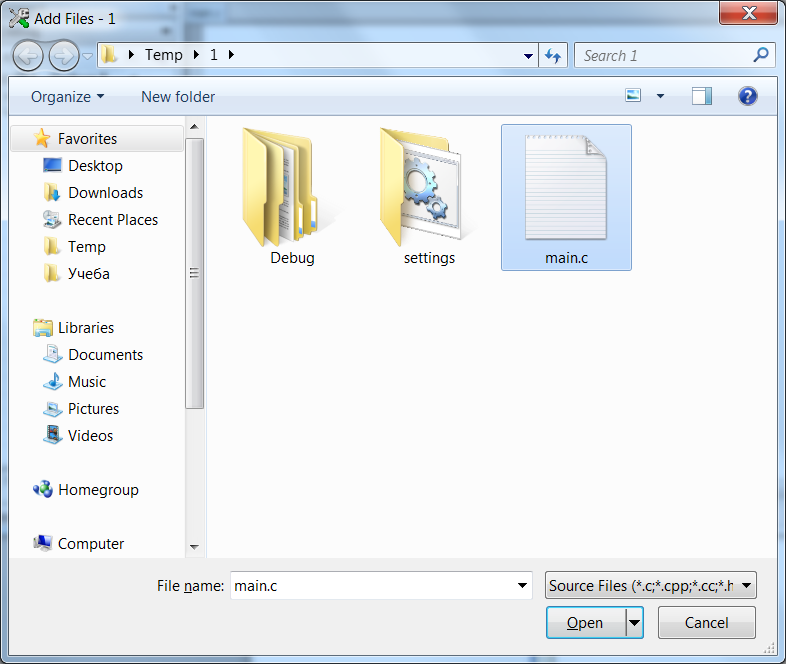
***Рис. 42. Новый пустой файл***

Сохраните файл в папке с файлом проекта. Для этого зайдите в меню File -> Save As. Задайте имя файла и сохраните его (рис. 45).



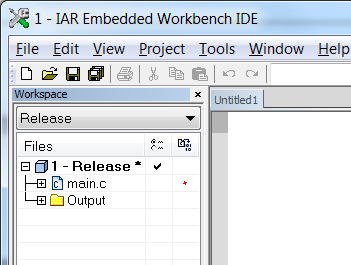
***Рис. 45. Сохранение файла***

Добавьте только что созданный файл в проект. Для этого зайдите в меню Project -> Add Files. Выберите только что созданный ‘С’ файл и нажмите кнопку Open (рис. 46).



***Рис. 46. Добавление файла в проект***

Теперь в проекте появился один файл. В разделе Workspace также появился файл в проекте (рис. 47).



***Рис. 47. Новый файл в проекте***

## Написание исходного кода

Проверяем, всё ли удачно было выполнено, для этого перейдём к написанию исходного кода. Хотим добиться мигания светодиода.

*#include <iom16.h>*

*#define LED1\_PORT PORTD\_PORTD4  
#define LED1\_DDR DDRD\_DDD4*

*int main()  
{  
 LED1\_DDR = 1;  
 LED1\_PORT = 1;*

*while (1)  
 {*

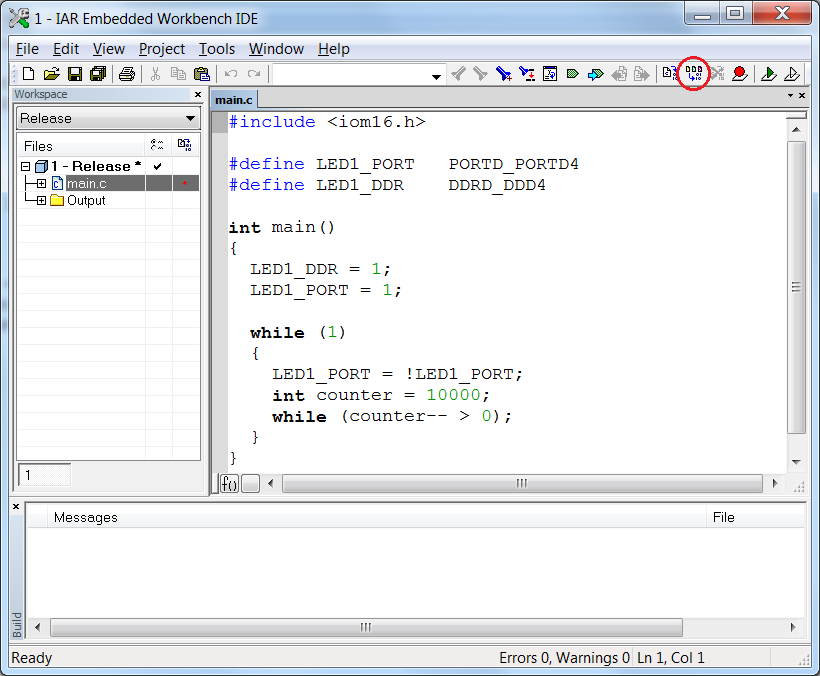
*LED1\_PORT = !LED1\_PORT;  
int counter = 10000;  
while (counter-- > 0);*

*}*

*}*

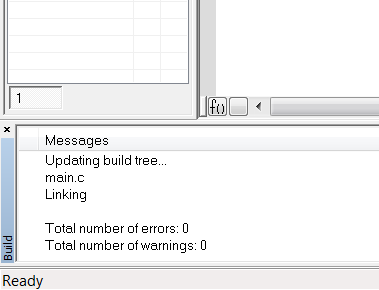
## Компиляция

Написав этот код, нажмите кнопку Make (рис. 48).



***Рис. 48. Расположение кнопки Make. Компиляция проекта***

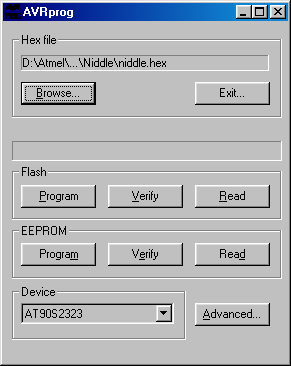
После нажатия кнопки Make в окне Build отображаются результаты сборки проекта (рис. 49).



***Рис. 49. Окно Build***

## Программирование процессора

Для программирования используется программа AVRPROG.EXE, разработанная фирмой ATMEL (рис. 50).

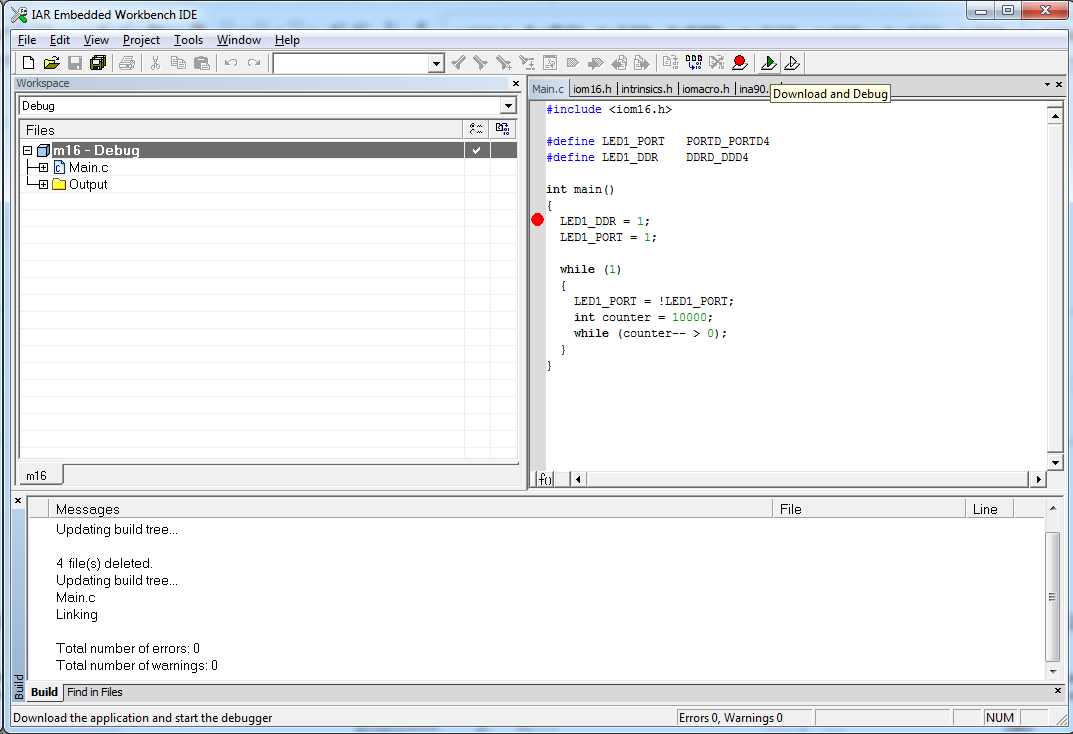


***Рис. 50. Программирование процессора***

В разделе Device выберите ATmega16. В разделе Hex file укажите путь к прошиваемому файлу (Папка проекта -> Release -> Exe -> “имя проекта”. hex). Нажмите Program в разделе Flash и дождитесь выполнения прошивки. Убедитесь, что светодиод мигает.

## Тестирование и отладка

Выполним пошаговую отладку программы. В первую очередь необходимо установить сборку проекта в режиме Debug. Затем, как показано на рис.51, установим Breakpoint, дважды кликнув по необходимой строке.



***Рис. 51. Установка «Breakpoint»***

Далее необходимо пересобрать проект и нажать кнопку Download and Debug.

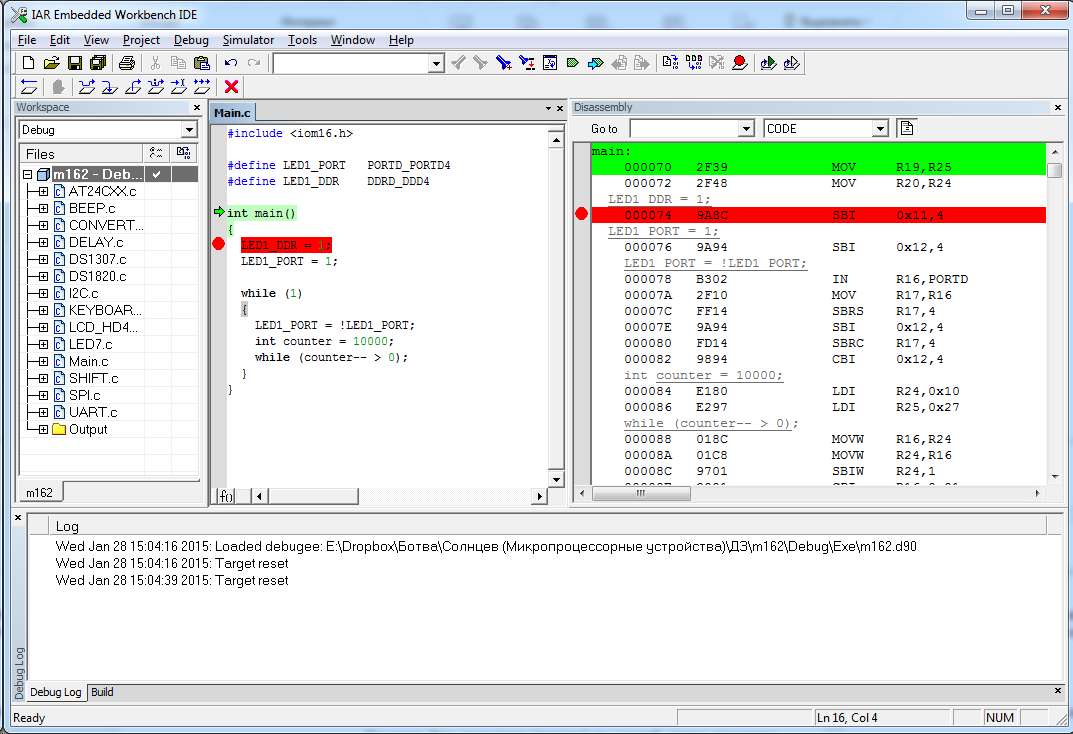
**Пошаговый режим** **и выход из отладчика**

Вы можете использовать отладчик, чтобы перемещаться по программе. Выберите Step Into и Step Over из меню Debug.

