Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет
им. Г.И. Носова»
Многопрофильный колледж

МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ И ПРАКТИЧЕСКИХ РАБОТ

ПМ.02 Применение микропроцессорных систем, установка и настройка периферийного оборудования
МДК.02.01 Микропроцессорные системы
для студентов специальности
09.02.01. Компьютерные системы и комплексы
базовой подготовки
Часть 2

Магнитогорск, 2017

ОДОБРЕНО:
Предметно-цикловой комиссией
«Информатики и вычислительной техники»
Председатель И.Г. Зорина
Протокол № 7 от 14 марта 2017 г.

Методической комиссией МпК
Протокол №4 от «23» марта 2017г

Составитель:
преподаватель МпК ФГБОУ ВО «МГТУ им. Г.И. Носова Татьяна Борисовна Ремез

Методические указания по выполнению лабораторных и практических работ разработаны на основе рабочей программы ПМ.02 «Применение микропроцессорных систем, установка и настройка периферийного оборудования»
Содержание лабораторных и практических работ ориентировано на формирование общих и профессиональных компетенций по основной профессиональной образовательной программе по специальности 09.02.01. «Компьютерные системы и комплексы» базовой подготовки: МДК.02.01 Микропроцессорные системы

СОДЕРЖАНИЕ

1 ВВЕДЕНИЕ

Важную часть теоретической и профессиональной практической подготовки студентов составляют практические и лабораторные занятия.

Состав и содержание практических и лабораторных работ направлены на реализацию действующего федерального государственного образовательного стандарта среднего профессионального образования.

Ведущей дидактической целью практических занятий является формирование практических умений - профессиональных (умений выполнять определенные действия, операции, необходимые в последующем в профессиональной деятельности), необходимых в последующей учебной деятельности по профессиональным модулям.

Ведущей дидактической целью лабораторных работ является экспериментальное подтверждение и проверка существенных теоретических положений (законов, зависимостей).

В соответствии с рабочей ПМ.02 «Применение микропроцессорных систем, установка и настройка периферийного оборудования», МДК.02.01 «Микропроцессорные системы» предусмотрено проведение лабораторных работ и практических занятий.

В результате их выполнения, обучающийся должен:

уметь:

- составлять программы на языке ассемблера для микропроцессорных систем;
- производить тестирование и отладку МПС;
- выбирать микроконтроллер/микропроцессор для конкретной системы управления

Содержание лабораторных работ ориентировано на формирование общих компетенций по профессиональному модулю основной профессиональной образовательной программы по специальности:

ОК 1. Понимать сущность и социальную значимость своей будущей профессии, проявлять к ней устойчивый интерес

ОК 2. Организовывать собственную деятельность, определять методы и способы выполнения профессиональных задач, оценивать их эффективность и качество

ОК 3. Принимать решения в стандартных и нестандартных ситуациях и нести за них ответственность

ОК 4. Осуществлять поиск и использование информации, необходимой для эффективного выполнения профессиональных задач, профессионального и личностного развития

ОК 5. Использовать информационно-коммуникационные технологии в профессиональной деятельности

ОК 6. Работать в коллективе и команде, эффективно общаться с коллегами, руководством, потребителями

ОК 7. Брать на себя ответственность за работу членов команды (подчиненных), результат выполнения заданий

ОК 8. Самостоятельно определять задачи профессионального и личностного развития, заниматься самообразованием, осознанно планировать повышение квалификации

ОК 9. Ориентироваться в условиях частой смены технологий в профессиональной деятельности

И овладению профессиональными компетенциями:

ПК 2.1. Создавать программы на языке ассемблера для микропроцессорных систем.

ПК 2.2. Производить тестирование, определение параметров и отладку микропроцессорных систем.

Выполнение студентами лабораторных работ по ПМ.02 «Применение микропроцессорных систем, установка и настройка периферийного оборудования», МДК.02.01 «Микропроцессорные системы» направлено на:

- обобщение, систематизацию, углубление, закрепление, развитие и детализацию полученных теоретических знаний по конкретным темам междисциплинарных курсов;
- формирование умений применять полученные знания на практике, реализацию единства интеллектуальной и практической деятельности;
- формирование и развитие умений: наблюдать, сравнивать, сопоставлять, анализировать, делать выводы и обобщения, самостоятельно вести исследования, пользоваться различными приемами измерений, оформлять результаты в виде таблиц, схем, графиков;
- приобретение навыков работы с различными приборами, аппаратурой, установками и другими техническими средствами для проведения опытов;
- развитие интеллектуальных умений у будущих специалистов: аналитических, проектировочных, конструктивных и др.;
- выработку при решении поставленных задач профессионально значимых качеств, таких как самостоятельность, ответственность, точность, творческая инициатива.

Практические и лабораторные занятия проводятся после соответствующей темы, которая обеспечивает наличие знаний, необходимых для ее выполнения.

Продолжительность выполнения практической или лабораторной работы составляет не менее двух академических часов (от 2 до 6) и проводится после соответствующего занятия, которое обеспечивает наличие знаний, необходимых для ее выполнения.

2 МЕТОДИЧЕСКИЕ УКАЗАНИЯ

ОПИСАНИЕ ЛАБОРАТОРНОГО СТЕНДА И ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТ

Лабораторные работы выполняются на лабораторном стенде - модуль микроконтроллера. Внешний вид модуля приведен на рис. 1.

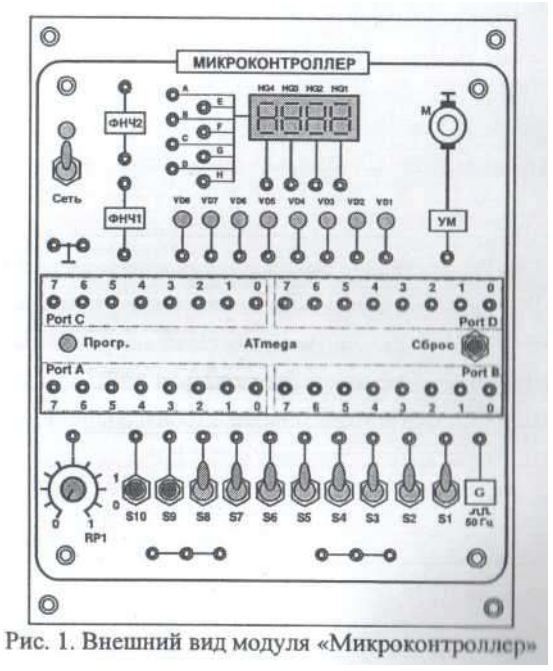


Рис. 1. Внешний вид модуля «Микроконтроллер»

Модуль «Микроконтроллер» предназначен для программирования и изучения функций микроконтроллера ATmega8535 семейства AVR, выпускаемого фирмой Atmel.

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 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 дляподачи на него управляющего напряжения;

- семисегментныйчетырехсимвольный 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.

Основные характеристики изучаемого микроконтроллера ATmega8535- приведены в табл. 1.

Таблица 1. Краткая характеристика микроконтроллера ATmega8535

№	Параметр	Значение
1	Объем памяти программ (Flash-память)	8 кБ
2	Объем энергонезависимой памяти (память	512 Б
3	Объем ОЗУ	512 Б
4	Количество программируемых входов/выходов	32
5	8-битные таймеры/счетчики с ШИМ	2 шт.
6	16-битный таймер/счетчик с ШИМ	1 шт.
7	Количество каналов АЦП	8
8	Разрядность АЦП	10
9	Аналоговый компаратор	есть
10	Источники внешних прерываний	3 шт.
11	Универсальный приемопередатчик USART	есть
12	TWI - интерфейс	есть
13	SPI - интерфейс	есть
14	JTAG - интерфейс	нет
15	Напряжение электропитания	2,7 - 5,5 В
16	Диапазон частот кварцевого резонатора	0... 16 МГц
17	Частота установленного кварцевого резонатора	8 МГц

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 работоспособностьпреподавателю на AVR-studio;
- произвести сборку схемы;
- *только после разрешения* преподавателя включить питание и приступить к программированию микроконтроллера и проверки его работы;
- представить программу на микроконтроллере 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 для решения задачи периферийных устройств, таймеров, прерываний, АЦП и т.п.;
   - расчеты требуемых характеристик работы периферийных устройств (периодов работы таймеров, разрядности АЦП и т.д.);
   - настройка регистров ввода/вывода для задания требуемого режима работыпериферийных устройств.
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 воFLASH-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.

Примечание: при составлении отчета необходимо выполнять требования ЕСКД: использовать пунктуацию, записывать название выполняемого пункта, схемы и таблицы должны быть пронумерованы и аккуратно построены.

Тема 1.8 Программирование микроконтроллеров
Практическая работа № 7
Изучение ассемблера МК AVR

Формируемые компетенции:

ПК 2.1. Создавать программы на языке ассемблера для микропроцессорных систем.

Цель работы: изучить основные понятия ассемблера МК AVR

Выполнив работу, Вы будете:

уметь:

- составлять программы на языке ассемблера для микропроцессорных систем

Материальное обеспечение:

не требуется

Теоретические сведения

Основные понятия

Ассемблер - это инструмент, с помощью которого создаётся программа для микроконтроллера. Ассемблер транслирует ассемблируемый исходный код программы в объектный код, который может использоваться в симуляторах или эмуляторах AVR. Также ассемблер генерирует код, который может быть не посредственно введен в программную память микроконтроллера.

При работе с ассемблером нет никакой необходимости в непосредственном соединении с микроконтроллером.

Язык ассемблера - это символическое представление машинного языка. Все процессы в машине на самом низком, аппаратном уровне приводятся в действие только командами (инструкциями) машинного языка. Отсюда понятно, что, несмотря на общее название, язык ассемблера для каждого типа компьютера свой.

Программа на ассемблере представляет собой совокупность блоков памяти, называемых сегментами памяти. Программа может состоять из одного или нескольких таких блоков-сегментов. Каждый сегмент содержит совокупность предложений языка, каждое из которых занимает отдельную строку кода программы.

Предложения ассемблера бывают четырех типов:

1) команды или инструкции, представляющие собой символические аналоги машинных команд. В процессе трансляции инструкции ассемблера преобразуются в соответствующие команды системы команд микропроцессора;
2) макрокоманды - оформляемые определенным образом предложения текста программы, замещаемые во время трансляции другими предложениями;
3) директивы, являющиеся указанием транслятору ассемблера на выполнение некоторых действий. У директив нет аналогов в машинном представлении;
4) строки комментариев, содержащие любые символы, в том числе и буквы русского алфавита. Комментарии игнорируются транслятором.

Предложения, составляющие программу, могут представлять собой синтаксическую конструкцию, соответствующую команде, макрокоманде, директиве или комментарию. Для того чтобы транслятор ассемблера мог распознать их, они должны формироваться по определенным синтаксическим правилам. Для этого лучше всего использовать формальное описание синтаксиса языка наподобие правил грамматики.

Исходный файл, с которым работает ассемблер, должен содержать мнемоники, директивы и метки.

Мнемоника – это краткое буквенное обозначение команды:

СЛОЖЕНИЕ→ADDITION→ADD

Перед каждой строкой программы можно ставить метку, которая является алфавитно-цифровой строкой, заканчивающейся двоеточием. Метки используются какуказания для безусловного перехода и команд условного перехода.

Строка программы может быть в одной из четырёх форм:

[Метка:] директива [операнды] [Комментарий]
[Метка:] команда [операнды] [Комментарий]
Комментарий
Пустая строка

Символы квадратной скобки [...] означают, что использование элемента необязательно.

Практически каждое предложение содержит описание объекта, над которым или при помощи которого выполняется некоторое действие. Эти объекты называются *операндами* - это объекты (некоторые значения, регистры или ячейки памяти), на которые действуют инструкции или директивы, либо это объекты, которые определяют или уточняют действие инструкций или директив.

Комментарий имеет следующую форму:

; [Текст]

Таким образом любой текст после символа “ ; ” игнорируется ассемблером и имеет значение только для пользователя.

Операнды можно задавать в различных форматах:

десятичный: 10,255
шестнадцатеричный (два способа): 0x1a или $1a
двоичный: 0b00001010,0b11111111
восьмеричный: 010, 077

Символы языка ассемблера

Допустимыми символами при написании текста программ являются:

1) все латинские буквы: A-Z, a-z. При этом заглавные и строчные буквы считаются эквивалентными;
2) цифры от 0 до 9;
3) знаки ?, @, $, _, &;
4) разделители , . [ ] ( ) <> { } + / * % ! ' " ? = # ^.

Предложения ассемблера формируются из лексем, представляющих собой синтаксически неразделимые последовательности допустимых символов языка, имеющие смысл для транслятора.

Директивы ассемблера

Ассемблер поддерживает множество директив. Директивы не транслируются непосредственно в коды операции. Напротив, они используются, чтобы корректировать местоположение программы в памяти, определять макрокоманды, инициализировать память и так далее. То есть это указания самому ассемблеру, а не команды микроконтроллера.

Все директивы ассемблера приведены в табл. 1.

Таблица 1. Директивы ассемблера

Директива	Описание
BYTE	Зарезервировать байт под переменную
CSEG	Сегмент кодов
DB	Задать постоянным(и) байт(ы) в памяти
DEF	Задать символическое имя регистру
DEVICE	Задать для какого типа микроконтроллера компилировать
DSEG	Сегмент данных
DW	Задать постоянное(ые) слово(а) в памяти
EQU	Установите символ равный выражению
ESEG	Сегмент EEPROM
EXIT	Выход из файла
INCLUDE	Включить исходный код из другого файла
LIST	Включить генерацию .1st - файла
NOLIST	Выключить генерацию .1st - файла
ORG	Начальный адрес программы
SET	Установите символ равный выражению

Синтаксис всех директив следующий:

. <директива>

То есть перед директивой должна стоять точка. Иначе ассемблер воспринимает это как метку.

Поясним наиболее важные директивы ассемблера.

1. *CSEG- Codesegment* - указывает на начало сегмента кодов. Ассемблируемый файл может иметь несколько кодовых сегментов, которые будут объединены в один при ассемблировании.

Синтаксис:

• CSEG

Пример:

```
.DSEG                   ; Начало сегмента данных
vartab: .BYTE 4         ; Резервируется 4 байта в СОЗУ
.CSEG                   ; Начало сегмента кодов
const:  .DW 2           ; Записать 0x0002  в программной памяти
mov r1,r0               ; Что-то делать
```

2. *DSEG – DataSegment* - указывает на начало сегмента данных. Ассемблируемый файл может содержать несколько сегментов данных, которые потом будут собраныв один при ассемблировании. Обычно сегмент данных состоит лишь из директивыBYTEи меток.

Синтаксис:

• DSEG

Пример:

```
.DSEG                   ; Начало сегмента данных
var1: .BYTE 1           ; Резервировать 1 байт под переменную
table: .BYTE tab_size   ; Резервировать tab_size байтов.
.CSEG
ldi r30,low(var1)
ldi r31,high(var1)
ld r1,Z
```

3. *ESEG -EEPROMSegment* - указывает на начало сегмента EEPROMпамяти Ассемблируемый файл может содержать несколько EEPROMсегментов, которьи будут собраны в один сегмент при ассемблировании. Обычно сегмент EEPRON состоит из DBи DWдиректив (и меток). Сегмент EEPROMпамяти имеет свой

собственный счетчик. Директива ORGможет использоваться для размещена переменных в нужной области EEPROM.

Синтаксис:

• ESEG

Пример:

```
.DSEG                    ; Начало сегмента данных
var1:.BYTE 1             ; Резервировать 1 байт под переменную
table: .BYTE tab_size    ; Зарезервировать  tab_size байт.
.ESEG
eevar1: .DW 0xffff       ; Записать 1 слово в EEPROM
```

4. *ORG - установить адрес начала программы* -присваивает значения локальным счетчикам. Используется только совместно с директивами .CSEG, .DSEG, .ESEG.

Синтаксис:

• ORG<адрес>

Пример:

```
.DSEG                   ; Начало сегмента данных
.ORG 0x37               ; Установить адрес СОЗУ  на 37h

variable: .BYTE 1       ; Зарезервировать байт СОЗУ по адресу 37h
.CSEG
.ORG 0x10               ; Установить счетчик команд на адрес 10h
mov r0,r1               ; Чего-нибудь делать
```

5. *DB – определить байт(ы) в программной памяти или в EEPROM*- резервирует ресурсы памяти в программной памяти или в EEPROM. Директиве должна предшествовать метка. DBзадает список выражений, и должна содержать по крайней мере одно выражение. Размещать директиву следует в сегменте кодов или в EEPROMсегменте.Список выражений представляет собой последовательность выражений, разделенных запятыми. Каждое выражение должно быть величиной между -128 и 255. Если директива указывается в сегменте кодов и список выражений содержит более двух величин, то выражения будут записаны .так, что 2 байта будут размещаться в каждом слове Flash-памяти.

Синтаксис:

LABEL: .DB< список выражений>

Пример:

```
.CSEG
consts: .DB 0, 255, 0b01010101, -128, 0xaa
.ESEG
const2: .DB 1, 2, 3
```

6. *DW – определить слово в программной памяти или в EEPROM* - Директива DWрезервирует ресурсы памяти в программной памяти или в EEPROM. Директиве должна предшествовать метка. DWзадает список выражений, . и должна содержать по крайней мере одно выражение. Размещать директиву следует в сегменте кодов или в EEPROMсегменте. Список выражений представляет собой последовательность выражений, разделенных запятыми. Каждое выражение должно быть величиной между -32768 и 65535.

Синтаксис:

LABEL: .DB< список выражений>

Пример:

```
.CSEG
varlist:  .DW 0, 0xffff, 0b1001110001010101, -32768, 65535
.ESEG
eevarlst: .DW 0,0xffff,10
```

7. *DEF – присвоить имя регистру*- позволяет присвоить символическое имя регистру. Регистр может иметь несколько символических имен.

Синтаксис:

.DEF< Имя>-<Регистр>

Пример:

```
.DEF temp=R16
.DEF ior=R0
.CSEG
ldi temp,0xf0        ; Загрузить 0xf0 в регистр temp
in ior,0x3f          ; Прочитать SREG в регистр ior
eor temp, ior        ; Выполнить операцию
```

8. *EQU – присвоить имя выражению*- директива присваивает значение метке. Эта метка может быть использована в других выражениях. Значение этой метки нельзя изменить или переопределить.

Синтаксис:

.EQU< метка>=<выражение>

Пример:

```
.EQU io_offset = 0x23
.EQU porta     = io_offset + 2
.CSEG                          ; Начало сегмента кодов
clr r2                         ; Очистить регистр r2
out porta,r2                   ; Записать в порт A
```

9. *INCLUDE – вставить другой файл* - директива говорит Ассемблеру начать читать из другого файл. Ассемблер будет ассемблировать этот файл до конца файла или до директивы EXIT. Включаемый файл может сам включать директивы INCLUDE.

Синтаксис:

.INCLUDE"<имя файла>"

Пример:

```
.EQU sreg = 0x3f    ; Регистр статуса
.EQU sphigh = 0x3e  ; Старший байт указателя стека.
.EQU splow  = 0x3d  ; Младший байт указателя стека.
; incdemo.asm
.INCLUDE "iodefs.asm"; Включить файл «iodefs.asm»
in r0,sreg          ; Прочитать регистр статуса
```

10. *EXIT – выйти из файла* – директива позволяет ассемблеру остановить ассемблирование текущего файла. Обычно ассемблер работает до конца файла. Если он встретит директиву EXIT, топродолжит ассемблировать со строки, следующей за директивой INCLUDE.

Синтаксис:

.EXIT

Пример:

```
.EXIT                ; выйти из этого файла
```

11. *DEVICE – указать для какого микроконтроллера ассемблировать* - директива позволяет пользователю сообщить ассемблеру, для какого типа устройства пишется программа. Если ассемблер встретит команду, которая не поддерживается указанным типом микроконтроллера, то будет выдано сообщение. Также сообщение появится в случае, если размер программы превысит объем имеющейся в этом устройстве памяти.

Синтаксис:

```
.DEVICE AT90S1200 |AT90S2313 | AT90S2323 | AT90S2333 | AT90S2343 |
AT90S4414 | AT90S4433 | AT90S4434 | AT90S8515 | AT90S8534 |
AT90S8535 | ATtiny11 | ATtiny12 | ATtiny22 | ATmega603 |
ATmega103| Atmega8535
```

Пример:

```
.DEVICE ATmega8535  ;использовать ATmega8535
.CSEG
.ORG 0000
jmp label1          ;При ассемблировании появиться сообщение, что
                    ;ATmega8535 не поддерживает команду jmp
```

Задание

1. Рассмотрите пример приведенной программы на ассемблере, определите основные использованные директивы. В тетради опишите их назначение, приведите примеры комментариев и операндов.Приведите примеры строк из программы, которые соответствуют формам:

   [Метка:] директива [операнды] [Комментарий],

   [Метка:] команда [операнды] [Комментарий].

   Обозначьте все элементы (метки, директивы, команды (мнемоника), операнды, комментарии).

   Программа, написанная на ассемблере, должна иметь определенную структуру. Предлагается следующий шаблон (для Atmega 8535):

```
;-----------------------------------------------;
; название программы,                           ;
; краткое описание, необходимые пояснения       ;
;-----------------------------------------------;
;       подключаемые дополнительные файлы
.include "m8535def.inc"           ; файл описания ATmega8535
.include "<имя_файла1.расширение>"; включение дополнительных
.include "<имя_файла2.расширение>"; файлов
;       глобальные константы
.equ      <имя1> = <константа1>
.equ      <имя2> = <константа2>
;       глобальные регистровые переменные
.def      <имя1>= <регистр1>
.def      <имя2>= <регистр2>
;       сегмент данных
.dseg
.org <адрес>          ; адрес первого зарезервированного байта
label1:  .BYTE 1      ; резервировать 1 байт под переменную label1
label2:  .BYTE m      ; резервировать m байт под переменную label2
;       сегмент EEPROM (ЭСППЗУ)
```

```
.eseg
.org <адрес>          ; адрес первого зарезервированного байта
.db  выражение1,выражение2,…  ; записать список байтов в EEPROM
.dw  выражение1,выражение2,…  ; записать список слов в EEPROM
;       сегмент кодов
.cseg
.org $0000            ; адрес начала программы в программной памяти
;       вектора прерываний (если они используются)
rjmp reset            ;прерывание по сбросу
.org $0002
rjmp INT0             ;обработчик прерывания INT0
.org $0004
rjmp INT1             ;обработчик прерывания INT1
.org adrINTx          ;адрес следующего обработчика прерываний
rjmp  INTx            ;обработчик прерывания x
……                    ;далее располагаются обработчики остальных
                      ;прерываний
;       начало основной программы
main:     <команда> xxxx
          …   …
;       подпрограмма 1
subr1: <команда> xxxx
……       ……        ……
          ret
;       подпрограмма 2
subr2: <команда> xxxx
……       ……        ……
          ret
;       программы обработчиков прерываний
INT0: <команда> xxxx
……       ……        ……
          reti
……
;       конец программы никак не обозначается
```

Контрольные вопросы

1. Ассемблер – это…
2. Директива – это…
3. Мнемоника – это…
4. В каких форматах можно задавать операнды в ассемблере?
5. Каково назначение комментариев в программе?
6. Каково назначение директивы CSEG?
7. Каково назначение директивы DB?
8. Каково назначение директивы DW?
9. Каково назначение директивы EXIT?
10. Каково назначение директивы DEVICE?

