МИКРОПРОЦЕССОРНЫЕ СИСТЕМЫ УПРАВЛЕНИЯ ЭЛЕКТРОПРИВОДОВ

Методические указания
к проведению лабораторных работ

Челябинск
2018

Хусаинов Р.З., Качалов А.В. Микропроцессорные системы управления электроприводов: Методические указания к выполнению лабораторных работ. – Челябинск, Учтех-Профи, 2018.– 77 с.

Методические указания предназначены для студентов средних и высших учебных заведений, изучающих дисциплины по архитектуре и программированию микропроцессорных систем. Методические указания также могут быть использованы для обучения учащихся профессионально-технических училищ и слушателей отраслевых учебных центров повышения квалификации инженерно-технических работников.

V.1-2018

ОГЛАВЛЕНИЕ

1. ОПИСАНИЕ ЛАБОРАТОРНОГО СТЕНДА И ПОРЯДОК ПРОВЕДЕНИЯ ЛАБОРАТОРНЫХ РАБОТ

Описание лабораторного стенда

Модуль «Микроконтроллер» предназначен для программирования и изучения функций микроконтроллера ATmega8535 семейства AVR, выпускаемого фирмой Atmel. Внешний вид модуля приведен на рис. 1.

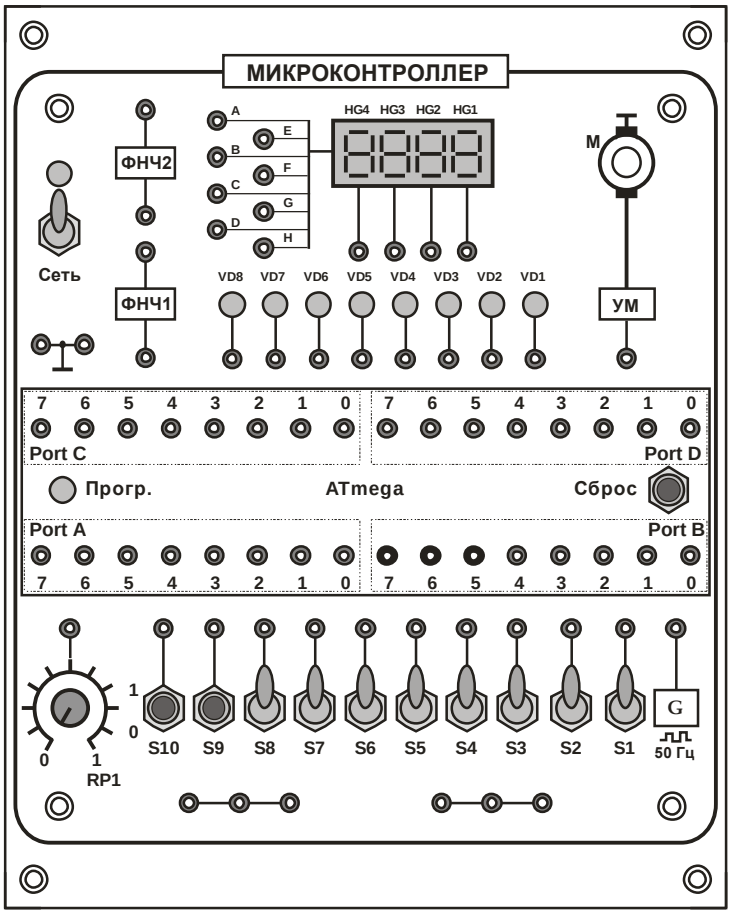


Рис. 1. Внешний вид модуля «Микроконтроллер»

На лицевой панели модуля расположены:

– переключатель «Сеть» со светодиодом индикации наличия напряжения. Переключатель осуществляет коммутацию напряжения, подаваемого на модуль;

– мнемосхему микроконтроллера с клеммами, связанными с портами ввода/вывода микроконтроллера;

– переключатели S1-S8 с выходными клеммами для подачи логических сигналов на микроконтроллер;

– кнопки S9, S10 с выходными клеммами для подачи логических сигналов на микроконтроллер;

– потенциометр RP1 с выходной клеммой для подачи аналогового напряжения на микроконтроллер;

– мнемосхема генератора низкочастотного прямоугольного сигнала 50 Гц и клемма выхода генератора;

– светодиоды VD1 – VD8 с клеммами для их подключения к источнику напряжения (например, к микроконтроллеру);

– электродвигатель постоянного тока М с усилителем мощности и клеммой для подачи на него управляющего напряжения;

– семисегментный четырехсимвольный светодиодный индикатор с клеммами подачи напряжения на сегменты A, B, C, D, E, F, G, H, а также на общую точку каждого сегмента индикатора;

– два фильтра низкой частоты для фильтрации ШИМ-сигналов на выходе микроконтроллера.

Табл. 1. Краткая характеристика микроконтроллера ATmega8535

Параметр	**Значение**
Частота установленного кварцевого резонатора	8 МГц
Напряжение электропитания	2,7 – 5,5 В
Объем внутренней Flash – памяти	8 кБайт
Объем энергонезависимой памяти	512 Байт
Объем внутренней ОЗУ	512 Байт
32 программируемых входа/выхода	32 на 4 портах
JTAG – интерфейс	нет
8-битные таймеры/счетчики с ШИМ	2 шт.
16-битный таймер/счетчик с ШИМ	1 шт.
10-разрядный аналогово-цифровой преобразователь	есть
Количество каналов АЦП	8
Аналоговый компаратор	есть
Источники внешних прерываний	3 шт.
Универсальный приемопередатчик USART	есть
SPI – интерфейс	есть
TWI – интерфейс	есть

Порядок выполнения лабораторных работ

При подготовке к лабораторной работе необходимо:

– ознакомиться с ее содержанием и, пользуясь рекомендованной литературой и лекциями, изучить теоретические положения, на которых базируется работа;

– в соответсвии со своим вариантом задания написать листинг программы;

– выполнить проверку рабоспособности программы на симуляторе AVR-studio;

– ответить на контрольные вопросы к лабораторной работе.

Перед выполнением лабораторной работы необходимо:

– представить отчет по предыдущей работе;

– представить листинг программы и необходимые расчет по своему варианту задания к выполняемой лабораторной работе;

– ответить на вопросы, задаваемые преподавателем.

При выполнении лабораторной работы необходимо:

– ввести программу в компьютер и показать ее работоспособность преподавателю на AVR-studio;

– произвести сборку схемы;

– *только после разрешения* преподавателя включить питание и приступить к программированию микроконтроллера и проверки его работы;

– представить программу на мирокконтроллере на проверку преподавателю;

– по окончании работы привести в порядок рабочее место.

Оформление отчетов по лабораторным работам

Все отчеты должны быть выполнены и сданы на проверку каждым студентом *индивидуально*. Работа считается сданной, если она проверена, не содержит ошибок и принята преподавателем.

Отчет помимо правильно оформленного титульного листа с указанием номера лабораторной работы, ее названия, фамилии и инициалов студента, выполнившего работу, номера группы и фамилии и инициалов преподавателя должен содержать (порядок оформления пунктов также должен соблюдаться):

1. **Цель работы**.

2. **Функциональная схема устройства**. Указываются все используемые входы/выходы микроконтроллера, периферийные элементы (тумблеры или потенциометры для подачи дискретных и аналоговых входных сигналов, резисторы, светодиоды, семисегментные индикаторы и т.п. выводимых данных), подключение питания микроконтроллера, подключение кварцевого генератора.

3. **Предварительные расчеты** (если они требуются). Обычно эти расчеты включают данные, требуемые для выполнения программы: выбор необходимых прерываний, расчеты периодов дискретизации таймеров и АЦП, разрядность АЦП и т.д).

4. **Листинг программы**. Листинг необходимо приводить обязательно с комментариями по основным элементам программы: пояснения по переменным, назначение группы инструкций в программе (стек, инициализация портов, инициализация таймера T0 и т.д.).

5. **Дисассемблер программы**. Дисассемблер должен приводиться полностью для всей программы, включая таблицу векторов прерываний и память данных во FLASH-области.

6. **Стек** (если он используется). Информацию по стеку во время исполнения программы с указанием: вершины стека, информации, которая записывается в стек и необходимыми пояснениями.

7. Другие необходимые пункты в соответствии с требованиями к лабораторной работе.

8. **Выводы по работе**.

Схемы и таблицы должны быть пронумерованы и аккуратно построены.

2. ЛАБОРАТОРНЫЕ РАБОТЫ

Работа № 7. Микропроцессорное управление скоростью шагового двигателя с использованием шестнадцатиразрядного таймера Т1

Цель работы

Освоить теоретический и практический материал по работе 16-разрядного таймера Т1 микроконтроллера Atmega8535/ATmega32 в режиме подсчета временных интервалов. Применить приобретенные навыки при написании программы управления шаговым двигателем.

Программа работы

1. Изучить необходимый теоретический материал о регистрах и функционировании таймера Т1.
2. Разобраться в программах по использованию таймера Т1, представленной в лабораторной работе.
3. Написать и отладить собственную программу управления шаговым двигателем с использованием таймера Т1 в соответствии с вариантом.

Пояснения к работе

1. Назначение и регистры таймера Т1

Помимо двух восьмиразрядных таймеров/счетчиков Т0 и Т2, микроконтроллер AtmegaXX содержит двухканальный шестнадцатиразрядный таймер/счетчик Т1.

Также как и восьмиразрядные таймеры Т1 служит для подсчета временных интервалов, регистрации внешних событий и работы в режиме ШИМ для вывода цифрового периодического сигнала с регулируемой скважностью и частотой. Таймер синхронизируется от источника тактового сигнала процессора, либо от источника внешнего сигнала, подаваемого на цифровой вход микроконтроллера.

Основным отличием таймера Т1 от таймеров Т0 и Т2 является его 16-разрядная организация, поэтому основные регистры таймера состоят из двух частей – старшей и младшей, обозначаемых, соответственно, буквами H и L. Так, например, регистр счета **TCNT1** таймера Т1 состоит из двух 8-разрядных регистров **TCNT1H** и **TCNT1L**.

В табл. 1 приведены регистры таймера Т1, управляющие его работой.

Табл.1. Регистры таймера Т1

№	Назавание регистра	Разряд-ность, бит	Обозначение регистра	Старший байт	Младший байт
1	Счетный регистр	16	TCNT1	TCNT1H	TCNT1L
2	Регистр сравнения канала А	16	OCR1A	OCR1AH	OCR1AL
3	Регистр сравнения канала В	16	OCR1B	OCR1BH	OCR1BL
4	Регистр захвата	16	ICR1	ICR1H	ICR1L
5	Регистры управления А	8	TCCR1A	–	–
6	Регистры управления В	8	TCCR1B	–	–

2. Работа таймера T1

Рассмотрим работу таймера при его работе от источника тактового сигнала процессора (рис. 1), в качестве которого в лабораторном стенде выступает кварцевый резонатор с тактовой частотой f_{CLK} = 8 МГц.

Счет импульсов источника частоты ведется в 16-разрядном счетном регистре таймера **TCNT1=TCNT1H:TCNT1L**. Перед тем, как попасть в счетный регистр, тактовый сигнал поступает на схему деления частоты, которая, в соответствии с параметрами, установленными при инициализации таймера, производит деление частоты тактового сигнала f_{CLK} на коэффициент предделителя $К_{ПД}$ в следующем соотношении:

– без делителя частоты тактового сигнала $К_{ПД}$ =1;

– с делителем частоты тактового сигнала $К_{ПД}$ =8;

– с делителем частоты тактового сигнала $К_{ПД}$ =64;

– с делителем частоты тактового сигнала $К_{ПД}$ =256;

– с делителем частоты тактового сигнала $К_{ПД}$ =1024.

Каждый импульс с предделителя частоты считается и инкрементирует значение, содержащееся в счетном регистре **TCNT1** (рис. 1).

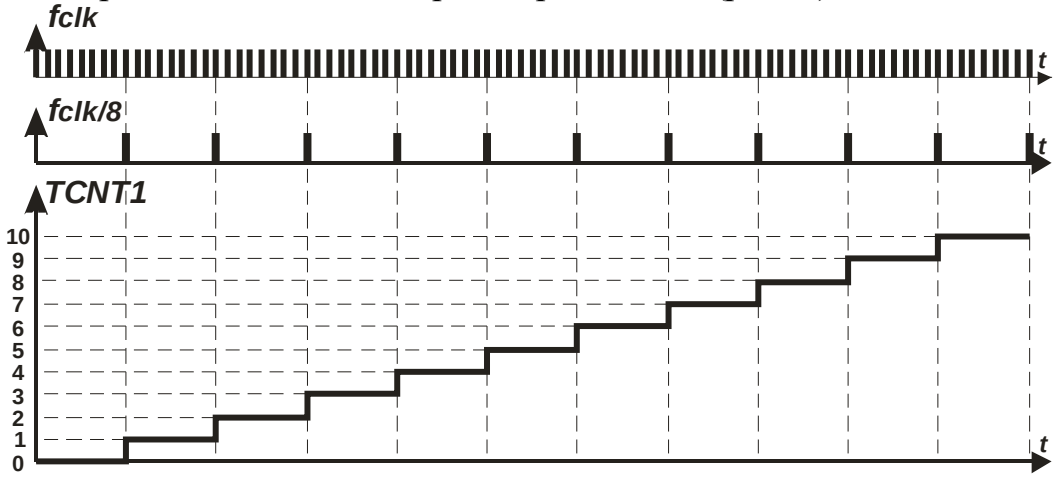


Рис. 1. Счетный регистр таймера T1 при коэффициенте предделителя $К_{ПД}$ =8

Когда счетный регистр **TCNT1** переполняется, т.е. при максимальном значении **TCNT1 =** 65535 на его вход приходит очередной импульс и регистр сбрасывается в нулевое состояние, формируется флаг прерывания по переполнению таймера **TOV1** (Timer Overflow Flag) (рис. 2).

Кроме события «переполнение таймера», таймер T1 позволяет реализовать события «совпадение таймера». В этом случае в таймер вводится регистр сравнения и его значение постоянно сравнивается со значением счетного регистра – при их равенстве в микроконтроллере возникает событие «совпадение». В отличие от восьмиразрядных таймеров, таймер T1 имеет два канала сравнения и соответственно позволяет использовать 2 независимых события, называемых «совпадение канала А» и «совпадение канала В». Для этого в таймер введены два 16-ти разрядных регистра сравнения:

– канал А: **OCR1A = OCR1AH : OCR1AL;**

– канал В: **OCR1B = OCR1BH : OCR1BL**.

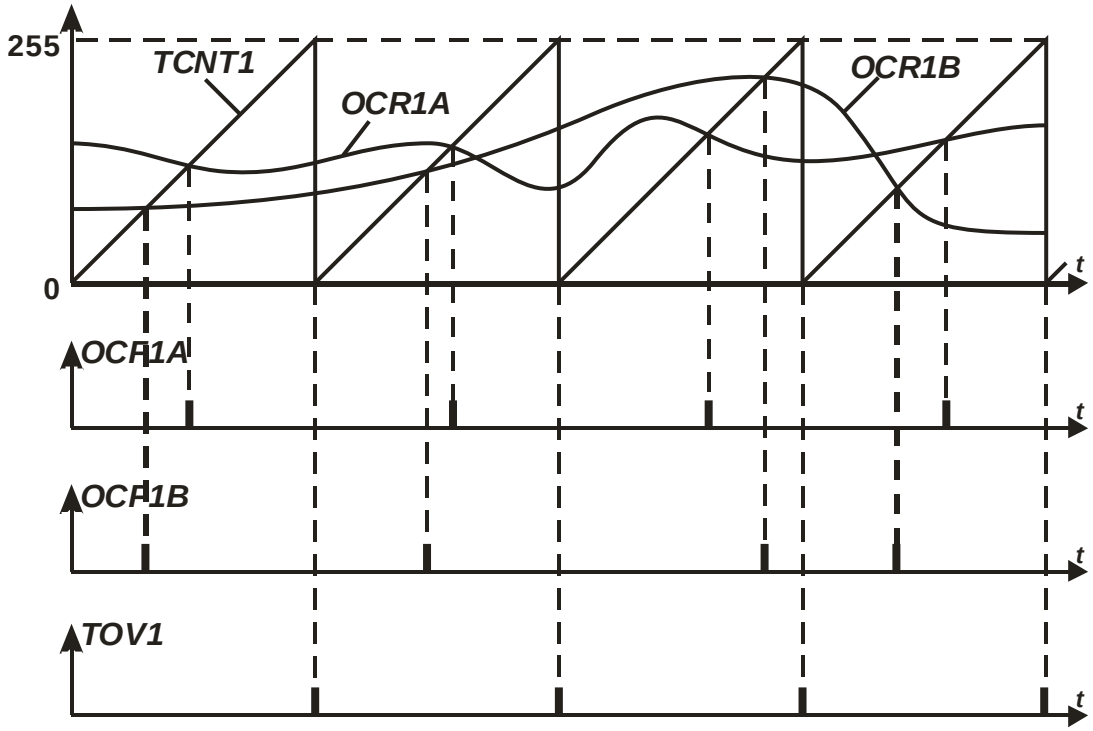


Рис. 2. Принцип работы таймера Т1 и формирование прерываний

В эти регистры записываются необходимые уставки срабатывания, то есть числа в диапазоне от 0 до 65535 (т.е. 2^{16}–1), при достижении которых счетным регистром **TCNT1** формируются флаги прерываний по совпадению канала А **OCF1A** и совпадению канала В **OCF1B** таймера Т1 (см. рис. 2).

3. Регистр управления TCCR1A таймера Т1

Управление таймером Т1 осуществляется через два 8-разрядных регистра управления **TCCR1A**, **TCCR1B**.

Регистр управления **TCCR1A** содержит биты, приведенные в табл. 2.

Табл. 2. Регистр управления TCCR1A таймера Т1

Бит	7	6	5	4	3	2	1	0
Название	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10

Биты 7 и 6 – COM1A1 и COM1A0. Микросхема контроллера Atmega 8535 содержит вывод OC1A таймера Т1 в качестве альтернативной функции вывода PD5 порта ввода/вывода D. Таймер Т1 может управлять этим выводом в соответствии с выбранным режимом работы, определяемым комбинацией бит COM1A1:COM1A0 (табл. 3).

Табл. 3. Вывод канала А таймера Т1 в режиме подсчета временных интервалов

COM1A1	**COM1A0**	**Пояснение**
0	0	Вывод OC1A отключен
0	1	Изменение OC1A на противоположное при совпадении канала А
1	0	Очистка OC1A при совпадении канала А
1	1	Установка OC1A при совпадении канала А

Биты 5 и 4 – COM1B1 и COM1B0. Микроконтроллер Atmega 8535 содержит вывод OC1B таймера Т1 в качестве альтернативной функции вывода PD4 порта ввода/вывода D. Таймер Т1 может управлять этим выводом в

соответствии с выбранным режимом работы, определяемым комбинацией бит COM1B1:COM1B0 (табл. 4).

Табл. 4. Вывод канала B таймера T1 в режиме подсчета временных интервалов

COM1B1	COM1B0	Пояснение
0	0	Вывод OC1B отключен
0	1	Изменение OC1B на противоположное при совпадении канала B
1	0	Очистка OC1B при совпадении канала B
1	1	Установка OC1B при совпадении канала B

Биты 3 и 2 – FOC1A и FOC1B. Эти биты действуют только в обычном режиме работы при подсчете временных интервалов. При записи в них логической «1» происходит срабатывание прерывания по совпадению соответствующего канала таймера и изменение выводов таймера OC1A и OC1B в соответствии с настройками бит COM1A1:COM1A0, COM1B1:COM1B0.

Биты 1 и 0 – WGM11, WGM10. Эти биты определяют режим работы таймера вместе с управляющими битами регистра TCCR1B.

4. Регистр управления TCCR1B таймера T1

Регистр управления TCCR1B содержит биты, приведенные в табл. 5.

Табл. 5. Регистр управления TCCR1B таймера T1

Бит	7	6	5	4	3	2	1	0
Название	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10

Бит 7 – ICNC1. Этот бит включает функцию фильтрации внешнего сигнала, поступающего на вывод PB1 микроконтроллера, когда таймер используется в качестве счетчика. Установка бита увеличивает надежность работы таймера. При активации этого бита между подачей сигнала на вывод PB1 и срабатыванием схемы счета проходит 4 периода тактовой частоты.

Бит 6 – ICES1. Этот бит определяет активный фронт сигнала на входе PB1. При ICES1=0 таймер срабатывает по спадающему фронту сигнала, иначе – по нарастающему фронту.

Бит 5. Зарезервирован и не используется.

Биты 4 и 3 – WGM13, WGM12. Эти биты вместе с битами WGM11, WGM10 регистра TCCR1A определяют режим работы таймера (табл. 6).

Табл. 6. Режимы работы таймера T1

Режим	WGM13	WGM12	WGM11	WGM10	Режим работы	Вершина	Обновление OCR1x	Переполнение TOV1
0	0	0	0	0	Нормальный режим	0xFFFF	Немедленно	Максимум
1	0	0	0	1	Фазовый ШИМ, 8 бит	0x00FF	На вершине	Минимум
2	0	0	1	0	Фазовый ШИМ, 9 бит	0x01FF	На вершине	Минимум
3	0	0	1	1	Фазовый ШИМ, 10 бит	0x3FFF	На вершине	Минимум
4	0	1	0	0	Очистка при совпадении	OCR1A	Немедленно	Максимум
5	0	1	0	1	Быстрый ШИМ, 8 бит	0x00FF	На вершине	На вершине
6	0	1	1	0	Быстрый ШИМ, 9 бит	0x01FF	На вершине	На вершине
7	0	1	1	1	Быстрый ШИМ, 10 бит	0x03FF	На вершине	На вершине
8	1	0	0	0	Фазовый и частотный ШИМ	ICR1	Внизу	Минимум
9	1	0	0	1	Фазовый и частотный ШИМ	OCR1A	Внизу	Минимум

10	1	0	1	0	Фазовый ШИМ	ICR1	На вершине	Минимум
11	1	0	1	1	Фазовый ШИМ	OCR1A	На вершине	Минимум
12	1	1	0	0	Очистка при совпадении	ICR1	Немедленно	Максимум
13	1	1	0	1	–	–	–	–
14	1	1	1	0	Быстрый ШИМ	ICR1	На вершине	На вершине
15	1	1	1	1	Быстрый ШИМ	OCR1A	На вершине	На вершине

Биты 2…0 – CS12…CS10. Эти биты определяют источник задания частоты таймера T1 и предделитель таймера (табл. 7).

Табл. 7. Предделитель таймера T1

CS12	CS11	CS10	Пояснение
0	0	0	Нет источника. Таймер остановлен
0	0	1	Предделитель $К_{ПД}$ =1
0	1	0	Предделитель $К_{ПД}$ =8
0	1	1	Предделитель $К_{ПД}$ =64
1	0	0	Предделитель $К_{ПД}$ =256
1	0	1	Предделитель $К_{ПД}$ =1024
1	1	0	Счет импульсов по спадающему фронту внешнего сигнала на входе T1 (бит PB1)
1	1	1	Счет импульсов по нарастающему фронту внешнего сигнала на входе T1(бит PB1)

5. Регистры масок и флагов прерываний таймеров

Помимо управляющих регистров **TCCR1A** и **TCCR1B**, счетного регистра **TCNT1** и регистров сравнения **OCR1A**, **OCR1B**, в микроконтроллере содержатся регистры масок прерываний и флагов таймеров соответственно **TIMSK** и **TIFR**.

