ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
ВЯТСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
Факультет автоматики и вычислительной техники
Кафедра автоматики и телемеханики

УСТРОЙСТВО МИКРОКОНТРОЛЛЕРА ATMEGA16

Теоретический материал для выполнения
Лабораторного практикума

Дисциплина “Микропроцессорные устройства систем управления”

Для специальности 220201 "Управление и информатика в технических системах",

3 курс дневного отделения

4 курс заочного отделения

Киров 2015

УДК

С о с т а в и т е л ь кандидат технических наук, доцент Л.А. Шабалин,

Д.Г. Гарш

кафедра АТ

Р е ц е н з е н т

Подп. в печ. Усл. печ. л. Зак. Тир. Бесплатно

ПРИП ВятГУ, 610000, г.Киров, ул.Московская, 36

Содержание

Стр.

1 ПАМЯТЬ И СИСТЕМА КОМАНД ATMEGA16A

1.1 Характеристики ATMega16A

Микроконтроллер (МК) ATMega16A семейства Mega фирмы Atmel – представляет собой 8-разрядную однокристальную микро-ЭВМ с упрощенной системой команд- RISC. Структура контроллера представлена на рисунке 1.1.

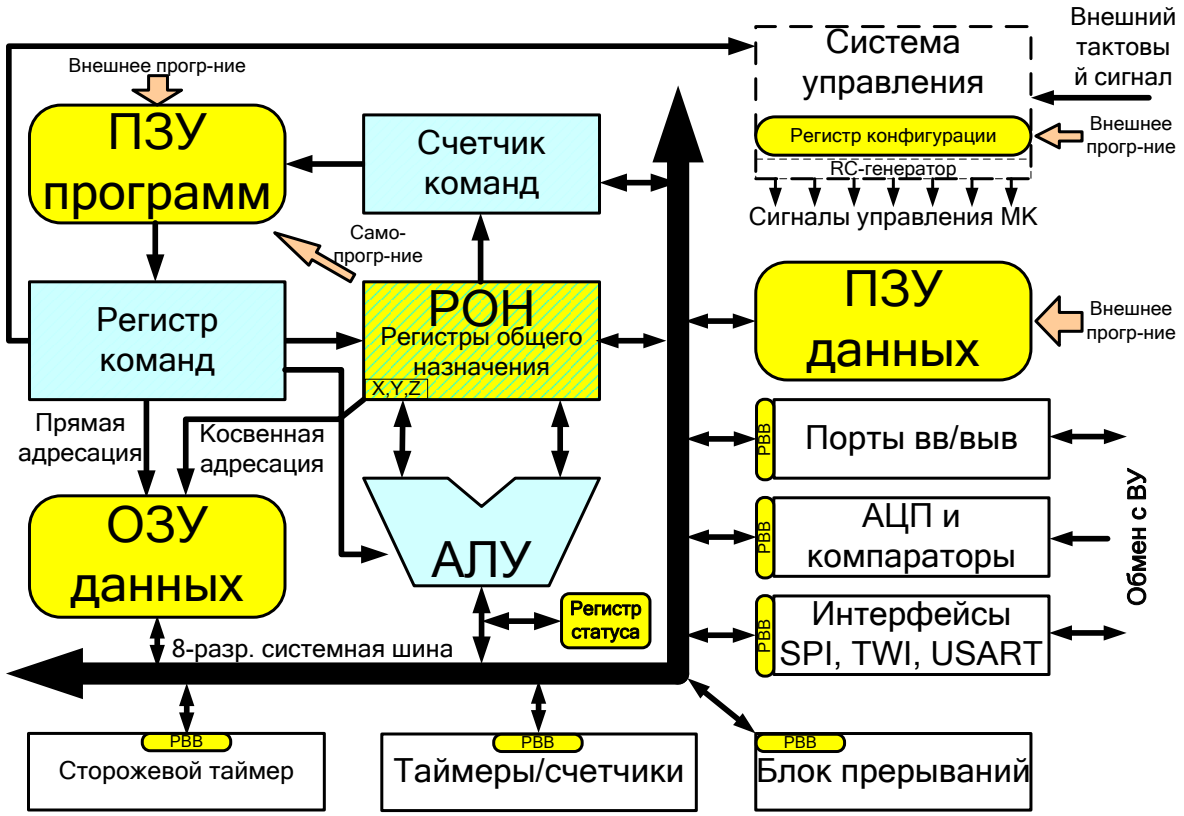


Рисунок 1.1 – Обобщенная структурная схема МК семейства Mega

Все функциональные части связывает между собой 8-битная шина данных. На одном кристалле размещаются:

- Центральной процессорное устройство состоящее из арифметико-логического устройства (АЛУ), регистров общего назначения (РОН), счетчика команд и регистра команд;
- Постоянная память программ (FLASH)объемом 16Кбайт;
- Оперативная память данных (SRAM) объемом 1Кбайт;
- Постоянная память данных (EEPROM) объёмом 512 байт;
- Распределенная система управления, вырабатывающая управляю-

щие сигналы для всех функциональных частей МК.

Помимо этого разработчику предлагается богатый набор периферии:

- Подсистема прерываний;
- Три таймера/счетчика (один 16-разрядный и два 8-разрядных);
- Сторожевой таймер;
- Встроенный RC-генератор;

Для связи с внешними устройствами предусмотрены:

- Четыре настраиваемых 8-битных порта ввода/вывода;
- 8-канальный 10-разрядный аналого-цифровой преобразователь и встроенный аналоговый компаратор;
- Аппаратная реализация распространенных последовательных интерфейсов USART, SPI, I^2C(TWI);

МК позволяет производить программирование постоянной памяти FLASH, EEPROM и ячеек конфигурации по различным интерфейсам: параллельный, ISP, JTAG, DebugWire. Также возможно самопрограммирование FLASH памяти программ (под управлением программы, которая находится в той же самой памяти программ).

Частота работы МК определяется параметрами внешнего тактового сигнала (если не выбран внутренний RC-генератор) и может достигать 16 МГц. МК изготавливается по КМОП-технологии (на основе полевых транзисторов), благодаря чему обеспечивается низкий уровень энергопотребления.

1.2 Центральное процессорное устройство

В сердце МК лежит 8-битное центральное процессорное устройство (ЦПУ), построенное на принципах RISC-архитектуры. ЦПУ отвечает за выполнение команд, записанных во FLASH - память.

1.2.1 Арифметико-логическое устройство

Арифметико-логическое устройство (АЛУ) выполняет все вычислительные операции в микропроцессорной системе. АЛУ поддерживает арифметические и логические операции между регистрами, а также между константой и регистром. Кроме того, АЛУ поддерживает действия с одним регистром. Для ветвления программы поддерживаются инструкции условных и безусловных переходов и вызовов процедур, позволяющих непосредственно адресоваться в пределах адресного пространства.

1.2.2 Регистры общего назначения

Непосредственно к АЛУ подключены 32 регистра общего назначения (РОН) (см. рисунок 1.2). Эти регистры служат для обработки данных и вычислительных операций. Большая часть команд в качестве своих параметров использует именно РОНы. Возможности этих регистров различны:

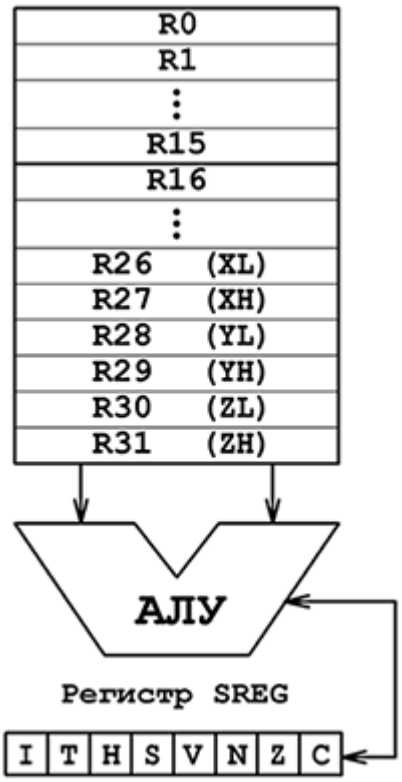


Рисунок 1.2 – Арифметико – логическое устройство

- Первая половина регистрового файла (R0-R15) не может быть использована для выполнения операций с константой;
- Вторая половина регистрового файла (R16-R31) может быть использована для любых операций, кроме операций в которых требуется адресация SRAM;
- Во второй половине регистрового файла имеются специфические регистровые пары X (R26, R27), Y (R28, R29) и Z (R30, R31) которые могут применяться для задания адресов ячеек SRAM в различных командах пересылки данных Указатель Z, кроме этого, необходим при чтении/записи FLASH-память программ.

1.2.3 Регистр статуса

АЛУ любого микропроцессора неразрывно связано с регистром флагов программы, который у ATMega16A имеет название SREG (Status Register) и расположен в ОЗУ по адресу 0x5F. Регистр статуса содержит информацию о результате только что выполненной арифметической инструкции. Данная информация может использоваться для ветвления программы по условию. Флаги этого регистра в большинстве случаев позволяют отказаться от использования инструкций сравнения, делая код программы более компактным и быстрым.

Описание битов SREG приведено в таблице 1.1. Все флаги SREG доступны как для чтения, так и для записи.

Таблица 1.1 – Флаги регистра статуса SREG

Номер бита в регистре	Название флага	Описание
0	C	Флаг переноса. Устанавливается в 1 если в результате операции произошел выход за границы байта.
1	Z	Флаг нуля. Устанавливается в 1 если результат операции равен нулю.
2	N	Флаг отрицательного результата. В этот флаг копируется содержимое 7-мого MSB результата операции.
3	V	Флаг переполнения в дополнительном коде. Устанавливается в 1 при переполнении разрядной сетки знакового результата.
4	S	Флаг знака. Содержимое флага определяется как N XOR V.
5	H	Флаг половинного переноса. Устанавливается в 1 если в результате операции сложения/вычитания произошел перенос/заем из 3-его бита в 4-тый.
6	T	Флаг хранения копируемого бита. Флаг используется в качестве источника и приемника командами копирования битов bld и bst соответственно.
7	I	Флаг глобального разрешения прерываний. При установке в 1 этого флага происходит разрешение всех немаскируемых прерываний.

1.2.4 Счетчик команд (PCL, PCH)

13-разрядный счетчик команд (PC) предназначен для хранения адреса ячейки программной памяти, из которой считывается команда. Аппаратно PC представляет собой суммирующий загружаемый счетчик. Разрядность счетчика равна 13 (число разрядов счетчика команд определяется емкостью программной памяти). Так как шина данных микроконтроллера имеет 8 разрядов, то счетчик команд состоит физически из двух частей: PCL (младший восьмиразрядном регистр), PCH (старший пятиразрядный).

В процессе выполнения программы счетчик всегда указывает на текущую выполняемую команду. При считывании кода команды значение счетчика увеличивается на один или два (в зависимости от длины команды). В командах передачи управления (переходы, вызовы подпрограмм и возвраты из подпрограмм) счетчик может загружаться новым значением, указанным командой. В него записывается новое значение адреса программ. После сброса микроконтроллера в счетчик команд записывается ноль.

1.2.5 Регистр команд (IR)

Этот регистр предназначен для хранения кода текущей команды, которая помещается в него при считывании ячейки программной памяти. Загрузка регистра команд производится по 16-разрядной шине команд. После загрузки регистра команд микроконтроллер работает в соответствии с этой командой.

1.3 Организация памяти

Всего в МК можно выделить 6 типов памяти (см. рисунок 1.2): память программ, 5 типов памяти данных (РОНы, РВВ, SRAM, EEPROM и биты конфигурации). В целях достижения максимальной производительности и параллелелизма у AVR-микроконтроллеров используется Гарвардская архитектура с раздельной памятью и шинами программ и данных.

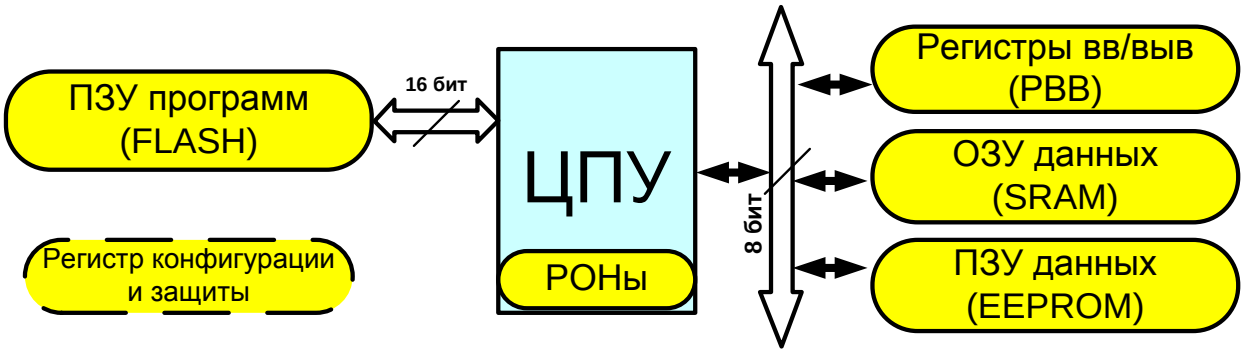


Рисунок 1.3 – Организация памяти МК

1.3.1 Память программ (FLASH - ROM)

Память программ предназначена для хранения последовательности команд, управляющих функционированием микроконтроллера. Поскольку МК имеет 16 битную систему команд, то и объем памяти удобнее указывать не в байтах, а в 16-битных словах. Объем памяти составляет 8192 слова (или 16 Кбайт). Структура флэш-памяти МК приведена на рисунке 1.4.

Самая первая команда располагается по адресу 0x0000, после сброса МК начинает работу именно с этой команды.

Память программ выполнена по технологии FLASH, и допускает многократное стирание и запись информации. Гарантированное число циклов перезаписи составляет не менее 10 тыс. циклов при типовом значении 100 тыс. циклов. Для работы МК необходимо, что бы в его память была занесена программа. Это делается с помощью специального устройства – программатора. Память имеет страничную организацию. Размер страницы флэш-памяти 64 слова у ATmega16. Соответственно количество страниц 128. При программировании микроконтроллера данные сначала загружаются в буфер страницы и только затем заносятся непосредственно в память программ. Прошивка всех ячеек страницы при этом осуществляется одновременно.

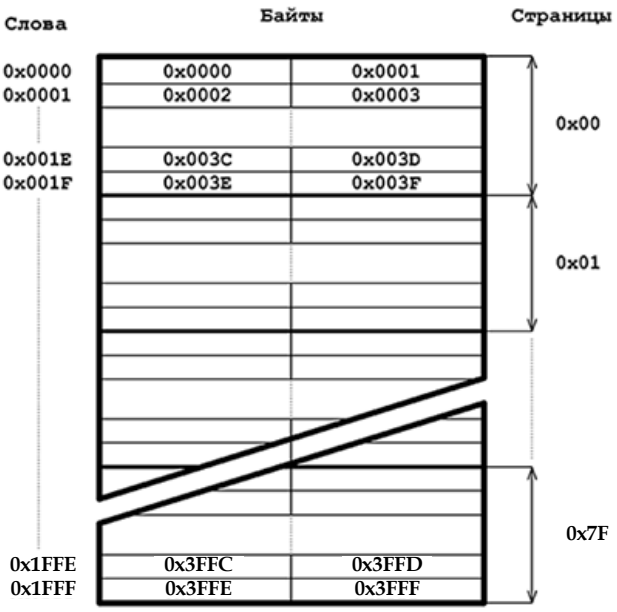


Рисунок 1.4 – Организация памяти программ

МК имеет возможность самопрограммирования, т. е. самостоятельного изменения содержимого своей памяти программ. Эта особенность позволяет создавать на его основе очень гибкие системы, алгоритм работы которых будет меняться самим микроконтроллером в зависимости от каких-либо внутренних условий или внешних событий. Для обеспечения самопрограммирования память программ разделена на 2 секции: секцию прикладной программы и секцию загрузчика (см. рисунок 1.5). Изменение памяти программ осуществляется программой загрузчиком, которая обязательно должна располагаться в одноименной секции. Размер секции загрузчика задается в регистре конфигурации (рассматривается далее) и лежит в интервале 128 – 1024 слов.

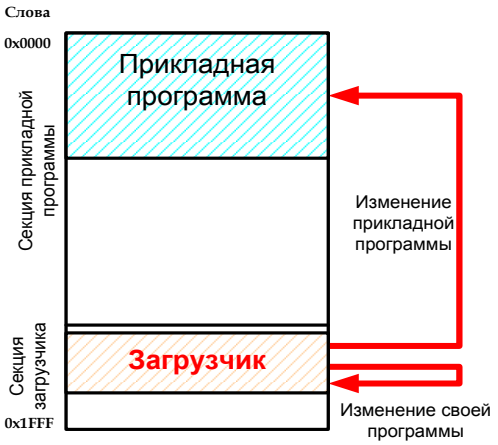


Рисунок 1.5 – Расположение программы загрузчика в памяти программ

1.3.2 Регистровая память

Регистровая память включает регистры общего назначения (РОН) и служебные регистра ввода/вывода (РВВ). И те и другие расположены в адресном пространстве ОЗУ, но не являются его частью. РОНы и РВВ имеют индивидуальный набор адресов, который могут использовать только команды определенного типа. Карта адресов оперативной памяти МК приведена на рисунке 1.6.

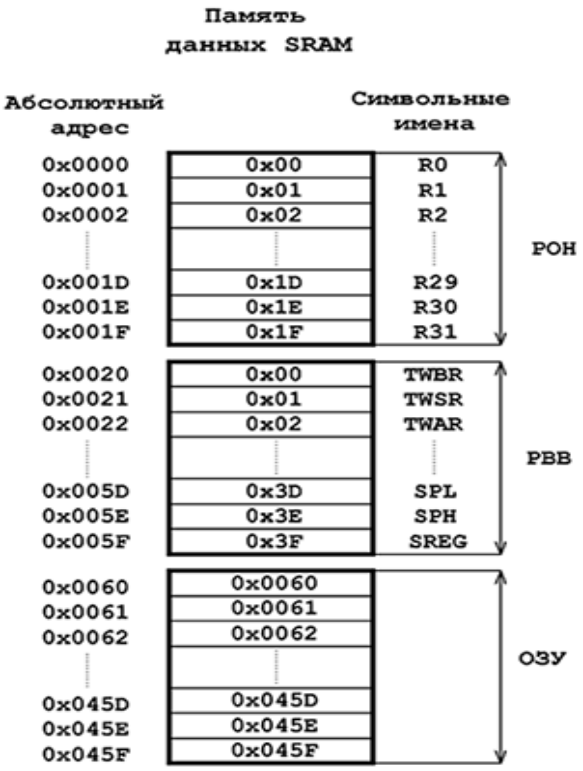


Рисунок 1.6 – Организация оперативной памяти данных

1.3.3 Регистры общего назначения

В самом начале адресного пространства находятся 32 регистра общего назначения с символьными именами R0…R31. Эти регистры подключены непосредственно к арифметико-логическому устройству (АЛУ). В индивидуальном и абсолютном адресных пространствах РОНы занимают адреса 0x0000…0x001F.

1.3.4 Регистры ввода/вывода

Регистры ввода/вывода предназначены для доступа ко всем внутренним устройствам МК и являются их неотъемлемой частью. В области регистров ввода – вывода расположены различные служебные регистры (указатель стека, регистр состояния SREG и др.), а также регистры встроенных периферийных устройств (таймеры, подсистема прерываний и др.). Управление МК сводится к обыкновенным процедурам чтения/записи этих регистров.

Эти регистры не имеют конкретного места расположения а распределены по аппаратным ресурсам МК доступ к ним осуществляется по системной шине. В адресном пространстве РВВ занимают следующие 64 ячейки памяти.

Описание и адреса РВВ приведено в приложении В.

1.3.5 Оперативная память данных (ОЗУ)

После РВВ, начинается адресное пространство внутреннего ОЗУ, основным предназначением которого является хранение переменных прикладной программы. Ячейки ОЗУ имеют байтовый формат и занимают адреса с 0x0060 по 0x045F в абсолютном адресном пространстве (объем - 1КБайт). Число циклов чтения и записи в RAM не ограничено, но при отключении питающего напряжения вся информация теряется.

Для программиста отличие между регистрами и оперативной памятью состоит в том, что с регистрами можно производить любые операции (арифметические, логические, битовые), а для работы с оперативной памятью разрешены операции только записи или чтения из этих регистров.

1.3.6 Программный стек

В ОЗУ может быть организован программный стек (см. рисунок 1.7) - область ОЗУ для формально безадресного хранения данных. Для адресации текущего элемента (вершины стека) используется указатель стека SP (Stack Pointer). Это двухбайтовый регистр ввода/вывода SPH:SPL с адресами (0x3D:0x3E). Аппаратно указатель стека представляет собой реверсивный счетчик. После загрузки в ячейку стека содержимое SP автоматически уменьшается на единицу, а перед чтением из стека происходит увеличение SP на единицу. Стек «растет» в сторону младших адресов. По сигналу сброса устанавливается SP=0.

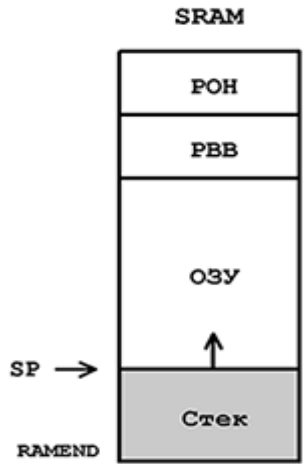


Рисунок 1.7 – Организация стека в ОЗУ

Программист должен самостоятельно определить местоположение стека в самом начале программы. С точки зрения максимальной его глубины, вершину стека нужно поместить в самом конце SRAM (значение 0x045F).

Стек обычно используется для сохранения и восстановления адресов возврата из подпрограмм. Когда микропроцессор встречает одну из инструкций вызовов rcall/call/ecall/icall/eicall, то адрес следующего за ними слова в памяти программ аппаратно копируется в стек. В момент выхода из подпрограммы по команде ret адрес возврата восстанавливается из стека в программный счетчик. При сохранении каждого байта содержимое SP уменьша-

ется не единицу, а при восстановлении, соответственно увеличивается. Помимо сохранения адресов возврата стек позволяет сохранять любые данные специально предназначенными для этого командами push Rr (загрузка в стек) и pop Rd (выгрузка из стека).

1.3.7 Энергонезависимая память данных (EEPROM)

Для долговременного хранения различной информации, которая может изменяться в процессе функционирования микроконтроллерной системы, используется EEPROM-память. Энергонезависимая память данных микроконтроллера EEPROM имеет объем 512 байт и отдельное адресное пространство 0x0 – 0x1FF Этот тип памяти, доступный программе микроконтроллера непосредственно в ходе ее выполнения, удобен для хранения промежуточных данных, различных констант, коэффициентов, серийных номеров, ключей и т.п. EEPROM может быть загружена извне как через SPI интерфейс, так и с помощью обычного программатора. Число циклов стирание/запись - не менее 100 тыс.

1.3.8 Ячейки конфигурации и защиты

В этом регистре хранятся различные параметры конфигурации МК, которые должны быть определены до его запуска. Операции чтения и записи с этими регистрами возможны только с программатора.

Различают два типа таких ячеек это:

- Ячейки защиты (Lock Bits) помогают защитить программное обеспечение МК от корыстного использования третьими лицами (дают возможность заблокировать доступ к содержимому памяти программ и данных);
- Ячейки конфигурации (Fuse Bits) определяют различные параметры МК, такие как источник тактовой частоты, задержка времени после включения питания, уровень срабатывания детектора пониженного напряжения и др.

Более подробную информацию об этих регистрах и способах их программирования можно узнать в документации производителя.

1.4 Система команд и способы адресации

МК имеет развитую систему команд, она насчитывает 131 инструкцию. Полный перечень команд МК приведен в приложении A. Перечень сокращений используемых при описании команд приведен в приложении D.

1.4.1 Формат команды

Любая команда МК состоит из двух частей:

- Код операции (КОП) – обязательная часть, содержит указание какую операцию выполнять в АЛУ. При разработке на ассемблере вместо кода операции указывается её символьное обозначение – мнемонику.
- Операнды – определяют данные, с которыми АЛУ будет работать при выполнении команды. Это могут быть адреса ячеек памяти МК с которыми будет производиться операция, константа, номер бита и д.р.

В качестве примера рассмотрим команду сложения ADD Rd,Rr. Формат команды приведен на рисунке 1.8. Символы ddddd определяют адрес регистра хранения первого операнда. Символы rrrrr определяют адрес регистра хранения второго операнда. Выполняемая операция условно обозначается следующим образом: Rd ← Rd + Rr – сложение значений в регистрах Rd и Rr и помещение результата в регистр Rd. Разряды команды 15-10 - поле кода операции. Часть этих разрядов подключена к АЛУ и предназначена для выбора устройства в АЛУ (в данном случае в АЛУ будет выбран сумматор).

Номера Разрядов	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Команда	0	0	0	0	1	1	r	d	d	d	d	d	r	r	r	r
	Разряды кода операции (КОП)							Адрес регистра Rd (операнда 1)					Адрес регистра Rr (операнда 2)			

Рисунок 1.8 – Формат команды ADD

1.4.2 Способы адресации

Каждая команда языка ассемблера «сообщает» процессору выполняемую операцию и методы доступа к операндам или место размещения операндов. Обрабатываемые в процессоре микроконтроллера данные (операнды) могут располагаться в регистрах РОН, в ячейках памяти или в регистрах ввода/вывода. В вычислительной системе с общей шиной данных, к которой относится и микроконтроллер, любой из этих источников данных имеет уникальный адрес. Способ задания адреса операнда (способ адресации) определяется командой. Название способа адресации определяется, в основном, местом размещения операнда.

В зависимости от того в каком виде в команде хранится адрес различают два способа адресации: **прямую** и **косвенную**. В первом случае адрес ячейки

задан явно (это может быть адрес регистра или адрес ячейки памяти), а во втором – адрес ячейки памяти находится в одном из регистров-указателей (16-разрядные регистры X,Y,Z). У МК оба способа адресации имеют множество вариаций.

1.4.1 Группы команд

В зависимости от выполняемых функций все команды микроконтроллера можно разделить на пять групп:

- арифметические и логические команды;
- команды операций с битами;
- команды пересылки данных;
- инструкции перехода;
- команды ветвления;
- инструкции управления МК.

Полный список приведен в приложении А. Примеры использования приведены в справочной литературе (см. приложение Е). Рассмотрим основные команды. При рассмотрении команд используются регистры R16, R17, но взамен этих команд могут использоваться любые команды.

Арифметические и логические команды

```
;-----------------------------------------------------------
;занести константу в регистр  Rd = K
;Допустимые операнды: R - R16..R31, K - 0..255
        ldi     R16, 0xFF
        ldi     R17, 0x00
        ldi     R18, 0x02
        ldi     R19, 0x10
;-----------------------------------------------------------
;Сложение двухбайтовых чисел  Rd1:Rd0 = Rd1:Rd0 + Rr1:Rr0
        add     R16, R18  ; сложение без учета переноса R6 = R16+R18
;Для связи байтов используется флаг переноса C в регистре SREG
;флаг устанавливается, когда разрядность суммы превысит 8 бит
        adc     R17, R19   ; сложение с учетом переноса R17 <- R17+R19+C

;-----------------------------------------------------------
;Сложение слова (регистровой пары) с константой          Rd1:Rd0= Rd1:Rd0 + K
;Допустимые операнды: R - R25:R24, R27:R26, R29:R28, R31:R30
;                                                        K - 0..63
;Эта операция удобна для организации счетчика
.equ    K = 0x0F
        ldi     R24, 0xFA
        ldi     R25, 0x00
        ;В качестве операнда указывается младший байт пары
        adiw    R24, K          ; R25:R24 = R25:R24 + K

;-----------------------------------------------------------
;Вычитание двухбайтовых чисел  Rd1:Rd0 = Rd1:Rd0 - Rr1:Rr0
```

```
        ldi     R16, 0x0F
        ldi     R17, 0xFF
        ldi     R18, 0x10
        ldi     R19, 0x01

        sub     R16, R18  ; вычитание R16 = R16-R18
;Для связи байтов используется флаг переноса C в регистре SREG
;флаг устанавливается, когда разрядность суммы окажется отрицательной
        sbc     R17, R19   ; сложение с учетом переноса R17 <- R17-R19-C

;-------------------------------------------------------------
;Вычитание константы из регистра Rd = Rd - K
;Допустимые операнды: R - R16..R31, K - 0..255
        ldi     R16, 0x0F
        subi    R16, 0x01       ; R16 = R16 - 0x0F

;В случае если K>Rd устанавливаются флаги SREG: S(знак),
                                        ; N(отрицательный результат) и C(перенос)
        ldi     R16, 0x0F
        subi    R16, 0x10       ; R16 = R16 - 0x10

;Вариант сложения регистра с константой: Rd = Rd - (-K)
        ldi     R16, 0x0F
        subi    R16, -0x01      ; R16 = R16 - (-0x01)

;-------------------------------------------------------------
;Умножение регистров (без знака) Rd = Rd * Rr
;результат помещается в регистровую пару R1:R0
        ldi     R16, 0xFF
        ldi     R17, 0x02
        mul     R16, R17                ; R1:R0 = R16*R17 = 0x01FE

;Умножение регистров (беззнаковое число на знаковое) Rd = Rd * -Rr
;результат помещается в регистровую пару R1:R0
;Допустимые операнды: Rd - R16..R31 Rd - положительное
;                                                       Rr - R16..R31
        ldi     R16, 0xFF
        ldi     R17, -0x01
;       Если полученное произведение отрицательное установится флаг C регистра SREG
        mulsu   R16, R17                ; R1:R0 = R16*R17 = -0x00FF

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Логические операции
;"НЕ"   Rd = инвертированный Rd
        ldi     R16, 0xFE
  com   R16     ;R16 = 0x01

;"И"    Rd = Rd AND Rr
        ldi     R16, 0xFE
        ldi     R17, 0x0F
        and     R16, R17        ;R16 = 0x0E

;"ИЛИ" Rd = Rd OR Rr
        ldi     R16, 0xFE
        ldi     R17, 0x0F
        or      R16, R17        ;R16 = 0xFF
```

```
;Исключающее ИЛИ   Rd = Rd XOR Rr
;Обмен содержимого двух регистров
        ldi     R16, 0xFE
        ldi     R17, 0x0F
        eor  R16,R17            ;R16 = R16 XOR R17
        eor  R17,R16            ;R17 = R17 XOR (R16 XOR R17) = R16
        eor  R16,R17            ;R16 = (R16 XOR R17) XOR R16 = R17
```