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Основные понятия и определения
3. Выполненное задание
4. Ответы на контрольные вопросы
5. Вывод по работе

Практическая работа № 8
Изучение системы команд МК AVR

Формируемые компетенции:

ПК 2.1. Создавать программы на языке ассемблера для микропроцессорных систем.

Цель работы: изучить основные команды ассемблера МК AVR

Выполнив работу, Вы будете:

уметь:

- составлять программы на языке ассемблера для микропроцессорных систем

Материальное обеспечение:

не требуется

Теоретические сведения

Система команд микроконтроллеров ATMELсемейства AVRочень большая и в тоже время эффективная. Одной из отличительных особенностей микроконтроллеров AVRявляется то, что почти все команды выполняются за 1 тактовыйцикл. Исключение составляют команды перехода. Это существенно увеличивает производительность микроконтроллера даже при относительно невысокой тактовой частоте.

Все команды можно классифицировать на 5 типов:

1. Арифметические команды.
2. Логические команды.
3. Команды перехода.
4. Команды передачи данных.
5. Побитовые команды и команды тестирования битов.

Система команд приведена **ПРИЛОЖЕНИИ 1**.

Некоторые особенности программирования

Память данных почти полностью доступна программе пользователя и большинство команд ассемблера предназначено для обмена данными с ней. Команды пересылки данных предоставляют возможность непосредственной и косвенной адресации ячеек СОЗУ, непосредственной адресации регистров ввода/вывода и регистров общего назначения. Так как каждому регистру сопоставлена ячейка памяти, то обращаться к ним можно не только командами адресации регистров, но и командами адресации ячеек СОЗУ.

Пример, команда:

```
MOV R10,R15    ; скопировать регистр R15 в регистр R10 делает
               ; абсолютно то же самое, что и команда:
LDS R10,$0015  ; загрузить в регистр R10 содержимое ячейки.с
               ; адресом $0015
```

То же самое относится и к регистрам ввода/вывода. Для них предусмотрены специальные команды:

При использовании этих команд номер порта указывается в диапазоне 0<P<63. При использовании команд

```
IN Rd,P        ; загрузить данные из порта I/O с номером P
               ; в регистр Rd
OUT P,Rd       ; записать данные из регистра Rd в порт I/O
               ; с номером P.
```

адресации ячеек памяти для работы с регистрами ввода/вывода указывается адрес регистра в памяти данных $0020-$005F.

Пример применения разных команд:

```
LDI R16,$FF
OUT $12,R16    ; записать в PORTD число 255
STS $0032,R16  ; записать непосредственно в ячейку $0032
```

Адрес регистра ввода/вывода в СОЗУ получается прибавлением к номеру порта числа $20.

Памятью программ является ПЗУ и изменяется только при программировании кристалла. Константы можно располагать в памяти программ в виде слов.

Пример: .dw$033f,$676d,$7653,$237e,$777f

Дли работы с данными, расположенными в памяти программ, предусмотрена командаLPM- загрузить байт памяти программ, на который указывает регистр Zв регистрR0.

Адрес байта константы определяется содержимым регистра Z. Старшие 15 битов определяют слово адреса (от 0 до 4к) состояние младшего бита определяет выбор младшего байта (0) или старшего байта (1).

При работе с портами ввода/вывода следует учитывать следующую особенность: если вывод порта сконфигурирован как выход, то его переключение производится через регистр данных (PORTA, PORTB, PORTC, PORTD), если вывод сконфигурирован как вход, то его опрос следует производить через регистр выводов входа порта (PINA, PINB, PINC, PIND).

Особенностью использования арифметических и логических команд является то, что некоторые из них работают только с регистрами R16-R31.

Пример:

CPIRd, к — сравнить регистр Rdc константой К, 16<d<31.

Команды CBIи SBIработают только с младшими 32-мя регистрами ввода/вывода.

При использовании подпрограмм нужно обязательно определять стек! Для этого нужно занести значения адреса вершины стека в регистры SPHи SPL.

Задание

Рассмотрите пример приведенной программы на ассемблере, определите основные использованные директивы и команды. В тетради опишите их назначение (используйте ПРИЛОЖЕНИЕ 1). Приведите примеры строк из

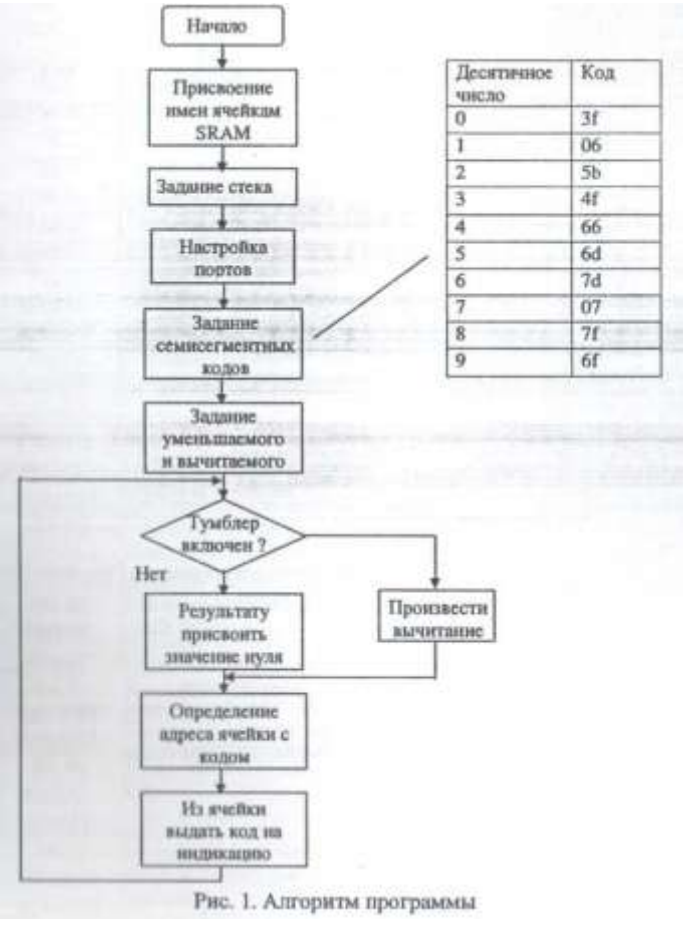
программы, которые соответствуют формам:

[Метка:] директива [операнды] [Комментарий],

[Метка:] команда [операнды] [Комментарий].

Обозначьте все элементы (метки, директивы, команды (мнемоника), операнды, комментарии).

Простейшая программа демонстрирует использование директив ассемблера. Программа вычитает из числа 5 число 3. Если подан сигнал разрешения («1» на вход PA4), то на индикацию (сегменты - биты PC0...PC7, индикатор - бит PB3) выдать результат вычитания. Если нет разрешения, то на индикацию вывести цифру ноль. На рис. 1 приведен алгоритм программы



Десятичное число	Код
0	3f
1	06
2	5b
3	4f
4	66
5	6d
6	7d
7	07
8	7f
9	6f

Рис. 1. Алгоритм программы

Листинг программы

```
; Программа №1
.include "m8535def.inc"  ;включить файл - описание для ATmega8535
.dseg                    ;сегмент данных
.equ      cod0=$64       ;присвоение имен ячейкам SRAM
.equ      cod1=$65
.equ      cod2=$66
.equ      cod3=$67
.equ      cod4=$68
.equ      cod5=$69
.equ      cod6=$6a
.equ      cod7=$6b
.equ      cod8=$6c
.equ      cod9=$6d
.cseg
.org      0
rjmp      reset
.org  $30           ;начало программы
reset:
ldi  r16,$00        ;определение стека с вершиной по адресу $00ff
out  sph,r16
ldi  r16,$ff
out  spl,r16
```

```
ldi  z1,$64          ;задание адреса начала зарезервированных ячеек
ldi  zh,$00

ldi  r16,$ff         ;настроить порт C на выход
out  ddrc,r16
ldi  r16,00          ;настроить порт A на вход
out  ddra,r16
ldi  r16,$c          ;настроить порт B: биты 2 и 3 на выход, остальные на вход
out  ddrb,r16
ldi  r16,$f0         ;настроить порт D: биты 0...4 на вход, остальные на выход
out  ddrd,r16

sbi  portB,3         ;выдать 1 на разряд 3 порта B
ldi  r17,$3f         ;задание семисегментных кодов
sts  cod0,r17
ldi  r17,$06
sts  cod1,r17
ldi  r17,$5b
sts  cod2,r17
ldi  r17,$4f
sts  cod3,r17
ldi  r17,$66
sts  cod4,r17
ldi  r17,$6d
sts  cod5,r17
ldi  r17,$7d
sts  cod6,r17
ldi  r17,$07
sts  cod7,r17
ldi  r17,$7f
sts  cod8,r17
ldi  r17,$6f
sts  cod9,r17

ldi  r17,5           ;задание уменьшаемого
ldi  r18,3           ;задание вычитаемого
m1:  sbis pina,4     ;если включен тумблер SA1,то пропустить
rjmp      m2         ;следующую команду
mov  r20,r17         ; в r20 поместить уменьшаемое
sub  r20,r18         ; вычесть вычитаемое
rjmp      vv
m2:  ldi  r20,0
vv:  push z1         ;сохранить z1 в стеке
add  z1,r20          ;сложить z1 с результатом
ld   r0,z            ;семисегментный код результата переслать в r20
pop   z1             ;извлечь z1 из стека
out  portc,r0        ;выдать результат на индикацию
rjmp      m1
```

Контрольные вопросы

Приведите примеры команд (если нет в программе, то из ПРИЛОЖЕНИЯ 1):

1. Арифметические команды.
2. Логические команды.
3. Команды перехода.
4. Команды передачи данных.
5. Побитовые команды и команды тестирования битов.

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Основные понятия и определения
3. Выполненное задание
4. Ответы на контрольные вопросы
5. Вывод по работе

Практическая работа № 9
Изучение работы AVRStudio

Формируемые компетенции:

ПК 2.1. Создавать программы на языке ассемблера для микропроцессорных систем.

Цель работы: изучить основные функции по созданию программ на ассемблере в среде программирования и отладкиAVRStudio.

Выполнив работу, Вы будете:

уметь:

- составлять программы на языке ассемблера для микропроцессорных систем

Материальное обеспечение:

ПК с установленной средой программирования и отладкиAVRStudio

Теоретические сведения

Т.к. микроконтроллер ATmega8535 является программируемым, пользовательдолжен освоить его программирование в различных программных средах и в различных языках высокого и низкого уровня.

Среди наиболее популярных методов написания программ можно выделить:

-написание программ на машинном коде микроконтроллера. Программы написанные таким способом, являются наиболее быстродействующими, однако для их написания требуется высокая квалификация программиста и глубокое знание архитектуры процессора;

-написание программы на ассемблере. Написание программ на ассемблере существенно проще, чем на машинном коде, однако также требует высокой квалификации программиста. Следует отметить, что язык ассемблера обычножестко привязан к конкретному типу микропроцессора и может существенноотличаться для разных микропроцессоров одного производителя;

-написание программы на языке высокого уровня (например, Си). Такие программы обычно являются кросс-платформенными, то есть практически не отличаются по синтаксису для микропроцессоров разных типов. Пользователь пишет программу на языке высокого уровня, а компилятор преобразует ее в ассемблер и машинный код конкретного микропроцессора. Однако обычно использование языка высокого уровня при написании программ для микроконтроллеров может существенно ограничить их быстродействие.

Создание проекта в среде AVRStudio

Для написания программ**,** необходимо создать проект.

С лабораторным комплексом поставляется программа AVRStudiover. 4. Действия по созданию проекта и работе с программой в разных версиях AVRStudiomогут несколько отличаться, однако большинство действий соответствуют изложенным далее пунктам.

1. Запустить программу AVRStudio. Ярлык для запуска программы находится на рабочем столе или в меню «Пуск» Windows. В появившемся окне приветствия программы будет предложено создать новый проект или открыть существующий (рис. 1).

2. При выборе нового проекта появляется окно, в котором предлагается выбрать язык программы AtmelAvrAssemblerили AvrGCC.Здесь необходимо выбрать AtmelAvrAssembler, указать имя проекта и имя инициализационного файла с расширением *.asm. Рекомендуется, чтобы имена проекта и инициализационного файла совпадали.

Очень важно: не допускать в имени файла, проекта или пути символов кириллицы. После этого следует нажать кнопку «Next» (Далее).

Рис. 1. Окно приветствия AVR Studio

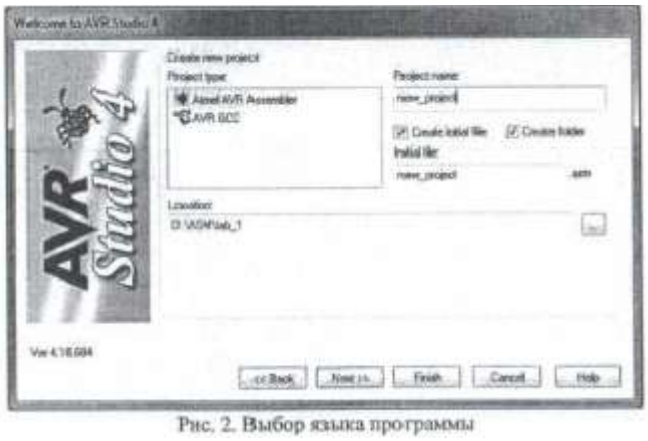


Рис. 2. Выбор языка программы

3. После выбора языка программы и имени проекта появляется окно выбора платформы (Debugplatform) и устройства (Device). Здесь необходимо выбратьAVRSimulatori процессор ATmega8535, который используется в лабораторном стенде. После этого следует нажать кнопку «Finish» (рис. 3).

4. При нажатии на кнопку «Finish» в программе открывается рабочее окно программирования (рис.4)

Рабочее окно содержит несколько областей:

- окно проекта «Project». Здесь отображается структура проекта, которая содержит его имя и список подключенных файлов и библиотек (рис.4 окно 1);
- центральная рабочая область, в которой осуществляется непосредственно набор программы (окно «2»);
- окно текущих сообщений «built» программы (окно «3»);
- окно регистров микроконтроллера «I/O View» (окно «4»),

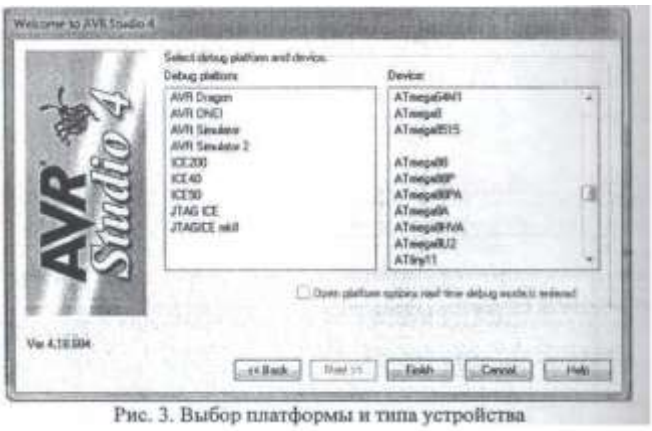


Рис. 3. Выбор платформы и типа устройства

Рис. 4. Рабочее окно программы AVR Studio

Задание.

Написать программу, осуществляющую вывод числа 01010101 или 10101010 на PORTCмикроконтроллера, в зависимости от состояния сигнала на входе PORTB0.

После набора программы необходимо произвести ее ассемблирование, то есть сборку, при которой ассемблер производит проверку синтаксиса написанной программы и при отсутствии ошибок преобразует код ассемблера в машинный код микропроцессора. При этом формируется файл с расширением *.hex, которыйдалее записывается в микроконтроллер.

Для ассемблирования написанной программы необходимо нажать кнопку F7 «Assemble». При отсутствии ошибок в окне текущих сообщений «built» AVRStudioотображается сообщение об успешном завершении операции, а при наличии ошибок выводится их список и положение в тексте программы.

Листинг программы

```
;---------------------------------------------------------------
;Пробная программа для микроконтроллера ATMEGA8535
;Входы: PORTB0
;Выходы: PORTC0...PORTC7

.include "m8535def.inc"  ;подключение библиотеки контроллера
.cseg                    ;начало сегмента кода
.org 0
reset:
   ldi r16,0xFF
   out DDRC,r16      ;назначение PORTC на вывод
   clr r16
   out DDRB,r16      ;назначение PORTB на ввод
main:
   sbis PINB,0       ;если на PINB0=0, то
   rjmp PINB0_is_0   ;переход на "PINB0_is_0"
   ldi r16,0xAA      ;иначе вывод на PORTC числа 0xAA (1010 1010)
   out PORTC,r16
   rjmp main         ;далее - возврат на main
PINB0_is_0:          ;PINB0=0
   ldi r16,0x55      ;вывод на PORTC числа 0x55 (0101 0101)
   out PORTC,r16
   rjmp main         ;далее - возврат на main
;---------------------------------------------------------------
```

Контрольные вопросы

1. Какие директивы использовались в программе?
2. Каково их назначение?
3. Какие команды использовались в программе?
4. К каким типам команд они относятся?
5. Приведите примеры операндов, использованных в программе.

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Основные понятия и определения
3. Выполненное задание
4. Ответы на контрольные вопросы
5. Вывод по работе

Лабораторная работа № 7
Работа в среде программирования и отладкиAVRStudio.

Формируемые компетенции:

ПК 2.1. Создавать программы на языке ассемблера для микропроцессорных систем.

Цель работы: изучить процесс создания и отладки программ на ассемблере в среде AVRStudio.

Выполнив работу, Вы будете:

уметь:

- составлять программы на языке ассемблера для микропроцессорных систем

Материальное обеспечение:

Стенд «Микроконтроллер», ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

В программное обеспечение AVRStudioвстроен симулятор микроконтроллеров, с помощью которого можно отладить программу, найти вней ошибки и исправить неточности в алгоритме, не осуществляя непосредственно программирования микросхемы контроллера. Также симулятор может быть полезен при написании программ в домашних условиях.

Симулятор AVRStudioосуществляет симуляцию микроконтроллера, егопортов, таймеров, АЦП, прерываний и т.д. Поскольку симулятор работает с hexфайлом проекта, то для запуска эмулятора необходимо написать программу исключить из нее явные ошибки, с которыми создание hex-файла невозможно.

Для отладки программы в симуляторе необходимо после ее написаниянажать сочетание клавиш Ctrl+F7 «AssembleandRun», после чего будет произведено ассемблирование программы и запуск эмулятора.

После успешной компиляции начало программы будет отмечено желтой стрелкой. Для управления процессами отладки программы в строке меню AVRStudioрасполагается меню «debug», а также панель функциональных кнопок управления симулятором, назначение которых приведено в табл. 1.

Таблица 1. Назначение кнопок управления эмулятором

Название	Сочетание клавиш	Назначение
Start Debugging (начать отладку)	Ctrl+Shift+Alt+F5	Начать процесс отладки программыв эмуляторе.
Stop Debugging (остановить отладку)	Ctrl+Shift+F5	Остановить процесс отладки программы и эмуляторе.
Run (запуск эмуляции)	F5	Эмулятор запускается и циклически производит эмуляцию программы без отображения текущих изменений регистров контроллера на экране.
Break (приостановить эмуляцию)	Ctrl+F5	Команда временно приостанавливал эмулятор без потери данных.
Reset (остановить эмуляцию)	Shift+F5	Команда полностью останавливает эмулятор с потерей данных эмуляции.
Stepinto (Сделать один шаг вперед)	Fit	Команда извлекает только одну инструкцию. После ее завершения вес рабочие экраны эмулятора обновляются.
Auto Step (Автовыполнение)	Alt+F5	Команда выполняет эмуляцию программы в пошаговом автоматическом режиме с оперативным обновлением информации на экранах эмулятора после каждого шага.
Toggle breakpoint (Точкаостановки)	F9	Команда добавляет или убирает пользовательскую точку остановки, при достижении которой программа будет приостановлена.

При симуляции программы рекомендуется выполнять пошаговое выполнение инструкций, пользуясь командой Stepinto(табл. 1). При этом необходимо контролировать содержимое регистров микроконтроллера, а также портов ввода/вывода.

Контроль состояния регистров и отдельных устройств микроконтроллера осуществляется в окне «4» (рис. 4). Каждое устройство можно развернуть и увидеть содержимое его регистров управления и контроля (рис. 5).

Запись программы в контроллер и проверка ее работоспособности

После успешной отладки программы необходимо «прошить» ее в микроконтроллер (т.е. записать ее в память микроконтроллера) и проверить работу.

Для «прошивки» программы используется та же программа AVRStudio.

Задание

Используя теоретические сведения и файлы с текстом пробной программы (Практическая работа №9) выполнить отладку программы и записать ее в микроконтроллер.