Команды Step позволяют “шагать” по каждой строке исходного текста. Текущая команда высвечивается на каждом шаге. Step Into позволяет войти в вызываемую функцию, Step Over – перешагнуть через неё, не входя во внутрь.

Для завершения работы с отладчиком в любой момент времени Вы можете выбрать Stop Debugging из меню Debug и возвратиться в режим редактирования.

Обратите внимание, что в режиме отладки на экране видны ещё два окна. Первое – окно кода, где в пошаговом режиме параллельно с исходным текстом на языке C идёт трассировка текста на ассемблере (см. рис. 52).



***Рис. 51. Отладка проекта***

Раздел

5

Практическая часть

Перед выполнением работы предполагается, что вы ознакомлены с материалами, представленными в частях **1**, **2** и **4** настоящего руководства.

🖞

**Особенно внимательно прочитайте раздел 3.**

Проверьте работоспособность вашего стенда согласно следующей процедуре:

1. подать на стенд питание (постоянное или переменное напряжение 8-15В, 0.3А);
2. контроль работоспособности узла стабилизатора напряжения на плате стенда. При работоспособности должно наблюдаться свечение красного светодиода рядом с радиатором. Если светодиод не горит, то проверить отсутствие замыкания или подключение нагрузки превышающей величину тока выдаваемого блоком питания или превышения величины 1.5А;
3. прошить в микроконтроллер тестовую программу, поставляемую вместе со стендом. Если программирование прошло удачно, то это означает работоспособность процессора и правильность его установки в стенд;
4. контроль запуска процессора, звукового излучателя, некоторых портов ввода-вывода. При включении стенда или нажатия на кнопку RESET, должен раздаться звуковой сигнал (при установке джампера SS) и начать перемигиваться светодиоды, расположенные около кнопки RESET;

🖞

1. контроль SPI интерфейса и 7 сегментного светодиодного индикатора. при старте программы на небольшое время зажигаются все сегменты индикатора, в дальнейшем сегменты могут отражать совершенно случайное состояние. Если индикатор ничего не отражает, то проконтролировать наличие сигналов CLK, MOSI, ST на сдвиговом регистре 74HC595, если сигналы есть, но индикатор не работает, то заменить регистр.
2. контроль ЖКИ индикатора. При включении питания должна загореться подсветка индикатора, в момент старта тестовой программы на индикатор выводятся символы, зажигающие все сегменты индикатора на время около 1сек. После этого на экран выводится надпись с названием стенда и версией на время 1.5сек, после этого стенд переходит в режим контроля АЦП, клавиатуры, времени и температуры;
3. контроль работоспособности клавиатуры. Поочередно нажимать на клавиши и контролировать соответствие нажимаемой клавиши и высвечиваемого символа в левом нижнем углу ЖКИ индикатора;
4. контроль работоспособности АЦП. Вращать переменные резисторы стенда и контролировать изменение результатов работы АЦП на ЖКИ;
5. контроль работоспособности интерфейса I2C и часов реального времени. Если RTC функционирует правильно, то в левой части ЖКИ должно отображаться время, установленное в этих часах;
6. контроль работоспособности цифрового термометра. Подключить термодатчик типа DS18B20 к стенду и контролировать показания на ЖКИ. Если термодатчик не работоспособен или не подключен, то будет отображаться температура 99градусов;
7. контроль работоспособности и интерфейса RS-232. Подключить стенд к компьютеру используя шнур удлинитель СОМ порта, загрузить терминальную программу и настроить пор т к которому подключен стенд на режим 57600 8N1. При старте программы в UART микроконтроллера выводится подсказка по командам стенда. Вводя команды можно контролировать работоспособность последовательного интерфейса;
8. контроль работоспособности EEPROM типа 24сХХ. используя подключение к компьютеру ввести команду записи в ячейку памяти и проконтролировать читаемое из нее значение. Команда записи w24=addr,data команда чтения r24=addr;
9. если все пункты выполнены и результаты адекватны ожидаемым, то это означает работоспособность стенда!

# Тестовая программа

Ниже приведена тестовая программа для микроконтроллера ATmega 16 в виде текста на языке C.

🖞

Блоки исходного кода демонстрируют работу с различными периферийными устройствами и могут быть использованы в качестве основы для выполнения домашнего задания и лабораторных работ.

Тестовая программа при старте генерирует звуковой сигнал, выводит приветствие на ЖКИ, а затем входит в бесконечный цикле, выводя текущую температуру и время. Параллельно осуществляется циклическое переключение светодиодов.

В последовательный порт (UART0) при старте на скорости **57600**кбит/сек и настройке порта **8N1** выводится текстовая подсказка по работе с программой, и воспринимаются команды по работе с периферией стенда.

***Примечание:* 8N1** – означает 8 бит данных, отсутствие четности (*No parity*) и 1 стоповый бит. Обозначение **7E2** означало бы - 7 бит данных, тип контроля четности – бит четности (*Even*), 2 стоповых бита.

Тест программы на C с комментариями приведен в ***Приложении 4***.

## Список команд терминала для тестовой программы

**Список команд**

**«1»** - при нажатии на кнопки цифровой клавиатуры стенда на LCD экран выводится значение кнопки, при нажатии на решетку происходит выход в главное меню;

**«2»** - при нажатии на кнопки цифровой клавиатуры стенда на 7-сегментном индикаторе выводится значение кнопки, при нажатии на решетку происходит выход в главное меню;

**«3»** - при нажатии на кнопки телефонной клавиатуры раздается сигнал, при нажатии на решетку происходит выход в главное меню;

**«4»** - при нажатии на кнопки (1, 2, 3) загорается или гаснет соответствующий световой индикатор, при нажатии на решетку происходит выход в главное меню;

**«5»** - в UART-порт выводится текущее время, полученное с датчика DS1307 (см. *Часы реального времени DS1307*);

**«6»** - в терминальной программе пользователь вводит время и дату и они устанавливаются как текущие в датчик DS1307;

**«7»** - в терминальной программе пользователь вводит адрес и значение ячейки, далее данные записываются в EEPROM AT24CXX (см. *EEPROM)*;

**«8»** - в терминальной программе пользователь вводит адрес ячейки, далее выводится значение ячейки EEPROM AT24CXX;

**«9»** - в UART-порт выводится температура, полученная с датчика DS1820 (см. );

Тестовая программа имеет следующую структуру:

1. 9 процедур, выполняющих задаваемые командами действия (**ExecuteTask**n);
2. процедура обслуживания прерывания, которая получает задания (**Serial\_ISR**);
3. основная функция (**main)**, которая запускает программу;
4. функция циклического вывода на экран (**AutonomousMode)**.

Каждая функция для работы с периферией использует вызовы библиотечных функций, расположенных в отдельных файлах (можно найти их в рабочей папке).

При выполнении домашнего задания целесообразно придерживаться следующей структуры программы:

1. Несколько функций, аналогичных функциям **ExecuteTask** тестовой программы, которые выполняют требуемые действия;
2. Функция **Serial\_ISR**, которая вызывает их в зависимости от введенного кода (берет с изменениями из тестовой программы);
3. Функция **main,** которая инициализирует устройства (берется из тестовой программы).

Для вставки ассемблерных инструкций в код Си необходимо использовать inline-операторы:

*asm("SBI 0x11, 0x04"); //Установить LED1\_DDR в 1*

# Рекомендации по выполнению домашнего задания

Тексты заданий приведены в ***Приложении 6***.

Последовательность выполнения домашнего задания следующая:

1. Ознакомиться с данным руководством;
2. Ознакомиться с тестовой программой;
3. Разработать программу в соответствии с полученным заданием на языке Си;
4. Осуществить компиляцию и выполнение программы в симуляторе, исправить синтаксические и семантические ошибки;
5. Осуществить тестирование программы на стенде, убедиться в ее работоспособности;

**Во всех заданиях необходимо устранить дребезг контактов.**

**Задания необходимо выполнять в прерываниях.**

**Работа и обмен информацией в режиме прерываний**

Прерывание — это аппаратное событие, например, байт пришел в порт, на выводе изменился логический уровень, АЦП обсчитала напряжение или в таймере произошло переполнение. Когда сигнал приходит, то периферийный блок в своем регистре поднимет флаг запроса на прерывание. Если прерывание разрешено, то происходит прерывание основной программы или программы обслуживание прерывания более низкого уровня по приоритету.

Когда приходит прерывание то контроллер завершит текущую команду (машинную инструкцию) сразу же начнет выполнять процедуру обработки прерывания, а как выполнит, то вернется к прерванной фоновой программе.

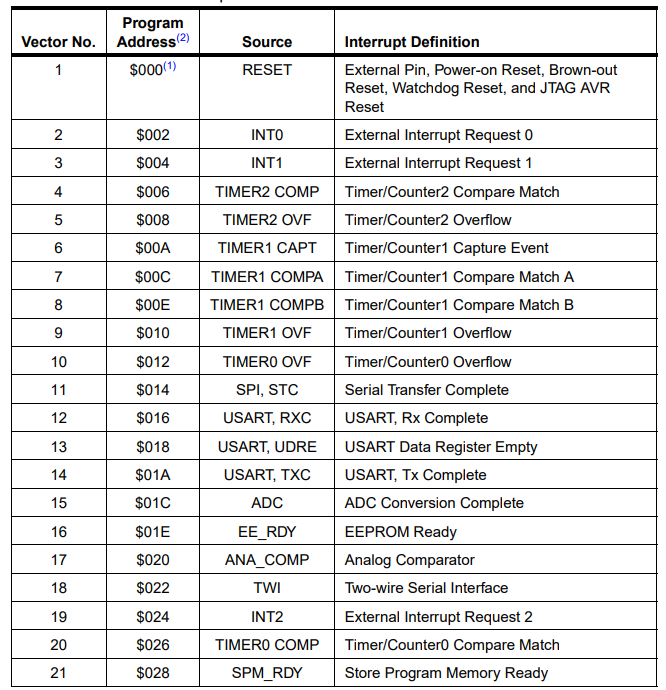
Прерывания можно, а часто необходимо запрещать, чтобы посреди критичного участка не начать выполнять обработка неизвестного события. Запрещать их можно глобально, флагом I в регистре SREG, а можно локально — запрещая источник каждого прерывания индивидуально. По умолчанию, при сбросе, все прерывания от устройств запрещены, глобальный флаг тоже сброшен. Включать их возможно по мере надобности. В любом случае надо формализовать последовательность работы с устройствами и основной программой.

Поскольку прерывание приходит внезапно, а у нас могут быть несохраненные данные, то обработчик их должен сохранить и при выходе в фоновую программу вернуть все как было.

Впрочем, если бездумно подходить к этому делу, то можно получить некорректную работу контролера. Особенно при использовании высокоуровневых языков вроде Си, где вся эта процедура скрыта от глаз программиста и если он не понимает инструкция языка ассемблер, и не понимает работу контроллера на уровне машинных инструкций, то ошибку найти не сможет.

**Вектора прерываний**

Процессор переедает управление программе обработки прерывания по вектору прерывания от конкретного устройства. У контролера есть таблица прерываний. Она должна быть в самом начале памяти программ.



Вектор — это адрес перехода. У каждого аппаратного события, имеющего прерывание есть свой вектор. Аппаратных событий у AVR много, поэтому таблица прерываний весьма большая, десятки адресов.

Когда происходит прерывание, то контроллер запоминает адрес следующей исполняемой команды прерванной программы в стеке и делает переход на адрес вектора соответствующего прерывания. В адресе вектора соответствующего прерывания в ячейку находится команда RJMP которая перенаправит контролер на программу обработки прерывания.

**Приоритеты прерываний**

Тот процесс, который начал раньше обработку прерывания, тот и выполняется. Когда процессор уходит на обработку прерывания, то аппаратно происходит запрет всех остальных прерываний.