Команды пересылки данных

Эти команды выполняют копирование содержимого источника в приёмник (при этом содержимое источника не изменяется). Эти инструкции не влияют на флаги регистра состояния SREG.

```
; Программа для изучения команд передачи данных
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Работа с РОН
;-----------------------------------------------------------
;Пепедача данных из регистра в регистр Rd = Rr
        ldi     R16, 0x00
        ldi     R17, 0xFF
        mov     R16, R17
;-----------------------------------------------------------
;Передача данных между регистровыми парами Rd1:Rd0 = Rr1:Rr0
;Пример: возведение в квадрат R16 с перемещением результата в R17:R16
        ldi     R16, 0x10
        mov   R17,R16  ; копирование R16 в R17
        mul   R17,R16  ; возведение в квадрат
;Команда movw перемещает данные между парами регистров
;Примечание: в качестве операндов указываются младшие регистры в парах
        movw  R16,R0   ; перемещение результата из R1:R0 в R17:R16

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Работа с ОЗУ (SRAM)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Косвенная адресация (адресация с помощью регистровых пар X,Y,Z)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Присваивание удобных имен регистровым парам
.def    XL      =R26
.def    XH      =R27
.def    YL      =R28
.def    YH      =R29
.def    ZL      =R30
.def    ZH      =R31

;Определение границ пространства ОЗУ с которым будет идти работа
.equ    adr_start = 0x60

;-----------------------------------------------------------
;обычная запись
        ldi     XL, low (adr_start)     ;занесение адреса ОЗУ
        ldi     XH, high(adr_start)     ;в регистровую пару X
        ldi     R16, 0xFF                           ;Подготовка данных для передачи
;Пересылка содержимого R16 в ячейку памяти ОЗУ по адресу adr_start = 0x60
        st      X, R16
```

```
;запись с постинкрементом адреса
        ldi     R16, 0x00                       ;Подготовка данных для передачи
;Пересылка содержимого R16 в ячейку памяти ОЗУ по адресу adr_start = 0x60
;с последующим автоинкрементированием содержимого X(приращение адреса)
;Примечание: отличие двух операций в символе "+"
;Примечание: отличие двух операций в символе "+"
        st      X+, R16
        subi    R16, -0x01      ;изменение содержимого данных (прибавляем 1)
        st      X+, R16
        subi    R16, -0x01
        st      X+, R16

;запись с преддекрементом
;Пересылка содержимого R16 в ячейку памяти ОЗУ по адресу adr_start+2 = 0x62
;с предварительным декрементированием содержимого X
        subi    R16, -0x01
        st      -X, R16
        subi    R16, -0x01
        st      -X, R16
        subi    R16, -0x01
        st      -X, R16 ; в итоге в ячейки памяти 0x60,0x61,0x62 запишутся 5,4,3

;запись со смещением
;Пересылка содержимого R16 в ячейку памяти ОЗУ по адресу Y+q
;Примечание: данная операция работает только с рег. парами Y и Z
        ldi     YL, low (adr_start)     ;занесение адреса ОЗУ
        ldi     YH, high(adr_start)     ;в регистровую пару Y
        subi    R16, -0x01
        std     Y+5, R16
        subi    R16, -0x01
        std     Y+6, R16
        subi    R16, -0x01
        std     Y+7, R16        ; в итоге в ячейки памяти 0x65,0x66,0x67 запишутся 6,7,8
;-------------------------------------------------------------
;чтение из ОЗУ
;Примечание: Команды чтения из ОЗУ аналогичны командам записи
;обычное чтение
;Пересылка содержимого ячейки ОЗУ по адресу X=0x60 в R16
        ld      R16, X
;чтение с постинкрементом адреса
;Пересылка содержимого ячейки ОЗУ по адресу X=0x60 в R16
;с последующим автоинкрементированием содержимого X(приращение адреса)
        ld      R16, X+
        ld      R17, X+
;чтение с преддекрементом адреса
;Пересылка содержимого ячейки ОЗУ по адресу X в R16
;с предварительным декрементированием содержимого X
        ld      R16, -X
        ld      R17, -X
;чтение со смещением адреса
;Пересылка содержимого ячейки ОЗУ по адресу Y+q в R16
;Примечание: данная операция работает только с рег. парами Y и Z
        ldd     R16, Y+5
        ldd     R17, Y+6
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Прямая адресация (в команде указывается абсолютный адрес)
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Запись
        ldi     R16, 0xFF
        sts     0x70, R16       ;Занести по адресу 0x70 содержимое регистра R16
;Чтение
        clr     R16             ;Очистить регистр R16
        lds     R16, 0x70       ;Считать в регистр R16 данные из ячейки 0x70

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Работа с РВВ (регистры ввода/вывода имеют адреса
                                        ;0x20:0x5F в абс-ном пространстве (область всей ОЗУ),
                                        ;0x00:0x3F в относительном пространстве (только РВВ)
;Для команд чтения/записи в РВВ требуется указывать относительные адреса

.equ    SREG  = 0x3F
;Чтение из РВВ
        ldi     R16, 0xFF
        ldi     R17, 0xF0
        add     R16,R17     ;складываем 2 числа в SREG устанавливаются флаги C,N,S
        in              R18, SREG       ;считываем данные из SREG
;Запись в РВВ
        ldi     R16, 0b10000000
        out     SREG, R16       ;устанавливаем в SREG флаг разрешения прерывания
;Примечание: подобная работа с SREG не рекомендуется (это не относится к др. РВВ)
;для установки флагов в SREG существуют отдельные команды, выполняющиеся за 1 цикл

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Работа со стеком
;Внимание: перед использованием стека (включая и вызов подпрограмм)
;         необходимо задать его указатель SPH:SPL(0x5E:0x5D -абс.адр, 0x3E:0x3D отн.адр)
;         в этот регистр записывается адрес последней ячейки RAM
.equ    RAMEND = 0x045F         ;адрес последней ячейки памяти
.equ    SPH = 0x3E              ;адрес указателя стека в пространстве РВВ (старший байт)
.equ    SPL = 0x3D              ;адрес указателя стека в пространстве РВВ (младший байт)
        ldi     R16, low(RAMEND) ;установка вершины стека
        out     SPL, R16
        ldi     R16, high(RAMEND)
        out     SPH, R16

        ldi     R16, 0x01
        ldi     R17, 0x02
        ldi     R18, 0x03
;сохранение регистров в стеке
        push    R16
        push    R17
        push    R18
;примечание: найдите, куда записываются данные из регистров
;очистка регистров
        clr     R16
        clr     R17
        clr     R18
;восстановление регистров из стека
;Примечание: при восстановлении необходимо учитывать принцип работы стека
;                               соблюдать правило первый вошел-последний вышел
```

```
        pop     R18
        pop     R17
        pop     R16
```

Команды перехода

К этой группе команд относятся команды вызовов и возвратов из подпрограмм и команды переходов.

```
; Программа для изучения команд передачи управления
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;Команды безусловного перехода
;-----------------------------------------------------------
;rjmp - относительный безусловный переход в пределах +2047…-2047 слов
                    ;в памяти программ от места где она расположена
                    ;переданное значение суммируется со счетчиком команд
                    ;достоинство - выполняется 2 такта
                    ;адресация только в пределах 8кБ (у Atmega16 - 16КБ памяти)
        rjmp  MARK1   ; перейти на метку MARK1

        ; начало подпрограммы
        FUNC1:
                nop     ;пропуск команды
                nop     ;пропуск команды
                ret     ;возврат из подпрограммы

;jmp - абсолютный переход (значение адреса копируется в счетчик команд)
                    ;достоинство - адресация по всей памяти
                    ;недостаток - выполняется за 3 такта
MARK1:
        nop
        jmp  MARK2
        nop
        nop
MARK2:
        nop
;ijmp - косвенный переход (переход по адресу, записанному в Z)
                    ;используется для организации процедур ветвления

;Вызов подпрограмм (аналог функций)
;-----------------------------------------------------------
;Отличие от команд безусловного перехода в том, что адрес следующей команды
;сохраняется в стеке. Благодаря этому возможен возврат из подпрограммы.
;Перед вызовом подпрограммы обязательно инициализировать указатель стека
;call    - абсолютный переход
;rcall   - относительный переход
;icall   - косвенный переход
.equ    RAMEND = 0x045F     ;адрес последней ячейки памяти
.equ    SPH = 0x3E          ;адрес указателя стека в пространстве РВВ (старший байт)
.equ    SPL = 0x3D          ;адрес указателя стека в пространстве РВВ (младший байт)
        ldi     R16, low(RAMEND) ;установка вершины стека
        out     SPL, R16
        ldi     R16, high(RAMEND)
        out     SPH, R16

        call    FUNC1 ;вызов подпрограммы
```

```
        nop
        jmp     MARK5
```

Команды ветвления

```
;Команды проверки и пропуска
;-------------------------------------------------------------
;Они выполняют проверку определенного условия и если оно истинно
;то следующая в тексте команда пропускается. При этом SREG не изменяется.

;Проверка на равенство регистров:
MARK3:
        ldi     R17, 0xFF
        jmp     MARK6
MARK5:
        ldi     R16, 0xFF
        ldi     R17, 0x00
MARK6:
        cpse    R16,R17         ;пропустить следующую в тексте команду, если R16=R17
        jmp     MARK3
        nop                             ;переход сюда когда R16=R17

        jmp     MARK8

;Проверка определенных битов регистра:
MARK7:
        clr     R16
        jmp     MARK9
MARK8:
        ldi     R16, 0x02
MARK9:
;sbrc Rr,b                      Пропуск команды, если бит регистра сброшен
        sbrc    R16,1           ;пропустить следующую если бит 1 R16 равен 0
        jmp     MARK7
        nop


;Команды ветвления
;-------------------------------------------------------------
;Переход по заданному адресу в зависимости от состояния флагов SREG
;используются совместно с командами сравнения
;cp Rd,Rs       -Сравнение Rd и Rs
;cpc Rd,Rs      -Сравнение с учетом бита переноса
;cpi Rd,K       -Сравнение регистра и константы

;проверка на равенство
;breq           Переход при Z=1 (нулевой результат)
;brne           Переход при Z=0 (НЕ нулевой результат)

        ldi     R16, 0x02
MARK10:
        inc     R16
        ldi     R17, 0x03
        cp      R16,R17         ;сравнить R16 и R17
        breq    MARK10          ;переход на MARK10 в случае равенства R16 и R17(если Z=1)
```

```
        ldi     R16, 0x05
MARK11:
        dec     R16
        cpi     R16, 0x03       ;сравнить R16 и константу
        brne    MARK11          ;переход на MARK10 в случае различия R16 и R17(если Z=0)

        nop

;Сравнение регистров
;brge           Переход при S=1 (больше или равно)
;brlt           Переход при S=0 (меньше)

        ldi     R16, 0x09
        ldi     R17, 0x05
MARK12:
        dec     R16
        cp              R16,R17         ;сравнить R16 и R17
        brge    MARK12          ;перейти если R16 больше или равно R17
        nop
        nop
```

1.5 Принципы исполнения команд

1.5.1 Сигналы управления

Микроконтроллер имеет встроенный тактовый генератор, вырабатывающий колебания импульсной формы с тактовой частотой F_{CLK}. Под действием кода команды и импульсов от тактового генератора формируется последовательность сигналов для управления всеми функциональными блоками микроконтроллера. Среди них: импульсы на входы синхронизации регистров, счетчиков, сигналы выбора устройств, импульсы записи/чтения памяти и др. Например, по команде записи в ячейку ОЗУ сначала подается сигнал открывания выходных тристабильных буферов устройства, от которого будет подан адрес ячейки памяти. В следующий момент времени подается сигнал открывания выходных буферов устройства, от которого будут поданы данные. В следующий момент времени подается сигнал (импульс) записи на соответствующий вход памяти.

1.5.2 Механизм исполнения команд (конвейеризация)

Опишем механизм исполнения команд:

- По системному тактовому сигналу из памяти программ в соответствии с содержимым счетчика команд (PCL,PCH) выбирается очередная команда и записывается в регистр команд по 16-ти разрядной шине;
- В это же время (в момент выбора команды из памяти программ)

происходит выполнение предыдущей выбранной команды. В регистре команд находится код команды и указание - где находятся операнды, с которыми выполняется операция. Благодаря тому, что РОНы имеют одно тактовый цикл доступа, загрузка из них операндов, исполнение команды в АЛУ (в соответствии с кодом операции) и выгрузка результата в регистр выполняется за один машинный цикл. Загрузка и выгрузка операндов выполняется по 8-ми разрядной шине данных;

- В случае если длина команды составляет 2 слова, выполнение этой команды требует дополнительного такта обработки (команда считывается из памяти за 2 такта);
- В случае команд переходов в конвейере образуется дополнительная задержка. Причина ее в следующем. По команде перехода должна считываться команда, размещенная по адресу перехода, а конвейером считывается следующая команда. Поэтому работа конвейера приостанавливается на время считывания команды, размещенной по адресу перехода. В случае команд классических условных переходов (при выполнении условия в команде производится переход, иначе выполняется следующая команда), если условие команды выполняется, то в конвейере происходит задержка на время считывания команды по адресу перехода, а при невыполнении условия задержки не происходит (следующая команда уже считана).
- Самая первая команда считывается из программной памяти без выполнения предыдущей команды.

1.6 Контрольные вопросы

1) Нарисуйте укрупненную структуру МК.
2) Каково назначение АЛУ?
3) Каково назначение РОНов?
4) Каково назначение регистр статуса?
5) Каково назначение счетчика команд?
6) Каково назначение регистра команд?
7) Какие типы памяти имеет МК?
8) Каково назначение и организация памяти программ?
9) Опишите механизм самопрограммирования МК.
10) Каково назначение регистров ввода/вывода?
11) Каково назначение и организация оперативной памяти данных?
12) Как в МК реализован программный стек?
13) Назначение энергонезависимой памяти данных?

14) Из каких частей состоит команда?
15) Опишите отличия косвенной и прямой адресации?
16) Кратко расскажите механизм исполнения команды

2 ПОРТЫ ВВОДА-ВЫВОДА И ПРЕРЫВАНИЯ ATMEGA16A

2.1 Назначение выводов ATMega16A

Физически ATMega16A представляет собой интегральную микросхему, содержащую в зависимости от типа корпуса 40 (корпус PDIP) или 44 вывода (корпус TQFP).

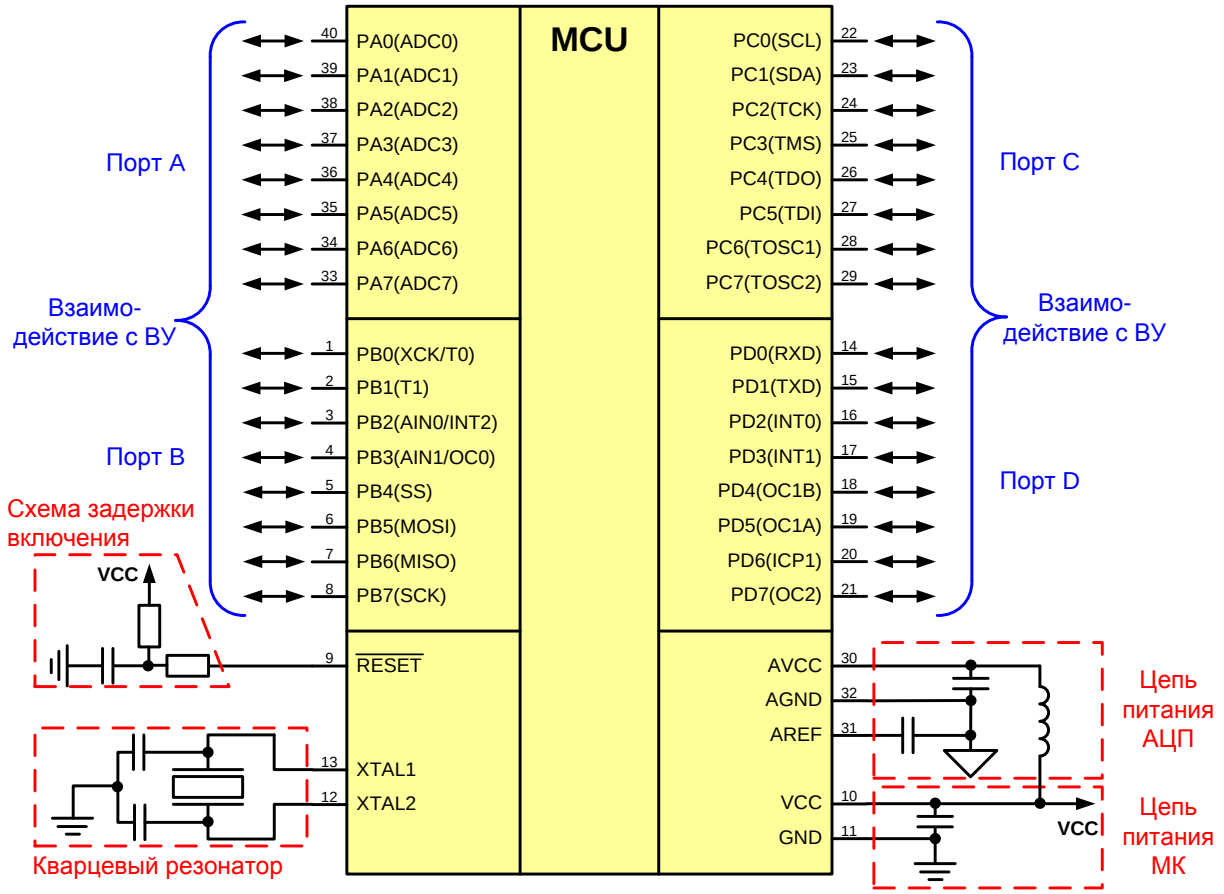


Рисунок 2.1 – УГО и типовая схема включения ATMega16A

На рисунке 2.1 приведено условно-графическое обозначение (УГО) МК в корпусе PDIP и типовая схема его включения. Все выводы («ножки») МК можно условно разделить на две группы: выводы обеспечения работы МК и порты ввода/вывода. Первые определяют саму возможность функционирования контроллера, к ним относятся:

RESET - вход сброса. Для выполнения сброса необходимо удерживать низкий уровень на входе более 50 нс. Для запуска МК требуется при включе-

нии питания подать на этот вывод положительный потенциал (или оставить этот вывод не подключенным).

XTAL1, XTAL2 - вход и выход инвертирующего усилителя генератора тактовой частоты. К выводам XTAL1, XTAL2 подключается кварцевый резонатор или внешняя RC-цепь. В случае конфигурирования МК на работу от внутреннего RC-генератора, допускается оставить эти выводы не подключенными.

VCC, GND – выводы для подключения напряжения питания микроконтроллера. Положительный вывод источника питания подключается к VCC, а отрицательный - к выводу GND. Диапазон допустимых напряжений питания составляет от 2.7 до 5,5 Вольт. Проводник, соединенный с выводами GND МК, называют общим или нулевым или «земля» и на схеме обозначают специальным символом.

AVCC и AGND – выводы для подачи напряжения питания на встроенный аналого-цифровой преобразователь (АЦП). Для минимизации помех на AVCC напряжение подается через фильтр нижних частот (лучше вообще использовать отдельный от VCC источник питания). Напряжение на AVCC должно соответствовать величине VCC. Если АЦП не используется, допускается соединить эти выводы с цифровыми питанием и землей напрямую.

AREF – вывод для подачи эталонного напряжения на блок встроенного аналого-цифровой преобразователя. Относительно значения напряжения на этом выводе выполняется измерение напряжения входного сигнала в АЦП. Если выбран внутренний источник опорного напряжения (ИОН) можно оставить этот вывод не подключенным (на схеме показан вариант с использованием внутреннего ИОН, конденсатор на землю необходим для подавления помех внутреннего опорного напряжения).

Порты ввода/вывода необходимы для взаимодействия с внешними устройствами (ВУ). В отличие от рассмотренной ранее группы контактов подключение их не является необходимым для работоспособности МК. Подробнее о портах ввода/вывода в следующем пункте.

2.2 Порты ввода/вывода

Порты ввода/вывода представляют собой регистры для хранения вводимых в микроконтроллер и выводимых из микроконтроллера данных. Это инструмент взаимодействия контроллера с внешним миром.

Микроконтроллер имеет четыре 8-разрядных порта A, B, C и D. Всего 32 линии, которые могут независимо конфигурироваться на ввод или вывод.

При выводе данных выходные буферы обеспечивают ток до 20 мА. К выводам разрядов портов подключаются внешние устройства управляющей системы (датчики и исполнительные механизмы). Все разряды портов могут использоваться как в обычном (дискретный ввод/вывод), так и в альтернативном режиме.

В альтернативном режиме порты настраиваются как входы или выходы сигналов, встроенных в микроконтроллер периферийных устройств. Альтернативная функция портов показана на УГО в круглых скобках (см. рисунок 2.1). Список всех альтернативных функций разрядов портов приведен в приложении С, здесь рассмотрим некоторые из них:

- Внешние напряжения для входов встроенного АЦП подаются через выводы порта А (ADC0-ADC7);
- Аналоговые напряжения для входов встроенного аналогового компаратора подаются через выводы разрядов PB2 и PB3 (AIN0-AIN1);
- Выводы разрядов PB2, PD2 и PD3 являются входами сигналов прерываний от внешних устройств (INT2, INT1, INT0);
- Импульсные сигналы управления (ШИМ) выдаются от встроенных таймеров через выводы PB3, PD4, PD5, PD7.
- Последовательный интерфейс SPI использует выводы PB4 – PB7, для обмена с встроенным модулем USART (универсальный синхронный - асинхронный приемопередатчик) используются линии PD0 и PD1.

2.2.1 Электрические сигналы

Перед изучением устройства портов ввода/вывода, необходимо разобраться в том, как происходит обмен с внешними устройствами на физическом уровне.

МК общается с ВУ благодаря электрическим сигналам. Электрические сигналы - это токи и вызываемые их протеканием напряжения. Для работы микросхемы в первую очередь требуется подать на неё напряжение питания. Отрицательный вывод источника питания подключается к выводу GND и его потенциал принимается за ноль вольт и относительно него измеряются все другие напряжения на ножках МК. Для сохранения работоспособности МК, эти напряжения должны быть выше -0.5Ви ниже чем напряжение питания МК, увеличенное на 0.5В.

Любой электрический сигнал является аналоговым, т.е. имеет определенный потенциал относительно GND в каждый момент времени и если он был 2В, а стал 4В, то он обязательно принимал все значения лежащие между

2-мя и 4-мя вольтами. В цифровой технике приняты некоторые правила по которым можно представить аналоговый сигнал как 1-битный цифровой сигнал или как одно из двух значений:

«1» - высокий логический уровень - логическая единица

«0» - низкий логический уровень - логический ноль

Идентификация «0» и «1» осуществляется по переходу напряжения сигнала через пороговые значения. Эти пороговые значения зависят от напряжения питания микросхемы VCC. Для ATMega16A в документации производителя приведены следующие зависимости (см. рисунки 2.2, 2.3):

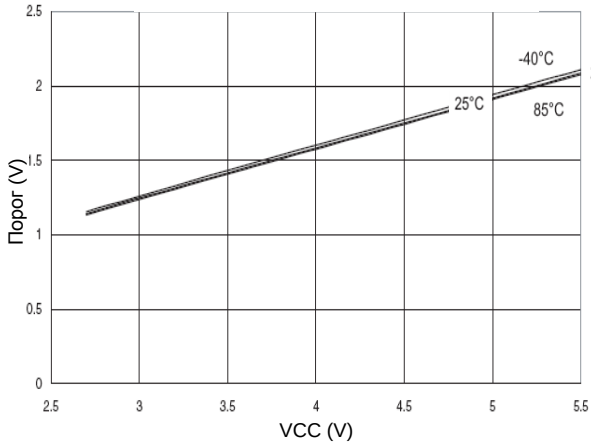


Рисунок 2.2 – Зависимость входного порогового напряжения идентификации логической «1» от VCC

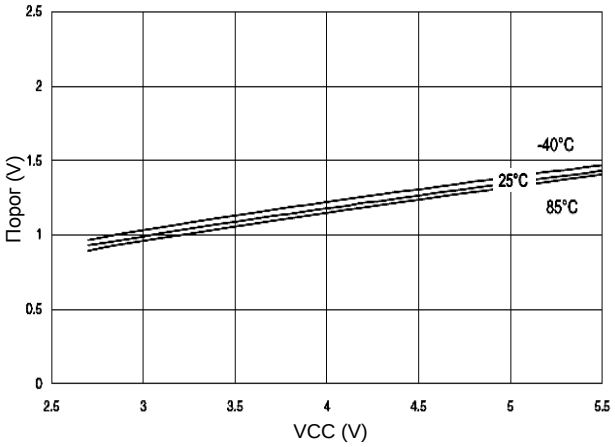


Рисунок 2.3 – Зависимость входного порогового напряжения идентификации логического «0» от VCC

Например, при напряжении питания 5В логическая «1» будет идентифицироваться при превышении входным сигналом напряжения 1.9В. А логический «0» - при уровне входного сигнала менее чем 1.3В. Любое изменение напряжения, лежащее между двумя пороговыми напряжениями от 1.3В до 1.9В, не будет влиять на логическое состояние вывода.

2.2.2 Устройство портов ввода/вывода

Каждый из 4-х портов управляется с помощью трех восьмиразрядных регистров, расположенных в пространстве памяти регистров ввода/вывода (X – название порта A, B, C или D): регистр хранения выводимых данных PORTX, регистр направления данных DDRX (ввод или вывод) и регистр значения входных данных PINX.

Упрощенная схема одного разряда n любого порта X представлена на

рисунке 2.4.

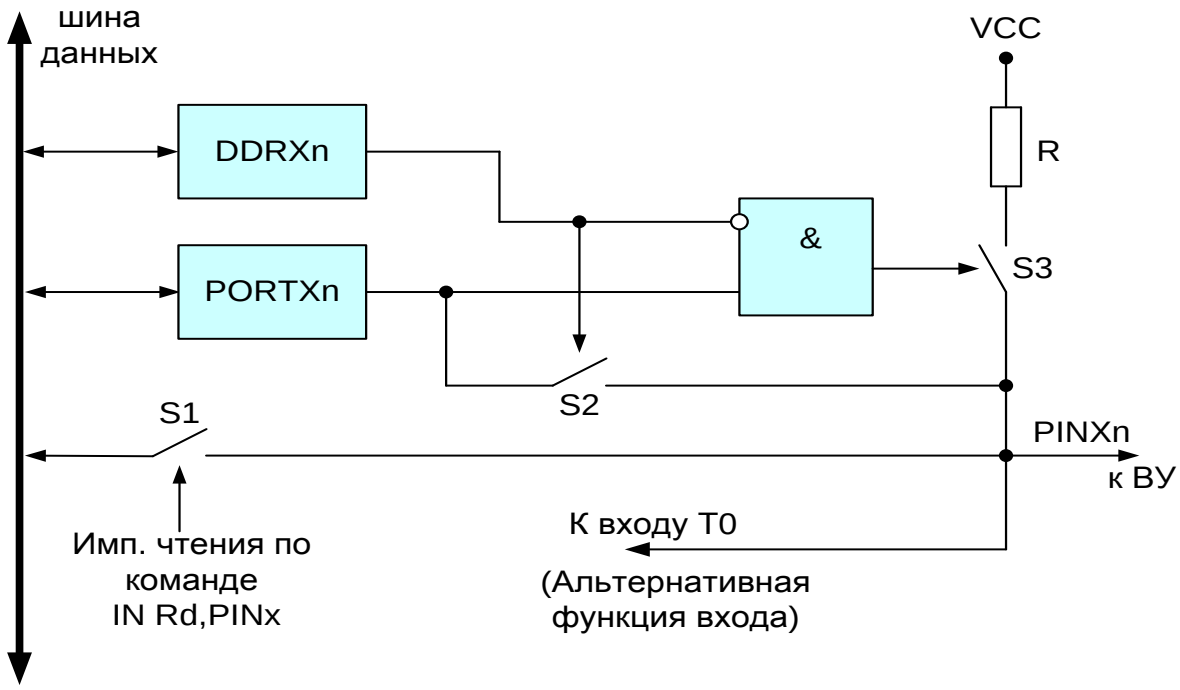


Рисунок 2.4 – Схема разряда порта ввода/вывода

Когда DDRXn=1 - выход триггера хранения выходного бита PORTXn через ключ S2 подключается к PINXn. «Ножка» настраивается на вывод данных. Что будет на этой «ножке» («0» или «1») определяется значением, записанным в PORTXn.

Для настройки разряда порта на ввод, должен быть сброшен разряд триггера DDRXn (выход PORTXn отключен от PINXn). Для считывания входных данных используется команда IN Rd, PINX, при выполнении которой формируется импульс чтения регистра PINX. Этим импульсом замыкается ключ S1, и состояние PINX считывается в регистр РОН. Таким образом, вводимые данные не сохраняются, а считываются непосредственно с контактов.

Резистор R (номинал около 40кОм) называется нагрузочным или подтягивающим. Через этот резистор осуществляется «подтяжка» напряжения на выводе к плюсу питания. «Подтяжку» можно использовать для создания четкой логической «1» на ножке МК, настроенной на ввод данных. Когда DDRXn=0 и PORTxn=1 - ключ S3 замыкается и вывод PINXn «подтягивается» к «1» через резистор. Для всех остальных комбинаций входных уровней на выходе элемента «И» будет логический «0», и резистор будет отключен.

2.2.3 Программная работа с портами ввода/вывода

Вся программная работа с портами ввода/вывода заключается в операциях с регистрами PINX, DDRX, PORTX. На рисунке 2.5 представлена информация об этих регистрах. В таблице содержатся следующие данные:

- Адрес регистра в пространстве памяти PBB (адрес регистра в общем пространстве ОЗУ приведен в скобках);
- Наименование битов регистра (от старшего бита к младшему);
- Название регистра (крайний правый столбец);
- Исходное значение, которое будет содержаться в этом регистре, после включения МК (или сброса);

Эти регистры находятся в пространстве памяти PBB, обращение к ним осуществляется по адресам:

	7	6	5	4	3	2	1	0	
0x1B(0x3B)	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	**PORTA**
0x18 (0x38)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	**PORTB**
0x15 (0x35)	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	**PORTC**
0x12 (0x32)	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	**PORTD**
Исх. код	0	0	0	0	0	0	0	0	
0x1A(0x3A)	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	**DDRA**
0x17(0x37)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	**DDRB**
0x14(0x34)	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	**DDRC**
0x11(0x31)	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	**DDRD**
Исх. код	0	0	0	0	0	0	0	0	
0x19 (0x39)	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	**PINA**
0x16 (0x36)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	**PINB**
0x13 (0x33)	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	**PINC**
0x10 (0x30)	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	**PIND**
Исх. код	Не определено								

Рисунок 2.5 – Регистры управления портами ввода/вывода

Настройки портов ввода вывода приведены в таблице 2.1.

Таблица 2.1 – Настройки портов ввода/вывода

DDRX	**PORTX**	**Направление**	**«Подтягивающий» резистор**	**Описание вывода**
0	0	Ввод	Не подключен	Третье состояние
0	1	Ввод	Подключен	Вывод подключен к VCC через резистор 40кОм
1	0	Вывод	Не подключен	Вывод подключен к GND через низ-

				коомный ключ (вх. ток до 20мА)
1	1	Вывод	Не подключен	Вывод подключен к VCC через низкоомный ключ (вых. ток до 20мА)

Работа с портами ввода/вывода осуществляется командами IN Rd,IO; OUT IO,Rr; CBI IO,b; SBI IO,b. Кроме того, при работе с портами могут использоваться команды условных переходов SBIC IO, b и SBIS IO, b. В этих командах IO – адрес порта ввода/вывода в пространстве PBB. Для удобства обращения адресам регистров можно присвоить символьное имя с помощью переменной .equ. Например, таким образом:

```
.equ    PORTA=0x1B
.equ    PINA  =0x19
.equ    DDRA  =0x1A
```

При выводе данных, порт должен быть предварительно настроен на вывод установкой разрядов регистра DDRx, а затем данные должны быть помещены в регистр PORTX по команде OUT PORTX,Rr. Данные на PINX будут переданы в следующем цикле.

Пример 1: вывести в порт А содержимое регистра R23.

```
LDI     R16,0xFF      ;байт 11111111 для настройки разрядов порта на вывод
OUT     DDRA,R16      ;настроить порт А на вывод
OUT     PORTA,R23     ;вывод в PORTA из регистра R23.
```

Пример 2: вывести в разряды 6 и 3 порта А соответственно единицу и ноль.

```
SBI     DDRA,6        ;настройка разряда 6 на вывод
SBI     DDRA,3        ;настройка разряда 3 на вывод
SBI     PORTA,6       ;установить разряд 6 порта А в 1
CBI     PORTA,3       ;сбросить разряд 3 порта  А в 0.
```

Регистр входных данных PINX обеспечивает только возможность чтения, он фактически представляет собой выводы («ножки») МК. Регистры данных PORTX и направления данных DDRX обеспечивают возможность и чтения и записи.

Пример 3: входные данные порта А поместить в R21.

```
LDI     R16,    0     ;байт 00000000 для настройки всего порта на ввод
OUT     DDRA,R16      ;настроить порт А на ввод
IN      R21,PINA      ;ввод из порта в регистр R23.
```

2.3 Структура программы

Типовой алгоритм программы контроллера представлен на рисунке 2.6

Условно структуру любой микропрограммы можно разделить на 2 части: заголовок и главный цикл.

В заголовке выполняется инициализация всех подсистем микроконтроллера, осуществляется задание режима работы микросхем на плате (если МК является ведущим устройством в системе). Здесь осуществляется инициализация подсистемы прерываний (подробнее в следующем пункте 2.4). В первую очередь необходимо правильно настроить порты ввода/вывода МК. По умолчанию они сконфигурированы на ввод (т.е. находятся в 3-м состоянии). Обычно эта часть программы выполняется один раз после включения питания МК.

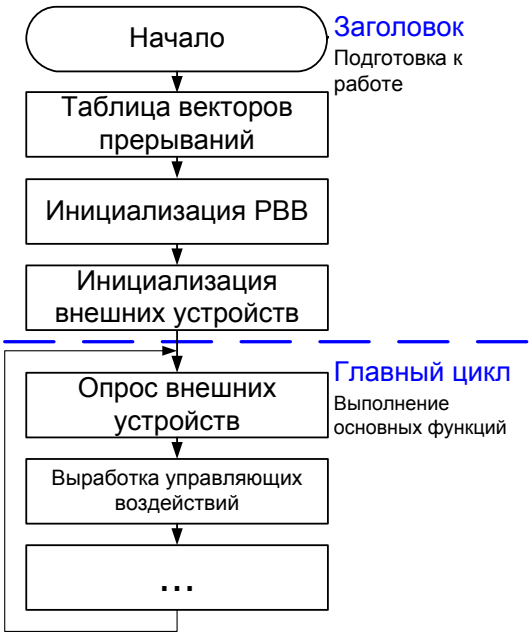


Рисунок 2.6 – Структура программы МК

Основная программа – это бесконечный цикл, по которому постоянно передвигается рабочая точка программы. В этом цикле микроконтроллер выполняет опрос состояния микросхем, кнопок, датчиков и т.д. И на основании этой информации вырабатывает управляющие воздействия в системе. Если главную функцию не «зацикливать», то программа выполнится только один раз и микроконтроллер после этого прекратит какие-либо действия. Поэтому рабочая точка программы должна находиться в постоянном движении.