Последовательность действий с использованием AVRStudioследующая:

-включить переключатель «Сеть» модуля «Микроконтроллер» для подачи на него напряжения питания;

-в меню «Tools» AVRStudioвыбрать пункт «ProgramAVR», в котором указать способ соединения «AutoConnect». При правильном подключении персональному компьютеру модуль микроконтроллера инициализируется на USBCOM-порт. Необходимо, чтобы номер этого порта был не более COM9 противном случае необходимо найти этот порт в диспетчере оборудования Windowsи переназначить номер порта;

-если номер USBCOM-порта соответствует указанным требованиям, топроисходит подключение микроконтроллера к среде AVR-Studioи появляется и окно программирования контроллера (рис. 6);

-в окне программирования необходимо выбрать вкладку main, в которой необходимо выбрать тип используемого контроллера, после чего перейти вовкладку «Program», в которой выбрать графу «Flash». В этой графе требуется указать путь к *.hexфайлу проекта, после чего произвести запись программы в микроконтроллер нажатием кнопки «Program».

По завершении записи программы необходимо проверить ее корректную работу на микроконтроллере и показать результат преподавателю.

Рис. 5. Контроль состояния регистров микроконтроллера при симуляции

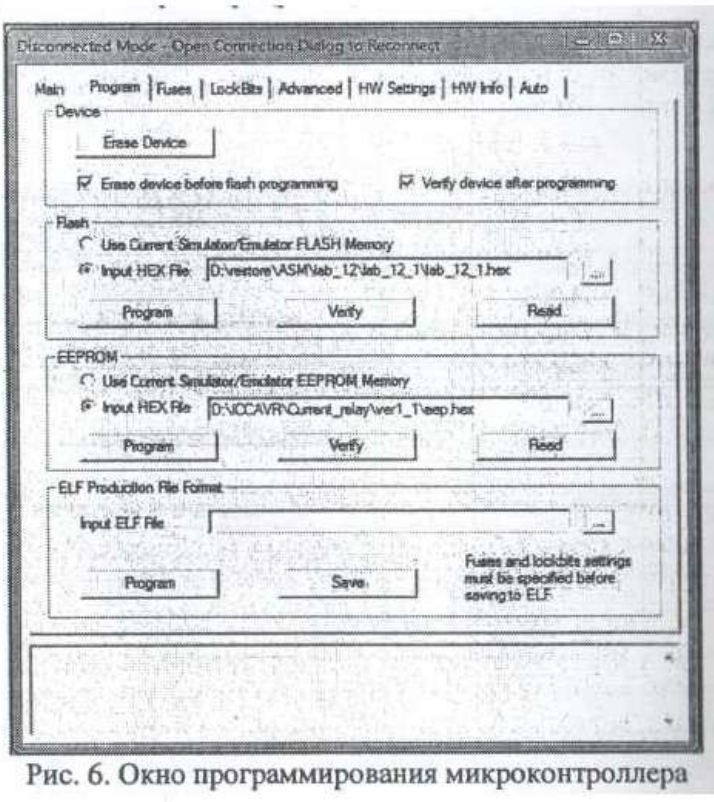


Рис. 6. Окно программирования микроконтроллера

Контрольные вопросы

1. Каково назначение AVR Studio?
2. Какой порт используется для программирования микроконтроллера?
3. Какие порты микроконтроллера использовались в программе и для чего?
4. Файл с каким расширением записывается в микроконтроллер?
5. В каком коде записывались числа 01010101 и 10101010 в программе?

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Выполненное задание
3. Ответы на контрольные вопросы
4. Вывод по работе

Практическая работа № 10
Изучение устройства параллельных портов МК Atmega8535.

Формируемые компетенции:

ПК 2.1. Создавать программы на языке ассемблера для микропроцессорных систем.

Цель работы: изучить принцип работы параллельных портов МК.

Выполнив работу, Вы будете:

уметь:

- составлять программы на языке ассемблера для микропроцессорных систем

Материальное обеспечение:

ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

Порты ввода/вывода.микроконтроллера предназначены для передачи и приема информации и последующей ее обработки. Микроконтроллеры различных типов-содержат различное количество портов ввода/вывода. Микроконтроллер Atmega8535 содержит 4 порта ввода/вывода, каждый из которых содержит 8 разрядов: PORTA, PORTB, PORTC, PORTD.

Порты ввода/вывода непосредственно связаны с выводами микросхемы микроконтроллера, при этом каждый конкретный вывод микроконтроллера жестко «привязан» к конкретному разряду порта ввода/вывода. На рис. 1

представлено расположение выводов микроконтроллера Atmega8535, выполненного в DIP-корпусе.

По умолчанию выводы микросхемы контроллера предназначены для выполнения функций ввода/вывода информации в соответствии с настройками регистров портов ввода/вывода. Однако функции большинства выводов микросхемы могут быть программно изменены. При этом к выводам микросхемы могут быть присоединены выходы таймеров, приемопередатчиков, входы аналогово-цифрового преобразователя, контроллера внешних прерываний.

Альтернативные функции выводов микроконтроллера представлены на рис. 1 (вскобках).

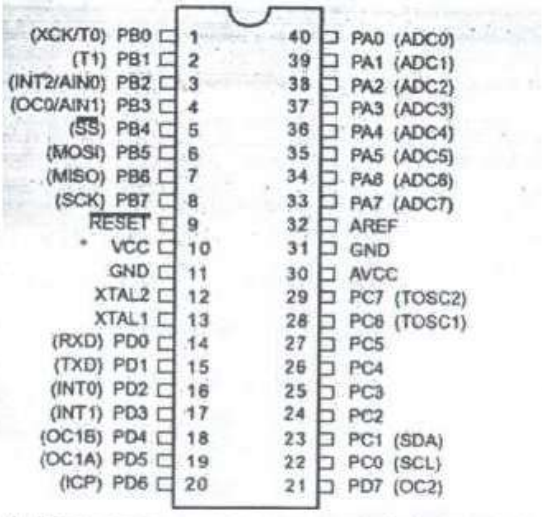


Рис. 1. Расположение выводов микросхемы контроллера ATmega8535 (DIP40)

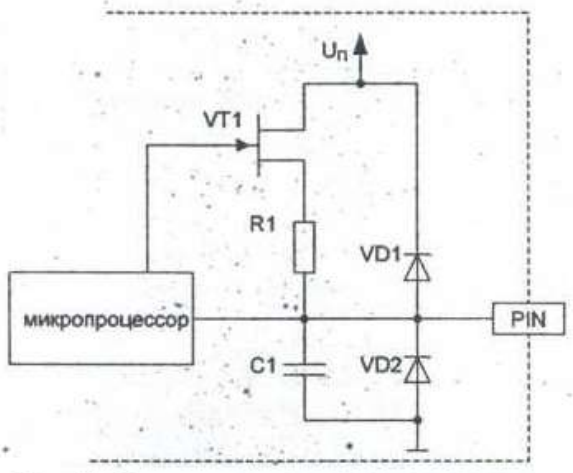


Рис. 2. Структура реализации вывода портов микроконтроллера

Каждый порт состоит из трех регистров, с помощью которых осуществляется установка направления работы порта и выдача/сбор информации (табл. 1).

Таблица 1. Регистры портов ввода/вывода

Наименование порта	Регистры		
	Регистр направления	Регистр данных	Регистр состояния
Порт A	DDRA	PORTA	PINA
Порт B	DDRB	PORTB	PINB
Порт C	DDRC	PORTC	PINC
Порт D	DDRC	PORTD	PIND

Регистры направления определяют режим работы портов ввода/вывода. Если в каком-либо разряде регистра установлена логическая «1», и соответствующий вывод микросхемы контроллера (рис. 1) работает на вывод информации из микроконтроллера. В противном случае соответствующий вывод микросхемы (рис. 1) работает на ввод информации в микроконтроллер.

Регистры данных предназначены для передачи данных навывода микросхемы контроллера. Если в каком-либо разряде регистра установлена логическая «1», а соответствующий вывод микросхемы сконфигурирован на выход с помощью регистра направления, то на вывод микросхемы контроллера подается сигнал, соответствующий логической «1». В противном случае на вывод микросхемы контроллера подается сигнал логического «0». Регистрыиспользуются для вывода информации из микроконтроллера.

Регистры состояния предназначены для отображения текущего состояния сигналов на выводах микросхемы контроллера. Так, если на выводе микросхемы находится сигнал логической «1», то соответствующий разряд регистрнаправления находится в состоянии логической «1». Регистры используются для ввода информации в микроконтроллер.

Инициализация порта на ввод и на вывод информации

Для того, чтобы освоить принципы установки порта на ввод или вывод информации, необходимо иметь представление о реализации его структуры.Каждый вывод порта выполнен по схеме, представленной на рис. 2.

При инициализации порта на ввод информации микроконтроллер принимает сигналы, поступающие от внешнего объекта. Эти сигналы подаются на выводы микросхемы (рис. 2, вывод PIN).

Диоды VD1 и VD2 выполняют функцию защиты микропроцессора от сигналов, величина которых находится за пределами диапазона 0...5В. Так, если напряжение на входе микроконтроллера превышает +5В, то открывается диодVD1, а если напряжение оказывается меньше 0В, то открывается диод VD2 (рис. 2). Конденсатор С1 выполняет защиту микроконтроллера от импульсных помех

Поскольку микропроцессор имеет высокое входное сопротивление, его вход является восприимчивым к воздействию помех. По этой причине важно, чтобы сигнал, подаваемый на микропроцессор, имел однозначное значение логического «0» или логической «1». Для этого в микроконтроллере присутствуют так называемые подтягивающие резисторы (PullUp).

Подтягивающий резистор имеет высокое сопротивление, измеряемое десятками кОм, и не оказываетвлияния на подключаемые к контроллеру сигналы. Через резистор R1 к порту ввода/вывода подключается напряжение питания с

помощью транзистора VT1. Если при этом к выводу микросхемы контроллера не подключена внешняя цепь и вывод «висит в воздухе», то на микропроцессор через подтягивающий резистор R1 подается сигнал логической «1». Это надежно защищает вывод микроконтроллера от воздействия внешних помех.

Для **инициализации порта на ввод информации** и подключения подтягивающих резисторов необходимо:

- задать в регистре направления DDRработу порта на ввод информации установкой нулевых значений в его разрядах;

- включить подтягивающие резисторы порта ввода/вывода установкой сигналов логической «1» в разрядах регистра PORTX.

Считывание данных с порта в данном режиме осуществляется путем опроса регистра состояния PIN.

Пример инициализации порта А на «ввод данных»:

```
ldi r16,0x00  ;в РОН r16 записывается число 0000 0000
out DDRA,r16  ;командой out значение r16 посылается в DDRA
ldi r16,0xFF  ;в РОН записывается число 1111 1111
out PORTA,r16 ;командой out значение r16 посылается в PORTA
in r16, PINA  ;командой in вводится значение PINA в РОН r16
```

При инициализации порта на вывод информации код, выдаваемый микропроцессором, поступает непосредственно на вывод микросхемы контроллера. Подтягивающие резисторы при этом должны быть отключены.

Для **инициализации порта на вывод информации** необходимо:

- задать в регистре направления DDRработу порта на вывод информации установкой его разрядов влогическую «1»;

- вывести необходимую информацию на порт записью в регистр данныхPORTнеобходимых значений.

Пример инициализации порта А на «вывод данных»:

```
ldi r16,0xFF   ;в РОН r16 записывается число 1111 1111
out DDRA,r16   ;командой out значение r16 посылается в DDRA
ldi r16,0x35   ;в r16 записывается любое число, например, 0x35
out PORTA,r16  ;командой out значение r16 посылается в PORTA
```

Задание

Составьте программы:

1. Инициализация порта B (C, D) на ввод данных
2. Инициализация порта B (C, D) на вывод данных

Контрольные вопросы

1. Сколько и каких портов имеет МК Atmega8535?
2. Каково назначение регистра направления?
3. Каково назначение регистра данных?
4. Каково назначение регистра состояния?
5. Каково назначение подтягивающих резисторов?

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Выполненное задание
3. Ответы на контрольные вопросы
4. Вывод по работе

Лабораторная работа № 8

Организация ввода-вывода информации через параллельные порты МК Atmega 8535.

Формируемые компетенции:

ПК 2.1. Создавать программы на языке ассемблера для микропроцессорных систем.

Цель работы: изучить процесс обмена информацией через порты ввода/вывода МК.

Выполнив работу, Вы будете:

уметь:

- составлять программы на языке ассемблера для микропроцессорных систем

Материальное обеспечение:

Стенд «Микроконтроллер», ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

Пример 1 (текст программы см. ПРИЛОЖЕНИЕ).

Написать программу, осуществляющую сложение двух младших и двух старших бит порта Сс выводом результата на порт D.

```
;---------------------------------------------------------------
;Программа для сложения двух двухбитных чисел A (биты PC7:PC6) и B
(биты PC1:PC0) с последующим выводом результата на PORTD
;Входы: PINC7:PINC6 и PINC1:PINC0
;Выходы: PORTD
.include "m8535def.inc"  ;Подключение библиотеки ATmega8535
.cseg                    ;Начало сегмента кода
.org 0
reset:                   ;Инициализация портов ввода/вывода
     ldi r16,0xFF        ;Установка PORTD на вывод информации: r16←0xFF
     out DDRD,r16        ;PORTD←r16
     clr r16             ;Установка PORTC на ввод информации:  r16←0
     out DDRC,r16        ; PORTD←r16
main:
     in r16,PINC         ;ввод данных из порта C в регистр: r16←PINC
     mov r17,r16         ;Копирование результата r17←r16
     andi r16,0x03       ;наложение маски: обнуление всех бит,кроме 0 и 1
     andi r17,0xC0       ;наложение маски: Обнуление всех бит,кроме 6 и 7
     lsr r17             ;Логический сдвиг r17 вправо на 6 бит
     lsr r17
     lsr r17
     lsr r17
     lsr r17
     lsr r17
     add r16,r17         ;Сложение r16 и r17: r16←r16+r17
     out PORTD,r16       ;Вывод результата в порт D: PORTD←r16
     rjmp main           ;возврат на main
;---------------------------------------------------------------
```

Рассмотрим программу более подробно.

1. Подключение стандартной библиотеки контроллера.

Строка .include "m8535def.inc"подключает библиотеку контроллера Atmega8535, которая содержитвсю необходимую информацию о контроллере, а именно список регистров с их адресами, объемы памяти, список периферийных устройств и другие особенности.

2. Выбор сегмента кода и адреса начала написания программы.

Строки. cseg и .org 0устанавливается начало сегмента кода на нулевой адрес.

3. Инициализация портов ввода/вывода.

Рассмотрим следующие строки программы:

ldirl6,0xFF
out DDRD,rl6
clr rl6
out DDRC,rl6

Эти команды инициализируют порт C на ввод данных, а порт D- на вывод. Поскольку порт Dнеобходимо инициализировать на вывод, в регистр DDRDтребуется записать 0xFF(1111 1111 в двоичном коде). Для этого сначала в регистр общего назначения r16 записывается число 0xFF(инструкция «ldi»), а затем содержимое этого регистра переписывается в регистр DDRD(инструкция «out»). Аналогичная операция производится с портом C.

4. Ввод данных и запись в регистры общего назначения. В главной программе согласно заданию необходимо, опросить состояние регистра PINCи выделить два числа: в первом хранятся два младших бита, во втором - два старших бита. Для этого вначале выполняется ввод, всего регистра PINCрегистры r16 и r17,(инструкции «in»и «mov»). Далее для корректного считывания чиселнеобходимо наложить, так называемые, маски на оба числа: в первом числе обнулить все биты кроме двух младших, во втором - все биты кроме двух старших. Это осуществляется с помощью команды «побитовое умножение» (инструкция «andi»): содержимое регистра r16 перемножается с константой 0x03 (0000 0011), а регистра r17 — с константой 0xC0 (1100 0000). Эти операции выполняются строками:

in rl6,PINC
movr17, rl6
andi rl6, 0x03
andirl7,0xC0

5. Приведение переменных к одному весовому коэффициенту. Полученные значения переменных в регистрах r16 и r17 имеют разные весовые коэффициенты. Необходимо преобразовать число XX00 0000, содержащееся в r17, в число вида 0000 00XX. Для этого в программе используется 6 раз одна и та же инструкция логического сдвига вправо «lsr»содержимого регистра r17 на 1 бит.

6. После выполнения операции сдвига r17 можно произвести сложение r16 и r17 и вывод результата сложения на PORTD:

add rl6,rl7
out PORTD,rl6

7. В конце программы ее необходимо «зациклить», т.е. вернуться на метку «main» для ее повторного выполнения:

rjmpmain

Пример 2 (текст программы см. ПРИЛОЖЕНИЕ).

Реализовать на микроконтроллере расчет логического уравнения:

$$D = A \bullet (\overline{A} \oplus B + C)$$

,

гдеA, B, C логические сигналы, поступающие на входынапример, PA0 (0 бит порта A), PA1 (1 бит порта A), PA2 (2 бит порта A), a Dрезультат решения логического уравнения, который выводится на 0 бит порта D.

```
;-----------------------------------------------------------------
;Программа для решения логического уравнения D=A*(NA⊕B+C)
;Входы:   A=PINA0
;         B=PINA1
;         C=PINA2
;Выход:   D=PORTD0

.include "m8535def.inc"  ;Подключение библиотеки Atmega8535
.def A=r20               ;Объявление переменных и присвоение их имен
.def B=r21               ;регистрам общего назначения r20…r23
.def C=r22
.def D=r23
.cseg                    ;начало сегмента кода
.org 0

reset:                   ;инициализация портов ввода/вывода
     ldi r16,0x01
     out DDRD,r16        ;PORTD0 - на вывод информации
     clr r16
     out DDRA,r16        ;PINA0 …PINA2 - на ввод информации
main:
     in r16,PINA         ;Ввод значения PINA в РОН r16
     mov A,r16           ;Копирование r16 в регистры A, B, C
     mov B,r16
     mov C,r16
     andi A,0x01         ;Выделение 0 бита числа A
     andi B,0x02         ;Выделение 1 бита числа B
     andi C,0x04         ;Выделение 2 бита числа C
     lsr B               ;Сдвиг B вправо на 1 бит
     lsr C
     lsr C               ;Сдвиг C вправо на 2 бита
     mov r16,A           ;Копирование A в РОН r16
     com r16             ;Инверсия содержимого r16
     andi r16,0x01       ;Формирование числа NA
     eor r16,B           ;Расчет r16=NA⊕B
     or r16,C            ;Расчет r16=r16+C
     and r16,A           ;Расчет r16=r16*A
     out PORTD,r16       ;Вывод результата на PORTD0
     rjmp main           ;Возврат на main
;-----------------------------------------------------------------
```

Рассмотрим программу более подробно.

1. Инициализация контроллера и переменных. Вначале выполняют инициализация микроконтроллера и присвоение имен A, B, C, Dрегистрам общего назначения r20 ... r23. Для этого используется директива«defA=r20»

.include "m8535def.inc"

.def A=r20

.def B=r21

.def C=r22

.defD=r23

. cseg . org 0

2. Инициализация портов ввода/вывода. В данной программе производится аналогично предыдущему примеру с следующими отличиями: порт Dинициализирован на вывод только 1 младшего бита, порт A - полностью на ввод данных.

3. Ввод данных. В основной программе сначала в регистр r16 вводится значение порта A, затем это значение переписывается в регистры A, B и C, и далее «наложение» маски на эти регистры: в этих регистрах выполняется выделение только отдельных битов: в A - 0 бит, B - 1 бит, C - 2 бит:

in rl6,PINA

mov A,rl6

movB,rl6

movC,rl6

andi A,0x01

andi B,0x02

andi C, 0x04

4.Выравнивание весовых коэффициентов переменных. Биты в регистрах A, B, C имеют разные весовые коэффициенты, для их выравнивания выполняется смещение значений регистров B и C вправо на 1 и 2 бита соответсвенно.

5. Формирование инверсного значения переменной. Для формирования инверсного значения регистра A это значение копируется в регистр r16, далее оно инвертируется (инструкция «com») и опять выделяется только младший бит:

movr16,A

comr16

andir16,0x01

6. Вычисление логического выражения и вывод данных. Логические операции выполняются согласно уравнению D=A•(A•B + C), далее результат выводится на индикацию в порт Di программа зацикливается:

andr16,B
orr16,C
andr16,A
outPORTD,r16
rjmpmain

Задание на выполнение

1. На базе примера №1 составить программу для вычисления:

а) суммы двух 3-разрядных чисел;
б) суммы двух 8-разрядных чисел;
в) разности двух 2-разрядных чисел;
г) разности двух 4-разрядных чисел.

2. Разработать логическую систему автоматизации - составить программу для своего варианта по реализации заданного логического уравнения, ввести программу в МК и проверить ее работоспособность на контроллере.

1. $PC3=\overline{(PA0+PA1)}\cdot PA3$	11. $PD5=\overline{PA1+PA2}\oplus PD0$
2. $PD4=PB1\oplus PB2+PB3\cdot PB4$	12. $PD3=PB7+\overline{PB6\cdot PB5}$
3. $PD0=PA1+\overline{(PA2\oplus PA3\cdot PA4)}$	13. $PC3=\overline{(PA0+PB0)}+PD7$
4. $PD7=\overline{PA7\oplus PA6\cdot PA5}$	14. $PC0=PD7\oplus \overline{PD1+PD2}$
5. $PC0=\overline{(PA3\cdot PA4)}+PA5$	15. $PA0=PC0\oplus\overline{PC1}\cdot PA7$
6. $PC7=PD2\oplus\overline{PD3+PD4}$	16. $PC1=PC2+PC3\cdot PC4$
7. $PD1=(PC0\cdot PC1)\oplus(PB0\cdot PB1)$	17. $PA0=PC0\cdot PC1+PD0\oplus PD1$
8. $PC1=PB2+\overline{PB3\cdot PB4}$	18. $PD7=\overline{PA0\cdot PB1}\oplus PC2$
9. $PA1=\overline{PC0}\oplus\overline{PC1}+\overline{PC6}\oplus PC7$	19. $PC1=\overline{PA0\cdot PA1+PA3}$
10. $PD7=\overline{PD0\cdot PD1}+PD2$	20. $PA7=PB0\oplus PB2\cdot\overline{PB1}+PB3$

3. Составить программу и проверить ее работоспособность в микроконтроллере, в которой число набранное в порт А изменяется (используя битовые операции) и выводится в порт С:

а) увеличивается в 2 раза;
б) уменьшается в 4 раза;
в) выводится в обратном коде;
г) выводится в дополнительном коде.