Его, конечно, можно включить программно, установив флаг I в регистре SREG и тогда у нас будут вложенные прерывания, но обращаться с этим следует очень осторожно. Так как множественные прерывания нагружают стек и может произойти его переполнение, что даст полный сбой работы программы, который сложно диагностировать.

По выходу из обработчика, по команде RETI флаг I вернется в прежнее состояние.

Если во время обработки одного прерывания придет другое, то оно не потеряется. У него взведется флаг и как только мы завершим обработку текущего прерывания автоматически произойдет переход к отложенному прерыванию. Единственное, что мы можем потерять количество одинаковых прерываний. Т.е. если обрабатывается, например, INT0 и прерывания запрещены, а в это время придет три раза INT1, то на выходе INT1 обработается только один раз.

А если во время обработки первого прерывания случится не одно, а, скажем, два или три? Какое из этих отложенных прерываний будет выполнено первым? А по порядку в таблице векторов. У кого адрес младше тот и будет обрабатываться.

Следует сделать вывод:

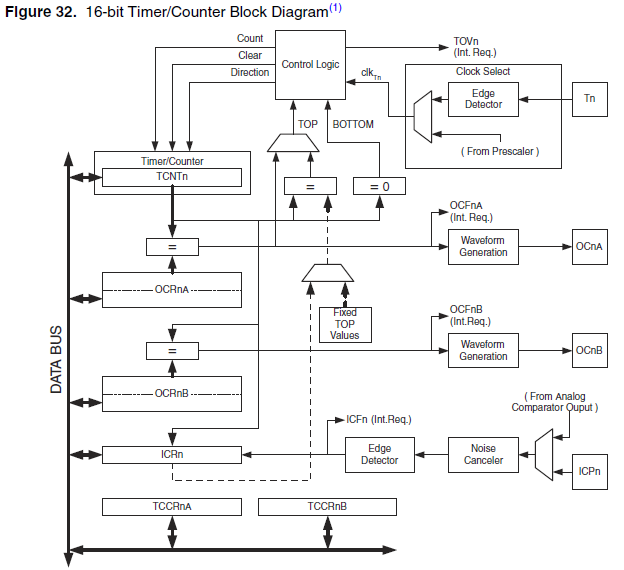
* Прерывания — это аппаратные события.
* У **каждого** прерывания есть **свой** персональный адрес вектор, по которому и будет послан контролер в случае чего.
* По умолчанию они запрещены локально и глобально.
* Вызов прерывания может быть когда угодно и где угодно, между любыми двумя командами.
* В обработчике прерывания надо учитывать тот факт, что данные нужно сохранять и восстанавливать при выходе.
* Приоритет прерываний работает по принципу «кто первый запустился, тот и выполняется». Остальные запрещены аппаратно, но можно разрешить программно уже в обработчике.
* Отложенные прерывания ждут своей очереди — права управлять контролером, в порядке жесткой очереди по таблице прерываний.
* Прерывания — это источник проблем и ошибок работы. Но без них не обойтись, если нельзя допустить простаивания процессора.
* Поскольку прерывание внезапное и может быть когда угодно, то обработчик прерывания должен выполняться максимально коротко и быстро. Зашел, отметился, вышел. Никаких задержек и длинных циклов. Все сложные вещи надо производить только в фоновой задаче.

**Обработка прерывания по таймеру.**

На данный момент рассмотрим прерывание, которое находится в таблице, размещённой выше на 7 позиции — **TIMER1 COMPA**, вызываемое по адресу 0x006.

Теперь рассмотрим наш 16-битный таймер или **TIMER1**.

Вот его структурная схема



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

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

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

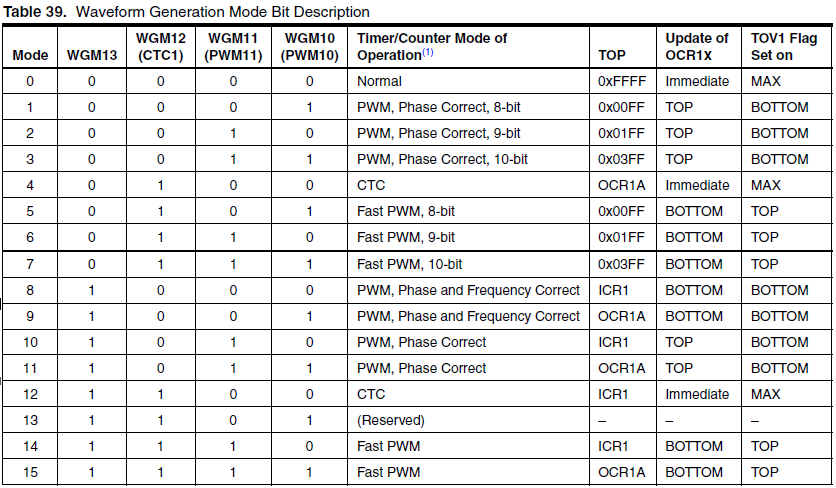
Есть также регистр TCCR. Данный регистр — это регистр управления. Там настраиваются определенные биты, отвечающие за конфигурацию таймера.

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

Он состоит из двух половинок, так как у нас контроллер 8-битный и в нем не может быть 16-битных регистров. Поэтому в одной половинке регистра (а физически в одном регистре) хранится старшая часть регистра, а в другом — младшая. Можно также назвать это регистровой парой, состоящей из двух отдельных регистров TCCR1A и TCCR1B. Цифра 1 означает то, что регистр принадлежит именно таймеру 1.

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

За установку режима отвечают биты WGM



Мы видим здесь очень много разновидностей режимов.

**Normal** — это обычный режим, таймер считает до конца.

**PWM** — это **ШИМ** только разные разновидности, то есть таймер может играть роль*широтно-импульсного модулятора.*

**CTC** — это сброс по совпадению. Здесь сравниваются регистры TCNT и OCR. Таких режима два, второй работает с регистром ICR.

**Применение таймера**

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

Поэтому первую функцию мы назовём **timer\_ini**

**//———————————————**

**void timer\_ini(void)**

**{**

**}**

**//———————————————**

Также давайте наши функции, а также какие-то законченные блоки с объявлением глобальных переменных, с прототипами функций будем отделять друг от друга вот такими чёрточками, которые за счет наличия двух слешей впереди компилятор обрабатывать не будет и примет их за комментарии. За счёт этих отчерчиваний мы будем видеть, где заканчивается одна функция и начинается другая.

Данная функция, как мы видим не имеет ни каких аргументов — ни входных, не возвращаемых. Давайте сразу данную функцию вызовем в функции main()

unsigned char butcount=0, butstate=0;

**timer\_ini();**

Теперь мы данную функцию начнём потихонечку наполнять кодом.

Начнем с регистра управления таймером, например, с TCCR1B. Используя нашу любимую операцию «ИЛИ», мы в определённый бит регистра занесём единичку

void timer\_ini(void)

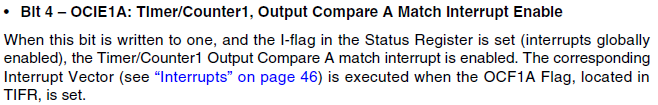
{

**TCCR1B |= (1<<WGM12); // устанавливаем режим СТС (сброс по совпадению)**

Из комментария мы видим, что мы работает с битами режима, и установим мы из них только бит WGM12, остальные оставим нули. Исходя из этого мы сконфигурировали вот такой режим:

image04

Также у таймера существует ещё вот такой регистр — **TIMSK**. Данный регистр отвечает за маски прерываний — **Interrupt Mask**. Доступен данный регистр для всех таймеров, не только для первого, он общий. В данном регистре мы установим бит **OCIE1A**, который включит нужный нам тип прерывания **TIMER1 COMPA**



TCCR1B |= (1<<WGM12); // устанавливаем режим СТС (сброс по совпадению)

**TIMSK |= (1<<OCIE1A); //устанавливаем бит разрешения прерывания 1ого счетчика по совпадению с OCR1A(H и L)**

Теперь давайте поиграемся с самими регистрами сравнения **OCR1A(H и L)**. Для этого придётся немного посчитать. Регистр **OCR1AH** хранит старшую часть числа для сравнения, а регистр **OCR1AL** — младшую.

Но прежде чем посчитать, давайте пока напишем код с любыми значениями данного регистра и потом поправим, так как дальше мы будем инициализировать делитель, и он тоже будет учувствовать в расчёте требуемого времени счёта. Без делителя таймер будет слишком быстро считать.

TIMSK |= (1<<OCIE1A); //устанавливаем бит разрешения прерывания 1ого счетчика по совпадению с OCR1A(H и L)

**OCR1AH = 0b10000000; //записываем в регистр число для сравнения**

**OCR1AL = 0b00000000;**

**TCCR1B |= ();//установим делитель.**

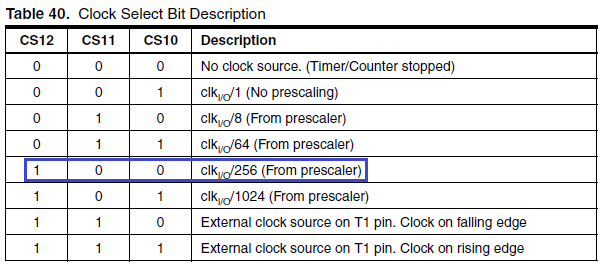
Пока никакой делитель не устанавливаем, так как мы его ещё не посчитали. Давайте мы этим и займёмся.

Пока у нас в регистре **OCR1A** находится число 0b1000000000000000, что соответствует десятичному числу 32768.

Микроконтроллер у нас работает на частоте 8000000 Гц.

Разделим 8000000 на 32768, получим приблизительно 244,14. Вот с такой частотой в герцах и будет работать наш таймер, если мы не применим делитель. То есть цифры наши будут меняться 244 раза в секунду, поэтому мы их даже не увидим. Поэтому нужно будет применить делитель частоты таймера. Выберем делитель на 256. Он нам как раз подойдёт, а ровно до 1 Гц мы скорректируем затем числом сравнения.

Вот какие существуют делители для 1 таймера



Мы видим, что нам требуется установить только бит **CS12**.

Так как делитель частоты у нас 256, то на этот делитель мы поделим 8000000, получится 31250, вот такое вот мы и должны занести число в TCNT. До такого числа и будет считать наш таймер, чтобы досчитать до 1 секунды. Число 31250 — это в двоичном представлении 0b0111101000010010. Занесём данное число в регистровую пару, и также применим делитель

OCR1AH = 0b**01111010**; //записываем в регистр число для сравнения

OCR1AL = 0b**00010010**;

TCCR1B |= (**1<<CS12**);//установим делитель.

С данной функцией всё.

Теперь следующая функция — обработчик прерывания от таймера по совпадению. Пишется она вот так

**ISR (TIMER1\_COMPA\_vect)**

**{**

**}**

И тело этой функции будет выполняться само по факту наступления совпадения чисел.

Нам нужна будет переменная. Объявим её глобально, в начале файла

Теперь, собственно, тело функции-обработчика. Здесь мы будем вызывать функцию LCD\_printchar. Затем будем наращивать на 1 переменную **i**. И чтобы она не ушла за пределы однозначного числа, будем её обнулять при данном условии

ISR (TIMER1\_COMPA\_vect)

{

**if(i>9) i=0;**

**LCD\_Printchar(i);**

**i++;**

}

**Пример использования прерывания по UART**

Прерывания по UART в ATmega16 позволяют контроллеру реагировать на состояния, связанные с передачей данных или приемом данных через UART. Когда происходит событие, которое инициирует прерывание, микроконтроллер переходит в режим обработки прерывания. В этом режиме микроконтроллер временно приостанавливает выполнение текущей программы и переходит к выполнению специальной подпрограммы, известной как обработчик прерывания.

Для использования прерываний по UART на ATmega16 вам следует выполнить следующие шаги:

1. Настройте аппаратный модуль UART для передачи и приема данных на нужной скорости передачи.
2. Включите прерывания по UART, установив соответствующий бит в регистре управления прерывания.
3. Настройте обработчик прерывания для аппаратного модуля UART.
4. Реализуйте код для обработки данных в обработчике прерывания.
5. В основной программе осуществите необходимые действия передачи и приема данных через UART.