2.4 Подсистема прерываний

2.4.1 Общие сведения

В реальных системах МК редко работает по линейному алгоритму.

В случае линейного исполнения программы высока вероятность пропуска короткого по времени события, или слишком медленная реакция на важное событие, требующее мгновенной обработки. Ввиду того, что контроллер в главном цикле выполняет множество действий, на опрос источника события выделяется только один малый квант времени. Событие должно произойти именно в этот промежуток времени, ведь последующий опрос будет выполнен только после следующего прохода цикла, после исполнения МК всех остальных операций. Помимо этого, необходимость постоянного опроса внешних устройств очень сильно загружает процессорную часть МК. Возни-

кают трудности с параллельным исполнением нескольких функций (например, как вырабатывать управляющие воздействия и одновременно вести счет реального времени).

Для решения этих проблем в МК реализована система прерываний.

2.4.2 Механизм обработки прерываний

Прерывания – это специальный инструмент, позволяющий остановить выполнение основной программы и переключить процессор на выполнение другой задачи по определенному внешнему или внутреннему событию. Графически процесс обработки прерывания представлен на рисунке 2.7.

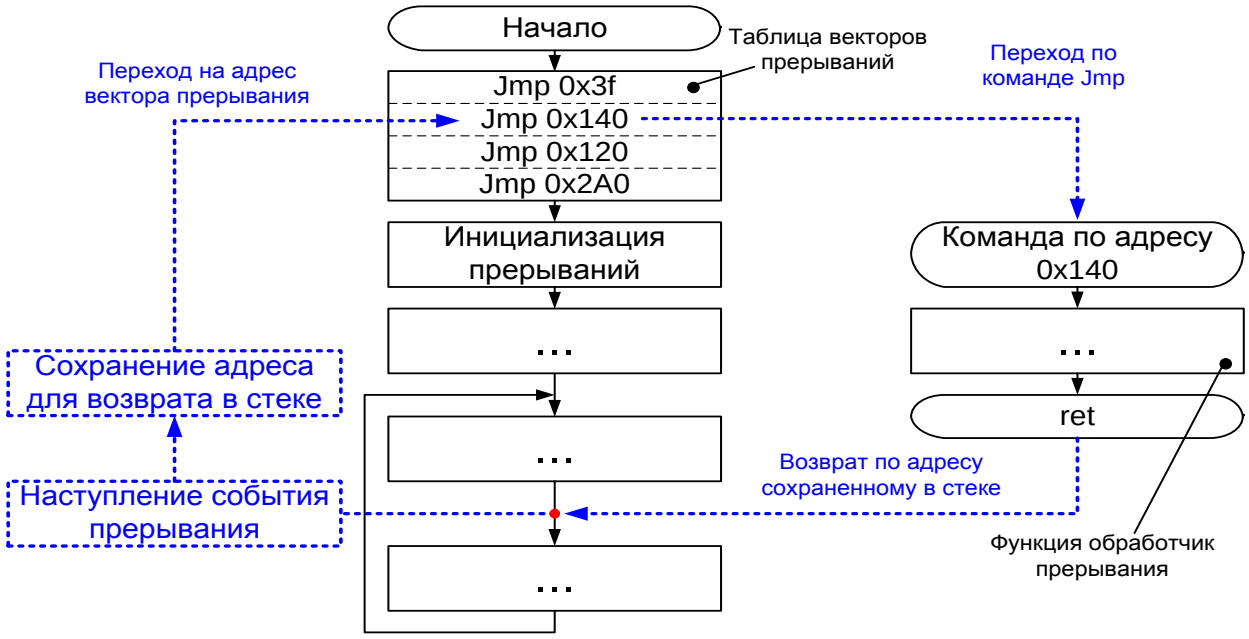


Рисунок 2.7 – Процесс обработки прерывания

Процессором микроконтроллера непрерывно производится обработка данных (выполняется текущая программа). При наступлении события прерывания процессор заканчивает выполнение текущей команды, сохраняет в стеке содержимое счетчика команд PC (адрес возврата в текущую программу после выполнения подпрограммы обработки прерывания) и загружает в счетчик команд (PC) адрес вектора прерывания. Этот адрес зарезервирован и является уникальным для каждого типа прерываний. Например, если текущее событие прерывания было вызвано сигналом на входе INT0 (вывод PD2), загружаемый в PC адрес вектора будет 0x02. По этому адресу в памяти программ была указанна команда JMP 0x140. Команда JMP приведет к новой загрузке PC адресом 0x140. Это адрес по которому расположена подпрограмма

обработки прерывания (адрес может быть любым, главное что бы там находился обработчик). Обработчик прерывания это обычная последовательность команд, которые выполняют обработку событий прерывания. Последней командой обработчика должна быть процедура RET – возврата из подпрограммы обработки прерывания, по которой сохраненное в стеке содержимое счетчика команд восстанавливается из стека, и происходит возврат в прерванную текущую программу.

Как уже упоминалось, каждому типу прерывания соответствует свой адрес вектора прерывания, в совокупности эти адреса образуют таблицу векторов прерываний (см. таблицу 2.2).

Таблица 2.2 – Таблица векторов прерываний

Адрес вектора	Источник	Условия возникновения прерывания
0x0000	RESET	Сброс по сигналу Reset, включению питания и сторожевому таймеру
0x 0002	INT0	Запрос внешнего прерывания 0
0x 0004	INT1	Запрос внешнего прерывания 1
0x 0006	TIMER2 COMP	Совпадение таймера/ счетчика 2
0x 0008	TIMER2 OVF	Переполнение таймера/счетчика 2
0x 000A	TIMER1 CAPT	Захват таймера/счетчика T1
0x 000C	TIMER1 COMPA	Совпадение A таймера/ счетчика 1
0x000E	TIMER1 COMPB	Совпадение B таймера/ счетчика 1
0x0010	TIMER1 OVF	Переполнение таймера/счетчика 1
0x0012	TIMER0 OVF0	Переполнение таймера/счетчика 0
0x0014	SPI,STC	Передача по SPI завершена
0x0016	USART, RXC	Завершение приема байта в последовательном канале USART
0x0018	USART, UDRE	Регистр данных USART пуст
0x001A	USART, TXC	Завершение передачи байта в последовательном канале USART
0x001C	ADC	Преобразование АЦП завершено
0x001E	EE_RDY	EEPROM готово
0x0020	ANA_COMP	Аналоговый компаратор
0x0022	TWI	Прерывание от модуля TWI
0x0024	INT2	Внешнее прерывание 2
0x0026	TIMER0 COMP	Совпадение таймера/счетчика T0
0x0028	SPM_RDY	Готовность SPM

2.4.3 Управление подсистемой прерываний

Для исключения конфликтов в системе прерываний для каждого источника прерывания существует свой бит индивидуального разрешения/запрета прерываний и, кроме того, имеется бит общего разрешения прерываний I,

размещенный в регистре SREG. Чтобы прерывание могло быть обслужено, необходимо: установить бит разрешения всех прерываний и бит индивидуального разрешения.

Каждому источнику прерываний соответствует также флаг прерывания, предназначенный для хранения сигнала запроса на прерывание. При поступлении сигнала запроса на прерывание устанавливается соответствующий флаг прерывания и, если прерывания от этого источника разрешены (установлены биты разрешения прерывания), производится вход в подпрограмму обработки прерывания (сохранение в стеке содержимого счетчика команд и загрузка в счетчик адреса вектора прерывания).

Когда в счетчик команд загружается вектор прерывания, соответствующий бит разрешения прерывания аппаратно очищается. Некоторые биты разрешения прерываний можно очистить программно записью логической единицы. При сбросе флага прерывания сбрасывается также бит **I** (разрешения всех прерываний) в регистре SREG, запрещая прерывания от других источников. Команда RET в конце процедуры обслуживания прерывания устанавливает бит I, разрешая возможные отложенные прерывания. Процедура прерывания может установить бит I, чтобы разрешить вложенные прерывания.

Если запрос прерывания возник, когда соответствующий бит разрешения прерывания сброшен, флаг прерывания будет сохранен в установленном состоянии, пока прерывание не будет разрешено или флаг не будет очищен программно. Если запросы прерываний возникают при сброшенном бите разрешения всех прерываний **I**, флаги прерываний будут сохранены в установленном состоянии, пока все прерывания не будут разрешены (установлен бит I в регистре SREG) и обработаны в порядке приоритетов.

Возможна такая ситуация, что в процессе работы одновременно возникнут сразу несколько запросов на прерывания (одновременно будут установлены несколько флагов прерывания). В этом случае первым будет вызван тот обработчик, чей адрес в таблице векторов прерывания находится выше. Например, при возникновении запросов от АЦП (адрес 0x001C) и компаратора (адрес 0x0020), первым будет обработан запрос от АЦП. Таким образом, каждое прерывание у AVR имеет свой собственный неизменный приоритет, который зависит от его местоположением в таблице векторов.

2.4.4 Настройка внешних прерываний INT0, INT1, INT2

Для мгновенной реакции на изменение логического состояния сигнала, МК имеет механизм внешних прерываний. На обработку прерываний по изменению логического входного сигнала могут быть настроены только выво-

ды PD2(INT0), PD3(INT1), PB2(INT2). Входы внешних прерываний INT0,INT1,INT2 могут быть программно настроены на фиксацию сигнала прерывания фронтом (восходящий импульс) или срезом (спадающим фронтом) импульса. Кроме того, входы INT0 и INT1 могут быть настроены на низкий уровень сигнала запроса прерывания. В этом случае сигнал запроса на прерывание поступает в систему прерываний, минуя флаг прерывания, т.е. не запоминается.

Биты разрешения прерываний со входов внешних прерываний INT1,INT0 и INT2 размещены в регистре GICR.

	7	6	5	4	3	2	1	0	
0x3B(0x5B)	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	**GICR**
Исх. код	0	0	0	0	0	0	0	0	

Рисунок 2.8 – Регистр разрешения внешних прерываний – GICR

GICR.7-5 - Разрешение внешних прерываний INT1, INT0, INT2. При установленных битах INT1,INT0,INT2 и установленном бите I регистра SREG разрешаются прерывания по соответствующим входам внешних прерываний. Активность сигнала по любому из этих выводов (фронт, срез или уровень 0) вызовет запрос прерывания, даже если вывод определен как выход. Запрос прерывания по логическому уровню, если он разрешен, будет существовать до тех пор, пока на входе будет низкий уровень сигнала.

Флаги внешних прерываний от входов INT1,INT0,INT2 размещены в регистре GIFR.

	7	6	5	4	3	2	1	0	
0x3A(0x5A)	INTF1	INTF0	INTF2	–	–	–	–	–	**GIFR**
Исх. код	0	0	0	0	0	0	0	0	

Рисунок 2.9 – Регистр флагов внешних прерываний – GIFR

GIFR.7-5 - Флаги внешних прерываний INTF1,INTF0,INTF2. При идентификации активного сигнала на входах INT1,INT0,INT2 соответствующий флаг прерывания INTF1, INTF0, INTF2 устанавливается в «1». Если бит I регистра SREG и соответствующий индивидуальный бит разрешения установлены, то вызывается подпрограмма, адрес которой соответствует адресу в соответствующем векторе.

Установки типа входов внешних прерываний производятся разрядами ISC01- ISC00 регистра управления MCUCR.

	7	6	5	4	3	2	1	0	
0x3A(0x5A)	–	ISC2	SE	SM	ISC11	ISC10	ISC01	ISC00	**MCUCR**

Исх. код	0	0	0	0	0	0	0	0

Рисунок 2.10 – Регистр управления процессора MCUCR

MCUCR.3 – MCUCR.0: Биты управления типом сигналов на входах прерываний INT1 и INT0. Запросы внешних прерываний на выводах INT1 – INT0 идентифицируются по значениям пар разрядов в соответствии с таблицей 2.3 Биты SC11,ISC10 - для входа INT1, а биты ISC01,ISC00 - для входа INT0. При изменении значений битов ISC прерывание должно быть запрещено очисткой бита разрешения в регистре GICR. Иначе может произойти прерывание в момент изменения значения битов.

Таблица 2.3 – Управление типом сигналов внешних прерываний INT0 и INT1

ISC11 (ISC01)	**ISC10 (ISC00)**	**Описание**
0	0	Запрос прерывания идентифицируется по низкому уровню на INT1(INT0)
0	1	Зарезервирован
1	0	Запрос прерывания идентифицируется по спадающему фронту на INT1(INT0)
1	1	Запрос прерывания идентифицируется по нарастающему фронту на INT1(INT0)

Прерывание по входу INT2 может быть сгенерировано только по фронту или по срезу сигнала. Если разряд ISC2 равен «0», то прерывание будет по срезу (перепад из «1» в «0»), а если ISC2 равен «1», то прерывание настраивается по фронту (перепад из «0» в «1»).

Прерывания от других устройств микроконтроллера (таймеры/счетчики, АЦП, компараторы и др.) управляются с помощью регистров управления, которые относятся к самим этим блокам и будут рассмотрены далее.

2.5 Контрольные вопросы

1) Назовите назначение выводов RESET, XTAL1, XTAL2, VCC, GND, AVCC, AGND, AREF.
2) Что такое порты ввода/вывода, сколько портов ввода/вывода доступно у ATMega16A?
3) Опишите работу портов ввода/вывода в обыкновенном режиме.
4) Что такое альтернативная функция вывода МК? Приведите примеры альтернативных функций портов ввода/вывода
5) Какие электрические сигналы использует МК для обмена данными с внешними устройствами?

6) Портами ввода вывода управляют регистры DDRX, PINX, PORTX назовите назначение этих регистров.
7) Опишите типовую структуру программы МК.
8) Что такое прерывания. Какие функции выполняет подсистема прерываний.
9) Кратко опишите механизм обработки прерываний.
10) Что такое таблица векторов прерываний?
11) В регистрах управления прерываниями имеются биты разрешения прерываний и флаги прерываний. Назовите их функции.
12) Для чего используются внешние прерывания INT0, INT1, INT2?

3 ТАЙМЕР/СЧЕТЧИК ATMEGA16A

3.1 Общие сведения о таймерах счетчиках

Микроконтроллер ATMega16 имеет три встроенных таймера/счетчика T0, T1 и T2. С помощью таймеров/счетчиков в управляющей или встроенной системе на микроконтроллере выполняются следующие функции: формирование временных сигналов управления объектом, измерение временных параметров сигналов от датчиков, подсчет числа внешних событий, формирование точных временных интервалов. К временным параметрам сигналов относятся: частота, период, длительность, временной сдвиг импульсов. Понятие «внешнее событие» соответствует фронту/срезу импульса, поступающего через вывод порта на вход таймера от датчика. Подсчет числа внешних событий может использоваться при подсчете числа деталей на конвейере или подсчете числа оборотов вала.

Аппаратно таймеры/счетчики представляют собой загружаемые суммирующие счетчики (в некоторых режимах ШИМ таймеры являются реверсивными). Основные режимы таймеров/счетчиков: режим счетчика и режим таймера. В режиме счетчика таймеры выполняют функции счета внешних импульсов, поступающих с внешних устройств (от датчиков) через контакты разрядов порта с альтернативными именами T0, T1, T2.

В режиме таймера таймерами/счетчиками производится счет импульсов, поступающих на их входы либо от тактового генератора напрямую, либо через делитель частоты, с коэффициентами деления 8, 64, 256, 1024, т.е. в режиме таймера производится счет временных импульсов. Этот режим используется для формирования точных временных интервалов, например, длительности импульса и длительности паузы при генерации импульсных сигналов.

Точность и разрешение 8-разрядных таймеров/счетчиков растет с уменьшением коэффициента предварительного деления. Высокий коэффициент предварительного деления удобно использовать при реализации медленных операций или синхронизации редко происходящих событий.

При переполнении T1 или T0 вызывается подпрограмма обработки прерывания по переполнению таймера/счетчика. Адреса подпрограмм (векторы) прерываний размещены в ячейках программной памяти с адресами 0x0010 и 0x0012 для T1 и T0 соответственно.

При записи значения в таймер/счётчик T0 или T1, если присутствуют тактовые импульсы, таймер счётчик продолжает счёт в следующем за опера-

цией записи тактовом цикле таймера/счетчика с загруженного значения, т.е. допускает запись «на лету».

Более подробно механизм работы таймеров/счетчиков МК рассмотрен на примере T1.

3.2 Структура таймера/счетчика T1

Структурная схема таймера/счетчика показана на рисунке 3.1.

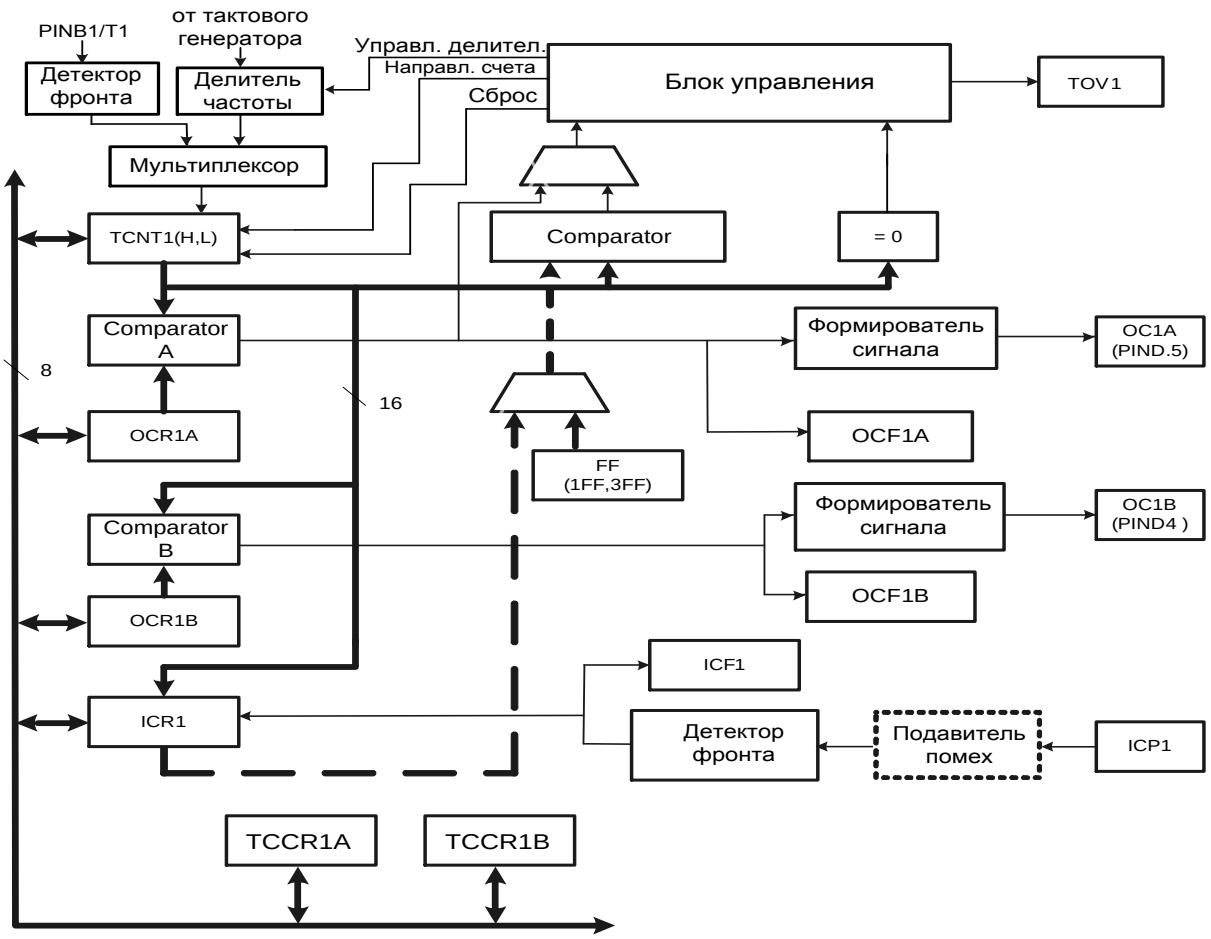


Рисунок 3.1 – Структура таймера/счетчика T1

16-разрядный таймер/счетчик T1 является наиболее функциональным по сравнению с остальными таймерами/счетчиками, доступными в ATMega16A. T1 предназначен для точного задания временных интервалов, генерации прямоугольных импульсов и измерения временных характеристик импульсных сигналов. Таймер/счетчик T1 кроме режимов счетчика и таймера, имеет режим захвата и 15 режимов сравнения. Часть режимов сравнения называют режимами генератора импульсов с широтно-импульсной модуляцией

(ШИМ).

3.3 Настройки таймера/счетчика Т1

При работе с таймером счетчиком Т1 требуется взаимодействие со следующими регистрами:

TCNT1H (0x002D (0x004D)), TCNT1L (0x002C (0x004C)) – старший и младший байты 16-ти разрядного регистра счета.

OCR1AH (0x002B (0x004B)), OCR1AL (0x002A (0x004A)) – старший и младший байты 16-ти разрядного регистра сравнения компаратора А.

OCR1BH (0x0029 (0x0049)), OCR1BL (0x0028 (0x0048)) – старший и младший байты 16-ти разрядного регистра сравнения компаратора В.

ICR1H (0x0027 (0x0047)), ICR1L (0x0026 (0x0046)) – старший и младший байты 16-ти разрядного регистра захвата.

Для доступа к 16-ти разрядным регистрам должна соблюдаться специальная процедура. Чтобы записать данные в 16-разрядный регистр, необходимо сначала записать старший байт, а затем младший. А при чтении 16-разрядного регистра, наоборот, сначала считывается младший байт, а затем старший.

Все настройки, за исключением прерываний задаются с помощью двух регистров TCCR1A и TCCR1B.

	7	6	5	4	3	2	1	0	
0x2F(0x4F)	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	**TCCR1A**
Исх. код	0	0	0	0	0	0	0	0	
	7	6	5	4	3	2	1	0	
0x2E(0x4E)	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	**TCCR1B**
Исх. код	0	0	0	0	0	0	0	0	

Рисунок 3.2 – Регистры управления таймером/счетчиком Т1

Основные настройки, которые разработчик должен определить в ходе инициализации таймера/счетчика Т1, представлены на рисунке 3.3.

Рисунок 3.3 – Настройки таймера/счетчика T1

3.3.1 Источник тактовых импульсов

Таймер-счетчик может тактироваться внутренне через предделитель (режим таймера) или внешне тактовым источником (режим счетчика).

В режиме таймера таймером/счетчиком T1 производится счет импульсов, поступающих от тактового генератора микроконтроллера через делитель частоты с программно устанавливаемым коэффициентом деления.

В режиме счетчика таймером/счетчиком T1 производится счет внешних импульсов со входа PB1(T1). Изменение состояния T1 может производиться по фронту или по срезу входного сигнала.

Если тактовый источник не задан, то таймер/счетчик находится в неак-

тивном состоянии. Выбор типа источника тактовых импульсов определяют разряды CS12-CS10 регистра TCCR1B в соответствии с таблицей 3.1.

Таблица 3.1 – Выбор источников тактовых импульсов T1

CS12	CS11	CS10	Описание
0	0	0	Таймер/счетчик 1 остановлен
0	0	1	Режим таймера (от внутреннего генератора) Коэфф. деления = 1
0	1	0	Режим таймера (от внутреннего генератора) Коэфф. деления = 8
0	1	1	Режим таймера (от внутреннего генератора) Коэфф. деления = 64
1	0	0	Режим таймера (от внутреннего генератора) Коэфф. деления = 256
1	0	1	Режим таймера (от внутреннего генератора) Коэфф. деления = 1024
1	1	0	Режим счетчика, переключение по срезу на PB1(T1)
1	1	1	Режим счетчика, переключение по фронту на PB1(T1)

3.3.2 Прерывания

Запрос на прерывание может быть сгенерирован при переполнении таймера/счетчика (переход значения TCNT1 с 0xFFFF на 0), при достижении таймером/счетчиком значения, записанного в регистре OCR1A или в OCR1B, а также при осуществлении захвата внешнего сигнала на входе PD6(ICP1). Все сигналы запросов на прерывание представлены в регистре флагов прерываний таймеров (TIFR). Все прерывания индивидуально маскируются регистром маски прерываний таймеров (TIMSK). Подробно настройка прерываний от таймера/счетчика будет описана далее.

3.3.3 Захват внешнего сигнала

Режим захвата используется для измерения временных параметров внешнего сигнала: периода, длительности и временного сдвига импульсов.

В режиме захвата по входу обеспечивается аппаратная загрузка текущего содержимого T1 в 16-разрядный регистр захвата ICR1 по фронту или по срезу входного (внешнего) импульса на входе захвата микроконтроллера PD6(ICP1). Этот же сигнал захвата может вызвать подпрограмму прерывания по захвату (устанавливается флаг ICF1). На входе захвата фронта предусмотрена схема цифровой фильтрации (подавитель шума) для снижения риска срабатывания схемы захвата от помехи. Для подавления шума выполняются четыре последовательных опроса состояния вывода и все четыре выборки должны иметь одинаковый уровень.

Условия захвата определяются битами ICES1 и ICNC1 регистра управления TCCR1B.

Бит ICES1 определяет выбор фронта срабатывания на входе захвата. При установленном бите содержимое таймера/счетчика пересылается в ре-

гистр захвата входа ICR1 по нарастающему фронту на выводе входа захвата. При сброшенном бите – по падающему фронту (срезу).

Бит ICNC1 определяет установку режима подавления шума на входе захвата. При сброшенном в состояние «0» бите ICNC1 функция подавления шума входного триггера захвата запрещена. При установленном бите – разрешена.

3.3.4 Режимы сравнения

Заданный режим сравнения определяет поведение таймера/счетчика в процессе счета. В режиме сравнения текущее содержимое работающего таймера в цифровых компараторах COMPARATOR A и COMPARATOR B сравнивается с содержимым специальных 16-разрядных регистров сравнения OCR1A и OCR1B. При равенстве значений таймера и соответствующего регистра сравнения может изменяться состояние соответствующего выхода сравнения микроконтроллера PD5(OC1A) или PD4(OC1B). Эти сигналы используются в управляющих системах на микроконтроллерах для генерации импульсных сигналов с заданными значениями временных параметров: частоты следования, периода, длительности.

Режим сравнения определяется разрядами WGM13 - WGM10 в соответствии с таблицей 3.2.

Таблица 3.2 – Режимы сравнения T1

№	WGM13	WGM12	WGM11	WGM10	Режим работы	Модуль счета
0	0	0	0	0	Нормальный	0xFFFF
1	0	0	0	1	ШИМ с точной фазой, 8-разрядный	0x00FF
2	0	0	1	0	ШИМ с точной фазой, 9-разрядный	0x01FF
3	0	0	1	1	ШИМ с точной фазой, 10-разрядный	0x03FF
4	0	1	0	0	Сброс при совпадении	OCR1A
5	0	1	0	1	ШИМ быстрый, 8-разрядн.	0x00FF
6	0	1	1	0	ШИМ быстрый, 9-разрядн.	0x01FF
7	0	1	1	1	ШИМ быстрый, 10-разрядн.	0x03FF
8	1	0	0	0	ШИМ с точной фазой и частотой	ICR1
9	1	0	0	1	ШИМ с точной фазой и частотой	OCR1A
10	1	0	1	0	ШИМ с точной фазой	ICR1
11	1	0	1	1	ШИМ с точной фазой	OCR1A
12	1	1	0	0	Сброс при совпадении	ICR1
13	1	1	0	1	Не используется	-
14	1	1	1	0	ШИМ быстрый	ICR1
15	1	1	1	1	ШИМ быстрый	OCR1A

3.3.5 Работа выходов PD5(OC1A) и PD4(OC1B)

Разряды COM1A1, COM1A0, COM1B1, COM1B0 определяют работу альтернативных функций выводов PD5(OC1A) и PD4(OC1B) при совпадении значений таймера TCNT1 со значениями в регистрах OCR1A и OCR1B.

Таблица 3.3 – Управление линиями PD5(OC1A) и PD4(OC1B)

COM1A1 (COM1B1)	**COM1A0 (COM1B1)**	**Описание**
0	0	Таймер/счетчик T1 отключен от вывода OC1A(B)
0	1	Переключение (смена состояния) выходной линии OC1A(B) Примечание: в режимах ШИМ при сброшенном бите WGM13 (режимы сравнения 1,2,3,5,6,7) T1 отключен от вывода OC1A(B)
1	0	Установка нуля на выводе OC1A(B) при совпадении в суммирующем режиме
1	1	Установка единицы на выводе OC1A(B) при совпадении в суммирующем режиме

Для использования альтернативных функций выводов PD5(OC1A) и PD4(OC1B) необходимо предварительно настроить их как выходы (должны быть установлены соответствующие биты регистра DDRD).

3.4 Настройка прерываний от таймеров/счетчиков

Прерывания от таймера позволяют разгрузить процессор МК при выполнении операций, в которых требуется организация или учет временных задержек. Таймер/счетчик, отсчитав необходимое число тактов генератора, может сгенерировать прерывание и вызвать подпрограмму, в которой будет находиться код для работы которого требовалась задержка времени.

Индивидуальные биты разрешения прерываний от внешних входов и от таймеров T0, T1 и T2 размещены в регистре TIMSK микроконтроллера.

	7	6	5	4	3	2	1	0	
0x39(0x59)	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	**TIMSK**
Исх. код	0	0	0	0	0	0	0	0	

Рисунок 3.4 – Регистр разрешения прерываний таймеров/счетчиков – TIMSK

TIMSK.7 – OCIE2: Разрешение прерывания по совпадению счетчика/таймера T2.

TIMSK.6 – TOIE2: Разрешение прерывания по переполнению счетчика/таймера 2. При установленном бите TOIE2 и установленном бите I регистра SREG разрешается прерывание при переполнении таймера 2.

TIMSK.5 – TICIE1: Разрешение прерывания по захвату счетчика/таймера T1. При TICIE1=1 и установленном бите I в регистре SREG разрешается прерывание от сигнала на входе захвата ICP.

TIMSK.4-3 - OCIE1A-OCIE1B: Разрешение прерывания по совпадению регистров OCR1A и OCR1B с таймером 1. При установленном бите OCIE1A или OCIE1B и установленном бите I регистра SREG разрешается прерывание при совпадении значения в регистре OCR1A или OCR1B со значением в рабочем регистре таймера 1.

TIMSK.2 - TOIE1: Разрешение прерывания по переполнению Таймера 1. При установленном бите TOIE1 и установленном бите I регистра SREG разрешается прерывание при переполнении таймера 1.

TIMSK. 1 – OCIE0: Разрешение прерывания по совпадению счетчика/таймера T0.

TIMSK. 0 – TOIE0: Разрешение прерывания по переполнению счетчика/таймера 0. При установленном бите TOIEO и установленном бите I регистра SREG разрешается прерывание при переполнении таймера 0.

Флаги прерываний от таймеров T0, T1 и T2 размещены в регистре TIFR микроконтроллера.

	7	6	5	4	3	2	1	0	
0x38(0x58)	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	**TIFR**
Исх. код	0	0	0	0	0	0	0	0	

Рисунок 3.5 – Регистр флагов прерываний таймеров/счетчиков – TIFR

TIFR.7 – OCF2: Флаг совпадения выхода таймера 2. Бит OCF2 устанавливается при совпадении значения в таймере 2 и содержимого регистра OCR2. Бит OCF2 аппаратно очищается при переходе на вектор прерывания. Возможна очистка бита записью во флаг логической «1».

TIFR.6 – TOV2: Флаг переполнения таймера 2. Бит TOV2 устанавливается при переполнении таймера 2. Он аппаратно очищается при переходе на вектор прерывания. Возможна очистка бита записью во флаг логической «1».

TIFR.5 – ICF1: Флаг входа захвата таймера 1. Бит ICF1 устанавливается при возникновении события (фронт/срез) на входе захвата. Он индицирует, что значение таймера 1 скопировано в регистр захвата ICR1. Бит ICF1 аппаратно очищается при переходе на вектор прерывания. Возможна очистка бита записью во флаг логической «1».

TIFR.4 - OCF1A: Флаг A совпадения выхода таймера 1. Бит OCF1A

устанавливается при совпадении значения в таймере 1 и содержимого регистра OCR1A. Бит OCF1A аппаратно очищается при переходе на вектор прерывания. Возможна очистка бита записью во флаг логической «1».