4. Составить программу и проверить ее работоспособность в микроконтроллере, в которой вводится два четырехразядных числаА(бит PA0...PA3 и B (биты PB0...PB3), результат выводится в порт С и выполняется следующая операция;

а) арифметическая сумма чисел;
б) поразрядная логическая сумма;
в) арифметическое произведение чисел;
г) поразрядное логическое произведение;
д) арифметическая разность

Контрольные вопросы

1. Сколько портов имеет микроконтроллер ATmega8535?
2. Какие регистры определяют режим работы порта? Поясните их назначение
3. Определите регистры работы порта С если известно, что 2 бита порта работают на ввод данных, остальные на вывод.
4. Для каких целей используется директивы ‘.def, ‘.cseg’, ‘.org’?
5. Какие инструкции по выполнению логических операций вы знаете?
6. Как наложить маску на считываемое значение регистра состояния?

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Выполненное задание
3. Ответы на контрольные вопросы
4. Вывод по работе

Практическая работа № 11
Изучение работы регистра состояний SREG МК Atmega8535.

Формируемые компетенции:

ПК 2.1. Создавать программы на языке ассемблера для микропроцессорных систем.

Цель работы: изучитьработу регистра состояний SREG МК.

Выполнив работу, Вы будете:

уметь:

- составлять программы на языке ассемблера для микропроцессорных систем

Материальное обеспечение:

ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

В микроконтроллерах фирмы Atmelприсутствует так называемый регистрсостояния SREG (StatusREGister).

Регистр состояния содержит информацию о результатах наиболее частоизвлекаемых арифметических операциях. Эта информация может быть использована при написании программы для создания переходов, циклов сравнения чисел и т.д.

Разряды регистра SREGназываются флагами. Всего этих флагов восемь:

№бита	7	6	5	4	3	2	1	0
Флаг	I	T	H	S	V	N	Z	C

Назначение разрядов регистра SREG:

Бит 0: C - флаг переноса.Флаг переноса индицирует появление переноса при выполнении арифметических или логических операций. Флаг переноса незаменим при совершении операций с n-байтными числами.

Пример 1. Прибавить к числу 0xFEпроизвольное 8-разряднос число. Результат - шестнадцатиразрядное число.

```
clr r18          ;очистка регистра r18
ldi r16,0xFE     ;запись в r16 значения 0xFE
ldi r17,0x05     ;запись любого числа в r17
mov r20,r16      ;копирование r16 в r20
add r20,r17      ;сложение r20 и r17
adc r21,r18      ;если при этом возникает перенос, то в r21
                 ;прибавляется 0x01. Таким образом, результат
                 ;формируется в r21:r20
```

Бит 1: Z- флаг нуля. Этот флаг индицирует нулевой результат привыполнении арифметических или логических операций. Этот флаг может быть полезен при выполнении такой операции, как сравнение.

Пример 2. Осуществить сравнение двух чисел, задаваемых на портах A и C. При их равенстве выдать сигнал логической «1» на PORTDO

```
main:
    in r16,PINA      ;ввод первого числа
    in r17,PINC      ;ввод второго числа
    cp r16,r17       ;сравниваются значения регистров r16 и r17
                     ;(проведением операции вычитания r16-r17). При их
                     ;равенстве устанавливается в '1' флаг Z
    breq m1          ;при Z=1 осуществляется переход на метку m1
    cbi PORTD,0      ;иначе бит PORTD0 очищается
    rjmp main        ;и идет возврат на main
m1:
    sbi PORTD,0      ;при переходе на m1 устанавливается бит PORTD0
    rjmp main        ;и идет возврат на main
```

Бит 2: N- флаг отрицательного значения. Этот флаг индицирует наличие отрицательного числа как результата выполнения арифметических или логических операций. В микроконтроллерах отрицательное число получается из положительного путем перевода в дополнительный код. Для этого все разряды числа инвертируются, а потом к числу прибавляется единица младшего разряда. Так, если 8-разрядное число +1 записывается в двоичном коде как 0000 0001, то отрицательное число -1 записывается как 1111 1111. При этом старший разряд (бит 7) числа определяет его знак.

Флаг отрицательного значения может быть полезен при совершении операции сравнения двух чисел.

Пример 3. Осуществить сравнение двух чисел A и B, задаваемых на портахAи C. Если число A≥B, то на PORTD0подается логическая «1». Иначе на PORTD0подается логический «0».

```
main:
    in r16,PINA      ;ввод числа A
    in r17,PINC      ;ввод числа B
    cp r16,r17       ;сравнение чисел A и B вычитанием A-B
    brmi m1          ;если при этом установлен флаг N, то
                     ;осуществляется переход на m1
    sbi PORTD,0      ;иначе устанавливается разряд PORTD0
    rjmp main        ;и идет возврат на main
m1:
    cbi PORTD,0      ;по метке m1 очищается разряд PORTD0
    rjmp main        ;и идет возврат на main
```

Бит 3: V - флаг переполнения. Этот бит устанавливается при переполнении регистра во время совершения операций над числами. Так, если в регистре было записано число 1111 1111 и к нему прибавили +2, то происходит переполнение. Результатом такой операции будет число 0000 0001 и установленный флаг V.

Необходимо различать флаг переполнения V и флаг переноса C. Флаг переполнения предназначен, в первую

очередь, для работы с дополнительным кодом. Как было сказано выше, в дополнительном коде старший разряд определяет знак числа, а значение числа ограничивается 7 разрядами.

Если при сложении двух положительных чисел в дополнительном коде происходит изменение старшего бита, то это означает, что произошло переполнение числа, хотя переноса не происходит (флаг С не меняется) Например, при сложении чисел:

```
      0110 0010
+     0110 1111
=     1101 0001
```

флаги устанавливаются следующим образом:

-поскольку произошло изменение старшего разряда, то флаг переполнения V устанавливается: V=l;

- так как не произошло переполнение разрядов регистра, то флаг переноса не возникает: С=0.

Если при сложении двух отрицательных чисел в дополнительном коде происходит изменение старшего бита, то это означает, что также произошло переполнение числа, при этом может возникнуть и флаг переноса. Например, присложении чисел:

```
      1001 1110
+     1001 0001
=   1 0010 1111
```

меняются как старший разряд, так и появляется бит переполнения:

-поскольку произошло изменение старшего разряда, устанавливается флаг переполнения V;

-поскольку произошло переполнение разрядов регистра, устанавливаем и флаг переноса С.

*Пример 4.*Используя инструкцию «add», сложить два положительныхчисла в дополнительном коде. Если происходит переполнение результата, то результат необходимо ограничить максимальным или минимальным числом. Результат выводится на PORTD.

```
main:
    in r16,PINA      ;ввод первого числа A
    in r17,PINC      ;ввод второго числа B
    add r16,r17      ;сложение A и B
    brvc m1          ;если нет переполнения (флаг V=0), то
                     ;осуществляется переход на m1
    brcs m2          ;иначе проверка флага переноса. Если флаг C=1,
                     ;то осуществляется переход на m2
    ldi r16,0x7F     ;если переноса нет, то результат -
                     ;максимальное число +127 (0111 1111)
    rjmp m1          ;далее - переход на m1
m2:
    ldi r16,0x80     ;если флаг переноса C=1, то результат -
                     ;минимальное число -128 (1000 0000).
m1:
    out PORTD,r16    ;по метке m1 - вывод результата на PORTD
    rjmp main        ; и возврат на main
```

Бит 4:S- флаг знака. Этот бит всегда является результатом совершения операции исключающего ИЛИ между флагом отрицательного числа Nи флагом переполнения V. Так, если при совершении операции не происходит переполнения (флаг V=0), то знак определяется флагом N. Если же происходи переполнение (V=l), то флаг знака принимает инвертированное значение флага N.

Бит 5:H - флаг половинного переноса. Половинным переносом называется процесс переноса из первой половины байта во вторую. Так, если в байте была комбинация 0000 1111 и к нему прибавили +1, то происходит половинный перенос, а именно 0001 0000, при этом формируется флаг H.

*Пример 5.*Организовать бегущий огонь в младшем полубайте PORTC.

```
      ldi r16,0xFF   ;в r16 записывается значение 0xFF
      out DDRC,r16   ;порт C инициализируется на вывод информации
      ldi r16,0x01   ;в r16 записывается единица младшего разряда
main:
      out PORTC,r16  ;значение r16 выводится на PORTC
      lsl r16        ;r16 сдвигается влево на один разряд
      brhc main      ;если флаг H снят, то переход на main
      ldi r16,0x01   ;иначе в r16 записывается 0x01
      rjmp main      ;и осуществляется переход на main
```

Бит 6: T - хранение бита информации. Инструкции копирования бит используют бит T регистра SREG. При копировании бита из какого-либо регистра он сохраняется в бите T регистра SREG, а затем извлекается из него при копировании в какой-либо другой регистр.

*Пример 6:*Скопировать из регистра r16в регистр r19пятый бит, не изменяя содержимое регистра r16**.**

```
Без использования флага T:
in r16,PINA    ;запись в r16 любого числа
mov r17,r16    ;копирование r16 в любой свободный РОН (r17)
andi r17,0x20  ;выделение 5-го бита в r17
or r19,r17     ;логическое ИЛИ r19 и r17
С использованием флага T:
in r16,PINA    ;запись в r16 любого числа
bst r16,5      ;копирование 5-го бита в SREG
bld r19,5      ;копирование содержимого флага T в 5-й бит r19
```

Бит 7: I - общее разрешение прерываний. Назначение этого флага будет пояснено в работах сиспользованием прерываний таймеров.

Контрольные вопросы

1. Для чего предназначен регистр состояния?
2. Перечислите биты регистра состояния и их назначение.
3. При выполнении каких операций изменяется для флага нуля? Флаг отрицательного значения? Флаг знака?
4. При суммировании двух 4-х разрядных чисел какие биты регистра состояния могут изменить свое значение? Двух восьмиразрядных?
5. Какой бит регистра состояния отвечает за разрешение работы всех прерываний?
6. Как связаны друг с другом флаги: знак, отрицательное значение переполнение?
7. Для каких целей используется бит T?

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Программные примеры работы с флагами
3. Ответы на контрольные вопросы
4. Вывод по работе

Лабораторная работа № 9
Исследование работы регистра состояний SREG МК Atmega8535.

Формируемые компетенции:

ПК 2.1. Создавать программы на языке ассемблера для микропроцессорных систем.

Цель работы: научиться применять биты регистра состояния при написании программ.

Выполнив работу, Вы будете:

уметь:

- составлять программы на языке ассемблера для микропроцессорных систем

Материальное обеспечение:

Стенд «Микроконтроллер», ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

Рассмотрим практическое использование флагов регистра состояния на примере программы вычитания двух чисел:первое число задается младшим полубайтом порта А и имеет формат:

Бит	7	6	5	4	3	2	1	0
A=	0	0	0	0	PA3	PA2	PA1	PA0

второе число задается старшим полубайтом порта А и имеет формат:

Бит	7	6	5	4	3	2	1	0
B=	PA7	PA6	PA5	PA4	0	0	0	0

Результат должен выводиться в следующем виде:

- младшие 4 разряда порта С - модуль разности (A-B);
- старший бит порта С - знак результата.

```
//-------------------------------------------------------------------
.include "m8535def.inc"  ;подключение библиотеки ATmega8535
.cseg                    ;начало сегмента кода
.org 0
    ldi r16,0xFF         ;инициализация портов ввода/вывода
    out PORTA,r16        ;порт A - на ввод информации
    out DDRC,r16         ;порт C - на вывод информации
main:
    in r16,PINA          ;ввод данных PINA в РОН r16
    mov r17,r16          ;копирование r16 в r17
    andi r16,0x0F        ;выделение числа A
    andi r17,0xF0        ;выделение числа B
    clr r18              ;очистка РОН r18
m1:
    lsr r17              ;сдвиг r17 на 4 разряда вправо
    inc r18
    cpi r18,0x04
    brne m1
    sub r16,r17          ;выполнение вычитания r16-r17 (A-B)
    brmi m2              ;если в результате установлен флаг N, то
                         ;осуществляется переход на метку m2
    rjmp m3              ;иначе - переход на метку m3
m2:                      ;выделение модуля результата
    com r16              ;инверсия результата
    inc r16              ;увеличение результата на +1
    ori r16,0x80         ;прибавление к результату сигнала «знак»
m3:
    out PORTC,r16        ;вывод результата на PORTC
    rjmp main            ;и возврат на main
//-------------------------------------------------------------------
```

Рассмотрим программу более подробно.

1. Инициализация портов ввода/вывода: порт A - на ввод, порт C - на выводинформации:

```
ldi rl6,0xFF
out PORTA,rl6
out DDRC,rl6
```

2. В начале основного цикла производится опрос регистра PINAприведение чисел к одному виду:

- результат заносится в регистр r16 (inr16, PINA)и r17 (movr17, rl6);
- затем в регистре r17 осуществляется сдвиг числа на 4 разряда вправо для того, чтобы преобразовать его из числа вида «XXXX0000» в число «0000 XXXX»:

```
main:
    in rl6,PINA
    mov rl7,rl6
    andi rl6,0x0F
    andi rl7,0xF0
    clr rl8
m1:
    lsr rl7
    inc rl8
    cpi rl8,0x04
    brne ml
```

3. Операция вычитания, после выполнения которой флаги в регистре SREGвыставятся определенным образом:

```
subrl6, rl7
```

4. Для обработки результатов вычитания необходимо проверить флаг отрицательного значения N:

- если он отсутствует, то результат - положительное число и его сразу можно выводить на индикацию;
- если же флаг Nустановлен, то результат - отрицательное число, и перед выводом на индикацию-необходимо выделить его модуль:

```
brmim2
rjmpm3
```

Команда brmim2 осуществляет проверку флага N. Если он установлен, осуществляется переход на метку m2, при переходе на которую осуществляется выделение модуля числа. Если флаг Nне установлен, выполняется следующая за командой директива rjmpm3. По этой команде выполняется переход на метку m3,по которой осуществляется вывод результата на индикацию.

5. При переходе на метку m2 результат сначала инвертируется (comr16), а затем к нему прибавляется +1 (incr 16). Таким образом, осуществляется выделение из дополнительного кода модуля результата вычитания:

```
m2:
    comr16
    incr16
    orir16,0x80
m3:
```

out PORTC, rl6

rjmpmain

Командой orirl6, 0x80 осуществляется установка сигнала «Знак» в старшем разряде регистра r16, который затем выдается на индикацию на PORTCoutPORTC,r16. После этого основной цикл программы замыкается.

Задание (одно на выбор)

1. Используя логические операции и биты регистра состояния реализовать на микроконтроллере схему:
- полусумматора;
- полного одноразрядного сумматора.
2. Используя биты регистра состояния вычислить модуль разности двух 5-ти разрядных чисел.
3. Используя биты регистра состояния реализовать компаратор: сравниваются два 4-х разрядных числа (порты A и B): если равны – включается 0 бит порта D, если первое число больше – 1 бит, если второе – 2 бит.
4. Используя бит хранения информации регистра состояния реализовать
   - RS-триггер;
   - D-триггер;
   - T-триггер

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Выполненное задание с комментариями (работоспособность программы демонстрируется на стенде)
3. Вывод по работе (указать какие флаги были использованы и почему)

Практическая работа № 12
Изучение работы стека МК Atmega8535.

Формируемые компетенции:

ПК 2.1. Создавать программы на языке ассемблера для микропроцессорных систем.

Цель работы: изучитьорганизацию и принцип работыстека МК.

Выполнив работу, Вы будете:

уметь:

- составлять программы на языке ассемблера для микропроцессорных систем

Материальное обеспечение:

ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

При выполнении программы извлечение и выполнение директив (команд) ведется последовательно. То есть сначала исполняется первая в списке команда, потом - вторая, третья и т.д.

Все микроконтроллеры и микропроцессоры имеют так называемый счетчик команд (ProgramCounter). В нем хранится адрес текущей выполняемой команды. При запуске программы счетчик команд равен 0, затем он инкрементируется по мере выполнения команд на 1 или на 2, в зависимости от длины команды (Приложение А). Если по какой-либо причине в программе осуществляется переход в другое место программы (безусловный или условный переход), то содержимое счетчика команд скачком изменяется, указывая на новый адрес вызванной команды.

Если при выполнении программы возникает необходимость осуществить переход на какую-либо подпрограмму, выполнить ее, а затем вернуться на старое место и продолжить выполнение программы, становится необходимым запоминать адрес возврата. Для этого и предназначен указатель стека.

Указатель стека - это специальный регистр, представляющий собой буфер, в котором реализован принцип «LastIn- FirstOut» LIFO(последним пришел - первым уйдешь). Этот принцип означает, что адрес перехода, который был записан в стек самым последним, будет считан самым первым. Таким образом, если в программе последовательно выполняются несколько переходов вподпрограммы, никогда не возникнет путаницы с порядком возврата и подпрограмм обратно.

В микроконтроллере Atmega8535 стек реализован в двух 8-разрядны регистрах SPH, SPL, которые вместе образуют 16-разрядный регистр SPH, SPL. Принцип работы стека поясняется на рис. 1.

При работе программы каждая ее команда имеет адрес, присваиваемый счетчиком команд (рис. 1). Если при выполнении программы возникла необходимость сделать переход на какую-либо подпрограмму, расположенную и какому-либо адресу, например, как показано на рис. 1, по адресу 243, необходимо иметь информацию, на какой адрес необходимо вернуться после выполнения подпрограммы. Поэтому при переходе, например, с адреса 006 на адрес 243 указатель стека записывается адрес возврата, то есть 007.

При вызове следующей подпрограммы, расположенной по адресу 120, указатель стека записывается также адрес возврата. В данном случае - 052.

Адрес возврата записывается в вершину стека.

Вершиной стека называется адрес ОЗУ процессора, в которую ведется запись адреса возврата.

Адрес вершины стека программист обязан указать самостоятельно, при этом общепринято, что вершина стека находится в конце ОЗУ процессора. Этоделается для того, чтобы область программы случайно не перекрылась с областью стека, так как если это произойдет, то адрес возврата, записанный в область стека, будет потерян.

Возможен случай, когда при выполнении подпрограммы происходит запрос на следующий переход на другую подпрограмму (рис. 2).

Когда происходит переход из основной программы в подпрограмму 1, в вершину стека записывается адрес

возврата 007. Если далее из подпрограммы 1 происходит переход в подпрограмму 2, то в стек записывается адрес 245 возврата в подпрограмму 1.

Рис. 1. Принцип работы указателя стека

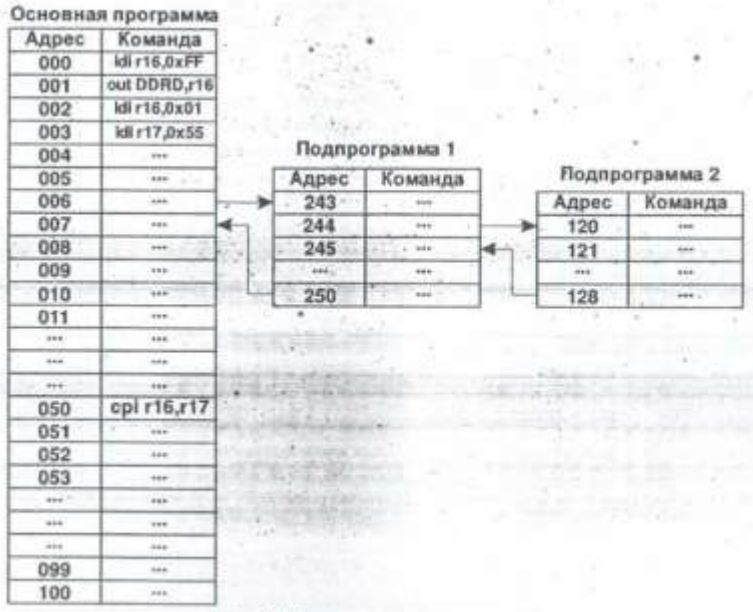


Рис. 2. Принцип заполнения стека

При окончании подпрограммы 2 первым будет прочитан адрес 245, а затем при окончании подпрограммы 1 - адрес 007. Таким образом, благодаря принципу LIFOполностью исключается неправильный переход в неправильное место программы.

В микроконтроллере Atmega8535 указатель стека состоит из двух 8- разрядных регистров SPHи SPL, которые вместе составляют 16-разрядный регистр SPH:SPL.Применение 16-разрядного регистра объясняется объемом памяти программ микроконтроллера - он составляет 8кБ (4096 слов памяти программ) и для обращения к конкретной ячейке памяти необходимо указания 2 байт адреса.

При написании программ вызов подпрограмм обычно осуществляс командой rcall, а возврат из подпрограммы осуществляется по директиве ret.

При вызове директивы rcallв регистр SPH:SPLзаписывается адрес ячейки памяти, в которую будет сохранен адрес возврата из подпрограммы, и адрес возврата автоматически записывается в стек, а при вызове директивыretпроисходит чтение из стека по указанному в регистре SPH:SPLадресу.

Контрольные вопросы

1. Что такое стек?
2. Что такое вершина стека?
3. Что такое указатель стека?
4. Что означает термин LIFO?
5. Что представляет собой стек в МК Atmega8535?

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Ответы на контрольные вопросы
3. Вывод по работе

Лабораторная работа № 10

Разработка программы для организации программной задержки (с использованием стека)

Формируемые компетенции:

ПК 2.1. Создавать программы на языке ассемблера для микропроцессорных систем.

Цель работы: научиться применять стек при организации программной задержки.

Выполнив работу, Вы будете:

уметь:

- составлять программы на языке ассемблера для микропроцессорных систем

Материальное обеспечение:

Стенд «Микроконтроллер», ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

Программная задержка

Одним из наиболее наглядных применений указателя стека является реализация программной задержки времени.

В микроконтроллере ATmega8535 присутствуют три таймера, с помощью которых можно достаточно просто реализовать любую временную задержку. Однако простейшую программную задержку необходимо уметь реализовывать без использования дополнительных аппаратных средств микроконтроллера.

Суть программной задержки состоит в том, чтобы заставить процесс выполнять циклически одно и то же действие, например, инкремент какого-то числа от нуля до максимума, с дальнейшим переходом при окончании выполнения этого действия к дальнейшему выполнению программы.