Пример кода для настройки прерываний по UART на ATmega16 выглядит следующим образом:

// Обработчик прерывания для USART0

ISR(USART0\_RX\_vect)

{ // Код для обработки приема данных

}

void main(void)

{

// Настройка параметров USART0

//... рассматрвиается в разделе 3

// Включение прерываний по приему данных USART0

UCSR0B |= (1 << RXCIE0);

// Включение глобальных прерываний

sei();

while (1)

{ // Основная программа

}

}

**Пример инициализации UART**

int init\_UART(void)

{

// Установка скорости 9600

UBRRH=0; // UBRR=f/(16\*band)-1 f=8000000Гц band=9600,

UBRRL=51; // нормальный асинхронный двунаправленный режим работы

// RXC - завершение приёма

// TXC - завершение передачи

// UDRE - отсутствие данных для отправки

// FE - ошибка кадра

// DOR - ошибка переполнение буфера

// PE - ошибка чётности

// U2X - Двойная скорость

// MPCM - Многопроцессорный режим

// 76543210

UCSRA=0b00000000;

// RXCIE - прерывание при приёме данных

// TXCIE - прерывание при завершении передачи

// UDRIE - прерывание отсутствие данных для отправки

// RXEN - разрешение приёма

// TXEN - разрешение передачи

// UCSZ2 - UCSZ0:2 размер кадра данных

// RXB8 - 9 бит принятых данных

// TXB8 - 9 бит переданных данных

// 76543210

UCSRB=0b00011000; // разрешен приём и передача по UART

// URSEL - всегда 1

// UMSEL - режим:1-синхронный 0-асинхронный

// UPM1 - UPM0:1 чётность

// UPM0 - UPM0:1 чётность

// USBS - топ биты: 0-1, 1-2

// UCSZ1 - UCSZ0:2 размер кадра данных

// UCSZ0 - UCSZ0:2 размер кадра данных

// UCPOL- в синхронном режиме - тактирование

// 76543210

UCSRC=0b10000110; // 8-битовая посылка

}

# Обработка прерываний на языке Си для микроконтроллеров AtMega16

Процедура обработки прерывания выглядит следующим образом:

*//Процедура обработки прерывания по таймеру 2 по сравнению*

*#pragma vector = TIMER2\_COMP\_vect*

*\_\_interrupt void Timer2\_COMP(void)*

*{*

*TCNT2 = 0; //Сбросить счетчик*

*}*

\_\_interrupt – ключевое слово, которое используется при объявлении функций, являющихся обработчиками прерываний. До объявления функции необходимо указать также вектор прерывания. Все внутренние регистры и вектора прерываний объявлены в файле iom16.h.

Пример объявления векторов прерывания из файла iom16.h:

*/\*==============================\*/*

*/\* Interrupt Vector Definitions \*/*

*/\*==============================\*/*

*/\* NB! vectors are specified as byte addresses \*/*

*#define RESET\_vect (0x00)*

*#define INT0\_vect (0x04)*

*#define INT1\_vect (0x08)*

*#define TIMER2\_COMP\_vect (0x0C)*

*#define TIMER2\_OVF\_vect (0x10)*

*#define TIMER1\_CAPT\_vect (0x14)*

*#define TIMER1\_COMPA\_vect (0x18)*

*#define TIMER1\_COMPB\_vect (0x1C)*

*#define TIMER1\_OVF\_vect (0x20)*

*#define TIMER0\_OVF\_vect (0x24)*

*#define SPI\_STC\_vect (0x28)*

*#define USART\_RXC\_vect (0x2C)*

*#define USART\_UDRE\_vect (0x30)*

*#define USART\_TXC\_vect (0x34)*

*#define ADC\_vect (0x38)*

*#define EE\_RDY\_vect (0x3C)*

*#define ANA\_COMP\_vect (0x40)*

*#define TWI\_vect (0x44)*

*#define INT2\_vect (0x48)*

*#define TIMER0\_COMP\_vect (0x4C)*

*#define SPM\_RDY\_vect (0x50)*

# Использование таймеров AtMega16

ATmega16 имеет в своем составе три таймера-счетчика - два 8-ми разрядных таймера-счетчика Т0 и Т2, и один 16-ти разрядный - Т1. Таймеры T0 и T2 аналогичны по своей функциональности.

Таймеры Т0 и Т2 имеют в своем составе три регистра:

* счетный регистр TCNTn,
* регистр сравнения OCRn,
* конфигурационный регистр TCCRn.

Таймеры Т1 имеют в своем составе три регистра:

* счетный регистр TCNT1,
* регистры сравнения OCR1A, OCR1B,
* конфигурационные регистры TCCR1A, TCCR1B.

Кроме того, есть еще три регистра, относящиеся ко всем трем таймерам ATmega16:

* конфигурационный регистр TIMSK,
* статусный регистр TIFR,
* регистр специальных функций SFIOR.

Для более подробного описания см. раздел Таймеры.

Наиболее простой способ использования прерывания – прерывание по совпадению с регистром сравнения.  Регистры сравнения OCR1A/OCR1B/OCRn, содержимое которых в каждом машинном цикле сравнивается с текущим значением счётного регистра TCNTn.

Пример. Обработка прерываний от таймера T1 и Т2.

*#define ENABLE\_BIT\_DEFINITIONS*

*#include "iom16.h"*

*#pragma vector = TIMER2\_COMP\_vect*

*\_\_interrupt void Timer2\_COMP(void)*

*{*

*TCNT2 = 0; //Сброс счетчика*

*}*

*#pragma vector = TIMER1\_COMPA\_vect*

*\_\_interrupt void Timer1\_COMP(void)*

*{*

*TCNT1 = 0; //Сброс счетчика*

*}*

*void main()*

*{*

*TIMSK = (1 << OCIE2) | (1<<OCIE1A); //Разрешить прерывание по*

*//сравнению для таймера 1 и 2*

*SREG |= 1 << 7; // Глобальное разрешение прерываний.*

*//AtMega16 с кварцем 14.7456 МГц*

*//Таймер 2. Прерывание каждые 0.1мс(10КГц).*

*OCR2 = 22; // При частоте 14.7456 МГц и делителе 64, 0.1мс - 23 такта. OCR2=22!*

*TCCR2 |= (0 << CS22) | (1 << CS21) | (1 << CS20); //Делить частоту на 64*

*//Таймер 1. Прерывание каждые 50мс.*

*OCR1A = 719; // При частоте 14.7456 МГц и делителе 1024, 50мс - 720 тактов. OCR1A=719!*