TIFR.3 - OCF1B: Флаг B совпадения выхода таймера 1. Бит OCF1B устанавливается при совпадении значения в таймере 1 и содержимого регистра OCR1B. Бит OCF1B аппаратно очищается при переходе на вектор прерывания. Возможна очистка бита записью во флаг логической «1».

TIFR.2 - TOV1: Флаг переполнения таймера 1. Бит TOV1 устанавливается при переполнении таймера 1. Он аппаратно очищается при переходе на вектор прерывания. Возможна очистка бита записью во флаг логической «1».

TIFR.1 – OCF0: Флаг совпадения выхода таймера 0. Бит OCF0 устанавливается при совпадении значения в таймере 0 и содержимого регистра OCR0. Бит OCF0 аппаратно очищается при переходе на вектор прерывания. Возможна очистка бита записью во флаг логической «1».

TIFR.0 - TOV0: Флаг переполнения таймера 0. Бит TOV0 устанавливается при переполнении таймера T0. Он аппаратно очищается при переходе на вектор прерывания. Возможна очистка бита записью во флаг логической «1».

3.5 Режимы сравнения таймера/счетчика T1

3.5.1 Нормальный режим

В этом режиме T1 работает как 16-разрядный суммирующий счетчик. При равенстве содержимого T1 и какого-либо регистра сравнения OCR1A или OCR1B устанавливается соответствующий флаг прерывания по совпадению OCF1A или OCF1B, может вызываться соответствующая подпрограмма обработки прерывания, а также может измениться значение выхода совпадения микроконтроллера PD5(OC1A) и PD4(OC1B). При переполнении T1 (когда значение TCNT1 изменяется с 0xFFFF до нуля) устанавливается флаг прерывания по переполнению TOV1 и может быть вызвана соответствующая подпрограмма прерывания. После переполнения таймера T1 начинается следующий цикл счета.

На рисунке 3.6 показана работа таймера/счетчика в нормальном режиме (вывод PD5(OC1A) настроен на смену состояния по совпадению TCNT1 и OCR1A).

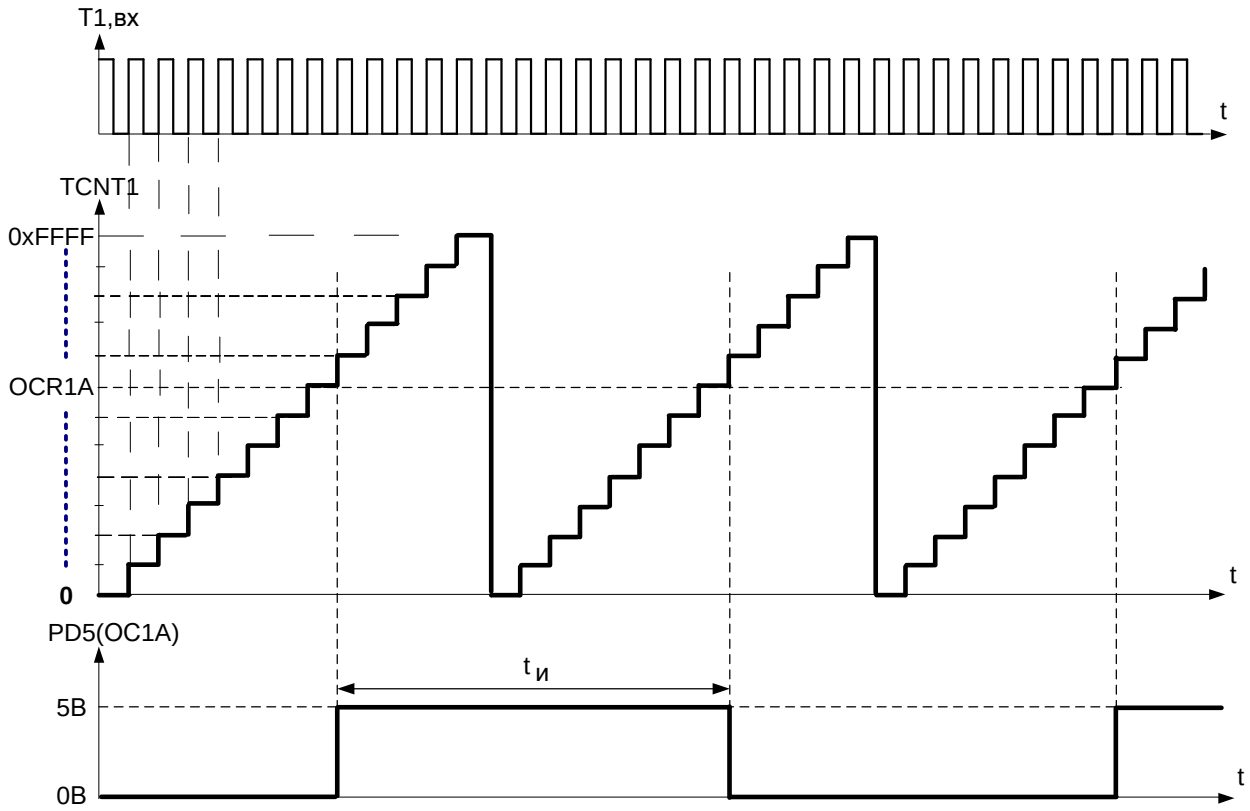


Рисунок 3.6 – Работа таймера T1 в нормальном режиме

3.5.2 Режим сброса по совпадению

В режиме №4 (сброс по совпадению с OCR1A, см. таблицу 3.2) работа T1 аналогична работе в нормальном режиме, но при совпадении содержимого TCNT1 и регистра совпадения OCR1A таймер T1 обнуляется, т.е. модуль счета таймера T1 определяется значением в регистре совпадения OCR1A. Далее в таймере T1 начинается новый цикл счета. При обнулении T1 устанавливаются флаги прерываний по совпадению OCF1A и по переполнению TOV1. Кроме того, при совпадении может изменяться состояние выхода совпадения PD5(OC1A).

В режиме №12 (сброс по совпадению с ICR1) сравнение значения T1 производится со значением в регистре захвата ICR1. При равенстве TCNT1 и ICR1 устанавливается флаг ICF1 и может измениться PD4(OC1B). В остальном, этот режим аналогичен режиму №4.

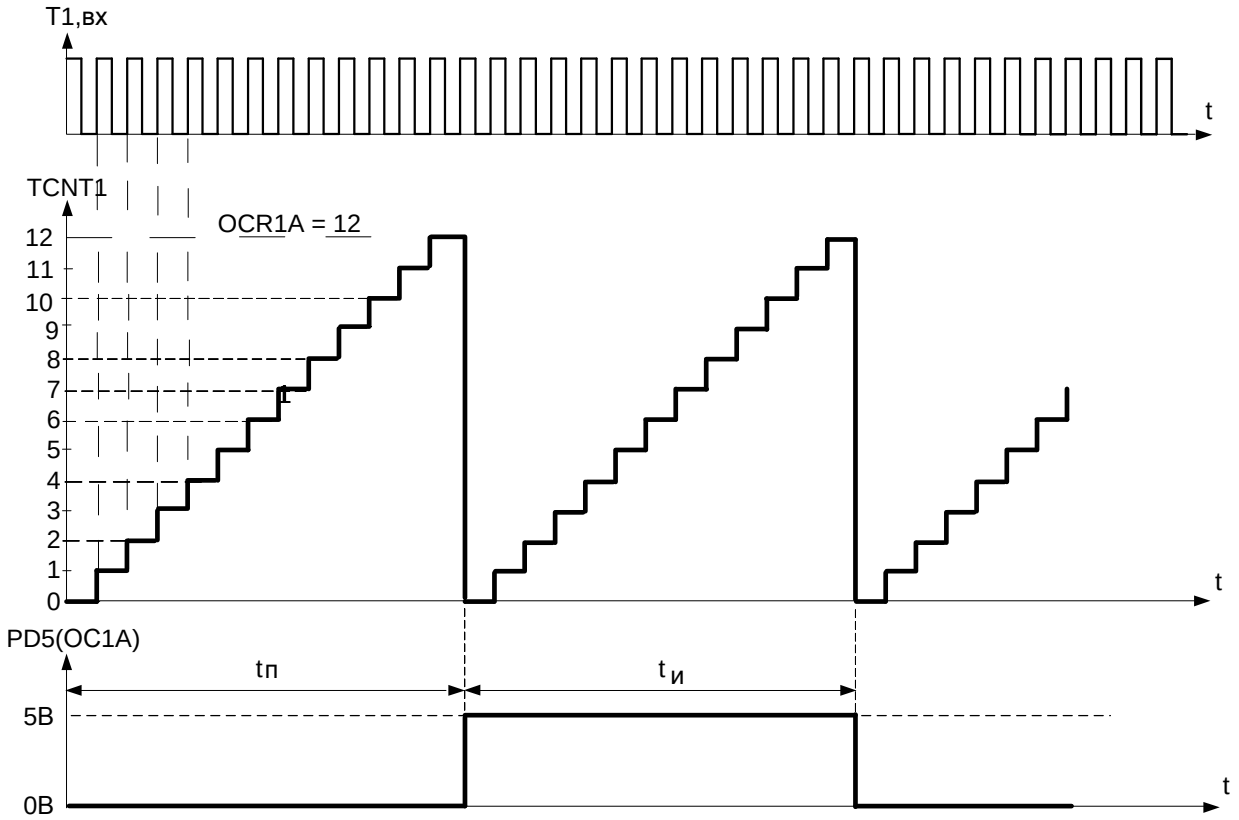


Рисунок 3.7 – Работа таймера Т1 в режиме сброса по совпадению с OCR1A

Все остальные режимы являются режимами генератора сигнала ШИМ.

3.5.3 Режимы ШИМ. Общие сведения.

Режимы генератора ШИМ используются для управления двигателями, в качестве узла специальных АЦП, для формирования аналогового сигнала и др. Режимы генератора широтно-импульсно-модулированного сигнала (ШИМ) отличаются тем, что модуль счета Т1 (максимальное значение TCNT1) может быть постоянным (0xFF, 0x1FF, 0x3FF) и программно устанавливаемым (равным содержимому регистров OCR1A или ICR1).

Во всех этих режимах при достижении таймером/счетчиком модуля счета, производится либо сброс Т1 (режимы быстрой ШИМ), либо переключение в сторону обратного счета (режим точной фазы и режим точной фазы и частоты). То есть таймер/счетчик Т1 может быть суммирующим или реверсивным.

Возможны два вида сигнала ШИМ: инвертирующий и неинвертирующий. Эта настройка определяется битами COM1A(B)1, COM1A(B)0 (см. таблицу 3.3) При инвертирующем сигнале ШИМ вывод OC1A (OC1B) устанав-

ливается в единичное состояние, при равенстве значений в суммирующем режиме и в нулевое состояние при равенстве в вычитающем режиме. При неинвертирующем сигнале ШИМ вывод OC1A (OC1B) сбрасывается в нулевое состояние при равенстве в суммирующем режиме и устанавливается в единичное состояние при равенстве значений в вычитающем режиме.

Режим быстрой ШИМ

На рисунке 3.8 представлены временные диаграммы работы таймера T1 в режиме №14 (быстрая ШИМ с модулем счета ICR1).

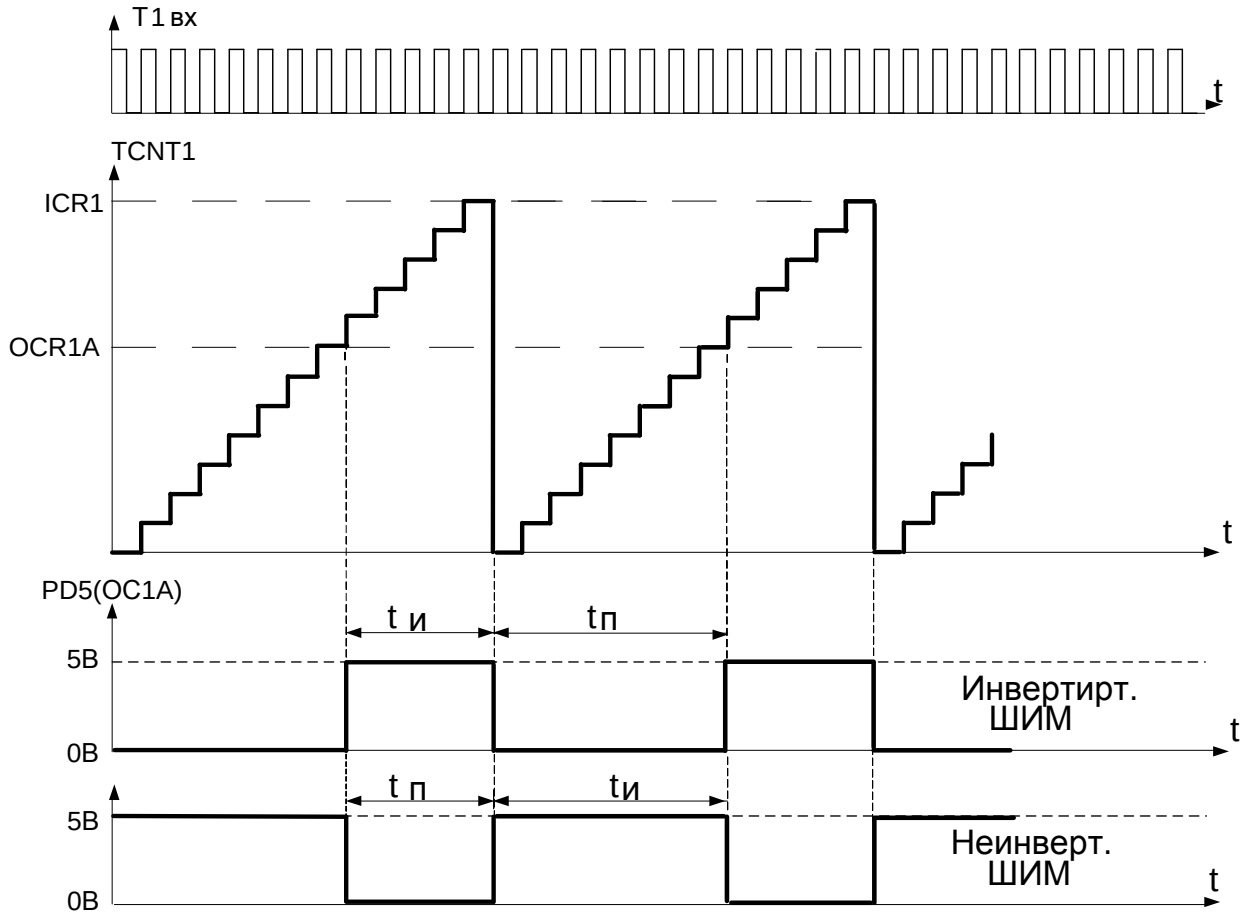


Рисунок 3.8 – Работа таймера в режиме быстрой ШИМ с модулем счета ICR1

Режим быстрой ШИМ предназначен для генерации импульсов повышенной частоты. В отличие от других режимов работы здесь используется однонаправленная работа счетчика (T1 работает как суммирующий счетчик). За счет однонаправленности счета рабочая частота для данного режима в два раза выше по сравнению с режимом ШИМ с точной фазой, где используется

двунаправленный счет. Возможность генерации высокочастотных ШИМ-сигналов делает использование данного режима полезным в задачах стабилизации питания, выпрямления и цифроаналогового преобразования. Высокая частота при этом позволяет использовать внешние элементы физически малых размеров (индуктивности, конденсаторы), тем самым снижая общую стоимость системы.

Формирование сигнала ШИМ происходит следующим образом.

Предварительно в регистры OCR1A и ICR1 загружены значения, определяющие параметры импульсов и включается таймер Т1. С приходом каждого импульса его содержимое увеличивается на единицу. Содержимое таймера сравнивается со значением в регистрах OCR1A и ICR1. Когда значение TCNT1 становится равным OCR1A на выводе PD5(OC1A) устанавливается единица для инвертируемого сигнала ШИМ или ноль для неинвертируемого ШИМ и устанавливается флаг прерывания по совпадению OCF1A. Таймер продолжает счет далее. При наступлении равенства TCNT1 и ICR1 таймер сбрасывается, на выходе PD5(OC1A) устанавливается уровень нуля (срез импульса) и устанавливается флаг прерывания по переполнению TOV1. Таймер продолжает счет от нулевого значения и т.д.

Загрузка регистров OCR1A и ICR1 из буфера производится в момент, когда значение в Т1 максимальное. В режиме №15 сравнение производится с регистром OCR1B, ШИМ - сигнал формируется на выходе OC1B, а модуль счета в регистре OCR1A.

Для расчета значений регистров используются формулы:

OCR1A = [tп / (Т такт *К)] - 1; ICR1 = [(tи +t п)/(Т такт *К)] - 1,

где:

Т такт - период тактовой частоты микроконтроллера;

К – коэффициент деления частоты входного предделителя;

Частота сигнала ШИМ определяется выражением:

F=fclk / (К* ICR1); где:

fclk – частота тактового генератора микроконтроллера;

Режим ШИМ с точной фазой

Этот режим предназначен для генерации ШИМ - сигнала с фазовой коррекцией и высокой разрешающей способностью. Режим основан на двунаправленной работе таймера-счетчика. Счетчик циклически выполняет счет в направлении от нижнего предела (0x0000) до верхнего предела, а затем об-

ратно от верхнего предела к нижнему пределу. При двунаправленной работе максимальная частота ШИМ - сигнала меньше, чем при однонаправленной работе (режимы быстрой ШИМ), однако за счет такой особенности, как симметричность в режимах ШИМ с двунаправленной работой, данные режимы предпочитают использовать при решении задач управления приводами.

Разрешающая способность ШИМ в данном режиме может быть либо фиксированной (8, 9 или 10 разрядов), либо задаваться с помощью регистра ICR1 или OCR1A. Минимальная разрешающая способность равна 2-м разрядам (ICR1 или OCR1A = 0x0003), а максимальная — 16-ти разрядам (ICR1 или OCR1A = 0xFFFF). Временная диаграмма для режима ШИМ ФК представлена на рисунке 3.9.

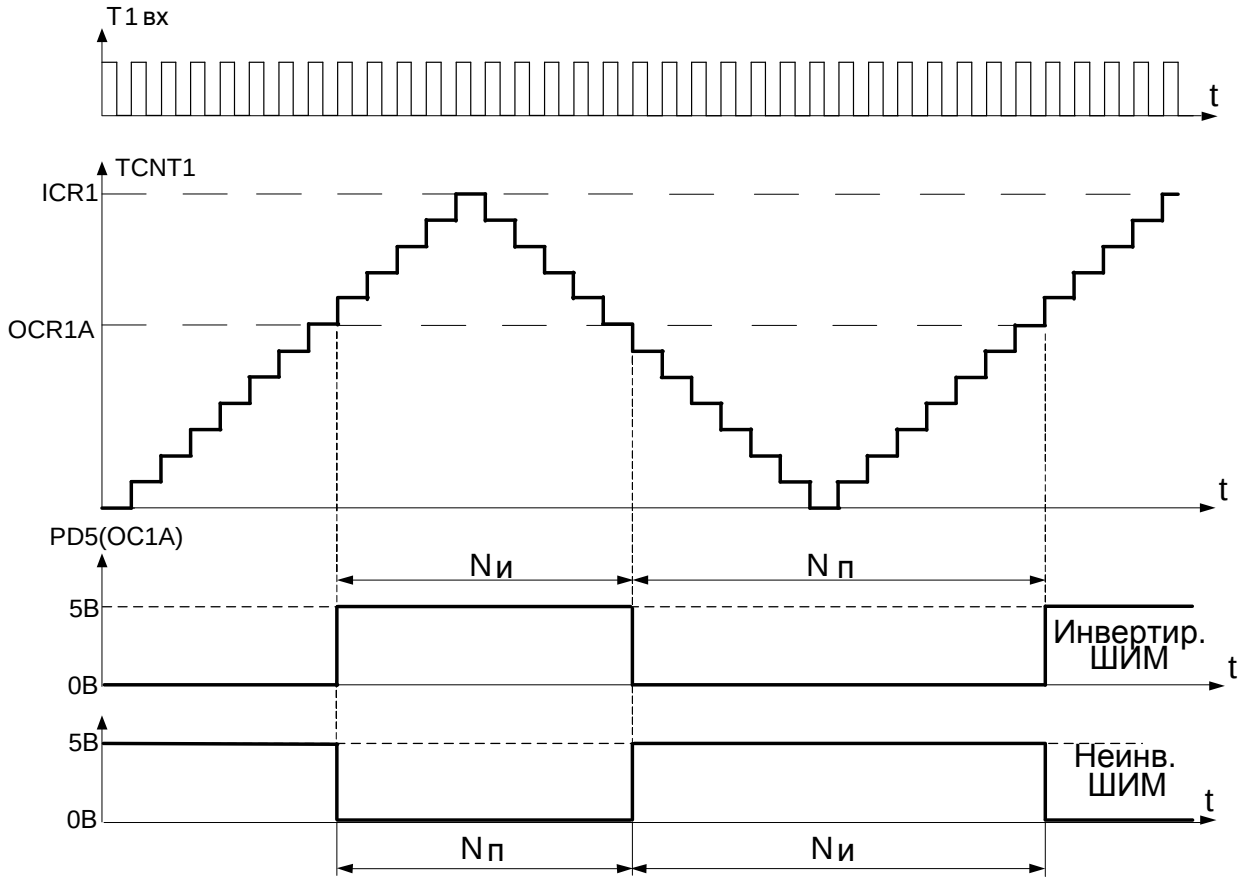


Рисунок 3.9 – Работа таймера в режиме ШИМ точной фазы со счетом по модулю с ICR1

Длительность импульсов может быть изменена изменением значения в регистре совпадения, а частота следования – изменением значения в регистре ICR1. Длительность импульса, паузы и периода в периодах импульсов на

входе Т1 определяется формулой:

Nи = 2*(ICR1 – OCR1A); Nп = 2*OCR1A ; NT = 2*ICR1.

Из этих формул можно получить значения регистров OCR1A и ICR1 по заданным значениям параметров сигнала ШИМ в периодах импульсов на входе Т1(Nи и Nп).

OCR1A = Nп /2; ICR1 = (Nи +N п)/2.

При использовании делителя частоты импульсов на входе таймера Т1, следует учесть коэффициент деления делителя К. С учетом этого расчет значений в регистрах по заданным значениям параметров импульсов в единицах времени следует выполнять по формулам:

OCR1A = tп / 2*(Т такт *К); ICR1 = (tи +t п) / 2*(Т такт *К),

где:

Т такт - период тактовой частоты микроконтроллера.

Частота сигнала ШИМ определяется выражением:

F=fclk / 2*К* ICR1; где:

fclk – частота тактового генератора микроконтроллера;

К – коэффициент деления частоты входного предделителя;

Режим ШИМ с точной фазой и частотой

Основное отличие режима ШИМ с точной фазой (ТФ) от режима с точной фазой и частотой (ТФЧ) состоит в моменте обновления регистра сравнения OCR1A из буферного регистра.

По команде загрузки любого из регистров OCR1A, OCR1B или ICR1 новым значением, оно помещается в специальный буферный регистр. В соответствующий регистр сравнения значение помещается из буфера когда Т1=0, а не в момент выполнения команды вывода в регистр. В отличие от режима ШИМ с ТФ, где обновление регистра OCR1A осуществляется сразу при команде записи, в этом режиме регистры OCR1A, OCR1B или ICR1 обновляются **на нижнем пределе счета**. В результате выходные импульсы имеют симметричную форму, а следовательно, и откорректированную частоту. Генерируемый выходной сигнал в этом случае симметричен на всех периодах. Временная диаграмма работы таймера в этом режиме соответствует рассмотренной ранее на рисунке 3.9.

В режиме № 9 работа Т1 аналогична работе в режиме №8, но сравнение значения Т1 производится со значением в регистре OCR1B, модуль счета Т1

определяется регистром OCR1A и импульсы ШИМ формируются на выводе PD4(OC1B).

3.6 Контрольные вопросы

1) Функции таймеров/счетчиков МК ATMega16A.
2) Работа таймеров/счетчиков в режиме таймера.
3) Работа таймеров/счетчиков в режиме счетчика.
4) Возможности таймера/счетчика T1.
5) Источники тактовых импульсов таймера/счетчика T1.
6) Источники прерываний таймера/счетчика T1.
7) Работа таймера/счетчика T1 в режиме захвата.
8) Режимы сравнения таймера/счетчика T1.
9) Работа таймера/счетчика T1 в нормальном режиме сравнения.
10) Отличительные особенности работы таймера/счетчика T1 в режиме сброса по совпадению.
11) Отличительные особенности работы таймера/счетчика T1 в режиме быстрый ШИМ.
12) Отличительные особенности работы таймера/счетчика T1 в режиме ШИМ с точной фазой и ШИМ с точной фазой и частотой.

4 АНАЛОГОВЫЕ УСТРОЙСТВА ATMEGA16A

4.1 Аналоговый компаратор

Аналоговый компаратор сравнивает уровни напряжения на положительном выводе PB2(AIN0/INT2) и отрицательном выводе PB3(AIN1/OC0). Если напряжение на положительном выводе больше, чем напряжение на отрицательном выводе, выход аналогового компаратора устанавливается в состояние «1». В противном случае результат сравнения равен «0». Выход компаратора сохраняется в разряде AC0 регистра управления ACSR. Выход компаратора может быть использован для управления входом захвата таймера/счетчика T1. Кроме того, компаратор может формировать свой запрос прерывания. Пользователь может задать условием формирования запроса на прерывание наличие на выходе компаратора нарастающего или спадающего фронта, а также переключение выхода.

Структурная схема компаратора представлена на рисунке 4.1.

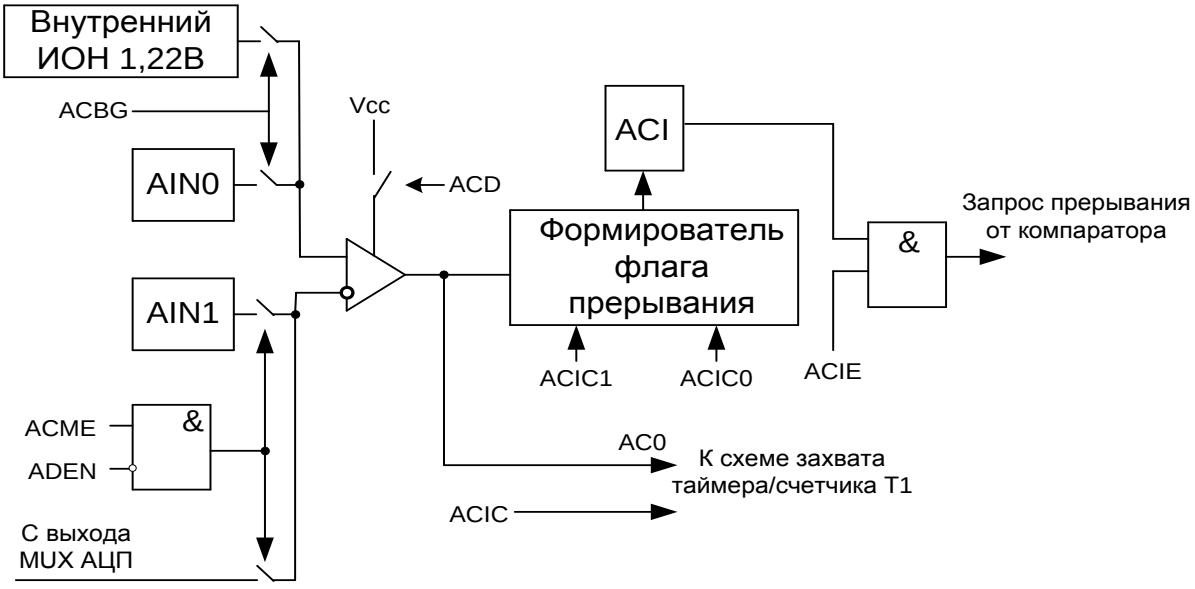


Рисунок 4.1 – Структурная схема аналогового компаратора

Для того чтобы разряды PB2(AIN0/INT2) и PB3(AIN1/OC0) могли использоваться как входы компаратора, необходимо их настроить на ввод (сбросить соответствующие разряды регистра DDRB) и отключить внутренний подтягивающий резистор (установить в «1» соответствующий разряд регистра PORTB).

К положительному входу аналогового компаратора может быть подклю-

чен выход встроенного источника опорного напряжения 1.22 В, вместо напряжения с контакта AIN0. К отрицательному входу аналогового компаратора может быть подключено напряжение с любого входа АЦП, вместо напряжения с контакта AIN1.

При переключении выхода аналогового компаратора может быть произведен захват таймера/счетчика T1, т.е. зафиксировано время изменения аналогового сигнала относительно некоторой величины (1.22 В или установленной на втором входе компаратора).

4.1.1 Регистры управления аналоговым компаратором

Управление аналоговым компаратором производится с помощью регистра ACSR.

	7	6	5	4	3	2	1	0	
0x08(0x28)	ACD	ACBG	AC0	ACI	ACIE	ACIC	ACIS1	ACIS0	**ACSR**
Исх. код	0	0	н/о	0	0	0	0	0	

Рисунок 4.2 – Регистр состояния и управления аналоговым компаратором

ACSR. 7 - ACD: запрет аналогового компаратора. При установленном бите ACD аналоговый компаратор выключен. Отключение аналогового компаратора позволяет снизить потребление. При изменении состояния бита ACD необходимо запрещать прерывание от аналогового компаратора очисткой бита ACIE в регистре ACSR. В противном случае при изменении состояния бита ACD может произойти прерывание.

ACSR. 6 - ACBG: подключение ко входу компаратора источника 1.22В.

ACSR.5 - ACO: выход аналогового компаратора. Значение бита ACO соответствует значению на выходе компаратора.

ACSR. 4 - ACI: флаг прерывания аналогового компаратора. Бит устанавливается в случае запроса компаратором прерывания, тип которого определяется битами ACIS1 и AC1SO. Подпрограмма обработки прерывания от аналогового компаратора будет выполняться при установленном бите ACIE и установленном бите I в регистре SREG. Бит ACI очищается аппаратно при переходе по соответствующему вектору прерывания. Его можно очистить также записью во флаг логической 1. При модификации других битов регистра ACSR командами SBI или CBI бит ACI будет очищен, если он был перед этими установлен.

ACSR.3 - ACIE: разрешение прерывания аналогового компаратора. При установленном бите ACIE и установленном бите I в регистре SREG разрешается прерывание по аналоговому компаратору. При сброшенном бите ACIE

прерывание запрещено.

ACSR.2 - ACIC: разрешение захвата по сигналу аналогового компаратора. Установленный бит ACIC разрешает срабатывание функции захвата Таймера/счетчика T1 по переключению аналогового компаратора. В этом случае выход аналогового компаратора подсоединяется к входной цепи логики захвата, что обеспечивает подавления шума и выбор типа прерывания. При очищенном бите ACIC соединения нет. Для запуска прерывания по захвату Таймера/счетчика T1 бит разрешения захвата TICIE1 в регистре TIMSK должен быть установлен.

ACSR.1,0 - ACIS1, ACIS0: выбор типа прерывания аналогового компаратора. Эти биты определяют тип события компаратора, при котором возникает запрос прерывания. Варианты установок показаны в таблице 4.1.

При изменении состояния битов ACIS1/ACIS0 прерывание аналогового компаратора должно быть запрещено. В противном случае при изменении состояния битов может произойти прерывание.

Таблица 4.1 – Управление типом прерываний аналогового компаратора

ACIS1	ACIS0	События, вызывающие прерывание
0	0	Переключение выхода компаратора
0	1	Зарезервирован
1	0	Спадающий фронт на выходе компаратора
1	1	Нарастающий фронт на выходе компаратора

4.2 Аналогово-цифровой преобразователь

Микроконтроллер оснащен 10-разрядным (10 – бит в результате) АЦП последовательного приближения. Блок АЦП включает: 8-канальный аналоговый мультиплексор, позволяющий использовать любой вывод порта A в качестве входа АЦП во время преобразования. Точность преобразования составляет ±2 младших разряда. Время преобразования 70...280 микросекунд. Быстродействие 15000 выборок в секунду.

4.2.1 Принцип преобразования

АЦП использует метод последовательных приближений (см. рисунок 4.3). В состав АЦП входят основные блоки: регистр последовательных приближений, 10-разрядный цифро-аналоговый преобразователь (ЦАП), аналоговый компаратор и устройство выборки и хранения (УВХ).