Назначение этих регистров следующее. Когда возникает переполнение или совпадение таймера T1, автоматически формируются соответствующие флаги прерываний в регистре **TIFR** – биты TOV1, OCF1B или OCF1A устанавливаются в единичное состояние. Если в регистре масок прерываний таймеров **TIMSK** на соответствующее прерывание наложена маска (т.е. бит установлен в единицу), то вызывается процедура обработки этого прерывания. Если же маска прерывания не наложена (бит установлен в ноль), то при совпадении или переполнении таймера ничего не происходит.

Поскольку регистр масок прерываний таймеров **TIMSK** управляет всеми таймерами микроконтроллера, то рассмотрим только биты, разрешающие работу прерываний таймера T1 – биты 2…5 (табл. 8):

Табл. 8. Регистр масок прерываний таймеров TIMSK

Бит	7	6	5	4	3	2	1	0
Название	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0

– бит 2 TOIE1 – разрешение прерывания по переполнению таймера T1;
– бит 3 OCIE1B – разрешение прерывания по совпадению канала B;
– бит 4 OCIE1A – разрешение прерывания по совпадению канала A;
– бит 5 TICIE1 – разрешение прерывания захвата таймера T1.

Флаги прерываний таймера T1 в регстре флагов **TIFR** (табл. 9):

Табл. 9. Регистр флагов прерываний таймеров TIFR

Бит	7	6	5	4	3	2	1	0
Название	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0

– бит 2 TOV1 – флаг прерывания по переполнению таймера T1.

– бит 3 OCF1B – флаг прерывания по совпадению канала B таймера T1;

– бит 4 OCF1A – флаг прерывания по совпадению канала A таймера T1;

– бит 5 ICF1 – флаг прерывания «захват» таймера T1.

Следует отметить, что использование в программе регистра флагов необязательно – для вызова обработчика прерывания можно использовать таблицу векторов, что и выполняется в большинстве программ с прерываниями таймеров.

6. Регистр захвата таймера T1

Таймер T1 содержит также шестнадцатиразрядный регистр захвата внешнего сигнала **ICR1 = ICR1H : ICR1L** (Input Capture Register). При использовании для счета внешнего источника сигнала на входе T1 (бит PB1) значение этого регистра будет обновляться по каждому заданному событию на входе T1. При этом в него будет записываться значение регистра счета **TCNT1**, содержащееся в данный момент в таймере. Этот режим полезен при подсчете длительности внешних сигналов.

Пример. Рассмотрим программу управления от контроллера скоростью шагового двигателя (ШД). Пусть требуется изменять скорость ШД, работающего в режиме униполярного несимметричного управления: при появлении сигнала «разрешение» (бит PA0), шаговый двигатель начинает вращение со скоростью 3 об/мин. При наличии сигнала «реверс» (бит PA1) ротор двигателя изменяет направление вращения с той же скоростью. Пусть 4 полуобмотки шагового двигателя A, B, /A, /B подключены к битам соответственно 0…3 порта B. Двигатель имеет стандартное разрешение по углу: N=200 имп/об, т.е. для несимметричного режима $N_{НЕСИМ}$=400 имп/об. Для получения импульсов заданной частоты необходимо использовать таймер T1 с прерыванием по совпадению, канал A.

Вначале выполним необходимые вычисления:

1. Рассчитаем требуемую частоту подачи импульсов на обмотки

$$f_{T1}=(3 \text{ об/мин} * 400 \text{ имп/об}) / 60 \text{ с} = 20 \text{ Гц}.$$

2. Рассчитаем параметры регистра сравнения таймера T1:

а) частота работы таймера $f_{T1} = f_{CLK}/(K_{ПД}*OCR1A))$,

где f_{CLK} – частота кварцевого генератора, f_{CLK}=8000000 Гц;

$K_{ПД}$ – коэффициент пределителя таймера T1;

OCR1A – значение регистра сравнения канала A таймера T1 в режиме совпадения;

б) для получения заданной частоты 20 Гц коэффициент предделителя необходимо выбрать равным $K_{ПД}$ = 8 и тогда

$$OCR1A= f_{CLK}/(K_{ПД}*f_{T1})=50000=0xC350,$$

т.е. OCR1AH=0xC3, OCR1BL=0x50.

3. Затем выберем настройку регистров управления TCCR1A и TCCR1B

TCCR1A=0b0000 0000=0;

TCCR1B=0b0000 1010=$0A – режим CTC и коэффициент пределителя 8.

4. Настройка прерывания:

В таблице векторов микросхемы ATmega32 адрес вектора прерывания таймера T1 по совпадению канала A равен $0E.

В регистре TIMSK 4 бит разрешает прерывание таймера T1 по совпадению канала A, следовательно, TIMSK=0b0001 0000=0x10.

5. Настройка портов ввода/вывода:

– порт A – на вход

– порт B – на выход

6. Последовательность переключения обмоток ШД при прямом вращении приведена в табл. 10, где первая строка значение переменной, которая инкрементирует значение от 0 до 7 и далее повторяет снова с нулевого значения (регистр общего назначения R_X), вторая строка – обмотка двигателя, третья строка – эквивалентное значение битов порта B, при этом учитывается, что обмотка A соединена с выводом PB0, обмотка B – PB1, обмотка /A – PB2, обмотка /B – PB3.

Табл. 10. Последовательность переключения обмоток ШД

R_X	0	1	2	3	4	5	6	7
Обмотка ШД	A	A+B	B	B+/A	/A	/A+/B	/B	/B+A
Порт B	0x01	0x03	0x02	0x06	0x04	0x0C	0x08	0x09

7. При реверсе последовательность перключения обмоток обратная, т.е. переменная R_X декрементирует свое значение от 7 до 0.

8. Для записи значений переключения обмоток используем FLASH-память:

а) данные запишем в свободную область памяти программ, например, с адреса $100, для этого в конце программы используем директиву «.db» и в ней запишем данные из табл. 10;

б) для чтения этих данных используем косвенную адресация данных через указатель Z=zh:zl=R31:R30 и команду LPM – чтение данных из FLASH-память. Напомним, что прежде чем, считывать значение из памяти надо настроить значение этого указателя Z:

– старший байт не меняется во время исполнения всей программы и равен удвоенному значению старшего байта адреса, т.е. zh=2;

– младший байт указателя изменяет свое значение от 0 до 7 при прямом вращении, и от 7 до 0 – при обратном.

Листинг программы приведен ниже.

```
;-----------------------------------------------------------
; НЕСИММЕТРИЧНОЕ УПРАВЛЕНИЕ ШД
; Скорость ШД  +/- 3 об/мин
; Управление - несимметричное
; Таймер T1 - CTC, прерывание по совпадению канала A
; Входы: PA0 - разрешение
;        PA1 - реверс
; Выходы PB0...PB3 - обмотки управления ШД
;-----------------------------------------------------------
.include"m32def.inc"
.cseg
```

```
.org 0
     rjmp reset
.org $0E
     rjmp T1_COMPA
reset:
     ldi r16,8      ; инициализация стека
     ldi r17,$5F
     out sph,r16
     out spl,r17
     clr R16        ; инициализация портов
     ser R17
     out DDRA,r16   ; порт A - на вход
     out DDRB,r17   ; порт B - на выход
     out PORTA,r17
     out TCCR1A,r16 ; инициализация таймера
     out TCCR1B,r16 ; стоп таймера T1
     out TCNT1H,r16
     out TCNT1L,r16
     ldi r17,$C3
     ldi r18,$50
     out OCR1AH,r17
     out OCR1AL,r18
     ldi r17,$0A    ; 0000 1010 - CTC, Кпд=8
     out TCCR1B,r17 ; пуск T1 с частотой 20Гц
     ldi zh,2       ; указатель Z на адрес $100
     clr zl
     ldi r17,$10    ; разрешение прерывания T1 по совпадению канала A
     out TIMSK,r17
     sei            ; глобальное разрешение прерываний
main:
     rjmp main      ; основной цикл
;---Обработчик прерывания T1 --------------------------------
T1_COMPA:
     sbis PINA,0    ; если 0 бит порта A в "1", то пропустить
                    ; следующую строку
     rjmp stop   ; переход на остановку двигателя
     sbic PINA,1    ; если 1 бит порта A в "0", то пропустить
                    ; следующую строку
     rjmp revers ; переход на реверс двигателя
     inc zl         ; прямое вращение
     cpi zl,8
     brlo read
     clr zl
     rjmp read
revers:             ; реверс
     dec zl
     cpi zl,8
     brlo read
     ldi zl,7
read:               ; чтение данных из FLASH-памяти
     lpm
     out PORTB,r0   ; вывод в порт B
     rjmp end
stop:               ; стоп двигателя
     out PORTB,r16
```

```
end:
     reti
;----- Данные во FLASH-памяти ---------------------------------
.org $100
.db 1,3,2,6,4,$C,8,9
;--------------------------------------------------------------
```

Задание на выполнение

1. Разработать программу «Управление двухфазным шаговым двигателем», с использованием таймера Т1 для задания скорости вращения двигателя. Во всех вариантах задания должен быть реализован реверс двигателя. Режим работы двигателя, скорость вращения, биты выдачи данных, тип прерывания таймера Т1 выбираются по таблицам в зависимости от персональных данных обучаемых.

Выбор исходных данных выполняется по таблицам:

а) Порт выдачи данных и режим работы системы управления шагового двигателя выбирается по первой букве фамилии:

Первая буква фамилии	Порт выдачи данных	Режим работы системы управления двигателя	Тип прерывания Т1
А, К, У	A	Симметричный, 1 фаза	Переполнение
Б, Л, Ф	C	Симметричный, 1 фаза	Совпадение А
В, М, Х	D	Симметричный, 1 фаза	Совпадение В
Г, Н, Ц	A	Несимметричный	Переполнение
Д, О, Ч	C	Несимметричный	Совпадение А
Е, П, Ш	D	Несимметричный	Совпадение В
Ж, Р, Щ	A	Симметричный, 2 фазы	Переполнение
З, С, Э	C	Симметричный, 2 фазы	Совпадение А
И, Т, Ю, Я	D	Симметричный, 2 фазы	Совпадение В

б) Биты управления шаговым двигателем, а также очередность их переключения определяется датой рождения:

Дата рождения	Биты управления шаговым двигателем
1, 15, 29	0, 1, 6, 7
2, 16, 30	1, 2, 5, 6
3, 17, 31	0, 2, 4, 6
4, 18	2, 3, 4, 7
5, 19	1, 3, 5, 7
6, 20	7, 6, 5, 1
7, 21	6, 5, 2, 1

Дата рождения	Биты управления шаговым двигателем
8, 22	4, 3, 1, 0
9, 23	3, 2, 1, 7
10, 24	7, 5, 3, 1
11, 25	6, 4, 2, 0
12, 26	4, 5, 0, 1
13, 27	3, 4, 6, 7
14, 28	5, 4, 1, 0

в) Скорости вращения двигателя при N=200 имп/об определяются по месяцу рождения с использованием таблицы:

Месяц рождения	Скорость вращения двигателя, об/мин
Январь	1 / 15
Февраль	2,5 / 25
Март	1,5 / 7,5

Месяц рождения	Скорость вращения двигателя, об/мин
Июль	4,0 / 20
Август	0,5 / 10
Сентябрь	0,8 / 8

Апрель	0,25 / 30
Май	5 / 60
Июнь	0,9 / 9

Октябрь	1,2 / 12
Ноябрь	2,0 / 20
Декабрь	3 / 33

В отчете привести:

– персональные данные и выбор параметров;

– функциональную схему электропривода;

– таблицу последовательности переключения обмоток при прямом вращении и реверсе;

– векторные диаграммы м.д.с. статора и расчет момента в различных фазах переключения обмоток;

– настройку таймера и расчет времени задержки, реализованной на таймере;

– листинг программы;

– блок-схему алгоритма.

2. Разработать программу «Управление m-фазным ШД» регулирования скорости вращения прозвольного m-фазного шагового двигателя с использованием таймера Т1. Все варианты должны обеспечивать униполярное управление, 2 заданные скорости регулирования и реверс двигателя. Система должна иметь: 3 входа (1 скорость, 2 скорость, реверс), N выходов (по количеству фаз двигателя). Параметры для расчета изменяются в зависимости от персональных данных обучаемых.

Выбор исходных данных выполняется по таблицам:

а) Тип шагового двигателя (количество обмоток) и способ переключения обмоток выбираются по дате рождения:

Дата рождения	Кол-во обмоток ШД	Способ управления
1, 16, 31	3	Сим.,1 фаза
2, 17	3	Несим.
3, 18	4	Сим., 1 фаза
4, 19	4	Сим., 2 фаза
5, 20	4	Несим.
6, 21	5	Сим., 1 фаза
7, 22	5	Сим., 2 фаза
8, 23	5	Сим, 3 фаза

Дата рождения	Кол-во обмоток ШД	Способ управления
9, 24	5	Несим. 1фаза/2 фазы
10, 25	5	Несим. 2фазы/3фазы
11, 26	6	Сим., 1 фаза
12, 27	6	Сим., 2 фаза
13, 28	6	Сим., 3 фаза
14, 29	6	Несим. 1фазы/2фазы
15, 30	6	Несим. 2фазы/3фазы

Примечание:

сим. – симметричный, *несимм*. – несимметричный, *1 фаза/2 фазы* – в первом такте включена 1 фаза, во втором – 2 фазы (аналогично *2 фазы/3 фазы*).

б) Частоты выдачи импульсов, определяющие скорость вращения ШД, определяются первой буквой фамилии по таблице:

Первая буква фамилии	Частота выдачи импульсов, Гц
А, У	1,0 / 2,0
Б, Л, Ф	0,5 / 1,5
В, М, Х	1,5 / 3,0
Г, Н, Ц	0,4 / 1,6
Д, О, Ч	0,75 / 2,25

Первая буква имени	Частота выдачи импульсов, Гц
Е, П, Ш	0,8 / 2,4
Ж, Р, Щ	1,5 / 4,5
З, С, Э	1,8 / 5,4
И, Т, Ю	2,5 / 7,5
К, Я	2,2 / 4,4

В отчете привести:

– персональные данные и выбор параметров;

– функциональную схему электропривода;

– таблицу последовательности переключения обмоток при прямом вращении и реверсе;

– векторные диаграммы м.д.с. статора и расчет момента в различных фазах переключения обмоток;

– настройку таймера и расчет времени задержки, реализованной на таймере;

– листинг программы;

– блок-схему алгоритма.

3. Напишите программу включения/выключения выхода PD0 с заданным периодом:

а) 1 мс	б) 5 мс	в) 10 мс	г) 20 мс
д) 50 мс	е) 500 мс	ж) 2 с	з) 5 с

4. Реализуйте программу светофор:

– в дневном режиме продолжительность включения сигналов красный, желтый, зеленый, желтый 10с/2с/4с/1с;

– в ночном режиме включается только желтый свет – 0,5с горит, 1с – не горит.

Режим включения задается входом PB0.

Контрольные вопросы

1. Сколько таймеров содержит микроконтроллер ATmega8535? Какие из них 8-ми разрядные, 16-ти разрядные?
2. Какие регистры определяют работу таймера T1?
3. Поясните назначение регистров управления в целом? За что отвечает регистр управления A? B?
4. Какие биты изменяют режим работы таймера T1? Режим работы вывода канала A? B?
5. Как установить коэффициент предделителя таймера T1 равный 8? 1024?
6. Объясните назначение регистров TIMSK и TIFR.
7. Что выполняют инструкции `sei` и `cli`?
8. Подсчитайте диапазон изменения задержек времени таймера T1.
9. Рассчитайте коэффициент деления и значение регистра сравнения при работе таймера T1 на частоте 2 Гц в режиме переполнения/сравнения по каналу A.

Работа № 8. Микропроцессорное управление скоростью двигателя постоянного тока по схеме ШИП-ДПТ с применением таймера Т1 в режиме ШИМ

Цель работы

Освоить теоретический и практический материал по работе 16-разрядного таймера Т1 микроконтроллера Atmega8535/Atmega32 в режиме широтно-импульсной модуляции. Применить приобретенные навыки при написании программы по управлению скоростью двигателя постоянного тока по схеме ШИП-ДПТ при симметричном или несмметричном управлении.

Программа работы

1. Изучить необходимый теоретический материал о регистрах и функционировании таймера Т1 в режиме ШИМ.
2. Разобраться в программе по использованию таймера Т1, представленной в лабораторной работе.
3. Написать и отладить собственную программу с использованием таймера по заданию преподавателя.

Пояснения к работе

Таймер Т1 микроконтроллера AtmegaXX может работать как в режиме подсчета временных интервалов, так и в режиме широтно-импульсной модуляции (ШИМ).

Поскольку таймер Т1 содержит два канала и два регистра сравнения OCR1A и OCR1B, то он может сформировать два ШИМ-сигнала одновременно. Для этого два канала таймера подключены к соответствующим выводам микроконтроллера. Так, к выходу T1A таймера подключен вывод PB5, а к выходу канала T1B таймера Т1 подключен вывод PD4 микроконтроллера.

В микроконтроллере AtmegaXX таймер Т1, как и таймеры Т0 и Т2, работает в двух режимах широтно-импульсной модуляции: быстрый ШИМ и фазовый ШИМ.

В режиме быстрого ШИМ счетный регистр **TCNT1** таймера производит формирование пилообразной развертки (рис. 1), инкрементируя свое значение по каждому импульсу с предделителя, который устанавливается в регистре управления **TCCR1B** таймера. При достижении счетным регистром максимального значения, определяемого в настройках регистров управления **TCCR1A**, **TCCR1B** происходит его автоматическое обнуление.

В режиме фазового ШИМ обеспечивается модуляция с высокой разрешающей способностью. Отличительной особенностью режима является сигнал развертки с двумя пологими фронтами – нарастающим и спадающим. Счетчик непрерывно производит счет импульсов с выхода предделителя таймера от минимального до максимального значения, а затем – от максимального до минимального. Принцип работы схемы формирования ШИМ пояснен на примере канала А таймера на рис. 2.

При неинвертирующем ШИМ соответствующий выход микроконтроллера (PD4 – для канала сравнения B, PD5 – для канала сравнения A таймера) очищается

при совпадении уставки, записанной в регистрах сравнения **OCR1A**, **OCR1B**, с величиной сигнала развертки, формируемой в регистре **TCNT1** при счете от минимального до максимального значения и устанавливается при счете от максимального до минимального значения (рис. 2). В случае инвертирующего ШИМ логика переключения вывода микроконтроллера инверсная.

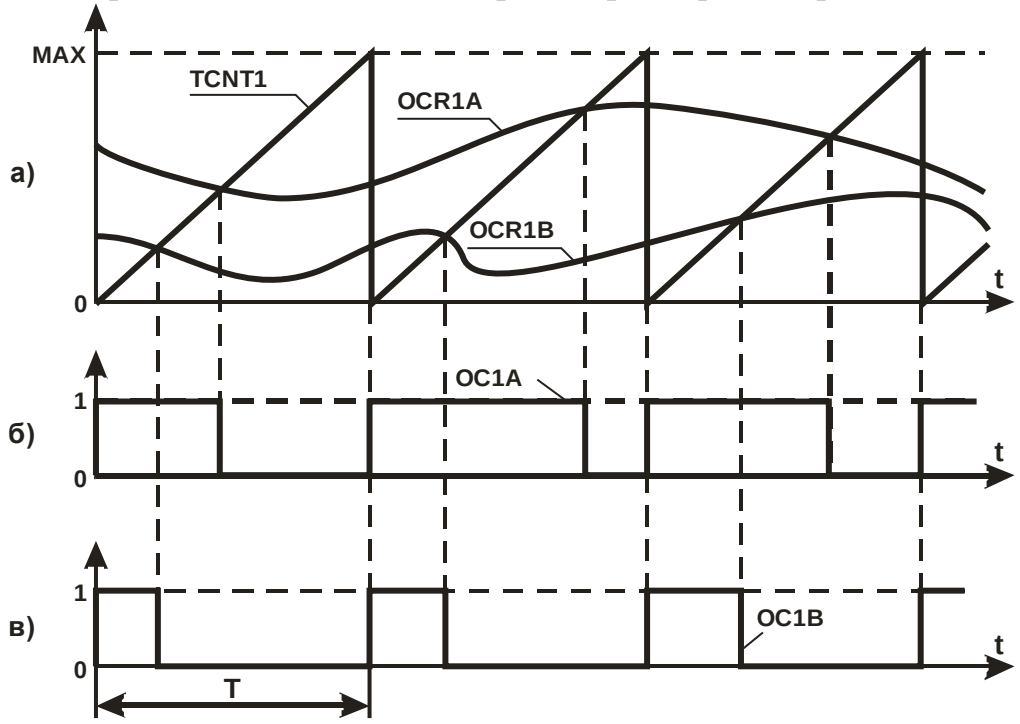


Рис. 1. Принцип работы таймера Т1 в режиме «Быстрый ШИМ»

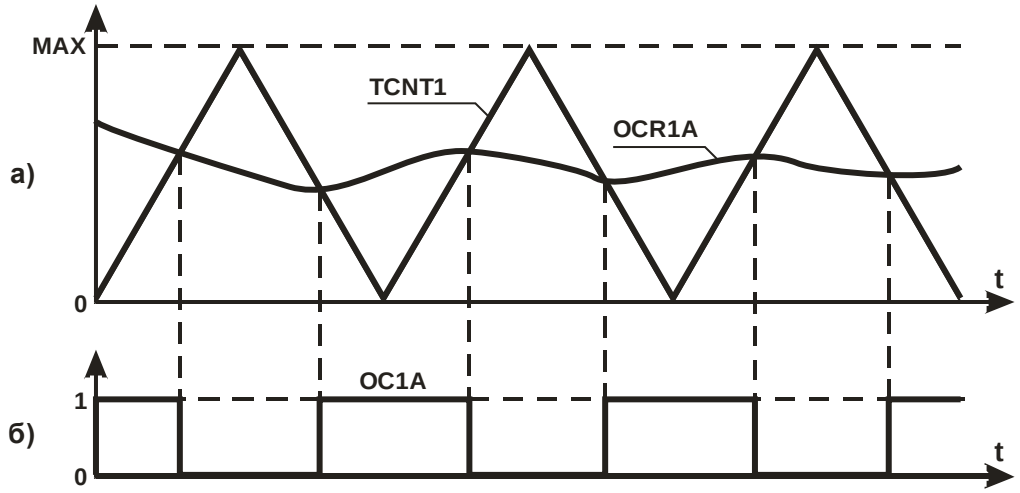


Рис. 2. Принцип работы таймера в режиме «Фазовый ШИМ»

Фазовый ШИМ, реализованный на таймере Т1 чрезвычайно полезен при практическом применении микроконтроллера в качестве устройства для управления, например, транзисторными преобразователями напряжения, когда необходимо тщательно выдерживать паузу между выключением одного сигнала и включением другого. Это требование легко реализуется использованием двух каналов сравнения таймера (рис. 3).

Канал сравнения А таймера инициализируется на неинвертирующий режим ШИМ, а канал сравнения В – на инвертирующий. В этом случае выходы ОС1А и ОС1В микроконтроллера будут работать в противофазе. Для реализации паузы Δt между изменением состояния каналов сравнения А и В в регистры сравнения

OCR1A и **OCR1B** необходимо занести уставки срабатывания, отличающиеся друг от друга на небольшую величину, соответствующую требуемой временной задержке (рис. 3).