*TCCR1B |= (1 << CS12) | (0 << CS11) | (1 << CS10);//Делить частоту на 1024*

*while(1);*

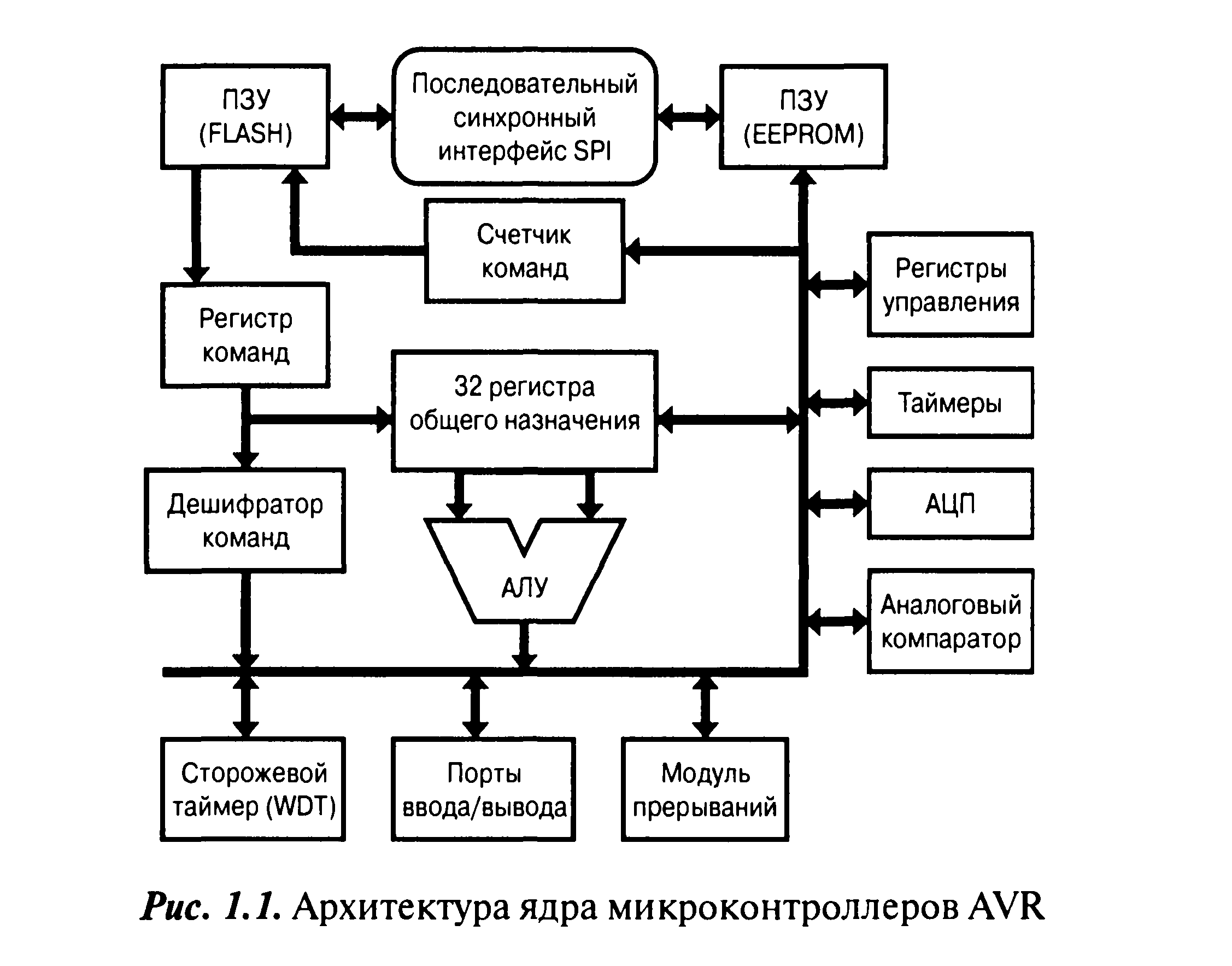
*}*

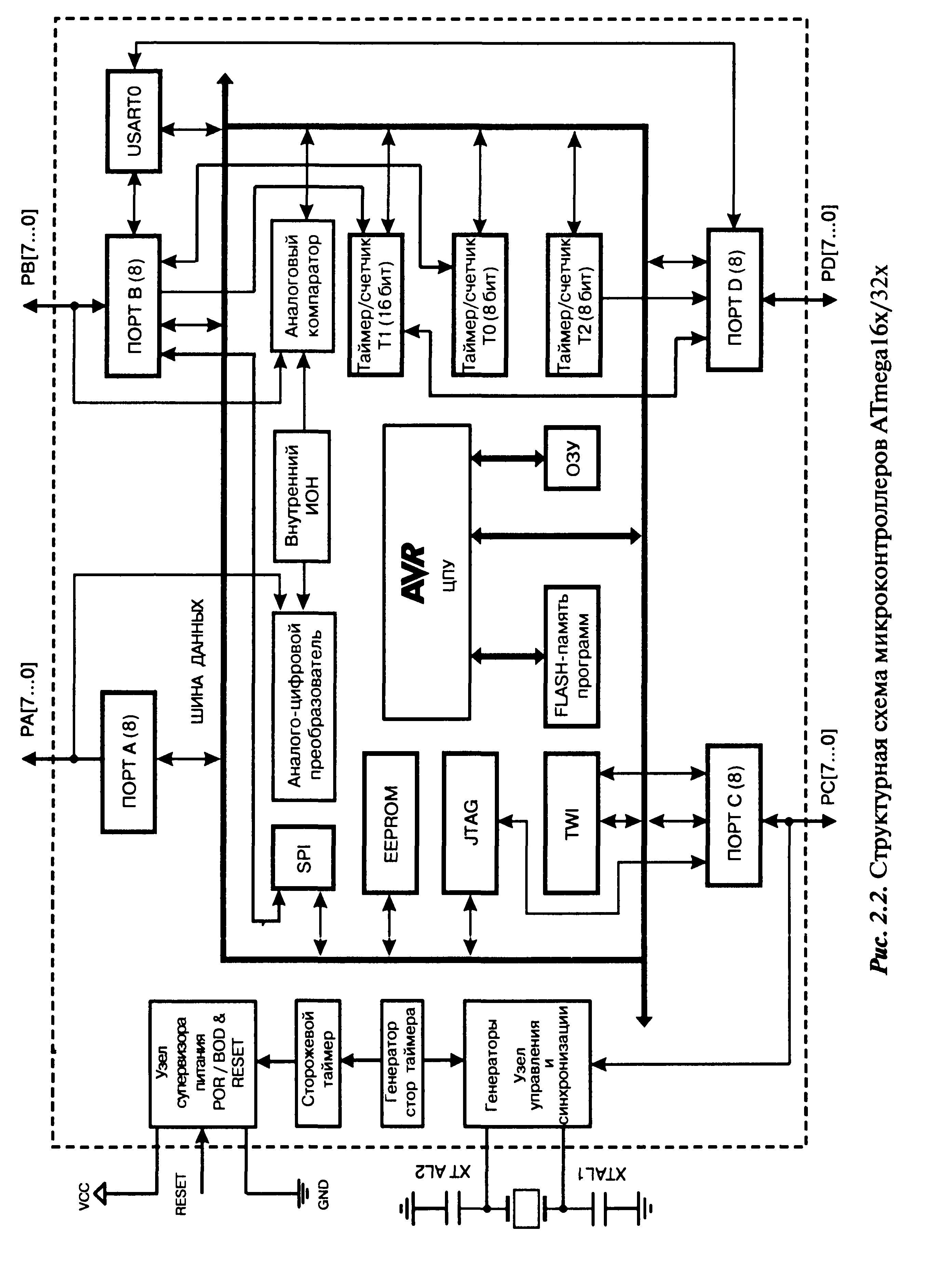
Раздел

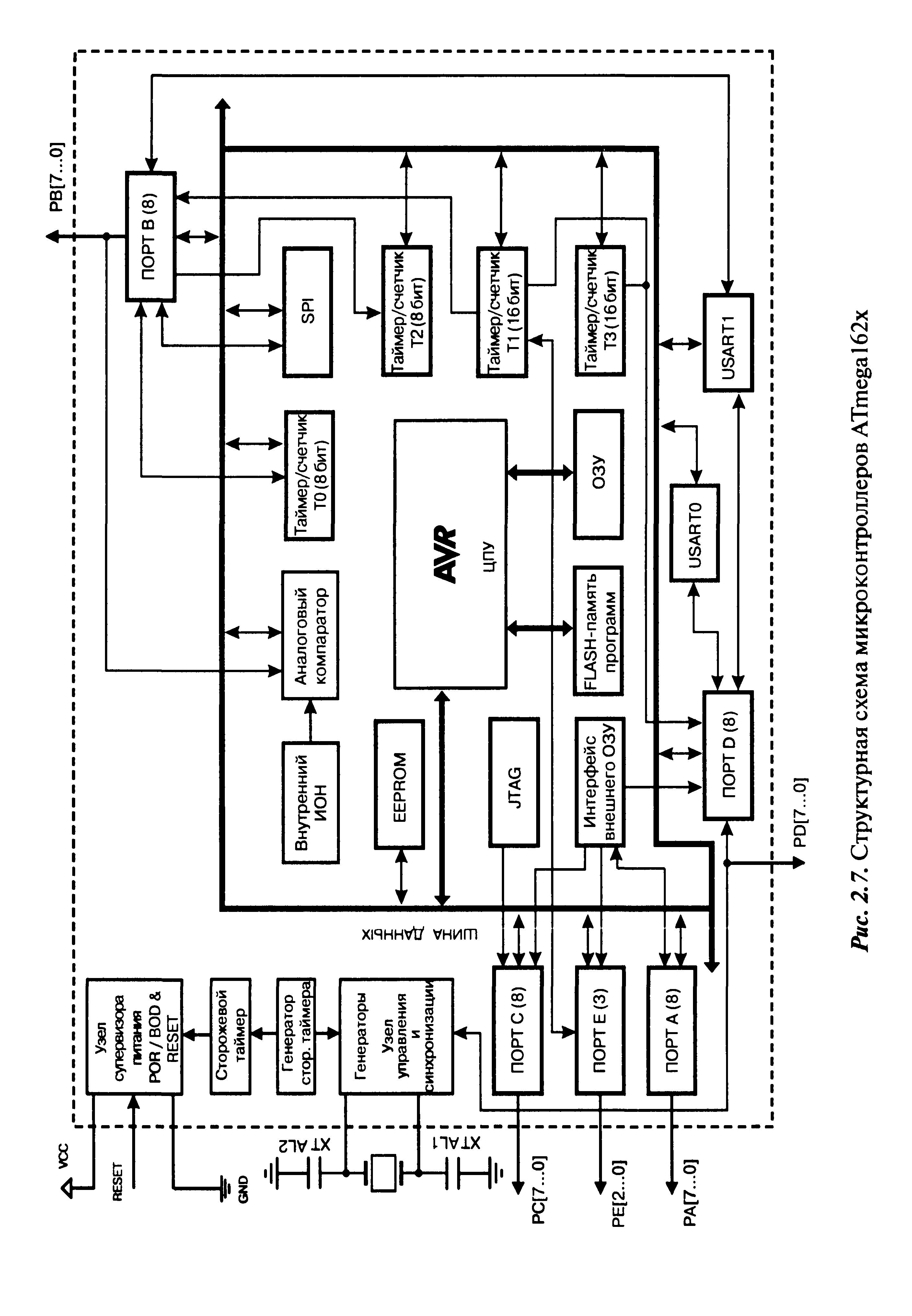
7

Приложения

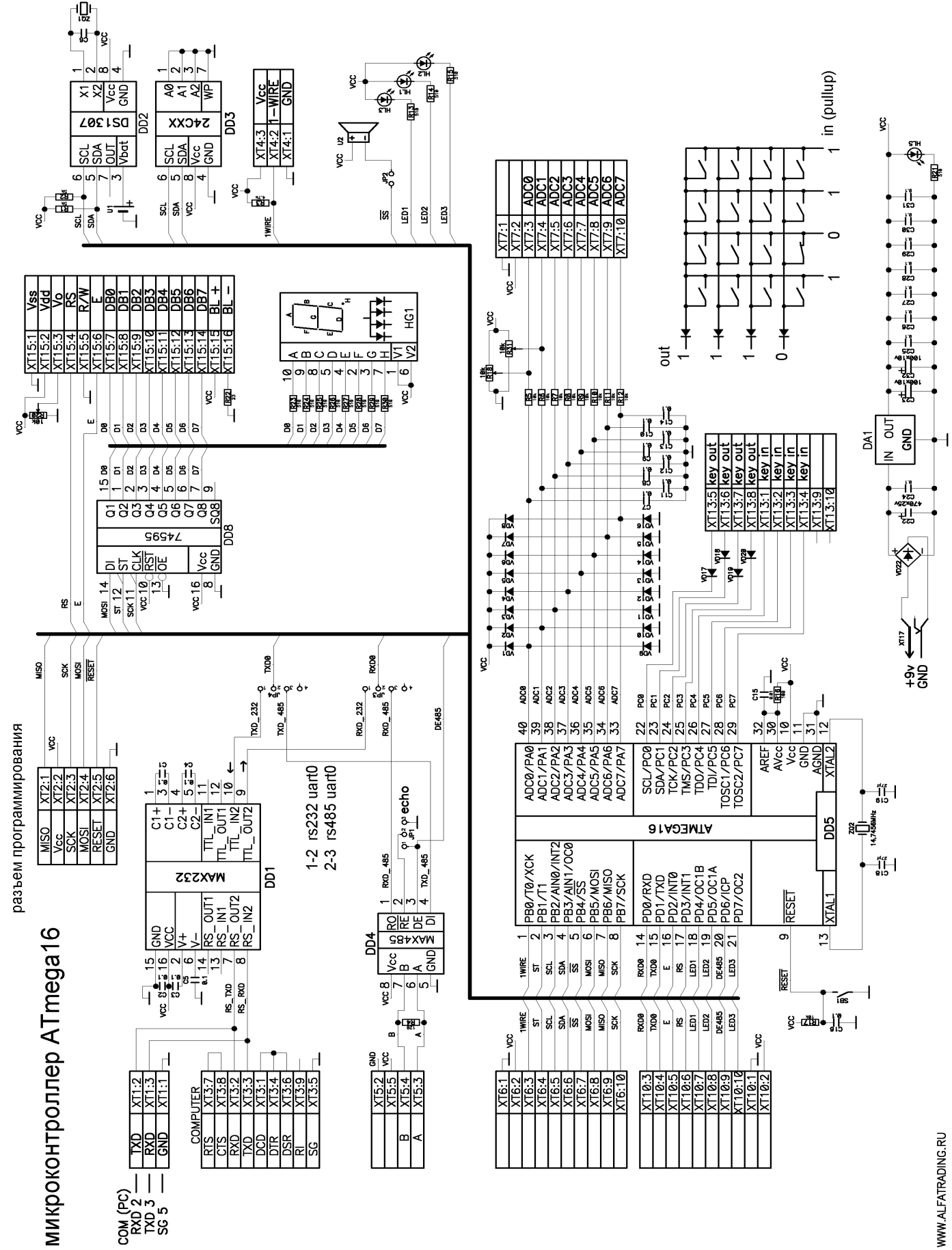
# Приложение 1. Архитектура МК

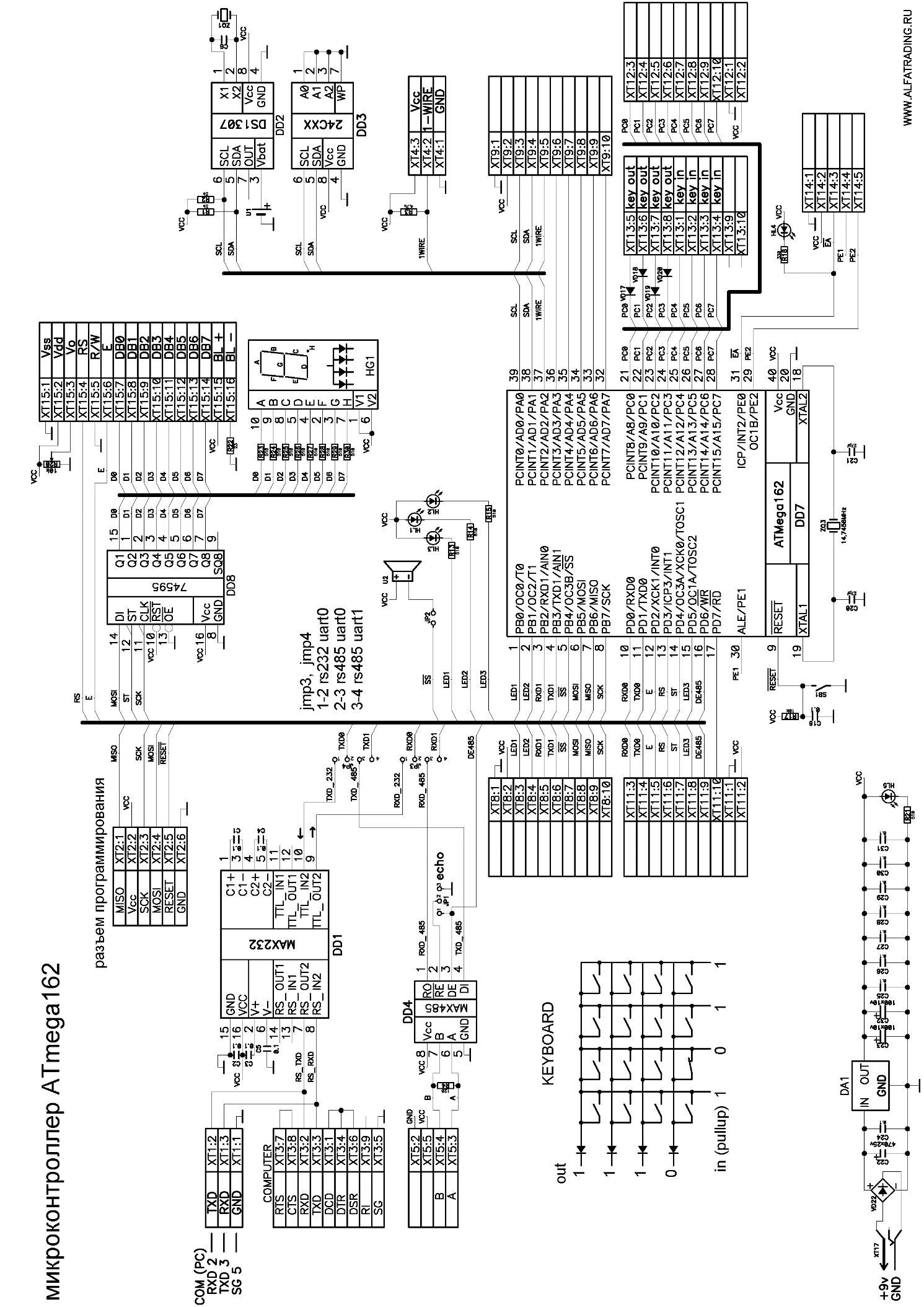






# Приложение 2. Электрическая схема стенда





# Приложение 3. Таблица ASCII кодов

**ASCII Table (7-bit)**

Decimal Octal Hex Binary Value

------- ----- --- ------ -----

000 000 000 00000000 NUL (Null char.)

001 001 001 00000001 SOH (Start of Header)

002 002 002 00000010 STX (Start of Text)

003 003 003 00000011 ETX (End of Text)

004 004 004 00000100 EOT (End of Transmission)

005 005 005 00000101 ENQ (Enquiry)

006 006 006 00000110 ACK (Acknowledgment)

007 007 007 00000111 BEL (Bell)

008 010 008 00001000 BS (Backspace)

009 011 009 00001001 HT (Horizontal Tab)

010 012 00A 00001010 LF (Line Feed)

011 013 00B 00001011 VT (Vertical Tab)

012 014 00C 00001100 FF (Form Feed)

013 015 00D 00001101 CR (Carriage Return)

014 016 00E 00001110 SO

015 017 00F 00001111 SI

016 020 010 00010000 DLE (Data Link Escape)

017 021 011 00010001 DC1 (XON) (Device Control 1)

018 022 012 00010010 DC2 (Device Control 2)

019 023 013 00010011 DC3 (XOFF)(Device Control 3)

020 024 014 00010100 DC4 (Device Control 4)

021 025 015 00010101 NAK

022 026 016 00010110 SYN (Synchronous Idle)

023 027 017 00010111 ETB (End of Trans. Block)

024 030 018 00011000 CAN (Cancel)

025 031 019 00011001 EM

026 032 01A 00011010 SUB

027 033 01B 00011011 ESC (Escape)

028 034 01C 00011100 FS (File Separator)

029 035 01D 00011101 GS

030 036 01E 00011110 RS (Request to Send)

031 037 01F 00011111 US

032 040 020 00100000 SP (Space)

033 041 021 00100001 !

034 042 022 00100010 "

035 043 023 00100011 #

036 044 024 00100100 $

037 045 025 00100101 %

038 046 026 00100110 &

039 047 027 00100111 '

040 050 028 00101000 (

041 051 029 00101001 )

042 052 02A 00101010 \*

043 053 02B 00101011 +

044 054 02C 00101100 ,

045 055 02D 00101101 -

046 056 02E 00101110 .

047 057 02F 00101111 /

048 060 030 00110000 0

049 061 031 00110001 1

050 062 032 00110010 2

051 063 033 00110011 3

052 064 034 00110100 4

053 065 035 00110101 5

054 066 036 00110110 6

055 067 037 00110111 7

056 070 038 00111000 8

057 071 039 00111001 9

058 072 03A 00111010 :

059 073 03B 00111011 ;

060 074 03C 00111100 <

061 075 03D 00111101 =

062 076 03E 00111110 >

063 077 03F 00111111 ?

064 100 040 01000000 @

065 101 041 01000001 A

066 102 042 01000010 B

067 103 043 01000011 C

068 104 044 01000100 D

069 105 045 01000101 E

070 106 046 01000110 F

071 107 047 01000111 G

072 110 048 01001000 H

073 111 049 01001001 I

074 112 04A 01001010 J

075 113 04B 01001011 K

076 114 04C 01001100 L

077 115 04D 01001101 M

078 116 04E 01001110 N

079 117 04F 01001111 O

080 120 050 01010000 P

081 121 051 01010001 Q

082 122 052 01010010 R

083 123 053 01010011 S

084 124 054 01010100 T

085 125 055 01010101 U

086 126 056 01010110 V

087 127 057 01011111 W

088 130 058 01011000 X

089 131 059 01011001 Y

090 132 05A 01011010 Z

091 133 05B 01011011 [

092 134 05C 01011100 \

093 135 05D 01011101 ]

094 136 05E 01011110 ^

095 137 05F 01011111 \_

096 140 060 01100000 `

097 141 061 01100001 a

098 142 062 01100010 b

099 143 063 01100011 c

100 144 064 01100100 d

101 145 065 01100101 e

102 146 066 01100110 f

103 147 067 01100111 g

104 150 068 01101000 h

105 151 069 01101001 i

106 152 06A 01101010 j

107 153 06B 01101011 k

108 154 06C 01101100 l

109 155 06D 01101101 m

110 156 06E 01101110 n

111 157 06F 01101111 o

112 160 070 01110000 p

113 161 071 01110001 q

114 162 072 01110010 r

115 163 073 01110011 s

116 164 074 01110100 t

117 165 075 01110101 u

118 166 076 01110110 v

119 167 077 01110111 w

120 170 078 01111000 x

121 171 079 01111001 y

122 172 07A 01111010 z

123 173 07B 01111011 {

124 174 07C 01111100 |

125 175 07D 01111101 }

126 176 07E 01111110 ~

127 177 07F 01111111 DEL

# 

# Приложение 4. Тестовая программа на C

#include "BEEP.H"

#include "DELAY.H"

#include "SPI.H"

#include "LCD\_HD44780.H"

#include "I2C.H"

#include "KEYBOARD.H"

#include "LED7.H"

#include "UART.H"

#include "DS1307.H"

#include "DS1820.H"

#include "AT24CXX.H"

#include "CONVERT.H"

#include "SHIFT.H"

#include "ADC.H"

#include <mega16.h>

// Строковые константы, которые хранятся в flash-памяти, чтобы избежать

// переполнения оперативной памяти

flash char UartMessage[] = " -= STEND RK-10 =-\r\n"

" Hardware ver. 1.0, Firmware ver. 2.0\r\n";

flash unsigned char LcdMessageLine1[] = { 45, 61, 67, 84, 69, 72, 224, 32, 32, 80, 75, 45, 49, 48, 61, 45, 0 };

flash unsigned char LcdMessageLine2[] = " Ver. 2.0";

flash char timeStr[] = "Time: ";

flash char dateStr[] = "Date: ";

flash char enterAddressStr[] = "Enter address(0..255): ";