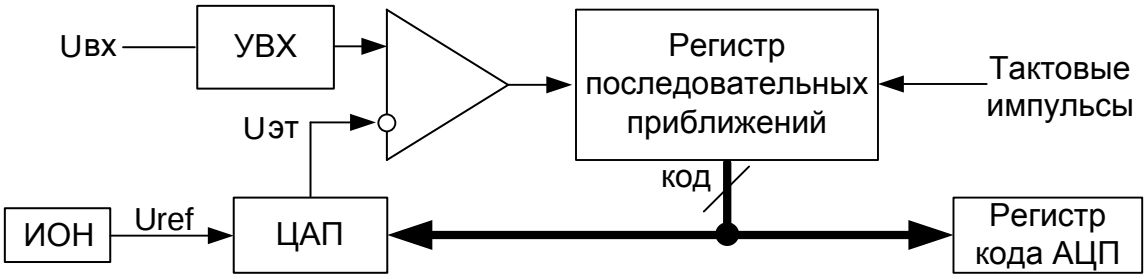


Рисунок 4.3 – Структура АЦП последовательного приближения

В последовательном АЦП входное напряжение последовательно сравнивается одним единственным компаратором с несколькими эталонными уровнями напряжения, генерируемыми ЦАПом, и в зависимости от результатов этого сравнения формируется выходной код.

Входное напряжение подается на вход компаратора, на другой вход которого подается эталонное напряжение, ступенчато изменяющееся во времени. Выходной сигнал компаратора подается на вход регистра последовательных приближений, тактируемый внешним тактовым сигналом. Выходной код регистра последовательных приближений поступает на ЦАП, который из опорного напряжения формирует меняющееся эталонное напряжение.

Регистр последовательных приближений работает так, что в зависимости от результата предыдущего сравнения выбирается следующий уровень эталонного напряжения по следующему алгоритму:

- В первом такте входной сигнал сравнивается в компараторе с половиной опорного напряжения.
- Если входной сигнал меньше половины опорного напряжения, то на следующем такте он сравнивается с четвертью опорного напряжения (то есть половина опорного напряжения уменьшается на четверть). Одновременно в регистр последовательных приближений записывается старший разряд выходного кода, равный нулю.
- Если же входной сигнал больше половины опорного напряжения, то на втором такте он сравнивается с 3/4 опорного напряжения (то есть половина увеличивается на четверть). Одновременно в регистр последовательных приближений записывается старший разряд выходного кода, равный единице.
- Затем эта последовательность сравнений повторяется нужное число раз с уменьшением на каждом такте вдвое ступени изменения эталонного напряжения (на третьем такте — 1/8 опорного напряже-

ния, на четвертом — 1/16 и т.д.). В результате опорное напряжение в каждом такте приближается к входному напряжению. Всего преобразование занимает n тактов. В последнем такте вычисляется младший разряд.

Принцип измерения входного напряжения в 4-х разрядном АЦП последовательного приближения представлен на рисунке 4.4. Здесь измеряемое напряжение равно 0.7Uref. Результат преобразования отображается разрядами D и равен 0b1011 (11 в десятичном форматре), что соответствует 11/16Uref или 0.6875Uref.

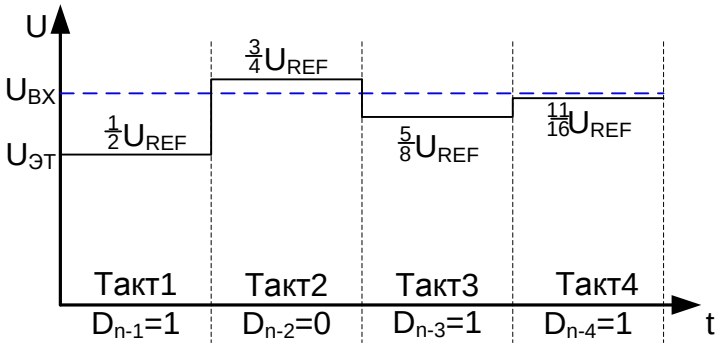


Рисунок 4.4 – Принцип измерения входного напряжения

Абсолютная погрешность преобразования равна одной единице младшего разряда кода. Очевидно, что для уменьшения погрешности преобразования необходимо увеличить разрядность АЦП.

4.2.2 Устройство выборки и хранения

В устройстве выборки и хранения (см. рисунок 4.5) значение входного аналогового сигнала запоминается и сохраняется постоянным на время преобразования. Дискретные напряжения на конденсаторе называются выборками (отсчетами) аналогового сигнала.

Когда ключ S замкнут, выходное напряжение схемы повторяет входное, т.е. Uвых=Uвх. При размыкании ключа на Cхр сохраняется последнее перед размыканием значение Uвых. Выходной повторитель на ОУ препятствует разряду конденсатора хранения Cхр на нагрузку схемы. Входное сопротивление повторителя должно быть как можно больше, поэтому обычно применяют ОУ с полевыми транзисторами на входе.

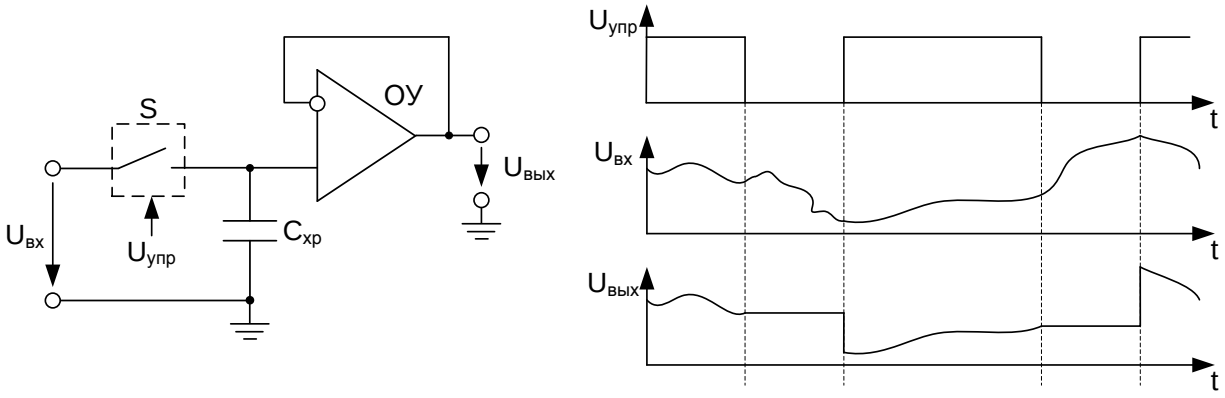


Рисунок 4.5 – Устройство выборки и хранения

4.2.3 Цифро-аналоговый преобразователь

Функция ЦАП заключается в формировании эталонных напряжений, соответствующих весовым коэффициентам входного двоичного кода: 512,256,…2,1.

Суть преобразования входного цифрового кода в выходной аналоговый сигнал довольно проста. Она состоит в суммировании нескольких токов (по числу разрядов входного кода), каждый последующий из которых вдвое больше предыдущего. Для получения этих токов используются резистивные матрицы, коммутируемые транзисторными ключами. Устройство типичного цап ЦАП с суммированием токов показано на рисунке 4.6.

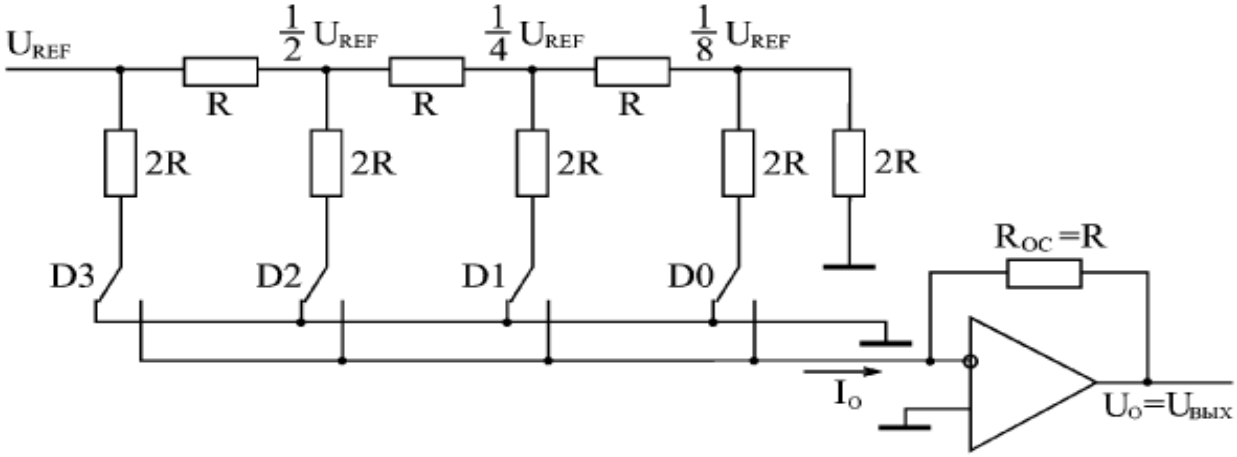


Рисунок 4.6 – ЦАП с матрицей R-2R с суммированием токов

Первым (левым по рисунку) ключом коммутируется ток величиной Uref/2R, вторым ключом — ток Uref/4R, третьим — ток Uref/8R, четвертым

— ток Uref/16R. То есть токи, коммутируемые соседними ключами, различаются вдвое, как и веса разрядов двоичного кода. Токи, коммутируемые всеми ключами, суммируются и преобразуются в выходное напряжение с помощью операционного усилителя с сопротивлением Roc = R в цепи отрицательной обратной связи. Суммарный ток Io от всех ключей создает на выходе операционного усилителя напряжение Uo=Io Roc = IoR. То есть вклад первого ключа (старшего разряда кода) в выходное напряжение составляет UREF/2, второго — UREF/4, третьего —UREF/8, четвертого —UREF/16. Таким образом, при входном коде N = 0000 выходное напряжение схемы будет нулевым, а при входном коде N = 1111 оно будет равно– 15 Uref/16.

4.2.4 Структура модуля АЦП в микроконтроллере

Схема АЦП, встроенного в ATMega16A, представлена на рисунке 4.7.

Источники входного сигнала (измеряемого напряжения)

Входные аналоговые напряжения подаются на входы порта А. К этим разрядам подключен аналоговый мультиплексор, с помощью которого производится подключение соответствующего внешнего напряжения непосредственно к АЦП. Возможна оцифровка как входных напряжений с каждого входа, которые поступают через аналоговый мультиплексор MUX(+) (измерение напряжения относительно земли), так и разности напряжений между одним из входов ADC0 – ADC2, выбранных мультиплексором MUX(-) и одним из входов ADC0 – ADC7, выбранных мультиплексором MUX(+). Во втором случае напряжения с выбранных входов поступают на входы предварительного дифференциального усилителя с программируемым коэффициентом усиления. Входное напряжение, либо разность напряжений подается на вход схемы выборки-хранения и компаратора АЦП. При изменении разности потенциалов возможно на положительный вход усилителя подавать значение не с порта А, а со встроенного источника напряжения 1.22В или с общего провода - 0В. Управление входными мультиплексорами производится разрядами MUX4 – MUX0 регистра ADMUX.

Источники опорного напряжения для ЦАПа

Питание ЦАПа производится от источника образцового напряжения (ИОН). В качестве ИОН можно выбрать встроенный источник напряжением 2,56 Вольта, либо внешний источник, подключенный к выводу AREF, либо напряжение питания АЦП (вывод AVCC микроконтроллера). Биты управления опорным напряжением ЦАПа находятся в регистре ADMUX.

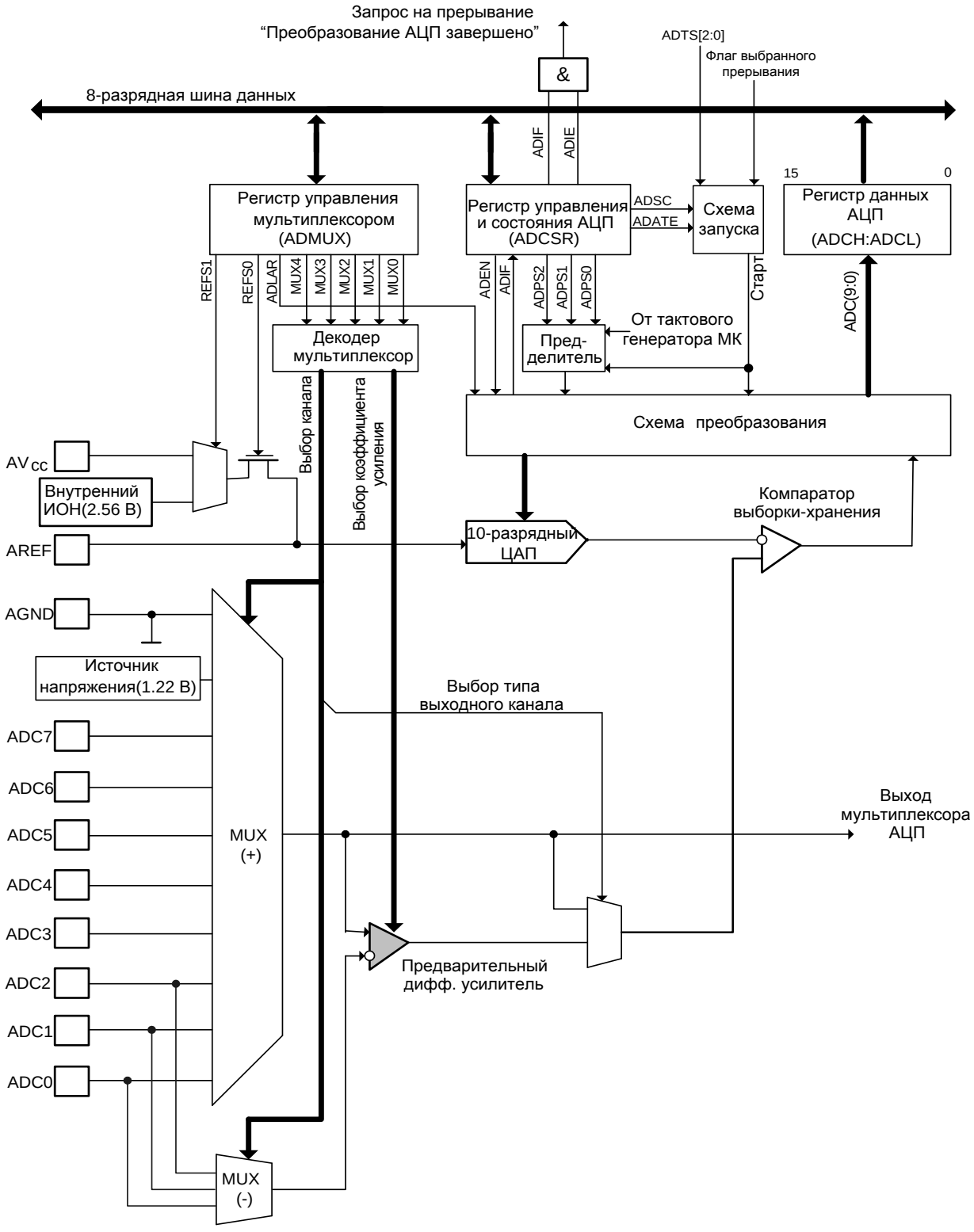


Рисунок 4.7 – Структурная схема АЦП в ATMega16A

Тактирование АЦП

Для формирования тактовой частоты АЦП используется делитель тактовой частоты микроконтроллера. Наивысшая точность преобразования достигается, если тактовая частота АЦП находится в пределах 50 – 200 кГц. Если точность преобразования меньше чем 10 разрядов достаточна, то можно использовать более высокую частоту и отбрасывать младшие разряды кода результата.

Запуск преобразования

Преобразование инициируется записью логической единицы в бит ADSC регистра ADCSR. Этот бит остается установленным все время преобразования и сбрасывается аппаратно, когда преобразование завершено.

АЦП имеет режимы: непрерывного и одиночного преобразования. Режим одиночного преобразования может быть произведен программно (установкой бита запуска АЦП), а также по прерыванию от некоторых устройств (от входа INT0, по переполнению T0 и T1 и др.). В режиме непрерывного преобразования новый цикл преобразования начинается после записи результата в регистр ADC. Биты ADTS2-ADTS0 регистра SFIOR определяют тип источника запуска преобразования.

При запуске АЦП установкой бита ADSC преобразование начинается по заднему фронту импульса синхросигнала АЦП. Один такт синхросигнала требуется на выборку-сохранение значения аналогового сигнала, после чего 13 тактов затрачивается на собственно преобразование и запись результата в регистры ADCL, ADCH. Далее перед новым преобразованием блоку АЦП необходим еще интервал в 2 такта, но если бит ADSC установлен, преобразование начнется немедленно.

Если АЦП работает в режиме однократного преобразования, то каждое преобразование должно быть инициировано пользователем. Работа АЦП разрешается установкой бита ADEN в регистре ADCSR. Первому преобразованию после разрешения АЦП предшествует пустое стартовое преобразование. Для пользователя это означает, что первое преобразование будет занимать на 13 циклов больше, чем обычно.

Сохранение результата преобразования

Выходной код АЦП помещается в регистр результата ADC, физически размещенный в двух 8-разрядных регистрах ADCH и ADCL. Возможны два варианта размещения результата. При выравнивании вправо 8 младших разрядов кода помещаются в регистр ADCL, а два старших разряда кода помещаются в два младших разряда регистра ADCH. За выравнивание результата отвечает БИТ ADLAR регистра ADMUX.

	7	6	5	4	3	2	1	0	
0x05(0x25)	-	-	-	-	-	-	ADC9	ADC8	**ADCH**
Исх. код	0	0	0	0	0	0	0	0	
0x04(0x24)	ADC7:ADC0								**ADCL**
Исх. код	0	0	0	0	0	0	0	0	

Рисунок 4.8 – – Регистр данных АЦП (ADLAR=0)

При выравнивании влево 8 старших разрядов кода помещаются в регистр ADCH, а два младших разряда кода помещаются в два старших разряда регистра ADCL. Должна соблюдаться последовательность считывания результата: сначала должно считываться содержимое регистра ADCL, а затем ADCH.

	7	6	5	4	3	2	1	0	
0x05(0x25)	ADC9:ADC2								**ADCH**
Исх. код	0	0	0	0	0	0	0	0	
0x04(0x24)	ADC1	ADC0	-	-	-	-	-	-	**ADCL**
Исх. код	0	0	0	0	0	0	0	0	

Рисунок 4.9 – – Регистр данных АЦП (ADLAR=1)

4.2.5 Регистры управления модулем АЦП

В блоке АЦП для управления используются регистры ADCSRA, ADMUX и четыре разряда регистра специальных функций SFIOR.

	7	6	5	4	3	2	1	0	
0x06(0x26)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	**ADCSRA**
Исх. код	0	0	0	0	0	0	0	0	

Рисунок 4.10 – Регистр состояния и управления АЦП

ADEN: Разрешение работы АЦП. Запись в данный бит «1» разрешает работу АЦП. Если в данный бит записать «0», то АЦП отключается, даже если он находился в процессе преобразования.

ADSC: Запуск преобразования. В режиме одиночного преобразования установка данного бита инициирует старт каждого преобразования. В режиме автоматического перезапуска установкой этого бита инициируется только первое преобразование, а все остальные выполняются автоматически.

ADATE: Если в данный бит записать «1», то АЦП перейдет в режим автоматического перезапуска. Запись «0» в этот бит прекращает работу в данном режиме.

ADIF: Флаг прерывания АЦП. Данный флаг устанавливается после за-

вершения преобразования АЦП и обновления регистров данных. Если установлены биты ADIE и I (регистр SREG), то происходит прерывание по завершении преобразования. Флаг ADIF сбрасывается аппаратно при переходе на соответствующий вектор прерывания. Альтернативно флаг ADIF сбрасывается путем записи лог. 1 в него.

ADIE: Разрешение прерывания АЦП. Установка бита в «1» разрешает прерывание по завершению преобразования АЦП.

ADPS2:0: Биты управления предделителем АЦП. Данные биты определяют на какое значение тактовая частота ЦПУ будет отличаться от частоты входной синхронизации АЦП.

Таблица 4.2 – Управление предделителем АЦП

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

Управление коммутацией входных цепей АЦП осуществляется с помощью регистра ADMUX.

	7	6	5	4	3	2	1	0	
0x07(0x27)	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	**ADMUX**
Исх. код	0	0	0	0	0	0	0	0	

Рисунок 4.11 – Регистр управления мультиплексором АЦП

REFS1:0: Биты выбора источника опорного напряжения. Данные биты определяют какое напряжение будет использоваться в качестве опорного для АЦП (см. таблицу 4.3).

Таблица 4.3 – Выбор опорного источника напряжения для АЦП

REFS1	**REFS0**	**События, вызывающие прерывание**
0	0	AREF, внутренний ИОН отключен
0	1	AVCC с внешним конденсатором на выводе AREF
1	0	Зарезервировано
1	1	Внутренний источник опорного напряжения 2.56В с внешним конденсатором на выводе AREF

ADLAR: Бит управления представлением результата преобразования. Бит ADLAR влияет на представление результата преобразования в паре регистров результата преобразования АЦП. Если ADLAR = 1, то результат пре-

образования будет иметь левосторонний формат, в противном случае - правосторонний.

MUX4:0: Биты выбора аналогового канала и коэффициента усиления. Данные биты определяют какие из имеющихся аналоговых входов подключаются к АЦП. Кроме того, с их помощью можно выбрать коэффициент усиления для дифференциальных каналов (см. таблицы).

Управление входными мультиплексорами производится разрядами MUX4 – MUX0. В таблице 4.4 указаны комбинации, при которых производится измерение напряжения между выбранным входом порта А (либо значением внутреннего источника напряжения) и AGND.

Таблица 4.4 – Выбор входного канала для АЦП (измерение относительно AGND)

MUX4:MUX0	**Однополярный вход МК**	**MUX4:MUX0**	**Однополярный вход МК**
00000	ADC0	00101	ADC5
00001	ADC1	00110	ADC6
00010	ADC2	00111	ADC7
00011	ADC3	11110	1.22 В от внутреннего источника напряжения
00100	ADC4	11111	0В

При комбинациях, указанных в таблице 4.5 производится измерение разности напряжений между входами, с соответствующим усилением разности напряжений.

Таблица 4.5 – Выбор входных каналов для измерения разности потенциалов между ними в АЦП

MUX4:MUX0	**Дифференциальный вход**		**Усиление входного напряжения**
	Положительный	**Отрицательный**	
01000	ADC0	ADC0	10
01001	ADC1	ADC0	10
01010	ADC0	ADC0	200
01011	ADC1	ADC0	200
01100	ADC2	ADC2	10
01101	ADC3	ADC2	10
01110	ADC2	ADC2	200
01111	ADC3	ADC2	200
10000	ADC0	ADC1	1
10001	ADC1	ADC1	1
10010	ADC2	ADC1	1
10011	ADC3	ADC1	1

10100	ADC4	ADC1	1
10101	ADC5	ADC1	1
10110	ADC6	ADC1	1
10111	ADC7	ADC1	1
11000	ADC0	ADC2	1
11001	ADC1	ADC2	1
11010	ADC2	ADC2	1
11011	ADC3	ADC2	1
11100	ADC4	ADC2	1
11101	ADC5	ADC2	1

Источник для запуска АЦП настраивается в регистре SFIOR

	7	6	5	4	3	2	1	0	
0x30(0x50)	ADTS2	ADTS1	ADTS0						**SFIOR**
Исх. код	0	0	0	0	0	0	0	0	

Рисунок 4.12 – Регистр специальных функций

Тип источника запуска определяется в соответствии с таблицей 4.6:

Таблица 4.6 – Источник запуска АЦП

ADTS2:ADTS0	**Источник запуска**
000	Режим непрерывного преобразования
001	Прерывание от аналогового компаратора
010	Внешнее прерывание INT0
011	Прерывание по совпадению T0
100	Прерывание по переполнению T0
101	Прерывание по совпадению в канале B таймера T1
110	Прерывание по переполнению T1
111	Прерывание по захвату T1

4.3 Контрольные вопросы

13) Принцип работы аналогового компаратора.
14) Возможности аналогового компаратора ATMega16A.
15) Назначение и характеристики АЦП ATMega16A.
16) Принцип работы АЦП последовательного приближения.
17) Принцип работы устройства выборки и хранения.
18) Принцип работы цифро-аналогового преобразователя.
19) Источники измеряемого напряжения для АЦП.
20) Источники опорного напряжения для ЦАПа.
21) Источники запуска преобразования.
22) Куда и как сохраняется результат аналогового - цифрового преоб-

разования?

5 ИНТЕРФЕЙСЫ ATMEGA16

5.1 Общие сведения об интерфейсах

Интерфейс представляет собой совокупность унифицированных аппаратных, программных, конструктивных средств, необходимых для реализации взаимодействия различных устройств между собой. Под взаимодействием понимается передача между устройствами данных. В микропроцессорной технике самый распространенный способ представления данных сигналами — двоичный (условно высокому (выше порога) уровню напряжения соответствует логическая единица), низкому — логический ноль (возможно и обратное представление).

5.1.1 Последовательные и параллельные интерфейсы

Для того чтобы передавать группу битов, используются два основных подхода к организации интерфейса:

Параллельный интерфейс — для каждого бита передаваемой группы используется своя сигнальная линия (обычно с двоичным представлением), и все биты группы передаются одновременно за один квант времени. Примеры: параллельный порт подключения принтера (LPT-порт, 8 бит), интерфейс ATA/ATAPI (16 бит), SCSI (8 или 16 бит), шина PCI (32 или 64 бита);

Последовательный интерфейс — используется лишь одна сигнальная линия, и биты группы передаются друг за другом по очереди; на каждый из них отводится свой квант времени (битовый интервал). Примеры: последовательный коммуникационный порт (COM-порт), последовательные шины USB и FireWire, PCI Express, интерфейсы локальных и глобальных сетей.

5.1.2 Классификация интерфейсов по направлению передачи

По режиму обмена информацией интерфейсы подразделяют на симплексные, полудуплексные, дуплексные, мультиплексные. В интерфейсах с симплексным режимом обмена информацией возможна лишь однонаправленная передача информации от одного абонента к другому. Соответственно, и буферы приемника и передатчика информации выполнены однонаправленными. В интерфейсах с полудуплексным режимом обмена в произвольный момент времени может производиться либо только прием, либо только передача данных между двумя абонентами; буферы приемопередатчика каждого из абонентов связи выполнены двунаправленными. В интерфейсах с дуплексным режимом обмена в любой произвольный момент времени может производиться одновременный прием и передача данных между двумя абонентами. Линии приема и передачи информации физически разделены, соот-

ветственно, контроллер обмена каждого абонента имеет два вывода (приемника и передатчика) и буферы этих выводов – однонаправленные. В интерфейсах с мультиплексным режимом обмена в каждый момент времени может осуществляться прием или передача данных между парой любых абонентов сети.

5.1.3 Режимы синхронизации

При последовательном обмене данными (бит за битом) требуется обеспечить побитную и покадровую синхронизацию. Побитная синхронизация необходима для правильного приема передаваемых битов, покадровая синхронизация – для выделения сообщения из принятой последовательности битов.

Известные в настоящее время интерфейсы периферийных устройств с последовательной передачей информации могут работать как в асинхронном, так и в синхронном режимах.

В синхронном режиме параллельно с передачей по линии данных последовательности информационных битов по линии синхросигналов передается последовательность синхроимпульсов, что позволяет, как правило, повысить скорость передачи и решить проблемы побитной синхронизации передатчика и приемника при передаче длинных информационных сообщений.

В асинхронном режиме побитная синхронизация приемника и передатчика осуществляется обычно по первому (стартовому) биту и затем поддерживается абонентами в течение времени передачи кадра стабильностью тактовых частот генераторов передатчика и приемника, частоты которых равны и, как правило, минимум в 16 раз превышают частоту передачи данных. С учетом этих обстоятельств, скорость передачи в асинхронном режиме ниже и число бит в информационной посылке (кадре) меньше.

Покадровая синхронизация в асинхронном режиме осуществляется обрамлением информации при передаче по линии стартовым и стоповым битами. Покадровая синхронизация в синхронном режиме осуществляется использованием специальных кодовых последовательностей (флагов или специальных знаков) в общем случае в начале и конце кадра. Поскольку в синхронном режиме информационные биты сообщения передаются непрерывным потоком, то для кодирования и декодирования кадров используют специальные договоренности по форматам кадров (протоколам обмена).

5.1.4 Возможности AtMega16A

В составе ATmega16A имеются следующие аппаратные модули позволяющие обмениваться данными с другими устройствами:

- Порты ввода/вывода (см. п. 2.2) – представляют собой регистры и буферы, входы и выходы которых выведены на ножки МК. Запись и чтение этих регистров осуществляется из ЦПУ МК. Это самый универсальный инструмент для обмена с внешними устройствами, однако он подходит только для простейшего дискретного ввода/вывода (опрос кнопок, управление светодиодами) , так как с усложнением протокола обмена увеличивается нагрузка на вычислительную часть системы.
- Модуль USART для обмена с удаленными устройствами, с помощью него можно реализовать распространенные интерфейсы RS-232, RS-422 и RS-485.
- Модули SPI и I2C для обмена данными с периферийными микросхемами.

5.2 USART - универсальный последовательный порт

Универсальный синхронный и асинхронный приемопередатчик предназначен для реализации последовательного интерфейса связи между микроконтроллером и удаленным микропроцессорным устройством, снабженным подобным модулем (другой МК или ЭВМ). Модуль USART обеспечивает полнодуплексный обмен данными (передача одновременно в обе стороны), синхронный и асинхронный режимы работы, контроль по четности, установку различной разрядности данных, генерацию прерываний.

Для обмена используются три вывода микроконтроллера (см. рисунок 5.1):

PD1(TXD) – выход передаваемых данных (Transmit - передавать);

PD0(RXD) – вход принимаемых данных (Receive - принимать);

PB0(XCK/T0) – вход/выход импульсов тактовой частоты в синхронном режиме.

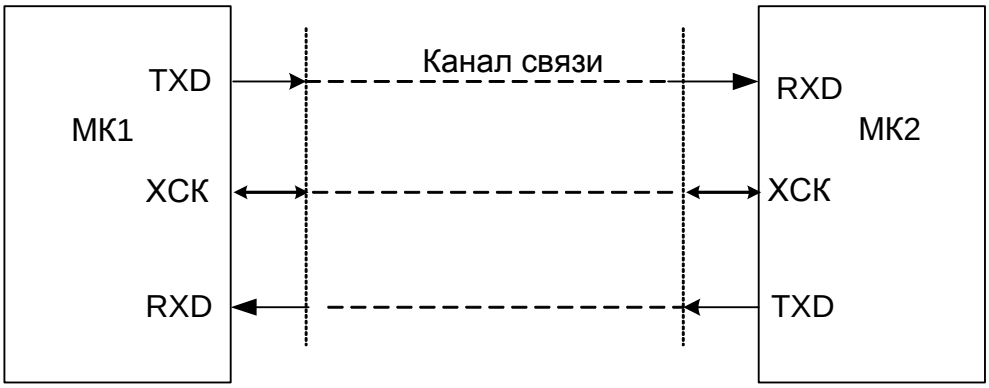


Рисунок 5.1 – Реализация обмена с помощью модуля USART

5.2.1 Формат кадра

Данные передаются последовательно специальными посылками, называемыми кадрами. Формат кадра представлен на рисунке 5.2.

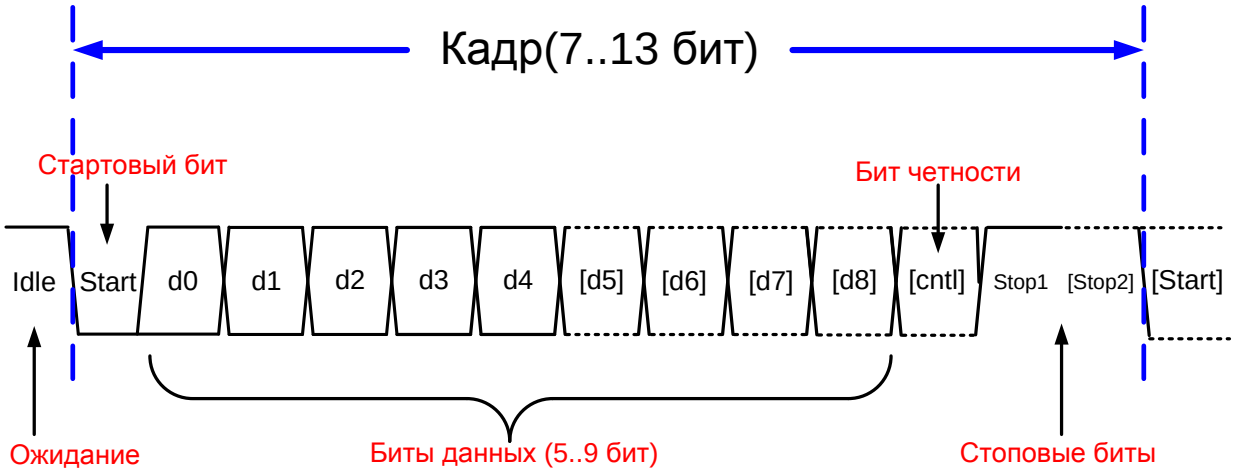


Рисунок 5.2 – Формат кадра USART

В состав кадра входят следующие биты:

Стартовый бит – этот бит всегда предшествует посылке младшего (нулевого) бита данных и сигнализирует приемнику о начале передачи. Стартовый бит всегда равен нулю.