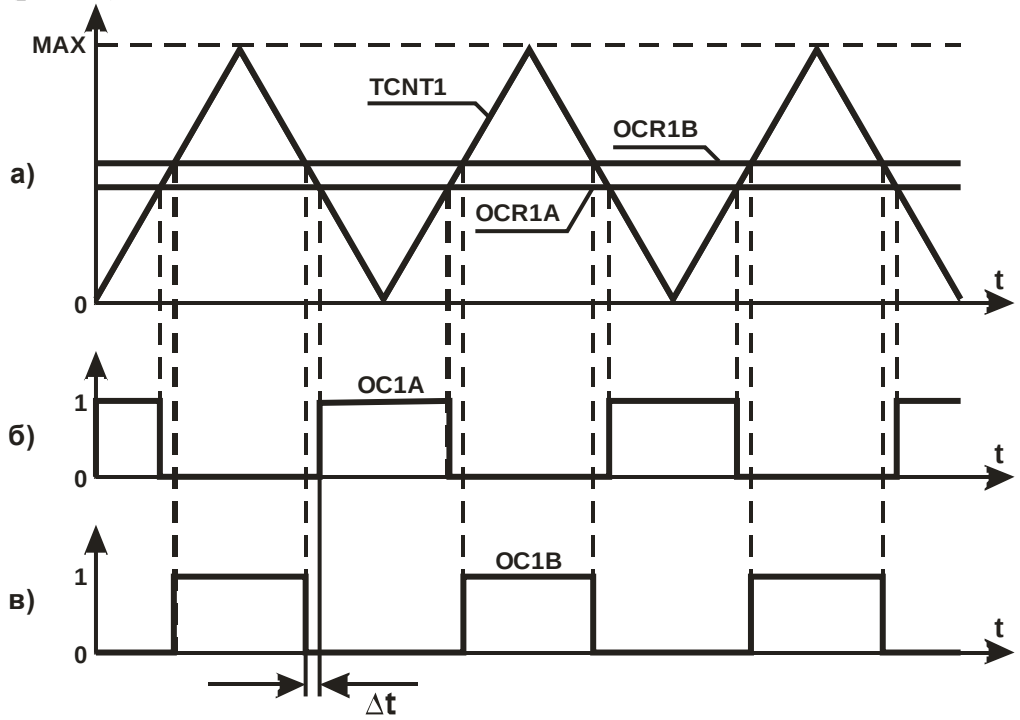


Рис. 3. Пример реализации задержки между двумя сигналами при использовании фазового ШИМ таймера Т1

Разновидностью фазового ШИМ является частотно-фазовый ШИМ. Отличие от фазового ШИМ в этом режиме заключается в моменте обновления регистров сравнения OCR1A и OCR1B. Если в режиме фазового ШИМ обновление регистров происходит при достижении счетным регистром TCNT1 максимального значения, то при использовании частотно-фазового ШИМ обновление регистров происходит при достижении счетным регистром минимального значения (рис. 4). Это позволяет получить в пределах периода *T* ШИМ одинаковые по длительности участки включенного или выключенного состояния выходов OC1A, OC1B микроконтроллера *t1=t3* (рис. 4).

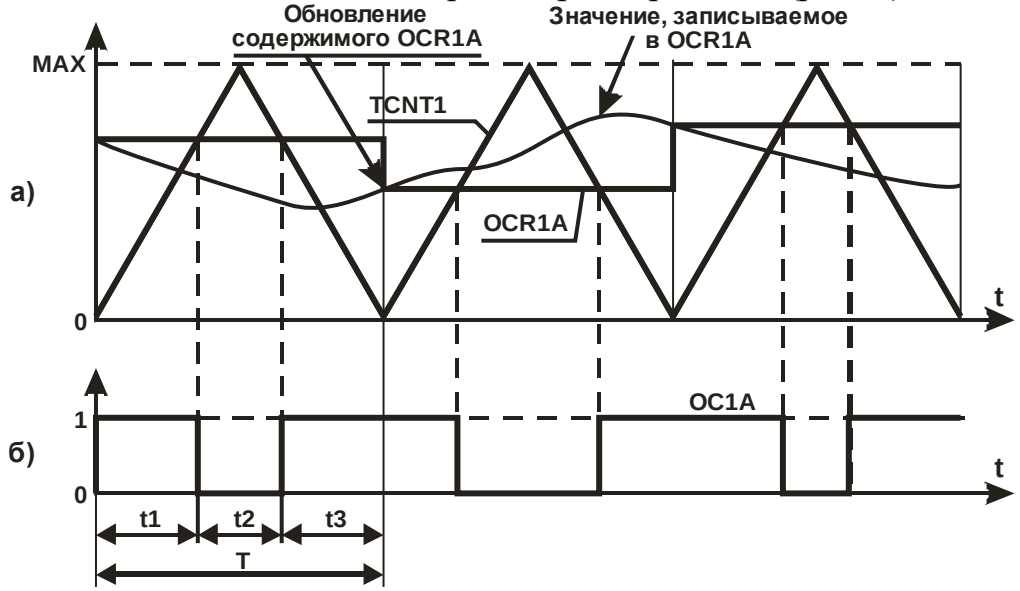


Рис. 4. Принцип работы таймера в режиме частотно-фазового ШИМ

Для управления режимами работы таймера используются регистры управления **TCCR1A**, **TCCR1B**.

Регистр управления **TCCR1A** состоит из управляющих бит, пояснение которых приведено в табл. 1.

Табл. 1. Регистр управления TCCR1A таймера T1

Бит	7	6	5	4	3	2	1	0
Название	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10

Биты 7 и 6 – COM1A1 и COM1A0. В режиме ШИМ эти биты служат для управления выводом OC1A микроконтроллера (PD5). Комбинацией управляющих бит можно либо отключить вывод от таймера, либо инициализировать инвертирующий или неинвертирующий режим ШИМ.

Табл. 2. Функции вывода OC1A таймера T1 в режиме ШИМ

COM1A1	**COM1A0**	**Пояснение**
0	0	Вывод OC1A отключен от таймера
0	1	При установке бита WGM13 в регистре TCCR1B WGM13=0 вывод отключен от таймера. При WGM13=1 изменение состояния OC1A на противоположное при совпадении.
1	0	Неинвертирующий ШИМ. Очистка вывода при совпадении при счете вверх и установка при совпадении при счете вниз.
1	1	Инвертирующий ШИМ. Установка вывода при совпадении при счете вверх и обнуление при совпадении при счете вниз.

Биты 5 и 4 – COM1B1 и COM1B0. Назначение этих бит аналогично вышеописанному для бит COM1A1, COM1A0, но управляют выводом OC1B микроконтроллера (PD4). В режиме COM1B1=0, COM1B0=1 вывод отключен от таймера.

Биты 3 и 2 – FOC1A и FOC1B. Эти биты действуют только в обычном режиме работы при подсчете временных интервалов. При записи в них логической «1» происходит срабатывание прерывания по совпадению соответствующего канала таймера и изменение выводов таймера OC1A и OC1B в соответствии с настройками бит COM1A1:COM1A0, COM1B1:COM1B0.

Биты 1 и 0 – WGM11, WGM10. Эти биты определяют режим работы таймера вместе с управляющими битами регистра TCCR1B.

Регистр управления **TCCR1B** состоит из управляющих бит, пояснение которых приведено в табл. 3.

Табл. 3. Регистр управления TCCR1B таймера T1

Бит	7	6	5	4	3	2	1	0
Название	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10

Бит 7 – ICNC1. Этот бит включает функцию фильтрации внешнего сигнала, поступающего на вывод PB1 микроконтроллера, когда таймер используется в качестве счетчика. Установка бита увеличивает надежность работы таймера. При

активации этого бита между подачей сигнала на вывод PB1 и срабатыванием схемы счета проходит 4 периода тактовой частоты.

Бит 6 – ICES1. Этот бит определяет активный фронт сигнала на входе PB1. При ICES1=0 таймер срабатывает по спадающему фронту сигнала, иначе – по нарастающему фронту.

Бит 5. Зарезервирован и не используется.

Биты 4 и 3 – WGM13, WGM12. Эти биты вместе с битами WGM11, WGM 10 регистра TCCR1A определяют режим работы таймера (табл. 4).

Табл. 4. Задание режима работы таймера T1

Режим	WGM13	WGM12	WGM11	WGM10	Название режима	Вершина	Обновление OCR1x	Переполнение TOV1
0	0	0	0	0	Нормальный режим	0xFFFF	Немедленно	Максимум
1	0	0	0	1	Фазовый ШИМ, 8 бит	0x00FF	На вершине	Минимум
2	0	0	1	0	Фазовый ШИМ, 9 бит	0x01FF	На вершине	Минимум
3	0	0	1	1	Фазовый ШИМ, 10 бит	0x3FFF	На вершине	Минимум
4	0	1	0	0	Очистка при совпадении	OCR1A	Немедленно	Максимум
5	0	1	0	1	Быстрый ШИМ, 8 бит	0x00FF	На вершине	На вершине
6	0	1	1	0	Быстрый ШИМ, 9 бит	0x01FF	На вершине	На вершине
7	0	1	1	1	Быстрый ШИМ, 10 бит	0x03FF	На вершине	На вершине
8	1	0	0	0	Частотно-фазовый ШИМ	ICR1	Внизу	Минимум
9	1	0	0	1	Частотно-фазовый ШИМ	OCR1A	Внизу	Минимум
10	1	0	1	0	Фазовый ШИМ	ICR1	На вершине	Минимум
11	1	0	1	1	Фазовый ШИМ	OCR1A	На вершине	Минимум
12	1	1	0	0	Очистка при совпадении	ICR1	Немедленно	Максимум
13	1	1	0	1	–	–	–	–
14	1	1	1	0	Быстрый ШИМ	ICR1	На вершине	На вершине
15	1	1	1	1	Быстрый ШИМ	OCR1A	На вершине	На вершине

Биты 3…0 – CS12…CS10. Эти биты определяют источник задания частоты таймера T1 и предделитель таймера (табл. 5).

Табл. 5. Функции бит CS12, CS11, CS10 таймера T1

CS12	CS11	CS10	Пояснение
0	0	0	Нет источника. Таймер остановлен.
0	0	1	Предделитель $f_{CLK}/1$
0	1	0	Предделитель $f_{CLK}/8$
0	1	1	Предделитель $f_{CLK}/64$
1	0	0	Предделитель $f_{CLK}/256$
1	0	1	Предделитель $f_{CLK}/1024$
1	1	0	Внешний сигнал на выводе T1 (по спадающему фронту)
1	1	1	Внешний сигнал на T1 (по нарастающему фронту)

В режиме широтно-импульсной модуляции нет необходимости инициализировать прерывания по совпадению или переполнению таймера T1, поэтому описание регистров TIFR и TIMSK в данной работе не приводится.

Пример. Перевести таймер Т1 в режим 10-битного ШИМ и управлять скоростью вращения нереверсивной схемы электродвигателя постоянного тока. Предусмотреть 2 скорости вращения электродвигателя, сигнал задания задавать дискретно с помощью тумблеров, подключенных к порту ввода/вывода А. Организовать предустановленные скорости необходимо в виде таблицы, хранящейся во FLASH-памяти контроллера.

```
//-----------------------------------------------------------------
;Программа для управления электродвигателем постоянного тока.
;Входы:
;         PA0…PA2 – дискретные сигналы задания скорости
;Выходы:
;         PD5 – ШИМ-сигнал на выходе микроконтроллера
.include"m8535def.inc"   ;подключение стандартной библиотеки
.cseg                    ;начало сегмента кода
.org$0
reset:
   ldi r16,low(RAMEND)   ;Инициализация стека
   ldi r17,high(RAMEND)
   out spl,r16
   out sph,r17
   ldi r16,0x00      ;Инициализация портов ввода/вывода
   out DDRA,r16      ;Порт A – на ввод информации
   ldi r16,0xFF
   out PORTA,r16
   out DDRD,r16      ;Порт D – на вывод информации.
   clr r16
   out OCR1AH,r16    ;обнуление регистров Т1
   out OCR1AL,r16
   out TCNT1H,r16
   out TCNT1L,r16
   ldi r16,0xC3      ; 0000 0011 – настройка таймера Т1 в режиме
   out TCCR1A,r16    ; 10-ти разрядного быстрого ШИМ
   ldi r16,0x1A      ; 0001 1010 – пуск таймера Т1
   out TCCR1B,r16
   ldi r31,0x02      ;Передача адреса FLASH в старший байт Z

main:
   in r30,PINA       ;происходит считывание данных, поступающих на
   andi r30,0x01     ;порт А и выделение младшего бита
   lsl r30           ;чтение 2 байт данных
   lpm r17,Z+        ;Считывание старшего байта данных из FLASH
   lpm r16,Z         ;Считывание младшего байта данных из FLASH
   out OCR1AH,r17    ;Передача считанных данных в регистр сравнения
   out OCR1AL,r16    ;канала А таймера Т1
   rjmp main

.org$100             ;По адресу 100 осуществляется
.db 0x00,0x7F,0x03,0xF8  ; запись 2 скоростей ЭД по 2 байта
//-----------------------------------------------------------------
```

Рассмотрим программу более подробно.

1. Сначала происходит подключение библиотеки контроллера Atmega 8535. И перехож к 0 адресу программы:

```
.include"m8535def.inc"
```

```
.cseg
.org$0
```

2. По метке reset осуществляется инициализация портов ввода/вывода и таймера/счетчика T1. В соответствии с заданием порт A инициализируется на ввод, а порт D – на вывод информации:

```
reset:
     ldi r16,low(RAMEND)
     ldi r17,high(RAMEND)
     out spl,r16
     out sph,r17
     ldi r16,0x00
     out DDRA,r16
     ldi r16,0xFF
     out PORTA,r16
     out DDRD,r16
     clr r16
     out OCR1AH,r16
     out OCR1AL,r16
     out TCNT1H,r16
     out TCNT1L,r16
     ldi r16,0xC3
     out TCCR1A,r16
     ldi r16,0x1A
     out TCCR1B,r16
```

Таймер/счетчик T1 инициализируется на режим быстрого ШИМ, при этом задействуется канал сравнения OCR1A, вывод которого работает в режиме неинвертирующего ШИМ.

3. По метке main осуществляется переход на основной цикл программы. В этом цикле осуществляется опрос порта ввода/вывода A, формирование адреса FLASH-памяти в соответствии с комбинацией управляющих входов, а также передача данных из FLASH в регистры сравнения таймера T1:

```
main:
   in r30,PINA
   andi r30,0x01
   lsl r30
   lpm r17,Z+
   lpm r16,Z
   out OCR1AH,r17
   out OCR1AL,r16
   rjmp main
```

Сначала осуществляется считывание данных порта A и выделение младшего бита данных.

Считанные данные преобразуются в адрес FLASH памяти контроллера. Работа с FLASH-памятью контроллера реализуется через специальный Z-регистр, состоящий из двух РОН R31(zh):R30(zl). Для чтения из FLASH необходимо в Z-регистр записать адрес в памяти контроллера, а затем специальной командой lpm считать данные из памяти по указанному адресу.

Поскольку начало массива с предустановленными скоростями соответствует адресу $100, то в регистр Z при необходимости считывания данных из FLASH необходимо записывать адреса от $200. По этой причине в регистр r31 записывается число 0x02, а в регистр r30 записывается адрес, формируемый умножением кода, считанного из PINA, на число 2 (`lsl r30`). Таким образом, при комбинации PINA=0000 0000 в Z-регистр будет записан адрес 0x0200, при комбинации PINA=0000 0001 в Z-регистр будет записан адрес 0x0202 и т.д. То есть адрес FLASH будет всегда указывать на старший байт предустановленной скорости.

При считывании данных из FLASH командой `lpm r17,Z+` в регистр r17 записывается значение FLASH по указанному адресу, то есть старший байт уставки по скорости, а затем адрес FLASH автоматически инкрементируется и при следующем обращении к FLASH командой `lpm r16,Z` в регистр r16 будет записано уже содержимое памяти по адресу, указанному в Z+1, то есть младший байт уставки по скорости.

После считывания данных из памяти они передаются в регистр сравнения канала А таймера Т1 (`out OCR1AH,r17;out OCR1AL,r16`). После этого процесс повторяется.

4. По адресу 100 из FLASH-памяти необходимо записать предустановленные скорости двигателя, в качестве которых выступает значение регистра сравнения OCR1A таймера Т1:

```
.org$100
.db 0x00, 0x7F, 0x03, 0xF8
```

Память разделена на страницы, в которых содержатся 2 байта информации – старший и младший. В случае, если необходимо считать один байт, необходимо умножать адрес страницы на 2. Так, если по адресу $100 содержатся два байта: $100(zh) – 0x00 и $100(zl) – 0x7F, то каждый из этих байт имеет следующий адрес: $200 – 0x00 и $201 – 0x7F.

Во FLASH – памяти записаны следующие уставки скорости:

– скорость 1: 0x007F;

– скорость 2: 0x03F8.

Задание на выполнение

1. Разработать программу «Микропроцессорное управление скоростью двигателя постоянного тока по схеме ШИП-ДПТ», позволяющую регулировать скорость ДПТ изменением скважности выходных сигналов таймера Т1. Во всех вариантах задания должен быть реализован реверс двигателя. Входные сигналы: разрешение работы системы, дискретные входы задания скорости, реверс. Выходные сигналы: дискретные сигналы и сигналы ШИМ, управляющие работой широтно-импульсного преобразователя в симметричном или несимметричном режимах. Режим работы ШИП, скорости вращения (скважности) выбираются по таблицам в зависимости от персональных данных обучаемых.

Выбор исходных данных выполняется по таблицам:

а) Режим работы ШИП, тип, разрдность и частота ШИМ выбираются по первой букве фамилии:

Первая буква фамилии	Режим управления ШИП	Тип ШИМ	Разрядность ШИМ	Частота ШИМ
А, Н, Щ	Симметричный	фазовый	8	f>8кГц
Б, О, Э	Несимметричный	фазовый	9	500Гц<f≤4кГц
В, П, Ю	Симметричный	быстрый	10	2кГц<f≤16кГц
Г, Р, Я	Несимметричный	быстрый	8	1кГц<f≤8кГц
Д, С	Симметричный	фазовый	9	2кГц<f≤16кГц
Е, Т	Несимметричный	фазовый	10	f>2кГц
Ж, У	Симметричный	быстрый	8	f>8кГц
З, Ф	Несимметричный	быстрый	9	1кГц<f≤8кГц
И, Х	Симметричный	фазовый	10	400Гц<f≤3,2кГц
К, Ц	Несимметричный	фазовый	8	f≤4кГц
Л, Ч	Симметричный	быстрый	9	f>8кГц
М, Ш	Несимметричный	быстрый	10	f≤2кГц

в) Скорости вращения двигателя (относительное значение) задаются датой рождения по таблице:

Дата рождения	Скорости вращения
1, 16	0,1 / 0,2 0,4 / 0,8
2, 17	0,05 / 0,1 0,2
3, 18	0,15 / 0,9
4, 19	0,25 /0,5 0,75
5, 20	0,04 / 0,12 0,24 / 0,48

Дата рождения	Скорости вращения
6, 21	0,08 / 0,16 0,24 / 0,32
7, 22	0,07 / 0,21 0,63
8, 23	0,11 / 0,55
9, 24	0,09 / 0,27 0,81
10, 25	0,03 / 0,09 0,27 / 0,81

Дата рождения	Скорости вращения
11, 26	0,12 / 0,18 0,36 /0,72
12, 27	0,14 / 0,21 0,42
13, 28	0,21 / 0,63
14, 29	0,17 / 0,34 0,51
15, 30, 31	0,19 / 0,29 0,49 / 0,99

В отчете привести:

– персональные данные и выбор параметров;

– функциональную схему электропривода;

– таблицу расчета скважности для заданных скоростей двигателя, включая реверс;

– временные диаграммы работы силовых ключей для одной из скоростей и реверса с учетом скважности сигнала и частоты ШИМ;

– настройку таймера и расчет времени задержки, реализованной на таймере;

– листинг программы;

– блок-схему алгоритма.

2. Составить программу для своего варианта, которая реализует вывод сигнала с ШИМ с изменением скважности для заданных таймеров по коду входных сигналов (биты задания).

Варианты задания

№ вар	Таймер (ы), канал (ы)	Режим	Диапазон частот ШИМ	Входы (биты)	Биты задания, количество дискретных значений, диапазон изменения скважности
1	T2	Б / П и И	[1000, 10000]	PA7-переключает между П и И	PC5…PC7, 8 положений, 0…0,5
2	T0 и T2	Ф / И	[1000, 10000]	PA0- включает таймер T0 или T2	PA4…PA7, 16 положений, 0,1…0,9
3	T0	Б / И	[100, 1000]	PC5 – меняет частоту ШИМ	Порт A, 256 положений, 0…1
4	T0 и T2	Ф / П и И	[200, 1000]	Одновременно работают 2 канала	PC0…PC3, 16 положений, 0…0,75
5	T2	Б и Ф / П и И	<200	PA0 – Б и Ф PA7 – П и И	PC0…PC4, 32 положения, 0,2…0,8
6	T0 и T2	Б / П	>10000	PA0 - вкл T0 PA1 - вкл T2	PB0…PB2, 8 положений, 0…1
7	T0	Ф / И	>10000	PC0-вкл/откл	PA0…PA3, 16 положений, 0…1
8	T0	Ф и Б / И	<200	PA4 - переключает между Ф и Б	Порт C, 256 положений, 0…1
9	T2	Б / П	[1000, 10000]	PC1 – изменяет частоту ШИМ	PA2…PA3, 4 положения, 0…0,5
10	T0 и T2	Ф / П и И	[1000, 10000]	PC7 - выбирает T0 или T2	PC0…PC5, 64 положения, 0…1
11	T0 и T2	Б / И	[200, 1000]	PA0-0, PA1-T0, PA2-T2,PA3-T0 и T2	PC4…PC7, 16 положения, 0…1
12	T0	Ф / П	[30, 35000]	PC0…PC2 - 6 частот ШИМ	Порт A, 256 положения, 0…0,5
13	T2	Б / П и И	<200	PA0 – выбор П и И	PA2…PA5, 16 положений, 0…1
14	T0 и T2	Ф / П	>10000	PA0=0-T0 и T2 выкл PA0=1- T0 и T2 вкл	PC3…PC5, 8 положений, 0…0,75
15	T0	Б / И	>10000	PC7- выкл/вкл	PB0…PB3, 16 положений, 0…0,8
16	T0 и T2	Ф / П и И	<200	PA0- меняет режим с П на И T0 и T2 одновременно	PA3…PA7, 32 положения, 0…1
17	T2	Б / И	[1000, 10000]	PB3 – вкл/выкл выход таймера	PB0…PB2, 8 положений, 0…1
18	T0 и T2	Ф и Б / П и И	[1000, 10000]	PA7- Ф и Б PA6- П и И	PA0…PA3, 16 положений, 0…0,5
19	T0 и T2	Б / П	[200, 1000]	PA4-T0 PA5-T2	PC1…PC3, 8 положений, 0,1…0,9
20	T0	Ф и Б / И	[200, 1000]	PA0 – вкл/выкл T0 PA1 – Б и Ф	PA4…PA7, 16 положений, 0,25…1
21	T2	Б / П и И	<200	PC0 – П и И	PB0…PB4, 32 положений, 0…1
22	T0 и T2	Ф / П и И	>10000	PA3- вкл T0 или T2	PA3…PA7, 32 положений, 0…0,8
23	T0	Б / И	>10000	PA0- частота ШИМ	PC, 256 положений, 0…1

24	Т0 и Т2	Ф / П и И	<200	РА0 - выбор Т0 или Т2, РА1 - выбор П или И	РА5…РА7, 8 положений, 0,2…0,9
25	Т0 и Т2	Ф и Б / П и И	[1000, 10000]	РС6 - выбор Ф или В, РС7 - выбор П или И	РВ0…РВ2, 8 положений, 0…0,75
26	Т0	Б / И	[200, 1000]	РА7- вкл/выкл	РА0…РА6, 128 положений, 0…1
27	Т2	Ф / И	[30, 35000]	РА0…РА2 – выбор частоты ШИМ	РС0…РС5, 64 положения, 0…0,8
28	Т0 и Т2	Б / П и И	[1000, 10000]	РС1-выбирает канал	РВ6…РВ7, 4 положения, 0…1
29*	Т0	Ф / П	>10000	sin ШИМ: f=var, A=1	РА0, РА1: 4 частоты ШИМ – 5,10,20,40 Гц
30*	Т2	Б / П	>10000	sin ШИМ: f=50Гц, A = var	РС0…РС2: 8 уровней амплитуды - 0…1,0

Условные обозначения: Б – быстрый ШИМ, Ф – фазовый ШИМ, П – прямой ШИМ, И – инверсный

В отчете привести:

– исходное задание;
– функциональную схему;
– представить расчет скважности, настройку таймера
– запись данных (скважности) в ОЗУ или ПЗУ;
– листинг программы;
– дизассемблированную программу;
– алгоритм.