Ввиду того, что процессор производит операции с большой скоростью,длясколько-нибудь ощутимой задержки приходится производить операции большими числами.

Пусть нам задан некоторый элемент программы. Рассчитаем примерноевремя исполнения этого элемента (в комментариях прописано количество тактов процессора для выполнения данной инструкции):

```
//-------------------------------------------------------------
    clr r16         ; 1 такт процессора
ml:                 ; не выполняется, нет задержки
    inc r16         ; 1 такт процессора
    cpi r16,0x20    ; 1 такт процессора
    brne met_1      ; 2 такта процессора, если условие выполняется
                    ; (переход к метке ml) и 1 такт процессора,
                    ; если условие не выполняется (выход из цикла)
//-------------------------------------------------------------
```

В программе значение регистра РОН r16 увеличивается на 1, пока значение в r16 не достигнет значения 0x20 (десятичное число 32). При этом постоянно идет возврат на метку ml, и только после того, как значение, записанное в r16достигает значения 0x20, разрешается выполнение последующих командпрограммы. Таким образом, в программе реализован цикл по переменной r16**,** время исполнения которого она «ничего не делает», т.е. как бы «зависает» некоторое время. Это процесс и называется *программной задержкой.*

Зная время выполнения команд (прил. 1) можно рассчитать времяисполнения любой части программы. Рассчитаем время исполнения рассмотренной части программы.

При попадании в цикл суммарное количество тактов процессора будет N,то есть при подсчете чисел от 1 до 32 суммарное количество тактов будет равняться N=4-32=128. После этого необходимо вычесть один цикл, который остался неучтенным при выполнении ложного условия brnemet_lв конце подсчета r16 и прибавить один такт на выполнение команды clrr16. Итого, суммарное количество тактов равно:N=4•32-1+1=128.

Если умножить полученное число на время выполнения одного такта процессора, которое обратно частоте колебаний кварцевого резонатора f_{OSC}(в нашем случае - 8 МГц), то можно найти время задержки $T_З$:

$$Тз = N * \frac{1}{fosc} = 128 * \frac{1}{8 * 10^6} = 16 \text{ мкс}$$

Дляреализации более существенных временных задержек приходится иметь дело с большими числами или вложенными циклами. Пример программной задержки с применением вложенных циклов приведен далее.

Пример. Написать программу бегущего огня на PORTC. При этом время задержки между переключениями разрядов порта задается с помощью программной задержки.

```
;-----------------------------------------------------------
.include"m8535def.inc"   ;стандартная библиотека Atmega 8535
.cseg                    ;начало сегмента кода
.org 0
   ldi r16,$5f           ;размещение вершины стека по адресу
   ldi r17,$02           ;старшей ячейки ОЗУ
   out spl,r16
   out sph,r17
   ldi r16,0xFF          ;инициализация портов ввода/вывода
   out DDRC,r16          ;PORTC - на вывод
   ldi r16,0x01          ;занесение в РОН r16 числа 1
main:                    ;начало главной программы
   out PORTC,r16         ;вывод на PORTC значения r16
   in r17,SREG           ;сохранение регистра SREG
   rcall delay1          ;вызов подпрограммы задержки
   out SREG,r17          ;восстановление SREG после возврата
   rol r16               ;циклический поворот r16 вправо
   rjmp main             ;возврат на main
delay1:        ;подпрограмма №1
   clr r18               ;очистка регистра r18
 met1:
   rcall delay2          ;переход на 2 подпрограмму - delay2
   inc r18               ;инкремент r18
   cpi r18,0x10          ;сравнение значения r18 с числом 16
   brne met1             ;если значение в r18≠15, то переход на met1
   ret                   ;иначе - возврат в основную программу
delay2:        ;подпрограмма №2
   clr r19               ;очистка регистра r19
 met2:
   rcall delay3          ;переход на 3 подпрограмму - delay3
   inc r19               ;инкремент r19
   cpi r19,0xFA          ;сравнение значения r19 с числом 250
   brne met2             ;если значение в r19≠255, то переход на met2
   ret                   ;иначе - возврат в подпрограмму №1
```

Рассмотрим программу более подробно.

1. В начале программы вершина стека помещается по адресу последней ячейки ОЗУ:

ldi rl6,$5F

```
ldi rl7,$02
out spl,rl6
out sph,rl7
```

2. Инициализируется порт ввода/вывода С, а регистру r16 присваивают значениеr16=0x01 (начальная позиция бегущего огня):

```
ldi rl6, 0xFF
out DDRC,rl6
ldi rl6,0x01
```

3. В главной программе значение регистра r16 выводится на индикацию PORTC, после чего вызывается подпрограмма задержки (rcalldelay),попрошествии которой значение r16 сдвигается на один разряд вправо (rolr16).Далее программа замыкается:

```
main:
    out PORTC,rl6
    in rl7,SREG
    rcall delay
    out SREG,rl7
    rol rl6
    rjmp main
```

Здесь необходимо отметить, что перед вызовом подпрограммы задержки значение регистра SREGнеобходимо сохранить, так как иначе в результате выполнения подпрограммы этот регистр может изменить свое значение. После возврата подпрограммы значение SREGвосстанавливается.

4. Подпрограмма задержки реализована по принципу, рассмотренному выше. В подпрограмме реализованы вложенные циклы:

```
delayl:
    clr rl8
    metl:
    rcall delay2
    inc rl8
    cpi rl8,0xl0
    brnemetl
    ret
delay2:
    clr rl9
    met2:
    rcall delay3
    inc rl9
    cpi rl9,0xFA
    brne met2
    ret
delay3:
    clr r20
    met3:
    inc r20
    cpi r20, 0xFA
    brne met3
    ret
```

Так, при входе в первый цикл, ограниченный меткой metli условием brnemetl, перед инкрементом регистра r18 происходит вызов подпрограммы delay2, в которой данная операция повторяется (выполняется второй цикл, вложенный в первый) и осуществляется переход на подпрограмму delay3, в которой выполняется третий цикл, вложенный во второй.Таким образом, сначала происходит выполнение цикла в подпрограмме delay3 (инкремент регистра r20 по метке met3), потом - цикла в подпрограмме delay2 (инкремент r19 по метке met2), после чегоснова осуществляется переход на delayl(инкремент r18 по метке met1)и т.д..

Дисассемблирование программы

При оформлении отчета по проделанной работе необходимо произвести дисассемблирование написанной программы. Дисассемблирование представляет собой инструмент AVRStudio, благодаря которому можно увидеть работу программы, включая указатель стека и счетчик команд.

Дисассемблирование программы выполняется выбором в меню *View* AVR Studioпункта*Disassembler*.В этом случае на экране появляется программа, в которой указывается адрес каждой команды и необходимая служебная информация.

Одновременно на экран необходимо вывести окно контроля содержимого памяти *Togglememorywindow*(для вызова набрать комбинацию Alt+0), в котором необходимо выбрать память данных *(Data).*В конце области этой памяти будет располагаться указатель стека.

В отчете по работе необходимо привести таблицу дисассемблера со значениями, записанными в стеке во время исполнения программы (ее первого прохода). В таблице необходимо указать:

- все инструкции и метки программы;
- адрес памяти всех инструкций;
- текущее значение указателя стека во время исполнения программы;
- значения всех ячеек ОЗУ, в которые записывается стек, во время исполнения программы (первого прохода).

Пример такой таблицы приведен в табл. 1.

Таблица 1: Дисассемблер программы со стекок

Адрес памяти программ	Команда / метка	Указатель стека SPHiSPL	Стек (ячейки ОЗУ)		
			025A:025B	025C:025D	025E:025F
+00000000	ldi rl6,$5F	00 00	FF: FF	FF: FF	FF: FF
+00000001	ldir17, $02	00 00	FF: FF	FF: FF	FF: FF
+00000002	out spl,rl6	00 5F	FF: FF	FF: FF	FF: FF
+00000003	out sph,rl7	02 5F	FF: FF	FF: FF	FF: FF
+00000004	ldi rl6,0xFF	02 5F	FF: FF	FF: FF	FF: FF
+00000005	out DDRC,rl6	02 5F	FF: FF	FF: FF	FF: FF
+00000006	ldi rl6,0x01	02 5F	FF: FF	FF: FF	FF: FF
	main:				
+00000007	out PORTC,rl6	02 5F	FF: FF	FF :FF	FF: FF
+00000008	in rl7,SREG	02 5F	FF: FF	FF: FF	FF: FF
+00000009	rcalldelayl	02 5D	FF: FF	FF: FF	00 : 0A
+0000000A	out SREG, r17	02 5F	00:15	00: 0F	00: 0A
+0000000B	rol rl6	02 5F	00:15	00: 0F	00: 0A
+0000000C	rjmp main	02 5F	00:15	00: 0F	00: 0A
	delayl:				
+0000000D	clr rl8	02 5D	FF :FF	FF: FF	00: 0A
	metl:				
+0000000E	rcall delay2	02 5B	FF:FF	00: 0F	00: 0A
+0000000F	inc rl8	02 5D	00:15	00: 0F	00: 0A
+00000010	cpi rl8,0x10	02 5D	00:15	00: 0F	00: 0A
+00000011	brnemetl	02 5D	00:15	00: 0F	00: 0A
+00000012	Ret	02 5F	00:15	00 : 0F	00 : 0A
	delay2:				
+00000013	clr rl9	02 5B	FF:FF	00 : 0F	00: 0A
	met2:				
+00000014	rcall delay3	02 59	00:15	00: 0F	00: 0A
+00000015	inc rl9	02 5B	00:15	00: 0F	00: 0A
+00000016	cpi rl9,0xFA	02 5B	00:15	00: 0F	00 : 0A
+00000017	brne met2	02 5B	00:15	00: 0F	00: 0A
+00000018	Ret	02 5D	00:15	00: 0F	00: 0A
	delay3:				
+00000019	clr r20	02 59	00:15	00: 0F	00: 0A
+0000001A	inc r20	02 59	00:15	00: 0F	00: 0A
+0000001B	cpi r20,0xFA	02 59	00:15	00: 0F	00: 0A
+0000001C	brne met3	02 59	00:15	00: 0F	00: 0A
+0000001D	ret	02 5B	00:15	00: 0F	00: 0A

Задание на выполнение

1. Разработать программу «бегущий огонь» с заданной по вариантам программной задержкой, которая изменяется в зависимости от состояния входов.

Варианты программы «бегущий огонь» представлены в таблице.

№ вар.	1 такт	2 такт	3 такт	4 такт
1	PD0, PD1 - 2 с	PD1, PD2 -1 с	PD2, PD3 -0,5 с	–
2	PA0…PA3 -0,1 с	PA4…PA7 – 0,2 с	PC0…PC3 -0,3 с	PC4…PC7 – 0,4 с
3	PC0 – 0,5 с	PC2 – 0,5 с	PC4 – 1,5 с	PC6 – 1,5 с
4	PB0 – 1 с	Нет – 0,5 с	PB1 – 1 с	Нет – 0,5 с
5	PC7 – 5 с	PC6, PC7 – 5 с	PC5…PC7 – 5 с	PC4…PC7 – 5 с
6	PA0 – 2 с	PA1 – 4 с	–	–
7	PC0…PC3 - 10 с	PC1…PC4 -10 с	PC2…PC5 - 10 с	–
8	PD7, PC7 -0,5 с	Нет – 2 с	PD0, PC0 - 0,5 с	Нет - 2 с
9	PC0 – 0,2 с	PC1 – 0,2 с	PC2 – 0,2 с	Нет – 1 с
10	PA0…PA3 – 0,1 с	Нет – 0,2 с	PA4…PA7 – 0,1 с	Нет – 0,2 с
11	Порт D – 5 с	Нет – 1 с	Порт C – 5 с	–
12	PA3 – 2 с	Нет – 4 с	–	–
13	PC7 – 1 с	Нет – 5 с	PC6 – 1 с	Нет – 5 с
14	PA0…PA3 – 2 с	Нет – 1 с	PD0…PD3 – 2 с	
15	PC0…PC3 – 0,1 с	PC1…PC3 -0,1 с	PC2…PC3 – 0,1 с	PC3 – 0,5 с
16	Порт D – 2 с	Порт A – 2 с	Нет – 4 с	
17	PC0… PC2 - 2 с	PC1…PC3 - 2 с	PC2…PC4 - 2 с	Нет – 5 с
18	Порт A -0,1 с	Нет – 1 с	Порт C -0,1 с	Нет – 1 с
19	PC0 – 0,1 с	PC1 – 0,1 с	PC2 – 0,1 с	Нет – 0,5 с
20	PB0 – 5 с	PB1 – 2,5 с	PB2 – 1,25 с	Нет – 5 с
21	PD7 – 1 с	PD6 – 1 с	PD0…5 – 5 с	–
22	PA6 – 1 с	PA7 – 4 с	–	–
23	PC0, PC3 – 2 с	PC1, PC4 – 2 с	Нет – 5 с	–
24	PD4…PD7 – 2,5 с	Нет – 5 с	PD0…PD3 – 2,5 с	Нет – 5 с
25	Нет – 1,5 с	Порт C – 0,5 с	Нет – 1,5 с	Порт A – 1,5 с
26	PB0 – 5 с	Нет – 10 с	–	–
27	PC6,7 – 2 с	PC4, PC5 – 2 с	PC2…PC3 – 2 с	PC0…PC1 – 4 с
28	PA0…PA3 – 2 с	PA4…PA7 – 2 с	PA0…PA7 – 2 с	Нет – 6 с
29	PD0,PD2,PD4 – 5 с	PD1, PD3,PD5 – 5 с	Нет – 10 с	–
30	PC0 – 1,5 с	PC1 – 3 с	PC2 – 4,5 с	–

Примечание: символ«-»обозначает что такт отсутствует, например, в 7вариантепрограмма выполняется только за 3 такта.

Контрольные вопросы

1. Поясните назначение стека.
2. В каких случаях в программе необходимо выполнять инициализацию стека?
3. Что произойдет с программой, если определить следующие значения указателя стека:
   а) SPH=0, SPL=0
   б) SPH=0, SPL=0x5F
   в) SPH=0x02, SPL=0

4. Каким образом посчитать время исполнения инструкции? Части программы?

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Выполненное задание с комментариями (работоспособность программы демонстрируется на стенде):
- листинг программы;
- дизассемблированную программу;
- алгоритм;
- представить расчет времени задержки по количеству команд;
- представить таблицу значений стека во время исполнения программы.

3. Вывод по работе

Практическая работа № 13
Изучение работы таймеров в различных режимах МК Atmega8535.

Формируемые компетенции:

ПК 2.1. Создавать программы на языке ассемблера для микропроцессорных систем.

Цель работы: изучитьработутаймеров МК в режиме ШИМ и в режиме создания временных интервалов.

Выполнив работу, Вы будете:

уметь:

- составлять программы на языке ассемблера для микропроцессорных систем

Материальное обеспечение:

ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

Широтно-импульсная модуляция (ШИМ)

При использовании микроконтроллеров очень часто требуется решать задачи создания точных временных задержек сигналов, оценивать длительность импульсов какого-либо внешнего сигнала, знать его частоту и скважность.

Все эти задачи решаются с помощью использования таймеров/счетчиков.

По сути, таймеры представляют собой счетчики, которые ведут подсчет импульсов либо от внешнего сигнала, либо от внутреннего генератора. Кроме этого таймеры/счетчики имеют возможность работать в режиме щиротно-

импудьсной модуляции (ШИМ).

Широтно-импульсной модуляцией называется формирование среднего значения сигнала с помощью сигналов логического «0» и логической «1» за период модуляции Т (рис. 1).

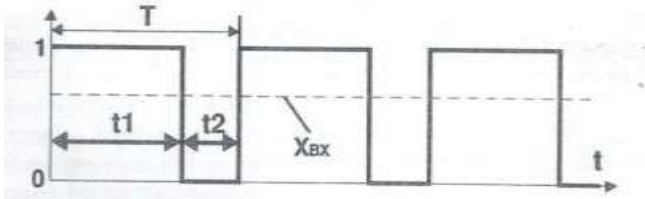


Рис. 1. Принцип широтно-импульсной модуляции сигнала

ШИМ дает возможность с помощью логических сигналов получать на выходе микроконтроллера аналоговый сигнал, среднее значение которого за период можно плавно изменять от 0 до 1, точнее от 0 до напряжения питания. Значение этого сигнала можно рассчитать по следующей формуле:

$$X_{BX}=1\cdot t_1/T,$$

где t_1- время включенного состояния (логической единицы);

t_2- время выключенного состояния (логического нуля);

T- период выходного сигнала.

Микроконтроллер ATmega8535 содержит три таймера общего назначения два восьмиразрядных таймера Т0 и Т2, один шестнадцатиразрядный таймер Т1. Восьмиразрядные таймеры Т0 и Т2 определяются и работают практическиидентично друг другу. Работа таймера Т1 имеет существенные отличия. В данной работе изучается работа только таймеров ТО и Т2.

Для управления работой таймера Т0 используются 3 регистра ввода/вывода:

- счетный регистр TCNT0;
- регистр сравнения OCR0;
- регистр управления TCCR0.

Соответствующие регистры таймера Т2 называются TCNT2, OCR2, TCCR2.

Для подключения входов/выходов таймеров к выводам микросхемы контроллера (т.е. соединения таймеров с внешним миром) используются альтернативные функции портов ввода/вывода. После включения альтернативной функции данный бит порта используется только для ввода информации в таймер (например, вход Т0/РВ0), или вывода информации с выхода таймера (например вывод ОС0/РВ3). Альтернативные функции портов для работы с таймерами приведены в табл. 1.

Таблица 1. Альтернативные функции портов, используемые таймерам)

Таймер	Обозначение	Описание	Вывод
Т0	Т0	Внешний вход таймера Т0	РВ0
	ОС0	Внешний выход таймера Т0	РВ3
Т1	Т1	Внешний вход таймера Т1	РВ1
	OC1A	Внешний выход А таймера Т1	PD5
	OC1B	Внешний выход В таймера Т1	PD4
Т2	TOSC1	Внешний вывод 1 для подключения резонатора таймера Т2	PC6
	TOSC2	Внешний вывод 2 для подключения резонатора для таймера Т2	PC7
	ОС2	Внешний выход таймера Т2	PD7

Рассмотрим работу таймера при его работе от источника тактового сигнала процессора, в качестве которого в лабораторном стенде выступает кварцевый резонатор с тактовой частотой 8 МГц.

Примечание: *для использования альтернативных функции портов соответствующие биты портов предварительно необходимо сконфигурировать на ввод или вывод. Например, при использовании альтернативной функции ОС0 для вывода информации из таймера Т0 вывод микросхемы РВ3 должен быть определен как выход: бит DDB3 равен 1*

Регистры таймера Т0

Счет импульсов источника частоты ведется в 8-разрядном счетном регистре таймера TCNT0 (TimerCouNTerT0). Перед тем, как попасть в схему счета импульсов, тактовый сигнал поступает на схему деления частоты, которая, в соответствии с параметрами, установленными при инициализации таймера, производит деление частоты тактового сигналаf_{CLK}в следующем соотношении:

- без делителя частоты тактового сигнала f_{CLK},
- с делителем частоты тактового сигналаf_{CLK}/8;
- с делителем частоты тактового сигналаf_{CLK}/ *64;*
- с делителем частоты тактового сигнала f_{CLK}/*256;*
- с делителем частоты тактового сигнала f_{CLK}/*1024.*

Каждый импульс с предделителя частоты считается и инкрементирует значение, содержащееся в счетном регистре TCNT0.

Каждый таймер содержит регистр сравнения. Таймер Т0 содержит 8- разрядный регистр OCR0

(OutputCompareRegister). В это регистр записывается уставка, то есть число от 0 до 255, при достижении которого счетным регистром TCNT0формируется флаг прерывания по совпадению таймера OCF0 (OutputCompareFlag).

В микроконтроллере Atmega8535 таймеры T0 и T2 работают в двух режимах широтно-импульсной модуляции: быстрый ШИМ и фазово-корректный ШИМ.

В режиме *быстрого ШИМ* счетный регистр TCNT0(2) таймера производит формирование пилообразной развертки (рис. 2, а), инкрементируя свое значение по каждому импульсу с предделителя, который устанавливается в регистре управления TCCR0(2) таймера T0(T2). При достижении счетным регистром значения 255 происходит его автоматическое обнуление.

В регистрах сравнения OCR0(2) таймеров T0 и T2 может быть записано любое число от 0 до 255. С этим числом сравнивается значение счетного регистра TCNT0(2) и при их равенстве происходит переключение вывода таймера OC0 или OC2 в соответствии с настройками регистра управления TCCR0(2) таймера. Вслучае, если при совпадении TCNT0(2)=OCR0(2) вывод OC0(2) таймера обнуляется, то ШИМ получается неинвертирующим (рис. 2, б), а если при совпадении вывод устанавливается в состояние логической «1», то ШИМ инвертируюший (рис2. в).

Рис. 1. Работа предделителя таймера T0 при коэффициенте делителя 8

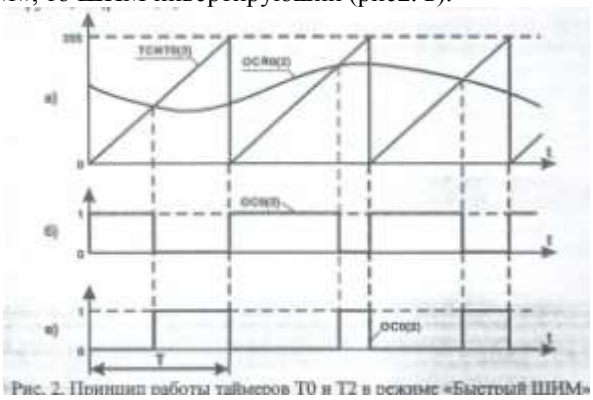


Рис. 2. Принцип работы таймеров T0 и T2 в режиме «Быстрый ШИМ»

В режиме фазового ШИМ счетный регистр TCNT0(2) формирует пилообразный сигнал развертки с нарастающим и спадающим фронтами. Сначала регистр инкрементирует свое значение от 0 до 255, а затем происходит егодекремент от 255 до 0 (рис. 3, а).

Если при превышении значения, содержащимся в счетном регистреTCNT0(2), значения, содержащегося в регистре сравнения OCR0(2) происходит обнуление вывода OC0(2) таймера, то ШИМ является неинвертирующим (рис.3, б). В противном случае ШИМ - инвертирующий (рис. 3, в).