Биты данных – располагаются последовательно от младшего к старшему. В зависимости от настроек могут передаваться 5, 6, 7, 8 или 9 бит данных.

Бит четности (паритета) – это специальный бит, который устанавливается в «1» или «0» и дополняет данные так, что бы общее число единиц было

четным (четный паритет) или нечетным (нечетный паритет). Проверка на четность (или нечетность) на приемной стороне позволяет обнаружить одиночную ошибку в принятых данных. Двойная ошибка обнаружена не будет. В зависимости от настроек контроль четности может быть отключен, в этом случае бит четности не передается.

Стоповые биты – эти биты передаются самыми последними и сигнализируют о завершении кадра. В зависимости от настроек может передаваться один, либо два стоповых бита.

В режиме ожидания (в отсутствие потока данных) передатчики устанавливают на линиях обмена логические «1».

5.2.2 Работа в асинхронном режиме

В асинхронном режиме обмена информация передается по одной линии. Дополнительные линии синхронизации не требуются. Целостность передаваемых данных обеспечивается за счет того, что кадры между передатчиком и приемником передаются с постоянной, строго определенной скоростью, а также благодаря специальному формату кадра. В асинхронном режиме работы используются только 2 контакта TXD для передачи данных и RXD для приема.

Благодаря тому, что пассивным состоянием входа и выхода USART является логическая «1», а стартовый бит всегда «0» начало кадра всегда одинаково идентифицируется. Приёмник USART ждёт перепада из 1 в 0 и отсчитывает от него временной промежуток в половину длительности бита (середина передачи стартового бита). Если в этот момент на входе всё ещё 0, то запускается процесс приёма кадра. Приёмник отсчитывает 9 битовых длительностей подряд (для 8-бит данных без проверки на четность) и в каждый момент фиксирует состояние входа. Первые 8 значений являются принятыми данными, последнее значение проверочное (стоп-бит). Значение стоп-бита всегда 1, если реально принятое значение иное, UART фиксирует ошибку.

Для формирования временных интервалов передающий и приёмный UART имеют источник точного времени (тактирования). Точность этого источника должна быть такой, чтобы сумма погрешностей (приёмника и передатчика) установки временного интервала от начала стартового импульса до середины стопового импульса не превышала половины (а лучше хотя бы четверти) битового интервала. Для 8-бит посылки 0,5/9,5 = 5 % (в реальности не более 3 %). Поскольку эта сумма ошибок приёмника и передатчика плюс возможные искажения сигнала в линии, то рекомендуемый допуск на точность тактирования UART не более 1,5 %.

5.2.3 *Работа в синхронном режиме*

В синхронном режиме работы биты передаются через вывод TXD, а через вывод XCK передаются импульсы синхронизации. Принимаются данные через вывод RXD. Импульсы синхронизации могут передаваться либо от передатчика, либо от приемника, т.е. вывод XCK является двунаправленным. В синхронном режиме каждый бит данных принимается в приемнике по фронту или срезу импульса синхронизации XCK.

5.2.4 *Структура приемопередатчика*

Структура приемопередатчика показана на рисунке 5.3. Модуль USART состоит из: блоков тактирования, передатчика и приемника.

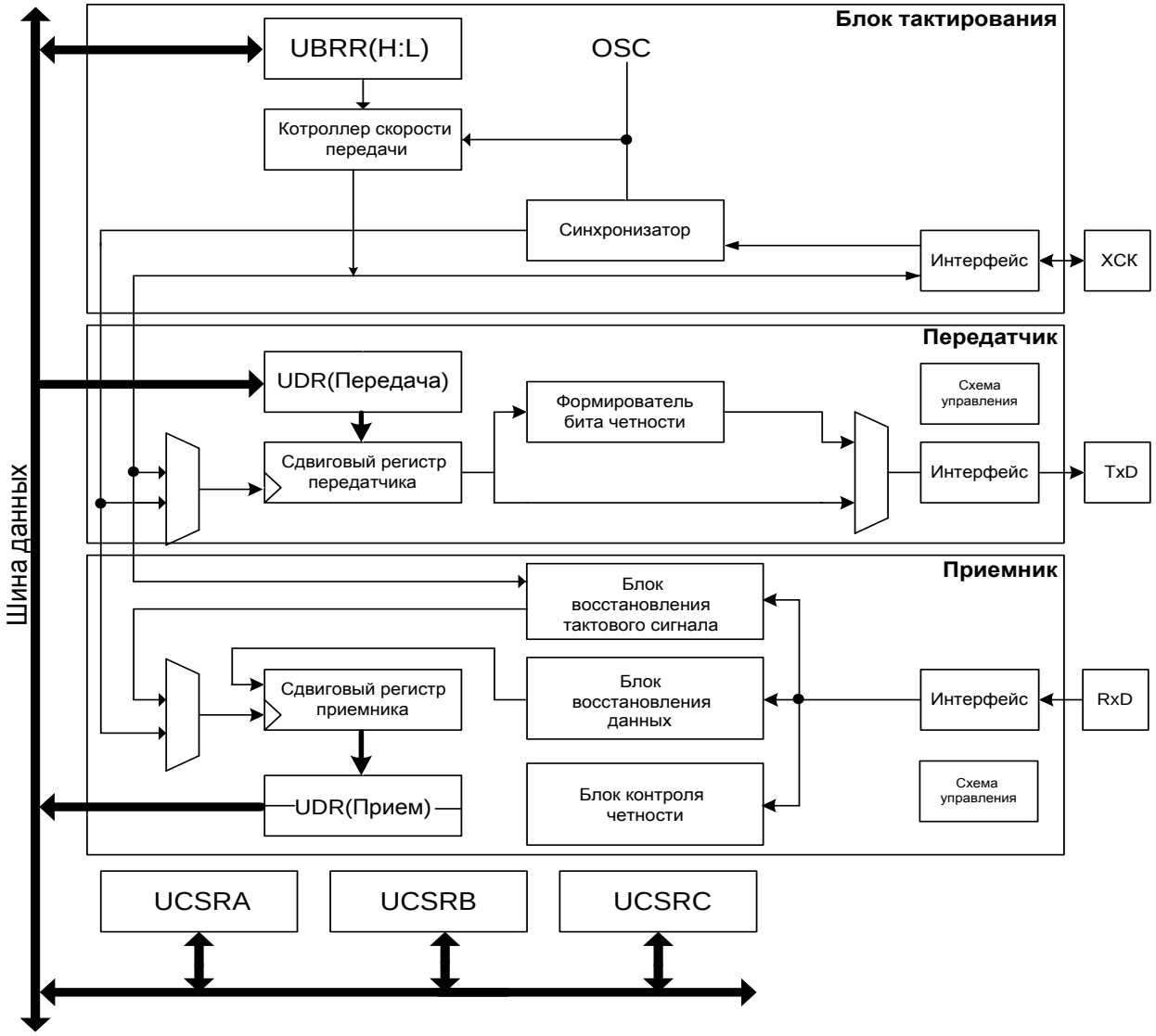


Рисунок 5.3 – Структура модуля USART ATMega16A

5.2.5 Передача данных

Регистр данных передатчика UDR предназначен для хранения данных, подлежащих передаче. Сдвиговый регистр передатчика выполняет функции преобразования параллельного формата данных в последовательный. В асинхронном режиме работы сдвиг данных в регистре производится импульсами с контроллера скорости передачи. В синхронном режиме, если микроконтроллер является ведущим, сдвиг производится импульсами с контроллера скорости. Если микроконтроллер является ведомым, то сдвиг производится импульсами со входа XCK, синхронизированными импульсами тактового генератора OSC в синхронизаторе.

Перед передачей байт данных в регистре UDR (передача) обрамляется стартовым и стоповым битами. Бит контроля по четности включается в кадр из схемы формирователя бита четности.

Если 10 (11)-разрядный сдвиговый регистр передатчика пуст, данные из регистра UDR передаются в сдвиговый регистр. При этом устанавливается бит UDRE (регистр данных пуст) регистра управления UCSRA. Это означает, что в регистр UDR может быть загружен следующий байт данных. Запись данных в регистр UDR очищает бит UDRE. Когда данные переданы из регистра UDR в сдвиговый регистр, к его коду вначале добавляется стартовый бит (состояние 0 - стартовый бит), а в конце - стоповый бит (состояние 1 - стоповый бит). Если в регистре управления UCR установлен бит CHR9 (т.е. выбран режим 9-разрядного слова данных – с битом контроля), то бит TxB8 регистра UCR пересылается в бит 9 сдвигового регистра передатчика.

Сразу после пересылки данных в сдвиговый регистр импульсом синхросигнала стартовый бит выдвигается на вывод TxD. За ним следуют биты данных, младший бит первым. После выдвижения стопового бита сдвиговый регистр загружается новыми данными, если символ был записан в регистр UDR во время передачи. В процессе загрузки бит UDRE находится в установленном состоянии. Если новые данные не будут загружены в UDR до выдачи стопового бита, флаг UDRE останется установленным. В этом случае в регистре порта UART (UCSRA) устанавливается флаг TxC завершения передачи.

Установленный бит TxEN регистра UCSRB разрешает передачу. При очищенном бите TxEN вывод TXD может быть использован в качестве вывода общего назначения. При установленном бите TxEN передатчик UART подключается к выводу TXD и использует его как выход, независимо от установки бита направления передачи DDRD.1 в регистре DDRD.

5.2.6 Прием данных

В блоке приемника биты кадра поступают из линии через блок восстановления данных на вход данных сдвигового регистра приема. В блоке восстановления производится восстановление фронтов принятых импульсов. В асинхронном режиме работы USART, сдвиг битов данных в регистре сдвига производится импульсами от контроллера скорости передачи, согласованными по фазе с битами данных в блоке восстановления. В синхронном режиме сдвиг производится импульсами от входа XCK (в случае ведомого), или от контроллера скорости (в случае ведущего). Принятое слово данных из регистра сдвига помещается в регистр данных приема UDR.

Работа приемника разрешается установкой разряда разрешения приема RXEN регистра UCSRB. При установке этого разряда вход приемника подключается к выводу RXD микроконтроллера независимо от значения разряда установки направления регистра DDRD.

Приемник проверяет состояние линии RXD с частотой в 16 раз большей, чем частота передачи, т.е. во время прихода одного бита проводится 16 выборок. Если передача не ведется (режим ожидания), то в линии установлен единичный уровень. При пассивном состоянии линии перепад «1» в «0» воспринимается как начало стартового бита и запускается последовательность его проверок. Вслед за переходом «1» «0» на выборках 8, 9 и 10 приемник вновь проверяет линию RXD на изменение логического состояния. Если две или три из этих трех выборок равны логической «1», то стартовый бит отвергается как шумовой всплеск и приемник начинает выявлять и анализировать следующие переходы из «1» в «0».

Если был обнаружен действительный стартовый бит (из выборок 8,9 и 10 две выборки из трех являются нулевыми), то принимаются следующие за стартовым битом информационные биты. Эти биты также тестируются на выборках 8, 9 и 10. Логическое состояние бита определяется по двум и более одинаковым состояниям выборок. Все биты вводятся в сдвиговый регистр приемника.

При поступлении стопового бита необходимо, чтобы не менее двух выборок из трех показали уровень 1. Если две или более выборок покажут состояние 0, то при пересылке принятого байта в регистр UDR, в регистре управления UCSRA устанавливается бит ошибки кадра FE. Пользователь перед чтением регистра UDR должен проверять состояние бита FE. Флаг FE очищается при чтении содержимого регистра данных UDR. Вне зависимости от того, принят правильный стоповый бит или нет, данные пересылаются в регистр UDR и устанавливается флаг завершения приема RXC в регистре

статуса UCSRA.

Если выбран режим обмена 9-разрядными словами данных (установлен бит CHR9 регистра UCSRB), то при пересылке данных в регистр UDR бит RXB8 регистра UCSRB загружается в 9-й бит сдвигового регистра передачи. Если после получения символа к регистру UDR не было обращения, начиная с последнего приема, в регистре UCSRA устанавливается флаг переполнения OR. Это означает, что данные, принятые в сдвиговый регистр, потеряны. Бит OR доступен тогда, кода из регистра UDR прочитан байт данных. Пользователю, для обнаружения потери данных, необходимо всегда проверять флаг OR после чтения регистра UDR.

5.2.7 Регистры управления

Запись передаваемых данных и считывание осуществляется через регистр данных UDR. В действительности регистр UDR представляет собой сдвоенный регистр данных - регистр передатчика и регистр приемника используют один и то же адрес 0x0C(0x2C). При записи в регистр обращение производится к регистру передатчика, а при чтении - к регистру приемника.

Установка параметров передачи осуществляется с помощью 3-х регистров UCSRA, UCSRB, UCSRC.

	7	6	5	4	3	2	1	0	
0x0B(0x2B)	**RXC**	**TXC**	**UDRE**	**FE**	**OR**	**PE**	**U2X**	**MPCM**	**UCSRA**
Исх. код	0	0	1	0	0	0	0	0	

Рисунок 5.4 – Регистр статус USART

RxC: Прием завершен. Этот бит устанавливается при пересылке принятого символа из сдвигового регистра приемника в регистр UDR. Бит устанавливается вне зависимости от отсутствия или наличия ошибки приема кадра. При установленном в регистре UCR бите RxCIE и установленном бите RxC выполняется прерывание по завершению приема UART. Бит RxC очищается при чтении регистра UDR. При приеме данных по прерыванию, процедуре обработки необходимо прочитать UDR для очистки бита RxC, иначе после ее завершения произойдет новое прерывание.

TxC: Передача завершена. Этот бит устанавливается, когда весь кадр (включая стоповый бит) выведен из сдвигового регистра передатчика, а в регистр UDR не записаны новые данные. Этот флаг используется при полудуплексном протоколе обмена, когда нужно освободить коммуникационную линию сразу после завершения передачи.

При установленном в регистре UCR бите TxCIE установка TxC приведет

к выполнению прерывания по завершению передачи UART. Флаг TxC очищается аппаратно при переходе по соответствующему вектору прерывания. Очистить бит TxC можно записью в бит логической «1».

UDRE: Регистр данных пуст. Этот бит устанавливается, когда весь символ, записанный в UDR, пересылается в сдвиговый регистр передатчика. Установка этого бита означает, что передатчик готов к получению нового символа.

Если бит UDRIE в UCR установлен, при установленном бите UDRE существует запрос прерывания по завершению передачи UART. Бит UDRE очищается при записи в регистр UDR. При приеме данных по прерыванию, процедура обработки прерывания должна прочитать регистр UDR, чтобы очистить бит UDRE, иначе после завершения процедуры произойдет новое прерывание.

FE: Ошибка кадра. Этот бит устанавливается, если при приеме стопового бита обнаружено состояние 0. Бит FE очищается при приеме стопового бита с логическим уровнем 1.

OR: Флаг переполнения. Устанавливается в 1, если заполнен регистр данных приемника, заполнен сдвиговый регистр приема и обнаружен стартбит на входе RXD.

PE: Флаг ошибки контроля четности. Устанавливается в 1, если в данных буфера приема выявлена ошибка четности.

U2X: Удвоение скорости обмена. Если разряд установлен, то коэффициент деления предделителя контроллера скорости уменьшается с 16 до 8; тем самым увеличивается скорость обмена в асинхронном режиме.

MPCM: Установка режима мультипроцессорного обмена.

	7	6	5	4	3	2	1	0	
0x0B(0x2B)	**RXCIE**	**TXCIE**	**UDRIE**	**FXEN**	**TXEN**	**CHR9**	**RXB8**	**TXB8**	**UCSRB**
Исх. код	0	0	0	0	0	0	0	0	

Рисунок 5.5 – Регистр управления USART

RXCIE: Разрешение прерывания по завершению приема. При установленном бите RxCIE и общем разрешении прерываний установка бита RxC приведет к прерыванию по завершению приема.

TXCIE: Разрешение прерывания по завершению передачи. При установленном бите TxCIE и общем разрешении всех прерываний установка бита TxC приведет к прерыванию по завершению передачи.

UDRIE: Разрешение прерывания по пустому регистру данных. При установленном бите UDRIE и общем разрешении прерываний установка бита UDRE приведет к прерыванию регистр данных UART пуст.

RXEN: Разрешение приемника. Установка в единицу подключает вход приемника к разряду PD0(RXD) и настраивает этот разряд порта на вход.

TXEN: Разрешение передатчика. Установка в единицу подключает выход передатчика к разряду PD1(TXD) и настраивает этот разряд порта на вывод.

CHR9: Формат кадра. Используется для задания размера кадра. Используется совместно с разрядами UCSZ1 и UCSZ регистра UCSRC.

RxB8: 8-разряд принимаемых данных. При установленном бите CHR9 бит RxB8 является девятым битом данных принятого кадра.

TxB8: 8-разряд передаваемых данных. При установленном бите CHR9 бит TxB8 является девятым битом данных передаваемого кадра.

	7	6	5	4	3	2	1	0	
0x20(0x40)	**URSEL**	**UMSEL**	**UPM1**	**UPM0**	**USBS**	**UCSZ1**	**UCSZ0**	**UCPOL**	**UCSRC**
Исх. код	1	0	0	0	0	1	1	0	

Рисунок 5.6 – Регистр управления USART

URSEL: разряд определяет в какой регистр производится запись. Если разряд установлен, то запись в регистр UCSRC, иначе запись в регистр установки скорости передачи UBRRH. Регистры UCSRC и UBRRH имеют одинаковый адрес и идентификация регистра производится этим разрядом.

UMSEL: режим работы блока. Если разряд сброшен, то асинхронный режим, если установлен – синхронный.

UPM1, UPM0: биты управления контролем четности 00 – контроль выключен, 10 – проверка на четность, 11 - проверка на нечетность.

USBS: количество стоповых битов, если USBS =0, то один стоп-бит, если USBS =1, то два стоповых бита.

UCSZ1 и UCSZ0: Совместно с битом UCSZ2 регистра UCSRB определяют количество битов данных в кадре.

UCPOL: В асинхронном режиме должен быть сброшен. В синхронном режиме разряд определяет момент выдачи и считывания бита на выводах модуля USART. Если равен нулю, то выдача данных на вывод TXD производится срезом импульса XCK, а считывание бита с вывода RXD производится фронтом импульса XCK. Если разряд равен единице, то наоборот.

Таблица 5.1 – Управление размером данных в кадре

UCSZ2	UCSZ1	UCSZ0	Количество разрядов данных на кадр
0	0	0	5 разрядов
0	0	1	6 разрядов
0	1	0	7 разрядов
0	1	1	8 разрядов
1	0	0	-
1	0	1	-
1	1	0	-
1	1	1	9 разрядов

5.2.8 Настройка скорости передачи

Скорость передачи задается контроллером скорости передачи, который представляет собой программируемый делитель частоты. Коэффициент деления задается содержимым регистра контроллера UBRR. Этот регистр фактически состоит из двух:

UBRRL – младшие 8 разрядов скорости передачи;

UBRRH – содержит старшие 4 разряда скорости передачи. Этот регистр совпадает по адресу с UCSRC. В какой регистр будет производиться запись определяется 7 разрядом URSEL. При записи в UBRRH этот разряд должен быть сброшен.

	7	6	5	4	3	2	1	0	
0x09(0x29)	**UBRR(7:0)**								**UBRL**
Исх. код	0	0	0	0	0	0	0	0	
	7	6	5	4	3	2	1	0	
0x20(0x40)	**URSEL**	-	-	-	**UBRR(11:8)**				**UBRH**
Исх. код	0	0	0	0	0	0	0	0	

Рисунок 5.7 – Регистр управления USART

В асинхронном режиме скорость передачи определяется также значением разряда U2X регистра UCSRA, если этот разряд установлен, то скорость передачи в два раза выше.

Скорость передачи в асинхронном режиме при U2X=0:

BAUD = FCK/16(UBRR+1).

Скорость передачи в асинхронном режиме при U2X=1:

BAUD = FCK/8(UBRR+1).

Скорость передачи в синхронном режиме для ведущего:

BAUD = FCK/2(UBRR+1)

где: BAUD - частота в бодах;

FCK - частота процессора;

UBRR = содержимое регистра UBRR (0 - 4095).

Необходимо заметить, что стандартные значения скоростей передачи для всего ряда (2400, 4800, 9600, 19200, 38400 и т.д.) могут быть получены только при нескольких значениях частоты тактового генератора и определенных значениях коэффициента деления. Ряд этих значений: 1,8432 МГц; 3,6864 МГц, 7,3728 МГц. Для любых других значениях тактовой частоты при любых значения коэффициента деления, полученные в контроллере значения скорости передачи отличаются от стандартных.

5.3 Последовательный периферийный интерфейс - SPI

Последовательный порт SPI (Serial Peripheral Interface) предназначен для связи МК с периферийными устройствами в микропроцессорной системе, основой которой он является. Шина SPI организована по принципу «ведущий-ведомый». В качестве ведущего шины обычно выступает микроконтроллер. Подключенные к ведущему шины внешние устройства образуют подчиненных шины. В их роли выступают различного рода микросхемы, в т.ч. запоминающие устройства (EEPROM, Flash-память, SRAM), АЦП/ЦАП, часы реального времени и др.

Внешние соединения между ведущим (мастером) и подчиненным ЦПУ через интерфейс SPI показаны на рисунке 5.8.

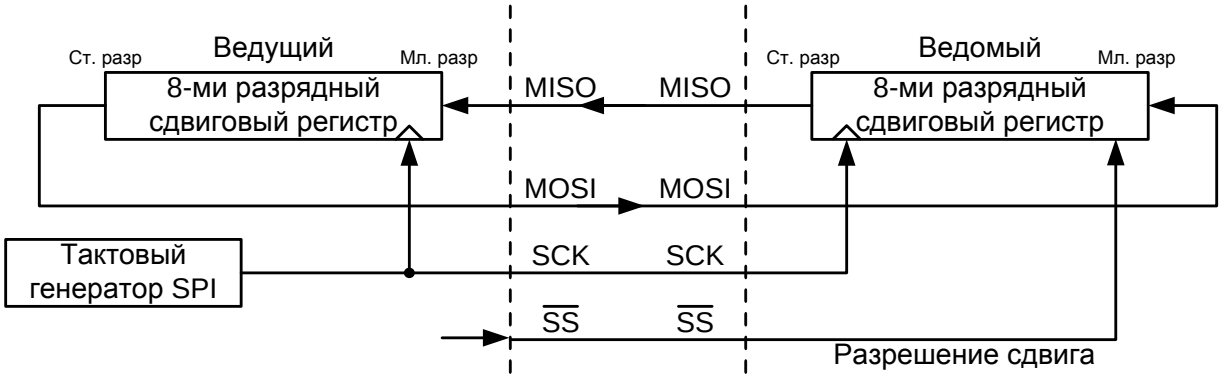


Рисунок 5.8 – Внешнее соединение ведущего и ведомого

Система состоит из двух сдвиговых регистров и генератора синхрониза-

ции. Ведущий инициирует сеанс связи подачей низкого уровня на вход SS того подчиненного устройства, с которым необходимо обмениваться данными. Оба респондента подготавливают данные к передаче в своем сдвиговом регистре, при этом на стороне ведущего генерируются также импульсы синхронизации на линии SCK. По линии MOSI (master - output, slave - input) всегда осуществляется передача данных от ведущего к подчиненному, а по MISO (master - input, slave - output), наоборот, от подчиненного к мастеру. По окончании передачи каждого пакета данных ведущий SPI должен засинхронизировать подчиненный путем подачи высокого уровня на линию SS.

Таким образом, главным блоком интерфейса SPI является обычный сдвиговый регистр, сигналы синхронизации которого и образуют интерфейсные сигналы. Непременным условием передачи данных по шине SPI является генерация сигнала синхронизации шины. Этот сигнал имеет право генерировать только ведущий шины и от этого сигнала полностью зависит работа подчиненного шины.

5.3.1 Временные диаграммы обмена

Существует четыре комбинации фазы и полярности сигнала SCK относительно передаваемых данных. Эти настройки индивидуальны для различных СБИС:

1) Передача по переднему фронту SCK (активный уровень SCK «1»);
2) Передача по переднему фронту SCK (активный уровень SCK «0»);
3) Передача по заднему фронту SCK (активный уровень SCK «1»);
4) Передача по заднему фронту SCK (активный уровень SCK «0»);

На рисунке 5.9 представлена временная диаграмма обмена по SPI для первого режима работы.

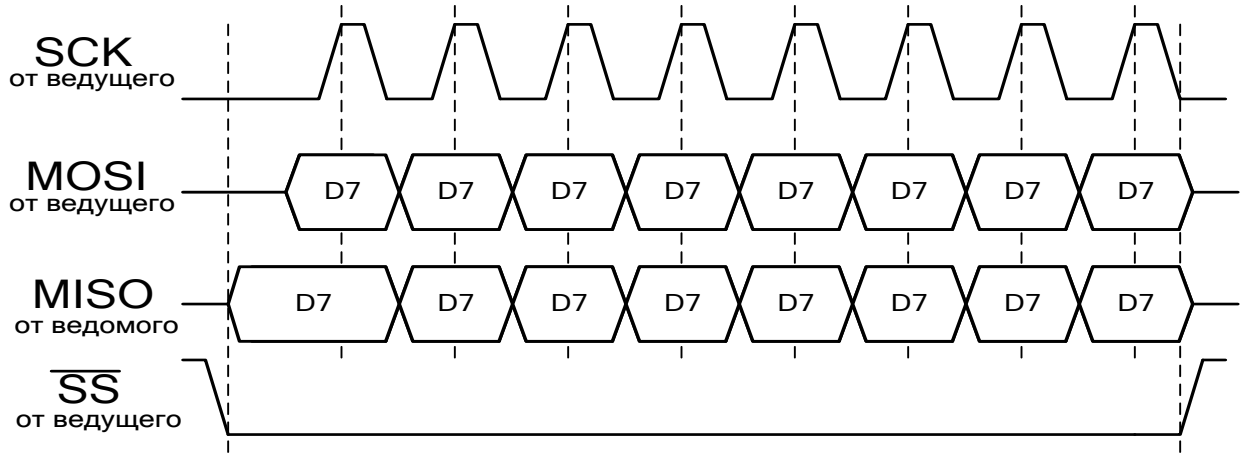


Рисунок 5.9 – Временная диаграмма обмена по SPI

5.3.2 Подключение нескольких микросхем к ведущему

При необходимости подключения к шине SPI нескольких микросхем используется параллельное подключение. Здесь, все сигналы, кроме выбора микросхем, соединены параллельно, а ведущий шины, переводом того или иного сигнала SS в низкое состояние, задает, с какой подчиненной ИС он будет обмениваться данными. Главным недостатком такого подключения является необходимость в дополнительных линиях для адресации подчиненных микросхем.

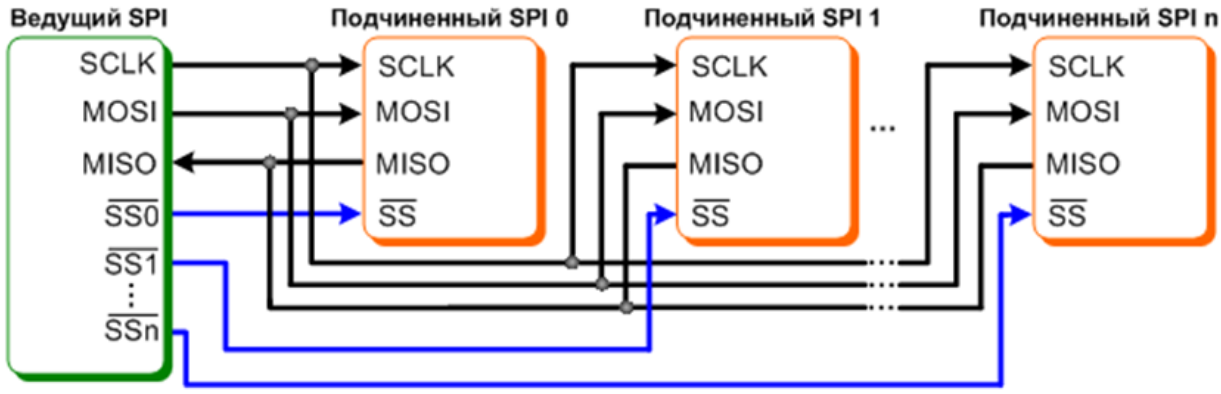


Рисунок 5.10 – Параллельное подключение нескольких ведомых к ведущему

Помимо параллельного подключения также может использоваться последовательное (каскадное) подключение по шине SPI, при котором выход передачи данных одной микросхемы соединяется со входом приема данных другой микросхемы, в этом случае задействуется меньше выводов МК.

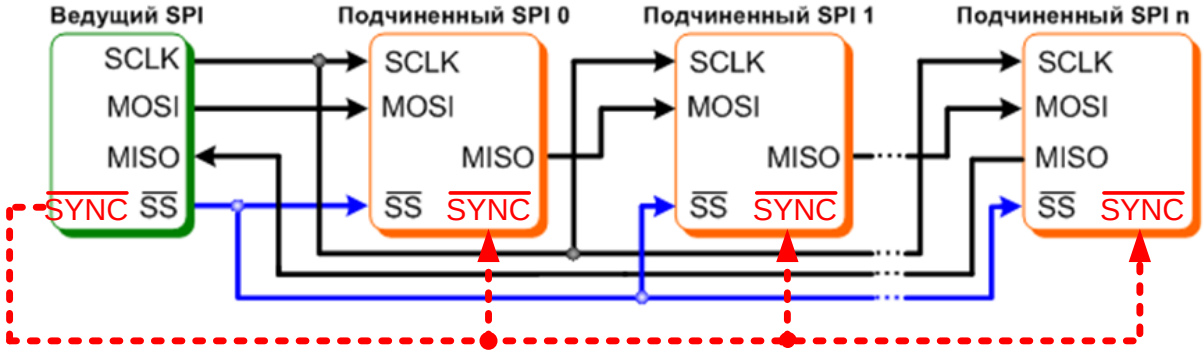


Рисунок 5.11 – Последовательное подключение ведомых к ведущему

В этом случае микросхемы образуют один большой сдвиговый регистр. Для загрузки этого составного сдвигового регистра новым содержимым придется передать через SPI последовательно n байтов информации. Микросхемы работающие в этом режиме входные данные буферируют, передача этих данных во внутренние регистры происходит по специальному стробирующему сигналу на входе –SYNC.

5.3.3 Устройство модуля SPI ATMega16A

В ATMega16A модуль SPI реализован на аппаратном уровне. Для обмена по SPI используются выводы PB5(MOSI), PB6(MISO), PB7(SCK) и PB4(SS). Структура модуля показана на рисунке

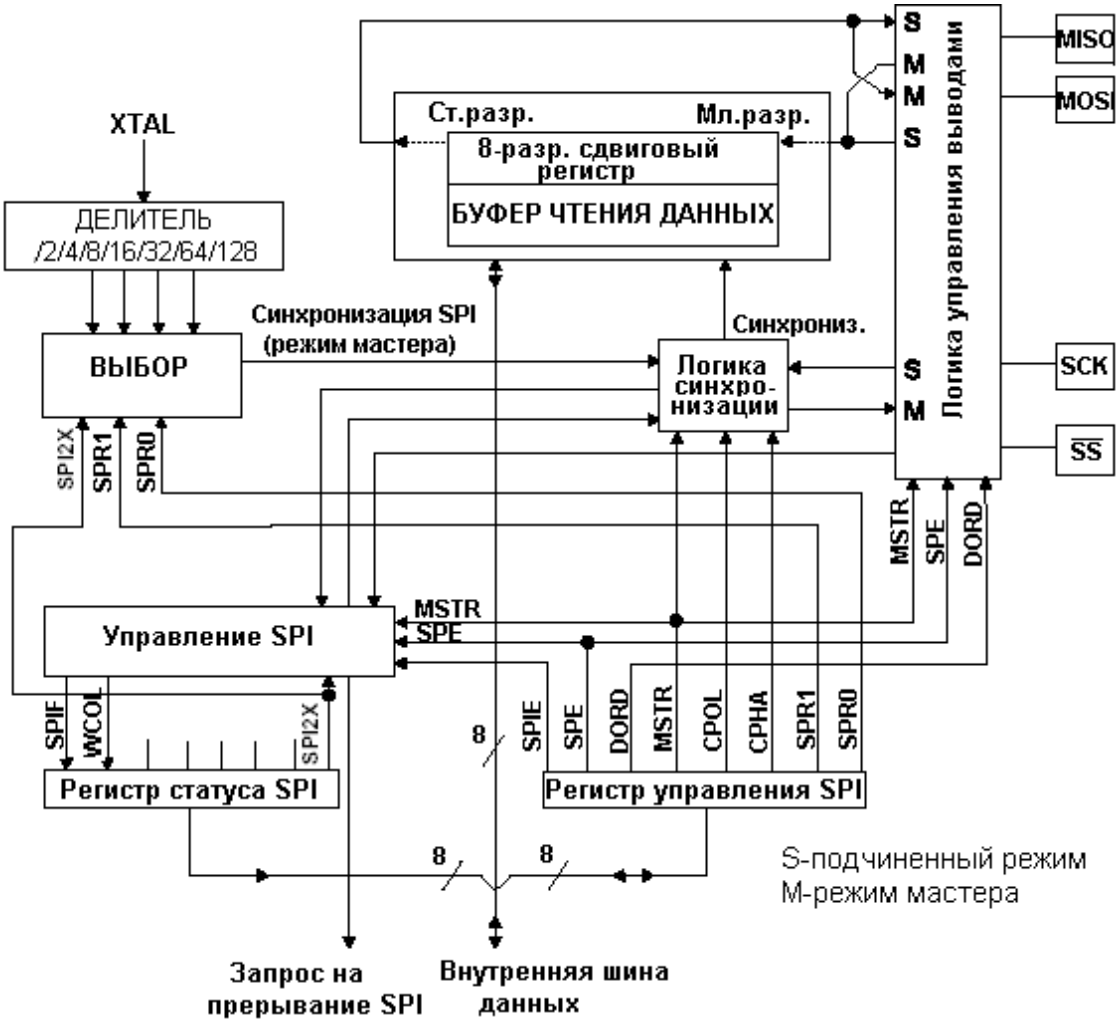


Рисунок 5.12 – Структура модуля SPI ATMega16A

В системе организована одиночная буферизация передающей стороны и двойная буферизация приемной стороны. Это означает, что передаваемые

символы не могут быть записаны в регистр данных SPI прежде, чем будет полностью завершен цикл сдвига. С другой стороны, при приеме данных принимаемый символ должен быть считан из регистра данных SPI прежде, чем будет завершен прием следующего символа, в противном случае предшествовавший символ будет потерян.