flash char enterValueStr[] = "Enter value(0..255): ";

flash char valueStr[] = "Value: ";

flash char temperatureStr[] = "Temperature: ";

flash char enterHourStr[] = "Enter hour: ";

flash char enterMinuteStr[] = "Enter minute: ";

flash char enterSecondStr[] = "Enter second: ";

flash char enterDayStr[] = "Enter day: ";

flash char enterMonthStr[] = "Enter month: ";

flash char enterYearStr[] = "Enter year: ";

flash char sensorNotConnectedStr[] = "Sensor is not connected";

// Задание 1: пользователь нажимает на кнопки(цифры) телефонной клавиатуры и

// соответствующая цифра появляется на LCD-экране

// при нажатии \* экран очищается

// при нажатии # осуществляется выход из задания

void ExecuteTask1()

{

unsigned char key = 0;

char keyname;

unsigned char keyCount = 0;

LCD\_clrscr();

while( key != KEY\_BAR )

{

key = KEY\_getkey();

if( !key )

continue;

keyname = KEY\_getkeyname( key );

if( keyname >= '0' && keyname <= '9' )

{

LCD\_printchar( keyname );

keyCount++;

if( keyCount == LCD\_WIDTH )

LCD\_gotoxy( 0, 1 );

}

else if( key == KEY\_ASTERISK )

{

LCD\_clrscr();

keyCount = 0;

}

KEY\_waitforkeyrelease();

}

}

// Задание 2: пользователь нажимает на кнопки(цифры) телефонной клавиатуры и

// соответствующая цифра появляется на семисегментном индикаторе

// при нажатии # осуществляется выход из задания

void ExecuteTask2()

{

unsigned char key = 0;

while( key != KEY\_BAR )

{

key = KEY\_getkey();

if( !key )

continue;

LED7\_setdigit( KEY\_getkeyname( key ) - '0' );

KEY\_waitforkeyrelease();

}

}

// Задание 3: при нажатии на кнопки телефонной клавиатуры раздается

// сигнал зуммера

// при нажатии # осуществляется выход из задания

void ExecuteTask3()

{

unsigned char key = 0;

while( key != KEY\_BAR )

{

key = KEY\_getkey();

if( !key )

continue;

Beep( 100 );

KEY\_waitforkeyrelease();

}

}

// Задание 4: при нажатии на кнопки телефонной клавиатуры(1,2,3) меняет

// состояние соответствующий этой кнопке LED-индикатор

// при нажатии # осуществляется выход из задания

void ExecuteTask4()

{

unsigned char key = 0;

while( key != KEY\_BAR )

{

key = KEY\_getkey();

if( !key )

continue;

switch( KEY\_getkeyname( key ) ) {

case '1':

LED1\_PORT = !LED1\_PORT;

break;

case '2':

LED2\_PORT = !LED2\_PORT;

break;

case '3':

LED3\_PORT = !LED3\_PORT;

break;

}

KEY\_waitforkeyrelease();

}

}

// Задание 5: в UART выводится текущее время и дата, полученные из

// микросхемы DS1307

void ExecuteTask5()

{

char time[9], date[9];

DS1307\_gettime(time);

DS1307\_getdate(date);

UART\_sendstring\_flash( timeStr );

UART\_sendstring( time );

UART\_sendcrlf();

UART\_sendstring\_flash( dateStr );

UART\_sendstring( date );

UART\_sendcrlf();

}

// Задание 6: через UART-интерфейс пользователь вводит новое время и дату,

// которые устанавливаются в микросхему DS1307

void ExecuteTask6()

{

unsigned char hour, minute, second;

unsigned char day, month, year;

UART\_sendstring\_flash( enterHourStr );

hour = UART\_receivevalue();

UART\_sendstring\_flash( enterMinuteStr );

minute = UART\_receivevalue();

UART\_sendstring\_flash( enterSecondStr );

second = UART\_receivevalue();

DS1307\_settime( hour, minute, second );

UART\_sendstring\_flash( enterDayStr );

day = UART\_receivevalue();

UART\_sendstring\_flash( enterMonthStr );

month = UART\_receivevalue();

UART\_sendstring\_flash( enterYearStr );

year = UART\_receivevalue();

DS1307\_setdate( day, month, year );

}

// Задание 7: через UART-интерфейс пользователь вводит адрес и значение ячейки,

// которые заносятся во внешнюю EEPROM память AT24CXX

void ExecuteTask7()

{

unsigned char addr, value;

UART\_sendstring\_flash( enterAddressStr );

addr = UART\_receivevalue();

UART\_sendstring\_flash( enterValueStr );

value = UART\_receivevalue();

AT24CXX\_write( addr, value );

}

// Задание 8: через UART-интерфейс пользователь вводит адрес ячейки,

// значение которой считывается из внешней EEPROM памяти AT24CXX

// и выводиться в UART

void ExecuteTask8()

{

unsigned char addr;

UART\_sendstring\_flash( enterAddressStr );

addr = UART\_receivevalue();

UART\_sendstring\_flash( valueStr );

UART\_sendvalue( AT24CXX\_read( addr ) );

UART\_sendcrlf();

}

// Задание 9: в UART-интерфейс выводиться текущее значение температуры,

// полученное из микрочхемы DS1820

void ExecuteTask9()

{

char strTemp[ 5 ];

if( !DS1820\_startconverttemp() )

{

UART\_sendstring\_flash( sensorNotConnectedStr );

UART\_sendcrlf();

return;

}

while( !DS1820\_getconvertstatus() );

if( DS1820\_gettemperature( strTemp ) )

{

UART\_sendstring\_flash( temperatureStr );

UART\_sendstring( strTemp );

UART\_sendcrlf();

}

}

// Функция выводит на LCD приветственное сообщение

void OutputStartLCDMessage()

{

LCD\_print\_flash( LcdMessageLine1 );

LCD\_gotoxy( 0, 1 );

LCD\_print\_flash( LcdMessageLine2 );

}

// Функция-обработчик прерывания, возникающего, когда во входном буфере UART

// имеются данные.