Контрольные вопросы

1. Укажите разрядность таймера Т0? Т1? Т2?
2. Какие функции в программе могут выполнять таймеры?
3. Перечислите регистры ввода/вывода, управляющие работой таймера Т0?
4. В каких режимах с ШИМ могут работать таймеры микроконтроллера ATmega8535?
5. Как настроить таймеры Т1 для работы в режиме быстрого ШИМ?
6. Какую функцию выполняет счетный регистр TCNT1 в режиме ШИМ?
7. Для каких целей используется регистр сравнения OCR1A? OCR1B?
8. Какие значения необходимо записать в регистр OCR1A для получения скважности 0, 0,5 и 0,75 при 8-ми битном ШИМе?

Работа № 9. Аналого-цифровой преобразователь

Цель работы

Освоить теоретический и практический материал по работе 10-разрядного аналого-цифрового преобразователя (АЦП) микроконтроллера AtmegaXX. Применить приобретенные навыки при написании программы.

Программа работы

1. Изучить необходимый теоретический материал о принципах работы и функционирования встроенного АЦП микроконтроллера AtmegaXX и освоить его инициализацию.
2. Разобраться в программе по использованию АЦП, представленной в лабораторной работе.
3. Написать и отладить собственную программу с использованием АЦП в соответствии с вариантом.

Пояснения к работе

При использовании микроконтроллера в качестве устройства управления каким-либо процессом часто возникает необходимость оценивать величины аналоговых сигналов, в качестве которых могут выступать сигналы с задающих потенциометров, датчиков обратных связей, термопар и др.

Однако, поскольку микроконтроллер является цифровым устройством, непосредственно оценить величину аналогового сигнала путем опроса ножки микроконтроллера, к которой он подключен, не представляется возможным.

Для преобразования аналогового сигнала в цифровой с целью его последующей обработки предназначен **аналого-цифровой преобразователь** (АЦП), встроенный в микроконтроллеры ATmega8535, ATmega32 и другие контроллеры семейства AVR фирмы Atmel.

АЦП микроконтроллера выполнен 10-разрядным, то есть аналоговый сигнал, поступающий на его вход, может быть разложен на 2^{10}=1024 дискреты. Таким образом, фактически, разрядность АЦП отвечает за его разрешающую способность, или чувствительность преобразователя. **Разрешающая способность АЦП** – это минимальная разница аналогового сигнала при двух измерениях, которую еще различает преобразователь.

Для правильного функционирования АЦП ему необходим некий «эталон», то есть напряжение, которое будет приниматься за базу, относительно которой будет измеряться подаваемый на АЦП аналоговый сигнал. Это эталонное напряжение принято называть **опорным напряжением**. В качестве источника опорного напряжения в микроконтроллерах Atmega8535 может выступать напряжение питания контроллера, внутренний стабилизированный источник 2,56В или внешний сигнал, подключаемый к выводу AREF микросхемы контроллера.

Минимальное напряжение, которое АЦП сможет распознать, можно рассчитать по формуле:

$$U_{min} = \frac{1}{2^n} \cdot U_{ref}.$$

При использовании в качестве источника опорного напряжения питания микроконтроллера 5 В:

$$U_{min} = \frac{1}{2^{10}} \cdot 5 = 4{,}88 \text{ мВ}.$$

Ввиду того, что в качестве АЦП в микроконтроллерах AVR используется **АЦП последовательного приближения**, процесс преобразования аналогового сигнала в пропорциональный ему цифровой код занимает некоторое время. Упрощенная структура АЦП последовательного приближения представлена на рис. 1.

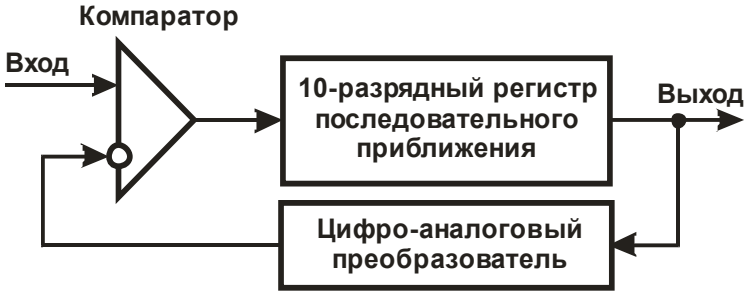


Рис. 1. Структура АЦП последовательного приближения

АЦП состоит из компаратора, регистра последовательного приближения и цифро-аналогового преобразователя. Работает АЦП следующим образом.

В начале преобразования все выходы регистра последовательного приближения устанавливаются в состояние логического «0», за исключением старшего разряда: 10 0000 0000. При этом на выходе внутреннего цифро-аналогового преобразователя формируется аналоговый сигнал, равный половине диапазона АЦП (рис. 2, интервал t0..t1). Компаратор при этом измеряет разницу между входным сигналом и сигналом с выхода ЦАП.

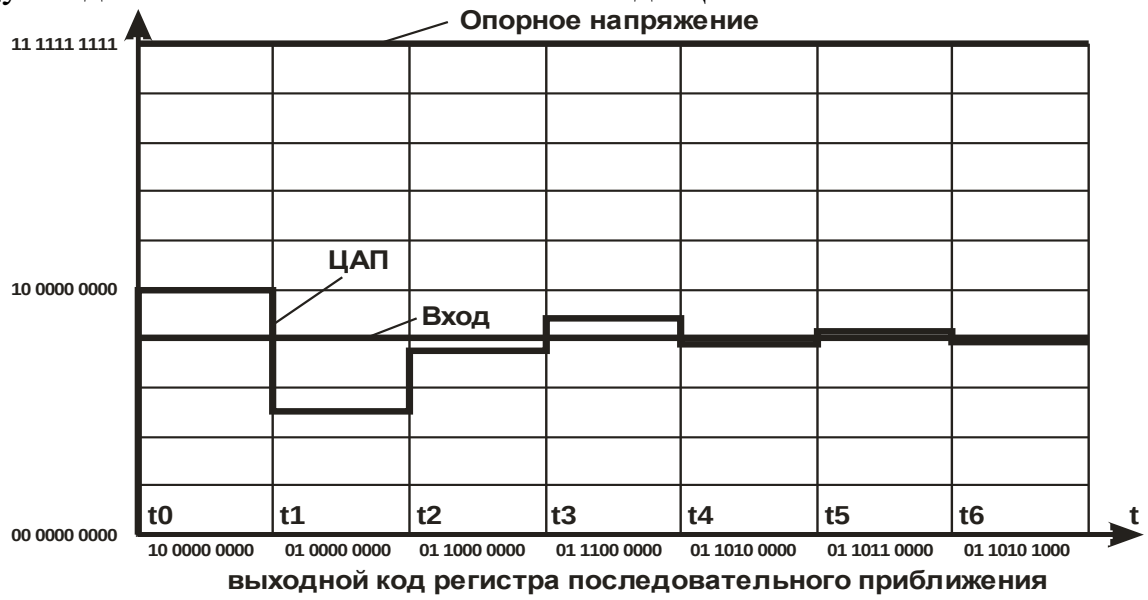


Рис. 2. Принцип работы АЦП последовательного приближения

Если напряжение на входе АЦП оказывается больше, чем установленное на выходе ЦАП, то регистр последовательного приближения сохраняет принятое изначально состояние 10 0000 0000. Если же измеряемое напряжение оказывается меньше подаваемого с ЦАП, регистр устанавливается в состояние 01 000 0000 (рис. 2, интервал t1-t2).

Если принятое состояние 01 0000 0000 оказывается меньше измеряемого сигнала, 8-й бит кода закрепляется и 7-й бит кода регистра последовательного приближения устанавливается в единичное состояние: 01 1000 0000 (рис. 2, интервал t2-t3). Поскольку и при прибавлении этого бита сигнал на выходе ЦАП оказался меньше входного (рис. 2), 7-й бит закрепляется, а к коду прибавляется 6-й бит: 01 1100 0000 (рис. 2, интервал t3-t4).

В этом случае выходной сигнал ЦАП оказывается больше сигнала на входе, поэтому 6-й бит принимается равным «0», а к коду прибавляется 5-й бит: 01 1010 0000 (рис. 2, интервал t4-t5). Далее процесс повторяется до установки последнего младшего бита кода регистра последовательного приближения.

По окончании процесса преобразования результат передается в два 8-разрядных регистра **ADCH** и **ADCL**.

В микроконтроллерах AVR время преобразования АЦП можно регулировать. Для этого в структуру преобразователя встроен предделитель, подобный тому, который встроен в таймеры T0…T2 микроконтроллера. Необходимо отметить, что с уменьшением времени преобразования уменьшается точность получаемого результата. Рекомендуемая частота работы АЦП находится в пределах 50…200 кГц. При работе преобразователя в этом диапазоне обеспечивается минимальная погрешность при получении результата.

Для того, чтобы правильно рассчитать время преобразования АЦП, необходимо помнить, что процесс преобразования занимает 13 тактов АЦП при его непрерывной работе и 25 циклов при его первом запуске. Аналого-цифровой преобразователь микроконтроллера AtmegaXX содержит 8 входов, подключенных к выводам порта ввода/вывода A. Поскольку преобразователь одновременно может работать только с одним входом, они коммутируются специальным коммутатором (мультиплексором), управляемым программно с помощью регистра **ADMUX** (табл. 1).

Табл. 1 Регистр управления мультиплексором АЦП **ADMUX**

Бит	7	6	5	4	3	2	1	0
Название	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0

Биты 6 и 7 – REFS1, REFS0. С помощью этих бит определяется источник опорного напряжения АЦП (табл. 2).

Табл. 2.Задание источника опорного напряжения АЦП

REFS1	**REFS0**	**Источник опорного напряжения**
0	0	Внешний источник, подключенный к выводу AREF.
0	1	Напряжение питания АЦП, подаваемое на вывод AVCC. К выводу AREF должен быть подключен конденсатор.
1	0	Комбинация не используется.
1	1	Внутренний стабилизированный источник 2,56 В. К выводу AREF должен быть подключен конденсатор.

Бит 5 – ADLAR. Этот бит отвечает за «левое» выравнивание результата. Поскольку для хранения 10-разрядного результата преобразования АЦП используются два 8-разрядных регистра ADCH и ADCL, некоторые биты регистра ADCH остаются неиспользуемыми. Если ADLAR=1, то результат преобразования АЦП сохраняется «левым» выравниванием по регистру ADCH. Это особенно

удобно, если пользователю не нужны два младших бита результата преобразования АЦП (табл. 3).

Табл. 3. Представление результата преобразования АЦП в зависимости от управляющего бита ADLAR

ADLAR=0								
ADCH	–	–	–	–	–	–	ADC9	ADC8
ADCL	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2	ADC1	ADC0
ADLAR=1								
ADCH	ADC9	ADC8	ADC7	ADC6	ADC5	ADC4	ADC3	ADC2
ADCL	ADC1	ADC0	–	–	–	–	–	–

Биты 4…0 – MUX4…MUX0. С помощью этих бит определяется, какой из входов АЦП подключен к преобразователю. Здесь различают два режима работы входов АЦП – одиночный и дифференциальный (табл. 4).

Табл. 4. Определение логики работы входов АЦП в соответствии с битами MUX4…MUX0

MUX4…MUX0	**Одиночные входы**	**Положительный дифференциальный вход**	**Отрицательный дифференциальный вход**	**Коэффициент усиления**
00000	ADC0	–		
00001	ADC1			
00010	ADC2			
00011	ADC3			
00100	ADC4			
00101	ADC5			
00110	ADC6			
00111	ADC7			
01000	–	ADC0	ADC0	10
01001		ADC1	ADC01	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
11110	1,22 В	–		

MUX4…MUX0	Одиночные входы	Положительный дифференциальный вход	Отрицательный дифференциальный вход	Коэффициент усиления
11111	0 В			

При решении типовых задач обычно используется одиночное включение входов (табл. 4), однако в некоторых случаях (например, для исключения воздействия на входы микроконтроллера сигнала помехи) возникает необходимость использования дифференциальных входов. В этом случае АЦП измеряет сигнал не между конкретным аналоговым входом ADC0…ADC7 и общей точкой, а между положительным и отрицательным дифференциальными входами (табл. 4).

В некоторых случаях, когда необходимо выловить разницу в очень близких сигналах, полезно перед аналого-цифровым преобразованием эти сигналы усилить. С этой целью перед подачей на АЦП микроконтроллер может усилить дифференциальный сигнал в 1…10…200 раз (табл. 4).

Для калибровки преобразователя можно также ко всем входам подключить источник стабильного напряжения 1,22 В (MUX4…0=11110) или общую точку (MUX4…0=11111).

Настройка аналого-цифрового преобразователя и управление им осуществляется с помощью регистра управления АЦП **ADCSRA** (табл. 5).

Табл. 5. Регистр управления АЦП ADCSRA

Бит	7	6	5	4	3	2	1	0
Название	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0

Бит 7 – ADEN. Этот бит разрешает использование АЦП. Записью логического «0» в ADEN АЦП будет немедленно выключено.

Бит 6 – ADSC. Запись логической «1» в этот бит разрешает преобразование АЦП. Необходимо записывать в ADCS логическую «1» для начала каждого преобразования. По окончании преобразования бит будет автоматически сброшен в состояние логического «0».

Бит 5 – ADATE. Записью логической «1» в этот бит разрешается автоматический запуск АЦП по событию. Событие выбирается комбинацией битов ADTS в регистре **SFIOR** микроконтроллера.

Бит 4 – ADIF. Этот бит – флаг готовности результата преобразования АЦП.Устанавливается автоматически по окончании процесса преобразования.

Бит 3 – ADIE. Этот бит – маска прерывания готовности результата преобразования АЦП. Если ADIE=1, то при появлении флага готовности результата ADIF будет сгенерировано прерывание.

Биты 2…0 – ADPS2…ADPS0. Комбинация этих бит устанавливает делитель тактовой частоты процессора для тактирования АЦП (табл. 6).

Табл. 6. Предделитель АЦП микроконтроллера Atmega8535

ADPS2	ADPS1	ADPS0	Предделитель
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8

1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

В регистре специальных функций битами ADTS2…ADTS0 задается источник автозапуска АЦП при активации этой функции в регистре **ADCSRA**.

Табл. 7. Регистр специальных функций контроллера SFIOR

Бит	7	6	5	4	3	2	1	0
Название	ADTS2	ADTS1	ADTS0	–	–	–	–	–

Табл. 8. Источник автозапуска АЦП (биты ADTS2…ADTS0)

ADTS2	**ADTS1**	**ADTS0**	**Источник автозапуска**
0	0	0	Непрерывное преобразование АЦП
0	0	1	Аналоговый компаратор
0	1	0	Внешнее прерывание INT0
0	1	1	Прерывание по совпадению таймера T0
1	0	0	Прерывание по переполнению таймера T0
1	0	1	Прерывание по совпадению канала B таймера T1
1	1	0	Прерывание по переполнению таймера T1
1	1	1	Прерывание по захвату сигнала таймера T1

Пример 1. Подать аналоговый сигнал на вход ADC2 АЦП и вывести 8-разрядный результат на порт ввода/вывода C. Не использовать прерывание готовности результата преобразования АЦП.

```
;-----------------------------------------------------------------
Входы:
     PA2 – аналоговый вход АЦП
Выходы:
     PC0…PC7 – индикация кода результата преобразования.

.include"m8535def.inc"   ;Подключение стандартной библиотеки
.cseg                    ;Начало сегмента кода
.org$0                   ;по адресу 0

   ldi r16,low(RAMEND)   ;Запись адреса вершины стека
   ldi r17,high(RAMEND)  ;В конце памяти данных
   out spl,r16
   out sph,r17
   ldi r16,0xFB      ;Инициализация неиспользуемых ножек порта A
   out PORTA,r16     ;на ввод информации
   clr r16           ;Используемый вход АЦП не инициализируется
   out DDRA,r16
   out PORTC,r16     ;Инициализация порта C на вывод информации
   ser r16
   out DDRC,r16
   ldi r16,0x62      ;Инициализация мультиплексора АЦП
   out ADMUX,r16
   ldi r16,0xC7
   out ADCSRA,r16    ;Инициализация АЦП и его запуск

main:                ;Начало основного цикла
   in r16,ADCSRA     ;Опрос регистра ADCSRA
```

```
   SBRC r16,4        ;Если бит 4 чист, то следующая команда
   пропускается
   rcall ADC_ready   ;Иначе вызов ADC_ready
   rjmp main         ;Возврат на main

ADC_ready:           ;По метке ADC_ready
   in r16,SREG       ;сохранение значения регистра SREG
   ldi r17,0x97      ;Обнуление флага ADIF записью в него логической
   «1»
   out ADCSRA,r17
   in r17,ADCH       ;Считывание результата преобразования из ADCH
   out PORTC,r17     ;и вывод результата на PORTC
   ldi r17,0xC7      ;Перезапуск АЦП
   out ADCSRA,r17
   out SREG,r16      ;Восстановление регистра SREG
   ret               ;возврат по адресу, хранящемуся в стеке
;---------------------------------------------------------------------
```

Рассмотрим программу более подробно.

1. Сначала происходит подключение стандартной библиотеки контроллера Atmega8535, указывается адрес начала сегмента кода и выполняется инициализация стека, вершина которого располагается в конце памяти данных:

```
.include"m8535def.inc"
.cseg
.org$0
     ldi r16,low(RAMEND)
     ldi r17,high(RAMEND)
     out spl,r16
     out sph,r17
```

2. Производится инициализация портов ввода/вывода. В программе используются порты ввода/вывода A и C. При этом необходимо запомнить, что, поскольку АЦП использует альтернативные функции порта A, то выводы порта, используемые АЦП, инициализировать не нужно. Остальные выводы рекомендуется «притянуть» к уровню логического «0» или «1» во-избежание наведения на них помех. Порт C инициализируется на вывод информации:

```
     ldi r16,0xFB
     out PORTA,r16
     clr r16
     out DDRA,r16
     out PORTC,r16
     ser r16
     out DDRC,r16
```

3. Далее выполняется инициализация АЦП установкой необходимых управляющих бит в регистрах ADMUX и ADCSRA:

```
     ldi r16,0x62
     out ADMUX,r16
     ldi r16,0xC7
     out ADCSRA,r16
```

В качестве источника сигнала опорного напряжения выбирается напряжение электропитания процессора, поданное на вывод AVCC. Также выбирается «левое» выравнивание результата преобразования ADLAR. Выбирается второй канал АЦП (`ldi r16,0x62;out ADMUX,r16`). В регистра ADCSRA устанавливаются биты ADEN (включение АЦП), ADSC (запуск преобразования АЦП), а также выбирается предделитель clk/128 для повышения точности преобразования.

4. В основном цикле ведется опрос флага окончания преобразования АЦП, расположенного в регистре ADCSRA:

```
main:
     in r16,ADCSRA
     SBRC r16,4
     rcall ADC_ready
     rjmp main
```

Сначала считывание значение регистра ADCSRA в РОН r16 (`in r16,ADCSRA`). После этого опрашивается 4-й бит этого регистра, соответствующий флагу ADIF (`SBRC r16,4`). Если бит чист, то пропускается следующая за директивой `SBRC` инструкция. Иначе происходит переход на метку ADC_ready (`rcall ADC_ready`).

5. По метке ADC_ready происходит обработка результата преобразования АЦП:

```
ADC_ready:
     in r16,SREG
     ldi r17,0x97
     out ADCSRA,r17
     in r17,ADCH
     out PORTC,r17
     ldi r17,0xC7
     out ADCSRA,r17
     out SREG,r16
     ret
```

При переходе по метке ADC_ready сначала в РОН r16 необходимо сохранить значение регистра SREG (`in r16,SREG`). После этого необходимо вручную сбросить флаг готовности преобразования АЦП записью в него логической «1» (`ldi r17,0x97; out ADCSRA,r17`). Необходимо запомнить, что все флаги в микроконтроллерах AVR сбрасываются записью в них логической «1». Далее в РОН r17 считывание результат преобразования АЦП, хранящийся в ADCH (`in r17,ADCH`). Два младших бита, хранящиеся в ADCL, отбрасываются. Содержимое r17 передается в порт ввода/вывода C (`out PORTC,r17`), осуществляется перезапуск АЦП, восстановление регистра SREG (`out SREG,r16`) и выход из подпрограммы на адрес, записанный в стеке (`ret`).

Пример 2. Задача повторяет пример 1, но используется прерывание по готовности результата преобразования АЦП. Не использовать «левое» выравнивание результата.

```
;-----------------------------------------------------------------
```

```
;Входы:
;    PA2 – аналоговый вход АЦП
;Выходы:
;    PC0…PC7 – индикация кода результата преобразования.
.include"m8535def.inc"   ;Подключение стандартной библиотеки
.cseg                    ;Начало сегмента кода
.org$0                   ;По адресу $0
rjmp reset               ;осуществляется переход на reset
.org$0E                  ;При переходе по адресу $0E
rjmp ADC_ready           ;осуществляется вызов функции обработки
                         ;прерывания готовности АЦП.
reset:
   cli                   ;запрет всех прерываний.
   ldi r16,low(RAMEND)   ;инициализируется стек
   ldi r17,high(RAMEND)
   out spl,r16
   out sph,r17
   ldi r16,0xFB          ;Производится инициализация портов
   out PORTA,r16         ;ввода/вывода
   clr r16
   out DDRA,r16
   out PORTC,r16
   ser r16
   out DDRC,r16
   ldi r16,0x42
   out ADMUX,r16         ;Инициализация АЦП
   ldi r16,0xCF
   out ADCSRA,r16
   sei                   ;глобальное разрешение прерываний
main:                    ;Главный цикл программы
   rjmp main             ;выполнен пустым

ADC_ready:               ;По метке ADC_ready:
   cli                   ;Осуществляется глобальный запрет прерываний
   in r16,SREG           ;Сохраняется регистр SREG
   in r17,ADCL           ;Считываются 8 младших бит результата из
   ADCL
   in r18,ADCH           ;и 2 старших бита из ADCH
   lsr r17               ;два младших бита результата отбрасываются
   lsr r17
   clr r19
met1:
   lsl r18               ;Два старших бита результата в РОН r18
   inc r19               ;смещаются на 6 позиций влево.
   cpi r19,0x06
   brne met1
   or r17,r18            ;Происходит логическое сложение r17 и r18
   out PORTC,r17         ;Результат выводится на PORTC.
   ldi r17,0xCF
   out ADCSRA,r17        ;Происходит перезапуск АЦП
   out SREG,r16          ;Восстанавливается регистр SREG
   sei                   ;Глобальное разрешение прерываний
   reti                  ;Выход из подпрограммы обработки прерывания
;---------------------------------------------------------------------
```

Рассмотрим программу более подробно, показав отличия от программы, рассмотренной в примере 1.