5.3.4 Функционирование линии SS

В режиме ведущего

Когда SPI-устройство определено как ведущее (бит MSTR регистра SPCR установлен), пользователь имеет возможность определить направление линии SS. Если вывод SS определен как выход, то он является выводом общего назначения и не участвует в работе интерфейса SPI.

Если же вывод SS определен как вход, то для обеспечения работы ведущего устройства SPI он должен удерживаться на высоком уровне. Если в режиме ведущего вывод SS является входом и внешней схемой на него подан низкий уровень, то интерфейс SPI воспримет его как обращение другого ведущего SPI к себе, как к ведомому, и переходит в режим ведомого. Таким образом, когда управляемое прерыванием передающее устройство SPI используется в режиме ведущего и существует вероятность подачи на вывод SS активного сигнала низкого уровня, процедура прерывания должна проверять установку бита MSTR. Если бит MSTR был очищен выбором режима ведомого, то он должен быть установлен пользователем для переназначения устройства ведущим.

В режиме ведомого

Если порт SPI является ведомым, то вывод SS постоянно работает как вход. При подаче на вывод SS низкого уровня устройство SPI активируется и вывод MISO становится выходом, если это определено пользователем. Все остальные выводы являются входами. Если вывод SS удерживается на высоком уровне, то все выводы являются входами, устройство SPI пассивно и не будет получать входных данных.

5.3.5 Регистры управления

Управление модулем SPI осуществляется с помощью регистра управления SPCR, информацию о состоянии модуля можно получить из регистра статуса SPSR. Для ввода и вывода данных используется регистр SPDR.

	7	6	5	4	3	2	1	0	
0x0F(0x2F)	**SPDR(7:0)**								**SPDR**
Исх. код	0	0	0	0	0	0	0	0	

Рисунок 5.13 – Регистр данных SPDR

Регистр данных SPDR представляет собой регистр с возможностью чтения/записи и предназначен для пересылки данных между регистровым файлом и сдвиговым регистром SPI. Запись в регистр SPDR инициирует передачу данных, считывание регистра приводит к чтению сдвигового регистра приемника.

	7	6	5	4	3	2	1	0	
0x0D(0x2D)	**SPIE**	**SPE**	**DORD**	**MSTR**	**CPOL**	**CPHA**	**SPR1**	**SPR0**	**SPCR**
Исх. код	0	0	0	0	0	0	0	0	

Рисунок 5.14 – Регистр данных SPDR

SPIE: Разрешение прерывания SPI. Установка этого бита разрешает установку бита SPIF регистра SPSR и, при разрешении всех прерываний битом I регистра SREG, обслуживание прерывания порта SPI.

SPE: Разрешение работы порта SPL. Установка этого бита разрешает подключение линий SS, MOSI, MISO и SCK к выводам PBO, PB1, PB2 и PB3.

DORD: Порядок пересылки данных. Установка этого бита приводит к передаче посылки данных вперед младшим битом. При очищенном бите первым передается старший бит данных.

MSTR: Выбор режима ведущий/ведомый. При установленном бите MSTR порт SPI работает в режиме ведущего, а при очищенном бите - в режиме ведомого. Если SS определен как вход и на него подан низкий уровень при установленном бите MSTR, бит MSTR будет сброшен, а бит SPIF регистра SPSR установлен. Чтобы вновь установить режим ведущего устройства, пользователь должен установить бит MSTR.

CPOL: Полярность тактового сигнала. При установленном бите CPOL сигнал SCK в режиме Idle имеет высокий уровень, при сброшенном бите CPOL - низкий уровень.

CPHA: Фаза тактового сигнала. Действие этого бита отображено на рисунке __.

SPR1, SPRO: Выбор частоты тактового сигнала. Эти два бита управляют частотой тактового сигнала устройства, работающего в режиме ведущего. В режиме ведомого значения битов влияния не оказывают. Состояния битов и устанавливаемый коэффициент деления частоты FCL указаны в таблице __.

Таблица 5.2 – Управление размером данных в кадре

SPR1	SPR0	Тактовая частота SCK
0	0	FCL/4
0	1	FCL/16
1	0	FCL/64
1	1	FCL/128

Формат регистра состояния канала SPI представлен на рисунке.

	7	6	5	4	3	2	1	0	
0x0D(0x2D)	**SPSR**	**WCOL**	-	-	-	-	-	**SPI2X**	**SPSR**
Исх. код	0	0	0	0	0	0	0	0	

Рисунок 5.15 – Регистр данных SPDR

SPIF: Флаг прерывания порта SPI. После завершения обмена бит SPIF устанавливается и, если установлены биты SPIE в регистре SPCR и I в регистре SREG, прерывание порта SPI обрабатывается. Бит SPIF очищается аппаратно при переходе на вектор прерывания. Бит SPIF может быть очищен также при первом чтении регистра состояния SPSR и последующем обращении к регистру данных SPDR.

WCOL: Флаг ошибки при записи. Бит WCOL устанавливается, если в процессе передачи данных выполнялась запись в регистр данных SPDR. Бит WCOL (и бит SPIF) аппаратно очищаются при первом чтении регистра SPSR и последующем обращении к регистру данных SPDR.

SPI2X: Бит удвоения скорости SPI. Если в этот бит записать «1» то скорость работы SPI (частота SCK) удвоится, если SPI находится в режиме ведущего. Если SPI работает как подчиненный, то работа SPI гарантирована только на частоте FCL/4 или менее.

5.4 Двухпроводной последовательный интерфейс TWI (I2C)

5.4.1 Подключение устройств к шине

I2C (у ATMega16 носит название TWI) - это последовательная шина данных для связи интегральных схем, использующая две линии связи. Шина поддерживается двунаправленную передачу между ведущими и ведомыми устройствами. Для связи нескольких микросхем по шине I2C(TWI) не требуется дополнительных линий, так как обращение в отличие от SPI осуществляется по адресу. Шина поддерживает скорость передачи до 100 кбит/сек.

I2C использует две двунаправленные линии, подтянутые к напряжению

питания и управляемые через открытый коллектор или открытый сток — последовательная линия данных SDA (serial data) и последовательная линия тактирования SCL (serial clock). Стандартные напряжения +5В или +3,3В, однако допускаются и другие.

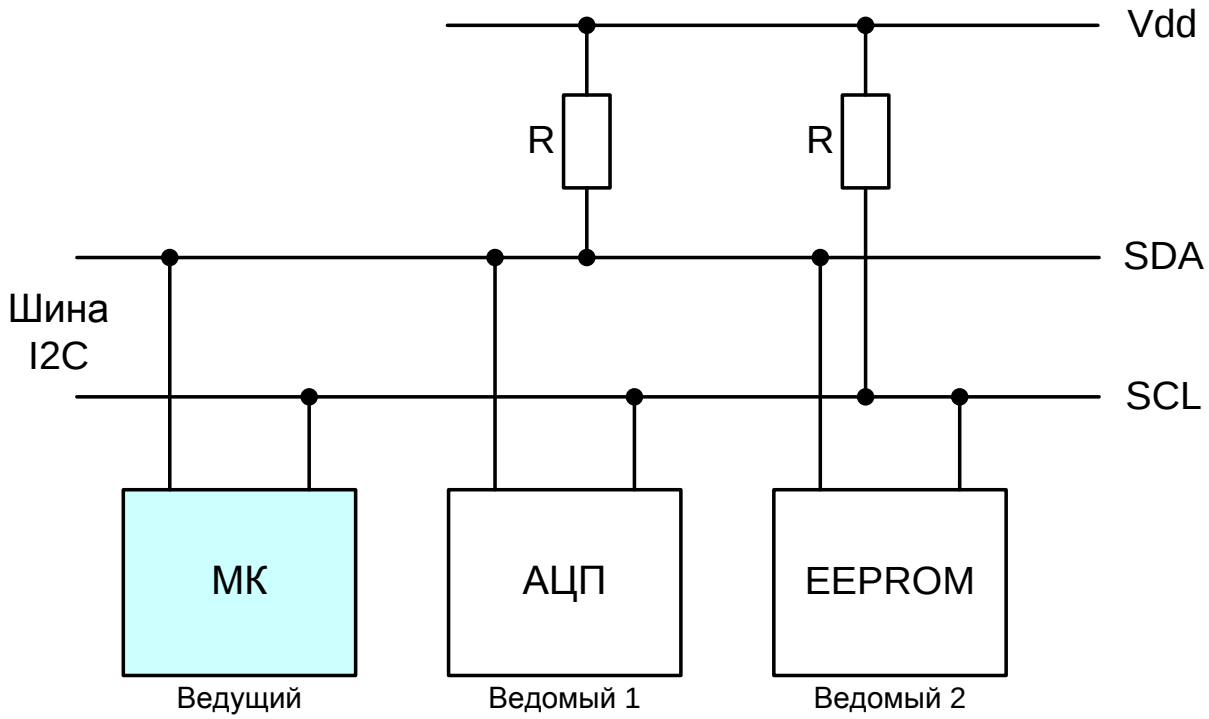


Рисунок 5.16 – Конфигурация шины I2C

5.4.2 Принципы организации связи

Временная диаграмма обмена приведена на рисунке

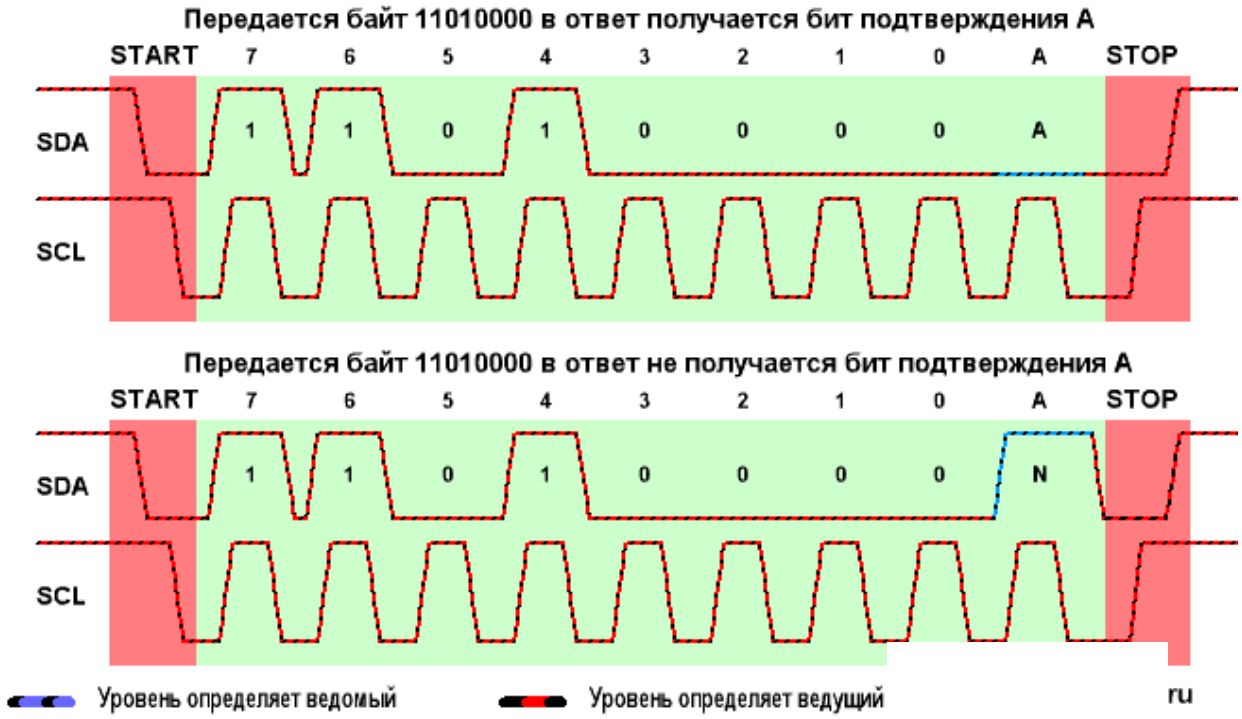


Рисунок 5.17 – Временная диаграмма обмена по шине

Адресация

В протоколе интерфейса I2C каждое устройство имеет уникальный адрес. Классическая адресация включает 7-битное адресное пространство с 16 зарезервированными адресами. Это означает до 112 свободных адресов для подключения периферии на одну шину. Когда ведущий хочет инициировать передачу данных, он сначала передает адрес устройства, к которому хочет обратиться. Все устройства на шине следят за выставляемым на шину адресом и сравнивают его с собственным.

Подтверждение

Вместе с адресом передается бит направления передачи R/W, который определяет, тип предстоящей операции запись или чтение. Каждый переданный кадр от ведущего ведомое обязано подтверждать битом подтверждения.

передаются подчиненным устройством к главному. Главное устройство возвращает бит подтверждения после всех принятых байтов, кроме последнего байта. В конце последнего принятого байта возвращается «нет подтверждения».

Условия старта и остановки

Когда нет передачи данных (режим ожидания), линии тактирования SCL и данных SDA приведены подтягивающим резистором к высокому уровню. Главное устройство генерирует все последовательные синхроимпульсы и условия START и STOP, определяющие начало и конец передачи данных.

Условие START определяется как переход SDA из высокого уровня в низкий при высоком уровне SCL, а условие STOP – как переход SDA из низкого уровня в высокий при высоком уровне SCL. Ввиду такого способа определения условий START и STOP при передаче данных, линия SDA может изменять свое состояние только при низком уровне SCL.

Когда ведущий не желает освобождать шину (выставив STOP), он должен выставить повторное условие START, которое идентично START (переход SDA из высокого уровня в низкий при высоком уровне на SCL), но выдается вслед за подтверждением, являясь началом следующей последовательной передачи, таким образом шина не освобождается. На рисунке показано, как осуществляется передача данных по шине в соответствии с принятым стандартом протокола связи.

Бит запись/чтение

В зависимости от направления передачи битов (R/W) для I2C-шины возможны два типа передачи данных:

Передача данных от главного передатчика к подчиненному приемнику. Первый байт, передаваемый передатчиком, является адресом подчиненного приемника. Затем следует несколько байтов данных. Подчиненный приемник возвращает бит подтверждения после каждого принятого байта.

Передача данных от подчиненного передатчика к главному приемнику. Первый байт (адрес подчиненного передатчика) передается главным устройством. Затем подчиненный передатчик возвращает бит подтверждения. Следующие несколько байтов данных

Рисунок 5.18 – Приостановка обмена ведомым

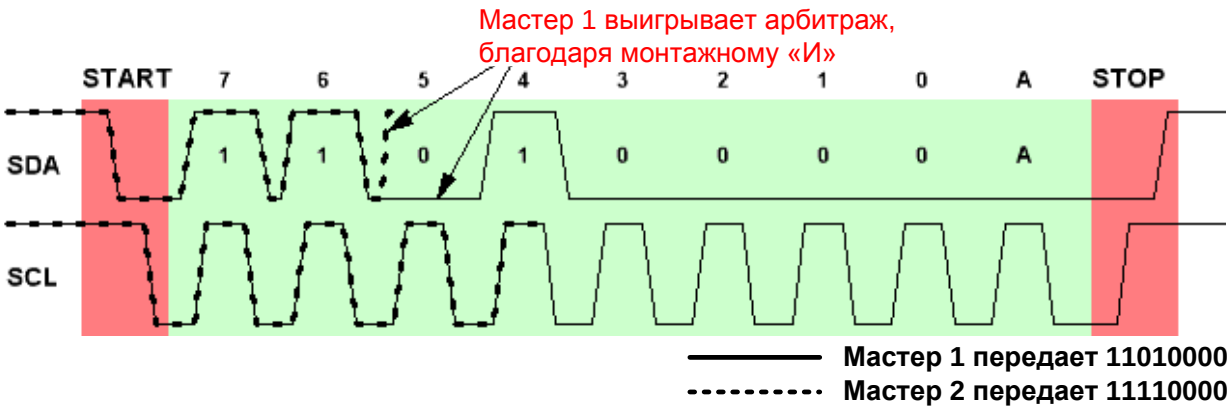


Рисунок 5.19 – Реализация арбитража на шине

5.4.3 Описание модуля TWI ATMega16A

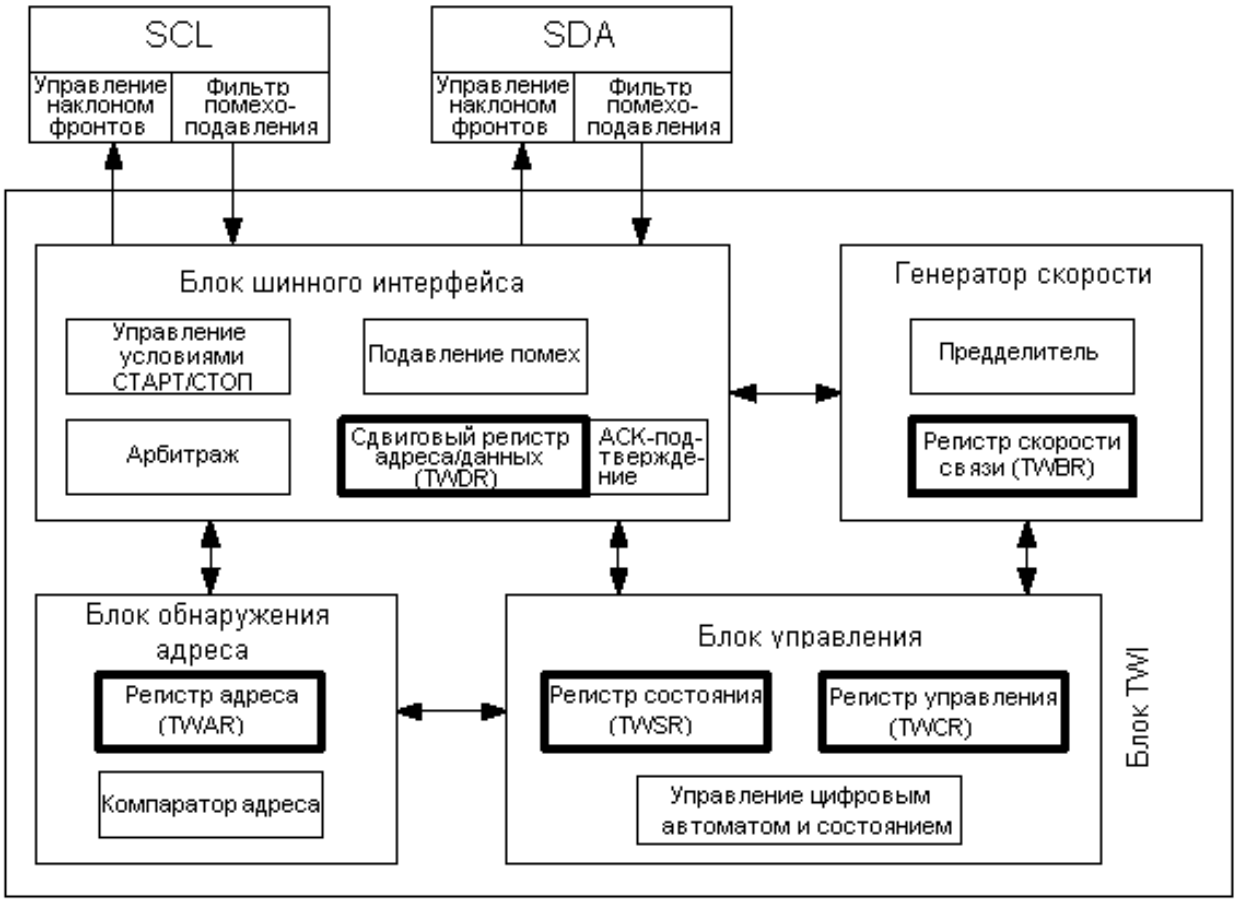


Рисунок 5.20 – Функциональная схема TWI ATMega16A

Выводы SCL и SDA

Данные выводы связывают двухпроводной интерфейс микроконтроллера с остальными микроконтроллерами в системе. Драйверы выходов содержат ограничитель скорости изменения фронтов для выполнения требований к TWI. Входные каскады содержат блок подавления помех, задача которого состоит в игнорировании импульсов длительностью менее 50 нс. Обратите внимание, что к каждой из этих линий можно подключить внутренний подтягивающий резистор путем установки разрядов PORTD.0 (SCL), PORTD.1 (SDA) (см. также "Порты ввода-вывода"). Использование встроенных подтягивающих резисторов в ряде случаев позволяет отказаться от применения внешних.

Блок генератора скорости связи

Данный блок управляет периодом импульсов SCL в режиме ведущего устройства. Период SCL задается регистром скорости TWI (TWBR) и значением бит управления предделителем в регистре состояния TWI (TWSR). В подчиненном режиме значения скорости или установки предделителя не оказывают влияния на работу, но частота синхронизации ЦПУ подчиненного устройства должна быть минимум в 16 раз выше частоты SCL.

Блок шинного интерфейса

Данный блок содержит сдвиговый регистр адреса и данных (TWDR), контроллер СТАРТа/СТОПа и схему арбитрации. TWDR содержит передаваемый байт адреса или данных, или принятый байт адреса или данных. Помимо 8-разр. регистра TWDR в состав блока шинного интерфейса также входит регистр, хранящий значение передаваемого или принятого бита (НЕТ) ПОДТВ. К данному регистру нет прямого доступа со стороны программного обеспечения. Однако во время приема он может устанавливаться или сбрасываться путем манипуляций с регистром управления TWI (TWCR). В режиме передатчика значение принятого бита (НЕТ) ПОДТВ можно определить по значению регистра TWSR.

Контроллер СТАРТа/СТОПа отвечает за генерацию и детекцию условий СТАРТ, ПОВТОРНЫЙ СТАРТ и СТОП. Контроллер СТАРТа/СТОПа позволяет обнаружить условия СТАРТ и СТОП, даже если микроконтроллер находится в одном из режимов сна. Этим обеспечивается возможность пробуждения микроконтроллера по запросу ведущего шины.

Если TWI инициировал передачу в качестве ведущего, то схема арбитрации непрерывно контролирует передачу, определяя возможность дальнейшей передачи. Если TWI теряет арбитрацию, то блок формирует соответствующий сигнал блоку управления, который выполняет адекватные действия и генерирует соответствующий код состояния.

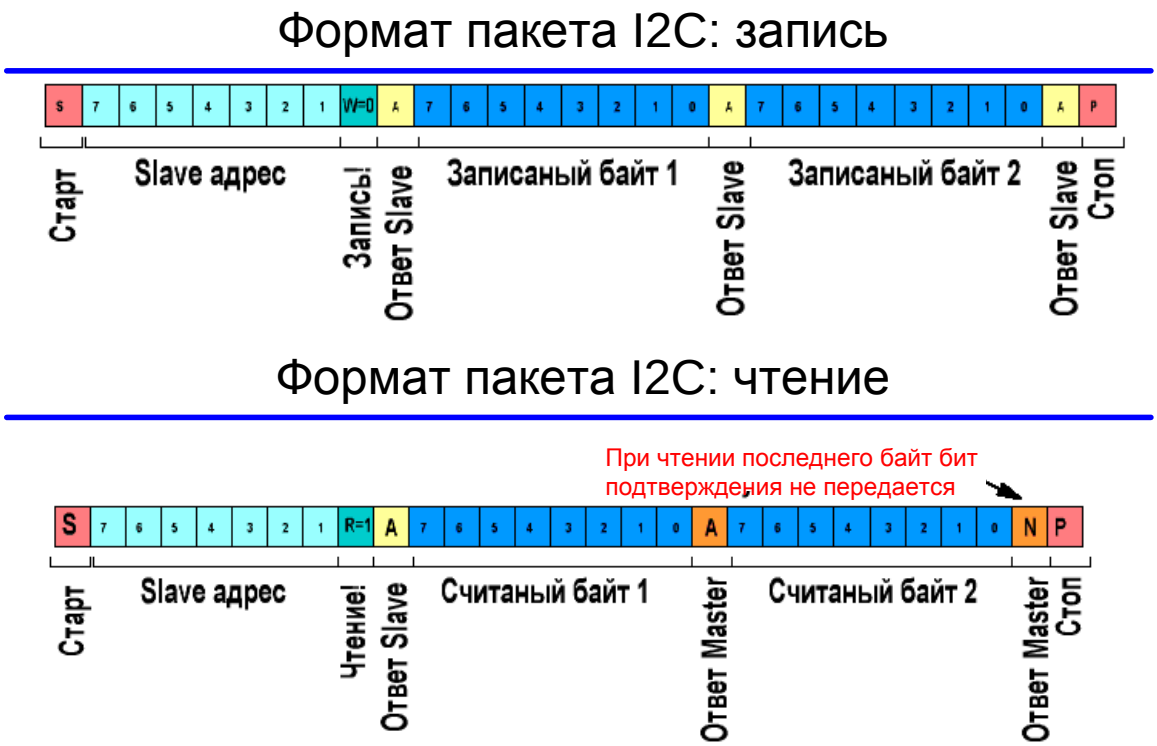
Блок обнаружения адреса

Блок обнаружения адреса проверяет равен ли принятый адрес значению 7-разр. адреса из регистра TWAR. Если установлен бит разрешения обнаружения общего вызова TWGCE в регистре TWAR, то все входящие адресные биты будут дополнительно сравниваться с адресом общего вызова. При адресном совпадении подается сигнал блоку управления, что позволяет выполнить ему необходимые действия. В зависимости от установки регистра TWCR подтверждение адреса TWI может происходит, а может и нет. Блок обнаружения адреса способен функционировать даже, когда микроконтроллер переведен в режим сна, тем самым позволяя возобновить нормальную

работу микроконтроллера по запросу мастера шины. Если при адресном совпадении TWI в экономичном режиме микроконтроллера, т.е. когда инициируется возобновление работы микроконтроллера, возникает другое прерывание (например, INT0), то TWI прекращает работу и возвращается к состоянию холостого хода (Idle). Если возникновение данного эффекта нежелательно, то необходимо следить, чтобы во время обнаружения адресования, когда микроконтроллер находится в режиме выключения (Power-down), было разрешено только одно прерывание.

Блок управления

Блок управления наблюдает за шиной TWI и генерирует отклики в соответствии с установками регистра управления TWI (TWCR). Если на шине TWI возникает событие, которое требует внимания со стороны программы, то устанавливается флаг прерывания TWINT. Следующим тактом обновляется содержимое регистра статуса TWI - TWSR, в котором будет записан код, идентифицирующий возникшее событие. Даная информация хранится в TWSR только тогда, когда установлен флаг прерывания TWI. Остальное время в регистре TWSR содержится специальный код состояния, который информирует о том, что нет информации о состоянии TWI. До тех пор пока установлен флаг TWINT линия SCL остается в низком состоянии. Этим обеспечивается возможность завершить программе все задачи перед продолжением сеанса связи.



5.5 Контрольные вопросы

1) Назначение и характеристики модуля USART ATMega16A.
2) Назначение линий используемых для обмена модулем USART.
3) Формат кадра USART, назначение битов в кадре.
4) Обмен в асинхронном режиме.
5) Обмен в синхронном режиме.
6) Подключение МК по интерфейсу RS-232.
7) Подключение МК по интерфейсу RS-485.
8) Механика работы модуля USART при приеме данных.
9) Механика работы модуля USART при передаче данных
10) Назначение режимов энергосбережения. Как перевести МК в режим энергосбережения?

6 ПРОЧИЕ УСТРОЙСТВА ATMEGA16

6.1 Сторожевой таймер (Watchdog Timer)

Сторожевой таймер используется для выхода из «зависания», которое может произойти при работе микроконтроллера в условиях сильных помех. Аппаратно сторожевой таймер представляет собой суммирующий 16-разрядный счетчик. При его переполнении формируется внутренний сигнал сброса микроконтроллера.

6.1.1 Принцип использования сторожевого таймера.

Приложение, кроме выполнения рабочих функций, должно обеспечивать периодический сброс сторожевого таймера. Правильно работающее приложение должно периодически обнулять сторожевой таймер (см. рисунок 6.1). Если приложение будет повреждено, то сторожевой таймер по окончании цикла счета произведет сброс программы (см. рисунок 6.2).

Сторожевой таймер тактируется отдельным встроенным генератором с частотой 1 МГц. Установкой коэффициента деления тактовой частоты можно изменять длительность интервала до сброса по сторожевому таймеру от 16К(16384) отсчетов до 2048К (2097152) отсчетов (от 16 до 2048 мс). Программно сторожевой таймер сбрасывается командой WDR (Watchdog Reset).

С момента сброса сторожевого таймера до завершения внутреннего сброса микроконтроллера длится период времени, который зависит от коэффициента деления тактовой частоты таймера. Если этот период завершился и другой сигнал сброса сторожевого таймера не поступил, микроконтроллер начинает работу с вектора сброса.

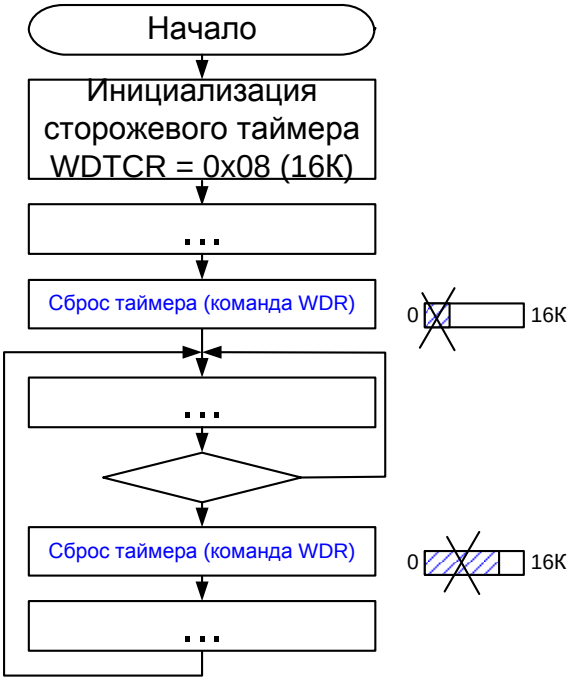


Рисунок 6.1 – Работа сторожевого таймера при нормальном исполнении программы

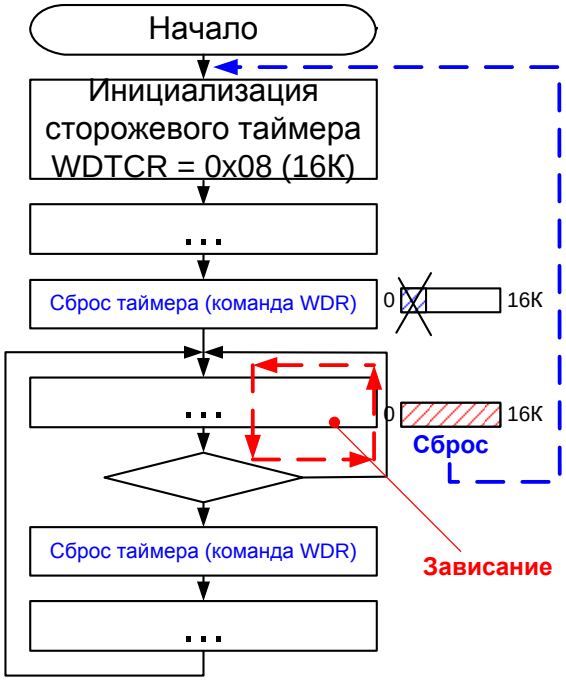


Рисунок 6.2 – Работа сторожевого таймера при «зависании» программы

6.1.2 Программная работа со сторожевым таймером

Для управления сторожевым таймером служит регистр WDTCR.