Несмотря на то, что в режиме фазового ШИМ частота ШИМ-сигнала меньше, чем в режиме быстрого ШИМ, первый режим обладает большей разрешающей способностью и используется для достижения большей точности модуляции.

Регистры таймера TO в режиме ШИМ

Режим широтно-импульсной модуляции, как и все остальные режимы работы таймера T0, инициализируется в регистре управления TCCR0 (табл. 2).

Таблица 2. Регистр управления TCCR0 таймера

Бит	7	6	5	4	3	2	1	0
Название	FOCO	WGM00	COM01	COMOO	WGM01	CS02	CS01	CSC

За установку режима работы таймера отвечают биты WGM01:WGM00. При установке WGM01=0, WGM00=1 активируется режим фазового ШИМ, при установке WGM01=1, WGM00=1 активируется режим быстрого ШИМ.

При активации режима широтно-импульсной модуляции биты COM01 и COM00определяют поведение вывода таймера T0, в качестве которого выступает вывод PB3 микроконтроллера. Поведение вывода в соответствии с этими битами представлено в табл. 3 и табл. 4.

Таблица 3. Внешний выход OCO таймера TO в режиме быстрого ШИМ

COM01	COMO	Описание
0	0	Нормальная функция порта, вывод OCO отключен
0	1	Комбинация бит зарезервирована
1	0	Очистка OCO при совпадении. Неинвертирующий ШИМ
1	1	Установка OCO при совпадении. Инвертирующий ШИМ

Таблица 4. Внешний выход OCO таймера TO в режиме фазового ШИМ

COM01	COMO	Описание
0	0	Нормальная функция порта, вывод OCO отключен
0	1	Комбинация бит зарезервирована
1	0	Очистка OCO при совпадении при счете вверх. Неинвертирующий ШИМ
1	1	Установка OCO при совпадении при счете вверх. Инвертирующий ШИМ

Частота ШИМ-сигнала устанавливается битами CS02...CS00. Эти биты определяют источник задания частоты и коэффициент делителя (табл. 5).

Таблица 5. Установка источника задания частоты таймера T0

CS02	CS01	CS00	Описание

0	0	0	Таймер остановлен
0	0	1	$f_{T0}=f_{CLK}$ / 1

0	1	0	$f_{T0}=f_{CLK}$ / 8
0	1	1	$f_{T0}=f_{CLK}$ / 64
1	0	0	$f_{T0}=f_{CLK}$ / 256
1	0	1	$f_{T0}=f_{CLK}$ / 1024
1	1	0	Внешний сигнал на входе TO. Спадающий фронт сигнала.
1	1	1	Внешний сигнал на входе TO. Нарастающий фронт сигнала.

В соответствии с табл. 5, частота ШИМ рассчитывается следующим образом:

- для режима быстрого ШИМ: $f = f_{CLK} / 256$;

-для режима фазового ШИМ: $f = f_{CLK} /512$.

Регистры таймера Т2 в режиме ШИМ

Режим широтно-импульсной модуляции, как и все остальные режимы работы таймера Т2, инициализируется в регистре управления TCCR2 (табл. 6).

Таблица 6. Регистр управления TCCR2 таймера

Бит	7	6	5	4	3	2	1	0
Название	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS2I

Описание и назначение управляющих бит регистра TCCR2 таймера аналогичны описанным битам регистра TCCR0 таймера TO, поэтому отдельно комментируются. Частота ШИМ-сигнала устанавливается битами CS22...CS; Эти биты определяют источник задания частоты и коэффициент делите (табл. 7).

Таблица 7. Установка источника задания частоты таймера ’

CS22	CS21	CS20	Описание
0	0	0	Таймер остановлен
0	0	1	$f_{T0}=f_{CLK}$ / 1
0	1	0	$f_{T0}=f_{CLK}$ / 8
0	1	1	$f_{T0}=f_{CLK}$ / 32
1	0	0	$f_{T0}=f_{CLK}$ / 64
1	0	1	$f_{T0}=f_{CLK}$ /128
1	1	0	$f_{T0}=f_{CLK}$ / 256
1	1	1	$f_{T0}=f_{CLK}$ / 1024

Необходимо отметить, что в режиме ШИМ необязательно устанавливать маски и разрешать прерывания по совпадению и переполнению таймера Т0 и Т2. Эти прерывания следует активировать только при необходимости использования в программе.

Для запуска режима ШИМ на таймере Т0 или Т2 необходимо:

- остановить таймер обнулением регистра управления TCCR0(2);
- обнулить регистр сравнения OCR0(2) и счетный регистр TCNT0(2);
- - установить в регистре управления TCCR0(2) режим ШИМ и выбрать частоту его работы.
- В процессе работы ШИМ в любой момент можно изменять содержимое регистра сравнения OCR0(2) таймеров, при этом среднее значение ШИМ-сигнала на выводе ОСО и ОС2 микроконтроллера будет пропорционально изменяться.

Работа счетчика Т0 в режиме *«Очистка при совпадении»*

Каждый таймер содержит регистр сравнения. Таймер ТО содержит 8- разрядный регистр OCRO (OutputCompareregister). В это регистр записывается уставка, то есть число от 0 до 255, при достижении которого счетным регистром TCNT0 формируется флаг прерывания по совпадению таймера OCFO (OutputCompareFlag).

Когда значение, записанное в регистре TCNT0, переполняет максимальное значение, которое можно записать в 8-разрядный регистр сравнения OCRO, формируется флаг прерывания по переполнению таймера TOVO (TimerOverflowFlag) (рис. 2).

Рис. 1. Работа предделителя таймера Т0 при коэффициенте делителя 8

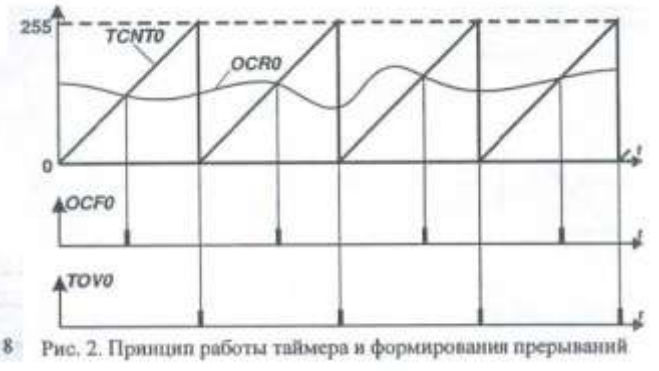


Рис. 2. Принцип работы таймера и формировании прерываний

Настройки таймера определяет регистр TCCRO (Timer/CounterControlregister).

Табл. 1. Управляющий регистр TCCR0 таймера TO

Бит	7	6	5	4	3	2	1	0
Название	FOC0	WGM00	COM01	COM00	WGM01	CS2	CS1	CS0

Бит 7 –FOC0- бит принудительной установки в единичное значение выходного сигнала таймера в режиме ШИМ (в данной работе не рассматривается).

Бит 6 и бит 3 –WGM00:WGM01 - биты управления режимом работы таймера. Эти биты определяют режим работы таймера: нормальный, ШИМ- режимы или сброс при совпадении (обратите внимание 6 бит регистра устанавливает младший разряд режима работы, 3 бит - старший разряд). Виды режимов работы представлены в табл. 2.

Табл. 2. Режимы работы таймера/Счетчика TO

WGM01	WGM00	Режим
0	0	Нормальный
0	1	Фазовый ШИМ
1	0	Очистка при совпадении
1	1	Быстрый ШИМ

В нормальном режиме работы счетный регистр TCNT0 изменяется от 0 до 255 и далее продолжает счет. В режиме *«очистка при совпадении»* значение уставки записывается в регистр OCR0. Когда значение счетного регистра TCNT0 доходит до уставкиOCR0, регистр TCNT0 автоматически очищается, а таймер - перезапускается. Этот режим очень удобен при необходимости периодического выполнения каких- либо операций (рис. 3).

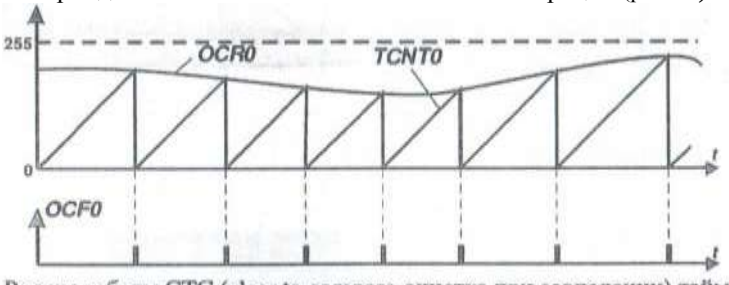


Рис. 3. Режим работы CTC (clear to compare-очистка при совпадении) таймера T0

Биты 5 и 4 – COM01:COM00 - биты управления выводом OC0. Таймер T0 может выводить сигнал на внешние контакты микросхемы, этот вывод обозначается OC0 и определяется как альтернативная функция вывода PB3 порта ввода/вывода B. Управление выводом OC0 выполняется комбинацией бит COM01:COM00 (табл. 3) и зависит также от режима работы таймера (биты WGM01:WGM00).

Табл. 3. Функции вывода OC0 таймера T0 в нормальном режиме работы

COM01	COMO	Описание
0	0	Нормальная функция порта, вывод OC0 отключен
0	1	Изменение OC0 на противоположное при совпадении OCF0
1	0	Очистка OCO при совпадении OCF0
1	1	Установка OCO при совпадении OCF0

Биты 3…0 - CS02…CS00 –биты пределителя таймера и источника задания частоты таймера (таблица 4.)

Табл. 4. Функции бит CS02, CS01, CS00 таймера T0

CS02	CS01	CS00	Описание
0	0	0	Таймер остановлен
0	0	1	$f_{T0}=f_{CLK}/1$
0	1	0	$f_{T0}=f_{CLK}/8$
0	1	1	$f_{T0}=f_{CLK}/64$
1	0	0	$f_{T0}=f_{CLK}/256$
1	0	1	$f_{T0}=f_{CLK}/1024$
1	1	0	Внешний сигнал на входе TO. Спадающий фронт сигнала.
1	1	1	Внешний сигнал на входе TO. Нарастающий фронт сигнала.

Помимо управляющего регистра **TCCR0,** счетного регистра **TCNT0** и регистра сравнения **OCR0,** в микроконтроллере содержатся регистр масок прерываний таймеров **TIMSK**и регистр флагов прерываний таймеров и **TIFR.** Регистры позволяют отслеживать то или иное событие (совпадение, переполнение, захват), происходящее во всех таймерах микроконтроллера.

Регистры **TIMSK**и **TIFR**общие для всех таймеров, поэтому они содержат установки для всех таймеров (TO, T1 и T2) микроконтроллера.

Регистр маски прерываний таймеров **TIMSK**(табл. 5) разрешает то или иное прерывание того или иного таймера. Так, например:

- 0 разряд TOIE0регистра установлен в единицу, это означает, что разрешается прерывание «Переполнение

таймера Т0»;

- 1 разряд OCIE0 регистра установлен в единицу, это означает, что разрешается прерывание «Совпадение таймера Т0» и т.д.

Табл. 5. Регистр масок прерываний таймеров **TIMSK**

Бит	7	6	5	4	3	2	1	0
Название	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0
Наименование бита разрешения прерывания	Совпадение таймера Т2	Переполнение таймера Т2	Захват таймера Т1	Совпадение А таймера Т1	Совпадение В таймера Т1	Переполнение таймера Т1	Совпадение таймера Т0	Переполнение таймера Т0

-

Регистр флагов прерываний таймеров **TIFR**(табл. 6) показывает какое прерывание произошло.
Например, для таймера ТО в регистре **TIFR**используются два бита:

- бит 1 –OCF0- флаг прерывания по совпадению таймера;

бит 0 –TOV0- флаг прерывания по переполнению таймера.

Табл. 6. Регистр флагов прерываний таймеров **TIFR**

Бит	7	6	5	4	3	2	1	0
Название	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0
Наименование флага прерывания	Флаг совпадения таймера Т2	Флаг переполнения таймера Т2	Флаг захвата таймера Т1	Флаг совпадения А таймера Т1	Флаг совпадения В таймера Т1	Флаг переполнения таймера Т1	Флаг совпадения таймера Т0	Флаг переполнения таймера Т0

Значение этих регистров следующее. Когда возникает переполнение или совпадение таймера Т0, автоматически формируется флаг прерывания TOV0или OCF0 в регистре TIFR. Если в регистре масок прерываний таймеров TIMSKна соответствующее прерывание наложена маска, то вызывается процедура обработки прерывания. Если же маска прерывания не наложена, то при совпадении или переполнении таймера ничего не происходит.

8-разрядный таймер/счетчик Т2

Принцип и режимы работы таймера Т2 практически не отличаются от рассмотренного ранее таймера Т0, за исключением следующего:

а) изменены коэффициенты предлелителя и источник задания частоты;

б) в таймере Т2 предусмотрен и асинхронный режим работы. Этот режим заключается в том, что если таймер ТО получает тактовый сигнал либо от источника частоты микроконтроллера f_{CLK}, либо от внешнего источника тактирующего сигнала, подающегося на вывод Т0 (РВ0), то таймер Т2 может работать в асинхронном режиме работы, будучи запитанным от отдельного источника частоты (асинхронный режим). В этом режиме тактовый сигнал поступает на выводы TOSC1, TOSC0микроконтроллера. Асинхронный режим таймера Т2 при проведении лабораторных работ не используется, теоретические аспекты его использования могут быть изучены студентами самостоятельно.

Таймер Т2, как и таймер Т0, содержит 8-разрядный управляющий регистр TCCR2, регистр счета TCNT2 и регистр сравнения TCNT2. Назначение этих регистров подобно назначению соответствующих регистров таймера Т0.

Табл. 7. Управляющий регистр TCCR2

Бит	7	6	5	4	3	2	1	0
Название	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20

Бит 7 - FOC2- бит режима принудительной установки выходного сигнала таймера в режиме ШИМ.

Бит 6 и бит 3 -WGM20:WGM21-биты режима работы таймера Т2 (табл.8).

Табл. 8. Режимы работы таймера Т2

WGM2	WGM2	Режим
0	0	Нормальный
0	1	Фазовый ШИМ
1	0	Очистка при совпадении
1	1	Быстрый ШИМ

Биты 5 и 4 - COM21:COM20 - режим подключения вывода ОС2 таймера Т2 в качестве альтернативной функции вывода PD7 порта ввода/вывода D. Таймер Т2 может управлять этим выводом в соответствии с выбранным режимом работы, определяемым битами WGM21:WGM20, а также в соответствии с комбинацией разрядов COM21:COM20 (табл. 9)

Табл. 9. Функции вывода OC2 таймера T2 в нормальном режиме работы

C	C	Пояснение
0	0	Вывод OC2 отключен
0	1	Изменение OC2 на противоположное при совпадении OCF2
1	0	Очистка OC2 при совпадении OCF2
1	1	Установка OC2 пои совпадении OCF2

Биты 3...0 - CS22...CS20- определяют источник задания частоты таймера 12 и предделитель таймера (табл. 10).

Табл. 10. Функции бит CS22, CS21, CS20 таймера T2

C	C	C	Пояснение
0	0	0	Таймер остановлен
0	0	1	$f_{T0}=f_{CLK}/1$
0	1	0	$f_{T0}=f_{CLK}/8$
0	1	1	$f_{T0}=f_{CLK}/32$
1	0	0	$f_{T0}=f_{CLK}/64$
1	0	1	$f_{T0}=f_{CLK}/128$
1	1	0	$f_{T0}=f_{CLK}/256$
1	1	1	$f_{T0}=f_{CLK}/1024$

Источник тактового сигнал выбирается в регистре ASSR (AsinchronousStatusRegister).

Табл. 11. Регистр асинхронного режима таймера T2

Бит	7	6	5	4	3	2	1	0
Название	-	-	-	-	AS2	TCN2UB	OCR2UB	TCR2UB

Бит 3 - AS2определяет, в каком режиме работает таймер. Если бит AS2=0, то таймер работает в синхронном режиме, используя в качестве тактового сигнала источник частоты микропроцессора. В случае, когда AS2=1, таймер получает тактовый сигнал от внешнего кварцевого резонатора, подключенного к выводам TOSC1, TOSC0микроконтроллера.

Биты 2, 1, 0 - используются в асинхронном режиме и в данной работе не рассматриваются.

В регистре флагов таймеров TIFR(табл. 5) таймер T2 имеет два бита: бит 7 OCF2 (флаг прерывания сравнения таймера T2); бит 6 TOV2 (флаг прерывания по переполнению таймера T2).

В регистре масок прерываний таймеров TIMSK(табл. 6) таймер T2 имеет два бита: бит 7 OCIE2 (маска прерывания сравнения таймера T2); бит 6 TOIE2 (маска прерывания по переполнению таймера T2).

Контрольные вопросы

1. Сколько таймеров содержит микроконтроллер ATmega8535?
2. Какие из них 8-ми разрядные, 16-ти разрядные?
3. Какие регистры определяют работу таймера T0?
4. Поясните назначение регистра управления в целом?
5. Какие биты изменяют режим работы таймера T0? Режим работы вывода?
6. Как установить коэффициент предделителя таймера T0 равный 256?
7. Объясните назначение регистров TIMSKи T1FR.
8. Какие функции в программе могут выполнять таймеры?
9. Перечислите регистры ввода/вывода, управляющие работой таймера TO?
10. В каких режимах с ШИМ могут работать таймеры микроконтроллера ATmega8535?
11. Как настроить таймеры T0/T2 для работы в режиме быстрого ШИМ?

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Ответы на контрольные вопросы
3. Вывод по работе

Лабораторная работа № 11
Организация работы 8-ми разрядного таймера в режиме ШИМ

Формируемые компетенции:

ПК 2.1. Создавать программы на языке ассемблера для микропроцессорных систем.

Цель работы: научиться применять таймеры в режиме ШИМ.

Выполнив работу, Вы будете:

уметь:

- составлять программы на языке ассемблера для микропроцессорных систем

Материальное обеспечение:

Стенд «Микроконтроллер», ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

Рассмотрим применение таймера на примере программы, в которой таймер TO работает в режиме быстрого

ШИМ. В зависимости от состояния битов порта А регулируется яркость светодиода, подключенного к выходу ШИМ таймера ТО. Логическим сигналом с 0 бита порта DШИМ изменяется с инвертирующего на неинвертирующий

```
;-----------------------------------------------------------------
;ШИМ на таймере T0
;входы:
;    PORTA0...PORTA7     - задание уставки таймера
;    PORTD0              - инвертирующий (1)/неинвертирующий (0) ШИМ
;выходы:
;    PORTB3              - ШИМ на таймере T0

.include"m8535def.inc"   ;подключение стандартной библиотеки
.cseg                    ;начало сегмента кода
.org$0                   ;по адресу 0
rjmp reset               ;и переход на reset

reset:
   ldi r16,0x01      ;инициализация портов ввода вывода.
   out PORTD,r16     ;PORTD0 - на ввод информации
   clr r16
   out DDRD,r16
   ldi r16,0xFF
   out PORTA,r16     ;PORTA - на ввод информации
   clr r16
   out DDRA,r16
   out PORTB,r16
   ldi r16,0x08
   out DDRB,r16      ;PORTB3 - на вывод информации
   clr r16
   out TCCR0,r16     ;сброс регистра управления таймера T0
   out OCR0,r16      ;сброс регистра сравнения таймера T0
   out TCNT0,r16     ;сброс регистра счета таймера T0
   ldi r16,0x69      ;установка режима быстрого неинвертирующего
   out TCCR0,r16     ;ШИМ таймера T0
main:                ;начало основной программы
   in r16,PINA       ;считывание значений порта ввода/вывода PORTA
   out OCR0,r16      ;и его отправка в регистр сравнения таймера T0
   in r16,PIND       ;считывание значений порта ввода/вывода PORTD
   andi r16,0x01     ;и выделение бита PORTD0
   cpi r16,0x01      ;Если PORTD0=1
   breq met1         ;то переход на метку met1
   ldi r17,0x69      ;иначе запись в r17 значения для неинверт. ШИМ
   rjmp met2         ;и переход на метку met2
met1:                ;По метке met1
   ldi r17,0x79      ;запись в r17 значения для инвертирующего ШИМ


met2:                ;По метке met2
   out TCCR0,r17     ;установка заданного режима работы таймера T0
   rjmp main         ;и возврат на main
;-----------------------------------------------------------------
```

Рассмотрим программу более подробно.

1. Инициализации портов ввода/вывода в соответствии с поставленным заданием:

ldir16,0x01
out PORTD,r16
clr rl6
out DDRD,rl6
ldi rl6,0xFF
out PORTA, rl6
clr rl6
out DDRA,rl6
out PORTB,rl6
ldi rl6,0x08
out DDRB,rl6

2. Инициализация таймера ТОна заданный режим работы:

clr rl6
out TCCRO, rl6
out OCRO,rl6
out TCNTO,rl6
ldirl 6, 0x69
out TCCRO,rl6

Сначалаочищаютсявсерегистрытаймера(clr r16; out TCCRO, r16; out OCR0, rl6; out TCNTO, rl6). После этого в регистр управления TCCR0, в соответствии с табл. 1 записываются необходимые комбинации управляющих бит. Установкой WGM01=1, WGM00=1 выбирается режим быстрого ШИМ; установкой COM01=1, COM00=0 выбирается режим неинвертирующего ШИМ; биты CS02:CS01:CS00=001 определяют работу таймера без

предlелителя частоты процессора clk/1.

3. В начале цикла происходит считывание данных с порта А и их запись в регистр сравнения таймера Т0:

```
main:
    in rl6,PINA
    out OCR0,rl6
```

4. После этого производится опрос порта Dи выделение младшего значащего бита:

```
    inr16,PIND
    andir16,0x01
```

5. Сравнение PORTDOс логическими уровнями 0 и 1:

```
    cpir16,0x01
    breqmetl
    ldirl7,0x69
    rjmpmet2
metl:
    ldirl7,0x79
```

Если в результате сравнения (cpiri6,0x0i) PORTDO=1, то происходит переход на метку metl, по которой в регистр r17 записывается значение, которое необходимо записать в регистр управления TCCR0таймера Т0 для реализации инвертирующего ШИМ (ldir17,0x79). Иначе в r17 производится запись значения TCCROдля неинвертирующего ШИМ (ldiП7, 0x69).