1. В начале программы, помимо подключения библиотеки контроллера Atmega8535 указываются адреса прерываний: reset – нулевое прерывание по адресу $0, прерывание готовности результата преобразования АЦП по адресу $0E:

```
.include"m8535def.inc"
.cseg
.org$0
rjmp reset
.org$0E
rjmp ADC_ready
```

При попадании на эти векторы происходит переход на соответствующие метки reset и ADC_ready.

2. Порты ввода/вывода и указатель стека инициализируются так же, как и в примере 1. Однако инициализация АЦП производится по-другому:

```
reset:
   cli
   ldi r16,low(RAMEND)
   ldi r17,high(RAMEND)
   out spl,r16
   out sph,r17
   ldi r16,0xFB
   out PORTA,r16
   clr r16
   out DDRA,r16
   out PORTC,r16
   ser r16
   out DDRC,r16
   ldi r16,0x42
   out ADMUX,r16
   ldi r16,0xCF
   out ADCSRA,r16
   sei
```

В регистре управления мультиплексором ADMUX убирается бит «левого» выравнивания результата. Это не связано с использованием прерывания по готовности АЦП, а сделано для демонстрации работы с двумя регистрами хранения результата АЦП: ADCH и ADCL. В регистре управления АЦП ADCSRA ставится маска ADIE на прерывание готовности АЦП для его активации. После всех установок необходимо осуществить глобальное разрешение прерываний (`sei`).

3. В отличие от примера 1, основной цикл программы выполнен пустым:

```
main:
     rjmp main
```

Это сделано по причине того, что при использовании нет необходимости занимать процессор постоянной проверкой флага прерывания готовности АЦП. При появлении флага прерывания основной цикл main будет автоматически прерван, адрес возврата сохранен в стеке, а программа перейдет по метке ADC_ready, указанной в векторе $0E.

4. При переходе по метке ADC_ready осуществляется обработка прерывания по готовности АЦП:

```
ADC_ready:
     cli
     in r16,SREG
     in r17,ADCL
     in r18,ADCH
     lsr r17
     lsr r17
     clr r19
met1:
     lsl r18
     inc r19
     cpi r19,0x06
     brne met1
     or r17,r18
     out PORTC,r17
     ldi r17,0xCF
     out ADCSRA,r17
     out SREG,r16
     sei
     reti
```

При входе в подпрограмму сначала производится глобальный запрет всех прерываний (`cli`). При использовании одного прерывания этого можно не делать, однако правильно при входе в любую подпрограмму обработки прерываний остальные – запрещать.

После этого производится сохранение регистра SREG в РОН r16. Обнуление флага прерывания не производится, так как при использовании прерывания это делается автоматически.

Далее результат преобразования считывается из регистров ADCH (старший) и ADCL (младший). Поскольку результат преобразования АЦП – 10-разрядное число, в регистре ADCH (r18) будут использованы два младших бита: 0000 00XX, а в регистре ADCL (r17) – все биты: XXXX XXXX.

Для использования восьми старших битов результата необходимо сместить с потерей 0 и 1 битов регистр r17, приведя его к виду: 00XX XXXX (`lsr r17; lsr r17`), сместить два младших бита регистра r18 на 6 позиций влево, приведя его к виду: XX00 0000:

```
met1:
     lsl r18
     inc r19
     cpi r19,0x06
     brne met1
```

После этого регистры r17 и r18 складываются (`or r17,r18`), а результат выводится на порт С. Далее АЦП перезапускается, регистр SREG восстанавливается (`out SREG,r16`), производится глобальное разрешение всех прерываний (`sei`), а потом – выход из подпрограммы обработки прерывания (`reti`).

Задание на выполнение

1. Напишите программу применения АЦП, чтобы сигнал с канала ADC0 АЦП (10 разрядный код) полностью выводился в два порта: порт D – младшие 8 разрядов кода, порт C – 2 старших разряда.

2. Измените программу таким образом, чтобы при левом выравнивании результата в регистре данных в порта D выводился:

– 8 разрядный код;

– 6 разрядный код;

– 4 разрядный код;

– 2 разрядный код.

3. Напишите программу вычисления разности по двум каналам ADC0 и ADC1 и вывода 10-разрядного сигнала в дополнительном коде.

4. Составьте программу, которая при входном аналоговом напряжении от 0 до $U_{ОП}/4$ выдает на семисегментный индикатор цифру «0», в диапазоне $U_{ОП}/4 < U_{ВХ} \leq U_{ОП}/2$ – цифру «1», и далее аналогично «2» и «3».

5. Напишите программу плавного управления скоростью вращения двигателя M: используйте АЦП для ввода аналогового сигнала с потенциометра и таймер T1 для вывода сигнала в виде ШИМ.

6. Напишите программу «Вольтметр» для динамической индикации входного напряжения на семисегментных индикаторах. В целях упрощения при обработке сигнала используются только 8 двоичных разрядов. Вывод выполняется:

– в шестнадцатеричном коде (00…FF);

– в десятичном коде (000…255);

– в виде напряжения (десятичный код с фиксированной запятой 0,00…5,00).

Работа № 10. Динамическая индикация символов

Цель работы

Освоить теоретический и практический материал по реализации динамической индикации с использованием микроконтроллера Atmega 8535. Применить приобретенные навыки при написании программы.

Программа работы

1. Изучить необходимый теоретический материал о принципах индикации символов на светодиодном семисегментном индикаторе.
2. Разобраться в программах, представленных в лабораторной работе.
3. Написать и отладить собственную программу динамической индикации.

Пояснения к работе

В настоящее время подавляющее число промышленных приборов оснащаются цифровыми индикаторами для отображения различных величин. В данной работе рассматриваются методы работы с семисегментными индикаторами и рассматриваются примеры программ.

Простейший семисегментный индикатор (рис. 1) представляет набор отдельных сегментов (светодиодов), при зажигании которых в определенной последовательности можно получить набор цифр и определенных символов.

Каждый сегмент индикатора имеет унифицированное буквенное обозначение в соответствии с латинским алфавитом. Верхний горизонтальный сегмент имеет обозначение «A», все последующие сегменты по часовой стрелке имеют обозначения «B», «C», «D», «E», «F», «G», «H» (рис. 1).

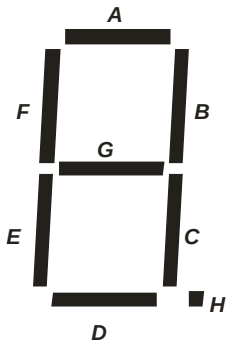
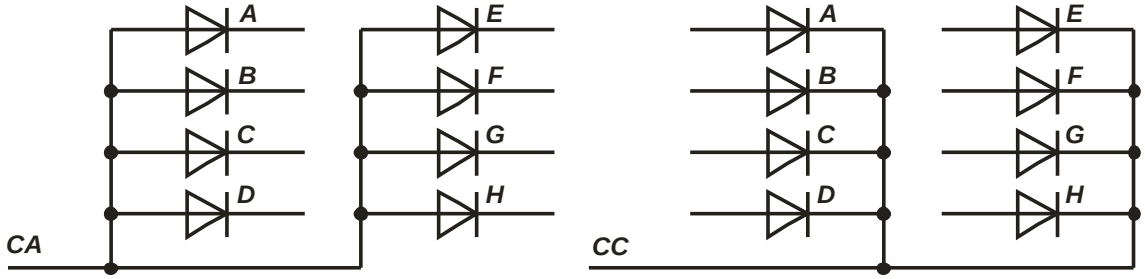


Рис. 1. Внешний вид и структура семисегментного индикатора

Для того, чтобы зажечь сегмент индикатора, необходимо подать напряжение на соответствующий светодиод A … H. Семисегментные индикаторы выпускаются двух типов – с общим анодом (рис. 2, а) и с общим катодом (рис. 2, б). Это делается для упрощения работы с индикатором и уменьшения количества выводов его микросхемы. Действительно, при использовании схемы с общим анодом аноды всех светодиодов объединяются, на них подается положительное напряжение. Для того, чтобы зажечь, например, сегмент «C», необходимо катод

соответствующего светодиода через токоограничивающий резистор присоединить к общему проводу.



а) б)

Рис. 2. Семисегментные индикаторы с общим анодом и общим катодом

При использовании индикаторов с общим катодом на него подключается шина с нулевым потенциалом, а на аноды светодиодов через токоограничивающие элементы подключается напряжение питания.

Таким образом, для того, чтобы зажечь, например, цифру 3, необходимо осуществить подачу напряжения на светодиоды A, B, C, D, G.

На практике для упрощения работы с индикаторами применяют схемы промежуточного усиления, предназначенные для усиления сигналов управления индикаторами и удобного управления их сегментами. Пример такой схемы приведен на рис. 3.

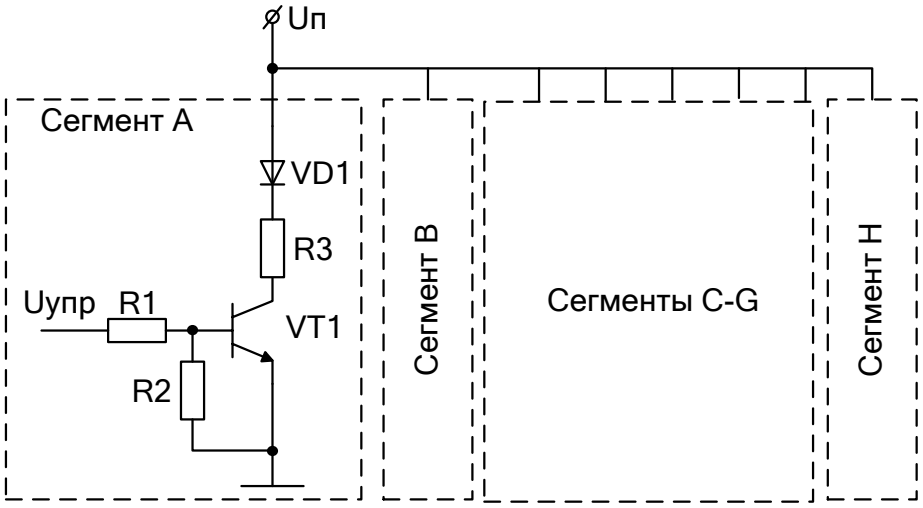


Рис. 3. Схема для управления индикатором

На общие аноды сегментов подается напряжение электропитания $U_{\text{П}}$. Катоды сегментов через токоограничивающий резистор и транзистор подключаются к общей шине. Очевидно, что для зажигания сегмента необходимо включить транзистор (для сегмента A – транзистор VT1).

Включение транзистора VT1 осуществляется подачей на его базу тока управления, который появляется при подаче напряжения управления $U_{\text{УПР}}$ на токоограничивающий резистор R1. Таким образом, при подаче от микроконтроллера сигнала логической «1» транзистор VT1 открывается и светодиод VD1 начинает светиться. Применение рассмотренной схемы позволяет включать сегменты сигналами логической «1», а не логического «0».

Пример 1. Написать программу, осуществляющую вывод на индикатор числа от 0 до F в шестнадцатеричном коде в соответствии с комбинацией сигналов на входе порта ввода/вывода A.

```
//-----------------------------------------------------------------
;Программа вывода чисел 0…F на один разряд семисегментного индикатора
;Входы:
;    PA3…PA0 – задание кода числа
;Выходы:
;    PD0 – управление подачей напряжения на индикатор
;    PC7…PC0   - управление сегментами индикатора

.include"m8535def.inc"   ;Подключение библиотеки Atmega8535
.cseg                    ;Начало сегмента кода.
.org$0                   ;По адресу 0
                         ;во Flash
reset:                   ;По метке reset осуществляется:
     ldi r16,low(RAMEND) ;Инициализация стека. Вершина стека – в
     ldi r17,high(RAMEND);конце памяти данных
     out spl,r16
     out sph,r17
     clr r16             ;Инициализация портов ввода/вывода
     out PORTC,r16       ;Порт C – на вывод
     out PORTD,r16       ;Порт D – на вывод
     ser r16
     out DDRC,r16
     out DDRD,r16
     ldi r16,0x0F
     out PORTA,r16       ;младшие 4 бита порта A – на ввод
     clr r16
     out DDRA,r16
     ldi r16,0x01        ;Установка младшего бита порта D с целью
     out PORTD,r16       ;   подачи   напряжения   на   общие   аноды
     индикатора

main:                    ;По метке main
     in r16,PINA         ;Происходит считывание данных с порта A
     andi r16,0x0F       ;и выделение бит PA3…PA0.
     mov r30,r16         ;формирование адреса flash исходя из
     ldi r31,0x02        ;считанных данных
     lpm r16,Z           ;извлечение в r16 данных из flash
     out PORTC,r16       ;и вывод их на порт C (катоды индикатора).
     rjmp main           ;переход на main и зацикливание программы

.org$100                 ;По адресу 100
.db 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07 ; таблица
.db 0x7F, 0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71 ; кодов символов

//-----------------------------------------------------------------
```

Рассмотрим программу более подробно.

1. Сначала производится подключение библиотеки используемого контроллера Atmega 8535. После этого объявляется адрес начала сегмента кода.

2. По метке reset сначала происходит инициализация стека, затем – инициализация периферийных устройств микроконтроллера. В данном случае из периферийных устройств используются только порты ввода/вывода.

```
reset:
     ldi r16,low(RAMEND)
     ldi r17,high(RAMEND)
     out spl,r16
     out sph,r17
     clr r16
     out PORTC,r16
     out PORTD,r16
     ser r16
     out DDRC,r16
     out DDRD,r16
     ldi r16,0x0F
     out PORTA,r16
     clr r16
     out DDRA,r16
     ldi r16,0x01
     out PORTD,r16
```

В регистр указателя стека записывается адрес вершины стека, который соответствует концу памяти данных (`ldi r16,low(RAMEND); ldi r17,high(RAMEND); out spl,r16; out sph,r17`). Это необходимо для правильной работы программы при использовании подпрограмм и переходов. Порт C в программе будет подключен к катодам индикатора, поэтому он инициализируется на вывод, младший бит порта D по заданию управляет подачей напряжения питания на общие аноды индикатора, поэтому порт инициализируется на вывод, биты PA3…PA0 порта ввода/вывода A инициализируются на ввод информации.

3. В основном цикле программы сначала происходит опрос порта ввода/вывода A:

```
main:
     in r16,PINA
     andi r16,0x0F
```

Производится опрос всего порта (`in r16,PINA`), после чего путем побитового умножения результата на число 0xFF (`andi r16,0x0F`) происходит выделение младших значащих бит порта A.

4. По результату, считанному с порта A, производится формирование адреса во Flash-памяти, по которому хранится соответствующий символ. Поскольку во Flash память данные хранятся словами (2 байта), то для доступа к отдельному байту указывается двойной адрес. Поскольку адрес начала массива данных 0x100 (адрес в словах), то адрес первого байта – 0x200 (в байтах).

Для чтения данных из Flash в Z-регистре (r31:r30) указывается адрес памяти, а затем командой lpm происходит считывание данных:

```
     mov r30,r16
```

```
ldi r31,0x02
lpm r16,Z
```

5. После считывание данных из Flash они выводятся на порт C, соединенный с индикатором:

```
out PORTC,r16
rjmp main
```

Далее программа зацикливается (`rjmp main`) для постоянного опроса порта ввода/вывода A.

6. По адресу Flash-памяти записывается таблица кодов символов:

```
.org$100                    ;По адресу 100
.db 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07 ; таблица
.db 0x7F, 0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71 ; кодов символов
```

Таблица кодов символов включает 16 кодовых комбинаций, каждая из которых соотвествует выводимому на индикатор символу в шестнадцатеричном коде. Таблица цифр от 0 до F приведена в табл. 1.

Табл. 1. Соответствие цифры и его двоичного и шестнадцатеричного кодов

Цифра	Двоичный код	Шестнадцатеричный код
0	0b00111111	0x3F
1	0b00000110	0x06
2	0b01011011	0x5B
3	0b01001111	0x4F
4	0b01100110	0x66
5	0b01101101	0x6D
6	0b01111101	0x7D
7	0b00000111	0x07
8	0b01111111	0x7F
9	0b01101111	0x6F
A	0x01110111	0x77
b	0b01111100	0x7C
C	0b00111001	0x39
d	0b01011110	0x5E
E	0b01111001	0x79
F	0b01110001	0x71

Динамическая индикация символов

Пример 1 показывает, как осуществляется подача символов на один индикатор. Однако часто требуется осуществлять индикацию больших чисел.

Рассмотрим варианты индикации числа 1234 на четырех семисегментных индикаторах. В примере будет использован индикатор с общим анодом (рис. 2, а).

В простейшем случае к общим анодам разрядов индикатора необходимо подключить напряжение питания, а на выводы A…H каждого разряда подать соответствующую кодовую комбинацию (рис. 4).

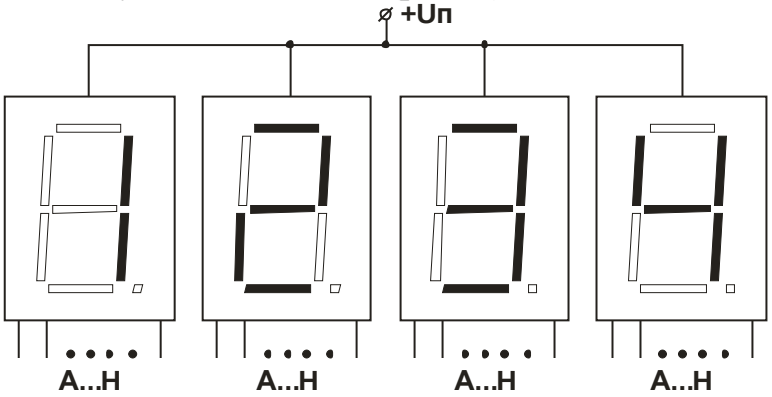


Рис. 4. Пример возможной реализации индикации символов на многоразрядном индикаторе

В случае подобной реализации многоразрядного индикатора возникает существенная проблема – при использовании четырех разрядов количество выводов микроконтроллера, используемых для индикации, составляет 32 шт, при этом всего рабочих выводов у микроконтроллера Atmega 8535 – 32, то есть ресурсы контроллера будут использованы полностью.

Для того, чтобы минимизировать ресурсоемкость процесса индикации, предлагается использовать метод динамической индикации. Этот метод основан на свойстве инерции человеческого зрения, при котором глаз не воспринимает разницу между быстро сменяющимися картинками, если они меняются с частотой, превышающей 25 Гц. Схема динамической индикации представлена на рис. 5.

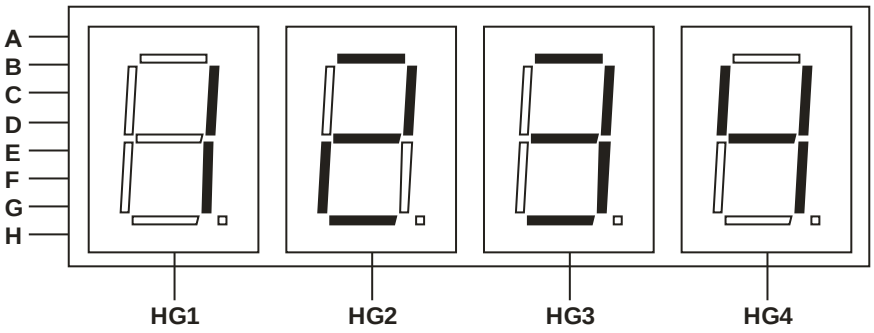


Рис. 5. Схема реализации динамической индикации символов

При динамической индикации соответствующие катоды всех индикаторов объединяются, то есть катоды сегмента A индикаторов HG1…HG4 объединяются в шину A, катоды сегмента B объединяются в шину B и т.д. Общие аноды индикаторов HG1…HG2, наоборот, разъединяются.

Если индикатор реализован подобным образом, то последовательность динамической индикации следующая:

– микроконтроллер выдает код числа 4 на катоды всех индикаторов, при этом напряжение питания подается только на разряд HG4. В результате цифра 4 светится только на индикаторе HG4;

– спустя время, не большее 1/100 секунды, на все сегменты выдается код числа 3, при этом напряжение питания подается только на разряд HG3. В результате цифра 3 светится только на индикаторе HG3;

– на следующем интервале времени на все индикаторы выдается код числа 2, при этом напряжение питания подается только на разряд HG2. В результате цифра 2 светится только на индикаторе HG2;

– в конце цикла на индикаторы подается код числа 1, а напряжение питания – на индикатор HG1. В результате цифра 1 светится только на разряд HG1. Далее процесс повторяется.

Поскольку каждый разряд индикатора обновляется с частотой как минимум 25 Гц, человеческий глаз не замечает мерцания индикатора и процесс отображения числа 1234 кажется постоянным, хотя в один момент времени всегда светится только один разряд индикатора. С увеличением частоты обновления цифр качество индикации улучшается.

Пример 2. Написать программу индикации на четырехразрядном семисегментном индикаторе числа 1234 с использованием динамической индикации. Код числа выдается на порт C, управление разрядами осуществляется с порта D.

```
//-----------------------------------------------------------------
; Программа демонстрации динамической индикации
; На индикатор выводится число 1234
;Выходы:
;    PORTC7…0 – управление сегменты (PC0 - A, PC1 – B, … , PC7 - H);
;    PORTD3 – разряд HG4;
;    PORTD2 – разряд HG3;
;    PORTD1 – разряд HG2;
;    PORTD0 – разряд HG1.

.include"m8535def.inc"   ;Подключение библиотеки Atmega8535
.cseg                    ;Начало сегмента кода.
.org$0                   ;По адресу 0

reset:                   ;По метке reset
     ldi r16,low(RAMEND) ;производится конфигурация указателя стека
     ldi r17,high(RAMEND)
     out spl,r16
     out sph,r17
     clr r16             ;Инициализируются порты ввода/вывода
     out PORTC,r16
     out PORTD,r16
     ser r16
     out DDRC,r16        ;PORTC – на вывод информации
     out DDRD,r16        ;PORTD – на вывод информации
     ldi r17,0x01        ;Включается младший разряд PORTD
     out PORTD,r17       ;Для подачи напряжения на HG1

main:                    ;По метке main
     cpi r17,0x01        ;производится опрос, какой разряд HG
     работает
     breq HG1            ;если первый, то переход на метку HG1
```

```
     cpi r17,0x02        ;если второй,
     breq HG2            ;то переход на метку HG2
     cpi r17,0x04        ;если третий,
     breq HG3            ;то переход на метку HG3
     cpi r17,0x08        ;если четвертый,
     breq HG4            ;то переход на метку HG4
HG1:                     ;По метке HG1
     ldi r31,0x02        ;в Z-регистр задается адрес числа 4 в Flash
     ldi r30,0x04
     rjmp met1           ;и переход на метку met1
HG2:                     ;По метке HG2
     ldi r31,0x02        ;в Z-регистр задается адрес числа 3 в Flash
     ldi r30,0x03
     rjmp met1           ;и переход на метку met1
HG3:                     ;По метке HG3
     ldi r31,0x02        ;в Z-регистр задается адрес числа 2 в Flash
     ldi r30,0x02
     rjmp met1           ;и переход на метку met1
HG4:                     ;По метке HG4
     ldi r30,0x01        ;в Z-регистр задается адрес числа 1 в Flash
     rjmp met1           ;и переход на метку met1
met1:                ;По метке met1
     lpm r16,Z       ;из Flash по указанному адресу считываются данные
     out PORTC,r16   ;и выводятся на PORTC
     clr r16         ;Осуществляется программная задержка времени
met2:
     inc r16
     cpi r16,0xFF
     brne met2
     lsl r17         ;Значение r17 сдвигается на один разряд влево.
     cpi r17,0x10    ;Если r17=0x10, то
     brne met3
     ldi r17,0x01    ;в r17 записывается 0x01
met3:
     clr r16
     out PORTC,r16   ;обнуляется PORTC
     out PORTD,r17   ;обнуляется PORTD
     rjmp main       ;осуществляется переход на main

.org$100             ;По адресу 100
.db 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07 ; массив данных
.db 0x7F, 0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71 ; кодов символов

//-------------------------------------------------------------
```

Рассмотрим программу более подробно.