	7	6	5	4	3	2	1	0	
0x21(0x41)	–	–	–	WDTOE	WDE	WDP2	WDP1	WDP0	**WDTCR**
Исх. код	0	0	0	0	0	0	0	0	

Рисунок 6.3 – Регистр управления сторожевым таймером

WDTCR.7..5 -Зарезервированные биты.

WDTCR.4 - WDTOE: Бит разрешения отключения сторожевого таймера. Этот бит должен быть установлен («1») при очистке бита WDE. Иначе сторожевой таймер не будет запрещен. Если бит WDTOE установлен, то сторожевой таймер аппаратно очищается через четыре тактовых цикла.

WDTCR.3 - WDE: Бит разрешения сторожевого таймера. Если бит WDE установлен («1»), то сторожевой таймер разрешен. Если бит WDE очищен («0»), то функционирование сторожевого таймера запрещено. Бит WDE может быть очищен только если установлен бит WDTOE.

WDTCR.2..0 - WDP2, WDP1, WDPO: Биты установки коэффициента

предварительного деления сторожевого таймера. Состояние битов WDP2, WDP1 и WDPO определяет коэффициент деления тактовой частоты сторожевого таймера (см. таблицу 6.1).

Таблица 6.1 – Коэффициенты деления частоты сторожевого таймера

WD2	WD1	WD0	Длительность цикла сторожевого таймера	Длительность периода сброса при VCC=5В , мс
0	0	0	16К	16
0	0	1	32К	32
0	1	0	64К	64
0	1	1	128К	128
1	0	0	256К	256
1	0	1	512К	512
1	1	0	1024К	1024
1	1	1	2048К	2048

Для запуска сторожевого таймера выполняется следующая последовательность действий:

– Выполнить команду WDR;
– Загрузить значение в счетчик команд (установить биты WDP2, WDP1, WDP0);
– Записать «1» в WDE;

```
WDT_ON:
        wdr
        ldi R16, 0b00000010    ;Инициализация таймера на 64мс
        out WDTCR, r16
        ldi R16, 0b00001010    ;Установка бита WDE
        out WDTCR, r16
        ret
```

Для запрещения работы сторожевого таймера необходимо выполнить следующую процедуру:

– Записать «1» в WDTOE и WDE. Логическая «1» должна быть записана в WDE, даже если этот бит был установлен перед началом операции запрета сторожевого таймера.
– За время последующих четырех циклов записать логический 0 в WDE. Сторожевой таймер будет запрещен.

```
WDT_OFF:
        ;Запись лог. 1 в WDTOE и WDE
        ldi R16, 0b00011000
        out WDTCR, r16
        ;Выкл. сторожевого таймера WDE=0
        ldi R16, 0b00001000
        out WDTCR, r16
```

```
ret
```

6.2 Энергонезависимая память данных

Микроконтроллер ATmega16 имеет в своем составе энергонезависимую память данных EEPROM емкостью 512 байт, содержимое которой не изменяется при выключении источника питания микроконтроллера. Энергонезависимая память данных используется для хранения изменяемых в процессе функционирования МК данных и настроек, значения которых должны сохраниться после выключения питания МК.

Для работы с EEPROM используются три регистра ввода-вывода: регистр адреса EEAR, регистр данных EEDR и регистр управления EECR.

Регистр адреса состоит из двух регистров: старших разрядов EEARH (адрес 0x1F(0x3F)) и младших разрядов EEARL (адрес 0x1E(0x3E)). При емкости 512 байт в регистре EEARH задействован только один разряд. В эти регистры помещается адрес ячейки EEPROM, к которой производится обращение.

В регистр данных (адрес 0x1D(0x3D)) помещается байт данных, который будет помещен в EEPROM при записи. При чтении памяти в него помещается байт, считанный из EEPROM.

Регистр управления EECR определяет доступ к EEPROM.

	7	6	5	4	3	2	1	0	
0x1C(0x3C)	-	-	-	-	EERIE	EEMWE	EEWE	EERE	**EECR**
Исх. код	0	0	0	0	0	0	0	0	

Рисунок 6.4 – Регистр управления энергонезависимой памятью данных

Бит EERE – бит чтения байта данных из EEPROM. При установке в состояние «1» байт данных из EEPROM загружается в регистр EEDR.

Бит EEMWE – бит разрешения записи в EEPROM. Единичное состояние этого разряда разрешает запись в EEPROM, а нулевое – запрещает.

Бит EEWE – бит записи в EEPROM. При установке этого разряда производится запись байта в EEPROM из регистра данных EEDR. По окончании записи этот бит сбрасывается аппаратно.

Бит EERIE – бит разрешения прерывания по окончании записи в EEPROM. Если этот разряд установлен и установлен бит I в регистре SREG (бит общего разрешения прерывания), то прерывания после записи в EEPROM будут разрешены. Адрес вектора прерывания от EEPROM – 0x1E.

Чтение и запись

Процедура записи в EEPROM:

- Загрузить байт данных в регистр EEDR.
- Загрузить адрес ячейки EEPROM в регистр EEAR.
- Установить бит EEMWE – разрешения записи.
- Установить бит EEWE записи в EEPROM.

Время записи байта в EEPROM составляет 7-9 мс. По окончании записи бит EEWE аппаратно сбрасывается, после чего можно начинать следующую запись.

Процедура чтения из EEPROM:

- Загрузить адрес ячейки EEPROM в регистр EEAR.
- Установить бит EERE – чтения EEPROM.
- Считать байт данных из регистра EEDR, полученный из памяти

После считывания байта данных в регистр EEDR бит EERE сбрасывается аппаратно.

6.3 Режимы энергосбережения

Спящий режим может оказаться очень полезным, если микроконтроллер какое-то время не производит никаких операций, а просто ожидает какого-нибудь события. А также, для выключения не используемой периферии МК в целях экономии ресурсов батареи питания.

Для перевода микроконтроллера в один из шести режимов сна необходимо предварительно установить бит SE в регистре MCUCR, а затем выполнить ассемблерную инструкцию SLEEP. Биты SM2, SM1 и SM0 регистра MCUCR задают в какой именно режим будет переведен микроконтроллер после выполнения команды SLEEP. Выход из режима сна происходит при возникновении разрешенного прерывания. В этом случае, помимо времени старта микроконтроллер приостанавливается на 4 машинных цикла, выполняет процедуру обработки прерывания и продолжает выполнять команды следующие за SLEEP. Содержимое файла регистров и статического ОЗУ остается неизменным после выхода из режима сна.

	7	6	5	4	3	2	1	0	
0x35(0x55)	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	**MCUCR**
Исх. код	0	0	0	0	0	0	0	0	

Рисунок 6.5 – Регистр управления МК

Бит 6 - SE: Разрешение спящего режима. Этот бит должен быть установлен в «1», чтобы МК смог войти в один из режимов сна.

Биты 7,5,4 - SM2:0: Выбор спящего режима

Таблица 6.2 – Управление типом режима сна

SM2	SM1	SM0	Наименование режима сна
0	0	0	Холостой ход («Idle»)
0	0	1	Уменьшение шумов АЦП («ADC Noise Reduction Mode»)
0	1	0	Выключение (Power-down)
0	1	1	Экономичный («Power-save»)
1	0	0	Зарезервирован
1	0	1	Зарезервирован
1	1	0	Дежурный («Standby»)
1	1	1	Расширенный дежурный («Extended Standby»)

Холостой ход. В этом режим останавливается ЦПУ, а периферия - SPI, USART, Аналоговый компаратор, ADC, TWI, таймеры/счетчики, сторожевой таймер и система прерываний продолжает работать.

Уменьшение шумов АЦП. В этом режиме останавливается процессор, но АЦП, внешние прерывания, TWI, таймер/счетчик 2, сторожевой таймер (если включен) продолжают работать. Этот режим служит для уменьшения разных наводок во время преобразования АЦП. Кроме прерывания по завершению преобразования АЦП, микроконтроллер из этого режима энергосбережения может вывести внешний сброс, сброс сторожевым таймером, прерывание TWI, прерывание таймера/счетчика2, прерывание готовности EEPROM, изменение уровня на INT0 или INT1.

Выключение. Самый экономный режим. В этом режиме останавливается все, что есть в микроконтроллере, кроме сторожевого таймера (если его включить), внешних прерываний и TWI. Только внешний сброс, сброс сторожевым таймером, прерывание TWI или изменение уровня на INT0 или INT1 может разбудить микроконтроллер. В этом режиме останавливается тактовый генератор, поэтому чтобы проснуться микроконтроллеру может понадобиться какое-то время.

Экономичный режим. Этот режим похож на режим «Выключение». Отличается он тем, что если таймер/счетчик 2 работает асинхронно, то он продолжит свою работу и во время сна. Это может быть полезно при реализации часов реального времени на микроконтроллере.

Дежурный и расширенный дежурный режимы. Эти режимы также похожи на режимы «Выключение». Но в этих режимах тактовый генератор

продолжает работать (если установлен внешний кварц). Из этих режимов МК просыпается за 6 тактов.

6.4 Контрольные вопросы

1) Назначение режимов энергосбережения. Как перевести МК в режим энергосбережения?
2) Назначение и принцип работы сторожевого таймера

ПРИЛОЖЕНИЕ А

(справочное)

Система команд

Таблица А.1 – Набор инструкций ATMega16A

Мнемокод	Операнды	Описание	Действие	Кол. цик-лов
Арифметические и логические инструкции				
ADD	Rd, Rr	Сложить два регистра	Rd <— Rd + Rr	1
ADC	Rd, Rr	Сложить два регистра с переносом	Rd <— Rd + Rr + C	1
ADIW	Rdl,K	Сложить слово с константой	Rdh:Rdl <— Rdh:Rdl + K	2
SUB	Rd, Rr	Вычесть два регистра	Rd <— Rd - Rr	1
SUBI	Rd, K	Вычесть константу из регистра	Rd <— Rd - K	1
SBC	Rd, Rr	Вычесть два регистра с учетом переноса	Rd <— Rd - Rr - C	1
SBCI	Rd, K	Вычесть константу из регистра с учетом переноса	Rd <— Rd - K - C	1
SBIW	Rdl,K	Вычесть константу из слова	Rdh:Rdl <— Rdh:Rdl-K	2
AND	Rd, Rr	Логическое И между регистрами	Rd <— Rd · Rr	1
ANDI	Rd, K	Логическое И между регистром и константой	Rd <— Rd · K	1
OR	Rd, Rr	Логическое ИЛИ между регистрами	Rd <— Rd v Rr	1
ORI	Rd, K	Логическое ИЛИ между регистром и константой	Rd <— Rd v K	1
EOR	Rd, Rr	Искл. ИЛИ между регистрами	Rd <— Rd □ Rr	1
COM	Rd	Дополнение до 0b11111111 ($FF), инверсия	Rd <— $FF - Rd	1
NEG	Rd	Дополнение до 0b00000000 ($00)	Rd <— $00 - Rd	1
SBR	Rd, K	Установка бит (бита) в регистре	Rd <— Rd v K	1
CBR	Rd, K	Сброс бит (бита) в регистре	Rd <— Rd · ($FF - K)	1
INC	Rd	Инкремент	Rd <— Rd + 1	1
DEC	Rd	Декремент	Rd <— Rd - 1	1

TST	Rd	Проверка на ноль или минус	Rd <— Rd . Rd	1
CLR	Rd	Сброс регистра	Rd <— Rd Rd	1
SER	Rd	Установка регистра	Rd <— $FF	1
MUL	Rd, Rr	Умножение без знака	R1:R0 <— RdxRr	2
MULS	Rd, Rr	Умножение со знаком	R1:R0 <— RdxRr	2
MULSU	Rd, Rr	Умножение знакового с беззнаковым числом	R1:R0 <— Rd x Rr	2
FMUL	Rd. Rr	Дробное умножение без знака	R1:R0 <— (RdxRr) << 1	2
FMULS	Rd, Rr	Дробное умножение со знаком	R1:R0 <— (RdxRr) << 1	2
FMULS	Rd, Rr	Дробное умножение со знаком	R1:R0 <— (RdxRr) << 1	2
FMULSU	Rd, Rr	Дробное умножение знакового с беззнаковым числом	R1:R0 <— (RdxRr) << 1	2
Инструкции пересылки данных				
MOV	Rd, Rr	Запись из регистра в регистр	Rd <— Rr	1
MOVW	Rd, Rr	Перезапись слова между регистрами	Rd+1:Rd <— Rr+1:Rr	1
LDI	Rd, K	Запись константы в регистр	Rd <— K	1
LD	Rd, X	Косвенное считывание из памяти в регистр	Rd <— (X)	2
LD	Rd, X+	Косвенное считывание из памяти в регистр и инкр.	Rd <— (X), X <— X + 1	2
LD	Rd,-X	Предварительный декремент, а затем косвенное считывание из памяти в регистр	X <— X - 1, Rd <— (X)	2
LD	Rd,Y	Косвенное считывание из памяти в регистр	Rd <— (Y)	2
LD	Rd, Y+	Косвенное считывание из памяти в регистр и инкр.	Rd <— (Y), Y <— Y + 1	2
LD	Rd,-Y	Предварительный декремент, а затем косвенное считывание из памяти в регистр	Y <— Y - 1, Rd <— (Y)	2
LDD	Rd,Y+q	Косвенное считывание из памяти в регистр со смещением	Rd <— (Y + q)	2
LD	Rd, Z	Косвенное считывание из памяти в регистр	Rd <— (Z)	2
LD	Rd, Z+	Косвенное считывание из памяти в регистр и инкр.	Rd <— (Z),Z <— Z+1	2
LD	Rd.-Z	Предварительный декремент, а затем косвенное считывание из памяти в регистр	Z <— Z - 1, Rd <— (Z)	2
LDD	Rd, Z+q	Косвенное считывание из	Rd <— (Z + q)	2

		памяти в регистр со смещением		
LDS	Rd, k	Непосредственное чтение из ОЗУ в регистр	Rd <— (k)	2
ST	X, Rr	Косвенная запись	(X) <— Rr	2
ST	X+, Rr	Косвенная запись и послед. инкремент	(X) <— Rr, X <— X + 1	2
ST	-X, Rr	Предв. декремент и косвенная запись	X <— X - 1, (X) <— Rr	2
ST	Y, Rr	Косвенная запись	(Y) <— Rr	2
ST	Y+, Rr	Косвенная запись и послед. инкремент	(Y) <— Rr, Y <— Y + 1	2
ST	-Y, Rr	Предв. декремент и косвенная запись	Y <— Y - 1, (Y) <— Rr	2
STD	Y+q,Rr	Косвенная запись со смещением	(Y + q) <— Rr	2
ST	Z, Rr	Косвенная запись	(Z) <— Rr	2
ST	Z+, Rr	Косвенная запись и послед. инкремент	(Z) <— Rr, Z <— Z + 1	2
ST	-Z. Rr	Предв. декремент и косвенная запись	Z <— Z - 1, (Z) <— Rr	2
STD	Z+q,Rr	Косвенная запись со смещением	(Z + q) <— Rr	2
STS	k, Rr	Непосредственная запись в ОЗУ	(k) <— Rr	2
LPM		Чтение из памяти программ	R0 <— (Z)	3
LPM	Rd, Z	Чтение из памяти программ	Rd <— (Z)	3
LPM	Rd, Z+	Чтение из памяти программ и последующий инкремент	Rd <— (Z), Z <— Z+1	3
ELPM		Расширенное чтение из памяти программ	R0 <— (RAMPZ:Z)	3
ELPM	Rd, Z	Расширенное чтение из памяти программ	Rd <— (RAMPZ:Z)	3
ELPM	Rd, Z+	Расширенное чтение из памяти программ и последующие инкремент	Rd <— (RAMPZ:Z), RAMPZ:Z <— RAMPZ:Z+1	3
SPM		Запись в память программ	(Z) <— R1 :R0	-
IN	Rd, P	Считывание из порта ввода-вывода в регистр	Rd <— P	1
OUT	P, Rr	Запись из регистра в порт ввода-вывода	P <— Rd	1
PUSH	Rr	Помещение содержимого регистра в стек	STACK <— Rr	2
POP	Rd	Извлечение из стека в регистр	Rd <— STACK	2

Инструкции перехода				
RJMP	A	Относительный переход	PC <— PC + A +1	2
IJMP		Косвенный переход по указателю (Z)	PC <— Z	2
JMP	A	Безусловный переход	PC <— A	3
RCALL	A	Относительный вызов процедуры	PC <— PC + A + 1	3
ICALL		Косвенный вызов процедуры по указателю (Z)	PC <— Z	3
CALL	A	Безусловный вызов процедуры	PC <— A	4
RET		Возврат из подпрограммы	PC <— STACK	4
RETI		Возврат из прерывания	PC <— STACK	4
CPSE	Rd,Rr	Сравнение и пропуск, если равно if (Rd = Rr)	PC <— PC + 2 или 3	1/2/3
CP	Rd,Rr	Сравнение	Rd-Rr	1
CPC	Rd,Rr	Сравнение с учетом переноса	Rd - Rr-C	1
CPI	Rd,K	Сравнение регистра с константой	Rd-K	1
SBRC	Rr,b	Пропуск, если бит в регистре сброшен	if (Rr(b)=O)PC <— PC + 2 или 3	1 /2/3
SBRS	Rr, b	Пропуск, если бит в регистре установлен	if (Rr(b)=1)PC <— PC+ 2 или 3	1/2/3
SBIC	P, b	Пропуск, если бит в регистре ввода-вывода сброшен	if (P(b)=O)PC <— PC + 2 или 3	1 /2/3
SBIS	P, b	Пропуск, если бит в регистре ввода-вывода установлен	if (P(b)=1)PC <— PC + 2 или 3	1 /2/3
BRBS	s, K	Переход, если флаг состояния установлен	if (SREG(s) = 1) then PC <— PC+ K + 1	1/2
BRBC	s, A	Переход, если флаг состояния сброшен	if (SREG(s) = 0) then PC <— PC+ A + 1	1 /2
BREQ	A	Переход, если равно	if (Z = 1) then PC <— PC + A + 1	1 /2
BRNE	A	Переход, если не равно	if (Z = 0) then PC <— PC + A + 1	1 /2
BRCS	A	Переход, если перенос установлен	if (C = 1)then PC <— PC + A + 1	1 /2
BRCC	A	Переход, если перенос сброшен	if (C = 0) then PC <— PC + A + 1	1 /2
BRSH	A	Переход, если больше или равно	if (C = 0) then PC <— PC + A + 1	1 /2
BRLO	A	Переход, если меньше	if (C = 1) then PC <— PC + A + 1	1 /2
BRMI	A	Переход, если минус	if (N = 1)then PC <— PC + A + 1	1 /2

BRPL	A	Переход, если плюс	if (N = 0) then PC <— PC + A + 1	1 /2
BRGE	A	Переход, если больше или равно с учетом знака	if (N e V= 0) then PC <— PC + A + 1	1 /2
BRLT	A	Переход, если меньше нуля с учетом знака	if (N e V= 1) then PC <— PC + A + 1	1 /2
BRHS	A	Переход, если флаг H установлен	if (H = 1)then PC <— PC + A + 1	1 /2
BRHC	A	Переход, если флаг H сброшен	if (H = 0) then PC <— PC + A + 1	1 /2
BRTS	A	Переход, если флаг T установлен	if (T = 1)then PC <— PC + A +1	1 /2
BRTC	A	Переход, если флаг T сброшен	if (T = 0) then PC <— PC + A + 1	1 /2
BRVS	A	Переход, если флаг V установлен	if (V = 1)then PC <— PC + A + 1	1 /2
BRVC	A	Переход, если флаг V сброшен	if (V = 0) then PC <— PC + A + 1	1 /2
BRIE	A	Переход, если прерывания разрешены	if (I = 1)then PC <— PC + A + 1	1 /2
BRID	A	Переход, если прерывания запрещены	if (I = 0) then PC <— PC + A + 1	1 /2
Команды операций с битами				
SBI	P,b	Установка бита в регистре ввода-вывода	l/O(P,b) <— 1	2
CBI	P,b	Сброс бита в регистре ввода-вывода	l/O(P,b) <— 0	2
LSL	Rd	Логический сдвиг влево	Rd(n+1) <— Rd(n), Rd(0) <— 0	1
LSR	Rd	Логический сдвиг вправо	Rd(n) <— Rd(n+1), Rd(7) <— 0	1
ROL	Rd	Вращение влево через перенос	Rd(0)<— C, Rd(n+1)<— Rd(n), C <— Rd(7)	1
ROR	Rd	Вращение вправо через перенос	Rd(7) <— C,Rd(n)<— Rd(n+1),C <— Rd(0)	1
ASR	Rd	Арифметический сдвиг вправо	Rd(n)<— Rd(n+1), n=0..6	1
SWAP	Rd	Обмен тетрадами	Rd(3..0) <— Rd(7..4), Rd(7..4) <— Rd(3..0)	1
BSET	s	Установка флага регистра SREG	SREG(s) <— 1	1
BCLR	s	Сброс флага регистра SREG	SREG(s) <— 0	1
BST	Rr, b	Запись бита регистра в T	T <— Rr(b)	1

BLD	Rd, b	Чтение из T в бит регистра	Rd(b) <— T	1
SEC		Установка переноса	C <— 1	1
CLC		Сброс переноса	C <— 0	1
SEN		Установка флага N	N <— 1	1
CLN		Сброс флага N	N <— 0	1
SEZ		Установка флага нуля Z	Z <— 1	1
CLZ		Сброс флага нуля Z	Z <— 0	1
SEI		Общее разрешение прерываний	l<— 1	1
CLI		Общий запрет прерываний	l <— 0	1
SES		Установка флага S	S <— 1	1
CLS		Сброс флага S	S <— 0	1
SEV		Установка флага V в регистре SREG	V <— 1	1
CLV		Сброс флага V в регистре SREG	V <— 0	1
SET		Установка флага T в регистре SREG	T <— 1	1
CLT		Сброс флага T в регистре SREG	T <— 0	1
SEH		Установка флага H в регистре SREG	H <— 1	1
CLH		Сброс флага H в регистре SREG	H <— 0	1
Инструкции управления микроконтроллером				
NOP		Нет операции		1
SLEEP		Перевод в режим сна	(см. подробное описание режима сна)	1
WDR		Сброс сторожевого таймера	(см. подробное описание сторожевого таймера)	1
BREAK		Прерывание	Только для встроенной отладки	-

ПРИЛОЖЕНИЕ В

(справочное)

Регистры ввода/вывода

Таблица В.1 – Адреса и описание регистров ввода/вывода AtMega16A

Адрес РВВ	Имя	Функция регистра
0x00(0x20)	TWBR	Регистр скорости передачи данных TWI
0x01(0x21)	TWSR	Регистр статуса TWI
0x02(0x22)	TWAR	Регистр адреса TWI
0x03(0x23)	TWDR	Регистр смещения адреса/данных TWI
0x04(0x24)	ADCL	Регистр данных АЦП (младший байт)
0x05(0x25)	ADCH	Регистр данных АЦП (старший байт)
0x06(0x26)	ADCSR	Регистр управления и состояния АЦП
0x07(0x27)	ADMUX	Регистр управления мультиплексором АЦП
0x08{0x28)	ACSR	Регистр управления и статуса аналогового компаратора
0x09(0x29)	UBRR	Регистр управления скоростью порта UART
0x0A(0x3A)	UCSRA	Регистр управления и состояния UART младший байт
0x0B(0x3B)	UCSRB	Регистр управления и состояния UART старший байт
0x0C(0x2C)	UDR	Регистр данных порта UART
0x0D(0x2D)	SPCR	Регистр управления SPI
0x0E(0x2E)	SPSR	Регистр состояния SPI
0x0F(0x2F)	CPDR	Регистр данных SPI
0x10(0x30)	PIND	Выводы входов порта D
0x11(0x31)	DDRD	Регистр направления данных порта D
0x12(0x32)	PORTD	Регистр данных порта D
0x13(0x33)	PINC	Выводы входов порта C
0x14(0x34)	DDRC	Регистр направления данных порта C
0x15(0x35)	PORTC	Регистр данных порта C
0x16(0x36)	PINB	Выводы входов порта B
0x17(0x37)	DDRB	Регистр направления данных порта B
0x18(0x38)	PORTB	Регистр данных порта B
0x19(0x39)	PINA	Выводы входов порта A
0x1A(0x3A)	DDRA	Регистр направления порта A
0x1B(0x3B)	PORTA	Регистр данных порта A
0x1C(0x3C)	EECR	Регистр управления EEPROM
0x1D(0x3D)	EEDR	Регистр данных EEPROM
0x1E(0x3E)	EEARL	Младший байт регистра адреса EEPROM
0x1F(0x3F)	EEARH	Старший байт регистра адреса EEPROM
0x20 (0x40)	UCSRC/ UBRRH	Скорость передачи UART
0x21 (0x41)	WDTCR	Регистр управления сторожевого таймера

0x22(0x42)	ASSR	Регистр состояния асинхронного режима
0x23(0x43)	OCR2	Регистр совпадения выхода таймера/счетчика T2
0x24(0x44)	TCNT2	Таймер T2
0x25(0x45)	TCCR2	Регистр управления таймером T2
0x26(0x46)	ICR1L	Регистр захвата таймера/счетчика T1(младший байт)
0x27(0x47)	ICR1H	Регистр захвата таймера/счетчика T1(старший байт)
0x28(0x48)	OCR1BL	Регистр совпадения выхода B (младший байт)
0x29(0x49)	OCR1BH	Регистр совпадения выхода B (старший байт)
0x2A(0x4A)	OCR1AL	Младший байт регистра A сравнения выхода таймера/счетчика T1
0x2B(0x4B)	OCR1AH	Старший байт регистра A сравнения выхода таймера/счетчика T1
0x2C(0x4C)	TCNT1L	Младший байт таймера/счетчика T1
0x2D(0x4D)	TCNT1H	Старший байт таймера/счетчика T1
0x2E(0x4E)	TCCR1B	Регистр управления B таймера/счетчика T1
0x2F(0x4F)	TCCR1A	Регистр управления A таймера/счетчика T1
0x30(0x50)	SFIOR	Регистр специальных функций
0x31(0x51)	OCDR/ OSCCAL	Регистр встроенного отладчика
0x32(0x52)	TCNT0	Таймер/счетчик T0
0x33(0x53)	TCCR0	Регистр управления таймером/счетчиком T0
0x34(0x54)	MCUSR	Регистр состояния микроконтроллера
0x35(0x55)	MCUCR	Общий регистр управления микроконтроллером
0x36(0x55)	TWCR	Регистр управления TWI
0x37(0x55)	SPMCR	Регистр управления при обмене с памятью программ
0x38(0x58)	TIFR	Регистр флагов прерываний от таймеров/счетчиков
0x39(0x59)	TIMSK	Регистр маски прерываний от таймеров/счетчиков
0x3A(0x5A)	GIFR	Общий регистр флагов прерываний
0x3B(0x5B)	GIMSK	Общий регистр маски прерываний
0x3D(0x5D)	SPL	Младший байт указателя стека
0x3E(0x5E)	SPH	Старший байт указателя стека
0x3F(0x5F)	SREG	Регистр состояния (статуса) процессора

ПРИЛОЖЕНИЕ С

(справочное)

Назначение выводов

В таблицах приведены назначение и описание альтернативных функций выводов микроконтроллера ATMega16A. Номера контактов приведены для корпуса PDIP-40.

Таблица С.1 – Выводы обеспечения работы МК

Обозначение	№	Назначение
VCC	10	Напряжение питания цифровых элементов
GND	11	Общий
RESET	9	Вход сброса. Если на этот вход приложить низкий уровень длительностью более минимально необходимой будет генерирован сброс независимо от работы синхронизации.
XTAL1	13	Вход инвертирующего усилителя генератора и вход внешней синхронизации.
XTAL2	12	Выход инвертирующего усилителя генератора
AVCC	30	Вход питания АЦП
AGND	32	Вход «земли» АЦП
AREF	31	Вход подключения источника опорного напряжения АЦП.

Таблица С.2 – Альтернативные функции порта А

Вывод порта	№	Альтернативная функция
PA7(ADC7)	40	Вход канала 7 АЦП
PA6(ADC6)	39	Вход канала 6 АЦП
PA5(ADC5)	38	Вход канала 5 АЦП
PA4(ADC4)	37	Вход канала 4 АЦП
PA3(ADC3)	36	Вход канала 3 АЦП
PA2(ADC2)	35	Вход канала 2 АЦП
PA1(ADC1)	34	Вход канала 1 АЦП
PA0(ADC0)	33	Вход канала 0 АЦП

Таблица С.3 – Альтернативные функции порта В

Вывод порта	№	Альтернативная функция
PB7(SCK)	8	Синхронизация последовательной связи шины SPI
PB6(MISO)	7	Ввод для ведущей/вывод для подчиненной шины SPI

PB5(MOSI)	6	Вывод для ведущей/ввод для подчиненной шины SPI
PB4(SS)	5	Вход выбора подчиненного режима интерфейса SPI
PB3(AIN1/OC0)	4	Отрицательный вход компаратора или выход ШИМ таймера-счетчика 0
PB2(AIN0/INT2)	3	Положительный вход компаратора или вход внешнего прерывания 2
PB1(T1)	2	Вход синхронизации таймера-счетчика 1
PB0(XCK/T0)	1	Вход/выход внешней синхронизации USART или вход синхронизации таймера-счетчика 0

Таблица С.4 – Альтернативные функции порта С

Вывод порта	№	Альтернативная функция
PC7(TOSC2)	29	Вход инвертирующего усилителя генератора часов реального времени
PC6(TOSC1)	28	Выход инвертирующего усилителя генератора часов реального времени
PC5(TDI)	27	Ввод данных при JTAG тестировании
PC4(TDO)	26	Вывод данных при JTAG тестировании
PC3(TMS)	25	Выбор режима JTAG тестирования
PC2(TCK)	24	Синхронизация JTAG тестирования
PC1(SDA)	23	Ввод/вывод последовательных данных TWI
PC0(SCL)	22	Синхронизация последовательной связи TWI

Таблица С.5 – Альтернативные функции порта D

Вывод порта	№	Альтернативная функция
PD7(OC2)	21	Выход ШИМ таймера-счетчика 2
PD6(ICP1)	20	Вход захвата таймера-счетчика1
PD5(OC1A)	19	Выход ШИМ таймера-счетчика 1 канал А
PD4(OC1B)	18	Выход ШИМ таймера-счетчика 1 канал В
PD3(INT1)	17	Вход внешнего прерывания 1
PD2(INT0)	16	Вход внешнего прерывания 0
PD1(TXD)	15	Вывод передачи данных USART
PD0(RXD)	14	Вход приема данных USART

ПРИЛОЖЕНИЕ D

(справочное)

Перечень принятых обозначений и сокращений

A	-	адрес при работе с памятью;
b	-	номер бита в операциях с регистрами ввода-вывода;
K	-	8-ми разрядная константа в операциях со "старшими" регистрами общего назначения (R16-R31);
P	-	регистр ввода-вывода. Это порты, таймеры и т.д.;
Rd	-	регистр - приемник, место, куда сохраняется результат выполнения команды;
Rr	-	регистр - источник в двухоперандных командах. Его значение после выполнения команды не изменяется;
X	-	регистровая пара, состоящая из регистров XL (R26) и XH (R27);
Y	-	регистровая пара, состоящая из регистров YL (R28) и YH (R29);
Z	-	регистровая пара, состоящая из регистров ZL (R30) и ZH (R31);
АЛУ	-	арифметико-логическое устройство;
АЦП	-	аналогово-цифровой преобразователь;
ВУ	-	внешнее устройство;
ИОН	-	источник опорного напряжения;
МК	-	микроконтроллер;
РВВ	-	регистр ввода/вывода;

РОН	-	регистр общего назначения;
УВХ	-	устройство выборки и хранения;
УГО	-	условно-графическое обозначение;
ЦАП	-	цифро-аналоговый преобразователь
ЦПУ	-	центральное процессорное устройство;
ШИМ	-	широтно-импульсная модуляция;

ПРИЛОЖЕНИЕ Е

(справочное)

Список используемой литературы