// В этой функции мы получаем номер задания и выполняем его.

#pragma vector = UART\_RXC\_vect

\_\_interrupt void UART0\_RX\_interrupt()

{

unsigned char task;

task = UART\_receivevalue();

switch( task ) {

case 1:

ExecuteTask1();

break;

case 2:

ExecuteTask2();

break;

case 3:

ExecuteTask3();

break;

case 4:

ExecuteTask4();

break;

case 5:

ExecuteTask5();

break;

case 6:

ExecuteTask6();

break;

case 7:

ExecuteTask7();

break;

case 8:

ExecuteTask8();

break;

case 9:

ExecuteTask9();

break;

}

UART\_sendstring( "Enter task number: " );

}

// Функция автономного режима, то есть режима, когда нет подключения к

// компьютеру через UART-интерфейс.

// На LCD-экран выводятся:

// в первой строке - текущая температура и текущее время

// во второй строке - текущая дата

// LED индикаторы меняют свое состояние

void AutonomousMode()

{

LCD\_clrscr();

static char strTemp[ 5 ] = { 0 };

LCD\_gotoxy( 0, 0 );

switch( DS1820\_gettemperature( strTemp ) ) { // Получаем температуру

case 0: // Сенсор не подключен

strTemp[ 0 ] = 0;

break;

case 2: // Данные не готовы

DS1820\_startconverttemp();

break;

}

LCD\_print( strTemp );

char time[9];

DS1307\_gettime(time);

LCD\_gotoxy( 6, 0 );

LCD\_print( time );

LCD\_gotoxy( 0, 1 );

char keyname = KEY\_getkeyname( KEY\_getkey() );

if( keyname != 0 )

LCD\_printchar( keyname );

LCD\_gotoxy( 3, 1 );

char strADC1[ 5 ];

ShortToString( ADC\_get( 0 ), strADC1 );

LCD\_print( strADC1 );

LCD\_gotoxy( 9, 1 );

char strADC2[ 5 ];

ShortToString( ADC\_get( 1 ), strADC2 );

LCD\_print( strADC2 );

static unsigned char ledCount = 0;

if( ledCount % 3 == 0 ) // Чтобы индикаторы слишком

{ // часто не мигали, меняем

LED1\_PORT = !LED1\_PORT; // их состояние один раз на

LED2\_PORT = !LED2\_PORT; // три вызова функции

LED3\_PORT = !LED3\_PORT;

}

ledCount++;

}

// Главная функция программы

// Производится инициализация устройств, вывод сообщения на LCD-экран, а также

// вывод сообщения в UART-интерфейс

// Также включается зуммер на 0.5 с

// В бесконечном цикле выполняется функция автономного режима

int main()

{

LED1\_DDR = 1;

LED2\_DDR = 1;

LED3\_DDR = 1;

I2C\_init(); // Инициализируем I2C интерфейс

SPI\_init(); // Инициализируем SPI интерфейс

LCD\_init(); // Инициализируем LCD дисплей

LCD\_clrscr(); // Очищаем экран LCD дисплея

UART\_init( CALC\_UBRR( 57600 ) );

\_\_enable\_interrupt();

DS1820\_startconverttemp();

UART\_sendstring\_flash( UartMessage );

OutputStartLCDMessage();

UART\_sendstring( "Enter task number: " );

Beep( 500 );

DELAY\_MS( 1000 );

while(1)

{

AutonomousMode();

SHIFT\_write(0);

DELAY\_MS( 200 );

}

}

# Приложение 5. Тексты домашних заданий

**Вариант 1.**

***Задача 1.***

При запуске программы на экран терминала выводится сообщение «Задача №1».

При нажатии на кнопку 1 клавиатуры стенда стенда зуммер дает сигналы длительностью 100 мс с паузой 100 мс, на экран терминала выводится текущее значение температуры, полученное с датчика DS1820. При повторном нажатии клавиши 1 зуммер отключается.

При нажатии на кнопку 2 клавиатуры стенда зуммер дает сигналы длительностью 200 мс с паузой 100 мс, на экран терминала выводится текущее время и дата, полученные с RTC DS1307. При повторном нажатии клавиши 2 зуммер отключается.

Нажатие на другие клавиши игнорируется.

***Задача 2.***

При запуске программы на экран терминала выводится сообщение «Задача №2».

При нажатии клавиши 1 PC на ЖКИ стенда выводится текущее значение температуры, полученное с датчика DS1820.

При нажатии клавиши 2 PC значение температуры выводится на экран терминала.

При нажатии клавиши 3 PC происходит очистка ЖКИ

АЦП: вывести показания на LCD экран. Светодиодами LED1 LED2 LED3 досчитать до 7 в двоичном коде ориентируясь на показания АЦП по формуле «показания/10» при более высоком значении зуммер дает сигналы длительностью 100 мс с паузой 100 мс.

Нажатие на другие клавиши игнорируется.

**Вариант 2.**

***Задача 1.***

При запуске программы на экран терминала выводится сообщение «Задача №1».

При нажатии на кнопки 0-7 клавиатуры PC загораются светодиоды LED1, LED2, LED3 соответственно двоичному представлению этой цифры, также соответствующая цифра выводится на семисегментный индикатор. При повторном нажатии все светодиоды и индикатор гаснут. Во время нажатия остальных клавиш зуммер дает 3 сигнала длительностью 100 мс с паузой 100 мс, после паузы 300 мс. – 3 сигнала длительностью 300 мс с паузой 100 мс, после паузы 300 мс - 3 сигнала длительностью 100 мс с паузой 100 мс.

Начальное состояние светодиодов - выключены.

***Задача 2.***

При запуске программы на экран терминала выводится сообщение «Задача №2».

При нажатии на кнопку 1 клавиатуры стенда зуммер дает сигналы длительностью 100 мс с паузой 100 мс и звучит 2 секунды, на ЖКИ выводится текущее значение температуры, полученное с датчика DS1820.

При нажатии на кнопку 2 клавиатуры стенда зуммер дает сигналы длительностью 200 мс с паузой 100 мс и звучит 4 секунды, на ЖКИ стенда в первой строке выводится текущее время, во второй дата полученные с RTC DS1307.

При нажатии на кнопку 3 клавиатуры зуммер дает сигналы длительностью 300 мс с паузой 100 мс и звучит 6 секунд, происходит очистка ЖКИ.

Нажатие на другие клавиши игнорируется.

Выводим показания АЦП на LCD экран, при совпадении показаний АЦП с температурным датчиком (+/- 10) мигаем светодиодами с периодом 2 секунды.

**Вариант 3.**

***Задача 1.***

При запуске программы на экран терминала выводится сообщение «Задача №1».

При нажатии на любую кнопку клавиатуры стенда соответствующий символ выводится на экран PC и зуммер дает 3 группы сигналов по 1 сигналу длительностью 300 мс с паузой 100 мс и 1 сигналу 100 мс, пауза между группами 300 мс.

Вывести показания АЦП на LCD экран в вольтах, при разнице показаний двух АЦП меньше 5, вывести на LCD экран слово «ровно».

***Задача 2.***

При запуске программы на экран терминала выводится сообщение «Задача №2».

При нажатии на кнопку 1 клавиатуры стенда зуммер дает сигналы длительностью 100 мс с паузой 100 мс и звучит 2 секунды, затем начинает мигать светодиод LED1 с периодом 1 секунда. При повторном нажатии зуммер звучит аналогично и светодиод гаснет.

При нажатии на кнопку 2 клавиатуры стенда зуммер дает сигналы длительностью 200 мс с паузой 300 мс и звучит 3 секунды, затем начинает мигать светодиод LED2 с периодом 2 секунды. При повторном нажатии зуммер звучит аналогично и светодиод гаснет.

При нажатии на кнопку 3 клавиатуры стенда зуммер дает сигналы длительностью 300 мс с паузой 100 мс и звучит 4 секунды, затем начинает мигать светодиод LED3 с периодом 3 секунды. При повторном нажатии зуммер звучит аналогично и светодиод гаснет.

Нажатие на другие клавиши игнорируется.

Начальное состояние светодиодов – выключены.

**Вариант 4.**

***Задача 1.***

При запуске программы на экран терминала выводится сообщение «Задача №1».

При нажатии на любую кнопку клавиатуры PC зуммер дает 3 сигнала длительностью 100 мс с паузой 100 мс, после паузы 300 мс – 3 сигнала длительностью 300 мс с паузой 100 мс, после паузы 300 мс - 3 сигнала длительностью 100 мс с паузой 100 мс, и символ выводится на ЖКИ стенда. При нажатии же клавиш 0-9 соответствующие цифры, помимо звукового сигнала, выводятся еще на семисегментный индикатор.

***Задача 2.***

При запуске программы на экран терминала выводится сообщение «Задача №2.

При нажатии клавиши 1 клавиатуры PC предлагается ввести время и дату, которые затем устанавливаются как текущие в RTC DS1307. При нажатии клавиши 2 на ЖКИ стенда в первой строке выводится текущее время, во второй дата. При нажатии клавиши 3 происходит очистка ЖКИ.

Нажатие на другие клавиши игнорируется.

Выводить на экран показания двух АЦП в вольтах, показания второго АЦП кратные 10 выводить на семисегментный индикатор. При показаниях больше 99 включить зумер.

**Вариант 5.**

***Задача 1.***

При запуске программы на экран терминала выводится сообщение «Задача №1».

При нажатии на любую цифровую кнопку клавиатуры стенда соответствующий символ выводится на ЖКИ стенда и семисегментный индикатор. При нажатии остальных зуммер дает 3 группы сигналов по 1 сигналу длительностью 300 мс с паузой 100 мс и 1 сигналу 100 мс, пауза между группами 300 мс. Реализовать удаление последнего введенного символа на экране ЖКИ кнопкой “\*”, а на семисегментном индикаторе при этом должна выводиться цифра, стоящая последней перед удаленной.

Вывести показания двух АЦП на ЖКИ, а так же их разность, при разности больше 50 включить LED1, больше 100, LED2, больше 150 LED3, больше 200 включить все три светодиода.

***Задача 2.***

При запуске программы на экран терминала выводится сообщение «Задача №2».

При нажатии на кнопку 1 клавиатуры стенда загорается светодиод LED1, зуммер дает сигналы длительностью 100 мс с паузой 100 мс и звучит 4 секунды. При повторном нажатии светодиод отключается и снова звучит такой же звуковой сигнал.

При нажатии на кнопку 2 клавиатуры стенда загорается светодиод LED2, зуммер дает сигналы длительностью 200 мс с паузой 100 мс и звучит 3 секунды. При повторном нажатии светодиод отключается и снова звучит такой же звуковой сигнал.