1. Сначала производится подключение библиотеки используемого контроллера Atmega 8535. После этого объявляется адрес начала сегмента кода:

```
.include"m8535def.inc"
.cseg
.org$0
```

2. По метке reset осуществляется конфигурирование периферийных устройств контроллера и инициализация указателя стека:

```
reset:
     ldi r16,low(RAMEND)
     ldi r17,high(RAMEND)
     out spl,r16
     out sph,r17
     clr r16
     out PORTC,r16
     out PORTD,r16
     ser r16
     out DDRC,r16
     out DDRD,r16
     ldi r17,0x01
     out PORTD,r17
```

После инициализации устройств в регистр r17 записывается число 0x01 (`ldi r17,0x01`), после чего содержимое регистра выводится на порт управления разрядами индикатора (`out PORTD,r17`). Далее в r17 будет находиться значение, соотвествующее включенному состоянию PORTD.

3. По метке main сначала производится опрос включенного состояния PORTD путем опроса регистра r17:

```
main:
     cpi r17,0x01
     breq HG1
     cpi r17,0x02
     breq HG2
     cpi r17,0x04
     breq HG3
     cpi r17,0x08
     breq HG4
```

В зависимости от состояния r17 производится переход на метки HG1…HG4. Так, если в данный момент работает разряд HG1 (`cpi r17,0x01`), то осуществляется переход на метку HG1 (`breq HG1`).

4. Далее, в зависимости от метки, производится запись в Z-регистр значения адреса Flash-памяти, по которому находится код нужного символа:

```
HG1:
     ldi r31,0x02
     ldi r30,0x04
     rjmp met1
HG2:
     ldi r31,0x02
     ldi r30,0x03
     rjmp met1
HG3:
     ldi r31,0x02
     ldi r30,0x02
     rjmp met1
HG4:
     ldi r30,0x01
     rjmp met1
met1:
     lpm r16,Z
```

```
     out PORTC,r16
```

После записи адреса символа осуществляется переход на метку met1, по которой происходит считывание данных из Flash-памяти в РОН r16 (`lpm r16,Z`) и вывод их на PORTC (`out PORTC,r16`).

5. В момент передачи данных на PORTC на одном из разрядов индикатора зажигается нужный символ, после чего необходимо сделать небольшую задержку времени:

```
     clr r16
met2:
     inc r16
     cpi r16,0xFF
     brne met2
```

Для реализации задержки сначала содержимое РОН r16 обнуляется (`clr r16`), после чего происходит его инкремент (`inc r16`) с последующим сравнением с уставкой (`cpi r16,0xFF`). При значении r16, меньшем уставки, происходит возврат на метку met2 (`brne met2`). При достижении уставки программа следует дальше.

6. По окончании выдержки времени необходимо выключить индикатор и переключиться на индикацию другого разряда:

```
     lsl r17
     cpi r17,0x10
     brne met3
     ldi r17,0x01
met3:
     clr r16
     out PORTC,r16
     out PORTD,r17
     rjmp main
```

С этой целью содержимое r17 последовательно сдвигается на одни разряд влево (`lsl r17`), после чего производится выдача его содержимого на PORTD (`out PORTD,r17`). Однако при достижении r17 значения 0x10 (`cpi r17,0x10`) в него необходимо записать значение 0x01 (`ldi r17,0x01`) для того, чтобы разряды PD0…PD3 включались циклично, а не происходило включение разрядов PD4…PD7 порта D. Во-избежание ложного отображения информации перед сменой включенного разряда индикатора происходит выключение сегментов индикатора (`clr r16; out PORTC,r16`). После этого программа зацикливается (`rjmp main`).

7. Адрес Flash-памяти, по которому записывается таблица кодов символов:

```
.org$100
.db 0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07
.db 0x7F, 0x6F, 0x77, 0x7C, 0x39, 0x5E, 0x79, 0x71
```

Пример 3. Динамическая индикация произвольного числа. Два предыдущих примера динамической индикации выполняют вывод на семисегментные индикаторы заведомо известного числа. А как составить программу для вывода произвольного десятичного числа?

Разработаем программу вывода на динамическую индикацию десятичного числа, эквивалентного 8 битам двоичного числа.

Пусть входного двоичное число подается на биты PA7…PA0, тогда диапазон изменения числа X на входе будет равен $X=00000000_2$ ….11111111_2. Такое двоичное число эквивалентно десятичному числу, изменяющемуся в диапазоне X=000…255. Таким образом, для вывода на индикацию потребуется три десятичных разряда, т.е. 3 семисегментных индикатора, работа которых должна быть реализована в форме динамической индикации: X2:X1:X0, где X2 – цифра сотен, X1 – цифра десятков, X0 – цифра единиц.

Для реализации динамической индикации трехразрядного десятичного числа необходимо разбить выполнение программы на 3 такта (продолжительность всех тактов должна быть одинакова и задаваться либо аппаратной задержкой времени на таймерах, либо программной задержкой времени), например, так:

– первый такт:
  – ввод двоичного числа;
  – расчет отдельных цифр числа (разделение исходного числа на разряды) – отдельная подпрограмма ***digit***;
  – вывод на индикаторы цифры единиц;
– второй такт:
  – вывод на индикаторы цифры десятков;
– третий такт:
  – вывод на индикаторы цифры сотен.

Задачу деления трехразрядного десятичного числа на отдельные составляющие – разряды единиц, десятков и сотен – выполним с помощью подпрограммы.

На языках высокого уровня, где существуют операции деление и деление с остатком, вычисление цифр единиц, десятков и сотен выполняется достаточно просто, например, цифру единиц X0 целого числа X на языке Си можно вычислить с помощью формулы:

$$X0=X\%10,$$

а цифру десятков X1 таким образом:

$$X1= (X/10)\%10.$$

К сожалению, на Ассемблере операция деления выполняется очень тяжело – имеет большое количество инструкций и существенно увеличивает время исполнения программы.

Поэтому используем другой алгоритм вычисления десятичных цифр:

– вначале рассчитаем цифру сотен, для этого обнуляем цифру сотен X2=0, далее из исходного числа X последовательно вычитаем число 100 до тех пор пока число не станем меньше 100. На каждом этапе вычитания добавляем 1 в переменную X2 и в конце этого этапа переменная X2 будет содержать цифру сотен исходного числа;

– на втором этапе аналогичным образом вычисляем цифру десятков X1;

– оставшееся значения переменной X даст нам значение цифры единиц X0.

Примечания: для вывода числа в другой системе счисления, отличной от 10, необходимо в формулах подставлять соответствующее значение основания, например, для восьмеричной системы счисления для расчета десятков числа необходимо вычитать число 8, для расчета цифры сотен числа – вычитать число 8^2=64, цифры тысяч – вычитать число 8^3=512 и т.д.

Указанный алгоритм деления числа на отдельные десятичный цифры представлен в подпрограмме ***digit***.

```
;-----------------------------------------------------------
; подпрограммы вычисления десятичных цифр
; для трехразрядного десятичного числа
.def D100 R22       ; имя регистра цифры сотен
.def D10  R21       ; имя регистра цифры десятков
.def D1   R20       ; имя регистра цифры единиц
…
digit:              ; начало подпрограммы расчета цифр
     in   r16,PINA  ; ввод 8-ми битного двоичного кода
     clr  D100      ; очистка регистра цифр сотен
     clr  D10       ; очистка регистра цифр десятков
     clr  D1        ; очистка регистра цифр единиц
sotni:              ; цикл расчета цифры сотен
     cpi  r16,100
     brlo desj
     subi r16,100
     inc  D100
     rjmp sotni
desj:               ; цикл расчета цифры десятков
     cpi  r16,10
     brlo edin
     subi r16,10
     inc  D10
     rjmp desj
edin:               ; цикл расчета цифры единиц
     mov  D1,r16
     ret
;-----------------------------------------------------
```

Задание на выполнение

1. Согласно своим персональным данным составить программу перевода двоичного числа в эквивалентное число другой системы счисления с выводом полученного числа на семисегментные индикаторы, т.е. вводится N-разрядное двоичное число, на индикаторы методом динамической индикации с аппаратной или программной задержкой времени, выводится эквивалент в системе счисления с основанием М. Данные (коды цифр) должны быть записаны и считываться из FLASH памяти.

Выбор исходных данных для составления индивидуального задания выполняется по таблицам:

а) Количество разрядов входного двоичного числа N и порт ввода числа определяются датой рождения по таблице:

Дата рождения	Количество разрядов двоичного числа	Порт ввода двоичного числа
1, 5, 9, 13, 17, 21, 25, 29	5	B

2, 6, 10, 14, 18, 22, 26, 30	6	C
3, 7, 11, 15, 19, 23, 27, 31	7	D
4, 8, 12, 16, 20, 24, 28	8	A

б) Основание системы счисления М определяется первой буквой фамилии по таблице:

Первая буква фамилии	Основание системы счисления М
А, Л, Х	4
Б, М, Ц	5
В, Н, Ч	6
Г, О, Ш	7
Д, П, Щ	8

Первая буква фамилии	Основание системы счисления М
Е, Р, Э	9
Ж, С, Ю	10
З, Т, Я	12
И, У	14
К, Ф	16

в) Время включения одного индикатора определяется месяцем рождения:

Месяц рождения	Время задержки времени Т, мс
1	0,1
2	0,5
3	1
4	2
5	5
6	10

Месяц рождения	Время задержки времени Т, мс
7	8
8	4
9	1,6
10	0,8
11	0,4
12	0,2

г) Тип используемого таймера и его прерывание при реализации задержки времени определяется первой буквой имени по таблице:

Первая буква имени	Тип таймера и его прерывания
А, И, С, Щ	Т0, переполнение
Б, К, Т, Э	Т1, переполнение
В, Л, У, Ю	Т2, переполнение
Г, М, Ф, Я	Т0, совпадение
Д, Н, Х	Т1, совпадение А
Е, О, Ц	Т1, совпадение В
Ж, П, Ч	Т2, совпадение
З, Р, Ш	Программная задержка

В отчете привести:

– свои персональные данные, выбор параметров при составлении задания;

– функциональную схему работы устройства;

– расчет времени задержки и настройку регистров таймеров (если они есть);

– расчет кодов цифр;

– листинг программы с комментариями;

– алгоритм выполнения программы;

– реальные значения задержек времени, полученные в программе AVR-Studio.

2. Составить программу реализации динамической индикации на таймерах T0…T2. Данные предварительно записать в ОЗУ, выводить их на индикацию по адресу записи.

В отчете привести:

– исходное задание;

– функциональную схему;

– расчет настройки таймеров;

– расчет кодов данных;

– листинг программы;

– дизассемблированную программу;

– алгоритм;

– таблицу значений стека во время исполнения программы.

Варианты индивидуальных заданий

№ вар.	Частота динамической индикации, Гц	Таймер	Биты PINA0 и PINA1			
			00	01	10	11
1	1000	T0	Дата.ДеньРождения	ГодРождения	–	–
2	200	T1	Имя	ГодРождения	–	–
3	500	T2	«Add»	«Ldi»	«Clr»	«cpi»
4	4000	T0	НомерГруппы	ГодПоступления	–	–
5	400	T1	«Пуск»	«Стоп»	–	–
6	3000	T2	Век	Год(2посл.Цифры)	–	–
7	250	T0	ДатаРождения	ДеньРождения	ГодРождения	
8	5000	T1	12	12 в 2 системе сч.	12 в 16 системе сч.	12 в 8 системе сч.
9	900	T2	«Abc»	«BCd»	–	–
10	300	T0	Кол-во студ. в группе	НомерГруппы	Год поступления	Год окончания
11	600	T1	«Ab»	«bc»	«cd»	«dE»
12	450	T2	123	456	789	000
13	1500	T0	Имя1	Имя2	–	–
14	150	T1	ДеньРождения	МесяцРождения	ГодРождения	
15	330	T2	1	12	123	1234
16	10000	T0	«in»	«out»	«call»	
17	333	T1	Дата рождения	Месяц рождения	Год рождения	«…»
18	700	T2	Дата.ДеньРождения	ГодРождения	–	–
19	3500	T0	Год рождения	«год»	–	–
20	1200	T1	15	15 в 2-ой системе счисл.	15 в 16-ой системе счисл.	15 в 8-ой системе счисл.
21	4200	T2	jan	feb	apr	Jun
22	560	T0	74	Члб	66	Свд
23	780	T1	руб	euro	dol	–
24	2200	T2	cos	sin	Ln	Lg
25	350	T0	32	64	128	256
26	125	T1	День рождения	Месяц рождения	Год рождения	Год поступления
27	220	T2	10 в 2-ой системе счисл.	10 в 3-ой системе счисл.	10 в 4-ой системе счисл.	10 в 8-ой системе счисл.
28	490	T0	31.28	31.30	31.30	31.31
29	850	T1	2009	2010	2011	2012
30	3300	T2	abc	def	ghi	–

Примечание: если в ячейке таблицы изображен символ «–» это означает, что в данном случае на индикаторы ничего не выводится.

3. Выполните вывод двухбайтного двоичного числа в шестнадцатеричном формате на семи сегментных индикаторах.

4. В порт A задается в двоичном формате первая переменная, в порт D – вторая переменная, на семисегментных индикаторах выводится результат:

– суммирования двух чисел A+B в десятичном формате;

– вычитания A–B в шестнадцатеричном формате;

– умножения A*B в десятичном формате;

– поразрядного логического умножения в шестнадцатеричном формате;

– поразрядного логического сложения в восьмеричном формате.

5. Напишите программу «Секундомер» для вывода с интервалом 1 сек. на семисегментный индикатор чисел от 0 до 200. Паузу реализуйте:

– программным путем;

– на таймере T1 с использованием прерывания по переполнению;

– на таймере T1 с использованием прерывания по совпадению A.

Контрольные вопросы

1. Для чего предназначен семисегментный индикатор?
2. Какие схемы подключения существуют для семисегментного индикатора?
3. Объясните термин «динамическая индикация».
4. При динамической индикации числа используется 6 семисегментных индикаторов. Как рассчитать минимальную частоту обновления индикаторов? Что будет происходить при малой частоте динамической индикации?
5. Поясните последовательность работы программы при выводе двухразрядного числа.
6. Как работает программы при выводе произвольного числа?
7. Каким образом в программе на языке высокого уровня можно выделить отдельную цифру числа? На языке Ассемблера?

Работа № 11. Внешние прерывания

Цель работы

Освоить работу с системой внешних прерываний микроконтроллера ATmegaXX.

Программа работы

1. Изучить необходимый теоретический материал о принципах работы внешних прерываний микроконтроллера ATmegaXX.
2. Разобраться в программе, представленной в лабораторной работе.
3. Написать и отладить собственную программу по заданию преподавателя.

Пояснения к работе

1. При работе высокоскоростных цифровых устройств, синхронизации автономных систем с электрической сетью, подсчете длительности импульсов микроконтроллеру требуется контролировать с высокой точностью момент появления того или иного сигнала.

Производить контроль состояния цифровых входов можно, например, простой директивой `in r16,PORTA`, производя это в основном цикле программы:

```
;-------------------------------------------------------------
main:
…
in r16,PINA
…
rjmp main
;-------------------------------------------------------------
```

Однако при использовании этого способа опроса состояния портов точность фиксации момента перехода вывода порта из одного состояния в другое будет зависеть от длины программы. Действительно, когда произойдет изменение битов порта А заранее неизвестно, самый худший вариант изменение произошло сразу после выполнения чтения регистра состояния `PINA`, и, следовательно, в программе это изменение будет выполнено с существенной задержкой – только после выполнения всех инструкций цикла `main`.

Для того, чтобы при изменении состояния цифровых входов ход выполнения программы происходил сразу же используются, так называемые, внешние прерывания. При возникновении внешнего прерывания исполнение основного цикла программы приостанавливается и осуществляется переход на обработку этого прерывания.

В микроконтроллере ATmegaXX предусмотрена система внешних прерываний, которые обозначаются как INT0, INT1, INT2. Соответственно с каждым прерыванием связан определенный вывод микросхемы: с прерыванием INT0 связан вывод PORTD2, с прерыванием INT1 – PORTD3, с прерыванием INT2 – PORTB2.

Каждому прерыванию соответствует свой адрес в таблице векторов контроллера:

– адрес вектора прерывания INT0 – 0x01;

– адрес вектора прерывания INT1 – 0x02;

– адрес вектора прерывания INT2 – 0x12.

В статическом режиме если состояние сигнала на входах внешних прерываний не меняется, то выполняется основной цикл программы. Когда происходит изменение сигнала внешнего прерывания, контроллер выполняет следующие действия:

– заканчивается выполнение текущей инструкции основного цикла;

– в стеке сохраняется адрес следующей инструкции основного цикла;

– система прерываний контроллера вызывает переход исполнения программы в строку вектора внешнего прерывания, которое произошло;

– по адресу, указанному в векторе прерываний, выполняется переход на подпрограмму-обработчик этого внешнего прерывания;

– после окончания обработки прерывания программа возвращается в основной цикл программы на инструкцию, адрес которой был запомнен в стеке.

Для инициализации внешних прерываний предназначены специальные регистры: общий регистр управления внешними прерываниями **GICR** (General Interrupt Control Register), регистр управления прерываниями INT0 и INT1 **MCUCR**, регистр управления прерыванием INT2 **MCUCSR** и регистр флагов прерываний **GIFR**.

2. Регистр **GICR**

Регистр разрешает работу с внешними прерываниями и содержит 5 разрядов, назначение которых определено (состав приведен в табл. 1).

Табл. 1. Регистр управления GICR

Бит	7	6	5	4	3	2	1	0
Название	INT1	INT0	INT2	–	–	–	IVSEL	IVCE

Биты 7…5: INT1, INT0, INT2. Эти биты отвечают за разрешение или запрещение работы внешних прерываний. Если соответствующий бит установлен в состояние логической «1», то прерывание, управляемое этим битом, активно. В этом случае режим работы внешних прерываний определятся установками в регистрах **MCUCR** и **MCUCSR**.

Биты 4…2. Эти биты в регистре **GICR** зарезервированы и не задействуются при программировании.

Биты 1,0: IVSEL, IVCE. Эти биты в микроконтроллере не предназначены для изменения пользователем и при программировании не изменяются.

3. Регистр **MCUCR**

Этот регистр управляет работой внешних прерываний INT0, INT1 и предназначен для определения логики срабатывания прерываний (табл. 2).

Табл. 2. Регистр управления **MCUCR**

Бит	7	6	5	4	3	2	1	0
Название	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00

Биты 7…4: SM2, SE, SM1, SM0. Эти биты в микроконтроллере не предназначены для изменения пользователем и при программировании не изменяются.

Биты 3, 2: ISC11, ISC10. Эти биты определяют событие, при котором происходит срабатывание прерывания INT1 (табл. 3).

Табл. 3. Биты управления прерыванием INT1

ISC11	ISC10	Описание
0	0	Прерывание по низкому уровню сигнала на PORTD3
0	1	Прерывание при любом изменении PORTD3
1	0	Прерывание по спадающему фронту на PORTD3
1	1	Прерывание по нарастающему фронту на PORTD3

Биты 1, 0: ISC01, ISC00. Эти биты определяют событие, при котором происходит срабатывание прерывания INT0 (табл. 4).

Табл. 4. Биты управления прерыванием INT0

ISC01	ISC00	Описание
0	0	Прерывание по низкому уровню сигнала на PORTD3
0	1	Прерывание при любом изменении PORTD3
1	0	Прерывание по спадающему фронту на PORTD3
1	1	Прерывание по нарастающему фронту на PORTD3

4. Регистр **MCUCSR**

Регистр управления прерыванием INT2 **MCUCSR** предназначен для определения логики срабатывания прерывания INT2 (табл. 5).

Табл. 5. Регистр управления **MCUCSR**

Бит	7	6	5	4	3	2	1	0
Название	–	ISC2	–	–	WDRF	BORF	EXTRF	PORF

Биты 7, 5, 4. Эти биты зарезервированы и для программирования не предназначены.

Биты 3, 2, 1, 0: WDRF, BORF, EXTRF, PORF. Эти биты в микроконтроллере не предназначены для изменения пользователем и при программировании не изменяются.

Бит 6: ISC2. Этот бит определяет логику срабатывания прерывания INT2. Если бит установлен в состояние логической «1», то прерывание срабатывает по спадающему фронту сигнала на входе, иначе – по нарастающему. При изменении состояния бита ISC2 возможно ложное срабатывание прерывания. Поэтому рекомендуется сначала остановить прерывание INT2 обнулением бита INT2 в регистре **GICR**, после этого изменить состояние бита ISC2, после чего очистить флаг прерывания INT2 в регистре **GIFR** и активировать прерывание установкой бита INT2 в регистре **GICR**.

5. Регистр флагов прерываний **GIFR**

Регистр **GIFR** содержит флаги прерываний при их срабатывании. Состав регистра приведен в табл. 6.

Табл. 6. Регистр управления **GIFR**

Бит	7	6	5	4	3	2	1	0
Название	INTF1	INTF0	INTF2	–	–	–	–	–

Биты 7…5: INTF1, INTF0, INTF2. Эти биты содержат флаги соответствующих внешних прерываний. Если прерывание срабатывает и оно разрешено в регистре GICR, выставляется его флаг. При выходе из подпрограммы обработки прерывания по команде reti осуществляется аппаратный сброс флага прерывания. Если есть необходимость ручного сброса прерывания, необходимо в соответствующий бит регистра GIFR записать значение логической «1».

Биты 4…0. Эти биты зарезервированы и для программирования не предназначены.

6. Пример использования внешнего прерывания

Рассмотрим пример, в котором с помощью внешних прерываний реализуется задержка времени.