6. При переходе на метку met2 происходит запись полученного в r17 значения в регистр управления TCCROтаймера ТО и зацикливание программы:

```
met2:
    out TCCRO,rl7
    rjmp main
```

Подробное рассмотрение программы показывает, что при использовании в таймере режима ШИМ пользователь может не использовать прерывания по совпадению и переполнению таймера, так как в режиме ШИМ все действия таймер делает автоматически. Это существенно облегчает организацию программы и уменьшает ее размер.

Задание на выполнение

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 (Условные обозначения: Б - быстрый ШИМ, Ф - фазовый ШИМ, П - прямой ШИМ, И - инверсный)

№ вар	Таймер (ы), канал (ы)	Режим	Диапазон частот ШИМ	Входы (биты)	Биты задания, количество дискретных значений, диапазон изменения скважности
1	Т2	Б/ПиИ	[1000, 10000]	РА7-переключает между ПиИ	РС5...РС7, 8 положений, 0...0,5
2	Т0 и Т2	Ф/И	[1000, 10000]	РАО- включает таймер ТО или Т2	РА4...РА7,16 положений, 0,1.. .0,9
3	Т0	Б/И	[100,1000]	РС5 - меняет частоту ШИМ	Порт А, 256 положений, 0... 1
4	Т0 и Т2	Ф/ПиИ	[200,1000]	Одновременно работают 2 канала	РС0...РС3, 16 положений, 0...0,75
5	Т2	БиФ/ ПиИ	<200	РАО-БиФ РА7 -ПиИ	РС0...РС4, 32 положения, 0,2...0,8
6	Т0 иТ2	Б/П	>10000	РАО - вкл ТО РА1 - вкл Т2	РВ0...РВ2, 8 положений, 0...1
7	Т0	Ф/И	>10000	РСО-вкп/откл	РАО...РАЗ, 16 положений, 0...1
8	Т0	ФиБ/И	<200	РА4 - переключает между ФиБ	Порт С, 256 положений, 0...1
9	Т2	Б/П	[1000, 10000]	РС1 - изменяет частоту ШИМ	РА2. . .РАЗ, 4 положения, 0...0.5
10	Т0 и Т2	Ф/ПиИ	[1000, 10000]	РС7 - выбирает ТО или Т2	РС0...РС5, 64 положения, 0.:.1
11	Т0 и Т2	Б/И	[200, 1000]	РА0-0, РА1-Т0, РА2-Т2,РАЗ-ТО и Т2	РС4...РС7, 16 положения, 0... 1
12	ТО	Ф/П	[30, 35000]	РС0...РС2 - 6 частот ШИМ	Порт А, 256. положения, 0...0.5
13	Т2	Б/ПиИ	<200	РАО - выбор ПиИ	РА2...РА5, 16 положений, 0...1
14	ТО и Т2	Ф/П	>10000	РА0=0-ТО и Т2 выкл РА0=1-ТО и Т2 вкл	РСЗ...РС5, 8 положений, 0... 0.75

15	ТО	Б/И	>10000	РС7- выкл/вкл	РВ0...РВ3, 16 положений, 0...0.8
16	ТО и Т2	Ф/ПиИ	<200	РАО- меняет режим сПнаИ ТО и Т2 одновременно	РАЗ...РА7,32 • положения, 0;..1
17	Т2	Б/И	[1000, 10000]	РВЗ - вкл/выкл выход таймера	РВ0...РВ2,'8 положений. 0...1
18	Т0иТ2	ФиБ/ ПиИ	[1000, 10000]	РА7- ФиБ РА6- ПиИ	РАО...РАЗ, 16 положений, 0...0.5
19	ТО и Т2	Б/П	[200, 1000]	РА4-Т0 РА5-Т2	РС1...РСЗ,? положений, 0,1.. .6.9
20	ТО	ФиБ/И	[200, 1000]	РАО - вкл/выкл ТО РА1 - Б и Ф	РА4...РА7,16 положений, 0,25... 1
21	Т2	Б/ПиИ	<200	РСО-ПиИ	РВ0...РВ4,32 положений, 0... 1
22	ТО иТ2	Ф/ПиИ	>10000	РАЗ-вкл ТО или Т2	РАЗ...РА7,32 ' положений, 0...0.8
23	ТО	Б/И	>10000	РАО- частота ШИМ	РС, 256 положений, 0...1
24	Т0иТ2	Ф/ПиИ	<200	РАО - выбор ТО или Т2, РА1 - выбор П или И	РА5...РА7,8 положений, 0,2...0,9
25	ТО и Т2	ФиБ/ ПиИ	[1000, 10000]	РС6 - выбор Ф или В, РС7 - выбор П или И	РВ0...РВ2, 8 положений, 0...0.75

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Выполненное задание с комментариями (работоспособность программы демонстрируется на стенде):
- исходное задание;
- функциональную схему;
- представить расчет скважности, настройку таймера
- запись данных (скважности) в ОЗУ или ПЗУ;
- листинг программы;
- дизассемблированную программу;
- алгоритм;
- представить таблицу значений стека во время исполнения программы.
3. Вывод по работе

Лабораторная работа № 12

Организация работы 8-ми разрядного таймера в режиме создания временных интервалов

Формируемые компетенции:

ПК 2.1. Создавать программы на языке ассемблера для микропроцессорных систем.

Цель работы: научиться применять таймеры в режиме создания временных интервалов.

Выполнив работу, Вы будете:

уметь:

- составлять программы на языке ассемблера для микропроцессорных систем

Материальное обеспечение:

Стенд «Микроконтроллер», ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

Рассмотрим применение таймера на примере программы, осуществляющей инкремент какого-либо регистра до значения, заданного на входе порта D, при этом содержимое регистра выводится с интервалом 500 мс в порт С.

```
;-----------------------------------------------------------------
;Программа вывода в PORTC с периодом 500 мс увеличивающегося значения
;двоичного кода до значения, заданного на входах PIND.
;Входы:
;PD0_PD7 - задание максимального значения числа
;PC0_PC7 - индикация результата
.include"m8535def.inc"   ;подключение библиотеки контроллера
.cseg                    ;начало сегмента кода
.org $0                  ;по адресу 0
   rjmp reset            ;переход на метку reset
.org $13                 ;адрес обработки прерывания по совпадению T0
   rjmp T0_compare       ;и переход по этому адресу в случае
                         ;возникновения прерывания
         ; инициализация стека
reset:
   ldi r16,low(RAMEND)   ;запись в указатель стека адреса конца
   ldi r17,high(RAMEND)  ;памяти данных
   out spl,r16
   out sph,r17
         ; инициализация портов
   ldi r16,0xFF          ;инициализация портов ввода/вывода:
   out PORTD,r16         ;порт D - на ввод
   out DDRC,r16          ;порт C - на вывод
   clr r16
   out DDRD,r16
   out PORTC,r16
         ; инициализация и запуск таймера T0
   ldi r16,0x00          ;обнуление регистров таймера T0:
   out TCCR0,r16         ;регистра управления TCCR0
   out TCNT0,r16         ;регистра счета TCNT0
   ldi r16,0x4E          ;запись в регистр сравнения OCR0 числа 0x4E
   out OCR0,r16          ;соответствующего уставке 10 мс
   ldi r16,0x05          ;запуск таймера T0 с предделителем clk/1024
   out TCCR0,r16         ;в нормальном режиме работы
         ; разрешение работы прерывания
   ldi r16,0x02          ;для работы прерывания по совпадению T0
   out TIMSK,r16         ;накладывается маска OCIE0 в регистре TIMSK
   clr r16               ;необходимые в программе регистры
   clr r17               ;предварительно очищаются
   clr r18
   sei                   ;глобальное разрешение прерываний
main:                    ;основная часть программы
   rjmp main             ;пустой цикл
         ; обработчик прерывания
T0_compare:              ;при совершении прерывания по совпадению T0
   cli                   ;глобальный запрет прерываний
   in r24,SREG           ;сохранение регистра SREG
   clr r16               ;обнуление счетного регистра TCNT0
   out TCNT0,r16
   inc r17               ;инкремент РОН r17
   cpi r17,0x32          ;сравнение r17 с уставкой 0x32 (50)
   brne quit             ;если r17 не равен уставке, то переход на
   quit
   clr r17               ;иначе очистка r17
   in r16,PIND           ;ввод данных, содержащихся на PIND
   cp r16,r18            ;и сравнение РОН r16 с регистром счета r18
   brpl met_1            ;если r16>r18, то переход на met_1
   clr r18               ;иначе очистка регистра счета r18
met_1:
   out PORTC,r18         ;далее - вывод на индикацию r18
   inc r18               ;и инкремент регистра счета
quit:
   out SREG,r24          ;восстановление регистра SREG
   sei                   ;глобальное разрешение прерываний
   reti                  ;выход из подпрограммы прерываний
;-----------------------------------------------------------------
```

Рассмотрим особенности программы.

1. Строками программы

.org$13

rjmp T0_compare

выполняется инициализация адреса памяти, по которому начинаетсяподпрограмма обработки прерывания по совпадению таймера T0. Адреса подпрограмм обработки прерываний для каждого микроконтроллера заданы жестко и изменению не подлежат. Таблица адресов приведена в приложении.

2. В программе используется стек. Это связано с использованием в программе прерывания и необходимости сохранения в стеке адреса возврата из прерывания в основную программу. Поэтому вначале программыинициализируются регистры указателя стека SPH:SPL.

3. Для использования таймера ТО вначале необходимо выполнить его инициализацию. Для этого выполняются:

- обнуление счетного регистра TCNT0 и регистра управления TCCR0;
- устанавливается значение уставки в регистр сравнения OCRO.

Выполним расчет уставки. В случае данной программы получить уставку500 мс на таймере не получается, так как даже при предделителеclk/1024 максимальная уставка таймера равняется:

$$T_{уст} = \left(\frac{8000000}{1024}\right)^{-1} \cdot 256 = 32{,}768 \text{ мс.}$$

Как реализовать на таймере ТО уставку большую значения 32,768 мс? Один из вариантов считать количество прерываний таймера. Например, для получения уставки 500 мс можно запрограммировать таймер на уставку 10 мс и производить подсчет срабатываний таймера — каждое 50-е срабатывание таймера будет соответствовать времени Туст= 50*10мс= 500мс. Рассчитаем значение уставки таймера в 10 мс при предделителеclk/1024:

$$OCR0 = 10 \cdot 10^{-3} \cdot \frac{8000000}{1024} = 78{,}125.$$

Таким образом, округленное значение, записываемое в регистр сравнения OCR0, равно 78 (0x4E).

- запуск таймера. Выполняется установкой коэффициента предделителя равного 1024, после этого таймер запускается в работу в нормальном режиме.

4. Разрешение работы прерывания по совпадению таймера Т0. Для этого:

- в регистре масок прерываний таймеров TIMSKуставливается в единичное значение 1 бит для разрешения прерывания по совпадению таймера;
- разрешается работа всех прерываний (инструкция sei) путем записи логической «1» в старший бит регистра SREG.

Основная программа

5. Основной цикл mainпустой, т.к. не содержит никаких операторов.

6. Выполнение обработчика прерывания. При наступлении прерывания по совпадению Т0 осуществляется переход из любого места основной программы (в данном случае только из строки rjmpmain) по адресу обработки прерывания, указанному в строке rjmpto_compare. После этого:

- запрещаются все прерывания (cli); которые разрешаются только по окончании обработки прерывания;
- сохраняется значение регистра SREG (inr24, SREG), который при выходе из подпрограммы восстанавливается (outSREG, R24);
- обнуление счетного регистра таймера Т0;
- счет количества срабатываний прерывания таймера. Регистр r17 постоянно инкрементируется и если его значение не совпадает с уставкой0x32 (десятичное значение равно 50) (инструкция cpir17,0x32) происходит выход из подпрограммы обработки прерывания (brnequit);
- при совпадении значения регистрar17 с уставкой0x32, т.е. каждые 500 мс счетчика прерываний обнуляется (clrr17), считываются данные с порта D (inrl6, PiND), выполняется сравнение с регистром счета r18 (cp r 16, r 18).
- если уставка, заданная на PIND, меньше, чем текущее значение r18, то r18 обнуляется. После этого значение r18 выводится на PORTC (outPORTC,r18) и значение этого регистра инкрементируется;
- в конце подпрограммы вновь разрешаются прерывания (инструкция sei) и завершается обработка прерывания (инструкция reti) и выполняется.

Задание на выполнение

1. С использованием таймера Т0 составить программу «бегущий огонь» по вариантам с включением заданных выходов портов, периодами их включения и изменениями в тактах работы.

Варианты программы «бегущий огонь»

Примечание: символ «-» обозначает что такт отсутствует, например, в 1 варианте программа выполняется только за 3 такта.

№ вар.	1 такт	2 такт	3 такт	4 такт
1	PD0, PD1 – 2 с	PD1, PD2 -1 с	PD2, PD3 -0,5 с	–
2	PA0…PA3 -0,1 с	PA4…PA7 – 0,2 с	PC0…PC3 -0,3 с	PC4…PC7 – 0,4 с
3	PC0 – 0,5 с	PC2 – 0,5 с	PC4 – 1,5 с	PC6 – 1,5 с
4	PB0 – 1 с	Нет – 0,5 с	PB1 – 1 с	Нет – 0,5 с
5	PC7 – 5 с	PC6, PC7 – 5 с	PC5…PC7 – 5 с	PC4…PC7 – 5 с
6	PA0 – 2 с	PA1 – 4 с	–	–
7	PC0…PC3 - 10 с	PC1…PC4 -10 с	PC2…PC5 - 10 с	–
8	PD7, PC7 -0,5 с	Нет – 2 с	PD0, PC0 - 0,5 с	Нет - 2 с
9	PC0 – 0,2 с	PC1 – 0,2 с	PC2 – 0,2 с	Нет – 1 с
10	PA0…PA3 – 0,1 с	Нет – 0,2 с	PA4…PA7 – 0,1 с	Нет – 0,2 с
11	Порт D – 5 с	Нет – 1 с	Порт C – 5 с	–
12	PA3 – 2 с	Нет – 4 с	–	–
13	PC7 – 1 с	Нет – 5 с	PC6 – 1 с	Нет – 5 с
14	PA0…PA3 – 2 с	Нет – 1 с	PD0…PD3 – 2 с	
15	PC0…PC3 – 0,1 с	PC1…PC3 -0,1 с	PC2…PC3 – 0,1 с	PC3 – 0,5 с
16	Порт D – 2 с	Порт A – 2 с	Нет – 4 с	
17	PC0… PC2 - 2 с	PC1…PC3 - 2 с	PC2…PC4 - 2 с	Нет – 5 с
18	Порт A -0,1 с	Нет – 1 с	Порт C -0,1 с	Нет – 1 с
19	PC0 – 0,1 с	PC1 – 0,1 с	PC2 – 0,1 с	Нет – 0,5 с
20	PB0 – 5 с	PB1 – 2,5 с	PB2 – 1,25 с	Нет – 5 с
21	PD7 – 1 с	PD6 – 1 с	PD0…5 – 5 с	–
22	PA6 – 1 с	PA7 – 4 с	–	–
23	PC0, PC3 – 2 с	PC1, PC4 – 2 с	Нет – 5 с	–
24	PD4…PD7 – 2,5 с	Нет – 5 с	PD0…PD3 – 2,5 с	Нет – 5 с
25	Нет – 1,5 с	Порт C – 0,5 с	Нет – 1,5 с	Порт A – 1,5 с
26	PB0 – 5 с	Нет – 10 с	–	–
27	PC6,7 – 2 с	PC4, PC5 – 2 с	PC2…PC3 – 2 с	PC0…PC1 – 4 с
28	PA0…PA3 – 2 с	PA4…PA7 – 2 с	PA0…PA7 – 2 с	Нет – 6 с
29	PD0,PD2,PD4 – 5 с	PD1, PD3,PD5 – 5 с	Нет – 10 с	–
30	PC0 – 1,5 с	PC1 – 3 с	PC2 – 4,5 с	–

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Выполненное задание с комментариями (работоспособность программы демонстрируется на стенде):
- исходное задание;
- функциональную схему;
- листинг программы; дизассемблированную программу;
- блок-схему алгоритма;
- представить настройку таймера и расчет времени задержки реализованной на таймере;
- представить таблицу значений стека во время исполнения программы.
3. Вывод по работе

Практическая работа № 14
Изучение работы АЦП МК Atmega8535.

Формируемые компетенции:

ПК 2.1. Создавать программы на языке ассемблера для микропроцессорных систем.

Цель работы: изучитьработу 10-ти разрядногоАЦП МК.

Выполнив работу, Вы будете:

уметь:

- составлять программы на языке ассемблера для микропроцессорных систем

Материальное обеспечение:

ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

При использовании микроконтроллера в качестве устройства управления каким-либо процессом часто возникает необходимость оценивать величины налоговых 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 микроконтроллер Atmega8535 и другие контроллеры семейства AVRфирмы 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 ЩП - 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 AREFмикросхемы контроллера.

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

$$Umin = \frac{1}{2^n} \cdot Uref$$

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

$$Umin = \frac{1}{2^{10}} \cdot 5 = 4{,}88 \text{ В}$$

Ввиду того, что в качестве АЦП в микроконтроллерах AVRиспользуется АЦП последовательного приближения, процесс преобразования аналогового сигнала в пропорциональный ему цифровой код занимает некоторое время.Упрощенная структура АЦП последовательного приближения представлена ни рис. 1 АЦП состоит из компаратора, регистра последовательного приближения н 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..tl). Компаратор при этом измеряет разницу между входным сигналом и сигналом с выхода ЦАП.Если напряжение на входе АЦП оказывается больше, чем установленное навыходе 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 00000000 (рис. 2, интервал tl-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, итервалt2-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 AVRвремя преобразования АЦП можно регулировать, для этого в структуру преобразователя встроен предделитель, подобный тому, который встроен в таймеры T0...T2 микроконтроллера. Необходимо отметить, что уменьшением времени преобразования уменьшается точность получаемого результата. Рекомендуемая частота работы АЦП находится в пределах 0...200 кГц. При работе преобразователя в этом диапазоне обеспечивается минимальная погрешность при получении результата.

Рис. 1. Структура АЦП последовательного приближения

Рис. 2. Принцип работы АЦП последовательного приближения

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 микроконтроллераAtmega8535 содержит8 входов, подключенных к выводам порта ввода/вывода А. 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 6и 7 - REFS1, REFS0. С помощью этих бит определяется источник упорного напряжения АЦП (табл. 2).

Табл. 2.3адание источника опорного напряжения АЦП

REFSl	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	Комбинациянеиспользуется.
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 ипользуются два 8-разрядных регистра ADCHи ADCL, некоторые биты регистра ADCHостаются 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 отуправляющего 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 извходов АЦП подключен к преобразователю. Здесь различают два режима работы входов АЦП - одиночный и дифференциальный (табл. 4).

При решении типовых задач обычно используется одиночное включение ходов (табл. 4), однако в некоторых случаях (например, для исключения и (действия на входы микроконтроллера сигнала помехи) возникает необходимость использования дифференциальных входов.В этом случае АЦП (меряет 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 напряжения 1,22В(MUX4...0=11110) или общую точку

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

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

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

Бит 5 - ADATE. Записью логической «1» в этот бит разрешается автоматический запуск АЦП по событию. Событие выбирается комбинацией битов ADTSв регистре SFIORмикроконтроллера.

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=l, то при появлении флага готовности результата ADIFбудет сгенерировано прерывание.

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

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	–		
11111	0В			

Табл. 5 Регистр управления АЦП ADCSRA

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

Табл. 6 Предделитель АЦП микроконтроллера Atmega 8535

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 источик 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	ADTS	ADTS0	Источникавтозапуска
0	0	0	Непрерывноепреобразование АЦП
0	0	1	Аналоговыйкомпаратор
0	1	0	Внешнеепрерывание 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

Контрольные вопросы

1. Что такое АЦП?
2. Какого типа АЦП используется в микроконтроллере Atmega 8535?
3. Сколько разрядов имеет АЦП МК?
4. Что такое опорное напряжение?
5. Что такое разрешающая способность АЦП?
6. Как рассчитать разрешающую способность АЦП (минимальное напряжение)?

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Ответы на контрольные вопросы
3. Вывод по работе

Лабораторная работа № 13
Организация работы АЦП МК Atmega 8535

Формируемые компетенции:

ПК 2.1. Создавать программы на языке ассемблера для микропроцессорных систем.

Цель работы: научиться применять АЦП МК.

Выполнив работу, Вы будете:

уметь:

- составлять программы на языке ассемблера для микропроцессорных систем

Материальное обеспечение:

Стенд «Микроконтроллер», ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

Рассмотрим применение АЦП на примере программ.