При нажатии на кнопку 3 клавиатуры стенда загорается светодиод LED3, зуммер дает сигналы длительностью 300 мс с паузой 100 мс и звучит 2 секунды. При повторном нажатии светодиод отключается и снова звучит такой же звуковой сигнал.

Начальное состояние светодиодов – выключены.

Нажатие на другие клавиши игнорируется.

# Приложение 6. Справочная информация

## Расположение пинов в DIP корпусе контроллеров Atmega 16

******

***Расположение пинов в DIP корпусе контроллеров Atmega 162***



## Описание выводов микроконтроллера ATmega 16

|  |  |  |  |
| --- | --- | --- | --- |
| **Обозначение** | **Номер вывода, DIP** | **Тип вывода** | **Описание** |
| XTAL1 | 13 | I | Вход тактового генератора |
| XTAL2 | 12 | O | Выход тактового генератора |
| RESET (инв.) | 9 | I | Вход сброса |
| **Порт A.8-битный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами** | | | |
| PA0(ADC0) | 40 | I/O | 0-й бит порта A  Вход АЦП |
| PA1(ADC1) | 39 | I/O | 1-й бит порта A  Вход АЦП |
| PA2(ADC2) | 38 | I/O | 2-й бит порта A  Вход АЦП |
| PA3(ADC3) | 37 | I/O | 3-й бит порта A  Вход АЦП |
| PA4(ADC4) | 36 | I/O | 4-й бит порта A  Вход АЦП |
| PA5(ADC5) | 35 | I/O | 5-й бит порта A  Вход АЦП |
| PA6(ADC6) | 34 | I/O | 6-й бит порта A  Вход АЦП |
| PA7(ADC7) | 33 | I/O | 7-й бит порта A  Вход АЦП |
| **Порт B.8-битный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами** | | | |
| PB0(T0/XCK) | 1 | I/O | 0-й бит порта B  Вход внешнего тактового сигнала  таймера/счётчика T0  Вход/выход внешнего тактового сигнала USART |
| PB1(T1) | 2 | I/O | 1-й бит порта B  Вход внешнего тактового сигнала  таймера/счётчика  T1 |
| PB2(AIN0/INT2) | 3 | I/O | 2-й бит порта B  Неинвертирующий вход компаратора  Вход внешнего прерывания |
| PB3(AIN1/OC0) | 4 | I/O | 3-й бит порта B  Инвертирующий вход компаратора  Выход таймера/счётчика  T0 |
| PB4(SS(инв))) | 5 | I/O | 4-й бит порта B  Выбор Slave устройства на шине SPI |
| PB5(MOSI) | 6 | I/O | 5-й бит порта B  Выход(Master) или вход(Slave) данных модуля SPI |
| PB6(MISO) | 7 | I/O | 6-й бит порта B  Вход(Master) или выход(Slave) данных модуля SPI |
| PB7(SCK) | 8 | I/O | 7-й бит порта B  Выход(Master) или вход(Slave) тактового данных модуля SPI |
| Порт C.8-битный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами | | | |
| PC0(SCL) | 22 | I/O | 0-й бит порта C  Вход/выход тактового сигнала модуля TWI |
| PC1(SDA) | 23 | I/O | 1-й бит порта C  Вход/выход тактового сигнала модуля TWI |
| PC2(TCK) | 24 | I/O | 2-й бит порта C  Тактовый сигнал JTAG |
| PC3(TMS) | 25 | I/O | 3-й бит порта C  Выбор режима JTAG |
| PC4(TDO) | 26 | I/O | 4-й бит порта C  Выход данных JTAG |
| PC5(TDI) | 27 | I/O | 5-й бит порта C  Вход данных JTAG |
| PC6(TOSC1) | 28 | I/O | 6-й бит порта C  Вывод для подключения резонатора к таймеру/счётчику T2 |
| PC7(TOSC2) | 29 | I/O | 7-й бит порта C  Вывод для подключения резонатора к таймеру/счётчику T2 |
| **Порт D.8-битный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами** | | | |
| PD0(RXD) | 14 | I/O | 0-й бит порта D  Вход USART |
| PD1(TXD) | 15 | I/O | 1-й бит порта D  Выход USART |
| PD2(INT0) | 16 | I/O | 2-й бит порта D  Вход внешнего прерывания |
| PD3(INT1) | 17 | I/O | 3-й бит порта D  Вход внешнего прерывания |
| PD4(OC1B) | 18 | I/O | 4-й бит порта D  Выход B таймера/счётчика T1 |
| PD5(OC1A) | 19 | I/O | 5-й бит порта D  Выход A таймера/счётчика T1 |
| PD6(ICP1) | 20 | I/O | 6-й бит порта D  Вход захвата таймера/счётчика T1 |
| PD7(OC2) | 21 | I/O | 7-й бит порта D  Выход таймера/счётчика T2 |
| AREF | 32 | P | Вход опорного напряжения для АЦП |
| AVCC | 30 | P | Вывод источника питания АЦП |
| VCC | 10 | P | Вывод источника питания |
| GND | 11,31 | P | Общий вывод |

## Описание выводов микроконтроллера ATmega 162

|  |  |  |  |
| --- | --- | --- | --- |
| **Обозначение** | **Номер вывода, DIP** | **Тип вывода** | **Описание** |
| XTAL1 | 19 | I | Вход тактового генератора |
| XTAL2 | 18 | O | Выход тактового генератора |
| RESET(инв.) | 9 | I | Вход сброса |
| Порт A.8-битный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами | | | |
| PA0(AD0/PCINT0) | 39 | I/O | 0-й бит порта A  0-й бит мультиплексированной ША/ШД для внешнего ОЗУ  Вход внешнего прерывания по изменению сигнала |
| PA1(AD1/PCINT1) | 38 | I/O | 1-й бит порта A  1-й бит мультиплексированной ША/ШД для внешнего ОЗУ  Вход внешнего прерывания по изменению сигнала |
| PA2(AD2/PCINT2) | 37 | I/O | 2-й бит порта A  2-й бит мультиплексированной ША/ШД для внешнего ОЗУ  Вход внешнего прерывания по изменению сигнала |
| PA3(AD3/PCINT3) | 36 | I/O | 3-й бит порта A  3-й бит мультиплексированной ША/ШД для внешнего ОЗУ  Вход внешнего прерывания по изменению сигнала |
| PA4(AD4/PCINT4) | 35 | I/O | 4-й бит порта A  4-й бит мультиплексированной ША/ШД для внешнего ОЗУ  Вход внешнего прерывания по изменению сигнала |
| PA5(AD5/PCINT5) | 34 | I/O | 5-й бит порта A  5-й бит мультиплексированной ША/ШД для внешнего ОЗУ  Вход внешнего прерывания по изменению сигнала |
| PA6(AD6/PCINT6) | 33 | I/O | 6-й бит порта A  6-й бит мультиплексированной ША/ШД для внешнего ОЗУ  Вход внешнего прерывания по изменению сигнала |
| PA7(AD7/PCINT7) | 32 | I/O | 7-й бит порта A  7-й бит мультиплексированной ША/ШД для внешнего ОЗУ  Вход внешнего прерывания по изменению сигнала |
| **Порт B.8-битный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами** | | | |
| PB0(T0/OC0) | 1 | I/O | 0-й бит порта B  Вход внешнего тактового сигнала  таймера/счётчика  T0  Выход таймера/счётчикаТ0 |
| PB1(T1/OC2) | 2 | I/O | 1-й бит порта B  Вход внешнего тактового сигнала  таймера/счётчика  T1  Выход таймера/счётчикаТ2 |
| PB2(AIN0/INT2) | 3 | I/O | 2-й бит порта B  Неинвертирующий вход компаратора  Вход UART1 |
| PB3(AIN0/RXD1) | 4 | I/O | 3-й бит порта B  Инвертирующий вход компаратора  Выход UART1 |
| PB4(SS(инв)/OC3B) | 5 | I/O | 4-й бит порта B  Выбор Slave устройства на шине SPI  Выход B таймера/счётчика Т3 |
| PB5(MOSI) | 6 | I/O | 5-й бит порта B  Выход(Master) или вход(Slave) данных модуля SPI |
| PB6(MISO) | 7 | I/O | 6-й бит порта B  Вход(Master) или выход(Slave) данных модуля SPI |
| PB7(SCK) | 8 | I/O | 7-й бит порта B  Выход(Master) или вход(Slave) тактового данных модуля SPI |
| **Порт C.8-битный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами** | | | |
| PC0(А8/PCINT8) | 21 | I/O | 0-й бит порта C  8-й бит ША для внешнего ОЗУ  Вход внешнего прерывания по изменению сигнала |
| PC1(А9/PCINT9) | 22 | I/O | 1-й бит порта C  9-й бит ША для внешнего ОЗУ  Вход внешнего прерывания по изменению сигнала |
| PC2(А10/PCINT10) | 23 | I/O | 2-й бит порта C  10-й бит ША для внешнего ОЗУ  Вход внешнего прерывания по изменению сигнала |
| PC3(А11/PCINT11) | 24 | I/O | 3-й бит порта C  11-й бит ША для внешнего ОЗУ  Вход внешнего прерывания по изменению сигнала |
| PC4(А12/TCK/PCINT8) | 25 | I/O | 4-й бит порта C  12-й бит ША для внешнего ОЗУ  Тактовый сигнал JTAG  Вход внешнего прерывания по изменению сигнала |
| PC5(А13/TMS/PCINT13) | 26 | I/O | 5-й бит порта C  13-й бит ША для внешнего ОЗУ  Выбор режима JTAG  Вход внешнего прерывания по изменению сигнала |
| PC6(А14/TDO/PCINT14) | 27 | I/O | 6-й бит порта C  14-й бит ША для внешнего ОЗУ  Выход данных JTAG  Вход внешнего прерывания по изменению сигнала |
| PC7(А14/TDI/PCINT14) | 28 | I/O | 7-й бит порта C  15-й бит ША для внешнего ОЗУ  Выход данных JTAG  Вход внешнего прерывания по |
| **Порт D.8-битный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами** | | | |
| PD0(RXD0) | 10 | I/O | 0-й бит порта D  Вход USART0 |
| PD1(TXD0) | 11 | I/O | 1-й бит порта D  Выход USART0 |
| PD2(XCK1/INT0) | 12 | I/O | 2-й бит порта D  Вход/выход внешнего тактового сигнала USART1  Вход внешнего прерывания |
| PD3(ICP3/INT1) | 13 | I/O | 3-й бит порта D  Вход захвата таймера/ счётчика Т3  Вход внешнего прерывания |
| PD4(XCK0/OC3A/TOSC1) | 14 | I/O | 4-й бит порта D  Вход/выход внешнего тактового сигнала USART0  Выход А таймера/ счётчика Т3  Вывод для подключения резонатора к таймеру/счётчику Т2 |
| PD5(OC1A/TOSC2) | 15 | I/O | 5-й бит порта D  Выход А таймера/ счётчика Т1  Вывод для подключения резонатора к таймеру/счётчику Т2 |
| PD6(WR(инв)) | 16 | I/O | 6-й бит порта D  Строб записи во внешнее ОЗУ |
| PD7(RD(инв)) | 17 | I/O | 7-й бит порта D  Строб чтения из внешнего ОЗУ |
| **Порт Е.3-битный двунаправленный порт ввода/вывода с внутренними подтягивающими резисторами** | | | |
| PE0(ICP1/INT2) | 31 | I/O | 0-й бит порта Е  Вход захвата таймера/ счётчика Т1 |
| PE1(ALE) | 30 | I/O | 1-й бит порта Е  Строб адреса внешнего ОЗУ |
| PE2(OC1B) | 29 | I/O | 2-й бит порта Е  Выход B таймера/ счётчика Т1 |
| VCC | 40 | P | Вывод источника питания |
| GND | 20 | P | Общий вывод |

# Список используемой литературы

1. А.В. Евстифеев. Микроконтроллеры AVR семейства Mega. Руководство пользователя. - М.: Издательский дом <<Додека – XXI>>,2007.
2. К. Бойт. Цифровая электроника. – М.: Техносфера, 2007.
3. Документация. http://www.alldatasheet.com