Пример. Написать программу, в которой, используя внешний источник 50 Гц и внешнее прерывание INT0, осуществляется изменение значения порта С с частотой 2 Гц. Вначале значение порта С инкрементируется от 0 до $F, далее от $F до 0 начинается декремент кода и затем процесс повторяется.

```
;-----------------------------------------------------------------
;Использование внеших прерываний
;Входы:
;    PD2 – подача прямоугольных импульсов с генератора 50Гц
;Выходы:
;    PC0…PC7 – индикация кода на светодиодах

.include"m8535def.inc"   ;подключение библиотеки Atmega8535
.cseg                    ;начало сегмента кода
.org$0                   ;по адресу 0
rjmp reset               ;происходит переход на метку reset
.org$1                   ;по адресу 1
rjmp INT0_ready          ;происходит переход на метку INT0_ready

reset:                   ;По метке reset
     cli                 ;происходит запрет всех прерываний
     ldi r16,low(RAMEND) ;Производится инициализация указателя стека
     ldi r17,high(RAMEND)
     out spl,r16
     out sph,r17
     ldi r16,0xFB        ;Инициализация портов ввода/вывода
     out PORTD,r16
     clr r16
     out DDRD,r16
     out PORTC,r16
     ser r16
     out DDRC,r16
     ldi r16,0x03        ;Инициализация регистра MCUCR
```

```
     out MCUCR,r16
     ldi r16,0x40        ;Инициализация регистра GICR
     out GICR,r16
     clr r16             ;Очистка необходимых РОН
     clr r17
     clr r18
     sei                 ;Разрешение всех прерываний

main:
     rjmp main           ;Бесконечный цикл

INT0_ready:              ;При появлении внешнего прерывания INT0
     cli                 ;Осуществляется запрет всех прерываний
     inc r16             ;Происходит инкремент r16
     cpi r16,$19         ;и его сравнение с уставкой 0x19 (25)
     breq m1             ;Если r16=0x19, то переход на метку m1
     rjmp m3             ;иначе переход на метку m3
m1:                      ;По метке m1
     clr r16             ;РОН r16 очищается
     cpi r18,0x01        ;и производится проверка r18
     breq rev            ;Если r18=0x01, то переход на метку rev
     inc r17             ;иначе происходит инкремент r17
     cpi r17,0x0F        ;и его сравнение с уставкой 0x0F
     brne m2             ;Если r17≠0x0F, то переход на метку m2
     ldi r18,0x01        ;Иначе установка r18=0x01
     rjmp m2             ;и переход на m2
     rev:                ;По метке rev
     dec r17             ;РОН r17 декрементируется
     cpi r17,0x00        ;и его значение сравнивается с уставкой
     brne m2             ;Если r17≠0, то переход на метку m2
     ldi r18,0x00        ;иначе обнуление r18
m2:                      ;По метке m2
     out PORTC,r17       ;осуществляется вывод в порт C
m3:                      ;По метке m3
     sei                 ;осуществляется разрешение всех прерываний
     reti                ;и выход из п/п обработки прерывания
;---------------------------------------------------------------------
```

Рассмотрим основные особенности программы.

1. Вначале программы указываются метки, на которые необходимо переходить программе при появлении прерываний и сбросе. Для этого указываются используемые вектора прерываний reset (адрес 0) и INT0 (адрес 1) согласно технической документации на контроллер:

```
     .org$0
     rjmp reset
     .org$1
     rjmp INT0_ready
```

2. Инициализация стека необходимо, так как в программе используется прерывание

3. Бит PD2 порта D используется на ввод информации (внешне прерывание INT0) для ввода сигнала с генератора 50 Гц.

```
     ldi r16,0xFB
     out PORTD,r16
```

```
  clr r16
  out DDRD,r16
```

4. Далее производится инициализация внешних прерываний:

```
  ldi r16,0x03
  out MCUCR,r16
  ldi r16,0x40
  out GICR,r16
```

Этими инструкциями выбирается режим срабатывания прерывания INT0 по нарастающему фронту сигнала и активируется прерывание INT0 с помощью установки соотвествующего бита в регистре GICR.

5. Обработчик внешнего прерывания. При появлении нарастающего фронта импульса на выводе PD2 микроконтроллера возникает внешнее прерывание, в результате чего происходит переход на вектор прерывания INT0, по которому осуществляется переход на метку INT0_ready. После входа в подпрограмму:

– запрещается срабатывание других прерываний (`cli`);

– в регистре r16 производится подсчет количества сработавших прерываний (`inc r16`);

– когда это число достигает 25 (шестнадцатеричное $19), что соответствует временной задержке 50 мс, осуществляется переход на метку m1 (`breq m1`).

– в противном случае осуществляется переход по метке m3 (`rjmp m3`).

```
INT0_ready:
  cli
  inc r16
  cpi r16,$19
  breq m1
  rjmp m3
```

6. По метке m1 счетчик прерываний очищается, после чего программа определяет, какой процесс в данный момент идет – инкремент или декремент счетчика кода, выдаваемого на индикацию. Если в регистре r18 установлено число 0x01, то это значит, что происходит декремент кода, если r18=0, то происходит инкремент кода. При инкременте производится проверка достижения кодом заданной уставки. При достижении этой уставки направление счета изменяется на противоположное. Аналогичное действие производится при декременте и достижении кодом нулевого значения.

```
m1:
  clr r16
  cpi r18,0x01
  breq rev
  inc r17
  cpi r17,0x0F
  brne m2
  ldi r18,0x01
  rjmp m2
rev:
  dec r17
  cpi r17,0x00
  brne m2
  ldi r18,0x00
```

7. По метке m2 производится вывод значения регистра r17 в порт C и выход из подпрограммы обработки прерываний. Перед выходом из подпрограммы разрешается срабатывание всех остальных прерываний.

Задание на выполнение

1. Используя внешнее прерывание INT1 и таймер T0, напишите программу расчета периода сети и вывода его в двоичном коде (в миллисекундах) в порт C. Примерный алгоритм вычисления может быть следующим. На вывод INT1 подается сигнал с генератора 50 Гц. При появлении нарастающего фронта этого сигнала программа начинает считать импульсы с таймера T0, работающего с частотой 1 кГц (период прохождения импульсов 1 мс). Счет заканчивается при возникновении нового прерывания INT1, после чего результат выдается в порт C. Для того, чтобы информация на выходе не слишком часто менялась, желательно обновлять информацию не 50 раз за секунду, а несколько реже, например, 1 раз в секунду.

2. Выполните аналогичную задачу с выводом информации на 7-сегментный индикатор, при этом используйте динамическую индикацию. Подумайте, как увеличить точность измерения периода.

3. Напишите программу определения частоты сети с точностью не хуже 1 Гц. Результат выводить на 7-ми сегментный индикатор с шагом по времени 1 секунда. Время между сетевыми импульсами измерять с помощью таймера.

Контрольные вопросы

1. Поясните назначение внешних прерываний в микроконтроллере.
2. Сколько внешних прерываний имеет конртоллер ATmegaXX и как они обозначаются?
3. Какие регистры ввода/вывода управляют работой системы внешних прерываний?
4. На какой вход микроконтроллера подается сигнал для оценки прерывания INT0? На какой INT1? INT2?
5. В чем особенность определения внешнего прерывания INT2?
6. Запишите значения битов регистров GICR, MCUCR и MCUCSR для использования:

а) прерывания INT0 по возрастающему фронту сигнала;

б) прерываний INT0 и INT1 по любому фронту сигналов;

в) прерывания INT2 по любому фронту сигналов.

7. В какой области программы прописываются адреса векторов внешних прерываний? Как определить эти адреса?
8. Поясните, как работает программа в приведенном примере.

ЛИТЕРАТУРА

1. Белов А.В. Самоучитель разработчика устройств на микроконтроллерах AVR. – СПб.: Наука и техника, 2008.– 544 с.

2. Бродин В. Б. Системы на микроконтроллерах и БИС программируемой логики / В. Б. Бродин, А. В. Калинин. – М.: ЭКОМ, 2002.- (Современная микропроцессорная техника).-398 с.: ил. (Микроконтроллеры MCS-51 - Микроконтроллеры ADUC812 - Микроконтроллеры Atmel).

3. Баранов В. Н. Применение микроконтроллеров AVR: Схемы, алгоритмы, программы / В. Н. Баранов; Фирма "Atmel".-М.: Додэка: Додэка- XXI, 2004.- (Мировая электроника).- 287 с.: ил.

4. Евстифеев А.В. Микроконтроллеры AVR семейства Tiny и Mega фирмы Atmel. – М.: Издательский дом «Додэка».– 2004.

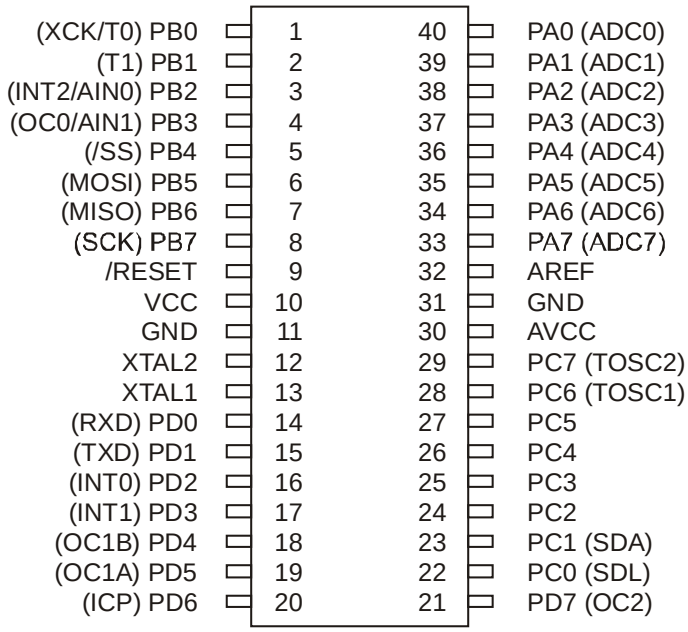
5. Мортон Дж. Микроконтроллеры AVR. Вводный курс. / Пер. с англ. – М.: «Додэка-XXI», 2006.–272 с.

6. Предко М. Руководство по микроконтроллерам: В 2 т./ М. Предко; Пер. с англ. Под ред. И. Шагурина, С. Б. Лужанского. – М.: Постмаркет, 2001-.- (Библиотека современной электроники). Т. 1, 2001.- 415 с.: ил.

7. Предко М. Руководство по микроконтроллерам: В 2 т. / М. Предко; Пер. с англ. под ред. И. И. Шагурина, С. Б. Лужанского.- М.: Постмаркет, 2001. (Библиотека современной электроники). Т. 2, 2001.- 487 с.: ил. (Микроконтроллеры PICMICRO, AVR и Basic Stamp).

8. Трамперт В. AVR-RISC микроконтроллеры. / Пер. с нем.–К.: «МК-Пресс», 2006.– 464 с.

ПРИЛОЖЕНИЕ 1.
Расположение выводов микроконтроллера ATmega8535



Назначение выводов:

RESET – сброс микроконтроллера
VCC – напряжение питания
GND – общий провод
XTAL1, XTAL2 – подключение кварцевого резонатора
AVCC – аналоговое питание для АЦП
AREF – внешний источник опорного напряжения для АЦП
PA0...PA7 – Выводы порта A
PB0...PB7 – Выводы порта B
PC0...PC7 – Выводы порта C
PD0...PD7 – Выводы порта D

Альтернативные функции выводов:

XCK – внешний тактовый вход интерфейса USART
T0, T1 – входы таймеров T0, T1
OC0, OC1A, OC1B, OC2 – выходы таймеров T0, T1, T2
ICP – вход захвата таймера T1
INT0, INT1, INT2 – входы внешних прерываний
AIN0, AIN1 – входы аналогового компаратора
SS – сетевой режим по интерфейсу SPI
MOSI – выход интерфейса SPI
MISO – вход интерфейса SPI
SCK – тактовый вход интерфейса SPI
RXD, TXD – вход и выход USART
SDA, SDL – линии последовательной передачи данных и тактовых импульсов по шине I^2C
TOSC2, TOSC1 – выводы подключение часового резонатора 32768 Гц
ADC0...ADC7 – каналы АЦП

ПРИЛОЖЕНИЕ 2. Регистры ввода/вывода микроконтроллера ATmega8535

Адрес	Обозначение	Наименование	Бит 7	Бит 6	Бит 5	Бит 4	Бит 3	Бит 2	Бит 1	Бит 0
$3F ($5F)	SREG	Регистр статуса	I	T	H	S	V	N	Z	C
			Общее разрешение прерывания	Хранение копируемого бита	Флаг половинного переноса	Флаг знака	Флаг переполнения дополнит. кода	Флаг отриц. значения	Флаг нуля	Флаг переноса
$3E ($5E)	SPH	Указатель стека, ст.байт	-	-	-	-	-	SP10	SP9	SP8
$3D ($5D)	SPL	Указатель стека,мл.байт	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0
$3C ($5C)	OCR0	Регистр сравнения T0								
$3B ($5B)	GICR	Регистр	INT1	INT0	INT2	-	-	-	IVSEL	IVCE
		Разрешения внешних прерываний	Разрешение внешнего прерывания INT1	Разрешение внешнего прерывания INT0	Разрешение внешнего. прерывания INT2				Размещение таблицы преры-ваний: 0-начало Flash,1-загрузчик	Разрешение изменения положения таблицы прерываний
$3A ($5A)	GIFR	Регистр флагов	INTF1	INTF0	INTF2	-	-	-	-	-
		внешних прерываний	Флаг внешнего прерыв. INT1	Флаг внешнего прерыв. INT0	Флаг внешнего прерыв. INT2					
$39 ($59)	TIMSK	Регистр маски	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0
		прерываний таймеров	Флаг разрешения прерывания по совпадению T2	Флаг разрешения прерывания по переполнению T2	Флаг разрешения прерывания по захвату T1	Флаг разрешения прерывания по совпадению T1A	Флаг разрешения прерывания по совпадению T1B	Флаг разрешения прерывания по переполнению T1	Флаг разрешения прерывания по совпадению T0	Флаг разрешения прерывания по переполнению T0
$38 ($58)	TIFR	Регистр флагов	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0
		прерываний таймеров	Флаг прерывания по совпадению T2	Флаг прерыв. по переполнению T2	Флаг прерывания по захвату T1	Флаг прерыв. по совпадению T1A	Флаг прерывания по совпадению T1B	Флаг прерыв. по переполнению T1	Флаг прерывания по совпадению T0	Флаг прерыв. по переполнению T0
$37 ($57)	SPMCR	Регистр управления памятью	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN
$36 ($56)	TWCR	Регистр управления	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE
		интерфейсом TWI	Флаг прерывания	Бит подтверждения	Бит условия старта	Бит условия остановки	Флаг коллизий записи	Бит разрешения		Разрешение прерывания
$35 ($55)	MCUCR	Регистр управления	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00
		микроконтроллера	Режим SLEEP	Разрешение режима SLEEP	Режим SLEEP (SM2:SM1:SM1): 000-Idle, 001- уменьшения шума ADC, 010-PowerDown, 011-PowerSave, 110-Standby, 111-Ext Standby		Условие генерации внешнего прерывания INT1: 00- по низкому уровню, 01- по любому фронту, 10- по отрицательному фронту, 11- по положительному фронту		Условие генерации внешнего прерывания INT0: 00- по низкому уровню, 01- по любому фронту, 10- по отрицательному фронту, 11- по положительному фронту	
$34 ($54)	MCUCSR	Регистр статуса и	–	ISC2	–	–	WDRF	BORF	EXTRF	PORF
		управления микроконтроллера		Чувствительность прерывания INT2			Флаг сброса WatchDog	Флаг сброса Brown-out	Флаг внешнего сброса	Флаг сброса питания
$33 ($53)	TCCR0	Регистр управления	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00
		таймера T0	Принудительное изменение состояния выхода OC0 (Normal и CTC)	Режим работы таймера (WGM01, WGM00): 00-Normal, 01-фазовый ШИМ	Режим работы блока сравнения: 00-таймер отключен от OC0, 01- состояние меняется на противоположное, 10- вывод сбрасыв. в 0,11- вывод сбрасыв. в 1		Режим работы таймера (WGM01, WGM00): 10-CTC, 11- быстрый ШИМ	Управление тактовым сигналом (делитель тактов): 000-таймер остановлен, 001-clk, 010- clk/8, 011-clk/64, 100-clk/256, 101-clk/1024, 110-вывод по отрицательному фронту, 111-вывод по положительному фронту		
$32 ($52)	TCNT0	Счетный регистр T0								
$31 ($51)	OSCCAL	Регистр калибровки clk								
$30 ($50)	SFIOR	Регистр специальных функций	ADTS2	ADTS1	ADTS0	ADHSM	ACME	PUD	PSR2	PSR10
			Источник запуска АЦП: 000- непрерывное преобразования, 001-прерывание от компаратора, 010- INT0, 011- совпадение T0, 100- переполнение T0, 101- совпадение T1, 110- переполнение T1, 111- захват T1			Увеличение скорости преобразования АЦП	Доступ к мультиплексору АЦП при работе компаратора (инверсный вход)	Pull-up недоступен (подтягивающие резисторы не подключаются)	Сброс предделителя таймера T2	Сброс предделителя таймеров T0, T1

$2F ($4F)	TCCR1A	Регистр управления А таймера Т1	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10
			Режим работы блока сравнения А при "Совпадении": 00- таймер отключен от вывода OC1A, 01- состояние меняется на противоположное, 10- вывод сбрасыв. в 0, 11- вывод сбрасыв. в 1		Режим работы блока сравнения при "Совпадении": 00- таймер отключен от вывода OC1B, 01- состояние меняется на противоположное, 10- вывод сбрасыв. в 0, 11- вывод сбрасыв. в 1		Принудительное изменение состояния выхода OC1A (Normal и CTC)	Принудительное изменение состояния выхода OC1B (Normal и CTC)	Режим работы таймера (вместе с битами WGM13 и WGM12 регистра TCCR1B), режимы Normal, CTC (сброс при совпадении), быстрый и фазовый ШИМ	
$2E ($4E)	TCCR1B	Регистр управления В таймера Т1	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10
			Включение схемы подавление помех (4 выборки)	Выбор фронта захвата (0- по отриц. Фронту, 1- по полож.)		Режим работы таймера (вместе с битами WGM11 и WGM10 регистра TCCR1A режимы Normal, CTC (сброс при совпадении), PWM (ШИМ)		Управление тактовым сигналом (делитель тактов): 000-таймер остановлен, 001-clk, 010- clk/8, 011-clk/64, 100-clk/256 101-clk/1024, 110-вывод по отриц.фронту, 111-вывод по полож. фронту		
$2D ($4D)	TCNT1H	Счетный регистр Т1	Старший байт							
$2C ($4C)	TCNT1L	Счетный регистр Т1	Младший байт							
$2B ($4B)	OCR1AH	Регистр сравнения А Т1	Старший байт							
$2A ($4A)	OCR1AL	Регистр сравнения А Т1	Младший байт							
$29 ($49)	OCR1BH	Регистр сравнения В Т1	Старший байт							
$28 ($48)	OCR1BL	Регистр сравнения В Т1	Младший байт							
$27 ($47)	ICR1H	Регистр захвата Т1	Старший байт							
$26 ($46)	ICR1L	Регистр захвата Т1	Младший байт							
$25 ($45)	TCCR2	Регистр управления Т2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20
			Принудительное изменение состояния выхода OC2 (Normal и CTC)	Режим работы таймера (WGM21, WGM20): 00-Normal, 01-фазовый ШИМ	Режим работы блока сравнения при "Совпадении": 00- таймер отключен от вывода OC0, 01- состояние меняется на противоположное, 10- вывод сбрасывается в 0, 11- вывод сбрасывается в 1		Режим работы таймера (WGM21, WGM20): 10-CTC, 11- быстрый ШИМ	Управление тактовым сигналом (делитель тактов): 000-таймер остановлен, 001-clk, 010- clk/8, 011-clk/32, 100-clk/64, 101-clk/128, 110-clk/256, 111-clk/1024		
$24 ($44)	TCNT2	Счетный регистр Т2								
$23 ($43)	OCR2	Регистр сравнения Т2								
$22 ($42)	ASSR	Регистр состояния асинхронного режима	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB
							Переключение режима работы: 0- внутренний такт, 1- со кварца TOSC1/TOSC2	Состояние обновления TCNT2: 0- готов к записи, 1- есть значение	Состояние обновления OCR2: 0- готов к записи, 1- есть значение	Состояние обновления TCCR2: 0- готов к записи, 1- есть значение
$21 ($41)	WDTCR	Регистр управления сторожевым таймером	–	–	–	WDTOE	WDE	WDP2	WDP1	WDP0
						Разрешение изменения сторожевого таймера	Сторожевой таймер включен	Коэффициент деления предделителя: 000-16K, 001-32K, 010- 64K, 011- 128K, 100-256K, 101-512K, 110-1024K, 111-2048K		
$20 ($40)	UBRRH	Регистр скорости передачи данных USART	URSEL	–	–	–	UBRR11	UBRR10	UBRR9	UBRR8
	UCSRC	Регистр В состояния и управления USART	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL
			Выбор регистра: 0- UBRRH, 1- UCSRC	Режим работы: 0- асинхронный, 1- синхронный	Контроли и формирование четности 00- выключен, 10- проверка на четность (even), 11- проверка на нечетность (odd)		Кол-во стоп-битов: 0- 1 стоп-бит, 1- 2 стоп-бита	Формат посылок (вместе с UCSZ2 регистра UCSRB): 00- 5 разрядов, 01-6 разр., 10- 7 разрядов, 11- 8 разрядов		Полярность тактового сигнала в синхронном режиме
$1F ($3F)	EEARH	Регистр адреса EEPROM	–	–	–	–	–	–	–	EEAR8
$1E ($3E)	EEARL	Регистр адреса EEPROM	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0
$1D ($3D)	EEDR	Регистр данных EEPROM								
$1C ($3C)	EECR	Регистр управления EEPROM	–	–	–	–	EERIE	EEMWE	EEWE	EERE
							Разрешение прерывания EEPROM	Разрешение записи в EPROM	Запись в EEPROM	Чтение из EEPROM
$1B ($3B)	PORTA	Регистр данных порта А	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0
$1A ($3A)	DDRA	Регистр напрев. порта А	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0
$19 ($39)	PINA	Выводы порта А	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0
$18 ($38)	PORTB	Регистр данных порта В	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0

$17 ($37)	DDRB	Регистр направ. порта B	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0
$16 ($36)	PINB	Выводы порта B	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0
$15 ($35)	PORTC	Регистр данных порта C	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0
$14 ($34)	DDRC	Регистр направ. порта C	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0
$13 ($33)	PINC	Выводы порта C	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0
$12 ($32)	PORTD	Регистр данных порта D	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0
$11 ($31)	DDRD	Регистр направ. порта D	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0
$10 ($30)	PIND	Выводы порта D	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0
$0F ($2F)	SPDR	Регистр данных SPI								
$0E ($2E)	SPSR	Регистр состояния SPI	SPIF	WCOL	–	–	–	–	–	SPI2X
			Флаг прерывания от SPI	Флаг конфликта записи						Удвоение скорости SPI
$0D ($2D)	SPCR	Регистр управления SPI	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0
			Разрешение прерывания от SPI	Включение SPI	Порядок передачи данных	Выбор режима: 0-Slave, 1-Master	Полярность тактового сигнала: 0-генер «+», 1- генер «-»	Фронт тактового сигнала: 0- передний SCK, 1-задний	Скорость передачи: 00-clk/4, 01- clk/16, 10- clk/64, 11- clk/128	
$0C ($2C)	UDR	Регистр данных USART								
$0B ($2B)	UCSRA	Регистр A состояния и управления USART	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM
			Флаг завершения приема	Флаг завершения передачи	Регистр передатчика пуст	Флаг ошибки кадрирования	Флаг переполнения приемника	Флан ошибки контроля четности	Флаг удвоения скорости передачи	Мультиплексорный обмен
$0A ($2A)	UCSRB	Регистр B состояния и управления USART	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8
			Разрешение прерывания по завершению приема	Разрешение прерывания по завершению передачи	Разрешение прерывания при очистке регистра данных	Разрешение приема	Разрешение передачи	Формат посылок: 0- 5...8 разрядов, 1- 9-разрядов	8- разряд принимаемых данных	8- разряд передаваемых данных
$09 ($29)	UBRRL	Регистр скорости передачи данных USART	Младший байт							
$08 ($28)	ACSR	Регистр состояния и управления аналогового компаратора	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0
			Выключение компаратора	Подключение к неинвертирующему входу внутр. ИОН	Выход компаратора	Флаг прерывания от компаратора	Разрешение прерывания от компаратора	Подключение компаратора к схеме захвата таймера T1	Условие возникновение прерывания от компаратора: 00- любое изменение выхода, 10- изм. с 1 на 0, 11- изм. с 0 на 1	
$07 ($27)	ADMUX	Регистр управления мультиплексором АЦП	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0
			Выбор источника опорного напряжения: 00- внешний, подключенный к AREF, 01 – внешний, подключенный к AVCC, 11– внутренний 2,56В с внешним конденсатором		Выравнивание результата преобразования (1- левое)	Канал АЦП: 00000...00111 – каналы ADC0...ADC7 с K=1; 01001 – (ADC1-ADC0)*10; 01011 – (ADC1-ADC0)*200; 01101 – (ADC3-ADC2)*10; 01111 – (ADC3-ADC2)*200; 10000...10111 - (ADCn-ADC1); 11000...11101 - (ADCn-ADC1); 11110 – внутренний источник 1,22В; 11111 – 0В				
$06 ($26)	ADCSRA	Регистр состояния и управления АЦП	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0
			Разрешение работы	Запуск преобразования	Режим работы: 0- одиночное преобразование, 1- автозапуск в зависимости от ADTS2...0 (SFIOR)	Флаг прерывания от компаратора	Разрешение прерывания от компаратора	Выбор коэффициента деления предделителя АЦП: 000 -2, 001-2, 010-4, 011-8, 100-16, 101-32, 110 -64, 111-128		
$05 ($25)	ADCH	Регистр данных АЦП	Старший байт							
$04 ($24)	ADCL	Регистр данных АЦП	Младший байт							
$03 ($23)	TWDR	Регистр данных TWI								
$02 ($22)	TWAR	Регистр адреса TWI	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE
			7-битный адрес							
$01 ($21)	TWSR	Регистр состояния TWI	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0
			Статус TWI						Предделитель TWI: 00 - K=1, 01 – K=4, 10 – K=16, 11 – K=64	
$00 ($20)	TWBR	Регистр скорости TWI	TWBR7	TWBR6	TWBR5	TWBR4	TWBR3	TWBR2	TWBR1	TWBR0