Пример 1. Подать аналоговый сигнал на вход ADC2 АЦП и вывести 8-разрядный результат на порт ввода/вывода С. 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 А
   out PORTA,r16     ;на ввод информации
   clr r16           ;Используемый вход АЦП не инициализируется
   out DDRA,r16
   out PORTC,r16     ;Инициализация порта С 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 rl6,low(RAMEND)
        ldir17, high (RAMEND)
        out spl,rl6
        out sph,rl7
```

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-избежаниенаведения на них помех. Порт C инициализируется на вывод информации:

```
        ldir16,0xFB
        out PORTA,rl6
        clr rl6
        out DDRA,rl6
        out PORTC,rl6
        ser rl6
        out DDRC,rl6
```

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

```
        ldir16, 0x62
        out ADMUX,rl6
        ldi rl6,0xC7
        outADCSRA,rl6
```

Вкачестве источника сигнала опорного напряжения выбирается напряжение электропитания процессора, поданное на вывод AVCC. Также выбирается «левое» выравнивание результата преобразования ADLAR. Выбирается второй канал АЦП(ldi г **1**6, 0**x**62; outADMUX,r16**).**В регистреADCSRAустанавливаются биты ADEN(включение АЦП), ADSC(запуск преобразования АЦП), а также выбирается предделительclk/128 для повышения точности преобразования.

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

```
main:
        in rl6,ADCSRA
        SBRC rl6,4
        rcallADC_ready
        rjmpmain
```

Сначала считывание значение регистра ADCSRAвРОНrl6 **(**inr16,ADCSRA)После этого опрашивается 4-й бит этого регистра, соответствующий флагу ADIF**(**SBRC**R**16,4). Если бит чист, то пропускается следующая за директивой **SBRC**инструкция. Иначе происходит переход на метку ADC_ready(rcallADC_ready).

5. По метке ADC_readyпроисходит обработка результата преобразованииАЦП:

```
ADC_ready:
        in rl6,SREG
        ldi rl7,0x97
        out ADCSRA,rl7
        in rl7,ADCH
        outPORTC,rl7
        ldi rl7,0xC7
        out ADCSRA,rl7
        out SREG,rl6
        ret
```

При переходе по метке ADC_readyсначала вРОНr16 необходимо сохранить значение регистра SREG (inr16,SREG).После этого необходимо вручную сбросить флаг готовности преобразования АЦП записью в него логической «1» (ldir17,0x97; outADCSRA,R17).Необходимо запомнить, что все флаги в микроконтроллерах AVRсбрасываются записью в них логической «1».Далее **в**РОНr17 считывание результат преобразования АЦП, хранящийся в ADCH(inr17,ADCH).Два младших бита, хранящиеся в ADCL, отбрасываются. Содержимое r17 передается в порт ввода/вывода C(out**PORTC,** rl7), осуществляется перезапуск АЦП, восстановление регистра SREG (outSREG, 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
 jmp reset               ;осуществляется переход на reset
 org$0E                  ;При переходе по адресу $0E
 jmp ADC_ready           ;осуществляется вызов функции обработки
                         ;прерывания готовности АЦП.
 eset:
  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
ain:                     ;Главный цикл программы
 rjmp main               ;выполнен пустым

DC_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
et1:
 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
rjmpADC_ready
```

При попадании на эти векторы происходит переход на соответствующие метки resetи 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
    ldir16, low(RAMEND)
    ldi rl7,high(RAMEND)
    out spl,rl6
    out sph,rl7
```

```
ldi rl6,0xFB
out PORTA,rl6
clr rl6
out DDRA,rl6
out PORTC,rl6
ser rl6
out DDRC,rl6
ldi rl6,0x42
out ADMUX,rl6
ldi rl6,0xCF
out ADCSRA,rl6
sei
```

В регистре управления мультиплексором ADMUXубирается бит «левого» выравнивания результата. Это не связано с использованием прерывания по готовности АЦП, а сделано для демонстрации работы с двумя регистрами хранения результата АЦП: ADCHи ADCL.В регистре управления АЦП ADCSRAставится маска ADIEна 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:
    rjmpmain
```

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

4. При переходе по метке ADC_readyосуществляется обработка прерывания по готовности АЦП:

```
ADC_ready:
    cli
    in rl6,SREG
    in rl7,ADCL
    in rl8,ADCH
    lsr rl7
    lsr rl7
    clr r19
stl:
    lsl r18
    inc rl9
    cpi rl9,0x06
    brnemetl
    or rl7,r18
    out PORTC,rl7
    ldir17, 0xCF
    out ADCSRA, rl7
    out SREG, rl6
    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 стальные — запрещать.

После этого производится сохранение регистра SREGв РОН r16. Обнуление флага прерывания не производится, так как при использовании прерывания это делаетсяавтоматически..

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

Для использования восьми старших битов результата необходимо сместить спотерей0 и 1 битов регистр r17, приведя его к виду: 00XXXXXX(lsrr17; lsr r17), сместить два младших бита регистра r18 на 6 позиций влево, приведя его к виду: XX00 0000.

```
met1
    lsrr18
    inc r19
    cpi r19, 0x06
    brnemet1
```

После этого регистры rl7 и rl8 складываются (orr17, r18),а результат выводится в порт С. Далее АЦП перезапускается, регистр SREGвосстанавливается (outSREG, R16),производится глобальное разрешение всех прерываний **(**sei**),** а этом - выход из подпрограммы обработки прерывания (reti).

Форма представления результата:

Отчет должен содержать:

1. Цель работы.

2. Выполненное задание (на выбор пример 1 или 2) - работоспособность программы демонстрируется на стенде:

- листинг программы;
- комментарии.

3. Вывод по работе

Практическая работа № 15

Изучение работы сегментного и ЖК индикаторов под управлением МК Atmega8535.

Формируемые компетенции:

ПК 2.1. Создавать программы на языке ассемблера для микропроцессорных систем.

Цель работы: изучитьработу семисегментного и жидкокристаллического индикаторов при работе под управлением МК.

Выполнив работу, Вы будете:

уметь:

- составлять программы на языке ассемблера для микропроцессорных систем

Материальное обеспечение:

Не требуется

Теоретические сведения

В настоящее время подавляющее число промышленных приборов оснащаются цифровыми индикаторами для отображения различных величин. 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).

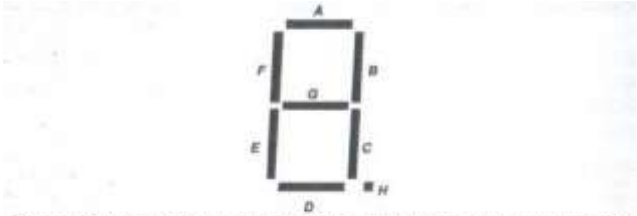


Рис. 1. Внешний вид и структура семисегментного индикатора

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.

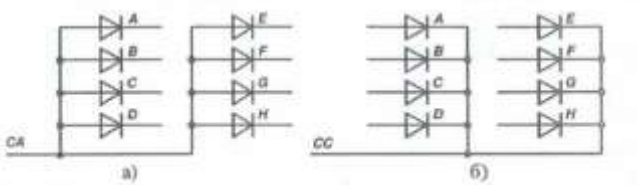


Рис. 2. Семисегментные индикаторы с общим анодом и общим катодом

Для того, чтобы зажечь, например, сегмент «C», необходимо катод соответствующего светодиода через токоограничивающий резистор присоединить к общему проводу.

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.

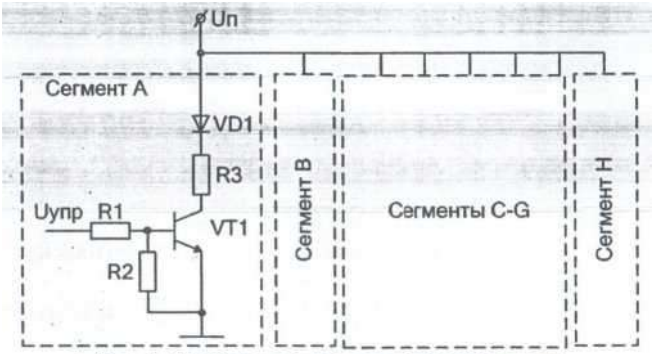


Рис. 3. Схема для управления индикатором

На общие аноды сегментов подается напряжение электропитания $U_п$. Катоды сегментов через токоограничивающий резистор и транзистор подключаются к общей шине. Очевидно, что для зажигания сегмента необходимо включить транзистор (для сегмента А - транзистор VT1).

Включение транзистора VT1 осуществляется подачей на его базу тока управления, который появляется при подаче напряжения управления $U_{упр}$ 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».

Динамическая индикация символов

Часто требуется осуществлять индикацию больших чисел, т.е. обращаться не к одному индикатору, ак нескольким.

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 подал соответствующую кодовую комбинацию (рис. 4).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 Atmega8535 - 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 А индикаторов HG1...HG4 объединяются в шу А, катоды сегмента В объединяются в шину В 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.

Контрольные вопросы

1. Каково назначение семисегментного индикатора?
2. Как конструктивно можно реализовать семисегментный индикатор?

3. В чем смысл динамической индикации?
4. С какой частотой необходимо подавать сигналы управления на индикатор при реализации динамической индикации?

Форма представления результата:

Отчет должен содержать:

1. Цель работы.
2. Ответы на контрольные вопросы
3. Вывод по работе

Лабораторная работа № 14

Разработка программы управления сегментным индикатором

Формируемые компетенции:

ПК 2.1. Создавать программы на языке ассемблера для микропроцессорных систем.

Цель работы: научиться составлять программы для вывода информации на семисегментные индикаторы под управлением МК.

Выполнив работу, Вы будете:

уметь:

- составлять программы на языке ассемблера для микропроцессорных систем

Материальное обеспечение:

Стенд «Микроконтроллер», ПК с установленной средой программирования и отладки AVRStudio

Теоретические сведения

Рассмотрим управление индикаторами на примере программ.

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

```
//-----------------------------------------------------------
;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 Atmega8535. После этого объявляется адрес начала сегмента кода.

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

```
set:
    ldi rl6,low(RAMEND)
    ldi rl7,high(RAMEND)
    out spl,rl6
    out sph,rl7
    clr r16
    out PORTC,rl6
    out PORTD,rl6
    ser rl6
    out DDRC,rl6
    out DDRD,rl6
    ldi rl6,0x0F
    out PORTA,rl6
    clr rl6                    ,
    out DDRA,rl6
    ldi rl6,0x01
    outPORTD,rl6
```

Врегиструказателястеказаписываетсяадресвершиныстекакоторыйответствуетконцупамятиданных(ldirl6, low(RAMEND);ldirl7, high.(RAMEND) ; outspl,rl6;outsph, rl7). Это необходимо для правильной работы программы при использовании подпрограмм и переходов, порт С в программе будет подключен к катодам индикатора, поэтому он инициализируется на вывод, младший бит порта Dпо заданию управляет подачей напряжения питания на общие аноды индикатора, поэтому порт инициализируется . вывод, биты РАЗ...РАО порта ввода/вывода А инициализируются на ввод информации.

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

```
in:
    in rl6,PINA
    andirl6,0x0F
```

Производится опрос всего порта (inrl6,PiNA),после чего путем побитового сложения результата на число 0xFF(andirl6,0x0F)происходит выделение младших значащих бит порта А.

4. По результату, считанному с порта А, производится формирование адреса Flash-памяти, по которому хранится соответствующий символ. Поскольку во Flashпамяти 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 Flashв Z-регистре (r31:r30) указывается адрес памяти, а затем командой 1pmпроисходит считывание данных:

```
    mov r30,rl6
    ldir31,0x02
    lpm rl6,Z
```

5. После считывание данных из Flashони выводятся на порт С, соединенный с индикатором:

```
    out PORTC,rl6
    rjmpmain
```

Далее программа зацикливается (rjmpmain)для постоянного опроса порта ввода/вывода А.

1. 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 соответствует выводимому на индикатор символу в шестнадцатеричном коде. Таблица цифр от 0 до Fприведена в табл. 1.

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

Цифра	Двоичный код	Шестнадцатеричный
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	0b01110111	0x77
b	0b01111100	0x7C
C	0b00111001	0x39
d	0b01011110	0x5E
E	0b01111001	0x79
F	0b01110001	0x71

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.

```
t1:                 ;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
t2:
   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
t3:
   clr r16
   out PORTC,r16    ;обнуляется PORTC
   out PORTD,r17    ;обнуляется PORTD
   rjmp main        ;осуществляется переход на main

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

```
   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
```

Рассмотрим программу более подробно.

1. Сначала производится подключение библиотеки используемого контроллера Atmega8535. После этого объявляется адрес начала сегмента кода:

```
include"m8535def.inc”
seg
rg$0
```

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

```
set:
ldi rl6,low(RAMEND)
ldi rl7,high(RAMEND)
out spl,rl6
out sph,rl7
clr r16
out PORTC,r16
out PORTD,rl6
ser rl6
out DDRC,rl6
out DDRD,rl6
ldi rl7,0x01
outPORTD,rl7
```

После инициализации устройств в регистр r17 записывается число 0x01 (ldirl7,0x01),после чего содержимое регистра выводится на порт управления разрядами индикатора (outPORTD,r17).Далее в r17 будет находиться значение, соответствующее включенному состоянию PORTD.

3. По метке maincначала производится опрос включенного состояния PORTDпутем опроса регистра r17:

```
main:
cpi rl7,0x01
breq HG1
cpi rl7,0x02
breq HG2
cpi rl7,0x04
breq HG3
cpi rl7,0x08
breqHG4
```

В зависимости от состояния r17 производится переход на метки HG1...HG4. Так если в данный момент работает разряд HG1 (cpiri7,0x0i),то осуществляется переход на метку HG1 (breqHGI).

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

```
HG1:
ldir31,0x02
ldir30,0x04
rjmpmetl
HG2:
ldi r31,0x02
ldi r30,0x03
rjmpmetl
HG3:
ldi r31,0x02
ldir30,0x02
rjmpmetl
HG4:
ldi r30,0x01
rjmpmetl
metl:
lpm rl6,Z
outPORTC,rl6
```

После записи адреса символа осуществляется переход на метку metl, по которой происходит считывание данных из Flash-памяти вPOHrl6 (lpmrl6,z) и вывод их на PORTC(outPORTC,rl6).

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

```
clr r16
met2:
inc rl6
cpi rl6,0xFF
brnemet2
```

Для реализации задержки сначала содержимое РОН r16 обнуляется (clrr16),после чего происходит его инкремент (incrl6)с последующим сравнением с уставкой(cpir16, 0xFF).При значении r16, меньшем уставки, происходит возврат на метку met2 (brnemet2).При достижении уставки программа следует дальше.

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

```
lslr17
cpir17,0x10
brne met3
ldi rl7,0x01
met3:
clr rl6
out PORTC,rl6
out PORTD,rl7
rjmpmain
```

С этой целью содержимое rl7 последовательно сдвигается на одни разряд влево (lslr17),после чего производится выдача его содержимого на PORTD(outPORTD, r17).Однако при достижении r17 значения 0x10 (cpirl7,0xl0)в него необходимо записать значение 0x01 (ldir17,0x01)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 (clrrl6; outPORTC,r16).После этого программа зацикливается (rjmpmain).

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
```

Задание на выполнение

Составить программу реализации динамической индикации для заданной частоты. Данные предварительно записать в ОЗУ, выводить их на индикацию по адресу записи.

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	Чяб	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	–

Форма представления результата:

Отчет должен содержать:

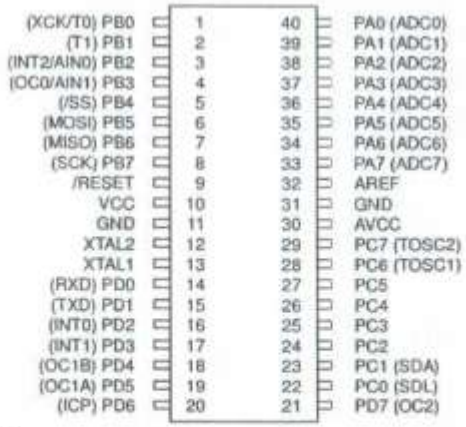
1. Цель работы.
2. Выполненное задание с комментариями (работоспособность программы демонстрируется на стенде):
   - исходное задание;
   - функциональную схему;
   - представить расчет частоты динамической индикации и кодов данных;

- листинг программы;
- дизассемблированную программу;
- алгоритм;
- -представить таблицу значений стека во время исполнения программы.

3. Вывод по работе

ПРИЛОЖЕНИЕ 1

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

ПРИЛОЖЕНИЕ 2

Регистры ввода-вывода МК Atmega8535

ПРИЛОЖЕНИЕ 3

Таблица векторов прерываний МК Atmega8535

№ вектора прерываний	Адрес	Источник	Примечание
1	$000	RESET	Сброс по выводу RESET и сторожевому таймеру (Hardware Fin, 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 Even)
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 USAFT (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 Comperator)
18	$011	TWI	Последовательный двухпроводной интерфейс Two-wire Serial Interface
19	$012	INT2	Внешнее прерывание External Interrupt Request 2
20	$013	TIMER0 COMP	Совпадение P при сравнении таймера/счетчика T0 Timer/Counter0 Compare Match
21	$014	SPM_RDY	Готовность Store Program. Memory Reacy

ПРИЛОЖЕНИЕ 4

Директивы ассемблера МК Atmega8535

Директива	Описание
BYTE	Зарезервировать байт под переменную
CSEG	Сегмент кодов
DB	Задать постоянным(и) байт(ы) в памяти
DEF	Задать символическое имя регистру
DEVICE	Задать для какого типа микроконтроллера компилировать
DSEG	Сегмент данных
DW	Задать постоянное(ые) слово(а) в памяти
EQU	Установите символ равный выражению
ESEG	Сегмент EEPROM
EXIT	Выход из файла
INCLUDE	Включить исходный код из другого файла
LIST	Включить генерацию .lst - файла
NOLIST	Выключить генерацию .lst - файла
ORG	Начальный адрес программы
SET	Установите символ равный выражению

ПРИЛОЖЕНИЕ 5

Система команд МК Atmega8535

Арифметические и логические команды

Мнемо-ника	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 dЄ{24,26,28, 30} 0≤K≤63	Сложить непосредственное значение 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 непосредственное значение	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 непосредственное значение с заемом	Rd←Rd-K-C	Z, C, N, V, H	1
SBIW	Rd, K dЄ{24,26,28, 30} 0≤K≤63	Вычесть непосредственное значение из слова	Rdh:Rdl←Rdh:Rdl-K	Z, C, N, V	2
AND	Rd, Rr 0≤d≤31 0≤r≤31	Выполнить логическое AND	Rd ← Rd • Rr	Z, N, V	1
ANDI	Rd, K 16≤d≤31 0≤k≤255	Выполнить логическое AND	Rd ← Rd • K	Z, N, V	1
OR	Rd, Rr 0≤d≤31 0≤r≤31	Выполнить логическое OR	Rd ←Rd v Rr	Z, N, V	1
ORI	Rd, K 16≤d≤31 0≤K≤255	Выполнить логическое OR с непосредственным значением	Rd ←Rd v K	Z, N, V	1

Мнемо-ника	Операнды	Описание	Операция	Флаги	Кол-во циклов
ROL	Rd 0≤d≤31	Сдвинуть влево через перенос	Rd(0) ←C, Rd(n+1)←Rd(n), C←Rd(7)	Z,C,N,V,H	1
ROR	Rd 0≤d≤31	Сдвинуть вправо через перенос	Rd(7) ←C, Rd(n)←Rd(n+1), C ← Rd(0)	Z,C,N,V	1
ASR	Rd 0≤d≤31	Арифметически сдвинуть вправо	Rd(n)←Rd(n+1), n=0...6, Rd(0)←C	Z,C,N,V	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
SBI	P,b 0≤P≤31 0≤b≤7	Установить бит в регистр I/O	I/O(P,b)← 1	Нет	2
CBI	P,b 0≤P≤31 0≤b≤7	Очистить бит в регистре I/O	I/O(P,b)← 0	Нет	2
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
BSC		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

ия	Флаги	Кол-во циклов
Rr	Z, N, V	1
-Rd	Z, C, N, V	1
- Rd	Z, C, N, V, H	1
v K	Z, N, V	1
FF - K)	Z, N, V	1
+ 1	Z, N, V	1
- 1	Z, N, V	1
Rd	Z, N, V	1
⊕ Rd	Z, N, V	1
FF	нет	1
r	Z, C, N, V, H	1
-C	Z, C, N, V, H	1
,	Z, C, N, V, H	1

ия	Флаги	Кол-во циклов
Rd(n), ←Rd(7)	Z,C,N,V,H	1
8(n+1), ←Rd(0)	Z,C,N,V	1

Мнемоника	Операнды	Описание	Операция	Флаги	Кол-во циклов
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
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
STS	k, Rr 0≤d≤31 0≤k≤65535	Загрузить непосредственно в ОЗУ	(k) ←Rr	Нет	3
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

Мнемоника	Операнды	Описание	Операция	Флаги	Кол-во циклов
CLI		Очистить флаг глобального прерывания	I←0	I	1
SES		Установить флаг знака	S← 1	S	1
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
ELPM		Расширенная загрузка из памяти программ в регистр RO	R0← (Z+RAMPZ)	Нет	3
MOV	Rd,Rr 0≤d≤31 0≤r≤31	Копировать регистр	Rd←Rr	Нет	1
LDI	Rd,k 16≤d≤31 0≤k≤255	Загрузить непосредственное значение	Rd←K	Нет	1
LDS	Rd,k 0≤d≤31 0≤k≤65535	Загрузить из ОЗУ	Rd←(k)	Нет	3

Мнемоника	Операнды	Описание	Операция	Флаги	Кол-во циклов
ICALL		Вызвать подпрограмму косвенно	PC←Z	Нет	3
CALL	k 0≤k≤64K	Выполнить длинный вызов подпрограммы	PC←k	Нет	4
RET		Вернуться из подпрограммы	PC ← STACK	Нет	4
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	½/3
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	½/3
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	½/3
SBIC	P, b 0≤P≤31 0≤b≤7	Пропустить, если бит в регистре I/O очищен	if I/O P(b)=0 then PC ← PC + 2 (or 3)	Нет	½/3
SBIS	P, b 0≤r≤31 0≤b≤7	Пропустить, если бит в регистре I/O установлен	If I/O P(b)=1 then PC ←PC + 2 (or 3)	Нет	½/3
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	Перейти, если равно или	if Rd<Rr(C=0) then PC←PC+ k+ 1	Нет	1/2

Мнемоника	Операнды	Описание	Операция	Флаги	Кол-во циклов
ST	-Y, Rr 0≤r≤31	Записать косвенно с предекрементом	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 предекрементом	Z←Z-1, (Z) ← Rr	Нет	2
STD	Z+q, Rr 0≤r≤31 0≤q≤63	Записать косвенно со смещением	(Z+q)←Rr	Нет	2
LPM		Загрузить байт из памяти программ	R0 ← (Z)	Нет	3
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	Rr 0≤r≤31	Загрузить в регистр из стека	Rr ← 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
IJMP		Перейти косвенно	PC←Z	Нет	2
JMP	k 0<k<4M	Перейти	PC←k	Нет	3
RCALL	k -2K<k≤2K	Вызвать подпрограмму относительно	PC ← PC + k + 1	Нет	3

Мнемоника	Операнды	Описание	Операция	Флаги	Кол-во циклов
BRLO	k -64≤k≤+63	Перейти, если меньше (без знака)	if Rd<Rr (C=1) then PC ←PC + k + 1	Нет	1/2
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