ПРИЛОЖЕНИЕ 3. Таблица векторов прерываний микроконтроллера ATmega8535

№ вектора прерываний	Адрес	Источник	Примечание
1	$000	RESET	Сброс по выводу RESET и сторожевому таймеру (Hardware Pin, Power-On Reset and Watchdog Reset)
2	$001	INTO	Запрос внешнего прерывания 0 (External Interrupt Reguest 0)
3	$002	INT1	Запрос внешнего прерывания 1 (External Interrupt Reguest 1)
4	$003	TIMER2 COMP	Совпадение при сравнении таймера/счетчика 2 (Timer/Conter2 Compare Match)
5	$004	TIMER2 OVF	Переполнение таймера/счетчика2 (Timer/Conter2 Overflow)
6	$005	TIMER1 CAPT	Захват таймера/счетчика1 (Timer/Conter1 Capture Event)
7	$006	TIMER1 COMPA	Совпадение А при сравнении таймера/счетчика 1 (Timer/Conter1 Compare Match A)
8	$007	TIMER1 COMPB	Совпадение В при сравнении таймера/счетчика 1 (Timer/Conter1 Compare Match B)
9	$008	TIMER1 OVF	Переполнение таймера/счетчика1 (Timer/Conter1 Overflow)
10	$009	TIMER0 OVF	Переполнение таймера/счетчика0 (Timer/Conter0 Overflow)
11	$00A	SPI, STC	Завершение пересылки SPI (SPI Serial Transfer Complete)
12	$00B	USART, RX	Завершение приема USART (UART, Rx Complete)
13	$00C	USART, UDRE	Регистр данных USART пуст (UART Data Register Empty)
14	$00D	USART, TX	Завершение передачи USART (USART, Tx Complete)
15	$00E	ADC	Завершение ADC преобразования (ADC Conversion Complete)
16	$00F	EE_RDY	Готовность EEPROM (EEPROM Ready)
17	$010	ANA_COMP	Срабатывание аналогового компаратора (Analog Comparator)
18	$011	TWI	Последовательный двухпроводной интерфейс Two-wire Serial Interface
19	$012	INT2	Внешнее прерывание External Interrupt Request 2
20	$013	TIMER0 COMP	Совпадение Р при сравнении таймера/счетчика Т0 Timer/Counter0 Compare Match
21	$014	SPM_RDY	Готовность Store Program. Memory Ready

ПРИЛОЖЕНИЕ 4. Таблица векторов прерываний микроконтроллера ATmega32

№ вектора прерываний	Адрес	Источник	Примечание
1	$000	RESET	Сброс по выводу RESET и сторожевому таймеру
2	$002	INTO	Внешнее прерывание 0
3	$003	INT1	Внешнее прерывание 1
4	$006	INT2	Внешнее прерывание 2
5	$008	TIMER2 COMP	Совпадение при сравнении таймера/счетчика T2
6	$00A	TIMER2 OVF	Переполнение таймера/счетчика T2
7	$00C	TIMER1 CAPT	Захват таймера/счетчика T1
8	$00E	TIMER1 COMPA	Совпадение A при сравнении таймера/счетчика T1
9	$010	TIMER1 COMPB	Совпадение B при сравнении таймера/счетчика T1
10	$012	TIMER1 OVF	Переполнение таймера/счетчика T1
11	$014	TIMER0 COMP	Совпадение при сравнении таймера/счетчика T0
12	$016	TIMER0 OVF	Переполнение таймера/счетчика T0
13	$018	SPI, STC	Завершение пересылки SPI
14	$01A	USART, RXC	Завершение приема USART
15	$01C	USART, UDRE	Регистр данных USART пуст
16	$01E	USART, TXC	Завершение передачи USART
17	$020	ADC	Завершение ADC преобразования
18	$022	EE_RDY	Готовность EEPROM
19	$024	ANA_COMP	Срабатывание аналогового компаратора
20	$026	TWI	Последовательный двухпроводной интерфейс
21	$028	SPM_RDY	Готовность памяти программ

ПРИЛОЖЕНИЕ 5
Система команд микроконтроллеров AVR

Арифметические и логические команды

Мнемокод	Операнды	Описание	Операция	Флаги	Кол-во циклов
ADD	Rd, Rr 0≤d≤31 0≤r≤31	Сложить без переноса	Rd ←Rd + Rr	Z, C, N, V, H	1
ADC	Rd, Rr 0≤d≤31 0≤r≤31	Сложить с переносом	Rd ← Rd + Rr+ C	Z, C, N, V, H	1
ADIW	Rd,K dE{24,26,28, 30}, 0≤K≤63	Сложить константу со словом	Rdh:Rdl← Rdh:Rdl+ K	Z, C, N, V	2
SUB	Rd,Rr 0≤d≤31 0≤r≤31	Вычесть без заема	Rd ← Rd – Rr	Z, C, N, V, H	1
SUBI	Rd, K 16≤d≤31 0≤K≤255	Вычесть константу	Rd ← Rd – K	Z, C, N, V, H	1
SBC	Rd, Rr 0≤d≤31 0≤r≤31	Вычесть с заемом	Rd←Rd-Rr-C	Z, C, N, V, H	1
SBCI	Rd, K 16≤d≤32 0≤K≤255	Вычесть константу с заемом	Rd←Rd-K-C	Z, C, N, V, H	1
SBIW	Rd, K dE{24,26,28, 30}, 0≤K≤63	Вычесть константу из слова	Rdh:Rdl← Rdh:Rdl-K	Z, C, N, V	2
AND	Rd, Rr 0≤d≤31 0≤r≤31	Выполнить логическое умножение	Rd ← Rd • Rr	Z, N, V	1
ANDI	Rd, K 16<d<31 0<k≤255	Выполнить логическое умножение с констатой	Rd ← Rd • K	Z, N, V	1
OR	Rd, Rr 0≤d≥31 0≤r≤31	Выполнить логическое сложение	Rd ←Rd v Rr	Z, N, V	1
ORI	Rd, K 16≤d≤31 0≤K≤255	Выполнить логическое сложение с константой	Rd ←Rd v K	Z, N, V	1
EOR	Rd, Rr 0≤d≤31 0≤r≤31	Выполнить исключающее ИЛИ	Rd ←Rd ⊕ Rr	Z, N, V	1
COM	Rd 0≤d≤31	Выполнить дополнение до единицы	Rd←SFF - Rd	Z, C, N, V	1

Мнемокод	Операнды	Описание	Операция	Флаги	Кол-во циклов
NEG	Rd 0≤d≤31	Выполнить дополнение до двух	Rd ← S00 - Rd	Z, C, N, V, H	1
SBR	Rd, K 16≤d≤31 0≤K≤255	Установить биты в регистре	Rd ← Rd v K	Z, N,V	1
CBR	Rd, K 16≤d≤31 0≤K≤255	Очистить биты в регистре	Rd ←Rd • (SFF-K)	Z, N, V	1
INC	Rd 0≤d≤31	Инкрементировать	Rd ←Rd + 1	Z, N,V	1
DEC	Rd 0≤d≤31	Декрементировать	Rd ← Rd - 1	Z, N,V	1
TST	Rd 0≤r≤31	Проверить на ноль или минус	Rd←Rd • Rd	Z, N, V	1
CLR	Rd 0≤d≤31	Очистить регистр	Rd ← Rd ⊕ Rd	Z, N, V	1
SER	Rd 16≤d≤31	Установить все биты регистра	Rd ←SFF	нет	1
MUL	Rd, Rr 0≤d≤31 0≤r≤31	Умножение без знака	R1:R0←Rd x Rr	Z, C	2
MULS	Rd, Rr 0≤d≤31 0≤r≤31	Умножение со знаком	R1:R0←Rd x Rr	Z, C	2
MULSU	Rd, Rr 0≤d≤31 0≤r≤31	Умножение знакового на беззнаковое	R1:R0←Rd x Rr	Z, C	2
FMUL	Rd, Rr 0≤d≤31 0≤r≤31	Умножение дробных чисел без знака	R1:R0← (Rd x Rr)<<1	Z, C	2
FMULS	Rd, Rr 0≤d,r≤31	Умножение дробных чисел со знаком	R1:R0← (Rd x Rr)<<1	Z, C	2
FMULSU	Rd, Rr 0≤d,r≤31	Умножение дробных чисел знакового на беззнаковое	R1:R0← (Rd x Rr)<<1	Z, C	2

Команды сдвигов и операций с битами

Мнемоника	Операнды	Описание	Операция	Флаги	Кол-во циклов
SBI	P,b O≤P≤31 0≤b≤7	Установить бит в регистр I/O	l/0(P,b)← 1	Нет	2
CBI	P,b 0≤P≤31 0≤b≤7	Очистить бит в регистре I/O	l/0(P,b)← 0	Нет	2

Мнемоника	Операнды	Описание	Операция	Флаги	Кол-во циклов
LSL	Rd 0≤d≤31	Логический сдвиг влево	Rd(n+1)←Rd(n), Rd(0)←0	Z,C,N, V,S,H	1
LSR	Rd 0≤d≤31	Логический сдвиг вправо	Rd(n)←Rd(n+1), Rd(7)←0	Z,C,N, V,S,H	1
ROL	Rd 0≤d≤31	Циклический сдвиг влево	Rd(0) ←C, Rd(n+1)←Rd(n), C←Rd(7)	Z,C,N, V,S,H	1
ROR	Rd 0≤d≤31	Циклический сдвиг вправо	Rd(7) ←C, Rd(n)←Rd(n+1), C ← Rd(0)	Z,C,N, V,S,H	1
ASR	Rd 0≤d≤31	Арифметический сдвиг вправо	Rd(n)←Rd(n+1), n=0...6	Z,C,N, V,S	1
SWAP	Rd 0≤d≤31	Поменять нибблы местами	Rd(3...0) <–> Rd(7...4)	Нет	1
BSET	s, 0≤s≤7	Установить флаг	SREG(s)← 1	SREG(s)	1
BCLR	s 0≤s≤7	Очистить флаг	SREG(s)← 0	SREG(s)	1
BST	Rd,b 0≤d≤31 0≤b≤7	Переписать бит из регистра во флаг T	T← Rd(b)	T	1
BLD	Rd,b 0≤d≤31 0≤b≤7	Загрузить T флаг в бит регистра	Rd(b) ← T	Нет	1
SEC		Установить флаг переноса	C← 1	C	1
CLC		Очистить флаг переноса	C←0	C	1
SEN		Установить флаг отрицательного значения	M←1	N	1
CLN		Очистить флаг отрицательного значения	N←0	N	1
SEZ		Установить флаг нулевого значения	Z←1	Z	1
CLZ		Очистить флаг нулевого значения	Z←0	Z	1
SEI		Установить флаг глобального прерывания	I← 1	I	1
CLI		Очистить флаг гло-бального прерывания	I←0	I	1
SES		Установить флаг знака	S← 1	S	1

Мнемоника	Операнды	Описание	Операция	Флаги	Кол-во циклов
CLS		Очистить флаг знака	S←0	S	1
SEV		Установить флаг переполнения	V← 1	V	1
CLV		Очистить флаг переполнения	V←0	V	1
SET		Установить флаг T	T←1	T	1
CLT		Очистить флаг T	T←0	T	1
SEH		Установить флаг полупереноса	H← 1	H	1
CLH		Очистить флаг полупереноса	H←0	H	1
NOP		Выполнить холостую команду		Нет	1
SLEEP		Установить режим SLEEP		Нет	1
WDR		Сбросить сторожевой таймер		Нет	1

Команды пересылки данных

Мнемоника	Операнды	Описание	Операция	Флаги	Кол-во циклов
MOV	Rd,Rr 0≤d≤31 0≤r≤31	Копировать регистр	Rd←Rr	Нет	1
MOVW	Rd,Rr 0≤d≤31 0≤r≤31	Копировать слово	Rd←Rr Rd+1←Rr+1	Нет	1
LDI	Rd,k 16≤d≤31 0≤k≤255	Загрузить константу	Rd←K	Нет	1
LD	Rd, X 0≤d≤31	Загрузить косвенно	Rd ← (X)	Нет	2
LD	Rd, X+ 0≤d≤31	Загрузить косвенно с постинкрементом	Rd ← (X), X←X+1	Нет	2
LD	Rd, –X 0≤d≤31	Загрузить косвенно с преддекрементом	X←X-1, Rd ← (X)	Нет	2
LD	Rd, Y 0≤d≤31	Загрузить косвенно	Rd←(Y),	Нет	2

Мнемоника	Операнды	Описание	Операция	Флаги	Кол-во циклов
LD	Rd, Y+ 0≤d≤31	Загрузить косвенно с постинкрементом	Rd←(Y), Y←Y+1	Нет	2
LD	Rd, –Y 0≤d≤31	Загрузить косвенно с преддекрементом	Y←Y-1, Rd ← (Y)	Нет	2
LDD	Rd, Y+q 0≤d≤31 0≤q≤63	Загрузить косвенно со смещением	Rd←(Y+q)	Нет	2
LD	Rd, Z 0≤d≤31	Загрузить косвенно	Rd ← (Z)	Нет	2
LD	Rd, Z+ 0≤d≤31	Загрузить косвенно с постинкрементом	Rd ← (Z), Z←Z+1	Нет	2
LD	Rd, –Z 0≤d≤31	Загрузить косвенно с преддекрементом	Z←Z-1, Rd←(Z)	Нет	2
LDD	Rd, Z+q 0≤d≤31 0≤q≤31	Загрузить косвенно со смещением	Rd ← (Z+q)	Нет	2
LDS	Rd,k 0≤d≤31 0≤k≤65535	Загрузить непосредственно из ОЗУ	Rd←(k)	Нет	2
ST	X, Rr 0≤r≤31	Записать косвенно	(X) ← Rr	Нет	2
ST	X+, Rr 0≤r≤31	Записать косвенно с постинкрементом	(X) ←Rr, X←X+ 1	Нет	2
ST	–X, Rr 0≤r≤31	Записать косвенно с преддекрементом	X←X-1, (X) ←Rr	Нет	2
ST	Y, Rr 0≤r≤31	Записать косвенно	(Y) ←Rr	Нет	2
ST	Y+, Rr 0≤r≤31	Записать косвенно с постинкрементом	(Y) ← Rr, Y←Y+ 1	Нет	2
ST	–Y, Rr 0≤r≤31	Записать косвенно с преддекрементом	Y←Y-1, (Y) ← Rr	Нет	2
STD	Y+q, Rr 0≤r≤31 0≤q≤63	Записать косвенно со смещением	(Y+q)←Rr	Нет	2
ST	Z, Rr 0≤r≤31	Записать косвенно	(Z) ←Rr	Нет	2
ST	Z+, Rr 0≤r≤31	Записать косвенно с постинкрементом	(Z) ← Rr, Z←Z+ 1	Нет	2
ST	–Z, Rr 0≤r≤31	Записать косвенно с преддекрементом	Z←Z-1, (Z) ← Rr	Нет	2

Мнемоника	Операнды	Описание	Операция	Флаги	Кол-во циклов
STD	Z+q, Rr 0≤r≤31 0≤q≤63	Записать косвенно со смещением	(Z+q)←Rr	Нет	2
STS	k, Rr 0≤d≤31 0≤k≤65535	Загрузить непосредственно в ОЗУ	(k) ←Rr	Нет	2
LPM		Загрузить из памяти программ	R0 ← (Z)	Нет	3
LPM	Rd, Z	Загрузить из памяти программ	Rd ← (Z)	Нет	3
LPM	Rd, Z+	Загрузить из памяти программ с послеинкрементом	Rd ← (Z), Z ← Z+1	Нет	3
SPM		Записать в память программ	(Z) ← R1:R0	Нет	–
IN	Rd, P 0≤d≤31 0≤P≤63	Загрузить данные из порта I/O в регистр	Rd←P	Нет	1
OUT	P, Rr 0≤r≤31 0≤P≤63	Записать данные из регистра в порт I/O	P←Rr	Нет	1
PUSH	Rr 0≤r≤31	Сохранить регистр в стеке	STACK ←Rr	Нет	2
POP	Rd 0≤r≤31	Загрузить в регистр из стека	Rd ← STACK	Нет	2

Команды переходов

Мнемоника	Операнды	Описание	Операция	Флаги	Кол-во циклов
RJMP	k -2K<k<2K	Перейти относительно	PC ←PC + k + 1	Нет	2
LJMP		Перейти косвенно	PC←Z	Нет	2
JMP	k 0<k<4M	Прямой переход	PC←k	Нет	3
RCALL	k -2K≤k≤2K	Вызвать подпрограмму относительно	PC ← PC + k + 1	Нет	3
ICALL		Вызвать подпрограмму косвенно	PC←Z	Нет	3
CALL	k 0≤k≤64K	Прямой вызов подпрограммы	PC←k	Нет	4
RET		Вернуться из подпрограммы	PC ← STACK	Нет	4

Мнемоника	Операнды	Описание	Операция	Флаги	Кол-во циклов
RETI		Вернуться из прерывания	PC ← STACK	I	4
CPSE	Rd, Rr 0≤d≤31, 0≤r≤31	Сравнить и пропустить, если равно	If Rd=Rr then PC ← PC + 2 (or 3)	Нет	1/2/3
CP	Rd, Rr 0≤d≤31 0≤r≤31	Сравнить регистры	Rd-Rr	Z, C, N, V, H	1
CPC	Rd, Rr 0≤d≤31 0≤r≤31	Сравнить регистры с учетом переноса	Rd-Rr-C	Z, C, N, V, H	1
CPI	Rd, K 16≤d≤31 0≤K≤255	Сравнить с константой	Rd-K	Z, C, N, V, H	1
SBRC	Rr, b 0≤r≤31 0≤b≤7	Пропустить, если бит в регистре очищен	if Rr(b)=0 then PC ← PC + 2 (or 3)	Нет	1/2/3
SBRS	Rr, b 0≤r≤31 0≤b≤7	Пропустить, если бит в регистре установлен	If Rr(b)=1 then PC ←PC + 2 (or 3)	Нет	1/2/3
SBIC	P, b 0≤P≤31 0≤b≤7	Пропустить, если бит в регистре I/O очищен	if P(b)=0 then PC ← PC + 2 (or 3)	Нет	1/2/3
SBIS	P, b 0≤r≤31 0≤b≤7	Пропустить, если бит в регистре I/O установлен	If P(b)=1 then PC ←PC + 2 (or 3)	Нет	1/2/3
BRBS	s, k 0≤s≤7 -64≤k≤+63	Перейти, если бит в регистре статуса установлен	if SREG(s)=1 then PC ← PC + k + 1	Нет	1/2
BRBC	s, k 0≤s≤7 -64≤k≤+63	Перейти, если бит в регистре статуса очищен	if SREG(s)=0 then PC ← PC + k + 1	Нет	1/2
BREQ	k -64≤k≤+63	Перейти, если равно	if Rd=Rr (Z=1) then PC ← PC + k + 1	Нет	1/2
BRNE	k -64≤k≤+63	Перейти, если не равно	if Rd≠Rr(Z=0) then PC<←PC+ k+ 1	Нет	1/2
BRCS	k -64≤k≤+63	Перейти, если флаг переноса установлен	if C=1 then PC ← PC + k + 1	Нет	1/2
BRCC	k -64≤k≤+63	Перейти, если флаг переноса очищен	if C=0 then PC ← PC + k + 1	Нет	1/2
BRSH	k -64≤k≤+63	Перейти, если равно или больше (без знака)	if Rd<Rr(C=0) then PC←PC+ k+ 1	Нет	1/2
BRLO	k -64≤k≤+63	Перейти, если меньше (без знака)	if Rd<Rr (C=1) then PC ←PC + k + 1	Нет	1/2

Мнемоника	Операнды	Описание	Операция	Флаги	Кол-во циклов
BRMI	k -64≤k≤+63	Перейти, если минус	if N=1 then PC ← PC + k + 1	Нет	1/2
BRPL	k -64≤k≤+63	Перейти, если плюс	if N=0 then PC ← PC + k + 1	Нет	1/2
BRGE	k -64≤k≤+63	Перейти, если больше или равно (со знаком)	if Rd>Rr (N⊕V=0) then PC←PC+k+1	Нет	1/2
BRLT	k -64≤k≤+63	Перейти, если меньше чем (со знаком)	if Rd<Rr (N⊕V=1) then PC←PC+k+1	Нет	1/2
BRHS	k -64≤k≤+63	Перейти, если флаг полупереноса установлен	if H=1 then PC ← PC + k + 1	Нет	1/2
BRHC	k -64≤k≤+63	Перейти, если флаг полупереноса очищен	if H=0 then PC ← PC + k + 1	Нет	1/2
BRTS	k -64≤k≤+63	Перейти, если флаг T установлен	if T=1 then PC ← PC + k + 1	Нет	1/2
BRTC	k -64≤k≤+63	Перейти, если флаг T очищен	if T=0 then PC ← PC + k + 1	Нет	1/2
BRVS	k -64≤k≤+63	Перейти, если флаг переполнения установлен	if V=1 then PC ← PC + k + 1	Нет	1/2
BRVC	k -64≤k≤+63	Перейти, если флаг переполнения очищен	if V=0 then PC ← PC + k + 1	Нет	1/2
BRIE	k -64≤k≤+63	Перейти, если глобальное прерывание разрешено	if I=1 then PC ← PC + k + 1	Нет	1/2
BRID	k -64k≤k≤+63	Перейти, если глобальное прерывание запрещено	if I=0 then PC ← PC + k + 1	Нет	1/2